

Diseño y construcción de un sistema para adquisición y  
análisis del consumo energético en el hogar

Jesús Sánchez de Lechina Tejada

Junio 2020



**UNIVERSIDAD  
DE GRANADA**

# Índice

<b>1. Introducción</b>	<b>4</b>
1.1. Objetivos y motivación . . . . .	4
1.2. Estructura del proyecto . . . . .	4
1.3. Internet de las Cosas . . . . .	6
1.3.1. ¿Qué es un dispositivo IoT? . . . . .	6
<b>2. Revisión de enchufes inteligentes existentes en el mercado</b>	<b>8</b>
2.1. ¿Qué es un enchufe? Tipos de enchufe . . . . .	8
2.2. ¿Qué es un enchufe inteligente? . . . . .	9
2.3. Funcionalidades comunes que ofertan los enchufes electrónicos actualmente . . . . .	9
2.3.1. Monitorización: . . . . .	9
2.3.2. Control: . . . . .	9
2.4. Tipos de enchufes . . . . .	10
2.5. Modelos en el mercado . . . . .	10
<b>3. Arquitecturas IoT</b>	<b>13</b>
3.1. Protocolos . . . . .	13
3.2. Modelos de interconexión de red . . . . .	14
<b>4. Diseño de la arquitectura de nuestro sistema</b>	<b>16</b>
4.1. Descripción general . . . . .	16
4.2. Modelo de red usado . . . . .	17
4.3. Protocolo MQTT . . . . .	18
4.4. Diseño broker, publishers, subscribers . . . . .	19
4.4.1. Mosquitto . . . . .	21
4.5. Enchufe . . . . .	22
4.5.1. Diseño y construcción del enchufe . . . . .	23
4.5.2. Programa del microcontrolador . . . . .	28
4.6. Servidor central . . . . .	30
4.6.1. Configuración de red . . . . .	30
4.6.2. Gestión de la base de datos . . . . .	31

4.6.3. Consulta a la base de datos . . . . .	32
4.7. Aplicación de escritorio . . . . .	33
<b>5. Pruebas y test</b>	<b>38</b>
5.1. Pruebas en el módulo WiFi . . . . .	38
5.2. Pruebas en el servidor central . . . . .	40
5.3. Pruebas en la aplicación de escritorio . . . . .	41
<b>6. Conclusión</b>	<b>42</b>
<b>A. Anexo: Código del proyecto</b>	<b>44</b>
<b>B. Anexo: Cómo usar este proyecto</b>	<b>45</b>
B.1. Enchufe Inteligente . . . . .	45
B.2. Servidor central . . . . .	45
B.3. Cliente de escritorio . . . . .	46

# 1. Introducción

En este documento se recoge una descripción de mi Proyecto de Fin de Grado, cuyo tema es: **Diseño y construcción de un sistema para adquisición y análisis del consumo energético en el hogar.**

Para ayudar en esta descripción se han reunido un conjunto de explicaciones de los procesos empleados, análisis y el desarrollo de algunas cuestiones relativas al proyecto.

## 1.1. Objetivos y motivación

El **principal objetivo** de este proyecto es doble. Del mismo modo que se pretende diseñar un sistema para medir y controlar el consumo energético en un hogar en sus distintos aspectos (hardware, software y telecomunicaciones necesarias) se busca también hacer ver al lector que, con ligeros conocimientos informáticos y la disposición por aprender adecuada, cualquier individuo puede llegar a replicar el sistema o incluso construir uno nuevo basándose en la descripción que se ofrece de este diseño.

Analizando los requisitos del objetivo principal encontramos una necesidad adicional: El sistema a diseñar requerirá una infraestructura y un paradigma sobre el cual planificar el proyecto. Esto resulta ser en realidad una ocasión para estudiar el Internet de las Cosas, explicando qué es y cómo en la actualidad ya es una solución a este problema aunque nos pueda parecer algo distante.

## 1.2. Estructura del proyecto

- **Sección 1:** Presentación del proyecto, motivación y objetivos del mismo. Se ofrece una introducción al Internet de las Cosas en el Apartado 1.3, así se podrá conocer qué significa este término y qué aplicaciones tiene actualmente. Al final de esta sección se motiva el núcleo de este proyecto mediante una de las aplicaciones del Internet de las Cosas, la monitorización.
- **Sección 2:** En la Sección 2 se enlaza el concepto de Internet de las Cosas con la aplicación a los enchufes inteligentes, haciendo una pequeña taxonomía sobre estos y presentando algunos modelos actuales en el mercado con sus funcionalidades.
- **Sección 3:** Previo a la descripción en detalle de la construcción del sistema es conveniente conocer el contexto de las arquitecturas en el paradigma del Internet de las Cosas. Este es el objetivo que trata la Sección 3, dar una visión

sobre cómo se puede dar soporte al problema que nos planteamos.

- **Sección 4:** Una vez explicada esta arquitectura pasamos a, en la Sección 4, profundizar en el uso que le hemos dado a distintos elementos de este paradigma para adecuarlos a las necesidades de nuestro proyecto. Explicando cómo hemos diseñado y construido todo el sistema desde la elección de sensores y microcontrolador hasta el diseño de la aplicación del usuario final pasando por toda la lógica de comunicación y almacenamiento de información intermedia.
- **Sección 5:** Recopilación de pruebas que se han usado para verificar el correcto desarrollo del proyecto.
- **Sección 6:** Reflexiones, conclusiones y análisis sobre el conjunto del resultado del proyecto.

### 1.3. Internet de las Cosas

Uno de los términos más recurrentes en el ámbito de la tecnología a día de hoy es el de “*Internet de las Cosas*” (en inglés, IoT, *Internet of Things*). Pero, puesto que el término puede resultar ambiguo, es conveniente dar una definición sobre esta que permita esclarecer el concepto.

El Internet de las Cosas es un paradigma tecnológico en sí mismo. Desglosando el concepto vemos que reúne dos tecnologías. Por un lado parte de “las cosas”, que abarca desde los dispositivos y electrodomésticos que podemos encontrar en nuestro día a día; que tienen una funcionalidad en sí misma (p.ej. una televisión, un frigorífico), hasta incluso podemos abstraer a personas o animales (una persona con un marcapasos, un ave con un geolocalizador u otros ejemplos<sup>[1]</sup>). A estas “cosas” se les añade el término “Internet”, que no es más que una abstracción de la conectividad que permite dotar a las cosas de comunicación con el exterior, extendiendo sus funcionalidades sin privarlas de su cometido original. Esta es su principal característica, la posibilidad de comunicación con otras “cosas” mediante internet y sin la necesidad de intervención humana.

La idea que trasciende de esto es que cualquier cosa o dispositivo que usemos habitualmente puede conectarse a una red, a internet. Haciendo que, en términos de redes, un dispositivo IoT se pueda equiparar a un ordenador convencional.

#### 1.3.1. ¿Qué es un dispositivo IoT?

En el símil anterior hemos adelantado el concepto de dispositivo IoT sin llegar a definirlo completamente. A continuación explicaremos qué es concretamente un dispositivo IoT y qué lo diferencia de otras “cosas”.

Un dispositivo IoT es una entidad (objeto, o incluso animales o personas) que tiene una funcionalidad en sí misma y a la cual dotamos de una capacidad de conexión y telecomunicación. De modo que sea capaz de, por sí misma, comunicarse con otros dispositivos en su entorno dotados de esta misma capacidad.

Los beneficios que podemos obtener de este paradigma se pueden aplicar en muchas áreas<sup>[2]</sup>, por ejemplo:

- Logística: Rastreo de envíos por correo o estado del stock en almacenes automatizando lectura de los items mediante WiFi.
- Transporte: Gestión automática de rutas por GPS; captura y procesado de infracciones de velocidad.
- Salud: Desde una pulsera que capte tus hábitos de vida hasta la supervisión de

personas ancianas que viven solas.

- Monitorización: Desde instalación de sensores en un bosque para medir datos de contaminación hasta medir en un hogar el consumo de un electrodoméstico.

Este último ejemplo es justo la motivación de nuestro trabajo. Podemos usar esta tecnología para crear un **enchufe inteligente** que nos permita saber nuestro consumo eléctrico y controlar el uso que le damos.

## 2. Revisión de enchufes inteligentes existentes en el mercado

### 2.1. ¿Qué es un enchufe? Tipos de enchufe

Un enchufe es una fuente de alimentación entre sistemas eléctricos. En este proyecto trataremos los enchufes dentro del ámbito doméstico o comercial frente al ámbito industrial, que tiende a trabajar con voltajes de un orden muy superior.

Un enchufe en el ámbito doméstico permite la conexión entre un dispositivo electrónico y la corriente alterna para la alimentación de este dispositivo.

Estos trabajan con unos voltajes entre 100V y 240V.<sup>[3]</sup> El tipo de enchufe, voltaje y frecuencia usados en una región geográfica vienen determinados por un convenio fijado por el gobierno de esa zona.

Por razones históricas<sup>[4]</sup> no existe una estandarización. Generalmente las zonas que se han desarrollado con una mayor influencia entre los países que la conforman tienden a compartir el mismo tipo de enchufe, por ejemplo el más frecuentemente usado en la Unión Europea es distinto del usado en Reino Unido a pesar de compartir las propiedades de frecuencia y potencial eléctrico y situarse geográficamente más cercanos. Otras regiones más distantes como Japón o Estados Unidos de América usan enchufes con características distintas a ambas. Se ofrecen algunos ejemplos de las propiedades de los enchufes por países en el Cuadro 1.

País	Potencial Eléctrico	Frecuencia
Indonesia	110V, 220V	50Hz
España	230V	50Hz
Reino Unido	230V	50Hz
México	127V	60Hz
E.E.U.U.	120V	60Hz
Japón	100V	50Hz, 60Hz
Marruecos	127V, 220V	50Hz

Cuadro 1: Distintas zonas utilizan enchufes con distintas características, a pesar de en algunos casos tener el mismo potencial eléctrico y frecuencia

Trabajaremos con el tipo de enchufe usado en España, que cuenta con un potencial de 230V y una frecuencia de 50Hz<sup>[5]</sup>.

## **2.2. ¿Qué es un enchufe inteligente?**

Un enchufe inteligente extiende esta definición de enchufe. Actuando como una interfaz que permite añadir un amplio abanico de funcionalidades relativas al manejo y supervisión de estos dispositivos externos. Dotándole de las ventajas de la conectividad.

Esto encaja con la definición de dispositivo IoT que mencionábamos previamente. Un objeto cotidiano que ha sido dotado de la capacidad de comunicación con otros dispositivos.

## **2.3. Funcionalidades comunes que ofertan los enchufes electrónicos actualmente**

### **2.3.1. Monitorización:**

Supervisar el uso de energía: Esto permite establecer un control sobre su uso, realizar estimaciones sobre el coste o incluso notificar frente a anomalías como el consumo en horarios no esperados o picos de voltaje.

### **2.3.2. Control:**

Manejar tus dispositivos de manera remota, programar el horario de funcionamiento de estos o la manera en la que funcionan son solo algunas de las opciones; destacando la posibilidad de integración con otros dispositivos que proporcionen una mayor accesibilidad (p.ej. la interfaz de voz de un asistente de móvil).

## 2.4. Tipos de enchufes

Atendiendo a sus características más distintivas podemos hacer una clasificación (no excluyente) de:

- Enchufes inteligentes (E.I.) por control remoto (WiFi, bluetooth, infrarrojos u otros tipos de señales electromagnéticas).
- E.I. programables, aquellos que permiten la programación temporal de eventos.
- E.I. de regletas, aquellos que permiten tratar con varios dispositivos simultáneamente en un mismo enchufe.

## 2.5. Modelos en el mercado

La comercialización de este producto está muy extendida, proporcionando una amplia oferta de productos los cuales podemos recoger bajo la clasificación previa:

- **Programación temporal:** Permiten la ejecución de tareas a horas concretas del día. Por ejemplo: [Garza 400603 \(analógico\)](#) (Figura 1), [Orbegozo PG 20 \(digital\)](#) (Figura 2).
- **Control remoto:** Se pueden controlar con mando a distancia, a través de una app o con control por voz. Por ejemplo: [TP-Link HS100](#) (Figura 3) o la [Regleta Xiaomi](#) (Figura 4), que es un ejemplo de intersección entre E.I. de regleta y por control remoto.



Figura 1: Enchufe programable analógico Garza 400603. Imagen de [6]



Figura 2: Enchufe programable digital Orbegozo PG20. Imagen de [7]



Figura 3: Enchufe por control remoto TP-Link HS100. Imagen de [8]



Figura 4: Enchufe regleta por control remoto Xiaomi. Imagen de [9]

### 3. Arquitecturas IoT

El paradigma de IoT viene a integrarse entre los mecanismos convencionales y el resto de redes. En sus primeros pasos se adaptaron sus diseños a las necesidades de estos sistemas. Pero según fue desarrollándose fueron naciendo algunas arquitecturas que ayudaban a definir y a trabajar mejor con los requisitos de estos nuevos sistemas.

Existen diferentes arquitecturas en la actualidad, que difieren en el enfoque que le dan a ciertos campos. Dos de los más conocidos son IoT-A e IIRA. Siendo el primero el más extendido desde su lanzamiento en 2012.[\[10\]](#)

Algunos campos de comparación entre arquitecturas de IoT son[\[11\]](#):

- El enfoque particular a casos de negocio frente a conceptos generales.
- Orientación a Internet: Si se introducen nuevos protocolos y se adaptan los ya existentes a sus necesidades.
- Tipo de dispositivo sobre el cual se orienta (como sensores o actuadores).

Estas arquitecturas coinciden en la definición de capas de manera análoga a TCP/IP u OSI[\[10\]](#), pero adaptándolas a las necesidades del IoT. Ante las diversas clasificaciones en múltiples capas se puede encontrar una similitud entre ellas que permite generalizar en 3 capas principales: Percepción, asociada a los dispositivos físicos, consistente en sensores; capa de red, encargada de transmitir información entre sensores y sistemas procesadores finales; y capa de aplicación, para el proceso final de la información[\[12\]](#).

Para conocer cómo se produce la comunicación a distintos niveles de la arquitectura es necesario analizar sus protocolos.

#### 3.1. Protocolos

Dos ordenadores convencionales necesitan una pila de protocolos para que puedan transmitir información a las distintas capas que conforman la comunicación. Ante los requisitos que surgían con los dispositivos IoT, emergían también protocolos que daban respuesta a estos. Estableciéndose así una pila de protocolos de red especializada para IoT.

A fin de permitir estas comunicaciones de manera liviana es necesario alguna tecnología a **nivel físico** que pueda solventar el problema de la saturación del espectro de frecuencias, algunas propuestas se basan en aprovechar el ruido blanco (frecuencias no usadas en el espectro de radio)[\[13\]](#). En la **capa de enlace** se encuentra

6LoWPAN[14] para conectar los dispositivos a una red IP, dicha red ha de ser del tipo IPv6 (**capa de red**) para dar soporte a la escalabilidad.

En la **capa de aplicación** existen alternativas para proporcionar un transporte de datos con baja sobrecarga en la comunicación. Algunas de estas opciones[15] son: **CoAP** (Constrained Application Protocol), usado como un protocolo web basado en *REST*[16]; **XMPP** (Extensible Messaging and Presence Protocol), usado especialmente en mensajería; **DDS** (Data Distribution Service), especializado en aplicaciones de tiempo real y con alta disponibilidad; o **MQTT** (Message Queue Telemetry Transport), para conectar dispositivos embebidos con aplicaciones y middleware.

Esta es una de las decisiones que más afecta al diseño, pues el resto del diseño de la arquitectura se ve afectado por ello.

### 3.2. Modelos de interconexión de red

De acuerdo a la topología de la red, el modo en el que se conectan los dispositivos involucrados, reconocen ocho tipos básicos de modelos de interconexión.[17]

- Punto a punto: Dos dispositivos finales se perciben directamente conectados entre sí.
- Cadena Margarita: Se conecta en serie cada computador al siguiente, los mensajes se retransmiten durante toda la cadena.
- Bus: Todos los dispositivos están conectados entre sí usando un cable central. Esto implica una difusión de cada mensaje a todos los nodos. Es más usual en redes locales.
- Estrella: Un conjunto de nodos periféricos se conectan con un nodo central que actúa como servidor.
- Anillo: Similar a una topología de “cadena margarita” donde existe un único bucle y la comunicación se realiza en un único sentido.
- Malla: Los dispositivos se conectan de muchos a muchos, puede ser completamente conectada o parcialmente conectada si todos los nodos pueden o no encontrarse conectados entre sí. Esta categoría se puede identificar con un grafo, y como tal incluye a la topología de árbol, un tipo particular de grafo.
- Híbrida: Aquellas que combinan varias topologías básicas para formar otras más complejas.

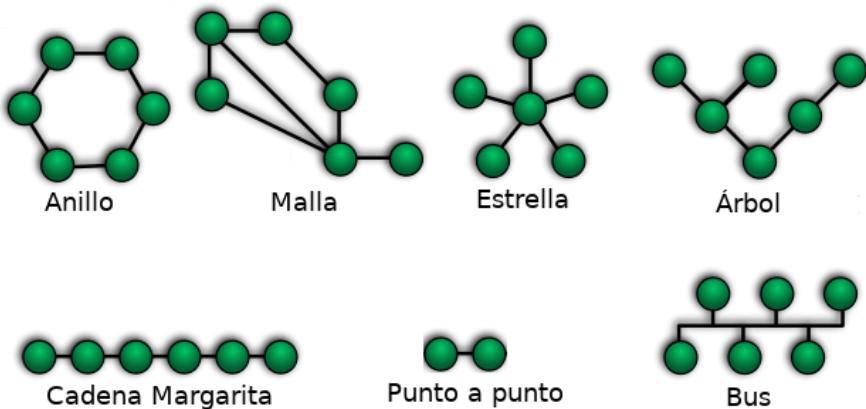


Figura 5: Visualización de algunas topologías básicas de red

El modelo de interconexión de red de un sistema no es inherente al Internet de las Cosas, es una cuestión recurrente en cualquier diseño de una red de telecomunicación y cuya decisión se realiza en tanto a las necesidades del sistema. Por ejemplo un modelo de malla consta de una mayor complejidad en las comunicaciones que una de anillo, pero este puede dificultar las comunicaciones entre distintos nodos. Una arquitectura de estrella simplifica este problema pero se vuelve dependiente de que el nodo central se encuentre siempre operativo.

## 4. Diseño de la arquitectura de nuestro sistema

### 4.1. Descripción general

Nuestro enchufe inteligente recoge mediante el sensor SCT013 la corriente que requiere el dispositivo conectado. En el módulo ESP32 se realiza el cálculo del consumo en Vatios y se emite hacia nuestro servidor, una Raspberry Pi con Mosquitto haciendo uso del protocolo de comunicación MQTT, de donde se podrá consultar desde una aplicación externa para supervisar el consumo y manipular el enchufe.

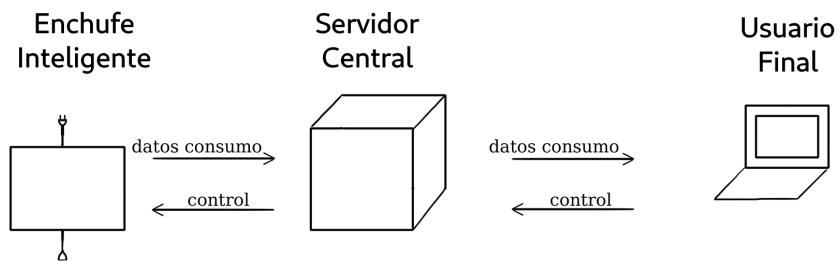


Figura 6: Representación simplificada del sistema y sus comunicaciones principales.

Como mencionábamos en la Sección 3, habitualmente las arquitecturas IoT se engloban bajo tres capas. La de percepción, nuestro sensor que recoge los datos; la capa de red, el uso de MQTT para gestionar las comunicaciones de manera liviana (sobre TCP/IP) y la capa de aplicación, que permite el procesamiento, manipulación y visualización por parte del usuario del sistema.

#### 4.2. Modelo de red usado

La topología de red usada en este proyecto es el modelo en estrella. En el centro se encuentra nuestro servidor que se encarga de gestionar tanto la información de los sensores como de otorgar la visualización y manejo sobre los enchufes a los usuarios. En los extremos se encontrarían tanto los enchufes inteligentes como los usuarios desde sus aplicaciones.

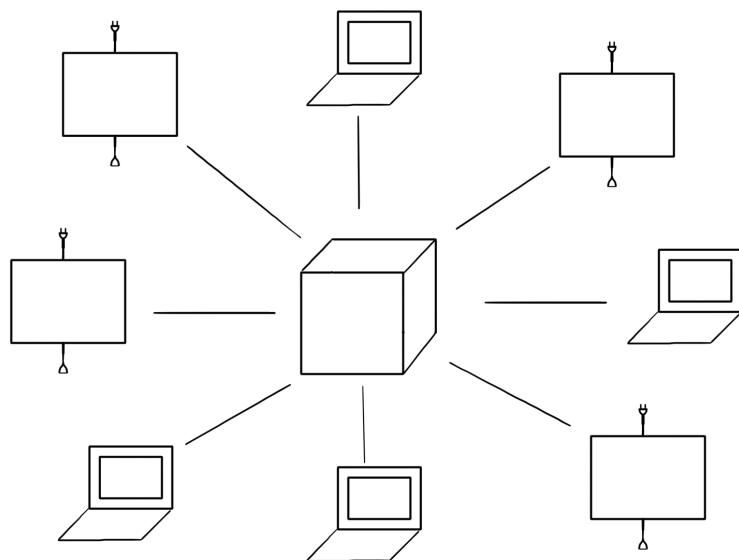


Figura 7: Topología del sistema

Se ha decidido utilizar este modelo tanto por su versatilidad como simplicidad para conectar los nodos periféricos al servidor central.

### 4.3. Protocolo MQTT

De los protocolos previamente mencionados [3.1] optamos por utilizar MQTT por su facilidad de integración con Mosquitto, un broker (agente intermediario) que implementa este protocolo y cuya ligera carga le permite ejecutarse en servidores de bajas prestaciones (Raspberry Pi, en nuestro caso).

MQTT utiliza el patrón de publicación/suscripción (publish-suscribe), un publicador escribe sobre un tema (topic) del cual un suscriptor lee. Está construido sobre TCP.

El tipo de comunicación en este protocolo es indicado en cada mensaje en la cabecera del paquete. Proporciona mensajes para conexión (CONNECT), desconexión (DISCONNECT), publicar en un tema (PUBLISH), suscribirse/desuscribirse a un tema (SUBSCRIBE/UNSUSCRIBE) y otros mensajes de control correspondientes a una respuesta ACK a los anteriormente mencionados (pues está basado en TCP).[18]

Todos los mensajes en MQTT se rigen bajo estas primitivas de control. Todos los paquetes que usan para comunicarse se denomina Paquete de control MQTT y tienen su propio formato como podemos apreciar en el Cuadro 2.

Cabecera fija: Presente en todos los paquetes de control
Cabecera variable: Presente en algunos paquetes de control
Carga: Presente en algunos paquetes de control

Cuadro 2: Estructura general de un paquete de control MQTT

En la cabecera fija se indica el tipo de mensaje de control y se reservan algunos bits como indicadores, *flags*, para indicar parámetros como el QoS (*Quality of Service*, garantías en el modo de entrega del mensaje).

En la cabecera variable se indican otros contenidos, por ejemplo un identificador de paquete cuando se establece un QoS que garantice la entrega del mensaje.

En la sección correspondiente a la carga se transporta información que complementa al mensaje de control. En un mensaje SUBSCRIBE se incluye identificador, tema a suscribirse, usuario y contraseña si se requiere, mientras que en un paquete PUBLISH en nuestro sistema la carga sirve para que los enchufes inteligentes transmitan el consumo que han calculado.

#### 4.4. Diseño broker, publishers, subscribers

Aunque existen otros subtipos de patrones publicador/suscriptor basados en tipo o en contenido[19], MQTT utiliza un modelo basado en temas.

Las principales componentes de este patrón son suscriptor, publicador y el broker. Un dispositivo interesado se registra como suscriptor de un tema y el broker se encargará de informarle cuando se publique algo en el tema. El publicador actúa como un generador de información de algún tema de interés. El broker es el intermediario que transmite a los interesados, teniendo en cuenta además comprobaciones de seguridad mediante autorización de publicadores y suscriptores.[15, 20]

Por este motivo, en este patrón pueden existir múltiples publicadores y suscriptores mientras que es la entidad que actúa como broker la que gestiona el flujo de comunicación. Un ejemplo general sobre esta arquitectura basada en temas lo podemos encontrar en la Figura 8. En ella unos sensores dotados de conectividad, como nuestros enchufes inteligentes, se encargan de publicar en unos temas una información determinada. Unos clientes interesados se suscriben a los temas que sean de su interés para su posterior uso.

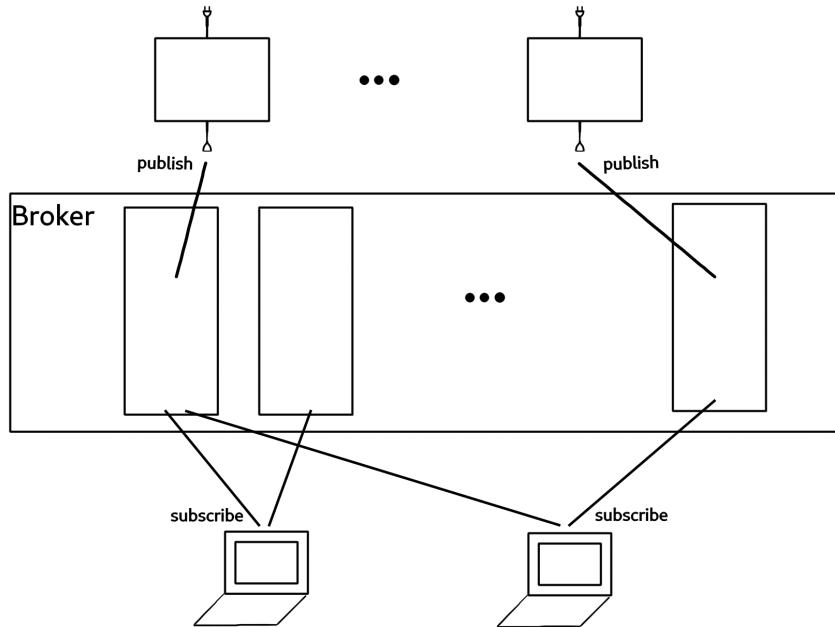


Figura 8: Ejemplo genérico del uso de un broker.

La arquitectura que hemos diseñado en nuestro sistema sigue el mismo patrón, pero se adapta al uso de diversos temas, tanto de consumo como de control. En la Figura 9 podemos ver cómo los enchufes inteligentes se encargan tanto de enviar como de recibir información del broker. Publican la información relativa al consumo de los dispositivos conectados en los temas de consumo y se suscriben a los temas

de control para que los usuarios finales puedan manipular el estado de los enchufes.

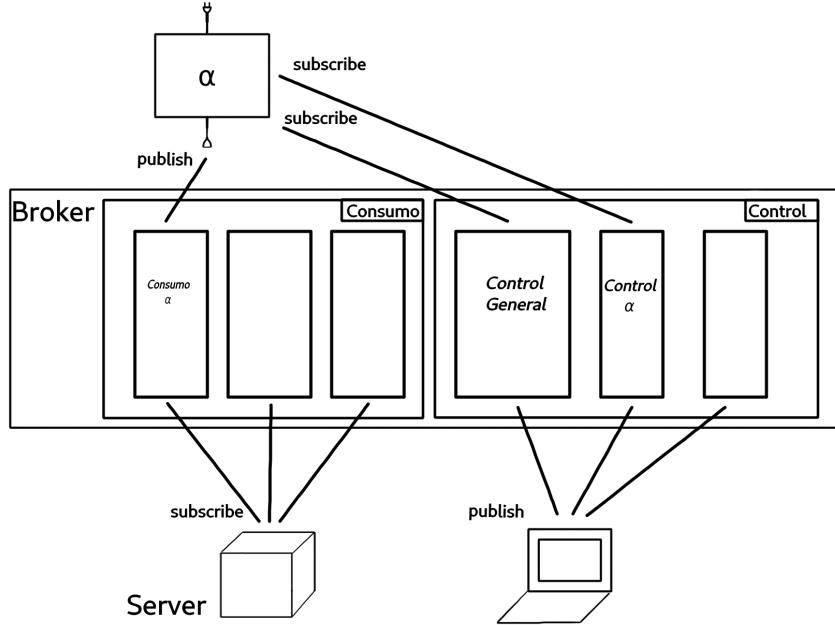


Figura 9: Comunicaciones MQTT en nuestro sistema.

Es destacable de la Figura 9 que sólo existe un suscriptor MQTT a los canales de consumo, el propio servidor central. Esto sucede para que exista una única base de datos centralizada y evitar problemas de inconsistencia en caso de que algún cliente se caiga y se encuentren bases de datos con estados distintos. La consulta de información relativa al consumo por parte del usuario final se produce vía HTTP y se explicará con mayor detallamiento en la Sección 4.6.3.

MQTT define tres niveles de QoS para adaptarse a los requisitos del sistema: 0 para una única entrega sin confirmación, 1 para al menos una entrega donde se requiere una confirmación, 2 para exactamente una entrega de mensaje utilizando una confirmación en 4 pasos (*four step handshake*). En definitiva a mayor QoS mayor fiabilidad en el envío de mensajes a expensas de una mayor latencia y requisitos de ancho de banda.

En nuestro sistema hemos optado por utilizar el QoS 0, pues la información a transmitir es tolerante a pérdidas. Se envían mensajes con el consumo con gran frecuencia y, por ser una serie temporal donde todos los datos guardan una relación con los inmediatamente contiguos, se puede realizar una estimación de valores perdidos por una interpolación sin que esto repercuta negativamente en el análisis.

Todos los mensajes se pueden configurar para ser retenidos incluso después de haber sido transmitidos a todos los suscriptores. Esto resulta especialmente útil si aparece algún suscriptor en algún tema que se actualice poco frecuentemente para que pueda acceder a la información con la que trabajar lo antes posible.

#### 4.4.1. Mosquitto

El broker que usamos en nuestro diseño es **Mosquitto**[21]. Agente que implementa MQTT v3.1.1, la versión de MQTT usada en este proyecto.

Uno de los requisitos más habituales en sistemas IoT es la integración de manera ubicua de las componentes en el entorno del usuario. A menudo esto implica utilizar sistemas compactos o embebidos con pocos recursos. En nuestro caso utilizamos un servidor con recursos limitados, una Raspberry Pi, por lo que resulta beneficioso utilizar un broker tan ligero que no suponga una sobrecarga en el sistema.

Los temas en Mosquitto se disponen siguiendo una estructura jerárquica similar a un sistema de archivos (/RUTA/AL/TEMA). En forma de árbol. Permitiendo a un suscriptor suscribirse tanto a un tema particular (nodo hoja) como a un nodo más general (cualquier nodo interno).

En su estructura también guardan metainformación relativa al propio broker: bytes enviados y recibidos, clientes conectados o mensajes pendientes entre otros.

Otra utilidad de mosquitto es la capacidad de establecer *puentes*. Múltiples brokers pueden estar conectados con esta funcionalidad. Esto es útil cuando se desea compartir información entre distintas localizaciones pero no toda la información ha de ser compartida.

Mosquitto permite numerosas opciones de configuración. Desde autenticación, cifrado y retención de mensajes hasta cuestiones de registros (*logging*), conectividad o uso los recursos del sistema.

#### 4.5. Enchufe

De los tres subsistemas que componen el diseño de nuestro sistema podríamos atribuirle el concepto de Enchufe Inteligente a este. Pues es el encargado de realizar las lecturas del consumo energético del dispositivo conectado, transmitir estas lecturas al servidor central y manejar mediante el uso de un relé la alimentación del dispositivo conectado para regular su uso.

El enchufe está compuesto por cuatro elementos principales:

- **Módulo WiFi:** Microcontrolador ESP32[22]. Núcleo del enchufe, encargado de ejecutar el programa principal que gestiona el manejo de lecturas y del relé. Dispone de funcionalidad WiFi para conectarse vía MQTT al servidor.
- **Relé:** Tongling jqc-3ff-s-z, un relé de 5V que controla el paso de corriente por el dispositivo conectado cual interruptor y que es gestionado por el módulo ESP32.
- **Sensor de corriente:** Sensor YHDC SCT013000 CT[23]. Un sensor de corriente no invasivo (basta con colocarlo alrededor del cable a medir) que actúa como un transformador de corriente. Transforma la corriente de entrada entre 0 y 100 Amperios en otra de salida entre 0 y 50 miliAmperios que, mediante el uso de una resistencia de carga que convierta a un voltaje adecuado, el módulo es capaz de procesar.

Los cables convencionales suelen estar compuestos por dos cables internos que conducen la corriente en sentido opuesto. Para evitar que se anulen las medidas el sensor se coloca en torno a uno de ellos. Además, es posible medir el sentido de la corriente para determinar si se está consumiendo o generando electricidad.

- **Fuente de alimentación:** Una fuente de alimentación de 5 Voltios y 2 Amperios para alimentar el módulo WiFi. Un conversor que toma la corriente alterna que llega al enchufe y alimenta al resto del circuito.

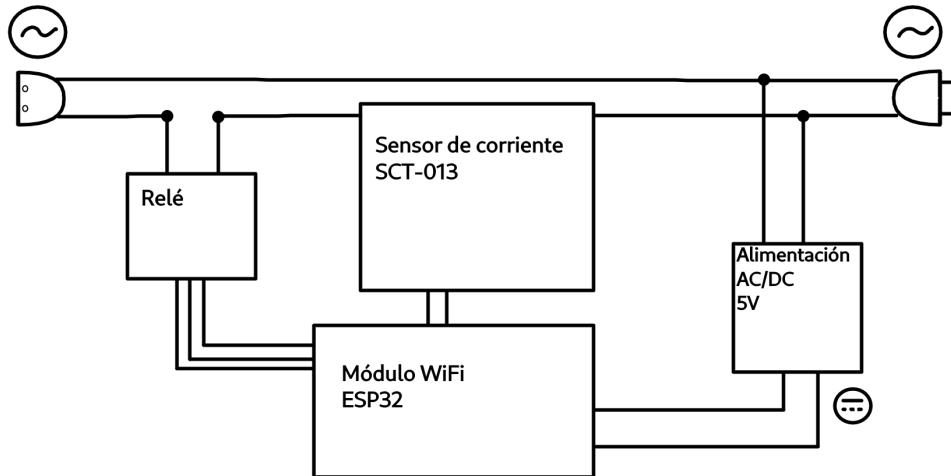


Figura 10: Esquema del Enchufe Inteligente.

#### 4.5.1. Diseño y construcción del enchufe

En un proceso de prueba y selección de componentes descartamos otras opciones en el diseño de nuestro enchufe:

El microcontrolador Arduino UNO carecía de funcionalidad WiFi y el uso de módulos externos requería alimentación extra. El ESP32 usado sí que disponía de WiFi y era compatible con el resto de componentes, reduciendo así la complejidad de este subsistema mientras otorgaba la misma funcionalidad.

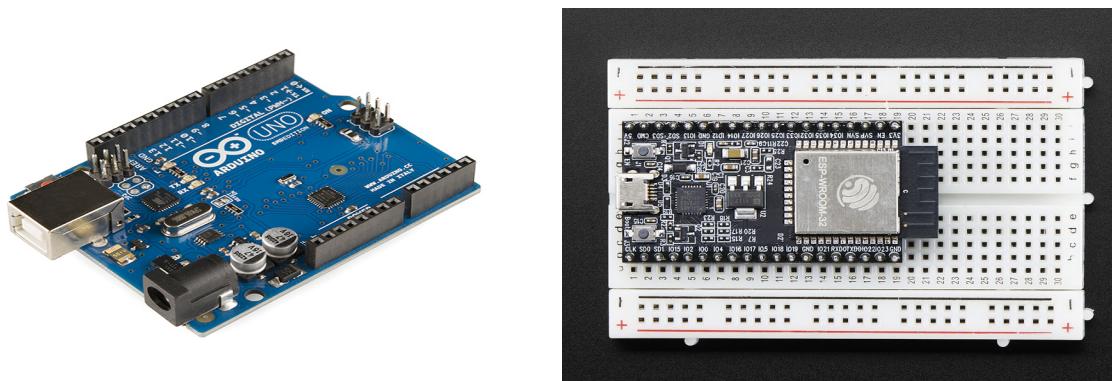


Figura 11: Arduino UNO, a la izquierda, imagen de [24]. A su derecha módulo ESP32, imagen de [25].

El sensor de corriente no invasivo fue seleccionado tras descartar el sensor ACS712[26]. A pesar de realizar varias pruebas con varios sensores para descartar que fuera defec-tuoso descartamos el uso de esta componente. Este introducía ruido en las lecturas, mostrando diferencias entre el consumo esperado y el registrado para componentes eléctricas que conocíramos de antemano (por ejemplo una bombilla).

En su lugar el sensor SCT013 fue más consistente en las lecturas y su diseño no invasivo simplifica notoriamente las labores de mantenimiento en caso de avería y reduce el riesgo en la instalación al exponer menos componentes.



Figura 12: Sensor de corriente no invasivo.

Tras esta selección procedimos a probar y, posteriormente, soldar el enchufe a una placa base. A fin de minimizar el número de cruces entre cables elaboramos el diseño visible en la Figura 13. Este representa la vista superior de la placa y cómo se disponen los cables y las componentes tanto por encima como por debajo de la misma. Es conveniente indicar que cada rectángulo representa un conjunto de pines que irán conectados a alguna componente del sistema: El rectángulo GND, 5V a la fuente de alimentación, el par In, Out la entrada y salida del sensor de corriente y la tres-upla Vcc, Gnd y S a la alimentación y control del relé.

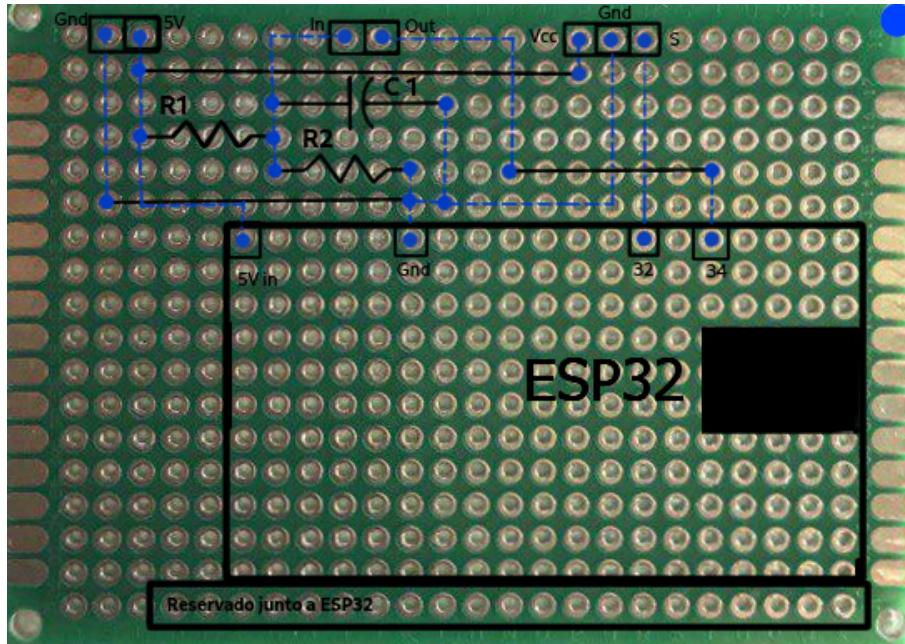


Figura 13: Vista superior de la placa, en negro componentes y cableado de la parte anterior. En azul, de la parte posterior.

Hacemos uso del condensador  $C1 = 10\mu F$  para absorber los posibles picos de tensión, reducir el ruido en las lecturas y garantizar calidad en las mismas.

Las resistencias  $R1 = R2 = 2k\Omega$  cumplen la función de un divisor de tensión y de resistencia de carga para garantizar el voltaje operativo del sensor de corriente y otorgar el rango de voltaje de salida esperado.

Para complementar esta visualización añadimos la parte posterior (trasera) de la vista trasera de la placa (volteo vertical), Figura 14, y la parte posterior de la vista superior, Figura 15.

Los adaptadores de los pines sobre los que se colocarán las otras componentes queda dispuestos de acuerdo a la Figura 16. Hacemos uso de estos adaptadores a fin de simplificar las tareas de mantenimiento y sustitución de componentes en caso de avería.

El circuito resultante es el que podemos observar en las figuras 17 y 18.

Tanto la alimentación como la medición de la corriente se produce a partir de un cable que hemos seccionado y empalmado a las componentes de nuestro circuito.

Para abstraer el diseño del enchufe y simplificar el uso del mismo al usuario lo hemos acoplado en una caja de conexión. De este modo el usuario final puede limitarse a, simplemente, conectar el dispositivo a medir a nuestro enchufe inteligente y este a una fuente de alimentación. Una conexión macho y una hembra.

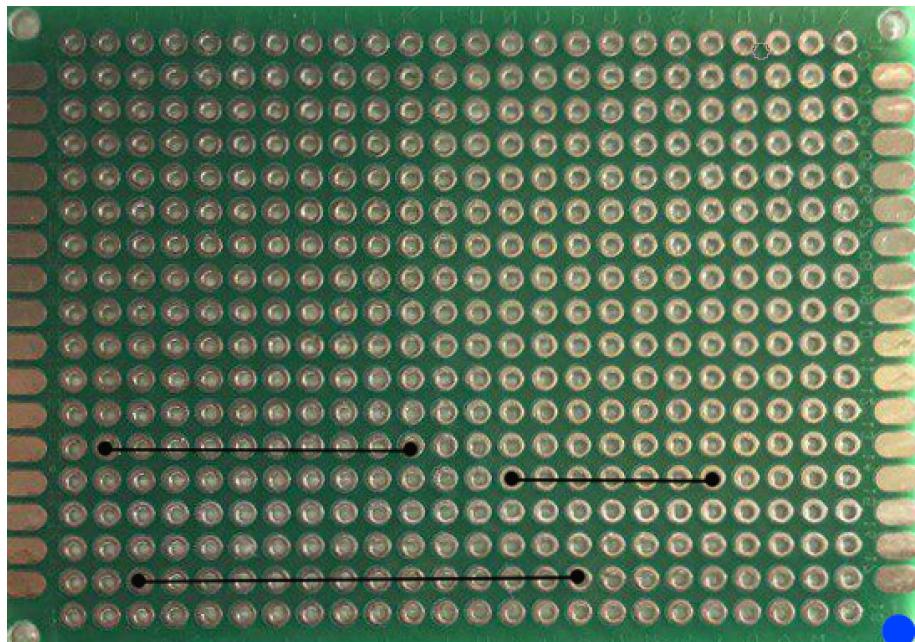


Figura 14: Vista trasera de la placa, parte posterior.

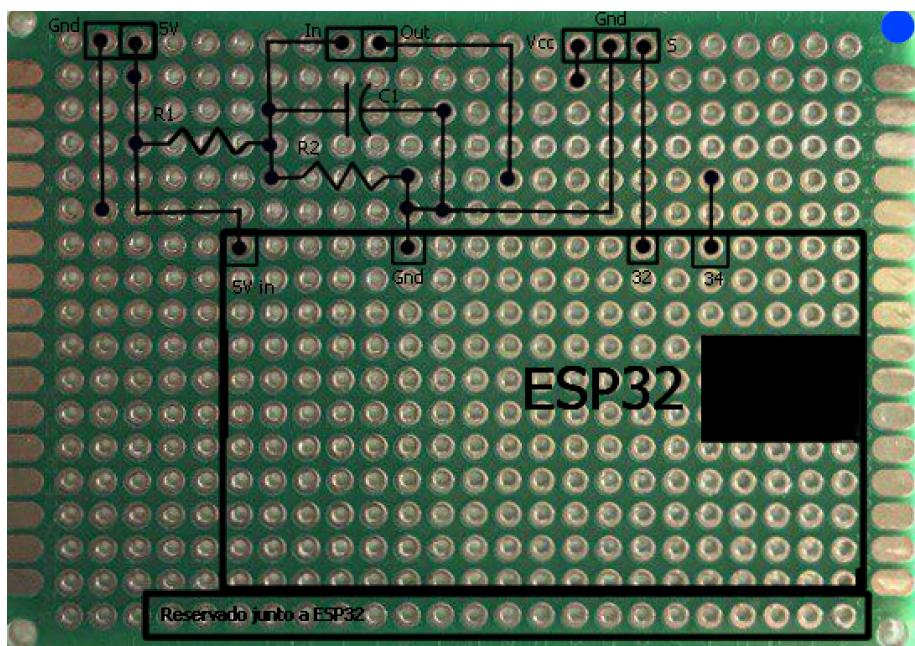


Figura 15: Vista trasera de la placa, parte posterior.

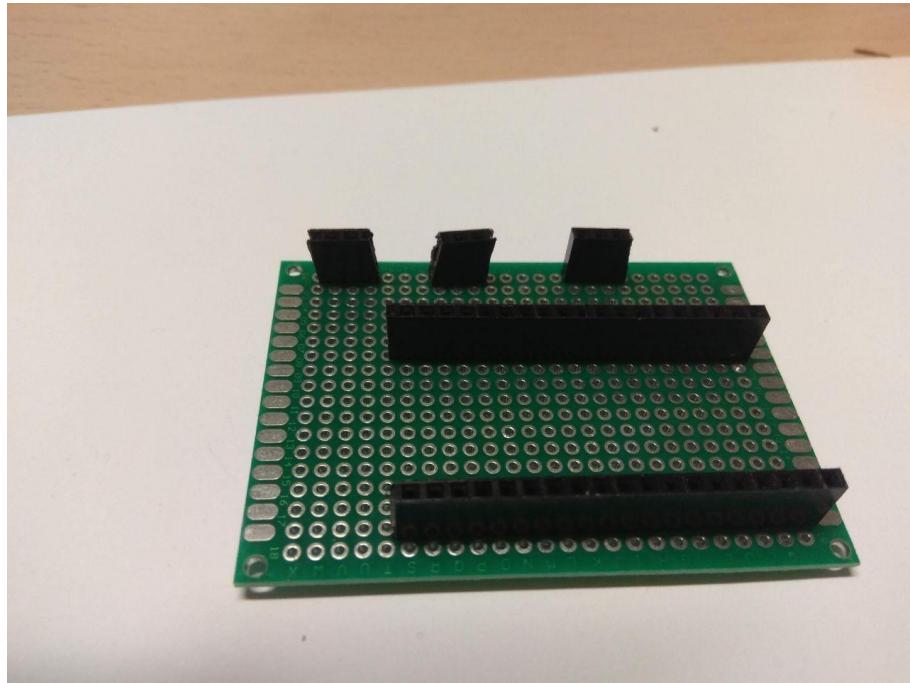


Figura 16: Adaptadores de las componentes.

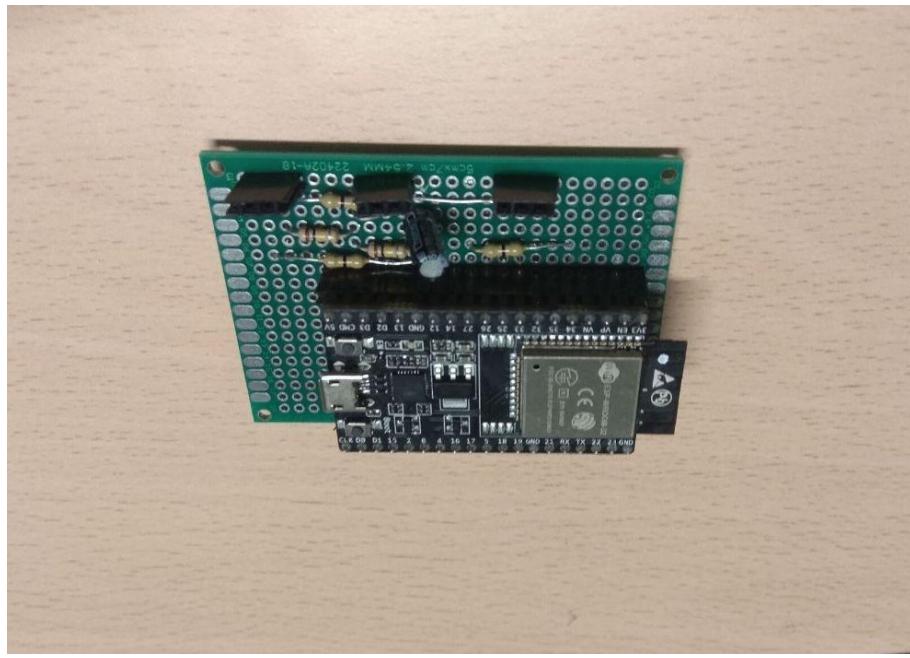


Figura 17: Soldadura parte superior placa.

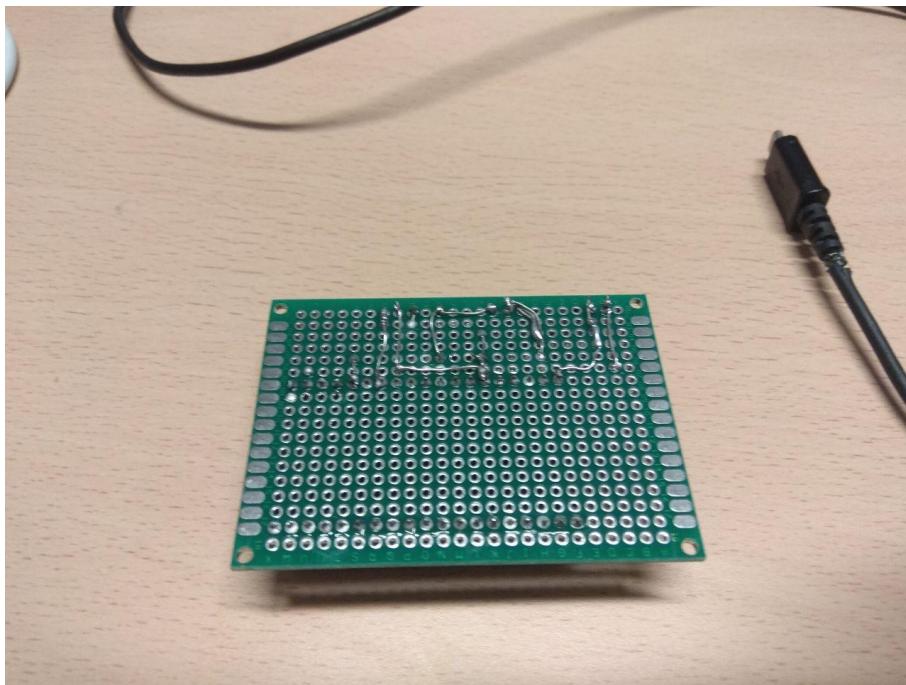


Figura 18: Soldadura parte inferior placa.

#### 4.5.2. Programa del microcontrolador

Toda la lógica del enchufe, tanto lecturas como comunicaciones, son gestionadas por el programa del microcontrolador. Y resulta conveniente comprender qué elementos hacen posible esta mecánica.

Por un lado se encuentra la configuración de la comunicación. Es necesario conectarse a la red sobre la que se trabajará para poder comunicarse con el broker, así que se requiere conocer SSID y contraseña de la red en la que se trabajará. Del mismo modo la información del broker, temas MQTT de suscripción o publicación y credenciales tendrán que ser conocidos. Dada la licencia permisiva de este proyecto y la flexibilidad de programación del módulo ESP32 no habrá problema en entregar el sistema al usuario final en el estado de uso inmediato o de permitir que este lo adapte a sus necesidades.

El enchufe publica el cálculo del consumo en un tema asociado a su identificador en el broker y se suscribe a los canales de control para gestionar su funcionamiento (actualmente manejo del relé, pero puede extenderse a configuración de otros parámetros).

A fin de procesar la información del consumo cada enchufe está identificado únicamente por un identificador que puede ser adaptado *ad-hoc* (se podrían utilizar otros identificadores como el del propio microcontrolador, pero se deja a opción del usuario para permitir flexibilidad en la nomenclatura).

En el apartado de las lecturas es destacable cómo se han realizado los cálculos.

En términos de electromagnetismo se define la potencia  $W$  (la variable que utilizamos para medir el consumo) como:

$$W = V \cdot A$$

Siendo  $V$  el voltaje del circuito, 230 voltios en España, y  $A$  la intensidad de la corriente que mide el sensor.

Con las mediciones del sensor de corriente calculamos la media cuadrática. La corriente alterna fluctúa en torno a valores positivos y negativos siguiendo un comportamiento sinusoidal. La manera de calcular la intensidad que realmente está pasando es obteniendo el valor eficaz. Este valor eficaz para la intensidad se corresponde con el valor cuadrático medio[27].

Para reducir hipotéticos picos de ruido se realiza un muestreo un intervalo de tiempo fijo y se calcula el consumo durante ese período de tiempo  $W$  tomando como valor de la corriente  $A$  la media de las medidas en el muestreo.

## 4.6. Servidor central

En el centro de nuestra arquitectura de estrella se encuentra el servidor central. Una Raspberry Pi 3 Modelo B que actúa como broker entre los publicadores y suscriptores de MQTT, como mencionábamos en 4.4.1, haciendo uso de Mosquitto para gestionar las publicaciones y suscripciones a los temas.

La configuración de nuestro broker no permite la conexión anónima, es decir, toda conexión requerirá usuario y contraseña. Permitirá la persistencia de los mensajes que publiquen en sus temas y almacenará otros tantos de depuración.

Es necesario que el broker esté activo y sea persistente a reinicios y caídas. Por esto hay que configurarlo para que Mosquitto se lance en cada inicio del servidor.

Este problema se puede abordar en un sistema Linux (y en particular en Raspbian, el Sistema Operativo que estamos usando) de diversas maneras. Una opción es usar el archivo `/etc/rc.local` para ejecutar una orden en un script. Esta opción es similar al uso de un *cronjob* definido en cada inicio. También existe la posibilidad de definir un servicio y activarlo al inicio con *systemd* (*daemons*, procesos en segundo plano, del sistema que se ejecutan al inicio). Cualquiera de estos métodos resulta válido para esta finalidad.

Este servidor supone el nodo central de nuestra topología de red, es por tanto el más determinante para el correcto funcionamiento del sistema. Si quisieramos mejorar la disponibilidad del sistema para poder seguir utilizándolo aunque este servidor esté caído puede ser una mejora utilizar un segundo servidor similar a este al cual el resto de nodos tomaran por broker en caso de que el primero estuviera inaccesible.

### 4.6.1. Configuración de red

Para favorecer la localización del servidor central en la red es conveniente asignarle bien una IP estática (configurar alguna interfaz o modificar la configuración del cliente dhcp en Raspberry Pi) o bien utilizar algún DNS y que se actualice la dirección de la misma.

En este caso hemos asignado una IP estática al servidor dentro de nuestra red y hemos solicitado un nombre de dominio que actualiza la IP pública del router de acceso al servidor, configurando además un reenvío de puertos para que un cliente pueda conectarse desde el exterior.

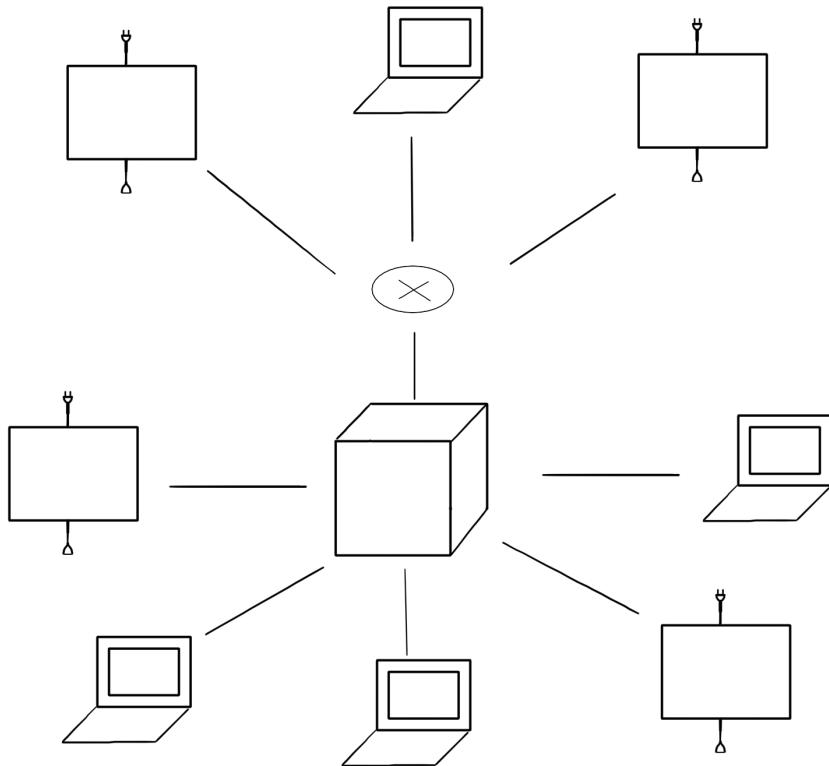


Figura 19: Esquema de la configuración de red y conexión a internet.

Como hemos indicado previamente los nodos periféricos se conectarán con el servidor central. Pero resulta interesante estudiar cómo se producen estas conexiones de acuerdo a la configuración de red.

En la Figura 19 podemos ver un esquema de cómo se realizan las conexiones. Aquellas que están directamente conectadas con el servidor central representan elementos en la red local, cuya conexión no desentraña ninguna complejidad. Sin embargo, aquellas que se encuentran conectadas a un nodo intermedio representan las conexiones provenientes desde redes externas. Desde internet. Y se pueden comunicar con el servidor central mediante el router (nodo intermedio) que debe tener la configuración oportuna de reenvío de puertos.

#### 4.6.2. Gestión de la base de datos

En un primer diseño ubicamos la base de datos en los propios usuarios finales, de modo que libraríamos al servidor central de mayor carga de gestión y limitaríamos las comunicaciones en la red a publicaciones y suscripciones de MQTT. Esta idea fue descartada finalmente pues resulta más valioso tener una base de datos sin

inconsistencias entre los distintos almacenamientos de los clientes que un servidor menos congestionado.

En el diseño final, en el servidor central se encuentra situada una base de datos donde se recogen las lecturas de los enchufes. De este modo se puede gestionar la difusión de esta información a los clientes sin comprometer la consistencia de los datos.

La estructura de esta base de datos consiste en una tabla que incorpora para cada medición una fila con los datos del enchufe en el que se ha leído, el consumo y la hora registrada según el servidor. La creación de esta base de datos se gestiona con un programa que crea una base de datos con este esquema si no existía previamente. El Sistema Gestor de Bases de Datos usado ha sido sqlite[28] por su capacidad para implementar una base de datos relacional SQL sencilla y rápida.

La manera en la que se pobla esta base de datos es mediante un programa que se ejecuta de manera concurrente al broker y que se comporta como un suscriptor más al tema de las publicaciones de consumo. Pero su procesamiento del mensaje implica su almacenamiento en la base de datos.

#### 4.6.3. Consulta a la base de datos

Una vez que la información del consumo energético de nuestros enchufes se encuentra almacenada en una base de datos en nuestro servidor central es necesario implementar algún medio de comunicación entre el servidor y el usuario final para que puedan acceder a esta información.

La solución que le hemos dado a este requisito es la creación de una API simple, un microservicio. Una interfaz que sirva a un usuario para consultar la información relativa a un enchufe. Todas las peticiones que se pueden hacer a esta API son exclusivamente de consulta (peticiones HTTP tipo GET), por lo que no existe el riesgo de que múltiples escritores generen problemas derivados de un acceso concurrente.

Esta API está optimizada para realizar las consultas a la base de datos de modo que se adecúen a los procesos en otros módulos para evitar un postprocesado innecesario en etapas posteriores. Ofreciendo los datos relativos a uno o varios enchufes y a un rango determinado de lecturas en un formato específico que sea más directamente interpretable por las aplicaciones que se comuniquen con la API.

#### 4.7. Aplicación de escritorio

El último eslabón de nuestro sistema es el cliente de escritorio. Una aplicación que permite al usuario final interactuar con el resto del sistema.

Este cliente es en realidad una interfaz que otorga control sobre las funcionalidades del sistema: Por un lado, acceder a la información sobre el consumo eléctrico y visualizarla y, por otro lado, manipular los enchufes inteligentes para controlar su estado.

En la topología de nuestro diseño este tipo de nodo forma parte de los nodos periféricos que se conectan al servidor central. Pero a diferencia de los otros nodos periféricos, los enchufes inteligentes, estos permiten una mayor flexibilidad y libertad de configuración. Por esto se le permite al cliente modificar desde la aplicación los parámetros de conexión.

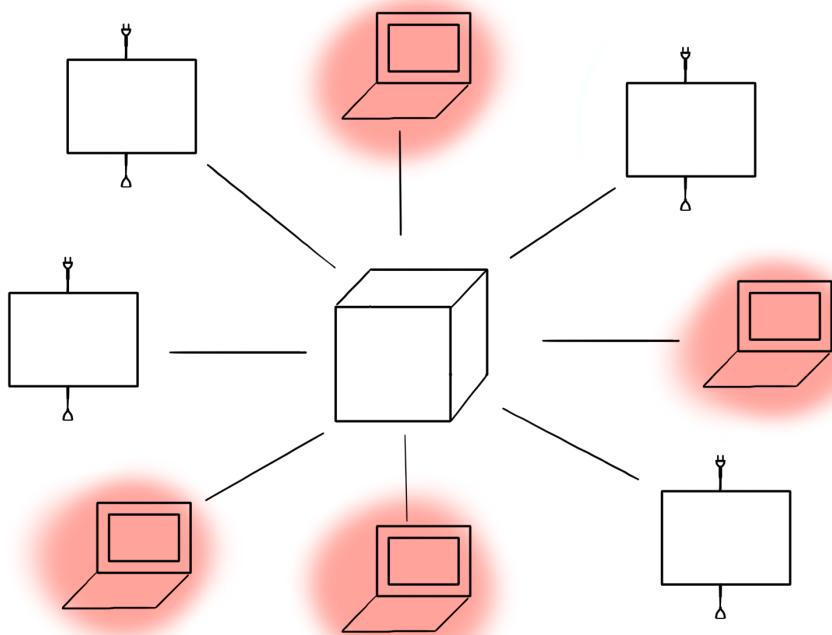


Figura 20: Topología del sistema, en rojo los clientes de escritorio.

Una vez conectado al servidor la aplicación puede obtener la información del mismo. Consultar su base de datos para obtener el consumo realizado y poder visualizarlo de manera sencilla o consultar los registros en mayor profundidad.

Al obtener esta información también estamos consultando qué enchufes se encuentran conectados y qué identificadores tienen. Lo cual nos permitirá controlar el estado de los enchufes (modificando el estado de su relé) publicando en el broker en los temas de control a los que los enchufes estarán suscritos.

A continuación se adjuntan algunas imágenes de la interfaz junto a una descripción.

ción de su utilidad:



Figura 21: Ventana principal.

Esta interfaz permite:

1. Conocer el estado de conexión actual con el broker
2. Acceder a la configuración de comunicación con el broker
3. Acceder a una lista con los enchufes accesibles

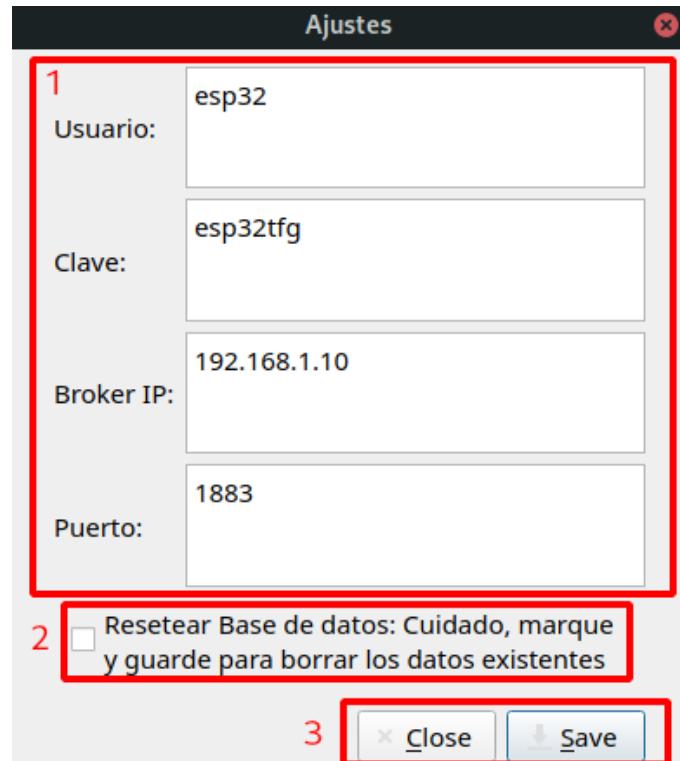


Figura 22: Ventana de ajustes.

En la ventana de configuración se permite:

1. Modificar parámetros de configuración de red
2. Restablecer configuración por defecto
3. Guardar o cancelar los cambios realizados



Figura 23: Lista de enchufes accesibles.

En esta ventana se muestra una lista de enchufes que se actualizan dinámicamente según se van conectando más enchufes al sistema. Permite acceder al control de cada enchufe pero también ofrece una opción para analizar y controlar simultáneamente todos los enchufes conectados al sistema.

La visualización de cada enchufe es la siguiente:



Figura 24: Ventana de un enchufe.

En esta vista se permite visualizar de manera rápida el consumo para cada enchufe (o para todos si se accede desde la opción general). Además se ofrece información exacta sobre el estado actual mostrando la última lectura, se permite el control individual con el interruptor de la parte superior derecha y se permite acceder a un registro en detalle del consumo de este enchufe con el botón *Ver más* en la parte inferior derecha.

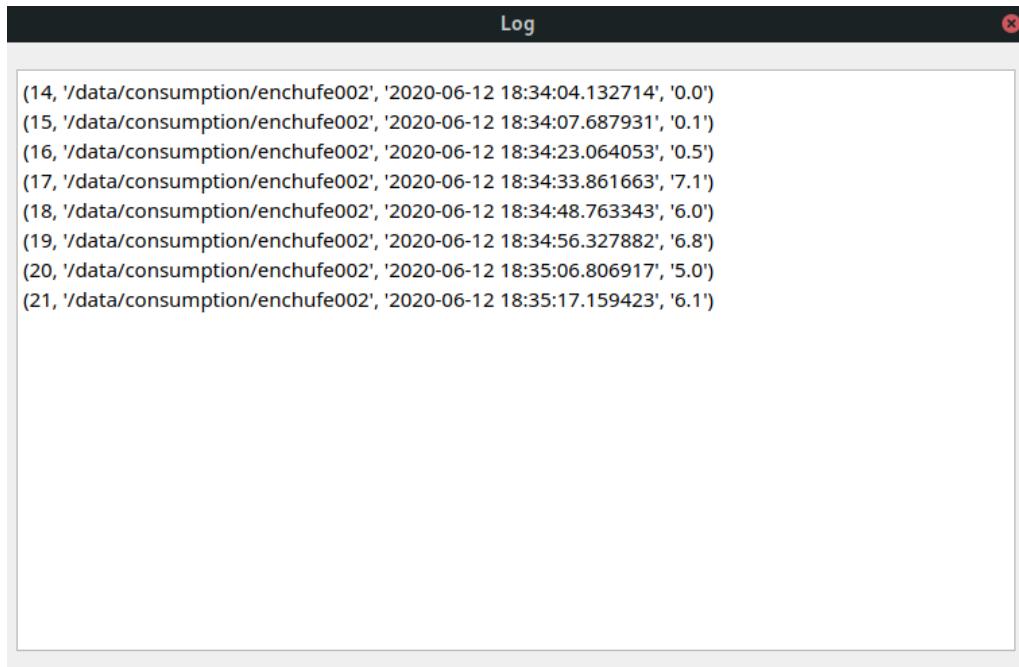


Figura 25: Log en detalle del consumo de un enchufe.

## 5. Pruebas y test

En el desarrollo de cualquier software es determinante asegurarse de que el código funcionará siempre en el contexto del sistema a pesar de que se introduzcan cambios en el propio código. La salida debe ser la esperada a pesar de avanzar en el desarrollo del producto.

Nuestro sistema, igualmente, necesita que el desarrollo sea pautado por unas pruebas o *tests* que nos garanticen que lo que estamos haciendo efectivamente es lo que deseamos obtener.

La particularidad a la hora de realizar *tests* en un sistema tan complejo reside en que cada elemento del sistema debe individualmente asegurarse de su correcto funcionamiento y debe existir algún tipo de prueba que compruebe que los componentes se están integrando y comunicando debidamente.

### 5.1. Pruebas en el módulo WiFi

En primer lugar analizaremos el proceso de prueba para el módulo WiFi. En realidad este subsistema al ser en esencia un microcontrolador se puede analizar como un sistema embebido como también pasa con Arduino.

Existen varias alternativas para realizar tests unitarios para Arduino o ESP32 [29, 30]. Algunas orientadas a la ejecución en la propia placa y la comunicación por puertos seriales y otras orientadas a la simulación por software de la placa. En cualquier caso herramientas que se encuentran en desarrollo y que distan de ser una solución consolidada.

Ambos métodos tienen sus carencias, ejecutar en una placa conlleva ocupar el uso de una placa en todo momento del desarrollo y puede ralentizar el avance por los tiempos de carga del programa e interpretación de los resultados obtenidos por serial. El proceso de simulación, si bien consigue ser más rápido, pues un procesador de un ordenador personal resulta de órdenes de magnitud más rápido que un microcontrolador, tiene dos inconvenientes principales: Por un lado la complejidad de la simulación, debes adaptar las clases y estructuras de datos a las que necesitaría tu programa (requisitos para simular seriales, por ejemplo) y, por otro lado, careces del acceso real a sensores u otros elementos que puedan estar influyendo en la salida de tu sistema.

Sea cual sea el método utilizado es necesario tener en cuenta que un sistema embebido puede fallar por múltiples razones y no nos va a garantizar el correcto funcionamiento. El objetivo será garantizar que el comportamiento sea correcto o, al menos, consistente en los errores (esperado).

En nuestro caso hemos hecho probado que el microcontrolador funcione adecuadamente simulando los valores obtenidos por los sensores. Nuestro objetivo es garantizar el correcto funcionamiento del programa del microcontrolador, asumiendo que los sensores funcionarán bien.

## 5.2. Pruebas en el servidor central

En el servidor central se sitúan dos elementos principales: El broker de Mosquitto, que es un software externo pero que podríamos usar para probar efectiva integración de este con enchufes y clientes finales; y la API que envuelve la funcionalidad de consulta de la base de datos, que debe ser debidamente probado.

Para comprobar el debido funcionamiento de los módulos en esta parte hemos realizado un conjunto de pruebas mediante una herramienta genérica de tests unitarios (*unittest* y *pytest* en Python).

Algunos de estos tests en el servidor han sido comprobar la creación de la base de datos o el comportamiento ante los errores de conexión del módulo al broker.

Estos nos han servido para diseñar mejor la conducta de este sistema. Por ejemplo se encontró el problema de que el programa no podía continuar su ejecución si el broker no estaba disponible. Captando la excepción oportuna en uno de los tests elaborar un comportamiento que mejorara la estabilidad del servidor.

```
test_create_database
.test_connection
.
-----
Ran 2 tests in 5.038s
OK
```

Figura 26: Salida de la ejecución los tests creados en el servidor

### 5.3. Pruebas en la aplicación de escritorio

Nuestra aplicación de escritorio también consta de dos elementos que resultan de interés: La interfaz de usuario y los módulos que dan soporte a la lógica de la aplicación.

Realizar tests sobre una interfaz de usuario puede resultar ser una tarea menos habitual en nuestro ámbito de trabajo usual, de hecho no suele ser suficiente utilizar una herramienta genérica. Por suerte existen herramientas específicas para comprobar el correcto funcionamiento de la interfaz de usuario. Este es el caso de la biblioteca que hemos utilizado, que dispone de una herramienta realizar test específicos tales como creación de elementos, contenido de campos o simulación de acciones (*QTest* para *PySide2*, en Python).

```
.test_main_window_creation
.test_plug_list_window
.test_plug_window
.test_check_connection
.test_get_from_db
.test_get_settings
.test_load_scene
.
-----
Ran 8 tests in 21.578s
OK
```

Figura 27: Salida de la ejecución los tests creados para la app

Para esta parte hemos tenido que comprobar múltiples apartados: La creación de la interfaz, su correcto funcionamiento y la funcionalidad de los módulos que se usan para dar soporte al subsistema de la aplicación de escritorio.

## 6. Conclusión

Ya sabíamos que el Internet de las Cosas traía consigo innumerables posibilidades para extender la concepción habitual de los objetos que nos rodean en nuestro día a día.

Con este proyecto se pretende hacer ver que el Internet de las Cosas no es un concepto distante y que no existe una barrera económica o tecnológica que solo permita a grandes empresas trabajar con él. El Internet de las Cosas es accesible para cualquiera tanto para su uso como para su desarrollo. Apto para cualquiera sin requerir grandes gastos económicos y con unas opciones de diseño y desarrollo que existen y que cada vez están mejor documentadas.

Entre los retos que se han planteado en este trabajo se encuentran trabajar con fundamentos de la física, comprender las utilidades de muchas componentes electrónicas que hemos acabado usando en nuestro diseño y que no son la materia de estudio usual que se puede encontrar un Ingeniero Informático en su trabajo pero que, sin embargo, debemos ser capaces de trabajar con ellos si la ocasión lo requiere. Lo cual nos enseña que la informática, y en particular en el ámbito del IoT, es un área que intersecta con muchas otras ramas del conocimiento como la física, la bioinformática o la medicina entre muchas otras. Y esto implica que debemos ser capaces de adaptar nuestros conocimientos para extenderlos a cualquier problema que se nos presente.

Otro de los retos que se han presentado y que es destacable es, en sí mismo, el diseño de un sistema tan grande. En muchas ocasiones nos especializamos en aprender a trabajar en campos muy concretos de la informática, inteligencia artificial, desarrollo de software, sistemas de información, etc. Pero para hacer proyectos reales por cuenta propia es necesario abarcar áreas que se escapan de la especialidad de cada uno. Ese ha sido uno de los motivos que me impulsó a hacer este trabajo, enfrentarme a una placa base sobre la que diseñar un circuito y soldar un microcontrolador, planificar unas bases de datos, gestionar la comunicación y elaborar la interfaz de un usuario final. En resumen, llevar un diseño de principio a fin, contemplando todas las posibilidades de diseño a la par que se investiga sobre aquellas que se desconocen.

Por este último motivo han surgido desafíos derivados. La importancia de establecer un desarrollo basado y dirigido por tests es un tema que en nuestra formación se ha procurado tener presente. Está claro que si se quiere desarrollar un software de calidad y escalable es vital establecer este paradigma de desarrollo. Sin embargo el ámbito de este proyecto ha sido amplio y ha resultado una dificultad procurar elaborar un conjunto de pruebas o tests que verifiquen el correcto desarrollo del proyecto y la calidad del mismo. Esto me ha permitido conocer más de cerca cómo se realiza este desarrollo en otros ámbitos menos usuales y, aunque se pueda mejorar

la calidad del mismo, establecer una buena aproximación.

Considero este proyecto como una oportunidad de aprender y de desarrollo personal como Ingeniero Informático, pero también espero que sirva de utilidad a quien quiera monitorizar el consumo en su hogar pero no enfrentarse a todo el proceso de diseño y sobre todo de inspiración a quien quiera desarrollar un proyecto *maker*, autónomo y al alcance de su mano, para demostrar que simplemente con la voluntad por aprender y un trabajo de investigación se puede llegar a conseguir lo que se proponga.

## A. Anexo: Código del proyecto

Todo el código del proyecto puede ser consultado en este repositorio de Github:

<https://github.com/jojelupipa/smart-plug>

El repositorio se ha estructurado de manera que se pueda consultar fácilmente cada sección de este trabajo (accesible a continuación mediante hiperenlace):

- **Código fuente:**

- Enchufe inteligente
- Servidor central
  - Dependencias (Python) del servidor
  - Provisionamiento del servidor
  - Configuración Mosquitto
- Aplicación de escritorio
  - Dependencias de la app

- **Documentación:**

- Memoria del proyecto
- Bibliografía del proyecto

## B. Anexo: Cómo usar este proyecto

En el [repositorio del proyecto](#) se recogen instrucciones de uso de cada una de las partes del sistema. En esta sección se busca proporcionar unas pautas para que el usuario pueda sacarle el máximo partido al sistema.

Dada la modularidad de este proyecto puede usarse en su conjunto o simplemente las partes que se necesiten, modificando el resto o sustituyéndolas por otras partes que resulten más convenientes.

Además, dada la licencia permisiva de este proyecto, se deja la libertad al usuario de modificar el diseño del sistema para que se adecue a sus necesidades.

Para este fin se adjuntan algunas indicaciones con los aspectos más relevantes a tener en cuenta si se desea integrar el sistema con otros elementos.

### B.1. Enchufe Inteligente

Usar el enchufe diseñado en este proyecto es, simplemente, una simplificación del trabajo. Es posible que no se dispongan de los materiales necesarios para construir este mismo enchufe, que se tuviera previamente otro enchufe o que se prefiera usar algún otro tipo de comunicador.

Cualquier sensor, enchufe o interfaz que actúe como transmisor debe hacer uso de un cliente MQTT que publique en el tema (*topic*) indicado.

`/data/consumption/*nombre_enchufe*`

*Sustitúyase \*nombre\_enchufe\* por el identificador de la interfaz a conectar*

Del mismo modo, si se desea dotar de una funcionalidad de control sobre el enchufe este deberá contener algún tipo de relé o interruptor que se controle mediante una suscripción al tema de control:

`/control/toggle/*nombre_enchufe*`

### B.2. Servidor central

Tampoco es necesario utilizar Mosquitto, mientras las credenciales utilizadas por los nodos finales (enchufes y clientes finales) sean las apropiadas para la comunicación el broker utilizado no será un problema para el correcto flujo de la información.

### B.3. Cliente de escritorio

El cliente de escritorio puede ser adaptado también a las necesidades del usuario, la obtención de información se hace mediante un conjunto de consultas a la API del servidor central. Por lo que todas las vistas pueden ser modificadas como se deseé.

La manipulación de los enchufes se realiza mediante publicaciones MQTT al tema correspondiente en el broker:

```
/control/toggle/*nombre_enchufe*
```

## Referencias

- [1] IoT Agenda (website). What is internet of things (IoT)?  
<https://web.archive.org/web/20191107082047/>  
<https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT>.
- [2] S. Vongsingthong, S.; Smachan. IoT examples: Suranaree Journal of Science and Technology. p5-17.
- [3] IEC. IEC - World Plugs: List view by potential.  
[http://www.iec.ch/worldplugs/list\\_byelectricpotential.htm](http://www.iec.ch/worldplugs/list_byelectricpotential.htm).
- [4] Nuevatribuna. El origen de las frecuencias eléctricas ¿Por qué 50 y 60 Hz?  
<https://www.nuevatribuna.es/articulo/ciencia/origen-frecuencias-electricas-50-y-60-hz/20150811110350118983.html>.
- [5] IEC - World Plugs: List view by location.  
[https://www.iec.ch/worldplugs/list\\_bylocation.htm](https://www.iec.ch/worldplugs/list_bylocation.htm).
- [6] Amazon (website). Garza 400603 Temporizador analógico mini, Blanco - Azul: Amazon.es: Bricolaje y herramientas.  
<https://web.archive.org/web/2019111121153/><https://www.amazon.es/Garza-Power-Temporizador-anal%C3%A9gico-programaci%C3%B3n/dp/B00URUVDW2/>.
- [7] Amazon (website). Orbegozo PG 20 - Programador eléctrico digital con pantalla LCD, temporizador de funcionamiento programable: Amazon.es: Hogar.  
<https://web.archive.org/web/20191112121450/><https://www.amazon.es/dp/B00ZJ1LQDK>, November 2019.
- [8] Amazon (website). TP-Link HS100 - Enchufe inteligente para controlar sus dispositivos desde cualquier lugar, sin necesidad de concentrador, funciona con Amazon Alexa y Google Home e IFTTT: Tp-Link: Amazon.es: Informática.  
<https://web.archive.org/web/20191112130933/><https://www.amazon.es/dp/B06W586CDZ>.
- [9] Amazon (website). Xiaomi, 18082, Regleta, Blanco: Xiaomi: Amazon.es: Electrónica.  
<https://web.archive.org/web/20191116174434/><https://www.amazon.es/dp/B07DJ2G1CW>, November 2019.
- [10] Michael Weyrich and Christof Ebert. Reference Architectures for the Internet of Things. *IEEE Software*, 33:112–116, January 2016.
- [11] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, October 2010.

- [12] Rafiullah Khan, Sarmad Ullah Khan, Rifaqat Zaheer, and Shahid Khan. Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges. In *2012 10th International Conference on Frontiers of Information Technology*, pages 257–260, Islamabad, Pakistan, December 2012. IEEE.
- [13] James Temperton. TV white space will connect the internet of things. *Wired UK*, February 2015.
- [14] Christian Peter Pii Schumacher, Nandakishore Kushalnagar, and Gabriel Montenegro. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. <https://tools.ietf.org/html/rfc4919>.
- [15] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, Fourthquarter 2015.
- [16] Web Services Architecture.  
<https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest>.
- [17] B. Bicsi. *Network Design Basics for Cabling Professionals*. McGraw-Hill Professional., (2002).
- [18] Andrew Banks and Rahul Gupta. MQTT Version 3.1.1. Edited by Andrew Banks and Rahul Gupta. 29 October 2014. OASIS Standard.  
<http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>. Latest version: <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>.
- [19] P.Th. Eugster, P.A. Felber, and R. Guerraoui. The Many Faces of Publish/Subscribe.
- [20] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. MQTT-S &#x2014; A publish/subscribe protocol for Wireless Sensor Networks. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, pages 791–798, Bangalore, India, January 2008. IEEE.
- [21] Eclipse Mosquitto. <https://mosquitto.org/>.
- [22] ESP32 Series Datasheet.
- [23] SCT-013-000 Datasheet PDF - Datasheet4U.com.  
<https://datasheet4u.com/datasheet-pdf/XiDiTechnology/SCT-013-000/pdf.php?id=1004703>.

- [24] Arduino\_Uno\_R3.  
[https://upload.wikimedia.org/wikipedia/commons/3/38/Arduino\\_Uno\\_-R3.jpg](https://upload.wikimedia.org/wikipedia/commons/3/38/Arduino_Uno_-R3.jpg).
- [25] Adafruit Industries. Espressif ESP32 Development Board - Developer Edition, November 2016.
- [26] ACS712 Datasheet(PDF) - Allegro MicroSystems.  
<https://www.alldatasheet.com/datasheet-pdf/pdf/168326/ALLEGRO/ACS712.html>.
- [27] Pablo Alcalde San Miguel. *Electrotecnia : Instalaciones Eléctricas y Automáticas p.424*. Paraninfo., Madrid, 6<sup>a</sup> edición edition, (2014).
- [28] SQLite Home Page. <https://sqlite.org/index.html>.
- [29] Matthew Murdoch. Mmurdoch/arduinounit, June 2020.
- [30] Pauli Salmenrinne. Susundberg/arduino-simple-unitest, August 2019.