UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE APLICACIÓN IoT PARA LA MONITORIZACIÓN DE CONSUMOS ELÉCTRICOS EN UNA VIVIENDA

TRABAJO FINAL DEL

Grado en Ingeniería Eléctrica

REALIZADO POR

David Llorca Llinares

TUTORIZADO POR

Tutor: Sapena Baño, Ángel

Segungo tutor: Puche Panadero, Rubén

CURSO ACADÉMICO: 2019/2020

ÍNDICE TFG:

AGRADECIMIENTOS:

A Dios por darme la vida y permitirme cumplir mis metas, a mi familia por darme su apoyo y amor. A mis amigos por acompañarme durante todo este proceso.

Expresar mi agradecimiento a los profesores de la Escuela Técnica Superior de Ingeniería del Diseño por haber compartido sus conocimientos conmigo y haberme ayudado durante estos años de estudio.

En especial a mis tutores Ángel Sapena y Rubén Puche, por los consejos y ayuda brindada durante estos complicados meses, La formación que me han dado estos dos últimos años y la buena relación que hemos tenido.

PALABRAS CLAVE:

ESP32, MQTT, IoT, WiFi, Arduino, NodeRed.

ACRÓNIMOS:

API: Application Programming Interface
MQTT: Message Queue Telemetry Transport
M2M: Machine-2-Machine
IoT: Internet of Things
SoC: System on Chip
PCB: Printed Circuit Board
GPS: Global Positioning System
PCI: Peripheral Component Interconnect
RF: Radio Frequency
SPI: Serial Peripheral Interface
SDK: Software Development Kit
CPU: Central Processing Unit
ADC: Analog Digital Converter
RTC: Real Time Clock
AGC: Automatic Gain Control
SRAM: Static Random Access Memory
UART: Universal Asynchronous Receiver-Transmitter
UE: Unión Europea
RDBMS: Sistema de Gestión de Bases de Datos Relacionales

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE APLICACIÓN IoT PARA LA MONITORIZACIÓN DE CONSUMOS ELÉCTRICOS EN UNA VIVIENDA

MEMORIA

ÍNDICE

1. INTRODUCCIÓN.

1.1. CONTEXTO ENERGÉTICO.

En los últimos años hay ciertos aspectos a los que la sociedad les está atribuyendo mucha importancia, como por ejemplo el acceso la información, la velocidad de transmisión, y la calidad de los datos. Gran parte de la culpa la tiene el avance de la tecnología. Debido a esto, el Internet de las Cosas (IoT) está ganando protagonismo, y más aún dentro del concepto de domótica.

El IoT se basa en la interconexión de dispositivos, los cuales están dotados de una conexión a internet para que puedan interaccionar entre ellos. Un ejemplo de elementos en una vivienda que pueden integrarse en una aplicación IoT serían: electrodomésticos, sensores distribuidos por las estancias, etc. En conclusión, cualquier elemento podría adquirir la capacidad de conectarse a internet de forma que pueda comunicarse entre otros dispositivos y ofrecer información importante para los usuarios.

Cada vez se tiene más en cuenta la necesidad de reducir el consumo y el gasto superfluo de energía. Ya en el año 2007, la Unión Europea estableció el objetivo de reducir el consumo anual de energía en la UE en un 20%. Más tarde en el año 2018 se estableció un nuevo objetivo, el de reducir el consumo de energía en al menos un 32,5% en el horizonte 2030. Actualmente se está debatiendo el marco de actuación para 2030 en adelante. Por lo que, La eficiencia energética es una de las prioridades de la UE, entre otras razones, por que, además de reducir las emisiones de gases de efecto invernadero, reducir costes, y abastecerse de energía sostenible, también contribuye a la competitividad de la Unión.

La tecnología IoT está cada vez más presente en el campo de la eficiencia energética. Ya que, podemos obtener información significativa sobre el consumo de energía que se está dando en todo momento, para saber qué medidas adoptar para el ahorro energético y cómo aplicarlas.

Por lo que tanto la industria, distintos negocios, y las viviendas, tienen que realizar su aportación según sus capacidades para alcanzar los objetivos de eficiencia energética, establecidos por la UE.

1.2. OBJETIVOS.

El objetivo general de este trabajo de fin de grado es desarrollar una aplicación IoT para la monitorización de consumos eléctricos en una vivienda. Al ser una aplicación IoT, es necesario establecer comunicaciones entre los dispositivos que van a integrar la aplicación, y estos conectarlos a la red. Para alcanzar con éxito el objetivo fundamental se establecen los objetivos parciales siguientes:

- Estudio de las necesidades reales del proyecto en cuanto a características y elementos a utilizar; y el estudio, planteamiento de los posibles elementos que puedan dar respuesta a estas necesidades.
- Selección de las magnitudes a medir y los sensores adecuados para monitorizar el consumo eléctrico de la instalación.
- Selección del equipo de control y desarrollo de los algoritmos necesarios para la recepción de la información obtenida por los sensores, el cálculo de los datos de consumo eléctrico y la transmisión de la información.
- Conexionado de los sensores al circuito eléctrico y al microcontrolador, con su respectivo acondicionamiento de la señal para la correcta recepción por parte del elemento de control.
- Conexión a la red Wifi por parte del microcontrolador para el desarrollo de la aplicación IoT.
- Implementación del protocolo de comunicación que facilite el intercambio de información, con plataformas que permitan el acceso a los datos de la instalación (corrientes, tensiones, potencias, consumos y base de datos) desde cualquier lugar del mundo, utilizando plataformas estandarizadas.
- Procesado de las señales de tensión que recibe el microcontrolador por parte de los sensores para la obtención y envío de los datos relevantes a la interfaz del usuario (Potencia, tensión, corriente y consumo).
- Diseño de una interfaz de usuario sencilla e intuitiva con diferentes apartados que muestren, de forma organizada, la información obtenida de los sensores, el resultado de los cálculos del elemento de control y la visualización de la base de datos.
- Almacenaje de los valores de consumo eléctrico ordenados por fecha y hora en una base de datos relacional, para su posterior consulta y/o análisis.

Una vez definidos los objetivos que se plantean en este sistema, la estructura que se va a seguir en el codumento es la siguiente. En primer lugar, se plantean los requisitos de los diferentes elementos que forman la aplicación. Seguidamente, es importante la realización de un estudio previo en el que se establezca las necesidades de la aplicación en cuanto a los elementos, características…

A continuación, se planteará el desarrollo de las posibles soluciones que pueden dar respuesta a las necesidades del proyecto, teniendo en cuenta lo que ofrece el mercado en esta industria. Finalmente, se procederá a la explicación detallada de la solución adoptada, explicando como se ha ido ejecutando hasta alcanzar el resultado final.

Cabe comentar que este prototipo puede tener diferentes versiones, dependiendo de en qué campo se vaya a implementar (industria, viviendas), los requisitos que pueda tener cada caso particularmente y el objetivo de la aplicación. Por lo que se podrían utilizar elementos de mayor potencia en entornos industriales, con sensores preparados para trabajar en este tipo de entorno. Y teniendo en cuenta el objetivo o finalidad con la que se quiera implementar la aplicación, se puede utilizar un número de elementos mayor o menor. Por ejemplo, en el caso de querer tener una visión general del consumo podría instalarse un solo sensor de corriente de forma que solo se obtendría la curva de carga de la totalidad de la instalación.

Así mismo, como el objetivo fundamental es conseguir la eficiencia energética, cuanto mayor número de instalaciones se estén monitorizando, más información se podrá recabar. Esta información es muy útil para realizar comparativas entre diferentes instalaciones, o comparativas del consumo de diferentes empresas de la misma industria. También, se pueden emplear estos datos para realiza estudios, e informes sobre como ha variado el consumo de energía a lo largo del tiempo, implementado el IoT. Siempre que el número de instalaciones permita realizar estos estudios.

Utilizando más sensores de corriente, se podría poner el foco en el circuito de especial interés, o emplear un sensor para cada circuito, obteniendo los consumos separados por circuitos y el general. Por lo que el diseño es flexible y escalable, adaptándose a las necesidades especificas de los clientes.

2. ESTUDIO DE NECESIDADES.

El proyecto cuenta con unos requisitos y necesidades que han de tenerse en cuenta a la hora de llevarlo a cabo. Serán necesarios distintos elementos que se explican en los siguientes subapartados. Un elemento de control, sensores que permitan medir señales para la obtención del consumo eléctrico, protocolos de comunicación, interfaz del usuario, base de datos.

2.1. ELEMENTO DE CONTROL.

- **Elemento de control**, se encargará de realizar los cálculos del consumo eléctrico en cada circuito y el consumo eléctrico total a partir de los valores de tensión y corriente de cada circuito, que recibirá de los sensores. Además de la conexión a internet y las comunicaciones. Debe cumplir con los siguientes requisitos:
    - Tamaño: debe estar diseñado para su utilización y funcionamiento en el entorno de una vivienda. Siendo de pequeño tamaño y fácil instalación.
    - Estandarización: debe cumplir con los estándares de comunicación y programación.
    - Flexible y escalable: debe de poder adaptarse y ampliarse en caso de que el proyecto creciera o hubiera cambios en el proceso.
    - Entradas analógicas: debe de tener un mínimo de 6 entradas analógicas para los sensores de la aplicación.
    - Memoria de carga: debe tener un mínimo de 4 MB.
    - Interfaces de comunicación: debe poseer conexiones USB para su programación. Además, debe tener comunicación Wifi.
    - Asistencia técnica: disponibilidad de documentación y manuales.
    - El software implementado en el elemento de control debe realizar los cálculos para la obtención de los consumos, así como la conexión a la red Wifi y utilizar las comunicaciones en cuestión.
    - Precio: debe tener un coste bajo para que el prototipo sea barato. De forma que puede ser accesible para un gran número de clientes. Ya que en un principio está pensado para su aplicación en el ámbito doméstico, el precio debe ser reducido para que cualquier persona pueda permitírselo.

2.2. SENSORES.

Se ha decidido emplear sensores para la medida de tensión y corriente ya que, para la aplicación que se va a desarrollar, la información que es necesaria recabar, se limita a las corrientes, tensiones y los consumos (a partir de los cálculos con la tensión y corriente). Un analizador de redes nos da una cantidad mayor de información, que para la finalidad de esta aplicación no es necesaria, por lo general son más caros que un pequeño transformador de tensión y corriente, lo que incrementa el precio final del prototipo.

- **Sensor de tensión**, para la adaptación de la onda de tensión, la cual va a ser la misma en toda la vivienda, por lo que solo es necesario emplear un sensor. Debe cumplir con los siguientes requisitos:
    - Rango de medida: la tensión nominal de una vivienda en España es de 230V por lo que debe se capaz de trabajar con esos niveles de tensión.
    - Compatibilidad: debe de poder adaptarse al elemento de control, tanto en la conexión como en el intercambio de información.
    - Fiabilidad y precisión: el resultado de las medidas debe dar unos resultados fieles a la realidad, siendo reflejo de la tensión en el circuito de la vivienda, con una precisión mínima de 0.2% - 0.3%.
    - Tamaño y robustez: debe de tener un tamaño pequeño para utilizarse en el entorno en que va a ser utilizado, al ser una vivienda el espacio es reducido por lo que lo ideal sería como máximo 50x20x20mm, pudiendo de ser de mayor tamaño para entornos industriales. Además, debe ser duradero.
    - Coste: teniendo en cuenta el presupuesto con el que se cuenta, se han de cumplir los máximos requisitos posibles de los elementos con este presupuesto. No superando los 10€.

Como se ha comentado anteriormente, una de las características de este prototipo es la escalabilidad. Por lo que, para su aplicación en un entorno industrial, en el que la red es trifásica, se emplearían tres sensores de tensión, uno para cada fase. En la vivienda se emplea uno, ya que es monofásica.

- **Sensores de corriente**, los cuales se encargarán de obtener las corrientes en cada circuito de la vivienda. Partiendo de que lo más habitual es que una vivienda tenga un grado de electrificación básica, y la electrificación básica tiene un máximo de cinco circuitos destinados a diferentes fines, se necesitan cinco sensores de corriente para que cada uno obtenga la corriente que hay en cada circuito, y de esta forma obtener el consumo eléctrico por cada circuito independientemente. Deben cumplir con los siguientes requisitos:

- Rango de medida: Debe de ser capaz de poder trabajar y medir con las corrientes que se vaya a encontrar en el circuito de la vivienda. Hay diferentes circuitos en la vivienda, cada uno tiene su propia intensidad máxima, por ejemplo: el circuito de iluminación 10A y el de cocina y horno 25A, por lo que el rango de corrientes que debe poder medir es de 0,1A a 25A. En caso de que el entorno fuera una industria, los niveles de corriente. Al tratarse de una aplicación escalable, y poder aplicarse a la industria, estos niveles de corriente se adaptarán a cada caso concreto de la industria.
- Compatibilidad: debe de poder adaptarse al elemento de control, tanto en la conexión como en el intercambio de información, para su fácil incorporación al sistema. La señal de tensión proporcional a la corriente medida, deberá estar comprendida entre 0V y 5V.
- Fiabilidad y precisión: los valores que se obtengan de las medidas deben ser lo más fiable posible, con una resolución mínima de 100mA.
- Ancho de banda: la frecuencia que se va a medir se encuentra entorno a 50Hz, por lo que como mínimo deberá tener un ancho de banda de 0-50Hz. Pero se va a considerar la frecuencia de 60Hz, debido a que es utilizada en otros países, esto le aporta más escalabilidad a la aplicación.
- Tamaño y robustez: debe de tener un tamaño pequeño para su fácil utilización e instalación en el circuito eléctrico de la vivienda, el espacio es reducido por lo que lo ideal sería como máximo 50x20x20mm, pudiendo de ser de mayor tamaño para entornos industriales. Además, debe ser duradero.
- Coste: teniendo en cuenta el presupuesto con el que se cuenta, se han de cumplir los máximos requisitos posibles de los elementos con este presupuesto. No superando los 10€.

2.3. INTERFAZ DEL USUARIO.

- **Interfaz del usuario**, es el medio por el que el usuario va a interactuar con la aplicación, mediante una interfaz web con las siguientes características:
  - Sencillez: en cuanto a su diseño y la programación.
  - Compatibilidad: debe de poder integrar aplicaciones IoT, trabajar con el protocolo de comunicación que se vaya a emplear.
  - Capacidad de comunicación: debe poder comunicarse con el elemento de control, para recibir los datos de consumo que este le facilitará. Y con la base de datos para enviarle los datos de consumo eléctrico.

2.4. BASE DE DATOS.

- **Base de datos**, es necesaria para almacenar los valores de consumo eléctrico que serán analizados posteriormente, para la tomar las decisiones correctas que lleven a alcanzar la eficiencia energética. Debe de cumplir los siguientes requisitos:
    - Capacidad de comunicación: debe poder comunicarse con los elementos del sistema para la recepción de la información a almacenar.
    - Estructura relacional: esto quiere decir que la forma en que la información es almacenada debe ser en forma de tabla, relacionando la información entre sí.

2.5. COMUNICACIONES.

- **Comunicaciones**, todos los elementos que están integrados en el sistema deben comunicarse entre sí, por lo que se va a necesitar un protocolo de comunicación para realizar esta tarea, cumpliendo los requisitos:
    - Basado en un modelo pub-sub: la estructura utilizada en aplicaciones IoT.
    - De fácil implementación en el sistema: funcional en elementos de potencia limitada.
    - Ampliamente utilizado y testeado en aplicaciones IoT.

3. ESTUDIO DE SOLUCIONES ALTERANTIVAS.

En los siguientes subapartados del estudio de soluciones alternativas, se van a abordar las diferentes opciones que se han tenido en cuenta y se podrían haber seleccionado para los distintos elementos que conforman el desarrollo del proyecto.

3.1. ELEMENTO DE CONTROL.

En primer lugar, se aborda la cuestión del elemento de control, para el cual se han barajado diferentes posibilidades: PLC, ordenador industrial o microcontrolador, expuestos a continuación.

- **PLC** (Controlador Lógico Programable). Son elementos, utilizados para la automatización industrial, a nivel de control. Nacieron con la misión de gobernar, en tiempo real, máquinas o procesos lógicos secuenciales en ambientes industriales. Actualmente han evolucionado, y pueden tener

módulos para diferentes procesos, lo que les aportan gran versatilidad. Algunas de sus características más destacables son las siguientes:

- Diseñados para trabajar en el entorno industrial, gran protección contra el ruido electromagnético.
- Flexibilidad: ante posibles cambios en el proceso, son fácilmente reprogramables para cumplir las funciones necesarias.
- Estructura modular: fácilmente adaptables a las arquitecturas de los proyectos, además de ampliar sus prestaciones gracias a los módulos por los que está formado. En la figura 1, se puede observar la estructura modular de un PLC.
- Control muy preciso.
- Gran capacidad de comunicación, además de módulos específicos para interfaces de comunicación.

Los autómatas programables se pueden programar en una gran variedad de lenguajes, como lenguajes gráficos, de contactos, diagramas de escalera, diagramas de funciones lógicas… Son elementos empleados en aplicaciones como por ejemplo el control de cadenas de montaje, máquinas herramienta, envasado, empaquetado… Cabe destacar algunas debilidades de los autómatas programables, por ejemplo: la velocidad de adquisición de datos no es lo suficientemente rápida como para poder las señales de tensión que va a recibir de los sensores. Además, su capacidad para realizar cálculos matemáticos también es reducida. Tras estudiar las necesidades de la aplicación y sus características, llevaría a descartar su uso en la aplicación.

Figura 1. PLC modular marca Omron.

- **Ordenador industrial** (PC industrial). Son sistemas electrónicos de control basados en los ordenadores convencionales. A diferencias de estos, los PC industriales están preparados para trabajar y resistir al entorno industrial. Además de las características normales de un ordenador convencional (placa base, CPU, RAM, buses de comunicación…) poseen elementos para su posible uso en la industria, como, por ejemplo:
  - Envolvente de protección.
  - Puertos internos especiales como GPIB, SCSI (Small Computer System Interface) …
  - Comunicaciones en serie y Ethernet.
  - Tarjetas de comunicación a nivel de campo.
  - Tarjetas de E/S digitales y analógicas.

- Sistemas de almacenamiento de datos redundantes.

Existen PC de diferentes tipologías, grandes ordenadores de altas prestaciones, con software y hardware diseñados a medida; *Racks* PC's para su montaje modular; y *Box* PC's de menor tamaño como el de la figura 2..

Las características más importantes de este tipo de ordenadores son las siguientes:

- Robustez para trabajar en entornos industriales.
- Integran sistemas operativos multitarea.
- Gran calidad y poder de control. Así como manejo de grandes cantidades de datos.
- Muchas facilidades para la comunicación entre sistemas.
- Posibilidades de visualización, obtención de gráficas…
- Gran modularidad y facilidad de ampliación.

Figura 2. Interfaces de comunicación de un Box PC.

- **Microcontrolador.** Es un circuito integrado programable, compuesto por diferentes bloques, diseñado para ejecutar un programa almacenado en su memoria. Esta formado por los siguientes elementos:

  - CPU (Unidad Central de Procesos).
  - Memoria RAM (almacena datos).
  - Memoria ROM (almacena el programa).
  - Pines de E/S.
  - Módulos para el control de periféricos (temporizadores, relojes, conversores ADC o DAC…)

  Se pueden emplear en distintas aplicaciones, como, por ejemplo, la industria informática, sistemas de comunicación, domótica…
  Las principales características de los microcontroladores son las siguientes:

  - Cubren funciones muy específicas.
  - Soportes poco exigentes.
  - Pequeño tamaño.
  - Ideales para presupuestos bajos.

  Por lo general, suelen ser programados en el lenguaje ensamblador del microprocesador que incorpora, o con compiladores específicos. El

lenguaje más utilizado para su programación es C++, aunque en algunos casos en que el tiempo de respuesta de la aplicación no sea un factor importante, puede utilizarse JAVA.

3.2. SENSORES.

Con respecto a los elementos a utilizar para medir tensión, también se han barajado varias opciones, siendo las siguientes:

- **Divisor de tensión en modo diferencial**. Es un sistema utilizado para que, a partir de una tensión alta de entrada, obtener otras tensiones de salida más pequeñas. Destinados a proporcional una tensión de alimentación menor que la fuente de alimentación. Esto se logra, pasando la tensión por resistencias colocadas en serie, de forma que obtenemos un valor de tensión que es fracción del valor de entrada, de esta forma se puede trabajar con señales con las que el elemento de control pueda trabajar. Se emplean junto con un octoacoplador para conseguir un aislamiento galvánico. Son utilizados entre otras aplicaciones, en potenciómetros, lecturas de sensores resistivos, lecturas de valores de tensión…

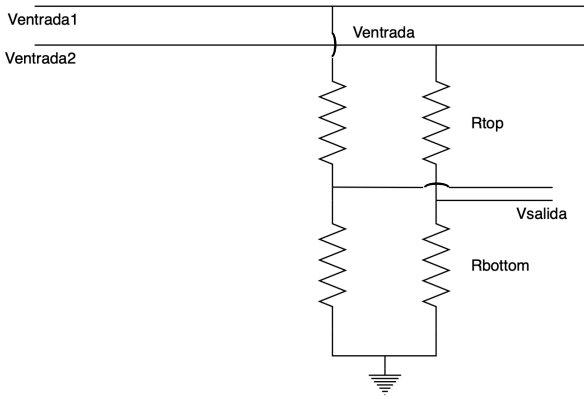


Figura 3. Divisor de tensión en modo diferencial.

- **Transformador de tensión.** Tienen diversos usos (disminuir pérdidas, distribución eléctrica…), pero en esta aplicación se emplearía para la medida de la tensión, adaptando el nivel de voltaje a un rango que el elemento de control pudiera manejar. Es un sistema sencillo y ampliamente utilizado para realizar medidas de tensión. No necesita de octoacoplador ya que el transformador ya supone un aislamiento en sí mismo. Suelen emplearse en medición de voltaje AC, monitoreo remoto y protección de equipos AC, retroalimentación para sistemas de control de voltaje AC entre otras aplicaciones.

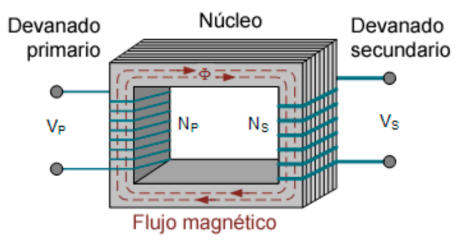


Figura 4. Esquema de un transformador de tensión

Los elementos que se han estudiado para realizar las medidas de la corriente eléctrica se exponen a continuación:

- **Transformador de corriente.** Construido en forma de pinza amperimétrica son sensores no invasivos que miden la corriente que atraviesa un conductor, esta medición se realiza por inducción electromagnética. Está formado por un núcleo ferromagnético dividido en dos partes ya que realiza la función de pinza. Puntos a tener en cuenta en este sensor:
    - Un pequeño hueco de aire en el cierre puede introducir variaciones de hasta un 10%.
    - Al ser una carga inductiva introduce un desfase que puede llegar a los 3º.
    - Los valores de salida pueden ser tanto positivos como negativos. Midiendo solamente corriente alterna.
    - La conexión de salida es de tipo jack, por lo que habría que sumar otros elementos para hacer posible la conexión.

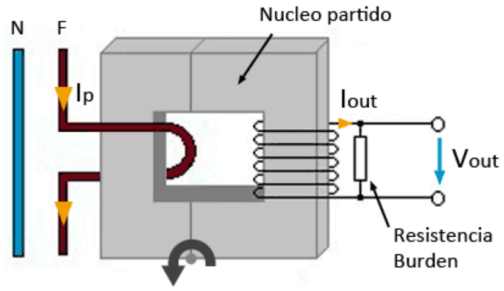


Figura 5. Esquema conexión pinza amperimétrica..

- **Sensor de efecto hall.** Este sensor esta formado por un circuito de bajo offset, con sensor Hall, posee una pista de cobre por la que fluye una corriente que genera un campo magnético detectado por el circuito integrado Hall y convertida en una tensión proporcional. A diferencia de los anteriores, estos son capaces de medir tanto corriente alterna como corriente continua. Unas de las características más significativas de estos sensores son las siguientes:
    - El espesor de la pista de cobre permite la resistencia del dispositivo en hasta 5 eventos de sobre corriente.
    - No introducen un desfase en la salida.
    - Viene en módulos, una bornera para la línea a medir y tres pines para la alimentación y la salida analógica, facilitando su conexión al sistema.
    - Fácil de instalar y tamaño reducido.
    - Detectan campos magnéticos estáticos.
    - Baratos.

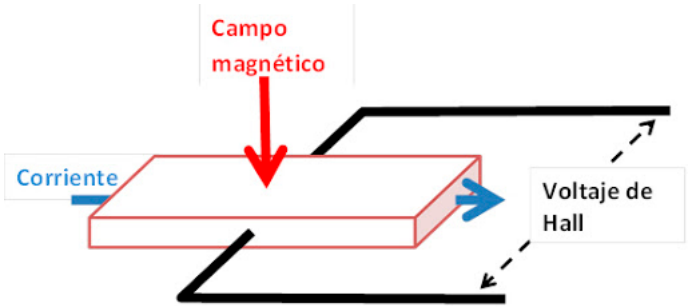


Figura 6. Esquema efecto hall..

3.3. ELEMENTO INTERFAZ DEL USUARIO.

En cuanto a la interfaz del usuario:

- **HMI.** Este tipo de pantallas son muy útiles para acceder rápidamente a la información en el lugar en que se ha realizado la instalación. Pero poco útiles si se quiere acceder a la información estando fuera de la vivienda o industria.

- **Página web.** Empleando una página web, se puede acceder a la información desde cualquier lugar. Simplemente accediendo a la web desde el dispositivo se puede consultar la información deseada.

No son opciones incompatibles, ya que cada una tiene su aplicación, si se deseará, también se podría instalar una pantalla HMI.

3.4. LENGUAJE PROGRAMACIÓN INTERFAZ DEL USUARIO.

Para el lenguaje o entorno de desarrollo de la interfaz con el usuario se han barajado varias opciones:

- **HTML *(HyperText Markup Language).*** Hace referencia al lenguaje de marcado para la realización de páginas web. Es el lenguaje que define el contenido de las páginas web, Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y los diversos elementos que la componen: imágenes, listas, vídeos… Sus características más destacables son las siguientes:
  - Este lenguaje se programa en documentos de texto, guardado con la extensión .html (Notepad++, textedit…).
  - Está organizado por etiquetas, que abre y cierran apartados y su contenido esta entre ambas etiquetas de un mismo grupo.
  - Tiene una estructura clara y es fácilmente comprensible. Está dividido en dos bloques *"<head>"* y *"<body>"*.
  - No es dinámico, aunque hay una versión llamada DHTML que sí lo es.

- **Java.** es un lenguaje de programación orientado a objetos que se incorporado al ámbito de la informática en los años 90. Diseñado para que puedan realizarse programas con la posibilidad de ejecutarse en cualquier sistema operativo, en cualquier ambiente, siendo así su portabilidad uno de sus principales logros.
  - Simple, es un lenguaje parecido a C++, facilitando su aprendizaje.
  - Extensas capacidades de interconexión TCP/IP.
  - Robusto, realiza verificaciones y comprobaciones de código y *ByteCodes* para asegurar el funcionamiento de la aplicación.
  - Arquitectura neutral para poder ser ejecutado desde cualquier máquina.

- Seguro, el código pasa por muchas pruebas y verificaciones para comprobar su seguridad.
- Portable, puede ser ejecutado en diferentes sistemas operativos (Linux, PC, Mac…).
- Dinámico, las librerías nuevas no paralizan la ejecución de las aplicaciones actuales.

- **JavaScript.** Es un lenguaje de scripts interpretado dinámico usado para crear páginas web dinámicas, incorporando, efectos, animaciones, acciones para el usuario... Hoy en día todos los navegadores pueden interpretar el código JavaScript de las páginas web. Es utilizado del lado del cliente, ejecutándose en nuestro ordenador y no en el servidor. Este lenguaje se integra en el HTML.

- **Node-Red.** Es una herramienta de programación visual que sirve para conectar diferentes elementos hardware, aplicaciones software y APIs, diseñada para la agilización de procesos IoT. Así mismo, ha sido diseñada para facilitar el diseño de interfaces web para usuarios con escasos conocimientos de programación web. Muestra visualmente las relaciones y funciones, y permite que usuario programe sin tener que utilizar lenguajes de programación como HTML, Java, entre otros. Node-Red es un editor de flujo basado en el navegador donde conectando una serie de nodos entre sí conseguimos que se comuniquen entre ellos.

  - Herramienta de código abierto.
  - Sencillez y facilidad de uso.
  - Permite utilizar tecnologías complejas de forma sencilla.
  - Herramienta muy ligera, puede ejecutarse en dispositivos limitados, por ejemplo, desde una Raspberry hasta en IBM Bluemix.
  - Facilidad de desarrollo y portabilidad de flujos, pudiendo aprovechar nodos desarrollados por terceros.

3.5. INTERFAZ DE COMUNICACIÓN.

Para la comunicación entre los diferentes elementos que integran la aplicación se han propuesto las siguientes opciones, ambas son inalámbricas, puesto que se quiere priorizar este tipo de conexión frente a cableadas, para de esta forma no tener que acceder continuamente al cuadro eléctrico.

- **Bluetooth.** Como punto a favor de este medio de comunicación cabe destacar su modo Low Energy, reduciendo considerablemente el consumo sin perder rango de alcance. Con gran rapidez de conexión tras estar en modo suspensión, pero siendo más lento en la transmisión de datos y el número de estos. Aunque, en modo Bluetooth normal, se pueden transmitir muchos datos, pero con un gran consumo de energía.

- **Wifi.** Es la interfaz de comunicación más empleada en aplicaciones IoT, debido a su velocidad de transmisión, 347Mbps en bajo consumo y hasta 1,3Gbps en estándar. Puede transferir información pesada como imágenes o videos. En cuanto a seguridad, las redes Wifi cuentan con protocolos de seguridad estándar, por lo general superiores a los Bluetooth.

3.6. PROTOCOLO DE COMUNICACIÓN.

Para cumplir con las necesidades del proyecto en el aspecto de las comunicaciones, se han planteado tres protocolos de comunicación:

- **AMQP (Advanced Message Queuing Protocol).** Es un protocol PubSub de Message Queue, diseñado para asegurar la confiabilidad e interoperabilidad. Destinado a aplicaciones con mayor rendimiento y redes de baja latencias. Aunque no resulta muy adecuado para aplicaciones IoT en dispositivos de baja capacidad.

- **HTTP (Protocolo de Transferencia de Hipertexto).** Es un protocolo que sigue un modelo unidireccional, cliente-servidor, el cliente solo puede recibir información en caso de realizar una petición, nunca de forma pasiva. Es un protocolo asíncrono y pesado por lo que no es ideal para redes de altas latencias. No es una buena opción como protocolo para aplicaciones IoT.

- **MQTT (Message Queue Telemetry Transport).** es un protocolo de mensajería asíncrona, usado para la comunicación machine-to-machine (M2M) en el ámbito de IoT. Tiene una estructura PubSub de Message Service que actúa sobre TCP. Destaca por ser ligero y sencillo de implementar. Trabaja perfectamente en dispositivos de baja potencia, frecuentemente utilizados en aplicaciones IoT. Está optimizado para trabajar con un gran número de clientes conectados de forma simultanea.

4. DETALLE DE LA SOLUCIÓN ADOPTADA.

Para poder llegar a ser eficiente energéticamente, es imprescindible conocer los datos de consumo que se están realizando. En una industria se podría saber si se está aprovechando toda la energía, la optimización de horarios, o decidir realizar cambios de maquinaria, como por ejemplo motores eléctricos, ya que la tecnología ha avanzado considerablemente, que puedan repercutir en un menor consumo energético. En una vivienda, conocer detalladamente el consumo energético que se está realizando puede llevar a cambiar las luminarias de la vivienda, renovar electrodomésticos, o tener un contrato por discriminación horaria, en el cual, en las horas de mayor consumo, el coste de la energía sea menor que en las de menor consumo. En conclusión, tener esta información puede llevar a tomar acciones que repercutirán en un ahorro económico y energético.

En la figura 7 se puede ver, el pasos que hay que seguir tras el estudio de necesidades y soluciones alternativas para el completo desarrollo de la aplicación.

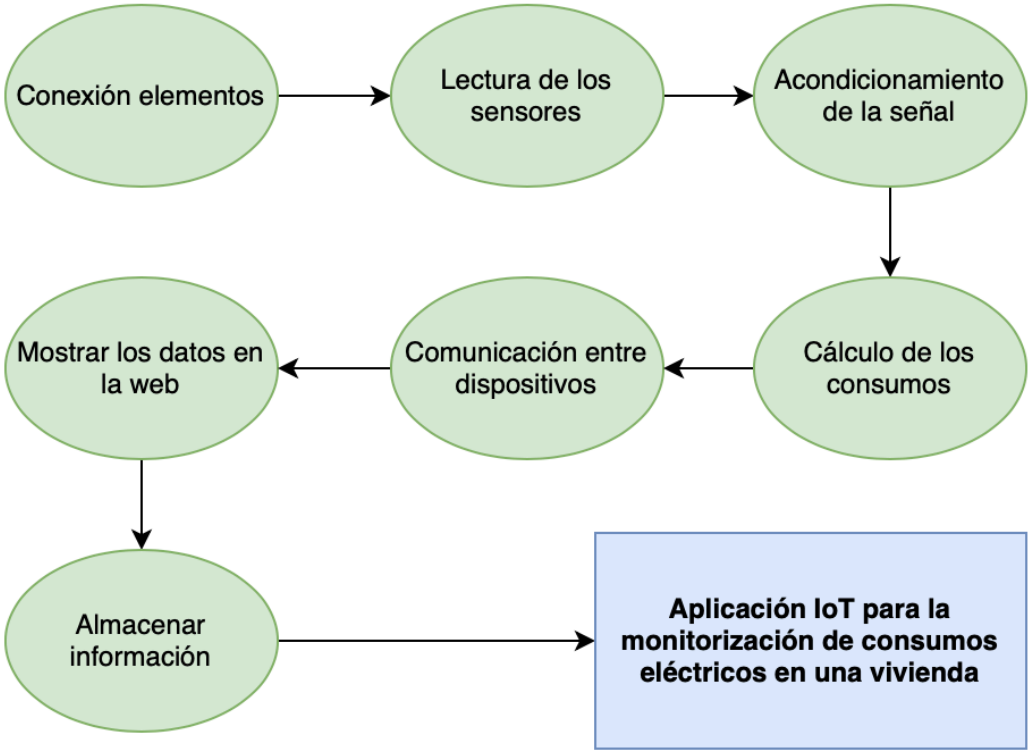


Figura 7. Pasos para alcanzar el objetivo fundamental

A continuación, se procede a detallar los distintos elementos seleccionados, de los anteriormente planteados.

4.1. MICROCONTROLADOR.

En primer lugar, se ha decidido emplear como elemento de control un microcontrolador, debido a su pequeño tamaño, reducido coste además de cumplir con todos los requisitos que establece el proyecto en cuanto a potencia, memoria, interfaces, conexión Wifi y capacidad de comunicación. Por lo que se ha descartado el uso de PLC o PC industrial. De entre todos los microcontroladores que hay en el mercado se ha optado por la familia ESP32 de la empresa Espressif Systems. A continuación, se presentan los diferentes tipos de chips, módulos y placas de desarrollo que se encuentran en esta familia, cual ha sido el elegido para la aplicación desarrollada.

En la Tabla 1, se puede diferenciar entre cada uno de los chips el tipo de núcleo que tienen para poder determinar cual es el que se ajusta a las necesidades de la aplicación.

4.1.1. Chips ESP32.

SoC	Core	Package (mm)	Pines	RAM (kB)	Flash (MB)	Modulo	Placa de Desarrollo
ESP32-D0WD-V3	Dual Core	QFN 5x5	48	520kB SRAM 448kB ROM 16kB SRAM in RTC	N/A	ESP32-WROOM-32E ESP32-WROOM-32UE ESP32-WROVER-E ESP32-WROVER-IE	ESP32-DevKitC ESP32-Vaquita-DSPG
ESP32-D0WD	Dual Core	QFN 5x5	48	520kB SRAM 448kB ROM 16kB SRAM in RTC	N/A	ESP32-WROOM-32D ESP32-WROOM-32U ESP32-WROVER-B ESP32-WROVER-IB	ESP32-DevKitC ESP32-LyraT ESP32-LyraT-Mini ESP32-LyraTD-MSC ESP32-LyraTD-DSPG ESP32-LyraTD-SYNA ESP32-Korvo ESP-WROVER-KIT
ESP32-D0WDQ6-V3	Dual Core	QFN 6x6	48	520kB SRAM 448kB ROM 16kB SRAM in RTC	N/A	N/A	N/A
ESP32-D0WDQ6	Dual Core	QFN 6x6	48	520kB SRAM 448kB ROM 16kB SRAM in RTC	N/A	ESP32-WROOM-32 ESP32-WROVER ESP32-WROVER-I	ESP32-DevKitC
ESP32-D2WD	Dual Core	QFN 5x5	48	520kB SRAM 448kB ROM 16kB SRAM in RTC	2	N/A	N/A

ESP32-U4WDH	Single Core	QFN 5x5	48	520kB SRAM 448kB ROM 16kB SRAM in RTC	4	N/A	N/A
ESP32-S0WD	Single Core	QFN 5x5	48	520kB SRAM 448kB ROM 16kB SRAM in RTC	N/A	ESP32-SOLO-1	ESP32-DevKitC
ESP32-PICO-V3	Dual Core	LGA 7x7	48	448KB ROM 520KB SRAM 16KB RTC SRAM	4	ESP32-PICO-V3-ZERO	ESP32-PICO-KIT
ESP32-PICO-D4	Dual core	LGA 7x7	48	448KB ROM 520KB SRAM 16KB RTC SRAM	4	N/A	ESP32-PICO-KIT

Tabla 1. Especificaciones de los chips de la familia ESP32

Según la nomenclatura de cada chip se pueden saber sus características, por ejemplo:

ESP32	**-**	**D**	**2**	**WD**
		D: Dual Core S: Single Core	2: 16Mbit 0: No Embedded Flash	WD: WiFi b/g/n + BT/BLE Dual Mode AC: WiFi a/b/g/n + BT/BLE Dual Mode CD: WiFi ac/c/b/g/n + BT/BLE Dual Mode

4.1.2. MÓDULOS ESP32.

A continuación, se presentan los distintos tipos de módulos de la familia ESP-32, que, por lo general tienen la siguiente forma, variando tamaños mayormente:

Figura 8. Módulo ESP-WROOM-32

ESP32-WROOM-32E

Este módulo integra el chip ESP32-D0WD-V3, de alta estabilidad y funcionamiento seguro. Tiene un tamaño de 18x25.5x3.1 mm, 38 pines y una memoria flash de diferentes tamaños, 4,8 o 16MB. Integra una antena PCB y se integra en la placa de desarrollo ESP32-DevKitC

ESP32-WROOM-32UE

Al igual que el anterior modulo, este también integra el chip ESP32-D0WD-V3. Las únicas diferencias que tienen son, en primer lugar el tamaño, ya que este es un poco más pequeño (18x19.2x3.2 mm) y la antena, ya que este monta una antena IPEX.

ESP32-WROOM-32D

Este módulo integra el chip ESP32-D0WD. Está pensado para una amplia variedad de aplicaciones, por ejemplo, desde redes de sensores de baja potencia hasta tareas exigentes como puede ser retransmisión de música o decodificación de MP3. Tiene un tamaño de 18x25.5x3.1mm, 38 pines, una memoria flash de 4, 8 o 16MB y una antena PCB. Este módulo también se monta en la placa de desarrollo ESP32-DevKitC.

ESP32-WROOM-32U

Este módulo integra el chip ESP32-D0WD y además integra un conector U.FL. (conector de frecuencia que es utilizable para señales de alta frecuencia de hasta 6GHz de banda, suelen usarse para conectar antenas GPS, minitarjetas PCI, y sobre todo para aquellas aplicaciones en las que el espacio es reducido, debido a que tiene pequeño tamaño). Tiene un tamaño de

18x19.2x3.2 mm, 38 pines, y memoria flash de 4, 8 o 16MB. Integra una antena IPEX y también es integrado en la placa de desarrollo ESP32-DevKitC.

ESP32-WROOM-32SE

Este módulo integra el chip ESP32-D0WD, además tiene un chip ATECC608A incorporado, que actúa como un almacenamiento seguro para certificados de dispositivos. Tiene un tamaño de 18x25.5x3.1 mm, 38 pines, y solo un tamaño de memoria flash, 4MB. Integra una antena PCB y es igualmente integrado en la placa de desarrollo ESP32-DevKitC.

ESP32-WROOM-32

Este módulo contiene el SoC ESP32, con un tamaño de 18x25.5x3.1 mm, 38 pines, memoria flash de 4MB. Tiene componentes discretos de alta precisión y una antena de PCB que proporciona un excelente rendimiento de RF en aplicaciones con espacio limitado. Este módulo es integrado en la placa de desarrollo ESP32-DevKitC. Este es el módulo que se ha utilizado para la realización de este proyecto.

ESP32-WROVER-E

Este módulo integra el chip ESP32-D0WD-V3, de alta estabilidad y funcionamiento seguro. Tiene un tamaño de 18x31.4x3.3 mm, 38 pines, una memoria flash de 4, 8 o 16MB. Y sobre todo, a gran diferencia de los wroom, frente a los wrover se encuentra en que estos segundos tienen una PSRAM de 8MB. También integra una antena PCB y son integrados en las placas de desarrollo ESP32-DevKitC, ESP32-Vaquita-DSPG y ESP32-Korvo.

ESP32-WROVER-IE

Tiene las mismas características que el anterior, integrando el mismo chip, tamaño, memoria flash, PSRAM… se diferencian en que este integra una antena IPEX y que solo es integrado en la placa de desarrollo ESP32-DevKitC.

ESP32-WROVER-B

Este es un potente y genérico módulo WiFi-BT-Bluetooth LE MCU pensado para una gran variedad de aplicaciones, que van desde redes de sensores de baja potencia hasta tareas más exigentes, como por ejemplo, codificación de voz o transmisión de música. Monta un chip ESP32-D0WD, tiene un tamaño de 18x31.4x3.3mm, memoria flash de 4, 8 o 16MB, PSRAM de 8MB, una antena PCB y puede ser integrado en numerosas placas de desarrollo:
ESP32-DevKitC, ESP32-LyriaT, ESP32-LyriaT-Mini, ESP32-LyriaTD-MSC, ESP32-LyriaTD-DSPG, ESP32-LyriaTD-SYNA, ESP-WROVER-KIT-VB y ESP-WROVER-KIT.

ESP32-WROVER-B

Tiene las mismas características que el anterior a diferencia de el, monta una antena IPEX y se integran en dos placas de desarrollo, la ESP32-DevKitC y la ESP-WROVER-KIT-VB.

ESP32-WROVER

Este módulo integral el chip ESP32-D0WDQ6, además presenta una memoria flash SPI externa de 4 MB y una PSRAM externa de 8 MB, por lo que se dirige a una amplia variedad de aplicaciones. Tiene un tamaño de 18x31.4x3.3 mm e integra una antena PCB.

ESP32-WROVER-I

Es muy parecido al anterior, se diferencian en que este usa una antena IPEX en vez de la antena PCB.

ESP32-PICO-V3-ZERO

Este es un módulo basado en el chip ESP32-PICO-V3, un dispositivo de sistema en paquete (SiP). Presenta funcionalidades completas de Wi-Fi y Bluetooth. El módulo integra una memoria flash SPI de 4 MB, con tamaño muy pequeño (16×23×2.3 mm y 77 pines), rendimiento robusto y bajo consumo de energía. Con una antena PCB.

ESP32-SOLO-1

Este es un potente y genérico módulo WiFi-BT-Bluetooth LE MCU pensado para una gran variedad de aplicaciones, que van desde redes de sensores de baja potencia hasta tareas más exigentes, como, por ejemplo, codificación de voz o transmisión de música. Monta un chip ESP32-D0WD, tiene un tamaño de 18x25.5x3.1mm, 38 pins, 4MB de memoria flash y antena PCB, este módulo es integrado en la placa de desarrollo ESP32-DevKitC.

4.1.3. PLACAS DE DESARROLLO ESP32.

A continuación, se comentan las placas de desarrollo más importantes dentro de la familia ESP-32.

ESP-EYE

Esta, es una placa de desarrollo para el reconocimiento de imágenes y el procesamiento de audio. Está formada por un chip ESP32, una cámara de 2 megapíxeles y un micrófono. ESP-EYE ofrece mucho almacenamiento, con una PSRAM de 8 MB y una memoria flash de 4 MB. También es compatible con la transmisión de imágenes a través de WiFi y a través de un puerto Micro-USB.

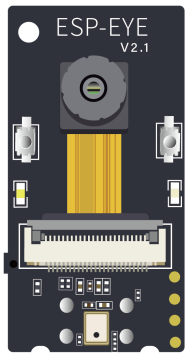


Figura 9. ESP-EYE

ESP32-DevKitC

Esta es la placa de desarrollo de nivel de comienzo, es la más usada. Tiene todos los pines del ESP32 expuestos por lo que es fácil de conectar y de usar. Tiene 4MB de memoria flash, la interfaz de conexiones son pines de entra y salida y un puerto USB que sirve para la alimentación de la placa, además posee dos botones (boot y reset) y un led interno. Esta es la placa de desarrollo que se empleará para realizar la aplicación.

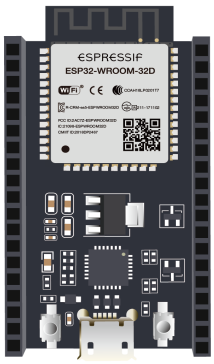


Figura 10. ESP32-DevKitC

ESP32-PICO-KIT

Es la placa de desarrollo más pequeña de Espressif Systems. Es completamente funcional con el número mínimo de componentes discretos, tiene todos los pines ESP32 expuestos. Cuenta con el mismo tamaño de memoria flash y de interfaces que el módulo ESP32-DevKitC.

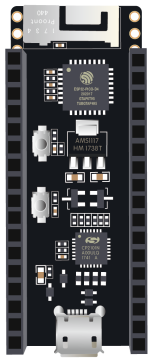


Figura 11. ESP-PICO-KIT

ESP32-Korvo

Es una placa de desarrollo de audio con matriz de micrófonos, junto con el SDK ESP-Skainet de reconocimiento de voz desarrollado por Espressif Systems. Esta placa es adecuada para aplicaciones de reconocimiento de voz de campo lejano con bajo consumo de energía. Está compuesto por dos placas

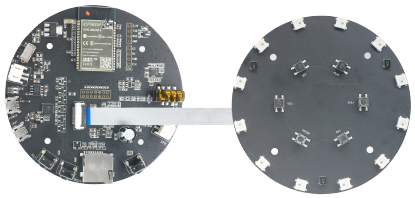


Figura 12. ESP-Korvo

conectadas por un cable FPC. La placa principal contiene el módulo ESP32-WROVER-E, el puerto de alimentación, la ranura para tarjeta micro SD, auriculares y conectores de altavoces. La placa secundaria contiene una matriz de micrófonos, botones de función y LED. Tiene 16MB de memoria flash, y 8 de PSRAM, botones y leds internos.

ESP32-LyraT

Esta placa está diseñada para el reconocimiento de voz. Integra el módulo ESP32-WROVER-B, el cual integra un procesador de doble núcleo y 4,5 MB de memoria operativa. Para poder implementar una solución de audio altamente integrada, se requieren pocos dispositivos periféricos. Tiene 4MB de memoria flash y 4MB de PSRAM, además posee ranura para tarjeta microSD, USB, altavoces, botones y leds internos.

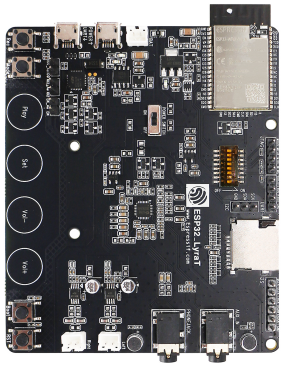


Figura 13. ESP-LyraT

ESP32-Vaquita-DSPG

Esta placa es la nueva solución incorporada de Alexa de Espressif Systems, impulsada por ESP32 y el SoC de audio DBMD5P de DSP Group. Incluye 2 micrófonos que permite la captación en 360 grados, que proporciona un rendimiento superior de reconocimiento de voz. Esta placa es una solución para crear fácilmente dispositivos conectados incorporados de Alexa que proporcionan habilitación de voz lista para usar y conectividad en la nube AWS-IoT.

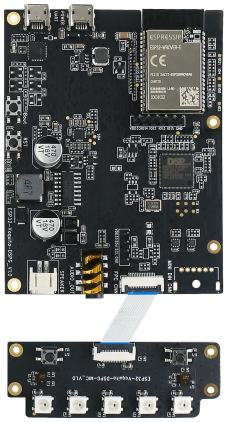


Figura 14. ESP-Vaquita-DSPG

ESP-WROVER-KIT-VB

Esta placa es una variante del ESP-WROVER-KIT que integra el módulo ESP32-WROVER-B. Cuenta con soporte para una pantalla LCD y tarjeta micro SD. Los pines de I/O se han separado del kit para tener fácil extensión. La placa integra un puente USB multiprotocolo avanzado (FTDI FT2232HL), que permite a los desarrolladores usar JTAG directamente para limpiar el módulo ESP32 a través del USB.

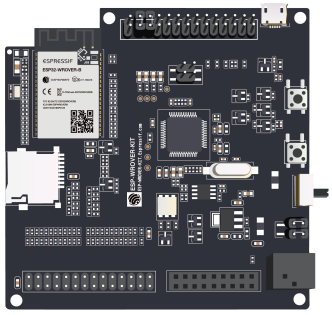


Figura 15. ESP-WROVER-KIT-VB

Placa de desarrollo	Descripción	Core	Memoria FLASH/PSRAM	Interfaces	UI	Wifi	E/S	Precio
ESP-EYE	Reconocimiento de imagen y procesamiento de audio	Dual Core	4 MB Flash + 8 MB PSRAM	E/S, USB	Buttons, LEDs	2.4 GHz Wi-Fi	48	17,91 €
ESP32-DevKitC	Placa base de desarrollo con pines de conexión expuestos	Dual Core	4 MB Flash	E/S, USB	Buttons, LEDs	2.4 GHz Wi-Fi	48	9,99 €
ESP32-Pico-Kit	Placa más pequeña desarrollada, con el mínimo número de componentes posibles	Dual Core	4 MB Flash	E/S, USB	Buttons, LEDs	2.4 GHz Wi-Fi	48	11 €
ESP32-Korvo	Placa de desarrollo de audio con matriz de micrófonos, junto con SDK ESP-Skainet de reconocimiento de voz.	Dual Core	16 MB Flash + 8 MB PSRAM	USB, Earphone jack, Speaker, Micro SD Card	Buttons, Speaker, LEDs	2.4 GHz Wi-Fi	48	31,50 €
ESP32-LyraT	Esta placa está diseñada para el reconocimiento de voz	Dual Core	4 MB Flash + 4 MB PSRAM	Micro SD Card, Audio, Output, USB, Speaker	Touch Buttons, Keys, LEDs	2.4 GHz Wi-Fi	48	18 €
ESP32-Vaquita-DSPG	Esta placa es la nueva solución incorporada de Alexa de Espressif Systems, reconocimiento de voz 360 grados.	Dual Core	16MB Flash + 8MB PSRAM	USB, 2xDMIC, input, Earphone jack, Speaker	5x RGB LEDs, 2x Function Keys	2.4 GHz Wi-Fi	48	40,50 €
ESP-WROVER-KIT-VB	Cuenta con soporte para una pantalla LCD y tarjeta micro SD. Los pines de I/O se han separado del kit para facil extensión.	Dual Core	4 MB Flash + 8 MB PSRAM	E/S, USB, JTAG, Camera, UART, SPI, Micro SD Card	LCD Screen, Buttons, RGB LED	2.4 GHz Wi-Fi	48	43,56 €

Tabla 2. Especificaciones de las placas de desarrollo de la familia ESP32

Como se ha podido observar en el anterior apartado y en la Tabla 2 de una forma más gráfica, las diferentes placas de desarrollo de Espressif están pensadas cada una para ser utilizadas en un tipo de aplicación, según las necesidades. Tras comprobar que la placa de desarrollo ESP32-DevKitC cumple con los requisitos establecidos en el proyecto, por ejemplo, en cuanto a los requisitos de memoria de carga (4MB), entradas analógicas para los sensores, también posee interfaz de comunicación Wifi (además Bluetooth), un tamaño reducido y un procesador suficientemente potente para llevar a cabo la aplicación, con un tamaño perfecto para la instalación en el entorno de la vivienda, y todo esto a un precio que se ajusta perfectamente a las necesidades.

4.1.4. PLATAFORMAS DE DESARROLLO.

Hay varios entornos y lenguajes de programación con los que se puede tratar el ESP32, voy a comentar algunos, pero me centraré en el que ha desarrollado Espressif Systems, ESP-IDF (Espressif IoT Development Framework) y el que he empleado yo en la aplicación Arduino IDE (Arduino Integrated Development Enviroment). Otros son, por ejemplo: entornos como Duktape (que corre motores de JavaScript), MicroPython IDE, o LUA-NodeMCU, PlatformIO.

4.1.4.1. ESP-IDF.

Es uno de los entornos de desarrollo analizados que nos da más facilidades y recursos para desarrollar aplicaciones en el ESP32, esto se debe a que está desarrollado por el propio fabricante, como he comentado antes. Se basa en lenguaje C/C++.

Podemos encontrar mucha documentación sobre el funcionamiento o funciones preestablecidas en la página web del Espressif.

Tenemos dos formas de utilizar esta herramienta: desarrollar los programas en un terminal con la instalación del toolchain correspondiente o instalar un software de terceros como Eclipse. En el primer caso nos encontramos con el inconveniente de que el proceso de compilar y cargar el programa en la memoria flash es muy lento. Como punto a favor, desde el terminal podemos acceder a la configuración de funciones como la encriptación, la seguridad… de forma que podemos cambiarlas según nuestras necesidades.

En cuanto a instalar el Eclipse, este es un entorno de código abierto, si lo configuramos de forma correcta, el desarrollo, compilación y cargar el programa se realiza de forma muy sencilla, esta suele ser la forma que más se recomienda.

4.1.4.2. Arduino IDE:

Una de las mayores ventajas de Arduino es su facilidad de uso tanto en su interfaz como en la programación y la gran comunidad que tiene detrás, en la que puedes encontrar mucha ayuda, consejos a la hora de estar trabajando con diferentes dispositivos, sobre todo cuando se ha estado muy familiarizado con la programación. Junto con ESP-IDF son los dos entornos más utilizados para el desarrollo de aplicaciones con el ESP32. Al igual que el entorno de Espressif, Arduino IDE también utiliza C/C++.

Además, las funciones que se aplican en ESP-IDF, son compatibles con Arduino, ya que fue Espressif quien desarrolló Arduino Core para trabajar con el ESP32, de forma que podemos sacarle todo el partido y potencial al dispositivo.

Una vez trabajando en el entorno, podemos configurar tanto el puerto, como la frecuencia, la velocidad del monitor serie (en el que vemos lo que nos está devolviendo el ESP32 o cualquier otro dispositivo que estemos utilizando), entre otros aspectos. Tiene una interfaz muy sencilla con pocos botones, destinados a compilar, subir, abrir o guardar los proyectos, debajo de la ventana de programación, vemos un espacio en el que nos saldrán los errores o si hemos tenido éxito a la hora de cargar el programa. Además, se pueden encontrar muchas librerías a mano que podemos utilizar en los proyectos.

No hay grandes diferencias entre programar con Arduino IDE o ESP-IDF. Este segundo, está enfocado para los productos de Espressif, por lo que aquello que se desarrolle solo podrá ser utilizado en productos de Espressif Systems. En cambio, desarrollando el programa con Arduino IDE, si se deseara cambiar de microcontrolador, por necesidades de proyecto o cualquier problema que surgiera, el código podría ser utilizado sin mayores problemas. Por ello y por el previo conocimiento de este entorno de programación se ha decidido por desarrollar la aplicación con Arduino.

A continuación, se muestra una imagen de la interfaz de Arduino IDE.

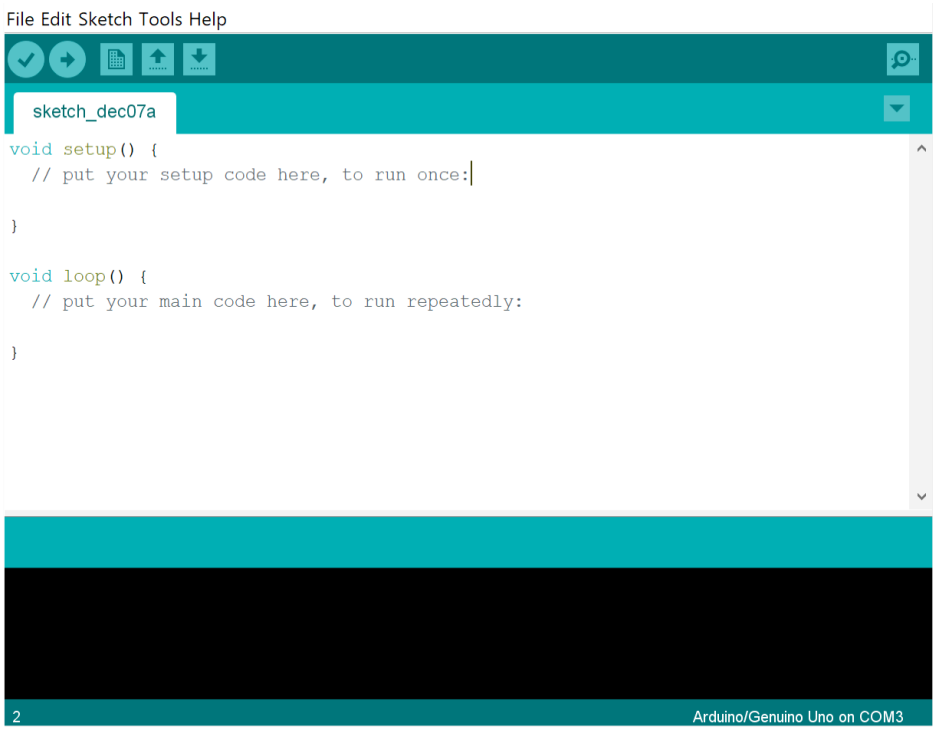


Figura 16. Interfaz Arudino IDE

4.2. SENSORES.

4.2.1. Sensor de tensión.

Para la medición de la tensión se va a emplear un transformador de tensión, ya que es un elemento muy utilizado en este tipo de aplicaciones, fácil de instalar y de adaptar al sistema, con una conexión sencilla con el microcontrolador y con mucha documentación sobre la utilización de transformadores en aplicaciones para la medición de tensión. Se empleará un **ZMPT101B**, capaz de trabajar con hasta 250V y que devuelve una señal de tensión adaptada al microcontrolador. También cumple con las necesidades de precisión, ya que cuenta con una precisión de 0.2%, con los requisitos de tamaño y de sencillez de implementación en el sistema. Cumpliendo además los requisitos de precio, siendo su coste de 6,99€.

4.2.2. Sensor de corriente.

Para la medición de las corrientes de los circuitos de la vivienda. Se ha decidido emplear sensores de efecto Hall, ya que es un tipo de sensor duradero, con una instalación sencilla y adaptación de la señal al microcontrolador. No produce desfases en la medida y se ajusta a los requisitos de tamaño y precio. Se va a utilizar un sensor **ACS712-30A**. Capaz de trabajar a 230V, con corrientes máximas de 30A, suficiente para mediciones en una vivienda, además cuenta con una sensibilidad de 66mV, y un ancho de banda de 80kHz. También cumple con los requisitos de tamaño y de precio, siendo su coste de 3,35€. Introduciendo por las entradas analógicas del microcontrolador una señal de tensión proporcional a la corriente medida, adaptando la señal perfectamente al sistema. Posteriormente se explicará el funcionamiento y características de este sensor.

4.3. INTERFAZ DEL USUARIO.

Se ha descartado el uso de la pantalla HMI como interfaz del usuario, ya que como se ha comentado en el apartado 3.3. la interfaz web aporta más funcionalidades a la aplicación, pudiendo consultar el estado de los consumos desde cualquier lugar. Por eso se ha seleccionado la opción de realizar una interfaz web.

Respecto al código de programación, en un primer lugar se planteo la posibilidad de desarrollarla en código HTML, pero por la carencia de conocimientos de este tipo de lenguaje y programación se llegó a una interfaz muy simple que necesitaba ser más desarrollada. Por lo que, tras un estudio de otras opciones se ha decidido utilizar Node-Red, ya que es una herramienta que simplifica mucho el trabajo, y que facilita el desarrollo de páginas web y comunicaciones para aplicaciones IoT. Es una herramienta muy sencilla de utilizar y trabajando en ella se puede alcanzar una interfaz óptima para la aplicación en cuestión. A continuación se explica como es el entorno Node-Red.

4.3.1. Node-Red

En primer lugar, es importante conocer qué es Node-Red y cómo funciona. Node-Red surge sobretodo para la agilización de procesos IoT. Es una herramienta de programación visual que sirve para conectar diferentes elementos hardware, aplicaciones software y APIs, diseñada para la agilización de procesos IoT. Muestra visualmente las relaciones y funciones, y permite que usuario programe sin tener que utilizar un lenguaje como pueden ser html, JavaScript entre otros. Node-RED es un editor de flujo basado en el navegador donde conectando una serie de nodos entre sí conseguimos que se comuniquen entre ellos. Esta unión se realiza mediante líneas de enlace que vienen definiendo el flujo de datos que se establece entre los nodos.

Existen tres tipos de nodos: indicadores, intermedios y terminales.

Los nodos indicadores se encargan de iniciar la ejecución del flujo de proceso, solamente tienen un puerto de salida y hay de tipo activo (revisan periódicamente la ejecución de un evento) y pasivo (están a la escucha de eventos, por ejemplo: una petición API).

Los nodos intermedios, se encargan de realizar tareas con la información que les llega y pasarlas al siguiente nodo, estas tareas pueden ser entre otras, cambio de formato, bases de datos, llamar a servicios externos, etc… tienen un puerto de entrada y una o varias salidas.

Los nodos terminales, son aquellos que están situados al final de las ramas del flujo de datos y se encargan de por ejemplo ejecutar otros flujos, realizar eventos de notificaciones o mensajería, llamar a servicios externos, entre otros. Solamente tienen un puerto de entrada.

Hay una gran cantidad de nodos, además, se pueden desarrollar los nuestros propios si así lo deseamos, por lo que también podemos encontrar nodos creados por otros usuarios y descargarlos para usarlos en diferentes proyectos.

A continuación, se muestra como es el espacio de trabajo de Node-Red:

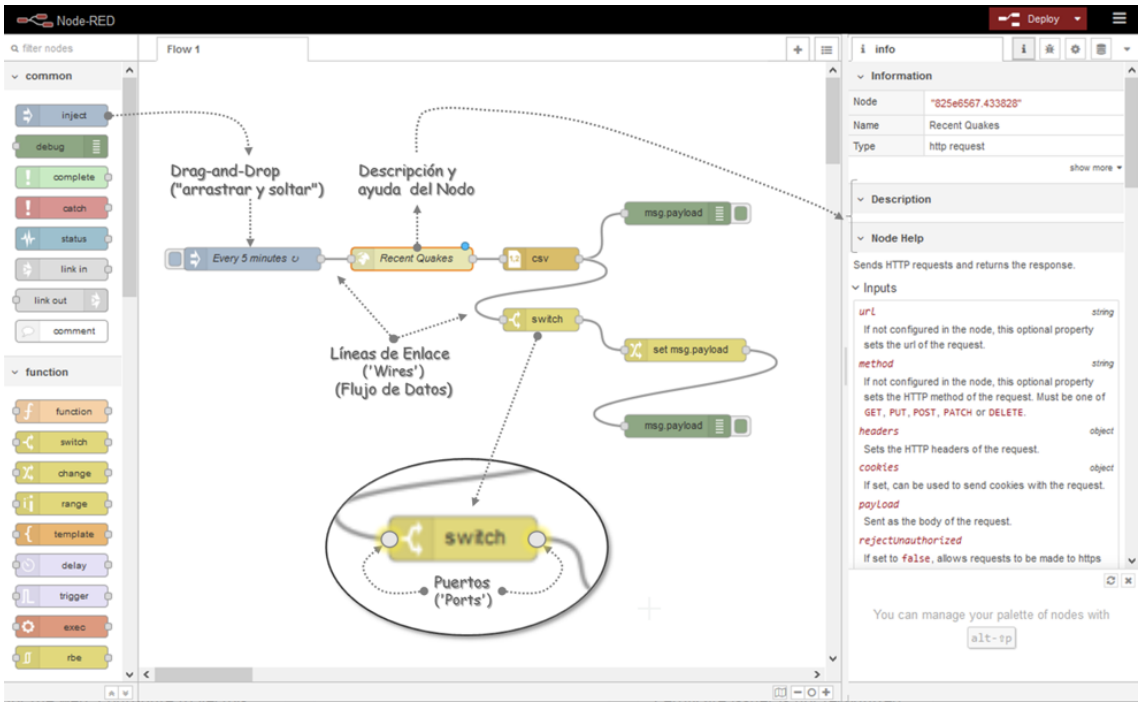


Figura 17. Espacio de trabajo de Node-Red.

4.4. PROTOCOLO DE COMUNICACIÓN.

Respecto al protocolo de comunicación a emplear, se ha tenido muy en cuenta su utilización en aplicaciones IoT por usuarios y referencias externas (blogs, expertos…). Tras comparar los diferentes protocolos comentados anterior mente y haber escuchado diferentes opiniones, se ha considerado que el protocolo MQTT es el mejor para este tipo de aplicaciones, debido a su facilidad de implementación, su ligereza y capacidad para trabajar con elementos de poca potencia, además de su estructura de publicación y subscripción es bidireccional por lo que a diferencia del protocolo HTTP, el intercambio de información pude ir en ambas direcciones entre los diferentes elementos que integran el sistema. Es de los protocolos más utilizado para realizar aplicaciones IoT. Más adelante se explica en detalle su funcionamiento y características.

4.5. BASE DE DATOS.

Se ha decidido emplear una base de datos relacional muy conocida y sencilla de utilizar e implementar en aplicaciones IoT. Es relativamente fácil realizar la comunicación con una interfaz web desarrollada en node red, de forma que los datos se muetren correctamente.

Desde el punto de vista industrial, esta misma aplicación podría aplicarse en ese entorno, ya que la finalidad y el funcionamiento es el mismo. La industria sería uno de los mayores beneficiados de esta aplicación, ya que el consumo de energía es grande y, por lo tanto, el ahorro económico y energético debido a los cambios realizados tras el análisis de los datos obtenidos de la monitorización de los consumos podría ser sustancial.

5. MATERIALES.

5.3. ESP32-WROOM-32.

El ESP32-WROOM-32 como se ha comentado, es el microcontrolador empleado para el desarrollo de la aplicación. Es el centro del sistema.

Las operaciones que va a realizar son las siguientes.:

En primer lugar, va a recibir los valores de tensión y corrientes de cada circuito de la vivienda por parte del sensor de tensión y los cinco sensores de corriente instalados (para ver la instalación, consultar el apartado de *PLANO 1. Conexión Eléctrica)*. El valor que recibe es una señal de tensión dentro de un rango limitado (de 0V a 5V), esta señal es proporcional al valor de la tensión y corrientes en el circuito eléctrico de la vivienda. Se conecta a un conversor analógico digital (ADC) con una resolución de 12 bits, que convierten la señal analógica a una señal digital. Dentro de la programación del microcontrolador, se encuentra la forma en que se pasa de esta señal proporcional al valor de tensión y corrientes real (apartados *7.3.3. Lectura del sensor de tensión* y *7.3.4. Lectura del sensor de corriente*).

Comentar, que la tensión de la vivienda se encuentra en 230V eficaces, que los cinco circuitos tienen su respectivo sensor de corriente y que para obtener la corriente total se sumarán las corrientes de cada circuito. La potencia activa en todo el circuito será el resultado de multiplicar la tensión por la corriente total.

Una vez obtenidos los valores de tensión, corrientes y potencia, se encargará de mandar estos valores a la interfaz web, los valores de tensiones y corrientes se actualizan cada segundo, pero los valores de consumo que se muestran en las curvas de carga se envían cada quince minutos, ya que recibiendo los valores de consumo cada este periodo de tiempo, aporta una

visión global del consumo cada hora, sin ser necesario conocer el consumo en cada instante de tiempo. Para ello, el microcontrolador, debe realizar la conexión a la red Wifi. Y al *broker* MQTT, que realiza de intermediario para que los mensajes lleguen al lugar correcto. Además, el ESP, se va a encargar de conocer la fecha y la hora en cada instante.

En la figura 10 se puede observar como es el ESP32 y su pin layout, en que se pueden diferenciar los diferentes tipos de pines de entrada que tiene.

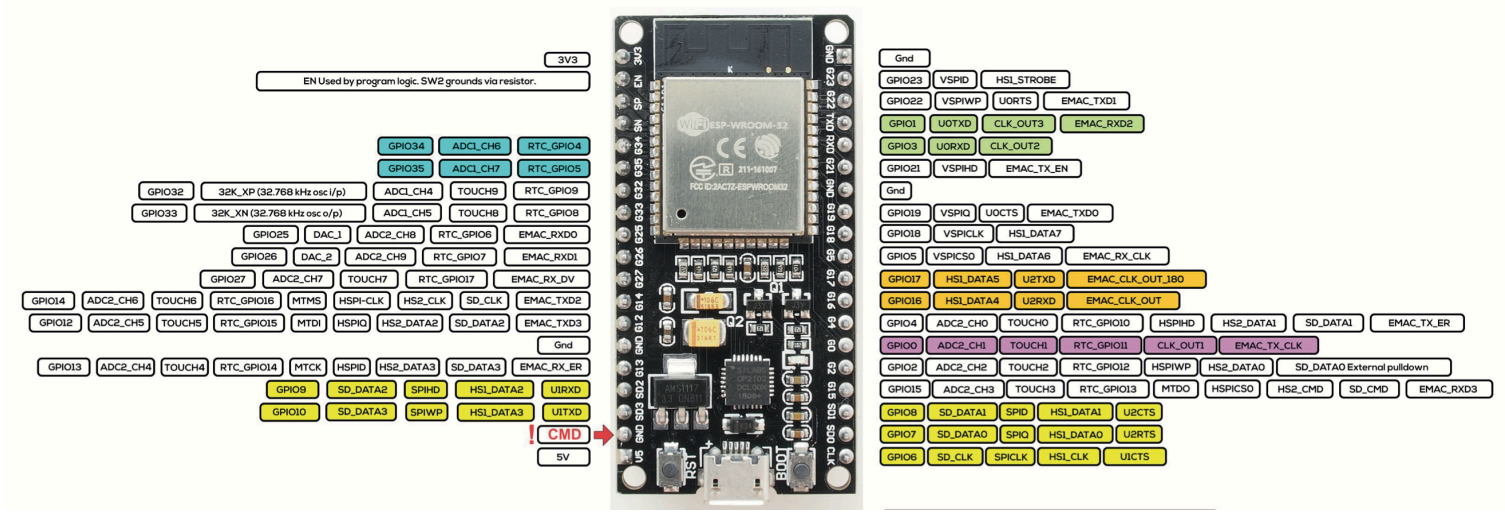


Figura 18. ESP-DevKitC V4 Pin Layout

5.4. SENSOR DE TENSIÓN ZMPT101B.

Este módulo transformador de tensión permite medir tensión alterna como la que tenemos en nuestro hogar, a diferencia del ESP32 que no puede leer tensiones superiores a un rango (0-5V). El módulo ZMPT101B soluciona el problema reduciendo el voltaje de entrada haciéndolo apto para que cualquier microcontrolador pueda leerlo. Por lo que cumple con las especificaciones de proyecto, al ser capaz de leer baja tensión y adaptarla al microcontrolador. Es de pequeño tamaño lo que lo hace fácil de ser instalado, (para ver la instalación, consultar el apartado de *PLANO 1. Conexión Eléctrica).*

Además, el tipo de conexión con es sencilla, ya que, con tres pequeños cables, conectando los pines de ambos elementos se realiza la conexión. Hay que tener en cuenta que habría que utilizar un par de cables más para realizar la conexión en el cuadro eléctrico.

La onda senoidal de salida está desplazada positivamente para que la onda no tenga voltajes negativos y de esta forma, poder leer la onda completamente en el pin de entrada ADC comentado en al anterior apartado. Este desplazamiento viene determinado por el voltaje con que se alimenta al sensor, por ejemplo: en caso de alimentarlo con 5V el desplazamiento será de

2.5V y si alimentamos el módulo con 3.3V el desplazamiento será de 1.65V. Además, tienen una precisión de un 0.2%, lo que nos asegura unas lecturas del valor de tensión suficientemente precisas. Estas características con un coste inferior a los 10€ lo hacen ideal para esta aplicación.

Las aplicaciones en las que suelen utilizarse son: medición de voltaje AC de baja tensión de hasta 250VAC, monitoreo remoto y protección de equipos AC y retroalimentación para sistemas de control de voltaje AC.

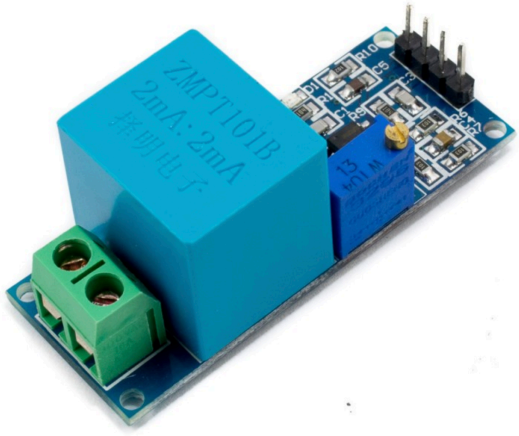


Figura 19. Sensor de tensión ZMPT101B

5.5. SENSOR DE CORRIENTE ACS712-30A.

El ACS712 es una solución bastante económica para poder medir corrientes. Trabaja con un sensor de efecto Hall que detecta el campo magnético que se produce por inducción de la corriente que circula por la línea que se está midiendo.

El sensor devuelve una señal de tensión proporcional a la corriente, dependiendo de las necesidades podemos usar diferentes tipos, dependiendo de su rango de medición: el ACS712-05A, ACS712-20A o el ACS712-30A, para rangos de 5A, 20A o 30A respectivamente, como el propio nombre indica. Por lo que es capaz de medir las corrientes que se darán en el circuito de la vivienda y adaptarlas a un nivel de señal que el ESP32 pueda leer. El modelo de 30A una sensibilidad de 66mV en la señal de salida al microcontrolador, lo que asegura la precisión de las mediciones, junto con un ancho de banda de 80kHz.

Este sensor trae tres pines, dos para conectar la alimentación y un pin para la salida analógica que irá conectada a un ADC del ESP32, por lo que la

conexión entre ambos elementos es sencilla. Junto con su pequeño tamaño para una fácil instalación, (para ver la instalación, consultar el apartado de *PLANO 1. Conexión Eléctrica).*

El ACS712-30A devuelve un valor de 2.5V para una corriente de 0A y a partir de ahí incrementa proporcionalmente de acuerdo con la sensibilidad, existiendo una relación lineal entre la salida de tensión del sensor y la corriente.

En la aplicación se emplean cinco de estos sensores, uno para cada circuito. Pero en futuras variaciones o aplicaciones, este número podría varia en función de los intereses, y finalidades del proyecto.

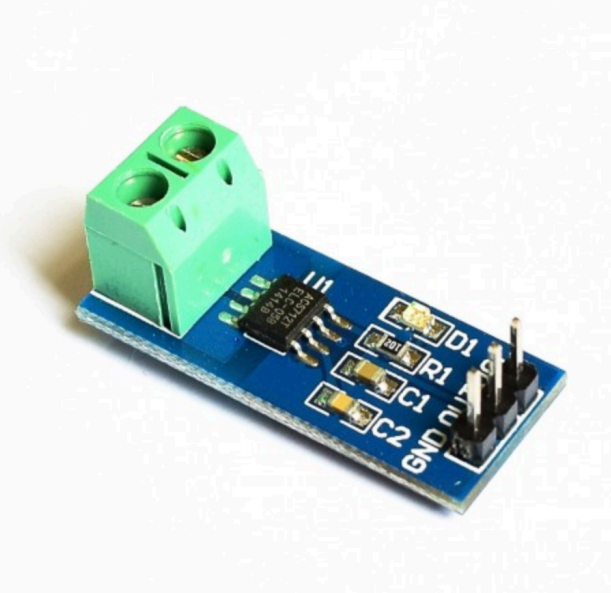


Figura 20. Sensor de corriente ACS712-30A

6. PROTOCOLO DE COMUNICACIÓN MQTT.

MQTT es un protocolo de mensajería asíncrona, usado para la comunicación machine-to-machine (M2M) en el ámbito de IoT. Fue inventado a finales de los años noventa, por dos ingenieros, Andy Standford-Clark de IBM y Arlen Nipper que pertenecía a Arcom, actualmente Eurotech. Lo crearon con el fin de tener un protocolo que consumiera poca batería y trabajara con el mínimo ancho de banda posible que fuera capaz de conectar los oleoductos usando conexión por satélite.

Es un protocolo que se basa en un modelo de publicación y suscripción, además es muy simple, por lo que es una gran opción a la hora de implementarlo en dispositivos sin muchos recursos, y poder utilizarlo en redes con poco ancho de banda también es un punto a favor. Estas dos características lo hacen muy adecuado para aplicaciones IoT, ya que suelen utilizarse redes con latencias altas y dispositivos con recursos limitados, por eso es el utilizado en esta aplicación.

Comparándolo con el protocolo HTTP (Protocolo de Transferencia de Hipertexto), utilizado en la red mundial, este no sería muy aplicable al IoT debido a que es un protocolo que se basa en un modelo unidireccional cliente-servidor, por lo que los clientes nunca podrían recibir información de forma pasiva, solamente mediante una petición. Además de ser un protocolo síncrono, es pesado por lo que podría tener problema en las redes de latencia alta., a diferencia del MQTT.

Como se ha nombrado anteriormente, es un protocolo que se basa en el modelo de publicación y subscripción. El emisor publica los mensajes sobre un *topic* (tema), el receptor es el que recibe estos mensajes, para ello, debe de estar suscrito a dicho *topic.* Esta conexión se logra mediante un *broker,* que realiza las funciones de intermediario, mandado los mensajes a cada receptor según al *topic* el que está suscrito.

El aspecto principal del modelo pub/sus es el desacoplamiento entre emisor y receptor, que se da en tres dimensiones:

- Desacoplamiento de espacio: el emisor y receptor solamente necesitan conocer la dirección IP del *broker* y del *topic*.
- Desacoplamiento de tiempo: emisor y receptor no es necesario que estén en funcionamiento a la vez.
- Desacoplamiento de sincronización: durante la publicación o la suscripción, las operaciones que realizan los dos intervinientes no se ven detenidas.

6.3. CLIENTE Y *BROKER.*

Hay dos elementos en este protocolo: el cliente y el *broker*.

El cliente MQTT, tiene la función de suscribirse a un *topic* o publicar un mensaje, pero, además, puede tener ambas funciones a la vez. Cliente puede ser cualquier dispositivo que esté conectado a un *broker* a través de internet, por ejemplo, puede ser tanto un ordenador, móvil o un microcontrolador, como es el caso del ESP32. Una gran ventaja de los clientes MQTT, es, que son sencillos de aplicar, ya que se pueden encontrar muchas librerías en diferentes lenguajes para ser aplicados, por lo que además de ser sencillo, y pesar poco, tiene mucha flexibilidad.

Respecto al *broker,* es el centro de este modelo, al que se conectan todos los clientes para realizar el intercambio de información. Tiene la responsabilidad de que los mensajes lleguen a los clientes que estén suscritos a cada *topic.* Además, autentica al cliente, que se conecta al *broker* con un usuario y una contraseña.

En la figura 21 , se puede visualizar un esquema que muestra un ejemplo de flujo de información. En la figura se muestra como el *broker* realiza de intermediario entre los distintos elementos. Recibe la información por parte de los sensores y por otro lado la petición de los elementos de recibir "x" información, y manda la información correspondiente a la petición.

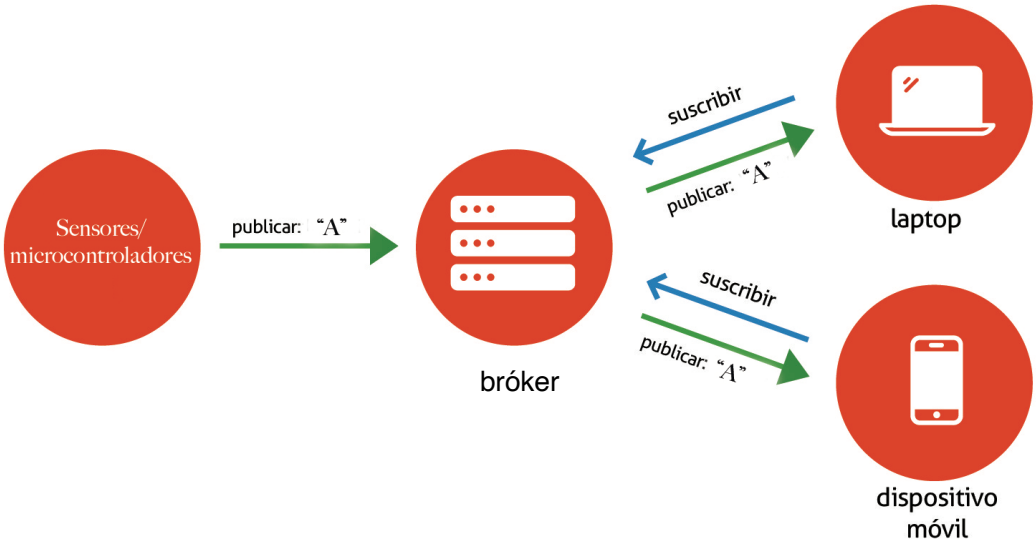


Figura 21. Esquema flujo información

El protocolo está basado en TCP/IP.

La conexión se realiza siempre de cliente a *broker*. El cliente le manda un mensaje para la conexión al *broker* (connect) y este le responde con un mensaje que informa mediante un código, de en qué estado se encuentra la conexión (connaback). En la aplicación que se ha desarrollado el mensaje de "connect" contiene el usuario y la contraseña para poder conectarse al *broker*, ¿qué ocurre?, estos mensajes no están cifrados por lo que no se realiza la conexión de forma segura. Para ello, habría que emplear TLS (Transport Layer Security) o certificados SSL (Secure Sockets Layer). El cliente seguirá conectado hasta que no mande un mensaje de desconexión al *broker.*

Códigos del connback

Código devuelto	Significado
0	Conexión aceptada.
1	Conexión rechazada, errónea versión del protocolo.
2	Conexión rechazada, identificador rechazado.
3	Conexión rechazada, servidor no disponible.
4	Conexión rechazada, usuario o contraseña incorrectos.
5	Conexión rechazada, no autorizada.

Tabla 3. Códigos del connback

6.4. *TOPIC.*

El *topic* es lo que le sirve al *broker* para diferenciar y distribuir los mensajes a los clientes. Por lo que, cuando el *broker* recibe un mensaje de un cliente que ha realizado una publicación, compara este *topic* con todos los *topic* de los clientes, suscriptores de ese *topic*, para mandarles el mensaje a ellos.

Un aspecto importante es que todos esos *topic* se pueden jerarquizar, por lo que, dentro de una misma temática, se puede seleccionar diferente información. El ejemplo más sencillo para comprenderlo es el de una casa, en el que, dentro de una misma habitación, podemos tener diferente información como la temperatura, o las luces.

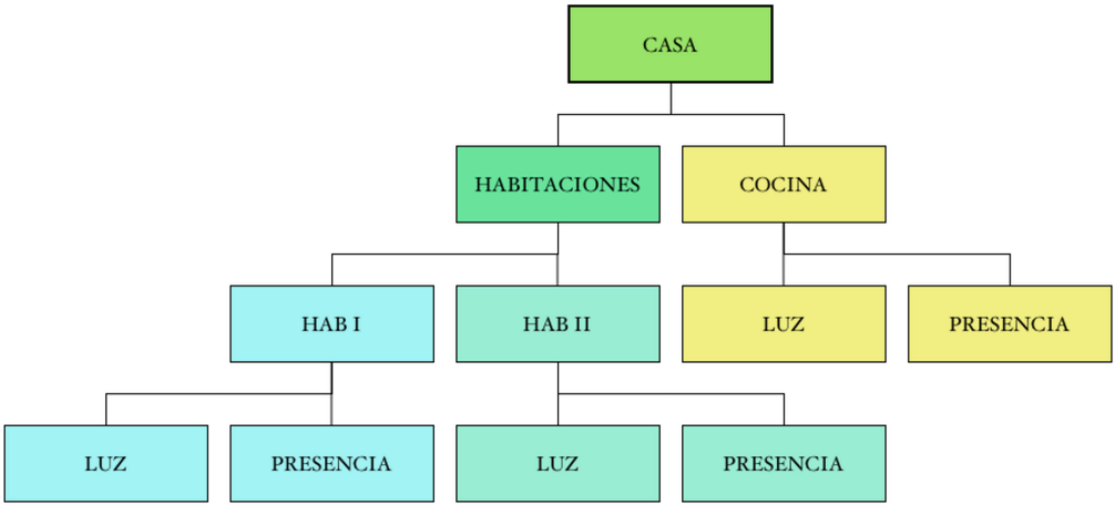


Figura 22. Ejemplo Diagrama de tópicos.

Los *topic* se escriben de la siguiente forma:

CASA/HABITACIONES/HAB I/LUZ, CASA/COCINA/PRESENCIA… Se debe tener en cuenta que se diferencia entre minúsculas y mayúsculas, reconoce los espacios, aunque no es muy recomendable su uso. La forma de jerarquizarlos es mediante el símbolo "/" el cual diferencia cada nivel. También se pueden utilizar símbolos como "+" que sustituyendo a un nivel y engloba todos los que están a ese nivel, por ejemplo, HABITACIONES Y COCINA. El otro símbolo que se puede utilizar es "#" que lo que hace es sustituir varios niveles, por ejemplo, CASA/COCINA/# englobaría tanto la luz como la presencia en la cocina.

En la siguiente página se puede observar la jerarquía de la aplicación desarrollada.

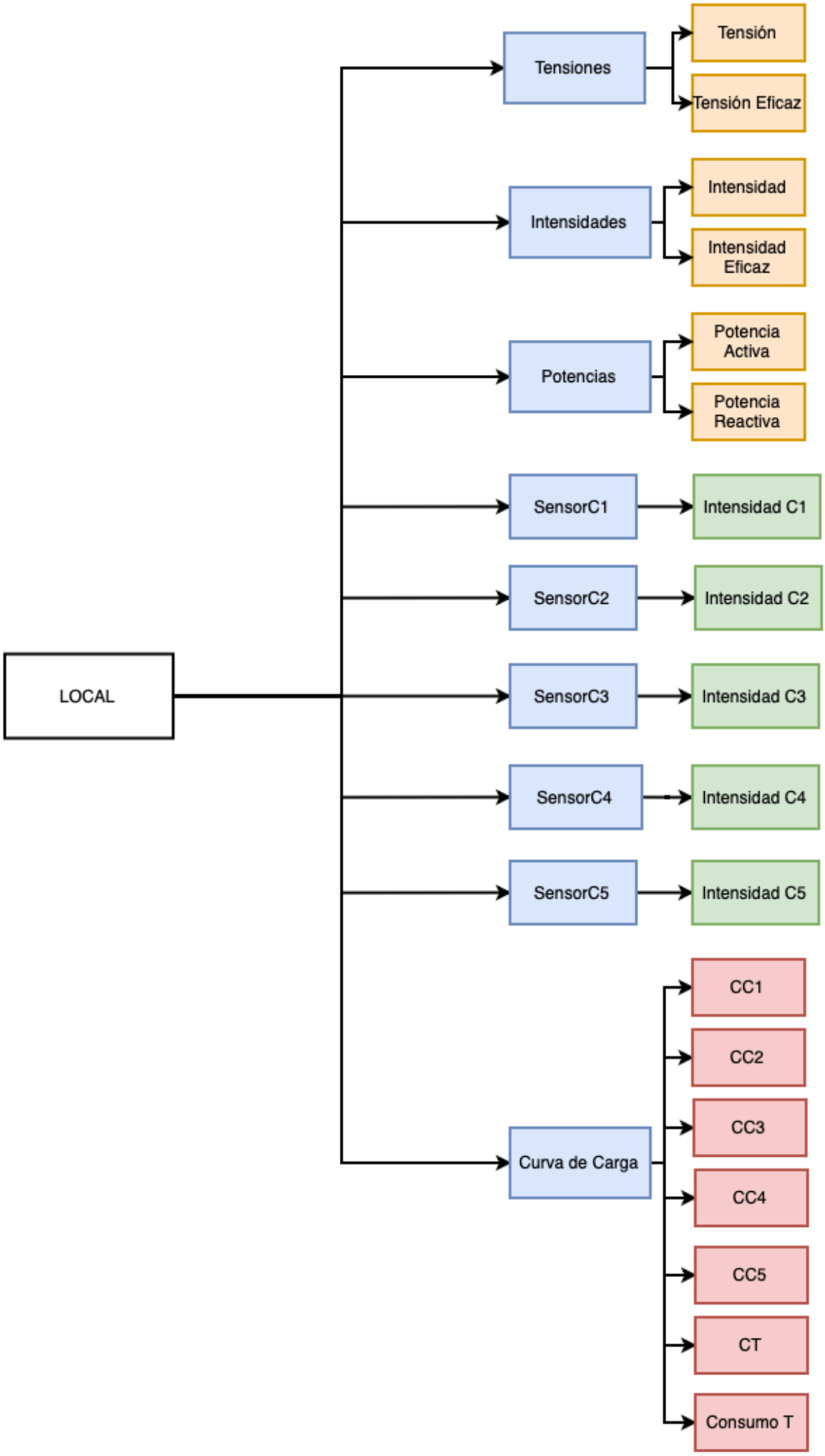


Figura 23. Diagrama de tópicos aplicación

6.5. QoS (Quality of Service).

La calidad del servicio (QoS) mide la garantía de entrega de los mensajes, define la forma en que se asegura que el cliente o el *broker* reciben los mensajes. De forma que tenemos una certeza de que los mensajes son recibidos. Cuanto mayor sea el QoS hay más fiabilidad, pero requieren mayor ancho de banda y mayor latencia. Hay hasta tres niveles:

- QoS 0: "A lo sumo una vez", quiere decir que el mensaje se envía una vez, pero no se recibe confirmación de la entrega. Lógicamente es el que menos recursos requiere y el que menos esfuerzo realiza a la hora de mandar los mensajes.
- QoS 1: "Al menos una vez", el mensaje enviado está garantizado que llegará al receptor. El mensaje será enviado por el emisor todas las veces que haga falta hasta que este reciba un mensaje de confirmación de la entrega, llamado *puback*. Se necesitan más recursos y la comunicación se ve un poco ralentizada.
- QoS 2: "Exactamente una vez", este nivel se asegura de que el mensaje se entrega una sola vez. Necesita de dos confirmaciones entre el emisor y el receptor. En cambio, en este nivel, el rendimiento del sistema se ve disminuido.

Se debe tener en cuenta que, por ejemplo, si un emisor publica el mensaje con un QoS 1, y hay dos receptores, uno suscrito con QoS 0 y el otro con QoS 1, el que está suscrito con QoS 0 recibirá el mensaje con QoS 0; en cambio el que estaba suscrito con QoS 1, recibirá el mensaje en QoS 1. Por lo que el receptor elige el nivel de calidad de servicio máximo al que va a recibir los mensajes.

7. DESCRIPCIÓN DE LA SOLUCIÓN.

7.1. EXPLICACIÓN DEL SISTEMA.

En primer lugar, es importante conocer como está distribuida la instalación eléctrica en una vivienda. Una vez viendo el cuadro eléctrico, se encuentran el IGA (interruptor general automático) cuyo fin es proteger la vivienda frente a sobrecargas o cortocircuitos; un ID (interruptor diferencial) destinado a la protección de las personas frente a posibles contactos con partes activas de la instalación; y por último los diferentes magnetotérmicos para los distintos circuitos**.** Los circuitos que se encuentran en una vivienda son generalmente: el C1 destinado a la luminaria de 10A, C2 que integra las tomas de corriente de uso general de 16A, C3 que da tensión a la vitrocerámica y al horno cuyos enchufes son de 25A, C4 destinado a la lavadora, lavavajillas y el calentador enchufes de 20A y, por último, C5 enchufes del baño y cocina (con posible contacto con agua) de 16A.

Comentar, que, puede haber casos en que varíe el número de circuitos, como añadiendo circuitos adicionales de iluminación o similares, dependiendo de la instalación, y tipo de vivienda. Para obtener una idea esquemática de la instalación, consultar el apartado *PLANOS 1. Conexión eléctrica.*

Para realizar esta aplicación IoT para la monitorización de consumos eléctricos de una vivienda se va a requerir de, un sensor de tensión el cual va a medir el valor de la tensión en la vivienda. Este sensor, está conectado tras el interruptor diferencial y en paralelo con los magnetotérmicos de los circuitos de la vivienda. Recibe la tensión de la instalación (230V) y la reduce a un nivel de tensión a su salida que el microcontrolador puede leer.

Además de este sensor de tensión, la instalación cuenta con cinco sensores de corriente, los cuales, están conectados aguas abajo de los magnetotérmicos de cada circuito de la vivienda. Están conectados en serie con las cargas. Harán la lectura del valor de corriente del circuito y enviarán una señal de tensión proporcional a este valor al microcontrolador.

Tras la realización de los cálculos que se van a comentar a continuación, el ESP32 enviará los datos mediante el protocolo MQTT, para mostrarlos en la interfaz y almacenarlos en la base de datos. En los próximos apartados se específica como se ha realizado la conexión a la red Wifi, al *broker* MQTT y como se efectúa el envío de la información.

En la siguiente figura se puede encontrar un diagrama de red para tener una visión esquemática de la estructura de la aplicación explicada a continuación. El servidor es local, por lo tanto, el servidor es el propio ordenador. El ordenador va a tener tres servicios diferentes: servicio *broker* MQTT, servicio Node-Red y servicio Mysql. Se le conoce a servicio como cada puerta que va a intercambiar información con algún elemento externo. En este caso, el servidor

va a estar recibiendo información desde el *broker* MQTT, y mandando información a Node-Red y a la base de datos Mysql.

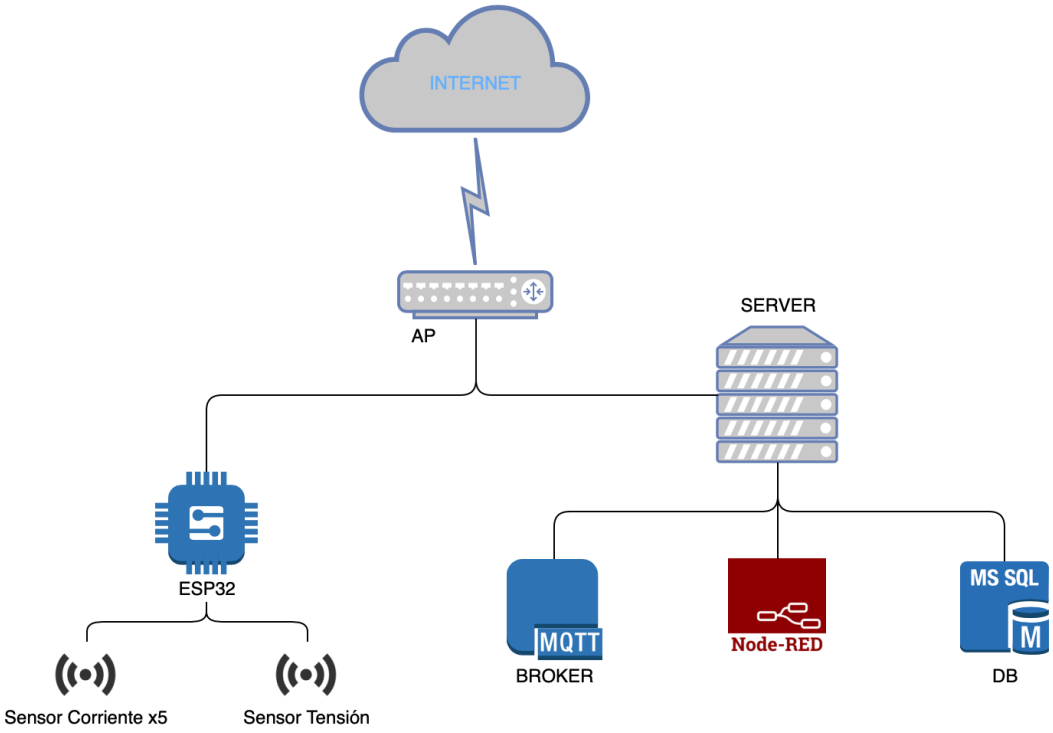


Figura 24. Diagrama de red.

7.2. CÁLCULOS.

7.2.1. Tensión y corriente eficaz.

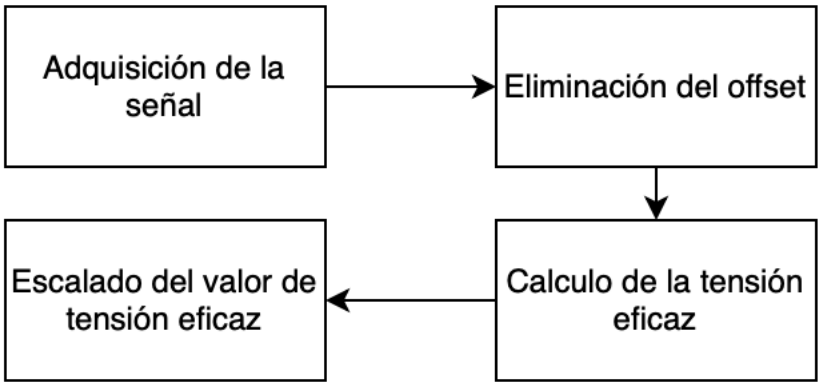


Figura 25. Muestreo de la onda senoidal.

Como se puede observar en la figura 25, se ha de seguir el siguiente procedimiento. En primero lugar, la señal senoidal de tensión debe ser muestreada, por lo que se ha establecido que, para tener una mayor precisión en las medidas, se van a realizar 20 tomas de datos por cada ciclo. Teniendo en cuenta la frecuencia es de 50Hz, un ciclo son 20 milisegundos, resultando la toma de un dato cada milisegundo. Es decir, frecuencia de muestreo de 1kHz. En la figura 25, se puede observar lo referente al muestreo de la señal senoidal.

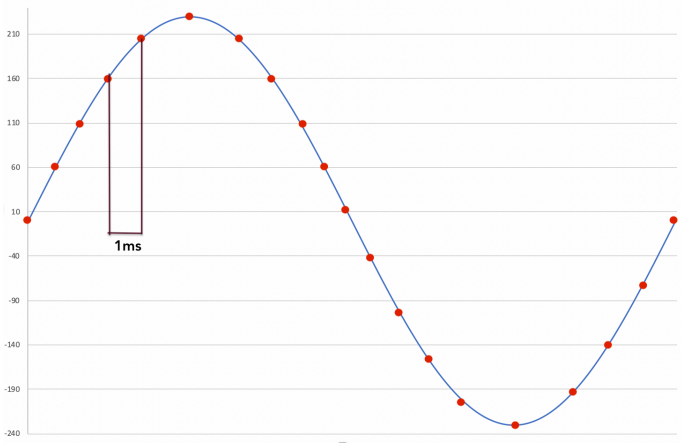


Figura 26. Muestreo de la onda senoidal.

Una vez realizado el muestreo, los 20 datos obtenidos se almacenarán en un vector. Estos valores tienen la característica de que van a ser todos positivos ya que el sensor le añade un offset a la señal de salida. Para corregir esto y tener valores tanto positivos como negativos, se sumarán los 20 datos y se dividirá entre el número de datos, en este aso 20. El valor resultante es el valor medio,

el cual habrá que restárselo a cada uno de los 20 datos tomados. De esta forma, habiendo eliminado el offset se obtiene una onda alterna pura.

El siguiente paso es calcular el valor eficaz de la tensión. Para ello se empleará la siguiente fórmula:

$$V_{ef} = \sqrt{\frac{1}{T} \cdot \int_0^t V(t)^2 \cdot dt}$$

Para calcular el valor de la integral, se ha de realizar el cuadrado de cada uno de los 20 valores calculados anteriormente y sumar los 20 resultados correspondientes. Seguidamente el producto de este valor resultante por los 20 milisegundos del periodo. Por último, es necesario escalar este valor de tensión, ya que la señal recibida por el microcontrolador se encuentra entre 0V y 5V, y realmente en el sistema hay 230V eficaces.

Para la obtención de la corriente eficaz el procedimiento es el mismo que el realizado para el cálculo de la tensión eficaz.

7.2.2. Potencia activa, aparente, reactiva y fdp.

Para el cálculo de la potencia activa, se necesitan los valores calculados que forman la onda senoidal pura. Partiendo de los 20 valores de tensión y 20 valores de corriente, que se debe de escalar el nivel de tensión y dividir el valor de corriente entre la sensibilidad del sensor. Posteriormente, se procede a multiplicar cada valor de tensión con el correspondiente valor de corriente, que se ha medido al mismo tiempo. Tras esto, se suman los resultados y se divide entre el número de datos, en este caso veinte, dando como resultado el valor de potencia activa.

El cálculo de potencia aparente se realiza a partir de la tensión y la corriente eficaces, con la siguiente formula:

$$S = V_{ef} \cdot I_{ef}$$

Para obtener el valor de la potencia reactiva, se va a utilizar el triangulo de potencias. La formula empleada será la siguiente:

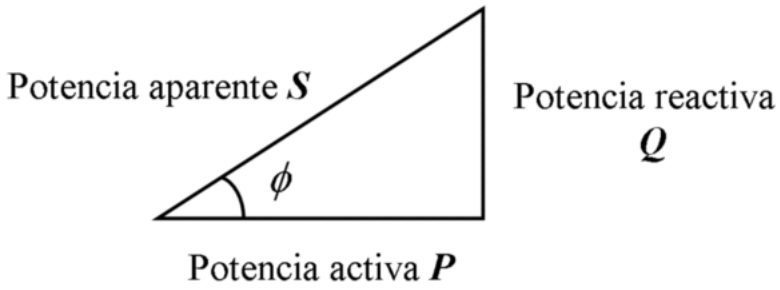


Figura 27. Triangulo de potencias.

$$Q = \sqrt{S^2 - P^2}$$

Además, el factor de potencia lo podemos calcular como:

$$cos\varphi = \frac{P}{S}$$

7.3. EXPLICACIÓN CÓDIGO DE ARDUINO

En primer lugar, se debe tener clara la estructura que va a tener el código, explicada a continuación:

- En la primera parte de código se incluyen las librerías que se van a utilizar (WiFi, PubSubClient, NTPClient, WiFiudp…). Después, se inicializan y declaran variables globales. Estas variables corresponden a los ítems relacionados con los sensores (pin de entrada, variables para almacenar valores…), y a la configuración de red que se cargará en el ESP32 para conectarse a la red local y al broker MQTT (contraseñas, usuarios, topics…). Además, también se pueden crear diferentes funciones que realizarán tareas concretas a lo largo del código.

- Función setup(): En esta parte del código se realiza la carga de la configuración general. Esta función es la primera que se ejecuta y lo hace una sola vez, en ella podemos encontrar la inicialización de varias funciones, establecer la velocidad de comunicación….

- Función loop(): Esta función contiene la parte del código que se va a ejecutar como el propio nombre indica, continuamente, realizando cálculos, obteniendo mediciones de sensores, almacenando valores en variables, y mandando mensajes MQTT, entre otras opciones.

7.3.1. Librerías.

- WiFi: facilita la realización de la conexión del microcontrolador a la red Wifi correspondiente.
- PubSubClient: va a permitir la conexión al broker MQTT y las operaciones subscripción y publicación en el protocolo MQTT.
- NTPClient: facilita la conexión del microcontrolador a un servidor NTP para que la fecha y la hora se mantengan actualizadas sin necesidad de utilizar un RTC (Real Time Clock).
- WifiUdp: aporta la comunicación UDP (Universal Datagram Protocol), que, a diferencia de la TCP, realiza los envios de información más rápido debido a que no espera un acuse de recibo del paquete enviado. Por lo que no sobrecarga al sistema. Esto se aplica para recibir la información de fecha y hora.

A continuación, en la figura 27, se muestra el flujograma general del código de arduino.

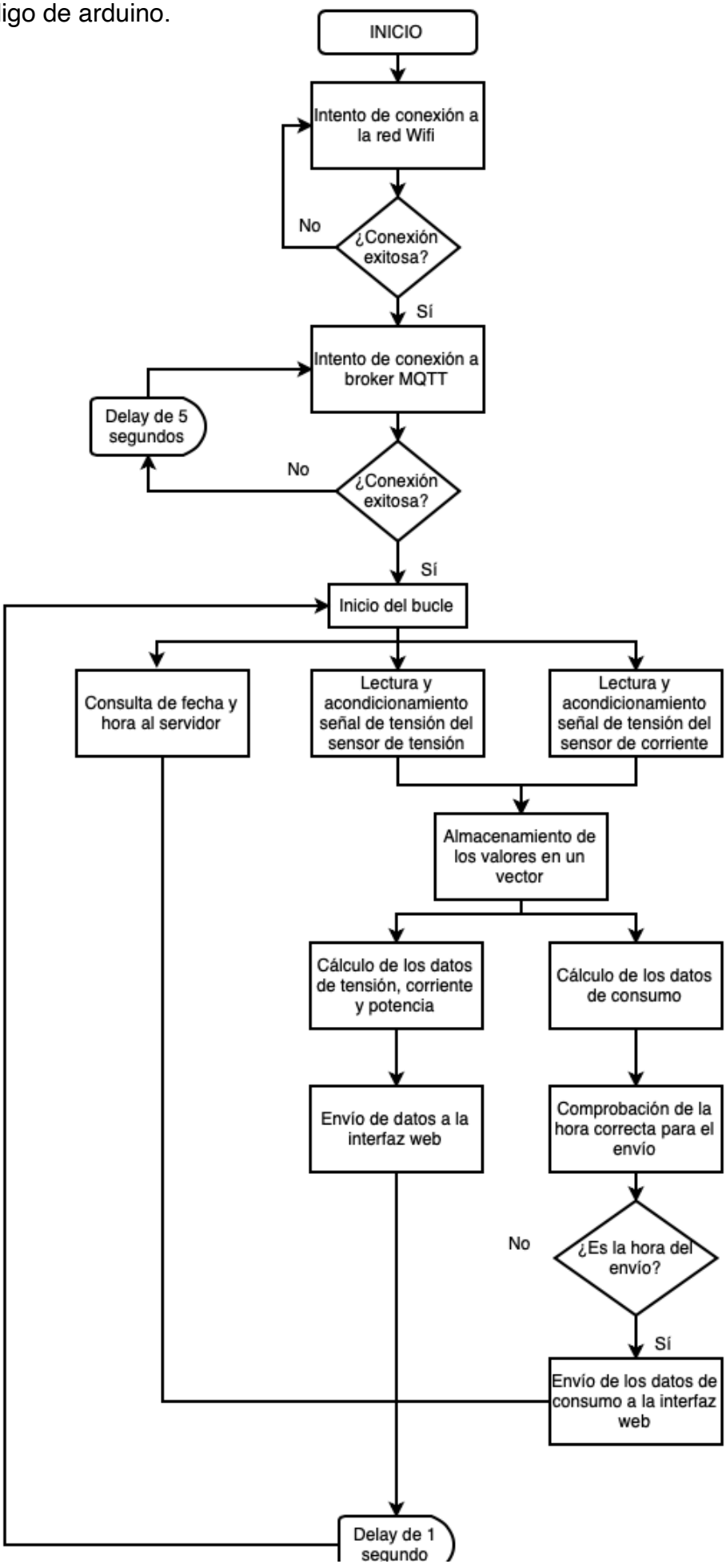


Figura 28.. Flujograma general.

En los siguientes apartados, se va a comentar brevemente cada parte del código, mostrando junto con la explicación un flujograma del proceso.

7.3.2. Conexión WiFi:

Como el propio significado de IoT indica, se va a emplear conexión Wifi, por lo que el ESP32 estará conectado a la red Wifi en este caso de una vivienda. Se va utilizar la función *stup_wifi()*, se debe facilitar el ssid y la contraseña para que pueda realizar la conexión, esto será lo primero que realice el ESP32 antes de conectarse al *broker*. Intentará la conexión hasta lograrlo y se mostrará la dirección IP del ESP32. Una vez esté conectado al Wifi, el ESP32 podrá conocer la fecha y la hora (explicado posteriormente). La conexión al Wifi es necesaria puesto que, gracias a esta, se podrá realizar la comunicación entre los elementos.

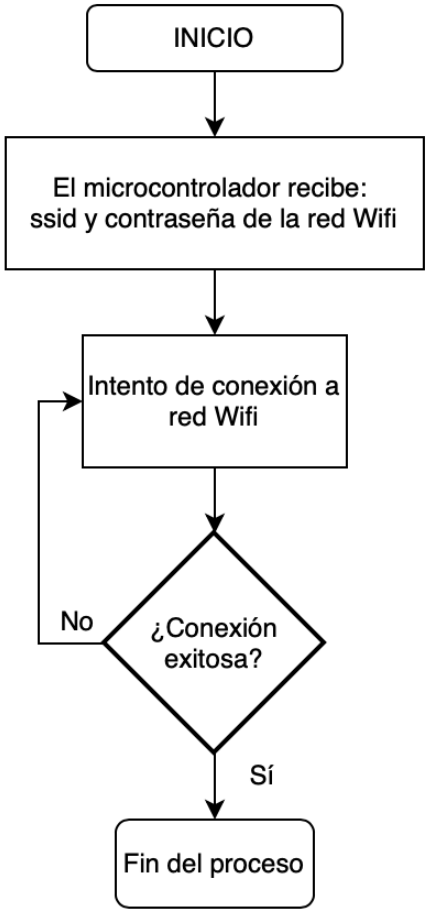


Figura 29. Flujograma conexión Wifi

7.3.3. Conexión *broker* MQTT:

Como se ha comentado anteriormente, se va a implementar una serie de funciones que van a permitir la bidireccionalidad de información con el *broker* de forma que el ESP32, pueda tanto suscribirse como publicar. En este caso se va a utilizar un localhost, de forma que el *broker* sea el propio ordenador.

La conexión al broker MQTT es esencial para poder realizar la comunicación entre los elementos ya que mediante este se van a enviar y recibir los mensajes con la información que requiere la aplicación.

En primer lugar, vamos a utilizar la librería *PubSubClient*, que permite realizar operaciones simples de publicación/subscripción sobre un servidor MQTT. Posteriormente, se ha de configurar el servidor, proveyendo al programa del servidor (dirección IP del equipo con el que se está trabajando), puerto, el nombre de usuario y la contraseña.

Posteriormente se procede a la conexión utilizando la función *reconnect*, en primer lugar, se creará un ID para el cliente que se va a conectar, de forma que, si hubiera varios, no se solaparan entre ellos, a continuación, se le facilitará el usuario y la contraseña a la función, y se procede a realizar la conexión. En caso de haber algún problema se mostraría un mensaje de error y se volvería a intentar la conexión al cabo de 5 segundos.

Cabe comentar que la parte de suscribirse no se va a utilizar, pero que está incorporada por si en un futuro fuera necesario dotar a la aplicación de otras utilidades, siendo esto muestra de la flexibilidad que tiene la aplicación. Una vez realizada la conexión con el *broker* MQTT, mediante la función *Callback* se reciben los mensajes entrantes en un *topic*. Esta función muestra y se encarga de actuar con respecto a la información que recibe.

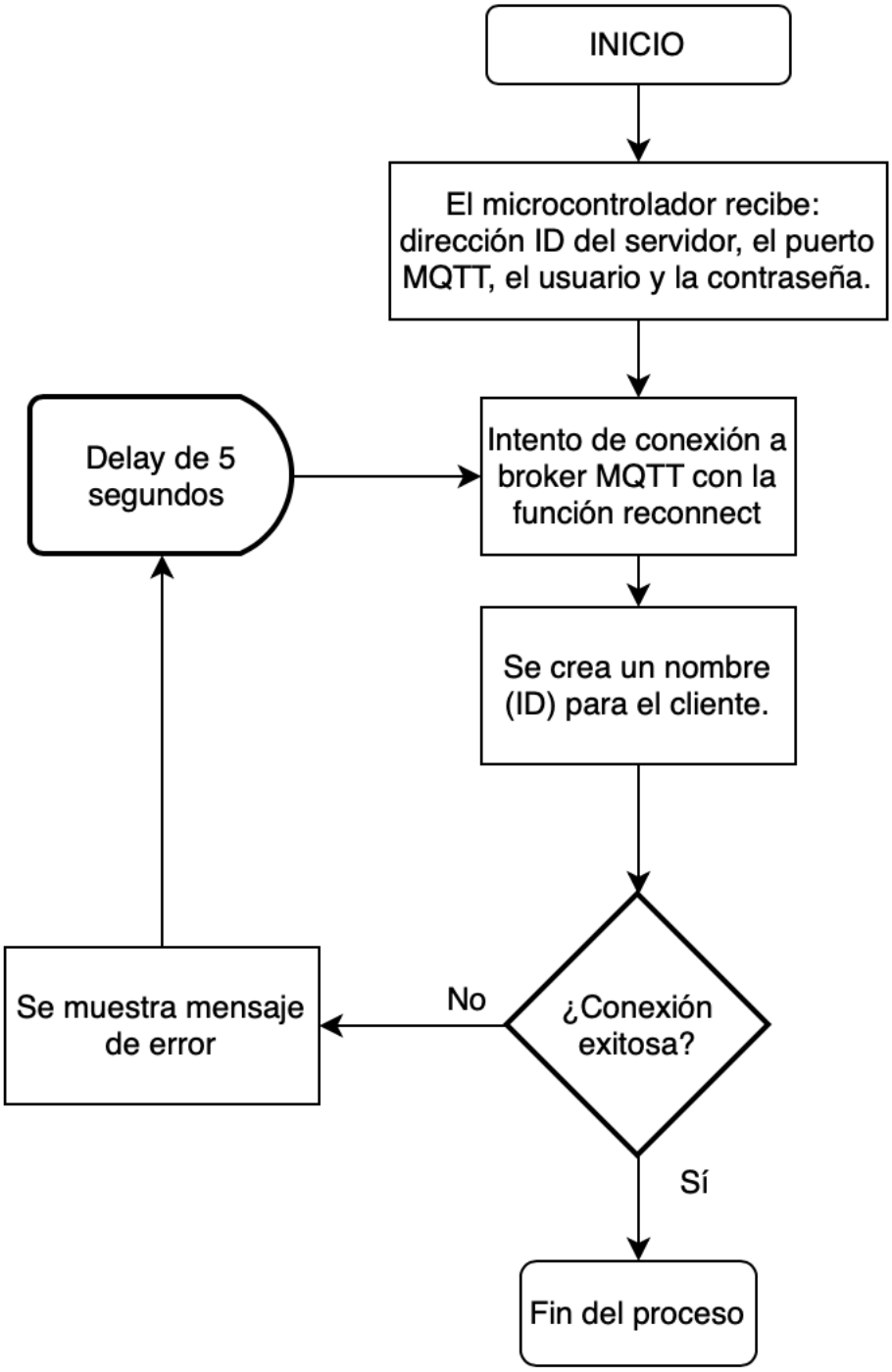


Figura 30. Flujograma conexión MQTT

7.3.4. Lectura del sensor de tensión y cálculo de Vrms:

Debemos conectar el sensor a un pin de entrada digital por ejemplo “A6” o también llamada “GPIO34”. Como se ha comentado anteriormente, se van a tomar 20 datos en 20ms, una frecuencia de muestreo de 1kHz. Estos valores se almacenarán dentro de un vector para el cálculo de la tensión eficaz comentado en el apartado.

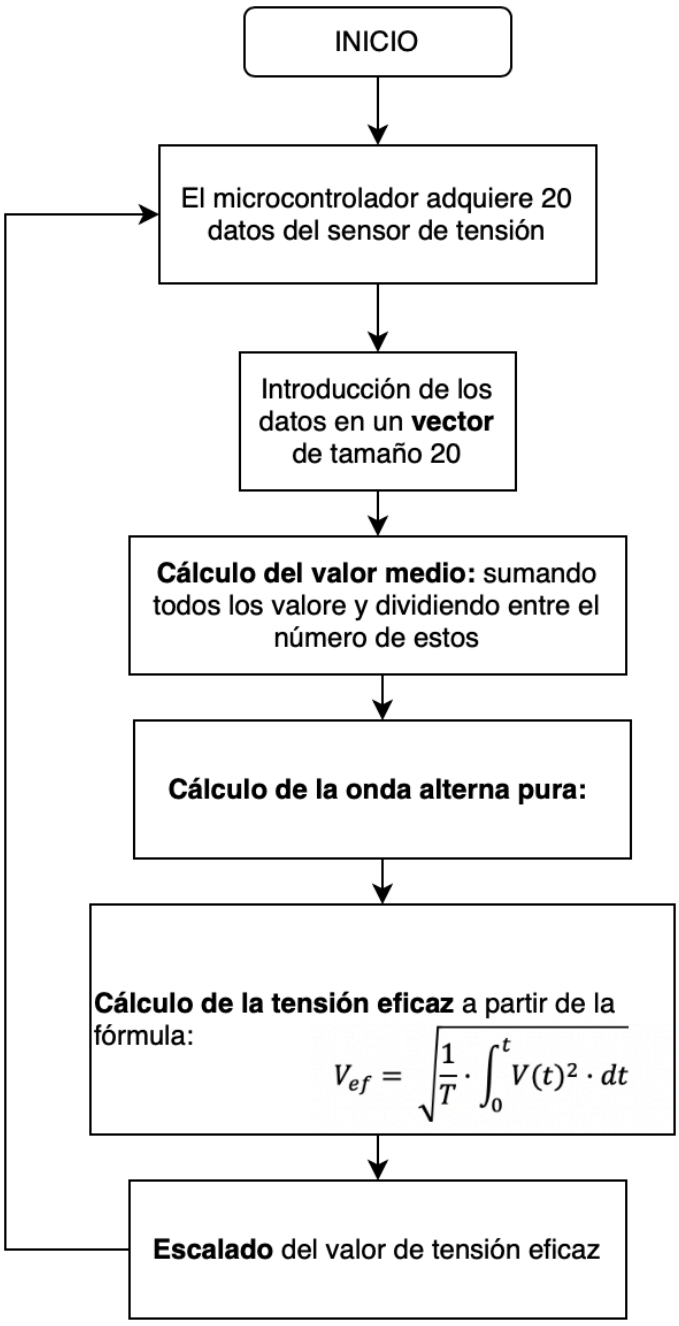


Figura 31. Lectura y cálculo de la tensión eficaz.

7.3.5. Lectura del sensor de corriente y cálculo de Irms:

Debemos conectar el sensor a un pin de entrada digital por ejemplo “A0” o también llamada “GPIO36”. Como se ha comentado anteriormente, el sensor de corriente hace lecturas de intensidades, pero devuelve una señal de tensión al ESP32, por lo que deberá ser convertida a valor de corriente con la fórmula: V = mI + 2,5 (establecida por el sensor, en la que “m” es la sensibilidad). Además, se le debe aportar la sensibilidad del sensor, que en el caso del utilizado en este proyecto es de 66mV/A. El proceso de obtención del valor de la corriente eficaz es el mismo que el de obtención de la tensión eficaz mostrado en el apartado anterior.

7.3.6. Cálculo de las potencias y fdp:

Como se ha explicado en el apartado, el cálculo de las potencias se realiza tras conocer los valores de tensión y de corriente. Siguiente el siguiente flujograma:

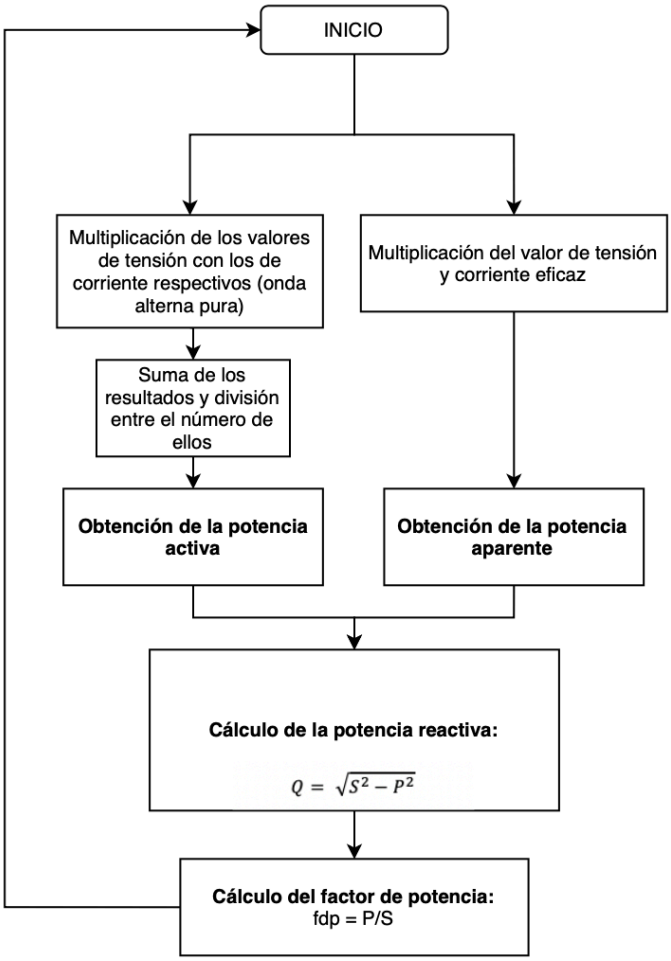


Figura 31. Flujograma cálculo de las potencias.

7.3.7. Envío de los datos:

Los *topic* de los mensajes que se van a mandar mediante MQTT se definen en la primera parte del código.

El envío de información se va a realizar de la siguiente manera, cuando se haya obtenido un valor que ha de ser mandado a la web, este valor se almacenará en su variable correspondiente, y se introducirá en un vector o *array* específico para este. De forma que, a la hora de publicar el valor, este *array* sea el que se mande por el *topic,* como se muestra en la siguiente parte de código, en que se manda el valor eficaz de la tensión. Mediante el protocolo MQTT se consigue realizar la comunicación entre elementos del sistema, y enviar los datos necesarios, cumpliendo de esta forma con los requisitos del sistema en este punto.

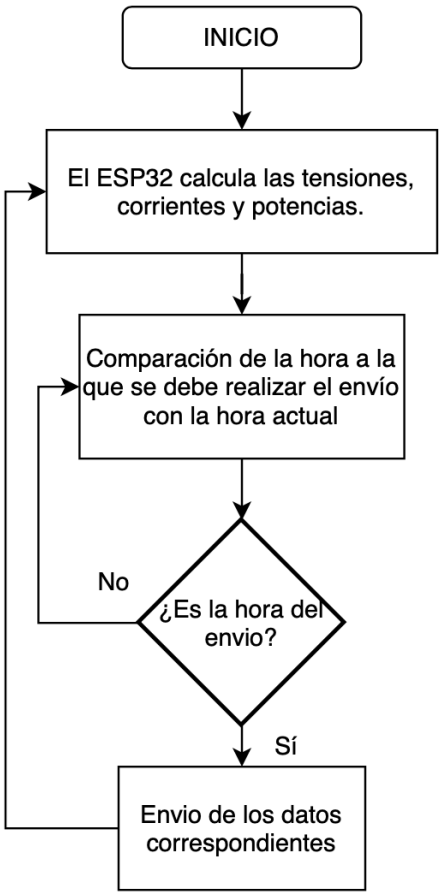


Figura 32. Flujograma envío de datos.

7.3.8. Obtención de fecha y hora:

En este proyecto se han de mandar los valores de consumos total a la base de datos y a la web a unas horas determinadas, de forma que nos aparezcan de forma ordenada y sabiendo en qué rango de horas se da más o menos consumo, para realizar el posterior estudio de el nivel de consumo.

Cuando se consulta la hora al servidor, se debe adaptar su formato para poder ser tratada.

A continuación, se procede a separar la fecha y la hora y almacenarlas en variables *String* diferentes, de forma que podamos mostrarlas por separado y/o utilizarlas en algún bucle para mandar datos en una hora exacta.

Hay que tener en cuenta que en el *setup* hay que configurar la zona horaria en la que nos encontramos, siendo el OffSet las horas de más o menos de nuestra zona horaria, en nuestro caso cogemos de referencia Madrid, con un Offset de +7200.

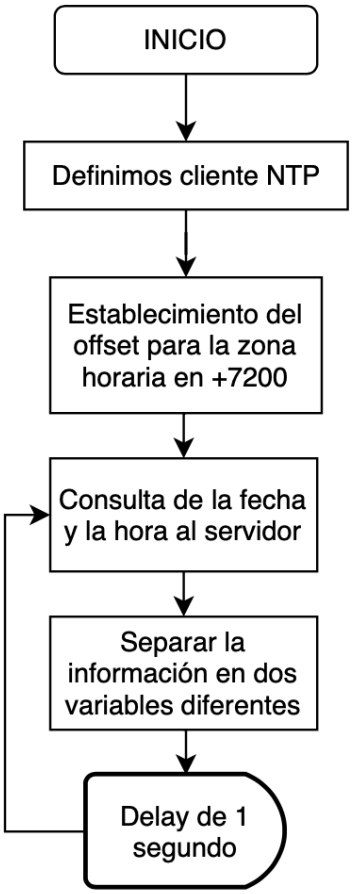


Figura 33. Flujograma obtención de la fecha y hora.

7.4. DESCRIPCIÓN INTERFAZ DEL USUARIO

Como se ha comentado al principio del documento, la interfaz con el usuario debe poder mostrar la información más importante del sistema: tensiones, corrientes, potencias y consumos.

En primer lugar, se ha de instalar un nodo llamado *Mosca*, este nodo es esencial, ya que permite la conexión del Node-Red al broker MQTT, para de esta forma poder recibir y mandar mensajes desde el node red. El cual es configurado con la misma información de dirección del servidor, puerto, contraseña y usuario que se introduce en el ESP32.

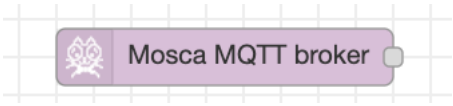


Figura 34. Nodo Mosca MQTT.

Partiendo de este se podrá ir construyendo la web según los requisitos de la aplicación.

7.4.1. Menú 1: tensiones, intensidades y potencias.

En el primer menú se deben mostrar los valores de tensión y corrientes en cada circuito, además de la potencia activa y reactiva que se está consumiendo en ese instante. Para abordar esto, se ha hecho uso de nodos que reciben mensajes MQTT desde el *broker*, y uniéndolos con los nodos de cuadro de texto que muestran estos valores por la página web. Además, se han añadido dos widgets para visualizar de forma gráfica el valor de la tensión y corriente eficaz. Hay que tener en cuenta que para que se muestren en el mismo menú, dentro de la configuración de cada nodo hay que asignarle el grupo correspondiente, en este caso serían: “tensión y corriente” o “potencias”; estos grupos se encuentran dentro de las tablas creadas para separar cada menú. En las siguientes imágenes se puede ver un ejemplo de como se muestran los valores de tensión y el resultado de este menú.

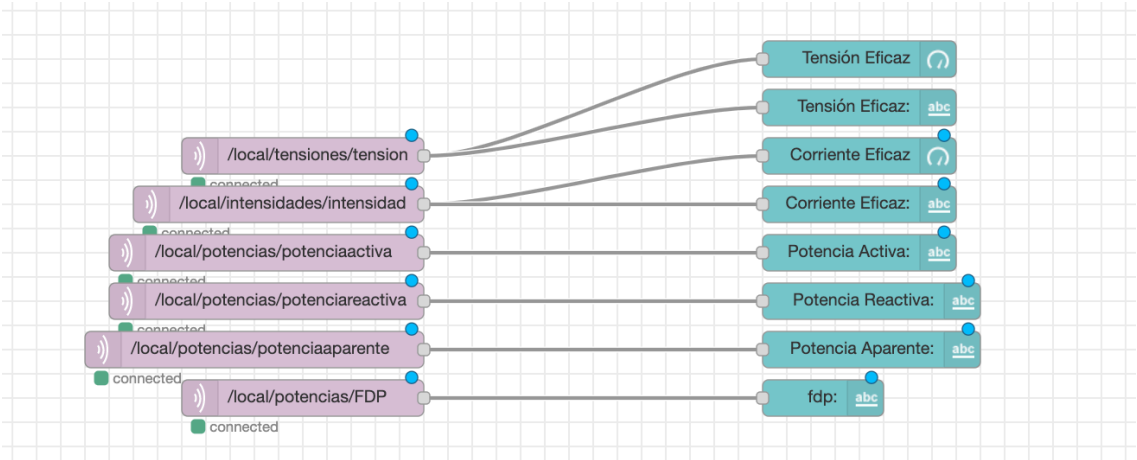


Figura 35. Mostrar mensaje MQTT de tensión, corriente, potencias y fdp..

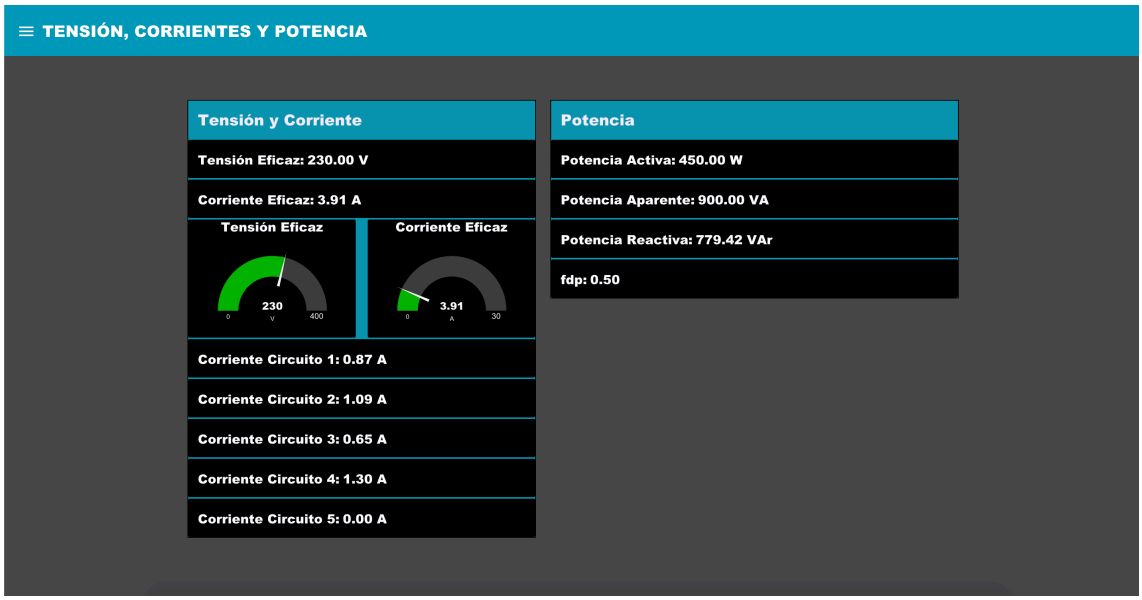


Figura 36. Menú Tensiones, Corrientes y Potencias.

7.4.2. MENÚ 2: curvas de carga circuitos.

En el segundo menú (Curvas de Carga Circuitos) se deben mostrar los consumos que se han dado cada dos horas por cada circuito, de esta forma se obtiene el consumo total por separado. Para lograrlo, se ha hecho uso dos tipos de nodos:

- Nodos que reciben los mensajes MQTT del broker.
- Nodos de gráficas de barras que muestran los valores de los consumos.

Al igual que en el menú anterior hay que asignar los grupos correspondientes a cada nodo de gráficas.

Los nodos de gráficas se pueden configurar para que tengan el tamaño y la separación deseados entre cada una de las gráficas.

A continuación, se muestran estos nodos y cómo se muestra este menú en la página web (descendiendo por la web se visualizan todas las curvas de carga).

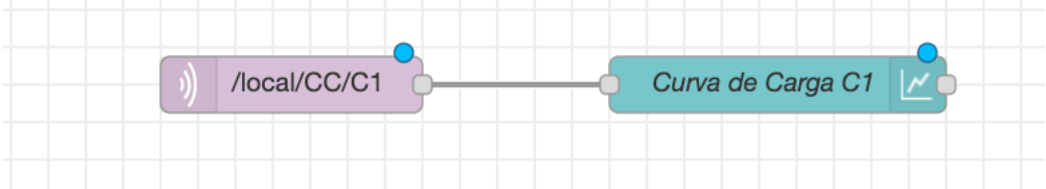


Figura 37. Ejemplo enviar mensaje de consumos del C1 a la gráfica de barras.

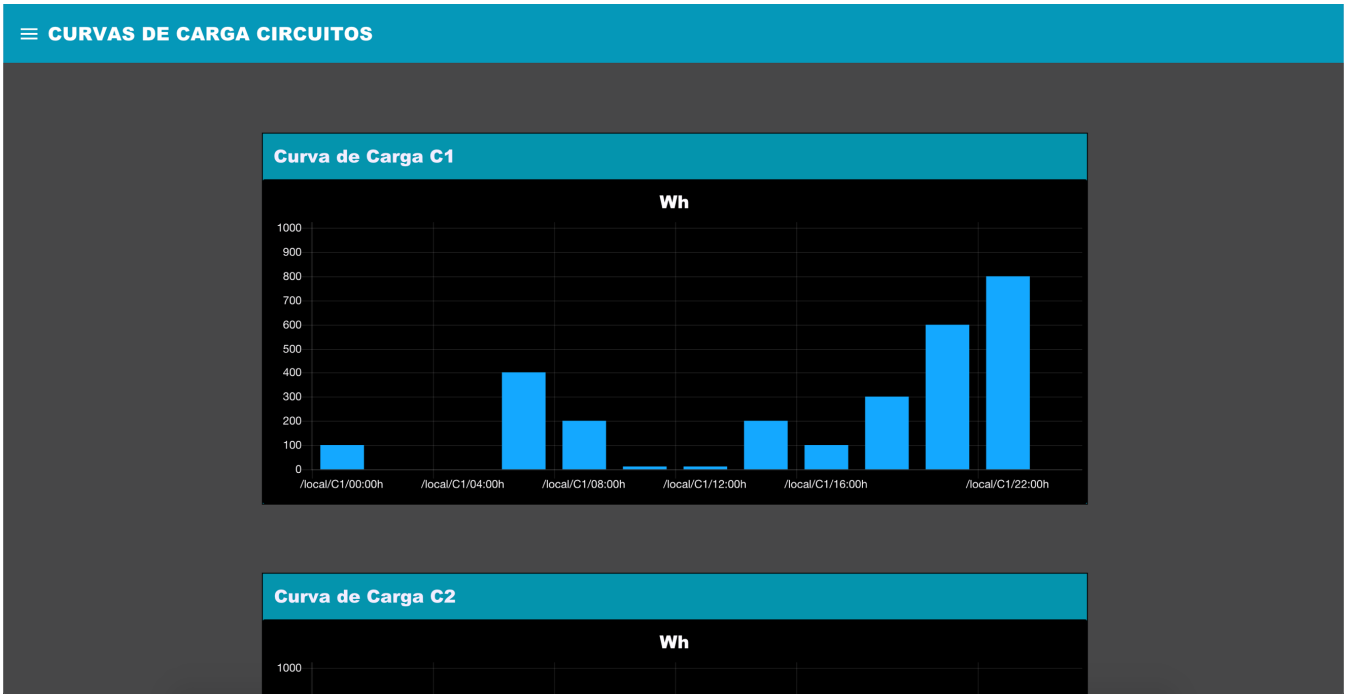


Figura 38. Menú Curvas de Carga Circuitos.

7.4.3. MENÚ 3: curva de carga total.

En el tercer menú (Curva de Carga Total) se va a mostrar la curva de carga total de la vivienda, y el valor total en un cuadro de texto. Para esto se ha hecho uso de tres tipos de nodos: nodos para recibir el mensaje del broker MQTT, nodos para mostrar la gráfica de barras y nodo de cuadro de texto que escribe el valor total del consumo (como el utilizado en el primer menú para mostrar las tensiones, corrientes y potencias). Al igual que en los casos anteriores debemos configurar los nodos para que estén en el grupo correspondiente de su menú.

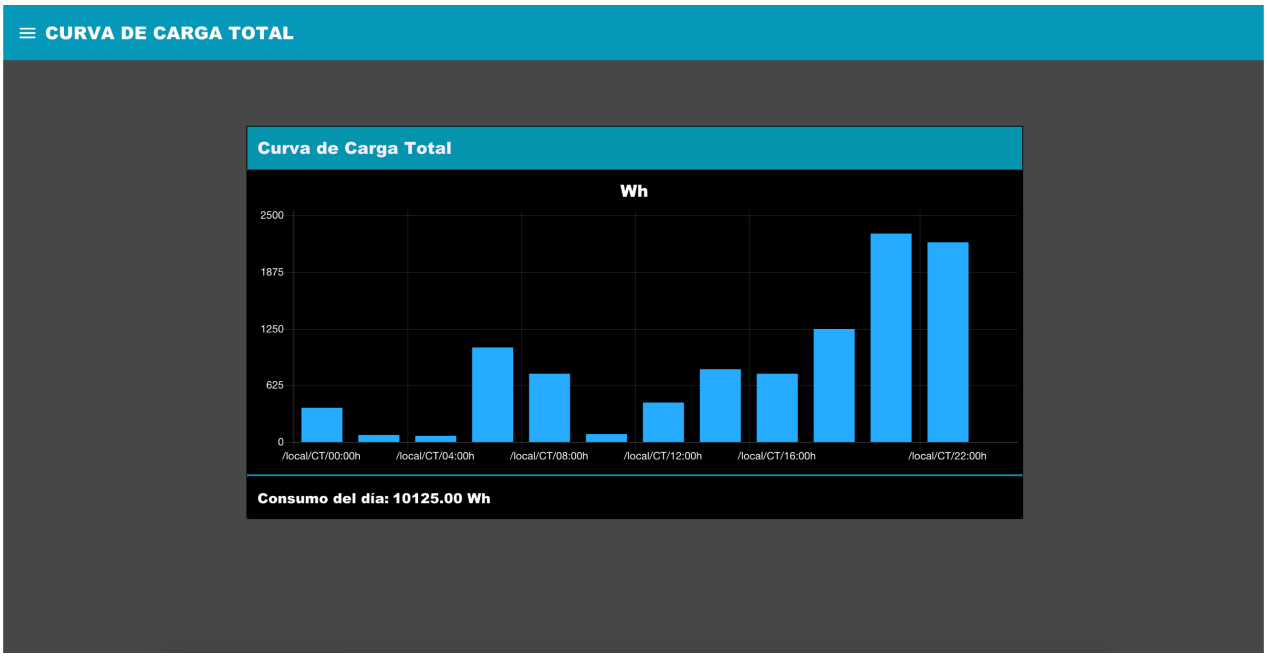


Figura 39. Menú Curva de Carga Total.

7.4.4. MENÚ 4: historial de consumos.

Por último, en el cuarto menú (Historial de Consumos), se va a mostrar una tabla con los valores almacenados en la base de datos, se van a emplear os siguientes nodos:

- Nodos de recepción de mensaje del *broker* MQTT, que van a recibir los valores de consumos.

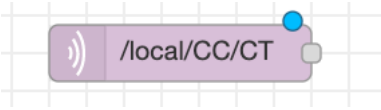
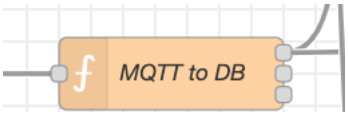


Figura 40. Nodo con mensaje de Consumo Total.

- Nodos de función que van a hacer posible la introducción de los valores de consumo, hora y fecha en la base de datos, además de leer la base de datos para poder mostrarla en la página web.




```
var fecha = { payload: msg.payload_fecha };
var hora = { payload: msg.payload_hora };
var newMsg1 = { payload: msg.payload };
fecha.topic = "INSERT INTO `DB_ConsumosVivienda`.`Tabla` (`Fecha`, `Hora`, `Consumo Wh`)
VALUES ('"+fecha.payload+"', '"+hora.payload+"','"+newMsg1.payload+"' )";
return fecha;
```

Figura 41. Nodo de función para la escritura en DB y su contenido.


```
msg.topic = "SELECT * FROM Tabla"
return msg;
```

Figura 42. Nodo de función para la lectura de la DB y su contenido.

- Nodos de fecha y hora, que van a introducir en la base de datos la fecha y la hora en la que se ha realizado cada consumo que se introduzca. Ambos nodos, están configurados para la zona horaria en la que se sitúa a vivienda.

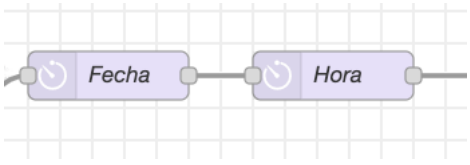


Figura 43. Nodos de fecha y hora.

- Nodo Mysql, este nodo sirve para introducir una base de datos en el Node-Red de forma que se puedan escribir o leer valores en la base de datos en cuestión. Está configurada para introducir los valores en la base de datos creada en Mysql WorkBench, introduciendo el nombre de la base de datos, el puerto, la dirección IP, usuario y contraseña.

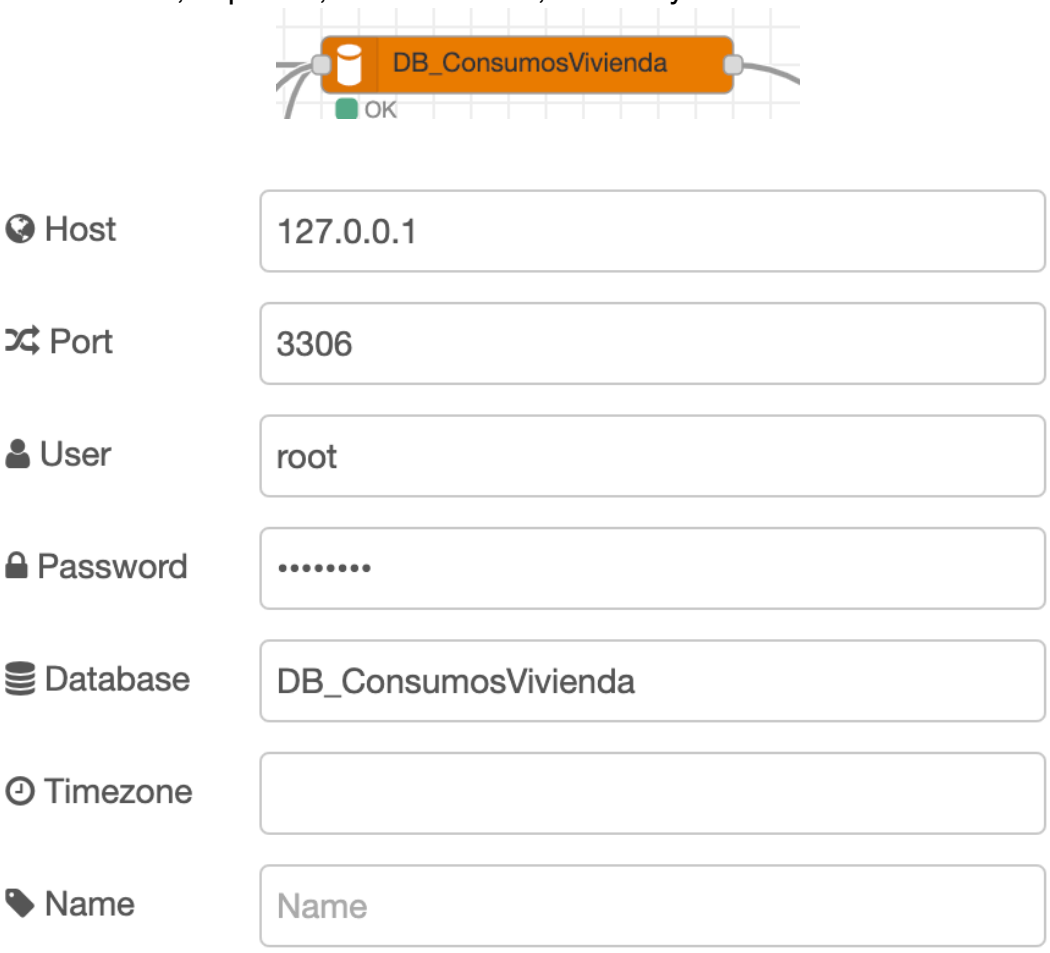


Figura 44. Nodo Mysql y su configuración.

- Nodo de tabla, mostrará la tabla con los valores de consumos, fecha, hora que se encuentran en la base de datos, de forma que se puedan consultar en cualquier momento. Para configurarla hay que introducir el nombre exacto de las columnas que hay en MySQL de forma que la lectura se realice correctamente.

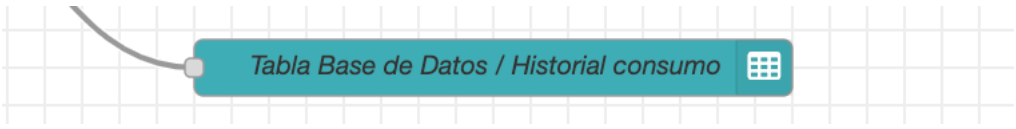


Figura 45. Nodo de Tabla.

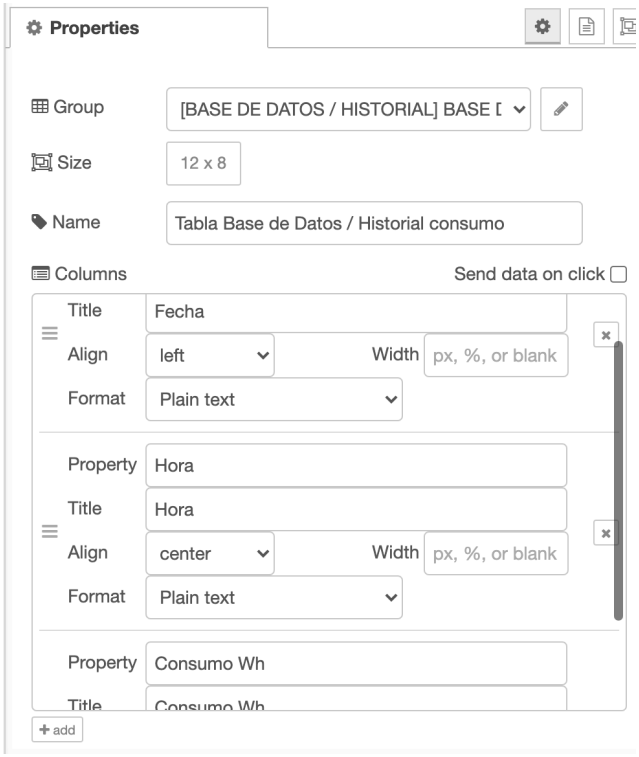


Figura 46. Configuración nodo de tabla.

BASE DE DATOS

Fecha	Hora	Consumo Wh
03/07/2020	14:00	650
03/07/2020	16:00	600
03/07/2020	18:00	900
03/07/2020	20:00	1650
03/07/2020	22:00	2400
04/07/2020	0:00	9525
04/07/2020	2:00	75
04/07/2020	4:00	75
04/07/2020	6:00	100
04/07/2020	8:00	1350

Figura 47. Tabla de datos en la interfaz web.

Cabe comentar, que se ha podido experimentar las facilidades que da Node-Red ya que se empezó a desarrollar una base de web usando código html sin haberlo estudiado nunca, me encontré con muchas dudas y problemas, llegando a obtener resultados muy simples. En la siguiente imagen se muestra la sencilla web que se llegó a obtener con el siguiente código:

Figura 48. Web con html.

```
String SendHTML(float Tension, float Intensidad, float Potencia, const int Resistencia){
  String ptr = "<!DOCTYPE html> <html>\n";
  ptr +="<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";
  ptr +="<title>ESP32 Web Server</title>\n";
  ptr +="<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}\n";
  ptr +="body{margin-top: 50px;} h1 {font-size: 2.5rem;color: #0080FF;margin: 25px auto 15px;text-align: center;text-decoration: underline;}\n";
  ptr +="h2 {color: #000000;margin-bottom: 10px;text-align: center;} p {color: #000000; text-align: left;} table {width:100%;margin-bottom: 80%;}";
  ptr +="</style>\n";
  ptr +="</head>\n";
  ptr +="<body>\n";
  ptr +="<div id=\"webpage\">\n";
  ptr +="<h1><b>ESP32 Web Server Gr&aacuteficas</b></h1>\n";

  ptr +="<table>\n";
  ptr +="<tr>\n";
  ptr +="<h2>\n";
  ptr +="<th><b><br>TENSI&OacuteN: </b>";
  ptr +=(float)Tension;
  ptr +="V </th><th></br><b>INTENSIDAD: </b>";
  ptr +=(float)Intensidad;
  ptr +="A </th><th><br><b>POTENCIA: </b>";
  ptr +=(float)Potencia;
  ptr +="W</th></br>";
  ptr +="</h2>\n";
  ptr +="</tr>\n";
  ptr +="</table>\n";

  ptr +="<p>*Valor de la Resistencia: ";
  ptr +=(float)Resistencia;
  ptr +=" Ohmios</p>";

  ptr +="</div>\n";
  ptr +="</body>\n";
  ptr +="</html>\n";
  return ptr;
}
```

Figura 49. Código html en arduino para la Web.

7.5. BASE DE DATOS MYSQL

Para guardar los datos de los consumos se va a hacer uso de una herramienta conocida en el mundo del IoT y de el tratamiento de datos llamada Mysql.

7.5.1. Explicación Mysql:

Mysql fue desarrollada en 1994 por una empresa sueca llamada MYSQL AB, tras haber pasado por manos de varios compradores, desde 2010 es propiedad de Oracle, empresa dedicada a la industria de las tecnologías de la información.

Centrándose en la definición de qué es Mysql, se podría decir que es un sistema de gestión de bases de datos relacionales de código abierto, con un modelo cliente-servidor. Si se disgrega la definición se encuentran varios conceptos importantes a la hora de comprender qué es Mysql, los cuales son: base de datos, código abierto y modelo cliente-servidor.

En primer lugar, una base de datos se puede definir como una colección de datos que se encuentran estructurados, es decir, un lugar en el que hay información almacenada y organizada. Como en la definición de Mysql se ha observado, base de datos va seguida de la palabra relacional, esto quiere decir, que la información almacenada se encuentra en forma de tabla, de forma que hay una relación dentro de las tablas, a las bases de datos relacionales, se les suele llamar RDBMS, y aquellas que no son compatibles con el modelo relacional, se les llama DBMS.

En segundo lugar, cuando se habla de código abierto, significa que el código es libre de ser usado y modificado por cualquiera. Esto quiere decir que, si hubiera necesidades específicas en un proyecto, el código fuente se podría modificar para cumplir con las especificaciones deseadas. Cabe tener en cuenta que, la Licencia Pública de GNU (GPL), establece los límites o bases de lo que se puede realizar según unas condiciones establecidas. Aunque, si se necesita una versión más flexible y un soporte más avanzado, hay una licencia comercial disponible.

Por último, el modelo cliente-servidor funciona de la siguiente manera: los ordenadores que ejecutan este software son los llamados clientes. Estos clientes siempre que requieren de información o datos, se conectan al servidor, de ahí nade la relación cliente-servidor. Exactamente igual que la comentada en el apartado de “Protocolo de comunicación MQTT”.

7.5.2. Entorno Mysql WorkBench

Antes de crear una base de datos, se debe estar conectado al servidor. Abriendo la aplicación de Mysql Workbench, seleccionaremos la ventana *conectar con base de datos,* en la que se introducirá la dirección del servidor, el puerto (3306 por defecto), además del usuario, y posteriormente deberemos introducir la contraseña. Una vez conectados al servidor, se puede proceder a trabajar con las bases de datos de Mysql.

La base de datos que se ha creado se llama “DB_ConsumosVivienda”, esta base de datos tiene en su interior cinco columnas, ID, Fechas, Hora y Consumo Wh. El ID se encarga de darle un nombre a cada fila cada vez que se le añade una, es autoincremental. En las columnas Fecha, Hora y Consumo Wh, se introduce como su propio nombre indica, el día en que se ha introducido el dato, la hora en que se ha recibido el dato y el valor del consumo recibido.

Estas variables se han definido al crear la tabla, siendo de la siguiente forma:

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
Consumo Wh	int		YES			select,insert,update,
Fecha	varchar(64)		YES	utf8mb4	utf8mb4_090...	select,insert,update,
Hora	varchar(64)		YES	utf8mb4	utf8mb4_090...	select,insert,update,
ID	int		NO			select,insert,update,

Figura 50. Configuración columnas base de datos Mysql.

A la base de datos de Mysql se accede mediante el paquete Mysql Workbech, que realiza la función de interfaz. En la se pueden crear nuevas bases de datos, tablas en su interior, configurarlas, además de visualizar los datos que contenga cada base. Se observa un menú donde se pueden desplegar diferentes opciones dentro de una base de datos. Seleccionando cada parte se visualiza el interior de la base de datos. En la parte superior central, hay diferentes ventanas con aquello que se ha seleccionado, en la parte inmediatamente superior a esta, hay un espacio de comandos, para el caso en que se quiera escribir directamente las acciones a realizar.

En la parte inferior central, se puede observar en caso de tener la ventana de la tabla abierta, la tabla de la base de datos, donde se podrá añadir información, eliminar o introducir más filas o columnas, todo eso seguido de un clic en el botón *apply,* situado en la parte inferior derecha de la ventana.

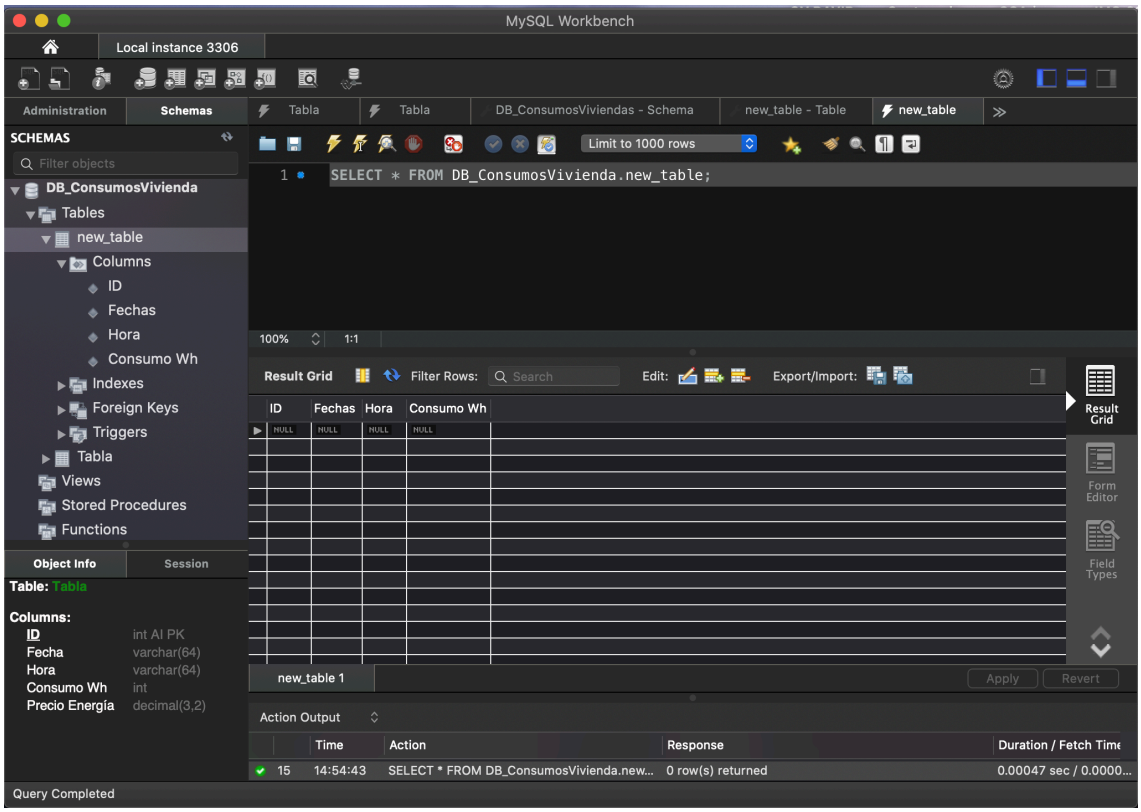


Figura 51. Espacio de trabajobase de datos Mysql.

	ID	Fecha	Hora	Consumo Wh
▶	1	03/07/2020	14:00	650
	2	03/07/2020	16:00	600
	3	03/07/2020	18:00	900
	4	03/07/2020	20:00	1650
	5	03/07/2020	22:00	2400
	6	04/07/2020	0:00	9525
	7	04/07/2020	2:00	75
	8	04/07/2020	4:00	75
	9	04/07/2020	6:00	100
	10	04/07/2020	8:00	1350
	11	NULL	NULL	NULL

Figura 46. Datos almacenados en la base de datos.

8. CONCLUSIONES.

A modo de conclusión comentar, que la aplicación desarrollada, tiene diferentes campos en los que aplicarse como se ha ido tratando de mostrar a lo largo del documento. En este caso ha sido una vivienda, pero podría aplicarse a una industria, a monitorizar el consumo eléctrico en la agricultura, invernaderos, integrarlo en una aplicación domótica, monitorizar el consumo eléctrico de un colegio, o de un laboratorio para saber como varia en función de la carga de trabajo o época del año.

Es una aplicación escalable, variable, y ampliable tanto en número de elementos como en funcionalidades, según los requisitos que vaya a tener el proyecto en cuestión y las necesidades del cliente.

Es una aplicación que facilita la compresión de la instalación eléctrica al usuario en cuestión. Además, dota al cliente de información detallada y conocimiento acerca de su consumo eléctrico. Siendo una herramienta que permite tener información para tomar decisiones que permitan un ahorro en el consumo de energía y un ahorro económico. Satisfaciendo la, cada vez mayor, necesidad de reducir el consumo de energía, como se ha comentado en la introducción del proyecto.

Otro punto a destacar es la posibilidad de visibilizar los datos de consumo desde cualquier dispositivo que esté conectado a la red.

9. BIBLIOGRAFÍA:

https://www.espressif.com/en/products/socs
https://www.espressif.com/en/products/modules
https://www.espressif.com/en/products/devkits
https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf
https://www.factor.mx/portal/base-de-conocimiento/mqtt/
https://www.techedgegroup.com/es/blog/fundamenos-node-red
http://nodered.org/docs/user-guide/
https://es.wikipedia.org/wiki/ESP32
https://es.wikipedia.org/wiki/ESP8266
https://www.luisllamas.es/esp8266/
https://www.luisllamas.es/esp32/
https://www.luisllamas.es/comparativa-esp8266-esp32/
https://naylampmechatronics.com/blog/48_tutorial-sensor-de-corriente-acs712.html
https://www.tecsc.com.ar/insumos/plc/modular/
https://noticiasit.tincan.es/ordenador-industrial-poe-para-aplicaciones-de-seguridad-inspeccion-optica-y-edge-computing/
https://naylampmechatronics.com/sensores-corriente-voltaje/393-transformador-de-voltaje-ac-zmpt101b.html
https://www.europarl.europa.eu/factsheets/es/sheet/69/la-eficiencia-energetica

Programas:

https://www.arduino.cc
https://nodered.org
https://dev.mysql.com

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE APLICACIÓN IoT PARA LA MONITORIZACIÓN DE CONSUMOS ELÉCTRICOS EN UNA VIVIENDA

PRESUPUESTO

ÍNDICE

1. COSTE DE LOS MATERIALES.

En este apartado se va a calcular los costes correspondientes a los materiales que se han de utilizar para la realización de este proyecto se necesitan, un ESP32, cinco sensores de corriente, un sensor de tensión y además las herramientas y materiales auxiliares para la realización de la instalación. Con respecto al ordenador, en el proyecto va a verse repercutida una cantidad del valor del ordenador utilizado. Se debe calcular la amortización, para ello utilizaremos los porcentajes de la pérdida de valor que establece la Agencia Tributaria. Para el caso de para *equipos de tratamiento de la información y sistemas* está establecido en un 26%. Por lo que si el ordenador tiene un valor de 700€, a un porcentaje del 26%, se deben de amortizar 182 euros anuales como máximo. Cuanto mayor demanda de prototipos, el valor referente a la amortización del ordenador se verá disminuido. Se va a suponer que se realizan 1000 prototipos, por lo tanto, a cada uno le corresponderán, 0,182€.

MATERIALES			
Elemento	**Cantidad**	**Precio**	**Total**
Sensor de tensión	1	6,99	6,99
Sensor de corriente	5	3,35	16,75
ESP32	1	9,99	9,99
Cable alimentación y transformador para ESP32	1	9,99	9,99
Amortización del ordenador (precio 700€)	1	0,182	0,182
Envolvente protectora	1	15,00	15
Elementos conexiones/instalación	1	8,00	8,00
		Total €	**66,90**

Tabla 4. Coste de los materiales.

2. DESCOMPUESTOS

En este apartado se van a calcular los costes de cada unidad de obra, diferenciando cada actividad como: realizar la programación, instalación, en definitiva, el desarrollo del proyecto. Posteriormente se obtendrá le precio total. Para ello se tendrá en cuenta las horas empleadas para cada actividad y el precio al que se cobra la hora. Para obtener el precio por hora de un ingeniero junior, se han consultado diferentes páginas web, y preguntado a recién titulados que están trabajando en distintas empresas. Según un articulo publicado por la Universidad de Alcalá, en España se trabaja 1.687 horas anuales. Teniendo en cuenta que el salario básico bruto que se establece en el convenio colectivo del sector eléctrico, Anexo I del BOE núm. 188, es de 23.310,93€, la hora de trabajo se va a cobrar a 13,81€.

ESTUDIO PREVIO			
Elemento	**Horas**	**Precio/H**	**Importe**
Estudio de necesidades	16	13,81	220,96
Estudio de soluciones	19	13,81	262,39
Planteamiento de soluciones	18	13,81	248,58
Estudio previo	53	13,81	731,93

Tabla 5. Coste del estudio previo.

PROGRAMACIÓN ARDUINO			
Elemento	**Horas**	**Precio/H**	**Importe**
Conexión WiFi	13	13,81	179,53
Conexión MQTT	13	13,81	179,53
Lectura Sensores	12	13,81	165,72
Desarrollo código	30	13,81	414,3
Comunicaciones/Envio de datos	40	13,81	552,4
Programación Arduino	108	13,81	1491,48

Tabla 6. Coste de la programación Arduino.

PROGRAMACIÓN NODE-RED			
Elemento	**Horas**	**Precio/H**	**Importe**
Diseño Web	10	13,81	138,1
Mostrar información	12	13,81	165,72
Gráficas	15	13,81	207,15
Conexión base de datos	25	13,81	345,25
Comunicaciones/Recepción de datos	35	13,81	483,35
Programación Node-Red	97	13,81	1339,57

Tabla 7. Coste de la programación Node-Red.

INSTALACIÓN			
Elemento	**Horas**	**Precio/H**	**Importe**
Comprobación del material	1,5	13,81	20,715
Montaje	6	13,81	82,86
Instalación	7,5	13,81	103,58

Tabla 8. Coste de la instalación.

REDACCIÓN DEL PROYECTO			
Elemento	**Horas**	**Precio/H**	**Importe**
Memoria	30	13,81	414,3
Anexos	2	13,81	27,62
Pliego de condiciones	4	13,81	55,24
Presupuesto	5	13,81	69,05
Presentación	4	13,81	55,24
Redacción del proyecto	45	13,81	621,45

Tabla 9. Coste de la redacción del proyecto.

ENSAYOS Y VERIFICACIONES			
Elemento	**Horas**	**Precio/H**	**Importe**
Comprobación del montaje	1,5	13,81	20,715
Prueba del montaje	2,5	13,81	34,525
Verificación del código	2	13,81	27,62
Ensayo de la aplicación	6	13,81	82,86
Ensayos y verificaciones	12	13,81	165,72

Tabla 10. Precio descompuestos.

En la siguiente tabla se muestra el presupuesto total de la mano de obra:

MANO DE OBRA			
Elemento	**Cantidad**	**Precio**	**Importe**
Estudio previo	1	731,93	731,93
Programación Arduino	1	1491,48	1491,48
Programación Node-RED	1	1339,57	1339,57
Instalación	1	103,575	103,58
Redacción del proyecto	1	621,45	621,45
Ensayos y verificaciones	1	165,72	165,72
		Total €	**4453,73**

Tabla 11. Coste de la mano de obra

A estos gastos habrá que sumarles un 8% de gastos generales, donde entran los gastos propios de la empresa y/o trabajador, como puede ser teléfono, transporte, agua, electricidad…

Gastos Generales 8%	356,30

3. BENEFICIO.

El margen de beneficio viene determinado por diferentes factores, además puede llegar a estar establecido dependiendo del mercado. En este caso se ha decidido aplicar un 20% de beneficio que se aplica sobre los costes sumados anteriormente, como se muestra en la siguiente tabla:

TOTAL	
Elemento	**Coste**
Materiales y mano de obra	4520,63
Gastos Generales 8%	356,30
Beneficio industrial 25%	1219,23
Total €	**6096,16**

Tabla 12. Beneficio Industrial y precio total sin IVA.

Finalmente, a este precio total debemos sumarle el 21% de I.V.A que tenemos actualmente en nuestro país, por lo que el coste total ascendería a 7.376,35€.

IVA 21%	**1280,19**
Total con IVA	**7376,35**

Tabla 13. IVA y precio total con IVA.

Este presupuesto es para realizar una unidad de producto, en caso de que se quisiera presupuestar un número mayor, el precio por unidad se vería reducido debido a que el coste de ingeniería se repartiría entre el número de unidades que se fuera a producir. Además, el coste de adquisición de los materiales disminuiría, debido a que los pedidos serían de mayor tamaño. De esta forma, repartiendo el coste de ingeniería entre los 1000 productos finales, obtenemos un coste de ingeniería, como la suma del coste de mano de obra y los gastos generales (4.819,03€), por producto, de 4,45€. A esto le sumamos el precio de los materiales, que, para pedidos de 1000 unidades, puede reducirse en un 50%. El precio del producto final con el beneficio industrial sería de: 47,37€. Incluyendo el IVA, el precio aumentaría hasta 57,32€.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

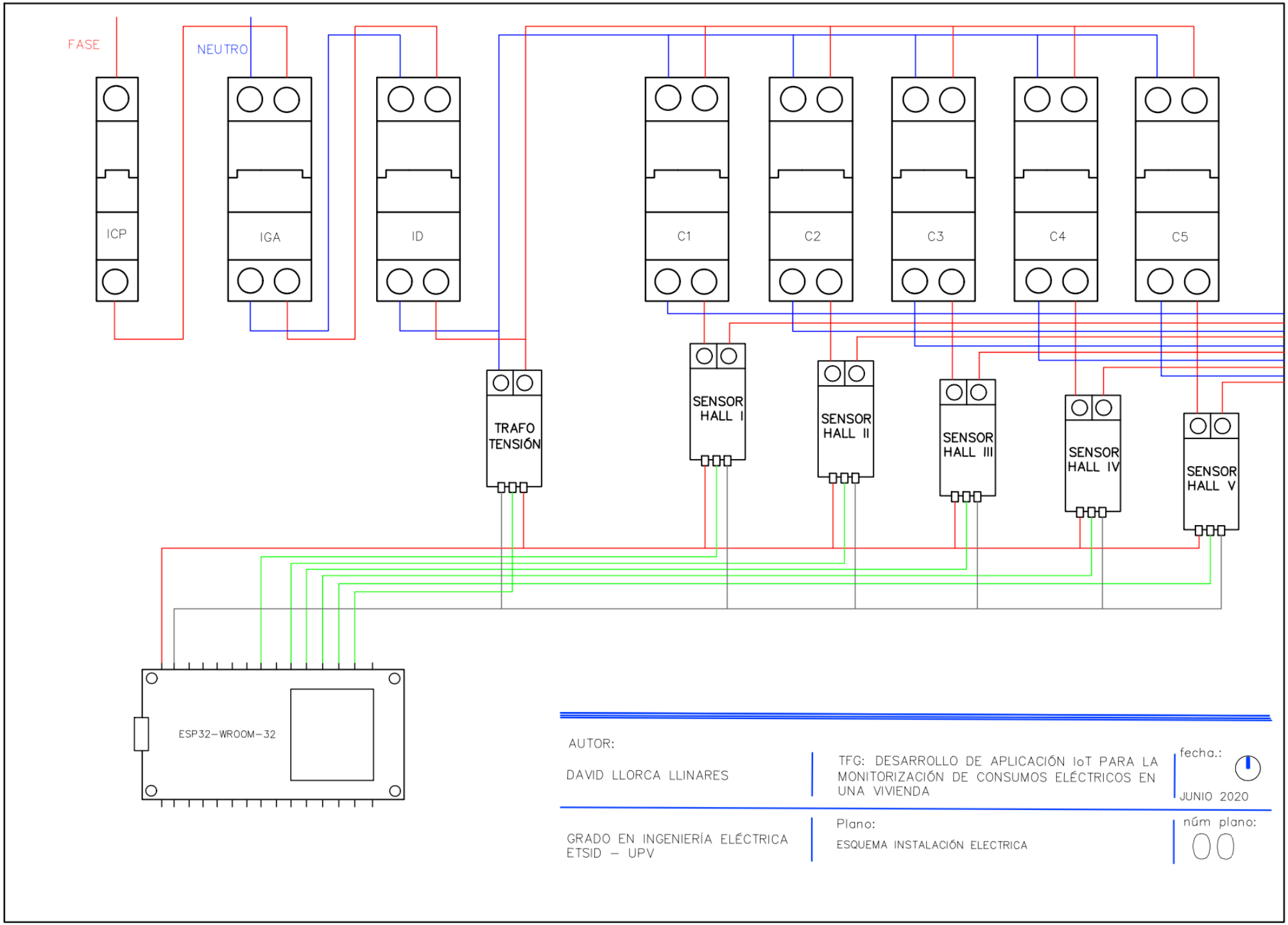
Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE APLICACIÓN IoT PARA LA MONITORIZACIÓN DE CONSUMOS ELÉCTRICOS EN UNA VIVIENDA

PLANOS



AUTOR:

DAVID LLORCA LLINARES

TFG: DESARROLLO DE APLICACIÓN IoT PARA LA MONITORIZACIÓN DE CONSUMOS ELÉCTRICOS EN UNA VIVIENDA

fecha.:

JUNIO 2020

GRADO EN INGENIERÍA ELÉCTRICA
ETSID – UPV

Plano:

ESQUEMA INSTALACIÓN ELECTRICA

núm plano:

00

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE APLICACIÓN IoT PARA LA MONITORIZACIÓN DE CONSUMOS ELÉCTRICOS EN UNA VIVIENDA

PLIEGO DE CONDICIONES

ÍNDICE

1. CONDICIONES GENERALES

1.1. Objetivo del pliego

Este documento tiene como principal objetivo definir los requerimientos y reglas relacionadas, económicas, legales, potestativas y técnicas del proyecto, siendo una extensión del contrato entre propiedad y contratista. Este documento se ajustará en los aspectos técnicos que abarcan el presente proyecto. Se realizará una descripción de los procesos constructivos y productos manejados.

Este documento abarca cuatro tipos básicos de condiciones:

- Condiciones técnicas: hacen referencia a los trabajos que hay que realizar, las características y calidad de los materiales, cuidados especiales y detalles concretos a tener presente durante la ejecución, y a los controles y ensayos de calidad preceptivos.
- Condiciones facultativas: hacen referencia a los derechos y obligaciones de las partes y sus representantes en el momento de ejecutar el proyecto.
- Condiciones económicas: hacen referencia a las garantías, la formación de precios, las formas de abono y las indemnizaciones por incumplimiento.
- Condiciones legales: hacen referencia al perfil de contratista, la forma de adjudicación, el tipo de contrato, la obligatoriedad de suscripción de seguros de responsabilidad civil y otros asuntos relacionados.

Este documento, tomando de base las condiciones anteriores, realizará una descripción detallada de la normativa legal a la que está sujeta el proyecto, y la seguridad y calidad tanto del proceso de montaje como de la ejecución del mismo.

En el caso de que apareciera algún tipo de contratiempo que no esté reflejado en el documento durante la instalación, montaje, puesta a punto o utilización del sistema, es imprescindible que se consulte con el proyectista para solucionar el problema en cuestión.

Al tratarse de un proyecto educacional con el objetivo de la obtención del título de ingeniero técnico por parte del proyectista muchos de los apartados expuestos a continuación carecen de sentido, ya que se ha construido un prototipo, por lo tanto, para la redacción del pliego de condiciones se supone y se explica qué normas se deberían cumplir en el caso de que la fabricación e instalación se llevara a cabo.

1.2. Normativa vigente

Elementos de este proyecto como los sensores, se encuentran directamente conectados a la red de corriente alterna (220v, 50Hz), por ese motivo el proyecto se rige según las normas del Reglamento Electrotécnico de Baja Tensión (RBT) y sus Instrucciones Complementarias. Además, se deben tener en cuenta las normas UNE y DIN.

Las normas son las siguientes:

- ITC-BT-18: instalaciones de puesta a tierra.
- ITC-BT-19: instalaciones interiores o receptoras. Prescripciones generales.
- ITC-BT-20: instalaciones interiores o receptoras. Sistemas de instalación.
- ITC-BT-22: instalaciones interiores o receptoras. Protección contra sobreintensidades.
- ITC-BT-23: instalaciones interiores o receptoras. Protección contra sobretensiones.
- ITC-BT-24: instalaciones interiores o receptoras. Protección contra contactos directos e indirectos.
- ITC-BT-51: instalaciones de sistemas de automatización, gestión técnica de la energía y seguridad.
- ITC-BT-43: instalación de receptores. Prescripciones generales.
- UNE 20 514 1M: reglas de seguridad para aparatos electrónicos y aparatos con ellos relacionados de uso doméstico o uno general análogo conectado a una red de energía.

2. CONDICIONES A SATISFACER POR LOS COMPONENTES

2.1. Ordenador personal

El ordenador que vaya a ser empleado puede ser tanto un ordenador de mesa como un ordenador portátil. Deberá contar con una CPU, un monitor o pantalla, un ratón (opcional en caso de tener trackpad) y un teclado. Debido a su utilización en el hogar se deberá tener en cuenta el lugar en que va a ser situado dentro de la vivienda.

Se deben tomar las precauciones ambientales de humedad y temperatura requeridas por cualquier otro equipo electrónico de estas características, respondiendo a la norma DIN 40050IP20. La temperatura de trabajo de este elemento estará comprendida entre los 0ºC y los 60ºC.

El ordenador puede ser utilizado además de para esta aplicación, para otras funciones cotidianas. Las características mínimas que debe tener para poder ser utilizado en esta aplicación son las siguientes:

-Microprocesador Intel core 2 duo a 1.8GHZ.
-Memoria RAM de 1GB.
-Espacio libre en el disco duro de 10Gb.
-Sistema operativo Windows 7 o MacOS X 10.14.

2.2. Microcontrolador

El microcontrolador estará alimentado por una fuente de alimentación de corriente continua de 5V conectada a la red. Está colocado sobre una protoboard o aquel elemento destinado para habitar al microcontrolador que mejor se ajuste al espacio y lugar de la instalación y este se conectará a tierra.

2.3. Transformador

El transformador es un elemento muy importante ya que se encargará de dar la tensión adecuada al microcontrolador. Se deberá de situar en un lugar cercano a donde se va a realizar la instalación eléctrica, cerca del cuadro eléctrico. Debe tener unas características que le permitan tener una tensión de salida de 5,1V aproximadamente.

2.4. Sensores

Los sensores de corriente y el sensor de tensión deben de estar alimentados con tensiones de entre 4,5-5V y 3,3-5V respectivamente. El sensor de tensión debe tener una tensión alterna de entrada máxima de 250V, respecto al sensor de corriente, puede soportar hasta 30A de entrada.

2.5. Conductores

Para evitar riesgos innecesarios, del tipo sobrecargas, cortocircuitos, daños físicos al usuario y deterioros del sistema, se van a utilizar todos los conductores asilados mediante un material aislante PVC o goma. Los cables que unan los sensores al microcontrolador deberán poder conducir tensiones de hasta 5V y tener una sección de 1mm2. Los cables que alimenten a los sensores deben de poder soportar como máximo 250V en corriente alterna.

La longitud de los cables dependerá de la distancia de los puntos a cablear, deberán estar debidamente apantallados, y protegidos frente a sobretensiones y cortocircuitos.

Además, se debe tener en cuenta que todos los elementos utilizados y a los que se hace referencia en este apartado deben cumplir el RD 2267/2004: Reglamento de seguridad contra incendios en los establecimientos industriales, por el que los cables deberán ser no propagadores de incendio y con baja emisión de humo y opacidad reducida.

2.6. Software

Para el desarrollo del código del proyecto se empleará el software de Arduino. Para la visualización de la información mediante una página web se hará uso de Node-Red, y para el almacenamiento de los datos de consumo se empleará Mysql. Siempre se utilizaría la última versión de cada software en caso de ser compatible con el código original. En caso de que por necesidades de proyecto se tuvieran que utilizar otros softwares o lenguajes, se realizaría una adaptación del código.

3. CONDICIONES CONSTRUCTIVAS

3.1. Proceso de soldadura

Para realizar las soldaduras se va autilizar estaño de 0.8mm o 1mm de diámetro con cinco almas de resina y colofina desoxidante en su interior. La aleación de dicho material será de un 60% de estaño y un 40% de plomo.

El soldador a emplear, tendrá una potencia de entre 30W y 40W y su temperatura máxima no excederá los 190°C. En el caso de soldar semiconductores con este elemento no se deberá superar los cinco segundos continuadamente en cada patilla.

Una vez completado el proceso de soldadura, se deberá realizar una revisión visual de las soldaduras en detalle, fijándose en si ha habido algún desperfecto o si se ha dado el efecto de “soldadura fría”, o si por el contrario se ha realizado correctamente. Posteriormente, se chequearán las conexiones realizadas mediante un multímetro que mida la continuidad de las uniones entre los elementos.

3.2. Montaje eléctrico

El montaje eléctrico de la base en la que se encuentran todos los dispositivos electrónicos se realizará de forma que los cables de transmisión de datos, los eléctricos y los de los equipos, vayan por el interior de las canaletas. De forma que todos los cables puedan circular por ellas y lleguen a todos los equipos instalados. Se debe intentar mantener una distancia suficiente entre los cables de alimentación y aquellos destinados a la transmisión de datos, para evitar problemas en la transmisión de estos.

Respecto a la longitud de los cables, esta no debe ser excesivamente larga para que no se produzcan problemas como enganchones o enredos, por lo que deben tener la longitud adecuada para que el montaje sea limpio y funcional.

4. PRUEBAS DE FUNCIONAMIENTO

Antes de realizar la conexión a la red, se realizarán varios ensayos para verificar que todo está colocado y conectado correctamente. Todas estas pruebas se deben hacer teniendo en cuenta la normativa expuesta en el RBT en la ITC-BT-05: verificaciones e inspecciones.

4.1. Revisión visual y de continuidad

Esta revisión consiste en verificar que todos los componentes y elementos se encuentran bien conectados, y que sus conexiones y cableado están bien sujetos y en su lugar correspondiente. Además, se verificarán como se ha explicado anteriormente todas las soldaduras.

4.2. Pruebas en tensión

Las pruebas en tensión consisten en conectar todos los dispositivos a la alimentación y comprobar que los dispositivos de seguridad funcionan correctamente. Tras haber comprobado los elementos de seguridad se comprobará que el resto de los componentes están en pleno funcionamiento y realizan su función correctamente.

4.3. Prueba final

Se comprobarán las conexiones wifi y la conexión con el servidor. En siguiente lugar, se verificará el correcto funcionamiento de las lecturas de los sensores instalados y el almacenamiento correcto en la base de datos. En caso de que todo transcurra con total normalidad, las pruebas de ensayo se darán por finalizadas y con ello el proceso de montaje y puesta a punto. En este punto el sistema está listo para funcionar, y dar servicio al usuario.

5. CONTROL DE CALIDAD

Dado que todos los dispositivos empelados en este proyecto son fabricado por una empresa en serio, pasan las comprobaciones y test de calidad que les exige la normativa a los fabricantes, por lo que el no entra dentro de las competencias del contratista realizar verificaciones de estos materiales. Aunque, antes de utilizar cada dispositivo, se realizará una comprobación visual, asegurándose de que no presentan daños visuales, ya que esto podría afectar al posterior funcionamiento del sistema.

Una vez instalados todos los dispositivos del sistema se verificará que no existe contacto alguno entre ellos, y que no hay derivaciones a tierra o cortocircuitos. Si los dispositivos están ubicados dentro de un armario, tampoco deberán tener contacto con la superficie de éste, además, se instalará un sistema de ventilación para la correcta refrigeración de los dispositivos, de forma que se mantenga una temperatura interna inferior a 50ºC.

6. CONDICIONES DE EJECUCIÓN

Para proceder con el montaje de los dispositivos del sistema, se debe disponer de todos los elementos, materiales auxiliares, y herramientas para el montaje necesarios, asegurándose de cumplen con los ensayos, verificaciones y comprobaciones comentadas anteriormente. En caso de querer utilizar este proyecto en serie, sería necesario revisar el proceso de ejecución y montaje de este, para mejorar la calidad, productividad y funcionalidad del mismo.

7. COMPRA DEL MATERIAL

Cuando se va a realizar la compra de material, se debe hacer un estudio de todos los proveedores que tengan material necesario, analizando, la calidad, el tiempo de recepción y los precios, para de esta forma llegar a un punto intermedio en el qué el producto obtenido se ajuste a las necesidades, al presupuesto y a las exigencias de calidad.

En el caso de este proyecto, al tratarse de un prototipo con intención educativa, se tratará de realizar el mínimo gasto posible, por lo que los dispositivos utilizados serán los de menor precio.

Si se deseara proceder a la comercialización y desarrollo del prototipo, se deberá realizar un estudio de mercado, conociendo todas las variables que vana influir en el resultado del producto (precio, tiempos de espera, calidad, impuestos aduanas, clientes, competencia…). Es muy importante contar con un equipo multidisciplinar para que la puesta en marcha de la comercialización de este proyecto pudiera ser exitosa.

Todos estos problemas no son objeto de estudio de este proyecto ya que en él, solo se realiza el montaje de un sistema y deberán ser solucionados en su caso por la empresa que desee montarlo y comercializarlo.

8. CONDICIONES ECONÓMICAS

8.1. Variaciones y mejoras del producto inicial

Como se ha comentado desde el principio, la finalidad de este proyecto es meramente educacional por lo que no existe ninguna necesidad de mejorar el proceso. Sin embargo, centrándose en la fabricación y comercialización, el propio fabricante debería ser quien desarrolle variaciones que incluyan mejoras, ya sea en prestaciones, tamaño, vida útil, entre otras posibles mejoras. En caso de que se realizara algún cambio se debería de consultar con el proyectista para que lo estudiara y posteriormente se deberían de replantear y renegociar las condiciones del contrato inicial.

8.2. Pagos de los trabajos

A continuación, se procede a establecer las condiciones a seguir por el contratista, el proyectista y el contratante.

- Si el pago se produce entre los cinco días naturales posteriores a la recepción definitiva del prototipo, inclusive con antelación a los mismos, el importe no sufrirá recargo alguno.
- Cuando el pago se efectúa entre seis y treinta días naturales después de la recepción del producto definitivo por parte del contratante, el producto sufre un recargo del 2% sobre el precio inicial previsto.
- Si se efectúa el pago entre los treinta uno y sesenta días posteriores a la recepción definitiva, este sufre un recargo del 4% sobre el importe establecido.
- Para un aplazamiento del pago entre los sesenta uno y noventa un días tras la recepción definitiva, el producto sufre un recargo del 6,5% sobre el precio final.
- Si se retrasase el pago entre los noventa uno y trescientos días naturales seguidos después de la entrega del producto, el precio de este sufriría un recargo del 30% sobre el presupuesto inicial.
- En caso de que el contratante optara por pagar a plazos, debe comunicarlo con anterioridad y establecerse a priori el número de plazos y el intervalo de tiempo entre cada pago fraccionado, sufriendo cada plazo un recargo en función del tiempo transcurrido y de acuerdo con los puntos anteriores.

- En caso del incumplimiento del pago en el término de tiempo escogido por el cliente, el fabricante tendrá derecho a demandarle ante los tribunales.

9. CONDICIONES LEGALES

9.1. Contrato

El contrato siempre se realizará por escrito, deberá cumplir todos los requisitos legales, estar firmado por todas las partes implicadas y debe quedar claramente expresado el precio inicial de fabricación y el coste de la unidad del producto, tanto en letra como en número. En el caso que procediera, a estas tarifas se les podrían aplicar recargos establecidos en los apartados anteriores del Pliego de Condiciones.

9.2. Adjudicación de la contrata

La empresa que realiza la adquisición del producto, será la que se encargue de elegir la empresa que elabore el proyecto así como su instalación, su programación y la comprobación del correcto funcionamiento.

9.3. Arbitraje y jurisdicción

En el caso de que se produjera algún desentendimiento entre las dos partes y no pudieran resolverlo entre sí, sería la dirección facultativa la encargada de arbitrarlas. Si no se llegara a un acuerdo, se procedería a presentar un técnico por cada parte que trabajaría para llegar a un acuerdo. Si las partes siguieran en desacuerdo después de haber entablado estas conversaciones, se deberá recurrir a los tribunales ordinarios de justicia de la ciudad que corresponda.

9.4. Impuestos

A la contrata se le será exigida que esté al corriente de pago de los impuestos, tasas y contribuciones necesarios para el desarrollo de la actividad empresarias.

Para la comercialización del producto se añadirá un impuesto sobre el precio de este, que corresponde al impuesto sobre el valor añadido (I.V.A.) que está estipulado actualmente en un 21%. En caso de que este variara, se modificaría para cumplir con la legislación vigente.

9.5. Rescisión del contrato

El contratante podría rescindir el contrato, en caso de que se produjera una demora excesiva en el tiempo de entrega del producto final fuera del plazo de la entrega, sin previo aviso por parte del contratista,

10. CONDICIONES FACULTATIVAS

El objetivo de este proyecto es la monitorización de los consumos eléctricos de una vivienda, a través de la tecnología IoT, utilizando sensores y microcontroladores. Además, este proceso se regirá por el protocolo MQTT, y será monitorizado por una página web haciendo uso de red WiFi y bases de datos.

En caso de que el proyecto se implantara dentro de un sistema industrial para su comercialización se deberían revisar los elementos para adaptarlos al entorno en el que van a estar trabajando.

11. DERECHOS Y DEBERES DEL CONTRATISTA

En primer lugar, el contratista debe tener pleno conocimiento de las leyes aplicables a su actividad profesional, para el correcto cumpliendo y desarrollo de la actividad. Además, debe de conocer las especificaciones de los elementos del proyecto y tener un conocimiento global del sistema.

Se ha de encargar de realizar las verificaciones que sean necesarias para el buen funcionamiento de los circuitos y dispositivos interconectados. Por otro lado, ha de disponer del Reglamento de Baja Tensión, obligándole a cumplir rigurosamente las normas establecidas. También, ha de cumplir con las especificaciones que establece la totalidad del proyecto.

La empresa contratista queda al margen de las consecuencias derivadas, de la responsabilidad por parte de la empresa contratante de obtener todos los permisos que sean de carácter obligatorio.

En caso de darse un retraso en el proceso de fabricación por causas justificadas, tratándose de causas ajenas a la empresa contratista, deberá ser aceptado por el contratante, no teniendo derecho a la reclamación por daños y perjuicios. Por contrario, en caso de una demora no justificada debidamente, podrá suponer el pago de un 5% del importe total de la fabricación y programación por cada fracción de retraso temporal que se haya estipulado en el contrato de las partes.

La empresa contratante, se comprometerá a proporcionar todas las facilidades posibles al contratista para que el proyecto se realice rápida y precisamente.

En caso de realizar variaciones o mejoras sustanciales respecto del proyecto encargado, deberán ser consultadas con el técnico diseñador.

Las características de los componentes utilizados en el desarrollo del proyecto deberán ser especificados en la memoria, teniendo en cuenta su colocación y uso. Durante el tiempo que se estima el montaje, el técnico proyectista puede anunciar la suspensión momentánea si lo considera oportuno.

El contratista tiene derecho a tener por escrito las especificaciones, los planos completos y cualquier otro documento del proyecto que pueda ser relevante en el desempeño de sus funciones.

Si la empresa instaladora incumple con las comprobaciones básicas, el proyectista queda fuera de la responsabilidad derivada del mal funcionamiento del sistema debido a dicho incumplimiento, por lo que queda fuera de su competencia.

Una vez la propiedad y el contratista estén comprometidos a cumplir con las clausulas establecidas en el contrato, la contratación del proyecto será valida, por lo que se procederá a firmar los documentos necesarios.

Desde el momento en que el proyecto se pone en funcionamiento tras haber sido comprobado que el funcionamiento del sistema es correcto, se considera que los servicios de la empresa contratista han finalizado.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE APLICACIÓN IoT PARA LA MONITORIZACIÓN DE CONSUMOS ELÉCTRICOS EN UNA VIVIENDA

ANEXOS

ÍNDICE

1. ANEXO I: Instalación de Arduino.

En primer lugar, se debe proceder a la descarga del programa en la siguiente página web: https://www.arduino.cc/en/Main/Software , teniendo en cuenta el sistema operativo del dispositivo en cuestión, Arduino se encuentra disponible para Windows, Mac y Linux.

En caso de que el sistema operativo sea Windows, nos deja descargarlo utilizando Windows Installer, o mediante la versión comprimida en zip, que es una versión portable o para aquellos que no tengan privilegios suficientes para instalar aplicaciones o simplemente quien quiera hacer una instalación manual. Si se hace uso de la segunda opción solamente se debe descomprimir en la carpeta deseada y ya se puede proceder al uso del programa.

Empleando Windows Installer u otro sistema operativo, tras haber descargado el programa, ejecutar el instalador. En caso de haber una versión nueva disponible el instalador desinstala la anterior (este es el proceso que sigue para realizar las actualizaciones). Se debe tener en cuenta que, si se han realizado modificaciones en el directorio de instalación, estas las perderemos.

Tras una correcta instalación, ya disponemos del IDE en el ordenador, se puede ejecutar el programa y empezar a hacer uso de el.

En el caso de querer descargar librerías para su uso dentro de los programas que se vayan a desarrollar, se debe seguir el siguiente procedimiento.

Las librerías se pueden instalar de dos formas: desde la misma aplicación de Arduino o manualmente. Desde la aplicación de arduino: se debe abrir la ventana *Programa/Incluir Librería/Administrar Bibliotecas..* Se abrirá una ventana en la hay un buscador, donde se debe introducir el nombre de la librería que se desea instalar, una vez buscada y encontrada, se selecciona la versión y se hace clic en instalar. Se mostrará un cuadro de texto en el que se informa de que se debe reiniciar el programa para realizar la instalación correctamente.

En caso de realizar la instalación de una librería de forma manual, se debe descargar el archivo .zip en el ordenador y ejecutarlo. Acudiendo a la ventana *Programa/Incluir Librería/Añadir Biblioteca .ZIP* se abrirá una ventana en la que deberemos buscar el archivo descargado y seleccionado. Al igual que en el caso anterior, se debe reiniciar el programa para la correcta instalación. Las librerías se pueden encontrar en diferentes páginas web y foros. Por ejemplo: https://github.com o https://forum.arduino.cc/ .

2. ANEXO II: Instalación de Node-Red.

En primer lugar, se ha de descargar la última versión de Node.js (entorno en tiempo de ejecución multiplataforma), desde la página web oficial: https://nodejs.org/es/ . Tras ejecutar el archivo descargado, habrá que introducir la contraseña de administrador, y aceptar los valores predeterminados para completar la instalación.

Seguidamente, se debe abrir la ventana de comando (Windows) o terminal (Mac). Es importante introducir el siguiente código para confirmar que la instalación se ha realizado con éxito: `node --version && npm -version` .

A continuación, se ha de instalar Node-Red. Para ello, en la misma ventana de comando o terminal, escribimos el siguiente código: `npm install -g -unsafe-perm node red` en el caso instalarlo en un sistema operativo Windows, y el siguiente código en caso de ser Mac: `sudo npm install -g -unsafe-perm node red.`

Una vez instalado Node-Red, procedemos a abrirlo, introduciento el código: `Node-Red` en comando o terminal, se abrirá el programa.

A la hora de instalar una librería de nodos que no está de serie en Noded-Red, hay que abrir el menú situado en la parte superior derecha de Node-Red y abrir la ventana *manage palette* la cual abrirá otra ventana en la que podremos, buscar nodos por su nombre e instalarlos, tras esto automáticamente aparecerán en el desplegable de los nodos que se tienen a disposición. Para encontrar nuevos nodos, se puede hacer uso de diferentes plataformas como por ejemplo el foro de Node-Red: https://discourse.nodered.org .

3. ANEXO III: Instalación de Mysql.

Para tener Mysql funcionando a pleno rendimiento, se debe instalar en primer lugar Mysql Server y seguidamente Mysql Workbench.

A la hora de descargar Mysql Server en Windows se debe tener en cuenta que el sistema tiene que tener Microsoft .NET Framework 4.5.2, disponible para descargar en el siguiente enlace: *https://www.microsoft.com/es-es/download/details.aspx?id=42642*. Para proseguir con la descarga, se visita la página web oficial de Mysql, y se accede a la parte de descargas, donde podremos encontrar los archivos para los diferentes sistemas operativos: https://dev.mysql.com/downloads/mysql/. Tras seleccionar el que se adapta al sistema en cuestión se abrirá una página en la que se pide entrar en una cuenta de Oracle o registrarse, pero fijándose en la parte inferior da la opción de seguir con la descarga sin identificarse de ningún modo.

Una vez en este punto se ejecutará el instalador, que, tras varios procesos, da varias opciones de qué forma se quiere realizar la instalación:

- *Developer Default*: Muy específica y orientada a desarrolladores.
- *Server only*: Instala únicamente MySQL Server.
- *Client only*: Instala los clientes de consola y gráfico, pero también componentes de desarrollo.
- *Full*: Instala todo lo que trae el paquete.
- *Custom*: Permite elegir qué queremos instalar.

Las opciones de *Developer* y *Full*, son muy seguras y son las elegidas en caso de no querer demasiadas complicaciones. En cuanto a la opción *Server only*, el propio nombre indica de qué forma lo va a instalar. Si se elige la opción *Client only*, instalará, además, componentes de desarrollo como plugins, ejemplos, documentación… Y eligiendo la opción *Custom*, en una ventana a parte, se da la opción de seleccionar qué instalar.

Posteriormente el instalador revisa que todo esté correctamente y procede con la instalación de Mysql. Se sigue ejecutando el instalador hasta que la instalación esté completa.

En cuanto a Mysql workbench, se puede instalar durante el proceso anterior si se elige una instalación *Custom* en la que se selección este paquete. Si no, también es posible descargarlo desde la página web oficial: https://www.mysql.com/products/workbench/ .

El procedimiento es igual que en la instalación del Mysql Server, a diferencia que en este caso no da opciones a la hora de instalarlo, simplemente ejecuta la instalación y ya está listo para trabajar.

4. ANEXO IV: Especificaciones técnicas ESP32-WROOM-32

Procesador:

Dependiendo del modelo monta uno o dos microprocesadores Tensilica Xtensa 32-bit LX6, los cuales consiguen obtener una frecuencia de reloj de hasta 240MHz y un rendimiento de hasta 600 DMIPS. Además, incorpora un procesador ULP (Ultra Low Power) que es capaz de trabajar cuando la CPU entra en el modo *deep-sleep* (ahorro de energía). El ESP32 puede ejecutar conversiones ADC, operaciones computacionales y chequear el estado de los pines con un ínfimo consumo. El procesador ULP y la memoria RTC permanecen encendidas durante el modo *deep-sleep*. Todo esto es de utilidad para diseñar aplicaciones en las que la CPU necesita despertarse debido a un evento, de forma que se mantenga con un consumo bajo el resto del tiempo.

Memoria:

Respecto a la memoria interna, se debe que tener en cuanta que además de las siguientes características, se le puede acoplar una memoria flash externa de 16MB, o una SRAM de 8MB. El ESP32 incluye las siguientes memorias flash:

- 448 KB de memoria ROM, para las funciones de core y boot.
- 520 KB de memoria SRAM, para datos e instrucciones.
- 8 KB de memoria RTC fast SRAM de, para el almacenamiento de datos. Esta memoria, puede ser usada por el núcleo principal durante la función de boot desde el modo deep-sleep.
- 8KB de memoria RTC slow SRAM, a la que el coprocesador puede acceder durante el modo Deep-sleep.
- Memoria flash embebida, dependiendo del modelo, suele tener un tamaño de entre 0 y 4MB.

Además, el ESP32 puede encriptar la memoria flash y aceleración criptográfica hardware como AES (Advanced Encryption Standard), SHA-2 (Secure Hash Algorithm 2), RSA, ECC (Elliptic Curve Cryptography) y RNG (Random Number Generator).

Timers y Watchdogs:

Tiene cuatro temporizadores de 64 bits de uso general embebidos en el chip, que se basan en prescalers de 16 bits (de 2 a 65536). Estos temporizadores se pueden configurar en cuenta ascendente o descendente y pueden ser generados como interrupción por nivel o por flanco.

Además, el ESP32 contiene tres Watchdog Timers, que se encuentran uno en cada núcleo, llamados: Main Watchdog Timer (MWDT); y otro en el RTC, llamado: RTC Watchdog Timer (RWDT). Estos se encargan de recuperar el dispositivo ante un fallo imprevisto. Tienen cuantro etapas de acción las cuales pueden ser configuradas o inhabilitadas independientemente. Estas etapas tienen un tiempo que es programado. En estas etapas se aplican acciones como reiniciar la CPU, el núcleo o el sistema hasta que se ha expirado el tiepo de la etapa. Solo el RWDT puede ejecutar el reinicio del sistema y del propio RTC (Real Time Clock).

Relojes del sistema:

Se pueden encontrar dos tipos de relojes en el chip, el reloj de la CPU y el reloj RTC.

El reloj de la CPU, una vez se reinicia, se establece como reloj predeterminado de la CPU una fuente externa del reloj de crital de 40MHz. Esta fuente está conectada a un PLL para generar un reloj de frecuencia que normalmente es de 160Mhz. Además, el chip cuenta con un oscilador interno de 8MHz. El programa puede elegir que reloj utilizar en el sistema, si el reloj de cristal, el de PLL (Phale Lock Loop) o el oscilador interno.

PLL: reloj de sonido generado por ultra bajo sonido fraccional PLL.

En cuanto al reloj RTC, tiene cinco posibles fuentes: un reloj de cristal externo de baja velocidad (32kHz), un reloj de cristal externo divido entre cuatro, un oscilador interno de 150KHz (regulable), un oscilador interno de 8MHz y otro reloj interno de 31,25kHz derivado del anterior.

Radio:

El módulo ESP32 consta de los siguientes bloques de radiofrecuencia:

- Un receptor de 2.4 GHz que demodula la señal RF (radiofrecuencia) y la convierte a dominio digital por medio de dos ADCs de alta velocidad y alta resolución. Para adaptarse a las condiciones variables de la señal, se emplean: filtros de RF, Control Automático de Ganancia (AGC), circuitos de cancelación de corriente continua y filtros de banda base integrados en el chip.

- Un transmisor de 2.4Ghz que modula la señal de cuadratura en banda base a señal de RF de 2.4Ghz, y acciona la antena con un amplificador de potencia basado CMOS. Gracias a la calibración digital de mejora la linealidad del amplificador y permite cancelar las imperfecciones de ruido.

- Un generador de reloj que produce una señal cuadratura de 2.4Ghz para el receptor y transmisor. Todos sus componentes están integrados dentro del chip.

Conectividad: WiFi y Bluetooth.

Esta es una de las características más importantes del ESP32, la capacidad para tener conexión WiFi y Bluetooth.

Respecto al WiFi, implementa TCP/IP, soporta las tecnologías estándar protocolo WiFi MAC 802.11 b/g/n, 802.11n a 2.4 GHz y velocidad de datos de hasta 150 Mbps. El protocolo WiFi MAC aplica funciones de bajo nivel como por ejemplo 4 interfaces virtuales WiFi, protección RTS, CTS y demás. Y hablando de seguridad, el dispositivo soporta los estándares de seguridad de 802.11: WFA, WPA/WPA2 y WAPI.

En cuanto al Bluetooth, los aspectos más importantes son que soporta la version v4.2 BR/EDR y además dispone de BLE (Bluetooth Low Energy) que está diseñado para proporcionar un bajo consumo de energía y unos costes considerablemente reducidos, manteniendo un rango de alcance de comunicación similar al convencional.

Modos de funcionamiento:

El SoC puede estar en distintos modelos de operación en función de la energía utilizada y son los siguientes:

- Activo: el chip se encuentra encendido y puede recibir, transmitir y escuchar.
- Modem-sleep: la CPU está operativa y el reloj es configurable. WiFi y Bluetooth están desactivados.
- Light-sleep: la CPU está pausada. La memoria, los periféricos RTC y el coprocesador ULP de bajo consumo están activos. Cualquier evento despertará al chip.
- Deep-sleep: solo la memoria y los periféricos RTC están encendidos. Los datos de WiFi y Bluetooth se encuentran almacenados en la memoria RTC. El coprocesador ULP es funcional.
- Hibernación: el oscilador interno de 8MHz y el coprocesador ULP están deshabilitados. Solamente el RTC Timer o un evento generado por algunos pines GPIO que estén activos pueden despertar al chip.

Periféricos y sensores:

El módulo ESP32 tiene 34 pines GPIO de uso general. Hay pines de distintos tipos: digitales, digitales y analógicos, y con función capacitive-touch. La mayoría de los pines GPIO se pueden configurar como pull-up, pull-down, o setearlos alta impedancia. También los analógicos y de función capacitive-touch pueden ser configurados a su vez como digitales.

El dispositivo tiene dos conversores analógico-digital (ADC) de 12-bit SAR, que pueden soportar hasta 18 canales de entrada analógica. Además, el coprocesador ULP también está diseñado para medir tensión mientras opera en modo sleep, por lo que habilita el bajo consumo. En adición a esto, integra un conversor digital-analógico (DAC) de 8 bits que permite convertir dos señales digitales en dos señales analógicas de salida.

El dispositivo cuenta con sensores internos como, por ejemplo: un sensor de temperatura y un sensor de efecto Hall, que permite detectar si existe campo magnético cerca del dispositivo. También se encuentran sensores tipo touch, en concreto 10 GPIOs capacitivos, capaces de detectar variaciones al acercar o tocar el pin con el dedo o con un objeto, gracias a su gran sensibilidad.

En el ESP32 se puede encontrar un controlador SD/SDIO/MMC que soporta traspaso de datos desde dispositivos externos de hasta 80Mhz en tres modos diferentes de bus: 1bit, de 4bits y de 8bits. También admite dos tarjetas SD / SDIO / MMC4.41 en un modo de bus de datos de 4 bits. Es compatible con una tarjeta SD que funciona a 1,8 V.

Para la comunicación de datos serie el dispositivo cuenta con tres transmisores receptores asíncronos universales (interfaces UART), que son capaces de proporcionar comunicación asíncrona a una velocidad de hasta 5Mbps.

El ESP32 también soporta dos interfaces I^2C, que pueden usarse como maestro o como esclavo dependiendo de la configuración, a una velocidad de hasta 5MHz. Tiene modo estándar (100Kbps) y modo rápido (400Kbps).

Por otra parte, soporta comunicaciones I^2S por dos interfaces estándar, de igual modo pueden emplearse como maestro o esclavo y pueden ser configurados con diferentes resoluciones de 8 a 64 bits como entradas o salidas.

En cuanto al SPI, se proporcionan cuatro modos de funcionamiento dependiendo de la polaridad y la fase del reloj del SPI, con una velocidad de hasta 80MHz. Además, todos los SPIs pueden conectarse a una memoria flash externa o SRAM o LCD.

Otra de las características del SoC es que posee un controlador remoto de infrarrojos, con ocho canales para la transmisión y envío de datos, que soporta varios protocolos, estos ocho canales comparten un bloque de memoria

de 512 x 32 bits que almacena la forma de onda de transmisión o recepción. Por último, el ESP32 también es capaz de generar hasta dieciséis canales de modulación de ancho de pulso (PWM) usado mayormente para hacer funcionar motores digitales y luces inteligentes. También posee una interfaz para la comunicación por bus CAN 2.0.

Wifi:

Como se ha comentado en apartados anteriores el ESP32 dispone del WiFi estándar 802.11 b/g/n con una velocidad de hasta 150Mbit/s a 2.4GHz. Sumándole a esto, dispone de varios modos de seguridad como lo son WPA o WPA2, también posee escáner de puntos de acceso activo y pasivo, y un modo para la monitorización del envío de paquetes.

Es importante tener clara una serie de ideas básicas de cómo funciona el dispositivo a la hora de trabajar con aplicaciones que requieran del uso de WiFi. En general, se trata de comunicación TCP/IP sobre un enlace inalámbrico.

Por si no queda claro qué es la comunicación TCP/IP (Protocolo de control de transmisión/Protocolo de Internet), es un conjunto de protocolos que permiten la comunicación entre los ordenadores pertenecientes a una red. Para conseguir que el intercambio de datos entre dos equipos se de forma fiable, se deben llevar a cabo varios procedimientos separados. Para realizar esto de forma sencilla, este protocolo usa diferentes capas jerarquizadas, de forma que cada capa se encarga de una función diferente, dependiendo de a qué se le solicitan los servicios.

Durante las operaciones con uso del WiFi, se dan eventos que tienen que ver con el estado de la conexión y que el ESP32 tiene que tratar con un manejador de eventos. Estos eventos tienen que ver con conexiones y desconexiones, asignar IP, la forma en que el WiFi es conectado, o, si una estación se ha conectado o no.

El ESP32 permite que el WiFi pueda trabajar en diferentes modos, a continuación, los explico:

- Modo estación (STA): en este modo nuestro dispositivo se conecta a un punto de acceso (una red WiFi), en la cual puede haber más dispositivos conectados. Al establecer la conexión llegan dos eventos al ESP32, un primero que indica que el dispositivo se ha conectado a un punto de acceso, y el segundo, que indica que se ha asignado una IP con el servidor del punto de acceso.
- Modo punto de acceso (Soft AP): el ESP32 crea su propio punto de acceso y son los terminales los que se han de conectar a dicha red. En este caso debemos definir el SSID y la password que vamos a utilizar para crear el AP. Llegará un primer evento que indica que el punto de acceso

ha sido creado con éxito. Cuando una estación se conecte al AP se generará un segundo evento y le asignará una direción IP con un servidor DHCP interno. Cabe destacar que, si se conectan más de una estación a la vez a este punto de acceso, las estaciones no se podrán comunicar entre sí, sino que lo tendrían que hacer por medio del punto de acceso, ya que el ESP32 realiza conexiones directas con ellas.
- Modo combinado (AP + STA): el dispositivo actúa como punto de acceso y a la vez está conectado a otra red.

Bluetooth:

El ESP32 también puede realizar conexiones inalámbricas mediante el Bluetooth clásico y Bluetooth LE de bajo consumo. Pero para poder realizar la comunicación hay que tener en cuenta la seguridad de la conexión, esta seguridad se consigue mediante un proceso llamado *pairing*, que consiste en generar claves únicas que solamente son compartidas entre ambos dispositivos. Para establecer el protocolo de Bluetooth hay que seguir los siguientes pasos:

- Perfil de acceso genérico (GAP) se encarga de determinar cuales son los dispositivos que interactúan entre sí, controlando las conexiones y los anuncios. Existen dos formas de implementar la conexión, enviando tramas de tipo broadcast que serán escuchadas por otros clientes cercanos, o mediante la conexión directa, en la que el dispositivo se conecta directamente tras que el servidor haya mandado el broadcast.
- Perfil de atributos genérico (GATT) que describe cómo los datos son transferidos una vez se ha establecido la conexión.

Al igual que en el caso de WiFi, el ESP32 dispone de eventos sobre Bluetooth, que nos van a permitir controlar el broadcast y la recogida de información.

5. Código Arduino.

```
//********** LIBRERIAS ****************************

#include <WiFi.h>

#include <PubSubClient.h>

#include <NTPClient.h>

#include <WiFiUdp.h>

//************* DEFINIMOS NTP CLIENT PARA COGER LA HORA************

WiFiUDP ntpUDP;

NTPClient timeClient(ntpUDP);

//***********VARIABLES FECHA Y HORA******************

String formattedDate;

String dayStamp;

String timeStamp;

String minutoStamp;

//**************CONFIRMAR ENVIO**********************

bool Enviado = false;

//********** CONFIGURACIÓN RED WIFI ***************

const char* ssid = "vodafone5B08";

const char* password =  "RMNKJGKFMYLZ2Y";

//********** CONFIGURACIÓN MQTT ******************

const char *MQTT_SERVER = "192.168.0.11";

const int MQTT_PORT = 1883;

const char *MQTT_USER = "Corona";

const char *MQTT_PASS = "virus2020";

// *************SENSOR TENSIÓN********************

const byte PinSensorT = A6;

float ValorSensorT;

// *************SENSOR INTENSIDAD******************

const byte PinSensor1 = A0;

const byte PinSensor2 = A3;
```

```
const byte PinSensor3 = A4;

const byte PinSensor4 = A5;

const byte PinSensor5 = A7;

float SensibilidadSensorI = 0.066;

float ValorSensorI;

//********** VARIABLES GLOBALES   *****************

WiFiClient espClient;

PubSubClient client(espClient);

//************ TOPIC TENSIÓN, CORRIENTES Y POTENCIAS ************

const char *RAIZ_TOPIC_PUBLISH_TEF = "/local/tensiones/tensioneficaz";

const char *RAIZ_TOPIC_PUBLISH_IEF_C1 = "/local/intensidades/intensidadeficaz_C1";

const char *RAIZ_TOPIC_PUBLISH_IEF_C2 = "/local/intensidades/intensidadeficaz_C2";

const char *RAIZ_TOPIC_PUBLISH_IEF_C3 = "/local/intensidades/intensidadeficaz_C3";

const char *RAIZ_TOPIC_PUBLISH_IEF_C4 = "/local/intensidades/intensidadeficaz_C4";

const char *RAIZ_TOPIC_PUBLISH_IEF_C5 = "/local/intensidades/intensidadeficaz_C5";

const char *RAIZ_TOPIC_PUBLISH_IEF_T = "/local/intensidades/intensidadeficaz_T";

const char *RAIZ_TOPIC_PUBLISH_PA = "/local/potencias/potenciaactiva";

const char *RAIZ_TOPIC_PUBLISH_PR = "/local/potencias/potenciareactiva";

const char *RAIZ_TOPIC_PUBLISH_PAP = "/local/potencias/potenciaaparente";

const char *RAIZ_TOPIC_PUBLISH_FDP = "/local/potencias/fdp";

//***************CONSUMOS CIRCUITOS Y TOTAL*****************************

const char *RAIZ_TOPIC_PUBLISH_C1 = "/local/CC/C1";

const char *RAIZ_TOPIC_PUBLISH_C2 = "/local/CC/C2";

const char *RAIZ_TOPIC_PUBLISH_C3 = "/local/CC/C3";

const char *RAIZ_TOPIC_PUBLISH_C4 = "/local/CC/C4";

const char *RAIZ_TOPIC_PUBLISH_C5 = "/local/CC/C5";

const char *RAIZ_TOPIC_PUBLISH_CT = "/local/CC/CT";

//**********ARRAY: TENSIÓN, CORRIENTES Y POTENCIAS***************

char TensionEficazArray[10];

char IntensidadEficazC1Array[10];
```

```
char IntensidadEficazC2Array[10];

char IntensidadEficazC3Array[10];

char IntensidadEficazC4Array[10];

char IntensidadEficazC5Array[10];

char IntensidadEficazTArray[10];

char PotenciaActivaArray[10];

char PotenciaReactivaArray[10];

char PotenciaAparenteArray[10];

char FDPArray[10];

//**********ARRAY CONSUMOS CIRCUITOS Y TOTAL**************

char C1Array[10];

char C2Array[10];

char C3Array[10];

char C4Array[10];

char C5Array[10];

char CTArray[10];

//*********VARIABLES NUMÉRICAS**************************


float TensionEficaz = 0;

float TensionEficazESC = 0;

float IntensidadEficazC1 = 0;

float IntensidadEficazC2 = 0;

float IntensidadEficazC3 = 0;

float IntensidadEficazC4 = 0;

float IntensidadEficazC5 = 0;

float IntensidadEficazT = 0;

float PotenciaActivaC1 = 0;

float PotenciaActivaC2 = 0;

float PotenciaActivaC3 = 0;

float PotenciaActivaC4 = 0;
```

```
float PotenciaActivaC5 = 0;

float PotenciaActivaT = 0;

float PotenciaReactiva = 0;

float PotenciaAparente = 0;

float fdp = 0;

float C1 = 0;

float C2 = 0;

float C3 = 0;

float C4 = 0;

float C5 = 0;

float CT = 0;

//***************VECTORES TENSIÓN E INTENSIDAD*****************

// Variables muestreo

const byte N = 20;

const unsigned long FrecuenciaMuestreo = 1000; //Microsegundos

float VectorTension[N];

float VectorTensionApura[N];

float VectorTensionApura2[N];

float VectorIntensidadC1[N];

float VectorIntensidadApuraC1[N];

float VectorIntensidadApuraC1_2[N];


float VectorIntensidadC2[N];

float VectorIntensidadApuraC2[N];

float VectorIntensidadApuraC2_2[N];

float VectorIntensidadC3[N];

float VectorIntensidadApuraC3[N];

float VectorIntensidadApuraC3_2[N];

float VectorIntensidadC4[N];

float VectorIntensidadApuraC4[N];
```

```
float VectorIntensidadApuraC4_2[N];

float VectorIntensidadC5[N];

float VectorIntensidadApuraC5[N];

float VectorIntensidadApuraC5_2[N];

float VectorPotenciaAC1[N];

float VectorPotenciaAC2[N];

float VectorPotenciaAC3[N];

float VectorPotenciaAC4[N];

float VectorPotenciaAC5[N];

float VectorPotenciaACT[N];

//**************VARIABLES CÁCULOS*****************************

float sumaT = 0;

float sumaT2 = 0;

float OffsetT = 0;

float Valor = 0;

float sumaIC1_1 = 0;

float sumaIC1_2 = 0;

float OffsetIC1 = 0;

float Valor1 = 0;

float sumaIC2_1 = 0;

float sumaIC2_2 = 0;

float OffsetIC2 = 0;

float Valor2 = 0;

float sumaIC3_1 = 0;

float sumaIC3_2 = 0;

float OffsetIC3 = 0;

float Valor3 = 0;

float sumaIC4_1 = 0;

float sumaIC4_2 = 0;

float OffsetIC4 = 0;
```

```
float Valor4 = 0;
float sumaIC5_1 = 0;
float sumaIC5_2 = 0;
float OffsetIC5 = 0;
float Valor5 = 0;
float sumaPAC1 = 0;
float sumaPAC2 = 0;
float sumaPAC3 = 0;
float sumaPAC4 = 0;
float sumaPAC5 = 0;
//********** FUNCIONES ****************************
void callback(char* topic, byte* payload, unsigned int length);
void reconnect();
void setup_wifi();
//*****************SETUP*************************
void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(MQTT_SERVER, MQTT_PORT);
  client.setCallback(callback);
  timeClient.begin();
  timeClient.setTimeOffset(7200);
}
//************************LOOP************************
void loop() {
  while(!timeClient.update()){
    timeClient.forceUpdate();
  }
  formattedDate = timeClient.getFormattedDate();
  // Consulta de la fecha
```

```
int splitT = formattedDate.indexOf("T");
dayStamp = formattedDate.substring(0, splitT);
// Consulta de la hora
timeStamp = formattedDate.substring(splitT+1, formattedDate.length()-1);
// Sacamos minutos de la hora
minutoStamp = formattedDate.substring(splitT+4, formattedDate.length()-1);
if (!client.connected()) {
  reconnect();
}
if (client.connected()){
//**************CÁLCULO TENSIÓN EFICAZ******************
unsigned long startTime = micros();
for (byte i = 0; i < N; i++){
  while (micros() - startTime < FrecuenciaMuestreo){
  /*No hace nada en este tiempo*/
  }
  startTime += FrecuenciaMuestreo;
  VectorTension[i] = analogRead(PinSensorT);
}
for (int i = 0; i < N; i++){
  sumaT += VectorTension[i];
}
// Calculamos offset
OffsetT = sumaT / N; // Valor medio
for (int i = 0; i < N; i++){
  VectorTensionApura[i] == VectorTension[i] - OffsetT;  // Onda Alterna Pura
}
// Calculamos integral
for (int i = 0; i < N; i++){
  VectorTensionApura2[i] == pow(VectorTensionApura[i],2);
```

```
}

 for(int i = 0; i <N; i++){

  sumaT2 += VectorTensionApura2[i];

 }

Valor = 0,02*sumaT2;

// Tensión eficaz

TensionEficaz = sqrt(Valor);

//Escalamos la señal de tensión

TensionEficazESC = map(TensionEficaz, 0, 1023, 5, 230);

//***************CÁLCULO INTESIDAD EFICAZ C1******************

for (byte i = 0; i < N; i++){

  while (micros() - startTime < FrecuenciaMuestreo){

  /*No hace nada en este tiempo*/

  }

  startTime += FrecuenciaMuestreo;

  VectorIntensidadC1[i] = ((analogRead(PinSensor1)*(30.0/1023.0))- 2.5)/SensibilidadSensorI; // Leemos del sensor, escalamos y pasamos a corriente

}

for (int i = 0; i < N; i++){

  sumaIC1_1 += VectorIntensidadC1[i];

}

// Calculamos offset

OffsetIC1 = sumaIC1_1 / N; // Valor medio

for (int i = 0; i < N; i++){

  VectorIntensidadApuraC1[i] == VectorIntensidadC1[i] - OffsetIC1;  // Onda Alterna Pura

}

// Calculamos integral

for (int i = 0; i < N; i++){

  VectorIntensidadApuraC1_2[i] == pow(VectorIntensidadApuraC1[i],2);

}
```

```
 for(int i = 0; i <N; i++){

  sumaIC1_2 += VectorIntensidadApuraC1_2[i];

 }

Valor1 = 0.02 * sumaIC1_2;

// Tensión eficaz

IntensidadEficazC1 = sqrt(Valor1);



  //***************CÁLCULO INTESIDAD EFICAZ C2******************

for (byte i = 0; i < N; i++){

  while (micros() - startTime < FrecuenciaMuestreo){

  /*No hace nada en este tiempo*/

  }

  startTime += FrecuenciaMuestreo;

  VectorIntensidadC2[i] = ((analogRead(PinSensor2)*(30.0/1023.0))- 2.5)/SensibilidadSensorI; // Leemos del sensor, escalamos y pasamos a corriente

}

for (int i = 0; i < N; i++){

  sumaIC2_1 += VectorIntensidadC2[i];

}

// Calculamos offset

OffsetIC2 = sumaIC2_1 / N; // Valor medio

for (int i = 0; i < N; i++){

  VectorIntensidadApuraC2[i] == VectorIntensidadC2[i] - OffsetIC2;  // Onda Alterna Pura

}

// Calculamos integral

for (int i = 0; i < N; i++){

  VectorIntensidadApuraC2_2[i] == pow(VectorIntensidadApuraC2[i],2);

}

 for(int i = 0; i <N; i++){

  sumaIC2_2 += VectorIntensidadApuraC2_2[i];
```

```
 }


 Valor2 = 0.02 * sumaIC2_2;

 // Tensión eficaz

 IntensidadEficazC2 = sqrt(Valor2);

  //***************CÁLCULO INTESIDAD EFICAZ C3******************

 for (byte i = 0; i < N; i++){

  while (micros() - startTime < FrecuenciaMuestreo){

  /*No hace nada en este tiempo*/

  }

  startTime += FrecuenciaMuestreo;

  VectorIntensidadC3[i] = ((analogRead(PinSensor3)*(30.0/1023.0))- 2.5)/SensibilidadSensorI; // Leemos del sensor, escalamos y pasamos a corriente

 }

 for (int i = 0; i < N; i++){

  sumaIC3_1 += VectorIntensidadC3[i];

 }

 // Calculamos offset

 OffsetIC3 = sumaIC3_1 / N; // Valor medio

 for (int i = 0; i < N; i++){

  VectorIntensidadApuraC3[i] == VectorIntensidadC3[i] - OffsetIC3;  // Onda Alterna Pura

 }

 // Calculamos integral

 for (int i = 0; i < N; i++){

  VectorIntensidadApuraC3_2[i] == pow(VectorIntensidadApuraC3[i],2);

 }


  for(int i = 0; i <N; i++){

  sumaIC3_2 += VectorIntensidadApuraC3_2[i];

  }
```

```
Valor3 = 0.02 * sumaIC3_2;

// Tensión eficaz

IntensidadEficazC3 = sqrt(Valor3);

  //***************CÁLCULO INTESIDAD EFICAZ C4******************

for (byte i = 0; i < N; i++){

  while (micros() - startTime < FrecuenciaMuestreo){

  /*No hace nada en este tiempo*/

  }

  startTime += FrecuenciaMuestreo;

  VectorIntensidadC4[i] = ((analogRead(PinSensor4)*(30.0/1023.0))- 2.5)/SensibilidadSensorI; // Leemos del sensor, escalamos y pasamos a corriente

}

for (int i = 0; i < N; i++){

  sumaIC4_1 += VectorIntensidadC4[i];

}

// Calculamos offset

OffsetIC4 = sumaIC4_1 / N; // Valor medio

for (int i = 0; i < N; i++){

  VectorIntensidadApuraC4[i] == VectorIntensidadC4[i] - OffsetIC4;  // Onda Alterna Pura

}

// Calculamos integral

for (int i = 0; i < N; i++){

  VectorIntensidadApuraC4_2[i] == pow(VectorIntensidadApuraC4[i],2);

}

 for(int i = 0; i <N; i++){

  sumaIC4_2 += VectorIntensidadApuraC4_2[i];

 }

Valor4 = 0.02 * sumaIC4_2;

// Tensión eficaz

IntensidadEficazC4 = sqrt(Valor4);
```

```
  //***************CÁLCULO INTESIDAD EFICAZ C5*****************

 for (byte i = 0; i < N; i++){

  while (micros() - startTime < FrecuenciaMuestreo){

  /*No hace nada en este tiempo*/

  }

  startTime += FrecuenciaMuestreo;

  VectorIntensidadC5[i] = ((analogRead(PinSensor5)*(30.0/1023.0))- 2.5)/SensibilidadSensorI; // Leemos del sensor, escalamos y pasamos a corriente

 }

 for (int i = 0; i < N; i++){

  sumaIC5_1 += VectorIntensidadC5[i];

 }

 // Calculamos offset

 OffsetIC5 = sumaIC5_1 / N; // Valor medio

 for (int i = 0; i < N; i++){

  VectorIntensidadApuraC5[i] == VectorIntensidadC5[i] - OffsetIC5;  // Onda Alterna Pura

 }

 // Calculamos integral

 for (int i = 0; i < N; i++){

  VectorIntensidadApuraC5_2[i] == pow(VectorIntensidadApuraC5[i],2);

 }

  for(int i = 0; i <N; i++){

  sumaIC5_2 += VectorIntensidadApuraC5_2[i];

  }

 Valor5 = 0.02 * sumaIC5_2;

 // Tensión eficaz

 IntensidadEficazC5 = sqrt(Valor5);



 //***************CÁLCULO INTESIDAD EFICAZ TOTAL*****************

 IntensidadEficazT = IntensidadEficazC1 + IntensidadEficazC2 + IntensidadEficazC3 + IntensidadEficazC4 + IntensidadEficazC5;
```

```
//**********************CÁLCULOS**************************

for ( int i = 0; i <N; i++){

    VectorPotenciaAC1[i] = VectorTensionApura2[i] * VectorIntensidadApuraC1_2[i];

    sumaPAC1 += VectorPotenciaAC1[i];

}


PotenciaActivaC1 = sumaPAC1/N;


for ( int i = 0; i <N; i++){

    VectorPotenciaAC2[i] = VectorTensionApura2[i] * VectorIntensidadApuraC2_2[i];

    sumaPAC2 += VectorPotenciaAC2[i];

}

PotenciaActivaC2 = sumaPAC2/N;


for ( int i = 0; i <N; i++){

    VectorPotenciaAC3[i] = VectorTensionApura2[i] * VectorIntensidadApuraC3_2[i];

    sumaPAC3 += VectorPotenciaAC3[i];

}

PotenciaActivaC3 = sumaPAC3/N;


for ( int i = 0; i <N; i++){

    VectorPotenciaAC4[i] = VectorTensionApura2[i] * VectorIntensidadApuraC4_2[i];

    sumaPAC4 += VectorPotenciaAC4[i];

}

PotenciaActivaC4 = sumaPAC4/N;


for ( int i = 0; i <N; i++){

    VectorPotenciaAC5[i] = VectorTensionApura2[i] * VectorIntensidadApuraC5_2[i];

    sumaPAC5 += VectorPotenciaAC5[i];
```

```
}
PotenciaActivaC5 = sumaPAC5/N;

PotenciaActivaT = PotenciaActivaC1 + PotenciaActivaC2 + PotenciaActivaC3 + PotenciaActivaC4 + PotenciaActivaC5;
PotenciaAparente = TensionEficazESC * IntensidadEficazT;
PotenciaReactiva = sqrt(pow(PotenciaAparente, 2) - pow(PotenciaActivaT,2));
fdp = PotenciaActivaT / PotenciaAparente;
C1 = TensionEficazESC * IntensidadEficazC1;
C2 = TensionEficazESC * IntensidadEficazC2;
C3 = TensionEficazESC * IntensidadEficazC3;
C4 = TensionEficazESC * IntensidadEficazC4;
C5 = TensionEficazESC * IntensidadEficazC5;
CT = C1 + C2 + C3 + C4 + C5;

//******************ENVIO DE TENSIÓN, CORRIENTES Y POTENCIAS*****************
  dtostrf(TensionEficaz, 10, 2, TensionEficazArray);
  client.publish(RAIZ_TOPIC_PUBLISH_TEF,TensionEficazArray);
  dtostrf(IntensidadEficazT, 10, 2, IntensidadEficazTArray);
  client.publish(RAIZ_TOPIC_PUBLISH_IEF_T,IntensidadEficazTArray);
  dtostrf(IntensidadEficazC1, 10, 2, IntensidadEficazC1Array);
  client.publish(RAIZ_TOPIC_PUBLISH_IEF_C1,IntensidadEficazC1Array);
  dtostrf(IntensidadEficazC2, 10, 2, IntensidadEficazC2Array);
  client.publish(RAIZ_TOPIC_PUBLISH_IEF_C2,IntensidadEficazC2Array);
  dtostrf(IntensidadEficazC3, 10, 2, IntensidadEficazC3Array);
  client.publish(RAIZ_TOPIC_PUBLISH_IEF_C3,IntensidadEficazC3Array);
  dtostrf(IntensidadEficazC4, 10, 2, IntensidadEficazC4Array);
  client.publish(RAIZ_TOPIC_PUBLISH_IEF_C4,IntensidadEficazC4Array);
  dtostrf(IntensidadEficazC5, 10, 2, IntensidadEficazC5Array);
  client.publish(RAIZ_TOPIC_PUBLISH_IEF_C5,IntensidadEficazC5Array);
  dtostrf(PotenciaActivaT, 10, 2, PotenciaActivaArray);
```

```
    client.publish(RAIZ_TOPIC_PUBLISH_PA,PotenciaActivaArray);

    dtostrf(PotenciaReactiva, 10, 2, PotenciaReactivaArray);

    client.publish(RAIZ_TOPIC_PUBLISH_PR,PotenciaReactivaArray);

    dtostrf(PotenciaAparente, 10, 2, PotenciaAparenteArray);

    client.publish(RAIZ_TOPIC_PUBLISH_PAP,PotenciaAparenteArray);

    dtostrf(fdp, 10, 2, FDPArray);

    client.publish(RAIZ_TOPIC_PUBLISH_FDP,FDPArray);



  //***************ENVÍO DE CONSUMOS CADA 15 MINUTOS**********************

  if((minutoStamp == "00:00" ll  minutoStamp == "15:00" ll minutoStamp == "30:00" ll minutoStamp == "45:00") &&
Enviado == false){

    Enviado = true;

    dtostrf(C1, 10, 2, C1Array);

    client.publish(RAIZ_TOPIC_PUBLISH_C1,C1Array);

    dtostrf(C2, 10, 2, C2Array);

    client.publish(RAIZ_TOPIC_PUBLISH_C2,C2Array);

    dtostrf(C3, 10, 2, C3Array);

    client.publish(RAIZ_TOPIC_PUBLISH_C3,C3Array);

    dtostrf(C4, 10, 2, C4Array);

    client.publish(RAIZ_TOPIC_PUBLISH_C4,C4Array);

    dtostrf(C5, 10, 2, C5Array);

    client.publish(RAIZ_TOPIC_PUBLISH_C5,C5Array);

    dtostrf(CT, 10, 2, CTArray);

    client.publish(RAIZ_TOPIC_PUBLISH_CT,CTArray);

  }

  Enviado = false;

  delay(1000);

  }

 client.loop();

}
```

```
//********** CONEXIÓN WIFI **************************
void setup_wifi(){
  delay(10);
  // Nos conectamos a red Wifi
  Serial.println();
  Serial.print("Conectando a ssid: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("Conectado a red WiFi!");
  Serial.print("Dirección IP: ");
  Serial.println(WiFi.localIP());
}
//********** CONEXIÓN MQTT **************************
void reconnect() {
  while (!client.connected()) {
    Serial.print("Intentando conexión Mqtt...");
    // Creamos un cliente ID
    String clientId = "CLIENTE_H_W_";
    clientId += String(random(0xffff), HEX);
    // Intentamos conectar
    if (client.connect(clientId.c_str(),MQTT_USER, MQTT_PASS)) {
      Serial.println("Conectado!");
      // Nos suscribimos (no utilizado)
      /*if(client.subscribe(RAIZ_TOPIC_SUBSCRIBE)){
        Serial.println("Suscripcion OK");
```

```
      } else {
        Serial.println("Error en la Suscripción");
      }*/
    } else {
      Serial.print("ERROR: ");
      Serial.println(client.state());
      Serial.println("Intentamos de nuevo en 5 segundos");
      delay(5000);
    }
  }
}
//********** CALLBACK ***************************
void callback(char* topic, byte* payload, unsigned int length){
  String incoming = "";
  Serial.print("Mensaje recibido desde: ");
  Serial.print(topic);
  Serial.println("");
  for (int i = 0; i < length; i++) {
    incoming += (char)payload[i];
  }
  incoming.trim();
  Serial.println("Mensaje: " + incoming);
}
```