UNIVERSIDAD DE OVIEDO

ASTURIAS
CAMPUS DE EXCELENCIA
INTERNACIONAL
AD FUTURUM

MÁSTER UNIVERSITARIO EN INGENIERÍA WEB

# TRABAJO FIN DE MÁSTER

# "BILROST: Interconexión de objetos inteligentes a través de redes sociales"

Daniel Meana Llorián

**El/la tutor/a autoriza la defensa del Trabajo Fin de Máster**

**Fdo. Dña. B. Cristina Pelayo García-Bustelo**

Oviedo, Mayo de 2016

*A Rubén y Lucía,*
*dos piezas fundamentales de mi vida.*

# Agradecimientos

Este proyecto fin de máster no habría sido posible sin el apoyo de todas las personas que me rodean, especialmente mi familia y pareja que han tenido que soportar muchos momentos de estrés y de mal humor.

Gracias a todos los compañeros del máster, han sido dos buenos años juntos donde han surgido nuevos amigos y a los compañeros del grado que, aunque el contacto no se mantenga con todos, no se olvidan ya que algo han aportado en su momento para poder avanzar hacia delante.

También he de agradecer a los compañeros de trabajo que he tenido durante dos meses en CSC que me han sabido aconsejar sobre mi futuro, sin esa experiencia no estaría donde estoy ahora.

Muchas gracias a todas las personas con las que he coincidido en el Laboratorio de Tecnologías Orientadas a Objetos que han hecho más ameno todo el tiempo que allí he pasado, tanto realizando este proyecto como trabajando para el grupo de investigación MDE-RG.

Gracias a Juan Manuel Cueva y a B. Cristina Pelayo por haberme dado la oportunidad de entrar al campo de la investigación con ellos, gracias a Cristian González por haberme acogido como uno más cuando era completamente nuevo y gracias a Jordán Pascual y Vicente Díaz por darme la oportunidad de realizar publicaciones junto a ellos.

También agradecer a los participantes de las pruebas su granito de arena a este proyecto.

Y, por último, gracias a los profesores del Máster y del Grado, en mayor o menor medida, pues gracias a ellos he podido llegar al punto en el que estoy.

# Resumen

En la actualidad, aunque la gente no sea consciente de ello, el mundo está repleto de objetos inteligentes con conexión a Internet como los teléfonos inteligentes o *smartphones*, las tabletas o *tablets*, los dispositivos *wearables* (pulseras, colgantes, relojes,…), microcontroladores como el Arduino, y otros muchos dispositivos. Sin embargo, estos dispositivos no están todos interconectados. Esa interconexión abriría una gran variedad de nuevas posibilidades derivadas del trabajo colaborativo entre los dispositivos. Lograr una interconexión de estos dispositivos para que puedan alcanzar objetivos comunes es un tema importante a abordar.

Existen diversos problemas en la creación de aplicaciones que interconecten dispositivos y que se han identificado en este proyecto fin de máster. Estos problemas son la necesidad de una infraestructura hardware que posibilite la comunicación entre dispositivos, la heterogeneidad de los objetos de Internet de las Cosas que hace que desarrollar una solución genérica sea muy difícil puesto que cada uno puede usar lenguajes de programación distintos, sistemas operativos diferentes sobre arquitecturas variadas, y la dificultad de desarrollar software especializado para gente que no tenga conocimientos sobre desarrollo de software.

En este trabajo fin de máster se propone la creación de un Lenguaje de Dominio Específico, al que se ha llamado Bilrost Specific Language (BSL), a través de la aplicación de la Ingeniería Dirigida por Modelos, cuyo fin es la definición de dispositivos compuestos por actuadores y/o sensores, que serán capaces de comunicarse con otros dispositivos o con personas usando para ello, las redes sociales. Por tanto, la principal contribución de este trabajo fin de máster es la interconexión de objetos heterogéneos y ubicuos usando como canal de comunicación las redes sociales y la posibilidad de que las personas interactúen con estos objetos mediante las mismas redes.

Gracias al uso del BSL, se facilita la generación de aplicaciones para distintas plataformas, que puedan ser desplegadas en los objetos definidos. Para ello se ha desarrollado un sistema, Bilrost, compuesto por dos editores, uno textual y otro gráfico, que permiten a los usuarios crear modelos usando la sintaxis del BSL o usando componentes visuales equivalentes a la sintaxis BSL. Este sistema transforma el modelo gráfico al modelo textual, cuya sintaxis está definida mediante una gramática libre de contexto, y que el sistema procesa, a través de un analizador léxico y sintáctico, para generar proyectos de aplicaciones en el lenguaje de programación de la plataforma del dispositivo. Estos proyectos generados ya tendrán implementada toda la lógica relacionada con la conexión con las redes sociales.

Para llegar a esta solución, se ha realizado un estudio de los conocimientos científico-técnicos relacionados, como el Internet de las Cosas, los objetos inteligentes, los Lenguajes de Dominio Específico, la Ingeniería Dirigida por Modelos y las Redes Sociales. Además, también fueron necesarios otros conceptos para investigaciones paralelas realizadas en el marco de Internet de las Cosas durante el transcurso de este trabajo fin de máster, como la lógica difusa; *Smart Cities, Smart Towns* y *Smart Homes*; y visión por computador. También se han desarrollado varios prototipos para abordar el desarrollo de soluciones de Internet de las Cosas, un prototipo basado en sensores de gases análogo a una nariz inteligentes y un prototipo de visión por computador con notificaciones a usuarios vía email.

Como resultado de todo el proceso de investigación realizado se han escrito 9 publicaciones de las cuales 6 son artículos de revista y 3 artículos de congreso. De estas publicaciones, hay 2 artículos de revista correspondientes a este proyecto fin de máster y 1 artículo de congreso.

**Palabras clave**: Internet de las Cosas; Objetos Inteligentes; Redes Sociales; Ingeniería Dirigida por Modelos; Lenguajes de Dominio Específico; Comunicación Persona-Máquina.

# Abstract

Currently, there are many Smart Objects with Internet capabilities in our environment although people do not realise it. These objects can be smartphones, tablets, wearable devices like smartwatch or smartbands, microcontrollers like Arduino, or many other devices capable of establishing a connection to the Internet. H However, not all devices are interconnected with each other, which could open a new wide range of possibilities based on the collaborative work between devices. An important issue to address is this interconnection, because interconnected devices would be capable of achieving common goals.

There are several problems related with the creation of applications that allow the interconnection of devices, and we describe them in this Master's Thesis. A problem is the necessity of an infrastructure that enables the communication between devices. Another one is the diversity of heterogeneous objects from the Internet of Things, which makes harder the development of a generic solution because devices have different programming languages, use different operative systems, and are based on different architectures. And another problem is the difficulty inherent in the software development process which requires people to have technical knowledge.

In this Master's Thesis, we propose the creation of a Domain Specific Language (DSL), which we called Bilrost Specific Language (BSL), by applying Model Driven Engineering. The aim of the BSL is the definition of devices composed by actuators and/or sensors, which will be able to communicate with other devices or with people through Social Networks.

Thus, the major contribution of this Master's Thesis is the interconnection of heterogeneous and ubiquitous objects by using Social Networks as communication channel, and the interaction between people and these objects through the same networks.

By using the BSL, we facilitate the generation of applications for different platforms and they can be deployed in the defined devices. For this purpose, we developed a system, called Bilrost, composed by two editors, a textual and a graphic editor, which allow users to create models by using the BSL syntax or by using graphic components which represent the same aspects that the textual syntax. This system translates the graphic model to textual model, whose syntax is defined by a free-context grammar. The system is able to process source code that uses the syntax defined in the grammar through a lexical and syntactical parser. The generated projects will already contain the required logic to establish the communication with the Social Networks.

To achieve this solution, we made an analysis of the state of the art related with the proposal. We analysed the Internet of Things, Smart Objects, Domain Specific Languages, Model Driven Engineering, and Social Networks. Moreover, we also analysed other concepts related with parallel researches in the field of the Internet of Things. These concepts were the Fuzzy logic; Smart Cities, Smart Town and Smart Homes; and Computer Vision. By other side, we also developed several prototypes in order to address the development of solutions for the Internet of Things, a prototype based on gas sensors in order to simulate an intelligent nose and a prototype about computer vision.

The results of the research process were 9 publications: 6 papers for journals, and 3 papers for congress. Three of these publications were focused in the proposal of this Master's Thesis: 2 papers for journals and 1 paper for a congress.

**Keywords**: The Internet of Things; Smart Objects; Social Networks; Model Driven Engineering; Domain Specific Language; Human-Machine communications.

# Tabla de contenidos

# Tabla de figuras

# Tabla de tablas

# Capítulo 1. Introducción

En este primer capítulo se introducirá este trabajo fin de máster haciendo hincapié en que temas se abordarán y las causas que han motivado su realización. Para ello, primero se realizará una revisión y análisis de las diversas tecnologías, los problemas que existen y motivan la realización de esta investigación, y la hipótesis que se propone como solución a dichos problemas.

Actualmente, todas las personas han tenido contacto con algún objeto inteligente o *Smart Object* aunque no sean conscientes de lo que son. La mayor parte de las tecnologías que se usan a diario están compuestas por sensores y/o actuadores y cierta inteligencia que les permiten realizar ciertas acciones de manera automática o semiautomática, en función de diferentes condiciones o ante ciertos eventos. Ejemplos de estos objetos inteligentes son los dispositivos que diariamente se usan como los teléfonos móviles más modernos, también llamados teléfonos inteligentes o *smartphones,* las tabletas, los dispositivos *weareable,* e incluso las nuevas televisiones inteligentes o *SmartTVs*, los coches inteligentes o cualquier otro dispositivo que posee cierta inteligencia y pueda reaccionar ante algún tipo de evento. Estos dispositivos inteligentes están en continuo crecimiento como demuestra (Fundación Telefónica, 2016).

Internet de las Cosas o Internet de los Objetos (IoT) se basa en la interconexión de objetos heterogéneos y ubicuos a través de internet, normalmente con cierta inteligencia, con el fin de lograr objetivos comunes, crear aplicaciones o suministrar servicios (Vermesan & Peter Friess, 2013). Hay muchas investigaciones en muchos ámbitos distintos que están relacionadas con Internet de las Cosas como investigaciones en el campo del cuidado de la salud (Pang et al., 2015), sistemas de rehabilitación (Fan, Yin, Member, Zeng, & Wu, 2014), sistemas que ayudan a personas con discapacidades (Domingo, 2012) o entornos inteligentes como por ejemplo: casas inteligentes donde el Internet de las Cosas puede ayudar a ahorrar energía (Lee & Kim, 2012), edificios inteligentes (Zheng, Qin, Gao, Duan, & Zhang, 2010), ciudades inteligentes (Caragliu, Del Bo, & Nijkamp, 2011; Mitchell, Villa, Stewart-Weeks, & Lange, 2013; Sanchez et al., 2014; Vienna University of Technology, 2015), pueblos inteligentes (Song, 2013) o incluso planeta inteligente (Howe et al., 2010).

Sin embargo, el uso de soluciones basadas en el Internet de las Cosas no está todavía muy extendido debido a la necesidad de diversos componentes hardware como sensores, actuadores o incluso alguna plataforma para desplegar las soluciones que interconecten los objetos inteligentes. Mayoría de las soluciones están basadas en usar una arquitectura de servicios web, frecuentemente REST, junto a un servidor central como hacen (González García, Pelayo G-Bustelo, Pascual Espada, & Cueva-Fernandez, 2014; D. Guinard, Fischer, & Trifa, 2010; Dominique Guinard, Trifa, Pham, & Liechti, 2009).

A lo largo de este trabajo fin de master se presentará una propuesta de un nuevo sistema de comunicación entre objetos inteligentes que evita la necesidad de tener muchos objetos inteligentes y sin la necesidad de tener una infraestructura donde desplegar las aplicaciones que se encargan de interconectar los objetos. Nuestra propuesta se basa en el uso de redes sociales como canal de comunicación entre objetos inteligentes y donde, además, pueden intervenir las personas como un actor más.

Las redes sociales están presentes en la vida diaria de muchas personas. La mayoría de la gente tiene el hábito de compartir muchos aspectos de sus vidas a través de ellas, qué están haciendo en cada momento, qué están comiendo, con quién están, qué piensan sobre un tema concreto, entre muchos otros aspectos. Pero, además, en el entorno profesional, las redes sociales pueden ser también un canal para promocionar los productos o servicios de una empresa y ganar así clientes.

La combinación de las redes sociales y el Internet de las Cosas no es algo que surja por casualidad. En (Atzori, Iera, & Morabito, 2014), menciona que científicos de Ericsson observaron que las personas son capaces de familiarizarse mejor con las tecnologías de Internet de las Cosas si existe una analogía entre cómo funcionan esas tecnologías y sus hábitos en las redes sociales como Facebook o Twitter. Debido a esto, en este trabajo fin de master se propone la combinación de las redes sociales e Internet de las Cosas con el objetivo de interconectar objetos inteligentes.

Por otro lado, según (Miranda et al., 2015), la integración de tecnologías de Internet de las Cosas en las vidas de las personas es un campo con mucho margen de mejora, y además, un requisito para poder alcanzar esa integración es hacer que los objetos inteligentes sean accesibles (Mashal et al., 2015). Este trabajo fin de máster, propone alcanzar esa integración y accesibilidad mediante la incorporación de objetos inteligentes en las redes sociales que las personas usan.

Un objetivo de Internet de las Cosas es expandir las comunicaciones existentes persona-persona hacia comunicaciones persona-cosas y cosas-cosas conectando el mundo físico y el mundo virtual (Mashal et al., 2015). En este trabajo fin de master se investigará la posibilidad de utilizar las redes sociales para permitir estas comunicaciones. Además, como ya se menciona en (Blackstock, Lea, & Friday, 2011), el uso de las redes sociales permite establecer un punto de entrada para que otros usuarios o aplicaciones puedan interactuar con los objetos disponibles en la red.

## 1.1. Problemas encontrados

Como ya se ha mencionado, el objetivo de Internet de las Cosas es interconectar objetos, sin embargo, existen una serie de problemas que aparece al intentar crear aplicaciones que interconecten objetos de internet de las cosas. Estos problemas son: la necesidad de hardware que conectar o donde desplegar las herramientas que se encargan de la intercomunicación, la gran variedad o heterogeneidad de objetos que existe, y la dificultad existente en el desarrollo de aplicaciones.

### 1.1.1. Necesidad de hardware

No todos los usuarios pueden tener acceso a las herramientas necesarias para conectar objetos o ni siquiera saben que disponen de ellos. Además, al tratarse, normalmente, de objetos físicos, no todo el mundo tiene el poder adquisitivo suficiente para adquirir muchos objetos que interconectar y para adquirir los medios que faciliten esa interconexión. Muchas de las investigaciones actuales en el campo de la interconexión de objetos del Internet de las Cosas necesitan, al menos, contar con un servidor que se encarga de realizar las comunicaciones como se verá en un capítulo posterior.

El usar hardware implica la necesidad de conocimientos sobre ese hardware haciendo más difícil la creación de aplicaciones que usen objetos de Internet de las Cosas. En esta propuesta se intentará solventar en parte estos problemas mediante el uso de redes sociales que las personas suelen usar diariamente. Además, como se menciona en (Atzori, Iera, & Morabito, 2010), científicos de Ericsson han afirmado que una analogía entre las redes sociales y las tecnologías de Internet de las Cosas hacen que las personas vean más sencillas las tecnologías de Internet de las Cosas.

### 1.1.2. Heterogeneidad de los objetos

Además de estos problemas, también existe uno que se encuentra en la propia definición de Internet de las Cosas, la heterogeneidad de los objetos.

Como se menciona en (Gama, Touseau, & Donsez, 2012), la heterogeneidad es uno de los problemas que dificulta la comunicación entre los objetos. Para solventarlo, es necesario que todos los objetos que se quieran interconectar usen un sistema común con el fin de que la comunicación sea entendible por todas las partes. Pero otro problema derivado de este sistema común y también relacionado con la heterogeneidad de los objetos es que cada uno usa sus propias tecnologías de desarrollo y el sistema creado para un objeto no tiene por qué ser compatible para otro objeto. Por tanto, además de necesitar un sistema para comunicar objetos en el que todas las partes entiendan los mensajes, ese sistema debe funcionar con tecnologías distintas.

Es decir, el problema de la heterogeneidad de los objetos se debe a las distintas implementaciones que hay que realizar para poder comunicar varios objetos entre sí debido a la existencia de diferentes sistemas operativos. Además, debido a la inexistencia de un estándar de comunicación entre objetos, cada desarrollador usa sus propios sistemas para establecer esas comunicaciones sin que otros usuarios puedan aprovechar los objetos de esos desarrolladores debido a que, aunque fuese objetos públicos, existiría una falta de entendimiento entre ellos debido a la inexistencia de interfaces, estándares o protocolos comunes (Dominique Guinard & Trifa, 2009).

En este proyecto se pretende afrontar el problema de la heterogeneidad de los objetos planteando la creación de aplicaciones adaptadas a cada dispositivo que usen el mismo canal de comunicación con mensajes que siguen una misma estructura.

### 1.1.3. Dificultades en el desarrollo de aplicaciones

Otro de los problemas de Internet de las Cosas es el desconocimiento sobre como interconectar objetos durante el desarrollo de las aplicaciones. El disponer de libertad a la hora de crear esa interconexión hace posible que los desarrolladores varíen el formato de los mensajes o no establezcan la comunicación de la forma más adecuada. Además, es necesario contar con amplios conocimientos de los lenguajes de programación de todos los objetos que se quieren comunicar, así como de conocimientos sobre el sistema de comunicación.

Por otro lado, los usuarios inexpertos, sin conocimientos técnicos estarían privados de interconectar sus propios objetos y crear aplicaciones que hagan uso de ellos teniendo que conformarse con soluciones ya existentes en el mercado.

Una posible solución a estos problemas es ofrecer un sistema que permite generar esas aplicaciones sin necesidad de conocimientos avanzados. Así, cualquier usuario podría generar sus aplicaciones según sus necesidades y asegurando que los mensajes que se usan para interconectar los objetos mantengan una homogeneidad. Para lograr esta solución, en esta investigación, se usarán los lenguajes de dominio específico (DSL – *Domain Specific Language*). De esta manera se abstrae la implementación de estas aplicaciones que conecten objetos.

## 1.2. Solución propuesta

La propuesta de este trabajo fin de máster, denominada Bilrost, es permitir a los usuarios generar aplicaciones para objetos inteligentes que permita la comunicación entre objetos inteligentes utilizando redes sociales a través de un Lenguaje de Dominio Específico (DSL – *Domain-Specific Language*).

Por otro lado, al lograr la comunicación entre objetos inteligentes y personas usando redes sociales, se consigue superar uno de los principales problemas de Internet de las cosas, la interoperabilidad entre objetos heterogéneos y ubicuos, y otros sistemas diferentes (Mashal et al., 2015).

La **hipótesis** de trabajo que esta investigación intentará confirmar es la siguiente:

> *Es posible facilitar la **creación de aplicaciones** que **integren objetos inteligentes** de Internet de las Cosas **en las redes sociales** de personas, permitiendo la interaccionar entre sí, y la interacción de las personas con los objetos inteligentes*.

# Capítulo 2. Fijación de Objetivos

En este capítulo se presentarán los objetos que pretende alcanzar este trabajo fin de máster partiendo de los problemas identificados en el capítulo anterior y de la hipótesis planteada. Además, también se expondrán los posibles ámbitos de aplicación de la propuesta realizada.

## 2.1. Objetivos de la investigación

El objetivo principal de esta investigación es facilitar el desarrollo de aplicaciones que interconecten objetos inteligentes de Internet de las Cosas de manera sencilla a través de redes sociales, es decir, sin la necesidad de disponer de un servidor intermedio que dependa del usuario. Para lograr este objetivo principal se deberán cumplir varios de manera progresiva puesto que cada objetivo depende del anterior:

- **Integrar objetos inteligentes en una red social de personas**: Para cumplir este objetivo se debe conseguir que los objetos inteligentes sean capaces de leer y procesar información de una red social de personas como puede ser Twitter o Facebook y capaces de publicar información en la misma red social. Para cumplir este objetivo se debe conseguir que los objetos inteligentes sean capaces de leer y procesar información de una red social usada por personas y también, que sean capaces de publicar información en la misma red social. Para abordar este objetivo, se usará la red social Twitter por ser una de las de uso más extendido y de las que más facilidades otorga a los desarrolladores.
- **Permitir la interacción de personas con los objetos inteligentes conectados a la red social**: Al cumplir el objetivo anterior se conseguirá que los objetos inteligentes sean capaces de leer y procesar información de una red social, y capaces de publicar en ella. El siguiente paso es que las personas sean capaces de interactuar con estos objetos aprovechando el objetivo cumplido. Para cumplir este objetivo se debe analizar la posibilidad de que, a través de mensajes en la red social, los usuarios puedan describir acciones que deseen que realicen los objetos inteligentes. Y que estos últimos, gracias al objetivo anterior, puedan procesarlos y reaccionar.
- **Interconectar objetos usando como único canal de comunicaciones una red social**: Una vez que se ha logrado la integración de objetos inteligentes en una red social y es posible que las personas interactúen con ellos, se presenta el siguiente objetivo, hacer que los propios objetos interactúen entre sí, es decir, se comuniquen a través de mensajes en la red social. Para cumplir este objetivo se propone la creación de un sistema de reglas que permitan automatizar las tareas de publicación y procesamiento de información de la red social, de manera que, sin la intervención humana, un objeto sea capaz de procesar información procedente de otro objeto y reaccionar, por ejemplo, publicando mensajes que interaccionen con otro objeto.
- **Definición de un Lenguaje de Dominio Específico**: Tras cumplir los anteriores objetivos, se tendrá el conocimiento necesario para poder definir un Lenguaje de Dominio Específico, que represente a los objetos inteligentes integrados en la red social capaces de interactuar entre sí y con personas. Este DSL será definido a través de una gramática basada en un metamodelo.

- **Facilitar la generación de aplicaciones que interconecten objetos inteligentes**: Una vez logrados todos los objetivos, es necesario hacer que la generación de aplicaciones que aprovechen la comunicación entre objetos a través de la red social sea sencilla. Para alcanzar este objetivo, se propone la creación de un sistema que, a partir del DSL definido en el objetivo anterior, y mediante la aplicación de MDE, sea capaz de generar el código de las aplicaciones que interconecten los objetos inteligentes a través de redes sociales.
- **Generar aplicaciones que soportar diferentes plataformas**: Por último, tras haber logrado la creación de aplicaciones que interconecten objetos inteligentes a través de redes sociales, el siguiente paso será habilitar la generación de aplicaciones en varias plataformas. Para ello, se propone integrar en el DSL definido, la opción de elegir la plataforma destino, de manera que el código generado a partir de ese DSL sea para esa plataforma concreta.

## 2.2. Posibles Ámbitos de Aplicación

Entre posibles ámbitos de aplicación, teniendo en cuenta las limitaciones existentes, se encuentran cualquier ámbito donde es aplicable Internet de las Cosas como, por ejemplo:

- **Aplicación en entornos domésticos**: El concepto de aplicar Internet de las Cosas en los hogares se conoce como *Smart Homes* y es un ámbito donde sería aplicable esta propuesta. Por ejemplo, un usuario podría generar aplicaciones para objetos inteligentes que se encarguen de la calefacción, del aire acondicionado y de las luces y para su propio teléfono inteligente de tal manera que cuando el usuario abandone el hogar se apaguen las luces y cuando el usuario está llegando a casa se enciendan las luces o que cuando la temperatura sea baja se encienda la calefacción y cuando la temperatura sea alta se encienda el aire acondicionado. Además, el usuario, de manera remota, podría encender la calefacción o el aire acondicionado publican un mensaje en las redes sociales para que se realice esa acción.
- **Aplicación en entornos profesionales**: En entornos profesionales, la aplicación de este sistema es más delicada debido a la naturaleza pública de la línea temporal de muchas redes sociales como Twitter. Sin embargo, puede ser útil para informar a sus clientes a través de las redes sociales de eventos que ocurran y puedan ser detectados de manera automática. Por ejemplo, cuando un nuevo producto llega a un negocio, un lector de códigos de barras puede publicar que ha llegado ese producto al negocio y además invocar las acciones correspondientes para enviar notificaciones a través de más medios como email considerando al email un actuador más.
- **Aplicación en sistemas de ahorro energético**: Mediante la automatización es posible el ahorro energético. Por ejemplo, mediante el apagado de las luces si no hay presencia humana. De esta manera un sensor de presencia podría identificar que no hay nadie en una sala y notificar al actuador encargado de controlar las luces para que este las apague.
- **Aplicación en sistemas distribuidos**: Todos los ejemplos anteriores usan dispositivos relacionados, sin embargo, el sistema propuesto sería capaz de utilizar objetos ubicuos. Por ejemplo, se podría usar para que sensores de terremotos ubicados en diferentes partes del mundo notificasen un terremoto a través de redes sociales. Y como resultado de esto se accione una alarma en algún centro que se dedique a controlar la actividad sísmica.

- **Aplicación en control de sistemas a través de dispositivos móviles**: Como último ejemplo de aplicación podría ser el control de sistemas remotos a través de dispositivos móviles como ya se ha visto en el entorno doméstico. Un usuario podría manejar todos sus actuadores sin necesidad de tener acceso a ellos mediante el envío de mensajes en las redes sociales y estar al tanto de todos los cambios que ocurran en sus sensores.

Estos son solo algunos de los ámbitos de aplicación del sistema ya que las posibilidades son prácticamente infinitas, es aplicable siempre que se requiera interconectar objetos inteligentes sin la necesidad de reglas complejas y siempre que los mensajes que se intercambian los objetos puedan ser públicos.

# Capítulo 3. Estado de los Conocimientos Científico-Técnicos

En este capítulo se mostrará el estado de los conocimientos científico-técnicos implicados en la investigación realizada en este trabajo fin de máster. Además, durante este trabajo fin de máster, se han realizado diversas investigaciones paralelas al objetivo principal de las que también se mostrará el estado de los conocimientos científico-técnicos en el último apartado del capítulo, aunque en una menor profundidad que los relacionados directamente con el proyecto principal.

El proyecto principal de este trabajo fin de máster, Bilrost, es la propuesta de un nuevo sistema para intercomunicar objetos inteligentes de Internet de las Cosas a través de redes sociales mediante la generación de aplicaciones que conecten estos objetos a las redes sociales.

Existen muchas investigaciones que tratan la interconexión de objetos inteligentes pero la mayoría están basados en una arquitectura orientada a servicios (SOA - *Service Oriented Architecture*) como puede ser REST (*Representational State Transfer*).

En (González García, Pelayo G-Bustelo, et al., 2014), los autores presentan una plataforma para la generación de aplicaciones que interconectan objetos inteligentes de una manera similar a la que se propone en este trabajo fin de máster pero haciendo uso de un servidor central y una arquitectura orientada a servicios web mediante REST. Sin embargo, este trabajo fin de máster propone usar redes sociales en vez de hacer uso de una infraestructura hardware compuesta por servidores físicos. Aun así, Bilrost si necesitará usar servicios web, pero no para interconectar los objetos sino para comunicarse con las APIs (*Application Programming Interfaces*) de las redes sociales, tarea que hacen directamente las aplicaciones generadas sin necesidad de un servidor intermedio.

En este trabajo fin de máster, se propone el uso de Lenguaje de Dominio Específico (DSL - *Domain-Specific Language*) para definir las características de las aplicaciones generadas según los requisitos de los usuarios. Este DSL contendrá todo lo necesario para establecer la comunicación con las redes sociales definiendo los componentes de los objetos inteligentes y las reglas de funcionamiento. Por ello, a continuación, se introducirán los conceptos involucrados en este proyecto, es decir, Internet de las Cosas, Objetos Inteligentes, Ingeniería dirigida por Modelos, Lenguajes de dominio específico y Redes Sociales.

Pero antes de introducir estos conceptos, es necesario indicar como estos conceptos son necesarios para lograr los objetivos planteados en el capítulo anterior.

- El **primer objetivo** trata de la integración de los objetos inteligentes de Internet de las Cosas en las redes sociales por lo que implica el estudio de **Internet de las Cosas**, **Objetos Inteligentes** y **Redes Sociales**.
- El **segundo objetivo** trata la posibilidad de que las personas puedan interactuar con los objetos inteligentes de Internet de las Cosas, ya integrados en las redes sociales. Por ello, también implica el estudio de **Internet de las Cosas**, **Objetos Inteligentes** y **Redes Sociales**.
- El **tercer objetivo** trata de la intercomunicación de los propios objetos inteligentes de Internet de las Cosas a través de las redes sociales por lo que, al igual que los objetivos anteriores, requieren el estudio de **Internet de las Cosas**, **Objetos Inteligentes** y **Redes Sociales**.

- El **cuarto objetivo** trata de la definición de un Lenguaje de Dominio Específico (DSL) que permita cubrir los tres objetivos anteriores mediante la definición de objetos inteligentes y como se integran en las redes sociales. Y el **quinto objetivo** trata de usar ese DSL para la generación de aplicaciones en diferentes plataformas mediante la integración de la opción de la plataforma en la propia sintaxis del lenguaje. Para ello se debe realizar un estudio de **Lenguajes de Dominio Específico (DSL)** e **Ingeniería Dirigida por Modelos (MDE)**, además de los estudios realizados para los objetivos anteriores.

## 3.1. Internet de las Cosas

Internet de las Cosas (IoT – *Internet of Things*) puede ser definido como la interacción entre cosas u objetos con el fin de cooperar entre ellos y crear aplicaciones os sistemas que siguen un objetivo común (Vermesan & Peter Friess, 2013). Según el consejo nacional de inteligencia de Estados Unidos (NIC - *National Intelligence Council*), Internet de las Cosas es una de las seis tecnologías más importantes hasta el 2025 (National Intelligence Council, 2008).

El concepto de Internet de las Cosas surgió como una solución a problemas existentes en las cadenas de suministro relacionados con la identificación de objetos, personas y animales. La primera vez que se introdujo el término fue en 1999. Kevin Ashton lo utilizó en un presentación de Procter & Gamble (P&G) donde habló sobre el uso de etiquetas RFID (*Radio Frequency IDentification*) en las cadenas de suministro (Vermesan & Peter Friess, 2013). Gracias al uso de etiquetas RFID, se ha podido mejorar la gestión de las cadenas de suministro mediante la identificación de los elementos implicados.

El objetivo principal del Internet de las Cosas es interconectar objetos heterogéneos y ubicuos, y diferentes sistemas entre sí lo que obliga a que estos objetos o sistemas dispongan de acceso a Internet (Lee & Kim, 2012). Se puede decir que la razón por la que surgió el Internet de las Cosas fue para extender el Internet a todas las cosas (Li, Xu, & Zhao, 2014). De todas formas, no solo la conexión entre objetos o sistemas, también conocido como comunicación máquina a máquina (M2M – *Machine-To-Machine*) y de las que se habla en (Borgia, 2014; Hasan, Hossain, & Niyato, 2013; Tan & Wang, 2010), es importante para Internet de las Cosas , sino que también la conexión entre objetos y personas y entre las propias personas. Estos otros dos tipos de comunicaciones se conocen como comunicación humano a máquina (H2M – Human-To-Machine) de la que se habla en (International Telecommunication Union, 2012) y como comunicación humano a humano (H2H – Human-To-Human) de la que se habla en (Tan & Wang, 2010), respectivamente. Los tres tipos de comunicaciones juntos permiten compartir información entre el mundo físico y el mundo virtual. Este proyecto, Bilrost, intenta ofrecer un nuevo método de establecer comunicaciones entre objetos y entre objetos y personas usando directamente redes sociales en vez de seguir el enfoque tradicional de usar servicios web.

Las aplicaciones de Internet de las Cosas suelen seguir una arquitectura básica compuesta de tres capas en la que cada capa tiene su propia función. Estas capas son: Capa de percepción, Capa de red y Capa de aplicación (Miao Yun & Bu Yuxin, 2010).

- **Capa de percepción:** La primera capa es la capa de percepción (*Perception Layer*). Esta capa es la encargada de identificar los objetos y recoger información de ellos. También suele ser llamada Capa de dispositivo (*Device Layer)* y es la capa donde se localizan los sensores y las etiquetas RFID entre otros componentes.
- **Capa de red:** La capa de red (*Network Layer*) es la capa que se encarga de comunicar las otras dos capas. Su principal función es la transmisión de datos y es donde se integran los medios de transmisión y los protocolos de comunicación como Bluetooth, ZigBee o WLAN entre otros.

- **Capa de aplicación:** La última capa es la capa de aplicación (*Application Layer*) y es la capa donde se localizan las aplicaciones finales. Su función es recoger la información de las capas inferiores y mostrarla al usuario a través de servicios o aplicaciones (Mashal et al., 2015).

La propuesta de este trabajo fin de master aprovechará los conocimientos adquiridos de este estudio ya que uno de sus objetivos es interconectar objetos inteligentes de manera que implementa la capa intermedia de la arquitectura básica mencionada.

Además, esta propuesta también abarcará el resto de capas facilitando su implementación ya que entre sus objetivos se encuentra facilitar la creación de aplicaciones que interconecten objetos mediante el uso de un DSL de manera que se cubriría las otras dos capas mediante la generación de aplicaciones que usen sensores y actuadores. Estas aplicaciones, se espera que tengan la mayor parte de la lógica de negocio ya implementada para que el usuario tenga que realizar el menor número de modificaciones para acceder a los sensores y/o actuadores y para obtener una aplicación final.

Por tanto, nuestra propuesta facilitará la creación de aplicaciones que se basen en estas tres capas dando la capa de red solucionada y facilitando el resto de capas.

## 3.2. Objetos Inteligentes

La definición mostrada de Internet de las cosas incluye la palabra objetos y, por tanto, es necesario tratar qué es un objeto y por qué es importante en Internet de las Cosas. Un objeto puede ser cualquier dispositivo o cosa sin tener en cuenta su nivel de inteligencia. Sin embargo, se pueden clasificar en dos tipos en función de si disponen de esa inteligencia o no. De esta manera tenemos dos tipos de objetos, los objetos inteligentes, conocidos como *Smart Objects* o *Intelligent Products*, y los objetos sin inteligencia que podemos denominar como *Not-Smart Objects.* En el mundo de Internet de las Cosas podemos encontrar tanto un tipo como otro.

Los objetos no inteligentes u objetos sin inteligencia, son dispositivos que necesitan de otro dispositivo para poder funcionar. En este concepto se engloban los **sensores** y los **actuadores**. Los **sensores** son dispositivos capaces de medir parámetros físicos como la fluctuación de la luz o de la temperatura, pero necesitan la ayuda de otro dispositivo para procesar la información ya que ellos solo son capaces de recoger datos, pero no disponen de la lógica necesaria para realizar otra tarea.

Por otro lado, tenemos los **actuadores**. Los actuadores no recogen datos, sino que realizan acciones simples como puede ser enviar un mensaje de texto, aunque lo más normal es que estas acciones sean mecánicas como encender o apagar un LED o mover un brazo robótico. Sin embargo, los actuadores no pueden funcionar por sí mismos, sino que necesitan de otro dispositivo que los controle y le indique como deben funcionar y bajo qué condiciones.

Los dispositivos que son capaces de hacer lo que los sensores y actuadores no pueden y necesitan, son los Objetos Inteligentes o ***Smart Objects***. Por ejemplo, un objeto inteligente puede ser una Raspberry Pi ya que esta es capaz de procesar los datos recogidos por los sensores que tenga conectados y realizar acciones bajo ciertas condiciones con los actuadores de los que disponga. Por tanto, se puede interpretar que los objetos inteligentes están formados por objetos no inteligentes. La principal diferencia entre un objeto inteligente y uno que no lo es, es que el objeto inteligente es capaz de funcionar de manera inteligente y autónoma bajo determinadas condiciones, además de poder interactuar con su entorno o con otros dispositivos gracias a disponer de un sistema operativo embebido en su interior y contar con diversos actuadores, sensores o incluso ambos (Hribernik, Ghrairi, Hans, & Thoben, 2011).

Los objetos inteligentes o *Smart Objects* son conocidos también con el nombre de productos inteligentes o ***Intelligent Products*** y pueden ser definidos como productos con la capacidad de monitorear, reaccionar y adaptarse al entorno que los rodea, manteniendo una comunicación activa y con un óptimo rendimiento (Ventä, 2007). Algunos ejemplos de objetos inteligentes que nos rodean diariamente son los teléfonos inteligentes, más conocidos como *smartphones*, tabletas o *tablets*, televisiones inteligentes o *Smart TVs*, o incluso algunos dispositivos que no esperaríamos que fuesen a llegar a ser inteligentes como cafeteras o algunos modelos de coches. Otro tipo de ejemplos de objetos inteligentes son los microcontroladores como Arduino ya que permite realizar prototipos de casi cualquier cosa sin necesidad de una gran inversión de dinero gracias a su gran adaptabilidad a proyectos e investigaciones de todo tipo como las realizadas en (Georgitzikis, Akribopoulos, & Chatzigiannakis, 2012; Hribernik et al., 2011; Piras, Carboni, Pintus, & Features, 2012). Se puede decir que cualquier objeto conectado a Internet y que sea capaz de manejar información puede ser considerado un objeto inteligente o *Smart Object*.

De acuerdo a (Meyer, Främling, & Holmström, 2009), podemos clasificar los objetos inteligentes a través de tres dimensiones que representan cualidad de la inteligencia de los objetos. Estas dimensiones son el nivel de inteligencia, la localización de la inteligencia y el nivel de agregación de la inteligencia.

- **Nivel de inteligencia:** Esta dimensión describe como de inteligente es un objeto usando tres niveles de inteligencia: la capacidad de **manejo de información**, la habilidad de **notificar eventos** y la capacidad de **tomar decisiones** por sí mismo.
- **Localización de la inteligencia:** Esta dimensión describe donde está localizada la inteligencia de un objeto. Según la clasificación descrita en (Meyer et al., 2009) hay dos posibles localizaciones: **inteligencia localizada en la red** e **inteligencia localizada en el objeto**. Aunque en (Meyer et al., 2009) solo describen estas dos posibilidades, muestran una tercera en una figura que se corresponde con la combinación de ambas posibilidades, es decir, **inteligencia repartida** entre la red y el objeto. Las plataformas que localizan la inteligencia en la red son conocidas como portales o ***portal platforms*** (Ramparany & Boissier, 2002), las plataformas que localizan la inteligencia en el objeto son conocidas como plataformas embebidas o ***embedded platforms*** (Ramparany & Boissier, 2002) y las plataformas que combinan ambas localizaciones de la inteligencia son conocidas como plataformas sustitutas o ***surrogated platforms*** (Ramparany & Boissier, 2002).
- **Nivel de agregación de la inteligencia:** Esta dimensión describe como se distribuye la inteligencia en los objetos ya que puede estar dividida en diferentes partes. Existen dos posibles niveles de agregación: **inteligencia a nivel de ítem** e **inteligencia a nivel de contenedor**. El primer nivel ocurre cuando el dispositivo no puede ser dividido en partes más pequeñas sin perder la inteligencia. Un ejemplo de este primer nivel podría ser un sensor con un pequeño sistema operativo integrado que aporte la inteligencia necesaria. El sensor no forma parte de un objeto más grande y tiene su propia inteligencia. El segundo nivel aparece cuando la inteligencia se encuentra en un contenedor de más objetos que pueden ser desconectados sin afectar a la inteligencia. Un ejemplo de este segundo nivel puede ser una placa Arduino con varios sensores conectados. Los sensores solo tienen la capacidad de medir variables físicas y el Arduino se encarga de recoger los datos y procesarlos, pero si se desconecta alguno de los sensores el Arduino sigue funcionando, aunque se reduzca su funcionalidad.

Gracias a los objetos inteligentes, Internet de las Cosas otorga a la red inteligencia y la posibilidad de realizar acciones además de la habilidad de transmitir datos.

La propuesta de este trabajo fin de máster tiene como objetivo hacer más fácil la conexión de objetos inteligentes mediante el uso de redes sociales como método de comunicación. Para lograr este objetivo, se propone la definición de un DSL, que a través de MDE permita generar aplicaciones que puedan funcionar en diversos objetos inteligentes con la implementación relacionada con la conexión, publicación y consulta de redes sociales ya completada.

Por lo tanto, mediante la propuesta realizada, se podrá otorgar a los objetos inteligentes la capacidad de tomar decisiones por sí mismos, localizar su inteligencia dentro de sí y distribuirla a nivel de contenedor.

## 3.3. Lenguajes de Dominio Específico

Un Lenguaje de Dominio Específico (DSL – *Domain-Specific Language*) es un lenguaje, normalmente declarativo, desarrollado para solventar problemas de un dominio específico (Ford & Davis, 2006) con un alto poder expresivo como se demuestras en (Deursen, Klint, & Visser, 2000; Deursen & Klint, 1998; Núñez-Valdez, Sanjuan-Martinez, Bustelo, Lovelle, & Infante-Hernandez, 2013).

Frecuentemente, las soluciones genéricas no aportan la solución más óptima. Esto es uno de los problemas de los lenguajes de propósito general y por lo que los DSLs aparecieron. El uso de un DSL permite el desarrollo de soluciones optimizadas y mejor adaptadas a problemas específicos (Spinellis, 2001).

Los DSLs abstraen el nivel de conocimiento y permiten reducir la dificultad de las APIs (*Application Programming Interface*) de manera que los usuarios finales puedan trabajar con modelos sin tener conocimientos específicos sobre el software o sobre las APIs (D. Thomas, 2004). Los DSLs son una pieza muy importante en MDE, y por tanto, deben estar basados en un metamodelo con el fin de trabajar de manera adecuada con MDE.

El uso de DSLs aporta varias ventajas según (Cook, Jones, Kent, & Wills, 2007) como la mejora en la productividad debido a la reducción de errores y a la facilidad de su detección, una mayor comprensión del lenguaje gracias al uso de términos relacionados con el dominio específico, la portabilidad, la posibilidad de reaprovechar modelos para propósitos comunices y su mantenimiento (Deursen & Klint, 1998). Sin embargo, también existen algunas desventajas (Cook et al., 2007) como un peor rendimiento derivado de la computación extra de elevar el nivel de abstracción, el tiempo y el coste destinado a la investigación sobre la creación del DSL, las dificultades derivadas de lograr un correcto funcionamiento del DSL, la documentación adicional necesaria para explicar cómo funciona el DSL y la preparación que los nuevos desarrolladores necesitarán sobre el uso del DSL.

Mediante el uso de un DSL se pretende facilitar el desarrollo de aplicaciones que interconecten objetos. El DSL que se propone usar deberá ser útil para generar aplicaciones que conecten objetos inteligentes, heterogéneos y ubicuos entre sí, y con las personas a través de redes sociales.

## 3.4. Ingeniería Dirigida por Modelos

La Ingeniería Dirigida por Modelos (MDE – *Model-Driven Engineering*) surgió con el fin de solventar los problemas que han existido en el desarrollo del software desde los años 60. Estos problemas ya fueron descritos por Dijkstra en (Dijkstra, 1972) y por la Organización del Tratado del Atlántico Norte (OTAN) en (Naur & Randell, 1968). Según (González et al., 2008) estos problemas siguen presente y son la baja calidad del software desarrollado, el incumplimiento de los presupuestos y planificaciones y el consiguiente incremento del coste de mantenimiento.

Sin embargo, han surgido pequeñas soluciones que ayudan a minimizar estos problemas mediante la reducción de la complejidad esencial (Brooks, 1987), complejidad que depende en de las características derivadas de los requisitos funcionales, y la complejidad accidental (Brooks, 1987), complejidad relacionada con la misma acción de programar y escribir código.

Una de las soluciones que han surgido para solventar estos problemas fue la industrialización y automatización del proceso de desarrollo del software. Uno de los intentos más populares de industrialización del software fue MDE, un enfoque para diseñar y desarrollar software basado en el uso de modelos como se muestra en (Hailpern & Tarr, 2006; Seidewitz, 2003; Selic, 2003a). El objetivo de MDE es incrementar la productividad y reducir el tiempo de desarrollo acercando el proceso de desarrollo al dominio del problema a través de la creación de uno o más modelos e incrementando el nivel de abstracción para que sea más fácil de entender la las personas, por ejemplo, haciendo uso de los lenguajes de dominio específico (DSL). Por tanto, MDE trata de resolver la complejidad del diseño y desarrollo del software moderno (Selic, 2008), permitiendo la generación de código o diagramas (Paper, 2000) mediante procesos automáticos o por lo menos semiautomáticos (Selic, 2003b).

MDE se basa en el uso de modelos en el ciclo de desarrollo. Para ello, MDE tiene una terminología propia compuesta, entre otros elemento, por dominio, metamodelo, sintaxis abstracta, sintaxis concreta, modelo y semántica estática y que se definen a continuación según (García-Díaz, 2011).

- **Dominio**: El dominio delimita el campo de conocimiento y es la entrada de MDE. Para aplicar MDE siempre es necesario un dominio.
- **Metamodelo**: Los conceptos más relevantes del dominio se definen de manera formal partir del metamodelo.
- **Sintaxis abstracta**: Mediante la sintaxis abstrae de un lenguaje se define las construcciones, las propiedades y los conectores que el lenguaje puede poseer. Aun así, también se pueden definir reglas del lenguaje en el metamodelo para evitar malas prácticas.
- **Sintaxis concreta**: Mediante la sintaxis concreta se define la notación que los usuarios usarán. En el mejor de los casos, cada concepto del dominio y representando en el metamodelo debe tener una representación en la sintaxis concreta mediante la notación específica correspondiente.
- **Semántica estática**: Cogiendo como partida la sintaxis abstracta, la semántica estática de los metamodelos se usa para asegurar que los modelos están bien construidos a través de comprobaciones semánticas en el mismo.
- **Modelo**: Los modelos son instancias de los metamodelos representadas mediante una sintaxis concreta. Los modelos tienen que respetar la semántica estática del metamodelos para que las construcciones sean coherentes con dentro del dominio.

También es necesario indicar lo que son las transformaciones entre modelos ya que se necesitarán en la propuesta de este trabajo fin de máster y forman parte de la Ingeniería Dirigida por Modelos. Las transformaciones entre modelos son llamadas *model transformation* o mapping y existen 3 tipos según (Sendall & Kozaczynski, 2003):

- **Manipulación directa del modelo** (*Direct model manipulation* o *pull*): Este tipo de transformación se basa en acceder a la representación interna del modelo y manipularla mediante el uso de una API que lo permita.
- **Representación intermedia** (*Intermediate representation*): Este tipo de transformación se basa en exportar el modelo a un formato estándar, normalmente XML, y transformarlo con una herramienta externa. Este tipo de transformaciones permite realizar transformaciones por lotes.

- **Transformación soportada por el lenguaje** (*Transformation language support*): Este tipo de transformación se basa en el uso de un lenguaje que contenga un conjunto de construcciones o mecanismos para expresar, componer y aplicar las transformaciones de manera explícita.

En este trabajo fin de máster se propone la creación de un DSL, como ya se ha mencionado anteriormente, que permitirá a los usuarios definir modelos de aplicaciones, y que mediante la utilización de MDE, se transformarán en aplicaciones compatibles con diferentes plataformas y/o sistemas operativos.

Estos modelos serán implementaciones de un metamodelo que se habrá definido basándose en Ecore y que identifica los componentes de la sintaxis abstracta en la que se basará la gramática libre de contexto que define la sintaxis del DSL.

Esta gramática será la semántica estática que la sintaxis concreta deberá respetar, es decir, la sintaxis del DSL que los usuarios podrán utilizar para definir los modelos necesarios para generar las aplicaciones.

## 3.5. Redes sociales

Las redes sociales, también conocidas en la literatura como *Online Social Networks* (OSN) están consideradas como una pieza importante para lograr la convergencia entre el mundo real y el mundo digital en la Web 2.0 (Blackstock et al., 2011). Las redes sociales son muy útiles para mantenerse en contacto con familiares, amigos o compañeros de trabajo gracias a la existencia de relaciones entre usuarios y a la capacidad de compartir contenidos con ellos. Además, las redes sociales suelen ofrecer servicios a terceros desarrolladores a través de APIs para crear nuevas aplicaciones sociales. Entre los servicios que ofrecen a través de sus APIs hay: servicios de autenticación, métodos para definir permisos de acceso, recolección de datos de la red social y la posibilidad, en algunos casos, de extender la funcionalidad de la propia red social integrando aplicaciones de terceros en ella.

Las redes sociales, especialmente Twitter, son usadas frecuentemente con fines de investigación en aquellas que necesiten recopilar datos sobre personas y/o eventos. Por ejemplo, en (Anantharam, Barnaghi, Thirunarayan, & Sheth, 2015), los autores usan Twitter para extraer información sobre eventos de tráfico mediante el procesado de lenguaje natural.

Twitter es una red social online y servicio de *microblogging* basada en mensajes cortos de hasta 140 caracteres muy usada para fines de investigación por las características que tiene. Las relaciones entre usuarios de redes sociales suelen basarse en la reciprocidad, es decir, un usuario tiene que aceptar una invitación de otro usuario para establecer una relación. Este es el sistema que utiliza por ejemplo Facebook. Sin embargo, Twitter basa las relaciones de sus usuarios en seguir a otros usuarios y en ser seguido por otros usuarios sin la necesidad de reciprocidad (Kwak, Lee, Park, & Moon, 2010). Los usuarios son libres para seguir a cualquier otro usuario sin que este último siga al primero. Otra característica importante de Twitter como servicio de *microblogging* es su naturaleza de funcionamiento en tiempo real. Además, Twitter cuenta con un lenguaje de marcado especializado que permite añadir cierta información semántica a los mensajes y hacer más fácil el desarrollo de prototipos. Debido a todas estas características, se ha decidido usar Twitter en Bilrost como red social para la transmisión de datos en la conexión de objetos inteligentes.

En el marco de Internet de las Cosas, Twitter también ha sido una red social bastante usada con fines de investigación. Por ejemplo, en (Doran, Gokhale, & Dagnino, 2013) se ha utilizado de manera que se han considerado a las personas como otro sensor más dentro de ciudades inteligentes o *Smart Cities*. También en (Sakaki, Okazaki, & Matsuo, 2010) se han utilizado a personas como sensores pero para lograr la detección de terremotos.

Otro ejemplo de uso de Twitter en el marco de Internet de las Cosas es el que se muestra en (Cacho et al., 2016). Los autores usan las opiniones de Twitter para ayudar a tomar decisiones inteligentes sobre el destino para hacer turismo.

Hay varias propuestas sobre la combinación de redes sociales e Internet de las Cosas basadas en una deducción hecha por científicos de Ericsson (Atzori et al., 2014). Estos científicos observaron que las personas se familiarizan mucho mejor con las tecnologías de Internet de las Cosas si existen una analogía entre estas y los hábitos diarios en las redes sociales como Twitter o Facebook. Muchas de las propuestas realizadas se vanas en aportar a los objetos inteligentes habilidades de socialización con el fin de establecer relaciones entre los propios objetos. Este concepto se denomina Internet de las Cosas Social (SIoT – *Social Internet of Things*) y fue presentado en (Atzori et al., 2014; Atzori, Iera, Morabito, & Nitti, 2012).

La propuesta de este trabajo fin de máster propone el uso de las redes sociales en el mundo de Internet de las Cosas, como medio de comunicación de objetos inteligentes, logrando así, una integración de objetos de Internet de las Cosas en las redes sociales y permitiendo la interacción entre objetos y personas.

## 3.6. Trabajo relacionado

En los apartados anteriores se han presentado los conceptos implicados en el proyecto principal de este trabajo fin de máster. Sin embargo, no se han presentado investigaciones que guarden algún parecido con la propuesta realizada. Hay más investigaciones que abarcan la comunicación entre objetos en la literatura, pero no hay muchas que usen las redes sociales para establecer esa comunicación.

### 3.6.1. Internet de las Cosas Social (SIoT)

Entre las investigaciones que se centran en el uso de redes sociales y objetos de Internet de las Cosas encontramos aquellas relacionadas con Internet de las Cosas Social (SIoT). En (Atzori et al., 2014, 2012) proponen otorgar habilidades sociales a los objetos sin la interacción humana. En (Atzori et al., 2014) también proponen las características que una red social debe tener para ser apropiada para los objetos inteligentes.

La propuesta de Bilrost difiere de SIoT debido a que no se basa en la socialización de objetos, sino que se centra en la creación de aplicaciones donde los objetos inteligentes se comunican entre ellos utilizando redes sociales en vez de servicios web, pero sin establecer relaciones sociales.

### 3.6.2. Beneficios de las redes sociales en Internet de las Cosas

Otro trabajo relacionado con Bilrost es el trabajo presentado en (Blackstock et al., 2011) donde se exponen los beneficios de usar redes sociales en lo que se denomina la Web de las Cosas o *Web of Things*, llamado al uso de servicios web para diversos objetivos dentro de Internet de las Cosas. Además, analizan el uso de contenedores de aplicaciones de terceros que algunas redes sociales ofrecen como servicio. Para ello crean aplicaciones que integran los objetos inteligentes en una de las redes sociales más usadas y que ofrece el servicio de contenedor de aplicaciones de terceros, Facebook.

Sin embargo, esta investigación no propone un sistema final para interconectar objetos inteligentes usando redes sociales como hace Bilrost.

### 3.6.3. Objetos capaces de publicar en Twitter

Otra propuesta sobre objetos que son capaces de publicar mensajes en Twitter es la presentada en (Kranz, Roalter, Michahelles, & Michahelles, 2010). Sus autores hablan sobre las posibilidades que surgen de publicar eventos procedentes de objetos inteligentes, como sensores o actuadores, en las redes sociales. Aun así, tampoco proponen un sistema final que permita de manera sencilla publicar eventos en Twitter como propone Bilrost.

La propuesta de este trabajo fin de máster pretende publicar en Twitter la información recopilada de los sensores y las llamadas a las acciones de los actuadores con el fin de acercar los objetos inteligentes a las personas, concretamente a los usuarios de las redes sociales.

### 3.6.4. Publicación del acceso a los objetos en Twitter

Una investigación sobre compartir objetos a través de redes sociales es la presentada en (D. Guinard et al., 2010) y cuya propuesta la denominaron *Social Access Controller* (SAC). Los autores proponen el uso de redes sociales para compartir con los usuarios la conexión a los objetos inteligentes mediante un sistema que exponen una API basada en servicios REST que permite acceder y controlar objetos inteligentes registrados en el sistema.

Sin embargo, el uso que ellos hacen de las redes sociales no es con el fin de comunicar los objetos entre sí, sino que las usan para compartir como conectarse al objeto inteligente que al final se realiza a través de una arquitectura REST haciendo uso de un servidor central o proxy.

Bilrost propone evitar el uso de un sistema intermedio para establecer la comunicación entre objetos mediante la generación de aplicaciones donde la conexión ya esté implementada y puedan ser ejecutadas en los objetos inteligentes compatibles directamente.

### 3.6.5. Comunicación entre objetos mediante REST

Existen más investigaciones que se centran en la comunicación entre objetos inteligentes a través de servicios REST en vez de usar redes sociales como se propone en este trabajo fin de máster.

La plataforma de Internet de las Cosas Midgar es una de estas investigaciones. Midgar (González García, Pascual Espada, Núñez-Valdez, & García-Díaz, 2014; González García, Pelayo G-Bustelo, et al., 2014) permite la interconexión de objetos heterogéneos y ubicuos entre sí usando un DSL gráfico con el fin de hacer más fácil la creación de esa interconexión para las personas sin conocimientos relacionados con el desarrollo de software o programación.

Una desventaja de Midgar respecto a la propuesta de Bilrost es la necesidad de un servidor donde registrar los objetos que pueden conectarse a través de la arquitectura REST. Con Bilrost, se propone evitar esa dependencia gracias al uso de las redes sociales. Por otro lado, Midgar solo permite conectar objetos entre sí mientras que Bilrost, gracias al uso de redes sociales, pretende permitir a las personas interactuar con los objetos inteligentes.

Sin embargo, Midgar permite la generación completa de objetos inteligentes mientras que en la propuesta inicial de Bilrost no se introduce esta fase ya que propone centrarse en la intercomunicación de objetos.

Midgar también permite una mayor flexibilidad a la hora de automatizar tareas de la que se espera que la propuesta de este trabajo fin de master abarque, pero la investigación de Bilrost no está enfocada en proponer un sistema final que permita generar aplicaciones que permitan realizar cualquier tarea de Internet de las Cosas sino comprobar que es posible interconectar estos objetos a través de redes sociales en vez de seguir el método tradicional de servicios web.

### 3.6.6. Uso de mensajes instantáneos

Otro enfoque para comunicar objetos con personas es el uso de mensajes instantáneos en vez de servicios web o redes sociales como hacen en (Aurell, 2005; Choi, Park, Ko, Moon, & Lee, 2009; Choi & Yoo, 2008). Este enfoque tiene ciertas desventajas como la necesidad de aplicaciones exclusivas para realizar la comunicación mientras que el uso de redes sociales permite usar las propias aplicaciones de las redes sociales o navegadores web con el fin de acceder a las publicaciones e los sensores o para controlar actuadores.

### 3.6.7. Paraimpu

Paraimpu (Pintus, Carboni, & Piras, 2012) es otra plataforma de Internet de las Cosas creada como una plataforma social. Sus usuarios pueden añadir, usar, compartir, componer o conectar objetos inteligentes o cualquier otro servicio compatible como alguna red social. Paraimpu permite crear aplicaciones donde las redes sociales puede ser actuadores que se usan para publicar mensajes en Facebook o en Twitter cuando ocurra un evento.

Sin embargo, este enfoque no es similar al enfoque de la propuesta de Bilrost puesto que Bilrost no usará las redes sociales como sensores o actuadores, sino que las usará como método de transmisión de datos entre los objetos inteligentes. Por otro lado, Paraimpu también usa las redes sociales para iniciar sesión en la plataforma, para recopilar los contactos y amigos de los usuarios.

Para finalizar con el trabajo relacionado, mencionar que también existen investigaciones que proponen el uso de redes sociales para publicar y recopilar datos de sensores como (Baqer & Kamal, 2009; Baqer, 2010; S. Thomas, Cho, & Srivastava, 2007) con el fin de lograr un objetivo pero no usan las redes sociales como lo se propone que lo haga Bilrost, para comunicar los objetos entre sí.

## 3.7. Investigaciones paralelas

Además de la investigación realizada para el proyecto principal de este trabajo fin de máster, se han realizado otras investigaciones relacionadas con los campos de investigación del proyecto principal pero no relacionadas directamente con el proyecto.

En los siguientes apartados se realiza una descripción detallada de la investigación realizada, organizada por los diferentes conceptos estudiados, de forma transversal a los objetivos principales planteados en el apartado 2.1. para este trabajo fin de máster.

### 3.7.1. Lógica difusa

El término lógica difusa fue introducido por Zadeh como una manera de tratar con los problemas de sentido común. Primero introdujo los conjuntos difusos en (L. a. Zadeh, 1965) y después la lógica difusa en (L. A. Zadeh, 1975). Desde entonces, la lógica difusa ha sido abordada en muchas investigaciones. Un tema común relacionado con la lógica difusa es su aplicación en el ahorro de batería de los dispositivos.

Por ejemplo, en (Larios, Barbancho, Molina, & León, 2012), proponen el uso de la lógica difusa para obtener la localización de un dispositivo logrando una reducción de errores y la mejora de la precisión. Además, este enfoque permite reducir el gasto energético de diferentes elementos de los dispositivos como el sensor GPS mejorando así el consumo de batería.

Otra investigación relacionada con el consumo de energía y la lógica difusa es la presentada en (Chamodrakas & Martakos, 2012). En ella, sus autores proponen el uso de conjuntos difusos como método para escoger la red a la que un dispositivo debe conectarse de una manera eficiente con el fin de conseguir un bajo consumo energético, una buena calidad del servicio (QoS – *Quality of Service*) y un buen rendimiento. También se ha propuesto en (Bagchi, 2011), el uso de la lógica difusa para mejorar la calidad de la reproducción de contenido multimedia en *Streaming* junto a una mejora del consumo energético. Como último ejemplo de reducción de consumo energético aplicando lógica difusa se puede mencionar el uso de la lógica difusa en vehículos con el fin de mejorar el intercambio de información (Cueva-Fernandez, Pascual Espada, García-Díaz, & Gonzalez-Crespo, 2015), concretamente el consumo energético, entre los sensores de los coches y los servidores de la aplicación propuesta en (Cueva-Fernandez, Espada, García-Díaz, García, & Garcia-Fernandez, 2014).

Otro uso de la lógica difusa es el mostrado en (Cueva-Fernandez, Espada, García-Díaz, Crespo, & Garcia-Fernandez, 2015), donde proponen un sistema para crear aplicaciones usando la voz gracias a la lógica difusa.

Todas estas investigaciones tienen en común el uso de la lógica difusa para tomar decisiones difíciles sin tener unas opciones claras que considerar.

La lógica difusa surgió para resolver los problemas que la lógica clásica no es capaz de abordar ya que esta última solo puede tratar con conjuntos de valores binarios (0 o 1). Sin embargo, hay muchos contextos en los que es necesario tener en cuenta más valores o posibilidades. Disponer de más de dos valores permite manejar más estados y no solo estados binarios lo que hace que las tomas de decisiones se puedan realizar con una mayor cantidad de información. Estos estados suelen denominarse «variables lingüísticas» y son capaces de representar características como el «tamaño» que puede tomar varios valores como «grande» o «pequeño» que no tienen por qué tener el mismo significado para todo el que lo interprete ya que depende de la «cognición individual».

Por ejemplo, responden a preguntas como cómo de alto es un edificio dependen de la interpretación de cada persona ya que, probablemente, no todas las personas responderían lo mismo. Para una persona que vive en el campo, una edificación de seis pisos puede ser grande, pero para una persona que vive en Nueva York, esa misma edificación puede ser pequeña debido a las grandes edificaciones de Nueva York.

Estas dificultades son las que la lógica difusa pretender afrontar. Hay otros muchos ejemplos donde usar la lógica difusa por requerir más de dos valores para representar su estado. Por ejemplo, en (Grant, 2007), sus autores aprovechan las ventajas de la lógica difusa para proponer un nuevo enfoque para el control de la diabetes.

La lógica difusa permite tratar con datos imprecisos y ambiguos con el fin de tomar decisiones como una persona lo haría. Esta lógica usa controladores denominados «controladores adaptativos» (Grant, 2007) o «sistemas expertos» (Russell & Norvig, 1995) basados en reglas del tipo «si X y Y entonces Z» para imitar el pensamiento difusa de los seres humanos. Estas reglas representan el conocimiento que dirige a los sistemas expertos y gracias a ese conocimiento, estos sistemas pueden llegar a decidir la decisión más óptima. La definición de las reglas es una tarea compleja y requiere del uso de variables lingüísticas puesto que la representación humana del conocimiento es difusa. El proceso de entender que significa una variable lingüística como por ejemplo «pequeño» se denomina *fuzzification* (Portmann, Andrushevich, Kistler, & Klapproth, 2010).

## 3.7.2. Smart Cities, Smart Towns and Smart Homes

Las *Smart Cities* o ciudades inteligentes son una parte muy importante en el Internet de las Cosas (Zanella, Bui, Castellani, Vangelista, & Zorzi, 2014). Una ciudad inteligente cuenta con diferentes tipos de sensores distribuidos a lo largo de la ciudad con el objetivo de recopilar información ella y ofrecer nuevos servicios a los ciudadanos (Clarke, 2013) gracias al uso de redes de comunicación ubicuas, redes de sensores (WSN – *Wireless Sensors Network*), redes de sensores y actuadores (WSAN – *Wireless Sensors Actuators Network*) y sistemas inteligentes, así como facilitar la vida diaria y mejorar la habitabilidad de la ciudad (Mitchell et al., 2013).

En (Vienna University of Technology, 2015) o en (Caragliu et al., 2011), se pueden encontrar algunas ciudades europeas inteligentes como Luxemburgo, Aberdeen, Oviedo u otras ciudades con su calificación basada en diferentes criterios. Además, también definen el concepto de ciudad inteligente y analizan las ciudades inteligentes de Europa en base a seis indicadores. Otro ejemplo de ciudad inteligente europea es el proyecto llamado *SmartSantander* mostrado en (Sanchez et al., 2014) y que proponen una arquitectura para aplicar Internet de las Cosas en las ciudades inteligentes y expone los diferentes servicios ofertados.

Por otro lado, también existen las *Smart Towns* o pueblos inteligentes que pueden ser definidos como pequeñas ciudades o pueblos con una gran cultura y patrimonio que necesita ser preservado y revitalizado en vez de intentar mejorar únicamente la habitabilidad como las ciudades inteligentes antes mencionadas. Los pueblos inteligentes tienen que proteger y expandir su cultura y patrimonio para evitar el olvido de sus construcciones, monumentos, paisajes, folclore, tradiciones y un largo etcétera. Por ejemplo, los pueblos inteligentes son capacees de compartir sus lugares, registrar su cultura y la forma de preparar su platos típicos, monitorizar las condiciones de un determinado lugar que necesita de condiciones especiales como una biblioteca o un museo, o con el fin de proteger sus monumentos (Jara et al., 2015).

Un concepto más cercano son las casas inteligentes o *Smart Homes* (Hribernik et al., 2011), también conocidas como hogares inteligentes o *Intelligent Homes*. Estas casas inteligentes buscan mejorar la habitabilidad de los hogares. Para ello proveen sistemas automáticos que son capaces de mejorar la vida diaria en su interior. Suelen estar basadas en redes de sensores y actuadores (WSAN) que controlan diferentes aspectos de la casa en base a ciertos eventos definidos y automatizados o a través de un control remoto que puede ser incluso un teléfono inteligente o *smartphone.* Se puede llegar a controlar las puertas y las ventanas usando los teléfonos inteligentes o etiquetas inteligentes como RFID (*Radio Frequency IDentification*) o NFC (*Near Field Communication*), a automatizar el sistema de luces y a ahorrar en calefacción mediante el uso de diferentes sensores y sistemas inteligentes, entre muchas otras posibilidades.

## 3.7.3. Visión por computador

Los ordenadores solo con capaces de procesar ceros y unos. Sin embargo, hace unos años, surgió la Inteligencia Artificial (AI – *Artificial Intelligent)* para ofrecer la posibilidad de crear programas que permita a los ordenadores aprender y mejorar. Este término fue acuñado por John McCarthy in 1955 en un conferencia de la Dartmouth Collage (McCarthy, Minsky, Rochester, & Shannon, 2006). Uno de los campos de la inteligencia artificial es la visión por computador. Este campo permite a los ordenadores aprender a reconocer, por ejemplo, imágenes o características de las imágenes. De esta manera, un computador puede llegar a ser capaz de reconocer personas, objetos, animales o una posición en una imagen. Por ello, el objetivo de la visión por computador es hacer que las máquinas sean capaces de entender el mundo (Wang, Komodakis, & Paragios, 2013).

Para investigar este objetivo, existen muchos algoritmos para el largo proceso de reconocer diferentes cosas en una imagen. Ejemplos de algoritmos que se usan para obtener características a partir de conjunto de datos usado para entrenar el modelo son «*Histogram of Oriented Gradient*» (HOG), «*Local Binary Patterns*» (LBP), «*Scale-Invariant Feature Transform*» (SIFT), «*Speeded-Up Robust Features*» (SURF), una combinación de varios, entre muchos otros. Otros algoritmos son los necesarios para entrenar el modelo usando las características extraídas con los algoritmos anteriores. Ejemplos de estos algoritmos son «*Support Vector Machine*» (SVM) y «Regresión logística» (*Logistic Regression*). Sin embargo, obtener un bueno modelo con el que trabajar es una tarea muy complicada. Es necesario partir de imágenes de alta calidad y realizar muchas pruebas con otras imágenes para comprobar si el modelo está funcionando correctamente. Además, los modelos pocas veces son reutilizables ya que si se crea un modelo general que puede usarse en diversos ámbitos probablemente tenga una precisión muy baja, por tanto, casi siempre es necesario crear nuevos modelos.

## 3.7.4. Trabajo relacionado

A partir de los conceptos teóricos investigados y presentados en los apartados anteriores, se han realizado diferentes investigaciones paralelas a este trabajo fin de máster, pero en el mismo campo de estudio, Internet de las Cosas.

En este apartado se presentarán estas investigaciones con el fin de mostrar el estado el trabajo relacionado y presente en la literatura científica. Estas investigaciones tratan los siguientes temas:

- La aplicación de lógica difusa el ámbito de Internet de las Cosas.
- La generación de objetos inteligentes de Internet de las Cosas.
- La aplicación de visión por computador en una plataforma de Internet de las Cosas.

### *3.7.4.1. IoFClime: La lógica difusa y el Internet de las Cosas para controlar la temperatura interior teniendo en cuenta las condiciones ambientales exteriores.*

Esta investigación paralela trata de la aplicación de la lógica difusa en un contexto de Internet de las Cosas, concretamente se propone un sistema para controlar la temperatura interior teniendo en cuenta las condiciones ambientales exteriores, es decir, la sensación térmica compuesta por la temperatura y por la humedad relativa. El control de la temperatura interior a través de un sistema de calefacción y un sistema de aire acondicionado que se basa en la temperatura actual y en la sensación térmica exterior para regularse. Además, gracias al uso de la lógica difusa se intenta lograr una optimización en el número de encendidos y apagados de ambos sistemas por lo que se reduce el consumo energético. También se debe destacar que esta investigación hace uso de varias plataformas de Internet de las Cosas para obtener los datos de la temperatura exterior y de la humedad relativa.

Hay varias soluciones comerciales que son capaces de controlar la temperatura de lugares específicos entre las que destacan Loxone[1] y Tadoº[2]. Además, y como ya se ha mencionado en un apartado anterior, existen diversas investigaciones que combinan la lógica difusa como en Vitruvius (Cueva-Fernandez, Espada, et al., 2015; Cueva-Fernandez, Pascual Espada, et al., 2015).

[1] Loxone smart home. easily controlled. intelligently automated. — loxone: http://www.loxone.com/enen/start.html

[2] For heating & air conditioning intelligent climate control — tado: https://www.tado.com/gb/

#### 3.7.4.1.1. Loxone

Loxone es un sistema de automatización del hogar que puede controlar las persianas, las luces, el sistema musical o el sistema de climatización. En lo que respecta a la temperatura, el sistema tiene algunas ventajas y desventajas en comparación con la propuesta de esta investigación paralela. Loxone tiene en cuenta la localización del usuario y registra la temperatura para calcular las tendencias y mejorar su funcionamiento posterior. Sin embargo, tiene un alto precio de adquisición y su instalación es bastante compleja. Además, Loxone no permite aprovechar datos de terceras partes mientras que la propuesta de esta investigación recopila los datos exteriores de dos plataformas online de Internet de las Cosas.

#### 3.7.4.1.2. Tadoº

Tadoº es termostato inteligente que puede ser controlado a través de una interfaz web o a través de un teléfono inteligente o *Smartphone.* Al igual que Loxone, aprovecha la localización del usuario, pero instalar Tadoº es algo más sencillo que Loxone. Tadoº también permite ahorrar gracias a un buen control de la temperatura. Además, es capaz de generar informes detallados y tiene en cuenta la predicción del tiempo para ajustar la temperatura del termostato. Esta última característica se puede comparar con la recopilación de datos de plataformas online de Internet de las Cosas. Por tanto, esto es una característica común entre esta propuesta y Tadoº. Sin embargo, su alto precio y su falta de inteligencia por no ser capaz de automatizar tareas, se presentan como desventaja respecto a la propuesta de esta investigación paralela.

#### 3.7.4.1.3. Vitruvius

Vitruvius (Cueva-Fernandez, Espada, et al., 2015; Cueva-Fernandez, Pascual Espada, et al., 2015) es una plataforma en la que el usuario puede generar aplicaciones en tiempo real que usan sensores instalados en vehículos. Para ello, Vitruvius dispone de una aplicación web compuesta por un editor de aplicaciones que permite diseñar las aplicaciones. Entre los sensores disponibles se pueden encontrar sensores de velocidad o de temperatura. Vitruvius combina Internet de las Cosas con la lógica difusa con el fin de reducir el frecuente envío de datos a los servidores.

En sus primeras versiones, Vitruvius no hacía uso de la lógica difusa por lo que enviaba datos cada segundo al servidor (Cueva-Fernandez et al., 2014). Gracias al uso de la lógica difusa, la cantidad de datos enviada se ha reducido y tanto la calidad de los datos como el rendimiento del análisis de estos, ha mejorado. Sin embargo, el usar lógica difusa hace que la detección de errores sea más difícil que sin aplicarla.

Mientras que Vitruvius usa la lógica difusa para mejorar la automatización del envío de datos al servidor buscando el mejor momento para ello, la propuesta de esta investigación usa la lógica difusa para automatizar los sistemas que controlar la temperatura con la intención de conseguir una buena sensación térmica y por tanto ahorrar energía a gracias al enfoque inteligente conseguido mediante la lógica difusa.

### *3.7.4.2. Midgar: Creación de un DSL gráfico para generar objetos inteligentes para escenarios de Internet de las Cosas usando MDE*

Esta investigación paralela trata de la ampliación de una plataforma IoT que se encargaba de interconectar objetos inteligentes a través de servicios Web (González García, Pelayo G-Bustelo, et al., 2014) con el fin de que sea capaz generar objetos completos mediante la utilización de un DSL gráfico. De esta manera, un usuario es capaz de generar aplicaciones completas para sus dispositivos (siempre que estén soportados por la plataforma) sin la necesidad de tener conocimientos de programación o desarrollo de software. Entre el trabajo relacionado de esta investigación se encuentran otras plataformas que permiten la creación de objetos inteligentes a través de un DSL gráfico.

Un problema de Internet de las Cosas es la necesidad de tener conocimientos sobre cómo desarrollar una aplicación que conecte tus objetos a Internet. Entre los dispositivos comunes que se pueden tener de Internet de las Cosas se encuentran los teléfonos inteligentes o los microcontroladores como Arduino. Sin embargo, las aplicaciones para estos dispositivos no se desarrollan de la mimas manera y tampoco usan el mismo lenguaje de programación. Por tanto, si se desea interconectar objetos heterogéneos es necesario saber cómo desarrollar aplicaciones y tener conocimientos de distintos lenguajes de programación. Pero el principal problema es la programación debido a que es un proceso difícil y complejo (Kaucic & Asic, 2011). Sin embargo, podemos encontrar algunas soluciones que facilitan esta tarea a las personas sin conocimientos de desarrollo. A continuación, se presentarán estas soluciones comparándolas con la propuesta en esta investigación paralela.

#### 3.7.4.2.1. Soluciones para teléfonos inteligentes

En este caso se han estudiado varios editores web gráficos que permiten la creación de aplicaciones para teléfonos inteligentes. En el caso aplicaciones para únicamente dispositivos Android existen, entre otras alternativas, AppsGeyser (BESTTOOLBARS, 2015), iBuildApp (iBuildApp, 2015), y Andromo (Indigo Rose Software, 2015). También existen otras alternativas que añaden a la creación de aplicaciones para dispositivos Android, la opción de crearlas para dispositivos con iOS y en HTML para cualquier teléfono inteligente como AppsBuilder (AppsBuilder SPA, 2013) y Infinite Monkeys (Infinite Monkeys, 2015). Y si además, se quiere extender el soporte a BlackBerry y Windows Phone, se puede usar Appypie (Appy Pie, 2015). Otra alternativa para generar aplicaciones para Android e iOS es Como (Como Ltd., 2015) y para Windows Phone AppMachine (AppMachine, 2011).

Todos estos editores web gráficos permiten crear aplicaciones para diferentes sistemas operativos de dispositivos móviles. Además, ofrecen soluciones de arrastrar y soltar (*Drag n Drop)* diferentes bloques para ir añadiendo la funcionalidad necesaria. Algunos, incluso ofrecen el uso de plantillas por defecto que ya incluyen un mínimo de funcionalidad que hace la creación de aplicaciones más fácil. Sin embargo, no permiten el uso de sensores de los dispositivos como el acelerómetro, el sensor de presión, el sensor de luz, etc. Normalmente, el único sensor que ofrecen es el GPS (Sistema de Posicionamiento Global). Este es un problema cuando los usuarios quieren crear aplicaciones para Internet de las Cosas que utilicen su teléfono inteligente. Debido a esto, en esta investigación se ha diseñado un DSL gráfico que permite crear aplicaciones para teléfonos inteligentes que usen sus sensores con el fin de mejorar la creación de aplicaciones de Internet de las Cosas.

#### 3.7.4.2.2. Bitbloq

Bitbloq es un editor web gráfico de la compañía española BQ que permite la creación de aplicaciones para sus dispositivos electrónicos como microcontroladores basados en Arduino. Este editor gráfico es una aplicación web que se basa en el uso de unas piezas de puzle de tipo «*jigsaw*» para crear el ciclo de vida del dispositivo. La primera pieza en la parte superior izquierda es la primera instrucción que el editor gráfico crea cuando se traducen las piezas a código fuente. El resto de la aplicación se traduce a partir de todas las piezas siguiendo la siguiente regla de prioridad: de derecha a izquierda y de arriba abajo. Además, existe la posibilidad de ver el código fuente en tiempo real de la aplicación que se está construyendo.

Las piezas tienen diferentes pestañas y huecos en blanco en función del tipo de la pieza. Por ejemplo, la pieza que representa a un sensor tiene un hueco en blanco para indicar el PIN de la placa Arduino a usar uniendo esa pieza con una pieza que represente un PIN de la placa.

Este editor gráfico tiene algunas piezas que representan sensores y actuadores, piezas para crear flujos como sentencias «*if*» o «*while*», piezas que representan valores lógicos como «*true*», «*false*» o condiciones y otras piezas que sirven para insertar números, colecciones, operadores, variables, texto o incluso piezas para establecer comunicaciones como enviar datos al USB o al Bluetooth.

Bitbloq usa un sistema basado en *Scratch*, lo que es una muy buena idea para crear un editor web gráfico puesto que *Scratch* (Lifelong Kindergarten, 2015) es un lenguaje gráfico muy usado en escuelas para que los estudiantes aprendan a programar (Garner, 2009; Kaucic & Asic, 2011). Sin embargo, esta investigación propone un nuevo DSL gráfico que evita piezas como las de comunicación, las variables y otras piezas con el fin de obtener una abstracción más rápida y sencilla para la gente sin conocimientos de desarrollo que necesiten crear aplicaciones para sus dispositivos como microcontroladores.

#### 3.7.4.2.3. Minibloq

MiniBloq v. 0.83 («MiniBloq», s. f.) es un editor de escritorio gráfico de código abierto para *Multiplo* y *Arduino*. Al igual que bitbloq, hace uso de piezas de puzle de tipo «*jigsaw*» para crear el ciclo de vida de un dispositivo. También genera el código en tiempo real y permite desplegar el código fuente directamente en el microcontrolador.

Sin embargo, en este caso, las piezas no tienen libertad de movimientos, es decir, si por ejemplo se quiere añadir una pieza que mejora otra pieza más básica, es necesario hacerlo desde el menú de la pieza básica en vez desde el menú general. Pero si se quiere empezar la siguiente instrucción si se debe hacer desde el menú general. También existe el caso contrario, la funcionalidad de las piezas está a veces replicada en otras piezas con iconos distintos. Otro problema de este editor es que automáticamente incluye por defecto los ficheros de cabecera «mbq.h» y «PinIRReceiber.h» aún si no se van a necesitar. Por otro lado, si se quiere hacer una tarea repetitiva es necesario agregar un bucle ya que el código generado se introduce solo en el método «*setup*» en vez de en el método «*loop*» de Arduino. Esto no es correcto ya que en Arduino, el método «*setup*» es para establecer las diferentes variables y pines mientras que el método «*loop*» es para crear las tareas repetitivas. Además, MiniBloq, al igual que BitBloq, necesitan usar variables en su flujo de trabajo.

Este editor gráfico facilita la generación de aplicaciones, pero tiene algún problema con los menús, con los iconos y con las acciones. Además, no el código generado no es completamente correcto. En esta investigación, se crea un DSL gráfico más abstracto que facilita aún más la generación de aplicaciones. Además, solo se genera el código necesario para realizar la tarea y se genera de forma correcta.

### *3.7.4.3. Midgar: Detección de personas a través de visión por computador en escenarios de Internet de las Cosas para mejorar la seguridad en Smart Cities, Smart Towns, y Smart Homes*

Esta investigación paralela propone la integración de la visión por computador en una plataforma de Internet de las Cosas con el objetivo de mejorar la seguridad. Para ello, se ha desarrollado un nuevo módulo para la plataforma de IoT Midgar (González García, Pelayo G-Bustelo, et al., 2014) que hace uso de una cámara IP que detecta la presencia de personas y notifica a través de la plataforma.

En la literatura actual, hay varios usos de cámara en combinación con Internet de las Cosas y sensores. En algunos casos, esta combinación se usa parar mejorar las condiciones de trabajo y obtener más datos sin la necesidad de desplazarse al lugar donde la cámara está localizada.

Uno de estos usos es para mejorar el cuidado de abejas y facilitar el trabajo de los apicultores (Murphy et al., 2015). Los autores usan un sensor de sonido que detecta cuando un sonido excede de un límite establecido y en ese caso envía imágenes a los apicultores a través de mensajes. A través de esa imagen, los apicultores deciden si deben visitar la colmena o no, dependiendo de lo que ellos vean en la imagen y de la información del sensor. Para ello han usado una red de sensores (WSN). Gracias a este sistema, los apicultores pueden reducir la frecuencia con la que visitan la colmena a momentos en los que la información recibida indique que es necesario. Además, en (Murphy et al., 2015), analizan la información de los sensores con un algoritmo que evita la interacción humana obteniendo el estado de la colmena pero aun así, la imagen debe ser revisada por un apicultor para saber que está pasando.

Otro campo de aplicación de la visión por computador junto a Internet de las Cosas, puede ser la educación. Por ejemplo, esta combinación puede ayudar a mejorar la forma de enseñar y aprender conocimientos recopilando datos de los estudiantes y del profesor y buscando la mejor manera de enseñar a través del análisis de los datos registrados. Además, gracias a mejorar el aprendizaje se puede conseguir que el patrimonio cultural de los pueblos y el folclore no se pierda (Jara et al., 2015).

Las propuestas anteriores combinan Internet de las Cosas con cámaras. Sin embargo, necesita de una persona que analice la imagen con el fin de tomar una decisión como es el primer caso. En la propuesta de esta investigación, se usa la cámara para obtener imágenes que son enviadas al módulo de visión por computador que se encarga de tomar una decisión y enviarla a la red de IoT. La propuesta automatiza este paso en base a un modelo que evita la intervención humana y acelera la respuesta, ya que, en algunos casos, las personas no son capaces de tomar decisiones inmediatamente ni de ver una gran cantidad de imágenes en poco tiempo.

## 3.7.5. Prototipos desarrollados

A parte de realizar investigaciones en conceptos teóricos, también se ha realizado la implementación de varios prototipos dentro del marco de la investigación, es decir, Internet de las Cosas. A continuación, se explicarán de manera breve, los prototipos desarrollados por el autor de este trabajo fin de máster durante la realización de este.

### *3.7.5.1. Prototipo de visión por computador*

Este prototipo ha servido de base para la realización del prototipo comentado en el apartado 3.7.4.3. sobre el uso de la visión de computador en Internet de las Cosas integrando un módulo de visión por computador en la plataforma Midgar.

El prototipo tiene la capacidad de detectar la presencia de personas mediante el análisis de imágenes tomadas por una cámara IP.

Se ha utilizado una cámara de seguridad IP equipada con el software adecuado para notificar cambios en la imagen, ya sean alteración de algún elemento o la presencia de un elemento nuevo.

El prototipo permite la detección de personas, la notificación de esa detección vía email adjuntando imágenes del reconocimiento donde se superpone un cuadrado indicando donde está la persona y el almacenamiento en sistemas de persistencia de la nube (Amazon, Azure, etc.) de fotos de retratos identificados durante las detecciones.


Figura 1. Esquema de funcionamiento del prototipo de visión por computador

Dispone de dos módulos distintos que varían su funcionamiento dependiendo de si se quiere usar el software de la cámara para detectar cambios en la grabación o si se desea hacer un análisis continuo del video capturado por la imagen.

El funcionamiento del prototipo está representando en la Figura 1. Si el módulo en funcionamiento es el que usa el software de la cámara, el funcionamiento empieza por una detección de movimiento de la cámara que envía las imágenes del movimiento a un servicio web alojado en Amazon Web Services. Una vez recibido, el prototipo, funcionando en AWS, analiza esas imágenes y envía las imágenes donde se han reconocido personas vía email a las direcciones subscritas. En el caso de no usar el software de la cámara, el servicio alojado en AWS se encarga de procesar, en directo, las imágenes de la cámara y notificar en caso de que detecte una persona.

Cada uno de estos módulos tiene sus ventajas e inconvenientes:

- El módulo que no usa el software de la cámara tiene como ventajas el acceso vía HTTP a la imagen y su compatibilidad con casi cualquier cámara IP del mercado. Sin embargo, como su funcionamiento se basa en el análisis continuo, el servicio tiene que analizar todas las imágenes que se obtienen de la cámara. Este proceso no es instantáneo por lo que se pierden *frames* provocando que alguna de las imágenes no se procese o que las que se procesen no sean las adecuadas. Además, el tráfico de datos entre el servicio y la cámara es constante lo que es un problema si se usan servicios que cobran por el tráfico de datos como muchos proveedores de servicios en la nube.

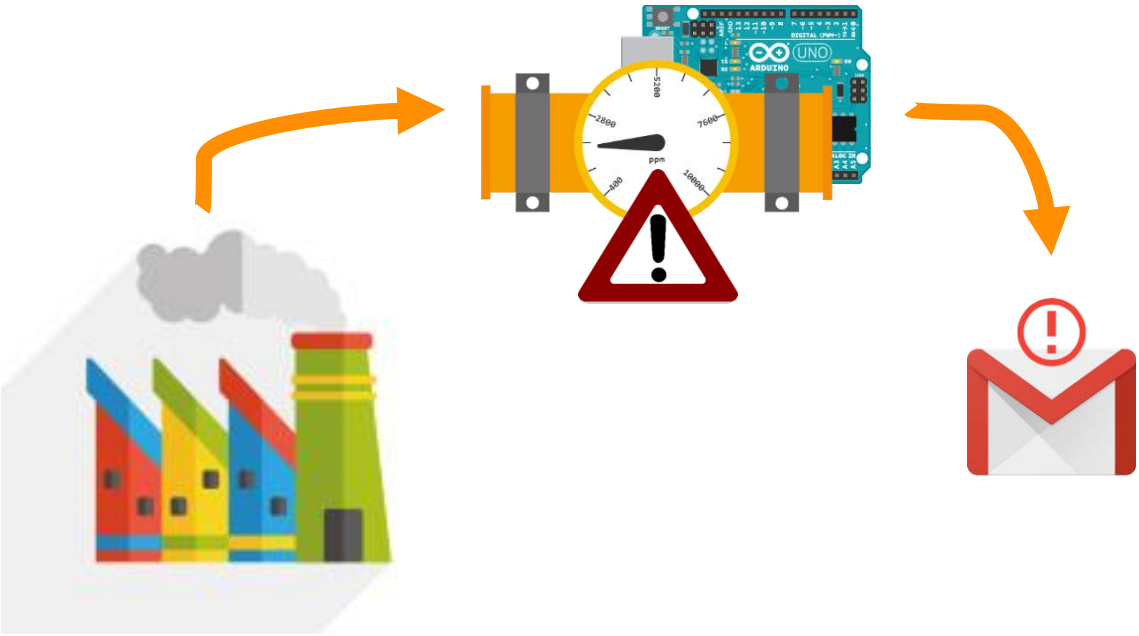



Figura 2. Esquema de funcionamiento del prototipo de nariz inteligente.

- El módulo que aprovecha el software de la cámara tiene como única desventaja el propio uso del software de la cámara ya que lo hace compatible únicamente con cámaras que tengan las funciones necesarias, en concreto, que dispongan software capaz de notificar a un servicio web la detección de movimiento mediante el envío de imágenes. Las ventajas de este módulo es la reducción del tráfico de datos y el número de *frames* que el algoritmo analiza ya que solo son necesarias las imágenes donde hay movimiento.

El prototipo ha sido desarrollado en Python 3.4 usando el servidor web Flask, para procesar los envíos de las fotos y gestionar la interfaz web. Se ha usado la librería OpenCV para visión por computador y la librería Boto para gestionar la conexión con Amazon Web Services.

### *3.7.5.2. Prototipo de nariz inteligente*

El prototipo desarrollado trata sobre el empleo de sensores de gases sobre un microcontrolador Arduino. A este prototipo se le ha llamado *IntelliNose* basándose en la combinación de la palabra *intelligent* y la palabra *nose*, es decir, nariz inteligente.

El prototipo implementa un medidor de gases que permite realizar acciones en función de reglas que sus usuarios pueden definir.

Las reglas se componen de un gas que se desea monitorizar, una condición que debe cumplir el valor detectado del gas por el sensor y la acción que se deberá efectuar si se cumple la condición.

Al tratarse de un prototipo, solo se realiza el envío de mails con información sobre el estado de los gases, pero ha sido preparado para poder integrar, de manera modular, más acciones como el envío de SMS, realización de llamadas, envío de notificaciones a dispositivos móviles, etc.

En la Figura 2 se muestra un esquema del funcionamiento del prototipo. El sistema compuesto por una placa Arduino y sensores de gases analiza los niveles de gases del ambiente y en caso de cumplir una condición se alerta al usuario a través de un email.

Se han utilizado 5 sensores de gases que permiten medir alcohol, metano, monóxido de carbono, dióxido de carbono e hidrógeno

El prototipo ha sido desarrollado en JavaScript sobre NodeJS, usando como base de datos MongoDB y utilizando en el cliente HTML5 generado a través de Jade y CSS generado con SASS.

# Capítulo 4. Descripción del Sistema

En este capítulo se presentará la solución propuesta para alcanzar los objetivos planteados en este proyecto fin de máster y que se resumen en la creación de un sistema que genere aplicaciones para diversas plataformas y/o sistemas operativos, que permitan interconectar objetos inteligentes de Internet de las Cosas a través de redes sociales mediante el uso de un DSL. Mediante la superación de esos objetivos se busca confirmar la hipótesis planteada.

Durante este capítulo se explicarán las distintas partes del sistema propuesto para confirmar la hipótesis comenzando por la creación, aplicando MDE, del DSL, llamado *Bilrost-Specific Language* (BSL), que permite definir objetos conectados a las redes sociales y finalizando con el prototipo desarrollado que hace uso de ese DSL para generar las aplicaciones correspondientes.

## 4.1. Bilrost-Specific Language (BSL)

El Lenguaje de Dominio Especifico que se ha definido para este proyecto fin de máster tiene como objetivo definir dispositivos capaces de integrarse en redes sociales y se le ha denominado ***Bilrost-Specific Language*** (BSL). Al definir un dispositivo se deberán definir ciertas propiedades de ese dispositivo y los objetos no inteligentes que contiene, es decir, los sensores y actuadores. Además, un dispositivo puede contar con reglas que le aporten cierto nivel de inteligencia.

En esta sección se presentará el metamodelo que implementan los modelos que los usuarios pueden definir mediante el DSL, la gramática que define el lenguaje y la sintaxis del lenguaje.

### 4.1.1. Metamodelo de BSL

La creación de ***Bilrost-Specific Language*** (BSL) se ha realizado aplicando MDE lo que conlleva seguir una serie de pasos. Lo primero que se necesita para aplicar MDE es definir el dominio del problema que se quiere resolver y que acota el campo de conocimientos. En esta propuesta, el dominio es la definición de dispositivos que se puedan conectar a redes sociales para publicar datos de sus sensores o para que sus actuadores puedan ser controlados por otros usuarios o dispositivos que hacen uso de reglas para automatizar el proceso.

Una vez que se tiene acotado el dominio, se debe definir el metamodelo a partir de un meta-metamodelo. La utilidad de los meta-metamodelos es hacer que los metamodelos creados a partir de ellos sean reutilizables, interoperables y portables, debido a que los meta-metamodelos son una abstracción de más alto nivel que define como debe ser el metamodelo.

El meta-metamodelo que se ha elegido para definir el metamodelo es Ecore (de Eclipse Foundation). El metamodelo describe los conceptos relevantes del dominio del problema y que se desean modelar mediante el DSL.

Los metamodelos definen la sintaxis abstracta del lenguaje, es decir, definen, a nivel conceptual, los componentes del modelo que se podrá definir con el lenguaje.

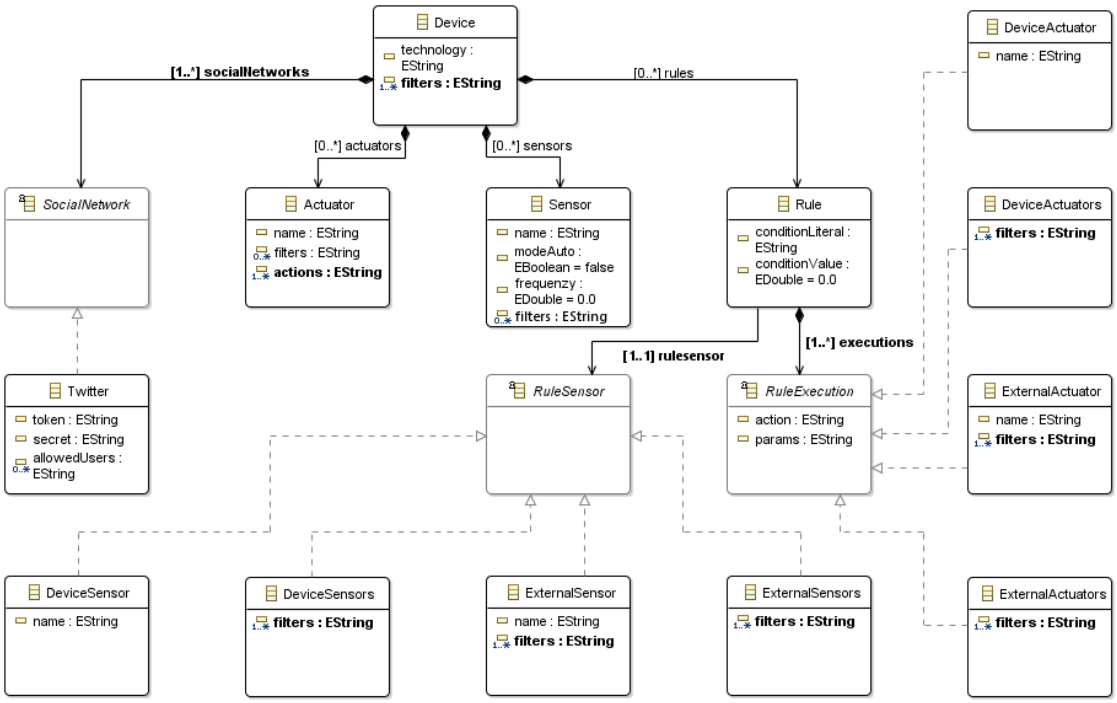



Figura 3. Metamodelo que describe los conceptos del dominio del problema.

En la Figura 3 se muestra el metamodelo del dominio este trabajo fin de máster y define los siguientes componentes:

- **El dispositivo** (*Device*): Este es el componente principal y su definición será el objetivo del lenguaje. Sus propiedades son la **tecnología** (*technology*) que representa la plataforma del dispositivo, por ejemplo, si el dispositivo es una Raspberry Pi, la plataforma podría ser Python ya que es uno de los lenguajes soportados por la Raspberry Pi; y los **filtros** (*filters*) que representan palabras claves necesarias para identificar los dispositivos en las redes sociales. Además, un dispositivo está compuesto por otros componentes: redes sociales, actuadores, sensores y reglas.
- **Red Social** (*SocialNetwork*): Un dispositivo podrá tener asociadas una o más redes sociales, pero al menos una. Las redes sociales son un componente abstracto que otro componente debe implementar, es decir, un dispositivo puede tener redes sociales con distintas propiedades, pero todas son redes sociales.
    - **Twitter**: En la solución adoptada se escogido Twitter como red social por lo que deberá de haber un componente que represente a esta red social y sus propiedades que son las **claves** requeridas por la API (*token* y *secret*) y los **usuarios** que podrán controlar al dispositivo (*allowedUsers*) aunque esta propiedad no debería ser obligatoria.
- **Actuador** (*Actuator*): Un dispositivo podrá estar formado por actuadores además de otros componentes. Mediante este componente se define cada uno de los actuadores del dispositivo. Sus propiedades son el **nombre** (*name*) del actuador, las **acciones** (*actions*) que puede efectuar (al menos una es necesaria) y los **filtros** (*filters*) que se podrán usar para identificar al actuador además del nombre.
- **Sensor** (*Sensor*): Un dispositivo podrá estar formado por sensores además de otros componentes como actuadores. Mediante este componente se define cada uno de los sensores de los que dispone el dispositivo. Sus propiedades son el **nombre** (*name*) del sensor, su **modo** (*modeAuto*) de funcionamiento que podrá ser automático o manual, la **frecuencia** (*frequency*) de funcionamiento en caso de ser automático y los **filtros** (*filters*) que se podrán usar para identificar al sensor además del nombre.

- **Regla** (*Rule*): Además de componentes físico como sensores o actuadores, un dispositivo podrá tener definidas reglas que automaticen las ejecuciones de acciones de sus actuadores o la invocación de acciones de actuadores de otros dispositivos en función de datos de sus sensores o de otros sensores disponibles en la red social. Sus propiedades son el **literal** de la condición (*conditionLiteral*), por ejemplo «mayor que», y el **valor** que debe cumplir la condición (*conditionValue*). Además de esas propiedades, una regla tiene un **sensor** (*RuleSensor*) y una serie de ejecuciones que se realizarán si se cumple la **condición** (*RuleExecution*).
- **Sensor de una regla** (*RuleSensor*): Una regla deberá tener un sensor del que tomar el valor que validará la condición para automatizar las ejecuciones de acciones. Este componente es un componente abstracto que implementan cuatro componentes diferentes ya que existen cuatro posibilidades a la hora de seleccionar el sensor de la regla: **un sensor del propio dispositivo** (*DeviceSensor*), **varios sensores del propio dispositivo** (*DeviceSensors*), **un sensor de un dispositivo ajeno o externo** (*ExternalSensor*) y **varios sensores de dispositivos externos** (*ExternalSensors*).
    - **Un sensor del propio dispositivo** (*DeviceSensor*): Las reglas pueden usar para evaluar la condición, el valor de un sensor del mismo dispositivo siendo necesario únicamente su **nombre** (*name*).
    - **Varios sensores del propio dispositivo** (*DeviceSensors*): Las reglas pueden usar para evaluar la condición, el valor de cualquier sensor del mismo dispositivo que contengan los **filtros** (filters) indicados.
    - **Un sensor de un dispositivo externo** (*ExternalSensor*): Las reglas pueden usar para evaluar la condición, el valor de un sensor de un dispositivo externo indicando el **nombre** (*name*) del sensor y los **filtros** (*filters*) que identifican a los dispositivos y/o sensor.
    - **Varios sensores de dispositivos externos** (*ExternalSensors*): Las reglas pueden usar para evaluar la condición, el valor de cualquier sensor de dispositivos externos identificados por los **filtros** (filters) indicados.
- **Ejecución de una regla** (*RuleExecution*): Una regla puede tener varias ejecuciones que realizar si la condición es correcta, pero al menos deberá disponer de una ejecución que hacer. Este componente es un componente abstracto que implementan cuatro componentes diferentes ya que existen cuatro posibilidades a la hora de seleccionar el actuador que realizará la ejecución de una acción: **un actuador del propio dispositivo** (*DeviceActuator*), **varios actuadores del propio dispositivo** (*DeviceActuators*), **un actuador de un dispositivo externo** (*ExternalActuator*) y **varios actuadores de dispositivos externos** (*ExternalActuators*). Además, tiene una serie de propiedades que comparten todas las posibilidades. Estas son el nombre de la **acción** (*action*) a realizar y los **parámetros** (*params*) que utilizará la acción.
    - **Un actuador del propio dispositivo** (*DeviceActuator*): Las reglas pueden realizar una acción de un actuador del mismo dispositivo siendo necesario únicamente su **nombre** (*name*).
    - **Varios actuadores del propio dispositivo** (*DeviceActuators*): Las reglas pueden realizar una acción de cualquier actuador del mismo dispositivo que contengan los **filtros** (filters) indicados.
    - **Un actuador de un dispositivo externo** (*ExternalActuator*): Las reglas pueden realizar una acción de un actuador de un dispositivo externo indicando el **nombre** (*name*) del actuador y los **filtros** (*filters*) que identifican a los dispositivos y/o actuador.
    - **Varios actuadores de dispositivos externos** (*ExternalActuators*): Las reglas pueden realizar una acción de cualquier actuador de dispositivos externos identificados por los **filtros** (filters) indicados.

## 4.1.2. Gramática del lenguaje BSL

Una vez obtenido el metamodelo se puede definir la semántica estática que rige las reglas que deberá respetar la sintaxis concreta, la sintaxis del DSL y la que los usuarios utilizarán. Como semántica estática se propone el uso de una gramática de tipo 2 o gramática libre de contexto.

En la Figura 4 se muestra la gramática definida para el lenguaje de dominio específico con notación de Backus-Naur (BNF).

```
<device>                          ::= DEVICE IN <platform> <properties>
                                      END
<platform>                        ::= PYTHON
                                    | JAVA
                                    | ANDROID
<properties>                      ::= <property>
                                    | <properties> <property>
<property>                        ::= <filter>
                                    | <social-networks>
                                    | <actuators>
                                    | <sensors>
                                    | <rules>
<filter>                          ::= FILTER BY <filters>
<filters>                         ::= WORD
                                    | <filters> COMMA WORD
<social-networks>                 ::= SOCIAL NETWORKS
                                      <social-networks-list>
<social-networks-list>            ::= <social-network>
                                    | <social-networks-list>
                                      <social-network>
<social-network>                  ::= CONNECT TO TWITTER
                                      <twitter-properties>
<twitter-properties>              ::= <token> <secret> <users>
                                    | <secret> <users> <token>
                                    | <users> <token> <secret>
                                    | <secret> <token> <users>
                                    | <token> <users> <secret>
                                    | <users> <secret> <token>
                                    | <secret> <token>
                                    | <token> <secret>
<token>                           ::= TOKEN WORD
<secret>                          ::= SECRET WORD
<users>                           ::= ALLOW <users-list>
<users-list>                      ::= WORD
                                    | <users-list> COMMA WORD
<actuators>                       ::= ACTUATORS <actuators-list>
<actuators-list>                  ::= <actuator>
                                    | <actuators-list> <actuator>
<actuator>                        ::= DEFINE ACTUATOR WORD
                                      <actuator-without-filters>
                                    | DEFINE ACTUATOR WORD
                                      <actuator-with-filters>
<actuator-without-filters>        ::= ACTIONS <actions>
<actuator-with-filters>           ::= <filter> ACTIONS <actions>
<actions>                         ::= WORD
                                    | <actions> COMMA WORD
<sensors>                         ::= SENSORS <sensors-list>
<sensors-list>                    ::= <sensor>
                                    | <sensors-list> <sensor>
<sensor>                          ::= DEFINE SENSOR WORD
                                      <sensor-without-filters>
                                    | DEFINE SENSOR WORD
                                      <sensor-with-filters>
<sensor-without-filters>          ::= <sensor-mode>
<sensor-with-filters>             ::= <filter> <sensor-mode>
<sensor-mode>                     ::= MODE AUTO NUMBER
                                      <frequency-multiplier>
                                    | MODE MANUAL
```

```
<frequency-multiplier>                ::= HOURS
                                        | MINUTES
                                        | SECONDS
<rules>                               ::= RULES <rules-list>
<rules-list>                          ::= <rule>
                                        | <rules-list> <rule>
<rule>                                ::= DEFINE RULE TO
                                            <rule-single-sensor> <rule-body>
                                        | DEFINE RULE TO
                                            <rule-multi-sensors> <rule-body>
<rule-single-sensor>                  ::= EXTERNAL SENSOR WORD <filter>
                                        | SENSOR WORD
<rule-multi-sensors>                  ::= ALL EXTERNAL SENSORS <filter>
                                        | ALL SENSORS <filter>
<rule-body>                           ::= <condition> <executions>
                                        | <executions> <condition>
<condition>                           ::= IF VALUE IS <literal> NUMBER
<literal>                             ::= LESS_THAN
                                        | EQUAL_TO
                                        | GREATER_THAN
                                        | <literal-composed>
<literal-composed>                    ::= LESS_THAN OR EQUAL_TO
                                        | GREATER_THAN OR EQUAL_TO
                                        | NOT EQUAL_TO
<executions>                          ::= <executions-list>
<executions-list>                     ::= <execution>
                                        | <executions-list> <execution>
<execution>                           ::= EXECUTE ACTION WORD WITH PARAMS
                                            PHRASE IN <execution-actuators>
                                        | EXECUTE ACTION WORD IN
                                            <execution-actuators>
<execution-actuators>                 ::= <execution-single-actuator>
                                        | <execution-multi-actuators>
<execution-single-actuator>           ::= EXTERNAL ACTUATOR WORD <filter>
                                        | ACTUATOR WORD
<execution-multi-actuators>           ::= ALL EXTERNAL ACTUATORS <filter>
                                        | <execution-multi-local-actuators>
<execution-multi-local-actuators>     ::= ALL ACTUATORS <filter>
                                        | ALL ACTUATORS <empty>
```

Figura 4. Gramática de Birlost-Specific Language en notación de Backus-Naur (BNF).

Esta gramática establece las reglas de la sintaxis del lenguaje de dominio específico desarrollado BSL (Bilrost-Specific Language) y que se detallará en el siguiente apartado.

### 4.1.3. Sintaxis del lenguaje BSL

A partir del metamodelo se han definido los componentes conceptuales del lenguaje, es decir, la sintaxis abstracta del lenguaje. Con esos elementos conceptuales se ha definido la semántica estática como una gramática libre de contexto que especifica las reglas que debe cumplir el lenguaje y a partir de esa gramática, se define la sintaxis concreta del lenguaje, la misma sintaxis que los usuarios usarán.

Entre las características de la sintaxis BSL se encuentran la no distinción entre mayúsculas y minúsculas y el permitir escribir todo el código en una única línea o en múltiples líneas con un sangrado a gusto del usuario.

La definición de un dispositivo se guarda en ficheros independientes, por lo que se deben crear tantos ficheros como dispositivos se quieran definir.

La especificación de un dispositivo está compuesta por 4 diferentes bloques: bloque de redes sociales, bloque de actuadores, bloque de sensores y bloque de reglas, y las propiedades del propio dispositivo. Estos bloques pueden ser definidos en cualquier orden, pero es necesario que esté presente el bloque de redes sociales y uno de los otros 3 bloques.

### *4.1.3.1. Definición del dispositivo*

Un dispositivo se puede definir indicando la tecnología necesaria para crear las aplicaciones para él y alguna palabra clave, llamadas filtros, que identifica al dispositivo en las redes sociales. Estas palabras clave se usan para filtrar las búsquedas de mensajes en las redes sociales en busca de mensajes cuyo objetivo sea el propio dispositivo. También son usados para publicar mensajes provenientes del dispositivo con el fin de saber que dispositivo es la fuente de los datos.

Además, un dispositivo tiene otros 4 bloques: un bloque para definir las redes sociales, un bloque para definir los actuadores, un bloque para definir los sensores y un bloque para especificar las reglas que automatizarán la invocación de acciones sobre actuadores si se cumplen las condiciones sobre los datos de los sensores.

En este prototipo se ha implementado la generación de proyectos escritos en Python, Java o Android y, por tanto, estos lenguajes son la tecnología que el dispositivo requiere.

El siguiente código es el esqueleto para definir un dispositivo.

```
DEVICE IN PYTHON | JAVA | ANDROID
  FILTER BY …
  SOCIAL NETWORKS …
  ACTUATORS …
  SENSORS …
  RULES …
```

Es obligatorio indicar la tecnología entre las posibilidades disponibles, los filtros del dispositivo y las redes sociales. El resto de bloques son opcionales, pero al menos uno de ellos debe estar presente.

Los filtros se utilizan para identificar el dispositivo en las redes sociales y al menos debe definirse uno para cada dispositivo.

Los mensajes que publique el dispositivo en redes sociales, siempre contendrán, al menos, sus filtros, y un dispositivo solo reaccionará ante mensajes que contengan sus filtros.

El número máximo de filtros no está definido pero el usuario debe tener en cuenta el límite de caracteres por mensaje de las redes sociales.

El siguiente código muestra el esqueleto para indicar los filtros del dispositivo.

```
FILTER BY 'filter1', 'filter2', …
```

### *4.1.3.2. Redes sociales*

El sistema está diseñado para soportar varias redes sociales. Este bloque comienza por las palabras *SOCIAL NETWORKS* seguidas de la configuración de cada red social. En el prototipo desarrollado se ha implementado la conexión a través de Twitter.

El siguiente código muestra el esqueleto para indicar que redes sociales usar en el dispositivo.

```
SOCIAL NETWORKS
   CONNECT TO TWITTER | OTHERS
```

Tras indicar la red social, el usuario debe definir los parámetros de la red social. En el caso de Twitter se necesitan dos *tokens* para publicar y buscar mensajes en la red social.

Además, se ha añadido otro parámetro que indicar que usuarios tienen permisos para controlar los actuadores del dispositivo, el parámetro *ALLOW*. Este parámetro es opcional, pero si está presente, el dispositivo solo reaccionará antes mensajes publicados por los usuarios indicados.

El siguiente código muestra el esqueleto de la configuración de Twitter como red social del dispositivo.

```
CONNECT TO TWITTER
  TOKEN 'tokenvalue'
  SECRET 'secretvalue'
  ALLOW 'user1', 'user2', …
```

### 4.1.3.3. Actuadores

Un dispositivo puede tener varios actuadores. Para introducir el bloque de actuadores es necesario escribir la palabra *ACTUATORS* y para introducir cada actuador es necesario escribir las palabras *DEFINE ACTUATOR* seguidas del nombre del actuador.

Todos los actuadores deben tener un nombre para identificarlo y controlarlo a través de mensajes en redes sociales.

Todos los actuadores deben tener una lista de acciones que puede realizar usando la palabra *ACTIONS*. Estas acciones serán los nombres de los métodos que el usuario deberá rellenar en la fase de completado del proyecto.

Cada actuador puede tener filtros con el fin de agrupar varios actuadores bajo una misma invocación. De esta manera, un mensaje puede invocar una acción de muchos actuadores si todos tienen filtros en común y tienen la acción.

El siguiente código muestra el esqueleto para definir actuadores que componen el dispositivo.

```
ACTUATORS
  DEFINE ACTUATOR 'name'
    FILTER BY 'filter1', 'filter2', …
    ACTIONS 'action1', 'action2', …
```

### 4.1.3.4. Sensores

Un dispositivo puede tener varios sensores. Para introducir el bloque de los sensores es necesario escribir la palabra *SENSORS* y para introducir cada sensor es necesario escribir las palabras *DEFINE SENSOR* seguidas del nombre del sensor.

Todos los sensores deben tener un nombre para identificarlo en los mensajes publicados en las redes sociales por el dispositivo.

Los sensores deben tener un parámetro llamado *MODE*. Este parámetro indica como publicará los datos el sensor:

- Modo manual indica que los usuarios deberán llamar explícitamente al método que publica los datos del sensor en redes sociales pasando por parámetros lo que desean que se publique. Esto deberán hacer en la fase de completado del proyecto.
- Modo automático, indica que el sensor publicará sus datos de acuerdo a una frecuencia definida. La frecuencia se define añadiendo un número después de la definición del modo seguido de la unidad de tiempo adecuada entre *SECONDS, MINUTES* o *HOURS*.

Cada sensor tiene filtros con el fin de agrupar varios sensores en la comprobación de valores de sensores en las reglas. De esta manera, un usuario puede definir reglas en las que la fuente de información pueda ser más de un sensor y si alguno de estos cumple la condición se realizarán las ejecuciones.

El siguiente código muestra el esqueleto para definir los sensores que componen un dispositivo.

```
SENSORS
  DEFINE SENSOR 'name'
    FILTER BY 'filter1', 'filter2', …
    MODE MANUAL |
    MODE AUTO num SECONDS | MINUTES | HOURS
```

### *4.1.3.5. Reglas*

Un dispositivo puede tener varias reglas que toman los datos de un sensor, comprueban una condición e invocan acciones de actuadores si la condición se cumple. Las reglas son útiles para automatizar la comunicación entre dispositivos.

Para introducir el bloque de reglas es necesario escribir la palabra *RULES* y a continuación definir cada regla que será precedida por las palabras *DEFINE RULE TO* seguidas de un sensor o sensores que el dispositivo tendrá que consultar.

El esqueleto de la definición de una regla completa es:

```
RULES
  DEFINE RULE TO ALL SENSORS
    FILTER BY 'filter1', 'filter2', …
    IF VALUE IS GREATER THAN -1
    EXECUTE ACTION 'action1'
      IN ACTUATOR 'actuator1'
    EXECUTE ACTION 'action2'
      IN ACTUATOR 'actuator2'
```

La sintaxis de una regla es más compleja que el resto del lenguaje y está compuesta por la fuente de los datos, la condición que los datos deben cumplir y las ejecuciones que se deben realizar si la condición se cumple.

La **fuente de los datos.** Los usuarios definen el sensor o sensores, internos o externos al dispositivo del que se tomará datos a utilizar en la regla. En las siguientes líneas se muestra las diferentes sintaxis.

- **Un sensor localizado en el mismo dispositivo**: Los usuarios deben escribir el nombre del sensor que hace de fuente de datos.

```
DEFINE RULE TO SENSOR 'name'
```

- **Varios sensores localizados en el mismo dispositivo**: Los usuarios deben indicar los filtros que esos sensores tienen en común.

```
DEFINE RULE TO ALL SENSORS
  FILTER BY 'filter1', 'filter2', …
```

- **Un sensor localizado en otro dispositivo**: los usuarios deben indicar el nombre del sensor y también los filtros para buscar el dispositivo donde el sensor está localizado.

```
DEFINE RULE TO EXTERNAL SENSOR 'name'
  FILTER BY 'filter1', 'filter2', …
```

- **Varios sensores localizados en otros dispositivos**: los usuarios tienen que escribir los filtros para buscar los dispositivos donde están localizados los sensores, pero, en este caso, no deben escribir el nombre de los sensores.

```
DEFINE RULE TO ALL EXTERNAL SENSORS
  FILTER BY 'filter1', 'filter2', …
```

**Condición a cumplir por los sensores.** Después de definir los sensores de la regla, los usuarios tienen que indicar la condición que sus datos deben cumplir. Para ello, deben usar la siguiente sintaxis escogiendo un único tipo de condición.

```
IF VALUE IS LESS THAN | LESS THAN OR EQUAL TO | EQUAL TO |
  NOT EQUAL TO | GREATER THAN OR EQUAL TO | GREATER THAN num
```

**Ejecuciones.** Los usuarios pueden definir varias acciones a ejecutar si la condición se cumple. Cada ejecución debe tener una acción y al menos un actuador que tenga esa acción. Una ejecución puede tener un actuador o varios actuadores, internos o externos al dispositivo donde se aplicará la acción. En las siguientes líneas se muestra como son las diferentes sintaxis:

- **Un actuador localizado en el mismo dispositivo**: Los usuarios tiene que indicar el nombre de la acción y el nombre del actuador que realizará la acción.

```
EXECUTE ACTION 'action'
  IN ACTUATOR 'actuator'
```

- **Varios actuadores localizados en el mismo dispositivo**: Los usuarios tiene que indicar los filtros que los actuadores tienen en común y el nombre de la acción que los actuadores realizarán.

```
EXECUTE ACTION 'action'
  IN ALL ACTUATORS
  FILTER BY 'filter1', 'filter2', …
```

- **Un actuador localizado en otro dispositivo**: En este caso, los usuarios tienen que indicarle nombre del actuador, los filtros por lo que buscar el dispositivo donde se localiza el actuador y el nombre de la acción a realizar.

```
EXECUTE ACTION 'action'
  IN EXTERNAL ACTUATOR 'actuator'
  FILTER BY 'filter1', 'filter2', …
```

- **Varios actuadores localizados en otros dispositivos**: Al igual que en el caso anterior, los usuarios tienen que escribir los filtros para buscar los dispositivos donde están localizados los actuadores, pero, además, deben indicar también los filtros que los actuadores tienen en común y también el nombre de la acción a realizar.

```
EXECUTE ACTION 'action'
  IN ALL EXTERNAL ACTUATORS
  FILTER BY 'filter1', 'filter2', …
```

En la Figura 5 se muestra la definición con BSL de un dispositivo a modo de ejemplo. Este dispositivo controla un ventilador y se comunica con otro dispositivo que controla la calefacción y dispone de un sensor de temperatura. El dispositivo definido invoca las acciones correspondientes de su ventilador y de la calefacción para apagar o encender ambos dispositivos bajo ciertas condiciones.

## 4.2. Prototipo propuesto: Bilrost

El prototipo Bilrost es una plataforma para conectar objetos inteligentes del mundo de Internet de las Cosas a través de redes sociales de personas y mediante el uso de un lenguaje de dominio específico y dos editores, uno textual y uno gráfico, que permiten definir el modelo de dos modos distintos.

Los proyectos generados por Bilrost permiten interconectan objetos ubicuos y heterogéneos usando Twitter como canal de comunicación. Estos proyectos generados pueden ser utilizados por los IDEs del lenguaje de programación correspondiente.

```
DEVICE IN PYTHON
  FILTER BY 'bilrost', 'rpi'
  SOCIAL NETWORKS
    CONNECT TO TWITTER
      TOKEN 'token'
      SECRET 'secret'
  ACTUATORS
    DEFINE ACTUATOR 'fan'
      ACTIONS 'on', 'off'
  SENSORS
    DEFINE SENSOR 'temperature'
      MODE AUTO 30 SECONDS
  RULES
    DEFINE RULE TO SENSOR 'temperature'
      FILTER BY 'bilrost', 'climate'
      IF VALUE IS GREATER THAN 25
      EXECUTE ACTION 'off'
        IN EXTERNAL ACTUATOR 'heating'
        FILTER BY 'bilrost', 'climate'
      EXECUTE ACTION 'on'
        IN ACTUATOR 'fan'
```

Figura 5. Ejemplo de dispositivo definido con BSL.

Los proyectos de aplicaciones generados no están completos y los usuarios deberán completarlos implementando la lógica específica que permite a las aplicaciones recopilar los datos de los sensores y manejar los actuadores de los objetos inteligentes.

En este apartado se describirá Bilrost primero a través de un enfoque general sobre cómo funciona y después se abordarán los principales componentes que forman su arquitectura.

Con el fin de explicar mejor como funciona Bilrost se recurrirá a un ejemplo sencillo que, aunque no muestra todas las posibilidades del BSL, es útil para entenderlo mejor.

**Ejemplo**: conectar una Raspberry Pi y un dispositivo Android a través de Twitter teniendo cada dispositivo un sensor y un actuador. La Raspberry Pi cuenta con un sensor de luz y un led rojo como actuador mientras que el dispositivo Android cuanta con el acelerómetro como sensor y el flash como sensor.

En la Figura 6 se puede ver el esquema general de los componentes que interactúan en la interconexión de dispositivos a través de redes sociales. El ejemplo que acabamos de introducir tiene componentes de todos esos tipos. La Raspberry Pi y el dispositivo Android son dispositivos y, por tanto, son representados con el número 2. El led rojo de la Raspberry Pi y el flash del dispositivo Android son actuadores y están representados con el número 3, mientras que el sensor de luz de la Raspberry Pi y el acelerómetro del dispositivo Android son sensores y están representados con el número 4. El número 1 muestra la red social que el caso del ejemplo es Twitter.

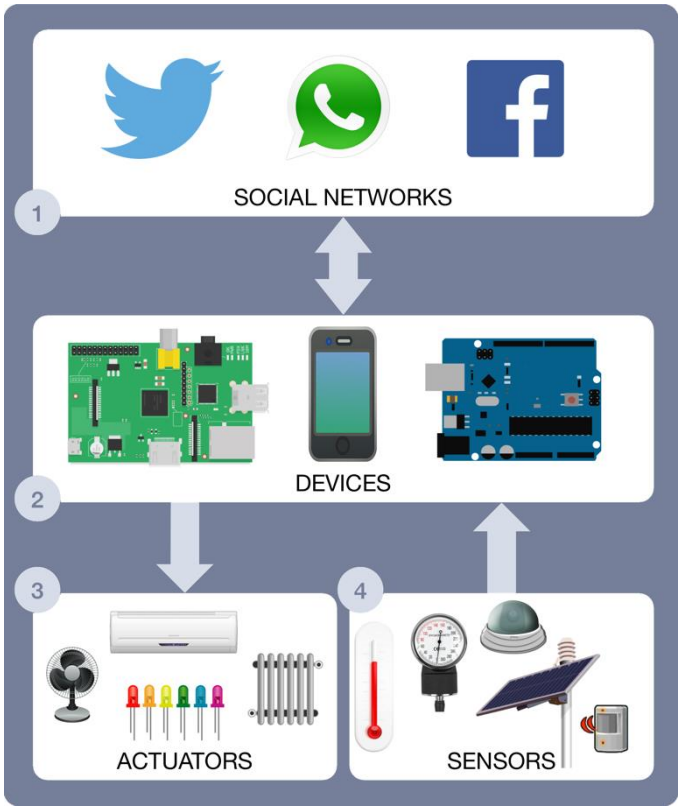



Figura 6. Componentes que intervienen en la interconexión de objetos a través de redes sociales.

### 4.2.1. Ciclo de trabajo de Bilrost

En este apartado se explicará cómo funciona Bilrost usando el ejemplo definido anteriormente. Pero antes de eso, es necesario definir el contexto donde los dispositivos del ejemplo pueden ser útiles.

El flash del dispositivo Android será una linterna que se encenderá cuando el sensor de luz de la Raspberry Pi detecte un nivel bajo de luz y el led de la Raspberry Pi será un indicador que se enciende cuando alguien esté pidiendo ayuda sacudiendo el acelerómetro del dispositivo Android.

Bilrost es capaz de hacer posible este ejemplo mediante el uso de BSL. Usando este lenguaje, un usuario puede definir un dispositivo indicando la tecnología que se usará en el proyecto generado como Android, Java o Python, las palabras claves que se usarán para buscar mensajes en Twitter o que se añadirán a los mensajes que contienen datos de los sensores, los datos necesarios por la API de cada red social, los actuadores del dispositivo con sus acciones, los sensores del dispositivo y las reglas que permiten automatizar la invocación de las acciones de los actuadores según los valores de los sensores y ciertas condiciones. Estas reglas estarán formadas por sensores y actuadores que pueden pertenecer al mismo dispositivo o dispositivos externos con lo que el dispositivo se comunicará a través de las redes sociales.

Los proyectos generados a partir de la definición de un dispositivo escrita en BSL no están completos. Aunque la conexión con Twitter ya esté implementada, los usuarios tendrán que implementar la lógica específica necesaria para acceder a los datos de los sensores o para controlar los actuadores ya que cada dispositivo, sensor y actuador tiene su funcionamiento específico.

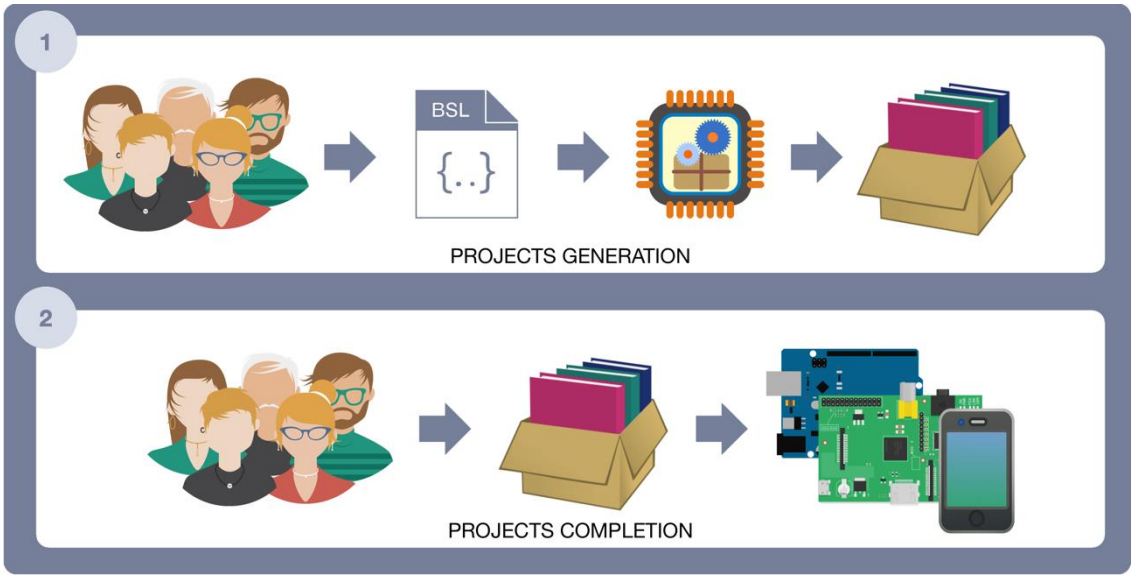



Figura 7. Pasos del ciclo de trabajo con interacción del usuario: 1. Generación de proyectos y 2. Completado de proyectos.

Debido a esto, el ciclo de trabajo de Bilrost se puede dividir en dos pasos en los que la interacción de los usuarios es necesaria. Como se puede ver en la Figura 7 los pasos son, la **generación de proyectos** y el **completado de proyectos**.

#### *4.2.1.1. Generación de proyectos*

El primer paso del ciclo de trabajo de Bilrost es la generación de proyectos de aplicaciones que conectan objetos inteligentes a las redes sociales como se puede ver en la Figura 7 representado con el número 1.

Los usuarios tienen que definir sus dispositivos usando la sintaxis de BSL. Los dispositivos del ejemplo anterior se podrían definir mediante la sintaxis BSL a través de uno de los dos editores que Bilrost pone a disposición del usuario, mediante el editor textual y mediante el editor gráfico. Este último define un dispositivo de manera gráfica creando un modelo gráfico que el sistema transforma al modelo textual antes de proseguir con la generación del proyecto mientras que el editor textual crea un modelo textual mediante la sintaxis de BSL.

Una vez concluida la definición de los dispositivos con BSL, Bilrost generará proyectos que cumplas las reglas definidas por los usuarios para cada dispositivo mediante el procesado de la definición del dispositivo escrita usando la sintaxis BSL. El usuario debe definir los dispositivos de uno en uno de manera que se genere un proyecto para desplegar en cada uno de los dispositivos.

Esta propuesta hace que la creación de aplicaciones que conecten objetos inteligentes a Twitter sea más fácil ya que abstrae la conexión a las redes sociales de la implementación. Los usuarios solo necesitan conocimientos básicos de cómo funcionan sus dispositivos, es decir, como leer los sensores y como controlar los actuadores además de conocer la sintaxis BSL.

Los proyectos generados usan la tecnología o lenguaje de programación que el usuario indicó mediante la definición del dispositivo en BSL con el fin de que sea más fácil para los usuarios implementar la lógica faltante y que además sea adaptable al dispositivo destino ya que no todos los dispositivos son compatibles con el mismo lenguaje de programación.

```
class LedActuator(ActuatorBase):

    def __init__(self):
        super().__init__('led')

    def on_action(self, params):
        # TODO Fill as you want
        pass

    def off_action(self, params):
        # TODO Fill as you want
        pass
```

Figura 8. Código de un actuador tras la generación del proyecto.

La Figura 8 muestra el código resultante de un actuador una vez generado el proyecto. Como se puede ver, en ese fragmento de código hay varios comentarios que indican al usuario donde debe completar la lógica específica que controla al actuador. A través de mensajes de Twitter y sin necesidad de que el usuario escriba código para ese fin, usuarios de la red social u otros objetos podrán llamar a esos métodos como se verá más adelante.

#### *4.2.1.2. Completado de proyectos*

El último paso es el completado de los proyectos. En este paso, los usuarios tienen que completar los proyectos generados por Bilrost. Estos proyectos se generan con un esqueleto de métodos en blanco que se corresponden con las acciones de los actuadores y los métodos de acceso a los datos de los sensores según lo que el usuario haya definido a través de la sintaxis BSL como se puede ver en la Figura 7. Este paso es un paso que debe realizar el usuario y donde Bilrost ya no hace tarea alguna ya que la propuesta de este trabajo fin de máster se basa en la interconexión de objetos a través de redes sociales en vez de en la generación de aplicaciones completas para objetos inteligentes. Automatizar este paso será trabajo futuro que será considerado para futuras investigaciones.

### 4.2.2. Arquitectura

La arquitectura de Bilrost se puede dividir en 4 capas, aunque no se tratan de capas físicas sino de 4 procesos en los que distintos componentes intervienen, unos forman parte de Bilrost y otros ajenos a él.

Las dos primeras capas son parte del prototipo desarrollado mientras que las otras dos capas no dependen del prototipo sino del propio usuario. Las 4 capas son: **Definición de código BSL**, **Generador de proyectos**, **Implementación de lógica específica** y **Despliegue de Objetos**.

**Definición de código BSL**

En la primera capa del sistema es donde se define código usando la sintaxis de BSL y en ellas se encuentran los editores web, tanto el gráfico como el textual, y el analizador de sintaxis BSL. El editor web gráfico permite definir un dispositivo mediante un modelo gráfico compuesto de elementos visuales y generar la definición del dispositivo en sintaxis BSL mediante una transformación de modelos, el editor web textual permite definir un dispositivo usando directamente la sintaxis BSL y el analizador de sintaxis BSL interpreta la definición del dispositivo escrita en BSL, es decir el modelo, generando un JSON que el generador de proyectos utilizará para crear los proyectos de acuerdo a las propiedades indicadas mediante la sintaxis BSL o mediante el editor web gráfico.

**Generador de proyectos**

En la segunda capa del sistema es donde se encuentra el generador de proyectos de aplicaciones. A esta capa le llega el resultado de la capa anterior, es decir, el JSON resultante del analizador de sintaxis BSL. El generador de proyectos procesa ese JSON y genera el proyecto apropiado que cumple con los requisitos definidos con la conexión con las redes sociales ya implementada y con un esqueleto para ayudar a los usuarios a completar la lógica específica de los actuadores y sensores.

**Implementación de lógica específica**

En la tercera capa ya no hay componentes del prototipo, sino que se encuentra el propio proyecto recibido de la capa anterior. En esta capa el usuario debe importar el proyecto al IDE correspondiente y completar la lógica específica del dispositivo que permita a la aplicación acceder a los datos de los sensores y controlar los actuadores definidos en las capas anteriores.

**Despliegue de Objetos**

En la cuarta capa se encuentran los objetos inteligentes soportados y los proyectos ya finalizados. En esta capa los usuarios deben generar las aplicaciones de la manera adecuada para el dispositivo y desplegarlas en él. Por ejemplo, si se ha definido un dispositivo Android se deberá generar el archivo APK correspondiente e instalarlo en el dispositivo.

En los siguientes apartados se explicarán los principales componentes del sistema propuesto, es decir, el editor web gráfico, el analizador de sintaxis BSL y el generador de proyectos. También se explicará cómo se debe completar la lógica específica.

#### *4.2.2.1. Editor web gráfico*

El editor web gráfico es un componente que los usuarios pueden usar modelar proyectos de aplicaciones. Se ha creado usando tecnologías web con el objetivo de ser accesible por cualquier usuario sin la necesidad de tener en cuenta el sistema operativo desde el que se accede.

El resultado obtenido del editor gráfico es la definición de un dispositivo escrito con sintaxis BSL que Bilrost ha generado a partir de los componentes del editor. Además del editor web gráfico también existe un editor web textual cuyo objetivo es el mismo, pero donde el usuario puede usar directamente la sintaxis BSL para definir el modelo.

La Figura 9 muestra el aspecto de este editor web gráfico con 4 zonas diferenciadas donde los usuarios pueden configurar las distintas partes de la definición de un dispositivo como se ha visto en el apartado 4.1.3. mediante la explicación de la sintaxis BSL.


Figura 9. Editor web gráfico de Bilrost.

**Device Box**


Figura 10. Editor gráfico - Device Box o Caja del dispositivo.

La Figura 10 muestra el bloque *Device Box* o caja del dispositivo, representado con el número 1 en la Figura 9. Este bloque permite configurar la tecnología o lenguaje de programación que el proyecto generado deberá usar y las palabras claves, aquí llamados filtros, que identifican al dispositivo en las redes sociales. En el caso de Twitter son *hashtags*. Ambos campos son obligatorios para generar la definición de un dispositivo en BSL.

**Social Networks Box**

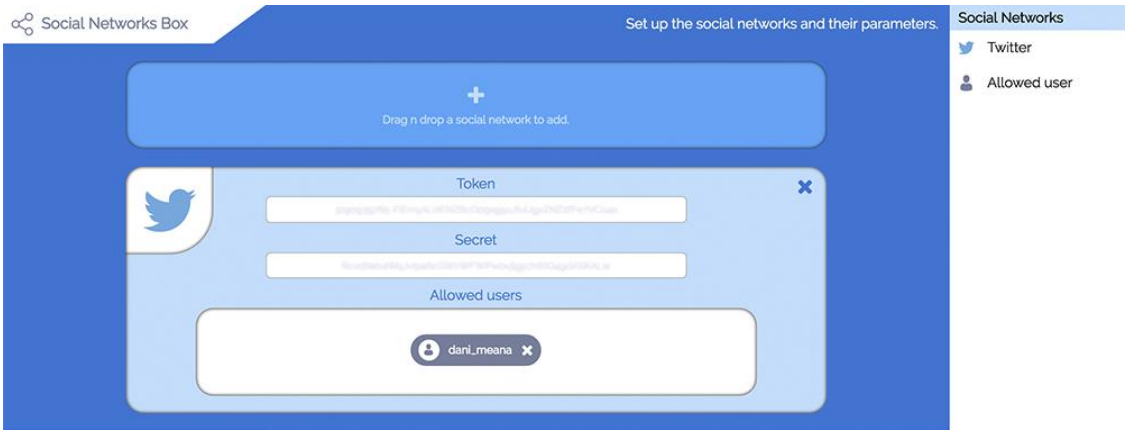



Figura 11. Editor gráfico - Social Networks Box o Caja de redes sociales.

La Figura 11 muestra el bloque *Social Networks Box* o caja de redes sociales, representando con el número 2 en la Figura 9 . Este bloque contiene la configuración de las redes sociales que el dispositivo usará, es decir, los parámetros necesarios por sus APIs y los usuarios que tienen permisos para comunicarse con el dispositivo. Un usuario puede ser una persona u otro dispositivo ya que se refiere a usuario de la red social y no a persona. El prototipo desarrollado solo es compatible con Twitter, aunque está preparado para añadir más redes sociales de manera sencilla ya que se contempla como trabajo futuro investigar qué red social se adapta mejor a las necesidades de los objetos.

**Objects Box**

La Figura 12 muestra el bloque *Objects Box* o caja de objetos, y está representando con el número 3 en la Figura 9. Los usuarios pueden usar este bloque para añadir sensores o actuadores a su dispositivo, así como establecer las propiedades de cada uno de estos objetos. Un sensor necesita un nombre, un modo de trabajo entre manual y automático y opcionalmente unos filtros que permitan identificarlo mejor en las redes sociales. Y un actuador necesita un nombre, una lista de acciones que puede realizar y que pueden ser ejecutadas a través de las redes sociales y opcionalmente unos filtros que permitan identificarlo mejor en las redes sociales a la vez que permitir ejecuciones simultaneas de varios actuadores que tengan el mismo filtro y acciones como se verá más adelante.

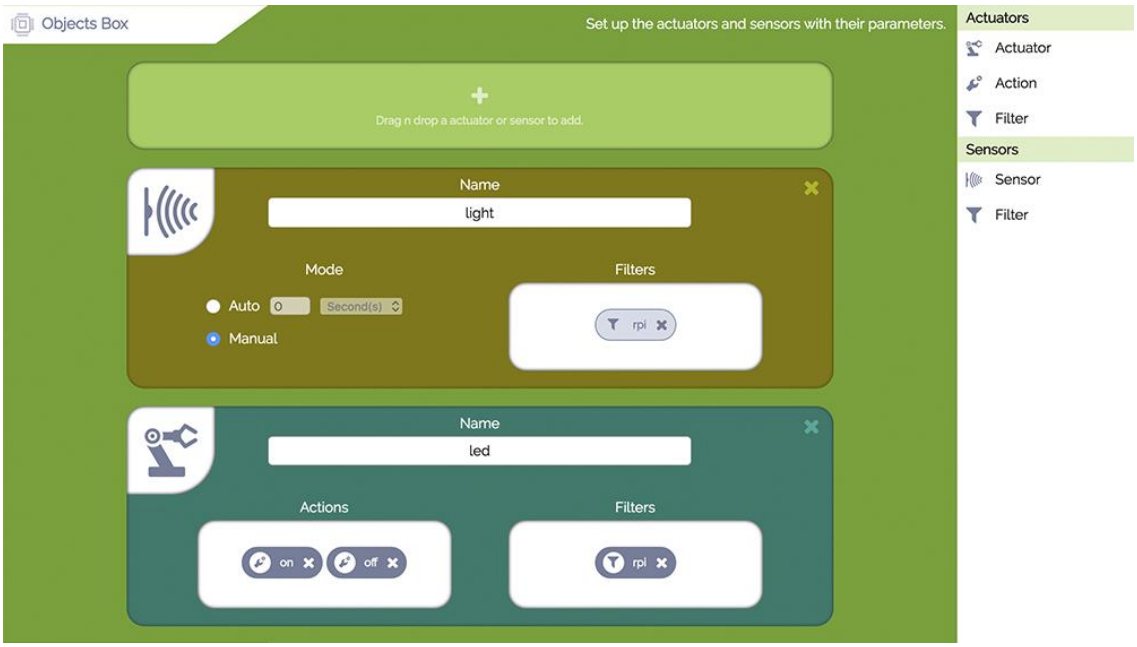



Figura 12. Editor gráfico - Objects Box o Caja de objetos.

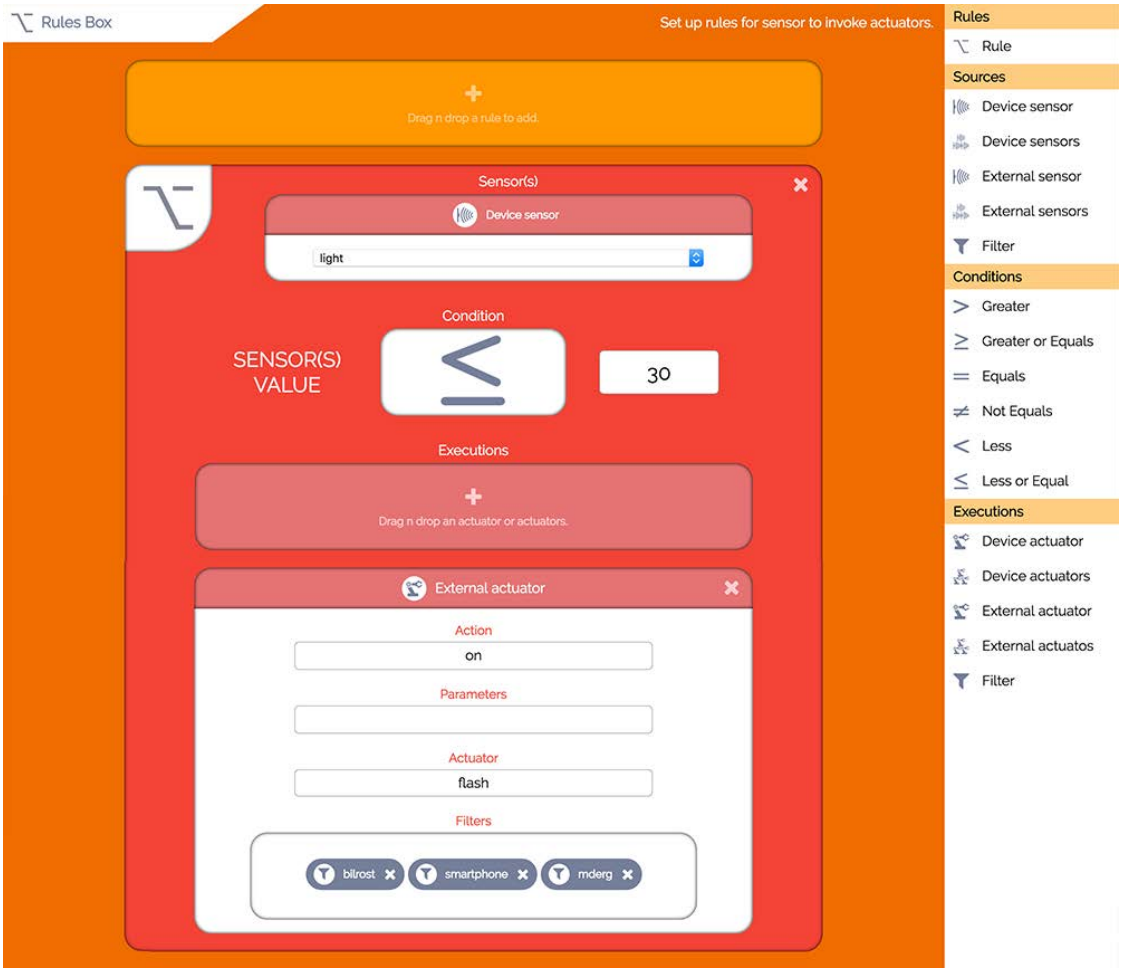



Figura 13. Editor gráfico - Rules Box o Caja de reglas.

**Rules Box**

La Figura 13 muestra el bloque *Rules Box* o caja de reglas, y está representando con el número 4 en la Figura 9. Este bloque es el más complejo de los 4 y puede ser usado para añadir reglas que automaticen tareas basadas en reglas.

Cada regla tiene 3 partes diferentes: la fuente de datos que se corresponde con un sensor si se indica su nombre o con un grupo de sensores si se indican filtros en vez del nombre, la condición que los datos recopilados de la fuente de datos deben cumplir y las ejecuciones que se realizarán si la condición se cumple.

Para explicar las reglas se va a utilizar el ejemplo definido al principio del apartado 4.2. Un usuario podría añadir una regla cuya fuente de datos es el sensor de luz de la Raspberry Pi, la condición podría ser «si el valor del sensor está por debajo de cierto valor» y una ejecución podría ser llamar a la acción «encender» del actuador «flash» del dispositivo Android. Si esta regla se establece en la definición de la Raspberry Pi se estaría usando un sensor del dispositivo e invocando una acción de un dispositivo externo a través de Twitter y si la regla se establece en la definición del dispositivo Android se estaría usando un sensor de un dispositivo externo por lo que se estaría leyendo el valor a través de Twitter e invocando una acción de un actuador del propio dispositivo. Teniendo esto en cuenta, un usuario podría definir un dispositivo que no tuviera sensores ni actuadores, pero si reglas de manera que funcionaría como un orquestador de acciones que invocarías acciones externas cuando un sensor externo tiene un valor concreto.

Además, los editores permiten guardar la definición escrita en BSL a un fichero local para poder cargarla más tarde con el fin de poder modificarla posteriormente.

Es importante mencionar que usando el editor gráfico el usuario no necesita conocer la sintaxis BSL ya que el editor gráfico es capaz de generar proyectos sin escribir una sola línea de código BSL, aunque existe la posibilidad de ver la definición en BSL una vez completada la definición con el editor gráfico.

### *4.2.2.2. Analizador de la sintaxis BSL*

La entrada del analizador de la sintaxis BSL es la definición de un dispositivo escrito en BSL. Esta definición puede venir tanto del editor web gráfico como del editor web textual, o incluso de un fichero local usando línea de comandos.

```
DEVICE IN PYTHON
  FILTER BY 'bilrost', 'rpi'
  SOCIAL NETWORKS
    CONNECT TO TWITTER
      TOKEN 'token'
      SECRET 'secret'
  ACTUATORS
    DEFINE ACTUATOR 'led'
      ACTIONS 'on'
  SENSORS
    DEFINE SENSOR 'light'
      MODE MANUAL
  RULES
    DEFINE RULE TO SENSOR 'light'
      IF VALUE IS LESS THAN 30
      EXECUTE ACTION 'on'
        IN EXTERNAL ACTUATOR 'flash'
        FILTER BY 'bilrost', 'smartphone'
```

Figura 14. Definición de la Raspberry Pi del ejemplo en BSL.

```
DEVICE IN ANDROID
  FILTER BY 'bilrost', 'smartphone'
  SOCIAL NETWORKS
    CONNECT TO TWITTER
      TOKEN 'token'
      SECRET 'secret'
  ACTUATORS
    DEFINE ACTUATOR 'flash'
      ACTIONS 'on'
  SENSORS
    DEFINE SENSOR 'shake'
    MODE MANUAL
  RULES
    DEFINE RULE TO SENSOR 'shake'
      IF VALUE IS EQUAL TO 1
      EXECUTE ACTION 'on'
        IN EXTERNAL ACTUATOR 'led'
        FILTER BY 'bilrost', 'rpi'
```

Figura 15. Definición del dispositivo Android del ejemplo en BSL.

El analizador efectúa tres pasos con el fin de entender la definición del dispositivo y generar un JSON que el generador de proyectes pueda usar para generar proyectos. Estos pasos son el análisis léxico, el análisis sintáctico y la generación del JSON. Para realizar estos pasos se han usado las implementaciones en Python de las herramientas de análisis *lex* y *yacc*.

A partir del ejemplo definido anteriormente, se puede deducir que hay que definir dos dispositivos usando BSL. En la Figura 14 se puede ver la definición de la Raspberry Pi en BSL y en la Figura 15 se puede ver la definición del dispositivo Android en BSL. En un apartado posterior se explicarán todos los componentes de la sintaxis BSL.

### *4.2.2.3. Generador de proyectos*

El generador de proyectos es el componente encargado de analizar el JSON generado en el analizador de la sintaxis BSL y extraer los datos necesarios para generar un proyecto que cumpla los requisitos.

En función de los datos del JSON, el generador de proyectos selecciona una plantilla y la completa con los datos extraídos del JSON hasta obtener un proyecto con un esqueleto definido con los actuadores, las acciones de los actuadores y los sensores definidos, además de tener implementada la conexión con las redes sociales.

Actualmente, el prototipo desarrollado es capaz de generar proyectos en 3 tecnologías distintas: Python, Java y Android. En el caso de Java, se crea un proyecto con la estructura necesaria para el IDE IntelliJ IDEA y en el caso de Android, se crea un proyecto con la estructura necesaria para el IDE Android Studio.

El proyecto resultante tiene un esqueleto con métodos vacíos, tal y como se vio en la Figura 8.

En la Figura 8 se puede ver la clase Python de un actuador llamado «Led». Se trata del actuador de la Raspberry Pi usado en el ejemplo. La clase tiene dos métodos que se corresponden con las dos acciones, el método para encender el led y el método para apagarlo. Estos métodos se invocarán de manera automática cuando el dispositivo lea en Twitter un mensaje que encaje con sus filtros y con las características del actuador.

```
import RPi.GPIO as GPIO

class LedActuator(ActuatorBase):

    def __init__(self):
        super().__init__('led')
        self.port = 6
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(self.port, GPIO.OUT)

    def on_action(self, params):
        GPIO.output(self.port, True)

    def off_action(self, params):
        GPIO.output(self.port, False)
```

Figura 16. Ejemplo de implementación de un actuador.

#### *4.2.2.4. Implementación específica*

La salida del componente generador de proyectos es un esqueleto de aplicación donde la conexión con las redes sociales ya está implementada pero que el usuario debe completar con la implementación específica de cada sensor, actuador y dispositivo con el fin de permitir a la aplicación controlar los actuadores y acceder a los datos de los sensores de manera automática o implementar la lógica que invoque al método que publica los datos del sensor en las redes sociales si se ha configurado el sensor como manual.

La Figura 16 muestra un ejemplo de una implementación final de un actuador. El código de la Figura 16 se corresponde con el esqueleto de la Figura 8 y es el actuador de las Raspberry Pi definida en el ejemplo usado. Este código implementa el encendido y apagado de un led situado en el PIN 6 de una Raspberry Pi.

### 4.2.3. Comunicación a través de Twitter

La comunicación a través de Twitter se realiza por medio de la publicación de mensajes o tuits en la línea temporal de Twitter. La estructura de cada tuit depende del objeto con el que interacciona, es decir, los mensajes o tuits serán diferentes si proceden de un sensor o si es un mensaje destinado a ejecutar una acción de un actuador. Sin embargo, ambos mensajes tienen una parte común, deben contener los filtros del dispositivo que contiene el objeto con el que interactúan. Para explicar cómo funciona la comunicación a través de Twitter se pueden clasificar los mensajes en dos grupos: **mensajes para controlar un actuador** y **mensajes que publican los datos de un sensor**.

#### *4.2.3.1. Mensajes para controlar un actuador*

La estructura de los mensajes para controlar un actuador está compuesta de los filtros del dispositivo, los filtros del actuador si los tiene, el nombre del actuador, la acción a llamar y los parámetros que la acción necesita si los necesita. Además, los mensajes pueden contener más texto plano sin interferir en el correcto funcionamiento del sistema.

Los filtros y el nombre del actuador van precedidos del símbolo almohadilla (#) debido a que las palabras clave de Twitter o *hashtags* van precedidos de ese símbolo. Sin embargo, las acciones son texto plano y los parámetros de la acción texto entre comillas. El resto del mensaje es ignorado sin importar su contenido.

Cuando la publicación se hace manera automática como fruto de la automatización de las reglas, junto a lo ya mencionado se publica el *timestamp* con el fin de que los tuits sean todos únicos y evitar así las restricciones de Twitter de publicación de dos tuits idénticos.

En las siguientes líneas se muestran ejemplos de tuits que manejan los actuadores definidos:

- **#bilrost #rpi #led on** – Invoca la acción llamada *on* del actuador llamado *led* del dispositivo que responde ante los filtros *bilrost* y *rpi*.
- **#bilrost #smartphone #flash on** – Invoca la acción llamada *on* del actuador llamado *flash* del dispositivo que responde ante los filtros *bilrost* y *smartphone.*
- **#bilrost #uniovi #climate #thermostat set "22"** – Invoca la acción llamada *set* del actuador llamado *thermostat* que tiene un filtro *climate* del dispositivo que responde ante los filtros *bilrost y uniovi*. Además, aparece entre comillas los parámetros de la acción *set* que en este caso es el valor 22. Los parámetros deben ir entre comillas y es el usuario el que se encarga de implementar como se deben tratar los parámetros en la fase del completado del proyecto.

Los filtros y el nombre del actuador se representan de la misma manera y por tanto no es posible identificar si el *hashtag* se corresponde con un nombre, un filtro de dispositivo o un filtro del actuador. Sin embargo, el sistema de procesado de mensajes de las redes sociales realiza una búsqueda ordenada de manera que primero comprueba que el tuit contenga los filtros del dispositivo y si así es se procede a la búsqueda del nombre del actuador entre los del dispositivo previa eliminación de los filtros del dispositivo del mensaje. Si se encuentra un actuador cuyo nombre aparece en lo que queda del mensaje y que pueda realizar la acción, se selecciona ese actuador. Sin embargo, en el caso de que no aparezca el nombre del actuador en el mensaje, se realizarían la acción en todos los actuadores que dispongan de ella. De esta manera, si el analizador llega a completar todos los pasos y obtiene una invocación completa la efectúa, ya sea a nivel de un único actuador o de varios actuadores.

Para realizar una ejecución de varios actuadores se puede ignorar el nombre del actuador de manera que el dispositivo buscará todos los actuadores que puedan ejecutar dicho método como se puede ver en el siguiente ejemplo. Por tanto, el nombre del actuador es opcional.

- **#bilrost #uniovi #lights on** – Invoca la acción llamada *on* de todos los actuadores que tengan el filtro *lights* del dispositivo que responde ante *bilrost* y *uniovi*.

Como Twitter tiene la restricción de no permitir la publicación de tuits idénticos por el mismo usuario, cuando una regla definida en un dispositivo invoca una acción a través de Twitter, se añade el *timestamp* como se puede ver en el siguiente ejemplo.

- **#bilrost #uniovi #screen show "Hello, my value is 1" 2016-04-05T10:18:36.898989** – Invoca la acción llamada *show* del actuador llamado *screen* del dispositivo que responde ante *bilrost* y *uniovi* y usando como parámetro «*Hello, my value is 1*».

#### *4.2.3.2. Mensajes que publican los datos de un sensor*

La estructura de los mensajes que publican los datos de un sensor en Twitter, están compuestos por los filtros del dispositivo, los filtros del sensor si los hubiese, el nombre del sensor y los datos del sensor. Además, los sensores publican el *timestamp* con el fin de evitar la restricción de Twitter de no poder publicar dos mensajes iguales.

Los filtros y el nombre del sensor van precedidos del símbolo almohadilla (#) debido a que las palabras clave de Twitter o *hashtags* van precedidos de ese símbolo y los datos del sensor entre comillas. El resto del mensaje es irrelevante y los dispositivos que lean el mensaje lo ignorarán.

En las siguientes líneas se muestran ejemplos de tuits que comparten el estado de los sensores del ejemplo definido.

- **#bilrost #rpi #light "50" 2016-04-05T10:18:24.571613** – Muestra que el sensor llamado *light* del dispositivo cuyos filtros son *bilrost* y *rpi* ha detectado el valor 50. Esto significa que el sensor de luz de la Raspberry Pi ha detectado que el nivel de luz es 50.
- **#bilrost #smartphone #shake "1" 2016-04-05T10:18:42.649773** – Muestra que el sensor llamado *shake* del dispositivo cuyos filtros son *bilrost* y *smartphone* ha detectado el valor 1. Esto significa que el acelerómetro del dispositivo Android ha detectado una sacudida.

Para que un sensor publique un valor es necesario que el usuario finalice el proyecto en la fase de completado del proyecto. Si el sensor se definido como automático se debe rellenar un método del que ya existe su esqueleto y que debe devolver exactamente lo que el usuario quiere que se publique en Twitter. El dispositivo llamará a ese método según la frecuencia configurada. Y si el sensor se ha definido como manual, el usuario deberá llamar a un método del sensor pasándole por parámetro lo que quiere que se publique en Twitter cuando él lo desee.

El sistema de filtros funciona de la misma manera que el de los actuadores. El sistema buscará sensores que cumplan los filtros o el nombre de este y evaluará su valor con la condición definida en la regla. Si la condición es correcta se ejecutarán las ejecuciones mediante la publicación de mensajes que controlan a un actuador si el actuador es externo ya que en caso de es interno la llamada se hace directamente sin pasar por Twitter.

### 4.2.4. Software y hardware utilizados

Con el fin de desarrollar este prototipo se han necesitado diferentes tipos de componentes software y hardware. Los componentes software utilizados son los siguientes:

- Los editores web fueron escritos usando JavaScript ES6 y usando la librería ReactJS 15.0.2. Además, se ha usado Webpack 1.13 junto a Babel para traducir el código a ES5, con el fin de que sea compatible con todos los navegadores actuales, y para crear un paquete con todo el código.
- El analizador de sintaxis BSL fue escrito usando Python 3.5.1 y usando la librería Ply 3.8.
- Los proyectos generados fueron escritos con Python 3.5.1, Java 8 y Android 5.1.1 aunque hay retro-compatibilidad con Python 3.x y Android 4.x.
- El proyecto generado en Python usa la librería Twython 3.4.0.
- Los proyectos generados de Java y Android usan las librearías Gson 2.6.2 y Twitter 4j 4.0.4.

Con fines de probar la propuesta se ha usado distintos componentes hardware entre los que destacar una Raspberry Pi 2 con varios componentes electrónicos conectados a sus puertos GPIO y un Galaxy Note 4 con Android 5.1 Lollipop.

# Capítulo 5. Evaluación del Sistema

En este capítulo se presentará el proceso seguido para evaluar la propuesta de este trabajo fin de máster y así afirmar si la hipótesis planteada es cierta o no. Para ello se describirá la metodología seguida, se mostrarán los resultados obtenidos y se hará una discusión sobre el significado de estos datos.

El objetivo de la propuesta planteada es comprobar si es posible facilitar la creación de aplicaciones que interconecten objetos inteligentes entre sí y con personas a través de redes sociales. Por otro lado, también se analizará a partir de los resultados obtenidos, cuál de las dos herramientas presentadas para definir dispositivos, los dos editores, es mejor para el objetivo de este proyecto fin de máster.

## 5.1. Metodología

La metodología seguida para la evaluación tiene dos propósitos diferentes: recopilar datos que permitan tomar la decisión de que editor es mejor para el objetivo planteado y recopilar opiniones de los usuarios de las que se pueda deducir si la propuesta planteada hace más sencilla la creación de aplicaciones que su desarrollo de la manera tradicional. Para lograr estos dos propósitos, se han reclutado 10 voluntarios con conocimientos sobre el desarrollo de aplicaciones para realizar la evaluación de manera anónima.

La metodología seguida se puede dividir en dos fases diferentes que recopilan los datos para cada uno de los propósitos que se acaban de plantear.

- **Fase 1**: En esta fase, se quiere recopilar datos sobre el uso que hacen los participantes de los dos editores para completar una tarea propuesta para la evaluación. Esta tarea es un caso de uso básico sobre la definición de un dispositivo que se debe comunicar con otro dispositivo ya existente, con el fin de realizar una tarea colaborativa.
- **Fase 2**: En esta fase, se quiere recopilar las opiniones de los usuarios que han utilizado el sistema en la fase anterior mediante una encuesta basada en la escala de Likert (Likert, 1932).

### 5.1.1. Fase 1

Antes de realizar esta fase, se ha definido un escenario que simula un escenario real en el que un usuario quiere conectar un nuevo dispositivo con otro dispositivo ya existente con el objetivo de automatizar ciertas acciones.

El escenario definido trata del desarrollo de una aplicación que pueda controlar la temperatura de una habitación mediante el control del aire acondicionado y de la calefacción, y mediante la medición de la temperatura con un sensor de temperatura. El ciclo de trabajo consiste en encender la calefacción y apagar el aire acondicionado cuando la temperatura sea inferior a cierto valor, y apagar la calefacción y encender el aire acondicionado cuando la temperatura sea superior a cierto valor.

Sin embargo, según la tarea que se propuso a los usuarios, estos no tenían que definir todo el sistema planteado, sino que solo tenían que considerar que el dispositivo que puede controlar la calefacción ya estaba definido y listo para ser controlado a través de Twitter.

La tarea definida consiste en definir un único dispositivo capaz de controlar el aire acondicionado y de medir la temperatura. Por otro lado, existe otro dispositivo, que ya está en funcionamiento, capaz de controlar la calefacción. El dispositivo definido debe ser capaz de automatizar el encendido y el apagado, tanto de la calefacción como del aire acondicionado, de acuerdo a la temperatura de la habitación. Si la temperatura es menor de 16ºC, el aire acondicionado tiene que apagarse y la calefacción tiene que encenderse. Por otro lado, si la temperatura es superior a 25ºC, el aire acondicionado tiene que encenderse y la calefacción tiene que apagarse. La temperatura deberá comprobarse cada 30 segundos.

Los participantes deberán tener en cuenta que el dispositivo ya definido usa los filtros *bilrost*, *evaluation*, and *external*, y controla la calefacción mediante un actuador llamado *heating* y cuyas acciones son *on* y *off*.

Para recoger datos sobre el uso del sistema por los participantes, se les ha proporcionado una descripción sobre cómo se describe un dispositivo que tuvieron que leer sin un límite de tiempo establecido. Durante esta lectura, los participantes podían realizar preguntas sobre dudas que les surgieses sobre el funcionamiento del sistema. Además, en esa descripción, también existe un ejemplo donde se definen dos dispositivos que están interconectados con el fin de mejorar la compresión del sistema.

Tras leer la descripción, se les ha enseñado el sistema y otorgado tiempo para probarlo con el objetivo de intentar eliminar de los resultados el tiempo de la curva de aprendizaje. Una vez que los usuarios confirmen que han entendido el sistema y que están listos, se les entrega la tarea que deben realizar y se les da más tiempo para leerla y razonar como llevarla a cabo, pero sin tener acceso al sistema.

Finalmente, cuando los participantes estén listos, se empiezan a medir los tiempos. Se realizan dos mediciones, el tiempo empleado en realizar la tarea con el editor textual y el tiempo empleado en realizar la tarea con el editor gráfico. Con el fin de comparar como afecta a los resultados el editor usado en primer lugar, los participantes se alteran el editor que usarán en primer lugar. Así, los participantes impares usaron primero el editor textual y los participantes pares el editor gráfico. Durante toda la prueba, los participantes tenían acceso a la descripción dada al principio de ella y que se puede consultar en el Anexo II.

Una vez que los participantes finalizan adecuadamente la tarea en ambos editores, comienza la segunda fase de la evaluación.

### 5.1.2. Fase 2

Una vez que los participantes hayan finalizado la tarea propuesta, rellenan la encuesta creada de manera anónima y sin ayuda.

Con el objetivo de medir y evaluar la encuesta, se ha decido usar la escala de Likert (Likert, 1932) ya que es una de las más usadas en el diseño de encuestas. La encuesta creada es una escala de Likert de 5 puntos que presenta a los participantes las siguientes opciones: 1 como «completamente en desacuerdo», 2 como «en desacuerdo», 3 como opción neutral, 4 como «de acuerdo» y 5 como «complemente de acuerdo».

La encuesta contiene 14 declaraciones sobre las que se pregunta a los participantes su opinión tras haber usado los dos editores. Las declaraciones están relacionadas con las posibilidades del sistema propuesto, su posible impacto en Internet de las Cosas y su posible impacto en las redes sociales. Las declaraciones se pueden ver en la Tabla 1.

| Pregunta | Descripción |
|---|---|
| **Q1** | El usuario entiende la funcionalidad de los elementos de ambos editores y del DSL y sus roles en el proceso de creación de aplicaciones. |
| **Q2** | Este DSL, a través de ambos editores, permite interconectar dispositivos entre sí y con personas a través de redes sociales de manera sencilla, usando pocas líneas de código y en poco tiempo. |
| **Q3** | Usando este DSL, a través de ambos editores, es más difícil cometer errores durante el modelado de aplicaciones que haciéndolo todo desde cero. |
| **Q4** | La solución propuesta ofrece una manera rápida de desarrollar la tarea indicada. |
| **Q5** | La solución propuesta ayuda a la creación de aplicaciones en las que hay objetos interconectados a través de redes sociales. |
| **Q6** | La solución propuesta ayuda a la gente a ser capaces de interactuar con los dispositivos a través de las redes sociales. |
| **Q7** | El editor TEXTUAL no requiere conocimientos de programación complejos. |
| **Q8** | El editor GRÁFICO no requiere conocimientos de programación complejos. |
| **Q9** | El DSL y ambos editores, incluyen suficientes elementos y funcionalidades para que el usuario pueda crear una gran numero de aplicaciones en las que se interconecten objetos a través de redes sociales |
| **Q10** | El DSL y ambos editores, incluyen suficientes elementos y funcionalidades para que el usuario pueda crear una gran numero de aplicaciones en las que la gente pueda interactuar con dispositivos a través de redes sociales. |
| **Q11** | Esta solución es una contribución positiva para mejorar el desarrollo de servicios y aplicaciones que necesiten interconectar objetos entre sí o con personas. |
| **Q12** | Internet de las Cosas puede beneficiarse de esta solución. |
| **Q13** | Las redes sociales pueden beneficiarse de esta solución. |
| **Q14** | El editor gráfico hace que la creación de la interconexión a través de redes sociales sea más fácil que con el editor textual. |

Tabla 1. Encuesta realizada a los participantes en la evaluación.

## 5.2. Resultados

Una vez hecha la evaluación con los participantes, se han obtenido datos de ambas fases. En los siguientes apartados se mostrarán los resultados obtenido de las mediciones de cada fase.

### 5.2.1. Fase 1

La Figura 17 muestra el tiempo que cada participante ha empleado en completar la tarea con cada editor y la media de todos los participantes con cada editor. El usuario que menos tiempo ha invertido en realizar la tarea con el editor textual ha empleado 294s mientras que el usuario que el menor tiempo invertido en el editor gráfico fue 201s. El mayor tiempo empleado en realizar la tarea con el editor textual fue 672s mientras que con el editor gráfico fue 730s.

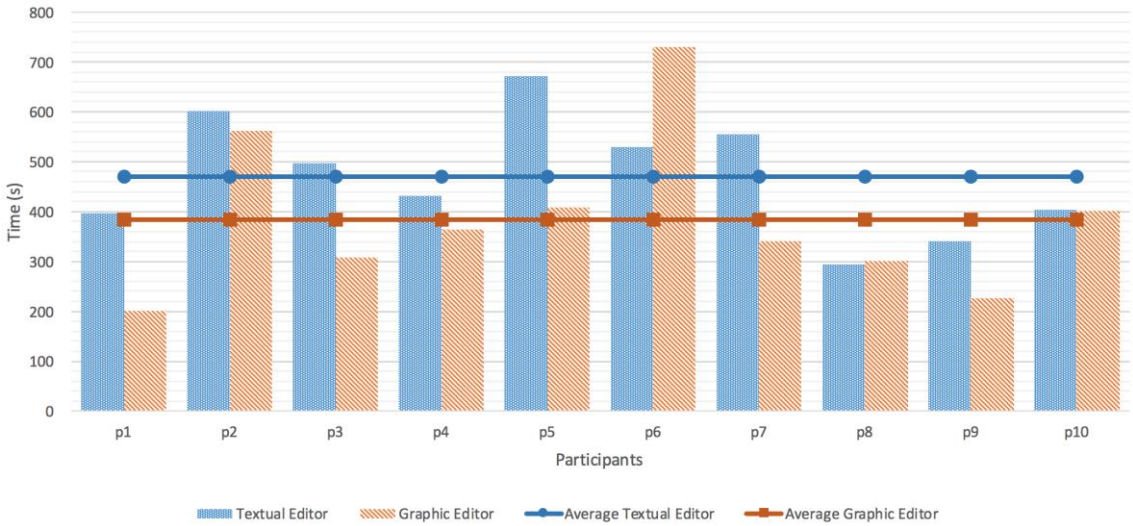



Figura 17. Tiempos empleados por cada participante en completar la evaluación.

La media de tiempo invertido en el editor textual fue 471,5s y la del editor gráfico fue 383,9s. La desviación estándar de los tiempos del editor textual fue 113,48s mientras que la del editor gráfico fue 150,16s.

También se han recogido datos agrupados en función del editor que los participantes hayan usado en primer lugar con el fin de saber si enfrentarse primero a uno u otro puede influir en los resultados. Estos datos se pueden visualizar en la Figura 18.

El grupo que ha empezado con el editor textual ha empleado una media de 491,4s en el editor textual, con un máximo de 672s, un mínimo de 339s y una desviación típica de 117,55s. Este grupo ha empleado en el editor gráfico un tiempo medio de 296,6s, con un máximo de 408s, un mínimo de 201s y una desviación típica de 75,65s.

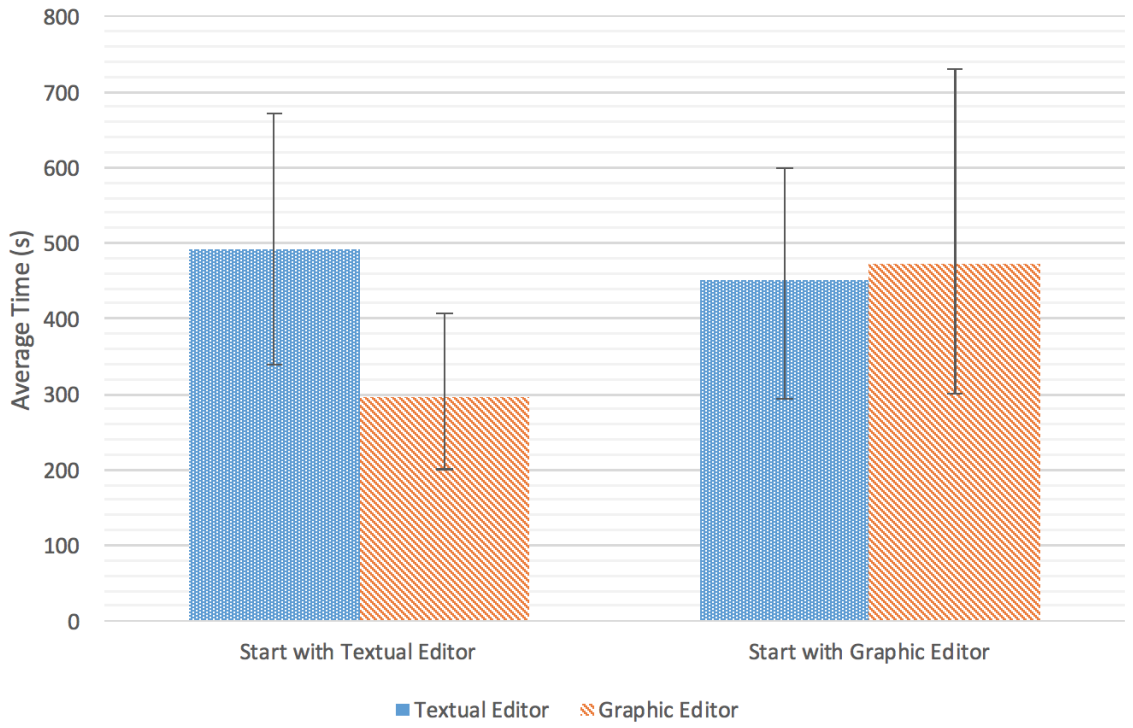



Figura 18. Tiempo medio usado para completar la tarea en función del primer editor usado.

| | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 | Q8 | Q9 | Q10 | Q11 | Q12 | Q13 | Q14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Min** | 3 | 3 | 4 | 4 | 4 | 3 | 2 | 3 | 3 | 3 | 4 | 4 | 2 | 3 |
| **Cuartil 1** | 4,25 | 4 | 4 | 4 | 4 | 4 | 3,25 | 4 | 4 | 3 | 4 | 4 | 3 | 4,25 |
| **Mediana** | 5 | 4,5 | 4,5 | 5 | 4,5 | 4,5 | 4 | 5 | 4 | 4 | 4,5 | 5 | 3,5 | 5 |
| **Cuartil 3** | 5 | 5 | 5 | 5 | 5 | 5 | 4,75 | 5 | 4,75 | 4,75 | 5 | 5 | 5 | 5 |
| **Max** | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| **Rango** | 2 | 2 | 1 | 1 | 1 | 2 | 3 | 2 | 2 | 2 | 1 | 1 | 3 | 2 |
| **Inter Qrt.- Rango** | 0,75 | 1 | 1 | 1 | 1 | 1 | 1,5 | 1 | 0,75 | 1,75 | 1 | 1 | 2 | 0,75 |
| **Moda** | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 4 | 3 | 5 | 5 | 5 | 5 |

Tabla 2. Estadística descriptiva obtenida de las respuestas de las encuestas.

El otro grupo, los usuarios que empezaron con el editor gráfico, ha empleado un tiempo medio de 451,6s en el editor textual, con un máximo de 600s, un mínimo de 294s y una desviación típica de 105,58s. Este grupo, ha empleado en el editor gráfico, un tiempo medio de 471,2s, con un máximo de 730s, un mínimo de 301s y una desviación típica de 155,33s.

### 5.2.2. Fase 2

La Tabla 2 muestra la estadística descriptiva obtenida de las encuesta hechas por los participantes. Esta gráfica muestra el desglose de cada pregunta: el mínimo, el primer cuartil, la mediana, el tercer cuartil, el máximo, el rango (diferencia entre máximo y mínimo), el rango entre cuartiles y la moda. Por otro lado, también se presenta la misma información en un diagrama de cajas y bigotes en la Figura 19.

Además, en la Figura 20 se puede ver un gráfico que representa la frecuencia de las respuestas de cada pregunta.

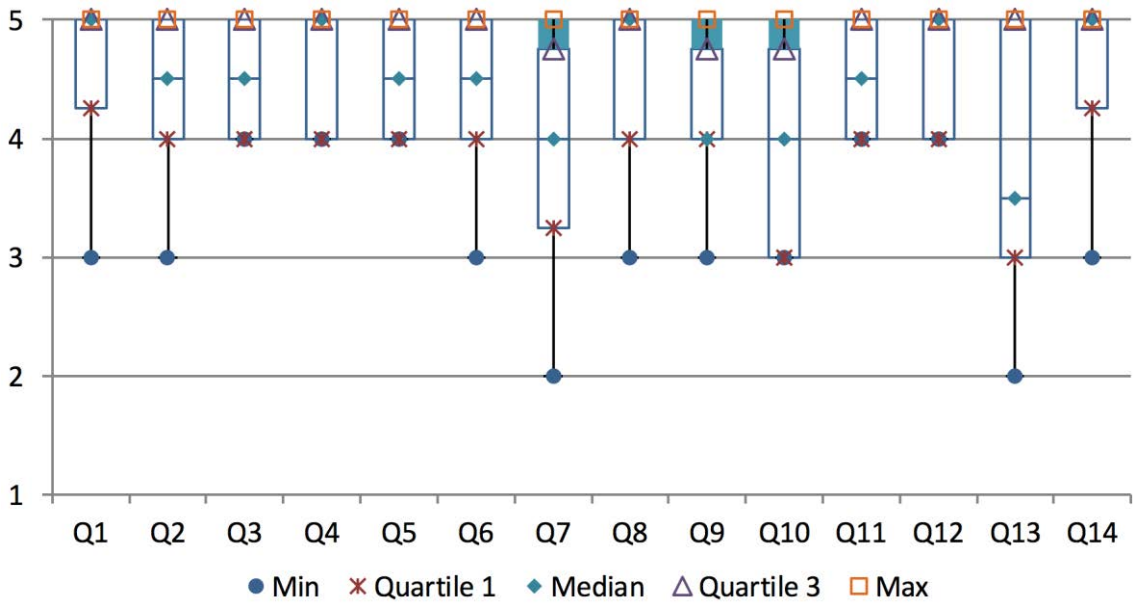



Figura 19. Diagramas de cajas y bigotes de cada pregunta.

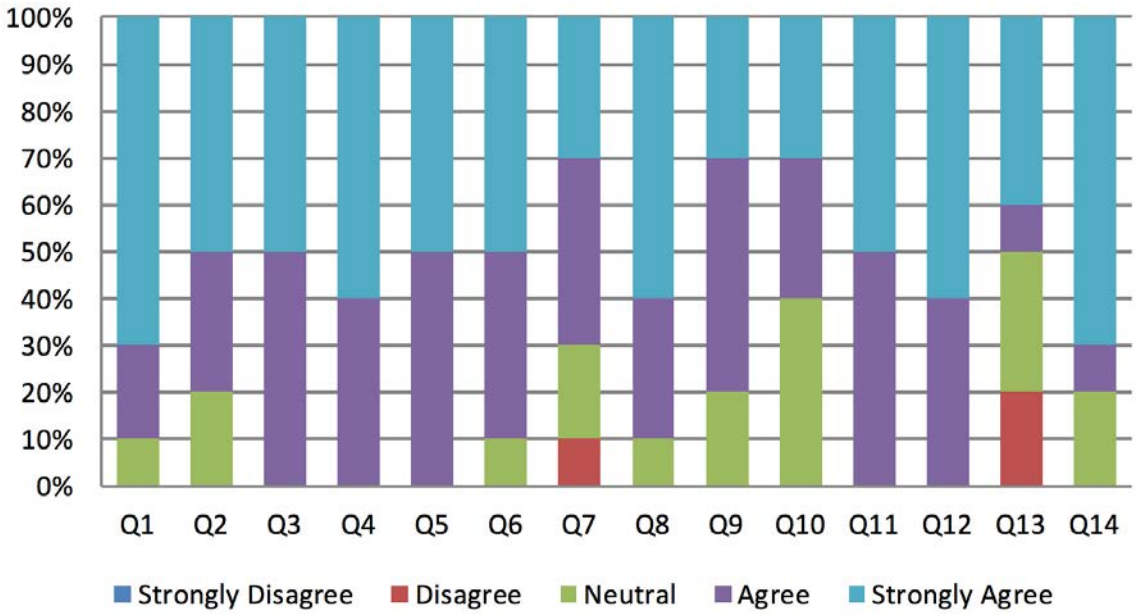



Figura 20. Frecuencias de las respuestas de la encuesta por pregunta.

Por otro lado, sumando los valores de las respuestas de cada usuario y calculando la media de la puntuación otorgada por cada usuario, se obtiene una puntuación media de 60,6, que dividiéndola entre el número de preguntas da una puntuación de 4,33 por cada pregunta.

## 5.3. Discusión

Tras obtener los resultados de cada fase, se procede a analizarlos y realizar una discusión sobre ellos.

### 5.3.1. Fase 1

De la Figura 17, donde se muestra el tiempo empleado por cada participante en cada editor para realizar la tarea asignada, se puede concluir que por norma general, el uso del editor gráfico conlleva un menor tiempo que el editor textual. Sin embargo, hay casos en los que no es así, pero esto puede ser justificable en el editor que han usado en primer lugar ya que el participante número 6 comenzó primero por el editor gráfico que es el que le ha llevado más tiempo y emplearía tiempo en entender correctamente el funcionamiento del sistema.

A continuación, se realizará un estudio estadístico para determinar si el editor que se ha usado en primer lugar ha influido en los resultados obtenidos con el otro editor.

Lo primer que se debe hacer es comprobar si los datos siguen o no una distribución normal. El conjunto de datos usado está dividido en 4 grupos:

- Tiempo empleado en el editor textual por los usuarios que empezaron con el editor textual.
- Tiempo empleado en el editor gráfico por los usuarios que empezaron con el editor textual.
- Tiempo empleado en el editor textual por los usuarios que empezaron con el editor gráfico.
- Tiempo empleado en el editor gráfico por los usuarios que empezaron con el editor gráfico.

Planteando como hipótesis nula que la muestra proviene de una población distribuida normalmente, se aplica el test de normalidad de Shapiro-Wilk obteniendo unos *p-values* de 0,9137, 0,7754, 0,9378 y 0,5215 respectivamente. Todos los *p-values* son mayores que el nivel de significación (0,05), por tanto, no se puede rechazar la hipótesis nula para ninguno de los conjuntos de datos.

Asumiendo que los datos siguen una distribución normal, se comprueba que los conjuntos de datos, en función del editor que se usa en primer lugar, tienen varianzas iguales. Para ello se aplica el test F de Fisher tomando como hipótesis nula la igualdad de varianzas. Al aplicar este test al grupo formado por los usuarios que empiezan por el editor textual cogiendo como variables los tiempos empleados en cada editor, y al grupo formado por los usuarios que empiezan por el editor gráfico se obtienen los *p-values* 0,4142 y 0,4729 respectivamente. Estos *p-values* son mayores que el nivel de significación (0,05), por tanto, no se puede rechazar la hipótesis nula en ninguno de los grupos. Es decir, se asume que las varianzas son iguales en ambos grupos.

Por último, teniendo en cuenta que se asume que las muestras siguen una distribución normal y las varianzas de ambos grupos son iguales, se aplica el test T de Student para muestras pareadas partiendo de la hipótesis nula planteada, es decir, que el editor que se usa en primer lugar no influye en los resultados finales. Los *p-values* obtenidos son los siguientes:

- Comenzando la prueba con el editor textual: 0,001319.
- Comenzando la prueba con el editor gráfico: 0,698.

En el caso de comenzar la prueba con el editor textual, se ha obtenido un *p-value* inferior al nivel de significación (0,05), por tanto, se puede rechazar la hipótesis nula y afirmar que usar el editor textual en primer lugar influye significativamente en el tiempo empleado en el editor gráfico.

Sin embargo, el *p-value* obtenido en el caso de comenzar la prueba con el editor gráfico es mayor que el nivel de confianza (0,05), por tanto, no se puede rechazar la hipótesis nula y se asume que usar el editor gráfico en primer lugar no influye en el tiempo empleado en el editor textual.

En base a estos datos y a la gráfica de la Figura 18, se concluyen que empezar por el editor textual hace que el uso del editor gráfico sea más sencillo y sin embargo, esto no ocurre en el caso contrario. Esto puede ser debido a que usar una sintaxis textual es más difícil que usar componentes visuales, por tanto, una vez que un participante ha completado la tarea con el editor textual, realizarla con el editor gráfico le parece más sencillo. Mientras que, en el caso contrario, para un usuario que se ha enfrentado primero al editor gráfico, enfrentarse al editor textual es posible que le parezca más tedioso y no tenga asimilado por completo el funcionamiento del sistema.

### 5.3.2. Fase 2

A partir de la Tabla 2 y de la Figura 19 se pueden realizar las siguientes interpretaciones:

- Q7 y Q13 tienen el menor mínimo. Algunos participantes están en desacuerdo con ellas. Estas preguntas tratan la dificultad de cometer errores con ambos editores y la necesidad de tener conocimientos de programación para usar el editor textual. Ambas están relacionadas ya que la necesidad de conocimientos que requiere el editor textual implica que se puedan cometer errores con él.
- El mínimo más alto lo tienen Q3, Q4, Q5, Q11 y Q12. Esto significa que todos los participantes estaban, al menos, de acuerdo con ellas.
- Excepto Q7, Q10 y Q13, el resto de declaraciones tiene el primer cuartil por encima de 4, es decir, al menos el 75% de los participantes están de acuerdo con ellas. Esas 3 declaraciones tienen el primer cuartil por encima de 3, es decir, que menos del 25% de los participantes están en desacuerdo con ellas.

- Q1, Q4, Q8, Q12 y Q14 tienen la mediana en el mismo valor que el tercer cuartil y en el máximo valor. Esto quiere decir que, al menos la mitad de los participantes están completamente de acuerdo con esas declaraciones. Q2, Q3, Q5, Q6 y Q11 tienen la mediana por encima de 4 y Q7, Q9, y Q10 tienen la mediana en el valor 4. Esto quiere decir que al menos el 50% por cierto de los participantes está de acuerdo con esas declaraciones. Sin embargo, Q13 tiene la mediana por debajo de 4 pero por encima de 3. Esto quiere decir que menos del 50% de los participantes está de acuerdo con la declaración.
- Excepto Q7, Q9 y Q10, el resto de declaraciones tienen el tercer cuartil en el máximo. Esto quiere decir que al menos el 25% de los participantes está completamente de acuerdo con estas declaraciones.
- Todas las declaraciones tienen como máximo misma opción, completamente de acuerdo. Esto quiere decir que, al menos 1 participante por cada declaración está completamente de acuerdo con ella.
- La moda de Q7 y Q9 ha sido de acuerdo, la de Q10 la opción intermedia y en el resto completamente de acuerdo. Por tanto, la mayoría de los participantes están, al menos, de acuerdo con las declaraciones planteadas.

Por otra parte, a partir de la Figura 20 se pueden obtener interpretaciones todavía no formuladas:

- Q7 y Q13 son las únicas declaraciones en la que algún participante está en desacuerdo con solo un 10% de participantes en desacuerdo en el caso de Q7 y un 20% en el caso de Q13.
- Q13 es la declaración peor valorada ya que solo el 50% de los participantes está de acuerdo con ella.
- Q1 y Q14 son las declaraciones con las que más participantes están completamente de acuerdo, pero las mejor valoradas son Q3, Q4, Q5, Q11 y Q12 ya que el 100% de los participantes esta, al menos, de acuerdo con ellas.

La puntuación media de cada pregunta es 4,33 y se ha calculado en función de la media de las puntuaciones totales de la encuesta por cada usuario, y dividiendo esa media por el número de preguntas. Esa puntuación media representa que los participantes estaban, al menos, de acuerdo con la totalidad de la encuesta, por tanto, podemos concluir que los usuarios han valorado positivamente la propuesta planteada y su utilidad para facilitar la creación de aplicaciones que conecten objetos inteligentes de Internet de las Cosas entre sí y con personas, a través de redes sociales.

# Capítulo 6. Conclusiones y Trabajo Futuro

En este capítulo se presentarán las conclusiones obtenidas tras la realización de la investigación de este trabajo fin de máster y se presentará el trabajo futuro que se puede hacer a partir de esta investigación y del sistema desarrollado.

## 6.1. Conclusiones

En este proyecto fin de máster se ha propuesto una nueva forma de comunicar objetos heterogéneos, ubicuos e inteligentes entre sí y con las personas. Esta comunicación se basa en la utilización de redes sociales.

Este trabajo se apoya en un estudio teórico de diversos conceptos relacionados con la propuesta y un análisis del trabajo relacionado. Entre los conceptos estudiados se encuentran Internet de las Cosas, Objetos Inteligentes, Lenguajes de Dominio Específico, Ingeniería Dirigida por Modelos y Redes sociales. Además, también se han estudiado otros conceptos que han servido para realizar otras investigaciones paralelas, entre los que se encuentran Lógica difusa; Smart Cities, Smart Towns y Smart Homes; y Visión por computador. Estos otros conceptos han servido de base para el desarrollo de otros prototipos y la realización de publicaciones.

Para ello se ha diseñado, mediante la aplicación de la Ingeniería Dirigida por Modelos, un Lenguaje de Dominio Específico, al que se ha llamado Bilrost-specific Language (BSL), y cuyo fin es la definición de objetos inteligentes capaces de conectar a redes sociales para compartir el estado de sus sensores y para permitir que otros objetos o personas puedan controlar sus actuadores.

También se ha desarrollado un prototipo, Bilrost, que permite la generación de proyectos de aplicaciones en diferentes lenguajes donde la conexión con las redes sociales ya está implementada y donde se suministra un esqueleto que el usuario deberá rellenar con la lógica específica para acceder a los datos de los señores y para controlar los actuadores del dispositivo donde se va a desplegar. Este prototipo, además, cuenta con dos editores para definir objetos usando el DSL definido. Usando esos editores, los usuarios podrán crear modelos gráficos o textuales de los dispositivos indicando sus propiedades, sus componentes (actuadores y sensores), los parámetros que las APIs de las redes sociales necesitan y una serie de reglas que automatizan los procesos de invocación de acciones de actuadores del dispositivo o de otros dispositivos.

Para evaluar el sistema propuesto, se ha hecho una comparación entre los dos editores creados midiendo el tiempo que los participantes de la evaluación han empleado en completar una tarea específica con ambos editores. De estas mediciones se ha obtenido que la media de tiempo requerida para completar la tarea con el editor textual es mayor que el tiempo requerido para completar la tarea con el editor gráfico.

Además, se ha realizado otra comparación entre ambos editores considerando si comenzar la tarea con un editor u otro afecta a los resultados. De esta comparación se ha obtenido que empezar la tarea con el editor textual afecta al tiempo requerido para completar la tarea por el editor gráfico mientras que empezar la tarea por el editor gráfico no afecta a los tiempos del editor textual. Concretamente, empezar por el editor textual hace que el tiempo invertido en el editor gráfico sea aún menor.

Por otro lado, la evaluación del sistema propuesto también incluye la realización de una encuesta donde los usuarios evalúan el sistema tras haberlo usado. Las preguntas que compusieron la encuesta trataban asuntos sobre la experiencia de usuario mientras usaban el sistema propuesto. La puntuación media obtenida por pregunta fue 4,33 sobre 5, lo que significa que los participantes estaban de acuerdo con las declaraciones presentadas en la encuesta y que pensaban que el sistema era útil para interconectar dispositivos a través de redes sociales.

Teniendo en cuenta los resultados del proceso de evaluación y valorando el trabajo realizado, se puede afirmar que el sistema es útil para conectar objetos entre sí y con personas mediante el uso de redes sociales y que se han cumplido los objetivos planteados en el Capítulo 2.

Se ha logrado integrar objetos inteligentes en una red social, se ha logrado que las personas puedan interactuar con estos objetos, se ha logrado que los objetos se comuniquen entre sí a través de la red social, se ha definido un Lenguaje de Dominio Específico que realiza esa integración y configura las comunicaciones, se ha logrado facilitar la generación de aplicaciones que interconecten objetos inteligentes mediante la aplicación de Ingeniería Dirigida por Modelos y se ha logrado que se puedan generar aplicaciones que soporten distintas plataformas mediante la integración de elementos en el DSL que indican la plataforma de destino.

En resumen, se ha conseguido confirmar la hipótesis planteada, es decir, es posible facilitar la creación de aplicaciones que integren objetos inteligentes de Internet de las Cosas en las redes sociales de personas, permitiendo la interaccionar entre sí, y la interacción de las personas con los objetos inteligentes.

## 6.2. Trabajo Futuro

Internet de las cosas podría ser el futuro, pero para ello es necesario abordar el desarrollo de soluciones que integran IoT en nuestras vidas. La investigación presentada en este proyecto fin de máster todavía no está concluida ya que queda mucho trabajo por hacer y líneas de investigación abiertas que seguir. A continuación, se enumeran sugerencias de trabajo futuro que se pueden realizar a partir de la investigación realizada en este proyecto fin de máster.

- **Añadir la generación de aplicaciones finales con la lógica específica ya implementada**: Las aplicaciones que la propuesta presentada puede generar no son aplicaciones para usuarios finales ya que los usuarios deben completar las aplicaciones con la lógica específica del dispositivo donde se van a desplegar. Mejorar la sintaxis del BSL con el objetivo de añadir componentes para indicar el dispositivo es un posible trabajo futuro. Indicando el dispositivo, el sistema debería ser capaz de implementar el código específico para este. Además, se deberán añadir componentes al lenguaje dependientes de cada dispositivo ya que no todos los dispositivos tienen los mismos componentes o usan la misma configuración.
- **Carga de módulos externos para nuevos dispositivos**: En relación con el trabajo futuro presentado en el punto anterior, otro trabajo futuro podría consistir en permitir la ampliación del BSL y del sistema mediante módulos externos de manera que cualquier usuario pueda desarrollar un módulo para un dispositivo externo e integrarlo en el sistema con el fin de generar proyectos finales sin tener que volver a implementar la lógica específica en cada generación.
- **Realizar un estudio sobre las posibilidades de las redes sociales de personas para conectar objetos inteligentes**: El sistema aquí presentado usa Twitter para interconectar los objetos, pero existen muchas otras redes sociales que podrían ser útiles para conectar objetos. Un trabajo futuro podría ser la realización de una comparación de las redes sociales actuales centrándose en las ventajas y desventajas de integrar objetos inteligentes en ellas.

- **Seguridad y privacidad**: Actualmente, el sistema desarrollado usa la línea temporal de Twitter para compartir los mensajes que se usan para realizar la comunicación. Esta línea temporal suele ser pública. Un trabajo futuro podría ser ampliar la sintaxis del BSL para añadir opciones que permitan usar los mensajes privados de las redes sociales en vez de la zona pública, con el objetivo de comunicar objetos privados que no deben ser controlados por otras personas ni sus datos publicados.
- **Usar lenguaje natural en las comunicaciones**: El sistema propuesto usa las palabras claves disponibles en las redes sociales para reconocer y decidir qué hacer. Sin embargo, esto es demasiado artificial para las personas y hace difícil la interacción entre personas y objetos. Por ello, un posible trabajo futuro sería evitar el uso de estas palabras clave y usar lenguaje natural mediante algoritmos de procesamiento de lenguaje natural. Esto haría que la integración del sistema en las redes sociales fuese más natural y cómodo para los usuarios.

# Capítulo 7. Gestión del Proyecto

Durante el transcurso del proyecto principal de este trabajo fin de máster, es decir, Bilrost, se ha llevado una gestión del proyecto basándose en los conocimientos de buenas prácticas para la dirección, gestión y administración de proyectos disponibles en el PMBOK (*Project Management Body of Knowledge*) (PMI, 2013).

El PMBOK reúne una serie de conocimientos y de «buenas prácticas» aplicables a la gestión de proyectos y formadas por varios procesos y áreas de conocimiento. Su contenido se desarrolló a partir de las prácticas obtenidas de profesionales del campo de la gestión de proyectos y describe normas establecidas, métodos a seguir, procesos y prácticas de la gestión de proyectos. Estás prácticas han ido ampliándose y mejorando durante los últimos veinte años con la colaboración de profesionales y académicos de distintos ámbitos, aunque especialmente de la ingeniería.

No se puede interpretar al PMBOK como una metodología ni una descripción de cómo llevar a cabo los proyectos sino como una guía a seguir o un marco de trabajo que suma el conocimiento de los profesionales de la gestión de proyectos en el que apoyarse proporcionando y promoviendo un vocabulario común y estandarizado dentro del ámbito de la gestión de proyectos. La existencia de un vocabulario estándar es un elemento esencial de cualquier profesión. Esta guía es aplicable durante la mayor parte de la vida de casi todos los proyectos, aunque debe adaptarse al dominio técnico de cada caso y contexto particular. Además, existe un amplio consenso sobre la utilidad de las «buenas prácticas» reflejadas en ella. Su importancia se basa en que provee una guía o marco de trabajo que orienta sobre la forma de avanzar en los procesos necesarios para alcanzar objetos y obtener resultados.

A través del PMBOK se tiene acceso a la información necesaria para iniciar, planificar, ejecutar, supervisar y controlar, y cerrar proyectos a través de las «buenas prácticas» que identifican procesos a seguir durante la dirección de proyectos. Estos procesos se aplican a nivel global y en todos los grupos de negocios o industriales. Se ha comprobado que estas «buenas prácticas» aumentan las posibilidades de éxito en una gran variedad de proyectos.

El PMBOK contiene en 13 capítulos donde los capítulos 1 y 2 contienen definiciones y conceptos acerca de la gestión de proyectos, el valor de negoció, los roles, etc. El capítulo 3 describe el estándar y los capítulos del 4 al 13 describen las 10 áreas de conocimiento y los 47 procesos agrupados en 5 grupos de procesos.

Las 10 áreas de conocimiento del PMBOK son las siguientes:

- Gestión de la integración del proyecto Gestión del alcance.
- Gestión del tiempo.
- Gestión de los costes.
- Gestión de la calidad.
- Gestión de los recursos humanos Gestión de las comunicaciones Gestión de los riesgos.
- Gestión de las adquisiciones.
- Gestión de los interesados.

Los 5 grupos de procesos del PMBOK son los siguientes:

- Procesos de inicio
- Procesos de planificación
- Procesos de ejecución
- Procesos de monitoreo y control
- Procesos de cierre

En este capítulo se presentará la gestión llevada a cabo del proyecto principal de este trabajo fin de máster identificando los procesos del PMBOK aplicados, justificando como se han llevado a cabo y presentando la evolución del proyecto desde su inicio hasta su fin mediante la planificación elaborada y las decisiones tomadas para su desviación, además del presupuesto del mismo.

## 7.1. Identificación de los procesos del PMBOK aplicados

En este apartado se identificarán los procesos del PMBOOK aplicados mediante la presentación de una adaptación de los procesos para el proyecto agrupándolos para facilitar su realización y simplificar la documentación a generar. La agrupación realizada se basa en las áreas de conocimiento de los procesos del PMBOOK.

### 7.1.1. Gestión de vida del proyecto

Este proceso adaptado engloba los procesos del PMBOOK numerados como 4.1, 4.2, 4.3, 4.4, 4.5 y 4.6; y que se pueden encontrar en (PMI, 2013). A continuación, se presenta las entradas de este proceso adaptado, como se realiza y las salidas que se obtienen.

#### *7.1.1.1. Entradas*

- Descripción del proyecto.
- Acuerdos tomados con el director.
- Pronóstico de duración del proyecto.

#### *7.1.1.2. Realización*

Para la realización del **acta de constitución** se analiza todas las entradas para obtener la justificación del proyecto, la descripción del proyecto más detallada, objetivos, requisitos de alto nivel, riesgos de alto nivel, hitos, primera aproximación al presupuesto, criterios de aceptación y requisitos no funcionales relacionados con acuerdos llegados con el director como una primera aproximación de los puntos de verificación de calidad.

La gestión de cambios no se necesita plasmar en una salida puesto que solo hay una persona involucrada en los cambios y por tanto no necesitan ser tratados por otros procesos.

#### *7.1.1.3. Salidas*

- Acta de constitución.

## 7.1.2. Gestión del tiempo del proyecto

Este proceso adaptado engloba los procesos del PMBOOK numerados como 5.4, 6.1, 6.2, 6.3, 6.4, 6.5, 6.6 y 6.6; y que se pueden encontrar en (PMI, 2013). A continuación, se presenta las entradas de este proceso adaptado, como se realiza y las salidas que se obtienen.

### *7.1.2.1. Entradas*

- Requisitos acordados con el director del proyecto.
- Acta de constitución del proyecto.

### *7.1.2.2. Realización*

A partir del documento del alcance, los requisitos y toda la información del acta de constitución se crea el cronograma del proyecto. Para ello primero se listan todas las tareas del proyecto y se crea el EDT para tener una división por módulos del proyecto. A partir del EDT se crea el cronograma junto a la duración de cada tarea y los recursos asociados a ellas incluyendo su coste.

El cronograma podrá sufrir modificaciones durante el proyecto y su revisión será diaria para conocer las tareas que se tiene que realizar o si es necesario reajustar.

### *7.1.2.3. Salidas*

- Cronograma del proyecto con las tareas, duraciones e hitos.

## 7.1.3. Gestión de los riesgos del proyecto

Este proceso adaptado engloba los procesos del PMBOOK numerados como 11.1, 11.2, 11.3, 11.4, 11.5 y 11.6; y que se pueden encontrar en (PMI, 2013). A continuación, se presenta las entradas de este proceso adaptado, como se realiza y las salidas que se obtienen.

### *7.1.3.1. Entradas*

- Alcance del proyecto del acta de constitución y requisitos acordados con el director del proyecto.
- Cronograma del proyecto con las tareas, duraciones e hitos.

### *7.1.3.2. Realización*

A partir del alcance, de los requisitos y del se realiza la lista de riesgos en la que se incluyen los riesgos junto al impacto y la probabilidad de que ocurran.

Los riesgos no son estáticos, sino que evolucionan a lo largo del proyecto ya que pueden aparecer nuevos riesgos, pueden variar los ya existentes y puede que riesgos no monitorizados tengan que ser monitorizados por aumentar su probabilidad. Por tanto, los riesgos se deben revisar con frecuencia, es decir, después de cada hito.

### *7.1.3.3. Salidas*

- Registro de riesgos.

### 7.1.4. Gestión de costes del proyecto

Este proceso adaptado engloba los procesos del PMBOOK numerados como 7.2 y 7.3; y que se pueden encontrar en (PMI, 2013). A continuación, se presenta las entradas de este proceso adaptado, como se realiza y las salidas que se obtienen.

#### *7.1.4.1. Entradas*

- Acta de constitución.
- Cronograma del proyecto con las tareas, duraciones e hitos.
- Registro de riesgos y plan de gestión de riesgos.

#### *7.1.4.2. Realización*

A partir de la información del acta de constitución, de los riesgos y del cronograma del proyecto se estiman los costes del proyecto y de estos se realiza el presupuesto añadiendo el beneficio y la reserva para cubrir los riesgos que puedan ocurrir y que no se hayan identificado.

#### *7.1.4.3. Salidas*

- Presupuesto presentado en una hoja de cálculo.

## 7.2. Aplicación de los procesos del PMBOOK

En este apartado se explicará cómo se han aplicado los procesos adaptados del PMBOK identificados en el apartado anterior. Para ello se volverán a mencionar las adaptaciones de los procesos y se explicará el trabajo realizado sobre cada proceso.

### 7.2.1. Gestión de vida del proyecto

Este proceso adaptado tiene como finalidad acotar la vida del proyecto partiendo de la descripción del proyecto, los acuerdos tomados con el director del proyecto y la estimación de duración del proyecto. Para ello se ha realizado el acta de constitución del proyecto donde se encuentra la justificación del proyecto, la descripción del proyecto más detallada, los primeros requisitos del sistema, criterios de aceptación, riesgos iniciales, objetivos del proyecto, hitos y una primera versión del presupuesto entre otra información importante para el comienzo del proyecto.

El acta de constitución se realizó a partir de las conclusiones obtenidas en la primera reunión con el director y se puede encontrar en el Anexo I.

### 7.2.2. Gestión del tiempo del proyecto

Este proceso adaptado tiene como finalidad controlar la duración del proyecto para finalizarlo dentro de los plazos previstos. Para ello, a partir de la información del acta de constitución y de una reunión con el director para obtener todos los requisitos, se ha elaborado una planificación inicial que se ha intentado cumplir en el transcurso del proyecto. Sin embargo, debido a diversos motivos no se ha podido seguir completamente la planificación propuesta.

En este apartado se presenta la planificación inicial, los cambios que han alterado esta planificación y se mostrarán las actas de reuniones realizadas con el director y que resumen las decisiones tomadas.

### *7.2.2.1. Planificación inicial*

A continuación, se muestra la planificación inicial creada usando Microsoft Project. Debido a su extensión se ha dividido en 4 partes.

| Id | Nombre de tarea | Duración |
|---|---|---|
| 1 | **Proyecto** | **92 días** |
| 2 | Reunión con los directores | 2 horas |
| 3 | **Definición del proyecto** | **0,63 días** |
| 4 | Resumen del proyecto | 1 hora |
| 5 | Identificación de objetivos | 0,5 horas |
| 6 | Determinación del alcance | 0,5 horas |
| 7 | Requisitos preliminares | 1 hora |
| 8 | Planificar gestión de riesgos | 2 horas |
| 9 | Realización de planificación | 3 horas |
| 10 | Realización de presupuesto | 1 hora |
| 11 | Reunión de aprobación | 2 horas |
| 12 | **Investigación previa** | **5,75 días** |
| 18 | Reunión con los directores | 2 horas |
| 19 | Proyecto aprobado por los directores | 0 días |
| 20 | **Gestión del proyecto** | **84,38 días** |
| 21 | Gestión de riesgos | 82 días |
| 22 | Gestión de tiempo | 82 días |
| 23 | **Analisis y diseño** | **2,25 días** |
| 24 | Especificación de requisitos | 4 horas |
| 25 | Diseño de casos de uso | 4 horas |
| 26 | Diseño de la arquitectura del sistema | 4 horas |
| 27 | Diseño de interfaces de usuario | 4 horas |
| 28 | Reunión de validación | 2 horas |
| 29 | Finalización del análisis y diseño | 0 días |
| 30 | **Inicio del desarrollo** | **8 días** |
| 31 | Instalación y configuración de herramientas | 2 días |
| 32 | Instalación y configuración del servidor | 2 días |
| 33 | Adquisición de sensores y actuadores | 2 días |
| 34 | Adquisición de Raspberrys Pi y Arduinos | 2 días |
| 35 | Instalación y configuración de Raspberry Pi y Arduino | 1 día |
| 36 | Creación de pilotos | 3 días |
| 37 | **Desarrollo del sistema** | **63,63 días** |
| 38 | **Redes sociales** | **2,88 días** |
| 39 | Selección de redes sociales | 3 horas |
| 40 | Implementación de conexión a las redes sociales | 2 días |

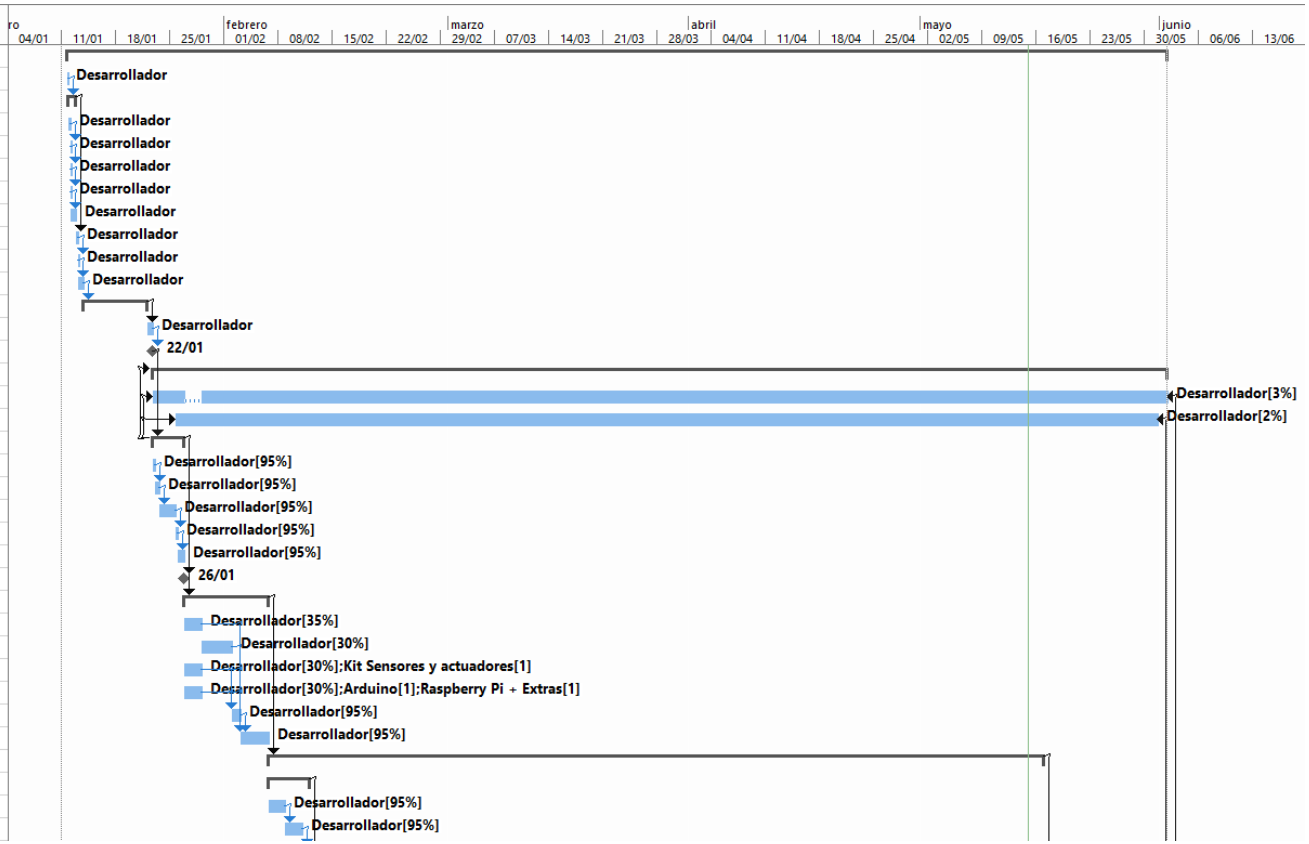



Figura 21. Planificación inicial - Parte I.

| Id | Nombre de tarea | Duración |
|---|---|---|
| 41 | Pruebas de implementación | 4 horas |
| 42 | **DSL Textual** | **24,38 días** |
| 43 | Definición del DSL textual | 2 horas |
| 44 | Reunión de validación del DSL textual | 1 hora |
| 45 | DSL textual definido y validado | 0 días |
| 46 | **Módulo actuadores** | **9,75 días** |
| 47 | Implementación del parseador del DSL | 5 días |
| 48 | Implementación del interprete | 4 días |
| 49 | Pruebas de implementación | 4 horas |
| 50 | Reunión de validación del módulo | 2 horas |
| 51 | Módulo actuadores DSL textual finalizado | 0 días |
| 52 | **Módulo sensores** | **6,75 días** |
| 53 | Implementación del parseador del DSL | 3 días |
| 54 | Implementación del interprete | 3 días |
| 55 | Pruebas de implementación | 4 horas |
| 56 | Reunión de validación del módulo | 2 horas |
| 57 | Módulo sensores DSL textual finalizado | 0 días |
| 58 | **Módulo reglas combinatorias** | **7,25 días** |
| 59 | **Implementación del parseador del DSL** | **2,25 días** |
| 60 | Combinación de parseadores de módulos anteriores | 2 horas |
| 61 | Actualización del parseador del DSL | 2 días |
| 62 | **Implementación del interprete** | **4,25 días** |
| 63 | Combinación de interpretes de módulos anteriores | 2 horas |
| 64 | Actualización del interprete | 4 días |
| 65 | Pruebas de implementación | 4 horas |
| 66 | Reunión de validación del módulo | 2 horas |
| 67 | Módulo reglas combinatorias DSL textual finalizado | 0 días |
| 68 | Pruebas de integración | 2 horas |
| 69 | DSL textual finalizado | 0 días |
| 70 | **Árticulo sobre DSL textual** | **7,88 días** |
| 76 | Artículo DSL textual enviado | 0 días |
| 77 | **DSL gráfico** | **20,25 días** |
| 78 | Definición del DSL gráfico | 2 horas |
| 79 | Reunión de validación del DSL gráfico | 1 hora |
| 80 | DSL gráfico definido y validado | 0 días |

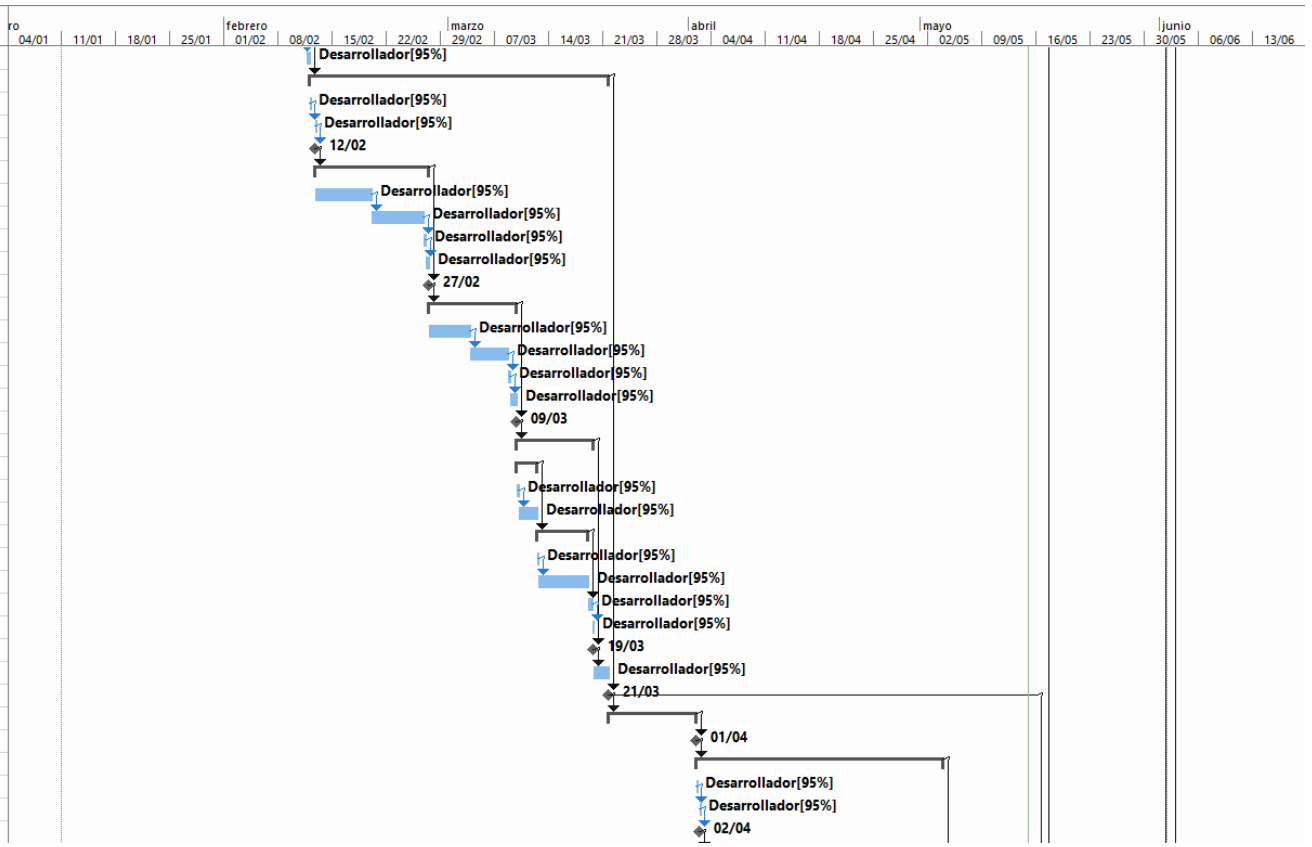
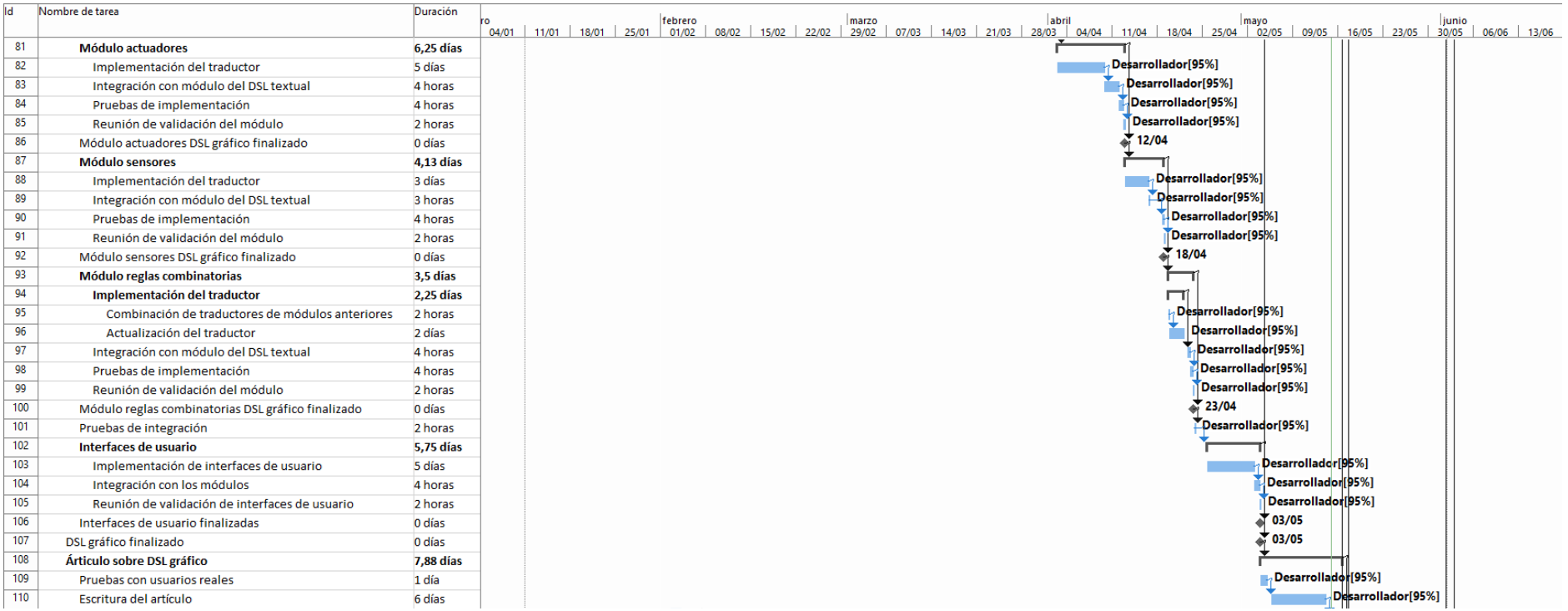



Figura 22. Planificación inicial - Parte II.



| Id | Nombre de tarea | Duración |
|---|---|---|
| 81 | **Módulo actuadores** | **6,25 días** |
| 82 | Implementación del traductor | 5 días |
| 83 | Integración con módulo del DSL textual | 4 horas |
| 84 | Pruebas de implementación | 4 horas |
| 85 | Reunión de validación del módulo | 2 horas |
| 86 | Módulo actuadores DSL gráfico finalizado | 0 días |
| 87 | **Módulo sensores** | **4,13 días** |
| 88 | Implementación del traductor | 3 días |
| 89 | Integración con módulo del DSL textual | 3 horas |
| 90 | Pruebas de implementación | 4 horas |
| 91 | Reunión de validación del módulo | 2 horas |
| 92 | Módulo sensores DSL gráfico finalizado | 0 días |
| 93 | **Módulo reglas combinatorias** | **3,5 días** |
| 94 | **Implementación del traductor** | **2,25 días** |
| 95 | Combinación de traductores de módulos anteriores | 2 horas |
| 96 | Actualización del traductor | 2 días |
| 97 | Integración con módulo del DSL textual | 4 horas |
| 98 | Pruebas de implementación | 4 horas |
| 99 | Reunión de validación del módulo | 2 horas |
| 100 | Módulo reglas combinatorias DSL gráfico finalizado | 0 días |
| 101 | Pruebas de integración | 2 horas |
| 102 | **Interfaces de usuario** | **5,75 días** |
| 103 | Implementación de interfaces de usuario | 5 días |
| 104 | Integración con los módulos | 4 horas |
| 105 | Reunión de validación de interfaces de usuario | 2 horas |
| 106 | Interfaces de usuario finalizadas | 0 días |
| 107 | DSL gráfico finalizado | 0 días |
| 108 | **Árticulo sobre DSL gráfico** | **7,88 días** |
| 109 | Pruebas con usuarios reales | 1 día |
| 110 | Escritura del artículo | 6 días |



Figura 23. Planificación inicial - Parte III.

| Id | Nombre de tarea | Duración |
|---|---|---|
| 111 | Reunión de revisión | 3 horas |
| 112 | Selección de revista | 3 horas |
| 113 | Envío del artículo | 1 hora |
| 114 | Artículo DSL gráfico enviado | 0 días |
| 115 | Despliegue en el hardware | 3 horas |
| 116 | Diseño y creación de ejemplos de uso | 3 días |
| 117 | Escritura de la memoria final | 5 días |
| 118 | Realización de la presentación | 1 día |
| 119 | Reunión de revisión | 3 horas |
| 120 | Presentación del proyecto | 1 hora |
| 121 | Proyecto presentado | 0 días |
| 122 | **Cierre de proyecto** | **0,38 días** |
| 123 | Reunión de valoración final | 1 hora |
| 124 | Archivado de entregables | 2 horas |

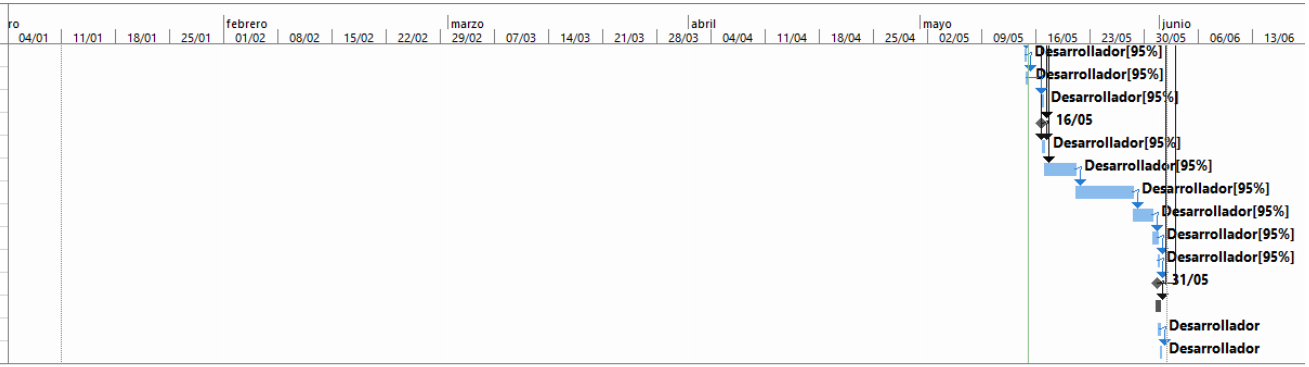



Figura 24. Planificación inicial - Parte IV.

Una vez hecha la planificación en Microsoft Project, se obtiene que la duración aproximada del proyecto es de 92 días, 552 horas, donde el proyecto comienza el lunes, 11 de enero del 2016 y finaliza el miércoles, 1 de junio del 2016.

El calendario que se ha utilizado está formado por 6 días laborables a la semana, es decir, se incluyen las mañanas de los sábados, y 6 horas de trabajo diarias por semana y 4 horas los sábados. Esto es una aproximación del trabajo estimado ya que al tratarse de un proyecto académico no existe un horario fijo para su desarrollo.

### *7.2.2.2. Valoración tras finalizar*

La planificación inicial se ha intentado seguir en el máximo posible, aunque se han experimentado desviaciones que han alterado significativamente el rumbo seguido. Todas las decisiones tomadas y que han hecho variar el rumbo se justifican con las reuniones realizadas como se verá a continuación.

Según la planificación inicial se debería haber realizado 16 reuniones, pero debido a diversos factores se han reducido. A continuación, se muestran las reuniones realizadas junto a una descripción sobre lo que se ha tratado y como ha afectado a la planificación si ha sido el caso.

1. **Reunión de aprobación de la idea propuesta para trabajo de fin de máster** – 12/01/2016

En esta reunión simplemente se ha presentado la idea siendo esta aceptada por el director y acordando una posterior reunión dentro de lo planificado para valorar los trabajos relacionados en la literatura científica.

2. **Reunión de aprobación del proyecto una vez revisada la literatura existente** – 21/01/2016

En esta reunión, el alumno ha presentado al director la información encontrada sobre los trabajos relacionados y la investigación realizada sobre tecnologías a usar y como realizar el desarrollo. El director ha aprobado las propuestas del alumno y se ha fijado una reunión para finalizar las tareas de análisis y diseño.

3. **Reunión de validación del análisis y diseño del sistema** – 26/02/2016

En esta reunión se fijan finalmente todos los requisitos del sistema, sus casos de uso, la arquitectura y la posible interfaz de usuario. Se acuerda una próxima reunión siguiendo lo planificado para validar la propuesta de DSL textual.

4. **Validación del DSL textual** – 12/02/2016

En esta reunión el alumno presenta el DSL textual y se proponen los cambios necesarios para que el director lo valide. Se acuerda centrarse solo en la primera parte del DSL textual (actuadores) por estar más clara y se retrasa la validación del resto del DSL a posteriores reuniones. Se fija la próxima reunión según la planificación para validar el módulo de actuadores.

5. **Validación del módulo de actuadores usando DSL textual** – 26/02/2016

En esta reunión se presenta al director el módulo de manejo de actuadores a través de redes sociales usando un DSL textual. A partir de este momento la planificación se altera ya que surge la posibilidad de escribir un artículo para un *Special Issue* sobre Internet de las Cosas sobre el trabajo realizado hasta este momento, es decir, el control de actuadores a través de redes sociales. Por ello se altera la planificación y se acuerda otra reunión para revisar el estado del artículo.

6. **Revisión del artículo sobre el módulo de actuadores usando DSL textual** – 03/03/2016

En esta reunión se revisa el estado del artículo, se proponen correcciones, se establece la metodología de evaluación y se acuerda comunicación vía email para el resto de revisiones antes de enviar el artículo al *Special Issue*. Se acuerda además que la próxima reunión será cuando todo el DSL textual esté finalizado y no por cada módulo ya que el módulo de sensores no aporta a trabajos ya existentes en la literatura y que la fecha será establecida por email en el momento de finalizar el desarrollo.

7. **Validación del conjunto de módulos usando DSL textual** – 04/04/2016

Por diversas causas ajenas al proyecto se retrasa su desarrollo finalizando con varias semanas de retraso. En la reunión se acuerda no realizar un artículo sobre el DSL textual para no arriesgar los plazos y se acuerda una próxima reunión cuando el desarrollo de todos los módulos del DSL gráfico ya esté finalizados. Además, se debate cómo será el DSL gráfico y la interfaz del editor.

8. **Validación del sistema bajo ambos DSLs**. – 03/05/2016

Tras un mes de desarrollo y sin reuniones de por medio, se hace una reunión en la que se prueba el sistema completo a falta de desarrollar la generación de proyectos en diversos lenguajes y el inicio de sesión con Twitter, tareas que se propone que se hagan en descansos de la documentación y del artículo del sistema completo. El director indica que se debe ir haciendo tanto la memoria como el artículo si se quiere cumplir los plazos. Además, una vez validado el sistema, comienzan las pruebas con el usuario por lo que se acuerda con el director como se deben realizar las pruebas con usuarios. Por último, se discute a donde reenviar el artículo enviado al *Special Issue* una vez corregido con las sugerencias del rechazo recibido.

9. **Revisión de documentación y artículo** – 18/05/2016

En esta reunión se hace una revisión de la documentación y del artículo acordando los puntos a mejorar o completar antes de realizar la solicitud de defensa, cuya fecha límite es el 25 de mayo. Se acuerda otra reunión para finalizar el artículo y la parte de la documentación necesaria para solicitar la defensa unos días antes de la fecha límite.

10. **Revisión final del artículo** – 23/05/2016

El director comprueba todo el artículo y propone los últimos cambios antes de enviarlo a una revista. También se discute a que revista enviar este artículo. Por último, se revisan los cambios en la documentación y se proponen nuevos cambios.

11. **Cierre del proyecto** – 30/05/2016

Última reunión antes del final del plazo de entrega. En ella se acuerdan los últimos cambios en la documentación y se prepara la entrega del trabajo fin de máster.

### 7.2.3. Gestión de los riesgos del proyecto

Este proceso adaptado tiene como finalizar identificar y planificar los riesgos del proyecto. Para ello, a partir de la información del acta de constitución, de una reunión con el director para definir todos los requisitos y de la planificación inicial con tareas, duraciones e hitos, se ha realizado una lista de los riesgos que podrían afectar a este proyecto junto al impacto y la probabilidad de que estos ocurran.

A continuación, se presentan el registro de riesgos identificados con el impacto y la probabilidad de que ocurran.

### *7.2.3.1. Registro de riesgos*

| ID | Nombre | Categoría | Probabilidad | Impacto | | | | Score | 0,40 Priorización | Respuesta |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Presupuesto | Planificación | Alcance | Calidad | | | |
| 1 | Planificación optimista | Riesgo de la gestión del proyecto (Estimación) | Media | Moderado | Alto | Moderado | Moderado | 0,20 | | Diseñar una estrategia proactiva de evitación. Disponer de una reserva de jornadas de trabajo porcentual al número de jornadas totales planificadas de manera que se puedan usar en caso de necesitar ampliar alguna tarea. |
| 2 | Planificación pesimista | Riesgo de la gestión del proyecto (Estimación) | Muy Baja | Bajo | Moderado | Muy Bajo | Muy Bajo | 0,02 | | Diseñar una estrategia proactiva de mejora. Tener en cuenta esta posibilidad para aumentar la reserva de jornadas para el riesgo anterior. |
| 3 | Herramientas muy complejas | Riesgo técnico (Complejidad e interfaces, y Tecnología) | Baja | Moderado | Alto | Bajo | Alto | 0,12 | | Usar una estrategia reactiva de retención. Se asume el riesgo y se revisan otras herramientas que puedan servir de alternativa. |

| ID | Nombre | Categoría | Probabilidad | Impacto | | | | Score | 0,40 Priorización | Respuesta |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Presupuesto | Planificación | Alcance | Calidad | | | |
| 4 | El desarrollador ya tiene conocimientos sobre las tecnologías | Riesgo técnico (Complejidad e interfaces, y Tecnología) | Alta | Bajo | Alto | Alto | Alto | 0,28 | | Diseñar una estrategia proactiva de explotación. Intentar explotar este riesgo seleccionado herramientas que ya conozca el desarrollador para aprovechar los conocimientos para mejorar la calidad del producto o para aportar valor extra. Definir características complementarias que se puedan hacer si aparece este riesgo. |
| 5 | El desarrollador no dispone de tiempo para el proyecto | Riesgo organizacional (Recursos) | Baja | Moderado | Alto | Moderado | Moderado | 0,12 | | Usar una estrategia reactiva de retención. No es posible saber que una persona no va poder avanzar en el proyecto. Si surge se deberá intentar reorganizar el tiempo para reducir el impacto. |
| 6 | Las APIs externas están limitadas | Riesgo técnico (Requisitos, tecnología)<br>Riesgo organizacional (Dependencias del proyecto) | Baja | Moderado | Moderado | Alto | Moderado | 0,12 | | Diseñar una estrategia proactiva de evitación. Consultar APIs que puedan ser usadas como alternativas. Si el problema está en la API de Twitter recurrir a otra red social como Facebook. |

| ID | Nombre | Categoría | Probabilidad | Impacto | | | | Score | 0,40 Priorización | Respuesta |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Presupuesto | Planificación | Alcance | Calidad | | | |
| 7 | Las APIs externas cambian continuamente | Riesgo técnico (Tecnología)<br>Riesgo organizacional (Dependencias del proyecto) | Muy Baja | Alto | Alto | Alto | Muy Alto | 0,08 | | Usar una estrategia reactiva de retención. Si es necesario se crea un adaptador de la API como otra API propia para no influir en el proyecto. |
| 8 | Ejemplos de las APIs y librerías re-aprovechables | Riesgo técnico (Tecnología) | Baja | Bajo | Alto | Bajo | Bajo | 0,12 | | Diseñar una estrategia proactiva de aceptación. Añadir los ejemplos al código del proyecto. |
| 9 | Implementaciones de módulos incompatibles entre si | Riesgo técnico (Requisitos)<br>Riesgo de la gestión del proyecto (Comunicación y control) | Muy Baja | Moderado | Muy Alto | Muy Alto | Alto | 0,08 | | Diseñar una estrategia proactiva de evitación. Acordar entre todos los desarrolladores el diseño de los módulos para garantizar compatibilidad. |
| 10 | Alta latencia en las comunicaciones | Riesgo técnico (Tecnología, Calidad)<br>Riesgo organizacional (Dependencia del proyecto)<br>Riesgo de la gestión del proyecto (Control) | Alta | Moderado | Alto | Alto | Muy Alto | 0,56 | | Diseñar una estrategia proactiva de evitación. Realizar mediciones durante la implementación del proyecto asegurándose de obtener el máximo rendimiento posible. En caso de que aun así no se mejore la latencia habrá que usar una estrategia de retención y asumir el riesgo. |

| ID | Nombre | Categoría | Probabilidad | Impacto | | | | Score | 0,40 Priorización | Respuesta |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Presupuesto | Planificación | Alcance | Calidad | | | |
| 11 | No se superan las pruebas de usabilidad | Riesgo técnico (Requisitos y Calidad)<br>Riesgo externo (Usuario) | Muy Baja | Bajo | Alto | Moderado | Muy Alto | 0,08 | | Diseñar una estrategia proactiva de evitación. Consultar durante la realización de las interfaces a distintos usuarios para asegurarse que la interfaz sea calificada como usable por potenciales usuarios. |
| 12 | Accidente o enfermedad del desarrollador | Riesgo organizacional (Recursos) | Media | Alto | Moderado | Bajo | Bajo | 0,20 | | Usar una estrategia reactiva de retención. No es posible saber que una persona va a tener un accidente o va a sufrir una enfermedad. Si surge se deberá buscar personal sustituto ya formado para reducir el impacto. |
| 13 | Proyecto final no cumple las expectativas | Riesgo técnico (Requisitos, Calidad)<br>Riesgo externo (Usuario)<br>Riesgo de la gestión del proyecto (Control y Comunicación) | Baja | Muy Alto | Muy Alto | Alto | Muy Alto | 0,24 | | Diseñar una estrategia proactiva de evitación. Revisar los requisitos, el alcance y los objetivos con el cliente y realizar reuniones periódicas para que valore el progreso del proyecto. |

Tabla 3. Registro de riesgos.

## 7.2.4. Gestión de costes del proyecto

Este proceso adaptado tiene como finalidad controlar el coste final del proyecto. Al tratarse de un proyecto perteneciente a un trabajo fin de máster no existen costes reales que afrontar ya que, por ejemplo, los costes materiales pueden anularse debido al uso de material disponible en la Universidad y los costes personales tampoco serían reales ya que la realización de este proyecto no implica la retribución de un salario. Aun así, y para fines académicos, se ha simulado la gestión de costes del proyecto mediante la elaboración de dos presupuestos ficticios, un presupuesto de costes y un presupuesto de cliente. Para la elaboración de estos presupuestos se han tenido en cuenta las salidas de los demás procesos de gestión de proyectos y las variables necesarias para realizar los cálculos correspondientes. Estas variables son las siguientes:

- **Horas de trabajo**: 552 horas (obtenidas de la planificación inicial del proyecto procedente del proceso de gestión del tiempo del proyecto).
- **Jornadas de trabajo mensuales**: 26 jornadas.
- **Horas de trabajo diarias**: 6 horas.
- **Porcentaje del salario para la seguridad social**: 35%.
- **Porcentaje del IVA en el momento de la realización del proyecto**: 21%.
- **Porcentaje de beneficio deseado**: 20%.

### *7.2.4.1. Presupuesto de costes*

En este apartado se desglosará el presupuesto de costes explicando cada uno de los conceptos que aparece en él.

El presupuesto está dividido en 5 categorías, materiales, trabajo personal, equipamiento e instalaciones, software y otros gastos.

#### 7.2.4.1.1. Materiales

Los materiales contemplan el hardware necesitado para este proyecto como parte de sus requisitos, que no es amortizable con otros proyectos posteriores puesto que pasará a posesión del cliente y que por tanto su gasto se computa en este proyecto.

En este proyecto se necesita un servidor donde desplegar las herramientas necesarias para generar aplicaciones, una Raspberry Pi junto a los accesorios necesarios y un Arduino para poder probar los prototipos realizados y un kit de sensores y actuadores compatibles con la Raspberry Pi y el Arduino anterior.

El coste de estos materiales es el siguiente:

- El **servidor** tiene un coste inicial de 3.000 €. El mantenimiento no se contempla en este presupuesto ya que correrá a cargo del cliente. El servidor contará con una distribución de Linux gratuita como Sistema Operativo, Ubuntu Server en su última versión LTS.
- La **Raspberry Pi + Extras** tiene un coste inicial de 70 €.
- El **Arduino** tiene un coste inicial de 30 €.
- El **kit de Sensores y Actuadores** tiene un coste inicial de 200 €. Este kit contendrá diversos sensores de varios tipos y actuadores necesarios para poner a prueba nuestro sistema.

La suma total del coste de estos materiales es 3.300,00 €.

#### 7.2.4.1.2. Trabajo personal

El trabajo personal contempla el trabajo realizado por los encargados de la realización del proyecto, es decir, por el único desarrollador del proyecto.

En este caso consta el salario del programador que es de 15 €/hora, que equivale a 2.340,00 € mensuales trabajando 6 horas al día y 26 días al mes, es decir, lo contemplado en las variables anteriores.

Además del salario aparece el coste de la instalación del sistema completo tendrá un coste único de 300 € siempre y cuando la ubicación de la instalación cumpla con lo necesario para llevarla a cabo, es decir, el espacio adecuado, tomas de corrientes conectadas a la red eléctrica y conexión a internet.

También se incluye la preparación de la oferta inicial con un coste de 1.000 €, dietas y traslados que puedan surgir durante el transcurso del proyecto con un coste estimado en 300 € al mes y un curso de formación curso que será impartido a los usuarios del sistema que tendrá un coste único de 600 €.

La suma total del coste del trabajo personal es 11.241,54 €.

#### 7.2.4.1.3. Equipamiento e instalaciones

El equipamiento e instalaciones contemplan el coste de los elementos que son necesarios para realizar el proyecto y que se intentan amortizar con la realización de este.

La localización donde se llevarán a cabo los trabajos será compartida entre un laboratorio de la Universidad de Oviedo y la vivienda del desarrollador por lo que no se computan en el presupuesto gastos de oficina. De todas formas, si se computan gastos de luz, agua, tóner, etc. con un coste de 400 € mensuales que hacen 1.415,38 € para la duración total del proyecto y gastos de internet con un coste de 50 € mensuales que hacen 176,92 € durante la realización de este proyecto.

El equipamiento que usará el desarrollador ha tenido un coste de 1.200 € y que se planea sustituir a los 4 años de vida, es decir, a los 48 meses. Además, con el fin de contemplar posibles averías o un incremento del precio de adquisición de nuevo equipamiento una vez transcurridos los 48 meses, se ha incrementado el coste a amortizar en 200 €. Teniendo en cuenta esto, podemos concluir que amortizar este equipamiento mes a mes tiene un coste de 29,17 € mensuales, es decir, 103,21 € para este proyecto.

La suma de estos costes hace un total de 1.695,51 €.

#### 7.2.4.1.4. Software

En Software se contemplan los gastos relacionados con las licencias del software necesario para la realización de este proyecto.

En el caso de este proyecto solo se hace uso de 3 productos con licencia de pago, y que, en este caso, son suscripciones mensuales. El software requerido es el siguiente:

- **Microsoft Office 365**: Tiene un coste de licencia basada en subscripción mensual de 8,80 € y es necesario para la realización de la documentación, el presupuesto, la planificación y la presentación final entre otros fines.
- **JetBrains IDEs**: Los entornos de desarrollo de JetBrains tienen una licencia basada en subscripción mensual con un coste de 64,90 € el pack que incluye todos los IDEs. En el caso de este proyecto es necesario recurrir a dos de sus IDEs, PyCharm para Python y IntelliJ IDEA para Java.

- **Adobe CC**: La suite de Adobe para diseño gráfico tiene una licencia basada en subscripción mensual con un coste de 12,09 € e incluye Adobe Photoshop y Adobe Lightroom. Siendo usada la suite para para la creación de recursos gráficos para el proyecto.

La suma total de los costes en licencias de software es 303,56 €.

#### 7.2.4.1.5. Otros gastos

En otros gastos se incluyen los demás gastos del proyecto.

Se han añadido unos gastos de imprevistos y reservas que puedan surgir a lo largo del proyecto con un valor total de 2.500 € independiente de la duración de este. Con estos gastos se pretende cubrir cualquier gasto no contemplado como la adquisición de más materiales o equipamiento, o la contratación de más personal.

Además, se han añadido los gastos mensuales de la gestoría que ayuda al desarrollador a realizar tareas de gestión necesarias para poder realizar su oficio. Esto tiene un coste de 60 € mensuales, es decir, 212,31 € durante este proyecto.

Por último, se añade el coste de la seguridad social del desarrollador que se está, un 35% de su salario según las variables anteriormente definidas, en este caso unos 5,25 € la hora que hacen un total de 2.898,00 € para el proyecto completo.

La suma de estos costes hace un total de 5.610,31 €.

#### 7.2.4.1.6. Total

Una vez calculadas todas las partes del presupuesto, se suman para obtener el total del presupuesto de costes. En este caso el total asciende a 22.150,92 €.

Este total es el coste interno, no es la cifra que se entregará al cliente ya que le hay que aplicar un margen de beneficios.

#### 7.2.4.1.7. Presupuesto

| Concepto | Recurso | Unidades | Precio Unidad | Precio/hora | Precio/mes | Total Recurso | Total Concepto |
|---|---|---|---|---|---|---|---|
| Materiales | | | | | | | 3.300,00 € |
| | Servidor | 1 | 3.000,00 € | | | 3.000,00 € | |
| | Raspberry Pi + Extras | 1 | 70,00 € | | | 70,00 € | |
| | Arduino | 1 | 30,00 € | | | 30,00 € | |
| | Kit sensores y actuadores | 1 | 200,00 € | | | 200,00 € | |
| Trabajo personal | | | | | | | 11.241,54 € |
| | Salario desarrollador | 1 | | 15,00 € | 2.340,00 € | 8.280,00 € | |
| | Instalación del sistema | 1 | 300,00 € | | | 300,00 € | |
| | Preparación de la oferta inicial | 1 | 1.000,00 € | | | 1.000,00 € | |
| | Dietas y traslados | 1 | | | 300,00 € | 1.061,54 € | |
| | Curso de formación de los usuarios | 1 | 600,00 € | | | 600,00 € | |
| Equipamiento e instalaciones | | | | | | | 1.695,51 € |
| | Conexión a internet | 1 | | | 50,00 € | 176,92 € | |
| | Equipamiento de los desarrolladores | 1 | | | 29,17 € | 103,21 € | |
| | Otros gastos (luz, agua, tóner, etc.) | 1 | | | 400,00 € | 1.415,38 € | |
| Software | | | | | | | 303,56 € |
| | Microsoft Office 365 | 1 | | | 8,80 € | 31,14 € | |
| | JetBrains IDEs | 1 | | | 64,90 € | 229,65 € | |
| | Adobe CC | 1 | | | 12,09 € | 42,78 € | |
| Otros gastos | | | | | | | 5.610,31 € |
| | Imprevistos y reservas | 1 | 2.500,00 € | | | 2.500,00 € | |
| | Gastos de gestoría | 1 | | | 60,00 € | 212,31 € | |
| | Seguridad Social del programador | 1 | | 5,25 € | 819,00 € | 2.898,00 € | |
| | | | | | | **TOTAL** | 22.150,92 € |

Tabla 4. Presupuesto de costes.

### *7.2.4.2. Presupuesto de cliente*

En este apartado se desglosará el presupuesto de cliente explicando cada uno de los conceptos que aparece en él.

El presupuesto está dividido en 2 categorías, materiales y trabajo personal.

#### 7.2.4.2.1. Materiales

Se trata del mismo concepto explicado en el presupuesto de costes. En esta categoría aparece el hardware necesitado para este proyecto como parte de sus requisitos, que no es amortizable con otros proyectos posteriores puesto que pasará a posesión del cliente y que por tanto su gasto se computa en este proyecto

No hay ninguna diferencia entre esta categoría y la del presupuesto de costes. Para más detalles véase 7.2.4.1.1.

#### 7.2.4.2.2. Trabajo personal

Se trata del mismo concepto explicado en el presupuesto de costes. En esta categoría aparecen todos los trabajos realizados de una manera clara para el cliente, es decir, no se mostrarán salarios, ni traslados,... sino que se mostrarán cada uno de los trabajos realizados con un coste asociado.

En esta categoría se muestran los distintos módulos que componen el sistema global con un coste asociado a cada uno de los módulos. Para calcular el coste de cada uno de los módulos, se han cogido los costes del proyecto que influyen en su desarrollo y se ha calculado el coste por hora de todos ellos juntos. También se ha sumado el beneficio por hora que queremos obtener. Después, se ha calculado el número de horas de cada módulo y se ha aplicado el coste por hora antes calculado para obtener el coste final de cada módulo:

- Módulo de actuadores con DSL textual: 4.378,91 €
- Módulo sensores con DSL textual: 3.892,37 €
- Módulo reglas combinatorias con DSL textual: 3.649,09 €
- Módulo de actuadores con DSL gráfico: 3.892,37 €
- Modulo sensores con DSL gráfico: 3.405,82 €
- Módulo reglas combinatorias con DSL gráfico: 3.162,55 €

Para calcular el número de horas del proyecto se han cogido las horas que no corresponden al desarrollo de ninguno de los módulos, se han sumado y repartido entre todos los módulos. Estas horas son las dedicadas a la obtención de requisitos, diseño de la arquitectura, etc.

También se incluyen aquí el coste de la instalación del sistema (300 €) y el coste del curso de formación para los usuarios (600 €).

La suma de estos costes hace un total de 23.281,11 €.

#### 7.2.4.2.3. Total

Una vez calculadas cada una de las partes anteriores se realiza la suma de todos los costes obteniendo el total sin impuestos, 26.581,11 €. Este total ya contempla el 20% de beneficio que se ha computado en el coste por hora de cada módulo.

Para entregárselo al cliente también hay que aplicarle los impuestos necesarios, en este caso, el 21% de IVA que hacen un total de 32.163,14 €.

#### 7.2.4.2.4. Condiciones del presupuesto

A la hora de entregar al cliente el presupuesto, no solo se debe entregar el coste del proyecto, sino que también es necesario concretar todos los detalles posibles para que no haya ninguna equivocación en el transcurso del desarrollo y tras este.

Para este proyecto se exponen las siguientes condiciones que el cliente debe aceptar si quiere que se lleve a cabo la realización del proyecto.

- La localización donde se instalará el sistema será adecuada. Es decir, contará con el espacio necesario para realizar la instalación del servidor, con tomas de corriente eléctrica y con conexión a internet.
- El cliente contará con una garantía de un año durante el que se realizarán actualizaciones de seguridad gratuitas y se resolverán hasta 30 incidencias que conlleven menos de 2 horas de trabajo. En caso de superar las 30 incidencias o las 2 horas de trabajo, se facturará por separado el exceso.
- La validez del presupuesto será de 30 días desde la entrega del mismo. Una vez pasado este periodo será necesario volver a solicitar presupuesto con el riesgo de que varíe el coste final.
- La licencia final del producto será del cliente que tendrá todos los derechos sobre el producto excepto la propiedad intelectual que será del desarrollador, tal y como rige la ley.

#### 7.2.4.2.5. Presupuesto

| Concepto | Recurso | Unidades | Precio Unidad | Total Recurso | Total Concepto |
|---|---|---|---|---|---|
| Materiales | | | | | 3.300,00 € |
| | Servidor | 1 | 3.000,00 € | 3.000,00 € | |
| | Raspberry Pi + Extras | 1 | 70,00 € | 70,00 € | |
| | Arduino | 1 | 30,00 € | 30,00 € | |
| | Kit sensores y actuadores | 1 | 200,00 € | 200,00 € | |
| Trabajo personal | | | | | 23.281,11 € |
| | Módulo actuadores con DSL textual | 1 | 4.378,91 € | 4.378,91 € | |
| | Módulo sensores con DSL textual | 1 | 3.892,37 € | 3.892,37 € | |
| | Módulo reglas combinatorias con DSL textual | 1 | 3.649,09 € | 3.649,09 € | |
| | Módulo actuadores con DSL gráfico | 1 | 3.892,37 € | 3.892,37 € | |
| | Modulo sensores con DSL gráfico | 1 | 3.405,82 € | 3.405,82 € | |
| | Módulo reglas combinatorias con DSL gráfico | 1 | 3.162,55 € | 3.162,55 € | |
| | Instalación del sistema | 1 | 300,00 € | 300,00 € | |
| | Curso de formación de los usuarios | 1 | 600,00 € | 600,00 € | |
| | | | | **TOTAL** | 26.581,11 € |
| | | | | **IVA (21%)** | 5.582,03 € |
| | | | | **TOTAL + IVA** | 32.163,14 € |

Tabla 5. Presupuesto de cliente.

# Capítulo 8. Difusión de los resultados

En este capítulo se presentan las publicaciones preparadas y enviadas para publicar tanto en revistas indexadas en el índice JCR (*Journal Citation Reports*), revistas científicas no indexadas y congresos. El orden de las publicaciones aquí mostrado se basa en el orden de realización de los artículos finalizando con el artículo correspondiente al proyecto principal de este trabajo fin de máster.

## 8.1. Bilrost: Domain-Specific Language to define actions for the Internet of Things actuators, triggered by Twitter users' posts

Este artículo trata una primera aportación de esta investigación, el control de objetos inteligentes a través de Twitter mediante los mensajes que los usuarios publican en la red social. Su contenido forma parte de esta investigación ya que se trata la interacción de objetos y personas, la integración de objetos inteligentes en las redes sociales y la creación de un DSL, la primera versión de BSL, que facilita la generación de aplicaciones que permiten a los objetos ser manejados a través de las redes sociales.

La motivación para la realización de este artículo fue la consideración de que manejar objetos inteligentes a través de redes sociales ya era una buena contribución respecto al estado de la literatura actual. Además, había aparecido un *Special Issue* en una revista indexada en el índice JCR (*Journar Citation Report*) sobre Internet de las Cosas. La revista era *Computer Communications* y el nombre del *Special Issue* fue *the Internet of Things: Research challenges and Solutions.*

El artículo resultó rechazado con la justificación de que el número de participantes en la evaluación era bajo. Tras esto se ha aumentado el número de participantes en el proceso de evaluación y se han analizado diferentes revistas para enviarlo.

### 8.1.1. Special Issue: IoT Challenges

A continuación, se muestran los datos de la revista del *special issue* donde se ha enviado el artículo en primer lugar.

- **Título**: Computer Communications
- **Subtítulo**: The International Journal for the Computer and Telecommunications Industry
- **Editorial**: Elsevier
- **ISSN**: 0140-3664
- **URL**: http://www.journals.elsevier.com/computer-communications
- **Índice de impacto:** 1.695
- **Índice de impacto de 5 años:** 1.625
- **Clasificación por categoría** (Año JCR: 2014):
  - *COMPUTER SCIENCE, INFORMATION SYSTEMS*: 33/139 – Q1
  - *ENGINEERING, ELECTRICAL & ELECTRONIC*: 88/249 – Q2

Figura 25. Carátula Computer Communications.

- **Velocidad de la revista** (2015): 12,2 semanas para la primera decisión y 17 semanas para la decisión final.

## 8.1.2. Análisis de revistas

A continuación, se muestra el análisis de las revistas del índice JCR que se han valorado para enviar el artículo.

### *8.1.2.1. Journal of Network and Computer Applications*

- **Editorial**: Elsevier
- **ISSN**: 1084-8045
- **URL**: http://www.journals.elsevier.com/journal-of-network-and-computer-applications
- **Índice de impacto:** 2.229
- **Índice de impacto de 5 años:** 2.223
- **Clasificación por categoría** (Año JCR: 2014):
  - *COMPUTER SCIENCE, HARDWARE & ARCHITECTURE*: 5/50 – Q1
  - *COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS*: 20/102 – Q1
  - *COMPUTER SCIENCE, SOFTWARE ENGINEERING*: 7/104 – Q1
- **Velocidad de la revista** (2015): 12,3 semanas para la primera decisión y 15,1 semanas para la decisión final.


Figura 26. Carátula de la revista Journal of Network and Computer Applications.

### *8.1.2.2. Pervasive and Mobile Computing*

- **Editorial**: Elsevier
- **ISSN**: 1574-1192
- **URL**: http://www.journals.elsevier.com/pervasive-and-mobile-computing
- **Índice de impacto:** 2.079
- **Índice de impacto de 5 años:** 2.325
- **Clasificación por categoría** (Año JCR: 2014):
  - *COMPUTER SCIENCE, INFORMATION SYSTEMS*: 20/139 – Q1
- **Velocidad de la revista** (2015): 12,2 semanas para la primera decisión y 17,5 semanas para la decisión final.

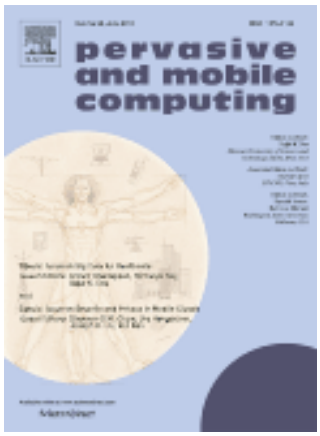



Figura 27. Carátula de la revista Pervasive and Mobile Computing.

### *8.1.2.3. Future Generation Computer Systems*

- **Subtítulo**: The International Journal of eScience
- **Editorial**: Elsevier
- **ISSN**: 0167-739X
- **URL**: http://www.journals.elsevier.com/future-generation-computer-systems/
- **Índice de impacto:** 2.786
- **Índice de impacto de 5 años:** 2.464
- **Clasificación por categoría** (Año JCR: 2014):
  - *COMPUTER SCIENCE, THEORY & METHODS*: 8/102 – Q1
- **Velocidad de la revista** (2015): 11,6 semanas para la primera decisión y 16,9 semanas para la decisión final.


Figura 28. Carátula de la revista Future Generation Computer Systems.

### *8.1.2.4. Computer Networks*

- **Subtítulo**: The International Journal of Computer and Telecommunications Networking
- **Carátula**:
- **Editorial**: Elsevier
- **ISSN**: 1389-1286
- **URL**: http://www.journals.elsevier.com/future-generation-computer-systems/
- **Índice de impacto:** 1.256
- **Índice de impacto de 5 años:** 1.714
- **Clasificación por categoría** (Año JCR: 2014):
  - *COMPUTER SCIENCE, HARDWARE & ARCHITECTURE*: 18/50 – Q2
  - *COMPUTER SCIENCE, INFORMATION SYSTEMS*: 54/139 – Q2
  - *ENGINEERING, ELECTRICAL & ELECTRONIC*: 122/249 – Q2
  - *TELECOMMUNICATIONS*: 32/77 – Q2
- **Velocidad de la revista** (2015): 9,6 semanas para la primera decisión y 15 semanas para la decisión final.


Figura 29. Carátula de la revista Computer Networks.

### *8.1.2.5. Journal of Systems and Software*

- **Editorial**: Elsevier
- **ISSN**: 0164-1212
- **URL**: http://www.journals.elsevier.com/journal-of-systems-and-software/
- **Índice de impacto:** 1.352
- **Índice de impacto de 5 años:** 1.485
- **Clasificación por categoría** (Año JCR: 2014):
  - *COMPUTER SCIENCE, SOFTWARE ENGINEERING*: 33/104 – Q2
  - *COMPUTER SCIENCE, THEORY & METHODS*: 31/102 – Q2
- **Velocidad de la revista** (2015): 8,6 semanas para la primera decisión y 14,5 semanas para la decisión final.


Figura 30. Carátula de la revista Journal of Systems and Software.

### *8.1.2.6. Personal and Ubiquitous Computing*

- **Editorial**: Springer
- **ISSN**: 1617-4909
- **URL**: http://www.springer.com/computer/hci/journal/779
- **Índice de impacto:** 1.518
- **Índice de impacto de 5 años:** 1.577
- **Clasificación por categoría** (Año JCR: 2014):
  - *COMPUTER SCIENCE, INFORMATION SYSTEMS*: 41/139 – Q2
  - *TELECOMMUNICATIONS*: 25/77 – Q2
- **Velocidad de la revista** (2014): 17 días desde la aceptación hasta la publicación.

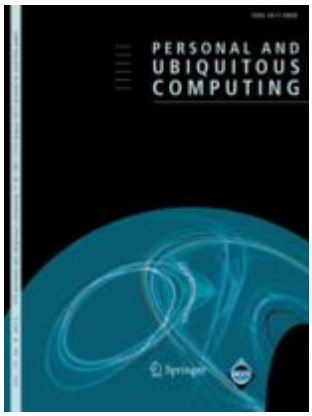



Figura 31. Carátula de la revista Personal and Ubiquitous Computing.

## 8.1.3. Decisiones tomadas

Tras el rechazo de la revista *Computer Communications* se realizó el análisis ya mostrado para seleccionar una revista a donde enviar este artículo y también el artículo final puesto que tratan del mismo proyecto.

La decisión tomada fue enviar a dos de las revistas con mayor índice de impacto entre las que se encuentran las siguientes:

- **Future Generation Computer Systems**
  - Índice de impacto: 2.786
  - Índice de impacto de 5 años: 2.464

  Esta revista ha actualizado su ámbito recientemente permitiendo que la propuesta de este trabajo fin de máster pueda encajar en ella ya que anteriormente no encajaba. Esta revista se ha seleccionado para enviar este artículo ya que la siguiente revista se ha escogido para enviar el artículo del proyecto completo y no se creía conveniente enviar los dos a la misma revista.

- **Journal of Network and Computer Applications**
  - Índice de impacto: 2.229
  - Índice de impacto de 5 años: 2.223

  Esta revista es la que mejor encaja con esta propuesta de las 3 seleccionadas y por ello se ha seleccionado para enviar el artículo correspondiente al sistema completo ya que las posibilidades de su publicación son mayores.

- **Pervasive and Mobile Computing**
  - Índice de impacto: 2.079
  - Índice de impacto de 5 años: 2.325

  Esta revista encaja con el proyecto ya que trata, entre otros temas, comunicaciones de dispositivos, redes de sensores, tecnologías de comunicación, servicios y protocolos en la capa de aplicación, etc. Sin embargo, tiene establecido un máximo de páginas (25) y un límite de palabras en el *abstract* (100).

Finalmente se ha decidido por enviar el artículo a la revista *Future Generation Computer Systems*. Se envió el día 20 de mayo del 2016. En el momento de la redacción de este documento, su estado es «*With editor*» y se le ha asignado el número de referencia FGCS-D-16-00425. La Figura 32 muestra una captura del estado del artículo.

| Manuscript Number | Title | Initial Date Submitted | Status Date | Current Status |
|---|---|---|---|---|
| FGCS-D-16-00425 | Bilrost: Domain-Specific Language to define actions for the Internet of Things actuators, triggered by Twitter users' posts | 20 May 2016 | 22 May 2016 | With Editor |

Figura 32. Estado del artículo Bilrost: Domain-Specific Language to define actions for the Internet of Things actuators, triggered by Twitter users' posts.

El artículo enviado se puede encontrar en el Anexo III.

## 8.2. IoFClime: The fuzzy logic and the Internet of Things to control indoor temperature regarding the outdoor ambient conditions

Este artículo se corresponde con la investigación paralela tratada en el apartado 3.7.4.1. En él, se propone un nuevo sistema de automatización de la temperatura de una habitación teniendo en cuenta la temperatura exterior, la humedad relativa exterior y la temperatura interior. A partir de estas variables, se propone el uso de la lógica difusa para controlar la temperatura de una habitación mediante el encendido y apagado de la calefacción y del aire acondicionado con la finalidad de ahorrar energía optimizando los apagados y encendidos.

La motivación para la realización de este artículo fue la realización de un trabajo fin de máster de la rama profesional dirigido por miembros del grupo de investigación MDE-RG. Una vez finalizado el trabajo fin de máster se propuso la realización de un artículo que refleje el sistema creado.

Durante la realización del artículo, se encontró un *Special Issue* sobre Internet de las Cosas aplicado en la industria donde este artículo encajaba. La revista de ese *Special Issue* fue *Computer Networks*, ya analizada en 8.1.2.4. , y el nombre del *Special Issue* fue *Industrial Technologies and Applications for the Internet of Things.*

El artículo fue rechazado pero los revisores han aportado una retroalimentación que ha sido usada para mejorar el artículo y preparar su envío para otro *Special Issue* de la revista *Future Generation Computer Systems*, ya analizada en 8.1.2.3. El título del *Special Issue* fue *Smart City and Internet of Things.*

Se envió el día 20 de noviembre del 2016. En el momento de la redacción de este documento, su estado es «*Under Review*» y tiene asignado el número de referencia FGCS-D-15-00824. La Figura 33 muestra una captura del estado del artículo.

| Manuscript Number | Title | Initial Date Submitted | Status Date | Current Status |
|---|---|---|---|---|
| FGCS-D-15-00824 | IoFClime: The fuzzy logic and the Internet of Things to control indoor temperature regarding the outdoor ambient conditions. | 20 Nov 2015 | 23 Jan 2016 | Under Review |

Figura 33. Estado del artículo IoFClime: The fuzzy logic and the Internet of Things to control indoor temperature regarding the outdoor ambient conditions.

El artículo enviado se puede encontrar en el Anexo IV.

## 8.3. Midgar: Creation of a graphic Domain-Specific Language to generate Smart Objects for the Internet of Things scenarios using Model-Driving Engineering

Este artículo se corresponde con la investigación paralela tratada en el apartado 3.7.4.2. . En él, se propone la ampliación de la plataforma Midgar (González García, Pelayo G-Bustelo, et al., 2014) mediante un nuevo DSL cuyo fin es la generación de objetos inteligentes completos. De esta manera, se consigue que cualquier usuario, aunque no tengan conocimientos de desarrollo de software, sea capaz de generar aplicaciones para sus dispositivos (siempre que estén soportados por la plataforma).

Durante la realización del artículo, se encontró un *Special Issue* sobre Internet de las Cosas aplicado en la industria donde este artículo encajaba. La revista de ese *Special Issue* fue *Computer Networks*, ya analizada en 8.1.2.4. , y el nombre del *Special Issue* fue *Industrial Technologies and Applications for the Internet of Things.*

El artículo fue rechazado pero los revisores han aportado una retroalimentación que se usará para mejorar el artículo y preparar su envío a otra revista.

Actualmente, se están aplicando los cambios propuestos por los revisores del envío anterior y se enviará más adelante.

El artículo enviado al *Special Issue* se puede encontrar en el Anexo V.

## 8.4. Midgar: Detection of people through computer vision in the Internet of Things scenarios to improve the security in Smart Cities, Smart Towns, and Smart Homes

Este artículo se corresponde con la investigación paralela tratada en el apartado 3.7.4.3. En él, se propone la ampliación de la plataforma Midgar (González García, Pelayo G-Bustelo, et al., 2014) con un nuevo módulo de visión por computador. El objetivo es mejorar la seguridad de las ciudades, pueblos u hogares mediante la detección de presencia de personas y la notificación de ello a través de la plataforma Midgar.

Durante la realización del artículo, se encontró un *Special Issue* sobre las ciudades inteligentes e Internet de las Cosas donde este artículo encajaba. La revista de ese *Special Issue* fue *Computer Networks*, ya analizada en 8.1.2.4. , y el nombre del *Special Issue* fue *Smart City and Internet of Things.*

Se envió el día 28 de octubre del 2016. En el momento de la redacción de este documento, su estado es «*Under Review*» y tiene asignado el número de referencia FGCS-D-15-00729. La Figura 34 muestra una captura del estado del artículo.

| Manuscript Number | Title | Initial Date Submitted | Status Date | Current Status |
|---|---|---|---|---|
| FGCS-D-15-00729 | Midgar: Detection of people through computer vision in the Internet of Things scenarios to improve the security in Smart Cities, Smart Towns, and Smart Homes | 28 Oct 2015 | 15 May 2016 | Under Review |

Figura 34. Estado del artículo Midgar: Creation of a graphic Domain-Specific Language to generate Smart Objects for the Internet of Things scenarios using Model-Driving Engineering.

El artículo enviado se puede encontrar en el Anexo VI.

## 8.5. Bilrost: Connecting the Internet of Things through human social networks with a Domain-Specific Language

Este artículo se realizó debido a un congreso que ha surgido durante el desarrollo de este trabajo fin de máster. Uno de los miembros del grupo de investigación MDE-RG recibió la propuesta de realizar un congreso y se presentó la propuesta de este trabajo fin de máster. El artículo trata la propuesta del sistema desarrollado en este trabajo fin de máster.

El artículo fue aceptado para su presentación en el congreso *International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) – 2016,* como se muestra en la Figura 35.

Respected Sir/madam,

Congratulation! For acceptance of your paper in IEEE-ICEEOT conference which is going to organize in DMI college of Engineering Chennai, Tamilnadu, India. The paper details as follows:

Title of the paper: Bilrost: Connecting the Internet of Things through human social networks with a Domain-Specific Language

List of Authors: Daniel Meana-Llorián , Cristian González, Jordán Pascual Espada and Vijay Bhaskar Semwa

Paper Code: **SS-RIDCE 3-09**

Name of the Session: **Research in Intelligent Device and Computing in Engineering**

Hope for see your wonderful presentation at conference. For more details, please find the attachments.

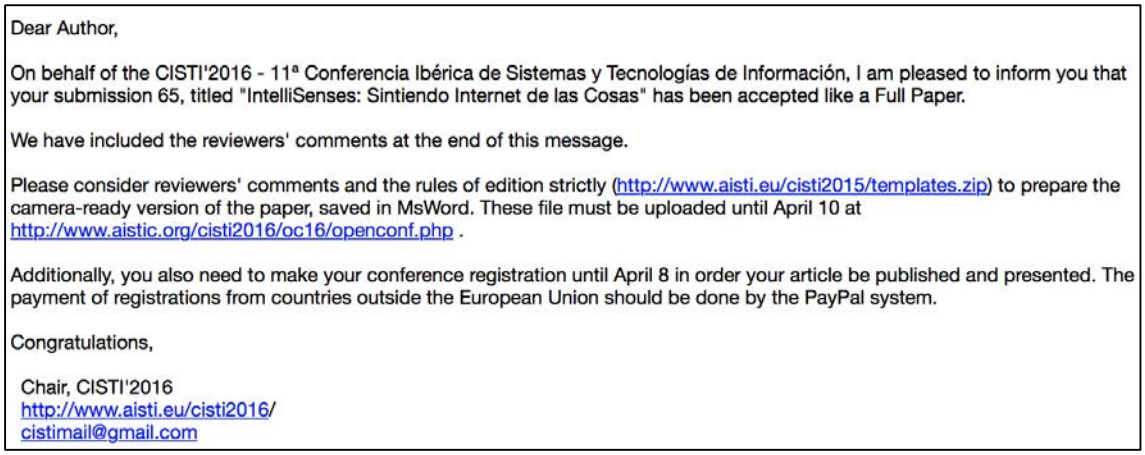
Figura 35. Aceptación del artículo Bilrost: Connecting the Internet of Things through human social networks with a Domain-Specific Language.

El artículo enviado se puede encontrar en el Anexo VII.

## 8.6. IntelliSenses: Sintiendo Internet de las Cosas

Este artículo se realizó para un congreso propuesto por uno de los integrantes del grupo de investigación MDE-RG. En él se propone la creación de un sistema que emule los 5 sentidos humanos a través de distintos módulos que emulan cada uno de los sentidos. La finalidad de la propuesta es crear un sistema inteligente que tome decisiones teniendo en cuenta los datos recopilados por todos los sentidos y no en base a uno solo como muchas de las investigaciones ya realizadas.

El artículo fue aceptado, con cambios menores, para su presentación en el congreso CISTI'2015 - 10ª Conferencia Ibérica de Sistemas y Tecnologías de Información, como se muestra en la Figura 36.



Dear Author,

On behalf of the CISTI'2016 - 11ª Conferencia Ibérica de Sistemas y Tecnologías de Información, I am pleased to inform you that your submission 65, titled "IntelliSenses: Sintiendo Internet de las Cosas" has been accepted like a Full Paper.

We have included the reviewers' comments at the end of this message.

Please consider reviewers' comments and the rules of edition strictly (http://www.aisti.eu/cisti2015/templates.zip) to prepare the camera-ready version of the paper, saved in MsWord. These file must be uploaded until April 10 at http://www.aistic.org/cisti2016/oc16/openconf.php .

Additionally, you also need to make your conference registration until April 8 in order your article be published and presented. The payment of registrations from countries outside the European Union should be done by the PayPal system.

Congratulations,

Chair, CISTI'2016
http://www.aisti.eu/cisti2016/
cistimail@gmail.com

Figura 36. Aceptación del artículo IntelliSenses: Sintiendo Internet de las Cosas.

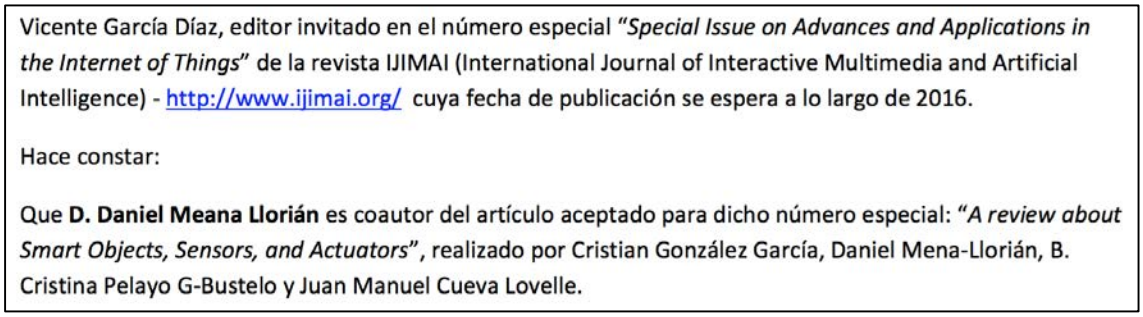
El artículo enviado y corregido según lo propuesto por los revisores se puede encontrar en el Anexo VIII.

## 8.7. A review about Smart Objects, Sensors, and Actuators

Este artículo se realizó para un *Special Issue* de la revista *The International Journal of Interactive Multimedia and Artificial Intelligence – IJIMAI*, organizado por uno de los miembros del grupo de investigación MDE-RG. En él se hace un repaso sobre los objetos inteligentes de Internet de las Cosas y los objetos no inteligentes como sensores y actuadores.

El nombre del *Special Issue* fue *Special Issue on Advances and Applications in the Internet of Things*.

El artículo fue enviado el 25 de marzo del 2016 y en la Figura 37, uno de los editores del *Special Issue* hace constar la aceptación del artículo. El artículo enviado y aceptado se puede encontrar en el Anexo IX.



Vicente García Díaz, editor invitado en el número especial "*Special Issue on Advances and Applications in the Internet of Things*" de la revista IJIMAI (International Journal of Interactive Multimedia and Artificial Intelligence) - http://www.ijimai.org/ cuya fecha de publicación se espera a lo largo de 2016.

Hace constar:

Que **D. Daniel Meana Llorián** es coautor del artículo aceptado para dicho número especial: "*A review about Smart Objects, Sensors, and Actuators*", realizado por Cristian González García, Daniel Mena-Llorián, B. Cristina Pelayo G-Bustelo y Juan Manuel Cueva Lovelle.

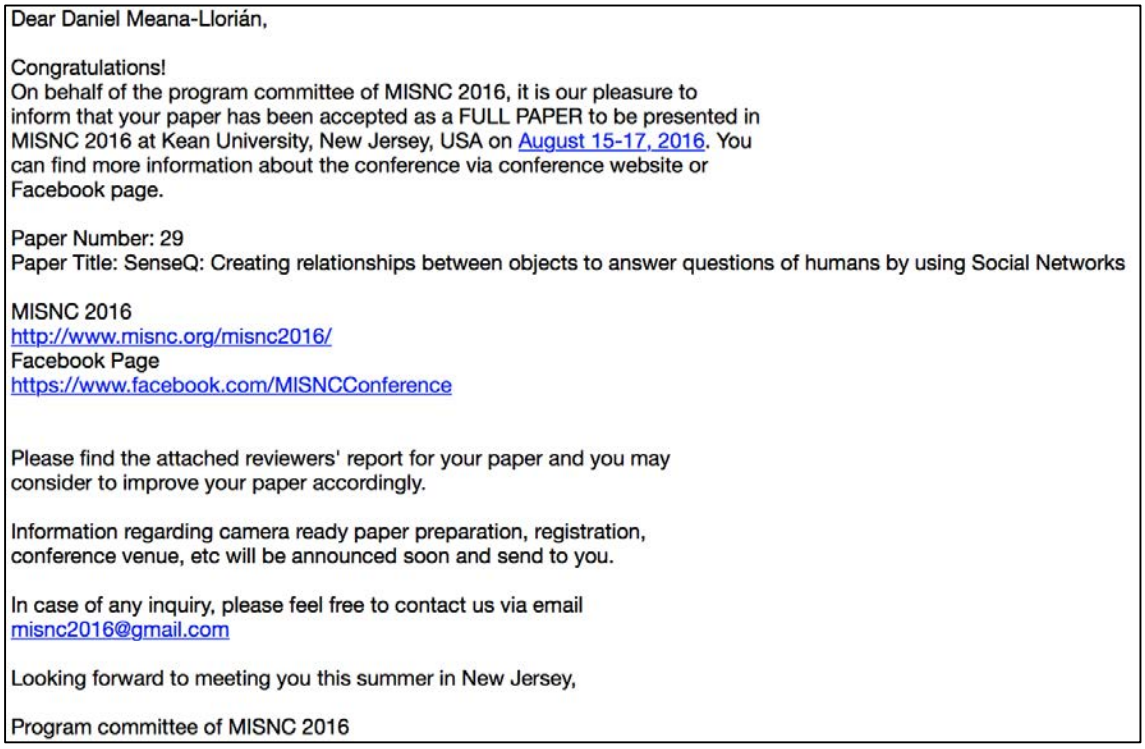
Figura 37. Aceptación del artículo A review about Smart Objects, Sensors, and Actuators.

## 8.8. SenseQ: Creating relationships between objects to answer questions of humans by using Social Networks

Este artículo se realizó para un congreso propuesto por uno de los integrantes del grupo de investigación MDE-RG. En él se propone la creación de un sistema mediante el que los usuarios de Twitter puedan comunicarse con sensores con el fin de obtener respuestas a preguntas. Para ello, los sensores establecen relaciones entre ellos de manera que una pregunta realizada en Twitter es contestada por el sistema tomando datos de todos los sensores relacionados.

Para lograr una colaboración de sensores se han propuesto 3 tipos de relaciones, relaciones basadas en la cercanía, relaciones basadas en la finalidad de los sensores y relaciones basadas en el propietario de los sensores.

El artículo fue aceptado, con la petición de cambios menores, para su presentación en el congreso MISNC 2016, The 3rd International Multidisciplinary Social Networks Conference, como se muestra en la Figura 38.



Dear Daniel Meana-Llorián,

Congratulations!
On behalf of the program committee of MISNC 2016, it is our pleasure to inform that your paper has been accepted as a FULL PAPER to be presented in MISNC 2016 at Kean University, New Jersey, USA on August 15-17, 2016. You can find more information about the conference via conference website or Facebook page.

Paper Number: 29
Paper Title: SenseQ: Creating relationships between objects to answer questions of humans by using Social Networks

MISNC 2016
http://www.misnc.org/misnc2016/
Facebook Page
https://www.facebook.com/MISNCConference

Please find the attached reviewers' report for your paper and you may consider to improve your paper accordingly.

Information regarding camera ready paper preparation, registration, conference venue, etc will be announced soon and send to you.

In case of any inquiry, please feel free to contact us via email
misnc2016@gmail.com

Looking forward to meeting you this summer in New Jersey,

Program committee of MISNC 2016

Figura 38. Aceptación del artículo SenseQ: Creating relationships between objects to answer questions of humans by using Social Networks.

El artículo enviado se puede encontrar en el Anexo X.

## 8.9. Bilrost: Using Domain-Specific Languages to connect Smart Objects through Social Networks

Este artículo es el artículo **principal** de este trabajo fin de máster. En él se plasma la propuesta completa de este proyecto resumiendo el estado de los conocimientos científico-técnicos, el trabajo relacionado, el sistema desarrollado, la evaluación, las conclusiones y el trabajo futuro plasmado en este documento.

Para su envío se analizaron las revistas ya planteadas en 8.1. y como se muestra en 8.1.3. la decisión fue enviarlo a la revista ***Journal of Network and Computer Applications*** ya que tiene un alto índice de impacto y encaja con el contenido del artículo.

Se envió el día 23 de mayo del 2016. En el momento de la redacción de este documento, su estado es «*With Editor*» y se le ha asignado el número de referencia JNCA-D-16-00576. La Figura 39 muestra una captura del estado del artículo.

| Manuscript Number | Title | Initial Date Submitted | Status Date | Current Status |
|---|---|---|---|---|
| JNCA-D-16-00576 | Bilrost: Using Domain-Specific Languages to connect Smart Objects through Social Networks | May 23, 2016 | May 23, 2016 | With Editor |

Figura 39. Estado del artículo Bilrost: Using Domain-Specific Languages to connect Smart Objects through Social Networks.

El artículo enviado se puede encontrar en el Anexo XI.

# Capítulo 9. Referencias

Anantharam, P., Barnaghi, P., Thirunarayan, K., & Sheth, A. (2015). Extracting City Traffic Events from Social Streams. *ACM Transactions on Intelligent Systems and Technology*, *6*(4), 1-27. http://doi.org/10.1145/2717317

AppMachine. (2011). AppMachine.

AppsBuilder SPA. (2013). AppsBuilder.

Appy Pie. (2015). Appypie.

Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, *54*(15), 2787-2805. http://doi.org/10.1016/j.comnet.2010.05.010

Atzori, L., Iera, A., & Morabito, G. (2014). From «smart objects» to «social objects»: The next evolutionary step of the internet of things. *IEEE Communications Magazine*, *52*(1), 97-105. http://doi.org/10.1109/MCOM.2014.6710070

Atzori, L., Iera, A., Morabito, G., & Nitti, M. (2012). The Social Internet of Things (SIoT) – When social networks meet the Internet of Things: Concept, architecture and network characterization. *Computer Networks*, *56*(16), 3594-3608. http://doi.org/10.1016/j.comnet.2012.07.010

Aurell, S. (2005). Remote controlling devices using instant messaging: building an intelligent gateway in Erlang/OTP. En *Proceedings of the 2005 ACM SIGPLAN workshop on Erlang* (pp. 46-51).

Bagchi, S. (2011). A fuzzy algorithm for dynamically adaptive multimedia streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications*, *7*(2), 1-26. http://doi.org/10.1145/1925101.1925106

Baqer, M. (2010). Enabling collaboration and coordination of wireless sensor networks via social networks. En *2010 6th IEEE International Conference on Distributed Computing in Sensor Systems Workshops (DCOSSW)* (pp. 1-2). IEEE. http://doi.org/10.1109/DCOSSW.2010.5593272

Baqer, M., & Kamal, A. (2009). S-Sensors: Integrating physical world inputs with social networks using wireless sensor networks. En *2009 International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)* (pp. 213-218). IEEE. http://doi.org/10.1109/ISSNIP.2009.5416815

BESTTOOLBARS. (2015). AppsGeyser.

Blackstock, M., Lea, R., & Friday, A. (2011). Uniting online social networks with places and things. En *Proceedings of the Second International Workshop on Web of Things* (pp. 5:1-5:6). New York, NY, USA: ACM. http://doi.org/10.1145/1993966.1993974

Borgia, E. (2014). The Internet of Things vision: Key features, applications and open issues. *Computer Communications*, *54*, 1-31. http://doi.org/10.1016/j.comcom.2014.09.008

Brooks. (1987). No Silver Bullet Essence and Accidents of Software Engineering. *Computer*, *20*(4), 10-19. http://doi.org/10.1109/MC.1987.1663532

Cacho, A., Figueredo, M., Cassio, A., Araujo, M. V., Mendes, L., Lucas, J., … Prolo, C. (2016). Social Smart Destination: A Platform to Analyze User Generated Content in Smart Tourism Destinations. En *New Advances in Information Systems and Technologies* (pp. 817-826). Springer. http://doi.org/10.1007/978-3-319-31232-3_77

Caragliu, A., Del Bo, C., & Nijkamp, P. (2011). Smart Cities in Europe. *Journal of Urban Technology*, *18*(2), 65-82. http://doi.org/10.1080/10630732.2011.601117

Chamodrakas, I., & Martakos, D. (2012). A utility-based fuzzy TOPSIS method for energy efficient network selection in heterogeneous wireless networks. *Applied Soft Computing Journal*, *12*(7), 1929-1938. http://doi.org/10.1016/j.asoc.2012.04.016

Choi, J., Park, S., Ko, H., Moon, H.-J., & Lee, J. (2009). Issues for Applying Instant Messaging to Smart Home Systems. En *Computational Science and Its Applications - Iccsa 2009, Pt I* (Vol. 5592, pp. 649-661). Springer. Recuperado a partir de <Go to ISI>://WOS:000271635000048

Choi, J., & Yoo, C. W. (2008). Connect with things through instant messaging. En *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 4952 LNCS, pp. 276-288). Springer. http://doi.org/10.1007/978-3-540-78731-0_18

Clarke, R. Y. (2013). Smart cities and the internet of everything: The foundation for delivering next-generation citizen services. *Alexandria, VA, Tech. Rep*.

Como Ltd. (2015). Como.

Cook, S., Jones, G., Kent, S., & Wills, A. C. (2007). *Domain-specific development with visual studio dsl tools*. Pearson Education.

Cueva-Fernandez, G., Espada, J. P., García-Díaz, V., Crespo, R. G., & Garcia-Fernandez, N. (2015). Fuzzy system to adapt web voice interfaces dynamically in a vehicle sensor tracking application definition. *Soft Computing*. http://doi.org/10.1007/s00500-015-1709-2

Cueva-Fernandez, G., Espada, J. P., García-Díaz, V., García, C. G., & Garcia-Fernandez, N. (2014). Vitruvius: An expert system for vehicle sensor tracking and managing application generation. *Journal of Network and Computer Applications*, *42*, 178-188. http://doi.org/10.1016/j.jnca.2014.02.013

Cueva-Fernandez, G., Pascual Espada, J., García-Díaz, V., & Gonzalez-Crespo, R. (2015). Fuzzy decision method to improve the information exchange in a vehicle sensor tracking system. *Applied Soft Computing*. http://doi.org/10.1016/j.asoc.2015.01.066

Deursen, A. van, & Klint, P. (1998). Little languages: Little maintenance? *Journal of software maintenance*, *10*(2), 75-92. http://doi.org/10.1016/j.jvs.2012.10.105

Deursen, A. Van, Klint, P., & Visser, J. (2000). Domain-specific languages: an annotated bibliography. *ACM Sigplan Notices*, *35*(6), 26-36.

Dijkstra, E. (1972). The humble programmer. *Communications of the ACM*, *15*(October 1972), 859-866.

Domingo, M. C. (2012). An overview of the Internet of Things for people with disabilities. *Journal of Network and Computer Applications*, *35*(2), 584-596. http://doi.org/10.1016/j.jnca.2011.10.015

Doran, D., Gokhale, S., & Dagnino, A. (2013). Human sensing for smart cities. En *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining - ASONAM '13* (pp. 1323-1330). New York, New York, USA: ACM Press. http://doi.org/10.1145/2492517.2500240

Fan, Y. J., Yin, Y. H., Member, S., Zeng, Y., & Wu, F. (2014). IoT-Based Smart Rehabilitation System. *Industrial Informatics, IEEE Transactions on*, *10*(2), 1568-1577.

Ford, N., & Davis, S. (2006). *No Fluff, Just Stuff Anthology*. Pragmatic Bookshelf.

Fundación Telefónica. (2016). *La Sociedad de la Información en España 2015*. Recuperado a partir de http://www.fundaciontelefonica.com/arte_cultura/publicaciones-listado/pagina-item-publicaciones/itempubli/483/

Gama, K., Touseau, L., & Donsez, D. (2012). Combining heterogeneous service technologies for building an Internet of Things middleware. *Computer Communications*, *35*(4), 405-417. http://doi.org/10.1016/j.comcom.2011.11.003

García-Díaz, V. (2011). MDCI: Model-driven continuous integration.

Garner, S. (2009). Learning to program from scratch. En *Advanced Learning Technologies, 2009. ICALT 2009. Ninth IEEE International Conference on* (pp. 451-452). Riga: IEEE. http://doi.org/10.1109/ICALT.2009.50

Georgitzikis, V., Akribopoulos, O., & Chatzigiannakis, I. (2012). Controlling physical objects via the internet using the arduino platform over 802.15.4 networks. *IEEE Latin America Transactions*, *10*(3), 1686-1689. http://doi.org/10.1109/TLA.2012.6222571

González García, C., Pascual Espada, J., Núñez-Valdez, E. R., & García-Díaz, V. (2014). Midgar: Domain-specific language to generate smart objects for an internet of things platform. *Proceedings - 2014 8th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2014*, 352-357. http://doi.org/10.1109/IMIS.2014.48

González García, C., Pelayo G-Bustelo, B. C., Pascual Espada, J., & Cueva-Fernandez, G. (2014). Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios. *Computer Networks*, *64*, 143-158. http://doi.org/10.1016/j.comnet.2014.02.010

González, E. P., Fernández, H. F., Díaz, V. G., Bustelo, B. C. P. G., Lovelle, J. M. C., & Martínez, O. S. (2008). General purpose MDE tools. *IJIMAI*, *1*(1), 72-75.

Grant, P. (2007). A new approach to diabetic control: Fuzzy logic and insulin pump technology. *Medical Engineering & Physics*, *29*(7), 824-827. http://doi.org/http://dx.doi.org/10.1016/j.medengphy.2006.08.014

Guinard, D., Fischer, M., & Trifa, V. (2010). Sharing using social networks in a composable Web of Things. En *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on* (pp. 702-707). http://doi.org/10.1109/PERCOMW.2010.5470524

Guinard, D., & Trifa, V. (2009). Towards the Web of Things : Web Mashups for Embedded Devices. En *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences)* (pp. 1-8). http://doi.org/10.1.1.155.3238

Guinard, D., Trifa, V., Pham, T., & Liechti, O. (2009). Towards physical mashups in the web of things. En *INSS2009 - 6th International Conference on Networked Sensing Systems* (pp. 196-199). http://doi.org/10.1109/INSS.2009.5409925

Hailpern, B., & Tarr, P. (2006). Model-driven development: The good, the bad, and the ugly. *IBM Systems Journal*, *45*(3), 451-461. http://doi.org/10.1147/sj.453.0451

Hasan, M., Hossain, E., & Niyato, D. (2013). Random access for machine-to-machine communication in LTE-advanced networks: issues and approaches. *IEEE Communications Magazine*, *51*(6), 86-93. http://doi.org/10.1109/MCOM.2013.6525600

Howe, B. M., Chao, Y., Arabshahi, P., Roy, S., McGinnis, T., & Gray, A. (2010). A Smart Sensor Web for Ocean Observation: Fixed and Mobile Platforms, Integrated Acoustics, Satellites and Predictive Modeling. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *3*(4), 507-521. http://doi.org/10.1109/JSTARS.2010.2052022

Hribernik, K. a., Ghrairi, Z., Hans, C., & Thoben, K.-D. (2011). Co-creating the Internet of Things - First experiences in the participatory design of Intelligent Products with Arduino. En *2011 17th International Conference on Concurrent Enterprising* (pp. 1-9).

iBuildApp. (2015). iBuildApp.

Indigo Rose Software. (2015). Andromo.

Infinite Monkeys. (2015). Infinite Monkeys.

International Telecommunication Union. (2012). Overview of the Internet of things. *Series Y: Global information infrastructure, internet protocol aspects and next-generation networks - Frameworks and functional architecture models*, 22. Recuperado a partir de https://www.itu.int/rec/T-REC-Y.2060-201206-I

Jara, A. J., Sun, Y., Song, H., Bie, R., Genooud, D., & Bocchi, Y. (2015). Internet of Things for Cultural Heritage of Smart Cities and Smart Regions. En *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops* (pp. 668-675). Gwangiu: IEEE. http://doi.org/10.1109/WAINA.2015.169

Kaucic, B., & Asic, T. (2011). Improving introductory programming with Scratch? En *2011 Proceedings of the 34th International Convention MIPRO* (pp. 1095-1100). Opatija: IEEE.

Kranz, M., Roalter, L., Michahelles, F., & Michahelles, F. (2010). Things That Twitter: Social Networks and the Internet of Things. En *What can the Internet of Things do for the Citizen (CIoT) Workshop at The Eighth International Conference on Pervasive Computing: Pervasive 2010* (pp. 1-10).

Kwak, H., Lee, C., Park, H., & Moon, S. (2010). What is Twitter, a social network or a news media? En *Proceedings of the 19th international conference on World wide web - WWW '10* (p. 591). New York, New York, USA: ACM Press. http://doi.org/10.1145/1772690.1772751

Larios, D. F., Barbancho, J., Molina, F. J., & León, C. (2012). LIS: Localization based on an intelligent distributed fuzzy system applied to a WSN. *Ad Hoc Networks*, *10*(3), 604-622. http://doi.org/10.1016/j.adhoc.2011.11.003

Lee, G. M., & Kim, J. Y. (2012). Ubiquitous networking application: Energy saving using smart objects in a home. En *2012 International Conference on ICT Convergence (ICTC)* (pp. 299-300). Jeju Island: Ieee. http://doi.org/10.1109/ICTC.2012.6386844

Li, S., Xu, L. Da, & Zhao, S. (2014). The internet of things: a survey. *Information Systems Frontiers*. http://doi.org/10.1007/s10796-014-9492-7

Lifelong Kindergarten. (2015). Scratch.

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, *22 140*, 55.

Mashal, I., Alsaryrah, O., Chung, T.-Y., Yang, C.-Z., Kuo, W.-H., & Agrawal, D. P. (2015). Choices for interaction with things on Internet and underlying issues. *Ad Hoc Networks*, *28*, 68-90. http://doi.org/10.1016/j.adhoc.2014.12.006

McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (2006). A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence. *AI Magazine*, *27*(4), 12-14. http://doi.org/10.1609/aimag.v27i4.1904

Meyer, G. G., Främling, K., & Holmström, J. (2009). Intelligent Products: A survey. *Computers in Industry*, *60*(3), 137-148. http://doi.org/10.1016/j.compind.2008.12.005

Miao Yun, & Bu Yuxin. (2010). Research on the architecture and key technology of Internet of Things (IoT) applied on smart grid. En *2010 International Conference on Advances in Energy Engineering* (pp. 69-72). IEEE. http://doi.org/10.1109/ICAEE.2010.5557611

MiniBloq. (s. f.).

Miranda, J., Makitalo, N., Garcia-Alonso, J., Berrocal, J., Mikkonen, T., Canal, C., & Murillo, J. M. (2015). From the Internet of Things to the Internet of People. *IEEE Internet Computing*, *19*(2), 40-47. http://doi.org/10.1109/MIC.2015.24

Mitchell, S., Villa, N., Stewart-Weeks, M., & Lange, A. (2013). The Internet of Everything for Cities: Connecting people, Process, Data, and Things to Improve the «Livability» of Cities and Communities, 1-21.

Murphy, F. E., Magno, M., O'Leary, L., Troy, K., Whelan, P., & Popovici, E. M. (2015). Big brother for bees (3B) - Energy neutral platform for remote monitoring of beehive imagery and sound. En *2015 6th International Workshop on Advances in Sensors and Interfaces (IWASI)* (pp. 106-111). Gallipoli: IEEE. http://doi.org/10.1109/IWASI.2015.7184943

National Intelligence Council. (2008). Discruptive civil tehnologies - six tehnologies with potential impacts on us interests out to 2025. *Conference Report CR 2008 - 07*. Apr. Recuperado a partir de http://www.dni.gov./nic/confreports_discruptive_tech.html

Naur, P., & Randell, B. (1968). Software Engineering, Conference Report. *NATO Science Committee, Garmisch (Germany)*, 7-11.

Núñez-Valdez, E. R., Sanjuan-Martinez, O., Bustelo, C. P. G., Lovelle, J. M. C., & Infante-Hernandez, G. (2013). Gade4all: Developing Multi-platform Videogames based on Domain Specific Languages and Model Driven Engineering. *International Journal of Interactive Multimedia and Artificial Intelligence*, *2*(Regular Issue), 33-42.

Pang, Z., Zheng, L., Tian, J., Kao-Walter, S., Dubrova, E., & Chen, Q. (2015). Design of a terminal solution for integration of in-home health care devices and services towards the Internet-of-Things. *Enterprise Information Systems*, *9*(1), 86-116. http://doi.org/10.1080/17517575.2013.776118

Paper, W. (2000). *Guarantee permanent Model/Code consistency: «Model driven Engineering» (MDE) vs «Roundtrip engineering» (RTE)*.

Pintus, A., Carboni, D., & Piras, A. (2012). Paraimpu: A platform for a social Web of Things. En *Proceedings of the 21st International Conference Companion on World Wide Web (WWW'12 Companion)* (pp. 401-404). http://doi.org/10.1145/2187980.2188059

Piras, A., Carboni, D., Pintus, A., & Features, D. M. T. (2012). A Platform to Collect , Manage and Share Heterogeneous Sensor Data. En *Networked Sensing Systems (INSS)* (pp. 1-2). Antwerp: IEEE. http://doi.org/10.1109/INSS.2012.6240570

PMI. (2013). A Guide to the Project Management Body of Knowledge (PMBOK® Guide)-Fifth Edition. *Project Management Journal*. http://doi.org/10.1002/pmj.21345

Portmann, E., Andrushevich, A., Kistler, R., & Klapproth, A. (2010). Prometheus - Fuzzy information retrieval for semantic homes and environments. En *3rd International Conference on Human System Interaction, HSI'2010 - Conference Proceedings* (pp. 757-762). http://doi.org/10.1109/HSI.2010.5514482

Ramparany, F., & Boissier, O. (2002). Smart devices embedding multi-agent technologies for a pro-active world. En *Proc. Uniquitous Computing Workshop*.

Russell, S., & Norvig, P. (1995). Artificial intelligence: a modern approach.

Sakaki, T., Okazaki, M., & Matsuo, Y. (2010). Earthquake Shakes Twitter Users : Real-time Event Detection by Social Sensors. En *Proceedings of the 19th international conference on World wide web - WWW '10* (p. 851). New York, New York, USA: ACM Press. http://doi.org/10.1145/1772690.1772777

Sanchez, L., Muñoz, L., Galache, J. A., Sotres, P., Santana, J. R., Gutierrez, V., … Pfisterer, D. (2014). SmartSantander: IoT experimentation over a smart city testbed. *Computer Networks*, *61*, 217-238. http://doi.org/10.1016/j.bjp.2013.12.020

Seidewitz, E. (2003). What models mean. *IEEE Software*, *20*(5), 26-32. http://doi.org/10.1109/MS.2003.1231147

Selic, B. (2003a). Model-driven development of real-time software using OMG standards. En *Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2003.* (pp. 4-6). IEEE. http://doi.org/10.1109/ISORC.2003.1199227

Selic, B. (2003b). Models, Software Models and UML. En *UML for Real* (pp. 1-16). Boston: Kluwer Academic Publishers. http://doi.org/10.1007/0-306-48738-1_1

Selic, B. (2008). MDA manifestations. *The European Journal for the Informatics Professional, IX (2)*, 12-16.

Sendall, S., & Kozaczynski, W. (2003). Model transformation: the heart and soul of model-driven software development. *IEEE Software*, *20*(5), 42-45. http://doi.org/10.1109/MS.2003.1231150

Song, H. (2013). Internet of things for rural and small town america. En *6th Annual Create West Virginia Training and Education Conference* (pp. 1-6).

Spinellis, D. (2001). Notable design patterns for domain-specific languages. *Journal of Systems and Software*, *56*(1), 91-99. http://doi.org/10.1016/S0164-1212(00)00089-3

Tan, L., & Wang, N. (2010). Future internet: The internet of things. En *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on* (Vol. 5, pp. V5-376).

Thomas, D. (2004). MDA: Revenge of the modelers or UML Utopia? *IEEE Software*, *21*(3), 15-17. http://doi.org/10.1109/MS.2004.1293067

Thomas, S., Cho, Y., & Srivastava, M. B. (2007). Exploiting Social Networks for Sensor Data Sharing with SenseShare. *Center for Embedded Network Sensing*.

Ventä, O. (2007). *Intelligent products and systems: Technology theme-final report*. VTT Technical Research Centre of Finland.

Vermesan, O., & Peter Friess. (2013). *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*. *Challenges*. River Publishers. http://doi.org/10.2139/ssrn.2324902

Vienna University of Technology. (2015). European Smart Cities.

Wang, C., Komodakis, N., & Paragios, N. (2013). Markov Random Field modeling, inference & learning in computer vision & image understanding: A survey. *Computer Vision and Image Understanding*, *117*(11), 1610-1627. http://doi.org/10.1016/j.cviu.2013.07.004

Zadeh, L. a. (1965). Fuzzy Sets. *Inf and Control*, *8*, 338-353. http://doi.org/10.1109/2.53

Zadeh, L. A. (1975). Fuzzy logic and approximate reasoning. *Synthese*, *30*(3-4), 407-428. http://doi.org/10.1007/BF00485052

Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of Things for Smart Cities. *IEEE Internet of Things Journal*, *1*(1), 22-32. http://doi.org/10.1109/JIOT.2014.2306328

Zheng, T., Qin, Y., Gao, D., Duan, J., & Zhang, H. (2010). A practical deployment of Intelligent Building Wireless Sensor Network for environmental monitoring and air-conditioning control. En *2010 2nd IEEE InternationalConference on Network Infrastructure and Digital Content* (pp. 624-628). IEEE. http://doi.org/10.1109/ICNIDC.2010.5657858

# Capítulo 10. Anexos

En este capítulo se encuentras los anexos del documento. Estos son los siguientes:

# ANEXO I. Acta de constitución del proyecto

UNIVERSIDAD DE OVIEDO

# ACTA DE CONSTITUCIÓN

**Título del proyecto:** Bilrost: Interconexión de objetos inteligentes a través de redes sociales

**Director del proyecto:** B. Cristina Pelayo García-Bustelo

**Fecha de preparación:** 3 de enero de 2016

**Jefe de proyecto:** Daniel Meana Llorián

**Cliente:** Universidad de Oviedo

**Justificación del proyecto:**

La motivación de este proyecto es el intentar usar tecnologías de Internet de las Cosas en combinación con redes sociales con el fin de atraer la atención de los usuarios hacia estas tecnologías sin necesidad de que deban tener un perfil informático o electrónico, es decir, sin necesidad de tener conocimientos avanzados. La temática de este proyecto se ha decido conjuntamente con el director del mismo para encajarlo en las investigaciones del grupo de investigación MDE-RG.

**Descripción del proyecto:**

Internet de las cosas, o IoT en sus siglas en inglés (*Internet of Things*), es un paradigma que está ganando mucho protagonismo desde la llegada de los dispositivos inteligentes, conocidos como Smart Objects, a nuestras vidas como los *smartphones*, *tablets*, *weareables*, *Smart Homes*, *Smart Cars*, *microordenadores*, *microcontroladores*, … Aún así, todavía no se ha definido un sistema de intercambio de información entre estos dispositivos amigable para cualquier persona sin necesidad de tener conocimientos de informática avanzados.

Pensando en esta falta de intercambio de información y en las personas sin conocimientos de informática avanzados ha surgido la idea de este proyecto, llamado ***Bilrost: Interconexión de objetos inteligentes a través de redes sociales***.

*Bilrost* pretende conectar dispositivos entre sí y con personas a través de una plataforma ya existente y conocida por la mayoría de los usuarios de Internet, la red social *Twitter*. Su funcionamiento basado en mensajes de corta extensión y públicos por defecto, permite a los dispositivos compartir su estado y leer el estado de otros dispositivos para realizar acciones concretas en función de los datos recopilados. Además, el uso de una red social de personas hace que las propias personas puedan ejercer el papel de otro dispositivo inteligente más, permitiéndoles controlar y leer otros dispositivos. Si nos acercamos más al campo de IoT, *Bilrost* funcionaría como una red social de objetos inteligentes y personas que permite compartir el estado de los sensores conectados a los objetos, transmitir a los actuadores de los objetos los datos necesarios para que realicen las acciones oportunas. Además, las propias personas podrían ser tratadas como objetos inteligentes, más concretamente como sensores, ya que podrían publicar estados que los actuadores interpretarían como datos de otros sensores. De esta manera, conseguiríamos interconectar los objetos con las personas de una forma fácil, sencilla y natural.

*Bilrost* estaría diseñado para que cualquier usuario pudiese configurar las conexiones a través de un lenguaje de dominio específico, DSL en su siglas en inglés (*Domain Specific Language*), En su primera etapa será textual para después llegar a ser gráfico con el fin de hacerlo más usable. Mediante este DSL, los usuarios podrán configurar todos los aspectos de la conexión entre dispositivos a través de *Twitter*. No obstante este proyecto se limitará a las conexiones entre dispositivos dejando la parte de programación concreta de actuadores y sensores a los propios usuarios pudiendo ser esto abarcado en un proyecto futuro derivado de este. Esta parte sería aquella que interpreta los datos recibidos de los sensores en los actuadores y aquella que interpreta los datos recopilados por los sensores y los emite a *Bilrost*.

**Requisitos del proyecto:**

Los requisitos iniciales del proyecto son los siguientes:

**Requisitos generales**

RG 1 El sistema estará deberá ser compatible con plataformas hardware de IoT, concretamente, Raspberry PI.
RG 2 Se adaptará a dispositivos Android para poder usar sus sensores y actuadores.
RG 3 El sistema podrá ejecutarse en PCs con el fin de automatizar tareas a modo de actuadores.
RG 4 El sistema intercomunicará sensores y actuadores a través de una o más redes sociales.
RG 5 El sistema podrá ser configurado mediante un DSL textual de alto nivel.
RG 6 El sistema podrá ser configura mediante un DSL gráfico.
RG 7 Cada módulo tendrá definido sus DSLs con una parte común usada para configurar la conexión a las redes sociales, la selección de la plataforma de salida y las palabras claves para filtrar la búsqueda de mensajes en las redes sociales.
RG 1.7 El sistema generará proyectos para Python, Java y Android.
RG 8 El sistema tendrá una aplicación Web que permita la generación de proyectos que conecten los objetos inteligentes a las redes sociales.
RG 1.8 La aplicación web usará el DSL gráfico para generar el DSL textual que se usará para generar los proyectos.
RG 2.8 La aplicación web debe contar con explicaciones para que cualquier usuario pueda usarla.
RG 3.8 La aplicación web se conectará a la red social seleccionada para obtener los datos necesarios para la conexión que variarán según la red social.
RG 9 La funcionalidad estará dividida en tres grandes módulos que pueden funcionar independientemente o en conjunto:
- Módulo de actuadores. (RA)
- Módulo de sensores. (RS)
- Módulo combinatorio sensores y actuadores. (RC)

**Requisitos del módulo de actuadores**

RA 1 El módulo de actuadores permitirá accionar actuadores a través de mensajes en las redes sociales seleccionadas.
RA 2 Los usuarios de la red social podrán publicar mensajes que accionen a los actuadores.
RA 3 Se deberá implementar un control permisos de usuarios.
RA 4 Los mensajes deberán contener el identificador y las acciones del actuador.
RA 5 La parte del DSL propia de este módulo contendrá lo necesario para recuperar los mensajes.

**Requisitos del módulo de sensores**

RS 1 El módulo de sensores permitirá compartir el estado de sensores a través de mensajes las redes sociales seleccionadas.
RS 2 Los usuarios de la red social podrán leer mensajes que contengan es el estado de los sensores.
RS 3 Los mensajes deberán contener el identificador del sensor, el estado y la hora y fecha.
RS 4 La parte del DSL propia de este módulo definirá lo necesario para publicar los mensajes.

**Requisitos del módulo combinación**

RC 1 El módulo combinación permitirá establecer condiciones en base a estados de sensores para realizar acciones de actuadores.
RC 2 El módulo podrá tanto publicar tanto acciones de los actuadores como leer estados de sensores de la red social.
RC 3 La parte del DSL propia de este módulo definirá lo necesario para establecer las condiciones y las consecuencias además de combinar los DSL de los otros dos módulos.

**Criterios de aceptación:**

Para que el proyecto sea aceptado se deberán cumplir los siguientes criterios:

- Deberán estar finalizada la conexión entre objetos inteligentes al 100% usando al menos una red social.
- La interfaz de usuario permitirá crear soluciones que se conecten a la red social a usuarios sin conocimientos avanzados de informática. Esto se reflejará en unas pruebas de usabilidad.
- Se deberá haber plasmado el resultado en por lo menos una publicación científica siendo esta una publicación de congreso, en revista o en revista JCR.
- La implementación de las acciones de los actuadores y de las mediciones de los sensores no tendrán que estar implementada pero deberá ser fácil su implementación sobre los proyectos generados.
- Deberá cumplir todos los requisitos y objetivos identificados.
- Los DSL usados tendrán una sintaxis legible al nivel de SQL.
- Existirá una documentación del proyecto que justifique su realización, compare las alternativas existentes y realice las aportaciones científicas del proyecto al campo de estudio.

**Riesgos iniciales:**

Los riesgos iniciales de este proyecto son los siguientes:

**Riesgos negativos**

- Enfermedad del desarrollador o accidente.
- Planificación optimista.
- Cambios en los requisitos.
- Limitaciones en las APIs de las redes sociales.
- Cambios continuos en las librerías usadas.
- Director de proyecto no disponible para las reuniones planificadas.
- Tareas mal estimadas.
- Desconocimiento de herramientas necesarias.
- Problemas en el hardware usado para el desarrollo y pruebas.
- Problemas de terceros durante la presentación.

**Riesgos positivos**

- Conocimiento de herramientas.
- Planificación pesimista.
- Reaprovechamiento de material anterior.
- Liberarías que faciliten el trabajo.

| Objetivos del proyecto | Criterios de éxito | Supervisor |
|---|---|---|
| **Alcance:** | | |
| Se deben generar proyectos que interconecten objetos a través de redes sociales sin conocimientos de programación pero será necesario programar las acciones y las lecturas de los sensores. Se hará una aplicación web que permita la generación de proyectos de forma sencilla e intuitiva. | Conseguir que usuarios que se presten voluntarios puedan generar aplicaciones que interconecten objetos según criterios que ellos establezcan y sobre ellos realizar implementaciones que satisfagan necesidades para comprobar que el proyecto tiene utilidad. | B. Cristina Pelayo García-Bustelo |
| **Tiempo:** | | |
| Finalizar el proyecto con márgenes suficientes entre módulo y módulo que permita realizar artículos relacionados con cada uno y disponer de tiempo al final para revisar y evaluar el proyecto. El tiempo máximo del proyecto deberá ser de 6 meses a contar desde finales de diciembre. | El proyecto deberá finalizarse y entregarse antes de finalizar el plazo, es decir, antes del 1 de junio del 2016. | B. Cristina Pelayo García-Bustelo |
| **Coste:** | | |
| El proyecto se deberá finalizar sin que el coste se desborde respecto al presupuesto planteado. | No se debe superar el coste ya indicado en el presupuesto, es decir, 24.887,50€ puesto que esta cifra ya cuenta con un margen para imprevistos. | B. Cristina Pelayo García-Bustelo |
| **Calidad:** | | |
| El proyecto será de calidad cuando cumpla todos los requisitos propuestos y los objetivos planteados. Además, debe haber superado todas las pruebas diseñadas, las pruebas accesibilidad y las de usabilidad realizadas por usuarios.<br>La calidad también se debe cumplir a nivel de implementación manteniendo una arquitectura modular y mantenible y un código legible. | Se deben haber cumplido el 100% de requisitos y objetivos comprobándolos al final del proyecto uno a uno y se debe poder ampliar el proyecto sin necesidad de recodificar lo ya implementado. Respecto a las interfaces de usuario, deberán aprobar los test de usabilidad realizados a los usuarios. | B. Cristina Pelayo García-Bustelo |
| **Publicaciones:** | | |
| El proyecto deberá generar publicaciones derivadas de la investigación realizada para llevar a cabo el proyecto. | Se deben haber escrito y enviado un artículo de congreso, revista o revista JCR por módulo. | B. Cristina Pelayo García-Bustelo |

| Resumen de hitos | Fecha |
|---|---|
| Reunión con el director de proyecto para fijar requisitos, objetivos, alcance y acordar la planificación incluyendo entregas y revisiones. | 22-01-2016 |
| Finalización del análisis y del diseño del sistema, entre lo que se incluyen todos los requisitos, riesgos, interfaces de usuario y clases del sistema. | 26-01-2016 |
| DSL textual definido para todos los módulos. | 12-02-2016 |
| Finalización de la primera parte (DSL textual) del primer módulo completo, módulo de actuadores. | 27-02-2016 |
| Finalización de la primera parte (DSL textual) del segundo módulo completo, módulo de sensores. | 09-03-2016 |
| Finalización de la primera parte (DSL textual) del tercer módulo completo, módulo de reglas combinatorias. | 19-03-2016 |
| Finalización del DSL textual | 21-03-2016 |
| Artículo correspondiente al DSL textual enviado | 21-03-2016 |
| DSL gráfico definido para todos los módulos. | 02-04-2016 |
| Finalización de la segunda parte (DSL gráfico) del primer módulo completo, módulo de actuadores. | 12-04-2016 |
| Finalización de la segunda parte (DSL gráfico) del segundo módulo completo, módulo de sensores. | 18-04-2016 |
| Finalización de la segunda parte (DSL gráfico) del tercer módulo completo, módulo de reglas combinatorias. | 23-04-2016 |
| Interfaces de usuario para el editor del DSL gráfico finalizadas. | 03-05-2016 |
| Finalización del DSL gráfico. | 03-05-2016 |
| Artículo correspondiente al DSL gráfico enviado. | 16-05-2016 |
| Proyecto finalizado y listo para presentar ante tribunal. | 31-05-2016 |

### Presupuesto estimado:

El presupuesto de costes estimado contempla los gastos de servidor y dominio donde se despliega la plataforma durante el desarrollo, el salario del desarrollador a 22€/hora incluyendo seguridad social, dietas y traslados, los gastos infraestructurales como la luz y la conexión a internet, la amortización de la máquina del desarrollador que según la legislación vigente debe ser de 4 años y una reserva para imprevistos y reservas.
Todo esto hace un presupuesto de costes estimado de 23.605,00€ como se puede ver más detalladamente en la siguiente tabla.

| Concepto | Recurso | Precio Rec. | Total Concepto |
|---|---|---|---|
| Materiales | | | 3.500,00 € |
| | Gastos de servidor y dispositivos. 3.500€ | 3.500,00 € | |
| Trabajo personal | | | 14.250,00 € |
| | Salario = 22 € x 600 horas | 13.200,00 € | |
| | Dietas y traslados. 300€ x 3.5 meses. | 1.050,00 € | |
| Equipamiento | | | 1.855,00 € |
| | Gastos infraestructurales. 500€ * 3.5 meses | 1.750,00 € | |
| | Máquina aproximado: 1.400€ / 4 años * 3.5 meses | 105,00 € | |
| Otros gastos | | | 4.000,00 € |
| | Imprevistos y reservas | 4.000,00 € | |
| | | Total | 23.605,00 € |

## Autoridad del jefe de proyecto

### Decisiones de personal:

Este proyecto no cuenta con personal para que el jefe de proyecto tenga que tomar decisiones. La única decisión es la elección de director de proyecto anterior a la creación de este documento.

### Gestión del presupuesto y variaciones:

Al tratarse de un proyecto personal y educativo, el jefe del proyecto puede considerar todas las variaciones del presupuesto que estime oportunas pero siempre consultando al director de proyecto para asegurarse de realizar la gestión correcta.

### Decisiones técnicas:

La naturaleza personal de este proyecto hace que todas las decisiones técnicas las haga el jefe de proyecto con total autoridad pero siempre es recomendable pedir la autorización al director del proyecto.

### Resolución de conflictos:

En este proyecto no deberían existir conflictos ya que no hay una plantilla de personal que los cause. El único conflicto que puede surgir es entre el director del proyecto y el jefe del proyecto que entre ambos deberán llegar a algún acuerdo si esto ocurriese.

**Vía de escala de autoridad ante limitaciones:**

En este proyecto no existe ninguna limitación de autoridad pero en caso de que así fuese el siguiente paso tras el jefe del proyecto es el tutor del proyecto.

**Aprobaciones:**

| | |
|---|---|
| Firma del jefe de proyecto | Firma del director del proyecto |
| Daniel Meana Llorián | B. Cristina Pelayo García-Bustelo |
| Nombre del jefe de proyecto | Nombre del director del proyecto |
| 3 de enero de 2016 | 3 de enero de 2016 |
| Fecha | Fecha |

# ANEXO II. Guion de pruebas

# Bilrost

## Objetivo de Bilrost

Bilrost es un sistema compuesto por un lenguaje de dominio específico o DSL llamado Bilrost Specific Language (BSL) que pretende servir de ayuda a la hora de interconectar objetos inteligentes usando para ello redes sociales, dos editores (uno textual y otro gráfico) y las herramientas para generar aplicaciones que interconecten objetos a través de las redes sociales en varios lenguajes.

Estos proyectos comunicarán sensores y actuadores de diferentes dispositivos, o de un único dispositivo, publicando y leyendo mensajes en redes sociales. Además, los usuarios tendrán la capacidad de invocar acciones de actuadores de forma manual mediante el uso de mensajes en las redes sociales utilizando una serie de palabras clave.

En este documento se presenta la sintaxis BSL y se proponen una serie de tareas para evaluar el mismo, usando tanto el editor textual como el editor gráfico. Este último transforma el modelo definido por el usuario en un documento con sintaxis BSL como el que se crearía con el editor textual.

Cada fichero escrito en BSL representa la definición de un único dispositivo y se describen sus actuadores, sensores y reglas de funcionamiento.

## Definición de un dispositivo

Un dispositivo se define indicando el lenguaje de programación que se desea usar en la aplicación, que palabras debe buscar para filtrar las búsquedas en la red social, la lista de redes sociales a las que se conecta, la lista de actuadores que se quieren utilizar, la lista de los sensores que publicarán sus datos en las redes sociales para que puedan ser utilizados por otros dispositivos y las reglas de funcionamiento que interconectan sensores y actuadores mediante condiciones. Los sensores y actuadores usados en las reglas pueden estar situados en el mismo dispositivo o en diferentes dispositivos como veremos más adelante. Es obligatorio que al menos uno de estos tres bloques (actuadores, sensores o reglas) esté presente.

```
DEVICE IN PYTHON | JAVA | ANDROID
      FILTER BY …
      SOCIAL NETWORKS …
      ACTUATORS …
      SENSORS …
      RULES …
```

Los posibles valores que puede tomar el lenguaje de programación son: PYTHON, JAVA o ANDROID. Pero solo puede tomar **UN** único valor.

## Filtros del dispositivo

Los dispositivos cuentan con una serie de filtros que usarán a modo de identificador en las redes social. De esta manera solo reaccionarán ante mensajes que contengan esos filtros y todos los mensajes que publiquen contendrán esos filtros. El número de filtros máximo permitido no está definido pero hay que tener en cuenta el límite de caracteres que puede tener la red social. Sin embargo si hay un mínimo y al menos debe tener **UN** filtro definido. Estos deben ir separados por comas y entre comillas (da igual dobles o simples).

```
FILTER BY 'filtro1', 'filtro2', …
```

## Redes sociales

Un dispositivo se puede conectar a varias redes sociales para escuchar o publicar mensajes en todas ellas aunque en el prototipo desarrollado solo se ha implementado la conexión con Twitter. Por esto la definición del dispositivo está pensada en varias redes sociales pero solo la sintaxis de Twitter ha sido creada.

```
SOCIAL NETWORKS
      CONNECT TO TWITTER | OTRAS REDES SOCIALES
            …
```

Tras indicar la red social a la que se conectará el dispositivo, se indicarán los parámetros necesarios para establecer la conexión como los tokens de sesión. Estos serán dependientes de cada red social.

- **Twitter**

Para conectarse a Twitter son necesarias las claves que proporciona la red social a través del panel de desarrollo. Estas claves las puede proporcionar el prototipo mediante un login en la red social. Sin embargo, para esta evaluación no se usará ninguno ya que el prototipo está preparado para autocompletar los valores con valores por defecto en caso de no haber iniciado sesión. La sintaxis es la siguiente:

```
CONNECT TO TWITTER
      TOKEN 'tokenvalue'
      SECRET 'secretvalue'
      ALLOW 'usuario1', 'usuario2', …
```

El atributo ALLOW es opcional y contiene una lista de nombres de usuarios de Twitter (u objetos inteligentes con cuentas propias), entre comillas y separados por comas, a los que se les daría permisos para controlar los actuadores del dispositivo desde la red social. Los nombres **NO** deben contener el símbolo @. Los atributos TOKEN y SECRET son las claves mencionadas en el párrafo anterior.

# Actuadores

Los dispositivos pueden tener varios actuadores. Para ello se abre el bloque de actuadores con la palabra reservada ACTUATORS y se inicia la definición de cada actuador con la palabra DEFINE ACTUATOR. Todos los actuadores deben tener un nombre que se usará como palabra clave a la hora de llamarlos a través de las redes sociales y una serie de acciones que puede realizar. Las acciones se representan con un nombre que se corresponderá con el nombre del método que se llamará en los proyectos generados. Además, los actuadores también disponen de un campo de filtros completamente opcional que permitirá agrupar varios actuadores bajo una misma invocación. De esta manera se podrá realizar la misma llamada en varios actuadores que cuenten con la misma acción y los mismos filtros sin usar el nombre de cada uno.

La sintaxis de los actuadores es la siguiente:

```
ACTUATORS
      DEFINE ACTUATOR 'nombre'
      FILTER BY 'filtro1', 'filtro2',
      ACTIONS 'acción1', 'acción2', …
```

Como ya se ha mencionado, se puede invocar una acción común a todos los actuadores que compartan algún filtro omitiendo el nombre del actuador en el mensaje de la red social y añadiendo los filtros adecuados ya que en ese caso se busca cualquier actuador que disponga de la acción requerida.

Si la acción necesitara recibir parámetros se publicarían en Twitter en un único *String* entre comillas tras el nombre de la acción.

- **Invocación de acciones a través de Twitter**

Para invocar acciones de los actuadores se debe twittear tweets como el siguiente:

**#filtroDispositivo #filtroDispositivo #nombreActuador #filtroActuador #filtroActuador nombreAcción “parametros”**

**filtroDispositivo** se corresponden con un filtro del dispositivo, **filtroActuador** se corresponde con un filtro del actuador si lo hubiera, **nombreActuador** es el nombre del actuador que debe reaccionar, **acción** es el nombre

de la acción que se debe invocar y **parámetros** es un *String* donde se pueden pasar parámetros a la acción (la forma de separar los parámetros dentro de las comillas es a gusto del usuario).

Tanto **nombreActuador** como **parámetros** son opcionales. Si no se usa **nombreActuador** se invocará la acción en todos los actuadores que dispongan de ella y respondan ante los filtros introducidos y si no se usan **parámetros** la acción no recibirá parámetros.

## Sensores

Los dispositivos pueden tener varios sensores. Para ello se abre el bloque de sensores con la palabra reservada SENSORS y se inicia la definición de cada sensor con las palabras DEFINE SENSOR. Todos los sensores deben tener un nombre que se usará como palabra clave a la hora de llamarlos a través de las redes sociales y un modo de funcionamiento que puede ser manual o automático. El modo manual implica que el usuario debe invocar a la función encargada de publicar en las redes sociales de forma manual según le convenga como en el caso de un sensor de presencia que debe invocarla cuando se detecte algo. El modo automático implica que el dispositivo publicará el estado del sensor cada cierta frecuencia. Además, los sensores también disponen de un campo de filtros completamente opcional que permitirá establecer reglas para varios sensores.

La frecuencia de publicación del sensor se compone de la palabra MODE seguida del tipo de publicación (MANUAL o AUTO). En caso de ser AUTO se debe indicar la frecuencia con un número real y una unidad de tiempo entre SECONDS, MINUTES u HOURS. La s final es opcional debido a la singularidad del número 1.

La sintaxis de los sensores es la siguiente:

```
SENSORS
      DEFINE SENSOR 'nombre'
            FILTER BY 'filtro1', 'filtro2', …
            MODE MANUAL
            MODE AUTO número SECONDS | MINUTES | HOURS
```

## Reglas

Los dispositivos pueden tener varias reglas. Para ello se abre el bloque de reglas con la palabra reservada RULES y se inicia la definición de cada sensor con las palabras DEFINE RULE.

Las reglas son más complejas que el resto del lenguaje. Están compuestas por 3 subsecciones: la fuente de datos o sensores, la condición y las ejecuciones que deben realizar si se cumple la condición.

Para comenzar la definición de una regla se debe indicar para que tipo de sensores entre las siguientes opciones:

- Un sensor del dispositivo

```
DEFINE RULE TO SENSOR 'nombre'
```

- Todos los sensores del dispositivo que respondan ante ciertos filtros

```
DEFINE RULE TO ALL SENSORS
   FILTER BY 'filtro1', 'filtro2', …
```

- Un sensor de un dispositivo externo (se deben indicar los filtros para llegar al dispositivo).

```
DEFINE RULE TO EXTERNAL SENSOR 'nombre'
   FILTER BY 'filtro1', 'filtro2', …
```

- Todos los sensores externos que respondan ante ciertos filtros.

```
DEFINE RULE TO ALL EXTERNAL SENSORS
      FILTER BY 'filtro1', 'filtro2', …
```

A continuación se debe definir la condición que los datos de estos sensores deben cumplir. Para ello se usa la siguiente sintaxis donde se debe escoger entre un tipo de condición.

```
IF VALUE IS LESS THAN |
      LESS THAN OR EQUAL TO |
      EQUAL TO | NOT EQUAL TO |
      GREATER THAN OR EQUAL TO |
      GREATER THAN número
```

Por último se deben indicar las ejecuciones que se deben realizar si se cumple la condición. Para ello se debe indicar la acción a invocar y el actuador o actuadores que la realizarán. Una regla puede tener cualquier número de ejecuciones pero como mínimo una.

Al igual que en el caso de los sensores, la sintaxis tiene 4 opciones dependiendo del tipo de ejecución.

- Un actuador del dispositivo

```
EXECUTE ACTION 'nombreAcción' IN ACTUATOR 'nombreActuador'
```

- Todos los actuadores del dispositivo que respondan ante ciertos filtros

```
EXECUTE ACTION 'nombreAcción'
   IN ALL ACTUATORS
   FILTER BY 'filtro1', 'filtro2', …
```

- Un actuador de un dispositivo externo (se deben indicar los filtros para llegar al dispositivo).

```
EXECUTE ACTION 'nombreAcción'
   IN EXTERNAL ACTUATOR 'nombreActuador'
   FILTER BY 'filtro1', 'filtro2', …
```

- Todos los actuadores externos que respondan ante ciertos filtros.

```
EXECUTE ACTION 'nombreAcción'
      IN ALL EXTERNAL ACTUATORS
      FILTER BY 'filtro1', 'filtro2', …
```

Finalmente, el aspecto que puede tener una regla sería el siguiente

```
RULES
      DEFINE RULE TO ALL SENSORS
            FILTER BY 'filtro1', 'filtro2', …
            IF VALUE IS GREATER THAN -1
            EXECUTE ACTION 'acción1' IN ACTUATOR 'actuador1'
            EXECUTE ACTION 'acción2' IN ACTUATOR 'actuador2'
      …
```

## Ejemplo de aplicación

A continuación se muestra la definición de dos dispositivos interconectados para ilustrar el uso del lenguaje.

Se definirá un dispositivo que usa Python como lenguaje de programación, como puede ser una Raspberry Pi que tiene un sensor de luz y un led de notificaciones y otro dispositivo Android que tenga un acelerómetro que permite reconocer sacudidas y un flash.

Se quiere que cuando el sensor de luz de un dispositivo detecte poca luz se encienda el flash del otro a modo de linterna de emergencia y cuando se detecten sacudidas en un dispositivo se encienda el led de notificaciones en el otro a modo de aviso de emergencia.

Las definiciones serían las siguientes:

```
DEVICE IN python
      FILTER BY 'bilrost', 'rpi'
      SOCIAL NETWORKS
            CONNECT TO twitter
                  TOKEN 'token'
                  SECRET 'secret'
      ACTUATORS
            DEFINE ACTUATOR 'led'
                  ACTIONS 'on'
      SENSORS
            DEFINE SENSOR 'light'
                  MODE MANUAL
      RULES
            DEFINE RULE TO SENSOR 'light'
                  IF VALUE IS LESS THAN 30
                  EXECUTE ACTION 'on'
                        IN EXTERNAL ACTUATOR 'flash'
                        FILTER BY 'bilrost', 'smartphone'

DEVICE IN android
      FILTER BY 'bilrost', 'smartphone'
      SOCIAL NETWORKS
            CONNECT TO twitter
                  TOKEN 'token'
                  SECRET 'secret'
      ACTUATORS
            DEFINE ACTUATOR 'flash'
                  ACTIONS 'on'
      SENSORS
            DEFINE SENSOR 'shake'
                  MODE MANUAL
      RULES
            DEFINE RULE TO SENSOR 'shake'
                  IF VALUE IS EQUAL TO 1
                  EXECUTE ACTION 'on'
                        IN EXTERNAL ACTUATOR 'led'
                        FILTER BY 'bilrost', 'rpi'
```

## Notas

- El lenguaje no es sensible a mayúsculas o minúsculas. Son válidas ambas notaciones.
- Las palabras entrecomilladas pueden estar entre comillas ‘simples’ o “dobles”.
- No debe haber espacios en blanco en los textos entrecomillados.
- Los comentarios de línea siguen la sintaxis de Python: *# Esto es un comentario*.
- El lenguaje no requiere ni el uso de saltos de línea ni de tabuladores pero se aconseja para facilitar su legibilidad.

## Acciones a realizar

Debes crear la definición de un dispositivo que cumpla los siguientes requisitos:

1. El objetivo es desarrollar una aplicación para que sea ejecutada en la Raspberry Pi utilizando el lenguaje de programación Python. El objetivo de la aplicación será regular la temperatura de una habitación mediante el control de un ventilador y el sistema de calefacción y contando además con un sensor de temperatura. La aplicación hará que cuando la temperatura sea alta la calefacción se apague y se encienda el ventilador mientras que cuando la temperatura sea baja se apague el ventilador y se encienda la calefacción.
2. Los mensajes que envíe y reciba la aplicación deben contener las palabras claves ***bilrost***, ***evaluation*** y ***test***.
3. La red social usada será Twitter. Los valores de TOKEN y SECRET se autocompletan en el editor textual al escribir las palabras TOKEN y SECRET seguidas de un espacio y salto de línea, y ya vienen predefinidos en el editor gráfico.
4. Se debe definir **UN** dispositivo que controla un ventilador en una sala y es capaz de medir la temperatura. Además existe otro dispositivo ya definido y en funcionamiento que controla la calefacción de la sala con un actuador llamado ***heating,*** cuyas acciones son ***on*** y ***off,*** y cuyos filtros son ***bilrost, evaluation*** y ***external.*** El dispositivo definido debe ser capaz de **encender el ventilador** y **apagar la calefacción** cuando la temperatura **exceda** los 25ºC. También debe ser capaz de **encender la calefacción** y **apagar el ventilador** cuando la temperatura sea **inferior** a 16ºC. La temperatura se debe comprobar cada 30 segundos.
5. El dispositivo se deberá definir tanto con el editor textual como con el editor gráfico.

Para finalizar debe rellenar el cuestionario que se pondrá a su disposición.

# ANEXO III. Bilrost: Domain-Specific Language to define actions for the Internet of Things actuators, triggered by Twitter users' posts

# Bilrost: Domain-Specific Language to define actions for the Internet of Things actuators, triggered by Twitter users' posts

Daniel Meana-Llorián[a,*], Cristian González García[a], B. Cristina Pelayo G-Bustelo[a], Juan Manuel Cueva Lovelle[a]

[a] *University of Oviedo, Department of Computer Science, Sciences Building, C/Calvo Sotelo s/n 33007, Oviedo, Asturias, Spain*

**Abstract**

In recent years many researches about social networks for the Internet of Things have been appeared. Their purpose is the interconnection of objects in a similar way to humans social networks. Furthermore, another perspective is the interconnection between objects and humans but with a small limitation, objects do not take into consideration people actions to take decisions. Moreover, all social networks used in previous researches were developed for the research or were current social networks where objects only share their status. In this article we have proposed the use of Twitter to interconnect objects with humans so that the objects could do actions based on humans' posts. For this, we propose a Domain-Specific Language that facilitates the definition of the actions that objects have to do when people publish specific content on social networks.

*Keywords:* Internet of Things, Smart Objects, Model-Driven Engineering, Domain-Specific Language, Twitter

*Corresponding author

*Email addresses:* danielmeanallorian@gmail.com (Daniel Meana-Llorián), gonzalezgarciacristian@hotmail.com (Cristian González García), crispelayo@uniovi.es (B. Cristina Pelayo G-Bustelo), cueva@uniovi.es (Juan Manuel Cueva Lovelle)

## 1. Introduction

The Internet of Things (IoT) is a term that has gained popularity in recent years among common people due to the wishes for being connected to whole things around us. We want to be connected to objects located at home like the fridge, the oven, and so on. A examples of our expectations can be the fridge could alert us when a product is running out, or the oven could be turned on when we are arriving home. The possibilities of IoT are not only the Smart Homes [1, 2, 3] but also the Smart Earth [4], the Smart Cities [4, 5] and any kind of distributed intelligence around heterogeneous and ubiquitous objects. Moreover, the rise of mobile devices [6] like smartphones, tablets, wearables or any other devices connected to internet like sensors, smart tags, and so on, has contributed to the popularity of IoT.

Despite of the popularity of the IoT, it is not so present in our lives as we expected. What are the causes? The answer could be the complexity of managing our actuators: what actions to do, when the actuator has to work or how it must do it; or understanding and interpreting data from sensors.

Our goal is to facilitate the creation of applications using a Domain-Specific Language (DSL) created with Model-Driven Engineering (MDE) that actuators could run. We have designed a language to define the rules and properties required to establish a communication between actuators and humans through humans social networks and to configure actuators and their actions. The DSL includes all of this information. It is also important to mention that Bilrost does not create end-use applications, Bilrost creates projects in which the actions are defined and the interconnection to social networks is implemented. However, the user has to implement the actions to complete the application.

Using social networks to make the interconnection has several advantages over commons solutions like an architecture client-server through HTTP. This novel approach avoids to have a specific application to communicate and, moreover, this solution is more intuitive for people who use social networks in their day-to-day lives.

The remainder of the paper is organised as follows. In Section 2, we introduce involved topics in this research like the IoT, Smart Objects, social networks, MDE, DSL and the works related with this research. Section 3 shows the Bilrost platform, what it is, how its architecture is and how its DSL, called Bilrost-Specific Language (BSL), is. In Section 4 we cover the evaluation and discussion of the data obtained from analyzing the process of creating projects with Bilrost. Section 5 contains the conclusions of this paper and finally, in Section 6 we describe the possible future work that can be done from here.

## 2. State of the art

Smart Objects is a concept very important in the context of the Internet of Things. Smart Objects are interconnected and exchange messages to achieve a common target. This objects can be separated by thousand of kilometres. Maybe behind the enemy lines in a battlefield or even in places where they cannot be reached after being set [7, 8]. Nevertheless, they can communicate with each other and they are capable of taking decisions based upon the messages they receive. However, there are a few problems to establish these communication between heterogeneous objects. First, the use of the same standard and protocol in order to interact with objects. Second, everybody do not know program to create programs that intercommunicate objects or to interact directly with the objects.

Most of researches about interaction among objects through the Internet are based on Service Oriented Architecture (SOA) like REST [9, 10, 11]. However, in this research, we want to use online social networks to communicate humans and Smart Objects, specially their actuators. Moreover, we defined a DSL to facilitate the managing of actuators and their actions.

Thus, in the following lines we talk about the concepts which are involved in this research: IoT, Smart Objects, online social networks, MDE, and DSL.

### *2.1. The Internet of Things*

During the last years, one of the most important themes in researches and business is the interconnection between heterogeneous and ubiquitous objects between themselves. This technology is better known as the Internet of Things [12, 13, 14, 15].

Furthermore, it is one of the most important technologies for the next years according to the ONU at Tunisia in 2005 [13] or the United States National Intelligence Agency [16] Councils report. In the first one, the United Nations has predicted a new era with more traffic generated by objects than Internet traffic. In the second one it has considered the IoT between one of the six technologies with more interest to the United States from here to 2025.

However, it does not have a standard or a unique way to do this. We can use different platform through the Internet to interconnect the objects as we can see in [12, 17]. On the other hand, we have different standards to communicate the objects like Near Field Communications (NFC), Radio Frequency Identification (RFID), or Bluetooth. For that, it is necessary to facilitate the development in the IoT.

### *2.2. Smart Objects*

Smart Objects, also known as Intelligent Products, are physical elements identifiable during their useful life and which can interact with the environment and/or other objects, and have automatic or semi-automatic behaviour depending on the data that they process or receive, or react according to the interactions with other Smart Objects [13, 18]. Some examples are Smart TVs, smartphones, tablets, and some cars.

Smart Object can be classified in three dimensions [19, 20]: **Level of Intelligence**, **Location of Intelligence**, and **Aggregation level of Intelligence**. The first one indicates the objects intelligence. The second one describes if the intelligence belongs to the object or to the network. The last dimension indicates if the intelligence is in the element, for example, when the object is

composed by various elements and each one has their own intelligence, or in the container, when it is the contrary case.

Our proposal offers Smart Objects with an intelligence in the container. The other two characteristics depend on the implementation that the user made in the objects. Then, it could have a low, medium or high Level of Intelligence, and it could have intelligence in the object, because the intelligence through the network is obligatory in our proposal.

### *2.3. Online Social networks*

Online Social Networks (OSN) are good resources to make applications that could be integrated in our lives. OSN provide many services we can use to create applications like identity and authorisation services, APIs to read or write in timelines, receive updates, receive and send private messages and, so on. OSN is a basic piece of the Web 2.0 and the convergence of the real world with OSNs allows the development of new applications which interconnect things and humans [21].

There are a lot of OSNs that could be used for this research but we chose Twitter due to its philosophy of short public messages and the common use of keywords, which are called hashtags, in the messages. Moreover, its API provides the functionality required for this research although there are few limitations.

There are a wide range of works that used Twitter to research. Twitter is a very important information source and it can be used for different proposals, from inferring user attributes [22] or predicting flu trends [23] to sharing sensors' status [24, 25].

The Social Internet of Things (SIoT) is an approach similar to Online Social Networks but focused in objects instead of humans. This social objects are a new generation of objects that can interact with other objects without intervention of their owners but with their permission. They can discover other objects, services, and useful information and they can share their services to the rest of objects in the network [26]. Based on this principies, in [26] they built their

own social network for Smart Objects. The disadvantages of this SIoT are the dependence on specific social network which is not used for other propose and the interaction of this SIoT is between objects whereas we propose the intercommunication between humans and objects.

Furthermore, scientists of Ericsson [26] have observed that people are able to familiarise themselves with IoT technologies and with the interaction with the Smart Objects if there is a analogy to their habits from OSNs like Facebook, Twitter or any other social network.

The combination between ONS and SIoT will bring new interesting applications and possibilities for IoT.

### *2.4. Model-Driven Engineering*

Model-Driven Engineering (MDE) appeared to solve software development problems that we have had since 1960s as Dijkstra described in [27] until now as we can see in [28]. One way to solve this is using MDE to automate or semi-automate several processes to facilitate the creation of repetitive processes [29]. Then, applying MDE, developers can abstract the problem and automate some parts to facilitate the production of similar problems. With this, developers could reduce the complexity of the application design and implementation [30].

We have used MDE to facilitate the generation of the interconnection between objects through Twitter. Thus, users will only need implement the objects functionality that they want because if they use BSL the interconnection will be created for them.

### *2.5. Domain-Specific Language*

A Domain-Specific Language (DSL) is a specific language to resolve a particular problem in one context. Normally, they are declarative and makes calls for sub-processes to create the necessary transformations or source code. However, it must apply MDE to abstract the problem and do the DSL to solve that problem.

The great advantage of use a DSL is the great expression power [31, 32]. This offers us an increment of the productivity, lesser chances of errors, portability, easy maintenance [33], and reutilisation [31, 33].

On the contrary, DSLs have worse efficiency than General-Purpose Languages (GPL), and they need knowledge about MDE to abstract the problem to create the DSL [31, 33].

We have applied MDE to create a DSL called BSL. BSL allows to create the interconnection between heterogeneous and ubiquitous objects and humans through Twitter. Exactly, BSL is a textual DSL. With BSL we have searched to facilitate the interconnection of humans with objects through Twitter and the logic that they objects need to obtain the necessary data for Twitter to do their actions.

### *2.6. Related work*

This research introduces a novel way of establishing a communication between humans and objects. We propose the use of actual humans social networks instead of use common web services and we use them to send messages to the objects so that they actuate taking in consideration that messages instead of using socials networks to retrieve information from the objects.

Taking into account other researches related with this topic is necessary although there are not very similar researches.

There are many researches that address the communication among objects [34, 26] and between humans and objects [25, 35, 36, 37].

A related work is the Midgar IoT platform [12, 17]. Midgar allows to interconnect heterogeneous and ubiquitous objects between themselves. It allows the interconnection of them using a graphic DSL to facilitate the creation of this interconnection for people without development knowledge. When users have defined their application, Midgar generates a demon to interconnect the objects according to users definition. Moreover, Midgar will include a graphic DSL to create Smart Objects. However, Midgar has to use their own server or another

private server and only allows interconnect objects whereas Bilrost do not need any server because of its interconnection with Twitter.

An approach similar to our approach is Social Access Controller (SAC) [25]. It uses social networks to share Smart Objects and it allows managing them. One advantage of SAC is that it supports not only handle Smart Objects remotely but also share their status. However, the use of the social networks is very different. In [25], social networks are used to know the owners's friends of the devices and then it allows them to access to Smart Objects through a REST architecture. However, with our approach, the access to Smart Objects is through Twitter and the users that can handle the actuators do not need to be friends of anyone, they only need to appear in the program written with our DSL.

Using **instant messages** to interconnect objects with other objects or humans is addressed in [35, 36, 37]. This approach has any disadvantages like the dependency of specific applications which are exclusive for establish the communication with the objects whereas the use of Twitter allows using the common Twitter application available in many smartphones and also, it allows using the web application without having the requirement of using a specific technology.

To sum up, Bilrost is a novel approach that allows establishing a communication between humans and objects without any requirements by the humans side. Humans only need access to Twitter. Moreover, this communication is based on the common use of the social networks, publishing in your timeline.

## 3. Bilrost platform

Bilrost is an Internet of Things platform develop to investigate the interconnection between objects and humans through humans social networks. In this paper we propose a platform to generate applications that interconnect humans and objects in an easy way. The main aim is that everybody could handle objects remotely without a specific knowledge. In this section, we will describe Bilrost platform making mention of its architecture and the DSL used to create

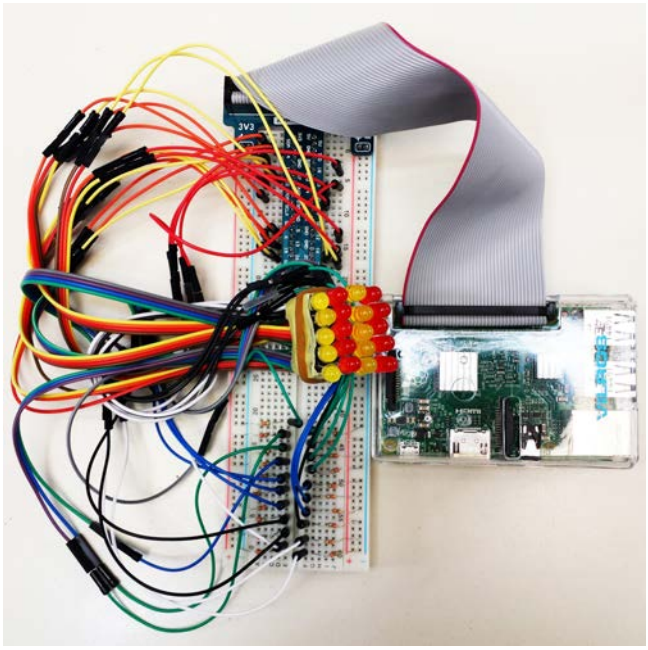


Figure 1: 4x5 red and yellow leds matrix connected to a Raspberry Pi 2.

the applications. Furthermore, we will use a example to describe how Bilrost works. We will try to handle a matrix of 4x5 red and yellow leds connected to a Raspberry Pi 2, Figure 1. We will be able to turn on and turn off the leds of a specific colour or all together and show a message which scrolls through the matrix.

### *3.1. Generation of applications for managing heterogeneous objects through Online Social Networks*

With Bilrost, everybody can generate applications that allow them to handle their objects' actuators through ONSs. In the prototype developed, the social networks used is Twitter but the prototype is extensible and adding new OSNs would be a minor change.

The work cycle consists in three steps:

1. Creating a code using BSL which defines the actuators' properties and social networks data.

2. Using Bilrost platform with the code created in the previous step to generate a program.
3. Completing the program implementing the actions to do when an action is triggered by the social network.

Bilrost does not implement the code that the actuators have to run when an action is called. The Bilrost's aim is the interconnection between objects and humans through social networks so it generates a skeleton so that the user could fill as he want with the code of the actions.

### *3.2. Architecture*

The system's architecture can be divided in three layers: **Bilrost-Specific Language code definition**, **Project generation**, **Actions implementation** and **Objects**. Each layer depends on the others. The first layer, **Bilrost-Specific Language code definition**, contains the parser of BSL and the result of the parser is sent to the next layer. The layer **Project generation** takes the result of the first layer and interprets it to generate a project which contains the code to establish the connection with the social network and the skeleton of the actions. The third layer, **Actions implementation**, is the involvement of the users who want to define their actuators' actions. When the users complete the actions implementation, the program will be finished and it will be able to be integrated into devices like Android, Raspberry Pi or other devices supported by BSL.

#### *3.2.1. BSL parser*

The input of the BSL parser is a file that contains the code written using the DSL that we created. This code contains the data that the program need to work which is the model defined by the user and the data that social networks require to establish the connection. In the Section 3.3 we will talk about the BSL with more details.

The parser generates a tree that is sent to the project generator and contains all required data and extracted from the model written with BSL.

Taking into account the example, 4x5 red and yellow leds matrix, we want to develop a program that turns on and turns off the leds of each colour and shows a message scrolling through the matrix. For that propose we have to define the BSL shows in Code 1.

```
DEFINE DEVICE IN PYTHON
  FILTER BY "bilrost", "uniovi"
  SOCIAL NETWORKS
    CONNECT TO TWITTER
      TOKEN "token value"
      SECRET "secret value"
      USERS "dani_meana", "bilrost_bridge"
  ACTUATORS
    DEFINE "red"
      LOCATION "rpi"
      ACTIONS "on", "off"
    DEFINE "yellow"
      LOCATION "rpi"
      ACTIONS "on", "off"
    DEFINE "screen"
      LOCATION "rpi"
      ACTIONS "show"
```

Code 1: BSL define to the example.

The output generated from the previous BSL is a JSON file that contains all required data to generate a project.

*3.2.2. Project generator*

The project generator waits for a JSON file with all required data to generate an application connected to a social network. In order to generate the application, the generator chooses a template that fits to the input data and fills it with the data of the input tree.

The prototype developed can generate projects for Java, Android and Python connected to Twitter as humans social network.

The user obtains a skeleton and he has to fill the code of each action how he wills the action works.

In our example, we generate a project for Python as we saw in the BSL defined before. We have three actuators, the screen, the yellow leds and the red leds. Next code shown in Code 2 the content of one actuator after the project generation and before the implementation.

```
class RedActuator(ActuatorBase):
    def __init__(self):
        super().__init__('red')
    def on_action(self, params):
        # TODO Fill as you want
        pass
    def off_action(self, params):
        # TODO Fill as you want
        pass
```

Code 2: Actuator code before the implementation.

The methods *on* and *off* would be called when a user mentions the actuator and the action in Twitter how it is shown in Section 3.2.3.

The interconnection to Twitter and the processing of the data is already implemented so the user only has to focus in the concrete implementation of each action.

### *3.2.3. Implementation and working*

*Implementation.* After the project generation, the user has to fill the skeleton with the implementation he wants for each action. In this article we are focused in the interconnection between humans and objects instead of the actions implementation so our aim is to provide a skeleton to fill as the user want but already connected to Twitter. The code shown in Code 3 is the skeleton shown

in the Section 3.2.2 but it is already filled.

```
import RPi.GPIO as GPIO
class RedActuator(ActuatorBase):
    def __init__(self):
        super().__init__('red')
        self.port = 4
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(self.port, GPIO.OUT)
    def on_action(self, params):
        GPIO.output(self.port, True)
    def off_action(self, params):
        GPIO.output(self.port, False)
```

Code 3: Example of an actuator implementation.

*Working.* In order to execute the actions through Twitter we should *tweet* a message with the required *hashtags*, action's name and its params if it requires them. The required *hashtags* are the filters, the actuator's location and the actuator's name. The actuator's name is not mandatory but the rest of them are. If the name was not specified, all actuators which are located in the location would execute the action specified.

In the following lines there are examples of *tweets* that handle the actuators defines in our example.

- `#bilrost #rpi #display show "hello world"`

  It invokes the action named *show* of the actuator named *display* and located in *rpi*. Moreover, it sends the parameter *hello world*to the action. Parameters must be between quotes and the user has to implement how to parse them.

- `#bilrost #rpi #red on`

  It invokes the action named *on* of the actuator named *red* and located in *rpi*.

- `#bilrost #rpi #yellow on`

  It invokes the action named *on* of the actuator named *yellow* and located in *rpi*.

- `#bilrost #rpi off`

  It invokes the action named *off* of all actuators located in *rpi*.

### *3.3. Bilrost-Specific Language*

We designed a DSL for Bilrost called Bilrost-Specific Language (BSL). BSL allows defining the application specifications that Bilrost will be able to generate. In Code 4 we will show the BSL's context-free grammar written in *Backus - Naur Form* (BNF) and we will explain the syntax using the example showed in Section 3.2.1.

BSL has similar characteristics to SQL. It does not distinguish between under-case and upper-case and it is not as strict as Python. We can write all code in a single line or in a several lines, mix upper-case and under-case and write the specifications in any order.

```
<device>                ::= DEVICE IN <platform> <properties> END
<platform>              ::= PYTHON
                            | JAVA
                            | ANDROID
<properties>            ::= <property>
                            | <properties> <property>
<property>              ::= <filter>
                            | <social-networks>
                            | <actuators>
<filter>                ::= FILTER BY <filters>
<filters>               ::= WORD
                            | WORD COMMA <filters>
<social-networks>       ::= SOCIAL NETWORKS <social-networks-list>
<social-networks-list>  ::= <social-network>
```

```
                            | social-networks-list social-network
<social-network>        ::= CONNECT TO TWITTER <twitter-properties>
<twitter-properties>    :=: <token> <secret> <users>
                            | <secret> <users> <token>
                            | <users> <token> <secret>
                            | <secret> <token> <users>
                            | <token> <users> <secret>
                            | <users> <secret> <token>
                            | <secret> <token>
                            | <token> <secret>
<token>                 ::= TOKEN WORD
<secret>                ::= SECRET WORD
<users>                 ::= ALLOW <users-list>
<users-list>            ::= WORD
                            | WORD COMMA <users-list>
<actuators>             ::= ACTUATORS <actuators-list>
<actuators-list>        ::= <actuator>
                            | <actuators-list> <actuator>
<actuator>              ::= DEFINE WORD <location> ACTIONS <actions>
                            | DEFINE WORD ACTIONS <actions> <location>
<location>              ::= LOCATION WORD
<actions>               ::= WORD
                            | <actions> COMMA WORD
```

Code 4: Context-free grammar in BNF.

Program written with BSL is the definition of a device which has actuators so any program define only one device. To write a programa with BSL we must start defining the project language that we want to generate and after that we must write the properties of the device like filters, social-networks to connect and their properties and the actuators.

The program that define the device of our example shows in Section 3.2.1

and its explanation is listed below.

1. `DEVICE IN PYTHON`

   It starts the device definition and defines the resulting project language. All programs must start with this line. The device's properties are the filters, the social networks and the actuators. The properties can be defined in any order.

2. `FILTER BY "bilrost", "uniovi"`

   It defines the device's filters. The device only respond if the messages of the social networks contains all filters. One filter is required but there can be more than one separated by commas.

3. `SOCIAL NETWORKS`

   It starts the social networks definition that will be used to gather the messages to handle the device.

4. `CONNECT TO TWITTER`

   It starts the properties block of a social network. In this case the social network is Twitter so the content of the block is exclusive for Twitter. There can be more than one social network but at least one. The social network's properties can be defined in any order.

   (a) `TOKEN "token value"`

   It is the token required by the Twitter API. We will not talk about how we can obtain this token.

   (b) `SECRET "secret value"`

   It is another key for Twitter API. As we said before, we will not talk about how we can obtain the API values.

   (c) `USERS "dani_meana", "bilrost_bridge"`

   It defines usernames of the users who can handle the device through Twitter. This property is not mandatory, it is optional, so if it was not defined, any Twitter user could handle the device.

5. `ACTUATORS`

   It starts the definition of the device's actuators.

6. `DEFINE "red"`

   It starts the definition of an actuator and defines its name.

   (a) `LOCATION "rpi"`

   It is another filter but in the actuator level that defines the actuator's location. It is required and only accepts one value.

   (b) `ACTIONS "on", "off"`

   It defines the actuator's actions. One action is required but there can be more than one separated by commas.

In addition to the above, there are also comments in BSL. The comments' syntax is the same as Python's comments syntax. It starts with a hash sign (#) and ends with a new line.

### *3.4. Used software and hardware*

In order to develop this research work the use of different types of software and hardware components were required. The software components are the following:

- The parser's language is Python 3.4 and it uses the library Ply 3.6[1].
- The resulting projects were written for Python 3.4, Java 7 and Android 5.0 although there is backward compatibility with Python 3.x and Android 4.x.
- The resulting Python project uses the library Twython 3.2.0[2].
- The resulting Java and Android projects use the libraries Gson 2.3.1[3] and Twitter4j 4.0.3[4].

For the evaluation of the proposal the hardware that we used were a Raspberry Pi 2 with a several electronic components connected to its GPIO and a Galaxy S4 with Android 5 Lollipop.

[1] Ply: http://www.dabeaz.com/ply/
[2] Twython: https://twython.readthedocs.org/
[3] Gson: https://code.google.com/p/google-gson/
[4] Twitter4j: http://twitter4j.org

## 4. Evaluation and discussion

In this section we will describe how we made the evaluation process selected and then we will show the obtained results. First, we will show the used evaluation methodology to perform the evaluations and, after that, we will show and discuss the obtained results.

### *4.1. Methodology*

The main aim of this evaluation process is to create an evaluation process that validates if the solution proposed is a step forward in the field of communication between Smart Objects and humans. Our intention is obtain information that allows us to know if the research is useful for different kind of users in order to model easily and quickly connected applications with humans through social networks for Smart Objects. For this proposed we split the process in two phases.

In the first one, we wanted to check whether users of various profiles were able to use BSL to complete an specific task which corresponded to a use case to handle several actuators.

After the end of the previous phase, the users who had taken part did a survey using the Likert scale [38] where they showed their opinion about declarations.

#### *4.1.1. Phase 1*

In this phase, users of several profiles had to complete a task that emulate a real scenario that required two actuators connected to Twitter with several actions. The entire task is in the follow paragraph.

*The assigned task.* The task consisted in define a device which had a fan and a thermostat. The device was a Raspberry Pi and the programming language for the development has to be Python. The device only looked for messages that contained the keywords *bilrost*, *evaluation* and *test*, it had to be connected to Twitter and the unique user who can handle the device was *@bilrost_bridge*.

The fan's actions were to turn on, turn off and set the speed and the thermostat's actions were to turn on, turn off, increase temperature and decrease temperature. The location of both actuator must have been the same. When the definition of the device has been finished, users identified the *tweets* that do the next actions:

- Turning on the fun.
- Turning on the thermostatic.
- Increasing the temperature.
- Setting the fun speed to 2000 rpm.

We chose different users between software developers (SDev) and people who had knowledges of IoT (IoTU). The first group was composed by people who were software developers but they had not knowledges of IoT whereas the second group was composed by users who were interested in IoT regardless of being developers or not. The SDevs evaluate the solution from a technical standpoint and the IoTU from a personal opinion. Moreover, the target of using a DSL is that everybody can develop, therefore, opinions from no developers users were very important.

We reached a total of 20 participants: 7 were SDevs and 13 were IoTUs.

During the evaluation, the users had a document with the objectives and an explanation of the BSL syntax. Before, we explained the task to the user and the process that he had to do.

When the user started the task we started the timer and we stopped it when the task has been correctly completed to calculate the time spent by the user. The time limit to complete the task was four times greater than the time spent by the creator of the prototype.

#### *4.1.2. Phase 2*

After finishing the first phase, the users had to complete a anonymous survey about Bilrost. We used the 5-points Likert Scale [38] because it is the most used

in the design of scales. The giving options were the following: 1 as strongly disagree, 2 as disagree, 3 as neutro, 4 as agree and 5 as strongly agree.

The questionary was created with ten declarations to ask for the user opinion on the creation of connected applications with this tool, its possibilities and its possible impact in the IoT.

The survey is made up of a set of ten declarations that are shown in Table 1.

| Question | Description |
|---|---|
| Q1 | The user understands the functionality of the Domain-Specific Language (DSL) elements and their role in application creation process. |
| Q2 | This DSL allows to interconnect devices and people easily, using a few code lines and spending little time. |
| Q3 | Using a DSL makes it difficult to make mistakes while the user is modelling the applications. |
| Q4 | This solution offers a fast way to developing the indicated task. |
| Q5 | This solution provides assistance to create applications to interconnect objects and people. |
| Q6 | The DSL does not require the user to use complex programming skills, as in traditional application development. |
| Q7 | The DSL includes enough elements and functionality for the user to create a wide range of applications to interconnect objects and people. |
| Q8 | This proposal is a positive contribution to encourage the development of services and applications that provide interconnection between objects and people. |
| Q9 | Internet of Things will be benefited by this solution. |
| Q10 | This DSL could be used to simplify the classic development process of software applications in other areas. |

Table 1: Survey given to the users.

### *4.2. Results*

In the following lines we will present the results obtained with the evaluation process defined before.

#### *4.2.1. Phase 1*

The results of the first phase can be shown in Figure 2. This results show how many time each participant took and the average of all of them which is 648.75 s. Analyzing the data we can see that the fastest participant needed 303 s to finish whereas the slowest participant needed 1504 s. The standard deviation was 252.08 s.

#### *4.2.2. Phase 2*

The responses of each participant are showed in Table 2 in an anonymous way. We do not take into consideration the user's profile.

In Table 3 it is shown the descriptive statistics of all the set and we can see the breakdown of each question: the minimum, the first quartile, the median,

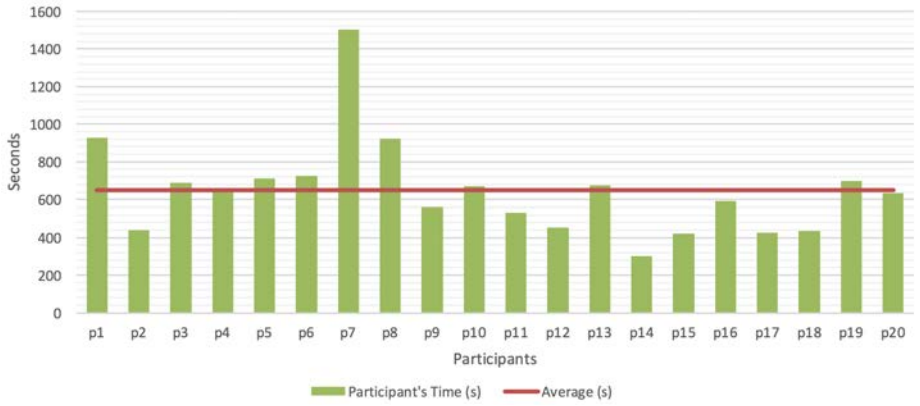



Figure 2: Time to complete the task by the participants.

the third quartile, the maximum, the range (maximum − minimum), the range between quartiles and mode. Figure 3 shows all this data in a Box and Whiskers Plot diagram. By analyzing Table 3 and Figure 3, we can suggest the following interpretations:

- Q2 and Q4 are the declarations with the highest minimum, in this case 4 out of 5. This means that all participants agreed with the declaration, at the very least.
- Q4 is the declaration with the best score. The first quartile is very close to the maximum value so the majority chose the maximum option.
- All questions have a maximum of 5. There is at least one participant that is completely agree by each question.
- Q2, Q4, Q5, Q8 and Q9 have the highest median, 5 out of 5. From this we can deduce that most of the participants agreed these declarations.
- Q2 and Q4 have a range of 1 so all participants had the same opinion on this declarations. However, Q10 is the only question with a range of 4 which is the worst possible range. This means that there are a lot of differences between the answers of each participant.

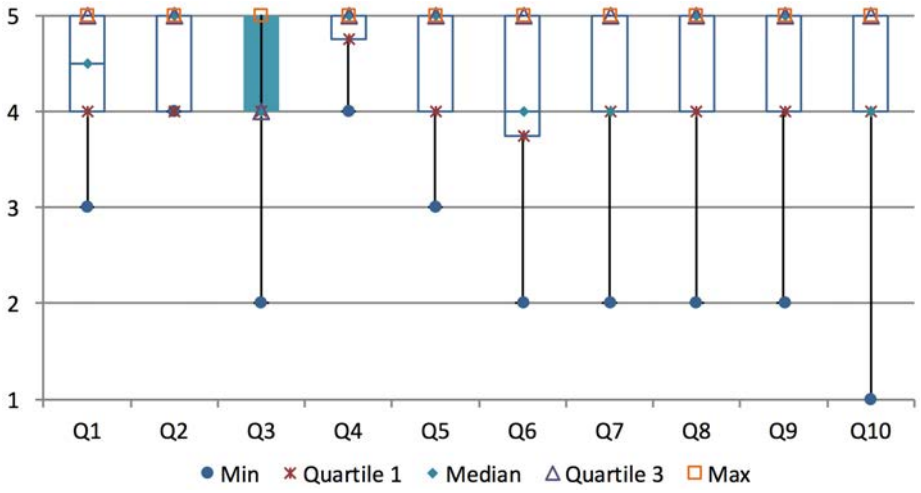



Figure 3: Box and whiskers plot for each question.

- Q6 is the question with the biggest dispersion. Only this answer has the first quartile below the answer *Agree*. Thus, there are more people that chose options different to *Strongly agree* or *Agree* than in the other questions.

- The answer chosen more times is *Strongly agree* because it is the mode of Q1, Q2, Q4, Q5, Q6, Q8 and Q9, the mode of Q3 and Q10 is the answer *Agree* and the mode of Q7 are the both answers, *Strongly agree* and *Agree*.

The frequencies for the responses to each question are shown in Table 4. Here we have a breakdown of each question: the number of votes for each decision and the percentage corresponding to both of them. Figure 4 shows a bar graphic with the frequency of the answers in the set formed by all the profiles. From Table 4 and Figure 4 we can figure things that could not be figured before. These are the interpretations:

- Q2 and Q4 have a 100% of votes for *Agree* and *Strongly Agree* and Q1 and Q5 have 90% or more of votes for *Agree* and *Strongly Agree* while

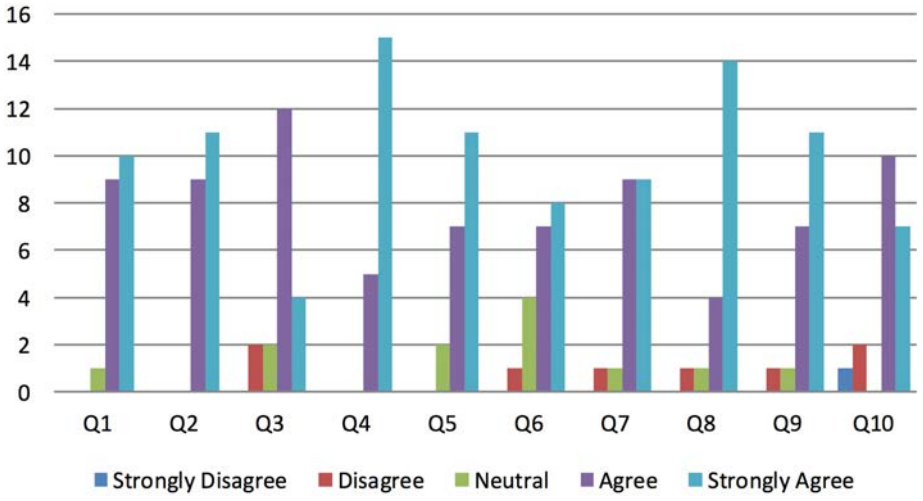



Figure 4: Overall response distribution.

only the 10% or less voted *Neutral*. This means that the majority of the participants agree theses declarations.

- Q6, Q7, Q8, and Q9 have a 10% of votes for *Disagree* and *Neutral* and Q10 have a 15% of votes for *Disagree* and *Strongly Disagree*. This means that the majority agree theses declarations but there are a few that are indecisive or do not believe in these declarations. Moreover, Q10 is the unique declaration with some votes for *Disagree* although the majority votes for *Agree* and *Strongly Agree*.

## 5. Conclusions

In this paper we introduce Bilrost, a novel proposal that provides a solution to handle heterogeneous and ubiquitous Smart Objects through humans social networks. Bilrost allows the actuators to be connected to human social networks and look for messages that contains actions to do.

For this proposal, we defined a DSL called BSL and with it we can define devices and its properties which are the social networks' configuration, filters

that the messages from social networks have to accomplish, actuators which compose the device and actuators' actions. We have shown the grammar of BSL and we have explained how it works with a example.

Due to BSL, Bilrost could be used by a wide range of people without and with knowledge about programming. However, the actions implementation has to been completed by the users. Thus, everybody can define the device with its properties and generate a project which is connected to social networks although the projects completion has to be made by a user with programming skills. However, this research's aim is to interconnect objects and humans through social networks.

To validate that people could effectively create applications connected to Twitter we performed various evaluations where we obtained that users were able to generate projects connected to Twitter and they were able to identify the messages that invoke the defined actions in an average of 648.75 s.

Moreover, the evaluation also consisted in filling a survey so that users gave their opinions. The questions that made up the survey were about user experience while using the proposed solution. The 80% of the declarations we posed to users obtained more than 80% positive or very positive assessments and the lowest rated declarations obtained a 75% of positive or very positive assessments.

Taking into consideration the results of the evaluations, we can say that Bilrost can be very useful to interconnect objects and humans and handle the objects in an easy way, even without complex programming skills or technical skills.

Bilrost could be a small step to achieve that the IoT was able to be more present in our day-to-day lives.

## 6. Future work

The Internet of Things could be the future, facilitating making its use easier is necessary. This research is still not finished and the future work arising from

it could be:

- **Application to obtain Twitter tokens and to write programs with BSL**: Create a web application where user could have to sign in with his Twitter user and then, he could write a program where the Twitter data is already written.

- **Addition of sensors to the BSL to generate applications that sensors could run and publish their state in the human social networks**: Updating the syntax of BSL to define sensors of a device and how to publish their data in social networks. It could be a way to establish a two-way communications between humans and objects through actual social networks.

- **Creation of a graphic DSL to make easier the applications generation**: A textual DSL is good for researches and for users who have minor knowledges about programming but a graphical DSL is necessary to allow everybody to create applications.

- **Development of a new layer to add the specific implementation in a easy way**: The generated applications are not end-user applications and users have to complete them as it was showed in Section 3.2.3. We should improve BSL to allow adding implementation to the generated applications so that everybody could create applications without any knowledge about programming.

- **Compare and study the use of more social networks**: Currently, BSL only supports Twitter as social networks but it is prepared to be extended with any other social network. A future work would be compare current social networks and choice social networks that could be useful to interconnect humans and objects.

## 7. Acknowledgements

This work was performed by the "Ingenieria Dirigida por Modelos MDE-RG" research group at the University of Oviedo under Contract No. FC-15-GRUPIN14-084 of the research project "Ingeniera Dirigida Por Modelos MDE-RG". Project financed by PR Proyecto Plan Regional.

## References

## References

[1] H. G. H. Gu, D. W. D. Wang, A Content-aware Fridge based on RFID in smart home for home-healthcare, in: 2009 11th International Conference on Advanced Communication Technology, Vol. 02, 2009, pp. 987–990.

[2] C. Han, J. M. Jornet, E. Fadel, I. F. Akyildiz, A cross-layer communication module for the Internet of Things, Computer Networks 57 (3) (2013) 622–633. doi:10.1016/j.comnet.2012.10.003.
URL http://www.sciencedirect.com/science/article/pii/S138912861200357X

[3] K. a. Hribernik, Z. Ghrairi, C. Hans, K.-D. Thoben, Co-creating the Internet of Things - First experiences in the participatory design of Intelligent Products with Arduino, in: 2011 17th International Conference on Concurrent Enterprising, 2011, pp. 1–9.

[4] L. Hao, X. Lei, Z. Yan, Y. ChunLi, The application and implementation research of smart city in China, in: 2012 International Conference on System Science and Engineering (ICSSE), 2012, pp. 288–292. doi:10.1109/ICSSE.2012.6257192.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6257192

[5] R. Lea, M. Blackstock, Smart Cities: An IoT-centric Approach, in: Proceedings of the 2014 International Workshop on Web Intelligence and Smart

Sensing, IWWISS '14, ACM, New York, NY, USA, 2014, pp. 12:1—-12:2. doi:10.1145/2637064.2637096.
URL http://doi.acm.org/10.1145/2637064.2637096

[6] F. Telefónica, La Sociedad de la Información en España 2014, Grupo Planeta Spain, 2015.

[7] N. Bulusu, D. Estrin, L. Girod, Scalable coordination for wireless sensor networks: self-configuring localization systems, in: Proc. of the 6th International Symposium on Communication Theory and Applications(ISCT A 01), Ambleside, UK, no. July, 2001, pp. 1–6.
URL http://gicl.cs.drexel.edu/people/regli/Classes/CS680/Papers/SensorNets/iscta-2001.pdf

[8] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Communications Magazine 40 (8) (2002) 102–105. doi:10.1109/MCOM.2002.1024422.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1024422

[9] D. Guinard, V. Trifa, Towards the Web of Things : Web Mashups for Embedded Devices, in: Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), 2009, pp. 1–8. doi:10.1.1.155.3238.

[10] D. Guinard, V. Trifa, E. Wilde, A resource oriented architecture for the web of things, in: 2010 Internet of Things, IoT 2010, 2010, pp. 1–8. doi:10.1109/IOT.2010.5678452.

[11] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, D. Savio, Interacting with the SOA-based internet of things: Discovery, query, selection, and on-demand provisioning of web services, IEEE Transactions on Services Computing 3 (3) (2010) 223–235. doi:10.1109/TSC.2010.3.

[12] C. González García, C. P. García-Bustelo, J. P. Espada, G. Cueva-Fernandez, Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios, Computer Networks 64 (C) (2014) 143–158. doi:10.1016/j.comnet.2014.02.010.
URL http://authors.elsevier.com/sd/article/S1389128614000528

[13] L. Atzori, A. Iera, G. Morabito, The Internet of Things: A survey, Computer Networks 54 (15) (2010) 2787–2805. doi:10.1016/j.comnet.2010.05.010.
URL http://linkinghub.elsevier.com/retrieve/pii/S1389128610001568

[14] K. Gama, L. Touseau, D. Donsez, Combining heterogeneous service technologies for building an Internet of Things middleware, Computer Communications 35 (4) (2012) 405–417. doi:10.1016/j.comcom.2011.11.003.
URL http://linkinghub.elsevier.com/retrieve/pii/S0140366411003586

[15] L. Tan, Future internet: The Internet of Things, in: 2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE), Ieee, Chengdu, 2010, pp. V5–376–V5–380. doi:10.1109/ICACTE.2010.5579543.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5579543

[16] The US National Intelligence Council, Six Technologies with Potential Impacts on US Interests out to 2025, Tech. rep., The National Intelligence Council (2008).

[17] C. Gonzalez Garcia, J. P. Espada, E. R. Nunez-Valdez, V. G. Diaz, Midgar: Domain-Specific Language to Generate Smart Objects for an Internet of Things Platform, in: 2014 Eighth International Conference on Innovative

Mobile and Internet Services in Ubiquitous Computing, IEEE, Birmingham, United Kingdom, 2014, pp. 352–357. doi:10.1109/IMIS.2014.48.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6975488

[18] C. Wong, D. McFarlane, A. Ahmad Zaharudin, V. Agarwal, The intelligent product driven supply chain, in: IEEE International Conference on Systems, Man and Cybernetics, Vol. vol.4, IEEE, 2002, p. 6. doi:10.1109/ICSMC.2002.1173319.
URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1173319http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1173319

[19] K. A. Hribernik, Z. Ghrairi, C. Hans, K.-d. Thoben, Co-creating the Internet of Things - First Experiences in the Participatory Design of Intelligent Products with Arduino, in: Concurrent Enterprising (ICE), 2011 17th International Conference on, no. Ice, Aachen, 2011, pp. 1–9.

[20] G. G. Meyer, K. Främling, J. Holmström, Intelligent Products: A survey, Computers in Industry 60 (3) (2009) 137–148. doi:10.1016/j.compind.2008.12.005.
URL http://linkinghub.elsevier.com/retrieve/pii/S0166361508001590

[21] M. Blackstock, R. Lea, A. Friday, Uniting online social networks with places and things, in: Proceedings of the Second International Workshop on Web of Things, WoT '11, ACM, New York, NY, USA, 2011, pp. 5:1—-5:6. doi:10.1145/1993966.1993974.
URL http://dx.doi.org/10.1145/1993966.1993974

[22] D. Rao, D. Yarowsky, A. Shreevats, M. Gupta, Classifying latent user attributes in twitter, in: Proceedings of the 2nd international workshop on Search and mining user-generated contents - SMUC '10, ACM Press, New York, New York, USA, 2010, p. 37. arXiv:1690219.1690245,

doi:10.1145/1871985.1871993.
URL http://portal.acm.org/citation.cfm?doid=1871985.1871993

[23] H. Achrekar, A. Gandhe, R. Lazarus, S. H. Yu, B. Liu, Predicting flu trends using twitter data, in: 2011 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2011, IEEE, 2011, pp. 702–707. doi:10.1109/INFCOMW.2011.5928903.

[24] M. Kranz, L. Roalter, F. Michahelles, F. Michahelles, Things That Twitter: Social Networks and the Internet of Things, in: What can the Internet of Things do for the Citizen (CIoT) Workshop at The Eighth International Conference on Pervasive Computing: Pervasive 2010, 2010, pp. 1–10.

[25] D. Guinard, M. Fischer, V. Trifa, Sharing using social networks in a composable Web of Things, in: Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on, 2010, pp. 702–707. doi:10.1109/PERCOMW.2010.5470524.

[26] L. Atzori, A. Iera, G. Morabito, From "smart objects" to "social objects": The next evolutionary step of the internet of things, IEEE Communications Magazine 52 (1) (2014) 97–105. doi:10.1109/MCOM.2014.6710070.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6710070

[27] E. Dijkstra, The humble programmer, Communications of the ACM 15 (October 1972) (1972) 859–866.

[28] E. González, H. Fernández, V. Díaz, General purpose MDE tools, International Journal of Interactive Multimedia and Artificial Intelligence 1 (2008) 72–75.

[29] V. García-Díaz, H. Fernández-Fernández, E. Palacios-González, B. C. P. G-Bustelo, O. Sanjuán-Martínez, J. M. C. Lovelle, TALISMAN MDE: Mixing MDE principles, Journal of Systems and Software 83 (7) (2010) 1179–1191. doi:10.1016/j.jss.2010.01.010.

[30] B. Selic, MDA manifestations, The European Journal for the Informatics Professional IX (2) (2008) 12–16.

[31] A. V. Deursen, P. Klint, J. Visser, Domain-specific languages: an annotated bibliography, ACM Sigplan Notices 35 (6) (2000) 26–36.

[32] E. R. Núñez Valdez, O. Sanjuan-Martinez, C. P. G. Bustelo, J. M. C. Lovelle, G. Infante-Hernandez, Gade4all: Developing Multi-platform Videogames based on Domain Specific Languages and Model Driven Engineering, International Journal of Interactive Multimedia and Artificial Intelligence 2 (Regular Issue) (2013) 33–42.

[33] A. van Deursen, P. Klint, Little languages: Little maintenance?, Journal of software maintenance 10 (2) (1998) 75–92. doi:10.1016/j.jvs.2012.10.105.

[34] L. Atzori, A. Iera, G. Morabito, SIoT: Giving a Social Structure to the Internet of Things, IEEE Communications Letters 15 (11) (2011) 1193–1195. doi:10.1109/LCOMM.2011.090911.111340.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6042288

[35] J. Choi, C. W. Yoo, Connect with things through instant messaging, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 4952 LNCS, Springer, 2008, pp. 276–288. doi:10.1007/978-3-540-78731-0_18.

[36] S. Aurell, Remote controlling devices using instant messaging: building an intelligent gateway in Erlang/OTP, in: Proceedings of the 2005 ACM SIGPLAN workshop on Erlang, ACM, 2005, pp. 46–51.

[37] A. Roychowdhury, S. Moyer, Instant messaging and presence for sip enabled networked appliances, Publisher unknown.

[38] R. Likert, A technique for the measurement of attitudes., Archives of Psychology 22 140 (1932) 55.

**Vitae**

**Daniel Meana-Llorián** is a Graduated Engineering in Computer Systems from School of Computer Engineering of Oviedo in 2014 (University of Oviedo, Spain). Currently, he is a student of M.S. in Web Engineering. His research interests include Mobile technologies, Web Engineering, the Internet of Things and exploration of emerging technologies related with the previous ones.

**Cristian González García** is a Technical Engineering in Computer Systems and M.S. in Web Engineering from School of Computer Engineering of Oviedo in 2011 and 2013 (University of Oviedo, Spain). Currently, he is a Ph.D. candidate in Computers Science. His research interests are in the field of the Internet of Things, Web Engineering, Mobile Devices and Modeling Software with DSL, and MDE.

**Cristina Pelayo G-Bustelo** is a Lecturer in the Computer Science Department of the University of Oviedo. Ph.D. from the University of Oviedo in Computer Engineering. Her research interests include Object-Oriented technology, Web Engineering, eGovernment, Modeling Software with BPM, DSL and MDA.

**Juan Manuel Cueva Lovelle** is a Mining Engineer from Oviedo Mining Engineers Technical School in 1983 (Oviedo University, Spain). Ph. D. from Madrid Polytechnic University, Spain (1990). From 1985 he is Professor at the Languages and Computers Systems Area in Oviedo University (Spain). ACM and IEEE voting member.His research interests include Object-Oriented technology, Language Processors, Human-Computer Interface, Web Engineering, Modeling Software with BPM, DSL and MDA.

| Id | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 | Q8 | Q9 | Q10 |
|---|---|---|---|---|---|---|---|---|---|---|
| p01 | 4 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 2 |
| p02 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 4 |
| p03 | 4 | 5 | 4 | 5 | 5 | 3 | 2 | 5 | 4 | 5 |
| p04 | 4 | 4 | 4 | 5 | 3 | 3 | 5 | 3 | 5 | 4 |
| p05 | 5 | 4 | 5 | 5 | 5 | 3 | 4 | 2 | 2 | 1 |
| p06 | 4 | 5 | 4 | 4 | 4 | 2 | 5 | 4 | 3 | 4 |
| p07 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 5 | 5 | 4 |
| p08 | 4 | 5 | 3 | 4 | 4 | 5 | 4 | 5 | 5 | 4 |
| p09 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 4 | 5 |
| p10 | 4 | 5 | 4 | 5 | 5 | 4 | 5 | 5 | 5 | 5 |
| p11 | 5 | 4 | 2 | 5 | 3 | 3 | 4 | 5 | 5 | 2 |
| p12 | 4 | 5 | 2 | 5 | 5 | 5 | 4 | 5 | 5 | 4 |
| p13 | 3 | 4 | 4 | 5 | 4 | 5 | 4 | 4 | 4 | 4 |
| p14 | 5 | 4 | 3 | 5 | 4 | 5 | 5 | 5 | 4 | 5 |
| p15 | 5 | 4 | 4 | 5 | 4 | 4 | 4 | 5 | 5 | 5 |
| p16 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| p17 | 4 | 4 | 4 | 4 | 5 | 4 | 4 | 4 | 4 | 4 |
| p18 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| p19 | 5 | 4 | 5 | 4 | 5 | 4 | 4 | 4 | 4 | 4 |
| p20 | 5 | 5 | 4 | 5 | 4 | 5 | 4 | 5 | 4 | 4 |

Table 2: Responses of the users for each question.

| | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 | Q8 | Q9 | Q10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Min | 3 | 4 | 2 | 4 | 3 | 2 | 2 | 2 | 2 | 1 |
| Quartile 1 | 4 | 4 | 4 | 4.75 | 4 | 3.75 | 4 | 4 | 4 | 4 |
| Median | 4.5 | 5 | 4 | 5 | 5 | 4 | 4 | 5 | 5 | 4 |
| Quartile 3 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Max | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Range | 2 | 1 | 3 | 1 | 2 | 3 | 3 | 3 | 3 | 4 |
| Inter Qrt.-Range | 1 | 1 | 0 | 0.25 | 1 | 1.25 | 1 | 1 | 1 | 1 |
| Mode | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 4 |

Table 3: Table with the general descriptive statistics.

| | | Strongly disagree | Disagree | Neutral | Agree | Strongly agree |
|---|---|---|---|---|---|---|
| Q1 | # | 0 | 0 | 5 | 45 | 50 |
| Q2 | # | 0 | 0 | 0 | 45 | 55 |
| Q3 | # | 0 | 10 | 10 | 60 | 20 |
| Q4 | # | 0 | 0 | 0 | 25 | 75 |
| Q5 | # | 0 | 0 | 10 | 35 | 55 |
| Q6 | # | 0 | 5 | 20 | 35 | 40 |
| Q7 | # | 0 | 5 | 5 | 45 | 45 |
| Q8 | # | 0 | 5 | 5 | 20 | 70 |
| Q9 | # | 0 | 5 | 5 | 35 | 55 |
| Q10 | # | 5 | 10 | 0 | 50 | 35 |

Table 4: Frequency table for the general responses.

# ANEXO IV. IoFClime: The fuzzy logic and the Internet of Things to control indoor temperature regarding the outdoor ambient conditions

# IoFClime: The fuzzy logic and the Internet of Things to control indoor temperature regarding the outdoor ambient conditions.

Daniel Meana-Llorián[a,*], Cristian González García[a], B. Cristina Pelayo G-Bustelo[a], Juan Manuel Cueva Lovelle[a], Nestor Garcia-Fernandez[a]

[a] *University of Oviedo, Department of Computer Science, Sciences Building, C/Calvo Sotelo s/n 33007, Oviedo, Asturias, Spain*

**Abstract**

The Internet of Things is arriving to our homes or cities. We can call these fields Smart Homes or Smart Cities. Monitoring environmental conditions of cities can allow adapting the indoor locations of the cities in order to be more comfortable for people who stay there. A way to improve the indoor conditions is an efficient temperature control. However, an efficient temperature control of a specific location depends on many factors like the possible combinations of outdoor temperature and humidity. Therefore, setting the indoor temperature is not setting a value according to other value. There are more factors so the traditional logic based in binary states cannot be used. Many problems cannot be solved with a set of binary solutions and we need a new way of development. Fuzzy logic can interpret many states, more than two states, giving to computers the capacity to react in a similar way to people. In this paper we will propose a new approach to control the temperature using the Internet of Things, its platforms and fuzzy logic in order to saving energy and regarding not only the indoor temperature but also the outdoor temperature and humidity. Finally, we will conclude that the fuzzy approach allows us to achieve an energy saving

*Corresponding author

*Email addresses:* danielmeanallorian@gmail.com (Daniel Meana-Llorián), gonzalezgarciacristian@hotmail.com (Cristian González García), crispelayo@uniovi.es (B. Cristina Pelayo G-Bustelo), cueva@uniovi.es (Juan Manuel Cueva Lovelle), nestor@uniovi.es (Nestor Garcia-Fernandez)

around 40% and thus, save money.

*Keywords:* Internet of Things, Fuzzy Logic, Temperature control, Temperature sensors

---

## 1. Introduction

The Internet of Things (IoT) is a term very popular nowadays. Everyday we can listen people taking about Smart Homes [1, 2], Smart Cities [3, 4], Smart Earth [3], and many other kind of distributed intelligence around heterogeneous and ubiquitous objects. The Internet of Things allows gathering huge quantity of data that can be processed to help making different decisions. These data could be very varied and confused and processing them could become inoperable.

Humans and computers make decisions in a different way. Whereas human reason using words, computers use numbers [5]. Moreover, although the logic applied by humans seems more primitive, they can make better decisions in the real-world when unexpected problems occur. These differences change the way of processing the data collected by IoT sensors. Computing with words could improve the capability of computers to deal with problems of real-world and thus, improving decision making [5].

The human capability to take decisions without computations is usually referred as 'common sense'. Common sense allows us to take decisions quickly although common sense is not always right. For example, in the past, we thought that the Earth was flat due to common sense [5]. Moreover, common sense provides a way to get solutions to problems with incomplete or imprecise information whereas classical logic is better to resolve problems well defined. Furthermore, if we always used classical logic, we would take only a few decisions throughout the day.

While classical logic deals with binary sets of values, 0 (false) and 1 (true), fuzzy logic deals with a range of values that represents different degrees of truth on a scale between completely false and completely true [6]. A range of values allows having more states and thus, making decisions knowing more information.

For example, decisions about when pushing collected data to a server could be optimised with fuzzy logic [6]. Applying classical logic, we would have only two states, push or wait, whereas if we applied fuzzy logic, we could have many states which would indicate the best moment to push data to the server and what data should be pushed.

Our daily life could be benefited with the combination of the IoT and fuzzy logic. For example, local businesses could enhance the management of temperature systems that involve the use of heating systems and air-conditioning systems in order to adapt the indoor conditions taking into consideration the outdoor conditions. Following this path, we would be able to achieve a really Smart City where the indoor locations will be capable of adapting to the outdoor conditions. Using fuzzy logic will allow us to decide the best moment to heat up or to cool down the environment considering the outdoor conditions.

Moreover, fuzzy logic could help to reduce the energy consumption of cities' buildings in order to achieve the zero energy building (ZEBs). The ZEBs are a part of the way towards achieving the Smart cities [7].

In addition to use fuzzy logic in the IoT, there are several platforms which allow not only publishing data gathered from our sensors but also consuming data gathered from third-party sensors.

The aim of this research is the development of a prototype that will combine fuzzy logic applied to the IoT and the use of IoT platforms where it will able to consume third-party data. The prototype will be able to automate a system that will control the environmental conditions of a specific place. Moreover, the prototype will allow saving energy which is a very important issue in the world of the IoT. Our goals are the following:

- Consuming data from online open IoT platforms.
- Consuming data from sensors located in microcontrollers like Arduino or microcomputers like Raspberry Pi and working with actuators located in these devices.
- Setting fuzzy rules to achieve a better performance in the suggested solu-

tion.

- Keeping an optimum temperature inside a room during a long time.
- Saving energy. Moreover, this can help to achieve economic savings.

The remainder of the paper is organised as follows. In Section 2, we will introduce involved topics in this research like Fuzzy Logic, the IoT, IoT platforms, and the works related with this research. Section 3 will show the case of study, how it works and how its architecture is. Section 4 will show the software and hardware used in this research. In Section 5 we will cover the evaluation and discussion of the data obtained from comparing the behaviour of our approach and the behaviour of the traditional approach. Section 6 will contain the conclusions of this paper and finally, in Section 7 we will describe the possible future work that can be done from here.

## 2. State of the Art

The principal aim of this research is to use fuzzy logic in the context of the IoT. However, before talking about the case study, review involved concepts is very necessary. Moreover, inside the context of the IoT we want to review a special concept, the IoT platforms. These platforms were created to bring the IoT to more people, industry, and researches and thus, to increase the possibilities of the IoT. With these platforms we can intercommunicate objects which are separated by thousands of kilometres or located in places that we cannot imagine. Maybe behind the enemy lines in a battlefield, inside a nuclear plant, in a sealed room full of chemical or radioactive products or even in places where they cannot be reached after being set [8, 9].

The fuzzy logic is also a technology presents in many researches. The uses of this technology are multiples and it could be useful in a wide range of fields. For example, the fuzzy logic can be used in the health simulating insulin pumps [10] or sending health data in the best moment without draining the batteries of the involved devices [11].

In the context of industry, the combination of the IoT and fuzzy logic can help in many situations like controlling the products life cycle [12], monitoring and detecting fires [13], and making decisions about when doing certain procedures in industries [14].

Thus, in this section we will talk about the concepts which are involved in this research: Fuzzy logic, the IoT, and the IoT platforms.

### *2.1. Fuzzy Logic*

The term Fuzzy Logic was introduced by Zadeh as a way to deal with common sense problems, first fuzzy sets were introduced in [15] and after, fuzzy logic in [16]. Since then, Fuzzy logic has been addressed in many researches. A common issue where fuzzy logic is applied is the battery saving. For example, Larios et al. [17] locates a device avoiding many location errors and thus, achieving a better accuracy. Also, their approach allows decreasing the energy consumption of different elements like the GPS. Chamodrakas and Martakos [18] proposed use fuzzy set representation method to select a network in order to be connected in an efficient way, with a low energy consumption, with a high Quality of Service (QoS), and with good performance. Bagchi [19] used fuzzy logic to keep the quality of playback of multimedia streaming and to achieve improvements in the energy consumption. Cueva-Fernandez et al. [6, 20] made two proposals about how to use fuzzy logic in vehicles. In [6], they proposed the improvement of information exchange between vehicles sensors and servers in order to save energy and in [20], they proposed a system to create applications through the voice using fuzzy logic. All of these researches have in common the use of fuzzy logic to make difficult decisions without having clear options to consider.

Fuzzy logic emerged to resolve problems that classical logic cannot address. Classical logic can only deal with binary sets of values (0 or 1), but there are many contexts in which dealing with more values is required. Having more than two values allows having more states and thus, making decisions with more information. These states are usually called linguistic variables and they

can represent characteristics like 'size' whose values could be 'small', 'big', and so on. This approach tries to resolve questions like how big a building is. The answer to this question depends on individual cognition because not everyone would respond the same. For example, for a person who lives in the countryside, a building composed by 6 floors could be big, but for a person who lives in New York, the same building could be small in comparison with the buildings placed in New York.

There are many examples that use fuzzy logic because they need more than two values to represent their states. Grant [10] took the advantages of fuzzy logic to propose a new approach to the diabetic control.

Fuzzy logic allows dealing with vague and ambiguous data in order to make decisions like a person would do it. This logic uses controllers called 'adaptive controllers' [10] or 'expert systems' [21] based in a rules (e.g., if X and Y then Z) to mimic the human fuzzy thinking. These rules represent the knowledge that drives the expert systems and due to this knowledge, the expert systems may decide the optimal decision. The definition of the rules is complex and it requires the use of the linguistic variables since the human representation of the knowledge is fuzzy. The process of understand what a linguistic variable like 'small' means is named fuzzification [22].

### *2.2. The Internet of Things*

In the last years, the interconnection between heterogeneous and ubiquitous objects is one of the most important issues in research and business. This is better known as the Internet of Things [23, 24, 25].

Moreover, the IoT is one of the most important technologies for the future according to the ONU in 2005 [23] where the United Nations predicted a new era with more data traffic generated by objects than generated by people, or the United States National Intelligence Agency [26] Councils report where the IoT is considered one of the six technologies which the United States is more interested in from 2008 to 2025.

However, there is not a standard or unique way to do this. We can use many

platforms to interconnect objects through Internet [27, 28]. Furthermore, there are a lot of standards to communicate the objects like Near Field Communications (NFC), Radio Frequency Identification (RFID), or Bluetooth.

The IoT are a set of technologies that can be used for many purposes. The use of sensors in combination with Smart Objects allow us to read data from the environment and develop a wide range of solutions for many topics.

### *2.3. IoT platforms*

Due to the popularity and the growth of the IoT, there have been developed several platforms to interconnect objects through the Internet. There are many platforms and many types and their aims can be very different.

Several platforms were developed for business community like Xively [29] and they allow businesses to be benefited from the IoT advantages in order to connect their products with their users in a safe way, manage their information, improve clients satisfaction and make new revenue sources.

Many others like ThingSpeak [30] or Paraimpu [31], allow registered users to consume open data in many formats like JSON, CSV, or XML and allow users to share their data from different sources. Both are free but the registrations are controlled by invitations or they require be approved.

There also an IoT platform called Midgar [27, 28] which generates applications which connect heterogeneous and ubiquitous objects through the Internet.

In our proposal, we opted for using ThingSpeak and Midgar in order to obtain data from third-party solutions.

#### *2.3.1. ThingSpeak*

Thinkspeak [30] is an open IoT platform for interconnecting objects using web standards. Some of its services are data registration, processing, and distribution; location based services, many plugins, and others useful services. The first thing to do in order to create new objects or services is to choose the device type (Arduino, Netduino, ioBridge, or others) and to give the access data (IP, port, subnet mask, and others). This step allows creating a new channel to be

used to push the connected devices data, show it, consume it through HTTP requests or download it in XML, JSON, or CSV format. Moreover, this platform also offers a data accessibility control.

#### *2.3.2. Midgar*

Midgar [27, 28] uses a Domain-Specific Language (DSL) called MOISL to generate applications that interconnect heterogeneous and ubiquitous objects through the Internet. Midgar users can interconnect different objects using a graphic DSL in an easy and fast way. Through the DSL, users can define the application flow in order to connect many objects, read any data from their services and invoke certain actions in the same objects or in other connected objects. These actions can be different among different objects and one application can interconnect many different objects and invoke different actions.

### *2.4. Related work*

There are several commercial solutions that allow handling the temperature of specific places. Two of this solutions are Loxone [32] and Tado° [33]. Moreover, there are also some researches that combine the IoT and fuzzy logic like Vitruvius [6, 20]

#### *2.4.1. Loxone*

Loxone [32] is a home automation system that can control blinds, lights, music systems, or climatisation systems. If we are focused in the temperature, this system has several advantages and disadvantages in comparison with our approach. Some advantages are that it takes into consideration the user geolocation and it registers the temperature to calculate trends and to improve the future work. However, it also has disadvantages like the high buying price and the installation complexity. Moreover, Loxone does not use third-party data whereas our purpose can gather data from online open IoT platforms.

#### *2.4.2. Tado°*

Tado° [33] is a smart thermostat that can be controlled through a web interface or with a smartphone. It can use the user geolocation like Loxone although its installation is easier than Loxone. Tado° also allows us to save money with a good temperature control. Moreover, it generates detailed reports and it takes into consideration the weather forecast to keep the ambient temperature. The last feature is comparable with the gathering of data from online open IoT platforms so this is a point in common between Tado° and our proposal. However, there are also disadvantages like the high buying price and it does not automate anything so it cannot be considered smart.

#### *2.4.3. Vitruvius*

Vitruvius [6, 20] is a platform which allows users to generate applications in real time that they use sensors installed in vehicles. For that purpose, Vitruvius has a web application composed by an editor of applications that allows designing the applications. Some of available sensors are speed sensors, temperature sensors, and many similar others. Vitruvius combines the IoT with fuzzy logic in order to reduce the push data frequency. Its first versions did not use fuzzy logic, therefore, the data were being pushed every second [34]. Due to fuzzy logic, the amount of data pushed were reduced and the data quality and the data analysis performance were improved. However, the use of fuzzy logic makes it harder to detect errors. Whereas Vitruvius used fuzzy logic to automate the pushing of data to server in the best moment to do it, our proposal use the fuzzy logic to automate systems that control the temperature with the intention of adjusting the temperature to obtain a good thermal sensation and thus, save money through the smart approach driven by fuzzy logic.

## 3. Case study

In this paper we propose a system that control the temperature of a specific place in an automated way using fuzzy logic. For this purpose, we developed a prototype inspired in the context of the IoT. The prototype consumes data

from sensors and IoT platforms. The sensors were installed in microcontrollers and microcomputers. However, the microcomputers were used not only for consuming data but also for handling the actuators that simulate the temperature control. For this paper, we simulated the temperature control with five LEDs (Figure 1) that represent the next five states in a closed room:

- The temperature would be extremely high and the air-conditioning systems would be on at maximum power. This state would be represented by two yellow LEDs.
- The temperature would be high and the air-conditioning systems would be on at normal power. This state would be represented by one yellow led.
- The temperature would be suitable and the heating systems and the air-conditioning systems would not be on in order to keep the suitable temperature. This state would be represented by the blue led located in the middle.
- The temperature would be low and the heating systems would be on at maximum power. This state would be represented by one red led.
- The temperature would be extremely low and the heating systems would be on at maximum power. This state would be represented by two red LEDs.

### *3.1. Temperature control*

The developed prototype uses three input values from different IoT sources to make the decisions that control the temperature.

- Outdoor humidity.
- Outdoor temperature.
- Indoor temperature.

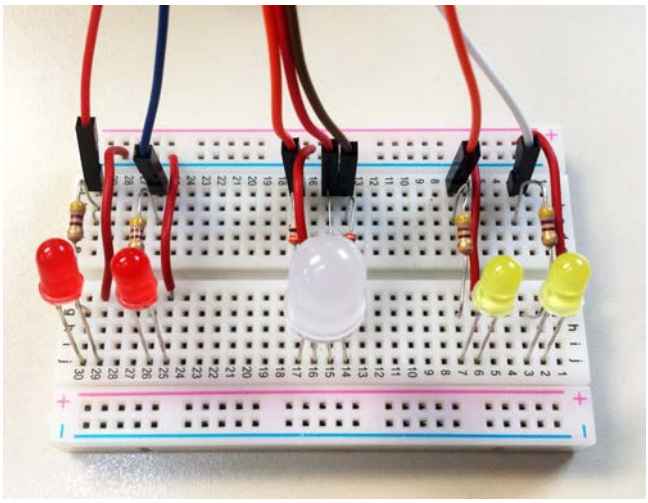


Figure 1: The five LEDs that simulate the temperature control.

| Temperature (ºC) | Humidity (%) | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 55 | 60 | 65 | 70 | 75 | 80 | 85 | 90 | 95 | 100 |
| 20 | 16 | 16 | 17 | 17 | 17 | 18 | 18 | 19 | 19 | 19 | 19 | 19 | 20 | 20 | 20 | 21 | 21 | 21 | 21 | 21 | 21 |
| 21 | 18 | 18 | 18 | 19 | 19 | 19 | 19 | 19 | 20 | 20 | 20 | 20 | 21 | 21 | 21 | 22 | 22 | 22 | 22 | 22 | 23 |
| 22 | 19 | 19 | 19 | 20 | 20 | 20 | 20 | 20 | 21 | 21 | 21 | 21 | 22 | 22 | 22 | 22 | 23 | 23 | 23 | 23 | 24 |
| 23 | 20 | 20 | 20 | 20 | 21 | 21 | 22 | 22 | 22 | 23 | 23 | 23 | 23 | 24 | 24 | 24 | 24 | 24 | 24 | 25 | 25 |
| 24 | 21 | 21 | 22 | 22 | 22 | 22 | 23 | 23 | 23 | 24 | 24 | 24 | 24 | 25 | 25 | 25 | 25 | 26 | 26 | 26 | 26 |
| 25 | 22 | 23 | 23 | 23 | 24 | 24 | 24 | 24 | 24 | 24 | 25 | 25 | 25 | 26 | 26 | 26 | 27 | 27 | 27 | 28 | 28 |
| 26 | 24 | 24 | 24 | 24 | 25 | 25 | 25 | 26 | 26 | 26 | 26 | 27 | 27 | 27 | 27 | 28 | 28 | 29 | 29 | 29 | 30 |
| 27 | 25 | 25 | 25 | 25 | 26 | 26 | 26 | 27 | 27 | 27 | 27 | 28 | 28 | 29 | 29 | 30 | 30 | 31 | 31 | 31 | 33 |
| 28 | 26 | 26 | 26 | 26 | 27 | 27 | 27 | 28 | 28 | 28 | 29 | 29 | 29 | 30 | 31 | 32 | 32 | 33 | 34 | 34 | 36 |
| 29 | 26 | 26 | 27 | 27 | 27 | 28 | 29 | 29 | 29 | 29 | 30 | 30 | 31 | 33 | 33 | 34 | 35 | 35 | 37 | 38 | 40 |
| 30 | 27 | 27 | 28 | 28 | 28 | 28 | 29 | 29 | 30 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 39 | 40 | 41 | 45 |
| 31 | 28 | 28 | 29 | 29 | 29 | 29 | 30 | 31 | 31 | 31 | 33 | 34 | 35 | 36 | 37 | 39 | 40 | 41 | 45 | 45 | 50 |
| 32 | 29 | 29 | 29 | 29 | 30 | 31 | 31 | 33 | 33 | 34 | 35 | 35 | 37 | 39 | 40 | 42 | 44 | 45 | 51 | 51 | 55 |
| 33 | 29 | 29 | 30 | 30 | 31 | 33 | 33 | 34 | 34 | 35 | 36 | 38 | 39 | 42 | 43 | 45 | 49 | 49 | 53 | 54 | 55 |
| 34 | 30 | 30 | 31 | 31 | 32 | 34 | 34 | 35 | 36 | 37 | 38 | 41 | 42 | 44 | 47 | 48 | 50 | 52 | 55 | | |
| 35 | 31 | 32 | 32 | 32 | 33 | 35 | 35 | 37 | 37 | 40 | 40 | 44 | 45 | 47 | 51 | 52 | 55 | | | | |
| 36 | 32 | 33 | 33 | 34 | 35 | 36 | 37 | 39 | 39 | 42 | 43 | 46 | 49 | 50 | 54 | 55 | | | | | |
| 37 | 32 | 33 | 34 | 35 | 36 | 38 | 38 | 41 | 41 | 44 | 46 | 49 | 51 | 55 | | | | | | | |
| 38 | 33 | 34 | 35 | 36 | 37 | 39 | 40 | 43 | 44 | 47 | 49 | 51 | 55 | | | | | | | | |
| 39 | 34 | 35 | 36 | 37 | 38 | 41 | 41 | 44 | 46 | 50 | 50 | 55 | | | | | | | | | |
| 40 | 35 | 36 | 37 | 39 | 40 | 43 | 43 | 47 | 49 | 53 | 55 | | | | | | | | | | |
| 41 | 35 | 36 | 38 | 40 | 41 | 44 | 45 | 49 | 50 | 55 | | | | | | | | | | | |
| 42 | 36 | 37 | 39 | 41 | 42 | 45 | 47 | 50 | 52 | 55 | | | | | | | | | | | |
| 43 | 37 | 38 | 40 | 42 | 44 | 47 | 49 | 53 | 55 | | | | | | | | | | | | |
| 44 | 38 | 39 | 41 | 44 | 45 | 49 | 52 | 55 | | | | | | | | | | | | | |
| 45 | 38 | 40 | 42 | 45 | 47 | 50 | 54 | 55 | | | | | | | | | | | | | |
| 46 | 39 | 41 | 43 | 45 | 49 | 51 | 55 | | | | | | | | | | | | | | |
| 47 | 40 | 42 | 44 | 47 | 51 | 54 | 55 | | | | | | | | | | | | | | |
| 48 | 41 | 43 | 45 | 49 | 53 | 55 | | | | | | | | | | | | | | | |
| 49 | 42 | 45 | 47 | 50 | 54 | 55 | | | | | | | | | | | | | | | |
| 50 | 42 | 45 | 48 | 50 | 55 | | | | | | | | | | | | | | | | |

Figure 2: Apparent temperature according to the temperature and humidity.

The prototype estimates the outdoor apparent temperature using the outdoor humidity, the outdoor temperature, and fuzzy rules. These fuzzy rules were based in the values of the table shown in Figure 2 whose input values are the temperature and the humidity. The output after applying the fuzzy rules are a set of values that indicate the degree of truth of seven linguistic variables that were used in order to represent the apparent temperature. The linguistic variables used were the next fuzzy sets:

- Extremely low: Between -15°C and -5°C.
- Very low: Between -7°C and 3°C.
- Low: Between 0°C and 18°C.
- Normal: Between 14°C and 24°C.
- High: Between 22°C and 30°C.
- Very high: Between 28°C and 38°C.
- Extremely high: Between 35°C and 50°C.

The humidity and the outdoor temperature were also defined using linguistic variables in order to apply fuzzy logic. The outdoor temperature linguistic variables that we used were the same as the used to represent the apparent temperature. Nevertheless, the humidity linguistic variables were the next fuzzy sets:

- Very low: Less than 40%.
- Low: Between 30% and 60%.
- Normal: Between 50% and 70%.
- High: Between 60% and 80%.
- Very high: More than 75%.

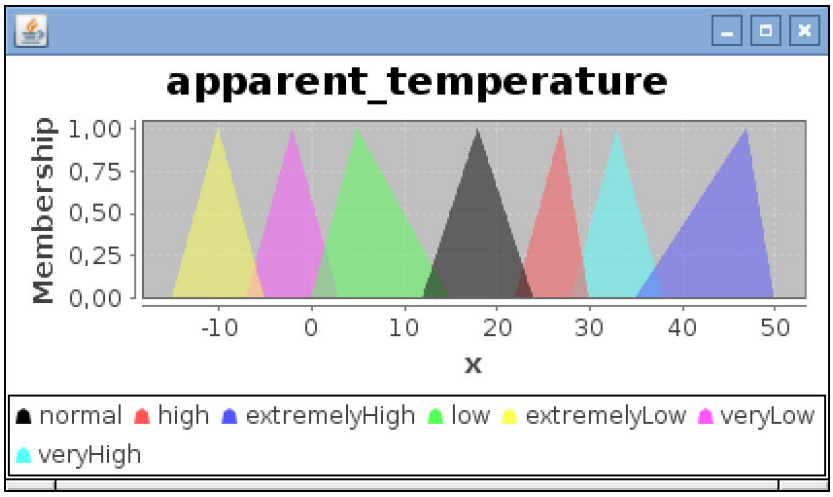



Figure 3: Distribution of linguistic variables for outdoor apparent temperature.

From the degree of truth of each linguistic variable we select the centre of gravity in order to take the inputs of the next step. We can see the distribution of the linguistic variables for the outdoor apparent temperature in Figure 3 and for the outdoor humidity in Figure 4.

The outdoor humidity was obtained from an IoT platform that we talked about before, ThingSpeak, and the format chosen was JSON. The outdoor temperature was obtained from a microcontroller Arduino UNO with a temperature sensor connected to the platform Midgar which was publishing the temperature data in JSON format. The indoor temperature was obtained from a temperature sensor located in a Raspberry Pi 2 where the prototype was running. The indoor temperature linguistic variables were the same as the ones used with the previous temperatures.

After estimating the outdoor apparent temperature, the prototype executes other fuzzy rules whose input is the indoor temperature centre of gravity and the outdoor apparent temperature centre of gravity in order to determine how the actuators should work. The output of this rules are the next linguistic variables and fuzzy sets and it defined the five states that we already mentioned before:

- Heating system at maximum power: Less than 15.

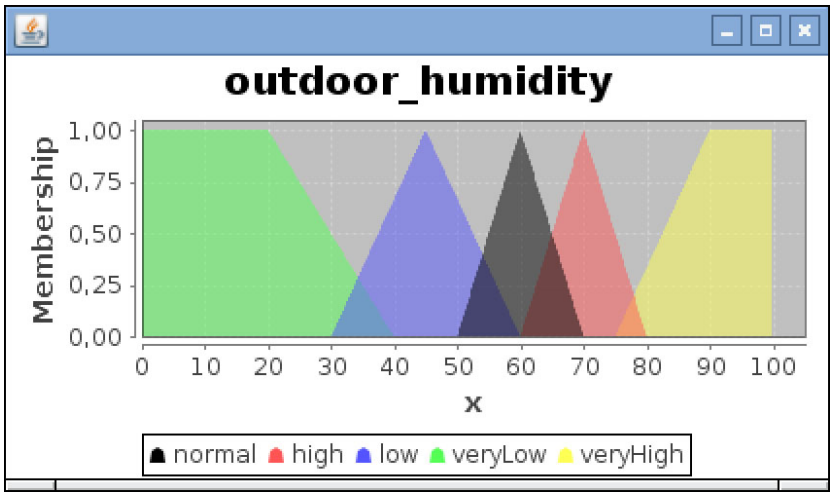



Figure 4: Distribution of linguistic variables for outdoor humidity.

- Heating system at normal power: Between 10 and 20.
- No system running: Between 18 and 22.
- Air condition system at normal power: Between 20 and 30.
- Air condition system at maximum power: More than 25.

After calculating the centre of gravity, we obtained the action to do. All values shown in this section were been influenced due to the experience obtained during the research and they should be adjusted to the conditions of each place.

### *3.2. Architecture*

The systems architecture can be divided in the next three layers: **Data collection**, **Data processor** and **Actuators**. The Figure 5 shows the three layers and their modules. Each layer depends on the others. The first one, **Data collections**, contains the modules which gather data to be processed from external and local sources. The layer **Data processor** takes the collected data and processes them in order to make decisions. The last layer, **Actuators**, takes the decisions made in the previous layer and routes them to the target actuator in order to do the appropriate action.

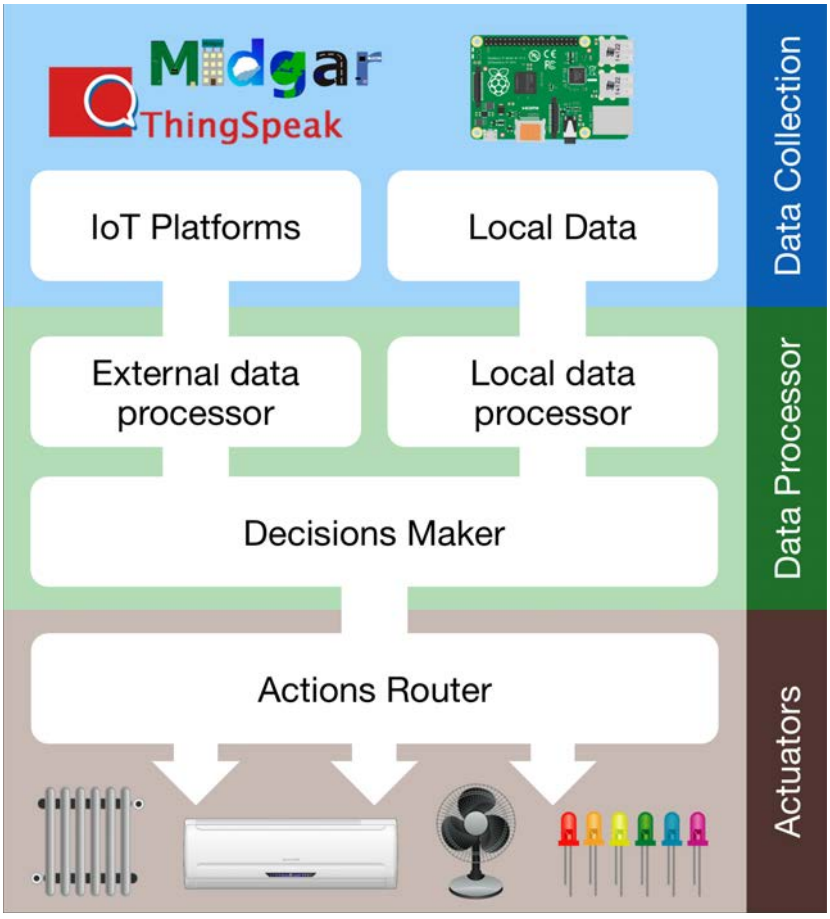



Figure 5: System architecture.

*3.2.1. Data collection*

Our approach requires data from different IoT platforms in order to handle variety data from external sources. The system is prepared to use any platform with minor changes. In our tests we focus in the use of two IoT platforms, one for temperature, Midgar, and another for humidity, ThingSpeak.

The IoT platform used to collect the temperature data was Midgar. Midgar is useful to interconnect Smart Objects in an easy way. In fact, knowledge about languages programming is not necessary [27, 28]. We used Midgar to connect an Arduino UNO with our system through HTTP requests. This Arduino had a temperature sensor that we used to obtain the outdoor temperature.

In order to collect the humidity data, we used another IoT platform, ThingSpeak. This platform allows us to gather the humidity data through HTTP, that other users of the platform had published before.

Our system collects data not only from external sources but also from local sources. We used a Raspberry Pi 2 with a temperature sensor to obtain the temperature of the room where we want to handle the heating and air condition systems.

The data obtained are pre-processed in order to transform it in a common format that the Data processor can handle.

*3.2.2. Data processor*

The output of the previous layer is a set of properly formatted values that represent the outdoor temperature, the outdoor humidity, and the indoor temperature. These data have to be processed in order to decide what actions to do. For this purpose, we implemented two modules that process the external data and the local data. The external data sources are IoT platforms whereas local data sources are devices located in the room where we want handle the temperature.

In this architecture layer, we convert the concrete values of temperatures and humidity in fuzzy values using the fuzzy sets and linguistic variables that it saw in Section 3.1.

| | | Outdoor temperature | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **AND** | **Extremely Low** | **Very Low** | **Low** | **Normal** | **High** | **Very High** | **Extremely High** |
| **Outdoor humidity** | **Very Low** | Extremely Low | Very Low | Very Low | Normal | Normal | High | Extremely High |
| | **Low** | Extremely Low | Very Low | Low | Normal | High | Very High | Extremely High |
| | **Normal** | Extremely Low | Very Low | Low | Normal | Very High | Extremely High | Extremely High |
| | **High** | Very Low | Low | Low | Normal | Very High | Extremely High | Extremely High |
| | **Very High** | Low | Low | Normal | Normal | Extremely High | Extremely High | Extremely High |

Table 1: Fuzzy rules used to calculate the degree of truth of the possible apparent temperature.

The external processor has fuzzy rules defined to be applied over the fuzzy values previously obtained that represent the outdoor temperature and humidity. These fuzzy rules are shown in Table 1. With this fuzzy rules we obtain the degree of truth of each possible linguistic variable for the apparent temperature.

After applying the fuzzy rules and getting the centre of gravity we obtain a value for apparent temperature. With this value and the indoor temperature, the system is able to make decisions about what action to do in order to handle the rooms temperature. For that, we apply another set of fuzzy rules, shown in Table 2, whose input are the apparent temperature and the indoor temperature and whose output is the degree of truth of the action to do.

The made decisions are send to the next layer in order to do the corresponding action with the corresponding actuator.

### *3.2.3. Actuators*

The last layer is composed by the actuators that are connected like the heating systems and the air condition systems, although more actuators could be available like fans, LEDs, thermostats, or any others.

The aim of this layer is to route the actions to the corresponding actuators. For example, if the decision made in the Data processor layer was switching on

| | | Aparent temperature | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **AND** | **Extremely Low** | **Very Low** | **Low** | **Normal** | **High** | **Very High** | **Extremely High** |
| **Indoor temperature** | **Extremely Low** | Heating sys. at max. power | Heating sys. at max. power | Heating sys. at max. power | Heating sys. at max. power | Heating sys. at max. power | Heating sys. at max. power | Heating sys. at max. power |
| | **Very Low** | Heating sys. at max. power | Heating sys. at max. power | Heating sys. at max. power | Heating sys. at max. power | Heating sys. at max. power | Heating sys. at max. power | Heating sys. at max. power |
| | **Low** | Heating sys. at max. power | Heating sys. at normal power | Heating sys. at max. power | Heating sys. at normal power | Heating sys. at normal power | Heating sys. at normal power | Heating sys. at normal power |
| | **Normal** | Heating sys. at normal power | Heating sys. at normal power | Heating sys. at normal power | No system running | No system running | No system running | Air condition sys. at normal power |
| | **High** | No system running | Air condition sys. at normal power | No system running | No system running | Air condition sys. at normal power | Air condition sys. at normal power | Air condition sys. at max. power |
| | **Very High** | Air condition sys. at normal power | Air condition sys. at normal power | Air condition sys. at normal power | Air condition sys. at normal power | Air condition sys. at max. power | Air condition sys. at max. power | Air condition sys. at max. power |
| | **Extremely High** | Air condition sys. at max. power | Air condition sys. at max. power | Air condition sys. at max. power | Air condition sys. at max. power | Air condition sys. at max. power | Air condition sys. at max. power | Air condition sys. at max. power |

Table 2: Fuzzy rules used to calculate the degree of truth of possible actions to do.

the heating system, the Actuators layer should have switched on the heating system.

## 4. Used software and hardware

In order to develop this solution proposed by this paper the use of different types of software and hardware components were required.

The developed systems language is Java 8 and it uses the libraries jFuzzyLogic[1] [35, 36], Pi4J 1.0[2], and Gson 2.3.1[3].

The used hardware was an Arduino UNO with a TMP36 temperature sensor and a Raspberry Pi 2 with several electronic components connected to its GPIOs:

- TMP36 temperature sensor.
- MCP3008 analogic-to-digital converter.
- 2 yellow LEDs, 2 red LEDs and 1 RGB LED.
- Several resistors.

[1] jFuzzyLogic: http://jfuzzylogic.sourceforge.net/html/index.html
[2] Pi4J: http://pi4j.com
[3] Gson: https://github.com/google/gson

## 5. Evaluation and discussion

This paper proposes a system to control the temperature in an automated and efficient way. Traditional thermostats control the temperature switching on and switching off the heating system or the air condition system when the temperature is higher or lower than a specific value which is set manually. For example, a thermostat could be set to switch on the heating system when the temperature is higher than 20°C. The problem is when the temperature is changing near 20°C (e.g., between 19°C and 21°C), because the system would switch on and switch off constantly. The use of fuzzy sets and linguistic variables allows us to determine ranges in which the temperature is suitable. Moreover, we are able to determine the best moment when heating systems and air condition systems have to be switched on and switched off taking the outdoor ambient conditions and using fuzzy logic.

In order to illustrate this, we made an evaluation of our approach where we compared the times that a heating system and an air condition system was switched on and switched off using a old thermostat with using our new fuzzy approach that also takes into account the outdoor conditions and the indoor temperature. In Table 3, it is shown the outdoor temperature, humidity, and the indoor temperature that we used to do the simulation in this evaluation.

With the outdoor temperature and humidity, we calculated the apparent temperature in order to use it with the indoor temperature to determine the best moment to switch on and to switch off the heating system and the air condition system. In Figure 6 we can see the three temperatures used in the evaluation. The blue curve is the outdoor temperature, the yellow curve is the apparent temperature calculated from outdoor temperature and humidity, and the orange curve is the temperature indoor during a full day without using any system to control the temperature. The apparent temperature and the outdoor temperature are different because the humidity was variable and the apparent temperature is calculated from outdoor temperature and humidity.

The old heating system that we simulate in this evaluation had two heating

| Time | Humidity | Temp. Outdoor | T. Indoor |
|---|---|---|---|
| 00:00 | 66 | 15 | 18 |
| 01:00 | 69 | 14 | 17 |
| 02:00 | 73 | 12 | 16 |
| 03:00 | 75 | 11 | 15 |
| 04:00 | 76 | 10 | 15 |
| 05:00 | 78 | 9 | 14 |
| 06:00 | 76 | 9 | 14 |
| 07:00 | 74 | 9 | 14 |
| 08:00 | 72 | 10 | 14 |
| 09:00 | 62 | 12 | 15 |
| 10:00 | 53 | 15 | 15 |
| 11:00 | 44 | 17 | 15 |
| 12:00 | 38 | 19 | 18 |
| 13:00 | 33 | 21 | 20 |
| 14:00 | 28 | 23 | 23 |
| 15:00 | 28 | 24 | 23 |
| 16:00 | 29 | 25 | 24 |
| 17:00 | 30 | 26 | 26 |
| 18:00 | 36 | 27 | 26 |
| 19:00 | 40 | 27 | 26 |
| 20:00 | 42 | 28 | 26 |
| 21:00 | 33 | 25 | 23 |
| 22:00 | 30 | 23 | 22 |
| 23:00 | 34 | 20 | 20 |

Table 3: Ambient conditions of a full day.

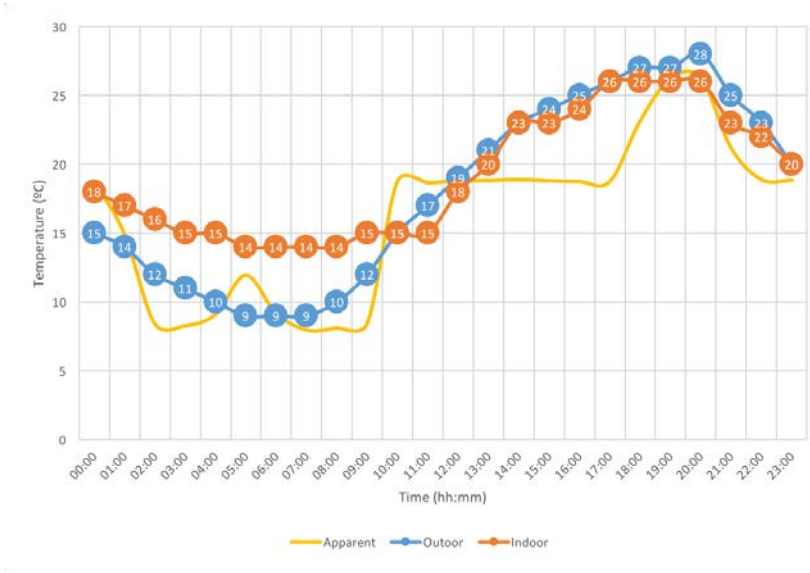



Figure 6: The obtained temperatures for the evaluation.

stages and the old air condition system had two airflow stages to control the temperature. The configuration used is the next:

- Heating system at maximum power: Below 15°C.
- Heating system at normal power: Between 15°C and 17 °C.
- Air condition system at maximum power: Above 25°C.
- Air condition system at normal power: Between 23°C and 25°C.
- Both systems switched off: Between 18°C and 22 °C.

The new fuzzy systems used another configuration based in ranges instead of single values. The used ranges are the same as the fuzzy sets shown in Section 3.1.

After configuring the systems with the obtained and calculated values we simulated what would have happened if these systems were used in the same conditions. In Figure 7, we can see when the systems switched on and when they switched off. The old heating system switched on one hour before our


Figure 7: Stages of each system for one day.

approach and it also switched off one hour later. Moreover, our approach for air condition system did not use the maximum power whereas the old system was at maximum power during 4 hours. Also, the new air condition system started working 4 hours later and stopped working 1 hour before.

Figure 7 shows how the stages changed during a day, nevertheless, the number of hours when the both systems were working is not clear enough. Figure 8 shows how many hours each system was switched on and switched off and we can see that our approach reduced the quantity of hours.

Firstly, the old heating system was on during 11 hours whereas the new heating system was on during 8 hours. There were 3 hours or approximately 27.27% of energy saving. And secondly, the old air condition system was on during 8 hours whereas the new air condition system was off during 21 hours so there were 5 hours or approximately 62.5% of energy saving. Therefore, we can assume that there was a certain saving energy.

These results must be interpreted carefully. In spite of defining a temperature ranges, the system did not switch on whenever the temperature was out of

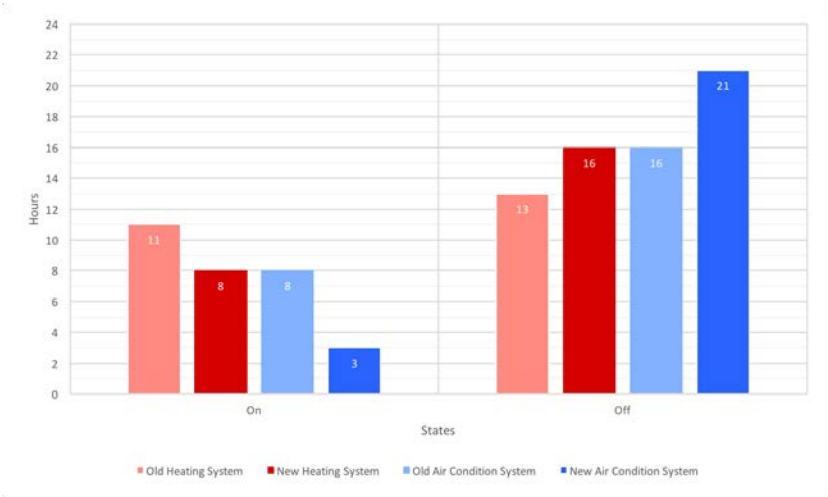



Figure 8: The number of hours per state.

range. Therefore, this approach may not be used in context of critical situations when the control of temperature cannot have margin of errors. The explanation of this behaviour could be the use of external conditions. Considering external conditions may be a good idea for the temperature control in rooms where there are people. It could be a way of adapting the room for people who are entering and leaving the room. However, there are a lot of situations that this approach could be used.

## 6. Conclusions

The future of computers is related with the IoT and each progress is very important for the development the technologies from the context of the IoT. In this paper we presented our approach about the integration of fuzzy logic into the context of the IoT.

We proposed the use of fuzzy logic in order to control the temperature of a specific room taking into consideration the outdoor conditions, specifically, the outdoor temperature and humidity to obtain the outdoor apparent temperature.

In order to obtain the outdoor data we required devices that analyse the

environment. These devices can be part of a Smart City or they can be part of a set of Smart nearby Cities.

With this data, the developed prototype simulated the control of the heating system and the air condition system so that they could be switched on and switched off considering the combination of outdoor conditions and indoor temperatures.

In the evaluation of our proposal, we realised that use fuzzy logic to help systems that control the temperature, allow us to save energy and thus, save money. For example, we achieve an energy saving of around 40% if we understand that time saving implies energy saving. However, the fuzzy logic causes that the range in which the systems are off was too broad. Therefore, this approach would not be valid in the context of critical situations where the temperature would be an accurate value due to the consideration of external conditions in addition to local conditions.

## 7. Future work

The aim of this research was to reach a first approach that combines IoT technologies and fuzzy logic in order to improve the the temperature control in specific locations, and from here, there are much work to do in future researches.

- **Improving the fuzzy logic configuration**: The configuration used in this paper is based in our experience during the research progress. Improving this configuration would allow getting better results, saving more energy and reaching a better temperature control.
- **Saving old temperatures in order to know if the temperature is rising or it is decreasing**: The actual approach does not take into consideration indoor temperature history. Knowing the previous temperatures help the system to make better decisions. The system behaviour may not be the same if the temperature is rising or if it is decreasing. Taking into consideration the the temperature progress could help avoid changing the state constantly.

- **Publishing the data in IoT platforms**: In this research we obtained the outdoor apparent temperature combining the outdoor temperature and humidity. The apparent temperature is a valued data and publishing it could be a good contribution.

- **Adding more sensors**: In this paper we used three values, the outdoor temperature, the outdoor humidity, and the indoor temperature. Taking more values would allow us to improve the calculation of the apparent temperature. Moreover, we also may calculate the apparent indoor temperature which depends not only on humidity and temperature but also on the quantity of people in the room and many other factors.

- **Creating a Domain-Specific Language in order to personalise the rules and the fuzzy sets in an easy way**: The fuzzy logic configuration depends on the location and on the people who is going to use the system, thus, adding a way of personalising the configuration for people who does not have knowledge about programming is a great improvement of this research. Developing a Domain-Specific Language (DSL) is the best solution to reach this proposal. A DSL may allow any user with minimum knowledge about computers, to personalise the system configuration.

## 8. Acknowledgements

This work was performed by the 'Ingeniería Dirigida por Modelos MDE-RG' research group at the University of Oviedo under Contract No. FC-15-GRUPIN14-084 of the research project 'Ingenierá Dirigida Por Modelos MDE-RG'. Project financed by PR Proyecto Plan Regional.

## References

[1] H. G. H. Gu, D. W. D. Wang, A content-aware fridge based on rfid in smart home for home-healthcare, in: 2009 11th International Conference on Advanced Communication Technology, Vol. 02, 2009, pp. 987–990.

[2] K. a. Hribernik, Z. Ghrairi, C. Hans, K.-D. Thoben, Co-creating the internet of things - first experiences in the participatory design of intelligent products with arduino, in: 2011 17th International Conference on Concurrent Enterprising, 2011, pp. 1–9.

[3] L. Hao, X. Lei, Z. Yan, Y. ChunLi, The application and implementation research of smart city in china, in: 2012 International Conference on System Science and Engineering (ICSSE), 2012, pp. 288–292. doi:10.1109/ICSSE.2012.6257192.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6257192

[4] R. Lea, M. Blackstock, Smart cities: An iot-centric approach, in: Proceedings of the 2014 International Workshop on Web Intelligence and Smart Sensing, IWWISS '14, ACM, New York, NY, USA, 2014, pp. 12:1—-12:2. doi:10.1145/2637064.2637096.
URL http://doi.acm.org/10.1145/2637064.2637096

[5] M. Nikravesh, Evolution of fuzzy logic: From intelligent systems and computation to human mind, Studies in Fuzziness and Soft Computing 217 (2) (2007) 37–53. doi:10.1007/978-3-540-73182-5_3.
URL http://dx.doi.org/10.1007/s00500-007-0192-9

[6] G. Cueva-Fernandez, J. Pascual Espada, V. García-Díaz, R. Gonzalez-Crespo, Fuzzy decision method to improve the information exchange in a vehicle sensor tracking system, Applied Soft Computingdoi:10.1016/j.asoc.2015.01.066.
URL http://www.sciencedirect.com/science/article/pii/S1568494615000873

[7] A. Kylili, P. A. Fokaides, European smart cities: The role of zero energy buildings, Sustainable Cities and Society 15 (2015) 86–95. doi:10.1016/j.scs.2014.12.003.

URL http://www.sciencedirect.com/science/article/pii/S2210670714001383

[8] N. Bulusu, D. Estrin, L. Girod, Scalable coordination for wireless sensor networks: self-configuring localization systems, in: Proc. of the 6th International Symposium on Communication Theory and Applications(ISCTA 01), Ambleside, UK, no. July, 2001, pp. 1–6.
URL http://gicl.cs.drexel.edu/people/regli/Classes/CS680/Papers/SensorNets/iscta-2001.pdf

[9] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Communications Magazine 40 (8) (2002) 102–105. doi:10.1109/MCOM.2002.1024422.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1024422

[10] P. Grant, A new approach to diabetic control: Fuzzy logic and insulin pump technology, Medical Engineering & Physics 29 (7) (2007) 824–827. doi:http://dx.doi.org/10.1016/j.medengphy.2006.08.014.
URL http://www.sciencedirect.com/science/article/pii/S1350453306001834

[11] S. S. Bhunia, S. K. Dhar, N. Mukherjee, ihealth: A fuzzy approach for provisioning intelligent health-care system in smart city, in: 2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2014, pp. 187–193. doi:10.1109/WiMOB.2014.6962169.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6962169

[12] R. Y. Chen, Intelligent iot-based tracing system for backward design using fcm and fuzzy rule, in: 2013 Fourth Global Congress on Intelligent Systems, 2013, pp. 229–233. doi:10.1109/GCIS.2013.43.

URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6805940

[13] M. Maksimović, V. Vujović, B. Perišić, V. Milošević, Developing a fuzzy logic based system for monitoring and early detection of residential fire based on thermistor sensors, Computer Science and Information Systems (00) (2014) 90.

[14] Z. Zinonos, C. Chrysostomou, V. Vassiliou, Wireless sensor networks mobility management using fuzzy logic, Ad Hoc Networks 16 (0) (2014) 70–87. doi:10.1016/j.adhoc.2013.12.003.
URL http://www.sciencedirect.com/science/article/pii/S1570870513002850

[15] L. Zadeh, Fuzzy sets, Inf and Control 8 (1965) 338–353. doi:10.1109/2.53.

[16] L. A. Zadeh, Fuzzy logic and approximate reasoning, Synthese 30 (3-4) (1975) 407–428. doi:10.1007/BF00485052.
URL http://dx.doi.org/10.1007/BF00485052

[17] D. F. Larios, J. Barbancho, F. J. Molina, C. León, Lis: Localization based on an intelligent distributed fuzzy system applied to a wsn, Ad Hoc Networks 10 (3) (2012) 604–622. doi:10.1016/j.adhoc.2011.11.003.
URL http://www.sciencedirect.com/science/article/pii/S1570870511002095

[18] I. Chamodrakas, D. Martakos, A utility-based fuzzy topsis method for energy efficient network selection in heterogeneous wireless networks, Applied Soft Computing Journal 12 (7) (2012) 1929–1938. doi:10.1016/j.asoc.2012.04.016.
URL http://www.sciencedirect.com/science/article/pii/S1568494612001998

[19] S. Bagchi, A fuzzy algorithm for dynamically adaptive multimedia streaming, ACM Transactions on Multimedia Computing, Communications, and Applications 7 (2) (2011) 1–26. doi:10.1145/1925101.1925106.
URL http://dl.acm.org/citation.cfm?id=1925101.1925106

[20] G. Cueva-Fernandez, J. P. Espada, V. García-Díaz, R. G. Crespo, N. Garcia-Fernandez, Fuzzy system to adapt web voice interfaces dynamically in a vehicle sensor tracking application definition, Soft Computingdoi:10.1007/s00500-015-1709-2.
URL http://link.springer.com/10.1007/s00500-015-1709-2

[21] S. Russell, P. Norvig, Artificial intelligence: a modern approach.

[22] E. Portmann, A. Andrushevich, R. Kistler, A. Klapproth, Prometheus - fuzzy information retrieval for semantic homes and environments, in: 3rd International Conference on Human System Interaction, HSI'2010 - Conference Proceedings, 2010, pp. 757–762. doi:10.1109/HSI.2010.5514482.

[23] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Computer Networks 54 (15) (2010) 2787–2805. doi:10.1016/j.comnet.2010.05.010.
URL http://www.sciencedirect.com/science/article/pii/S1389128610001568

[24] K. Gama, L. Touseau, D. Donsez, Combining heterogeneous service technologies for building an internet of things middleware, Computer Communications 35 (4) (2012) 405–417. doi:10.1016/j.comcom.2011.11.003.
URL http://www.sciencedirect.com/science/article/pii/S0140366411003586

[25] Lu Tan, Neng Wang, Future internet: The internet of things, in: 2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE), Vol. 5, IEEE, 2010, pp. V5–376–V5–380. doi:10.1109/ICACTE.2010.5579543.

URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5579543

[26] National Intelligence Council, Discruptive civil tehnologies - six tehnologies with potential impacts on us interests out to 2025 (2008).
URL http://www.dni.gov./nic/confreports_discruptive_tech.html

[27] C. González García, B. C. Pelayo G-Bustelo, J. Pascual Espada, G. Cueva-Fernandez, Midgar: Generation of heterogeneous objects interconnecting applications. a domain specific language proposal for internet of things scenarios, Computer Networks 64 (2014) 143–158. doi:10.1016/j.comnet.2014.02.010.

[28] C. G. García, J. P. Espada, E. R. Núñez valdez, V. García-Díaz, Midgar : Domain-specific language to generate smart objects for an internet of things platform.

[29] Xively by logmeinxively.
URL https://xively.com/

[30] Internet of things - thingspeak.
URL https://www.thingspeak.com/

[31] Paraimpu: Connect, compose, share. the social tool for the internet of things.
URL https://www.paraimpu.com/

[32] Loxone smart home. easily controlled. intelligently automated. — loxone.
URL http://www.loxone.com/enen/start.html

[33] For heating & air conditioning intelligent climate control — tado.
URL https://www.tado.com/gb/

[34] G. Cueva-Fernandez, J. P. Espada, V. García-Díaz, C. G. García, N. Garcia-Fernandez, Vitruvius: An expert system for vehicle sensor tracking and managing application generation, Journal of Network and Computer Applications 42 (2014) 178–188. doi:10.1016/j.jnca.2014.02.

013.

URL http://dx.doi.org/10.1016/j.jnca.2014.02.013

[35] P. Cingolani, J. Alcalá-Fdez, jfuzzylogic: a java library to design fuzzy logic controllers according to the standard for fuzzy control programming (2013) 61–75.

[36] P. Cingolani, J. Alcala-Fdez, jfuzzylogic: a robust and flexible fuzzy-logic inference system language implementation, in: Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on, IEEE, 2012, pp. 1–8.

# ANEXO V. Midgar: Creation of a graphic Domain-Specific Language to generate Smart Objects for the Internet of Things scenarios using Model-Driving Engineering

# Midgar: Creation of a graphic Domain-Specific Language to generate Smart Objects for the Internet of Things scenarios using Model-Driving Engineering

Cristian González García[a*], Vicente García-Díaz[a], Daniel Meana-Llorián[a], B. Cristina Pelayo G-Bustelo[a], Juan Manuel Cueva Lovelle[a]

[a] *University of Oviedo, Department of Computer Science, Sciences Building, C/Calvo Sotelo s/n 33007, Oviedo, Asturias, Spain. Tel: +34985103397*

[a] gonzalezgarciacristian@hotmail.com, *garciavicente@uniovi.es*, *danielmeanallorian@gmail.com*, *crispelayo@uniovi.es*, *cueva@uniovi.es*

* Corresponding author.

*Abstract*— Currently, we have around us many Smart Objects. With the use of these objects, we can obtain benefits in our daily life, recommendations and help when we travel, or we can increment and improve our industrial processes through the automation of certain task. Notwithstanding, we need to use a specific software or we need to develop our own applications. Nevertheless, the principal problem is when we need to develop our own application because we need to save money or the existing applications do not adapt to us. In this case, maybe we need to learn new things, we will spend money, and such a process is likely to have problems related to the Software Crisis term. Here the world has the problem and we have a hypothesis: Could we facilitate the object creation in an easy and fast way for people? As a possible solution, we have developed a graphic Domain-Specific Language using the Midgar platform. To validate our proposal we did a comparative with two graphic editors and then we did a quantitative and a qualitative evaluation to obtain the necessary data to demonstrate if we accomplished our hypothesis.

***Industrial Internet of Things; Internet of Things; Model-Driven Engineering; Domain-Specific Language; Smart Objects;***

## I. INTRODUCTION

The Internet of Things (IoT) is the interconnection of heterogeneous and ubiquitous objects between themselves [1–4]. Currently, people have things to make an IoT network. People's daily life have a lot of objects with Internet connection like smartphones [5], tablets, Smart TVs, micro-controllers [6,7], Smart Tags [3], computers, laptops, cars, cheaper sensors, and the improved wireless connections [8]. With these things, people have heterogeneous objects because these are of different type and people have ubiquitous objects because objects are installed in different places and some of these objects can be move round the world. If we think in this, we could already think that we have the Internet of Things.

With IoT we can create an enormous network to interconnect objects and facilitate our daily life. Some examples: to improve the tracking of deliveries, objects' situation, to improve the factory's production, or security in industry [9–11]; to prevent natural disasters, to automate actions in farms, or to obtain data about the ecosystem in order to protect the fauna and flora according to certain situations, this is known as Smart Earth [12–14]; the improvement of Smart Cities to help citizens in their daily life, for instance, to park, to avoid traffic jams, to automate traffic lights, to change the public transport in real time or so on [15–17]; to improve the people life with Smart Homes using of a Smart Fridge to help ill people or facilitating the garden maintenance [18–20]. However, people need to create the application or buy it. The ideal world would be if each person could put their objects in the IoT with no problems, with only a minimum knowledge about IoT and the object that they use. Thus, we could have a lot of sensors and data to create a huge IoT for our City or our World.

Nevertheless, people depend on the important and big companies, which are developing new IoT systems. The companies' products depend on the object, maybe you must use a specific software and hardware. For example, LG, Samsung, Telefónica, and other companies offer packs for creating an IoT network in your own home. Nonetheless, these solutions only allow interconnecting objects with other objects of the same brand. For example, if you use an object of a determinate company, you only can use the mobile application that this company provides to interact with its objects. Another example is when you use a server and you only can interconnect with this server objects of the same server's brand. This breaks the heterogeneity of the Internet of Things because companies' objects do not use the same language to talk to each other. Each company uses in his IoT products its own language. This problem obliges to use only the same product brand to create an ecosystem. In this case, maybe people cannot use the better sensor of another brand to create an IoT system for his company, or house because then people should need to change all the system or people should use both systems. Maybe, they should use both because for example each system has the better sensor in different areas. This problem impedes the evolution of the

IoT because it limits the evolution according to companies' progress. In this case, if companies improve their systems, their sensors, their IoT servers, their products, thereafter the IoT will evolves. Nevertheless, all the evolution depends on the software and hardware that companies provide to us.
Then, our hypotheses were: can we apply Model-Driven Engineering to obtain an abstraction and create a DSL to facilitate the object creation to people who know how to create and manage an object but who do not know to programme? Will the object creation be a significant improvement in the IoT?
For solve our two hypotheses, a possible solution is to allow creating and interconnecting different objects between themselves. We proposed a solution based on a graphic Domain-Specific Language (DSL) called **Midgar Object Interconnection Specific Language** (MOISL) to solve the interconnection problem between heterogeneous and ubiquitous objects in [21] using the Midgar IoT platform. With this solution, people without development knowledge can interconnect IoT network's objects. However, this is only a partial solution for the problem that we have presented because it does not solve the problem of generation applications. In our first proposal, people needed to have a registered application in the IoT platform, which have to interpret specific messages. In this article, we propose a possible solution to create the necessary software for the objects. We propose a graphic DSL called **Midgar Object Creation Specific Language** (MOCSL). In this proposal, we want to offer a graphic DSL to facilitate the object creation to people without development knowledge. In this way, people only need a smartphone and know their smartphone brand and/or an Arduino and know how to connect sensors and actuators to their Arduino. In the Arduino case, people does not need how to create the source code to use the temperature sensor, the flame sensor, or the servomotor. People only need how to connect sensor and actuator in an Arduino.
For that, we have researched to offer this abstraction in order to facilitate the interconnection between the real world and the virtual world with no develop source code. With this, we want to allow a quicker and easier way for the object creation for improving the implementation of the IoT in our daily life, for example, in our industry, our home, our education, our security, or in many other situations.
The remainder of this paper is structured as following: In section II, we explain the state of the art where we discuss about the Internet of Things, Smart Objects, Model-Driven Engineering, Domain-Specific Languages, the more important current IoT platforms and we present our research on applications that generate objects. In section III we explain the proposal. We present the idea about generation of objects for the Internet of Things platform and the proposal architecture. Section IV contains the two evaluations and our discussion. We will explain the methodology and the results. Section V has our conclusions about this research. Finally, in the section VI we describe some possible options as future work of this proposal.

## II. STATE OF THE ART

### *A. Internet of Things*

In recent years, we could see the growth of the interaction among heterogeneous and ubiquitous objects with each other. It is known as the Internet of Things (IoT) [1,3,21,22,20,23]. This was announced by the National Intelligence Council of United States in [24] and by the United Nations [1]. The origin was the necessity of the supply chains and the identification of objects, people and animal through the use of RFID intelligent tags [1,4,25]. Then, with the IoT we can identify the different objects of the world to make interact the real world with the virtual world. It offers to us a lot of possibilities as we can see in many researches about Smart Earth [16], Smart Home [2,26,27], Smart Cities [16] and others. All of them facilitate the life of humans with different automations and notifications. Nevertheless, the Internet of Things requires an integrated intelligence, connectivity and interaction [3]. For that, we have different Internet of Things platforms to interconnect our objects.
However, we have a problem when we work with the Internet of Things. We must to develop the software to make to function Smart Objects. After that, we have to do the interconnection the different objects between themselves.

### *B. Smart Objects*

Smart Objects, also known as Intelligent Products, are physical elements with different properties, which are identifiable during their useful life. They can interact with other Smart Objects, with the environment, or under certain conditions with autonomous behaviour [2,9]. They can obtain different environmental data like the humidity, location, temperature, gravity, acceleration, and so on. Besides, they have various actuators to do different things when they obtain a determinate datum from other object, the environmental or through and intelligent network like an IoT platform. For example, they could vibrate, show a notification, make a call, or send messages if it is a smartphone, tablets or similar. Other possibilities are when they could turn on/turn off the sound, accelerate/decelerate an engine or a fan, and anything that you can create with a microcontroller like an Arduino.
Nevertheless, users have to create the software to obtain the object's data, have to create the interconnection between different Smart Objects, or use a specific software with a specific hardware although in this case, users lost the possibility of interconnecting heterogeneous objects.

### C. *Model-Driven Engineering*

Software engineering continually offers new methods and tools to improve the software development process. Although the use of modelling standards such as the Unified Modeling Language [28] is a commonly accepted practice, in recent years it is gaining strength the use of a relatively new approach for development called Model-Driven Engineering (MDE) [29], based entirely on the use of models as first class artefacts for development. With that, models like those provided by the UML standard are gaining importance. Thus, the Model-Driven Architecture (MDA) [30] standard has appeared as a standardised way of conducting MDE from UML models and metamodels. The use of models, with common underlying technologies, makes it to increase the portability, reusability, and interoperability of all software that is built. Thus, technologies such as those included in the Eclipse Modeling Project [31] offering a complete toolset for working with models to build any type of software has arisen.

For this proposal, we used MDE to study the problem of obtaining an abstraction of it. With this, we could obtain a complete vision to create a Domain-Specific Language to try to solve our hypotheses.

### D. *Domain-Specific Languages*

From the use of standardised models as a basis for the development of software, the Domain-Specific Languages (DSLs) [32] have also gained in popularity. Thus, very powerful tools such as Xtext [33] are currently been widely used. Xtext, for example, is a framework that focuses on the creation of DSLs using models as the code technology. The main idea is to define small programming languages tailored to a specific domain of knowledge, avoiding technological concepts and focusing on keywords and constructs related to the specific domain being dealt with [34]. Thus, common General-Purpose Languages (GPLs) such as C++, Java, or C# use many keywords (e.g., static, protected, double, import, etc.) that have nothing to do with the final purpose of the applications that are being defined. A DSLs can be designed to only use specific domain concepts, allowing that experts in a field can use it to create solutions, even without the high technical knowledge that GPLs require [35]. DSLs can be interpreted or compiled like any other language, but most often, the code is benefit from the reuse of model-based technologies and with standardised tools based on a common root metamodel (MOF/Ecore) [36]. Thus, programs are translated into well stablished languages to be interpreted by their virtual machines and compilers.

Then, we have created a graphic DSL (MOCSL) in order to allow the object creation for people without development knowledge or people who need to create objects in an easy and fast way. We have design MOCSL using Ecore as a Meta-metamodel and we created the editor using JavaScript.

### E. *Internet of Thing Platforms*

Currently, we can choose between different Internet of Things platforms to interconnect our objects. However, they are very different in their final proposal and in how users have to design the interconnections and register their objects. For that, we can classify the IoT platforms in four groups [22]:

- Business platforms: Xively [37], Exosite [38], SensorCloud [39], Etherios [40], ThingWorx [41], and Carriots [42].
- Research platforms: Midgar [21], Paraimpu [43], QuadraSpace [44], and SIoT [45].
- Platforms in beta state: ThingSpeak [46], Sensorpedia [47,48], SenseWeb [49,50], Evrythng [51], and Open.Sen.se [52].
- Open Source platforms: Nimbits [53] and Kaa [54].

All these IoT platforms are great and all have different important qualities to improve the IoT. Nonetheless, these IoT platforms require a minimum of knowledge about development because all of them need that users must create the object's software, the interconnection between the object and the platform, or both. Sometimes, the IoT platforms help users with an Application Programming Interface (API) to facilitate the interconnection with the IoT platform and/or to read sensor data in different Smart Object or micro-controllers like Android smartphones, Arduino, Raspberry Pi, and so on. In this case, the Iot platforms reduce the time and users' effort for developing applications but users still need knowledge about software development and users still need to develop the program.

One problem in the IoT is the necessity of developing applications to interconnect Smart Objects between themselves. Some IoT platforms help to interconnect Smart Objects through a DSL or form. For this problem, we proposed a possible solution with MOISL to improve this problem in [21].

Another problem is when users want to use a Smart Object or a micro-controller but users do not have knowledge about how to develop an application. Then, users have some possibilities. They could learn to programme, but, for this, they need a lot of time or they could buy a specific application for their own Smart Object. For this problem, a possible solution is to use a similar software used for the previous problem but the IoT platforms do not have this option. However, Arduino community and BQ offer a software to facilitate this task but with some restrictions. We will explain more about this in the next section. For that, we propose in this article a new DSL in the Midgar Platform to facilitate the generation of Smart Objects and micro-controller for the IoT scenarios.

### *F. Applications that generate software to manage Smart Objects*

As we said before, one problem of the Internet of Things is the necessity to know how to develop an application to interconnect your object. Currently, you could have a smartphone or a micro-controller. For this, we need to develop different kind of applications in different programming languages. Then, if you want to interconnect heterogeneous objects, you need to know how to develop an application and how to programme in different programming languages. A problem is about the programming because it is complex and difficult [55]. However, we can find some solutions to facilitate this task for people without development knowledge. However, these solutions include the possibility of working with data through the flow modification inside the object. We did not include this functionality, because we have this functionality with MOISL, in the Midgar platform, because we search the object creation for the Internet of Things. In this section, we are going to talk about these solutions.

#### *1) Solutions for smartphones*

In this case, we have studied various graphic web editors. To create applications for Android, we have AppsGeyser [56], iBuildApp [57], and Andromo [58]. To create applications for Android, iOS, and HTML, we have AppsBuilder [59] and Infinite Monkeys [60]. If we also want support Blackberry and Windows Phone, we can use Appypie [61]. Another application to generate applications for Android and iOS is Como [62] or if we also want support Windows Phone we have AppMachine [63].

All these graphic web editors allow us to create applications for different smartphone operating system. Furthermore, they offer solutions with the drag & drop system with different blocks to drop the necessary functionality type. Others even offer the use of default templates with a minimum functionality to do easier the creation. However, they do not allow incorporating the use of sensors like accelerometer, pressure, and others, in their application. The only sensor, which they offer, is the GPS. This is a problem when users want to create an application for the IoT to interconnect their smartphones. For that, we have designed a graphic DSL, which supports the application creation for smartphones as a possible solution to improve the application creation for the IoT.

#### *2) Bitbloq*

Bitbloq [64] is a graphic web editor of the BQ company to create applications for its electronic devices like micro-controllers. This graphic web editor is a web application. It has a jigsaw puzzles type system to create the device's life cycle. The first piece on the top left corner is the first instruction that the graphic web editor create when translate the pieces to source code. The rest of the program translates all pieces following the next priority rule: right to left and up to down. You can see the source code in real time meanwhile you are building the application if you select the corresponding tab.

The pieces have different tabs and blanks according to the piece type. For example, sensor pieces have the pin blank on the right to link a pin piece, the normal blank up to link another piece before, the normal tab down to link another piece after, and a blank on the left to use this piece to continue a sequence or to link this piece in a communication piece like the Universal Serial Bus (USB) or Bluetooth.

This graphic web editor has some sensor and actuator pieces; pieces for creating a flow like ‘if’ or ‘while’; logic pieces like ‘true’, ‘false’, or conditions; other pieces to insert numbers, arrays , or operations; text pieces, communications pieces to print data on the USB, or Bluetooth; pieces to create variables and work with them.

Bitbloq uses a system based on Scratch system and it is a very good idea to create a graphic web editor because Scratch [65] is very often used in schools to learn how to programme for students [66,55]. For that, this is a very important proposal for learning our new generations about how to programme. Nevertheless, we propose a new graphic DSL, which avoids the communication, variables, and many pieces to obtain an easier and quicker abstraction for people without development knowledge who need to create an application and need to create the code to their micro-controller.

#### *3) Minibloq*

MiniBloq v. 0.83 [67] is an open source graphic editor for Multiplo and Arduino. This graphic editor is a desktop application. It has a jigsaw puzzles type system to create the device's life cycle. In addition, it creates the source code in real time and it allows uploading the source code to the micro-controller.

However, in this case, the pieces do not have free movement. For example, if you want to add a piece to improve the basic piece, you need to add this piece from the piece's menu instead general menu. However, if you want to start the next instruction, you must use the general menu. There is also has the opposite case: it has the same function under different pieces with different icon, for example, to use the digital pins from the general menu with the double arrow piece or from the USB with the digital pin piece. For example, MiniBloq offers less free to use the pins because these depends on the piece type. Another problem is that it includes by default the header files ‘mbq.h’ and ‘PingIRReceiver.h’ including if you do not need their modules. In addition, to repeat a task, you need to insert a loop because all the source code that it generates is in the ‘setup method’ instead the ‘loop method’. This is wrong because in Arduino, the ‘setup method’ is for establishing the different variables and pins and the ‘loop method’ is for creating our repetitive task. Besides, MiniBloq need variables in the same way to BitBloq.

This graphic editor facilitates the application generation but it has some problem in the graphic editor with its menus, icons and actions and it has no a good source code generation. Then, if we need a critical application for industrial area, we could have some problems with the source code of this graphic editor. In our proposal, we create a more abstract graphic DSL to facilitate the application generation. Furthermore, our graphic DSL only creates the necessary source code because it only imports the code that you need.

## III. CASE STUDY: MIDGAR

Midgar is an Internet of Things platform developed to research the different problems of the IoT. Currently, Midgar offers support to create the interconnection between heterogeneous and ubiquitous objects using the MOISL [21]. In this paper, we try to find a solution for the object generation in order to facilitate the creation of Smart Objects. In this section, we will describe the Midgar Platform's architecture, which includes the new graphic DSL: MOCSL.

### *A. Generation of applications for Smart Objects*

With **Midgar Object Creation Specific Language** (MOCSL), users will be able to create the necessary logic for their objects without writing source code using a drag & drop system. Users only need to choose the sensors and actuators that they want to use in their smartphone or Arduino. After, users will obtain the application for their smartphone or Arduino. With MOCSL, users have two option: creating the application for their smartphone or for their micro-controller. On the one hand, users only need to select the sensors and actuators that they want to use and allow in their smartphone. On the other hand, users need to assemble the required sensors and actuators on their real micro-controller and define in MOCSL the sensors and actuators that they want to use. Afterward, they obtain an application to connect the micro-controller with the computer's USB port and the application that they need to upload to the micro-controller. The computer application supports the Midgar's functionality. For this, if users want to interconnect objects, they need to register and create the interconnection using MOISL as we have described in [21].

### *B. Midgar's Architecture for MOCSL*

We have included a new part in Midgar platform to implement the new graphic DSL for object creation, MOCSL. The system's architecture has three layers, as it can be seen in Figure 1: **Process Definition**, **Object Generation**, and the **Objects**. Each one is a process within the global set of the infrastructure. Firstly, the **Process Definition** includes the user's process. In this layer, users (Figure 1.1) must define their application using **MOCSL**. For that, we offer a HTML5 application. When the user finalises the definition of their application (Figure 1.2), the model that the user created with **MOCSL** is serialised in an eXtensible Markup Language (XML) file. The **serialised model** contains all the information about the user's application: the IoT networtk that he chose, the application name, the select device, and the exactly device type. Afterwards, the second layer, **Object Generation**, receives the **serialised model** and processes it in the **Processor** (Figure 1.3). Then, the **Processor** processes the information and creates and compiles the **Generated Program** (Figure 1.4). The **Generated Program** has all the functionality that the user had defined using **MOCSL** and the necessary logic to interconnect the objects with **Midgar**. When user has the **Generated Program**, he only needs to upload the program on his device (Figure 1.5).

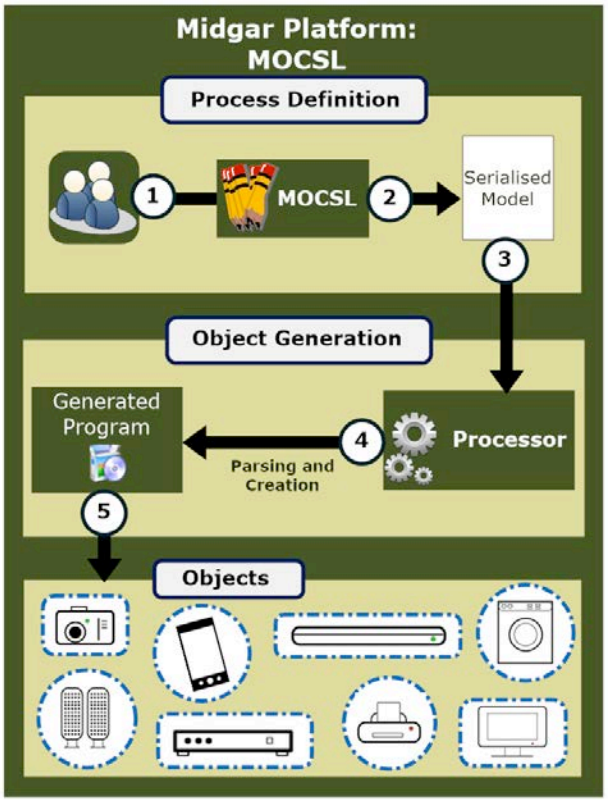



**Figure 1 Architecture of the Midgar Platform using MOCSL**

*C. Implementation*

In this sub-section, we are going to describe the new components and layers of Midgar's platform and their interaction. We are going to explain the created DSL, MOCSL. Afterwards, we are going to explain how Midgar processed the serialised model which is created by users using MOCSL when users create an application and how the **processor** parses it. After, we are going to talk about the Android and Arduino applications.

*1) Midgar's Domain Specific Language*

We have developed MOCSL using the HTML5's canvas element. MOCSL shows all the configurable options to users. MOCSL can be divided in four different areas as we can see in Figure 2.

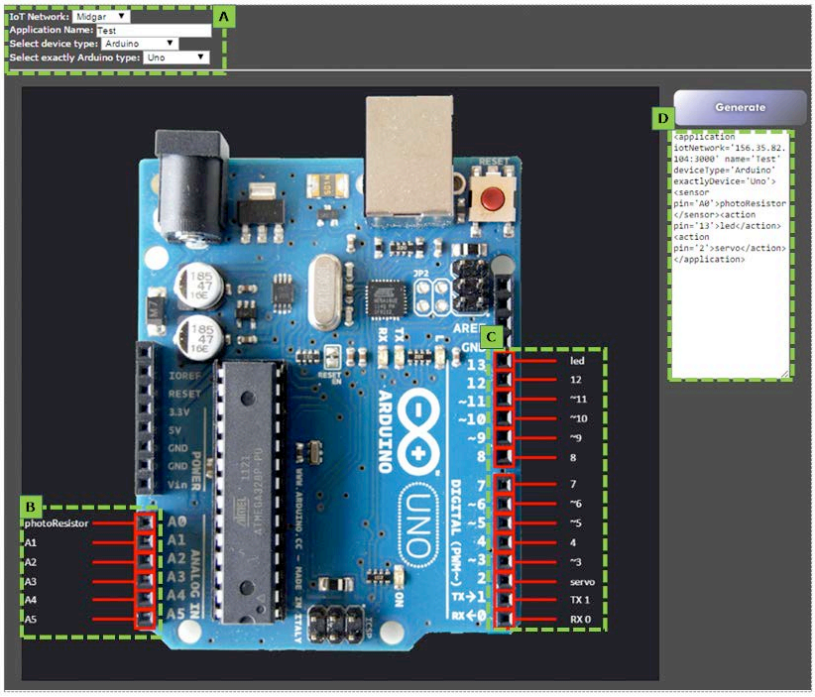



**Figure 2 MOCSL for the object creation of the Midgar platform for Arduino**

The first area (Figure 2A) contains the application data about the IoT network to use, the application name, the device type like smartphone or Arduino, and the exactly device type like Nexus 4, Arduino Uno, and so on. Depending of the 'device type', the next combo box can contain only smartphones or only micro-controllers. Then, when a user selects the 'exactly type' in this combo box, the canvas show the exactly element. In our example, the Arduino Uno. In Figure 2B the user can select the analogue pins and in Figure 2C the user has the digital pins. On the right, the user has the serialised model which represents the object that the user defined (Figure 2D). When a user selects a pin, he receives a popup with different sensors and actuators as we can see in Figure 3. In this popup, the user can select a sensor or actuator for the pin that he selected. If user selects 'cancel', the pin will return to its default state.

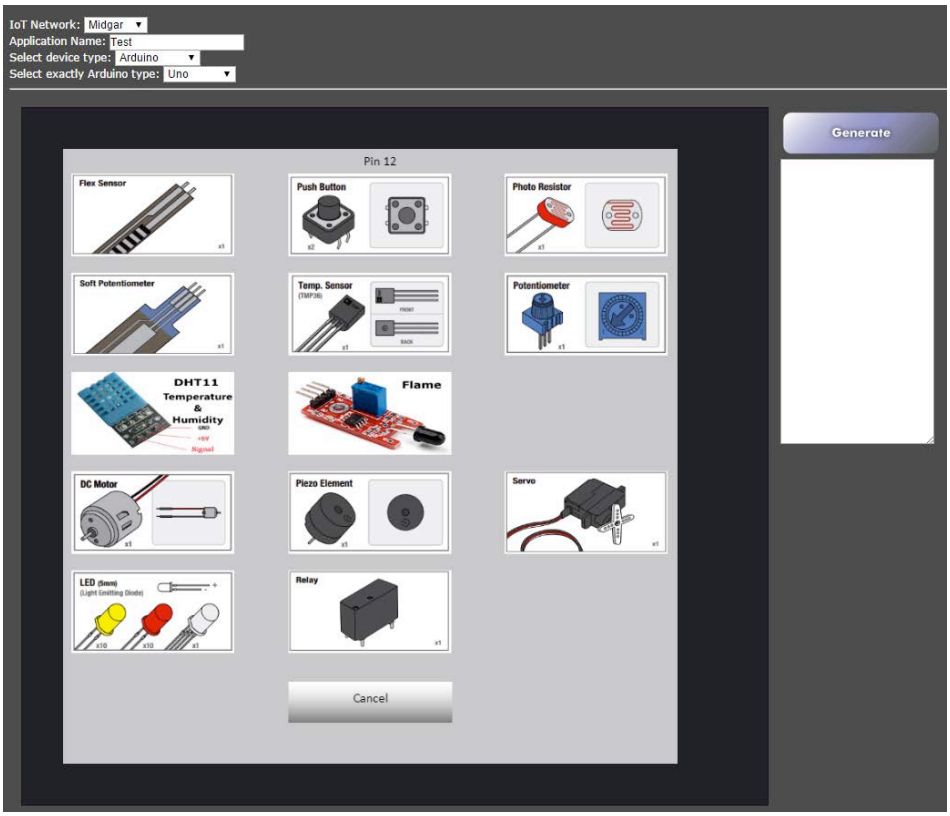



**Figure 3 Arduino's sensors and actions in the MOCSL's selector**

In Figure 4 we show an example of a smartphone, exactly, a Nexus 4. The user sees the different Nexus 4's sensors (green tick and red cross) and the sensors which with no support by their version or sensors which does not have in their smartphone (red dash). The user only needs to click on the sensor that he wants for changing the state between green tick if he wants it or for the red cross if he does not want it.

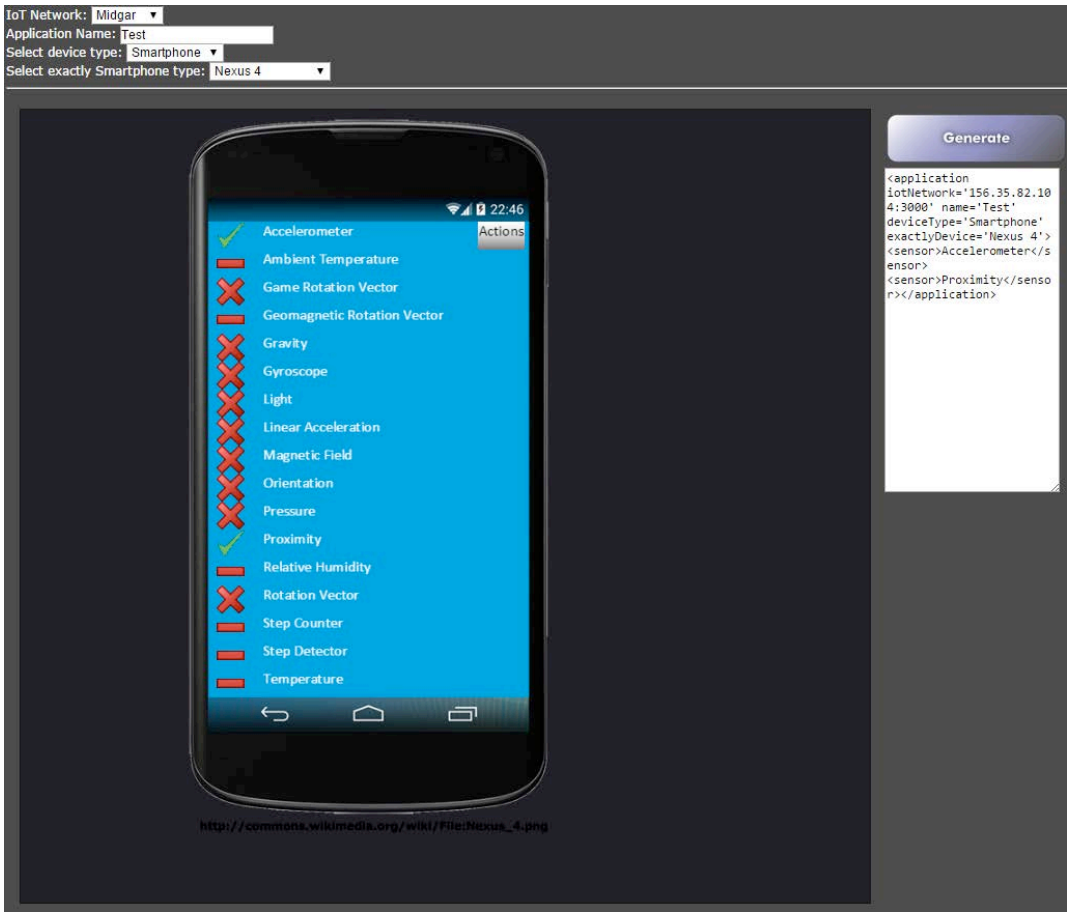



**Figure 4 MOCSL for the object creation of the Midgar platform for Android smartphones: selecting sensors**

The Figure 5 shows the actions on the example of the Android smartphone. This part functions exactly the sensors part.

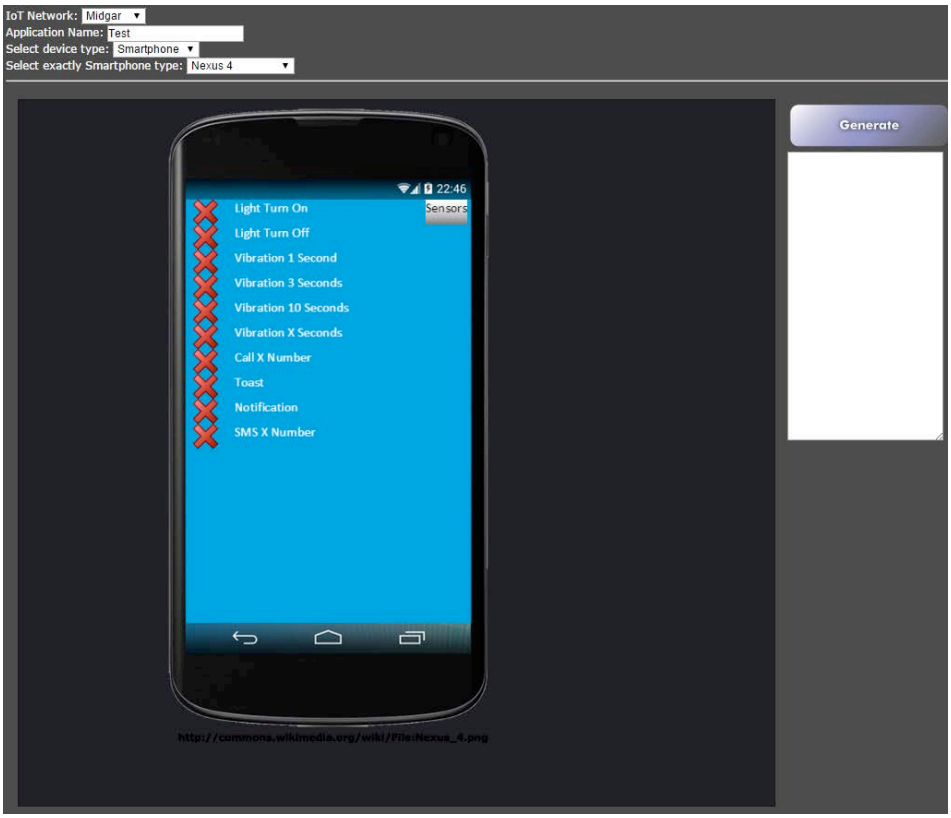



**Figure 5 MOCSL for the object creation of the Midgar platform for Android smartphones: selecting actions**

*2) Serialised Model*

MOCSL creates a serialised model using XML syntax. We have respected in the serialisation model the same nodes that we have in the graphic concrete syntax and the abstract syntax (Figure 6), which we made with Ecore. We have three nodes: **application**, **sensor**, and **action**. **application** is the parent node and it contains the attributes and the nodes **sensor** and **action**. The node **application** has the attributes of the Figure 2A: **iotNetwork**, **name**, **deviceType**, and **exactlyDevice**. Besides, it can have an infinity number of children nodes **sensor** and a**ction**.

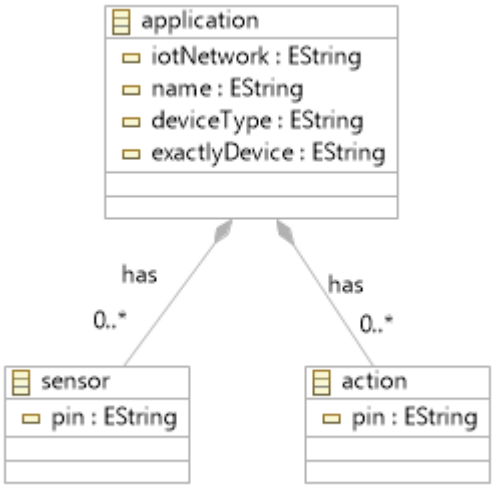



**Figure 6 Metamodel and Abstract Syntax**

We show an example in
```
<application iotNetwork='156.35.82.104:3000' name='Test' deviceType='Smartphone' exactlyDevice='Nexus 4'>
    <sensor>Accelerometer</sensor>
    <sensor>Proximity</sensor>
    <action>Toast</action>
</application>
```
Source Code 1 where we show the serialised model of an Android smartphone, exactly the Nexus 4, with accelerometer and proximity sensor and the action 'Toast'.

```
<application iotNetwork='156.35.82.104:3000' name='Test' deviceType='Smartphone' exactlyDevice='Nexus 4'>
    <sensor>Accelerometer</sensor>
    <sensor>Proximity</sensor>
    <action>Toast</action>
</application>
```

**Source Code 1 Serialised model of an Android smartphone example**

The nodes **sensor** and **action** are similar but with different name because they have a semantic meaning to the processor. They can have the attribute '**pin**' when the device type is a micro-controller. In this attribute, we save the pin of that sensor or action. In source code 2, we show an Arduino Uno with a photo resistor sensor in the pin A0 and three action: a led in pin 13, a DC motor in pin 7, and a servo in pin 2.

```
<application iotNetwork='156.35.82.104:3000' name='Test' deviceType='Arduino' exactlyDevice='Uno'>
    <sensor pin='A0'>photoResistor</sensor>
    <action pin='13'>led</action>
    <action pin='7'>dcMotor</action>
    <action pin='2'>servo</action>
</application>
```

**Source Code 2 Serialised model of an Arduino example**

*3) Objects Generation*

As Figure 7 shows, the second layer is composed of one process. Second layer receives the **Serialised Model** in XML format. Then, the processor parses the nodes and nodes' attributes in order to obtain the necessary information about the application, which was defined by the user. With this information, the processor replaces in the templates the string tokens with the necessary information about Midgar, sensors, and actions. Clearly, the **processor** takes the corresponding template depending on the attribute **deviceType** and **exactlyDevice**. In the case of an Arduino, the **processor** also creates the Arduino application and imports the necessary libraries for the corresponding sensor, for example, native libraries or libraries in C++. When the **processor** finishes the parsing, creates the files and obtains the **Generated Application**. The **Generated Application** is the application that the user need to deploy in their smartphone or in their micro-controller.

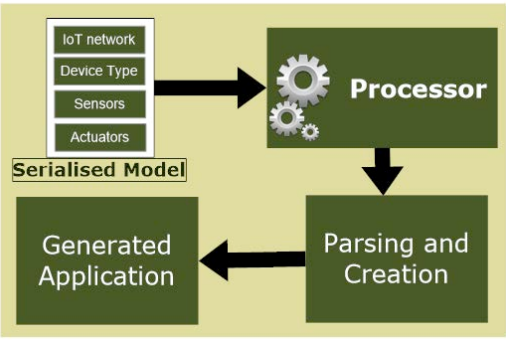



**Figure 7 Internal working of the Object Generation**

*4) Android*

We have developed an Android application using the 5.1.1 version, which corresponds with the API22 called Lollipop. We have chosen the 5.1.1 version because this is the latest version, which is supported by Google, but we have including support until the older supported version, Froyo 2.2. Thus, we have sought to ensure a compatibility with all supported and used versions in the current daily life. In Figure 8 we can see the three used Android smartphones. The application can work with all sensors between both versions. This Android application shows a list of device's sensor, the sensors' value, and supports the communication with Midgar: registering the application in Midgar, starting the data shipping to Midgar, and stopping the data shipping to Midgar. Based on this application, we created our template. For that, we change all the parts, which MOCSL could modify with string tokens.


**Figure 8 The used Android smartphones**

*5) Arduino*

To solve the Arduino micro-controller part, we use an Arduino with USB connection with our Personal Computer (PC) (Figure 9). Then, we have used a Java application as intermediary. The Java application reads and sends data through the USB to the Arduino and interconnects the Arduino and Midgar through the Internet. Using this intermediary, we could implement more task in the intermediary like saving all data in our PC for other intentions, for instance, to apply Big Data to improve our industrial processes. Thus, we have two applications: the Java intermediary and the Arduino source code in C or C++, it depends on the sensor that we use. After to check some sensor and their possibilities with the Arduino, we create the template. We changed all the parts that MOCSL could modify with string tokens, besides we have in different files the sensors' source code. Then, when we work with the Arduino template, we need to include code in the two parts: Java application and Arduino application, but we process the template as only one application.

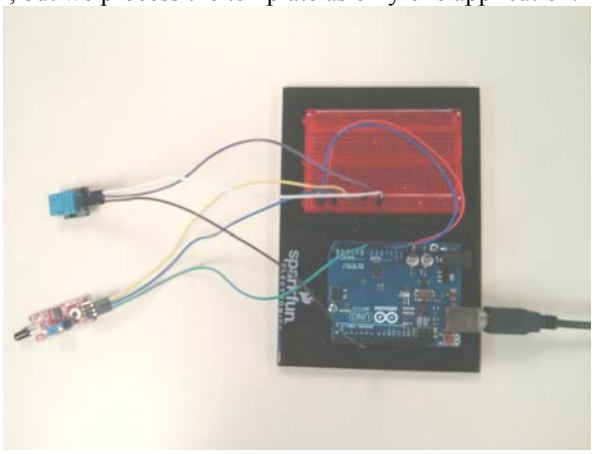


**Figure 9 Arduino picture during a testing with the DHT11 sensors and flame**

*D. Used Software and Hardware*

To develop this research work it was required different software types:

- The Midgar server is based on Ruby 2.2.2p95 and it uses the Rails framework 4.2.1.
- Thin web server 1.6.3
- MySQL Database 5.5.43
- The graphic DSL, MOCSL, was developed by using the element HTML5's canvas and JavaScript.
- The application generator module was developed using Java 8.

For the evaluation of the proposal, we used the next hardware components:

- One Raspberry Pi 2 Model B as a dedicated server with Raspbian 1.4.1.
- Three Android smartphones: a Nexus 5 running version 5.1.1, a Nexus 4 running version 5.1.1, a Motorola with version 2.2.2, and a Samsung Galaxy Mini S5570 with version 2.3.6.
- One Arduino Uno micro-controller board based on the ATmega328.
- During the various tests we used: the temperature sensor TMP36, a speaker, a servomotor, a DC motor, several LEDs, two buttons, a photo resistor, the temperature and humidity sensor DHT11, and a flame sensor.

## IV. EVALUATION AND DISCUSSION

In this section, we are going to explain in detail the process of evaluation, which we used. Afterwards, we are going to show the results that we obtained during the evaluation. For this, firstly we are going to talk about the methodology that we used to perform the evaluation. After that, we are going to show the obtained results and discussed them.

### *A. Methodology*

The main objective of this evaluation process is to validate our initial hypotheses. These are about the creation of a DSL to facilitate the object creation for Smart Objects and if it could be a significant improvement in the IoT. To validate our hypotheses, we implemented an evaluation process with two phases to validate if our contribution is useful and if it accomplishes our hypotheses. We tried to obtain a completely evaluation to verify in the best way our hypotheses using two different evaluations: one quantitative and another qualitative evaluation.

- **Phase 1**: In this first phase, we proposed to create an application using two graphic editors and MOCSL. This application was a basic use case using an Arduino. This has the quantitative evaluation.
- **Phase 2**: Participants who had done the phase 1 will make a survey which uses the Likert scale [68] where participants select their opinion about a declarations. These declarations are based on the phase 1 and about how our research could improve some IoT's aspects. This phase has the qualitative evaluation.

#### *1) Phase 1*

For this phase, we used two graphic editors and MOCSL. As graphic editors, we chose MiniBloq and Bitbloq. We chose MiniBloq because it is a graphic editor for programming Arduino. The other election was BitBloq because it is a specific graphic web editor to create applications for micro-controllers like ZUM and Arduino. Then, these two editors and MOCSL have as final and common platform the Arduino and for this reason, those were our election for the qualitative evaluation. During each task in each platform, we counted the next parameters: time in seconds to do the task, the centimetres that the participants moved the mouse, the clicks in the right mouse button, and the clicks in the left mouse button. To obtain these data we used the Mousotron tool [69].

Firstly, we defined a basic task for participants to emulate a real scenario but we had to beware with the sensor and actuators of each platform, because BitBloq does not support all types.

For that, the assigned task was to create an object with one sensor and two actuators. We chose the photo resistor sensor and a led and servomotor actuators. The task was created an object to simulate the turn on the light and a motor to renovate the air when the photo resistor receives light.

#### *2) Phase 2*

With this second phase, we want to be able to evaluate the simplicity of object creation for Smart Objects and the opinion about how it could be affect to IoT. Then, to improve our evaluation, we design a qualitative evaluation to obtain the participant experience and participant opinion about MOCSL. For that, we used the Likert Scale as a measurement method for the survey because this one is the most used in the design of scales. We design a survey with thirteen questions which have five possible answers based on 5-points Likert Scale giving the following options: 1 as strongly disagree, 2 as disagree, 3 as somewhat agree, 4 as agree, and 5 as strongly agree. Furthermore, we decided to use this method because it is a very used method in software engineering field to obtain information that effectively supports the decision-making [70]. This is exactly our case because we cannot measure the efficiency of object creation through of a DSL nor its potential. For that, we have used the surveys to obtain more information to help us to obtain more exact answers to our hypotheses.

The participants DONE the survey once they had finished the task in the three platforms, then participants responded the survey, anonymously and without help. We tried to ask in the survey about the participant opinion on the use of MOCSL, MOCSL possibilities, and the impact of MOCSL in the Internet of Things and Smart Objects. We show the survey in Table 1.

| # | Question |
|---|---|
| Q1 | The user understands the functionality of the editor elements and their role in application creation process |
| Q2 | This editor allows to create Smart Objects in an easy way, using a few clicks and without having to programme code |
| Q3 | The editor approach makes it difficult to make mistakes while the user is modelling the application |

| | |
|---|---|
| Q4 | This solution offers a fast way to developing the indicated task |
| Q5 | This solution provides assistance to create applications to Smart Objects |
| Q6 | Based on previous experience with the use of other tools, this editor provides a greater ability to generate such applications |
| Q7 | The editor does not require the user to use complex programming skills, as in traditional application development |
| Q8 | The editor includes enough elements and functionality for the user to create a wide range of applications to Smart Objects |
| Q9 | This proposal is a positive contribution to encourage the development of services and applications for Smart Objects |
| Q10 | Internet of Things and Smart Objects will benefit by this solution |
| Q11 | This editor could be used to simplify the classic development process of software applications in other areas |
| Q12 | This editor is easier to use than MiniBloq |
| Q13 | This editor is easier to use than BitBloq |

**Table 1 Survey given to participants**

*B. Results*

In this section, we are going to show and discuss the obtained results in each phase. In the first sub-section, we are going to show the quantitative results. After that, we are going to show the qualitative evaluation.

*1) Phase 1*

In this first phase, we took of each participant the time in seconds to do the task, the clicks in the primary mouse button, the clicks in the secondary mouse button, and the centimetres that the participant moved the mouse.

The Figure 10 shows the time that each participant needed to make the same task in each platform.

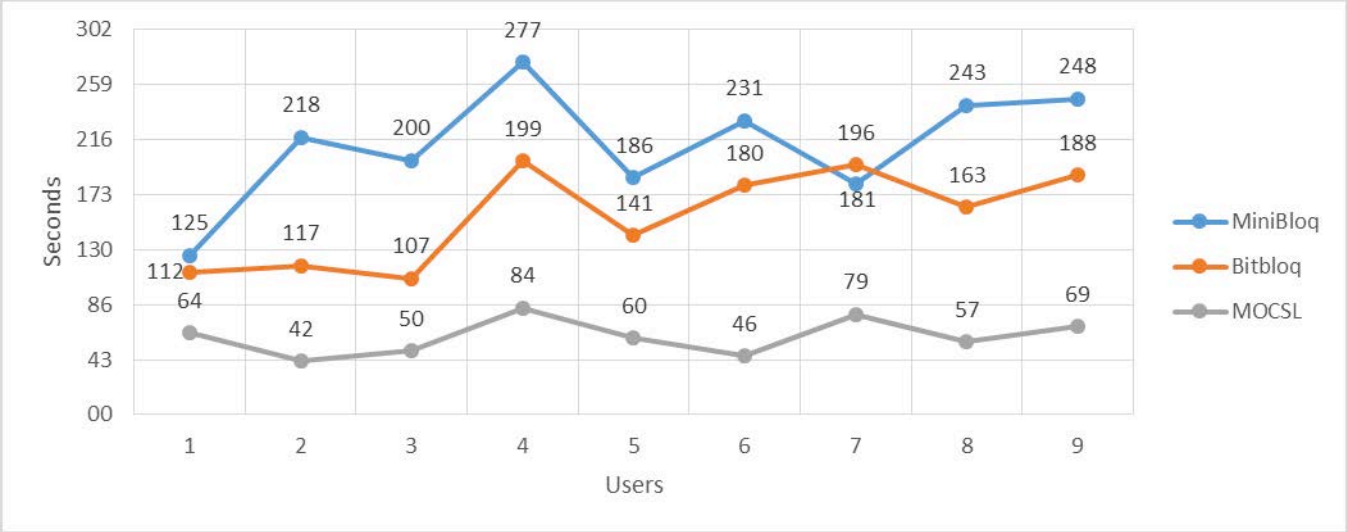



**Figure 10 Participants' time in each platform**

Analysing this chart, we can suggest the following interpretations:

- The faster participant in each platform needed 125 seconds in MiniBloq, 107 seconds in BitBloq, and 42 seconds in MOCSL. Then MOCSL is the quickest platform based on the best results.
- The slower participant in each platform needed 277 seconds in MiniBloq, 199 seconds in BitBloq, and 84 seconds in MOCSL. Then MOCSL is the quickest platform based on the worst results.
- The average time for the nine participants were 212 seconds for MiniBloq, 156 for BitBloq, and 61 for MOCSL. The participants in MOCSL only need the 39% of time than BitBloq and the 28% of time than MiniBloq to make the same application.
- MiniBloq is the slowest except in the case of participant 7.
- MOCSL is faster than the MiniBloq and BitBloq to create objects in all cases. This indicates that MOCSL offers more speed to solve the object creation.
- The worst time using MOCSL, 84 seconds, is better than the best time of MiniBloq and BitBloq, 107 seconds.

In Figure 11 we show the necessary mouse's primary clicks that each participant needed to make the same task in each platform.

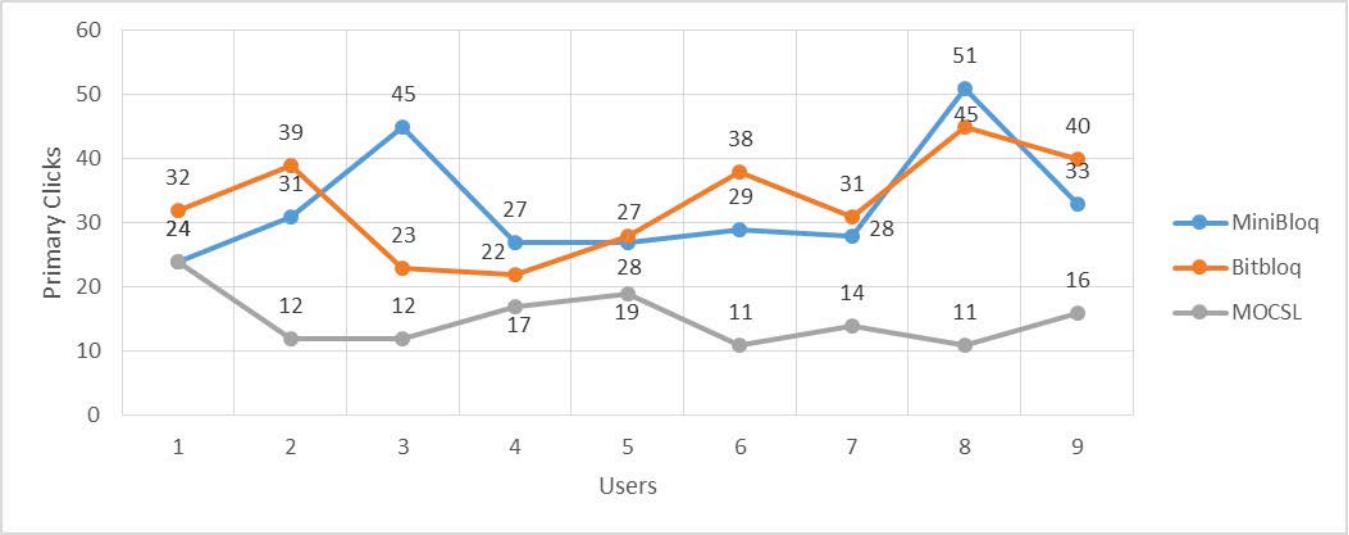



Figure 11 Participants' mouse's primary clicks in each platform

If we analyse this chart we can interpret:

- The biggest number of mouse's primary clicks in each platform was 51 clicks in MiniBloq, 45 in BitBloq, and 24 in MOCSL. Basing on this, MOCSL need less clicks than other platforms in the worst case.
- The lowest number of mouse's primary clicks in each platform was 24 in MiniBloq, 22 in BitBloq, and 11 in MOCSL. According to the best participants, MOCSL needs less clicks.
- The average primary clicks in each platform was 32.78 in MiniBloq, 33.11 in BitBloq, and 15.11 in MOCSL. Then, MiniBloq and BitBloq are similar, meanwhile MOCSL needs less than a half of primary clicks than the others to create an object.
- MiniBloq has the maximum click number but has less average than BitBloq. This indicates that MiniBloq is less understandable than BitBloq depending on people.
- MOCSL is always the platform with less clicks except for participant one who needed the same clicks that in BitBloq. This indicates that MOCSL is the most useful platform.

Figure 12 shows the necessary mouse's secondary clicks that each participant needed to make the same task in each platform. However, in the case of MOCSL, it avoids the use of the secondary button to reduce the complexity and it do it all with the primary button. In the case of MiniBloq and BitBloq, the secondary button has shortcuts and the delete option.

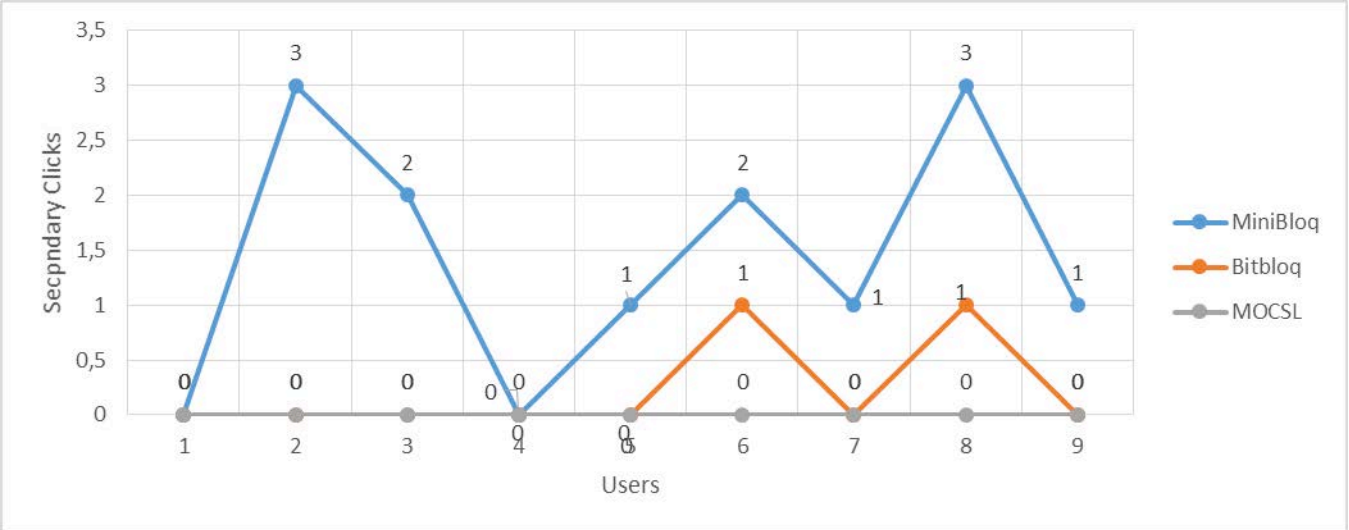



Figure 12 Participants' mouse's secondary clicks in each platform

We can do the next interpretation based on the previous figure:

- MiniBloq is the platform with more mouse's secondary clicks with 13, BitBloq had 2, and MOCSL had 0. Then, BitBloq produces less fails between the participants.
- Only two participants did not use the secondary button in MiniBloq. On the contrary, in BitBloq, only two participants used the secondary button. In this case, BitBloq participants had less fails.

We show in Figure 13 the distance in centimetres, which each participant needed in each platform.

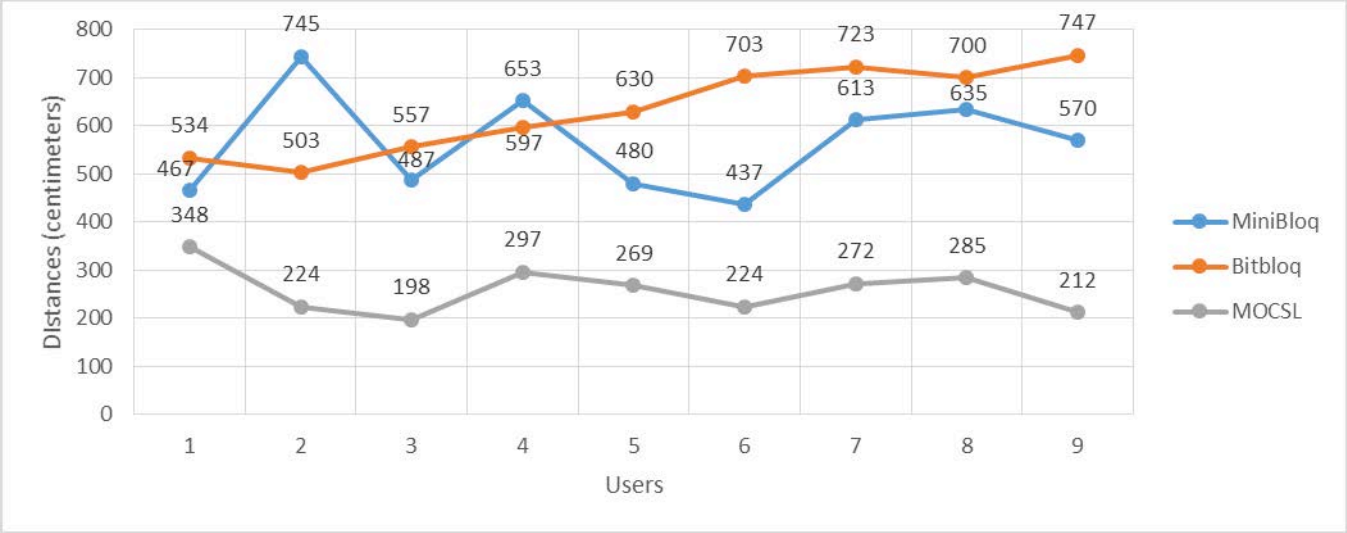



**Figure 13 Participants' distances in each platform**

If we analyse this last chart we can suggest the next interpretations:

- The biggest distance in each platform was 745 centimetres in MiniBloq, 747 in BitBloq, and 348 in MOCSL. This indicates that in the worst case MiniBloq and BitBloq need the same distance but MOCSL only needs less than a half of these.
- The lowest distance in each platform was 437 centimetres in MiniBloq, 503 in BitBloq, and 198 in MOCSL, which indicates that in the best case, MiniBloq needs less distance than BitBloq, but MOCSL needs less than a half of these.
- The average distance in each platform was 565.22 centimetres in MiniBloq, 632.67 in BitBloq, and 258.78 in MOCSL. Then, BitBloq is the platform that requires more mouse movement to make the task and MOCSL is the platform, which needs less distance to solve the object creation problem.
- The worst distance in MOCSL, 348 centimetres, is less than the best distance in MiniBloq and BitBloq, 437 centimetres. For this, MOCSL is always the platform, which needs less distance to solve the task.

*2) Phase 2*

We show in Table 2 all the answers of each participant in an anonymous way for the thirteen questions.

| Id | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 | Q8 | Q9 | Q10 | Q11 | Q12 | Q13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P01 | 5 | 5 | 4 | 5 | 5 | 4 | 4 | 5 | 5 | 5 | 1 | 5 | 5 |
| P02 | 5 | 5 | 5 | 5 | 3 | 5 | 5 | 4 | 4 | 4 | 5 | 5 | 4 |
| P03 | 3 | 5 | 5 | 5 | 5 | 5 | 3 | 3 | 5 | 4 | 4 | 5 | 4 |
| P04 | 3 | 5 | 4 | 5 | 4 | 5 | 5 | 3 | 4 | 4 | 1 | 5 | 5 |
| P05 | 5 | 5 | 5 | 4 | 5 | 4 | 5 | 5 | 3 | 4 | 4 | 5 | 3 |
| P06 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| P07 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| P08 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 |
| P09 | 5 | 5 | 5 | 5 | 4 | 4 | 5 | 4 | 5 | 5 | 5 | 5 | 5 |

**Table 2 Participants' responses for each question**

The Table 3 shows the descriptive statistics of the survey. This table is composed by the itemisation of each question: the minimum, the first quartile, the median or second quartile, the third quartile, the maximum, the range between quartiles, and the mode. We can see in Figure 14 the same data but in a Box and Whiskers plot diagram.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Min** | 3 | 5 | 4 | 4 | 3 | 4 | 3 | 3 | 3 | 4 | 1 | 5 | 3 |
| **Quartile 1** | 5 | 5 | 4 | 5 | 4 | 4 | 5 | 4 | 4 | 4 | 4 | 5 | 4 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Median** | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 |
| **Quartile 3** | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| **Max** | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| **Range** | 2 | 0 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 1 | 4 | 0 | 2 |
| **Inter Qrt.-Range** | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| **Mode** | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 |

Table 3 Table with the general descriptive statistics

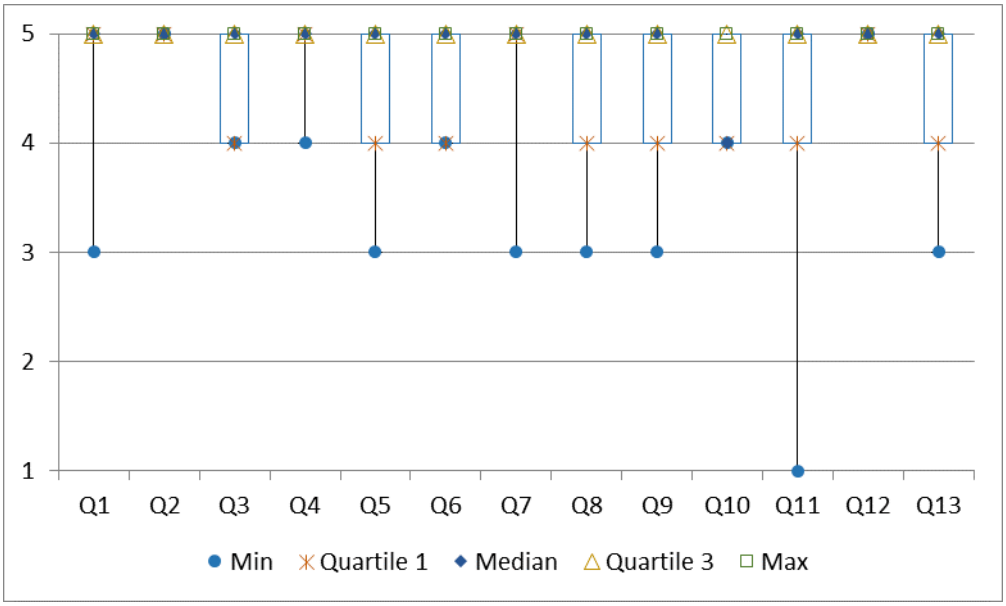



Figure 14 Box and whiskers plot for each question

We can suggest the next interpretations by analysing Table 3 and Figure 14:

- Q2 and Q12 have the highest minimum, 5 out of 5. This means that all participants agree with these declarations, so in this case, they are strongly agree. On the other hand, with the lowest minimum is the Q11. This indicates that Q11 is a question with a high difference of opinions between the participants.
- Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q11, Q12, and Q13 have the highest median, exactly, 5 out of 5. From this, data we can interpret that most of participants are strongly agree with these declarations. Another question is Q10 with a median of 4, then, all participants are minimum agree.
- Q2 and Q12 are the questions with a range of 0. This demonstrates that all participants have the same opinion on these declarations. In our case, the participants' opinion in these questions are 'Strongly Agree'. On the contrary, Q11 has a range of 4. This expresses a high difference between participants' opinion about this question.
- Regarding the mode, all questions except Q10, have a mode of 5 which indicates that the most chosen answer was 'Strongly agree'. In Q10, the mode is 4, then the most answer chosen by the participants was 'Agree'.
- If we study the maximum, we can see that all questions have a 5. This indicates that in all question, somebody chose 'Strongly agree'.
- About the Interquartile range, Q1, Q2, Q4, Q7, and Q12 have a 0, which indicates that the participants had a very close decision between themselves.

The Table 4 shows the different frequencies for each question based on the participants' answers. In this table, we have a breakdown of each question to show the number of votes for each decision and the corresponding percentage for them. In Figure 15 we shows the same data but using a graph bar with the answers' frequency.

| Question | | Strongly Disagree | Disagree | Neutral | Agree | Strongly Agree |
|---|---|---|---|---|---|---|
| Q1. | # | 0 | 0 | 2 | 0 | 7 |
| | % | 0% | 0% | 22% | 0% | 78% |
| Q2. | # | 0 | 0 | 0 | 0 | 9 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | % | 0% | 0% | 0% | 0% | 100% |
| Q3. | # | 0 | 0 | 0 | 4 | 5 |
| | % | 0% | 0% | 0% | 44% | 56% |
| Q4. | # | 0 | 0 | 0 | 1 | 8 |
| | % | 0% | 0% | 0% | 11% | 89% |
| Q5. | # | 0 | 0 | 1 | 2 | 6 |
| | % | 0% | 0% | 11% | 22% | 67% |
| Q6. | # | 0 | 0 | 0 | 3 | 6 |
| | % | 0% | 0% | 0% | 33% | 67% |
| Q7. | # | 0 | 0 | 1 | 1 | 7 |
| | % | 0% | 0% | 11% | 11% | 78% |
| Q8. | # | 0 | 0 | 2 | 2 | 5 |
| | % | 0% | 0% | 22% | 22% | 56% |
| Q9. | # | 0 | 0 | 1 | 2 | 6 |
| | % | 0% | 0% | 11% | 22% | 67% |
| Q10. | # | 0 | 0 | 0 | 5 | 4 |
| | % | 0% | 0% | 0% | 56% | 44% |
| Q11. | # | 2 | 0 | 0 | 2 | 5 |
| | % | 22% | 0% | 0% | 22% | 56% |
| Q12. | # | 0 | 0 | 0 | 0 | 9 |
| | % | 0% | 0% | 0% | 0% | 100% |
| Q13. | # | 0 | 0 | 1 | 2 | 6 |
| | % | 0% | 0% | 11% | 22% | 67% |

Table 4 Frequency table for the general responses

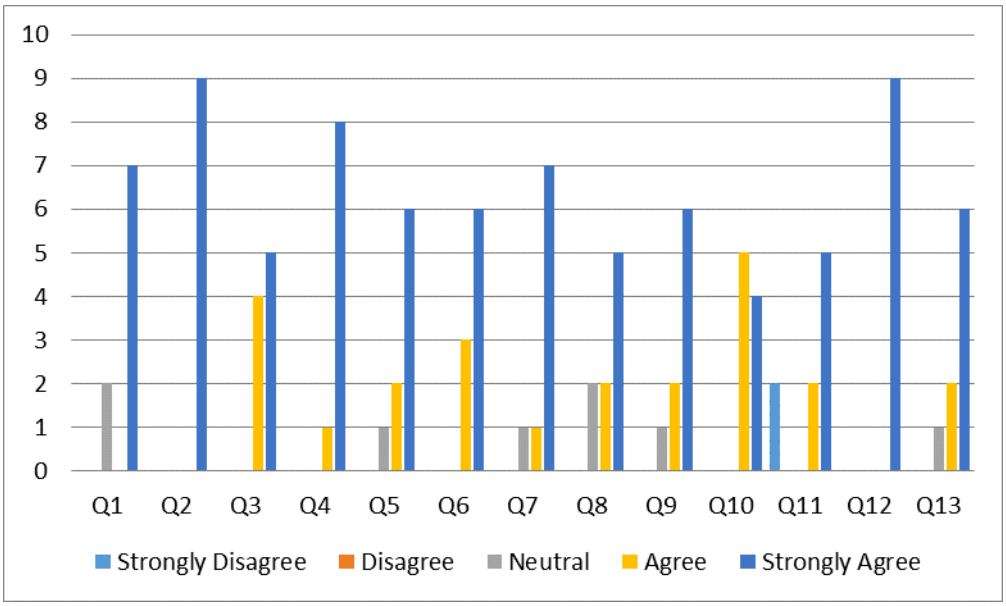



Figure 15 Overall response distribution

Basing on these, we can suggest the following interpretations:

- Q2 and Q12 have a 100% of the votes, the correspondence in numbers is 9 votes, in 'Strongly Agree'. This indicates that all participants are strongly agree with these sentences.

- With an 89% of votes, we have the Q4. This is an amount of 8 votes for 'Strongly Agree' and one vote for 'Agree'. Q3 and Q6 have a minimum of 44% and 33% of votes in 'Agree', respectively. This indicates that all participants are agree with these sentences.
- Q1 and Q7 have a 78% of votes in 'Strongly Agree'. This is 7 votes. Meanwhile, Q1 has 2 votes in 'Neutral' and Q7 has 1 vote in 'Agree' and another in 'Neutral'. This indicates that the majority of people are agree with these sentences but there are a minority with some doubt.
- Q8 has a 56% of votes for 'Strongly Agree' and 22% of votes for 'Agree' and 'Neutral'. These numbers correspond with 5, 2, and 2, respectively. This indicates the majority of participants agree but there are an important percentage indecisive or who do not believe in this declaration.
- Q11 has a 54% of votes in 'Strongly Agree' and 22% of votes in 'Agree' and 'Strongly Agree'. In numbers, this means 5, 2, and 2, respectively. This indicates that the participants were indecisive about this question or maybe we had formulated wrong this question.

## V. CONCLUSIONS

In this paper, we improve the Midgar platform with a novel proposal about a new graphic DSL called MOCSL, which provides a solution to create Smart Objects and all the logic that they need to interconnect them through Midgar. Normally, the creation of Smart Objects is complex because this process needs that people create an application to recollect the objects' data, process it, and send it to another object or to an IoT platform. Then, this requires complex programming skills because people need the knowledge of different technologies, APIs, and how to work with sensors and actuators.

Midgar had a partial solution: Midgar allowed interconnecting different registered objects using MOISL. Now, Midgar allows creating the software that these objects need to obtain the data, to work with their actuators, and to interconnect with the platform. We create a solution using a graphic DSL to obtain an abstraction to facilitate this task for any people without development knowledge. In our case, people only need knowledge about the domain, what they want, and how their objects are.

In order to validate the effectiveness of our proposal, we compared MOCSL with two graphic editor, which offer a similar solution: MiniBloq for Arduino micro-controllers and BitBloq for Arduino and BQ micro-controllers. Then, we made two evaluations to check exhaustively our proposal. We made a quantitative evaluation in which the participants made the same task in each platform and a qualitative evaluation, which consisted in a survey.

The quantitative evaluation demonstrated that MOCSL is the fastest because the participants only needed an average of **61 seconds** to do the task. In comparison with the other platforms, this supposed the **39% of time than BitBloq** and the **28% of time than MiniBloq** to make the same application. Furthermore, MOCSL only needed **the half of primary clicks than the others**. Although MOCSL avoids the use of mouse's secondary button and MiniBloq and BitBloq use this button to delete or other shortcuts, MOCSL had less mouses' primary clicks than the other two platform. This demonstrates that MOCSL reduces the complexity in the control without incrementing the number of clicks. Besides, MOCSL is the platform, which needed less distance to create the same task, exactly an average of **258.78 centimetres** versus 565.22 centimetres in MiniBloq and 632.67 in BitBloq. These last two data suppose that MOCSL is the more usable.

Respecting the qualitative evaluation, we could interpret that the participants chose 'Strongly Agree' in the 71% of the declarations and they chose 'Agree' in a 21% of occasions. Thus, the 92% of declarations has a positive or very positive assessment. In this way, the participants think that MOCSL accomplishes with its functionality to facilitate the object creation in an easy way without the necessity of write source code. In addition, they believe that MOCSL could offer benefits for the Internet of Things and Smart Objects.

Noteworthy that some participants chose in a 7% of responses as 'Neutral' and in a 2% as 'Strongly Disagree'. The meaning in the first one is that we can improve some aspects like the usability and include new elements to offer more options. In the case of the 'Strongly Disagree' answers, we can see that these correspond with Q11. The reason is that MOCSL allows creating a great number of application for different devices but it is impossible to allow modelling any imaginable application. In future extensions we will give a special emphasis on new modelling elements to allow new extensions and possibilities for application development like the inclusion of fuzzy logic, artificial intelligence, other protocols, and other complex tasks. Nevertheless, when we will make these extensions, we must respect the domain of MOCSL to avoid the inclusion of functionalities with an undue complexity.

Thus based on these results we can say that MOCSL can be very useful to facilitate the object creation and reduces the time and the complexity of creating this type of applications. Then, MOCSL is a DSL, which complements the Midgar platform and helps to people without development knowledge to create objects for the Internet of Things in an easy and fast way.

The communication among Smart Objects through the Internet is one of the goals of the Internet of Things. Nonetheless, not all people have the necessary knowledge or time to develop the necessary application. For that, we need to bring certain facilities to the industry and the daily life. This is the purpose of Midgar, exactly, with this last research, MOCSL.

## VI. FUTURE WORK

The Internet of Things has many improvements to do and researches to investigate. As mentioned before, there are still many problems to solve. In the next points, we explain some lines, which could be solved from this investigation.

- **Merging of MOISL and MOCSL:** Merging MOISL and MOCSL for allowing the object creation with some internal restriction or to have only one DSL to create all the process in one-step.
- **Incorporate in Smart Objects artificial intelligence using a DSL:** Create a graphic DSL to allow incorporating and defining the required artificial intelligence by people in an easier and quicker way.
- **Security and privacy in IoT:** Investigating secure methods to register objects in the IoT platforms and networks to prevent the steal and modification of transferred data when an object need to register in the platform and generate secure unique identifiers for objects.
- **Scalability of IoT platforms:** Studying and testing different implementations in the IoT platforms to obtain a correct way of supporting the maximum number of Smart Objects.
- **Performance of Smart Objects:** Researching possible improvements in application like protocols, messages, and others, to improve the object's performance when the object processes data and sends data to save battery and resources.

## VII. ACKNOWLEDGEMENTS

This work was performed by the 'Ingeniería Dirigida por Modelos MDERG' research group at the University of Oviedo under Contract No. FC-15-GRUPIN14-084 of the research project 'Ingeniería Dirigida Por Modelos MDERG'. Project financed by PR Proyecto Plan Regional.

## VIII. REFERENCES

[1] L. Atzori, A. Iera, G. Morabito, The Internet of Things: A survey, Comput. Networks. 54 (2010) 2787–2805. doi:10.1016/j.comnet.2010.05.010.

[2] K.A. Hribernik, Z. Ghrairi, C. Hans, K. Thoben, Co-creating the Internet of Things - First Experiences in the Participatory Design of Intelligent Products with Arduino, in: Concurr. Enterprising (ICE), 2011 17th Int. Conf., Aachen, 2011: pp. 1–9.

[3] L. Tan, Future internet: The Internet of Things, in: 2010 3rd Int. Conf. Adv. Comput. Theory Eng., Ieee, Chengdu, 2010: pp. V5–376–V5–380. doi:10.1109/ICACTE.2010.5579543.

[4] K. Gama, L. Touseau, D. Donsez, Combining heterogeneous service technologies for building an Internet of Things middleware, Comput. Commun. 35 (2012) 405–417. doi:10.1016/j.comcom.2011.11.003.

[5] Fundación Telefónica, La Sociedad de la Información en España 2014, 2015. http://www.fundaciontelefonica.com/arte_cultura/publicaciones-listado/pagina-item-publicaciones/?itempubli=323.

[6] A. Piras, D. Carboni, A. Pintus, A platform to collect, manage and share heterogeneous sensor data, in: 2012 Ninth Int. Conf. Networked Sens., IEEE, Antwerp, 2012: pp. 1–2. doi:10.1109/INSS.2012.6240570.

[7] T. Yamanoue, K. Oda, K. Shimozono, A M2M System Using Arduino, Android and Wiki Software, in: 2012 IIAI Int. Conf. Adv. Appl. Informatics, IEEE, Fukuoka, 2012: pp. 123–128. doi:10.1109/IIAI-AAI.2012.33.

[8] I.W.S.Y.S.E.C. Akyildiz, A survey on sensor networks, IEEE Commun. Mag. (2002) 102–114. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1024422 (accessed June 17, 2013).

[9] G.G. Meyer, K. Främling, J. Holmström, Intelligent Products: A survey, Comput. Ind. 60 (2009) 137–148. doi:10.1016/j.compind.2008.12.005.

[10] C. Perera, C.H. Liu, S. Jayawardena, The Emerging Internet of Things Marketplace From an Industrial Perspective: A Survey, IEEE Trans. Emerg. Top. Comput. PP (2015) 13. doi:10.1109/TETC.2015.2390034.

[11] C. Perera, C.H.I.H. Liu, S. Jayawardena, M. Chen, A Survey on Internet of Things From Industrial Market Perspective, IEEE Access. 2 (2014) 1660–1679. doi:10.1109/ACCESS.2015.2389854.

[12] K. Aberer, Smart Earth: From Pervasive Observation to Trusted Information, in: 2007 Int. Conf. Mob. Data Manag., Conference and Custom Publishing, Mannheim, 2007: pp. 3–7. doi:10.1109/MDM.2007.10.

[13] S. Wang, Spatial data mining under Smart Earth, in: 2011 IEEE Int. Conf. Granul. Comput., IEEE, Kaohsiung, 2011: pp. 717–722. doi:10.1109/GRC.2011.6122686.

[14] C.G. García, J.P. Espada, MUSPEL: Generation of Applications to Interconnect Heterogeneous Objects Using Model-Driven Engineering, in: V.G. Díaz, J.M.C. Lovelle, B.C.P. García-Bustelo (Eds.), Handb. Res. Innov. Syst. Softw. Eng., IGI Global, 2015: p. 21. doi:10.4018/978-1-4666-6359-6.ch15.

[15] A. Martinez-Balleste, P. Perez-martinez, A. Solanas, The pursuit of citizens' privacy: a privacy-aware smart city is possible, IEEE Commun. Mag. 51 (2013) 136–141. doi:10.1109/MCOM.2013.6525606.

[16] C. Hao, X. Lei, Z. Yan, The application and Implementation research of Smart City in China, in: Syst. Sci. Eng. (ICSSE), 2012 Int. Conf., Dalian, Liaoning, 2012: pp. 288–292. doi:10.1109/ICSSE.2012.6257192.

[17] M.C. Falvo, R. Lamedica, A. Ruvio, An environmental sustainable transport system: A trolley-buses Line for Cosenza city, in: Int. Symp. Power Electron. Power Electron. Electr. Drives, Autom. Motion, Ieee, 2012: pp. 1479–1485. doi:10.1109/SPEEDAM.2012.6264625.

[18] D. Ding, R. a Cooper, P.F. Pasquina, L. Fici-Pasquina, Sensor technology for smart homes, Maturitas. 69 (2011) 131–6. doi:10.1016/j.maturitas.2011.03.016.

[19] M. Rothensee, A high-fidelity simulation of the smart fridge enabling product-based services, in: 3rd IET Int. Conf. Intell. Environ. (IE 07), Iee, 2007: pp. 529–532. doi:10.1049/cp:20070420.

[20] G.M. Lee, J.Y. Kim, Ubiquitous networking application: Energy saving using smart objects in a home, in: 2012 Int. Conf. ICT Converg., Ieee, Jeju Island, 2012: pp. 299–300. doi:10.1109/ICTC.2012.6386844.

[21] C.G. García, C.P. García-Bustelo, J.P. Espada, G. Cueva-Fernandez, Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios, Comput. Networks. 64 (2014) 143–158. doi:10.1016/j.comnet.2014.02.010.

[22] C.G. García, J.P. Espada, E.R.N. Valdez, V.G. Diaz, Midgar: Domain-Specific Language to Generate Smart Objects for an Internet of Things Platform, in: 2014 Eighth Int. Conf. Innov. Mob. Internet Serv. Ubiquitous Comput., IEEE, Birmingham, United Kingdom, 2014: pp. 352–357. doi:10.1109/IMIS.2014.48.

[23] S. Luo, H. Xia, Y. Gao, J.S. Jin, R. Athauda, Smart Fridges with Multimedia Capability for Better Nutrition and Health, in: Ubiquitous Multimed. Comput. 2008. UMC '08. Int. Symp., Ieee, Hobart, ACT, 2008: pp. 39–44. doi:10.1109/UMC.2008.17.

[24] The US National Intelligence Council, Six Technologies with Potential Impacts on US Interests out to 2025, 2008.

[25] G. Kortuem, F. Kawsar, D. Fitton, V. Sundramoorthy, Smart objects as building blocks for the Internet of things, IEEE Internet Comput. 14 (2010) 44–51. doi:10.1109/MIC.2009.143.

[26] H. Gu, D. Wang, A Content-aware Fridge Based on RFID in Smart Home for Home-Healthcare, in: Adv. Commun. Technol. 2009. ICACT 2009. 11th Int. Conf., Phoenix Park, 2009: pp. 987–990.

[27] C. Han, J.M. Jornet, E. Fadel, I.F. Akyildiz, A cross-layer communication module for the Internet of Things, Comput. Networks. 57 (2013) 622–633. doi:10.1016/j.comnet.2012.10.003.

[28] J. Rumbaugh, I. Jacobson, G. Booch, Unified Modeling Language Reference Manual, The (2nd Edition), Pearson Higher Education, 2004.

[29] S. Kent, Model Driven Engineering, Comput. Comput. Soc. 2335 (2002) 286–298.

[30] A.G. Kleppe, J. Warmer, W. Bast, MDA Explained: The Model Driven Architecture: Practice and Promise, Addison-Wesley Longman Publishing Co., Inc., 2003.

[31] R.C. Gronback, Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit, Addison-Wesley Professional, 2009.

[32] M. Fowler, Domain Specific Languages, Addison-Wesley Professional, 2010.

[33] M. Eysholdt, H. Behrens, Xtext: implement your language faster than the quick and dirty way, in: Proceeding OOPSLA '10 Proc. ACM Int. Conf. Companion Object Oriented Program. Syst. Lang. Appl. Companion, New York, New York, USA, 2010: pp. 307–309. doi:10.1145/1869542.1869625.

[34] M. Mernik, J. Heering, A.M. Sloane, When and how to develop domain-specific languages, J. ACM Comput. Surv. 37 (2005) 316–344. doi:10.1145/1118890.1118892.

[35] T. Stahl, M. Voelter, K. Czarnecki, Model-Driven Software Development: Technology, Engineering, Management, John Wiley & Sons, 2006.

[36] E. Biermann, K. Ehrig, C. Köhler, G. Kuhns, G. Taentzer, E. Weiss, Graphical Definition of In-Place Transformations in the Eclipse Modeling Framework, Lect. Notes Comput. Sci. 4199 (2006) 425–439. doi:10.1007/11880240_30.

[37] LogMeIn, Xively, (2013). https://xively.com/ (accessed July 29, 2015).

[38] Exosite, Exosite, (2013). http://exosite.com/ (accessed July 29, 2015).

[39] LORD MicroStrain, Sensor Cloud, (n.d.). http://www.sensorcloud.com/ (accessed July 29, 2015).

[40] Etherios, Etherios, (2008). http://www.etherios.com/ (accessed July 29, 2015).

[41] ThingWorx™, ThingWorx, (2015). http://www.thingworx.com/ (accessed July 29, 2015).

[42] Carriots, Carriots, (2011). https://www.carriots.com/ (accessed July 29, 2015).

[43] Paraimpu srl, Paraimpu, (2012). http://paraimpu.crs4.it/ (accessed July 29, 2015).

[44] QuadraSpace, (2010). http://www.quadraspace.org/ (accessed July 29, 2015).

[45] Department of Electrical and Electronic Engineering (University of Cagliari), SIoT, Http://platform.social-Iot.org/. (2012). http://platform.social-iot.org/.

[46] IoBridge, Thingspeak, (2013). http://www.thingspeak.com (accessed July 29, 2015).

[47] Oak Ridge National Laboratory, Sensorpedia, (2009). http://www.sensorpedia.com/ (accessed July 29, 2015).

[48] B.L. Gorman, D.R. Resseguie, C. Tomkins-Tinch, Sensorpedia: Information sharing across incompatible sensor systems, 2009 Int. Symp. Collab. Technol. Syst. (2009) 448–454. doi:10.1109/CTS.2009.5067513.

[49] Microsoft, SenseWeb, Http://research.microsoft.com/en-Us/projects/senseweb/. (2008). http://research.microsoft.com/en-us/projects/senseweb/.

[50] A. Kansal, S. Nath, J. Liu, W.I. Grosky, SenseWeb : An Infrastructure for Shared Sensing, IEEE Multimed. 14 (2007) 8–13. doi:10.1109/MMUL.2007.82.

[51] EVRYTHNG, EVRYTHNG, (2012). https://www.evrythng.com/ (accessed July 29, 2015).

[52] Sen.se, Open.Sen.se, (2015). http://open.sen.se/ (accessed July 29, 2015).

[53] I. Nimbits, Nimbits, (2015). http://www.nimbits.com (accessed July 29, 2015).

[54] CyberVision Inc., KAA, (2014). http://www.kaaproject.org (accessed July 29, 2015).

[55] B. Kaucic, T. Asic, Improving introductory programming with Scratch?, in: 2011 Proc. 34th Int. Conv. MIPRO, IEEE, Opatija, 2011: pp. 1095–1100.

[56] BESTTOOLBARS, AppsGeyser, (2015). http://www.appsgeyser.com (accessed July 29, 2015).

[57] iBuildApp, iBuildApp, (2015). http://ibuildapp.com/ (accessed July 29, 2015).

[58] Indigo Rose Software, Andromo, (2015). http://www.andromo.com (accessed July 29, 2015).

[59] AppsBuilder SPA, AppsBuilder, (2013). http://www.apps-builder.com (accessed July 29, 2015).

[60] Infinite Monkeys, Infinite Monkeys, (2015). http://www.infinitemonkeys.mobi (accessed July 29, 2015).

[61] Appy Pie, Appypie, (2015). http://www.appypie.com (accessed July 29, 2015).

[62] Como Ltd., Como, (2015). http://my.como.com (accessed July 29, 2015).

[63] AppMachine, AppMachine, (2011). http://www.appmachine.com (accessed July 29, 2015).

[64] Mundo Reader S.L., Bitbloq, (2010). http://bitbloq.bq.com/ (accessed July 29, 2015).

[65] Lifelong Kindergarten, Scratch, (2015). https://scratch.mit.edu/ (accessed June 9, 2015).

[66] S. Garner, Learning to program from scratch, in: Adv. Learn. Technol. 2009. ICALT 2009. Ninth IEEE Int. Conf., IEEE, Riga, 2009: pp. 451–452. doi:10.1109/ICALT.2009.50.

[67] MiniBloq, (n.d.). http://blog.minibloq.org/ (accessed July 20, 2015).

[68] R. Likert, A technique for the measurement of attitudes, Arch. Psychol. 22 (1932) 1–55.

[69] Blacksun Software, Mousotron, (2013). http://www.blacksunsoftware.com/mousotron.html.

[70] M. Kasunic, Designing an effective survey, Pittsburgh, 2005. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA441817 (accessed July 29, 2014).

# ANEXO VI. Midgar: Detection of people through computer vision in the Internet of Things scenarios to improve the security in Smart Cities, Smart Towns, and Smart Homes

# Midgar: Detection of people through computer vision in the Internet of Things scenarios to improve the security in Smart Cities, Smart Towns, and Smart Homes

Cristian González García[a*], Daniel Meana-Llorián[a], B. Cristina Pelayo G-Bustelo[a], Juan Manuel Cueva Lovelle[a], Nestor Garcia-Fernandez[a]

[a] *University of Oviedo, Department of Computer Science, Sciences Building, C/Calvo Sotelo s/n 33007, Oviedo, Asturias, Spain. Tel: +34985103397*

[a] gonzalezgarciacristian@hotmail.com, *danielmeanallorian@gmail.com, crispelayo@uniovi.es, cueva@uniovi.es, nestor@uniovi.es*

* Corresponding author.

*Abstract*— Could we use Computer Vision in the Internet of Things for using pictures as sensors? This is the principal hypotheses that we want to resolve. Currently, in order to create a safety area, cities, or homes, people use IP Cameras. Nevertheless, this system needs people to watch the cameras, to watch the recording after something occurred, or to watch when the camera advices them about a new movement. This is an inconvenient. Furthermore, there are many Smart Cities and Smart Homes around the world. For that, we thought in using the idea of the Internet of Things to add a way of automating the use of IP cameras. In our case, we propose the analysis of the pictures using Computer Vision to detect people in the pictures. Then, with this analysis, we can obtain if the picture contains people and to treat pictures as if they were sensors with two states. Notwithstanding, Computer Vision is a very complicated field. For that, we needed a second hypothesis: Could we work with Computer Vision in the Internet of Things with a good accuracy to automate or semi-automate this kind of events? Then, the demonstration of these hypotheses required a testing over our Computer Vision module to check the possibilities that we have to use this module in a possible real environment with a good accuracy. Our proposal, as a possible solution, is the analysis of entire sequence instead of isolated pictures for using pictures as sensors in the Internet of Things.

***Smart Cities; Smart Towns; Smart Homes; Internet of Things; Smart Objects; Computer Vision; Vigilance;***

## I. INTRODUCTION

Currently, we live in the information era. We have many things in our daily life with access to the Internet and capable of making our daily life more comfortable like smartphones, tablets, computers, some cars, Smart TVs, and so on. Every new day, we have more devices and better Internet connection [1]. These devices are able to run programmes, which use the devices' sensors, or to do other tasks like to create alarms or notifications, turn on or turn off the device, and et caetera. These objects are known as Smart Objects [2].

Smart Objects provide us with many possibilities and every day we have more different Smart Objects. However, Smart Objects can be more useful in the case of being interconnected with each other and with other objects like sensors, actuators, computers, or micro-controllers. This interconnection is called the Internet of Things (IoT). The Internet of Things allows creating huge or small networks to obtain a collective intelligence through the objects' information processing. The first example is how it was born because the first time that the IoT was used was in the supply chains, the object identification in chemistry plants, people, and animal using Smart Tags as RFID and NFC [3–5]. This can be applied in cities, also known as Smart Cities, in order to offer different services that improve citizens' 'livability' [6]. Smart Towns use a similar application although they use it to preserve their culture, folks and heritage in small cities and towns [7]. In other cases, we can use the IoT to create Smart Homes that can control and automate certain things of our houses like doors, windows, fridges [8–10], irrigation systems, lights, distribution multimedia [11], and so on. By the contrary, some governments apply IoT to control and care in better way the Earth, which also known as Smart Earth [12], in front of different dangers like fire, earthquakes, tsunamis, or floods. Moreover we can use IoT to anticipate and prevent human disasters like the case of the Deepwater Horizon at Gulf of Mexico [13] or the problem in the security system of the Nuclear Central of Fukushima to detect the tsunami and automate the turn off of the diesel motors or activate an protection over them. Nonetheless, these systems need a central service to control and manage their data and their objects and sometimes, to create intelligence and take the decisions. Then, they need an IoT platform.

Presently, IoT platforms exist for many and different uses: business platforms, research platforms, platforms in beta state, and open source platforms [14]. All these are more or less similar because they allow working with objects and interconnecting them. Some platforms offer people an Application Programming Interface (API) to facilitate this task

A very interesting application for the IoT would be the recognition of people using Computer Vision. Using Computer Vision, an IoT platform could offer more security systems to Smart Cities, Smart Towns, Smart Homes, and Smart Earth because they

could recognise a person in the incorrect place in the wrong hour because maybe, he is a thief or a potentially dangerous person for the environment like a pyromaniac. We could obtain this functionality without seeing the camera by a person and facilitate this labour to see only the critical pictures. We can see a similar research in [15] where he used sensors to obtain data and when the platform obtained certain condition, the camera took a picture. A similar idea was proposed in [7] to protect the heritage of Smart Towns. Notwithstanding, they did not apply Computer Vision because they only take a picture under certain conditions. After that, they need a person to see that picture and evaluate the situation.

For these reasons our hypotheses were: could we insert the use of Computer Vision in the Internet of Things? Could we use the pictures from an IP Camera as a sensor? Could we obtain a good accuracy to automate or semi-automate this kind of events? A possible solution for solving these hypotheses is the creation of a Computer Vision module and insert this module in an IoT platform. In our case, we used the Midgar IoT platform [14,16]. We developed a Computer Vision module and modified Midgar to support Computer Vision. To test our first one hypothesis, we used an Internet Protocol (IP) camera connected to the IoT platform and we took different pictures to test the functionality. To test the last both, we took many pictures of the inside of the University. After, we tested all this pictures in a batch process with our Computer Vision module to obtain the accuracy that it has. However, in order to improve the possible detection of people, we analyse the entire sequence to find people in that sequence instead of analysing picture by picture in the way of improving the identification of people by the use all pictures of a movement.

In the rest of this article, we will discuss in section 2 about what is the Internet of Things, Smart Objects, different types of IoT like Smart Cities, Smart Towns, and Smart Homes. Besides, we will explain a brief of Computer Vision, we will continue talking about some of the current IoT platform, and we will present the related work. In section III, we will describe the case study, in our case, the Midgar IoT platform. After, section IV will explain the methodology that we used. After, it shows all the results of our evaluation and the discussion of them. To finally this paper, we will give our conclusions in section V and we will show some ideas of possible future work in section VI.

## II. STATE OF THE ART

Currently, the Internet of Things (IoT) is one of the most used technologies with interest for some country, like the United States of America [17], and the United Nations [4]. Nevertheless, the IoT needs many improvements because it was born few years ago and has many problems to resolve as we can see in different recent articles about Smart Towns [7], protocols [18], security [19], or others [4,11]. The goal is to use the IoT to interconnect everything, from food to computers and then, automate different process to improve our daily life. Now, we have different Smart Object with a small but smart functionality in our life. Every day we have more Smart Objects with Internet connection like smartphones, cars, tablets, or computers [1]. Besides, we can use these devices to make our life easier in some Smart Cities with special services like Santander [20], for instance, to park or to manage our Smart Homes, or automate some tasks like the irrigation service. However, sometimes this is very difficult. For example, if we have a movement alarm in our house and we have animals, we would have a problem because maybe, the alarm would sound for the animal and we only want that the alarm would sound when it recognise a person. One solution that we propose to solve this problem is the use of Computer Vision through the Midgar IoT Platform [14,16].

### *A. Internet of Things*

The Internet of Things allows the interconnection of physical and virtual things. These things can be objects of the physical world or information of the virtual world. This interconnection can be at whatever time like meanwhile you move around the world, on the move or at night, in any place like outdoor or indoor, and between anything like Human to Human (H2H), Machine to Machine (M2M), or between humans and machines (H2M) [21]. Then, the IoT allows creating a Smart World, which is the fusion of heterogeneous and ubiquitous objects, Smart Cities, Smart Towns [7], and Smart Homes [22], with all devices interacting between themselves [7].

However, many heterogeneous things compose the IoT. The most important are the Wireless Sensor Networks (WSN), they are the core of the Internet of Things [23]. A WSN interconnects sensors to obtain data with a server or special system to work and maybe automate task in one place. Other components are actuators, which allow executing actions, like motors, funs, machines, and so on. Another type of network is the fusion between a WSN and the actuators, known as Wireless Sensor and Actuator Network (WSAN). Besides, Smart Objects are other important components because they can perform actions as actuators, they can sense because normally they have sensors, and they are smart to do and process information, data, and actions. Nevertheless, all these components need a connection with the Internet, but currently, every object has networking capabilities because every object can be connected to the Internet [24].

For all this, the definition of what is the IoT is: the Internet of Things is the interconnection of heterogeneous and ubiquitous objects between themselves through the Internet [4,5,14,16]. The goal of the IoT is interconnect the whole world through the creation of different smart places to automate, improve, and facilitate our daily life [16].

### B. *Smart Objects*

A Smart Object or Intelligent Product is a physical thing with the capacity of interacting with the environment, in some cases with intelligence to take decisions, with autonomous behaviour, and identifiable in all his useful life. As we said before, Smart Objects are one of the fundamental parts of the IoT. Some examples of Smart Objects are smartphones, Smart TVs, Tablets, some cars, IP Cameras, computers, micro-controllers, some freezer prototypes, and etcetera.

However, some objects which are connected have a limited memory, CPU, or power [24]. For instance, an IP camera might not have the necessary space or power computation to run applications with the capabilities of applying Vision Computer for one or more pictures. Then, these cases need another place for obtaining the necessary computing. One option is to use an IoT server for offering that intelligence through the network as occurs with the IoT platform Midgar [16]. In this way, we can expand the possibilities of objects like sensors and Smart Objects.

We can classify the Smart Objects according to the three dimensions which were proposed by Meyer in [2]: **Intelligence Level**, **Location of the Intelligence**, and **Addition of the Intelligence Level**. In our case, our IP Camera is a Smart Object with an Intelligence level of **Notification of the Problem**, with the **Intelligence through the Network**, and **Intelligence in the Element**.

### C. *Smart Cities ,Smart Towns, and Smart Homes*

Smart Cities are a very important part in the IoT [25]. A Smart City has different type of sensor distributed around the city to obtain information about it. Then, using ubiquitous communication networks, WSN, WSAN, and intelligent systems, the Smart City can offer new services to citizens [26], and facilitate their daily life and improve the city's 'livability' [27]. We can see some European Smart Cities like Luxemburg, Aberdeen, Oviedo, and many other cities with their qualification in based of different criterions in [28] or in [29], which define the Smart City concept and analyse the Smart Cities of Europe in based on six indicators. Another example is a big European project called SmartSantander in [20], which proposes an architecture for the IoT in Smart Cities and shows the different offered services.

On the contrary, Smart Towns also exist: small cities or towns with a great culture and heritage that need to preserve and revitalise instead of only improve the 'livability' as in the Smart Cities. Smart Towns have to protect and expand their culture and heritage to avoid the oblivion of their buildings, monuments, landscapes, folklore, tradition, and a long etcetera. For instances, Smart Towns can share their places, can record their culture and the way to make their typical dishes, monitoring the conditions in a determinate place that need special condition like libraries or museums, or protect the monuments [7].

Smart Homes [30], also known as Intelligent Homes, are more close. They search the 'livability' in our homes. They provide us with an automated system to improve our daily life in our homes. They are based on a WSAN to control different things in the home in based to certain defined and automated events or using a remote control or smartphone to control it. We can control the doors or the windows using our smartphones or using Smart Tags like RFID or NFC, create an automate system lights, save money in the heater with different sensors and an intelligent system, and so on.

With this proposal, we want to improve the $4^{th}$ and $5^{th}$ principles of the 'livability' [6,7]. We want to improve the security in these three IoT areas using Computer Vision for detecting actions, people, or certain behaviour. By means of using an IP Camera, we can send pictures when we detect objects in move, removed objects, abandoned objects, and when the camera was manipulated. After, with these pictures, we can apply Computer Vision to detect dangerous situation for the important things like our families or things in homes, monuments, heritage, and people in the towns, and citizens in the cities.

### D. *Computer Vision*

Computers can only process zeros and ones. Nevertheless, years ago, the Artificial Intelligent (AI) was born to offer the possibility of creating programmes to allow the computers to learn. John McCarthy coined this term in 1955 in the conference of Dartmouth [31]. Inside the AI, one of the fields is the Computer Vision. Computer Vision is the field that allows the computers to 'learn' to recognise a picture or the characteristics of a picture. This allows identifying objects, humans, animals, or a position in a picture. For that, the goal of Computer Vision is that a machine can understand the world [32].

For reaching this goal, there are many algorithms for the long process of recognise something in a picture. Some algorithms to obtain the features of the dataset that you use to train the model are Histogram of Oriented Gradients (HOG) , Local Binary Patterns (LBP), Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), a mix of various, and so on. Other algorithms are the necessaries to train the model using the extracted features that you obtained previously. Some examples of these algorithms are Support Vector Machine (SVM) and Logistic Regression. However, the task of obtaining a good model is very difficult. You need to take many good pictures and try many times with other pictures to check that your model works well. Besides, you need to create a model to solve your problem, because if you create a general model maybe, it could have a low accuracy. It is the case of the example model of some application or our case, in this article, because we use the example model of OpenCV.

We use Computer Vision in this proposal to recognise the existence of people in the pictures that the camera send to the IoT platform and in the positive case, do an action. Then, we could use the pictures as a special sensor.

### E. Internet of Things Platforms

As we explained before, to obtain the best potential of Smart Objects, we need to interconnect them between themselves. Notwithstanding, we need a 'brain' which can manage and notify the Smart Objects and sometimes, for working as the brain for some other objects like actuators. This 'brain' is an IoT platform. For this purpose, we have different IoT platforms with different pros and cons. We can classify this IoT platforms in the next four groups [14]:

- Business platforms: Xively [33], Exosite [34], SensorCloud [35], Etherios [36], ThingWorx [37], and Carriots [38].
- Research platforms: Midgar [16], Paraimpu [39], QuadraSpace [40], and SIoT [41].
- Platforms in beta state: ThingSpeak [42], Sensorpedia [43,44], SenseWeb [45,46], Evrythng [47], and Open.Sen.se [48].
- Open Source platforms: Nimbits [49] and Kaa [50].

Some of these IoT platforms have qualities that others do not have. However, none of them have a module of Computer Vision for working with pictures as sensors. You can use an IP camera as an actuator, namely, you can connect the IP camera and take pictures under certain condition. Our intention is to use the IP cameras as sensors. For example, you connect the IP camera and send the pictures under certain condition. Then, when the Iot platform receives the picture, the IoT platform will have to analyse the picture for searching, for example, people. In the case that the IoT platform detects people, the IoT platform triggers the action that the user had defined for this case. For that, we propose in this article one possible solution to use the IP camera as a sensor using Computer Vision to detect things in the pictures.

### F. Related Work

In the current literature, there are some uses of cameras in combination with IoT and sensors. In some cases, they use this combination to improve the job conditions, obtain more data without traveling to the place, or learning.

One of those uses is to improve the care of bees and facilitate the job of beekeeper [15]. They used a sound sensor to send a picture when the sound exceed some limit to send a message with this picture to the beekeeper. Then, the beekeeper could decide if the hive needs his visit or not depending on the things that he saw in the picture and the sensor information, which was obtained using a WSN. With this system, the keepers could reduce the frequency of their visits to the moments that they receive critical information, as demonstrated in [15], because they can see information remotely. They analyse the sensor information with an algorithm to avoid the human interaction for obtaining the state of the beehive but they have to see the picture when they receive it to see what happen.

Another example is the proposal of using this combination for learning. For instance, the IoT could help to learn and show different arts between master and students by collecting data and find the best way to train. With this way, they can help to protect the heritage of towns and its folks [7].

The previous proposals used the IoT with cameras. However, they needed a person to see the picture in order to take a decision in the first case or they record the movement to add more information to the sensors in the second one case. In our proposal, we use the camera to obtain the picture but we send the picture to a Vision Computer module. This module is the responsible for taking the decision and sending this decision to the IoT network, which is the manager of the service. Then, we automate this step in based on a model to avoid the intermediary and accelerate the response because maybe, in some cases, is impossible for people see many pictures or take a decision immediately.

## III. CASE STUDY: MIDGAR

Midgar is an Internet of Things platform to investigate different solutions for the IoT [14,16]. In this paper, we try to find a solution for the integration of Computer Vision in an IoT platform for analysing pictures from IP cameras to find a determinate object in the pictures and use the pictures of the IP camera as sensors. In this section, we are going to describe the changes that we did in the Midgar platform and show our proposed solution to add the Computer Vision module in Midgar.

### A. Midgar's Architecture

The system architecture is very similar to the original Midgar's architecture. It has the same four layers as we can see in Figure 1: **Process Definition**, **Service Generation**, **Data Processor and Object Manager** and **Objects**. However, we added the Computer Vision module in the third layer and we had to modify the different layers to support the new functionality.

The first one layer is the **Process Definition**, which contains the user's process. This is the only layer with the user interaction. The user (Figure 1.1) must define the process that he needs using **Midgar Object Interconnection Specific Language** (**MOISL**) that we developed in [16] and we can see in Figure 1.2. **MOISL** was developed using HTML5's canvas. When the user finished defining and click in the editor's generate button, the editor generates the **Serialised Model** (Figure 1.3). Then, the editor serialises the model that the user has defined using **MOISL** in an eXtensible Markup Language (XML) file. This **Serialised Model** contains all the necessary information about the model, which was created by the user, and the information that the second layer needs to generate the **Active Process**.

Afterwards, the **Service Generation** receives the **Serialised Model**. This second layer parses and processes the information of the **Serialised Model** in the **Processor** (Figure 1.4), which creates, compiles, and executes the **Active Process** that interconnects the objects (Figure 1.5).
The **Active Process** is in the third layer, **Data Processor and Object Manager.** The **Active Process** keeps working in the server while is performing the defined user's task. This process has a continued and directly communication with the **Midgar Store** (Figure 1.6), which is the Midgar's core because it contains the database with the services, the objects, the actions and the data.
The last layer is the layer that contains the **Objects**. In this case, our IP camera. These objects implement the message interface to keep a permanent and bidirectional connection with the server (Figure 1.7). However, the IP camera cannot implement this message service because the IP camera has private software. Nevertheless, it can send pictures by HTTP protocol. Then, the IP camera has to send the picture using the REpresentational State Transfer (REST) of Midgar service. After, the Midgar service, which is in the third layer, realises that it is a picture because Midgar analyses the Multipurpose Internet Mail Extensions (MIME) type of the request and sends this request to the **Computer Vision module** (Figure 1.7). The **Computer Vision module** analyses the picture and responds if the picture has or not a pedestrian.

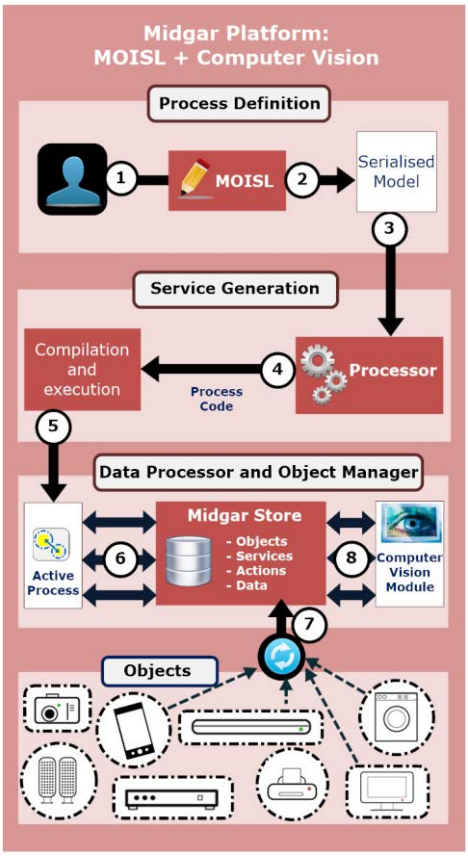



**Figure 1 Architecture of the Midgar Platform with the Computer Vision module**

*B. Implementation*

In this sub-section, we are going to explain the new implementation of Midgar. Firstly, we are going to explain the flow through the different layers. Afterwards, we are going to describe the functionality and the interconnection of our Computer Vision module in Midgar. Lastly, we are going to talk about the functionality that the IP camera offers.

*1) Midgar's flow*

In order to add the capacity of Computer Vision to Midgar, we had to create a module with this capacity. The rest of Midgar is equal to the previous platform [16]. Then, the difference is when Midgar receives a picture. When this occurs, Midgar detects that the request contains a picture because Midgar analyse its MIME type. For instance, when an object like an Arduino or other, which has the possibility to insert in it an application, is connected to Midgar, this object can send a message using the XML standard style of Midgar. In this case, the Canon IP Camera does not allow modifying the software, as occur with others IP cameras, but we can analyse the MIME type to see that it is a picture as we can see in Figure 2. Then, when Midgar receives

a picture, Midgar saves the picture in a folder. When Midgar passes five seconds without receives new pictures, Midgar sends the picture sequence to the Computer Vision module. After that, the Computer Vision module analyses the folder, which contains the whole picture sequence, to find people in at least, one picture. In the case that the module finds one or more people, it will respond to Midgar with a 'True', in other case, a 'False'. Then, Midgar will store this response in the database, as if it were a sensor with only these two possible responses.

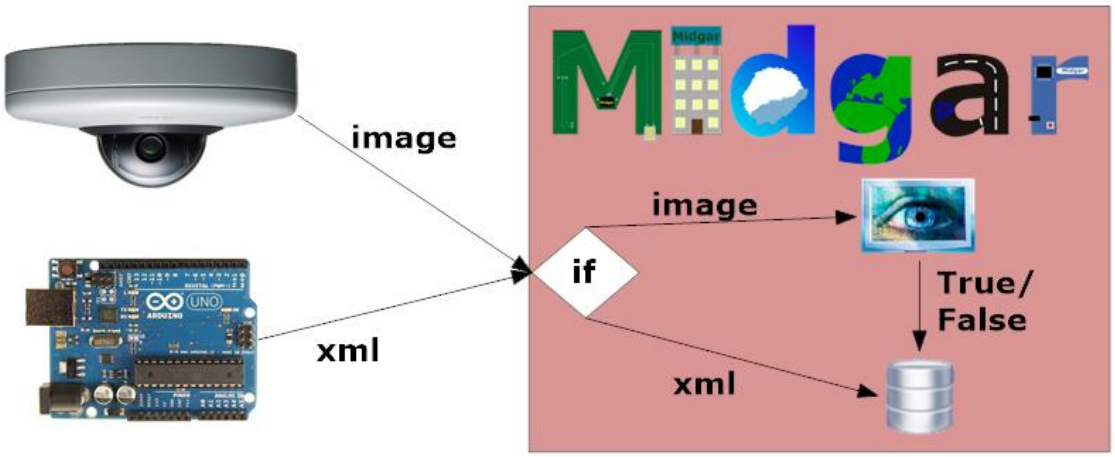



Figure 2 Midgar's flow

*2) Computer Vision Module*

We chose as a possible solution the creation of a separated module. In this way, we could call it when we need to evaluate a picture or a picture sequence. The decision was to separate the implementation of the Computer Vision module from the IoT platform. This allows us to execute different tests with the same module and the same architecture for the evaluation of this proposal. We only have to change the parameters that we use to call.

The Computer Vision module was developed in Python in order to use the library Open CV that does the body detection work. The use of Open CV required use the library Numpy. The workflow of this module consists on loading an image from a file, converting it to a bytes array, transforming it to grey scale, and using the OpenCV library to detect the number of bodies in the image. If there is any body, the module will return the Boolean value 'True' else, it will return the Boolean value 'False'. However, we chose to improve the module's recognition by using of picture sequences instead only one picture. In this way, we could obtain more accuracy because our objective is the detection of a dangerous movement.

However, OpenCV needs setup a few variables. The first one is to indicate the scale factor. It is necessary to create the scale pyramid that the algorithm uses to find objects in different depth, which we set this value to '1.01'. The second is the minimum number of objects that have been detected near each other to detect the whole set as a single object. We setup this second with the value '15'. The last parameter is the minimum size of each detection window that we setup to '(200, 200)'. Nevertheless, the values that we used to setup the OpenCV library depend on the context. Furthermore, an external XML file is required because OpenCV loads the classifier from external file in order to reuse the same code with different classifiers. In our proposal, we decided to use one of the sample classifiers that is available in the downloaded OpenCV files whose filename is 'haarcascade_upperbody.xml'. With this classifier, OpenCV is capable of detecting upper bodies. If we wanted to detect other things, we would create new classifiers extracting the needed features. Furthermore, this model is an example and it is a bit weak. For that reason, if we want a better recognition, we will have to train a new model to obtain a better and more specific classifier. However, we could improve in any way the movement detection in the case that we analyse the sequence to obtain at least one picture. For instance, in the case that we would have a better classifier, we could increase this number and require at least of three or five pictures with the object that we want to recognise. It is very useful if for instance, we want to use this system to detect dangerous people like burglars, thieves, and so on in our home, or pedestrian in some private or dangerous area. In these cases, it is preferred that the module gives false negatives instead false positives cases. The reason is because if we received a false negative, we only received a false alarm, but in the contrary, if we obtain a false negative, we could have a thief in our home.

In order to test the body detection, other module was developed that skips Midgar. This module, also developed in Python, uses the library Flask to receive the images from the camera and follows the work flow of the other module although it does not return a Boolean value. It saves the images that the camera sends in a directory and saves another image if the module detects a body and draws a green rectangle round the body in it. With this information, we will be able to do the evaluation of our proposal.

*3) Canon IP Camera*

We used the Canon VB-S30D as IP Camera but before, we updated the firmware to the last version 1.2 of May 26th of 2015. We connected the camera through Ethernet connection. This camera is a Smart Object because it recognises its data and it can have decision according to the video. This IP camera allows doing streaming using an URL or send pictures or emails when the IP camera detects some changes in the video. For instance, if it recognises some movement or the modification of some object in the scene, it can send pictures or emails with the pictures of that moment.

In Figure 3 we can to see the five detection types that are offered by the camera. The first one is the 'moving object detection' and it consists of detecting some movement. Secondly, the 'abandoned object detection' when it detects after few seconds new objects. Another is the 'removed object detection' when it detects after few seconds the vanishing of one object. The fourth is the 'camera tampering detection', when it detects that the camera was manipulated. The last one is the 'step detection' when it detects a movement over a defined line. These five events are configurable and allow us to receive pictures or emails only when the scene was changed.

Then, we can choose two modes in the camera. We can analyse the streaming or analyse the pictures that we receive. For working with the camera in Midgar, firstly, we have to configure the IP camera using Internet Explorer and set it up in it, configure the IP and port of the Midgar's REST service, which the camera has to send the picture. Afterwards, we have to register the IP camera in the platform so that the camera can be selected in MOISL. After we can select the camera in MOISL, create the interconnection, and work with the camera.

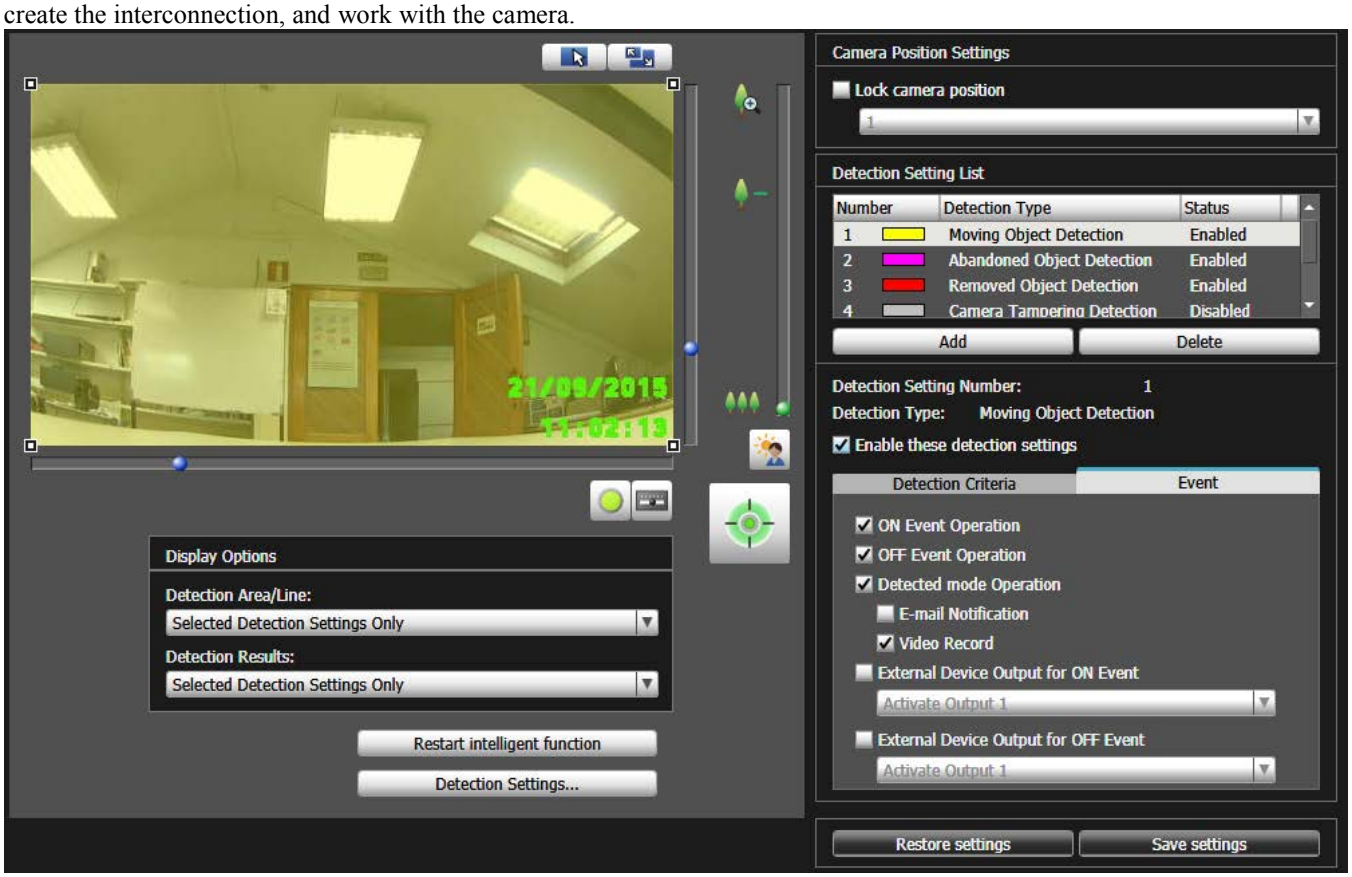



**Figure 3 Canon IP Camera with the 'Moving Object Detection' selected**

*C. Used Software and Hardware*

We used the next software to develop this research work:

- Midgar:
  - The Midgar server is based on Ruby 2.2.2p95 and it uses the Rails framework 4.2.1.
  - Thin web server 1.6.4
  - MySQL Database 5.5.43
  - The graphic DSL, MOISL, was developed by using the element HTML5's canvas and JavaScript.
  - The application generator module was developed using Java 8.
- Computer Vision module:

    - The computer vision module was developed using Python 3.4.3
    - The library OpenCV 3.0.0 to apply Computer Vision to the pictures
    - Numpy 1.10.0 because it is a requirement of OpenCV
    - Flask 0.10.1 to develop a mini-server to the test
    - Model: 'haarcascade_upperbody.xml'
- Camera IP:
    - Canon VB-S30D with firmware v.1.2.0
    - Internet Explorer 11 to access to the camera configuration

For the evaluation of the proposal, we used the next hardware components:

- One Raspberry Pi 2 Model B as a dedicated server with Raspbian 1.4.1.
- Three Android smartphones: a Nexus 4 running version 5.1.1, a Motorola with version 2.2.2, and a Samsung Galaxy Mini S5570 with version 2.3.6.
- One Arduino Uno micro-controller board based on the ATmega328.
- During the various tests we used as actuators: a speaker, a servomotor, a DC motor, and several LEDs.

## IV. EVALUATION AND DISCUSSION

In this section, we describe with all detail the methodology that we did to evaluate our hypotheses. After that, we show the results that we obtained in our evaluation. We divided the evaluation in two phases: manual pictures and automatic pictures. We are going to explain each sub-section through these two phases.

### *A. Methodology*

The main objective of this evaluation process is verify our hypotheses:

- Could we insert the use of Computer Vision in the Internet of Things?
- Could we use the pictures from an IP Camera as a sensor?
- Could we obtain a good accuracy to automate or semi-automate this kind of events?

We demonstrated the possibility of the use of Computer Vision in the Internet of Things in the previous section, in Implementation. Then, we demonstrated our first hypothesis. Now, we are to explain how to try to validate if our second hypothesis is possible. For validating the second hypothesis, we used two different phases but both with the same objective: to evaluate the accuracy of the Computer Vision module using pictures to detect people. In both phases, we used pictures without people and pictures with people. Then, we used the pictures without people to detect the false positives and true negatives cases in our module. With the pictures with people, we also obtained the false negatives and true positives cases. For these both phases, we use the module without the Midgar interaction because we need to obtain the picture with the green rectangle to evaluate in a quantitative way if our module works well. Our two phases are the following:

- **Phase 1 - Manual pictures**: in this first one phase, we used pictures that we took with the Canon IP camera inside our laboratory manually. In this phase, we tested the Computer Vision module with isolated pictures without any relationship with the rest.
- **Phase 2 - Automatic pictures**: in the second phase, we used picture sequences that the camera sent us when it detected some movement in the laboratory with its sensors. In this case, the pictures of the sequence to evaluate have a relation between themselves because the pictures belong to the same movement. With this case, we wanted to try the Computer Vision module with a picture sequence to improve the detection algorithm using the relationship of the pictures of one same moment.

For these both phases, we used the same model, the OpenCV's example 'haarcascade_upperbody.xml'. This is important, because we have to evaluate using under the same conditions to obtain a truly evaluation. This is a default-trained model of the library OpenCV. This model can only detect the upper body of the people and it has a low accuracy because needs a very good pictures. Then, it has a low accuracy because it needs very good pictures with people in the correct position. However, if we need a better accuracy, we could use another model or create a new model. For taking the pictures that we used in this evaluation, we placed the IP camera in the middle of our laboratory in position to watch the entrance door as we show in Figure 4.

**Figure 4 Situation of the Canon Camera IP in the laboratory with the background that we used to take the pictures**

*1) Phase 1: Manual pictures*

For this first phase, we used the manual mode of the IP camera. We divided the pictures in two folders according to pictures without people and pictures with people. We analyse each folder with our module. For each folder, we obtained a new folder with the detected pedestrian inside a green rectangle. Then, we reviewed manually each picture according to the expected result because maybe, the green rectangle could mark an incorrect thing like a wardrobe or a signal instead of a person. In that case, we count the picture as wrong.

For this phase, we took 160 pictures manually: 64 with people and 96 without people. In Figure 5, we show an example with three pictures: two with a person and another without people. Afterwards, we process these pictures with our Computer Vision module to detect the accuracy of our module. With this test, we tried to evaluate the accuracy of the model that we use in our Computer Vision module.


**Figure 5 Example with indoor pictures took manually**

*2) Phase 2: Automatic pictures*

In the second phase, we programmed the camera to send a picture when the camera detected some movement in the area. In this case, the camera sent a picture sequences from the first moment that the camera detects the movement until the last detection. Then, we are going to analyse all the sequence to obtain if this is a valid method to use the Computer Vision with an IP camera as a sensor. In Figure 6, we show some pictures of this sequences about how the camera detected the initial movement in the first picture and after, it continues sending one picture per second until the movement ceases. The camera can send from one picture per second to thirty pictures per second. However, to do the evaluation, we selected the maximum value in the camera's configuration.


**Figure 6 Example of sequence of the pictures that the Canon IP Camera send when detects movement**

Then, the number of pictures depend on the movement time. The camera sent each picture to another web service created in Python with Flask to avoid the interaction of Midgar and to test only our Computer Vision module. In this case, we intended to evaluate if we could obtain an improvement of system using the camera's sensor or maintain the same level. Besides, this way allows us a reduction of the network traffic, a reduction of the necessary process computer, and avoid using the streaming option. For this phase, we used 979 pictures divided in 17 movement events, from which 8 sequences with 817 pictures are movement contain with people and 9 sequences with a total of 162 pictures are movement without people. Clarify, the sequences have many pictures without people because they have only a half body or an arm. Then, these sequences contain many invalidated pictures but these pictures are part of the sequence. For the negative sequences, we used the movement of different object in front of the camera like balls, mugs, or papers.

### *B. Results*

In this section, we are going to show the results. To facilitate and analyse in a better mode, we design a table for both phases. These tables contain the result of testing the pictures with our Computer Vision module and the results of the module classified in the different four groups:

- True Negative: pictures without people with a negative result. This is the best result for pictures without people because it means that the module does not detect people in pictures without people.
- False Positive: pictures without people with a positive result. This is the wrong case when we search people in pictures without people because is when the module says that it found people.
- False Negative: pictures with people with a negative result. This is the worst result because is when the module analyses pictures with people but the module did not detect people in that picture.
- True Positive: pictures with people with a positive result. This is the best result when the module analyses pictures with people because it is when the module found people.

Next, in the phase 1, we are going to describe the result with manual pictures. After that, we are going to show the second sub-section, which contains the phase 2 with the result of using the camera's sensor and analyse sequences of pictures instead of an isolated picture.

#### *1) Phase 1: Manual pictures*

Table 1 shows the result of the evaluation of the manual pictures. We can see in this table the evaluation of the 160 pictures. These was divided in 96 pictures without people negatives and 64 pictures with pictures.

| Types | | People | |
|---|---|---|---|
| | | Negative (96) | Positive (64) |
| Result | Negative | True Negative<br>96 / 100% | False Negative<br>57 / 89.062% |
| | Positive | False Positive<br>0 / 0% | True Positive<br>7 / 10.937% |

**Table 1 Results of the evaluation of the manual pictures**

Figure 7 Three pictures of the true positive cases

By analysing Table 1, we can suggest the following interpretations:

- We used 96 pictures without people. The Computer Vision module analyses 96 of these pictures as negative. This represents a success of 100%. This is the best accuracy for the negative cases because has the highest success in the true negative cases.
- To the positive cases, we used 64 pictures. The Computer Vision module analysed 57 of these pictures, corresponding with the 89.062% of the total, as negative cases. It is a very high unsuccessful result because the module does not detect people in pictures with people.
- The Computer Vision module obtained only a 10.937% of success in the positive cases. It is a very low accuracy. In these cases, the module obtained nine pictures, but when we analysed those pictures manually, we saw that two of those nine pictures had the green rectangle in the wrong place. For that, we count those two pictures as false negative. We show that two cases in Figure 8. As you can see, in the first one, the module detects the t-shirt as an upper body. Meanwhile, the second one detects the sign as the head. In other cases, the picture contains a blurry person, the person is not exactly in front of the camera, or the picture does not contain the full body of the person. In Figure 7, we show three of the seven pictures that the Computer Vision module detected as true positive cases.
- The Computer Vision module with the model that we used does not have false positive cases because it has a 0% of error. However, it has a very low accuracy in the true positive cases and it has a very high false accuracy in the negative cases.


Figure 8 The two pictures that we discarded because the wrong detection

*2) Phase 2: Automatic pictures*

In this second phase, we analysed the pictures of the seventeen sequences. We used nine sequences without people and eight with people. We analysed the pictures but, in this case, we analysed all the sequence as a one item. We can see some of the detected pictures in Figure 9.

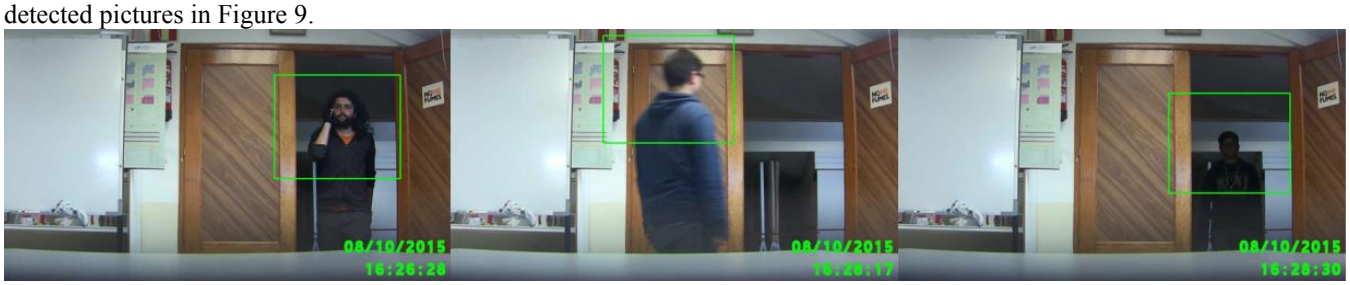


Figure 9 Collage with pictures of the True Positive sequences

In Table 2, we show the results of applying our Computer Vision module to the picture sequences.

| Types according on the sequence | | People | |
|---|---|---|---|
| | | Negative (9) | Positive (8) |
| Result | Negative | True Negative<br>8 / 88.888% | False Negative<br>0 / 0% |
| | Positive | False Positive<br>1 / 11.111% | True Positive<br>8 / 100% |

**Table 2 Results of the Computer Vision module analysing the sequences**

We can suggest the next interpretations analysing the Table 2:

- We used nine negative cases and we obtained eight true negative cases. This represents the 88.888% of the total. It is a high success.
- In this second phase, we obtained one false positive case. This represents the 11.111%. It means that the module recognised one object as an upper body of a person. In Figure 10, we can see the picture with this False Positive case. It detected an upper body in an area that contains a thumb, a mug, and the door.
- The Computer Vision module with our configuration obtained eight true positive cases for the eight positive sequences. It means the 100% of accuracy. This indicates that using of picture sequences, the accuracy of the module improves.
- The module had zero false negative. This indicates that in all sequences, the module at least detected one picture as person.

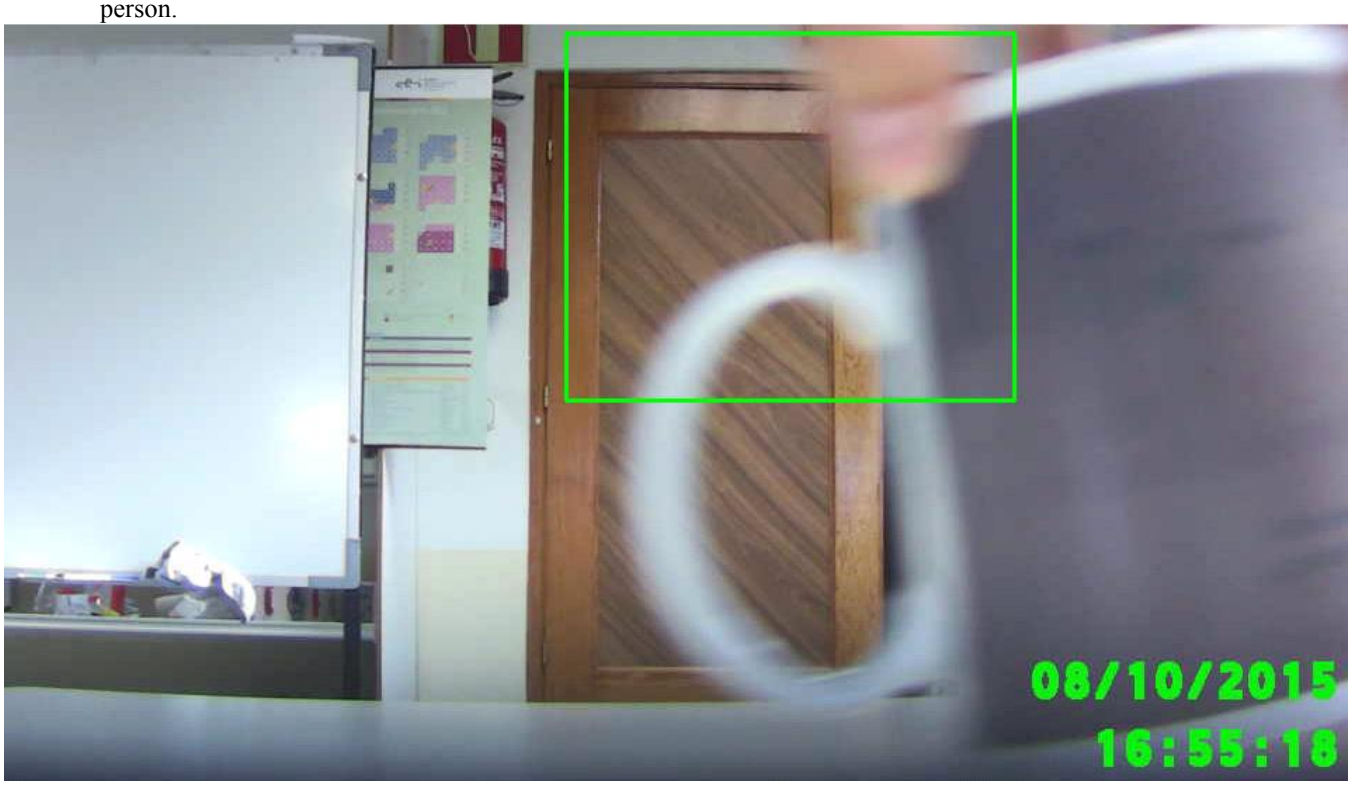



**Figure 10 The false positive picture**

Table 3 shows the information about each sequence: name, if it contains people or not, the total pictures that the camera sent when detects the movement, which they are the pictures that conform that sequence, the number, and percent of pictures that our module checked as picture with people, and the results according to our module.

| Sequence Name | Sequence with People? | Total Pictures | Identified Pictures | Detection % | Result |
|---|---|---|---|---|---|
| C1 | Yes | 78 | 1 | 1.282 | True Positive |
| C2 | Yes | 229 | 7 | 3.056 | True Positive |
| D1 | Yes | 53 | 15 | 28.301 | True Positive |
| D2 | Yes | 49 | 9 | 18.367 | True Positive |

| | | | | | |
|---|---|---|---|---|---|
| D3 | Yes | 54 | 22 | 40.740 | True Positive |
| D4 | Yes | 207 | 5 | 2.415 | True Positive |
| L1 | Yes | 84 | 6 | 7.142 | True Positive |
| L2 | Yes | 63 | 2 | 3.174 | True Positive |
| C | No | 12 | 0 | 0 | True Negative |
| F | No | 35 | 0 | 0 | True Negative |
| M | No | 9 | 0 | 0 | True Negative |
| P1 | No | 7 | 0 | 0 | True Negative |
| P2 | No | 25 | 0 | 0 | True Negative |
| P3 | No | 17 | 0 | 0 | True Negative |
| SP | No | 30 | 0 | 0 | True Negative |
| T | No | 20 | 1 | 5 | False Positive |

**Table 3 Information about the sequences**

According with Table 3, we can suggest the next interpretations:

- The detection percent does not have relation with the number of pictures: 'C2' has 229 picture and it only obtained a 3.052% of identified pictures. Meanwhile, 'D3', with 54 pictures, has a 40.740%. It depends on the picture quality: people's position, picture sharpness, full body shots because many pictures only contain the arm or a half body.
- The same case occurs with the Negative cases because 'F' has 35 pictures with a 0% of detections and 'T' has 20 pictures with a 5% of detection.

## V. CONCLUSIONS

In this paper, we present a possible solution to use Computer Vision in the Internet of Things. This could allow using IP Cameras and pictures as sensors. Besides, this could open the door to other similar applications like to automate things using Optical Character Recognition (OCR), face detection, or gestures.

In this way, we presented a possible architecture to integrate Computer Vision in an IoT platform, in our case Midgar. This Computer Vision module, analyses the picture, or, in our case, picture sequences, and returns to the IoT platform a Boolean result as if it was a button sensor. However, we showed that if you have a weak model, the Computer Vision module could have a low accuracy. For instance, we obtained in the first phase an **89.062% of False Negative** but a **100% of True Negative when** we analyse the isolated picture. Nevertheless, as we proposed to analyse all the sequence, we obtained better results: **100% of True Positives** and an **88.888% of True Negatives**. This was possible because we centred our module in the analysis of all the sequence of movement. However, our module has one fail, but it was a False Positive. Then, for important things like to detect thieves or robbers, it is better, in our humble opinion, to have False Positives instead of False Negatives. Clearly, it is not the best result but we could improve the result with a better model or maybe, we could create our own model to our exact case. As well as, we saw that the number of pictures of sequences does not affect to the result because some cases with many pictures have the less detection percent. In the case of **'C2' with 229 picture and a 3.052% of identified pictures**. The contrary with **'D3', with 54 pictures and a 40.740%**. In this case, the important thing is the quality of pictures. For that, probably is better to analyse sequences instead of isolated pictures, which is the thing that we do.

In function of this article, we can say that it is possible the use of Computer Vision in the IoT. Besides, with a very interesting application. In addition, we show that we can use pictures as sensors and with a model not perfect or weak because if we analyse all the sequence that we need, we can improve the result. However, it is better and more recommended the use of a good model because we could improve the accuracy, for instance, in our case, of the False Positives.

Automating is one of the ways of the Internet of Things. In this way, if we could improve and add new functionalities with Computer Vision, we could improve a lot the use level of the Internet of Things in our daily life and we could create new ways to communicate us with our environment. In this case, we use our research to automate and improve the security of our homes, towns, cities, and the Earth. An automatisation to help in our daily life or our job.

## VI. FUTURE WORK

This proposal is a small possible improvement of all the necessary problems and improvements that the IoT needs. For that, there are many ways to continue this research. In the next points, we propose some ways of continuing from this research:

- **Creation of a Domain-Specific Language using Model-Driven Engineering to facilitate the Computer Vision**: In our proposal, we offered the possibility to detect only pedestrians in the pictures because you need to extract the features and train the model to detect another thing. For that, one possibility is to offer a Domain-Specific Language to facilitate these steps in an IoT platform.

- **Creation of a system to control and define the different actions of an IP camera**: In this proposal, we used the IP camera as a sensor. In other proposals, they used the camera like an actuator but only to take picture. One possible way would be to develop an API to allow the total camera control like move it, change its properties, and so on.
- **Scalability of IoT platforms:** The computer vision need many computer resources. Then, in the case that the computer receives many pictures or has many IP cameras, the computer needs more time to process the whole information. Then, another future work way could be the study and test of different implementations in the IoT platforms to obtain a correct way of supporting the maximum number of IP Cameras, or how to discard many repeated pictures.
- **Optical Character Recognition or Face detection:** Other field in Computer Vision are the character detection and the face detection. Then, we could automate our doors to open it when it detects a determinate identification card or face. However, for that, we have to check different secure options because it is a very problematic field because the security that it needs.
- **Gesture recognition:** In this proposal, we analysed a sequence to detect people. Another possibility is to analyse the sequence to detect a determinate gesture to trigger an action. Then, we can use a picture sequence to detect a determinate gesture, which could throw a determinate action, previously assigned. It will allow using the pictures as a sensor with different outputs.

## VII. ACKNOWLEDGEMENTS

This work was performed by the 'Ingeniería Dirigida por Modelos MDERG' research group at the University of Oviedo under Contract No. FC-15-GRUPIN14-084 of the research project 'Ingeniería Dirigida Por Modelos MDERG'. Project financed by PR Proyecto Plan Regional.

## VIII. REFERENCES

[1] Fundación Telefónica, La Sociedad de la Información en España 2014, 2015. http://www.fundaciontelefonica.com/arte_cultura/publicaciones-listado/pagina-item-publicaciones/?itempubli=323.

[2] G.G. Meyer, K. Främling, J. Holmström, Intelligent Products: A survey, Comput. Ind. 60 (2009) 137–148. doi:10.1016/j.compind.2008.12.005.

[3] G. Kortuem, F. Kawsar, D. Fitton, V. Sundramoorthy, Smart objects as building blocks for the Internet of things, IEEE Internet Comput. 14 (2010) 44–51. doi:10.1109/MIC.2009.143.

[4] L. Atzori, A. Iera, G. Morabito, The Internet of Things: A survey, Comput. Networks. 54 (2010) 2787–2805. doi:10.1016/j.comnet.2010.05.010.

[5] K. Gama, L. Touseau, D. Donsez, Combining heterogeneous service technologies for building an Internet of Things middleware, Comput. Commun. 35 (2012) 405–417. doi:10.1016/j.comcom.2011.11.003.

[6] U.S. Department of Transportation, Livability, (2015). http://www.transportation.gov/livability/101 (accessed September 25, 2015).

[7] A.J. Jara, Y. Sun, H. Song, R. Bie, D. Genooud, Y. Bocchi, Internet of Things for Cultural Heritage of Smart Cities and Smart Regions, in: 2015 IEEE 29th Int. Conf. Adv. Inf. Netw. Appl. Work., IEEE, Gwangiu, 2015: pp. 668–675. doi:10.1109/WAINA.2015.169.

[8] H. Gu, D. Wang, A Content-aware Fridge Based on RFID in Smart Home for Home-Healthcare, in: Adv. Commun. Technol. 2009. ICACT 2009. 11th Int. Conf., Phoenix Park, 2009: pp. 987–990.

[9] S. Luo, H. Xia, Y. Gao, J.S. Jin, R. Athauda, Smart Fridges with Multimedia Capability for Better Nutrition and Health, in: Ubiquitous Multimed. Comput. 2008. UMC '08. Int. Symp., Ieee, Hobart, ACT, 2008: pp. 39–44. doi:10.1109/UMC.2008.17.

[10] M. Rothensee, A high-fidelity simulation of the smart fridge enabling product-based services, in: 3rd IET Int. Conf. Intell. Environ.

(IE 07), Iee, 2007: pp. 529–532. doi:10.1049/cp:20070420.

[11] E. Borgia, The Internet of Things vision: Key features, applications and open issues, Comput. Commun. 54 (2014) 1–31. doi:10.1016/j.comcom.2014.09.008.

[12] C. Hao, X. Lei, Z. Yan, The application and Implementation research of Smart City in China, in: Syst. Sci. Eng. (ICSSE), 2012 Int. Conf., Dalian, Liaoning, 2012: pp. 288–292. doi:10.1109/ICSSE.2012.6257192.

[13] A. Broring, P. Maue, C. Malewski, K. Janowicz, Semantic mediation on the Sensor Web, in: 2012: pp. 2910–2913.

[14] C.G. García, J.P. Espada, E.R.N. Valdez, V.G. Diaz, Midgar: Domain-Specific Language to Generate Smart Objects for an Internet of Things Platform, in: 2014 Eighth Int. Conf. Innov. Mob. Internet Serv. Ubiquitous Comput., IEEE, Birmingham, United Kingdom, 2014: pp. 352–357. doi:10.1109/IMIS.2014.48.

[15] F.E. Murphy, M. Magno, L. O'Leary, K. Troy, P. Whelan, E.M. Popovici, Big brother for bees (3B) - Energy neutral platform for remote monitoring of beehive imagery and sound, in: 2015 6th Int. Work. Adv. Sensors Interfaces, IEEE, Gallipoli, 2015: pp. 106–111. doi:10.1109/IWASI.2015.7184943.

[16] C.G. García, C.P. García-Bustelo, J.P. Espada, G. Cueva-Fernandez, Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios, Comput. Networks. 64 (2014) 143–158. doi:10.1016/j.comnet.2014.02.010.

[17] The US National Intelligence Council, Six Technologies with Potential Impacts on US Interests out to 2025, 2008.

[18] K.T. Nguyen, M. Laurent, N. Oualha, Survey on secure communication protocols for the Internet of Things, Ad Hoc Networks. (2015). doi:10.1016/j.adhoc.2015.01.006.

[19] T. Polk, S. Turner, Security Challenges For the Internet Of Things, in: Work. Interconnecting Smart Objects with Internet, Prague, 2011. https://www.iab.org/wp-content/IAB-uploads/2011/03/Turner.pdf.

[20] L. Sanchez, L. Muñoz, J.A. Galache, P. Sotres, J.R. Santana, V. Gutierrez, et al., SmartSantander: IoT experimentation over a smart city testbed, Comput. Networks. 61 (2014) 217–238. doi:10.1016/j.bjp.2013.12.020.

[21] I. International Telecommunication Union, Overview of the Internet of things, (2012) 14. https://www.itu.int/rec/T-REC-Y.2060-201206-I.

[22] J.A. Stankovic, Research Directions for the Internet of Things, IEEE Internet Things J. 1 (2014) 3–9. doi:10.1109/JIOT.2014.2312291.

[23] M. Delamo, S. Felici-Castell, J.J. Pérez-Solano, A. Foster, Designing an open source maintenance-free Environmental Monitoring Application for Wireless Sensor Networks, J. Syst. Softw. 103 (2015) 238–247. doi:10.1016/j.jss.2015.02.013.

[24] P. Martinez-Julia, E.T. Garcia, J.O. Murillo, A.F. Skarmeta, Evaluating Video Streaming in Network Architectures for the Internet of Things, in: 2013 Seventh Int. Conf. Innov. Mob. Internet Serv. Ubiquitous Comput., IEEE, Taiwan, 2013: pp. 411–415. doi:10.1109/IMIS.2013.76.

[25] A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, Internet of Things for Smart Cities, IEEE Internet Things J. 1 (2014) 22–32. doi:10.1109/JIOT.2014.2306328.

[26] R. Jesner Clarke, Smart Cities and the Internet of Everything: The Foundation for Delivering Next-Generation Citizen Services, (2013) 1–18. http://www.cisco.com/web/strategy/docs/scc/ioe_citizen_svcs_white_paper_idc_2013.pdf.

[27] S. Mitchell, N. Villa, M. Stewart-Weeks, A. Lange, The Internet of Everything for Cities: Connecting people, Process, Data, and Things to Improve the "Livability" of Cities and Communities, (2013) 1–21. http://www.cisco.com/web/strategy/docs/gov/everything-for-cities.pdf.

[28] Vienna University of Technology, European Smart Cities, Http://www.smart-Cities.eu. (2015). http://www.smart-cities.eu (accessed September 25, 2015).

[29] A. Caragliu, C. Del Bo, P. Nijkamp, Smart Cities in Europe, J. Urban Technol. 18 (2011) 65–82. doi:10.1080/10630732.2011.601117.

[30] K.A. Hribernik, Z. Ghrairi, C. Hans, K. Thoben, Co-creating the Internet of Things - First Experiences in the Participatory Design of Intelligent Products with Arduino, in: Concurr. Enterprising (ICE), 2011 17th Int. Conf., Aachen, 2011: pp. 1–9.

[31] J. McCarthy, M.L. Minsky, N. Rochester, C.E. Shannon, A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, AI Mag. 27 (2006) 12–14. doi:10.1609/aimag.v27i4.1904.

[32] C. Wang, N. Komodakis, N. Paragios, Markov Random Field modeling, inference & learning in computer vision & image understanding: A survey, Comput. Vis. Image Underst. 117 (2013) 1610–1627. doi:10.1016/j.cviu.2013.07.004.

[33] LogMeIn, Xively, (2013). https://xively.com/ (accessed July 29, 2015).

[34] Exosite, Exosite, (2013). http://exosite.com/ (accessed July 29, 2015).

[35] LORD MicroStrain, Sensor Cloud, (n.d.). http://www.sensorcloud.com/ (accessed July 29, 2015).

[36] Etherios, Etherios, (2008). http://www.etherios.com/ (accessed July 29, 2015).

[37] ThingWorx™, ThingWorx, (2015). http://www.thingworx.com/ (accessed July 29, 2015).

[38] Carriots, Carriots, (2011). https://www.carriots.com/ (accessed July 29, 2015).

[39] Paraimpu srl, Paraimpu, (2012). http://paraimpu.crs4.it/ (accessed July 29, 2015).

[40] QuadraSpace, (2010). http://www.quadraspace.org/ (accessed July 29, 2015).

[41] Department of Electrical and Electronic Engineering (University of Cagliari), SIoT, Http://platform.social-Iot.org/. (2012). http://platform.social-iot.org/.

[42] IoBridge, Thingspeak, (2013). http://www.thingspeak.com (accessed July 29, 2015).

[43] Oak Ridge National Laboratory, Sensorpedia, (2009). http://www.sensorpedia.com/ (accessed July 29, 2015).

[44] B.L. Gorman, D.R. Resseguie, C. Tomkins-Tinch, Sensorpedia: Information sharing across incompatible sensor systems, 2009 Int.

Symp. Collab. Technol. Syst. (2009) 448–454. doi:10.1109/CTS.2009.5067513.

[45] Microsoft, SenseWeb, Http://research.microsoft.com/en-Us/projects/senseweb/. (2008). http://research.microsoft.com/en-us/projects/senseweb/.

[46] A. Kansal, S. Nath, J. Liu, W.I. Grosky, SenseWeb : An Infrastructure for Shared Sensing, IEEE Multimed. 14 (2007) 8–13. doi:10.1109/MMUL.2007.82.

[47] EVRYTHNG, EVRYTHNG, (2012). https://www.evrythng.com/ (accessed July 29, 2015).

[48] Sen.se, Open.Sen.se, (2015). http://open.sen.se/ (accessed July 29, 2015).

[49] I. Nimbits, Nimbits, (2015). http://www.nimbits.com (accessed July 29, 2015).

[50] CyberVision Inc., KAA, (2014). http://www.kaaproject.org (accessed July 29, 2015).

# ANEXO VII. Bilrost: Connecting the Internet of Things through human social networks with a Domain-Specific Language

# Bilrost: Connecting the Internet of Things through human social networks with a Domain-Specific Language

Daniel Meana-Llorián; Cristian González García;
Jordán Pascual Espada
University of Oviedo
Oviedo, Spain
danielmeanallorian@gmail.com;
gonzalezgarciacristian@hotmail.com;
pascualjordan@uniovi.es

Vijay Bhaskar Semwal
Indian Institute of Information technology
Allahabad, India
vsemwal@gmail.com

***Abstract*—Nowadays, we have many Smart Objects near us connected to the Internet. These objects could make things together if an easy platform existed. There are many researches about interconnecting Smart Objects but we propose a novel approach using human social networks and a Domain-Specific Language. This approach makes easier the creation of intercommunications not only among objects but also between objects and humans. We propose a novel DSL that allows defining objects with sensors and actuators, and defining connections to social networks in order to publish the sensors' values and to do actions based on events triggered by messages. Moreover, with the DSL we will be able to establish external actions and status in order to make the objects be aware of the environment.**

***Keywords—Internet of Things; Smart Objects; Model-Driven Engineering; Domain-Specific Language; Social Networks***

## I. INTRODUCTION

Currently, the Internet of Things (IoT) is a term very popular because everybody has Smart Objects in his pocket like smartphones, wearables, tablets [1], and many others devices connected to the Internet. The possibilities of IoT are very diverse and they can include from Smart Homes [2]–[4] to Smart Earth [5], Smart Cities [5], [6] or any type of intelligence located in heterogenous and ubiquitous objects.

The interconnection among objects is very important in the context of the IoT. Smart Objects can be located in places separated by millions of kilometres or in unreachable places. Nevertheless, the objects can be connected and take decisions taking into account other Smart Objects. However, there are many problems related with the interconnections. First, there is not an unique standard or protocol that allow interacting with objects. Moreover, everybody has not knowledge about programming or creating programs that established the connection among objects or even, they have not knowledge about how interact with objects.

The aim of this proposal is the creation of applications using a Domain-Specific Language (DSL) created applying Model-Driven Engineering (MDE) With these applications, devices with sensors or/and actuators would be connected to other devices through social networks. We propose the creation of a language that define the different required aspects to define the devices, the data required by social networks, the sensors with their properties, the actuators with their properties and conditions that make the actuators run or make the devices publish their sensors' values.

Moreover, the usage of social networks prevent us from developing a specific platform and it is a solution more intuitive for people who use social networks daily.

The remainder of the paper is organised as follows. In Section II, we introduce involved topics in this research the works related. Section III shows our proposal, the Bilrost platform, how its DSL, called Bilrost-Specific Language (BSL), is. Section IV contains the conclusions of this paper and the possible future work that can be done from here.

## II. STATE OF THE ART

In order to present a theoretical frame of our proposal, we have to talk about the Internet of Things, Smart Objects, Online Social Networks, Model-Driven Engineering, and the related work.

### A. The Internet of Things

Currently, the IoT is one of the most important topics in many researches and business [7]–[10]. It could defined as the interconnection between heterogeneous and ubiquitous objects between themselves.

### B. Smart Objects

One important part in the IoT are the Smart Objects. They can be classified in three dimensions [4], [11]: Level of Intelligence, Location of Intelligence, and Aggregation level of Intelligence. The first one indicates the object's intelligence. The second one describes if the intelligence belongs to the object or to the network. The last dimension indicates if the intelligence is in the element, for example, when the object is composed by various elements and each one has their own intelligence, or in the container, when it is the contrary case.

Our proposal offers Smart Objects with an intelligence in the container. The other two characteristics depend on the implementation that the user made in the devices.

**978-1-4673-9939-5/16/$31.00 ©2016 IEEE**

### C. Online Social networks

However, according to interconnect different objects, we need some network. They could be WSAN, IoT Networks or, in our case, Online Social Networks (OSNs). OSNs provide many services to create applications like identity and authorisation services, Application Programming Interfaces (APIs) to read or write in timelines, receive updates, receive and send private messages, and so on. OSN is a basic piece of the Web 2.0 and the convergence of the real world with OSNs allows the development of new applications which interconnect things and humans [12].

Moreover, scientists of Ericsson [13] observed that if there is a analogy of using IoT technologies and social networks, people are capable of familiarising better with IoT technologies.

### D. Model-Driven Engineering

Model-Driven Engineering (MDE) appeared to solve software development problems that we have had since 1960s [14]. With MDE we can automate process to make the creation of repetitive process easier [15]. Applying MDE, developers can abstract the problem and automate some parts to facilitate the production of similar solutions. For instance, with the creation of Domain Specific Languages (DSL).

### E. Related work

Our proposal is a novel way of intercommunicating Smart Objects using actual human social networks instead of common web services to publish messages. These messages could be a sensor's status or calls to actuators' actions. However, there are not very similar researches related with this proposal. Many researches that deal with these issues about interconnecting objects through the Internet are based on Service Oriented Architecture (SOA) like REST [16]–[18]. However, we propose the usage of OSNs to establish the communication not only among Smart Objects but also between humans and Smart Objects. Related with the communcation among objects there are some researches like [13], [19] and between humans and objects could be considered [20]–[23].

There are some IoT platforms that allow interconnecting objects like Midgar [7], [24]. Midgar uses a graphic DSL to make the creation of the interconnection easier for people without out development skills. However, Midgar has the requirement of use a server and only allows interconnecting objects whereas Bilrost does not need any server because of the usage of OSNs and allows interconnecting objects with objects and humans with objects.

However, Social Access Controller (SAC) [20] could be an approach similar to our proposal. It uses social networks to share Smart Objects and it allows managing them and knowing their states. However, the usage of the social networks is very different. It uses it to share your Smart Objects with your friends in order to allow them to use it.

The interconnection not only can be done through social networks. Old researches used instant messages to interconnect objects and humans [21]–[23]. The principal disadvantage of this approach its the dependency of specific applications which were developed to the research whereas social networks are used by everybody.

## III. BILROST PLATFORM

Bilrost platform is our proposal to investigate the possibility of interconnecting Smart Objects through human social networks. The usage of human social networks gives us the possibility of interconnecting objects in an environment enough tested by millions of real users. Moreover, these networks not only allow us to interconnect objects, but also allow us to interconnect humans and objects. In this way, we would be making the Internet of Things bigger due to the addition of people to the network.

We propose the usage of a DSL to make easier the creation of the interconnections. However, in first steps, Bilrost will not be able to implement the logic to access to the sensors' values or to implement the logic to control the devices' actuators. The users will have to implement this logic in the project generated by Bilrost. So, Bilrost will be able to generate projects that will be able to interconnect objects through the social networks but the users will have to complete them with the specific logic for each device.

Therefore, our proposal will have the two principal parts: **projects generation** and **specific logic programming**.

In this section we will present how the generated projects will be able to interconnect objects through social networks and the two stages in which the user interaction will be required, the projects generation and the projects completion.

### A. Interconnecting objects through social networks

The aim of Bilrost will be the interconnection between objects through human social networks. Currently, there are many popular social networks that could be used in our proposal like Whatsapp, Facebook, Twitter, and many others. However, in the first stages of our proposal we will use Twitter because of its philosophy of short public messages and the common use of keywords, which are called hashtags.

Bilrost will create projects that devices will be capable of running. For that, we will implement a DSL that we call Bilrost Specific Language (BSL). The content of the BSL will be describe in the subsection III.B. After processing the BSL program, Bilrost platform will be able to generate projects where the connection to the social networks will have already been implemented.

The generated projects will connect devices like Raspberry Pi or Arduino to the social networks selected by the users. These projects will have specific parts that retrieve the sensors' values and do actions with the actuators. As Fig. 1 shows, the devices will be connected to social networks in order to publish the retrieve sensors's value and to do the actuators' actions invoked by social networks' messages.

In order to explain how Bilrost will work, we will use a simple example. We want to interconnect a Raspberry Pi and an Arduino with several sensors and actuators. Fig. 1 represents these Arduino and Raspberry Pi as devices with the number 2, their sensors with the number 3 and their actuators with the number 4. The Arduino is controlling a door lock and it has a light sensor. The Raspberry Pi has a presence sensor and it controls the lights of a room. With Bilrost, we will be able to interconnect the two devices through a social network like

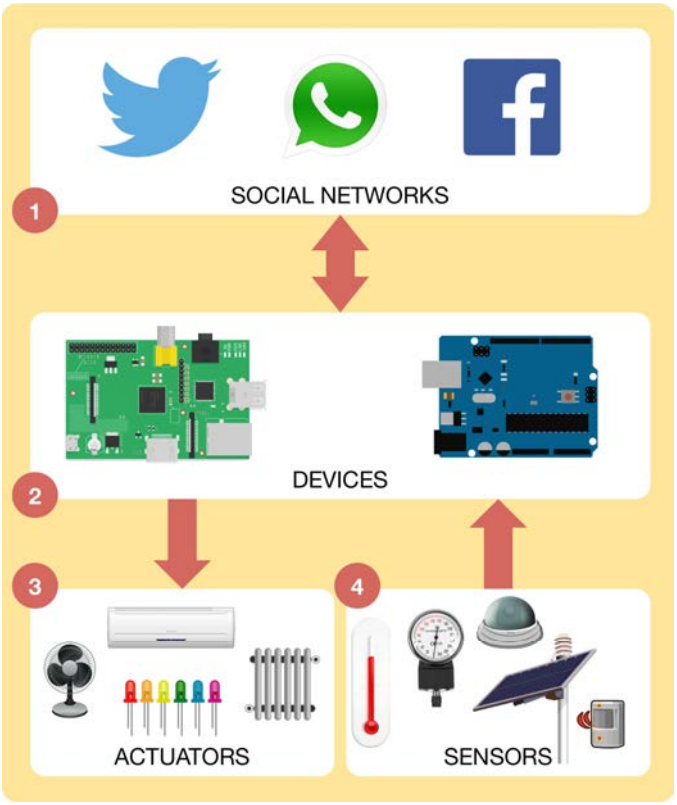



Fig. 1. Interconnection of devices through social networks.

Twitter which is represented in Fig. 1 with the number 1. If the Arduino detected that the light level is low, it would publish in Twitter, using special keywords, the level of the light. Then, the Raspberry Pi would receive the level of the light and it would turn on the light if the value sensed accomplish certain conditions. Moreover, if the Raspberry Pi detected a person, it would publish in Twitter that it detected anything and the Arduino would lock the door in order to prevent this person from going into the room.

This example will possible with Bilrost. Using the BSL, we will be able to specify the language of resulting projects, the keywords to use when publish messages or when listen messages, the usernames that the devices can listen, the tokens required by the social networks APIs, the actuators and their possible actions that the devices have, the sensors that the devices have and how the devices have to publish the sensors' status like the refreshing time. Furthermore, with the BSL, we will also be able to specify external states and actions. The external states will allow Bilrost to invoke actions of its own actuators when other devices publish a state that accomplish certain conditions specified with the BSL, and the external actions will allow Bilrost to invoke actuators' actions of other devices when the state of its own sensors accomplish certain conditions also specified with the BSL.

Besides, the usage of human social networks will open the possibility of handle human messages like they are devices messages. Thus, humans will be able to manage the devices. For example, a person will be able to turn off the heating system remotely publishing a message in a social network or a person will be able to know the temperature of its house searching the messages published by the device that has a temperature sensor located at his house.

However, there are a limitation of our proposal. In the early stages, the user will have to implement the access to the sensors' values and the invocation to actuators' actions inside of the projects generated by Bilrost. Bilrost will have the capacity of use the implementation of the users without any other interaction.

### *B. The user interaction*

Bilrost will use a DSL, called BSL, to define devices with actuators and sensors, and also, to define the social networks that it will use to connect the devices and the external actions and events that the devices will listen through the social network. However, the user interaction is required to define the BSL that generates the projects and to complete the generated projects as Fig. 2 shows. In this section we will explain these two stages.

#### *1) Projects generation*

In order to generate a project that connects a device to a social network, firstly, the user will have to define the device using BSL. With BSL, the user will be able to define the programming language of the generated project thinking in the device where the project will run. Running the project in a smartphone is not the same than running the project in a Raspberry Pi because the first one needs programs programmed with Java language and the second one needs programs programmed with Python language. However, due to our proposal, the users do not need advance language programming in these languages because they only will have to implement the access to the sensors' values and the actuators' actions, they will not implement the connection to social networks and many other aspects related with the connection.

Fig. 2 shows the process of projects generation with the number 1. This stage will start with the users of Bilrost. These users will have to define the BSL as they want. Afterwards, Bilrost will process the BSL file and it will generate a project that accomplish the rules defined in it the BSL.

The final project will be a project in a specific programming language for the device, and it will contain the logic required to connect the device to the social network specified in the BSL with the data required by the social network's API. Moreover, the project will have the necessary logic to invoke certain empty methods that users will have to complete, as we will explain in subsection III.B.2, and to accomplish the conditions specified in the BSL related with external states and actions as we explained in section III.A.

#### *2) Projects completion*

The project generation will not be the last stage in which the user interaction will be required. The users will have to complete the projects that Bilrost will generate with the logic that retrieves the sensors' value and with the logic that controls the actuators. Our proposal does not include the generation of specific code because we are focused on the interconnection of objects through social networks. Thus, the specific code will have to be coded by the users, at least in the first stages of the investigation.

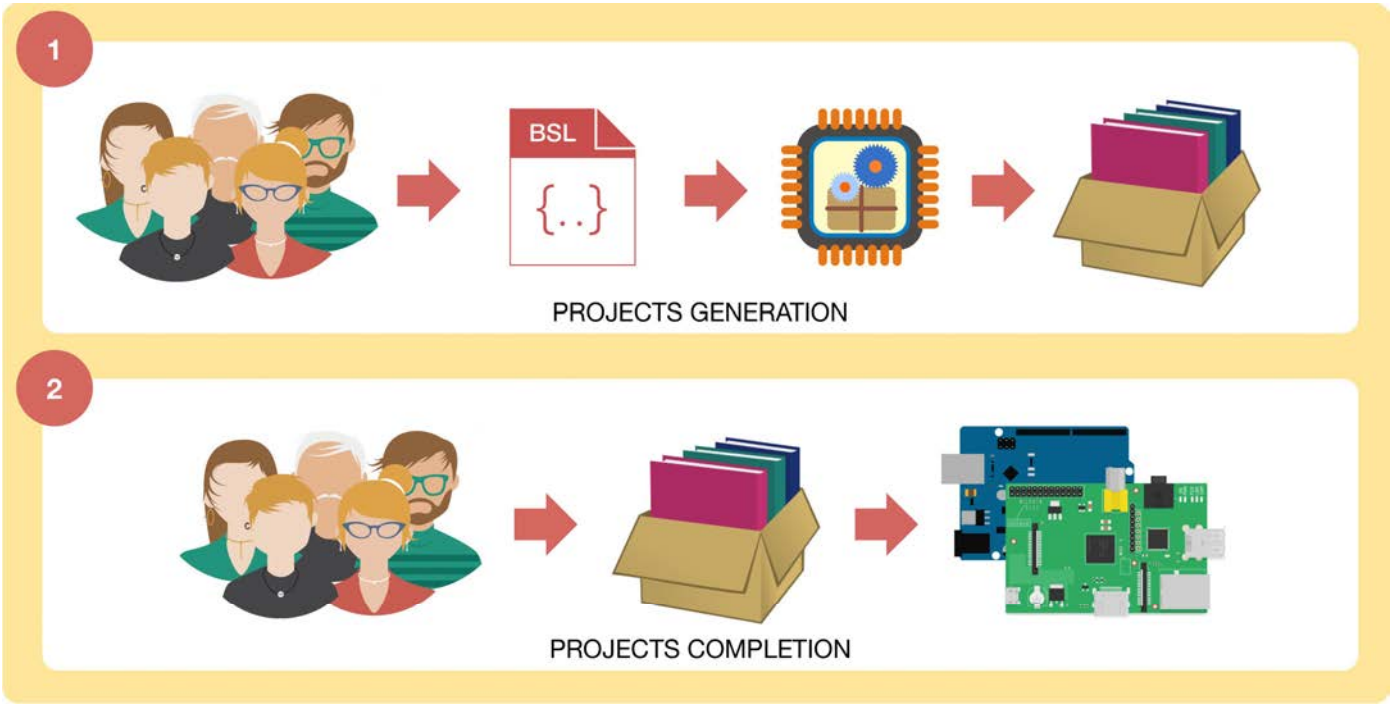



Fig. 2. Stages with user interaction: 1. Projects generation and 2. Projects completion.

Fig. 2 shows the process of projects completion with the number 2. This stage will start with the users of Bilrost. The users will have to take the generated project and update it. The projects will have methods with empty body but with descriptions that will explain to the users what they will have to implement and how they will used the existing code. Afterwards, the users will have to deploy the project in the devices and run it.

## IV. Conclusions

We presented our proposal about interconnecting Smart Objects from the Internet of Things through human social networks. The usage of human social networks opens the possibility of interconnect objects and humans. Thus, our proposal would make the Internet of Things bigger because of the addition of humans. In order to make the connection easier, we proposed the used of a Domain-Specific Language to define devices that are connected to social networks using specific keywords. The Bilrost Specific Language will also define the devices' actuators with their actions and the devices' sensors with the time that it will be used to decide when publish the sensed value. Moreover, the BSL will define external states that the devices will have to interpret and respond doing some action with their actuators, and external actions that the devices will be able to use in order to invoke actuators located in other devices.

However, there are more work that will be done related with this proposal in a future. One possible future work could be the addition of the specific implementation in the BSL like the way of getting the sensors' values or the logic of the actuators' actions. Moreover, social networks are not similar so, doing a investigation that analyses the available social networks in order to decide the best one to use with objets is a good future work too. Finally, we made our proposal thinking in the Internet of Things and devices like Arduino or Raspberry Pi but there are many others fields where our proposal could be used and a future work is to discover it.

## Acknowledgment

This work was performed by the "Ingeniería Dirigida por Modelos MDE-RG" research group at the University of Oviedo under Contract No. FC-15-GRUPIN14-084 of the research project "Ingeniería Dirigida Por Modelos MDE-RG". Project financed by PR Proyecto Plan Regional.

## References

[1] F. Telefónica, *La Sociedad de la Información en España 2014*. Grupo Planeta Spain, 2015.

[2] H. G. H. Gu and D. W. D. Wang, "A Content-aware Fridge based on RFID in smart home for home-healthcare," in *2009 11th International Conference on Advanced Communication Technology*, 2009, vol. 02, pp. 987–990.

[3] C. Han, J. M. Jornet, E. Fadel, and I. F. Akyildiz, "A cross-layer communication module for the Internet of Things," *Comput. Networks*, vol. 57, no. 3, pp. 622–633, 2013.

[4] K. a. Hribernik, Z. Ghrairi, C. Hans, and K.-D. Thoben, "Co-creating the Internet of Things - First experiences in the participatory design of Intelligent Products with Arduino," in *2011 17th International Conference on Concurrent Enterprising*, 2011, pp. 1–9.

[5] L. Hao, X. Lei, Z. Yan, and Y. ChunLi, "The application and implementation research of smart city in China," in *2012 International Conference on System Science and Engineering (ICSSE)*, 2012, pp. 288–292.

[6] R. Lea and M. Blackstock, "Smart Cities: An IoT-centric Approach," in *Proceedings of the 2014 International Workshop on Web Intelligence and Smart Sensing*, 2014, pp. 12:1–12:2.

[7] C. González García, B. C. Pelayo G-Bustelo, J. Pascual Espada, and G. Cueva-Fernandez, "Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios," *Comput. Networks*, vol. 64, pp. 143–158, 2014.

[8] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A

survey," *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[9] K. Gama, L. Touseau, and D. Donsez, "Combining heterogeneous service technologies for building an Internet of Things middleware," *Comput. Commun.*, vol. 35, no. 4, pp. 405–417, Feb. 2012.

[10] Lu Tan and Neng Wang, "Future internet: The Internet of Things," in *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, 2010, vol. 5, pp. V5–376–V5–380.

[11] G. G. Meyer, K. Främling, and J. Holmström, "Intelligent Products: A survey," *Comput. Ind.*, vol. 60, no. 3, pp. 137–148, Apr. 2009.

[12] M. Blackstock, R. Lea, and A. Friday, "Uniting online social networks with places and things," in *Proceedings of the Second International Workshop on Web of Things*, 2011, pp. 5:1–5:6.

[13] L. Atzori, A. Iera, and G. Morabito, "From 'smart objects' to 'social objects': The next evolutionary step of the internet of things," *IEEE Commun. Mag.*, vol. 52, no. 1, pp. 97–105, Jan. 2014.

[14] E. Dijkstra, "The humble programmer," *Commun. ACM*, vol. 15, no. October 1972, pp. 859–866, 1972.

[15] V. García-Díaz, H. Fernández-Fernández, E. Palacios-González, B. C. P. G-Bustelo, O. Sanjuán-Martínez, and J. M. C. Lovelle, "TALISMAN MDE: Mixing MDE principles," *J. Syst. Softw.*, vol. 83, no. 7, pp. 1179–1191, Jul. 2010.

[16] D. Guinard and V. Trifa, "Towards the Web of Things : Web Mashups for Embedded Devices," in *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences)*, 2009, pp. 1–8.

[17] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," in *2010 Internet of Things, IoT 2010*, 2010, pp. 1–8.

[18] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the SOA-based internet of things: Discovery, query, selection, and on-demand provisioning of web services," *IEEE Trans. Serv. Comput.*, vol. 3, no. 3, pp. 223–235, 2010.

[19] L. Atzori, A. Iera, and G. Morabito, "SIoT: Giving a Social Structure to the Internet of Things," *IEEE Commun. Lett.*, vol. 15, no. 11, pp. 1193–1195, Nov. 2011.

[20] D. Guinard, M. Fischer, and V. Trifa, "Sharing using social networks in a composable Web of Things," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, 2010, pp. 702–707.

[21] J. Choi and C. W. Yoo, "Connect with things through instant messaging," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4952 LNCS, Springer, 2008, pp. 276–288.

[22] S. Aurell, "Remote controlling devices using instant messaging: building an intelligent gateway in Erlang/OTP," in *Proceedings of the 2005 ACM SIGPLAN workshop on Erlang*, 2005, pp. 46–51.

[23] A. Roychowdhury and S. Moyer, "Instant messaging and presence for sip enabled networked appliances," *Publ. Unkn.*, 2001.

[24] C. G. García, J. P. Espada, E. R. Núñez-valdez, and V. García-díaz, "Midgar : Domain-Specific Language to generate Smart Objects for an Internet of Things platform."

# ANEXO VIII. IntelliSenses: Sintiendo Internet de las Cosas

# IntelliSenses: Sintiendo Internet de las Cosas

# *IntelliSenses: Sensing The Internet of Things*

Daniel Meana-Llorián, Cristian González García, B. Cristina Pelayo G-Bustelo, Juan Manuel Cueva Lovelle
Departamento de Informática, Universidad de Oviedo
Oviedo, España
danielmeanallorian@gmail.com,
gonzalezgarciacristian@hotmail.com,
crispelayo@uniovi.es, cueva@uniovi.es

Victor Hugo Medina García
Universidad Distrital "Francisco José de Caldas"
Bogotá, Colombia
vhmedina@gmail.com

*Resumo* — **El auge de Internet de las Cosas permite que cada vez existan más dispositivos conectados a Internet y por tanto más información disponible. Sin embargo, hay muchos datos que por sí solos pueden no tener mucho valor pero que combinándolos pueden ser más valiosos. Esto también es aplicable a la información obtenida por los sentidos del cuerpo humano. El cuerpo humano hace uso de los sentidos para recopilar datos del entorno, y combina esa información para tomar las decisiones oportunas. Nuestra propuesta se basa en el mismo principio, la creación de un sistema dividido en varios subsistemas que simulen los sentidos humanos y un subsistema central que orqueste y tome decisiones en función de los datos obtenidos por los demás subsistemas. De este modo, y al igual que hace el cuerpo humano con los sentidos, la información obtenida por un subsistema se complementaría con la información obtenida de los demás subsistemas.**

***Palabras Clave – Internet de las Cosas; Sensores; Ingeniería Dirigida por Modelos; Inteligencia artificial; Sentidos.***

*Abstract* — **The popularity of the Internet of Things allows the existence of many connected devices and thus, the quantity of data available is higher. However, many data are unusable if they are alone or out of context but a combination of these data can be valuable. This idea is also present in the information gathered by the human senses. The human body uses the senses to gather data about the environment in order to take the best possible decisions taking into consideration the whole environment. Our proposal is based on the same idea. We propose the creation of several subsystems that would simulate the human senses and another subsystem capable to manage them, and take decisions based on the data gathered by all subsystems. In this way, the data gathered by one subsystem would be complemented with the data gathered by the rest of systems as the human body does.**

***Keywords - The Internet of Things, Sensors, Model-Driven Engineering; Artificial Intelligence; Senses.***

## I. INTRODUCCIÓN

En la actualidad, muchos dispositivos se encuentran conectados continuamente a Internet, desde teléfonos inteligentes hasta coches; televisores; relojes y muchos otros dispositivos. Esta tendencia es creciente con una previsión de crecimiento que pasa de 3750 millones de objetos conectados en 2014 a 25000 millones de objetos conectados en 2020 [1]. Este crecimiento de objetos conectados a Internet ha hecho que aparezca el término Internet de las Cosas (*The Internet of Things* o IoT en inglés) y que puede ser definido como la interconexión entre objetos heterogéneos y ubicuos a través de Internet [2], [3].

Las posibilidades de Internet de las Cosas son muy amplias. En la actualidad existen diversos campos donde se aplican conceptos relacionados con IoT como las casas inteligentes o *Smart Homes* [4], los pueblos como los *Smart Towns* [5] y las *Smart Cities* [6]–[10] o ciudades inteligentes, en el medioambiente por medio de lo conocido como *Smart Earth* [10] o cualquier inteligencia distribuida en objetos heterogéneos y ubicuos, también denominados *Smart Objects*.

Internet de las Cosas puede ayudar a las personas en muchas situaciones ayudándolos a percibir el entorno y facilitándoles la vida. En [11] usan tecnologías de IoT y RFID, para ayudar a personas invidentes a usar el transporte público. En [12] también usan RFID junto a códigos QR (*Quick Response*) para ayudar a personas invidentes, pero en este caso, para moverse por interiores debido a las limitaciones del GPS. Otro enfoque útil para personas invidentes es el uso de cámaras para reconocer objetos o texto y transmitir esa información al usuario como en [13], donde han desarrollado un sistema que utiliza reconocimiento de caracteres y conversión de texto a audio para leer textos a personas invidentes. En [14] combinan ambos enfoques desarrollando un sistema que otorga capacidad de reconocimiento visual y movimiento autónomo a personas invidentes a través de un sensor laser, un sensor con unidad de medición inercial (IMU) y una cámara.

IoT no solo es útil para personas invidentes, sino que también permite simular otros aspectos de las personas como hacer uso del olfato o el gusto para clasificar sabores. En [15] hacen uso de sensores químicos para clasificar cervezas en función del olor y en [16] hacen uso de sensores para clasificar vinos en función del sabor. El sentido del olfato y el gusto están muy relacionados ya que las personas los utilizamos conjuntamente. Por esto, en [17], han desarrollado un sistema que combina sensores que simulan ambos sentidos con el fin de mejorar el reconocimiento de sabores. Otro sistema que se puede comparar con otro sentido, el tacto, es el propuesto en [18], en el que han desarrollado un sistema que permite integrarse en superficies y medir la presión que se ejerce sobre ellas. Mediante el análisis de la presión se puede deducir la acción que el usuario está realizando, como por ejemplo, ejercicios de Yoga.

Aunque los sistemas propuestos intentan ser análogos a los sentidos de los humanos no hacen uso de los 5 sentidos. Usando todos los sentidos y un subsistema inteligente central, un sistema podría ser capaz de interpretar los diferentes datos de una manera más próxima a las personas. Debido a esto, nosotros proponemos la creación de sistemas que simulen todos los sentidos, además de una unidad central e inteligente que las conecte. Para ello hemos comenzado el desarrollo de prototipos acerca de cada uno de los sentidos, como veremos en la siguiente sección.

Aún con el auge y las posibilidades de Internet de las Cosas, este no está muy extendido entre la población de la calle puesto que requiere ciertos conocimientos relacionados. Una posible solución sería integrar Internet de las Cosas en tareas frecuentes como pueden ser las redes sociales ya que según científicos de Ericsson [19], la analogía entre el uso de tecnologías relacionadas con Internet de las Cosas y redes sociales permiten que las personas sean capaces de familiarizarse mejor con esas tecnologías. Por esta razón, una de nuestras propuestas está relacionada con el uso de redes sociales para intercomunicar los subsistemas propuestos. Con el fin de facilitar la configuración del sistema, aplicamos técnicas de Ingeniería Dirigido por Modelos (*Model-Driven Engineering* o MDE en inglés). MDE apareció para resolver los problemas del desarrollo del software que existen desde 1960s [20]. El uso de MDE nos permite elevar el nivel de abstracción y automatizar el proceso de configuración de los sistemas gracias al uso de Lenguajes de Dominio Específico (*Domain-Specific Language* o DSL).

Durante las siguientes secciones mostraremos nuestras propuestas comparándolas con los 5 sentidos del cuerpo humano para finalmente concluir con los beneficios de nuestra propuesta y el trabajo futuro que se deberá hacer de aquí en adelante.

## II. CONTRIBUCIONES

Nuestra propuesta se basa en la creación de un sistema que sea capaz de interpretar el mundo que lo rodea a modo de distintos subsistemas que representen los 5 sentidos del cuerpo humano y actuar en función de la combinación de los datos obtenidos. De esta manera se podrían entrelazar datos de diferente tipo, entre los que puede haber en algunos casos datos irrelevantes que combinándolos pueden formar un contexto donde cobren un mayor sentido. Para poder llegar a esto es necesaria la existencia de un sistema inteligente que coordine y comunique los distintos subsistemas. Nosotros proponemos la creación de varios subsistemas que puedan simular individualmente cada sentido o parte de ellos y un subsistema central que intercomunique todos los demás subsistemas permitiendo interpretar todos los subsistemas conjuntamente y por tanto tomar decisiones de manera combinada.

Los 5 sentidos del cuerpo humano son la vista, el oído, el olfato, el tacto y el gusto. Para realizar nuestra propuesta es necesario buscar una analogía entre estos sentidos y el mundo del Internet de las Cosas.

El sentido de la vista se podría modelar a través del uso de cámaras con la capacidad de reconocer objetos, colores y/o formas o sensores de ultrasonidos que permitan reconocer formas. El sentido del oído a través de sensores relacionados con cambios sonoros como aquellos capaces de identificar las variaciones del nivel de ruido o la frecuencia del sonido. También se podrían utilizar las redes sociales como parte del sentido del oído puesto que pueden servir como sistema de escucha de lo que está sucediendo en el mundo que nos rodea. Los sensores de olores y gases permitirían modelar el sentido del olfato mientras que los sensores químicos podrían modelar el sentido del gusto. Para modelar el sentido del tacto se podría recurrir a sensores de flexibilidad o presión. Otra opción sería la de usar sensores ambientales que puedan medir la temperatura o la humedad puesto que nosotros percibimos la temperatura a través de este sentido.

Todos estos sistemas deben ser capaces de comunicarse con otros dispositivos para compartir la información detectada y computarla, ya sea a través de internet, o a través de conexiones directas a los dispositivos.

Con el fin de actuar de mediador y combinar los datos obtenidos, debería aparecer un nuevo actor que puede ser comparado con un cerebro puesto que se sitúa en el centro de todos los sentidos y debe de orquestarlos. Este sistema debería contener cierto nivel de inteligencia que le permita analizar los datos de manera autónoma y decidir las consecuencias oportunas en función de los resultados obtenidos. Para ello se podrían aplicar técnicas de Inteligencia Artificial que permitan al sistema aprender de manera autónoma. No obstante, esto debería de ir acompañado de Big Data para tratar el gran volumen de datos que pueden llegar a aparecer.

A lo largo de esta sección hablaremos de nuestras propuestas para simular los 5 sentidos y de los sistemas propuestos para procesar e intercomunicar los sentidos. Para ello se introducirán varios prototipos ya desarrollados e ideas para futuros desarrollos.

### *A. Sentidos*

Como ya hemos mencionado, pretendemos crear una simulación de los sentidos humanos en el marco de Internet de las Cosas. A continuación, hablaremos más en detalle de la analogía de cada uno de los 5 sentidos.

#### *1) Vista*

El sentido de la vista permite a las personas reconocer formas y colores a través de imágenes procesadas por el cerebro. Nuestra propuesta se basa en algo similar, el uso de sensores para reconocer formas y/o colores que nos permitan identificar objetos que estamos buscando o que queremos evitar. Para ello se podrían usar diversos tipos de sensores como los sensores ultrasónicos que se basan en el cálculo de la distancia de un objeto al sensor en función del tiempo que tardan los ultrasonidos emitidos por el sensor en rebotar y volver al sensor. Combinando las distintas distancias se consigue obtener las diferentes formas de los objetos.

Nosotros hemos desarrollado un prototipo basado en visión por computador en vez de en el uso de sensores. Para ello usamos una cámara de vigilancia IP dotada con software que permite enviar imágenes a un servidor cuando se detectan cambios en la grabación.

El prototipo desarrollado analiza la grabación en dos fases. En la primera realiza un análisis de todas las imágenes recibidas por la cámara en busca de personas y en la segunda fase, analiza

las imágenes con resultados positivos en busca de rostros. Al completar la primera fase se envían las imágenes con los cuerpos detectados y marcados a usuarios subscritos al prototipo con el fin de notificarles la nueva presencia en la sala y al completar la segunda fase, si se encuentran rostros, se envían las imágenes a un *Bucket* de Amazon Web Services (AWS) con el fin de permitir un tratamiento futuro haciendo uso de tecnologías de Big Data. En la Fig. 1 se muestra un ejemplo del reconocimiento efectuado por el prototipo.

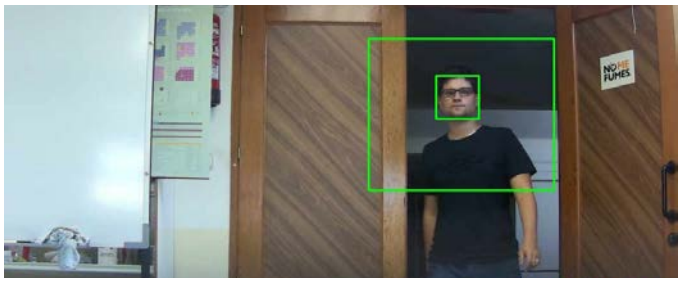


Figura 1. Reconocimiento efectuado por el prototipo de la vista.

Para la realización de este prototipo se utilizaron como tecnologías el lenguaje de programación Python 3.4 junto a la librería OpenCV 3.0.0 para visión por computador.

*2) Oido*

El sentido del oído nos permite identificar y reconocer sonidos de nuestro alrededor. Nuestro cerebro es capaz de asociar sonidos a otros elementos como palabras u objetos, lo que nos permite entender el lenguaje humano o reconocer que objeto está emitiendo sonidos. Nosotros proponemos el uso de sensores que puedan medir el nivel de ruido o la frecuencia sonora. Así, procesando la información obtenida de los sensores y mediante técnicas de aprendizaje proponemos realizar el reconocimiento de objetos. Además, con el fin de simular el entendimiento del lenguaje humano, proponemos la incorporación de las redes sociales. Mediante el análisis de las redes sociales se puede extraer información relevante para el sistema relacionada con el entorno que le rodea. Por ejemplo, tuits relativos a un accidente, a un evento deportivo, o cualquier tipo de suceso que tenga lugar en el entorno próximo.

Sin embrago, en el caso del oído, todavía no hemos desarrollado un prototipo funcional por lo que estas propuestas son una línea de investigación futura.

*3) Olfato*

Gracias al sentido del olfato podemos detectar cuando algo fuera de lo normal está sucediendo en el entorno y actuar en consecuencia. El sentido del olfato nos puede servir tanto para detectar situaciones buenas relacionadas con olores agradables como comida recién hecha, limpieza o situaciones cotidianas que son identificables por el olor, como para detectar situaciones que puede acarrear algún problema como escapes de gases, comida en mal estado o cualquier otra situación en la que mediante con el olor se pueda identificar un peligro. Nuestra propuesta se basa en el uso de sensores para identificar situaciones parecidas a las ya mencionadas en nuestro entorno. El sentido del olfato funciona mediante quimiorreceptores por lo que para simular su comportamiento se pueden usar sensores de gases. Mediante el uso de estos sensores se podría llegar a reconocer los distintos patrones de gases que permitirían identificar diferentes olores.

Nosotros hemos desarrollado un prototipo cuya función es controlar el nivel de gases de una sala y realizar acciones en función de una serie de reglas definidas con el fin de simular el sentido del olfato en lo que se refiere a reaccionar en función de la composición química del ambiente. Para ello hemos creado una aplicación web donde los usuarios pueden definir reglas seleccionando el gas que quieren controlar, el valor y la condición de control, y la acción a realizar, como por ejemplo enviar un email. En la Fig. 2 se muestra el hardware implicado en el prototipo.

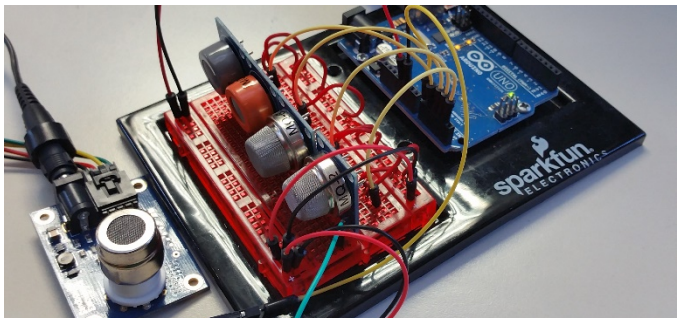



Figura 2. Hardware implicado en el prototipo del olfato.

Para la realización de este prototipo se utilizaron como tecnologías el lenguaje de programación JavaScript sobre Node.js 5.5.0. A nivel de hardware se han usado 5 sensores de gases: MG811, MQ3, MQ4, MQ7 y MQ8 capaces de medir dióxido de carbono ($CO_2$), alcohol ($C_2H_5OH$), metano ($CH_4$), monóxido de carbono (CO) e hidrógeno ($H_2$) respectivamente y una placa Arduino UNO.

*4) Gusto*

El sentido del gusto nos permite identificar sabores de alimentos o bebidas gracias a quimiorreceptores ubicados en la lengua. Mediante esta identificación podemos reconocer de que está compuesto un alimento o bebida y en algunos casos se puede saber si está en buenas condiciones o no. El gusto tiene una estrecha relación con el sentido del olfato puesto que se complementa con los olores para afinar la identificación. Nuestra propuesta referente a este sentido se basa en el uso de sensores químicos que permitan identificar la composición de distintos elementos, por ejemplo, mediante un sensor sumergido en líquidos. Para poder llegar a identificar sabores será necesario disponer de un sistema inteligente y entrenado mediante técnicas de aprendizaje e inteligencia artificial. Mediante esta propuesta, por ejemplo, se podrían identificar componentes dañinos en líquidos que deberían ser inocuos.

Sin embargo, en el caso del gusto, todavía no hemos desarrollado un prototipo funcional siendo estas propuestas líneas de investigación futuras.

*5) Tacto*

El tacto es uno de los sentidos más importantes del cuerpo humano. Este sentido nos permite percibir distintas características de los objetos de nuestro entorno como la forma, la suavidad, la rugosidad o cualquier otra característica física. También nos permite percibir características del medio como la

temperatura, la humedad o la presión. Para ello, nuestro cuerpo cuenta con diferentes tipos de receptores nerviosos que recogen la información de nuestro entorno y la envían al cerebro para que este la interprete. Entre estos receptores se encuentran quimiorreceptores, mecanorreceptores y termoreceptores.

A modo de prototipo relacionado con el sentido del tacto, hemos desarrollado un sistema capaz de reaccionar en función de la sensación térmica. Para ello tomamos el valor de la temperatura exterior a través de sensores localizados en un Arduino UNO y haciendo uso de la plataforma de IoT Midgar, de la que hablaremos más adelante, el valor de la humedad exterior a través de servicios web de ThinkSpeak y la temperatura interior a través de un sensor de temperatura conectado a una Raspberry Pi 2. La combinación de los factores exteriores teniendo en cuenta el cuadro de la Fig. 3, nos permite obtener la sensación térmica exterior.

| Temperatura (ºC) | Humedad (%) | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 55 | 60 | 65 | 70 | 75 | 80 | 85 | 90 | 95 | 100 |
| 20 | 16 | 16 | 17 | 17 | 17 | 18 | 18 | 19 | 19 | 19 | 19 | 19 | 20 | 20 | 20 | 21 | 21 | 21 | 21 | 21 | 21 |
| 21 | 18 | 18 | 18 | 19 | 19 | 19 | 19 | 19 | 20 | 20 | 20 | 20 | 21 | 21 | 21 | 22 | 22 | 22 | 22 | 22 | 23 |
| 22 | 19 | 19 | 19 | 20 | 20 | 20 | 20 | 20 | 21 | 21 | 21 | 21 | 22 | 22 | 22 | 22 | 23 | 23 | 23 | 23 | 24 |
| 23 | 20 | 20 | 20 | 20 | 21 | 21 | 22 | 22 | 22 | 23 | 23 | 23 | 23 | 24 | 24 | 24 | 24 | 24 | 24 | 25 | 25 |
| 24 | 21 | 21 | 22 | 22 | 22 | 22 | 23 | 23 | 23 | 24 | 24 | 24 | 24 | 25 | 25 | 25 | 25 | 26 | 26 | 26 | 26 |
| 25 | 22 | 23 | 23 | 23 | 24 | 24 | 24 | 24 | 24 | 24 | 25 | 25 | 25 | 26 | 26 | 26 | 27 | 27 | 27 | 28 | 28 |
| 26 | 24 | 24 | 24 | 24 | 25 | 25 | 25 | 26 | 26 | 26 | 26 | 27 | 27 | 27 | 27 | 28 | 28 | 29 | 29 | 29 | 30 |
| 27 | 25 | 25 | 25 | 25 | 26 | 26 | 26 | 27 | 27 | 27 | 27 | 28 | 28 | 29 | 29 | 30 | 30 | 31 | 31 | 31 | 33 |
| 28 | 26 | 26 | 26 | 26 | 27 | 27 | 27 | 28 | 28 | 28 | 29 | 29 | 29 | 30 | 31 | 32 | 32 | 33 | 34 | 34 | 36 |
| 29 | 26 | 26 | 27 | 27 | 27 | 28 | 29 | 29 | 29 | 29 | 30 | 30 | 31 | 33 | 33 | 34 | 35 | 35 | 37 | 38 | 40 |
| 30 | 27 | 27 | 28 | 28 | 28 | 28 | 29 | 29 | 30 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 39 | 40 | 41 | 45 |
| 31 | 28 | 28 | 29 | 29 | 29 | 29 | 30 | 31 | 31 | 31 | 33 | 34 | 35 | 36 | 37 | 39 | 40 | 41 | 45 | 45 | 50 |
| 32 | 29 | 29 | 29 | 29 | 30 | 31 | 31 | 33 | 33 | 34 | 35 | 35 | 37 | 39 | 40 | 42 | 44 | 45 | 51 | 51 | 55 |
| 33 | 29 | 29 | 30 | 30 | 31 | 33 | 33 | 34 | 34 | 35 | 36 | 38 | 39 | 42 | 43 | 45 | 49 | 49 | 53 | 54 | 55 |
| 34 | 30 | 30 | 31 | 31 | 32 | 34 | 34 | 35 | 36 | 37 | 38 | 41 | 42 | 44 | 47 | 48 | 50 | 52 | 55 | | |
| 35 | 31 | 32 | 32 | 32 | 33 | 35 | 35 | 37 | 37 | 40 | 40 | 44 | 45 | 47 | 51 | 52 | 55 | | | | |
| 36 | 32 | 33 | 33 | 34 | 35 | 36 | 37 | 39 | 39 | 42 | 43 | 46 | 49 | 50 | 54 | 55 | | | | | |
| 37 | 32 | 33 | 34 | 35 | 36 | 38 | 38 | 41 | 41 | 44 | 46 | 49 | 51 | 55 | | | | | | | |
| 38 | 33 | 34 | 35 | 36 | 37 | 39 | 40 | 43 | 44 | 47 | 49 | 51 | 55 | | | | | | | | |
| 39 | 34 | 35 | 36 | 37 | 38 | 41 | 41 | 44 | 46 | 50 | 50 | 55 | | | | | | | | | |
| 40 | 35 | 36 | 37 | 39 | 40 | 43 | 43 | 47 | 49 | 53 | 55 | | | | | | | | | | |
| 41 | 35 | 36 | 38 | 40 | 41 | 44 | 45 | 49 | 50 | 55 | | | | | | | | | | | |
| 42 | 36 | 37 | 39 | 41 | 42 | 45 | 47 | 50 | 52 | 55 | | | | | | | | | | | |
| 43 | 37 | 38 | 40 | 42 | 44 | 47 | 49 | 53 | 55 | | | | | | | | | | | | |
| 44 | 38 | 39 | 41 | 44 | 45 | 49 | 52 | 55 | | | | | | | | | | | | | |
| 45 | 38 | 40 | 42 | 45 | 47 | 50 | 54 | 55 | | | | | | | | | | | | | |
| 46 | 39 | 41 | 43 | 45 | 49 | 51 | 55 | | | | | | | | | | | | | | |
| 47 | 40 | 42 | 44 | 47 | 51 | 54 | 55 | | | | | | | | | | | | | | |
| 48 | 41 | 43 | 45 | 49 | 53 | 55 | | | | | | | | | | | | | | | |
| 49 | 42 | 45 | 47 | 50 | 54 | 55 | | | | | | | | | | | | | | | |
| 50 | 42 | 45 | 48 | 50 | 55 | | | | | | | | | | | | | | | | |

Figura 3. Sensación térmica en función de la temperatura y la humedad[1].

Para realizar estos cálculos de una manera más humana recurrimos a la utilización de la lógica difusa puesto que nuestro cuerpo no distingue valores concretos, sino que se maneja en rangos de términos como *"caliente"* o *"frio"* cuya interpretación puede ser distinta en función de la combinación de humedad y temperatura.

Una vez obtenida la sensación térmica exterior y la temperatura interior, nuestro sistema es capaz de ajustar la temperatura de la estancia para maximizar el confort y ahorrar energía mediante el apagado del sistema de aire acondicionado o calefacción en caso de no ser necesario.

Para la realización de este prototipo se utilizaron como tecnologías el lenguaje de programación Java en su versión 8 junto a la librería jFuzzy-Logic. A nivel de hardware hemos usado un sensor de temperatura analógico conectado a una Raspberry Pi 2 mediante el uso de un conversor analógico digital. También se ha usado un Arduino con un sensor de temperatura conectado a la plataforma Midgar.

### *B. Procesamiento y comunicaciones*

En la sección anterior mencionamos los distintos sentidos y su posible analogía con el mundo de Internet de las Cosas. Sin embargo, para completar la simulación, es necesario un sistema inteligente en el centro de los sentidos, simulando el cerebro, que facilite la realización de acciones en función de los datos captados y que comunique a los sensores con ellos mismos y con el exterior con el fin de poder distribuirse en diferentes ubicaciones o de permitir a terceros agentes interaccionar con el sistema. Para ello hemos desarrollado dos sistemas que permiten estas tareas. Actualmente se encuentran separados ya que tienen distintos objetivos de investigaciones. Sin embargo, al tener cierta funcionalidad similar se plantea su integración en un único sistema como trabajo futuro.

#### *1) Midgar*

Uno de los sistemas desarrollados que permiten crear aplicaciones basadas en IoT de manera que se puede usar para conectar las propuestas anteriores y realizar acciones de acuerdo a los datos obtenidos es Midgar [21]–[23].

Midgar es una plataforma de IoT que desarrollamos para realizar diversas investigaciones en el marco de Internet de las Cosas [21]. Mediante Midgar abstraemos la creación de la conexión entre los objetos y el software necesario evitando que los usuarios tengan que programar gracias al uso de lenguajes de dominio específico (DSL) gráficos. Midgar ofrece dos DSLs, uno para la creación de la interconexión entre objetos heterogéneos y ubicuos, como son los smartphones o los microcontroladores basados en Arduino, llamado *Midgar Object Intercconection Specific Language* (MOISL) [22] y otro para la creación del software para los distintos dispositivos llamado *Midgar Object Creation Specific Language* (MOCSL) [23]. De esta manera, el usuario solo necesita tener unos conocimientos mínimos de IoT y saber qué es lo que quiere que haga la aplicación generada ya que Midgar genera la aplicación final que el usuario puede usar sin la necesidad de tener conocimientos de programación y sin necesidad de saber o tener que sincronizar los diferentes objetos, pues Midgar se encarga de todo este trabajo sin necesidad de interacción por parte de los usuarios. Por tanto, el objetivo de Midgar es facilitar a usuarios no necesariamente desarrolladores la creación de diferentes componentes y la interconexión de estos en IoT.

El propósito de integrar Midgar en el conjunto del sistema propuesto es disponer de una unidad central que se encargue de procesar los datos obtenidos de los distintos sentidos y en función de eso realizar diversas acciones, es decir, pretendemos que Midgar aporte cierto nivel de autonomía al sistema y por tanto de inteligencia.

#### *2) Bilrost*

El otro sistema desarrollado que puede ser usado como eje central del sistema es Bilrost [24]. Bilrost, a diferencia de Midgar, está enfocado en investigar las posibilidades de las redes sociales como método de interconexión entre diferentes dispositivos u objetos de IoT.

[1] Datos obtenidos de http://www.tutiempo.net/meteorologia/sensacion-termica.html

Bilrost es una plataforma para la creación de la interconexión de objetos inteligentes a través de redes sociales de humanos. El fin de usar redes sociales es aprovechar sus beneficios como la garantía de estabilidad que aportan al contar con millones de usuarios. Además, el incorporar objetos a estas redes sociales hace que se pueda establecer una comunicación entre objetos y personas.

El prototipo realizado hace uso de la red social Twitter y su funcionamiento se basa en la publicación de tuits con información de los sensores o con invocaciones a actuadores de objetos inteligentes [25]. Estos tuits los define el usuario a través de un DSL al que hemos llamado *Bilrost Specific Language* (BSL). Mediante el BSL, un usuario puede definir el contenido de los tuits indicando los hashtags necesarios para realizar el filtrado en las búsquedas de tuits, los hashtags que invocan las acciones de actuadores, los tiempos de publicación de los datos de sensores y las reglas que automatizan el proceso de publicación de datos de sensores y el invocado de acciones de actuadores. Además, los usuarios también pueden publicar tuits que accionen diferentes actuadores como si fuesen un sensor más.

Por ejemplo, en Código fuente 1 se muestra un extracto de una definición de un actuador usando el BSL textual. En este ejemplo se define una alarma con dos acciones: alertar y parar.

```
ACTUATORS
       DEFINE ACTUATOR "alarm"
       ACTIONS "alert", "stop"
```

Código fuente 1. Ejemplo de definición de un actuador usando BSL.

A través de Twitter sería posible invocar a esas acciones usando los hashtags apropiados. Por ejemplo, se puede invocar a la acción de alertar mediante el uso de hashtags y cadenas de texto que identifiquen al dispositivo y los necesarios para invocar la acción del actuador como se puede ver en el siguiente tuit: *#bilrost #mderg #alarm alert.*

De la misma manera se pueden definir sensores y el modo de publicación de sus valores en las redes sociales, y reglas que automaticen acciones en función de valores de sensores, tanto del mismo dispositivo como de dispositivos externos.

Por lo tanto, el propósito de Bilrost es interconectar objetos entre sí y con personas a través de redes sociales además de incorporar un nivel de inteligencia al permitir establecer reglas que controlen la publicación e invocación de acciones.

## III. Conclusiones y Trabajo futuro

A lo largo de la sección anterior hemos presentado nuestras propuestas para un sistema inteligente capaz de interaccionar con el entorno que lo rodea a través de una analogía con los sentidos humanos. Por ello, nuestra propuesta se basa en la creación de varios subsistemas que simulen los 5 sentidos y otro subsistema central que se encargue de orquestarlos, comunicarlos y tomar decisiones de acuerdo a los datos obtenidos. De esta forma, las decisiones obtenidas habrán tenido en cuenta mucha más información que si solo se hace uso de un único y aislado sistema.

Como trabajo futuro se encuentra la creación de prototipos que abarquen todas las propuestas realizadas en secciones anteriores. Además, Bilrost y Midgar tienen cierta similitud por el hecho de que incorporan cierto nivel de inteligencia basado en el automatismo por lo que en un futuro se deberán integrar en una única plataforma convirtiéndose Bilrost en un módulo de Midgar para el uso de redes sociales a modo de canal de comunicación. Por último, el sistema central debería incorporar una inteligencia artificial capaz de aprender del entorno y tomar decisiones en función de los datos obtenidos de todos los subsistemas.

## Agradecimientos

Este trabajo ha sido realizado por el grupo de investigación "Ingeniería Dirigida por Modelos MDERG" de la Universidad de Oviedo bajo el contrato No. FC-15-GRUPIN14-084 del proyecto de investigación "Ingeniería Dirigida Por Modelos MDERG". Proyecto financiado por PR Proyecto Plan Regional.

## Referencias bibliográficas

[1] F. Telefónica, *La Sociedad de la Información en España 2014*. Grupo Planeta Spain, 2015.

[2] G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, "Smart objects as building blocks for the Internet of things," *IEEE Internet Comput.*, vol. 14, no. 1, pp. 44–51, Jan. 2010.

[3] K. a. Hribernik, Z. Ghrairi, C. Hans, and K.-D. Thoben, "Co-creating the Internet of Things - First experiences in the participatory design of Intelligent Products with Arduino," in *2011 17th International Conference on Concurrent Enterprising*, 2011, pp. 1–9.

[4] H. G. H. Gu and D. W. D. Wang, "A Content-aware Fridge based on RFID in smart home for home-healthcare," in *2009 11th International Conference on Advanced Communication Technology*, 2009, vol. 02, pp. 987–990.

[5] H. Song, "Internet of things for rural and small town america," in *6th Annual Create West Virginia Training and Education Conference*, 2013, pp. 1–6.

[6] A. Kylili and P. A. Fokaides, "European Smart Cities: The Role of Zero Energy Buildings," *Sustain. Cities Soc.*, vol. 15, pp. 86–95, Jan. 2015.

[7] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[8] R. Lea and M. Blackstock, "Smart Cities: An IoT-centric Approach," in *Proceedings of the 2014 International Workshop on Web Intelligence and Smart Sensing*, 2014, pp. 12:1–12:2.

[9] A. J. Jara, Y. Sun, H. Song, R. Bie, D. Genooud, and Y. Bocchi, "Internet of Things for Cultural Heritage of Smart Cities and Smart Regions," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, 2015, pp. 668–675.

[10] L. Hao, X. Lei, Z. Yan, and Y. ChunLi, "The application and implementation research of smart city in China," in *2012 International Conference on System Science and Engineering (ICSSE)*, 2012, pp. 288–292.

[11] J. Al Kalbani, R. B. Suwailam, A. Al Yafai, D. Al Abri, and M.

Awadalla, “Bus detection system for blind people using RFID,” in *2015 IEEE 8th GCC Conference & Exhibition*, 2015, pp. 1–6.

[12] S. Alghamdi, R. van Schyndel, and A. Alahmadi, “Indoor navigational aid using active RFID and QR-code for sighted and blind people,” in *2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2013, vol. 1, pp. 18–22.

[13] R. Neto and N. Fonseca, “Camera Reading for Blind People,” *Procedia Technol.*, vol. 16, pp. 1200–1209, 2014.

[14] M. L. Mekhalfi, F. Melgani, A. Zeggada, F. G. B. De Natale, M. A. M. Salem, and A. Khamis, “Recovering the sight to blind people in indoor environments with smart technologies,” *Expert Syst. Appl.*, vol. 46, pp. 129–138, Oct. 2016.

[15] C. Pornpanomchai and N. Suthamsmai, “Beer classification by electronic nose,” in *2008 International Conference on Wavelet Analysis and Pattern Recognition*, 2008, vol. 1, pp. 333–338.

[16] A. Riul, H. C. de Sousa, R. R. Malmegrim, D. S. dos Santos, A. C. P. L. F. Carvalho, F. J. Fonseca, O. N. Oliveira, and L. H. C. Mattoso, “Wine classification by taste sensors made from ultra-thin films and using neural networks,” *Sensors Actuators B Chem.*, vol. 98, no. 1, pp. 77–82, Mar. 2004.

[17] M. Cole, J. A. Covington, and J. W. Gardner, “Combined electronic nose and tongue for a flavour sensing system,” *Sensors Actuators B Chem.*, vol. 156, no. 2, pp. 832–839, Aug. 2011.

[18] J. Cheng, M. Sundholm, B. Zhou, M. Hirsch, and P. Lukowicz, “Smart-surface: Large scale textile pressure sensors arrays for activity recognition,” *Pervasive Mob. Comput.*, Jan. 2016.

[19] L. Atzori, A. Iera, and G. Morabito, “From ‘smart objects’ to ‘social objects’: The next evolutionary step of the internet of things,” *IEEE Commun. Mag.*, vol. 52, no. 1, pp. 97–105, Jan. 2014.

[20] E. Dijkstra, “The humble programmer,” *Commun. ACM*, vol. 15, no. October 1972, pp. 859–866, 1972.

[21] C. González García, “MIDGAR: Plataforma para la generación dinámica de aplicaciones distribuidas basadas en la integración de redes de sensores y dispositivos electrónicos IoT,” University of Oviedo, 2013.

[22] C. González García, B. C. Pelayo G-Bustelo, J. Pascual Espada, and G. Cueva-Fernandez, “Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios,” *Comput. Networks*, vol. 64, pp. 143–158, 2014.

[23] C. González García, J. Pascual Espada, E. R. Núñez-Valdez, and V. García-Díaz, “Midgar: Domain-specific language to generate smart objects for an internet of things platform,” *Proc. - 2014 8th Int. Conf. Innov. Mob. Internet Serv. Ubiquitous Comput. IMIS 2014*, pp. 352–357, 2014.

[24] D. Meana-Llorián, C. González García, J. Pascual Espada, and V. B. Semwal, “Bilrost: Connecting the Internet of Things through human social networks with a Domain-Specific Language,” *unpublished*.

[25] D. Meana-Llorián, C. González García, B. C. Pelayo G-Bustelo, and J. M. Cueva Lovelle, “Bilrost: Domain-Specific Language to define actions for the Internet of Things actuators, triggered by Twitter users posts,” *unpublished*.

# ANEXO IX. A review about Smart Objects, Sensors, and Actuators

# A review about Smart Objects, Sensors, and Actuators

Cristian González García, Daniel Meana-Llorián, B. Cristina Pelayo G-Bustelo, and Juan Manuel Cueva Lovelle, *University of Oviedo, Department of Computer Science*

***Abstract* — Smart Objects and the Internet of Things are two ideas which describe the future, walk together, and complement each other. Thus, the interconnection among objects can make them more intelligent or expand their intelligence to unsuspected limits. This could be achieved with a new network that interconnects each object around the world. However, to achieve this goal, the objects need a network that supports heterogeneous and ubiquitous objects, a network where exists more traffic among objects than among humans, but supporting for both types. For these reasons, both concepts are very close.**

**Cities, houses, cars, machines, or any another object that can sense, respond, work, or make easier the lives of their owner. This is a part of the future, an immediate future. Notwithstanding, first of all, there are to resolve a series of problems. The most important problem is the heterogeneity of objects.**

**This article is going to show a theoretical frame and the related work about Smart Object. The article will explain what are Smart Objects, doing emphasis in their difference with Not-Smart Objects. After, we will present one of the different object classification system, in our opinion, the most complete.**

***Keywords* — Smart Objects, Internet of Things, Sensors, Actuators**

## I. What is an object?

Throughout the difference literature about the universe of the Internet of Things, we could see the word '**object**' in general. Why? The reason is simple. The word '**object**' is used to refer to any device or thing, which can be intelligent or not. It means that when people talk about the interconnection among objects they talk about the interconnection among Smart Objects, among Not-Smart Objects, or between both.

Nevertheless, there are a lot of problems in the literature and the people's understanding when the word '**object**' is said, furthermore, some people use the word '**thing**' instead of '**object**', or some authors use both words interchangeably. The reason is that both definitions are very ambiguous due to the use of these word in the articles or our daily life. This is why, in this first section, we are going to introduce the exact meaning in order to delete this ambiguity.

**Object according to *WordReference* [1]**
1. Anything that can be seen or touched and is for the most part stable or lasting in form, and is usually not alive.
2. A thing, person, or matter to which thought or action is directed; the cause of such thought or action.

**Thing according to *WordReference* [1]**
1. An object, usually not a person or animal.

**Object according to Oxford [2]**
1. A material thing that can be seen and touched.

**Thing according to Oxford [2]**
1. An object that one need not, cannot, or does not wish to give a specific name.

**Object according to Cambridge [3]**
1. A thing that can be seen or felt.

**Thing according to Cambridge [3]**
1. An object; something that is not living.

As we could see, the definition depends on the site where you consult and, even so, in these cases, the definitions can be very ambiguous. The definition that is better adapted for the typical use of the word '**object**' in the universe of the Internet of Things is the first definition of '**object**', the definition obtained from WordReference. Based on this, one possible definition to the word '**object**' in the universe of the Internet of Things could be:

Any electronic device that can be connected to the Internet and collect data, like a sensor, or perform an action in an object, normally called actuator.

Manuscript received March 16, 2016.

González García, Cristian is with the University of Oviedo, Department of Computer Science, Sciences Building, C/Calvo Sotelo S/N, 33007, Asturias, Spain (corresponding author e-mail: gonzalezgarciacristian@hotmail.com).

Meana-Llorián, Daniel is with the University of Oviedo, Department of Computer Science, Sciences Building, C/Calvo Sotelo S/N, 33007, Asturias, Spain (corresponding author e-mail: danielmeanallorian@gmail.com).

P. G-B., B. Cristina is with the University of Oviedo, Department of Computer Science, Sciences Building, C/Calvo Sotelo S/N, 33007, Asturias, Spain (e-mail: crispelayo@uniovi.es).

C.L., Juan Manuel is with the University of Oviedo, Department of Computer Science, Sciences Building, C/Calvo Sotelo S/N, 33007, Asturias, Spain (e-mail: cueva@uniovi.es).

In the following sections, we will detail the differences between Smart Objects and Not-Smart Objects, explaining what are their and using examples of each one.

## II. Not-Smart Objects

In the previous section, we explained what the **objects** are and what elements compose this group. These elements are the objects without intelligence and objects with intelligence which are also known as **Smart Objects**. Due to the existence of these elements, it is essential to know how to distinguish the different type of objects and know the way in which these objects can interact with us. In this section we will address the objects of the second group, the objects without intelligence or Not-Smart Objects, and in later sections we will deepen in the **Smart Objects**. The Not-Smart Objects can be formed by **sensors** and **actuators**.

**Sensors** are electronic devices composed of sensitive cells [1] that are able to measure physical parameters like the light fluctuation using a photoresistor, the temperature using a thermistor, to detect flames, sounds, movements, or any other fluctuation in the environment [1], [4]. Thus, sensors are specific physical elements that allow us to measure a concrete physical parameter or detect something of the sensor's immediate environment.

However, **actuators** can be **mechanic actuators** which allow actions over themselves or over other devices, and **actions** which a specific object allow to perform. Thus, we can divide **actuators** in two different groups: mechanic devices and actions. Examples of **mechanic actuators** could be motors, servomotors or hydraulic bombs, and examples of **actions** could be to send a message, control LEDs, turn on lights or control the movement of a robot or any other available robot's actions.

According to the previous definitions, we could find devices that combine both types of Not-Smart Objects, they not only would have actuators and sensors but also would have both. An example of these are smartphones or any other Smart Object that are composed by sensors and actuators. Another similar example could be a microcontroller like an Arduino. The Arduino microcontroller is capable of manage almost any type of electronic device. Thus, an Arduino allows creating a system composed only of actuators, only of sensors, or both. Therefore, the **Smart Objects** are formed by Not-Smart Objects.

Figure 1 shows a concept map that explains the composition of the **objects**. This figure is useful to understand better the difference between Not-Smart Objects and **Smart Objects**. As we can see in Figure 1, Not-Smart Objects can be sensors or actuators, and actuators are divided into mechanic actuators and actions. Moreover, in order to improve the understandability, Figure 1 shows several examples of each group.

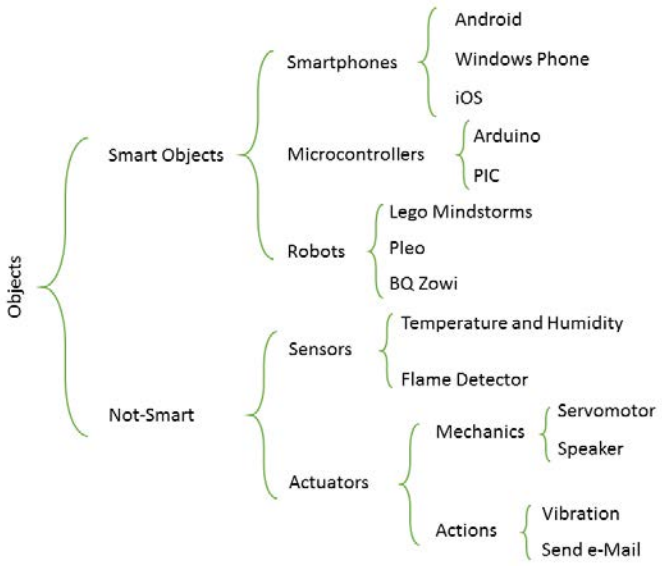



Figure 1 Composition of objects using examples

## III. Smart Objects

The definition of **Smart Object** depends on its author. Nevertheless, some authors agree with other authors and therefore, we can get a premise of their definitions. Below is our premise which was created using the definitions obtained from [5]–[9].

A **Smart Object**, also known as **Intelligent Product**, is a physical element that can be identified throughout its life and interact with the environment and other objects. Moreover, it can act in an intelligent way and independently under certain conditions. Furthermore, **Smart Objects** have an embedded operating system and they usually can have actuators, sensors, or both [5]. This allows Smart Objects to communicate with other objects, process environment data, and do events. However, there are definitions that differ from the previous which was obtained from [5]–[9].

The definition from [10] is very different from the previous. In [10], they consider as **Intelligent Products** the objects which are constantly monitoring, which react and adapt to the environment, which have an optimum performance, and which hold an active communication.

In our daily life, we are surrounded by examples of **Smart Object** and there are also examples in our everyday objects like smartphones, tablets, Smart TVs, microcontrollers like Arduino [5], [11]–[13], and even some coffee pots and some cars are also Smart Objects. Therefore, an object connected to the Internet [14] and capable of manage information [15] can be a **Smart Object**.

As we can see, Smart Objects can be very different from each other. A smartphone has little in common with a microcontroller and microcomputer. They only have in common some electronic components. Each one has their own sensors and actuators, their own intelligence, and their own operating system when they have one.

**Smart Objects** can be classified through three dimensions according to [15] and like Figure 2 shows. This classification is

useful to distinguish the different data that a Smart Object can give us about its architecture. Each dimension represents a quality of the intelligence. With the three dimensions, we can determine the intelligence that an object has and the type of **Smart Object** that it is. The three dimensions are the level of intelligence, the location of the intelligence, and the aggregation level of the intelligence.

### A. Level of intelligence

The first dimension is the level of intelligence. This describes how much intelligent an object can be. It is formed by three levels **information handling**, **notification of the problem,** and **decision making**.

#### 1) Information handling

The **information handling** is the capacity of the object to manage the information gathered from sensors, readers, or from any other techniques.

This is the most basic intelligence level and all Smart Object must have it, thus, any Smart Object must be able to manage the information that receives. Otherwise, it would not be a Smart Object and it would be just a **Not-Smart Object**.

#### 2) Notification of the problem

The **notification of the problem** is the ability of an object to notify its owner under certain conditions or when an event occurs like flames detection, an unusual decrease of the temperature, or any other event like these. In this level, the **objects** do not have free will.

#### 3) Decision making

The **decision making** is the highest level of intelligence that an object can have. An **object** has this level when it has the other two levels and it is able to take decisions by itself. It does not require any type of intervention, thus, it has free will.

### B. Location of the intelligence

The second dimension is the location of the intelligence and is formed by two categories according to [15], but we have added one extra category. Thus, this dimension has three categories: **intelligence through the Network**, **intelligence in the Object**, and **combined intelligence**. Moreover, we added a third level that combines both levels.

#### 1) Intelligence through the Network

The **intelligence through the Network** consists in that the intelligence depends totally on an external agent due to the lack of intelligence in the object. This agent can be a network where the object is connected, usually known as **portal platforms** [16], a server that runs the agents or another object that takes decisions or has the global intelligence.

#### 2) Intelligence in the Object

The **intelligence in the Object** means that the objects with this level, can process information by themselves, so, they do not need any external agent in order to be intelligent. The platforms that have **objects** with this level are usually called **embedded platforms** [16].

#### 3) Combined intelligence

The **combined intelligence** is a level that [15] does not include in their classification but they talk about it and they include it in an example graph. In this level, the object has the both intelligences. It has its own intelligence and it is capable of use the intelligence located in the Network. This platforms are usually called **surrogated platforms** [16].

### C. Aggregation level of the intelligence

The last dimension is the **aggregation level of the intelligence** which is formed by three categories. This dimension is useful to describe the **objects** that are composed of several parts. Depending on the aggregation level we could say that an object is indivisible or every part is independent. For example, we can connect a Raspberry Pi with an Arduino and connect sensors or actuators to both devices. The Not-Smart Objects like the sensors or actuators, do not have their own intelligence but the Raspberry Pi and the Arduino are Smart Objects. Therefore, if we disconnected the Arduino and the Raspberry Pi, they could run independently, whereas if we disconnect the Not-Smart Objects they could not work by themselves.

The two categories are: **intelligence in the item**, **intelligence in the container**, and **distributed intelligence**.

#### 1) Intelligence in the item

The first category is the **intelligence in the item**. This category includes the objects that are capable of handling information, notifications and/or decisions. Moreover, if these objects are composed of different components, these components must not be independents. Examples of objects that belong to this category are the smartphones. They are composed of sensors and actuators that cannot be separated because they are embedded.

#### 2) Intelligence in the container

The second category is the **intelligence in the container**. The objects of this category must be able to handle information, notifications and/or decisions and they must know their components in order to work as a proxy between their components and the Internet or the intelligence. Moreover, these **objects** are capable of working as containers or Smart Objects in spite of removing some of their components. An Arduino with at least two sensors belongs to this category. If we removed a sensor from the Arduino, the Arduino would be able to continue working as container. Another example could be intelligent shelve [15] that notify when a product is out of stock.

#### 3) Distributed Intelligence

The second category is the **distributed intelligence**. This category is the fusion between the other two. Here, items and containers have intelligent but, in this case, they can negotiate between themselves according to take the best decision to the object in base on the whole system and the rest of items. This category was added by us because we have worked with object which need this interaction. An example of this category is when you have a Smart Object which is composed by other Smart Objects, for instance, a Raspberry Pi which has connected two Arduinos. In this case, each Arduino has its own intelligence and it can take their own decisions, but sometimes, it has to ask to the Raspberry Pi about some data or the state of the another Arduino to do some action.

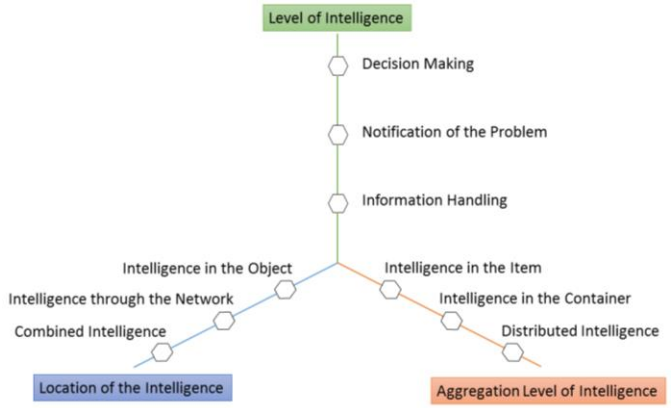



Figure 2 Classification of the intelligence based on Meyer's classification

From our point of view, the most important type of **Smart Objects** is the **combined intelligence** or the **intelligence through the network** from the dimension **location of the intelligence**. This type of objects alongside with the Internet of Things, allow adding intelligence to the network and actions according to the data that these objects collect, and services that they offer. In this way, the objects that are connected to the network could have intelligence, or even, be more intelligent.

## IV. Application Areas

**Smart Objects** are presents in our daily life for a long time. We usually consider that **Smart Objects** and the **Internet of Things** go together, although, there are many examples about the usage of **Smart Objects** without the usage of the **IoT**.

We can find Smart Objects in different systems on the **commercial field** in order to control the manufacturing like happens in [6]. Furthermore, in [7], [15], they use **Smart Objects** in order to improve the distribution and the products management in supply chains to have the products located during all their life cycle.

In the first paper, the authors describe when use different elements like readers in order to know the states of the products, monitoring them, and access to their history. Whereas, in the second paper, they mention an example which we already talked about. They mention intelligent shelves which notify when a product is out of stock.

These kind of applications are very useful to companies because they obtain advantages to improve and avoid problems related with the lack of stock during the all chain of the product life.

Following with possible uses of Smart Objects, another usage is proposed in [17]. This proposal consists in analysing the usage of rented items in order to collect the appropriate quantity of money, and also, punishing improper usage of the object. The system is good for clients as well as for companies. The clients pay exactly for the usage of the product and companies can detect improper usages and be compensated.

**Smart Objects** can also be used to improve the safety at work. In [17], the authors proposed a system to alert nearby employees about the incorrect and insecure storage of chemical material. The system proposed can be very useful because it allow managing the storage of hazardous substances and avoiding many problems or disasters.

The medical field is another field where Smart Objects can be used. A research in this field is [18]. In this research, the authors proposed a system that monitor patients with problems. Thanks to systems like this, many human lives could be saved. An example could be the connection of a cardiac pacemaker with a monitoring centre in order to detect, immediately, heart attacks or failures in the pacemaker.

## V. Conclusions

In this paper, we analysed the differences between Smart objects and Not-Smart Objects. In the literature, we cannot find the exactly differences or we can see as some authors use the both words indistinctly. In fact, this creates a problem to understand the exactly devices that they use.

Smart Objects can be used for resolving a lot of problems. We have showed a few examples of it, from supply chain to security and health.

Notwithstanding, as we say before, objects need a central system to create the interconnection between themselves. According to this goal, we can use a specific system or some Internet of Thing (IoT) platform. We can see some examples in [19] and a classification of the different IoT network types in [20]. The last one contains examples of different IoT platforms which support heterogeneous and ubiquitous objects and interconnect the objects between themselves.

We can see that the combination between Smart Objects and the Internet of Things can offer many advantages and improve the peoples' life because it can interconnect and communicate the different object to create more complex applications. Besides, we have added two new categories to the Meyer's classification in order to adapt to the new type of objects and applications in the Internet of Things.

## Acknowledgment

This work was performed by the 'Ingeniería Dirigida por Modelos MDERG' research group at the University of Oviedo under Contract No. FC-15-GRUPIN14-084 of the research project 'Ingeniería Dirigida Por Modelos MDERG'. Project financed by PR Proyecto Plan Regional.

## References

[1] WordReference.com LLC, "WordReference," 2016. [Online]. Available: http://www.wordreference.com/. [Accessed: 17-Feb-2016].

[2] Oxford University Press, "Oxford Dictionaries," 2016. [Online]. Available: http://www.oxforddictionaries.com/. [Accessed: 18-Feb-2016].

[3] Cambridge University Press, "Cambridge Dictionaries Online," 2016. [Online]. Available: http://dictionary.cambridge.org/. [Accessed: 18-Feb-2016].

[4] Alphabet Inc., "Google," 2016. [Online]. Available: http://www.google.es. [Accessed: 17-Feb-2016].

[5] K. A. Hribernik, Z. Ghrairi, C. Hans, and K. Thoben, "Co-creating the Internet of Things - First Experiences in the Participatory Design of Intelligent Products with Arduino," in *Concurrent Enterprising*

*(ICE), 2011 17th International Conference on*, 2011, pp. 1–9.

[6] D. McFarlane, S. Sarma, J. L. Chirn, C. . Wong, and K. Ashton, "Auto ID systems and intelligent manufacturing control," *Eng. Appl. Artif. Intell.*, vol. 16, no. 4, pp. 365–376, Jun. 2003.

[7] C. Y. Wong, D. McFarlane, A. Ahmad Zaharudin, and V. Agarwal, "The intelligent product driven supply chain," in *IEEE International Conference on Systems, Man and Cybernetics*, 2002, vol. vol.4, p. 6.

[8] R. Van Kranenburg, D. Caprio, E. Anzelmo, A. Bassi, S. Dodson, and M. Ratto, "The Internet of Things," in *1st Berlin Symposium on Internet and Society*, 2011, no. October 2015, p. 84.

[9] M. Kärkkäinen, J. Holmström, K. Främling, and K. Artto, "Intelligent products—a step towards a more effective project delivery chain," *Comput. Ind.*, vol. 50, pp. 141–151, 2003.

[10] O. Ventä, *Intelligent products and systems: Technology theme-final report*, no. 635. VTT Technical Research Centre of Finland, 2007.

[11] V. Georgitzikis, O. Akribopoulos, and I. Chatzigiannakis, "Controlling Physical Objects via the Internet using the Arduino Platform over 802.15.4 Networks," in *Latin America Transactions, IEEE (Revista IEEE America Latina)*, 2012, vol. 10, no. 3, pp. 1686–1689.

[12] A. Piras, D. Carboni, and A. Pintus, "A Platform to Collect, Manage and Share Heterogeneous Sensor Data," in *Networked Sensing Systems (INSS)*, 2012, pp. 1–2.

[13] Arduino, "Arduino," 2016. [Online]. Available: https://www.arduino.cc/. [Accessed: 09-Feb-2016].

[14] G. M. Lee and J. Y. Kim, "Ubiquitous networking application: Energy saving using smart objects in a home," in *2012 International Conference on ICT Convergence (ICTC)*, 2012, pp. 299–300.

[15] G. G. Meyer, K. Främling, and J. Holmström, "Intelligent Products: A survey," *Comput. Ind.*, vol. 60, no. 3, pp. 137–148, Apr. 2009.

[16] F. Ramparany and O. Boissier, "Smart devices embedding multi-agent technologies for a pro-active world," in *Proc. Uniquitous Computing Workshop*, 2002.

[17] G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, "Smart objects as building blocks for the Internet of things," *IEEE Internet Comput.*, vol. 14, no. 1, pp. 44–51, Jan. 2010.

[18] I. F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[19] C. G. García, C. P. García-Bustelo, J. P. Espada, and G. Cueva-Fernandez, "Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios," *Comput. Networks*, vol. 64, no. C, pp. 143–158, Feb. 2014.

[20] C. G. García, J. P. Espada, E. R. N. Valdez, and V. G. Diaz, "Midgar: Domain-Specific Language to Generate Smart Objects for an Internet of Things Platform," in *2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2014, pp. 352–357.

**Cristian González García** is a Technical Engineering in Computer Systems and M.S. in Web Engineering from School of Computer Engineering of Oviedo in 2011 and 2013 (University of Oviedo, Spain). Currently, he is a Ph.D. candidate in Computers Science.

His research interests are in the field of the Internet of Things, Web Engineering, Mobile Devices, and Modelling Software with DSL and MDE.

**Daniel Meana-Llorián** is a Graduated Engineering in Computer Systems from School of Computer engineering of Oviedo in 2014 (University of Oviedo, Spain). Currently, he is a student of M.S. in Web Engineering.

His research interests include Mobile technologies, Web Engineering, the Internet of Things, and exploration of emerging technologies related with the previous ones.

**B. Cristina Pelayo G-Bustelo** is a Lecturer in the Computer Science Department of the University of Oviedo. Ph.D. from the University of Oviedo in Computer Engineering.

Her research interests include Object-Oriented technology, Web Engineering, eGovernment, and Modelling Software with BPM, DSL, and MDA.

**Juan Manuel Cueva Lovelle** is a Mining Engineer from Oviedo Mining Engineers Technical School in 1983 (Oviedo University, Spain). Ph. D. from Madrid Polytechnic University, Spain (1990). From 1985 he is Professor at the Languages and Computers Systems Area in Oviedo University (Spain). ACM and IEEE voting member.

His research interests include Object-Oriented technology, Language Processors, Human-Computer Interface, Web Engineering, and Modelling Software with BPM, DSL, and MDA.

# ANEXO X. SenseQ: Creating relationships between objects to answer questions of humans by using Social Networks

# SenseQ: Creating relationships between objects to answer questions of humans by using Social Networks

Daniel Meana-Llorián
University of Oviedo
Department of Computer Science
Oviedo, Asturias, Spain
+34 985 10 33 97
danielmeanallorian@gmail.com

Cristian González García
University of Oviedo
Department of Computer Science
Oviedo, Asturias, Spain
+34 985 10 33 97
gonzalezgarciacristian@hotmail.com

Vicente García-Díaz
University of Oviedo
Department of Computer Science
Oviedo, Asturias, Spain
+34 985 10 33 26
garciavicente@uniovi.es

B. Cristina Pelayo G-Bustelo
University of Oviedo
Department of Computer Science
Oviedo, Asturias, Spain
+34 985 10 33 26
crispelayo@uniovi.es

Juan Manuel Cueva Lovelle
University of Oviedo
Department of Computer Science
Oviedo, Asturias, Spain
+34 985 18 24 52
cueva@uniovi.es

## ABSTRACT
Social Networks are a source of information very use to know what happen around us. While, the Internet of Things is a world that tries to measure everything by gathering data from sensors and interconnect different objects. Why not combine both? In this paper, we propose the creation of a system that gathers information from users' sensors, processes it and exposes this information to Social Networks through questions that users ask to the system. Users will be able to ask for data available in distributed sensors using a Social Network and our system will answer to the users also through the Social Network. In order to gather the data from sensors and combine them, we propose three different type of relationship between sensors: neighboring relationship, familiar relationship and work relationship.

## CCS Concepts
**• Information systems→World Wide Web→Web applications→Social networks**

**• Information systems→Information systems applications→Collaborative and social computing systems and tools→Social networking sites**

## Keywords
Social Networks; Sensors; The Internet of Things; Twitter

## 1. INTRODUCTION
Social Networks (SN) are very presents in our lives. Many of us share all aspects of their life on Social Networks like posting messages about what they are doing at that moment, sharing photos with friends, giving their opinion about other posts, and so on. Moreover, Social Networks are also used by business to promote their products or services and even, SNs are a new channel of communication between companies and clients.

Furthermore, Social Networks are also a source of data where users can search any type of information like the closing time of a shop, the restaurant's phone, the weather of a specific zone through the profile of an expert and so on. In this paper, we will present our proposal of a source of information accessible via Social Networks. The information that we want to provide is the data gathered by many sensors. Hence, we propose the combination of Social Networks and the world of the Internet of Things (IoT) to inform the user about any data available in our network of sensors.

Currently, there are many references to the Internet of Things around us. Some references could be the smartphones, wearables, tablets [1] or any other device with Internet capabilities. These smart devices are also called Smart Objects. The world of IoT allows the creation of many smart environments like Smart Homes [2]–[4], Smart Towns [5], Smart Cities [6]–[9] or Smart Earth [6], [10]. These new possibilities can exist due to the combination of different actuators, sensors or Smart Objects. Our proposal is based on a system that collects information from a sensor network to answer questions that users makes through Social Networks.

In this paper, we will present a proposal of integration of Social Networks and the Internet of Things in order to research how technologies of the Internet of Things can communicate with humans by using Social Networks.

The remainder of the paper is organized as follows. In Section 2, we introduce involved topics in this research. Section 3 shows our proposal, the system to answer questions of users by using sensor networks and Section 4 contains the conclusions of this paper and the possible future work that can be done from this proposal.

## 2. STATE OF THE ART
In this section, we will present a theoretical frame about the concepts involved in our proposal. These concepts include the Internet of Things, Smart Objects and Social Networks, because our proposal could be interpreted as a way of make queries to Smart Objects from the Internet of Things through Social Networks.

## 2.1 The Internet of Things

The term "the Internet of Things" (IoT) is defined as the interconnection of heterogeneous and ubiquitous objects between themselves [3], [11]–[14]. The concept includes not only the connection between objects also known as Machine-to-Machine (M2M) [15]–[17] but also the communication between humans (H2H) [17] and humans to machines (H2M) [18].

There are many researches that address the IoT in large range of fields. Researches related with our proposal could be researches address on the IoT and Social Networks.

The Social Internet of Things (SIoT) [13] combines the IoT and Social Networks obtaining a new generation of objects, the social objects. These objects are capable of interacting with other objects by themselves. They are able to discover services and useful information from other objects because all of them share their services and information in the network [19].

Moreover, the combination of IoT technologies and Social Networks habits help people to familiarize themselves with the IoT according to scientists of Ericsson [19].

## 2.2 Smart Objects

In the definition of the IoT we mentioned the word objects. In the IoT the objects usually have any type of intelligence and therefore, they are called Smart Objects. The Smart Objects are objects with an embedded system that allows them to process information, communicate with other devices and perform actions based on other actions or events [3].

According to [3], [20], Smart Objects can be classified in three levels: Level of Intelligence, Location of Intelligence y Aggregation level of Intelligence.

Examples of Smart Objects can be smartphones, micro-controllers like Arduino [3], [21], [22] or any other object connected to the Internet and capable of handling information. The combination of the IoT and Smart Objects achieves that the network not only transports data but also is provided of intelligence and actions.

## 2.3 Social Networks

A way of integrate applications in our lives is adding social aspects to the applications. Good resources for that propose are the Social Networks (SN). Many of Social Networks provide several Application Programming Interfaces (API) that allow reading or writing in timelines, receiving or sending private messages or updates, and so on. The convergence of the real world with SNs is an important piece of the Web 2.0 because SNs like Facebook or Twitter allow their users to communicate, exchange and share contents. Thus, creating applications that interconnect things and humans is also possible [23].

Currently, there are many available Social Networks to use for research purposes but we have chosen Twitter due to its way of work based on short messages composed by keywords, which are called hashtags. Furthermore, Twitter is suitable for this research although it has some limitations.

# 3. PROPOSAL

We propose a system capable of answering questions that people would make through Social Networks using information from sensors connected to our system. Moreover, the sensors will have social capacities because they will be able to have friends in order to answer people's questions jointly. When a user will ask for specific information, the system will try to get the information from the registered sensors or from the relationship between sensors. For example, if a user asks for climate data the system will response with data from several sensors like temperature sensors and humidity sensors because these sensors have a friendship relationship based on the clime.

In order to explain our proposal, this section is composed by three different subsections: the registration of sensors, the socialization of sensors and the social interface.

## 3.1 The registration of sensors

Our system needs the registration of sensors that can gather the data required to answer the questions made by people. For that reason, we propose that users can register sensors in a web application with the parameters that allow us to identify the sensors properly. These parameters would be its location, its finality, its owner and the privacy state and some of them are used to refine the relationship between sensors as we will explain in subsection 3.2.

### *3.1.1 Location*

The location will be a required parameter to register a sensor. The location will be composed by several locations because a sensor will be located in one place it will be reachable through different location levels. For instance, a sensor located in a university classroom would be reachable through different levels like "*University*", "*City*" or "*Country*".

This parameter will allow the system to know the neighbors of a sensor in order to allow the socialization between neighboring sensors as we will explain in subsection 3.2.

### *3.1.2 Finality*

Another required parameter will be the finality. This parameter will indicate the role of the sensor. The finality will be composed by several finalities because a sensor could be useful for different proposes. For example, a sensor of temperature would be handy for knowing the temperature but it would be also handy for knowing the climate conditions alongside other sensors.

This parameter is also used to allow the socialization between sensors with the same finality as we will explain in subsection 3.2.

### *3.1.3 Owner*

The sensor's owner will be another required parameter because the system will need the owner in order to store and retrieve the sensors. This parameter will not be introduced by the user but the system will assign the owner considering the user who registers the sensor in the system automatically.

This parameter is also used to allow the socialization between sensors whose owner is the same as we will explain in subsection 3.2.

### *3.1.4 Privacy state*

The privacy is very important when we handle data obtained from sensors because we can use sensors that gather sensitive information like health problems. To ensure the privacy, the users will be able to specify different levels of privacy. Four privacy levels will be available in our system. Users will have to choose if their sensors are reachable by any user request, only by requests that include a specific location, only by requests made by its owner or a combination of the two previous levels.

Like the other parameters, this one is also used to allow the socialization between sensors but it is used not only to allow it, but also to avoid it as we will explain in subsection 3.2.

## 3.2 The socialization of sensors

The aim of our proposal is the creation of a network of sensors where they will have different types of relationships according to

the parameters we exposed in subsection 3.1 in order to give certain information to user when it asks something through a social interface. Available relationships will be based on the four previous parameters and they will be: neighboring relationship, work relationship and familiar relationship.

### *3.2.1 Neighboring relationship*

Through a socialization based on neighboring relationship, sensors will be able to know the nearest sensors. This type of relationship is used to set the different levels of privacy. One level of privacy will restrict the access to the sensor to only located requests.

For instance, if a user asks for the air quality of a building which is stored as location parameter of some sensors, the system will collect the information from sensors located in the building. Then, the system will combine all the gathered information and it will send it to the user.

### *3.2.2 Work relationship*

The primary relationship of the proposed system is the work relationship. The aim of our proposal is answer the question that users ask through a Social Network, therefore, the system must have a way of recognize what the user want. This is the finality parameter that we explained in subsection 3.1.2.

The work relationship will not affect the privacy but it will affect the functionality of the system. When a user asks for a certain information, the system will search sensors whose finalities are suitable to answer the user's question taking into consideration all other relationships. Then, the system will combine all the gathered information and it will send it to the user.

### *3.2.3 Familiar relationship*

The familiar relationship will allow the system to know sensors which share their owner. This type of relationship is also used to set the different levels of privacy. One level of privacy will restrict the access to the sensor to only requests from its owner.

An example is if a user asks for the humidity level without indicate a location, the system will collect the information only from sensors of the user which made the request instead of collecting the information from all public sensors that are suitable to answer the user's question. Then, the system will combine all the gathered information and it will send it to the user.

## 3.3 The social interface

One of our aims is the use of Social Networks to connect humans with sensors. We propose the use of Twitter as input interface through which the users will ask for information and the proposed system will answer.

Twitter is based on short messages composed by text and keywords that can represent users or topics. Our system will use these keywords to identify some parameters required to ask for information. The owner parameter will not be present in the text, instead it will be in the properties of the Twitter message because the owner will be the user of Twitter who makes the question. The rest of parameters will be find analyzing the text and hashtags. In order to identify the requested information, which is the finality parameter, the system will analyze the text and it will analyze the hashtags to find the location parameter.

Because of the use of Twitter, the system will have to store the username of sensors' owners. For that, users will have to sign up in the web application using their Twitter profile.

Examples of use could be the next:

- **Asking for the temperature in New York City**: The system will consult all sensors registered in New York City and it will combine the information to send the appropriate value.
  - *@user:* *@senseq What is the temperature of #NewYork*
  - *@senseq:* *@user The temperature of #NewYork is 23ºC*
- **Asking where it is raining using your sensors:** The system will consult all your sensors and it will respond with the locations of sensors that return that it is raining.
  - *@user:* *@senseq Where is it raining?*
  - *@senseq:* *@user It is raining in #Manhattan and #Jersey.*

As we can see in the previous examples, the proposed system will require techniques to analyze and understand natural language in order to obtain what the user is asking for.

# 4. CONCLUSIONS

In this paper, we propose a system that combines technologies of the Internet of Things and Social Networks in order to make easy the access to sensors data by users. Moreover, we propose three different types of relationships that filter the information that a user can access based on the location, the finality and the owner of each sensor. Finally, we explain how it the communication with the system through a Social Network, Twitter, by using public messages that contain hashtags and mentions.

From this work we will able to propose the next new future works:

- **Adding new relationships based on owner friends:** In this paper we talk about relationships between sensors, nevertheless, taking into consideration the relationships between users that use the system is another type of relationship that could be useful and we would to study to integrate in our system.
- **Establishing relationships between the system and users:** Currently, we only use the Social Networks as a method of communication, but we would take the advantages of the Social Networks to establish friends relationships in order to, for instance, give automatically recommendations to users based on previous queries.
- **External platforms:** Our proposed is a standalone system that it is not connected with other system and neither it allows the connection of an external system to retrieve information. As a future work could be the creation of an API so that other platforms or users can make queries to our system. Moreover, we could add to the system, platforms as source of information like Midgar [14], [24]. In this way, the system could use other platforms as a sensor and retrieve information from them.

# 5. ACKNOWLEDGMENTS

This work was performed by the "Ingeniería Dirigida por Modelos MDE-RG" research group at the University of Oviedo under Contract No. FC-15-GRUPIN14-084 of the research project "Ingeniería Dirigida Por Modelos MDE-RG". Project financed by PR Proyecto Plan Regional.

# 6. REFERENCES

[1] F. Telefónica, *La Sociedad de la Información en España 2014*. Grupo Planeta Spain, 2015.

[2] C. Han, J. M. Jornet, E. Fadel, and I. F. Akyildiz, “A cross-layer communication module for the Internet of Things,” *Comput. Networks*, vol. 57, no. 3, pp. 622–633, 2013.

[3] K. a. Hribernik, Z. Ghrairi, C. Hans, and K.-D. Thoben, “Co-creating the Internet of Things - First experiences in the participatory design of Intelligent Products with Arduino,” in *2011 17th International Conference on Concurrent Enterprising*, 2011, pp. 1–9.

[4] D. Ding, R. A. Cooper, P. F. Pasquina, and L. Fici-Pasquina, “Sensor technology for smart homes,” *Maturitas*, vol. 69, no. 2, pp. 131–136, Jun. 2011.

[5] H. Song, “Internet of things for rural and small town america,” in *6th Annual Create West Virginia Training and Education Conference*, 2013, pp. 1–6.

[6] L. Hao, X. Lei, Z. Yan, and Y. ChunLi, “The application and implementation research of smart city in China,” in *2012 International Conference on System Science and Engineering (ICSSE)*, 2012, pp. 288–292.

[7] R. Lea and M. Blackstock, “Smart Cities: An IoT-centric Approach,” in *Proceedings of the 2014 International Workshop on Web Intelligence and Smart Sensing*, 2014, pp. 12:1–12:2.

[8] R. Y. Clarke, “Smart cities and the internet of everything: The foundation for delivering next-generation citizen services,” *Alexandria, VA, Tech. Rep*, 2013.

[9] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of Things for Smart Cities,” *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[10] B. M. Howe, Y. Chao, P. Arabshahi, S. Roy, T. McGinnis, and A. Gray, “A Smart Sensor Web for Ocean Observation: Fixed and Mobile Platforms, Integrated Acoustics, Satellites and Predictive Modeling,” *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 3, no. 4, pp. 507–521, Dec. 2010.

[11] G. M. Lee and J. Y. Kim, “Ubiquitous networking application: Energy saving using smart objects in a home,” in *2012 International Conference on ICT Convergence (ICTC)*, 2012, pp. 299–300.

[12] G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, “Smart objects as building blocks for the Internet of things,” *IEEE Internet Comput.*, vol. 14, no. 1, pp. 44–51, Jan. 2010.

[13] L. Atzori, A. Iera, G. Morabito, and M. Nitti, “The Social Internet of Things (SIoT) – When social networks meet the Internet of Things: Concept, architecture and network characterization,” *Comput. Networks*, vol. 56, no. 16, pp. 3594–3608, Nov. 2012.

[14] C. González García, B. C. Pelayo G-Bustelo, J. Pascual Espada, and G. Cueva-Fernandez, “Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios,” *Comput. Networks*, vol. 64, pp. 143–158, 2014.

[15] E. Borgia, “The Internet of Things vision : Key features , applications and open issues,” *Comput. Commun.*, vol. 54, pp. 1–31, 2014.

[16] Lu Tan and Neng Wang, “Future internet: The Internet of Things,” in *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, 2010, vol. 5, pp. V5–376–V5–380.

[17] M. Hasan, E. Hossain, and D. Niyato, “Random access for machine-to-machine communication in LTE-advanced networks: issues and approaches,” *IEEE Commun. Mag.*, vol. 51, no. 6, pp. 86–93, Jun. 2013.

[18] International Telecommunication Union, “Overview of the Internet of things,” *Ser. Y Glob. Inf. infrastructure, internet Protoc. Asp. next-generation networks - Fram. Funct. Archit. Model.*, p. 22, 2012.

[19] L. Atzori, A. Iera, and G. Morabito, “From ‘smart objects’ to ‘social objects’: The next evolutionary step of the internet of things,” *IEEE Commun. Mag.*, vol. 52, no. 1, pp. 97–105, Jan. 2014.

[20] G. G. Meyer, K. Främling, and J. Holmström, “Intelligent Products: A survey,” *Comput. Ind.*, vol. 60, no. 3, pp. 137–148, Apr. 2009.

[21] V. Georgitzikis, O. Akribopoulos, and I. Chatzigiannakis, “Controlling physical objects via the internet using the arduino platform over 802.15.4 networks,” *IEEE Lat. Am. Trans.*, vol. 10, no. 3, pp. 1686–1689, 2012.

[22] A. Piras, D. Carboni, A. Pintus, and D. M. T. Features, “A Platform to Collect , Manage and Share Heterogeneous Sensor Data,” in *Networked Sensing Systems (INSS)*, 2012, pp. 1–2.

[23] M. Blackstock, R. Lea, and A. Friday, “Uniting online social networks with places and things,” in *Proceedings of the Second International Workshop on Web of Things*, 2011, pp. 5:1–5:6.

[24] C. González García, J. Pascual Espada, E. R. Núñez-Valdez, and V. García-Díaz, “Midgar: Domain-specific language to generate smart objects for an internet of things platform,” *Proc. - 2014 8th Int. Conf. Innov. Mob. Internet Serv. Ubiquitous Comput. IMIS 2014*, pp. 352–357, 2014.

# ANEXO XI. Bilrost: Using Domain-Specific Languages to connect Smart Objects through Social Networks

# Bilrost: Using Domain-Specific Languages to connect Smart Objects through Social Networks

Daniel Meana-Llorián, Cristian González García, B. Cristina Pelayo G-Bustelo, Juan Manuel Cueva Lovelle
Department of Computer Science
University of Oviedo
Oviedo, Spain
danielmeanallorian@gmail.com, gonzalezgarciacristian@hotmail.com, crispelayo@uniovi.es, cueva@uniovi.es

***Abstract*—Currently, many Smart Objects with Internet connection like smartphones, tablets, wearables, microcontrollers like Arduino, and so on, are around us. However, interconnecting these objects in order to achieve a common goal in an easy way is an issue to address. In this paper, we propose a novel approach to interconnect objects with each other and people. Our proposal is the use of Social Networks as communication channel in order to achieve a better integration of the Internet of Things in our lives. We propose a Domain Specific Language to define devices, their sensors, and actuators, the parameters required to establish communications through Social Networks and rules to automate the communication process. In order to prove this proposal, we developed a system composed by a textual and a graphic editor. We made an evaluation composed by a comparison between these two editors and a survey to gather users' opinions. We conclude that our proposal is useful to connect Smart Objects with each other and with people in an easy way.**

***Keywords—Internet of Things; Smart Objects; Model-Driven Engineering; Domain-Specific Language; Social Networks, Human-Machine communications***

## I. INTRODUCTION

Nowadays, everybody knows Smart Objects [1] although they do not realise what a Smart Object is. The most technology we use are composed by sensors and/or actuators and has some intelligence to perform actions that depend on several conditions or events. Examples of these Smart Objects or devices are smartphones, tablets, wearables devices, Smart TVs, Smart cars, and many other devices whit any type of intelligence. These devices are rising continuously [2] and therefore, the Internet of Things (IoT) is also gaining popularity.

There are many researches in a wide range of topics related with the Internet of Things, such as researches on the field of health care [3], rehabilitation systems [4], systems that support people with disabilities [5] or smart environments like smart homes [6]–[9], or [10] where they use the IoT to save energy at home, smart buildings [11], smart cities [12]–[16], smart towns [17], or even smart earth [18].

However, developing solutions based on the IoT is not very easy because the need for hardware requirements like many sensors, actuators, or even a platform to deploy the solutions that interconnect these objects. The most solutions proposed are based on using a Representational State Transfer (REST) architecture in combination with a central server [19]–[21]. This paper proposes a novel way of communicate Smart Object without the necessity of having access to the many Smart Objects and without the necessity of having an infrastructure to deploy applications. We propose the use of Social Networks as way to communicate Smart Objects.

Social Networks is another technology very popular. They are very present in our lives daily lives. Most people are used to sharing many aspects of their lives like what they are doing, what they are eating, who they are with, what they think about something, and so on. Moreover, in the business environment, Social Networks are also a channel to promote products and services or to win customers.

The combination of Social Networks and the Internet of Things is not casual. According to researches of Ericsson [22], people understand better IoT technologies if it exists an analogy between our habits in Social Networks like Facebook or Twitter and how IoT technologies work. Because of that, we proposed this combination by using Social Networks to communicate Smart Objects.

Moreover, according to [23], the integration of IoT technologies in humans lives leaves much room for improvement. To achieve this integration, is very important make Smart Objects accessible [24]. For that, we propose Bilrost, a way of integrate Smart Objects in human Social Networks. As we will explain in this paper, Bilrost allows users to generate applications which have the connection with other objects implemented through Social Networks by using a Domain-Specific Language (DSL). In this way, users do no need knowledge about interconnecting objects neither an infrastructure to connect them.

Furthermore, Bilrost use Social Networks to establish a channel of communication between Smart Objects and people. Thus, we achieve a vital issue in the Internet of Things, the interoperability between heterogeneous and ubiquitous objects and different systems [24]. One goal of the IoT is to expand existing human-human communications to human-things and things-things communication by connecting the physical world and the virtual world [24]. This goal can be achieved by using Social Networks which open an entry point for users or applications to interact with objects available on the web [25].

Because of all that, we proposed Bilrost, a system that allows the generation of applications that interconnect Smart Objects through Social Networks without a complex programming knowledge due to the use of a DSL called Bilrost-

Specific Language (BSL). With Bilrost, we want answer the next questions.

- Is it possible integrate Smart Objects in Social Networks developed for people?
- Could people interact with Smart Objects through the Social Networks that they usually use?
- Could Domain-Specific Languages support the interconnection of Smart Objects through Social Networks?

Through these questions, we want to check the next hypothesis:

*"Facilitate the creation of applications that enable the integration of Smart Objects from the Internet of Things in Social Networks, and allow the interaction among objects and between objects and people, is possible."*

The remainder of this paper is organised as follows. In Section II, we introduce involved topics in this research like the IoT, Smart Objects, MDE, DSL, and Social Networks. Section III shows the Bilrost platform, what it is, how its architecture is, how its DSL is, and how interact with Twitter. In Section IV we cover the evaluation and discussion of the data obtained from analysing the process of creating projects with Bilrost. Section V contains an analysis of previous researches that are related with our work. In Section VI we show the conclusions obtained from this research and finally, in Section 0, we describe the possible future work that can be done from here.

## II. Background

We propose a novel way to intercommunicate Smart Objects from the Internet of Things by creating applications that connect Smart Objects to Social Networks. Most of researches about the connection of Smart Objects are based on Service Oriented Architecture (SOA) like REST which is used in [19].

We propose the used of Social Networks instead of use a server with SOA although we use SOA to interact with Social Networks' Application Programming Interfaces (APIs). Moreover, we created a DSL to generate these applications with the connection already implemented according to users' requirements. In this section, we will introduce the concepts involved in this proposal: The Internet of Things, Smart Objects, Model-Driven Engineering, Domain-Specific Languages, and Social Networks.

### A. Internet of Things

The Internet of Things (IoT) can be defined as the interaction between things in order to cooperate with others and create applications or services that follow common aims [26]. This concept was announced by the National Intelligence Council of United States as one of the six technologies with more impact in the United States from here to 2025 [27].

The concept of the IoT appeared to resolve a problem of supply chains related with the identification of objects, persons, and animals. In 1999, Kevin Ashton introduced this term in a presentation to Procter & Gamble (P&G) where he talked about the use of Radio Frequency Identification (RFID) tags in supply chains [28]. The use of RFID tags allows identifying objects and improving the managing of supply chains.

The aim of the IoT is interconnect heterogeneous and ubiquitous objects and different systems between each other. For that reason, enabling things with Internet access is mandatory [10]. We can say that the IoT was introduced in order to extend the Internet to things [29]. However, not only the interconnection between objects, also called Machine-To-Machine (M2M) [30]–[32], is important in the IoT, but also the connections between humans and machines (H2M) [33] and among humans (H2H) [31] are also very important because the three types of communications together allow sharing information between the physical world and the virtual world [24]. Our proposal tries to offer a way of establishing communications among objects and between objects and humans, by using Social Networks directly instead a common approach of using web services.

IoT applications usually use a basic architecture composed by three layers: Perception layer, Network layer, and Application layer [34]. Each layer has its own function. The first layer, **Perception layer**, is the layer that identifies objects and gather information. This layer is also called Device layer. In this layer is where sensors and RFID tags are located. The next layer, **Network layer**, is the layer that communicate the other both layers. Its function is to ensure the transmission of data between both layers. In this layer, we can integrate transmission mediums and communication protocols like Bluetooth, ZigBee or any other. The last layer, **Application layer**, is the layer where the final applications are located. It functions is to collect the information from lower layers and show it to the user through applications [24]. Our proposal is a solution based on Social Network, which is situated in the Network layer. We proposed the use of Social Networks as transmission medium. Also, we provide functions of the other layers because our system is ready for gathering information from sensors and controlling actuators, and moreover, our system is capable of generating applications that the communication between devices is already implemented.

### B. Smart Objects

The definition of the IoT includes the word objects. Objects can be any device or thing without depends on their intelligence. Thus, there are intelligent objects, better known as Smart Objects or Intelligent Products, and Not-Intelligent objects or Not-Smart Objects. In the IoT world we can find both types of objects.

The Not-Smart Objects or objects without intelligence are devices that need another device to work. This group is formed by **sensors** and **actuators**. Sensors are able to measure physical parameters like the light fluctuation or the temperature, but they do not know how to process it without other device that is programmed to process data. By the other side, actuators are able to perform actions like send a message or turn on or turn off a LED, but they need another device that order them when actuate taking into consideration certain conditions. The devices capable of doing what sensors and actuators are not able to do are **Smart Objects**. For instance, a Raspberry Pi can be considered a Smart Objects because it has the ability of process data and perform actions under certain conditions, and moreover, it can be connected several sensors and actuators to measure physical parameters or to perform physical actions like

a light sensor or different LEDs. Thus, we can say that Smart Objects are composed of Not-Smart Objects. The principal difference between a Not-Smart Object and a Smart Object is that the second is capable of acting in an intelligent way and independently under certain conditions, and interact with the environment and other objects because of having an embedded operating system and actuators, sensors, or both [7].

Smart Objects can also defined as **Intelligent Products** with the capacity of monitoring, reacting, and being adapting to the environment, by holding an active communication and with a good performance [35]. Some examples of Smart Objects that round us daily are smartphones, tablets, Smart TVs, or even some things that we do not expect that they are smart like coffee pots or some cars. Other examples of Smart Objects are microcontrollers like Arduino [7], [36], [37] due to their adaptability to make almost everything without make a big cost. Thus, any object connected to the Internet and capable of handling information can be considered a Smart Object.

In [1], we can find a classification of **Smart Objects** through three dimensions which represent qualities of object's intelligence. These dimensions are the level of intelligence, the location of the intelligence, and the aggregation level of the intelligence.

- **Level of intelligence**: This dimension describes how much intelligence an object can have through three levels: The capacity of **handling information**, the ability to **notify events**, and the capacity of **taking decision** by its own.
- **Location of the intelligence**: This dimension describes where the object's intelligence is located. The two possible locations according to the classification showed in [1] are the **intelligence through the network** and the **intelligence in the object**. In [1], authors only mention these two locations but show a third possible location in a figure that is the combination of other both. Platforms which support the first location are called **portal platforms** [38], platforms that support the second location are called **embedded platforms** [38], and platforms which support both are called **surrogated platforms** [38].
- **Aggregation level of the intelligence**: This dimension describes the distribution of the intelligence in objects that can be composed by several parts. Two options are possible: can be **intelligence in the item** and **intelligence in the container**. The first option occurs when objects cannot be divided in parts without lost the intelligence like a sensor with a little intelligence. The second option occurs when the intelligence is in a container that has smaller parts that can be removed, for instance, an Arduino with several sensors.

Our proposal aims to make easier the connection between Smart Objects by using Social Networks. To achieve this, we provide the needed tools to create applications that can run in Smart Objects and are already connected to Social Networks. Moreover, our proposal is capable of making Smart Objects that are able to take decisions by their own, have their intelligence inside themselves, and have the intelligence in the container.

### *C. Model-Driven Engineering*

Model-Driven Engineering (MDE) appeared to solve problems that have existed in software development since 1960s. These problems was descripted by Dijkstra [39] and by the North Atlantic Treaty Organization (NATO) [40], and nevertheless they are still present [41]. These problems are the low quality of the developed software, the breach of budgets and planning, and the increment on the maintenance cost. However, there are some small solutions that help minimise these problems by reducing the effects of the essential complexity that depends on the features from functional requirements, and the accidental complexity [42]that is related with the action of programing and writing code [42].

A solution that emerged to solve these problems was the industrialisation and automatisation of the software development process. A quite popular attempt of industrialisation of the software was MDE that is an approach to design and develop software based on the use of models [43]–[45]. The aim of MDE is to increase the productivity and to reduce the development time by allowing the development process to be closer to the domain problem through the creation of one or more models, and increasing the abstraction level to be more understandable for people, for instance, through the creation of new Domain-Specific Languages. Hence, MDE tries to resolve the complexity of the design and the development of modern software [46], allowing the generation of code or diagrams [47] by an automatic or semi-automatic process [48].

In our proposal, we applied MDE to create a new DSL that raises the level of abstraction, is easier to understand than a General Purpose Language (GPL) and allows the generation of applications that interconnect heterogeneous and ubiquitous objects through social networks in different languages.

### *D. Domain-Specific Languages*

A Domain-Specific Language (DSL) is a language, commonly declarative, developed to solve problems of a specific domain [49], with a greater power of expression [50]–[52].

Sometimes, generic solutions are not the best optimum solutions which is one of the problems of GPLs. Due to that, DSLs was appeared. The use of DSL allows developing optimised solutions and a better adaptability to a specific problem [53].

DSLs abstract the knowledge level and allow decreasing the difficulty of APIs. In that way, final users can work with models without having specific knowledge about the software or APIs [54]. DSLs are a very important piece in MDE [51], hence, they must be based on a formal metamodel in order to work with MDE properly.

The use of DSLs give us some advantages [55] like the improvement in the productivity due to the reduction and detection of errors, a better compression of the language due to using terms related with a specific domain, the portability, the possibility of reuse models for commons proposals and the maintenance [52]. Nevertheless, they have some disadvantages [55] like a worse performance, the time and cost spent in researching about how to create the DSL, the difficulties to make the DSL works properly, the additional documentation

needed to explain how the DSL works and the preparation of the developers to use the DSL.

We have applied MDE to create a DSL called Bilrost-Specific Language (BSL). The use of the BSL allows generating applications that connect heterogeneous and ubiquitous Smart Objects between each other and with humans through Social Networks by defining the properties of a Smart Object or device, its actuators or sensors, the parameters required by the social networks, the rules which automate the communication between devices. For that, we have created two tools that allow writing textual models using the syntax of BSL, and writing graphic models through a graphic editor. With the BSL, we try to make easier the creation of that type of applications.

### *E. Social Networks*

Social Networks or Online Social Networks (OSN) are considered an important piece to achieve the convergence of the real world and digital world in the Web 2.0 [25]. OSNs are useful to keep in contact with family, friends, and co-workers due to the existence of relationships between users and the capacity to share content with them. Moreover, OSNs often offer their data to developers in order to create new social applications [56]. Among services that OSNs usually provide to third-applications through APIs are: services to do the authentication, methods to defined authorisation access rights, streaming of data available on the Social Network, and extending the Social Network's features through third applications.

Social Networks, especially Twitter, are usually used to research purposes that want to gather data about people and events. For instance, in [57], authors use Twitter to extract information about traffic events using natural language processing.

Twitter is an OSN and microblogging service based on short text-based messages of up 140 characters, very used for research purposes because of some features that it has. The users' relationships in Social Networks are often based on reciprocation, a user has to accept you in order to be connected, such as Facebook. However, Twitter bases their users' relationships on following or being followed without the needed of reciprocation [58]. Users are free to follow another user without the other user follows back. Another important feature of Twitter as microblogging service is its real-time nature. Furthermore, Twitter has a specialised mark-up language that allow adding semantic information to messages and makes easier the development of prototypes. Due to all these features, we decided to use Twitter in our proposal of interconnecting Smart Objects through Social Networks.

In the frame of the Internet of Things, Twitter is also very used in researches, for instance, considering humans as another type of sensor as it is made in [59] for use in Smart Cities or for detecting Earthquakes [60], or also to support smart decisions about the destination of tourism according to the opinions of Twitter's users [61].

There are several proposals about merging Social Networks and the Internet of Things based on a deduction made by scientists of Ericsson [22]. They observed that an analogy between IoT technologies and Social Networks habits, makes people capable of familiarising better with IoT technologies. Many of these proposals are based on giving social abilities to Smart Objects in order to establish relationships between objects. This concept was called Social Internet of Things (SIoT) [22], [62].

## III. BILROST PLATFORM

Our proposal, that we have called Bilrost, is a platform to connect Smart Objects from the world of the Internet of Things through human Social Networks by using a DSL that can be used in two ways, using a textual editor and using a graphic editor. The platform enables the creation of projects of applications that interconnect ubiquitous and heterogeneous objects using Social Networks as communication channel without the necessity of having own servers, as well as more advantages.

Social Networks are very used by millions of real users every day, hence, use them give us an environment enough tested to guarantee a full availability and security. We propose the use of human Social Networks not only to interconnect Smart Objects but also to connect objects with users of networks allowing them to control actuators or to read sensor's data. In this way, we are making the Internet of Things bigger due to the addition of people and making the Social Networks bigger due to the addition of Smart Objects.

In order to achieve our proposal and to make easier the creation of applications which connect objects though Social Networks, we propose the use of a DSL created for this

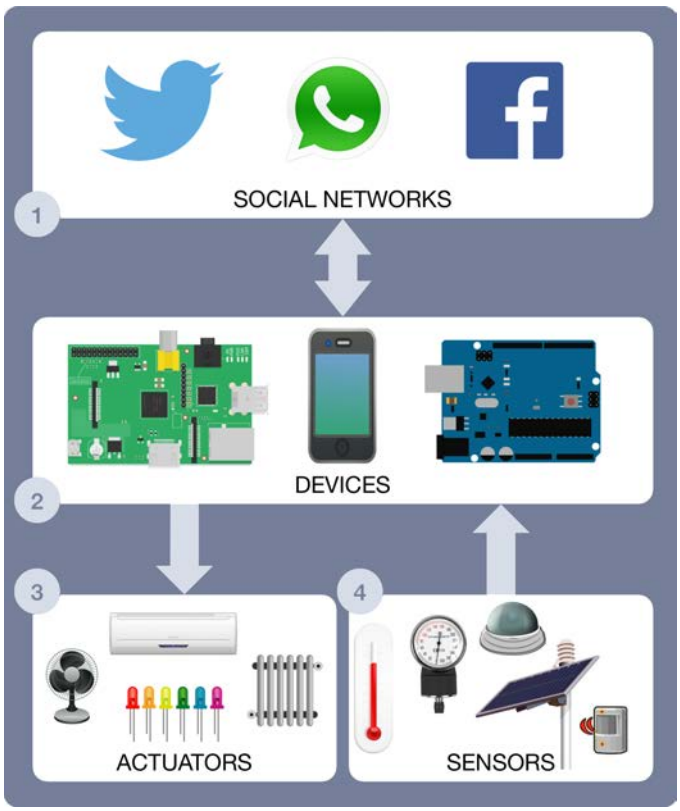



Fig 1. Components of the interconnection of devices through social networks.

proposal. Besides, in evaluation section we will evaluate both DSLs by means of users' opinions.

Our proposal addresses the connection of objects to Social Networks, hence, we have made a prototype that generates projects of applications where the connection to a Social Networks, Twitter, is already implemented. However, users will have to implement the specific logic that allow the application to gather information from sensors or to control the actuators.

In this section, we will describe Bilrost platform making mention of how Bilrost works, its architecture, and the two DSLs used to create the applications. Firstly, we will give a general approach of how Bilrost works, after that we will address its architecture. Finally, we will present the DSL that enables the creation of applications.

Moreover, we will use a simple example to make easier the explanation about how Bilrost works although the possibilities are greater. We will connect a Raspberry Pi and an Android smartphone by Twitter. Each device has a sensor and an actuator. The Raspberry Pi has a light sensor and a red led as actuator whereas the Android smartphone has the accelerometer as sensor and the flash as actuator.

In Fig 1 we can see a general scheme about the components that take place in the interconnection of devices through social networks. The example that we will use in this paper has components of all these types. The Raspberry Pi and the Android smartphone are devices so they are represented with the number 2. The Raspberry Pi's light sensor and the Android smartphone's accelerometer are sensors, so they are represented with the number 4. Lastly, the Raspberry Pi's red led and the Android smartphone's flash are actuators, so they are represented with the number 3. The number 1 shows the social networks which in our example is Twitter.

### *A. Bilrost work cycle*

In this section, we will explain how Bilrost works by using the example defined. Firstly, we must define a context where the devices from the example could be useful. The Android smartphone's flash will be a lantern that will be turned on when the Raspberry Pi's light sensor detects a low value of light level and the Raspberry Pi's led will be an indicator that will be turned on when someone ask for help by shaking the Android smartphone's accelerometer.

Bilrost is capable of making possible this example by using its own DSL which we have called Bilrost-Specific Language (BSL). Using this language, a user can define the technology to use in the project like Android, Java, or Python; the keywords to search messages in Twitter or to add in the messages that contain data from sensors; the data needed by the APIs of each Social Network; the actuators with their actions, the sensors and the rules that automatise the invocation of actuator's actions according to sensors' values and certain conditions. These rules can be formed by sensors and actuators from the device or from other external devices.

However, the projects generated from a device defined with BSL are not completed. Though the connection to Twitter is already implemented, users have to implement the specific logic to access to sensors' data and to control actuators. Due to this, the work cycle of Bilrost can be divided in two steps where the interaction of users is needed as Fig 2 shows, **projects generation,** and **projects completion**.

#### *1) Projects generation*

The first step is the generation of projects of applications that connect Smart Objects to Social Networks as Fig 2 represents with the number 1. In this step, users have to define their devices using BSL. The components of BSL will be explained in a next section. Users can define the two devices

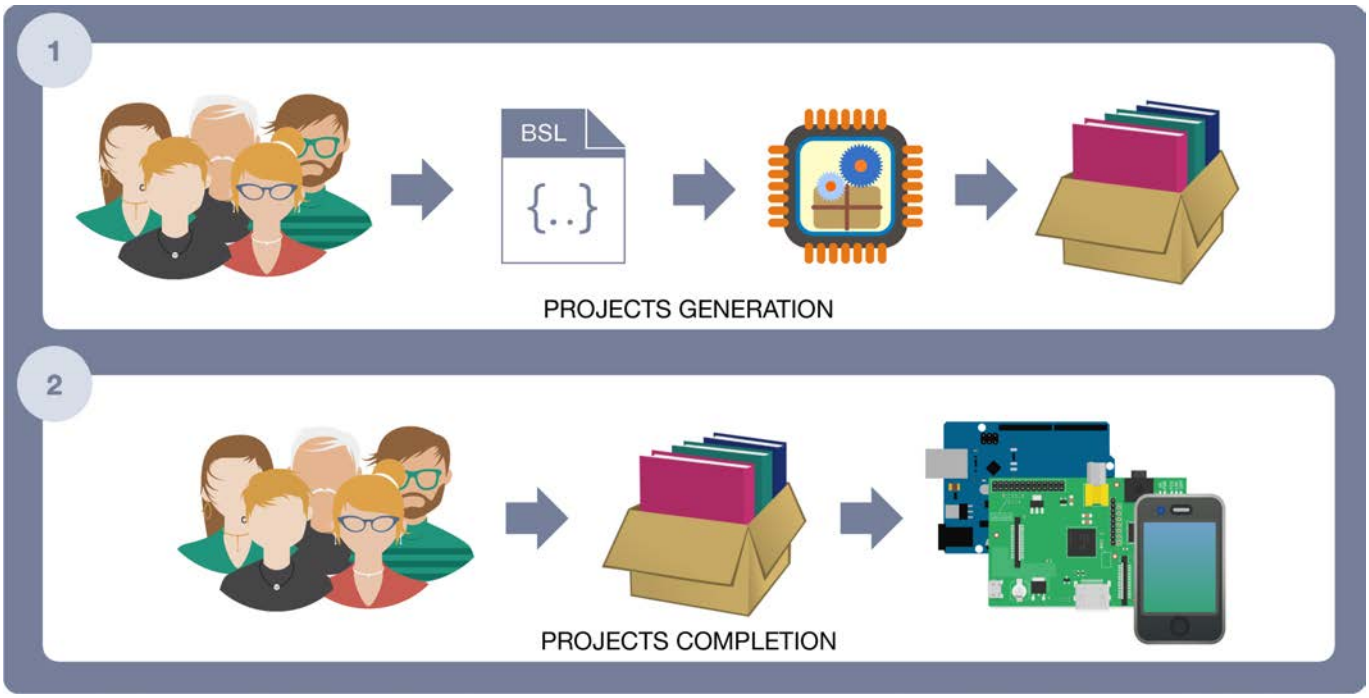



Fig 2. Stages with user interaction: 1. Projects generation and 2. Projects completion.

```
class LedActuator(ActuatorBase):

    def __init__(self):
        super().__init__('led')

    def on_action(self, params):
        # TODO Fill as you want
        pass

    def off_action(self, params):
        # TODO Fill as you want
        pass
```

Fig 3. Actuator code after the project generation.

that represents the Raspberry Pi and the Android Smartphone of our example using BSL. Moreover, users have two options to define the devices, using a textual editor or using a graphic editor. The textual editor uses directly the BSL syntax to create the model, whereas the graphic editor translates a graphic model to the textual model that uses the BSL syntax before generating a project. After define the devices with BSL, Bilrost will generate projects that accomplish the rules defined by the users for each device by processing the definition of the devices in BSL.

Our proposal makes easier the creation of applications to connect Smart Objects to Twitter because it abstracts this part of implementation. Users only need knowledge about how its devices works, how read their sensors, how control their actuators, and know the BSL syntax.

The generated projects use the technology that users indicated in BSL in order to be easier to users implement the missing logic. Fig 3 shows the resultant code of an actuator after generate a project. In this code there are several comments that indicate where to write the specific implementation to control the actuator. Through messages in Twitter, users or other objects will be able to call these methods.

*2) Projects completion*

The last step is the completion of projects. In this step, users have to complete the projects generated by Bilrost. The projects are generated with empty methods that represent the actuators' actions or methods that must access to sensors data according to the device defined with the BSL as we can see in Fig 3. This step is a user step and Bilrost does not take part because our proposal is focused in the interconnection of Smart Objects through Social Networks instead of generating general applications to Smart Objects. Automating this step will be future work to take into consideration in future researches.

### *B. Architecture*

Our system has an architecture that can be divided in four layers: **Bilrost Specific Language Code Generation**, **Specific Logic Implementation** and **Objects Deployment**. The first and second layer are part of our prototype whereas the third and fourth layer do not depend on the prototype. They depend on the user itself.

The first layer is *Bilrost Specific Language Code Definition.* It is composed by the graphic web editor, the textual web editor, and the BSL parser. The graphic editor defines a device using visual elements and generates the device definition in BSL, the textual editor is used to write a device definition in BSL, and the BSL parser interprets the code and generates a JavaScript Object Notation (JSON) which the projects generator will use to create the projects according to the properties defined with BSL.

The second layer is *Project Generation*. This layer takes the result of the previous layer and interprets it to generate the project of application with the connection with social networks already implemented and with the skeleton of the specific logic for actuators and sensors.

The third layer is *Specific Logic Implementation* and it is a task of users. Users have to take the projects generated in the previous layer and complete the specific implementation for the actuators and sensors.

The fourth and last layer is *Objects Deployment*, which consists on integrating the final applications in supported Smart Objects like Raspberry Pi, smartphones, Arduinos, etc.

The following subsections will explain the principal components of our system: Graphic web editor, BSL parser, and project generator, and also we will explain how complete the specific implementation and how the communication works through Twitter.

*1) Graphic web editor*

The graphic web editor is a component which users can use to model application projects. It was created using web technologies in order to be accessible by everyone without considering the operating system and the device. The result obtained from this graphical editor is a device definition written in BSL that Bilrost create from graphic components. There is also available as a textual web editor to write the device definition using the textual syntax of BSL.

Fig 4. Graphic web editor of Bilrost system.Fig 4 shows the aspect of this editor where there are 4 different zones where users can configure the parts of the device definition as we will see when we will explain the BSL syntax.

The first block is *Device Box* and it is marked in Fig 4 with the number 1. This block allows setting the technology used in the generated project and the filters or keywords (called *hashtags* in Twitter) that identify the device in the Social Networks. Both options are required to generate the device definition in BSL.

The second block is *Social Networks Box* and its marked in Fig 4 with the number 2. This block contains the configuration of the Social Networks that the device will use, the parameters needed by their APIs and the users that will be able to communicate with the device. Our prototype is only compatible with Twitter although it is ready to add in an easy way more Social Networks. It will be contemplated as future work.

The third block is *Objects Box* and its marked in Fig 4 with the number 3. Users can use this block to add sensors and actuators to their device with their properties. A sensor needs a name, a mode of work, and filters. An actuator needs a name, filters and the actions that it can perform and that can be controlled over the Social Networks.

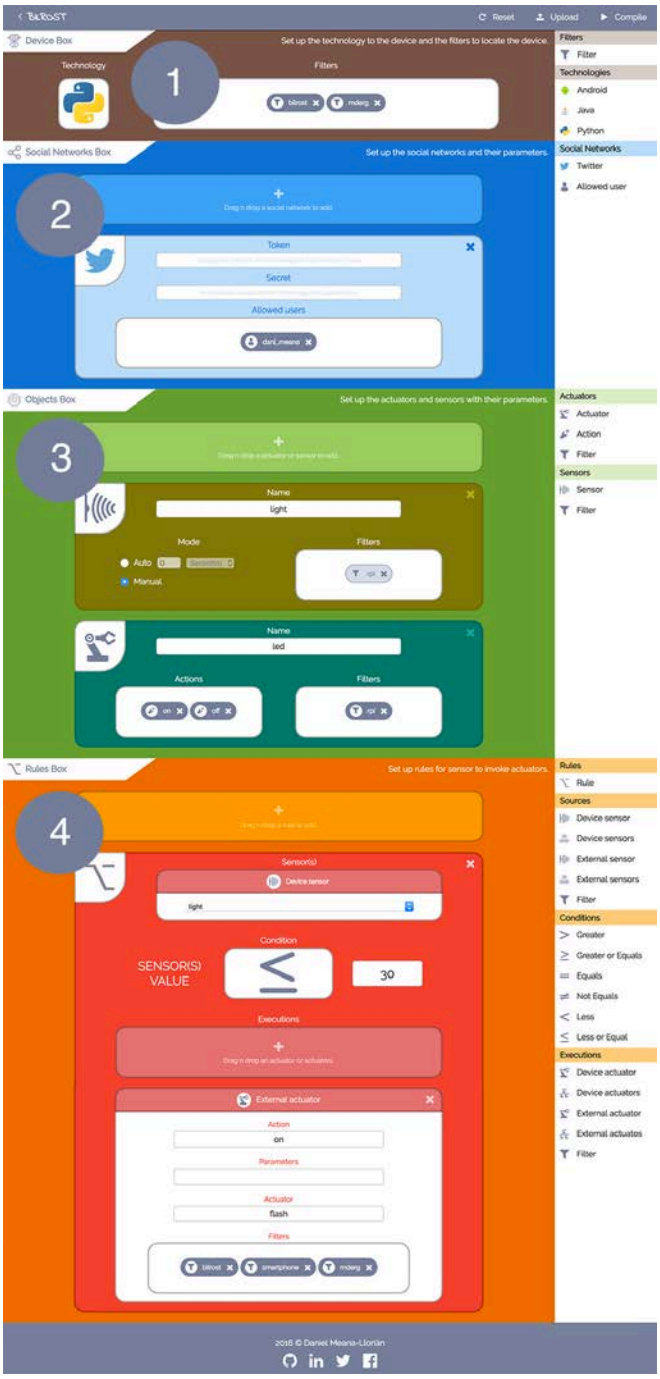



Fig 4. Graphic web editor of Bilrost system.

The fourth and last block is *Rules Box* and its marked in Fig 4 with the number 4. This box is the most complex. Users can use it to add rules that allow the automation of tasks based on rules. Each rule has three different parts: the source of data which is a sensor or a group of sensors, the condition that the data gathered has to accomplish and several executions that will be performed if the conditions is true. Using the example defined before, a user could add a rule that use as source of data the Raspberry Pi's light sensor, the condition could be *"if data is less than a specific value then…"* and the executions would be calling the action *on* of an external actuator called *flash* with the filters used in the definition of the Android smartphone device. The rules enable the use of sensors from the device or from external devices and actuators from the device or from external devices. Thus, a user could define a device that does not have any sensor or actuator but it has several rules that connect an external sensor with an external actuator.

Moreover, the editor is able to save the definition written in BSL to a local file and to load it later to modify the device definition, although, user do not need to read or write the definition in BSL because the graphic editor is capable of generate projects directly. However, the possibility of view the definition in BSL is available.

*2) BSL parser*

The input of the BSL parser is a device definition written in BSL. This definition can come from the graphic editor or from the textual editor, or even from a local file by using command line. The parser performs three steps in order to understand the device definition and to generate a JSON that the project generator use to create projects. These steps are the lexical analysis, the syntactic analysis, and JSON generation. To perform these steps, we have used an implementation in Python of the parsing tools *lex* and *yacc*.

From the example defined in previous section, we need to define two devices in BSL as we can see in Fig 5 which shows the definition of the Raspberry Pi in BSL and Fig 6 which shows the definition of the Android smartphone in BSL. In a next section, we will explain all components of the BSL syntax.

*3) Project generator*

This component takes the JSON generated in the BSL parser to extract the data needed to generate a project. According to that data, the project generator chooses a template of a project that fits the needs and fills it with the data from the JSON.

Currently, the prototype generated is capable of generate projects written in Python, Java, and Android. The resultant projects contain a skeleton with empty methods, as we can see in Fig 3, that users must complete in order to control actuators or share sensor data automatically, or methods that users must

```
DEVICE IN PYTHON
  FILTER BY 'bilrost', 'rpi'
  SOCIAL NETWORKS
    CONNECT TO TWITTER
      TOKEN 'token'
      SECRET 'secret'
  ACTUATORS
    DEFINE ACTUATOR 'led'
      ACTIONS 'on'
  SENSORS
    DEFINE SENSOR 'light'
      MODE MANUAL
  RULES
    DEFINE RULE TO SENSOR 'light'
      IF VALUE IS LESS THAN 30
      EXECUTE ACTION 'on'
        IN EXTERNAL ACTUATOR   'flash'
        FILTER BY 'bilrost', 'smartphone'
```

Fig 5. Definition of the Raspberry Pi from the example in BSL.

```
DEVICE IN ANDROID
  FILTER BY 'bilrost', 'smartphone'
  SOCIAL NETWORKS
    CONNECT TO TWITTER
      TOKEN 'token'
      SECRET 'secret'
  ACTUATORS
    DEFINE ACTUATOR 'flash'
      ACTIONS 'on'
  SENSORS
    DEFINE SENSOR 'shake'
    MODE MANUAL
  RULES
    DEFINE RULE TO SENSOR 'shake'
      IF VALUE IS EQUAL TO 1
      EXECUTE ACTION 'on'
        IN EXTERNAL ACTUATOR 'led'
        FILTER BY 'bilrost', 'rpi'
```

Fig 6. Definition of the Android Smartphone from the example in BSL.

call to share sensors data if they prefer the sensors' manual mode.

In Fig 3, we can see the class of an actuator which is called *led*. It is the Raspberry Pi's actuator of our example. It has two methods, a method to turn on the led and a method to turn off the led. These methods will be called when a message with the appropriate keywords arrives.

*4) Specific implementation*

The output of the project generator component is a skeleton of application in which the connection with Twitter is already implemented. Users only have to complete the project with the specific implementation of each device in order to access to sensor data and to control actuators. Fig 7 shows an example of the final implementation of code from Fig 3 which it is the code of the Raspberry Pi's actuator. This code is the implementation to turn on and turn off a led located in the port number 6.

*5) Communication through Twitter*

The communication through Twitter is made by posting messages or tweets in the Twitter timeline. The structure of a tweet depends on the source of the message. The messages that comes from a sensor, are different to the messages that an actuator needs to perform an action. Even so, both types share a common part. Both messages have the device's filters. We can separate the messages in Twitter in two groups: messages to control an actuator and messages to publish sensor data.

```
import RPi.GPIO as GPIO

class LedActuator(ActuatorBase):

    def __init__(self):
        super().__init__('led')
        self.port = 6
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(self.port, GPIO.OUT)

    def on_action(self, params):
        GPIO.output(self.port, True)

    def off_action(self, params):
        GPIO.output(self.port, False)
```

Fig 7. Example of the implementation of an actuator.

*a) Messages to control an actuator*

The structure of a message to control an actuator is composed by the device's filters, the actuator's filters, the actuator's name, the action to call, and the parameters that the action needs. Moreover, in order to avoid the Twitter limitations, the messages can contain more content without annoying the functionality.

The filters and the name of the actuator are preceded by a hash symbol because they must be *hashtags* of Twitter whereas the action are plain text and the parameters are inside quotes. The rest text in messages is ignored but it is needed because Twitter does not allow posting two identical tweets. When a device controls other device's actuators, the device posts messages with the timestamp in order to avoid this limitation.

In the following lines there are examples of tweets that handle the actuators defines in our example.

- **#bilrost #rpi #led on** – It invokes the action named *on* of the actuator named *led* of the device with filters *bilrost* and *rpi*.
- **#bilrost #smartphone #flash on** – It invokes the action named *on* of the actuator named *flash* of the device with filters *bilrost* and *smartphone.*

In order to present the use of parameters, we will present another example out of the example defined in a previous section.

- **#bilrost #uniovi #climate #thermostat set "22"** – It invokes the action named *set* of the actuator named *thermostat* which has the filter *climate* of the device with filters *bilrost* and *uniovi.* Moreover, it sends the parameter 22 to the action. Parameters must be between quotes and the user has to implement how to parse them in the *project completion* step explained in section A.2).

The filters and the name are the same type of words so it is not possible to know if a hashtag is a name or is a filter. However, the generated code checks all possible combinations that can fit with the defined device.

Moreover, as we said before, Twitter has a limitation avoiding identical tweets. Because of that, when a rule defined in a device calls an action through Twitter, it adds the timestamp as we show in the next example.

- **#bilrost #uniovi #screen show "Hello, my value is 1" 2016-04-05T10:18:36.898989** – It invokes the action named *show* of an actuator named *screen* of the device with filters *bilrost* and *uniovi.* Moreover, it sends the parameter *Hello, my value is 1* to the action.

As we said, a message to control an actuator is composed by filters, the device's name, the action and the parameters. Nevertheless, the device's name is not mandatory and the

system call all actuators that fit the filters if it does not find the name of an actuator in the message.

*b) Messages to share sensor data*

The structure of a message to share data from a sensor is composed by the device's filters, the sensor's filters, the sensor's name, and the sensor's data. Moreover, in order to avoid the Twitter limitations, the messages contain the timestamp.

The filters and the name of the sensor are preceded by a hash symbol because they must be *hashtags* of Twitter whereas the sensor data is inside quotes. The remaining text in messages is irrelevant and it used to prevent from Twitter prohibition of posting two identical tweets.

In the following lines there are examples of tweets that share the state of sensors from our example.

- **#bilrost #rpi #light "50" 2016-04-05T10:18:24.571613** – It shows that the sensor named *light* of the device filtered by *bilrost* and *rpi* has sensed the value 50. It represents that the Raspberry Pi's light sensor has sensed that the light level is 50.
- **#bilrost #smartphone #shake "1" 2016-04-05T10:18:42.649773** – It shows that the sensor named *shake* of the device filtered by *bilrost* and *smartphone* has sensed the value 1. It represents that the Android smartphone's accelerometer sensor has detected a shake.

The rule system works in the same way as actuators work. The system searches sensors that fit the filters and if it finds some sensor, it evaluates the condition and perform the executions by posting a message that control an actuator.

### C. Bilrost-Specific Language

For this proposal we designed a textual DSL that we called Bilrost-Specific Language (BSL). Users can define devices using the textual editor and defining textual models or using the graphic editor defining graphic models. The graphic editor converts the graphic model to the textual model, which uses the BSL syntax, before sending the code to the BSL parser.

BSL does not distinguish between under-case and upper-case, and allows writing all code in a single line or multiple lines mixing upper-case and under-case and changing the order of the different blocks that define a device. Each file written in BSL defines a unique device, hence, users must create as many files as devices they want to define. The definition of a device is composed by four different blocks and the properties of the device.

*1) Device definition*

A device is defined indicating the technology needed to create the application for it and some keywords that identify the device in the Social Networks. These keywords are used to filter the searches in the Social Networks and find messages whose addressee is the device. Also, they are used to post the messages of the device in order to know which device is the source of the data. Moreover, a device has other four blocks: a block to define the Social Networks that it will use, a block to define the actuators that the device will have, a block to define the sensors that the device will have, and a block to specify rules to automatise process based on conditions that sensors data have to accomplish in order to invoke actions from actuators.

In this prototype we implemented the generation of projects written in Python, Java, or Android. Thus, these languages are the technology that the definition require.

The next code is the skeleton to define a device.

```
DEVICE IN PYTHON | JAVA | ANDROID
  FILTER BY …
  SOCIAL NETWORKS …
  ACTUATORS …
  SENSORS …
  RULES …
```

It is mandatory to indicate the technology between the possibilities available, the filters of the device, and the Social Networks. The rest of blocks are optional but at least one must be present.

The filters of each device are useful to identify the device in the Social Network. The device only reacts to messages that contains these filters and all messages that the device posts contain the filters. The max number of filters is not defined but user must have into consideration the limit of characters of messages in the social network. At least one filter is mandatory.

The next code is the skeleton to indicate filters.

```
FILTER BY 'filter1', 'filter2', …
```

*2) Social Networks*

The system has been designed to support several social networks, hence, there is a block to define more than one social networks. This block starts with the words *SOCIAL NETWORKS* which is followed by the configuration of each Social Network. In our prototype, only Twitter has been implemented, so the unique Social Network that users can add is Twitter.

The next code is the skeleton to indicate which social networks will use the device.

```
SOCIAL NETWORKS
    CONNECT TO TWITTER | OTHERS
```

After indicate the social network, the user must define the parameters needed by the Social Network. Twitter needs two user tokens in order to publish and search messages. Moreover, we added another parameter which is the users that can control the device's actuators, the parameter *ALLOW*. This parameter is optional and if it is written, only the messages published by the indicated users will be taken into consideration.

The next code shows the skeleton to configure Twitter as device's social network.

```
CONNECT TO TWITTER
  TOKEN 'tokenvalue'
  SECRET 'secretvalue'
  ALLOW 'user1', 'user2', …
```

*3) Actuators*

A device can have several actuators. To introduce the actuator block is necessary to write the word *ACTUATORS* and

to introduce each actuator is necessary to write the words *DEFINE ACTUATOR* followed by the name of the actuator. All actuators must have a name which will be used to control it through messages in Twitter. Moreover, all actuators must also have a list of actions that they can perform using the word *ACTIONS*. These actions will be the name of the methods that the user will have to fill in the *project competition* step.

Furthermore, each actuator can have filters in order to group several actuators in the same invocation. In this way, one message can invoke an action of many actuators if all of them have filters in common and have that action.

The next code shows the skeleton to indicate the actuators that compose the device.

```
ACTUATORS
  DEFINE ACTUATOR 'name'
    FILTER BY 'filter1', 'filter2', …
    ACTIONS 'action1', 'action2', …
```

*4) Sensors*

A device can have several sensors. To introduce the sensor block is necessary to write the word *SENSORS* and to introduce each sensor is necessary to write the words *DEFINE SENSOR* followed by the name of the sensor. All sensors must have a name which will be used to identify the sensor in the posted messages by the device. Moreover, sensors have a parameter named *MODE*. This is how the sensor has to publish its data. There are two options, the manual mode or the auto mode. The manual mode means that users must call a method to publish the sensor's data when they are in the *project competition* step and the auto mode means that the sensor will publish its data according to the defined frequency. The frequency is defined adding a number after defining the mode, following by the proper unit of time. The unit of time can be seconds, minutes, or hours.

Furthermore, each sensor can have filters in order to group several sensors when a rule is checking sensors' data. In this way, a user can define rules in which the source of data is more than one sensor and if one of these sensors accomplishes the condition then the rule performs the executions.

The next code shows the skeleton to indicate the sensors that compose the device.

```
SENSORS
  DEFINE SENSOR 'name'
    FILTER BY 'filter1', 'filter2', …
    MODE MANUAL |
    MODE AUTO num SECONDS | MINUTES |
                  HOURS
```

*5) Rules*

A device can have several rules that take the data of a sensor, check a condition, and invoke an action of an actuator if the condition is true. The rules are useful to automate the communication between devices.

To introduce the rules block is necessary to write the word *RULES* and then define each rule. Each rule is introduced by the words *DEFINE RULE TO* followed by the sensor or sensors that the device has to consult the data.

The syntax of a rule is more complex than the rest of the language and it is composed by three subsections, the source of data, the condition that the data have to accomplish, and the executions that have to be performed if the condition is true.

The first subblock is the source of data. In this block, users can define that the rule will take the data from a sensor located in the same device, from several sensors also located in the same device, or from a sensor or sensors located in external devices. In the following lines, we will show the different syntaxes for these cases.

- **A sensor located in the same device**: Users have to write the name of the sensor which is the source of data.

```
DEFINE RULE TO SENSOR 'name'
```

- **Several sensors located in the same device**: Users have to indicate the filters that these sensors have in common.

```
DEFINE RULE TO ALL SENSORS
  FILTER BY 'filter1', 'filter2', …
```

- **A sensor located in other device**: In this case, users have to write the name of the sensor and also the filters to search the device where the sensor is located.

```
DEFINE RULE TO EXTERNAL SENSOR 'name'
  FILTER BY 'filter1', 'filter2', …
```

- **Several sensors located in other devices**: As in the previous one, users have to write the filters to search the devices where the sensors are located but, in this case, they must not write the name of the sensors.

```
DEFINE RULE TO ALL EXTERNAL SENSORS
  FILTER BY 'filter1', 'filter2', ...
```

After define the sensors of a rule, users have to indicate the condition that their data must accomplish. For that, they must use the next syntax by choosing one type of condition.

```
IF VALUE IS LESS THAN |
  LESS THAN OR EQUAL TO |
  EQUAL TO | NOT EQUAL TO |
  GREATER THAN OR EQUAL TO |
  GREATER THAN num
```

The last subblock of a rule is the execution block. In this block, users can define several executions that will be perform if the condition is true. All executions must have an action and at least an actuator which has the action. However, an execution can have one actuator or several actuators from the device or from other devices. In the following lines we will be show the different syntaxes for these cases.

- **An actuator located in the same device**: Users have to write the name of the action and the name of the actuator which will do the action.

```
EXECUTE ACTION 'action'
  IN ACTUATOR 'actuator'
```

- **Several actuators located in the same device**: Users have to indicate the filters that the actuators have in common and the name of an action that all actuators can do.

```
EXECUTE ACTION 'action'
  IN ALL ACTUATORS
  FILTER BY 'filter1', 'filter2', …
```

- **An actuator located in other device**: In this case, users have to write the name of the actuator and also the filters to search the device where the actuator is located.

```
EXECUTE ACTION 'action'
  IN EXTERNAL ACTUATOR 'actuator'
  FILTER BY 'filter1', 'filter2', …
```

- **Several actuators located in other devices**: As in the previous one, users have to write the filters to search the devices where the actuators are located and also, filters that all actuators have in common.

```
EXECUTE ACTION 'action'
  IN ALL EXTERNAL ACTUATORS
  FILTER BY 'filter1', 'filter2', …
```

To finalise the explication of the rules block we will show a complete rule definition in the next code.

```
RULES
  DEFINE RULE TO ALL SENSORS
    FILTER BY 'filter1', 'filter2', …
    IF VALUE IS GREATER THAN -1
    EXECUTE ACTION 'action1'
      IN ACTUATOR 'actuator1'
    EXECUTE ACTION 'action2'
      IN ACTUATOR 'actuator2'
```

Fig 5 and Fig 6 shows two examples of devices defined using the BSL syntax.

### D. Used software and hardware

In order to develop this research, the use of different types of software and hardware components were required. The software components are the following:

- The web editors were written using JavaScript ES6 and it uses the library ReactJS 15.0.2. Moreover, we used Webpack 1.13 with Babel to translate the code to ES5, in order to work with all browsers, and to create a bundle with all code.
- The parser's language is Python 3.5.1 and it uses the library Ply 3.8.
- The resulting projects were written for Python 3.5.1, Java 8 and Android 5.1 although there is backward compatibility with Python 3.x and Android 4.x.
- The resulting Python project uses the library Twython 3.4.0.
- The resulting Java and Android projects use the libraries Gson 2.6.2 and Twitter4j 4.0.4.

For the testing the proposal the hardware that we used were a Raspberry Pi 2 with a several electronic components connected to its GPIO and a Galaxy Note 4 with Android 5.1 Lollipop.

## IV. Evaluation

In this section, we will explain the process that we follow to evaluate our proposal by describing the followed methodology, presenting the results that we obtained, and making a discussion about the showed results.

The aim of this proposal is check if making the creation of applications that interconnect objects and people through Social Network easier is possible. Also, as we presented two tools to define the devices, another aim is to decide which tool, between the textual editor and the graphic editor, could be better.

In order to gather data that allow us to know if our proposal is useful and what editor is better, we made an evaluation in which 10 people have participated.

### A. Methodology

The evaluation methodology that we follow have two proposals. Gathering data that allow us to decide what editor is better and gather users' opinion that tell us if our proposal makes the creation of the applications easier than a traditional way. To achieve these goals, we recruit 10 people who have some knowledge about developing any type of applications.

In the followed methodology we can considerer two different phases that gather data for each goal.

- **Phase 1**: In this phase, we want to gather data about how people use both editors of our proposal to complete an assigned task. This task is a basic use case about defining a device which must communicate with another device, already defined, in order to perform a collaborative task.
- **Phase 2**: In this phase, we want to gather the users' opinions about the system after use it through a Likert Scale Survey [63].

#### 1) Phase 1

Before performing this phase, we defined a scenario that emulate a real scenario in which the user wants to connect a new device to another existing device in order to automate some actions.

The scenario was the development of an application that can control the room temperature by handling the air condition system and the heating system, and measuring the temperature with a temperature sensor. The workflow consists on turning the heating system on and turning the air condition system off when the temperature is under a specific value and turning the heating system off and the air condition system on when the temperature is over a specific value.

However, users have not to define all the system, they must take into consideration that a device that can control the heating system is already defined and ready to be controlled through Twitter.

Finally, the task consisted in defining only one device capable of handling the air condition system and measuring the

temperature. Furthermore, it exists another device, which is already working, capable of handling the heating system. The defined device must be capable of automating the regulation of both systems according to the room temperature. If the temperature is lower than 16ºC, the air condition system has to be turned off and the heating system has to be turned on, whereas if the temperature is greater than 25ºC, the air condition system has to be turned off and the air condition system has to be turned on. The temperature must be checked every 30 seconds.

Users must take into consideration that the device, that is already defined, uses the filters *bilrost*, *evaluation*, and *external,* and controls the heating system with an actuator called *heating* and whose actions are *on* and *off*.

To gather data about how people use our system, we gave them a description about how define devices to read without a limit of time. During the reading, the participants could ask any doubt about the system. Moreover, in that description, we also provided an example that define two devices that are interconnected. After the reading, we showed them the system and gave them more time to test it in order to try to remove the learning curve from our results. When the users said that they had understood everything and they were ready, we gave them the task and gave them more time to read it and think about it but without using the system.

Finally, when they had already read and understood the task, the measurement of times started. We took two measures, the time spent in the textual editor and the time spent in the graphic editor. In order to compare how affect what editor is the first that they use, the participants were alternating the first editor that they use. The odd participants used firstly the textual editor and the even participants used firstly the graphic editor. It is important to mention that the participants had the documentation available to consult.

When they finished properly the task in both editors, we started the second phase.

*2) Phase 2*

Once the participants had finished the proposed tasks, they started to fill a survey, anonymously and without help.

To measure and evaluate the survey, we decided to use the Likert Scale [63] because it is one of the most used to design surveys. The survey that we created is a 5-points Likert Scale that gives to the participants the following options: 1 as strongly disagree, 2 as disagree, 3 as neutral option, 4 as agree, and 5 as strongly agree.

The survey contains 14 declarations that ask the participants about their opinion after using both editors. The declarations are related with the possibilities of our proposal, the possible impact in the Internet of Things, the possible impact in Social Networks. These declarations are shown in Table 1.

### B. Results

After doing the evaluation with the users we obtained data from both phases. In the following lines, we will show the results that we obtained from the measurements of each phase.

*1) Phase 1*

Fig 8 shows the time that each participant spent to complete the task with each editors, and the average of all of them with each editor. The lowest time spent in completing the task with the textual editor was 294s whereas with the graphic editor was 201s. The highest time spent with the textual editor was 672s and with the graphic editor was 730s. The average of times spent in the textual editor was 471.5s and with the graphic editor was 383.9s. The standard deviation with textual editor was 113.48s and with graphic editor was 150.16s.

We also collected data grouped by the editor that the participants used firstly in order to know if using one editor before the other products different results. This data is showed in Fig 9.

The group that started with textual editor spent an average of 491.4s in the textual editor, with a maximum of 672s, a

| Question | Description |
|---|---|
| Q1 | The user understands the functionality of the DSL elements, the both editors, and their role in the application creation process. |
| Q2 | This DSL, through both editors, allows interconnecting devices between themselves and with people through Social Networks easily, by using a few code lines and spending little time |
| Q3 | Using this DSL, through both editors, make it more difficult to make mistakes while the user is modelling applications in contrast to making it from scratch |
| Q4 | The proposed solution offers a fast way to developing the indicated task. |
| Q5 | The proposed solution provides assistance to create applications in which there are interconnected devices through Social Networks. |
| Q6 | The proposed solution helps people to be able to interact with devices through Social Networks. |
| Q7 | The TEXTUAL editor does not require to use complex programming skills. |
| Q8 | The GRAPHIC editor does not require to use complex programming skills. |
| Q9 | The DSL and both editors include enough elements and functionality for the user to create a wide range of applications in which there are interconnected devices through Social Networks. |
| Q10 | The DSL and both editors include enough elements and functionality for the user to create a wide range of applications in which people can interact with devices through Social Networks. |
| Q11 | This solution is a positive contribution to encourage the development of services and applications that need to interconnect objects between themselves or with people. |
| Q12 | The Internet of Things can be benefited by this solution. |
| Q13 | Social Networks can be benefited by this solution. |
| Q14 | The graphic editor makes the creation of the interconnection through Social Networks easier than the textual editor |

Table 1. Survey given to the users

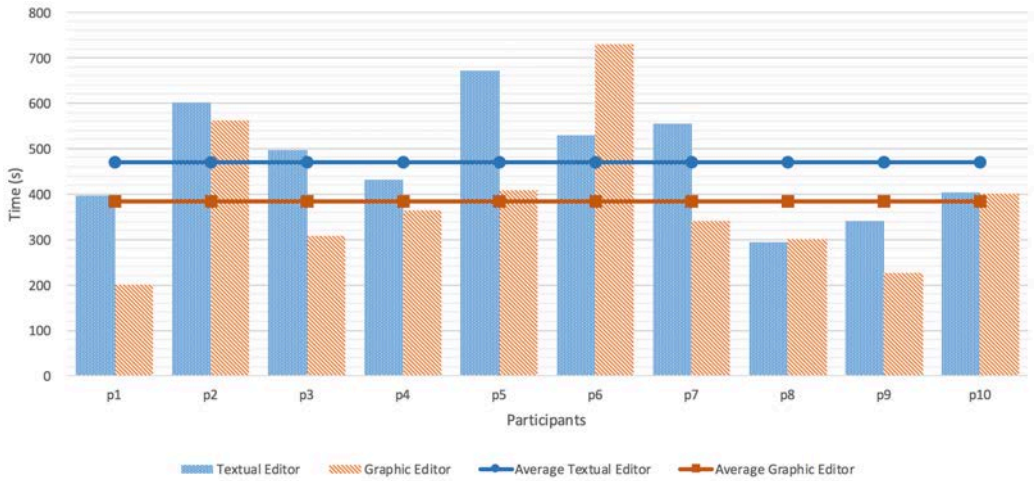



Fig 8. Time to complete the task by the participants.

minimum of 339s, and a standard deviation of 117.55s. This group spent with the graphic editor an average of 296.6s, with a maximum of 408s, a minimum of 201s, and a standard deviation of 75.65s.

The other group, who started with the graphic editor, spent an average of 451.6s in the textual editor, with a maximum of 600s, a minimum of 294s, and a standard deviation of 105.58s. This group spent with the graphic editor an average of 471.2s, with a maximum of 730s, a minimum of 301s, and a standard deviation of 155.33s.

*2) Phase 2*

Table 2 shows the descriptive statistics gathered from the surveys made by the participants. It shows the breakdown of each question: the minimum, the first quartile, the median, the third quartile, the maximum, the range (maximum–minimum), the range between quartiles, and the mode. Moreover, we present the same data as a Box and Whiskers Plot diagram in Fig 10. Moreover, Fig 11 shows a graphic that represents the frequency of each answer for each question.

By other side, the average of the punctuations that each participant give to our proposal through the survey is 4.33 over 5.

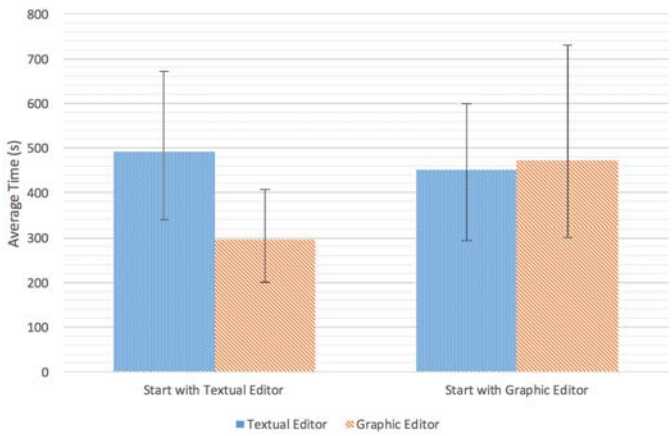



Fig 9. Average time to complete the task according to the first editor used.

## C. Discussion

After gathering the results of each phase, we will analyse them and make a discussion about them.

*1) Phase 1*

From Fig 8, we can conclude that, normally, the use of graphic editor takes less time than the use of the textual editor. However, there are some cases in which the participant spent more time with the graphic editor like the participant 6. This is due to the participants that spent more time in the graphic editor, had started using that editor.

In the following lines, we will do a statistic analysis to conclude if the editor that participants used before, affects the time spent with the second editor.

We used the Shapiro–Wilk test to check the normality of the collected data and we used the F-test to check if the data grouped by the editor used before have equal variances. From these tests we were able to assume a normal distribution of the data with equal variances.

Considering the data obtained from the previous tests, we applied a t-test for paired samples using as null hypothesis that the editor used in first place does not affect the final results. We obtain a *p-value* of 0.001319 to results of starting with textual editor and a *p-value* of 0.698 to results of starting with graphic editor.

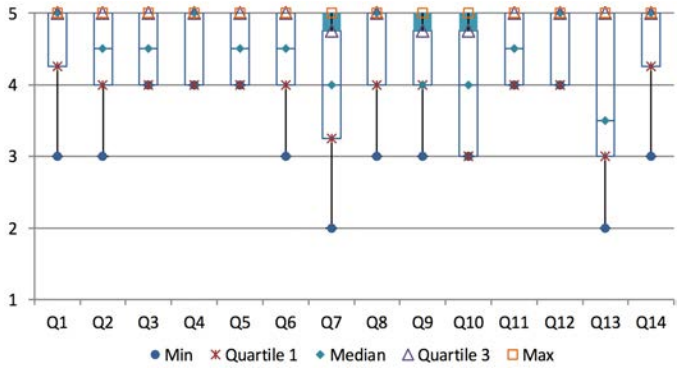



Fig 10. Box and whiskers plot for each question.

| | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 | Q8 | Q9 | Q10 | Q11 | Q12 | Q13 | Q14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Min | 3 | 3 | 4 | 4 | 4 | 3 | 2 | 3 | 3 | 3 | 4 | 4 | 2 | 3 |
| Quartile 1 | 4.25 | 4 | 4 | 4 | 4 | 4 | 3.25 | 4 | 4 | 3 | 4 | 4 | 3 | 4.25 |
| Median | 5 | 4.5 | 4.5 | 5 | 4.5 | 4.5 | 4 | 5 | 4 | 4 | 4.5 | 5 | 3.5 | 5 |
| Quartile 3 | 5 | 5 | 5 | 5 | 5 | 5 | 4.75 | 5 | 4.75 | 4.75 | 5 | 5 | 5 | 5 |
| Max | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Range | 2 | 2 | 1 | 1 | 1 | 2 | 3 | 2 | 2 | 2 | 1 | 1 | 3 | 2 |
| Inter Qrt.-Range | 0.75 | 1 | 1 | 1 | 1 | 1 | 1.5 | 1 | 0.75 | 1.75 | 1 | 1 | 2 | 0.75 |
| Mode | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 4 | 3 | 5 | 5 | 5 | 5 |

Table 2. Table with the general descriptive statistics.

The first *p-value* is less than the significance level (0.05), hence, we can discard the null hypothesis and confirm that using the textual editor in first place, affects the time spent on the graphic editor.

However, the second *p-value* is greater that the significance level (0.05), hence, we cannot discard the null hypothesis and we have to assume that using the graphic editor first does not affect the time spent on the textual editor.

With these results and showing Fig 9, we can conclude that using the textual editor first, makes the use of the graphic editor easier but, the use of the graphic editor in first place does not affect the use of the textual editor. The reason of this behaviour could be that understanding a textual syntax is harder than understanding visual components.

*2) Phase 2*

From Table 2 and Fig 10, we can suggest the following interpretations:

- Q7 and Q13 have the lowest minimum. Some participants disagree with them. These declarations are about the difficulty of making errors using both editors and the necessity of programming knowledge to use the textual editor. Both declarations are related because the necessity of programming knowledge required by the textual editor implies that a user could have errors.
- Q3, Q4, Q5, Q11, and Q12 have the highest minimum. This means that all participants agree with these declarations.
- Apart from Q7, Q10, and Q13, the remaining declarations have the first quartile over 4. This means that, at least, the 75% of participants agree with the declarations. Q7, Q10, and Q13 have the first quartile over 3. This means that, there are less than 25% of participants that disagree with these declarations.
- Q1, Q4, Q8, Q12, and Q14 have the median at the same value as the third quartile and the maximum value. This means that, at least, the half of participants agreed completely with these declarations. Q2, Q3, Q5, Q6, and Q11 have the median over 4 and Q7, Q9, and Q10 have a median of 4. This means that, at least, the 50% of participants agree with these declarations. However, Q13 have the median between 3 and 4, this means that less than the half of participants agree with this declaration.
- Apart from Q7, Q9, and Q10, the remaining declarations have the third quartile at the maximum value. This means that, at least, the 25% of participants agree completely with these declarations.
- The maximum of all declarations is the same and is the maximum value. This means that, at least, exists one participant by each declaration who agree completely with the declaration.
- The mode of Q7 and Q9 was "agree", the mode of Q10 was the neutral option, and the mode of the remaining declarations was "completely agree". This means that, the majority of participants agree with the declarations.

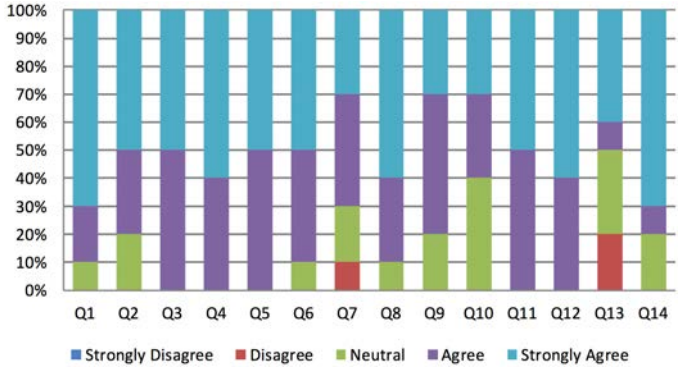



Fig 11. Frequency of responses for each question.

By other side, from Fig 11, we can figure things that could not be figured before:

- Q7 and Q13 are the only declarations that some participant disagree with them. A 10% of participants disagree with Q7 and a 20% of participants disagree Q13.
- Q13 is the declaration with the worst assessment because only the 50% of participants agree with it.
- Q1 and Q14 are the declarations with more participants that strongly agree with them. But the declarations with the best assessment are Q3, Q4, Q5, Q11, and Q12 because all participants agree with them.

The average score of each declaration is 4.33. This means that all participants agree with the whole survey. Thus, we can conclude that the participants have valued positively our proposed and its usefulness to facilitate the generation of applications that interconnect Smart Objects from the Internet of Things with each other and with people through Social Networks.

## V. Related Work

We have finished to present and evaluate our proposal about a novel way of creating communications between objects through human Social Networks where users can join in. There are more researches that address the communication between objects in the literature but there are no many researches that use human Social Networks to establish this communication. Researches that talk about the use of Social Networks and objects are the researches related with Social Internet of Things (SIoT). The researches realised in [22], [62] propose giving social characteristics to objects but without the interaction of humans. They proposed the characteristics that a Social Network must have to be proper to Social Objects [22]. Our proposal differs from SIoT because we do not propose the socialisation of objects, we propose a new way of creating applications, where Smart Objects are connected through Social Networks instead of use common web services.

Another related work is the work proposed in [25]. They talk about the benefits of use Online Social Networks with Web of Things and they explored the use of containers of applications of some Social Networks like Facebook to create applications that integrate Smart Objects in Facebook. Nevertheless, this research does not propose a final approach to interconnect Smart Objects with each other using Social Networks as we do.

The proposal about objects that are capable of sending messages in Twitter was already addressed in [64]. They talk about the possibilities that result from publishing events from Smart Objects like sensors or actuators in Social Networks. Notwithstanding, they do not propose a final system to make easier the publication of events on Twitter as Bilrost does. Bilrost publishes in Twitter sensors' data and invocations of actuators' actions in order to bring closer objects and people.

Social Access Controller (SAC) [20] is a proposal about sharing objects through Social Networks. In that research, authors propose the use of Social Networks to share the connection of Smart Objects although the connection to Smart Objects is not through Social Networks. They propose the use of a system that exposes a public REST API to access and control the registered Smart Objects and has the ability of sharing the access to the Smart Object through Social Networks. However, at least instance, they do no use Social Networks to communicate Smart Objects, but they use Social Networks only to public how to make the connection what it is made by a REST architecture. Our approach does not need an intermediate system to establish the communication between objects because we generate applications which the connection to Social Networks implemented that can run directly in Smart Objects.

There are other researches that address the connection between Smart Objects through REST services instead of using Social Networks. Midgar IoT platform [19], [65] is one of these researches. Midgar allows interconnecting heterogeneous and ubiquitous objects with each other by using a graphic DSL in order to make easier the creation of the connections for people without much knowledge about software development or programming. Whereas Midgar needs a server where register the objects that can be connected through REST architecture. Our proposal uses directly Social Networks avoiding the dependence of a central server. Moreover, Midgar only allows the connection between objects whereas Bilrost allows the connection between objects and people by using of human Social Networks. However, Midgar allows the complete generation of Smart Objects with the specific logic that Bilrost's users must complete.

Other approaches to interconnect objects is the use of instant messages [66]–[68] instead of Web Services or Social Networks. This approach has some disadvantages. It needs specific applications which are exclusive to perform the communication whereas Social Networks allow the use of common Social Networks applications or web browsers to access to sensors or execute actuator's actions.

Paraimpu [69] is a platform created as a social platform for the Internet of Things. Its users can add, use, share, compose, or connect Smart Objects or other supported services like Social Networks. Paraimpu allows creating applications where Social Networks are actuators in order to post messages on Facebook or Twitter when an event happens. However, this approach is not similar to our approach because we do not use Social Networks as sensors or actuators, we use it as a way of establishing a communication between Smart Objects. Moreover, Paraimpu uses Social Networks to login in the platform and to gather user's contacts and friends.

Furthermore, there are also projects that proposed the use of Social Networks to publish and collect data from sensors [70]–[72] in order to achieve an aim but they do not use social networks to communicate the sensors as Bilrost does.

## VI. Conclusions

In this paper, we present Bilrost, a novel proposal that provides a solution to establish communications between heterogeneous and ubiquitous Smart Objects and between these objects and people through human Social Networks. Traditionally, to interconnect Smart Objects in order to achieve a common goal, it is required having physical access to the devices and an infrastructure that enables the interconnection like a central server. Moreover, people were capable of interacting with these Smart Objects directly as if they were another device.

Bilrost is a new system that enables the creation of applications where the interconnection with Social Networks is already implemented. These applications are already ready to allow controlling the actuators of the device where the application is deployed through Social Networks, and also, it is ready to share the state of the device's sensors in Social Networks. The actuator can be controlled by other devices or by people due to use human Social Networks.

In order to facilitate the creation of these applications, we defined a Domain-Specific Language (DSL) that we called Bilrost-Specific Language (BSL) and we created two editors to define devices using this DSL, a textual editor where users can use the BSL syntax and a graphic editor. With this graphic editor, users can define visual models of devices using visual components and after that, the system is able to translate these visual models to textual models which use the BSL syntax.

Through the BSL, user can define the properties of a device, its components like actuators and sensors, and setup the properties required by the APIs of Social Networks. In this paper, we show how Bilrost works, how the communication through Social Networks, specially through Twitter, works, and how the BSL is.

The system is capable of generating projects of applications which already implement the logic needed to establish communications through the Social Networks defined, and provide a skeleton with empty methods that users must complete to access to sensors' data and to control actuators.

As we developed two editors, we made a comparison between them measuring the time that participants spent in completing a specific task with both editors. We obtained that the average of the time needed to complete the task with the textual editor, which was 471.5s, was greater than the average of the time needed to complete the task with the graphic editor, which was 383.9s.

Moreover, we made another comparison between both editors considering if doing a specific task starting with a specific editor affects the results. We obtained that starting the task with the textual editor affects to the time needed to complete the task with the graphic editor but starting with the graphic editor does not affect to the time needed to complete the task with the textual editor. Users who started with the textual editor spent an average of 491.4s in completing the task with the textual editor, and an average of 296.6s in completing the task with the graphic editor. The other group spent an average of 471.2s in the graphic editor, and an average of 451.6s in completing the task with the textual editor.

By other side, the evaluation of our proposal also included a survey so that users evaluated the system after had used it. The questions that made up the survey were about user experience while using the proposed solution. The average of score obtained for each question was 4.33 over 5 which means that participants agreed with the survey and think that Bilrost is useful for interconnecting devices through Social Networks.

Considering the results from the evaluation process, we can say that Bilrost can be useful to connect objects with each other and with people by using Social Networks, answer the questions posed in the introduction confirming our hypothesis.

Integrating Smart Objects in Social Networks developed for people is possible, people can interact with Smart Objects through the Social Networks that they usually use, and a Domain-Specific Language can support the interconnection of Smart Objects through Social Networks.

In conclusion, facilitate the creation of applications that enable the integration of Smart Objects from the Internet of Things in Social Networks, and allow the interaction among objects and between objects and people, is possible.

## VII. Future work

The Internet of Things could be the future and developing solutions to integrate it in our lives is an important issue to address. The research presented in this paper is still not finished and there are many research lines and future work to do from it. In the following lines we proposed some of these future work and research lines.

- **Adding specific logic implementation for some devices**: The applications that our proposal generates are not end-user applications because users must to complete them. Improving the syntax of BSL to specify the specific logic could be a future work. This specific logic could allow people without technical knowledge to generate applications that interconnect Smart Objects through Social Networks.
- **Making a study about the possibilities of each Social Networks to decide the best one to connect Smart Objects:** The proposal of this paper uses Twitter to interconnect objects but there are many other Social Networks that can be useful to connect objects. A future work could be a comparison of current social networks by focusing in the advantages and disadvantages of integrating objects in those Social Networks.
- **Security and privacy**: Currently, our proposal uses the public timeline of Twitter. However, this approach is not valid for all proposals. A future work could be to add options that enable the use of private messages instead the public zone of Social Networks in order to communicate private objects.
- **Using natural language to do the communications**: In the system proposed, we used keywords to recognise what to do. However, a future work could be to avoid using of keywords and try the use natural language in order to integrate the system in our lives in a more natural and comfortable way for users.

## Acknowledgment

This work was performed by the "Ingeniería Dirigida por Modelos MDE- RG" research group at the University of Oviedo under Contract No. FC-15- GRUPIN14-084 of the research project "Ingeniería Dirigida Por Modelos MDE- RG". Project financed by PR Proyecto Plan Regional.

## References

[1] G. G. Meyer, K. Främling, y J. Holmström, «Intelligent Products: A survey», *Comput. Ind.*, vol. 60, n.º 3, pp. 137-148, abr. 2009.

[2] Fundación Telefónica, *La Sociedad de la Información en España 2015*. 2016.

[3] Z. Pang, L. Zheng, J. Tian, S. Kao-Walter, E. Dubrova, y Q. Chen, «Design of a terminal solution for integration of in-home health care devices and services towards the Internet-of-Things», *Enterp. Inf. Syst.*, vol. 9, n.º 1, pp. 86-116, ene. 2015.

[4] Y. J. Fan, Y. H. Yin, S. Member, Y. Zeng, y F. Wu, «IoT-Based Smart Rehabilitation System», *Ind. Informatics, IEEE Trans.*, vol. 10, n.º 2, pp. 1568-1577, 2014.

[5] M. C. Domingo, «An overview of the Internet of Things

for people with disabilities», *J. Netw. Comput. Appl.*, vol. 35, n.º 2, pp. 584-596, mar. 2012.

[6] C. Han, J. M. Jornet, E. Fadel, y I. F. Akyildiz, «A cross-layer communication module for the Internet of Things», *Comput. Networks*, vol. 57, n.º 3, pp. 622-633, 2013.

[7] K. a. Hribernik, Z. Ghrairi, C. Hans, y K.-D. Thoben, «Co-creating the Internet of Things - First experiences in the participatory design of Intelligent Products with Arduino», en *2011 17th International Conference on Concurrent Enterprising*, 2011, pp. 1-9.

[8] D. Ding, R. A. Cooper, P. F. Pasquina, y L. Fici-Pasquina, «Sensor technology for smart homes», *Maturitas*, vol. 69, n.º 2, pp. 131-136, jun. 2011.

[9] G. Chong, L. Zhihao, y Y. Yifeng, «The research and implement of smart home system based on Internet of Things», en *2011 International Conference on Electronics, Communications and Control (ICECC)*, 2011, pp. 2944-2947.

[10] G. M. Lee y J. Y. Kim, «Ubiquitous networking application: Energy saving using smart objects in a home», en *2012 International Conference on ICT Convergence (ICTC)*, 2012, pp. 299-300.

[11] T. Zheng, Y. Qin, D. Gao, J. Duan, y H. Zhang, «A practical deployment of Intelligent Building Wireless Sensor Network for environmental monitoring and air-conditioning control», en *2010 2nd IEEE InternationalConference on Network Infrastructure and Digital Content*, 2010, n.º December, pp. 624-628.

[12] E. Theodoridis, G. Mylonas, y I. Chatzigiannakis, «Developing an IoT Smart City framework», en *IISA 2013*, 2013, pp. 1-6.

[13] L. Hao, X. Lei, Z. Yan, y Y. ChunLi, «The application and implementation research of smart city in China», en *2012 International Conference on System Science and Engineering (ICSSE)*, 2012, pp. 288-292.

[14] R. Y. Clarke, «Smart cities and the internet of everything: The foundation for delivering next-generation citizen services», *Alexandria, VA, Tech. Rep*, 2013.

[15] A. Zanella, N. Bui, A. Castellani, L. Vangelista, y M. Zorzi, «Internet of Things for Smart Cities», *IEEE Internet Things J.*, vol. 1, n.º 1, pp. 22-32, feb. 2014.

[16] R. Lea y M. Blackstock, «Smart Cities: An IoT-centric Approach», en *Proceedings of the 2014 International Workshop on Web Intelligence and Smart Sensing*, 2014, pp. 12:1-12:2.

[17] H. Song, «Internet of things for rural and small town america», en *6th Annual Create West Virginia Training and Education Conference*, 2013, pp. 1-6.

[18] B. M. Howe, Y. Chao, P. Arabshahi, S. Roy, T. McGinnis, y A. Gray, «A Smart Sensor Web for Ocean Observation: Fixed and Mobile Platforms, Integrated Acoustics, Satellites and Predictive Modeling», *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 3, n.º 4, pp. 507-521, dic. 2010.

[19] C. González García, B. C. Pelayo G-Bustelo, J. Pascual Espada, y G. Cueva-Fernandez, «Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios», *Comput. Networks*, vol. 64, pp. 143-158, 2014.

[20] D. Guinard, M. Fischer, y V. Trifa, «Sharing using social networks in a composable Web of Things», en *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, 2010, pp. 702-707.

[21] D. Guinard, V. Trifa, T. Pham, y O. Liechti, «Towards physical mashups in the web of things», en *INSS2009 - 6th International Conference on Networked Sensing Systems*, 2009, pp. 196-199.

[22] L. Atzori, A. Iera, y G. Morabito, «From "smart objects" to "social objects": The next evolutionary step of the internet of things», *IEEE Commun. Mag.*, vol. 52, n.º 1, pp. 97-105, ene. 2014.

[23] J. Miranda, N. Makitalo, J. Garcia-Alonso, J. Berrocal, T. Mikkonen, C. Canal, y J. M. Murillo, «From the Internet of Things to the Internet of People», *IEEE Internet Comput.*, vol. 19, n.º 2, pp. 40-47, mar. 2015.

[24] I. Mashal, O. Alsaryrah, T.-Y. Chung, C.-Z. Yang, W.-H. Kuo, y D. P. Agrawal, «Choices for interaction with things on Internet and underlying issues», *Ad Hoc Networks*, vol. 28, pp. 68-90, may 2015.

[25] M. Blackstock, R. Lea, y A. Friday, «Uniting online social networks with places and things», en *Proceedings of the Second International Workshop on Web of Things*, 2011, pp. 5:1-5:6.

[26] O. Vermesan y Peter Friess, *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*. River Publishers, 2013.

[27] National Intelligence Council, «Discruptive civil tehnologies - six tehnologies with potential impacts on us interests out to 2025», *Conference Report CR 2008 - 07*. Apr, 2008.

[28] K. Ashton, «That 'internet of things' thing», *RFiD J.*, vol. 22, n.º 7, pp. 97-114, 2009.

[29] S. Li, L. Da Xu, y S. Zhao, «The internet of things: a survey», *Inf. Syst. Front.*, abr. 2014.

[30] E. Borgia, «The Internet of Things vision: Key features, applications and open issues», *Comput. Commun.*, vol. 54, pp. 1-31, 2014.

[31] L. Tan y N. Wang, «Future internet: The internet of

things», en *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, 2010, vol. 5, pp. V5-376.

[32] M. Hasan, E. Hossain, y D. Niyato, «Random access for machine-to-machine communication in LTE-advanced networks: issues and approaches», *IEEE Commun. Mag.*, vol. 51, n.º 6, pp. 86-93, jun. 2013.

[33] International Telecommunication Union, «Overview of the Internet of things», *Ser. Y Glob. Inf. infrastructure, internet Protoc. Asp. next-generation networks - Fram. Funct. Archit. Model.*, p. 22, 2012.

[34] Miao Yun y Bu Yuxin, «Research on the architecture and key technology of Internet of Things (IoT) applied on smart grid», en *2010 International Conference on Advances in Energy Engineering*, 2010, pp. 69-72.

[35] O. Ventä, *Intelligent products and systems: Technology theme-final report*. VTT Technical Research Centre of Finland, 2007.

[36] V. Georgitzikis, O. Akribopoulos, y I. Chatzigiannakis, «Controlling physical objects via the internet using the arduino platform over 802.15.4 networks», *IEEE Lat. Am. Trans.*, vol. 10, n.º 3, pp. 1686-1689, 2012.

[37] A. Piras, D. Carboni, A. Pintus, y D. M. T. Features, «A Platform to Collect , Manage and Share Heterogeneous Sensor Data», en *Networked Sensing Systems (INSS)*, 2012, pp. 1-2.

[38] F. Ramparany y O. Boissier, «Smart devices embedding multi-agent technologies for a pro-active world», en *Proc. Uniquitous Computing Workshop*, 2002.

[39] E. Dijkstra, «The humble programmer», *Commun. ACM*, vol. 15, n.º October 1972, pp. 859-866, 1972.

[40] P. Naur y B. Randell, «Software Engineering, Conference Report», *NATO Sci. Committee, Garmisch*, pp. 7-11, 1968.

[41] E. P. González, H. F. Fernández, V. G. Díaz, B. C. P. G. Bustelo, J. M. C. Lovelle, y O. S. Martínez, «General purpose MDE tools», *IJIMAI*, vol. 1, n.º 1, pp. 72-75, 2008.

[42] Brooks, «No Silver Bullet Essence and Accidents of Software Engineering», *Computer (Long. Beach. Calif).*, vol. 20, n.º 4, pp. 10-19, 1987.

[43] B. Selic, «Model-driven development of real-time software using OMG standards», en *Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2003.*, 2003, pp. 4-6.

[44] B. Hailpern y P. Tarr, «Model-driven development: The good, the bad, and the ugly», *IBM Syst. J.*, vol. 45, n.º 3, pp. 451-461, 2006.

[45] E. Seidewitz, «What models mean», *IEEE Softw.*, vol. 20, n.º 5, pp. 26-32, sep. 2003.

[46] B. Selic, «MDA manifestations», *Eur. J. Informatics Prof. IX*, pp. 12-16, 2008.

[47] W. Paper, «Guarantee permanent Model/Code consistency: "Model driven Engineering" (MDE) vs "Roundtrip engineering" (RTE)», 2000.

[48] B. Selic, «Models, Software Models and UML», en *UML for Real*, Boston: Kluwer Academic Publishers, 2003, pp. 1-16.

[49] N. Ford y S. Davis, *No Fluff, Just Stuff Anthology*. Pragmatic Bookshelf, 2006.

[50] A. Van Deursen, P. Klint, y J. Visser, «Domain-specific languages: an annotated bibliography», *ACM Sigplan Not.*, vol. 35, n.º 6, pp. 26-36, 2000.

[51] E. R. Núñez-Valdez, O. Sanjuan-Martinez, C. P. G. Bustelo, J. M. C. Lovelle, y G. Infante-Hernandez, «Gade4all: Developing Multi-platform Videogames based on Domain Specific Languages and Model Driven Engineering», *Int. J. Interact. Multimed. Artif. Intell.*, vol. 2, n.º Regular Issue, pp. 33-42, 2013.

[52] A. van Deursen y P. Klint, «Little languages: Little maintenance?», *J. Softw. Maint.*, vol. 10, n.º 2, pp. 75-92, abr. 1998.

[53] D. Spinellis, «Notable design patterns for domain-specific languages», *J. Syst. Softw.*, vol. 56, n.º 1, pp. 91-99, 2001.

[54] D. Thomas, «MDA: Revenge of the modelers or UML Utopia?», *IEEE Softw.*, vol. 21, n.º 3, pp. 15-17, 2004.

[55] S. Cook, G. Jones, S. Kent, y A. C. Wills, *Domain-specific development with visual studio dsl tools*. Pearson Education, 2007.

[56] M. N. Ko, G. P. Cheek, M. Shehab, y R. Sandhu, «Social-Networks Connect Services», *Computer (Long. Beach. Calif).*, vol. 43, n.º 8, pp. 37-43, ago. 2010.

[57] P. Anantharam, P. Barnaghi, K. Thirunarayan, y A. Sheth, «Extracting City Traffic Events from Social Streams», *ACM Trans. Intell. Syst. Technol.*, vol. 6, n.º 4, pp. 1-27, jul. 2015.

[58] H. Kwak, C. Lee, H. Park, y S. Moon, «What is Twitter, a social network or a news media?», en *Proceedings of the 19th international conference on World wide web - WWW '10*, 2010, p. 591.

[59] D. Doran, S. Gokhale, y A. Dagnino, «Human sensing for smart cities», en *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining - ASONAM '13*, 2013, pp. 1323-1330.

[60] T. Sakaki, M. Okazaki, y Y. Matsuo, «Earthquake Shakes Twitter Users : Real-time Event Detection by

Social Sensors», en *Proceedings of the 19th international conference on World wide web - WWW '10*, 2010, p. 851.

[61] A. Cacho, M. Figueredo, A. Cassio, M. V. Araujo, L. Mendes, J. Lucas, H. Farias, J. Coelho, N. Cacho, y C. Prolo, «Social Smart Destination: A Platform to Analyze User Generated Content in Smart Tourism Destinations», en *New Advances in Information Systems and Technologies*, Springer, 2016, pp. 817-826.

[62] L. Atzori, A. Iera, G. Morabito, y M. Nitti, «The Social Internet of Things (SIoT) – When social networks meet the Internet of Things: Concept, architecture and network characterization», *Comput. Networks*, vol. 56, n.º 16, pp. 3594-3608, nov. 2012.

[63] R. Likert, «A technique for the measurement of attitudes.», *Arch. Psychol.*, vol. 22 140, p. 55, 1932.

[64] M. Kranz, L. Roalter, F. Michahelles, y F. Michahelles, «Things That Twitter: Social Networks and the Internet of Things», en *What can the Internet of Things do for the Citizen (CIoT) Workshop at The Eighth International Conference on Pervasive Computing: Pervasive 2010*, 2010, pp. 1-10.

[65] C. González García, J. Pascual Espada, E. R. Núñez-Valdez, y V. García-Díaz, «Midgar: Domain-specific language to generate smart objects for an internet of things platform», *Proc. - 2014 8th Int. Conf. Innov. Mob. Internet Serv. Ubiquitous Comput. IMIS 2014*, pp. 352-357, 2014.

[66] S. Aurell, «Remote controlling devices using instant messaging: building an intelligent gateway in Erlang/OTP», en *Proceedings of the 2005 ACM SIGPLAN workshop on Erlang*, 2005, pp. 46-51.

[67] J. Choi y C. W. Yoo, «Connect with things through instant messaging», en *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4952 LNCS, Springer, 2008, pp. 276-288.

[68] J. Choi, S. Park, H. Ko, H.-J. Moon, y J. Lee, «Issues for Applying Instant Messaging to Smart Home Systems», en *Computational Science and Its Applications - Iccsa 2009, Pt I*, vol. 5592, Springer, 2009, pp. 649-661.

[69] A. Pintus, D. Carboni, y A. Piras, «Paraimpu: A platform for a social Web of Things», en *Proceedings of the 21st International Conference Companion on World Wide Web (WWW'12 Companion)*, 2012, pp. 401-404.

[70] S. Thomas, Y. Cho, y M. B. Srivastava, «Exploiting Social Networks for Sensor Data Sharing with SenseShare», *Cent. Embed. Netw. Sens.*, 2007.

[71] M. Baqer y A. Kamal, «S-Sensors: Integrating physical world inputs with social networks using wireless sensor networks», en *2009 International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2009, pp. 213-218.

[72] M. Baqer, «Enabling collaboration and coordination of wireless sensor networks via social networks», en *2010 6th IEEE International Conference on Distributed Computing in Sensor Systems Workshops (DCOSSW)*, 2010, pp. 1-2.