Universidad Nacional de Ingeniería

Dirección de Área de Conocimiento Industria y Producción

Prototipo de analizador de redes trifásico para monitoreo de magnitudes eléctricas en el área administrativa dentro de las instalaciones de la Universidad Nacional de Ingeniería en el Recinto Universitario Simón Bolivar (UNI-RUSB).

Trabajo Monográfico para optar al título de Ingeniero Eléctrico

Elaborado por:

Br. Douglas Alfonso Flores Suarez
Carnet: 2020-0060U

Br. Fredy Jose Chavarría Madrid
Carnet: 2020-0078U

Tutor:

Ing. Marlovio José Sevilla Hernández

22 de mayo de 2025
Managua Nicaragua

DEDICATORIA

Este proyecto monográfico se lo dedico primeramente a Dios, él es la fuente de toda mi fortaleza, brindándome siempre la sabiduría y la perseverancia para culminar exitosamente mi carrera.

A mis padres, que son los pilares fundamentales de mi educación, aconsejándome y dándome las mejores lecciones de vida que cualquier hijo pueda tener, su apoyo incondicional fue indispensable para perseverar en mi carrera y llegar a este punto.

A mis amigos, personas integras que me acompañaron en este largo proceso, que, a pesar de los inconvenientes, estuvieron incondicionalmente.

Douglas Alfonso Flores Suarez

DEDICATORIA

Este proyecto monográfico lo dedico a Dios, por haberme brindado la sabiduría y fortaleza necesaria para culminar mi carrera y durante el desarrollo de este proyecto.

A mis padres, ya que sin su sacrificio no hubiese podido llegar hasta este punto. Sus consejos y palabras de aliento fueron fundamentales para que pudiera ser perseverante durante toda mi carrera, sus vidas son un ejemplo de perseverancia ya que no se rindieron aun en los momentos difíciles.

A mi familia y amigos, por todo el apoyo incondicional que fue de mucha ayuda para seguir adelante, aún en los desafíos que se presentaron.

Fredy Jose Chavarria Madrid

AGRADECIMIENTOS

Queremos agradecer inmensamente a nuestro tutor Marlovio José Sevilla Hernández, por su gran apoyo durante todo este largo proceso de culminación de estudios. Su esfuerzo y dedicación nos permitió realizar el proyecto de la mejor manera. Su guía fue fundamental para desarrollar nuestro proyecto monográfico, compartiéndonos sus conocimientos y aconsejándonos para realizar una excelente entrega final.

Además, queremos extender un cálido agradecimiento al director especifico de Ing. Mecánica e Ing. Eléctrica, la responsable del Laboratorio de Máquinas Eléctricas quienes fueron vitales al brindar su apoyo a través de todo el proceso de la monografía. Asimismo, al responsable de Oficina de Servicios Administrativos RUSB, al responsable de Unidad de Mantenimiento RUSB y al Supervisor de Mantenimiento RUSB, fueron quienes nos proporcionaron el tiempo, apoyo y los permisos necesarios para culminar exitosamente.

Douglas Alfonso Flores Suarez

Fredy Jose Chavarria Madrid

RESUMEN

En esta investigación monográfica que se presenta a continuación se enfoca en el desarrollo y el diseño de un prototipo de analizador de redes trifásico para monitoreo de magnitudes eléctricas en el que internamente está compuesto por el microcontrolador ESP32; que gracias a sus características facilitan la integración de servicios de Internet de las Cosas (IoT), además de los sensores PZEM-004T que permiten la medición de los parámetros eléctricos y las respectivas conversiones de señales analógicas a digitales. El dispositivo tiene como objetivo facilitar y ofrecer una alternativa a la optimización energética y la gestión eficiente de recursos en el área administrativa de la Universidad Nacional de Ingeniería en el Recinto Universitario Simón Bolívar. El prototipo mide los parámetros de voltaje, corriente, potencia activa, reactiva, aparente, factor de potencia, frecuencia y energía en kwh, así bien enviar todos estos de manera inalámbrica a la plataforma InfluxDB Cloud que permite recopilar, almacenar, procesar, además de poder visualizar los datos de series temporales a través de una herramienta nativa llamada Grafana.

En esta investigación se organizó en temas que permite la compresión de los objetivos planteados, en los primeros temas se encuentra la exploración por el estado de arte además de generalidades, funcionalidades e importancia de los analizadores de redes y medidores eléctricos, mencionando también las principales expresiones matemáticas que son los fundamentos del comportamiento de los parámetros eléctricos a medir. Se definieron los conceptos necesarios para el pleno entendimiento de lo términos y tecnologías necesarias que se ocuparon.

En el diseño y desarrollo del prototipo nos enfocamos en mencionar y explicar las características, funcionalidades y aspectos técnicos que muestran tanto el software y hardware escogido en del presente prototipo. Seguidamente para el integro entendimiento del prototipo se encuentra ilustrado un esquema de conexión entre los componentes además de tablas específicas de conexión

acompañados también de diferente explicaciones e ilustraciones de diferentes diagramas de flujos necesarios para el entendimiento y comprensión minuciosa del funcionamiento conjunto interno del prototipo al realizar las tareas asignadas, respaldado a su vez por la explicación de las partes fundamentales del código.

Simultáneamente se mostró una breve guía de las configuraciones iniciales y las creaciones de cuentas en los servicios (IoT), además de evidencias del montaje y del desarrollo final del prototipo.

Al finalizar como parte esencial de nuestros objetivos se ejecutaron pruebas en el laboratorio de Maquinas eléctricas ubicado de la Universidad Nacional de Ingeniería en el Recinto Universitario Simón Bolívar, además de instalarlo en tableros eléctricos asignados por el área administrativa donde se comprobó la correcta funcionalidad, precisión, las consistencias de las trasmisiones, monitorizaciones, visualizaciones y grabado de las múltiples mediciones.

Al concluir la lectura de esta investigación monográfica, se evidenciarán el cumplimiento de todos y cada uno de los objetivos planteados además pretender ser los cimientos de futuras investigaciones en áreas como monitoreo, análisis de magnitudes eléctricas, monitoreo remoto en tiempo real, y el uso de tecnologías (IoT) facilitando una herramienta más para la optimización energética y la gestión eficiente de recursos eléctricos.

Palabras clave: Prototipo analizador de redes trifásico, ESP32, Internet de las Cosas (IoT), PZEM-004T, Medición de parámetros eléctricos, Conversión analógica-digital, InfluxDB Cloud, Grafana, Optimización energética, Gestión eficiente de recursos eléctricos.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE ECUACIONES

1. INTRODUCCIÓN

En el contexto actual, la eficiencia energética y la adecuada gestión de los recursos energéticos son temas importantes para las instituciones académicas y las empresas. La Universidad Nacional de Ingeniería en Recinto Universitario Simón Bolívar (UNI-RUSB), también está enfocada en esta premisa para optimizar el uso de la energía en sus recintos. El objetivo de este proyecto es desarrollar un modelo de monitor de red trifásico para monitorizar el consumo energético en tiempo real mediante comunicación inalámbrica (WIFI), visualizar datos a través de una plataforma informática y gestionar directamente la energía para prevenir fallos y aumentar la eficiencia. Este proyecto no sólo proporcionará una valiosa herramienta para la gestión eléctrica de UNI-RUSB, sino que también ayudará a promover la investigación y el uso de tecnologías IoT (Internet of Things) en sistemas eléctricos.

Las magnitudes medidas (tensión, corriente, factor de potencia, potencia activa, potencia reactiva, potencia aparente), además de ser visualizadas, se almacenarán en una memoria externa y con la opción de ser descargados los datos en un archivo desde una plataforma informática. Los componentes esenciales con los que se desarrollará el proyecto son el microprocesador ESP-WROOM-32 y los sensores PZEM-004t, mediante los cuales se realizarán todas las acciones.

Además de medir las magnitudes eléctricas, se implementará un sistema de respaldo; en caso de ausencia de medición en los parámetros eléctricos el dispositivo será capaz de enviar un aviso a una pantalla digital ubicada en el dispositivo.

2. OBJETIVOS

2.1 General

- Desarrollar un prototipo de analizador de redes trifásico con integración de tecnologías Internet of Things (IoT), para monitoreo y análisis de magnitudes eléctricas en tiempo real, facilitando la optimización energética y la gestión eficiente de recursos en el área administrativa de la Universidad Nacional de Ingeniería en el Recinto Universitario Simón Bolívar (UNI-RUSB).

2.2 Específicos

- Desarrollar un prototipo que utilice sensores destinado a medir parámetros eléctricos esenciales (voltaje, corriente, potencia activa, reactiva, aparente, factor de potencia y frecuencia) en sistemas trifásicos, y que almacene estos datos de manera segura para su análisis y visualización a través de una plataforma basada en Internet of Things (IoT).

- Implementar una plataforma IoT con conectividad Wi-Fi que permita el monitoreo remoto en tiempo real de las magnitudes eléctricas, utilizando un sistema de transmisión estable que almacene y visualice los datos energéticos en un software de código abierto.

- Diseñar en una plataforma digital permitiendo la visualización de datos recolectados por el prototipo en tiempo real, proporcionando herramientas para el análisis detallado y el almacenamiento de datos históricos para estudios de tendencias y comportamiento energético.

- Ejecutar pruebas en condiciones de laboratorio con el fin de verificar la precisión, confiabilidad y seguridad del prototipo, identificando aspectos a mejorar y asegurando el cumplimiento de estándares de seguridad eléctrica y eficiencia energética.

3. JUSTIFICACIÓN

Las mediciones eléctricas son datos fundamentales de información en la cual se desprende la cuantificación, ubicación y la clasificación de la energía consumida en cualquier instalación eléctrica, siendo esto el punto de partida para establecer e identificar la eficiencia energética, la utilización y la seguridad que deben cumplir en base a normativas vigentes en el país.

Con referencia a lo anterior las mediciones eléctricas conllevan la utilización de equipos de alto valor, además de un soporte técnico correspondiente. La ejecución del trabajo pretende diversificar y actualizar los métodos para la obtención y monitoreo remoto de los parámetros eléctricos en centros de cargas puntuales además de la prevención de fallos. Agregando un sistema de monitoreo y medición remoto constante modular, con él registrarán los parámetros de interés de forma histórica que la Universidad Nacional de Ingeniería en el Recinto Universitario Simón Bolívar (UNI-RUSB) aún no cuenta.

Precisando de una vez el funcionamiento del dispositivo a desarrollar propone medir, mostrar, monitorear de manera remota y continua, los parámetros eléctricos normales y anormales que se puedan presentar en la instalación (fluctuaciones de corriente, fluctuaciones de frecuencia y fluctuaciones de voltaje). El prototipo servirá como una herramienta que nos permitirá mejorar la capacidad de planeación de manteamientos preventivos, correctivos eficiencia energética, mejorar la productividad del personal al realizar los análisis de los parámetros eléctricos, asimismo facilitar el monitoreo y visualización a través de un software con registro de consumo.

4. MARCO TEÓRICO

4.1 Fundamentos Epistemológicos del Problema de Investigación

Es pertinente destacar que el Trabajo de Integración Curricular de (Paúl, 2022) incluyó la creación de un medidor de factor de potencia, con el objetivo de diseñarlo y construirlo, para monitorear el factor de potencia en la Universidad Indoamérica además de evaluar los resultados obtenidos por el medidor de factor de potencia dentro la Universidad Indoamérica. De esta manera plateando que el siguiente dispositivo podrá monitorear y adquirir de manera automatizada los parámetros pertinentes en la web, asimismo impactando novedosamente al poseer la capacidad de identificar posibles daños a las en las instalaciones eléctricas o penalizaciones efectuadas por alto factor de potencia. Igualmente pretende ser la base de futuras investigaciones relacionadas al monitoreo de los parámetros eléctricos en Universidad Tecnológica Indoamérica y beneficiando así a los administrativos teniendo así un método de comprobación de consumo y un seguro monitorio del sistema eléctrico.

Además del estudio anterior, se presenta otra investigación realizada por (Nascimento & Dantas Diógenes, 2020) se centra en la gestión eficiente de la energía, explorando la evolución histórica que han tenido los medidores de energía eléctrica convencionales y la aplicación de fundamentos teóricos; presentan un diseño de un prototipo innovador revelando la importancia de integrar conocimientos previos con avances tecnológicos actuales como IoT (Internet of Things) y plataformas en la nube. De esta forma, el prototipo no solo aporta una solución para el monitoreo de magnitudes eléctricas, también promueve la integración de teoría, práctica e innovación tecnológica para la búsqueda de soluciones eficientes y sostenibles en el manejo de la energía eléctrica.

Por otro lado, (Cayetano, 2020) menciona la importancia de la reducción de los componentes, el tamaño compacto y búsqueda de componentes económicos; para crear un prototipo innovador de esta forma se crea un dispositivo de bajo

costo y menos invasivo en el sistema eléctrico que se instale. De igual manera, promueve la utilización de las tecnologías modernas para crear dispositivos de medición, los cuales son una alternativa a los medidores convencionales. Plantea que al utilizar componentes electrónicos puede llegar a mejorar la precisión del dispositivo, de esa manera se logra garantizar un dispositivo de calidad y de alta precisión; además, de práctico y seguro.

4.2 Analizador de redes eléctricas trifásicas

Un analizador de redes trifásicas es un dispositivo con el cual se realiza un análisis de las propiedades de una instalación eléctrica. Además, verifica la capacidad de carga, ayuda a conocer el consumo, el voltaje y la sobretensión. Por lo tanto, su uso permite solucionar cualquier problema que haya en la red eléctrica, si se lleva un mantenimiento periódico, se pueden evitar riesgos y lograr un ahorro energético. (Distron, 2022) también menciona que el dispositivo tiene como objetivo examinar y ofrecer información sobre las características de un sistema eléctrico. Los analizadores de redes están diseñados para ser capaces de emplearse en cualquier tipo de instalación y poseen una memoria interna en la que se archivan los parámetros de medición. Un analizador como el de la figura 1 son capaces de exportar y/o mostrar los parámetros eléctricos y lo hacen de forma directa o indirecta a través de un navegador web, display, un programa, etc.

Fig. 1: Analizador de energía y de calidad eléctrica Fluke 437 Serie II (FLUKE , s.f.)

4.2.1 Ventajas de instalar un analizador de redes eléctricas

Según la investigación de (Castro, 2020), se presentan algunos beneficios consecuentes de la implementación de un analizador de redes:

- Ahorro de energía eléctrica:
  - Descubrir y evitar los excesos de consumo.
  - Análisis de curvas de carga para localizar los puntos de máxima demanda energética.
  - Detección de deficiencias en la instalación, como los problemas relacionados con el bajo factor de potencia.
  - Detección de fraudes en contadores energéticos.

- Prevención de riesgos en la red eléctrica:
  - Realización de mantenimientos periódicos de la red eléctrica. Miden curvas de arranques en motores, detección de saturación en transformadores de potencia, calidad insuficiente del suministro eléctrico.

- Solución de problemas en la red:
  - Con el uso de los analizadores de redes es posible solventar problemas de fugas diferenciales, disparos ocasionales, resonancias, armónicos, recalentamiento de cables, desequilibrio en las fases, etcétera.
  - Permite un diseño adecuado de los filtros activos y pasivos de armónicos, así como filtros de variadores de velocidad.

4.2.2 Funcionamiento analizador de redes

Para poder estimar la calidad, el flujo, la optimización y la cantidad de las redes eléctricas se emplean los llamados analizadores de red. Sin embargo, existen varios tipos los que se enlistarán más adelante. Es importante mencionar que estos instrumentos sirven para medir y testear las propiedades comentadas en una red eléctrica. (Rubio, 2019)

4.2.3 Aplicaciones de los analizadores de redes

El analizador de redes eléctricas es una herramienta utilizada en la industria eléctrica para medir y analizar diferentes parámetros relacionados con la calidad y el rendimiento de la energía eléctrica. Algunas de las aplicaciones del analizador de redes eléctricas son:

- Análisis de calidad de energía:
  - Este dispositivo se utiliza para monitorear y analizar la calidad de la energía eléctrica suministrada. Puede detectar problemas como armónicos, desequilibrios de voltaje y corriente, fluctuaciones de voltaje, caídas de tensión, interrupciones y otros eventos transitorios.

- Diagnóstico de problemas eléctricos:
  - El equipo es útil para diagnosticar problemas en sistemas eléctricos. Puede identificar problemas de carga desequilibrada, sobrecargas, pérdida de eficiencia energética, mal funcionamiento de equipos, problemas de conexión a tierra y otros problemas relacionados con la distribución y uso de la energía eléctrica.

- Cumplimiento de normas y regulaciones:
  - Este instrumento puede ayudar a garantizar el cumplimiento de las normas y regulaciones relacionadas con la calidad de la energía eléctrica. Permite realizar mediciones y generar informes que demuestren el cumplimiento de los estándares establecidos por organismos reguladores y normas técnicas.

- Optimización del consumo de energía:
  - Al analizar los datos recopilados, el analizador de redes eléctricas puede identificar oportunidades para optimizar el consumo de energía. Puede proporcionar información sobre patrones de carga, demanda máxima,

eficiencia energética de equipos y sistemas, y sugerir medidas para reducir el consumo y los costos de energía.

- Monitoreo de equipos y sistemas eléctricos:
  - El dispositivo se utiliza para monitorear el rendimiento de equipos y sistemas eléctricos en tiempo real. Puede detectar problemas de funcionamiento, desgaste excesivo, sobrecalentamiento, variaciones de voltaje y corriente, y proporcionar alertas tempranas para tomar acciones preventivas.

- Estudios de ingeniería eléctrica:
  - Como herramienta fundamental en los estudios de ingeniería eléctrica, permite recopilar datos para realizar análisis de carga, estudios de cortocircuito, análisis armónico, estudios de estabilidad de voltaje, estudios de coordinación de protecciones y otros análisis técnicos necesarios para el diseño y operación de sistemas eléctricos.

4.2.4 Tipos de analizadores de redes.

Los analizadores de redes se clasifican en tres tipos: Scalar Network Analyzer (SNA), Vector Network Analyzer (VNA) y analizadores de redes de señales grandes, que ayudan a medir tanto la fase como la magnitud.

- Analizador de redes escalares:
  - El término SNA significa analizador de redes escalares y es un analizador de redes de tipo Radio Frequency (RF) que se utiliza para medir simplemente las propiedades de amplitud de un Device Under Test (DUT) o dispositivo bajo prueba. Al contrario que un VNA, el SNA no mide tanto la fase como la amplitud del DUT. Un analizador de tipo escalar se utiliza principalmente para medir diferentes parámetros, como la pérdida de retorno o la relación de transmisión de potencia, que simplemente necesitan la medición de la magnitud de la señal a una frecuencia específica.

Fig. 2: Analizador de red Agilent E8362C (CEYUAN ELECTRONIC INSTRUMENT CO.,LTD., s.f.)

- Analizador vectorial de redes:
  - Los VNA se utilizan principalmente para probar las especificaciones de los componentes y también para verificar las simulaciones de diseño y confirmar que los sistemas y sus componentes funcionan correctamente o no. El VNA es una forma más práctica de analizador de redes de RF en comparación con el SNA, porque es capaz de determinar parámetros adicionales relativos al DUT. Estos analizadores no sólo miden la respuesta de amplitud, sino que también miden la fase. Por eso se llama a este analizador un analizador de redes automático, o bien un medidor de ganancia y fase.

Fig. 3: SNA5000A Series Vector Network Analyzer (SIGLENT, s.f.)

- Analizador de redes a gran escala:
  - El término LSNA significa Analizador de Redes de Gran Señal. Es muy especializado para los analizadores de redes de RF, ya que es capaz de investigar las características del dispositivo en condiciones de gran señal. Este analizador también es capaz de observar las no lineales y los

armónicos de una red en diferentes condiciones, así como proporcionar un análisis de funcionamiento completo. (Electrositio.com, 2019)

Fig. 4: PNA-X Network Analyzers (KEYSIGHT, s.f.)

4.3 Medidores de energía eléctrica

Debido a que nuestro prototipo de analizador de redes presentará ciertas características dentro de su funcionalidad similares a las de los medidores de energía eléctrica; presentaremos información de los equipos mencionados anteriormente.

4.3.1 Medidores para sistemas eléctricos trifásicos

Un contador trifásico es un dispositivo utilizado para medir y registrar el consumo de energía eléctrica en sistemas trifásicos. Estos sistemas generalmente se encuentran en redes de distribución de energía eléctrica industrial o en aplicaciones de gran escala.

Un contador trifásico registra y mide la cantidad de energía consumida en cada una de las tres fases. Está compuesto por tres circuitos de medición, cada uno mide la corriente y el voltaje en cada fase, además calcula la potencia consumida. Estos valores se suman para obtener la energía total consumida en el sistema.

4.3.2 Medidores trifásicos en Nicaragua

La compañía encargada de la distribución de energía eléctrica en Nicaragua (DISNORTE-DISSUR); luego de hacer el análisis del censo de carga y una vez determinado que la instalación requiere una alimentación trifásica, se encarga de proporcionar e instalar los medidores adecuados para este tipo de magnitudes.

Los medidores trifásicos que se instalan más comúnmente en Nicaragua son de la marca Schlumberger, como el que se muestra en la figura 5.

Fig. 5: Contador De Energía Trifásico - Marca Schlumberger (todocoleccion, s.f.)

4.4 Teoría básica de funcionamiento

La tarea crítica de un microcontrolador dentro de una aplicación de medición de electricidad es el cálculo preciso del voltaje RMS, la corriente RMS, la potencia activa, la potencia reactiva, la potencia aparente, energía activa y energía reactiva. Las energías activa y reactiva se utilizan para determinar el gasto energético y/o facturación.

4.4.1 Cálculo de la corriente y voltaje

Una fuente de tensión sinusoidal v alimentando una carga lineal producirá una corriente sinusoidal i. Para determinar las corrientes y tensiones se utilizan las siguientes ecuaciones:

$v(t) = V_m \cos(\omega_t + \theta_u)$ *Ecuación 1. Tensión instantánea*

$i(t) = I_m \cos(\omega_t + \theta_i)$ *Ecuación 2. Corriente instantánea*

donde Vm e Im son las amplitudes (o valores pico) y teta v y teta i son los ángulos de fase de la tensión y la corriente, respectivamente.

4.4.2 Cálculo de la potencia aparente (VA).

Es el producto de la tensión rms y la corriente rms. La potencia aparente se llama así porque aparentemente la potencia debería ser el producto voltaje-corriente. Está dada por:

$S = v_{rms} * I_{rms}$ *Ecuación 3. Potencia aparente*

También puede determinarse mediante la implementación de la potencia reactiva y reactiva, las cuales se explicarán a continuación.

$S = \sqrt{P^2 + Q^2}$ *Ecuación 4. Método alternativo de potencia aparente*

4.4.3 Cálculo de la potencia activa (W).

La potencia activa, también conocida como potencia real, es el valor medio de la potencia instantánea durante el intervalo del tiempo;

$p = v_{rms} * I_{rms}\, cos(\theta_v - \theta_i)$ *Ecuación 5. Potencia real*

La potencia real P es la potencia promedio en watts suministrada a una carga; es la única potencia útil. Es la verdadera potencia disipada en la carga.

4.4.4 Cálculo de la potencia reactiva (VAR).

La potencia reactiva Q es una medida del intercambio de energía entre la fuente y la parte reactiva de la carga. La unidad de Q es el volt-ampere reactivo (VAR), la cual se determina con la siguiente ecuación:

$Q = v_{rms} * I_{rms}\, Seno(\theta_v - \theta_i)$ *Ecuación 6. Potencia reactiva*

4.4.5 Factor de potencia.

Es el coseno de la diferencia de fase entre la tensión y la corriente. También es igual al coseno del ángulo de la impedancia de la carga. El valor del fp va de cero a la unidad. En el caso de una carga puramente resistiva, la tensión y la corriente están en fase, de modo que $\theta_v - \theta_i$ y fp 1. Esto implica que la potencia aparente es igual a la potencia promedio. En el caso de una carga puramente reactiva, $\theta_v - \theta_i = \pm 90°$ y fp 0. En esta circunstancia la potencia promedio es de cero. Entre estos dos casos extremos, se dice que el fp está adelantado o atrasado. Un factor

de potencia adelantado significa que la corriente se adelanta a la tensión, lo cual implica una carga capacitiva. Un factor de potencia atrasado significa que la corriente se atrasa de la tensión, lo que implica una carga inductiva. El factor de potencia afecta las cuentas de electricidad que pagan los consumidores a las compañías suministradoras. Entonces el factor de potencia se calcula: (Alexander & Sadiku, 2006)

$fp = \frac{P}{S} = cos(\theta_v - \theta_i)$ Ecuación 7. Principal relación con el factor de potencia

4.5 El internet de las cosas (IoT): conceptos básicos y funcionamiento

El término IoT, se refiere a la red colectiva de dispositivos conectados y a la tecnología que facilita la comunicación entre los dispositivos y la nube, así como entre los propios dispositivos. Gracias a la llegada de los chips de computador de bajo costo y a las telecomunicaciones de gran ancho de banda, ahora tenemos miles de millones de dispositivos conectados a Internet. Esto significa que los dispositivos de uso diario, como los cepillos de dientes, las aspiradoras, los coches y las máquinas, pueden utilizar sensores para recopilar datos y responder de forma inteligente a los usuarios. (aws.amazon, s.f.) Según lo expresado anteriormente, el prototipo se relacionará directamente con el Internet de las cosas debido a que implica la conexión de un microcontrolador a una red Wi-Fi y la comunicación con base de datos para almacenar y visualizar los datos de energía en tiempo real.

Los ingenieros en informática llevan agregando sensores y procesadores a los objetos cotidianos desde los años 90. Sin embargo, el progreso fue inicialmente lento porque los chips eran grandes y voluminosos. Los chips para computadoras de baja potencia se utilizaron por primera vez para el seguimiento de equipos caros. A medida que se reducía el tamaño de los dispositivos informáticos, estos chips también se hacían más pequeños, más rápidos y más inteligentes.

El costo de la integración de la potencia de computación en objetos pequeños se redujo en gran medida. Por ejemplo, es posible agregar conectividad por medio de las capacidades de los servicios de voz de Alexa a las MCU con menos de

1MB de RAM integrada, como en el caso de los interruptores de luz. Surgió todo un sector con el objetivo de llenar nuestros hogares, empresas y oficinas de dispositivos de IoT. Estos objetos inteligentes pueden transmitir automáticamente datos hacia y desde Internet. Todos estos “dispositivos de computación invisibles” y la tecnología asociada a ellos se denominan de manera colectiva Internet de las cosas.

4.5.1 Conceptos básicos

El IoT involucra varios termino y conceptos que pueden ocasionar confusión o simplemente pueden ser desconocidos. Se presentarán varios temas que es de vital importancia conocer al respecto para comprender el contenido del proyecto.

4.5.1.1 Redes de comunicación

Las redes de comunicación son esenciales para el funcionamiento del IoT, gracias a estas los dispositivos, sensores y periféricos con la capacidad de conectarse a internet; son capaces de comunicarse entre ellos. Aumentado las posibilidades y aplicaciones de algunos componentes, los cuales estarían limitados si no se conectasen a otros elementos. Esto es posible debido a la presencia del WiFi, se hablará un poco más del término a continuación.

4.5.1.2 Wi-Fi

Wi-Fi es una tecnología de redes inalámbricas que utiliza ondas de radio para proporcionar acceso a internet inalámbrico de alta velocidad. Un error común es pensar que el término Wi-Fi es la abreviatura de “fidelidad inalámbrica”, pero Wi-Fi es una frase registrada que hace referencia a los estándares IEEE 802.11x.
El Wi-Fi se originó en Hawái en 1971, cuando se utilizó una red de paquetes UHF inalámbrica llamada ALOHAnet para conectar las islas. Los protocolos posteriores desarrollados en 1991 por NCR y AT&T, denominados WaveLAN, se convirtieron en los precursores de los estándares IEEE 802.11.

La Wi-Fi Alliance se formó en 1999 y actualmente posee la marca registrada Wi-Fi. Define específicamente Wi-Fi como cualquier “producto de red de área local

inalámbrica (WLAN) que se base en los estándares 802.11 del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE)".

Inicialmente, Wi-Fi se utilizó en lugar únicamente del estándar 802.11b de 2,4 GHz, sin embargo, la Wi-Fi Alliance ha ampliado el uso genérico del término Wi-Fi para incluir cualquier tipo de red o producto WLAN basado en cualquiera de los estándares 802.11, incluidos 802.11b, 802.11a, etc. en un intento de detener la confusión sobre la interoperabilidad de las redes LAN inalámbricas. (Beal, 2022)

4.5.1.3 Sensores

Según (Merriam-Webster, s.f.) sensor se puede definir como un dispositivo que responde a un estímulo físico (como calor, luz, sonido, presión, magnetismo o un movimiento particular) y transmite un impulso resultante (como para medir u operar un control). Hoy en día los sensores se utilizan en todas partes como en alarmas, radares y en muchos dispositivos electrónicos.

Con la constante actualización de la industria se utilizan más los sensores, ya que los procesos deben ser precisos y seguros. En la industria 4.0 se utilizan sensores inteligentes los cuales ya tienen incorporado microprocesadores los cuales, en conjunto con actuadores, son capaces de realizar grandes procesos. Además, se utilizan para monitorizar estos mismos procesos, permitiendo un seguimiento continuo de las condiciones de producción, ayudando de esta manera a la prevención de fallas o problemas.

4.5.2 Funcionamiento del IoT

Un sistema común de IoT funciona mediante la recopilación y el intercambio de datos en tiempo real. Un sistema del IoT tiene tres componentes:

- Dispositivos inteligentes:

Se trata de dispositivos, como un televisor, una cámara de seguridad o un equipo de ejercicio, a los que se les dotó de capacidades de computación. Recopila datos

de su entorno, de las entradas de los usuarios o de los patrones de uso y comunica los datos a través de Internet hacia y desde su aplicación de IoT.

- Aplicación de IoT:

Una aplicación de IoT es un conjunto de servicios y software que integra los datos recibidos de varios dispositivos de IoT. Utiliza tecnología de machine learning o inteligencia artificial (IA) para analizar estos datos y tomar decisiones informadas. Estas decisiones se comunican al dispositivo de IoT y este responde de forma inteligente a las entradas.

- Una interfaz de usuario gráfica:

El dispositivo de IoT o la flota de dispositivos pueden administrarse a través de una interfaz de usuario gráfica. Algunos ejemplos comunes son una aplicación móvil o un sitio web que pueden utilizarse para registrar y controlar dispositivos inteligentes.

4.6 Arduino IDE como entorno de programación

Arduino es una plataforma electrónica de código abierto basada en hardware y software fáciles de usar. Las placas Arduino pueden leer entradas (luz en un sensor, un dedo en un botón, etc.) y convertirlas en una salida (activar un motor, encender un LED, publicar algo en línea, etc.). Puedes indicarle a tu placa qué hacer enviando un conjunto de instrucciones al microcontrolador de la placa. Para ello, utiliza el lenguaje de programación Arduino (basado en Wiring) y el software Arduino (IDE), basado en Processing. (Arduino IDE, 2018)

Para el desarrollo del código de programación del prototipo se usó Arduino IDE gracias a su sencillez (en comparación a otros entornos de programación) y accesibilidad a las bibliotecas necesarias, lo que lo hace perfecto para desarrollar un programa que cumpla con las capacidades de nuestro prototipo.

4.7 InfluxDB Cloud como base de datos

InfluxDB Cloud es una plataforma diseñada específicamente para recopilar, almacenar, procesar y visualizar datos de series temporales. Los datos de series temporales son una secuencia de puntos de datos indexados en orden cronológico. Los puntos de datos suelen consistir en mediciones sucesivas realizadas a partir de la misma fuente y se utilizan para realizar un seguimiento de los cambios a lo largo del tiempo.

4.7.1 Conceptos claves para Organización de datos

El modelo de datos de InfluxDB organiza los datos de series temporales en grupos y mediciones. Un grupo puede contener varias mediciones. Las mediciones contienen varias etiquetas y campos.

- Bucket: ubicación con nombre donde se almacenan los datos de series temporales. Un bucket puede contener varias mediciones.
- Medición: Agrupamiento lógico de datos de series temporales. Todos los puntos de una medición dada deben tener las mismas etiquetas. Una medición contiene múltiples etiquetas y campos.
- Etiquetas: pares clave-valor con valores que difieren, pero que no cambian con frecuencia. Las etiquetas están pensadas para almacenar metadatos de cada punto; por ejemplo, algo que identifique la fuente de los datos, como el host, la ubicación, la estación, etc.
- Campos: pares clave-valor con valores que cambian con el tiempo, por ejemplo: temperatura, presión, precio de las acciones, etc.

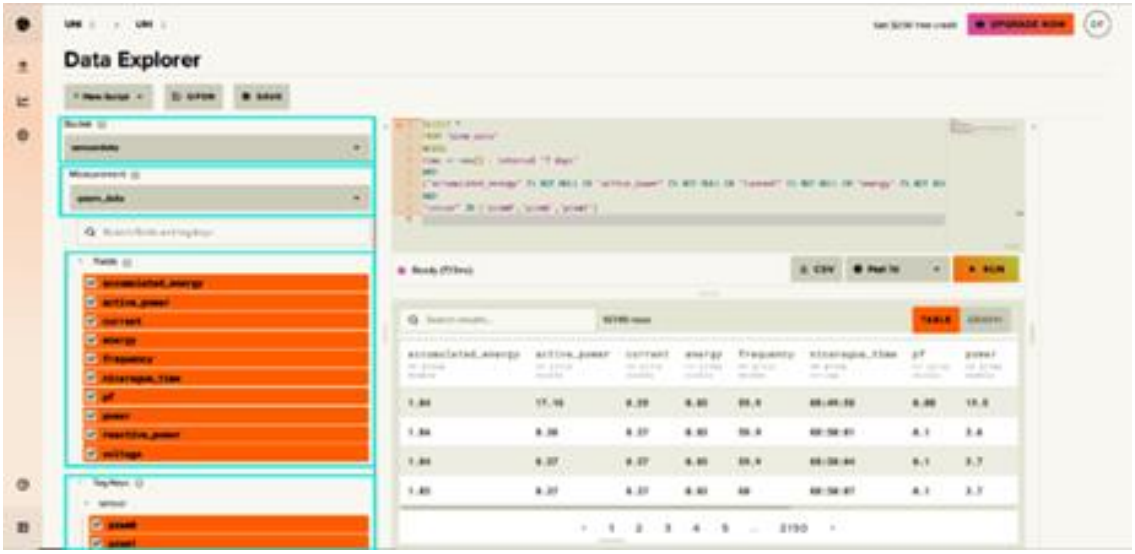


Fig. 6: Ejemplo de consultas del prototipo en InfluxDB Cloud (Fuente Propia)

- Marca de tiempo: marca de tiempo asociada a los datos. Cuando se almacenan en el disco y se consultan, todos los datos se ordenan por hora.
- Punto: Registro de datos único identificado por su medición, claves de etiqueta, valores de etiqueta, clave de campo y marca de tiempo.
- Serie: Un grupo de puntos con la misma medida, claves de etiqueta y valores, y clave de campo.

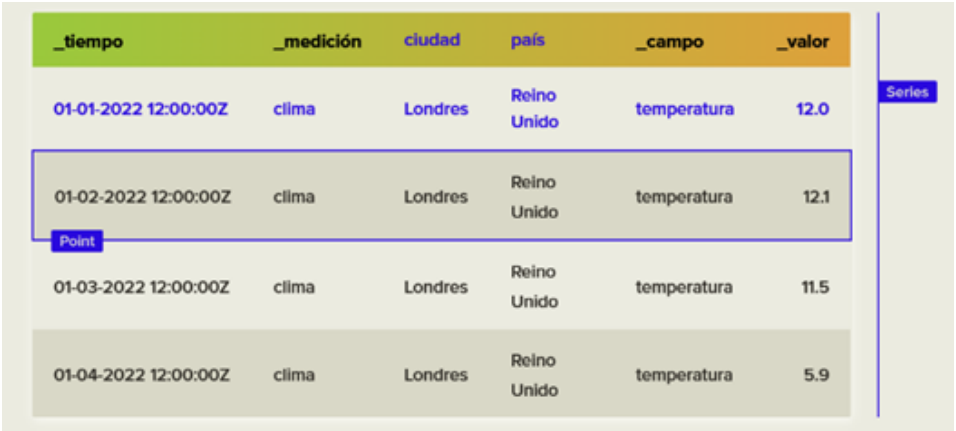


Fig. 7: Ejemplo de resultados de consultas en InfluxDB Cloud (InfluxData, Inc., 2025)

4.7.2 Beneficios de la implementación de InfluxDB Cloud

En nuestro caso se ha seleccionado a InfluxDB Cloud como la herramienta IoT para almacenar toda la información obtenida del prototipo debido a los beneficios

que ofrece incluso en su versión gratuita, comparada con otros servicios véase en **Anexo 1**

Su versión gratuita, en los aspectos de interés ofrece Escrituras de 5MB/5 mins, Consultas / Lecturas de 300MB/5mins, 2 Buckets (Bases de datos) con una retención total de datos máximos de 30 días, así mismo cumple con estándares de control de la seguridad de la información tales como SOC 2 Type 2 y ISO 27001- Stage 1 (cruz, 2023)

Igualmente es una plataforma autogestionada que permanece en constante revisión para detectar cualquier tipo de interrupción o incidente, gestionándolo de la mejor manera.

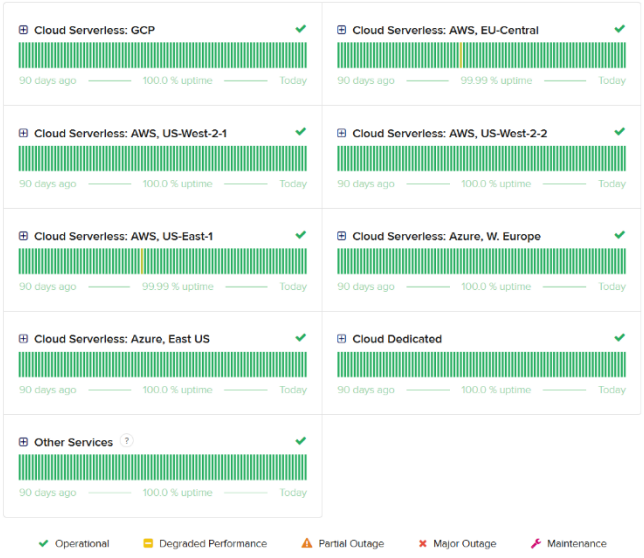


Fig. 8: ilustración del estado de nubes de InfluxDB cloud (InfluxData, Inc., 2025)

Toda la información es enviada a una ubicación central en la nube, que permite acceder fácilmente a sus datos para el análisis en línea con su propia interfaz intuitiva. una vez accediendo a la plataforma puede descargar de forma segura los datos almacenados en la nube en formatos CSV (Comma Separated Values). Con el fin de que el usuario pueda analizar los datos a través de la herramienta

Excel. Además, ambos métodos permiten organizar o elegir el tipo la cantidad de información a analizar delimitado por el rango de tiempo o por las variables escogidas.

Además de contar con un conjunto de potentes bibliotecas de clientes tales como Arduino, C#, Go, Java, JavaScript, Kotlin, Node.js, PHP, Python, R, Ruby, Scala y Swift. Se puede acceder fácilmente a estas bibliotecas de cliente en una nueva pestaña de la interfaz de usuario los que ofrece una amplia variedad de entornos para poder desarrollar un prototipo.

La plataforma también brinda un excelente manejo al Recopilar, analizar y almacenar datos de métricas de alta frecuencia sin que se vea afectada en gran medida el rendimiento de la plataforma.

Su compatibilidad con SQL nativo lo que facilita la curva de aprendizaje y optimizando el tiempo para realizar cualquier tipo de consulta. paralelo a que su editor de scripts se convierte en más visual e intuitivo para el usuario.

Por último, su interoperabilidad con ecosistemas dispone de complementos útiles para acceder a la base de datos contenidas en esta, con el fin de poder armonizar el almacenamiento de métricas con plataformas de visualización de datos en nuestro caso Grafana permite el monitoreo de la base de datos con alto desempeño de la cual hablaremos más adelante. (InfluxData, Inc., 2025)

4.8 Grafana cloud como herramienta de monitoreo

Grafana Cloud es una plataforma de observación de alto rendimiento, escalable y de alta disponibilidad para sus aplicaciones e infraestructura. Proporciona una vista centralizada de todos sus datos de observación, ya sea que los datos residan en los servicios de métricas de Grafana Cloud o en sus propios entornos físicos y en la nube. Con soporte nativo para muchas fuentes de datos populares como en nuestro caso que es InfluxDB cloud. todo lo que se tiene que hacer para comenzar es crear paneles y consultar datos de métricas configurando las fuentes de datos en Grafana Cloud. (Grafana Labs, 2025)

4.8.1 Conceptos claves para visualización de datos.

4.8.1.1 Métricas

Las métricas nos indican cuánto existe de algo, como por ejemplo cuánta memoria tiene disponible un sistema informático o cuántos centímetros de largo tiene un escritorio. En el caso de Grafana, las métricas son más útiles cuando se registran repetidamente a lo largo del tiempo. Esto nos permite comparar aspectos como la forma en que la ejecución de un programa afecta la disponibilidad de los recursos del sistema. (Grafana Labs, 2025)

4.8.1.2 Un panel de control de Grafana

Consta de paneles que muestran datos en gráficos, diagramas y otras visualizaciones atractivas. Estos paneles se crean utilizando componentes que transforman datos sin procesar de una fuente de datos en visualizaciones. El proceso implica pasar datos a través de tres puertas: un complemento, una consulta y una transformación opcional.

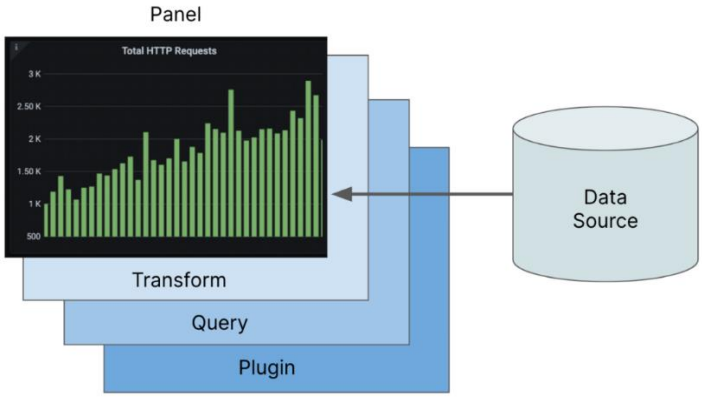


Fig. 9 Arquitectura de componentes de tablero de mando (Grafana Labs, 2025)

4.8.1.3 Fuentes de datos

Una fuente de datos hace referencia a cualquier entidad que conste de datos. Puede ser una base de datos SQL, Grafana Loki, Grafana Mimir o una API basada en JSON. Incluso puede ser un archivo CSV básico. El primer paso para crear

una visualización de panel es seleccionar la fuente de datos que contiene los datos que necesita.

4.8.1.4 Complementos

Un complemento de Grafana es un software que agrega nuevas capacidades a Grafana y actúan como intermediarios para facilitar la conexión de diferentes sistemas de almacenamientos de datos, su principal función es traducir las consultas escritas en un lenguaje compatible con la fuente de datos y convertir los datos en un formato que Grafana pueda usar.

4.8.1.5 Consultas

Las consultas en Grafana permiten filtra y reducir las métricas que se visualizan en datos específicos y manejables, creando una visualización útil y especifica.

4.8.1.6 Transformación

Brinda la flexibilidad de modificar los datos para que se ajusten a las necesidades antes de mostrarlo en el panel, de tal manera que podría combinar datos de 2 tipos de campos y mostrarlo en un único y nuevo campo, además que permitir ordenar los datos de mayor a menor solo con el soporte propio de Grafana entre muchas más funcionalidades.

4.8.1.7 Paneles

Una vez que se obtienen los datos, se los consulta y se los transforma, pasan a un panel, que es la última puerta de entrada a una visualización de Grafana. Un panel es un contenedor que muestra la visualización y le proporciona varios controles para manipularla. La configuración del panel es donde especifica cómo desea ver los datos. Por ejemplo, puede utilizar un menú desplegable en la parte superior derecha del panel para especificar el tipo de visualización que desea ver.

Algunos de los Gráficos y diagramas que ofrece Grafana son:

- La visualización gráfica principal y predeterminada es la serie temporal. Este panel generalmente usado para revisar tendencias.

- ❖ Cronología estatal de los cambios estatales a lo largo del tiempo.
- ❖ Historial de estado para estados periódicos a lo largo del tiempo.
- ❖ El gráfico de barras muestra cualquier dato categórico.
- ❖ El histograma calcula y muestra la distribución de valores en un gráfico de barras.
- ❖ El mapa de calor visualiza datos en dos dimensiones y se utiliza normalmente para medir la magnitud de un fenómeno, entre muchas otras.

Fig. 10: ilustración de ejemplo de panel de control Grafana cloud (Grafana Labs, 2025)

4.8.2 Serie temporal

Para la finalidad de nuestro proyecto optamos por escoger la visualización gráfica principal serie temporal debido a que son la forma predeterminada de mostrar las variaciones de un conjunto de valores de datos a lo largo del tiempo. Cada punto de datos se corresponde con una marca de tiempo y esta serie temporal se muestra como un gráfico. La visualización puede representar series como líneas, puntos o barras y es lo suficientemente versátil como para mostrar casi cualquier tipo de datos de series temporales.

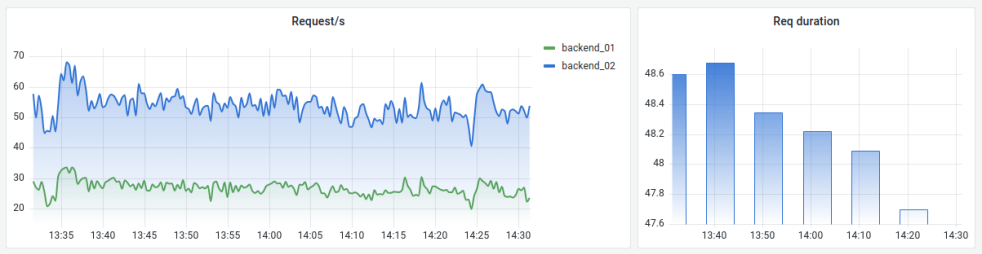


Fig. 11: ilustración de ejemplo de la visualización gráfica serie temporal Grafana cloud (Grafana Labs, 2025)

Una visualización de series temporales muestra un gráfico xy con la progresión temporal en el eje x y la magnitud de los valores en el eje y. Esta visualización es ideal para mostrar una gran cantidad de puntos de datos cronometrados que serían difíciles de rastrear en una tabla o lista (Grafana Labs, 2025)

4.8.2 Beneficios de la implementación de Grafana Cloud

Entre los beneficios principales de permite visualizaciones altamente personalizables, lo que permite una excelente adaptabilidad a cualquier tipo de presentación de datos, la capacidad de integrarse con un gran cantidad de fuentes de datos, se puede visualizar de manera muy sencilla en el navegador, el plan gratuito es para siempre, cuenta con un ecosistema y comunidad que aporta con complementos útiles para agregar funcionalidad, además de contar con su propia documentación para aprender a usar el interfaz y las diversas herramientas, Los paneles y las visualizaciones se pueden compartirse fácilmente entre los miembros del equipo o de forma pública con Grafana, además también posee ofrece funciones de seguridad como interfaz de usuario administrativa, cumplimiento normativo (SOC2, ISO 27001, PCI, GDPR, CSA e HIPAA) y gestión de usuarios y permisos Garantizando la protección de los datos confidenciales. (Edge Delta, 2024)

Los limites general del plan gratuito de Grafana cloud recaen en la cantidad de usuarios activos que expresamente son 3 pero no hay un límite explicito en la cantidad de consultas que puede hacer grafana cloud a una una fuente de datos externa como InfluxDB Cloud, por lo que cualquier restricción estarán definidas

por la capacidad de la cuenta de InfluxDB Cloud y no por grafana cloud. (Grafana Labs, 2025)

4.9 Componentes del prototipo.

A continuación, se presentan los componentes necesarios para el correcto funcionamiento del prototipo de analizador de redes trifásico.

4.9.1 Microprocesador ESP-WROOM-32

Se ha elegido el ESP-WROOM-32 como el dispositivo de control para el prototipo del analizador de redes trifásico, debido a sus destacadas características. Este microprocesador maneja hasta 32 bit, tiene un doble núcleo y está fabricado con sensores de WiFi y Bluetooth. Además, su equilibrio entre capacidad de procesamiento, conectividad inalámbrica integrada y bajo consumo energético. A diferencia de la Raspberry Pi, que necesita un sistema operativo y consume más energía; por otro lado, el ESP-WROOM-32 es más eficiente para aplicaciones en tiempo real. Comparado con los microcontroladores PIC y las placas Arduino, el ESP-WROOM-32 ofrece una integración más sencilla con redes WiFi y Bluetooth, lo que facilita la implementación de sistemas de monitoreo en la nube. Además, su bajo costo y disponibilidad lo convierten en la mejor opción para un sistema de monitorización de parámetros eléctricos. En el **Anexo 2** se especifican las características, ventajas y desventajas de los microprocesadores antes mencionados.

El ESP-32 es una serie de System on Chip (SoC) y módulos de bajo costo y bajo consumo de energía creado por Espressif Systems. Esta nueva familia es la sucesora del conocido ESP8266 y su característica más notable es que, además de Wi-Fi, también soporta Bluetooth. (Carmenate, s.f.) El microprocesador que se muestra en la Figura 12, es el encargado de gestionar todos los procesos que realiza el prototipo, gracias a sus características es capaz de realizar múltiples procesos al mismo tiempo. Para más información revisar ficha técnica en **Anexo 3**, además revisar el pin-out en **Anexo 4.**

Fig. 12: Ilustración del microprocesador ESP-WROOM-32 (Fuente Propia)

El ESP-32 es capaz de realizar múltiples tareas a la vez, gracias a que está fabricado con dos núcleos, los cuales pueden soportar el sistema operativo en tiempo real libre (por sus siglas en inglés, FreeRTOS). Este sistema operativo dota de algunos beneficios en los procesamientos del microprocesador, puede dividir las tareas en unidades más pequeñas, permitiéndole administrar múltiples tareas al mismo tiempo ampliando aún más las capacidades del ESP-32.

4.9.1.1 Solución al Problema de Reconocimiento del ESP-32 en Arduino IDE

Normalmente se deben hacer algunos procedimientos previos a la ejecución de programas en el ESP-32 mediante Arduino IDE. Primero se debe agregar el siguiente URL en la casilla de preferencias en los ajustes del programa (véase figura 13): https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json. Esto le proporcionará el paquete de bibliotecas más utilizadas en el ESP-32, que serán útiles para el proyecto.

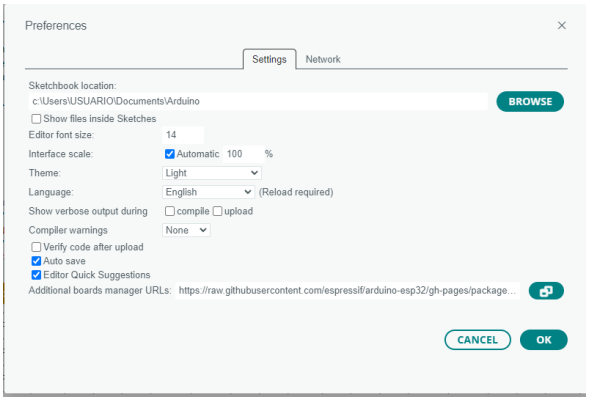


Fig. 13: Añadiendo el ESP-32 a Arduino IDE. (Fuente Propia)

Luego, diríjase al gestor de tarjetas para instalar la información del link como se muestra en la figura 14.

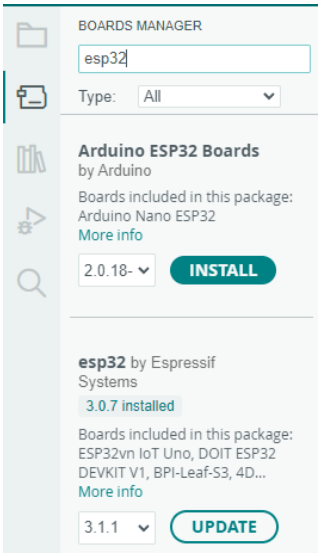


Fig. 14: Instalación de la placa ESP-32 en Arduino IDE (Fuente Propia)

Por último, para que la computadora detecte y sea capaz de interactuar con el microprocesador se requiere de un driver llamado “CP210x” de Silicon Labs, éste se descarga y se ejecuta como administrador. Se puede confirmar su correcta instalación en el administrador de dispositivos de la computadora, debería de mostrarse en el apartado “Puertos (COM y LPT)” un driver llamado “Silicon Labs CP210x USB to UART Bridge”.

4.9.2 Sensores PZEM-004T

El módulo de medidor multifunción permite medir el voltaje RMS, corriente RMS, potencia activa y energía que toma una carga conectada a una línea monofásica de 120 / 240V con límite superior hasta 260 V AC. (ssdielect electronica sas, s.f.) Tiene como función principal censar los parámetros eléctricos esenciales para realizar las debidas conversiones para calcular los parámetros restantes. Véase la hoja de datos en **Anexo 5**.

Fig. 15: Ilustración del sensor PZEM-004T (Innovators Guru, 2019)

El sensor PZEM-004T tiene una comunicación basada en el protocolo Modbus RTU, a través de una interfaz UART con la conexión física de los pines RX y TX. Para su correcto funcionamiento se necesita la biblioteca llamada “PZEM004Tv30.h”, la cual se debe buscar en repertorios en internet como GitHub e instalarse correctamente en Arduino IDE, de esta manera se garantiza la ejecución exitosa.

Para el correcto funcionamiento de varios PZEM conectados en paralelo con los pines 16 y 17 del ESP32 se debe de asignar una dirección distinta para cada uno que se conecte, evitando así un choque en la transmisión y recepción de datos. Esta manera se le indica al ESP32 cuales son los sensores que están conectados y funcionando correctamente con un nombre diferente.

4.9.3 Transformador de corriente

Transformador de corriente de núcleo partido 300 A con relación 3000/1 EARU ELÉCTRIC. A pesar de que el PZEM-004T tiene por defecto un transformador de corriente toroidal de 100 A con relación 1000/1, se decidió cambiarlo por el que se observa en la Figura 16 y descrito anteriormente, con el objetivo de ampliar el margen de utilización del prototipo ya que tiene mayor rango de medición y facilidad de instalación.

Fig. 16: Ilustración del transformador de corriente de núcleo partido de 300 A (Fuente Propia)

4.9.4 Caja MAKERELE

Caja MAKERELE para conexiones impermeable con cubierta transparente, IP67 Cajas de proyectos impermeables a prueba de polvo para electrónica 11.4 × 7.5 × 5.5 pulgadas（290 × 190 × 140mm). Componente esencial para la seguridad y orden del prototipo esto asegura un funcionamiento sin interrupciones debidas a problemas con el hardware, de esta manera se reduce la posibilidad de manipulación no autorizada.

Fig. 17: Caja para las conexiones (Fuente Propia)

4.9.5 Adaptador Tarjeta SD

La placa de circuito de conversión de nivel que puede interconectar el nivel es de 5 V o 3,3 V. Elemento importante para el almacenamiento de todos los datos medidos. Más información de sus especificaciones en **Anexo 6**.

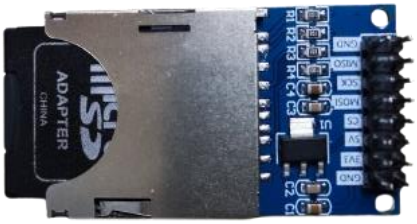


Fig. 18: Adaptador de Tarjeta SD (Fuente Propia)

El adaptador funciona mediante un protocolo SPI (Serial Peripheral Interface) el cual requiere que estén conectados los siguientes pines: MOSI (Master Out, Slave In), MISO (Master In, Slave Out), SCK (Serial Clock) y CS (Chip Select). Además, requiere de las bibliotecas “SD.h” y “SPI.h”, estas bibliotecas se encuentran en el repertorio interno del Arduino IDE.

4.9.6 Microprocesador Arduino Mega 2560

Arduino Mega es una tarjeta de desarrollo open-source (código abierto) construida con un microcontrolador modelo Atmega2560 que posee pines de entradas y salidas (E/S), analógicas y digitales. Utilizado para el procesar los datos censados y enviarlos a la pantalla, de esta forma el ESP32 no se sobre carga con otro proceso más a realizar. Véase la hoja de datos en **Anexo 7 y Anexo 8**.

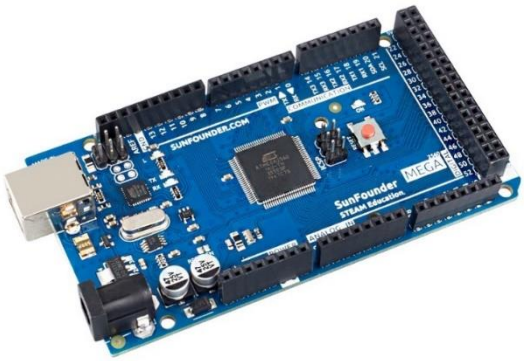


Fig. 19: Microprocesador Arduino Mega 2560 (Sunfounder, s.f.)

4.9.7 Pantalla LCD

Pantalla 2.4´´ TFT LCD shield, 240 x 320 pixels. Pantalla táctil resistente viene preinstalada con el módulo como beneficio adicional, por lo que puede detectar fácilmente las pulsaciones de sus dedos en cualquier parte de la pantalla. (OPL Display, 2023). Diseñada para conectarse directamente al Mega2560 el cual la controla para mostrar los datos censados en tiempo real aun cuando no esté conectado a internet.

Fig. 20: 2.4” TFT LCD Shield (Fuente Propia)

Se comunica a través del protocolo SPI explicado anteriormente, por tal motivo utiliza la misma biblioteca “SPI.h”. Debido a que se conecta sobre el Mega 2560 no hay necesidad de conectar mediante cableado, pero a continuación se especifican los pines necesarios para su funcionamiento: MOSI (Master Out, Slave In), MISO (Master In, Slave Out), SCK (Serial Clock), CS (Chip Select), DC (Data/Command) y RST (Reset).

4.9.8 Reloj en Tiempo Real

Reloj en tiempo real RTC DS3231. Necesario para poder sincronizar la hora de las lecturas, se asegura que la hora en que se almacenan los datos sea la correcta y no se desincronice con otros procesos. Más características y especificaciones del dispositivo en **Anexo 9**.

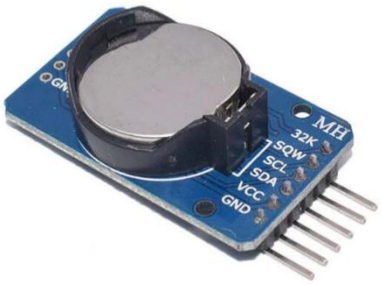


Fig. 21: RTC DS3231 (Electromer, s.f.)

El módulo RTC funciona conectando los pines SCL (Serial Clock Line) y SDA (Serial Data Line). Se necesitan las bibliotecas llamadas “Wire.h” y “RTClib.h”.

4.9.9 Tarjeta PCB Perforada

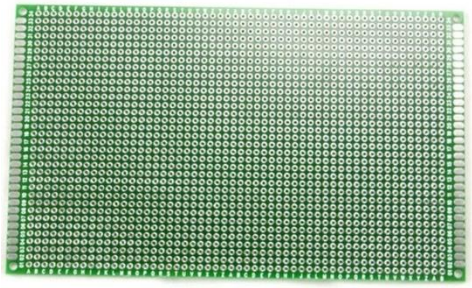


Fig. 22: Tarjeta PCB de baquelita perforada (fuente propia)

En la figura 22 se muestra una tabla de nodos para las conexiones de algunos componentes los cuales tienen puntos en común con el ESP y no se pueden conectar directamente.

4.9.10 Cables para conexiones

Cables jumpers para facilitar la conexión entre los microprocesadores y sensores del prototipo.

Fig. 23: Cables jumpers. (Fuente Propia)

5. FUNCIONAMIENTO DEL PROTOTIPO

5.1 Diagrama de Flujo

A continuación, se muestra el funcionamiento del dispositivo mediante la realización de un diagrama de flujo. De esta manera, se logra interiorizar más en el comportamiento de cada uno de los periféricos y su importancia de uso en el proyecto.

El ESP-32 comienza iniciando la comunicación con el Mega y con los PZEM ya que estos se conectan con el protocolo UART de comunicación serial. Además, inicia la comunicación con el RTC mediante la el tipo de comunicación serial llamado I2C (Inter-Integrated Circuit) verificando y ajustando la hora si es necesario. Lee el delay almacenado en la memoria no volátil, continúa inicializando las credenciales de Wi-Fi con las cuales intenta conectarse a internet automáticamente (si no se conecta, envía mensaje de error de conexión), una vez conectado se sincroniza el Real Time Clock (RTC) con el servidor Network Time Protocol (NTP). Seguido, inicializa los sensores PZEM y una cola para los datos censados. Luego, permite configurar el servidor web dando la posibilidad de ajustar el tiempo de censado y conectar a una red Wi-Fi. Continua, creando las tareas “tareasSensores” para leer y guardar los datos censados por los PZEM y “tareasInflux” para enviar los datos a InfluxDB Cloud. Al final, en el Loop Principal, maneja las peticiones del servidor web y muestra las mediciones cuando no está conectado a internet. (Véase la figura 24).

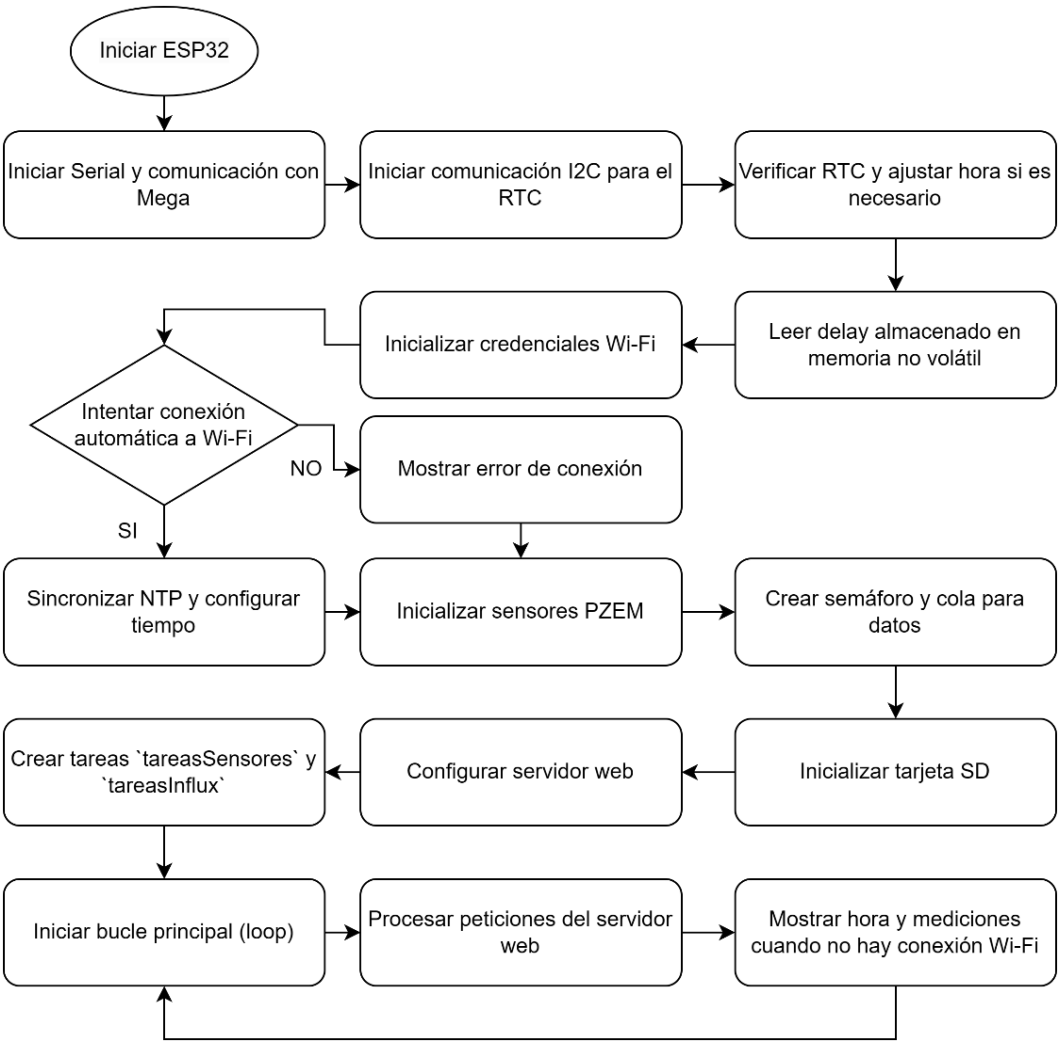


Fig. 24: Diagrama de flujo procesamiento ESP-32. (Fuente Propia)

Como se ha expresado anteriormente, uno de los procesos del ESP-32 es crear tareas tanto para InfluxDB (véase figura 25) como para los sensores PZEM-004T (véase figura 26). En el caso de tareasInflux, comienza creando la estructura de DatosSensor y entrando en un bucle el cual espera los datos de la cola, procede a leer los datos de la cola los cuales son los datos de los sensores. Luego, verifica la conexión al Wi-Fi, en un dado caso que no esté conectado a Wi-Fi; se mostrarán los datos y en pantalla digital. En caso de que si esté conectado a Wi-Fi, obtendrá la hora y fecha actual, se creará el formato de datos requerido por InfluxDB (en este caso payload) para el envío de datos desde el ESP-32 y la recepción de datos en InfluxDB.

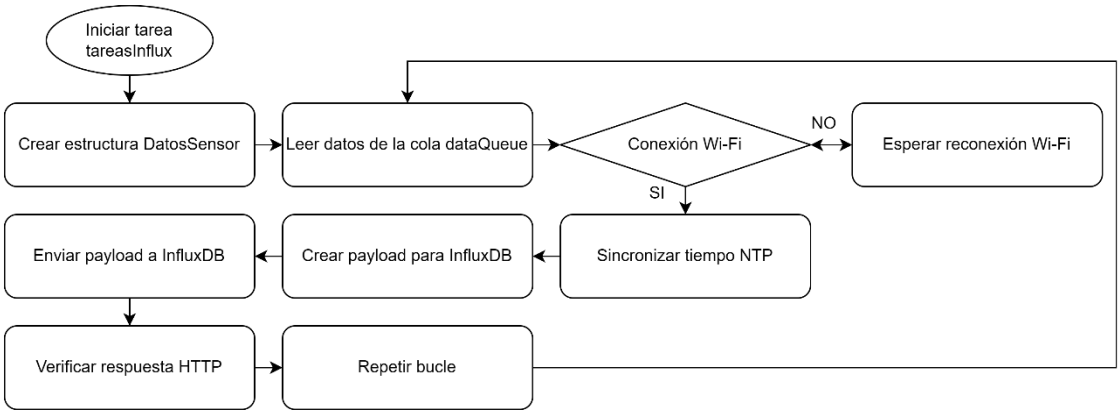


Fig. 25: Diagrama de flujo "tareasInflux" (continuación ESP-32). (Fuente Propia)

Al mismo tiempo, se crea las tareasSensores donde el ESP-32, obtiene la fecha y hora del RTC, inicia un bucle para que lea los datos de los sensores; luego los procesa para asegurar que estos sean valores válidos para calcular los parámetros eléctricos restantes como la energía (kW/h) y las potencias (Aparente, Activa y Reactiva). Una vez se calculan los parámetros, se envían a cola para enviarlos al Mega 2560 (para mostrar en pantalla) y al adaptador SD (para almacenar los datos en un archivo CSV. (Véase figura 26).

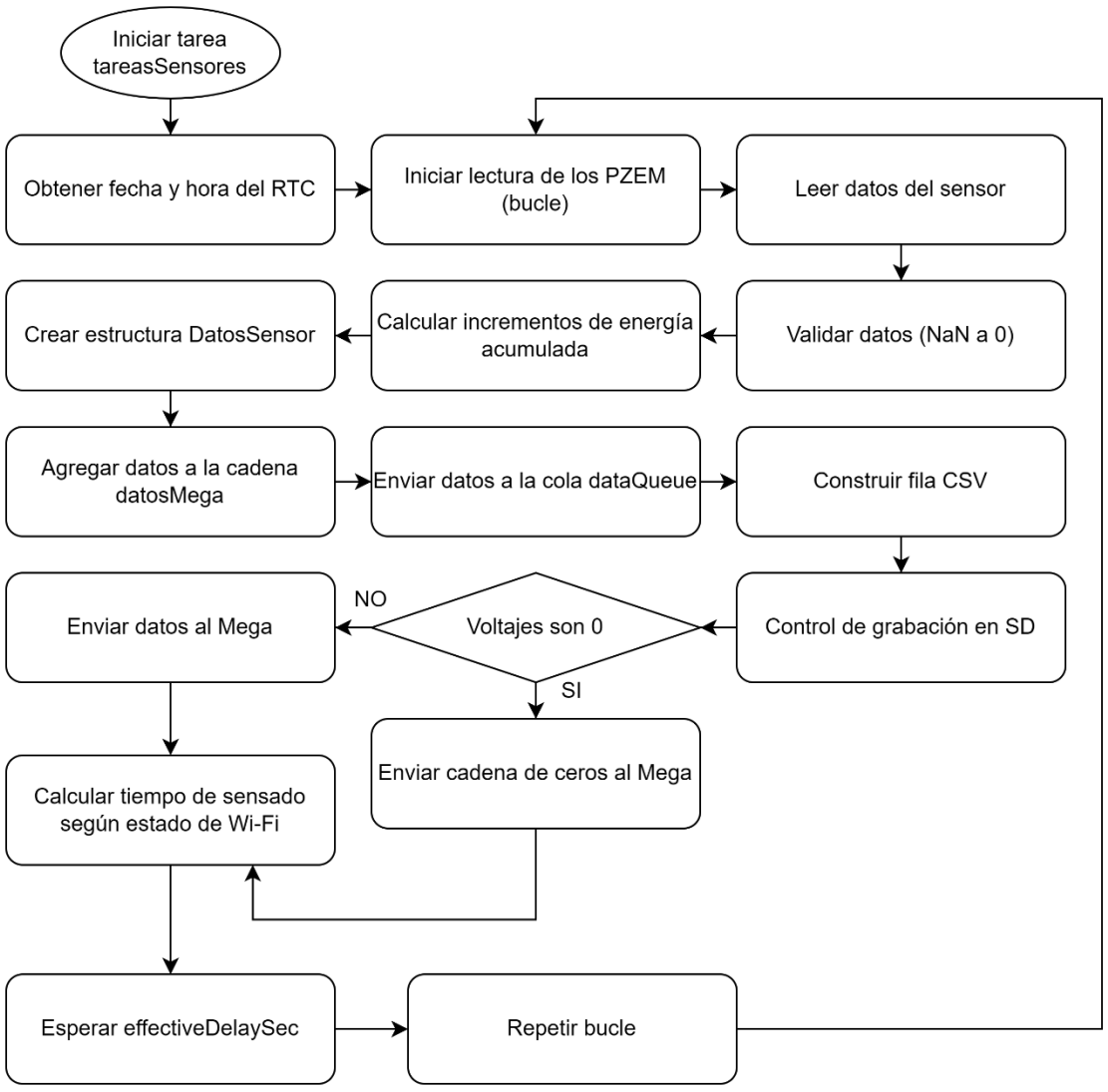


Fig. 26: Diagrama de flujo “tareasSensores” (continuación ESP-32). (Fuente Propia)

Por último, se inicializa el Mega configurando la comunicación serial entre ambos microprocesadores, además se inicializa la pantalla TFT conectada al mega. En el Loop Principal, entra en bucle y verifica los datos recibidos del ESP-32. Lee los datos, los cuales son las mediciones de los sensores, luego se parsean (se analiza la cadena de texto y se extrae la información significativa) se almacenan en una estructura llamada “DatosSensor” para la fila de datos de cada sensor. Se actualiza la pantalla mostrando los parámetros eléctricos divididos en L1, L2 y L3. Además, se verifican los valores de voltajes si todos los voltajes son “0” se muestra

un mensaje de error en pantalla; mientras que si los voltajes son valores validos se espera a la siguiente lectura. (Véase la figura 27)

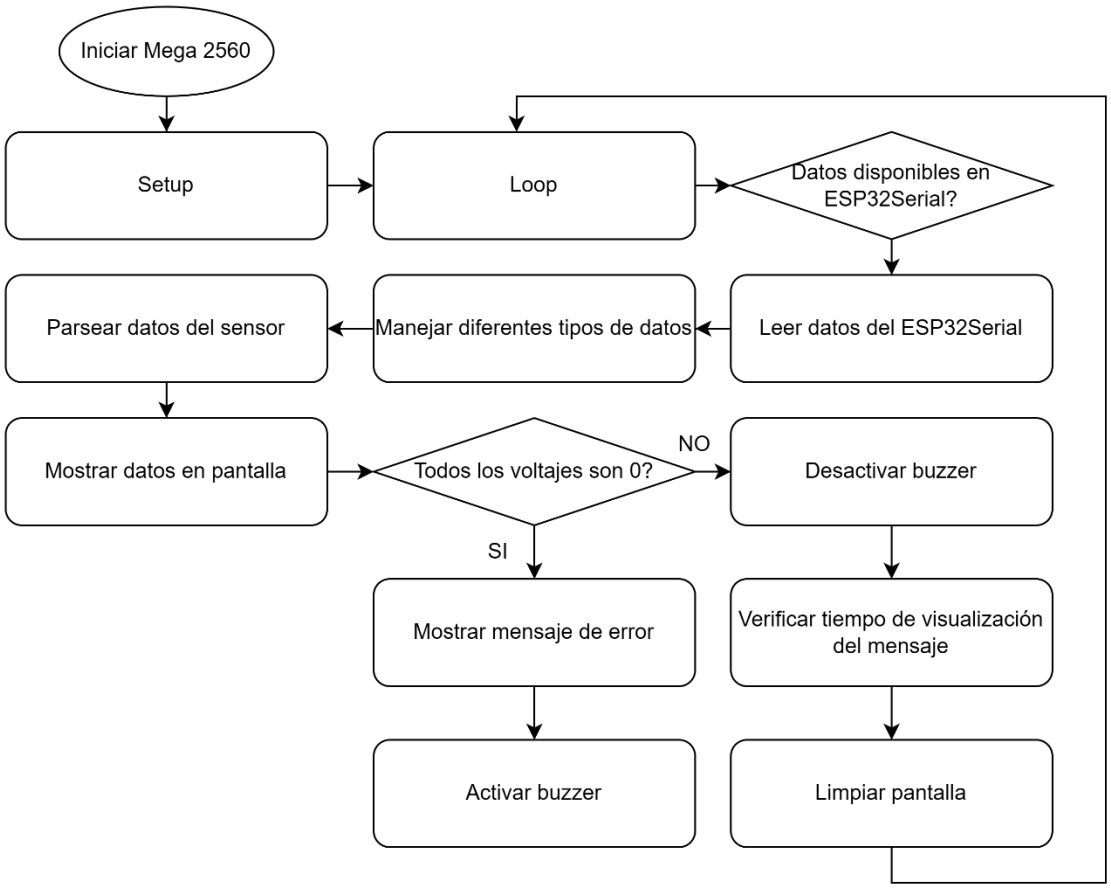


Fig. 27: Diagrama de flujo del Mega 2560. (Fuente Propia)

5.2 Esquema de Conexiones Generales

En figura 28 se muestra las conexiones entre los componentes del prototipo necesarias para correcto funcionamiento. El esquema realizado con el programa de simulación Fritzing, refleja como el ESP-32 es el punto en común entre todos los periféricos, siendo este el encargado de recibir, procesar y enviar los datos y la información entre los componentes. En el **Anexo 10** se presentan las tablas ilustrando las conexiones entre los componentes con sus respectivos pines.

Cabe mencionar que al momento de subir el código en el ESP-32, es preferible que no esté conectado a ningún periférico, de esta manera se evitan problemas

de compilación de código. Cuando más se presentaron inconvenientes fue cuando se están utilizando los pine RX y TX del ESP-32, causando error en la compilación. A continuación, se muestra la ilustración de las conexiones entre los componentes:

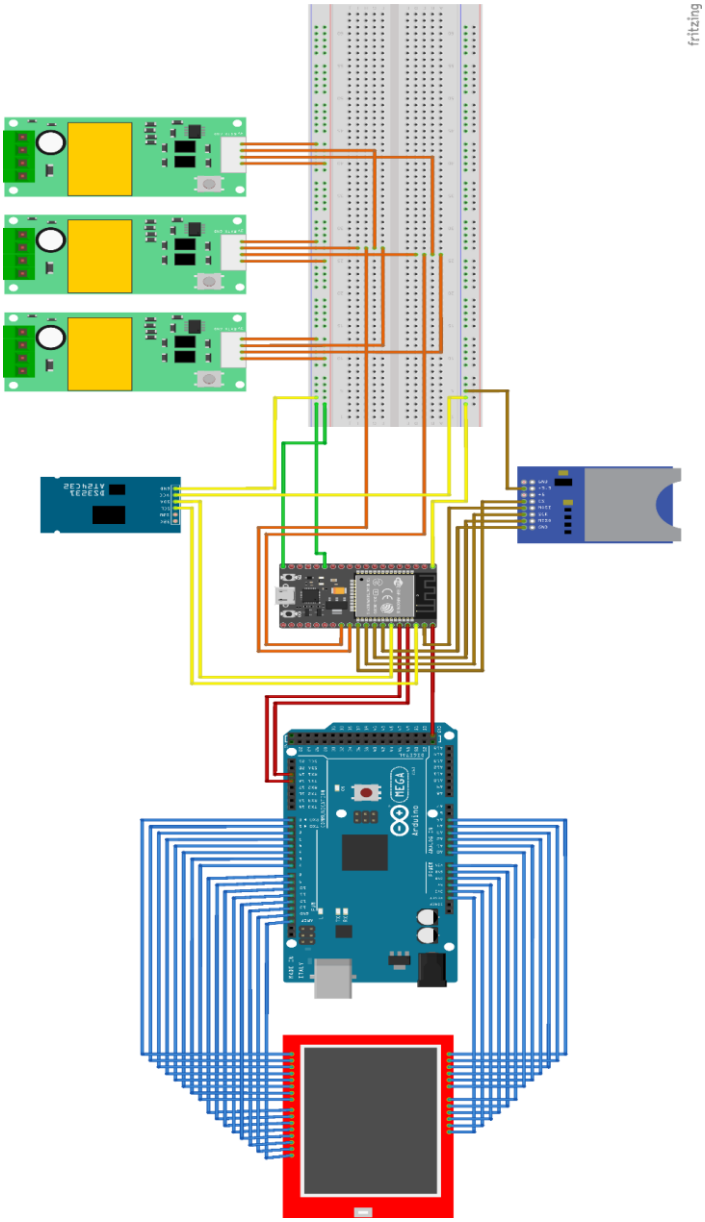


Fig. 28: Esquema de conexión del prototipo. (Fuente Propia)

5.3 Explicación de códigos de programación

5.3.1 Código para asignar dirección a los PZEM-004T

Anteriormente, se comentó que se deben asignar direcciones para evitar conflictos en la comunicación entre el microcontrolador y los sensores. A continuación, se muestra el código que se encuentra dentro de la biblioteca `PZEM004Tv30.h` en la sección de ejemplos, llamado PZEMChangeAddress.ino:

```
#include <PZEM004Tv30.h>

#if !defined(PZEM_RX_PIN) && !defined(PZEM_TX_PIN)
#define PZEM_RX_PIN 16
#define PZEM_TX_PIN 17
#endif

#if !defined(PZEM_SERIAL)
#define PZEM_SERIAL Serial2
#endif

#if defined(USE_SOFTWARE_SERIAL)
#include <SoftwareSerial.h>

SoftwareSerial pzemSWSerial(PZEM_RX_PIN, PZEM_TX_PIN);
PZEM004Tv30 pzem(pzemSWSerial);

#elif defined(ESP32)

PZEM004Tv30 pzem(PZEM_SERIAL, PZEM_RX_PIN, PZEM_TX_PIN);

#else

PZEM004Tv30 pzem(PZEM_SERIAL);

#endif

 //En este apartado se debe elegir la nueva dirección
 La dirección puede ser entre 0x01 y 0xF7
 #if !defined(SET_ADDRESS)
#define SET_ADDRESS 0x10
#endif
```

5.3.2 Código compilado en el ESP-32

5.3.2.1 Bibliotecas

A continuación, se describe cada biblioteca utilizada en la programación del dispositivo.

- `#include <WiFi.h>`
  Favorece a que el ESP-32 se conecte a una red Wi-Fi.
- `#include <WebServer.h>`
  Permite crear un servidor web.
- `#include <PZEM004Tv30.h>`
  Posibilita al ESP-32 interactúe con los sensores PZEM-004T.
- `#include <HTTPClient.h>`
  Permite realizar solicitudes HTTP para interactuar con el servidor web.
- `#include <InfluxDbClient.h>`
  Facilita la conexión y envío de datos a una base de datos InfluxDB.
- `#include <InfluxDbCloud.h>`
  Al igual que la anterior, pero esta interactúa directamente con InfluxDB Cloud.
- `#include <Preferences.h>`
  Habilita al ESP-32 para almacenar y recuperar los datos persistentes en la memoria no volátil, en el prototipo almacena las redes Wi-Fi conocidas.
- `#include <NTPClient.h>`
  Permite obtener la hora actual desde un servidor Network Time Protocol (NTP), de esta manera se sincroniza la hora en el prototipo.
- `#include <WiFiUdp.h>`
  Habilita la comunicación User Datagram Protocol (UDP) sobre el Wi-Fi, mejora la interacción con el Network Time Protocol (NTP).
- `#include <Wire.h>`
  Permite que el ESP32 se conecte a sensores, pantallas y otros periféricos, que requiere una comunicación con protocolo tipo I2C (Inter-Integrated Circuits).
- `#include <RTClib.h>`
  Facilita la interacción con el módulo de reloj en tiempo real DS3231.
- `#include <SD.h>`
  Permite leer y escribir en la tarjeta SD utilizando protocolo Serial Peripheral Interface (SPI).

- `#include <SPI.h>`
  Permite la comunicación Serial Peripheral Interface (SPI).
- `#include <ArduinoJson.h>`
  Facilita la serialización y deserialización de datos en formato JSON, en este caso optimiza el manejo de las respuestas de las rede Wi-Fi disponibles.

5.3.2.2 Cuerpo del código

A continuación, se explicará lo más esencial del código de manera resumida. Ya que este es muy extenso, se presenta el código completo en **Anexo 11**.

```
// Parámetros de InfluxDB
#define INFLUXDB_URL "https://us-east-1-1.aws.cloud2.influxdata.com"
#defineINFLUXDB_TOKEN
"T_FlzDcJvywbusJRcvqbXd9cR0sjp26UhpihYetC4kRMASUE3uI1A2GQHE0CxHKKBoK6-
udFd0M1MeleI-OusQ=="
#define INFLUXDB_ORG "2ac6b59b1078deff"
#define INFLUXDB_BUCKET "datossensores"
#define TZ_INFO "UTC-6"
```

Se definen las constantes necesarias para la conexión con InfluxDB, como el URL del servidor al que se envían los datos (en nuestro caso el servidor está alojado en la región us-east-q de Amazon Web Services, este es un servidor gratuito y predefinido por InfluxDB). Además, se define el token de autenticación el cual es necesario para acceder a la API de InfluxDB; continuando con el ID de la organización en InfluxDB, esta es la que permite agrupar y gestionar los usuarios, buckets y tareas. Por último, se define el nombre del bucket donde se almacenan los datos y se define la zona horaria del sistema correspondiente a la zona horaria de Nicaragua.

```
// Configuración PZEM
#define PZEM_RX_PIN 16
#define PZEM_TX_PIN 17
#define PZEM_SERIAL Serial2
#define NUM_PZEMS 3
PZEM004Tv30 pzems[NUM_PZEMS];
```

En este apartado se definen los pines en los que deben estar conectados los módulos PZEM y la comunicación serial entre el ESP-32 los sensores. Cabe aclarar de que los pines 16 y 17 del ESP-32 son los que actúan como receptor y transmisor (RX y TX) respectivamente. Además, verificar que se conecten pin17 al RX y pin 16 al TX del PZEM. Además, se declara el arreglo de objetos de la clase PZEM004Tv30 con la variable NUM_PZEMS para determinar la cantidad de sensores PZEM conectados.

```
// Configuración de la tarjeta SD
#define SD_CS_PIN 5
File archivoDatos;
```

Se asigna el pin 5 del ESP para comunicarse con el pin CS (Cip Select) del módulo adaptador de la tarjeta SD. También se declara el objeto archivosDatos de la clase File para manipular el archivo en la SD.

```
// Variables para la comunicación con el Mega
#define MEGA_TX_PIN 1 // ESP32 TX conectado a Mega RX
#define MEGA_RX_PIN 3 // ESP32 RX conectado a Mega TX
HardwareSerial MegaSerial(1); // Usar Serial1 para comunicación con el Mega
```

Se definen los pines que se conectan al Mega, para el envío y recepción de datos. Además, se declara la comunicación Serial 1.

```
// Variables globales
float energiaAcumulada[NUM_PZEMS] = { 0.0 };
unsigned long retrasoMedicion = 1;
bool recordInfinite = true;
bool isRecording = true;
unsigned long recordIntervalMinutes = 0;
DateTime recordStartTime;
String currentFilename = "/mediciones_infinite.csv";
```

En este apartado se inicializan las variables para las lecturas y almacenamiento de energía, el estado del tiempo de grabación y tiempos de censado de los sensores PZEM.

```
// Crear servidor web y prefrencias
WebServer server(80);
Preferences preferences;
RTC_DS3231 rtc;  // Crear un objeto para el RTC DS3231
```

Se crean objetos para iniciar el servidor web en el puerto 80, permitiendo gestionar solicitudes HTTP; además, permite guardar y recuperar datos en la memoria flash (memoria no volátil) del ESP-32. Por último, se crea el objeto para controlar el reloj en tiempo real que en nuestro proyecto es el módulo RTC DS3231.

```
// Inicialización del Cliente de InfluxDB
InfluxDBClient client(INFLUXDB_URL, INFLUXDB_ORG, INFLUXDB_BUCKET,
INFLUXDB_TOKEN);
Point sensor("datos_PZEM");
```

Se inicializa la conexión con InfluxDB Cloud, se crea un cliente utilizando las constantes previamente mencionadas como el URL, organización, bucket y token de autenticación. Además, se define un punto de medición llamado datos_PZEM, este es el espacio donde se almacenan los datos enviados.

```
// Configuración del cliente NTP
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", -21600, 60000); // UTC-6
(Nicaragua)
```

Se crea un objeto para manejar la comunicación UDP, necesaria para el NTP. Luego, utiliza el ntpUDP para conectarse, se sincroniza con el servidor gratuito llamado “pool.ntp.org”, el valor de -21600 es el ajuste de la zona horaria en segundos que para Nicaragua es UTC-6. Por último, el valor de 60000 dado en milisegundos es el intervalo de actualización; estos pasos son necesarios para sincronizar la hora, almacenada por el módulo RTC, con la del servidor en internet.

```
// HTML para la configuración
<!DOCTYPE html>
<html lang="es">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Configuraciones</title>
  <style>
  //configuración de estilo de la página
  </style>
</head>
<body>
//creación de contenedores y botones
  <script>
    const statusAlert = document.getElementById('statusAlert');
    function showAlert(message, type = 'success') {
      statusAlert.textContent = message;
      statusAlert.className = `alert ${type}`;
      statusAlert.style.display = 'block';
      setTimeout(() => statusAlert.style.display = 'none', 1000);
    }
    async function updateDelay(e) {
      e.preventDefault();
      const delay = document.getElementById('delay').value;
      fetch('/setDelay', { method: 'POST', headers: { 'Content-Type':
'application/x-www-form-urlencoded' }, body: `delay=${delay}` })
        .then(res => res.ok ? showAlert(`Tiempo actualizado: ${delay}s`) :
showAlert('Error', 'error'));
    }
    async function connectWiFi(e) {
      e.preventDefault();
      const ssid = document.getElementById('ssid').value;
      const password = document.getElementById('password').value;
      fetch('/connectWiFi', { method: 'POST', headers: { 'Content-Type':
'application/x-www-form-urlencoded' }, body:
`ssid=${encodeURIComponent(ssid)}&password=${encodeURIComponent(password)}
` })
        .then(res => res.ok ? showAlert(`Conectado a: ${ssid}`) :
showAlert('Error', 'error'));
    }
    async function clearCache() {
      if (confirm('¿Borrar credenciales?')) {
        fetch('/clearCache', { method: 'POST' })
          .then(res => res.ok ? showAlert('Credenciales borradas') :
showAlert('Error', 'error'));
      }
    }
  </script>
```

```
</body>
</html>
```

En esta fracción del código se define una de las partes fundamentales para la configuración del prototipo desde un servidor web. Se define la interfaz web en HTML, posibilitando al usuario configurar el tiempo de censado y conexión a la red Wi-Fi. De manera simplificada, este incluye un formulario para cambiar el tiempo de censado (/setDelay) con valor mínimo de 3 segundo y 10 segundos como máximo. Un formulario para conectar a una red Wi-Fi (/connectWifi) muestra las redes más cercanas y permite ingresar la contraseña para conectarse. También se muestra un botón para limpiar el caché del ESP-32 (/clearCache), de este modo se pueden limpiar las credenciales de Wi-Fi almacenadas y permitir conectarse a otra red si se desea. Con el fin de optimizar el código, se decidió utilizar funciones de tipo JavaScript como: updateDelay(), connectWifi(), clearWifi(), clearcache(); necesarias para envía el nuevo tiempo de censado al servidor, envía las credenciales de Wi-Fi al ESP-32, borra las credenciales almacenadas y borra datos temporales en el ESP-32.

```
// Estructura para almacenar datos de medición
struct DatosSensor {
  int indiceSensor;
  float voltaje;
  float corriente;
  float potenciaAparente;
  float frecuencia;
  float fp;
  float energiaAcumulada;
  float potenciaActiva;
  float potenciaReactiva;
};
```

Esto representa la estructura de los datos de las mediciones realizadas por los sensores, para su posterior almacenamiento y visualización.

```
// Semáforo y cola para datos
SemaphoreHandle_t wifiSemaphore;
QueueHandle_t dataQueue;
```

Se declaran dos elementos importantes con un Sistema Operativo en Tiempo Real (FreeRTOS). El primero, un “semáforo” que controla el acceso a recursos que es compartido por varias tareas, esto evita que varias tareas accedan al mismo recurso al mismo tiempo. El segundo, es una “cola” se usa para almacenar y enviar datos entre tareas de forma segura.

```
// Función para inicializar los sensores PZEM
void inicializarPZEMS() {
    for (int i = 0; i < NUM_PZEMS; i++) {
         pzems[i] = PZEM004Tv30(PZEM_SERIAL, PZEM_RX_PIN, PZEM_TX_PIN, 0x10
+ i);
        ultimaActualizacion[i] = millis()/1000;
        Serial.print("Sensor PZEM inicializado: ");
        Serial.println(i);
    }
}
```

Función para inicializar cada sensor PZEM, asignándole una dirección única a cada uno comenzando con 0x10 y aumenta hasta la cantidad de sensores que se conecten, en nuestro proyecto se presentan tres. Por lo tanto, 0x10 es para el módulo conectado a la L1, 0x11 es la dirección del módulo conectado a la L2 y, por último, la dirección 0x12 es la dirección para el módulo conectado a la L3. También, registra el tiempo de última actualización convertido a segundos.

```
// Leer datos de los PZEM
void tareasSensores(void *parametro) {
  while (true) {
    String datosMega = "";          // Cadena para enviar al Mega
    bool todosVoltajesCero = true; // Verificar si todos los voltajes son 0
   String filaCSV = ""; // Para la fila del CSV con datos de todos los
sensores

    // Obtener la fecha y hora del RTC
    DateTime now = rtc.now();
    String fechaHora = String(now.year()) + "-" + String(now.month()) + "-"
+ String(now.day()) + " " + String(now.hour()) + ":" + String(now.minute())
+ ":" + String(now.second());

    // Leer datos de los sensores PZEM
```

```
    for (int i = 0; i < NUM_PZEMS; i++) {
      float voltaje = pzems[i].voltage();
      float corriente = pzems[i].current() * 3;
      float frecuencia = pzems[i].frequency();
      float fp = pzems[i].pf();
      fp = fp - (fp * 0.05);
      float potenciaAparente = voltaje * corriente;
      float potenciaActiva = potenciaAparente * fp;

      uint64_t tiempoActualMillis = esp_timer_get_time() / 1000;
      float tiempoTranscurrido = (tiempoActualMillis - ultimaVezMillis[i])
/ 3600000.0f;
      ultimaVezMillis[i] = tiempoActualMillis;

      float incremento = potenciaActiva * tiempoTranscurrido;
      incremento = (incremento < 0 || incremento > 1000) ? 0 : incremento;
      energiaAcumulada[i] += incremento;

      // Crear estructura con datos del sensor
      DatosSensor datosSensor = { i, voltaje, corriente, potenciaAparente,
frecuencia, fp, energiaAcumulada[i], potenciaActiva, potenciaAparente *
sqrt(1 - (fp * fp)) };

      // Enviar datos a la cola
      xQueueSend(dataQueue, &datosSensor, portMAX_DELAY);

    }

    // Lógica de grabación controlada por intervalo
    if (isRecording) {
      if (!recordInfinite) {
        TimeSpan tiempoTranscurrido = now - recordStartTime;
                        unsigned     long     minutosTranscurridos     =
tiempoTranscurrido.totalseconds() / 60;

        if (minutosTranscurridos >= recordIntervalMinutes) {
          isRecording = false;
          recordInfinite = true;
          recordIntervalMinutes = 0;
          currentFilename = "/mediciones_infinite.csv";
          recordStartTime = now;
          isRecording = true;
        }
      }
```

```
      if (isRecording) {
        if (!SD.exists(currentFilename)) {
          archivoDatos = SD.open(currentFilename, FILE_WRITE);
          if (archivoDatos) {
            archivoDatos.println("Fecha y Hora,Voltaje1 (V), Corriente1 (A),
Potencia_Aparente1 (VA), Potencia_Activa1 (W), Potencia_Reactiva1 (VAR),
Frecuencia1 (Hz), FP1, Energia1 (W/h), …");
            archivoDatos.close();
          }
        }

        archivoDatos = SD.open(currentFilename, FILE_APPEND);
        if (archivoDatos) {
          archivoDatos.print(filaCSV);
          archivoDatos.close();
        }
      }
    }

    if (todosVoltajesCero) datosMega = "0,0,0,0,0,0,0,0,0;";

    if (datosMega.length() > 0) MegaSerial.println(datosMega);

     unsigned long effectiveDelaySec = (WiFi.status() == WL_CONNECTED) ?
((retrasoMedicion < 3) ? 3 : retrasoMedicion) : ((retrasoMedicion < 1) ? 1
: retrasoMedicion);
    vTaskDelay(pdMS_TO_TICKS(effectiveDelaySec * 1000));
  }
}
```

Se debe aclara que este es un resumen del apartado completo del código original. La tarea se encarga de obtener la fecha y hora utilizando el RTC almacenándola en una cadena, lee los valores censados por los PZEM, si los valores son NaN los reemplaza por 0, calcula la energía acumulada en cada línea, crea una estructura de datos con los valores leídos y los envía a una cola para su procesamiento, genera y guarda un archivo CSV para guardarlo en la tarjeta SD, envía todos los valores al mega para mostrarse en la pantalla, por último, espera el tiempo definido para repetir todo el proceso. También, se agregan las instrucciones del funcionamiento de los tiempos de grabación.

```
//Enviar datos a InfluxDB
void tareasInflux(void *parametro) {
  DatosSensor data;
  bool wasConnected = false;
  while (true) {
    if (xQueueReceive(dataQueue, &data, portMAX_DELAY) == pdTRUE) {
      if (WiFi.status() == WL_CONNECTED) {
        if (!wasConnected) {

        // Obtener la hora local (ya configurada por NTP)
        unsigned long epochTime = getCurrentEpochTime();
        if (epochTime == 0) {
          Serial.println("Error al obtener la hora.");
          continue;
        }

        // Obtener la hora de Nicaragua (UTC-6) como una variable adicional
        String horaNicaragua = timeClient.getFormattedTime();

        // Determinar el índice y se le asigna el nombre al sensor
        String sensorNombre;
        switch (data.indiceSensor) {
          case 0:
            sensorNombre = "L1";
            break;
             ..........
        }

        // Crear el payload para InfluxDB
        String payload = "datos_PZEM,sensor=" + sensorNombre;
        payload += " voltaje=" + String(data.voltaje, 2);
        payload += ",corriente=" + String(data.corriente, 2);
        payload += ",potenciaAparente=" + String(data.potenciaAparente, 2);
        payload += ",energiaAcumulada=" + String(data.energiaAcumulada, 2);
        payload += ",frecuencia=" + String(data.frecuencia, 2);
        payload += ",fp=" + String(data.fp, 2);
        payload += ",potenciaActiva=" + String(data.potenciaActiva, 2);
        payload += ",potenciaReactiva=" + String(data.potenciaReactiva, 2);
        payload += ",horaNicaragua=\"" + horaNicaragua + "\"";
        payload += " " + String(epochTimeNs);

        HTTPClient http;
              http.begin(String(INFLUXDB_URL)  +  "/api/v2/write?org="  +
INFLUXDB_ORG + "&bucket=" + INFLUXDB_BUCKET + "&precision=ns");
```

```
        http.addHeader("Content-Type", "text/plain");
        http.addHeader("Authorization", "Token " + String(INFLUXDB_TOKEN));

        Serial.println("Enviando payload a InfluxDB (sin timestamp):");
       Serial.println(payload.substring(0, payload.lastIndexOf(" ")));   //
Excluir timestamp al imprimir
        int httpResponseCode = http.POST(payload);
        }
        http.end();
      }
    }
  }
}
```

TareasInflux es el encargado de enviar los datos censados a Influx DB Cloud de manera continua. Primero recibe los datos que están en cola, luego verifica la conexión a Wi-Fi y se sincroniza el tiempo con el servidor NTP. Se le asignan los nombres de las líneas (L1, L2 y L3) dependiendo del índice del sensor (pzem 0, pzem 1 y pzem 2). Se crea la cadena de texto que es compatible con influx llamada payload con los datos de los sensores en la estructura requerida por influx. Por último, utiliza el HTTPClient para enviar el payload a influx.

```
// Función para generar HTML con opciones de redes Wi-Fi disponibles
String getNetworksHTML() {
  String networksHTML = "";
  if (WiFi.status() != WL_CONNECTED) {
    int n = WiFi.scanNetworks();
    for (int i = 0; i < n; ++i) {
         networksHTML  +=  "<option  value=\""  +  WiFi.SSID(i)  +  "\">"  +
WiFi.SSID(i) + " (" + WiFi.RSSI(i) + " dBm)</option>";
    }
    WiFi.scanDelete();
    if (networksHTML == "") {
      networksHTML = "<option>No se encontraron redes</option>";
    }
  } else {
    preferences.begin("wifi-creds", true);
    String ssid = preferences.getString("ssid", "");
    preferences.end();
    if (ssid.length() > 0) {
```

```
      networksHTML = "<option value=\"" + ssid + "\" disabled>Conectado a:
" + ssid + "</option>";
    }
  }
  return networksHTML;
}

// Función para manejar la solicitud de escaneo de redes Wi-Fi
void handleScanNetworks() {
  int n = WiFi.scanNetworks();
  DynamicJsonDocument doc(2048);
  JsonArray networks = doc.to<JsonArray>();
  for (int i = 0; i < n; i++) {
    JsonObject network = networks.createNestedObject();
    network["ssid"] = WiFi.SSID(i);
    network["rssi"] = WiFi.RSSI(i);
  }
  WiFi.scanDelete();

  String response;
  serializeJson(doc, response);
  server.send(200, "application/json", response);
}
```

Estas funciones se encargan de generar un listado en el HTML con todas las redes Wifi cercanas. Si el prototipo no está conectado, escanea las redes disponibles y las muestra en el servidor. Si el prototipo está conectado, muestra el nombre de la red a la que se encuentra conectado.

```
// Función para esperar y sincronizar el NTP una vez
void syncRTCWithNTP() {
  int attempts = 0;
  const int maxAttempts = 10;

    // Actualizar el tiempo del RTC DS3231
    if (rtc.begin()) {
      rtc.adjust(DateTime(timeClient.getEpochTime()));
      DateTime now = rtc.now();
      Serial.print("Hora almacenada en el RTC DS3231: ");
      Serial.print(now.year(), DEC);
      Serial.print('/');
      Serial.print(now.month(), DEC);
```

```
      Serial.print('/');
      Serial.print(now.day(), DEC);
      Serial.print(" ");
      Serial.print(now.hour(), DEC);
      Serial.print(':');
      Serial.print(now.minute(), DEC);
      Serial.print(':');
      Serial.println(now.second(), DEC);
    } else {
      Serial.println("Error al inicializar el RTC DS3231.");
    }
  } else {
     Serial.println("No se pudo sincronizar con NTP después de varios
intentos. Usando la hora del RTC DS3231.");

    if (rtc.begin()) {
      DateTime now = rtc.now();
      Serial.print("Hora recuperada del RTC DS3231: ");
      Serial.print(now.year(), DEC);
      Serial.print('/');
      Serial.print(now.month(), DEC);
      Serial.print('/');
      Serial.print(now.day(), DEC);
      Serial.print(" ");
      Serial.print(now.hour(), DEC);
      Serial.print(':');
      Serial.print(now.minute(), DEC);
      Serial.print(':');
      Serial.println(now.second(), DEC);

      // Configurar el tiempo del ESP32 usando el tiempo del RTC DS3231
      struct tm t;
      t.tm_year = now.year() - 1900;
      t.tm_mon = now.month() - 1;
      t.tm_mday = now.day();
      t.tm_hour = now.hour();
      t.tm_min = now.minute();
      t.tm_sec = now.second();
      time_t rtcTime = mktime(&t);
      timeval tv = { rtcTime, 0 };
      settimeofday(&tv, NULL);
    } else {
      Serial.println("Error al leer el RTC DS3231.");
    }
  }
```

```
}
```

Esta función intenta sincronizar el tiempo con el servidor NTP, si se logra sincronizar se actualiza la hora en el módulo RTC con la hora obtenida y si no se logra sincronizar por falta de conexión wifi, se utiliza la hora almacenada en el mismo módulo.

```
// Función para obtener la hora actual en formato epoch
unsigned long getCurrentEpochTime() {
  time_t now;
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    Serial.println("Error obteniendo la hora local.");
    return 0;
  }
  now = mktime(&timeinfo);
  return now;
}

// Función para formatear un intervalo de tiempo dado en minutos
String formatInterval(unsigned long totalMinutes) {
    unsigned long days = totalMinutes / (24 * 60);
    unsigned long hours = (totalMinutes % (24 * 60)) / 60;
    unsigned long minutes = totalMinutes % 60;
    // Retorna el intervalo formateado como "días horas minutos"
    return String(days) + "d " + String(hours) + "h " + String(minutes) +
"m";
}
```

Las funciones getCurrentEpochTime y formatInterval son las encargadas de obtener la hora actual en formato epoch, el cual cuanta los segundos transcurridos desde el 1 de enero de 1970 a las 00:00:00 UTC (muy utilizado en el ámbito computacional). Utiliza get LocalTime para obtener la hora local, luego la convierte en tiempo epoch. Además, en algunas funcionalidades del proyecto se necesita otro formato del tiempo, entonces se convierten en días, horas y minutos solamente.

```
// Función para configurar el servidor web
void setupServer() {
  const char *defaultSSID = "ANALIZADOR DE REDES - UNI";
  const char *defaultPassword = "12345678";
  WiFi.mode(WIFI_AP_STA);
  WiFi.softAP(defaultSSID, defaultPassword);
  WiFi.softAPConfig(IPAddress(192, 168, 4, 1), IPAddress(192, 168, 4, 1),
IPAddress(255, 255, 255, 0));

  Serial.print(F("IP del punto de acceso: "));
  Serial.println(WiFi.softAPIP());

  // Ruta principal "/"
  server.on("/", HTTP_GET, []() {
  });

  // Servir la imagen desde la SD
  server.on("/logo.png", HTTP_GET, []() {
      });

  // Devolver redes disponibles
  server.on("/scanNetworks", HTTP_GET, handleScanNetworks);

  // Redirección en caso de no encontrar la ruta
  server.onNotFound([]() {
    server.sendHeader("Location", "http://192.168.4.1/", true);
    });

  // Configurar Intervalo de Grabación
  server.on("/setRecordInterval", HTTP_POST, []() {
  });

  // Configurar Retardo de Medición
  server.on("/setDelay", HTTP_POST, []() {
  });

  // Conectar a Wi-Fi
  server.on("/connectWiFi", HTTP_POST, []() {
  });

  // Eliminar credenciales Wi-Fi
  server.on("/clearWiFi", HTTP_POST, []() {

  });
```

```
  // Limpiar caché de escaneo Wi-Fi
  server.on("/clearCache", HTTP_POST, []() {
  });

  server.begin();
  Serial.println(F("Servidor web iniciado"));
}
```

En la función void setupServer() se encarga de configurar y de inicializar el servidor web en el ESP32. Estableciendo la dirección IP del punto de acceso, permitiendo que el microcontrolador se conecte a una red Wi-Fi. Sirve la página HTML donde se pueden realizar las configuraciones de los tiempos de censado y de red. Escanea las redes que estén disponibles, guarda las credenciales en la memoria del ESP32.

```
// Función para enviar mensajes al Mega
void enviarMensajeMega(const String &mensaje) {
  MegaSerial.println(mensaje);
  Serial.println("Enviado al Mega: " + mensaje);
}
```

Esta función es útil para comunicar datos desde el ESP32 al Mega.

```
void setup() {
// Iniciar comunicación I2C
Wire.begin(21, 22);

// Leer el delay almacenado en memoria no volátil
preferences.begin("settings", true);
retrasoMedicion = preferences.getUInt("retrasoMedicion", 1);
preferences.end();

// Inicializar las credenciales Wi-Fi guardadas en memoria no volátil
preferences.begin("wifi-creds", true);
String ssid = preferences.getString("ssid", "");
String password = preferences.getString("password", "");
preferences.end();

WiFi.begin(ssid.c_str(), password.c_str());
while (WiFi.status() != WL_CONNECTED && attempts < 30) { delay(1000); }
```

```
// Inicializar los sensores PZEM
inicializarPZEMS();

// Sincronizar el NTP y configurar el tiempo
syncRTCWithNTP();

// Inicializar la tarjeta SD
if (!SD.begin(SD_CS_PIN)) {  }

// Configurar servidor web
setupServer();

// Crear tareas
xTaskCreatePinnedToCore(tareasSensores, "tareasSensores", 4096, NULL, 4,
NULL, 1);
xTaskCreatePinnedToCore(tareasInflux, "tareasInflux", 4096, NULL, 3, NULL,
0);
```

La funsion void setup () es la encargada de inicializar un gran conjunto de tareas a realizar por el esp32. Se configura la comunicación serial con el Mega. Además, se inicializa el bus I2C (Inter-Integrated Circuit) uno de los protocolos de comunicación más comunes que permite transferir datos entre los periféricos, en este caso se aplica con el módulo RTC, se ajusta la hora del RTC. Se leen las configuración previas y credenciales que estén guardadas en la memoria no volátil (NVS). Se inicializan los sensores PZEM y se sincroniza la hora con el servidor NTP; además, la memoria SD para el almacenamiento de datos. Por último, se crean las tareas para manejar la lectura de los datos de los sensores y del envío a influx sin que se ocasione una interferencia en la transmisión y recepción de datos.

```
void loop() {
  server.handleClient();  // Procesar las peticiones del servidor web

  // Mostrar la hora desde RTC y las mediciones de los sensores
  if (WiFi.status() != WL_CONNECTED) {
    DateTime now = rtc.now();
    String formattedTime = String(now.year()) + "-" + String(now.month()) +
"-"  +  String(now.day())  +  "  "  +  String(now.hour())  +  ":"  +
String(now.minute()) + ":" + String(now.second());
```

```
    Serial.println("Hora (sin conexión Wi-Fi): " + formattedTime);

    delay(1000);
  }
}
```

Por último, en el void loop() se llama a la función server.handleClient() para verificar si hay nuevas peticiones entrantes al servidor web. Además, se verifica si esta desconectado del WiFi, si no hay conexión se obtiene la hora actual del RTC.

5.3.3 Código compilado en el Mega2560

A continuación, se explicará lo más esencial del código de manera resumida. Ya que este es muy extenso, se presenta el código completo en **Anexo 12**.

5.3.3.1 Bibliotecas

```
#include <MCUFRIEND_kbv.h>
```

Permite controlar las pantallas TFT LDC, 2,4, 2,8, 3,5, 3,6 y 3,95 pulgadas

```
#include <SoftwareSerial.h>
```

Facilita la comunicación serial entre el ESP 32 y el Arduino mega 2560

5.3.3.2 Cuerpo del código

```
#define MEGA_RX_PIN 19
#define MEGA_TX_PIN 18
#define BUZZER_PIN 44

HardwareSerial &ESP32Serial = Serial1;

MCUFRIEND_kbv tft;

struct DatosSensor {
    int indiceSensor;
    float voltaje;
    float corriente;
    float potenciaAparente;
    float frecuencia;
```

```
    float fp;
    float energiaAcumulada;
    float potenciaActiva;
    float potenciaReactiva;
};
```

En el apartado de “define”, representa los componentes que permiten nombrar constantes definidas en Arduino, en este caso definir los pines que se utilizan para las respectivas comunicaciones. Luego de eso creamos una referencia “&” entre los pines (rx y tx) de el esp32 y los pines de serial1 (rx y tx) del Arduino mega 2560, luego implementamos una técnica de estructura para organizar los datos con números decimales (float)y también enteros (int)

```
DatosSensor lineData[3]; // Información de las tres líneas
bool lineActive[3] = {false, false, false}; // Indica si cada línea tiene
datos recibidos
bool datosRecibidos = false; // Bandera para verificar si se reciben datos

unsigned long messageDisplayTime = 0; // Tiempo en el que se inició la
visualización del mensaje
const unsigned long messageDuration = 3000; // Duración del mensaje en
milisegundos (3 segundos)
bool displayingMessage = false; // Bandera para verificar si se está
mostrando un mensaje

int errorCount = 0; // Contador de errores de voltaje 0
const int maxErrorCount = 10; // Máximo número de veces que el buzzer
sonará por error

bool buzzerActive = false; // Bandera para controlar si el buzzer está
activo
```

luego establecemos un arreglo para que se establezca una matriz, que al contener valores boleanos estos podrán de cambiar de false a true si en algún momento se recibe información en una línea, o de manera general, ademas de definir cuánto tiempo se debe visualizar un mensaje en pantalla antes de ocultarlo o de cambiar el estado del sistema, luego si no se reciben desde el primer conteo que se realiza, sonara zumbador 10 veces, y si ya se activado por un error, espera que cambie de estado, y volverá a funcionar en caso que pase de nuevo a estado de fallo

```
// Prototipos de funciones que se ocupan mas adelante
void parseSensorData(String data);
void displaySensorData(const DatosSensor *data, int displayIndex);
void displayErrorMessage();
void printSensorDataToSerial(const DatosSensor *data);
bool allVoltagesZero();
void resetSensorData(DatosSensor &data);
void displayConnectionMessage(const String &mensaje);
void soundBuzzer();
void stopBuzzer();

void setup() {
    Serial.begin(115200);
    ESP32Serial.begin(9600);
    pinMode(BUZZER_PIN, OUTPUT); // Configurar el pin del buzzer como
salida

    uint16_t identifier = tft.readID();
    if (identifier == 0x0 || identifier == 0xFFFF) {
        identifier = 0x9341;
    }
    tft.begin(identifier);
    tft.setRotation(1);
    tft.fillScreen(TFT_BLACK);
    tft.setTextColor(TFT_WHITE);

    Serial.println("Mega listo para comunicarse con ESP32");
}
```

Al principio se declaron varias funciones que se ocuparan más adelante, para luego void setup funcine una vez se encienda el arduino, configurando asi las velocidades de comunicación de entre dispositivos externos , definir el pin de salida, luego se inicia el proceso de intentar leer el tipo de controlador de la pantalla tft, en donde compara los valores y definir si es un controlador soportado, con el controlador identificado se procede a iniciar la pantalla, y configura la orientación, el fondo de pantalla el color de las letras y un mensaje de verificación.

```
void loop() {
    // Verificar si hay datos disponibles desde el ESP32
if (ESP32Serial.available() > 0) {
```

```
    String data = ESP32Serial.readStringUntil('\n'); // Leer hasta nueva
línea
    Serial.print("Datos recibidos: ");
    Serial.println(data); // Mostrar los datos crudos para depuración

    if (data.startsWith("Conexión exitosa a Wi-Fi:")) {
        Serial.println("ESP32 conectado a Wi-Fi exitosamente.");
        displayConnectionMessage("Wi-Fi conectado: " +
data.substring(22)); // Mostrar el nombre de la red Wi-Fi
    } else if (data.startsWith("Desconectado de Wi-Fi")) {
        Serial.println("ESP32 desconectado de Wi-Fi.");
        displayConnectionMessage("Desconectado de Wi-Fi"); // Mostrar
mensaje de desconexión
    } else if (data.startsWith("Nuevo tiempo de censado:")) {
        Serial.println(data); // Mostrar el nuevo tiempo de censado
// se presentan todos los mensajes en pantalla
                datosRecibidos = true;
        parseSensorData(data); // Parsear y procesar los datos
        }

        // Mostrar los datos en la pantalla, cada línea tiene una posición
fija
        if (!displayingMessage) {
            for (int i = 0; i < 3; i++) {
                if (lineActive[i]) {
                    displaySensorData(&lineData[i], i); // Mostrar datos
en posición fija
                } else {
                    resetSensorData(lineData[i]); // Restablecer los datos
                    displaySensorData(&lineData[i], i); // Borrar de
pantalla
                }
            }

          }
        }
    }
}
    // Verificar si todos los voltajes son 0
    if (datosRecibidos) {
        if (allVoltagesZero()) {
            if (!buzzerActive) {
                displayErrorMessage(); // Mostrar mensaje de error si
todas las líneas tienen voltaje 0
                soundBuzzer(); // Activar el buzzer
```

```
                buzzerActive = true;
                errorCount = 0; // Reiniciar el contador de errores
            }
        } else {
            if (buzzerActive) {
                stopBuzzer(); // Desactivar el buzzer
                buzzerActive = false;
            }
        }
    }
        tft.fillScreen(TFT_BLACK); // Limpiar la pantalla
    }
}
```

Entramos el ciclo principal, primero comprueba si hay datos disponibles provenientes del esp32 en los puertos ya previamente designados y si los encuentra, se encarga de leer la cadena completa hasta encontrar el carácter de nueva línea ('\n'). Con los datos recibidos, comienza a evaluar todas las variables con diferentes condicionales, en caso que se cumpla o no, imprimirá un mensaje que muestre el estado de la condicional. Luego de pasar por todas las condicionales, si se reciben los datos comenzara a extraer los valores con el arreglo de linedata, verificando constantemente si las tres líneas tienen datos si no limpia o resetea la línea que este ausente, luego los datos se muestran en la pantalla en una posición fija, en el caso que una línea no este activa, la actualización de la pantalla restablecerá los ultimo datos de la línea ausente, además de verificar el caso en el que el conjunto de datos de tensión sea 0, de ser asi, se activara el buzzer y mostrando un mensaje de error en la pantalla, para todos los casos de mostrar dato o error, la pantalla está constantemente actualizándose y refrescando los valores.

```
// Función para restablecer los datos de una línea
void resetSensorData(DatosSensor &data) {
    data.voltaje = 0.0;
    data.corriente = 0.0;
    data.potenciaAparente = 0.0;
    data.frecuencia = 0.0;
    data.fp = 0.0;
    data.energiaAcumulada = 0.0;
```

```
    data.potenciaActiva = 0.0;
    data.potenciaReactiva = 0.0;
}
```

La primera funciona crea una referencia de los valores de datossensor, cuando se reinicia limpia los últimos valores y les asigna el valor de 0 a cada uno de los campos numéricos visibles, garantizando que no queden valores de lecturas anteriores. Luego la siguiente función procede a imprimirlos de delimitando visualmente la información en la pantalla, mostrando las etiquetas de los sensores de manera más natural desde 1 en vez de desde 0 y escribiendo el nombre y la unidad de medida de cada parámetro eléctrico que se imprimirá en la pantalla mostrando los valores numéricos con 2 decimales de precisión.

```
// Función para descomponer los datos del ESP32
void parseSensorData(String data) {
    }

    while (data.length() > 0) {
        int separatorIndex = data.indexOf(';');
        String lineDataString = (separatorIndex != -1) ? data.substring(0,
separatorIndex) : data;
        data = (separatorIndex != -1) ? data.substring(separatorIndex + 1)
: "";

        String values[10];
        int valueIndex = 0;
        while (lineDataString.length() > 0 && valueIndex < 10) {
            int commaIndex = lineDataString.indexOf(',');
            values[valueIndex] = (commaIndex != -1) ?
lineDataString.substring(0, commaIndex) : lineDataString;
            lineDataString = (commaIndex != -1) ?
lineDataString.substring(commaIndex + 1) : "";
            valueIndex++;
        }

        // Convertir los datos de texto a los valores numéricos
        int indiceSensor = values[0].toInt();
        Serial.print("Procesando datos para L");
        Serial.println(indiceSensor + 1);
```

```
        if (indiceSensor >= 0 && indiceSensor < 3) { // Validar que el
indiceSensor esté en el rango 0-2
            if (!lineActive[indiceSensor]) { // Solo procesar si no se ha
recibido datos previamente para este índice
                lineActive[indiceSensor] = true;
                lineData[indiceSensor].indiceSensor = indiceSensor;
                lineData[indiceSensor].voltaje = values[1].toFloat();
                lineData[indiceSensor].corriente = values[2].toFloat();
                lineData[indiceSensor].potenciaAparente =
values[3].toFloat();
                lineData[indiceSensor].frecuencia = values[4].toFloat();
                lineData[indiceSensor].fp = values[5].toFloat();
                lineData[indiceSensor].energiaAcumulada =
values[6].toFloat();
                lineData[indiceSensor].potenciaActiva =
values[7].toFloat();
                lineData[indiceSensor].potenciaReactiva =
values[8].toFloat();
            } else {
                continue; // Saltar a la siguiente iteración
            }
        } else {
            Serial.println("Error: indiceSensor fuera de rango");
        }
    }
}
```

Descompone los datos que envia el esp32, y en un bucle mientras se envíen datos se comienzan a separar los datos por ‘;’ esto en espera de un arreglo de cadena llamados values que espera 10 valores y luego utiliza otro bucle while para dividir las cadenas usando ',' como separador, luego arregla los valores numéricos para poder imprimir un dato más natural y correspondiente que es l1, l2,l3, el bloque de código siguiente se encarga de verificar y almacenar los datos recibidos del esp32 y ponerlos en su posición correspondiente, comprobando el índice o etiqueta de los sensores y en caso de que no mostrar un mensaje de error, además de verificar errores de información duplicada.

```
// Función para verificar si todos los voltajes son 0
bool allVoltagesZero() {
    for (int i = 0; i < 3; i++) {
```

```
        if (lineActive[i] && lineData[i].voltaje > 0.0) {
            return false; // Si alguno de los voltajes no es 0, retornar
falso
        }
    }
    return true; // Todos los voltajes son 0
}

// Función para mostrar los datos en la pantalla
void displaySensorData(const DatosSensor *data, int displayIndex) {
    int yOffset = displayIndex * 80;

    tft.fillRect(0, 10 + yOffset, 320, 80, TFT_BLACK);
    String lineLabel = "L" + String(displayIndex + 1);

    tft.setCursor(10, 10 + yOffset);
    tft.print(lineLabel);

    tft.setCursor(10, 30 + yOffset);
    tft.print("Voltaje: ");
    tft.print(data->voltaje, 2);
    tft.print(" V");

    tft.setCursor(10, 50 + yOffset);
    tft.print("Corriente: ");
    tft.print(data->corriente, 2);
    tft.print(" A");

    // de la misma manera para el resto de los datos
}

// Función para mostrar un mensaje de error
void displayErrorMessage() {
}

// Función para mostrar un mensaje de conexión
void displayConnectionMessage(const String &mensaje) {
}

void soundBuzzer() {
    for (int i = 0; i < 5; i++) { // Ahora solo sonará 5 veces en cada
evento
        tone(BUZZER_PIN, 2000); // Suena con 2000 Hz
        delay(150); // Se mantiene el sonido por 150ms
        noTone(BUZZER_PIN); // Apagar el buzzer
```

```
        delay(250); // Pausa más larga para que sea menos molesto
    }
}

// apaga el buzzer
void stopBuzzer() {
    noTone(BUZZER_PIN);
}
```

Revisa si el voltaje es 0 en las tres línea, comprobando que las 3 líneas tienen valores mayores 0.0, luego se crea el posicionamiento de todos los datos que se mostraran en la pantalla TFT además de definir que las líneas ocuparan 80 pixeles de alto, y asignando posiciones fijas a las etiquetas (l1,l2,l3) fijando los parámetros eléctricos y unidades de medición, la siguiente función corresponde a el mensaje de error, en donde en caso de error, se limpia la pantalla en negro, aparece e mensaje correspondiente al error, en las coordenadas (10, 100) se establecen las letras del mensaje color en blanco y estas notificaciones son mostradas atreves de displayConnectionMessage que se declararon en void loop , esta notificaciones son por ejemplo, informativas a cerca del estado de conexión de wifi, el tiempo de censando, si se cambió el intervalo de tiempo o de grabado etcetera, luego en el bloque siguiente de programación hace que el buzzer suene 5 veces con un tono de 2000 Hz, manteniendo el sonido 150 ms y luego haciendo una pausa de 250 ms y luego void stopBuzzer() Apaga el buzzer de forma inmediata.

5.3.4 Código de InfluxDB Cloud

```
SELECT *
FROM "pzem_data"
WHERE
time >= now() - interval '30 days'
AND
("reactive_power" IS NOT NULL OR "active_power" IS NOT NULL OR "accumulated_energy" IS NOT NULL OR "energy" IS NOT NULL OR "frequency" IS NOT NULL OR "pf" IS NOT NULL OR "power" IS NOT NULL OR "voltage" IS NOT NULL OR "current" IS NOT NULL)
```

AND
"sensor" IN ('pzem0','pzem1','pzem2')

Para realizar una consulta básica en la consola o interfaz de influx db cloud requiere clausulas como lo son:
La primera clausula es “ SELECT * ” , esto indica que se seleccionaran tas las filas y columnas que cumplan con las condiciones que luego se especificaran luego.
La segunda clausula es “ FROM “ esto especifica el bucket , deposito o tabla donde se consultaran los datos.
“WHERE” es un comando que filtra los datos en relación de condiciones especificadas tanto como el rango de tiempo, campos específicos o valores de etiquetas específicos (influxData.inc, n.d.)
Destacamos que, a pesar de explicar el código para hacer la consulta de manera escrita, e incluso agregar más filtros, influx db cloud tiene una herramienta que sirve como motor de consulta llamado “SQL sync” en donde únicamente activándolo y haciendo click a los campos de y etiquetas necesarias obtendrás una consulta sin la necesidad de escribir el código de consulta manualmente.

5.3.5 Código de paneles en Grafana

A continuación, se muestra un ejemplo de los códigos con los que se logra la comunicación con Influx DB y la visualización de los datos en Grafana:

```
from(bucket: "datossensores")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "datos_PZEM")
  |> filter(fn: (r) => r["_field"] == "voltaje")
  |> filter(fn: (r) => r["sensor"] == "L1" or r["sensor"] == "L2" or r["sensor"] == "L3")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> yield()
```

Para configurar los paneles se inicia con “from” ya que este determinará el bucket donde se encuentran almacenados los datos. Dentro de la función “from” se escribe el nombre del depósito de datos, en nuestro caso se extraen de un “bucket” situado en Influx DB con el nombre de “datossensores”. A partir de eso se utiliza el operador de canalización “|>” el cual permite encadenar las funciones pasando de una a otra. Primero, declara la función de cálculo “range” Este paso limita los datos al rango de tiempo seleccionado en el dashboard de Grafana. Las variables v.timeRangeStart y v.timeRangeStop se definen dinámicamente según la configuración del panel, permitiendo que la gráfica se actualice de acuerdo al periodo de tiempo elegido. Luego, se continua con la función “filter” funciona similar a las clausulas SELECT y WHERE de Influx, identifica que filas de datos se reciben, en nuestro caso se define el argumento “r” llamado “_measurement” que es el “datos_PZEM” donde se almacenan los diferentes parámetros eléctricos, en este ejemplo se menciona el “voltaje”. Además, se define de cual “sensor” (línea) se reciben los datos. Seguidamente, se declara la función de agrupación de los datos en ventajas de tiempo fijas llamada “aggregateWindow” en donde los parámetros de “period” admite todas las unidades de duración como días, semanas o meses. Se utiliza la función “mean” con el objetivo de no sobre cargar los paneles cuando se estén presentando grandes cantidades de datos. El parámetro de “createEmpty: false” es el encargado de que no se creen intervalos vacíos y se omiten esos periodos de tiempo sin datos. Por último, la función “|> yield()” finaliza y devuelve los datos obtenidos en la consulta.

Destacamos que esta estructura de código se repite en cada uno de los paneles de visualización, en donde lo que cambia es el tipo de variables/campo que queremos visualizar (voltaje, corriente, potencia activa, reactiva, factor de potencia, frecuencia energía) ya que el bucket es donde se encuentra toda la base de datos y siempre extraemos la información de todos los sensores.

5.4 Base de datos y visualización

5.4.1 Creación de base de datos en InfluxDB Cloud

Como anteriormente mencionamos, decidimos ocupar los servicios de gratuitos, el primer paso que debemos seguir es crearnos una cuenta, ya sea con nuestros servicios Google y Microsoft.

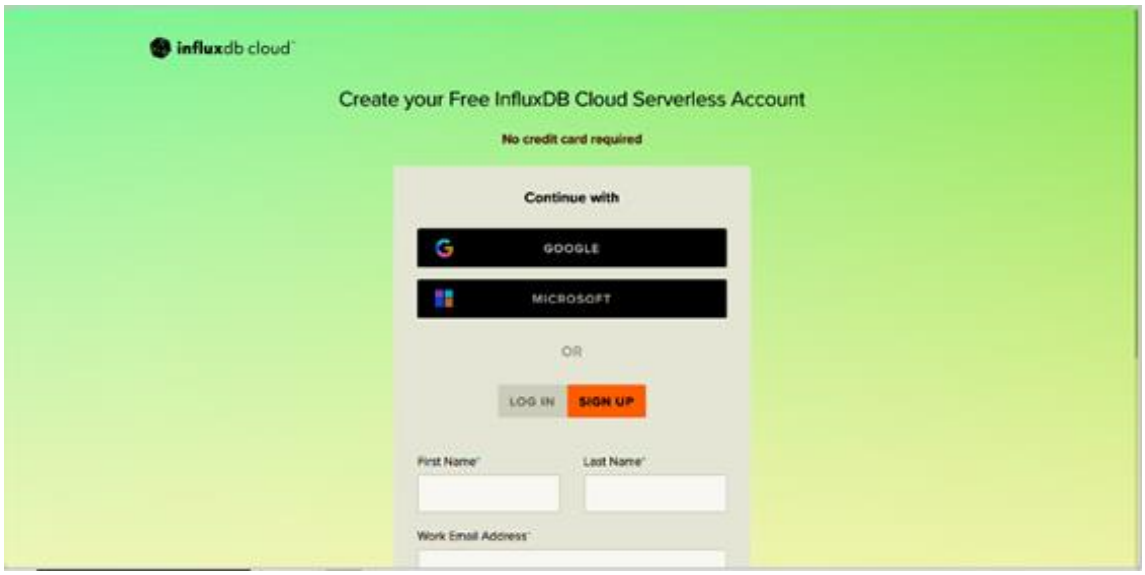


Fig. 29: crear cuenta (InfluxData, Inc., 2025)

En este apartado solo se configuran los nombres con el que se identificara el interfaz, puede ocupar cualquier tipo de nombre, es nuestro será UNI, además se expresa el proveedor de almacenamiento que obtendrás, siempre sin costo alguno.

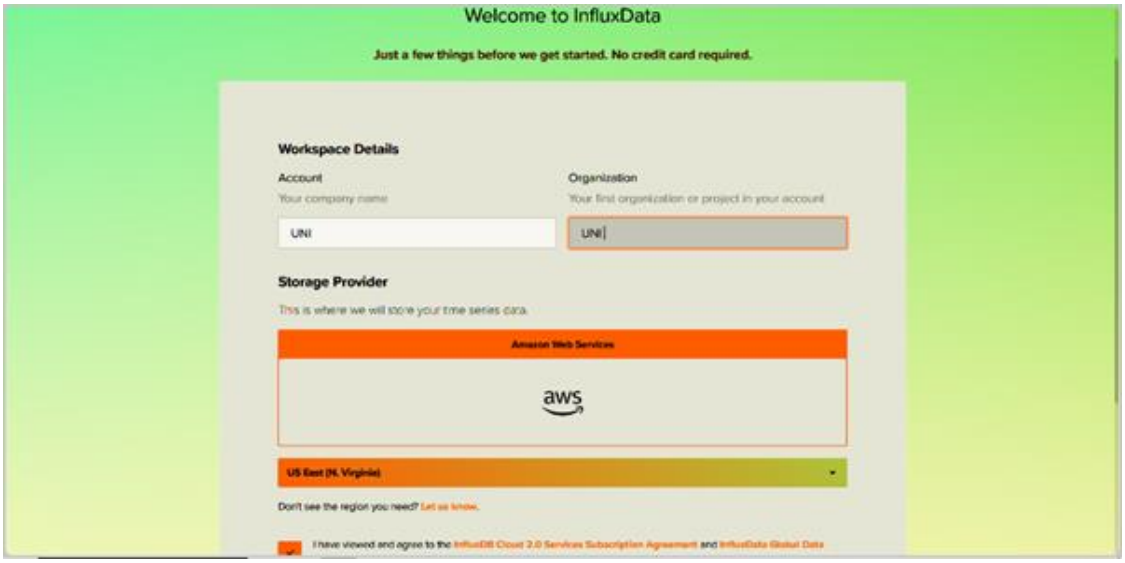


Fig. 30: configuraciones iniciales, términos y condiciones (InfluxData, Inc., 2025)

Una vez, finalizado podrás observar la cantidad bibliotecas de datos con las que es compatible, es nuestro caso es de interés vincularlo con Arduino ide.

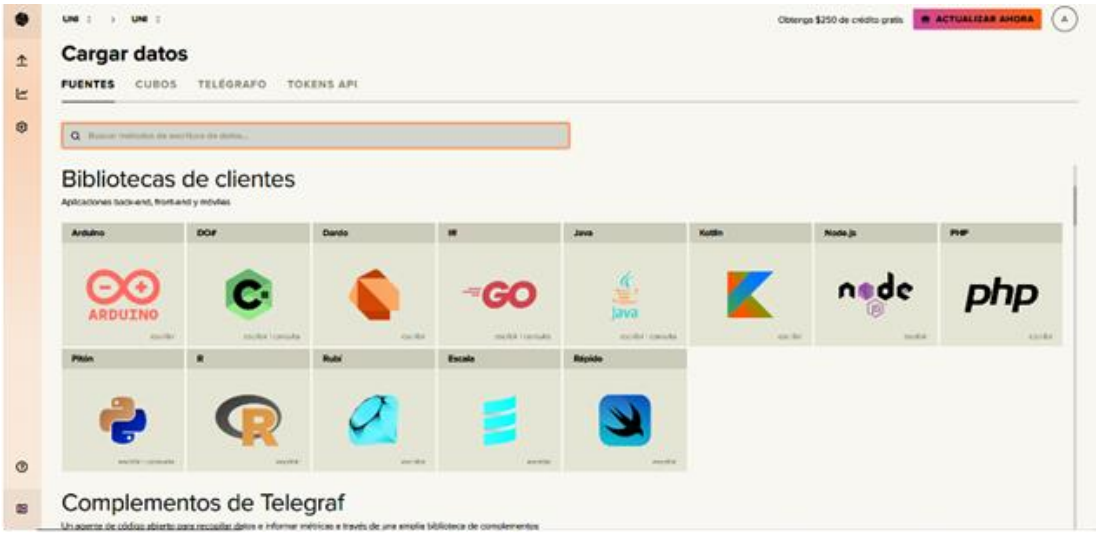


Fig. 31: selección de bibliotecas (InfluxData, Inc., 2025)

Una vez seleccionada la biblioteca de interés para nuestro caso Arduino ide podrás observar los pasos básicos e intuitivos para poder desarrollar el proyecto con Arduino ide y con la placa de desarrollo del ESP32.

Fig. 32: paso iniciales Arduino IDE y ESP32 (InfluxData, Inc., 2025)

De igual manera observaras también las bibliotecas necesarias que permiten la comunicación entre el código de Arduino ide y el InfluxDB Cloud

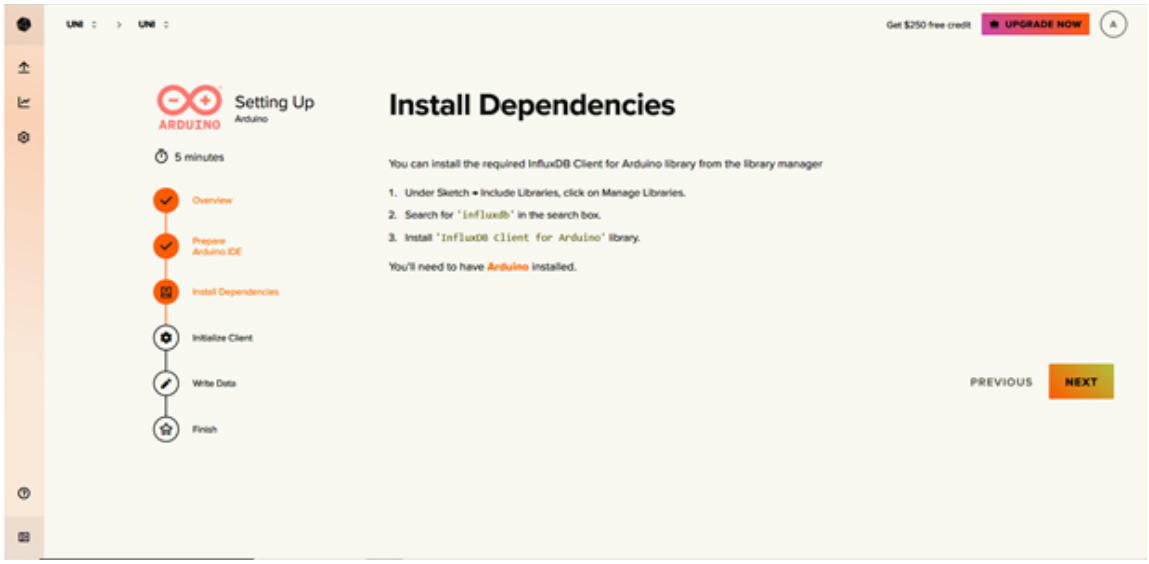


Fig. 33: Instalación de bibliotecas (InfluxData, Inc., 2025)

Para iniciar a guardar datos en el almacenamiento de InfluxDB Cloud debes primero crear un depósito (bucket) el cual, el cual se deberá identificar en el código de Arduino ide, además de poder configurar la retención total de datos, que en la versión que ocuparemos será de máximo 30 días

Fig. 34: Creación de depósitos (InfluxData, Inc., 2025)

Entonces luego de configurar InfluxDB Cloud, podemos observar una pestaña que muestra las credenciales de conexión y que ocuparemos en la lógica de nuestro código en Arduino ide, para establecer la comunicación en envió de datos, y este proceso crea un api token que es lo que permite a dispositivos y aplicaciones acceder a la base de datos de manera segura.

Fig. 35: credenciales iniciales (InfluxData, Inc., 2025)

5.4.2 Integración de base de datos de InfluxDB Cloud con Grafana Cloud

Una vez verificado el envío de datos, procedimos a comunicarlo con el plugin de grafana, asi mismo como se explicó en el apartado de influx db cloud, creamos una cuenta y una vez dentro el home del software, procedimos a configurar los paneles, para eso debemos indicarle a grafana la fuente de datos donde extraerá la información, en nuestro caso influx db cloud.

Ubicados en la sección de home> conexiones > fuentes de datos

Debemos configurar el nombre de nuestra fuente de datos con el propósito de poder identificarla, además de agregar el idioma de consulta (de programación), que en nuestro caso será “flux” que es uno de los idiomas de consulta que soporta influx db cloud y grafana

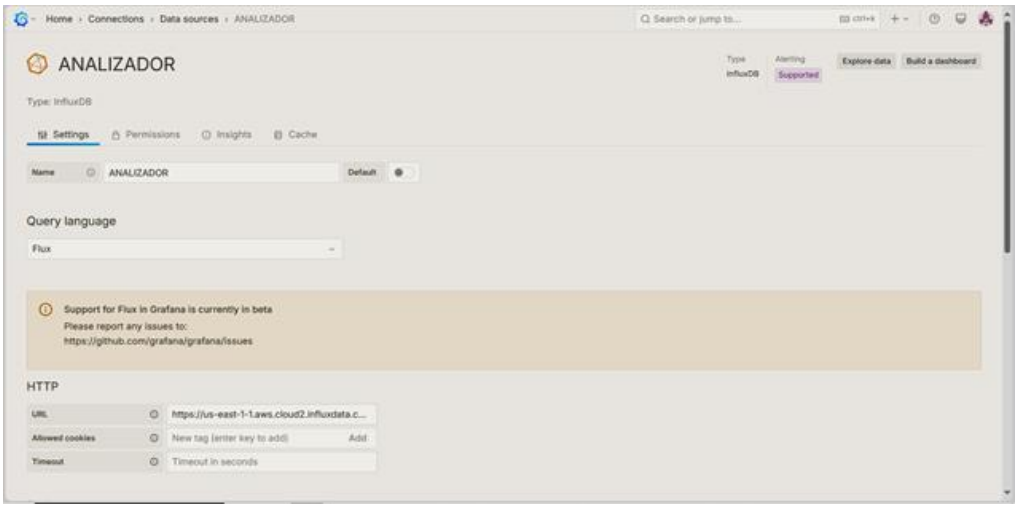


Fig. 36: Configuración inicial de fuente de datos (Grafana Labs, 2025)

Tambien debemos rellenar el apartado de detalles de influx con las credenciales de organización, el token que por motivo de seguridad una vez digitado muestra configurado, y el nombre de buquet o deposito creado en influx, todas estas configuraciones fueron creadas y estan declaradas en el codigo, y se puede apreciar en la figura 35.

Fig. 37: Configuración inicial de fuente de datos (Grafana Labs, 2025)

Luego de haber configurado exitosamente la fuente de datos, podemos hacer consultas para extraer de manera personalizada los datos basta con en un campo de visualización previamente creado, editarlo, y nos aparecerá la opción de agregar nuestra fuente de datos que recién configuramos.

5.5 Servidor web para interfaz gráfica

En la tarea llamada void setupServer() se inicializa el servidor que da lugar a la interfaz gráfica, esta permite configurar los parámetros esenciales para que el prototipo comience a crear archivos de los datos censados. La creación del servidor es gracias a la capacidad del ESP 32 de generar un punto de acceso el cual tiene una dirección IP estática (192.168.4.1), dando la posibilidad de que cualquier celular o laptop pueda conectarse al microcontrolador sin necesidad de un router. Este proceso lleva varios apartados, pero para comenzar es importante conceptualizar la palabra “rutas” en un servidor web. Según (Nunez, 2018), las rutas corresponden a una parte del URL que identifica el recurso que se desea obtener. Una vez explicado el termino, se entiende que el servidor maneja varias rutas para su funcionamiento, la ruta principal (http://192.168.4.1/) que corresponde al URL que muestra todas las opciones que se pueden configurar, la ruta para visualizar el logo de la UNI (/logo.png) extrae el logo desde la tarjeta SD y se muestra en el servidor, la ruta que escanea las redes Wi-Fi (/scanNetwork) es la que muestra las redes disponibles, el portal cautivo el cual redirige a la página principal si el usuario intenta acceder a un URL que no existe, la ruta para configurar el intervalo de grabado (/setRecordInterval) donde le permite al usuario asignar el tiempo en el que se desea registrar los datos medidos (días, horas, minutos e infinito) la opción infinito permite registrar los datos hasta que se desconecte el prototipo, la ruta para configurar el tiempo de censado (/setDelay) siendo el mínimo de 3 segundo cuando esté con Wi-Fi y 1 segundo cuando no está conectado a la red, la ruta para conectar a Wi-Fi (/connectWiFi) permite seleccionar el nombre de la red y escribir la contraseña para conectarse, por último, la ruta borrar credenciales de Wi-Fi (/clearWiFi) elimina las credenciales guardadas. En el **Anexo 13** se muestran las capturas del servidor con todas las funciones antes descritas.

6. ANALISIS Y PRESENTACIÓN DE RESULTADOS

La totalidad del diseño y desarrollo culmino exitosamente cumpliendo con todos los objetivos establecidos siendo esas capas de medir de manera funcional todos los parámetros eléctricos mencionados en los objetivos que fueron previamente contrastados con otros tipos de medidores disponibles en el laboratorio de Máquinas Eléctricas, ubicado UNI-RUSB, además de proporcionar la seguridad eléctrica necesaria a nivel prototipo y se puede apreciar en la figura 38.

6.1 Evidencia del Ensamblaje Final del Prototipo

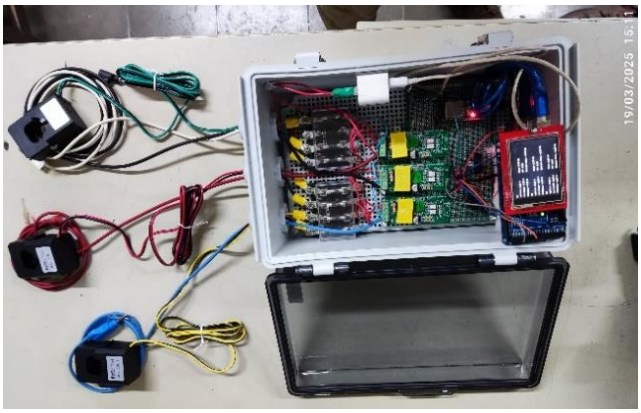


Fig. 38: Ensamblaje final del prototipo (Fuente Propia)

El **Anexo 14** contiene las diferentes etapas de construcción del prototipo final.

Consecuentemente a pesar que no forma parte de los objetivos se consideró necesario realizar algunas comparaciones entre distintos analizadores de redes, lo cual está presente en el **Anexo 15**. En el **Anexo 27** se presenta la tabla de precios donde se desglosan el valor de cada componente del prototipo.

6.2 Instalación del Prototipo en Tableros Eléctricos

6.2.1 Tablero Eléctrico Ubicado en la Residencia Estudiantil UNI-RUSB

Uno de los tableros asignados por el área administrativa fue el tablero que se muestra en el **Anexo 16**, se instaló el prototipo por 2 días con 4 horas y 40 minutos, este intervalo de tiempo se debe a la disponibilidad del personal técnico

para poder asistir como parte del procedimiento en la instalación. En donde se puede apreciar que se cumplió parte de los objetivos al ponerlo a prueba, mostrar el correcto y estable funcionamiento del equipo, se monitorearon y recopilaron los datos de la manera esperada, sin percances en ningún momento de la medición. Se muestran los resultados en un archivo CSV, el archivo se cargó en Google Drive, el link de acceso es el siguiente: https://drive.google.com/drive/folders/1DqCQ1quKmpJ38bfLeUsRBhdN2yf9-SxS?usp=drive_link. En este contienen todos los datos recopilados previamente establecidos en los objetivos de este trabajo monográfico, además se le agregó los mínimos, medios y máximos de cada parámetro tanto de la L1 como de la L2. También, se cargó un documento donde se adjuntaron los gráficos de cada parámetro extraídos desde Grafana, donde se pueden observar los comportamientos de ambas líneas.

6.2.1.1 Análisis de Voltajes L1-N y L2-N.

A continuación, se detalla el comportamiento del voltaje de fase registrado en el tablero eléctrico:

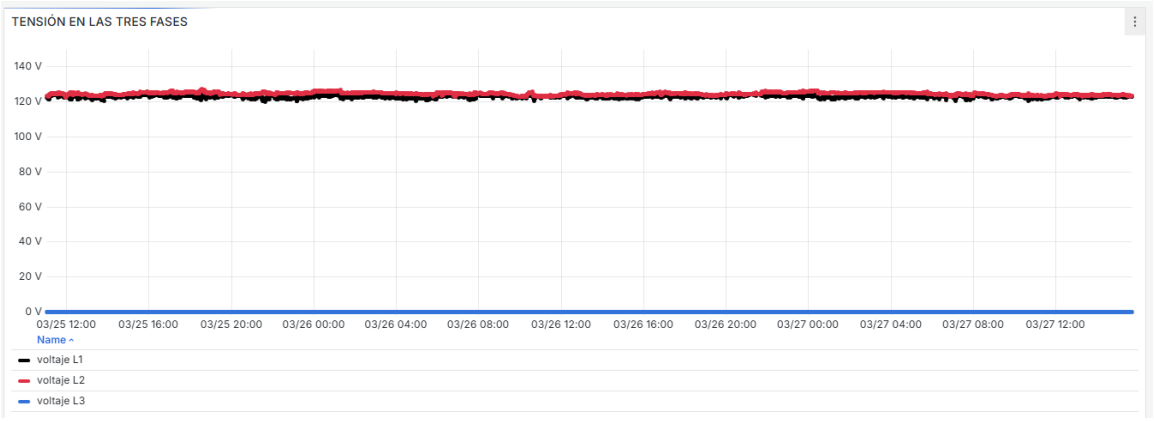


Fig. 39: Voltajes de fase alimentación de tablero. (Fuente Propia)

Según la figura 39, las mediciones de tensión reflejan un desbalance menor del 3% entre cada una de las fases, según la norma ANSI C84.1 (ANSI), dicho valor es adecuado para el sistema eléctrico. Cabe mencionar que la alimentación del tablero monofásico es de 120/240V.

6.2.1.2 Análisis de Corrientes de Fase.

Para analizar los desbalances en las mediciones realizadas, se analizó la situación de la carga en cada una de las fases que alimentan al tablero. A continuación, se presenta el gráfico de la carga dada en amperios para cada una de las fases:

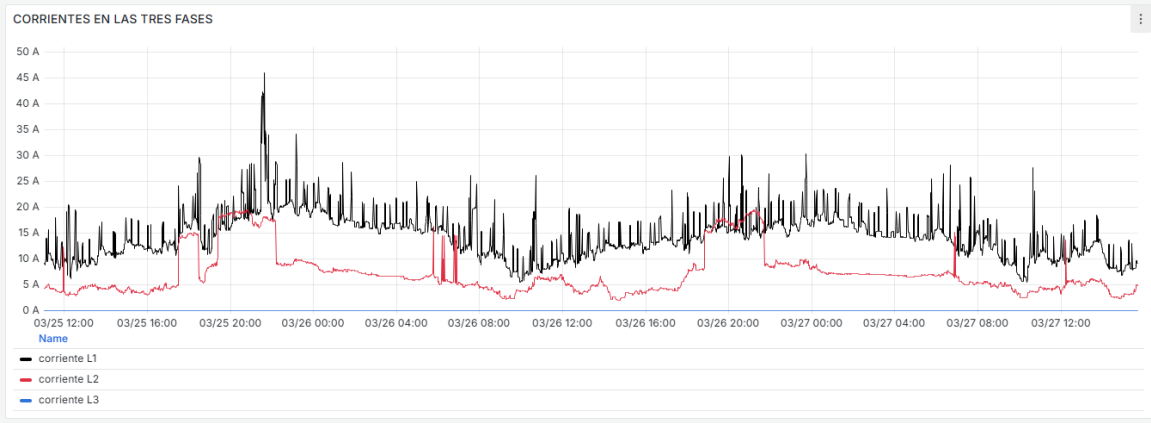


Fig. 40: Corrientes de fase. (Fuente Propia)

El desbalance promedio de corriente se calcula mediante la siguiente ecuación:

$$\%Desb.Prom.Corriente = \frac{C_{prom.mayor} - C_{prom.menor}}{C_{prom.mayor}} * 100$$

$$\%Desb.Prom.Corriente = \frac{14.59\ A - 7.16\ A}{14.59\ A} * 100 = 50.9\%$$

Los valores se extrajeron de la tabla de mediciones.

El porcentaje de desbalance promedio de corriente es de 50.9% aproximadamente, debido a que la línea 2 se encuentra sometida a menos demanda de corriente que la línea 1. Este porcentaje de desbalance es mayor al 5% que es el valor máximo de desbalance permitido por el NEC 2020 (National Fire Protection Association , 2020). Además, se aprecia que con respecto a la protección principal del tablero la corriente máxima se encuentra muy por debajo de la capacidad del mismo.

6.2.1.3 Análisis del Comportamiento de Demanda de las Tres Potencias (Activa, Reactiva y Aparente) por Fase.

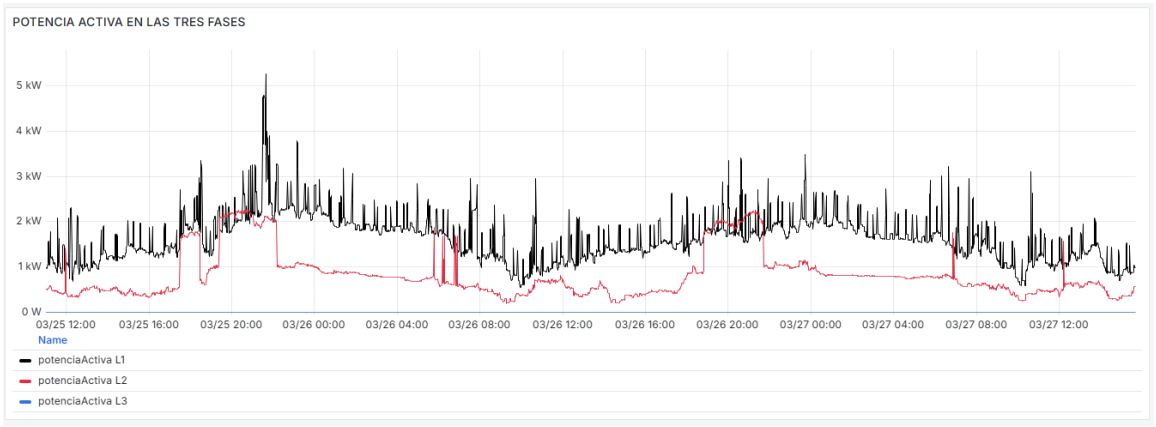


Fig. 41: Curvas de demanda de potencia activa por fase. (Fuente Propia)

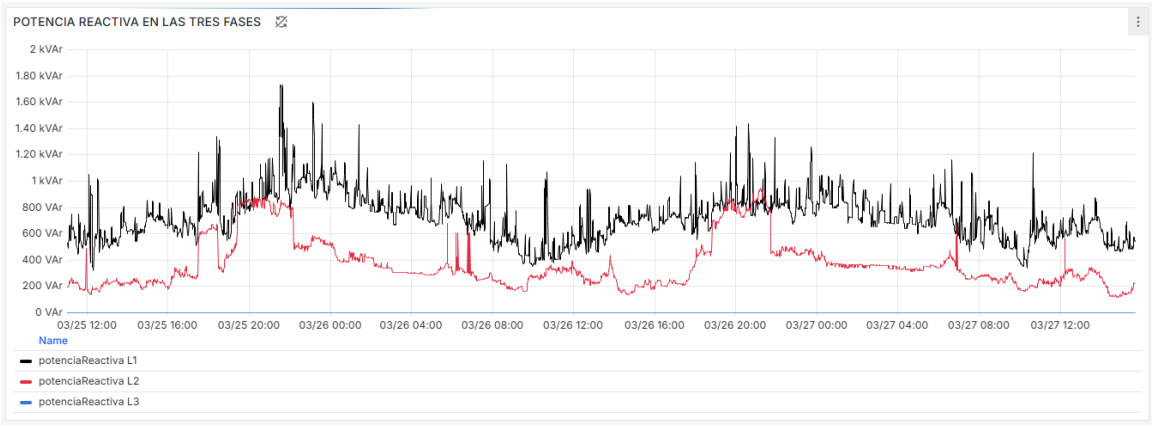


Fig. 42: Curvas de demanda de potencia reactiva por fase. (Fuente Propia)

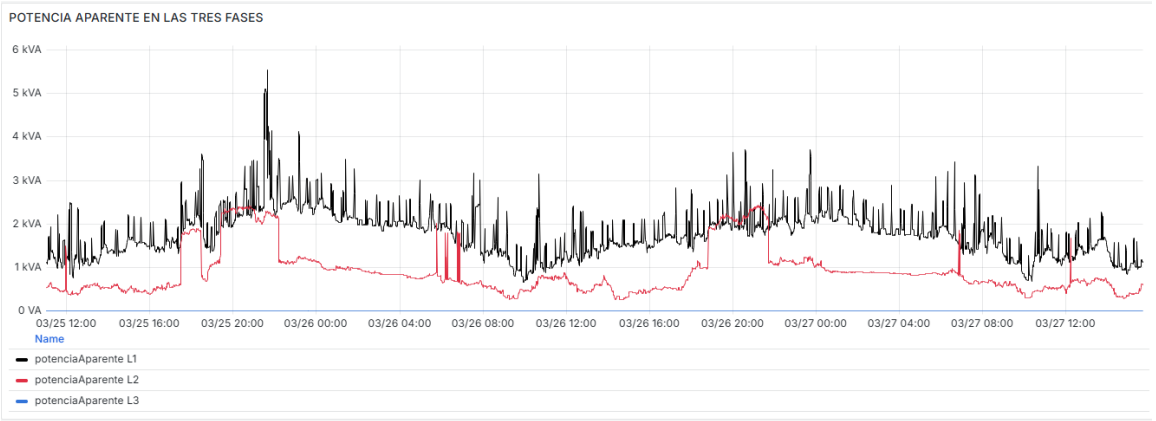


Fig. 43: Curvas de demanda de potencia aparente por fase. (Fuente Propia)

Como consecuencia del desbalance de las corrientes, se observan las curvas de demanda en todas las potencias, siendo mayor en la línea 1 con respecto a la línea 2. En relación a las características del tablero se determina que la demanda se encuentra muy por debajo de sus capacidades nominales.

6.2.1.4 Análisis del factor de potencia.

A continuación, se presenta el comportamiento del factor de potencia:

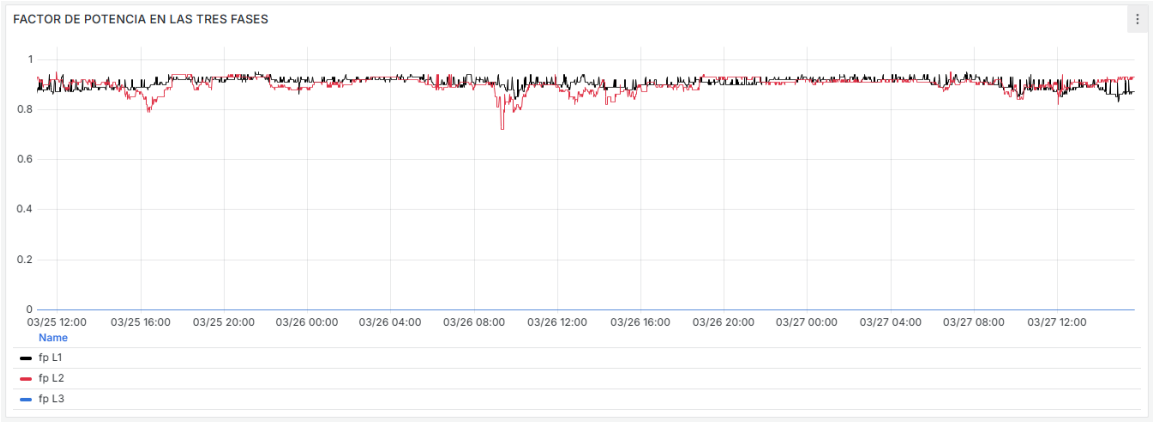


Fig. 44: Factor de potencia por fase. (Fuente Propia)

El factor de potencia se encuentra dentro del rango normal de operación establecido por la distribuidora de electricidad el cual no debe ser menor de 0.9, a pesar de que en la línea 2 baja considerablemente escasas veces, en promedio se mantiene dentro del rango permitido. Esto se puede observar en los valores dentro de las tablas de máximos, medios y mínimos en el archivo CSV en Google Drive, donde el valor promedio en cada línea es de 0.91 exactamente.

6.2.2 Tablero Eléctrico Ubicado en el Tercer Piso del Edificio de Ing. de Sistemas UNI-RUSB

El segundo tablero designado por el área administrativa fue el tablero mostrado en el **Anexo 17**, se instaló el prototipo por 2 días y 45 minutos, este intervalo de tiempo se debe a la disponibilidad del personal técnico y de la oficina donde está ubicado el tablero. En donde se puede apreciar que se cumplió parte de los objetivos al ponerlo a prueba, mostrar el correcto y estable funcionamiento del

equipo, se monitorearon y recopilaron los datos de la manera esperada, sin percances en ningún momento de la medición. Se muestran los resultados en un archivo CSV, el archivo se cargó en Google Drive, el link de acceso es el siguiente: https://drive.google.com/drive/folders/101gsCB55ZqnFzsRxWUVDWyPGRcBtcH_E?usp=drive_link. En este contienen todos los datos recopilados previamente establecidos en los objetivos de este trabajo monográfico, además se le agregó los mínimos, medios y máximos de cada parámetro tanto de la L1, L2 y L3. También, se cargó un documento donde se adjuntaron los gráficos de cada parámetro extraídos desde Grafana, donde se pueden observar los comportamientos de las tres líneas.

6.2.2.1 Análisis de Voltajes L1-N, L2-N y L3-N.

A continuación, se detalla el comportamiento del voltaje de fase registrado en el tablero eléctrico:

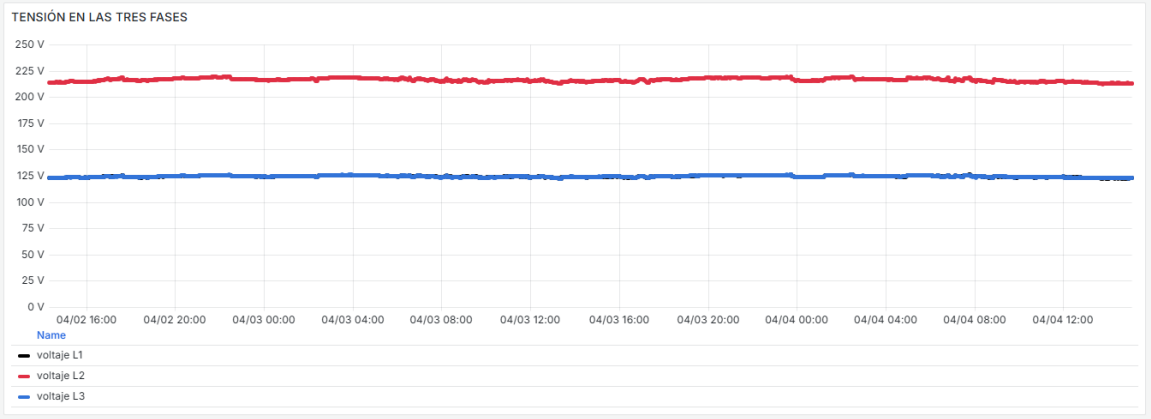


Fig. 45: Voltajes de fase alimentación de tablero. (Fuente Propia)

Según la figura 45, en este caso la alimentación del tablero trifásico es de 120/240V con línea griega 208V debido a conexión Delta aterrizado en el banco de transformadores. Las variaciones de tensión están dentro de lo normado, según la norma ANSI C84.1 (ANSI).

6.2.2.2 Análisis de Corrientes de Fase.

Para analizar los desbalances en las mediciones realizadas, se analizó la situación de la carga en cada una de las fases que alimentan al tablero. A continuación, se presenta el gráfico de la carga expresada en amperios para cada una de las fases:

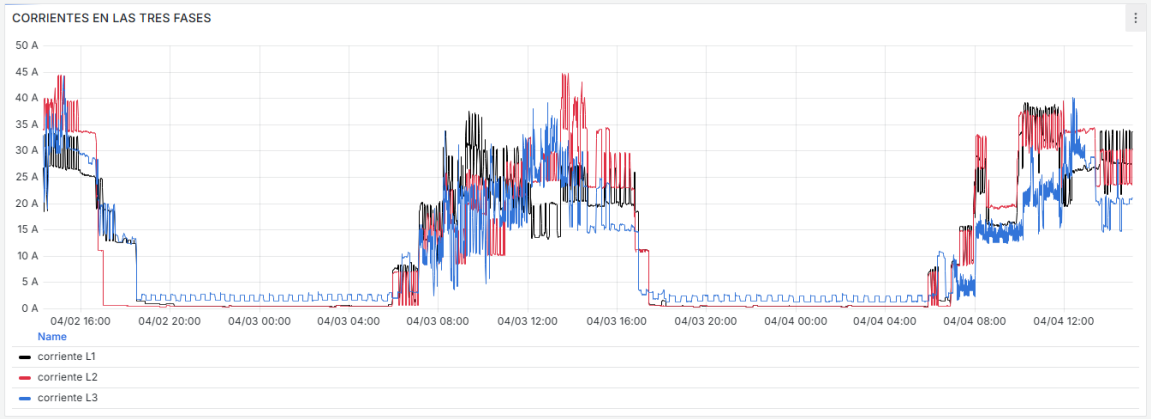


Fig. 46: Corrientes de fase. (Fuente Propia)

El desbalance promedio de corriente se calcula mediante la siguiente ecuación:

$$\%Desb.Prom.Corriente = \frac{C_{prom.mayor} - C_{prom.menor}}{C_{prom.mayor}} * 100$$

$$\%Desb.Prom.Corriente = \frac{11.4\ A - 10.21\ A}{11.5\ A} * 100 = 10.34\%$$

Los valores se extrajeron de la tabla de mediciones.

El porcentaje de desbalance promedio de corriente es de 10.34% aproximadamente, siendo la línea 3 la que se encuentra sometida a menos demanda de corriente que la línea 2. Este porcentaje de desbalance es mayor al 5% que es el valor máximo de desbalance permitido por el NEC 2014. Además, se aprecia que con respecto a la protección principal del tablero la corriente máxima se encuentra muy por debajo de la capacidad del mismo.

6.2.2.3 Análisis del Comportamiento de Demanda de las Tres Potencias (Activa, Reactiva y Aparente) por Fase.

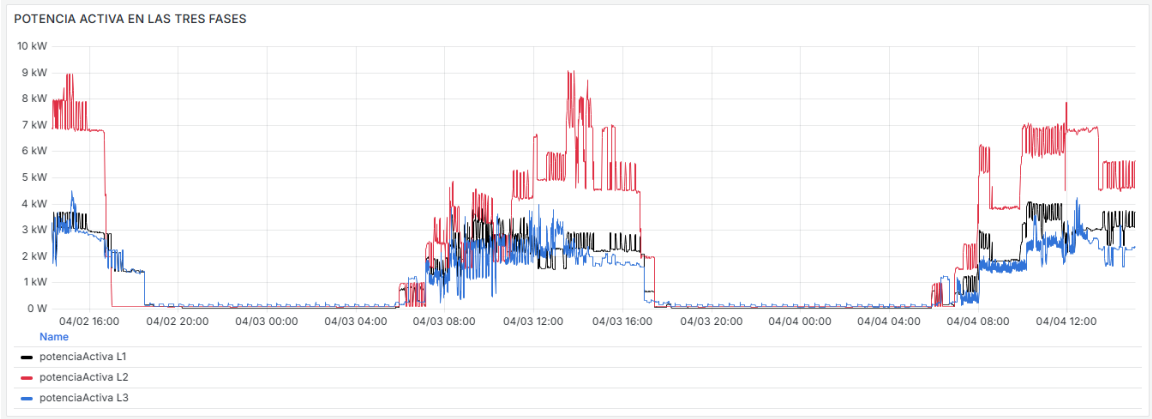


Fig. 47: Curvas de demanda de potencia activa por fase. (Fuente Propia)

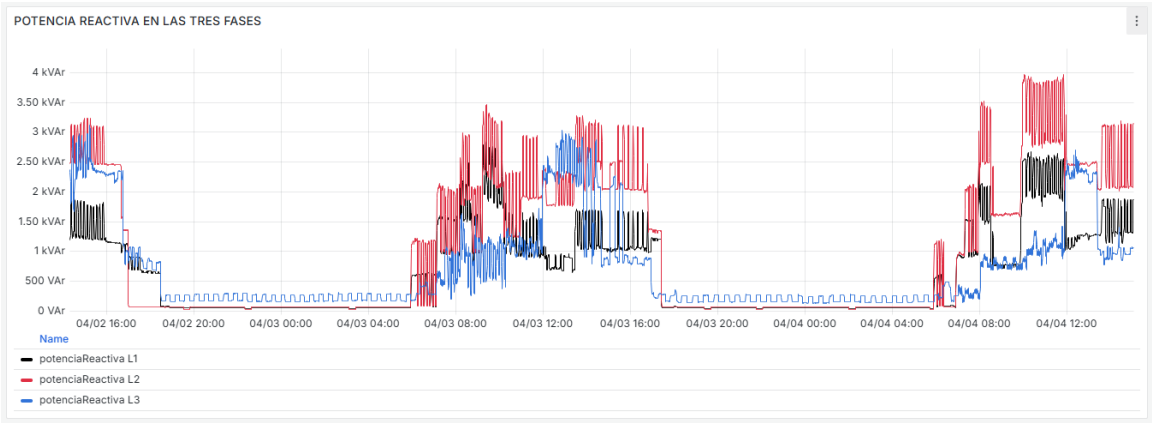


Fig. 48: Curvas de demanda de potencia reactiva por fase. (Fuente Propia)

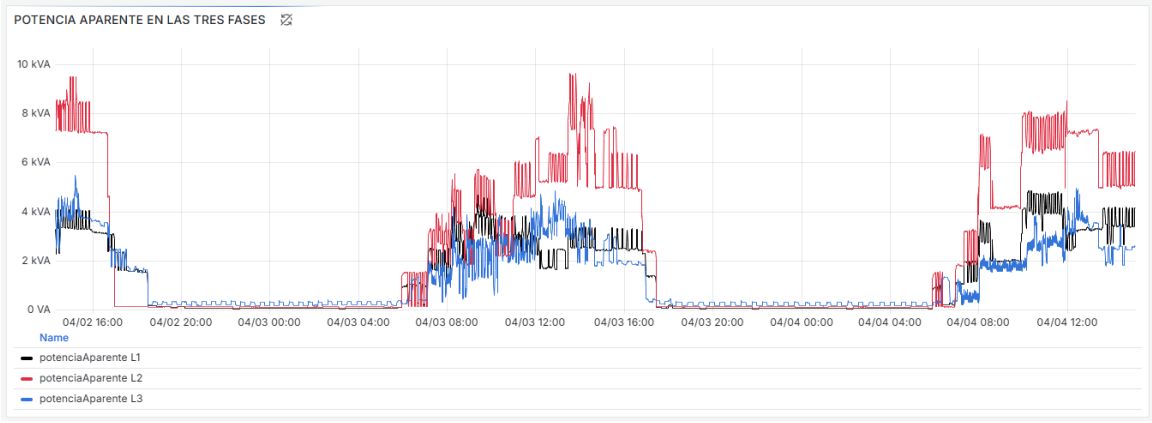


Fig. 49: Curvas de demanda de potencia aparente por fase. (Fuente Propia)

Debido al desbalance de las corrientes se observan curvas de demanda en todas las potencias mayor en la línea 2 siendo la línea griega con respecto a las demás líneas. En relación a las características del tablero se determina que la demanda se encuentra muy por debajo de sus capacidades nominales.

6.2.2.4 Análisis del factor de potencia.

A continuación, se presenta el comportamiento del factor de potencia:

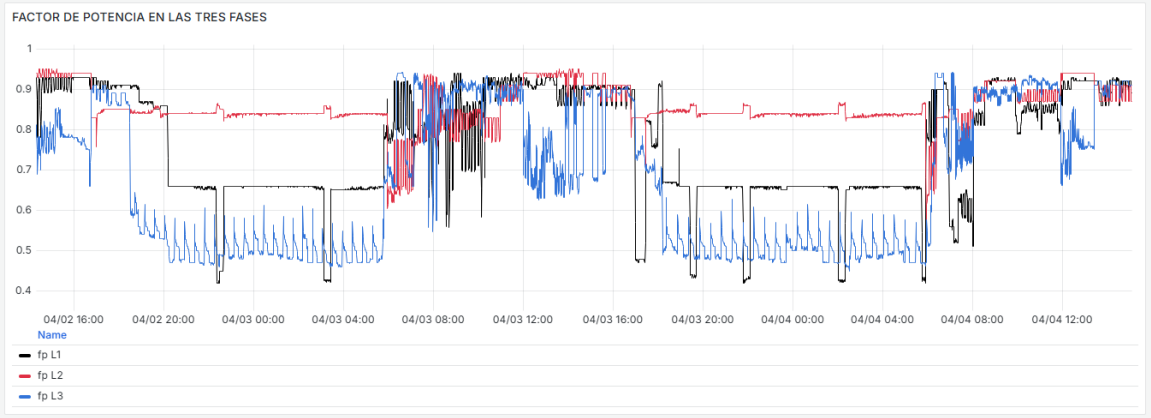


Fig. 50: Factor de potencia por fase. (Fuente Propia)

El factor de potencia tiene comportamientos irregulares, principalmente fuera del horario laboral. Se recomienda identificar las posibles cargas que presenten estas irregularidades, de esta manera se podría evitar un factor de potencia muy por debajo del permitido por la distribuidora. Esto se puede observar en los valores dentro de las tablas de máximos, medios y mínimos en el archivo CSV en Google Drive, donde el valor promedio trifásico es de 0.76 exactamente.

7. METODOLOGÍA Y DESARROLLO DEL TEMA

El diseño metodológico es fundamental para estructurar y desarrollar un prototipo exitosamente. En este caso, se ha determinado que la metodología mixta es la más conveniente, ya que permite combinar la recolección de datos cuantitativos como los son todos los parámetros eléctricos de sistema eléctrico y cualitativos como los son las percepciones acerca del prototipo, ofreciendo una visión completa del trabajo.

7.1 Identificación del Problema y Establecimiento de Objetivos

7.1.1 Recopilación de Información Preliminar

- Se llevó a cabo una serie de entrevistas semiestructuradas y grupos focales con parte del personal técnico y administrativo de la Universidad Nacional de Ingeniería en el Recinto Universitario Simón Bolívar (UNI-RUSB).
- Se comprobó el problema principal a través de la hipótesis: “falta de medición seccionada en el área administrativa dentro de las instalaciones de la UNI-RUSB”.

7.1.2 Cumplimientos de Objetivos

- Se definieron claramente los objetivos que se procuran cumplir con el trabajo realizado, al ejecutarse las mediciones necesarias en el panel eléctrico y habiendo analizado las respuestas obtenidas en las entrevistas y grupos focales.

7.2 Investigación y Revisión de Literatura

7.2.1 Indagación de Literatura

- Se revisaron monografías, artículos científicos y otras publicaciones sobre analizadores de redes eléctricas, tecnologías aplicables, tecnologías ya vigentes y sistemas de monitoreo de energía eléctrica.

- Se identificaron investigaciones previas relacionadas con prototipos de analizadores de redes trifásicas utilizando tecnologías IoT, con la finalidad de encontrar una oportunidad de innovación para sistemas trifásicos.

7.3 Evaluación y Análisis de Impacto

7.3.1 Recolección de Datos Cuantitativos y Cualitativos

- Datos cuantitativos: voltaje, corriente, potencia activa, potencia reactiva, potencia aparente y factor de potencia.
- Datos cualitativos: percepciones del personal técnico sobre la efectividad, precisión y utilidad del dispositivo; percepciones del personal administrativo sobre el impacto del monitoreo en la eficiencia energética y seguridad eléctrica.

7.3.2 Recolección de datos cuantitativos

- Con en el equipo de medición existente en el área de mantenimiento se realizaron mediciones en el panel donde se instaló el prototipo, de esta manera se determinaron los datos que se pretendían monitorear con el dispositivo.

7.3.3 Aplicación de las herramientas para la recolección de datos cualitativos

Se utilizaron herramientas como las entrevistas semiestructuradas (Véase **Anexo 18**) y grupos focales para profundizar en opiniones y sugerencias del personal administrativo y técnico.

En la entrevista que se realizó a un ingeniero eléctrico con 8 años de experiencia en el campo de diseño de sistemas eléctricos en baja y media tensión, además en el análisis de calidad de energía; explicó que para determinar el tiempo que se necesita instalar un analizador de redes en un sistema eléctrico será en dependencia del problema que se esté presentando y en los momentos del día en que se presenta mayor carga en el sistema, pero se concluyó que el tiempo promedio que se instala un analizador de redes es de 3 días como mínimo y que

según su experiencia y trayectoria la mayor cantidad de días de instalación fue de 7 días.

En búsqueda de una segunda opinión un ingeniero eléctrico especializado en el campo de análisis de calidad de energía; ratificó que en promedio el tiempo mínimo de instalación de estos dispositivos es de 3 días.

En coordinación con el responsable de la Unidad de Mantenimiento UNI-RUSB, se realizó una entrevista al Supervisor de Mantenimiento de la UNI-RUSB, el cual comentó que la principal medida de ahorro fue establecer un horario de encendido y apagado de las unidades de aires acondicionados. Se le cuestionó sobre la importancia de la constante supervisión del consumo por área, a lo cual respondió que lo mejor sería focalizarlo en las áreas donde se presente un consumo bastante elevado y que si se brindan varios equipos de medición como el prototipo sería vital para una cobertura de monitoreo más amplia. Por último, mencionó que una herramienta de medición más si es necesaria, ya que ayudaría a diversificar el control constante y verificar el cumplimiento del horario del encendido/apagado de los aires acondicionados.

Así mismo, se consultó al docente, MSc. Ing. Eléctrico, quien encontró potenciales beneficios para las mediciones dentro de la institución; considerando las capacidades de la tecnología IoT del prototipo, concluyó que dichas funcionalidades facilitaría el trabajo de recopilación de datos para el personal de mantenimiento eléctrico de la institución.

7.4 Desarrollo del Prototipo

7.4.1 Definición de Componentes

- Se seleccionaron los componentes necesarios, priorizando mantener bajos costos y alta calidad.
- Se implementaron tecnologías IoT para el envío inalámbrico de datos y su visualización mediante Grafana.

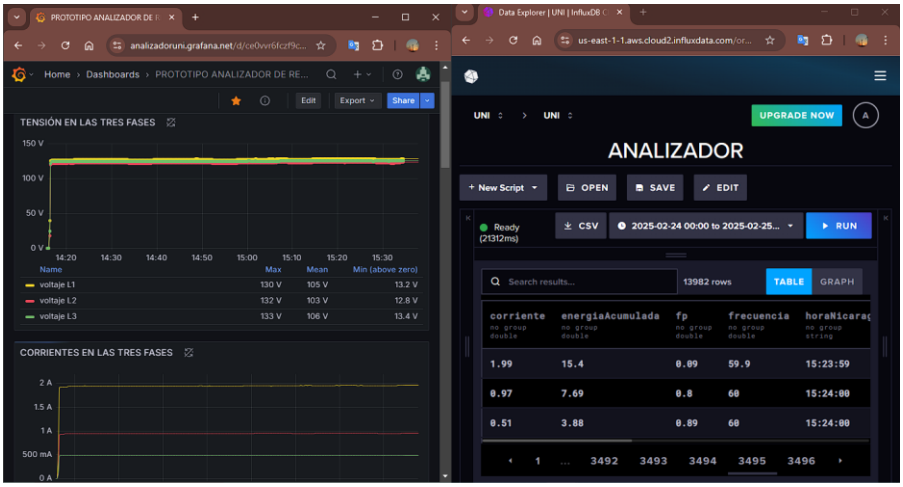


Fig. 51: Ejemplo de visualización en Grafana y almacenamiento de datos en InfluxDB. (Fuente Propia)

7.4.2 Construcción y Prueba del Prototipo:

- Se desarrollo del prototipo físico.

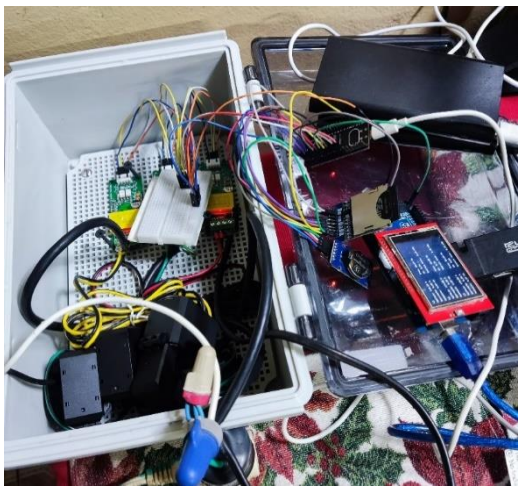


Fig. 52: Armado del dispositivo físico. (Fuente Propia)

- Se solicitó el uso del laboratorio de máquinas eléctricas para realizar pruebas. (Véase **Anexo 19**, **Anexo 20** y **Anexo 21**)
- Pruebas iniciales para determinar precisión y asegurar el correcto funcionamiento. (Véase **Anexo 24**)

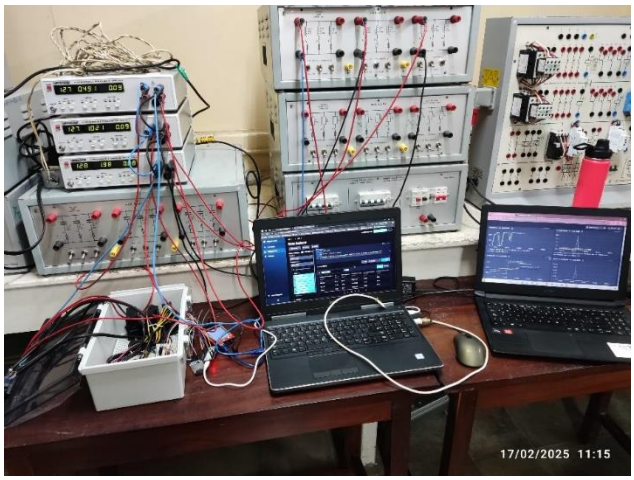


Fig. 53: Pruebas de laboratorio utilización el sistema de medición con el método de los tres vatímetros con tres PROGRAMMABLE POWER METERS HM8115-2. (Fuente Propia)

Fig. 54: Pruebas de laboratorio con un analizador de redes monofásico. (Fuente Propia)

- Se analizaron los resultados y se estableció que el dispositivo funciona correctamente. Además, se determinó el porcentaje de error comparado con diferentes dispositivos. (Véase **Anexo 24.5, 24.10, 24.15**)
- Además, se realizaron los cálculos basados en la teoría para determinar los valores de los parámetros eléctricos y se compararon con los datos medidos por el prototipo. (Véase **Anexo 24.16, 24.17, 24.18**).

7.5 Obtención de permisos, Instalación y Monitoreo

7.5.1 Solicitud de permisos:

- Se solicitó y se aprobó por parte logística y administrativa la instalación y pruebas del prototipo en tableros eléctricos. (Véase **Anexo 22** y **Anexo 23**)

7.5.2 Instalación del Prototipo:

- Se instaló del dispositivo en un tablero eléctrico donde se nos autorizó por parte del área administrativa de la UNI-RUSB. Primero se instaló en el tablero monofásico 120/240 V, ubicado dentro de la Residencia Estudiantil. Luego, se instaló en el tablero trifásico 120/240 V, ubicado en el tercer piso del Edificio de Ing. de Sistemas.
- Se obtuvieron los datos necesarios para llevar un control del consumo de energía eléctrica.

7.5.3 Monitoreo y Análisis de Datos:

- Monitoreo remoto en tiempo real mediante comunicación inalámbrica (WiFi).
- Se visualizaron los datos a través de Grafana. (Véase **Anexo 25**)

7.6 Procedimientos Operativos y Análisis Estadístico

7.6.1 Recolección de Datos:

- Recolección automática de datos eléctricos por el prototipo durante 2 días.
- Monitoreo y almacenamiento de los parámetros eléctricos en una base de datos para su posterior análisis. Como parte de nuestro compromiso con el responsable del área administrativa de la UNI-RUSB, se hizo entrega de un reporte con los resultados de las mediciones para contribuir a un posterior estudio más detallado de las instalaciones. (Véase **Anexo 26**)

7.6.2 Análisis Estadístico Descriptivo:

- Mediante del recurso de Excel se realizará un resumen del comportamiento de las magnitudes eléctricas medidas (mínimos, medios, máximos).

8. CONCLUSIONES Y RECOMENDACIONES

8.1 Conclusiones

Como resultado de todo lo antes evidenciado, el diseño y desarrollo de este proyecto logró cumplir de manera exitosa todos los objetivos inicialmente planteados en primer lugar. En efecto se logró desarrollar un prototipo de analizador de redes trifásicas con integraciones de tecnología (IoT) el cual facilito el análisis del comportamiento y el monitoreo de los parámetros eléctricos esenciales. Destacando también el correcto funcionamiento del almacenamiento tanto local como en la nube, además de transmisión en tiempo real.

La finalización de la construcción del prototipo incorporó sensores eléctricos que a través de su conversión de señales analógicas digitales permitieron efectuar el correcto y preciso censado de parámetros eléctricos, además mediante las capacidades del microcontrolador ESP32 se demostraron sus capacidades para ofrecer consistentes comunicaciones entre los distintos dispositivos electrónicos periféricos y entre los servicios (IoT) que estos a su vez proporcionaron de manera estable y eficaz las herramientas de almacenamiento, visualización en tiempo real, y monitoreo remoto. Por lo que estas características permiten diversificar y actualizar los métodos de obtención y monitoreo de remoto continuo de parámetros eléctricos en las instalaciones del área administrativa (UNI-RUSB).

Por otra parte, al ejecutar pruebas en condiciones de laboratorio se verificó la precisión, fiabilidad y seguridad del prototipo siendo esta parte fundamental de la investigación para destacar su funcionalidad y confiabilidad inclusive en implementaciones más comerciales en el campo de mediciones eléctricas.

En síntesis, este prototipo brinda una solución técnica y económica que pretende fomentar la capacidad de planeación de manteamientos preventivos, correctivos eficiencia energética, mejorando la productividad del personal al realizar los análisis de los parámetros eléctricos.

8.2 Recomendaciones

Como resultado del presente trabajo monográfico, se identificaron aspectos de mejora los cuales se pueden desarrollar en futuras implementaciones. Los cuales fortalecerán sus actuales capacidades y mejorando para entornos incluso comerciales e industriales.

Para empezar, recomendamos explorar distintas opciones de módulos que sean compatibles y cuenten con las capacidades de medir los fenómenos eléctricos de los armónicos, ofreciendo un prototipo más competitivo y robusto.

Se recomienda un sistema de respaldo integrado con baterías, en caso de ausencia de tensión, ofrezca alimentación completamente ininterrumpida. Esto con el fin de no interrumpir momentáneamente los procesos de cómputo en el prototipo, además de desarrollar un circuito que integre en su totalidad todos los periféricos agregados (pantalla, circuitos protección, sistema de respaldo)

La implementación de una aplicación que permita visualizar los datos y gráficos, compatible en todos los entornos que permita unificar todas las cualidades y capacidades que posee el prototipo, facilitando una experiencia intuitiva y amigable al usuario.

Además, para expandir aún más su campo de aplicación, se recomienda implementar circuitos integrados con sensores de tensión que posean mayor rango de medición. De esta manera tendría la capacidad para instalarse en tableros con una tensión de alimentación mayor.

Por último, se recomienda actualizar la pantalla por una de mayor tamaño y mejores características con el objetivo de optimizar la experiencia visual y mejorar la productividad.

9. BIBLIOGRAFÍA

Alexander, C. K., & Sadiku, M. O. (2006). *Fundamentos de circuitos eléctricos.* McGraw-Hill.

Aliexpress. (s.f.). *SD Adapter Micro SD Storage Expansion Board Micro SD TF Card Memory Shield Module SPI For Arduino*. Obtenido de https://www.aliexpress.us/item/3256802779477250.html?srcSns=sns_WhatsApp&spreadType=socialShare&bizType=ProductDetail&social_params=21687455841&aff_fcid=0133b556231b445696c65c10c95c7f91-1737046969065-09663-_mtQAnqx&tt=MG&aff_fsk=_mtQAnqx&aff_platform=defa

ANSI. (s.f.). *American National Standards Institute (ANSI).* Obtenido de https://www.ansi.org/?_gl=1*1o64sgz*_gcl_au*MTI1OTg3OTQ3OS4xNzQ0MjEyNDkw

Arduino. (2023). *Arduino Boards and Microcontrollers*. Obtenido de https://www.arduino.cc/

Arduino IDE. (5 de Febrero de 2018). *¿Qué es Arduino?* Obtenido de https://www.arduino.cc/en/Guide/Introduction?_gl=1*3lw6km*_up*MQ..*_ga*MTA1ODU2NDk5OC4xNzM3NTgwMjI1*_ga_NEXN8H46L5*MTczNzU4MDIyNC4xLjEuMTczNzU4MDIzOS4wLjAuMTcyMTY2Mzk1MQ..

Arduino. (s.f.). *Products*. Obtenido de Arduino Mega2560: https://store-usa.arduino.cc/products/arduino-mega-2560-rev3?selectedStore=us

aws.amazon. (s.f.). *¿Qué es IoT (Internet de las cosas)?* Obtenido de https://aws.amazon.com/es/what-is/iot/

Beal, V. (4 de enero de 2022). *Definiciones*. Obtenido de Wifi: https://www.webopedia.com/definitions/wifi/

Carmenate, J. G. (s.f.). *Progranarfacil.com*. Obtenido de https://programarfacil.com/esp8266/esp32/

Castro, I. C. (2020). *Analizador de Redes Trifásicas.* Catalunya.

Cayetano, A. J. (2020). *Prototipo de Medidor de Potencia de Bajo Coste con Aplicaión de Visualización en Dispositivos con Sistemas Operativos Android.* Sevilla: Universidad de Sevilla.

CEYUAN ELECTRONIC INSTRUMENT CO.,LTD. (s.f.). *Productos*. Obtenido de Analizadores de red: http://es.cyelectronic.com/network/used-network-analyzer-agilent-e8362c.html

CIRCUTOR. (s.f.). Obtenido de CIRCUTOR.COM : https://circutor.com/productos/medida-y-control/analizadores-de-redes-portatiles/product/M82523/

cruz, J. d. (2 de 02 de 2023). *jorgedela cruz.es*. Obtenido de Grafana: Monitorización gratuita, para siempre, usando InfluxDB Cloud, y Grafana Cloud: https://www.jorgedelacruz.es/2023/01/02/grafana-monitorizacion-gratuita-para-siempre-usando-influxdb-cloud-y-grafana-cloud/

Distron. (20 de 04 de 2022). *analizadores-redes-tipos.* Recuperado el junio de 2023, de https://distron.es/analizadores-redes-tipos/

Edge Delta. (3 de junio de 2024). Obtenido de https://edgedelta.com/company/blog/grafana-pros-and-cons

Electromer. (s.f.). *Modulos para arduino*. Obtenido de RTC DS3231: https://electromer.com.py/product/rtc-ds3231/

Electrositio.com. (2019). *Electrositio.com*. Recuperado el junio de 2023, de https://electrositio.com/que-es-el-analizador-de-redes-funcionamiento-y-sus-aplicaciones/

Espressif Systems. (2023). *ESP32 Technical Reference Manual*. Obtenido de https://www.espressif.com

Espressif Systems. (2023). *Espressif.* Obtenido de documentation: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

Fluke . (s.f.). *Fluke Corporation*. Obtenido de https://www.fluke.com/en/product/electrical-testing/power-quality/1732-1734

FLUKE . (s.f.). *Productos/Calidad eléctrica*. Obtenido de 437 Series II: https://www.fluke.com/es-bo/producto/comprobacion-electrica/calidad-electrica/437-series-ii

Grafana Labs. (2025). *Grafana Labs*. Obtenido de Grafana documentation: https://grafana.com/docs/grafana-cloud/

Grupo Novelec. (s.f.). Obtenido de Novelec : https://www.gruponovelec.com/p/hti_0179

influxdata documentacion. (s.f.). Obtenido de influx data: https://docs.influxdata.com/influxdb/cloud/account-management/limits/

InfluxData, Inc. (2025). *InfluxData, Inc.* Obtenido de InfluxData, Inc.: https://www.influxdata.com/products/influxdb-cloud/

influxData.inc. (s.f.). *influxdata* . Obtenido de https://docs.influxdata.com/influxdb3/cloud-serverless/query-data/sql/basic-query/

Innovators Guru. (2019). *PZEM-004T.* Obtenido de https://innovatorsguru.com/pzem-004t-v3/

KEYSIGHT. (s.f.). *Products*. Obtenido de Network Analyzers: https://www.keysight.com/us/en/products/network-analyzers/pna-network-analyzers/pna-x-network-analyzers.html

Merriam-Webster. (s.f.). *Dictionary*. Obtenido de Sensor: https://www.merriam-webster.com/dictionary/sensor

Microchip Technology. (2023). *PIC Microcontrollers Overview*. Obtenido de https://www.microchip.com/

Mischianti. (17 de Febrero de 2021). *DOIT ESP32 DEV KIT v1: especificaciones y pines de alta resolución.* Obtenido de https://mischianti.org/doit-esp32-dev-kit-v1-high-resolution-pinout-and-specs/

Nascimento, B. d., & Dantas Diógenes, D. P. (2020). Contador de magnitudes eléctricas con acceso remoto. *Revista Electrónica de Ingeniería Eléctrica e Ingeniería Mecánica*(ISSN 2763-5325), 12.

National Fire Protection Association . (2020). *National Electrical Code.* national fire protection fire .

Nunez, J. (2018). *¿Cómo se hace?* Obtenido de Concepto de Rutas en HTTP y REST: https://costaricamakers.com/concepto-de-rutas-en-http-y-rest/?utm_source=chatgpt.com

OPL Display. (01 de 03 de 2023). *OPL DISPLAY NOTICIAS*. Obtenido de ARTICULO 17512: https://www.opldisplaytec.com/article/17512

Paúl, S. B. (2022). *DESARROLLO DE UN MEDIDOR DE FACTOR DE POTENCIA BASADO EN INDUSTRIA 4.0 PARA EL CAMPUS MANUELA SÁENZ DE LA UNIVERSIDAD TECNOLÓGICA INDOAMÉRICA.* AMBATO.

Raspberry Pi Foundation. (2023). *Raspberry Pi Documentation*. Obtenido de https://www.raspberrypi.org

Rubio, A. (2019). *tipos-de-analizadores-de-red-cual-es-mejor*. (Instrumentación Digital) Recuperado el junio de 2023, de https://www.instrumentaciondigital.es/tipos-de-analizadores-de-red-cual-es-mejor/

Schneider electric. (s.f.). Obtenido de Schneider electric : https://www.se.com/es/es/product/METSEPM5100/pm5100-meter-without-communication-up-to-15th-h-1do-33-alarms/

SIGLENT. (s.f.). *Vector Network Analyzer*. Obtenido de SNA5000A Series Vector Network Analyzer: https://www.siglenteu.com/vector-network-analyzer/sna5000a-series/

ssdielect electronica sas. (s.f.). *Modulo pzem-004t medidor multifunción-UART*. Obtenido de https://ssdielect.com/magnitudes-electricas-1/168-md-pzem-004t.html

Staff, M. (18 de 04 de 2022). *Digikey*. Obtenido de MAKER.IO: https://www.digikey.com/en/maker/blogs/2022/arduino-cloud-overview-features-and-plans?utm_source=chatgpt.com

Sunfounder. (s.f.). *Components*. Obtenido de Component Mega2560: https://docs.sunfounder.com/projects/vincent-kit/en/latest/components/component_mega2560.html

ThingSpeak. (s.f.). Obtenido de ThingSpeak: https://thingspeak.mathworks.com/prices/thingspeak_student

todocoleccion. (s.f.). *Electricidad*. Obtenido de https://www.todocoleccion.net/antiguedades-tecnicas/antiguo-contador-luz-trifasico-marca-schlumberger~x344797308

10. ANEXOS

Anexo 1: Tabla comparativa de bases de datos. (influxdata documentacion, s.f.), (Staff, 2022), (ThingSpeak, s.f.)

Base de datos	Características	Ventajas	Desventajas	Limitaciones Versión Gratuita
INFLUXDB CLOUD	Base de datos orientadas a datos de series temporales. Diseñada para manejar grades cantidades de datos y metricas en tiempo real de diferentes sensores industriales etc	Alta eficiencia en la gestión y consulta de datos de series de tiempo. Mantenimiento y actualizaciones a cargo del proveedor. Librerias propias compatibles para el ESP 32 integracacion de herramienta de grafana	La versión gratuita tiene límites en velocidad de almacenamiento y retención de datos. Curva de aprendizaje intermedia en consultas, vinculacion con grafana	- 30 días de retención de datos. Entrada de datos : velocidad de 5 MB cada 5 minutos Lectura : Velocidad de 300 MB por 5 minutos Recursos disponibles : 2 cubos (excluyendo los cubos _monitoringy _tasks)
ARDUINO CLOUD	Plataforma en la nube para proyectos arduino arduino e IoT Permite creacion de dashboards para monitoreo y control	Integración directa con el ecosistema Arduino. Facil uso para prototipos IoT. Todo se puede gestionar en la propia nube. Integración nativa con ESP32 usando Arduino	El plan gratuito restringe el número de dispositivos y la cantidad de datos. resulta limitado y menos flexible para aplicaciones IoT de gran escala.	5 dispositivos. 5 variables 1 día de retención. 100MB/mes de datos salientes.
THINGSPEAK	Plataforma IoT en la nube para recopilar, almacenar y visualizar datos en canales. Soporta comunicación mediante HTTP y MQTT.	Configuración sencilla y abundante documentación. Dashboards básicos para visualización. Fácil comunicacion mediante HTTP o MQTT.	Limitaciones en la frecuencia de envío de datos (intervalos mínimos) y número de canales en la versión gratuita. Menos personalizable que otras plataformas de IoT.	numero de mensajes:3 millones/año (~8200/día) intervalo de actualizacion:Cada 15 segundos numero de canales: 4 numero de campos: 6

Anexo 2: Tabla comparativa de microcontroladores y plataformas de desarrollo (Arduino, 2023) (Espressif Systems, 2023) (Microchip Technology, 2023) (Raspberry Pi Foundation, 2023)

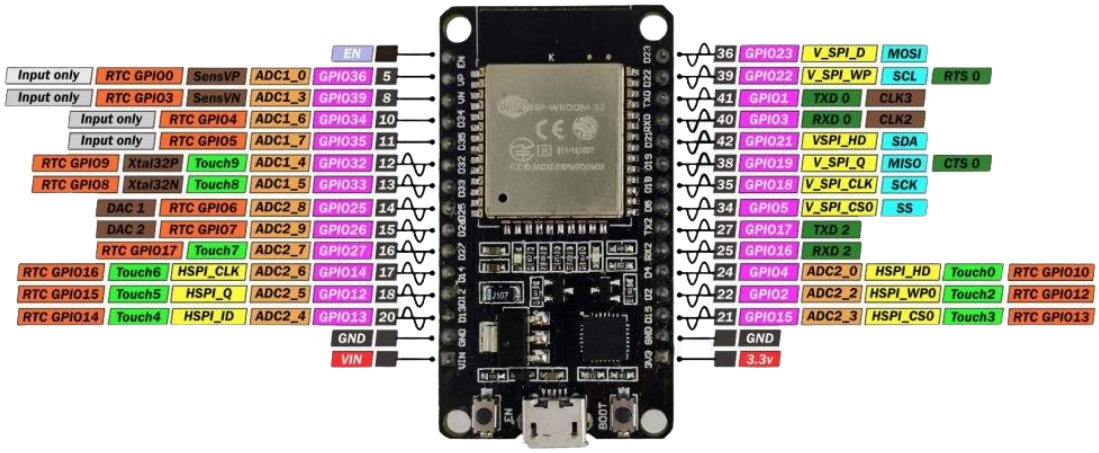
Dispositivo	Características	Ventajas	Desventajas
ESP-32S	Microcontrolador conocido por su capacidad de conectividad, con tecnología Wi-Fi y Bluetooth. Además, su bajo consumo energético, sobresale entre otros dispositivos.	Potente procesador, amplia gama de priféricos, flexibilidad en proyectos y bajo costo.	Programación más compleja y limitada ya que necesita ser programado en C/C++ o MicroPython.
Arduino MKR WIFI 1010	Plataforma de desarrollo con microcontroladores de SAMD12, perfecto para principiantes en la programación.	Compatible con la nube de Arduino IoT, extensa documentación de soporte, ideal para prototipos pequeños.	Menor capacidad de procesamiento, ha presentado problemas de conexión wifi, limitaciones de memoria.
PIC32MZ	Microcontroladores con un rendimiento equilibrado, con puerto ethernet y usb.	Bajo consumo de recursos, variedad de modelos, soporte para gráficos integrado y versatilidad en sus aplicaciones.	Programación más compleja, bajo rendimiento y no tiene conectividad a wifi ni bluetooth.
Raspberry Pi Pico W	Microcontrolador RP2040, es una tarjeta de desarrollo flexible, compacto.	Compatible con diversos lenguajes de programación y gran comunidad de soporte.	Menor capacidad de procesamiento, menor cantidad de pines analogicos y digitales.

Anexo 3: Hoja de datos del microcontrolador ESP-WROOM-32 (Espressif Systems, 2023)

Categories	Items	Specifications
Certification	RF certification	See certificates for ESP32-WROOM-32
	Wi-Fi certification	Wi-Fi Alliance
	Bluetooth certification	BQB
	Green certification	RoHS/REACH
Test	Reliablity	HTOL/HTSL/uHAST/TCT/ESD
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 μs guard interval support
	Center frequency range of operating channel	2412 ~ 2484 MHz
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and Bluetooth LE specification
	Radio	NZIF receiver with –97 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH
	Audio	CVSD and SBC
Hardware	Module interfaces	SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC, Two-Wire Automotive Interface (TWAI®), compatible with ISO11898-1 (CAN Specification 2.0)
	Integrated crystal	40 MHz crystal
	Integrated SPI flash	4 MB
	Operating voltage/Power supply	3.0 V ~ 3.6 V
	Operating current	Average: 80 mA
	Minimum current delivered by power supply	500 mA
	Recommended operating ambient temperature range	–40 °C ~ +85 °C
	Package size	18 mm × 25.5 mm × 3.10 mm
	Moisture sensitivity level (MSL)	Level 3

Anexo 4: Pin-out del microprocesador ESP32-WROOM

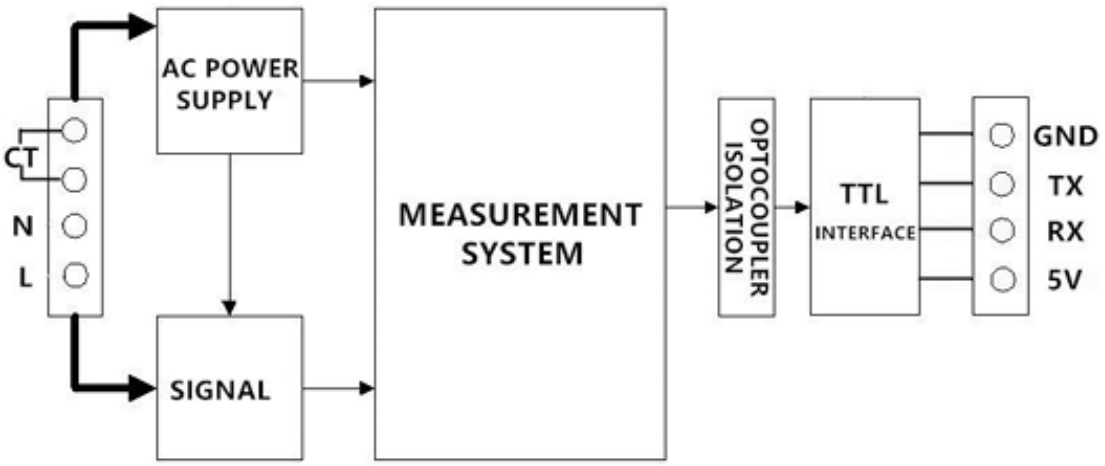
A continuación se presenta una breve descripción de cada uno de los pines del dispositivo:



- Pines GPIO: la placa ofrece numerosos pines de entrada/salida de propósito general (GPIO) que se pueden utilizar para diversas funciones de entrada/salida digital. Estos pines también admiten funciones como PWM, I2C, SPI y más.
- Entradas analógicas: varios pines del ESP32 DevKit V1 son capaces de leer señales analógicas, lo que los hace adecuados para interactuar con sensores analógicos.
- Pines de 3,3 V y GND: se utilizan para alimentar componentes o sensores externos.
- 5V y GND: La placa también puede proporcionar una salida de 5V, que es útil para alimentar módulos externos que requieren más energía.
- VIN: este es el pin de voltaje de entrada, que se puede utilizar para alimentar la placa cuando no se utiliza la conexión USB.
- ES: Este es el pin de habilitación. Se utiliza para reiniciar el microcontrolador.
- TX/RX: Estos pines se utilizan para la comunicación en serie.

- Interfaz SPI: La placa tiene pines para comunicación SPI, lo que permite una rápida transferencia de datos con periféricos como pantallas o memoria flash.
- Interfaz I2C: El ESP32 DevKit V1 admite la comunicación I2C, que se utiliza ampliamente para interactuar con sensores y otros periféricos.
- Pines del sensor táctil: algunos GPIO se pueden utilizar como entradas táctiles capacitivas, lo que ofrece una interfaz para dispositivos de entrada táctiles.
- VP/VN: Estos son los pines para el sensor de efecto Hall interno.
- Puente USB a UART: esta característica es crucial para programar el ESP32 mediante un cable USB y también para la comunicación en serie con una computadora u otros dispositivos host USB. (Mischianti, 2021)

Anexo 5: Hoja de datos del sensor PZEM-004T



1. Voltage

Measuring range:80～260V

Resolution: 0.1V

2 Current

Measuring range: 0～100A

Starting measure current: 0.02A

Resolution: 0.001A

Measurement accuracy: 0.5%

3 Active power

Measuring range: 0～23kW(PZEM-004T-100A)

Starting measure power: 0.4W

Resolution: 0.1W

Display format:

＜1000W, it display one decimal, such as: 999.9W

≥1000W, it display only integer, such as: 1000W

Measurement accuracy: 0.5%

4 Power factor

Measuring range: 0.00～1.00

Resolution: 0.01

Measurement accuracy: 1%

5 Frequency

Measuring range: 45Hz～65Hz

Resolution: 0.1Hz

Measurement accuracy: 0.5%

6 Active energy

Measuring range: 0～9999.99kWh

Resolution: 1Wh

Measurement accuracy: 0.5%

Display format:

＜10kWh, the display unit is Wh(1kWh=1000Wh), such as: 9999Wh

≥10kWh, the display unit is kWh, such as: 9999.99kWh

Reset energy: use software to reset.

Anexo 6: Especificaciones del adaptador de Tarjeta SD

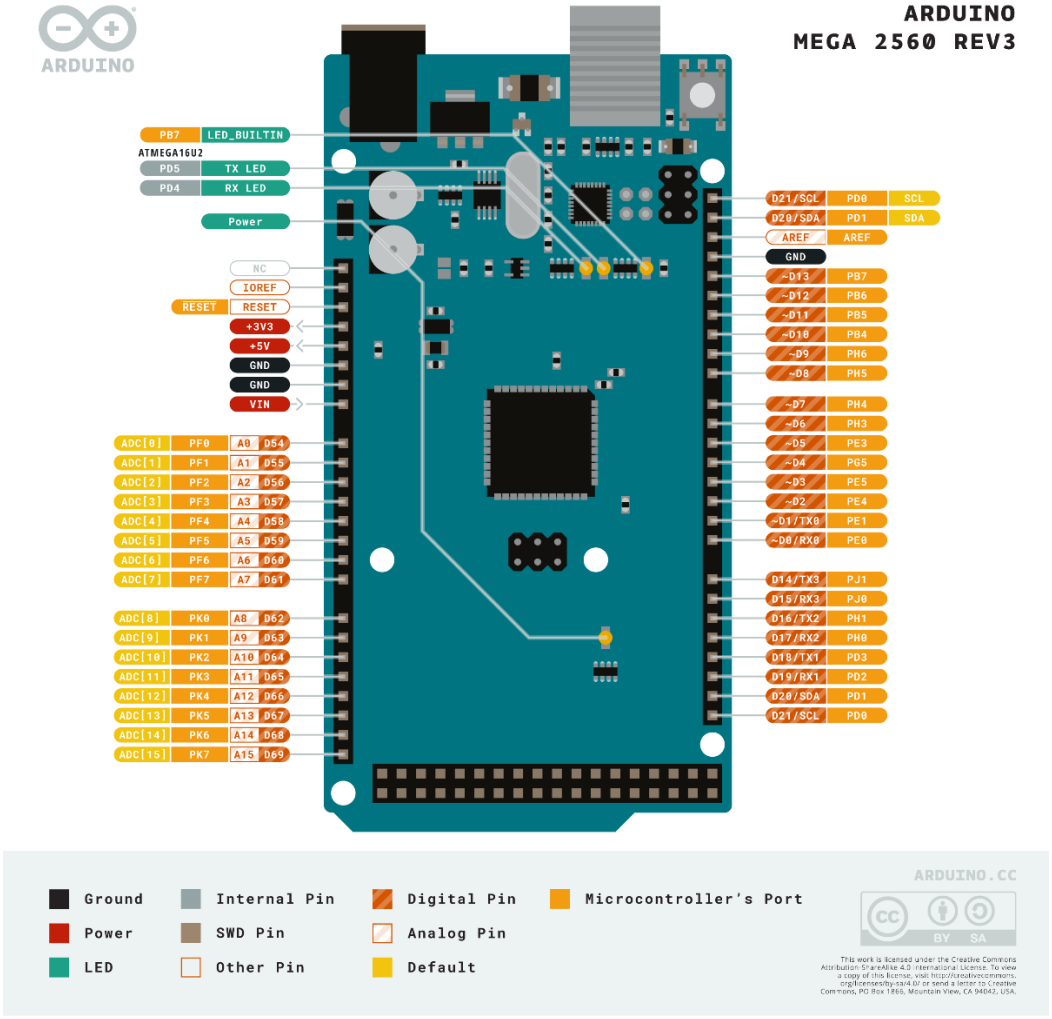
- Admite tarjeta Micro SD, tarjeta Micro SDHC (tarjeta de alta velocidad).
- La placa de circuito de conversión de nivel que puede interconectar el nivel es de 5 V o 3,3 V.
- La interfaz de comunicación es una interfaz SPI estándar.
- Interfaz de control: un total de seis pines (GND,VCC,MISO,MOSI,SCK,CS),GND a tierra.
- VCC es la fuente de alimentación, MISO,MOSI,SCK es el bus SPI,CS es el pin de señal de selección del chip.
- Circuito regulador de 3,3 V: salida del regulador LDO de 3,3 V como chip convertidor de nivel, suministro de tarjeta Micro SD.
- Circuito de conversión de nivel: tarjeta Micro SD en la dirección de las señales a 3,3 V, tarjeta MicroSD hacia la dirección de la interfaz de control La señal MISO también se convierte a 3,3 V, el sistema de microcontrolador AVR general puede leer la señal.
- Conector de tarjeta Micro SD: plataforma para facilitar la inserción y extracción de la tarjeta.
- Orificios de posicionamiento: 4 tornillos M2 con un diámetro de orificio de posicionamiento de 2,2 mm, posicionamiento fácil de instalar, para lograr una combinación entre módulos. (Aliexpress, s.f.)

Anexo 7: Hoja de datos del microprocesador Mega2560 (Arduino, s.f.)

Microcontroller	Mega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA

Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13

Anexo 8: Pin-out del microprocesador Mega2560 (Arduino, s.f.)



Anexo 9: Características del Reloj en Tiempo Real RTC DS3231

El DS3231 es un reloj en tiempo real (RTC) de bajo consumo de energía con un oscilador de cristal (TCXO) y cristales integrados compensados por temperatura. El dispositivo incorpora una entrada de batería y mantiene la hora exacta cuando se interrumpe la alimentación principal del dispositivo.

A continuación, se describen algunas de las características más importantes del DS3231:

- Precisión: El DS3231 tiene una precisión de ± 2 ppm de 0 ° C a + 40 ° C. Esto significa que la hora del reloj solo variará 2 segundos cada 3,16 años.
- Memoria interna: El DS3231 tiene una memoria interna de 32 bytes. Esta memoria se puede utilizar para almacenar datos, como la hora, la fecha o la configuración del reloj.
- Interfaz: El DS3231 utiliza la interfaz I2C. Esta interfaz es muy común y compatible con una amplia gama de microcontroladores.
- Voltaje de alimentación: El DS3231 funciona con un voltaje de alimentación de 3,3 V a 5 V. Esto lo hace compatible con una amplia gama de microcontroladores.

Voltaje de alimentación: El DS3231 funciona con un voltaje de alimentación de 3,3 V a 5 V. Esto lo hace compatible con una amplia gama de microcontroladores.

Anexo 10: Tablas de conexiones entre los componentes del prototipo.

Pantalla Shield TFT	Mega 2560
LCD_RST	Pin digital 8
LCD_CS	Pin digital 9
LCD_RS	Pin digital 10
LCD_WR	Pin digital 11
LCD_RD	5V
GND	GND
5V	5V
LCD_D0	Pin digital 22
LCD_D1	Pin digital 23
LCD_D2	Pin digital 24
LCD_D3	Pin digital 25
LCD_D4	Pin digital 26
LCD_D5	Pin digital 27
LCD_D6	Pin digital 28
LCD_D7	Pin digital 29

Mega 2560	ESP-32
TX1 (Pin 18)	RX0 (GPIO 3)
RX1 (Pin 19)	TX0 (GPIO 1)
GND	GND

Real Time Clock (RCT) DS3231	ESP-32
SCL	GPIO 22
SDA	GPIO 21
VCC	3.3 V
GND	GND

Adaptador SD	ESP-32
CS	GPIO 5
MOSI	GPIO 23
SCK	GPIO 18
MISO	GPIO 19
3.3 V	3.3 V
GND	GND

PZEM-004T	ESP-32
RX	TX2 (GPIO 17)
TX	RX2 (GPIO 16)
5 V	5V
GND	GND

Anexo 11: Código completo compilado en el ESP-32

```
#include <WiFi.h>
#include <WebServer.h>
#include <PZEM004Tv30.h>
#include <HTTPClient.h>
#include <InfluxDbCloud.h>
#include <Preferences.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <Wire.h>
#include <RTClib.h>
#include <SD.h>
#include <SPI.h>
#include <ArduinoJson.h>

// Parámetros de InfluxDB
#define INFLUXDB_URL "https://us-east-1-1.aws.cloud2.influxdata.com"
```

```
#define INFLUXDB_TOKEN
"T_FlzDcJvywbusJRcvqbXd9cR0sjp26UhpihYetC4kRMASUE3uI1A2GQHE0CxHKKBoK6-
udFd0M1MeleI-OusQ=="
#define INFLUXDB_ORG "2ac6b59b1078deff"
#define INFLUXDB_BUCKET "datossensores"
#define TZ_INFO "UTC-6"

// Configuración PZEM
#define PZEM_RX_PIN 16
#define PZEM_TX_PIN 17
#define PZEM_SERIAL Serial2
#define NUM_PZEMS 3
PZEM004Tv30 pzems[NUM_PZEMS];

// Configuración de la tarjeta SD
#define SD_CS_PIN 5
File archivoDatos;

// Variables para la comunicación con el Mega
#define MEGA_TX_PIN 1          // ESP32 TX conectado a Mega RX
#define MEGA_RX_PIN 3          // ESP32 RX conectado a Mega TX
HardwareSerial MegaSerial(1);  // Usar Serial1 para comunicación con el
Mega

// Variables globales
float energiaAcumulada[NUM_PZEMS] = { 0.0 };
float potencia[NUM_PZEMS] = { 0.0 };
unsigned long ultimaActualizacion[NUM_PZEMS] = { 0 };
unsigned long retrasoMedicion = 1;  // Tiempo de censado (inicialmente 1
s)

// Variables para medir el tiempo
uint64_t ultimaVezMillis[NUM_PZEMS] = { 0 };  // Último tiempo registrado
en milisegundos

// Control de grabación
bool recordInfinite = true;             // Inicia en modo infinito
bool isRecording = false;                // Indica si se está grabando
unsigned long recordIntervalMinutes = 0; // Minutos de grabación (si no es
infinito)
DateTime recordStartTime;               // Momento en que inició la
grabación
String currentFilename = "/mediciones_infinite.csv";  // Archivo actual

// Crear servidor web y prefrencias
```

```
WebServer server(80);
Preferences preferences;
RTC_DS3231 rtc;  // Crear un objeto para el RTC DS3231


// Configuración del cliente NTP
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", -21600, 60000);  // UTC-6
(Nicaragua)

// HTML para la configuración
const char *htmlForm = R"rawliteral(
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Configuraciones Analizador</title>
  <style>
    :root {
      --primary: #2563eb;
      --primary-dark: #1d4ed8;
      --background: #f3f4f6;
      --text: #1f2937;
      --error: #ef4444;
      --success: #22c55e;
    }

    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: system-ui, -apple-system, sans-serif;
      line-height: 1.5;
      background: var(--background);
      color: var(--text);
      padding: 1rem;
    }

    .container {
      max-width: 480px;
      margin: 0 auto;
```

```
      background: white;
      padding: 2rem;
      border-radius: 0.5rem;
      box-shadow: 0 1px 3px rgba(0,0,0,0.1);
    }

    h1 {
      font-size: 1.5rem;
      margin-bottom: 1.5rem;
      color: var(--primary);
    }

    h2 {
      font-size: 1.25rem;
      margin: 2rem 0 1rem;
    }

    .form-group {
      margin-bottom: 1.5rem;
    }

    label {
      display: block;
      margin-bottom: 0.5rem;
      font-weight: 500;
    }

    input, select {
      width: 100%;
      padding: 0.5rem;
      border: 1px solid #e5e7eb;
      border-radius: 0.375rem;
      font-size: 1rem;
      margin-bottom: 0.5rem;
    }

    input[type="checkbox"] {
      width: auto;
      margin-left: 0.5rem; /* Para separar la casilla de la palabra
"Infinito" */
    }

    input:focus, select:focus {
      outline: 2px solid var(--primary);
      outline-offset: 1px;
```

```
}

button, .button {
  display: inline-block;
  background: var(--primary);
  color: white;
  border: none;
  padding: 0.625rem 1.25rem;
  border-radius: 0.375rem;
  font-size: 1rem;
  cursor: pointer;
  transition: background-color 0.2s;
  width: 100%;
}

button:hover, .button:hover {
  background: var(--primary-dark);
}

.danger {
  background: var(--error);
}

.danger:hover {
  background: #dc2626;
}

.alert {
  padding: 0.75rem;
  border-radius: 0.375rem;
  margin: 1rem 0;
  display: none;
}

.alert.success {
  background: #dcfce7;
  color: #166534;
}

.alert.error {
  background: #fee2e2;
  color: #991b1b;
}

@media (max-width: 480px) {
```

```
      .container {
        padding: 1rem;
      }
    }
  </style>
</head>
<body>
  <div class="container">
    <img src="/logo.png" alt="Logo" style="display: block; margin: 0 auto;
max-width: 150px; height: auto;">

    <h1>Configuraciones Analizador</h1>

    <div id="statusAlert" class="alert"></div>

    <!-- Tiempo de Censado (delay) -->
    <form id="delayForm" class="form-group" onsubmit="updateDelay(event)">
      <h2>Tiempo de Censado</h2>
      <label for="delay">Intervalo (s):</label>
      <input
        type="number"
        id="delay"
        name="delay"
        value="1"
        min="%DELAY_MIN%"   //Aquí se permite 1 o 3 según WiFi
        required
      >
      <button type="submit">Actualizar Intervalo</button>
    </form>

    <!-- Grabado de Datos (con Días, Horas, Minutos e Infinito) -->
    <form id="recordIntervalForm" class="form-group"
onsubmit="updateRecordInterval(event)">
      <h2>Grabado de Datos</h2>
      <label for="recordDays">Días:</label>
      <input
        type="number"
        id="recordDays"
        name="recordDays"
        value="0"
        min="0"
        required
      >

      <label for="recordHours">Horas:</label>
```

```
  <input
    type="number"
    id="recordHours"
    name="recordHours"
    value="0"
    min="0"
    required
  >

  <label for="recordMinutes">Minutos:</label>
  <input
    type="number"
    id="recordMinutes"
    name="recordMinutes"
    value="1"
    min="0"
    required
  >

  <label for="recordInfinite">
    Infinito
    <input
      type="checkbox"
      id="recordInfinite"
      name="recordInfinite"
    >
  </label>

  <button type="submit">Actualizar Intervalo de Grabado</button>
</form>

<!-- Conexión WiFi -->
<form id="wifiForm" class="form-group" onsubmit="connectWiFi(event)">
  <h2>Conexión Wi-Fi</h2>
  <label for="ssid">Red:</label>
  <select id="ssid" name="ssid" required>
    %NETWORKS%
  </select>

  <label for="password">Contraseña:</label>
  <input
    type="password"
    id="password"
    name="password"
    required
```

```
      autocomplete="current-password"
    >
    <button type="submit">Conectar</button>
  </form>

  <!-- Mantenimiento -->
  <div class="form-group">
    <h2>Mantenimiento</h2>
    <button type="button" class="danger" onclick="clearCache()">
      Borrar Credenciales
    </button>
  </div>
</div>

<script>
  const statusAlert = document.getElementById('statusAlert');

  function showAlert(message, type = 'success') {
    statusAlert.textContent = message;
    statusAlert.className = `alert ${type}`;
    statusAlert.style.display = 'block';
    setTimeout(() => {
      statusAlert.style.display = 'none';
    }, 1000);
  }

  // Tiempo de Censado
  async function updateDelay(e) {
    e.preventDefault();
    const delay = document.getElementById('delay').value;

    try {
      const response = await fetch('/setDelay', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/x-www-form-urlencoded',
        },
        body: `delay=${delay}`
      });

      if (response.ok) {
        showAlert(`Tiempo actualizado: ${delay}s`);
      } else {
        throw new Error('Error al actualizar');
      }
```

```
      } catch (err) {
        showAlert(err.message, 'error');
      }
    }

    // Grabado de Datos (Días, Horas, Minutos o Infinito)
    async function updateRecordInterval(e) {
      e.preventDefault();
      const days = parseInt(document.getElementById('recordDays').value,
10);
      const hours = parseInt(document.getElementById('recordHours').value,
10);
      const minutes =
parseInt(document.getElementById('recordMinutes').value, 10);
      const isInfinite =
document.getElementById('recordInfinite').checked;

      let totalMinutes;
      if (isInfinite) {
        // -1 representará "Infinito" en el backend
        totalMinutes = -1;
      } else {
        totalMinutes = (days * 24 * 60) + (hours * 60) + minutes;
      }

      try {
        const response = await fetch('/setRecordInterval', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
          },
          body: `recordInterval=${totalMinutes}`
        });

        if (response.ok) {
          if (isInfinite) {
            showAlert('Intervalo de grabado actualizado: Infinito');
          } else {
            showAlert(`Intervalo de grabado actualizado: ${days}d
${hours}h ${minutes}m (${totalMinutes} min)`);
          }
        } else {
          throw new Error('Error al actualizar el intervalo de grabado');
        }
      } catch (err) {
```

```
        showAlert(err.message, 'error');
      }
    }

    async function connectWiFi(e) {
      e.preventDefault();
      const ssid = document.getElementById('ssid').value;
      const password = document.getElementById('password').value;

      try {
        const response = await fetch('/connectWiFi', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
          },
          body:
`ssid=${encodeURIComponent(ssid)}&password=${encodeURIComponent(password)}
`
        });

        if (response.ok) {
          showAlert(`Conectado a: ${ssid}`);
        } else {
          throw new Error('Error de conexión');
        }
      } catch (err) {
        showAlert(err.message, 'error');
      }
    }

    async function clearCache() {
      if (!confirm('¿Estás seguro? Esto borrará todas las credenciales
guardadas.')) {
        return;
      }

      try {
        const response = await fetch('/clearCache', {
          method: 'POST'
        });

        if (response.ok) {
          showAlert('Credenciales borradas');
        } else {
          throw new Error('Error al borrar');
```

```
        }
      } catch (err) {
        showAlert(err.message, 'error');
      }
    }
  </script>
</body>
</html>
)rawliteral";

// Estructura para almacenar datos de medición
struct DatosSensor {
  int indiceSensor;
  float voltaje;
  float corriente;
  float potenciaAparente;
  float frecuencia;
  float fp;
  float energiaAcumulada;
  float potenciaActiva;
  float potenciaReactiva;
};

QueueHandle_t dataQueue;

// Prototipos de funciones
void inicializarPZEMS();
void setupServer();
void tareasSensores(void *parametro);
void tareasInflux(void *parametro);
String getNetworksHTML();
void enviarMensajeMega(const String &mensaje);
void syncRTCWithNTP();
unsigned long getCurrentEpochTime();
void handleScanNetworks();
void resetEnergyAccumulated();

// Función para inicializar los sensores PZEM
void inicializarPZEMS() {
  for (int i = 0; i < NUM_PZEMS; i++) {
    pzems[i] = PZEM004Tv30(PZEM_SERIAL, PZEM_RX_PIN, PZEM_TX_PIN, 0x10 +
i);
    ultimaActualizacion[i] = millis();
    Serial.print("Sensor PZEM inicializado: ");
    Serial.println(i);
```

```
  }
}

// Función para reiniciar la energía acumulada
void resetEnergyAccumulated() {
  for (int i = 0; i < NUM_PZEMS; i++) {
    energiaAcumulada[i] = 0.0;
  }
  Serial.println("Energía acumulada reiniciada a 0.");
}

// Leer datos de los PZEM
void tareasSensores(void *parametro) {
  while (true) {
    String datosMega = "";          // Cadena para enviar al Mega
    bool todosVoltajesCero = true;  // verificar si todos los voltajes son
0
    String filaCSV = "";            // Para la fila del CSV con datos de
todos los sensores

    // Obtener la fecha y hora del RTC
    DateTime now = rtc.now();
    String fechaHora = String(now.year()) + "-" + String(now.month()) + "-
" + String(now.day()) + " " + String(now.hour()) + ":" +
String(now.minute()) + ":" + String(now.second());

    // Agregar fecha y hora al CSV
    filaCSV += fechaHora + ",";

   for (int i = 0; i < NUM_PZEMS; i++) {
      float voltaje = pzems[i].voltage();
      float corriente = pzems[i].current() * 3;
      float frecuencia = pzems[i].frequency();
      float fp = pzems[i].pf();
      fp = fp - (fp * 0.05);
      float potenciaAparente = voltaje * corriente;
      float potenciaActiva = potenciaAparente * fp;


      // Validar si los datos son NaN, reemplazar con 0 si es necesario
      voltaje = isnan(voltaje) ? 0 : voltaje;
      corriente = isnan(corriente) ? 0 : corriente;
      potenciaAparente = isnan(potenciaAparente) ? 0 : potenciaAparente;
      frecuencia = isnan(frecuencia) ? 0 : frecuencia;
      fp = isnan(fp) ? 0 : fp;
```

```
      potenciaActiva = isnan(potenciaActiva) ? 0 : potenciaActiva;

      if (voltaje > 0) {
        todosVoltajesCero = false;
      }else {
                // Si el voltaje es 0, establecer corriente y potencias a
0
                corriente = 0;
                potenciaAparente = 0;
                fp = 0;
           }

      uint64_t tiempoActualMillis = esp_timer_get_time() / 1000;
      float tiempoTranscurrido = (tiempoActualMillis - ultimaVezMillis[i])
/ 3600000.0f;  // Tiempo en horas
      ultimaVezMillis[i] = tiempoActualMillis;

      float incremento = potenciaActiva * tiempoTranscurrido;
      incremento = (incremento < 0 || incremento > 1000) ? 0 : incremento;
      energiaAcumulada[i] += incremento;

      // Crear estructura con datos del sensor
      DatosSensor datosSensor = {
        i, voltaje, corriente, potenciaAparente, frecuencia, fp,
energiaAcumulada[i],
        potenciaActiva, potenciaAparente * sqrt(1 - (fp * fp))
      };

      // Construir cadena para Mega
      datosMega += String(datosSensor.indiceSensor) + "," +
String(datosSensor.voltaje) + "," + String(datosSensor.corriente) + "," +
String(datosSensor.potenciaAparente) + "," +
String(datosSensor.frecuencia) + "," + String(datosSensor.fp) + "," +
String(datosSensor.energiaAcumulada) + "," +
String(datosSensor.potenciaActiva) + "," +
String(datosSensor.potenciaReactiva) + ";";

      // Enviar datos a la cola
      xQueueSend(dataQueue, &datosSensor, portMAX_DELAY);

      // Construir fila del CSV
      filaCSV += String(datosSensor.voltaje) + "," +
String(datosSensor.corriente) + "," + String(datosSensor.potenciaAparente)
+ "," + String(datosSensor.potenciaActiva) + "," +
String(datosSensor.potenciaReactiva) + "," +
```

```
String(datosSensor.frecuencia) + "," + String(datosSensor.fp) + "," +
String(datosSensor.energiaAcumulada) + ",";
    }

    // Remover la última coma y agregar un salto de línea
    if (filaCSV.length() > 0) {
      filaCSV = filaCSV.substring(0, filaCSV.length() - 1) + "\n";
    }

     // LÓGICA DE GRABACIÓN CONTROLADA POR INTERVALO
    if (isRecording) {
      // Si NO es infinito, se verifica si se cumplió el intervalo
      if (!recordInfinite) {
        TimeSpan tiempoTranscurrido = now - recordStartTime;
        unsigned long minutosTranscurridos =
tiempoTranscurrido.totalseconds() / 60;

        // Si ya pasaron los minutos configurados, se detiene la grabación
        if (minutosTranscurridos >= recordIntervalMinutes) {
          isRecording = false;
          Serial.println("Grabación automáticamente detenida. Intervalo
cumplido.");

          // se retorna a modo infinito
          recordInfinite = true;
          recordIntervalMinutes = 0;
          currentFilename = "/mediciones_infinite.csv";
          recordStartTime = now;   // Reiniciamos el tiempo base
          isRecording = true;      // Volvemos a grabar de inmediato en
modo infinito

          Serial.println("Regresando a grabación infinita en " +
currentFilename);
        }
      }

      // Si se sigue en modo grabación (sea infinito o no), se guardan
datos
      if (isRecording) {
        // Crear archivo con encabezados si no existe
        if (!SD.exists(currentFilename)) {
          archivoDatos = SD.open(currentFilename, FILE_WRITE);
          if (archivoDatos) {
            // Encabezados
            archivoDatos.println(
```

```
              "Fecha y Hora,"
              "Voltaje1 (V), Corriente1 (A), Potencia_Aparente1 (VA),
Potencia_Activa1 (W), Potencia_Reactiva1 (VAR), Frecuencia1 (Hz), FP1,
Energia1 (W/h),"
              "Voltaje2 (V), Corriente2 (A), Potencia_Aparente2 (VA),
Potencia_Activa2 (W), Potencia_Reactiva2 (VAR), Frecuencia2 (Hz), FP2,
Energia2 (W/h),"
              "Voltaje3 (V), Corriente3 (A), Potencia_Aparente3 (VA),
Potencia_Activa3 (W), Potencia_Reactiva3 (VAR), Frecuencia3 (Hz), FP3,
Energia3 (W/h)"
            );
            archivoDatos.close();
          }
        }

        // Agregar la fila de datos
        archivoDatos = SD.open(currentFilename, FILE_APPEND);
        if (archivoDatos) {
          archivoDatos.print(filaCSV);
          archivoDatos.close();
          Serial.println("Datos guardados en " + currentFilename);
        } else {
          Serial.println("Error al abrir el archivo de grabación: " +
currentFilename);
        }
      }
    }
// Si todos los voltajes son 0, se envía también la cadena completa de
ceros
    if (todosVoltajesCero) {
      datosMega = "0,0,0,0,0,0,0,0,0;";
    }
    // Enviar datos al Mega
    if (datosMega.length() > 0) {
      MegaSerial.println(datosMega);
      Serial.println("Enviado al Mega: " + datosMega);
    }
    // Calcular el tiempo de censado:
    unsigned long effectiveDelaySec;
    if (WiFi.status() == WL_CONNECTED) {
      effectiveDelaySec = (retrasoMedicion < 3) ? 3 : retrasoMedicion;
    } else {
      effectiveDelaySec = (retrasoMedicion < 1) ? 1 : retrasoMedicion;
    }
    vTaskDelay(pdMS_TO_TICKS(effectiveDelaySec * 1000));
```

```
  }
}

// Enviar datos a InfluxDB
void tareasSensores(void *parametro) {
  while (true) {
    String datosMega = "";          // Cadena para enviar al Mega
    bool todosVoltajesCero = true;  // verificar si todos los voltajes son
0
    String filaCSV = "";            // Para la fila del CSV con datos de
todos los sensores

    // Obtener la fecha y hora del RTC
    DateTime now = rtc.now();
    String fechaHora = String(now.year()) + "-" + String(now.month()) + "-
" + String(now.day()) + " " + String(now.hour()) + ":" +
String(now.minute()) + ":" + String(now.second());

    // Agregar fecha y hora al CSV
    filaCSV += fechaHora + ",";

   for (int i = 0; i < NUM_PZEMS; i++) {
      float voltaje = pzems[i].voltage();
      float corriente = pzems[i].current() * 3;
      float frecuencia = pzems[i].frequency();
      float fp = pzems[i].pf();
      fp = fp - (fp * 0.05);
      float potenciaAparente = voltaje * corriente;
      float potenciaActiva = potenciaAparente * fp;


      // Validar si los datos son NaN, reemplazar con 0 si es necesario
      voltaje = isnan(voltaje) ? 0 : voltaje;
      corriente = isnan(corriente) ? 0 : corriente;
      potenciaAparente = isnan(potenciaAparente) ? 0 : potenciaAparente;
      frecuencia = isnan(frecuencia) ? 0 : frecuencia;
      fp = isnan(fp) ? 0 : fp;
      potenciaActiva = isnan(potenciaActiva) ? 0 : potenciaActiva;

      if (voltaje > 0) {
        todosVoltajesCero = false;
      } else {
        // Si el voltaje es 0, establecer corriente y potencias a 0
        corriente = 0;
        potenciaAparente = 0;
```

```
        frecuencia = 0;
        fp = 0;
      }

      uint64_t tiempoActualMillis = esp_timer_get_time() / 1000;
      float tiempoTranscurrido = (tiempoActualMillis - ultimaVezMillis[i])
/ 3600000.0f;  // Tiempo en horas
      ultimaVezMillis[i] = tiempoActualMillis;

      float incremento = potenciaActiva * tiempoTranscurrido;
      incremento = (incremento < 0 || incremento > 1000) ? 0 : incremento;
      energiaAcumulada[i] += incremento;

      // Crear estructura con datos del sensor
      DatosSensor datosSensor = {
        i, voltaje, corriente, potenciaAparente, frecuencia, fp,
energiaAcumulada[i],
        potenciaActiva, potenciaAparente * sqrt(1 - (fp * fp))
      };

      // Construir cadena para Mega
      datosMega += String(datosSensor.indiceSensor) + "," +
String(datosSensor.voltaje) + "," + String(datosSensor.corriente) + "," +
String(datosSensor.potenciaAparente) + "," +
String(datosSensor.frecuencia) + "," + String(datosSensor.fp) + "," +
String(datosSensor.energiaAcumulada) + "," +
String(datosSensor.potenciaActiva) + "," +
String(datosSensor.potenciaReactiva) + ";";

      // Enviar datos a la cola
      xQueueSend(dataQueue, &datosSensor, portMAX_DELAY);

      // Construir fila del CSV
      filaCSV += String(datosSensor.voltaje) + "," +
String(datosSensor.corriente) + "," + String(datosSensor.potenciaAparente)
+ "," + String(datosSensor.potenciaActiva) + "," +
String(datosSensor.potenciaReactiva) + "," +
String(datosSensor.frecuencia) + "," + String(datosSensor.fp) + "," +
String(datosSensor.energiaAcumulada) + ",";
    }

    // Remover la última coma y agregar un salto de línea
    if (filaCSV.length() > 0) {
      filaCSV = filaCSV.substring(0, filaCSV.length() - 1) + "\n";
    }
```

```
    // LÓGICA DE GRABACIÓN CONTROLADA POR INTERVALO
    if (isRecording) {
      // Si NO es infinito, se verifica si se cumplió el intervalo
      if (!recordInfinite) {
        TimeSpan tiempoTranscurrido = now - recordStartTime;
        unsigned long minutosTranscurridos =
tiempoTranscurrido.totalseconds() / 60;

        // Si ya pasaron los minutos configurados, se detiene la grabación
        if (minutosTranscurridos >= recordIntervalMinutes) {
          isRecording = false;
          Serial.println("Grabación automáticamente detenida. Intervalo
cumplido.");

          // se retorna a modo infinito
          recordInfinite = true;
          recordIntervalMinutes = 0;
          currentFilename = "/mediciones_infinite.csv";
          recordStartTime = now;   // Reiniciamos el tiempo base
          isRecording = true;      // Volvemos a grabar de inmediato en
modo infinito

          Serial.println("Regresando a grabación infinita en " +
currentFilename);
        }
      }

      // Si se sigue en modo grabación (sea infinito o no), se guardan
datos
      if (isRecording) {
        // Crear archivo con encabezados si no existe
        if (!SD.exists(currentFilename)) {
          archivoDatos = SD.open(currentFilename, FILE_WRITE);
          if (archivoDatos) {
            // Encabezados
            archivoDatos.println(
              "Fecha y Hora,"
              "Voltaje1 (V), Corriente1 (A), Potencia_Aparente1 (VA),
Potencia_Activa1 (W), Potencia_Reactiva1 (VAR), Frecuencia1 (Hz), FP1,
Energia1 (W/h),"
              "Voltaje2 (V), Corriente2 (A), Potencia_Aparente2 (VA),
Potencia_Activa2 (W), Potencia_Reactiva2 (VAR), Frecuencia2 (Hz), FP2,
Energia2 (W/h),"
```

```
              "Voltaje3 (V), Corriente3 (A), Potencia_Aparente3 (VA),
Potencia_Activa3 (W), Potencia_Reactiva3 (VAR), Frecuencia3 (Hz), FP3,
Energia3 (W/h)"
            );
            archivoDatos.close();
          }
        }

        // Agregar la fila de datos
        archivoDatos = SD.open(currentFilename, FILE_APPEND);
        if (archivoDatos) {
          archivoDatos.print(filaCSV);
          archivoDatos.close();
          Serial.println("Datos guardados en " + currentFilename);
        } else {
          Serial.println("Error al abrir el archivo de grabación: " +
currentFilename);
        }
      }
    }

    // Enviar datos al Mega, incluyendo energía acumulada siempre
    if (datosMega.length() > 0 || todosVoltajesCero) {
      MegaSerial.println(datosMega);
      Serial.println("Enviado al Mega: " + datosMega);
    }

    // Calcular el tiempo de censado:
    unsigned long effectiveDelaySec;
    if (WiFi.status() == WL_CONNECTED) {
      effectiveDelaySec = (retrasoMedicion < 3) ? 3 : retrasoMedicion;
    } else {
      effectiveDelaySec = (retrasoMedicion < 1) ? 1 : retrasoMedicion;
    }
    vTaskDelay(pdMS_TO_TICKS(effectiveDelaySec * 1000));
  }
}

// Función para escanear redes Wi-Fi y manejar la solicitud
void handleScanNetworks() {
  int n = WiFi.scanNetworks();
  DynamicJsonDocument doc(2048);
  JsonArray networks = doc.to<JsonArray>();
  for (int i = 0; i < n; i++) {
    JsonObject network = networks.createNestedObject();
```

```
    network["ssid"] = WiFi.SSID(i);
    network["rssi"] = WiFi.RSSI(i);
  }
  WiFi.scanDelete();

  String response;
  serializeJson(doc, response);
  server.send(200, "application/json", response);
}

// Función para esperar y sincronizar el NTP una vez
void syncRTCWithNTP() {
  int attempts = 0;
  const int maxAttempts = 10;

  while (!timeClient.update() && attempts < maxAttempts) {
    delay(1000);  // Esperar a que NTP actualice el tiempo
    Serial.println("Esperando sincronización con NTP... Intento " +
String(attempts + 1));
    attempts++;
  }

  if (attempts < maxAttempts) {
    Serial.println("Sincronización NTP exitosa. Hora actual: " +
String(timeClient.getFormattedTime()));
    // Configurar el tiempo del ESP32 (RTC)
    configTime(-21600, 0, "pool.ntp.org");  // UTC-6 (Nicaragua)
    Serial.println("Hora configurada en RTC: " +
String(timeClient.getFormattedTime()));

    // Actualizar el tiempo del RTC DS3231
    if (rtc.begin()) {
      rtc.adjust(DateTime(timeClient.getEpochTime()));
      DateTime now = rtc.now();
      Serial.print("Hora almacenada en el RTC DS3231: ");
      Serial.print(now.year(), DEC);
      Serial.print('/');
      Serial.print(now.month(), DEC);
      Serial.print('/');
      Serial.print(now.day(), DEC);
      Serial.print(" ");
      Serial.print(now.hour(), DEC);
      Serial.print(':');
      Serial.print(now.minute(), DEC);
      Serial.print(':');
```

```
      Serial.println(now.second(), DEC);
    } else {
      Serial.println("Error al inicializar el RTC DS3231.");
    }
  } else {
    Serial.println("No se pudo sincronizar con NTP después de varios
intentos. Usando la hora del RTC DS3231.");

    if (rtc.begin()) {
      DateTime now = rtc.now();
      Serial.print("Hora recuperada del RTC DS3231: ");
      Serial.print(now.year(), DEC);
      Serial.print('/');
      Serial.print(now.month(), DEC);
      Serial.print('/');
      Serial.print(now.day(), DEC);
      Serial.print(" ");
      Serial.print(now.hour(), DEC);
      Serial.print(':');
      Serial.print(now.minute(), DEC);
      Serial.print(':');
      Serial.println(now.second(), DEC);

      // Configurar el tiempo del ESP32 usando el tiempo del RTC DS3231
      struct tm t;
      t.tm_year = now.year() - 1900;
      t.tm_mon = now.month() - 1;
      t.tm_mday = now.day();
      t.tm_hour = now.hour();
      t.tm_min = now.minute();
      t.tm_sec = now.second();
      time_t rtcTime = mktime(&t);
      timeval tv = { rtcTime, 0 };
      settimeofday(&tv, NULL);
    } else {
      Serial.println("Error al leer el RTC DS3231.");
    }
  }
}

// Función para obtener la hora actual en epoch
unsigned long getCurrentEpochTime() {
  time_t now;
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
```

```
    Serial.println("Error obteniendo la hora local.");
    return 0;
  }
  now = mktime(&timeinfo);
  return now;
}

// Función para formatear un intervalo de tiempo dado en minutos
String formatInterval(unsigned long totalMinutes) {
    unsigned long days = totalMinutes / (24 * 60);
    unsigned long hours = (totalMinutes % (24 * 60)) / 60;
    unsigned long minutes = totalMinutes % 60;
    // Retorna el intervalo formateado como "días horas minutos"
    return String(days) + "d " + String(hours) + "h " + String(minutes) +
"m";
}

// Función para configurar el servidor web
void setupServer() {
  const char *defaultSSID = "ANALIZADOR DE REDES - UNI";
  const char *defaultPassword = "12345678";
  WiFi.mode(WIFI_AP_STA);
  WiFi.softAP(defaultSSID, defaultPassword);
  IPAddress apIP(192, 168, 4, 1);

  // Configurar la red del AP
  WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));

  Serial.print(F("IP del punto de acceso: "));
  Serial.println(WiFi.softAPIP());

  // Ruta principal "/", al acceder a la ruta principal envia al servidor
web
  server.on("/", HTTP_GET, []() {
    // Cargar la plantilla base (htmlForm)
    String html = FPSTR(htmlForm);

    // Si hay conexión WiFi, mínimo = 3, si no, mínimo = 1
    if (WiFi.status() == WL_CONNECTED) {
      html.replace("%DELAY_MIN%", "3");
    } else {
      html.replace("%DELAY_MIN%", "1");
    }
```

```
    // Reemplaza %NETWORKS% ubicado en el HTML, por la función
getNetworksHTML
    html.replace("%NETWORKS%", getNetworksHTML());

    // Enviar el HTML final al cliente
    server.send(200, "text/html", html);
  });


  // Servir la imagen desde la SD
  server.on("/logo.png", HTTP_GET, []() {
    File file = SD.open("/logo.png");
    if (!file) {
      server.send(404, "text/plain", "Imagen no encontrada");
      return;
    }
    server.streamFile(file, "image/png");
    file.close();
  });

  // Endpoint para devolver las redes disponibles
  server.on("/scanNetworks", HTTP_GET, []() {
    handleScanNetworks();
  });

  // Manejar solicitudes no encontradas (Portal Cautivo)
  server.onNotFound([]() {
    server.sendHeader("Location", "http://192.168.4.1/", true);
    server.send(302, "text/plain", "Redirigiendo al portal...");
  });


  // Manejar la Solicitud POST para Configurar Intervalo:
server.on("/setRecordInterval", HTTP_POST, []() {
  //Leer el parámetro enviado desde el formulario
  String recordIntervalParam = server.arg("recordInterval");
  long totalMinutes = recordIntervalParam.toInt();

  //Decidir si es infinito o un valor en minutos
  recordInfinite = (totalMinutes == -1);
  recordIntervalMinutes = recordInfinite ? 0 : totalMinutes;

  //Obtener hora actual
  DateTime now = rtc.now();
```

```
  if (recordInfinite) {
    // Modo infinito en archivo fijo
    currentFilename = "/mediciones_infinite.csv";
  } else {
    // Modo finito en archivo nuevo
    char filename[50];
    sprintf(filename, "/mediciones_%04d%02d%02d_%02d%02d%02d.csv",
            now.year(), now.month(), now.day(),
            now.hour(), now.minute(), now.second());
    currentFilename = String(filename);
  }

  //Iniciar la grabación
  isRecording = true;
  recordStartTime = now;
  resetEnergyAccumulated();  // Reiniciar la energía acumulada

  server.send(200, "text/plain", "OK");
  Serial.println("Grabación configurada. Modo infinito? " +
String(recordInfinite ? "Sí" : "No"));

  // Enviar mensaje al Mega para mostrar el cambio de intervalo de grabado
de datos
  if (recordInfinite) {
    enviarMensajeMega("Intervalo de grabado cambiado a: Infinito");
  } else {
    enviarMensajeMega("Intervalo de grabado cambiado a: " +
formatInterval(recordIntervalMinutes));
  }
});

  server.on("/setDelay", HTTP_POST, []() {
    if (server.hasArg("delay")) {
      String delayValue = server.arg("delay");
      unsigned long newDelay = delayValue.toInt();

      // Validar el valor mínimo según el estado del WiFi:
      if (WiFi.status() == WL_CONNECTED) {
        if (newDelay < 3) {
          newDelay = 3;  // Mínimo 3 s si hay conexión a internet
        }
      } else {
        if (newDelay < 1) {
          newDelay = 1;  // Mínimo 1 s si no hay conexión
        }
```

```
      }
      retrasoMedicion = newDelay;
      String message = "Nuevo tiempo de censado: " +
String(retrasoMedicion) + " s";
      server.send(200, "text/html", "<h1>" + message + "</h1><p><a
href='/'>Volver</a></p>");
      Serial.println(message);

    // Envia mensaje al Mega para mostrar el cambio de tiempo de intervalo
    enviarMensajeMega("Intervalo de tiempo cambiado a: " +
String(retrasoMedicion) + " s");
  } else {
      server.send(400, "text/html", "<h1>Error: No se recibió el parámetro
'delay'</h1>");
    }
  });

  // Manejar el POST para conectar a Wi-Fi
  server.on("/connectWiFi", HTTP_POST, []() {
    String ssid = server.arg("ssid");
    String password = server.arg("password");

    // Guardar las credenciales en memoria no volátil
    preferences.begin("wifi-creds", false);
    preferences.putString("ssid", ssid);
    preferences.putString("password", password);
    preferences.end();

    Serial.println(F("Credenciales Wi-Fi guardadas."));

    WiFi.begin(ssid.c_str(), password.c_str());

    int attempts = 0;
    while (WiFi.status() != WL_CONNECTED && attempts < 30) {
      delay(1000);
      attempts++;
      Serial.print(F("."));
    }

    if (WiFi.status() == WL_CONNECTED) {
      server.send(200, "text/html", "<h1>Conectado a Wi-Fi: " + ssid +
"</h1><p><a href='/'>Volver</a></p>");
      Serial.println(F("\nConexion exitosa a Wi-Fi."));

      // Enviar mensaje al Mega para mostrar el cambio de red Wi-Fi
```

```
    enviarMensajeMega("Cambio de red Wi-Fi a: " + ssid);
    } else {
      server.send(400, "text/html", F("<h1>Error: No se pudo conectar a la
red Wi-Fi</h1>"));
    }
  });

  // Manejar el POST para eliminar las credenciales Wi-Fi
  server.on("/clearWiFi", HTTP_POST, []() {
    preferences.begin("wifi-creds", false);
    preferences.clear();
    preferences.end();
    server.send(200, "text/html", F("<h1>Credenciales Wi-Fi
eliminadas</h1><p><a href='/'>Volver</a></p>"));
    Serial.println(F("Credenciales Wi-Fi eliminadas."));
  });

  // Manejar el POST para limpiar la caché de escaneo Wi-Fi
  server.on("/clearCache", HTTP_POST, []() {
    WiFi.scanDelete();
    WiFi.disconnect(true);
    server.send(200, "text/plain", "Cache borrada");
    Serial.println("Cache de escaneo y conexión borrada");
  });
  server.begin();
  Serial.println(F("Servidor web iniciado"));
}

// Función para enviar los mensajes al Mega
void enviarMensajeMega(const String &mensaje) {
  MegaSerial.println(mensaje);
  Serial.println("Enviado al Mega: " + mensaje);
}

void setup() {
  // Inicializar Serial1 para comunicar con Mega
  Serial.begin(115200);
  MegaSerial.begin(9600, SERIAL_8N1, MEGA_RX_PIN, MEGA_TX_PIN);
  Serial.println("Inicializando comunicación con el Mega...");

  // Iniciar comunicación I2C para el RTC
  Wire.begin(21, 22);  // SDA en GPIO21 y SCL en GPIO22

  // Verificar si el RTC está conectado correctamente
  if (!rtc.begin()) {
```

```
    Serial.println("No se encontró el RTC DS3231");
    while (1)
      ;
  }

  // Comprobar si el RTC tiene la hora inicializada
  if (rtc.lostPower()) {
    Serial.println("RTC perdió la alimentación. Estableciendo la
hora...");
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }

  Serial.println("RTC DS3231 conectado con éxito.");

  // Leer el delay almacenado en memoria no volátil
  preferences.begin("settings", true);
  retrasoMedicion = preferences.getUInt("retrasoMedicion", 1);  // Valor
por defecto: 1 s
  preferences.end();
  Serial.println("Tiempo de censado inicial: " + String(retrasoMedicion) +
" s");

  // Inicializar las credenciales Wi-Fi guardadas en memoria no volátil
  preferences.begin("wifi-creds", true);
  String ssid = preferences.getString("ssid", "");
  String password = preferences.getString("password", "");
  preferences.end();

  if (ssid.length() > 0 && password.length() > 0) {
    Serial.print("Intentando conectar automáticamente a: ");
    Serial.println(ssid);

    WiFi.begin(ssid.c_str(), password.c_str());

    int attempts = 0;
    while (WiFi.status() != WL_CONNECTED && attempts < 30) {
      delay(1000);
      Serial.print(".");
      attempts++;
    }

    if (WiFi.status() == WL_CONNECTED) {
      Serial.println("\nConexion automatica exitosa.");
      Serial.println("IP: " + WiFi.localIP().toString());
```

```
      // Enviar mensaje al Mega para mostral la conexión automática
exitosa
      enviarMensajeMega("Conexion automatica exitosa a Wi-Fi: " + ssid);
    } else {
      Serial.println("\nNo se pudo conectar automáticamente.");
      // Enviar mensaje al Mega para mostrar el fallo de conexión
automática
      enviarMensajeMega("Error: No se pudo conectar automaticamente a Wi-
Fi");
    }
  } else {
    Serial.println("No se encontraron credenciales Wi-Fi guardadas.");
    // Enviar mensaje al Mega para mostrar la falta de credenciales Wi-Fi
    enviarMensajeMega("Error: No se encontraron credenciales Wi-Fi
guardadas");
  }

  // Inicializar los sensores PZEM
  inicializarPZEMS();

  // Sincronizar el NTP y configurar el tiempo
  syncRTCWithNTP();

  // Inicializar la tarjeta SD
  if (!SD.begin(SD_CS_PIN)) {
    Serial.println("Error: No se pudo inicializar la tarjeta SD");
  } else {
    Serial.println("Tarjeta SD inicializada correctamente");
  }

  // Crear cola para los datos
  dataQueue = xQueueCreate(10, sizeof(DatosSensor));

  // Configurar servidor web
  setupServer();

  // Crear tareas
  xTaskCreatePinnedToCore(tareasSensores, "tareasSensores", 4096, NULL, 4,
NULL, 1);
  xTaskCreatePinnedToCore(tareasInflux, "tareasInflux", 4096, NULL, 3,
NULL, 0);
}
void loop() {
  server.handleClient();  // Procesar las peticiones del servidor web
```

```
  // Mostrar la hora desde RTC y las mediciones de los sensores cuando no
hay conexión Wi-Fi
  if (WiFi.status() != WL_CONNECTED) {
    DateTime now = rtc.now();
    String formattedTime = String(now.year()) + "-" + String(now.month())
+ "-" + String(now.day()) + " " + String(now.hour()) + ":" +
String(now.minute()) + ":" + String(now.second());
    Serial.println("Hora (sin conexión Wi-Fi): " + formattedTime);

    delay(1000);  // Esperar un segundo antes de la siguiente iteración
  }
}
```

Anexo 12: Código Completo Compilado en el Mega 2560

```
#include <MCUFRIEND_kbv.h>

#define MEGA_RX_PIN 19
#define MEGA_TX_PIN 18
#define BUZZER_PIN 44

HardwareSerial &ESP32Serial = Serial1;

MCUFRIEND_kbv tft;

struct DatosSensor {
    int indiceSensor;
    float voltaje;
    float corriente;
    float potenciaAparente;
    float frecuencia;
    float fp;
    float energiaAcumulada;
    float potenciaActiva;
    float potenciaReactiva;
};

DatosSensor lineData[3]; // Información de las tres líneas
bool lineActive[3] = {false, false, false}; // Indica si cada línea tiene
datos recibidos
bool datosRecibidos = false; // Bandera para verificar si se reciben datos

unsigned long messageDisplayTime = 0; // Tiempo en el que se inició la
visualización del mensaje
```

```
const unsigned long messageDuration = 3000; // Duración del mensaje en
milisegundos (3 segundos)
bool displayingMessage = false; // Bandera para verificar si se está
mostrando un mensaje

int errorCount = 0; // Contador de errores de voltaje 0
const int maxErrorCount = 10; // Máximo número de veces que el buzzer
sonará por error

bool buzzerActive = false; // Bandera para controlar si el buzzer está
activo

// Prototipos
void parseSensorData(String data);
void displaySensorData(const DatosSensor *data, int displayIndex);
void displayErrorMessage();
void printSensorDataToSerial(const DatosSensor *data);
bool allVoltagesZero();
void resetSensorData(DatosSensor &data);
void displayConnectionMessage(const String &mensaje);
void soundBuzzer();
void stopBuzzer();

void setup() {
    Serial.begin(115200);
    ESP32Serial.begin(9600);
    pinMode(BUZZER_PIN, OUTPUT); // Configurar el pin del buzzer como
salida

    uint16_t identifier = tft.readID();
    if (identifier == 0x0 || identifier == 0xFFFF) {
        identifier = 0x9341;
    }
    tft.begin(identifier);
    tft.setRotation(3);
    tft.fillScreen(TFT_BLACK);
    tft.setTextColor(TFT_WHITE);

    Serial.println("Mega listo para comunicarse con ESP32");
}

void loop() {
    // Verificar si hay datos disponibles desde el ESP32
if (ESP32Serial.available() > 0) {
```

```
    String data = ESP32Serial.readStringUntil('\n'); // Leer hasta nueva
línea
    Serial.print("Datos recibidos: ");
    Serial.println(data); // Mostrar los datos crudos para depuración

    if (data.startsWith("Conexión exitosa a Wi-Fi:")) {
        Serial.println("ESP32 conectado a Wi-Fi exitosamente.");
        displayConnectionMessage("Wi-Fi conectado: " +
data.substring(22)); // Mostrar el nombre de la red Wi-Fi
    } else if (data.startsWith("Desconectado de Wi-Fi")) {
        Serial.println("ESP32 desconectado de Wi-Fi.");
        displayConnectionMessage("Desconectado de Wi-Fi"); // Mostrar
mensaje de desconexión
    } else if (data.startsWith("Nuevo tiempo de censado:")) {
        Serial.println(data); // Mostrar el nuevo tiempo de censado
        displayConnectionMessage("Tiempo de censado actualizado: " +
data.substring(22)); // Mostrar el nuevo tiempo de censado
    } else if (data.startsWith("Reconexion exitosa a Wi-Fi:")) {
        Serial.println("ESP32 reconectado a Wi-Fi exitosamente.");
        displayConnectionMessage("Reconexion exitosa a Wi-Fi: " +
data.substring(26)); // Mostrar el nombre de la red Wi-Fi
    } else if (data.startsWith("Error: No se pudo reconectar a la red Wi-
Fi")) {
        Serial.println("Error: No se pudo reconectar a la red Wi-Fi.");
        displayConnectionMessage("Error: No se pudo reconectar a la red
Wi-Fi"); // Mostrar mensaje de error
    } else if (data.startsWith("Conexión automatica exitosa a Wi-Fi:")) {
        Serial.println("Conexión automática exitosa a Wi-Fi.");
        displayConnectionMessage("Conexión automatica exitosa a Wi-Fi: " +
data.substring(32)); // Mostrar el nombre de la red Wi-Fi
    } else if (data.startsWith("Error: No se pudo conectar automaticamente
a Wi-Fi")) {
        Serial.println("Error: No se pudo conectar automaticamente a Wi-
Fi.");
        displayConnectionMessage("Error: No se pudo conectar
automaticamente a Wi-Fi"); // Mostrar mensaje de error
    } else if (data.startsWith("Error: No se encontraron credenciales Wi-
Fi guardadas")) {
        Serial.println("Error: No se encontraron credenciales Wi-Fi
guardadas.");
        displayConnectionMessage("Error: No se encontraron credenciales
Wi-Fi guardadas"); // Mostrar mensaje de error
    } else if (data.startsWith("Cambio de red Wi-Fi ")) {
        Serial.println("Cambio de red Wi-Fi.");
```

```
        displayConnectionMessage("Cambio de red Wi-Fi " +
data.substring(20)); // Mostrar el nombre de la nueva red Wi-Fi
    } else if (data.startsWith("Intervalo de tiempo cambiado ")) {
        Serial.println("Intervalo de tiempo cambiado.");
        displayConnectionMessage("Intervalo de tiempo cambiado " +
data.substring(28)); // Mostrar el nuevo intervalo de tiempo
    } else if (data.startsWith("Intervalo de grabado cambiado ")) {
        Serial.println("Intervalo de grabado cambiado.");
        displayConnectionMessage("Intervalo de grabado cambiado " +
data.substring(30)); // Mostrar el nuevo intervalo de grabado de datos
    } else {
        datosRecibidos = true;
        parseSensorData(data); // Parsear y procesar los datos
        // Restablecer las líneas inactivas
        for (int i = 0; i < 3; i++) {
            if (!lineActive[i]) {
                resetSensorData(lineData[i]);
            }
        }

        // Mostrar los datos en la pantalla, cada línea tiene una posición
fija
        if (!displayingMessage) {
            for (int i = 0; i < 3; i++) {
                if (lineActive[i]) {
                    displaySensorData(&lineData[i], i); // Mostrar datos
en posición fija
                } else {
                    resetSensorData(lineData[i]); // Restablecer los datos
                    displaySensorData(&lineData[i], i); // Borrar de
pantalla
                }
            }

            // Mostrar los datos en pantalla y en el Serial Monitor
            for (int i = 0; i < 3; i++) {
                if (lineActive[i]) {
                    printSensorDataToSerial(&lineData[i]);
                }
            }
        }
    }
}
    // Verificar si todos los voltajes son 0
    if (datosRecibidos) {
```

```
        if (allVoltagesZero()) {
            if (!buzzerActive) {
                displayErrorMessage(); // Mostrar mensaje de error si
todas las líneas tienen voltaje 0
                soundBuzzer(); // Activar el buzzer
                buzzerActive = true;
                errorCount = 0; // Reiniciar el contador de errores
            }
        } else {
            if (buzzerActive) {
                stopBuzzer(); // Desactivar el buzzer
                buzzerActive = false;
            }
        }
    }

    // Verificar si el tiempo de visualización del mensaje ha expirado
    if (displayingMessage && (millis() - messageDisplayTime >=
messageDuration)) {
        displayingMessage = false; // Marcar que ya no se está mostrando
el mensaje
        tft.fillScreen(TFT_BLACK); // Limpiar la pantalla
    }
}

// Función para restablecer los datos de una línea
void resetSensorData(DatosSensor &data) {
    data.voltaje = 0.0;
    data.corriente = 0.0;
    data.potenciaAparente = 0.0;
    data.frecuencia = 0.0;
    data.fp = 0.0;
    data.energiaAcumulada = 0.0;
    data.potenciaActiva = 0.0;
    data.potenciaReactiva = 0.0;
}

// Función para mostrar los datos en el Serial Monitor
void printSensorDataToSerial(const DatosSensor *data) {
    Serial.println("=======================================");
    Serial.print("Línea: L");
    Serial.println(data->indiceSensor + 1);
    Serial.print("Voltaje: ");
    Serial.print(data->voltaje, 2);
    Serial.println(" V");
```

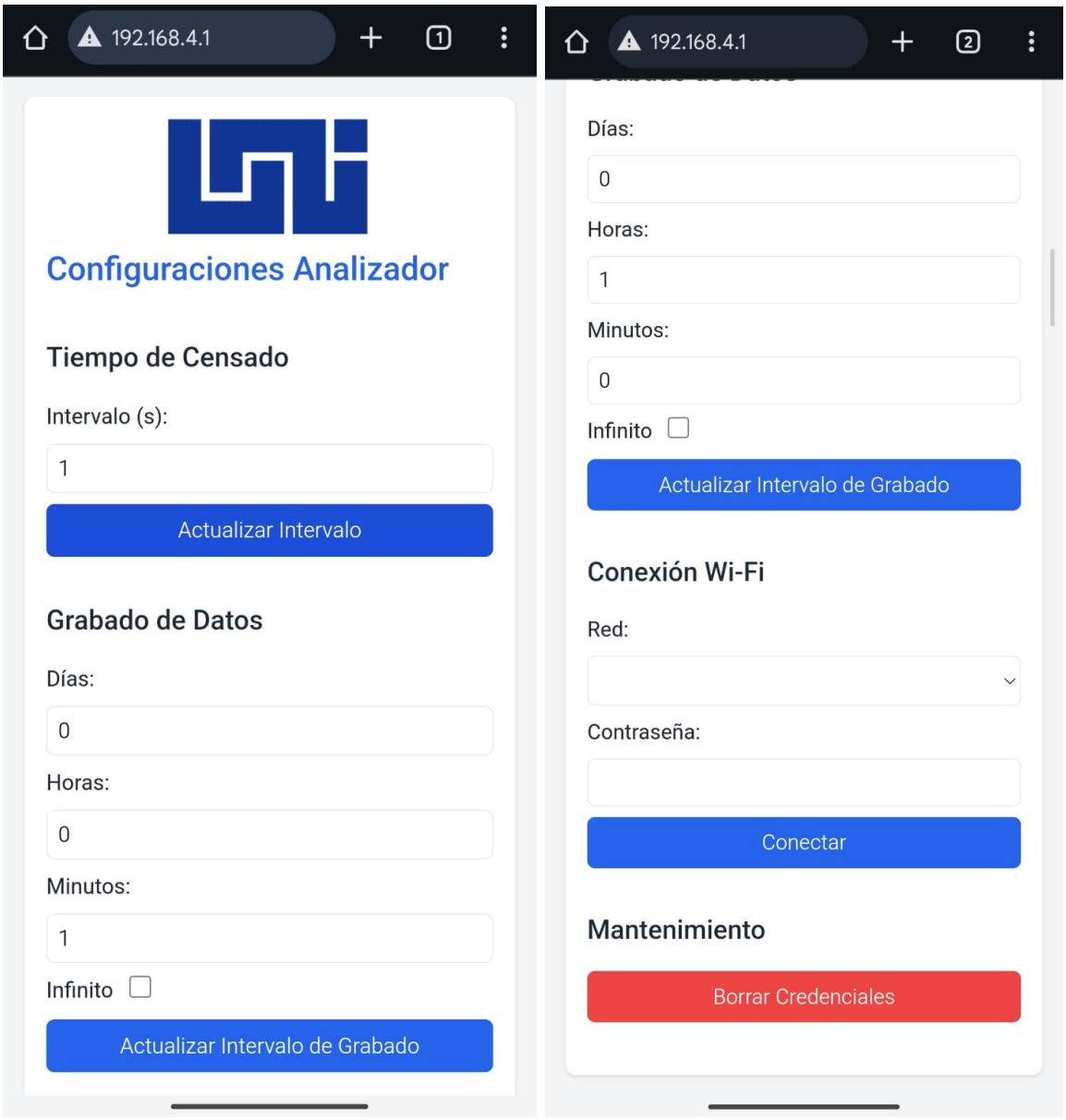
```
    Serial.print("Corriente: ");
    Serial.print(data->corriente, 2);
    Serial.println(" A");
    Serial.print("Potencia_Aparente: ");
    Serial.print(data->potenciaAparente, 2);
    Serial.println(" VA");
    Serial.print("Frecuencia: ");
    Serial.print(data->frecuencia, 2);
    Serial.println(" Hz");
    Serial.print("Factor de Potencia: ");
    Serial.println(data->fp, 2);
    Serial.print("Energía Acumulada: ");
    Serial.print(data->energiaAcumulada, 2);
    Serial.println(" Wh");
    Serial.print("Potencia Activa: ");
    Serial.print(data->potenciaActiva, 2);
    Serial.println(" W");
    Serial.print("Potencia Reactiva: ");
    Serial.print(data->potenciaReactiva, 2);
    Serial.println(" VAR");
    Serial.println("=======================================");
}

// Función para parsear los datos del ESP32
void parseSensorData(String data) {
    // Reinicializar el estado de las líneas
    for (int i = 0; i < 3; i++) {
        lineActive[i] = false; // Suponemos que no se reciben datos para
la línea
    }

    while (data.length() > 0) {
        int separatorIndex = data.indexOf(';');
        String lineDataString = (separatorIndex != -1) ? data.substring(0,
separatorIndex) : data;
        data = (separatorIndex != -1) ? data.substring(separatorIndex + 1)
: "";

        String values[10];
        int valueIndex = 0;
        while (lineDataString.length() > 0 && valueIndex < 10) {
            int commaIndex = lineDataString.indexOf(',');
            values[valueIndex] = (commaIndex != -1) ?
lineDataString.substring(0, commaIndex) : lineDataString;
```

```
            lineDataString = (commaIndex != -1) ?
lineDataString.substring(commaIndex + 1) : "";
            valueIndex++;
        }

        // Convertir los datos de texto a los valores numéricos
        int indiceSensor = values[0].toInt();
        Serial.print("Procesando datos para L");
        Serial.println(indiceSensor + 1);

        if (indiceSensor >= 0 && indiceSensor < 3) { // Validar que el
indiceSensor esté en el rango 0-2
            if (!lineActive[indiceSensor]) { // Solo procesar si no se ha
recibido datos previamente para este índice
                lineActive[indiceSensor] = true;
                lineData[indiceSensor].indiceSensor = indiceSensor;
                lineData[indiceSensor].voltaje = values[1].toFloat();
                lineData[indiceSensor].corriente = values[2].toFloat();
                lineData[indiceSensor].potenciaAparente =
values[3].toFloat();
                lineData[indiceSensor].frecuencia = values[4].toFloat();
                lineData[indiceSensor].fp = values[5].toFloat();
                lineData[indiceSensor].energiaAcumulada =
values[6].toFloat();
                lineData[indiceSensor].potenciaActiva =
values[7].toFloat();
                lineData[indiceSensor].potenciaReactiva =
values[8].toFloat();
            } else {
                // Ignorar el indiceSensor duplicado directamente sin
advertencias
                continue; // Saltar a la siguiente iteración
            }
        } else {
            Serial.println("Error: indiceSensor fuera de rango");
        }
    }
}

// Función para verificar si todos los voltajes son 0
bool allVoltagesZero() {
    for (int i = 0; i < 3; i++) {
        if (lineActive[i] && lineData[i].voltaje > 0.0) {
            return false; // Si alguno de los voltajes no es 0, retornar
falso
```

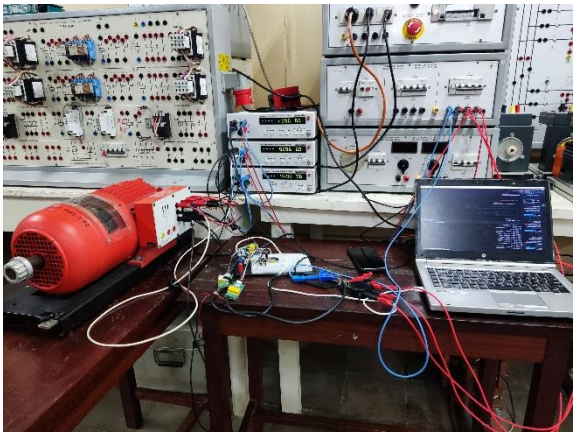
```
        }
    }
    return true; // Todos los voltajes son 0
}

// Función para mostrar los datos en la pantalla
void displaySensorData(const DatosSensor *data, int displayIndex) {
    int yOffset = displayIndex * 80;

    tft.fillRect(0, 10 + yOffset, 320, 80, TFT_BLACK);
    String lineLabel = "L" + String(displayIndex + 1);

    tft.setCursor(10, 10 + yOffset);
    tft.print(lineLabel);

    tft.setCursor(10, 30 + yOffset);
    tft.print("Voltaje: ");
    tft.print(data->voltaje, 2);
    tft.print(" V");

    tft.setCursor(10, 50 + yOffset);
    tft.print("Corriente: ");
    tft.print(data->corriente, 2);
    tft.print(" A");

    tft.setCursor(10, 70 + yOffset);
    tft.print("Potencia: ");
    tft.print(data->potenciaActiva, 2);
    tft.print(" W");

    tft.setCursor(150, 30 + yOffset);
    tft.print("Freq: ");
    tft.print(data->frecuencia, 2);
    tft.print(" Hz");

    tft.setCursor(150, 50 + yOffset);
    tft.print("FP: ");
    tft.print(data->fp, 2);

    tft.setCursor(150, 70 + yOffset);
    tft.print("Energia Acum.: ");
    tft.print(data->energiaAcumulada, 2);
    tft.print(" Wh");
}
```

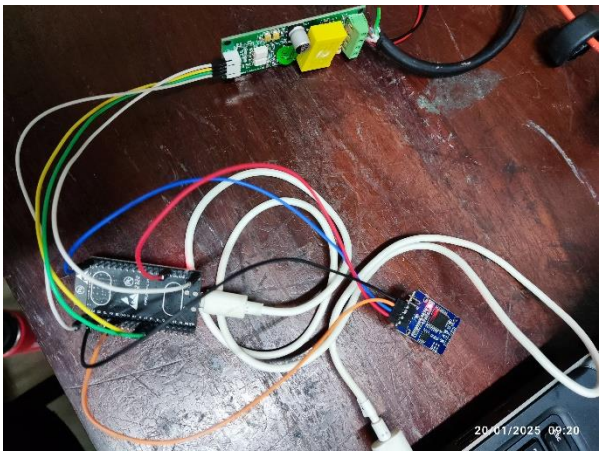
```
// Función para mostrar un mensaje de error
void displayErrorMessage() {
    // Agregar un pequeño delay antes de mostrar el mensaje de error
    tft.fillScreen(TFT_BLACK);
    tft.setCursor(10, 100);
    tft.setTextColor(TFT_WHITE);
    tft.setTextSize(1);
    tft.print("Error: No se reciben datos");
}

// Función para mostrar un mensaje de conexión
void displayConnectionMessage(const String &mensaje) {
    tft.fillScreen(TFT_BLACK);
    tft.setCursor(50, 100);
    tft.setTextColor(TFT_WHITE);
    tft.setTextSize(1);
    tft.println(mensaje);

    messageDisplayTime = millis(); // Registrar el tiempo de inicio de la
visualización del mensaje
    displayingMessage = true; // Marcar que se está mostrando un mensaje
}

void soundBuzzer() {
    for (int i = 0; i < 5; i++) { // Ahora solo sonará 5 veces en cada
evento
        tone(BUZZER_PIN, 2000); // Suena con 500 Hz (tono más suave)
        delay(150); // Se mantiene el sonido por 150ms
        noTone(BUZZER_PIN); // Apagar el buzzer
        delay(250); // Pausa más larga para que sea menos molesto
    }
}

void stopBuzzer() {
    noTone(BUZZER_PIN); // Apagar el buzzer
}
```

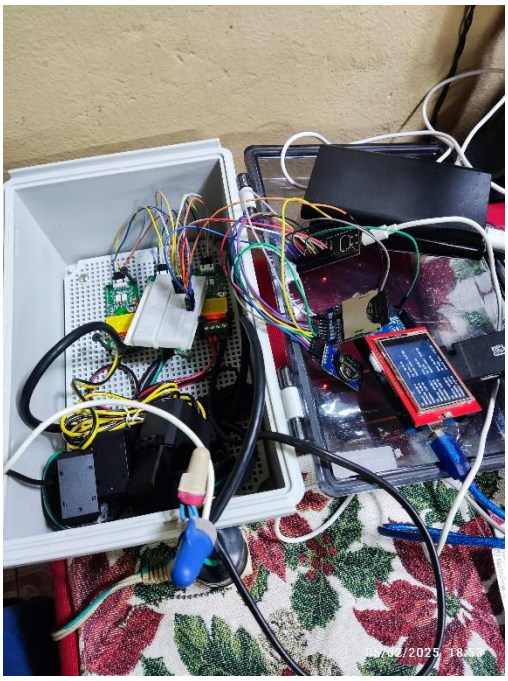
Anexo 13: Interfaz Gráfica en el Servidor Web

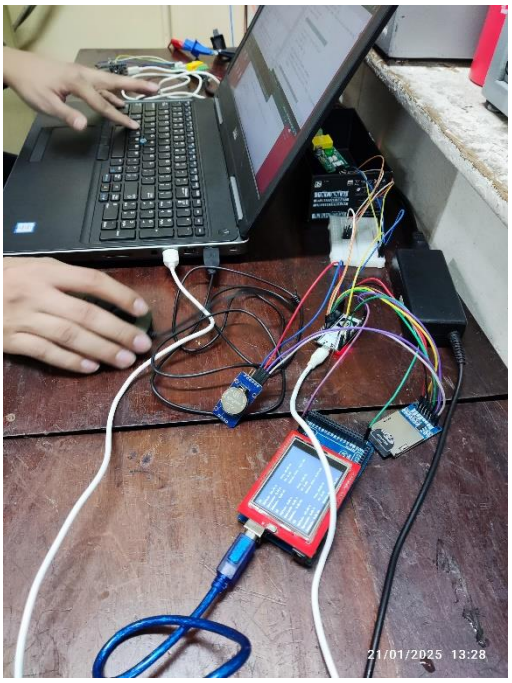


Anexo 14: Evidencia de construcción del dispositivo.




20/01/2025 09:20




21/01/2025 13:28

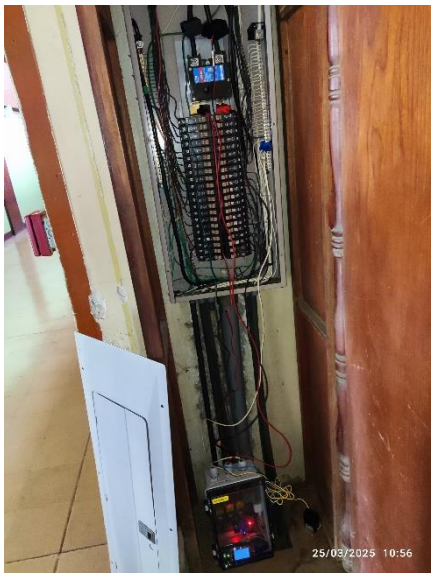

17/02/2025 10:03

Anexo 15: tabla comparativa entre algunos analizadores de energía.

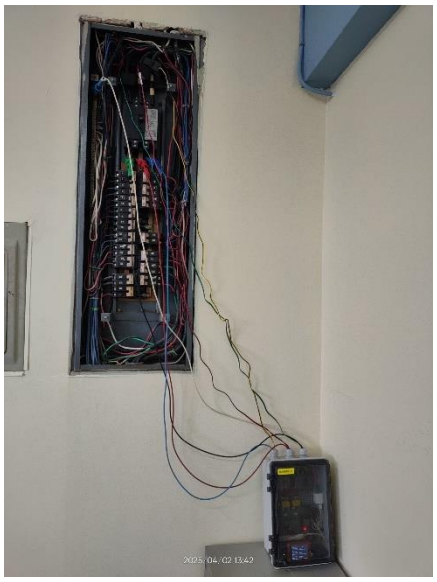
(Schneider electric, s.f.), (Fluke , s.f.), (Grupo Novelec, s.f.), (CIRCUTOR, s.f.)

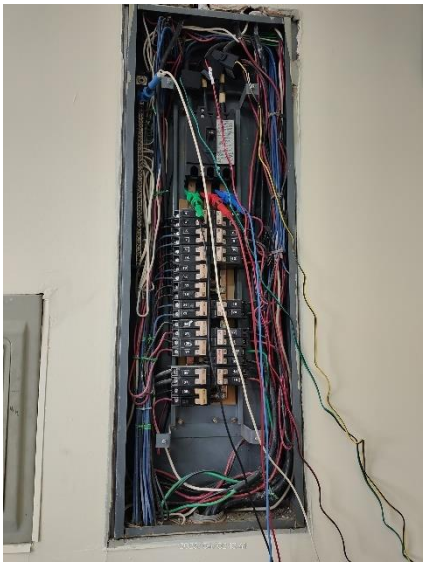
Componentes	Características	Exactitudes	Precio
PROTOTIPO ANALIZADOR DE REDES	Voltaje:80~260VLL Corriente: 300A Frecuencia:45~65Hz factor de potencia: 0.00~1.00	Energía activa +/- 0.5 % Potencia activa +/- 0.5 % Potencia aparente +/- 0.5 % Frecuencia +/- 0.5 % Factor de potencia +/- 1% Corriente +/- 0.5 % Tensión +/- 0.5 %	217$
MEDIDOR POWERLOGIC PM5100 C0,5 HASTA 15TH S/COMUNICACIÓN 1SD (sin sonda ni toroidal) tipo fijo	Voltaje:35~690VLL Corriente: 1 0 5A Frecuencia:47 a 63Hz factor de potencia: 0.00~1.00 armónico:15vo	Energía activa +/- 0.5 % Potencia activa +/- 0.5 % Potencia aparente +/- 0.5 % Frecuencia +/- 0.05 % Factor de potencia +/- 0.5 Corriente +/- 0.5 % Tensión +/- 0.5 % Energía aparente +/- 0.5 % Potencia reactiva +/- 2 %	638.97$
Registrador de energía trifásico Fluke 1732 con Iflex 1500-12 (CALIBRADO 2023-2024)-	Voltaje:1000v Corriente: 1500A Frecuencia:42.5~69Hz factor de potencia: 0.00~1.00	Energía activa +/- 1,2% + 0,005 potencia activa +/- 01,2% + 0,005 Potencia aparente 01,2% + 0,005 Frecuencia +/- 0.1 % Factor de potencia +/- 0.5 Corriente +/- (1% + 0,02%) Tensión +/- (0,2% + 0,01%) Energía aparente +/- (1.2% + 0,002%) Energía reactiva +/- 2,5 % Potencia reactiva +/- 2,5 %	4 147.00$
ANALIZADOR RED PQA820 TRIFÁSICO IP65	Voltaje:460 vl Corriente: 1000A Frecuencia:45~65Hz factor de potencia: 0.00~1.00 THDI Y THDV	Energía activa +/- 1,% + 0,005 potencia activa +/- 01,2% + 0,005 Potencia aparente 01,2% + 0,005 Frecuencia +/- 0.1 % Corriente +/- (0,5 % + 0,2 %) Tensión +/- (0,5 % de la lectura + 0,2 %) Energía aparente +/- (1) Energía reactiva +/- 2 % Potencia reactiva +/- 2 %	2295.71$
Circutor AR6 E-FLEX	Voltaje:1000V Corriente: 10000A Frecuencia:45~65Hz factor de potencia: 0.00~1.01	Energía activa +/- 1,% potencia activa +/- 1 Potencia aparente +/- 1 Frecuencia +/- 0.01 % Corriente +/- 0.5% Tensión +/- 0,5%) Energía aparente +/- (1) Energía reactiva +/- 1 % Potencia reactiva +/- 1%	4352.34$

Anexo 16: Instalación del prototipo analizador de redes en tablero ubicado en la Residencia Estudiantil UNI-RUSB.



Anexo 17: Instalación de prototipo de analizador de redes en tablero ubicado en el tercer piso del Edificio de Ing. de Sistemas UNI-RUSB.



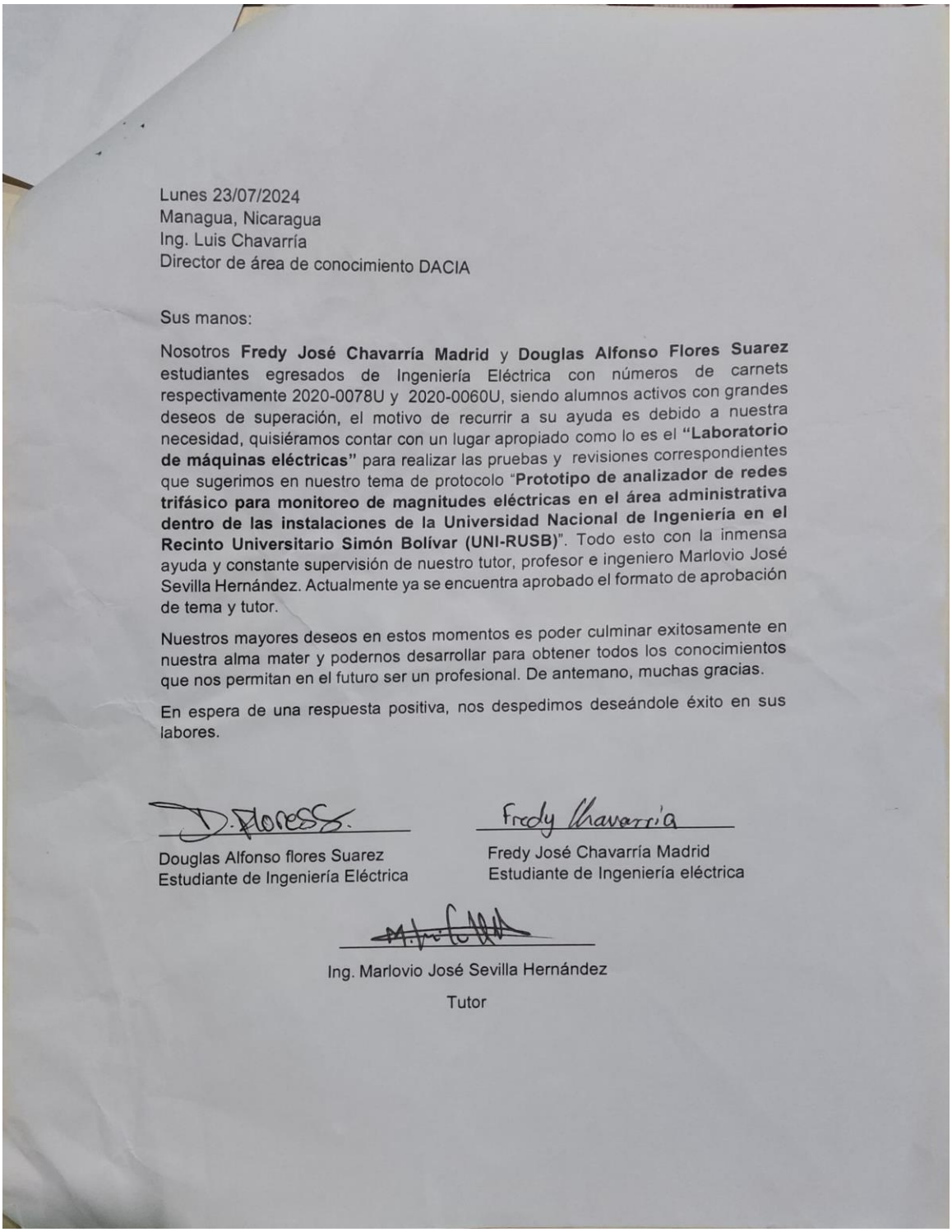


Anexo 18: Entrevista Semiestructurada.

- **Objetivo:** Obtener información sobre la problemática de la falta de medición seccionada en el área administrativa y recopilar opiniones y sugerencias del personal técnico y administrativo de la UNI-RUSB.
- **Datos generales:** profesión del entrevistado y cargo del entrevistado.
- **Preguntas:**

1. ¿Cuáles cree que son las principales dificultades o desafíos asociados con la falta de medición seccionada?
2. ¿Qué impacto ha tenido esta problemática en las operaciones diarias del área administrativa?
3. ¿Cómo afecta la falta de medición seccionada en el área administrativa a la planificación y gestión del consumo energético?
4. ¿Cuáles serían los beneficios potenciales de implementar un sistema de medición seccionada de energía eléctrica?
5. ¿Qué características o funcionalidades considera que serían importantes en un prototipo de analizador de redes trifásico para abordar esta problemática?
6. ¿Cree que la implementación de un prototipo de analizador de redes trifásico podría generar ahorros significativos en el consumo energético?

Anexo 19: Carta de solicitud del laboratorio para el segundo semestre del 2024. (Fuente Propia)



Lunes 23/07/2024
Managua, Nicaragua
Ing. Luis Chavarría
Director de área de conocimiento DACIA

Sus manos:

Nosotros **Fredy José Chavarría Madrid** y **Douglas Alfonso Flores Suarez** estudiantes egresados de Ingeniería Eléctrica con números de carnets respectivamente 2020-0078U y 2020-0060U, siendo alumnos activos con grandes deseos de superación, el motivo de recurrir a su ayuda es debido a nuestra necesidad, quisiéramos contar con un lugar apropiado como lo es el **"Laboratorio de máquinas eléctricas"** para realizar las pruebas y revisiones correspondientes que sugerimos en nuestro tema de protocolo "**Prototipo de analizador de redes trifásico para monitoreo de magnitudes eléctricas en el área administrativa dentro de las instalaciones de la Universidad Nacional de Ingeniería en el Recinto Universitario Simón Bolívar (UNI-RUSB)**". Todo esto con la inmensa ayuda y constante supervisión de nuestro tutor, profesor e ingeniero Marlovio José Sevilla Hernández. Actualmente ya se encuentra aprobado el formato de aprobación de tema y tutor.

Nuestros mayores deseos en estos momentos es poder culminar exitosamente en nuestra alma mater y podernos desarrollar para obtener todos los conocimientos que nos permitan en el futuro ser un profesional. De antemano, muchas gracias.

En espera de una respuesta positiva, nos despedimos deseándole éxito en sus labores.

D. Flores S.

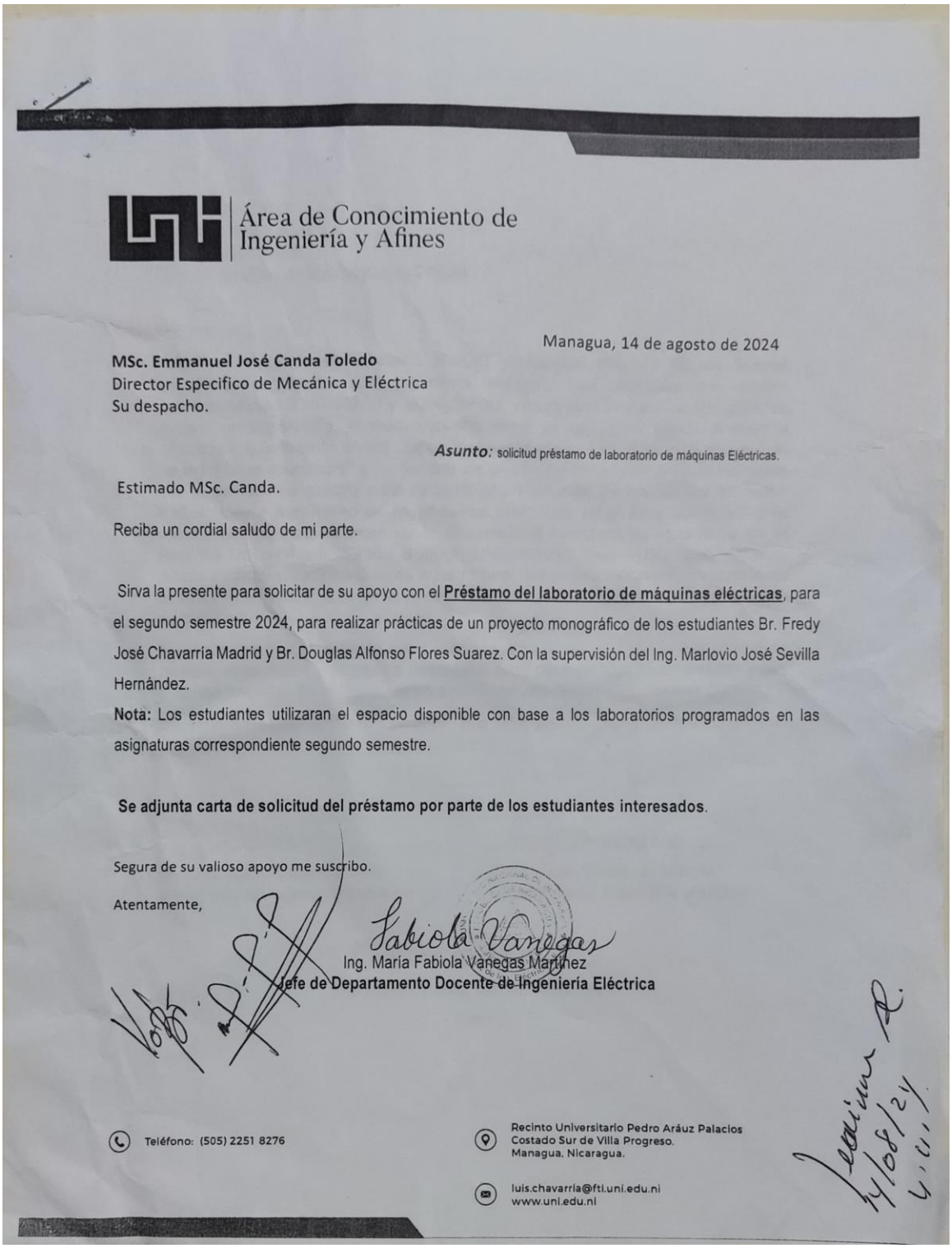
Douglas Alfonso flores Suarez
Estudiante de Ingeniería Eléctrica

Fredy Chavarría

Fredy José Chavarría Madrid
Estudiante de Ingeniería eléctrica

Ing. Marlovio José Sevilla Hernández
Tutor

Anexo 20: Carta de solicitud de apoyo y firma indicando la autorización por parte de la autoridad correspondiente (Fuente Propia)



Área de Conocimiento de Ingeniería y Afines

Managua, 14 de agosto de 2024

MSc. Emmanuel José Canda Toledo
Director Especifico de Mecánica y Eléctrica
Su despacho.

Asunto: solicitud préstamo de laboratorio de máquinas Eléctricas.

Estimado MSc. Canda.

Reciba un cordial saludo de mi parte.

Sirva la presente para solicitar de su apoyo con el **Préstamo del laboratorio de máquinas eléctricas**, para el segundo semestre 2024, para realizar prácticas de un proyecto monográfico de los estudiantes Br. Fredy José Chavarría Madrid y Br. Douglas Alfonso Flores Suarez. Con la supervisión del Ing. Marlovio José Sevilla Hernández.

Nota: Los estudiantes utilizaran el espacio disponible con base a los laboratorios programados en las asignaturas correspondiente segundo semestre.

Se adjunta carta de solicitud del préstamo por parte de los estudiantes interesados.

Segura de su valioso apoyo me suscribo.

Atentamente,

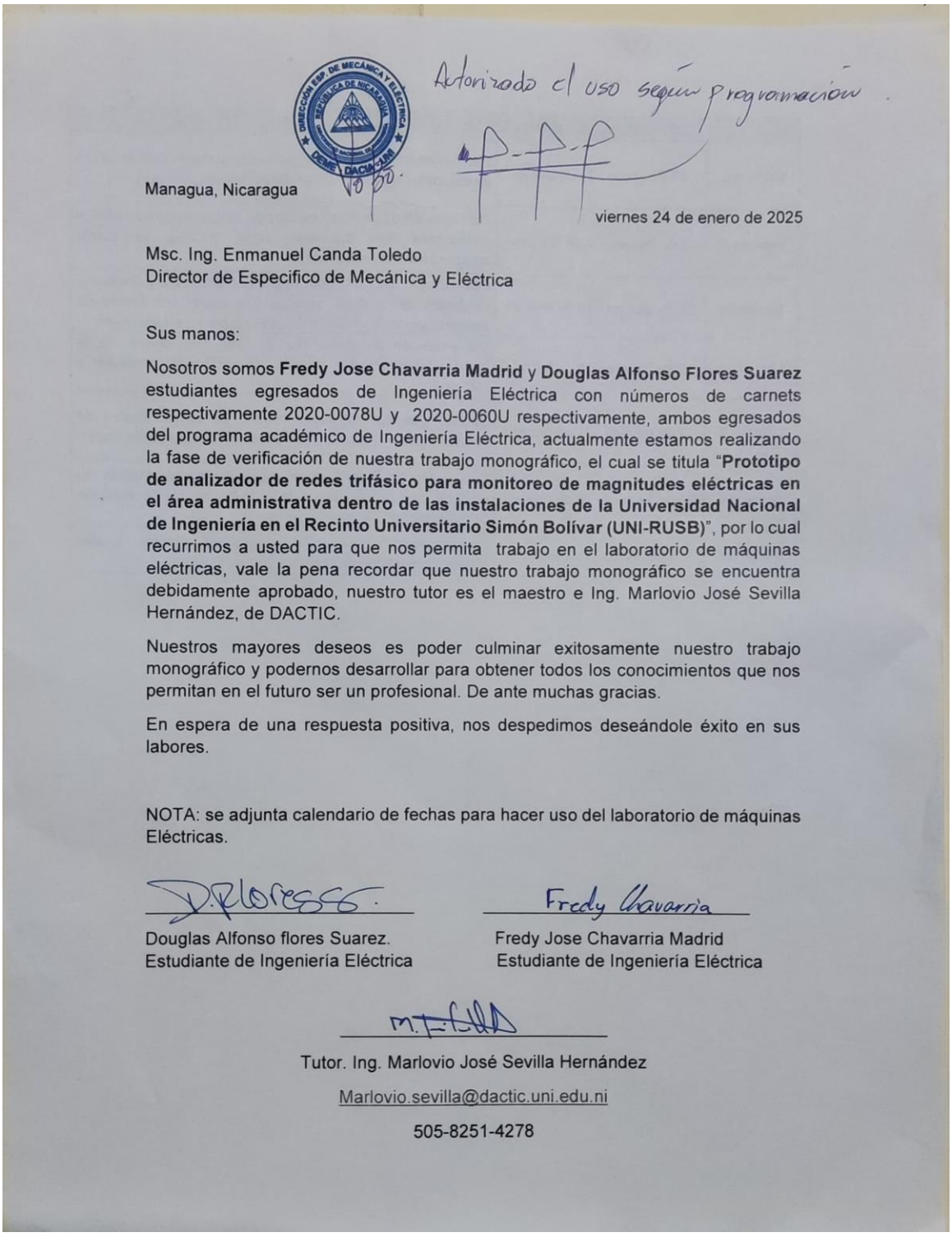
Ing. María Fabiola Vanegas Martínez
Jefe de Departamento Docente de Ingeniería Eléctrica

Teléfono: (505) 2251 8276

Recinto Universitario Pedro Aráuz Palacios
Costado Sur de Villa Progreso.
Managua, Nicaragua.

luis.chavarria@fti.uni.edu.ni
www.uni.edu.ni

Anexo 21: Carta de solicitud para el primer semestre del 2025 y firma indicando autorización. (Fuente Propia)



Autorizado el uso según programación.

Managua, Nicaragua

viernes 24 de enero de 2025

Msc. Ing. Enmanuel Canda Toledo
Director de Especifico de Mecánica y Eléctrica

Sus manos:

Nosotros somos **Fredy Jose Chavarria Madrid** y **Douglas Alfonso Flores Suarez** estudiantes egresados de Ingeniería Eléctrica con números de carnets respectivamente 2020-0078U y 2020-0060U respectivamente, ambos egresados del programa académico de Ingeniería Eléctrica, actualmente estamos realizando la fase de verificación de nuestra trabajo monográfico, el cual se titula "**Prototipo de analizador de redes trifásico para monitoreo de magnitudes eléctricas en el área administrativa dentro de las instalaciones de la Universidad Nacional de Ingeniería en el Recinto Universitario Simón Bolívar (UNI-RUSB)**", por lo cual recurrimos a usted para que nos permita trabajo en el laboratorio de máquinas eléctricas, vale la pena recordar que nuestro trabajo monográfico se encuentra debidamente aprobado, nuestro tutor es el maestro e Ing. Marlovio José Sevilla Hernández, de DACTIC.

Nuestros mayores deseos es poder culminar exitosamente nuestro trabajo monográfico y podernos desarrollar para obtener todos los conocimientos que nos permitan en el futuro ser un profesional. De ante muchas gracias.

En espera de una respuesta positiva, nos despedimos deseándole éxito en sus labores.

NOTA: se adjunta calendario de fechas para hacer uso del laboratorio de máquinas Eléctricas.

Douglas Alfonso flores Suarez.
Estudiante de Ingeniería Eléctrica

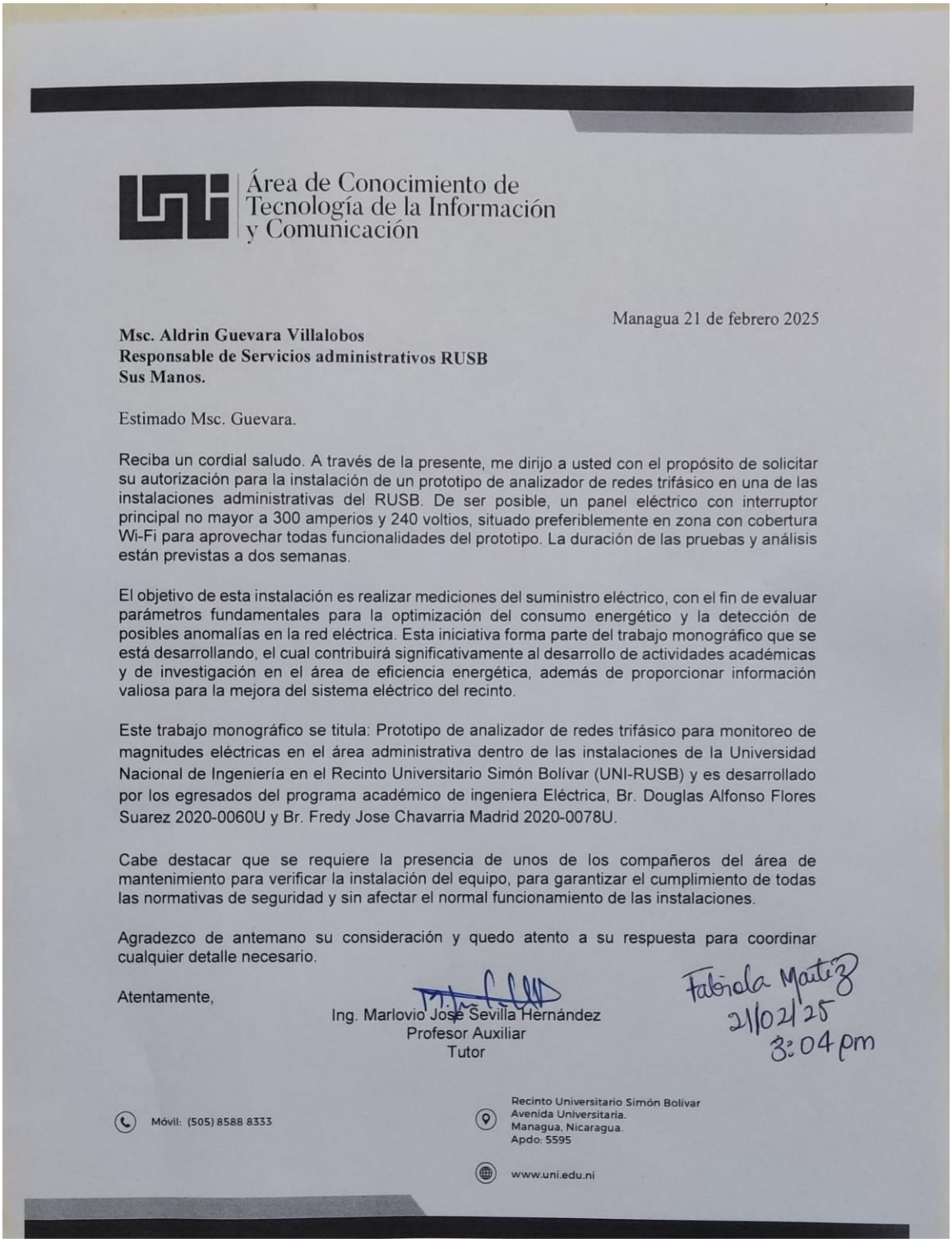
Fredy Jose Chavarria Madrid
Estudiante de Ingeniería Eléctrica

Tutor. Ing. Marlovio José Sevilla Hernández

Marlovio.sevilla@dactic.uni.edu.ni

505-8251-4278

Anexo 22: Carta de solicitud para instalación del prototipo en un tablero. (Fuente Propia)



Área de Conocimiento de
Tecnología de la Información
y Comunicación

Managua 21 de febrero 2025

Msc. Aldrin Guevara Villalobos
Responsable de Servicios administrativos RUSB
Sus Manos.

Estimado Msc. Guevara.

Reciba un cordial saludo. A través de la presente, me dirijo a usted con el propósito de solicitar su autorización para la instalación de un prototipo de analizador de redes trifásico en una de las instalaciones administrativas del RUSB. De ser posible, un panel eléctrico con interruptor principal no mayor a 300 amperios y 240 voltios, situado preferiblemente en zona con cobertura Wi-Fi para aprovechar todas funcionalidades del prototipo. La duración de las pruebas y análisis están previstas a dos semanas.

El objetivo de esta instalación es realizar mediciones del suministro eléctrico, con el fin de evaluar parámetros fundamentales para la optimización del consumo energético y la detección de posibles anomalías en la red eléctrica. Esta iniciativa forma parte del trabajo monográfico que se está desarrollando, el cual contribuirá significativamente al desarrollo de actividades académicas y de investigación en el área de eficiencia energética, además de proporcionar información valiosa para la mejora del sistema eléctrico del recinto.

Este trabajo monográfico se titula: Prototipo de analizador de redes trifásico para monitoreo de magnitudes eléctricas en el área administrativa dentro de las instalaciones de la Universidad Nacional de Ingeniería en el Recinto Universitario Simón Bolívar (UNI-RUSB) y es desarrollado por los egresados del programa académico de ingeniera Eléctrica, Br. Douglas Alfonso Flores Suarez 2020-0060U y Br. Fredy Jose Chavarria Madrid 2020-0078U.

Cabe destacar que se requiere la presencia de unos de los compañeros del área de mantenimiento para verificar la instalación del equipo, para garantizar el cumplimiento de todas las normativas de seguridad y sin afectar el normal funcionamiento de las instalaciones.

Agradezco de antemano su consideración y quedo atento a su respuesta para coordinar cualquier detalle necesario.

Atentamente,

Ing. Marlovio Jose Sevilla Hernández
Profesor Auxiliar
Tutor

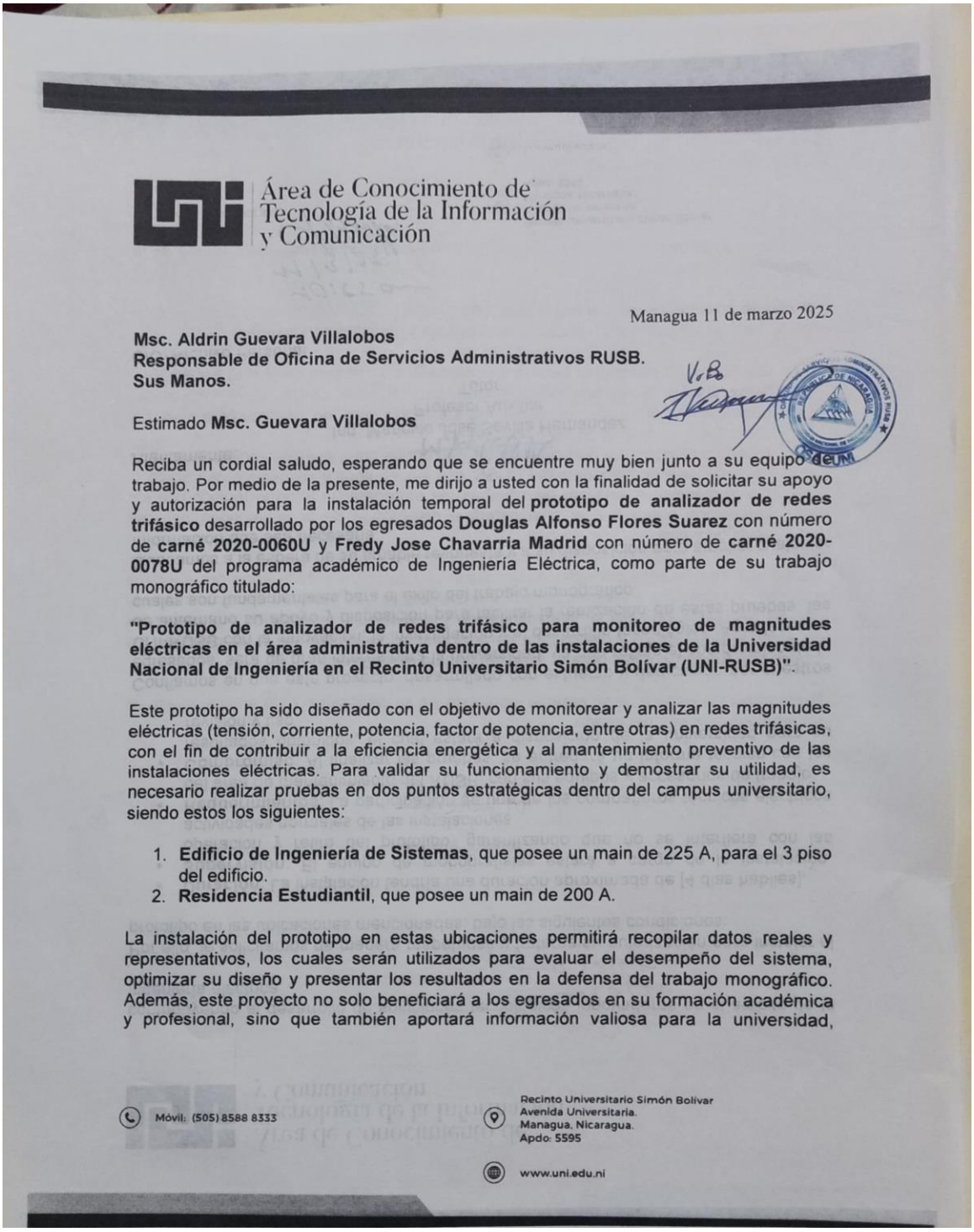
Fabiola Martiz
21/02/25
3:04 pm

Móvil: (505) 8588 8333

Recinto Universitario Simón Bolívar
Avenida Universitaria.
Managua, Nicaragua.
Apdo: 5595

www.uni.edu.ni

Anexo 23: Carta con sello y firma de aprobación de instalación del prototipo. (Fuente Propia)



Área de Conocimiento de Tecnología de la Información y Comunicación

Managua 11 de marzo 2025

Msc. Aldrin Guevara Villalobos
Responsable de Oficina de Servicios Administrativos RUSB.
Sus Manos.

Estimado **Msc. Guevara Villalobos**

Reciba un cordial saludo, esperando que se encuentre muy bien junto a su equipo de trabajo. Por medio de la presente, me dirijo a usted con la finalidad de solicitar su apoyo y autorización para la instalación temporal del **prototipo de analizador de redes trifásico** desarrollado por los egresados **Douglas Alfonso Flores Suarez** con número de **carné 2020-0060U** y **Fredy Jose Chavarria Madrid** con número de **carné 2020-0078U** del programa académico de Ingeniería Eléctrica, como parte de su trabajo monográfico titulado:

"Prototipo de analizador de redes trifásico para monitoreo de magnitudes eléctricas en el área administrativa dentro de las instalaciones de la Universidad Nacional de Ingeniería en el Recinto Universitario Simón Bolívar (UNI-RUSB)".

Este prototipo ha sido diseñado con el objetivo de monitorear y analizar las magnitudes eléctricas (tensión, corriente, potencia, factor de potencia, entre otras) en redes trifásicas, con el fin de contribuir a la eficiencia energética y al mantenimiento preventivo de las instalaciones eléctricas. Para validar su funcionamiento y demostrar su utilidad, es necesario realizar pruebas en dos puntos estratégicas dentro del campus universitario, siendo estos los siguientes:

1. **Edificio de Ingeniería de Sistemas**, que posee un main de 225 A, para el 3 piso del edificio.
2. **Residencia Estudiantil**, que posee un main de 200 A.

La instalación del prototipo en estas ubicaciones permitirá recopilar datos reales y representativos, los cuales serán utilizados para evaluar el desempeño del sistema, optimizar su diseño y presentar los resultados en la defensa del trabajo monográfico. Además, este proyecto no solo beneficiará a los egresados en su formación académica y profesional, sino que también aportará información valiosa para la universidad,

Móvil: (505) 8588 8333

Recinto Universitario Simón Bolívar
Avenida Universitaria.
Managua, Nicaragua.
Apdo: 5595

www.uni.edu.ni

Área de Conocimiento de
Tecnología de la Información
y Comunicación

contribuyendo al desarrollo de soluciones tecnológicas innovadoras en el ámbito de la ingeniería eléctrica.

Por ello, le solicitamos de manera respetuosa y fraterna su autorización para instalar el prototipo en las ubicaciones mencionadas, bajo las siguientes condiciones:

- **Duración**: La instalación tendría una duración aproximada de [4 dias habiles].
- **Supervisión**: El equipo de monografistas estará a cargo de la instalación, operación y retiro del prototipo, garantizando que no se interfiera con las actividades normales de las instalaciones.
- **Requerimientos**: La participación de uno de los compañeros técnicos eléctricos del área de mantenimiento del RUSB, con sus equipos y accesorios de trabajo.
- **Compromiso**: Al finalizar las pruebas, se entregará un informe técnico con los resultados obtenidos, el cual estará a disposición de la universidad para su revisión y uso.

Confiamos en que este proyecto, desarrollado con esfuerzo y dedicación por nuestros egresados, será de gran interés para la universidad, ya que refleja el compromiso de la UNI-RUSB con la innovación, la investigación y el desarrollo tecnológico. Agradecemos de antemano su apoyo y disposición para facilitar la realización de estas pruebas, las cuales son fundamentales para el éxito del trabajo monográfico.

Quedamos a la espera de su amable respuesta y a su disposición para ampliar cualquier información que sea necesaria.

Atentamente,

Ing. Marlovio José Sevilla Hernández
Profesor Auxiliar
Tutor

C/C. Archivo Personal

10:55 am
7/3/25

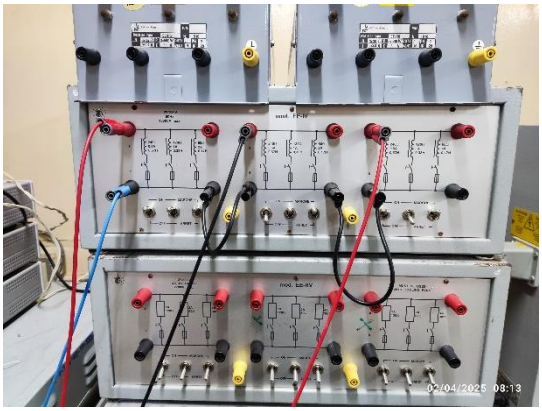
Móvil: (505) 8588 8333

Recinto Universitario Simón Bolívar
Avenida Universitaria.
Managua, Nicaragua.
Apdo: 5595

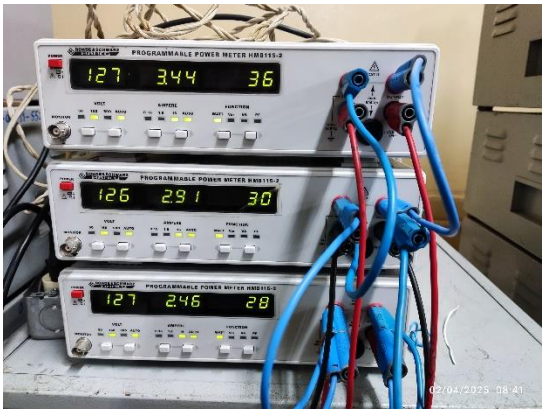
www.uni.edu.ni

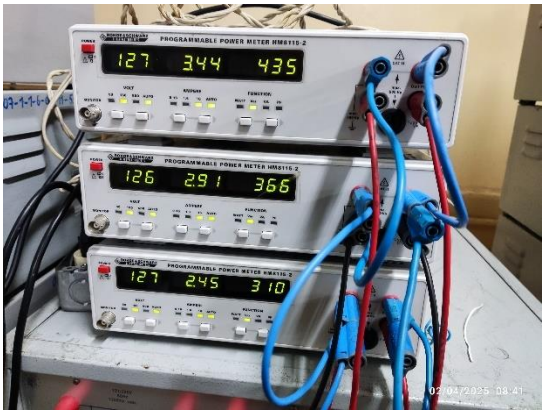
Anexo 24: Evidencia de pruebas de precisión y tablas comparativas.

Anexo 24.1: Prueba con cargas asimétricas de tipo inductivas.



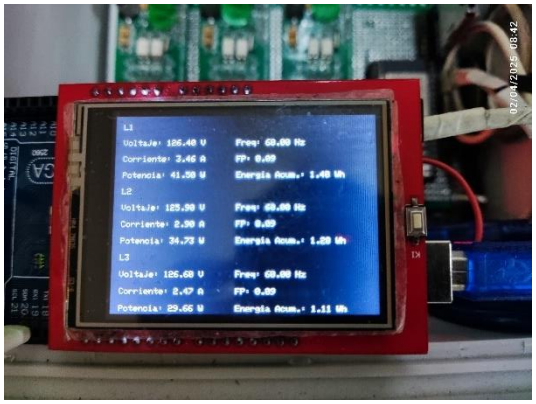
Anexo 24.2: Medición de voltaje (V), corriente (A), potencia activa (W) y potencia reactiva (VAR) en las tres líneas con el método de los tres vatímetros.





Anexo 24.3: Medición de potencia aparente (VA) y factor de potencia (FP).

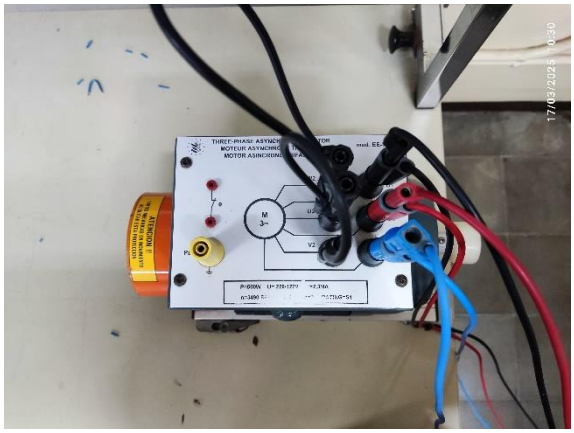
Anexo 24.4: Visualización de datos en pantalla y base de datos Influx DB.



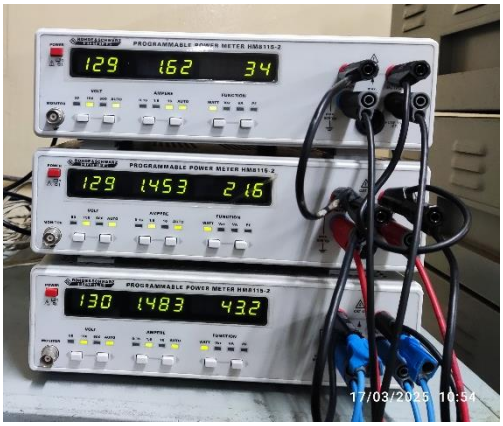
Anexo 24.5: Tabla comparativa de mediciones del dispositivo con respecto a los tres vatímetros donde la carga fueron inductores.

TABLA COMPARATIVA DE RESULTADOS DE MEDICÓN				
ESPECIFICACIONES DE LA CARGA SEGÚN ETIQUETA:		V= 220/127 V		I1= 3.5 A I2= 3 A I3= 2.5 A
	POWER METERS	PROTOTIPO	% ERROR	% ERROR PROMEDIO
V L1	127.00	126.70	0.24%	3.23%
V L2	126.00	125.90	0.08%	
V L3	127.00	126.60	0.31%	
A L1	3.44	3.46	0.58%	
A L2	2.91	2.90	0.34%	
A L3	2.46	2.47	0.40%	
W L1	36.00	41.50	13.25%	
W L2	30.00	34.73	13.62%	
W L3	28.00	29.66	5.60%	
VAR L1	435.00	434.86	0.03%	
VAR L2	366.00	363.96	0.56%	
VAR L3	310.00	310.78	0.25%	
VA L1	436.00	436.84	0.19%	
VA L2	367.00	365.61	0.38%	
VA L3	312.00	312.20	0.06%	
FP L1	0.08	0.09	11.11%	
FP L2	0.08	0.09	11.11%	
FP L3	0.09	0.09	0.00%	
* NOTA: PRUEBAS EN CONDICIONES DE LABORATORIO				

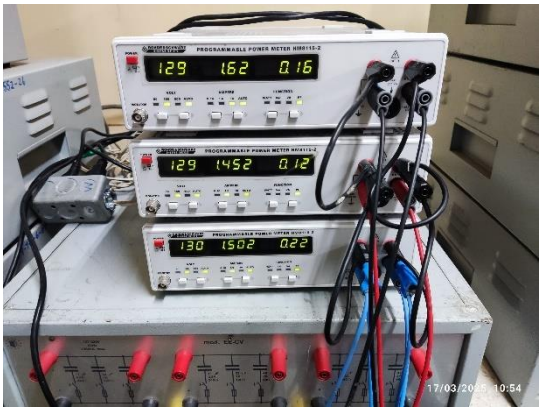
Anexo 24.6: Prueba con motor trifásico en vacío.



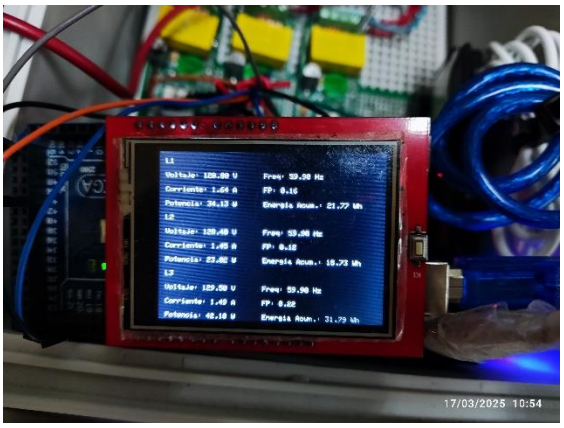
Anexo 24.7: Medición de voltaje (V), corriente (A), potencia activa (W) y potencia reactiva (VAR) en las tres líneas con el método de los tres vatímetros.



Anexo 24.8: Medición de potencia aparente (VA) y factor de potencia (FP).



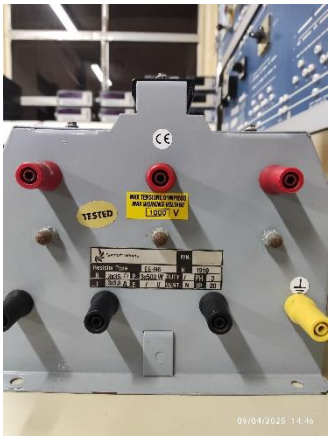
Anexo 24.9: Visualización de datos en pantalla y base de datos Influx DB.



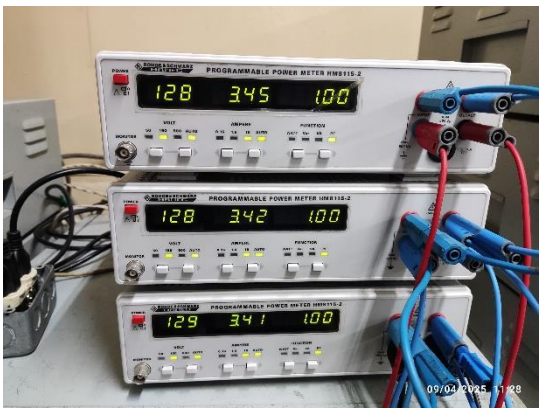
Anexo 24.10: Tabla comparativa de mediciones del prototipo con respecto a los tres vatímetros donde la carga fue un motor trifásico en vacío (conectado en estrella).

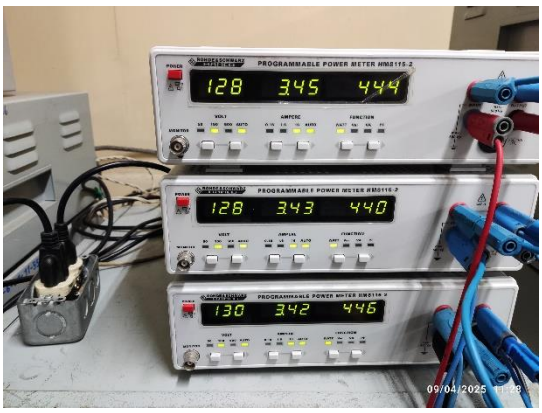
TABLA COMPARATIVA DE RESULTADOS DE MEDICÓN				
ESPECIFICACIONES DE LA CARGA SEGÚN ETIQUETA:		P= 500 W	V= 220/127 V	I= 2.3/4 A
	POWER METERS	PROTOTIPO	% ERROR	% ERROR PROMEDIO
V L1	129.00	128.80	0.16%	1.55%
V L2	129.00	128.40	0.47%	
V L3	130.00	129.50	0.38%	
A L1	1.62	1.64	1.22%	
A L2	1.45	1.45	0.00%	
A L3	1.48	1.49	0.67%	
W L1	34.00	34.13	0.38%	
W L2	21.60	23.02	6.17%	
W L3	43.20	42.10	2.55%	
VAR L1	206.00	204.06	0.94%	
VAR L2	184.00	183.1	0.49%	
VAR L3	189.00	184.86	2.19%	
VA L1	210.00	207.11	1.38%	
VA L2	188.20	184.51	1.96%	
VA L3	195.40	189.44	3.05%	
FP L1	0.16	0.17	5.88%	
FP L2	0.12	0.12	0.00%	
FP L3	0.22	0.22	0.00%	
* NOTA: PRUEBAS EN CONDICIONES DE LABORATORIO				

Anexo 24.11: Prueba con resistencias eléctricas

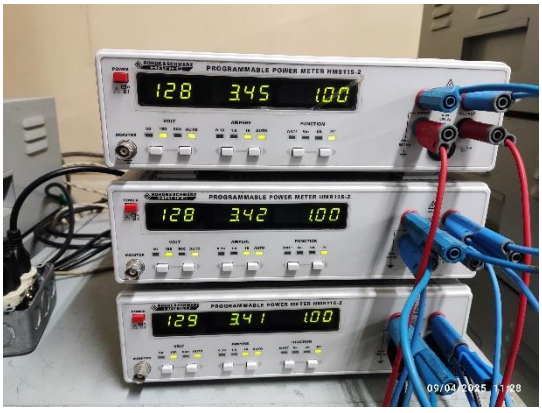


Anexo 24.12: Medición de voltaje (V), corriente (A), potencia activa (W) y potencia en las tres líneas con el método de los tres vatímetros.

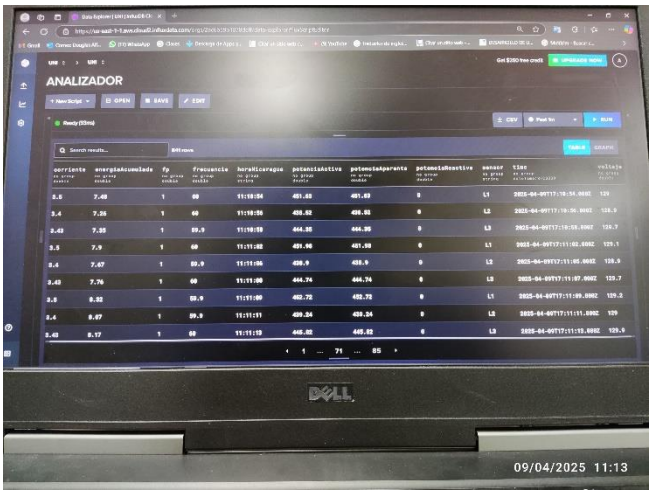




Anexo 24.13: Medición de potencia aparente (VA) y factor de potencia (FP) en las tres líneas con el método de los tres vatímetros.



Anexo 24.14: Visualización de datos en base de datos Influx DB.



Anexo 24.15: Tabla comparativa de mediciones del prototipo con respecto a los tres vatímetros donde la carga fueron resistencias eléctricas.

TABLA COMPARATIVA DE RESULTADOS DE MEDICÓN				
ESPECIFICACIONES DE LA CARGA SEGÚN ETIQUETA:		V= 220/127 V		R1= 36.72Ω , R2= 37.37Ω , R3= 37.70Ω
	POWER METERS	PROTOTIPO	% ERROR	% ERROR PROMEDIO
V L1	128.00	129.00	0.78%	0.81%
V L2	128.00	128.90	0.70%	
V L3	129.00	129.70	0.54%	
A L1	3.42	3.50	2.29%	
A L2	3.42	3.40	0.58%	
A L3	3.41	3.43	0.58%	
W L1	444.00	461.63	3.82%	
W L2	440.00	438.52	0.34%	
W L3	446.00	444.35	0.37%	
VA L1	441.00	461.63	4.47%	
VA L2	439.00	438.52	0.11%	
VA L3	444.00	444.35	0.08%	
FP L1	1.00	1.00	0.00%	
FP L2	1.00	1.00	0.00%	
FP L3	1.00	1.00	0.00%	
* NOTA: PRUEBAS EN CONDICIONES DE LABORATORIO				

Anexo 24.16: Tabla comparativa de mediciones del prototipo con respecto a los cálculos teóricos donde la carga fueron inductores.

TABLA COMPARATIVA DE RESULTADOS DE MEDICÓN				
ESPECIFICACIONES DE LA CARGA SEGÚN ETIQUETA:		V= 220/127 V		I1= 3.5 A I2= 3 A I3= 2.5 A
	Calculos	PROTOTIPO	% ERROR	% ERROR PROMEDIO
V L1	127.00	126.70	0.24%	3.81%
V L2	126.00	125.90	0.08%	
V L3	127.00	126.60	0.31%	
A L1	3.50	3.46	1.14%	
A L2	2.97	2.90	2.36%	
A L3	2.47	2.47	0.00%	
W L1	36.00	41.50	13.25%	
W L2	30.00	34.73	13.62%	
W L3	28.00	29.66	5.60%	
VAR L1	443.07	434.86	1.85%	
VAR L2	373.02	363.96	2.43%	
VAR L3	313.17	310.78	0.76%	
VA L1	444.50	436.84	1.72%	
VA L2	374.22	365.61	2.30%	
VA L3	314.45	312.20	0.72%	
FP L1	0.08	0.09	11.11%	
FP L2	0.08	0.09	11.11%	
FP L3	0.09	0.09	0.00%	
* NOTA: PRUEBAS EN CONDICIONES DE LABORATORIO				

Sistema de alimentación
Fuente= 240/120 Y-Δ
Frecuencia=60hz
Inductancias
$L_1 = 0.67H$
$L_2 = 0.33H$
$L_3 = 0.17H$

La reactancia total del circuito para la fase 1

$x_N = 2*\pi*f*L$
$X_1 = 2*\pi*f*0.67H$
$X_1 = 252.4\Omega$

$X_2 = 2*\pi*f*0.33H$
$x_2 = 124.3\Omega$

$X_3 = 2*\pi*f*0.17H$
$X_3 = 64.1\Omega$

Reactancia equivalente
$X_{eq} = 1/(1/252.4\Omega + 1/124.3\Omega + 1/64.1\Omega)$
$X_{eq} = 36.23\ \Omega$

Intensidad
$I_{L1} = V_{L1-N} / x_{eq}$
$I_{L1} = 127\ V / 36.23\Omega = 3.5A$

Potencia aparente
$S_{L1} = V_{L1-N}$ rms * I rms
$S_{L1} = 127\ V * 3.5\Omega = 444.5$ va

A pesar que el circuito debería poseer propiedades meramente inductivas en la práctica siempre hay pequeñas resistencias internas en los bobinados, la potencia activa (P_{L1}) medida por el power meeter fue de 36w
Factor de potencia
$Fp = (P/S) = 36/444.5$
$FP_{L1} = (0.080)$

Potencia reactiva
$Q_{L1} = S_{L1}$ seno($\cos^{-1}(0.080)$)
$Q_{L1} = 444.575$ seno($\cos^{-1}(0.080)$)
$Q_{L1} = 443.075$var

La reactancia total del circuito para la fase 2

$x_N = 2*\pi*f*L$
$X_2 = 2*\pi*f*0.33H$
$X_2 = 124.3\Omega$
$X_3 = 2*\pi*f*0.17H$
$X_3 = 64.1\Omega$

Reactancia equivalente
$X_{eq} = 1/(1/124.3\Omega + 1/64.1\Omega)$
$X_{eq} = 42.37$

Intensidad
$X_{L2} = V_{L2-N} / x_{eq}$
$x_{L2} = 127\ V / 42.37 = 2.97A$

Potencia aparente
$s_{L2} = V_{L1-N}$ rms * I rms
$s_{L2} = 126\ V * 2.97 = 374.22$

Potencia activa ideal (0)
Potencia activa medida en power meeter (P_{L2}) 30w

Factor de potencia
$Fp = (P/S) = 30/374.22$
$FP_{L2} = (0.080)$

Potencia reactiva
$Q_{L2} = S_{L2}$ seno($\cos^{-1}(0.080)$)
$Q_{L2} = 374.22$ seno($\cos^{-1}(0.080)$)
$Q_{L2} = 373.020$ var

La reactancia total del circuito para la fase 3

$x_N = 2*\pi*f*L$

$X_2 = 2*\pi*f*0.67H$
$X_2 = 124.3\Omega$
$X_3 = 2*\pi*f*0.17H$
$X_3 = 64.1\Omega$

Reactancia equivalente
$X_{eq} = 1/(1/252.4\Omega + 1/64.1\Omega)$
$X_{eq} = 51.8\ \Omega$

Intensidad
$X_{L3} = V_{L2-N} / x_{eq}$
$x_{L3} = 127\ V / 51.8 = 2.476A$

Potencia aparente
$s_{L3} = V_{L1-N}$ rms * I rms
$s_{L3} = 127\ V * 2.47 = 314.45$

Potencia activa ideal (0)
Potencia activa medida en power meeter (P_{L3}) 28w

Factor de potencia
$FP_{L3} = (P/S) = 28/314.45$
$FP_{L3} = (0.09)$

Potencia reactiva
$Q_{L3} = S_{L2}$ seno($\cos^{-1}(0.090)$)
$Q_{L3} = 314.45$ seno($\cos^{-1}(0.090)$)
$Q_{L3} = 313.173$ var

Anexo 24.17: Tabla comparativa de mediciones del prototipo con respecto a los cálculos teóricos donde la carga fue un motor trifásico en vacío (conectado en estrella).

TABLA COMPARATIVA DE RESULTADOS DE MEDICÓN				
ESPECIFICACIONES DE LA CARGA SEGÚN ETIQUETA:		P= 500 W	V= 220/127 V	I= 2.3/4 A
	CÁCULOS	PROTOTIPO	% ERROR	% ERROR PROMEDIO
V L1	129.00	128.80	0.16%	1.20%
V L2	129.00	128.40	0.47%	
V L3	130.00	129.50	0.38%	
A L1	1.62	1.64	1.22%	
A L2	1.45	1.45	0.00%	
A L3	1.48	1.49	0.67%	
W L1	33.44	34.13	2.02%	
W L2	22.45	23.02	2.48%	
W L3	42.33	42.10	0.54%	
VAR L1	206.28	204.06	1.08%	
VAR L2	185.69	183.1	1.39%	
VAR L3	187.68	184.86	1.50%	
VA L1	208.98	207.11	0.89%	
VA L2	187.05	184.51	1.36%	
VA L3	192.40	189.44	1.54%	
FP L1	0.16	0.17	5.88%	
FP L2	0.12	0.12	0.00%	
FP L3	0.22	0.22	0.00%	
* NOTA: PRUEBAS EN CONDICIONES DE LABORATORIO				

$S_{L1} = V_{L1\text{-}N} * I_{L1}$
$S_{L1} = 129\ V * 1.62\ A = 208.98\ VA$

$S_{L2} = V_{L2\text{-}N} * I_{L2}$
$S_{L2} = 129\ V * 1.45\ A = 187.05\ VA$

$S_{L3} = V_{L3\text{-}N} * I_{L3}$
$S_{L3} = 130\ V * 1.48\ A = 192.4\ VA$

$P_{L1} = S_{L1} * FP_{L1}$
$P_{L1} = 208.98\ VA * 0.16 = 33.44\ W$

$P_{L2} = S_{L2} * FP_{L2}$
$P_{L2} = 187.05\ VA * 0.12 = 22.45\ W$

$P_{L3} = S_{L3} * FP_{L3}$
$P_{L3} = 192.4\ VA * 0.22 = 42.33\ W$

$Q_{L1} = S_{L1} * \text{sen}\ (\cos^{-1}(FP_{L1}))$
$Q_{L1} = 208.98\ VA * \text{sen}\ (\cos^{-1}(0.16))$
$Q_{L1} = 208.98\ VA * \text{sen}\ (80.79°) = 206.28\ VAR$

$Q_{L2} = S_{L2} * \text{sen}\ (\cos^{-1}(FP_{L2}))$
$Q_{L2} = 187.05\ VA * \text{sen}\ (\cos^{-1}(0.12))$
$Q_{L2} = 187.05\ VA * \text{sen}\ (83.11°) = 185.69\ VAR$

$Q_{L3} = S_{L3} * \text{sen}\ (\cos^{-1}(FP_{L3}))$
$Q_{L3} = 192.4\ VA * \text{sen}\ (\cos^{-1}(0.22))$
$Q_{L3} = 192.4\ VA * \text{sen}\ (77.29°) = 187.68\ VAR$

Anexo 24.18: Tabla comparativa de mediciones del prototipo con respecto a los cálculos teóricos donde la carga fueron resistencias eléctricas.

TABLA COMPARATIVA DE RESULTADOS DE MEDICÓN				
ESPECIFICACIONES DE LA CARGA SEGÚN ETIQUETA:		V= 220/127 V		R1= 36.72Ω , R2= 37.37Ω , R3= 37.70Ω
	CÁLCULOS	PROTOTIPO	% ERROR	% ERROR PROMEDIO
V L1	128.00	129.00	0.78%	0.75%
V L2	128.00	128.90	0.70%	
V L3	129.00	129.70	0.54%	
A L1	3.48	3.50	0.57%	
A L2	3.42	3.40	0.58%	
A L3	3.40	3.43	0.87%	
W L1	445.44	461.63	3.51%	
W L2	437.76	438.52	0.17%	
W L3	439.60	444.35	1.07%	
VA L1	445.44	461.63	3.51%	
VA L2	437.76	438.52	0.17%	
VA L3	439.60	444.35	1.07%	
* NOTA: PRUEBAS EN CONDICIONES DE LABORATORIO				

$I_{L1} = V_{L1\text{-}N} / R_1$
$I_{L1} = 128\ V / 36.72\ \Omega = 3.48\ A$

$I_{L2} = V_{L2\text{-}N} / R_2$
$I_{L2} = 128\ V / 37.37\ \Omega = 3.42\ A$

$I_{L3} = V_{L3\text{-}N} / R_3$
$I_{L3} = 129\ V / 37.70\ \Omega = 3.40\ A$

$S_{L1} = V_{L1\text{-}N} * I_{L1}$
$S_{L1} = 128\ V * 3.48\ A = 445.44\ VA$

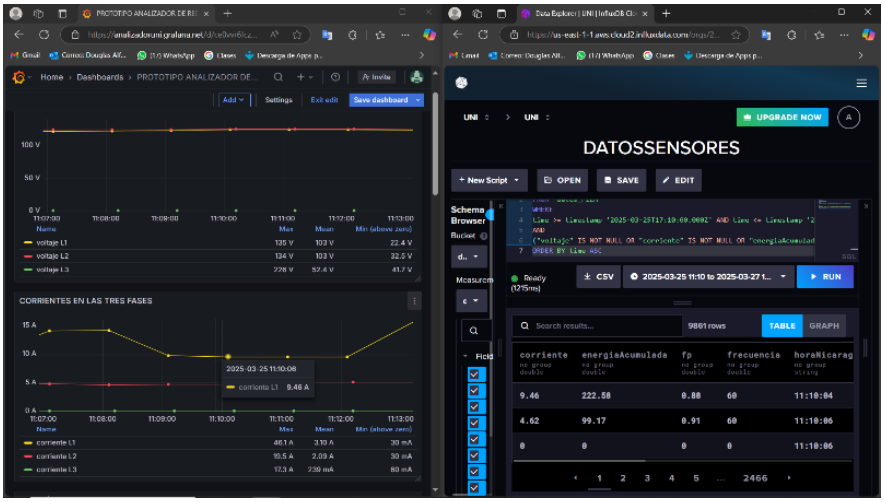
$S_{L2} = V_{L2\text{-}N} * I_{L2}$
$S_{L2} = 128\ V * 3.42\ A = 437.76\ VA$

$S_{L3} = V_{L3\text{-}N} * I_{L3}$
$S_{L3} = 129\ V * 3.40\ A = 439.60\ VA$

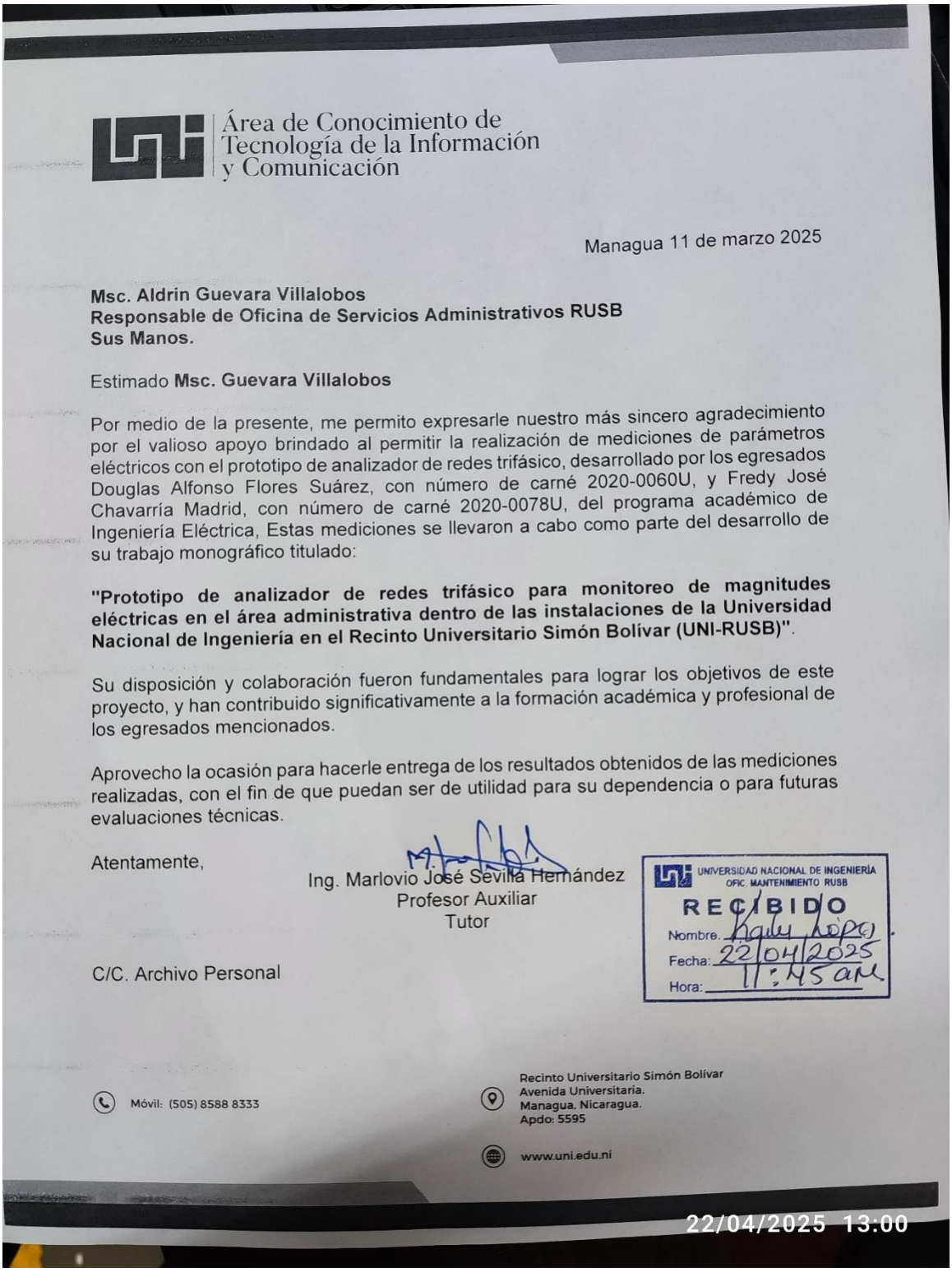
$S_{L1} = P_{L1} = 445.44\ W$
$S_{L2} = P_{L2} = 437.76\ W$
$S_{L3} = P_{L3} = 439.60\ W$

*Al ser una carga totalmente resistiva el factor de potencia es 1 en todas las líneas; además, la potencia reactiva es igual 0, por ese motivo se omite de la tabla.

Anexo 25: Monitoreo y visualización en tablero dentro de la Residencia Estudiantil UNI-RUSB.



Anexo 26: Carta de agradecimiento al responsable de servicios administrativos, donde se adjuntó el documento de los resultados de las mediciones; con sello de recibido por parte de su despacho.



Área de Conocimiento de
Tecnología de la Información
y Comunicación

Managua 11 de marzo 2025

Msc. Aldrin Guevara Villalobos
Responsable de Oficina de Servicios Administrativos RUSB
Sus Manos.

Estimado **Msc. Guevara Villalobos**

Por medio de la presente, me permito expresarle nuestro más sincero agradecimiento por el valioso apoyo brindado al permitir la realización de mediciones de parámetros eléctricos con el prototipo de analizador de redes trifásico, desarrollado por los egresados Douglas Alfonso Flores Suárez, con número de carné 2020-0060U, y Fredy José Chavarría Madrid, con número de carné 2020-0078U, del programa académico de Ingeniería Eléctrica, Estas mediciones se llevaron a cabo como parte del desarrollo de su trabajo monográfico titulado:

"Prototipo de analizador de redes trifásico para monitoreo de magnitudes eléctricas en el área administrativa dentro de las instalaciones de la Universidad Nacional de Ingeniería en el Recinto Universitario Simón Bolívar (UNI-RUSB)".

Su disposición y colaboración fueron fundamentales para lograr los objetivos de este proyecto, y han contribuido significativamente a la formación académica y profesional de los egresados mencionados.

Aprovecho la ocasión para hacerle entrega de los resultados obtenidos de las mediciones realizadas, con el fin de que puedan ser de utilidad para su dependencia o para futuras evaluaciones técnicas.

Atentamente,

Ing. Marlovio José Sevilla Hernández
Profesor Auxiliar
Tutor

UNIVERSIDAD NACIONAL DE INGENIERÍA
OFIC. MANTENIMIENTO RUSB
RECIBIDO
Nombre: Kayla López
Fecha: 22/04/2025
Hora: 11:45 am

C/C. Archivo Personal

Móvil: (505) 8588 8333

Recinto Universitario Simón Bolívar
Avenida Universitaria.
Managua, Nicaragua.
Apdo: 5595

www.uni.edu.ni

22/04/2025 13:00

Anexo 27: Tabla de precios

Precios de Componentes		
Cantidad	**Componente**	**Precio**
1	ESP-WROOM-32	C$ 476.06
1	Tarjeta PCB Perforada	C$ 128.17
6	Fusibles de 100mA	C$ 124.51
3	PZEM-004T	C$1,208.46
-	Estaño y Jumpers	C$ 201.41
1	Adaptador Tarjeta SD	C$ 238.03
1	RTC DS3231 + batería 3V	C$ 274.65
1	Pantalla LCD TFT Shield 2.4”	C$ 439.44
1	Mega2560	C$ 915.50
3	Transformadores de corriente	C$ 915.50
1	Caja de conexiones	C$ 512.68
4	Caimanes y Cables Conductores	C$ 219.72
3	Conectores Presa Stopa	C$ 128.17
1	Bateria para alimentación	C$ 256.34
1	Cargador para alimentación ESP32	C$ 76.90
-	Mano de obra	C$1,831.00
Total		C$7,946.54