Universidad Tecnológica Nacional
Facultad Regional Villa María
Departamento de Electrónica
Cátedra Trabajo Final de Grado

Medidor monofásico IoT

Autor/es:

SANABRIA, Angel Javier

Tutor/es:

Ingeniero en Electrónica José Luis Catalano

Director:

Ingeniero en Electrónica Fabián Marcelo Sensini

2021

Acreditación:

Fecha:

Comité Evaluador

Presidente: Apellido y Nombres

1º Vocal: Apellido y Nombres

2º Vocal: Apellido y Nombres

Dedicatorias

A mi familia y amigos. Siempre fueron parte para que esto pase.

Agradecimientos

Por acompañarme en los años de carrera a familia, amigos y la UTN FRVM. Por acompañarme en mi desarrollo profesional a las empresas que me tomaron como empleado y hacerme sentir ingeniero aun antes de presentar este trabajo.

Resumen

El proyecto del medidor de consumo monofásico se desarrolla bajo el concepto de la denominada "Internet de las cosas" (Internet of things, IoT), una de las tendencias tecnológicas más nombradas en los últimos años. El medidor tiene la función de mostrar el consumo eléctrico de un hogar o una carga elegida. El IoT potencia al medidor sumándole las características de poder monitorear los parámetros medidos en tiempo real e históricos de manera simple a través de una computadora, o celular, con acceso a internet.

El proyecto se puede dividir en tres módulos. Uno de medición, que es el que se encarga de obtener las señales de corriente y tensión de sistema a monitorear. Este se comunica por UART al módulo Main (principal). Este segundo módulo, al recibir la data del módulo de medición, la muestra en un display y puede disparar alarmas (o triggers) cuando se excede un consumo o potencia instantánea máxima seteada desde la web. El tercero es el módulo IoT, se comunica con el módulo Main y tiene la función de conectar a internet el sistema.

***Palabras clave**: Internet de las cosas (Internet of Things, Iot), Wifi, Valor eficaz cuadrático (RMS), Microcontrolador, Firmware, Dashboard, HTTP (Protocolo de transferencia de hipertextos), MQTT (Message Queue Telemetry Transport), Medición de energía, ThingSpeak, CloudMQTT.*

ÍNDICE

Introducción

El Internet de las Cosas (Internet of Things, IoT) es una de las tecnologías de mayor crecimiento en los últimos años. Día a día el número de dispositivos de todo tipo conectados a internet es mayor y cada vez se expanden más sus áreas de aplicación. Cómo en agricultura, control industrial, cuidado de la salud en humanos y animales, telemedicina, seguridad, economía, generación y distribución de energía, cosas de la vida cotidiana, casas y ciudades inteligentes entre otros.

Gracias al uso de sensores de cualquier tipo en distintos ámbitos combinada con la posibilidad de conectarlos a internet nos da por resultado cantidades de datos que posteriormente se convertirán en información útil para tomar decisiones que ayuden a mejorar la productividad, optimizar procesos, economizar en uso de recursos, aumentar el impacto económico, reducir riesgos, encontrar soluciones o lo que sea que incida positivamente en nuestro caso de estudio.

Además de sensores para la recolección de datos, un dispositivo IoT puede estar diseñado con actuadores para realizar acciones de forma remota, por medio de la orden de una persona, otro dispositivo IoT o alarmas en base a datos tomados por el mismo dispositivo u otros. El agregado de circuitos de potencia, de control de switches, solenoides, relés o transductores más sofisticados son los necesarios para convertir las señales digitales del dispositivo IoT y hacer que sucedan cosas grandes o de impacto físico para el sistema.

El consumo energético está pasando a ser un tema a tomar en cuenta en nuestro país por los actuales y venideros cambios en las políticas del mercado de la energía. Por lo que conocer el consumo en tiempo real e histórico en días, semanas y mes en el hogar, o de una carga específica, va a pasar a ser una herramienta útil. La monitorización va a traer ventajas desde puntos de vista de control, ahorro, seguridad y también para obtener un perfil de consumo, que puede ayudar a cálculos para la instalación de equipos de energía solar fotovoltaica que están de moda últimamente.

En la industria la medición de consumo es clave. Obtener información del consumo de distintos sectores de producción, o cargas específicas como pueden ser los motores, ayudan a tomar mejores soluciones para ahorro o uso eficiente de la energía disponible. Una de las ventajas más importantes que va a presentar el proyecto en cuestión aplicado en la industria es la de recibir alarmas al pasar picos de consumo máximo para evitar multas por partes del proveedor de energía. Además de sectores de producción, la monitorización puede aplicarse a cargas especiales para obtener sus perfiles de consumo y poder detectar fallas tempranas con métodos de análisis preventivos.

Se va a desarrollar una página web del tipo Dashboard para el medidor. La comunicación con el medidor es por medio de servicios de plataformas IoT gratuitas. ThingSpeak va a ayudar a hacer los gráficos y tablas históricos con día y hora de los parámetros y MqttCloud se va a encargar de la muestra de los valores en tiempo real y la detección y comunicación de alarmas.

Se van a mostrar los parámetros de tensión, corriente, factor de potencia y energía consumida en tiempo real. Y de forma histórica los de energía consumida al día, semana y mes.

Con el uso de las plataformas IoT mencionadas se automatizan las tareas de recogida, almacenamiento y visualización de información juntándolas en una única página web de fácil manejo y lectura.

Justificación

El proyecto está orientado a demostrar que con lo desarrollada que está la tecnología del IoT tanto en hardware como software, en gran parte, gracias a la comunidad de desarrolladores en internet. Se puede dar conectividad a casi cualquier dispositivo electrónico que creemos. El poder transformar los dispositivos en "inteligentes o smart", recibiendo o transmitiendo información hace que se nos faciliten tareas y vuelve más eficiente nuestro entorno.
En este caso se transforma un medidor de consumo de energía eléctrica. Pero el mismo concepto puede ser aplicado a cualquier dispositivo que creamos necesario controlar u obtener métricas en internet. Las métricas podrían guardarse en una memoria sd o pendrive, pero hoy en dia internet es una parte fundamental de nuestras vidas, por lo que la facilidad de ver esos datos desde un celular con conexión a internet la hace la opción más conveniente.

Objetivos

Diseño y desarrollo de un dispositivo medidor de consumo monofásico no invasivo para una fácil instalación con transformadores de tensión y corriente con posibilidad de subir los parámetros medidos a internet a través de una conexión WiFi.

Los parámetros a medir son tensión y corriente (RMS), factor de potencia, potencia y energía (kW-h). La corriente medida va a ser de 0 a 100 amperes.

Se va a contar con una pantalla LCD para ver los parámetros medidos y un circuito para la activación de una alarma indicadora de potencia instantánea máxima o consumo máximo excedido en el día.
Por medio de una página web se va a poder monitorizar los valores en tiempo real de todos los parámetros medidos y de forma histórica de la energía consumida por medio de gráficos en tiempo de días, semanas y meses. Desde esta página también se van a poder configurar los valores de alarma de potencia instantánea máxima y de consumo máximo en el día.

Objetivos general

Diseñar un circuito para la medición de consumo hogareño o de una carga no invasiva, donde el dispositivo sea no invasivo, es decir, que no se tenga que intervenir en las líneas de alimentación y sea de fácil instalación.

Objetivos específicos

- Diseñar un circuito en base a amplificadores operacionales para la medición de corriente y tensión con transformadores.
- Mostrar los datos del circuito de medición de forma local con un display LCD y de forma remota con dashboard en una página de internet.
- Simular las mediciones del circuito para el ajuste de parámetros.
- Tomar mediciones del circuito real con algún parámetro como puede ser un multímetro.

Internet de las Cosas

Introducción

El internet de las cosas o simplemente IoT, es un término muy presente y utilizado en la actualizado que hace referencia a la conexión digital entre dispositivos con acceso a internet que dan información del entorno que los rodea, pudiendo interactuar con este a partir de ordenes manuales dadas por un operador o automatizadas como resultado del análisis de los datos recolectados con la finalidad de mejorar la calidad de vida de las personas.

El concepto de *internet de las cosas* fue propuesto en 1999, por Kevin Ashton, en el MIT (Massachusetts Institute of Technology), en donde se realizaban investigaciones en el campo de la identificación por radiofrecuencia en red (RFID) y tecnologías de sensores. Actualmente, el término *internet de las cosas* se usa con una denotación de conexión avanzada de dispositivos, sistemas y servicios que va más allá del tradicional M2M (Machine to machine - *máquina a máquina*) y abarca una amplia variedad de protocolos, dominios y aplicaciones. [1]

Importancia

Desde sus inicios el IoT se fue involucrando en distintas áreas de aplicación. Pero una de los avances más importantes fue involucrarse con otros paradigmas como la ciencia de datos, el aprendizaje automático, *Big Data* u otros métodos computacionales. Sea cual sea el área de aplicación los millones de datos, y el flujo constante de estos, que pueden ofrecer los dispositivos *IoT* pueden ser aprovechados por cualquiera de las tecnologías mencionadas para ser analizados y/o procesados en función de obtener resultados que mejoren nuestra calidad de vida.

Aplicaciones

El número de aplicaciones y servicios es prácticamente ilimitado, y su uso se va incrementando año a año con la aparición de nuevas tecnologías o la aplicación en nuevos campos. La meta es aplicar el IoT en cualquier campo para mejorar la calidad de la vida humana en múltiples formas.
En nuestro caso el campo de aplicación es la medición de consumo eléctrico. Abajo se deja una lista como muestra de diversas aplicaciones y servicios basados en IoT con ejemplos de aplicación.

- *Edificios inteligentes*: Mejoras en la eficiencia energética y seguridad
- Smart cities: Optimización del transporte público, integración de servicios de seguridad, gestión inteligente de servicios, tráfico y semáforos
- *Transporte*: Seguimiento de transporte público en tiempo real
- *Educación*: Intercambio de informes y resultados en tiempo real, gestión de asistencia
- *Electrónica de consumo*: Heladeras inteligentes, wearables
- *Salud*: Monitoreo de enfermedades crónicas, biochips
- *Domótica*: sensores y actuadores inteligentes para controlar electrodomésticos

- *Smart Cars*: Mantenimiento preventivo y predictivo
- *Agricultura y medio ambiente*: Medición y control de la contaminación del medio ambiente, etiquetas RFID pasivas asociadas a los productos agrícolas
- *Servicios de energía*: Medición inteligente de datos de consumo, análisis y predicción de comportamientos de consumos, pronosticar tendencias y necesidades futuras
- *Conectividad inteligente*: Métodos de autenticación biométrica, análisis de grandes datos
- *Compras*: RFID y otras etiquetas electrónica y lectores, control de inventarios

Tecnología utilizada

Procesadores

Se encargan de administrar la información. Es decir, el uso de los sensores o actuadores de nuestro sistema y la comunicación con internet. Existe la posibilidad en que el mismo modulo con comunicación a internet, ya sea Wifi, GPRS/2G, 3G, 4G/LTE tiene la capacidad de comunicarse con los sensores y actuadores. Pero en la mayoría de los casos se opta por dejar el modulo encargado para la comunicación de internet con solo esa función, y usar un procesador o microcontrolador externo encargado de procesar los datos y tomar decisiones de que información se va a comunicar con el modulo con salida a internet.

Ya que varios sensores tienen un procesamiento complejo de su información o un protocolo de comunicación complejo que es más fácil desarrollar en microcontrolador externo y dedicado a eso, en vez de embeber el firmware como una capa de aplicación del módulo con comunicación a internet. Al tener un microcontrolador a parte, se tiene más libertad en el desarrollo.

También es importante para el caso de que haya sensores a los cuales se tenga que acondicionar la señal, es decir, adaptar la señal de sensor para que puedan ser procesadas correctamente por el convertidor analógico digital del microcontrolador. También hay casos donde se necesitan circuitos externos para amplificar, atenuar, filtrar, asilar, excitar o alinear una señal de sensor.

En la práctica, el acondicionamiento de la señal forma parte de las tareas propias de la ingeniería electrónica, lo cual va a ser una parte fundamental de este proyecto.

Sensores y actuadores en el IoT

Los sensores y actuadores son el hardware que tienen función de unir e interactuar entre la tecnología IoT y el entorno, es decir de recibir y/o proporcionar información digitalizada desde y para el entorno. En la actualidad, se tienen sensores para la medición de muchísimas magnitudes tanto eléctricas, magnéticas, temperatura, humedad, posición, fuerza, aceleración, ruido, etc. En la mayoría de los casos para obtener las mediciones de los sensores se debe utilizar un microcontrolador capaz de leerlas directamente con un protocolo propio del sensor, o con la lectura de alguna tensión o corriente proporcional a la magnitud medida. Y un dispositivo IoT puede tener uno, ninguno o múltiples sensores o actuadores en su hardware.

En cuanto a los actuadores, existen de muchos tipos como hidráulicos, eléctricos, mecánicos, neumáticos, entre otros. Pero la función de estos se resume en que a partir de una señal eléctrica la convierten en una acción en el entorno físico que se encuentran.

Comunicaciones

Los tipos de comunicaciones más usadas y más elegidas para las aplicaciones IoT son Wifi, Celular (2G/3G/4G) y Bluetooth por ser tecnologías rápidas, de bajo costo y sin pago de licencias para su uso. Existen tecnologías para aplicaciones con más alcance o donde no se tiene conexión a internet con las anteriores mencionadas como pueden ser Sigfox, LoRa, NB-Iot, que plantean una solución de enlaces inalámbricos de kilómetros con un consumo moderado de energía. La desventaja del uso de estas tecnologías es el pago de licencias, solicitud de servicios y agregado de antenas para el funcionamiento correcto.

Plataformas IoT

En un proyecto IoT una de las partes claves es cómo mostrar la información. Una vez que se tiene automatizadas tareas como pedidos de datos por eventos o por tiempo, estos se almacenan en bases de datos. Para que esta información sea útil, se necesita analizarla y definir cómo se van a visualizar los datos. Para facilitar esto se usan las plataformas IoT que se encargar de las tareas de gestión y visualización de datos.

En el mercado hay una cantidad enorme de estas plataformas como Amazon AWS IoT, Microsoft Azure iot Suite, FI-WARE, Carriots, etc. Por lo que es necesario considerar algunos puntos claves para su elección:

- *Conectividad y normalización*: Es necesario que incluyan diferentes protocolos y formatos de datos que permitan garantizar una transmisión precisa entre todos los dispositivos.
- *Almacenamiento de datos*: Mas allá del espacio que pueden ocupar nuestros datos, se tienen en cuenta el tipo de bases de datos que la plataforma puede manejar (relacionales, no relaciones).
- *Procesamiento y gestión de las acciones*: El uso de scripts, basado en reglas de evento-acción.
- *Analítica*: Permite llevar a cabo una serie de procesamientos complejos de datos a partir de la agrupación de datos básicos. Posterior, aplicación de técnicas de aprendizaje automático, lo que al final le dará el máximo valor a los datos tomados.
- *Visualización de datos*: Técnicas de visualización para la detección de patrones y tendencias en tiempo real.
- *Interfaces externas*: Incorporación de APIs y SDKs que actúen como interfaces de sistemas de terceras partes.

También existe la posibilidad de usar prestaciones de distintos servicios y combinarlos en un servidor propio para abaratar costos. Por ejemplo, usar la base de datos de AWS, y hacer todo el demás desarrollo en un servidor propio.

Análisis de los protocolos IoT elegidos

Existen varios protocolos utilizados para IoT enfocándose en el bajo consumo de recursos del microprocesador o microcontrolador y poco ancho de banda necesario. Se van a describir los dos protocolos utilizados en este proyecto.

HTTP

El HTTP (Hypertext Transfer Protocol) es la base del modelo cliente-servidor usado para la Web. El método más seguro de implementar en un dispositivo IoT es incluir solo un cliente. Otros protocolos IoT, utilizan HTTP como base para funcionar. Se basa en que un cliente interactúa con un servidor enviando mensajes de consulta o pedidos y el servidor responde sobre el protocolo TCP/IP, como se muestra en la Fig, N° 1. El navegador web de un celular o computadora es el cliente HTTP más común que podemos usar. Igualmente, también es fácil trasladar este protocolo a un dispositivo IoT con stack de protocolo TCP/IP.

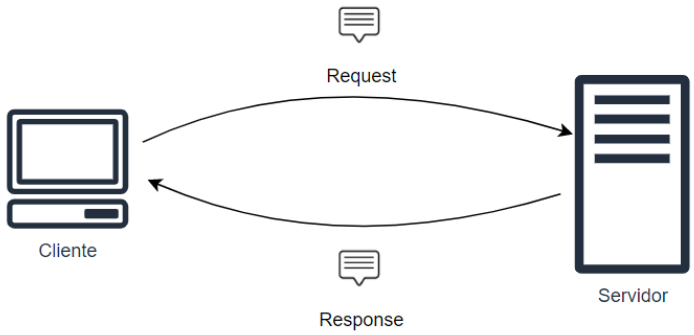


Fig. N° 1 *Comunicación Cliente Servidor.*

Existen muchos tipos de consultas (requests) HTTP que un cliente puede enviar. Dos de las más usadas y comunes para los dispositivos IoT son GET y POST. El método GET es usado para pedir o recuperar datos del server. Y el POST se utiliza para almacenar o actualizar los datos.

MQTT

La tecnología MQTT (Message Queuing Telemetry Transport) es utilizada generalmente en dispositivos con recursos limitados, ya que tiene las características de mensajes ligeros y ocupa poco ancho de banda. Por estas razones es comúnmente usado en dispositivos IoT.
Este protocolo se basa en patrones de comunicación del tipo Publish/Subscribe (publicación subscripción) sobre un tema determinado llamado Topic (tópico). Existe un servidor llamado Broker que centraliza toda la información de los dispositivos que forman parte de la red y es el que se encarga de recibir y distribuir la información. Para recibir los Publish de un topic, los dispositivos deben estar Subscriptos a ese topic. Si no lo están, el broker no le enviará la información. Otra característica destacable del protocolo, es que los dispositivos envían un mensaje del tipo Keep-alive cada cierto tiempo para darle un estatus al bróker de su estado de conexión. También existen mensajes que le avisan al bróker que el dispositivo se va a desconectar de la red, entre otros.

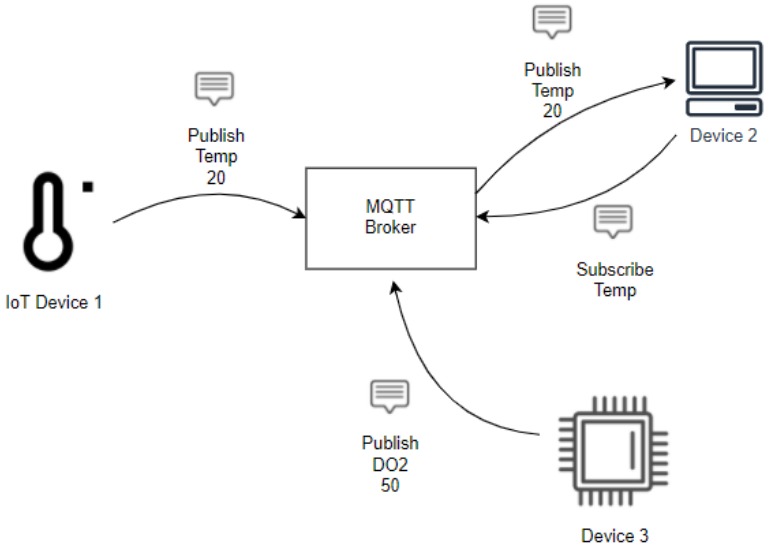


Fig. N° 2 *MQTT en funcionamiento.*

En la Fig, N° 2, se muestra un ejemplo de un esquema MQTT en funcionamiento. El bróker tiene asociados 3 dispositivos. El dispositivo 1, es un sensor temperatura que publica en toppic /Temp el valor de la temperatura leída. El dispositivo 3, es un sensor de dióxido de carbono que publica en el toppic /DO2 su valor leído. Y un dispositivo 2, subscripto solamente al toppic /Temp, por lo que el bróker le publica el valor leído del sensor 1, y no el del dispositivo 3.

Hardware utilizado

En esta sección se detallan los componentes de hardware más destacables del sistema. Se dejan fuera conectores, fuentes de alimentación, borneras, transistores, capacitores, resistencia, optoacopladores, relés, display lcd, cables y demás componentes que son básicos para desarrollos de proyectos de electrónica y prototipado.

Microcontrolador PIC12F1840

El PIC12F1840 es un microcontrolador de Microchip de la familia PIC12. Este va a ser el usado para la placa medición de corriente del proyecto, Es de un tamaño reducido, de 32 MHz de oscilador interno y de solo 8 pines. En la siguiente figura se muestra un diagrama, Fig, N° 3.

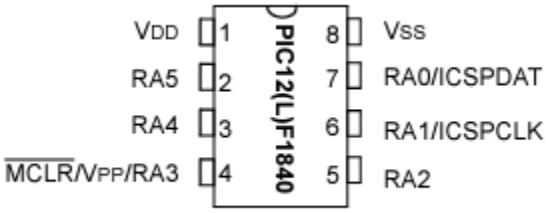


Fig. N° 3 *PIC12F1840 de Microchip [2].*

Características destacables:

- *4 Kbytes Data EEPROM*
- *256 Bytes SRAM*
- *32 MHz de oscilador interno*
- *Oscilador interno de 16 MHz*
- *Conversor analógico a digital (ADC)*
  - *10 bits de resolución hasta 30 canales*
  - *Capacidad de auto adquisición*
- *2.3 a 5.5V de alimentación*
- *Precisión 16 MHz Internal Oscillator Block*
- *Analog-to-Digital Converter (ADC) module*
  - *10-bit resolution, up to 4 external channels*
  - *Auto-acquisition capability*
- *Un módulo "Enhanced Universal Synchronous Asynchronous Receiver Transmitter" (EUSART)*

El datasheet completo se cita en la referencia [2].

Microcontrolador PIC18F23K22

El PIC18F26K22 es un microcontrolador de Microchip de la familia PIC18. Este va a ser el usado para la placa Main del proyecto, En la siguiente figura se muestra un diagrama, Fig, N° 4.

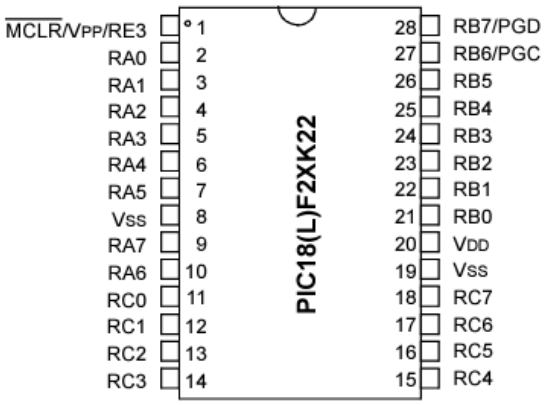


Fig. N° 4 *PIC18F26K22 de Microchip [3].*

Características destacables:

- *1024 Bytes Data EEPROM*
- *64 Kbytes Linear Program Memory Addressing*
- *Oscilador interno de 16 MHz*
- *Conversor analógico a digital (ADC)*
  - *10 bits de resolución hasta 30 canales*
  - *Capacidad de auto adquisición*
- *Precisión 16 MHz Internal Oscillator Block*
- *Analog-to-Digital Converter (ADC) module*
  - *10-bit resolution, up to 30 external channels*
  - *Auto-acquisition capability*
- *2.3V a 5.5V de alimentacion*
- *Dos módulos "Enhanced Universal Synchronous Asynchronous Receiver Transmitter" (EUSART)*

El datasheet completo se cita en la referencia [3].

Amplificador operacional

El MCP6001/2/4 es un amplificador operacional de Microchip diseñado para aplicaciones de propósito general, ver Fig, N° 5. Es del tipo rail-to-rail, por lo que es capaz de manipular voltajes cercanos a la alimentación. También no necesita una alimentación simétrica con referencia de voltaje negativa. Se usan 2 de los amplificadores operacionales para el acondicionamiento de las señales de tensión y corriente a medir como salida de los transformadores.

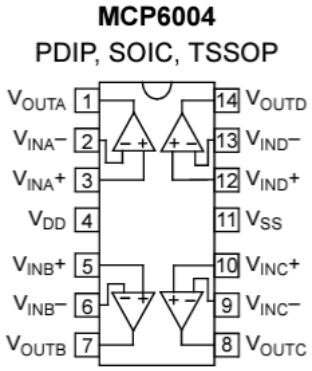


Fig. N° 5 Amplificadores operacional *MCP6004 de Microchip [4]*.

Características:

- Ancho de banda: 1MHz
- Tipo: Rail-to-rail input/output
- Alimentación: 1.8V a 6.0V
- Corriente de consumo: Iq = 100 uA

El datasheet completo se cita en la referencia [4].

Regulador de regencia ajustable TL431CLP

El TL431CLP es un dispositivo regulador de referencia de 3 pines. El voltaje de salida puede ser seteado entre 2.5V a 36V con dos resistencias externas. Al no colocar estas resistencias, la referencia es de 2.5V. Este voltaje de 2.5V se usa como el offset en ambos aplicadores operaciones para elevar las señales medidas en la placa de medición.

Regulador MCP1703

El MCP1703 es un regulador de voltaje low dropout (LDO) que puede entregar hasta 250mA con un consumo propio de solo 2 uA. Se usa para regular el voltaje de los microcontroladores y obtener los 5 V de Vdd de la placa. En la Fig, N° 6 se muestra el utilizado.

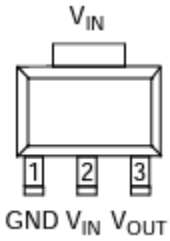


Fig. N° 6 *Regulador MCP1703 de Microchip [5]*.

- 2uA de consumo
- Alimentación: 2.7V a 16V
- Voltaje de salida: 5.0V

- Protección contra cortocircuito
- Protección de sobre temperatura

El datasheet completo se cita en la referencia [5].

ESP8266 y NODEMCU

El ESP8266 es una placa que cuenta con un microcontrolador de propósito genera y un integrado con el stack TCP/IP con antena incorporada para comunicación Wifi, se muestra en la Fig, N° 7.

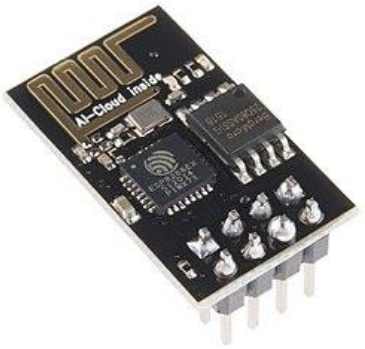


Fig. N° 7 *ESP8266 [6]*.

Viene unas muchas versiones, con antena tipo chip, microstrip o conector para antena externa En nuestro proyecto se usa la placa NodeMCU que tiene como componente principal el ESP8266, la ventaja de usar está placa es que tiene un circuito de comunicación serial a USB para poder programarlo directamente con la computadora, ver F Fig, N° 8. Además, de unas tiras de pines para poder usar los pines de comunicación y propósito general del microntrolador.

Fig. N° 8 *NodeMCU [7]*.

El ESP8266 puede ser programado en 3 diferentes lenguajes que son LUA, MicroPython y C. En nuestro proyecto usamos el lenguaje de programación C, con el IDE de Arduino.
Al ser parte del proyecto Arduino, se tienen muchas librerías de código abierto desarrolladas por la comunidad, y esto facilita el desarrollo con los usos de los protocolos HTTP y MQTT necesarios para este proyecto en funciones de alto nivel.

Medición de señales

Transformadores

El dispositivo ni el circuito de medición están diseñados para soportar tensiones y corrientes altas como pueden ser los 220 V AC. Además, sería peligroso estar en contacto con esta señal. Por lo que estas magnitudes se van a medir a través de un transformador de tensión y uno de intensidad o corriente. En estos transformadores se considera el devanado primario bajo la acción de la magnitud que se quiere medir y el secundario conectado al circuito de acondicionamiento de la señal para que pueda ser muestreada por el microcontrolador. De esta forma se aísla el circuito de medición de las magnitudes a medir y se baja su amplitud a niveles trabajables.

Transformador de tensión

La relación de transformación, es la fórmula resultante de la relación de espiras de los devanados, la tensión aplicada y la resultante, y es igual a la siguiente expresión (Ecu. 4)

$$m_U = \frac{U_1}{U_2} \qquad \textit{(Ecu. 4)}$$

Este transformador se conecta en paralelo, el devanado primario va en paralelo con el circuito a medir. Y el secundario a los bornes del circuito de medida. Ver Fig, N° 9.

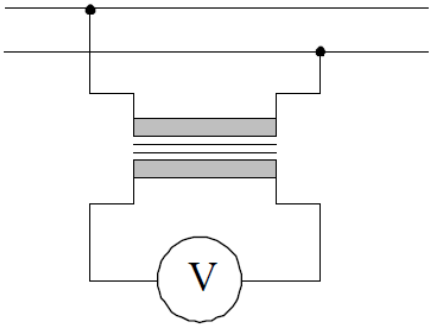


Fig. N° 9 *Transformador de tensión.*

Transformador de corriente

En nuestro caso usamos un transformador de núcleo dividido o también llamado toroidal. Es decir, un núcleo que permite una apertura y después el cierre. Esto otorga la ventaja de que, para pasar el cable a medir, no es necesario cortarlo ni desconectarlo para que quede dentro del núcleo, sino que funciona tipo pinza amperométrica abriéndose rodear el cable y después cerrando para unir el núcleo. Dándole la característica de no invasivo a nuestro sistema. Ver Fig, N° 10.

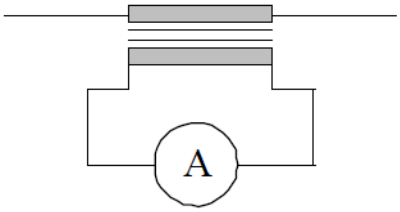


Fig. N° 10 *Transformador de corriente.*

El funcionamiento se basa en que el flujo magnético alterno del cable a medir se induce sobre el núcleo que lo rodea y se acopla al secundario. El primario se considera de una sola espira ya que pasa por dentro del núcleo y la relación de transformación puede resumirse en la siguiente ecuación (Ecu 5)

$$m = \frac{I_1}{I_2} \qquad \textit{(Ecu. 5)}$$

Donde:

- I_1 es la corrinte del primario
- I_2 es la corriente del secundario

Principios de la medición de energía

El medidor tiene como objetivo la medición de energía que es la potencia activa en el tiempo. La potencia, en términos simples es el producto de la tensión y corriente. Por lo que el primer paso es obtener las medidas de tensión y corriente. Se va a describir el análisis de una señal suponiendo que es la del voltaje, pero es el mismo análisis para corriente.

True RMS

La forma más común de describir o analizar las señales eléctricas es por la medición del valor eficaz RMS (Root mean square - Valor cuadrático medio) de tensión y corriente. El RMS en el tiempo se define como la siguiente función (Ecu. 12).

$$V_{rms} = \sqrt{\frac{1}{T} \cdot \int_0^T v^2(t)\, dt} \qquad \textit{(Ecu. 7)}$$

En nuestro caso necesitamos las funciones de forma discreta ya que necesitamos hacer los cálculos con un microcontrolador de gama media-baja, por lo que calcular una integral no es posible.

En que en nuestro país contamos con señales sinodales de 50 HZ. Aprovechando el análisis de la onda sinodal, se puede aproximar el valor RMS de la señal como (Ecu. 7):

$$V_{rms} = \frac{V_{pico}}{\sqrt{2}} \qquad \textit{(Ecu. 7)}$$

Donde:

- V_{pico} Valor máximo de la señal de voltaje

Esto puede ser aprovechado como una forma de analizar las señales con pocos recursos, ya que con solo obtener el valor pico, y suponiendo que las señales son senoidales se puede llegar a un resultado aproximado.

En nuestro caso, aprovechando el uso del microcontrolador, podemos usar la medida de True RMS (True root mean square - Verdadero valor eficaz). Está toma varias muestras de los valores de la señal a lo largo de un ciclo y utilizando la siguiente ecuación (Ecu. 8)

$$V_{TRUE\ rms} = \sqrt{\frac{{V_1}^2 + {V_2}^2 + \ldots + {V_n}^2}{n}} \qquad \textit{(Ecu. 8)}$$

Donde:

- n Es el número de muestras tomadas

Aplicando está medida tenemos una medición más aproximada a la real comparada con el resultado del valor RMS. Además, tenemos en cuenta que las señales no son senoidales puras, sobre todo la de corriente, ya que al pasar por cargas que no son resistivas puras se deforma la señal.

Voltaje alterno

En nuestro país la bajada de línea eléctrica para las casas comúnmente es de 200 V de corriente alterna, a excepción de las que necesitan línea trifásica. Esto quiere decir que la tensión varia periódicamente en magnitud y sentido en forma de oscilaciones senoidales a 50 HZ con un valor eficaz de 220 V. Por lo que la forma de onda de corriente alterna va a ser de la misma forma para cargas puramente resistivas, y va a recibir deformaciones para otros tipos de cargas, que son las más comunes dentro de un hogar.

Algunos de los parámetros básicos que vamos a usar para describir este tipo de señales son la frecuencia, el periodo, valor máximo y valor eficaz. Ver Fig. N° 11.

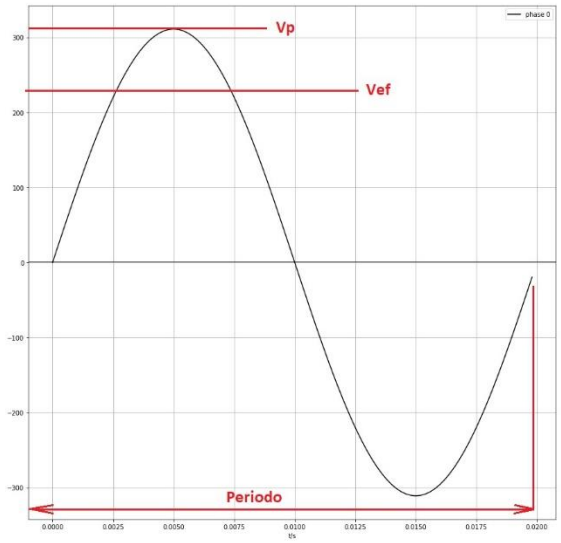


Fig. N° 11 *Transformador de corriente.*

Donde:

- V_p Valor máximo de la señal
- V_{ef} Valor eficaz de la señal
- Periodo: $P = \frac{1}{f}$

Potencia activa

La relación entre el voltaje y la corriente es la potencia.
Se define como Potencia Activa (P) como el valor medio de la potencia instantánea para un periodo completo de la excitación y tiene relación directa con el defasaje de las señales. Ver Ecu. 9.

$$P = V \cdot I \cdot \cos(\varphi) \quad [W] \qquad \textit{(Ecu. 9)}$$

Donde:

- V Valor eficaz de voltaje
- I Valor eficaz de corriente
- $\cos(\varphi)$ Coseno de fi
- φ Angulo de defasaje de la corriente con la tensión

Nótese que la potencia activa es tanto mayor cuanto menor sea fi y si fi es 0 el coseno es 1.

Está potencia se suele llamar potencia útil del sistema, ya que es la que se aprovecha para realizar trabajo en el sistema. [8]

Potencia reactiva

Es parte de la potencia total que no se transforma en trabajo efectivo es decir la potencia que se necesita para generar los campos eléctricos y magnéticos en la carga. Se mide en Volt-Amper reactivo VAr y se define por la siguiente formula (Ecu. 10).

$$Q = V \cdot I \cdot \sin(\varphi) \quad [VAr] \qquad \textit{(Ecu. 10)}$$

- V Valor eficaz de voltaje
- I Valor eficaz de corriente
- φ Angulo de defasaje de la corriente con la tensión

Potencia aparente

Es la suma total de la potencia consumida por la carga. Es decir, el producto de los valores eficaces de tensión y corriente y se mide en volt-ampere VA (Ecu. 11).

$$S = V \cdot I \qquad [VA] \qquad \textit{(Ecu. 11)}$$

Factor de potencia

El factor de potencia representa el valor del coseno del ángulo de desfasaje de la tensión y corriente que toma un valor de 0 a 1, y da la idea de cuánto de la potencia total se aprovecha como potencia activa.

Diseño del Proyecto

Diagrama en bloques

Se va abordar el proyecto como el conjunto de tres módulos indicados en la siguiente figura, Fig, N° 12:

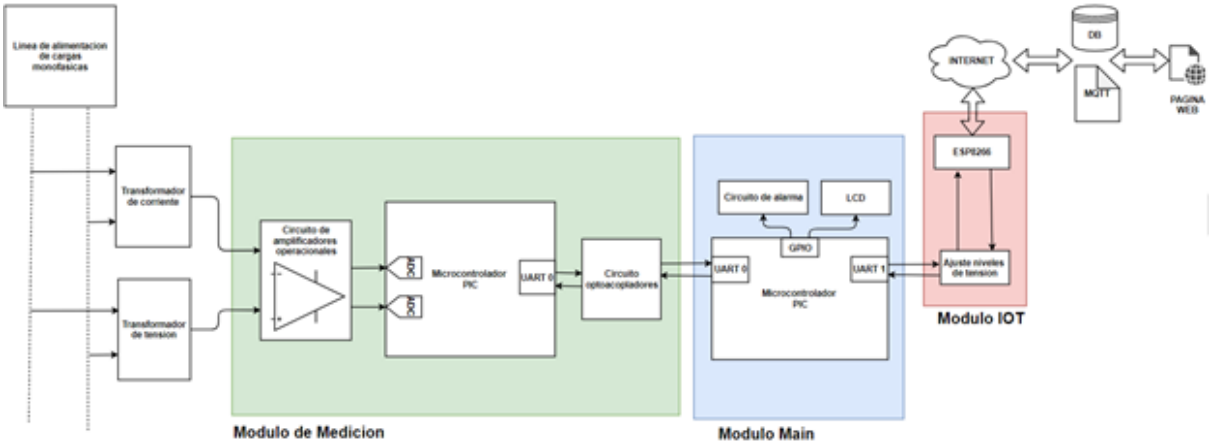


Fig. N° 12 *Diagrama en bloques de todo el sistema.*

Módulo de medición

El módulo de medición se basa en transformadores de corriente y tensión pudiendo ser conectado al sistema sin necesidad de interrumpir el circuito a medir. Por medio de un circuito de amplificadores operacionales se acondicionan las señales a niveles manejables de entradas analógicas de un microcontrolador PIC de Microchip, que se va a encargar de muestrearlas y aplicar los algoritmos necesarios para la obtención de los parámetros deseados. Los principales parámetros medidos son voltaje, corriente, coseno de fi, factor de potencia y frecuencia. Estos datos se van a preparar en base a un protocolo básico para enviarlos al módulo Main por medio de UART y un circuito de optoacopladores para aislar los módulos.

Encargado de obtener las señales de corriente y tensión monofásicas del sistema a monitorear, procesarlas y obtener los parámetros de tensión, corriente, factor de potencia, coseno de fi y frecuencia. Estos datos se los comunica al Módulo Main. No se va a utilizar un circuito integrado comercial especializados en la medición de energía. Sino que, para reducir costos, se van a utilizar unos transformadores de tensión y corriente, un circuito con amplificadores operacionales para

acondicionar las señales y ajustarlas a niveles que un microcontrolador PIC de Microchip pueda muestrear y procesar por medio de módulos ADC. El microcontrolador va a ejecutar el algoritmo de la obtención del valor medio cuadrático (RMS) de las señales de corriente y tensión. También se va a obtener el factor de potencia, coseno de fi y frecuencia.

Módulo Main

Este módulo también es en base a un Microcontrolador PIC. Recibe los datos del módulo de medición y es el encargado de mostrarlos en una pantalla LCD. En base a estos datos y un divisor timer para obtener el tiempo en segundos se calcula potencia y energía en kWh. Ejecuta alarmas (o triggers) cuando se pasa un consumo o potencia máxima en el día o instantánea. También prepara los paquetes con los datos necesarios a subir a la web y se los comunica al Módulo IoT

Módulo IoT

Es el encargado de subir los datos a internet por medio de WiFi cada cierto tiempo y de recibir avisos de internet para setear alarmas o triggers. Se va a usar un ESP8266, módulo de bajo consumo que cuenta con el stack TCP/IP integrado y con una antena para conectarse a internet de forma inalámbrica por WiFi. Se va a comunicar de forma serial con el módulo de medición para el intercambio de parámetros, datos de medición y alarmas.

Conexión del medidor

Se presentó el medidor de consumo IoT con la función de medir el consumo en una casa o para una carga específica. La conexión de este para medir el consumo del hogar se describe en la FIG C, donde se coloca después de las protecciones eléctricas del hogar. Si se quiere usar para medir una carga, es similar, solo que se coloca antes de la carga.
En cualquiera de los casos la conexión es no invasiva, es decir, que no se tiene que abrir, ni modificar la red de alimentación. En el caso de la colocación del transformador de corriente (CT), como es del tipo toroidal de núcleo partido, se usa como una pinza, donde se abraza el cable a medir. Para el caso del transformador de tensión (TT) se prepara en el primario con un enchufe macho para poder ser colocado en cualquier toma corriente de pared cercano. Ver Fig, N° 13.

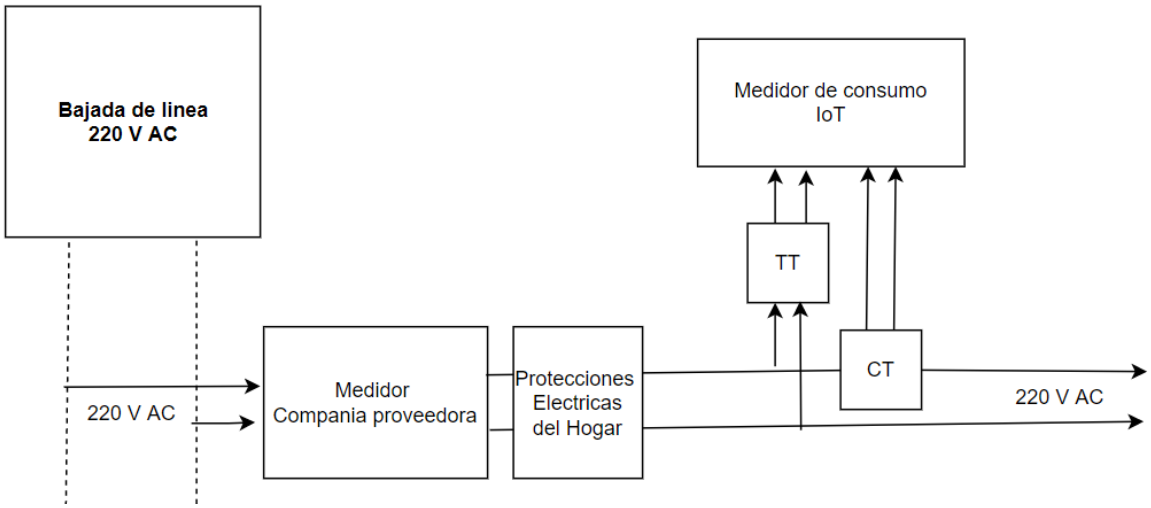


Fig. N° 13 *Conexión del medidor.*

Firmware

El desarrollo de firmare para el *Modulo Main* y el *Modulo de medición*, tienen microcontroladores de las familias PIC12 y PIC18 de Microchip. El lenguaje utilizado para el desarrollo de su firmware es C.

En cuanto al *Módulo IoT*, se desarrolla también en el lenguaje C, pero con el uso del IDE de Arduino para la programación.

Módulo de medición

Esta es la placa encargada de la obtención de los datos de las señales de tensión y corriente. Para estos se aplican algunos algoritmos que se van a describir en esta sección. Los algoritmos son para la obtención de True RMS, frecuencia, coseno de fi, valor máximo. La obtención de estos valores para con las señales de voltaje y corriente van a ser suficientes para hacer el cálculo de potencia activa y consumo. El cálculo de potencia y consumo se va a hacer en el *Modulo Main*. En el ANEXO 1, se muestra el diagrama de flujo del funcionamiento de este módulo.

En esta sección se van a describir los esquemáticos de circuito usados para procesar la señal. También se van a describir los algoritmos usados para obtener los valores de las señales. Se van a describir los aspectos más destacables de este módulo. El esquemático completo de esta placa se va a adjuntar en el ANEXO 2.

Regulador de tensión a 5V

Se usan fuentes de 12V DC que se conectan directamente a la tensión de línea de 220V AC. Se usan dos fuentes para evitar el ruido entre las placas, ya que la comunicación entre estas esta optoacoplada.

Por lo tanto, se necesita un circuito para tener los 5V DC de alimentación en ambos módulos. En la siguiente figura (Fig, N° 14), se muestra el esquema usado para el MCP1703 que es prácticamente el recomendado por el datasheet [5].

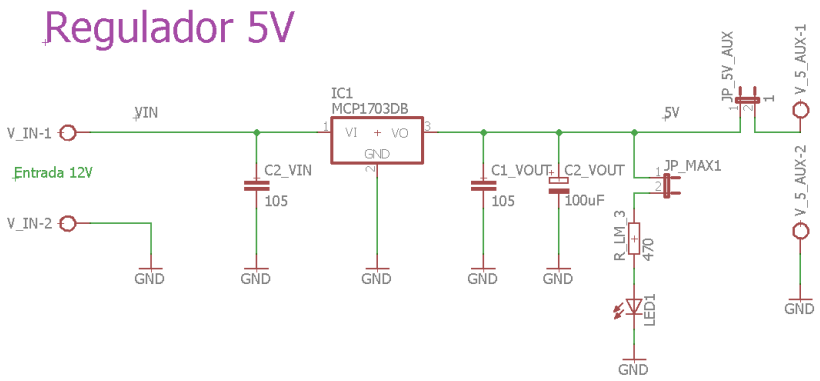


Fig. N° 14 *Circuito regulador de 5V DC.*

Para el funcionamiento de las placas con alimentación externa de 12V DC se tiene que tener conectado el jumper JP_MAX1, y desconectado el jumper JP_5V_AUX. La alimentación externa se conecta en la bornera V_IN.

Si se quiere evitar el uso de la fuente de 12V DC, y se tiene alimentación de 5V DC ya regulados. La conexión y desconexión de los jumpers se hace al revés y se tiene que alimentar con una fuente de 5V DC en la bornera V_5_AUX.

Circuito con amplificadores operaciones para acondicionar las señales

Como se describió anteriormente, se van a obtener niveles de señal de tensión y corriente manejables y seguros para el circuito de medición. En la Fig, N° 15 se representa el diagrama del circuito. Las entradas no inversora e inversora y la salida out del amplificador operacional están representada por los tags A-, A+ y OUT_A.

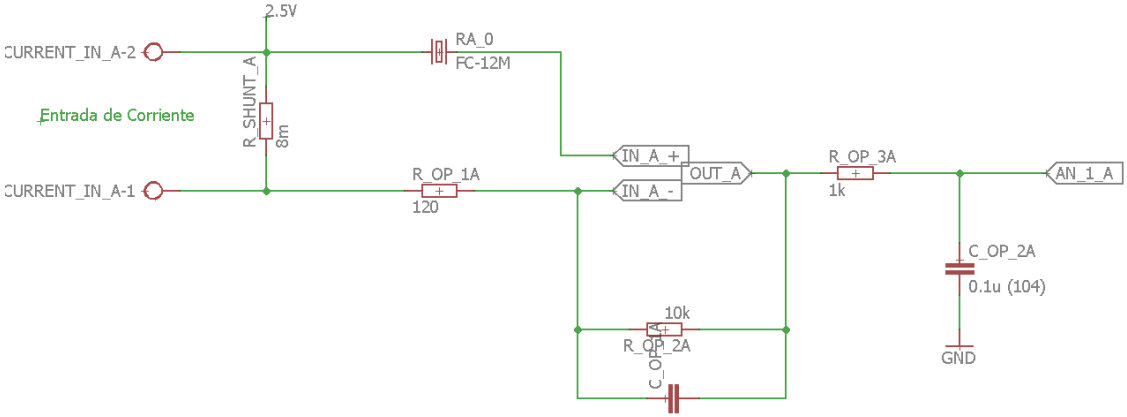


Fig. N° 15 *Circuito acondicionador de señal.*

La resistencia R_SHUNT_A, es la que provoca la señal diferencial para el amplificador operacional

en configuración de inversor. El capacitor C_OP_1A completa la función de filtro pasa bajos para el filtro de la señal.

Se tiene una ganancia del circuito inversor dada por la siguiente ecuación (Ecu. 12)

$$A_v = \frac{R_f}{R_1} \qquad \textit{(Ecu. 12)}$$

- R_f = R_OP_2A
- R_1 = R_OP_AA
- A_v Ganancia del amplificador inversor

Para el cálculo final con el microcontrolador, a esta ganancia se la tiene que multiplicar por la constante resultante del transformador correspondiente de tensión o de corriente y la resistencia shunt. En la siguiente expresión (Ecu. 13), se demuestra agrega la constante del transformador y se presenta la constante final que se va a usar para la transformación del valor True RMS obtenida con el microcontrolador.

$$K_V = K_t \cdot A_v \qquad \textit{(Ecu. 13)}$$

- K_V Constante de conversión final
- K_t Constante del transformador y el shunt
- A_V Ganancia del amplificador inversor

Algoritmo de valor RMS para el microcontrolador

Lo principal a calcular por el microcontrolador es el valor True RMS de las señales. Esto se hace con el convertidor A/D que pasa los valores analógicos a valores digitales. Por lo que se define una etapa de adquisición de datos de las señales, donde el convertidor de ADC de 10 bits se va a encargar de tomar la mayor cantidad de muestras posibles, por lo que se deja el tiempo recomendado del microcontrolador como tiempos de carga del capacitor además del de conversión del ADC para tener mediciones fiables.

Para nuestra señal de 50 Hz, tenemos un periodo de 20 ms. Teniendo en cuenta que el tiempo de carga y conversión del ADC es aproximadamente de $20\ \mu s$. Pero a este tiempo se le va a agregar un tiempo asociado a unas operaciones de conversión del valor leído, estas conversiones son con dato del tipo flotante, por lo que el tiempo es alto comparado con el tiempo de adquisición y conversión del micro. Este tiempo total por muestra queda en aproximadamente $264\ \mu s$.
El algoritmo va a tomar 85 muestras, en un tiempo de $22.5\ ms$ para tener cubierto un ciclo de la señal. Esto va a definir nuestra frecuencia de muestreo 4250 Hz.

Por el teorema de muestreo de Nyquist, (Ecu. 14) se determina la cantidad de muestras necesarias para representar una señal como mínimo dos veces superior al ancho de banda de la señal. En nuestro caso suponemos una señal senoidal, debemos tener un muestreo mayor a dos veces la frecuencia, es decir un mínimo de muestreo de 100 Hz.

$$f_s = 2 \cdot f_N \qquad \textit{(Ecu. 14)}$$

- f_N Frecuencia de Nyquist
- f_s Frecuencia máxima de la señal

Se cumple con el teorema de Nyquist ya la frecuencia de toma de muestras es varias veces mayor a f_N.

Es común hacer una tabla en RAM del microcontrolador para tomar las medidas de las muestras y después hacer todo el cálculo necesario para tener el valor de TRUE RMS. En nuestro caso a tener buena velocidad del microcontrolador comparada se decidió ir aplicando el algoritmo a las muestras tomadas y una vez obtenidas todas se hace la raíz cuadrada que es la que tiene mayor tiempo de procesamiento para el microcontrolador.

La fórmula de conversión final es la siguiente, Ecu. 15.

$$V_{TRUE\ rms} = \sqrt{\frac{{V_1}^2 + {V_2}^2 + \ldots + {V_n}^2}{n}} \qquad \text{(Ecu. 15)}$$

Pero dado que la señal se acondicionó con un offset de 2.5V positivo para elevar la señal y que no pueda tener valores negativos que el conversor ADC no puede soportar. Se puede ver en la Fig, N° 16.

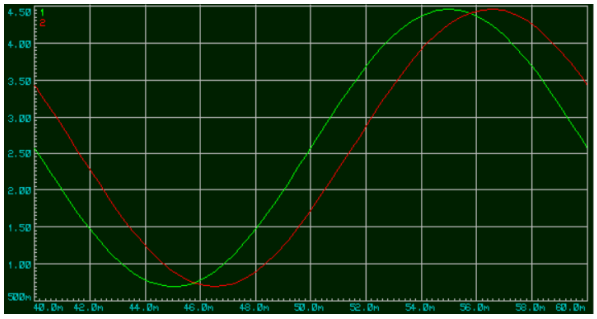


Fig. N° 16 *Acondicionamiento de señal.*

La Ecu. 15 de True RMS tiene que restar ese offset, quedando la ecuación (Ecu. 16).

$$V_{TRUE\ rms} = \sqrt{\frac{(V_1 - Vref)^2 + (V_2 - Vref)^2 + \ldots + (V_n - Vref)^2}{n}} \qquad \text{(Ecu. 16)}$$

A cada una de estas muestras se le aplica la siguiente formula de conversión para dejar el valor multiplicado por la constante de escala, se le resta el offset y se va haciendo la sumatoria de todas las muestras.

```
adc=(float)(valor*KV)-2.5;
// adc=(float)(read_adc()*KV)-2.50;

resultado_rms=(adc*adc)+resultado_rms;
// sumatoria de los valores al cuadrado
```

- KV es la constante de conversión del ADC. Igual a 5.00/1023 = 0.0048875

Y al finalizar la toma de todas las muestras, se hacen las siguientes operaciones al resultado de la sumatoria de sus valores.

```
promedio_rms=promedio_rms/(muestras);          // promedio rms
promedio_rms=sqrt(promedio_rms);               // promedio rms
```

- muestras, es 85 la cantidad total de muestras
- sqrt(), aplica la función de raíz cuadrada al resultado

De esta forma se obtiene el valor True RMS de cada señal. La función de cálculo está definida como:

```
float promedio_adc(int1 canal);
```

Se va aplica el mismo algoritmo con la función anterior en las señales de tensión y de corriente.

La función completa se encuentra en el código del ANEXO 4.

Algoritmo de valor máximo de la señal

Se muestrean las señales como una frecuencia aun mayor que el algoritmo anterior. Esto porque no se tiene que hacer ninguna operación con cada muestra tomada. Sino que solamente al mayor valor del conversor ADC. Se toma una cantidad de muestras tal que se obtenga datos de un periodo completo. A continuación, se muestra el algoritmo.

```
// toma de muestras y sumatoria de la cantidad de muestras seteadas
for ( unsigned int8 i=0; i<= 254 ; i++ ){
      set_adc_channel(canal);
      delay_us(22);
      valor = read_adc();
      if ( valor > valor_m ){
         valor_m = valor;
      }
}

valor_max = (float)(valor_m*KV) - 2.479;
if ( valor_max < 0.01 ) valor_max = 0.00;
```

Una vez obtenido el mayor valor del ADC se aplica la constante de conversión y se resta el valor del offset. También se agregó un filtro de valor mínimo para evitar un máximo demasiado pequeño producto del ruido del sistema.

Algoritmo de coseno de fi

Para la obtención del coseno de fi se buscó obtener los cruces por 0 de las señales, teniendo en cuenta que nuestro 0 es 2.5V del offset. Una vez obtenido el cruce por 0 de la señal de tensión se cambie el adc al de corriente para tener su cruce, aplicar la diferencia de tiempo y pasarlo a grados. Se van a describir los pasos seguidos por el algoritmo. En el código completo del ANEXO 4, se tienen estos pasos como comentarios en la función “int8 cos_fi(void)”.

- Paso 1: Activar el timer con su respectivo PS y setear el Channel 0 del adc
- Paso 2: Realizar lecturas y esperar que la señal sea negativa (menor a 2.5V, o sea, 511 de lectura adc)
- Paso 3: Esperar a que la señal sea igual o mayor que 511 (paso por Cero analógico)
- Paso 4: Setear el timer en 0 y cambiar el analógico al Channel 1.
- Paso 5: Verifico si estoy atrás o adelante de la onda de tensión, Para definir si es capacitiva o inductiva
- Paso 6:
    - a - La señal es menor que el límite, por lo que está atrasada. Espero a que sea mayor y tomo el valor del timer (hago corrección)
    - b - La señal es mayor que el límite, por lo que esta adelantada Espero a que sea menor, le sumo 10ms (mitad del periodo de los 50 Hz) y le aplico la corrección al valor del timer

Algoritmo de frecuencia

Para la obtención de la frecuencia de la señal de voltaje se aplica un algoritmo similar al anterior. Donde se espera el cruce por cero, después de la señal sea negativa, y se cuenta el tiempo hasta el próximo cruce por cero desde señal negativa. En el ANEXO 4, es la función “float frecuencia(void)”.

Conexiones y PCB del módulo de Medición

La placa del módulo de medición tiene las siguientes conexiones:

- Bornera de alimentación de fuente externa
- Bornera de alimentación auxiliar de 5V DC
- Jumpers de selección de fuente de alimentación
- Tira de pines de programación de PIC.
- Tira de pines de comunicación UART con *Modulo Main*

Los PCB fueron diseñados en Eagle 7.6.0. En la Fig, N° 17 y Fig, N° 18, se muestran el top y bottom layer respectivamente. Las líneas punteadas marcan el contorno de los planos de masa definidos en ese layer.

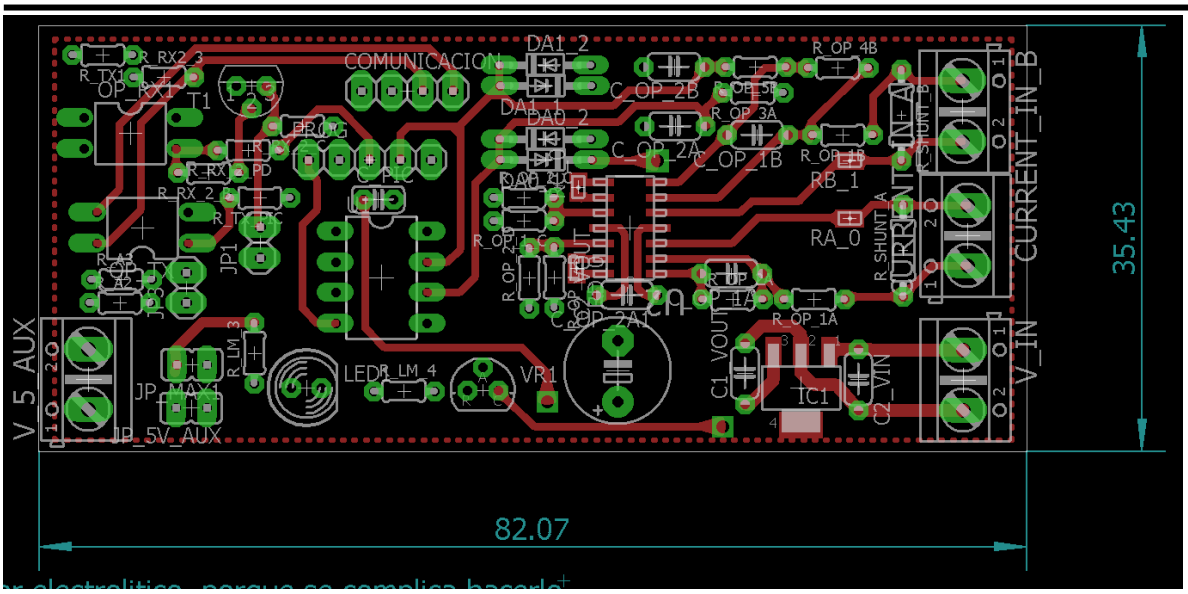


Fig. N° 17 *PCB módulo de medición – Top Layer.*

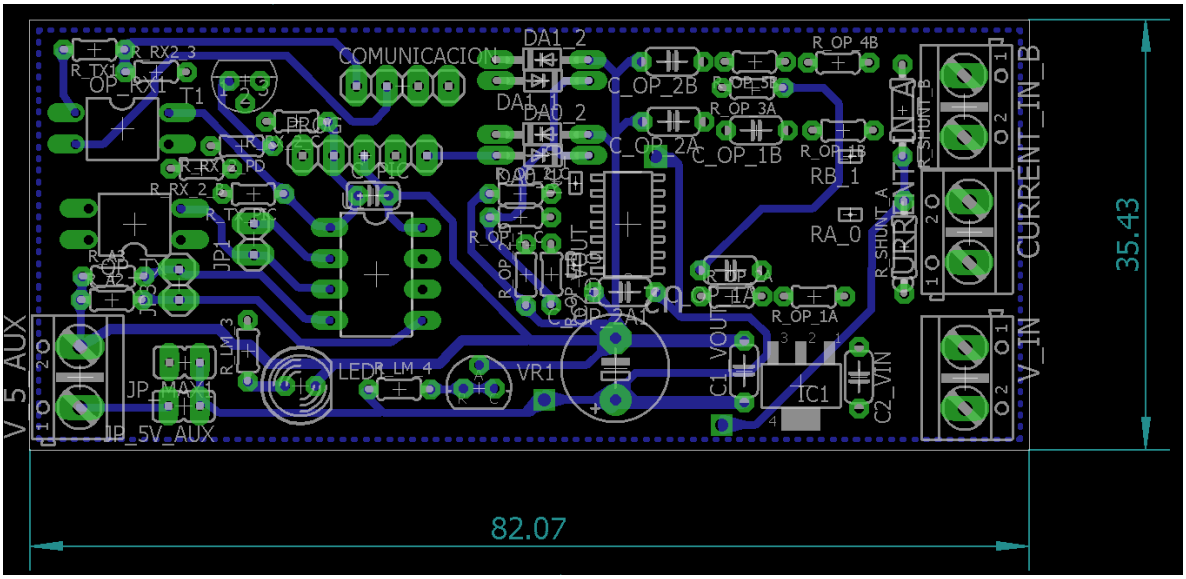


Fig. N° 18 *PCB módulo de medición – Bottom Layer.*

Módulo de Main

A continuación, se listan las funciones de este módulo.

- Recibir datos desde el módulo de medición
- Calculo de potencia y energía
- Mostrar en display LCD las mediciones
- Envío de medidas al *Módulo IoT*

- Alarma de potencia máxima con acción de relé
- Recepción de parámetro de potencia máxima desde el *Módulo IoT* seteado desde la página web.

En esta sección se van a describir algunos de los esquemáticos principales del módulo. El esquemático completo está en el ANEXO 3. También se va a describir el protocolo de comunicación usado entre los módulos.

Medición de energía

A partir de los valores True RMS de tensión y corriente que los comunica el *Modulo de Medición*, se calcula la potencia activa como el producto de estos valores.
Con la potencia podemos calcular la potencia consumida como describe la siguiente expresión (Ecu. 17).

$$E_{kWh/dia} = \frac{P_W \cdot t_{h/dia}}{1000_{W/kW}} \quad [kW\ hora] \qquad \text{(Ecu. 17)}$$

Protocolo de comunicación entre los módulos

Se definió un protocolo sencillo de comunicación entre los módulos. El cual se basa en usar un byte para definir el "Frame type (tipo de frame)" que define la función del paquete. Además, se les asignó una dirección de 1 byte a cada módulo. S hace una descripción del protocolo y en la Tabla 1, una descripción de los frame types creados.

```
#define ADDRESS_MED     0x00    // Dirección asignada al Módulo de medición
#define ADDRESS_MAIN    0x01    // Dirección asignada al Módulo Main
#define ADDRESS_ESP     0x02    // Dirección asignada al Módulo IoT
```

A continuación, se describe el protocolo usado

[Byte Inicio][Dir_Destino][Cant_Bytes][Frame_Type][Payload][CheckSum]

- [Byte Inicio]: 0x02 usado como byte de inicio del protocolo
- [Byte Inicio]: Dirección del módulo que tiene que recibir el mensaje
- [Byte Inicio]: Cantidad de bytes del payload (length)
- [Byte Inicio]: Tipo de frame, describe la información útil tiene el frame
- [Byte Inicio]: Payload, es la información útil, definida por el frame type y con cantidad de bytes definida en el byte [Cant_Bytes]
- [Byte Inicio]: Checksum

En Tabla 2, se describen los frames usados

Nombre	**Char**	**Hexadecimal**	**Función**
RMS_Voltaje	'a'	0x61	Valor True RMS de voltaje
RMS_Corriente	'b'	0x62	Valor True RMS de corriente

RMS_Potencia	'c'	0x63	Valor aproximado de potencia
Frec_Voltaje	'd'	0x64	Frecuencia de la señal de voltaje
Max_Voltaje	'e'	0x65	Valor máximo de voltaje
Max_Corriente	'f'	0x66	Valor máximo de corriente
Coseno_fi	'g'	0x67	Coseno de fi

Tabla 2: *Frame types del protocolo de comunicación.*

En la Fig, N° 19, se muestra el armado de los frames en el debugger de la simulación.

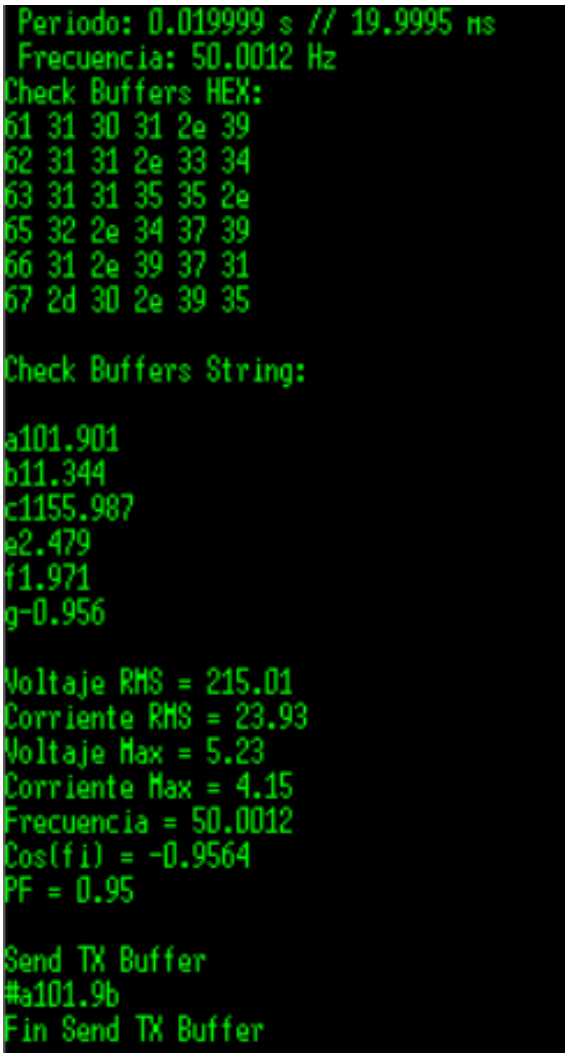


Fig. N° 19 *Armado de frames en debugger de comunicación.*

Conexiones y PCB del Módulo Main

La placa del *Módulo Mai*n tiene las siguientes conexiones:

- Bornera de alimentación de fuente externa
- Tira de pines de programación de PIC.
- Tira de pines de comunicación UART con *Modulo de Medición*
- Tiras de pines 2x16 para conexión de display LCD
- Bornera de alimentación de fuente externa para los relés
- *Borneras de salida de los relés*

Los PCB fueron diseñados en Eagle 7.6.0. En la Fig, N° 20 y Fig, N° 21, se muestran el top y bottom layer respectivamente. Las líneas punteadas marcan el contorno de los planos de masa definidos en ese layer.

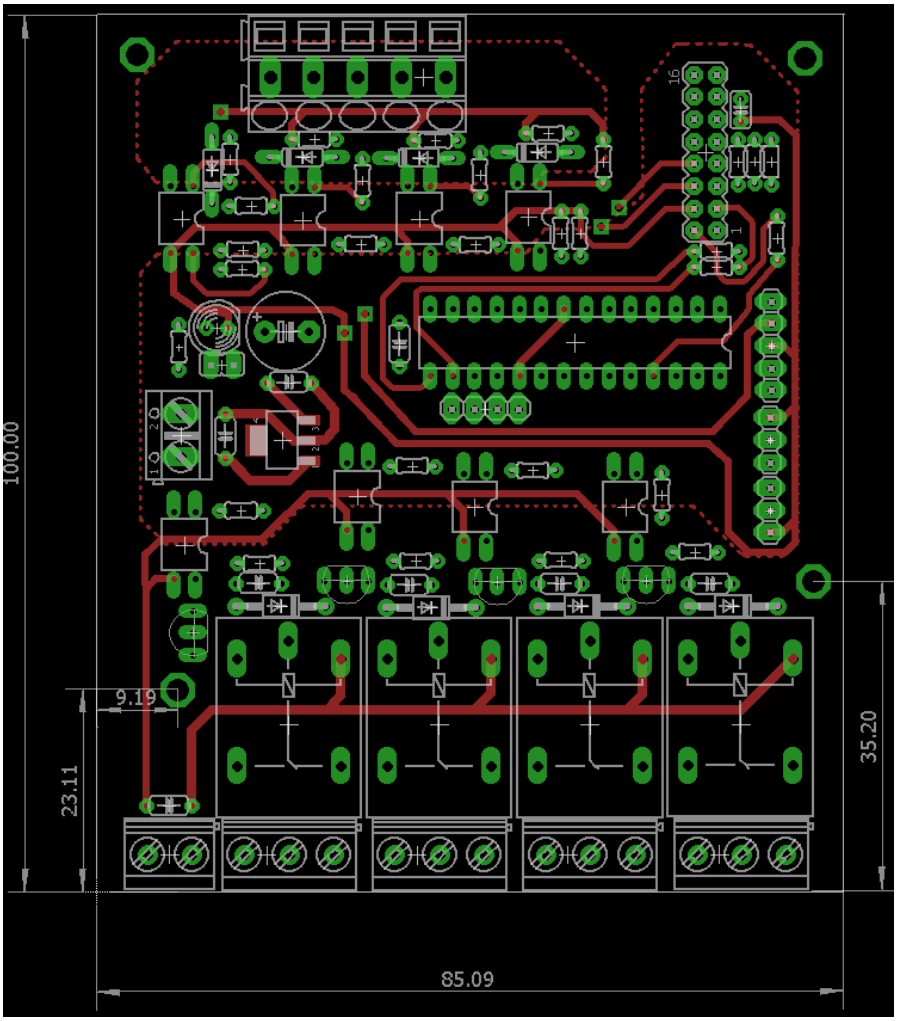


Fig. N° 20 *PCB Módulo Main – Top Layer.*

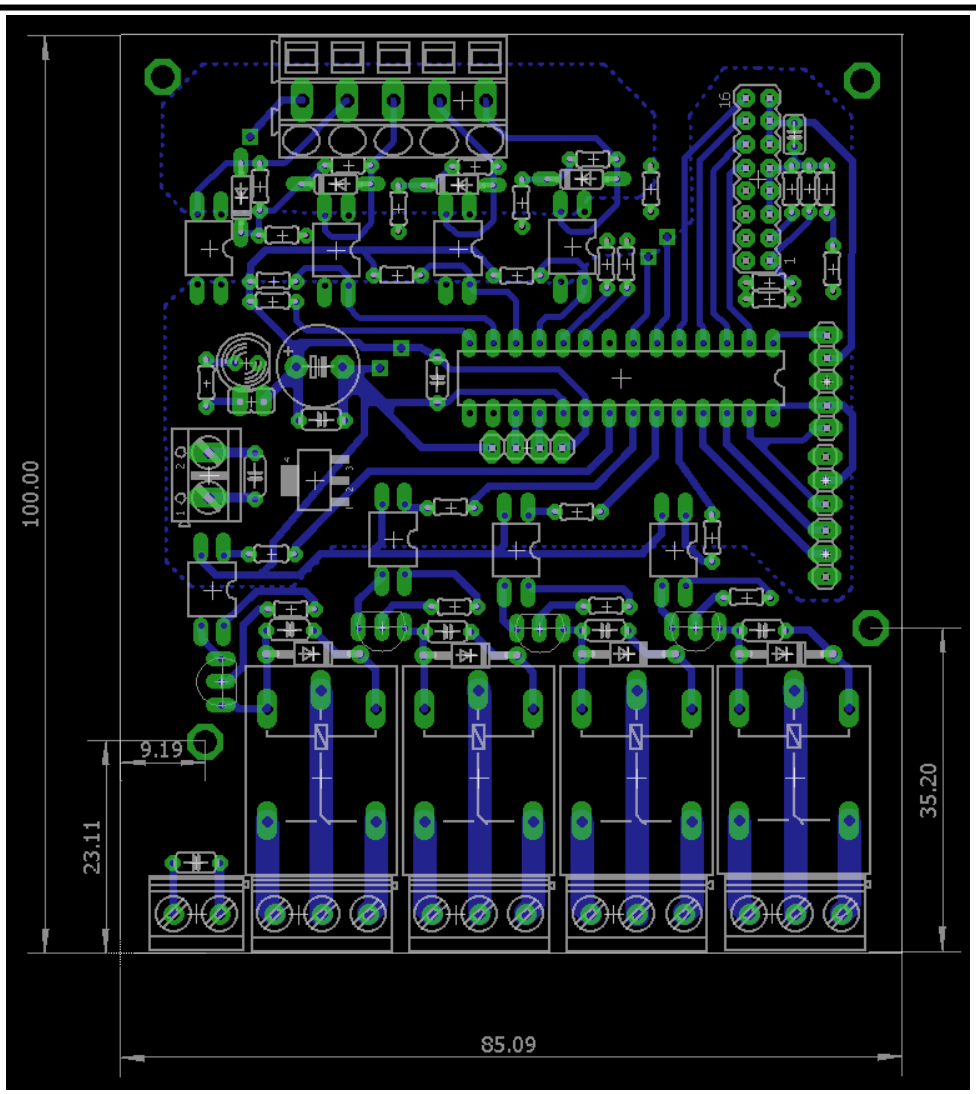


Fig. N° 21 *PCB Módulo Main – Bottom Layer*.

Módulo IoT

El firmware de este módulo se desarrolla con el IDE de Arduino en lenguaje C. Las librerías para IoT del ESP8266 estan desarrolladas por la comunidad, por lo que las funciones son casi de alto nivel. Por lo que se considera más beneficioso documentar el uso de los servicios IoT para esta sección.

A continuación, se listan las funciones de este módulo.

- Recibir datos desde el *Módulo Main*
- *Recibir datos por MQTT*
- Enviar al *Módulo Main* un valor de actualización para la alarma de potencia máxima
- Enviar paquetes HTTP con la información de las señales medidas

Uso de CludMQTT

Esta es una plataforma IoT que provee el servicio de bróker con plan gratuito con uso limitado de 10kb/ y la creación de hasta 2 brokers o instancias. Se va a usar para setear el valor de alarma de consumo máximo.

El primer paso es crear el bróker o instancia. A este se van a agregar “Usuarios” y los “Topics” que son los dispositivos que van a usar el bróker con los métodos de Publish/Subscribe. En la Fig, N° 23, se muestra un usuario creado, a que tópico está subscripto y permisos de read o write.

Type	Pattern	Read/Write	
topic	node1 - outTopic	true/true	Delete
topic	node1 - analogico	true/true	Delete
topic	node1 - lednode	true/true	Delete
topic	node1 - outTopic3	true/true	Delete
topic	node1 - outTopic2	true/true	Delete

Fig. N° 23 *Tabla de usarios y tópicos en CloudMQTT.*

Para el firmware se usa una librería genérica de Publish/Subscrite, donde se van a cargar las credenciales correspondientes de CloudMQTT y del bróker creado. En el código siguiente se muestra la librería y la carga de las credenciales para la conexión al bróker.

```
#include <ESP8266WiFi.h>

#include <PubSubClient.h>

// MQTT Broker

const char *mqtt_broker = "broker.emqx.io";

const char *topic = "esp8266/consumo";

const char *mqtt_username = "nodemcu1";

const char *mqtt_password = "public";
```

```
const int mqtt_port = 1883;
```

Una vez que la aplicación del ESP8266 se conecta al bróker, se utilizan las siguientes funciones para publicar o subscribirse a un tópico.

```
// publish and subscribe

client.publish(topic, "valor");

client.subscribe(topic);
```

Uso de ThingSpeak

Esta es una plataforma IoT con un plan gratuito que permite la comunicación entre dispositivos, sitios web y web services por medio de APIs sencillas. Una de las APIs permite hacer envíos de información del tipo GET al servidor de ThingSpeak. El servidor se encarga del almacenamiento de la información en una base de datos y permite visualizarla en graficas en función del tiempo que fue actualizado el dato.

Esta plataforma se va a encargar de almacenar y graficar los datos del consumo eléctrico.

El ESP8266 va a funcionar como un cliente que envía la información a la API por medio de una función de la librería de Arduino. Las librerías usadas son:

```
#include "ThingSpeak.h"
#include <ESP8266WiFi.h>
```

Luego de crear una cuenta en ThingSpeak, el primer paso es un crear un “Channel” o canal. Un canal permite recolectar información enviada desde un dispositivo, otro canal o una página web.

En el canal se crea un campo a usar, en este caso se creó “Consumo kWh”. En la Fig. N° 24, se puede ver la configuración básica del canal.

Una vez configurado el canal, se puede empezar a escribir y leer data con el uso de las APIs “Write API Key” y “Read API Key” respectivamente. En la Fig N° 25, se pueden ver las APIs en formato GET y la Key del canal.

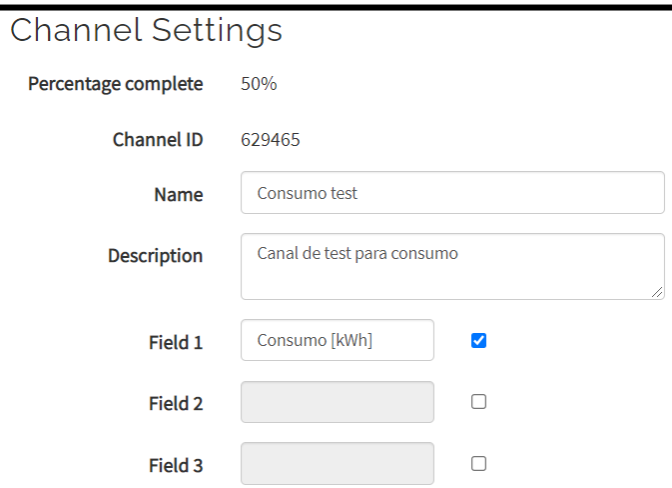


Fig. N° 24 *Configuracion de Channel en ThingSpeak.*

La Key del canal, es la que obtiene la información y los permisos para poder escribirlo con las APIs del método GET. Si la Key es errónea en el método, el dato no va a llegar al canal.

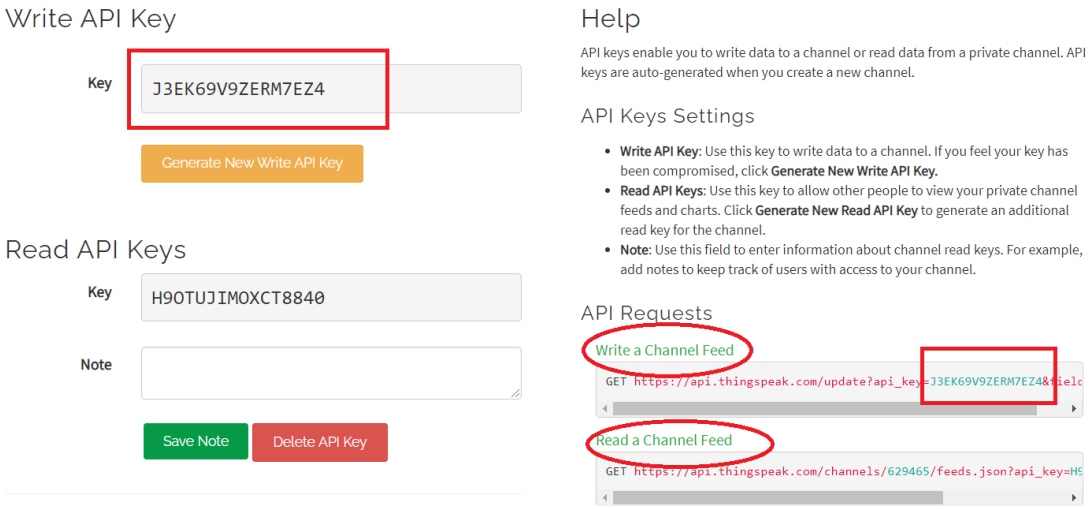


Fig. N° 25 *APIs y Keys de ThingSpeak.*

En el código del firmware se deben agregar estas Keys y el ID del canal creado para que la información pueda llegar al servicio desde el ESP8266.

```
unsigned long myChannelNumber = Mi_Channel_Id; // Agregar channel Id
const char * myWriteAPIKey = Mi_Key_Id;        // Agregar key del canal
```

Una vez que se tiene datos en el canal, se puede leer con una API de visualización. En la Fig. N° 26, se muestra un gráfico en funcion del tiempo de los valores de test que se enviaron al canal de “Consumo Test”. Además, ThingSpeak provee un bloque de código HTML para poder copiar directamente en la página web para tener la gráfica del canal si esta seteado como público. Con las siguientes líneas de código se puede escribir un valor:

```
ThingSpeak.setStatus(nuevo_valor_consumo);
(void)ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
```

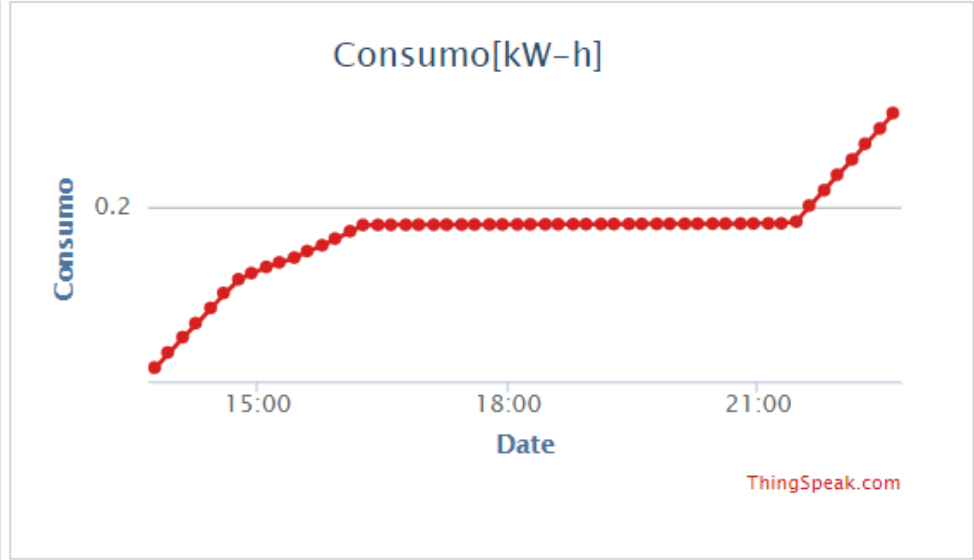


Fig. N° 26 *Grafica de consumo histórico en el tiempo.*

Evaluación Final del Sistema

Validación de algoritmos

Para la rápida validación de los algoritmos descriptos en la sección del *Módulo de Medición* se usó un software CAD de simulación. Este software permite agregar un microntrolador y su archivo .hex del firmware resultante de la compilación. En la Fig. 27, se puede ver el esquemático usado. Donde se incluyen al circuito un simulador de osciloscopio para ver las señales y serial terminal para conectarlo a una UART y poder ver los datos.

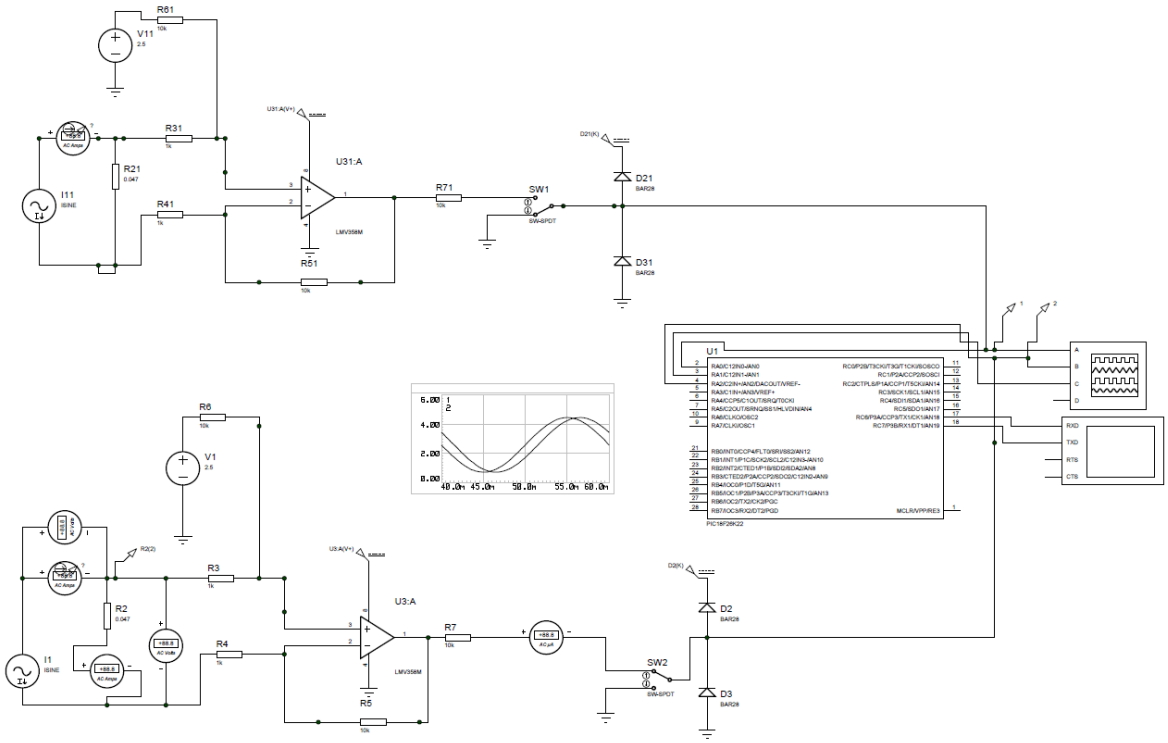


Fig. N° 27 *Simulación del Módulo de Medición.*

Además de la simulación, se hizo un debbug console por medio de una UART del microcontrolador para imprimir mensajes en el serial terminal virtual.

Se fueron dejando mensajes de debug en las partes del código que se querían testear para hacer una rápida depuración de bugs en los algoritmos. También se imprimieron los resultados de las mediciones. En la Fig. 28, se muestra un ejemplo de los mensajes de debug en el serial terminal virtual.

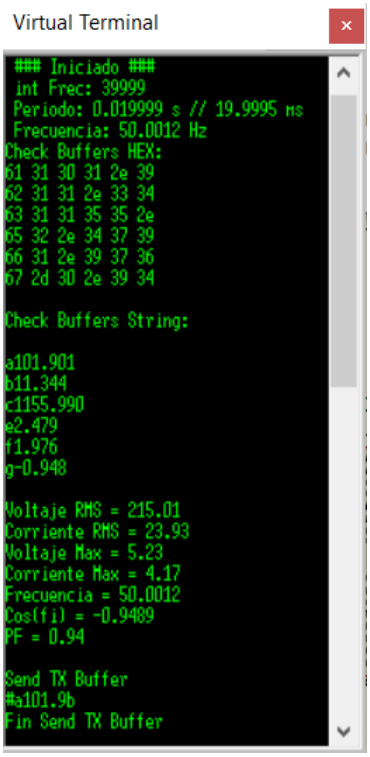


Fig. N° 28 *Debug en serial virtual terminal.*

Preparación de prototipo final

Prototipado

En una primera etapa se testeo solamente la parte de medición de tensión y corriente. Para validar las medidas tomadas por el *Módulo de Medición* y al comunicarlas al *Módulo Main* e imprimirlas en el display se podían ajustar parámetros en el algoritmo de True RMS. En la Fig 29, se muestra el montaje para el prototipado sin el módulo IoT cerrado y en la Fig. 30 sin la tapa.

En cuanto al módulo IoT, no se hizo hardware ya que el módulo NodeMCU es una placa de desarrollo con todo incluido. Por lo que se solo se puse en un protoboard y se conectaron los cables Rx, Tx y GND para la comunicación con la *placa Main*. En la Fig. 31, se muestra la conexión de este módulo.

En la Fig. 32, se muestra el sistema completo.

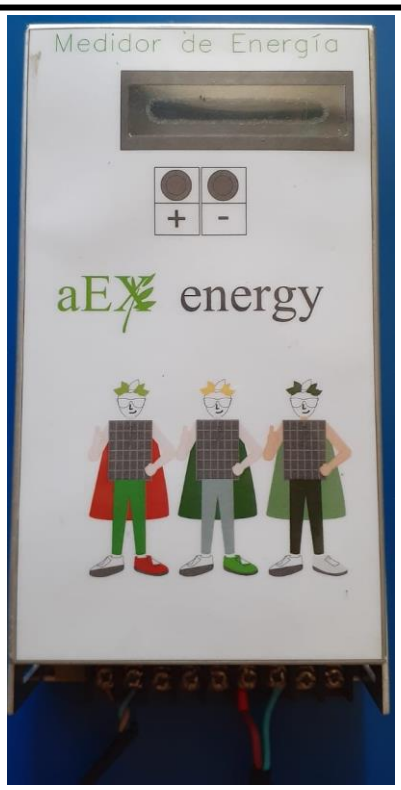


Fig. N° 29 *Prototipado sin módulo IoT con frente.*

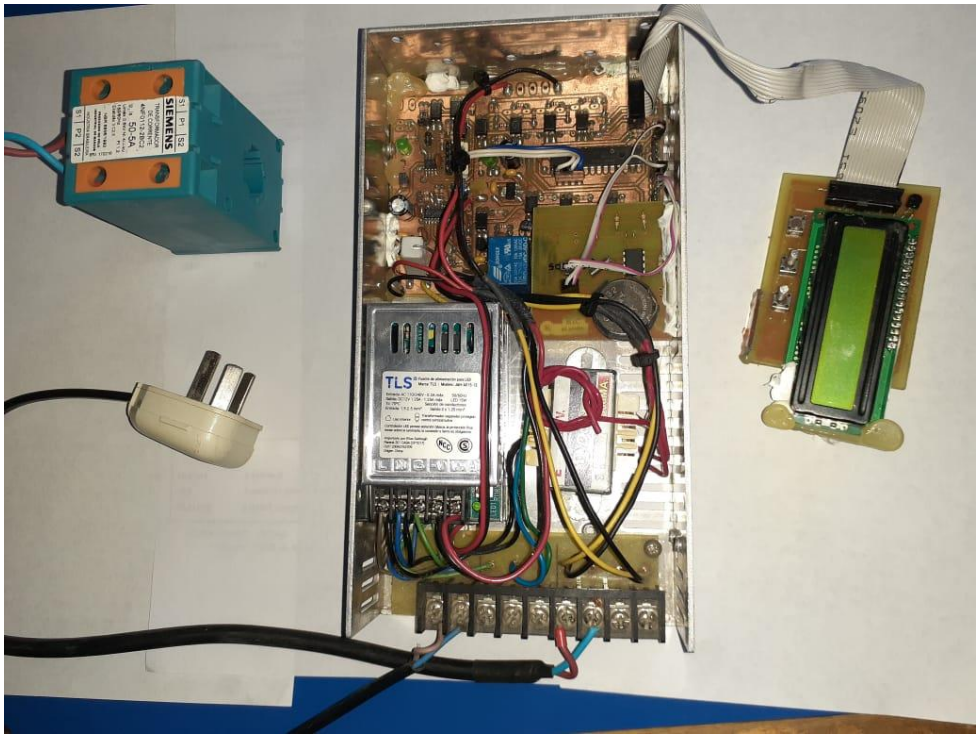


Fig. N° 30 *Prototipado sin módulo IoT sin el frente.*

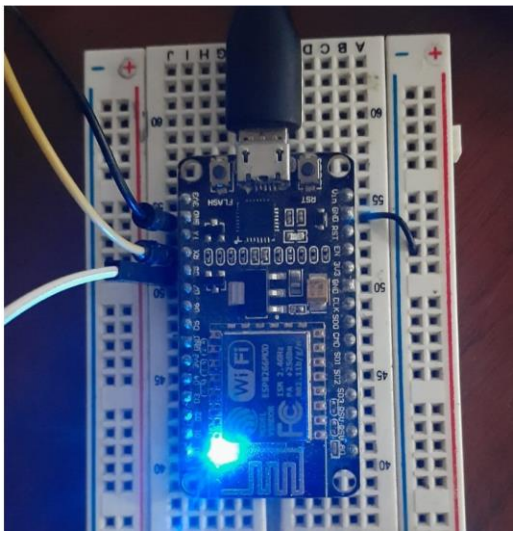


Fig. N° 31 *Prototipado módulo IoT.*

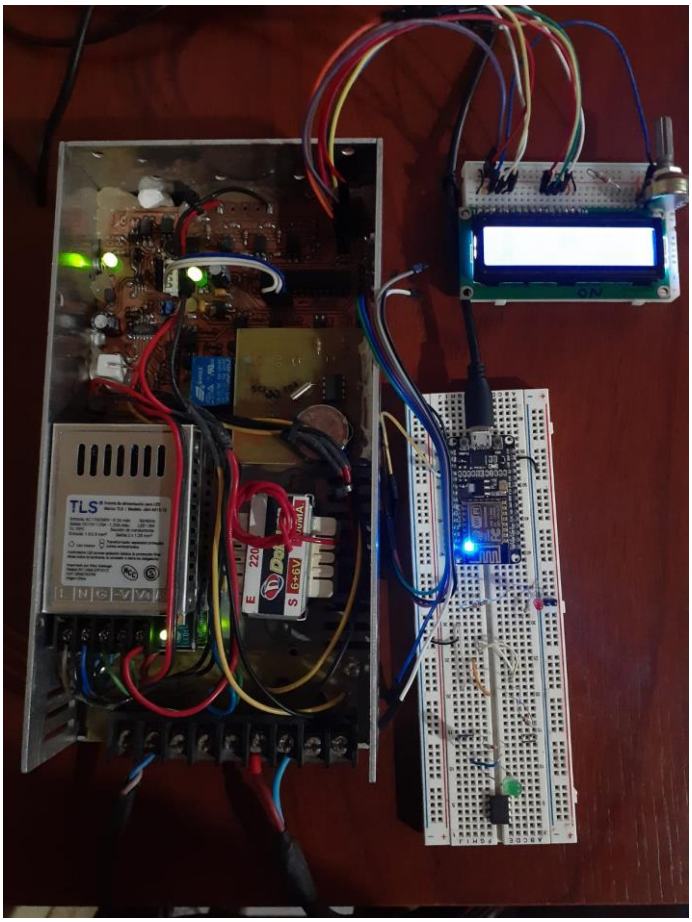


Fig. N° 32 *Prototipo completo.*

Pantallas del display LCD

En el display LCD se muestra información con un refresh de 1 segundos. La tensión y la corriente en la primera línea. La segunda línea va cambiando cada 10 segundos entre mostrar consumo y potencia instantánea. Estas dos pantallas se muestran en Fig. 33 y Fig. 34.

Fig. N° 33 *Display LCD con consumo.*

Fig. N° 34 *Display LCD con potencia instantanea.*

Interfaz web

Para mostrar todas las mediciones se desarrolló una interfaz web usando las tecnologías HTML, CSS y Bootstrap. Además de los scripts Javascript (js) necesarios para el MQTT. Se quitaron las partes desarrolladas en PHP para no tener que usar un servidor local o externo con un módulo Apache. Al resolver solamente con las tecnologías mencionadas y al estar usando servicios IoT con APIs, se puede correr la página directamente en el navegador Web sin ejecutar ningún servidor.
Se dividió en 3 secciones con un header fijo para ir switcheando de una a otra sin hacer un refresh de la página.

La primera sección es *"Valores Actuales"*, donde se muestran los valores de tensión, corriente y potencia actual, con un refresh de 10 segundos. Se dejó un refresh alto, ya que esta sección es de modo demostrativo, los valores actuales de las medidas se muestran en el display LCD del medidor. En la Fig. 34, se muestra la primera sección.

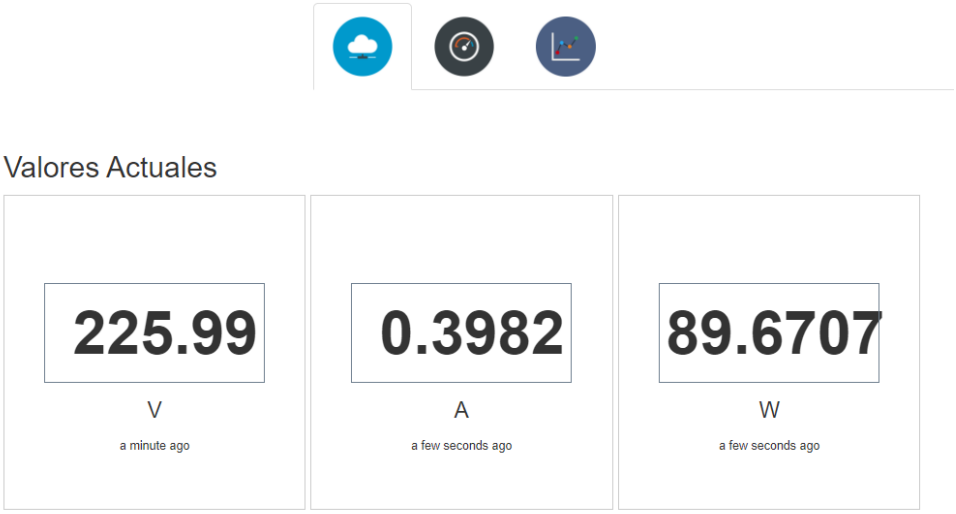


Fig. N° 34 *Vista valores actuales de página web.*

La segunda sección es *"Set Potencia Instantánea Máxima"*, en esta se tienen botones del 20 al 240, con saltos de 20, que representan el valor de trigger de potencia en Watts. En la Fig. 35, se muestra la segunda sección.

Fig. N° 35 *Vista Set Potencia Instantánea Máxima de página web.*

La tercera sección es *"Valores Históricos"*, en esta se muestra el último valor guardado de consumo en kW-h junto con una gráfica capaz de mostrar hasta 1000 puntos de valores guardados en el tiempo. En la Fig. 36, se muestra la tercera sección.

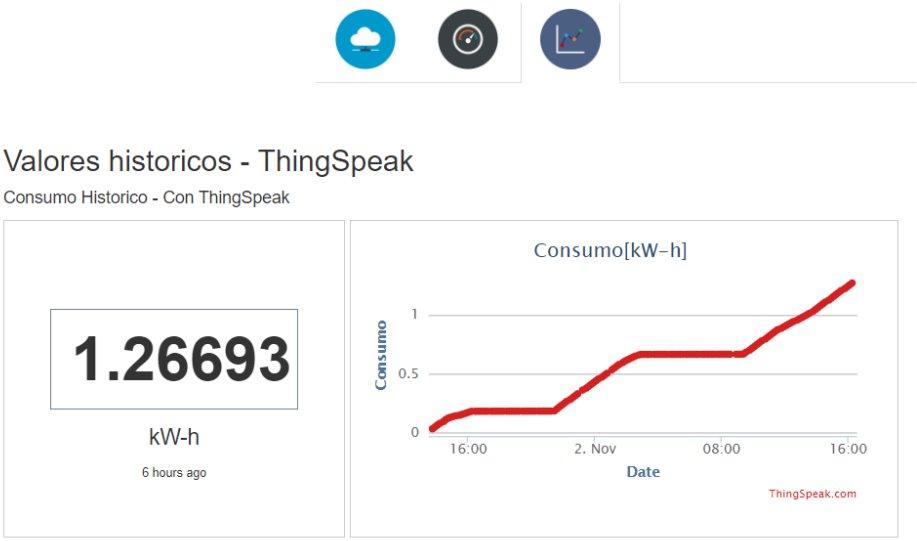


Fig. N° 37 *Vista Valores históricos de página web.*

Mediciones del sistema vs multímetro digital

Para validar las mediciones del *Medidor de consumo IoT* se utilizó un multímetro digital UNI-T *UT39A*. Hacer una comparativa de estas medidas sirvió para un ajuste fino de las constantes de ganancia. Se fue tomando nota de las mediciones con distintas cargas, para hacer un barrido aproximado entre 0 A y 10A. En la Tabla 3, se detallan las medidas tomadas por *Medidor IoT* y el multímetro mencionado. En la gráfica de la Fig. 37, se muestran las medidas en contraste una sobre otra de la tabla anterior.

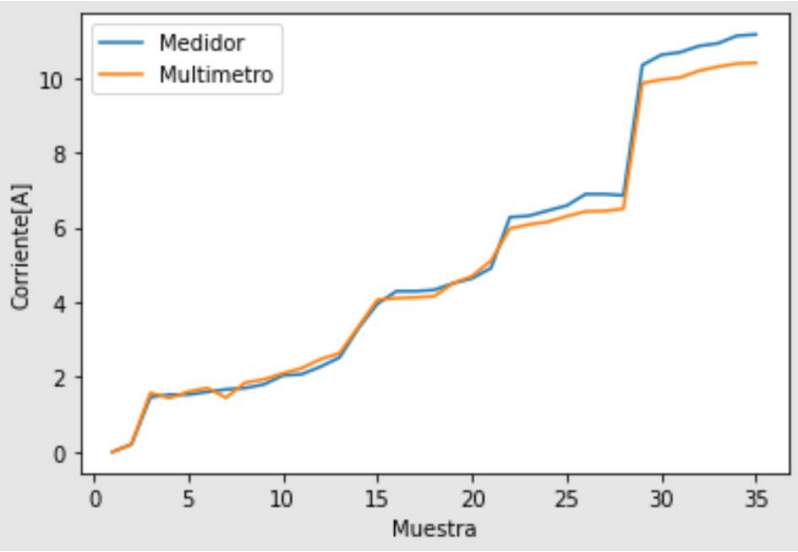


Fig. N° 37 *Grafica comparativa de corriente medida en el medidor IoT y un multímetro*

Muestra	Medidor IoT	Multimetro
1	0,0006	0,0
2	0,2036	0,21
3	1,4629	1,58
4	1,5329	1,44
5	1,5329	1,61
6	1,6029	1,71
7	1,6728	1,45
8	1,7078	1,85
9	1,8058	1,94
10	2,0437	2,10
11	2,0786	2,24
12	2,2815	2,48
13	2,5264	2,64
14	3,3100	3,35
15	3,9607	4,07
16	4,2966	4,11
17	4,2966	4,13
18	4,3315	4,16
19	4,5065	4,51
20	4,6394	4,70
21	4,9123	5,11
22	6,2766	5,97
23	6,3116	6,08
24	6,4515	6,15
25	6,5844	6,30
26	6,8923	6,43
27	6,8923	6,44
28	6,8573	6,50
29	10,3410	9,86
30	10,6145	9,95
31	10,6840	10,01
32	10,8524	10,19
33	10,9223	10,30
34	11,1252	10,38
35	11,1602	10,40

Tabla 3: *Muestras Medidor IoT y multimetro*

Conclusiones

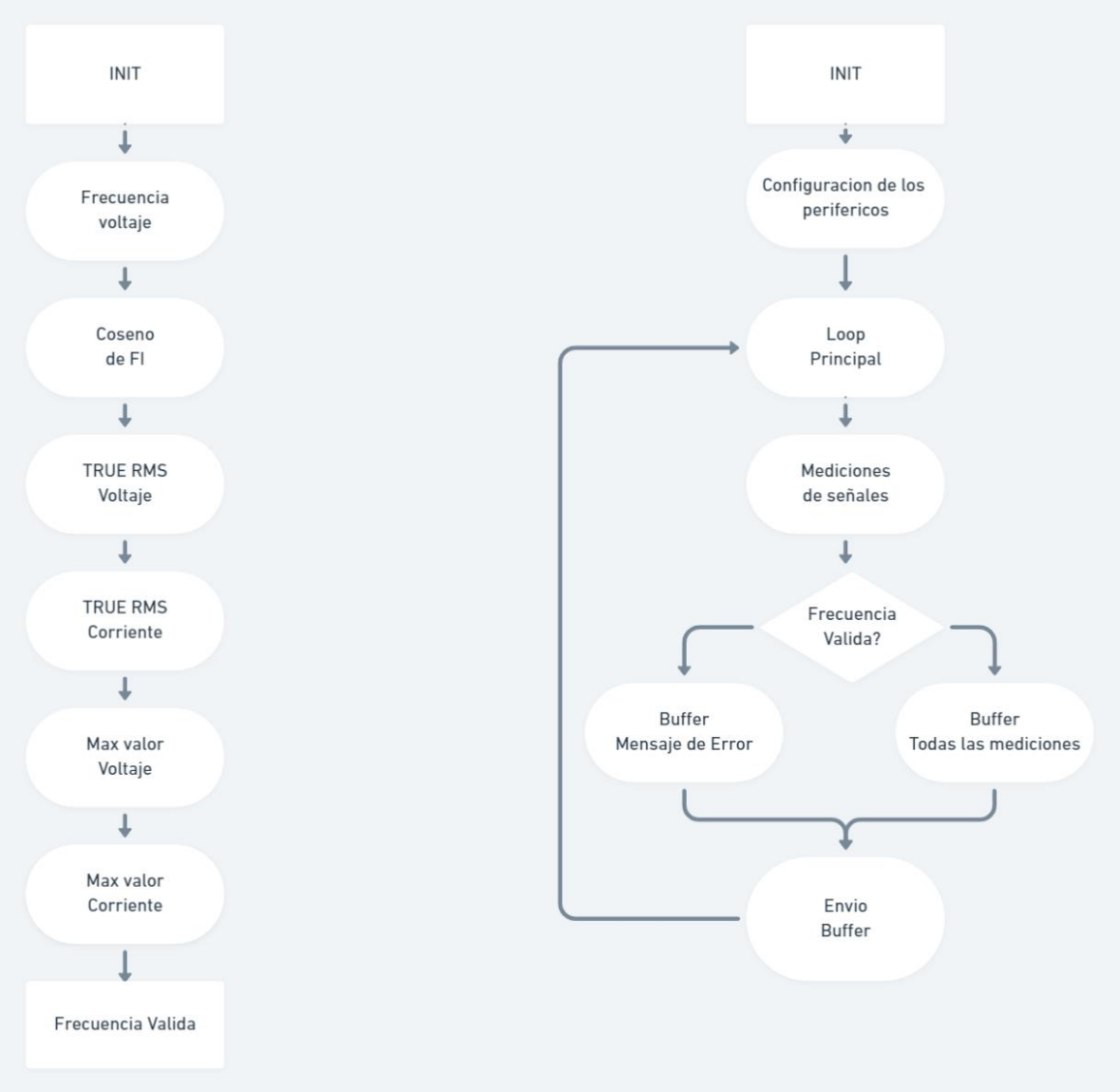
El scope, o alcance, del proyecto no se basó en la exactitud de las mediciones, ya que se conocen las limitaciones de hardware al utilizar productos y herramientas de bajo costo. Como principal limitante se tiene a los microcontroladores de Microchip. Los usados en este proyecto son de gama baja, con poca memoria RAM y ROM, por lo que se dificultó el desarrollo de firmware del *Módulo de medición*, ya que varias de las operaciones matemáticas que este módulo tiene que hacer requería mucha memoria. Existen microcontroladores de NXP con Cortex m0, del mismo costo y superiores en prestaciones con la tecnología utilizada. Pero el alto costo de las herramientas de desarrollo de NXP fue la razón por la que se descartó su uso.

Se obtiene como resultado que se puede dar conexión a internet a casi cualquier sensor o dispositivo con el agregado de agregado de un módulo Iot y un microcontrolador en caso de ser necesario. Incluso integrar el IoT a sistemas con comunicaciones offline cómo pueden los que usan RS-232, RS-485, Modbus, Zigbee 802.15.4, entre otros.

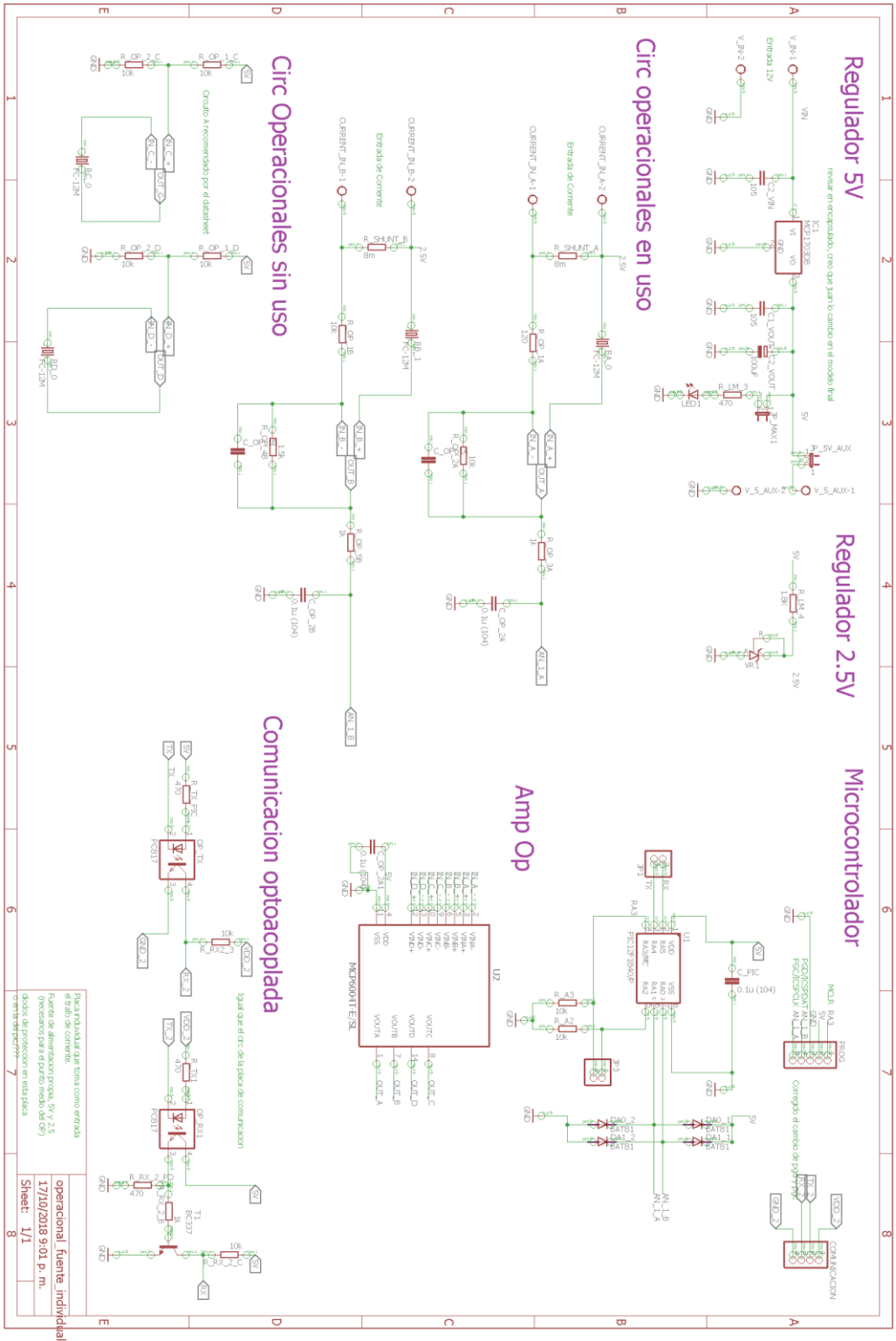
Elegir un hardware con comunidad detrás, como lo es el ESP8266, para el proyecto Arduino facilita el desarrollo ya que se llega a funciones casi de alto nivel y aplicación directa modificando punteros y directivas de precompilación.

El agregado de IoT a los sistemas permite un mejor análisis de la data obtenida, pudiendo aplicar técnicas de data science y machine learning a los datos almacenados.

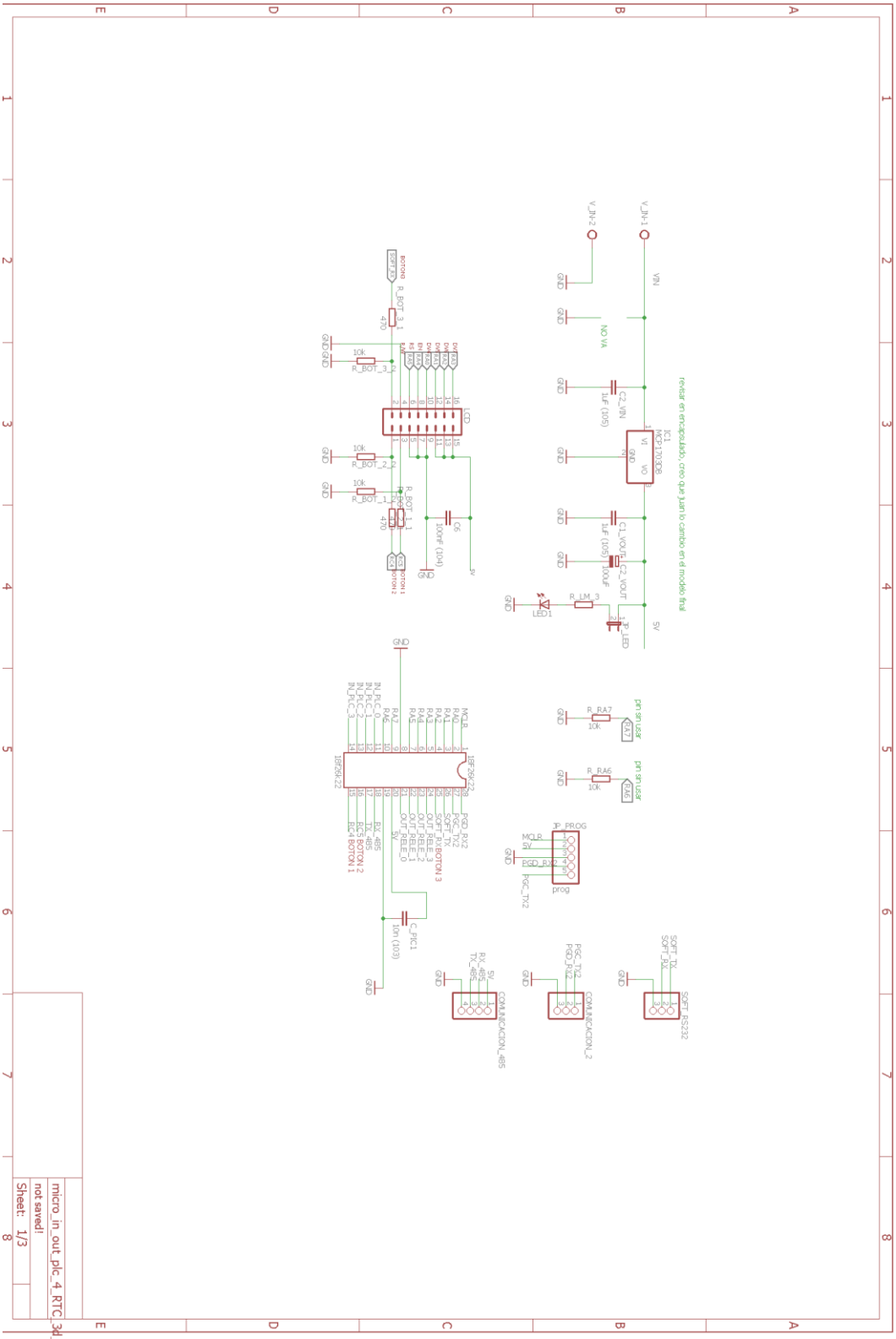
Anexo 1: Diagrama de flujo del modulo de medición

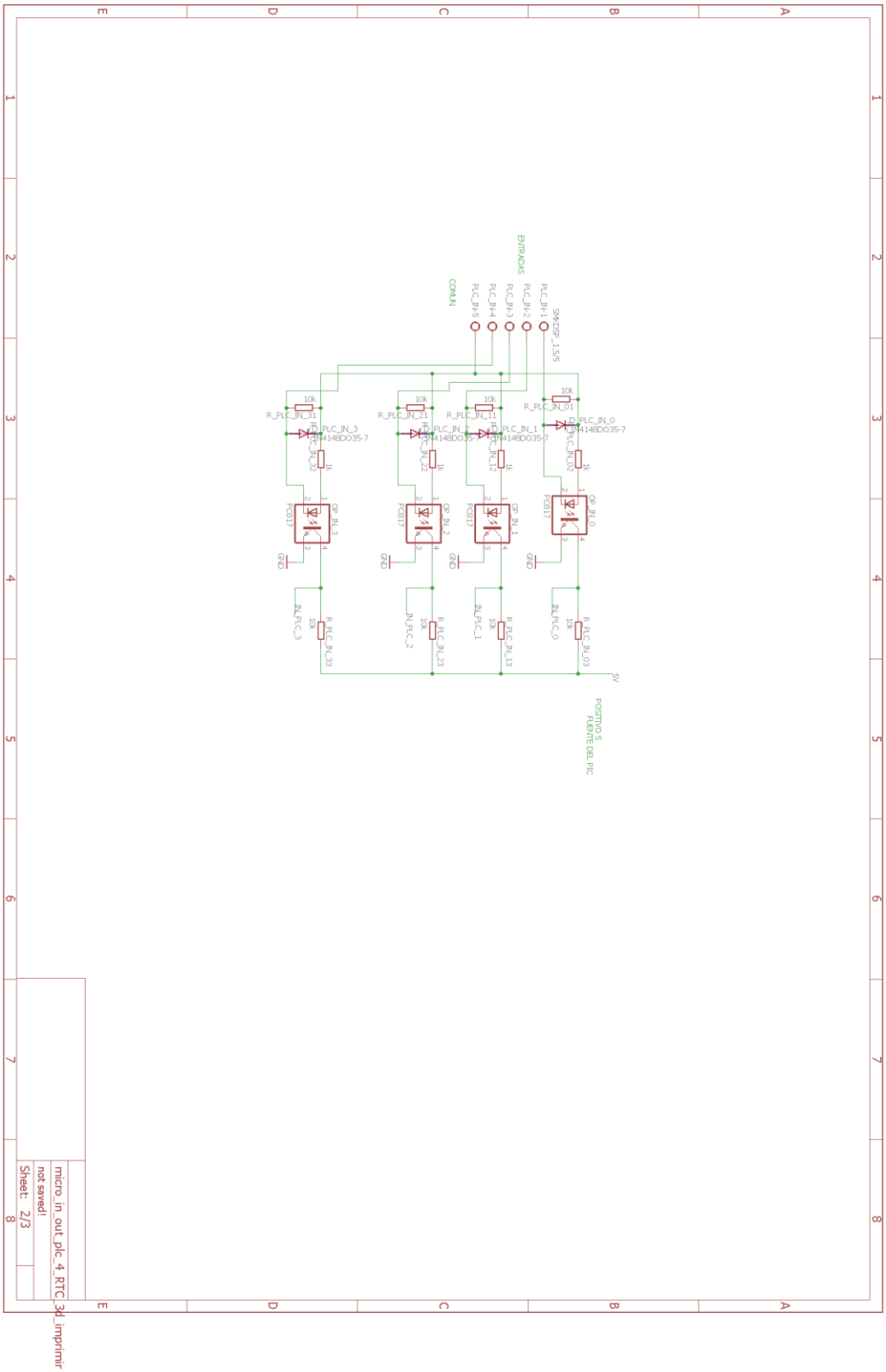


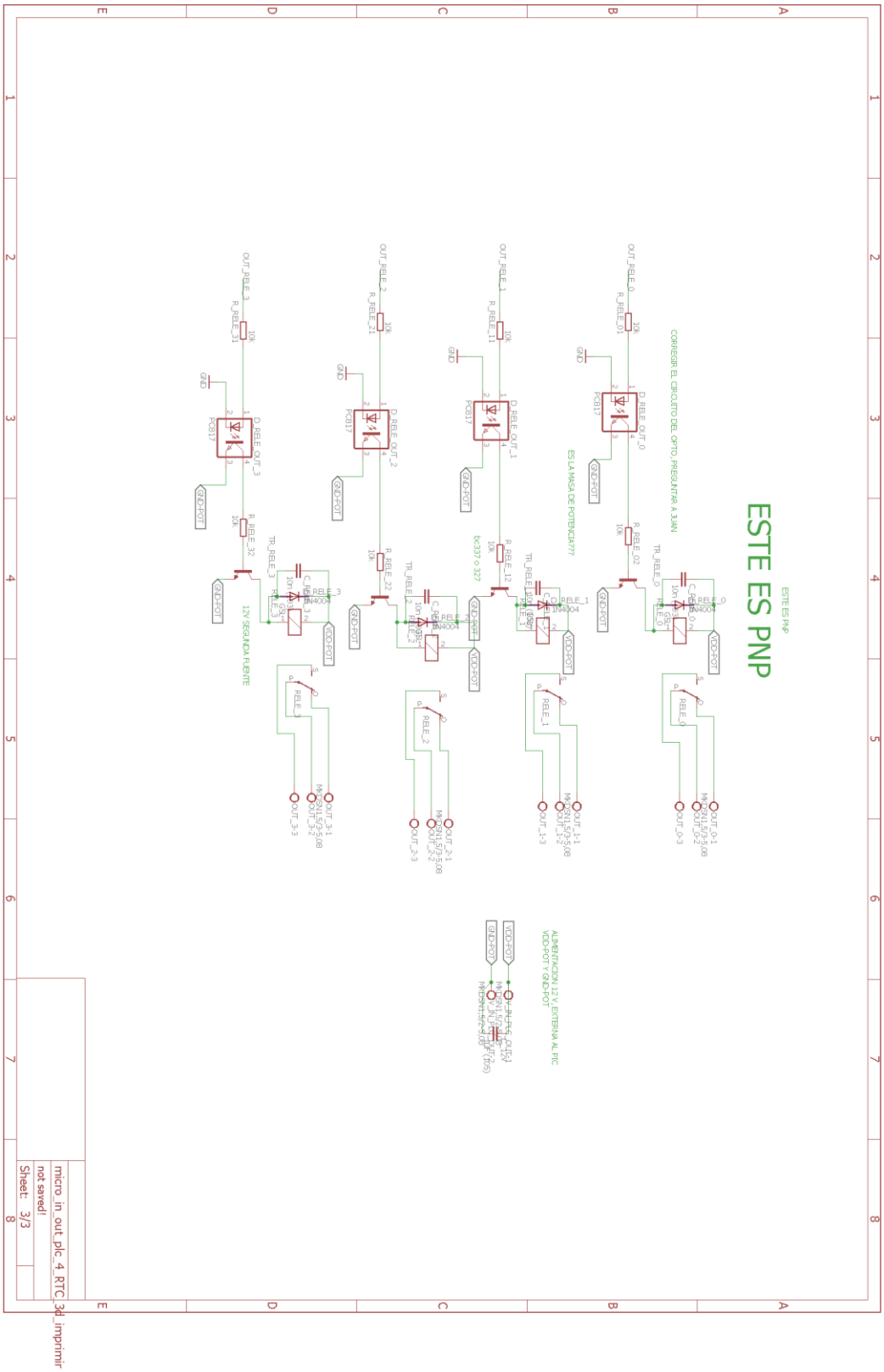
Anexo 2: Modulo de medición



Anexo 3: Modulo main




ENTRADAS
COMUN
PLC_IN+5
PLC_IN+4
PLC_IN+3
PLC_IN+2
PLC_IN+1
SM4DSP_1.5/5
10k
R_PLC_IN_31
R_PLC_IN_21
R_PLC_IN_11
R_PLC_IN_01
D_PLC_IN_3
D_PLC_IN_2
D_PLC_IN_1
D_PLC_IN_0
1N4148DO35-7
1k
PLC_IN_32
PLC_IN_22
PLC_IN_12
PLC_IN_02
PC817
OP_IN_3
OP_IN_2
OP_IN_1
OP_IN_0
GND
IN_PLC_3
IN_PLC_2
IN_PLC_1
IN_PLC_0
R_PLC_IN_33
R_PLC_IN_23
R_PLC_IN_13
R_PLC_IN_03
5V
POSITIVO 5
FILEATE DEL PIC
micro_in_out_plc_4_RTC_34_imprimir
not saved!
Sheet: 2/3


ESTE ES PNP
CORREGIR EL CIRCUITO DEL OPTO, PREGUNTAR A JUAN
ES LA MASA DE POTENCIA???
12V SEGUNDA FUENTE
ALIMENTACION 12 V, EXTERNA AL PIC
VDD-POT Y GND-POT
micro_in_out_plc_4_RTC_3d_imprimir
not saved!
Sheet: 3/3

Anexo 4: Código del Módulo de Medición

```
/*

// Modulo de medicion

*/


#include "main.h"

// #include <float.h>

#include <math.h>

// #include <string.h>


//DEFINE CONSTANTES

#DEFINE  Lim_Cero_Flotante 511

/// Pruebas con 250 y 85 (minimo) Buenos Resultados

#define muestras 250        /// USAR MENOS DE 250 muestras, sino hay que cambiar
el contador a una variable de 16 BITS!!

#define promedio_n   10    /// Cantidad de veces que se toman la cantidad de
muestras para hacer un promedio

#define KV 0.00488         /// KV = Cte conversion a Voltaje KV=5.00/1023 =
0.0048875

                           /// Probar si hace bien el calculo poniendo la
contante con 4 decimales,

#define BYTE_INICIO     2    // Byte delimitador de inicio

#define ADDRESS_WIFI     0x01    // Dirección asignada al nodo receptor

#define ADDRESS_WIFI2    0x04    // Dirección asignada al nodo receptor

#define TAM_TRAMA       8       // Cantidad de bytes en una trama


// Seniales conectadas a los canales de ADC

enum signals{ch_voltaje = 0, ch_corriente = 1};

enum buffers{buf_voltaje = 0, buf_corriente = 1, buf_potencia = 2, buf_frec = 3,
buf_v_vmax = 4, buf_i_max = 5, buf_cos_fi = 6};


// Buffers para enviar los datos

static char buffer_rms_v[10];
```

```
static char buffer_rms_i[10];
static char buffer_rms_p[10];
static char buffer_frec[10];
static char buffer_max_v[10];
static char buffer_max_i[10];
static char buffer_cos_f[10];

// Variables globales
//!int buffer2[10];
//!int buffer3[10];
int16 retardo = 0;

// prototipos de funciones
// void serialOut(int DatoTx);
// void Transmitir(void);
// void Transmitir2(void);
float promedio_adc(int1 canal);
float valor_max_adc(int1 canal);
int8 cos_fi(void);
void show_time_diff_signals(void);
float frecuencia(void);
void config_hardware(void);
void reset_buffers(void);
void reset_buffer(char *reset_buff);
void check_buffers(void);
void check_buffer (char * check_buff);
void send_tx_buffer(char *selected_buff);
void serialOut(int DatoTx);

// fin prototipos
```

```
void main()
{

// VARIABLES LOCALES
   unsigned int16 count = 0;
   unsigned int8 valid_cos_fi = 0;
   unsigned int8 valid_frec = 0;

   float coseno;
   float fi;

   float rms_voltaje, rms_corriente;
   float max_voltaje, max_corriente;
   float frec_v;



//INICIO
   config_hardware();
   reset_buffers();



   printf("\r\n ### Iniciado ###");

   while(TRUE)
   {
      restart_wdt();

      if (count/* >= 10500 */)         // Consulta y resultados aporx cada 2
segundos
         {
//!         output_toggle(PIN_A2);
```

```
//Consulta funciones


         frec_v = frecuencia();
         if ( frec_v == 0 ){
            fprintf(WIFI,"\r\nError Frec");
            valid_frec = 0;
         }
         else {
            valid_frec = 1;
         }


         valid_cos_fi = cos_fi();
         if ( valid_cos_fi == 0){
            fprintf(WIFI,"\r\nError Cos fi");
         }
//!         show_time_diff_signals();
         fi = (float) retardo * 0.000157;             // 0.000157 es el
resultado de la conversion del conteo del timer 1 a tiempo y multiplicado por
2*pi*f
         coseno = cos(fi);


         rms_voltaje = promedio_adc(ch_voltaje);
         rms_corriente = promedio_adc(ch_corriente);
         rms_corriente +=10.00;
         rms_voltaje +=100.0;


         max_voltaje = valor_max_adc(ch_voltaje);
         max_corriente = valor_max_adc(ch_corriente);    // Falta multiplicar
por la constante de ganancia


         fprintf(WIFI,"\r\nCheck Buffers HEX:\r\n");
         sprintf(buffer_rms_v, "a%2.3f",rms_voltaje);      // String
         sprintf(buffer_rms_i, "b%2.3f",rms_corriente);      // String
```

```
        sprintf(buffer_rms_p, "c%2.3f",rms_voltaje*rms_corriente);      //
String

        if(valid_frec) sprintf(buffer_frec,  "d%2.3f",frec_v);      // String

        else{

           sprintf(buffer_cos_f, "DError-");      // String Error

        }

        sprintf(buffer_max_v, "e%2.3f",max_voltaje);      // String

        sprintf(buffer_max_i, "f%2.3f",max_corriente);      // String

        if(valid_cos_fi) sprintf(buffer_cos_f, "g%2.3f",coseno);      // String

        else {

           sprintf(buffer_cos_f, "gError-");      // String Error

        }

        check_buffers();


        fprintf(WIFI,"\r\nCheck Buffers String:\r\n");

        fprintf(WIFI,"\r\n%s",buffer_rms_v);

        fprintf(WIFI,"\r\n%s",buffer_rms_i);

        fprintf(WIFI,"\r\n%s",buffer_rms_p);

        fprintf(WIFI,"\r\n%s",buffer_max_v);

        fprintf(WIFI,"\r\n%s",buffer_max_i);

        fprintf(WIFI,"\r\n%s\r\n",buffer_cos_f);


//Resultados y muestra

        fprintf(WIFI,"\r\nVoltaje RMS = %f",rms_voltaje*2.11);         //
Multiplico por la constante de ganancia

        fprintf(WIFI,"\r\nCorriente RMS = %f",rms_corriente*2.11);

        fprintf(WIFI,"\r\nVoltaje Max = %f",max_voltaje*2.11);         //
Multiplico por la constante de ganancia

        fprintf(WIFI,"\r\nCorriente Max = %f",max_corriente*2.11);

        fprintf(WIFI,"\r\nFrecuencia = %2.4f",frec_v);

        fprintf(WIFI,"\r\nCos(fi) = %2.4f",coseno);

        fprintf(WIFI,"\r\nPF = %2.2f",abs(coseno));      // valor absoluto y
redonde de 2 decimales - Usar funcion Ceil ? o mucha rom?
```

```
         fprintf(WIFI,"\r\n");


         fprintf(WIFI,"\r\nSend TX Buffer\r\n");
         send_tx_buffer(buffer_rms_v);
         fprintf(WIFI,"\r\nFin Send TX Buffer\r\n");


//!         show_time_diff_signals();


//!         else {
//!            fprintf(WIFI,"\r\nERROR Factor de Potencia");
//!         }// fin consulta cos_fi
//!         count = 0;
      }
      else {
         count++;
      }
   }//fin loop
}// fin main




//!//**************************************************************************
****
//!//*                     config_hardware
//!//*
//!//* Descripción   :     Configuracion de hardware. Mapeo, adc wdt y timer 1.
//!//* Argumentos    :
//!//* Retorna       :     Nada, usa la variable global "retardo" para devolver
el conteo del timer
//!//* Notas         :
//!//**************************************************************************
*****
void config_hardware(void){
```

```
   PMD0 = 0xFF;

   PMD1 = 0xFF;

   PMD2 = 0xFF;

   TMR1MD = 0;

   UART2MD = 0;

   UART1MD = 0;

   setup_adc_ports(NO_ANALOGS);

//adc

   TRISA = 0b00000011;   /// 0 y 1 como entradas

   setup_adc_ports(sAN0|sAN1, VSS_VDD);

   setup_adc(ADC_CLOCK_DIV_64|ADC_TAD_MUL_16);    // TESTEAR SI FUNCIONA MEJOR

//timer

   setup_timer_1(T1_INTERNAL|T1_DIV_BY_4);      //32.7 ms overflow   // Tmax = (
65536 * PS ) * Tinst

                                                 // Ps, prescaler     // Tinst =
4 / fosc

                                                 // Tiempo = t1 * 4 * (Tinst)

   setup_timer_1(T1_DISABLED);

//general

//!   setup_wdt(WDT_16);      //~4.0 s reset

   return;

}



//!//*****************************************************************************
****

//!//*                     promedio adc

//!//*

//!//* Descripción   :     Devuelve el valor promedio del canal seleccionado por
medio del

//                        calculo de la integral de valor discreto. RMS = Sqrt{
[sumatoria(ADCn^2)] / [n] }     // n - numero de muestras // ADCn - enésima
muestra del adc
```

```
//!//* Argumentos    :     int1 canal -- Selecciona el canal a convertir ( 0 o 1
)

//!//* Retorna       :     float promedio_rms  -- Devuelve el promedio del canal
elegido

//!//* Notas         :     la variable "muestras" es un define global. Las
pruebas se hicieron con 85 (tiempo = 22.5 ms) y con 300 (65.7 ms, 3 periodos de
señal)

//                         Si se van a usar menos de 256 muestras, puedo
declarar el contador "i" como varible sin signo de 8 bits.

//!//*************************************************************************
*****

float promedio_adc(int1 canal)

{

//!      unsigned int8 i;               // Contador de for

      unsigned int16 valor;           // lectura del adc

      float adc;                      // conversion a voltaje

      float resultado_rms;            // acumulador de la sumatoria de los
valores cuadraticos

      float promedio_rms;             // promedio rms ( raiz cuadrada del
promedio de la acumulacion)



      promedio_rms = 0;

      for (int8 j=0; j<=(promedio_n-1); j++)        // cantidad del promedio

         {

         resultado_rms=0;

//!         output_high(PIN_A2); //// PARA MEDIR TIEMPO EN ALTO    /// Tiempo
final =  263us !

         for (unsigned int8 i=0; i<= muestras-1 ; i++ )      // toma de muestras
y sumatoria de la cantidad de muestras seteadas

         {

            set_adc_channel(canal);        // Comienzo de la aquisicion, 20us de
muestra

            delay_us(20);

            valor = read_adc();
```

```
            adc=(float)(valor*KV)-2.5;                //
adc=(float)(read_adc()*KV)-2.50;

            resultado_rms=(adc*adc)+resultado_rms;    // sumatoria de los
valores al cuadrado

         }

//!     output_low(PIN_A2);      //// MEDIR TIEMPO EN ALTO

         promedio_rms = promedio_rms + resultado_rms; // sumatoria de promedios

         }

      promedio_rms =  promedio_rms/promedio_n;                // promedio total

      promedio_rms=promedio_rms/(muestras);           // promedio rms

      promedio_rms=sqrt(promedio_rms);                // promedio rms


      return promedio_rms;                            // retorno del promedio
rms del canal seleccionado

}
```

```
//!//*********************************************************************
****

//!//*                     valor_max_adc

//!//*

//!//* Descripción   :     Muestreo de la senial, se guarda el valor maximo. La
cantidad de muestras tiene que hacer que

//!//*                     se llegue a por lo menos un periodo de la senial.
20ms

//!//* Argumentos    :     int1 canal -- Selecciona el canal a convertir ( 0 o 1
)

//!//* Retorna       :     float valor_max  -- Devuelve un float del valor
maximo de voltaje. Este valor tiene que se multiplicado por la Ganancia

//!//* Notas         :     la variable "muestras" es un define global. Las
pruebas se hicieron con 85 (tiempo = 22.5 ms) y con 300 (65.7 ms, 3 periodos de
señal)

//                        Si se van a usar menos de 256 muestras, puedo
declarar el contador "i" como varible sin signo de 8 bits.
```

```
//!//*************************************************************************
*****
float valor_max_adc(int1 canal){

   float valor_max = 0;
   unsigned int16 valor;
   unsigned int16 valor_m;
   valor_m = 0;            // lectura del adc
   valor = 0;            // lectura del adc

   for ( unsigned int8 i=0; i<= 254 ; i++ )      // toma de muestras y sumatoria
de la cantidad de muestras seteadas
   {
      set_adc_channel(canal);
      delay_us(22);
      valor = read_adc();
      if ( valor > valor_m ){
         valor_m = valor;
      }
   }

   valor_max = (float)(valor_m*KV) - 2.479;
   if ( valor_max < 0.01 ) valor_max = 0.00;

//!   fprintf(WIFI,"\r\nVmax signal= %2.2f",valor_max);

   return valor_max;

}

//!//*************************************************************************
****
```

```
//!//*                     cos_fi

//!//*

//!//* Descripción   :     Por ahora calculo el tiempo de retardo de una señal
con la otra despues del paso por cero.

//!//*                     En los pasos se explica el proceso

//!//* Argumentos    :

//!//* Retorna       :     Nada, usa la variable global "retardo" para devolver
el conteo del timer

//!//* Notas         :

//!//**************************************************************************
*****

int8 cos_fi(void){

// tomo como valor medio de 1023 valores al 511, despues parar ser mas exacto
con el cero, hay que medirlo

// ya que no es en 2.5V clavados

// Paso 1: Activar el timer con su respectuvo PS y setear el Chanel 0 del adc

// Paso 2: Realizar lecturas y esperar que la señal sea negativa (menor a 2.5V o
sea, 511 de lectura adc)

// Paso 3: Esperar a que la señal sea igual o mayor que 511 (paso por Cero
analogico)

// Paso 4: Setear el timer en 0 y cambiar el analogico al Chanel 1.

// Paso 5: Verifico si estoy atras o adelante de la onda de tensión, Para
definir si es capacitiva o inductiva

// Paso 6:

//          a - La señal es menor que el limite, por lo que esta atrasada.

//             Espero a que sea mayor y tomo el valor del timer (hago
correccion)

//          b - La señal es maroy que el lim, por lo que esta adelantada

//             Espero a que sea menor, le sumo 10ms (mitad del periodo de los 50
Hz) y le aplico la correcion al valor del timer


// Midiendo la tensión con respecto a la corriente. El primer canal tiene que
ser la medida de la entrada de tension.


   unsigned int16 lectura_adc;

   unsigned int1 f = 0;
```

```
   //Agregado de escape
   unsigned int16 escape=0;


// Paso 1
   setup_timer_1(T1_INTERNAL|T1_DIV_BY_4);
   set_adc_channel(0);
   delay_us(20);
   f = 0;
// Paso 2
   do{    // aprox 47us cada lectura (47-22 = 25 del resto)
      ADCON = 1;
      delay_us(23);
      ADCDone = 1;
//!      fprintf(WIFI,"\r\nZ");
      while ( ADCDone == 1 ); // wait till GODONE bit is zero
      lectura_adc = read_adc(ADC_READ_ONLY);
      if ( lectura_adc < Lim_Cero_Flotante ) f = 1;
      } while (!f);
// Paso 3
   f = 0;
   do{
      ADCON = 1;
      delay_us(23);
      escape++;
      ADCDone = 1;
//!      fprintf(WIFI,"\r\nY");
      while ( ADCDone == 1 );
      lectura_adc = read_adc(ADC_READ_ONLY);
      if ( lectura_adc > Lim_Cero_Flotante ) f = 1;
      else if ( escape >= 0x0100 ){
//!         fprintf(WIFI,"\r\nYY");
```

```
         return 0;
      }
      } while (!f);
// Paso 4
   f = 0;
   set_timer1(0);
   set_adc_channel(1);
   delay_us(20);
// Paso 5
   ADCON = 1;
   delay_us(23);
   ADCDone = 1;
//!   fprintf(WIFI,"\r\nX");
   while ( ADCDone == 1 );
   lectura_adc = read_adc(ADC_READ_ONLY);
   if ( lectura_adc < Lim_Cero_Flotante ) f = 1;
// Paso 6a
   if ( f )
      {
         f = 0;
         do {
            ADCON = 1;
            delay_us(23);
            ADCDone = 1;
//!            fprintf(WIFI,"\r\nW");
            escape++;
            while ( ADCDone == 1 );
            lectura_adc = read_adc(ADC_READ_ONLY);
            if ( lectura_adc > Lim_Cero_Flotante ) f = 1;
            else if (escape>0x0100){
//!               fprintf(WIFI,"\r\nEscape");
```

```
               return 0;
            }
            } while( !f );

            retardo = get_timer1();
            if (retardo > 48 ) retardo = retardo - 48;      // Correccion e; ( -
48 )
//!            fprintf(WIFI,"\r\nInductivo");
            return 1;
      }
// Paso 6b
   else{
         f = 0;
         do {
            ADCON = 1;
            delay_us(23);
            ADCDone = 1;
            while ( ADCDone == 1 );
            lectura_adc = read_adc(ADC_READ_ONLY);
            if ( lectura_adc < Lim_Cero_Flotante ) f = 1;
            } while( !f );

            retardo = get_timer1();
            retardo = retardo + 19480;                // correcion de retardo =
retardo + 20000;
//!            fprintf(WIFI,"\r\nCapacitivo");
            return 1;
         }

}

//!//*************************************************************************
****
```

```
//!//*                     frecuencia
//!//*
//!//* Descripción   :
//!//* Argumentos    :
//!//* Retorna       :
//!//* Notas         :
//!//*************************************************************************
*****
float frecuencia(void){

   unsigned int16 lectura_adc;
   unsigned int1 f = 0;
   unsigned int16 frec_count = 0;
   float frec;
   unsigned int16 escape = 0;

// Paso 1
   setup_timer_1(T1_INTERNAL|T1_DIV_BY_4);
   set_adc_channel(0);
   delay_us(20);
   f = 0;
//!   fprintf(WIFI,"\r\nJ");
// Paso 2
   do{    // aprox 47us cada lectura (47-22 = 25 del resto)
      ADCON = 1;
      delay_us(23);
      escape++;
      ADCDone = 1;
//!      fprintf(WIFI,"\r\nK");
      while ( ADCDone == 1 ); // wait till GODONE bit is zero
      lectura_adc = read_adc(ADC_READ_ONLY);
      if ( lectura_adc < Lim_Cero_Flotante ){
```

```
         f = 1;
      }
      else if ( escape >= 0x0100 ){
         frec = 0;
//!         fprintf(WIFI,"\r\nL");
         return 0;
      }
      } while (!f);
// Paso
   f = 0;
   escape = 0;
   do{
      ADCON = 1;
      delay_us(23);
      escape++;
//!      fprintf(WIFI,"\r\nM");
      ADCDone = 1;
      while ( ADCDone == 1 );
      lectura_adc = read_adc(ADC_READ_ONLY);
      if ( lectura_adc > Lim_Cero_Flotante ){
         set_timer1(0);
         f = 1;
      }
      else if ( escape >= 0x0100 ){
         frec = 0;
//!         fprintf(WIFI,"\r\nN");
         return 0;
      }
   } while (!f);
// Paso
   f = 0;
```

```
do{
   ADCON = 1;
   delay_us(23);
   ADCDone = 1;
   while ( ADCDone == 1 );
   lectura_adc = read_adc(ADC_READ_ONLY);
   if ( lectura_adc < Lim_Cero_Flotante ){
      f = 1;
   }
} while (!f);

f = 0;
do{
   ADCON = 1;
   delay_us(23);
   ADCDone = 1;
   while ( ADCDone == 1 );
   lectura_adc = read_adc(ADC_READ_ONLY);
   if ( lectura_adc > Lim_Cero_Flotante ){
      frec_count = get_timer1();
      f = 1;
   }
} while (!f);
if (frec_count <= 48 ) return 0.00;
else{
   frec_count -=40;
}
fprintf(WIFI,"\r\n int Frec: %Lu",frec_count);
frec = (float) frec_count/2000000;     // en segundos
fprintf(WIFI,"\r\n Periodo: %1.6f s // %2.4f ms",frec,frec*1000);
frec = 1/frec;
```

```
   fprintf(WIFI,"\r\n Frecuencia: %2.4f Hz",frec);


 return frec;

}


/*

NOTAS:

El do- while no me funcionó con una condicion en el whiel(condicion), por eso
los if para evaluar la condicion de flag.

*/


void show_time_diff_signals(void){

   float dt;

   float fi;

   float tm = 0;

   float coseno;


   dt = (float) retardo / 2000;                 // Borrar en programa final, es
solo de muestra del Diferencial de tiempo

   tm = (float) retardo / 2000000;           // tm = retardo * 4 * (4 / 32e6)

   tm = tm*314;                              // fi = 2 * pi * f * tm
// f = 50hz

   fi = (float) retardo * 0.000157;             // 0.000157 es el resultado de
la conversion del conteo del timer 1 a tiempo y multiplicado por 2*pi*f

   coseno = cos(fi);


   fprintf(WIFI,"\r\nRe = %Lu ",retardo);

   fprintf(WIFI,"\r\nDt = %2.4f ms",dt);

   fprintf(WIFI,"\r\nFi = %1.6f rad",fi);

   fprintf(WIFI,"\r\nCos(fi) = %2.4f",coseno);

   fprintf(WIFI,"\r\nPF = %2.2f",abs(coseno));      // valor absoluto y redonde
de 2 decimales - Usar funcion Ceil ? o mucha rom?

   fprintf(WIFI,"\r\n\n");
```

```
}

void reset_buffers(void){

   reset_buffer(buffer_rms_v);
   reset_buffer(buffer_rms_i);
   reset_buffer(buffer_rms_p);
   reset_buffer(buffer_max_v);
   reset_buffer(buffer_max_i);
   reset_buffer(buffer_cos_f);
}

void reset_buffer (char *reset_buff){
   for (char j = 0; j <=5 ; j++){
      reset_buff[j] = 0;
   }
}

void check_buffers (void){

   check_buffer(buffer_rms_v);
   check_buffer(buffer_rms_i);
   check_buffer(buffer_rms_p);
   check_buffer(buffer_max_v);
   check_buffer(buffer_max_i);
   check_buffer(buffer_cos_f);
}

void check_buffer (char * check_buff){
   for (char j = 0; j <=5 ; j++){
      fprintf(WIFI,"%x ",check_buff[j]);
```

```
   }
   fprintf(WIFI,"\r\n");
}

void serialOut_uart1(int DatoTx)
{
         TXREG1 = DatoTx;   //TRANSMIT STATUS AND CONTROL REGISTER
         while(!TRMT1);        //Transmit Shift Register Status bit
}



void send_tx_buffer(char *selected_buff){


      serialOut_uart1(BYTE_INICIO); // Enviar el byte delimitador de inicio
      serialOut_uart1(ADDRESS_WIFI); // Mandar la direccion del receptor
      serialOut_uart1(0x23);
      serialOut_uart1(selected_buff[0]);     // Tipo de dato buf_voltaje,
buf_corriente, buf_potencia, buf_v_vmax, buf_i_max, buf_cos_fi
      serialOut_uart1(selected_buff[1]);     // Mandar el byte de dato 1   (ej
12.34)
      serialOut_uart1(selected_buff[2]);     // Mandar el byte de dato  2
      serialOut_uart1(selected_buff[3]);     // Mandar el byte de dato  .
      serialOut_uart1(selected_buff[4]);     // Mandar el byte de dato  3
      serialOut_uart1(selected_buff[5]);     // Mandar el byte de dato  4
      serialOut_uart1(ADDRESS_WIFI + selected_buff[0]);// Calcular y mandar el
checksum (direccion + datos)
}
```

Bibliografía

[1] Internet de las cosas. (2021). Wikipedia. Recuperado de https://es.wikipedia.org/wiki/Internet_de_las_cosas

[2] Microchip. "*PIC12(L)F1840 Data Sheet"*. DS41441B datasheet. [Revisado Octubre 2021].

[3] Microchip. "*PIC18(L)F2X/4XK22 Datasheet*". DS40001412G datasheet. [Revisado Octubre 2021].

[4] Microchip. "*MCP6001/1R/1U/2/4 Datasheet*". DS20001733L datasheet. [Revisado Octubre 2021].

[5] Microchip. "*MCP1703 Datasheet*". DS22049F datasheet. [Revisado Octubre 2021].

[6] ESP826. (2021). Wikipedia. Recuperado de https://es.wikipedia.org/wiki/ESP8266

[7] NodeMCU. (2021). Wikipedia. Recuperado de https://es.wikipedia.org/wiki/NodeMCU

[8] Pueyo, Héctor, Marco, Carlos y QUEIRO, Santiago; "Circuitos Eléctricos: Análisis de Modelos Circuitales 3ra Ed. Tomo 1"; Editorial Alfaomega; 2009.