ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN PROTOTIPO DE SISTEMA *IOT* PARA LA MEDICIÓN DEL CONSUMO DE ENERGÍA ELÉCTRICA EN UNA VIVIENDA

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES

DYLAN MARTIN CHÁVEZ HERAS

DIRECTOR: LEANDRO ANTONIO PAZMIÑO ORTIZ

DMQ, FEBRERO 2024

CERTIFICACIONES

Yo, DYLAN MARTIN CHÁVEZ HERAS declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

DYLAN MARTIN CHÁVEZ HERAS

dylan.chavez@epn.edu.ec

dyllan78@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por DYLAN MARTIN CHÁVEZ HERAS, bajo mi supervisión.

LEANDRO ANTONIO PAZMIÑO ORTIZ

DIRECTOR

leandro.pazmino@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

DYLAN MARTIN CHÁVEZ HERAS

CI: 010599253-1

DEDICATORIA

Querida mamá Inés y amada novia,

A ustedes, pilares inquebrantables de mi vida, les dedico el momento en el que me encuentro con todo mi corazón. Mamá, tu apoyo constante y amor incondicional han sido mi mayor inspiración. Siempre has creído en mí, incluso cuando dudaba de mis propias capacidades. Gracias por ser mi roca y guía a lo largo de este viaje académico a pesar de tantas dificultades que hemos afrontado.

A ti, mi dulce Mildred, agradezco por tu presencia constante y el aliento que me has brindado en cada paso. Tu amor ha sido mi motivación, y tu fuerza ha sido mi refugio. Juntos hemos superado desafíos y celebrado cada triunfo, y no puedo imaginar este camino sin ti a mi lado.

Esta tesis no solo representa mi esfuerzo, sino también el apoyo inquebrantable que ambas me han proporcionado siendo las mujeres más importantes en mi vida. Con profundo agradecimiento, les dedico este logro, sabiendo que sin ustedes no habría llegado tan lejos.

Martin

AGRADECIMIENTO

Quiero expresar mi profundo agradecimiento a la Universidad Escuela Politécnica Nacional, mi alma mater, por brindarme una educación integral que ha sido fundamental en mi formación académica y personal. A lo largo de mi trayectoria, he sido testigo de la dedicación y el compromiso de la institución con la excelencia educativa, y estoy agradecido por la oportunidad de aprender y crecer aquí.

Agradezco de manera especial a mis queridos docentes, cuyo conocimiento y orientación han sido cruciales en mi desarrollo académico. Cada lección impartida ha dejado una marca significativa en mi aprendizaje, y estoy agradecido por la dedicación y pasión que han compartido conmigo.

A mi tutor, el ingeniero Leandro Pazmiño, le debo un agradecimiento especial. Su guía experta y paciencia infinita fueron fundamentales en el proceso de investigación y redacción de mi tesis a pesar de cualquier circunstancia estuvo pendiente en cada paso del proceso. Su compromiso con mi crecimiento académico ha sido inspirador, y estoy agradecido por el privilegio de haber contado con su dirección.

También quiero expresar mi gratitud a Víctor Rivera, quien desinteresadamente me brindó su apoyo y asistencia en diversos aspectos de mi tesis. Su colaboración fue invaluable, y su disposición para ayudarme hizo que este proceso fuera más llevadero.

En resumen, agradezco a mi universidad, a mis destacados profesores y, especialmente, a Leandro Pazmiño y Víctor Rivera por su contribución significativa a mi proyecto de tesis. Este logro no habría sido posible sin el respaldo de esta comunidad académica excepcional.

Martin

ÍNDICE DE CONTENIDOS

RESUMEN

El actual documento presenta la elaboración de un sistema *IoT* para la medición del consumo energético de una vivienda el cual puede ser controlado por medio de Internet utilizando una aplicación *web* y celular creada en *Arduino IoT Cloud*, el sistema está diseñado para realizar mediciones de voltaje, corriente y por medio de estas mediciones conseguir la potencia y el consumo eléctrico de un sistema monofásico y presentar todos los datos de las mediciones en las aplicaciones desarrolladas.

La primera sección del proyecto establece los objetivos a alcanzar junto con los componentes desarrollados en el proyecto, así como los conceptos fundamentales necesarios para su implementación.

La segunda sección detalla la metodología empleada para cumplir con los objetivos establecidos, lo que permite establecer la secuencia seguida en el desarrollo del proyecto.

En la tercera sección se presentan los resultados obtenidos en el logro de cada objetivo, que incluyen los requisitos del proyecto, la justificación de la selección de *hardware* y *software*, el diseño del código junto con la aplicación en *Arduino IoT Cloud*, la implementación completa del sistema en una vivienda y las pruebas que confirman el funcionamiento del prototipo

Finalmente, se presentan las conclusiones y recomendaciones derivadas del desarrollo del proyecto, así como las fuentes de investigación que fundamentaron el trabajo teórico, junto con los anexos que incluyen el video de pruebas y el código final del sistema.

PALABRAS CLAVE: consumo energético, medidor *IoT*, *Arduino Cloud*, convertidor analógico digital.

ABSTRACT

The current document presents the development of an IoT system for measuring the energy consumption of a house which can be controlled via the Internet using a web and mobile application created in Arduino IoT Cloud, the system is designed to perform measurements of voltage, current and through these measurements get the power and electricity consumption of a single-phase system and present all the measurement data in the developed applications.

The first section of the project establishes the objectives to be achieved along with the components developed in the project, as well as the fundamental concepts necessary for its implementation.

The second section details the methodology used to meet the established objectives, which allows establishing the sequence followed in the development of the project.

The third section presents the results obtained in the achievement of each objective, which include the project requirements, the justification for the selection of hardware and software, the design of the code together with the application in Arduino IoT Cloud, the complete implementation of the system in a house and the tests that confirm the operation of the prototype.

Finally, the conclusions and recommendations derived from the development of the project are presented, as well as the research sources that grounded the theoretical work, together with the annexes that include the testing video and the final code of the system.

KEYWORDS: *energy consumption, IoT meter, Arduino Cloud, analog-to-digital converter.*

DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El proyecto aborda el desarrollo de un sistema de Internet de las Cosas (*IoT*) destinado a la medición del consumo energético en viviendas, con capacidad de control remoto a través de Internet mediante una aplicación *web* y móvil creada en *Arduino IoT Cloud*. El enfoque principal se centra en la adquisición y presentación de datos relacionados con el voltaje, la corriente, la potencia y el consumo eléctrico de un sistema monofásico.

Se establecen los objetivos a alcanzar, que incluyen la creación de componentes clave para la medición y control del consumo energético en una vivienda. También se presentan los conceptos fundamentales necesarios para comprender el desarrollo del sistema.

La metodología detallada en la segunda sección proporciona una guía para cumplir con los objetivos establecidos. Esta metodología define la secuencia de pasos seguidos durante el desarrollo del proyecto, desde la selección de *hardware* y *software* hasta la implementación práctica en una vivienda.

El prototipo abarca los requisitos específicos del proyecto, la elección de ciertos componentes de hardware y software, el diseño del código necesario para la operación del sistema en *Arduino IoT Cloud*, así como la implementación completa del sistema en una vivienda real. Además, se presentan las pruebas realizadas para confirmar el correcto funcionamiento del prototipo.

El prototipo se encuentra adecuado en una vivienda para poder establecer datos reales del consumo eléctrico, la persona que pueda acceder a la aplicación será solamente la quien cuente con las credenciales de acceso con las que se creó el programa.

Para finalizar, la alimentación del prototipo se realizará con un cargador de la marca LG proporcionando el suficiente voltaje y corriente para el correcto funcionamiento del microcontrolador *ESP32*.

Objetivo general

Implementación de un prototipo de sistema *IoT* para la medición del consumo de energía eléctrica en una vivienda.

Objetivos específicos

- Identificar los requerimientos para el diseño del prototipo.

- Seleccionar el *hardware* acorde a los requerimientos establecidos.
- Diseñar el prototipo del sistema *IoT.*
- Implementar el prototipo.
- Realizar pruebas de funcionamiento del prototipo.

Alcance

Por medio del presente proyecto se busca implementar un prototipo de sistema *IoT* que permita medir el consumo de energía eléctrica en una vivienda o de cualquier dispositivo conectado al prototipo. Por tal motivo, tendrá la capacidad de realizar las siguientes acciones:

- Medir el consumo eléctrico de una vivienda o de cualquier dispositivo.
- El dispositivo de medición y comunicación inalámbrica se desarrollará empleando *hardware* libre.
- Envió de la información de consumo medida de manera inalámbrica hacia Internet.
- La información del consumo eléctrico podrá ser visualizada desde una aplicación *web* y/o desde una aplicación celular.

Marco Teórico

Sistema *IoT* para la medición de consumo energético

Un sistema *IoT* para la medición energética es una solución tecnológica avanzada que integra dispositivos electrónicos, sensores y conectividad a Internet para monitorizar y gestionar el consumo de energía en tiempo real. Este sistema permite recopilar datos precisos sobre el uso de la energía en diferentes puntos de un entorno, como hogares, edificios o instalaciones industriales. Los dispositivos de medición, como sensores de voltaje, corriente y potencia se conectan a una red *IoT*, permitiendo la transmisión de datos a una plataforma central. [1]

A través de esta plataforma, los usuarios pueden acceder a información detallada sobre el consumo energético, verificar en tiempo real posibles anomalías, y tomar decisiones para optimizar la eficiencia energética. Además, la capacidad de análisis de datos avanzada permite identificar patrones de consumo, pronosticar tendencias y mejorar la planificación de recursos. Este enfoque de medición energética *IoT* no solo facilita un

control más efectivo de los costos de energía, sino que también contribuye significativamente a la sostenibilidad al promover prácticas de consumo más conscientes y eficientes.

ESP32-WROOM

El *ESP32-WROOM* es un módulo inalámbrico altamente versátil y potente desarrollado por *Espressif Systems*. Basado en el *chip ESP32*, este módulo combina conectividad *WiFi* y *Bluetooth* de bajo consumo en un paquete compacto. El *ESP32-WROOM* ha ganado popularidad en el ámbito *IoT* debido a su capacidad para integrar funcionalidades avanzadas en dispositivos electrónicos. Equipado con un procesador de doble núcleo de 32 *bits*, el *ESP32-WROOM* como se muestra en la **Figura 0.1** ofrece un rendimiento robusto y una flexibilidad que lo hace adecuado para una amplia gama de aplicaciones, desde proyectos simples hasta soluciones industriales complejas [2].

Su soporte para diversas interfaces de *hardware*, capacidad de bajo consumo energético, y la posibilidad de ejecutar *firmware* personalizado lo convierten en una opción popular para desarrolladores que buscan una plataforma integral para proyectos *IoT*. Además, su comunidad activa y el respaldo de *Espressif* garantizan un continuo desarrollo y mejora, lo que consolida al *ESP32-WROOM* como una herramienta valiosa en el ecosistema de desarrollo *IoT*.

Figura 0.1 Placa *ESP32 [3]*

Arduino Cloud

Arduino Cloud es una plataforma en línea que amplía las capacidades de los dispositivos al ofrecer servicios en la nube para la gestión y monitorización remota. Desarrollada por *Arduino*, esta solución facilita la conexión y el control de proyectos basados en una extraordinaria cantidad de placas desde cualquier lugar con acceso a Internet. Los usuarios pueden utilizar *Arduino Cloud* para visualizar datos en tiempo real, configurar actuadores, y recibir actualizaciones sobre el estado de sus dispositivos conectados.

Además, ofrece integración con servicios *web* populares y permite la programación remota, simplificando el desarrollo y la gestión de proyectos *IoT*. *Arduino Cloud* se distingue por su enfoque amigable para los principiantes, pero al mismo tiempo ofrece herramientas avanzadas para usuarios experimentados [4].

Esta plataforma contribuye significativamente a la accesibilidad y escalabilidad de los proyectos de *hardware*, permitiendo a los desarrolladores concentrarse en la innovación y funcionalidad de sus aplicaciones.

Sensor *SCT-013*

El sensor *SCT-013* visualizado en la **Figura 0.2** es un dispositivo de medición de corriente alterna no invasivo que aprovecha el efecto *Hall* para detectar el campo magnético generado por la corriente en un conductor. Su diseño sin contacto físico con el cable conductor facilita su instalación en diversos entornos, ya que no es necesario cortar ni modificar el circuito eléctrico. Disponible en varios modelos con diferentes rangos de corriente, este sensor se utiliza comúnmente en aplicaciones domésticas, industriales y de energía. Permite una fácil integración con microcontroladores como *Arduino o ESP32*. Su versatilidad lo hace ideal para la monitorización del consumo eléctrico en el hogar, la medición de corriente en sistemas de energía renovable y el seguimiento del rendimiento de equipos eléctricos para propósitos de mantenimiento predictivo [5].

Figura 0.2 Sensor de corriente *[6]*

Sensor *ZMPT101B*

El sensor de voltaje *ZMPT101B* visualizado en la **Figura 0.3** es un dispositivo especializado diseñado para medir voltajes de corriente alterna (VCA) en un rango específico. Emplea el principio de transformación de potencial para proporcionar una salida analógica proporcional al voltaje de entrada, permitiendo su fácil integración con microcontroladores como *Arduino* o *ESP32.* Su aplicación se encuentra comúnmente en proyectos de electrónica y sistemas de monitoreo de energía, donde se requiere una medición precisa y en tiempo real de los voltajes.

Con un rango de operación de 0 a 250 (VAC), el *ZMPT101B* es versátil para aplicaciones que van desde la monitorización del consumo eléctrico en el hogar hasta proyectos industriales que requieren mediciones de calidad de energía. Al utilizar este sensor, es esencial seguir las recomendaciones del fabricante y consultar la documentación técnica para garantizar un uso seguro y preciso en los diferentes escenarios de aplicación [7].

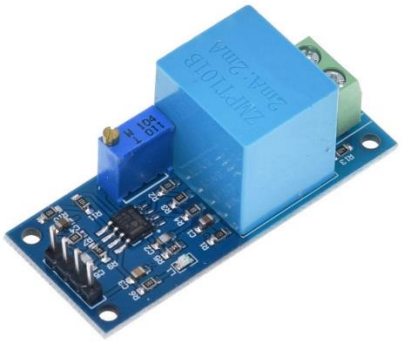

Figura 0.3 Sensor de voltaje *[8]*

ADS1115

El *ADS1115* **Figura 0.4** es un convertidor analógico a digital (ADC) altamente preciso, conocido por su capacidad para realizar conversiones de señales analógicas con una resolución de hasta 16 *bits*. Su flexibilidad se destaca al ofrecer cuatro canales de entrada diferenciales o dos canales de entrada únicos, permitiendo la medición simultánea de varias señales analógicas. El interfaz *I2C* facilita su integración con microcontroladores y otros dispositivos, además su amplificador de ganancia programable permite adaptarse a distintos niveles de señal. Este ADC es especialmente útil en proyectos de electrónica que requieren mediciones precisas, como la monitorización de sensores de temperatura, presión o voltaje [9].

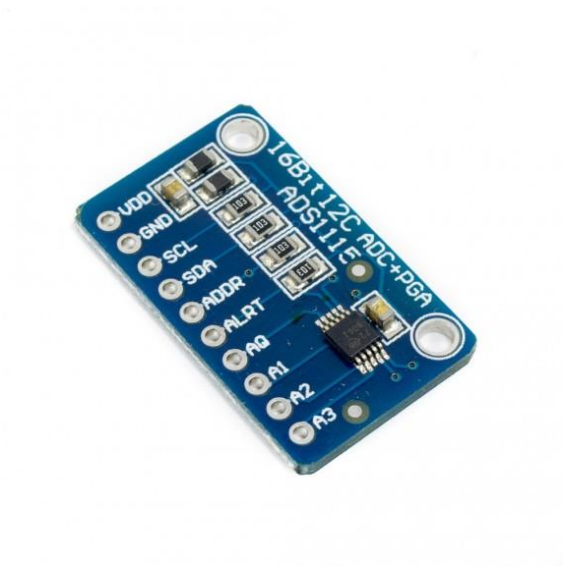

Figura 0.4 *ADS1115 [10]*

Módulo *I2C*

El módulo *Inter-Integrated Circuit (I2C)* **Figura 0.5**, es un componente esencial en la interconexión de dispositivos electrónicos en un sistema. Diseñado para facilitar la comunicación entre diferentes microcontroladores, sensores, y otros periféricos, el protocolo *I2C* utiliza un bus serial bidireccional para la transmisión eficiente de datos.

Este sistema maestro-esclavo permite la conexión de múltiples dispositivos al mismo bus, cada uno identificado por una dirección única. Un aspecto destacado del módulo *I2C* es su capacidad para simplificar las conexiones, ya que requiere solo dos líneas: una para la transmisión de datos *Serial Data (SDA)* y otra para la señal de reloj *Serial Clock (SCL)*. Esta simplicidad en el cableado lo convierte en una elección común para sistemas electrónicos donde la eficiencia en la transmisión de datos y la gestión de múltiples dispositivos son fundamentales [11].

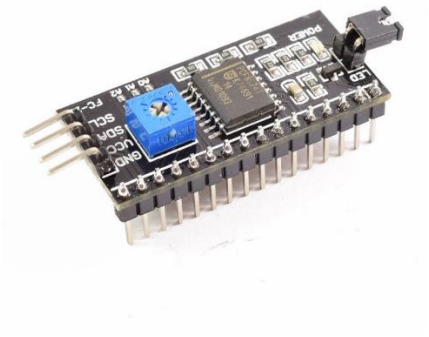

Figura 0.5 I2C *[11]*

METODOLOGÍA

Para el cumplimiento del desarrollo del proyecto de un sistema *IoT* que permita medir el consumo energético de una vivienda, se implementó una metodología basada en 5 pasos. **Figura 0.1**.

Figura 0.1 Metodología del medidor

Como se muestra en la imagen a través de una investigación inicial, se determinaron los requisitos esenciales para la elaboración del prototipo, con el fin de asegurar el cumplimiento de los parámetros definidos dentro de los objetivos, y así, establecer las principales especificaciones que debe satisfacer el dispositivo de medición

Tras la identificación de los requisitos pertinentes, se llevó a cabo un análisis para definir las características tanto de *hardware* con del *software* necesarios para realizar el medidor de consumo de energía eléctrica. En la sección dedicada a la selección del hardware, se evaluaron las propiedades de diversos microcontroladores para respaldar la elección del módulo *ESP32*, así como otros componentes físicos, como sensores, seleccionados en función de los requisitos establecidos para el prototipo. Respecto a la selección del *software*, se analizaron las características de las plataformas existentes que permiten el desarrollo de códigos abiertos compatibles con los microcontroladores.

Para la sección del diseño de prototipo se compararon distintas plataformas para desarrollar el programa, siendo elegida la plataforma de *Arduino IoT Cloud*. Luego se realizaron esquemas para adecuar el circuito electrónico y definir los elementos

necesarios para el circuito. Además, se elaboraron los códigos de los programas que deberán ser implementados en el microcontrolador.

Una vez completada las anteriores secciones, se procedió a la implementación del sistema de medición, ensamblando las partes desarrolladas y montándolas en una caja diseñada en 3D. Se realizaron pruebas de funcionamiento para validar su desempeño, utilizando los resultados para corregir posibles errores y garantizar que el prototipo cumpla con todos los requisitos establecidos.

RESULTADOS

En la siguiente sección se proporcionan las especificaciones detalladas sobre los criterios para la identificación de requisitos, elección de *software* y *hardware*, elaboración del diseño, implementación del prototipo y realización de pruebas de funcionamiento. El propósito es alcanzar los objetivos establecidos en el plan de titulación.

3.1 Identificación de los requerimientos para la implementación del prototipo

En base a una investigación se reconocieron los siguientes requerimientos para implementar el prototipo del sistema *IoT* que permita la medición de voltaje y corriente para de esta manera conseguir el consumo energético dentro de una vivienda usando *hardware* libre, además de usar una aplicación *web* y celular para visualizar datos relacionados con el consumo energético.

Requerimiento de alimentación

El requerimiento de alimentación para el sistema de medición de energía eléctrica es fundamental para asegurar su funcionamiento ininterrumpido y la recopilación precisa de datos durante períodos prolongados. El sistema debe mantenerse encendido constantemente para garantizar la captura continua de información relevante sobre el consumo de energía. Para cumplir con este requisito, se ha planteado el uso de un cargador, como la fuente de alimentación principal debido a su capacidad para proporcionar una tensión constante y estable.

El cargador proporcionará la energía necesaria para alimentar todos los componentes del sistema de medición, incluyendo sensores, microcontroladores y módulos de comunicación, asegurando así su funcionamiento. La estabilidad de la tensión de salida del cargador es crucial para evitar fluctuaciones que puedan afectar la precisión de las

mediciones de energía eléctrica. Se enfatiza la importancia de elegir un cargador confiable y de alta calidad que cumpla con los estándares de seguridad eléctrica pertinentes. Esto garantizará la integridad del sistema de medición y reducirá el riesgo de fallos o daños durante su operación continua.

Requerimiento de sistema monofásico

Se establece la necesidad de capturar y analizar el consumo de energía en un entorno de suministro de electricidad de una sola fase. Este requisito implica que el sistema de medición debe ser diseñado específicamente para trabajar con una fase de alimentación, lo que implica la utilización de sensores y dispositivos de medición adecuados para monitorear la corriente y el voltaje en esta configuración monofásica.

La adaptación del sistema de medición para operar en un entorno monofásico también requiere considerar las características específicas de este tipo de sistema eléctrico, como la carga que puede presentarse en algunos casos.

Requerimientos de plataforma *web*

Para satisfacer este requerimiento se necesita de una plataforma *web* capaz de recibir y presentar los datos de consumo de manera accesible. La interfaz *web* debe ser intuitiva, permitiendo a los usuarios visualizar el consumo energético en tiempo real. Además, se necesita que dicha plataforma tenga compatibilidad con los distintos navegadores utilizados por los usuarios a quienes se encuentra dirigido el sistema *IoT* para la medición del consumo energético, así también esta plataforma *web* debe contar con soporte para la placa de desarrollo seleccionada.

Requerimientos de Aplicación Móvil

El uso de una aplicación móvil se vuelve esencial para brindar a los usuarios la capacidad de acceder a los datos de consumo desde cualquier lugar. La aplicación debe ser compatible con sistemas operativos populares como *Android* y *iOS*, proporcionando una experiencia de usuario intuitiva y funciones sencillas que no aturdan al usuario, presentando datos esenciales como voltaje, corriente, potencia y consumo energético.

Implementación basada en *hardware* libre

Se debe priorizar la selección de componentes y dispositivos que respeten estándares abiertos y permitan una integración flexible. En términos de sensores, se podrá optar por dispositivos de medición de corriente y voltaje de código abierto, facilitando así la interoperabilidad y la adaptación a diferentes entornos eléctricos. Además, la elección

de microcontroladores y plataformas de desarrollo de *hardware* libre, como *Arduino* entre otros, proporcionan una base sólida para la creación de proyectos eficientes y personalizables para la gestión de datos, creación de entornos *IoT* y comunicación entre dispositivos asegurando la transparencia y la accesibilidad, aspectos fundamentales en el desarrollo de soluciones sostenibles y colaborativas en el ámbito de la medición del consumo eléctrico mediante *IoT* para una vivienda o cualquier dispositivo [12].

Selección de sensores para el sistema *IoT*

El sistema *IoT* requiere el uso de sensores los cuales permitirán obtener los datos del voltaje y la corriente que pasa por la vivienda o de algún otro dispositivo eléctrico, de esta manera se podrá asegurar la obtención de mediciones acertadas tanto de voltaje como de corriente para obtener la potencia y de esta manera asegurar la medición del consumo energético, entre las características que se buscan es que dichos sensores se encuentren diseñados para soportar tensiones y corrientes comunes dentro de un hogar, además que estos sean compatibles con los demás equipos necesarios para ser plenamente configurados y ser funcionales dentro del prototipo, por último los precios de los mismos deben ser relativamente bajos por lo cual se buscará distintos modelos y marcas que se encentren disponibles.

Implementación del sistema *IoT*

Se establece la necesidad de desarrollar un plan detallado que guíe la implementación del prototipo. Este plan debe abordar aspectos técnicos, cronogramas y asignación de recursos, proporcionando una estructura sólida para la ejecución del proyecto. Además, es muy importante realizar pruebas de integración del *hardware* y del *software*, para verificar la compatibilidad y el rendimiento conjunto de los componentes, identificando posibles conflictos o deficiencias antes de la implementación completa. Por último, se establece la obligación de garantizar la estabilidad y confiabilidad del sistema durante todo el proceso de implementación. En conjunto, estos requerimientos forman la base esencial para una implementación efectiva y exitosa del prototipo, asegurando la calidad y la funcionalidad del sistema *IoT*.

3.2 Determinación del *hardware* y *software* requeridos

Deterrminación de *hardware*

Placa de Desarrollo

Para la presente elección se toma en cuenta la existencia de múltiples placas utilizadas para el desarrollo de proyectos de electrónica.

La placa de desarrollo *ESP32* es una gran herramienta que facilita la creación de proyectos de domótica y cuenta con una gran cantidad de características por lo cual se ve superior frente a otras placas de desarrollo como puede ser *Arduino UNO*, a continuación, se muestra la **¡Error! No se encuentra el origen de la referencia.** donde se detallan sus características comparadas con la placa *Arduino UNO*.

Tabla 0.1 Tabla comparativa de placas de desarrollo

Característica	Arduino Uno	ESP32 WROOM
Microcontrolador	ATmega328P	Xtensa Dual-Core 32-bit LX6
Frecuencia de reloj	16 (MHz)	160 o 240 (MHz)
Núcleos	1	2
Conectividad inalámbrica	No	Wi-Fi, Bluetooth LE
Memoria Flash	32 (KB)	4 (MB) (puede variar según el módulo)
Memoria RAM	2 (KB)	520 (KB)
Pines de entrada/salida digital	14	36
Pines de entrada analógica	6	18
Pines	6	16
Interfaces de comunicación	UART, SPI, I2C	UART, SPI, I2C, I2S, CAN, etc.
Alimentación	5 (V)	3.3-5 (V)
Consumo de energía	Mayor	Menor
Compatibilidad con *shields*	Sí	No
Comunidad y soporte	Amplia	Amplia

La elección de la placa *ESP32* sobre la *Arduino Uno* se justifica por la notable expansión de capacidades que ofrece la *ESP32* en términos de rendimiento y funcionalidad. Equipada con un microcontrolador de doble núcleo de 32 *bits*, la placa de desarrollo *ESP32* supera significativamente la velocidad de procesamiento y la capacidad de memoria tanto *RAM* como *Flash* en comparación con el *ATmega328P* presente en la *Arduino Uno*. Además, *ESP32* incorpora conectividad *Wi-Fi* y *Bluetooth* de forma nativa, abriendo la puerta a una amplia gama de aplicaciones *IoT.* Con más pines de entrada y salida, capacidades avanzadas de comunicación, y opciones de desarrollo versátiles.

ESP32 ofrece a los diseñadores y desarrolladores un entorno más potente y flexible para llevar a cabo proyectos complejos. La elección de la placa *ESP32* se convierte así en la opción preferida cuando se buscan características avanzadas y una mayor capacidad de integración de tecnologías inalámbricas.

Sensor de voltaje

Para la medición de voltaje en este proyecto se ha elegido el sensor *ZMPT101B* **Figura 0.3** en base a diversas razones técnicas y funcionales; como lo son la precisión y fiabilidad**,** el sensor *ZMPT101B* es conocido por su precisión en la medición de voltajes, lo cual es esencial para obtener datos confiables en el contexto de este trabajo. La fiabilidad del sensor es crucial para asegurar la integridad de los resultados obtenidos y la validez de las conclusiones derivadas de ellos.

Otra razón es que el sensor *ZMPT101B* tiene un rango de medición que va hasta los 250 (VAC) lo cual se adapta perfectamente a los requisitos domésticos de este proyecto, en el caso de llevarlo a nivel industrial u otro ambiente se deberá considerar un sensor con más capacidad para realizar las mediciones.

La facilidad con la que el sensor se puede integrar con otros componentes electrónicos y su compatibilidad con la mayoría de las placas de desarrollo es otra más de las razones por las cuales se usó dicho sensor. Además de contar con una amplia documentación, lo cual facilita en gran medida el uso adecuado, por último, este ofrece un equilibrio adecuado entre costo y funcionalidad. Considerando las limitaciones presupuestarias y la necesidad de obtener resultados de alta calidad, la elección de este sensor es económica sin comprometer la calidad de las mediciones.

Sensor de corriente

Para la medición de corriente se optó por el sensor *SCT-013* **Figura 0.2** el cual está diseñado específicamente para medir corrientes de hasta 100 (A) de manera precisa lo que lo vuelve ideal para censar las corrientes que se pueden llegar a tener a nivel doméstico. Este sensor proporciona una salida de voltaje proporcional a la corriente medida. Esta característica simplifica la interfaz con sistemas de adquisición de datos y facilita la conversión de la señal analógica en datos interpretables, además este cuenta con una amplia documentación la cual facilita su calibración y uso.

En conclusión, sus características técnicas específicas, su capacidad para cumplir con los requisitos del proyecto, su facilidad de uso, su seguridad en aplicaciones no invasivas, y su amplia disponibilidad en el mercado lo vuelven la mejor opción para el desarrollo del proyecto.

Pantalla *LCD*

Para presentar de una manera más adecuada los datos del consumo sin tener que acceder a las aplicaciones para el medidor se implementó una pantalla *LCD* con una resolución de 16x2 como se muestra en la **Figura 0.1** lo que se ajusta de manera correcta a los parámetros que se van a visualizar en el proyecto, esta funciona a un voltaje de 5 (V) suministrados por la placa *ESP32*, además de contar con microcontrolador que usa el protocolo de comunicación *I2C* el cual ayuda en gran medida a reducir el número de conexiones que se deben realizar , además cabe mencionar que dicho controlador cuenta con un potenciómetro integrado el cual ayuda a controlar el brillo de la pantalla para ajustar las preferencias del usuario de una manera muy sencilla.

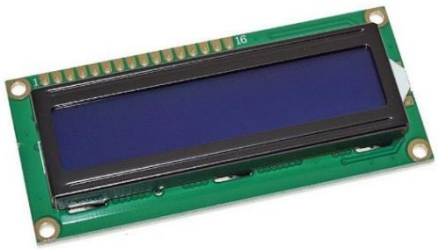

Figura 0.1 Pantalla *LCD [13]*

Determinación de *software*

Arduino IDE es una herramienta de desarrollo integral que permite a los usuarios escribir, compilar y cargar código en placas *Arduino* de manera local. Ofrece una interfaz de usuario familiar y una experiencia de desarrollo directa. Los usuarios tienen control total sobre su entorno de desarrollo y pueden acceder a una amplia variedad de bibliotecas y recursos. Sin embargo, la dependencia de la instalación local puede ser un inconveniente para aquellos que buscan una solución más accesible y colaborativa [14].

Arduino Cloud, por otro lado, representa una solución basada en la nube que simplifica el proceso de desarrollo y permite a los usuarios programar y monitorear sus dispositivos *Arduino* a través de una interfaz en línea. Esta plataforma ofrece la ventaja de la accesibilidad desde cualquier lugar con conexión a Internet, eliminando la necesidad de instalaciones locales y facilitando la colaboración en proyectos. Además, *Arduino IoT*

Cloud integra herramientas de administración de dispositivos y servicios en línea que pueden mejorar la eficiencia y la experiencia de desarrollo [4].

En conclusión, aunque el IDE de *Arduino* es una opción robusta para el desarrollo local, *Arduino Cloud* destaca como una solución más moderna y accesible, especialmente para proyectos colaborativos y para aquellos que valoran la facilidad de uso y la gestión simplificada de dispositivos en un entorno en línea.

1.3 Diseño del prototipo de medición de consumo energético

Elección de la plataforma

Para la elección de la plataforma en la cual se implementará el proyecto se tuvo en cuenta las características más importantes, que ayuden al cumplimiento de los requerimientos del proyecto. Partiendo de este punto se toma en cuenta 2 plataformas de desarrollo en la nube como lo son *Blynk* y *Arduino Cloud.*

Blynk **Figura 0.2** es una plataforma de desarrollo *IoT* que ha ganado popularidad por su enfoque intuitivo y su capacidad para simplificar la creación de aplicaciones para Internet de las Cosas. Este cuenta con una interfaz gráfica amigable y fácil de usar, la plataforma utiliza un sistema de *widgets* que permite a los usuarios diseñar interfaces gráficas personalizadas para controlar y monitorear dispositivos conectados. Esta característica simplifica el desarrollo de aplicaciones [15].

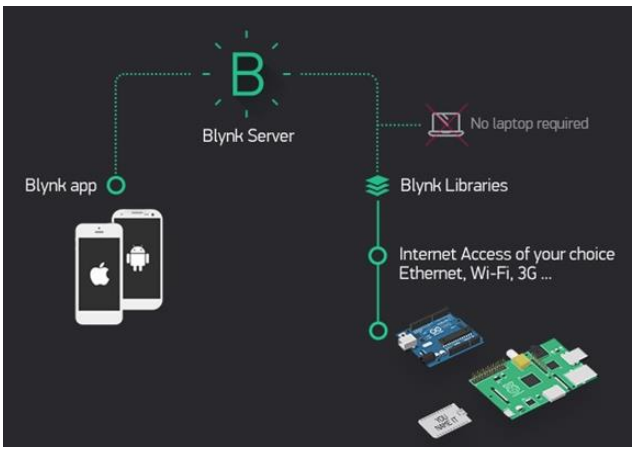

Figura 0.2 Plataforma *Blynk [15]*

Otra característica muy importante sobre esta plataforma es su capacidad para trabajar con una variedad de dispositivos, no limitándose exclusivamente a *hardware* de *Arduino*. Ofrece soporte para plataformas populares como *Raspberry Pi*, *NodeMCU,* y otros microcontroladores y placas de desarrollo. Esta versatilidad amplía las posibilidades de aplicación y permite a los usuarios seleccionar la mejor opción para sus proyectos.

Arduino Cloud **Figura 0.3** es una plataforma integral diseñada para simplificar el desarrollo y la gestión de proyectos basados en distintas plataformas como lo es *Arduino* ya que todas las placas se encuentran de manera nativa en esta herramienta; así como también otras muy populares y utilizadas por desarrolladores como lo son *ESP32*, *Raspberry PI* entre otras. Sus características destacadas son fundamentales para facilitar la programación, monitoreo y control de dispositivos *IoT* [4].

La plataforma se integra de manera fluida con otros servicios en línea de *Arduino*, como *Arduino IoT Cloud API*. Esta integración permite acceder a funcionalidades adicionales y a servicios de nube complementarios, mejorando la versatilidad y potencial de los proyectos desarrollados en *Arduino Cloud*. La plataforma es altamente escalable, permitiendo a los usuarios expandir sus proyectos sin problemas a medida que crecen. Además, se presta especial atención a la seguridad de los datos y la privacidad, asegurando la protección de la información transmitida y almacenada en la nube.

Arduino Cloud se destaca por su interfaz de usuario intuitiva y fácil de usar. Para aquellos que están comenzando con el desarrollo de proyectos *IoT* o incluso no cuentan con experiencia en el campo de la programación *Arduino Cloud* se vuelve una de las mejores opciones al momento de crear un proyecto *IoT*, además de brindar soporte dentro de su comunidad el uso y la aplicación de las distintas bibliotecas se vuelve algo sumamente fácil para cualquiera que se encuentre empezando en el mundo de la programación.

Figura 0.3 *Arduino IoT Cloud [4]*

En resumen, *Arduino Cloud* destaca como una solución integral para el desarrollo de proyectos *IoT* basados en *hardware* libre. Su capacidad de programación remota, gestión centralizada, integración con servicios en línea, su enfoque en la facilidad de uso y la gran cantidad de información que existe en cuanto a su manejo por parte de la comunidad y la misma plataforma hacen que sea la mejor opción. Por lo tanto, a partir

de la información planteada se ha decidido optar por *Arduino IoT Cloud* para la implementación del proyecto. Además, que esta plataforma si permite la integración de una aplicación celular al igual que la *web* sin tener la necesidad de volver hacer la interfaz, por otro lado, *Blynk* no cuenta con un editor *web* para la generación y edición del código que facilite su manejo; dándole a *Arduino IoT Cloud* aún más ventaja sobre *Blynk* y siendo esto fundamental para el cumplimiento de los requerimientos del proyecto.

Esquema general del proyecto

Como se observa en la **Figura 0.4** se muestra el esquema general del proyecto, en el cual se presenta el funcionamiento a grandes rasgos el funcionamiento del sistema *IoT* de medición del consumo energético. El sistema se encuentra desarrollado sobre la placa conocida como *ESP WROOM 32*, la cual actúa como agente central de todo el prototipo, este mismo recibirá los datos obtenidos de los sensores de corriente y voltaje para luego calcular la potencia y el consumo energético.

La placa de desarrollo mantendrá conexión a internet mediante *Wi-Fi* con lo cual se establecerá una comunicación con la plataforma de *Arduino IoT Cloud,* para poder visualizar los datos en la aplicación *web* y celular en tiempo real. Para finalizar todos los componentes estarán adecuados en una placa de circuito impreso *(PCB)* que a su vez se encontraran dentro de una caja impresa en 3D para su mejor manipulación y protección, dicha caja contará con conexiones externas para los dos sensores y una salida para la fuente de alimentación, además de presentar los datos necesarios en la pantalla *LCD* [2].

Al ingresar a la aplicación *web* o celular el usuario podrá iniciar o detener la medición realizada por el prototipo, además de observar todos los datos presentados en la pantalla *LCD*.

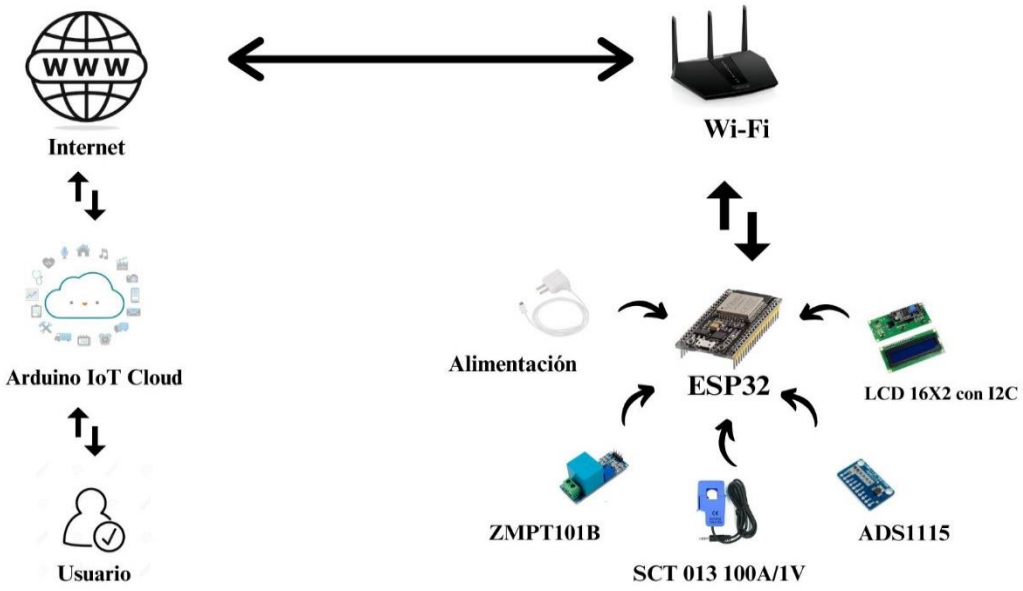

Figura 0.4 Esquema General del Proyecto

Desarrollo del código en el editor *web* de *Arduino IoT Cloud*

Para el desarrollo del código fuente del proyecto se tomó en consideración el circuito creado previamente en la herramienta de *software* llamada *Proteus,* como se muestra en la **Figura 0.5** a continuación, utilizando la placa de desarrollo *ESP32*.

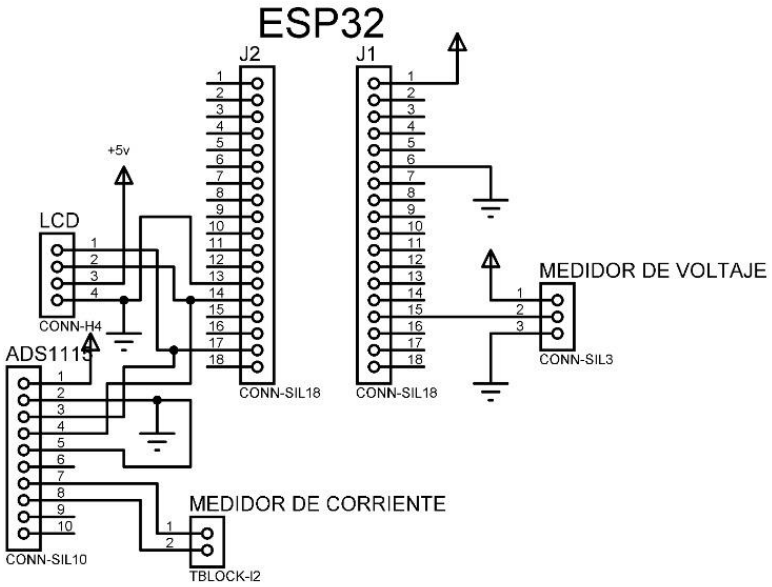

Figura 0.5 Circuito del Medidor *IoT*

Como se observa en el diseño del circuito se cuenta con una alimentación de 5 (V) y se tiene una conexión a tierra las cuales son necesarias para el correcto funcionamiento del *ESP32*; para el medidor de corriente se implementó una conexión de 2 pines los cuales van conectados al *ADS1115* para el envío de la información del sensor. El sensor de voltaje cuenta con 3 pines de los cuales 2 de estos serán utilizados para la alimentación del mismo, dejando 1 pin analógico para el envío de información hacia el *ESP32*.

El *ADS1115* fue representado utilizando 10 pines, de los cuales 2 fueron utilizados para la conexión a tierra y la alimentación, 2 pines específicamente el 3 y el 4 fueron usados para el *SCL* y el *SDA* los cuales están representados en los pines 17 y 14 del *ESP32* del circuito presentado, por último, los pines 7 y 8 se encuentran conectados al sensor de corriente.

Para el *LCD* que cuenta con el módulo *I2C* se colocaron 4 pines de los cuales 2 fueron destinados a la alimentación y a tierra, dejando los pines 1 y 2 para el *Serial Clock* y el *Serial Data* que van conectados al pin 17 y 14 en el circuito que representa la placa de desarrollo *ESP32*.

A partir del planteamiento del circuito desarrollado en *Proteus* se elaboró el código fuente del prototipo utilizando el editor *web* de *Arduino IoT Cloud*, para lo cual fue necesario la creación de una cuenta, para ello se debe contar con un correo electrónico activo o caso contrario poder vincular a *Facebook* o *Google* como se muestra en la **Figura 0.6**.

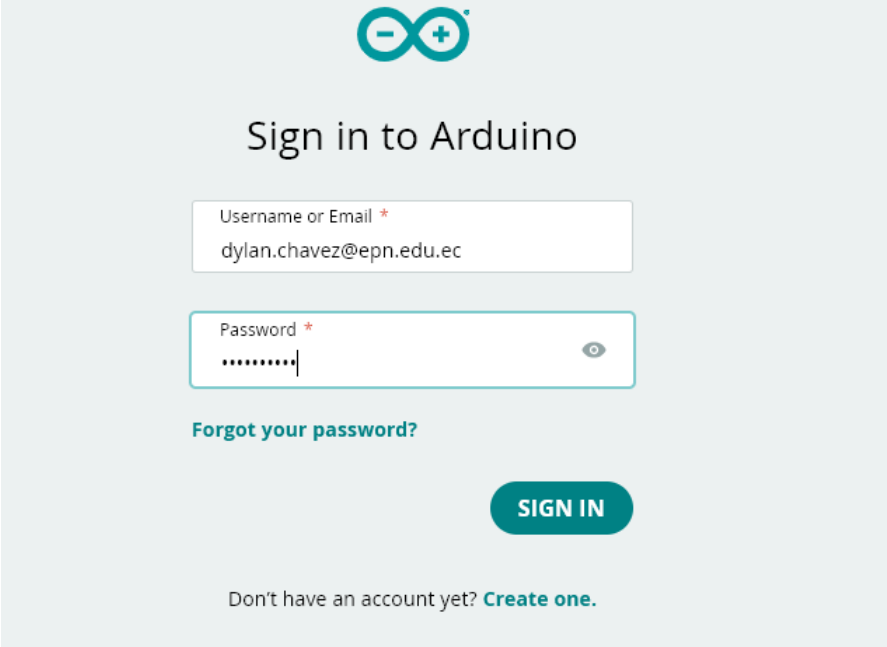

Figura 0.6 Inicio en la Plataforma de *Arduino Cloud*

Al ingresar con la cuenta se podrá acceder al apartado *Arduino Cloud en* el cual se deberá ingresar las variables que se vayan a utilizar dentro del código fuente, para el proyecto se utilizó 5 variables, que son el máximo que permite utilizar la plataforma sin pagar como se muestra en la siguiente **Figura 0.7**.

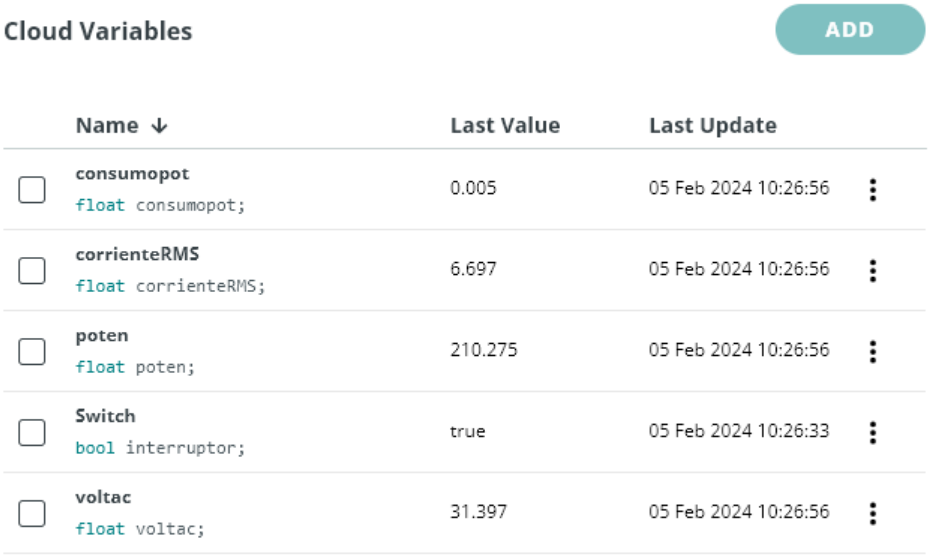

Figura 0.7 Variables de la Nube

La variable **interruptor** de tipo booleana permitirá acceder al lazo para ejecutar las funciones del código, esta se encuentra representada por un *Switch* virtual en el cual al presentar *HIGH* o 1 ingresará al lazo de lo contrario saldrá del lazo y mostrará un *Banner* indicando al usuario que inicie la medición y no mostrará valor alguno de la aplicación.

La variable **corrienteRMS** presentará un número flotante que indicará el valor de corriente medido por el sensor de corriente *sct013*, la variable flotante de **voltac** indicará la medición de voltaje obtenida del sensor de voltaje *zmtp101b*; la variable flotante **poten** indicará el valor de la potencia medido a través de la multiplicación del voltaje y la corriente medidos por los sensores. La variable **consumopot** presentará el consumo energético medido en Kilovatios hora (KWh).

Las librerías empleadas para el desarrollo del prototipo se muestran a continuación en la **Figura *0.8***.

La librería thingProperties.h es propia de la plataforma de *Arduino* esta ayudará integrando las variables anteriormente mencionadas al código así mismo con la conexión a la plataforma, la librería Wire.h es un estándar de *Arduino* utilizada para la comunicación con dispositivos que admiten *I2C*, como el módulo *Liquid Crystal Display (LCD)*, la librería LiquidCrystal_I2C.h utilizará la dirección de la pantalla *LCD* y mediante el microcontrolador *I2C* presentará los valores requeridos en la pantalla, la librería ZMPT101B.h es exclusivamente para el uso del sensor de voltaje con el mismo nombre esta proporciona funciones para realizar mediciones de voltaje, la librería Adafruit_ADS1X15.h sirve para interactuar con convertidores analógico a digital (ADC) de la serie *ADS1115* de *Adafruit*.

En este código, se utiliza para leer la entrada analógica proveniente del sensor de corriente.

```
#include "thingProperties.h"
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <ZMPT101B.h>
#include <Adafruit_ADS1X15.h>
```

Figura 0.8 Librerías

La librería thingProperties.h utiliza los datos previamente proporcionados en la sección de *Setup* como lo son el nombre de la red *Wi-Fi* a la que se va a conectar la placa y la

contraseña de la misma, además de proporcionar medidas de seguridad como lo es una clave secreta para el servicio que se haya asociado, mediante esta librería *Arduino Cloud* agrega las variables y realiza la conexión a la plataforma, todo esto de manera automática.

```
// Code generated by Arduino IoT Cloud, DO NOT EDIT.

#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>

const char DEVICE_LOGIN_NAME[]  = "813a4aa2-665d-4a22-9a6a-405a4ad9709e";

const char SSID[]               = SECRET_SSID;    // Network SSID (name)
const char PASS[]               = SECRET_OPTIONAL_PASS;    // Network password (use for WPA, or use as key for WEP)
const char DEVICE_KEY[]  = SECRET_DEVICE_KEY;    // Secret device password

void onConsumopotChange();
void onCorrienteRMSChange();
void onPotenChange();
void onVoltacChange();
void onInterruptorChange();

float consumopot;
float corrienteRMS;
float poten;
float voltac;
bool interruptor;

void initProperties(){

  ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
  ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
  ArduinoCloud.addProperty(consumopot, READWRITE, ON_CHANGE, onConsumopotChange);
  ArduinoCloud.addProperty(corrienteRMS, READWRITE, ON_CHANGE, onCorrienteRMSChange);
  ArduinoCloud.addProperty(poten, READWRITE, ON_CHANGE, onPotenChange);
  ArduinoCloud.addProperty(voltac, READWRITE, ON_CHANGE, onVoltacChange);
  ArduinoCloud.addProperty(interruptor, READWRITE, ON_CHANGE, onInterruptorChange);

}

WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```

Figura 0.9 Librería ***thingProperties.h***

Se definen las variables como la de sensibilidad del sensor de voltaje obtenido previamente de la calibración del mismo usando los repositorios de la librería ZMPT101B.h en el cual se proporciona el código para realizar dicha acción usando una placa de *Arduino UNO*, además de otras variables usadas en el código para el cálculo de corriente, voltaje, potencia y consumo.

```
#define SENSITIVITY 543.750f    // sensibilidad del sensor de voltaje obtenido de la calibracion del
#define vsensor 34              // pin analogico para el sensor de voltaje

const float factor = 100;// factor del sensor de corriente
const float multiplier = 0.0309F;//factor para el calculo de la corriente
unsigned long previousMillis = 0;
const long interval = 1000; //constante utilizada para el calculo del consumo
float sumapot = 0; //inicio dela variable que guarda el consumo cada segundo
long dt = 0;//inicio de lavariable que guarda la diferencia de tiempo desde la ultima medicion
long t0 = 0;//inicio de la variable que guarda el tiempo para el calculodel consumo
```

Figura 0.10 Variables Globales

Para finalizar se agregó la dirección de la pantalla *LCD* para su correcto funcionamiento, además de la frecuencia para trabajar con el sensor de voltaje y el objeto necesario para trabajar con el *ADS1115*.

```
LiquidCrystal_I2C lcd(0x27, 16, 2);//direccion i2c, sda=21, scl=22
ZMPT101B voltageSensor(vsensor, 60.0);//frecuencia en Hz para el sensor de voltaje
Adafruit_ADS1115 ads;//objeto necesario para el manejo del ADS1115
```

Figura 0.11 Objetos Declarados

Al contar con todo lo necesario para el funcionamiento de programa, se comienza con la creación del código para el medidor de consumo, definiendo la función principal conocida como ***void setup()*****,** esta función se encuentra inicializada una sola vez con cada encendido del prototipo.

La función ***void setup()*** en el código se encarga de la configuración inicial del dispositivo, antes de que comience el bucle principal de ejecución ***void loop()***. En primer lugar, se inicializan varias bibliotecas necesarias para el proyecto, incluyendo la comunicación *I2C*, el control de una pantalla *LCD*, la interacción con el sensor de voltaje *ZMPT101B* y el convertidor analógico a digital *ADS1115*.

Estas bibliotecas proporcionan las herramientas necesarias para la lectura de sensores y la visualización de datos en el *display LCD*. Posteriormente, se realiza una serie de configuraciones específicas del proyecto, como el inicio de la comunicación serial, la configuración de la resolución del pin analógico y la asociación del pin analógico al sensor de voltaje. Además, se establece la sensibilidad del sensor de voltaje y se configura el rango de ganancia del convertidor analógico a digital para garantizar mediciones precisas de la corriente.

Además de las configuraciones iniciales, la función ***void setup()*** se encarga de realizar la inicialización de propiedades para la comunicación con la nube *IoT* a través de *Arduino IoT Cloud*. Esto implica la llamada a la función ***initProperties(),*** que se encarga de definir las propiedades que el dispositivo puede enviar y recibir a través de la nube. Una vez que se han inicializado las propiedades, se inicia la conexión con *Arduino IoT Cloud* mediante la llamada a ***ArduinoCloud.begin()***. Esto establece la comunicación entre el dispositivo y la nube, permitiendo el intercambio de datos y la monitorización remota del sistema. Además, se configura el nivel de mensajes de depuración para facilitar la detección y solución de problemas durante la ejecución del programa. Estas configuraciones son esenciales para asegurar un funcionamiento correcto del sistema de medición.

```
void setup() {

  Wire.begin();//inicio de la lbreria Wire.h
  lcd.init();//Inicio de la pantalla
  lcd.clear();//Limpieza de la pantalla
  lcd.backlight();//encendido de la luz de fondo de la pantalla LCD
  lcd.print("IOT MEDIDOR");//Impresion del texto

  lcd.setCursor(0, 1); //posicion del cursor de la pantalla LCD
  lcd.print ("DE CONSUMO");//Impresion del texto

  Serial.begin(9600);//inicio del monitor serial
  analogReadResolution(12);//Configuración la resolución de lectura analógica
  adcAttachPin(vsensor);// Asociación del pin analógico para el sensor de voltaje
  //Serial.begin(9600);
  //analogReadResolution(12);
  //adcAttachPin(vsensor);

  voltageSensor.setSensitivity(SENSITIVITY);//Configuración la sensibilidad del sensor de voltaje

  ads.setGain(GAIN_TWO); //+/-2.048 /1 bit = 1mV, ganancia 2 utilizada para el ADS1115
  ads.begin();

  delay(3000);//Retraso agregado
  lcd.clear();//limpiar la pantalla lcd
  t0 = millis();//igualar el tiempo con la funcion millis que guarda el tiempo en milisegundos desde que inicia la placa


  initProperties();//Inicializa las propiedades definidas en thingProperties.h


  ArduinoCloud.begin(ArduinoIoTPreferredConnection);// Inicia la conexión con Arduino IoT Cloud


  setDebugMessageLevel(2);//nivel de mensajes de depuracion de arduino
  ArduinoCloud.printDebugInfo();//informacion sobre ArduinoCloud
}
```

Figura 0.12 *Void Setup*

La función ***void loop()*** es fundamental para el funcionamiento continuo del dispositivo. Comienza con la llamada a ***ArduinoCloud.update()***, una función esencial que mantiene la conexión con *Arduino IoT Cloud* actualizada. Esto permite una comunicación bidireccional entre el dispositivo y la plataforma en la nube, lo que posibilita recibir actualizaciones de la nube y enviar datos al servidor para su procesamiento y almacenamiento.

Después de actualizar la conexión con la nube, el código verifica el estado del interruptor mediante una estructura condicional ***if***. Si el interruptor está activado ***(HIGH)***, se inicia un temporizador para controlar la frecuencia de las mediciones. Cuando se alcanza el intervalo de tiempo especificado, se realizan mediciones de voltaje y corriente, y se calcula el consumo de energía llamando a la función secundaria ***valores().*** Estos valores se muestran en la pantalla *LCD* para que el usuario pueda monitorear el consumo en tiempo real. Además, el consumo de energía se imprime en el monitor serial para propósitos de depuración.

Por otro lado, si el interruptor está desactivado ***(LOW),*** se reinicia el contador de consumo de energía a 0 y se muestra un mensaje en la pantalla *LCD* indicando al usuario que inicie la medición. Esto garantiza que el consumo de energía no se registre cuando el dispositivo esté en un estado de reposo. El temporizador se reinicia para comenzar a contar el próximo intervalo de medición. Este flujo de control permite al

dispositivo realizar mediciones automáticas del consumo de energía cuando esté activo y detener las mediciones cuando no sean necesarias, contribuyendo a una gestión eficiente de la energía y los recursos del sistema.

En cada iteración del bucle ***void loop()***, se realizan una serie de tareas importantes para el funcionamiento del dispositivo, desde mantener la conexión con la nube hasta realizar mediciones y actualizar la pantalla *LCD*. Esta estructura de bucle infinito permite al dispositivo ejecutar continuamente su lógica principal, lo que le permite funcionar de manera autónoma y responder a los cambios en su entorno o estado.

```
void loop() {

  ArduinoCloud.update(); //actualiza la comunicacion con Arduino Cloud

  if (interruptor == HIGH)//bucle cuando el interruptor este activado
  {
    unsigned long currentMillis = millis();

    if (currentMillis - previousMillis >= interval) {
      previousMillis = currentMillis;

      valores();//llama a la funcion valores

      lcd.clear();//limpia la pantalla
      //Este bloque de código actualiza la pantalla LCD con los valores medidos de corriente (corrienteRMS),
      //voltaje (voltac), consumo de energía (consumopot), y potencia (poten).
      //Utiliza las funciones lcd.setCursor() para posicionar el cursor en la pantalla LCD y lcd.print() para mostrar los valores

      lcd.setCursor(0, 1);
      lcd.print("I:");
      lcd.print(corrienteRMS, 3);

      lcd.setCursor(0, 0);
      lcd.print("V:");
      lcd.print(voltac, 1);
```

Figura 0.13 *Void Loop* Parte 1

```
      lcd.setCursor(8, 1);
      lcd.print("Kwh:");
      lcd.print(consumopot, 2);

      lcd.setCursor(8, 0);
      lcd.print("P:");
      lcd.print(poten, 2);


      Serial.println(consumopot, 5);//imprime el valor del consumo en el monitor serial con 5 decimales

  }
```

Figura 0.14 *Void Loop* Parte 2

```
  } else
  {

    consumopot = 0.0;//reiniciando el contador de consumo de energía cuando el interruptor está desactivado.


    unsigned long currentMillis = millis();

    if (currentMillis - previousMillis >= interval) {

      previousMillis = currentMillis;
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("INICIE LA");
      lcd.setCursor(0, 1);
      lcd.print("MEDICION");


    }

  }

}
```

Figura 0.15 *Void Loop* Parte 3

Para finalizar se tiene la función sin retorno ***void valores()*** la cual desglosa el proceso de medición y cálculo del consumo de energía del dispositivo. Estas líneas declaran variables locales para almacenar los valores temporales necesarios durante el proceso de medición y cálculo. **volt_diferencial** almacena el valor diferencial del voltaje, **corriente** almacena el valor de corriente, **sum** es la suma acumulada de potencia, **tiempo** registra el tiempo actual en milisegundos y **counter** cuenta el número de mediciones tomadas durante el intervalo de tiempo definido. Además, se utiliza un temporizador para limitar el tiempo durante el cual se realizan las mediciones, lo que garantiza mediciones precisas dentro de un intervalo de tiempo definido.

Dentro de un bucle **while**, se lleva a cabo la toma de mediciones de voltaje y corriente. El voltaje RMS se obtiene del sensor de voltaje *ZMPT101B*, mientras que la corriente se calcula a partir de lecturas analógicas diferenciales utilizando el *ADS1115*. Estos valores se utilizan para calcular la potencia instantánea, que se suma a la variable **sumapot** para obtener el consumo acumulado de energía durante el intervalo de tiempo especificado.

Una vez finalizado el bucle de medición, se realiza el cálculo de la corriente RMS promedio utilizando la raíz cuadrada de la suma de los cuadrados de las mediciones de corriente tomadas durante el intervalo de tiempo. Esta corriente RMS se utiliza junto con el voltaje medido para calcular la potencia instantánea. El tiempo transcurrido desde la última medición se registra para el cálculo del consumo de energía en función del tiempo.

Finalmente, se actualiza el valor de **consumopot**, que representa el consumo de energía en kilovatios hora (KWh). Esta variable se calcula dividiendo la suma acumulada de potencia por el tiempo total transcurrido en segundos y luego por 3 600 para

convertirlo a horas. Esto proporciona una medida precisa del consumo de energía del dispositivo durante el intervalo de tiempo definido. Además, se verifica el estado del interruptor dentro de la función **valores()**, y si el interruptor cambia a estado bajo, las variables de consumo se reinician a 0 para evitar mediciones incorrectas.

```
void valores()//se define la fucion valores para realizarlos calculos
{
  float volt_diferencial;//almacena el valor diferencial del voltaje
  float corriente;//almacena el valor diferencial del voltaje
  float sum = 0; //inicializacion de la variable
  long tiempo = millis();//registra el tiempo actual en milisegundos
  int counter = 0;//cuenta el número de mediciones tomadas durante el intervalo de tiempo definido

  while (millis() - tiempo < 1000)//bucle while se ejecuta mientras no haya pasado un segundo completo desde el inicio de la medición
  {
    voltac = voltageSensor.getRmsVoltage(); //zmpt101b devuelve el valor RMS del voltaje medido por el sensor y lo asigna a la variable

    if (voltac <= 6.24)//se considera un valor insignificante y se establece voltac en 0.0
    {
      voltac = 0.0;
    }
    if (interruptor == LOW) {//verificacion del estado del interruptor
        consumopot = 0.0;//igualar la variable a 0 si se cambia el estado del interruptor
        sumapot = 0.0;//igualar la variable a 0 si se cambia el estado del interruptor

        return; // Salir de la función si el interruptor está apagado
    }
```

Figura 0.16 *Void* Valores Parte 1

```
        return; // Salir de la función si el interruptor está apagado
    }
    volt_diferencial = ads.readADC_Differential_0_1() * multiplier;//lee el valor del canal diferencial 0-1 del convertidor analógico digital
    //calcula la corriente multiplicando el voltaje diferencial por el factor de conversión
    corriente = volt_diferencial * factor;
    //se divide por 1000 para convertir la corriente de mA a A.
    corriente /= 1000.0;
    //agregan el cuadrado de la corriente actual a la suma acumulada de potencia (sum) y aumentan el contador de mediciones (counter) en 1
    sum += sq(corriente);
    counter = counter + 1;
  }

  corriente = sqrt(sum / counter);
  corrienteRMS = corriente;
  poten = voltac * corrienteRMS;
  dt = millis() - t0;
  t0 = millis();
  sumapot = (sumapot + poten) * (dt / 1000); //consumo cada segundo
  consumopot = sumapot / 3600000; //consumo cada hora o KWH

}
```

Figura 0.17 *Void* Valores Parte 2

A continuación, se muestra el diagrama de flujo del código para el sistema de medición de energía *IoT,* en el cual se muestra la secuencia de pasos necesarios para el control del programa. Como se muestra en la

Figura *0.18*.

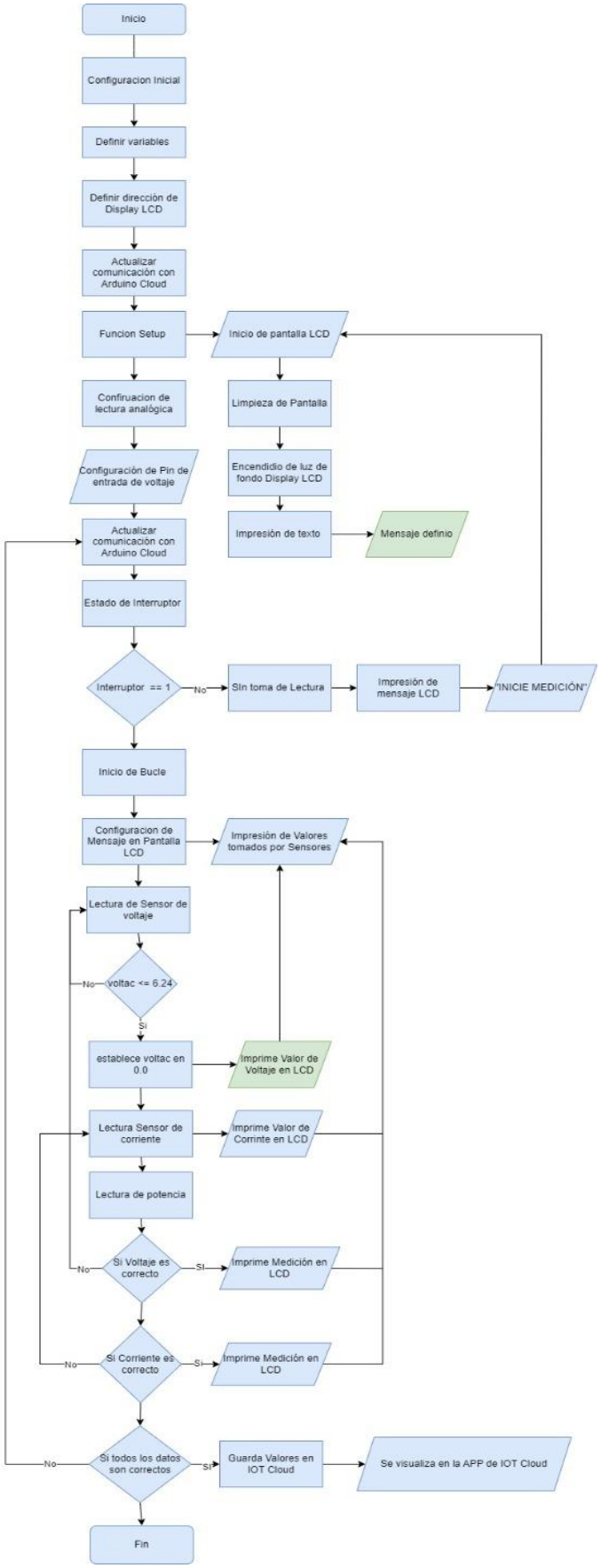

Figura 0.18 Diagrama de Flujo

Creación de la aplicación

En la presente sección se planteará como se llevó a cabo la creación de la aplicación en la plataforma de *Arduino IoT Cloud*, para lo cual es necesario cotar con las credenciales de acceso, estas son las mismas que se utilizaron para el desarrollo del código en el editor *web* de *Arduino*.

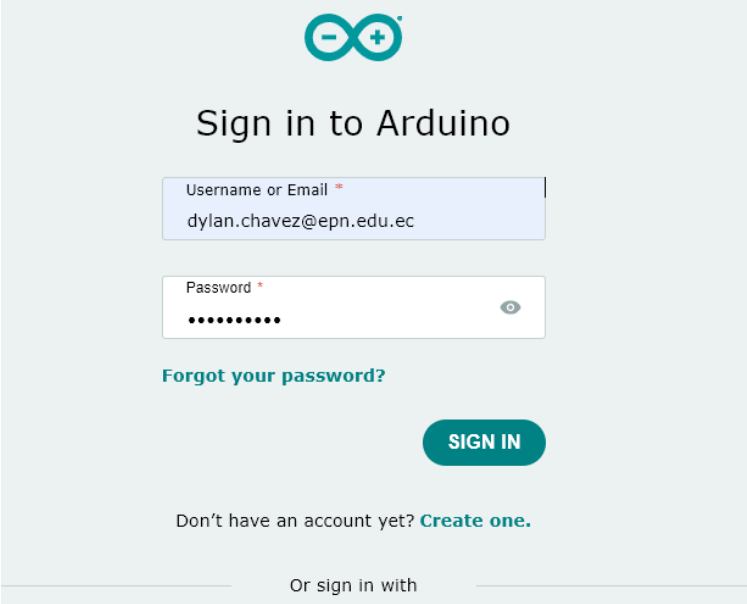

Figura 0.19 Inicio de Sesión

Al estar dentro de la página principal de *Arduino Cloud*, se podrá apreciar algunos apartados **Figura 0.20** en la parte izquierda de la página como: *Home, Sketches, Things, Dashboards, Triggers, Resources IoT Templates,* entre otros. Cabe mencionar que ciertas de estas opciones se encuentran habilitadas solo para cuentas pagada.

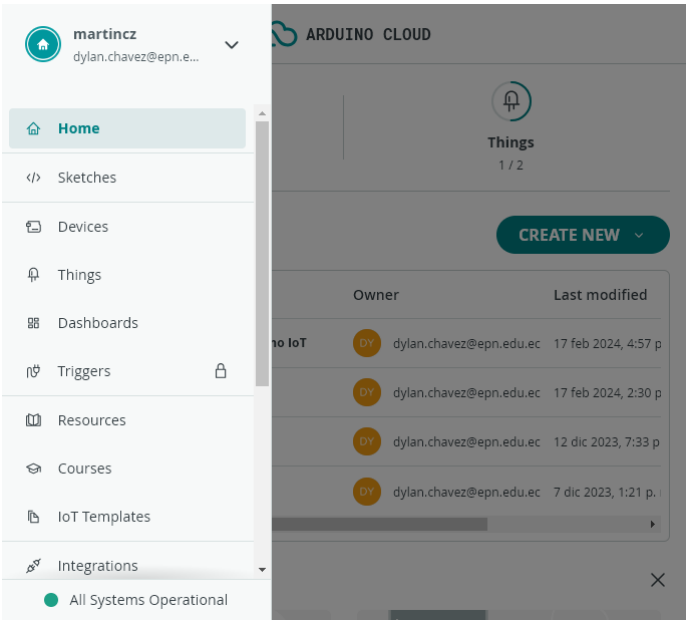

Figura 0.20 Apartados de *Arduino Cloud*

Dentro de los apartados mencionados, se deberá configurar como primer requisito la sección de *Things* en la cual se deberá crear un nuevo proyecto. Al ingresar se deberá configurar distintos puntos, como primer paso se deberá asociar un dispositivo **Figura 0.21** en el cual se podrá seleccionar la placa que se encuentra trabajando, para este apartado es necesario contar con el modelo justo con el que se realice el trabajo, en el caso del proyecto se utilizó la placa de desarrollo *ESP32* **Figura 0.22** que se asocia al modelo *NodeMCU-32S,* luego se podrá asignar un nombre al servicio configurado.

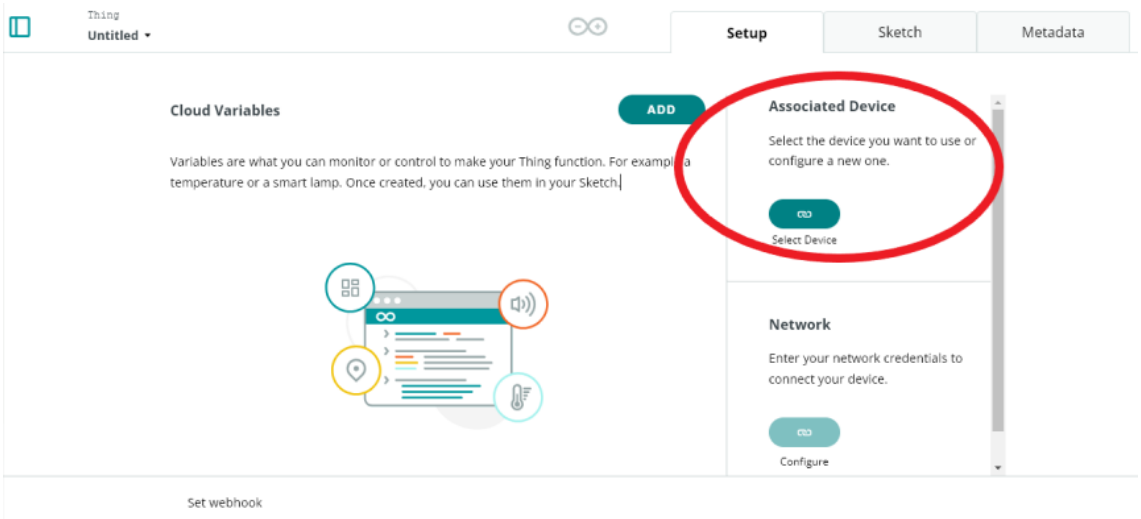

Figura 0.21 Asociación del Dispositivo

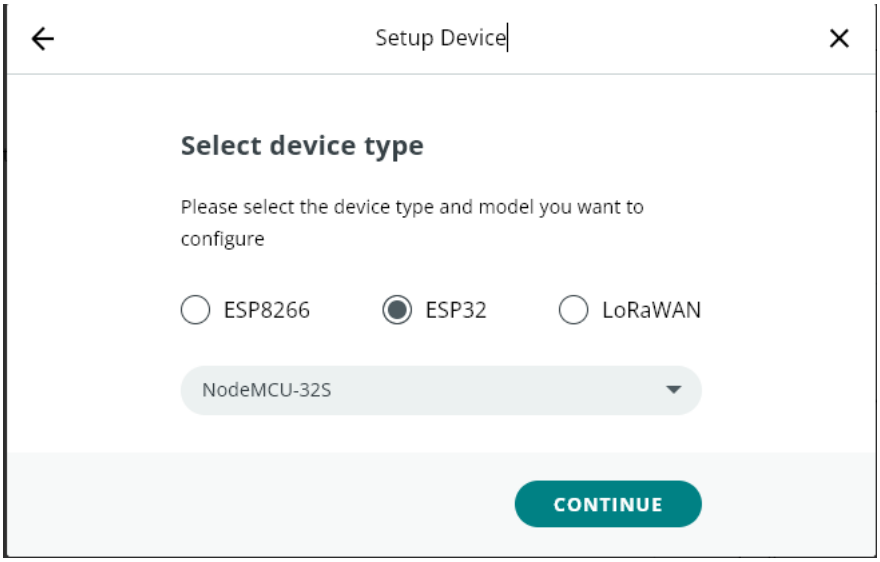

Figura 0.22 Elección de la Placa

Luego de asignar el nombre al Dispositivo la plataforma genera un identificador y una clave en un documento el cual se deberá descargar y guardar **Figura 0.23** para su posterior uso en la configuración de la red, se debe tener en cuenta que, si no se guarda la clave y se llega a perder, la placa de desarrollo no podrá tener acceso a internet, teniendo que volver a configurar un dispositivo para asociar la placa.

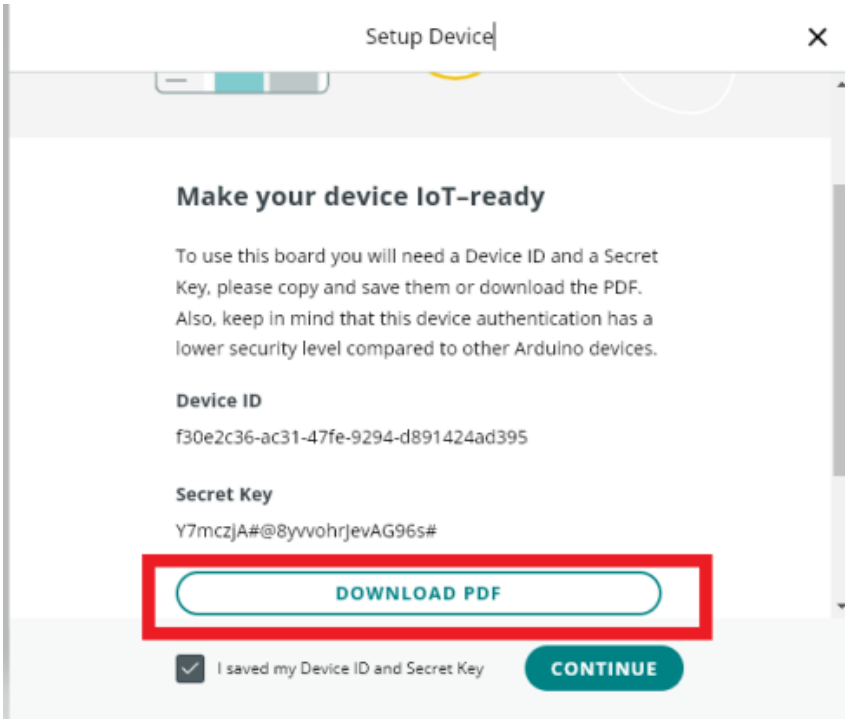

Figura 0.23 Descarga de la Clave Privada

En el siguiente paso se procederá a configurar la sección de la red **Figura 0.24** para contar con la conexión *Wi-Fi* que necesita la placa para enviar datos a la plataforma de *Arduino IoT Cloud*, en dicha sección se deberá contar con el identificador de conjunto de servicios (*SSID*), el cual representa el nombre público de la red *Wi-Fi, la contraseña de la red y la clave secreta que se proporcionó en la configuración del dispositivo.*

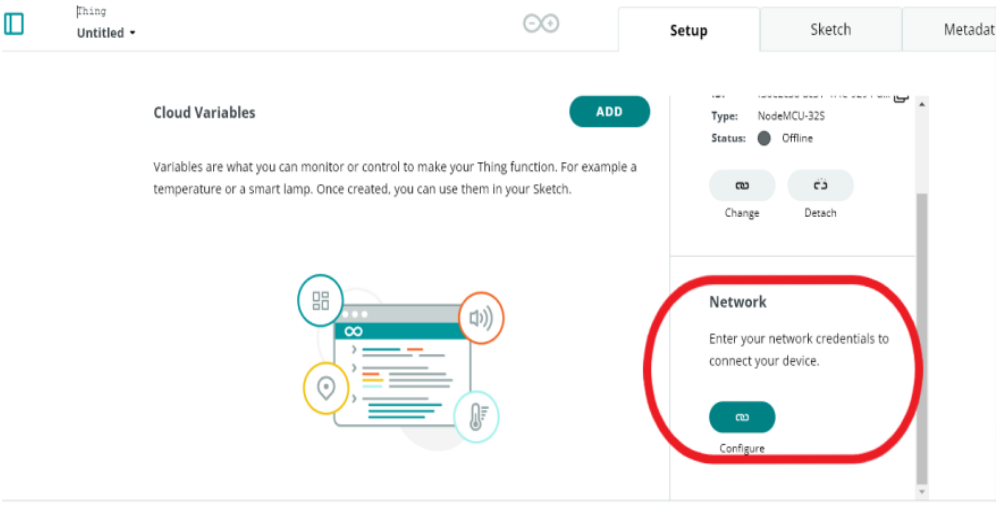

Figura 0.24 Configuración de la Red *Wi-Fi*

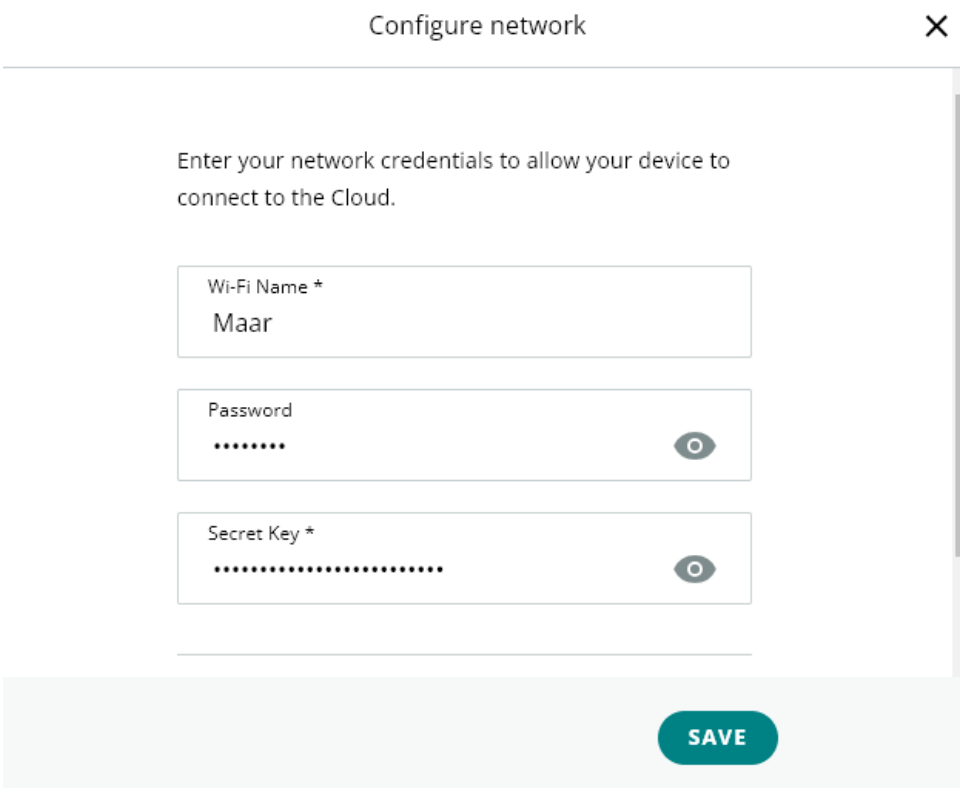

Figura 0.25 Credenciales para la Conexión *Wi-Fi*

Luego de configurar la red, se deberá ingresar las variables presentes en el código del medidor para ser asociadas a los distinto *widgets* utilizados para visualizar los datos necesarios en la aplicación *web* y celular. Para agregar las variables se asigna un nombre, el tipo de variable, la declaración en el código, los permisos y la pólitica para la actualización de las variables; la plataforma permite realizar esta tarea de una forma fácil y guiada.

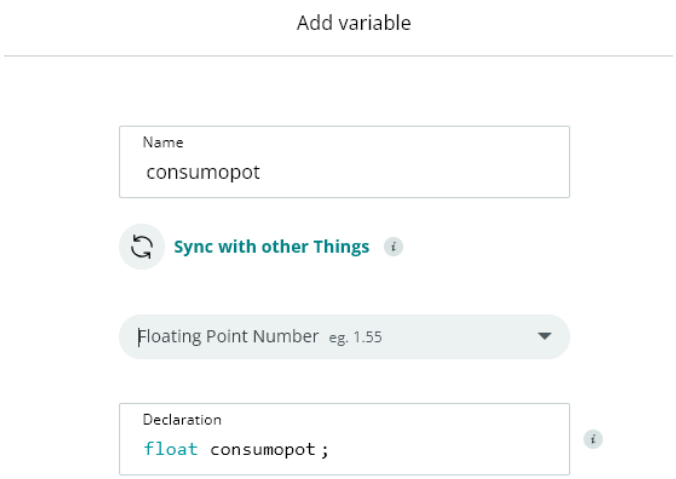

Figura 0.26 Ingreso de Variables

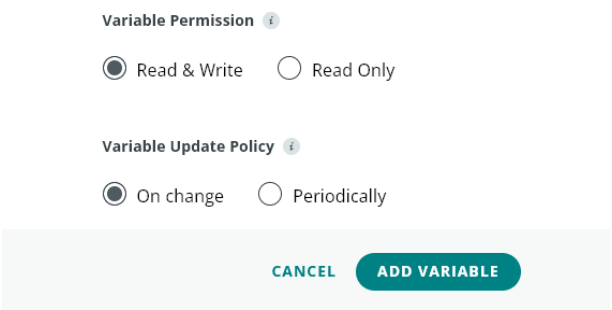

Figura 0.27 Configuración de Permisos y Políticas de Actualización

Dentro de las variables agregadas se tienen 5 que son el máximo permitido en la versión gratuita de *Arduino Cloud* como se muestra en la **Figura 0.28**.

Figura 0.28 Variables Usadas en el Código

Por último, se procederá al desarrollo del tablero en la opción de *Dashboards* en la cual se va a elegir los *widgets* necesarios para presentar la información requerida en el proyecto, dichos *widgets* se encontrarán asociados a las 5 variables presentes en la **Figura 0.28**.

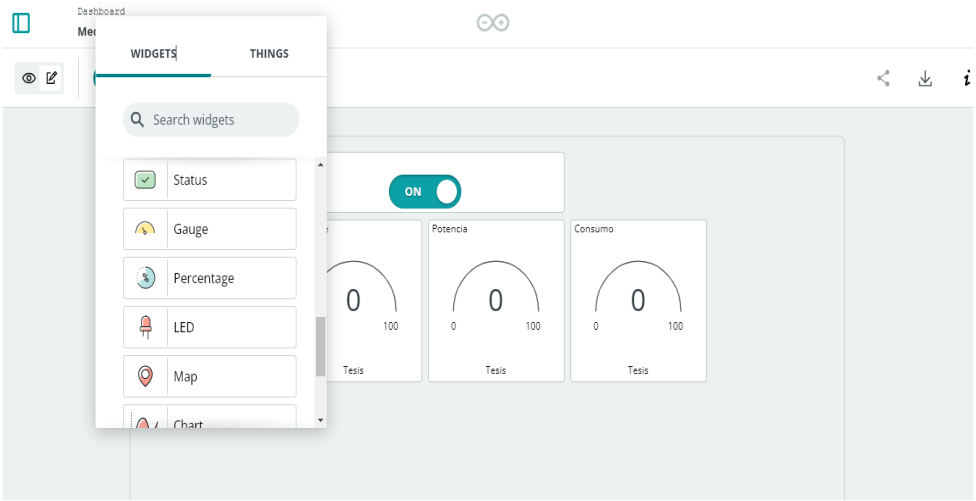

Figura 0.29 Tablero de la Aplicación

Para el tablero se utilizaron los siguientes *widgets*:

- ***Switch:*** este se encuentra vinculado a la variable tipo booleana interruptor, la cual toma el valor *HIGH* o *LOW* para iniciar y detener la medición del sistema *IoT*.

- ***Gauge:*** para el caso del proyecto se utilizó 4 *widgets* tipo gauge, los cuales están vinculados a las demás variables de tipo flotante como **consumopot** variable que guarda el valor del consumo energético, **voltac** variable que guarda el valor del voltaje, **corrienteRMS** que representa el valor de la corriente, y **poten** que presenta el valor de la potencia calculada.

El apartado de *Dashboard* en *Arduino* permite adecuar el tablero, contando con algunas opciones de personalización. Con estas opciones se puede obtener un diseño muy sencillo de implementar y modificar a gusto personal.

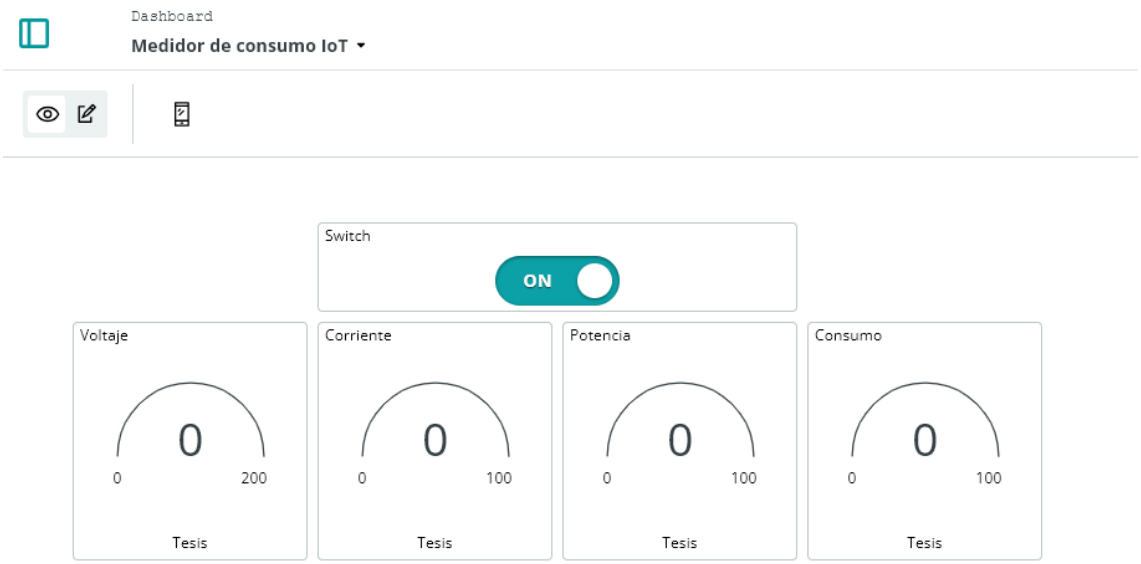

Figura 0.30 *Dashboard* de la Aplicación *Web*

A continuación, se muestra la **Figura 0.31** donde se puede apreciar le diseño de la aplicación móvil para el medidor de consumo *IoT*.

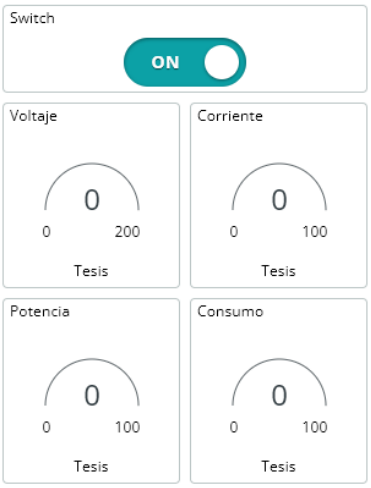

Figura 0.31 *Dashboard* de la Aplicación Celular

Alimentación

El prototipo de sistema *IoT* para la medición de consumo energético se alimentará mediante un puerto micro-USB, el cual se encuentra integrado en el microcontrolador *ESP32*. La conexión se llevará a cabo utilizando el cargador ilustrado en la figura, que suministra un voltaje de 5 (V) en su salida. Este cargador será conectado a un corta picos adecuados en la pared para brindar seguridad con sobretensiones que puedan dañar el prototipo. Para el prototipo, se utilizará un cargador de la marca LG modelo MCS-V02EA **Figura 0.32**, que ofrece un voltaje de salida de 5 (V) y una corriente máxima de salida de 2 (A).

Es importante destacar que el voltaje de operación del *ESP32* se encuentra dentro del rango de 3.3 (V) a 5 (V). Por lo tanto, el cargador seleccionado cumple con los requisitos de voltaje y corriente necesarios para garantizar el correcto funcionamiento del sistema. Esta elección se basa en la capacidad del cargador para proporcionar el voltaje y la corriente adecuados, asegurando así un suministro de energía estable y confiable para el funcionamiento óptimo del microcontrolador y otros componentes del sistema.

Figura 0.32 Alimentación

Implementación del prototipo

Desarrollo de la placa de circuito impreso

Para el desarrollo de la placa se utilizó la herramienta de *software* Proteus la cual permite crear esquemáticos de circuitos, diseñar placas de circuito impreso (PCB) y realizar simulaciones para verificar el funcionamiento y la eficiencia de los diseños antes de fabricar los componentes físicos. Es una herramienta versátil que se utiliza ampliamente en la industria electrónica para agilizar el proceso de diseño y reducir costos de desarrollo.

Para el diseño del circuito se tomó en cuenta el esquema creado en Proteus **Figura 0.4**, con esto se consideró los distintos dispositivos que deberán ir colocados en la PCB al no contar con las librerías necesarias para todos los elementos, se utilizaron regletas para simular los pines necesarios, por último, el circuito se implementó sobre una baquelita de 7 x 6 (cm).

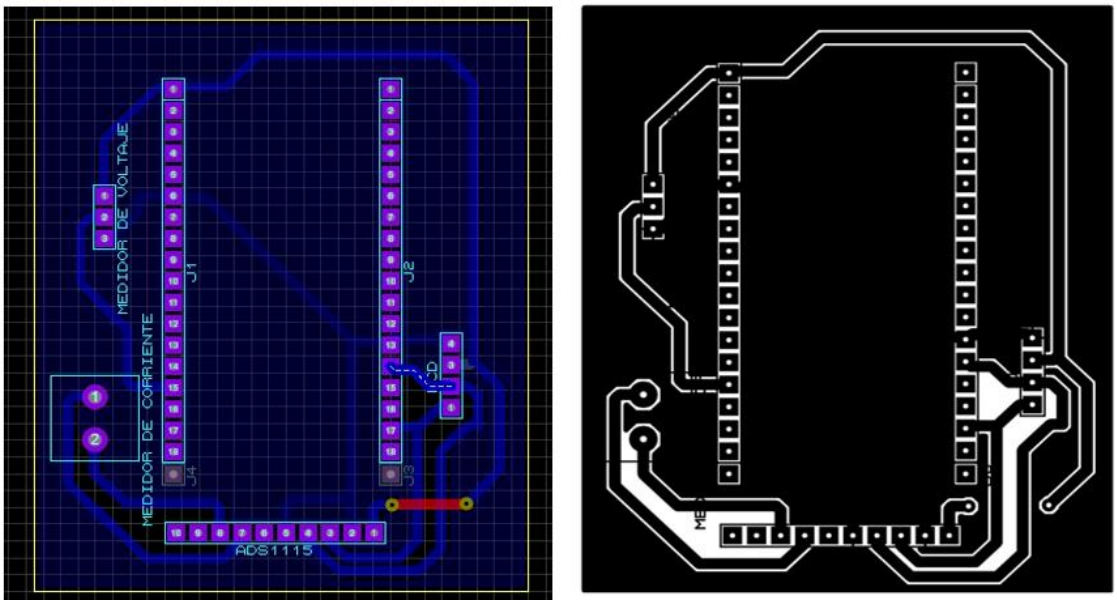

Figura 0.33 Diseño *PCB*

Para el desarrollo de la PCB se utilizó el método de trasferencia térmica, por lo cual se limpió la baquelita con un estropajo de hierro para que la transferencia del circuito sea correcta, para la impresión del circuito se utilizó papel fotografía y una impresora láser para obtener una impresión fácil de transferir, para la trasferencia del circuito en la baquelita se utilizó una plancha domestica fundiendo la tinta del papel en el cobre ejerciendo presión y realizando movimientos continuos. Una vez que el tono del papel cambió y el papel se pegó al cobre se precede al revelado.

Por último, la placa se sumerge en ácido para disolver el cobre que no se encuentra cubierto con la tinta de la transferencia; quedando lista para la perforación realizada con una broca de 1 (mm) y posterior el montaje de los elementos electrónicos.

Con la PCB ya perforada se colocaron espadines hembra para situar la placa de desarrollo *ESP32* y el *ADS1115* **Figura 0.35**, dichos componentes irán fijos en la PCB. Además, se implementó una bornera de 2 pines para el sensor de corriente, de igual manera 2 espadines hembra para el sensor de voltaje y la pantalla *LCD* **Figura 0.34**.

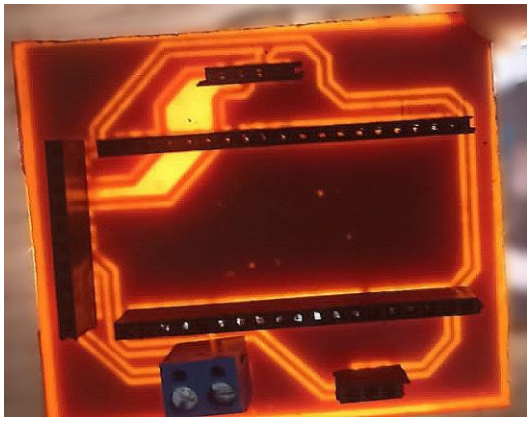

Figura 0.34 *PCB*

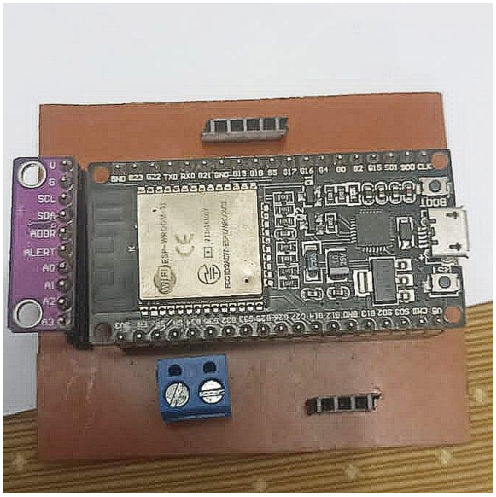

Figura 0.35 Integración del *ESP32* y el *ADS1115*

Una vez se pusieron todos los elementos en su lugar se procedió a soldar todos los elementos a la PCB **Figura 0.36**, además de estañar la parte posterior para evitar el deterioro de la placa.

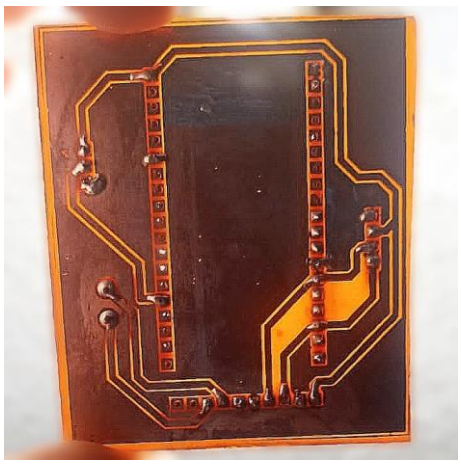

Figura 0.36 Placa Soldada

Elaboración de la caja 3D para el prototipo.

Para la protección y mejor manipulación del prototipo se elaboró una caja modelada en 3D de polietileno tereftalato glicol (PETG), el cual es un tipo de plástico muy usado en la impresión 3D, el cual brinda una excelente resistencia a los impactos además de ser un tipo de plástico reciclable loque lo convierte en una opción respetuosa con el medio ambiente.

Esta caja cuenta con ciertas aberturas e indicadores para las conexiones de los sensores y una abertura para la colocación de la pantalla *LCD*.

La caja fue diseñada utilizando *Tinkercad* la cual es una herramienta que permite diseñar modelos 3D de una manera sencilla e intuitiva siendo una gran opción para personas sin tanta experiencia en modelado 3D [16].

Las dimensiones con las que cuenta la caja son de 10 (cm) x 6 (cm) x 7 (cm) lo que brinda el espacio suficiente para adecuar todo el prototipo dentro de la caja, dejando solo los conectores banana hembra para los sensores, siendo la pinza del sensor de corriente el único elemento electrónico del prototipo fuera de la caja, ya que debe estar en contacto con el sistema eléctrico de la vivienda.

Montado del sistema en la vivienda

Para la instalación del prototipo se procedió con sumo cuidado a retirar la tapa de la caja de distribución **Figura 0.37** para acceder a los cables de fase y neutro que alimentan a la vivienda eligiendo solo la fase que alimenta a la vivienda inferior ya que solo se puede medir un sistema monofásico, posteriormente con precaución, se realizaron las conexiones necesarias para el funcionamiento del prototipo, como las conexiones de fase y neutro para el medidor de voltaje y la instalación de la pinza amperimétrica del sensor de corriente **Figura 0.38**.

Figura 0.37 Retiro de la tapa de la caja de distribución

Figura 0.38 Conexión de la pinza

Para mayor facilidad al momento de manipular el prototipo se decidió hacer uso de una extensión eléctrica en la cual se colocará la fuente de alimentación de 5 voltios del prototipo.

Figura 0.39 Colocación de la extensión

El medidor se colocó en la pared con ayuda de cinta doble *faz* como se muestra en las imágenes.

Figura 0.40 Colocación de cinta doble *faz*

Figura 0.41 Instalación del Sistema

Instalación y acceso a la aplicación Android

Para la aplicación se deberá acceder a la tienda de aplicaciones de *Google* conocida como *Play Store*, una vez dentro se buscará la aplicación de *Arduino IoT Cloud* para dispositivos *Android* y se realizará la instalación de este.

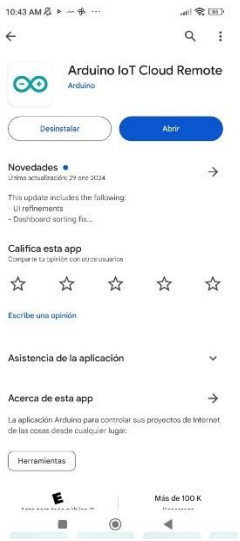

Figura 0.42 Aplicación de *Arduino Cloud* para *Android*

Al contar con la aplicación instalada será necesario usar las mismas credenciales que fueron utilizadas para la creación de la aplicación en la página *web* de *Arduino IoT Cloud* una vez dentro se podrá visualizar el nombre del *dashboard* **Medidor de consumo *IoT*,** el cual fue creado en el proceso de la elaboración del proyecto.

Figura 0.43 Ingreso al tablero creado

Realización de pruebas de funcionamiento del prototipo

Al terminar con todo el proceso de la implementación del prototipo se procedió a realizar pruebas para asegurar el funcionamiento del sistema.

Una vez encendido el prototipo y establecida la conexión se presentará en la pantalla LCD el mensaje de Inicie la medición con lo cual, una vez se ingrese a la aplicación móvil se podrá activar el *switch* virtual con el cual cuenta el programa para iniciar la medición de voltaje, corriente, potencia y consumo; estos datos podrán ser visualizados tanto en la pantalla LCD como en la aplicación móvil y web en simultaneo.

Figura 0.44 Funcionamiento del prototipo

Para finalizar, el usuario tiene la posibilidad de desactivar el switch virtual para detener las mediciones.

Figura 0.45 Medición Apagada

En la siguiente tabla se muestra un resumen del funcionamiento del prototipo de medición.

Tabla 0.2 Funcionamiento del prototipo

Medición	Correcto	Incorrecto
Medición de Voltaje	X	
Medición de Corriente	X	
Medición de Potencia	X	
Medición de Consumo	X	

Costo del prototipo

En la siguiente tabla se muestra el costo de cada elemento para la implementación del prototipo, los cuales están basados en los índices manejados dentro del mercado ecuatoriano.

Tabla 0.3 Costo del prototipo

Dispositivo	Cantidad	Valor Unitario	Valor Total
ESP32	1	$ 17.99	$ 17.99
Adaptador serial I2C	1	$ 2.24	$ 2.24
Sensor de voltaje AC ZMPT101B	1	$ 6.61	$ 6.61
Sensor de corriente no invasivo SCT-013	1	$ 10.19	$ 10.19
Módulo convertidor analógico digital ADS1115	1	$ 7.15	$ 7.15

LCD1602 Retroiluminación azul	1	$ 5.02	$ 5.02
Cables protoboard	13	$ 0.08	$1.04
Caja 3D	1	$ 6.25	$ 6.25
Mano de obra	18 (h)	$ 10.00	$ 180.00
Conectores banana hembra	4	$ 0.20	$ 0.80
Conector banana macho	4	$ 0.10	$ 0.40
Conector Lagarto	2	$ 0.10	$ 0.20
		Total	$ 237.89

CONCLUSIONES

- Se concluyó que uno de los requerimientos más esenciales para este proyecto es el uso de la plataforma de *Arduino IoT Cloud* ya que esta herramienta fue la que proporcionó la interoperabilidad entre el *hardware* y las aplicaciones que el usuario final va a constatar para poder acceder a los datos brindados por el sistema de medición.
- La implementación realizada usando *hardware* libre facilitó la creación del proyecto ya que al ser gratuito este tipo de *hardware* cuenta con una variedad de información brindada por la comunidad que previamente usó este *hardware*, por lo que varios de los problemas o inconvenientes pueden ser consultados y ser resueltos de una manera muy sencilla.
- El envió de la información hacia la plataforma de *Arduino IoT Cloud* se vuelve sencilla por el uso de la placa de desarrollo *ESP32* ya que cuenta de manera nativa con el módulo para establecer la conexión *Wi-Fi*, además la plataforma de *Arduino Cloud* cuenta con una librería, la cual de manera automática establece la conexión a través de internet con el dispositivo lo que facilita aún más el envío de datos, sin tener que agregar más líneas de código para establecer la conexión con internet.
- El uso de una aplicación móvil para visualizar el consumo energético que proporciona el prototipo es de gran utilidad ya que en la actualidad la mayoría de las personas cuentan con un dispositivo móvil, por lo que al contar con una conexión a Internet pueden acceder fácilmente a los datos del consumo del sistema desde cualquier lugar y momento.
- En conclusión, el desarrollo de este sistema de medición de consumo eléctrico permite obtener datos precisos sobre el uso de energía en una vivienda o

dispositivo siempre y cuando el sistema cuente con una sola fase, lo que facilita la monitorización y la implementación de medidas para mejorar la eficiencia energética.

- La utilización de *hardware* libre en el desarrollo del dispositivo de medición y comunicación inalámbrica garantiza la accesibilidad, la transparencia y la posibilidad de personalización, promoviendo así la innovación y la colaboración en este ámbito.
- La transmisión inalámbrica de los datos de consumo eléctrico hacia Internet proporciona una forma eficiente y práctica de acceder a la información en tiempo real desde cualquier ubicación a través de la aplicación móvil o *web*, lo que facilita la visualización de los datos en tiempo real.
- La disponibilidad de visualización de los datos de consumo eléctrico a través de una aplicación *web* y/o móvil brinda a los usuarios la comodidad y la flexibilidad de supervisar su consumo en cualquier momento y desde cualquier lugar, supervisando el uso de energía en alguna vivienda o de algún dispositivo.
- Para la implementación del prototipo fue requerido realizar ciertas conexiones de manera precautelada en la caja de distribución eléctrica de la vivienda, situación que representa un cierto riesgo al manipular el cable de fase para la conexión con el sensor de voltaje, de igual manera el prototipo debe encontrarse fijo ya que los cables conectados a los sensores no deben sufrir ningún tipo de cambio.
- Por medio de las pruebas realizadas se llegó a la conclusión, de que para llegar a interpretar las señales tan pequeñas del sensor de corriente cuando se median cargas bajas, era necesario el uso del ADS1115 con este dispositivo se llegó a interpretar de mejor manera las señales entregadas por el sensor.

RECOMENDACIONES

- El sistema de medición de energía eléctrica fue diseñado para medir únicamente una fase, por lo cual se recomienda desarrollar un prototipo que se encuentre enfocado en la medición de sistemas bifásicos y trifásicos como se presenta en algunos hogares que tienden a demandar el uso de más energía por la gran cantidad de dispositivos que manejan.
- Se recomienda agregar una fuente de poder que permita asegurar el funcionamiento del prototipo incluso cuando exista un corte de energía en la

vivienda, ya que esto favorecerá a la conservación de los datos acerca del consumo que se tengan hasta el momento.

- Se recomienda el desarrollo de un sistema que soporte condiciones industriales, por lo que tanto los sensores de voltaje y corriente deben ser actualizados a unos sensores que manejen más voltaje y corriente adecuadas a un uso industrial.
- Para el manejo del sensor de voltaje se recomienda tomar las medidas de precaución necesarias para evitar descargas en operadores que manipulen el prototipo, además se enfatiza en colocar adecuadamente la línea de fase y neutro en el sensor de voltaje ya que la mala conexión puede ocasionar que se dañe de manera irreversible el dispositivo.
- Se recomienda tomar en cuenta el voltaje y la corriente máxima que pueden soportar los sensores, ya que al ser implementados en ambientes donde se supere las condiciones para los cuales fueron diseñados pueden tener un mal funcionamiento e incluso dañarse.

REFERENCIAS BIBLIOGRÁFICAS

[1] P. Contact, «PHOENIX CONTACT,» PHOENIX CONTACT S.A., [En línea]. Available: https://www.phoenixcontact.com/es-cl/tecnologias/gestion-energia/gestion-energia-basada-iot. [Último acceso: 2 Febrero 2024].

[2] «Arduino Proyectos y Tutoriales,» [En línea]. Available: https://descubrearduino.com/esp32-modulo-esp32-wroom-gpio-pinout/. [Último acceso: 1 Febrero 2024].

[3] N. M. SAC, «Naylamp Mechatronics,» [En línea]. Available: https://naylampmechatronics.com/espressif-esp/384-nodemcu-32-30-pin-esp32-wifi.html. [Último acceso: 2 Febrero 2024].

[4] «Arduino Cloud | Build, Control, Monitor Your IoT Projects,» [En línea]. Available: https://cloud.arduino.cc/how-it-works/. [Último acceso: 1 Febrero 2024].

[5] «Naylamp Mechatronics - Perú,» [En línea]. Available: https://naylampmechatronics.com/blog/51_tutorial-sensor-de-corriente-ac-no-invasivo-sct-013.html. [Último acceso: 1 febrero 2024].

[6] Ferretronica, «Ferretronica,» [En línea]. Available: https://ferretronica.com/products/sensor-corriente-ac-no-invasivo-100a-1v-sct-013-100. [Último acceso: 1 Febrero 2024].

[7] «Arduino Spain,» [En línea]. Available: https://sp.arduino-france.site/zmpt101b-arduino/. [Último acceso: 12 Diciembre 2024].

[8] U. ELECTRONICS, «Unit ELECTRONICS,» [En línea]. [Último acceso: 2 Febrero 2024].

[9] «Programarfacil Arduino y Home Assistant,» [En línea]. Available: https://programarfacil.com/blog/arduino-blog/ads1115-convertidor-analogico-digital-adc-arduino-esp8266/. [Último acceso: 19 Febrero 2024].

[10] N. M. SAC, «Naylamp Mechatronics,» [En línea]. Available: https://naylampmechatronics.com/conversores/394-modulo-adc-ads1115.html. [Último acceso: 1 Febrero 2024].

[11] naylampmechatronics, «Naylamp Mechatronics,» [En línea]. Available: https://naylampmechatronics.com/blog/35_tutorial-lcd-con-i2c-controla-un-lcd-con-solo-dos-pines.html. [Último acceso: 1 Febrero 2024].

[12] S. Roca, «Cenditel,» 31 Marzo 2020. [En línea]. Available: http://hl.cenditel.gob.ve/2020/03/31/hardware-libre-que-significa/. [Último acceso: 12 Diciembre 2024].

[13] Megatronica, «MEGATRONICA,» [En línea]. Available: https://megatronica.cc/producto/pantalla-lcd-1602-arduino-3d/. [Último acceso: 2 Febrero 2024].

[14] arduinodesdecero, «arduinodesdecero,» [En línea]. Available: https://arduinodesdecero.com/tipos/ide/. [Último acceso: 2 Febrero 2024].

[15] T. HUMANIZADA, «TECNOLOGIA HUMANIZADA,» 6 Noviembre 2018. [En línea]. Available: https://humanizationoftechnology.com/blynk-plataforma-de-internet-de-las-cosas-en-la-red/revista/2018/volumen-4-2018/11/2018/. [Último acceso: 20 Diciembre 2024].

[16] A. M, «3D Natives,» 22 Septiembre 2023. [En línea]. Available: https://www.3dnatives.com/es/tinkercad-software-200420202/. [Último acceso: 20 Febrero 2024].

ANEXOS

La lista de los **Anexos** se muestra a continuación:

ANEXO I. Certificado de originalidad

ANEXO II. Enlaces

ANEXO III. Conjunto de datos extensos

ANEXO I: Certificado de Originalidad

F_AA_236

CERTIFICADO DE ORIGINALIDAD
TRABAJO DE INTEGRACIÓN CURRICULAR

Quito, D.M. 20 de febrero de 2024

De mi consideración:

Yo, **LEANDRO ANTONIO PAZMIÑO ORTIZ**, en calidad de Director del Trabajo de Integración Curricular titulado **IMPLEMENTACIÓN DE UN PROTOTIPO DE SISTEMA IOT PARA LA MEDICIÓN DEL CONSUMO DE ENERGÍA ELÉCTRICA EN UNA VIVIENDA** asociado al proyecto **IMPLEMENTACIÓN DE UN PROTOTIPO DE SISTEMA IOT PARA LA MEDICIÓN DEL CONSUMO DE ENERGÍA ELÉCTRICA EN UNA VIVIENDA** elaborado por el estudiante **DYLAN MARTIN CHÁVEZ HERAS** de la carrera en **TECNÓLOGÍA SUPERIOR EN REDES Y TELECOMUNICACIONES**, certifico que he empleado la herramienta antiplagio "TURNITIN" para la revisión de originalidad del documento escrito producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 12%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el informe generado por la herramienta Turnitin.

Atentamente,

Leandro Pazmiño Ortiz
Docente
Escuela de Formación de Tecnólogos

ANEXO II: Enlaces

Anexo II.I Código QR de la implementación y pruebas de funcionamiento del prototipo de medidor de consumo eléctrico

Link: Presentación Medidor IoT DC.mp4

ANEXO III: Códigos Fuente

```
#include "thingProperties.h"//librería de Arduino

#include <Wire.h>//librería para la comunicación con servicios I2C

#include <LiquidCrystal_I2C.h>//Librería para utilizar la pantalla LCD

#include <ZMPT101B.h>//Librería para el uso del sensor de voltaje

#include <Adafruit_ADS1X15.h>//Librería para el manejo del ADS1115


#define SENSITIVITY 543.750f     // sensibilidad del sensor de voltaje obtenido de la calibración del sensor

#define vsensor 34               // pin analógico para el sensor de voltaje


const float factor = 100;// factor del sensor de corriente

const float multiplier = 0.0309F;//factor para el cálculo de la corriente

unsigned long previousMillis = 0;

const long interval = 1000; //constante utilizada para el cálculo del consumo

float sumapot = 0; //inicio de la variable que guarda el consumo cada segundo

long dt = 0;//inicio de la variable que guarda la diferencia de tiempo desde la última medicion

long t0 = 0;//inicio de la variable que guarda el tiempo para el cálculo del consumo


LiquidCrystal_I2C lcd(0x27, 16, 2);//direccion i2c, sda=21, scl=22

ZMPT101B voltageSensor(vsensor, 60.0);//frecuencia en Hz para el sensor de voltaje

Adafruit_ADS1115 ads;//objeto necesario para el manejo del ADS1115


void setup() {
```

```
  Wire.begin();//inicio de la librería Wire.h
  lcd.init();//Inicio de la pantalla
  lcd.clear();//Limpieza de la pantalla
  lcd.backlight();//encendido de la luz de fondo de la pantalla LCD
  lcd.print("IOT MEDIDOR");//Impresión del texto

  lcd.setCursor(0, 1); //posición del cursor de la pantalla LCD
  lcd.print ("DE CONSUMO");//Impresión del texto

  Serial.begin(9600);//inicio del monitor serial
  analogReadResolution(12);//Configuración la resolución de lectura analógica
  adcAttachPin(vsensor);// Asociación del pin analógico para el sensor de voltaje

  voltageSensor.setSensitivity(SENSITIVITY);//Configuración la sensibilidad del sensor de voltaje

  ads.setGain(GAIN_TWO); //+/-2.048 /1 bit = 1mV, ganancia 2 utilizada para el ADS1115
  ads.begin();

  delay(3000);//Retraso agregado
  lcd.clear();//limpiar la pantalla lcd
  t0 = millis();//igualar el tiempo con la función millis que guarda el tiempo en milisegundos desde que inicia la placa

  initProperties();//Inicializa las propiedades definidas en thingProperties.h
```

```
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);// Inicia la conexión con Arduino IoT Cloud



  setDebugMessageLevel(2);//nivel de mensajes de depuración de Arduino
  ArduinoCloud.printDebugInfo();//información sobre ArduinoCloud
}

void loop() {

  ArduinoCloud.update(); //actualiza la comunicación con Arduino Cloud

  if (interruptor == HIGH)//bucle cuando el interruptor este activado
  {
    unsigned long currentMillis = millis();

    if (currentMillis - previousMillis >= interval) {
      previousMillis = currentMillis;

      valores();//llama a la función valores

      lcd.clear();//limpia la pantalla
      //Este bloque de código actualiza la pantalla LCD con los valores medidos de corriente (corrienteRMS),
      //voltaje (voltac), consumo de energía (consumopot), y potencia (poten).
      //Utiliza las funciones lcd.setCursor() para posicionar el cursor en la pantalla LCD y lcd.print() para mostrar los valores
```

```
      lcd.setCursor(0, 1);
      lcd.print("I:");
      lcd.print(corrienteRMS, 3);

      lcd.setCursor(0, 0);
      lcd.print("V:");
      lcd.print(voltac, 1);

      lcd.setCursor(8, 1);
      lcd.print("Kwh:");
      lcd.print(consumopot, 2);

      lcd.setCursor(8, 0);
      lcd.print("P:");
      lcd.print(poten, 2);


      Serial.println(consumopot, 5);//imprime el valor del consumo en el monitor serial con
5 decimales

    }

  } else
  {
```

```
    consumopot = 0.0;//reiniciando el contador de consumo de energía cuando el interruptor está desactivado.



    unsigned long currentMillis = millis();

    if (currentMillis - previousMillis >= interval) {

      previousMillis = currentMillis;
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("INICIE LA");
      lcd.setCursor(0, 1);
      lcd.print("MEDICION");
      consumopot = 0.0;//igualar la variable a 0 si se cambia el estado del interruptor
      voltac = 0.0;//igualar la variable a 0 si se cambia el estado del interruptor
      corrienteRMS = 0.0;
      poten = 0.0;

    }

  }

}

void valores()//se define la función valores para realizarlos cálculos
{
```

```
  float volt_diferencial;//almacena el valor diferencial del voltaje

  float corriente;//almacena el valor diferencial del voltaje

  float sum = 0; //inicialización de la variable

  long tiempo = millis();//registra el tiempo actual en milisegundos

  int counter = 0;//cuenta el número de mediciones tomadas durante el intervalo de tiempo definido



  while (millis() - tiempo < 1000)//bucle while se ejecuta mientras no haya pasado un segundo completo desde el inicio de la medición

  {

    voltac = voltageSensor.getRmsVoltage(); //zmpt101b devuelve el valor RMS del voltaje medido por el sensor y lo asigna a la variable


    if (voltac <= 6.24)//se considera un valor insignificante y se establece voltac en 0.0

    {

      voltac = 0.0;

    }


    volt_diferencial = ads.readADC_Differential_0_1() * multiplier;//lee el valor del canal diferencial 0-1 del convertidor analÃ³gico digital

    //calcula la corriente multiplicando el voltaje diferencial por el factor de conversiÃ³n

    corriente = volt_diferencial * factor;

    //se divide por 1000 para convertir la corriente de mA a A.

    corriente /= 1000.0;

    //agregan el cuadrado de la corriente actual a la suma acumulada de potencia (sum) y aumentan el contador de mediciones (counter) en 1

    sum += sq(corriente);
```

```
    counter = counter + 1;
  }


  corriente = sqrt(sum / counter);
  corrienteRMS = corriente;
  poten = voltac * corrienteRMS;
  dt = millis() - t0;
  t0 = millis();
  sumapot = (sumapot + poten) * (dt / 1000); //consumo cada segundo
  consumopot = sumapot / 3600000; //consumo cada hora o KWh


}
```