AN IOT PLATFORM FOR INDOOR AIR QUALITY MONITORING USING THE WEB OF THINGS

DANIEL IBASETA[1], JULIO MOLLEDA[2], FIDEL DÍEZ[1] & JUAN C. GRANDA[2]
[1]CTIC Technological Centre, Spain
[2]Department of Computer Science and Engineering, University of Oviedo, Spain

ABSTRACT

Platforms populating the Internet of Things (IoT) use dedicated software closely coupled with proprietary hardware, devices and interfaces, which creates silos and a lack of interoperability. The Web of Things (WoT) is a paradigm that incentives the use of web standards to interconnect all kinds of devices and defines an application layer for IoT applications. Multiple organizations and consortiums are pursuing the definition of architectures and standards to deliver interoperability to the IoT application layer. Air quality monitoring is a field in which IoT has a great role to play as it is based on different kinds of sensors and devices which monitor air pollution. Quite a few wireless sensor networks have been proposed in the literature to deal with this monitoring process. In this paper, we propose a low-cost, indoor air quality monitoring platform following the recommendations of the World Wide Web Consortium (W3C) about WoT. The platform is built based on a Web of Things capable of exposing its own Thing Description with 15 to more than 2000 resources, depending on the underlying hardware and the application protocol selected. These resources can serve requests providing measurements of the attached sensors, perform actions on the environment and/or generate events based on these measurements. Although the system is proposed for ambient monitoring, the software architecture developed in this work can be adapted to many embedded applications in the IoT.

Keywords: Internet of Things (IoT), Web of Things (WoT), application layer for IoT, air quality monitoring, ambient air monitoring.

1 INTRODUCTION

The Internet of Things (IoT) is a paradigm combining embedded systems and low-power wireless communication to provide physical objects with Internet connectivity [1]. Services offered by the IoT are the core of smart environments, such as smart homes, smart cities, or smart industries among others. One of the greatest challenges of the IoT is the interoperability of the devices used to create the network of physical objects.

A large number of IoT solutions make use of dedicated software applications coupled with proprietary hardware creating silo solutions that limit the interoperability across different platforms. To overcome this limitation, an application layer is required to communicate among different platforms. The Web of Things (WoT) is a paradigm that focusses on addressing this problem, improving the interoperability and usability of the IoT [2], using open, well-known Web standards to create an application layer for IoT applications. Using these standards, the cyber world and the physical world can communicate through an interoperable infrastructure. The key architectural ideas and technologies enabling the WoT are surveyed in Zeng at al. [3].

One of the fields that can take advantage of the use of IoT is air quality monitoring (AQM). Air is 99.9% nitrogen, oxygen, water vapour, and inert gases. Monitoring the quality of air requires measuring the concentration of several pollutants, such as carbon monoxide (CO), sulphur dioxide (SO_2), nitrogen dioxide (NO_2), and ozone (O_3). In Europe, the threshold levels specified for these pollutants are published in the National Emission Ceilings Directive [4]. These pollutants are released into the air by several human activities and natural sources, harming human health and the environment. Examples of these activities and sources include

WIT Transactions on Ecology and the Environment, Vol 236, © 2019 WIT Press
www.witpress.com, ISSN 1743-3541 (on-line)
doi:10.2495/AIR190051

burning of fossil fuels in electricity generation, transport, industry and households; industrial processes and solvent use, for example in the chemical and mining industries; agriculture; waste treatment; natural sources, including volcanic eruptions, windblown dust, sea-salt spray and emissions of volatile organic compounds from plants.

As in many other fields, most of the IoT devices and services developed for AQM depend on particular platforms or technologies, some of them proprietary, making it difficult to develop a widely accessible application composing all the devices and services. In this paper we propose a low-cost, indoor air quality monitoring (IAQM) platform following the recommendations of the Word Wide Web Consortium (W3C), based on our previous work in this field [5]. The main benefit of this platform is the interoperability of its sensors with other IoT-based AQM platforms, since they are able to provide a description of their features, making it possible to retrieve measurements whatever the manufacturer. The sensors are accessible worldwide through a standard Web browser, or any other application designed to communicate through HTTP or MQTT. Using this type of sensor, a worldwide accessible AQM platform could be conceived allowing, for instance, remote monitoring of geographically separated environments using only open Web standards.

2 WEB TECHNOLOGIES FOR PHYSICAL OBJECTS

Physical objects were connected through Web technologies to create smart environments worldwide more than two decades ago [6]. One of the first reported works about bridging the Web and the physical world was the Cooltown project by HP Labs [7], which explored an infrastructure to support Web presence for people, places and things. Web servers were integrated into physical objects that were accessible through URLs; on top of the infrastructure, Internet connectivity was used. More recently, the IoT was considered part of the Internet [8], avoiding limitations around host-to-host communications, and focusing on the publishing and retrieval of information, which is the most common use of the Internet. Other pioneer solutions [9]–[11] also demonstrated that it was possible to integrate physical objects directly into the Web, or through the use of a gateway, such as Hwang at al. [12]. Modern solutions bridge heterogeneous Web services from the Internet into the IoT network by creating proxies. Thus, clients access transparently to IoT devices and Web services in the network, as in Jin and Kim [13].

The WoT applies Web solutions to access and retrieve information as well as to provide services by physical objects in the IoT. In IoT solutions, each physical object has an associate digital entity called a “Thing”; in WoT each physical object is expressed as a “Web Thing” [14]. A Web Thing commonly exposes metadata in HTML or JSON representation, provides an API to gain access to its properties, and defines an OWL-based semantic description. Web Things may be integrated in the Web in three different ways [15]: (i) hosted by a Web Server embedded into physical objects; (ii) hosted by a Web Server embedded in a gateway device; or (iii) hosted by a Web Server allocated in a cloud service.

Simple static or dynamic Web pages can be used to abstract functionalities of physical objects in WoT; however, reusable Web services are the preferred choice. A Web service is a service designed for machine-to-machine (M2M) communication through the Web. The most commonly used architecture for Web services is the Web Services Architecture (WSA) defined by the W3C [16]. Two categories of Web services are identified: WS-* style and REST style. In the WS-* style, Web services may expose an arbitrary set of operations, and use HTTP as the transportation medium to provide Remote Procedure Calls (RPC). RESTful services, the name given to Web services designed following REST architecture style [17], manipulate representations of Web resources using a uniform set of stateless operations,

where every service is seen as a resource, and each resource is identified by a Uniform Resource Identifier (URI).

In the WoT, physical and virtual entities are abstracted as Web resources. In addition, use of RESTful services is the preferred choice because of their low level of complexity and loose-coupling stateless interaction [2], [3]. Services implementing a RESTful design provide the following features: (i) identification of resources via URI; (ii) uniform interfaces to access the resources, using four HTTP methods: GET, POST, PUT and DELETE; (iii) representation of resources that can be accessed in different formats, such as HTML, JSON or XML, and are self-descriptive as they contain the complete context; and (iv) stateless interaction: the server and clients do not maintain session state as the HTTP request contains all the information about the resource and the message.

2.1 IoT and WoT architectures and standards

Many organizations and consortiums are seeking to define architectures and standards to bring interoperability to the application layer of the IoT for a broad range of applications and industries. They aim to describe, among other scenarios, how to access IoT devices from Web browsers, how to bridge physical objects to the Web, or how to control distributed devices as they are discovered.

Some of the proposals focused on using Web technologies to counter IoT interoperability issues are the following. IEEE proposed a Standard for an Architectural Framework for the Internet of Things [18]; ISO/IEC released a reference architecture for the IoT [19]; ITU-T proposed an architecture where WoT brokers bridge the Web and the physical objects, using HTTP and REST services [20]; OCF developed an open source implementation and a certification program allowing heterogeneous devices to communicate [21]; OGC released a suite of standards to create Web-based interoperable and scalable networks of heterogeneous systems [22]; oneM2M develop architectures to provide interoperability in M2M and IoT solutions [23]; the OpenFog Consortium released an open reference architecture for fog computing [24]; and the W3C proposed recommendations and an architecture for the Web of Things [14].

2.2 W3C WoT

In this work we follow the recommendations of the W3C to overcome interoperability issues in IAQM platforms. In 2016, the W3C launched the Web of Things Working Group (WoT-WG) to propose recommendations and develop standards for the Web of Things. The four main objectives of this group are: (i) counter the fragmentation of the IoT; (ii) reduce the cost of development of IoT solutions; (iii) lessen the risks to both investors and customers; and (iv) foster exponential growth in the market for IoT devices and services. These recommendations are based on scripting languages like JavaScript, data encodings like JSON, and protocols like HTTP and WebSockets, defining also multiple architectures for which such a set of recommendations will be suitable. The W3C Web of Things is devised as the application layer of the IoT, interconnecting existing IoT platforms and complementing available standards.

Following these objectives, the WoT-WG identified four technological building blocks as the key to realizing the WoT [14]: (i) Thing, the abstraction of a physical or virtual entity represented in IoT applications; (ii) Thing Description (TD), the structured data that augments a Thing providing metadata about itself, its interactions, data model, communication and security; (iii) binding templates, the collection of communication

metadata that explains how to interact with different IoT platforms; and (iv) scripting API, an optional building block that eases the development of IoT applications by providing a runtime system for IoT applications similar to a Web browser.

3 INDOOR AIR QUALITY MONITORING USING WIRELESS SENSOR NETWORKS AND IOT

Air quality monitoring is a common IoT application. In Postolache et al. [25], a sensor network for indoor and outdoor air quality monitoring using hardwired and wireless air quality sensors is proposed. This system implements a neural network to provide temperature and humidity compensated gas concentration values. The air quality of different locations can be monitored and published on the Web. The sensors express physical magnitudes through voltage levels that are sent to a network controller and Web server for TCP/IP communication to a PC or Web publishing.

A wireless sensor network based on Libelium nodes is used in Bhattacharya et al. [26] to compute an Air Quality Index (AQI). Each node is composed of a gas sensor board and the Waspmote processing board. The AQI computed is published in a Context Aware Framework to command Heating, Ventilating and Air Conditioning (HVAC) systems. This system is able to control HVAC devices based on building occupancy estimated from CO_2 measurements, and is also able to raise an alarm when the AQI is higher than a given threshold.

A wireless sensor network was also used in Yu et al. [27] to monitor temperature, relative humidity and CO_2. This system addressed the problem of firmware update through special purpose messages in the communication protocol among servers, gateways and sensor nodes. Later, the system was evolved to incorporate analysis and prediction capabilities [28].

An air quality monitoring wireless sensor network based on the Arduino open-source development platform is proposed in Abraham and Li [29]. This system uses XBee modules to provide mesh capabilities based on ZigBee specification, and low-cost micro gas sensors able of measuring six air quality parameters. An estimation method for sensor calibration and measurement conversion was also proposed for this system.

A real-time IAQM system using also wireless sensor networks with the ability to measure the concentrations of six gases in addition to temperature and humidity is proposed in Benammar et al. [30]. This solution emphasizes the use of a gateway in processing collected air quality data and its reliable dissemination to end-users through a Web server. The system uses the Raspberry Pi 2 model B single-board computer for the gateway, the open-source IoT platform Emoncms for the Web server, and Libelium sensor nodes (composed of calibrated sensors, the Gas Pro Sensor Board interface and the Waspmote processing board).

IAQ solutions can also monitor working environments, as in Marques and Pitarma [31], where a real-time system collects an IAQ index, temperature, relative humidity and barometric pressure in a laboratory. This system integrates a Web server and a mobile application to provide historical readings, analysis and push notifications to alert users in a timely manner.

4 INDOOR AIR QUALITY MONITORING PLATFORM USING WOT

All the wireless monitoring systems reviewed above can provide air quality measurements of indoor environments. However, these solutions are designed as standalone systems with restricted and/or proprietary communication protocols creating silos in the IoT. Even though all these systems communicate through wireless processing boards, gateways and Web servers, they cannot to communicate with each other since they do not speak the same language. The sensors of a given system are only able to communicate with the network hardware of the same system.

In this work we follow the W3C recommendations to design an IAQM platform based on current Web standards to overcome interoperability issues. This platform uses low-cost nodes abstracted as Web Things, one of the building blocks of the WoT architecture. Any sensor of this platform can be accessible worldwide from any standard Web browser using several URIs. For instance, requesting a temperature reading from a given sensor would require accessing http://WoTthing/temperature, given that WoTthing is defined in some DNS server and links to the root directory of the Web server of the sensor.

The IAQM platform proposed in this work implements the WoT architecture shown in Fig. 1. This architecture, which is a partial implementation of the W3C WoT recommendation [14], is built based on four blocks: Thing, Thing Description, WoT scripting API, and WoT protocol bindings. In addition, a system API provides access to various sensors. The software components of the proposed platform are developed in MicroPython [32], a software implementation of the Python 3 programming language optimized to run on microcontrollers. Therefore, the processing board for this platform must provide MicroPython support.

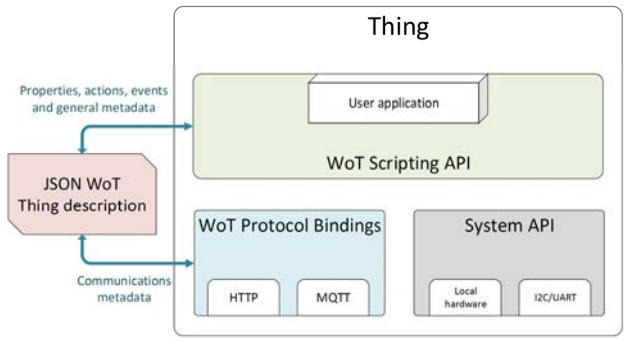

Figure 1: WoT architecture concept.

The scripting API in the proposed IAQM platform exposes Things using two different protocols: HTTP or MQTT.

The exposed description is formatted as required in the W3C WoT recommendations. A Thing can have multiple resources, classified in three groups:

- Properties: A property represents an internal state of a Thing as a value that can be accessed, and sometimes modified, through the corresponding URI.
- Actions: An action is a function that the Thing is capable of performing. Actions can change the internal state of the Thing, manipulate input data or even actuate in the physical world.
- Events: An event is a signal triggered by a change in an internal state or a physical interaction with the Thing. Values that trigger these events may not be exposed as properties.

The interactions are listed and described in the Thing Description, so the larger the number of resources, the larger the JSON file. The WoT protocol binding layer in the proposed IAQM platform manages the handlers and adapters necessary to communicate using the HTTP and MQTT protocols. The HTTP and the MQTT protocol bindings are implemented using the PicoWeb [33] and the AsyncMQTT [34] libraries, respectively. The main benefit of this

platform is asynchrony, as it is built upon the uasyncio library [35], a port of the Python asyncio, which enables Micropython to run different processes in parallel. This allows monitoring different data sources at the same time and maintaining the MQTT or HTTP server online without using threading and avoiding all the complications it implies.

The core of each Thing in the IAQM platform is the processing board. The processing board must provide support for the Thing to communicate with a wide variety of sensors. In addition, it must provide wireless connectivity. The processing board must be selected taking into account the requirements imposed by the application software and firmware that will be deployed in the system.

The Things in the proposed IAQM platform can be run using either of two selected processing boards. Firstly, the DOIT ESP32 DevKit V1, which is designed for mobile, wearable electronics and IoT applications. This processing board features an ESP32-WROOM-32 MCU with two CPU cores (low-power Xtensa 32-bit LX6) that can be individually controlled, CPU clock frequency adjustable from 80 MHz to 240 MHz, 520 KB of on-chip SRAM memory, and 4 MB of flash memory. It also provides Wi-Fi, Bluetooth and BLE wireless interfaces. Secondly, the Pycom GPy processing board can also run in the platform. This processing board uses a dual core ESP32 microcontroller, provides more memory (520 KB of on-chip SRAM memory, 4 MB SRAM memory, and 8 MB of flash memory), as well as Wi-Fi, BLE and LTE wireless interfaces. Both processing boards run native Micropython, with some hardware related differences.

In the proposed IAQM platform, ambient sensors (temperature, humidity, gas) are chosen taking power consumption into account. In the case of gas sensors, there are two measuring techniques: (i) heating an area of the sensor and measuring the chemical reaction in the air; and (ii) measuring particles in the air directly through infrared (IR) sensors. Resistive heating-based gas sensors consume a lot of energy and, thus, greatly reduce the lifetime of battery-powered sensor nodes. In contrast, IR sensors consume less energy since they do not have to heat the sensor. The sensors selected are the MH-Z16 infrared sensor, for measuring concentrations of CO_2, and the fully calibrated SHT25 sensor, for measuring temperature and relative humidity.

The devices in the IAQM platform implement Wi-Fi as the main connection technology. Although it is quite a high consumption communication technology, the latest implementations provide a reasonably low consumption profile (about 100 milliamps while connected) and tools to reduce connection time, as well as deep sleep modes. Another advantage of using Wi-Fi is that it is widely implemented and available, making it simpler to deploy a network of IAQM devices, not needing to implement a custom network and allowing the re-use of existing network infrastructure.

The sensor network will be able to access the internet if the Wi-Fi network has internet connection. A gateway is not required (but it is recommended for security reasons) to access the exposed resources as the IAQM things are designed as standalone servers. The network follows a star topology as seen in Fig. 2.

Air quality monitoring platforms manage data acquired by sensors and provide information that can be analysed, for instance, to optimize the operation of an HVAC system. Usually, such analysis involves processing historical and real-time data. Processing historical data, or data at rest, is a time-consuming task that can be done in batch mode and there is no need for “always on” infrastructure. On the other hand, processing real-time data, or data in motion, usually requires a stream or real-time method running on a low latency infrastructure. Predictive analytics can be used to make decisions supported on the knowledge inferred by the analysis, such as automatically adapt and control the HVAC system. Based on the WoT

model used in this platform, Things can consume each other using WoT methods and automatically make decisions based on that data.

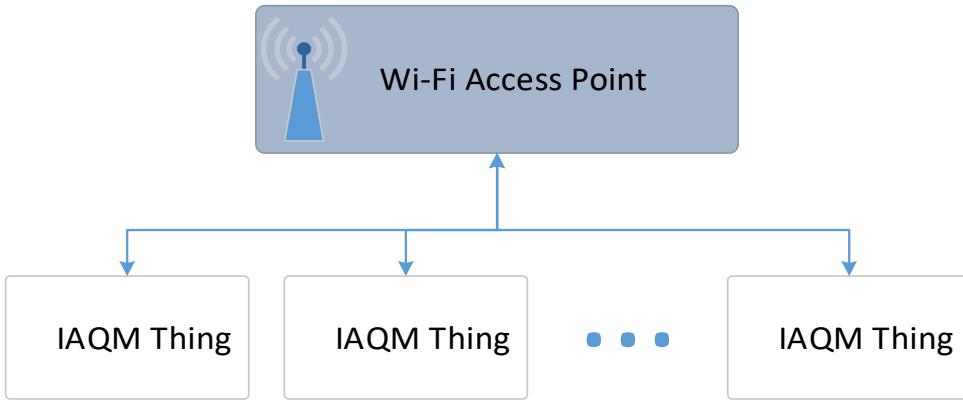

Figure 2: IAQM network topology.

5 EXPERIMENTAL RESULTS

In this section, the proposed WoT architecture and the IAQM platform are evaluated. Systems with the two previously mentioned boards (DOIT ESP32 DevKit V1 and Pycom GPy) will be used. The experiments will be driven using a Lenovo Thinkpad W550s laptop equipped with an Intel Core i7 and 12 GB RAM. A D-Link DIR-655 2.4 GHz Wi-Fi access point is also used to communicate the devices with the laptop.

Two types of experiments have been designed. Firstly, performance tests, in which the capacities of both the hardware and the software components are tested to determine the responsiveness and stability of the platform. Secondly, measurement tests, to check the device while measuring real ambient variables.

5.1 Performance test

The evaluation of the Web of Things is based on load and concurrency tests. Each experiment was run 1000 times. In the HTTP implementation, HTTP REST GET requests are used to access the properties of the Web Thing. In order to make those requests, the ApacheBench tool [36] was used to test the concurrency and load tests. In MQTT, each request consists of a message to a topic and a reply in a response topic which contains the requested property value. MQTT also requires an MQTT broker which is deployed within the same laptop used for the tests.

The load experiment consists of a large number of requests: 1000 sequential requests, with an increasing number of properties exposed. This test was performed for both HTTP and MQTT, although, the concurrency tests are only made for the HTTP implementation, as MQTT is a queuing protocol and the requests are served sequentially, independently of how they are generated.

The concurrency tests create multiple requests at the same time, evaluating how the Thing responses. The Thing holds, for any number of concurrent requests, a constant number of properties exposed in both devices: 45. This value is selected according to DOIT ESP32 board, leaving a margin at its upper limit, since it is the less powerful of the two.

Table 1 shows a brief comparison of the maximum capacities of both processing boards. As can be seen, the Pycom GPy is more powerful than the DOIT ESP32 because it has more RAM, can expose more properties and hold bigger Thing Descriptions, reporting also proportionally better response times. The experiments run in the DOIT ESP32 board return poor results because this board has constrained features. The available RAM memory to run these experiments (less than 500KB) is very low due to the Micropython environment, therefore limiting the amount of properties that can be exposed by the Web Things. In contrast, Web Things deployed on the Pycom GPy, that has 8 times more RAM, are able to expose thousands of properties.

Table 1: IAQM platform load and concurrency tests for two different processing boards.

TESTS	DOIT ESP32		Pycom GPy	
	HTTP	MQTT	HTTP	MQTT
Maximum number of exposed resources	52	15	1,000	2,000
Mean property response time with max. number of resources exposed	334 ms	361 ms	505 ms	388 ms
Mean Thing Description response time with max. number of resources exposed	384 ms	355 ms	836 ms	44,474 ms
Maximum concurrent requests	80	N/A	100	N/A

Fig. 3 shows the average response time exposing the resources through HTTP, using the two different types of processing boards. As can be seen, the response time for properties, which is the most common feature to be used, is always under 500 ms. Although it is a bit high, it depends greatly on the quality of the Wi-Fi access point. A ping to the devices using the same network results in 4 ms on average.

Fig. 4 shows the average response time exposing the resources through MQTT, using both types of processing boards. Using this protocol, the average response time is higher because two topics must be used and the MQTT protocol is not designed to transport large data files, as it is required, for example, to retrieve the TD. It also shows an exponential growth of the TD, because of the amount of resources exposed, that causes the response time to increase gradually as the TD grows in size.

Fig. 5 shows how the devices handle the concurrency of requests. There are two aspects to take in account when analysing these results: (i) Micropython, as well as native Python, is executed in a single thread; therefore, the requests have to be processed one by one in any case; however; (ii) the accumulation of requests creates a different behaviour because there is no time lost between them.

5.2 Measurement tests

To evaluate the stability and reliability of the system in long term operations, a custom software that exposes three resources, each one of them accessible through HTTP GET requests, was developed. The software makes use of the Micropython WoT runtime implementation to expose the properties in a microcontroller (DOIT ESP32 in this case), and another software component, in a remote computer, requests a measurement every 10 mins, and stores it in a Sqlite Data Base.

Fig. 6 shows a six-day period of data gathered with this system. The data loss during this period is lower than to 0.01% and is caused mostly by Wi-Fi disconnections.

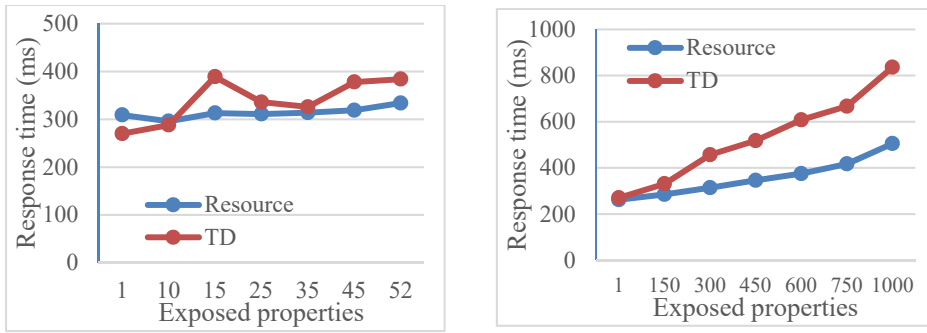

Figure 3: Average response time with HTTP using (a) DOIT ESP32; and (b) PyCom GPy.

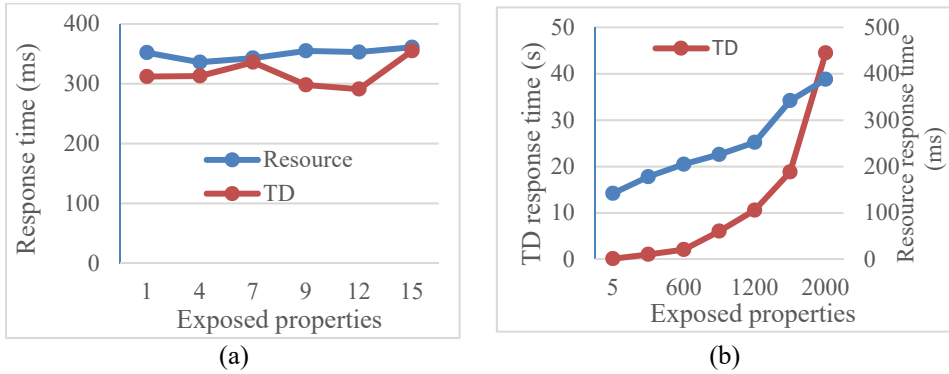

Figure 4: Average response time with MQTT using (a) DOIT ESP32; and (b) PyCom GPy.

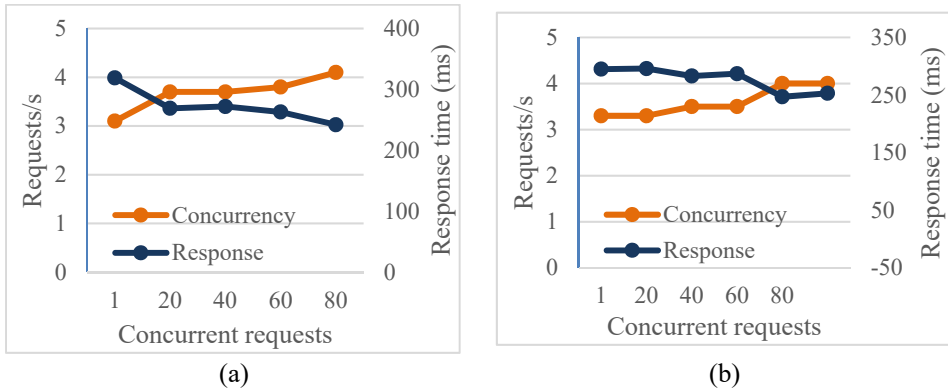

Figure 5: Average number of requests processed and response time with HTTP running on (a) DOIT ESP32; and (b) PyCom GPy.

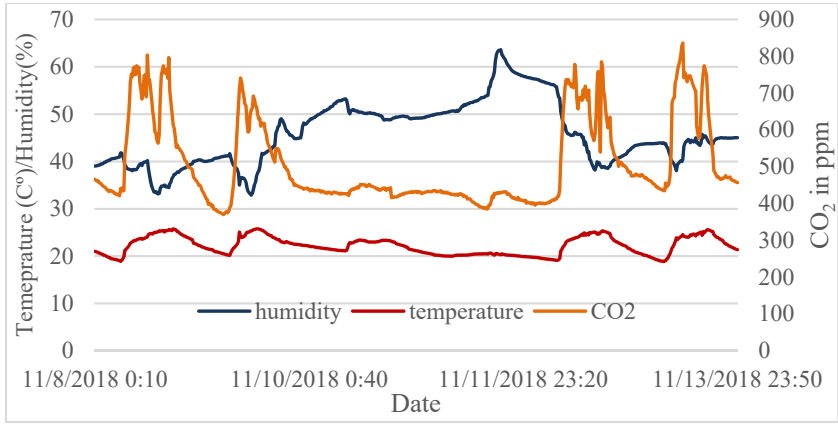

Figure 6: CO_2, temperature and humidity acquired with MH-Z16 and SHT25 sensors.

6 CONCLUSIONS

Air quality monitoring is a key instrument to determine air pollution issues. It is also one of the fields where IoT plays a great role. However, most of the IoT devices and services developed in this field depend on particular platforms or technologies, some of them proprietary, making it difficult to develop a widely accessible application. Using Web standards, the fragmentation between different sensors has been countered, making them able to be accessed through common Web protocols.

In this work, we deal with indoor air quality monitoring and measuring by following the recommendations of the W3C WoT Working Group to design, develop and deploy an ambient sensor network which can be accessed by any standard web browser. The implementation of two protocols, HTTP and MQTT, enables the solution to be more flexible for the desired scenario. The use of WoT standards abstracts access to the useful data from the acquisition process, making it easier to collect, store and process it.

This work is built on our previous work [5], where the W3C WoT standard proposal implementation in an indoor air quality sensor was in an early stage, including few of the required features. In this work, the implementation added and enhanced more building blocks: the Scripting API and Protocol bindings; the Thing description is generated automatically and has more autogenerated metadata required in the proposal. In addition, the performance and interoperability tests of the air quality monitoring sensor were run in two different boards, as well as the interoperability measuring tests.

In future work, the implementation of additional security and Thing discovery will increase the interoperability while reducing the configuration process of Things, fostering the determination of air pollution issues in indoor environments.

ACKNOWLEDGEMENTS

This work has been funded by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 768921 and by the University of Oviedo under project 2018/00061/012.

REFERENCES

[1] Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H. & Zhao, W., A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, **4**(5), pp. 1125–1142, 2017.

[2] Guinard, D., Trifa, V., Mattern, F. & Wilde, E., From the Internet of Things to the Web of Things: Resource-oriented architecture and best practices. *Architecting the Internet of Things*, eds D. Uckelmann, M. Harrison & F. Michahelles, Springer: Berlin and Heidelberg, pp. 97–129, 2011.

[3] Zeng, D., Guo, S. & Cheng, Z., The Web of Things: A survey (invited paper). *JCM*, **6**(6), pp. 424–438, 2011.

[4] National Emission Ceilings Directive, European Environment Agency. www.eea.europa.eu/themes/air/national-emission-ceilings/national-emission-ceilings-directive. Accessed on: 25 May 2019.

[5] Ibaseta, D., Molleda, J., Díez, F. & Granda, J.C., Indoor air quality monitoring sensor for the Web of Things. *Proceedings*, **2**(23), p. 1466, 2018.

[6] Agranat, I.D., Engineering Web technologies for embedded applications. *IEEE Internet Computing*, **2**(3), pp. 40–45, 1998.

[7] Kindberg, T. et al., People, places, things: Web presence for the real world. *Proceedings of the Third IEEE Workshop on Mobile Computing Systems and Applications*, pp. 19–28, 2000.

[8] Atzori, L., Iera, A. & Morabito, G., The Internet of Things: A survey. *Computer Networks*, **54**(15), pp. 2787–2805, 2010.

[9] Guinard, D., Trifa, V., Pham, T. & Liechti, O., Towards physical mashups in the Web of Things. *2009 Sixth International Conference on Networked Sensing Systems (INSS)*, pp. 1–4, 2009.

[10] Akribopoulos, O., Chatzigiannakis, I., Koninis, C. & Theodoridis, E., A web services-oriented architecture for integrating small programmable objects in the Web of Things. *2010 Developments in E-systems Engineering*, pp. 70–75, 2010.

[11] Ostermaier, B., Kovatsch, M. & Santini, S., Connecting things to the web using programmable low-power Wifi modules. *Proceedings of the Second International Workshop on Web of Things*, pp. 2:1–2:6, 2011.

[12] Hwang, K.I., In, J., Park, N.K. & Eom, D.-S., A design and implementation of wireless sensor gateway for efficient querying and managing through world wide web. *IEEE Transactions on Consumer Electronics*, **49**(4), pp. 1090–1097, 2003.

[13] Jin, W. & Kim, D., Development of virtual resource based IoT proxy for bridging heterogeneous web services in IoT networks. *Sensors*, **18**(6), p. 1721, 2018.

[14] Web of Things (WoT) Architecture. www.w3.org/TR/wot-architecture/. Accessed on: 25 May 2019.

[15] Tran, N.K., Sheng, Q.Z., Babar, M.A. & Yao, L., Searching the Web of Things: State of the art, challenges, and solutions. *ACM Computing Surveys*, **50**(4), pp. 55:1–55:34, 2017.

[16] Web Services Architecture. www.w3.org/TR/ws-arch/. Accessed on: 25 May 2019.

[17] *RESTful Web Services* (Book). www.oreilly.com/library/view/restful-web-services/9780596529260/. Accessed on: 25 May 2019.

[18] IEEE 2413-2019 – IEEE Approved Draft Standard for an Architectural Framework for the Internet of Things (IoT). https://standards.ieee.org/content/ieee-standards/en/standard/2413-2019.html. Accessed on: 25 May 2019.

[19] 14:00-17:00, ISO/IEC 30141:2018, ISO. www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/56/65695.html. Accessed on: 25 May 2019.

[20] ITU-T Recommendation database, ITU. www.itu.int/ITU-T/recommendations/rec.aspx?rec=12647&lang=en. Accessed on: 25 May 2019.
[21] Open Connectivity Foundation (OCF), https://openconnectivity.org/. Accessed on: 25 May 2019.
[22] Domains that use and develop OGC standards OG'. www.opengeospatial.org/ogc/markets-technologies/swe. Accessed on: 25 May 2019.
[23] oneM2M – Why oneM2M. www.onem2m.org/about-onem2m/why-onem2m. Accessed on: 25 May 2019.
[24] OpenFog Consortium. www.openfogconsortium.org/. Accessed on: 25 May 2019.
[25] Postolache, O.A., Pereira, J.M.D. & Girao, P.M.B.S., Smart sensors network for air quality monitoring applications. *IEEE Transactions on Instrumentation and Measurement*, **58**(9), pp. 3253–3262, 2009.
[26] Bhattacharya, S., Sridevi, S. & Pitchiah, R., Indoor air quality monitoring using wireless sensor network. *2012 Sixth International Conference on Sensing Technology (ICST)*, pp. 422–427, 2012.
[27] Yu T.-C. et al., Wireless sensor networks for indoor air quality monitoring. *Medical Engineering & Physics*, **35**(2), pp. 231–235, 2013.
[28] Yu T.-C. & Lin, C.-C., An intelligent wireless sensing and control system to improve indoor air quality: Monitoring, prediction, and preaction. *International Journal of Distributed. Sensor Networks*, **11**(8), p. 140978, 2015.
[29] Abraham, S. & Li, X., A cost-effective wireless sensor network system for indoor air quality monitoring applications. *Procedia Computer Science*, **34**, pp. 165–171, 2014.
[30] Benammar, M., Abdaoui, A., Ahmad, S., Touati, F. & Kadri, A., A modular IoT platform for real-time indoor air quality monitoring. *Sensors*, **18**(2), pp. 581, 2018.
[31] Marques, G. & Pitarma, R., An Internet of Things-based environmental quality management system to supervise the indoor laboratory conditions. *Applied Sciences*, **9**(3), pp. 438, 2019.
[32] MicroPython – Python for microcontrollers. http://micropython.org/. Accessed on: 25 May 2019.
[33] Sokolovsky, P., Really minimal web application framework for MicroPython. https://github.com/pfalcon/picoweb. Accessed on: 25 May 2019.
[34] Hinch, P., A "resilient" asynchronous MQTT driver. https://github.com/peterhinch/micropython-mqtt. Accessed on: 25 May 2019.
[35] uasyncio, Core Python libraries ported to MicroPython. https://github.com/micropython/micropython-lib. Accessed on: 30 May 2019.
[36] ab – Apache HTTP server benchmarking tool. https://httpd.apache.org/docs/2.4/programs/ab.html. Accessed on: 30 May 2019.