Universidad de Oviedo

Memoria del Trabajo Fin de Máster realizado por

ISMAEL MELGAR TORNAY

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

SERVIDOR WEB EMBEBIDO - LINUX - ARM

TUTOR: JOSÉ ANTONIO CANCELAS CASO

JULIO-2022

Índice

Figuras

RESUMEN

Con el fin de satisfacer la demanda y la evolución de dispositivos IIOT en el mercado actual, la empresa *Electrónica y Comunicaciones Noreste, S.L.* pretende alcanzar los conocimientos necesarios para lanzar sus propios dispositivos a partir de la investigación y el software desarrollado en este proyecto.

Se plantea la construcción de un sistema de servidores embebidos en Linux sobre un microprocesador de la familia STM32MP1x. Con el fin de monitorizar sensores y actuadoras de plantas industriales, en tiempo real, mediante una conexión a internet.

El proyecto abarca desde la configuración de pines programados a bajo nivel, la confección de una distribución de Linux propia hasta el desarrollo software en el espacio de usuario.

Se crea un prototipo basado en la placa de evaluación OSD32MP1-RED. Sobre ella se implementa un sistema dual de servidores para actuar y monitorizar los dispositivos conectados físicamente a dicha placa. El sistema permite al proyecto alcanzar un entorno general en su uso ya que puede comunicarse tanto con máquinas (HMI, PLCs, etc), basadas en el estándar de comunicación industrial OPC-UA, como con usuarios mediante un navegador web.

La arquitectura software desarrollada permite que ambos servidores sean modulares e independientes pudiendo ser instalados en diferentes dispositivos de una misma red o sobre el mismo.

Como ejemplo, el prototipo desarrollado permite controlar y monitorizar botones y leds, así como un sensor de temperatura y humedad. Todos ellos implementados sobre la placa de evaluación.

CAPÍTULO 1 INTRODUCCIÓN

1.1 Antecedentes

La interacción máquina a máquina mediante la conexión a una red, o también conocido como IoT (Internet of Things), es ya una realidad y no sólo una visión de futuro. Esta tecnología permite mediante diferentes dispositivos la recopilación datos y su posterior análisis, telecontrol e incluso iniciar protocolos de actuación de forma automática.

El departamento de I+D, de la empresa *Electrónica y Comunicaciones Noreste, S.L,* se encargar de los servicios de diseño a medida, industrialización de nuevos equipos y la elaboración de proyectos internos. El rápido crecimiento de la tecnología IoT y las posibles aplicaciones de esta para la evolución hacia la industria 4.0, ha provocado la inversión de recursos por parte del departamento I+D en este campo, con el fin de poder desarrollar sus propios hardware y software genéricos.

Para poder implantar la tecnología IoT se requiere de microprocesadores con una gran capacidad de cómputo, que permitan albergar un sistema operativo. Capaz de administrar módulos complejos (Ethernet, wifi, etc), facilitando así el trabajo a los desarrolladores de aplicaciones. *Electrónica y Comunicaciones Noreste, S.L* apuesta por el fabricante de microcontroladores STMicroelectronics para el desarrollo de esta tecnología, concretamente, la familia STM32MP1. Las características principales de estos microprocesadores son:

- Dos núcleos A7 de 650MHz para la implementación de Linux embebido.
- Un tercer núcleo M4 de 209MHz, completamente independiente, enfocado en tareas de tiempo real.

Para comenzar en el campo de control inteligente, el SOM[1] OSD32MP157C-512M-BAA del fabricante OctavoSystems es óptimo. Este ahorra los excesivos tiempos de desarrollo que conlleva la construcción de la infraestructura de una CPU.

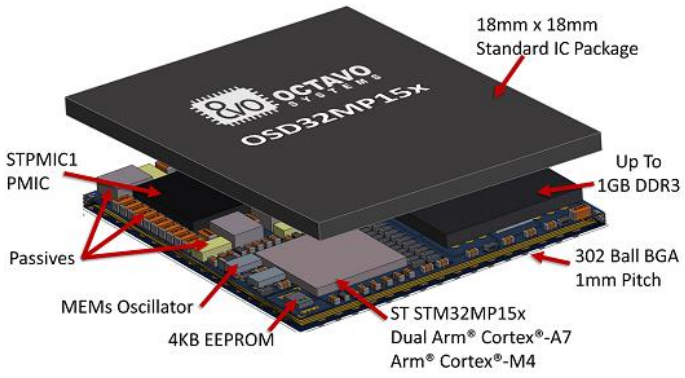


Figura 1. SOM Osd32mp15x.

[1] System-On-Module. Es un paquete con todos los componentes principales de un sistema de procesamiento integrado. Esto incluye los núcleos del procesador, las configuraciones de comunicación y los bloques de memoria implementados sobre una PCB de dimensiones reducidas.

Como placa de evaluación, la empresa propone el modelo OSD32MP1-RED, basada en el SOM anteriormente nombrado. Proporciona acceso a una serie de interfaces de comunicación estándar como Wifi y Bluetooth, Ethernet de 1 Gb y CAN. Es compatible con pantallas HDMI o DSI y tiene un conector para una cámara DCIM. El OSD32MP1-RED se expande fácilmente al proporcionar conectores que son compatibles con Raspberry Pi, MikroElektronika mikroBUS™ y STMicroelectronics Motor Control Header. Además, estas cabeceras pueden ser configuradas de diversas formas como puertos USART, SPI o I2C. Cuenta con una eMMC de 8GB y puerto para tarjeta microSD.

Con sus interfaces de comunicación, periféricos y capacidades de expansión, el OSD32MP1-RED es una plataforma perfecta para desarrollar rápidamente aplicaciones IOT, HMI de alta gama o control en tiempo real.

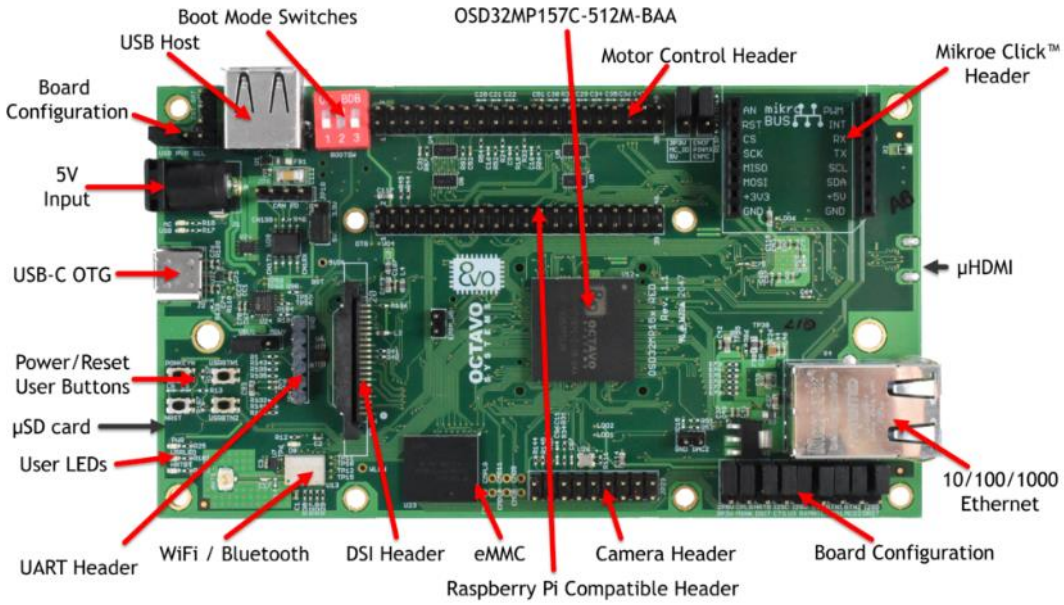


Figura 2. Placa de evaluación OSD32MP1-RED.

Como contexto de este proyecto, *Electrónica y Comunicaciones Noreste, S.L* delega a la empresa *Gradiant (Centro Tecnolóxico de Telecomunicacións de Galicia)* la investigación y documentación de los siguientes puntos, relacionados con la programación de la placa de evaluación:

- Creación de un entorno de desarrollo software.
- Generar y editar el Device Tree de linux.
- Interacción entre los núcleos principales Cortex A7 y el coprocesador M4.
- Desarrollo de drivers para el kernel de Linux.

1.2 Objetivos

El objetivo base de este proyecto es crear, los cimientos en la programación de microprocesadores con Linux embebidos, para el desarrollo de dispositivos inteligentes

por parte de la empresa. Para ello es necesario el entendimiento y configuración a bajo nivel de estos sistemas operativos, permitiendo así configurar diferentes dispositivos adaptados a su uso. Como objetivo de implementación, se propone la creación de un servidor web embebido sobre la placa de evaluación presentada. Dicho servidor deberá permitir la interacción entre usuarios/máquinas externas y los diferentes dispositivos periféricos que pueda controlar el microprocesador.

Como requisito de *Electrónica y Comunicaciones Noreste, S.L* el diseño y la arquitectura del proyecto debe realizarse orientada a una gestión general para un dispositivo que pueda adaptarse y reutilizarse para diferentes proyectos. La creación de una arquitectura dual de servidores modulares e independientes permite abarcar dicho diseño general y robusto. Un primer servidor para la comunicación entre máquinas. Este exportará los datos etiquetados y escalados (si se desea), acercándose así a la conectividad y requisitos que busca la industria 4.0. Un segundo servidor, el cual acceda como cliente al primero e implemente los datos en una página web para el usuario.

Para facilitar el diseño y la documentación de los proyectos mediante modelos de información estandarizados se implementa el modelo físico del estándar ISA-88, sobre los datos exportados. Este permite que la información fluya con facilidad de los niveles bajos de control y programación de una instalación, hacia los niveles superiores donde los datos se monitorizan.

1.3 Alcance y limitaciones

Requisitos funcionales del proyecto:

- Software que permite una comunicación cliente/servidor, entre máquinas (M.2.M.). Permitiendo monitorizar sensores y actuadoras conectados a la placa de evaluación.
- Monitorización de los sensores y actuadores, conectados a la placa de evaluación desde un navegador web.
- Inicio de sesión y registro de nuevo usuarios en el servidor web, así como la actualización de estos.
- Historización y generación de eventos desde el servidor M.2.M.
- Actualización remota del software.
- Base de datos para registrar usuarios y datos de los dispositivos.
- Diferenciar dos tipos de usuarios, el usuario convención, podrá monitorizar datos, además de editar su perfil. Por otro lado, el usuario administrador que puede hacer modificaciones por parte de un técnico.
- Activación de acciones de control entre sensores y actuadores de forma remota.
- Notificación en tiempo real de incidencias sobre los dispositivos.

Requisitos no funcionales:

- Creación de distribución Linux minimalista basada en el proyecto OpenStLinux de STMicroelectronics. Sobre ella se debe implantar el software necesario para los requisitos funcionales.
- Habilitar los módulos wifi, bluetooth, ethernet y la interfaz de cámara conectados al MCU de la placa de evaluación.
- Prioridad en la utilización de lenguaje Python en la medida de lo posible.
- El software del servidor M.2.M. debe utilizar un protocolo estandarizado y ligero.
- La programación del servidor M.2.M. debe ser muy liviana y orientada a objetos. Dicha programación debe ejecutarse y depurarse sobre el IDE de programación que utiliza la empresa, STM32CubeIDE.
- Implantar un modelo de información que permita identificar sensores y actuadores de forma estándar.
- El servidor M.2.M. debe controlar sensores y actuadoras digitales, así como dispositivos que utilicen bus I2C para la comunicación con la placa de evaluación. Ya que la empresa cuenta con drivers para los diferentes tipos de interfaz el interés está en la estructura general del proyecto que pueda implementar dichos drivers.
- Los bucles que implementen las acciones de control deben ejecutarse en paralelo sin ralentizar el funcionamiento de los servidores.
- El servidor web debe implementar protocolos de comunicación Http y websockets.
- Ambos servidores deben poder correr sobre la misma maquina o diferentes. Por ello debe existir una modularidad e independencia entre los servidores.
- Ambos servidores deben iniciarse de forma automática al arrancarse la placa de evaluación.

Queda fuera del alcance de este proyecto:

- Investigación y propuesta de la placa de evaluación.
- Visualización de eventos y datos históricos desde el servidor web.
- Encriptación de datos mediante protocolo https.
- Servidores para su implementación en modo producción, es decir, cumplir con los requisitos de ciberseguridad para la comercialización de estos dispositivos.
- Configuración de red y modem para permitir las conexiones desde IP externas.
- Diseño y desarrollo hardware a medida con el que integrar el software desarrollado.

CAPÍTULO 2 ESTADO DEL ARTE

En este capítulo se presenta la investigación de las diferentes tecnologías disponibles para llevar a cabo el proyecto. Así como la evaluación de la tecnología óptima para cada requisito de proyecto.

2.1 Paquetes de software para STM32MP1

El fabricante STMicroelectronics ofrece una serie de software para microprocesadores STM32.

El **paquete** de inicio para comenzar fácilmente con cualquier dispositivo con microprocesador STM32MP1. Se genera a partir del paquete de distribución.

El **paquete de desarrollador** para agregar sus propios desarrollos además de la distribución de software integrado STM32MPU, o para reemplazar los archivos binarios prediseñados del paquete de inicio. El paquete de desarrollador se genera a partir del paquete de distribución.

- Modificación y puesta a punto de las piezas de software entregadas como código fuente que conforman el Device-Tree.
- Desarrollo del firmware que se ejecuta en el procesador ARM Cortex -M.
- Añadir aplicaciones Linux o frameworks de aplicaciones Linux, así como aplicaciones de confianza (T-A) y de arranque U-Boot.

El **Paquete de Distribución** para crear su propia distribución de Linux, su propia imagen del paquete de inicio y generar su propio paquete de desarrollador. Incluye el código fuente de todas las piezas de software de la distribución STM32MPU Embedded Software (incluidos los marcos de aplicación), además de un marco de construcción basado en OpenEmbedded y Yocto Project.

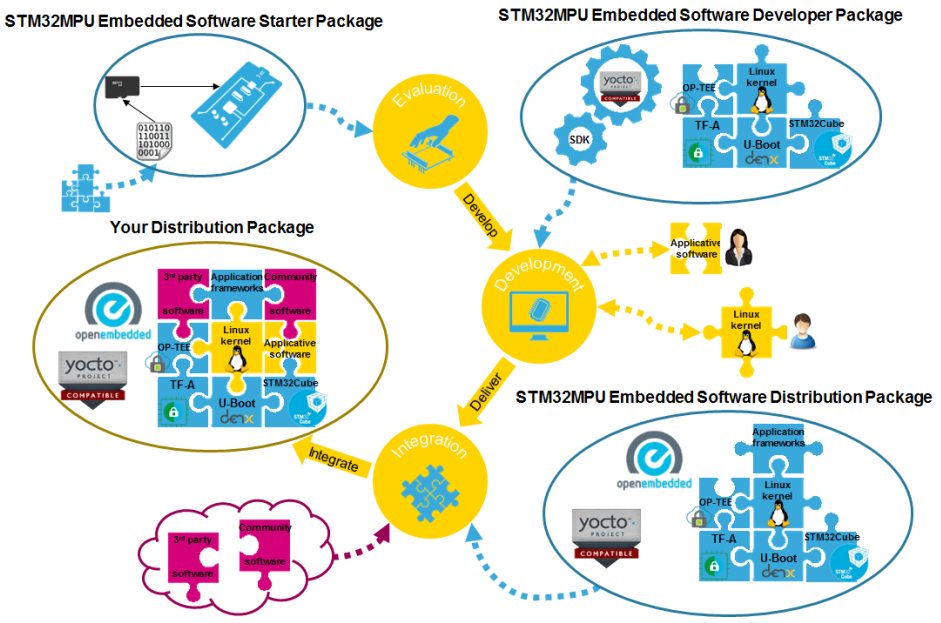


Figura 3. Paquetes para STM32MPU.

2.1.1 Device-Tree

EL árbol de dispositivos es una estructura de datos que permite describir de forma detallada las características de un hardware en concreto. Incluye CPU, periféricos que tiene conectados, configuración de GPIOs, frecuencias de reloj, etc. El sistema Device tree dispone de un compilador (**Device Tree Compiler, DTC)** que toma como entrada un fichero tipo clave/valor escrito por humanos (fichero con extensión **.dts**) y genera una versión optimizada para su uso por una máquina (fichero con extensión **.dtb**). Por otra parte, device tree puede definir sistemas de gran complejidad que pueden hacer difícil la tarea de trabajar con grandes ficheros .dts. Además de esto, a menudo existen sistemas hardware que comparten entre sí multitud de componentes lo que resulta en archivos .dts que repiten una y otra vez los mismos fragmentos de configuración. Como solución a esto Device tree implementa mecanismos de organización del código en distintos ficheros lo que permite su reutilización entre distintas configuraciones hardwares. Para ello se crea un nuevo tipo de fichero, en este caso con extensión **.dtsi**

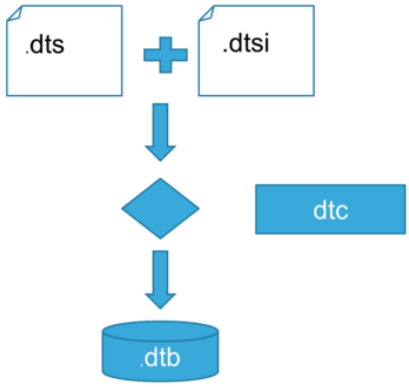


Figura 4. Device Tree.

En los sistemas STM32MP1 el device tree se utiliza en tres componentes:

- **TF-A:** Trusted Firmware-A, Se trata de una de las piezas necesarias en el proceso de arranque seguro de los procesadores ARM, en este caso los cortex A (ARMv7 y ARMv8).
- **U-Boot:** Encargado de iniciar los periféricos necesarios y de transferir el control al kernel de Linux de forma ordenada. En los dispositivos STM32MP1 U-Boot implementa una cadena de arranque en dos etapas que permite cargar y arrancar un firmware en el coprocesador M4 para posteriormente transferir el control al kernel de Linux que se ejecutará en los núcleos A7.
- **Kernel Linux:** Es la última pieza de la cadena y la base del sistema operativo Linux que gestiona todo el hardware del dispositivo. El kernel necesita para su arranque que se le proporcione un archivo de device tree **.dtb** en el que se incluya la descripción detallada del hardware sobre el que se ejecuta.

2.1.2 Yocto Project

Se trata de un proyecto de la Fundación Linux de código abierto. Permite crear distribuciones Linux personalizadas para software embebido e IoT que sean independientes de la arquitectura subyacente del hardware. El proyecto fue anunciado por la Fundación Linux en 2010 y lanzado en marzo de 2011, en colaboración con 22 organizaciones, incluyendo OpenEmbedded[2].

Yocto Project proporciona herramientas interoperables, metadatos y procesos que permiten el desarrollo rápido y repetible de sistemas embebidos basados en Linux en los que cada aspecto del proceso de desarrollo puede ser personalizado.

Yocto Project se basa en un modelo de capas para el desarrollo y creación de Linux embebido que lo distingue de otros sistemas de compilación simples.

El modelo de capas está diseñado para admitir tanto la colaboración como la personalización al mismo tiempo. Las capas son repositorios que contienen conjuntos de instrucciones relacionadas que le indican al sistema de compilación qué hacer. Los usuarios pueden colaborar, compartir y reutilizar capas. Las capas pueden contener cambios en las instrucciones o configuraciones anteriores en cualquier momento.

Esta poderosa capacidad de anulación es lo que le permite personalizar capas colaborativas o proporcionadas por la comunidad anteriores para adaptarse a los requisitos de su producto.

2.2 Servidores M.2.M.

La comunicación machine to machine o M2M describe el intercambio de información automatizado entre terminales sin intervención humana. Esta tecnología se usa en diferentes ámbitos, desde la supervisión, el control y la regulación de máquinas y autómatas.

Este tipo de comunicación permite transmisiones más rápidas y la posibilidad de planificación temporal de las transmisiones de datos bajo un reducido consumo energético. También ofrece el control remoto de dispositivos, una demanda de mantenimiento y una probabilidad de fallo muy bajas, lo que reduce el nivel de gastos en los dispositivos.

Este tipo de tecnología es óptima para el desarrollo IoT y el control remoto de instalaciones, destacando los protocolos MQTT y OPC-UA como los más utilizados en dichos entornos.

[2] OpenEmbedded permite a los desarrolladores crear una distribución completa de Linux para sistemas integrados.

2.2.1 MQTT

MQTT (Message Queque Telemetry Transport) Es un protocolo de comunicación M2M de tipo message queue. MQTT fue creado por el Dr. Andy Stanford-Clark de IBM y Arlen Nipper de Arcom (ahora Eurotech) en 1999 como un mecanismo para conectar dispositivos empleados en la industria petrolera.

MQTT está diseñado para poder ser implementado en dispositivos de bajos recursos, en redes de bajo ancho de banda y de alta latencia. Está basado en la pila TCP/IP como base para la comunicación. En el caso de MQTT cada conexión se mantiene abierta y se "reutiliza" en cada comunicación.

Se basa en un modelo suscripción – publicación (pub-sub). En el que los publicadores envían mensajes a un servidor, denominado bróker, y este es quien reenvía los mensajes a los suscriptores. De esta forma se evita las conexiones punto a punto, esto permite que los suscriptores no necesiten saber quién proporciona la información a la que está suscrito.

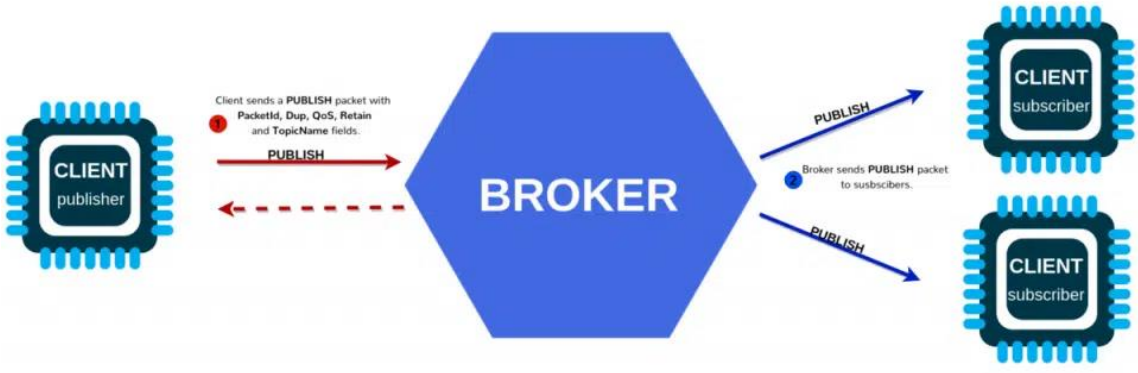


Figura 5. Protocolo de comunicación MQTT.

La conexión cliente – servidor para el envío de los mensajes se hace mediante una conexión TCP o una conexión TLS cifrada en caso de necesidad de envío de mensajes seguros.

El mensaje puede tener cualquier formato de datos para la carga útil, como JSON, XML, binario cifrado o Base64. Esto da mucha flexibilidad al protocolo, pero el cliente destino debe poder analizar el tipo de carga.

2.2.2 OPC-UA

La Arquitectura Unificada OPC (**OPC UA**) es un protocolo de comunicación máquina a máquina para la automatización industrial desarrollado por la Fundación OPC .

En los servidores OPC UA se definen los servicios que pueden proporcionar a los clientes, un modelo de objetos que puede ser descubierto dinámicamente por los clientes y un modelo de datos siguiendo los data-types definidos en el propio protocolo.

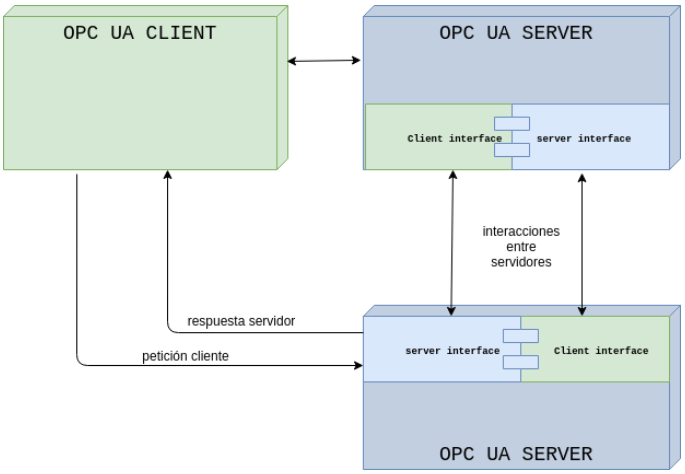


Figura 6. Protocolo de comunicación OPC-UA.

Las características distintivas son:

- Basado en una comunicación cliente-servidor, extensible a protocolos de suscripción-publicación.
- Código abierto: disponible gratuitamente e implementable bajo licencia GPL 2.0.
- Multiplataforma. No está vinculado a un sistema operativo o lenguaje de programación. Facilita la forma de operar entre los dispositivos de distintos fabricantes estableciendo una forma común de comunicación, esto lo hace especialmente útil en entornos industriales.
- Arquitectura orientada a servicios (SOA). Se basa en la creación de nodos con características.
- Proporciona mecanismos de seguridad para la autentificación de los usuarios, clientes y servidores, la comprobación de la integridad de las comunicaciones y aportar seguridad a nivel de transporte encriptando y firmando los mensajes.
- Existe una extensa documentación sobre el estándar para definir la arquitectura a programar.

2.2.3 Justificación de la tecnología escogida

Para este proyecto se escoge la tecnología OPC-UA para entornos industriales. Buscando unificar la forma de operar con distintos fabricantes de dispositivos industriales siendo una solución completa que abarca desde la comunicación de los dispositivos a su interoperabilidad.

MQTT no tiene implementada la búsqueda del topic ni del broker donde se publica el mensaje, de tal forma que el cliente debe conocerlo previamente. De la misma forma tampoco tiene ningún mecanismo implementado para la escritura de variables. Todo ello sí que es soportado por OPC-UA.

2.3 Servidores WEB

El rol principal de un servidor web es almacenar y transmitir el contenido solicitado de un sitio web al navegador del usuario. El servidor web es responsable de garantizar todos los archivos propios de una página web (imágenes, textos, videos, etc.) y transmitirlos a los usuarios a través de los navegadores.

Un Hosting es un servicio de alojamiento de páginas web. Y un Servidor web, es un software programado para devolver información, al cliente que la solicita.

El hosting, sirve para alojar el contenido de un sitio web. Pero para que ese contenido sea visitado por alguna persona en internet, debe estar almacenado en un servidor web. De tal forma, existen dos tipos, servidores dedicados y el otro, los servidores compartidos.

En los servidores web dedicados, tienes la libertad de configurarlo, personalizarlo y controlarlo. También ofrecen una mayor escalabilidad y mayores recursos. Esto indica que, la sobrecarga o falta de espacio en el servidor no serán un problema.

Por otro lado, los servidores compartidos tienen como principal ventaje el ser configurados y manejados por las empresas que te brindan el servicio.

Los servidores web, a diferencia de los servidores M.2.M, están diseñados para transferir la información a usuario a través del navegador web. El diseño gráfico y robustez adquieren mayor importancia, disminuyendo los requisitos de latencia en las comunicaciones.

En general los servidores web se dividen en dos partes o estructuras del proyecto:

- El **Frontend** es la parte de un programa o dispositivo a la que un usuario puede acceder directamente. Son todas las tecnologías de diseño y desarrollo web que corren en el navegador y que se encargan de la interactividad con los usuarios (HTML, CSS, JavaScript).
- El **Backend** es la capa de acceso a datos de un software o cualquier dispositivo, que no es directamente accesible por los usuarios. Además, contiene la lógica de la aplicación que maneja dichos datos (Java, C ++, Ruby on Rails, PHP o Python).

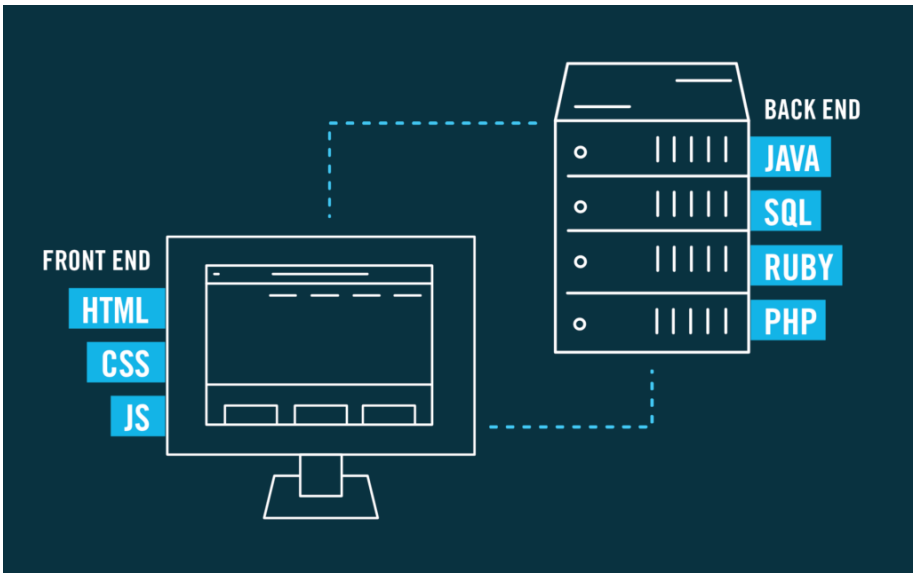


Figura 7. Back-end y front-end de servidores web.

2.3.1 Servidores tradicionales

Existen multitud de servidores web, cada uno con sus propias características, a continuación, enumerar los más destacados:

- **Apache**. Es el servidor web de referencia, el más popular y extendido. Lleva 25 años siendo el líder indiscutible, a pesar de que nuevos competidores le hayan robado cuota de mercado. Es de código abierto y gratuito, lo que permite ser instalado en la gran mayoría de sistemas operativos existentes. Su desfasada arquitectura frente a otros tipos de servidor es su principal punto débil.
- **Nginx**. También es código abierto (aunque ofrece una versión comercial), destaca por su alto rendimiento. Utiliza un proxy inverso, que protege la identidad de los servidores y mejora la seguridad de la información que acogen. Su configuración es sencilla, pero también muy personalizable, consumiendo pocos recursos. No se puede integrar con PHP de forma nativa.
- **Lighttp**. Alternativa a Apache, permite gestionar grandes cargas de trabajo. Por ello, su diseño busca ante todo la rapidez operativa, algo que consigue consumiendo pocos recursos. Esto lo convierte en una opción idónea para servidores VPS (servidor virtual privado) de bajos recursos. Sin embargo, carece de versión oficial para sistemas operativos Windows, lo que limita su aplicación.

2.3.2 Microframework Python-Flask

Un framework es un entorno de trabajo con un conjunto de herramientas, librerías y módulos ofreciendo una estructura base para elaborar proyectos con objetivos específicos. Similar a una plantilla que sirve como punto de partida para la organización y desarrollo de software.

Este micro-framework web permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código en lenguaje Python. Se caracteriza por:

- Se adapta a cada proyecto instalando extensiones específicas para el mismo.
- Incluye servidor web propio para pruebas basado en la especificación WSGI de Werkzeug.
- El diseño minimalista de su estructura le permite ser rápido y con un gran desempeño.
- Cuenta con documentación extensa para el desarrollo de aplicaciones.
- Es muy sencillo de utilizar, por lo que es el indicado para empezar a programar con Python.
- Se integra con otras herramientas para incrementar sus funciones, como Jinja2 (motor de plantillas web) o SQLAlchemy (kit de herramientas SQL de código abierto).

2.3.3 Framework Python-Django

Django es lo que se conoce como un framework "*full stack*", con el cual se pueden abordar todo tipo de proyectos en este lenguaje como el desarrollo web escalable y de alta calidad.

Las características principales son:

- Cuenta con un sistema de autentificación de usuarios.
- Manejo de versiones que permite una distribución simple de actualizaciones.
- Ofrece un gran rendimiento y flexibilidad, pudiendo escalar proyectos de forma sencilla.
- Trabajar bajo un patrón MVC (Modelo Vista Controlador), lo que permite un desarrollo ágil y reutilizable.
- Incorpora una amplia variedad de paquetes de librerías (más de 4000).
- Dispone de una inmensa comunidad de usuarios en internet.
- Incluye opciones de protección para las aplicaciones, por ejemplo, contra ataques de SQL injection o ataques XSS (cross site scripting).
- Proporciona una estructura de código autogenerado.
- Cuenta con panel de administración para bases de datos.

2.3.4 Protocolo HTTP

El Protocolo de Transferencia de HiperTexto *(Hypertext Transfer Protocol)* es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP.

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL.

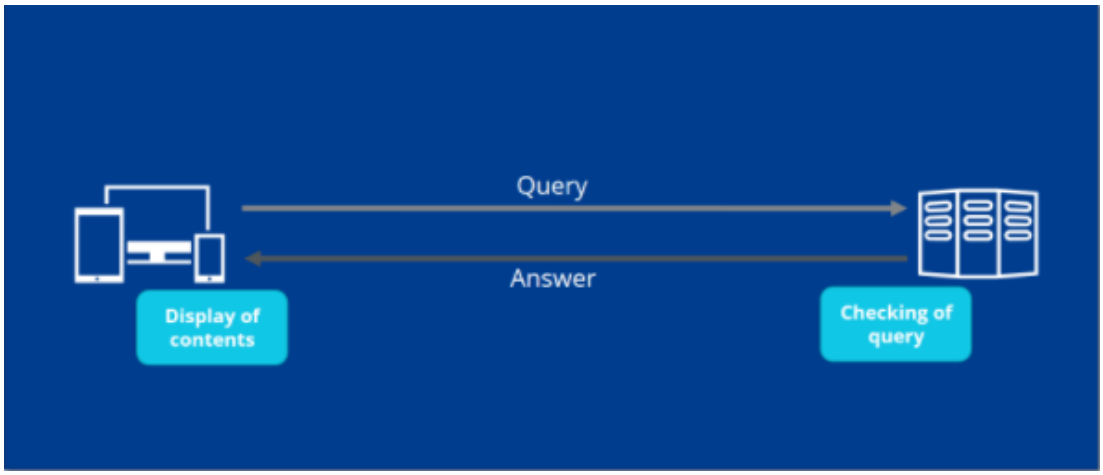


Figura 8. Protocolo HTTP.

Las principales características del protocolo HTTP son:

- Toda la comunicación entre los clientes y servidores se realiza a partir de caracteres de 8 bits. De esta forma, se puede transmitir cualquier tipo de documento: texto, binario, etc.
- Permite la transferencia de objetos multimedia. El contenido de cada objeto intercambiado está identificado por su clasificación MIME.
- Los tres principales métodos que un cliente puede utilizar para dialogar con el servidor son: **GET**, para recoger un objeto, **POST**, para enviar información al servidor y **HEAD**, para solicitar las características de un objeto (por ejemplo, la fecha de modificación de un documento HTML).
- Cada operación HTTP implica una conexión con el servidor, que es liberada al término de esta. Es decir, en una operación se puede recoger un único objeto.
- No mantiene estado. Cada petición de un cliente a un servidor no es influida por las transacciones anteriores. El servidor trata cada petición como una operación totalmente independiente del resto.
- Cada objeto al que se aplican los métodos del protocolo está identificado a través de la información de situación del final de la URL.

2.3.5 Protocolo WEBSOCKETS

Se trata de un protocolo de red basado TCP/IP. Con WebSocket, basta con que el cliente establezca una conexión con el servidor, que se confirma mediante el ***WebSocket***

Protocol Handshake. Con él, el cliente envía al servidor todos los datos de identificación necesarios para el intercambio de información.

El canal de comunicación queda abierto tras el *handshake*. El servidor puede activarse por sí mismo y enviar información al cliente, sin necesidad de recibir una solicitud específica para ello.

Para iniciar el intercambio, con WebSockets el cliente envía una solicitud, al igual que en el clásico HTTP. Sin embargo, la conexión se establece mediante TCP y permanece abierta tras el ***handshake*** entre el cliente y el servidor.

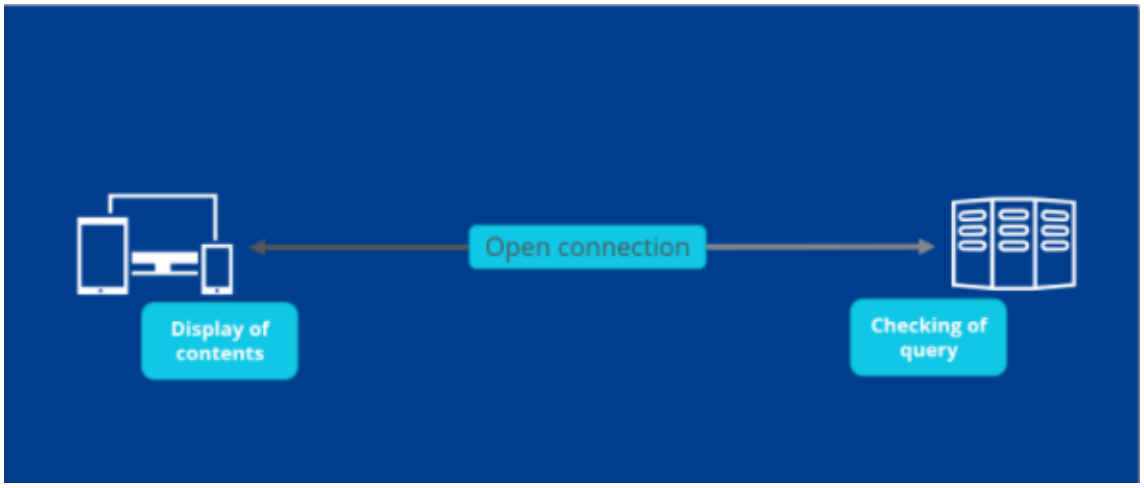


Figura 9. Protocolo WEBSOCKETS.

Websockets necesita que el navegador cliente sea compatible con el protocolo. Actualmente está implementado en los siguientes navegadores:

- Internet Explorer a partir de la versión 10.
- Firefox a partir de la versión 6.
- Chrome a partir de la versión 14.
- Opera a partir de la versión 12.10.
- Safari a partir de la versión 6.

Del lado del servidor, WebSocket puede implementarse con los siguientes lenguajes de programación y *frameworks*:

- Node.js
  - Socket.IO
  - WebSocket-Node
  - ws
- Java
  - Jetty
  - Ruby
  - EventMachine

- Python
  - pyWebSocket
  - Tornado
  - Flask-sockeIo
- C++
  - libWebSockets

2.3.6 Justificación del servidor web escogido

Se escoge el micro-framework Python-Flask debido a su sencilles y escalabilidad para proyectos embebidos. Este módulo cuenta con una gran soporte y documentación. Además del interés, por parte de la empresa, en el lenguaje de programación Python. Se utiliza su propio servidor para el desarrollo del proyecto. Para servidores en producción, Flask permite incrustar servidores como Nginx, que permiten certificados SSL.

El uso tradicional de las conexiones HTTP tiene el inconveniente de que el cliente siempre carga el HTML al completo. La tecnología **AJAX** (**A**synchronous, **J**avaScript, **A**nd **X**ML) soluciona esto, al permitir actualizar o refrescar solo una parte del HTML. AJAX solo permite la comunicación en una dirección, lo cual daría lugar a largos tiempos de espera en determinadas aplicaciones. WebSocket, en cambio, crea conexiones bidireccionales, lo que hace posible el contacto directo con el navegador permitiendo **cortos periodos de carga**. Por todo ello se utilizarán ambas tecnologías en función a los requisitos temporales del servidor.

2.4 Bases de datos

Hoy en día existen multitud de base de datos las cuales son fundamentales para el uso servidores web. A continuación, se describen los dos grandes grupos que sesgan las bases de datos según su modelo de datos.

2.4.1 Base de datos relacionales

SQL, son base de datos relacionales, escrita en el lenguaje de consulta estructurado SQL (Structured Query Language). Disponen de una relación predefinida entre sus elementos, donde cada registro pueda ser identificado de forma inequívoca.

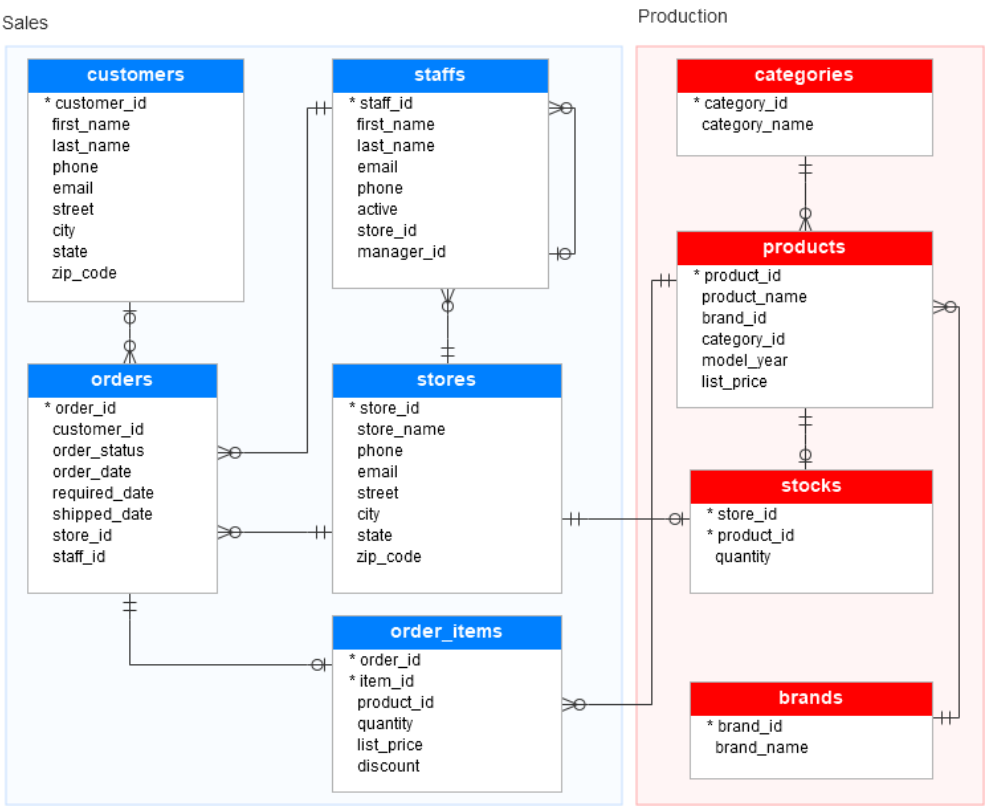


Figura 10. Estructura de base de datos SQL.

Las bases de datos SQL disponen de una serie de ventajas que las han convertido en el tipo de base de datos más utilizada. Las principales características son:

- Los datos se organizan en forma de tablas. Cada tabla tiene un conjunto de atributos o columnas, y cada fila, también conocida como tupla, puede tener una relación.
- Evita la duplicidad de registros, garantizando la integridad referencial (al eliminarse un registro, se eliminan todos los registros relacionados dependientes del mismo).
- Tiene un gran soporte debido a su extensión en el mercado.
- Atomicidad de la información. Al realizar cualquier operación en la base de datos, si surge algún problema, la operación no se realiza.
- Dispone de un sistema estándar bien definido (SQL) para las operaciones con la base de datos, como inserción, actualización o consultas.

Las bases de datos requieren de software de gestión de datos (RDBMS) para interactuar con ellas mediante operaciones simples. Los siguientes RDBMS son los valorados para este proyecto:

- **PostgreSQL**. Este RDBMS de código abierto es gratuito. Se basa en una arquitectura cliente- servidor para administrar las conexiones. Ofrece una extensibilidad notable. Puede usarlo junto con otros RDBMS populares como

Oracle y MySQL o bases de datos NoSQL populares como MongoDB. proporciona funciones de seguridad a nivel de aplicación, base de datos, entorno y usuarios. Soporta bien la concurrencia. Además, ofrece otras características poderosas como transacciones anidadas.

- **MySQL**. Es un proyecto de código abierto, propiedad de Oracle, conllevan un coste si tus aplicaciones son comerciales. Requiere de un servidor para ejecutarse que administre las conexiones de distintos usuarios mediante una arquitectura cliente-servidor. Su configuración es compleja, pero permite autenticación con un nombre de usuario, contraseña y SSH. Soporta gran cantidad de tipos de datos. Es óptima para grandes bases de datos con acceso a múltiples usuarios o proyectos que requieran más escalabilidad.
- **SQLite**. Se trata de un proyecto de código abierto para base de datos integradas en una aplicación debido a su fácil configuración. La biblioteca SQLite tiene un tamaño aproximado 250 KB y la información se almacena en un solo archivo. SQLite lee y escribe directamente en archivos de disco ordinarios. Cuenta con librerías de acceso para diferentes lenguajes de programación. Soporta texto en formato UTF-8 y UTF-16, así como datos numéricos de 64 bits. Soporta funciones SQL definidas por el usuario (UDF).

2.4.2 Base de datos no relacionales

NoSQL o *Not Only SQL* son bases de datos **no relacional** que no cuenta con un identificador que relacione un conjunto de datos con otro. En las bases de datos No SQL la información es organizada generalmente como documentos y no requieren que los datos estén estructurados para poder manipularlos. Predominando los tipos de organización de la siguiente imagen.

Figura 11. Tipos de bases de datos NoSQL.

Las principales ventajas de las bases de datos NoSQL son:

- Son bases de datos versátiles que permiten agregar información o hacer cambios en el sistema sin necesidad de agregar configuraciones extras.
- Es más fácil su expansión debido al escalado horizontal, es decir, al soportar estructuras distribuidas se pueden instalar nuevos nodos operativos que balancean la carga de trabajo.
- Permiten guardar datos de cualquier tipo**,** en cualquier momento, sin requerir una verificación.

- Realizan consultas utilizando JSON (JavaScript *Object Notation*, formato sencillo de intercambio de texto).

Alguno de los tipos de base de datos NoSQL más utilizados en la actualidad son los siguientes:

- **Cassandra**. Se trata de una base de datos creada por Apache del tipo clave–valor. Dispone de un lenguaje propio para realizar consultas CQL (Cassandra Query Language). Cassandra es una aplicación Java por lo que puede correr en cualquier plataforma que cuente con máquinas virtuales de java. Está orientada para base de dato que requiera un mayor número de escritura que de lectura, siendo óptimas para guardar datos de manera flexible.
- **Redis**. Se trata de una base de datos de código abierto. Su estructura de datos se basa en clave–valor. Tiene una arquitectura cliente-servidor con soporte de datos para cadenas, hashes, conjuntos de datos o listas. Sus operaciones son atómicas y persistentes. Redis no permite realizar consultas, sólo se puede insertar y obtener datos. Es una opción muy habitual para aplicaciones de almacenamiento en cache.
- **MongoDB**. También basada en una arquitectura cliente-servidor de código abierto. Su estructura de datos está orientada a documentos, de esquema libre, es decir, que cada entrada puede tener un esquema de datos diferente que nada tenga que ver con el resto de los registros almacenados. Para el almacenamiento de la información, utiliza un sistema propio de documento conocido con el nombre BSON, que es una evolución del conocido JSON, pero con la peculiaridad de que puede almacenar datos binarios.

2.4.3 Justificación de la base de datos escogida

Se decanta por las bases de dato relacionales debido a la estandarización del lenguaje SQL y por recomendación de la empresa. Se escoge una base de datos SqLite para este proyecto debido a su facilidad de uso, para integrarse con python-flask, además de su ligereza y velocidad óptimas para pequeñas aplicaciones con poca cantidad de datos.

CAPÍTULO 3 DISEÑO DEL SISTEMA

En este capítulo se presenta la arquitectura y deseño escogidas para el proyecto. Así como las funcionalidades de este.

3.1 Prototipo hardware

Se desarrolla un prototipo de sistema sobre la placa OSD32MP1-RED. Este controla y monitoriza tanto los botones (*Users Bottons*) como los leds (*Users LEDs*) que tiene integrados. Además, se añade un sensor AM2320 para monitorizar temperatura y humedad a través del bus i2c-5.

Para la conexión wifi, de la placa de evaluación, se requiere de una antena con conector U.FL. Para este prototipo se utiliza la antena W24P-U de Inventek.

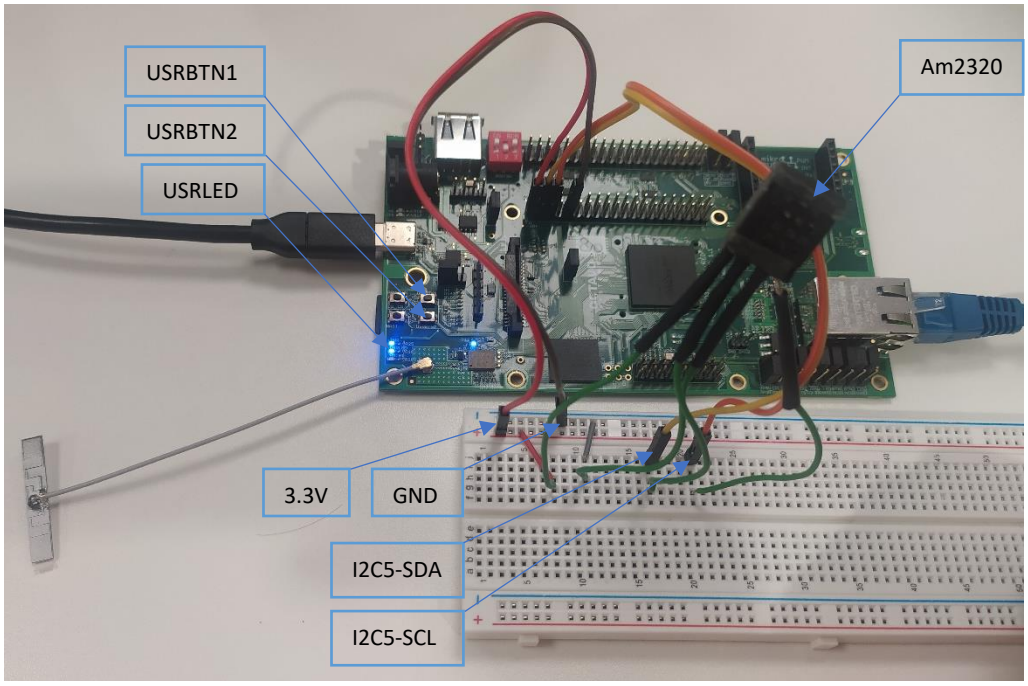


Figura 12.Prototipo hardware.

3.1.1 Sensor am2320

Se trata de un sensor de temperatura y humedad del fabricante *Aosong Electronics. Cuenta con las siguientes características:*

- Voltaje de funcionamiento: 3.1-5.5V.
- Rango de medición de humedad: 0-99.9%.
- Error de medición de humedad: ±3%.
- Rango de medición de temperatura: -80 a 40 °C
- Error de medición de temperatura: ±0.5 °C.
- Interfaz I2C en la dirección 0x5C.

- Interfaz de 4 pines: VCC, datos serie SDA, puerto bidireccional, GND, reloj serie SCL.
- Dimensiones: 21x23mm.

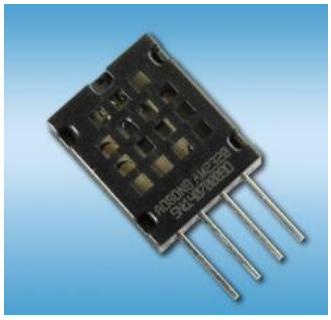


Figura 13. Sensor am2320.

3.2 Prototipo Software

El sistema cuenta con la siguiente arquitectura dual de servidores para cumplir los objetivos del proyecto:

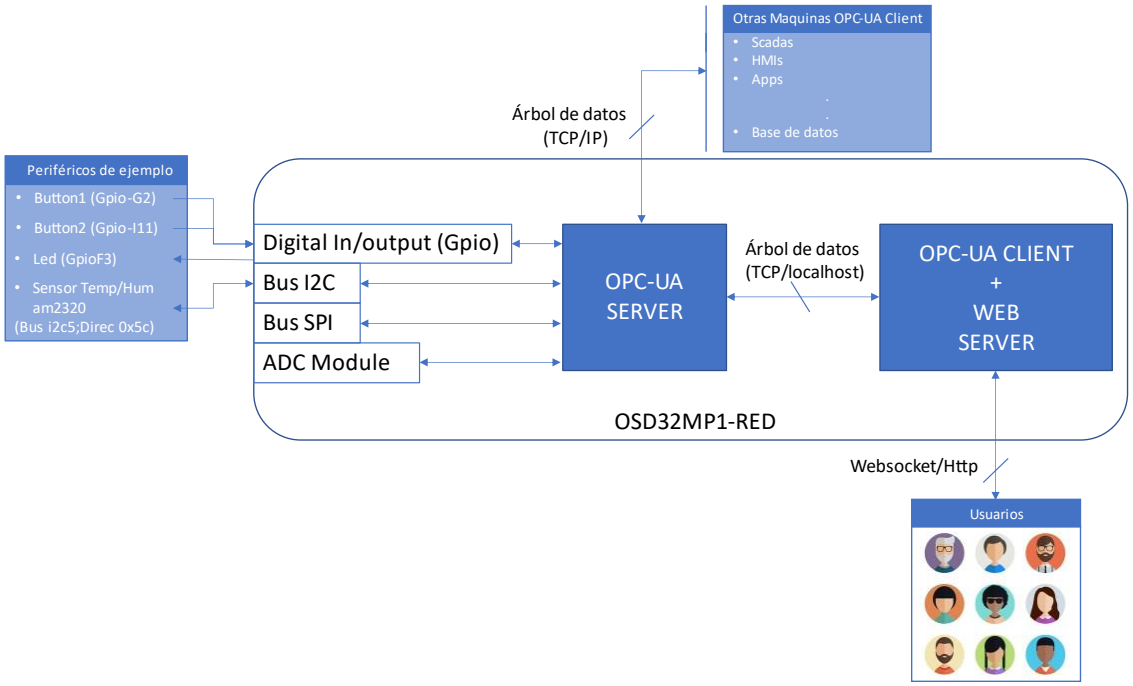
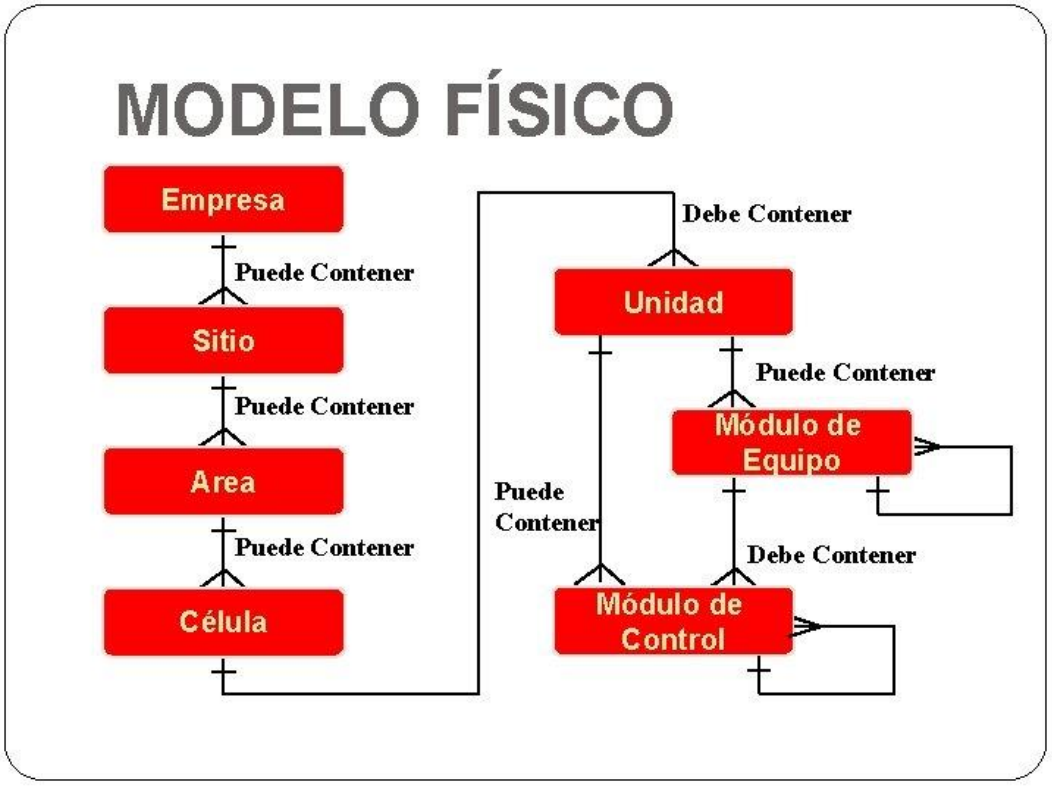


Figura 14. Arquitectura software

3.2.1 Modelo Físico ISA-88

Es modelo permite organizar de forma jerárquica los activos físicos de una empresa dedicada a la fabricación de productos por lotes. El modelo cuenta con siete niveles. Los tres niveles superiores se definen simplemente para identificar correctamente la relación de los equipos de nivel inferior con la empresa de fabricación y no son descritos

en detalle por el modelo ISA-S88. Los cuatro niveles inferiores (células de proceso, unidades, módulos de equipos y módulos de control) se definen por las actividades de ingeniería y los procesos a llevar a cabo.



Una célula de proceso contiene todas las unidades, módulos de equipamiento y módulos de control necesarios para fabricar uno o varios lotes
Una célula de proceso es una agrupación lógica de equipo que incluye el equipo requerido para la producción de uno o más lotes.

Una unidad se compone de módulos de equipamiento y módulos de control. Combina todos los equipos de procesamiento y control físico necesarios para llevar a cabo esas actividades como un equipo independiente. Una o varias de las principales actividades de procesamiento: - como reaccionar, cristalizar, y crear una disolución - pueden llevarse a cabo en una unidad.

Los módulos de equipamiento combinan todos los equipos de procesamiento y control físico necesarios para llevar a cabo dichas actividades. Estos módulos puede ser parte de una unidad o un equipo independiente dentro de una célula proceso. Pueden llevar a cabo un número finito de actividades de procesamiento menores, tales como la dosificación, extracción.

Un módulo de control es típicamente una colección de sensores, actuadores y los equipos de procesamiento asociado que, desde el punto de vista de control, es operado como una sola entidad. Un módulo de control también puede estar formado por otros módulos de control. Algunos ejemplos de módulos de control son:

- Un dispositivo que consiste en una válvula de bloqueo automático ON/OFF, está realimentado con interruptores de posición y que es operado a través de una consigna.
- Cilindro automático de doble efecto con dos finales de carreara para indicar su posición de avance o retroceso.
- Un único sensor o actuador.

3.2.2 Descripción del Servidor OPCUA

OPC-UA permite diseñar un servidor que exporte los valores de los periféricos conectados al stm32mp1 de forma estándar. Permite que cualquier máquina que esté en la misma red y tenga un cliente OPC-UA pueda acceder a dichos datos de forma segura y fácil.

OPC-UA se basa en una estructura de nodos ID estos permite tener características como descripción, nombre, rango, Tipo de nodo, etc. Dichos nodos se pueden relacionar entre sí, pueden existir varios nodos padres con varios nodos hijos y estos, a su vez, tener otros nodos que lo componen, creando así una estructura en árbol, de los datos.

Siguiendo el modelo físico de ISA-88 el árbol de datos del servidor quedaría de la siguiente forma:

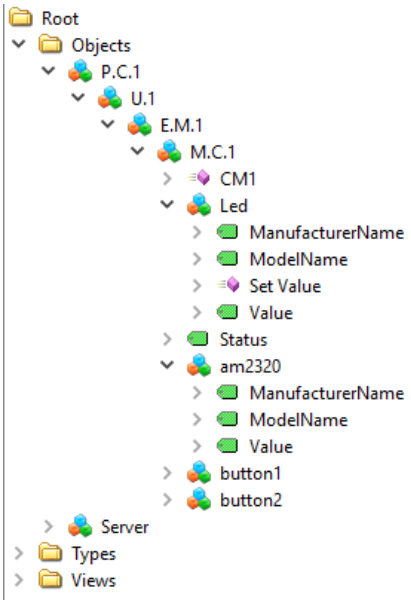


Figura 15. Árbol de datos del servidor OPC-UA

Todo servidor OPC-UA importa el nodo Root como nodo padre de todos los nodos del servidor. Los nodos Types y view son importados por defecto en la configuración. Type contiene todos los tipos de nodos existentes en el servidor, mientras que views puede ser configurado para mostrar determinadas vistas de nodos, en nuestro caso no contendrá ninguna. El nodo objects contiene todos objectos que componen el servidor.

El objeto Server se exporta por defecto en la configuración. El objeto P.C.1 (Célula de proceso) conforma el modelo de datos basado en ISA-88. En un servidor OPCUA podemos diferenciar entre:

- Objetos (Célula de proceso, Unidad, Modulo de equipamiento, Módulo de control, button1, etc).
- Atributos (ManufactureName, Value, ModelName), se trata de las características de cada objeto.
- Métodos (Set Value), son funciones con posibles valores de entra y salida que tienen el servidor.

Las características de nuestro árbol de datos son las siguientes:

- Cada célula de proceso tendrá un nombre, una descripción y el número de unidades que la forman.
- Cada unidad tendrá un nombre, una descripción y el número de módulos de equipamiento que la forman.
- Cada módulo de equipamiento tendrá un nombre, una descripción y el número de módulos de control que lo forman.
- Cada módulo de control tendrá un nombre, una descripción, los sensores y actuadores que lo componen. También se añade una variable Status de tipo string que podrá indicar el estado en él se encuentra el módulo, (en el ejemplo del cilindro automático la variable status podría ser: en avance o en retroceso). Además, podrá contener un método para ejecutar una acción de control sobre sus dispositivos y variable Status.
- Los sensores y actuadores tienen tres variables comunes: ManufactureName, indica el fabricante del dispositivo mediante una variable string, ModelName, hace referencia al nombre exacto que proporciona el fabricante, también se trata de un tipo string; Value, corresponde con el valor de la lectura del dispositivo, puede ser de cualquier tipo de dato incluido arrays para aquellos dispositivos con más de un dato a exportar o importar. Los actuadores incluyen un método llamado, Set Value para modificar la variable Value de forma segura. El nombre de los objetos sensores y actuadores corresponden al roll de estos.

Por lo tanto, un cliente OPC-UA puede monitorizar el valor de un periférico haciendo referencia al atributo *value* del objeto que lo define.

3.2.3 Descripción del Servidor Web

La principal característica del servidor desarrollado es la modularidad e independencia que tiene hacia el servidor OPC-UA, puesto que permite conectarse a cualquier servidor OPC-UA sin llevar a cabo ningún cambio en su código fuente. Manteniendo así la arquitectura general de servidores. Además, dicha modularidad permite ser instalado en hardware diferentes del servidor OPC-UA o sobre el mismo, ambos corriendo en paralelo.

Se trata de un servidor **dedicado** o **local**, cuyo **backend** se basa en el microframework Python-flask.

El **frontend** se basa en HTML para las plantillas de las vistas. CSS para el estilo y diseño de los objetos que conforman las plantillas. JavaScript para llevar a cabo funciones y solicitudes complejas por parte del cliente.

Por tanto, se crea una aplicación web **dinámica** basada en HTTP para peticiones de archivos HTML, CSS y JS. Para las funciones donde se precise un procesado por parte del cliente web se utilizar AJAX. Mediante Websocket se abre un canal de comunicación que permita la monitorización de los nodos del servidor OPCUA, la solicitud de métodos de este y mostrar avisos y acontecimientos emitidos por el servidor web.

Para la conexión con el servidor OPC-UA se crea su respectivo cliente, integrado en el servidor web.

El servidor web cuenta con una base de datos SQLite para el registro de usuarios y la gestión de nodos del servidor OPC-UA.

3.2.3.1 Aplicación web

Desde un navegador un usuario puede acceder a la aplicación web mediante la url:http://192.168.0.101/, tratándose de la IP fija con la que se configura el servidor web (esta depende, realmente, de las IPs libres que tenga el modem).

La aplicación web cuenta con las siguientes vistas:

- Home. Se trata de la vista inicial de la aplicación web. Contiene una breve descripción del servidor.
- Login. Esta vista permite iniciar sesión a los usuarios. Diferenciando entre dos tipos. El usuario convencional podrá monitorizar datos y llamar a métodos del servidor OPC-UA, además de editar su perfil. Por otro lado, el usuario administrador obtiene permisos para editar la base de datos.
- SignUp. Permite registrar nuevos usuarios en el servidor. No permite registrar usuarios con el mismo email.
- Init. Se trata de la primera vista tras autenticarse y es meramente informativa. Su intención es mostrar información de la ubicación del servidor, características de este, etc.
- Monitored Item. Monitorización de nodos variables y llamada a métodos del servidor OPC-UA. Gestiona e interpreta los datos recibidos por el canal.
- Profile. Esta vista permite a los usuarios modificar su nombre, email y contraseña.
- Users. Vista solo para usuarios administradores. Muestra los usuarios almacenados en la base de datos. Además, permite editar parámetros como el nombre, el email y el tipo de acceso del usuario.

- Opc-Ua Node. Vista solo para usuarios administradores. Muestra todos los nodos almacenados en la base de datos por el cliente OPC-UA.
- UploadOpcUaServer. Vista solo para usuarios administradores. Permite cargar una actualización del binario del servidor OPC-UA.

CAPÍTULO 4 ANALISIS Y DESARROLLO

En este capítulo se profundiza en las características y requisitos del sistema desarrollado para el proyecto.

4.1 Requisitos

Los requisitos de desarrollo son todas aquellas herramientas software que han sido necesarias desplegar para el sistema. Estas abarcan desde la configuración de pines del microprocesador, la creación de una distribución de Linux propia, hasta la instalación de librerías Python.

Se parte de un entorno de desarrollo creado a partir de la documentación de Gradiant. Este entorno es una máquina virtual con Linux basado en la distribución 20.04 de Ubuntu. Mediante el paquete de distribución de STM32MP1 se crea la propia distribución para la placa de evaluación partiendo de OpenStLinux.

4.1.1 STM32CubeIDE

Para el desarrollo de aplicaciones tanto para los Cores Cortex A7 como para el Cortex M4, el fabricante STMicroelectronics proporciona STM32CubeIDE. Se una plataforma de desarrollo C/C++ avanzada con funciones de configuración de periféricos, generación y compilación de código, así como la depuración para microcontroladores y microprocesadores STM32. Se basa en el marco Eclipse ® /CDT™ y la cadena de herramientas GCC para el desarrollo y GDB para la depuración. Permite la integración de plugins que aumentan su funcionalidad.

4.1.2 Librerías Open62541

La librería Open62541 es una implementación de código abierto y libre de OPC-UA (Arquitectura unificada OPC) escrita en C99 y C++ 98. La librería se puede utilizar con todos los compiladores principales y proporciona las herramientas necesarias para implementar clientes y servidores OPC UA dedicados, o para integrar la comunicación basada en OPC-UA en aplicaciones existentes. 0pen62541 es independiente de la plataforma. Por ello y por recomendación de la comunidad stm32mp1 será la librería utilizada. Se lleva a cabo la instalación de su SDK con la que se implanta el protocolo OPC-UA desde Stm32CubeIde. Se minimiza el tamaño del binario que implementa el servidor ajustando la configuración de compilación del SDK. Con open2541, es posible configurar servidores mínimos que requieren menos de 100kB de RAM y ROM.

La construcción de la librería open62541 junto a su integración en STM32CubeIde queda documentada en el manual de desarrollo del servidor OPC-UA.

4.1.3 Visual Studio Code

Se desarrolla un eterno virtual de programación mediante la aplicación *Visual Studio Code* para programar el servidor web y posteriormente ser embebido en la placa de evaluación.

Se trata de un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código de diferentes lenguajes. Esta aplicación nos permitirá editar y programar los diferentes archivos que maneja el servidor, HTML, CSS, JavaScript y Python.

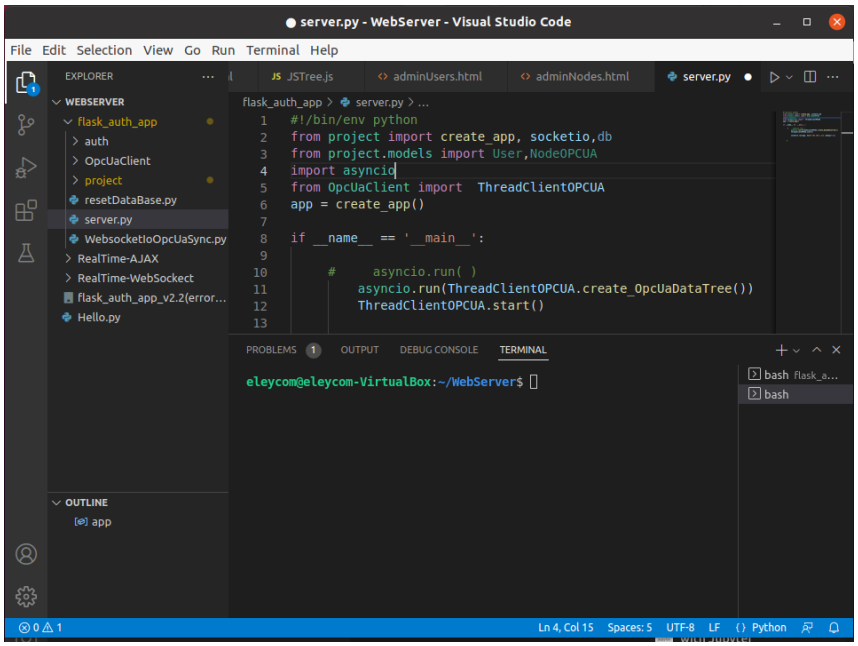


Figura 16. Visual Code.

4.1.4 Módulos externos para Python-Flask

Este servidor web se ha desarrollado con los siguientes módulos:

- Flask (v_1.1.4). Se trata de la última versión que soporta el paquete de distribución de STM32MP1, basado en Yocto Project. Con él se crea la estructura base del servidor web.
- Flask-SocketIO (v_5.5.1) que requiere de python-socketio (v_5.5.2) y python-engineio (v_4.3.1). Este módulo brinda a las aplicaciones Flask acceso a comunicaciones bidireccionales de baja latencia entre los clientes y el servidor. En la mayoría de los casos, la conexión se establecerá con WebSocket, en los casos donde no sea posible, se recurrirá al sondeo largo de HTTP. Cuando se pierde la conexión, el cliente intentará volver a conectarse automáticamente.
- Flask-SQLAlchemy (v_2.5.1). Agrega soporte para SQLAlchemy a la aplicación Flask. SQLAlchemy es un ORM (Object Relational Mapper) el cual permite

manipular las tablas de una base de datos como si fueran objetos de nuestro programa.

- Flask-Login (v_0.5.0). Este módulo permite administrar las sesiones de usuarios tras la autenticación. Se usa el módulo SQLite3 para evitar tener que instalar dependencias adicionales para la base de datos.
- Eventlet (v_0.33). Es un módulo de red concurrente para Python que le permite cambiar la forma en que ejecuta su código, no cómo lo escribe. Es requerido por Flask-SocketIO y requiere del módulo greenlet (v_1.1.2).
- Asyncua (v_0.9.92). Se trata del módulo que implementa el cliente necesario para la conexión con el servidor OPC-UA.

Para el desarrollo del servidor web se crea un entorno virtual que permita instalar módulos determinados de Python evitando problemas con las versiones de los mismo. Por ello se requiere también del módulo virtualenv. Dicha virtualización se crea sobre un terminal de virtual Studio, en nuestro entorno de desarrollo Ubuntu.

4.2 Configuración de pines y generación de Device Tree

El plugins STM32CubeMX de STM32CubeIDE configura los pines del microprocesador STM32MP157C permitiendo redefinirlos para distintos diseños de hardware. Dicha configuración de pines generará el Device Tree necesario para Linux. Para el SOM utilizado se parte del proyecto minimalista, ***OSD32MP157C-512M-BAA_MinimalConfig*** que ofrece el fabricante para la versión 5.10 de Linux. Dicho proyecto corresponde a la siguiente figura.

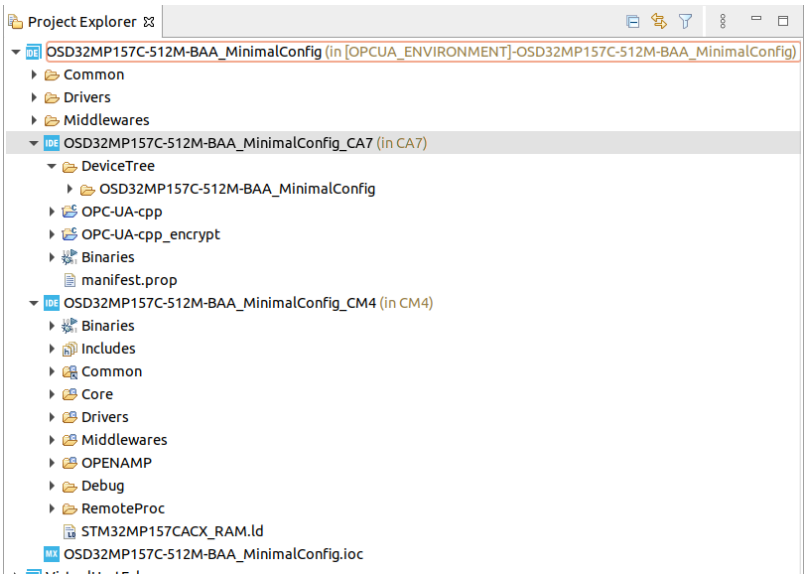


Figura 17. Proyecto OSD32MP157C-512M-BAA_MinimalConfig.

Este proyecto minimalista no cuenta con la configuración de los pines encargados de comunicarse con los módulos wifi/ bluetooth, ethernet y la interfaz de cámara. Para la

correcta configuración se basa el Device Tree para Linux 4.19[3] aportado por OctavoSystem.

Para añadir la configuración de pines se accede al archivo *OSD32MP157C-512M-BAA_MinimalConfig.ioc. Este es ejecutado automáticamente, STM32CubeMX, abriendo el siguiente IDE.*

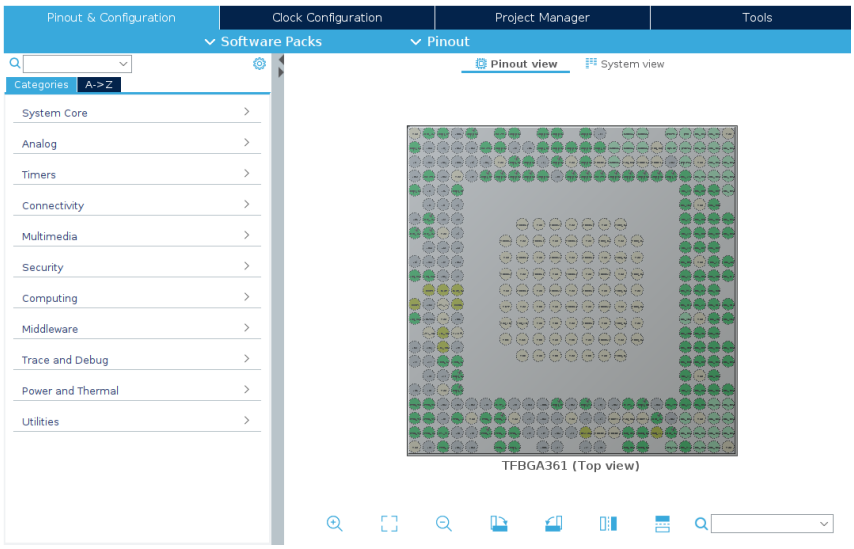


Figura 18. Plugin STM32CubeMX para la configuración de pines.

La configuración con el módulo ethernet, se lleva a cabo sobre la pestaña *connectivity>ETH1*. Se escoge el modo RGMII[4] que configura los siguientes pines siguiendo el esquema[5] de la placa de evaluación:

[3] Esta versión no es compatible con las herramientas de STM32CubeIDE.
[4] RGMII. Interfaz independiente de medios reducida, Se trata de un estándar industrial para conectar las capas MAC y PHY. Permite alcanzar los 1000 Mbps.
[5] Esquemas y planos de la placa de evaluación en el documento *osd32mp15x-red-schematic-pdf*

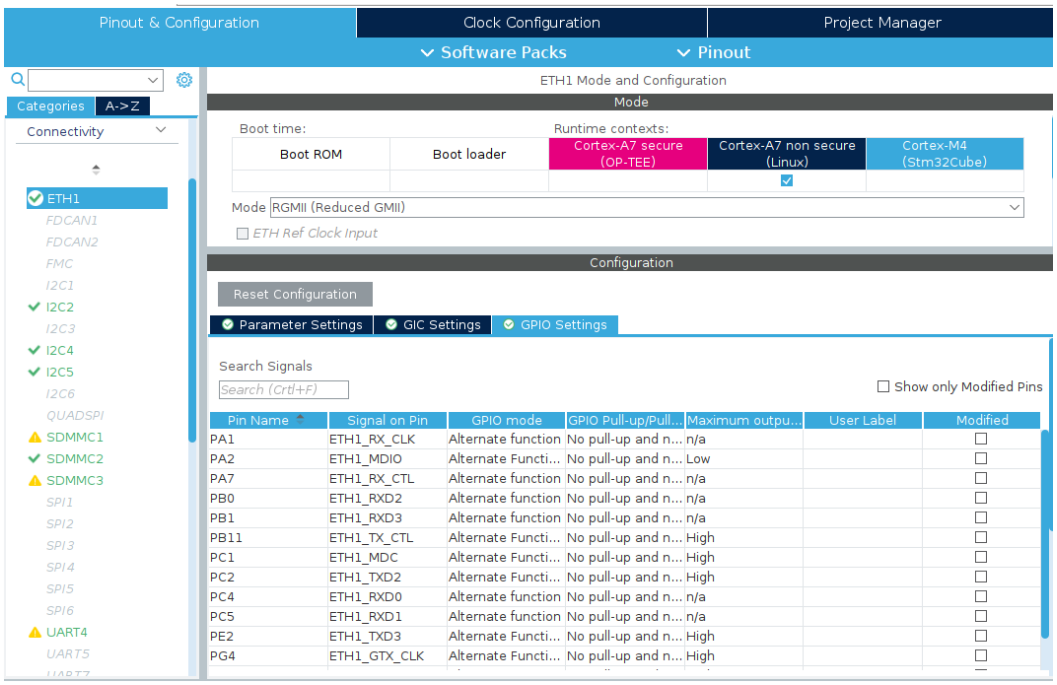


Figura 19. Configuración de pines para Ethernet.

El **runtime context** hace referencia a los núcleos que pueden controlar estos periféricos. Aquí puede verse la independencia entre los dos núcleos A7, que implementan Linux y el M4 encargado de las tareas en tiempo real**. Boot loader** indica si es necesario incluirlo en sistema de arranque de la placa.

El módulo wifi/Bluetooth, de la placa de evaluación, queda conectado al puerto SDMM3[6] para los datos wifi y el puerto USART2 para la información bluetooth. Por lo que se implementa la siguiente configuración, tanto en el contexto de Linux como en el arranque seguro, Boot loader.

Estos microprocesadores cuentan con la interfaz de cámara conocida como DCMI[7] situada en *Multimedia>DCMI*. Además, se requiere de un puerto I2C para el control y configuración de la cámara. En este caso el adaptador de la placa de evaluación utiliza el I2C2.

Por último, la conexión con el sensor am2320 requiere de la configuración de un bus I2C. Se escoge el I2C-5 ya que tienes pines de salida en el cabeza de conexionado "Rasberry pi", en la placa de evaluación.

La siguiente tabla muestra la configuración de los pines mencionados hasta ahora, siguiendo la misma filosofía de la *Figura 19*.

[6] SDMM3. Bus de comunicación paralelo.

[7] DCMI. Bus de datos paralelo síncrono que admite hasta 14 líneas de datos. Permite la comunicación directa con módulos de cámara CMOS de 8/10/12/14 bits

Pin Name	Signal On Pin	GPIO Mode	GIPIO Pull-up/Pull-Down	Maximun output speed	User Label	Modified
PA1	ETH1_RX_CLK	Alternate function	No pull-up and no pull-down	n/a		false
PA2	ETH1_MDIO	Alternate Function Push Pull	No pull-up and no pull-down	Low		false
PA7	ETH1_RX_CTL	Alternate function	No pull-up and no pull-down	n/a		false
PB0	ETH1_RXD2	Alternate function	No pull-up and no pull-down	n/a		false
PB1	ETH1_RXD3	Alternate function	No pull-up and no pull-down	n/a		false
PB11	ETH1_TX_CTL	Alternate Function Push Pull	No pull-up and no pull-down	High		false
PC1	ETH1_MDC	Alternate Function Push Pull	No pull-up and no pull-down	High		false
PC2	ETH1_TXD2	Alternate Function Push Pull	No pull-up and no pull-down	High		false
PC4	ETH1_RXD0	Alternate function	No pull-up and no pull-down	n/a		false
PC5	ETH1_RXD1	Alternate function	No pull-up and no pull-down	n/a		false
PE2	ETH1_TXD3	Alternate Function Push Pull	No pull-up and no pull-down	High		false
PG4	ETH1_GTX_CLK	Alternate Function Push Pull	No pull-up and no pull-down	High		false
PG13	ETH1_TXD0	Alternate Function Push Pull	No pull-up and no pull-down	High		false
PD3	USART2_CTS	Alternate function	No pull-up and no pull-down	n/a		false
PD4	USART2_RTS	Alternate Function Push Pull	No pull-up and no pull-down	Low		false
PD5	USART2_TX	Alternate Function Push Pull	No pull-up and no pull-down	Low		false
PD6	USART2_RX	Alternate function	No pull-up and no pull-down	n/a		false
PD3	USART2_CTS	Alternate function	No pull-up and no pull-down	n/a		false
PD7	SDMMC3_D3	Alternate Function Push Pull	No pull-up and no pull-down	Medium		true
PF0	SDMMC3_D0	Alternate Function Push Pull	No pull-up and no pull-down	Medium		true
PF1	SDMMC3_CMD	Alternate Function Push Pull	No pull-up and no pull-down	Medium		true
PF4	SDMMC3_D1	Alternate Function Push Pull	No pull-up and no pull-down	Medium		true
PF5	SDMMC3_D2	Alternate Function Push Pull	No pull-up and no pull-down	Medium		true
PG15	SDMMC3_CK	Alternate Function Push Pull	No pull-up and no pull-down	High		true
PA4	DCMI_HSYNC	Alternate function	No pull-up and no pull-down	n/a		false
PA6	DCMI_PIXCLK	Alternate function	No pull-up and no pull-down	n/a		false
PA10	DCMI_D1	Alternate function	No pull-up and no pull-down	n/a		false
PB9	DCMI_D7	Alternate function	No pull-up and no pull-down	n/a		false
PC6	DCMI_D0	Alternate function	No pull-up and no pull-down	n/a		false
PE0	DCMI_D2	Alternate function	No pull-up and no pull-down	n/a		false
PE1	DCMI_D3	Alternate function	No pull-up and no pull-down	n/a		false
PE4	DCMI_D4	Alternate function	No pull-up and no pull-down	n/a		false

PE13	DCMI_D6	Alternate function	No pull-up and no pull-down	n/a		false
PG9	DCMI_VSYNC	Alternate function	No pull-up and no pull-down	n/a		false
PI4	DCMI_D5	Alternate function	No pull-up and no pull-down	n/a		false
PH5	I2C2_SDA	Alternate Function Open Drain	No pull-up and no pull-down	Low		false
PZ6	I2C2_SCL	Alternate Function Open Drain	No pull-up and no pull-down	Low		false
PA11	I2C5_SCL	Alternate Function Open Drain	Pull-up	Low		true
PA12	I2C5_SDA	Alternate Function Open Drain	Pull-up	Low		true

Figura 20. Configuración de pines para Ethernet, modulo Wifi/Bluetooth, cámara y bus I2C (sensor am2320).

Tras configurar los pines es necesario determinar el reloj y los tiempos de ejecución de cada uno de los periféricos. Se puede llevar a cabo desde el IDE *CLOCK CONFIGURATION.*

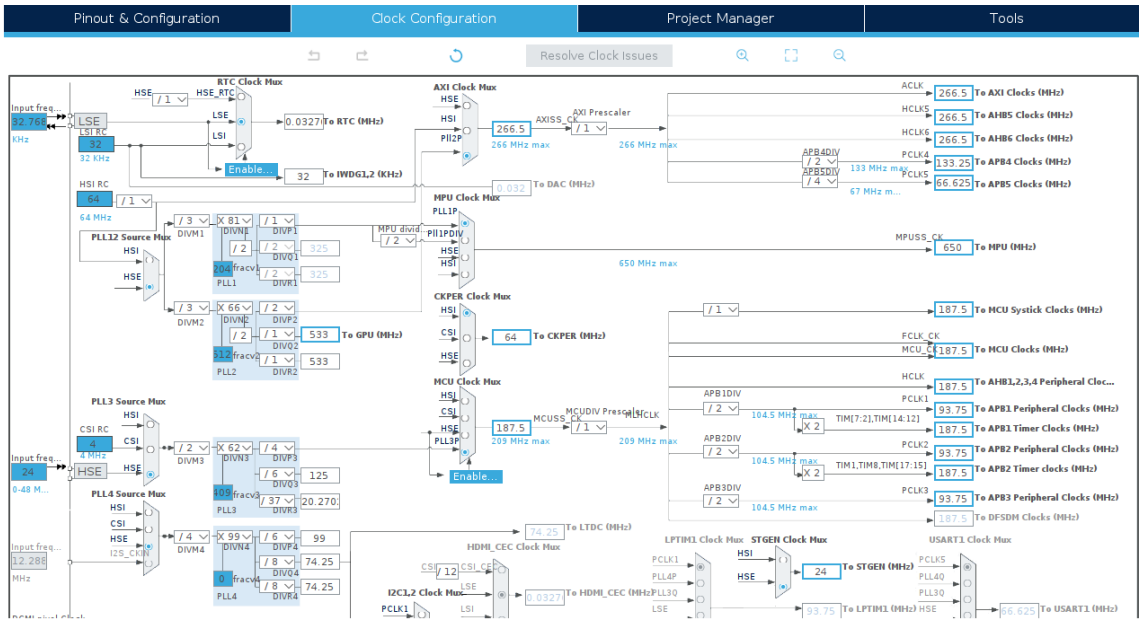


Figura 21. Configuración de reloj SMT32CubeMX.

Se trata de una ardua tarea que en caso de no precisar correctamente puede provocar la inhabilitación de los periféricos o errores en la trama de datos. En concreto ethernet debe trabajar a 125MHz. Para ello se escoge la división PLL3 de reloj HSE (24MHz), se le aplica ciertos múltiplos y divisores para obtener los 125MHz en la salida PLLQ3, la cual comandará el RGMII.

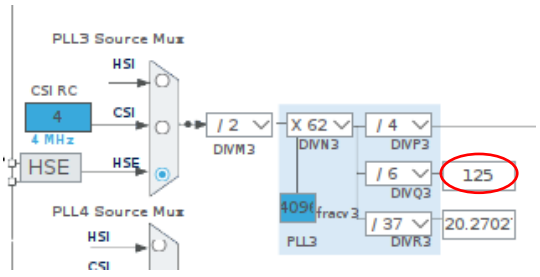


Figura 22. Parámetros para el reloj de ethernet.

En cuanto al puerto sdmm3 encargado del módulo wifi, el fabricante indica que debe trabajar a 99Mhz. Siguiendo la misma filosofía de ethernet se le asigna el PLL4P.

Es importante controlar los múltiplos y divisores implicados de tal forma que no modifique el resto de sus relojes vecinos (PLL3R, PLL3P, PLL4Q, etc).

Una vez configurado los pines se indica al IDE que genere el código necesario para el Device tree, actualizando los siguientes archivos.

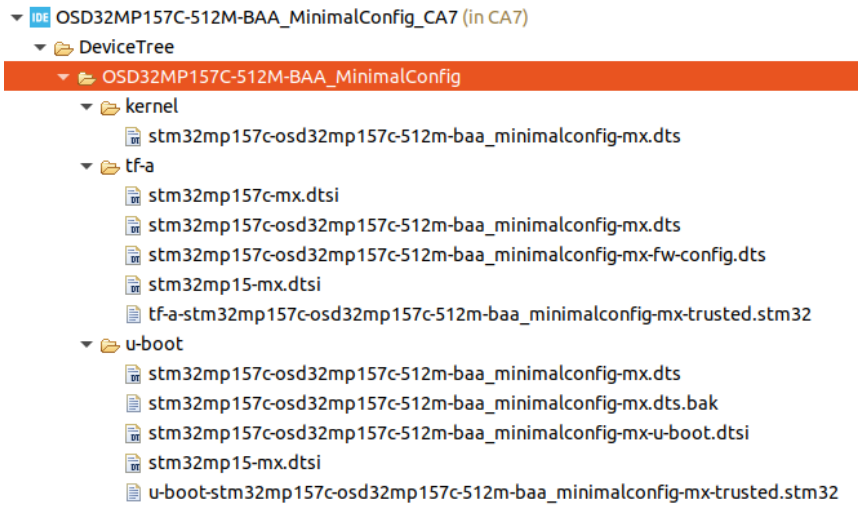


Figura 23. Componentes del Device Tree.

- **Tf-a/stm32mp1c-osd32mo157m-baa_minimalconfig-mx-fw-config.dts**. Se trata de la pieza de arranque seguro. En ella queda implementada la configuración de los relojes del sistema.
- **u-boot/stm32mp1c-osd32mo157m-baa_minimalconfig-mx.dts**. Encargado de iniciar los periféricos necesarios y de transferir el control al kernel de Linux de forma ordenada. Implementa la configuración de los pines para el arranque desde el M4 antes de transferírselos al kernel del sistema operativos
- **Kernel/stm32mp1c-osd32mo157m-baa_minimalconfig-mx.dts**. En él se implementa la configuración de los pines, pero para el sistema operativo Linux.

4.3 Insertar drivers en el device tree

Los archivos .dts del device tree se escriben siguiendo una estructura donde se diferencia tres partes:

- En el inicio se incluyen los archivos **.dtsi** que heredará y se definen variables a utilizar.

```
#include <dt-bindings/pinctrl/stm32-pinfunc.h>

#include "stm32mp157.dtsi"
#include "stm32mp15xc.dtsi"
```

```
#include "stm32mp15xxac-pinctrl.dtsi"
#include "stm32mp15-m4-srm.dtsi"

/* USER CODE BEGIN includes */
#include <dt-bindings/mfd/st,stpmic1.h>
#include <dt-bindings/rtc/rtc-stm32.h>
/* USER CODE END includes */

/ {
     model = "STMicroelectronics custom STM32CubeMX board";
     compatible = "st,stm32mp157c-osd32mp157c-512m-baa_minimalc
onfig-mx", "st,stm32mp157";
```

- Posteriormente se relaciona los pines al nodo (representación de los periféricos) que los utilice. Esto es el código generado por STM32CubeMX coincidiendo con los parámetros dado en la tabla x. Estos nodos pueden ser heredados y rescritos de nuevo.

```
usart2_pins_mx: usart2_mx-0 {
     u-boot,dm-pre-reloc;
     pins1 {
     u-boot,dm-pre-reloc;
     pinmux = <STM32_PINMUX('D', 3, AF7)>, /* USART2_CTS */
     <STM32_PINMUX('D', 6, AF7)>; /* USART2_RX */
     bias-disable;
     };
pins2 {
     u-boot,dm-pre-reloc;
     pinmux = <STM32_PINMUX('D', 4, AF7)>, /* USART2_RTS */
     <STM32_PINMUX('D', 5, AF7)>; /* USART2_TX */
     bias-disable;
     drive-push-pull;
     slew-rate = <0>;
            };
     };

usart2_sleep_pins_mx: usart2_sleep_mx-0 {
     u-boot,dm-pre-reloc;
     pins {
```

```
     u-boot,dm-pre-reloc;
     pinmux = <STM32_PINMUX('D', 3, ANALOG)>, /* USART2_CTS*/
 <STM32_PINMUX('D', 4, ANALOG)>, /* USART2_RTS */
             <STM32_PINMUX('D', 5, ANALOG)>, /* USART2_TX */
                    <STM32_PINMUX('D', 6, ANALOG)>; /* USART2_
RX */
             };
     };
```

- Por último, se relacionan los nodos que describen la configuración de pines con los SoC, que indica los registros y drivers de Linux a utilizar. Estos SoC son definidos en los archivos de microprocesadores inferiores, es decir, "*stm32mp157.dtsi/stm32mp153.dtsi/stm32mp151.dtsi*". Si son llamados posteriormente se redefinen.

```
&usart2{
            u-boot,dm-pre-reloc;
     pinctrl-names = "default", "sleep";
     pinctrl-0 = <&usart2_pins_mx>;
     pinctrl-1 = <&usart2_sleep_pins_mx>;
     status = "okay";

     /* USER CODE BEGIN usart2 */
     uart-has-rtscts;
     bluetooth {
            shutdown-gpios = <&gpioe 10 GPIO_ACTIVE_HIGH>;
            compatible = "brcm,bcm43438-bt";
            max-speed = <3000000>;
            };
     /* USER CODE END usart2 */
};
```

Esta placa de evaluación cuenta con el módulo LBEE5KL1DX-TEMP para la conexión wifi/bluetooth que hace uso del driver "brcm". Para la conexión de la cámara se utiliza el driver "OV5640" el cual corresponde con el módulo de cámaras que desea utilizar la empresa en proyectos futuros. La conexión ethernet requiere de los drivers "snps,dwmac-mdio".

4.4 Configuración del paquete de distribución

ST proporciona la distribución OpenSTLinux, basada en Yocto Project. Esta incluye el kernel del sistema operativo junto con el bootloader u-boot, los sistemas de arranque seguro y los paquetes necesarios para la construcción de un sistema operativo completo.

Para generar una imagen del contenido de la tarjeta SD para una placa nueva y distinta de las soportadas por defecto por las herramientas de ST es necesario añadir el nuevo device tree generado para la placa de evaluación y posteriormente compilarlo junto con el resto de los paquetes que componen el sistema operativo. Las tareas de compilación las realizará de forma automática la herramienta **bitbake** que forma parte del Yocto Project.

Se requiere iniciar sesión en GitHub para utilizar la herramienta **repo** que inicializa el esqueleto del proyecto de STM.

```
$ git config --global user.email "<email>"
$ git config --global user.name "<Nombre de usuario>"
```

En este caso se opta por la versión *openstlinux-5.10-dunfell-mp1-21-03-31* la última versión en el momento de instalarse. El comando Repo sync permite descargar los repositorios de los distintos proyectos. Colocándose en las ubicaciones adecuadas para formar el árbol de directorios que **bitbake** utilizará para construir la imagen.

```
$ repo init -u https://github.com/STMicroelectronics/oe-manifes
t.git -b refs/tags/openstlinux-5.10-dunfell-mp1-21-03-31
$ repo sync
```

Este proyecto cuenta con una imagen completa con un tamaño aproximado de 4.6GB, pero no corresponde para la placa de evolución utilizada. Por ello es necesaria identificar una nueva máquina (placa de evaluación), para poder insertar el device tree generado anteriormente.

Primero se crea la carpeta que contendrá el device tree:

```
$ mkdir -p openstlinux-5.10-dunfell-mp1-21-03-31$ cd layers/met
a-st/meta-st-stm32mp-addons/mx/
$ cd openstlinux-5.10-dunfell-mp1-21-03-31/layers/meta-st/meta-
st-stm32mp-addons/mx/
```

Posteriormente se copian los directorios que representan el Device Tree (T-FA, Kernel y U-boot) del proyecto OSD32MP157C-512M-BAA_MinimalConfig en el directorio recién creado.

Mediante un fichero de configuración .conf se le indica al proyecto Yocto la existencia de una nueva máquina para la que se pueden compilar imágenes. El fichero de configuración correspondiente se crea en el directorio **openstlinux-5.10-dunfell-mp1-21-03-31/layers/meta-st/meta-st-stm32mp-addons/conf.** Para ello se toma como base la plantilla **stm32mp1-mx.conf** que se puede encontrar en el mismo directorio.

Los cambios incluyen la configuración de arranque seguro usando TF-A, la creación de la imagen para la tarjeta SD y la configuración de la ruta en la que se encuentran los archivos del device tree generados, así como su nombre. Además, se incluye los drivers para el módulo wifi/bluetooth, requeridos por el espacio de usuario de linux.

```
#############################################################
#############
#
# User machine customization sections
#
#############################################################
#############

# Boot Scheme
# ============================================================
BOOTSCHEME_LABELS += "trusted"
#BOOTSCHEME_LABELS += "optee"
# WORKAROUND to generate U-BOOT SPL for DDR Tuning tools usage
UBOOT_CONFIG += "basic"
# Boot Device Choice
# ============================================================
# Define the boot device supported
#BOOTDEVICE_LABELS += "sdcard"
 BOOTDEVICE_LABELS += "sdcard"
#BOOTDEVICE_LABELS += "emmc"
#BOOTDEVICE_LABELS += "nand-4-256"
#BOOTDEVICE_LABELS += "nor-sdcard"

# Support Feature Choice
# ============================================================
# Define the features to enable on board
MACHINE_FEATURES += "bluetooth"
MACHINE_FEATURES += "wifi"

# Specific firmwares and kernel modules configuration
```

```
# ================================================================
# Set the list of kernel module to be auto-loaded during boot
#KERNEL_MODULE_AUTOLOAD += ""

# Set Bluetooth related package list needed when 'bluetooth' fe
ature is enabled
BLUETOOTH_LIST += "linux-firmware-bluetooth-bcm4343"
# Set Wifi related package list needed when 'wifi' feature is e
nabled
WIFI_LIST += "linux-firmware-bcm43430"
# CubeMX Project Config
# ================================================================
# Assign CubeMX Board devicetree and project path name
CUBEMX_DTB = "stm32mp157c-osd32mp157c-512m-baa_minimalconfig-mx
"
CUBEMX_PROJECT = "mx"
```

Por último, se crea un enlace simbólico al archivo **ST_EULA_SLA** que incluye las licencias para el uso de binarios específicos del microprocesador de ST.

```
$ openstlinux-5.10-dunfell-mp1-21-03-31/layers/meta-st/meta-st-
stm32mp-addons/conf/eula
$ ln -s ST_EULA_SLA stm32mp1-stm32mp157c-osd32mp157c-512m-baa_m
inimalconfig-mx
```

En este punto el sistema está completamente configurado para la creación de imágenes utilizando el device tree generado. Para compilar la nueva imagen se realiza indicado la variable de entorno **MACHINE** el identificador de la nueva placa que se ha creado.

```
$ cd openstlinux-5.10-dunfell-mp1-21-03-31/
$ MACHINE=stm32mp1-stm32mp157c-osd32mp157c-512m-baa_minimalconf
ig-mx DISTRO=openstlinux-weston source layers/meta-st/scripts/e
nvsetup.sh
$ bitbake st-image-core
```

La receta st-image-core que compila bitbake crea una imagen minimalista de Linux con las distintas particiones que permiten crear la tarjeta SD para la placa. Para la creación de una imagen completa se utiliza el script **create_sdcard_from_flashlayout.sh** junto con el archivo **FlashLayout_sdcard_stm32mp157c-osd32mp157c-512m-baa_minimalconfig-mx-trusted.tsv** que define la asignación de las distintas particiones con los archivos correspondientes.

```
$ cd openstlinux-5.10-dunfell-mp1-21-03-31/build-openstlinuxwes
ton-stm32mp1-stm32mp157c-osd32mp157c-512m-baa_minimalconfig-mx
/tmp-glibc/deploy/images/stm32mp1-stm32mp157c-osd32mp157c-512m
-baa_minimalconfig-mx/scripts/
$ ./create_sdcard_from_flashlayout.sh ../flashlayout_st-image-c
ore/trustedFlashLayout_sdcard_stm32mp157c-osd32mp157c-512m-baa
_minimalconfig-mx-trusted.tsv
```

Se obtiene el archivo **FlashLayout_sdcard_stm32mp157c-osd32mp157c-512m-baa_minimalconfig-mx-trusted.raw**. Este archivo contine el sistema operativo completo en menos de 1.6GB y puede ser grabado directamente en una tarjeta microSD.

4.5 Insertar paquetes de instalación en la Distribución

Para poder añadir a la distribución paquetes externos a OpenSTLinux se crea una nueva capa. Es necesario acceder al repositorio del proyecto y cargan las variables de entorno que definen el dispositivo para el que se va a compilar:

```
$ cd openstlinux-5.10-dunfell-mp1-21-03-31/
$ MACHINE=stm32mp1-stm32mp157c-osd32mp157c-512m-baa_minimalconf
ig-mx DISTRO=openstlinux-weston source layers/meta-st/scripts/e
nvsetup.sh
```

Posteriormente se crea la nueva capa utilizando **bitbake**. Indicando los parámetros de prioridad y la ruta donde crearla. En este caso se recomienda crear subcapas dentro de la carpeta **layers/meta-st/** para mantener la organización recomendada por la comunidad de ST. Además, es necesario añadir la nueva capa al entorno de trabajo.

```
$ bitbake-layers create-layer --priority 7 ../layers/meta-st/me
ta-st-eleycom
$ bitbake-layers add-layer ../layers/meta-st/meta-st-eleycom/
```

Dentro de ella se añaden un directorio por cada módulo o paquete a instalar. Cada uno de estos directorios contiene como mínimo un archivo **.bb** (receta) con las indicaciones que requiere Yocto para su instalación.

4.5.1 Recetas para módulos Python

Aquellas recetas para la instalación de modulo Python se utiliza el paquete **pipoe**. Este permite crear recetas a partir de la versión del módulo, la versión de Python y el nombre que corresponda en la distribución pypi, para paquetes Python.

```
$ pip3 install pipoe
```

Se crea el directorio que contendrá la receta. El nombre del directorio debe coincidir con el de la receta, a excepción de la versión de la receta. Yocto permite tener diferentes versiones de recetas, pero compilará por defecto la última disponible.

```
$ cd openstlinux-5.10-dunfell-mp1-21-03-31/layers/meta-st/meta-st-eleycom/recipes-st-eleycom/
$ mkdir python3-asyncua
$ pipoe -p asyncua -v 0.9.92 -y python3
```

Es posible que **pipoe** requiera el nombre de licencia del módulo. Para ellos se accede al portal web de **pypi**, se busca el módulo a instalar y se copia la licencia que utiliza.

Pipoe también crea las recetas de los módulos de los que dependerá el solicitado, junto con un archivo **.inc** donde se indica la versión de módulo.

Para la librería asyncua se obtiene la receta python3-asyncua_0.9.92.bb con el siguiente contenido:

```
SUMMARY = "Pure Python OPC-UA client and server library"
HOMEPAGE = "http://freeopcua.github.io/"
AUTHOR = "Olivier Roulet-Dubonnet <olivier.roulet@gmail.com>"
LICENSE = "LGPLv3+"
LIC_FILES_CHKSUM = "file://COPYING;md5=e6a600fd5e1d9cbde2d983680233ad02"

SRC_URI = "https://files.pythonhosted.org/packages/09/b4/75286f234bb26cba7091117f9bc668d7939816601ab0ab148926f6290042/asyncua-0.9.92.tar.gz"
SRC_URI[md5sum] = "05b6d5825846241bae2f081324125f75"
SRC_URI[sha256sum] = "7fed0034935485b76605c87f65519b8a93bb6416f03cfd7156a21311912b4c1f"

S = "${WORKDIR}/asyncua-0.9.92"

RDEPENDS_${PN} = "python3-aiofiles python3-aiosqlite python3-dateutil python3-pytz python3-cryptography python3-sortedcontainers"

inherit setuptools3
```

Cabe destacar de esta receta:

- **SRC_URI.** Se utiliza para descubrir y descargar los archivos de código fuente en el servidor pypi. Esto datos pueden obtenerse manualmente desde el portal web,

accediendo al apartado "*dowload files/source distribution"* y posteriormente escogemos los algoritmos de codificación indicados (md5sum, sha256sum).

- **S.** Indica el directorio donde instalar el paquete, dentro del sistema operativo.
- **RDEPENDS_${PN}.** Indica los módulos de los que depende.
- **inherit setuptools3.** Se hereda setuptools3 como paquete de instalación de módulos, es decir, se encarga de instalar los módulos, a partir de los datos indicados en la receta.

Siguiendo estos mismos pasos se crean las recetas para el resto de los módulos Python indicados en el apartado **4.1.4** .

4.5.2 Recetas propias

La implementación de los servidores en la distribución también se hace mediante recetas. En este caso son creadas manualmente. Se necesitarán tres nuevas recetas:

- **network-config**. Encargada de configura la ip fija para el puerto ethernet. También permite la conexión Wifi especificando el SSID y la contraseña del dispositivo al que se desea conectar.
- **Opcua-server**. Instancia el binario del servidor Opcua y le indica al sistema operativo que lo ejecute en el arranque.
- **Web-server**. Instancia los directorios y archivos necesarios para el servidor web y le indica al sistema operativo que lo ejecute en el arranque.

Se accede al directorio de la capa creada y se crea el un directorio para cada uno al igual que los casos anteriores. Este directorio contiene la nueva receta y un nuevo directorio con los archivos a instanciar en el sistema operativo de la placa de evaluación.

Para configuración de red en OpenStLinux se crean archivos ***.network*** bajo el directorio `/etc/sysconfig/network`, siendo un archivo para cada interfaz de red.

Primero se accede al directorio de la receta y se crea un nuevo directorio con los archivos necesarios para la configuración:

```
$ cd openstlinux-5.10-dunfell-mp1-21-03-31/layers/meta-st/meta-st-eleycom/recipes-st-eleycom/network-config
$ mkdir file
$nano /file/04-eth.network
$nano /file/wlan0.network
$nano /file/wlan0.network
$nano /file/wpa_supplicant-wlan0.conf
```

El archivo **04-eth.network** contiene la siguiente configuración para una dirección ip fija sobre ethernet:

```
[Match]
Name=eth0
[Network]
Address=192.168.1.101/24
Gateway=192.168.1.1
```

El archivo **wlan0.network** contiene la configuración para una ipv4 dada por el servidor DHCP:

```
[Match]
Name=eth0
[Network]
DHCP=ipv4
```

Para la conexión wifi mediante la interfaz wlan0 es necesario crear un archivo de configuración en la dirección **/etc/wpa_supplicant.conf** para los parámetros de autenticación.

```
ctrl_interface=/var/run/wpa_supplicant
eapol_version=1
ap_scan=1
fast_reauth=1 red={
       ssid="<ssid_nombre>"
       psk="<clave_ssid>"
}
```

A continuación, se crea la receta donde se incluirán los archivos anteriores.

```
$ nano network-config.bb
```

Esta receta debe contener los siguientes datos:

```
LICENSE = "CLOSED"
LIC_FILES_CHKSUM = ""
FILESEXTRAPATHS_prepend:= "${THISDIR}/files:"
PACKAGECONFIG_append = " networkd resolved "
SRC_URI = "file://04-eth.network \
file://wlan0.network \
file://wpa_supplicant-wlan0.conf \
```

```
"
#SYSTEMD_AUTO_ENABLE = "enable"
###SYSTEMD_SERVICE_${PN}_append = " wpa_supplicant@wlan0.servic
e "
do_configure () {
     :
}
do_compile () {
     :
}
do_install() {
    install -d ${D}${sysconfdir}/systemd/network
    install -m 0644 ${WORKDIR}/wlan0.network ${D}${sysconfdir}/
systemd/network
    install -m 0644 ${WORKDIR}/04-eth.network ${D}${sysconfdir}
/systemd/network
    install -d ${D}${sysconfdir}/wpa_supplicant/
    install -D -m 600 ${WORKDIR}/wpa_supplicant-wlan0.conf ${D}
${sysconfdir}/wpa_supplicant/wpa_supplicant-wlan0.conf


    install -d ${D}${sysconfdir}/systemd/system/multi-user.targ
et.wants/
    ln -s ${systemd_unitdir}/system/wpa_supplicant@.service ${D
}${sysconfdir}/systemd/system/multi-user.target.wants/wpa_supp
licant@wlan0.service


}
FILES_${PN} += "${sysconfdir}/systemd/network/*"
```

- **FILESEXTRAPATHS_prepend.** Esta variable de entorno indica donde se encuentra los paquetes a instalar.
- **PACKAGECONFIG_append**. Indica el paquete/paquetes de configuración que es necesario añadir.
- **SRC_URI**. Contiene la ruta de los archivos que son necesarios para la ejecución de la propia receta.
- **install -d**. Comprueba si existe el directorio indicado y en caso de no existir lo crea.
- **install -m 644**. Instala el paquete dado como primer parámetro en el directorio indicado como segundo parámetro. Los dígitos 644, codificados en octal, indican los permisos de este archivo. El primero, comenzando por la izquierda, hace referencia al propietario, el segundo al grupo al que pertenece y el tercero al resto de usuarios. Puede tener valores desde 0 al 7 para diferenciar entre las posibles combinaciones de ejecución, lectura y escritura.

- **ln -s.** Se utiliza para crear enlaces simbólicos. Se crea el acceso al archivo dado como primer parámetro en la dirección del segundo parámetro. En este caso se utiliza para lanzar un servicio en el arranque para la conexión wifi.
- **FILES_${PN}.** Indica a Yocto que haga permanente los cambios realizados en el directorio pasados como parámetro. En caso de no añadir esta instrucción bitbake asume que se trata de archivos temporales y son borrados del resultado final.

Para instanciar el servidor OPC-UA generado desde STM32CubeIde se añade el binario que crea la aplicación (OPC-UA-cpp) al directorio de la receta. se añade el archivo de configuración del nuevo servicio en Systemd **OpcUaServer.service** encargado de ejecutar el servidor en el arranque del sistema operativo. Dicho archivo ***.service*** contiene lo siguiente:

```
[[Unit]
Description=Flask web application example
After=network-online.target
Wants=network-online.target
StartLimitIntervalSec=0
[Service]
Restart=always
RestartSec=10

Type=simple
ExecStart=/usr/bin/OpcUaServer/OPC-UA-cpp
StandardInput=tty-force

[Install]
WantedBy=multi-user.target
```

El servicio de Systemd lanza la aplicación que se encuentra en **/usr/bin**, ubicación en la que se instalará utilizando las recetas de Bitbake tal y como se muestra a continuación.

```
LICENSE = "CLOSED"
LIC_FILES_CHKSUM = ""

FILESEXTRAPATHS_prepend := "${THISDIR}/files:"
RDEPENDS_${PN} = "libgpiod"

SRC_URI = "file://OpcUaServer.service \
           file://OPC-UA-cpp"
```

```
do_configure () {
     :
}
do_compile () {
     :
}
do_install () {
     install -d ${D}/usr/bin/OpcUaServer
     install -m 777 ${WORKDIR}/OPC-UA-cpp  ${D}/usr/bin/OpcUaSe
rver
     install -d ${D}/etc/systemd/system
     install -d ${D}/etc/systemd/system/multi-user.target.wants
/
     install -m 0644 ${WORKDIR}/OpcUaServer.service ${D}/etc/sy
stemd/system
     ln -s /etc/systemd/system/OpcUaServer.service ${D}/etc/sys
temd/system/multi-user.target.wants/OpcUaServer.service
}
FILES_${PN} += "/usr/bin/* /etc/systemd/system/*"
```

Cabe destacar la inclusión de la librería "libgpiod" de linux utilizada en el servidor.

Para el caso de servidor web la dinámica es la misma. Se copia el directorio que contiene todos los paquetes del servidor, así como el ejecutable python que instancia el servidor. Además, se incluye el siguiente servicio de arranque del servidor.

```
[Unit]
Description=Flask web application
After=network-online.target OpcUaServer.service
Wants=network-online.target network.target
StartLimitIntervalSec=0

[Service]
Type=simple
WorkingDirectory=/usr/bin/WebServer
ExecStart=/usr/bin/WebServer/server

[Install]
WantedBy=multi-user.target
```

De la receta para el servidor web que incluye los archivos mencionados destaca el comando **cp -R** utilizado para copiar el directorio completo que recibe como primer parámetro. El resultado de dicha receta sería el siguiente.

```
LICENSE = "CLOSED"
LIC_FILES_CHKSUM = ""
FILESEXTRAPATHS_prepend := "${THISDIR}/files:"
RDEPENDS_${PN} = "python3-core\
                  python3-flask-socketio"
SRC_URI = "file://server \
file://project \
            file://python3-WebServer.service"
do_configure () {
     :
}
do_compile () {
     :
}
do_install () {
     install -d ${D}/usr/bin/WebServer
     install -m 755 ${WORKDIR}/server  ${D}/usr/bin/WebServer
     cp -R  ${WORKDIR}/project ${D}/usr/bin/WebServer

     install -d ${D}/etc/systemd/system
     install -d ${D}/etc/systemd/system/multi-user.target.wants
/
     install -m 0644 ${WORKDIR}/python3-WebServer.service ${D}/
etc/systemd/system
     ln -s /etc/systemd/system/python3-WebServer.service ${D}/e
tc/systemd/system/multi-user.target.wants/python3-WebServer.se
rvice
}
FILES_${PN} += "/usr/bin/* /etc/systemd/system/*"
```

4.5.3 Añadir recetas a la imagen del sistema operativo

Tras haber creado nuestras recetas propias es necesario añadir a la receta que implementa Linux. Para ello se accede a la receta **st-image-core** ubicada en la ruta **openstlinux-5.10-dunfell-mp1-21-03-31/layers/meta-st/meta-st-openstlinux/recipes-st/images.** En ella se añaden las recetas generadas en el apartado CORE_IMAGE_EXTRA_INSTALL.

```
include recipes-st/images/st-image.inc
inherit core-image
IMAGE_LINGUAS = "en-us"
IMAGE_FEATURES += "\
    package-management  \
    ssh-server-dropbear \
    "
# INSTALL addons
#
CORE_IMAGE_EXTRA_INSTALL += " \
    resize-helper \
    \
    packagegroup-framework-core-base    \
    packagegroup-framework-tools-base   \
    \
    ${@bb.utils.contains('COMBINED_FEATURES', 'optee', 'package
group-optee-core', '', d)}   \
    ${@bb.utils.contains('COMBINED_FEATURES', 'optee', 'package
group-optee-test', '', d)}   \
    python3 \
    python3-flask \
    python3-flask-socketio \
    python3-flask-login \
    python3-flask-sqlalchemy \
    vsftpd \
    python3-flask-mail \
    python3-eventlet \
    python3-typing-extensions \
    python3-cryptography \
    python3-dateutil \
    python3-pytz \
    python3-asyncua \
    network-config \
    opcua-server \
    web-server \
    "
```

Aquellas recetas añadidas que no hemos generados son requeridas durante la compilación de bitbake para que no resulte errónea. Ya que en algunos archivos de los paquetes de python hacen uso de ellas.

4.6 Software Servidor OPC-UA

Para alcanzar un diseño y arquitectura de programación general, de tal forma que pueda ser reutilizado en diversos proyectos, se opta por la programación orientada objeto y el lenguaje C++ siguiendo las recomendaciones de la empresa.

Su programación se implementa como una aplicación del proyecto OSD32MP157C-512M-BAA_MinimalConfig sobre el que se llevó a cabo la configuración del Device Tree. Tras compilar el proyecto el IDE genera un archivo binario que contendrá el servidor para la comunicación con máquinas. Permitiendo su ejecución desde la distribución de linux creada sin depender de compiladores.

Para conseguir dicho paradigma sobre la librería Open62541 es necesario definir **ObjectTypes** en una jerarquía con relación de herencia. Definiéndose así 7 ObjectTypes, según el modelo físico de ISA-88, asignando se cada uno de ellos a un nodo del servidor:

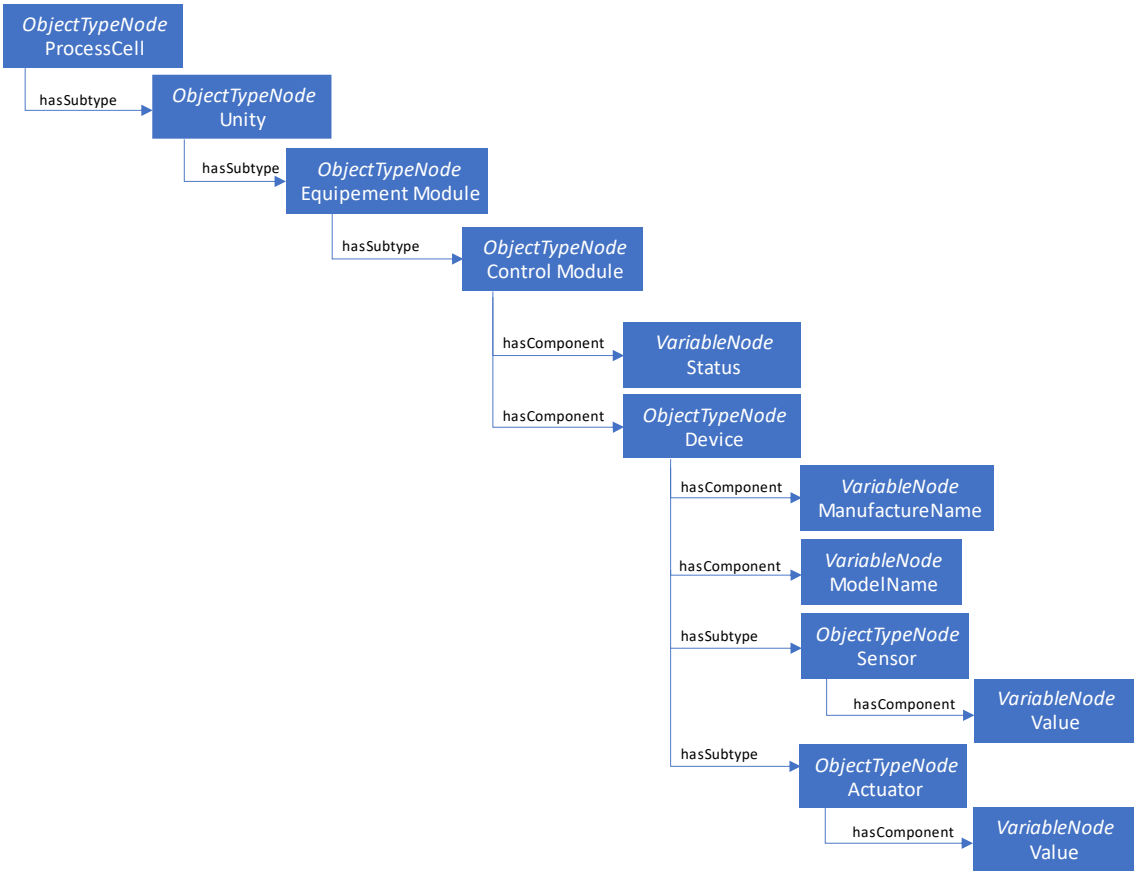


Figura 24. Estructura de objetos para el servidor OPC-UA.

Para continuar con este paradigma de programación orientada a objetos se crea una clase por cada ObjectTypes permitiendo así definir objetos e instanciarlos mediante una programación de alto nivel lo que facilita el uso de la librería open62541. El siguiente

diagrama de clases otorga las características y relaciones de las clases creadas. El desarrollo de los atributos y métodos queda documentado en el ***Manual de desarrollo - Servidor OPC-UA.***

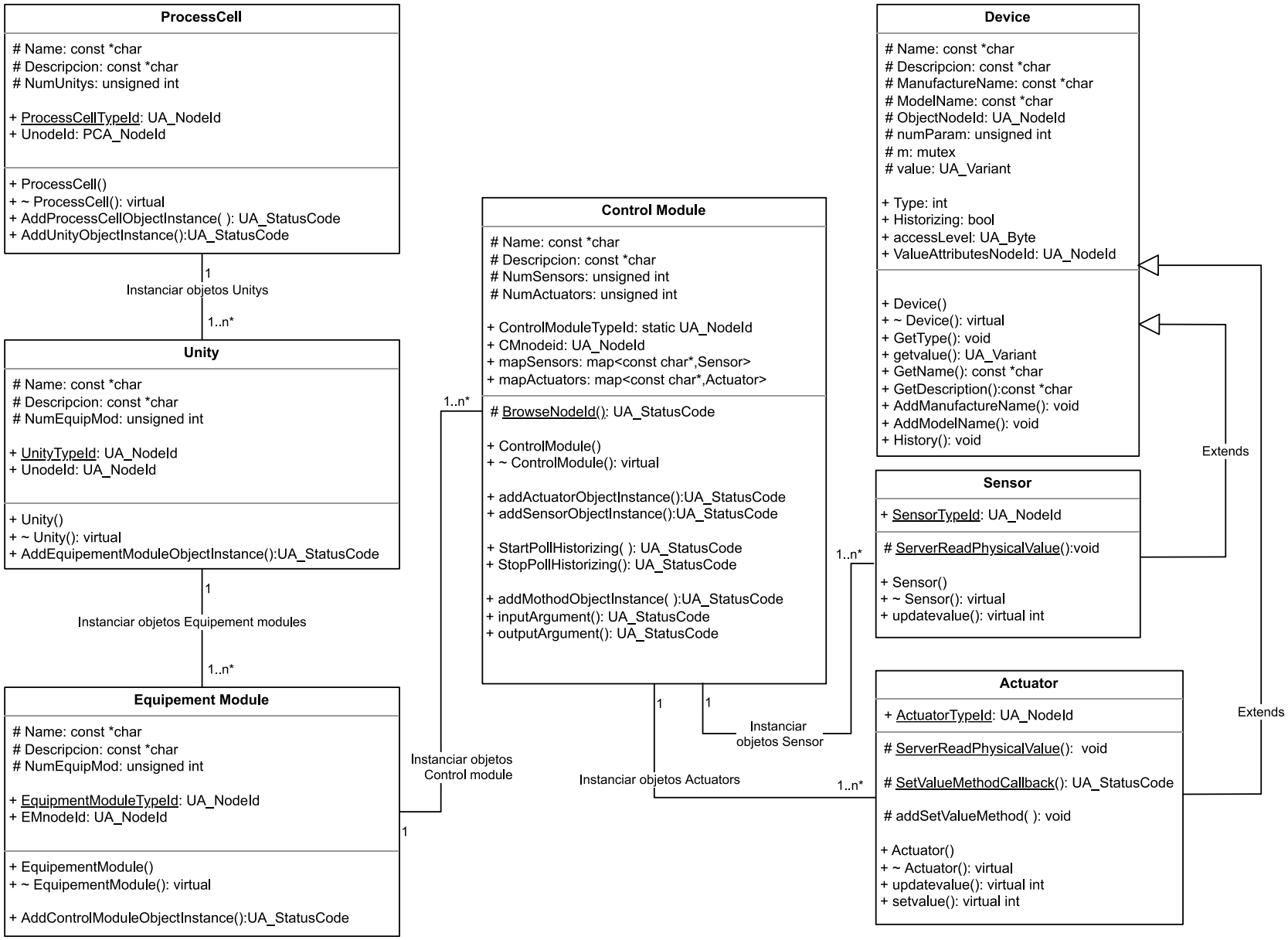


Figura 25. Diagrama de clases según el modelo físico de ISA-88.

El **polimorfismo** es una de las características con mayor potencial de la orientación a objetos, se trata de la capacidad de llamar a funciones distintas con un mismo nombre. Esto permite que una instancia del tipo de la clase base pueda invocar métodos del tipo de la clase derivada. Siempre y cuando sean métodos redefinidos y que hayan sido declarados en la clase base con la palabra virtual. De esta forma se crearán clases derivadas, de las clases bases Sensor y Actuator, que permitan definir el entorno y el periférico que utiliza cada dispositivo (Gpio, I2C, SPI, etc) aunque para el resto de las clases del servidor sigan siendo simples sensores y actuadores. Luego el paradigma se vuelve completamente extensible a cualquier periférico sin necesidad de modificar las clases ya creadas.

Es necesario crear una clase para cada tipo de periférico a usar, las clases existentes son: *DigitalInputGpio, DigitalOutpuGpio, I2cSensingDevice y I2cActuationDevice.* El documento **manual de programador- Servidor OPC-UA** contiene información para crear nuevas clases de periféricos.

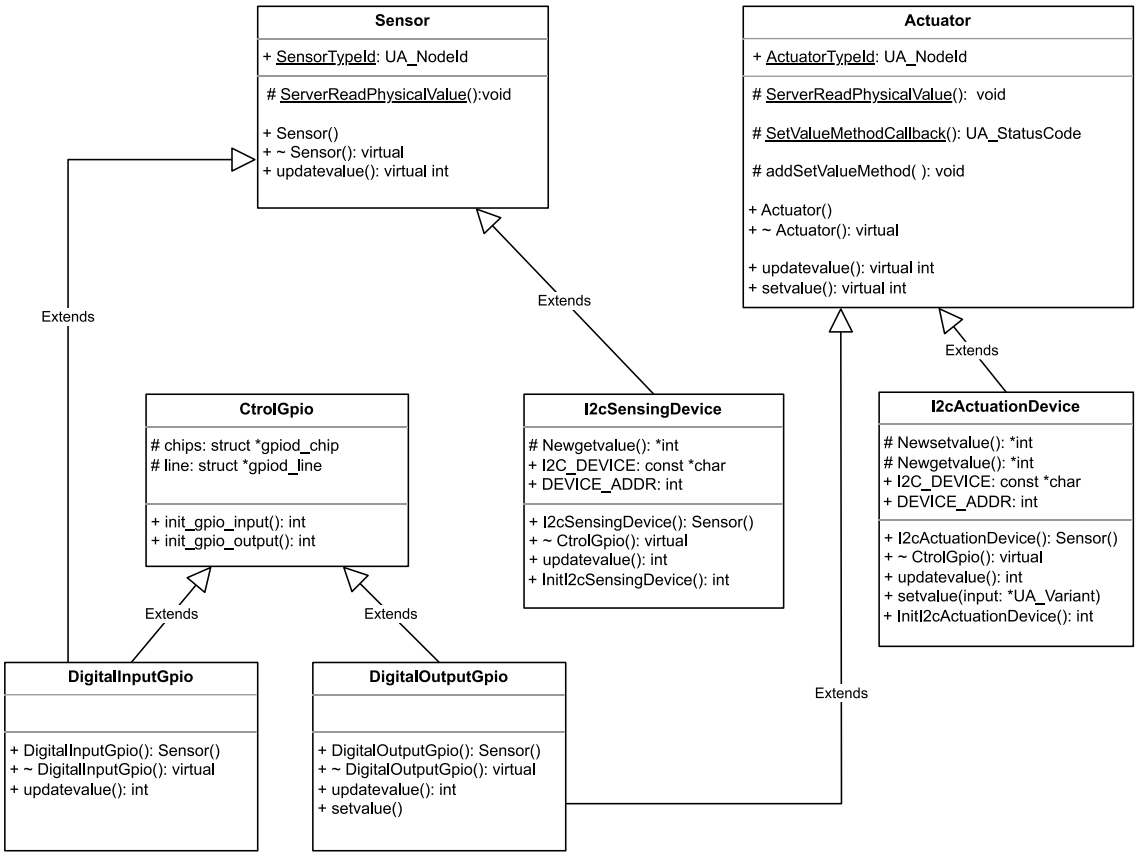


Figura 26. Diagrama de clase para sensores y actuadores del servidor OPC-UA.

A continuación, se presenta un diagrama secuencial de la interacción entre clientes y el servidor OPC-UA.

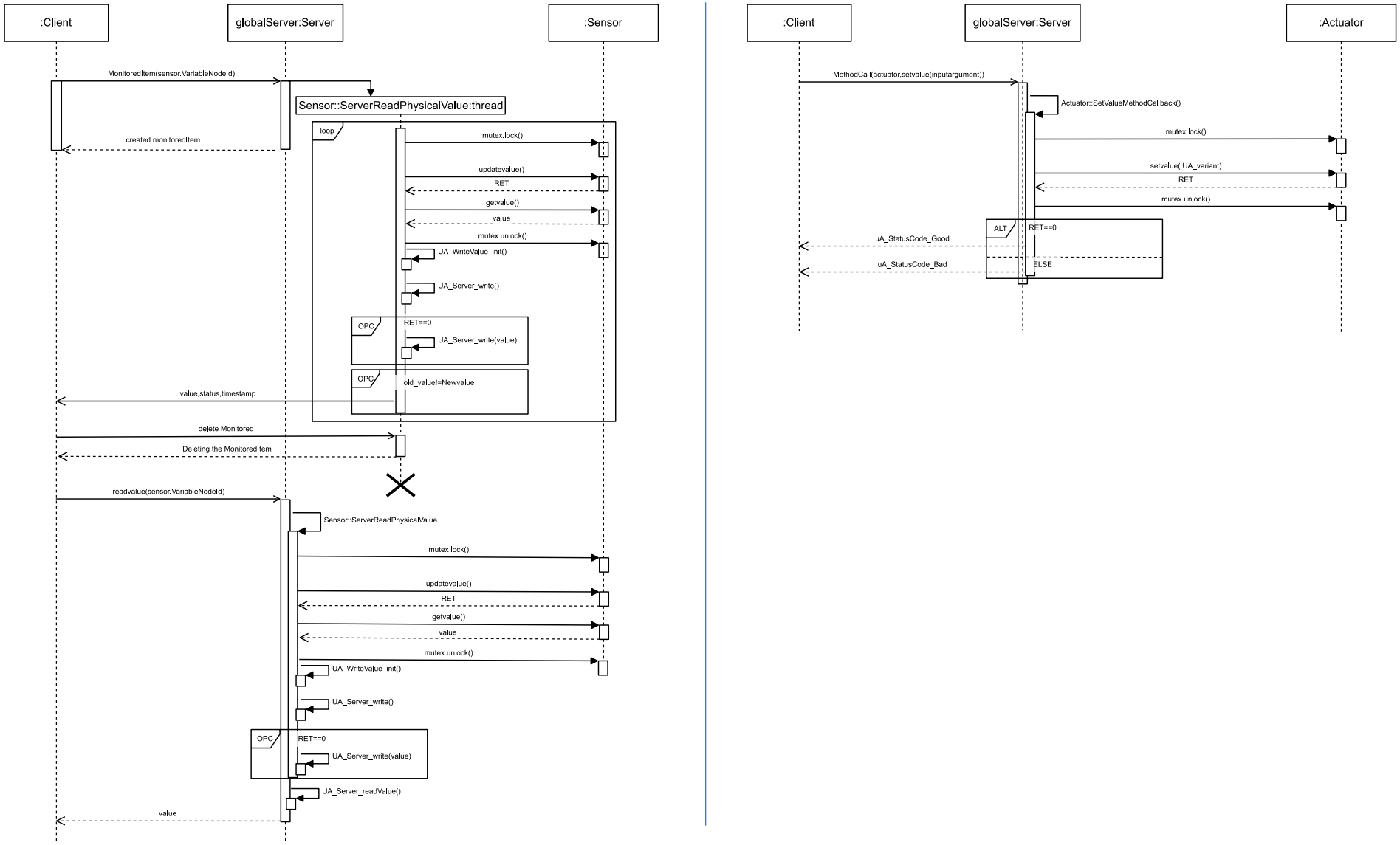


Figura 27. Diagrama secuencial servidor OPC-UA.

4.7 Software Servidor Web

El funcionamiento de servidor se basa en atender solicitudes HTTP para el envío plantillas HTML y solicitudes de datos que no con lleven características de tiempo real. Mediante canales SocketIo se atienden las solicitudes de monitoreo de dispositivos, así como la llamada a métodos del servidor OPC-UA.

Cada vez que un usuario convencional solicita una nueva vista el servidor le envía el HTML que le corresponde. Cuando se requiere cargar bloques de datos y no una vista completa se utiliza AJAX. Este permite devolver conjunto de datos tipo JSON al navegador del usurario web sin recargar la plantilla completa. Es utilizado para cargar el árbol de datos del servidor OPC-UA entre otros.

El servidor web integra un **cliente OPC-UA** que corren en hilos diferentes, pero el último queda a la espera de que algún usuario web solicite datos de los dispositivos conectados al servidor OPC-UA. Cuando el servidor web recibe una petición de monitoreo de dispositivos crea un hilo de fondo denominado ***Worker*** este será encargado de solicitar una suscripción al cliente OPC-UA, el cual crea una monitorización del nodo del servidor OPC-UA. Posteriormente el Worker accede a los datos que dicho cliente inyecta en la tabla de datos ***Subcription_nodes*** y los envía al usuario web. Se crea un Worker por cada usuario web permitiendo así cumplir los requisitos de tiempo real y evita saturar al servidor web. Cuando un usuario deja de monitorizar nodos su correspondiente Worker envía una solicitud de darse de baja y muere. En el caso de que no exista ninguna suscripción de usuarios web a los nodos de *Subcription_nodes,* el cliente OPC-UA elimina la monitorización de dichos nodos y borra sus registros de la base de datos.

Las comunicaciones directas entre Workers y el cliente OPC-UA se lleva a cabo mediante *queue*[8] de tal forma que se acceda a los datos de forma segura desde los diferentes hilos. El acceso a la tabla *Subcription_nodes* se lleva a cabo de forma segura ya que cada hilo inicia una sesión diferente en *SqLite*, y es la base de datos quien gestiona y mantiene la integridad de los registros.

El siguiente diagrama secuencial representa las relaciones entre los objetos del servidor web cuando el usuario solicita monitorizar nodos.

[8] El módulo *queque* implementa colas de múltiples productores y múltiples consumidores. Es especialmente útil en la programación de subprocesos cuando la información debe intercambiarse de forma segura entre varios subprocesos.

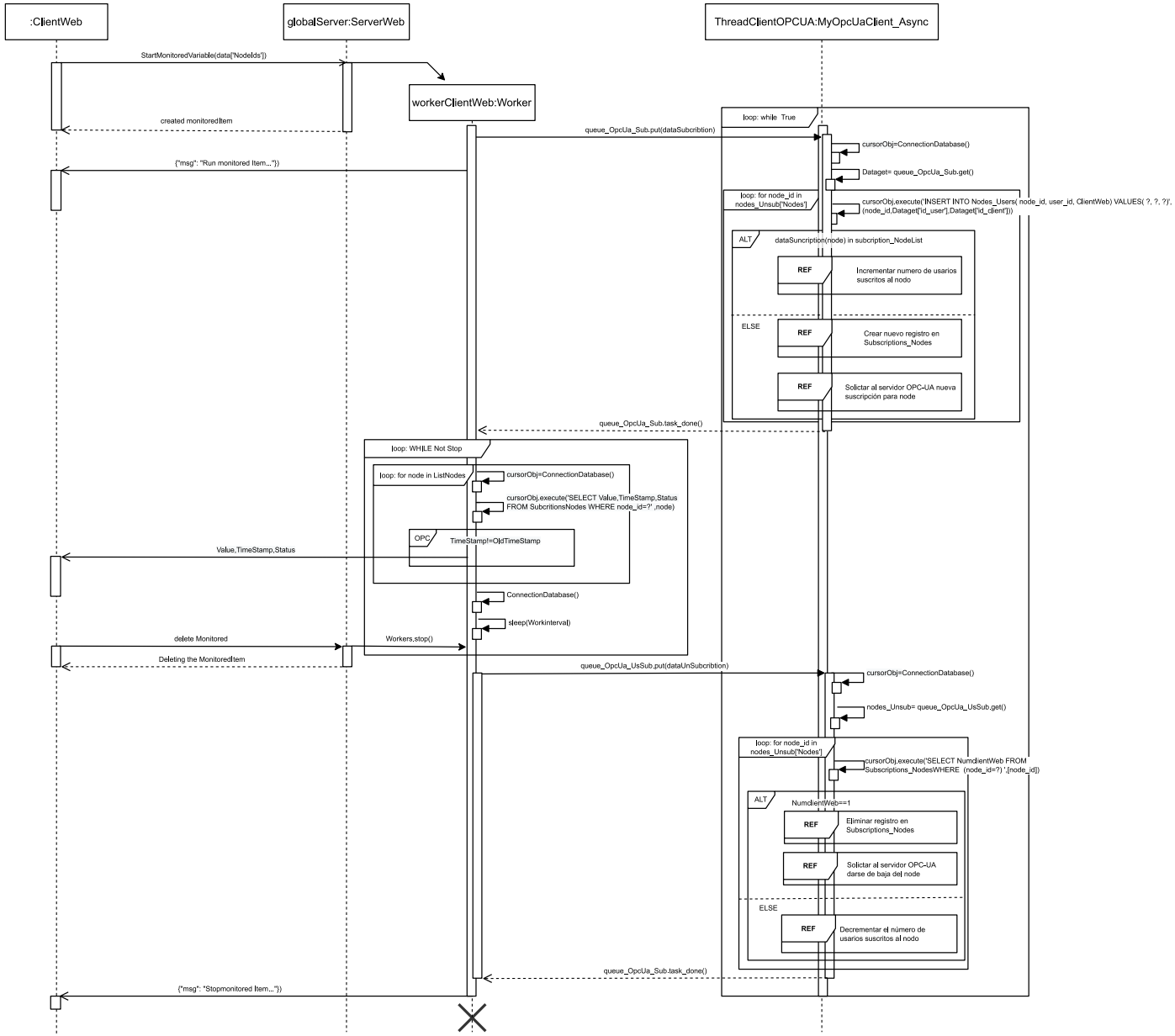


Figura 28. Diagrama secuencial servidor Web.

4.7.1 Estructura de proyecto

La estructura del software para el servidor web sigue el esqueleto indicado por Flask.

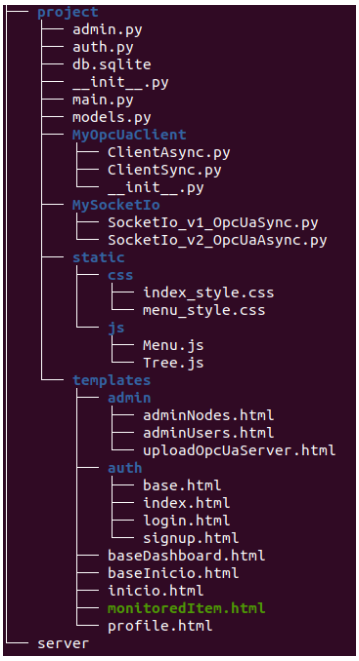


Figura 29. Estructura de Flask para servidor web.

- ***Auth.py.*** Contiene todas las funciones encargas de atender las solicitudes HTTP para autorizar e identificar un cliente web, es decir, las solicitudes Login, Logout, Sing Up y Index.
- ***Admin.py***. Contiene todas las funciones encargas de atender las solicitudes HTTP por parte de los usuarios administradores.
- ***Main***.py. Contiene todas las funciones encargas de atender las solicitudes HTTP por parte de los usuarios convencionales.
- ***Models.py***. Contiene los modelos creados para la base de datos, mediante el ORM de SQLalchemy.
- ***Db.sqlite***. Se trata de la propia base de datos.
- ***__init__.py***. En ella se define la configuración de la app que interpretará el módulo Flask. También incluye la creación de la base de datos en caso de aun no existir, con su correspondiente configuración. ***MySocketIo***. Módulo propio para la gestión del canal socketIo de servidor, encargando de la comunicación Websocket.
- ***MyOpcUaClient***. Módulo propio para la gestión e implementación del cliente OPC-UA.
- ***static***. Es la carpeta por defecto donde Flask buscará los archivos estáticos, CSS y JavaScript.
- ***templates***. Es la carpeta por defecto donde Flask buscará las plantillas HTML que puede exportar el servidor.
- ***Server.*** Se encarga de instanciar la app de Flask indicando la ip y el puerto a utiliza. También instancia el cliente OPC-UA de forma asíncrona.

4.7.2 Base de datos

La arquitectura y tablas que definen la base de datos creada es la siguiente:

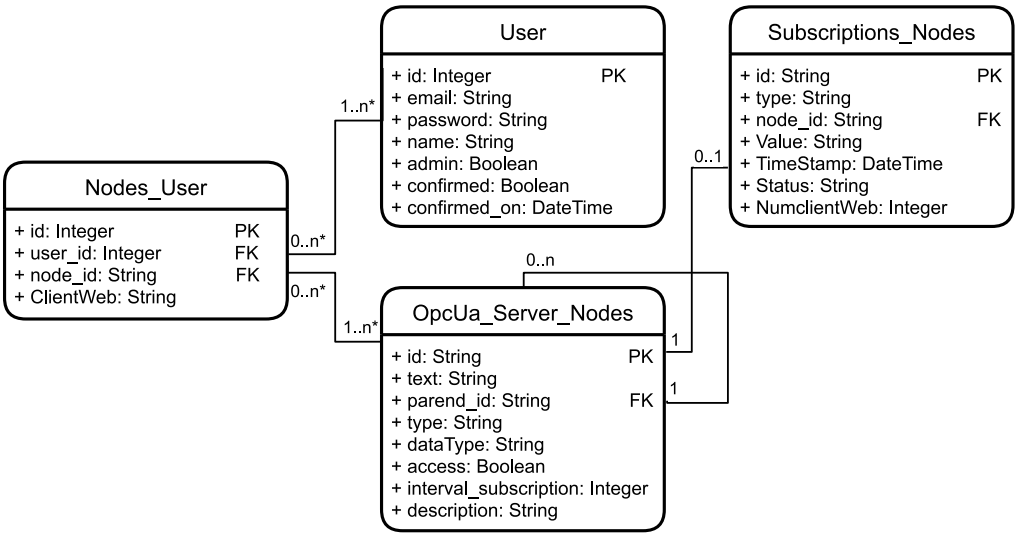


Figura 30. Diagrama de base de datos del servidor web.

- ***User***. Contiene el registro de usuarios. El atributo *admin* permite diferenciar entre usuarios administradores (admin=True) y usuarios convencionales (admin=False). *Confirmed* corresponde a la validación de la cuenta del usuario mediante el envío de un email tras el registro.
- ***OpcuaServerNodes***. Almacena los nodos que conforman el servidor OPC-UA. Parent_id es una relación a un mismo tipo de entidad, de tal forma que cada nodo debe tener un solo padre. Con esta tabla se evita tener que solicitar el árbol de datos completo al servidor OPC-UA. Evitando saturar la red en los proyectos donde los servidores estén ubicados en dispositivos diferentes.
- ***SubscriptionsNodes***. Permite almacenar las suscripciones llevadas a cabo en tiempo de ejecución. Sus datos son volátiles, es decir, se eliminan tras no solicitar ninguna suscripción o detener los servidores. Es utilizada por el cliente OPC-UA y los trabajadores socketIo del servidor web.
- ***Nodes_User.*** Se trata de una tabla abstracta utilizada para relacionar los usuarios con las suscripciones a nodos que tiene hecha en tiempo real. Al tratarse de una relación de muchos a muchos se recomienda crear una tabla específica para este tipo de relaciones ya que las tablas no deben crecer horizontalmente. El atributo *clientWeb* es el id creado por cada aplicación abierta en el navegador siendo independiente del usuario de la sesión.

CAPÍTULO 5 CONCLUSIONES

Con el fin principal de desarrollar un dispositivo general que permite la monitorización y el control remoto de instalaciones, no se podrá valorar su fiabilidad al completo hasta la aplicación del sistema sobre proyectos reales. Existen singularidades en cada proyecto que deben ser solventadas.

Tratándose de una programación en C++, del servidor OPC-UA, la integridad de este dependerá finalmente del programador y su correcta implementación. El desarrollar una programación de alto nivel sobre la librería Open62541 permite tener acceso completo al código fuente, pero no alcanza la versatilidad que proporcionaría un IDE gráfico en la programación.

La utilización de diferentes lenguajes de programación en el proyecto, Linux, C++, Python, JavaScript, HTML y CSS aumenta considerablemente la dificultad de desarrollo por terceros. Por consiguiente, los manuales que complementan al proyecto son imprescindible para la reutilización de los servidores. Dicha documentación no solo incluye la descripción de las aplicaciones sino también de los entresijos de la programación y ejemplos de utilización.

La investigación y configuración de pines de microprocesadores pertenecientes a la familia STM32MP1 permite a la empresa desarrollo hardware en sus propias instalaciones. Abaratando así los costes a la hora de lanzar dispositivos al mercado.

La programación orientada a objetos empleada desde las librerías de bajo nivel hasta los diferentes componentes del servidor web permite la reutilización de código para otros proyectos de diferente índole. Consiguiendo así demostrar la importación del paradigma orientado a objetos para softwares modulares.

La creación de una distribución propia a partir de OpenStLinux no necesita preconfigurar el sistema operativo, desde un terminal, antes de desplegar los servidores. De esta forma se puede distribuir este sistema entre los dispositivos diseñados por la empresa de forma directa.

La modularidad del proyecto es sin duda su mayor avance. La independencia entre los servidores aumenta considerablemente los casos de uso del proyecto. Destacando los siguientes:

- Construcción de un sistema completo de monitorización y control de dispositivos.
- Monitorizar cualquier servidor OPC-UA de otros proveedores mediante el servidor web.
- Incluir el servidor OPC-UA en una red de comunicación entre PLCs.

5.1 Mejoras futuras

Las principales mejoras que se pretenden llevar a cabo, a corto plazo sobre el prototipo desarrollado son:

- Alcanzar los requisitos de ciberseguridad implantando validación por certificado SSL. Para ello es posible utilizar nginx como un proxy inverso de front-end que pase las solicitudes https y wss al servidor Flask.
- Permitir al servidor web conectar con varios servidores OPC-UA de forma simultánea.
- Crear historización de datos y notificación de eventos desde la aplicación web.
- Añadir nuevos periféricos al servidor OPC-UA, teniendo prioridad para SPI y UART.

CAPÍTULO 6 REFERENCIAS

Configuración ethernet:

OSD32MP1-RED Ethernet Issue - Octavo Systems

https://community.st.com/s/question/0D50X0000B8iBSBSQ2/stm32mp157-ethernet-problem

Descripción general de Ethernet - stm32mpu

https://wiki.st.com/stm32mpu/wiki/OpenEmbedded

Ethernet device tree configuration - stm32mpu

How to configure ethernet interface - stm32mpu

Configuración modulo Wifi/bluetooth:

WLAN overview - stm32mpu

WLAN and Bluetooth hardware component - stm32mpu

meta-wifi-credentials/wpa-supplicant_%.bbappend at master · drewmoseley/meta-wifi-credentials · GitHub

How to setup wifi connection - stm32mpu

Configuración interfaz cámara:

GitHub - STMicroelectronics/stm32-ov5640: proporciona el controlador ov5640, parte del componente STM32Cube BSP para todas las series STM32xx.

Librería Open62541:

https://github.com/open62541/open62541/tree/1.2

https://open62541.org/doc/current/tutorial_server_object.html

How to install OPC UA - stm32mpu

Implementing an OPC-UA server using open62541 | by Daniel Garcia Coego | Gradiant Talks | Medium

Adding Methods to Objects — open62541 1.2.0-rc2-44-ge5eba7bd documentation

open62541/examples at master · open62541/open62541 · GitHub

Control Gpio desde el espacio de usuario de linux:

Controlling GPIO from Linux User Space

¿Dónde se encuentra la biblioteca libgpiod en STOpenLinux SDK?

How to use the IIO user space interface - stm32mpu

https://elinux.org/Interfacing_with_I2C_Devices

https://www.kernel.org/doc/html/latest/i2c/index.html

Módulos Python:

https://flask.palletsprojects.com/en/2.0.x/

https://flask-socketio.readthedocs.io/en/latest/

https://flask-sqlalchemy.palletsprojects.com/en/2.x/quickstart/#a-minimal-application

https://www.digitalocean.com/community/tutorials/how-to-add-authentication-to-your-app-with-flask-login-es

Tutorial de Flask en español: Desarrollando una aplicación web en Python

https://flask-socketio.readthedocs.io/en/latest/deployment.html

https://github.com/FreeOpcUa/opcua-asyncio

https://virtualenv.pypa.io/en/latest/

Yocto Project:

https://wiki.st.com/stm32mpu/wiki/OpenSTLinux_distribution

https://www.yoctoproject.org/docs/1.6/bitbake-user-manual/bitbake-user-manual.html

https://pypi.org/project/pipoe/

https://stackoverflow.com/questions/38862088/how-do-i-add-more-python-modules-to-my-yocto-openembedded-project

GitHub - UpdateHub/meta-updatehub: capa de soporte de OpenEmbedded/Yocto Project para UpdateHub

Programación HTML y CSS:

CSS Colors

HTML 5: Agregar y Eliminar filas de una tabla con JQuery - YouTube

What are glyphicons in Bootstrap ? - GeeksforGeeks

Funciones avanzadas de las tablas HTML y accesibilidad - Aprende sobre desarrollo web | MDN

HTML Table Style Generator by eli geske

Base de datos:

Base de datos: SQL y NoSQL

MongoDB vs SQL – Knowi

SQLite vs MySQL: Análisis A Detalle Para Tu Conveniencia

Cassandra ¿qué es y cuándo usarla? - Refactorizando