UNIVERSIDAD DE MÁLAGA

ESCUELA DE INGENIERÍAS INDUSTRIALES

ESCUELA DE INGENIERÍAS INDUSTRIALES

Departamento: Ingeniería de Sistemas y Automática

Área de conocimiento: Ingeniería de Sistemas y Automática

TRABAJO DE FIN DE MÁSTER

Diseño e Implementación de un Sistema IoT para Control Domótico y para la Gestión Remota en Entornos Industriales

Design and Implementation of an IoT System for Domotic Control and Remote Management in Industrial Environments.

Máster en Ingeniería Industrial

Universidad de Málaga, UMA.

Autor: Francisco Luque del Castillo

Tutor: Francisco Ángel Moreno Dueñas

Co-tutor: Víctor Eugenio Torres López

Málaga Junio, 2025

AGRADECIMIENTOS

A mi familia, por la motivación que me han dado día tras día, en especial a mi padre, por la gran ayuda que me ha dado.

A mi tutor y a mi cotutor por su orientación, disponibilidad y confianza durante el desarrollo de este Trabajo Fin de Máster. Su experiencia y sugerencias han sido clave para dar forma al proyecto y poder probar el funcionamiento del sistema.

A mis compañeros de clase, con quienes he compartido dudas, aprendizajes y buenos momentos a lo largo del máster. Gracias por el apoyo mutuo, la colaboración y el compañerismo que han hecho más llevadero este camino.

También quiero agradecer a los profesores del máster por su dedicación y por haber despertado mi interés en el ámbito del IoT y la automatización industrial.

RESUMEN

Este Trabajo Fin de Máster tiene como objetivo el diseño e implementación de un sistema IoT basado en una arquitectura de microservicios, orientado a la automatización del hogar (domótica) y a la gestión remota de procesos en entornos industriales. Se plantea una solución híbrida que combina dispositivos comerciales con dispositivos DIY (*Do It Yourself*), lo cual permite reducir costes. La arquitectura desarrollada destaca por su escalabilidad, modularidad y facilidad de integración, permitiendo la incorporación de nuevos servicios, sensores o actuadores sin afectar a la estructura general.

El sistema propuesto utiliza tecnologías de contenedores Docker para desplegar los distintos microservicios necesarios, y se apoya en protocolos de comunicación ligeros y eficientes como MQTT, junto con redes de bajo ancho de banda como Zigbee. Para la interacción del usuario se emplean plataformas *open source* como *Home Assistant* y Node-RED, que permiten visualizar datos en tiempo real, gestionar dispositivos conectados y automatizar escenas específicas para cada entorno. Además, se ha garantizado un enfoque multiplataforma, que facilita su acceso desde diferentes dispositivos y sistemas operativos.

El proyecto ha sido validado mediante la instalación y prueba del sistema en dos escenarios reales: una vivienda de dos plantas, y el aula Beckhoff de la Escuela de Ingenierías Industriales de la Universidad de Málaga. En ambos casos, el sistema ha demostrado ser funcional, adaptable y seguro, ofreciendo mejoras en eficiencia energética, confort y control de dispositivos.

Los resultados obtenidos demuestran que es posible desarrollar una solución IoT robusta y de bajo coste con tecnologías ampliamente accesibles, capaz de adaptarse tanto a las necesidades del entorno doméstico como del industrial. Asimismo, el trabajo fomenta la cultura *maker* y el aprendizaje de tecnologías emergentes en el campo de la automatización y la industria 4.0.

ABSTRACT

This Master's Thesis aims to design and implement an IoT system based on a microservices architecture, oriented towards home automation (domotics) and the remote management of processes in industrial environments. A hybrid solution is proposed, combining commercial devices with DIY (Do It Yourself) components, which allows for cost reduction. The developed architecture stands out for its scalability, modularity, and ease of integration, enabling the incorporation of new services, sensors, or actuators without impacting the overall system structure.

The proposed system uses Docker container technology to deploy the necessary microservices and relies on lightweight and efficient communication protocols such as MQTT, along with low-bandwidth networks like Zigbee. For user interaction, open-source platforms such as Home Assistant and Node-RED are employed, allowing real-time data visualization, device management, and the automation of customized scenes for each environment. Furthermore, a cross-platform approach has been ensured, facilitating access from various devices and operating systems.

The project has been validated through the installation and testing of the system in two real-world scenarios: a two-story house and the Beckhoff lab at the School of Industrial Engineering

at the University of Málaga. In both cases, the system proved to be functional, adaptable, and secure, offering improvements in energy efficiency, comfort, and device control.

The results obtained demonstrate that it is possible to develop a robust and low-cost IoT solution using widely accessible technologies, capable of adapting to both domestic and industrial needs. Additionally, the project promotes the maker culture and the learning of emerging technologies in the fields of automation and Industry 4.0.

Índice

Índice de Figuras

Índice de tablas

CAPÍTULO 1. INTRODUCCIÓN

1.1. Motivación

En estos últimos años, el Internet de las Cosas (IoT) se ha convertido en una de las tecnologías con mayor potencial de transformar la industria como el hogar que conocemos hoy. La capacidad de interconexión entre los dispositivos inteligentes ha permitido la automatización de procesos, mejorando la eficiencia energética y el uso de recursos.

En un entorno doméstico, el IoT permite la gestión remota de múltiples dispositivos, como la iluminación, la climatización o los sistemas de seguridad, mejorando el confort y reduciendo el consumo eléctrico, además de ser personalizable para cada usuario. En el sector Industrial, se ha convertido en una herramienta esencial para la monitorización de procesos, la prevención y la automatización de tareas repetitivas. Esto se traduce en un aumento de la productividad y la seguridad y en la reducción de tiempos muertos.

La motivación de este proyecto surge del interés por desarrollar una solución versátil que pueda responder a las necesidades de ambos entornos, y que, al mismo tiempo sirva como recurso académico. La posibilidad de poder implantar un sistema basado en esta tecnología tanto en un hogar como en un laboratorio de automatización industrial (concretamente el aula Beckhoff de la Escuela de Ingenierías Industriales), permite explorar la capacidad y el potencial de estos sistemas IoT, demostrando su funcionalidad, escalabilidad y adaptabilidad a los distintos entornos.

Además, con este proyecto se pretende explorar la posibilidad de integrar dispositivos DIY (*Do It Yourself*) en plataformas de IoT existentes, reduciendo, de esta forma, el coste de los sensores a utilizar y fomentando la cultura *maker*.

1.2. Objetivo del proyecto

El objetivo de este proyecto es diseñar e implementar un sistema IoT basado en una arquitectura de microservicios que se comunican entre sí, versátil y adaptable, que pueda emplearse tanto en entornos domésticos como industriales.

El sistema IoT estará conformado por una red de dispositivos comerciales y de dispositivos DIY, que convivirán en el mismo sistema, pudiendo comunicarse de manera transparente entre ellos sin problemas de compatibilidad.

Es también importante destacar que este sistema debe de ser escalable, siendo capaz de integrar nuevos servicios y aplicaciones. Por ejemplo, es necesario que pueda aceptar nuevos protocolos de comunicación aumentando así su compatibilidad con diferentes dispositivos. Las ampliaciones del sistema no pueden afectar a la arquitectura principal del sistema, para evitar errores y que sea fácil de implementar en el sistema.

1.3. Estructura del documento

El presente Trabajo Final de Máster (TFM) se estructura en 6 capítulos, diseñados para introducir al lector de forma progresiva en el ámbito del IoT, llegando hasta su implementación práctica en los dos entornos mencionados anteriormente. La estructura de este documento es la siguiente.

Capítulo 1. Introducción. Se presenta la motivación del proyecto, sus objetivos y una visión general de los sistemas IoT.

Capítulo 2. Estado del arte. Se analizan los conceptos relacionados con el IoT así como su aplicación en entornos domésticos e industriales en la actualidad.

Capítulo 3. Descripción del sistema y herramientas. En este capítulo se describen las tecnologías y recursos utilizados para el desarrollo del sistema, incluyendo contenedores Docker, protocolos de comunicación, dispositivos IoT como sensores y actuadores, y diferentes microservicios utilizados.

Capítulo 4. Instalación y pruebas en un entorno doméstico. Se documenta el proceso de diseño e integración del sistema IoT en una vivienda real.

Capítulo 5. Instalación y pruebas en un entorno industrial. Se expone el diseño e integración del sistema IoT en un entorno industrial de laboratorio.

Capítulo 6. Conclusiones. Se sintetizan las opiniones y los resultados alcanzados. Además, se exponen las líneas de mejoras y la posible expansión del sistema en desarrollos futuros.

CAPÍTULO 2. ESTADO DEL ARTE

2.1. Definición de IoT. ¿Qué es el Internet of the Things?

El Internet de las Cosas (IoT, por sus siglas en inglés) es una red de dispositivos físicos conectados a internet que pueden recopilar, intercambiar y actuar sobre datos [1]. Estos dispositivos, que van desde electrodomésticos y sensores industriales hasta vehículos y sistemas de salud, están equipados con tecnologías que les permiten comunicarse entre sí y con sistemas centrales, lo que facilita la automatización, el monitoreo en tiempo real y la toma de decisiones inteligente en diversos ámbitos.

En la actualidad existen millones de dispositivos conectados a internet gracias a la llegada de los chips de bajo coste y consumo. La gran mayoría de los dispositivos que usamos a diario están conectados a internet y recopilan datos e información sobre nosotros. Entre estos dispositivos podemos encontrar cepillos de dientes, aspiradoras, coches, relojes, neveras, lavavajillas, etc. La información que se recopila, en tiempo real, sirve para emitir órdenes a los dispositivos y responder a estímulos de forma inteligente.

Habitualmente, en un sistema IoT se distinguen tres componentes principales (Figura 2.1):

- **Dispositivos Inteligentes**. Son las "cosas" conectadas a la red. Pueden ser máquinas, actuadores, sensores, electrodomésticos, controladores industriales, etc. Recogen datos del entorno o ejecutan acciones según las órdenes recibidas.

- **Procesamiento**. Es el sistema que se encarga de gestionar la información recibida, la procesa y manda las órdenes para responder de forma inteligente. En la actualidad se utilizan tecnologías de *Machine Learning* (ML) o *Inteligencia Artificial* (IA) para actuar de forma autónoma e inteligente.

- **Interfaz de Usuario**. Es la interfaz gráfica (normalmente una aplicación web o móvil) donde el usuario puede visualizar los datos, gestionar los dispositivos inteligentes y controlar su comportamiento.

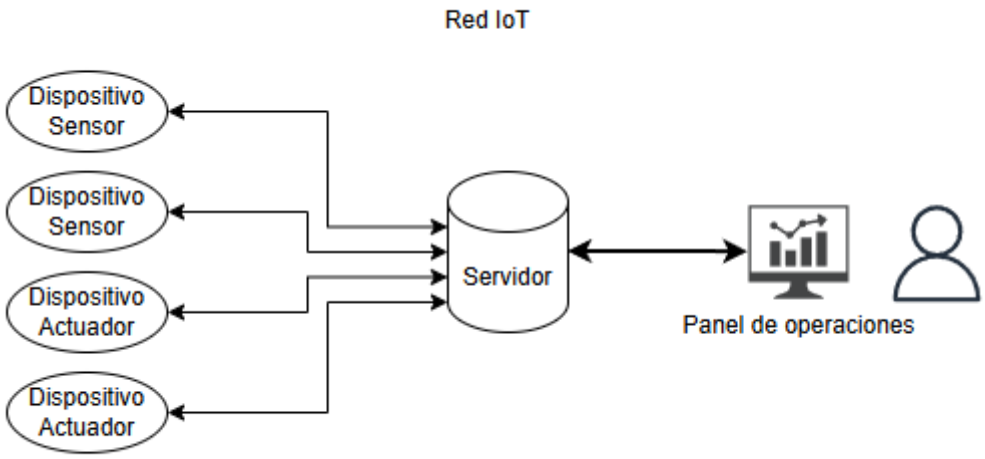

Figura 2.1. Esquema componentes IoT

Los sistemas IoT han evolucionado tanto en los últimos años que está surgiendo una gran variedad de aplicaciones en diversos campos, como, por ejemplo:

- **Hogares**: Control inteligente de las viviendas. Incluye control en la iluminación, climatización, seguridad y gestión electrodomésticos.
- **Sanidad**: Monitorización remota de pacientes y gestión de dispositivos médicos.
- **Industria**: Automatización de líneas de producción, mantenimiento predictivo, control energético, seguridad.
- **Agricultura**: Riego inteligente, monitoreo de cultivos.
- **Transporte**: Vehículos autónomos.
- **Energía**: Medidores inteligentes, redes eléctricas inteligentes, buscan la optimización del uso y distribución de la energía.

El IoT está permitiendo la creación de entornos inteligentes donde las interacciones entre los dispositivos y con el entorno se vuelven más fluidas y eficientes, transformando la manera en la que vivimos y trabajamos.

2.2. Domótica

El termino domótica refiere a la tecnología que es capaz de automatizar y controlar los procesos domésticos, que, habitualmente, suelen estar relacionados con el ahorro de energía, con aumentar el bienestar y mejorar la calidad de vida [3].

A modo de ejemplo, la incorporación de los sistemas domóticos en los entornos domésticos permite gestionar de forma inteligente la climatización, la iluminación, el agua caliente, gestiona los electrodomésticos y es capaz de manejar la energía producida por las placas solares, si existen en el domicilio. Además, el sistema domótico puede aprovechar la energía según las tarifas horarias. Por consecuencia, el sistema consigue reducir la factura energética, aumentando la seguridad y confort [4].

La tecnología que se usa está compuesta de sensores que son capaces de recopilar datos e información del entorno como, por ejemplo, un sensor de movimiento que detecta la presencia de una persona en una habitación. Posteriormente, el sistema domótico procesa la información de que el sensor, que se encuentra en el pasillo principal, ha detectado a la persona y, además, hay otro sensor que indica que hay poca luz. Por último, emite la orden a un actuador: en este caso un relé que controla la iluminación del pasillo, ordenando encender la iluminaria para que la persona pudiera ver sin tener que buscar el interruptor del pasillo.

El ejemplo descrito se denomina “escena” y es programada de forma que se ajuste a la vivienda y a las necesidades del usuario (véase en el apartado 4.5 los pasos a seguir para programar estas escenas).

Los beneficios que proporcionan los sistemas de Internet de las cosas son, en general, la mejora del confort, la seguridad y el ahorro energético. Aunque una desventaja de estos

sistemas es la complejidad a la hora de diseñarse. Debido a que cada casa y necesidades son diferentes, no es lo mismo las necesidades de una pareja recién casada que las necesidades de una persona mayor. Por lo tanto, habrá que configurar las diferentes escenas acordes al usuario.

Actualmente, los sistemas domóticos son escasos y tienen un coste excesivo. Sin embargo, en España existe una adopción masiva de dispositivos individuales capaces de integrarse con dispositivos de control de empresas privadas (como Alexa [5] y Google Home [6]). Aunque estos sistemas están limitados y no permiten un ajuste personalizado a la vivienda de cada usuario, es un buen comienzo.

2.3. IoT en la industria

Los sistemas IoT están revolucionando la industria al habilitar la conectividad y la automatización de las líneas de producción y de procesos. En estos entornos, la relación beneficio-coste es muy superior al entorno domótico por lo que su adopción está siendo masiva en los últimos años. Los sistemas IoT en la industria se conocen como IIoT, *Industrial Internet of Things* (Internet Industrial de las Cosas, en español) [7].

De manera análoga a los sistemas IoT domóticos, los sistemas IIoT están compuestos de una red de sensores, instrumentos y dispositivos que se pueden comunicar. Esta infraestructura facilita la recopilación de los datos y los analiza en tiempo real, siendo capaz de optimizar la producción, mejorar la eficiencia y reducir los costes de fabricación.

El IIoT es uno de los pilares fundamentales de la Industria 4.0 [9]. Este concepto engloba la digitalización de todas las operaciones industriales mediante tecnologías como la inteligencia artificial, la robótica, y el análisis de datos. Esto está promoviendo la transición de la fábrica tradicional a un modelo de fábrica inteligente, donde todos los sistemas están interconectados.

El IIoT se diferencia de otras tecnologías debido a sus características, que lo convierten en una herramienta poderosa y necesaria en la industria.

- **Interconectividad**. Las máquinas, sensores, sistemas y trabajadores están conectados a través de una red segura, permitiendo el envío de información en tiempo real.

- **Monitoreo en Tiempo Real**. Los dispositivos IIoT recopilan datos continuamente, proporcionando información del estado de las máquinas, procesos y el entorno al operario de forma inmediata. Estos datos se ilustran en un panel de usuario simple e intuitivo.

- **Análisis de Datos**. Los sistemas IIoT procesan los datos recopilados por los dispositivos con algoritmos avanzados, identificando patrones, anomalías, etc.

- **Confiabilidad, Seguridad y Escalabilidad**. Estos sistemas están diseñados para ser robustos y garantizan la continuidad en las líneas de producción evitando errores humanos y accidentes. Además, los sistemas IIoT pueden escalarse fácilmente para conectar nuevos dispositivos y funciones.

Dentro de la industria existes muchas aplicaciones donde se pueden utilizar la red de IIoT, como, por ejemplo:

- **Mantenimiento Predictivo**. El IIoT permite el monitoreo constante del estado de las máquinas y equipos mediante sensores. La información recopilada se analiza para realizar mantenimientos preventivos evitando averías.

- **Automatización de Procesos**. Gracias a los sensores y actuadores, los sistemas IIoT pueden automatizar procesos industriales. Esto mejora la eficiencia y reduce los errores humanos.

- **Gestión de la Cadena de Suministro**. Estos sistemas son capaces de mejorar la trazabilidad de los productos y el seguimiento de los materiales a lo largo de toda la cadena de producción y suministro.

- **Consumo Energético**. El IIoT permite gestionar de forma eficaz los recursos energéticos y administrar el consumo ajustando la calefacción, la luz, refrigeración, etc. según las necesidades y los recursos energéticos disponibles.

- **Seguridad**. Con sensores distribuidos por toda la planta, las fábricas pueden detectar condiciones con potencial riesgo, como la presencia de gases tóxico, ruidos excesivos, entrada de personas no autorizadas en áreas restringidas, etc. La fábrica puede actuar ante estos estímulos, parando o ralentizar las operaciones cuando un trabajar en una zona de riesgo.

2.4. Definiciones

A lo largo de este documento, se mencionarán algunos conceptos cuyas definiciones simplificadas se muestran a continuación, a modo de recopilación.

- **Inteligencia Artificial**: Es un campo de la informática que se dedica a desarrollar sistemas capaces de realizar tareas que normalmente requieren inteligencia humana [10].

- ***Machine Learning***: (Aprendizaje Automático) Es una rama de la Inteligencia Artificial que se enfoca en desarrollar sistemas capaces de aprender y mejorar de forma automática a partir de la experiencia, sin necesidad de ser programadas expresamente para ello [11].

- **Industria 4.0**: Es un concepto que representa la cuarta revolución industrial, que se caracteriza por la integración de tecnologías digitales.

- **Escena**: Configuración de varios dispositivos inteligentes, programados para realizar una acción en específico en función de ciertas condiciones, comando o necesidades.

- **Microservicios**: Son un estilo de arquitectura de *software* que divide una aplicación en componentes más pequeños e independientes (servicios) que se comunican entre sí a través de una API [12].

- **Imágenes**: Son plantillas que contienen todo lo necesario para ejecutar una aplicación en un entorno aislado (contenedor).

- ***Kernel***: Es el parte central de un Sistema Operativo. Es el núcleo que actúa de intermediario entre el software y el hardware. Se encarga de gestionar los recursos del sistema, como el procesador, la memoria y los dispositivos I/O.

- ***Layout***: Disposición física de los elementos en un espacio definido.

- ***Plug and Play***: Es una tecnología que permite que los dispositivos hardware sean detectados y configurados de forma automática sin necesidad de intervención manual.

- **Paletas**: Es un conjunto de nodos disponibles en el editor visual. Se asemejan a las librerías en otros lenguajes de programación.

CAPÍTULO 3. DESCRIPCIÓN DEL SISTEMA Y HERRAMIENTAS

3.1. Introducción a las herramientas

Antes de abordar en detalle las herramientas empleadas en el desarrollo del sistema IoT, es fundamental comprender las necesidades que condicionan su diseño. Estas necesidades se agrupan principalmente en tres pilares: (i) *software*, (ii) comunicaciones y (iii) *hardware*. Su correcta integración y funcionamiento son clave para garantizar la eficiencia, escalabilidad y fiabilidad del sistema.

En primer lugar, el ***software*** será el encargado de mantener todos los dispositivos interconectados, de analizar los datos recopilados por los sensores y enviar órdenes a los actuadores. Además, debe ofrecer flexibilidad para adaptar e integrar nuevos servicios a medida que evoluciona el sistema. Por ello, se ha optado por el uso de "contenedores", los cuales permiten una gestión modular y eficiente de los distintos componentes del sistema (véase el Apartado 3.2). Así se permite la integración de nuevos servicios de forma rápida y sin problemas al sistema IoT.

Por otro lado, la **conectividad** es fundamental en un sistema IoT. Las comunicaciones entre dispositivos deben de ser fiables y fluidas. Existen diferentes protocolos de red y de comunicación y su uso dependerá del entorno donde se quiera implantar el sistema:

- **Protocolos de comunicación**:
  - **Protocolos ligeros** como MQTT (*Message Queuing Telemetry Transport*), CoAP (*Constrained Application Protocol*), AMQP (*Avanced Message Queuing Protocol*) permiten la transmisión de datos de forma eficiente. Son modelos que consumen pocos recursos y se suelen usar en sensores remotos y en sistemas de monitorización en tiempo real. En este TFM usaremos MQTT como protocolo para comunicar muchos de los sensores y actuadores de las redes IoT que desplegaremos.
  - **Protocolos más robustos** como HTTP/HTTPS, Bacnet y Websocket que requieren una interacción con servicios *web*. Necesitan más recursos y son más pesados.

- **Protocolos de red**:
  - **Wifi**: Asegura la conectividad en espacios domésticos e industriales pequeños.
  - **Redes de bajo consumo**: Son redes diseñadas para la transmisión de datos consumiendo pocos recursos Zigbee o Z-wave. Algunos de los sensores utilizados para este TFM usarán Zigbee como protocolo de comunicaciones.
  - **Redes de bajo ancho de banda**: Son redes de largo alcance, entre las que se encuentran LoRaWAN, NB-IoT, ZIgbee.
  - **Ethernet**: Conexión muy estable, segura y rápida. Suele usarse en entornos industriales.

Por último, se detallará el ***hardware*** necesario para el correcto funcionamiento de un sistema IoT. Para ello, hay que diferenciar entre los diferentes dispositivos:

- **Nube/Servidor**: Un equipo que recopila, almacena y procesa la información.
- **Antena (*Dongle*):** Dispositivo que permite crear una red.
- **Sensores**: Dispositivos encargados de recolectar información.
- **Actuadores**: Los encargados de interactuar con el entorno.

Estos elementos se explicarán con más detalle en la Sección 3.4.

3.2. Contenedores

Un contenedor es una tecnología que permite aislar y empaquetar aplicaciones con su entorno de ejecución [13], incluyendo las librerías, APIs y los archivos necesarios para la ejecución de la aplicación. La principal ventaja de usar este sistema es que se eliminan los posibles conflictos de compatibilidad que puedan surgir al usar las aplicaciones en diferentes equipos, ya que aíslan los procesos que ejecuta el sistema IoT del resto de procesos que está ejecutando el servidor. Esto facilita enormemente la migración a otros equipos, ya que los servicios no dependen directamente de los procesos del servidor. También hay que destacar que es una tecnología *Open Source*, existiendo mucha documentación en internet sobre la correcta creación de los contenedores.

El entorno que se usará para gestionar los contenedores en este TFM se llama Docker [14]. Es un gestor de contenedores con una comunidad muy grande, donde se pueden encontrar una gran variedad de plantillas, conocidas como imágenes, que contienen todo lo necesario para ejecutar una aplicación en un entorno aislado.

Además, Docker permite usar los contenedores como pequeñas máquinas virtuales livianas consiguiendo una gran flexibilidad y dando una gran oportunidad de personalizar el sistema IoT, ya que se podrían correr aplicaciones en C++, Python, crear contenedores con redes neuronales, IA, etc.

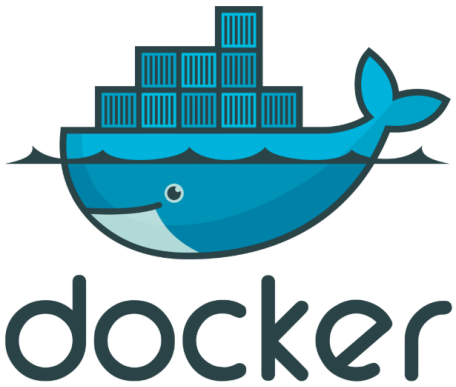

Figura 3.1. Logo Docker

Una alternativa al uso de contenedores sería usar máquinas virtuales y dentro de estas correr los diferentes microservicios que vamos a necesitar para que funcione nuestro sistema IoT. De forma muy breve, estos microservicios son componentes independientes y especializados que se encargan de realizar tareas concretas dentro de nuestro sistema IoT (aportaremos más información sobre los microservicios en la Sección 3.3). Sin embargo, la opción de usar máquinas virtuales no es viable debido a que se necesitarían más recursos y sería difícil hacer funcionar muchas máquinas virtuales en equipos livianos con pocos recursos. Además, se necesitaría una configuración inicial muy laboriosa.

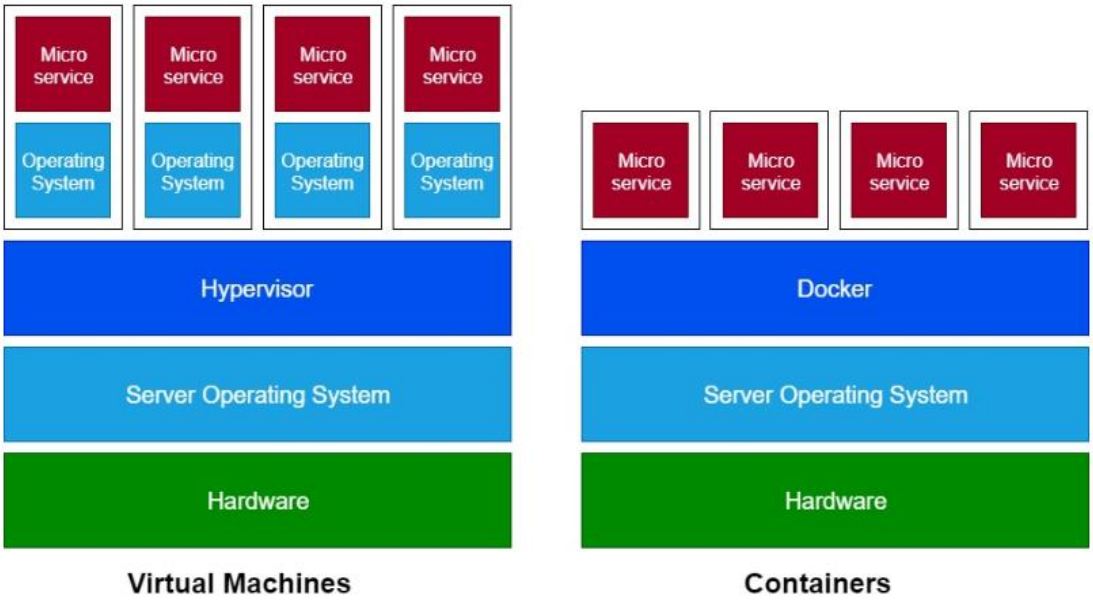

Figura 3.2. Componentes de una máquina virtual (izquierda) y Docker (derecha). Imagen reproducida de [16]

En la Figura 3.2 se puede observar, a la izquierda, la estructura de un sistema operativo corriendo varias máquinas virtuales para ejecutar los microservicios. A la derecha se representa un sistema operativo ejecutando Docker y éste ejecuta los diferentes microservicios. Como se puede observar, en Docker, no hace falta estar ejecutando el sistema operativo de cada microservicio, por lo que se ejecutarán de una manera más fluida y tendrá un menor coste computacional, permitiendo ejecutar todos los contenedores en sistemas de bajo consumo y rendimiento.

En resumen, Docker simplifica el proceso de diseño y ejecución de *software*, lo que lo convierte en una herramienta esencial para el desarrollo de este proyecto. Los contenedores son capaces de encapsular la aplicación y sus dependencias compartiendo el *kernel* con el sistema operativo del servidor, lo que lo hace más eficientes que usar máquinas virtuales.

3.2.1. Funcionamiento de Docker

Docker utiliza una arquitectura basada en cliente-servidor y está conformado de varios elementos:

- ***Docker Engine***: motor de Docker que permite crear y gestionar los contenedores.
- ***Docker Client***: interfaz que permite a los usuarios interactuar con Docker.
- ***Docker Daemon***: Proceso que se ejecuta en el servidor en segundo plano que gestiona los objetos de Docker, ya sean contenedores, imágenes, redes y volúmenes.
- ***Docker Images***: plantillas de Docker que se pueden descargar de la comunidad *Docker Hub*. Contienen la aplicación a ejecutar y todas las dependencias necesarias.
- ***Dockerfile***: Script que se define para personalizar los componentes y dependencias de la imagen. También contienen las instrucciones a la hora de compilar la imagen, comandos a ejecutar y que archivos se deben de incluir.

- ***Docker Compose***: Es una herramienta que define y ejecuta aplicaciones en multi-contenedor. Permite configurar servicios, redes y volúmenes en un solo archivo YAML, facilitando la gestión de aplicaciones complejas.

Todos estos son los elementos que permiten ejecutar los contenedores. Para este proyecto se van a hacer uso de varias imágenes descargadas en *Docker Hub* y la herramienta de *Docker Compose*, ya que es una herramienta muy potente que permite gestionar varios contenedores a la vez y facilita la migración del sistema IoT a otros servidores. En el siguiente apartado se explicará cómo se ha diseñado el fichero *Docker Compose* para ejecutar todos los contenedores.

3.2.2. Estructura de Docker Compose

Docker Compose utiliza un archivo de configuración llamado "*docker-compose.yml",* que reúne toda la información y configuración para gestionar los contenedores de Docker de forma simultánea. Así, podemos lanzar, de manera sencilla, varios contenedores que se ejecutarán a la vez. En este punto se pretende explicar la estructura de este archivo para poder programar los contenedores del siguiente capítulo.

```yaml
version: '3.8'

services:
  zigbeeNetwork:
      container_name: zigbee2mqtt
      image: koenkk/zigbee2mqtt
      restart: unless-stopped
      volumes:
        - ./zigbee2mqtt/data:/app/data
        - /run/udev:/run/udev:ro
      ports:
        # Frontend port
        - 8080:8080
      environment:
        - TZ=Europe/Berlin
      devices:
        # Make sure this matched your adapter location
        - /dev/ttyUSB0:/dev/ttyUSB0

networks:
  zigbee:
```

Figura 3.3. Archivo ejemplo docker-compose.yml

El archivo se compone de las siguientes partes:

- **La versión (*Version*).**
  Esta sección especifica la versión de la sintaxis de *Docker Compose*.

- **Los servicios (*Services*)**.
  Es la sección principal, donde se definen los contenedores. Cada servicio se ejecuta en un contenedor. A su vez los servicios se componen de:
  - *Container_name*: define el nombre del contenedor.
  - *Image*: Especifica la imagen del contenedor, la cual puede ser una imagen de *Docker Hub* o una imagen personalizada.

- **Los puertos (*Ports*)**.
  En esta sección se mapean los puertos que requerirá el servicio, a los puertos del *host*. Declarar correctamente los puertos es esencial para que el servicio sea accesible desde fuera del contenedor. Un ejemplo para el mapeo de los puertos se puede observar en la Figura 3.3 aparece el puerto repetido dos veces de esta manera. "*8080:8080*", esta estructura representa el puerto de la máquina *host* y el puerto del contenedor (Puerto *Host*:Puerto conetenedor). Siendo el puerto del *host* por donde se podrá acceder al servicio desde fuera.

- **Los volúmenes (*Volumes*).**
  Permiten almacenar los datos para que no se pierdan al reiniciar el contenedor. Un volumen es un directorio compartido entre el host y el contenedor. La estructura funciona de forma similar a los puertos. A la izquierda se encuentra la ruta de la carpeta en la máquina host y a la derecha la ruta de la carpeta en el contenedor.

- **Variables de entorno (*Environment*).**
  En esta sección se definen las variables de entorno que usará el contenedor, son útiles para configurar parámetros por defecto o poner credenciales.

- **Redes (*Networks*).**
  Define las redes que se van a utilizar para que los servicios puedan comunicarse entre sí.

- **Dispositivos (*Devices*).**
  Esta opción se utiliza para mapear dispositivos del host dentro del contenedor, se suele usar para darle acceso al contenedor a puertos USB o tarjetas gráficas. Sigue una estructura similar a los volúmenes y puertos.

3.3. Microservicios

Para el diseño de sistemas IoT, el uso de una arquitectura basada en microservicios resulta muy beneficiosa. Este enfoque permite dividir el sistema en componentes independientes que interactúan entre sí, facilitando la escalabilidad, la reutilización de servicios y la facilidad de mantenimiento. Cada funcionalidad del sistema se despliega como un servicio autónomo que se comunica con el resto de los servicios.

Para este TFM, necesitaremos microservicios que se encarguen de: (i) definición y despliegue de nodos y generación de escenas de la red IoT, (ii) creación del panel de usuario, y (iii) definición de protocolos de comunicación. En esta sección describimos los microservicios escogidos.

3.3.1. Esquema de conexión de los servicios

El propósito de este apartado es describir e ilustrar el conexionados de los servicios para formar el sistema IoT.

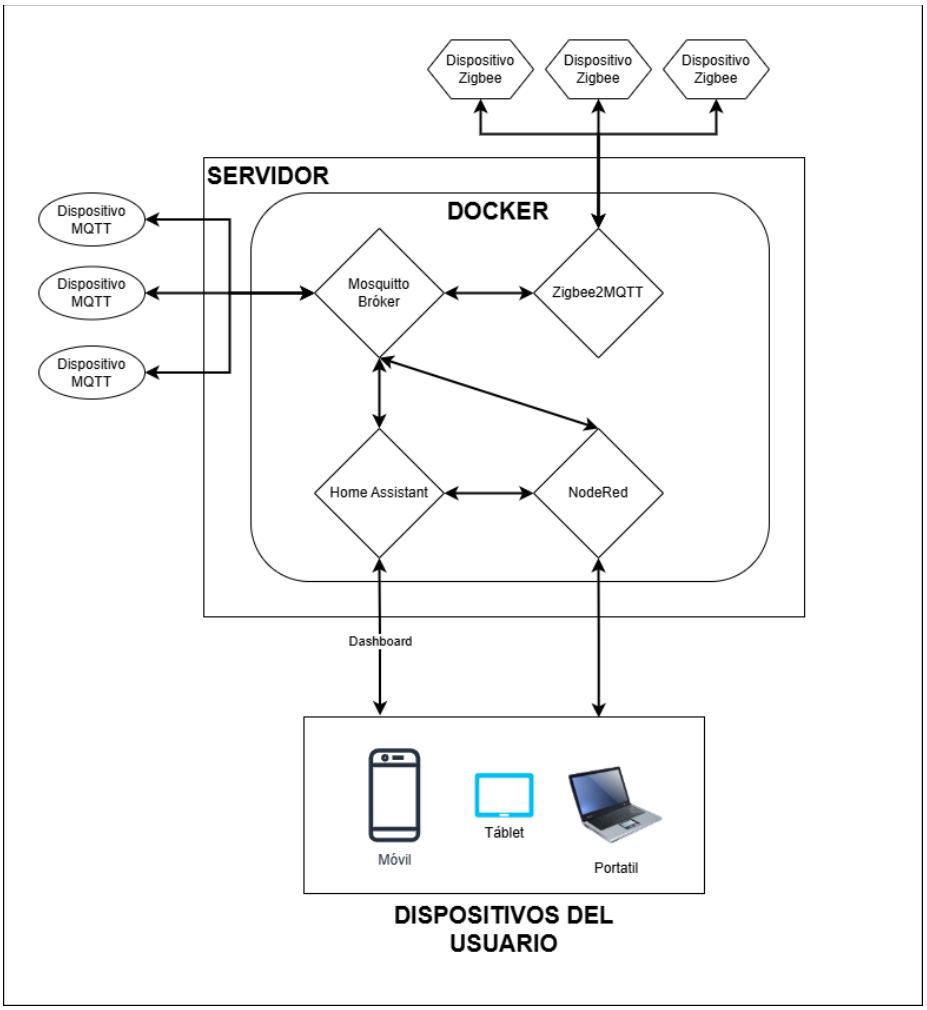

Figura 3.4. Esquema conexión servicios

En la Figura 3.4 se muestrán los servicios que se necesitan para desarrollar el sistema IoT. Los cuatro servicios son Mosquitto Broker, Zigbee2MQTT, Home Assistant y NodeRed. El primer servicio ejerce la función de *broker* en MQTT. Zigbee2MQTT se encarga de gestionar la red Zigbee donde se conectan los dispositivos IoT comerciales. Home Assistant es una plataforma IoT que se utilizará para gestionar los dispositivos de manera intuitiva a través de un panel de operación. Por último, en NodeRed se programarán las escenas necesarias para personalizar el sistema IoT a las características del usuario y del entorno.

La Figura 3.4 muestra como se comunican los servicios entre sí y con el medio. Se puede observar la pecularidad de que los dispositivos Zigbee no se comunican directamente con el servicio de Home Assistan o NodeRed. Esto es debido a que para establecer la comunicación es necesario realizar una conversión de protocolos. Primero, los dispositivos Zigbee envían sus datos a Zigbee2MQTT, que actua de puente, traduciendo los mensajes Zigbee al protocolo MQTT. Posteriormente, son publicados en el *broker* Mosquitto, que se encarga de gestionar y distribuir la información a los suscriptores.

El sistema planteado integra de manera eficiente los servicios que se mencionarán en este capítulo, permitiendo una comunicación fluida entre los dispositivos y garantizando un control centralizado. La arquitectura implementada está basada en el protocolo MQTT, asegurando una comunicación flexible y escalable entre los distintos componentes. Asimismo, el sistema ofrece posibilidades de ampliación, permitiendo la incorporación de nuevos servicios y funcionalidades. Esto lo convierte en una solución adaptable a futuras necesidades y escalable para proyectos más complejos de automatización del hogar o la industria.

3.3.2. Software de nodos y escenas

En primer lugar, vamos a describir el servicio seleccionado para la programación de los nodos y las escenas. La primera plataforma es NodeRed, una plataforma muy intuitiva de desarrollo basada en flujos, lo que nos va a permitir procesar la información recopilada por los sensores y procesarla a través de los flujos compuestos por los diferentes nodos para posteriormente enviar la orden a los actuadores.

Un **flujo** en Node-RED es una secuencia estructurada de nodos conectados entre sí que define la lógica de funcionamiento de la escena. Cada flujo representa un conjunto de operaciones que se ejecutan de manera encadenada: desde la recepción de datos, pasando por su análisis y evaluación, hasta la ejecución de acciones concretas en función de esa información. En el contexto de este proyecto, los flujos permiten, por ejemplo, recibir datos provenientes de sensores a través del *broker* MQTT, interpretar dichos datos mediante condiciones lógicas, y finalmente enviar instrucciones al sistema para actuar sobre los dispositivos conectados.

Los **nodos** en Node-RED son bloques funcionales preconfigurados que representan acciones o eventos dentro de un flujo. Cada nodo tiene una función específica: puede ser un nodo de entrada (como recibir datos de MQTT), de procesamiento (como transformar datos, filtrar o aplicar lógica), o de salida (como enviar un mensaje o activar un actuador). Estos nodos se arrastran desde una **paleta lateral**, que contiene una colección organizada por categorías, y se colocan en el espacio de trabajo para construir flujos lógicos. Cada nodo tiene propiedades configurables y puede conectarse con otros nodos mediante enlaces visuales, facilitando la programación sin necesidad de escribir código en la mayoría de los casos.

Las principales características de Node-RED son:

- **Interfaz gráfica** intuitiva que permite crear flujos de trabajo conectando nodos en un editor *web* (ver Figura 3.5).
- Está basado en **Node.js** [18], es un entorno de ejecución de JavaScript que permite ejecutar código JavaScript fuera del navegador, es decir, en el servidor.
- **Gran compatibilidad**. Permite la integración de protocolos como MQTT, HTTP, Websockets, Base de datos (SQL y NonSQL) y muchos otros servicios, gracias a su gran comunidad y desarrollo de paletas.
- **Comunidad activa**. Posee una comunidad muy grande que aportan flujos, nodos y asistencia para ampliar los usos de la herramienta, especialmente en su foro [19].

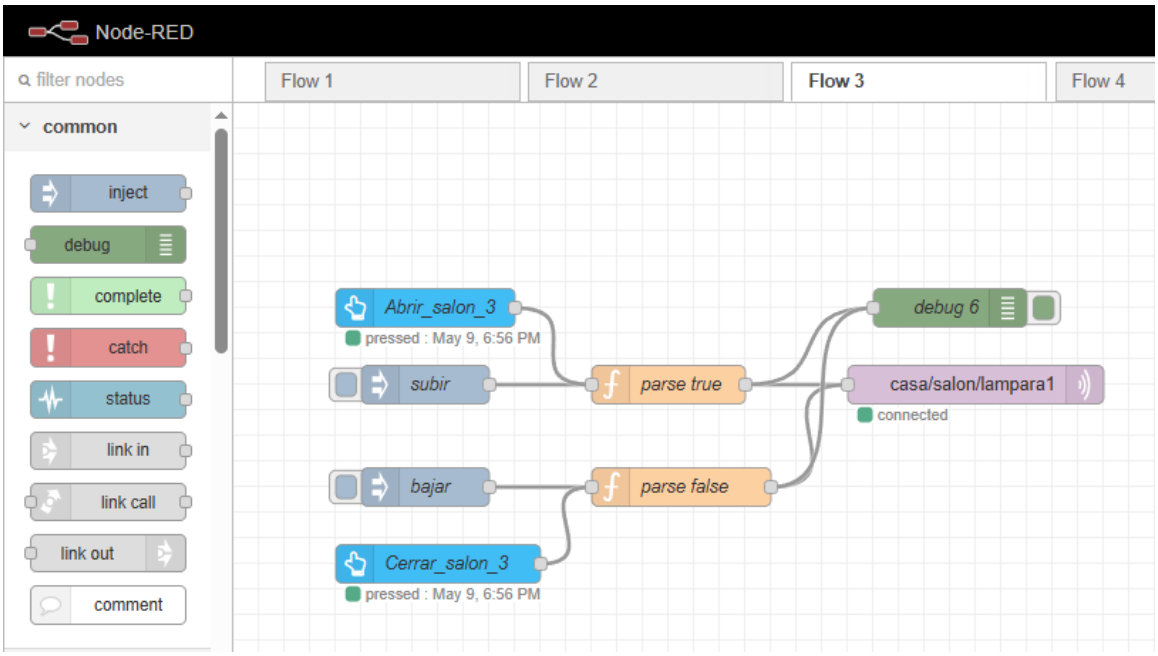

Figura 3.5. Flujo NodeRed

3.3.3. Panel de usuario

Un sistema IoT debe proporcionar al usuario un panel de operación donde pueda visualizar en tiempo real el estado de los dispositivos conectados, supervisar datos recolectados por los sensores, recibir alertas ante eventos importantes, y controlar de forma remota las acciones del sistema según sus necesidades.

Para crear un panel de usuario existen varias opciones: se pueden usar plataformas IoT con paneles de usuario ya definidos o se puede desarrollar un panel de usuario desde cero.

Para crear un panel de usuario desde cero se puede desarrollar desde el lenguaje de programación Python, por ejemplo, un lenguaje de alto nivel con una gran variedad de librerías. Las librerías básicas que se necesitan son:

- **Reflex**: Permite crear una interfaz *web* desde donde se podría interactuar con los diferentes dispositivos.
- **Paho-MQTT**: Permite suscribirse y publicar en los tópicos de MQTT.
- **Zigpy**: Es un *stack* de *software* de Zigbee para Python, permite la comunicación con los dispositivos a través de la red Zigbee.
- **PyModbus**: Para las comunicaciones a través de Modbus. Esta librería se usaría más para entornos industriales.
- **Xknx**: Una librería que permite la comunicación por KNX, útil para los sistemas domóticos, aunque precisan de una instalación cableada.

Estas son las librerías más básicas y útiles para el sistema IoT, sin embargo, existen muchas más librerías que pueden ayudar a la conectividad del servicio a través de otros protocolos de comunicación. En este proyecto, sin embargo, solo se usarán los dos primeros protocolos de comunicación.

La opción de usar Python se descartó por la complejidad para montar el sistema, aunque se está trabajando actualmente para implantar una interfaz gráfica personalizada por el

estudiante, y así no depender de ningunas plataformas IoT ya existente, aunque sea *Open Source*.

La opción que se ha seleccionado para este TFM ha sido utilizar la plataforma HA (*Home Assistant*) [20], una plataforma de automatización de código abierto, que es compatible con una gran variedad de protocolos de comunicación, entre los que se encuentran: MQTT, Zigbee, Z-wave, Modbus, Wifi, Sockets, KNX, Bluetooth entre otros.

Sus principales características son:

- No dependen de un servidor ajeno en la nube. Permite un control de los dispositivos en local.
- Compatibilidad con más de 1000 dispositivos, KNX, Zigbee, MQTT, Alexa, Google Home, etc.
- Permite automatizaciones complejas de escenas usando ficheros "*.yml*" o NodeRed.
- Posee una gran variedad de *dashboards* con diferentes temáticas que se adaptan al entorno donde se quiere instalar, como la mostrada en la Figura 3.6.

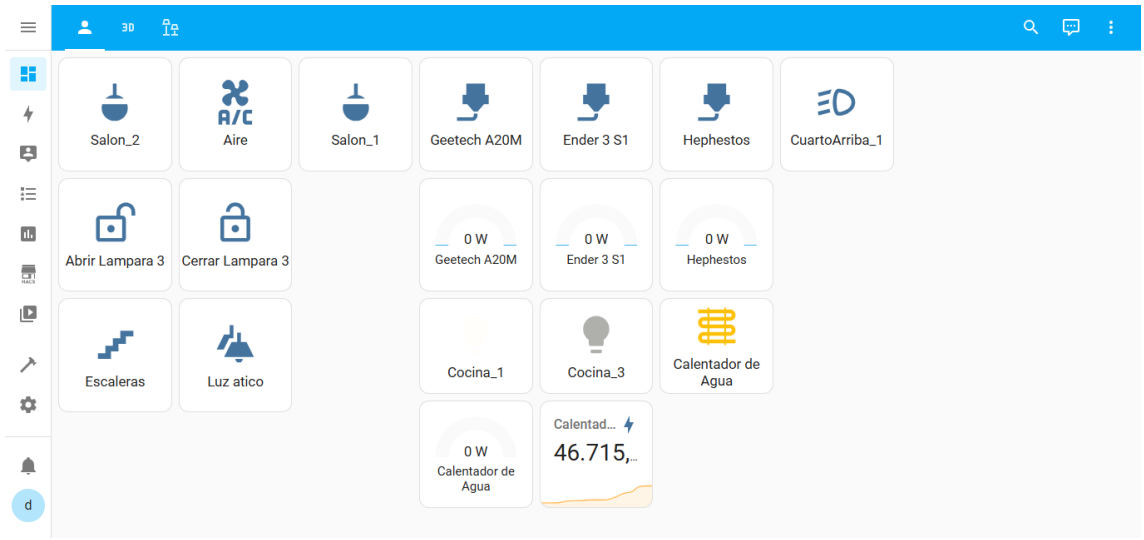

Figura 3.6. Home Assistant

3.3.4. Protocolos de comunicación

En esta sección se abordan los protocolos de comunicación que se han seleccionado para que los dispositivos puedan comunicarse entre sí y con el servidor.

Como se ha mencionado anteriormente, se ha escogido MQTT (*Message Queuing Telemetry Transport*) como protocolo principal. Este protocolo de comunicación inalámbrico está muy expandido y se usa bastante en el mundo IoT, gracias a que MQTT permite el intercambio de información entre los dispositivos IoT o IIoT, dispositivos embebidos, sensores, PLCs, etc. [23].

La principal ventaja de MQTT radica en ser un protocolo de mensajería liviano, ideal para su implementación en dispositivos con recursos limitados.

Está basado en el protocolo TCP/IP con una topología de Publicación/Suscripción y aglutina dos componentes: clientes y *brokers* (ver Figura 3.7) [24]. El *broker* es el servidor con el que se comunican los clientes (pueden existir varios *brokers*) y se encarga de recibir los datos de los clientes para, posteriormente, enviar esta información a otros clientes o respondiéndole al

mismo. En este sentido, MQTT es un protocolo basado en eventos donde no hay transmisión de datos periódica, sino que el cliente sólo publica datos cuando tiene información para enviar y el *broker* solo responde cuando llegan nuevos datos.

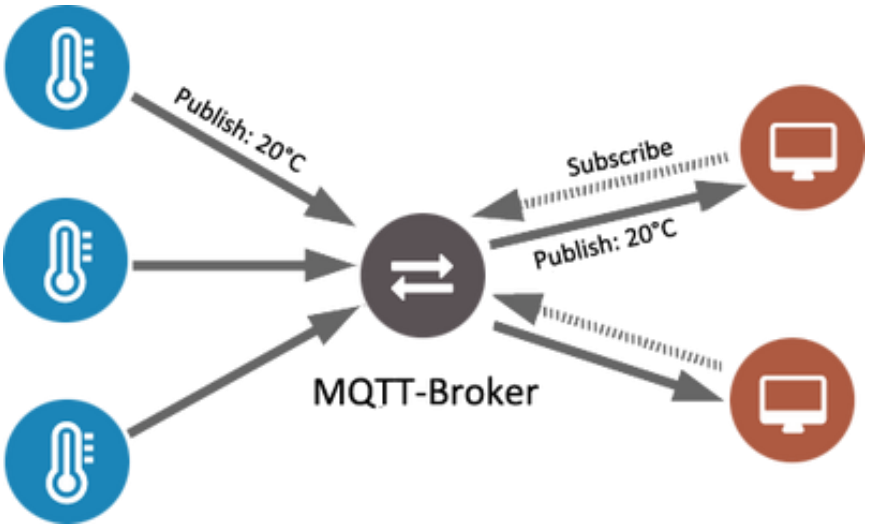

Figura 3.7. Funcionamiento MQTT

La transmisión de la información se realiza mediante *topics*, canales de comunicación que se organizan en una estructura jerárquica utilizando la barra (/) como delimitador. Esta jerarquía funciona de manera similar a un árbol de directorios de un ordenador. Por ejemplo, una estructura como "*escaleras/sensores/presencia"* permite a un suscriptor filtrar y recibir los datos de los clientes que se encargan de detectar presencia en las escaleras. Si se quiere obtener toda la información de los sensores que se encuentran en la escalera solo hay que suscribirse a "*escaleras/sensores*".

3.3.5. Red de bajo ancho de banda

Además de la red de comunicaciones con MQTT, en este TFM utilizamos una red de bajo ancho de banda para conectar algunos de los sensores que hemos utilizado en el sistema IoT. El ancho de banda viene definido como la cantidad de datos que se pueden transferir entre dos dispositivos de una red en un tiempo determinado [26]. Por tanto, una red de bajo ancho de banda es aquella en la que la capacidad de transmisión está limitada, normalmente en un rango bajo de kilobits por segundo (kbps). Estas redes están diseñadas para tener un consumo energético bajo y para transmitir datos de la manera más eficiente.

Las aplicaciones más comunes de estas redes son:

- Internet de las Cosas (IoT).
- Domótica.
- Redes de sensores inalámbricos.
- Sistemas de monitorización y control.

Estas redes suelen ser instaladas en áreas extensas, como zonas industriales, entornos agrícolas, ciudades inteligentes, redes de monitoreo ambiental, ya que estas redes ofrecen una gran cobertura, permiten la conexión de cientos de dispositivos sin colapsar y con un consumo energético bajo.

Algunos ejemplos de estas redes son:

- **LoRaWAN** (0.3-50 kbps): Está diseñada para transmisión de datos a largo alcance, se suele usar en agricultura inteligente, ciudades inteligentes, en general se usa en zonas extensas permitiendo la conexión entre los dispositivos a larga distancia con un consumo energético bajo [27].
- **Zigbee** (hasta 250 kbps): Se utiliza principalmente en instalaciones domóticas y automatizaciones del hogar. En la actualidad existen en el mercado una gran variedad de dispositivos (sensores, actuadores, luces, sistemas de seguridad) que facilitan el control y la monitorización de la vivienda de manera eficiente.[28]
- **Sigfox** (hasta 100 bps): Está enfocada a aplicaciones que requieren la transmisión de pequeños paquetes. Sigfox se usa para tareas de monitoreo remoto, gestión de archivos y sensores industriales.
- **Wi-Sun** (50-300 kbps): Ha sido diseñada para aplicaciones IoT en áreas extensas, como redes inteligentes y ciudades inteligentes. Wi-Sun es adecada para aplicaciones que requieran una cominicación estable, robusta y eficiente en áreas geográficamente amplias.

En resumen, estas redes de bajo ancho de banda son esenciales para diversas aplicaciones en el ecosistema IoT. En este proyecto la red que se utilizará será Zigbee, porque es una red que está pensada para instalaciones pequeñas. Lo bueno del sistema IoT es la escalabilidad, a futuro se podrá instalar nuevas redes de bajo ancho de banda para sufragar las necesidades del entorno. En este caso, el entorno industrial es pequeño y no se requiere el uso de redes de monitorización con una gran cobertura, ya que el entorno simulado se encuentra en un aula.

3.4. Hardware necesario

En este subapartado se detalla el *hardware* necesario para que todo el sistema y la red IoT funcione de manera fluida y sin fallos.

El primer dispositivo necesario es un servidor, un dispositivo que haga de “nube” y que pueda ejecutar todos los servicios que componen el sistema IoT. Como se ha mencionado en apartados anteriores, no es necesario un equipo con mucha potencia. Es posible ejecutar los servicios en un servidor NAS doméstico, una Raspberry Pi o un ordenador.

La selección del dispositivo con el rol de servidor dependerá de las necesidades del entorno, necesitando más potencia a medida que se incorporen nuevos servicios. Por ejemplo, en un domicilio que tenga una sola planta con una Raspberry Pi bastará. Sin embargo, en una empresa o un domicilio más grande (chalets de varias plantas, por ejemplo) hará falta un equipo más potente. En este TFM, haremos uso de un ordenador con bajos recursos para el entorno doméstico y de una Raspberry Pi para el aula. Una vez seleccionado el servidor, necesitaremos una antena capaz de transmitir la información con el ancho de banda seleccionado. En el mercado existen varias opciones, como, por ejemplo:

- ZZH Multiprotocol RF Stick (CC2652R1 - External Antenna) [30]
- SONOFF Zigbee 3.0 USB Dongle Plus ZBDongle-P [31]
- LAUNCHXL-CC1352P [32]

Además, existen kits solo con la antena o kits completos de desarrollo que integran microcontroladores que permite crear la red sin necesidad de un equipo nodo. Estos últimos kits

están optimizados para la creación de este tipo de redes, pero no están pensadas para procesar datos y montar sistemas de cierta complejidad.

Las dos antenas primeras antenas: ZZH y la de SONOFF son las opciones más viables, ya que se pueden conectar a equipos capaces de procesar los datos. Entre estas dos opciones, se ha seleccionado la antena de SONOFF porque es una de las opciones más actualizadas y está optimizada para las redes Zigbee, una de las redes que se usarán para comunicarse con los dispositivos en este TFM (véase la Sección 3.3.5).

Por último, se necesitarán los dispositivos inteligentes que interactuarán con el medio, es decir, los dispositivos IoT. Estos dispositivos son los sensores, actuadores, reguladores, electrodomésticos, etc., los cuales proporcionarán datos y actuarán según las necesidades del entorno. En el sistema IoT que se ha desarrollado solo se hará uso de sensores y actuadores, tal y como se detalla en el Apartado 3.5 y 3.6.

En resumen, el hardware necesario para un sistema IoT abarca varios componentes para que su funcionamiento sea eficiente y efectivo. Primero, un servidor adecuado, ya sea una nube o un equipo físico como una Raspberry Pi o un servidor NAS doméstico. Las antenas son esenciales para la transmisión de datos y para la comunicación con los dispositivos. La selección de la antena debe basarse en la compatibilidad y optimización para las redes que se utilizarán, como Zigbee en este caso. Finalmente, los dispositivos IoT, incluyendo sensores y actuadores, interactúan directamente con el entorno. Todos estos componentes son fundamentales para que el sistema IoT pueda operar de manera fluida y eficiente, optimizando recursos y mejorando la gestión de procesos en diversos entornos.

3.5. Sensores

Los sensores son los dispositivos que detectan y recopilan información del entorno. Esta información se convierte en señales eléctricas que posteriormente se transmiten a través de la red, ya sea a través de Zigbee o de MQTT, directamente.

En el mercado existen una gran variedad de estos dispositivos, ya sean sensores de temperatura, presencia, lluvia, interruptores, etc. Estos sensores cumplen su función de manera eficiente en las diferentes aplicaciones, desde domótica hasta la industria. Además, en el marco de este proyecto es fundamental que el sistema IoT también sea compatible con sensores personalizados o sensores DIY, ya que tienen la ventaja de ser más económicos y se pueden ajustar más a las necesidades del entorno, siendo más adaptables y personalizables. Para crear estos sensores, habitualmente se usan microcontroladores como el ESP8266, ESP32 o Arduino, que pueden comunicarse por protocolos como MQTT de manera sencilla.

Los sensores se pueden clasificar en digitales y analógicos, según el tipo de señal que generen: los sensores digitales envían señales discretas, representados por valores binarios (0 y 1) y los sensores analógicos emiten una señal continua que puede tomar un valor cualquiera dentro de un rango de valores determinado.

En este TFM hemos utilizado los siguientes sensores en los dos ambientes:

- **Doméstico**: hemos desplegado sensores de luz, gas, presencia, ultrasónicos, temperatura y humedad combinando dispositivos comerciales con otros DIY diseñados y creados específicamente para este TFM.

- **Industrial**: en el laboratorio de automatización, la captura de información de la planta se realiza por parte del PLC y éste se comunica mediante MQTT con el servidor del sistema IoT. Sólo se ha añadido un sensor de medición de consumo eléctrico que forma parte de un enchufe inteligente.

Cabe destacar que la mayoría de los sensores utilizados en este TFM estarán basados en placas de microcontrolador, con el objetivo de reducir el coste del despliegue del sistema IoT. La programación de los sensores, por parte del alumno, se puede encontrar en el siguiente enlace: https://github.com/PacoLC/IoT_TFM/tree/main/Microcontroladores. A continuación, se analizarán los sensores comerciales y sus análogos usando microcontroladores

3.5.1. Luz

Los sensores de luz miden la iluminancia, que es la cantidad de luz que incide sobre una superficie.

De entre las distintas opciones que están disponibles en el mercado, hemos optado por desarrollar un dispositivo DIY basado en el sensor TSL2561 [35], lo que nos permite obtener una alternativa versátil y económica a los sensores comerciales. A modo de ejemplo, el sensor comercial Tuya Zigbee Luminance [36] tiene un coste habitual de 8€ mientras que el desarrollado para este TFM tiene un coste total de 2€. El TSL2561 es compatible con microcontroladores como Arduino y ESP, lo que permite integrarlo fácilmente en sistemas personalizados. Su principal ventaja es la posibilidad de conectar múltiples sensores a un mismo microcontrolador, permitiendo la creación de un *hub* de sensores capaz de recopilar diferentes datos dentro de una habitación. Esto facilita la gestión centralizada de la información y mejora la eficiencia del sistema IoT.

Características	
Voltaje de operación	2'7 V a 3'6 V
Consumo corriente	0'6 mA
Rango de medición	0'1 lux hasta 40.000 lux
Sensibilidad	Similar al ojo humano con detección de luz infraroja (IR)
Salida	Valor de la medición a través de I2C
Resolución	16 bits

Tabla 3.1. Características sensor TSL2561

En la Figura 3.8 se muestra el sensor TSL2561, su conexión con una placa Arduino UNO y una imagen del sensor DIY creado.

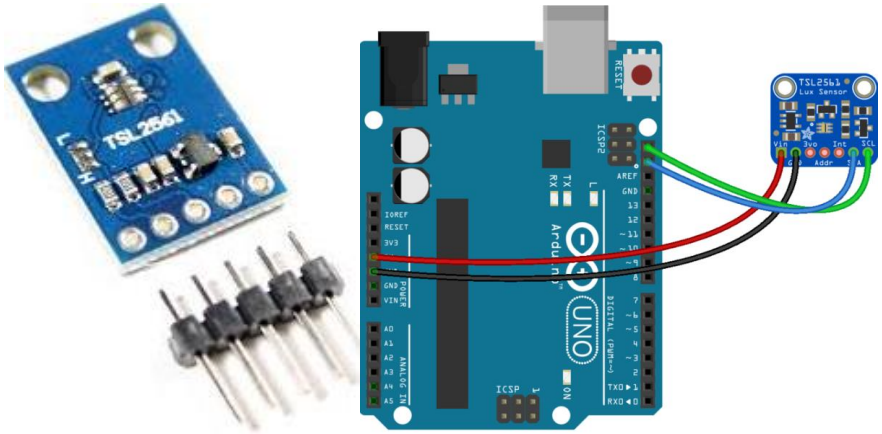

Figura 3.8. Sensor TSL2561

Figura 3.9. Hub con sensor TSL2561

3.5.2. Gas

Los sensores de gas detectan gases inflamables y humos, estos sensores suelen utilizarse para detectar fugas de gas, aumentando la seguridad en el hogar o en entornos industriales.

De entre las distintas opciones que están disponibles en el mercado, hemos optado por desarrollar un dispositivo DIY basado en el sensor MQ2 [38], lo que nos permite obtener una alternativa versátil y económica a los sensores comerciales. A modo de ejemplo, el sensor comercial Sensor Aqara JT-BZ-01AQ/A [39] tiene un coste habitual de 30€ mientras que el desarrollado para este TFM tiene un coste total de 1€.

El sensor Aqara detecta diversos gases inflamables y tiene una alarma sonora para alertar a las personas de una exposición a un gas inflamable y/o tóxico. El problema de estos sensores son el precio. Por esta razón, se ha buscado una solución con un microcontrolador, el sensor MQ-2.

Los sensores MQ son dispositivos capaces de detectar componentes químicos en el aire. Cada modelo de la familia MQ está pensada para detectar una o más sustancias, como son los gases inflamables, calidad del aire o detección de alcohol en aire respirado. El sensor que se ha utilizado detecta únicamente humos, gases inflamables y alcohol.

Características	
Voltaje de operación	5 V
Consumo corriente	150 mA
Gases detectados	GLP, Butano, Propano, Metano, Alcohol, Hidrógeno y Humo
Salida analógica	Voltaje proporcional a la concentración de gas
Salida digital	Indica presencia de gas
Sensibilidad	Ajustable
Tiempo de respuesta	<10 s

Tabla 3.2. Características sensor MQ-MQ2

Figura 3.10. MQ-MQ2 Gas Sensor

3.5.3. Presencia

Los sensores de presencia conocidos también como sensores de movimiento. Estos sensores detectan el movimiento de personas, animales u objetos en un área específica. Se suelen utilizar en aplicaciones como alarmas de seguridad e iluminación automática, entre otras.

Los sensores de movimientos se clasifican según su método de detección, véase los diferentes tipos de sensores en la Tabla 3.3.

Tipo de Sensor	Detecta	Ventajas	Desventajas
PIR (Infrarrojo Pasivo)	Cuerpos calientes en movimiento	Bajo consumo y barato	No detecta los cuerpos estáticos
Microondas (Radar Doppler)	Movimiento general	Mayor alcance y puede atravesar paredes	Mayor consumo y más caros
Ultrasónico	Movimiento basado en el eco del sonido	Preciso y no necesita luz	Sensible a materiales que absorben el sonido

Tabla 3.3. Clasificación de sensores de movimiento

Los sensores más comunes en el mercado son los sensores PIR, ya que son los más baratos y los que menor consumo tienen. Por ejemplo, el sensor ZG-204ZM [41] es un sensor PIR de la marca Tuya. Es económico y es fácil de integrar en la red. Sin embargo, siguiendo la metodología de los subapartados anteriores, se ha usado un sensor PIR DIY basado en microcontrolador.

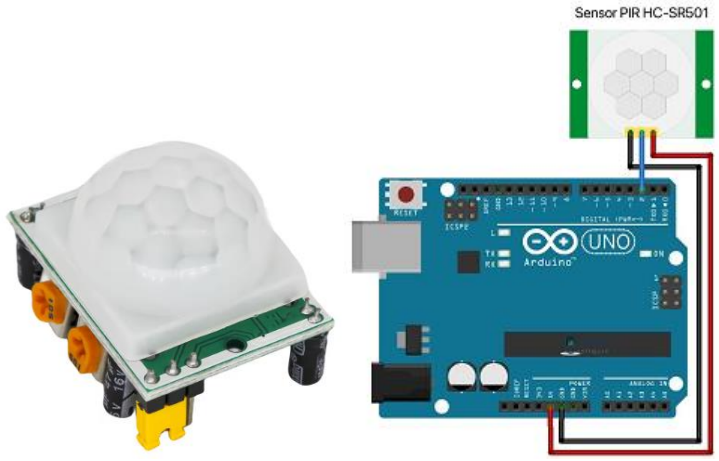

Figura 3.11. Sensor HC-SR501

El sensor HC-SR501 [42] es el sensor para microcontroladores más usado para la detección de movimiento. Tiene 3 conexiones: alimentación, tierra y la salida digital. Este sensor tiene dos modos de disparo: Se puede emitir un pulso único cuando se detecta el movimiento o puede mandar constantemente pulsos hasta que deje de detectar el movimiento, esta función es bastante practica para aplicaciones de domótica y sistemas de alarmas.

Además, el sensor tiene dos potenciómetros donde se puede ajustar el rango de detección y el tiempo de retardo. Véase en la Tabla 3.4 las características de este sensor.

Características	
Voltaje de operación	4'5 V a 20 V DC
Consumo corriente	50 µA
Distancia de detección	3 a 7 m (ajustables)
Ángulo de detección	120º
Tiempo de retardo	0'3 a 300 s (ajustable)
Modo de disparo	Repetitivo (H)
	Único (L)

Tabla 3.4. Características sensor HC-SR501

Figura 3.12. Hub con sensor HC-SR501

3.5.4. Temperatura y Humedad

En los sistemas IoT los sensores de temperatura y humedad juegan un papel fundamental para el control climático de un área. Estos sensores se emplean en hogares y en industrias. En hogares estos sensores se integran con los sistemas de calefacción y aire acondicionado. En las industrias los sensores de temperatura y humedad se integran con el sistema HVAC (*Heating, Ventilation and Air Conditioning*).

En el mercado existen muchos proveedores con este tipo de sensores, debido a que son muy comunes y usados. Una marca conocida es Aqara, se ha hablado de esta marca en otros sensores. Aqara tiene varios sensores de temperatura como son el WSDCGQ11LM [43] y el WSDCGQ12LM [44]. Ambos son sensores que funcionan con pilas y son fáciles de integrar con Zigbee.

Un sensor de temperatura y humedad para microcontroladores es el DHT11 [45]. Este sensor de temperatura y humedad es el más común y usado por la comunidad de personas que utilizan microcontroladores. Es un sensor barato y pequeño, fácil de instalar en microcontroladores como Arduino y ESP. Otro sensor conocido en el mundo DIY es el DHT22, aunque es una opción más costosa ya que tiene mejores prestaciones. Sim embargo, en este

proyecto no se requiere una precisión excesiva por lo tanto se ha decidido usar el sensor DHT11. La programación, instalación y conexión de ambos sensores es similar.

Características	DHT11	DHT22
Voltaje de operación	3V a 5V	3V a 5V
Rango de temperatura	0ºC a 50ºC	-40ºC a 80ºC
Precisión temperatura	±2ºC	±0'5ºC
Rango de humedad	20% a 90%	0% a 100%
Precisión de humedad	±5%	±2%
Frecuencia de muestreo	1 Hz	0'5Hz
Tiempo de respuesta	1s	2s

Tabla 3.5. Características sensor DHT11 y DHT22

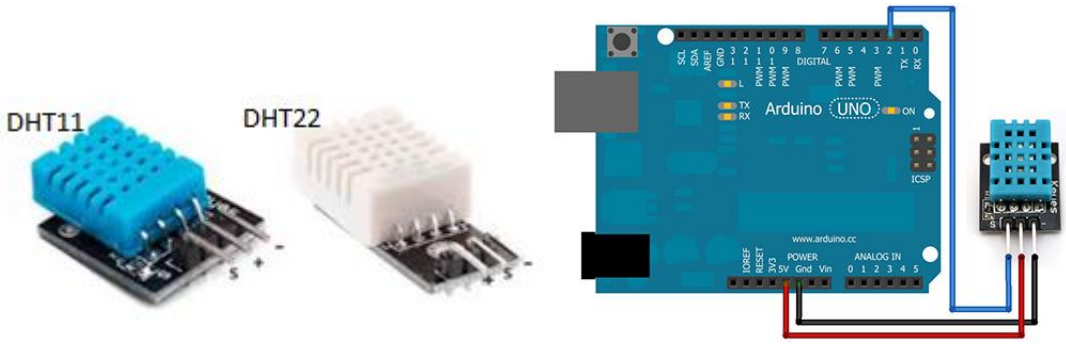

Figura 3.13. Sensor DHT11 y DHT22

Figura 3.14. Hub con sensor DHT11

3.5.5. Lluvia

Los sensores de lluvia son menos comunes en el mercado, aunque tienen bastante utilidad. Por ejemplo, estos sensores detectan lluvias inesperadas que pueden perjudicar una colada que está tendida en el exterior.

De entre los dispositivos comerciales, Tuya ha desarrollado el sensor RB-SRAIN01 [46], un sensor de lluvia que funciona con energía solar y además detecta la cantidad de luz que hay en el exterior, detectando así las nubes. La información que envía este sensor es la iluminancia, iluminancia media de los últimos 20 minutos, iluminancia máxima del día, lluvia, intensidad de lluvia, nivel de batería y si hace falta limpiar el sensor.

Para los sistemas basados en microcontrolador existe un sensor conocido como *water level sensor* [47], bastante común en la comunidad de microcontroladores, que es capaz de detectar el nivel de agua que hay en su superficie y por ende se podría detectar la lluvia cuando varias gotas caigan en este sensor.

Es un sensor simple que tiene 3 conexiones: alimentación, tierra y datos. El sensor está formado por una malla de cobre que actúa como una resistencia variable, cuando el agua toca la malla la resistencia eléctrica del sensor disminuye detectando así la lluvia.

Características	
Voltaje de operación	3'3 V a 5 V
Consumo corriente	20 mA
Salida	Analógica, valor en función de la resistencia variable
Tiempo de respuesta	<1s

Tabla 3.5. Características sensor lluvia

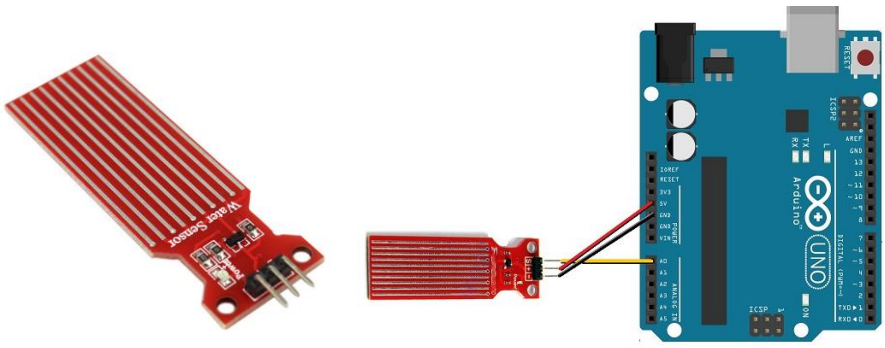

Figura 3.15. Sensor lluvia

Figura 3.16. Hub con sensor lluvia

3.5.6. Medidor consumo eléctrico

Dentro de los dispositivos utilizados en este proyecto se encuentra un enchufe inteligente con capacidad de medición de potencia, el cual cumple una doble función en el sistema: **actuador** y **sensor**.

Este tipo de enchufes comerciales está equipado con sensores que miden parámetros eléctricos como la potencia instantánea (W), el consumo acumulado (kWh), e incluso en algunos modelos, el voltaje (V) y la intensidad de corriente (A). Estos valores son recogidos por el dispositivo y publicados a través del protocolo MQTT, lo que permite su integración directa en plataformas como Home Assistant o Node-RED, desde donde pueden visualizarse, registrarse y emplearse para generar automatismos basados en el consumo energético.

Gracias a esta funcionalidad, el sistema no solo puede controlar dispositivos de forma remota, sino también obtener métricas que ayudan a evaluar la eficiencia energética, detectar posibles fallos (como un dispositivo que consume más de lo esperado), o aplicar lógicas como el apagado automático tras superar un determinado umbral de potencia. De este modo, el enchufe inteligente se convierte en un elemento híbrido muy valioso dentro del ecosistema IoT propuesto.

Figura 3.17. Medidor consumo eléctrico

3.6. Actuadores

Los actuadores son los dispositivos encargados de ejecutar acciones físicas en respuesta de las señales eléctricas enviadas por el sistema IoT. Estos dispositivos permiten la automatización de tareas, como el control de las luces, transformando señales digitales en movimientos mecánicos o cambios eléctricos, permitiendo modificar e interactuar con el entorno.

En el mercado existe una gran variedad de actuadores, como relés, válvulas, bombillas, solenoides, enchufes inteligentes, entre otros. De nuevo, en el contexto de este proyecto, es imprescindible que el sistema IoT sea compatible con actuadores personalizados DIY.

Los actuadores DIY ofrecen la ventaja de ser más económicos y adaptables a necesidades específicas. Al igual que los sensores DIY, para su implementación, se emplean microcontroladores como ESP8266, ESP32 o Arduino, que pueden comunicarse mediante protocolos como MQTT para recibir órdenes y ejecutarlas en tiempo real.

Los actuadores pueden clasificarse en dos grandes grupos según su tipo de operación:

- **Actuadores digitales**: Funcionan con señales binarias (encendido/apagado), como los relés que controlan luces o electrodomésticos.
- **Actuadores analógicos**: Permiten variaciones graduales, como los motores de velocidad variable o servomecanismos, que ajustan su posición en función de la señal recibida.

En este TFM hemos utilizado los siguientes actuadores en los dos ambientes:

- **Doméstico**: hemos instalado relés, bombillas, pulsadores, emisores infrarrojos y enchufes inteligentes.
- **Industrial**: en el laboratorio de automatización, los actuadores que el sistema controla son los propios de la estación. No ha requerido una instalación tan amplia como en el entorno de doméstico, pero si se ha colocado un enchufe inteligente que además mide el consumo eléctrico, tal y como explicaremos posteriormente.

3.6.1. Relés

Los relés son actuadores digitales que dependen de una señal binaria. Se utilizan para dar alimentación a dispositivos, generalmente se usan para alimentar instalaciones lumínicas.

Estos dispositivos IoT se suelen conectar detrás de un interruptor que controla las luces, funcionando como un interruptor extra que se puede activar de forma remota o se puede activar de forma automática por una orden del servidor. Existen dos tipos de relés que se pueden clasificar según su alimentación:

- **Con cable neutro**. Necesitan conectarse a la red directamente y a la señal del último interruptor del cableado.
- **Sin cable neutro**. Se conectan a cualquier interruptor y el relé manipula la señal de este interruptor. Son algo más complejos y propensos a fallos.

Los relés que se han seleccionado para la instalación han sido los relés que tienen cable neutro. Son más duraderos y su instalación no es complicada. En el mercado existe una gran variedad de opciones como los relés de SON/OFF, o Tuya, entre otras opciones.

Para este trabajo, se ha seleccionado el relé AVATTO [48] de Tuya, un relé de pequeñas dimensiones y con un consumo muy pequeño. Además, existen diferentes modelos si se quieren

controlar más de una línea lumínica a la vez. En el caso de este proyecto se ha escogido solo de un canal, es decir, que solo puede controlar una línea a la vez.

A continuación, se explica el funcionamiento del relé. El interruptor físico y las luminarias no están conectadas directamente por un medio físico, es decir, los interruptores se conectan al relé y las luces también se conectan al relé. El relé inteligente detecta las órdenes del usuario mediante el cambio de estado del interruptor físico, y es el encargado de ejecutar la acción correspondiente, ya sea el encendido o el apagado del sistema de iluminación. Este dispositivo actúa como intermediario entre el control manual tradicional y el sistema de automatización, permitiendo conservar el uso del interruptor convencional mientras se habilita el control remoto a través de la plataforma domótica (véase la explicación en más detalle en la sección 4.3).

Figura 3.18. Relé Tuya Avatto

Como en apartados anteriores, se propone diseñar un dispositivo DIY que cumpla con las necesidades. Se podría realizar con un microcontrolador ESP o Arduino con un relé de 230V. El inconveniente principal de diseñar un dispositivo DIY es la integración con los interruptores, ya que se tendría que utilizar algún dispositivo capaz de detectar la señal de 230V en alterna como un valor digital. Esta tarea es algo más compleja y por esta razón se decidió utilizar dispositivos del mercado.

3.6.2. Bombillas

Las bombillas inteligentes son bombillas led que se pueden controlar de forma remota a través de protocolos de red como Zigbee o Wifi.

Son muy utilizadas en la actualidad y se caracterizan por poder controlar diferentes parámetros como la temperatura, intensidad y algunas pueden cambiar de color.

Figura 3.19. Bombilla inteligente

Las bombillas inteligentes tienen un problema y es que solo funcionan cuando están energizadas, es decir, están conectadas a la red. Por ejemplo, si se coloca una bombilla inteligente en una habitación y se apaga mediante el interruptor físico, el usuario no podrá volverla a encender por medios remotos, tendrá que interactuar de nuevo con el interruptor para encender la bombilla.

Este problema se puede solucionar de dos formas, usando un relé del apartado anterior o cambiar el interruptor para que siempre esté la bombilla conectada y el interruptor interactúe con el servidor a través de la red Zigbee o Wifi.

La opción más simple es usar el relé para evitar modificar los interruptores de los entornos de pruebas. Por lo tanto, las bombillas inteligentes no son realmente útiles a menos que se quieran usar para controlar la intensidad, color y temperatura de las zonas donde se quieren instalar.

3.6.3. Pulsadores

Los pulsadores son dispositivos diseñados para pulsar botones, interruptores y activar mecanismos que requieren de una interacción manual.

Se suelen utilizar para automatizar acciones como la pulsación de botones sin realizar modificaciones complejas en la instalación. Por ejemplo, el encendido y apagado de un aire acondicionado centralizado que no tiene mando, se necesitaría ir a la zona donde está el control y accionar el botón de encendido/apagado. Con este dispositivo se puede interactuar con el controlador de forma remota.

En este proyecto se ha hecho uso del dispositivo Tuya Fingerbot Pusher [49], un dispositivo pequeño, fácil de integrar con la red Zigbee y no requiere de una instalación compleja, viene con pegatinas y se puede adherir a cualquier superficie sin problemas. La alimentación es a través de pilas.

Figura 3.20. Tuya Fingerbot Pusher

3.6.4. Emisor Infrarrojos

Los emisores de infrarrojos son dispositivos universales de control remoto mediante señales infrarrojas (IR).

Estos dispositivos conectan la tecnología de red Zigbee o Wifi con las señales IR, por lo que se pueden enviar señales IR a los electrodomésticos de forma remota. Esto es bastante útil para controlar televisiones, aires acondicionados, ventiladores, reproductores, entre otros, siempre y cuando estos dispositivos detecten señales infrarrojas.

En el mercado no existen muchas opciones, pero la empresa Tuya ha desarrollado un control IR universal Zigbee [50]. Es fácil de integrar con la red Zigbee, pero es algo complejo de configurar. Primero se tiene que escanear el mando del dispositivo que se quiere controlar, almacenar los códigos IR del mando y luego generar funciones que envíen esos códigos a través de señales IR. La alimentación del dispositivo es a través de un cable micro-usb.

Figura 3.21. Tuya Smart IR Remote Control

3.6.5. Enchufes

Los enchufes inteligentes son adaptadores que se conectan a una toma de corriente estándar y son una herramienta bastante usada en el mundo de la domótica. Permiten interactuar con dispositivos electrónicos conectados a ellos de forma remota.

Como se ha mencionado anteriormente los enchufes inteligentes se usan en muchas aplicaciones dentro del hogar, por ejemplo: controlar electrodomésticos (encendido y apagado), monitorizar consumos y energía de los aparatos que están conectados al enchufe.

En la actualidad existen muchas empresas que han desarrollado enchufes inteligentes, pero la más reconocida es la marca Tuya. Esta empresa ha sacado al mercado una gran variedad de enchufes inteligentes de diferentes amperajes. Los enchufes utilizados en la prueba de este proyecto son de esta marca.

Figura 3.22. Tuya Smart Plug

CAPÍTULO 4. INSTALACIÓN Y PRUEBAS EN UN ENTORNO DOMÉSTICO

4.1. Plan de trabajo

Esta sección detalla el plan de trabajo para la instalación de un sistema IoT en un entorno doméstico. Se presentan tanto la planificación teórica como la implementación práctica en la vivienda del estudiante.

1. **Planificación de la Instalación.**
   Antes de proceder con la instalación, es fundamental realizar un análisis del entorno y de las necesidades de la vivienda, para ello se siguen los siguientes pasos:

   - **Análisis del entorno**: Recopilación de características del espacio, dimensiones y ubicación.
   - **Identificación de necesidades**: Determinar qué dispositivos requieren automatización, como el encendido de luces, control del consumo energético y dispositivos eléctricos conectados, entre otros.
   - **Evaluación de la instalación eléctrica**: Identificación de modificaciones necesarias en la instalación eléctrica.
   - **Selección de la ubicación del servidor**: Determinar el mejor lugar para instalar el servidor IoT, que funcionará como nodo maestro de la red.
   - **Diseño de la distribución de dispositivos**: Planificación de la ubicación de sensores y actuadores para optimizar la cobertura y eficiencia del sistema.

2. **Características de la Vivienda**.
   La vivienda donde se realiza la instalación es un dúplex de 140 m², con paredes de pladur y una escalera en el salón. Está ubicada en Málaga y cuenta con las siguientes necesidades específicas:

   - **Control de iluminación**: Automatización de las luces del salón y la planta superior, además de encendido automático de la escalera.
   - **Gestión energética**: Monitoreo del consumo del calentador de agua, con posibilidad de integración de placas solares en el futuro.
   - **Control remoto de dispositivos**: Gestión de impresoras 3D, aire acondicionado y calefactor.

3. **Implementación del Sistema IoT**.
   Una vez planificada la instalación, se procede ejecutando las siguientes tareas:

   - **Instalación del servidor**: Configuración de un servidor con sistema operativo Linux, ubicado en la planta superior con conexión directa al *router* para asegurar la cobertura de la red Zigbee.
   - **Pruebas iniciales**: Conexión de dispositivos que no requieren modificaciones eléctricas para comprobar el correcto funcionamiento de la red.
   - **Instalación de sensores y actuadores**: Montaje de los dispositivos en sus ubicaciones planificadas, respetando las normas de seguridad establecidas.
   - **Modificaciones de la instalación eléctrica necesarias**: Adaptación de la instalación eléctrica para la integración de los actuadores remotos de las luces.

- **Integración de los dispositivos a la red IoT:** Configuración y conexión de todos los elementos al sistema central.
- **Programación y personalización**: Desarrollo del panel de usuario, definición de escenas y configuración de automatizaciones según las necesidades de la vivienda.
- **Pruebas finales y optimización**: Verificación del funcionamiento del sistema, asegurando una cobertura estable y eficiente.
- **Seguridad y mejoras**: Implementación de medidas de seguridad como VPN y gestor de tráfico de red para evitar accesos no autorizados y mejorar la privacidad en la navegación.

4. **Consideraciones**.
   Es fundamental que, en caso de que el servidor falle, el usuario pueda seguir interactuando con la vivienda de forma manual mediante los interruptores convencionales. Este procedimiento garantiza una instalación eficiente y segura del sistema IoT, proporcionando un entorno doméstico inteligente y optimizado para el usuario.

   Para finalizar, se elabora un apartado de conclusiones basado en las experiencias y aprendizajes obtenidos antes, durante y después de la instalación del sistema en la vivienda. Este procedimiento asegura una implementación eficiente y segura del sistema IoT en un entorno doméstico, proporcionando una plataforma de pruebas y aprendizaje para futuros desarrollos.

4.2. Necesidades de la vivienda

En este punto se detallan las necesidades de la vivienda y del estudiante. Después, se explicará brevemente como se suplirán todas las necesidades y se irán implantando en la vivienda de forma progresiva para no suponer una gran inversión al principio.

Antes de comenzar con las necesidades se ilustra un plano conceptual de la vivienda sin escalas.

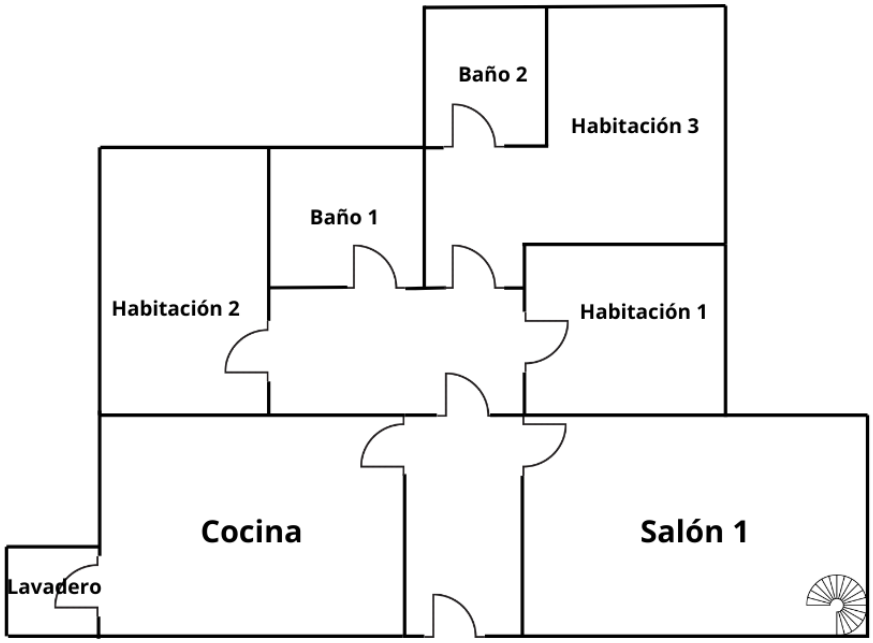

Figura 4.1. Plano conceptual vivienda planta inferior

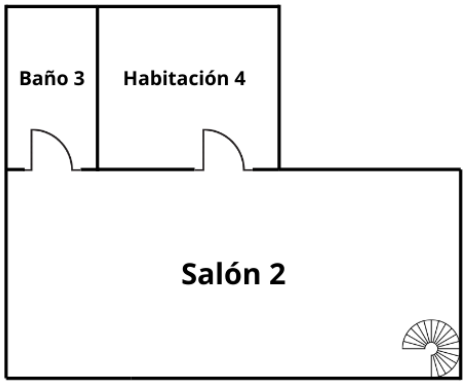

Figura 4.2. Plano conceptual vivienda planta superior

Como se puede observar en la vivienda existen cuatro habitaciones que se han enumerado de la 1 a la 4. Una cocina, dos salones, uno situado en la planta inferior y el segundo en la planta superior. Además, la vivienda cuenta con una escalera de caracol que comunica las dos plantas a través de los salones.

Las necesidades se van a dividir en dos ramas, necesidades funcionales y necesidades no funcionales. Las funcionales hacen referencia a las características del sistema domótico para

cumplir las expectativas y garantizar su correcto funcionamiento, las necesidades no funcionales representan los requerimientos generales de seguridad, compatibilidad y escalabilidad que no afecta directamente al desempeño del sistema.

Las necesidades funcionales:

- Iluminación Inteligente:
  - Control remoto de luces desde dispositivos móviles, tablets, ordenadores, entre otros.
  - Programación de escenarios de luces.
  - Instalación de sensores de movimientos para zonas de paso, como escaleras y pasillo.

- Climatización:
  - Control de las temperaturas de diferentes zonas.
  - Control de aire acondicionado y calefacción automática.

- Seguridad:
  - Sensores de apertura de puertas y ventanas.
  - Avisadores de lluvia.

- Entretenimiento y Confort:
  - Diseño de un *dashboard* simple y funcional.
  - Creación de escenas para la comodidad del estudiante. (Control de temperatura, luz de los salones según la acción del usuario).

- Gestión Energética:
  - Monitoreo de consumo eléctrico de electrodomésticos de alto consumo.
  - Automatización para optimizar la energía.

Necesidades no funcionales:

- Seguridad:
  - Cifrado de datos en las comunicaciones.
  - Autentificación de dispositivos que acceden desde fuera de la red.
  - Control del tráfico de red.

- Compatibilidad e Integración:
  - Soporte de protocolos domóticos estándares (Z-Wave, Zigbee, Wifi).
  - Capacidad de integrar asistentes de voz (Alexa, Google Assistant, Siri).
  - Capacidad de integración de un sistema de generación solar.

- Escalabilidad y mantenimiento:
  - Posibilidad de ampliación.
  - Capacidad para aceptar nuevos protocolos de comunicación (KNX, Bacnet, TCP/IP, entre otros).
  - Sistema modular.

Todas estas necesidades se han recogido por el estudiante para realizar pruebas de funcionamiento del sistema domótico. Debido al elevado precio de todos los dispositivos IoT, la domotización de la vivienda se realizará de forma progresiva y se irá completando poco a poco. Además, se podrán incluir nuevas necesidades y requerimientos según vaya avanzando la domotización de la vivienda.

Actualmente, se han instalados controladores lumínicos en ambos salones, la habitación de la planta superior, escaleras y cocina. Se han instalado sensores de temperatura en el salón superior y escaleras. Se han colocado enchufes inteligentes en diferentes tomas de corriente para controlar el encendido/apagado y monitorizar el consumo de diferentes electrodomésticos. Se ha integrado un pulsador en la centralita del aire acondicionado para apagar y encender el aire de forma remota y por temperatura. Se han diseñado y colocado sensores de presencia para el control de las luces en las escaleras y salón de la planta de arriba. Por último, se ha instalado en la terraza un sensor de lluvia.

En un futuro se irá ampliando el número de dispositivos IoT, para controlar más habitaciones de la vivienda. Además, se pretende instalar en un futuro placas solares que se integrarán al sistema para optimizar el uso de la energía.

4.3. Instalación eléctrica

Las modificaciones necesarias de la instalación eléctrica son sencillas y poco complejas. El objetivo es instalar entre el interruptor y las luces un relé Zigbee (véase el esquema de instalación en la Figura 4.4). Para realizar estas modificaciones en la instalación eléctrica es necesario seguir las normas de seguridad conocidas como 5 reglas de oro.

1. Cortar la tensión.
2. Bloquear el seccionamiento.
3. Verificar la ausencia de tensión.
4. Poner a tierra y en cortocircuito.
5. Señalizar y delimitar la zona de trabajo.

La única dificultad surge a la hora de identificar los cables que se dirigen a la bombilla, ya que existen 3 tipos de interruptores: simples, conmutados y cruzados. Los interruptores simples tienen dos terminales y solo permite encender o apagar un circuito desde un solo punto. Los interruptores conmutados tienen tres terminales y permite controlar una misma luz desde dos puntos diferentes. Por último, los interruptores cruzados se usan cuando se necesita controlar una luz desde tres o más puntos diferentes, tienen cuatro terminales y funciona en combinación con dos interruptores conmutados, el interruptor cruzado se coloca entre los dos interruptores conmutados.

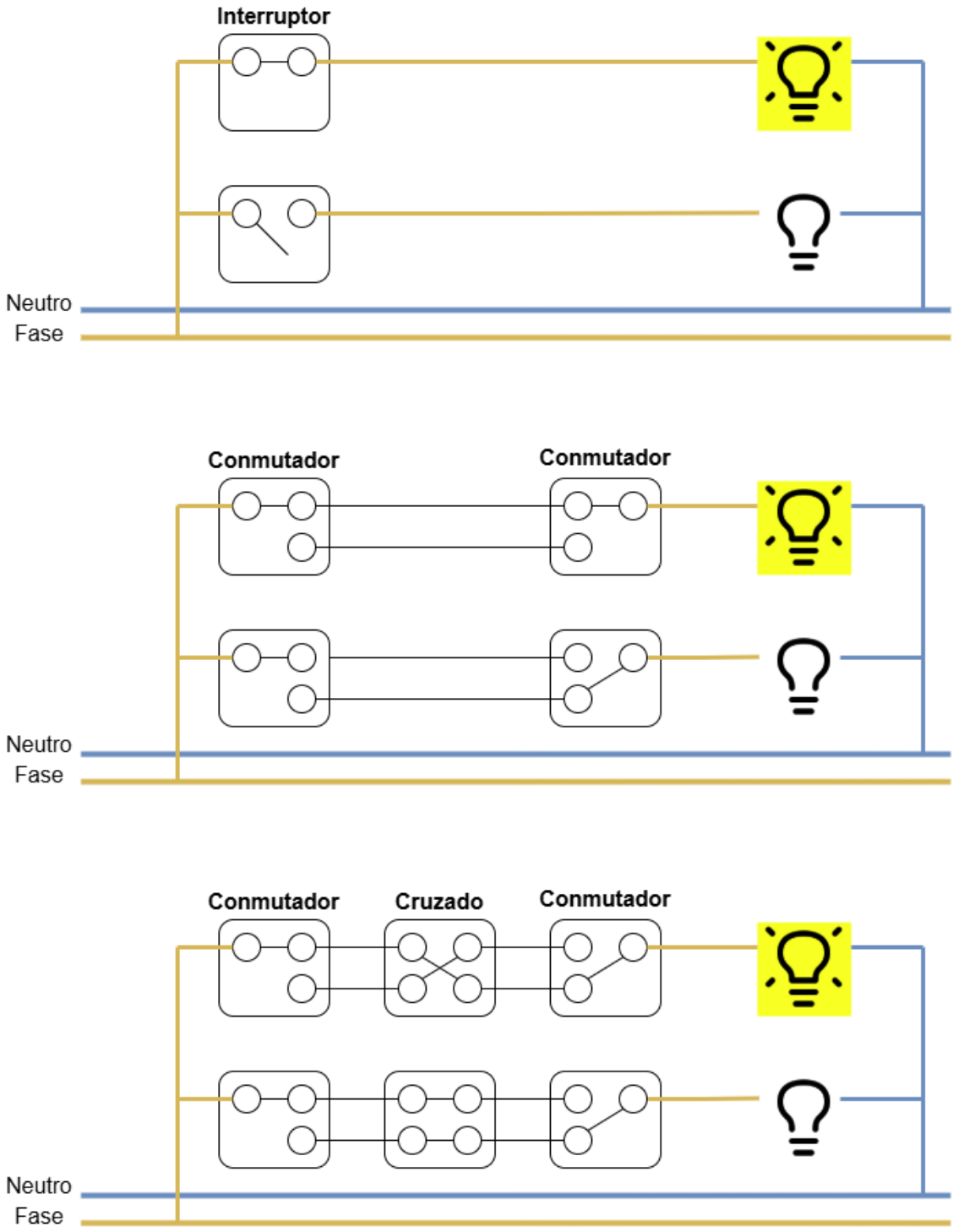

Figura 4.3. Tipos de interruptores

Una vez localizado el interruptor, se corta el cable que conecta el interruptor con la bombilla. Por último, se conecta el interruptor, la bombilla al relé y este se conecta también a la red para tener tensión. Observe el esquema de conexión en la Figura 4.4.

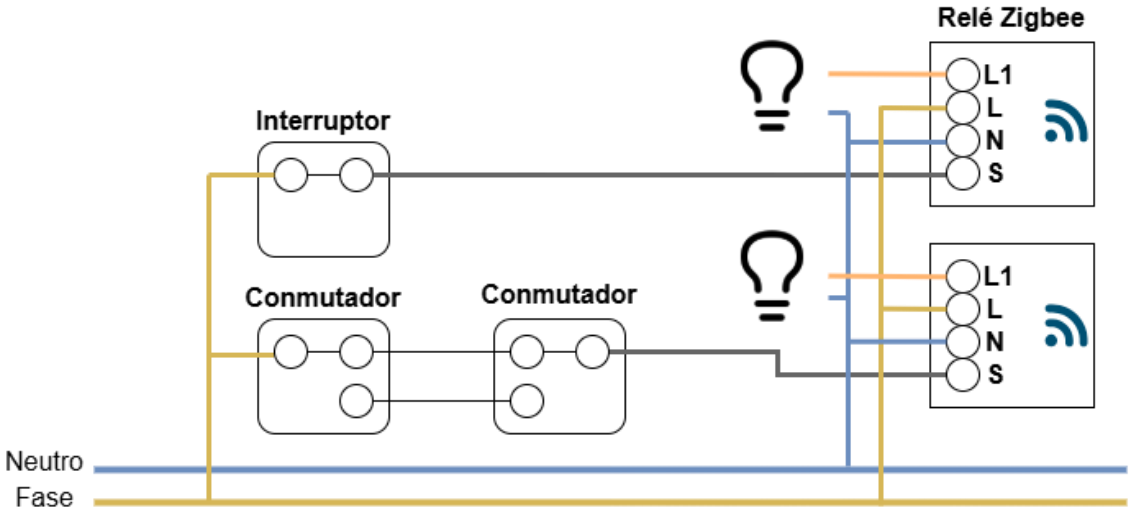

Figura 4.4. Esquema conexión relé

Figura 4.5. Localización del cable que va a la bombilla

Figura 4.6. Instalación Relé

4.4. Panel de usuario personalizado

Para desarrollar el panel de usuario se ha hecho uso del servicio Home Assistant descrito anteriormente.

Lo más útil de Home Assistant es que permite al usuario crear su propio panel de forma muy intuitiva y sencilla. El panel que incorpora HA contiene diferentes temáticas haciéndolo muy personalizable y ajustado a las necesidades del usuario.

En el caso de la vivienda del estudiante, se han diseñado varios *dashboards*, cada uno correspondiente a una zona específica que se desea monitorizar (véanse Figuras 4.10 y 4.11). Adicionalmente, se ha creado un *dashboard* global que integra todos los dispositivos para ofrecer una visión unificada del sistema (Figura 4.12). Asimismo, se han incorporado **dos** *dashboards* de planta que permiten visualizar de forma esquemática la distribución y el estado de los dispositivos en el conjunto de la vivienda (Figuras 4.13 y 4.14).

Para crear y modificar el *dashboard* hay que seguir los siguientes pasos:

1. Se pulsa en los tres puntos situados en la esquina superior izquierda, donde se muestra en la Figura 4.7, rodeada de color rojo.

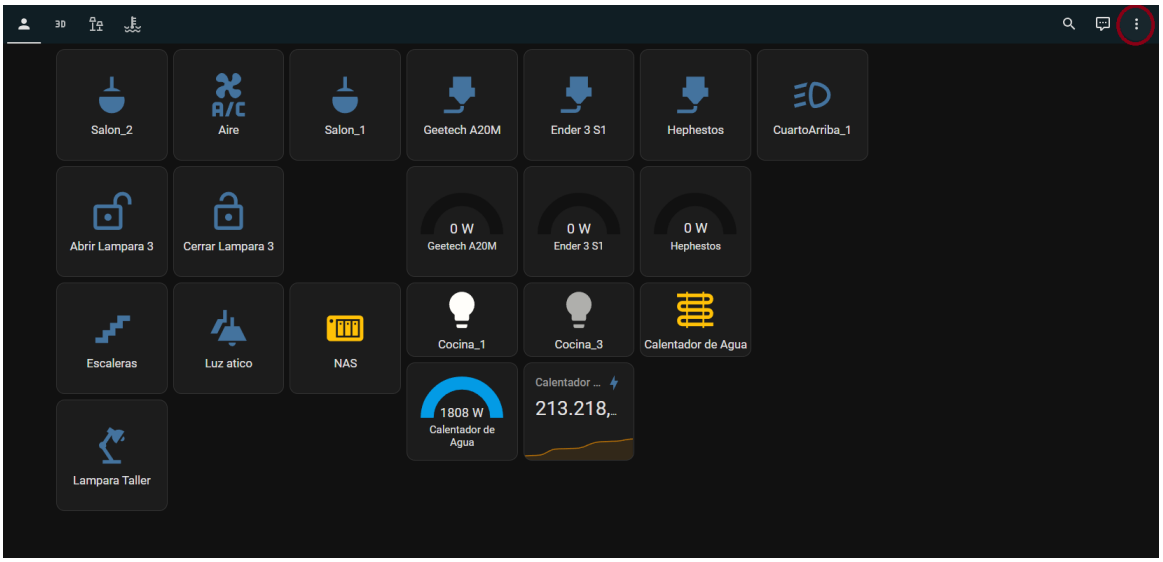

Figura 4.7. Panel Home Assistant

2. Se pulsa sobre *Editar panel de control*

3. Para añadir una tarjeta, se presiona en el botón *AÑADIR TARJETA*

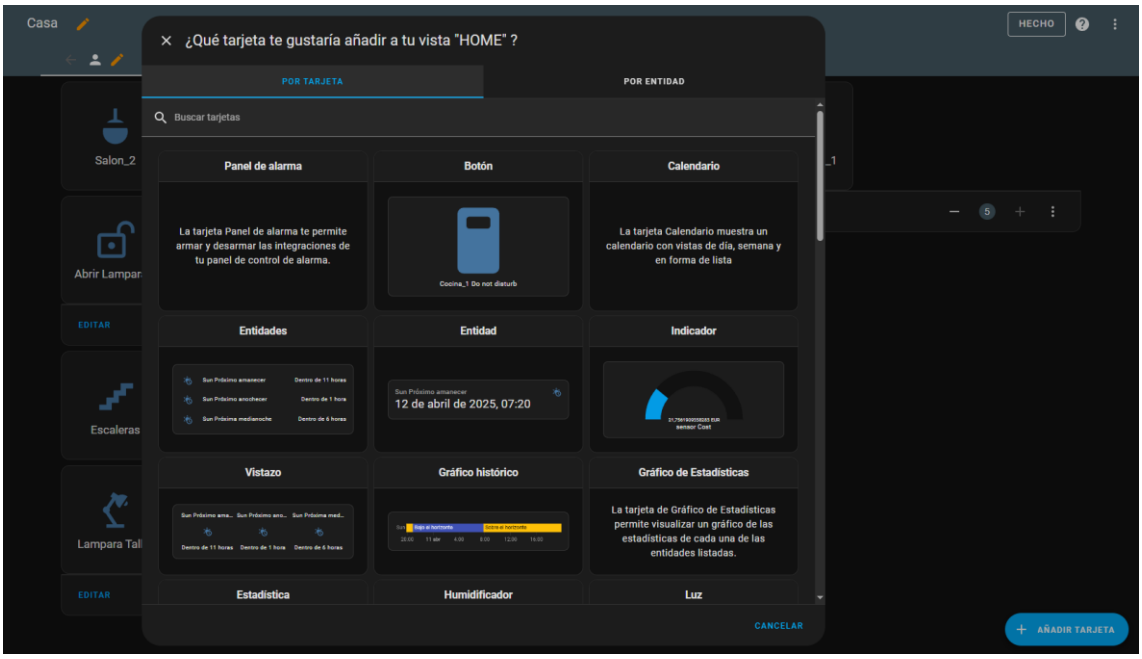

Figura 4.8. Configuración de tarjetas

4. En este menú se selecciona la tarjeta que se quiera colocar en el *dashboard*, por ejemplo, se selecciona *Botón.*

5. Por último, se rellena todos los campos que se muestran en la Figura 4.9.

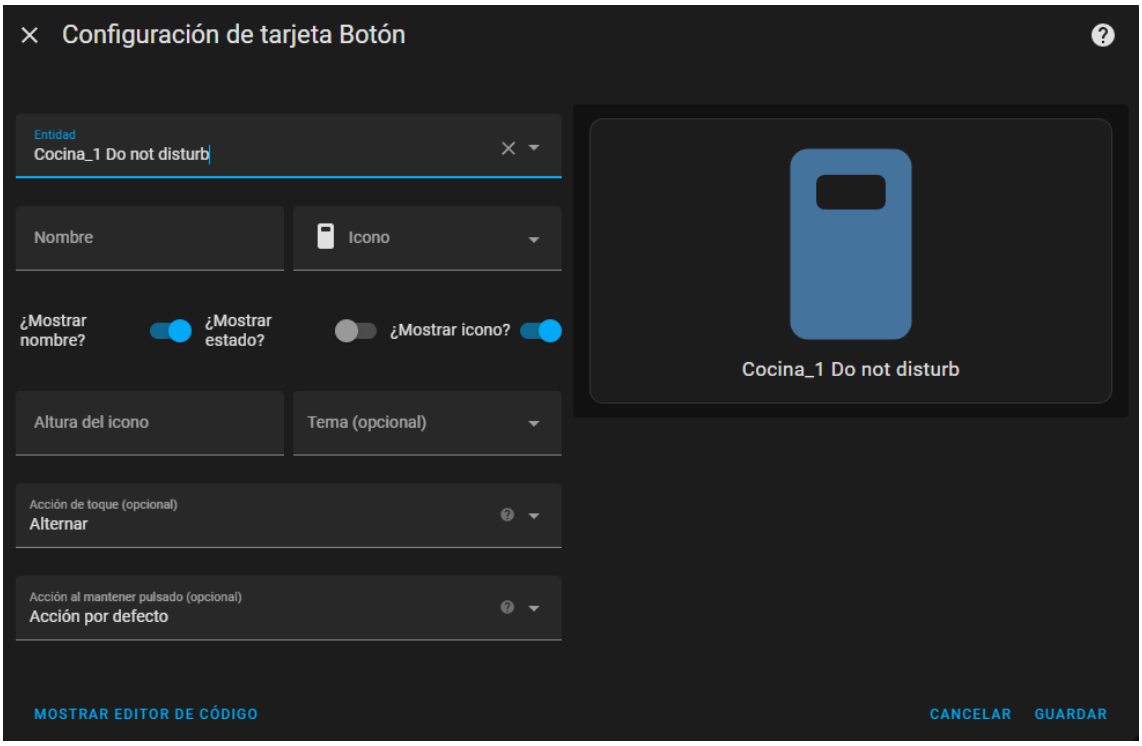

Figura 4.9. Configuración de pulsador

Con estos simples pasos ya hemos creado un botón en el panel de usuario. A continuación, se muestran los *dashboard* que se han creado para el control de la vivienda.

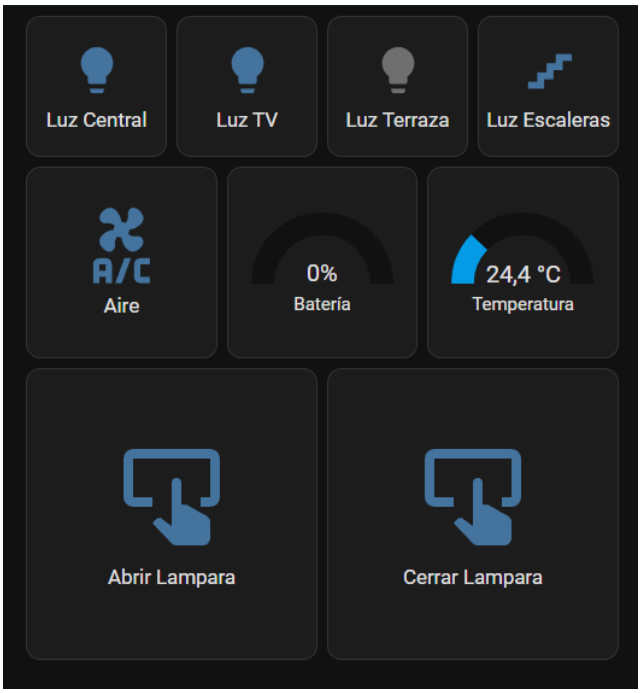

Figura 4.10. Dashboard salón

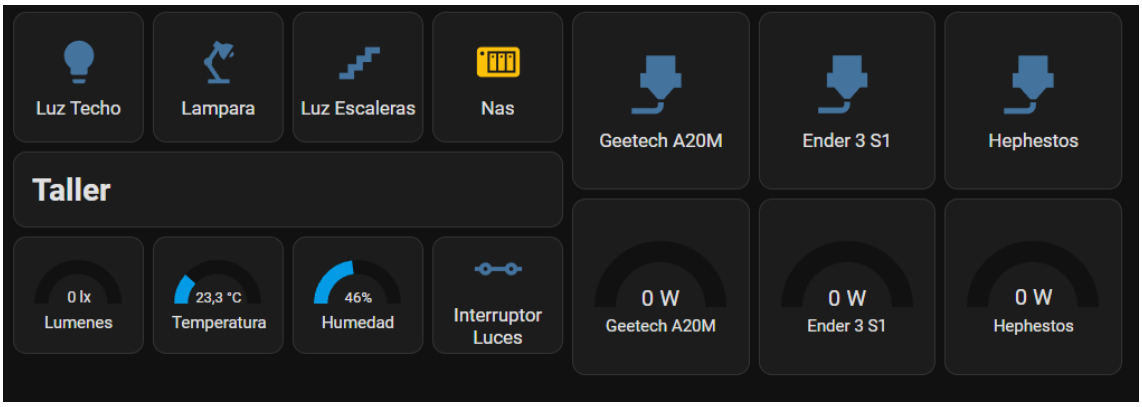

Figura 4.11. Dashboard taller

Figura 4.12. Dashboard global

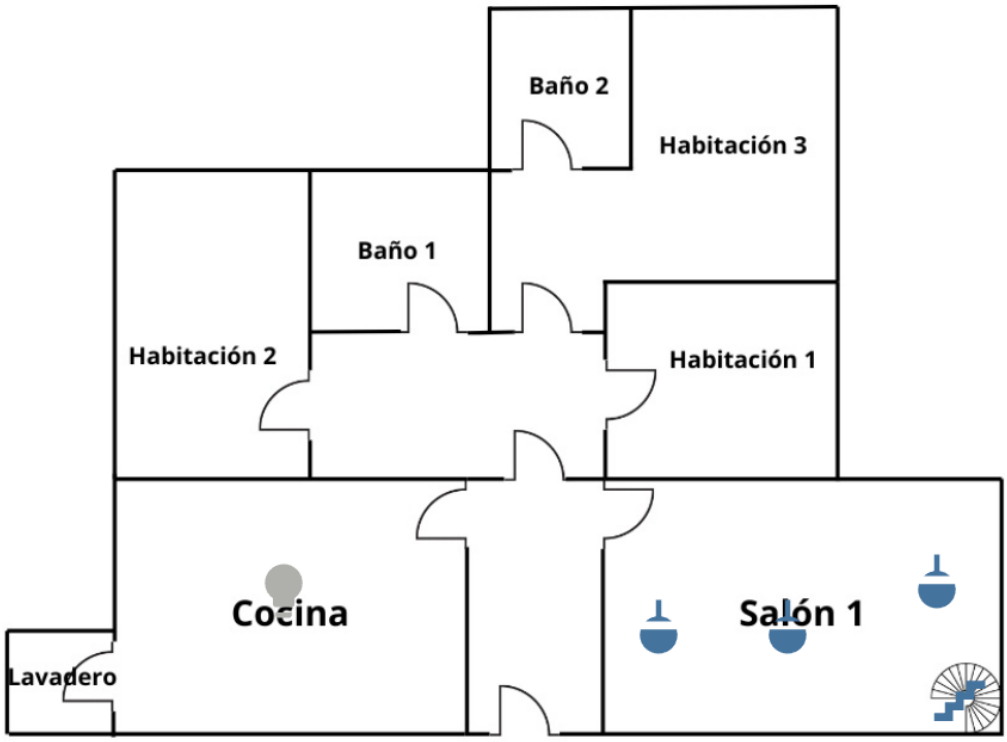

Figura 4.13. Dashboard plano de planta baja

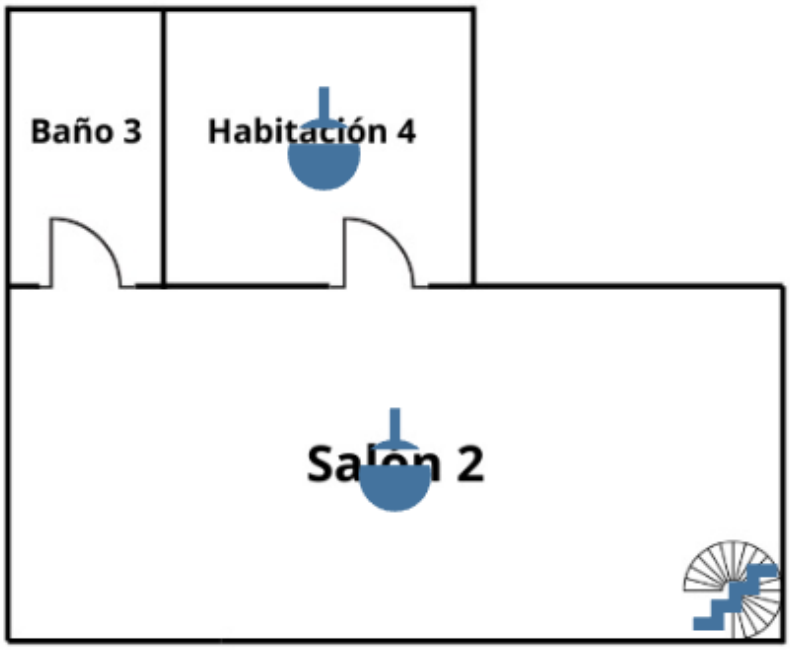

Figura 4.14. Dashboard plano de planta superior

4.5. Escenas

El diseño de las escenas es una tarea relativamente compleja, ya que parte directamente de las necesidades específicas del usuario, las cuales deben estar claramente definidas para garantizar su correcta implementación. Las escenas constituyen el componente más personalizable del sistema IoT, puesto que su diseño y programación dependen del entorno particular y del uso que se pretenda dar al sistema. En este apartado se describen únicamente un par de escenas representativas a modo de ejemplo, con el objetivo de ilustrar su funcionamiento y lógica. No obstante, el conjunto completo de escenas desarrolladas puede consultarse en el repositorio oficial del proyecto, disponible en el enlace https://github.com/PacoLC/IoT_TFM/tree/main/Microservicios.

Para programar las escenas se parte de las condiciones que se tienen que dar en el entorno para realizar una acción. Por ejemplo, el encendido automático de las luces de la escalera cuando se detecta presencia. Para realizar esta tarea se debe de partir de la información de los sensores instalados previamente.

El sensor de movimiento detecta presencia y envía esta información a NodeRed. El siguiente paso no es encender la luz directamente, sino que comprueba que en la escalera no haya la luz suficiente para que el ojo humano pueda ver correctamente. Si no hay luz y hay alguien en la escalera se enciende la luz; en cualquier otro caso no se enciende.

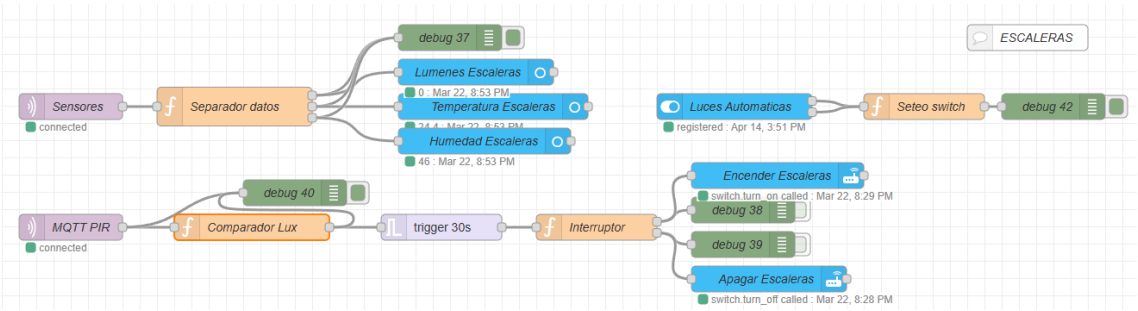

Figura 4.15. Flujo escaleras

En la Figura 4.15 se muestra el flujo que se ha diseñado para el ejemplo mencionado en el párrafo anterior. A continuación, se explica el funcionamiento de este flujo.

1. En la parte izquierda de encuentran dos nodos que reciben los datos por MQTT. Llamados *Sensores* y *MQTT PIR*. Comenzando con el nodo *Sensores,* este flujo recopila los datos de temperatura, humedad y la Iluminancia de forma periódica. En cambio, por el otro nodo llega la información en el momento que detecta presencia.

2. Seguido a estos nodos se encuentra el bloque función, donde se puede programar scripts utilizando Javascript. El bloque *Separador datos* envía por cada canal el dato que recibe de cada sensor y guarda en una variable global los lúmenes detectados en la última actualización. La función *Comparador Lux* compara que los luxes detectados por el sensor no sean inferiores al umbral establecido. Si los luxes son inferiores envía un mensaje al siguiente bloque.

3. El siguiente bloque es un temporizador, que cuando se activa envía un valor *HIGH* durante 30s, pasado este tiempo se envía un valor *LOW*.

4. Este valor *HIGH* o *LOW* es el que se interpreta en el siguiente nodo función para encender y apagar las luces de las escaleras.

5. Además de estos dos flujos analizados, se puede observar un tercer flujo a la derecha de la figura *Luces Automáticas → Seteo switch*, este flujo crea un interruptor ficticio en Home Assistant que me permite desactivar el flujo anterior para depurar la escena.

Otros ejemplos de escenas que se han creado con NodeRed en el entorno doméstico son: controlar la energía consumida por algunos dispositivos durante días (Figura 4.17), controlar dispositivos DIY (actuadores, Figura 4.16), apagar y encender el calefactor o aire acondicionado según la temperatura (Figura 4.19), integración de interruptores DIY para controlar dispositivos de la vivienda que no están conectados directamente, entre otras.

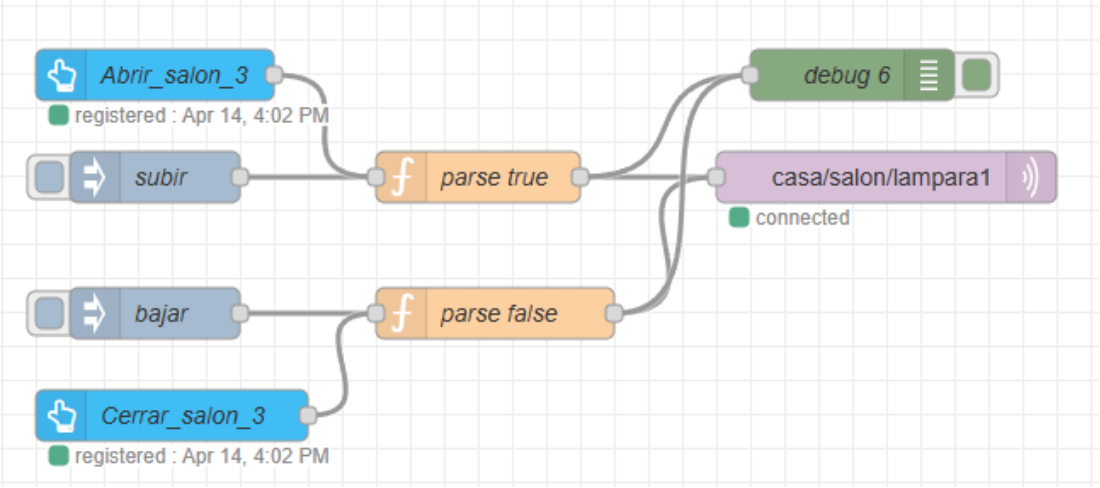

Figura 4.16. Flujo control de un actuador y una lámpara

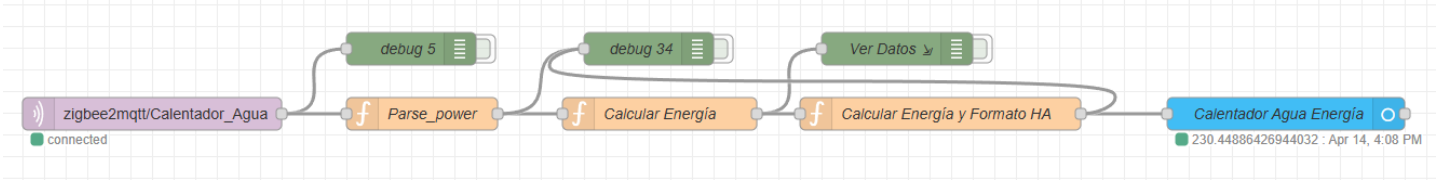

Figura 4.17. Flujo monitoreo energético

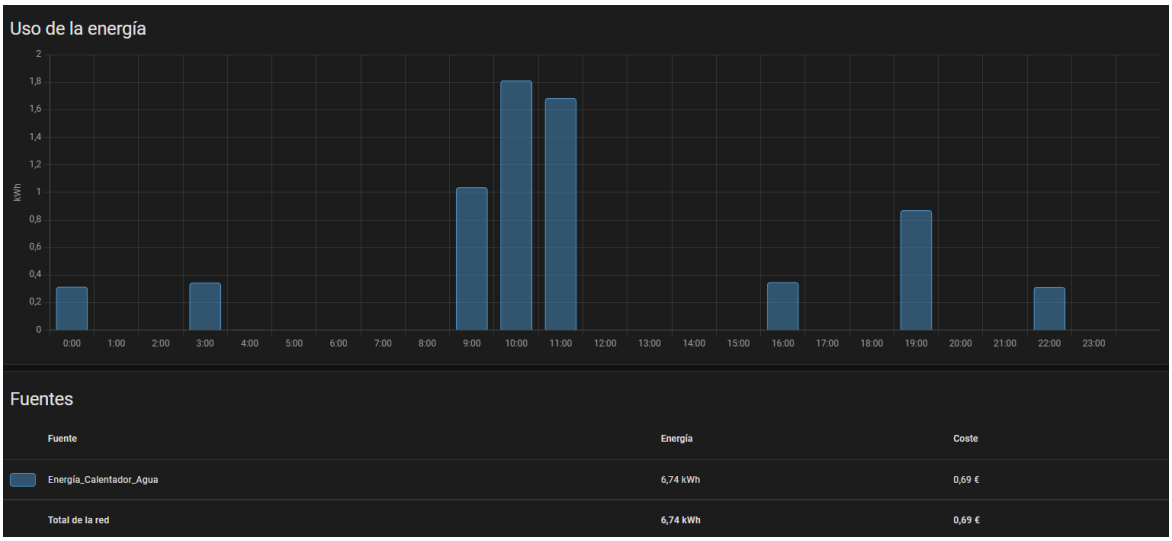

Figura 4.18. Gráfico energético

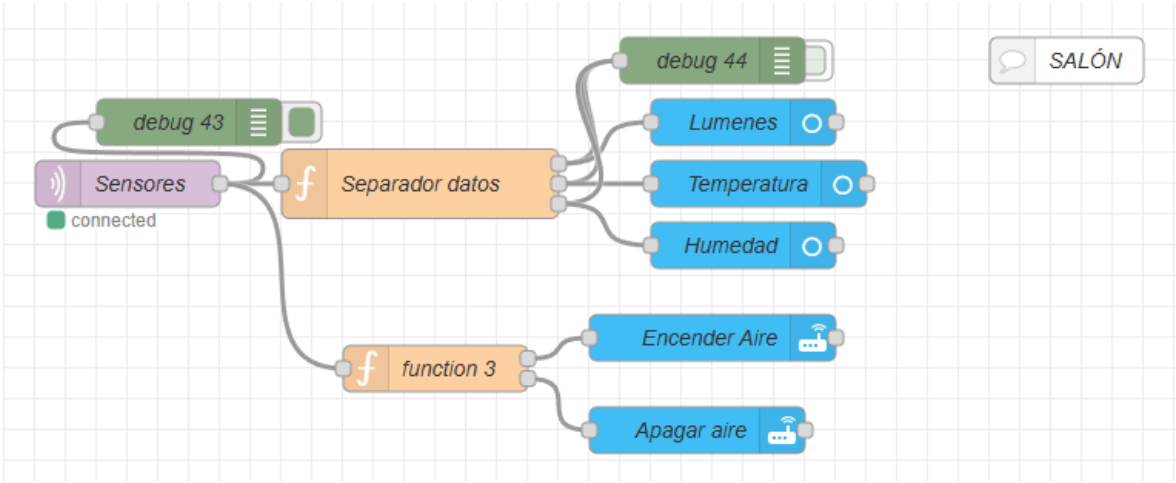

Figura 4.19. Flujo salón

4.6. Seguridad

La seguridad del sistema es un punto muy importante que se tiene que tratar y analizar con cautela. En el caso del estudiante, no se dispone de un sistema de alarma en el domicilio, pero se podría integrar. El sistema IoT no puede tener vulnerabilidades que puedan afectar a los sistemas de alarma o de vigilancia.

En un principio el sistema se diseñó para poder abrirse a la red, es decir, el usuario pudiera acceder al panel de usuario desde cualquier parte. Sin embargo, esto no va de la mano de la seguridad, si el usuario puede acceder por una IP o una URL, una persona con conocimientos de *hacking* podría vulnerar el sistema. La solución a estos problemas podría ser cerrando la red local y que solo se pudiera acceder al sistema IoT mediante una conexión a la red y no desde fuera.

Si se cierra el sistema a solo la red local, el sistema sería más seguro, pero no se podría controlar desde cualquier lugar. No obstante, existe una manera de abrir el sistema a la red sin estar tan expuesto.

La idea es mantener el sistema en local, pero crear una VPN (*Virtual Private Network*) que permita a los dispositivos redirigir el tráfico de internet y simular que está conectada a la red local [52]. Es decir, se crea un servicio que reciba conexiones externas y las redirige por el *router* local, como si estuvieran conectados estos dispositivos al *router* del domicilio. Esta opción no expone directamente el sistema IoT a internet, sino que permite "conectarte desde cualquier parte al wifi del domicilio", y, por tanto, permite controlar el sistema IoT.

Esta opción es más segura que abrir el sistema IoT completo, ya que esto implica dar acceso remoto a todos los servicios, creando un alto número de puertas traseras, aunque el sistema tenga cuentas y contraseñas. Una persona experta podría vulnerar los accesos y afectar al sistema IoT desde cualquier parte. Sin embargo, crear una VPN permite controlar los dispositivos que se conectan a la misma, ya que para conectarte por la VPN necesitas un *token* único y cíclico que se genera de forma manual en el servicio.

En este proyecto se ha usado la VPN de WireGuard [53] y se ha levantado desde un contenedor Docker.

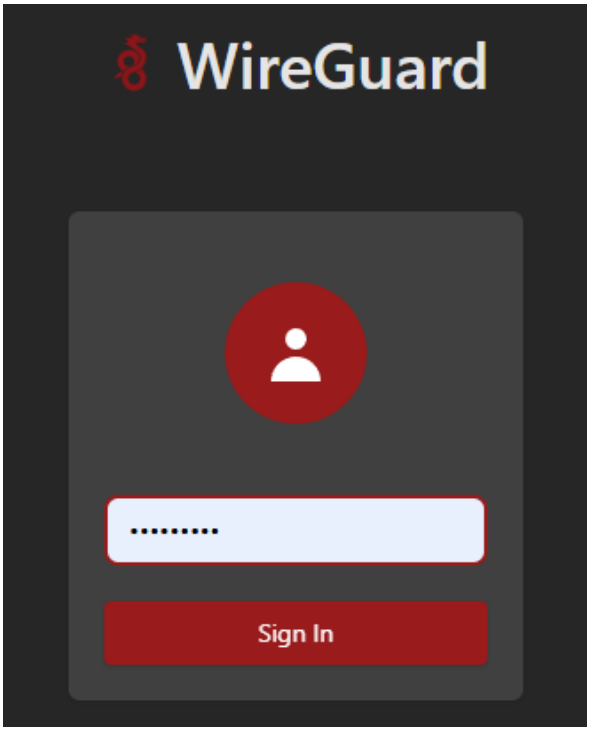

Figura 4.20. VPN WireGuard

Al acceder al panel de control nos permite gestionar los clientes que se conectan a través de la VPN, generar nuevos clientes (añadir dispositivos a la VPN) y crear copias de seguridad.

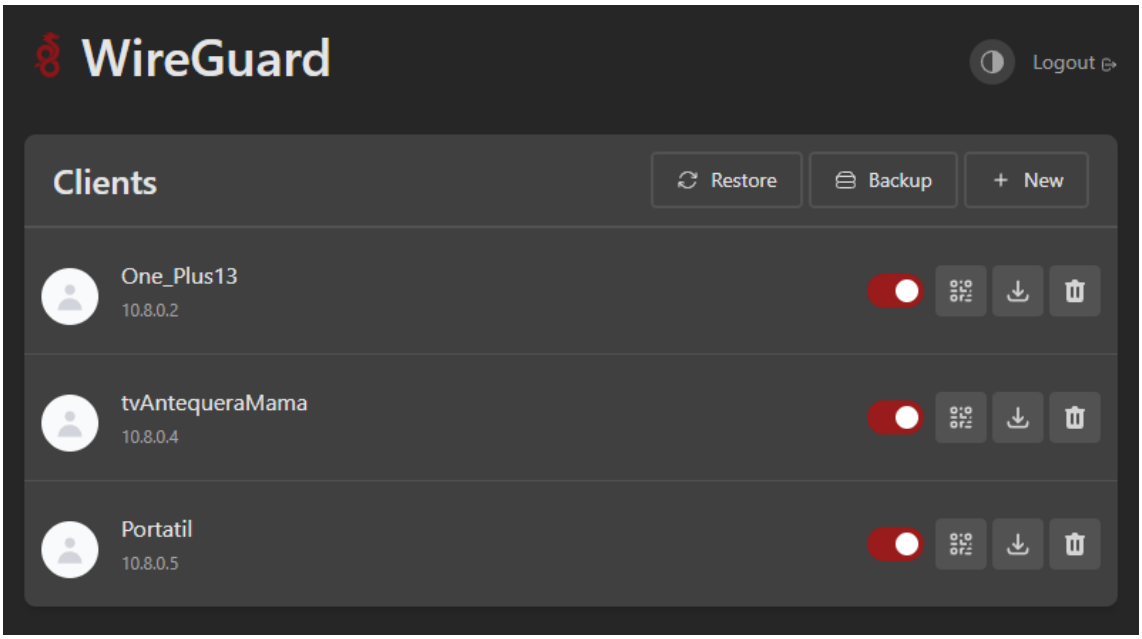

Figura 4.21. Clientes VPN

En resumen, el uso de una VPN como medida de seguridad representa un equilibrio entre protección y accesibilidad. Permite mantener el sistema IoT accesible desde cualquier punto sin comprometer la seguridad, evitando su exposición directa a internet. Gracias a la herramienta WireGuard, que se implementa mediante contenedores, se consigue una gestión centralizada y segura de los accesos remotos, asegurando que solo los dispositivos autorizados puedan interactuar con el sistema.

4.7. Pruebas realizadas en el entorno doméstico

Con el objetivo de validar el funcionamiento del sistema IoT en un entorno real, se llevó a cabo una serie de pruebas en la vivienda del estudiante. Este entorno sirvió como caso práctico para comprobar la fiabilidad, estabilidad y utilidad del sistema. Durante el despliegue, se integraron diversos dispositivos: sensores de movimiento, relés inteligentes, enchufes medidores de consumo, controladores IR y actuadores Zigbee, todos gestionados a través de Home Assistant y Node-RED.

Las pruebas incluyeron tanto el control remoto de dispositivos mediante el *dashboard* desarrollado, como la ejecución automatizada de escenas en función de variables como la presencia detectada, temperatura de las habitaciones, entre otras. Se verificó la capacidad de respuesta de los dispositivos, el tiempo de comunicación mediante MQTT y la estabilidad de los flujos de Node-RED, incluso tras reinicios del sistema o interrupciones puntuales de red.

Los resultados fueron satisfactorios: el sistema respondió de forma precisa y coherente ante los eventos definidos, permitiendo un control eficiente de la iluminación, el confort y el consumo eléctrico. Asimismo, se comprobó la utilidad del sistema de monitorización para detectar consumos anómalos o estados prolongados no deseados, lo que aporta un valor añadido en términos de eficiencia energética. Estas pruebas permitieron afinar algunos aspectos del diseño, como el ajuste de temporizadores de apagado de luces, la mejora de mensajes MQTT o la redefinición de ciertas condiciones en las escenas.

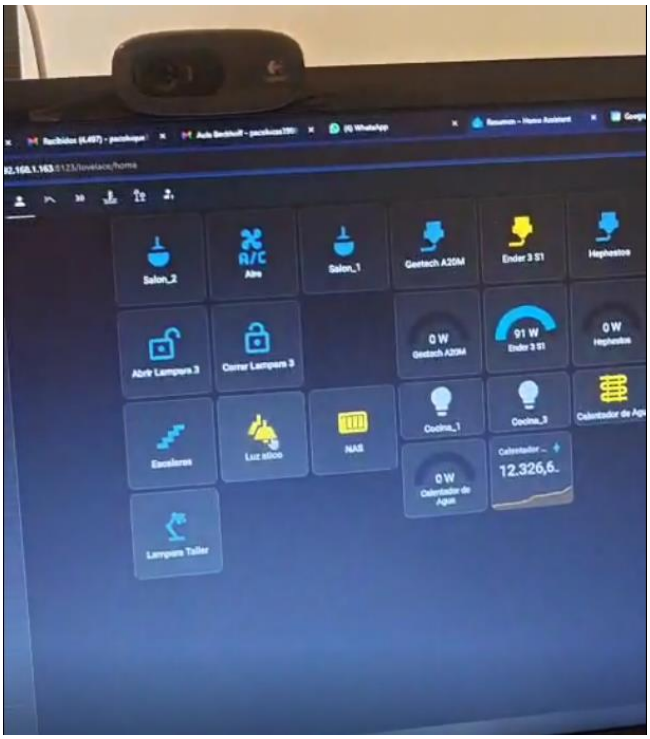

Figura 4.22. Prueba del dashboard en la vivienda

Figura 4.23. Prueba de sensores en la vivienda

CAPÍTULO 5. INSTALACIÓN Y PRUEBAS EN UN ENTORNO INDUSTRIAL

5.1. Plan de trabajo

Este apartado detalla el plan de trabajo para la instalación de un sistema IoT en un entorno industrial. En este caso, la instalación no se realizará en una industria real, sino en un entorno docente en el ámbito de la automatización industrial: el laboratorio 3.506-L1 (aula Beckhoff) de la Escuela de Ingenierías Industriales de la Universidad de Málaga. A continuación, se presentan tanto la planificación teórica como la implementación práctica en dicho entorno.

1. **Planificación de la Instalación.**
   Antes de proceder con la instalación, es fundamental realizar un análisis del entorno y definir las necesidades específicas del sistema. Para ello, se siguen los siguientes pasos:

   - **Análisis del entorno**: Recopilación de características del espacio, dimensiones y ubicación.
   - **Identificación de necesidades**: Determinar qué dispositivos requieren automatización, como el encendido de luces, control del consumo energético de maquinaria, monitoreo de presión de aire comprimido, gestión del sistema HVAC, entre otros.
   - **Evaluación de la infraestructura eléctrica**: Identificación de modificaciones necesarias en la instalación eléctrica.
   - **Selección de la ubicación del servidor**: Determinar el mejor lugar para instalar el servidor IoT, que funcionará como nodo maestro de la red. Dado que los entornos industriales suelen ser grandes, es posible que se necesiten repetidores o redes con conexión física para garantizar la cobertura.
   - **Diseño de la distribución de dispositivos**: Planificación de la ubicación de sensores y actuadores para optimizar la eficiencia y cobertura del sistema.

2. **Características del Entorno.**
   La instalación se lleva a cabo en el aula Beckhoff (ver Figura 5.1), ubicada en la tercera planta de la Escuela de Ingenierías Industriales de la Universidad de Málaga. Este laboratorio contiene cinco estaciones automatizadas de la serie FMS200 de *SMC International Training* [54], controladas mediante PCs embebidos de la marca Beckhoff. Estos controladores tienen la capacidad de comunicarse en red mediante el protocolo MQTT, por lo que pueden actuar como emisores de la información que obtienen de los sensores de la planta que controlan (o cualquier otro dato que utilicen en su lógica de control) y también como receptores de comandos desde un dispositivo externo. En concreto, para este trabajo, se centraron los esfuerzos en conectar la estación FMS201 (Estación de inserción de bases, ver Figura 5.2 al sistema IoT. Esta estación incluye una serie de sensores (finales de carrera, inductivos, de presión y de vacío) y actuadores (cilindros neumáticos de desplazamiento lineal, ventosas de vacío, motores eléctricos para una cinta transportadora, lámparas y avisadores sonoros). La estación también dispone de un compresor para dotar de energía neumática a los actuadores.

   Se identificaron las siguientes necesidades específicas para el laboratorio de automatización:

- **Monitorización del consumo energético**: Medición del consumo del compresor de aire sin modificar la instalación eléctrica, utilizando dispositivos *Plug and Play*.
- **Gestión de las estaciones**: Control manual remoto y comunicación con las estaciones del aula para futuras automatizaciones.

3. **Implementación del Sistema IoT**.
   Una vez planificada la instalación, se procede con la ejecución de las siguientes tareas:

   - **Instalación del servidor**: Configuración de un servidor con sistema operativo Debian, alojado en una Raspberry Pi conectada a la red del laboratorio.
   - **Pruebas iniciales**: Conexión de dispositivos Plug and Play para verificar el funcionamiento de la red y la comunicación con las máquinas mediante programas de prueba.
   - **Programación y personalización:** Desarrollo del panel de usuario, definición de escenas y configuración de automatizaciones según las necesidades del aula.
   - **Pruebas finales y optimización:** Verificación del funcionamiento del sistema, asegurando una cobertura estable y eficiente.
   - **Seguridad y mejoras**: Implementación de medidas de seguridad para proteger el servidor IIoT, evitando accesos no autorizados mediante la ejecución en una red local sin acceso externo.

4. **Consideraciones.**
   Es fundamental garantizar que, en caso de que el servidor IIoT falle, los usuarios puedan seguir interactuando con las máquinas de forma manual.

   Para finalizar, se elabora un apartado de conclusiones basado en las experiencias y aprendizajes obtenidos antes, durante y después de la instalación del sistema en el aula Beckhoff. Este procedimiento asegura una implementación eficiente y segura del sistema IoT en un entorno industrial simulado, proporcionando una plataforma de pruebas y aprendizaje para futuros desarrollos.

5.2. Necesidades del aula

En esta sección se recopilan las necesidades del aula y se detallará la solución optada para suplir esos requerimientos.

Antes de comenzar el detallado de las necesidades hay que conocer el aula donde se pretende instalar el sistema. El aula es una habitación con varios ordenadores de sobremesa y en medio de la sala se encuentran 5 estaciones. Estás estaciones se utilizan para enseñar a los alumnos de Grados y Másteres a programar PLCs.

Figura 5.1. Aula Beckhoff

La estación que se conectará al sistema será la FMS-201. Es una estación modular de fabricación flexible diseñada para fines educativos. Representa una célula de manufactura automatizada que integra componentes industriales reales, lo que permite la simulación de procesos de producción reales. La función principal de la estación es colocar automáticamente una base sobre un palet del sistema de transferencia, verificando previamente su orientación. Si la base está invertida, se rechaza, y si es correcta, se inserta mediante un manipulador con sistema de vacío [55].

Se compone de un sistema de transporte tipo banda, sensores ópticos, detectores inductivos, un sistema de alimentación de piezas, actuadores neumáticos y un PC embebido de Beckhoff.

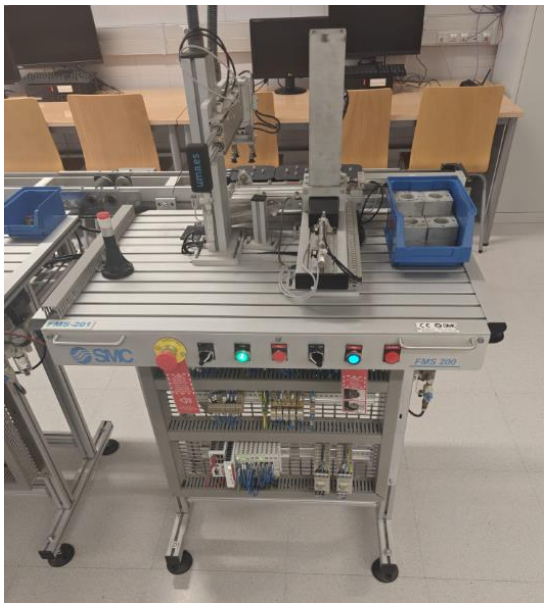

Figura 5.2. Estación FMS-201

A continuación, se detallan las necesidades del aula, que se dividen en dos ramas, necesidades funcionales y necesidades no funcionales.

Las necesidades funcionales:

- Comunicar la máquina con un servidor MQTT:
  - Conectar la máquina a un servidor MQTT alojado en una Raspberry Pi.
  - Establecer una comunicación entre el servidor y la máquina.

- Control remoto de la máquina:
  - Control remoto de los actuadores.
  - Visualización remota del estado de la máquina.
  - Establecimiento remoto del número de unidades a fabricar.
  - Visualizar el estado del compresor de aire del aula.

- Interacción:
  - Desarrollo de un *dashboard* interactivo simple y funcional.

Las necesidades no funcionales:

- Monitorización remota de la máquina:
  - Monitorización remota de la máquina desde cualquier terminal conectado a la red.
  - Monitorización de los sensores y los actuadores.

- Monitorización de elementos externos:
  - Monitorizar el consumo del compresor de aire del aula.
  - Monitorizar el estado del compresor de aire.

Todas las necesidades descritas anteriormente han sido recogidas por el alumno y el profesor después de analizar con detalle el funcionamiento y características de la estación y del aula.

5.3. Programación de las máquinas

El control de la estación FMS-201 se realiza mediante un PC embebido de la marca Beckhoff, el cual permite programar lógicas de operación utilizando lenguajes estándares por la norma IEC61131-3 [56], como:

- Ladder (LD)
- Texto estructurado (ST)
- Diagrama de bloques (FBD)
- Diagrama secuencial de funciones (SFC)
- Lista de instrucciones (IL)

El control de la estación está implementado en un programa que combina los lenguajes ST y SFC (para la lógica secuencial de control), mientras que la comunicación a través de MQTT se realiza mediante otro programa, que se ejecuta en paralelo al programa de control, y que se encarga de la conexión con el *broker*, y el envío y la recepción de información con el panel diseñado. Al proporcionar Beckhoff librerías específicas para el control de la comunicación MQTT, no es necesario realizar ninguna modificación *hardware* al sistema para conseguir dicha funcionalidad. De esta forma, enviaremos el estado de los sensores y actuadores, a la vez que recibiremos órdenes para modificar las variables internas del código, modificando así el estado real de los actuadores, contadores, luces, y alarmas, entre otros. El mensaje enviado por MQTT en ambos sentidos tendrá formato JSON.

La programación de la estación se ha realizado con el entorno TwinCAT3 [57], el *software* de automatización desarrollado por Beckhoff para el control industrial, que convierte un ordenador en un sistema de control en tiempo real. También tiene un potente HMI que muestra el estado de los actuadores y sensores de la máquina, facilitando la comprensión del estado de la estación. La principal desventaja del uso del HMI integrado en TwinCAT3 es que, para observar el estado de otra de las estaciones en el sistema, se necesitaría otro ordenador o conectarse a otra estación por red, dificultando así el control de varias estaciones con el mismo HMI (Figura 5.4). Aunque el presente TFM se centre solo en la conexión de la estación FMS201, el procedimiento seguido es fácilmente ampliable al resto de estaciones, pudiendo obtener así la información de todas las estaciones en un mismo panel.

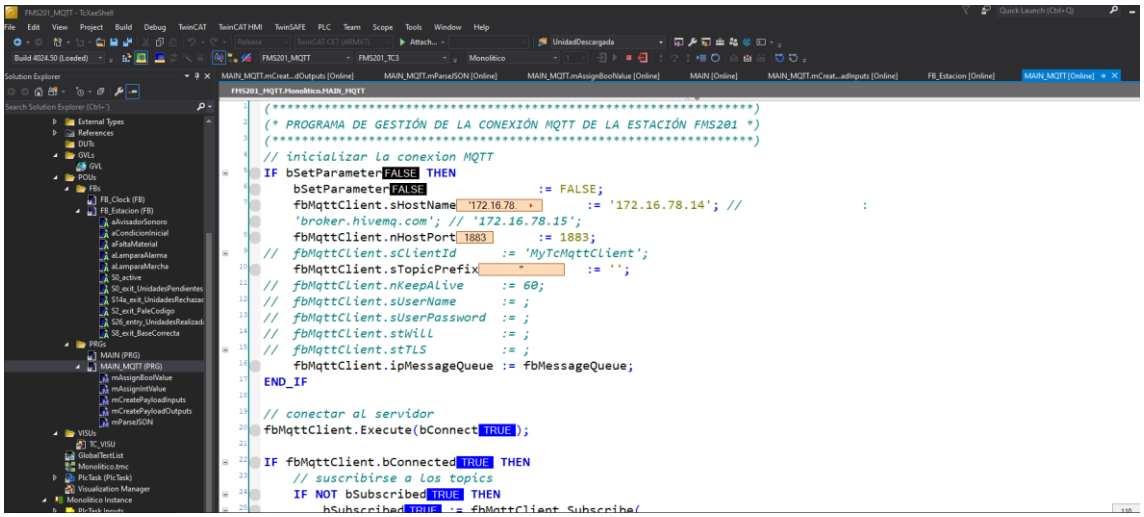

Figura 5.3. Programación en TwinCat3

Figura 5.4. HMI FMS-201

El código que controla la comunicación MQTT de la estación se puede consultar en https://github.com/PacoLC/IoT_TFM/tree/main/PLC y, básicamente, implementa la siguiente secuencia de acciones:

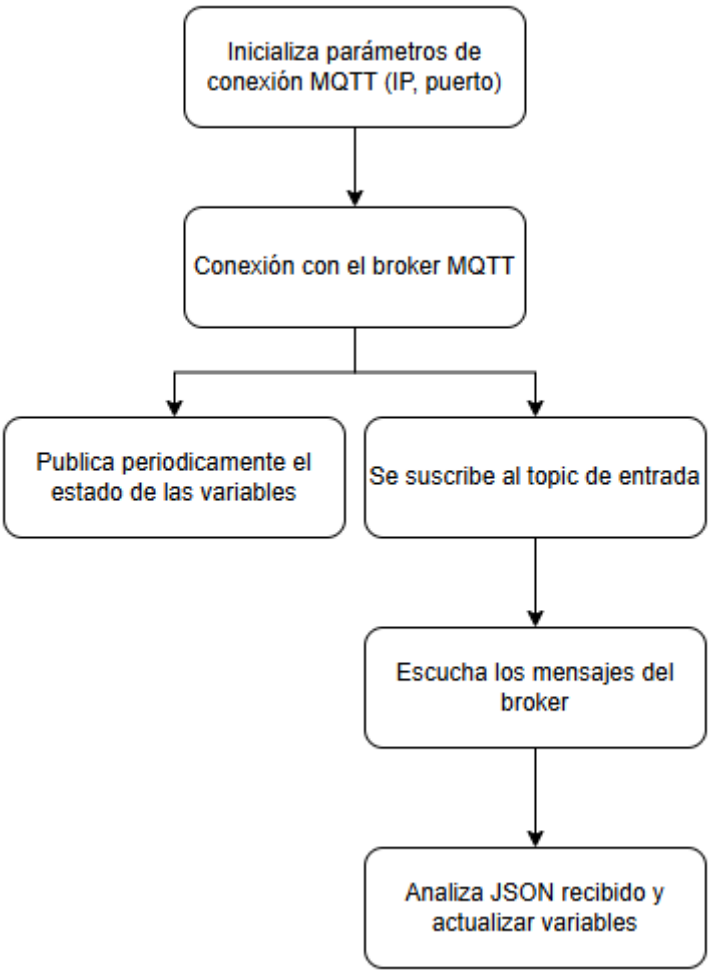

Figura 5.5. Dashboard alua Beckhoff

1. **Inicialización de conexión MQTT**. Se configuran los parámetros de red del cliente MQTT: dirección IP del *broker*, puerto y colas de mensajes. Solo se realiza una vez al iniciar el programa.

2. **Conexión al *broker***. Se intenta establecer conexión con el *broker* MQTT. Si la conexión es exitosa, se activa un flag ("bConnected") y continúa la lógica.

3. **Suscripción al *topic*.** Una vez conectado, el programa se suscribe a un *topic* definido en ("sTopicRcv") para poder recibir mensajes remotos desde el *dashboard* o sistema de supervisión.

4. **Publicación periódica.** Usando un temporizador, cada cierto intervalo se genera un mensaje JSON con los estados actuales de las entradas y salidas del sistema y se publica en sus respectivos *topics* ("sTopicPubE" y "sTopicPubS"). Buscando un compromiso entre una comunicación ágil y la evitación de la saturación de la red, se ha establecido que se envíen los mensajes cada 200 ms.

5. **Recepción de mensajes remotos.** El programa revisa si hay mensajes pendientes en la cola (“fbMessageQueue”). Si hay alguno, se extrae el *payload* y el *topic* del mensaje recibido.

6. **Análisis del mensaje JSON.** Se llama al método (“mParseJSON”) que interpreta el contenido del mensaje entrante. Según la clave, se actualizan las variables booleanas o enteras del PLC (usando “mAssignBoolValue” o “mAssignIntValue”).

5.4. Panel de usuario personalizado

Para realizar el panel de usuario para el aula Beckhoff se ha optado por usar el mismo servicio para crear nodos y escenas, NodeRed. Este servicio a-parte de ofrecer la programación por nodos permite crear paneles de usuario personalizados e intuitivos.

Se ha optado por esta opción, en lugar de *Home Assistant*, porque el servicio HA está pensado principalmente para instalaciones en entornos domésticos. Se planteó usar una interfaz de usuario diferente a la que ofrece NodeRed como es Grafana [58], pero se descartó la idea debido a que ésta, de forma nativa, no soporta datos de entrada, es decir, no se pueden crear botones interactivos, solo se puede mostrar la información proveniente de la máquina.

El *dashboard* resultante es el de la Figura 5.6, donde se distinguen cuatro bloques: Panel, Planta, Tareas y la parte inferior de las condiciones. Además, se ha creado una pestaña para observar el estado del compresor (Figura 5.7).

Figura 5.6. Dashboard aula Beckhoff

El bloque denominado “Panel” agrupa las entradas y salidas asociadas a los controles e indicadores que permiten la interacción directa del operador con el sistema. En este panel se

integran elementos como pulsadores de inicio y parada, botones de alarma, desconexión, activación de alarma sonora, así como indicadores visuales mediante luces LED. Estos componentes facilitan tanto el manejo seguro del sistema como la supervisión del estado operativo en tiempo real.

En el bloque “Planta” se representan todos los sensores (indicadores cuadrados bajo las columnas 0 y 1) y actuadores de la estación (botones de color rojo con las etiquetas + y -), mostrando su estado. Además, permite modificar el estado de los actuadores cuando la máquina se encuentra en modo manual. El estado de los sensores se representa mediante leds de color (gris: apagado, amarillo: encendido), cuya información es recopilada por el PC embebido controlador y enviada por MQTT. Para poder accionar los actuadores, se han creado unos botones que cambian de color según el estado (rojo: sin acción, verde: accionado). Estos botones son capaces de mostrar el estado del actuador y enviar información al controlador a través de MQTT.

El bloque “Tareas” representa el estado actual de la tarea que se está realizando, indicando el número de unidades que se ha solicitado, el número de unidades pendientes y las bases rechazadas y aceptadas desde que la estación se puso en movimiento. Estos indicadores, a diferencia de los mostrados en “Planta” son de tipo numérico. El primero de ellos (unidades solicitadas) es un indicador de entrada y salida, ya que permite no solo visualizar el número actual sino modificar su valor de manera remota.

El bloque inferior visualiza si la estación cumple sus condiciones iniciales y de marcha, además de incluir dos botones para poner la estación en pausa o reiniciar su funcionamiento (llevarlo a su estado inicial) de forma remota.

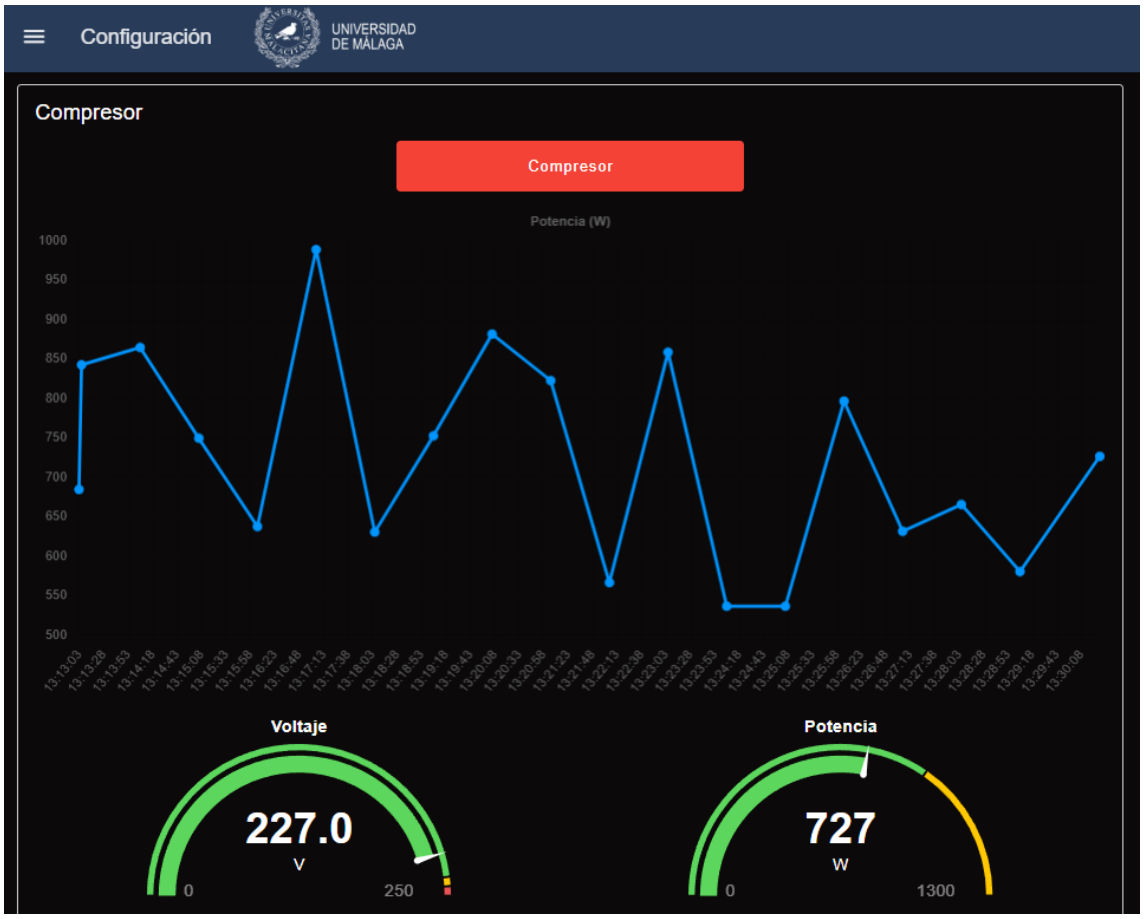

Figura 5.7. Pestaña compresor

En la pestaña del compresor se visualiza el estado (encendido o apagado) y se visualiza la potencia del compresor a través de un dial y de una gráfica a lo largo del tiempo.

Para realizar todos estos bloques, se han usado nodos de la paleta "*flowfuse/node-red-dashboard*" [59], los nodos para representar los leds son los nodos *ui-led* y para los botones *ui-button*. La arquitectura se muestra en las Figuras 5.8, 5.9 y 5.10.

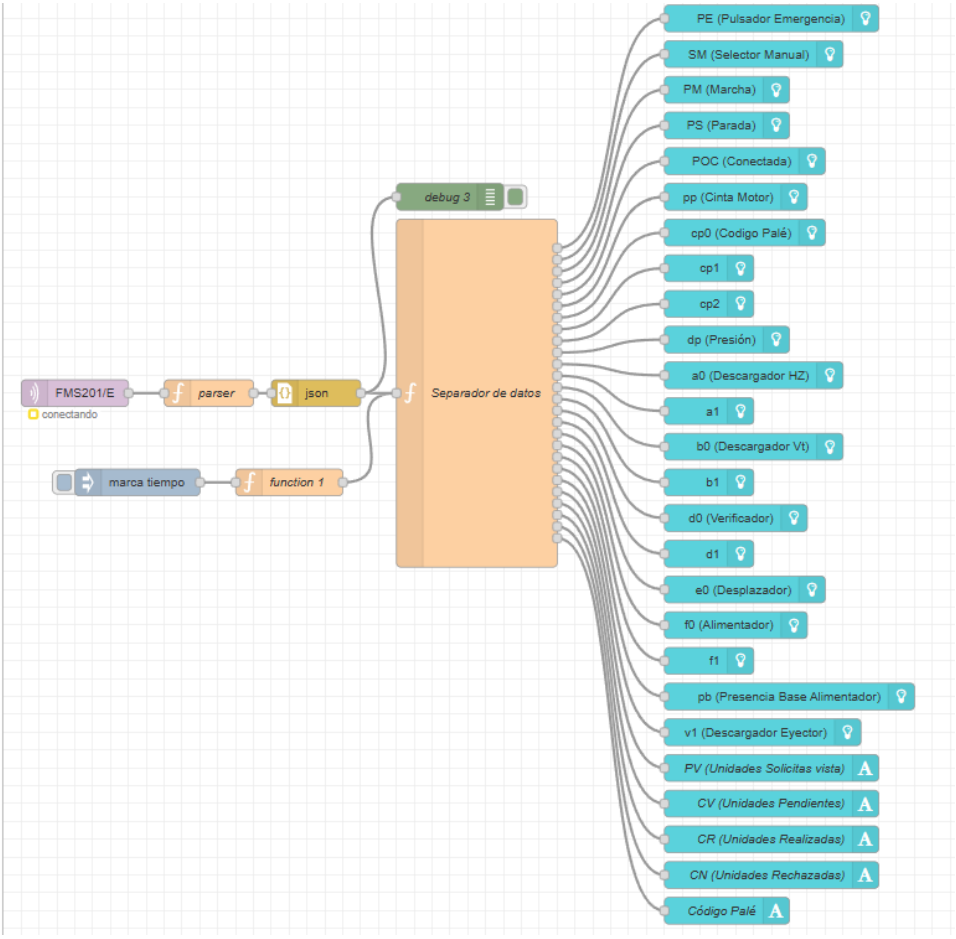

Figura 5.8. Flujo datos entrada

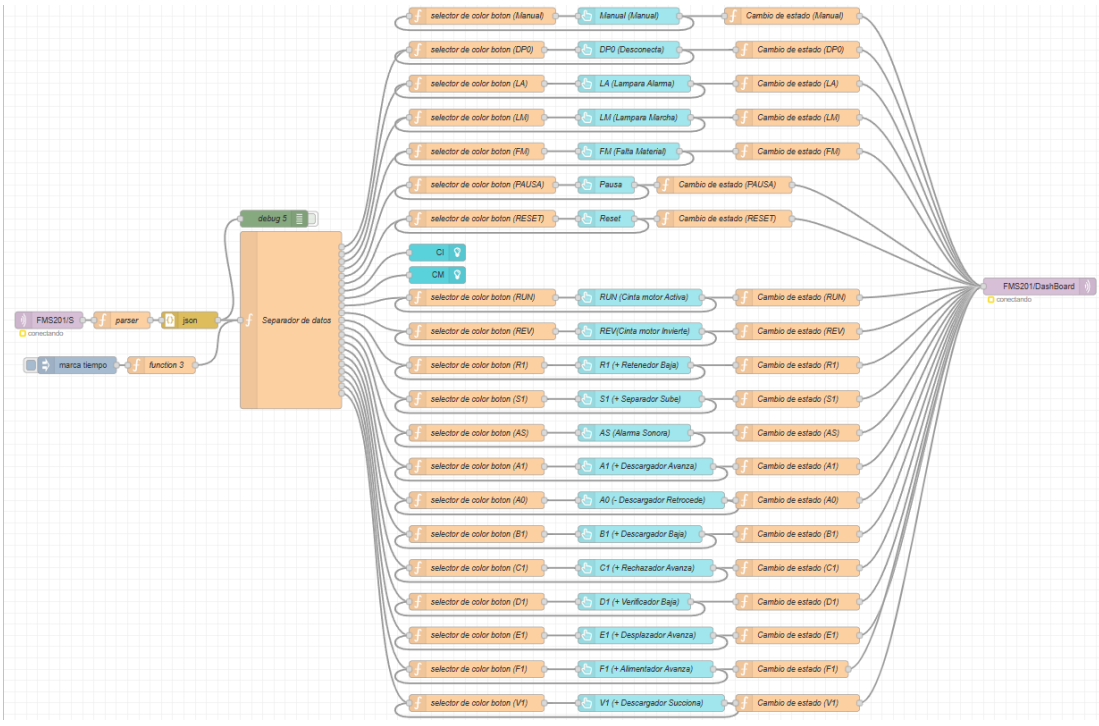

Figura 5.9. Flujo datos entrada/salida

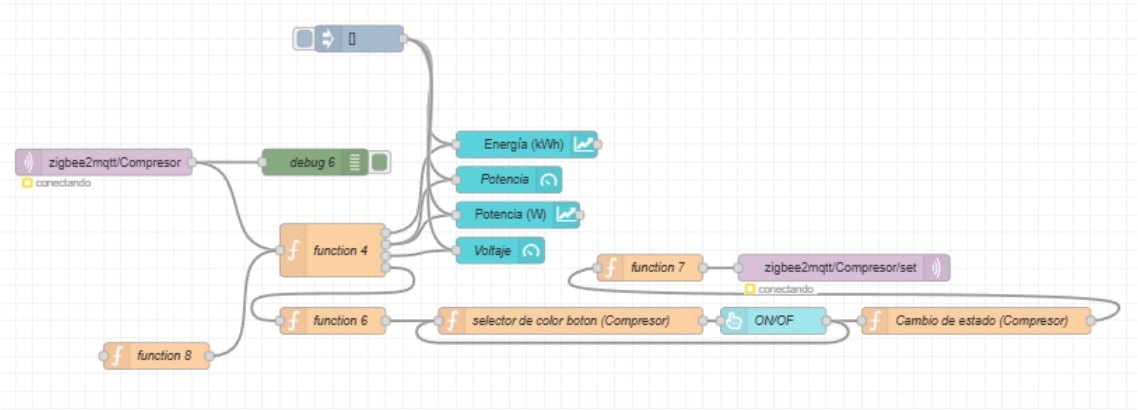

Figura 5.10. Flujo pestaña compresor

El procedimiento de muestra de información es muy sencillo: (i) el controlador de la estación envía por MQTT todos los datos en formato JSON, (ii) NodeRed los recibe y separa a través del nodo *function* y (iii) posteriormente los ilustra en el *dashboard*.

Para los botones es algo más complejo, ya que no solo deben mostrar el estado del elemento, sino que deben permitir cambiar su estado. Por tanto, existe una realimentación de información para almacenar el estado actual y el estado nuevo, para luego mandar el estado nuevo de manera correcta.

5.5. Pruebas realizadas en el aula

Una vez implementado el sistema en el entorno industrial de pruebas, concretamente en el aula Beckhoff de la Escuela de Ingenierías Industriales, se procedió a realizar una serie de pruebas funcionales y de integración. El objetivo principal era comprobar la correcta comunicación entre el PLC y el sistema de supervisión basado en NodeRED y así evaluar el comportamiento del sistema ante órdenes remotas y cambios en las variables de proceso.

Durante las pruebas, se validaron flujos de trabajo representativos como el arranque y parada de la estación desde el panel de control remoto, la lectura y publicación periódica de las variables del sistema a través de MQTT, y la correcta recepción e interpretación de comandos enviados desde el *dashboard*. También se verificó el comportamiento del sistema ante distintas condiciones de trabajo, simulando eventos como la activación del modo manual, la gestión de actuadores específicos (cinta, descargador, alimentador, etc.) y la visualización en tiempo real de los estados de los actuadores y sensores mediante el *dashboard* durante una prueba de funcionamiento en modo automático.

El sistema demostró una alta fiabilidad y robustez en este entorno, operando con precisión y sin pérdidas de comunicación. Las latencias fueron mínimas, incluso en situaciones de alta frecuencia de eventos. Además, se observaron ventajas en términos de ergonomía, ya que el uso del *dashboard* redujo la necesidad de interacción física con el sistema, facilitando tareas de supervisión y mantenimiento. Estas pruebas confirmaron que la arquitectura desarrollada es perfectamente aplicable a un entorno industrial real, con capacidad de adaptación a distintas estaciones y procesos.

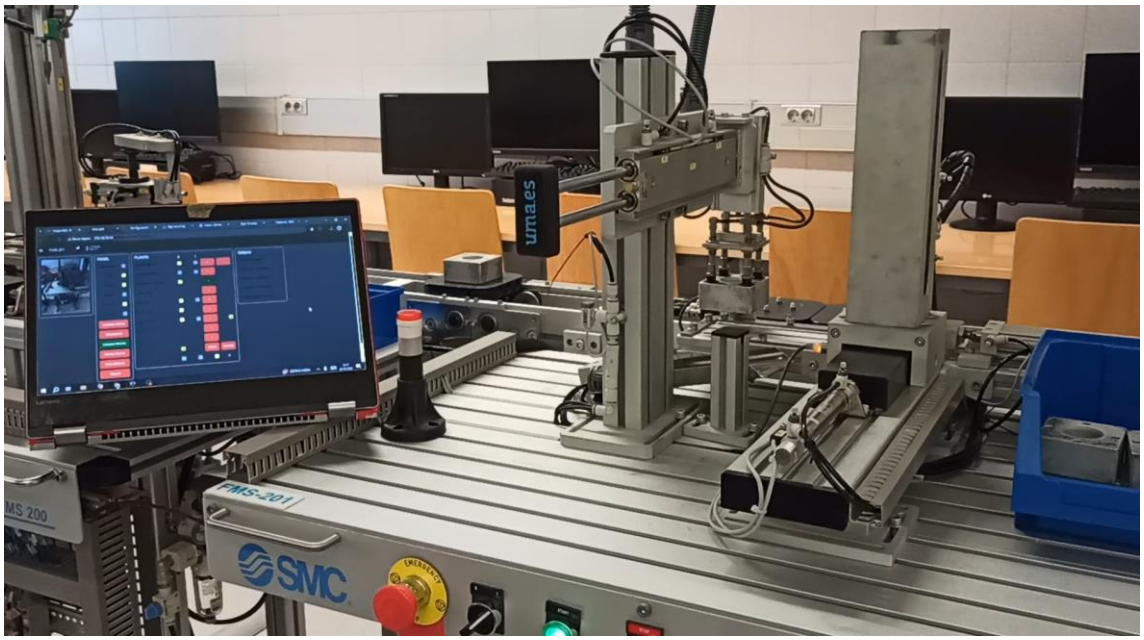

Figura 5.11 Pruebas de comunicación estación-sistema IoT

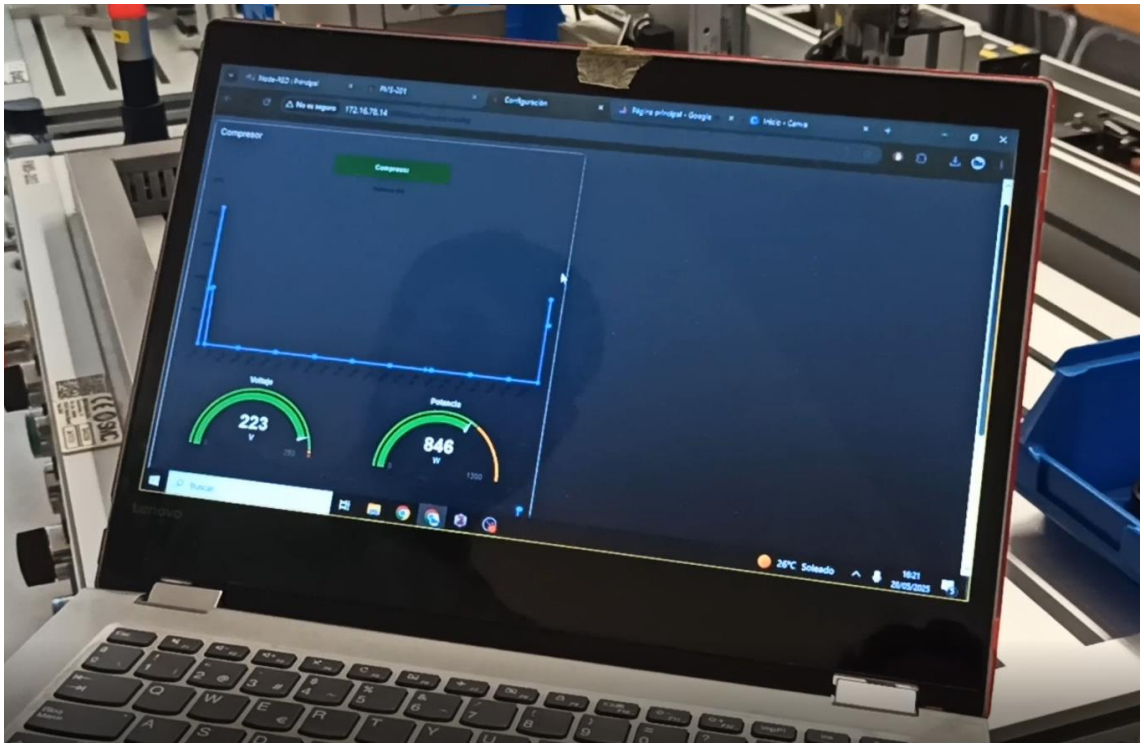

Figura 5.11. Pruebas dashboard compressor.

CAPÍTULO 6. CONCLUSIONES

6.1. Conclusiones

En este trabajo final de máster se ha desarrollado una solución versátil para el despliegue de una red IoT en dos entornos diferenciados: un escenario doméstico y otro industrial. Durante su desarrollo se ha demostrado la viabilidad y eficiencia de utilizar una tecnología basada en contenedores (Docker) para el desarrollo de este tipo de sistemas, integrando diferentes microservicios que proporcionan funcionalidades clave para IoT como la gestión de las comunicaciones, la creación de paneles de monitorización y actuación, o la especificación de escenas que controlen el comportamiento del sistema. Así, la elección de estos contenedores como tecnología base ha permitido el uso de diferentes microservicios como NodeRed, Home Assistant, una red Zigbee, MQTT, o WireGuard, entre otros. Además, gracias a la encapsulación de estos servicios, el sistema puede ser ejecutado en diferentes dispositivos sin comprometer el rendimiento o la estabilidad. De esta forma, el uso de contenedores ha propiciado que el sistema sea modular, escalable y de sencillo despliegue en diferentes entornos. Asimismo, hace viable la integración de sensores DIY, lo cual redunda en automatizaciones flexibles, personalizadas y muy económicas.

Un aspecto a destacar de este TFM es el haber sido validado en dos entornos distintos, adaptando la infraestructura a las necesidades de cada uno: un escenario más habitual situado en una casa con un conjunto de sensores y actuadores domóticos, y otro más industrial donde el sistema IoT incorpora comunicaciones remotas con un controlador lógico programable.

La instalación del sistema en el entorno doméstico ha supuesto un reto para el alumno, ya que se ha modificado parte de la instalación eléctrica y la selección de los dispositivos se ha hecho a base de prueba y error, viendo cuales son los dispositivos que mejor se adaptaban a las condiciones de la vivienda, espacio en las cajas de registro y derivación, instalación de nuevas tomas de corriente para sensores, etc. Las primeras modificaciones demandaron bastante tiempo y trabajo, pero una vez asimilados los pasos a seguir, resultó ser una tarea más sencilla y mecánica. En cambio, en el entorno industrial no hubo que modificar la instalación eléctrica, por lo que la incorporación del sistema ha sido más directa y requirió menos trabajo.

Tras realizar la validación en el hogar, el resultado obtenido ha sido satisfactorio, pudiéndose controlar dispositivos de la vivienda desde fuera y dentro del hogar con un terminal móvil. Asimismo, el sistema desplegado realiza tareas automatizadas como el control de luminarias o el ajuste del encendido del calentador según el precio de la electricidad a lo largo del día. Aunque quedan cosas por implementar para que el sistema controle la vivienda al 100%, por el momento el sistema está operando correctamente.

El sistema en el entorno industrial también ha sido validado correctamente, cumpliéndose los objetivos propuestos. Se ha logrado comunicar el PLC con un bróker mediante MQTT enviando el estado de los sensores y actuadores de la planta controlada, a la vez que se ha creado un panel de monitorización y control remoto de los actuadores de la misma. Además, este panel permite la integración de más dispositivos y estaciones FMS de manera sencilla, pudiéndose crear una interfaz capaz de controlar todas las máquinas desde el mismo terminal sin cambiar de aplicación o web.

Como conclusiones generales, se han alcanzado los resultados previstos y, a nivel personal, el alumno queda satisfecho con el trabajo realizado, tras una cantidad considerable de tiempo desarrollando el sistema y trabajando para conseguir la estabilidad de todos los servicios. El resultado final aporta una solución robusta y económica, siendo capaz de adaptarse a cualquier

entorno y evolucionar fácilmente mediante la incorporación de nuevos servicios y/o dispositivos cumpliendo los objetivos de este proyecto.

En el camino, el alumno ha adquirido conocimientos y habilidades en el desarrollo de este tipo de sistemas, lo que le supondrá una ventaja en su futuro (y presente) laboral. También ha disfrutado de la experiencia de trabajar con IoT en un entorno industrial, menos habitual que el doméstico.

6.2. Trabajos Futuros

A pesar de los resultados obtenidos, el sistema desarrollado presenta una gran cantidad de oportunidades de mejora y expansión. A continuación, se detallan trabajos futuros que se pueden realizar:

1. **Desarrollar una interfaz gráfica personalizada**, desarrollar un panel de usuario propio mediante frameworks como React, Angular, Dash en lugar de depender de Home Assistant.

2. **Incorporación de nuevos servicios**, como por ejemplo un servicio de base de datos, para almacenar históricos de los sensores y poder utilizarlo a futuro con herramientas de Machine Learning.

3. **Sistema de mantenimiento predictivo y alertas**, implementar análisis de datos (por ejemplo, Machine Learning) para generar avisos para realizar mantenimientos predictivos, identificando fallos o patrones, especialmente en entornos industriales.

4. **Mejorar la seguridad**, aunque el sistema tiene medidas de seguridad básicas como la VPN, sería interesante instalar más mecanismos de autenticación y certificados digitales.

5. **Incluir nuevos protocolos**, añadir soporte para nuevos protocolos de comunicación como KNX, BACnet o Modbus, facilitando la integración del sistema en entornos más profesionales.

6. **Orquestación avanzada de contenedores**, Integrar herramientas como Kubernetes permitiría escalar el sistema a una arquitectura más compleja, pero mejoraría la gestión de recursos y la tolerancia a fallos.

7. **Despliegue en entornos reales de mayor magnitud**, validar el funcionamiento del sistema en entornos industriales de mayor magnitud, para evaluar el desempeño y analizar las mejoras que se deben de aplicar.

BIBLIOGRAFÍA

[1] *¿Qué es IoT? - Explicación del Internet de las cosas - AWS*. (s. f.). Amazon Web Services, Inc. Recuperado 27 de diciembre de 2024, de https://aws.amazon.com/es/what-is/iot/

[2] *¿Qué es internet de las cosas? | IoT explicada*. (s. f.). SAP. Recuperado 15 de diciembre de 2024, de https://www.sap.com/spain/products/artificial-intelligence/what-is-iot.html

[3] *Qué es la domótica, cómo se aplica y cuáles son sus beneficios*. (s. f.). Ferrovial. Recuperado 27 de diciembre de 2024, de https://www.ferrovial.com/es/recursos/domotica/

[4] *Domótica | Idae*. (s. f.). Recuperado 27 de diciembre de 2024, de https://www.idae.es/tecnologias/eficiencia-energetica/edificacion/domotica

[5] *Alexa Smart Home—Learn about Home Automation | Amazon.com*. (s. f.). Recuperado 15 de diciembre de 2024, de https://www.amazon.com/alexa-smart-home/b?ie=UTF8&node=21442899011&ref=pe_alxhub_aucc_en_us_IC_HP_1_HUB_SMA

[6] *Gestiona tu hogar inteligente con Google Home | Google Home*. (s. f.). Recuperado 15 de diciembre de 2024, de https://home.google.com/intl/es_es/welcome/

[7] *¿Qué es IIOT? Internet Industrial de las Cosas—Iberdrola*. (s. f.). Recuperado 27 de diciembre de 2024, de https://www.iberdrola.com/innovacion/que-es-iiot

[8] *Home*. (s. f.). Industry IoT Consortium. Recuperado 28 de diciembre de 2024, de https://www.iiconsortium.org/

[9] *An inside look at the drivers for Industry 4.0| Nokia*. (s. f.). Recuperado 28 de diciembre de 2024, de https://www.nokia.com/thought-leadership/articles/industry-4-0/three-key-drivers-for-success/

[10] *Qué es la Inteligencia Artificial | Plan de Recuperación, Transformación y Resiliencia Gobierno de España.* (s. f.). Recuperado 28 de diciembre de 2024, de https://planderecuperacion.gob.es/noticias/que-es-inteligencia-artificial-ia-prtr

[11] *¿Qué es el machine learning y que usos tiene?* (s. f.). REPSOL. Recuperado 28 de diciembre de 2024, de https://www.repsol.com/es/energia-futuro/tecnologia-innovacion/machine-learning/index.cshtml

[12] *¿Qué son los microservicios? | AWS*. (s. f.). Amazon Web Services, Inc. Recuperado 28 de diciembre de 2024, de https://aws.amazon.com/es/microservices/

[13] *Explicación sobre los contenedores: Concepto e importancia*. (s. f.). Recuperado 30 de diciembre de 2024, de https://www.redhat.com/es/topics/containers

[14] *Docker y microservicios—Blog de hiberus*. (s. f.). Recuperado 15 de diciembre de 2024, de https://www.hiberus.com/crecemos-contigo/docker-y-microservicios/

[15] *¿Qué es Docker y cómo funciona? Ventajas de los contenedores Docker*. (s. f.). Recuperado 30 de diciembre de 2024, de https://www.redhat.com/es/topics/containers/what-is-docker

[16] *VM vs Containers*. (s. f.). Comparativa Clouds. Recuperado 30 de diciembre de 2024, de https://comparacloud.com/servicios-clouds/precios-cloud/

[17] Admin-factoria. (2024, septiembre 9). Docker: Qué es y cómo funciona la contenerización de aplicaciones. *Ausum Cloud: Servicios y Soluciones Cloud*. Recuperado 15 de diciembre de 2024, de https://ausum.cloud/docker-que-es-como-funciona-la-contenerizacion-de-aplicaciones/

[18] *¿Qué es Node-RED y para qué sirve? | Actualidad de Grupo Sinelec*. (2021, febrero 4). Recuperado 15 de diciembre de 2024, de https://blog.gruposinelec.com/actualidad/que-es-node-red-y-para-que-sirve/

[19] *Running under Docker: Node-RED*. (s. f.). Recuperado 21 de febrero de 2025, de https://nodered.org/docs/getting-started/docker

[20] Assistant, H. (s. f.). *Home Assistant*. Home Assistant. Recuperado 21 de febrero de 2025, de https://www.home-assistant.io/

[21] *linuxserver/homeassistant—Docker Image | Docker Hub*. (s. f.). Recuperado 21 de febrero de 2025, de https://hub.docker.com/r/linuxserver/homeassistant

[22] *Docker Hub Container Image Library | App Containerization*. (s. f.). Recuperado 15 de diciembre de 2024, de https://hub.docker.com

[23] *MQTT Essentials—All Core Concepts Explained*. (s. f.). Recuperado 19 de febrero de 2025, de https://www.hivemq.com/mqtt/

[24] *¿Qué es MQTT? Definición y detalles*. (s. f.). Recuperado 15 de diciembre de 2024, de https://www.paessler.com/es/it-explained/mqtt

[25] *Eclipse-mosquitto—Official Image | Docker Hub*. (s. f.). Recuperado 19 de febrero de 2025, de https://hub.docker.com/_/eclipse-mosquitto

[26] *Ancho de banda: Definición y detalles*. (s. f.). Recuperado 20 de febrero de 2025, de https://www.paessler.com/es/it-explained/bandwidth

[27] TEKTELIC. (2023, octubre 23). LoRaWAN vs Zigbee for Your IoT Project | TEKTELIC Blog. *TEKTELIC*. https://tektelic.com/expertise/lorawan-vs-zigbee/

[28] fgarcia. (2018, abril 19). *⇨ Descubre qué es Zigbee y para qué se utiliza*. efectoLED. https://www.efectoled.com/blog/es/que-es-zigbee/

[29] *¿Qué es Zigbee? La tecnología de red eléctrica inteligente más popular del mundo | Homey*. (s. f.). Recuperado 15 de diciembre de 2024, de https://homey.app/es-es/wiki/que-es-zigbee/

[30] *zzh Multiprotocol RF Stick (CC2652R1—External Antenna)*. (s. f.). Electrolama. Recuperado 30 de diciembre de 2024, de https://electrolama.com/products/zzh-multiprotocol-rf-stick

[31] Zhang, L. (2021, septiembre 18). *SONOFF Zigbee 3.0 USB Dongle Plus-P*. SONOFF Official. https://sonoff.tech/product/gateway-and-sensors/sonoff-zigbee-3-0-usb-dongle-plus-p/

[32] *LAUNCHXL-CC1352P Evaluation board | TI.com*. (s. f.). Recuperado 30 de diciembre de 2024, de https://www.ti.com/tool/LAUNCHXL-CC1352P

[33] *SONOFF ZBDongle-E control via MQTT | Zigbee2MQTT*. (s. f.). Recuperado 21 de febrero de 2025, de https://www.zigbee2mqtt.io/devices/ZBDongle-E.html

[34] *Zigbee2MQTT*. Supported devices(s. f.). Recuperado 24 de febrero de 2025, de https://www.zigbee2mqtt.io/supported-devices/

[35] *GY-2561 TSL2561 Light intensity module Sensor module Super intensity module—AliExpress 502*. (s. f.). Recuperado 24 de febrero de 2025, de https://www.aliexpress.com/item/1005006047383759.html

[36] *Tuya ZigBee Light Sensor Luminance Sensor ZigBee Beam Sensor Smart Home Automation App Control Tuya Light Illumination Detector—AliExpress 44*. (s. f.). Aliexpress. Recuperado 24 de febrero de 2025, de https://www.aliexpress.com/item/1005008259438794.html

[37] *Arduino Light Sensor—TSL2561 and Experiments with Infrared and Visible Light*. (2022, noviembre 28). Maker Portal. https://makersportal.com/blog/2018/4/19/arduino-light-sensor-tsl2561-and-experiments-with-infrared-and-visible-light

[38] *MQ-2 MQ2 Smoke Gas LPG Butane Hydrogen Gas Sensor Detector Module For arduino—AliExpress 502*. (s. f.). Recuperado 24 de febrero de 2025, de https://www.aliexpress.com/item/32328262269.html

[39] *Aqara JT-BZ-01AQ/A control via MQTT | Zigbee2MQTT*. (s. f.). Recuperado 24 de febrero de 2025, de https://www.zigbee2mqtt.io/devices/JT-BZ-01AQ_A.html

[40] Llamas, L. (s. f.). *Detector de gases con Arduino y la familia de sensores MQ*. Luis Llamas. Recuperado 24 de febrero de 2025, de https://www.luisllamas.es/arduino-detector-gas-mq/

[41] *Tuya ZG-204ZM control via MQTT | Zigbee2MQTT*. (s. f.). Recuperado 24 de febrero de 2025, de https://www.zigbee2mqtt.io/devices/ZG-204ZM.html

[42] *HC-SR501 Adjust IR Pyroelectric Infrared PIR Human Infrared Motion Sensor Detector Module for Arduino for Raspberry Pi + Case—AliExpress 502*. (s. f.). Aliexpress. Recuperado 19 de marzo de 2025, de https://www.aliexpress.com/item/1005007558216824.html

[43] *Aqara WSDCGQ11LM control via MQTT | Zigbee2MQTT*. (s. f.). Recuperado 26 de marzo de 2025, de https://www.zigbee2mqtt.io/devices/WSDCGQ11LM.html

[44] *Aqara WSDCGQ12LM control via MQTT | Zigbee2MQTT*. (s. f.). Recuperado 26 de marzo de 2025, de https://www.zigbee2mqtt.io/devices/WSDCGQ12LM.html

[45] *Adafruit Industries. (2025). DHT11, DHT22 and AM2302 Sensors. Mouser Electronics.* Recuperado el 26 de marzo de 2025, de https://www.mouser.com/datasheet/2/737/dht-932870.pdf?srsltid=AfmBOorFtxj-tg-RMZh4AiPynWgyEky3G_j_MJVvKmuZxrg-HookpCHe

[46] *Tuya RB-SRAIN01 control via MQTT | Zigbee2MQTT*. (s. f.). Recuperado 26 de marzo de 2025, de https://www.zigbee2mqtt.io/devices/RB-SRAIN01.html

[47] *Arduino y el sensor de agua – Prometec*. (s. f.). Recuperado 26 de marzo de 2025, de https://www.prometec.net/sensor-agua/

[48] *AVATTO Tuya WiFi Zigbee Smart Light Switch Module,No Neutral Wire 2 Ways Control Mini DIY Breaker Work for Alexa, google home—AliExpress 13*. (s. f.). Recuperado 26 de marzo de 2025, de https://www.aliexpress.com/item/1005004823480465.html

[49] *Peely Bot|smart Fingerbot Plus—Zigbee & Bluetooth-compatible, Alexa & Google Home Compatible*. (s. f.). Aliexpress. Recuperado 26 de marzo de 2025, de https://www.aliexpress.com/item/1005005168070181.html

[50] *Tuya Zigbee Smart IR Remote Control Universal Infrared Remote for Smart Home for AC TV DVD works with Alexa Google Home—AliExpress*. (s. f.). Aliexpress. Recuperado 26 de marzo de 2025, de https://www.aliexpress.com/item/1005003878194474.html

[51] *Enchufe inteligente Tuya WIFI EU 16A/20A Wifi/Zigbee, monitoreo de potencia, Control de hogar inteligente, compatible con Google Home, Alexa—AliExpress 44*. (s. f.). aliexpress. Recuperado 26 de marzo de 2025, de https://es.aliexpress.com/item/1005007055724435.html

[52] *¿Qué es una red privada virtual o VPN? ¿Por qué debería usar una red privada virtual o VPN? | Microsoft Azure*. (s. f.). Recuperado 30 de marzo de 2025, de https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-vpn

[53] *Donenfeld, J. A. (s. f.). WireGuard: Fast, modern, secure VPN tunnel*. Recuperado 30 de marzo de 2025, de https://www.wireguard.com/

[54] *FMS-200—Sistema didáctico modular de ensamblaje flexible*. (s. f.). Recuperado 27 de mayo de 2025, de https://www.smctraining.com/es/webpage/indexpage/287

[55] *SMC International Training*. (s. f.). *FMS-200: Catálogo de estaciones. Estación FMS-201 – Alimentación de la base* [PDF]. SMC International Training.

[56] *Introducción al estándar IEC 61131-3.pdf*. (s. f.). *infoPLC*. Recuperado 27 de mayo de 2025, de https://drive.google.com/file/d/1XE6I2xJOfW1WQpoFzB247VSwxr6YlHft/

[57] *Germany, B. A. G. & C. K., Hülshorstweg 20, 33415 Verl.* (s. f.). *TwinCAT 3 PLC. Beckhoff Automation*. Recuperado 27 de mayo de 2025, de https://www.beckhoff.com/es-es/support/training-offerings/twincat-3-plc/

[58] Grafana OSS | Leading observability tool for visualizations & dashboards. (s. f.). *Grafana Labs*. Recuperado 27 de mayo de 2025, de https://grafana.com/oss/grafana/

[59] *Node-RED Dashboard 2.0*. (s. f.). Recuperado 2 de febrero de 2025, de https://dashboard.flowfuse.com/

ANEXO I.

Instalación de los microservicios.

Instalación de NodeRed ¡Error! Marcador no definido.

Para la instalación de NodeRed se ha usado Docker, creando un contenedor que lo ejecuta de la siguiente manera:

1. Crear un *Docker Compose* mediante un archivo con el nombre "*docker-compose.yml"*, donde se irán añadiendo los diferentes servicios que componen el sistema IoT.
   Se copia el *Docker Compose* que se proporciona en el foro de NodeRed [19] para descargar la imagen de NodeRed que ya existe.

2. Se ejecuta por consola "`docker-compose up`", para descargar la imagen en el servidor y levantar el servicio. Esperar la descarga de la imagen y comprobar su correcto funcionamiento. Para ello se entra desde un navegador al enlace: http://localhost:1880 (si se está ejecutando en una máquina diferente a la local se debe de cambiar *localhost* por la dirección IP de la máquina).

3. Una vez levantado, configuraremos las paletas según las necesidades. Para este sistema se han añadido las siguientes paletas:

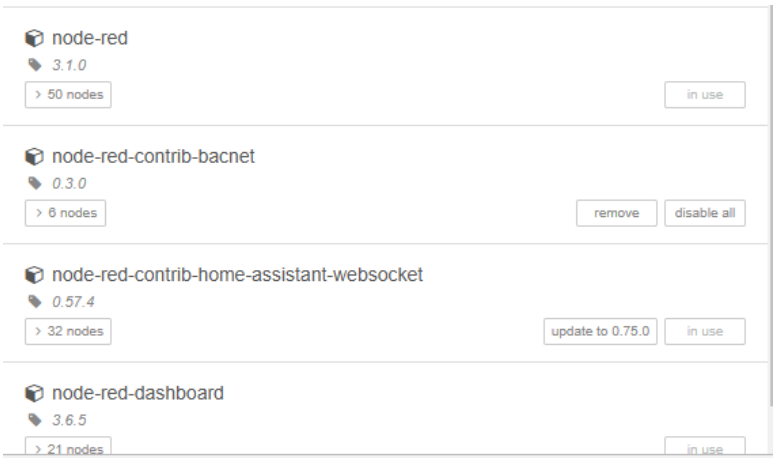

Figura A1.1. Paletas NodeRed instaladas

Instalación Home Assistant

Para la instalación de HA, haremos uso de nuevo de un contenedor de Docker con la imagen de Home Assistant. Los pasos a seguir son los siguientes:

1. Buscar la imagen de HA en el foro de *Docker Hub* [21].

2. Usar el *Docker Compose* que se muestra en la documentación, añadiéndolo al "*docker-compose.yml"* ya creado en el apartado anterior.

3. Levantar el *Docker Compose* ejecutando en la terminal *docker-compose up*, como en la sección anterior.

   Esperar la descarga de la imagen y comprobar su correcto funcionamiento. Para ello se entra desde un navegador al enlace: http://localhost:8123 (si se está ejecutando en una máquina diferente a la local se debe de cambiar *localhost* por la dirección IP de la máquina).

4. Para finalizar, se deben de descargar las APIS de NodeRed y MQTT. Para ello, se entra en Ajustes → Dispositivos y Servicios → Añadir integración. Una vez aparezca el menú, se busca MQTT y NodeRed y se instalan (ver Figura A1.2).

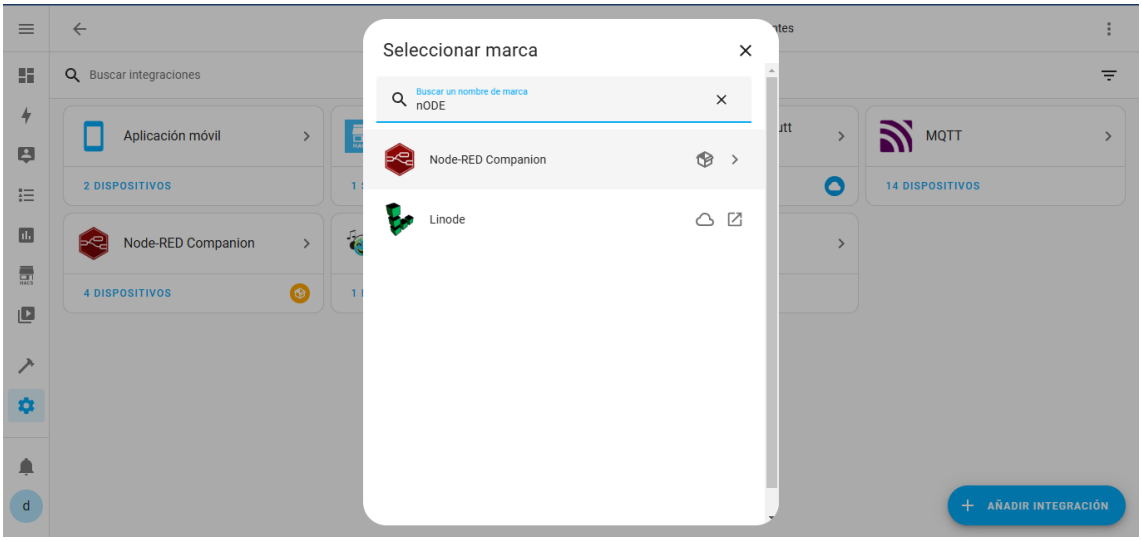

Figura A1.2. Integraciones necesarias Home Assistant

Instalación Bróker MQTT

El *broker* de MQTT más popular es Mosquitto, que posee una comunidad de desarrollo bastante grande y, gracias a su popularidad, existe una imagen de Docker con este bróker instalado. Por tanto, para instalar Mosquitto se deben seguir los siguientes pasos.

1. Buscar la imagen de Mosquitto en *Docker Hub* [25].

2. Añadir el servicio, siguiendo la estructura de los otros servicios anteriores, al documento "*docker-compose.yml*".

3. Levantar el servicio para descargar la imagen de la misma manera que para los anteriores servicios.

4. Probar el funcionamiento del servicio, realizamos el siguiente procedimiento:
    a. Entrar en NodeRed.
    b. Añadir un nodo de "Inyección" (ver Figura A1.3).
    c. Añadir un nodo de MQTT de publicación y otro de suscripción, y configurar ambos nodos añadiendo el mismo *topic* (ver Figura A1.4).
    d. Añadir un nodo de "*Debug*", el cual sirve para mostrar por en un panel de NodeRed la información transmitida entre nodos.
    e. Por último, desplegar el flujo y mandar un dato con el nodo "Inyección". Si Mosquitto está funcionando correctamente se debería ver el mensaje en el panel de "*Debug*" (ver Figura A1.5).

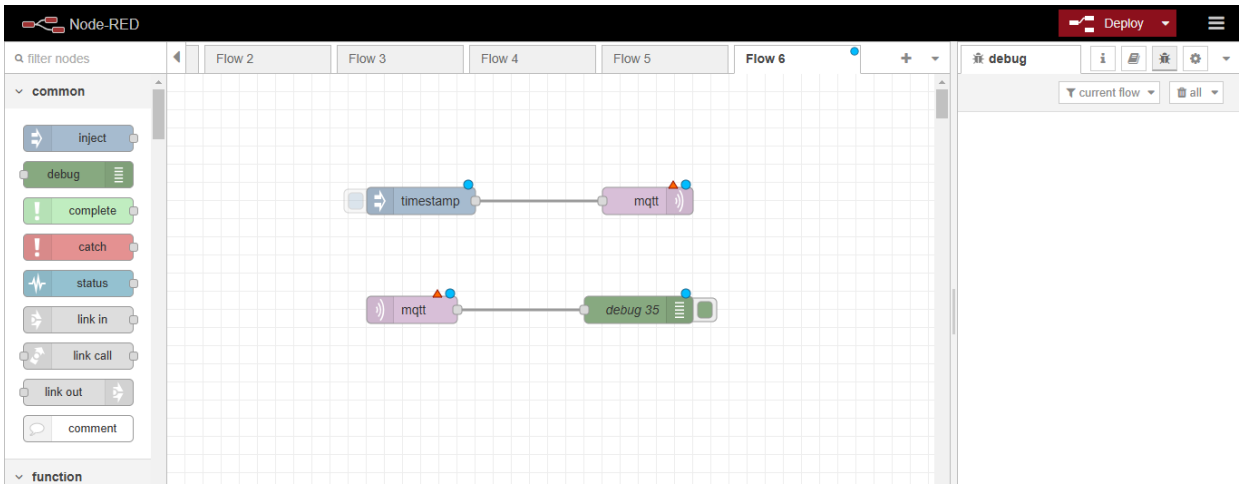

Figura A1.3. Primer paso para probar MQTT

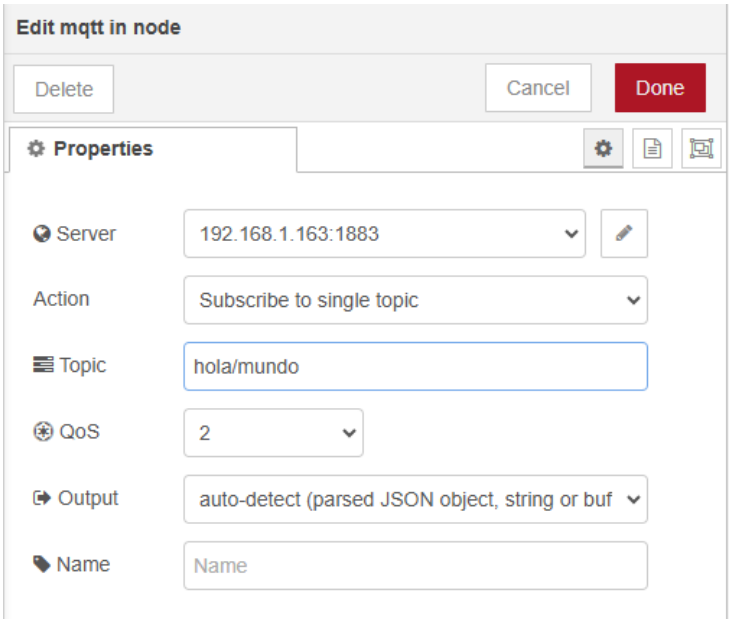

Figura A1.4. Configuración bróker MQTT

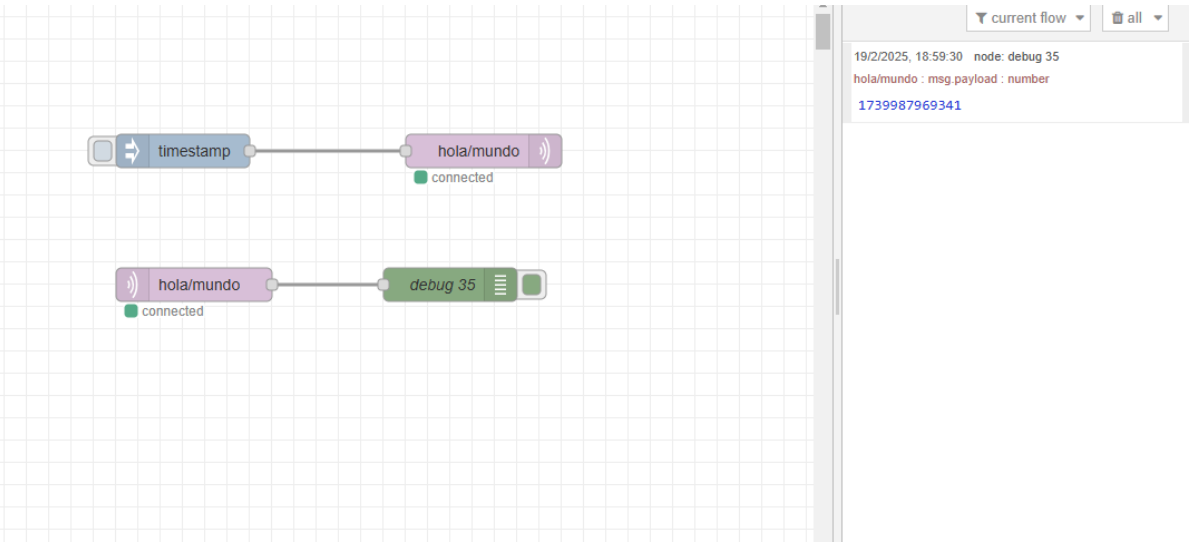

Figura A1.5. Prueba conexión MQTT

Instalación Red Zigbee

La instalación de una red Zigbee es algo más complejo que los servicios instalados anteriormente. Se necesita instalar de forma similar la imagen de una red Zigbee en un contenedor de docker, pero hay que sincronizar el contenedor de docker con una antena que será la encargada de dar cobertura a la red.

El primer paso consiste en seleccionar la antena adecuada, entre las opciones que se analizarán en el apartado 3.4. En dicho apartado se explicará la elección de la antena SONOFF, optimizada específicamente para su uso en redes Zigbee.

A continuación, hay que *flashear* la antena para cargar el *firmware* del coordinador Zigbee. Para realizar esta tarea hay que descargar primero el firmware correspondiente al modelo de la antena [33]. Posteriormente, se instala en el ordenador el software "*Flash Programmer 2*" de Texas Instruments.

El siguiente paso es conectar el dongle en modo "*bootloader*", lo que se consigue pulsando 10s el botón que se muestra en la siguiente figura.

Figura A1.6. Dongle Zigbee

Por último, se carga el binario en el programa, se inicia el proceso de flasheo y se espera a que finalice. Si todo ha ido bien no debería de salir ningún error.

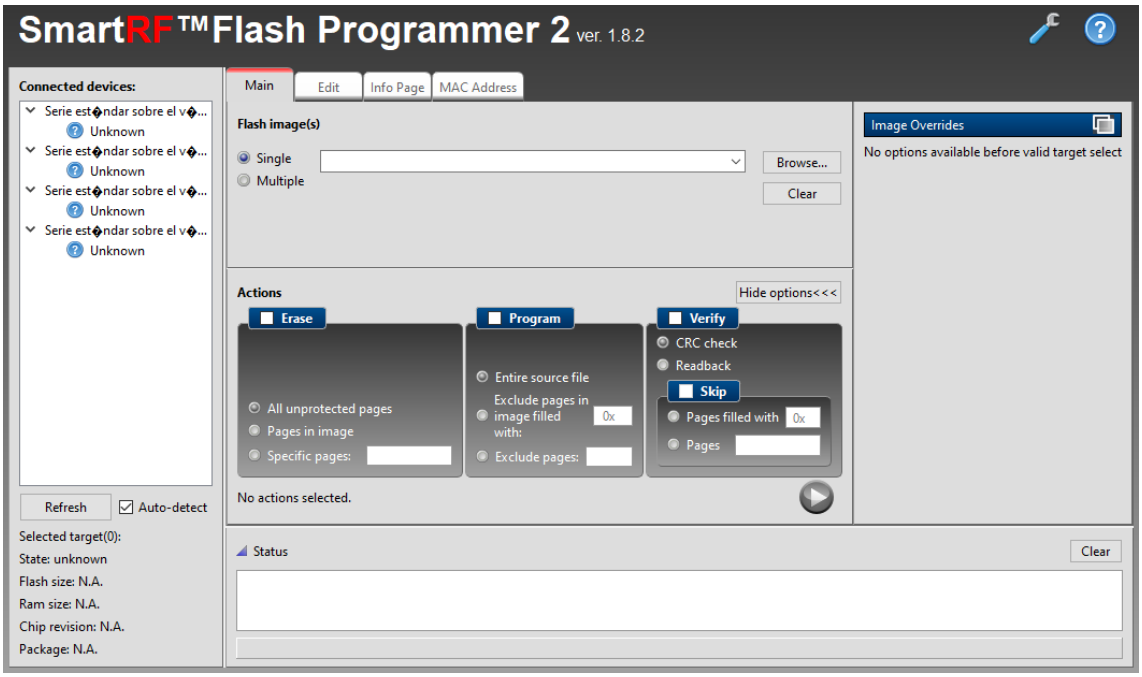

Figura A1.7. Programa Texas Instrument

Una vez se tenga flasheado el dongle, se añade al "*docker-compose.yml*" el servicio que coordinará la red de Zigbee. El servicio encargado de coordinar Zigbee es Zigbee2MQTT, que permite coordinar la red Zigbee además de traducir los mensajes de la red a mensajes MQTT, facilitando así la integración de la red con demás servicios.

Una vez se haya descargado la imagen, en el documento docker-compose se tiene que configurar la antena que anteriormente se ha flasheado. Para ello, hay que asegurarse de que la localización del dongle sea el correcto:

1. Se abre una terminal y se ejecuta:
   `ls /dev/serial/by-id/`
   Este comando devuelve el puerto donde se localiza el dongle.

2. Añadimos el puerto en el docker compose, en este caso el puerto es el por defecto: "*/dev/ttyUSB0*"

3. Por último, se levanta el servicio para descargar la imagen y que se genere la red de Zigbee. Para comprobar el correcto funcionamiento se puede entrar en la interfaz gráfica de la red a través del puerto 8080.

Archivo “docker-compose.yml”

A continuación, se muestra el archivo “*docker-compose.yml*” resultante:

```
version: '3.8'


services:
  homeassistant:
    container_name: homeassistant
    image: "ghcr.io/home-assistant/home-assistant:latest"
    volumes:
      - ../datafiles/ha/config:/config
      - ../datafiles/ha/etc/localtime:/etc/localtime:ro
    restart: unless-stopped
    ports:
      - 8123:8123
    privileged: true


  node-red:
    image: nodered/node-red:latest
    environment:
      - TZ=Europe/Amsterdam
    ports:
      - "1880:1880"
    networks:
      - node-red-net
    volumes:
      - ../datafiles/node-red-data:/data
    restart: unless-stopped
```

```
 mosquitto:
    image: eclipse-mosquitto:latest
    container_name: mosquitto
    environment:
      - TZ=Europe/Madrid
    volumes:
      - ../datafiles/mosquitto/config:/mosquitto/config
      - ../datafiles/mosquitto/data:/mosquitto/data
      - ../datafiles/mosquitto/log:/mosquitto/log
    ports:
      - 1883:1883
      - 9001:9001
    restart: unless-stopped


  zigbee2mqtt:
      container_name: zigbee2mqtt
      image: koenkk/zigbee2mqtt
      restart: unless-stopped
      volumes:
        - ../datafiles/zigbee2mqtt/data:/app/data
        - /run/udev:/run/udev:ro
      ports:
        - 8080:8080
      environment:
        - TZ=Europe/Berlin
      devices:
        - /dev/ttyUSB0:/dev/ttyUSB0


volumes:
  node-red-data:
networks:
  node-red-net:
```