UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

Diseño e implementación de un dispositivo IoT para monitorización y control remoto de luminarias.

Trabajo Fin de Máster

Máster Universitario en Ingeniería Industrial

AUTOR/A: Lacasa López, Alejandro

Tutor/a: Sanchis Saez, Javier

Cotutor/a: Simarro Fernández, Raúl

CURSO ACADÉMICO: 2023/2024

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCOLA TÈCNICA SUPERIOR ENGINYERIA INDUSTRIAL VALÈNCIA

TRABAJO FIN DE MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

DISEÑO E IMPLEMENTACIÓN DE UN DISPOSITIVO IOT PARA MONITORIZACIÓN Y CONTROL REMOTO DE LUMINARIAS

AUTOR: Alejandro Lacasa López

TUTOR: Javier Sanchis Saez

Curso Académico 2023-24

RESUMEN

En el presente trabajo de fin de máster se aborda el diseño e implementación del prototipo de un dispositivo IoT (Internet of things) para el control de luminarias. Este aparato, podrá ser controlado a través de ordenadores y teléfonos móviles con conexión a internet, independientemente de la red a la que estén conectados.

Mediante el dispositivo se podrá apagar y encender la luminaria, conocer si alguien acciono la luminaria desde el interruptor de pared, conocer la potencia instantánea consumida por la luminaria y visionar el histórico de energía consumida por la luminaria en la franja temporal seleccionada por el usuario.

Con la intención de crear un dispositivo con las funcionalidades anteriormente comentadas, se han elegido de forma justificada los diferentes sistemas hardware y software que lo componen. Para posteriormente diseñar la forma en la que estos se comunicarán.

En cuanto al software, en el presente TFM se recoge el desarrollo de las diferentes capas que conformarán el dispositivo. Se ha programado desde el FW del dispositivo en lenguaje C++, hasta las APIs e interfaz de usuario empleando: flujos de Node-RED, funciones de JavaScript y consultas en lenguaje SQL.

Palabras clave: IoT, Internet de las cosas, Internet of things, UI, User interface, Interfaz de usuario, ESP8266, ACS712, SCT-013, ADS1115, Relé, MQTT, BBDD, MySQL, SQL, Node-RED, Mosquitto, JSON, Lámpara, Luminaria.

RESUM

En el present treball de fi de màster s'aborda el disseny i implementació del prototip d'un dispositiu IoT (Internet of things) per al control de lluminàries. Este aparell, podrà ser controlat a través d'ordinadors i telèfons mòbils amb connexió a internet, independentment de la xarxa a la qual estos estiguen connectats.

Mitjançant el dispositiu es podrà apagar i encendre la lluminària, conéixer si algú accione la lluminària des de l'interruptor de paret, conéixer la potència instantània consumida per la lluminària i visionar l'històric d'energia consumida per la lluminària en l'espai de temps definit per l'usuari.

Per a obtindre un dispositiu amb les funcionalitats anteriorment comentades, s'han triat de forma justificada els diferents sistemes hardware i software que ho componen. Per a posteriorment dissenyar la forma en la qual estos es comunicaran.

En quant al Software, en el present TFM s'arreplega el desenvolupament de les diferents capes que el conformen. S'ha programat des del FW del dispositiu en C++, fins a les APIs i Interfícies d'usuari emprant fluxos de Node-RED, funcions de JavaScript i consultes en llenguatge SQL.

Paraules clau: IoT, Internet de las cosas, Internet of things, UI, User interface, Interfaz de usuario, ESP8266, ACS712, SCT-013, ADS1115, Relé, MQTT, BBDD, MySQL, SQL, Node-RED, Mosquitto, JSON, Lámpara, Luminaria.

ABSTRACT

This master's thesis deals with the design and implementation of the prototype of an IoT (Internet of things) device for the control of luminaires. This device can be controlled through computers and mobile phones with internet connection, regardless of the network to which they are connected.

The device can be used to switch the luminaire on and off, find out if someone has activated the luminaire from the wall switch, find out the instantaneous power consumed by the luminaire and view the history of energy consumed by the luminaire in the time frame selected by the user.

In order to obtain a device with the aforementioned functionalities, the different hardware and software systems that make it up have been chosen in a justified manner. In order to subsequently design the way in which they will communicate with each other.

As for the software, this TFM includes the development of the different layers that make it up. It has been programmed from the FW of the device in C++, to the APIs and user interface using Node-RED flows, JavaScript functions and SQL queries.

Key words: IoT, Internet de las cosas, Internet of things, UI, User interface, Interfaz de usuario, ESP8266, ACS712, SCT-013, ADS1115, Relé, MQTT, BBDD, MySQL, SQL, Node-RED, Mosquitto, JSON, Lámpara, Luminaria.

ÍNDICE DE TÍTULOS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE GRÁFICAS

1-Introducción y objetivos

El objetivo del presente trabajo es diseñar y desarrollar un prototipo de dispositivo IoT para el control de luminarias, así como documentar dicho proceso.

Antes de profundizar más en el objeto del presente TFM será necesario definir qué es IoT. Oracle (una de las mayores compañías de software del mundo, enfocada en desarrollo de soluciones en la nube y locales), define el termino IoT como se muestra a continuación:

"*La Internet de las cosas (IoT) describe la red de objetos físicos ("cosas") que llevan incorporados sensores, software y otras tecnologías con el fin de conectarse e intercambiar datos con otros dispositivos y sistemas a través de Internet. Estos dispositivos van desde objetos domésticos comunes hasta herramientas industriales sofisticadas*." (Oracle, 2024)

Otras grandes organizaciones como Deloitte proponen definiciones muy similares para el término IoT:

"*La definición de IoT podría ser la agrupación e interconexión de dispositivos y objetos a través de una red (bien sea privada o Internet, la red de redes), dónde todos ellos podrían ser visibles e interaccionar. Respecto al tipo de objetos o dispositivos podrían ser cualquiera, desde sensores y dispositivos mecánicos hasta objetos cotidianos como pueden ser el frigorífico, el calzado o la ropa*." (Deloitte, 2024)

A través de las definiciones anteriores y el conocimiento del sector se ha elaborado una definición propia para IoT. Las siglas IoT provienen del inglés, Internet of things, y hacen referencia a aparatos con capacidad para conectarse a internet (habitualmente en ausencia de cableado), con el fin de ser controlados de forma remota y en tiempo real a través de internet mediante una interfaz de usuario o UI.

Comúnmente, el término IoT se encuentra fuertemente unido a la domótica con dispositivos como: Robots aspiradores, cafeteras, cámaras de vigilancia, patinetes eléctricos, etc.

Sin embargo, el internet de las cosas se está haciendo un hueco cada vez más grande en la industria, formando parte de la denominada Industria 4.0. En esta todos los sistemas, desde sensores hasta sistemas MES o ERPs están conectados.

Para que un dispositivo IoT cumpla con su cometido serán necesario tres actores fundamentales que se verán en profundidad más adelante:

- Dispositivo IoT.
- Servidor.
- Interfaz de usuario.

Así pues, en el presente TFM se han desarrollado las soluciones hardware, software y de comunicaciones necesarias para hacer funcionar el dispositivo en cuestión.

En cuanto al hardware, se han elegido los diferentes componentes que forman parte del dispositivo. También se ha diseñado la arquitectura que estos deben seguir para conformar el dispositivo.

En lo que se refiere al software, se ha desarrollado el programa que gobierna el dispositivo, la interfaz de usuario con la que se controla y las interfaces que permitirán comunicar estos dos anteriores. También se ha definido el protocolo de comandos que permite comunicar interfaz de usuario y elemento físico.

Adicionalmente, se han configurado elementos en el servidor como gestores de BBDD o gestores de protocolos de comunicación entre otros.

A continuación, se muestra un diagrama general de las partes que componen el dispositivo.

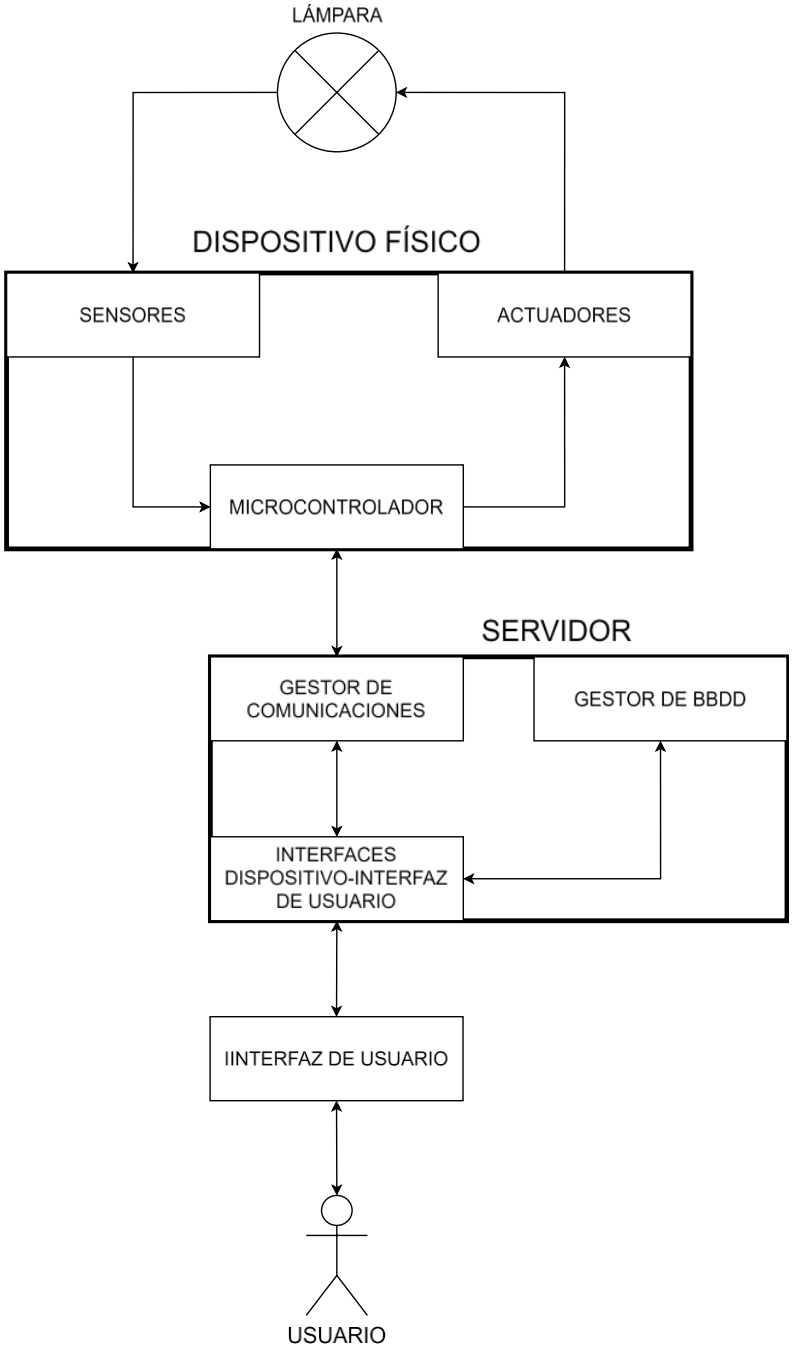


Ilustración 1: Diagrama general de las partes que componen el dispositivo IoT.

1.1-Como funciona IoT

1.1.1-El dispositivo IoT

A través de las definiciones de IoT expuestas con anterioridad, se puede inferir que un dispositivo IoT es un aparato físico con capacidad para conectarse a internet, pudiendo monitorizar ciertas magnitudes físicas de su entorno y actuar sobre ellas.

Esta capacidad para monitorizar y actuar sobre el entorno es percibida por el usuario final como funcionalidades. Independientemente de su ámbito de aplicación, para cumplir con las funciones para las que han sido diseñados deben poseer diferentes sensores y actuadores que le permiten relacionarse con el entorno que les rodea.

Pero ¿De qué sirven los sentidos y las extremidades sin un cerebro que las controle? Es aquí donde entra un tercer elemento; el microcontrolador. Mediante la información recopilada por los sensores y a través del firmware, controlará los actuadores para garantizar un adecuado desempeño del producto.

Hasta este punto, el aparato descrito no difiere, por ejemplo, de una cafetera de cápsulas tradicional, una aspiradora de mano o incluso un microondas. Lo que le otorga a un dispositivo el apellido IoT es su capacidad para conectarse a internet. Desempeñar esta función requiere de la existencia de un módulo WiFi. A través de internet, el aparato recibirá las órdenes dadas por el usuario y le notificará su estado.

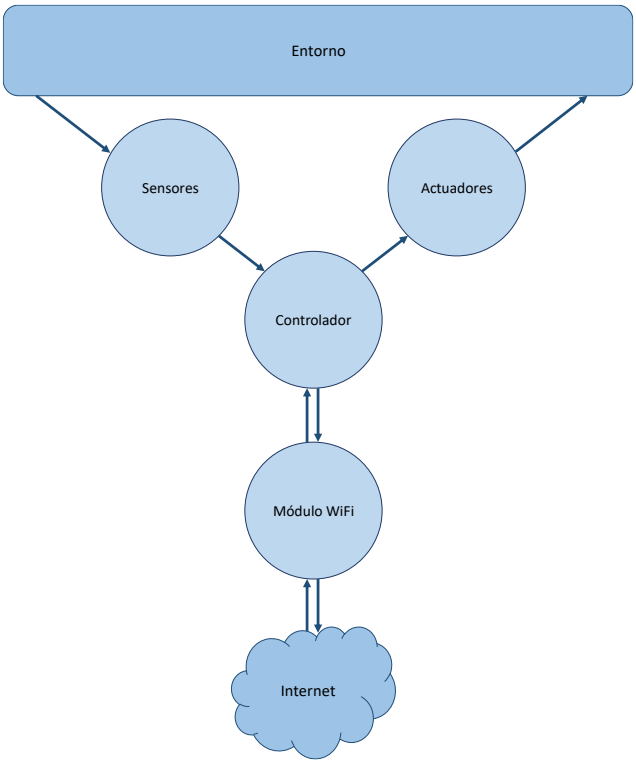


Ilustración 2: Esquema de la arquitectura de un dispositivo WiFi IoT.

Existe otra modalidad de productos IoT, que en lugar de un módulo WiFi poseen un módulo Bluetooth de baja energía (BLE). Estos, no tienen la capacidad para conectarse a internet directamente, pero pueden enlazarse a un dispositivo que sí disponga de dicha capacidad; generalmente un teléfono móvil.

Dado que el dispositivo desarrollado en el presente TFM es del primer tipo, no se hará mayor mención sobre aquellos aparatos que disponen de módulos BLE.

1.1.2-Servidor

Según la RAE (Real Academia Española), un servidor es:

"*Unidad informática que proporciona diversos servicios a computadoras conectadas con ella a través de una red.*" (RAE, 2024)

Acotando la definición de la RAE para el contexto IoT, se puede concluir que el servidor es una unidad informática que proporciona diferentes servicios al dispositivo IoT y a la interfaz de usuario a través de una red, pudiendo ser esta el propio internet.

En función de la ubicación física de los servidores, podemos distinguir dos categorías:

- Servidores locales: Están dispuestos físicamente en la red del cliente.
- Servidores cloud: Están ubicados en grandes centros de datos, como es el caso de Amazon AWS, Azure o IBM.

Este actor es sumamente complejo y está constituido por numerosas partes, que en conjunto permiten el correcto funcionamiento del aparato en cuestión. Su arquitectura varía mucho en función del tipo de aparato. Sin embargo, existe un servicio común a casi todas las aplicaciones, que permite garantizar las comunicaciones ente todos los actores. Este es el bróker MQTT.

Un bróker no es más que un software que permite la comunicación entre dos o más dispositivos. En este caso concreto, un Bróker MQTT se refiere a un software que permite la comunicación entre dos o más actores mediante protocolo MQTT.

En la mayoría de los dispositivos IoT actuales, la comunicación se lleva a cabo mediante protocolo MQTT y no mediante HTTP. El protocolo MQTT es de tipo publicación suscripción. Un actor publica un mensaje en un tema o *topic*, de forma que cualquier otro actor suscrito a ese tema puede leer lo que en él hay publicado. La razón por la que es tan utilizado en esta industria es su ligereza en comparación con el protocolo HTTP.

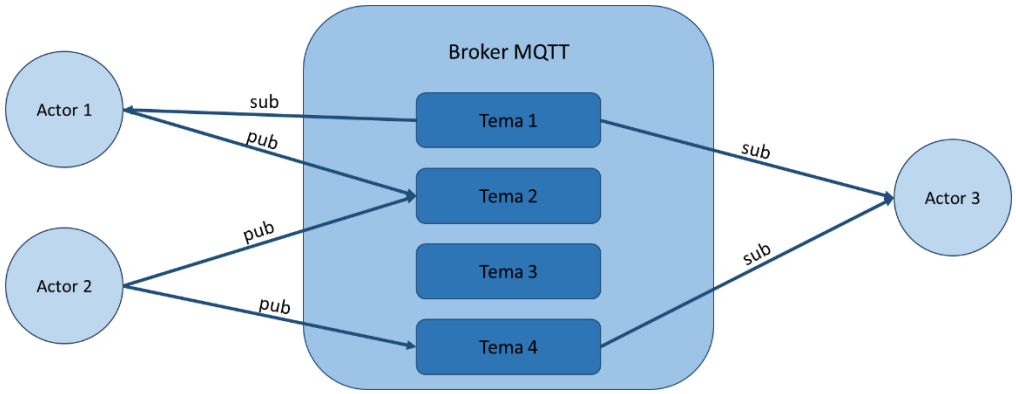


Ilustración 3: Esquema del funcionamiento de un protocolo de comunicación tipo publicación-suscripción.

Habitualmente, se dispone también de servicios complementarios, que, aunque no están ligados directamente a la comunicación, resultan igualmente indispensables para muchas aplicaciones. Entre ellos, podemos encontrar bases de datos junto con sus gestores correspondientes, servicios para verificación de credenciales o APIs.

1.1.3-Interfaz de usuario

Una interfaz de usuario o UI es la parte de un software con la que el consumidor de este interactuará para beneficiarse del servicio que dicho software proporciona.

En los dispositivos IoT las UI permiten al usuario final monitorizar el estado del dispositivo de forma remota y a través de internet. También, otorgan la capacidad de mandar ordenes al mismo, con el fin de ejecutar las diferentes tareas para las que ha sido programado.

Comúnmente el usuario final del dispositivo IoT, tiene acceso a la interfaz a través de una aplicación (móvil o de ordenador) o un navegador web.

2-Motivación

Actualmente se está viviendo auge de la domótica y la venta de electrodomésticos con conectividad crece cada vez más. Por esta razón, resulta interesante la creación de un dispositivo IoT para el control de luminarias, que pueda ser instalado de forma sencilla en viviendas que no fueron concebidas para ser domotizadas.

El interés no reside únicamente en el dispositivo físico, sino también en el desarrollo de los elementos software que se ejecutarán en el servidor. Se pretende que estos sirvan como base para generar una plataforma, que permita incluir dispositivos de diferente naturaleza.

Adicionalmente, este TFM posee una motivación académica, se trata de un reto que me permitirán adquirir una serie de habilidades y conocimientos de alto valor en la industria.

3-Descripción del problema a resolver

Se busca desarrollar el prototipo de un dispositivo capaz de controlar las luminarias de un hogar, pero no sólo desde una interfaz de usuario, sino también mediante los interruptores de pared tradicionales. Esto implica que no existan incompatibilidades entre ambos modos de control.

Con lo que respecta al encendido y apagado de la luminaria, se desea que el dispositivo disponga de dos modos de funcionamiento. En el primero, al que nos referiremos como *Modo Libre,* la lámpara se podrá controlar libremente tanto desde la UI como desde el interruptor de pared. En el segundo, al que nos referiremos como *Modo Bloqueo,* el estado de la luminaria solo podrá ser modificado desde la UI, de manera que si se acciona el interruptor de pared este retorne al estado anterior, adicionalmente el usuario será avisado de este suceso a través de la UI. Se añadirá también la posibilidad de reiniciar el dispositivo como una medida de contención ante posibles bugs.

Otra funcionalidad que resulta indispensable es la capacidad para controlar el dispositivo de forma remota desde cualquier parte del mundo, siempre y cuando se disponga de conexión a internet.

A partir de la funcionalidad anterior nace la necesidad de conocer en tiempo real el estado del dispositivo a través de la UI. De lo contrario, al usuario le sería imposible discernir si la luminaria esta encendida o apagada cuando la manipula en remoto. Adicionalmente, el usuario final tendrá la capacidad de conocer el consumo de la lámpara en tiempo real, así como acceder a gráficos históricos del consumo del dispositivo.

Finalmente, se desea que el dispositivo disponga de versatilidad en lo que a su instalación y uso se refiere. Con esto, se persiguen esencialmente tres objetivos: En primera instancia, debe ser completamente inalámbrico, es decir: que carezca de la necesidad de la existencia de una instalación de red o similares. En segundo lugar, la instalación debe ser lo menos invasiva posible. En tercer lugar, el dispositivo debe poder soportar todas las funcionalidades anteriormente descritas, para un amplio rango de potencias en la luminaria.

Así pues, podemos resumir lo anteriormente comentado en la siguiente lista:

- Operar de forma remota desde cualquier parte del mundo mediante una UI.
- Coexistir con una instalación de alumbrado tradicional.
- Apagar y encender una luminaria con dos modos de funcionamiento:
    - Modo Libre.
    - Modo Bloqueo.
- Reiniciar cuando el usuario lo requiera.
- Conocer el estado de la luminaria.
- Conocer la potencia que demanda la luminaria en tiempo real.
- Mostrar histórico de consumo.
- Operar de forma inalámbrica.
- Instalar fácilmente.
- Soportar amplio rango de potencias de operación.

4-Descripción de la solución adoptada

En él mercado existen soluciones similares a la que se propondrá a continuación. Sin embargo, estas no satisfacen completamente los requisitos expuestos en el apartado 3-Descripción del problema a resolver.

Entre estas soluciones la más famosa es quizás el sistema *Dali*. Este, está diseñado para controlar y monitorizar todas las luminarias de una casa. El sistema *Dali* cuenta con numerosos actores que se interconectan entre sí a través de un bus *Dali*, las luminarias led con las que este opera necesitan integrar obligatoriamente un controlador especifico. Debido a la necesidad de una infraestructura cableada y el uso de luminarias con un controlador específico equipado, el sistema *Dali* carece de la flexibilidad que se busca en el dispositivo objeto del presente TFM.

Otra solución es el uso de luminarias con módulo WiFi integrado, las cuales pueden ser controladas a través de aplicaciones como *Tuya Smart* y carecen de incluir cableado durante su instalación. Sin embargo, presentan problemas para ser compatibles con interruptores de pared tradicionales, cuando el interruptor impide el paso de corriente se corta la alimentación a la bombilla en cuestión y esta no puede operarse mediante la aplicación. Adicionalmente, estas no suelen tener incorporados equipos para medir su consumo y su oferta en cuanto a potencias es limitada.

Ilustración 4: Ejemplo de luminaria con modulo WiFi integrado.

Resumiendo, en el mercado actual es muy complicado encontrar un dispositivo que pueda ser integrado sin necesidad de infraestructura física y que además aporte el abanico de funcionalidades concebidos en el objeto del presente TFM.

4.1-Elección de elementos hardware y software

A continuación, se expondrán los diferentes elementos Hardware y Middleware que se han empleado para poder unificar las funcionalidades definidas anteriormente.

4.1.1-Conectividad

4.1.1.1-Hardware

El primer requisito que debe cumplir el dispositivo es su capacidad para ser operado de forma remota desde cualquier parte del mundo mediante una UI. A nivel Hardware, cualquier dispositivo que sea capaz de satisfacer esta condición debe poseer necesariamente un módulo WiFi y un microcontrolador.

El microcontrolador es necesario para poder recibir señales, procesarlas y responder mediante el uso de los actuadores. Es, por tanto, la parte que nos permite operar el dispositivo. Por otro lado, aquello que dotará al dispositivo físico de conectividad y, por tanto, de poder ser controlado desde cualquier parte del mundo, es un módulo WiFi. Habitualmente, cuando se habla de módulo WiFi, se tiende a omitir que estos dispositivos incluyen su propio microcontrolador, que puede resultar suficiente para aplicaciones que no precisen de la ejecución de pesados cálculos.

Se ha barajado el uso de dos posibles chips como módulo WiFi: ESP8266 y ESP32. Ambos tienen capacidad para conectarse a internet, así como su propio microcontrolador, por lo que parece una solución ideal para satisfacer la presente aplicación. La información acerca de dichos chips se ha recopilado de la página web del fabricante, *Espressif.*

Adicionalmente, ambos presentan la capacidad de ser programados mediante la IDE y APIs propias de Arduino, siendo necesario realizar únicamente pequeñas configuraciones antes de lanzar el proyecto al chip. También, existen tarjetas de desarrollo que emplean estos chips como base y que facilitan enormemente el prototipado.

Sin embargo, existen ciertas diferencias entre ambos, las cuales quedan recogidas en la siguiente tabla.

Chip	Número núcleos del controlador	Número canales DAC	Número bits de canales DAC	Función Bluetooth	Coste unitario (€)
ESP32	2	2	8	✔	4-10
ESP8266	1	0	-	✘	3-6

Tabla 1: Diferencias ESP32 y ESP8266.

De esta lista de características se concluye que los ESP32 poseen mejor rendimiento y capacidades, pero su precio es también más elevado que el de los ESP8266. Por esta razón se ha decidido emplear un ESP8266. Adicionalmente, muchas de las características del ESP32 se estarían infrautilizando para la presente aplicación, como su capacidad para comunicarse mediante Bluetooth o la existencia de canales DAC.

Realmente en el desarrollo del prototipo se ha empleado una placa NodeMCU que usa el ESP8266 como base.

Ilustración 5: Apariencia de una tarjeta NodeMCU que emplea ESP8266 como base.

4.1.1.2-Middleware y servicios

En primera instancia se debe explicar el concepto de Middleware, para ello se hará uso de la definición proporcionada por Amazón Web Service, uno de los proveedores de servicios de computación más grandes del mundo:

"*El middleware es un software con el que las diferentes aplicaciones se comunican entre sí. Brinda funcionalidad para conectar las aplicaciones de manera inteligente y eficiente, de forma que se pueda innovar más rápido. El middleware actúa como un puente entre tecnologías, herramientas y bases de datos diversas para que pueda integrarlas sin dificultad en un único sistema."* (AWS A. , 2024)

En el punto anterior se determinó el microcontrolador que servirá como base para constituir el dispositivo. Sin embargo, para lograr la capacidad de control remoto a través de internet se necesitan dos agentes adicionales: el servidor y la interfaz de usuario.

La máquina donde se alojará el Middleware es el servidor. En lo que a este respecta se ha escogido una opción local. Es decir, se dispondrá de un dispositivo conectado a internet en la red local del usuario, este dispositivo puede ser por ejemplo una RasberryPi, aunque para el prototipado se ha empleado un PC convencional.

Los servicios de los que se dispondrán son cuatro:

- Broker MQTT.
- BBDD.
- API dispositivo-UI.
- Cliente DNS.

En primer lugar, el Broker MQTT gestionará mediante protocolo MQTT las comunicaciones entre dispositivo y UI. Se han barajado tres opciones:

- Eclipse Mosquitto: Para conocer más información se puede visitar el siguiente sitio web https://mosquitto.org.
- Hive MQ: Para conocer más información se puede visitar el siguiente sitio web https://www.hivemq.com.
- EMQ X Broker: Para conocer más información se puede visitar el siguiente sitio web https://www.emqx.io.

Dado que se busca una opción local y open source se ha descartado Hive MQ. Entre Eclipse Mosquitto y EMQ X Broker se ha escogido la primera opción. Ambos brókeres son open source. Sin embargo, Mosquitto parece a priori una opción más sencilla de integrar que cuenta con gran cantidad de documentación.

En segundo lugar, se necesitará un servicio de BBDD para garantizar el visionado histórico de consumo a través de la UI. Se han barajado tres tecnologías para almacenamiento de datos:

- BBDD relacionales SQL.
- BBDD NoSQL.
- Almacenamiento en ficheros.

Es cierto que, para poder implementar esta funcionalidad, no será necesario desarrollar complejas estructuras y relaciones en lo que a BBDD se refiere. Sin embargo, realizar operaciones de lectura y escritura sobre ficheros de texto o similares puede generar latencia en el visionado del histórico, sobre todo si se desean observar largos periodos de tiempo. Esto se debe a la necesidad de recorrer todas las filas del fichero, para posteriormente tratar toda la información.

Entre BBDD SQL y NoSQL se ha escogido la primera opción, los motivos de esta elección es la existencia de un lenguaje estandarizado para la realización de consultas. Adicionalmente, las BBDD NoSQL suelen usarse para masivas cantidades de datos que poseen una estructura no muy bien definida, esta aplicación se aleja mucho de las necesidades del presente trabajo.

Se empleará una BBDD SQL y SQLite como gestor de esta. Existen otros gestores de BBDD, pero suelen estar enfocados a trabajos en remoto como MySQL o son de pago como Microsoft SQL Serve. Además, SQLite es altamente indicado para prototipado por su elevada flexibilidad.

En tercer lugar, será necesaria una UI donde poder operar y monitorizar el dispositivo, para dar solución a este problema se ha optado por emplear el Dashboard de node-red. Node-red es una herramienta de programación que permite unir dispositivos hardware, APIs y servicios online, esta herramienta nos permite cubrir los siguientes puntos asociados a la UI:

- Back-end: Es la parte encargada de realizar todas las tareas que el usuario no ve directamente mediante la UI. En la presente aplicación cumplirá las siguientes funciones:
    - Leer los datos dispuestos en los temas del bróker a los que este suscrito.
    - Publicar los datos en los temas del bróker correspondientes para que sean leídos por el dispositivo.

    - Realizar las consultas de lectura y escritura a la BBDD cuando sea necesario.
- Front-end: Es esencialmente la UI con la que interactúa el usuario. Será generada a través del Dashboard de node-red y deberá incluir:
    - Indicador del estado de la luminaria.
    - Consumo actual de la luminaria.
    - Actuador para encender/apagar la luminaria.
    - Capacidad para cambiar el modo de trabajo.
    - Visualización del histórico de consumo.
    - Capacidad para reiniciar el dispositivo.
- API: Son siglas que provienen del inglés, Application programming interface. Para la presente aplicación permitirán adaptar los datos que recoge del back-end al formato que requiere el front-end.

Finalmente se requerirá poder operar el dashboard desde cualquier dispositivo que disponga de conexión a internet, independientemente del lugar del mundo donde se encuentre el usuario. Para comprender la siguiente solución será necesario explicar a grandes rasgos cómo funciona el sistema de direcciones IP públicas y privadas.

Cuando un dispositivo se conecta a una red local (LAN) se le otorga una IP, un código numérico único que lo identifican dentro de esta. La IP tiene dos partes, la primera es común a todos los dispositivos conectados a dicha red y la segunda es única para cada elemento de esta. A continuación, se adjunta una imagen que ejemplifica dicho funcionamiento.

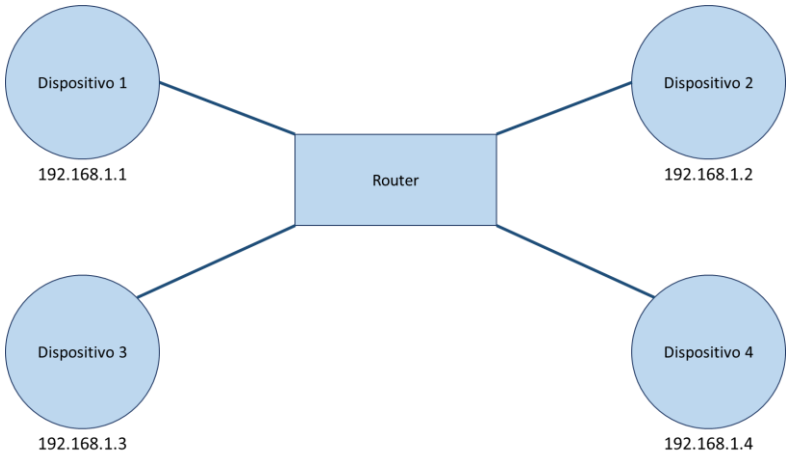


Ilustración 6: Esquema de una red local.

Cuando se está operando un dispositivo y se accede a internet, no se emplea la IP que este posee dentro de la LAN, IP privada. En su lugar el aparato en cuestión se comunica con el router, que accede a internet haciendo uso de una IP pública, finalmente le traslada los contenidos solicitados al dispositivo en cuestión. Cada vez que un usuario accede a internet dentro de una LAN lo hace a través de la IP pública del router. A continuación, se emplea una imagen que ejemplifica el funcionamiento.

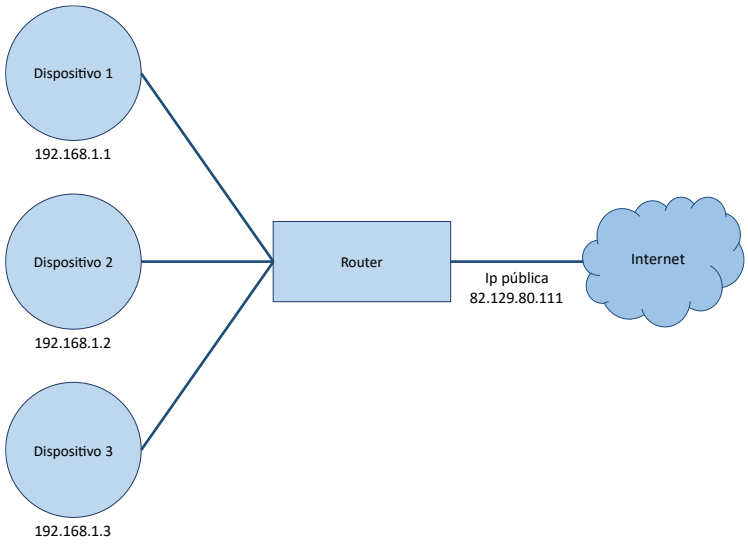
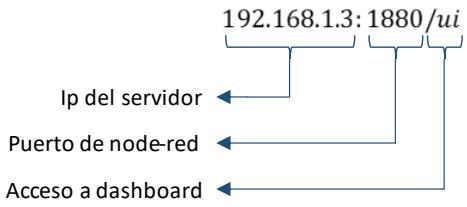


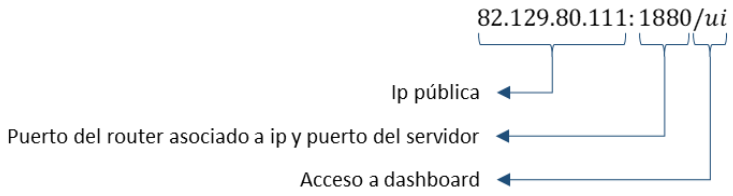
Ilustración 7: Esquema de una red local con acceso a internet.

Si dentro de una LAN existe un dispositivo donde está corriendo node-red, otro aparato de esa misma LAN podrá acceder a este servicio empleando un navegador web. Para ello se deberá introducir la IP privada del servidor de node-red, seguido del puerto asignado a dicho servicio, el puerto por defecto para node-red es el 1880. Si al final de dicha expresión se introducen los caracteres "/ui" se podrá visualizar e interactuar con el dashboard.



Expresión 1: Ejemplo de acceso a node-red desde dentro de LAN.

Cuando el servidor y el aparato que solicita acceso no están dentro de la misma red se debe emplear la IP pública del router. Sin embargo, es necesario aplicar ciertas configuraciones sobre este, el objetivo de estas es desviar todo el tráfico que pase por un puerto concreto del router a la IP del servidor, concretamente al puerto donde está corriendo node-red. Para este caso concreto se conectarán el puerto 1880 del router con el puerto 1880 del servidor de node-red. A esta práctica se la conoce como *port forwarding*.



Expresión 2: Ejemplo de acceso a node-red desde fuera de LAN.

La IP pública que proporciona el router es dinámica, lo que implica que va cambiando, sería inviable tener que conocer la IP pública de nuestra red cada vez que se desea acceder al dashboard de node-red. Habitualmente este problema se soluciona mediante servidores DNS dinámicos, que asocian en su interior urls a direcciones IP. Se puede entender una url como un código alfanumérico que ubica de forma inequívoca un recurso en la web.

Cuando se pulsa sobre una url en un dispositivo con acceso a internet, es un servidor DNS el que le proporciona la IP asociada a dicha cadena alfanumérica, permitiendo al navegador web acceder a los contenidos solicitados por el usuario.

Por esta razón, será necesario instalar en el servidor un cliente DNS, que envíe cada cierto tiempo la IP pública de la LAN a un host DNS en la nube. En este, será donde la IP dinámica del rúter quede asociada a una dirección url estática, pudiendo garantizar el acceso al dashboard.

Se han estudiado tres proveedores diferentes:

- DuckDNS: Para conocer más información se puede visitar el siguiente sitio web https://www.duckdns.org.
- No-IP: Para conocer más información se puede visitar el siguiente sitio web https://www.noip.com/es-MX.
- DynDNS: Para conocer más información se puede visitar el siguiente sitio web https://account.dyn.com.

En primer lugar, DynDNS es un host de pago, con lo que queda descartado debido a la finalidad académica del presente trabajo. Habitualmente, los hostings DNS de pago ofrecen funcionalidades de las que carecen los proveedores gratuitos, como la capacidad para modificar la propia url.

En cuanto a No-IP, es necesario reactivar la url de forma periódica para evitar su caducidad, lo que puede resultar bastante incómodo durante el desarrollo. Finalmente, se ha escogido DuckDNS como proveedor. Esta carece de la necesidad de reactivar las urls, además, la configuración del host en la nube y del servicio local es muy sencilla e intuitiva, disponiendo incluso de una interfaz en ambos casos.

4.1.2-Sensores

Tal y como se expuso en la definición de funcionalidades, el usuario final debe ser capaz de identificar a través de la UI el estado de la luminaria y la potencia en tiempo real. También, deberá poder visualizar el histórico de consumo. Para cumplir con estos tres requisitos se necesita algún elemento Hardware capaz de medir corriente. Para lo cual se estudió el uso de dos posibles sensores: ACS712 o SCT-013.

4.1.2.1-Sensor ACS712

El primero será un sensor de la serie ACS712, se trata de un sensor invasivo, pues para instalarlo será necesario seccionar el conductor por donde circula la corriente a medir y conectar ambos extremos al ACS712.

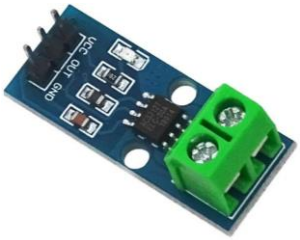


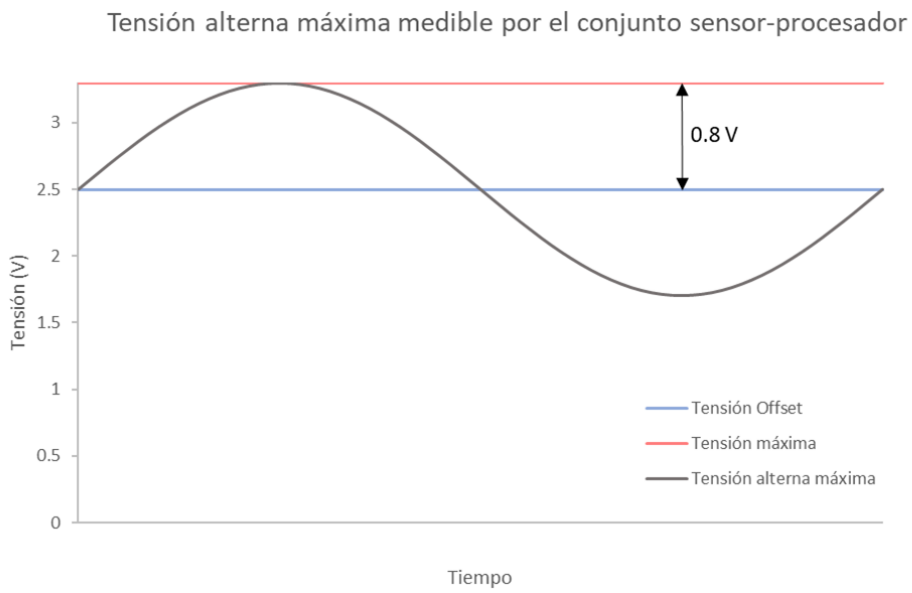
Ilustración 8: Apariencia de los sensores de medición de corriente de la serie ACS712.

Esta serie de sensores precisan de alimentación a 5V y otorgan una tensión de salida proporcional a la corriente que miden, dicha tensión de salida presentará un desfase u offset de 2.5 V. La sensibilidad del sensor dependerá del modelo concreto tal y como se observa en la siguiente tabla.

Modelo	Sensibilidad (mV/A)	Offset (V)
ACS712X05	185	2.5
ACS712X20	100	2.5
ACS712X30	66	2.5

Tabla 2: Sensibilidad y Offset de los sensores de la serie ACS712.

Un punto importante para tener en cuenta en la elección del sensor será la corriente RMS máxima que se podrá medir con el conjunto formado por sensor y ESP8266. Mientras que el sensor otorga un rango de salida de 0-5 V, el controlador solo podrá detectar sin saturar rangos de tensión entre 0-3.3 V, esto es debido a que el controlador ESP8266 ha de alimentarse a 3.3 V. Es por esta razón que la amplitud de la onda sinusoidal que el conjunto sensor-controlador será capaz de medir es de 0.8 V tal y como se aprecia en la siguiente gráfica.



Gráfica 1: Tensión alterna máxima medible por el conjunto ACS712-ESP8266.

Como se puede apreciar en la imagen, la tensión alterna que otorga el sensor en su salida deberá tener una amplitud máxima de 0.8 V para poder realizar medidas sin saturar el controlador. Para obtener el valor de tensión máxima deberemos aplicar la siguiente fórmula.

$$V_{RMS} = \frac{V_{max}}{\sqrt{2}} = \frac{0.8}{\sqrt{2}} = 0.566\,V$$

Ecuación 1: Relación entre tensión pico y tensión eficaz.

Será necesario determinar la intensidad RMS máxima que se podrá medir sin saturar para cada modelo de ACS712, dicho dato podrá ser extraído a través de la sensibilidad del sensor tal y como se aprecia en la siguiente ecuación.

$$I_{IN} = \frac{V_{out}}{S}$$

Ecuación 2: Relación entre intensidad efectiva de entrada al sensor y tensión efectiva de salida.

Modelo	Sensibilidad (mV/A)	Tensión RMS max (V)	Corriente RMS max (A)	P max (W)
ACS712X05	185	0.57	3.06	703.28
ACS712X20	100	0.57	5.66	1301.08
ACS712X30	66	0.57	8.57	1971.33

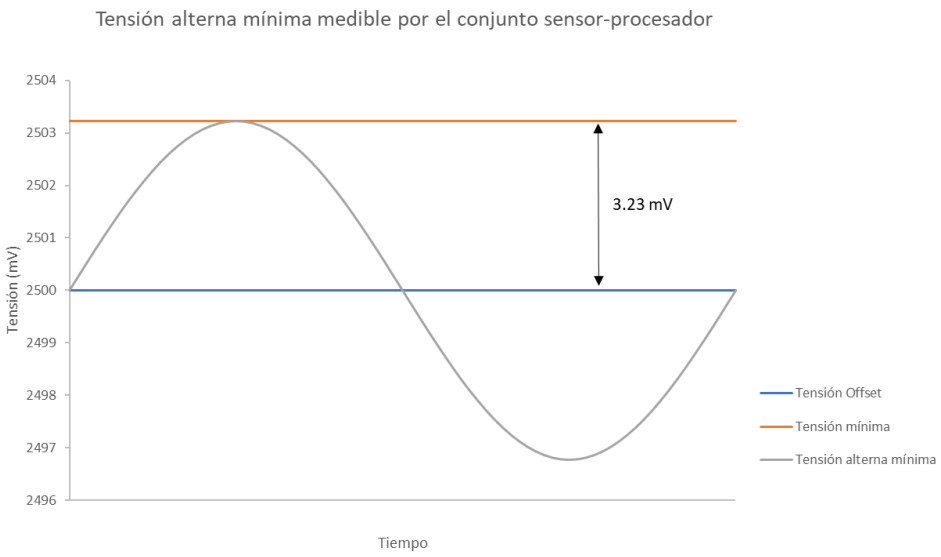
Tabla 3: Corriente y potencia eficaz máxima medible para el conjunto sensor-controlador para cada modelo de ACS712.

En la tabla superior se puede ver el valor de corriente efectiva máxima medible por el conjunto sensor-controlador para cada modelo de sensor ACS712, estos datos se han calculado mediante las expresiones comentadas con anterioridad. Se aprecia como se ha traducido la corriente a potencia, pues es un dato más habitual en la hoja de características de las luminarias.

Ahora será necesario realizar el cálculo de la mínima corriente eficaz medible por el conjunto sensor-controlador. En este caso será el convertidor analógico digital (ADC) del ESP8266 el que limite la medición. El ESP8266 posee un convertidor ADC de 10 bits y como se comentó anteriormente debe alimentarse a 3.3 V, se puede calcular la resolución de dicho convertidor empleando la siguiente expresión.

$$R = \frac{V_{alim}}{2^N} = \frac{3.3}{2^{10}} = 3.23 \ \frac{mV}{unidad}$$

Ecuación 3: Cálculo de la resolución de un convertidor analógico digital.



Gráfica 2: Tensión alterna mínima medible por el conjunto ACS712-ESP8266.

Para obtener la corriente efectiva y potencia mínima medible por el conjunto sensor-controlador se realizaron cálculos análogos a los de los valores máximos, así se obtuvieron los siguientes resultados.

Modelo	Sensibilidad (mV/A)	Tensión RMS min (mV)	Corriente RMS min (A)	P min (W)
ACS712X05	185	2.28	0.01	2.84
ACS712X20	100	2.28	0.02	5.25
ACS712X30	66	2.28	0.03	7.95

Tabla 4: Corriente y potencia eficaz mínima medible para el conjunto sensor-controlador para cada modelo de ACS712.

4.1.2.2-Sensor SCT-013 y ADS1115

El segundo será un sensor de la serie SCT-013, se trata de un sensor no invasivo con forma de pinza que se coloca alrededor del conductor cuya corriente desea ser medida.

Ilustración 9: Apariencia de los sensores de medición de corriente de la serie SCT-013

Esta serie de sensores no precisan de alimentación y otorgan una tensión de salida proporcional a la corriente que miden, dicha tensión de salida no presentará offset alguno. La sensibilidad del sensor dependerá del modelo concreto de sensor tal y como se observa en la siguiente tabla.

Modelo	Sensibilidad (mV/A)
SCT-013-005	200.00
SCT-013-010	100.00
SCT-013-015	66.67
SCT-013-020	50.00
SCT-013-025	40.00

Tabla 5: Sensibilidad de los sensores de la serie SCT-013.

En la presente aplicación se desea medir corriente alterna, con lo que a la salida del sensor seguro aparecen tensiones negativas, esto supone un problema, pues el rango de medición de tensiones para el ESP8266 es de 0-3.3 V. Si conectásemos el sensor directamente al controlador este saturaría inferiormente proporcionando una medida similar a la gráfica inferior.

Gráfica 3: Tensión medida por el controlador al conectarlo directamente al sensor de familia SCT-013.

Dado que necesitamos conocer la potencia consumida por la luminaria en tiempo real, esta no es una opción. Como solución se optó por emplear un convertidor analógico digital externo, que admitiese medición diferencial y transmitiese los valores medidos mediante protocolo I2C y no a través de un lazo de tensión. El ADC escogido es un ADS1115.

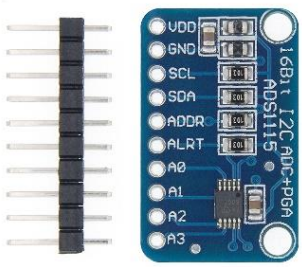


Ilustración 10: Apariencia del convertidor analógico digita ADS1115.

Este conversor puede ser alimentado en un amplio rango de tensiones 2-5.5 y es de 16 bits. Dado que admite la medida de valores positivos y negativos uno de estos bits se empleará para determinar el signo, con lo que realmente cuenta con 15 bits para la medición del valor absoluto de la tensión. Este elemento se comunicará con el ESP8266 mediante protocolo I2C. Este es un protocolo de comunicación serie para transferir bits entre dos o más dispositivos digitales. I2C emplea una arquitectura de tipo maestro-esclavo, para la presente aplicación el ESP8266 actuará como maestro y el ADS1115 como esclavo. El protocolo I2C emplea un bus de dos hilos:

- SCL (Serial Clock): Es la señal de reloj generada por el maestro y permite sincronizar la transferencia de datos.
- SDA (Serial Data): Es la línea de datos por la cual se transmiten los bits de información.

Para gestionar esta comunicación controlador-ADC los dispositivos de la familia ADS disponen de una librería compatible con la IDE de Arduino. A continuación, se adjunta un pequeño esquema de cómo se realizaría el conexionado del ADS1115 para la presente aplicación.

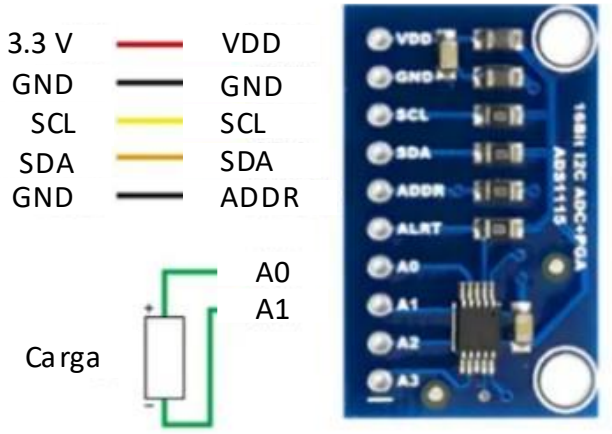


Ilustración 11: Esquema de conexionado ADS1115.

Al igual que se realizó con la solución anterior se debe estudiar cuales son los valores de corriente RMS y potencias máximas y mínimas que se podrían obtener empleando sensores de la serie SCT y el ADS1115.

Se conoce a través de la datasheet del producto cual es la corriente RMS máxima que puede medir cada modelo de sensor SCT. Sin embargo, dado que el ADC no admite tensiones de entrada superiores a su alimentación (3.3 V para esta aplicación) se ha calculado la tensión pico máxima que podría devolver cada modelo de sensor a su salida.

En aquellos casos que superan la tensión de alimentación, se ha fijado como tensión de entrada pico 3.3 V, que equivale aproximadamente a un valor eficaz de 2.33 V. Finalmente, se calculó la máxima corriente eficaz medible que pasa por el conductor para el conjunto sensor-ADC. Este valor de corriente ha sido traducido a potencia, multiplicándolo por la tensión eficaz presente en las redes de baja tensión (230 V). Los resultados pueden apreciarse en la tabla inferior.

Modelo	Sensibilidad (mv/A)	Corriente inst max SCT (A)	Corriente RMS max SCT (A)	Tensión RMS max SCT (V)	Tensión RMS max ADS1115 (V)
SCT-013-005	200	30	21.21	4.24	2.33
SCT-013-010	100	35	24.75	2.47	2.33
SCT-013-015	66.67	40	28.28	1.89	2.33
SCT-013-020	50	50	35.36	1.77	2.33
SCT-013-025	40	55	38.89	1.56	2.33

Modelo	Sensibilidad (mv/A)	Tensión RMS max (V)	Corriente RMS max (A)	Tensión red RMS (V)	P max (W)
SCT-013-005	200	2.33	11.67	230.00	2683.47
SCT-013-010	100	2.33	23.33	230.00	5366.94
SCT-013-015	66.67	1.89	28.28	230.00	6505.38
SCT-013-020	50	1.77	35.36	230.00	8131.73
SCT-013-025	40	1.56	38.89	230.00	8944.90

Tabla 6: Corriente y potencia eficaz máxima medible para el conjunto sensor-ADC para cada modelo de SCT.

Los valores mínimos que esta solución permite medir vendrán determinados por la resolución del convertidor y la sensibilidad del sensor. Considerando 15 bits para la medición del valor absoluto de la tensión, se tiene una resolución aproximada de 0.1 mV/unidad para una alimentación de 3.3 V. A continuación, se presenta una tabla con las corrientes RMS y potencias mínimas que se podrán medir.

Modelo	Sensibilidad (mv/A)	Corriente RMS min sensor-ADC (mA)	P min (W)
SCT-013-005	200.00	0.36	0.08
SCT-013-010	100.00	0.71	0.16
SCT-013-015	66.67	1.07	0.25
SCT-013-020	50.00	1.42	0.33
SCT-013-025	40.00	1.78	0.41

Tabla 7: Corriente y potencia eficaz mínima medible para el conjunto sensor-ADC para cada modelo de SCT.

4.1.2.3-Elección de sensor de medición de corriente

Como se comentó en la definición de funciones, se desea que el dispositivo otorgue una alta flexibilidad, es decir, que pueda ser instalado tanto en pequeñas lámparas LED de pocos vatios como en potentes luminarias. Con la opción del sensor intrusivo se dispone de un rango de medición bastante más pequeño que con el sensor tipo pinza, preocupa especialmente el límite inferior pues se pueden encontrar luminarias en el mercado de potencia inferior a 2.84 W, en lo que respecta al límite superior es más que suficiente para ambas opciones en el contexto de la presente aplicación.

Por otro lado, para poder desarrollar el dispositivo empleando el sensor ACS712 se necesitaría contar con dos fuentes de alimentación diferentes: una para el sensor que necesita de 5 V y otra para el ESP8266 que requiere de 3.3 V. Alternativamente, se podría emplear una única fuente de 5 V, introduciendo en el diseño del dispositivo un circuito reductor para poder alimentar el controlador a 3.3 V.

Al emplear el sensor SCT-013, se requerirá de un convertidor analógico digital de 16 bits, que si bien otorgará muchísima más precisión en la medida también introducirá un incremento de coste de fabricación. A pesar de esto, el coste de los sensores de la serie ACS712 es inferior a los de la serie SCT, adicionalmente si se desea emplear el medidor de corriente tipo pinza será necesario un ADC externo que incrementará el coste aún más. Se puede concluir por tanto que la segunda solución resulta más cara que la primera.

Finalmente se decidió emplear el conjunto de sensor SCT y ADS1115, pues a pesar de su precio elevado permite una mayor flexibilidad y facilidad en cuanto a la instalación, concretamente se ha optado por el modelo SCT-013-005 pues otorga mejores características de medición mínimas.

4.1.3-Actuadores

Mediante la UI, el usuario mandará una orden al microcontrolador para encender o apagar la lámpara en cuestión, por esta razón se debe dotar al dispositivo de un mecanismo para cambiar el estado de la luminaria a través del ESP8266. El elemento escogido para esta tarea será un relé. Mediante una señal de tensión, el microcontrolador será capaz de modificar el estado del relé, a través de la excitación de una bobina que este posee en su interior.

El relé debe estar alimentado, preferiblemente a la misma tensión que el microcontrolador para ahorrar en elementos hardware adicionales. También debe ser capaz de abrir y cerrar un conductor expuesto a la tensión de la red (230 V AC) y por el que circula corriente.

Conociendo estos factores se ha escogido un relé alimentado a 3.3 V y con capacidad para abrirse y cerrarse en condiciones límite de 10 A y 250 V ambos en corriente alterna.

4.1.4-Resumen de la elección de sistemas

Para concluir el apartado 4.1-Elección de elementos hardware y software, se va a repasar todos los elementos seleccionados que conformarán el dispositivo.

En primer lugar, como microprocesador que gobernará el dispositivo se va a emplear un chip ESP8266. Para conocer la corriente consumida por la o las luminarias a las que se enlazará el dispositivo se va a emplear un sensor de corriente SCT-013-005 junto con un convertidor analógico digital ADS1115. Adicionalmente, se empleará un relé de 3.3 V para poder actuar sobre la o las luminarias en cuestión.

En segundo lugar, como sistema encargado de la gestión de comunicaciones entre dispositivo e interfaz de usuario se empleará Mosquitto. Por otro lado, Node-red será la plataforma empleada para desarrollar la UI, así como la APIs que la comunicarán con Mosquitto.

Finalmente, se han seleccionado dos servicios adicionales. SQLite será el gestor de la BBDD, este escribirá o leerá de la BBDD en función de las consultas realizadas a través de las APIs de node-red. Para poder controlar el dispositivo desde cualquier parte del mundo, será necesario un servicio DNS, DuckDNS será el encargado de aportar esta funcionalidad.

4.2-Desarrollo del prototipo

En este apartado, se expondrá como se ha logrado elaborar un prototipo que cumpla satisfactoriamente las funciones definidas con anterioridad, integrando las diferentes soluciones adoptadas que se comentaron en el punto anterior.

4.2.1-Hardware

Se ha elaborado un banco de trabajo que incluye tanto el prototipo como los elementos necesarios para validar las diferentes funciones del dispositivo. A continuación, se adjunta un esquema de como se ha realizado el conexionado de cada una de las partes.

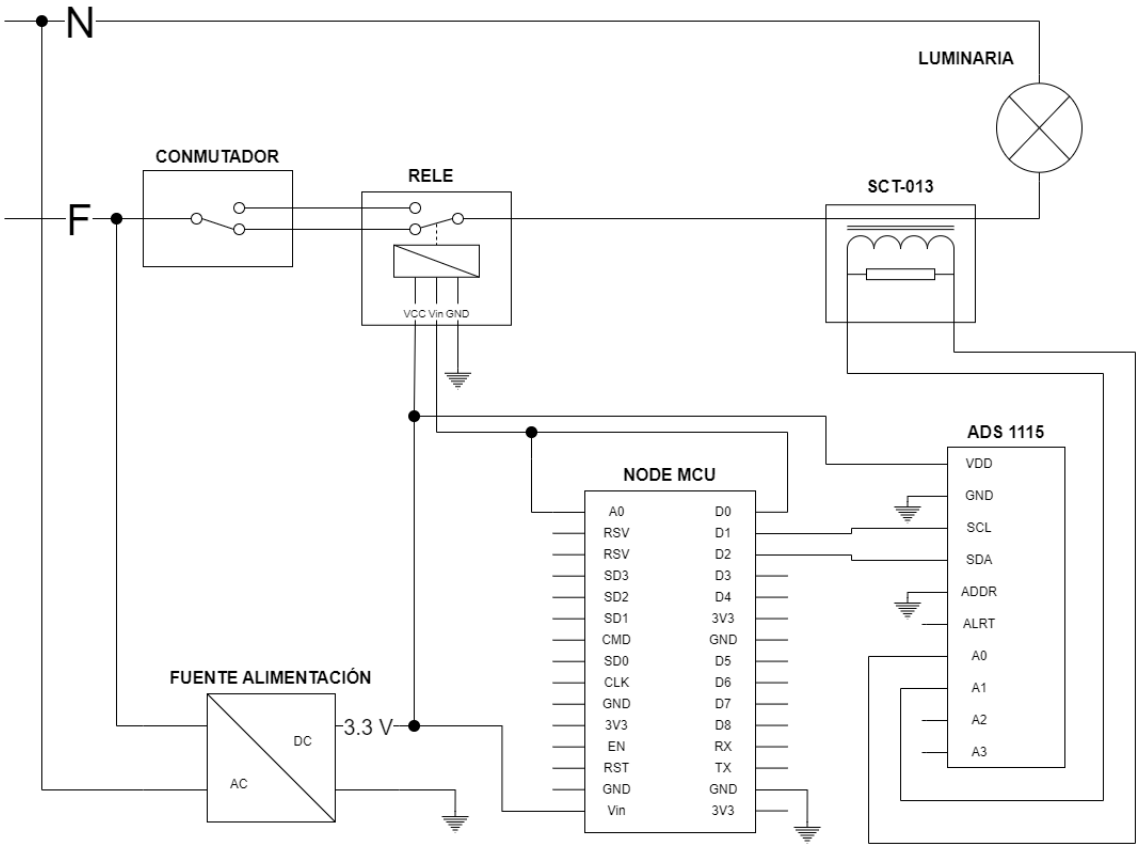


Ilustración 12: Esquema eléctrico del banco de trabajo junto al prototipo.

A grandes rasgos se pueden distinguir dos zonas en el esquema, la zona de alterna con una tensión entre fase y neutro de 230 V y la zona de continua con una tensión respecto a masa de 3.3 V. Para transformar la corriente alterna a continua se hará uso de una fuente de alimentación.

En la zona de alterna se pueden observar cuatro elementos. El primero de ellos es un conmutador de pared tradicional, este se ha empleado en la validación de una de las funcionalidades descritas en el apartado 3-Descripción del problema a resolver, “Coexistir con una instalación de alumbrado tradicional”. A continuación, se han dispuestos el relé y sensor de corriente expuestos anteriormente. Finalmente está la luminaria conectada a la línea de fase y neutro que también será clave en la validación del modelo.

En cuanto a la parte de continua, el NodeMCU actúa como punto central. Su pin digital D0 se empleará para conmutar el relé, debido a la conexión entre relé y conmutador, un nivel alto en D0 no siempre corresponderá con un encendido de la lámpara y viceversa, para conocer el estado de dicho pin este se ha conectado a A0, que actúa como entrada analógica a la tarjeta.

El NodeMCU se deberá comunicar también con el ADS 1115 mediante protocolo I2C, para ello se han conectado las señales SCL y SDA a los pines D1 y D2, pines reservados para dicha comunicación.

Finalmente, la pinza medidora de tensión se ha enlazado con el ADC externo en modo medición diferencial tal y como se había comentado con anterioridad.

4.2.2-Transmisión de mensajes

Antes de comenzar a programar el ESP8266, las APIs o el dashboard se debe aclarar dos aspectos fundamentales en lo referente a la comunicación entre dispositivo y dashboard.

En primer lugar, se van a crear dos temas en el bróker MQTT:

- sub_dispositivo: En este tema publicará el back-end (definido en el punto 4.1.1.2-Middleware y servicios) del dashboard y el ESP8266 estará suscrito. La información viajará desde el dashboard hasta el dispositivo.
- pub_dispositivo: En este tema publicará el ESP8266 y estará suscrita el back-end del dashboard. La información viajará desde el dispositivo hasta el dashboard.

En segundo lugar, se debe definir un protocolo de transmisión de comandos que permitan tanto al dashboard como al dispositivo interpretar la información recibida y emplearla debidamente. Por esta razón, los mensajes que se publiquen en ambos temas van a ser objetos JSON serializados, Es decir, cadenas de caracteres que siguen la estructura de un objeto JSON. Esta manera de intercambiar información permite compartir en un solo mensaje el valor de múltiples variables de una manera ordenada y fácil.

Una buena definición del formato JSON es la que proporciona Amazon AWS:

"JSON es un formato para el intercambio de datos que pueden leer tanto humanos como las máquinas. Aunque el nombre JSON es el acrónimo de JavaScript Object Notation (notación de objetos JavaScript), el formato de JSON es independiente de cualquier lenguaje de programación … JSON representa los datos de dos formas:

- *Un objeto, que es una colección sin ordenar de pares de nombre-valor. Un objeto se define entre las llaves izquierda ({) y derecha (}). Cada par de nombre-valor comienza por el nombre, seguido de dos puntos, seguido del valor. Los pares de nombre-valor están separados por comas.*
- *Una matriz, que es una colección ordenada de valores. Una matriz se define entre los corchetes izquierdo ([) y derecho (]). Los elementos de la matriz están separados por comas."*

(AWS A. , 2024)

Los mensajes que se publiquen en el tema sub_dispositivo, informarán al dispositivo de los cambios que efectúa el usuario en el dashboard, estos mensajes van estarán constituidos por tres pares clave valor:

- Switch: Tendrá asociado un valor booleano, cuando desde la UI se presione el botón para encender o apagar la luminaria se negará su valor actual, indicará al ESP8266 que desde el dashboard el usuario ha solicitado modificar el estado de la lámpara.
- Modo: Tendrá asociado un valor en forma de cadena de caracteres, esencialmente se le asociará dos valores "Libre" o "Bloqueo", informa al ESP8266 de en qué modo de los dos posibles debe operar.
- Reiniciar: Tendrá asociado un valor booleano, indicará al ESP8266 cuando el usuario ha solicitado reiniciar el dispositivo desde el dashboard.


```
{
  "Switch": true,
  "Modo": "Libre",
  "Reiniciar": false
}
```


Ilustración 13: Ejemplo de mensaje publicado en el tema sub_dispositivo.

Los mensajes que se publiquen en el tema pub_dispositivo, informarán al dashboard del estado del dispositivo, estos mensajes van estarán constituidos por cinco pares clave valor:

- Potencia_consumida: Tendrá asociado un valor decimal. Informará al dashboard de la potencia que está consumiendo la lámpara.
- Led: Tendrá asociado un valor booleano. Informará al dashboard de si la lámpara está encendida o apagada.
- Modo: Tendrá asociado un valor en forma de cadena de caracteres, esencialmente se le asociará dos valores "Libre" o "Bloqueo". Indicará al dashboard en qué modo está operando la lámpara en dicho instante.

- Bloqueo_roto: Tendrá asociado un valor booleano. Indicará al dashboard si el estado de la lámpara fue modificado mediante un conmutador de pared cuando el dispositivo estaba en modo bloqueo.
- Fin_reinicio: Tendrá asociado un valor booleano. Indicará al dashboard si el dispositivo ha terminado de reiniciarse, mientras el dispositivo no se esté reiniciando se dispondrá a nivel alto.


```
{
  "Potencia_consumida": 0,
  "Led": false,
  "Modo": "Libre",
  "Bloqueo_roto": false,
  "Fin_reinicio": false
}
```


Ilustración 14: Ejemplo de mensaje publicado en el tema pub_dispositivo

4.2.3-Firmware

En el contexto del presente proyecto se entiende por Firmware al Software que controla la tarjeta NodeMCU y se encuentra almacenado en la memoria de esta. Dicho Firmware se ha elaborado y embebido en el interior de la tarjeta empleando la IDE de Arduino.

A grandes rasgos el proyecto se ha dividido en el módulo principal y tres módulos de funciones, en estos últimos es donde se han definido las funciones principales y auxiliares que se emplearan en el módulo principal. Esta estructura se ha elaborado con el fin de facilitar enormemente la depuración del código. Los módulos de funciones son:

- *Conexion_MQTT*
- *Lampara*
- *Logica*

A continuación, se procederá a explicar la estructura del código de lo más pequeño a lo más grande.

4.2.3.1-Conexion_MQTT

En este módulo se recogen todas las funciones referentes a la conectividad y comunicación del prototIPo, tanto con la red como con el bróker MQTT. Este módulo se apoya en tres librerías de Arduino.

- *ESP8266WiFi*: Facilita la implementación de la conexión del módulo ESP8266 a una red WiFi.
- *PubSubClient*: Facilita la implementación de la comunicación mediante protocolo MQTT.
- *ArduinoJson*: Permite trabajar con objetos tipo JSON en la IDE de Arduino.

El proceso de conexión a la red WiFi se realizará mediante una función de nombre *ESP_to_Router*, empleando la API proporcionada por la librería ESP8266WiFi. Tras aportar el SSID y la contraseña de la red WiFi el ESP8266 tratará de conectarse a esta y no parar de intentarlo hasta que lo consiga.

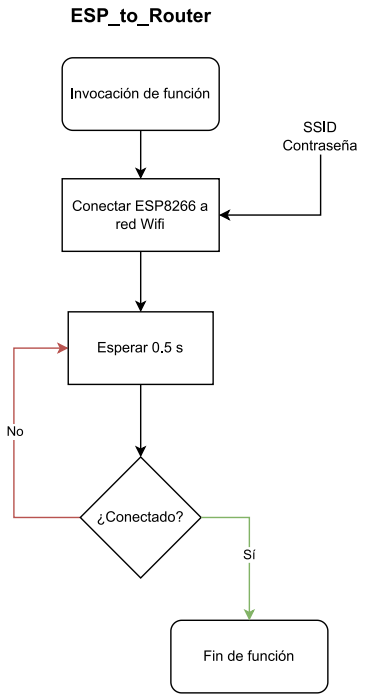


Ilustración 15: Diagrama de flujo de la función ESP_to_Router.

A continuación, se desarrolló la función que permitirá al ESP8266 conectarse como cliente al Bróker MQTT y subscribirse a los temas en los que se va a realizar el intercambio de información, a esta se le ha asignado el nombre de *ESP_to_BrokerMQTT*. Para el desarrollo del presente dispositivo se han empleado dos temas diferentes.

La función realizará las acciones comentadas con anterioridad a través de un objeto de la clase *PubSubClient* previamente configurado, el nombre que se le ha otorgado al objeto en cuestión es *ClienteESP_MQTT* y su configuración se expondrá en la parte final de esta sección.

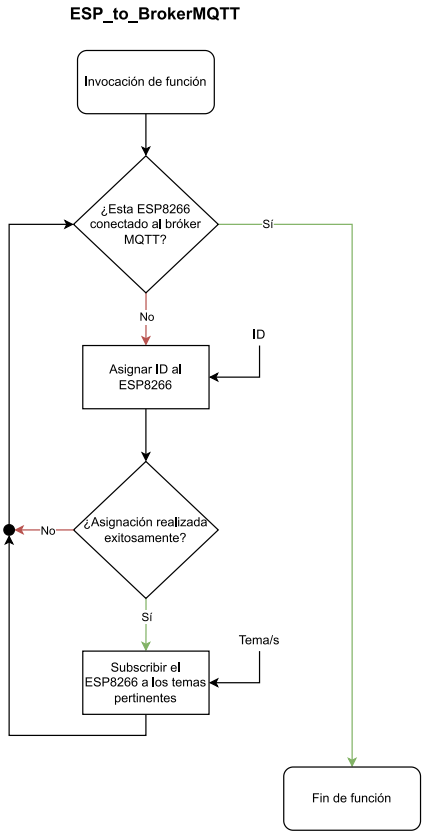


Ilustración 16: Diagrama de flujo de la función ESP_to_BrokerMQTT.

Los mensajes recibidos por el ESP8266 como consecuencia de su suscripción a un tema se encuentran en forma de agrupaciones de bytes. Será necesario transformar estos datos en una cadena de caracteres, para posteriormente decodificarlo en un objeto JSON del que poder extraer la información y almacenarla en una serie de variables globales. Este proceso se debe definir en una función *callback*, que servirá para alimentar el objeto *ClienteESP_MQTT* y garantizar una correcta gestión de los mensajes de entrada.

Dicha función se ha llamado *GestionaEntradaMensajes_Callback* y posee tres argumentos de entrada: El tema del que proviene el mensaje, la cadena de bytes anteriormente mencionada y la longitud de dicha cadena.

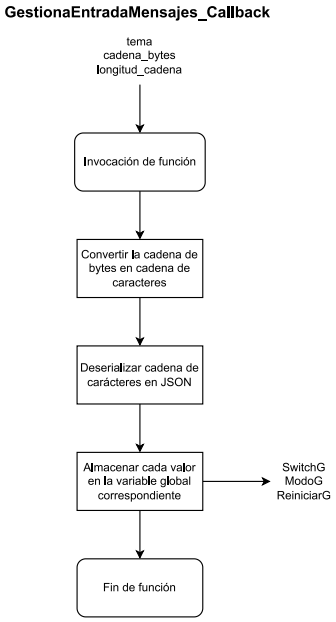


Ilustración 17: Diagrama de flujo de la función GestionaEntradaMensajes_Callback.

Al igual que ha sido necesaria una función para gestionar la recepción de mensajes, se precisará de otra que gestione la publicación de estos. En este caso se dispone de la información que se desea compartir en cinco variables diferentes. El primer paso, será crear un objeto JSON con cinco pares clave-valor, donde en cada uno se almacene la información de la variable correspondiente. A continuación, este objeto se serializará en un string que posteriormente se transformará en una cadena de caracteres lista para ser publicada. Dicha función se ha llamado *PublicaMensaje*.

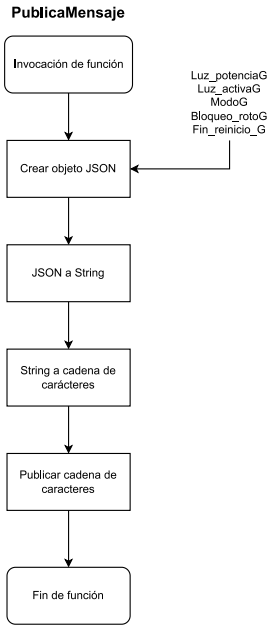


Ilustración 18: Diagrama de flujo de la función PublicaMensaje.

Finalmente se ha elaborado una función que permita inicializar todos los aspectos relativos a la conectividad tras ser invocada. En esta parte del código se configura el objeto *ClienteESP_MQTT*, introduciendo la dirección IP del bróker, puerto de escucha y el nombre de la función *callback* para la gestión de los mensajes recibidos. Además, se invocarán las funciones que permiten conectarse a la red WiFi y al bróker MQTT.

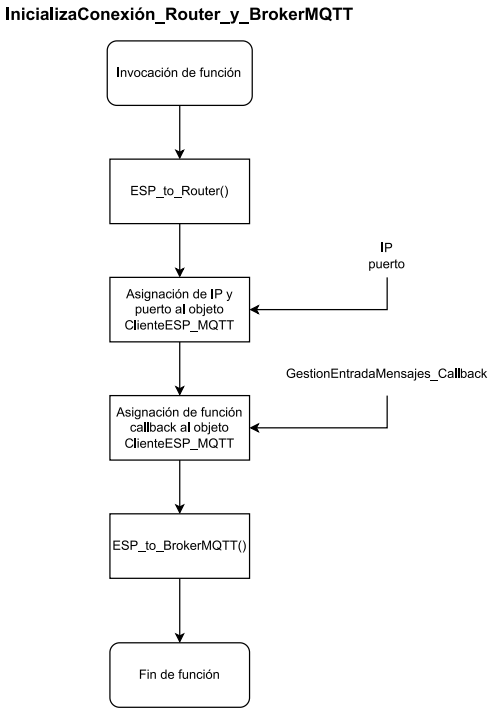


Ilustración 19: Diagrama de flujo de la función InicializaConexión_Router_y_BrokerMQTT.

4.2.3.2-Lámpara

En este módulo se recogen todas las funciones referentes a la configuración del ADS1115, comunicación con el ADS1115 y cálculo de la potencia consumida por la luminaria. Este módulo se apoya en tres librerías de Arduino:

- *Wire*: Facilita la implementación de la comunicación mediante protocolo I2C
- *Adafruit_ADS1X15*: Proporciona una API para interactuar con los convertidores analógico a digital (ADC) de la familia ADS1X15 de la empresa Adafruit. Requiere de la librería Wire, pues estos ADC se comunican mediante protocolo I2C con el microcontrolador.
- *Math*: Proporciona funciones matemáticas para realizar operaciones matemáticas más avanzadas que las operaciones básicas incluidas en el lenguaje C++ estándar. Se empleará para el cálculo de la corriente RMS que discurre por la lámpara.

Para interactuar con el ADS1115 se empleará una instancia de la clase *Adafruit_ADS1115*, definida en la librería *Adafruit_ADS1X15*. A este objeto se le ha dado el nombre *adc*, la clase a la que pertenece tiene definidos numerosos métodos, que actúan como API para configurar el conversor y leer la información que recoge.

En este módulo, la primera función que se ha definido se llama *InicializaADC*, permite configurar e iniciar el conversor. El único parámetro que se debe definir es la ganancia, la librería *Adafruit_ADS1X15* permite elegir entre seis ganancias diferentes. En este caso se empleará una ganancia unitaria.

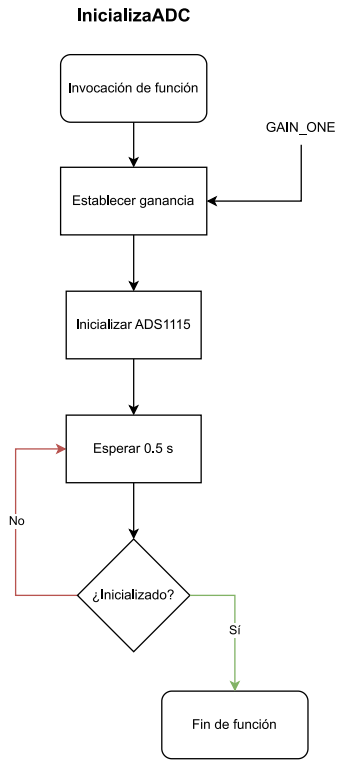


Ilustración 20: Diagrama de flujo de la función InicializaADC.

A continuación, se definió la función encargada de obtener la corriente instantánea que circula por la luminaria. Esta, de nombre *ObtenerCorrienteInst*, recogerá la tensión diferencial que mide el convertidor entre ambos terminales de la pinza. El ADC proporcionará un valor entre cero y dos elevado a la quince, que deberá ser multiplicado por la resolución de este para obtener dicha medida en unidades de tensión. Finalmente, el valor de tensión en Voltios se multiplicará por la sensibilidad del SCT-013, para conocer la corriente en Amperios que circula por la luminaria.

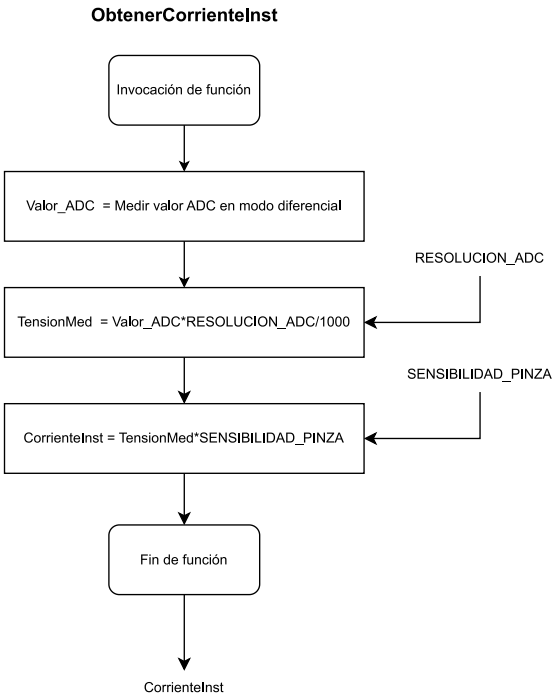


Ilustración 21: Diagrama de flujo de la función ObtenerCorrienteInst.

La finalidad última de este módulo es el cálculo de la potencia consumida por la lámpara, para ello no basta con obtener la corriente instantánea en un momento dado, se deberá conocer la corriente eficaz que circula por la luminaria. Esta se puede calcular a través de sucesivas muestras de corriente instantánea mediante la siguiente fórmula, siendo N el número de muestras.

$$\sqrt{\frac{1}{N} \times \sum_{i=1}^{N} {I_{inst,i}}^2}$$

Ecuación 4: Cálculo de corriente eficaz.

Conociendo esta posibilidad, se ha elaborado una función de nombre *ObtenerCorrienteRMS*. Esta se encargará de realizar el cálculo expuesto en la ecuación 4 con las muestras tomadas durante 200 ms.

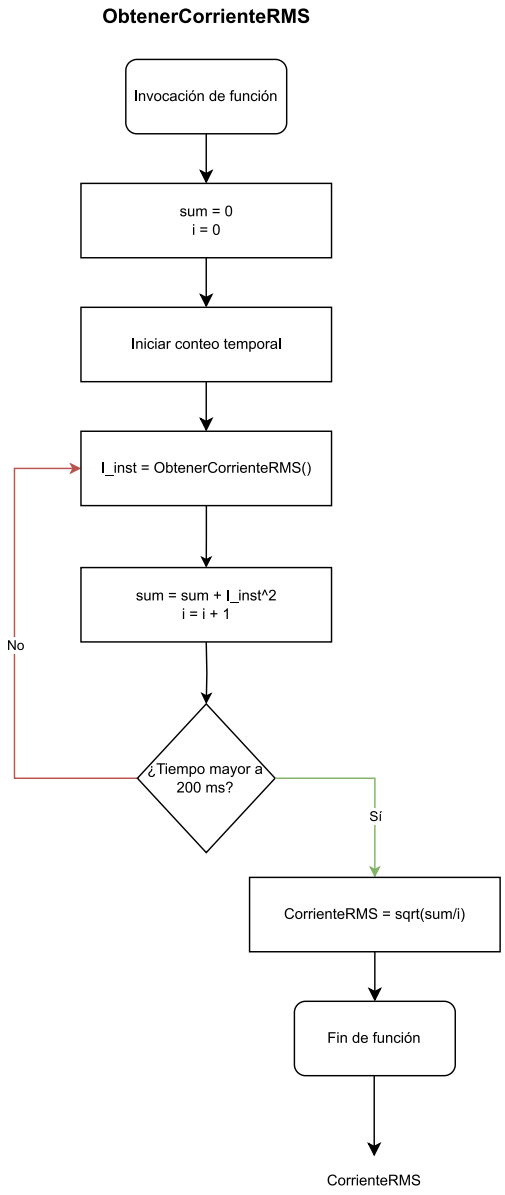


Ilustración 22: Diagrama de flujo de la función ObtenerCorrienteRMS.

Debido a la alta resolución que posee el ADC, pequeñas variaciones aleatorias de offset en el sensor de corriente pueden ser captadas por el convertidor externo, cuando la luminaria está apagada. Para ello se ha definido un umbral de corriente eficaz, por debajo del cual el dispositivo considerará que la lámpara está apagada. Tanto la implementación de este valor umbral, como la conversión de corriente eficaz a potencia se ha desarrollado en la función *GestionaPinza*.

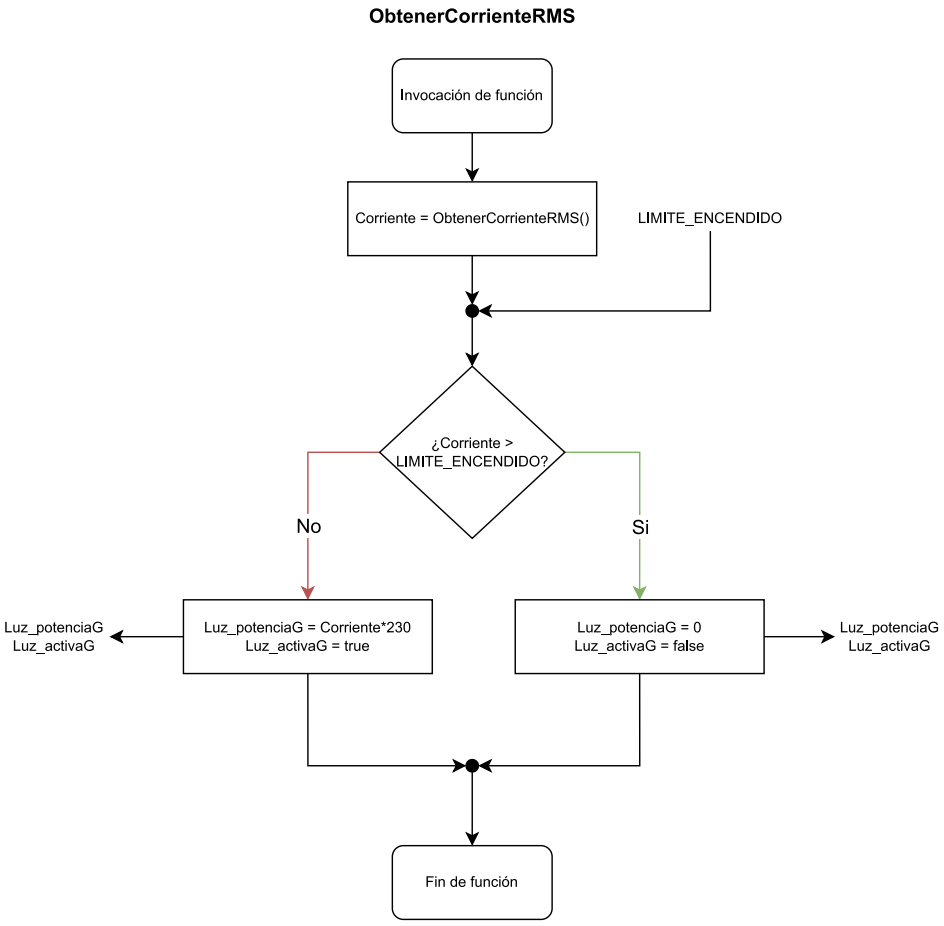


Ilustración 23: Diagrama de flujo de la función GestionaPinza.

4.2.3.3-Lógica

En este módulo se recogen todas las funciones que definen el comportamiento del dispositivo, en función de las indicaciones recibidas desde la aplicación y las mediciones realizadas sobre la luminaria, también se inicializará las variables necesarias que intervienen en estos procesos.

Este módulo empleará como base las funciones definidas en los módulos *Conexión_MQTT* y *Lampará*.

La primera función elaborada en este módulo, *Inicia_Logica*, permitirá inicializar las variables necesarias una vez el dispositivo sea encendido o reiniciado.

Primeramente, se llamará a la función *GestionaPinza*, asignando así valores iniciales a las variables encargadas de almacenar el estado y consumo de la lámpara (Luz_*activaG* y *Luz_potenciaG* respectivamente).

En segundo lugar, se definirán las variables necesarias para publicar el primer mensaje y seguidamente se publicará. Se desea que el dispositivo comience a funcionar en modo libre, esto implica que la variable encargada de disparar el aviso cuando la luminaria modifica su estado y el dispositivo está en modo bloqueo se dispondrá a nivel bajo.

A continuación, se leerá el último mensaje publicado por el dashboard. En el mensaje, el valor asociado a la clave Switch se almacenará en la variable *SwitchG*, el asociado a la clave Modo se guardará en la variable *ModoG* y por último el asociado a la clave Reiniciar será contenido en la variable *ReiniciarG*. En cuanto a la variable *ReinicarG*, independientemente del valor obtenido a través del mensaje se pondrá a nivel bajo durante la inicialización.

Seguidamente se otorgarán los valores iniciales a cuatro variables:

- *Luz_activaG_ant*: Indica el valor de la variable *Luz_activaG* en la iteración anterior, en su inicialización toma al valor de esta última.
- *SwitchG_ant*: Indica el valor de la variable *SwitchG* en la iteración anterior, en su inicialización toma al valor de esta última.
- *Cambio_luzG_ant*: Indica si en la iteración anterior *Luz_activaG_ant* y *Luz_activaG* tenían un valor diferente, en su inicialización toma valor bajo.
- *Cambio_switchG_ant*: Indica si en la iteración anterior *SwitchG_ant* y *SwitchG* tenían un valor diferente, en su inicialización toma valor bajo.

Finalmente, se define el valor para los dos disparadores, que se han empleado para indicar si es la primera iteración que realiza el controlador en modo bloqueo y cambiar el estado del relé (*Flag_primer_bloqueoG* y *FlagReleG* respectivamente).

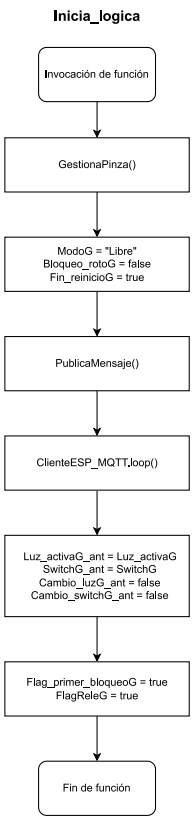


Ilustración 24: Diagrama de flujo de la función Inicia_Logica.

La segunda función elaborada, *Set_Flags_ModoLibre*, pondrá la variable disparadora del cambio de estado del relé a nivel alto o bajo cuando el dispositivo está trabajando en modo libre, la variable en cuestión se denomina *FlagReleG*.

Si en la iteración anterior el dispositivo se encontraba en modo bloqueo y ha trabajado más de un ciclo en este estado, la variable *Flag_primer_bloqueoG* tendrá un nivel bajo. Por esta razón, la primera operación será comprobar si esta condición es cierta, en caso afirmativo la variable en cuestión se dispondrá a nivel alto, además, al existir un cambio de modo a *FlagReleG* se le asignará un valor bajo.

Para que el relé modifique su estado, el usuario debe haber pulsado el botón correspondiente en el dashboard, dicho de otra manera, *FlagReleG* tomará valor alto cuando *SwitchG_ant* y *SwitchG* sean diferentes.

Una vez el disparador del relé esta activo, se debe estIPular una condición para su desactivación. En este caso, *FlagReleG* volverá a tomar valor bajo cuando se encuentre en estado alto y además la luminaria haya modificado su estado, esta última condición se traduce como *Luz_activaG_ant* diferente de *Luz_activaG*.

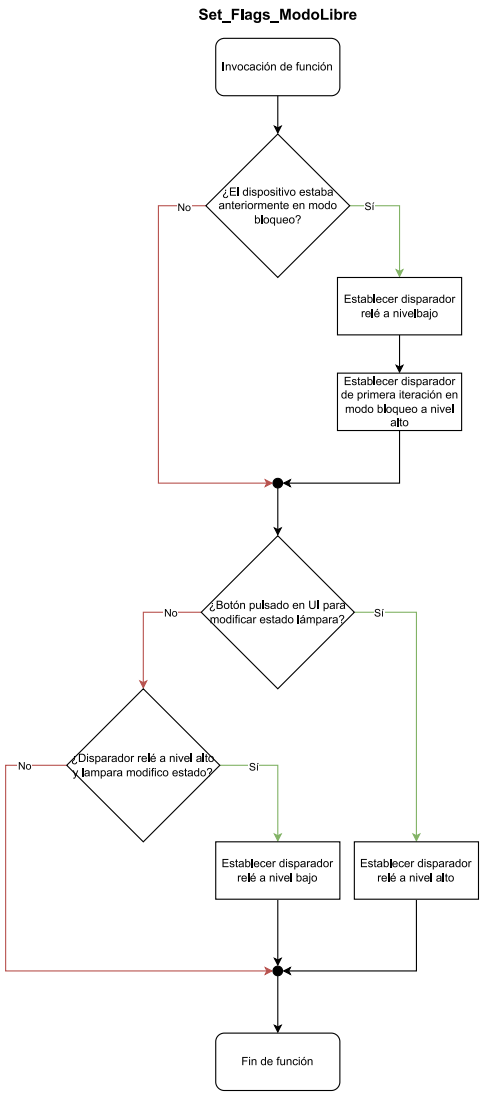


Ilustración 25: Diagrama de flujo de la función Set_Flags_ModoLibre.

Seguidamente, se procedió a desarrollar una función homóloga a la anterior para el modo bloqueo, *Set_Flags_ModoBloqueo*. Si es la primera iteración del dispositivo en modo bloqueo el disparador del relé se establecerá a nivel bajo y la variable *Flag_primer_bloqueoG* también.

En esta función, se deberá conocer si el estado de la lámpara ha sido modificado mediante un interruptor de pared, con la finalidad de avisar al usuario a través del dashboard, si esto sucede se modificará el valor de una variable booleana de nombre *Bloqueo_rotoG*. Dicha condición se cumple cuando los siguientes tres puntos ocurren simultáneamente:

- El estado de la luminaria en el presente ciclo es diferente respecto al anterior.
- El estado de la luminaria no cambió en los dos ciclos anteriores.
- El valor de la clave Switch del mensaje publicado en el bróker MQTT no cambió en los dos ciclos anteriores.

Adicionalmente, se deberán establecer las condiciones para poner a nivel alto la variable disparadora del relé. El relé se deberá activar en dos casuísticas:

- Cuando el estado de la lámpara se modifique desde el interruptor de pared. Esto sucede al cumplirse simultáneamente las tres condiciones mentadas en el párrafo anterior, dicho de otra manera, la variable *Bloqueo_rotoG* está en nivel alto.
- Cuando el usuario accione el botón para modificar el estado de la luminaria en la UI. Esto sucede al modificar el valor asociado a la clave Switch del mensaje publicado por el dashboard entre la iteración actual y la anterior.

Una vez el disparador del relé esta activo, se debe estIPular una condición para su desactivación. En este caso, *FlagReleG* volverá a tomar valor bajo cuando se encuentre en estado alto, la luminaria no haya modificado su estado desde la iteración anterior y se cumplan una de las siguientes dos condiciones:

- El estado de la luminaria cambió en los dos ciclos anteriores.
- El valor de la clave Switch del mensaje publicado en el bróker MQTT cambió en los dos ciclos anteriores.

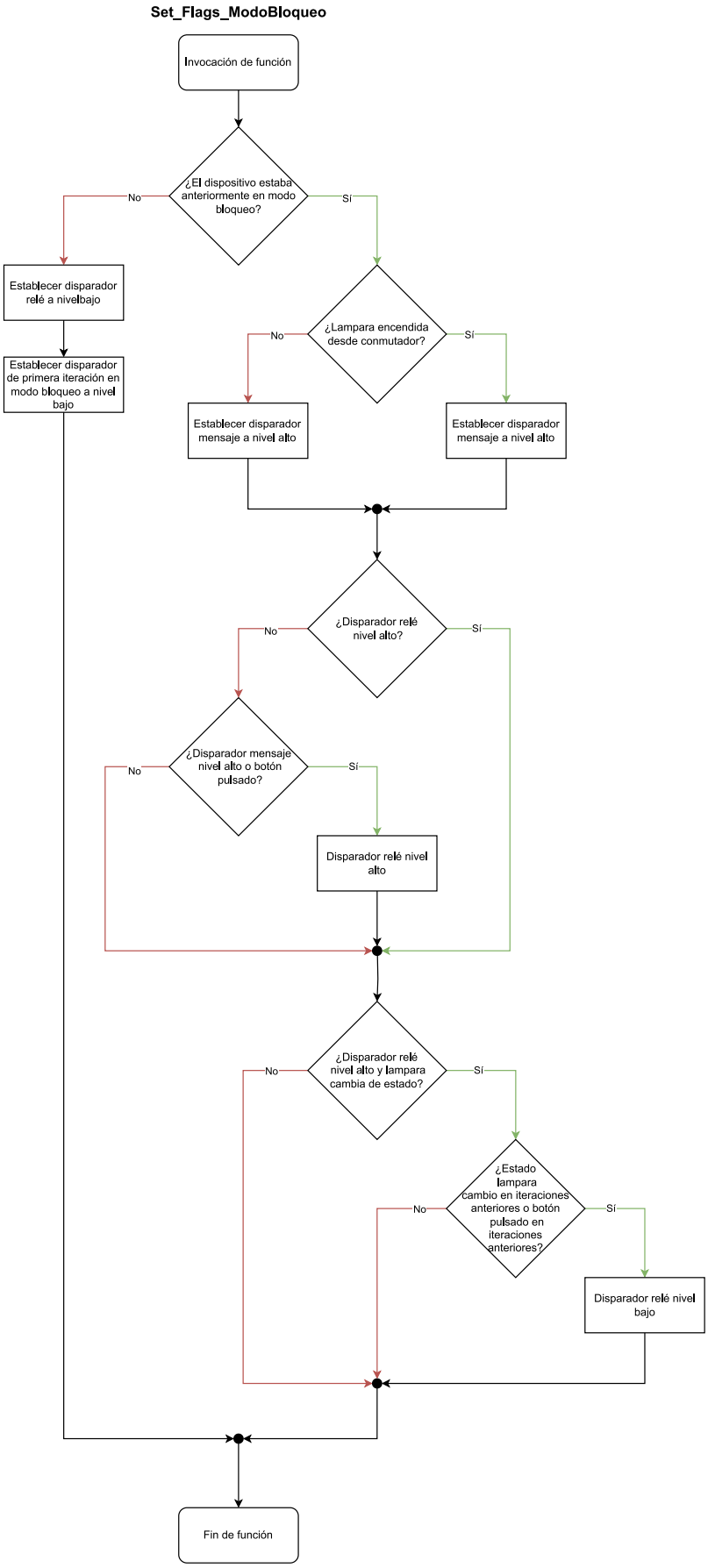


Ilustración 26: Diagrama de flujo de la función Set_Flags_ModoBloqueo.

Finalmente, se desarrolló la función que modificará el estado del relé cuando el disparador asociado se disponga a nivel alto, *Control_Rele*. Primero, se debe conocer si el relé esta activo o inactivo, para ello se realizará una lectura de la tensión existente en el pin A0 de la tarjeta NodeMCU, pues está se encuentra conectada a la tensión de entrada que recibe el relé. El conversor ADC interno del ESP8266 es de 10 bits, si el relé esta alimentado se obtendrá un valor en la medición cercano a 1024, si por el contrario si no lo está, aportará una medición cercana a 0.

Cuando el disparador del relé se encuentre en estado alto y la medición este por encima de 500 se cortará la tensión de entrada a este, si por el contrario la medición está por debajo de 500 se establecerá una tensión de entrada al relé de 3.3 V.

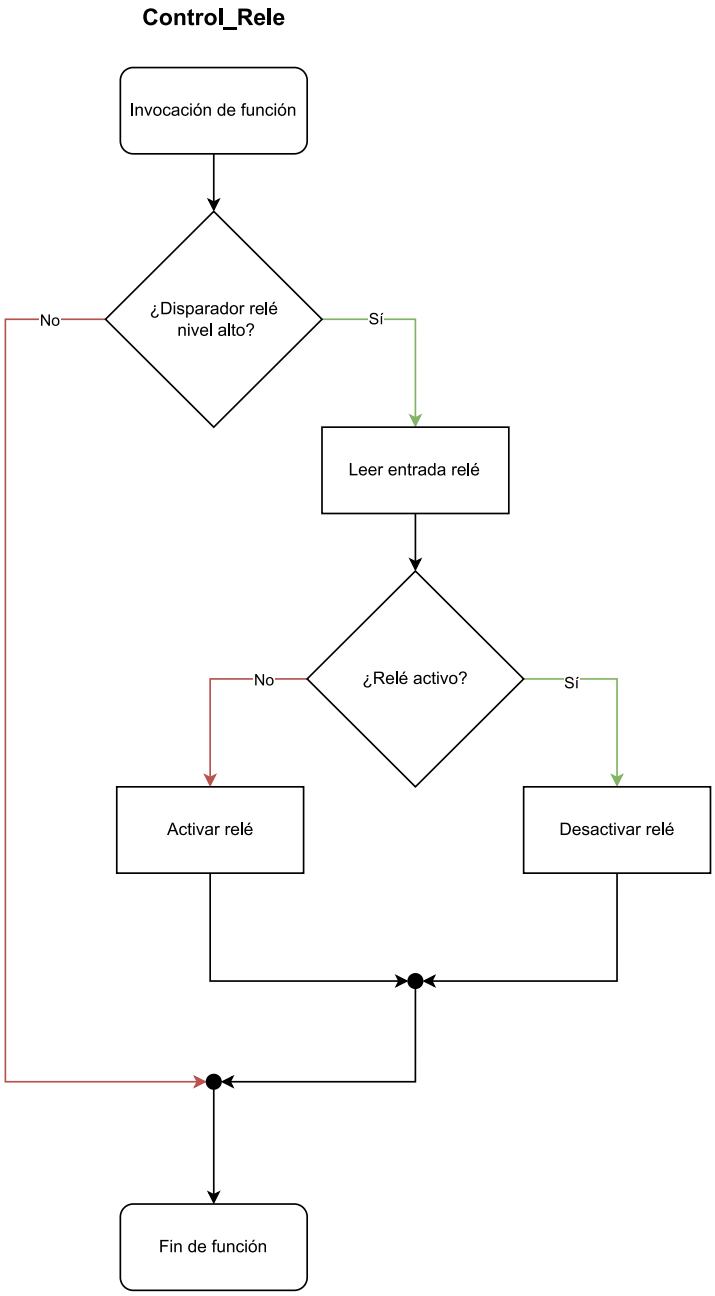


Ilustración 27: Diagrama de flujo de la función Control_Rele.

4.2.3.4-Módulo principal

En este módulo es donde se definirán todas las variables globales y objetos empleadas en el resto de los módulos y funciones. En este existen dos funciones:

- *setup*: Se ejecuta al encender por primera vez el dispositivo.
- *loop*: Se ejecuta periódicamente, cuando termina de ejecutarse vuelve a iniciarse.

En la función setup, se definirán los pines del ESP8266 que se van a emplear como entrada y como salida, se inicializará el ADC, se conectará el ESP8266 al rúter, se conectará el ESP8266 al bróker MQTT e inicializará el valor de las variables.

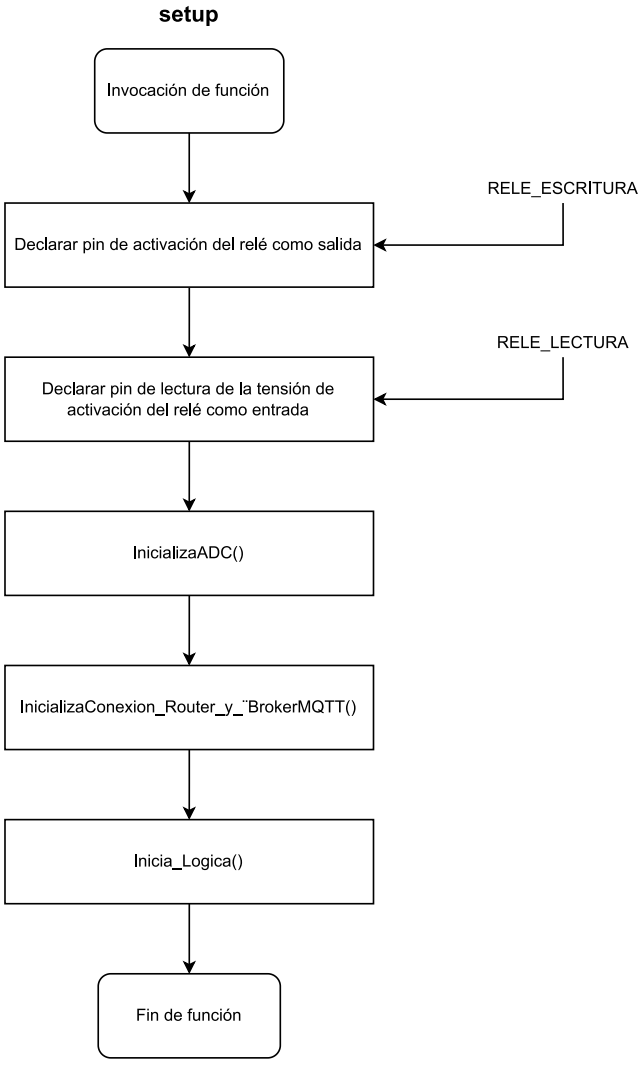


Ilustración 28: Diagrama de flujo de la función setup.

En la función loop lo primero que se comprobará es si el ESP8266 está conectado al bróker MQTT, en caso negativo se intentará reestablecer la conexión e inicializar las variables nuevamente.

En caso de que el dispositivo continue conectado al bróker MQTT, se procederá a comprobar si desde el dashboard se ha dado la orden de reiniciar el dispositivo, en cuyo caso se invocará de nuevo a la función *setup*. Si por el contrario no se dio la orden de reiniciar, se leerá el estado y potencia consumida por la lámpara.

Posteriormente se invocará la función *Set_Flags_ModoLibre* o *Set_Flags_ModoBloqueo* en función del modo de trabajo del dispositivo, se cambiará el estado del relé si es preciso y se actualizarán los valores de las variables que almacenan datos asociados a iteraciones anteriores.

Finalmente, se leerá el mensaje publicado por el dashboard y se actualizarán las variables correspondientes, en caso de haberse pulsado el botón de reiniciar se actualizará el estado para la variable que indica el fin de este. Restará únicamente la publicación de un nuevo mensaje por parte del dispositivo, que contendrá los datos actualizados en lo que a la presente iteración se refiere.

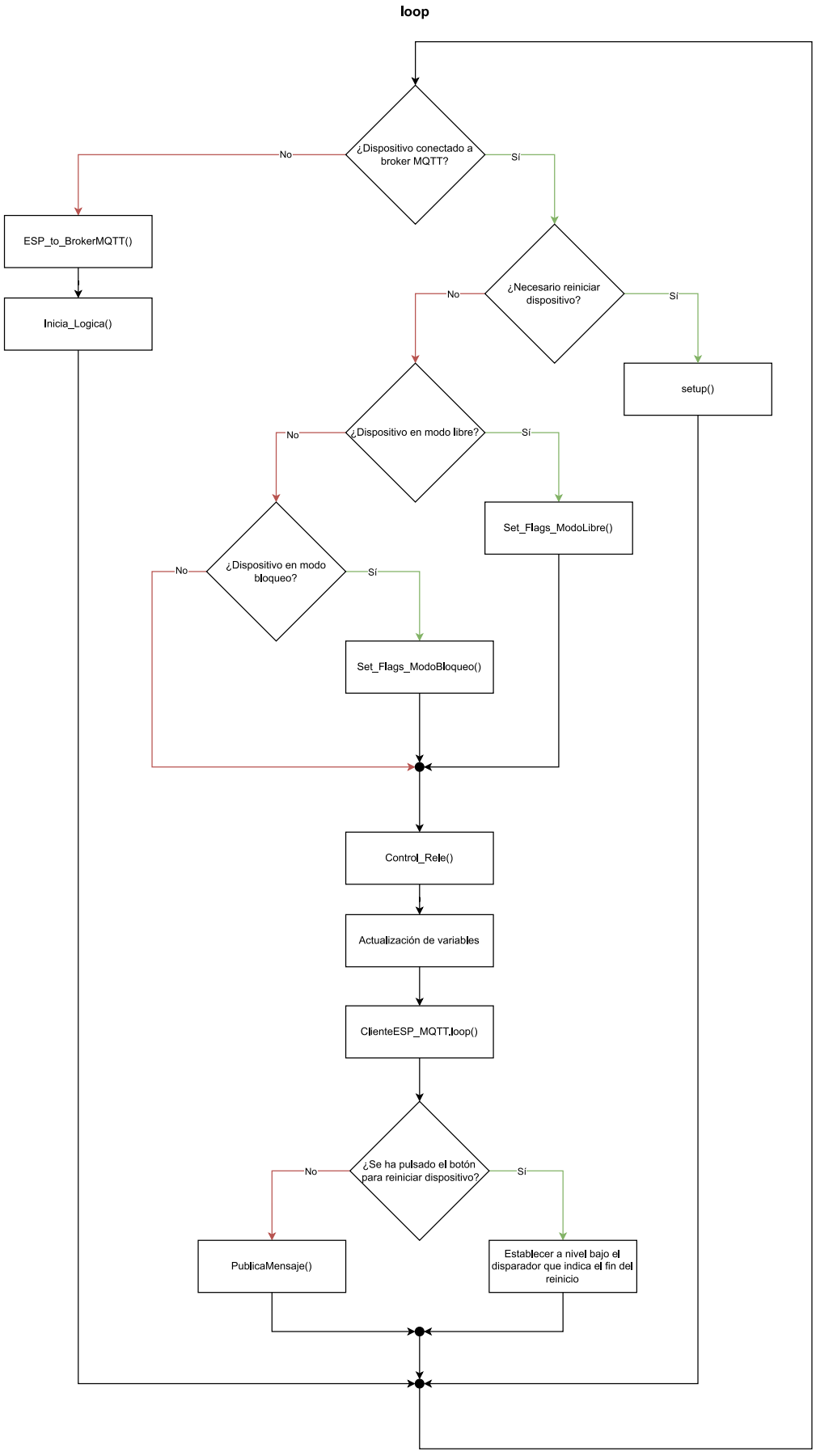


Ilustración 29: Diagrama de flujo de la función loop.

4.2.4-Desarrollo UI y APIs

Antes de profundizar en cómo se ha empleado esta herramienta, será necesario definirla y explicar su funcionamiento básico. La propia organización Node-Red proporciona una definición de su propia herramienta:

"*Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.*

It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click." [Node-RED es una herramienta de programación para conectar dispositivos de hardware, API y servicios en línea de formas nuevas e interesantes.

Proporciona un editor basado en navegador que facilita la conexión de flujos utilizando la amplia gama de nodos en la paleta que se pueden implementar en su tiempo de ejecución con un solo clic.]

(Node-Red, 2024)

Esta herramienta, emplea como base unos elementos denominados nodos, que deben interconectarse entre sí. Existen muchos tipos de nodos, especializados en funciones concretas, estos permiten realizar diferentes acciones como: generar mensajes, modificar mensajes, realizar consultas a bases de datos, mostrar valores a través de un dashboard, etc.

Las interconexiones entre nodos permiten transmitir la salida de unos nodos a las entradas de los siguientes. Un nodo se activará cuando reciba en su entrada un nuevo mensaje. El mensaje se manda de un nodo a otro en formato JSON, este formato para la transmisión de información ya ha sido definido en el punto 4.2.2-Transmisión de mensajes.

Tal y como se comentó con anterioridad node-red se empleó como soporte para elaborar un dashboard con el que monitorizar y actuar sobre el dispositivo. Para que el dashboard cumpla con las funciones definidas anteriormente, no bastará solo con disponer los elementos visuales con los que interactuará el usuario, serán necesarios una serie de procesos transparentes al usuario:

- Recepción y tratamiento inicial del mensaje: Se deberá recoger cada mensaje nuevo publicado por el dispositivo para ser dividido posteriormente en cinco buses diferentes, uno por elemento del JSON.
- Tratamiento del mensaje para monitorización y visualización de la información: De los cinco buses en los que se ha dividido el mensaje inicial, cuatro de ellos contienen información que debe ser visualizada por el cliente.
- Tratamiento del mensaje para comunicación con el dispositivo IoT y publicación: De los cinco buses en los que se ha dividido el mensaje inicial, dos de ellos contienen información que al ser procesada permitirán que el cliente controle el dispositivo. Para alcanzar este objetivo, será necesario también, disponer actuadores en el dashboard que el usuario pueda manipular.
- Gestión de la BBDD y representación gráfica de los datos: Este proceso, permitirá realizar las consultas de escritura y lectura sobre la BBDD mediante lenguaje SQL.

También dispondrá de los actuadores necesarios en el dashboard para representar dicha información de manera gráfica.

A continuación, se adjunta un esquema de conjunto donde se integran todos los elementos del presente trabajo.

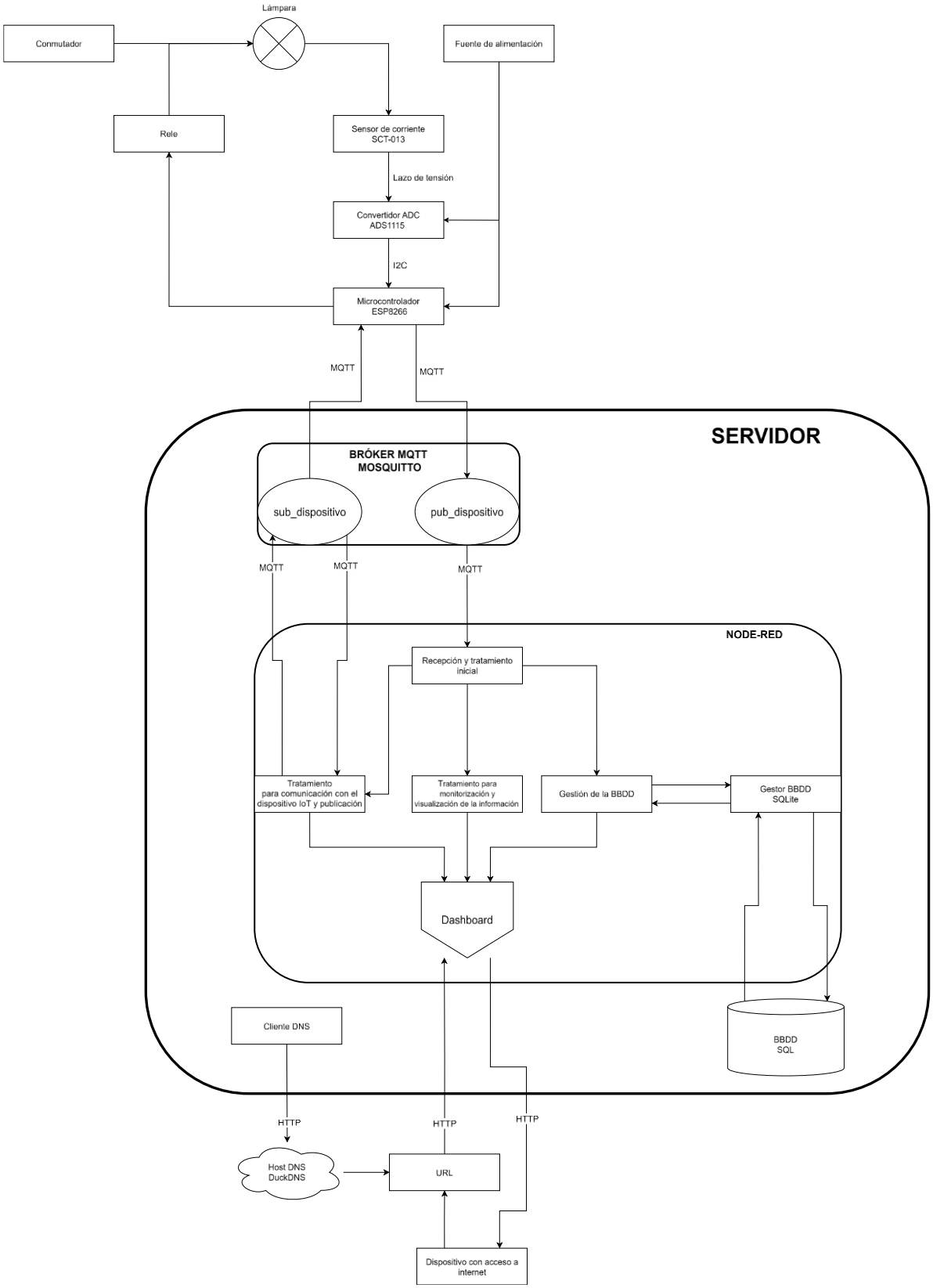


Ilustración 30: Esquema de conjunto del presente trabajo.

4.2.4.1-Interfaz de usuario

Antes de comenzar a programar las APIs y back-end del dashboard se definió la UI con la que interactuará el usuario final.

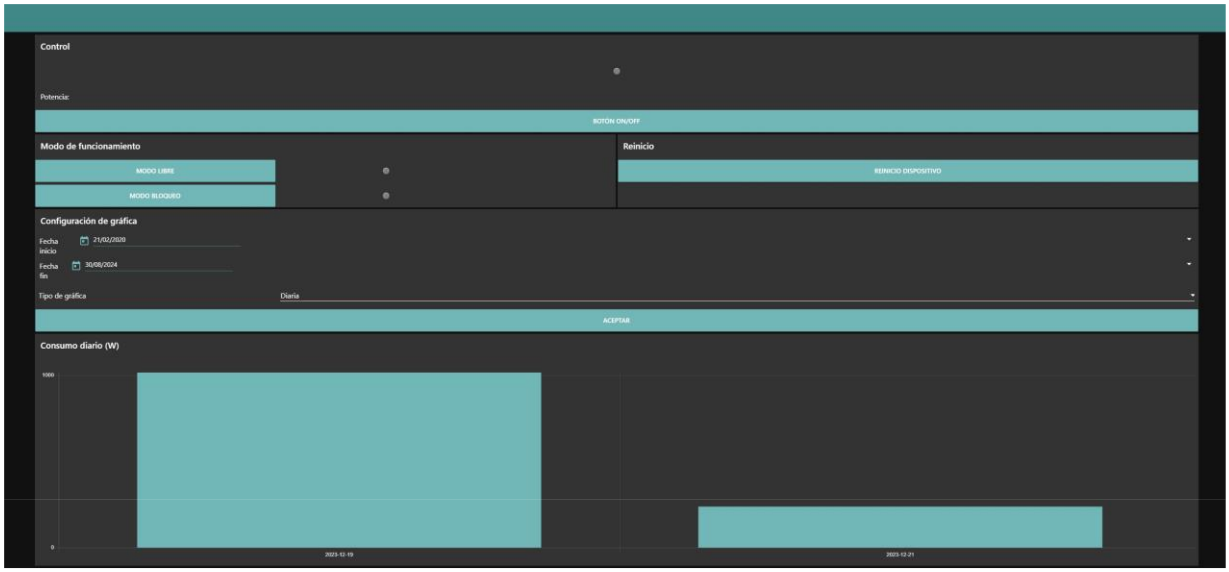


Ilustración 31: Estructura del dashboard.

La interfaz de usuario posee seis partes bien diferenciadas. La sección de control permitirá monitorizar si la lámpara esta encendida o apagada, así como conocer la potencia que está consumiendo, también otorgará la capacidad de apagar o encender la luminaria a través de un botón.

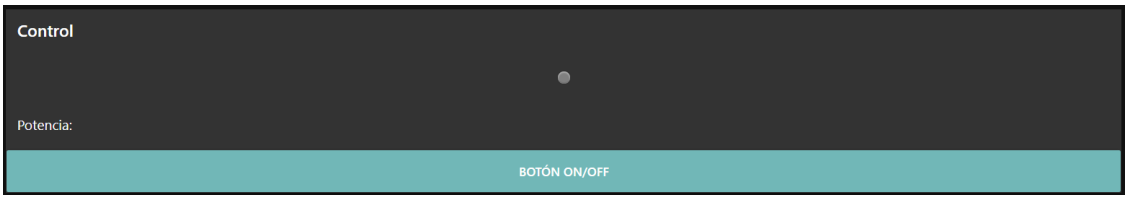


Ilustración 32: Sección Control del Dashboard.

El sector “Modo de funcionamiento” permitirá establecer si se desea que la luminaria actúe en modo libre o bloqueo. El indicador asociado al modo en el que se encuentra el dispositivo estará iluminado en color verde, mientras el otro se mantiene en color gris.

Ilustración 33: Sección Modo de funcionamiento del Dashboard.

Inmediatamente a la derecha, se encuentra el botón que permite reiniciar el dispositivo. Mientras este se está reiniciando, un texto lo indicara justo debajo del botón.

Ilustración 34: Sección Reinicio del Dashboard.

Finalmente, la parte "Configuración de gráfica" permitirá mostrar un gráfico de la energía consumida una vez pulsado el botón "Aceptar". La cantidad de tiempo que se mostrará en el gráfico vendrá determinada por las fechas de inicio y fin establecidas. Además, se podrá mostrar la energía consumida agrupada de tres formas diferentes: horaria, diaria, mensual y anual.

Ilustración 35: Sección Configuración de gráfica del Dashboard.

4.2.4.2-Recepción y tratamiento inicial del mensaje

En esta parte de las rutinas de node-red se busca subscribirse al tema donde está publicando el dispositivo, así como recibir el mensaje en formato JSON y dividirlo en cinco mensajes diferentes, uno por clave.

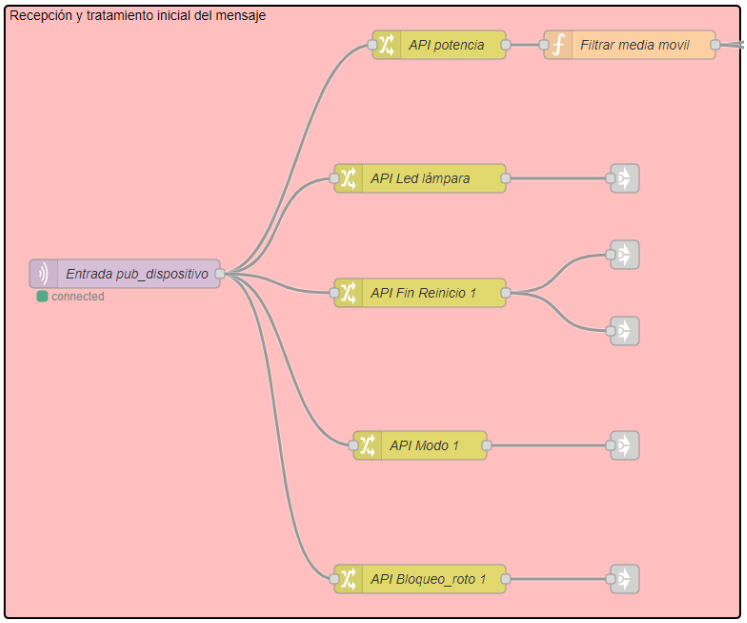


Ilustración 36: Flujo para la recepción y tratamiento inicial del mensaje.

Como se puede apreciar, se ha dividido el mensaje inicial en cinco buses diferentes, que de arriba abajo son: Potencia, Led, Fin_reinicio, Modo y Bloqueo_roto. Finalmente, los diferentes sub-mensajes pasan a un tipo de nodo que los distribuirán al resto de flujos. Se observa también, como el valor de la potencia pasa a través de una función que actúa como filtro de media móvil, tanto el concepto detrás de este filtro como la razón de ser será explicado en la sección 4.3.1-Primera iteración.

4.2.4.3-Tratamiento del mensaje para monitorización y visualización de la información

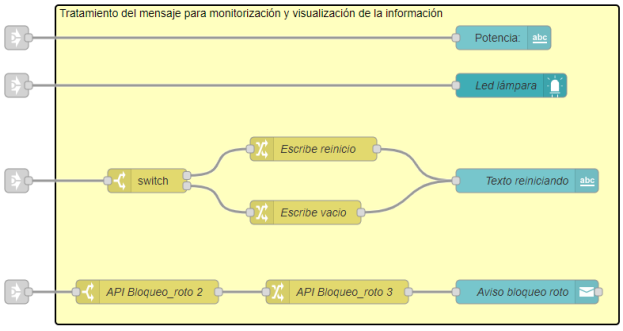


Ilustración 37: Flujo para el tratamiento del mensaje para monitorización y visualización de la información.

Este proceso permitirá mostrar en la UI ciertos estados información del dispositivo. Por un lado, la potencia y el estado de la lámpara se podrán mostrar directamente sin necesidad de aplicar tratamiento alguno sobre los sub-mensajes del punto 4.2.4.2-Recepción y tratamiento inicial del mensaje.

Por otro lado, el sub-mensaje Fin_reinicio necesita de un nodo *switch* que distinga si su valor es alto o bajo y escriba nada o "Reiniciando…" respectivamente. Por último, el cuarto flujo solo se activará cuando el sub-mensaje Bloqueo_roto tenga un valor alto, en cuyo caso se mostrará una notificación en la UI.

4.2.4.4-Tratamiento del mensaje para comunicación con el dispositivo IoT y publicación

Este flujo realiza los procesos necesarios para preparar el mensaje que se publicará en el tema pub_dispositivo, además de encargarse de la publicación de este. Como se puede apreciar en la imagen inferior está constituido por cuatro subprocesos.

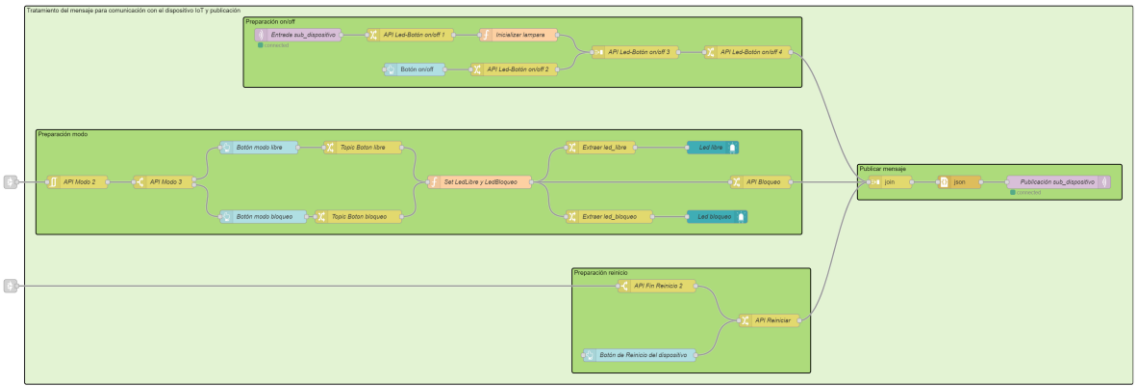


Ilustración 38: Flujo para tratamiento del mensaje para comunicación con el dispositivo IoT y publicación.

Tal como se comentó en el apartado 4.3.4, el dispositivo sabrá que se ha emitido la orden de cambiar el estado de la lámpara desde el dashboard cuando el valor de la clave *Switch* del mensaje publicado en el tema *sub_dispositivo* cambie de alto a bajo o viceversa. El subproceso *Preparación on/off*, se encarga de decidir si el valor asociado a la clave *Switch* debe ser alto o bajo una vez el usuario acciona el botón.

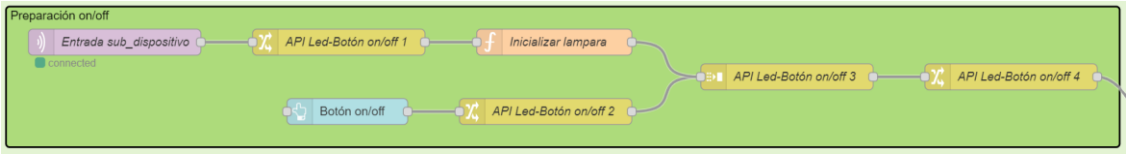


Ilustración 39: Flujo del subproceso Preparación on/off.

Para conseguir esto se debe leer el mensaje publicado en el tema en cuestión, así como preparar una función que establezca el valor de inicialización cuando aún no se ha publicado mensaje alguno. Mediante el nodo *API Led-Botón on/off 3* se coordina la pulsación del botón con la lectura del mensaje publicado en el tema *sub_dispositivo*. Finalmente, el ultimo nodo niega el valor actual de la clave *Switch*, aportando así el valor que esta deberá de contener el siguiente mensaje publicado.

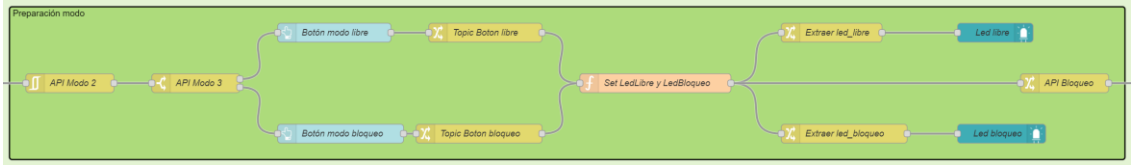


Ilustración 40: Flujo del subproceso Preparación modo.

A continuación, se desarrolló el flujo *Preparación modo*, este cumple con tres funciones. La primera es garantizar una correcta indicación del modo de funcionamiento cuando el dispositivo se encuentre recién iniciado. Para ello será necesario leer del tema *pub_dispositivo* el modo en el que se encuentra el dispositivo. A continuación, se hace pasar esta información por un filtro, de forma que el flujo solo se active ante cambios en el modo de trabajo del dispositivo. Finalmente, se dispone un nodo *Switch* que dividirá el flujo en función de su mensaje.

La segunda función es garantizar una correcta iluminación de los indicadores de modo de trabajo ante una modificación de estos por parte del usuario. Para ello se ha desarrollado la función *Set LedLibre y LedBloqueo*, en función del botón que se haya pulsado se generarán dos posibles mensajes. Dicha información pasará por los nodos *Extraer led_libre* y *Extraer led_bloqueo* que encenderán o apagarán sus indicadores asociados.

Modo libre

```
{
  "led_libre": true,
  "led_bloqueo": false,
  "modo": "Libre"
}
```

Modo bloqueo

```
{
  "led_libre": false,
  "led_bloqueo": true,
  "modo": "Bloqueo"
}
```

Ilustración 41: Posibles mensajes de salida de la función Set LedLibre y LedBloqueo.

La tercera función es establecer el valor asociado a la clave *Modo* en el próximo mensaje a publicar, esta información es extraída de la salida de la función Set *LedLibre y LedBloqueo* por el nodo *API Bloqueo*.

El siguiente subproceso que se desarrolló recibe el nombre de Preparación *reinicio*. Este flujo se encargará de establecer el valor asociado a la clave Reiniciar, tomará valor alto cuando se pulse el botón de reinicio y valor bajo cuando el dispositivo comience a reiniciar.

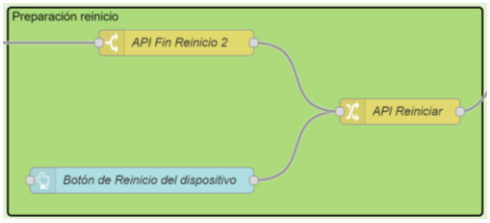


Ilustración 42: Flujo del subproceso Preparación reinicio.

Finalmente, se deben recolectar los valores asignados en los diferentes subprocesos, unificarlos en un mensaje con estructura JSON y publicarlo en el tema sub_dispositivo. De esto se encarga el subproceso *Publicar mensaje*.

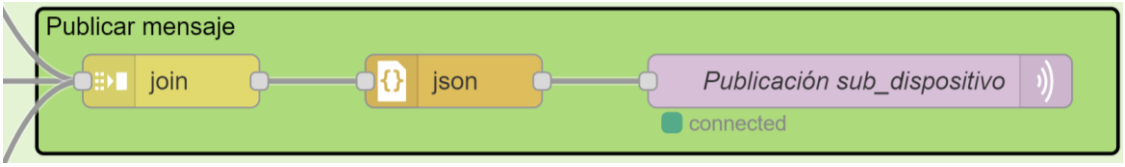


Ilustración 43: Flujo del subproceso Publicar mensaje.

4.2.4.5-Gestión de la BBDD y representación gráfica de los datos

Este flujo realiza los procesos necesarios para escribir en la BBDD, leer de la BBDD y representar la información leída de manera gráfica, para ello se apoya en cinco subprocesos.

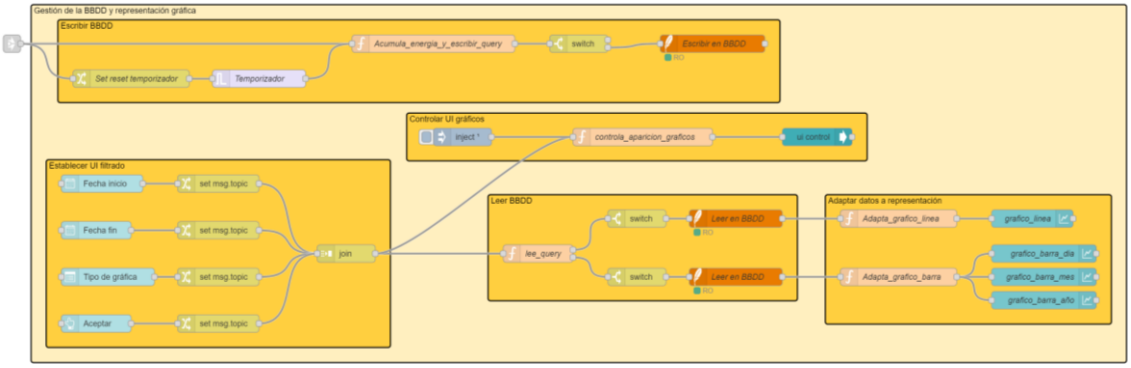


Ilustración 44: Flujo para la gestión de la BBDD y representación gráfica de los datos.

El subproceso *Escribir BBDD* permite realizar la escritura en la BBDD de la energía producida cada cinco minutos. Como entrada a este flujo, se tiene la potencia instantánea consumida por la luminaria. Cada vez que se recibe un nuevo mensaje de potencia, la función *Acumula_energia_y_escribir_query* calcula la energía consumida en el periodo anterior, a través de la variación de tiempo entre recepción de mensajes. Este valor de energía se acumula en una variable globa, a través de la suma con el valor anterior de la misma. Cada cinco minutos el valor de la variable global se almacena en la BBDD junto con el *timestamp* del instante en que se realiza la escritura, tras la escritura se le asigna a la variable global el valor cero para continuar con un nuevo ciclo de acumulación. Adicionalmente, si no se enviasen mensajes nuevos durante cinco minutos consecutivos, se entenderá que existe algún fallo y el valor escrito en la BBDD será cero.

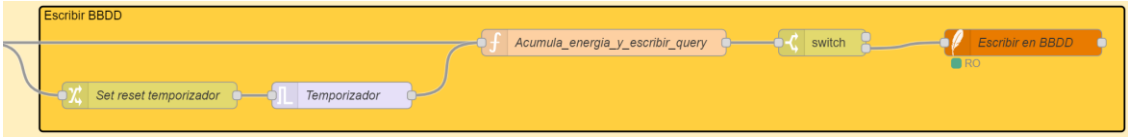


Ilustración 45: Flujo del subproceso Escribir BBDD.

La UI tendrá cuatro elementos asociados a la representación gráfica de información. Dos seleccionadores de fechas, uno para el inicio y otro para el fin de la representación de datos. Un desplegable, que permitirá establecer si se desea ver una representación horaria, diaria, mensual o anual. Un botón de aceptar, para hacer efectivo las elecciones realizadas.

Sera necesario un subproceso que recoja los mensajes generados por los elementos anteriores y los combine en uno solo, este servirá de entrada para otros flujos de este mismo proceso. El subproceso encargado de esta tarea se ha denominado *Establecer UI filtrado*.

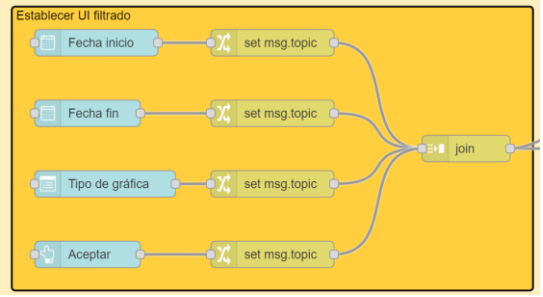


Ilustración 46: Flujo del subproceso Establecer UI filtrado.

El subproceso *Leer BBDD* se encargará de leer los datos de la BBDD que entren dentro del rango seleccionado, así como realizar las operaciones de agregación y agrupaciones necesarias.

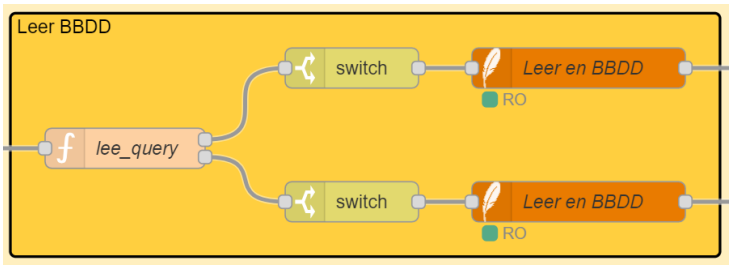


Ilustración 47: Flujo del subproceso Leer BBDD.

La función *leer_query* podrá realizar cuatro consultas diferentes para un mismo rango temporal, la consulta realizada dependerá de la opción escogida por el usuario en el desplegable.

Si el usuario decide ver los datos de manera horario se empleará un gráfico de líneas, mientras que para el resto de visualización se empleará el de barras. La salida superior de la función *leer_query* se empleará para obtener el resultado de la consulta que alimentarán el grafico de líneas, la inferior realizará una función homóloga para los gráficos de barras.

Los datos obtenidos tras realizar la consulta a la BBDD necesitan de una función que modifique su formato, para poder ser representado adecuadamente por los nodos *Chart* de node-red. La función *Adapta_grafico_linea* realizará este proceso para el gráfico de líneas, mientras que *Adapta_grafico_barra* se encargará de este proceso para el gráfico de barras. Una vez los datos han sido adaptados queda inyectar el mensaje a los nodos *Chart.* Estas operaciones descritas se desarrollarán en el subproceso *Adapta datos a representación*.

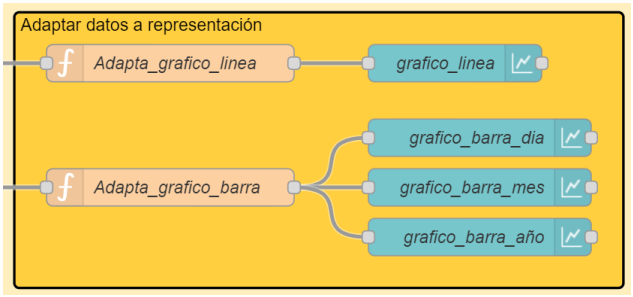


Ilustración 48: Flujo del subproceso Adapta datos a representación.

El usuario solo deberá visualizar un único gráfico, aquel asociado a la opción del desplegable escogida, este control de flujos en la UI se ha desarrollado en el subproceso *Control UI gráficos*. Se empleará la salida del subproceso *Establecer UI filtrado* para ocultar y mostrar las gráficas adecuadas.

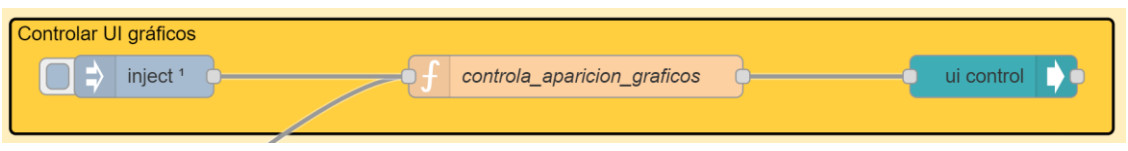


Ilustración 49: Flujo del subproceso Control UI gráficos.

4.3-Metodología seguida para el desarrollo

Las soluciones comentadas en el apartado anterior han seguido un proceso de desarrollo iterativo, con el fin de minimizar errores y aumentar al máximo su trazabilidad. En total, el proceso de implementación ha consistido en cinco iteraciones. Para validar el resultado de cada iteración y del prototipo final, se emplearán tres luminarias diferentes, conectadas a un conmutador:

- Bombilla led de potencia nominal 13 W.
- Bombilla incandescente de potencia nominal 40 W.
- Bombilla incandescente de 75 W.

4.3.1- Primera iteración

En la primera iteración se ha empleado una placa de pruebas donde se han dispuesto: el microcontrolador ESP8266, el conversor ADS1115 y el sensor de corriente SCT-013, conectados a la luminaria como se indica en el apartado 4.2.1-Hardware. En esta iteración fue donde se desarrolló el módulo de funciones de nombre Lámpara.

En primera instancia, se deseaba conocer como de precisos eran los cálculos de la potencia realizados por el microcontrolador. Para ello, los valores de potencia calculados a partir de la medición del sensor SCT-013 con el posterior muestreo del convertidor ADS1115 han sido impresos en el monitor serie y recogidos en diferentes ficheros de texto. Para validar esta medición, se ha empleado un multímetro con pinza amperimétrica, dispuesto junto al sensor de corriente y envolviendo el mismo conductor.

Los valores de corriente eficaz medidos por el multímetro se han multiplicado por la tensión de la red (230 V en España), traduciendo así esta medida de corriente a potencia.

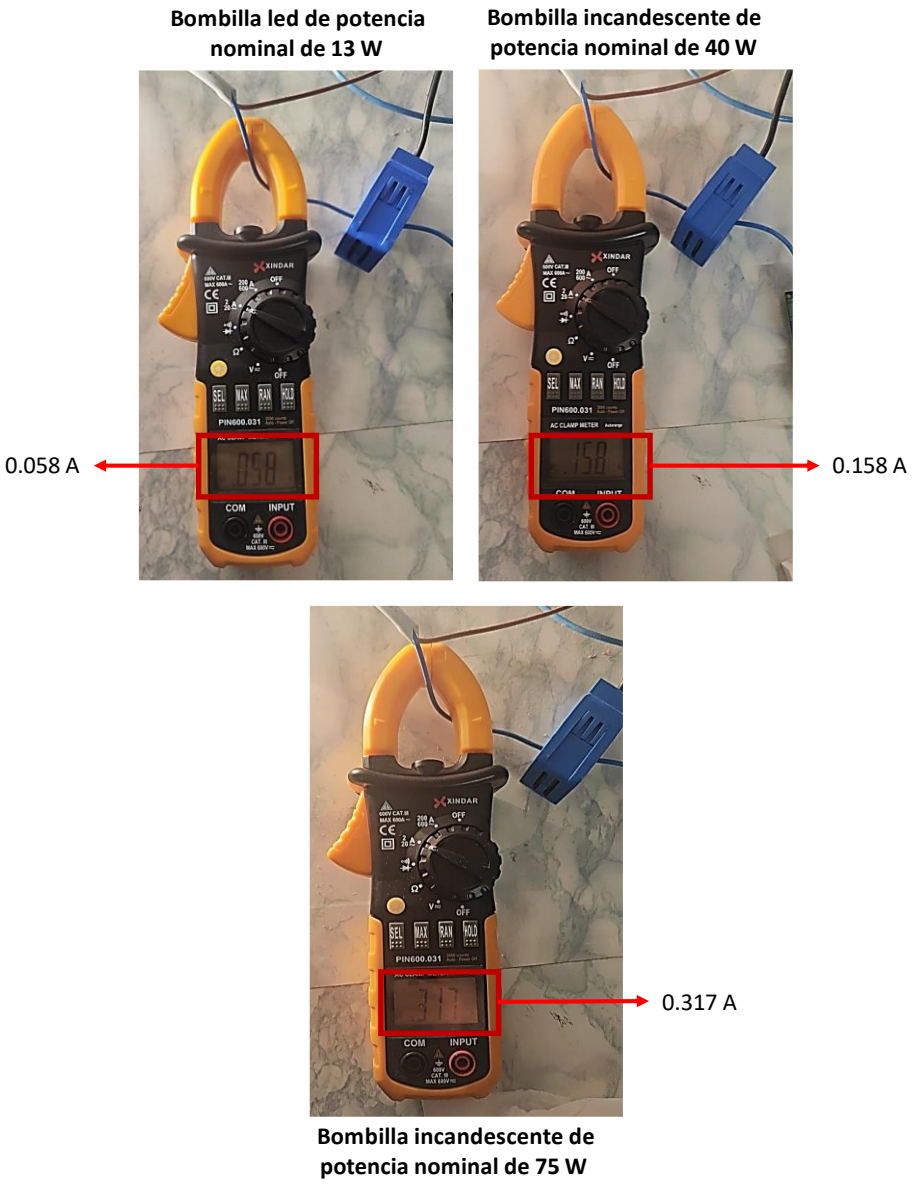
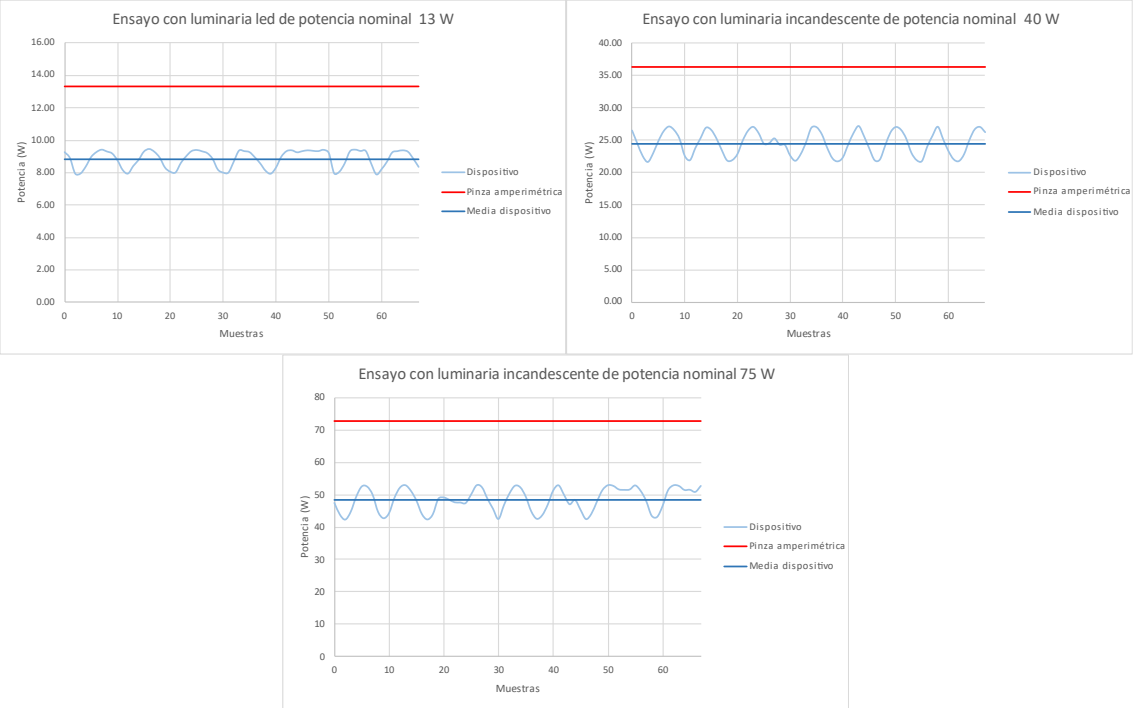


Ilustración 50: Medición de la corriente mediante pinza amperimétrica para diferentes luminarias.

Como se puede observar en la Ilustración 50, se han obtenido las siguientes mediciones para cada tipo de luminaria:

- Bombilla led de potencia nominal 13 W: Se mide una corriente de 0.058 A, lo que se traduce en una potencia consumida de 13.34 W.
- Bombilla incandescente de potencia nominal 40 W: Se mide una corriente de 0.158 A, lo que se traduce en una potencia consumida de 36.34 W.
- Bombilla incandescente de potencia nominal 75 W: Se mide una corriente de 0.317 A, lo que se traduce en una potencia consumida de 72.75 W.

A continuación, se procedió a comparar gráficamente las mediciones realizadas por el dispositivo IoT y por el multímetro, obteniendo el siguiente resultado.



Gráfica 4: Comparación de las mediciones realizadas por la pinza amperimétrica y el dispositivo IoT.

En la Gráfica 4, se pueden observar dos fenómenos diferentes. En primer lugar, las mediciones realizadas por el dispositivo tienen valores alternantes, esto es debido a que durante el cálculo de la corriente eficaz se emplean tantos valores de corriente instantánea como el microcontrolador es capaz de obtener en 200 ms. Al aumentar esta franja temporal, la variación entre mediciones se reduce. Sin embargo, se ralentiza la respuesta del dispositivo cuando se actúa sobre el dashboard.

En segundo lugar, se observa como la media de las mediciones realizadas con el dispositivo IoT distan mucho de las proporcionadas por la pinza amperimétrica. Para solventar este problema se propone añadir un factor de calibración. Se han estudiado dos posibles formas de calibración:

- Calibración por offset: Mediante esta forma de calibración, se sumará un factor de calibración a cada valor de potencia diferente de cero.
- Calibración por ganancia: Mediante esta forma de calibración, se multiplicará un factor de calibración a cada valor de potencia.

Para decidir tanto el factor como el método de calibración se ha realizado un pequeño estudio que se muestra a continuación.

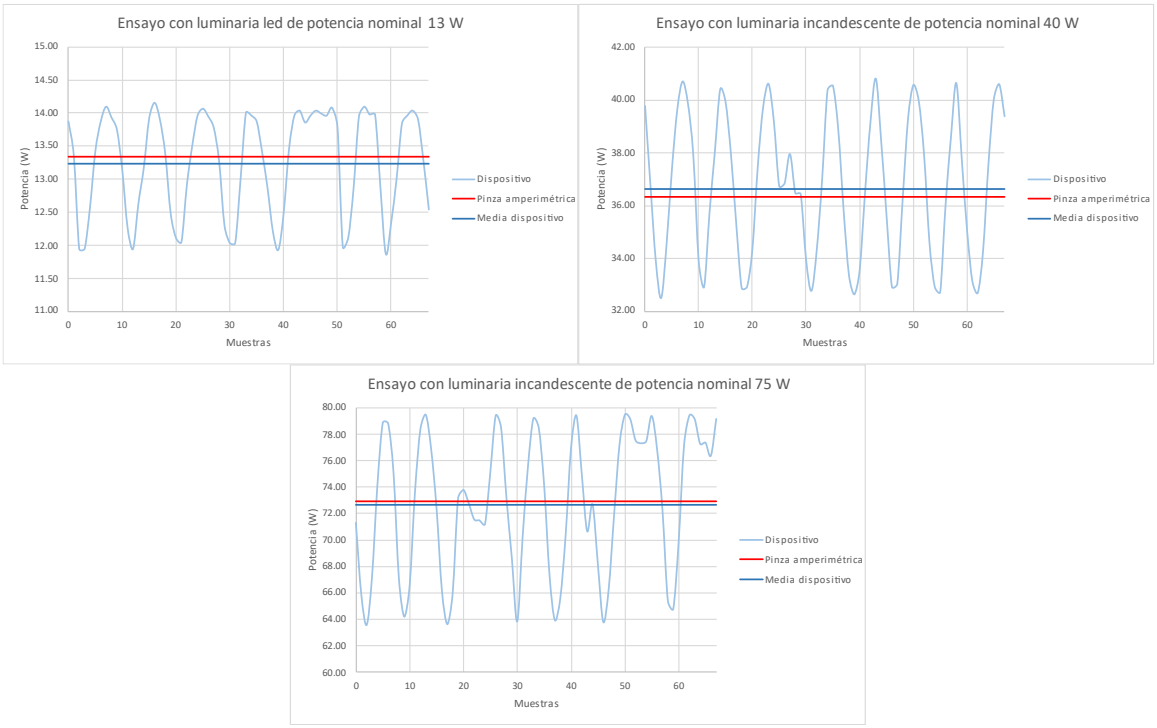
Tipo de luminaria	Pinza amperimétrica (W)	Media dispositivo (W)	Calibración por offset (W)	Calibración por ganancia (u)
Led 13 W	13.34	8.82	4.52	1.51
Incandescente 40 W	36.34	24.42	11.92	1.49
Incandescente 75 W	72.91	48.45	24.46	1.50

Indicador	Calibración por offset (W)	Calibración por ganancia (u)
Media	10.96	1.50
Desviación típica	10.08	0.01

Tabla 8: Estudio de calibración.

Dado que una de las funciones del dispositivo es ser capaz de funcionar bien con luminarias de un amplio rango de potencias, se ha descartado la calibración por offset, debido a su alta desviación típica. Por el contrario, empleando una calibración por ganancia con un factor de 1.5 unidades, se obtendrán medidas muy similares a la de la pinza amperimétrica para las tres luminarias, así queda reflejado en su baja desviación típica.

Se han simulado los valores que se obtendrían tras la calibración por ganancia con un factor de 1.5 unidades. A continuación, se muestran los resultados de dicha simulación.

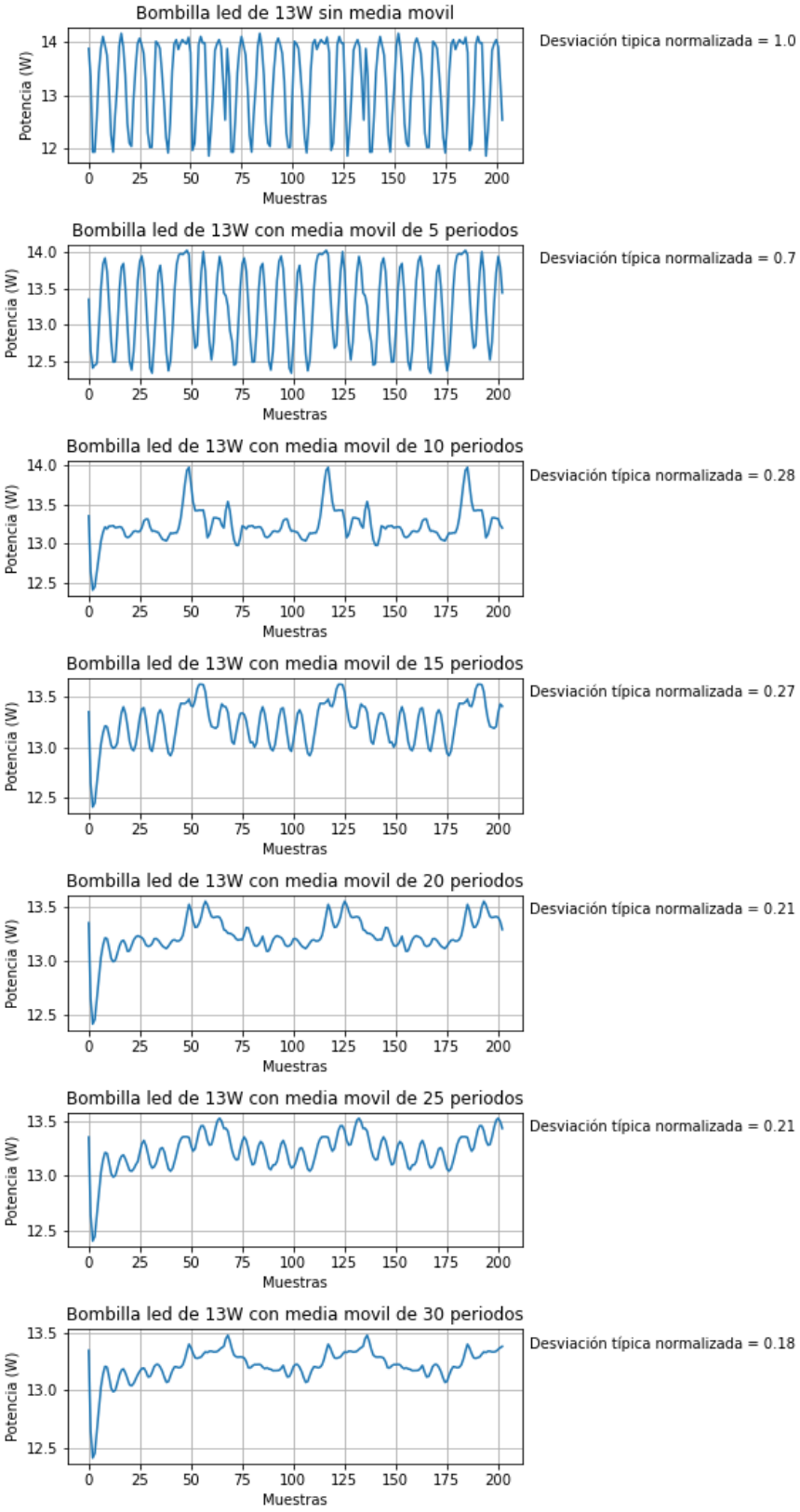


Gráfica 5: Comparación de las mediciones realizadas por la pinza amperimétrica y el dispositivo IoT tras aplicar calibración por ganancia.

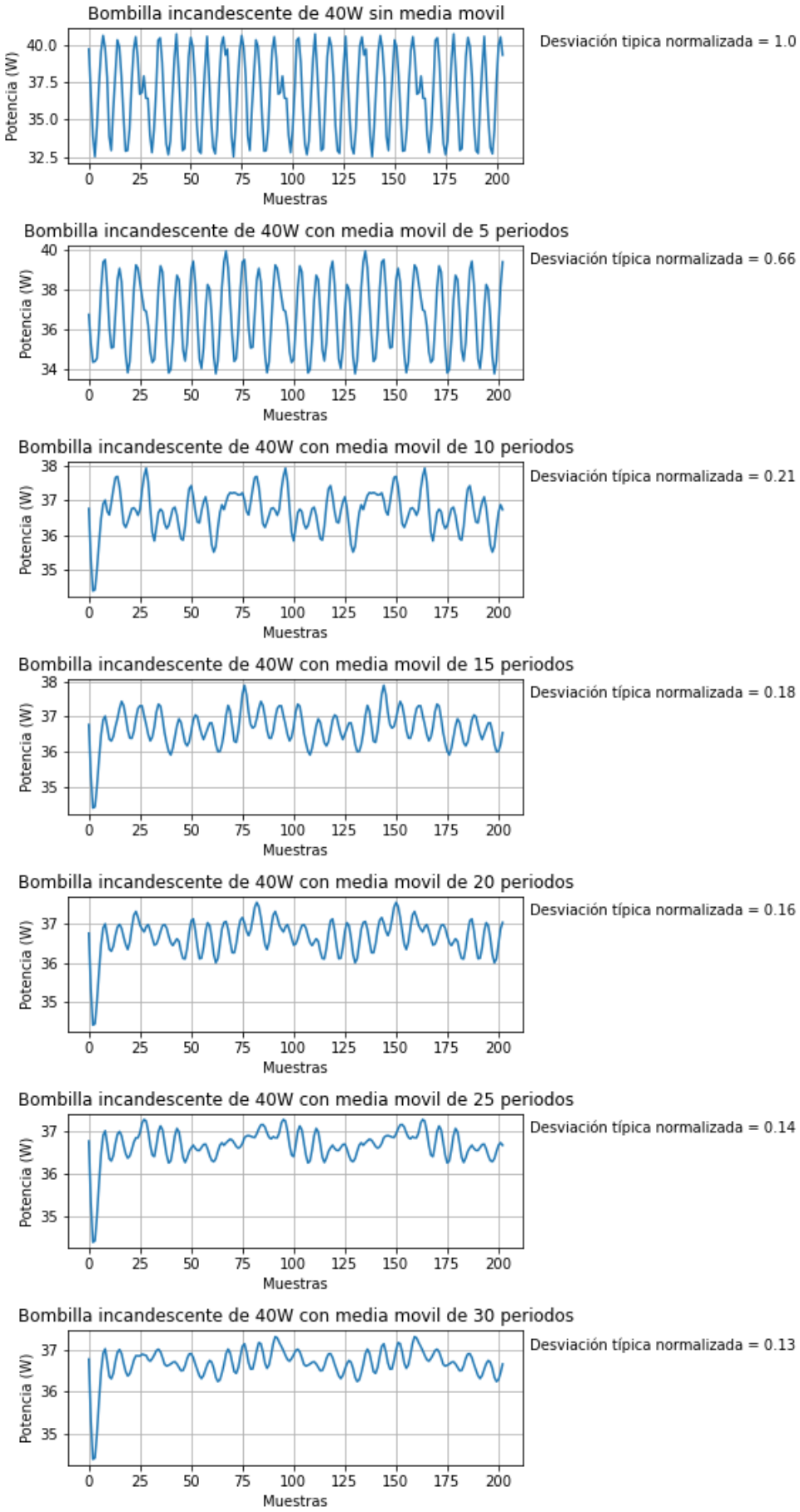
En la Gráfica 5, se muestra como tras la calibración, la media de las mediciones realizadas con el dispositivo IoT son muy similares a las de la pinza amperimétrica. Sin embargo, se observa como la variabilidad de las mediciones realizadas por el dispositivo a aumentado. Es lógico pensar, que al aplicar una calibración por ganancia se multiplica tanto la media como la desviación típica de los datos.

Para solucionar este aumento en la variabilidad se ha decidido emplear un filtro de media móvil. Este filtro se emplea para suavizar señales, recibe como entrada una señal y ofrece como salida la media de los n últimos datos de entrada. Comúnmente, al valor n se le denomina periodos del filtro, este es un valor constante que el programador del filtro deberá decidir.

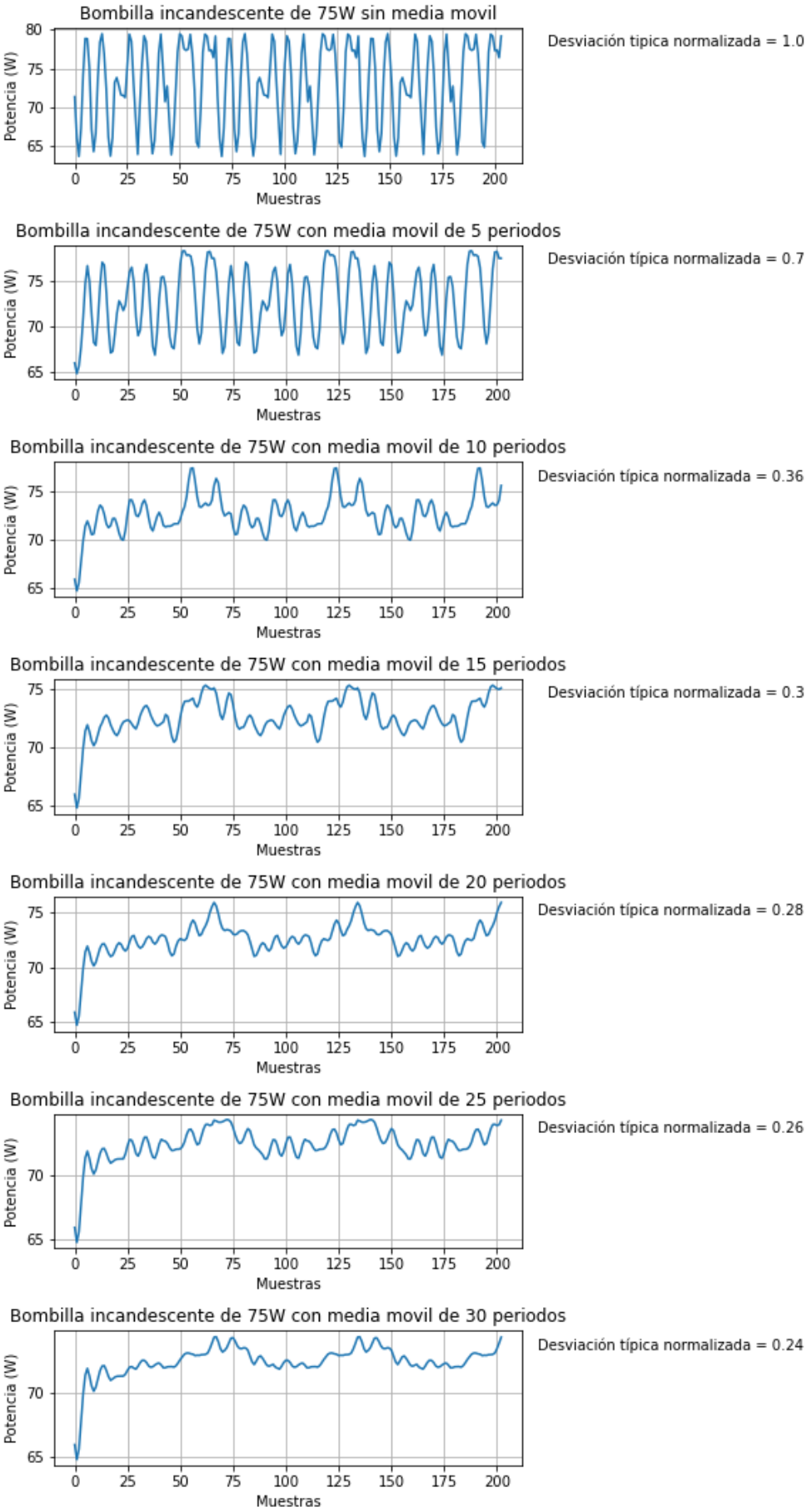
Dado que no se desea ralentizar la respuesta del dispositivo IoT, dicho filtro se ha implementado en node-red. Previamente a su implementación, se ha realizado un estudio con el fin de decidir el número de periodos con el que trabajará dicha media móvil. En este estudio se han simulado filtros de media móvil de: 5, 10, 15 y 20 periodos.



Gráfica 6: Estudio de filtro de media móvil para la bombilla led de potencia nominal 13 W.



Gráfica 7: Estudio de filtro de media móvil para la bombilla incandescente de potencia nominal 40 W.



Gráfica 8: Estudio de filtro de media móvil para la bombilla incandescente de potencia nominal 75 W.

El filtro de media móvil que se ha simulado presenta una particularidad, este solo se activará cuando los valores de potencia que otorga el ESP8266 sean superiores a cero. Se escogerá un filtro de 15 muestras, pues otorga una disminución de la oscilación respecto de la media, entre un 70% y 80%, tal y como se pueden ver en las gráficas superiores.

4.3.2- Segunda iteración

En la segunda iteración se instalaron y desarrollaron los elementos necesarios para garantizar la comunicación entre el microcontrolador y el servidor. Esto implica la instalación del bróker MQTT en el servidor, así como el desarrollo del módulo de funciones de nombre Conexion_MQTT.

Adicionalmente, se desarrolló el flujo de node-red encargado de la recepción la información publicada en el bróker, en este flujo se incluye el filtro de media móvil comentado anteriormente, Ilustración 36: Flujo para la recepción y tratamiento inicial del mensaje. Para verificar que el ESP8266 estaba publicando adecuadamente, se diseñó un sencillo dashboard que permitía monitorizar los valores publicados en el tema pub_dispositivo. Este se utilizaría más adelante como base para construir el dashboard final.

4.3.3- Tercera, cuarta y quinta iteración

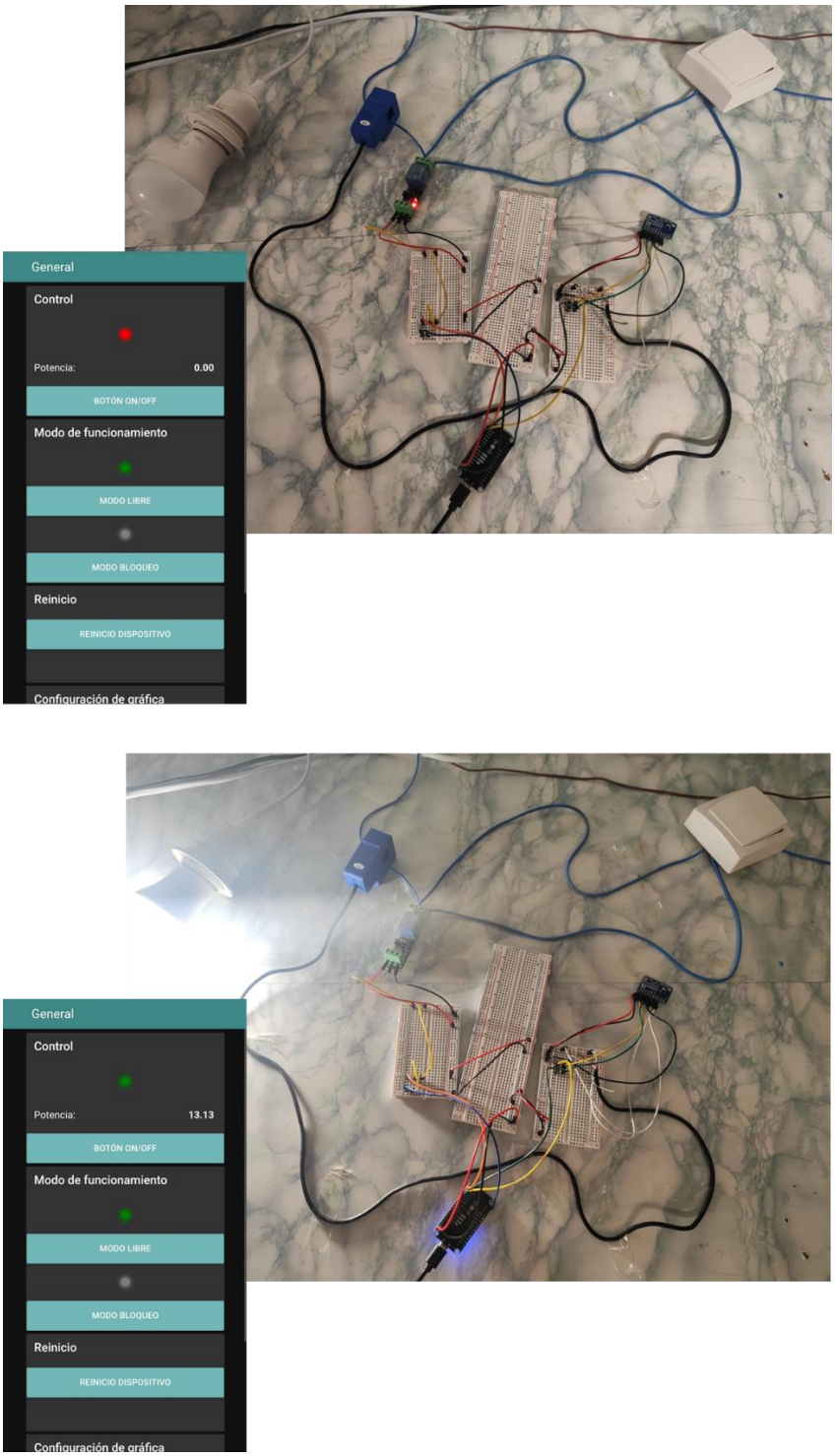
En la tercera iteración se añadió al prototipo el relé, conectado tal y como se indica en el apartado 4.2.1-Hardware. A continuación, se programó el módulo de funciones de nombre Lógica. Finalmente, se desarrollaron el resto de los flujos de node-red, a excepción de los relacionados con la representación gráfica y BBDD. Para validar esta iteración se construyó un dashboard que empleaba como base el de la iteración anterior, esta nueva interfaz ya permitía actuar sobre el dispositivo y no únicamente monitorizarlo.

En la cuarta iteración fue donde se instaló el gestor de BBDD, se creó la BBDD, se diseñó el sistema de consultas en node-red y se añadió la función de representación gráfica al dashboard. Para validar esta etapa se alimentó la BBDD con una serie de datos ficticios inyectados mediante un script de Python.

En la quinta y última iteración se instaló el host DNS y se comprobó que el dispositivo podía ser controlado remotamente a través de un teléfono móvil. Adicionalmente, se trabajó la estética final del dashboard.

5- Resultados

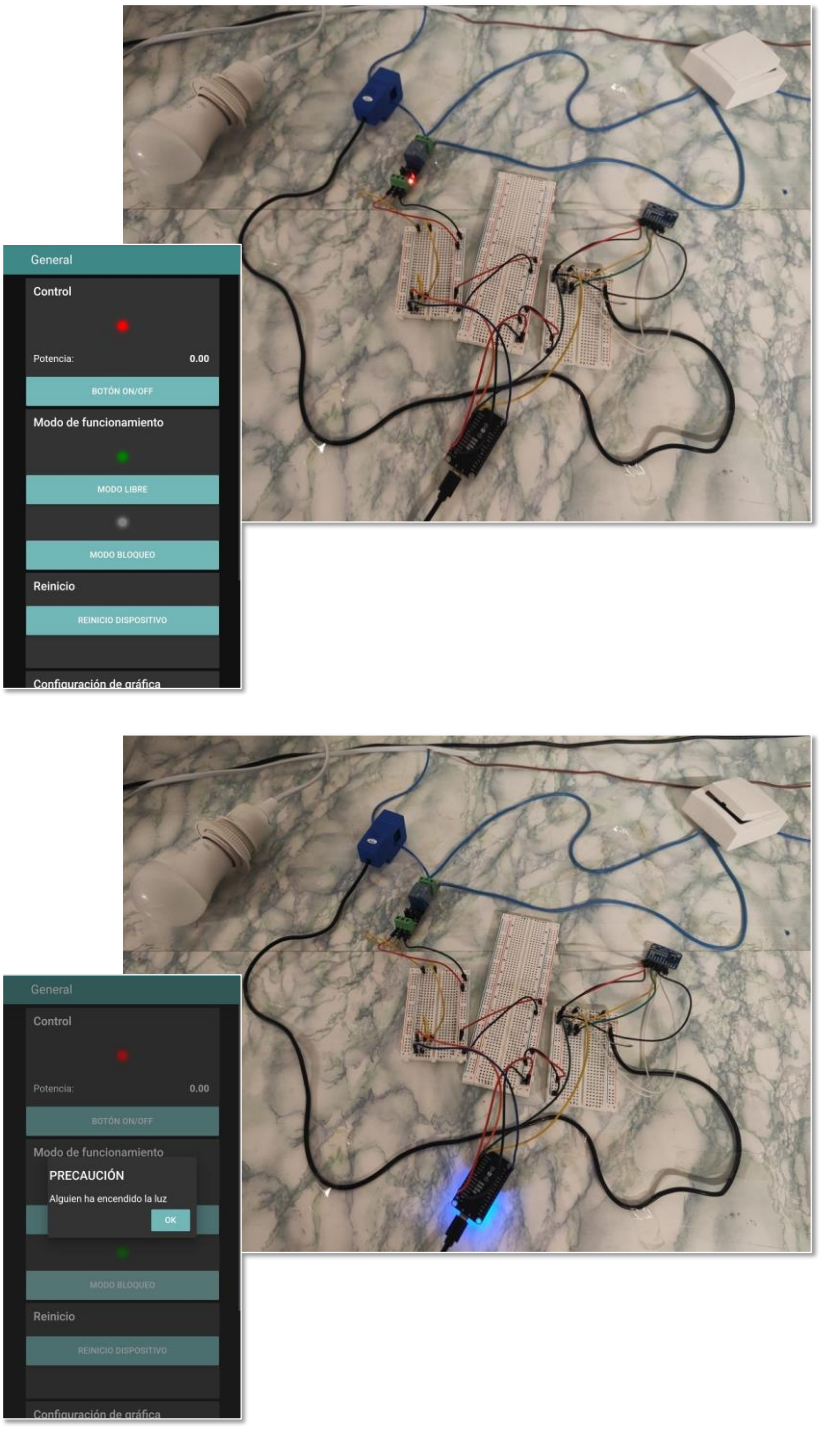
En este apartado se mostrará el funcionamiento final del prototipo. En primer lugar, se demostrará cómo la luminaria puede ser apagada y encendida en *modo libre* desde un teléfono móvil que no está conectado a la LAN de la vivienda.



Gráfica 9: Prueba de encendido y apagado en modo libre.

Adicionalmente, se observa también como el dashboard muestra la potencia instantánea consumida por la lámpara y su estado.

En las siguientes imágenes se muestra la capacidad del dispositivo para cambiar entre dos modos de funcionamiento, así como la aparición de una ventana emergente cuando el dispositivo está en *modo bloqueo* y se intenta manipular la lámpara desde el conmutador.



Gráfica 10: Prueba en modo bloqueo.

Finalmente queda demostrar el funcionamiento del visionado histórico de consumo, en las imágenes inferiores se muestran cuatro tipos de gráficos diferentes. Estos permiten visualizar el consumo entre las fechas establecidas de forma horaria, diaria, mensual y anual respectivamente.

Gráfica 11: Prueba de visionado histórico.

6- Conclusiones

En el presente trabajo de final de máster se ha logrado elaborar un prototipo de dispositivo IoT para el control de luminarias. Se ha realizado una selección de elementos hardware, que han permitido monitorizar y actuar sobre una luminaria de forma remota.

En cuanto a las telecomunicaciones, ha sido necesario elaborar una arquitectura que permitiese alcanzar las funcionalidades comentadas al inicio del presente documento. Para ello se ha creado un servidor local, en el que han diferentes tecnologías de la comunicación como son el bróker MQTT y el cliente DNS. Además, se ha elaborado un protocolo de transmisión de comandos, que empleando el protocolo MQTT permite el intercambio de información con el dispositivo de forma ordenada y eficiente.

El software se ha trabajado en dos niveles diferentes. El primer nivel, ha sido el propio firmware del microcontrolador, donde se han desarrollado tres módulos de funciones, que en términos generales han permitido al dispositivo: conocer el estado de la luminaria y la potencia instantánea consumida, conectar el microcontrolador a la LAN del hogar a través de WiFi, conectar el microcontrolador al bróker MQTT de un servidor local, actuar sobre la luminaria de acuerdo con instrucciones mandadas a través de una interfaz de usuario.

El segundo nivel ha sido la programación de node-red. Este actúa como agente aglomerante de los diferentes servicios residentes en el servidor, ha permitido gestionar los diferentes flujos de información generados por al bróker, el dashboard y la BBDD.

Para poder desarrollar el presente prototipo se han empleado conocimientos de muchas áreas diferentes. Han sido necesarios conocimientos del ámbito de la electrónica para poder escoger justificadamente el sensor de medición de corriente, el conversor analógico digital, el relé y el microcontrolador.

También se han empleado competencias propias de la rama del control y la automática en la calibración de la potencia calculada por el microcontrolador y en su posterior filtrado.

Adicionalmente, ha sido necesario comprender elementos pertenecientes al área de las telecomunicaciones cómo: el funcionamiento del enrutamiento de la información tanto a nivel LAN como a través de internet, el funcionamiento de los protocolos de comunicación MQTT y I2C, el funcionamiento de servicios DNS y el funcionamiento de transmisión de comandos mediante formato JSON.

Finalmente, para poder desarrollar el prototipo se han usado diferentes lenguajes de programación. El ESP8266 ha sido programado en C++ con la IDE y APIs propias de Arduino, las funciones de node-red han sido desarrolladas en JavaScript y las consultas a la BBDD han empleado lenguaje SQL. Aunque no directamente en el prototipo, también se ha empleado Python para realizar el estudio del filtro de media móvil y alimentar la BBDD con valores ficticios correspondientes a cuatro meses de funcionamiento.

7-Bibliografía

Adafruit. (29 de Marzo de 2024). *Adafruit_ADS1X15*. Obtenido de GitHub: https://github.com/adafruit/Adafruit_ADS1X15

Alldatasheet. (29 de Marzo de 2024). *Repositorio de datasheets para sensores de la serie SCT-013*. Obtenido de Sitio web Alldatasheet: https://www.alldatasheet.es/view_datasheet.jsp?Searchword=SCT013

Alldatasheet. (29 de Marzo de 2024). *Repositorio del datasheet para el conversor ADC ADS1115.* Obtenido de Sitio web Alldatasheet: https://www.alldatasheet.com/datasheet-pdf/pdf/292735/TI/ADS1115.html

Alldatasheet. (29 de Marzo de 2024). *Repositorio del datasheet para el sensor ACS712*. Obtenido de Sitio web Alldatasheet: https://www.alldatasheet.es/datasheet-pdf/pdf/168326/ALLEGRO/ACS712.html

Arduino. (29 de Marzo de 2024). *ESP8266WiFi*. Obtenido de GitHub: https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi

AWS, A. (19 de Febrero de 2024). *¿Qué es el middleware?* Obtenido de Sitio web Amazon AWS: https://aws.amazon.com/es/what-is/middleware/

AWS, A. (20 de Marzo de 2024). *SDK de AWS para JavaScript*. Obtenido de Sitio web Amazon AWS: https://docs.aws.amazon.com/es_es/sdk-for-javascript/v2/developer-guide/working-with-json.html

Deloitte. (18 de Febrero de 2024). *IoT - Internet Of Things*. Obtenido de Sitio web Deloitte: https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html

DuckDNS. (29 de Marzo de 2024). *Instalación DuckDNS*. Obtenido de Sitio web DuckDNS: https://www.duckdns.org/install.jsp

Esspresif. (29 de Marzo de 2024). *ESP32 Series Datasheets.* Obtenido de Sitio web Esspresif: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

Esspresif. (29 de Marzo de 2024). *ESP8266 Technical Reference.* Obtenido de Sitio web Esspresif: https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf

knolleary, U. (29 de Marzo de 2024). *pubsubclient*. Obtenido de GitHub: https://github.com/knolleary/pubsubclient

Mosquitto. (29 de Marzo de 2024). *Documentación.* Obtenido de Sitio web Mosquitto: https://mosquitto.org/documentation/

Node-Red. (23 de Febrero de 2024). *Node-RED*. Obtenido de Sitio web Node-Red: https://nodered.org/

Node-RED. (29 de Marzo de 2024). *node-red-dashboard*. Obtenido de GitHub: https://github.com/node-red/node-red-dashboard

Oracle. (18 de Febrero de 2024). *¿Qué es el IoT?* Obtenido de Sitio web Oracle Corporation: https://www.oracle.com/es/internet-of-things/what-is-iot/

RAE. (18 de Febrero de 2024). *Definición de servidor*. Obtenido de Sitio web RAE: https://dle.rae.es/servidor

8-Presupuesto

8.1- Consideraciones previas

Para el cálculo de la valoración económica del trabajo objeto del presente TFM se ha realizado un análisis de costes. Para determinar el coste que le supondría un trabajador a la empresa se ha tomado como referencia el último convenio colectivo para la industria, las nuevas tecnologías y los servicios del sector del metal de la provincia de Valencia.

A continuación, se adjunta la tabla salarial para el citado convenio.

		TABLAS 2024 APLICABLES DESDE ENERO 2024				
CLASIFICACION PROFESIONAL	GRUPO	Salario Grupo	Sal. Ex Categ.	Plus Conv.	SALARIO ANUAL	Salario anual con Excat
		2024	2024	2024	2024	2024
PERSONAL TECNICO						
Ingenieros/as y Licenciados/as	1	2.259,77 €		146,99 €	34.681,20 €	
Analista de sistemas	1	2.259,77 €		146,99 €	34.681,20 €	
PERSONAL TECNICO						
Técnicos/as y Aparejadores/as	2	1.968,76 €		146,99 €	30.442,15 €	
Graduados/as Sociales	2	1.968,76 €		146,99 €	30.442,15 €	
Ayudante Técnico Sanitario	2	1.968,76 €		146,99 €	30.442,15 €	
PERSONAL TECNICO						
Jefe/a de Primera (Téc.Org.Tr.)	3	1.686,75 €	21,65 €	159,62 €	26.485,77 €	26.788,87 €
Jefe/a de Primera (Téc.Labor,)	3	1.686,75 €	21,65 €	159,62 €	26.485,77 €	26.788,87 €
Programador/a	3	1.686,75 €		159,62 €	26.485,77 €	
Deliniante Proyectista	3	1.686,75 €		159,62 €	26.485,77 €	
PERSONAL OPERARIO						
Jefe/a de Taller	3	1.686,75 €	42,39 €	159,62 €	26.485,77 €	27.079,23 €
PERSONAL EMPLEADO						
Jefe/a de Primera (Admivo)	3	1.686,75 €	21,65 €	159,62 €	26.485,77 €	26.788,87 €
PERSONAL TECNICO						
Jefe/a de Segunda (Téc.Org.Tr.)	4	1.576,54 €		159,62 €	24.880,37 €	
Jefe/a de Segunda (Téc. Labo.)	4	1.576,54 €		159,62 €	24.880,37 €	
Maestros/as Industriales	4	1.576,54 €		159,62 €	24.880,37 €	
PERSONAL DE OPERACIONES						
Encargado/a	4	1.622,05 €		159,62 €	25.543,30 €	
PERSONAL EMPLEADO						
Jefe/a de Segunda. (Admivo/a)	4	1.578,34 €		159,62 €	24.906,59 €	
Cajero/a (empresas de 250 a 1000 de plantilla)	4	1.578,34 €		159,62 €	24.906,59 €	
PERSONAL DE OPERACIONES						
Oficial/a de Primera (Profesional de Oficio)	5	46,93 €		6,75 €	22.814,99 €	
Oficial/a de Segunda (Profesional de Oficio)	5	46,93 €		6,75 €	22.814,99 €	

Tabla 9: Tabla salarial del convenio colectivo para la industria, las nuevas tecnologías y los servicios del sector del metal de la provincia de Valencia.

El salario bruto anual para un Ingeniero y Licenciado es de 34.681,20€ tal y como muestra la Tabla 9.

El coste real de este trabajador para la empresa es lo que se conoce como coste de empresa y se determina añadiendo el al salario bruto anual del trabajador a los costes que la empresa paga en concepto de seguridad social, indemnización, gastos de formación, etc.

CÁLCULO APROXIMADO DEL COSTE ANUAL EMPRESA		
Salario bruto (convenio colectivo 2023-2026)		34,681.20 €
Seguridad social a cargo de la empresa	23.60%	8,184.76 €
Tipo general de desempleo para trabajador indefinido	5.50%	1,907.47 €
FOGASA	0.20%	69.36 €
Formación profesional	0.70%	242.77 €
Total		45,085.56 €

Tabla 10: Cálculo aproximado del coste total anual para una empresa de un Ingeniero.

CÁLCULO APROXIMADO DEL COSTE ANUAL EMPRESA	
Días 2023	365
Días festivos	14
Días vacaciones	30
Días en sabado y domingo	105
Días totales laborales	**216**
Horas de trabajo al año (8h/dia)	**1728**
Coste bruto total (€/h)	**20.07**

Tabla 11: Cálculo aproximado del coste horario para una empresa de un ingeniero.

Al coste de la empresa debería aplicarse unos porcentajes en concepto de gastos generales. Este depende de cada empresa, pero se ha decidido emplear el que se aplica en proyectos de ingeniería cuando se trabaja con la administración pública y que corresponde al 13%.

8.2-Cuadro de precios descompuestos

Código	Ud	Descripción	Rendimiento	Precio	Importe
1	CAP1	Elección, configuración y diseño de la arquitectura para los sistemas involucrados en el funcionamiento del dispositivo IoT.			
1.1	Ud	Diseño y elección de sistemas hardware			
		Diseño general de la arquitectura de conexiones para los sistemas hardware del dispositivo			
A1	Ud	Ingeniero industrial	8	20.07	160.56
		Elección de los sistemas hardware del dispositivo IoT			
A1	Ud	Ingeniero industrial	16	20.07	321.12
		Diseño de detalle de la arquitectura de conexiones para los sistemas hardware del dispositivo			
A1	Ud	Ingeniero industrial	4	20.07	80.28
			Coste total		561.96 €
1.2	Ud	Diseño y elección de sistemas software			
		Diseño general de la arquitectura para los sistemas software y servicios alojados en el servidor			
A1	Ud	Ingeniero industrial	5	20.07	100.35
		Elección de los sistemas software y servicios alojados en el servidor			
A1	Ud	Ingeniero industrial	2	20.07	40.14
		Diseño de detalle de la arquitectura para los sistemas software y servicios alojados en el servidor así como diseño del protocolo de transmisión de comandos			
A1	Ud	Ingeniero industrial	5	20.07	100.35
			Coste total		240.84 €

Código	Ud	Descripción	Rendimiento	Precio	Importe
2	CAP2	Desarrollo y depuración del firmware del dispositivo en C++programado en la IDE de Arduino			
2.1	Ud	Desarrollo de funciones			
		Desarrollo del módulo de funciones para la monitorización del estado y potencia de la luminaria			
A1	Ud	Ingeniero industrial	16	20.07	0
		Desarrollo del módulo de funciones para la conexión del dispositivo a la red y al servidor			
A1	Ud	Ingeniero industrial	14	20.07	280.98
		Desarrollo del módulo de funciones para el establecimiento de la lógica del dispositivo			
A1	Ud	Ingeniero industrial	20	20.07	401.4
		Desarrollo de la función principal mediante programación modular			
A1	Ud	Ingeniero industrial	6	20.07	120.42
			Coste total		802.80 €
2.2	Ud	Pruebas y depuración			
		Prueba y depuración del módulo de funciones para la monitorización del estado y potencia de la luminaria			
A1	Ud	Ingeniero industrial	3	20.07	60.21
		Prueba y depuración del módulo de funciones para la conexión del dispositivo a la red y al servidor			
A1	Ud	Ingeniero industrial	2	20.07	40.14
		Prueba y depuración del módulo de funciones para el establecimiento de la lógica del dispositivo			
A1	Ud	Ingeniero industrial	3	20.07	60.21
		Prueba y depuración de la función principal elaborada mediante progaramación modular			
A1	Ud	Ingeniero industrial	2	20.07	40.14
			Coste total		200.70 €

Código	Ud	Descripción	Rendimiento	Precio	Importe
3	CAP3	Desarrollo y depuración de flujos , funciones y dashboard de node-red			
3.1	Ud	Desarrollo de flujos y dashboard			
		Desarrollo de los flujos para la recepción y tratamiento inicial del mensaje			
A1	Ud	Ingeniero industrial	10	20.07	200.7
		Desarrollo de los flujos para el tratamiento del mensaje para monitorización y visualización de la información			
A1	Ud	Ingeniero industrial	9	20.07	180.63
		Desarrollo de los flujos para el tratamiento del mensaje para comunicación con el dispositivo IoT y publicación			
A1	Ud	Ingeniero industrial	26	20.07	521.82
		Desarrollo de los flujos para la gestión de la BBDD y representación gráfica de los datos			
A1	Ud	Ingeniero industrial	19	20.07	381.33
			Coste total		1,284.48 €
3.2	Ud	Pruebas y depuración			
		Prueba y depuración de los flujos para la recepción y tratamiento inicial del mensaje			
A1	Ud	Ingeniero industrial	2	20.07	40.14
		Prueba y depuración de los flujos para el tratamiento del mensaje para monitorización y visualización de la información			
A1	Ud	Ingeniero industrial	1	20.07	20.07
		Prueba y depuración de los flujos para el tratamiento del mensaje para comunicación con el dispositivo IoT y publicación			
A1	Ud	Ingeniero industrial	4	20.07	80.28
		Prueba y depuración de los flujos para la gestión de la BBDD y representación gráfica de los datos			
A1	Ud	Ingeniero industrial	3	20.07	60.21
			Coste total		200.70 €

Código	Ud	Descripción	Rendimiento	Precio	Importe
4	CAP4	Adquisición de los sistemas hardware para construcción del prototipo, montaje y depuración final			
4.1	Ud	Adquisición de sistemas hardware			
		Adquisición de luminaria led de 13 W para construcción del prototipo			
S1	Ud	Luminaria led 13 W	1	5.60	5.6
		Adquisición de luminaria incandescente de 40 W para construcción del prototipo			
S2	Ud	Luminaria incandescente 40 W	1	4.50	4.5
		Adquisición de luminaria incandescente de 75 W para construcción del prototipo			
S3	Ud	Luminaria incandescente 75 W	1	8.50	8.5
		Adquisición de conjunto constituido por tres placas de pruebas y conectores			
S4	Ud	Placa de pruebas	1	13.99	13.99
		Adquisición de la placa NodeMCU con ESP8266 para construcción de prototipo			
S5	Ud	NodeMCU con ESP8266	1	7.99	7.99
		Adquisición de sensor de corriente SCT-013-005 para construcción del prototipo			
S6	Ud	SCT-013-005	1	10.39	10.39
		Adquisición de convertidor analógico digital ADS1115 para construcción del prototipo			
S7	Ud	ADS1115	1	4.99	
			Coste total		50.97 €
4.2	Ud	Montaje del prototipo y depiuración final			
		Montaje del prototipo de acuerdo al diseño de detalle realizado			
A1	Ud	Ingeniero industrial	2	20.07	40.14
		Prueba y depuración del prototipo final con todos los sistemas integrados			
A1	Ud	Ingeniero industrial	18	20.07	361.26
			Coste total		401.40 €

8.3- Presupuesto y mediciones

Nº Orden	Ud	Descripción	Medición	Precio	Importe
1		Elección, configuración y diseño de la arquitectura para los sistemas involucrados en el funcionamiento del dispositivo IoT.			
1.1	Ud	Diseño y elección de sistemas hardware	1	561.96	561.96
1.2	Ud	Diseño y elección de sistemas software	1	240.84	240.84
	%	Costes indirectos	0.03	802.80	24.084
				Total capítulo 1	**826.88 €**
2		Desarrollo y depuración del firmware del dispositivo en C++programado en la IDE de Arduino			
2.1	Ud	Desarrollo de funciones	1	802.80	802.80
2.2	Ud	Pruebas y depuración	1	200.70	200.7
	%	Costes indirectos	0.03	1003.50	30.105
				Total capítulo 2	**1,033.61 €**
3		Desarrollo y depuración de flujos , funciones y dashboard de node-red			
3.1	Ud	Desarrollo de flujos y dashboard	1	1284.48	1284.48
3.2	Ud	Pruebas y depuración	1	200.70	200.7
	%	Costes indirectos	0.03	1485.18	44.5554
				Total capítulo 3	**1,529.74 €**
4		Montaje del prototipo de acuerdo al diseño de detalle realizado			
4.1	Ud	Montaje de prototipo	1	50.97	50.97
4.2	Ud	Técnico industrial superior	1	401.40	401.4
	%	Costes indirectos	0.03	452.37	13.5711
				Total capítulo 4	**465.94 €**

8.4- Presupuesto final

PRESUPUESTOS

CAP1	Elección, configuración y diseño de la arquitectura para los sistemas involucrados en el funcionamiento del dispositivo IoT	826.88 €
CAP2	Desarrollo y depuración del firmware del dispositivo en C++programado en la IDE de Arduino	1,033.61 €
CAP3	Desarrollo y depuración de flujos , funciones y dashboard de node-red	1,529.74 €
CAP4	Montaje del prototipo de acuerdo al diseño de detalle realizado	465.94 €

TOTAL.....	3,856.17 €
13% Gastos generales.....	501.30 €
PRESUPUESTO TOTAL.....	4,357.47 €

Asciende el presupuesto proyectado, a la expresada cantidad de
CUATRO MIL TRESCIENTOS CINCUENTA Y SIETE EUROS CON CUARENTA Y SIETE CÉNTIMOS