Universidad Tecnológica Nacional
Facultad Regional Bahía Blanca

INGENIERÍA ELECTRÓNICA

PROYECTO FINAL

SMART ENERGY CONTROL

AUTORES

BINI, RODRIGO EMANUEL
CESARI, FEDERICO ANDRES

DIRECTORES

MG. ING. FRIEDRICH, GUILLERMO
ING. LAIUPPA, ADRIAN

Bahía Blanca | 22 de diciembre de 2022

AGRADECIMIENTOS

El desarrollo de este Proyecto Final no hubiera sido posible sólo con nuestro esfuerzo, sino que ha sido un pilar fundamental el apoyo de nuestras familias y amigos. Por tal motivo, hacemos extensivo el agradecimiento a todas aquellas personas que de alguna u otra forma han contribuido para que este Proyecto se haya hecho realidad.

Una mención especial para los Ingenieros Guillermo Friedrich y Adrián Laiuppa, quienes nos han acompañado durante su desarrollo, con permanentes contribuciones y siempre nos han brindado puntos de mejora.

Rodrigo Bini

Federico Cesari

Bahía Blanca, 2022

RESUMEN

Smart Energy Control integra un conjunto de dispositivos inteligentes, cuyo principal objetivo es realizar la medición del consumo de una red eléctrica monofásica hogareña o de una pequeña industria, en pos de contribuir, mediante diversas funcionalidades del sistema, a la eficiencia energética. Asimismo, todos estos dispositivos forman una red de objetos físicos conocida como Internet de las cosas (IoT), permitiendo al usuario realizar un control en tiempo real, tanto en la supervisión de los diversos parámetros eléctricos medidos y del ambiente, como así también realizar un manejo de interruptores de forma remota. Entre sus funciones, se destacan:

- Registro de usuarios en una base de datos, los que accederán mediante un nombre y contraseña, asignando a cada uno de ellos un ID único.
- Medición de diversos parámetros en tiempo real (eléctricos y del ambiente).
- Control de interruptores en forma remota.
- Acceso a un historial y gráficas de valores medidos, almacenados en un servidor remoto y gratuito.
- Envío de alertas a los usuarios registrados.
- Visualización de datos en forma local (a través de un display en los dispositivos) y de forma remota a través de cualquier dispositivo con conexión a internet (smartphone, tablet, notebook).
- Control del sistema mediante el uso de una aplicación móvil.

El sistema consta de un gabinete principal, y diversos sensores independientes, aunque conectados de forma inalámbrica entre sí, sustentados en diversos protocolos de comunicación y utilizando como principal la conexión WiFi.

Cada dispositivo puntual cuenta con un módulo ESP32, elegido fundamentalmente por su capacidad de procesamiento, y sabiendo que se encuentra orientado a la implementación de dispositivos IoT.

El desarrollo del software para la totalidad de los dispositivos, está basada en el funcionamiento de la multitarea, haciendo uso de la arquitectura doble núcleo del mencionado procesador.

El gabinete principal integra un tercer protocolo o comunicación, mediante el uso de los comandos AT para el envío de alertas a través de SMS. Para su implementación, se utilizó el módulo A6, que se conecta al ESP32 mediante el puerto serie.

El proyecto utiliza para su funcionamiento 2 (dos) servidores de datos. El principal es Firebase, y el fundamento de su uso se sustenta en la funcionalidad de realizar el control de usuarios registrados. Por otro lado, también se hará uso del servidor Thingspeak, que será implementado en todos los dispositivos con el objetivo de guardar un historial de los parámetros medidos, y que estos puedan ser representados mediante gráficas.

Se desarrolló una aplicación móvil para el control del sistema, utilizando los servidores mencionados como enlace entre la App y los dispositivos.

En la Figura 1 se realiza un diagrama esquemático del sistema.

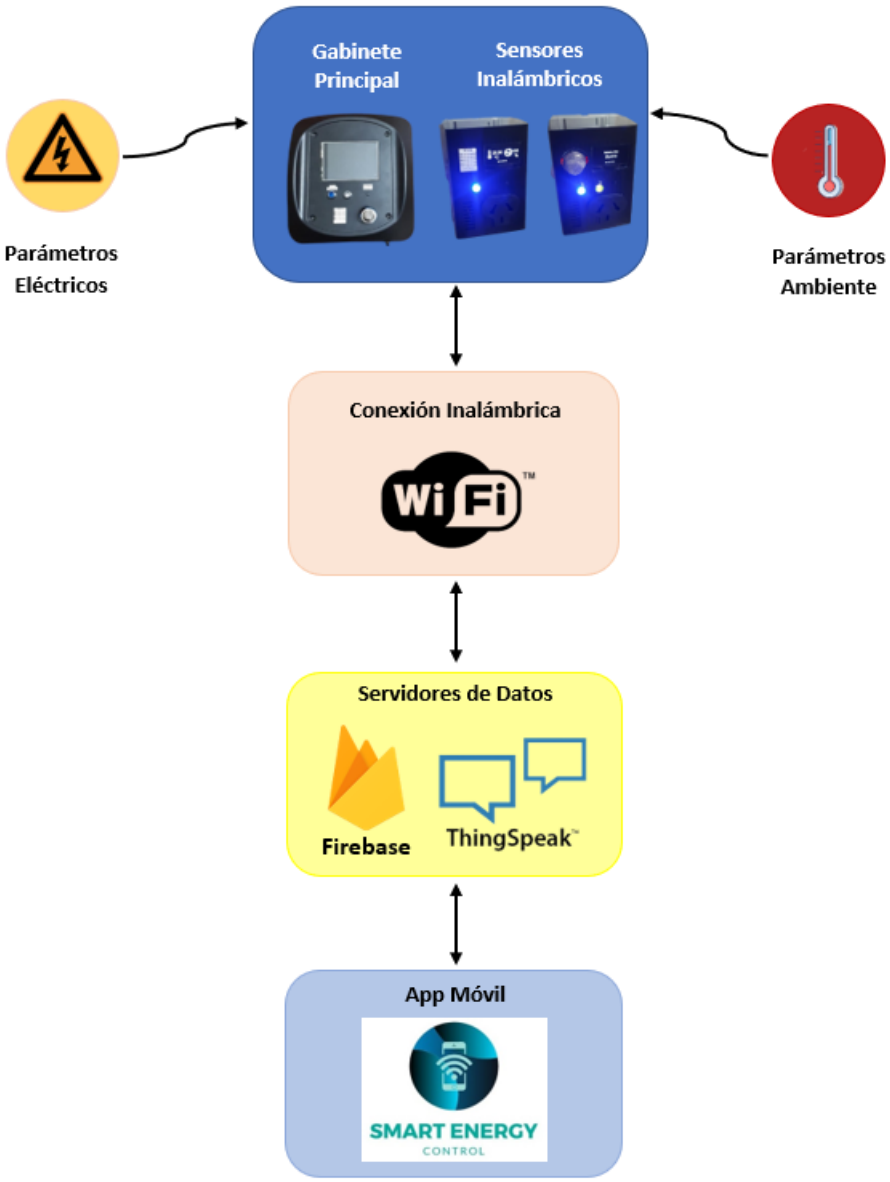

Figura 1. Diagrama Smart Energy Control

De esta manera, tanto el gabinete principal como los sensores realizan las mediciones de todos los parámetros del sistema, para luego, a través de una conexión WiFi, enviar los datos a los servidores Firebase y Thingspeak. Luego, mediante el uso de una Aplicación móvil, se accede a dichos datos y se efectúan otras interacciones como, por ejemplo, el control de interruptores o la recepción de alarmas.

PALABRAS CLAVE

Eficiencia energética

Consumo energético

IoT

Microcontrolador

Multitareas

Sensores

Conectividad WiFi

Servidor de datos

Aplicación móvil

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN Y DESCRIPCIÓN DEL PROBLEMA

1.1. INTRODUCCIÓN

El término Internet de las Cosas (Internet of Things – IoT) fue concebido en el Instituto de Tecnología de Massachusetts (MIT) en el año de 1999, estableciendo una revolución en la forma en que se relacionan personas y objetos, pasando de una relación elemental para convertirse en un intercambio inteligente de datos en tiempo real a través de la red. En los últimos años, se ha convertido en una de las tecnologías más importantes del siglo XXI.

Se refiere a un conjunto de dispositivos y sensores electrónicos interconectados entre sí que se encargan de medir, recopilar y enviar datos a un servidor centralizado o a la nube. Una vez que estos datos son tratados, y se procesó la información que se considera importante, los dispositivos IoT pueden recibir, del servidor o de la nube, una serie de instrucciones para realizar una determinada acción.

Su implementación es cada día más frecuente para entornos domésticos, mediante el flujo de información de electrodomésticos y dispositivos inteligentes, diseñados para facilitar la vida en nuestros hogares, con la capacidad añadida de intercambiar información a Internet.

Aprovechando el auge de esta tecnología, se ha diseñado un sistema basado en IoT, mediante la interacción de un conjunto de sensores y que, a través de una conexión inalámbrica, se suben valores a la nube para un control integral desde una aplicación móvil. El valor agregado del Proyecto es que integra el uso de IoT en el desarrollo de un sistema cuyo principal objetivo es, mediante el análisis de los datos de consumo energético, y las funcionalidades implementadas, contribuir a que el usuario pueda lograr una **eficiencia energética**.

1.2. OBJETIVOS

El principal objetivo del Proyecto es generar un sistema de monitoreo de variables eléctricas, con la finalidad de contribuir a realizar un uso eficiente de la energía. Tomando como punto de partida esta iniciativa, se pueden destacar los siguientes objetivos:

- Desarrollar un sistema inalámbrico integral de medición, donde interactúan un gabinete principal que recolecta las mediciones de todos los sensores conectados inalámbricamente a él.
- Construir un sistema de bajo costo, en comparación a otros dispositivos comerciales, fácilmente escalable, que permita realizar el agregado de nuevos sensores de una forma sencilla.
- Visualizar, en tiempo real, los parámetros medidos. Esto es posible a través del display de cada dispositivo, o desde una aplicación para dispositivos móviles.
- Utilizar los datos de consumos de energía, para contribuir a realizar un uso eficiente de la energía eléctrica. Asimismo, el sistema cuenta con diversas funciones implementadas que buscan maximizar la energía consumida, como el control de un interruptor a través de un sensor de temperatura.
- Integrar en el sistema diversos protocolos de comunicación, evitando de esta manera la pérdida de datos. Todos ellos serán almacenados en una tarjeta micro SD, a la vez que los datos son subidos a un servidor para mantener un historial de mediciones.

2. Desarrollo

2.1. Diseño e implementación de Gabinete Principal

El gabinete del proyecto es el dispositivo central, y su principal tarea es la de obtener los consumos energéticos totales del inmueble donde será instalado el sistema. Para ello, realiza la medición de los parámetros eléctricos sobre la línea monofásica de entrada al hogar. Asimismo, permite la conexión de dos circuitos individuales –establecidos por el usuario– para obtener mediciones parciales de consumo, a través de un sistema de medición no invasivo.

Su carcasa fue construida en plástico, mediante el proceso de termo-formado, y contiene un display táctil para la visualización de los datos e interacción con el usuario. Además, integra los sensores para medición de los parámetros del ambiente, como lo son el sensor de temperatura y humedad y el detector de humo/gases.

Sobre su base se encuentra montada la placa madre diseñada para el dispositivo, donde se conectan el microcontrolador, la placa de envío de mensajes de texto para alertas, y los diferentes sensores mencionados.

Figura 2. Gabinete Smart Energy Control

2.1.1. Diagrama de Bloques

El diagrama de la Figura 3 representa el funcionamiento del gabinete.

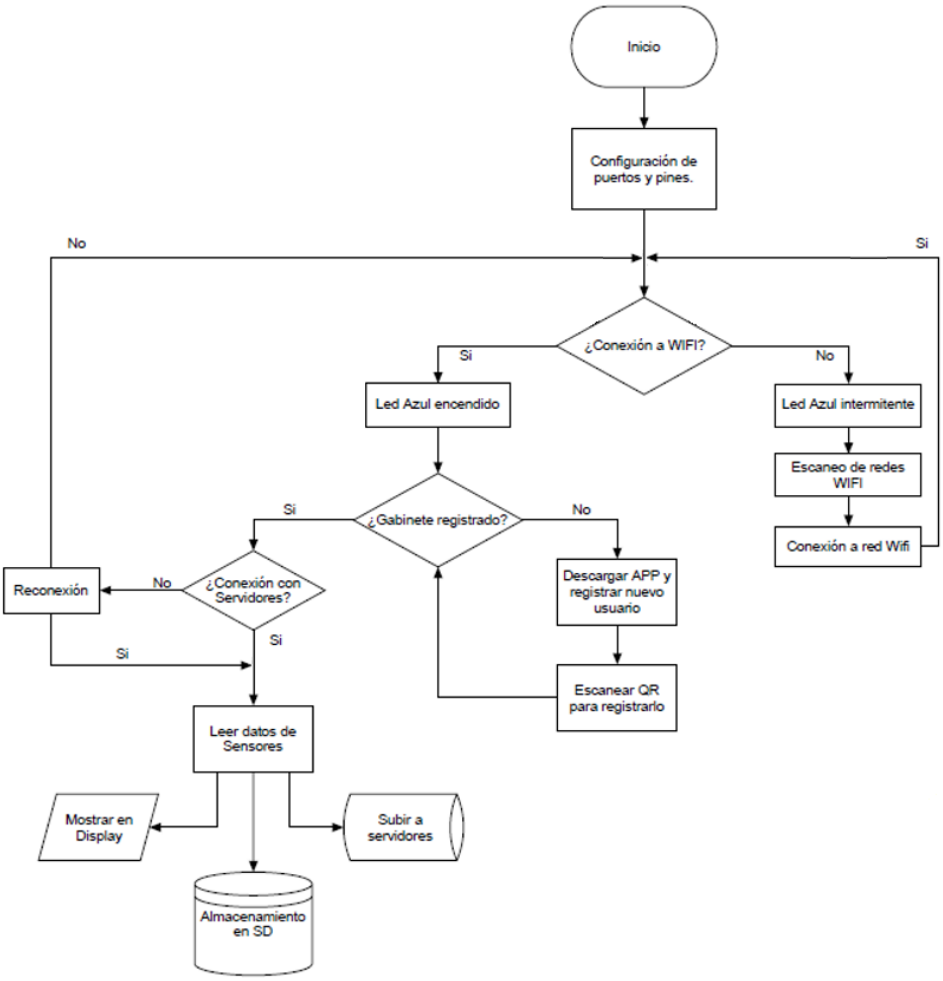

Figura 3. Diagrama de Bloques Gabinete

Al iniciar el dispositivo, tras realizar la configuración de puertos y pines, verificará el estado de la conexión WiFi. En caso de no poder conectarse –ya sea porque es el primer uso, o la red almacenada en el dispositivo no está disponible–, activará el menú para que el usuario realice la conexión a una red inalámbrica disponible.

Una vez conectado, se verificará si el dispositivo ya fue registrado por parte del usuario –en caso de no haberlo hecho, será requerido–, y posteriormente se inicializará la lectura de los sensores, junto a la conexión a los servidores de datos para el almacenamiento de los datos medidos.

2.1.2. HARDWARE UTILIZADO

2.1.2.1. MOTHERBOARD

Las placas de circuito impreso (del inglés: Printed Circuit Board, PCB), conforman la superficie constituida por caminos, pistas o buses de material conductor laminadas sobre una base no conductora, utilizada para conectar eléctricamente a través de las pistas conductoras.

A continuación, se observan el esquemático y el PCB de la placa realizada en un software de diseño electrónico (Figura 4 y Figura 5, respectivamente). En las Figuras 6 y 7 se puede observar la placa física realizada.

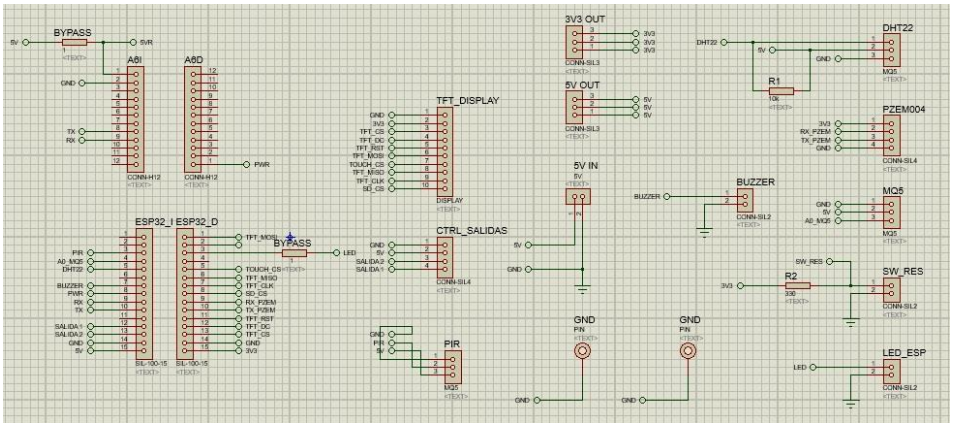

Figura 4. Esquemático Placa Base

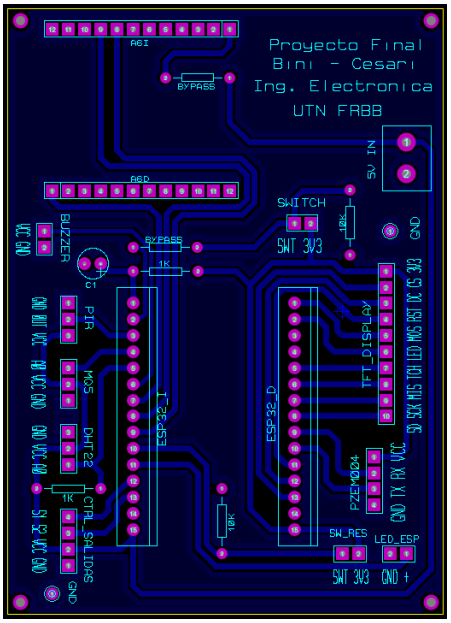

Figura 5. PCB Placa Base

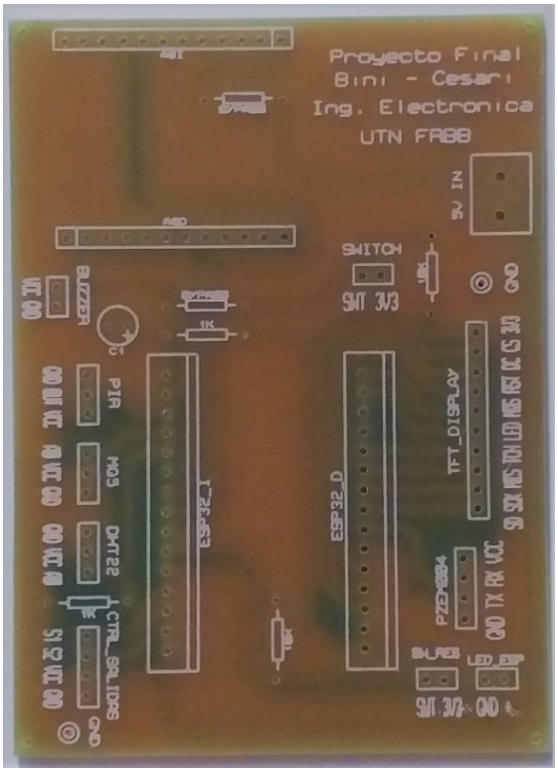

Figura 6. PCB Placa Base – Vista superior

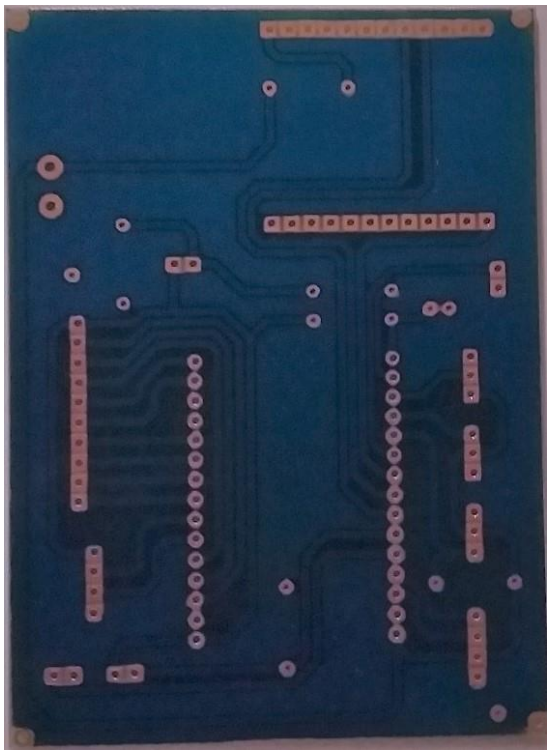

Figura 7. PCB Placa Base – Vista inferior

2.1.2.2. MICROCONTROLADOR ESP32

El microcontrolador implementado, tanto en el gabinete como en los sensores inalámbricos, es el ESP32 [1], de la compañía Espressif. El motivo de su elección no es sólo por la potencia del mismo y su bajo consumo, sino que se encuentra orientado al desarrollo de dispositivos para el "Internet de las cosas" o IoT. Además, brinda una integración con la conectividad WiFi, esencial para el desarrollo de nuestro prototipo. Otra de sus grandes ventajas es la arquitectura del procesador doble núcleo, aumentando su poder de procesamiento y brindando la posibilidad de desarrollar un sistema operativo multitarea, realizando una asignación estratégica de las diversas funciones en dichos núcleos. Ver más detalles en la sección 2.5.1.

Figura 8. Microcontrolador ESP32

2.1.2.3. PLACA PZEM-004T

A través de este módulo, se realiza la medición global de los parámetros eléctricos de la red monofásica: voltaje, corriente y potencia. Funciona con una bobina transformadora de corriente no invasiva.

Figura 9. Placa PZEM-004t con bobina transformadora

La parte principal del módulo PZEM-004T [2] es el chip SD3004 de SDIC Microelectronics Co., Ltd. La placa tiene una memoria EEPROM de 2K bits, borrable eléctricamente con un rango de voltaje de 4.5V a 5.5V, y dos optoacopladores PC817, que proporcionan aislamiento galvánico de la interfaz en serie.

La Tabla I contiene sus principales características.

	Voltaje [V]	Corriente [A]	Potencia activa [kW]	Energía [kWh]	Frecuencia [Hz]	Factor de potencia
Rango de medición	80 ~ 260	0 ~ 100	0 ~ 23	0 ~ 9999.99	45 ~ 65	0.00 ~ 1.00
Resolución	0.1	0.001	0.1	1	0.1	0.01
Precisión	0.5%	0.5%	0.5%	0.5%	0.5%	1%

Tabla I. Características Técnicas módulo PZEM-004T

COMUNICACIÓN SERIAL

El módulo está equipado con una interfaz de comunicación de datos en serie TTL a través del puerto serie que se puede leer y establecer los parámetros relevantes.

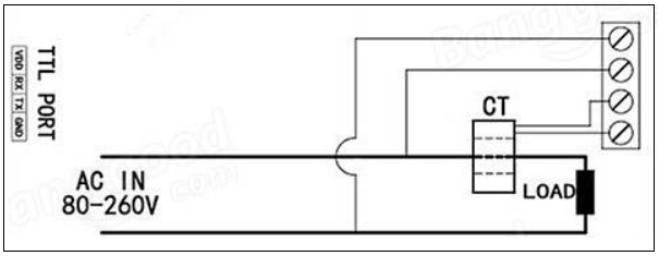

Figura 10. Diagrama de conexionado – PZEM-004t

2.1.2.4. SENSOR DE CORRIENTE SCT013

El sensor SCT013 [3] es un transformador de corriente no invasivo, que mide la intensidad de la corriente que circula por un conductor sin necesidad de cortar o modificar el mismo, ya que cuenta con un mecanismo de núcleo dividido, de forma análoga a una pinza amperométrica. Gracias a ello, permite su fácil instalación en un tablero eléctrico domiciliario.

Su funcionamiento consiste en realizar la medición de la intensidad que cruza un circuito obteniendo, por medio de electromagnetismo, una salida de corriente proporcional a la real consumida.

Figura 11. Sensor de corriente SCT013

La Tabla II contiene sus principales características.

Características sensor SCT013	
Medición	Corriente AC
Rango de medición	0 A a 100 A
Relación de transformación	2000:1 (100 A : 50 mA)
Diámetro interior de la pinza	1.3 cm aprox
Temperatura de operación	-25 ºC a +70 ºC

Tabla II. Características Técnicas sensor SCT013

Incorpora un diodo TVS de protección contra transitorios, incluyendo la desconexión súbita cuando el transformador está energizado.

La conexión se realiza con un conector de audio estéreo estándar de 3.5 mm (Plug). La conexión central o anillo (Ring) no está conectada.

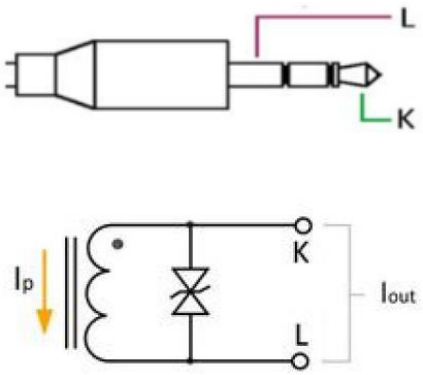

Figura 12. Sensor de Corriente SCT013 - Esquema de Conexión

Ip es la intensidad del primario (la intensidad a medir), *Iout* la intensidad en el secundario (la suministrada por el transformador), ambas en alterna.

ADAPTACIÓN DE LA SEÑAL

La primera consideración al trabajar con redes alternas monofásicas, es el hecho de que se debe realizar una adaptación o acondicionamiento de dichas señales para que puedan ser procesadas correctamente por el microcontrolador. Para ello, la relación de transformación del sensor, junto a un circuito auxiliar, permiten la adaptación para que el microcontrolador pueda procesar la señal, mediante el uso de una entrada ADC; es decir, la utilización de un conversor analógico a digital.

Un conversor A/D es un chip que proporciona una representación digital de una señal de entrada analógica en un instante de tiempo. El conversor A/D realiza "muestras" periódicas de la señal a una frecuencia determinada. Posteriormente, estas muestras son reconstruidas para obtener una señal equivalente a la original.

El ESP32 puede valerse de su compatibilidad con Arduino para usar con facilidad convertidores externos con librerías incluidas en el entorno Arduino IDE. Además, su conversor es de 12 bits, lo que quiere decir que la entrada puede tomar hasta 4095 valores diferentes, aumentando de esta manera la precisión de las lecturas con respecto a un conversor de 10 bits.

Por otro lado, como la tensión y la corriente de la red eléctrica es alterna, alternará entre valores negativos y positivos, mientras que el microcontrolador sólo trabaja con valores positivos de entrada entre 0 V y 3,3 V. Por lo tanto, hay que reducir, filtrar y añadir un offset a las señales de los circuitos de sensado para que entreguen señales dentro de esos márgenes. La placa permite obtener suficientes muestras de las señales

analógicas para su posterior reconstrucción, ya que la frecuencia del cristal que incorpora es de 40 MHz.

CALIBRACIÓN

El sensor utilizado mide una corriente alterna máxima de 100A. Este valor de 100A es la corriente RMS, que también se denomina corriente eficaz. La corriente RMS es igual al valor máximo que puede alcanzar la corriente (pico de corriente) dividido por la raíz cuadrada de dos.

Por lo tanto la corriente pico máxima medida es:

$$I\,(\text{medido}) = \sqrt{2} \times \text{I}\,(\text{RMS}) = 1{,}414 \times 100\text{A} = 141{,}4\,\text{A}$$

Sabiendo que para una corriente de 100A en el primario, produce 50mA en el secundario, solo usa la fórmula de la relación de transformación. El resultado será:

$$\frac{N_1}{N_2} = \frac{I_2}{I_1}$$

- I_1 = corriente primaria (corriente a medir);
- I_2 = corriente secundaria;
- N1 = número de vueltas en el primario (en el caso de este sensor, N1 será igual a 1);
- N_2 = número de vueltas del secundario, que el SCT 013 tiene 2000 vueltas.

La corriente en la salida del sensor es inversamente proporcional al número de vueltas (2000):

$$I\,(\text{sensor}) = \frac{I(\text{medido})}{\text{número de vueltas}} = \frac{141.4\text{A}}{2000} = 0.0707\text{A}$$

Como este sensor entrega una corriente, se debe calcular una resistencia de carga, para poder entregar una tensión al ADC del ESP32.

Como estos sensores de corriente se van a dar un uso hogareño, no se medirán valores mayores a 10A, por lo que se considerará una corriente máxima 10 veces menor a la que puede medir el sensor:

$$I\,(\text{sensor}) = \frac{I(\text{medido})}{\text{número de vueltas}} = \frac{14.14\text{A}}{2000} = 0.00707\text{A}$$

Para maximizar la resolución de la medición, el voltaje a través de la resistencia de carga en el pico de corriente debe ser igual a la mitad del voltaje de referencia analógico del microcontrolador. (A_{REF} / 2)

$$R\ (carga) = \frac{V(sensor)}{I(sensor)} = \frac{\frac{3.3V}{2}}{0.00707\text{A}} = 233\Omega$$

Por lo tanto, se selecciona un valor comercial de **220 Ω**.

Luego, para convertir el voltaje alterno en continuo, se emplea un divisor resistivo con dos resistencias iguales de 10KΩ. Esto permite que la entrada al pin del microcontrolador varíe de 0V a 3,3V, estrictamente requerido dado que el voltaje alterno variará en un rango comprendido por [-1,65V, +1,65V], con posibilidad de dañar el microcontrolador ya que no puede recibir tensiones negativas.

En la Figura 13 se visualiza el circuito de adaptación.

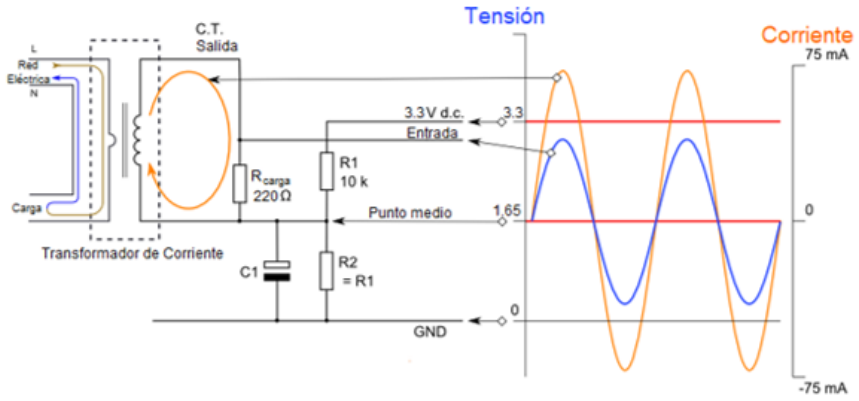

Figura 13. Adaptación del Sensor de Corriente SCT013 al ESP32

2.1.2.5. DISPLAY ILI9341

Se utilizó una pantalla táctil TFT de 3.2'' TJCTM24024 [4], de 240x320 pixeles de resolución. El módulo tiene integrado un controlador ILI9341 para manejar el LCD y un controlador táctil XPT2046 para el control táctil. El módulo dispone también de una ranura de conexión para una tarjeta SD. Tanto el LCD, como el táctil y la SD se controlan por SPI, lo que brinda una visualización mucho más rápida. Ver estándar SPI en sección 4.1.

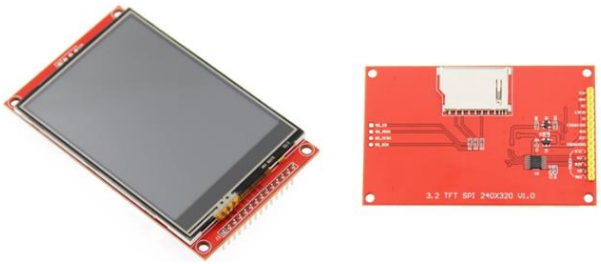

Figura 14. Display ILI9341

2.1.2.6. FUENTE AC-DC Y BATERÍA

La fuente de alimentación y los módulos de batería, forman un sistema completo y continuo de energía hacia el microcontrolador.

Durante el funcionamiento normal, la tensión de entrada procede desde una fuente 220VAC/5VDC, y la misma cumple dos funciones: alimentar la placa principal, y en forma simultánea, la carga de las baterías. En caso de fallo de la red eléctrica (caída de la tensión de entrada) el sistema pasa inmediatamente a alimentarse con una de las baterías. En el momento en que se restablece el suministro eléctrico, el sistema vuelve al funcionamiento normal y el cargador integrado nuevamente comienza a recargar las baterías.

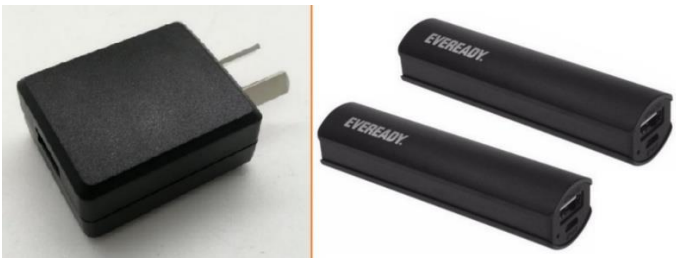

Figura 15. Fuente AC/DC y Batería

La fuente utilizada es de 5V 2A, mientras que como batería se utilizan 2 (dos) Power Bank EVEREADY, DC 5V 1A, y baterías de Li-ion, con una capacidad de 2200 mAh. Una de las baterías alimenta la placa principal, y la restante se utiliza para la alimentación del módulo A6 para envío de alertas.

En casos de corte de energía, se considerarán las siguientes acciones del dispositivo, con el objetivo de optimizar la batería y extender su autonomía:

- El display se apagará. En caso de ser necesaria su visualización, con un toque sobre la pantalla, el mismo se encenderá nuevamente.
- El módulo A6 será encendido únicamente en los casos en que sea necesario el envío de una alerta, y luego volverá a su estado de reposo/apagado.

2.1.2.7. SENSORES MEDICIÓN DE PARÁMETROS DEL AMBIENTE

Para realizar la medición de los parámetros del ambiente, se utilizarán los sensores DHT22 (temperatura y humedad) y MQ-2 (detección de humo/gases), los que se corresponden con los implementados en los sensores inalámbricos. Por tal motivo, serán desarrollados en las secciones 2.2.3.1. (página 30) y 2.2.3.2. (página 32), respectivamente.

2.1.3. ENTORNO GRÁFICO LVGL

El display ILI 9341 -ver sección 2.2.2.4.- implementado en el gabinete principal tiene una interfaz gráfica desarrollada mediante la librería de gráficos integrados LVGL [5]. Esta librería es gratuita y de código abierto. LVGL está escrito en C (compatible con C++) y tiene enlaces a MicroPython, PikaScript, JavaScript (similar a React) y Berry.

El diseño se realizó a través del software SquareLine Studio [6], en su versión 1.1.1., un editor de interfaz de usuario que permite arrastrar y soltar widgets para simplificar el desarrollo. Una de sus principales características es la compatibilidad con cualquier microcontrolador, procesador y sistema operativo, con el objetivo de controlar pantallas o monitores OLED, ePaper y TFT.

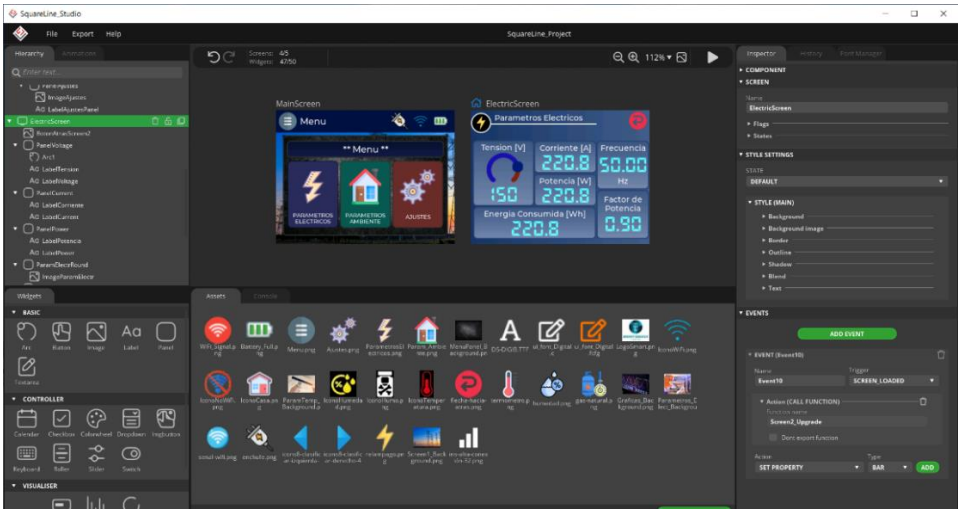

Figura 16. Software SquareLine Studio

El software tiene las siguientes características:

- Potentes bloques de construcción: botones, gráficos, listas, controles deslizantes, imágenes, etc.
- Efectos gráficos avanzados: animaciones, anti-aliasing, opacidad, desplazamiento suave, etc.
- Escrito en lenguaje C para una máxima compatibilidad (compatible con C ++).
- Simulador: admite el diseño de GUI integrado en PC sin hardware integrado.

LVGL utiliza lv_tasks que son funciones llamadas periódicamente en lv_task_handler. En la lv_task se puede obtener el estado de los sensores, búferes, etc. y llamar a las funciones de LVGL para actualizar la GUI.

2.1.3.1. PANTALLA PRINCIPAL

Una vez iniciado el Gabinete, se accederá a la Pantalla Principal, tal como se indica en la Figura 17.

Figura 17. Pantalla Principal Gabinete

La pantalla posee una barra de estado, donde se podrán observar los siguientes datos:

- **Estado de la batería.** El ícono indicará el nivel de carga de la batería.
- **Estado de la conexión WiFi.** El ícono indicará que la conexión a la red está establecida y funciona correctamente. De lo contrario, se dará la indicación de conexión interrumpida.
- **Estado de la red eléctrica.** El ícono indicará que la red eléctrica funciona correctamente. De lo contrario, el ícono mostrará el estado de corte de energía.
- **Acceso a Menú Principal.** A través del botón se podrá acceder al panel del menú principal, descripto en la sección 2.2.3.2.

En el centro de la pantalla, mediante un widget circular, se indicarán la fecha y hora actuales.

2.1.3.2. MENÚ PRINCIPAL

Como se indicó en la sección 2.2.3.1., a través del botón  se accede al menú principal, indicado en la Figura 18.

Figura 18. Pantalla Menú Principal Gabinete

Dentro del menú, se disponen 3 opciones:

- **Parámetros eléctricos.** Se accederá a la medición en tiempo real de los parámetros eléctricos medidos, a saber: Tensión, Corriente, Potencia, Energía consumida, Frecuencia y Factor de Potencia.
- **Parámetros Ambiente.** Se accederá a la medición en tiempo real de las variables controladas en el ambiente, a saber: Temperatura, Humedad y Presencia de Humo/Gases.
- **Ajustes.** Se accede a un nuevo menú que permite la configuración de tres tópicos: la conexión WiFi, el manejo de la tarjeta microSD, y la visualización de la pantalla con los créditos de realización del Proyecto.

2.2. Diseño e Implementación de Sensores inalámbricos

El funcionamiento del Gabinete se complementa con diversos sensores inalámbricos. Estos controlarán distintas variables, como es el caso de temperatura y humedad, o presencia de gas/humo, y permiten ser distribuidos por los distintos ambientes del inmueble sin necesidad de realizar cableados; únicamente se requiere de un tomacorriente 220 VAC. Asimismo, cada uno de ellos dispone de un tomacorriente controlado de forma remota mediante una aplicación web –ver sección 6.3. –.

La información obtenida de cada sensor puede visualizarse a través de una pantalla OLED 1.3" ubicada en el mismo sensor, o de forma remota y en tiempo real a través de la mencionada Aplicación.

El diagrama de la Figura 19 representa el funcionamiento de los sensores.

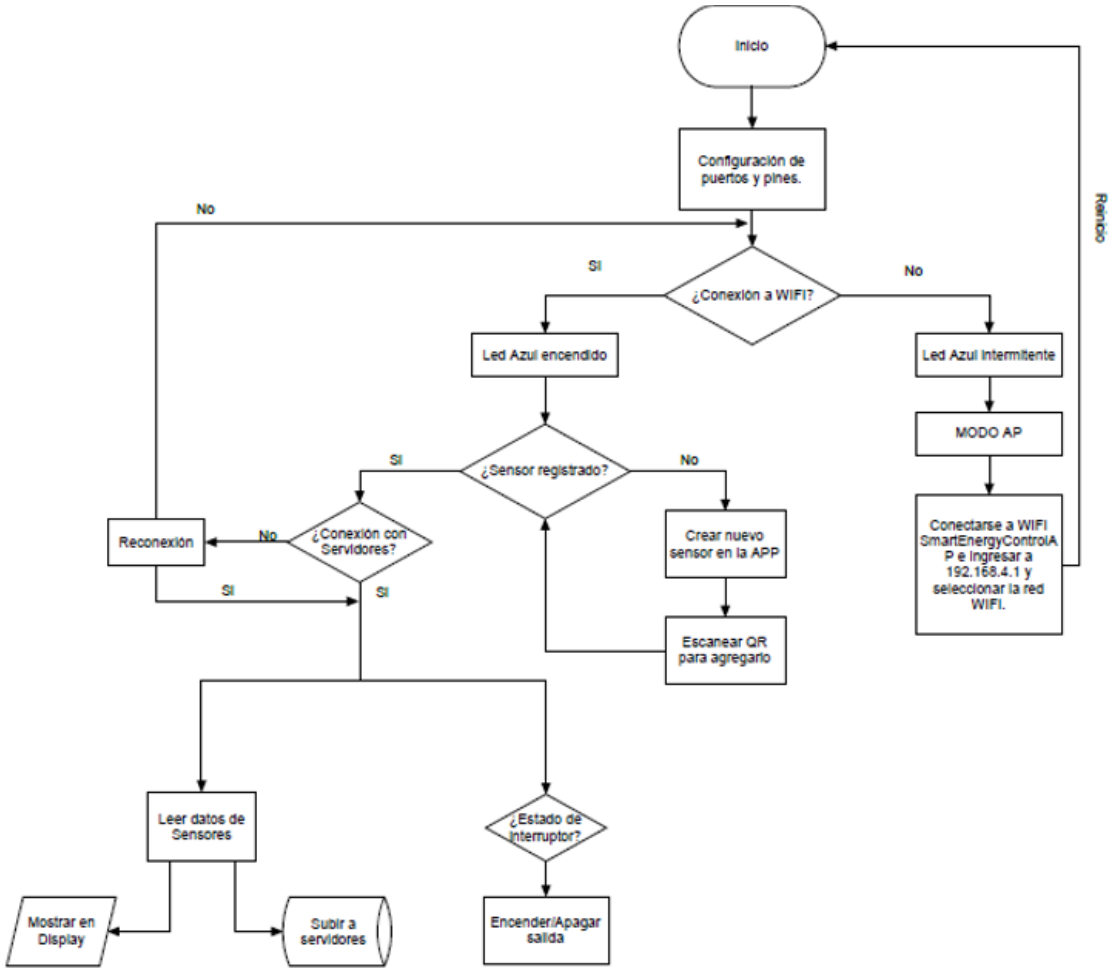

Figura 19. Diagrama de Bloques Sensores

2.2.1. SENSOR DE TEMPERATURA Y HUMEDAD

Mediante la implementación del sensor DHT22/AM2302, se realizarán mediciones de temperatura y humedad.

Estos valores pueden observarse de manera local a través de la pantalla OLED integrada, o de forma remota. En la pantalla, además, se pueden visualizar los datos del sensor y datos de la red de conexión a internet. Mediante el pulsador, se podrán alternar las distintas visualizaciones de datos.

El led azul es el indicador de la conexión a internet, mediante 2 (dos) configuraciones:

- Si el led parpadea, indica que el dispositivo no está conectado a una red WiFi.
- Si, por el contrario, se mantiene encendido de manera fija, implica que el dispositivo se encuentra conectado a una red inalámbrica de internet.

1.

Figura 20. Sensor de Temperatura y Humedad

El interruptor, controlado de forma remota, mide la corriente y potencia consumidas, con la implementación del sensor de Efecto Hall Acs712 y su circuito de control.

2.2.2. SENSOR DE HUMO

Mediante la implementación del sensor MQ-2, se realizarán mediciones para establecer la calidad del aire.

Los valores pueden observarse de manera local a través de la pantalla OLED integrada, o de forma remota (mediante la App). En la pantalla, además, se pueden visualizar los datos del sensor y datos de la red de conexión a internet. Mediante el pulsador, se podrán alternar las distintas visualizaciones de datos.

El led azul es el indicador de la conexión a internet, mediante 2 (dos) configuraciones: si el led parpadea, indica que el dispositivo no está conectado a una red WiFi. Si, por el contrario, se mantiene encendido de manera fija, implica que el dispositivo se encuentra conectado a una red inalámbrica de internet.

El led verde/rojo es un indicador visual rápido del estado del aire sensado. El led verde encendido es un indicador de un aire sin presencia de humo o gases, mientras que sucederá lo contrario si el led rojo se enciende, junto a la alarma sonora.

Figura 21. Sensor de gases/humo

En el caso particular de este sensor, el interruptor puede ser controlado de forma automática estableciendo una temperatura de encendido, y otra de apagado. De esta forma, el interruptor se encenderá únicamente en el rango de valores establecido, con el objetivo de optimizar el consumo energético.

2.2.3. HARDWARE UTILIZADO

2.2.3.1. SENSOR DE TEMPERATURA Y HUMEDAD DHT 22/AM 2302

A través de este sensor DHT 22 [7], de tipo capacitivo, se realiza la medición de la temperatura y humedad del ambiente, tanto para el Gabinete principal como en el caso del sensor inalámbrico de temperatura y humedad.

La carcasa tiene dos partes, y en su interior se encuentran, por un lado, un componente de detección de humedad, compuesto de dos electrodos con sustrato que retiene la humedad (generalmente una sal o un polímero plástico conductor) intercalado entre ellos. El componente restante es un sensor de temperatura NTC (Coeficiente de temperatura negativo), lo que significa que la resistencia disminuye con el aumento de la temperatura.

Posee una alta precisión en sus mediciones, y una excelente estabilidad a largo plazo, combinadas con un bajo consumo de energía. La resolución de medición de temperatura es de 0.1 °C y de 0.1% RH para humedad.

Figura 22. Componentes internos sensor DHT22/AM2302

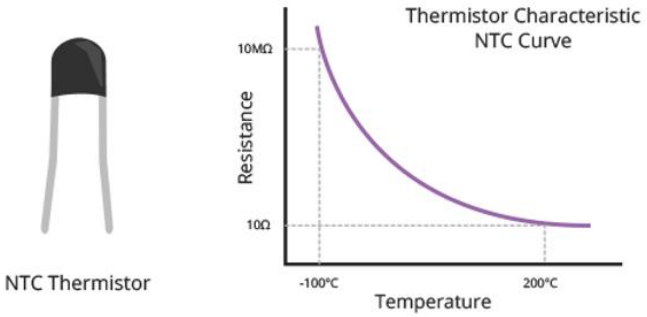

Figura 23. Termistor NTC con curva característica

Características sensor DHT22			
Voltaje de Funcionamiento	3.3V a 5.5V	**Rango de medición temperatura**	-40 °C a 80 °C
Consumo de corriente máx	2.5 mA	**Precisión de medición temperatura**	<±0.5 °C
Velocidad de muestreo	0.5 Hz	**Rango de medición humedad**	0% a 100% RH *
Resolución	16 bits	**Precisión de medición de humedad**	2% RH
* 0.5°C de variación			

Tabla III. Características Técnicas sensor DHT22

CONEXIÓN CON ESP32

El sensor DHT22 tiene 4 pines, a saber: tierra o GND, señal o datos, null y alimentación VCC que va desde 3.3 V hasta 5V. La conexión a la placa ESP32 se realizar de la siguiente manera:

- Pin VCC a la salida de 3.3v del ESP32
- Pin GND a GND del ESP32
- Pin datos "OUT" a un pin digital del ESP32 (En este caso, pin D4)

Nota: requiere una resistencia de 4.7kΩ o 10kΩ en modo Pull-up, entre el pin de Datos y VCC.

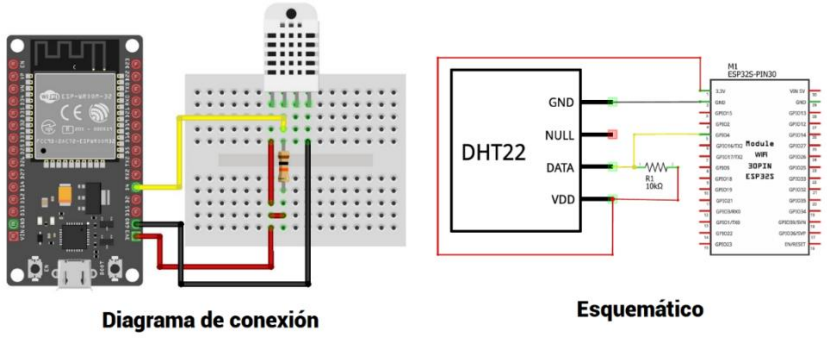

Figura 24. Conexión entre DHT22 y el módulo ESP32

2.2.3.2. SENSOR DE GAS MQ-2

El sensor MQ-2 [8] es un sensor de gas inflamable y humo que detecta las concentraciones de gas combustible en el aire y emite su lectura como un voltaje analógico.

Puede detectar o medir gases como **GLP, alcohol, propano, hidrógeno, humo, CO e incluso metano**, por lo que puede utilizarse para una variedad de aplicaciones. Es de tipo semiconductor de óxido de metal (MOS), ya que la detección se basa en el cambio de resistencia del material de detección cuando el gas entra en contacto con el mismo.

Este módulo tiene dos posibilidades de salida: una salida analógica (A0) y una salida digital (D0). Smart Energy Control utiliza la salida analógica para medir el volumen de fugas de gas, y generar una alarma cuando la concentración de gas es mayor a un umbral prestablecido por el usuario.

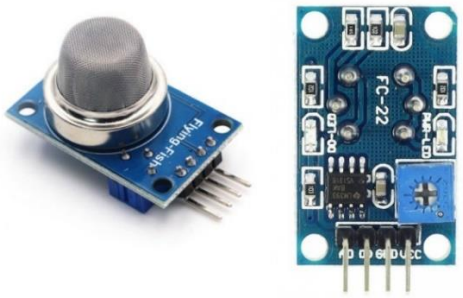

Figura 25. Sensor de gas MQ-2

Características MQ-2	
Voltaje de Operación	5V DC
Rango de detección	300 - 10000 ppm
Tiempo de Respuesta	≤ 10 s
Tiempo de recuperación	≤ 30 s
Temperatura de trabajo	-20 °C ~ 55°C
Alta sensibilidad	Gas LP, Gas Natural, Gas de Carbón
Baja sensibilidad	Alcohol, Humo

Tabla IV. Características Técnicas sensor MQ-2

En el gráfico de la Figura 26 se puede obtener la concentración del gas a partir de la relación entre la resistencia del sensor R0 y la resistencia medida Rs. También es necesario conocer la resistencia Rl empleada en el módulo.

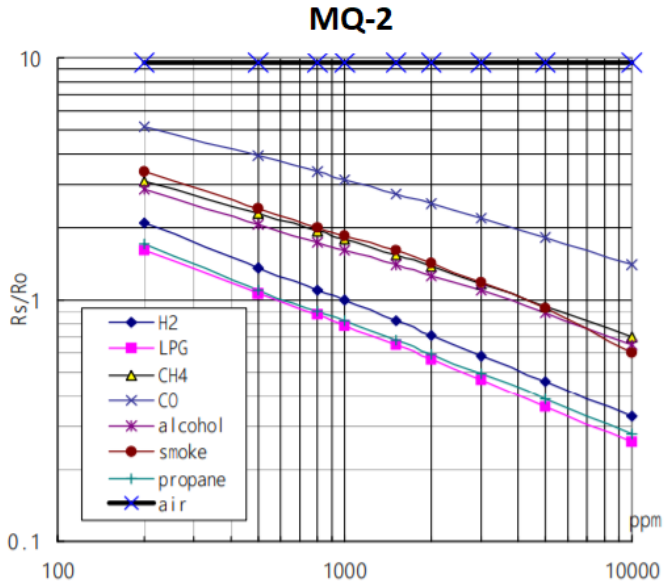

Figura 26. Características de sensibilidad Sensor MQ-2

La gráfica se dispone en escala logarítmica para ambos ejes y, en general, son aproximadamente rectas bajo estas escalas. La concentración será:

$$Concentración = 10^{(A.\log\left(\frac{R_S}{R}\right)+B)}$$

Para ello, se requiere la recta que la aproxima, a través de la selección de dos puntos de las gráficas P0 = {X0, Y0} y P1 = {X1, Y1}, resultando la ecuación de la recta:

$$Y = Ax + B$$

Siendo

$$A = \frac{Y_1 - Y_0}{X_1 - X_0}$$

$$B = Y_0 - AX_0$$

Por ejemplo, a una temperatura de 20 ℃ y Humedad del 65%, la concentración de O2 es de 21%, RL = 20kΩ, Ro: resistencia del sensor a

1000ppm de H2 en el aire limpio, Rs: resistencia del sensor a varias concentraciones de gases.

2.2.3.3. Sensor de Corriente ACS712-30A

El sensor el ACS712 [9] mide corrientes en alterna o continua, mediante el empleo de un sensor de efecto Hall que detecta el campo magnético que se produce por inducción de la corriente que circula en la línea medida.

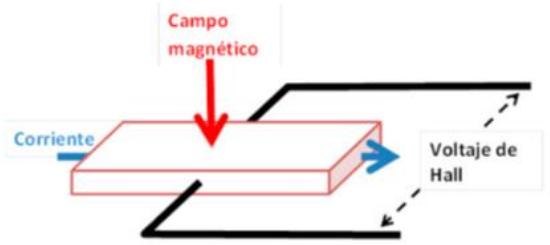

Figura 27. Esquema de efecto Hall.

El sensor devuelve una señal de tensión proporcional a la corriente, 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. Para el desarrollo del Proyecto, se ha seleccionado el modelo de 30A que cuenta con 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.

Para su conexión, se utilizan tres pines: dos corresponden a la alimentación, y el tercero a la salida analógica que irá conectada a un ADC del ESP32. En la bornera se conecta la línea a medir.

Figura 28. Sensor de corriente ACS712-30A

El ACS712-30A devuelve un valor de 2.5V para una corriente de 0A, y a mayor corriente a medir se incrementará proporcionalmente la tensión de salida de acuerdo con la sensibilidad, siendo el máximo posible a medir 30A y entregará 5V. Estos valores de tensiones que entrega la salida analógica no están dentro del rango de tensiones de los pines del ESP32 (0V-3.3V), por lo que se aplicara un divisor resistivo para adaptarlo a la entrada del ADC del ESP32.

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in} = \frac{2K\Omega}{1K\Omega + 2K\Omega} 5V = 3.3V$$

A su vez ACS712-30A posee una relación lineal entre la salida de tensión del sensor y la corriente, que al representarse en una gráfica de voltaje vs corriente es una línea recta, donde la pendiente es la sensibilidad(m), e intersecta con el eje Y en los 2,5V.

$$V = m\, I + 2.5$$

Para hallar la corriente del sensor se despeja la fórmula:

$$I = \frac{V - 2.5}{m}$$

Despejando nuevamente esta fórmula se calibra el sensor para minimizar el error en las mediciones, a partir de tomar dos mediciones se obtiene una nueva constante de sensibilidad del sensor:

$$sensibilidad = m = \frac{V_2 - V_1}{I_2 - I_1} = \frac{1.543 - 1.5}{0.33 - 0} = 0.1303\, V/A$$

Esta sensibilidad se utilizará en el código del microcontrolador:

```
float get_corriente(){
  float voltajeSensor, corriente=0, Imax=0, Imin=0;
  float Sensibilidad=0.1303, offset=0.27;
  long tiempo=millis();
  while(millis()-tiempo<500){
    voltajeSensor = analogRead(ACS_712) * (3.3 / 4095);
    corriente=0.9*corriente+0.1*((voltajeSensor-1.543)/Sensibilidad);
    if(corriente>Imax)Imax=corriente;
    if(corriente<Imin)Imin=corriente;
  }
  return(((Imax-Imin)/2)-offset);
}
```

2.3. ARDUINO IDE

Para la programación de los microcontroladores, se utilizó el entorno de desarrollo integrado (IDE) Arduino [10], una herramienta de código abierto. Esta plataforma está escrita en lenguaje JavaScript, y puede implementarse en los sistemas operativos Windows, Mac y Linux.

Para inicializar la programación, se debe configurar en primera instancia la placa que se va a utilizar. Una vez hecho esto, se deberán descargar las librerías que serán necesarias para la ejecución del programa.

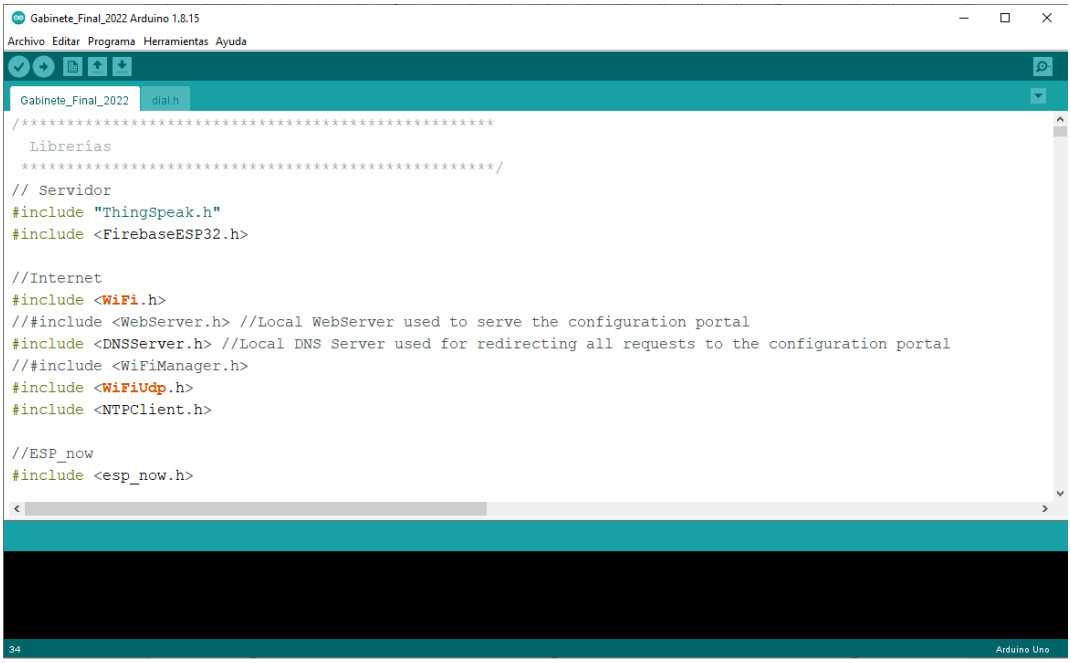

Figura 29. Entorno de desarrollo Arduino IDE

2.4. GESTIÓN DE MEMORIA Y DE PROCESOS EN SISTEMAS EMBEBIDOS

2.4.1. ESP32 – ARQUITECTURA

Como se indicó en la sección 2.1.2.2., el microcontrolador elegido para el Proyecto es el ESP32. El motivo de su elección no es sólo por la potencia del mismo y su bajo consumo, sino que se encuentra orientado al desarrollo de dispositivos para el "Internet de las cosas" o IoT. Además, brinda una integración con la conectividad WiFi, esencial para el desarrollo de nuestro prototipo.

Una de las características fundamentales y que destacan a esta placa, es su arquitectura doble núcleo, función implementada en la realización de nuestro Proyecto. Ver Figura 30.

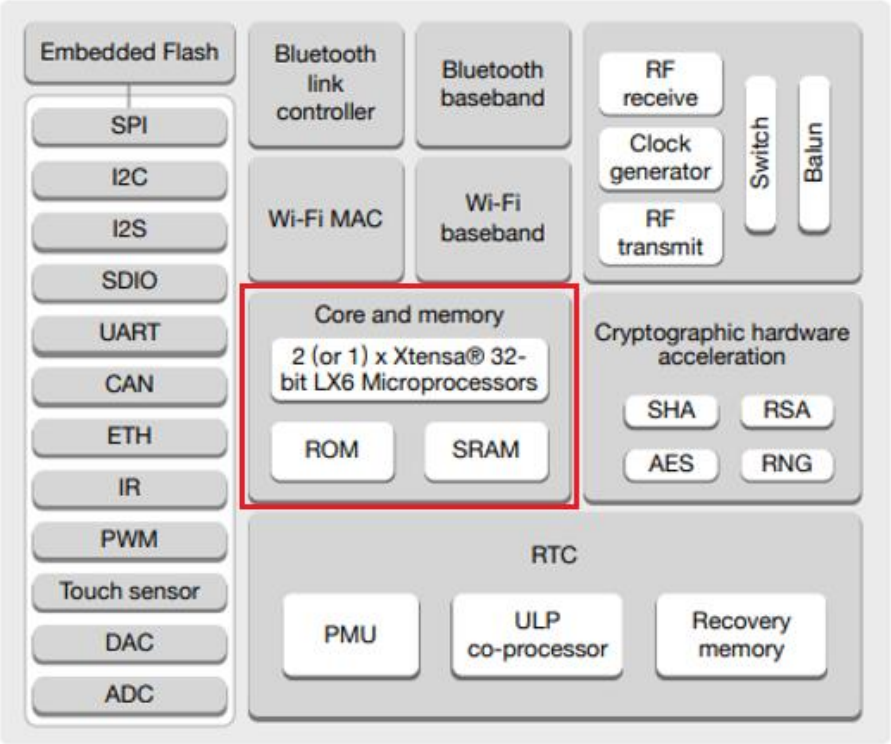

Figura 30. Arquitectura ESP32

El procesador Xtensa Dual-Core LX6 de 32 bits a 160 ó 240 MHz, y coprocesador de ultra baja potencia (ULP), fueron utilizados para implementar el proyecto mediante la multitarea, realizando una asignación de las diversas funciones a ambos núcleos, de una manera estratégica. En próximas secciones, se detallarán estas asignaciones e implementación.

A continuación, se seleccionarán y desarrollarán aquellas características utilizadas para la implementación en el proyecto.

Conectividad WiFi

El chip cuenta con conectividad WiFi, siendo compatible con 802.11 b/g/n en la banda de los 2.4GHz, alcanzando velocidades de hasta 150 Mbits/s.

Memoria

Hay diversos tipos de memoria, a saber:

- **Memoria de programa.** Permite almacenar el sketch.
- **Memoria SRAM.** Permite almacenar las variables que se utilizan en el código.
- **Memoria EEPROM.** Es una memoria no volátil, por lo que su función es la de almacenar variables que no pierdan su valor aun cuando el dispositivo está apagado.

El proyecto implementado utiliza la memoria EEPROM para almacenar los datos de la red WiFi (nombre de la red y su contraseña), y datos del usuario registrado (número telefónico y dirección de correo electrónico) para el envío de las alertas.

Para la escritura/lectura de datos, se utilizaron las siguientes funciones:

```
void grabar(int addr, String a) {
  int tamano = a.length();
  char inchar[50];
  a.toCharArray(inchar, tamano+1);
  for (int i = 0; i < tamano; i++) {
    EEPROM.write(addr+i, inchar[i]);
  }
  for (int i = tamano; i < 50; i++) {
    EEPROM.write(addr+i, 255);
  }
  EEPROM.commit();
}

String leer(int addr) {
   byte lectura;
   String strlectura;
   for (int i = addr; i < addr+50; i++) {
      lectura = EEPROM.read(i);
      if (lectura != 255) {
        strlectura += (char)lectura;
      }
   }
   return strlectura;
}
```

Para poder realizar lecturas o escrituras en la memoria, en primer lugar, debe inicializarse:

```
void setup() {
  EEPROM.begin(512);
  leermemoria();
}
```

Como puede observarse en la declaración, utilizando la librería EEPROM, es posible utilizar hasta 512 bytes en la memoria flash. Esto significa que tiene 512 direcciones diferentes y puede guardar un valor entre 0 y 255 en cada posición de dirección.

Luego se llama a la función *leermemoria()* que guardará los datos almacenados en la memoria, en las variables utilizadas por el programa, de la siguiente manera:

```
void leermemoria(){
  ssidName = leer(0);
  Serial.println(ssidName);
  password = leer(50);
  Serial.println(password);
  numero_A6 = leer(100);
  Serial.println(numero_A6);
  e_mail = leer(150);
  Serial.println(e_mail);
}
```

Interfaces Periféricas

Entre las interfaces disponibles en ESP32, podemos destacar I2C, SPI, PWM, CAN, entre otros. Para el conexionado de pantallas y sensores, se han utilizado I2C y SPI.

- ✓ **I2C para conexión de display OLED 1.3”**

I2C significa Inter Integrated Circuit, y es un protocolo de comunicación síncrono, multi-maestro, multi-esclavo. Utiliza dos cables para compartir información. Uno se utiliza para la señal de reloj (SCL) y el otro se utiliza para enviar y recibir datos (SDA).

Todos los sensores de Smart Energy Control cuentan con una pantalla OLED 1.3” para la visualización de datos. La conexión con el ESP32 se realiza mediante I2C, de la siguiente manera:

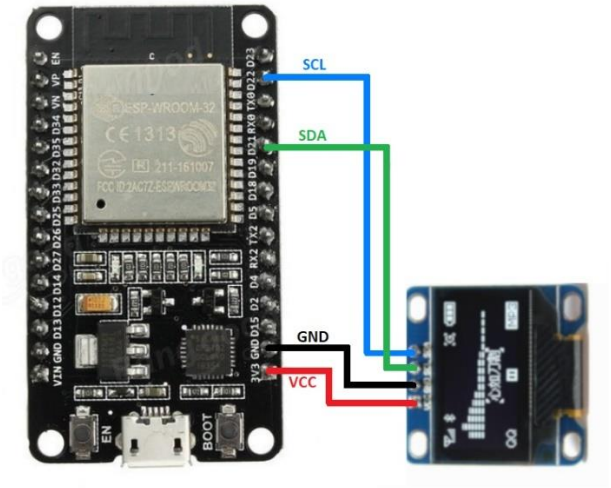

Figura 31. Interfaz I2C

- ✓ **SPI para conexión de display táctil + SD**

El bus de interfaz SPI es un estándar para el control de electrónica digital. Es un bus de tres líneas, sobre el cual se transfieren paquetes de información de 8 bits. Incluye una línea de reloj, un dato entrante y uno saliente, y un pin de chip select (CS), que conecta o desconecta la operación del dispositivo con el que uno desea comunicarse. De esta forma, se genera una multiplexación de la línea de reloj. Tomando como ejemplo el caso del display -ver Figura 32-, puede observarse que tanto la pantalla, como el módulo táctil y el módulo de la tarjeta SD comparten los pines MISO (SDO, T_DO y SD_MISO), MOSI (SDI, T_DIN y SD_MOSI) y SCK (SCK, T_CLK y SD_SCK), mientras que cada módulo se identifica con los pines CS (CS representa la pantalla, T_CS al módulo táctil, y SD_CS al módulo de la tarjeta SD).

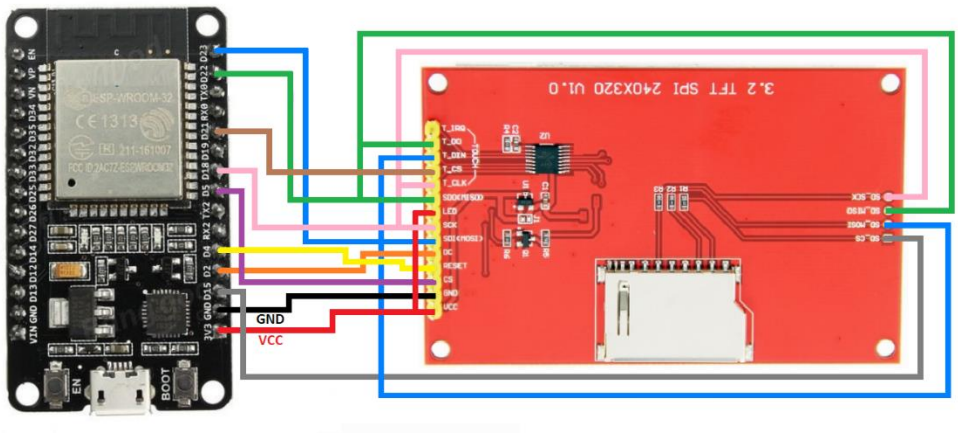

Figura 32. Interfaz SPI

UART

Los módulos UART también son conocidos como puertos Serial. Estos permiten establecer comunicaciones asíncronas entre dos dispositivos utilizando solamente dos pines. ESP32 cuenta con tres puertos UART:

- **UART 0**
- **UART 1**
- **UART 2**

UART 1 es el puerto utilizado para la conexión de la placa A6, encargada del envío de los mensajes de alertas.

2.4.2. USO DE FREERTOS

Un sistema operativo es el núcleo de código encargado de gestionar el hardware de forma independiente de los programas. FreeRTOS (Free Real-time operating system) [11], como su nombre lo indica, es un sistema operativo de tiempo real para dispositivos embebidos. Asimismo, una de sus características destacadas es su funcionalidad multitarea, de gran utilidad para implementar un proyecto donde varias partes del programa actúan de forma simultánea. Es decir, permite administrar varias tareas simultáneas con plazos de tiempo estrictos.

Hay diversas tareas que pueden implementarse en un sistema operativo de tiempo real:

- **Tareas periódicas.** Atienden eventos que ocurren constantemente y a una frecuencia determinada. Por ejemplo, el destello de un led.
- **Tareas aperiódicas.** Atienden eventos que no se saben cuándo van a ocurrir. Estas tareas están inactivas (bloqueadas) hasta que no ocurre el evento de interés. En nuestro caso, la detección de gas por parte del sensor.
- **Tareas de procesamiento continuo.** Tareas que trabajan en forma permanente.

Un aspecto importante es que **no todas estas tareas requieren la misma frecuencia de operación.** Por ejemplo, la lectura de los sensores tendrá que realizarse más frecuentemente que, por ejemplo, el cambio de estado de un interruptor para control de una pantalla.

El otro punto importante es que **no todas esas tareas tienen la misma prioridad.** Por ejemplo, la lectura de un sensor de presencia de humo o monóxido de carbono tendrá una prioridad más alta a, por ejemplo, mostrar los datos en la pantalla.

Teniendo en cuenta estos conceptos, los códigos implementados -tanto para el gabinete principal como los diversos sensores- se desarrollaron mediante la definición de funciones específicas (tareas), donde se buscó un equilibrio de tal forma de maximizar su funcionamiento. En líneas generales, la mayoría de las tareas tienen la misma prioridad; es decir, en cada ciclo de operación, se realiza una lectura de todos los sensores, y los datos son enviados a los servidores. Aquellas que son aperiódicas, corresponden al control táctil del display, y la generación de las alarmas. Sin embargo, esta última es una tarea crítica, que debe dar respuesta inmediata ante la ocurrencia de un evento. Por este motivo, podemos decir que esta última tarea debe tener una alta prioridad, aunque no debe perderse de vista que la asignación de una prioridad alta a una tarea, puede provocar que otra de menor prioridad, pero de mayor frecuencia, quede solapada.

2.4.3. DECLARACIÓN DE FUNCIONES/TAREAS

En nuestra implementación, la multitarea funcionará de la siguiente forma:

- Cuando una tarea esté en estado preparado (ready), y no haya otra tarea ejecutándose, pasará a estado de ejecución (running).
- Cuando la tarea acabe su parte del código se bloqueará mediante la llamada a la función vTaskDelay(Ticks) que permitirá pasar la tarea al estado bloqueado (blocked) durante un tiempo Ticks, establecido en milisegundos.
- En el caso de que haya dos tareas preparadas para pasar a ejecución, la que tenga una menor prioridad pasará a estado de ejecución.
- Si una tarea está ejecutándose y llega otra de mayor prioridad, la de mayor prioridad pasará a ejecución.

Para programar una tarea, se debe crear una función que contenga el código que desea ejecutar y luego crear una tarea que llame a esta función. Al crear una tarea, se puede elegir en qué núcleo se ejecutará, así como su prioridad. Los valores de prioridad comienzan en 0, donde 0 es la prioridad más baja. El procesador ejecutará primero las tareas con mayor prioridad.

CREACIÓN DE UNA TAREA

Se debe crear un identificador de la tarea:

```
TaskHandle_t Tarea1;
```

Luego, la declaración de la tarea:

```
xTaskCreatePinnedToCore(
      CodigoTarea1, /* Función para implementar la tarea */
      "Tarea1", /* Nombre de la tarea */
      10000,  /* Tamaño de pila en palabras */
      NULL,  /* Parámetro de entrada de tarea */
      0,  /* Prioridad de la tarea */
      &Tarea1,  /* Identificador de tareas */
      0); /* Núcleo donde debe ejecutarse la tarea */
```

Por último, se debe crear la función que contenga el código para la tarea creada:

```
void CodigoTarea1( void * parameter) {
      for(;;) {
            //Código para la tarea 1 en un bucle infinito
             (...)
      }
}
```

El for(;;) crea un bucle infinito, por lo que esta función se ejecutará de manera similar al loop().

Para eliminar la tarea creada, se utiliza la función vTaskDelete(), que acepta el identificador de tareas (Tarea1) como argumento:

```
vTaskDelete(Tarea1);
```

Una función utilizada, común tanto al gabinete principal como a los sensores, es la que realiza el control del led azul indicando el estado de conexión WiFi del dispositivo, de la siguiente manera: si el led se encuentra encendido fijo, el dispositivo se encuentra conectado, mientras que, si led parpadea, no hay conexión establecida.

Se encuentra declarada de la siguiente manera:

```
TaskHandle_t ntLEDTaskHandler;

void control_led_wifi(){
  vTaskDelay(10);
  xTaskCreate(Control_LED_WiFi,
              "LedWiFi",
              4096,
              NULL,
              0,
              &ntLEDTaskHandler
              0);
}

void Control_LED_WiFi(void *pvParameters){
  vTaskDelay(10);
  while(1) {
    if(WiFi.status() != WL_CONNECTED) {
      digitalWrite(LED_ESP, HIGH);
      vTaskDelay(1000);
      digitalWrite(LED_ESP, LOW);
      vTaskDelay(1000);
    }else{
      digitalWrite(LED_ESP, HIGH);
    }
  }
}
```

Como se puede observar, la función se encuentra asociada al núcleo 0 del procesador, con prioridad 0.

2.4.3.1. GABINETE

En el Gabinete, la multitarea se implementó de la siguiente manera:

Núcleo 0	Núcleo 1
Control LED WiFi	Conexión WiFi
Lectura Sensores	Conexión Servidor

Tabla V. Asignación de Tareas en Gabinete

2.4.3.2. SENSOR DE TEMPERATURA

En el Sensor de Temperatura, la multitarea se implementó de la siguiente manera:

Núcleo 0	Núcleo 1
Control LED WiFi	Conexión WiFi
Lectura Sensor TyH	Conexión Servidor
Pantalla OLED	

Tabla VI. Asignación de Tareas en Sensor Temperatura

2.4.3.3. SENSOR DE GAS

En el Sensor de Gas, la multitarea se implementó de la siguiente manera:

Núcleo 0	Núcleo 1
Control LED WiFi	Conexión WiFi
Lectura Sensor Gas	Conexión Servidor
Pantalla OLED	

Tabla VII. Asignación de Tareas en Sensor Gas

2.4.4. USO DE INTERRUPCIONES

La interrupción funciona como una suspensión temporal de la ejecución de un proceso, para pasar a ejecutar una subrutina de servicio de interrupción. Una vez finalizada dicha subrutina, se reanuda la ejecución del programa.

Su uso es útil ya que, si la acción necesita ser atendida inmediatamente, esperar hasta el punto de programa donde se realiza la consulta puede ser inaceptable.

Para su implementación, se requiere:

- Un pin del microcontrolador que recibirá la señal de disparo.
- Una condición de disparo.
- Una función, que se ejecutará cuando se dispara la interrupción (Llamada call back function).

Las condiciones de disparo pueden ser diversas. En la Tabla VIII se detallan las alternativas posibles.

DISPARO DE INTERRUPCIONES					
Condición	LOW	CHANGE	RISING	FALLING	HIGH
Disparo de Interrupción	El estado del pin es LOW	El pin cambia de LOW a HIGH, o viceversa	En el flanco de subida (el pin cambia de LOW a HIGH)	En el flanco de bajada (el pin cambia de HIGH a LOW)	El estado del pin es HIGH

Tabla VIII. Interrupciones

Todos los sensores construidos para nuestro Proyecto, cuentan con un pulsador que opera mediante una interrupción, a través de la cual se realiza un intercambio entre distintas pantallas mostradas en el display OLED. Cada vez que el usuario aprieta el pulsador, se acciona esta interrupción y el display modifica la información presentada, entre las siguientes alternativas:

- Valores de temperatura y humedad actuales, en el caso del sensor de temperatura; o calidad del aire en caso del sensor de gas.
- Fecha y hora actual.
- Datos del sensor (nombre, id asignado).
- Datos de la red WiFi (nombre de la red, IP asignada, y nivel de señal de la conexión).

La declaración se realizó de la siguiente manera:

```
/****************************************
 Declaración de Pines
 ****************************************/
#define SW_RES 15                  //PULSADOR

/*****************************************************
 Funciones Auxiliares
 *****************************************************/

unsigned long lastDetection = 0;
unsigned long debounceTime = 500;

void IRAM_ATTR SWITCH(){
  if(millis() - lastDetection > debounceTime){
    if(displayScreenNum < displayScreenNumMax){
    displayScreenNum++;
    }else
      displayScreenNum = 0;
    lastDetection = millis();
  }
}

void setup(){
  .
  .
  attachInterrupt(digitalPinToInterrupt(SW_RES), SWITCH, RISING);
  .
  .
}
```

Como puede observarse en las líneas de código, se declara la interrupción en la función principal setup(), donde se indica que la misma está asociada al pin definido como SW_RES, la función asociada es IRAM_ATTR SWITCH(), y la condición de disparo es RISING, cuando el pin cambia de LOW a HIGH (ver Tabla VIII). Cada vez que se acciona el pulsador, se activa la interrupción y el valor de la variable 'displayScreenNum' se incrementa en 1 para alternar las distintas pantallas mostradas en el display OLED. Cuando la variable llega a su valor máximo, vuelve a 0 (pantalla inicial).

2.5. Redes de datos. Protocolos de comunicación

2.5.1. Comunicación WiFi

La comunicación WiFi es la principal comunicación utilizada por el dispositivo. Por tal motivo, se requiere que el mismo se conecte a una red disponible. Smart Energy Control permite 2 alternativas de conexión.

2.5.1.1. Protocolo TCP/IP

TCP/IP corresponde a las siglas de Transmission Control Protocol/Internet Protocol (Protocolo de control de transmisión/Protocolo de Internet). Representa un conjunto de reglas estandarizadas que permiten a los equipos comunicarse en una red como Internet. Es una conexión bidireccional a través de la cual los datos pueden fluir en ambas direcciones.

Si bien conforman un conjunto, TCP e IP son dos protocolos distintos para redes informáticas. IP es la parte que obtiene la dirección a la que se envían los datos, mientras que TCP se encarga de la entrega de los datos una vez hallada dicha dirección IP.

Es el protocolo predeterminado en la ESP32 y usa WiFi como transporte. Su funcionamiento, en simples líneas, se basa en la división de tareas a través de cuatro capas, donde cada una de ellas tiene una función distinta, pero en su conjunto forman un sistema estandarizado.

Un módulo ESP32 puede configurarse como un oyente o punto de acceso para una conexión TCP/IP.

2.5.1.2. Administrador de conexión inalámbrica

En este método, se puede configurar un punto de acceso y una estación. La configuración se realiza a través de un portal en el navegador.

Cómo funciona

Al realizar la configuración inicial de un sensor, en primera instancia se deberá realizar la conexión del mismo a una red WiFi disponible. Para ello, se iniciará un portal de configuración WiFi y se guardarán los datos de la

red y contraseña en la memoria no volátil. A partir de entonces, el portal de configuración sólo se iniciará nuevamente si se presiona un botón en el módulo ESP. De lo contrario, utilizará los datos almacenados para la conexión al iniciar el dispositivo.

A continuación, se detalla el flujo de configuración paso a paso:

1. El sensor ingresará al modo Access Point, generando una red para que cualquier dispositivo habilitado para WiFi, se conecte a él. Una vez vinculados, con un navegador, se deberá ingresar a la dirección 192.168.4.1.

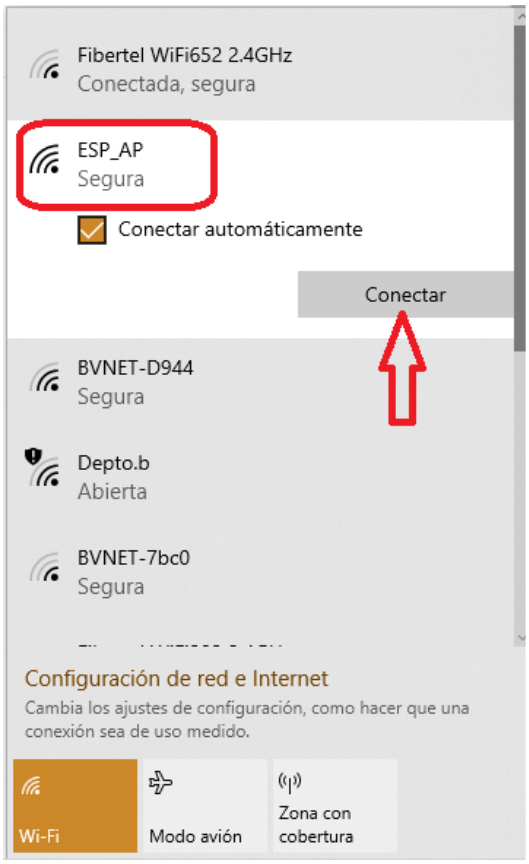

Figura 33. Portal conexión WiFi – Paso 1

2. Se deberá seleccionar la opción “Configure WiFi” para conectarse a una red WiFi existente:

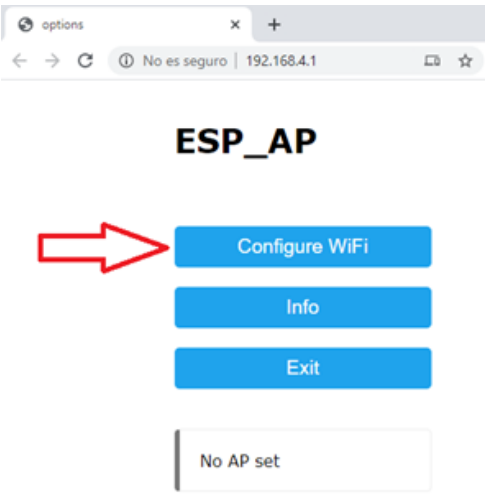

Figura 34. Portal conexión WiFi – Paso 2

3. Luego elegir una de las redes e ingresar la contraseña (si es necesario). Luego guardar y esperar a que ESP se reinicie.

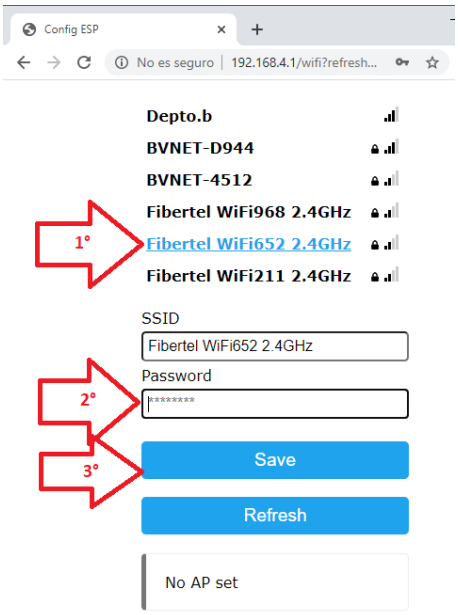

Figura 35. Portal conexión WiFi – Paso 3

4. Cada vez que se inicie, el sensor intentará conectarse a la red guardada. Si esto no es posible, nuevamente se habilitará como punto de acceso, para repetir los pasos anteriormente indicados.

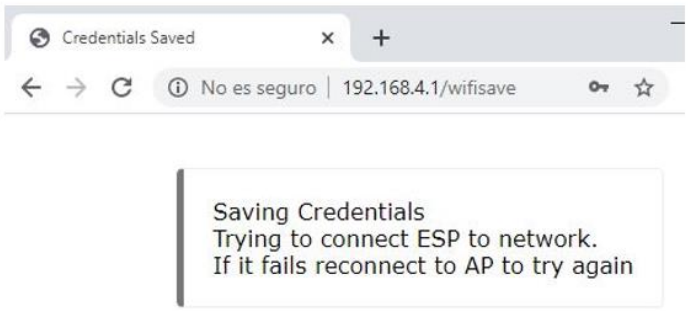

Figura 36. Portal conexión WiFi – Paso 4

2.5.2. Comunicación ESP-NOW

ESP-NOW es un protocolo privado, desarrollado por el fabricante de las placas Espressif, similar a la conectividad inalámbrica de 2,4 GHz de baja potencia. Esta comunicación permite que varios dispositivos se comuniquen entre sí sin usar Wi-Fi. El emparejamiento entre dispositivos es necesario antes de su comunicación.

Una vez que se realiza el emparejamiento, la conexión es persistente; es decir, si de repente una de las placas pierde energía o se reinicia, tras dicho reinicio, se conectará automáticamente a su par para continuar la comunicación. Por otro lado, se debe considerar que, si bien es una comunicación rápida, se puede usar para intercambiar mensajes pequeños de hasta 250 bytes.

Cada dispositivo se puede comunicar con un máximo de 20 pares. Entre sus características se pueden destacar la velocidad de la conexión entre dispositivos –aproximadamente 10 veces más rápidas que una conexión WiFi–, y la reducción del consumo eléctrico, primordial en el caso de dispositivos alimentados por baterías.

Entre las distintas configuraciones de comunicación, se puede mencionar:

- Un maestro y un esclavo
- Un maestro y varios esclavos
- Varios maestros y un esclavo

Para la implementación de Smart Energy Control, se utilizó la comunicación bidireccional, donde el gabinete establece una comunicación de ida y vuelta con cada uno de los sensores conectados. En la Figura 37 se representa esta topografía.

Figura 37. Comunicación bidireccional ESP-NOW

De esta manera, cuando se produce un corte de a conexión WiFi, no habrá una pérdida en la comunicación de los distintos dispositivos, ya que continuarán haciéndolo mediante el protocolo ESP-NOW. En esta comunicación bidireccional implementada, el gabinete y los diversos sensores funcionarán como maestros y esclavos en forma simultánea.

En primer lugar, como requisito fundamental para esta conexión, se debe conocer la dirección MAC, mediante el siguiente código:

```
#include <WiFi.h>

void setup(){
  Serial.begin(115200);
  Serial.println();
  Serial.print("Dirección MAC:  ");
  Serial.println(WiFi.macAddress());
}

void loop(){

}
```

A través del monitor serie, se obtiene esta dirección.

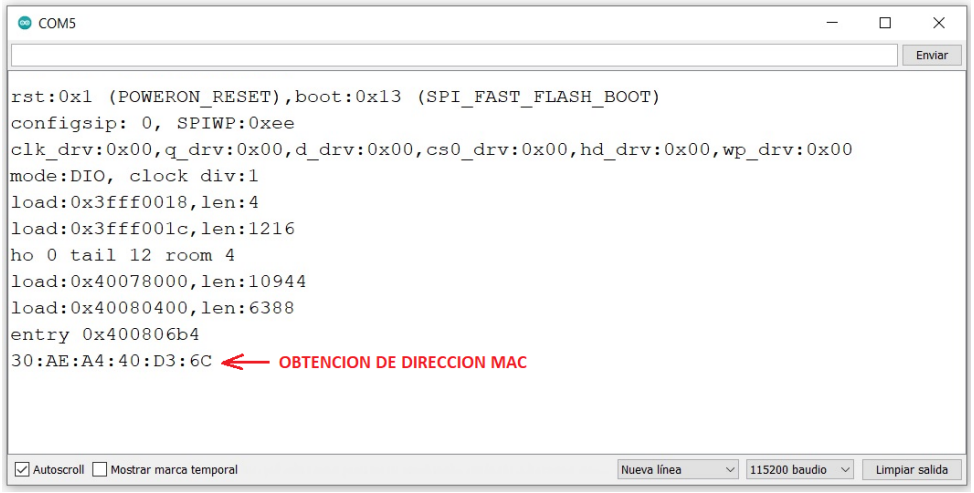

Figura 38. Obtención de dirección MAC a través de monitor serie

Una vez obtenida la dirección MAC, se deberán crear las estructuras de datos y funciones de envío/recepción de los datos. A continuación, se ejemplifica el caso de la estructura para un sensor de humo:

```
//Estructura de envío de datos sensor Humo
typedef struct struct_message {
    int id;
    float humo;
    int readingId;
} struct_message;
```

Es decir, la estructura estará conformada por un identificador del sensor que envía los datos (int id), el valor medido de presencia de humo en el ambiente (float humo), y el número de medición (int readingId).

En el caso de los sensores de temperatura, esta estructura será análoga, con el reemplazo de la variable float humo, por las variables float temp y float humedad, para medición de la temperatura y humedad del ambiente, respectivamente.

```
//Estructura de envío de datos sensor TyH
typedef struct struct_message {
    int id;
    float temp;
    float humedad;
    int readingId;
} struct_message;

//Se crea una "struct_message" llamda Datos
struct_message Datos;
```

Luego, las funciones para el envío y recepción de datos:

Función de envío de datos

```
esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &Datos, sizeof(Datos));
```

Función de recepción de datos

```
// Función callback a ejecutar cuando se reciben datos por ESP_now
void OnDataRecv(const uint8_t * mac_addr, const uint8_t *incomingData, int len) {
  // Copies the sender mac address to a string
  char macStr[18];
  Serial.print("Packet received from: ");
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
           mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4], mac_addr[5]);
  Serial.println(macStr);
  memcpy(&incomingReadings, incomingData, sizeof(incomingReadings));
  Serial.printf("Board ID %u: %u bytes\n", incomingReadings.id, len);
  Serial.printf("humo value: %4.2f \n", incomingReadings.humo);
  Serial.printf("readingID value: %d \n", incomingReadings.readingId);
  if(incomingReadings.humo > 2000)
    Serial.printf("Hay humo en el ambiente\n");
  else
    Serial.printf("No hay humo en el ambiente\n");
  Serial.println();
}
```

En la Tabla IX se representan otras funciones utilizadas para la ejecución de este protocolo.

Nombre Función	Detalles
esp_now_init()	Inicializa ESP-NOW
esp_now_add_peer()	Empareja un dispositivo. Requiere como argumento la dirección MAC del par
esp_now_send()	Envío de datos con ESP-NOW
esp_now_register_send_cb()	Cuando se envía un mensaje, se llama a una función; esta función devuelve si la entrega fue exitosa o no
esp_now_register_rcv_cb()	Cuando se reciben datos a través de ESP-NOW, se llama a una función

Tabla IX. Funciones Protocolo ESP-NOW

2.5.3. COMANDOS AT

Los comandos AT son instrucciones codificadas que conforman un lenguaje de comunicación entre el hombre y un terminal modem.

Aunque la finalidad principal de los comandos AT es la comunicación con módems, la telefonía móvil GSM también ha adoptado como estándar este lenguaje para poder comunicarse con sus terminales.

Para su implementación en el proyecto, se utilizó el módulo A6 GSM/GPRS [12], un módem GSM en miniatura, que cuenta con las características básicas de un teléfono celular: permite enviar o recibir mensajes de texto (SMS), realizar o recibir llamadas telefónicas, conectarse a internet a través de GPRS o TCP/IP. Además, el módulo es compatible con la red GSM/GPRS de cuatro bandas, lo que significa que funciona prácticamente en cualquier parte del mundo.

Por lo tanto, el módulo A6 se comanda a través de comandos AT, en donde se testea su estado, luego se lo configura, se verifica la intensidad de señal y se procede al envío del mensaje, de acuerdo a las alertas configuradas.

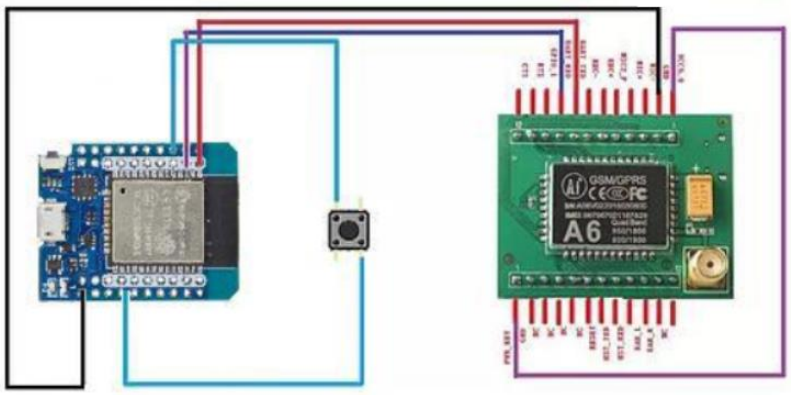

Figura 39. Conexión Módulo A6 con ESP32

IMPLEMENTACIÓN

Para utilizar el módulo en el dispositivo, no se requieren librerías, sino que se utiliza uno de los puertos seriales para establecer la comunicación AT. En nuestro caso utilizamos el puerto serial 1, modificando los pines de recepción y transmisión de dicho serial en el archivo HardwareSerial.cpp.

```
#ifndef RX1
#define RX1 26
#endif

#ifndef TX1
#define TX1 25
#endif
```

Luego el código cuenta con la inicialización del serial en la función principal setup(), un pin para el encendido del módulo y una variable con el número de teléfono a donde se enviará el SMS:

```
int pwr = 33;
String numero_A6;
void setup()
{
  Serial1.begin(115200);
  pinMode(pwr, OUTPUT);
  digitalWrite(pwr, LOW);
}

void loop()
{
  digitalWrite(led, LOW);
  sendSMS(message);
}

void updateSerial()
{
  while (Serial.available())
    Serial1.write(Serial.read());
  while (Serial1.available())
    Serial.write(Serial1.read());
}
```

La función principal enciende el módulo por primera vez y verifica la comunicación con el comando “AT”, estableciéndolo en modo de envío de SMS con el formato predeterminado, que es Protocol Data Unit (PDU). Posteriormente, envía el SMS y apaga el módulo.

La función updateSerial() se encarga de la comunicación entre el Serial 0 con el Serial 1 y viceversa, para lograr la comunicación AT.

```
void sendSMS() {
  if(on_A6 <1){
   Serial.println(F("GSMmodule ready... se está prendiendo"));
   digitalWrite(A6_PWR, HIGH);
   delay(2000);
   digitalWrite(A6_PWR, LOW);
   delay(3000);

  Serial1.println("AT\r");
  updateSerial();
  delay(3000);

  Serial1.println("AT+CMGF=1\r"); //Configurar el módulo en modo SMS
  updateSerial();
  delay(3000);

  Serial1.println("AT+CNMI=2,2,0,0,0\r"); //Configurar el módulo para enviar
  updateSerial();                         //datos SMS en serie al recibirlos
  delay(3000);


   Serial1.print("AT+CMGD=1,4\r"); //eliminar todo el mensaje
   updateSerial();                 //en el almacenamiento
   delay(3000);

   on_A6=1;
   }

   Serial1.print("AT+CMGS = +54");
   Serial1.print(numero_A6);
   Serial1.println("\r");
   updateSerial();
   delay(100);
   // Envio de SMS
   Serial1.print("Se a detectado un corte de luz.");
   delay(100);
   // Termine el comando AT con ^ Z, código ASCII 26
   Serial1.println((char)26);
   updateSerial();
   delay(100);
   Serial1.println();
   // Dar tiempo al módulo para enviar SMS
   delay(5000);

   //Apagado del A6
   Serial1.print("AT+CPOF\r");
   updateSerial();
   delay(1000);
   on_A6=0;
}
```

2.6. Servidor de datos y aplicación Android

2.6.1. Base de datos de Google Firebase

El principal servidor de datos utilizado es Firebase. Firebase Realtime Database[13] es una base de datos gratuita alojada en la nube. Es un almacén de datos NoSQL al que sus clientes pueden acceder directamente en tiempo real.

A diferencia de SQL, en Firebase no hay un esquema para la base de datos, ni tablas, ni columnas, es solo una combinación de pares clave/valor, conformando un árbol de datos, al que se accede mediante rutas determinadas. Permite realizar una administración de usuarios registrados, los que accederán mediante una dirección de correo electrónico y contraseña. De esta manera, se conformará una base de datos, donde a cada uno de ellos se le asigna un ID único en forma automática, siendo ésta la manera de identificarlos, y la utilizada para organizar la información de cada uno de ellos. En la Figura 40 se observa la lista de los usuarios registrados.

Figura 40. Registro de usuarios en servidor Firebase

Sus funciones en tiempo real son adecuadas para grandes bases de datos, ya que fueron diseñadas para grandes conjuntos de datos. De esta manera, puede contener millones de usuarios, con soporte para manejar 1 millón de conexiones simultáneas y 10.000 escrituras por segundo.

En la Figura 41 se muestra de qué forma se organiza la información dentro de la base de datos, en la forma de árbol anteriormente mencionada:

Figura 41. Organización de datos en Firebase

Para poder acceder al servidor, a través de los sensores, se deberán definir el Host y la clave de autenticación. Asimismo, se deberá definir un objeto de Firebase, en este caso llamado firebaseData.

```
#define Firebase_Host "smartenergycontrol-29552.firebaseio.com"
#define Firebase_Auth "AUs1cvGlTwVyk5t9WIHgVuSRuKGoWMaayEbyI75a"

//Define un objeto de Firebase
FirebaseData firebaseData;
```

Luego, para acceder a la información dentro de la base de datos, se deberá especificar la ruta de acceso, organizada en forma de árbol. Por ejemplo, si se quiere acceder al sensor de temperatura declarado –tomando como modelo la Figura 42– la ruta se conformará de la siguiente manera:

```
String path_sensor = "/Usuarios/Id/Sensores/Temp_Cocina";
```

Figura 42. Rutas de acceso a datos en Firebase

Si luego se quiere acceder a alguna variable correspondiente al mencionado sensor –por ejemplo, el valor de la variable Temperatura– se realizará de la siguiente manera:

```
String path_sensor = "/Usuarios/Id/Sensores/Temp_Cocina/Temperatura";
```

De esta misma manera, se realizará el acceso a las variables restantes.

Si se desea obtener el valor de una variable, se realizará mediante la función GET. Continuando con el ejemplo precedente, el valor de la variable Temperatura se obtiene de la siguiente manera:

```
Firebase.getInt(firebaseData, path_sensor);
```

Si por el contrario, se quisiera modificar el valor, se realiza mediante la función SET:

```
Firebase.setInt(firebaseData, path_sensor,24);
```

siendo:

```
path_sensor = "/Usuarios/Id/Sensores/Temp_Cocina/Temperatura";
```

En ejemplo anterior, se trabajó con variables enteras, pero de manera análoga se efectúan estas mismas operaciones con otros tipos de variables, según se indica en la Tabla X.

Función	Tipo de variable	Detalles
GET	Firebase.getInt(Objeto_Firebase, Ruta de acceso)	Lectura de una variable Int de la base de datos
	Firebase.getFloat(Objeto_Firebase, Ruta de acceso)	Lectura de una variable Float de la base de datos
	Firebase.getString(Objeto_Firebase, Ruta de acceso)	Lectura de una variable String de la base de datos
SET	Firebase.getInt(Objeto_Firebase, Ruta de acceso)	Escritura de una variable Int en la base de datos
	Firebase.getFloat(Objeto_Firebase, Ruta de acceso)	Escritura de una variable Float en la base de datos
	Firebase.getString(Objeto_Firebase, Ruta de acceso)	Escritura de una variable String en la base de datos

Tabla X. Funciones de Firebase

DATOS JSON

Todos los datos de Firebase Realtime Database se almacenan como objetos JSON. La base de datos puede conceptualizarse como un árbol JSON alojado en la nube. A diferencia de una base de datos de SQL, no hay tablas ni registros. Al agregar datos al árbol JSON, estos se convierten en un nodo de la estructura JSON existente con una clave asociada.

A continuación, se representa un ejemplo de su utilización en Smart Energy Control:

```
json.add("Tipo_Alarma", alarm_type);
json.add("Nombre_Sensor", nombre_sensor);
json.add("Dia", day);
json.add("Mes", month);
json.add("Ano", year);
json.add("Horas", hour);
json.add("Minutos", minute);
Firebase.pushJSON(firebaseData, path_alarmas, json);
```

De esta manera, cada vez que se genere una alarma de detección de humo, se almacenará en la base de datos una nueva alarma, con la información detallada en la declaración, que corresponde al tipo de alarma, el nombre del sensor que dio origen a la misma, y la fecha.

2.6.2. Base de datos de Thingspeak

El servidor restante utilizado, es Thingspeak [14], una aplicación de "Internet de las cosas de código abierto" (IoT) que permite la comunicación entre dispositivos, sitios web y web services por medio de APIs sencillas. El objetivo principal de su utilización es la de registrar un historial de los valores medidos por los sensores, junto a la realización de los gráficos temporales para su representación.

Las comunicaciones de equipos en una red se realizan a través de protocolos. El modelo OSI es una representación abstracta de la comunicación entre procesos. Se ocupa de los estándares para la comunicación entre sistemas. TCP/IP combina varias capas de dicho modelo OSI en una única capa. Las capas de nivel alto (aplicación, presentación y sesión) son reducidas en sólo una capa, denominada “capa aplicación”. Esta última es la utilizada por Thingspeak.

Thingspeak utiliza el protocolo de comunicación HTTP (Protocolo de Transferencia de Hipertexto), a través de Internet o a través de una red de área local. HTTP permite las transferencias de información; es decir, está orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. El cliente realiza una solicitud enviando un mensaje, con cierto formato al servidor y el servidor le envía un mensaje de respuesta.

Existen funciones predefinidas de HTTP que pueden utilizarse, donde cada una de ellas indican qué acción efectúa sobre un recurso identificado. Estas funciones son las siguientes:

- **GET.** Pedir al servidor un servicio.
- **POST.** Actualizar una variable creando un nuevo registro.
- **PUT.** Modificar un valor de un registro.
- **DELETE.** Borrar un valor de un registro.

Para su implementación, se creó un canal denominado “Parámetros”, cuya identificación se realiza a través del indicador Channel ID. Una vez configurado, se pueden comenzar a escribir y leer los datos con el uso de las APIs “Write API Key” y “Read API Key”, respectivamente.

Figura 43. Configuración canal Thingspeak

Este canal permite el manejo de 8 (ocho) variables, enumeradas a continuación:

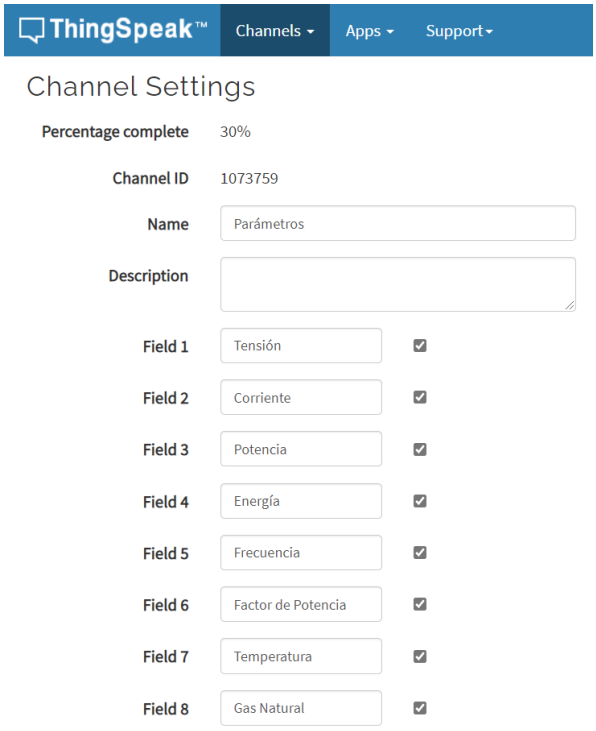

Figura 44. Campos canal Thingspeak

Como se puede observar, se utilizan 6 (seis) campos correspondientes a la medición de los parámetros de la red eléctrica, mientras que los 2 (dos) restantes se utilizan para la medición de los parámetros del ambiente.

Se debe tener en consideración que el tiempo mínimo de subida al servidor entre una serie de valores y otra, es de al menos 15 (quince) segundos.

El acceso a Thingspeak se realiza mediante la definición de diversos parámetros, correspondientes a la URL, la dirección IP y el acceso al puerto 80 por el cual un servidor HTTP "escucha" la petición hecha por un cliente.

Estos parámetros se declaran de la siguiente manera:

```
#define THINGSPEAK_URL "api.thingspeak.com"
#define THINGSPEAK_IPADDRESS IPAddress(184,106,153,149)
#define THINGSPEAK_PORT_NUMBER 80
```

Por otro lado, se deberá identificar el canal al que se subirán los datos:

```
unsigned long channelID = 1605215;
const char* writeAPIKey = "8REJH8KBZJFGUWNX";
const char* server = "api.thingspeak.com";
const int intervalo = 15000;
```

"Channel ID" hace referencia al número de canal asignado por el servidor al momento de crear el usuario, mientras que **"Write API Key"** es la llave de escritura es la que permite acceder al servidor para cargar un valor. Estos datos son asignados automáticamente por Thingspeak al crear un nuevo canal.

El **intervalo** define el tiempo que debe transcurrir entre la carga de un valor y otro. En su versión gratuita –utilizada en el Proyecto–, los datos pueden ser enviados con un muestreo de 15 segundos (o 15000 ms) por canal.

Una vez definidas estas variables, se deberá iniciar el servidor:

```
ThingSpeak.begin(client);
```

La función "ThingSpeak.getLastReadStatus()" permite conocer la situación de la conexión; si el dato pudo ser cargado al servidor o si, por el contrario, hubo algún inconveniente.

```
int valor = ThingSpeak.getLastReadStatus();
Serial.print("Estado del Canal: ");
Serial.print(valor);
Serial.println();
```

Para efectuar la conexión al servidor, en primera instancia intenta el nexo mediante la URL definida previamente. En caso de no poder establecer la conexión, se realiza el intento con la dirección IP. El resultado obtenido es almacenado en la variable booleana "connectSuccess", siendo "true" si logró establecer la conexión con éxito, o "false" en caso de no haber sido posible.

```
bool connectThingSpeak()
{
        bool connectSuccess = false;
        if(this->customIP == INADDR_NONE && NULL == this->customHostName)
 {
                #ifdef PRINT_DEBUG_MESSAGES
                    Serial.print("              Connect to default ThingSpeak URL...");
                #endif
                connectSuccess = client->connect(THINGSPEAK_URL,THINGSPEAK_PORT_NUMBER);
     if(!connectSuccess)
     {
                    #ifdef PRINT_DEBUG_MESSAGES
                        Serial.print("Failed. Try default IP...");
                    #endif
                    connectSuccess = client->connect(THINGSPEAK_IPADDRESS,THINGSPEAK_PORT_NUMBER);
     }
}
```

Retornado un error de falla de conexión al servidor (Error -301) si la variable booleana connectSuccess es falsa.

```
if(!connectThingSpeak())
{
        // Failed to connect to ThingSpeak
        return ERR_CONNECT_FAILED;
}
```

Una vez establecida la conexión, se definen el número de canal y la llave de escritura (APIKey):

```
ThingSpeak.setField(campo,dato);
```

Por último, se define el campo al cual se quiere subir el valor, seguido del valor en sí mismo, mediante la siguiente instrucción:

```
ThingSpeak.writeFields(numeroCanal, llaveEscritura);
```

Tomando como ejemplo su implementación en el Gabinete Principal del Proyecto, mediante las siguientes líneas:

```
if (client.connect(server, 80)){     //conexión a través del puerto 80
      ThingSpeak.writeFields(channelID,writeAPIKey);
      ThingSpeak.setField (1,t);
      ThingSpeak.setField (2,h);
    }
```

Se subirán a los campos 1 y 2 del canal, los valores de las variables 't' y 'h', representando los valores de temperatura y humedad del sensor, respectivamente.

Una vez que los datos son subidos al canal, se pueden leer con una API de visualización. Además, ThingSpeak provee un bloque de código HTML para poder copiar directamente en la página web y visualizar la gráfica del canal si este último está configurado como público.

En las Figuras 45 y 46 se realiza la representación de las gráficas de temperatura y humedad, respectivamente, para el sensor. Asimismo, se indican las líneas de código necesarias para su visualización.

Temperatura

https://thingspeak.com/channels/1605215/charts/1?bgcolor=%23ffffff&color=%230f2a7a&dynamic=true&results=60&title=Temperatura&type=line

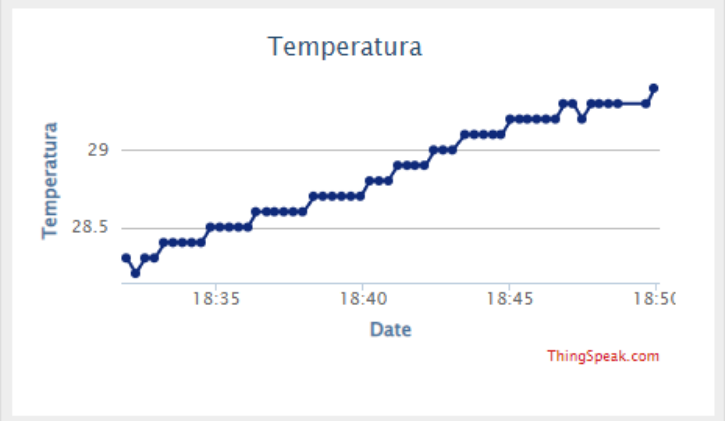

Figura 45. Gráfico Temperatura vs Tiempo (Uso de HTML)

Humedad

https://thingspeak.com/channels/1605215/charts/2?bgcolor=%23ffffff&color=%23bc0ecc&dynamic=true&results=60&title=Humedad&type=line

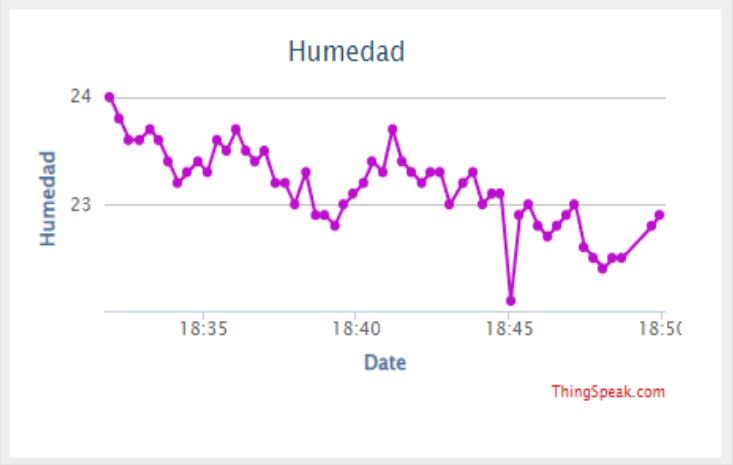

Figura 46. Gráfico Humedad vs Tiempo (Uso de HTML)

2.6.3. Aplicación Smart App

Se ha desarrollado una aplicación para dispositivos Android con el fin de establecer una comunicación con el dispositivo, mediante el uso del servidor Thinkspeak desarrollado en el ítem anterior. Asimismo, se apoya en la plataforma Firebase.

El diagrama de la Figura 47 representa el funcionamiento de los sensores.

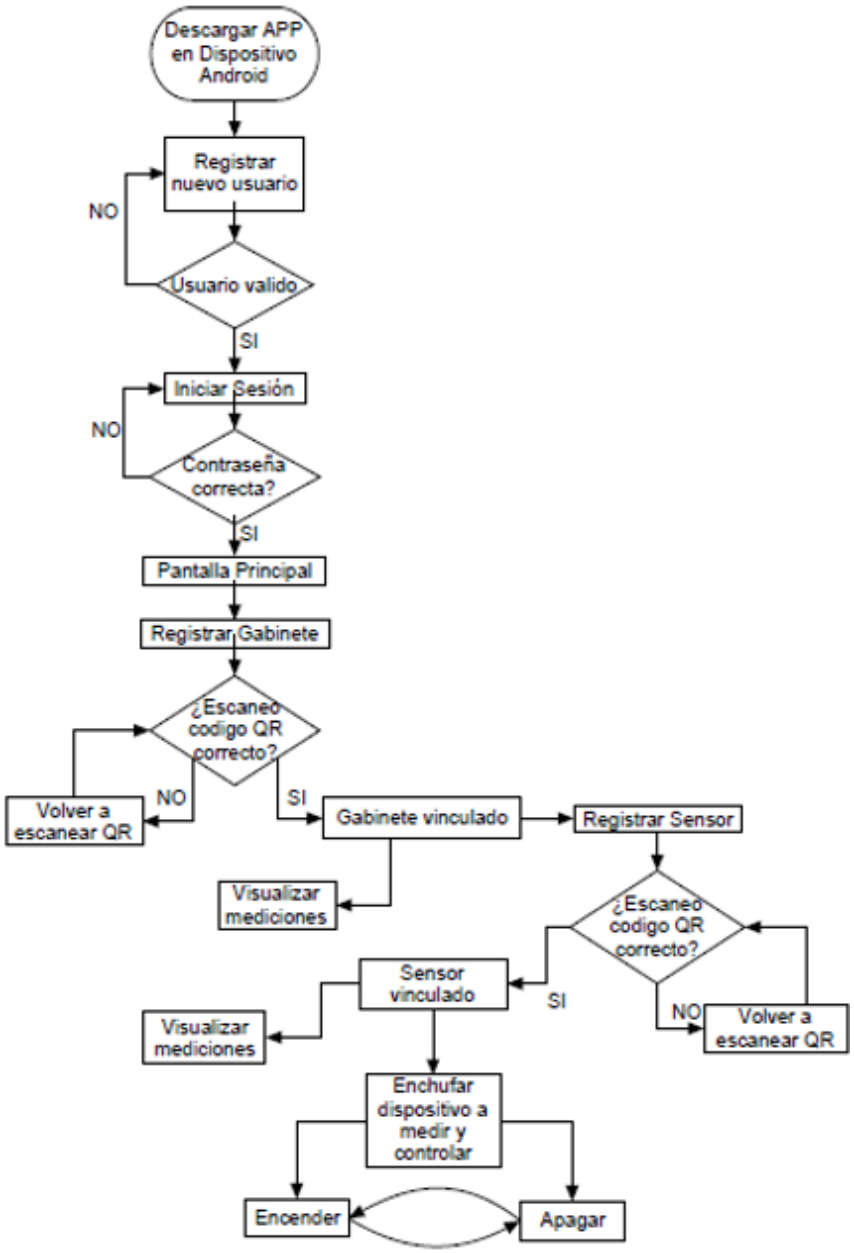

Figura 47. Diagrama de Bloques Sensores

La App ha sido programada mediante Kodular [15], el que dispone de un kit de desarrollo de software mediante un lenguaje orientado a objetos y de código abierto. En la Figura 48 se observa un ejemplo de programación para la pantalla de inicio de sesión.

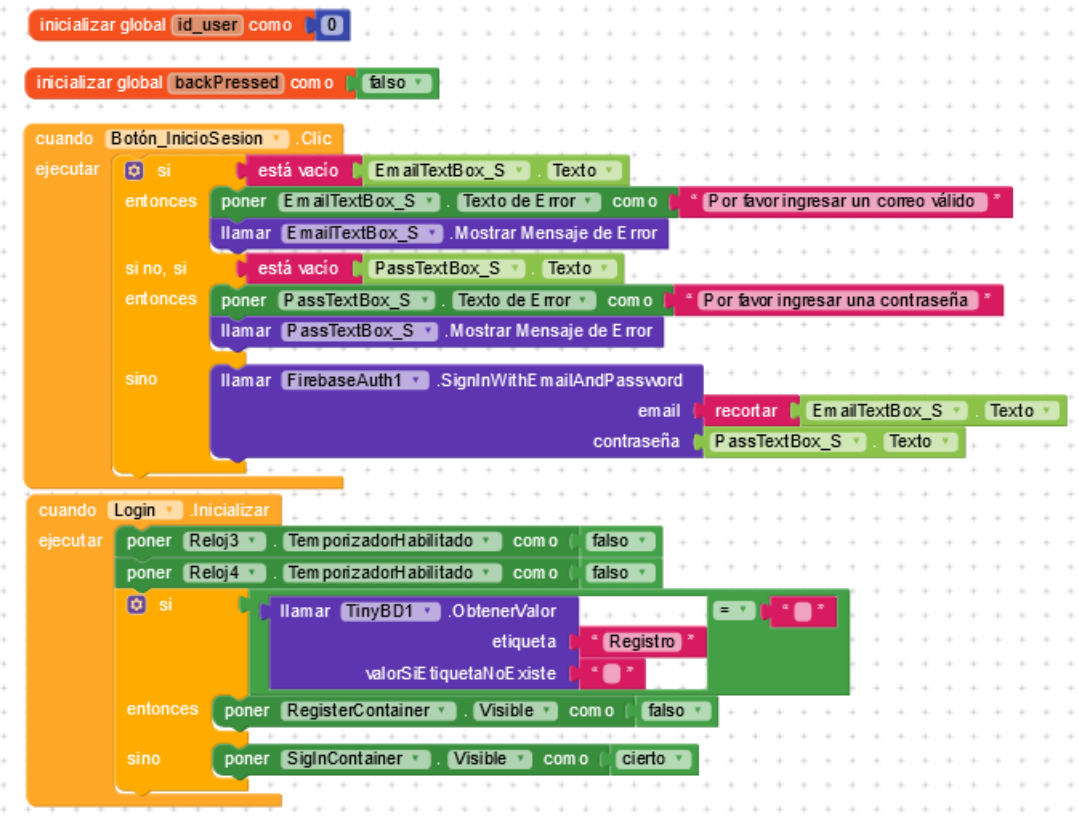

Figura 48. Smart App – Programación en bloques

Esta aplicación brinda las funciones enumeradas a continuación:

- Visualización, en tiempo real, de los parámetros medidos por el dispositivo.
- Visualización de gráficas de estos parámetros, con un historial de mediciones.
- Solapa para el control del gabinete principal, y solapa para el control de los diversos sensores configurados.
- Solapa de notificaciones/alarmas.
- Solapa con los datos del usuario, con posibilidad de modificaciones.

La principal característica de esta aplicación es su escalabilidad, permitiendo el agregado de nuevos sensores de una manera sencilla.

Para el uso de la aplicación, es necesario realizar un nuevo registro, creando una cuenta única por usuario.

Por lo tanto, al iniciar la aplicación, nos encontramos con el siguiente menú principal:

Figura 49. Smart App – Pantalla de Registro (izquierda) y Login (derecha)

Una vez iniciada la sesión, se accederá a la pantalla principal -ver Figura 50-, referida al Gabinete principal y controlador del sistema, en la parte superior. En caso de no contar aún con un gabinete configurado, nos permitirá hacerlo, y en caso de haberlo hecho, podremos acceder a la visualización de los parámetros del ambiente, como así también datos del consumo energético. En relación a ello, permite acceder a una nueva pantalla donde se brindan tips de ahorro energético.

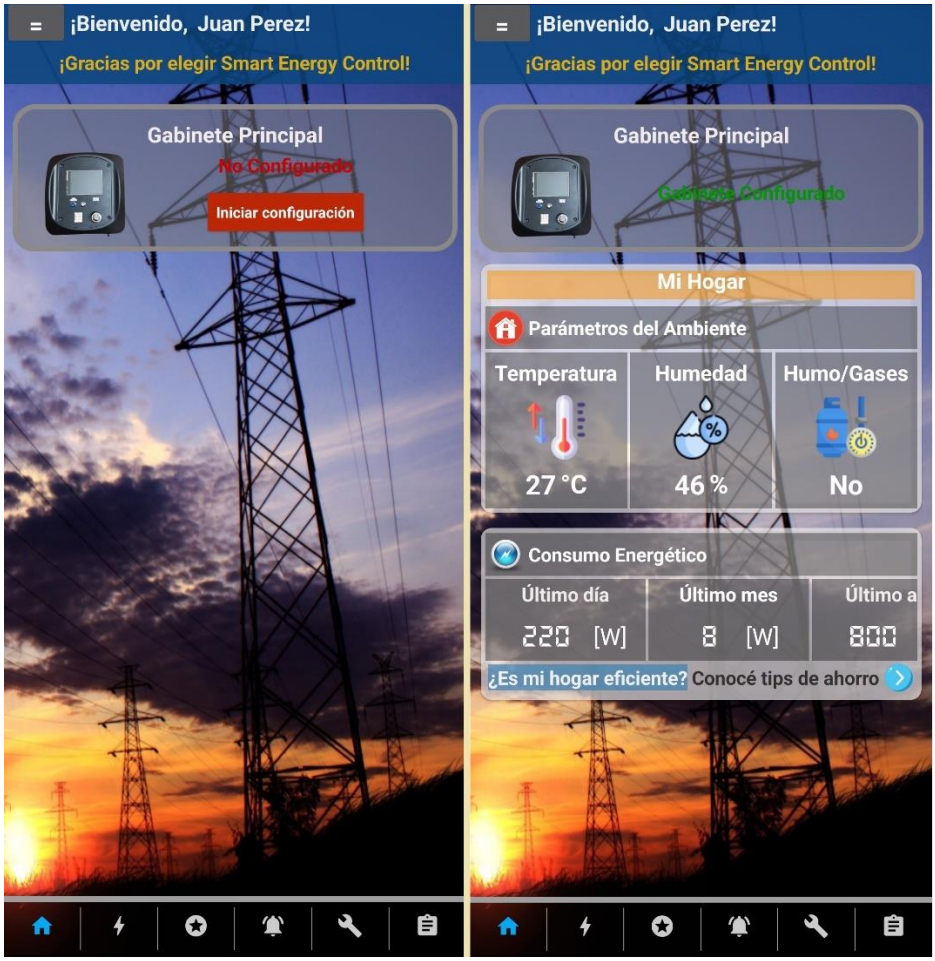

Figura 50. Smart App – Pantalla Principal

Al ingresar a la opción “Ver Mediciones”, se podrán observar tanto los parámetros de la red como del ambiente. Asimismo, desde cualquiera de ellos, se podrá acceder a las gráficas en tiempo real e historial de valores medidos.

Figura 51. Smart App – Pantalla “Mediciones” del Gabinete

En la solapa “Sensores”, se mostrará la lista con los sensores configurados. Pulsando sobre cada uno de ellos, se accederá a una nueva pantalla -ver Figura 52-, donde se observarán los parámetros medidos en tiempo real, como así también el control del interruptor integrado. En caso de haber configurado sensores, se indicará mediante mediante el texto “No hay sensores configurados”.

Esta lista es escalable, por lo que a medida que se agreguen nuevos sensores, serán visualizados uno debajo de otro.

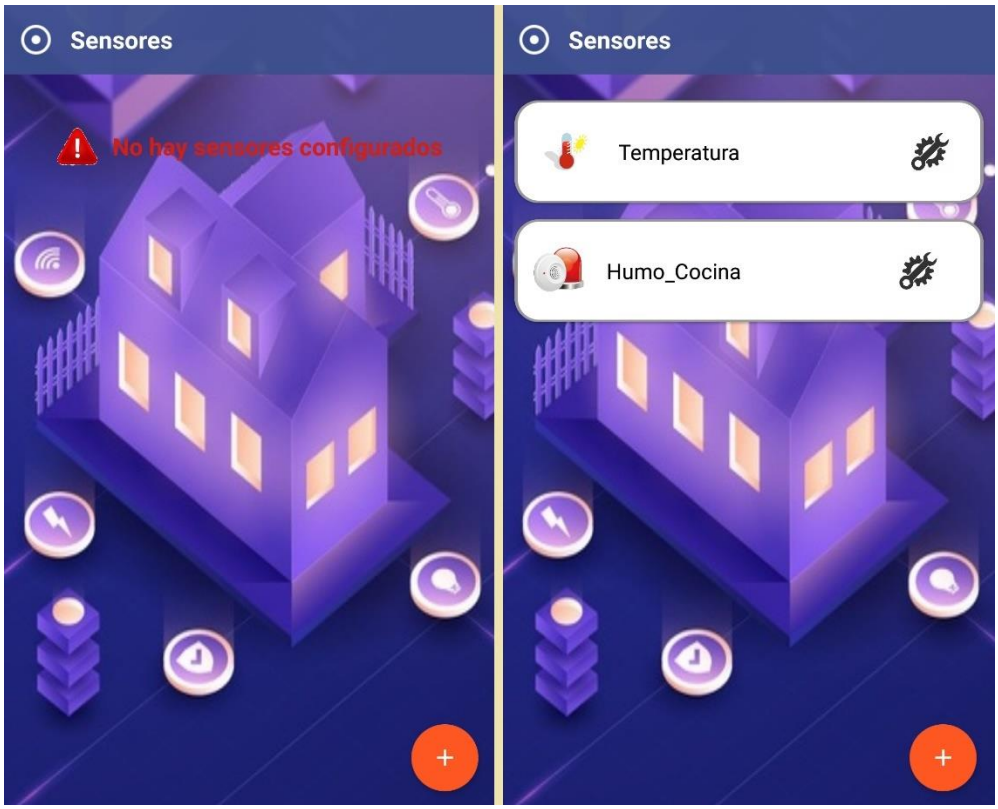

Figura 52. Smart App – Pantalla “Sensores”

Accediendo a cada uno de los sensores, se podrá acceder a su información, visualizar los parámetros medidos en tiempo real y gráficas, comandar los interruptores, entre otras opciones. En la Figura 53 se visualizan datos de un sensor de temperatura (imagen derecha) y un sensor de humo (imagen izquierda).

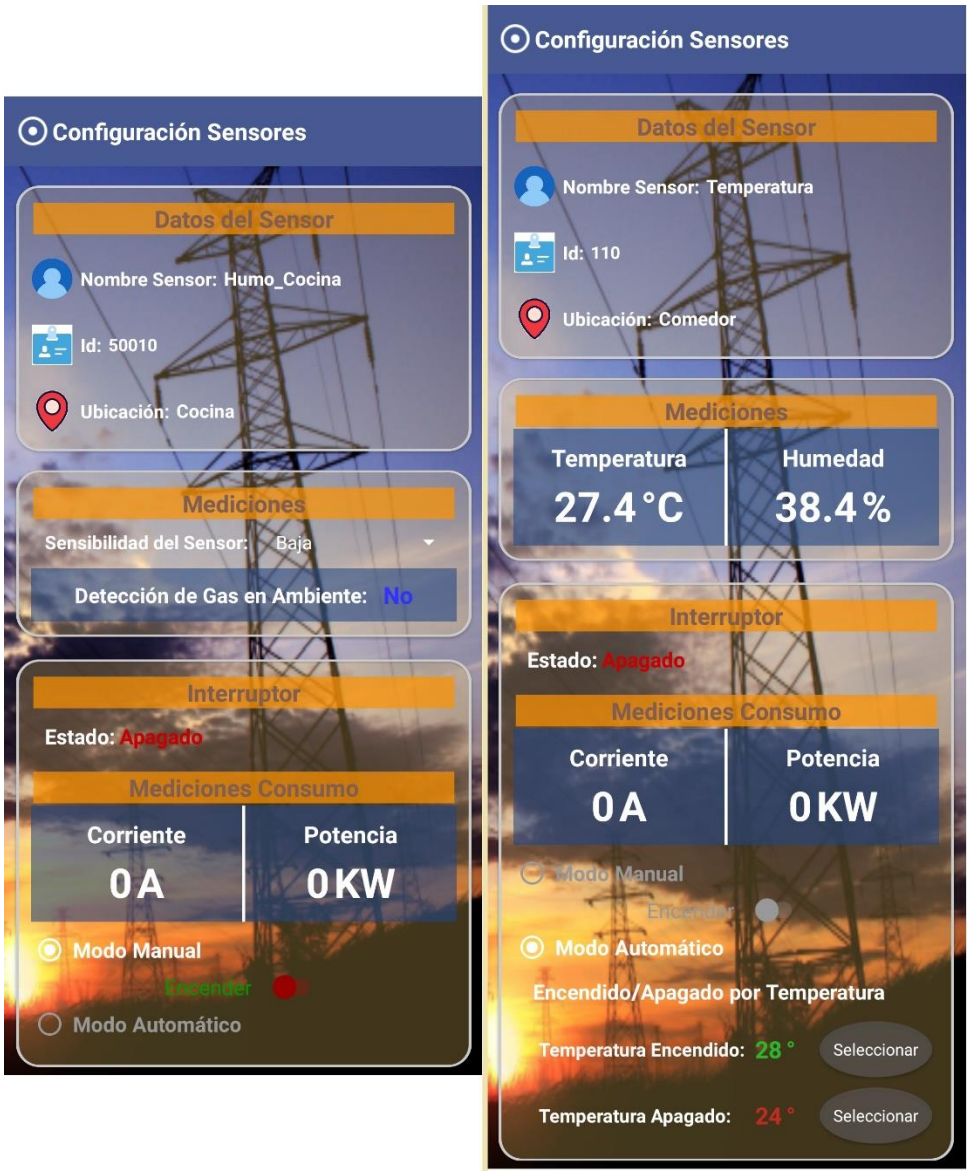

Figura 53. Smart App – Configuración de Sensores

En la solapa “Notificaciones”, se mostrará la lista de alarmas o notificaciones, entre las que se pueden mencionar:

- Detección de presencia de humo/gases, por parte de los sensores de medición.
- Encendido/Apagado de interruptores.
- Alertas de baja o alta tensión.

Cada una de estas alarmas incluirá la fecha y hora del evento, como así también el dispositivo que dio origen a las mismas.

En caso de no haber registrado notificaciones, se indicará mediante el texto “No hay sensores configurados”. Ver Figura 54.

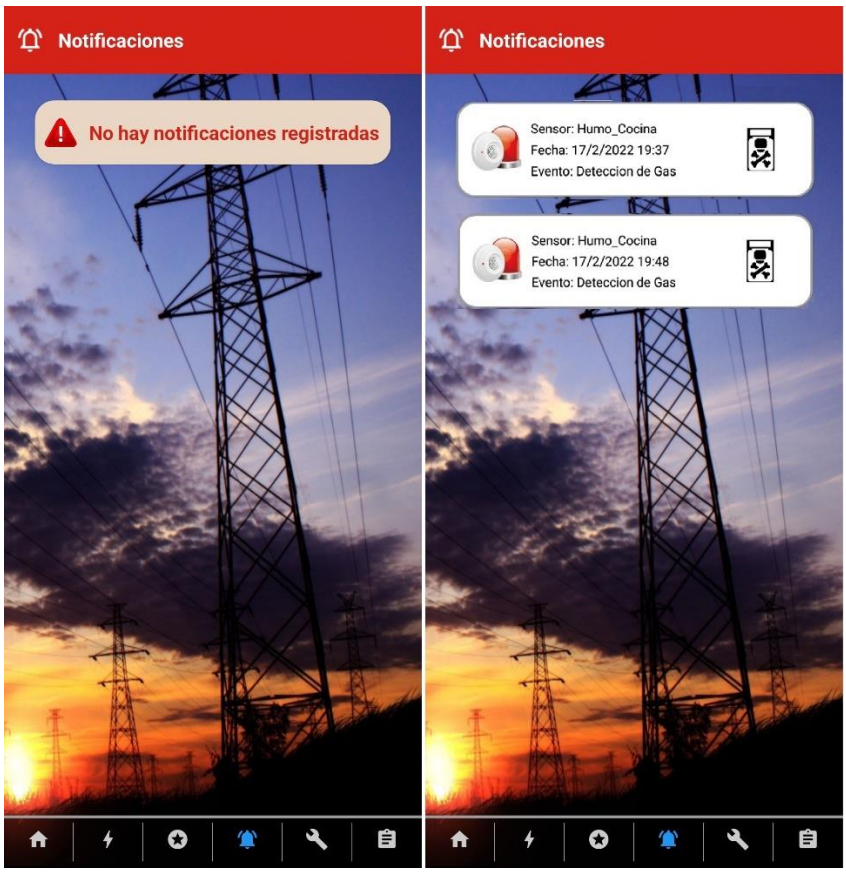

Figura 54. Smart App – Pantalla “Notificaciones”

Desde la solapa “Configuración”, se mostrarán los datos del usuario, y se permitirá la modificación del teléfono donde el usuario recibirá las notificaciones. Ver Figura 55.

Figura 55. Smart App – Pantalla “Configuración”

2.7. ENSAYOS REALIZADOS

A continuación, se presentarán las tablas y gráficos con las muestras tomadas para los distintos parámetros medidos, detallando frente que equipos patrones se contrastaron las medidas. Luego con dichas mediciones se calculará el error con la siguiente formula.

$$\%\boldsymbol{error} = \frac{Valor\ de\ referencia - Valor\ medido}{Valor\ de\ referencia} * 100$$

Siendo el valor de referencia los equipos comerciales tomados como patrón y el valor medido por el del Smart Energy Control.

2.7.1. PRUEBAS DE MEDICIÓN DE VOLTAJE AC

El objetivo de esta prueba es verificar que el equipo mide valores reales de voltaje, para esto se tomaron varias medidas de voltaje AC de la red eléctrica y se comparó con las medidas del equipo patrón. Como equipo patrón se utilizó el multímetro "True RMS UNIT-T UT39C". El multímetro muestra los valores de voltaje con una exactitud de ± 0.8% valor medido + 3 dígitos, y la resolución de nuestro sensor es de 0,1V con una exactitud de ± 0.5%.

Figura 56. Medición de voltaje AC.

Con los resultados experimentales de la siguiente tabla se verifica que el prototipo mide valores reales de voltaje (TRUE RMS).

Muestra	UNI-T UT39C [V]	PZEM-004T [V]	%Error [%]
#1	219	219,9	0,4
#2	220	220,1	0,0
#3	222	222,2	0,1
#4	221	221,9	0,4
#5	223	222,8	0,1
#6	220	220,5	0,2
#7	221	220,8	0,1
#8	223	223	0,0

Tabla XI. Mediciones de Voltaje AC

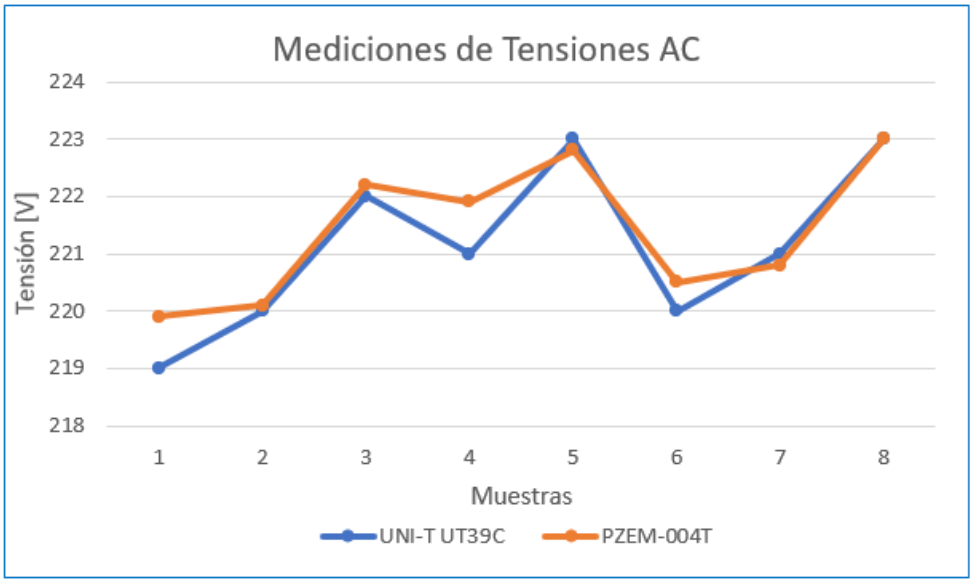

Figura 57. Gráfico de Medición de voltaje AC.

2.7.2. PRUEBAS DE MEDICIÓN DE CORRIENTE AC

Esta prueba determina la exactitud de las mediciones de corriente, para ello se tomaron 10 lecturas de corriente sometiendo al medidor a diferentes cargas (Foco, Secador y Pava eléctrica). En este caso el equipo patrón utilizado fue la pinza amperométrica YF-8020, que tiene una resolución de 0.01 A en la escala de 0 a 20 A, de 0.1 A de 20 a 200 A, y de 1 A de 200 A a 600 A con un error de ±1%.

Figura 58. Medición de corriente AC sin carga.

En la Tabla XII se presentan los resultados de esta prueba y se verifica que el equipo puede medir valores de corriente con un error absoluto máximo de 4,3 A y es sensible incluso para corrientes AC despreciables.

Med	Artefacto/Condición					YF-8020 [A]	PZEM-004T [A]	%Error [%]
#1	Sin carga					0,03	0,04	3,30
#2	Foco LED 7W					0,05	0,07	4,00
#3	Secador De Pelo	Ventilador	Mín	Calor	Mín	0,68	0,7	2,90
#4			Máx		Mín	1,3	1,27	2,30
#5			Mín		Máx	4,04	3,87	4,20
#6			Máx		Medio	4,07	3,92	3,70
#7			Máx		Máx	7,73	7,4	4,30
#8	Pava Eléctrica					8,45	8,23	2,60
#9						8,46	8,17	3,40
#10						8,53	8,34	2,20

Tabla XII. Mediciones de corriente AC

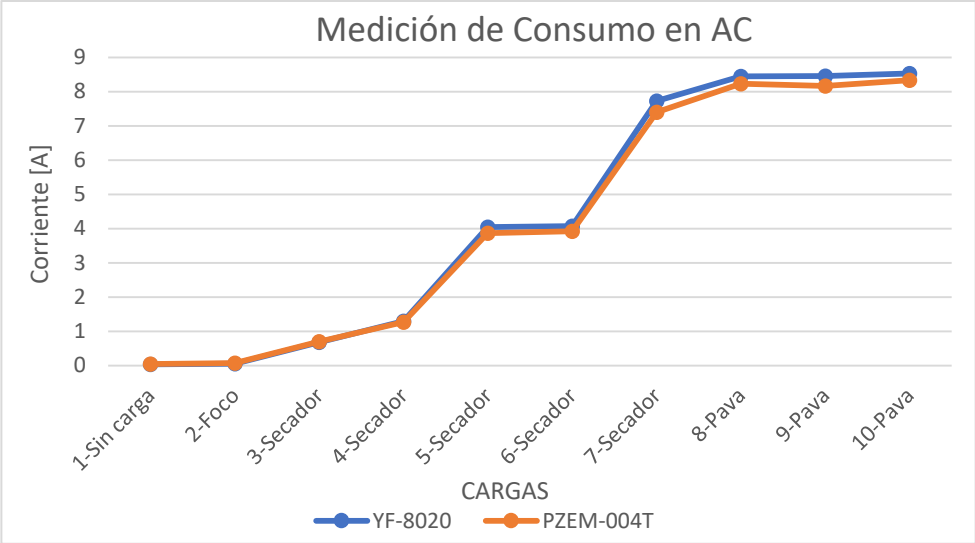

Figura 59. Gráfico de Medición de corriente AC.

Figura 60. Medición de corriente de un foco led de 7W

Figura 61. Medición de corriente de secador de 2200W

Figura 62. Medición de corriente de una pava eléctrica de 2400W

2.7.3. PRUEBAS DE MEDICIÓN DE TEMPERATURA

Por último, se realizaron pruebas de medición de temperatura. En este caso, se utilizó como instrumento patrón un anemómetro UNI-T UT363, que mide en un rango de -10°C a 50°C, con una precisión de ±2°C y resolución de 0,1°C.

Se tomaron diversas lecturas, mediante la variación de la temperatura del ambiente con un aire acondicionado. En primera instancia, se encendió el equipo en modo calor a 29°C, con una reducción progresiva de la temperatura en 1°C hasta colocarlo en modo frío a 23°C.

Figura 63. Medición de temperatura

En la Tabla XIII se presentan los resultados obtenidos de la temperatura del ambiente, observando que el equipo puede medir valores de temperatura con un error absoluto menor al 1%, lo cual lo hace muy preciso.

Muestra	Temperatura [°C]		%Error [%]
	UNI-T UT363	DHT22	
1	25,0	24,9	0,4
2	25,2	25,2	0,0
3	25,5	25,6	0,4
4	28,1	28,2	0,4
5	29,2	29,2	0,0
6	29,3	29,4	0,3
7	29,1	29,2	0,3
8	28,3	28,3	0,0
9	27,6	27,6	0,0
10	26,4	26,5	0,4
11	25,5	25,5	0,0
12	24,7	24,8	0,4
13	23,9	24,0	0,4
14	23,0	23,2	0,9
15	22,5	22,6	0,4
16	21,6	21,8	0,9
17	21,1	21,2	0,5
18	20,9	21,0	0,5

Tabla XIII. Mediciones de Temperatura

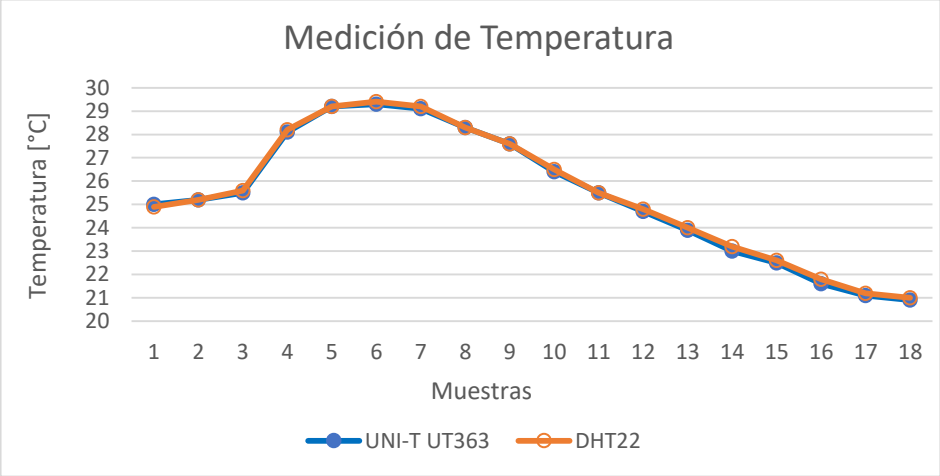

Figura 64. Gráfico de Medición de Temperatura

Figura 65. Medición de temperatura con aire acondicionado en 29°

Figura 66. Medición de temperatura con aire acondicionado en 23°

3. CONCLUSIONES

El desarrollo de este proyecto ha sido marcado por un dinamismo en cuanto a cambios implementados y puntos de mejora, teniendo en cuenta que su objetivo original era el análisis de la red eléctrica, y se logró crear, finalmente, un prototipo que integra esa medición de los parámetros eléctricos, pero basado en el concepto de "Internet de las cosas", quedando demostrada la viabilidad del desarrollo de objetos IoT para el hogar y para distintos ámbitos. Es decir, no sólo fue posible alcanzar los objetivos y especificaciones planteadas al inicio del desarrollo, sino que, resolviendo los problemas y desafíos surgidos, se pudo lograr un dispositivo totalmente funcional, pensado y diseñado para poder ser adaptado fácilmente a diversas necesidades y aplicaciones.

Analizando el prototipo final, puede afirmarse que los resultados fueron muy satisfactorios, y sin dudas superaron las expectativas planteadas al inicio. Brinda con gran precisión el consumo real al usuario, facilitándole la compresión del consumo eléctrico, y mostrando la información detallada para tomar decisiones que permitan un ahorro en el consumo de energía y, en consecuencia, un ahorro económico. Además, ofrece la facilidad de visualizar los datos obtenidos en tiempo real, tanto en los mismos dispositivos a través de su display, como en cualquier dispositivo inteligente que tenga conexión a internet.

Otra de sus principales características, y como punto a favor, es su escalabilidad: su desarrollo modular hace que sea fácilmente integrable con módulos (sensores) especializados que le otorgan nuevas funcionalidades. Por medio del uso de herramientas como códigos QR y generación de puntos de acceso, se consiguió que tanto la instalación como la puesta en marcha, y utilización del producto, sean sumamente simples para cualquier usuario.

Sin embargo, como en todo proyecto, siempre existen muchas posibilidades de realizar mejoras, que se enumeran a continuación:

- Realizar una adaptación para hacer un análisis integral de las redes trifásicas, con sensores de mayor amperaje, y sistemas de comunicación adaptados al entorno industrial, lo que convertiría a Smart Energy Control en un producto aplicable a cualquier tipo de industria. De esta manera, se podría ofrecer un servicio integral para la toma de decisiones, en lo que respecta a si el funcionamiento de la industria es eficiente, qué maquinarias renovar primero, siempre con el objetivo enfocado en lograr un uso eficiente de la energía.
- Desarrollar y utilizar un servidor web dedicado y diseñado exclusivamente para esta aplicación, evitando de esta manera que su funcionamiento dependa de servidores de terceros, los que pueden sufrir modificaciones, pérdidas de soporte, u otros inconvenientes que afecten el normal funcionamiento del sistema.

4. BIBLIOGRAFÍA

[1] ESP-32 – Datasheet
https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[2] PZEM 004t
https://manuals.plus/wp-content/sideloads/innovatorsguru-ac-communication-module-pzem-004t-v3-0-manual-optimized.pdf

[3] SCT013 100A 50mA
https://en.yhdc.com/product/SCT013-401.html

[4] TFT Display ILI9341
https://www.instructables.com/Arduino-TFT-display-and-font-library/

[5] Interfaz gráfica LVGL
https://lvgl.io/

[6] Software SquareLine Studio (v. 1.1.1)
https://squareline.io/

[7] Sensor DHT22
https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf

[8] Sensor de gas MQ-2
https://www.mouser.com/datasheet/2/321/605-00008-MQ-2-Datasheet-370464.pdf

[9] Sensor de corriente ACS712 - Datasheet
https://puntoflotante.net/ACS712-Datasheet.pdf

[10] Entorno de desarrollo integrado Arduino
https://www.arduino.cc/en/software

[11] FreeRTOS
https://www.freertos.org/

[12] Módulo GPRS A6
https://www.makerfabs.com/desfile/files/A6_A7_A6C_datasheet-EN.pdf

[13] Servidor FIREBASE
https://firebase.google.com/docs/guides

[14] Servidor THINGSPEAK
https://www.mathworks.com/help/thingspeak/index.html?s_tid=CRUX_lftnav

[15] Desarrollo de Aplicaciones - KODULAR
https://docs.kodular.io/components/google/firebase-database/