Proyecto Medidor de Corriente para Microcontroladores

Ramon Stiven Sarmiento Castro 2200503
Carlos Humberto Díaz Salazar 2182353

3 de junio de 2025

I. Introducción

El propósito de este proyecto es desarrollar un sistema para medir la corriente de microcontroladores, como el ESP32. Se lleva a cabo la medición en dos niveles de corriente para un microcontrolador que opera en dos modos: uno de bajo consumo y otro de alto consumo, y se especifica que el microcontrolador cambia de modo cada 10 segundos.

El circuito no solo se encargará de medir la corriente, sino que también transmitirá los datos de forma inalámbrica a un servidor Web, Blynk, para su monitoreo en tiempo real. Además, el sistema contará con dos modos de operación: manual y automático, lo que permitirá al usuario cambiar entre diferentes rangos de medición de forma sencilla, adaptándose a corrientes pequeñas (μA) y grandes (mA).

Las resistencias shunt que se utilizaron tendrán un burden voltage de 1 mV/100 mA para medir corrientes más altas y de 1 mV/100μA para corrientes más bajas, lo que garantiza que se puedan hacer mediciones precisas en ambos rangos.

II. Objetivos

II-A. *Objetivo general*

Desarrollar un sistema de medición de corriente en un rango de 1μA-1A, voltaje de alimentación y potencia de un microcontrolador y transmitir los datos de forma inalámbrica a un aplicativo Web para su monitoreo.

II-B. *Objetivos específicos*

- Diseñar un sistema de medición de alta precisión de corriente basado en resistencias shunt, que permita medir corrientes en rangos de microamperios (μA) a miliamperios (mA).
- Diseñar un sistema de modo automático que tenga una respuesta rápida para proteger los componentes del sistema.
- Implementar una interfaz inalámbrica utilizando el servidor Web Blynk para transmitir en tiempo real las mediciones de corriente, voltaje y potencia del microcontrolador, permitiendo su monitoreo remoto y la visualización de los datos en dispositivos móviles o computadoras.
- Realizar la verificación de los parámetros de diseño a través de simulación teniendo en cuenta tolerancia de los componentes, no idealidades de circuitos integrados y ruido.
- Diseñar la implementación en PCB del sistema para poder volverlo un producto real.

III. Marco Teórico

III-A. *Microcontrolador Usado*

III-A1. *ESP32*

El ESP32 es un microcontrolador desarrollado por Espressif Systems que se destaca por integrar conectividad Wi-Fi y Bluetooth, múltiples pines de entrada/salida, periféricos analógicos y digitales, y capacidades de bajo consumo. Gracias a su versatilidad y bajo costo, es ampliamente utilizado en aplicaciones IoT, domótica y sistemas embebidos.

Uno de los aspectos más relevantes del ESP32 para aplicaciones portátiles es su capacidad de operar en múltiples modos de energía, permitiendo reducir significativamente el consumo cuando no se requiere procesamiento intensivo o comunicación constante.

Modo Activo: En este modo, el microcontrolador ejecuta instrucciones, procesa datos y realiza tareas de comunicación. Es el modo de mayor consumo energético. Según la documentación oficial, el consumo puede variar entre 80 mA y 260 mA dependiendo del uso del procesador y la activación de periféricos como Wi-Fi o Bluetooth [1]. Por ejemplo:

- CPU activa sin Wi-Fi: **80-160 mA**.
- CPU activa con Wi-Fi: hasta **240-260 mA**.
- CPU con Bluetooth activo: aproximadamente **150 mA**.

Modo de Sueño Profundo (Deep Sleep): Este es uno de los modos de bajo consumo más eficientes del ESP32. En este modo, se apaga la mayoría de los periféricos, incluyendo la CPU y la RAM, manteniéndose activos

únicamente los circuitos esenciales para el temporizador RTC o interrupciones externas. El consumo de energía en este estado puede reducirse a:

- **6 a 150 μA** dependiendo de la configuración.

Modo de Sueño Ligero (Light Sleep): El microcontrolador detiene la CPU pero mantiene encendidos algunos periféricos y memorias. Tiene un tiempo de recuperación más rápido que el modo de sueño profundo, pero consume más energía:

- **0.8 a 5 mA**.

III-A2. ***Resistencia Shunt***

Una resistencia shunt, también conocida como resistencia de derivación, es un componente clave en la medición de corriente eléctrica. Su funcionamiento se basa en la Ley de Ohm, donde la corriente que circula por una resistencia genera una caída de tensión proporcional a su valor resistivo:

$$V = I \cdot R \tag{1}$$

Configuración de Resistencias:

- **Shunt de Baja Corriente (10 Ω)**:
  - Rango óptimo: $1\,\mu$A a $1\,$mA.
  - Sensibilidad: $10\,\Omega \cdot 1\,\mu\text{A} = 10\,\mu$V (detectable por el INA228).
  - Burden voltage: $10\,\Omega \cdot 1\,\text{mA} = 10\,\text{mV}$.
- **Shunt de Alta Corriente (10 mΩ)**:
  - Rango óptimo: $1\,$mA a $320\,$mA.
  - Sensibilidad: $10\,\text{m}\Omega \cdot 1\,\text{mA} = 10\,\mu$V.
  - Burden voltage: $10\,\text{m}\Omega \cdot 320\,\text{mA} = 3{,}2\,\text{mV}$.

En sistemas de medición, se utiliza una resistencia de valor conocido y muy bajo para minimizar la pérdida de potencia, y se mide el voltaje generado en sus extremos. Este voltaje es directamente proporcional a la corriente que atraviesa el circuito, lo que permite calcularla con precisión [2].

III-A3. ***INA228 / INA219***

Los sensores de la familia INA (Current/Power Monitors) de Texas Instruments permiten la medición precisa de corriente, voltaje y potencia mediante una interfaz digital [3], [4]. Estos sensores están diseñados para trabajar con resistencias shunt, aprovechando la caída de tensión generada para calcular la corriente.

INA219*:* Es un sensor de corriente bidireccional que integra un amplificador de precisión y un convertidor analógico-digital (ADC) de 12 bits. Permite medir tanto la corriente como la tensión del bus en aplicaciones de baja y media potencia. Utiliza una interfaz I2C para comunicar los datos al microcontrolador [3].

- Rango de voltaje de bus: hasta 26 V.
- Rango típico de corriente: hasta 3.2 A (dependiendo de la resistencia shunt).
- Precisión: hasta $\pm 1\,\%$.
- Comunicación digital: I2C.
- Resolución: 12 bits.

INA228*:* Es una versión más avanzada, diseñada para aplicaciones que requieren mayor precisión y mayor capacidad de monitoreo. Cuenta con un ADC de 20 bits, lo que permite una resolución mucho más alta [4] para la medición de corrientes muy pequeñas, como las que se generan en el modo de sueño profundo de microcontroladores modernos.

- ADC interno de 20 bits.
- Alta precisión y bajo error de offset.
- Comunicación I2C.
- Monitoreo de voltaje, corriente, potencia y energía acumulada.
- Ideal para aplicaciones de medición de bajo consumo y sistemas energéticamente eficientes.

Comparación y elección*:* Mientras que el **INA219** puede ser una opción adecuada para prototipos o validación de conceptos básicos, el **INA228** representa una alternativa de mayor precisión, ideal para productos finales o pruebas detalladas. En este proyecto se contempla el uso del INA228 por su resolución superior y su capacidad de caracterizar de forma precisa corrientes muy bajas, alineándose con el objetivo de medir el consumo en los estados de operación y sueño del ESP32.

III-B. ***Comunicación I2C***

El protocolo **I2C** (Inter-Integrated Circuit) es un estándar de comunicación serial sincrónica desarrollado por Philips Semiconductor (ahora NXP) [5]. Está diseñado para facilitar la comunicación entre múltiples dispositivos digitales usando sólo dos líneas: una para datos (SDA) y otra para la señal de reloj (SCL).

Características principales:

- **Bus de dos hilos:** SDA (datos) y SCL (reloj).
- **Modo maestro-esclavo:** uno o más maestros controlan la comunicación, y varios dispositivos esclavos pueden conectarse al mismo bus.
- **Direcciones únicas:** cada dispositivo esclavo posee una dirección de 7 o 10 bits.
- **Velocidades comunes:** 100 kbps (estándar), 400 kbps (rápido), y hasta varios Mbps en modos avanzados.
- **Requiere resistencias pull-up** en ambas líneas para asegurar el nivel lógico alto.

Funcionamiento básico: La comunicación I2C es iniciada por un dispositivo maestro que genera una condición de *start*, seguida de la dirección del esclavo al que desea comunicarse, y una señal de lectura o escritura. Si el esclavo responde (ACK), se procede al intercambio de datos. Al finalizar, el maestro emite una condición de *stop* para liberar el bus.

III-C. *Comunicación Serial UART y Puertos COM*

La comunicación serial UART (Universal Asynchronous Receiver-Transmitter) es uno de los métodos más básicos y ampliamente utilizados para la transmisión de datos entre microcontroladores, sensores, y dispositivos periféricos [6]. Esta comunicación se basa en el envío de datos bit a bit a través de una línea de transmisión (TX) y una de recepción (RX), sin necesidad de una señal de reloj adicional.

III-C0a. Características del protocolo UART:

- **Asíncrono:** no requiere una señal de reloj externa, ya que los dispositivos acuerdan una velocidad de transmisión común (baud rate).
- **Full-duplex:** puede transmitir y recibir simultáneamente por canales separados.
- **Configuración común:** 9600 bps, 8 bits de datos, sin paridad, 1 bit de parada.
- **Uso típico:** conexión entre microcontroladores y PCs, módulos Bluetooth, GPS, sensores, y más.

III-D. *Protocolo Wi-Fi*

El protocolo Wi-Fi, basado en el estándar IEEE 802.11, permite la comunicación inalámbrica entre dispositivos a través de ondas de radio [7], facilitando el acceso a redes locales e Internet sin necesidad de cables. Es una de las tecnologías más utilizadas en aplicaciones IoT, ya que permite a microcontroladores como el ESP32 conectarse a servidores, bases de datos o aplicaciones móviles de forma remota.

Características principales:

- **Frecuencia de operación:** típicamente en 2.4 GHz y 5 GHz.
- **Velocidad de transmisión:** puede superar los 100 Mbps en versiones modernas, aunque en microcontroladores suele limitarse a unos pocos Mbps.
- **Topología:** admite redes en modo infraestructura (con router) o modo punto a punto (modo SoftAP).
- **Protocolo TCP/IP:** se utiliza para asegurar la comunicación estructurada entre dispositivos.
- **Alcance:** de 30 a 100 metros en interiores dependiendo del entorno.

Wi-Fi en microcontroladores: Dispositivos como el **ESP32** tienen integrado un módulo Wi-Fi [1], lo que les permite conectarse directamente a redes inalámbricas. Esto facilita el desarrollo de aplicaciones embebidas capaces de:

- Servir páginas web locales.
- Enviar o recibir datos desde aplicaciones móviles.
- Comunicar sensores con servidores web, bases de datos o plataformas como Blynk o Firebase.

III-E. *Lógica de Conmutación*

El circuito utiliza un MOSFET IRF540N para alternar entre R shuntes, controlado por Blynk o el esp32 según el modo y rango detectado:

$$\text{Selección} = \begin{cases} 10\,\Omega & \text{si } I_{\text{medida}} < 1\,\text{mA}, \\ 10\,\text{m}\Omega & \text{si } I_{\text{medida}} \geq 1\,\text{mA}. \end{cases} \tag{2}$$

IV. Diseño del Sistema

IV-A. *Modos de Operación*

IV-A1. *Hardware*

Para poder implementar el modo automático en el sistema, era necesario poner una configuración en las Rshunts para poder escoger cuál se estará usando y así cambiar entre un rango de medida u otro. Como las resistencias son de 10 Ω y 10 mΩ, se puede configurar un unico interruptor en serie a la resistencia de bajo valor de tal forma que si este se activa, ambas queden en paralelo, aproximando la resistencia equivalente a la de menor valor. Así, se puede escoger con un único interruptor entre ambos rangos de medición (Ver figura ***8***).

Como la resistencia tiene un valor muy bajo, un interruptor en serie que se active por medio de la ESP32 del sistema puede ser un MOSFET, sin embargo, se tiene una resistencia de encendido Rds_{ON}, la cual debía ser la mínima posible, por ello se escogió el NMOS ISCH42N04LM7, que cuenta con una $Rds_{ON} = 0{,}8m\Omega$. Algo a tener en cuenta es que el MOSFET es de canal N, por lo que el sistema se convierte en un sensor de corriente *Low-side*.

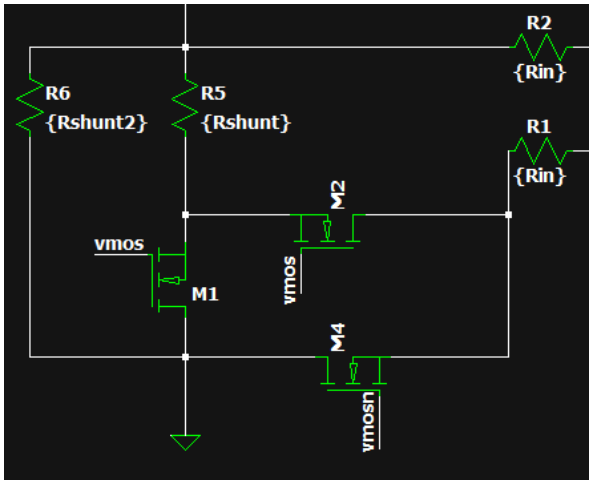

Figura 1: Implementación cambio de rango de medición.

IV-A2. Simulación de modos

A continuación se presenta el esquemático y resultados de simulación para un Sweep DC en cada rango.

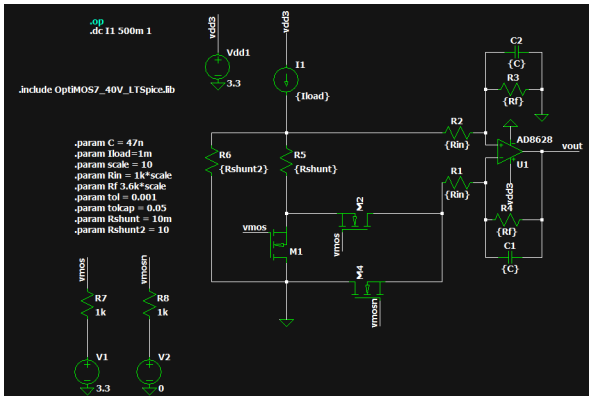

Figura 2: Esquemático para simulación Sweep DC.

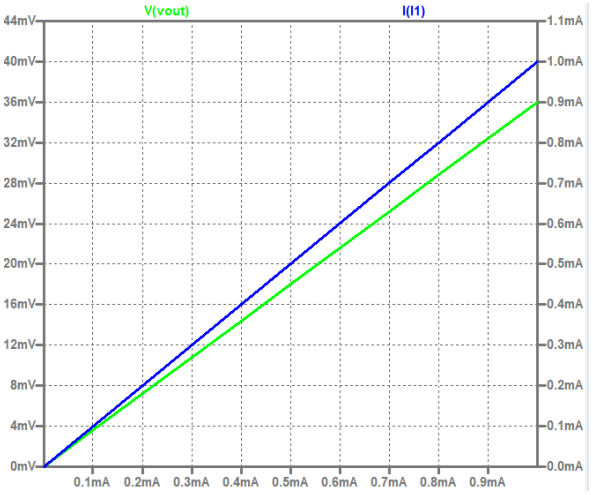

Figura 3: Simulación DC en uA.

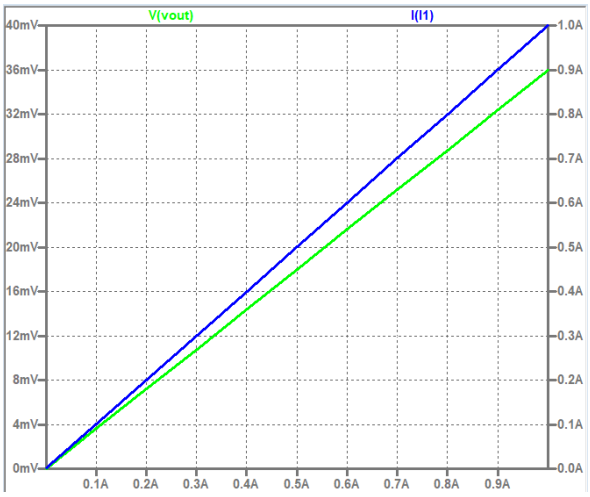

Figura 4: Simulación DC en mA.

Se destaca que ambos rangos tiene por salida de la etapa de pre amplificación un rango de 36uV a 36mV. Se pueden encontrar ambas simulaciones en el repositorio de proyecto. Sweep DC uA.

IV-B. ***Programación del ESP32-s2 mediante CP2102***

Para cargar un programa en el módulo ESP32-s2 sin usar una placa de desarrollo (como el NodeMCU o el DevKit), es posible emplear un convertidor USB a UART como el CP2102, el cual permite establecer comunicación serial entre el computador y el ESP32.

El proceso de conexión y programación se realiza de la siguiente manera:

- **Conexión UART**:
  - El pin **TX** del CP2102 se conecta al pin **RX** del ESP32.
  - El pin **RX** del CP2102 se conecta al pin **TX** del ESP32.
  - Ambos dispositivos deben compartir una conexión común a tierra (GND).
- **Control de arranque (modo bootloader)**:
  - El pin **EN** o **RESET** del ESP32 debe conectarse a un pulsador que permita reiniciar el módulo.
  - El pin **IO0 (GPIO0)** debe conectarse a otro pulsador (con resistencia pull-up) para forzar al ESP32 a entrar en modo de programación.
- **Secuencia para cargar el programa**:
  1. Mantener presionado el botón conectado al pin **IO0**.
  2. Presionar y soltar el botón de **RESET** (EN), mientras IO0 está en bajo (LOW).
  3. Soltar el botón de **IO0**.
  4. Esta secuencia hace que el ESP32 entre en modo *bootloader*, permitiendo su programación desde

el entorno Arduino IDE mediante el puerto COM virtual del CP2102.

Este procedimiento permite cargar código en el ESP32-s2 directamente, sin necesidad de una placa de desarrollo, facilitando la integración del microcontrolador en diseños personalizados [8].

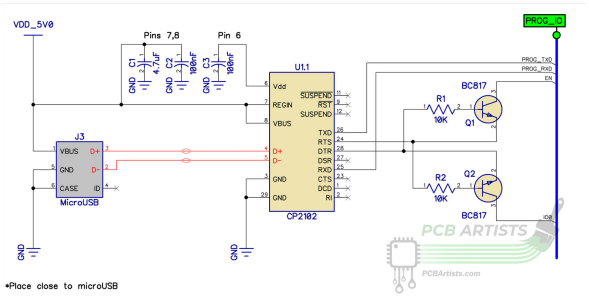

Figura 5: Esquema de conexión

IV-C. Sistema de Conmutación de Modo y Rango

El circuito implementa un sistema dual para medición de corriente que permite operar en modo **automático** o **manual**, en los rangos de microamperios (μA) y miliamperios (mA).

IV-C1. Arquitectura del Sistema

- **Resistencias Shunt**:
  - **10** Ω: Para medición de corrientes bajas (1 μA a 1 mA)
  - **10 m**Ω: Para corrientes altas (1 mA a 1 A)
- **Elementos de Conmutación**:
  - MOSFET IRF540N para alternar entre Rshunts

IV-D. Modos de Operación del Sistema de Medición

El sistema cuenta con dos modos de operación: automático y manual. En ambos, se utiliza el sensor INA228 para medir la corriente que atraviesa una resistencia shunt seleccionada por el microcontrolador ESP32, puesto que existe una etapa de preamplificación, el esp32 permite seleccionar entre dos resistencias: 36 Ω para corrientes del orden de μA y 36 mΩ para corrientes en el rango de mA.

IV-D1. Hardware

Para la implementación del modo automático se diseño un control digital en hardware que permita un cambio en el rango de medición rápido. Este está compuesto de dos comparadores que toman unas referencias de tensión para determinar el rango al cual se debe acomodar el sistema mediante el encendido o apagado de los transistores, y un Latch SR que guarda el estado actual y cambia su salida dependiendo de la decisión de los comparadores.

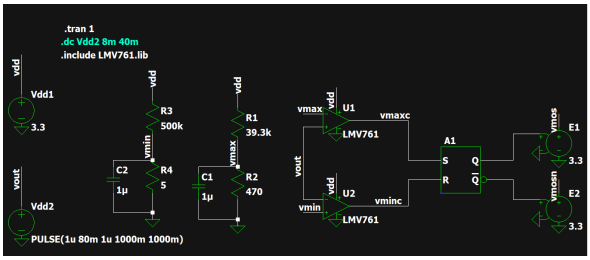

Figura 6: Esquema del control digital

La simulación de LTspice se puede encontrar en el repositorio del proyecto en Github. Simulación control digital.

IV-D2. Modo Automático

En este modo, el sistema selecciona dinámicamente la resistencia shunt adecuada según la magnitud de la corriente detectada:

1. Se realiza una lectura inicial de la corriente usando el valor de shunt por defecto (36 Ω).
2. Si $I < 1$ mA:
   - Se selecciona la resistencia shunt de 36 Ω (GPIO14 en `LOW`).
   - Se calcula la corriente usando esta resistencia.
3. Si $I \geq 1$ mA:
   - Se conmuta a la resistencia shunt de 36 mΩ (GPIO14 en `HIGH`).
   - Se recalcula la corriente con el nuevo valor de resistencia.
4. El LED conectado al GPIO4 permanece apagado indicando modo automático.

IV-D3. Modo Manual

En este modo, el usuario selecciona manualmente el rango de corriente mediante la aplicación Blynk. Esto permite mayor control cuando se conoce el orden de magnitud esperado de la corriente medida:

1. El usuario activa el modo manual desde la app (Modo), encendiendo el LED en GPIO4.
2. Luego, selecciona el rango deseado con otro botón (Rango):
   - **Medición en μA**: GPIO14 en `LOW`, selecciona la resistencia de 36 Ω.
   - **Medición en mA**: GPIO14 en `HIGH`, selecciona la resistencia de 36 mΩ.
3. Esta configuración se mantiene hasta que el usuario la cambie nuevamente.

IV-D4. Lógica de Conmutación (Pseudocódigo)

```
SI (modo_auto == TRUE Y corriente < 1 mA):
    GPIO14 = LOW      // Selecciona shunt de 36 ohmios
    rshunt = 36.0
SI (modo_auto == TRUE Y corriente >= 1 mA):
    GPIO14 = HIGH     // Selecciona shunt de 36 mili-ohmios
    rshunt = 0.036
```

```
SI (modo_auto == FALSE):
    GPIO14 = (valor seleccionado por el usuario
        )
    rshunt = (36.0 o 0.036)
```

Listing 1: Lógica de conmutación entre rangos

IV-D5. ***Implementación en Código***

La lógica anterior ha sido implementada en simulación en proteus con un ArdunioUNO y la programación en hecha en ArduinoIDE (ARDUINO.ino) Apéndice, además se ha utilizando la plataforma Blynk para la interacción remota y el sensor INA219 como emulación del la etapa de preamplificación y el uso del INA228 para la medición de corriente. La selección del rango y del modo se realiza mediante dos switches virtuales en la aplicación Blynk:

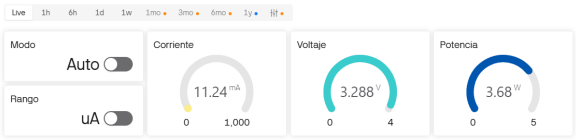

Figura 7: Widgets de Blynk

- Modo: Cambia entre modo automático (apagado) y manual (encendido).
- Rango: En modo manual, selecciona el rango de corriente.

IV-D6. ***Implementación de proteus a Blynk***

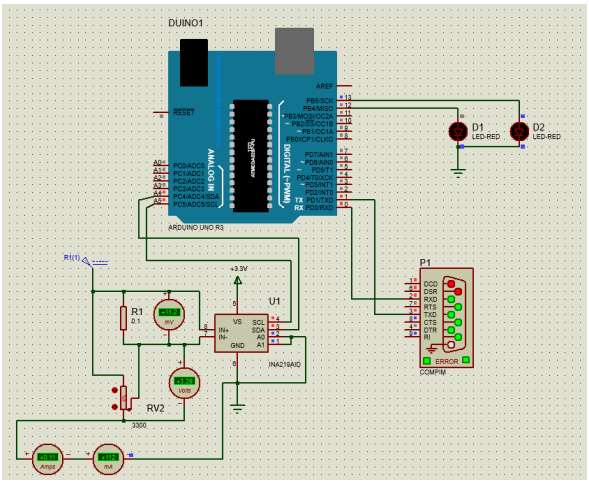

Figura 8: Esquema en proteus (blynk.pdsprj)

Componentes clave:

- **Control por Arduino**: Pin13 (Modo) y Pin12 (Rango)
- **Control por ESP32**: GPIO4 (Modo) y GPIO14 (Rango)

IV-D7. ***Migración a ESP32 y código final***

Posteriormente, la lógica implementada en Arduino UNO fue adaptada para su ejecución en un microcontrolador ESP32, que permite una integración directa con redes Wi-Fi y una conexión más eficiente con la plataforma Blynk. Se desarrolló un código en ArduinoIDE que permite al ESP32 leer los datos del sensor INA228 mediante I2C, seleccionar automáticamente o manualmente el rango de medición, y enviar periódicamente los valores de corriente, voltaje y potencia a la interfaz gráfica de Blynk. El código configura adecuadamente el registro de rango del INA228 y realiza la conmutación entre resistencias shunt mediante los pines GPIO, en función del modo operativo (ESP32_proyecto.ino) Apéndice. Esta versión del sistema mejora la precisión de las mediciones y permite un monitoreo inalámbrico más robusto, superando así las limitaciones del entorno simulado con Arduino UNO en Proteus.

IV-E. ***Pre-amplificación***

Debido a que el *INA228* se configuró para tener un rango de medición de $\pm 40mV$, y la caída de tensión máxima de las resistencias Shunt es de 10mV, se estableció una etapa de preamplificación para acomodar la señal *full-scale* en la entrada del *INA228*. Por temas de protección se escogió una ganancia $Av = 3{,}6$ y luego acondicionando la medición por Software para cuadrar unidades y precisión en la medida. Las resistencias fueron de $R_{in} = 10k\Omega$ y $R_f = 36k\Omega$ para mejorar el desempeño del amplificador teniendo impedancia de entrada alta y un capacitor en paralelo a la resistencia Rf para mejorar la medición DC que es la importante para el proyecto. Las resistencias están en el rango de las decenas de kilo Ohms para reducir el ruido del sistema.

Como amplificador operacional se escogió el *AD8628* pues es de alimentación simple, con el rango adecuado para la aplicación, es de bajo consumo y presenta un offset de máximo $1\mu V$ lo cual es perfecto para temas de precisión. Por efectos de simulación se puso otro operacional en serie como buffer para simular el offset del *INA228* debido a que este no cuenta con modelo *SPICE* pero su valor de offset es similar al del amplificador.

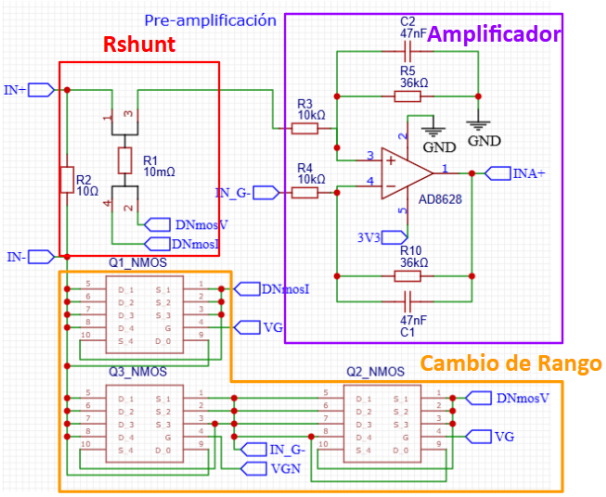

Figura 9: Etapa de preamplifición.

IV-E1. Simulación Montecarlo

Esta etapa es la que puede inducir más errores en la medición, por la misma precisión o errores de los elementos que la componen, el amplificador, resistencias y capacitores, una buena forma de medir la precisión de esta es con la simulación montecarlo. El cual es un análisis estadístico y aleatorio, donde bajo ciertos margenes de tolerancia varían los valores de los diferentes elementos y se hace una simulación por cada punto. Para este caso, se hicieron 300 puntos de simulación para el rango de mili amperios y micro amperios, estos fueron los resultados y el esquemático de la simulación.

Todos los componentes tienen una tolerancia máxima de 0.1 %, los datasheet de todos los componentes pasivos y activos del proyecto se encuentran en el repositorio de Github. Datasheets.

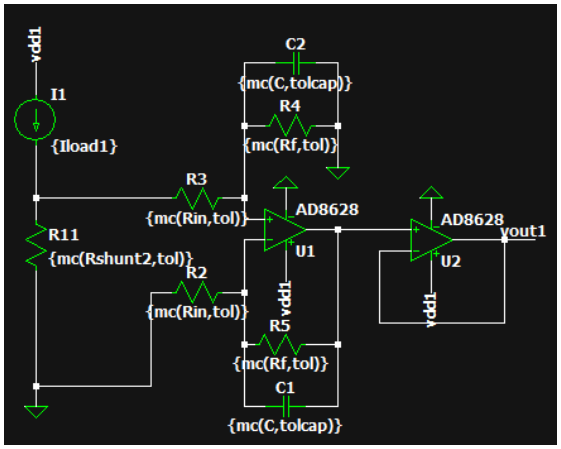

Figura 10: Esquemático para la simulación montecarlo.

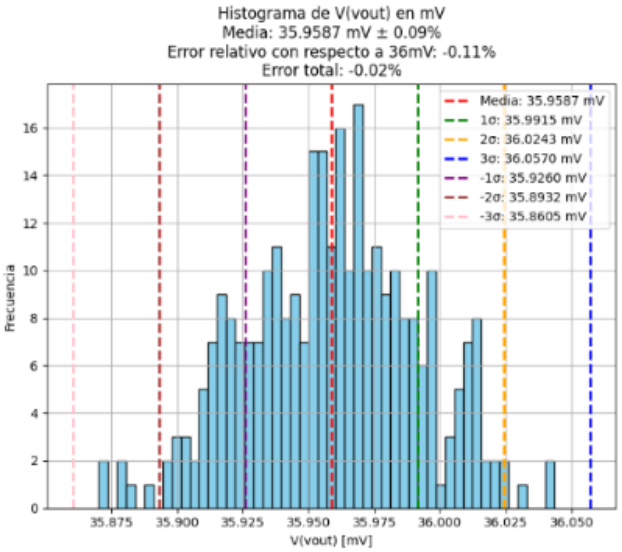

Figura 11: Resultados para el rango de los milli amperios.

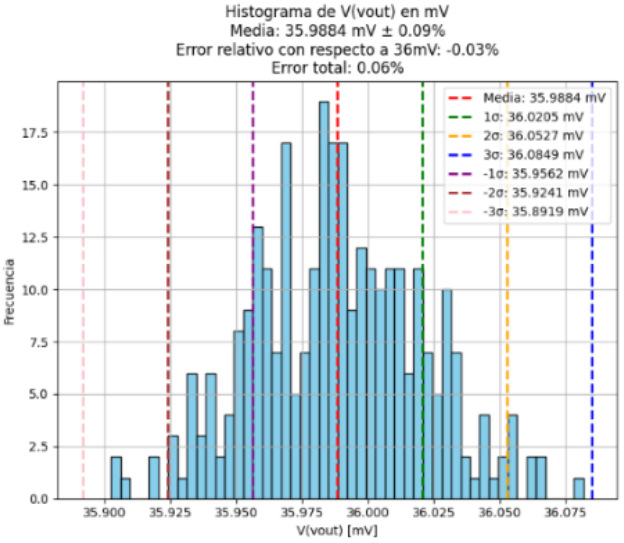

Figura 12: Resultados para el rango de los micro amperios.

Como se puede observar, el error máximo es de $0{,}11\,\%$ con una desviación estándar máxima de $0{,}09\,\%$. Logrando uno de los objetivos específicos del proyecto que es la alta precisión en la medición.

IV-E2. Simulación Ruido

A continuación se presenta el esquemático y los resultados para la simulación de ruido en la etapa de preamplificación.

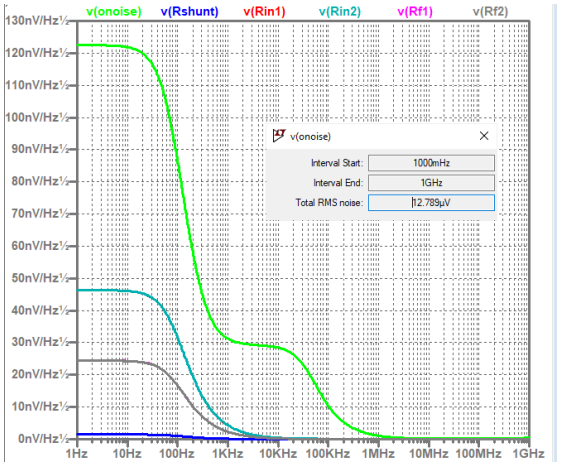

Figura 13: Simulación de Ruido en el rango de uA.

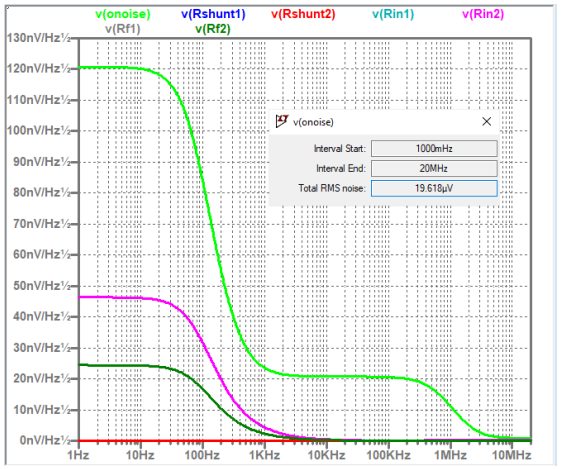

Figura 14: Simulación de Ruido en el rango de mA.

Se observa que en el rango de los uA hay menor nivel de ruido RMS, esto es debido a que solo está midiendo una de as resistencias Shunt, la otra tiene el transistor en circuito abierto y el ruido asociado a esta no se ve representado en la salida.

IV-F. ***Alimentación***

La tensión de alimentación típica para un ESP32 es de 3.3V, este mismo valor de tensión se puede usar para la alimentación del INA228 y del AD8628, pero es usual que pueda existir un consumo de corriente considerable, por lo que una batería tipica de este valor no es suficiente si no puede alimentar la carga del sistema y tener un tensión constante para asegurar el buen comportamiento. Por ello, se escogió primero el regulador TC1262 en su versión de 3.3V, ya que es un regulador que puede entregar sin problema al menos 500mA y mantener una tensión constante, además de que su *Dropout* es pequeño, por lo que la tensión de entrada no tiene que ser excesivamente mayor a los 3.3V de la salida. Se agregó un capacitor de $1\mu F$ tanto a la entrada como a la salida para mejorar la estabilidad de la alimentación.

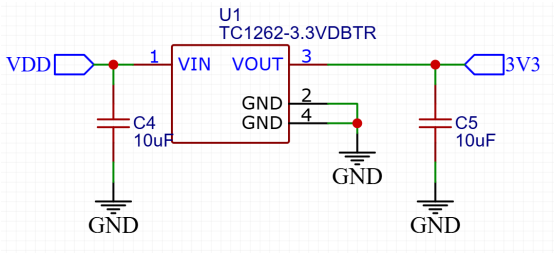

Figura 15: Regulador de voltaje para alimentción de 3.3V.

Para escoger la batería se tomó en cuenta los valores de consumo de los diferentes elementos de mayor consumo en el sistema:

Componente	Consumo máximo (mA)
ESP32	350 @Tx
AD8628	1
INA228	1
TC1262	0.15 @IL=0
Total	**352.15**

Tabla 1: Consumo máximo estimado de los componentes

Se escogió entonces la batería *Li-Po 3.7V 4000mAh 606090*, pues su tensión nominal es suficiente para el buen comportamiento del regulador, y la capacidad de 4000mAh le permite poder alimentar el sistema por lo menos por 11h suponiendo un consumo constante.

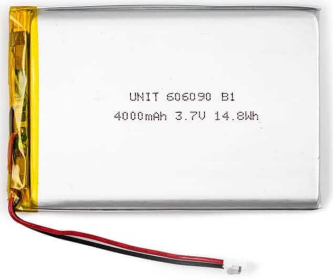

Figura 16: Batería para alimentar el sistema.

IV-G. Módulo de carga

La batería es recargable, por lo que es conveniente implementar un módulo de carga para este tipo de baterías, existen diferentes tipos, el más famoso es el TP4056, pero al ser un módulo lineal es muy poco eficiente; el MCP73831 es un buen módulo para carga de

baterías pero de baja capacidad, por lo que se descarta. Se decidió usar entonces una placa ya diseñada que se basa en el integrado IP2312U, el cual es un DCDC conmutado, lo que mejora mucho la eficiencia. La placa puede puede cargar la batería con un corriente de hasta 2.1A, lo que facilita la carga de la misma para seguir utilizando el sistema, y cuenta con protecciones contra temperatura y niveles de tensión.

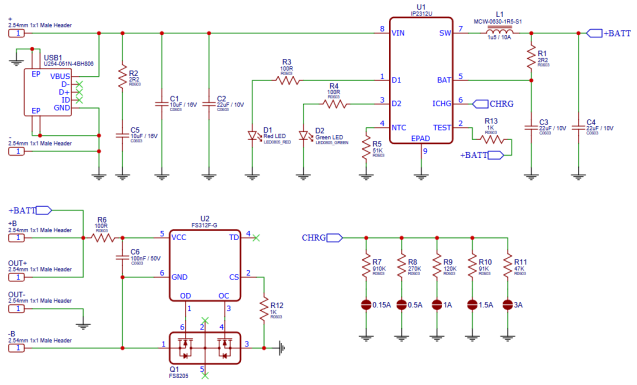

Figura 17: Módulo de carga para la batería [9].

IV-H. Protecciones

Se implemetaron distintos tipos de protecciones al sistema, inversión de alimentación, sobrecorriente y sobrevoltaje.

IV-H1. Inversión de alimentación

Es común que se cometa el error de conectar la batería con la polaridad inversa, esto desafortunadamente tiene consecuencias catastróficas para el sistema, pues la alimentación es simple e invertir la polarización podría dañar todos los componentes. Por ello, se implementó un PMOS en la alimentación que esta siempre en triodo si la polarización de la batería es correcta, de lo contrario, se pone en modo *cut-off*. Además, tiene una Rds_{ON} baja para evitar un consumo de potencia en este.

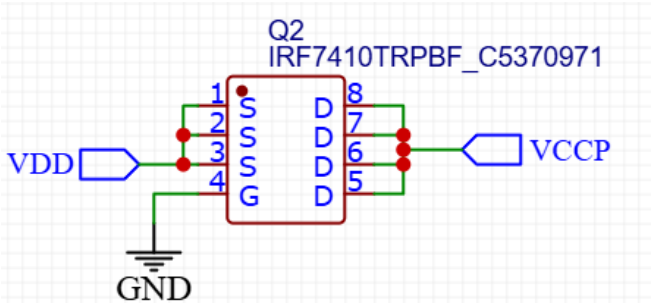

Figura 18: Protección contra inversión de la batería.

IV-H2. Corriente y voltaje

Para protección contra sobre voltajes se utilizó el diodo TVS PESDNC5D5VB, ya que su valor nominal de tensión de encendido corresponde al valor máximo de entrada para el regulador. Para protecciones por sobre coriente se utilizaron fusibles en la alimentación del sistema y en la entrada de medición de corriente, con valores de ruptura de 0.5A y 1A, respectivamente, no se desea que se superen esos valores pues son los máximos de los componentes.

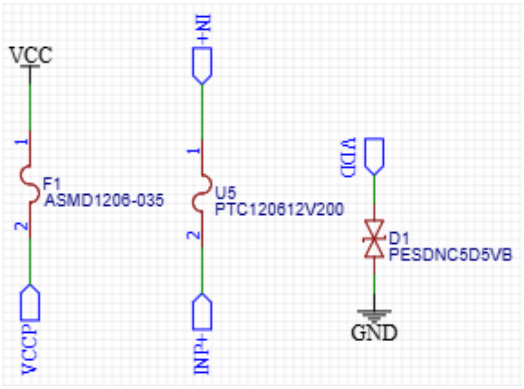

Figura 19: Protecciones de sobrevoltaje y sobrecorriente.

IV-I. Comunicación entre Módulos

En este proyecto se emplea comunicación I2C y UART para la adquisición de datos desde el sensor de corriente, programación del ESP32 y comunicación Wi-Fi para la visualización remota en la interfaz Blynk. A continuación se detallan las dos implementaciones: una con Arduino UNO para simulación y otra con ESP32 para el diseño final.

El archivo `blynk-ser.bat` permite conectar el Arduino UNO simulado en proteus por puerto serial con la aplicación Blynk. Esto se hace a través de la herramienta Blynk Serial Bridge.

Código para Arduino UNO con INA219 (Simulación)

Este código ejemplifica el uso del sensor INA219 con Arduino UNO, midiendo la corriente en miliamperios y enviándola al pin virtual `V1` en Blynk vía conexión serial.

Código para ESP32 con INA228 (Diseño Final)

Para el sistema real, se emplea el sensor INA228 conectado vía I2C al ESP32. Este código lee la corriente, transmite los datos vía Wi-Fi hacia Blynk y además controla el cambio automático de resistencias shunt para dos rangos de medición. En Blynk se muestra tanto la corriente como el rango de medición seleccionado (bajo o alto)

Nota: Aunque en la simulación se empleó el sensor INA219, el diseño final se basa en el sensor `INA228`, el cual ofrece mayor precisión y permite rangos de medición más amplios. Antes del INA228 se ha implementado una etapa de amplificación que adapta la señal proveniente de la resistencia shunt.

IV-J. Sistema completo

A continuación, se presenta la hoja de diseño de todo el sistema.

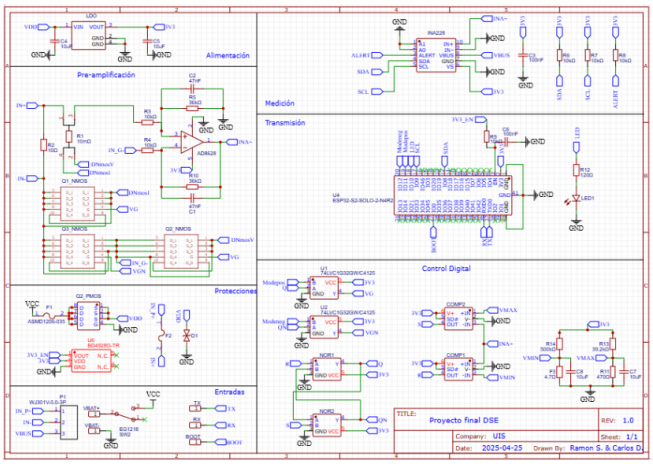

Figura 20: Plano de diseño del sistema.

IV-K. PCB

A continuación se presenta el diseño de la PCB para la implementación del proyecto. los archivos para la fabricación de la PCB se encuentran en el repositorio de proyecto, incluyendo la lista BOM y los precios de todos los elementos.PCB

Además, se realizó un análisis de costos para la fabricación de una unidad o de 25 unidades, en el cuál se presenta un ahorro de aproximadamente 290 dólares, por lo cuál la decisión de fabricar el proyecto en masa en una mejor elección si se deseara continuar con este proyecto y volverlo un producto real, en vez de un simple diseño. Costos

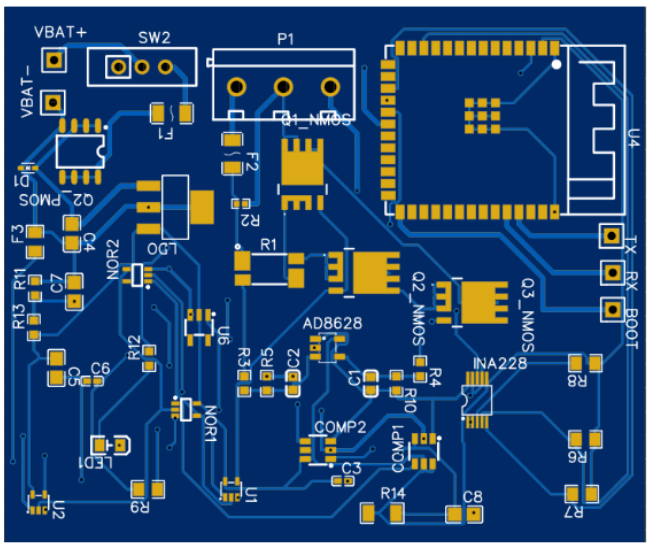

Figura 21: PCB en vista 2D.

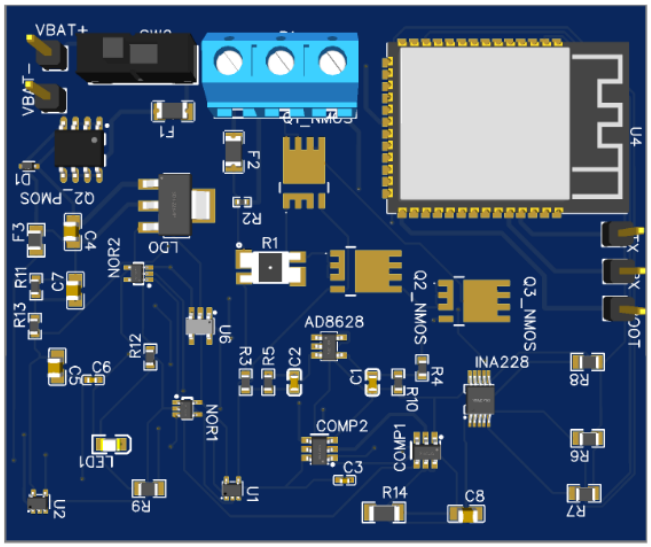

Figura 22: PCB en vista 3D.

V. Conclusiones

Existen errores debido a la precisión de los componentes, a pesar de que se escogieron elementos que destacan en su precisión y bajo consumo, las especificaciones de burden voltage y el diseño del modo automático afecta esta especificación y finalmente errores en la medición.

La falta de modelos de simulación o herramientas para integrar todo el sistema de tipo mixed signal provocaron que para poder verificar el diseño se hiciera por partes y por analogías entre modelos de simulación de los elementos escogidos con los disponibles.

Aunque Blynk es una plataforma intuitiva y útil para proyectos de monitoreo remoto, su versión gratuita presenta varias limitaciones importantes. Solo permite conectar hasta 10 dispositivos y acceder desde un único usuario, lo cual restringe la escalabilidad y el trabajo colaborativo. Además, el límite de 30.000 mensajes mensuales se agota rápidamente en proyectos con múltiples widgets y actualizaciones frecuentes, como en este caso, donde se emplean cinco widgets que reportan datos constantemente. Finalmente, aunque existen planes premium que amplían estas capacidades, el costo de aproximadamente 100 dólares mensuales para 50 dispositivos y 50 usuarios puede resultar elevado para fines académicos o personales.

Apéndice

En el siguiente repositorio se encuentra disponible el código fuente del proyecto, incluyendo la implementación en Arduino IDE, la configuración de Blynk y los esquemas de simulación utilizados en Proteus y seguido de este un video demostrativo.

- Repositorio GitHub: REPOSITORIO Proyecto Medidor de Corriente para Microcontroladores

- Video simulación: VIDEO Proyecto Medidor de Corriente para Microcontroladores
- Tabla de Costos: Costos

Referencias

[1] E. Systems, *ESP32 Technical Reference Manual*, 2022. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

[2] A. Devices, "Current sensing with shunt resistors – application note," 2016. [Online]. Available: https://www.analog.com/media/en/technical-documentation/application-notes/an105.pdf

[3] T. Instruments, *INA219: Zero-Drift, Bidirectional Current/Power Monitor with I2C Interface*, 2017. [Online]. Available: https://www.ti.com/lit/ds/symlink/ina219.pdf

[4] ——, *INA228: High-Precision Digital Power Monitor With 20-Bit ADC*, 2021. [Online]. Available: https://www.ti.com/lit/ds/symlink/ina228.pdf

[5] N. Semiconductors, *I2C Bus: Inter-Integrated Circuit*, 2000. [Online]. Available: https://www.nxp.com/docs/en/user-guide/UM10204.pdf

[6] S. Electronics, "What is uart?" 2015. [Online]. Available: https://learn.sparkfun.com/tutorials/serial-communication/all

[7] I. S. Association, "Ieee 802.11 wireless lans," 2020. [Online]. Available: https://standards.ieee.org/standard/802_11-2020.html

[8] P. Artists, "Esp32 cp2102 programmer schematic," 2023, tutorial en línea. [Online]. Available: https://pcbartists.com/design/esp32-cp2102-programmer-schematic/

[9] J. M. G. S, "Modulo para la carga de baterias li-po ip2312u," 2020. [Online]. Available: https://www.youtube.com/watch?v=nw36sya94_Q