

Proyecto Diseño y Mejoras  $\mu$ Current

Brayan Andres Celis Godoy, Jeiffer Ivan Bernal Tellez

Profesor

Jaime Guillermo Barrero Perez

Universidad Industrial de Santander

Facultad de Fisico-Mecanicas

Escuela de Eléctrica, Electrónica y Telecomunicaciones

Electrónica

Bucaramanga

2024

**Tabla de Contenido**

	<b>Pág.</b>
Lista de Figuras.....	3
Lista de Apéndices.....	3
Glosario.....	4
Introducción.....	5
1.Marco teórico.....	5
2. Objetivos.....	7
2.1 Objetivo General.....	7
2.2 Objetivos Específicos.....	7
2.2.1 Desarrollo del sistema de medición mediante simulación.....	7
2.2.2 Desarrollo del diseño del PCB.....	7
2.2.3 Desarrollo del sistema de visualización de lecturas.....	7
3. Metodología.....	8
3.1 Solución Ideal.....	8
3.2 Solución por bloques.....	10
3.2.1 Selector de Modo Automático.....	10
3.2.2 Módulo de Carga de Batería.....	14
3.2.3 Módulo Conversor DC-DC.....	15
4. Desarrollo del PCB.....	16
4.1 Esquemáticos PCB.....	17
5. Desarrollo de aplicación.....	18
5.1 Aplicativo mediante servidor web.....	19
5.2 Aplicativo dedicado.....	20
6. Conclusiones.....	22
Referencias Bibliográficas.....	24
Apéndices.....	25

### Lista de Figuras

	<b>Pág.</b>
Figura 1: Circuito medición de corriente a voltaje.....	9
Figura 2: Circuito de Medición de Corriente en Rango de $\mu\text{A}$ y $\text{mA}$ con Resultados de Simulación.....	10
Figura 3: Circuito comparador umbral mínimo y máximo.....	11
Figura 4: Circuito de resistencias de medición en rango automático.....	12
Figura 5: Circuito de lógica selección automática.....	13
Figura 6: Circuito medición de corriente autónomo.....	14
Figura 7: Resultados de simulación.....	14
Figura 8: Disponibilidad del Módulo Cargador TP4056: Distribuidores Nacionales y Opciones Online.....	15
Figura 9: Disponibilidad del Módulo Reductor de Voltaje LM2596: Opciones en Distribuidores Nacionales y Plataformas Online.....	16
Figura 10: Esquemático Preliminar del PCB.....	17
Figura 11: Esquemático Final.....	18
Figura 12: Visualización de Corriente en Tiempo Real Mediante Interfaz Web Servida por ESP32: Compatible con PC y Celular.....	20
Figura 13: Visualización de Corriente Mediante Aplicación Dedicada.....	22

### Lista de Apéndices

	<b>pág.</b>
Apéndice A. Programa Esp-32 Visor online.....	26
Apéndice B. Programa ESP-32 aplicación dedicada.....	29
Apéndice C. Tabla Valores Implementos.....	33
Apéndice D. Esquemas PCB.....	34
Apéndice E. Detalles de los Archivos Disponibles en el Repositorio GitHub.....	35

### Glosario

**ESP32:** Microcontrolador de bajo consumo con conectividad Wi-Fi y Bluetooth, utilizado en sistemas embebidos para procesamiento y transmisión de datos.

**MOSFET:** Transistor de efecto de campo utilizado como interruptor en circuitos electrónicos. En el proyecto, se emplean para seleccionar entre diferentes rangos de medición de corriente.

**Resistencia Shunt:** Componente electrónico utilizado para medir la corriente a través de la caída de voltaje generada en ella, según la Ley de Ohm.

**Opamp (Amplificador Operacional):** Componente electrónico utilizado para amplificar señales de voltaje, en este proyecto se utiliza tanto para amplificar la señal de corriente como para comparar valores de umbral.

**Burden Voltage:** Caída de voltaje en una resistencia shunt cuando circula corriente a través de ella. Es un parámetro clave para la precisión de la medición.

**Conversor DC-DC Buck:** Dispositivo que reduce el voltaje de entrada a un nivel de salida más bajo de manera eficiente, utilizado para regular el voltaje en el proyecto.

**TP4056:** Controlador de carga para baterías de litio, utilizado en el proyecto para gestionar de manera segura la carga de la batería Li-Ion.

**$\mu$ Current:** Circuito desarrollado por David Jones que permite medir corrientes extremadamente bajas con gran precisión, usado como base para este proyecto.

**CurrentRanger:** Versión mejorada del  $\mu$ Current, desarrollado por Low Power Lab, que optimiza la medición de corriente y la alternancia automática entre rangos de medición.

**Endpoint:** Punto de acceso o URL utilizado en la comunicación entre dispositivos, en este caso, para la transmisión de datos entre la ESP32 y un dispositivo remoto.

## Introducción

La falta de herramientas adecuadas para la medición precisa de corriente en dispositivos de bajo consumo, como el **ESP32**, afecta negativamente la optimización del rendimiento y la duración de la batería en aplicaciones **IoT**. Medir el consumo de corriente de manera eficiente es fundamental para garantizar que estos dispositivos operen correctamente en sus distintos modos, desde miliamperios hasta microamperios, sin comprometer la eficiencia energética.

Este trabajo se basa en la arquitectura de circuitos previamente desarrollados, como **μCurrent** (Jones, 2010) y **CurrentRanger** (LowPowerLab, 2018), que ofrecen soluciones para la medición de corriente de bajo consumo. Estos proyectos proporcionan una base sólida para este estudio, aunque es necesario mejorar su capacidad de alternar automáticamente entre rangos de medición y reducir las pérdidas energéticas.

El propósito de este proyecto es diseñar una herramienta de medición de corriente que permita alternar automáticamente entre diferentes rangos, optimizando tanto la precisión como el consumo energético en dispositivos portátiles y sistemas embebidos. Esta herramienta busca maximizar la vida útil de las baterías en aplicaciones **IoT**, donde el monitoreo preciso del consumo de corriente es crucial para optimizar tanto el hardware como el software de los dispositivos.

El enfoque de este proyecto incluye el uso de **resistencias shunt**, **MOSFETs**, y **opamps en realimentación** para la amplificación precisa de la señal de corriente, mejorando la exactitud de las mediciones. Además, la **ESP32** se encarga de procesar y transmitir los datos a una interfaz gráfica para su visualización en tiempo real, integrando estas mejoras en el diseño general.

## 1.Marco teórico

1. Baterías de Litio en Sistemas Electrónicos: Las **baterías de ion-litio** son fundamentales en dispositivos electrónicos portátiles debido a su alta densidad energética y capacidad de recarga. Con un voltaje nominal de 3.7V, son ideales para alimentar sistemas que requieren una fuente de energía estable y de larga

duración, como la **ESP32**, que necesita funcionar de manera autónoma. Estas baterías requieren **circuitos de protección** para evitar sobrecargas o descargas profundas, que pueden reducir su vida útil. En este proyecto, se utiliza el **TP4056** para gestionar de forma segura la carga de la batería.

2. Cargadores de Batería de Litio: El **TP4056** es un cargador lineal eficiente para baterías de litio de celda única, con circuitos de protección integrados que controlan el proceso de carga para evitar sobrecargas. El proceso de carga sigue tres fases: carga rápida, carga constante y terminación de carga, optimizando el ciclo de vida de la batería.
3. Conversores DC-DC Buck (Step-Down): Los **convertidores DC-DC Buck** permiten reducir el voltaje de entrada a niveles más bajos de manera eficiente, siendo más efectivos que los reguladores lineales. En este proyecto, se utiliza un convertidor **DC-DC buck** para bajar el voltaje de la batería a los **3.3V** requeridos por la **ESP32**. Entre las opciones disponibles, destacan el **MP1584** y el **LM2596** por su eficiencia y costo.
4. Medición de Corriente con Resistencias Shunt: Las **resistencias shunt** permiten la medición de corriente al medir la caída de voltaje a través de ellas. En este proyecto, son esenciales para realizar mediciones precisas, con un **Burden Voltage** bajo para evitar errores. Un ejemplo típico es 5mV por 500mA.
5. MOSFETs como Interruptores de Baja Rds(on): Los **MOSFETs** se utilizan como interruptores para seleccionar entre diferentes resistencias shunt. Los **MOSFETs de baja Rds(on)**, como el **IRLZ44N**, son ideales en este proyecto para minimizar las pérdidas y mejorar la precisión en la medición de corriente.
6. Placa de Desarrollo ESP32: La **ESP32** es una placa de desarrollo con conectividad **Wi-Fi y Bluetooth**, ampliamente utilizada en sistemas embebidos. En este proyecto, se encarga de procesar las mediciones de corriente y transmitir los datos a través de **Wi-Fi**, permitiendo el monitoreo remoto del sistema.
7. Amplificadores Operacionales (Opamps): Los **amplificadores operacionales (opamps)** son componentes esenciales en el procesamiento de señales electrónicas, especialmente en sistemas de medición de corriente. En este proyecto, los opamps son utilizados tanto como **amplificadores de señal** para aumentar las pequeñas caídas de voltaje en las resistencias shunt, como también **comparadores** en los circuitos de control. Como amplificadores, permiten que las variaciones de corriente en microamperios o miliamperios se conviertan en señales de voltaje más fácilmente detectables. En su función como comparadores, los opamps ayudan a decidir cuándo realizar un cambio entre los diferentes rangos de medición.

8. **Endpoints y Comunicación entre Dispositivos:** En el proyecto, la **ESP32** se comunica con otros dispositivos mediante **endpoints**, los cuales actúan como puntos de acceso a través de una red. Un **endpoint** es una URL o dirección específica que permite el intercambio de información entre la **ESP32** y cualquier dispositivo conectado, como un celular o computador. En este caso, los datos de medición de corriente se envían a través de estos endpoints, lo que permite que los dispositivos conectados reciban los valores en tiempo real y los muestren en la interfaz de usuario. Este enfoque asegura una comunicación eficiente y flexible entre el hardware y los visualizadores remotos.

## **2. Objetivos**

### **2.1 Objetivo General**

Desarrollar un sistema de medición de corriente para microcontroladores que permita monitorizar de manera precisa el consumo de corriente en diferentes modos operativos, con capacidad de cambio automático entre rangos de corriente.

### **2.2 Objetivos Específicos**

#### ***2.2.1 Desarrollo del sistema de medición mediante simulación***

Diseñar y simular un sistema de medición de corriente que permita validar el comportamiento del circuito de manera virtual y verificar su precisión.

#### ***2.2.2 Desarrollo del diseño del PCB***

Desarrollar el esquema y diseño del PCB del sistema de medición de corriente, optimizando tamaño, consumo energético y costo, integrando los componentes seleccionados.

#### ***2.2.3 Desarrollo del sistema de visualización de lecturas***

Desarrollar un sistema de visualización inalámbrica que permita monitorizar las lecturas de corriente en tiempo real, proporcionando una interfaz de usuario sencilla y eficiente.

### 3. Metodología

El diseño del circuito debe cumplir con características específicas que garanticen su correcto funcionamiento en microcontroladores. El sistema será operado por batería, lo que requiere un bajo consumo energético para maximizar la duración de la misma. El circuito estará orientado a la medición de **corriente continua**, y los rangos de medición serán de **500 mili amperios** y **500 micro amperios**.

La transición entre estos rangos deberá ser manual o automática, dependiendo de las necesidades operativas. Para garantizar una interferencia mínima en el sistema, el circuito deberá tener un **voltaje de fuga** de **5 mV/500 mA** y **5 mV/500  $\mu$ A**, con una **salida máxima** de **500 mV** en cualquier rango.

Finalmente, los datos de la medición deberán transmitirse **inalámbrica mente** a un dispositivo externo, como un celular o una computadora, para permitir el monitoreo remoto en tiempo real.

Todo esto se realizará mediante un desglose progresivo, partiendo de un modelo ideal del circuito hasta llegar a un desarrollo completo y detallado del sistema. Para validar el comportamiento del circuito, se realizarán simulaciones utilizando **LTspice**, lo que permitirá verificar el diseño antes de proceder a una posible implementación física.

#### 3.1 Solución Ideal

El circuito de medición de corriente a voltaje es ampliamente utilizado en la industria y su construcción se basa en un **amplificador operacional (opamp)** configurado en **retroalimentación negativa**. En este tipo de configuración, el opamp actúa como un amplificador de transimpedancia, convirtiendo la corriente de entrada en un voltaje proporcional, lo que facilita la medición precisa de corrientes.

*Para* lograr una medición efectiva de corriente en microcontroladores, debemos tener en cuenta las **resistencias de medición** asociadas a los **voltajes de fuga** característicos del sistema. Los valores de estas



resistencias se pueden calcular utilizando la relación entre el **voltaje de fuga** y la **corriente máxima medida**. De esta manera, obtenemos:

- $R_{mA} = \frac{5\text{ mV}}{500\text{ mA}} = 10\text{ [m}\Omega\text{]}$

- $R_{\mu A} = \frac{5\text{ mV}}{500\text{ }\mu\text{A}} = 10\text{ [}\Omega\text{]}$

Con estos valores de resistencias, cumplimos las características de los voltajes de fuga exigidos por el sistema, asegurando una medición precisa sin introducir una carga significativa en el circuito.

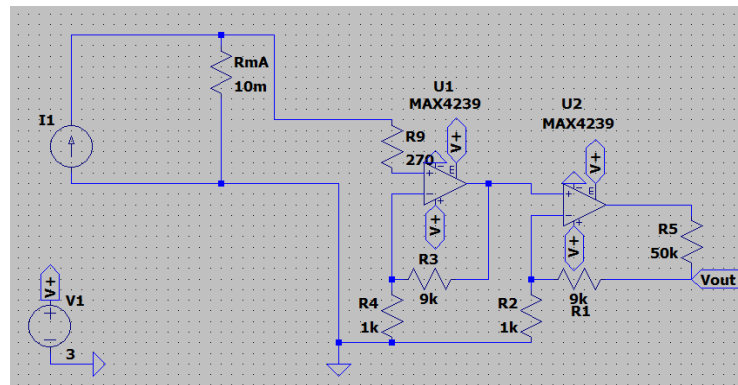
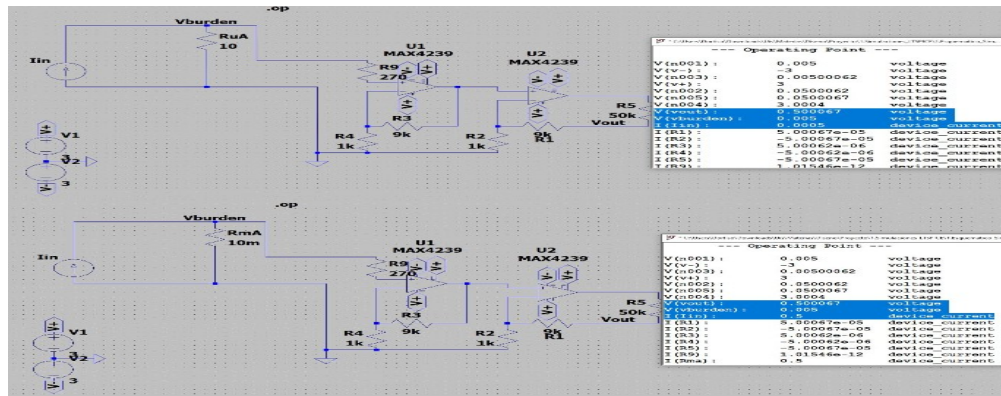


Figura 1: Circuito medición de corriente a voltaje

Como se puede observar en la Figura 1, el circuito utiliza **dos amplificadores operacionales** conectados en cascada para mejorar el **filtrado de la señal** y optimizar la adquisición de datos. Los **valores de amplificación** de estos opamps se han ajustado para asegurar que el sistema sea capaz de aprovechar el **rango completo de saturación** del amplificador, permitiendo una medición adecuada en los dos modos de operación (miliamperios y microamperios).



**Figura 2: Circuito de Medición de Corriente en Rango de  $\mu\text{A}$  y mA con Resultados de Simulación**

Como podemos observar en la Figura 2 la simulación ideal nos muestra que, con una corriente de **500  $\mu\text{A}$**  y **500 mA**, se obtiene una salida de **500 mV**, lo cual era el objetivo de diseño. Este resultado se alcanzó mediante la implementación de una **amplificación con ganancia de 100** en el circuito, lo que permitió escalar las señales de corriente a niveles adecuados para su procesamiento.

Además, para ajustar el voltaje de salida dentro del rango deseado, se aplicó una **división de voltaje**. Este ajuste se logró mediante la colocación de una **resistencia en serie** a la salida del opamp, con un factor de división de 6, lo que permitió reducir el voltaje de salida al valor final de **500 mV** en el punto de medición (**Vout**). De esta manera, conseguimos que el sistema entregue los **500 mV** requeridos, tanto en el rango de **500  $\mu\text{A}$**  como en el de **500 mA**, cumpliendo con las especificaciones del proyecto.

### 3.2 Solución por bloques

A partir de la solución ideal para la medición de corriente y su correspondiente rango de salida, se procede a desglosar el desarrollo de las demás características solicitadas, así como las mejoras que se desean implementar en el sistema. Estas mejoras se estructuran en bloques funcionales, cada uno de los cuales cumple una función específica dentro del sistema global.

### 3.2.1 Selector de Modo Automático

Para el modo de rango automático, se necesita un monitoreo constante de la salida. Cuando se supera un **umbral de voltaje máximo**, se debe cambiar del rango de  $\mu\text{A}$  a  $\text{mA}$ , y en el caso de caer por debajo del **umbral de voltaje mínimo**, se debe cambiar de  $\text{mA}$  a  $\mu\text{A}$ . Esto se puede lograr mediante el uso de **amplificadores operacionales (opamps)** configurados como **comparadores de señal**, encargados de detectar los límites y realizar el cambio de rango de manera automática.

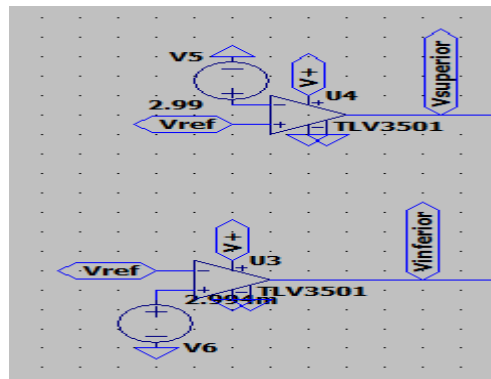


Figura 3: Circuito comparador umbral mínimo y máximo

Como se muestra en la Figura 3 el montaje de estos dos comparadores está diseñado para detectar los límites de cambio de rango. Cuando los comparadores detectan que se ha alcanzado el umbral predefinido, un **MOSFET** utilizado como **switch** actúa para cambiar el rango de medición. Este MOSFET permite una transición suave y eficiente entre los rangos de **microamperios** y **miliamperios**, ajustando la configuración del circuito en función de la corriente que se está midiendo.

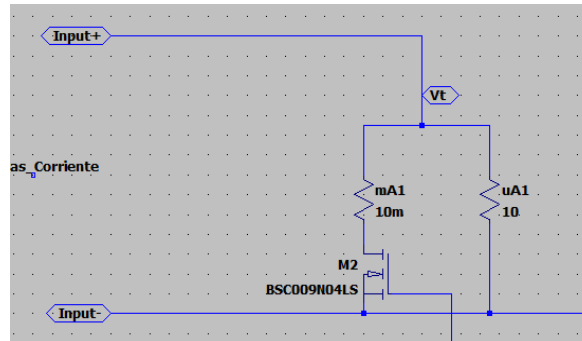


Figura 4: Circuito de resistencias de medición en rango automático

En la **Figura 4**, se puede observar que el **MOSFET** se coloca en serie con la resistencia de **10 mΩ**. Esto permite que el rango de medición de **micro amperios** permanezca activo utilizando la resistencia de **10 Ω**, mientras que, al alcanzar el valor máximo de salida en este rango, el MOSFET se activa, realizando el cambio de rango automáticamente.

Físicamente, la corriente siempre tomará el **camino de menor resistencia**, por lo que, una vez que el **MOSFET** se encienda, la corriente pasará por la resistencia de **10 mΩ**, permitiendo medir correctamente en el rango de **mili amperios**. Esto introduce una característica adicional importante en el circuito: la **resistencia de encendido del MOSFET** no debe ser muy alta, ya que podría alterar el valor de la medición en el rango de **mA**. Si la resistencia es demasiado elevada, afectaría la precisión de las mediciones al introducir una caída de tensión no deseada.

Dado que ya contamos con una **señal lógica** generada al superar los umbrales de voltaje definidos por los comparadores, es posible implementar un **sistema lógico** que controle el encendido del **MOSFET** de manera precisa. Este sistema lógico permitirá automatizar el cambio de rango entre **microamperios** y **miliamperios**, asegurando la independencia total del proceso de medición.

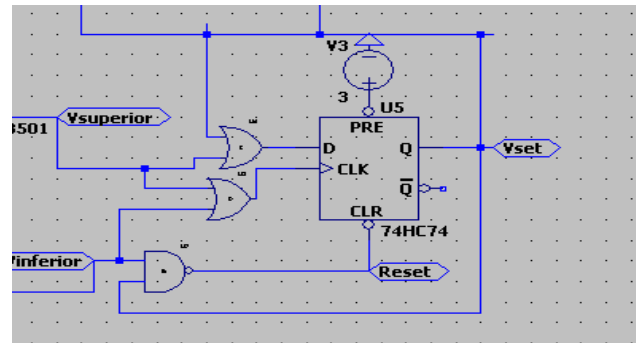


Figura 5: Circuito de lógica selección automática

En el circuito mostrado en la **Figura 5**, obtenemos la lógica de encendido utilizando un **latch Set/Reset** (SR). En la entrada **D** (Set), se encuentra una **compuerta OR** que recibe tanto la señal del **umbral superior** como la **salida del propio latch**. Esta configuración asegura que, una vez que la corriente supera el umbral superior, el latch se mantenga en el estado activado hasta que sea necesario cambiar de nuevo. La **compuerta OR** permite que la señal del umbral superior o el estado del latch puedan establecer el sistema, evitando que el cambio de estado ocurra accidentalmente debido a la medición.

Para gestionar los cambios entre rangos, utilizamos la entrada **CLK** del latch, que responde a **flancos ascendentes**. Las señales de los **umbrales superior e inferior** se combinan mediante otra **compuerta OR**, lo que genera el pulso de reloj necesario para realizar el cambio de rango. Así, el sistema puede cambiar automáticamente entre los rangos de **micro amperios** y **mili amperios** según el valor de la corriente medida.

El **pin CLR** (Reset), que tiene una **lógica negada** (se activa con un nivel bajo), está controlado por una **compuerta NAND**. Esta lógica garantiza que el sistema regrese al rango de  $\mu A$  cuando la corriente cae por debajo del umbral inferior y estaba previamente en el rango de **mA**. Esta combinación asegura que el sistema vuelva al modo de baja corriente solo cuando sea necesario, evitando cambios inesperados en el estado del latch.

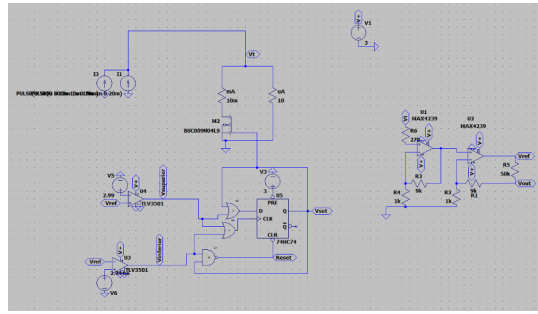


Figura 6: Circuito medición de corriente autónomo

En la **Figura 6** se presenta el **circuito completo de medición** junto con el **selector de rangos autónomos**. Este circuito permite el cambio automático entre los rangos de **mili amperios** y **micro amperios**, de **acuerdo** con los umbrales predefinidos. Para observar el comportamiento del circuito en simulación, se utilizan dos **fuentes de corriente**: una que recorre los valores correspondientes al rango de **mA** y otra para los valores de **μA**.

Al simular el circuito con estas dos fuentes, es posible visualizar las respuestas del sistema tanto en la señal de salida como en las corrientes que atraviesan las **resistencias de medición**. De esta manera, podemos verificar que el sistema cambia correctamente entre los rangos de medición de **10 mΩ** para los mili amperios y **10 Ω** para los micro amperios, dependiendo de los valores de corriente aplicados. Esto asegura que el selector de rangos autónomos funcione de manera eficiente y precisa en el control de la medición.

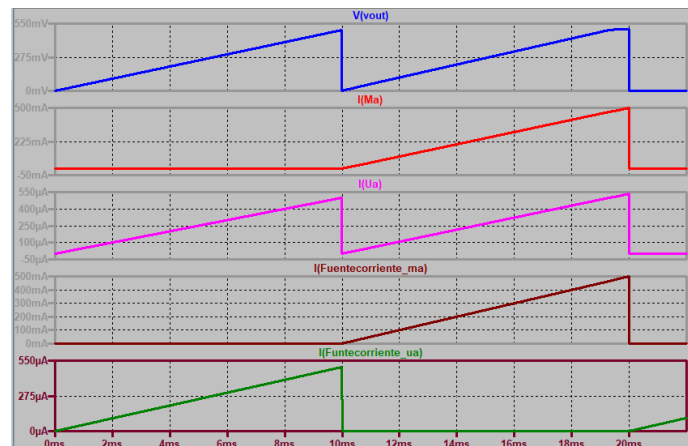


Figura 7: Resultados de simulación

Como se puede observar en los resultados de simulación (**Figura 7**), los gráficos inferiores muestran el comportamiento del circuito cuando las dos fuentes de corriente alimentan el sistema. Las **segunda y tercera**

**gráficas** representan las corrientes que atraviesan las resistencias de medición. En el momento en que la corriente alcanza valores de **mili amperios**, la resistencia asociada a este rango se activa automáticamente, aunque se puede notar una pequeña fuga a través de la resistencia del rango de **micro amperios**.

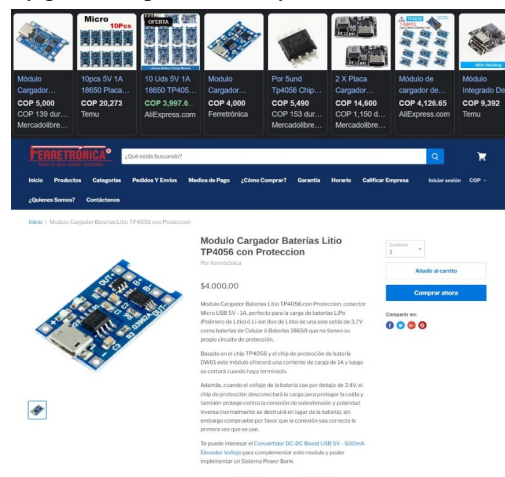
Finalmente, en la **primera gráfica** se muestra el **voltaje de salida**, que se mantiene en un rango cercano a **500 mV** en ambos casos. Sin embargo, se observa que, debido a la resistencia asociada al MOSFET, el valor de salida en el rango de **mili amperios** se ve afectado, mostrando una pequeña desviación con respecto al valor ideal.

### 3.2.2 Módulo de Carga de Batería

Para maximizar el aprovechamiento de los recursos y garantizar un uso eficiente del circuito, se ha optado por utilizar una batería **Li-Ion** debido a su alta densidad energética y capacidad de recarga. Esto evita el desperdicio de recursos al eliminar la necesidad de reemplazar la batería constantemente.

Existen dos enfoques principales para implementar el sistema de carga:

1. **Compra de un módulo de carga preensamblado:** Esta opción incluye módulos como el **TP4056**, que ya vienen con las protecciones necesarias (contra sobrecarga, descarga profunda, etc.) y un puerto de carga USB integrado, lo que facilita su uso y reduce el tiempo de implementación.
2. **Diseño propio del circuito de carga:** Se puede diseñar un circuito utilizando componentes como el controlador **TP4056**, lo que ofrece mayor flexibilidad en la configuración de la carga. Sin embargo, este enfoque es más laborioso y puede implicar un mayor costo de desarrollo.



**Figura 8: Disponibilidad del Módulo Cargador TP4056: Distribuidores Nacionales y Opciones Online**

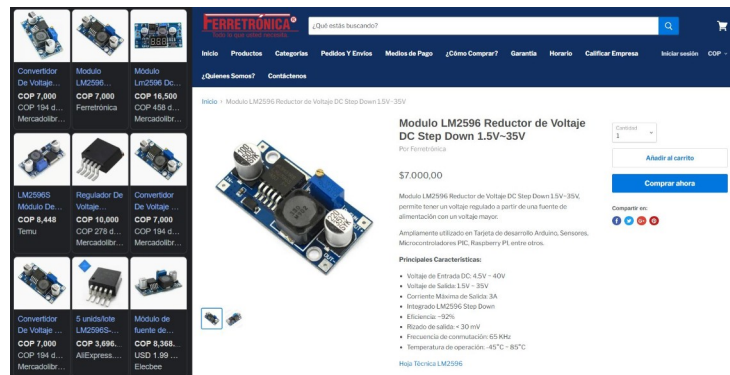
Dado que los módulos **preensamblados** ya incluyen todas las protecciones y puertos necesarios, suelen ser la opción más rentable y sencilla de integrar en el diseño.

En la **Figura 8**, se presenta un ejemplo de las opciones disponibles para la compra del módulo TP4056, tanto en distribuidores nacionales como en tiendas online.

### 3.2.3 Módulo Conversor DC-DC

Un componente adicional que se puede incorporar al diseño es un **convertor DC-DC**, el cual se utilizará para alimentar el **ESP32** y su módulo Wi-Fi, lo que permitirá el envío de datos de manera inalámbrica. Este convertor también será esencial para mantener un valor de alimentación fijo de **3V**, dado que fue bajo este voltaje que se realizó la calibración del sistema.

Al igual que el módulo de carga, el convertor **DC-DC** puede ser adquirido como un módulo preensamblado que cumpla con las características necesarias para garantizar la estabilidad de la salida. Una alternativa es diseñar un circuito propio utilizando componentes discretos, como el **LM2596**, que permite ajustar el voltaje de salida de forma precisa y garantiza la estabilidad del sistema.



**Figura 9: Disponibilidad del Módulo Reductor de Voltaje**

### **LM2596: Opciones en Distribuidores Nacionales y Plataformas**

#### **Online**

La elección entre adquirir un módulo preensamblado o diseñar uno propio dependerá de factores como el costo y el tiempo disponible. En muchos casos, optar por un módulo preensamblado es la opción más práctica, ya

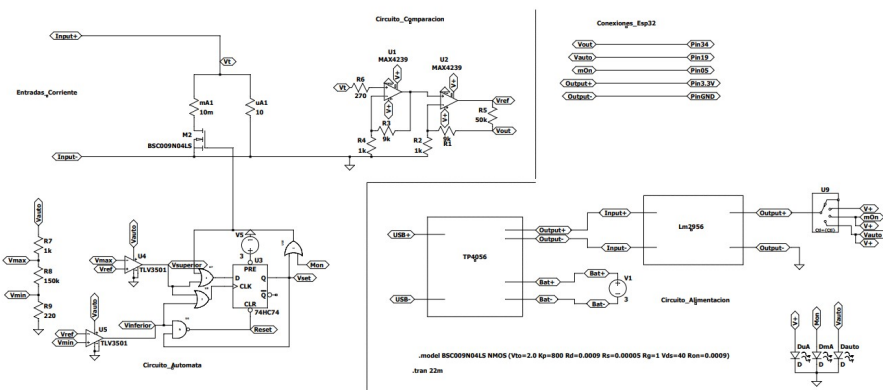


que garantiza la estabilidad del voltaje sin poner en riesgo la calibración del sistema o la integridad del microcontrolador.

En la Figura 9, se muestra la disponibilidad del módulo **LM2596**, tanto en distribuidores nacionales como en plataformas de venta online.

#### 4. Desarrollo del PCB

El desarrollo del **PCB** con los elementos seleccionados comienza con la realización de un bosquejo del **esquemático** en **LTspice**. Este esquema preliminar sirve como una representación inicial, permitiendo organizar los componentes y definir las conexiones, antes de decidir en qué programa se completará el diseño final del **PCB** para proceder con su construcción.

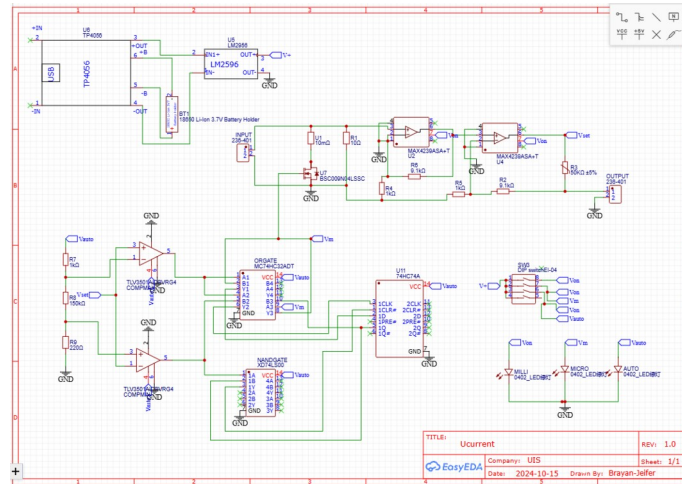


**Figura 10: Esquemático Preliminar del PCB**

En la **Figura 10**, se puede observar la organización y separación de los diferentes bloques del circuito, como el **circuito de comparación**, el **circuito automático**, y los integrados de **alimentación**, tales como el **TP4056** y el **LM2596**. También se incluyen implementaciones finales, como los **LEDs de encendido** para indicar cada modo de rango, así como las conexiones adicionales para la **ESP32**.

##### 4.1 Esquemáticos PCB

Con todos los componentes del sistema identificados y claramente definidos, el siguiente paso es seleccionar el programa adecuado para la realización del **PCB**. Para este proyecto, se ha optado por utilizar **EasyEDA**, debido a su accesibilidad en línea y su amplio soporte de bibliotecas actualizadas. **EasyEDA** ofrece una gran variedad de componentes y proveedores, lo que facilita la integración de los elementos seleccionados en el diseño del **PCB**.



### Figura 11: Esquemático Final

En la **Figura 11**, se presenta el resultado final del diseño en **EasyEDA**, mostrando la construcción del circuito con todos los componentes y sus respectivas referencias. Este enfoque también permite consultar precios y disponibilidad de los elementos en el mercado, información que se detalla en el **Apéndice C**. Además, en el **Apéndice D**, se encuentra el montaje de las capas del **PCB** para su fabricación.

## 5. Desarrollo de aplicación

Dadas las especificaciones requeridas para el circuito, donde la única función obligatoria del **ESP32** es enviar los datos a una **interfaz gráfica** para su visualización en un computador o dispositivo móvil, es importante tener en cuenta la incertidumbre que se genera al convertir una señal analógica a digital. El **ESP32** tiene un convertidor analógico a digital (ADC) de **12 bits**, por lo que debemos calcular la incertidumbre considerando el **rango máximo de salida**, que en este caso es de **500 mV**.

$$\text{incertidumbre} = \frac{2^{\text{bits}}}{\text{rango Max}} = \frac{2^{12}}{500 \text{ mV}} = 0,122 \text{ mV}$$

Con este dato, obtenemos que la incertidumbre en la visualización será de aproximadamente  $\pm 0,122 \text{ mV}$  en la medición, debido a la resolución de los **12 bits** del ADC del **ESP32**. A partir de este valor, el siguiente paso es definir cómo se realizará la lectura y visualización de los datos. Existen dos posibles enfoques para el desarrollo de la aplicación:

1. **Servidor web con el ESP32:** Se decidió optar por esta opción debido a la **autonomía del circuito**, ya que el ESP32 solo funcionará como un mediador entre el circuito de medición y el dispositivo de visualización. En este caso, se desarrollará un **servidor web** utilizando el ESP32, permitiendo que cualquier dispositivo conectado a su red visualice los datos a través de un navegador. La página web se desarrollará en **HTML**, lo que permitirá que los datos sean accesibles desde cualquier dispositivo con un navegador, siempre y cuando esté conectado a la red generada por el ESP32. Esta opción es práctica y versátil, permitiendo el acceso sin necesidad de instalar software adicional.
2. **Aplicativo dedicado:** Aunque se ha optado por el servidor web, se incluirá un apartado para explorar la posibilidad de desarrollar un **aplicativo dedicado**. Este enfoque permitiría cambiar el procesamiento de la señal, controlando completamente el sistema a través del microcontrolador **ESP32**. Un aplicativo dedicado podría desarrollarse tanto para computador como para dispositivos móviles, lo que ofrecería mayor control sobre el sistema y optimizaría la experiencia del usuario. Sin embargo, esta opción presenta la desventaja de requerir versiones distintas para cada tipo de dispositivo, además de la necesidad de que los usuarios instalen la aplicación.

Cada enfoque tiene sus ventajas y desventajas. El **servidor web** es más versátil y fácil de implementar, mientras que un **aplicativo dedicado** ofrece mayor control y personalización, con la posibilidad de modificar cómo se procesa y controla la señal directamente desde el **ESP32**.

### 5.1 Aplicativo mediante servidor web

El diseño del circuito es completamente autónomo en el procesamiento de la señal, y la **ESP32** se utilizará únicamente para transmitir la información a un dispositivo móvil o un computador. Por lo tanto, el desarrollo del

**aplicativo de visualización** se centrará exclusivamente en mostrar el valor medido. Dado que el circuito del **latch** maneja la lógica del cambio de rangos, la **ESP32** solo deberá implementar un sistema que permita visualizar los datos a través de un navegador web.

El desarrollo del aplicativo web incluirá:

- Un **puerto de entrada** de la **ESP32** para la visualización del valor medido (corriente).
- Un **puerto de detección** que indicará si el sistema está operando en el rango de **miliamperios (mA)** o **microamperios ( $\mu$ A)**.
- Un **puerto para el modo automático**, que indicará si el sistema está realizando los cambios de rango de manera automática.

Otra opción para visualizar los datos es conectando la **ESP32** a una red Wi-Fi existente. Conociendo la **dirección IP** de la **ESP32**, los usuarios podrán acceder a la interfaz web y visualizar los datos desde cualquier dispositivo conectado a la misma red. Esta alternativa ofrece mayor flexibilidad, permitiendo el acceso remoto a los datos sin necesidad de depender únicamente de la red generada por la **ESP32**.

Este diseño web permitirá a los usuarios visualizar, en tiempo real, el estado y las mediciones del sistema desde cualquier dispositivo conectado a la red de la **ESP32**, ya sea a través de su red local o una red Wi-Fi existente.



**Figura 12: Visualización de Corriente en Tiempo Real**  
**Mediante Interfaz Web Servida por ESP32: Compatible con**  
**PC y Celular**

En la **Figura 12**, se muestra la página web tanto en un **PC** como en un **dispositivo móvil**, ilustrando la interfaz diseñada para la visualización de los datos. En el **Apéndice A** se incluye el código fuente utilizado para el desarrollo del servidor web, junto con una explicación detallada de su funcionamiento. En este caso, la **ESP32** ha sido configurada para generar su propia red Wi-Fi, a la cual se conectan los dispositivos que visualizarán los datos.

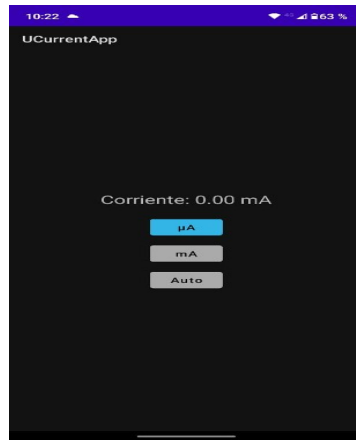
## 5.2 Aplicativo dedicado

En un enfoque donde se utilice un **aplicativo dedicado**, la mejor opción sería eliminar la lógica de control del circuito en favor de un procesamiento centralizado en la **ESP32**. En lugar de depender del **latch** para los cambios de rango, el **procesamiento de la señal** se realizaría directamente en la **ESP32**, que se encargaría también de controlar el **MOSFET** encargado de los cambios entre los diferentes rangos de medición.

El desarrollo de este aplicativo dedicado incluirá:

- Un **puerto de entrada** en la **ESP32** para la visualización del valor medido (corriente).
- Un **puerto de activación**, que se encargará de controlar el encendido del **MOSFET** para alternar entre los diferentes rangos (microamperios, miliamperios y modo automático).

El uso de un **aplicativo dedicado** permite una mejora en la eficiencia del sistema, ya que elimina el procesamiento HTML en el propio ESP32. De esta manera, el microcontrolador puede operar en **modo de sueño ligero** la mayor parte del tiempo, despertándose únicamente cuando se detecte un cambio en la corriente o cuando sea necesario enviar datos. Este enfoque reduce el consumo de energía del sistema y optimiza el rendimiento general.



**Figura 13: Visualización de Corriente Mediante  
Aplicación Dedicada.**

En la Figura 13 , se muestra el desarrollo de la aplicación dedicada, la cual está diseñada para obtener los datos desde la **ESP32** mediante **endpoints** en la red. Los endpoints son **puntos de acceso** que permiten que la aplicación reciba la información desde la ESP32 de manera directa y eficiente. Este enfoque mejora la **reducción del procesamiento** en la ESP32, ya que solo se envían los datos básicos de la medición, y todo el procesamiento se realiza en la aplicación móvil. Al centralizar el procesamiento en la app, se asegura una experiencia de usuario más ágil y flexible, mientras que la ESP32 se encarga únicamente de la adquisición de datos.

## **6. Conclusiones**

Este proyecto logró cumplir con los objetivos planteados al diseñar un sistema de medición de corriente preciso y eficiente para dispositivos embebidos, como el ESP32, capaz de alternar automáticamente entre rangos de microamperios y miliamperios. Gracias al uso de resistencias shunt, MOSFETs, amplificadores operacionales y la ESP32, se optimizó el monitoreo del consumo de corriente, mejorando la eficiencia energética y extendiendo la vida útil de las baterías en aplicaciones IoT.

Adicionalmente, se realizaron simulaciones en LTspice para validar el comportamiento del circuito antes de su implementación física, lo que garantizó la precisión en las mediciones y minimizó las pérdidas de voltaje. El

diseño del PCB se llevó a cabo utilizando EasyEDA, lo que facilitó la integración de los componentes y el análisis de costos y disponibilidad.

**Flexibilidad en la visualización:** El desarrollo de dos opciones de visualización (servidor web y aplicación dedicada) demostró la flexibilidad del sistema. Esto permite que el dispositivo se adapte a diferentes necesidades de los usuarios, ya sea para un acceso rápido a través de un navegador o para un control más detallado mediante una aplicación especializada.

**Experiencia integral:** El proyecto también permitió aplicar y profundizar conocimientos en diversas áreas de la ingeniería electrónica, como el diseño de circuitos analógicos, la programación de microcontroladores y el desarrollo de interfaces de usuario. Este trabajo ha sido una experiencia integral que refleja la naturaleza multidisciplinaria de la carrera.

**Cambio automático de rangos:** La implementación del cambio automático entre rangos de medición ha sido un gran avance. Este sistema permite una transición suave entre los rangos de microamperios y miliamperios sin intervención manual, lo que facilita enormemente su uso en aplicaciones prácticas.

Finalmente, todos los desarrollos, desde el diseño del **PCB** hasta las aplicaciones de visualización, están disponibles en el repositorio de GitHub (Celis Godoy, 2024), promoviendo el uso abierto y la posibilidad de futuras mejoras por parte de la comunidad. En el **Apéndice E**, se detallan todos los archivos, códigos y simulaciones subidos al repositorio, junto con instrucciones para su uso y descarga.

A lo largo del desarrollo del proyecto, se identificaron varias mejoras que no se implementaron en esta fase, pero que podrían aumentar significativamente la funcionalidad y versatilidad del sistema. Una de las mejoras más importantes es la **conectividad de la ESP32 a una red Wi-Fi existente**, lo que permitiría comunicar las mediciones de corriente a cualquier parte del mundo a través de una conexión a Internet. Con el desarrollo de un sistema basado en una **base de datos en la nube**, sería posible acceder a los datos desde cualquier dispositivo. Un ejemplo de esta funcionalidad sería el uso de **Google Firebase**, una plataforma que podría gestionar y almacenar los datos de forma remota, facilitando el monitoreo global de los dispositivos.

Otra mejora significativa sería la **implementación de una pantalla en el propio dispositivo**, permitiendo al usuario visualizar directamente las mediciones sin necesidad de una interfaz externa. Esto agregaría portabilidad y mayor comodidad en el uso del sistema.

Además, con las herramientas y conocimientos adquiridos durante la carrera, sería posible desarrollar un **modelo 3D** del sistema para su impresión en una impresora 3D, creando una carcasa protectora que asegure todos los componentes del circuito. Este enfoque no solo mejoraría la durabilidad del dispositivo, sino que también lo haría más resistente a condiciones ambientales adversas, aumentando su aplicabilidad en el campo.



### Referencias Bibliográficas

- Celis Godoy, B. A. (2024). **Proyecto Ucurrent UIS** [Repositorio GitHub]. GitHub.  
<https://github.com/BrayanACelisGodoy/ProyectoUcurrentUIS>
- Jones, D. (2010). **μCurrent - Low Level Current Measurement Adapter**. *EEVblog*.  
<https://www.eevblog.com/projects/ucurrent/>
- Libbey, R. (2017). **Resistencias Shunt para Medición de Corriente**. *All About Circuits*.  
<https://www.allaboutcircuits.com>
- LowPowerLab. (2018). **CurrentRanger - A Low Power Precision Current Meter and Data Logger**.  
*Low Power Lab*. <https://lowpowerlab.com/guide/currentranger/>
- Redondo, S. (2020). **Baterías de Ion-Litio: Usos y Características**. *Electro-Tech Online*.  
<https://www.electro-tech-online.com>
- SparkFun Electronics. (2021). **ESP32 Development Guide**. *SparkFun Tutorials*.  
<https://www.sparkfun.com>
- Texas Instruments. (2019). **DC-DC Buck Converter Applications**. *Technical Note*. <https://www.ti.com>
- Wolf, D. (2018). **MOSFET Switching and Application in Power Electronics**. *Power Electronics Magazine*. <https://www.powerelectronics.com>

## Apéndices

### Apéndice A. Programa Esp-32 Visor online

```
#include <WiFi.h>

#include <WebServer.h>

const char* ssid = "ESP32_AP";

const char* password = "123456789";

// Crear una instancia del servidor web

WebServer server(80);

// Definir los pines de entrada

const int pinAutoInput = 19;    // Pin para la entrada Auto (GPIO19)

const int pinRangeInput = 5;    // Pin para la entrada de rango (uA/mA) (GPIO5)

const int inputPin = 34;        // Pin 34 como entrada de señal analógica

// Función para leer la corriente sin conversión

float readCurrent() {

    int sensorValue = analogRead(inputPin);

    float voltage = (sensorValue / 4095.0) * 3300.0; // Convertir a milivoltios (0-3300mV)

    return voltage;

}

// Función para la página principal

void handleRoot() {

    // Leer los estados lógicos de las entradas

    bool isAuto = digitalRead(pinAutoInput);

    bool isMilliAmp = digitalRead(pinRangeInput); // HIGH = mA, LOW = uA

    // Definir la unidad de medida dependiendo del estado del pin de rango

    String displayUnit = "";

    if (isAuto) {

        displayUnit = "Auto";
```

```
} else if (isMilliAmp) {  
    displayUnit = "mA";  
}  
else {  
    displayUnit = "μA";  
}  
  
float currentValue = readCurrent(); // Leer la corriente  
  
// Crear el contenido HTML  
  
String html = "<html><head>";  
  
html += "<meta charset='UTF-8'>"; // Añadir la codificación UTF-8  
  
html += "<style>";  
  
html += "html, body { height: 100%; margin: 0; font-family: Arial; font-size: 1.6em; background-color:  
#87CEEB; }"; // Fondo celeste  
  
html += "body { display: flex; justify-content: center; align-items: center; }";  
  
html += ".container { text-align: center; }";  
  
html += ".box { border: 1px solid black; display: inline-block; padding: 20px; margin-bottom: 20px;  
border-radius: 15px; font-size: 1.6em; }";  
  
html += ".button { padding: 30px 60px; font-size: 1.6em; margin: 10px; background-color: grey; color:  
white; border: none; cursor: pointer; border-radius: 8px; }";  
  
html += ".button.selected { background-color: white; color: black; }"; // Botón seleccionado en blanco  
  
html += ".small-text { font-size: 0.8em; color: #333; }";  
  
html += "</style>";  
  
// Añadir JavaScript para auto-actualización  
  
html += "<script>";  
  
html += "setInterval(function() { window.location.reload(); }, 1000);"; // Refrescar cada 1 segundo  
  
html += "</script>";  
  
html += "</head><body>";  
  
html += "<div class='container'>";
```

```

html += "<div class='box'><h1>Lectura de Corriente:</h1>";

html += "<h2>" + String(currentValue, 2) + " " + displayUnit + "</h2></div>";

html += "<h3>Selector de medicion</h3>";

// Botones con estado visual según las entradas

html += "<input class='button ' + String((isAuto) ? \"selected\" : \"\") + \" type='button' value='Auto'>\";

html += "<input class='button ' + String((!isAuto && isMilliAmp) ? \"selected\" : \"\") + \" type='button'
value='mA'>\";

html += "<input class='button ' + String((!isAuto && !isMilliAmp) ? \"selected\" : \"\") + \" type='button'
value='μA'>\";

html += "<p class='small-text'>Rangos:<br>0-500 μA<br>0-500 mA</p>\";

html += "</div>\";

html += "</body></html>\";

server.send(200, \"text/html\", html);

}

void setup() {

  Serial.begin(115200);

  pinMode(inputPin, INPUT);

  pinMode(pinAutoInput, INPUT);

  pinMode(pinRangeInput, INPUT);

  WiFi.softAP(ssid, password);

  Serial.println(\"Punto de acceso iniciado\");

  Serial.print(\"IP del ESP32: \");

  Serial.println(WiFi.softAPIP());

  server.on(\"/\", handleRoot);

  server.begin();

}

```

```

void loop() {

  server.handleClient();

  if (WiFi.status() == WL_CONNECTED) {

    esp_sleep_enable_timer_wakeup(5000000); // Configurar para despertar cada 5 segundos

    esp_light_sleep_start();           // Entrar en light sleep

  }

}

```

**Apéndice B.** Programa ESP-32 aplicación dedicada.

```

#include <WiFi.h>

#include <WebServer.h>

const char* ssid = "ESP32_AP";
const char* password = "123456789";

// Crear una instancia del servidor web
WebServer server(80);

// Definir el pin de salida para el MOSFET
const int mosfetPin = 18;    // Pin para controlar el MOSFET (GPIO18)

// Variables para guardar el estado del sistema
String mode = "uA";         // Modo actual (uA, mA, Auto)

float currentValue = 0.0;    // Valor de corriente leído

String range = "µA";        // Escala de medición actual (uA o mA)

// Función para leer la corriente sin conversión

```

```
float readCurrent() {  
  
    int sensorValue = analogRead(34); // Pin 34 como entrada de señal analógica  
  
    float voltage = (sensorValue / 4095.0) * 500.0; // Convertir a milivoltios (0-3300mV)  
  
    return voltage;  
  
}  
  
// Función para procesar el modo automático  
  
void handleAutoMode(float currentValue) {  
  
    if (currentValue >= 495) {  
  
        // Activar el MOSFET cuando la corriente supere los 495mV  
  
        digitalWrite(mosfetPin, HIGH);  
  
        range = "mA"; // Cambiar la escala a miliamperios cuando el MOSFET esté encendido  
  
    } else if (currentValue < 1) {  
  
        // Apagar el MOSFET cuando la corriente sea inferior a 1mV  
  
        digitalWrite(mosfetPin, LOW);  
  
        range = "µA"; // Cambiar la escala a microamperios cuando el MOSFET esté apagado  
  
    }  
  
}  
  
// Función para manejar el cambio de modo desde la aplicación  
  
void handleSetMode() {  
  
    if (server.hasArg("mode")) {  
  
        mode = server.arg("mode");  
  
  
  
        // Lógica para los modos  
  
        if (mode == "mA") {  
  
            digitalWrite(mosfetPin, HIGH); // Encender MOSFET en mA
```

```
    range = "mA";           // Fijar el rango en miliamperios
  } else if (mode == "uA") {
    digitalWrite(mosfetPin, LOW); // Apagar MOSFET en uA
    range = "µA";           // Fijar el rango en microamperios
  }

  // No encendemos/apagamos directamente en modo Auto, lo hace handleAutoMode()
}

// Responder a la aplicación confirmando el cambio de modo
server.send(200, "text/plain", "Mode changed to " + mode);
}

// Función para devolver los datos actuales en formato JSON
void handleData() {
  currentValue = readCurrent(); // Leer la corriente

  // Si el modo es "Auto", procesar automáticamente y cambiar la escala
  if (mode == "Auto") {
    handleAutoMode(currentValue);
  }

  // Crear el JSON con las variables que queremos enviar
  String jsonData = "{";

  jsonData += "\"current\": " + String(currentValue, 2) + ","; // Agregar lectura de corriente

  jsonData += "\"mode\": \"" + mode + "\","; // Enviar el modo actual

  jsonData += "\"range\": \"" + range + "\""; // Enviar la escala actual (mA o µA)

  jsonData += "}";
```

```
// Enviar los datos en formato JSON

server.send(200, "application/json", jsonData);

void enterLightSleep() {

  Serial.println("Entrando en Light Sleep...");

  // Configurar el tiempo de sueño (por ejemplo, 10 segundos)

  esp_sleep_enable_timer_wakeup(10 * 1000000); // 10 segundos en microsegundos

  Serial.flush(); // Limpiar el buffer serial

  esp_light_sleep_start(); // Entrar en light sleep

  wakeup_count++; // Contador de veces que ha despertado

  Serial.println("Despertado del Light Sleep");

}

enterLightSleep();

}

void setup() {

  Serial.begin(115200);

  // Configurar los pines

  pinMode(34, INPUT); // Pin 34 para la señal de corriente

  pinMode(mosfetPin, OUTPUT); // Pin para el MOSFET

  // Configurar el WiFi

  WiFi.softAP(ssid, password);
```



```

Serial.println("Punto de acceso iniciado");

Serial.print("IP del ESP32: ");

Serial.println(WiFi.softAPIP());

// Configurar las rutas del servidor

server.on("/setMode", handleSetMode); // Endpoint para cambiar el modo

server.on("/data", handleData);      // Endpoint para obtener los datos actuales

server.begin();                      // Iniciar el servidor
}

void loop() {

    server.handleClient(); // Manejar las solicitudes del cliente

}

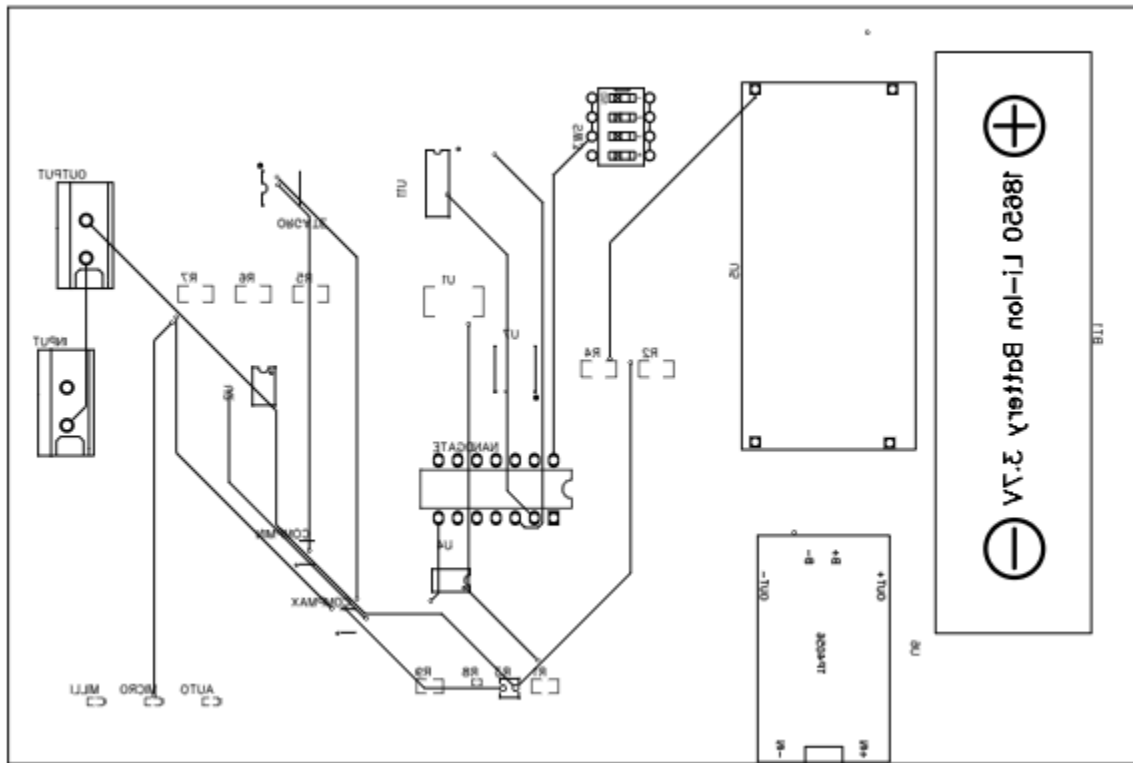
```

#### Apéndice C. Tabla Valores Implementos.

ID	Name	Quantity	Price	Total
1	Leds	3	0,15	0,45
2	TLV3501AIDBVR	2	3,48	6,96
3	236-401	2	1,26	2,52
4	XD74LS00	1	0,173	0,173
5	MC74HC32ADT	1	0,163	0,163
6	10Ω	1	1,19	1,19
7	9.1kΩ	2	0,034	0,068
8	50KΩ ±5%	1	0,41	0,41
9	1kΩ	3	0,1	0,3
10	150kΩ	1	0,41	0,41
11	220Ω	1	0,008	0,008
12	DIP switch EI-04	2	0,45	0,9
13	10mΩ	1	0,1	0,1
14	MAX4239ASA+T	2	4,6	9,2
15	LM2956	1	1	1
16	TP4056	1	2	2
17	BSC009N04LSSC	1	3,09	3,09
18	74HC74A	1	9,059	9,059
	Total			38,001












Estos valores fueron comprobados mediante investigación algunos implementos siendo obtenidos y colocados en <https://www.digikey.com/short/qqwfvwtq> para ver y comprobar su disponibilidad algunos como los módulos referentes a la carga y alimentación fueron investigados y promediados en el mercado local

#### Apéndice D. Esquemas PCB



Siendo estos los montajes superior e inferior del pcb

**Apéndice E.** Detalles de los Archivos Disponibles en el Repositorio GitHub

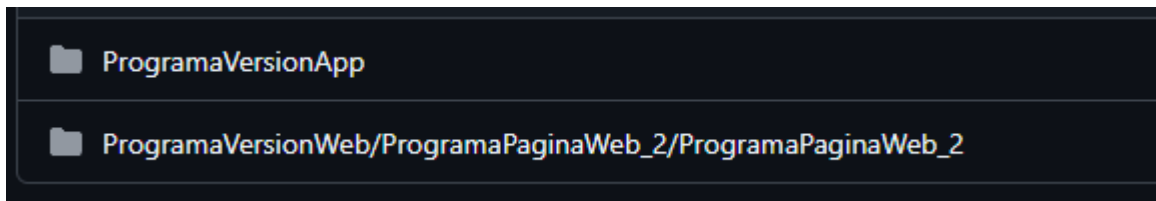
 <b>BrayanACelisGodoy</b> OrganizacionFinal	692f40a · now	 <b>73 Commits</b>
 <b>.idea</b>	q	5 days ago
 <b>1.Simulaciones_LTSPICE</b>	UodateInforme_Sims	10 hours ago
 <b>2ProgramcionEsp32/ProgramaVersionApp</b>	OrganizacionFinal	now
 <b>2Programcion_Visualizador</b>	Organizacion	21 minutes ago
 <b>3.PCB_EasyEDA</b>	UpdatesInforme	45 minutes ago
 <b>4.Informe</b>	update_informee	1 minute ago
 <b>.gitattributes</b>	Reapply "Initial commit"	3 weeks ago
 <b>.gitignore</b>	a	1 minute ago
 <b>desktop.ini</b>	Reapply "Initial commit"	3 weeks ago

Siendo esta la organización del github la cual es

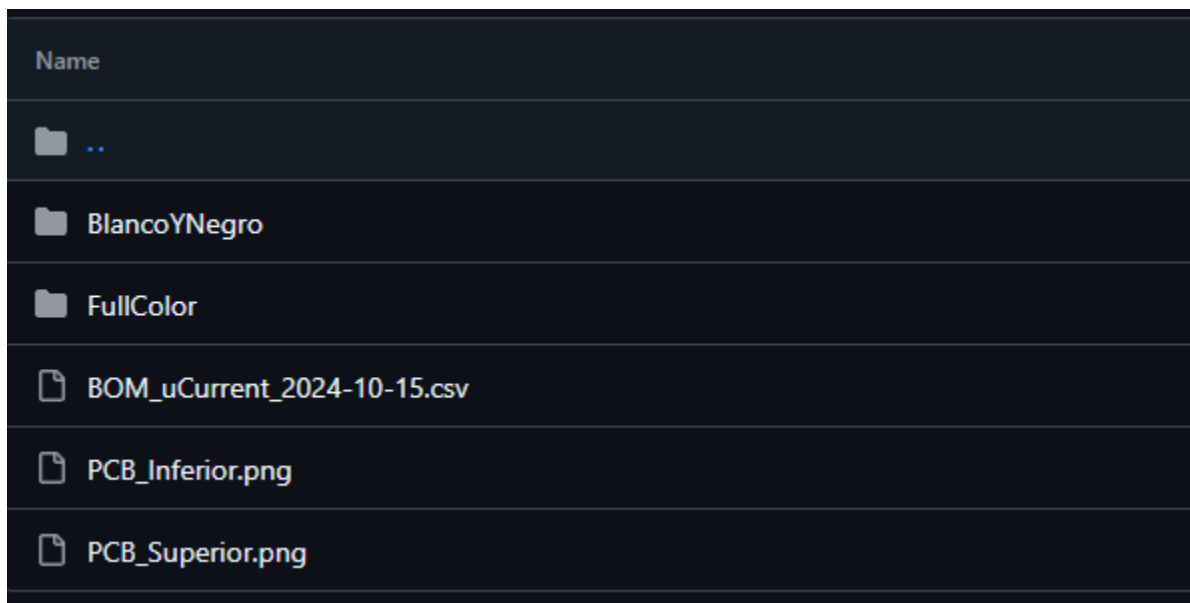
**1.Simulador\_LTSPICE:** Contiene las simulaciones del circuito ideal, del circuito autónomo y el esquemático preliminar.



**2.PProgramacionEsp32:** Incluye la versión del código para el visualizador online y la del aplicativo dedicado, así como el archivo APK para la instalación de la aplicación dedicada.



3. **PCB\_EasyEDA**: Contiene los diagramas del PCB y los costos estimados de los componentes.



4. **Informe**: Carpeta que contiene el documento completo que detalla el proyecto, siendo este informe.