Este trabajo tiene como objetivo el análisis de un medidor para pequeñas corrientes comercial y su rediseño para adecuarlo a una aplicación de monitorización de acelerador de partículas.

El proyecto cuenta con fase de planteamiento y requerimientos, diseño de esquemáticos, placa de circuito impreso y firmware, montaje del sistema y pruebas de funcionamiento.

**Pablo Criado Asensio** es un ingeniero electrónico industrial de El Ejido, Almería. Con este trabajo finaliza su Máster en Electrónica Industrial en la Universidad de Granada.

**Andrés M. Roldán Aranda** es el responsable académico de este proyecto y tutor del estudiante. Es profesor del Departamento de Electrónica y Tecnología de los Computadores en la Universidad de Granada.

TRABAJO FIN DE MÁSTER

**Medidor de corriente aislado para acelerador de partículas**

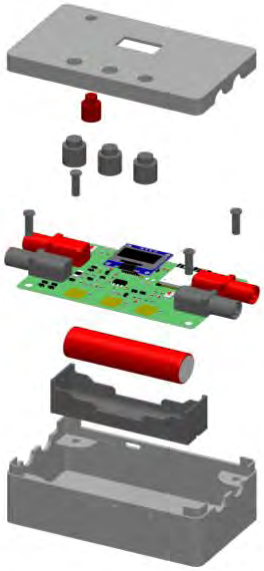
Pablo Criado Asensio

ELECTRÓNICA INDUSTRIAL

UNIVERSIDAD DE GRANADA

Máster en Electrónica Industrial

Trabajo Fin de Máster



# Medidor de corriente aislado para acelerador de partículas

Pablo Criado Asensio

2023/2024

Tutor: Andrés María Roldán Aranda

Printed in Granada, September 2024.

**“Medidor de corriente aislado para acelerador de partículas”**


**MÁSTER EN**
**ELECTRÓNICA INDUSTRIAL**

**Trabajo Fin de Máster**

# *"Medidor de corriente aislado para acelerador de partículas"*

CURSO ACADÉMICO: 2024

Pablo Criado Asensio


MÁSTER EN ELECTRÓNICA INDUSTRIAL

# *"Medidor de corriente aislado para acelerador de partículas"*

AUTOR:

**Pablo Criado Asensio**

SUPERVISADO POR:

**Prof. Andrés Roldán Aranda**

DEPARTAMENTO:

**Departamento de Electrónica y Tecnología de los Computadores**

# “Medidor de corriente aislado para acelerador de partículas”

**Pablo Criado Asensio**

**KEYWORDS:**

Current Measurement, PCB Design, Altium Designer, Visual Studio Code, SolidWorks, ESP32.

**ABSTRACT:**

In a particle accelerator it is important to know accurately the electric current generated during its operation, both to monitor the experiments and to ensure the safety of people and devices. Due to the high voltage required for its operation, it is necessary to use an isolated current meter capable of detecting currents in the order of nanoamperes to milliamperes.

In this work, a study of a commercial current meter is carried out to evaluate its performance and find its shortcomings and possible improvements, with the aim of carrying out a redesign of the system, from the firmware to the design and assembly of the printed circuit board, to make it suitable for the measurement of currents in particle accelerators.

The current meter redesign incorporates an ESP32 microcontroller, which is ideal for this project thanks to its wireless capabilities, as well as robust connectors to ensure system integrity, lithium battery charger, OLED display and LEDs to indicate operating modes, as well as a USB-C connector for programming and communication.

# “Medidor de corriente aislado para acelerador de partículas”

**Pablo Criado Asensio**

**PALABRAS CLAVE:**

Medida de corriente, Diseño de PCB, Altium Designer, Visual Studio Code, SolidWorks, ESP32.

**RESUMEN:**

En un acelerador de partículas es importante conocer con exactitud la corriente que se genera durante su operación, tanto para monitorizar los experimentos como para garantizar la seguridad de las personas y los instrumentos. Debido al elevado voltaje requerido para su funcionamiento, es necesario contar con un medidor de corriente aislado capaz de detectar corrientes del orden de nanoamperios a miliamperios.

En este trabajo se realiza el estudio de un medidor de corriente comercial, para evaluar su funcionamiento y hallar sus carencias y posibles mejoras, con el objetivo de llevar a cabo un rediseño del sistema, desde el firmware hasta el diseño y ensamblaje de la placa de circuito impreso, que lo haga adecuado para la medición de corrientes en aceleradores de partículas.

El rediseño del medidor de corriente incorpora un microcontrolador ESP32, ideal para este proyecto por sus capacidades inalámbricas, así como conectores robustos para asegurar la integridad del sistema, cargador para batería de litio, pantalla OLED y LEDs para indicar los modos de operación, así como un conector USB-C para programación y comunicación.

## *Agradecimientos:*

Quiero darle las gracias a mi tutor Andrés, por haberme guiado durante la elaboración de este trabajo y haberme permitido adquirir una gran cantidad de habilidades y conocimientos de ingeniería en el proceso, así como a todos los compañeros de GranaSAT por su ayuda y amabilidad en los momentos necesarios. Agradezco también a la Universidad de Granada y al profesorado del Máster en Electrónica Industrial por todos los conocimientos y enseñanzas adquiridas durante su desarrollo.

Gracias también a mi familia, por su apoyo durante toda mi etapa educativa hasta la culminación con la elaboración de este trabajo. A Teresa, por animarme en los momentos más duros y ayudarme a seguir siempre adelante.

# Índice general

**Autorización defensa** vii

**Autorización depósito librería** ix

**Abstract** xi

**Agradecimientos** xvii

**Índice general** xix

**Índice de figuras** xxii

**Índice de Tablas** xxvii

**1. Introducción** 1

1.0.1. Objetivos . . . 1

1.0.2. Aceleradores de partículas . . . 1

1.0.3. Medidor de corriente . . . 2

1.0.4. Amperímetro CurrentRanger R3 . . . 3

**2. Requisitos** 4

2.1. Hardware . . . 4

2.2. Firmware . . . 4

2.3. Mecánicos . . . 5

**3. Análisis de un medidor de corriente comercial** 6

3.1. Análisis de hardware . . . 6

3.1.1. Alimentación . . . 7

3.1.2. Electrónica de acondicionamiento . . . 8
3.1.3. Microcontrolador . . . 9
3.1.4. Simulaciones de la electrónica analógica . . . 10
3.2. Análisis de software . . . 12
3.3. Pruebas de funcionamiento . . . 14
3.3.1. Preparación de la PCB . . . 14
3.3.2. Medición de corriente . . . 16
3.4. Mejoras del firmware . . . 22
3.4.1. Copia de seguridad del firmware . . . 22
3.4.2. Cambio de nombre y adición de nuevos comandos . . . 23
3.4.3. Obtención de medidas a la máxima frecuencia . . . 27

**4. Rediseño del medidor de corriente** **29**

4.1. Diseño del circuito electrónico . . . 29
4.1.1. Microcontrolador ESP32-WROOM-32E . . . 29
4.1.2. Conector USB-C . . . 31
4.1.3. Cargador de batería . . . 32
4.1.4. Botón de encendido/apagado . . . 33
4.1.5. Regulador de tensión . . . 33
4.1.6. Convertidor USB a UART . . . 34
4.1.7. Pines de programación . . . 35
4.1.8. Control de medida . . . 36
4.1.9. Amplificador . . . 37
4.1.10. Pulsadores y *pads* táctiles . . . 38
4.1.11. ADC y DAC . . . 38
4.1.12. Pantalla OLED . . . 39
4.1.13. LEDs . . . 40
4.1.14. Conectores tipo banana . . . 41
4.1.15. Diagrama de bloques . . . 41
4.1.16. Consumo estimado del sistema . . . 42
4.2. Diseño del firmware . . . 42

4.2.1. Librerías utilizadas . . . 43
4.2.2. Funcionalidades eliminadas . . . 43
4.2.3. Funcionalidades modificadas . . . 44
4.2.4. Funcionalidades nuevas . . . 46
4.2.5. FreeRTOS . . . 48
4.2.6. Prueba de funcionamiento del firmware . . . 48
4.3. Diseño de la placa de circuito impreso . . . 50
4.4. Montaje de la placa de circuito impreso . . . 52
4.4.1. Diseño de la carcasa . . . 58
4.4.2. Librería de Python . . . 59
4.5. Pruebas del medidor de corriente . . . 61
4.5.1. Pruebas de medición . . . 61
4.5.2. Comunicación por Bluetooth . . . 64
4.5.3. Pruebas de la batería . . . 65
4.5.4. Comparación entre CurrentRanger y medidor de corriente con ESP32 . . . 73
4.5.5. Soluciones y mejoras futuras del medidor de corriente . . . 74

**5. Conclusiones y líneas futuras** **76**

**Bibliografía** **79**

**A. Diagrama de Gantt** **80**

**B. Presupuesto del proyecto** **82**

B.1. Material . . . 82
B.1.0.1. Instrumentación . . . 82
B.1.0.2. Fabricación . . . 82
B.2. Recursos humanos . . . 82

**C. Lista de materiales** **83**

**D. Archivos Gerber para la fabricación** **85**

# Índice de figuras


1.1. Modelo 3D del acelerador de partículas que se va a instalar en GranaSAT . . . . . 2
1.2. Esquema de conexión del amperímetro con el sistema . . . . . 3
1.3. CurrentRanger R3 . . . . . 3

3.1. PCB del CurrentRanger R3 . . . . . 6
3.2. Esquemático del CurrentRanger R3 . . . . . 7
3.3. Bloque de alimentación . . . . . 7
3.4. Bloque de acondicionamiento analógico . . . . . 8
3.5. Bloque amplificador . . . . . 9
3.6. Bloque del microcontrolador . . . . . 9
3.7. Esquemático del CurrentRanger en LTspice . . . . . 10
3.8. Simulación con corriente de 100 $\mu$A . . . . . 11
3.9. Simulación con diferentes valores de corriente . . . . . 11
3.10. Simulación con corriente variable . . . . . 11
3.11. Diagrama de flujo del código . . . . . 12
3.12. Conector de 4 pines soldado al CurrentRanger . . . . . 15
3.13. CurrentRanger montado y encendido . . . . . 15
3.14. Comunicación por puerto serie . . . . . 16
3.15. Fuente de corriente Keithley 220 . . . . . 16
3.16. Configuración de la fuente de corriente Keithley 220 . . . . . 17
3.17. Cable triaxial de la fuente de corriente . . . . . 18
3.18. Conexión de la fuente de corriente al CurrentRanger . . . . . 18
3.19. Medición con el CurrentRanger . . . . . 19
3.20. Interfaz GPIB (IEEE 488) de la fuente de corriente . . . . . 20

3.21. Adaptador GPIB a USB . . . 20
3.22. Osciloscopio HMO2024 . . . 21
3.23. Voltajes medidos con osciloscopio a la salida del CurrentRanger . . . 21
3.24. Reinicio del CurrentRanger en bootloader . . . 22
3.25. Memoria interna del CurrentRanger . . . 23
3.26. Comando ACMEAS . . . 27
3.27. Comando ACBUFFER . . . 27
3.28. Corriente medida y FFT . . . 28
3.29. Medición con forma de onda de sierra . . . 28

4.1. Logo de Altium Designer . . . 29
4.2. Microcontrolador ESP32-WROOM-32E . . . 30
4.3. Esquemático del ESP32-WROOM-32E . . . 30
4.4. Esquemático del conector USB-C . . . 31
4.5. Conector USB-C . . . 31
4.6. Esquemático del cargador de la batería . . . 32
4.7. BQ21040 . . . 32
4.8. Esquemático del botón de encendido/apagado . . . 33
4.9. Esquemático del regulador de 3.3 V . . . 34
4.10. AP7361C . . . 34
4.11. Esquemático del convertidor USB a UART . . . 35
4.12. CH340 . . . 35
4.13. Circuito de programación y reinicio automático . . . 36
4.14. Circuito de control de medida . . . 36
4.15. Circuito amplificador . . . 37
4.16. MAX4239 . . . 37
4.17. Pulsadores y pads . . . 38
4.18. ADC y DAC . . . 39
4.19. Esquemático de conexión de la pantalla por I2C . . . 39
4.20. SH1106 . . . 40
4.21. LEDs del sistema . . . 40

4.22. Conexiones de los conectores . . . 41
4.23. Conectores tipo banana . . . 41
4.24. Diagrama de bloques del sistema . . . 42
4.25. Software para la programación del sistema . . . 43
4.26. Librerías utilizadas . . . 43
4.27. Calibración del ADC . . . 44
4.28. Lectura en AC . . . 44
4.29. Impresión por USB del búfer AC . . . 45
4.30. Lectura del voltaje en DC . . . 45
4.31. Configuración del temporizador para el watchdog . . . 46
4.32. Lectura de los *pads* táctiles . . . 46
4.33. Configuración de la lectura por I2S . . . 47
4.34. Envío de comandos por USB o Bluetooth . . . 47
4.35. Reinicio del sistema . . . 47
4.36. Lectura del botón de encendido/apagado . . . 48
4.37. Creación de tareas con FreeRTOS . . . 48
4.38. Placa de desarrollo ESP32 . . . 49
4.39. Compilación y memoria ocupada por el programa . . . 49
4.40. Prueba del firmware . . . 49
4.41. Diseño de la PCB del medidor de corriente . . . 50
4.42. Parte superior de la PCB . . . 51
4.43. Parte inferior de la PCB . . . 52
4.44. Tiendas de componentes y fabricación . . . 52
4.45. PCB del medidor y stencil . . . 53
4.46. Aplicación de la pasta de soldadura . . . 53
4.47. Desempaquetado de componentes . . . 54
4.48. Microscopio electrónico . . . 54
4.49. Colocación de los componentes . . . 55
4.50. Componentes colocados antes de soldar . . . 55
4.51. Proceso de soldadura en horno infrarrojo . . . 56
4.52. Placas soldadas con horno infrarrojo . . . 56

4.53. Eliminación del estaño sobrante en el conector USB-C . . . 57
4.54. Prueba de encendido y carga del programa . . . 57
4.55. Logo de SolidWorks . . . 58
4.56. Carcasa del sistema . . . 58
4.57. Vista explosionada . . . 59
4.58. Inicialización de la librería . . . 60
4.59. Funciones de pertura y cierre de la comunicación con el puerto serie . . . 60
4.60. Función de envío de comandos . . . 60
4.61. Funciones de lectura de datos del medidor . . . 61
4.62. Ejemplo de envío de comandos . . . 61
4.63. Medición de corriente continua . . . 62
4.64. Medición de corriente negativa . . . 62
4.65. Medición diente de sierra . . . 63
4.66. Medición onda senoidal . . . 63
4.67. Medición onda cuadrada . . . 64
4.68. Pruebas de comunicación Bluetooth . . . 65
4.69. Cargador de baterías ALC-8500 . . . 66
4.70. Software ChargeProfessional para el ALC-8500 . . . 66
4.71. Pantalla de configuración del ALC-8500 . . . 67
4.72. Batería de prueba del ALC-8500 . . . 67
4.73. Cables para la batería . . . 68
4.74. Conexión de la batería al ALC-8500 . . . 68
4.75. Descarga de la batería . . . 69
4.76. Inicio de la carga de la batería . . . 70
4.77. Final de la carga de la batería . . . 70
4.78. Batería del medidor de corriente . . . 71
4.79. Conexión del porta baterías a la PCB . . . 72
4.80. Funcionamiento del sistema con batería . . . 72
4.81. Carga de la batería . . . 73
4.82. Puente para pines . . . 74
4.83. Solución para el botón de encendido/apagado . . . 75

A.1. Diagrama de Gantt del proyecto . . . . . 81

D.1. Bottom layer . . . . . 85
D.2. Bottom overlay . . . . . 85
D.3. Bottom solder mask . . . . . 86
D.4. Top layer . . . . . 86
D.5. Top overlay . . . . . 86
D.6. Top paste mask . . . . . 87
D.7. Top solder mask . . . . . 87

# Índice de Tablas


2.1. Requerimientos de hardware . . . . . 4
2.2. Requerimientos de firmware . . . . . 4
2.3. Requerimientos mecánicos . . . . . 5

3.1. Modos de operación del CurrentRanger R3 . . . . . 13
3.2. Comandos del CurrentRanger R3 . . . . . 14
3.3. Pruebas con el CurrentRanger . . . . . 19
3.4. Comando de información del CurrentRanger R3 . . . . . 23
3.5. Comando para bootloader del CurrentRanger R3 . . . . . 23
3.6. Comandos de unidades del CurrentRanger R3 . . . . . 23
3.7. Comandos de medición del CurrentRanger R3 . . . . . 24
3.8. Comando de batería del CurrentRanger R3 . . . . . 24
3.9. Comandos de calibración del CurrentRanger R3 . . . . . 25
3.10. Comandos de comunicación del CurrentRanger R3 . . . . . 25
3.11. Comandos de depuración de pads táctiles del CurrentRanger R3 . . . . . 25
3.12. Comandos de formato del CurrentRanger R3 . . . . . 26
3.13. Comandos de velocidad del CurrentRanger R3 . . . . . 26
3.14. Comandos de funcionalidades del CurrentRanger R3 . . . . . 26

4.1. Tabla de verdad de los pines de programación . . . . . 36
4.2. Consumo estimado de potencia y corriente del sistema . . . . . 42

5.1. Recapitulación de requerimientos de hardware . . . . . 76
5.2. Recapitulación de requerimientos de firmware . . . . . 77
5.3. Recapitulación de requerimientos mecánicos . . . . . 77

B.1. Presupuesto de los instrumentos y equipos necesarios para el desarrollo del proyecto . . . . . 82

B.2. Presupuesto de fabricación y ensamblaje del sistema . . . . . 82

B.3. Presupuesto de recursos humanos . . . . . 82

C.1. Lista de materiales de la PCB . . . . . 84

# Capítulo 1

# Introducción

Los aceleradores de partículas son instrumentos que emplean campos electromagnéticos para acelerar distintos tipos de partículas a velocidades muy altas. Con el fin de garantizar la seguridad y monitorizar los experimentos, es necesario contar con un dispositivo que mida la corriente generada durante su funcionamiento.

Este Trabajo Fin de Máster tiene diferentes finalidades. En primer lugar, se va a analizar y comprobar el funcionamiento de un amperímetro adquirido por el grupo GranaSAT, destinado a futuros experimentos que involucren aceleradores de partículas. Posteriormente, se va a realizar un rediseño de dicho amperímetro, en el que se evaluarán los requisitos y necesidades de los experimentos para implementar mejoras.

El trabajo se ha desarrollado con el grupo GranaSAT, en cuyos laboratorios se ha dispuesto de una gran variedad de instrumentos y equipos para realizar realizar las pruebas necesarias.

### 1.0.1. Objetivos

Los objetivos del trabajo se listan a continuación.

1. Análisis y comprobación del funcionamiento de un amperímetro adquirido por el grupo GranaSAT.

2. Rediseño del amperímetro a nivel de hardware y software, y posterior fabricación y montaje de la placa de circuito impreso y su carcasa.

3. Creación de librerías de Python que permitan comunicación y control de los amperímetros e instrumentos empleados en el desarrollo del trabajo.

### 1.0.2. Aceleradores de partículas

Los aceleradores de partículas son instrumentos que presentan una gran variabilidad en tamaño, forma y método de funcionamiento según su aplicación. Todos se basan en la interacción entre cargas eléctricas y campos electromagnéticos estáticos o dinámicos. Las frecuencias de funcionamiento de los campos magnéticos pueden ser tan bajas como la red eléctrica (50 Hz) o llegar hasta los MHz o GHz. Los aceleradores de partículas están formados por dos elementos principales: el inyector o fuente de partículas, y el acelerador principal [1].

Tras la aceleración, que puede ser lineal o circular, el haz se dirige hacia un blanco para estudiar las colisiones e interacciones de alta energía. Esto se denomina blanco fijo, y es la forma más básica de experimentación con aceleradores de partículas. Los impactos de partículas cargadas generan pequeñas

corrientes, del orden de miliamperios o menores, que circulan entre el acelerador y la fuente de alto voltaje que lo polariza.



El acelerador cuyas corrientes se desean obtener con este trabajo, pese a que aún no se encuentra instalado en el laboratorio de GranaSAT, va a estar alimentado a -100 kV, y con una corriente máxima de 6 mA, con un consumo máximo total de 600 W.

### 1.0.3. Medidor de corriente

Como se ha expuesto en el apartado anterior, el acelerador de partículas se encuentra polarizado a -100 kV. Por ello, es indispensable que el amperímetro sea flotante, es decir, que se encuentre aislado de la tierra junto con las partes en tensión del acelerador y al mismo potencial que estas.

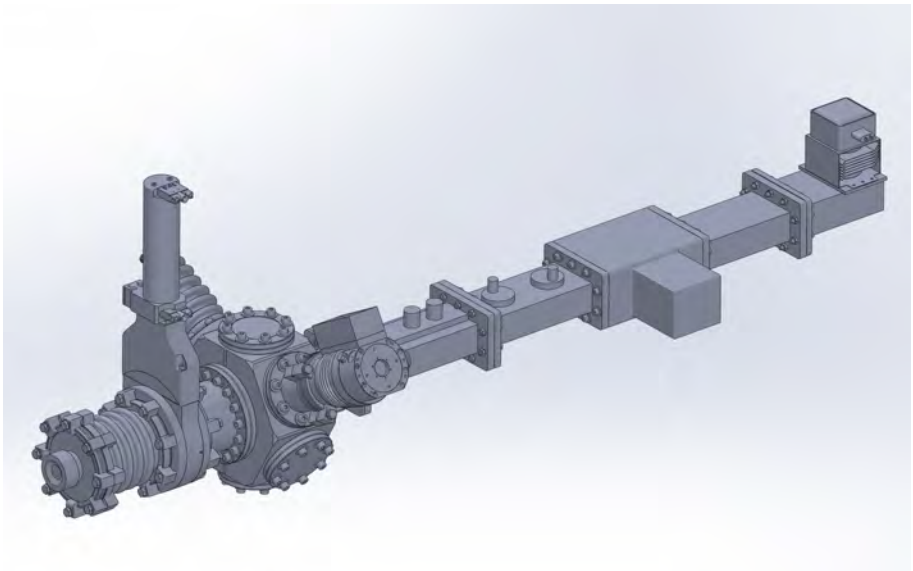


**Figura 1.1** – *Modelo 3D del acelerador de partículas que se va a instalar en GranaSAT*

Se desean obtener medidas de corriente con órdenes de magnitud entre nanoamperios y miliamperios. Esto influye en la elección de las resistencias internas del amperímetro, así como en el diseño del circuito analógico de amplificación y medición.

Los amperímetros empleados a lo largo de este trabajo utilizan para sus mediciones distintas resistencias con valores suficientemente pequeños que permitan obtener una pequeña diferencia de potencial en sus terminales sin perjudicar el funcionamiento normal de los instrumentos que se desean medir. Este voltaje posteriormente se amplifica, de forma que pueda ser medido correctamente por un conversor analógico-digital y procesado por un microcontrolador.

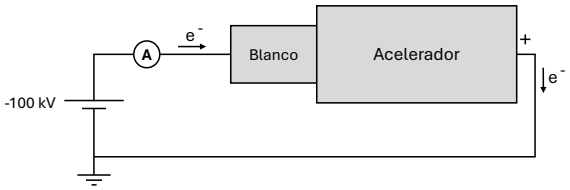



**Figura 1.2** – *Esquema de conexión del amperímetro con el sistema*

### 1.0.4. Amperímetro CurrentRanger R3

El CurrentRanger R3 es la tercera versión del amperímetro para pequeñas corrientes diseñado por LowPowerLab. Se trata de un amperímetro que puede medir corrientes con orden de magnitud entre nanoamperios y miliamperios, y que cuenta con diversas funciones como auto rango, medidas unidireccionales o bidireccionales y opciones para implementar batería y módulo de Bluetooth externo [2].

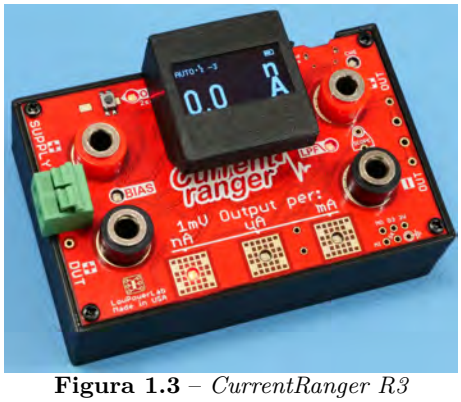



**Figura 1.3** – *CurrentRanger R3*

El CurrentRanger ha sido diseñado como un amperímetro económico, destinado tanto a mediciones para aplicaciones de aficionado como profesionales. Gracias a sus resistencias de precisión, con tolerancias menores al 0.5 %, permite obtener medidas de la corriente con gran exactitud, ya sea de forma digital mediante su microcontrolador como con conexión directa a osciloscopio.

Algunas de sus características más importantes se detallan a continuación:

- Utiliza el microcontrolador ATSAMD21G18 de Microchip.
- Cuenta con una pantalla OLED de 128 x 64 píxeles, controlada por I2C, donde se muestra información importante sobre el funcionamiento del sistema.
- En la salida puede activarse un filtro paso bajo para mejorar la medición.
- Opción de auto apagado.
- Utiliza dos amplificadores de precisión MAX4239 para acondicionar la medición al convertidor analógico-digital del microcontrolador.

El análisis detallado del funcionamiento de este dispositivo se explicará más adelante.

# Capítulo 2

# Requisitos

## 2.1. Hardware

| Ref. | Requerimientos |
|---|---|
| H.1 | Comunicación Bluetooth. |
| H.2 | Comunicación USB 2.0. |
| H.3 | Botón para encendido/apagado |
| H.4 | Pantalla OLED con interfaz I2C. |
| H.5 | Alimentación mediante USB-C (5 V). |
| H.6 | Alimentación mediante batería Li-Ion (3.7 V) y cargador USB-C. |
| H.7 | Medición de corrientes entre nA y mA. |
| H.8 | Protección contra alto voltaje de entrada. |
| H.9 | Conexiones para medir de forma externa el voltaje de salida. |

**Tabla 2.1** – *Requerimientos de hardware*

## 2.2. Firmware

| Ref. | Requerimientos |
|---|---|
| F.1 | Control y lectura por puerto serie. |
| F.2 | Control y lectura por Bluetooth. |
| F.3 | Buffer de mediciones para obtener señales de alta frecuencia. |
| F.4 | Almacenamiento de variables de funcionamiento en memoria no volátil. |
| F.5 | Librería de Python para controlar el dispositivo y descargar los valores de las mediciones. |

**Tabla 2.2** – *Requerimientos de firmware*

## 2.3. Mecánicos

| Ref. | Requerimientos |
|---|---|
| M.1 | Carcasa para proteger la PCB que facilite las conexiones y el manejo por el usuario. |
| M.2 | Conectores tipo banana para entrada de corriente y salida de voltaje. |
| M.3 | Conector USB-C |
| M.4 | Pantalla que muestre el estado del medidor. |
| M.5 | Tornillos M4 para sujeción de la PCB a la carcasa. |
| M.6 | Soporte para batería en la carcasa. |

**Tabla 2.3** – *Requerimientos mecánicos*

# Capítulo 3

# Análisis de un medidor de corriente comercial

En este apartado se va a analizar un medidor de corriente comercial adquirido por el grupo GranaSAT. Se trata del medidor CurrentRanger R3, mencionado anteriormente en la introducción de este trabajo. Es la tercera versión del medidor de pequeñas corrientes desarrollado por LowPowerLab.

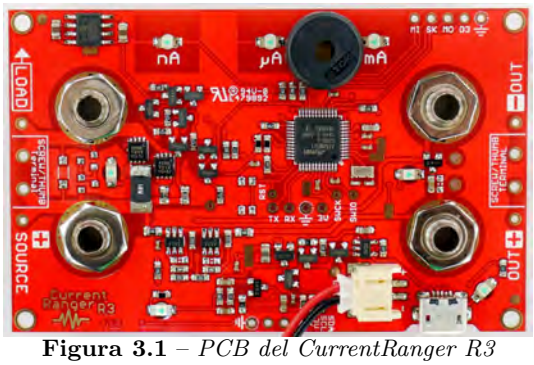



**Figura 3.1** – *PCB del CurrentRanger R3*

Se va a realizar una descripción de su hardware, tanto de forma teórica como mediante simulación con LTspice, así como un análisis de su código. Posteriormente, se va a evaluar su funcionamiento y se van a proponer e implementar algunas mejoras.

## 3.1. Análisis de hardware

El circuito del CurrentRanger R3 se divide en tres partes principales bien diferenciadas: alimentación, electrónica de acondicionamiento y microcontrolador. Además, hay otros elementos como LEDs, botones y pulsadores táctiles. El esquemático completo se puede visualizar en la figura 3.2. A continuación, se van a estudiar con detalle cada una de sus partes.

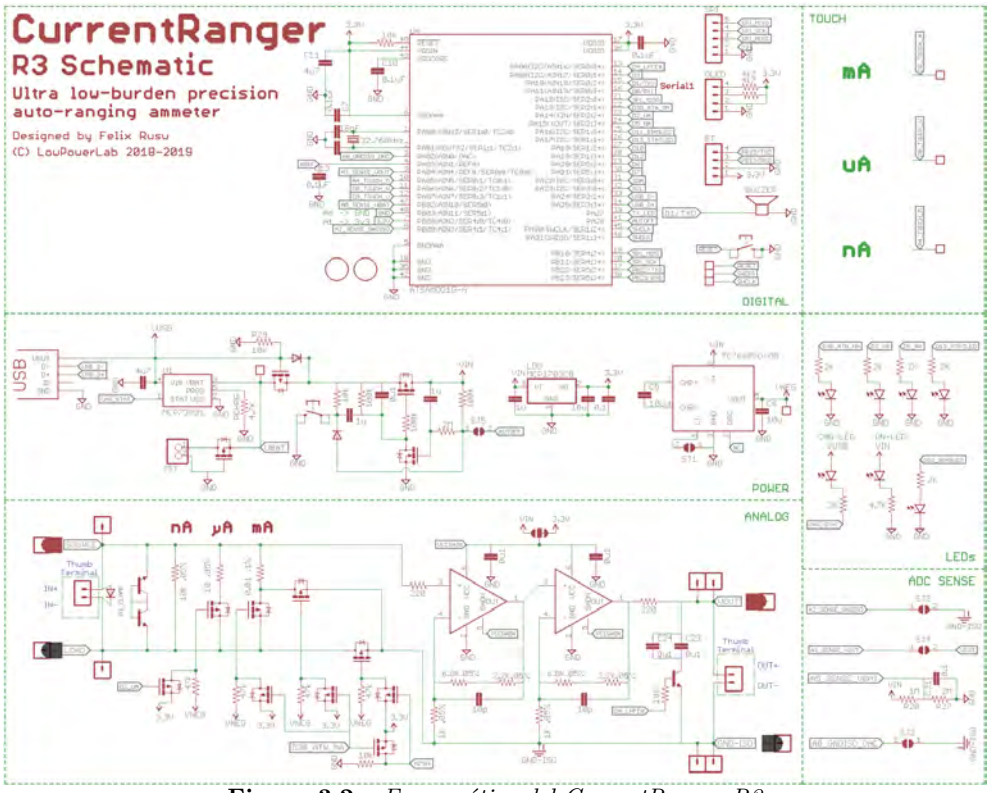



**Figura 3.2** – *Esquemático del CurrentRanger R3*

### 3.1.1. Alimentación

La alimentación del circuito puede realizarse mediante conexión por USB Micro-B o con una batería de litio de 3.7 V.

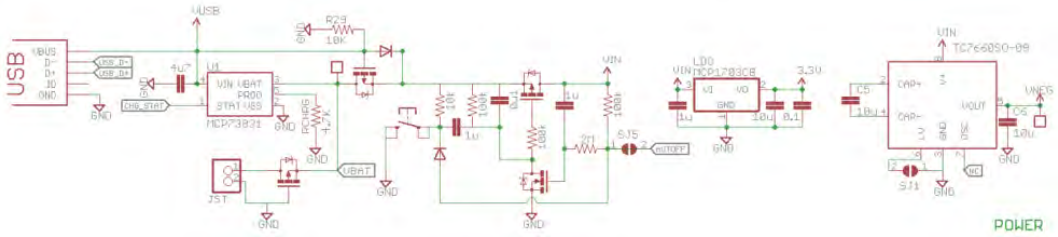



**Figura 3.3** – *Bloque de alimentación*

Si se alimenta con USB, el voltaje VUSB se puede usar para cargar la batería. Para ello, emplea el cargador MCP73831, de Microchip. Además de esto, un diodo permite la alimentación del circuito mediante el cable USB, pero impide el paso de corriente desde el resto del circuito hacia la línea USB cuando este está desconectado.

Junto al conector de la batería, se utiliza un PMOS que protege frente a una conexión inversa de la misma. Otro PMOS separa el circuito de carga de la batería de la línea de USB que alimenta al resto del circuito.

VIN se genera a partir de la batería o del USB. Para permitir el paso de la corriente o cortarla, el sistema cuenta con un circuito analógico de encendido/apagado, formado por un pulsador, 2 MOSFET y condensadores. Esto permite evitar el uso de un interruptor, lo que añade la ventaja de poder controlar el apagado desde software.

A partir de VIN se generan 3.3 V con el regulador MCP1703, de Microchip. Este regulador tiene una caída de voltaje de entre 0.5 y 0.7 V. También se genera, además, VNEG, con valor de -VIN, empleando para ello el TC7660, también de Microchip.

Un problema que presenta este circuito es que para generar 3.3 V es necesario un voltaje VIN superior a 3.8 V. Utilizando la alimentación por USB, alrededor de 5 V, esto no supone ningún problema, pero si se emplea una batería, esta debería estar completamente cargada. Esto es debido a que, cuando se descarga un poco la batería (el voltaje nominal suele ser de 3.7 V en baterías Li-Ion), el regulador deja de proporcionar el voltaje necesario. El funcionamiento del sistema depende del límite inferior de voltaje que puede soportar el microcontrolador que le permita funcionar correctamente.

### 3.1.2. Electrónica de acondicionamiento

El circuito analógico de acondicionamiento consiste principalmente en el empleo de tres resistencias de precisión (10 kΩ para medir nA, 10 Ω para medir $\mu$A y 0.01 Ω para medir mA) y dos amplificadores MAX4239, de Analog Devices, con ganancia de 10, colocados en cascada para conseguir una ganancia total de 100. Como principio general de funcionamiento, se hace circular la corriente que se desea medir por las resistencias para crear una pequeña caída de tensión que se amplifica para poder posteriormente ser medida por el microcontrolador. En función del orden de magnitud de la corriente, se cierran o se abren las ramas de las respectivas resistencias.

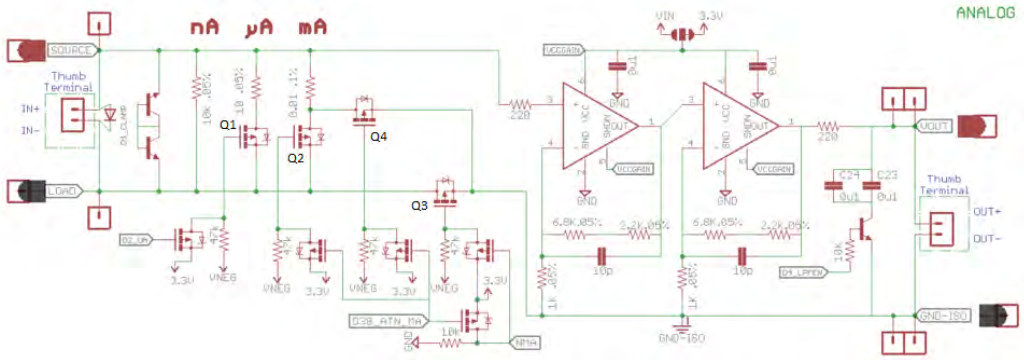



**Figura 3.4** – *Bloque de acondicionamiento analógico*

Por diseño, la rama con resistencia de 10 kΩ para medir nA siempre está cerrada. Esto no supone ningún problema, ya que si se quieren medir $\mu$A o mA, como las resistencias de sus ramas tienen valores mucho menores, apenas circulará corriente por la rama de nA y no influirá en la medida. Para medir nA y $\mu$A, el transistor Q3 está cerrado, permitiendo así el paso del voltaje. Si se quieren medir $\mu$A, se abren Q2 y Q4. Si se quieren medir mA, se abren Q1 y Q3, y se cierran Q2 y Q4. El propósito de Q4 y Q3 es el siguiente: como la resistencia de mA es tan pequeña (0.01 Ω), la resistencia que presenta Q2 durante la conducción afecta a la caída de voltaje. Es por eso que se cierra Q4 y se abre Q3, permitiendo así conocer y amplificar el voltaje exacto que cae en la resistencia de 0.01 Ω y no en el transistor Q2.

Los dos amplificadores de ganancia 10 se realizan con los integrados MAX4239 en configuración no inversora.

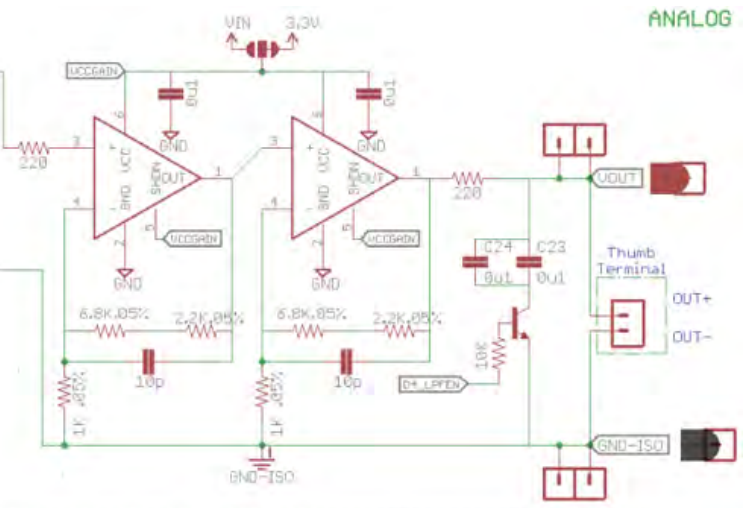



**Figura 3.5** – *Bloque amplificador*

GND-ISO tiene un voltaje de unos 30 mV, generado por el DAC del microcontrolador en el pin A0. Este voltaje actúa como referencia de la medida, consiguiendo así que para corrientes muy pequeñas el microcontrolador pueda leer de forma correcta el voltaje (ya que es posible que no se lean correctamente los valores cercanos a 0 V). Por ejemplo, con una corriente de 1 mA, el voltaje que cae en la resistencia es de 10 $\mu$V, que cuando se amplifica pasa a ser de 1 mV. Este voltaje es casi cero y el ADC del microcontrolador tiene problemas para leerlo, por lo que generando con el DAC el voltaje en GND-ISO, el voltaje a medir pasa a ser 1 mV + 30 mV, un valor más alto que el ADC ya sí puede leer.

### 3.1.3. Microcontrolador

El bloque digital del CurrentRanger R3 está basado en el microcontrolador ATSAMD21G18 [3]. Es un microcontrolador de Atmel, actualmente Microchip, con Cortex M0, 256 kB de memoria flash y 32 kB de memoria SRAM, frecuencia de hasta 48 MHz y comunicación USB, además de UART, SPI e I2C. Puede alimentarse entre 1.62 y 3.63 V.

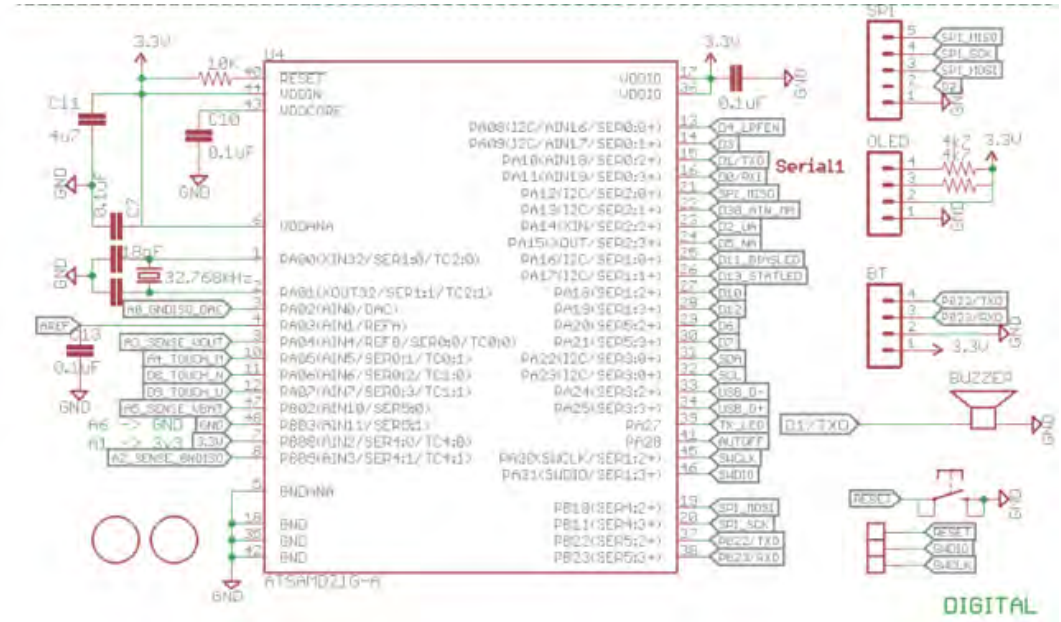



**Figura 3.6** – *Bloque del microcontrolador*

En el esquemático de la figura 3.6 se pueden observar todas las entradas, salidas y conexiones del microcontrolador con el resto de elementos del circuito. Es destacable la resistencia de *pull-up* conectada al pin RESET. Con un pulsador, cada vez que este pin se lleva a masa, se reinicia el programa almacenado en el microcontrolador. Además, se han colocado condensadores de estabilización de 100 nF junto a la alimentación, en los pines VDDIO. El microcontrolador también utiliza un oscilador externo con un cristal de cuarzo de 32.768 kHz, conectado entre los pines PA00 y PA01, junto con condensadores de 18 nF. Son destacables también los conectores de I2C para la pantalla OLED, así como los conectores para añadir un módulo de Bluetooth externo. Por último, es posible añadir un *buzzer* que actúa como alarma en algunas situaciones descritas en el código, el cual se explica en el siguiente apartado.

### 3.1.4. Simulaciones de la electrónica analógica

Las simulaciones del circuito de acondicionamiento se han realizado empleando el software LTspice. Para ello, se ha utilizado el esquemático de la figura 3.7.



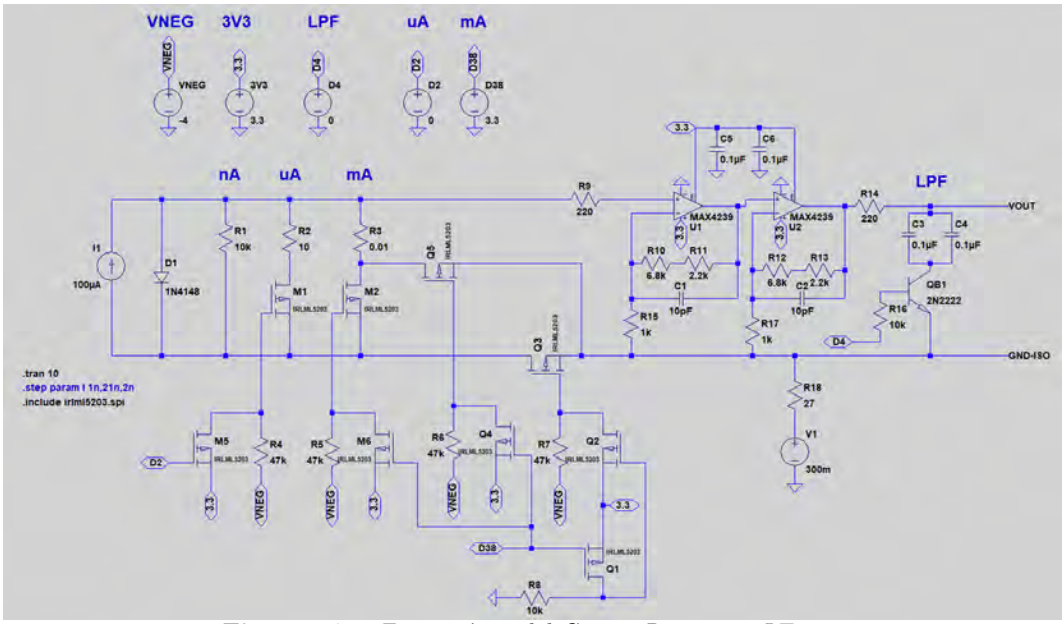



**Figura 3.7** – *Esquemático del CurrentRanger en LTspice*

Como se puede apreciar en el esquemático, se ha añadido a LTspice el modelo de los transistores IRLML5203. Estos transistores son PMOS con una resistencia en conducción muy reducida y que permiten una corriente de drenador de hasta 3 A, lo cual los hace adecuados para aplicaciones donde las caídas de voltaje deban ser muy reducidas o donde se necesiten corrientes elevadas [4].

Los pines del microcontrolador se han representado con fuentes de voltaje y sus respectivas etiquetas. Cuando el pin se encuentra en alto, se utiliza un valor de 3.3 V, y cuando está en bajo, de 0 V. Además, se han creado fuentes para la alimentación con valor de 3.3 V y para VNEG con un valor de -4 V.

La corriente entrante en el sistema se ha modelado mediante una fuente de corriente ideal. El DAC del microcontrolador, por su parte, se ha modelado con una fuente de voltaje conectada a GND-ISO por una resistencia de valor pequeño.

Si se realiza una primera simulación con una corriente de 100 $\mu$A, los resultados se pueden observar en la figura 3.8.

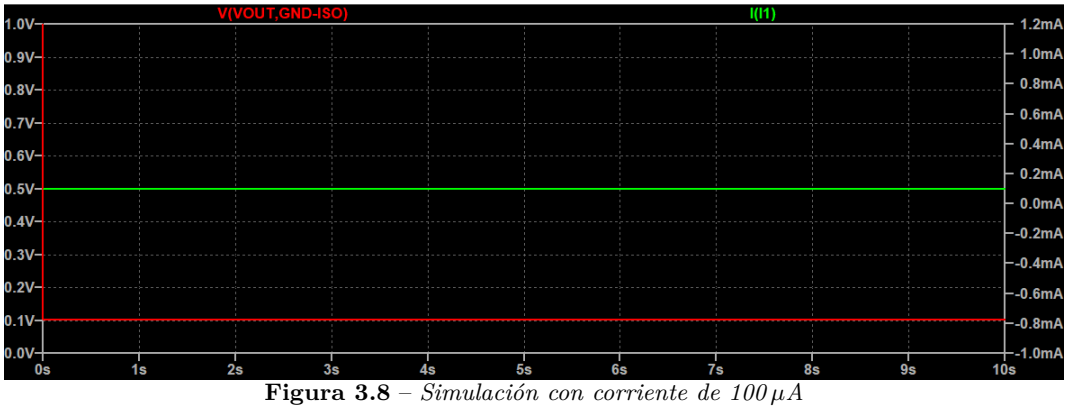



**Figura 3.8** – *Simulación con corriente de 100 µA*

Como se puede apreciar, el voltaje obtenido, en verde, es de 0.1 V para una corriente, en rojo, de 100 µA, cuando el pin de medición de µA se encuentra activo. Es decir, la conversión producida es de 1 mV/µA. Si se realiza ahora una simulación empleando un parámetro de escalón que aumente el valor de la corriente, se obtiene el resultado mostrado en la figura 3.9.

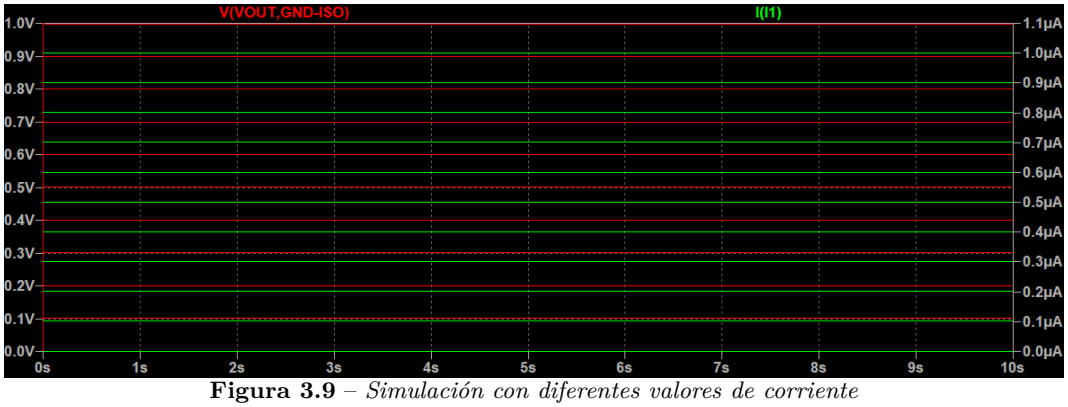



**Figura 3.9** – *Simulación con diferentes valores de corriente*

En este caso, la corriente ha variado desde 0 hasta 1000 nA en intervalos de 100 nA. El voltaje obtenido a la salida, con el pin de nA activo, es proporcional a este valor, variando desde 0 V hasta 1 V. Por último, se muestra un caso de corriente con rizado, cuyo valor medio es de 200 µA y su amplitud de 20 µA.

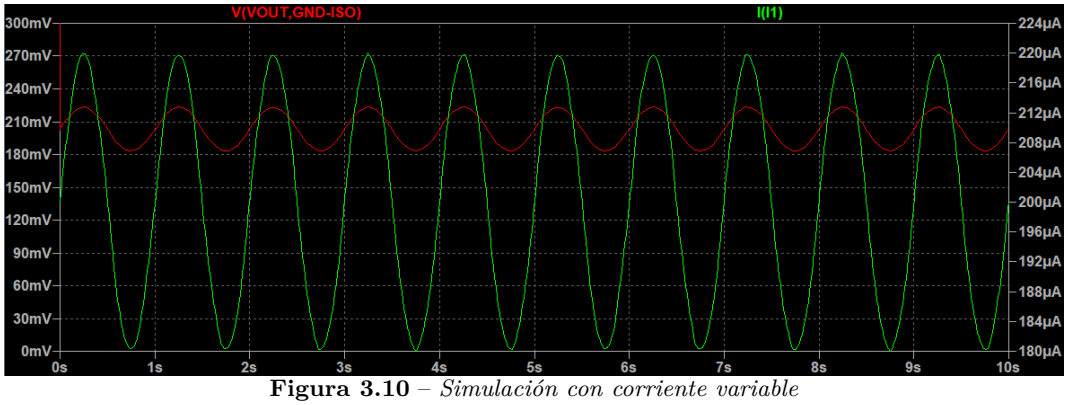



**Figura 3.10** – *Simulación con corriente variable*

Se observa que el voltaje en la salida va siguiendo el valor de la corriente de entrada.

Por lo tanto, con estas simulaciones se comprueba el funcionamiento teórico del circuito de acondicionamiento, tanto los elementos de selección de resistencia como los amplificadores en cascada.

## 3.2. Análisis de software

El código para el funcionamiento del CurrentRanger R3 ha sido proporcionado por el fabricante en Github, y permite su uso y modificación de manera libre. Está realizado con el entorno de desarrollo de Arduino. Por ello, el programa tiene la estructura de funciones de *setup* y *loop* típica de Arduino. Todas las variables y funciones están declaradas en un mismo archivo.

El código utiliza tres librerías externas: FlashStorage.h 1.0.0, de Arduino, para emular una memoria eeprom en el microcontrolador, Adafruit_FreeTouch.h 1.1.3, de Adafruit, para manejar los *pads* táctiles de la placa, y U8g2lib.h 2.34.22, de oliver, para comunicarse con la pantalla OLED. Tanto FlashStorage como Adafruit_FreeTouch son librerías específicas de los microcontroladores SAMD.

El diagrama de flujo del código se muestra en la figura 3.11.

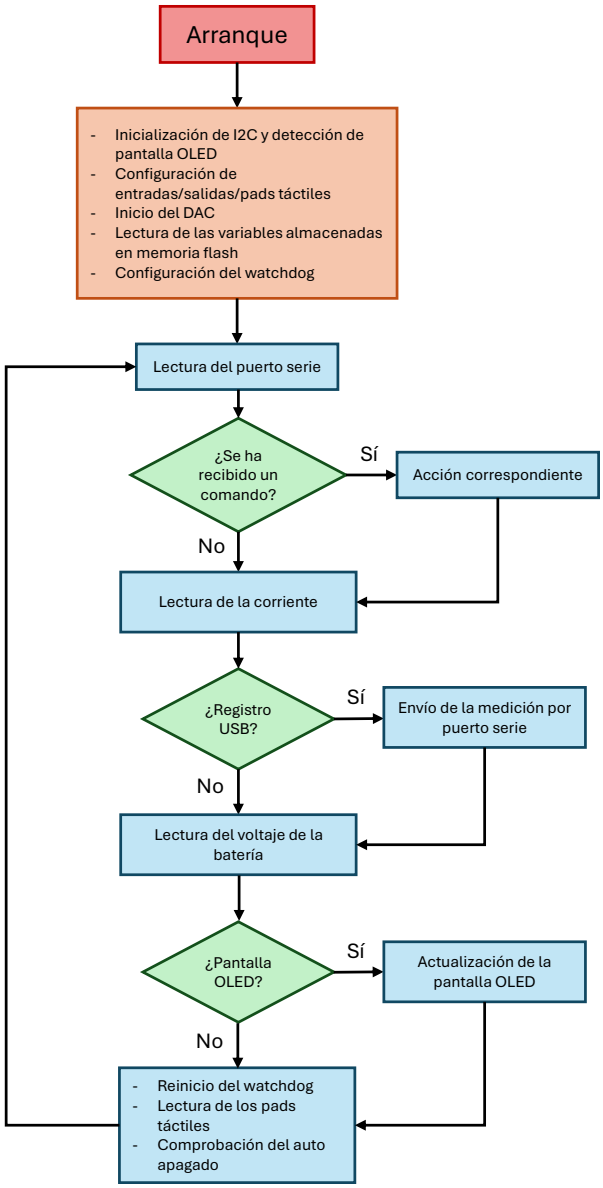



**Figura 3.11** – *Diagrama de flujo del código*

El CurrentRanger R3 cuenta con diferentes modos de funcionamiento, los cuales pueden ser cambiados por el usuario utilizando la comunicación por puerto serie mediante USB. Estos se resumen en la tabla 3.1.

| **Modo de operación** | **Descripción** |
|---|---|
| Miliamperios | Se mide la corriente en miliamperios. |
| Microamperios | Se mide la corriente en microamperios. |
| Nanoamperios | Se mide la corriente en nanoamperios. |
| Auto apagado | Puede ser default, smart o disabled. Default: se apaga tras 10 minutos de inactividad. Smart: Se apaga tras 10 minutos de inactividad si no está activo el registro de datos. Disabled: el dispositivo no se apaga. |
| Registro USB | Si se activa, se envían los datos de medición por puerto serie. |
| Registro Bluetooth | Si se activa, se envían los datos de medición por Bluetooth. |
| Formato de registro | Puede ser exponencial, nA, $\mu$A, mA o ADC. Exponencial: los datos se muestran en amperios, en potencias de 10. nA, $\mu$A, mA: los datos se muestran en número completo en función de la unidad seleccionada. ADC: se muestra el valor de 12 bits leído por el ADC. |
| Rango por GPIO | Se utilizan los pines SCK, MISO y MOSI para indicar la unidad de medida. SCK: mA. MISO: $\mu$A. MOSI: nA. |
| Reiniciar en *bootloader* | Reinicia el sistema para entrar en modo *bootloader*. |
| Velocidad de muestreo ADC | Puede ser *average* (media de 64 muestras), *fast* (media de 16) o *slow* (media de 256). |
| Información de *pads* táctiles | Si se activa, envía la medida de los *pads* táctiles por puerto serie. |
| Filtro paso bajo | Activa o desactiva el uso del LPF en la salida. |
| Corriente alterna | Eleva el voltaje de GND-ISO para permitir medir corriente alterna (positiva y negativa). |
| Auto rango | El dispositivo cambia automáticamente de rango para obtener la medida de corriente. |

**Tabla 3.1** – *Modos de operación del CurrentRanger R3*

En cuanto a los comandos de funcionamiento del Currentranger R3, estos se muestran en la tabla 3.2, así como su función y la información que devuelven.

| Comando | Descripción | Devuelve |
|---|---|---|
| a | Cambia modo de auto apagado. | Modo de auto apagado actual |
| b | Conmuta modo Bluetooth. | Aviso de activado/desactivado. |
| f | Cambia formato de datos impresos. | Formato actual. |
| g | Conmuta indicación de rango por GPIO. | Aviso de activado/desactivado. |
| r | Reinicia en *bootloader*. | Aviso de reinicio. |
| s | Cambia número de muestras del ADC. | Modo de ADC actual. |
| t | Conmuta información de depuración de los *pads* táctiles. | Aviso de activado/desactivado. |
| u | Conmuta modo USB. | Aviso de activado/desactivado. |
| < | Reduce valor del LDO (-1 mV). | Valor actual del LDO. |
| > | Aumenta valor del LDO (+1 mV). | Valor actual del LDO. |
| + | Aumenta valor de la ganancia (+1). | Valor actual de la ganancia. |
| - | Reduce valor de la ganancia (-1). | Valor actual de la ganancia. |
| * | Aumenta valor del *offset* (+1). | Valor actual del *offset*. |
| / | Reduce valor del *offset* (-1). | Valor actual del *offset*. |
| 1 | Rango en mA. | Nada. |
| 2 | Rango en $\mu$A. | Nada. |
| 3 | Rango en nA. | Nada. |
| 4 | Conmuta el LPF. | Nada. |
| 5 | Conmuta el modo BIAS. | Nada. |
| 6 | Conmuta el modo auto rango. | Nada. |
| ? | Imprime valores de calibración y menú de comandos. | Información del dispositivo y lista de comandos. |

**Tabla 3.2** – *Comandos del CurrentRanger R3*

## 3.3. Pruebas de funcionamiento

Una vez se han analizado el hardware y el software del CurrentRanger, se ha procedido a realizar las pruebas de funcionamiento del mismo.

### 3.3.1. Preparación de la PCB

En primer lugar, se ha soldado un conector hembra de cuatro pines en los agujeros correspondientes de la PCB para alojar la pantalla OLED, tal y como se muestra en la figura 3.12.

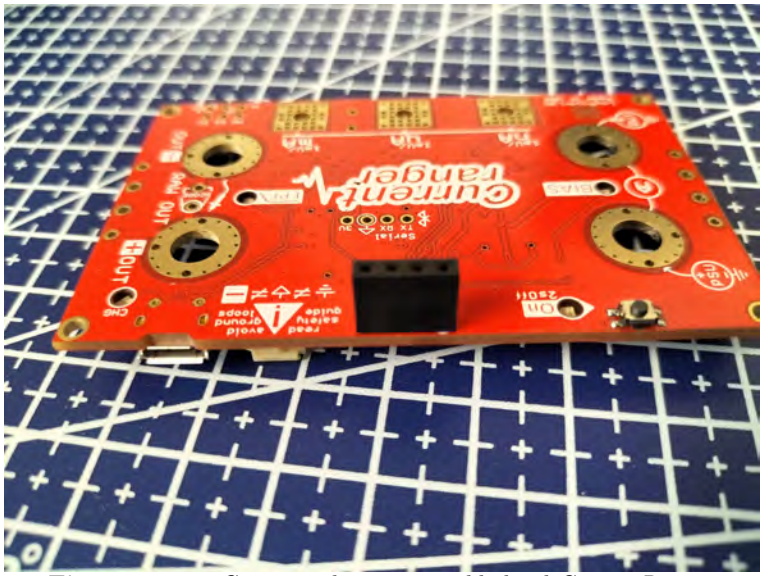


**Figura 3.12** – *Conector de 4 pines soldado al CurrentRanger*



A continuación, se han colocado los conectores tipo banana, los cuáles se sujetan con una tuerca, y se ha conectado la pantalla. Para comprobar el funcionamiento, se ha utilizado un cable USB Micro-B conectado a un ordenador y se ha pulsado el botón de encendido.

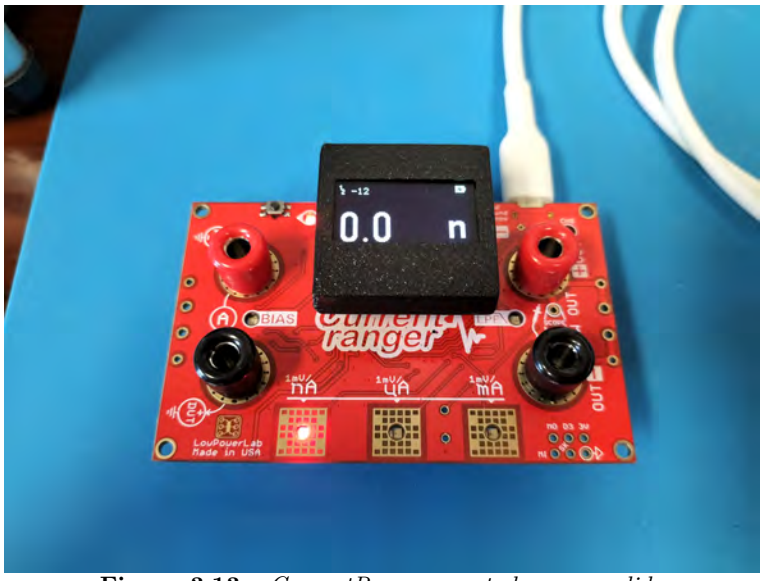



**Figura 3.13** – *CurrentRanger montado y encendido*

Se ha comprobado que, al tocar los *pads* táctiles, cambia el modo de medición, se enciende el LED correspondiente y se actualiza la pantalla. También se activan correctamente el LPF, el modo BIAS y el modo auto rango. Al mantener pulsado el botón de on/off durante más de dos segundos, el dispositivo se apaga.

También se ha llevado a cabo una prueba de comunicación por puerto serie, donde se han enviado todos los comandos y se ha recibido la respuesta del sistema. Para ello, se ha utilizado el entorno de desarrollo de Arduino y su monitor serial. Al ser comunicación directa por USB con el microcontrolador, sin necesidad de conversión a UART, no ha sido necesario configurar la velocidad de transmisión.

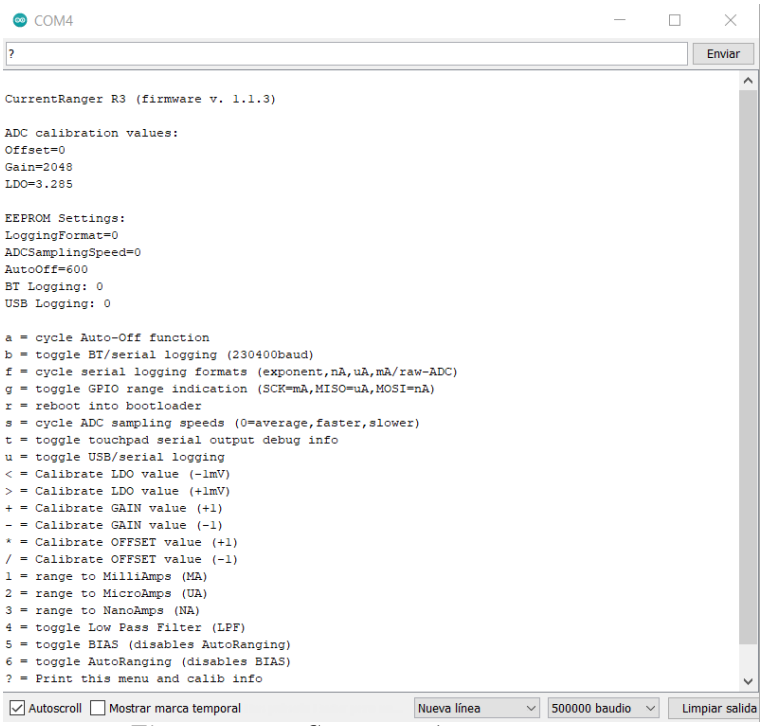



**Figura 3.14** – *Comunicación por puerto serie*

### 3.3.2. Medición de corriente

Para poder llevar a cabo las pruebas necesarias con el medidor de corriente, ha sido necesario configurar una fuente de corriente perteneciente al laboratorio de GranaSAT, la Keithley Model 220.


**Figura 3.15** – *Fuente de corriente Keithley 220*

La fuente de corriente permite configurar hasta 100 posiciones de memoria con valor de corriente entre 2 nA y 100 mA, límite de tensión entre 1 V y 105 V y tiempo de permanencia entre 3 ms y 999.9 s. Para ello se utilizan los botones presentes en el panel frontal, en la zona DISPLAY. Además, cuenta con tres modos de funcionamiento diferentes:

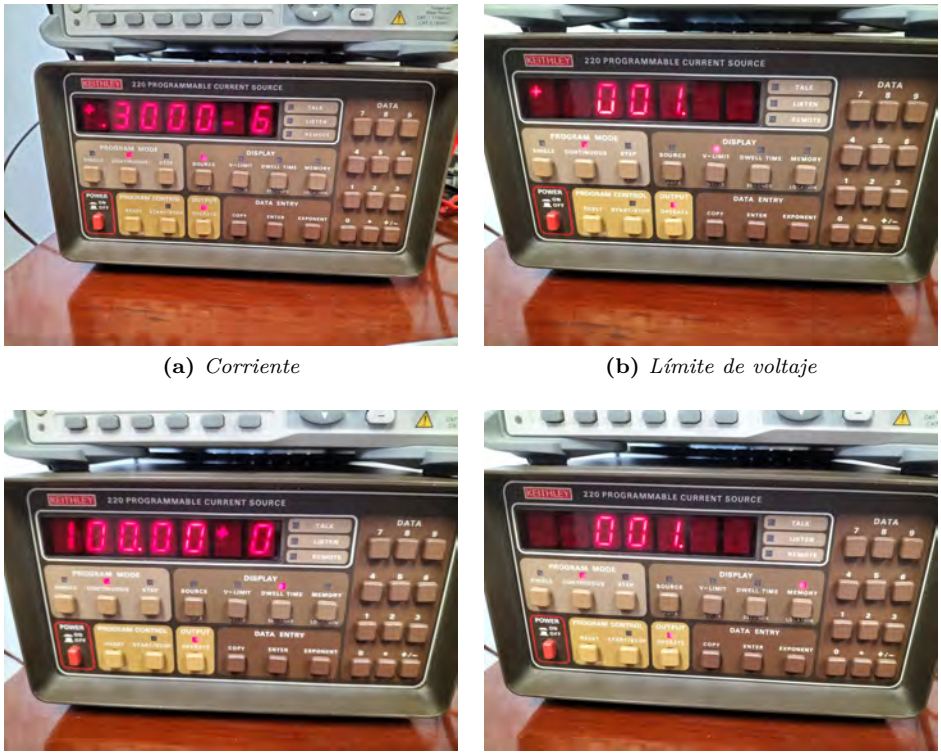
- SINGLE: El programa pasa una vez por cada posición de memoria y no reinicia al finalizar.

- CONTINUOUS: El programa pasa por cada posición de memoria y reinicia al final. De esta forma, se pueden programar formas de onda de la corriente que funcionen continuamente.

- STEP: Se debe pulsar el botón START/STOP para avanzar a la siguiente posición de memoria.

Para que la fuente emita la corriente programada, es necesario pulsar el botón OPERATE y que comprobar que se enciende el LED que hay sobre él. Este botón representa un mecanismo de seguridad para el usuario, ya que permite cortar la corriente para poder manipular los cables sin peligro.

En la figura 3.16 se muestra un ejemplo de programación manual de la fuente de corriente. La opción activa se muestra con el LED situado sobre los botones de DISPLAY. En este caso, se ha configurado la posición de memoria 1 con una corriente de 300 nA, un límite de voltaje de 1 V y un tiempo de 100 s.




**(a)** *Corriente* **(b)** *Límite de voltaje*

**(c)** *Tiempo de permanencia* **(d)** *Posición de memoria*

**Figura 3.16** – *Configuración de la fuente de corriente Keithley 220*

El cable utilizado para dar salida a la corriente generada por la fuente, mostrado en la figura 3.17, consiste en un cable triaxial. Cuenta con polo positivo, en rojo, otro negativo, en negro, y tierra de protección, en verde.

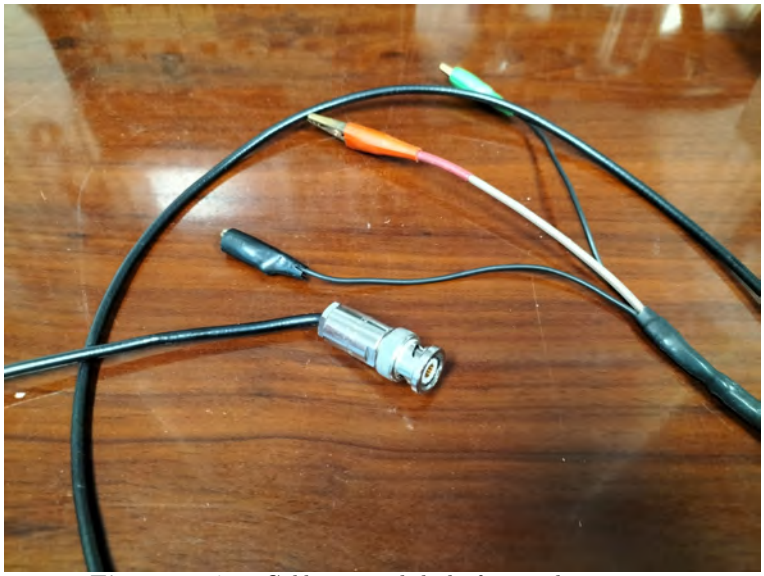


**Figura 3.17** – *Cable triaxial de la fuente de corriente*

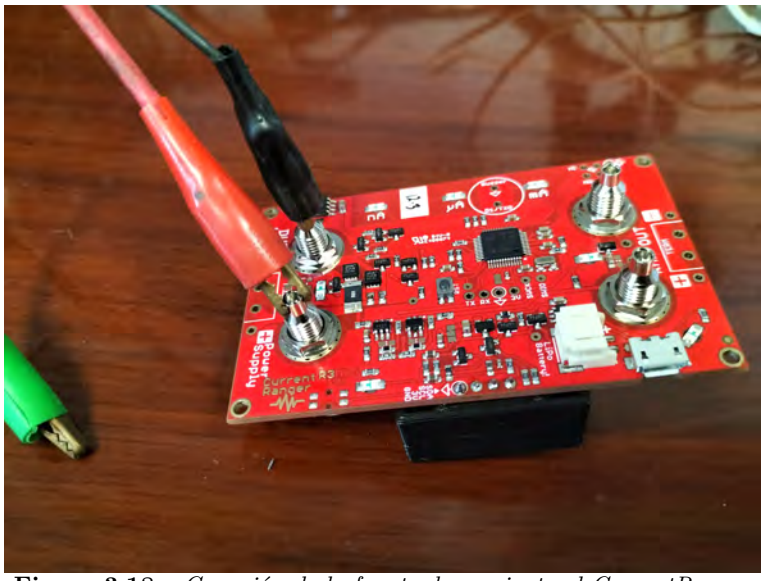


**Figura 3.18** – *Conexión de la fuente de corriente al CurrentRanger*

Realizando una primera prueba con una corriente fija de 100 mA, en la figura 3.19 se puede observar que el medidor proporciona la medida correcta con una variación muy pequeña.

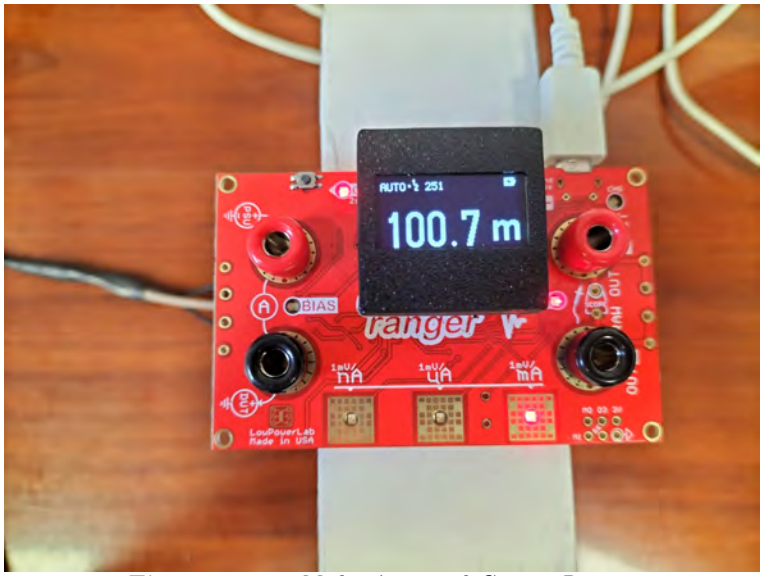



**Figura 3.19** – *Medición con el CurrentRanger*

El CurrentRanger R3 permite realizar una calibración variando algunos parámetros internos de funcionamiento, los cuales se almacenan en la memoria no volátil y son leídos siempre que se inicia el programa. Para optimizar estos valores, se han realizado una serie de pruebas con distintos valores hasta obtener la mayor exactitud posible en las medidas. En la tabla 3.3 se observan los resultados finales para mediciones de corrientes de distintos órdenes de magnitud y sus respectivos errores.

| Valor real de la corriente | Rango de mediciones | Precisión | Tolerancia | Error absoluto medio | Error relativo medio |
|---|---|---|---|---|---|
| 100 nA | [36.7 , 169.2] nA | 102.95 ± 66.2 nA | 64.30% | 2.95 | 2.90% |
| 500 nA | [474.3 , 586.2] nA | 525.2 ± 60.9 nA | 11.60% | 25.2 | 5.04% |
| 50 µA | [49.2, 50.4] µA | 49.8 ± 0.6 µA | 1.20% | 0.2 | 0.40% |
| 100 µA | [99.4 , 101.2] mA | 100.3 ± 0.9 µA | 0.89% | 0.3 | 0.30% |
| 500 µA | [499.2 , 501.2] µA | 500.2 ± 1 µA | 0.20% | 0.2 | 0.04% |
| 10 mA | [9.2 , 10] mA | 9.6 ± 0.4 mA | 4.16% | 0.4 | 4% |
| 50 mA | [49.7, 50.9] mA | 50.3 ± 0.6 mA | 1.19% | 0.3 | 0.60% |
| 100 mA | [99.1 , 100.7] mA | 99.9 ± 0.8 mA | 0.80% | 0.1 | 0.10% |

**Tabla 3.3** – *Pruebas con el CurrentRanger*

De la tabla 3.3 se puede extraer que las mediciones para nanoamperios son muy imprecisas, y, aunque el valor medio de las mismas está cerca del valor real, su tolerancia es muy elevada. Esto es debido principalmente al ruido electromagnético, el cual afecta enormemente a la medida por tratarse de corrientes muy pequeñas. El ruido procede principalmente de los cables de alimentación, de otros equipos cercanos e incluso de la red eléctrica, por lo que, para mejorar la precisión, hay que aislar el sistema todo lo posible frente a dichas interferencias. Los resultados para las medidas de microamperios y miliamperios son satisfactorios, pues tienen una gran precisión. Sin embargo, también es notable cómo la precisión disminuye para corrientes más pequeñas en ambos rangos.

Además de la configuración manual, que puede ser tediosa si se desean programar un gran número de posiciones de memoria, el fabricante pone a disposición de los usuarios un manual de programación para hacer este proceso desde el ordenador. Para realizar la conexión, la fuente cuenta con una interfaz IEEE 488, también conocida como GPIB.


**Figura 3.20** – *Interfaz GPIB (IEEE 488) de la fuente de corriente*

Para poder comunicarse con la fuente desde el ordenador, se ha utilizado un adaptador de GPIB a USB, mostrado en la figura 3.21.

**Figura 3.21** – *Adaptador GPIB a USB*

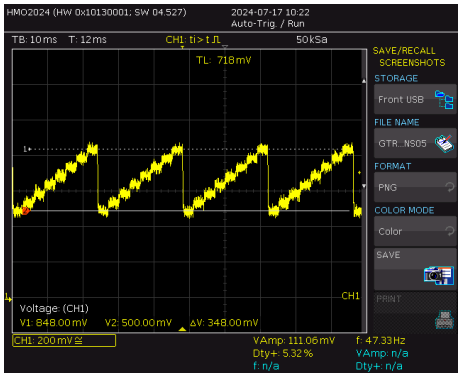
Y, además, ha sido necesario instalar el software Connection Expert y los drivers VISA, de Keysight, para poder establecer la comunicación desde el ordenador [5]. Una vez instalado, se ha creado una librería de Python para permitir la comunicación con la fuente de corriente y poder programarla utilizando un *script*.

Para comprobar el funcionamiento de la fuente, se ha utilizado el osciloscopio HMO2024, de ROHDE & SCHWARZ, el cual se ha conectado, con una misma sonda, a los conectores del voltaje de salida: OUT+ y OUT-.

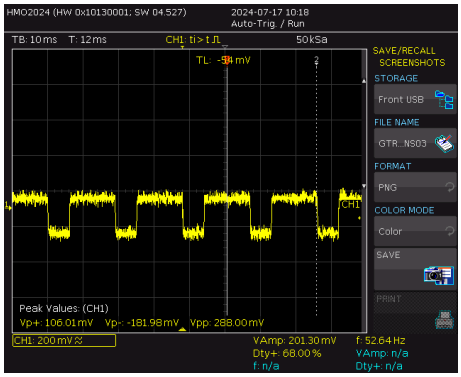

**Figura 3.22** – *Osciloscopio HMO2024*



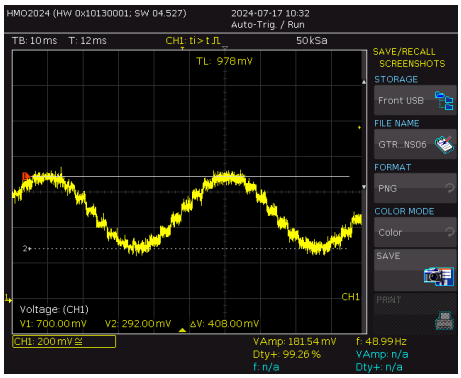
En la figura 3.18 se muestra la conexión entre la fuente de corriente y el CurrentRanger, y en la figura 3.23 se muestra el voltaje medido con el osciloscopio en la salida.




**(a)** *Onda de sierra*




**(b)** *Onda cuadrada*




**(c)** *Sinusoidal*

**Figura 3.23** – *Voltajes medidos con osciloscopio a la salida del CurrentRanger*

En la librería de Python, además de los comandos necesarios para programar cada posición de memoria,

se han creado funciones para representar los tres tipos de ondas mostrados en la figura 3.23, donde se puede configurar el valor medio, la amplitud y la frecuencia, y el ciclo de trabajo en el caso de la señal cuadrada. Los pequeños escalones que se aprecian en las ondas se corresponden a la duración mínima de una prosición de memoria, que, como se ha comentado anteriormente, es de 3 ms. Por ello, para crear las distintas formas de onda, ha sido necesario emplear corrientes continuas con duraciones muy cortas y distintos valores en función de su posición de memoria. El ruido observado se debe a que las corrientes son muy pequeñas, del orden de microamperios, por lo que cualquier pequeña perturbación es también amplificada.

El siguiente paso es comprobar si el microcontrolador es capaz de leer estos valores. El código original del CurrentRanger está diseñado para medir solo valores continuos o con muy baja frecuencia, ya que realiza una media de los últimos valores medidos, por lo que para poder obtener mediciones de señales de mayor frecuencia es necesario implementar mejoras en el firmware del sistema.

## 3.4. Mejoras del firmware



Para permitir a los usuarios actualizar libremente el firmware del sistema, el diseñador envía el producto con un *bootloader* UF2, el cual permite, mediante conexión USB, obtener el firmware actual y reemplazarlo copiando y pegando el nuevo archivo desde el ordenador. Además, también se puede realizar la programación desde el IDE de Arduino, utilizando simplemente el botón "Subir".

Los cambios realizados sobre el firmware han sido principalmente dos: cambio de nombre de los comandos para hacerlos más inteligibles y poder comunicarse mediante una librería de Python, y modificación de la velocidad de medida para obtener señales con mayor frecuencia.

### 3.4.1. Copia de seguridad del firmware

Para copiar el firmware actual hay que conectar mediante USB el CurrentRanger al ordenador y enviar el comando “r” para reiniciar en modo de *bootloader*.

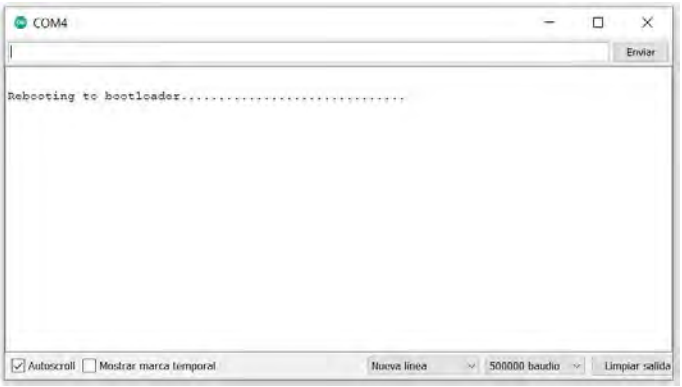



**Figura 3.24** – *Reinicio del CurrentRanger en bootloader*

Después del reinicio, se abre una ventana que da acceso a la memoria interna del CurrentRanger, donde se puede encontrar el programa actual y hacer una copia de seguridad. También se puede subir un programa con formato UF2 de esta forma, simplemente copiando y pegando.

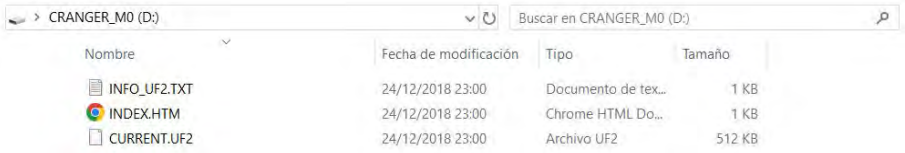



**Figura 3.25** – *Memoria interna del CurrentRanger*

### 3.4.2. Cambio de nombre y adición de nuevos comandos

Como se expuso en la tabla 3.2, los comandos del CurrentRanger R3 pueden resultar confusos a la hora de recordarlos o entenderlos. Por ello, se han renombrado y se han añadido nuevos comandos que faciliten el uso del sistema con una librería de Python.

Los nuevos comandos del CurrentRanger se muestran y explican a continuación.

| Comando | Descripción | Devuelve |
|---|---|---|
| INFO | Imprime la información del sistema y los parámetros almacenados. | Nombre del dispositivo, versión, parámetros almacenados y lista de todos los comandos. |

**Tabla 3.4** – *Comando de información del CurrentRanger R3*

El comando "INFO" se utiliza para comprobar que hay comunicación con la placa, así como para conocer la versión del firmware, parámetros internos almacenados y una lista con todos los comandos disponibles del sistema.

| Comando | Descripción | Devuelve |
|---|---|---|
| BOOTLOADER | Reinicia el sistema en modo *bootloader* | "Rebooting to bootloader" |

**Tabla 3.5** – *Comando para bootloader del CurrentRanger R3*

Con el comando BOOTLOADER, el CurrentRanger se reinicia y entra en modo *bootloader*, lo cual permite acceder a su memoria interna y actualizar o copiar su firmware.

| Comando | Descripción | Devuelve |
|---|---|---|
| NA | Establece rango de medida en nA. | "RANGE: nA" |
| UA | Establece rango de medida en $\mu$A. | "RANGE: uA" |
| MA | Establece rango de medida en mA. | "RANGE: mA" |

**Tabla 3.6** – *Comandos de unidades del CurrentRanger R3*

Los comandos NA, UA y MA se utilizan para determinar el rango de medición del sistema. Son equivalentes a pulsar el correspondiente *pad* táctil. Devuelven un mensaje que confirma el cambio de rango de medida.

| Comando | Descripción | Devuelve |
|---|---|---|
| ACMEAS | Hace mediciones continuas hasta que el búfer se llene e imprime el tiempo que ha tardado entre la primera y última medición. | Tiempo entre primera y última medición. |
| ACBUFFER | Si está activo el registro por USB o Bluetooth, imprime el búfer AC completo. | 30k valores de voltaje en mV. |
| DCMEAS | Si está activo el registro por USB o Bluetooth, imprime el voltaje actual. | Voltaje actual de la salida en mV. |
| CONTDC:ON | Si está activo el registro por USB o Bluetooth, imprime el voltaje de salida continuamente. | Valor del voltaje cada 100 ms. |
| CONTDC:OFF | Si está activo el registro por USB o Bluetooth, imprime el voltaje de salida continuamente. | Valor del voltaje cada 100 ms. |

**Tabla 3.7** – *Comandos de medición del CurrentRanger R3*

El comando DCMEAS realiza una medición de la corriente e imprime su valor por puerto serie. ACMEAS realiza 5000 mediciones a la máxima velocidad posible y almacena todos los valores en un búfer. Se ha elegido este número por haber suficiente espacio en la memoria del microcontrolador para poder almacenar el búfer. Realizando pruebas con un mayor tamaño, se ha comprobado que se producían errores en el funcionamiento del sistema. CONTDC:ON y CONTDC:OFF activan y desactivan, respectivamente, una impresión constante por puerto serie de los valores DC medidos.

| Comando | Descripción | Devuelve |
|---|---|---|
| VBAT | Si está activo el registro por USB o Bluetooth, imprime el voltaje de la batería en V. | Último voltaje medido de la batería en V. |

**Tabla 3.8** – *Comando de batería del CurrentRanger R3*

El comando VBAT imprime, si estána activos los registros por USB o Bluetooth, el voltaje actual de la batería en voltios. Es una variable de tipo *float*. El voltaje de la batería se actualiza cada 5 segundos.

| Comando | Descripción | Devuelve |
|---|---|---|
| GAIN+ | Aumenta el valor de corrección de ganancia en uno. | Valor actual de la corrección de ganancia. |
| GAIN- | Disminuye el valor de corrección de ganancia en uno. | Valor actual de la corrección de ganancia. |
| OFFSET+ | Aumenta el valor de corrección de offset en uno. | Valor actual de la corrección de offset. |
| OFFSET- | Disminuye el valor de corrección de offset en uno. | Valor actual de la corrección de offset. |
| LDO+ | Aumenta el valor del voltaje del LDO. | Valor actual del voltaje del LDO. |
| LDO- | Disminuye el valor del voltaje del LDO. | Valor actual del voltaje del LDO. |

**Tabla 3.9** – *Comandos de calibración del CurrentRanger R3*



Los comandos GAIN+ y GAIN- se utilizan para configurar el ADC interno del microcontrolador SAMD21 y aumentar la precisión en las medidas. OFFSET+ y OFFSET-, por su parte, varían el resultado calculado de la medición de corriente, permitiendo así ajustar el valor medido sin necesidad de afectar al funcionamiento del ADC. Por último, LDC+ y LDO- se utilizan para variar el valor almacenado que representa el voltaje del regulador de 3.3 V, ya que este no es completamente exacto y debe ser medido de forma externa. Esto se utiliza para la conversión entre la medición del ADC y el valor final de la corriente.

| Comando | Descripción | Devuelve |
|---|---|---|
| USB:ON | Activa registro de datos por USB. | "USB_LOGGING_ENABLED" |
| USB:OFF | Desactiva registro de datos por USB. | "USB_LOGGING_DISABLED" |
| BT:ON | Activa registro de datos por Bluetooth. | "BT_LOGGING_ENABLED" |
| BT:OFF | Desactiva registro de datos por Bluetooth. | "BT_LOGGING_DISABLED" |

**Tabla 3.10** – *Comandos de comunicación del CurrentRanger R3*

USB:ON y USB:OFF activan y desactivan el registro de datos por USB, mientra que BT:ON y BT:OFF realizan la misma función para el registro por Bluetooth. Cuando se encuentran activos, el sistema puede enviar los valores numéricos obtenidos de las mediciones de corriente, batería o depurado de los *pads* táctiles.

| Comando | Descripción | Devuelve |
|---|---|---|
| TOUCH:ON | Activa información de depuración de los pads táctiles. | "TOUCH_DEBUG_ENABLED" y los valores de los pads táctiles de forma continua. |
| TOUCH:OFF | Desactiva información de depuración de los pads táctiles. | "TOUCH_DEBUG_DISABLED" |

**Tabla 3.11** – *Comandos de depuración de pads táctiles del CurrentRanger R3*

TOUCH:ON y TOUCH:OFF activan y desactivan el envío por USB o Bluetooth de los valores leídos en los *pads* táctiles, con el objetivo de ajustar sus valores para optimizar el funcionamiento y corregir errores.

| Comando | Descripción | Devuelve |
|---|---|---|
| FORMAT:EXP | Formato de registro exponencial. | “LOGGING_FORMAT_EXPONENT” |
| FORMAT:NANOS | Formato de registro en nA. | “LOGGING_FORMAT_NANOS” |
| FORMAT:MICROS | Formato de registro en $\mu$A. | “LOGGING_FORMAT_MICROS” |
| FORMAT:MILLIS | Formato de registro en mA. | “LOGGING_FORMAT_MILLIS” |

**Tabla 3.12** – *Comandos de formato del CurrentRanger R3*

FORMAT:EXP hace que los datos medidos se envíen en amperios con formato exponencial. FORMAT:NANOS, FORMAT:MICROS y FORMAT:MILLIS hacen que los datos se envíen en nanoamperios, microamperios y miliamperios, respectivamente.



| Comando | Descripción | Devuelve |
|---|---|---|
| ADC:AVG | ADC promedio de 64 muestras. | “ADC_SAMPLING_SPEED_AVG” |
| ADC:FAST | ADC promedio de 16 muestras. | “ADC_SAMPLING_SPEED_FAST” |
| ADC:SLOW | ADC promedio de 256 muestras. | “ADC_SAMPLING_SPEED_SLOW” |

**Tabla 3.13** – *Comandos de velocidad del CurrentRanger R3*

ADC:AVG establece la media de medidas del ADC en 64 muestras antes de proporcionar un resultado. ADC:FAST realiza la media de 16 muestras, y ADC:SLOW realiza la media de 256 muestras. Cuanto mayor es el número de muestras, más lenta se realiza la medición.

| Comando | Descripción | Devuelve |
|---|---|---|
| AUTOFF:ON | Activa modo de auto apagado. | “AUTOOFF_ENABLED” |
| AUTOFF:OFF | Desactiva modo de auto apagado. | “AUTOOFF_DISABLED” |
| LPF:ON | Activa el filtro paso bajo en la salida. | “LPF ON” |
| LPF:OFF | Desactiva el filtro paso bajo en la salida. | “LPF OFF” |
| BIAS:ON | Activa el modo BIAS. | “BIAS ON” |
| BIAS:OFF | Desactiva el modo BIAS. | “BIAS OFF” |
| AUTORANGE:ON | Activa el modo auto rango. | “AUTORANGE ON” |
| AUTORANGE:OFF | Desactiva el modo auto rango. | “AUTORANGE OFF” |

**Tabla 3.14** – *Comandos de funcionalidades del CurrentRanger R3*

Los comandos de AUTOFF:ON y AUTOFF:OFF activan o desactivan el modo de auto apagado, el cual tiene una duración de 10 minutos. LPF:ON y LPF:OFF activan o desactivan el filtro paso bajo del voltaje de salida. Este filtro es útil para medir corrientes continuas, pero debe desactivarse si se quieren medir variaciones de mayor frecuencia. BIAS:ON y BIAS:OFF activan y desactivan el modo BIAS, en el que se eleva el valor de referencia de GND-ISO para poder medir corrientes tanto positivas como negativas. AUTORANGE:ON y AUTORANGE:OFF activan y desactivan el modo de autorango, en el que el medidor trata de situarse en el rango correcto de medición de la corriente según su valor.

### 3.4.3. Obtención de medidas a la máxima frecuencia

Como se ha expuesto en el apartado anterior, se ha modificado el programa y se han creado comandos para poder medir las variaciones de alta frecuencia de la corriente.

En primer lugar, se ha creado una variable búfer para almacenar 5000 valores, y se ha configurado el ADC para medir a la máxima velocidad hasta llenar el búfer. Utilizando el comando ACMEAS, se mide la corriente 5000 veces y se envía el tiempo que ha tardado por el puerto serie.

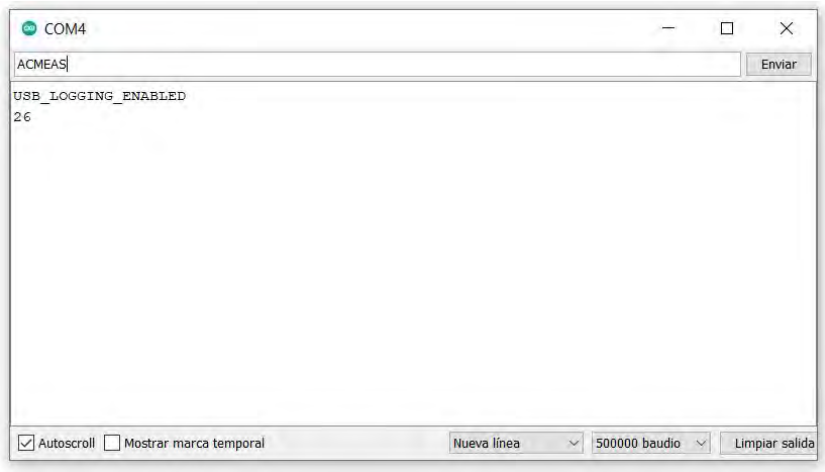



**Figura 3.26** – *Comando ACMEAS*

El tiempo que ha tardado en realiza las 5000 medidas ha sido de 26 ms, por lo tanto, la frecuencia de muestreo es de alrededor de 192000 muestras por segundo, lo que indica, según el teorema de muestreo de Nyquist, que se pueden medir señales con frecuencia hasta 100 kHz. Para obtener los valores, solo hay que utilizar el comando ACBUFFER.

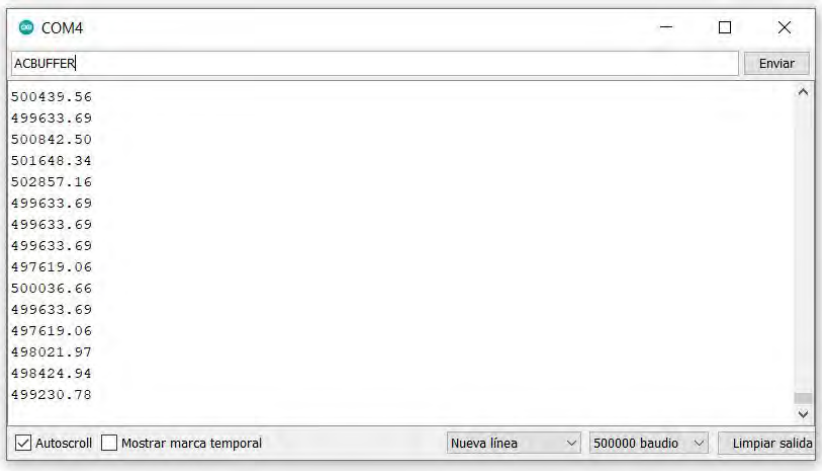



**Figura 3.27** – *Comando ACBUFFER*

Como se observa en la figura 3.27, se imprimen por puerto serie todos los valores del búfer uno a uno. Esto no es muy útil para realizar un estudio, por lo que se ha creado una librería de Python para enviar los comandos, almacenar estos valores en un vector y representarlos gráficamente, además de hacer la FFT para comprobar qué frecuencias tienen las interferencias de la señal.

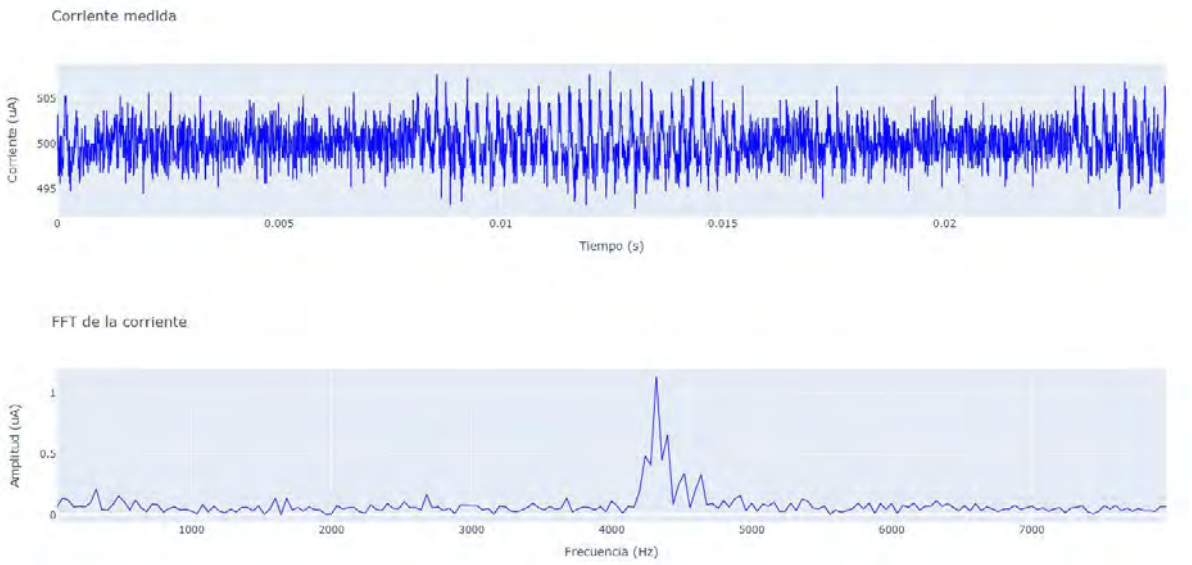



**Figura 3.28** – *Corriente medida y FFT*

En la figura 3.28 se puede apreciar que el sistema permite medir frecuencias superiores a 1 kHz. Es notable la interferencia con frecuencia de 4.3 kHz, que, aunque su amplitud no es muy grande, destaca frente al resto de frecuencias representadas. El origen de esta interferencia es posible que se deba al TC7660, utilizado para generar voltajes negativos, que opera a frecuencias cercanas a los 10 kHz y afecta a la polarización de los transistores de selección de rango.

Si se configura la fuente de corriente para generar una señal de diente de sierra entre 350 $\mu$A y 650 $\mu$A a 50 Hz y se realiza una medición con el CurrentRanger, el resultado representado con Python se muestra en la figura 3.29.

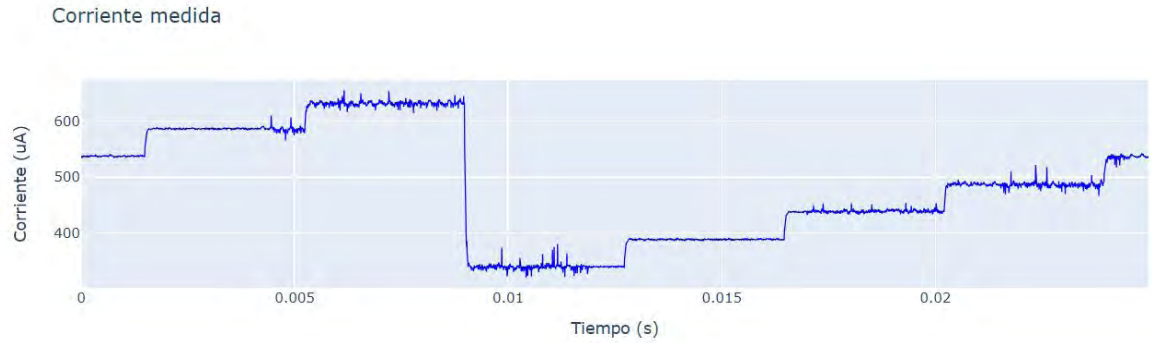



**Figura 3.29** – *Medición con forma de onda de sierra*

Se puede observar perfectamente la forma de onda de diente de sierra realizada con distintos valores de corriente continua de 3 ms cada uno, así como el ruido presente en la señal.

Por lo tanto, gracias a los cambios implementados en el código, se ha confirmado la capacidad del CurrentRanger de medir distintas formas de onda y obtener componentes alternas hasta una frecuencia de 100 kHz.

# Capítulo 4

# Rediseño del medidor de corriente

Una vez se ha estudiado el CurrentRanger R3 con profundidad, se han visto sus distintas funcionalidades y se han realizado mejoras en el firmware, se ha procedido con el rediseño del medidor para intentar mejorar aún más sus prestaciones. Para ello, y como parte principal del sistema, se ha decidido emplear un ESP32 gracias a su versatilidad y su capacidad de utilizar Bluetooth de forma integrada.

## 4.1. Diseño del circuito electrónico

El primer paso para realizar el rediseño del medidor de corriente ha consistido en la comprobación de qué partes del circuito electrónico pueden optimizarse y cuáles deben cambiarse para permitir el uso del nuevo microcontrolador. A continuación se exponen cada una de las partes del sistema.

El programa empleado para el diseño de los esquemáticos que se muestran en este apartado y el posterior diseño de la placa de circuito impreso ha sido Altium Designer.


**Figura 4.1** – *Logo de Altium Designer*

### 4.1.1. Microcontrolador ESP32-WROOM-32E

La familia ESP32, de Espressif, es una serie de microcontroladores con funcionalidades de Wi-Fi, Bluetooth y Bluetooth Low Energy diseñado con tecnología de 40 nm. Para este trabajo se ha escogido el chip ESP32-WROOM-32E, ya que es la versión más reciente y la recomendada por el fabricante para los nuevos diseños [6].


**(a)** *Modelo real* **(b)** *Huella*

**Figura 4.2** – *Microcontrolador ESP32-WROOM-32E*

El ESP32-WROOM-32E incluye una antena de PCB para las comunicaciones inalámbricas. Tiene dos núcleos que pueden ser controlados individualmente, y la frecuencia de reloj puede ajustarse entre 80 MHz y 240 MHz. Además, cuenta con un modo de bajo consumo. Entre sus periféricos, destacan 10 sensores táctiles capacitivos, 2 pines con DAC, 16 pines con ADC, 2 controladores SPI, 3 UART, I2S e I2C. Además, utiliza el sistema operativo FreeRTOS, el cual permite crear y gestionar diferentes tareas para controlar completamente el flujo del programa [7].

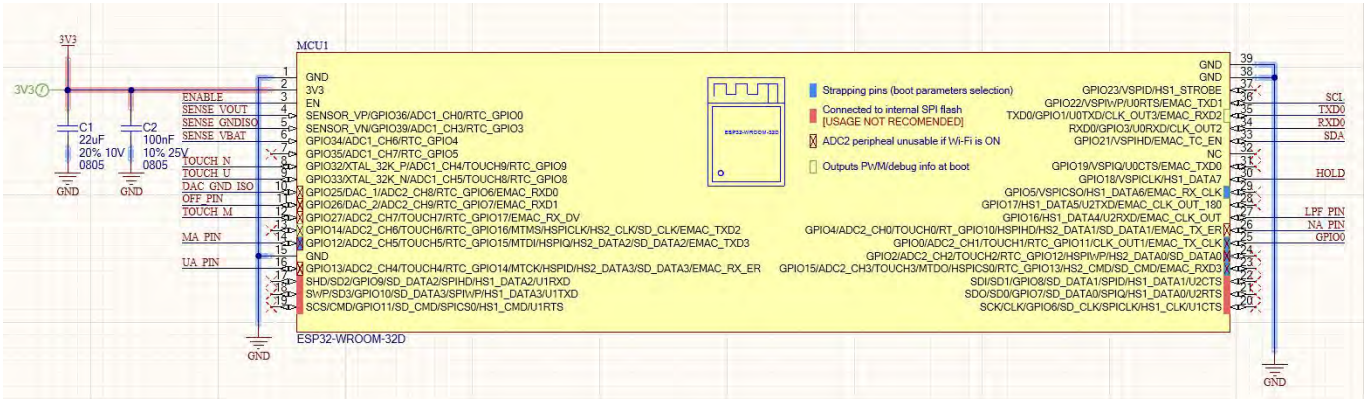



**Figura 4.3** – *Esquemático del ESP32-WROOM-32E*

En la figura 4.3 se puede observar el esquema de las conexiones de los pines del ESP32 con el resto del circuito. Los nombres de las conexiones se han escogido para dar la mayor claridad posible a su uso, pero a continuación se explicará con detalle cada parte del sistema.

La alimentación se realiza a 3.3 V, y se han colocado dos condensadores, uno de 22 $\mu$F y otro de 100 nF, con el objetivo de garantizar la estabilidad del voltaje y filtrar perturbaciones con bajas y altas frecuencias, respectivamente.

El ADC del ESP32 realiza mediciones con 12 bits. Por lo tanto, su resolución es:

$$\textbf{Resolución}_{\textbf{ADC}} = \frac{\mathbf{3{,}3\,V}}{\mathbf{2^{12}}} = \mathbf{0{,}8\,mV} \tag{4.1.1}$$

Según las espeficaciones proporcionadas por el fabricante Espressif, el rango de funcionamiento del ADC del ESP32 se sitúa entre 150 y 2450 mV. Además, los valores proporcionados en la medición pueden variar según la referencia de voltaje interna, la cual no es igual para todos los chips. Por ello, el ESP32 incluye funciones de calibración para lograr la máxima corrección posible de la medida. Esto se explicará en el apartado de firmware.

### 4.1.2. Conector USB-C

Para este proyecto se ha decidido emplear un conector USB-C, ya que representa el estándar actual para prácticamente la totalidad de los dispositivos comerciales.

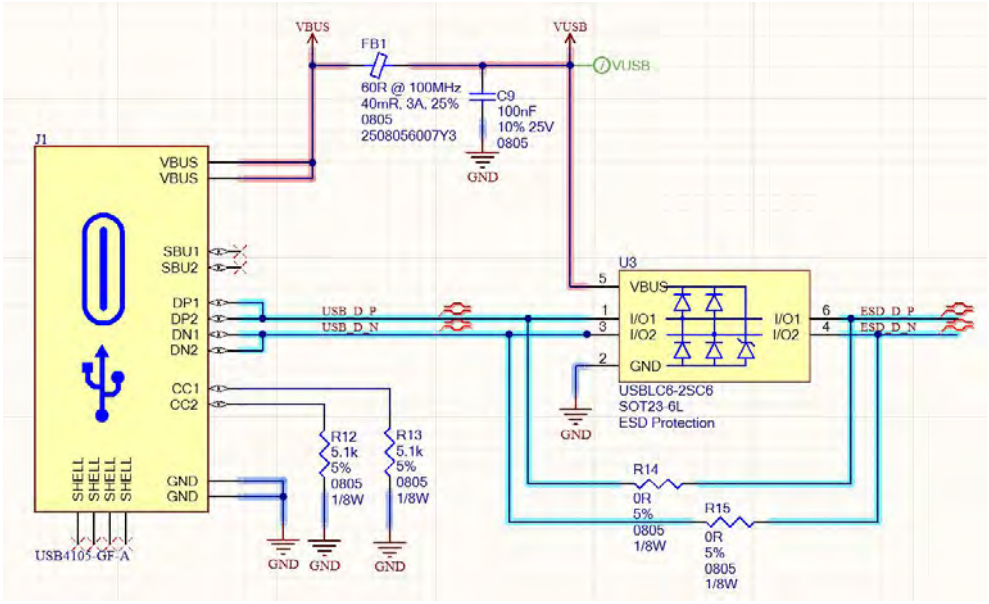
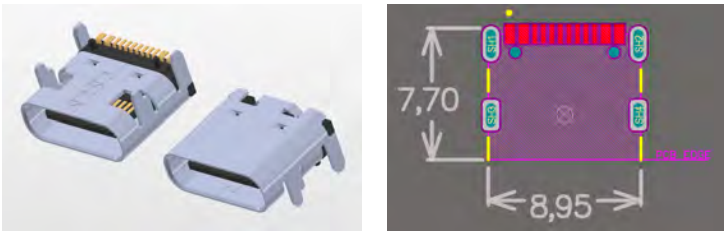



**Figura 4.4** – *Esquemático del conector USB-C*




**(a)** *Modelo 3D* **(b)** *Huella*

**Figura 4.5** – *Conector USB-C*

El conector USB-C, según el estándar, necesita dos resistencias de 5.1 kΩ conectadas en los pines CC1 y CC2, sin las cuales la comunicación no puede funcionar correctamente.

En este caso, como se va a utilizar comunicación USB 2.0, solo se utilizan los pines DP1, DP2, DN1 y DN2 para el envío o recepción de datos.

El voltaje de alimentación del bus USB se hace pasar a través de un núcleo de ferrita para filtrar las interferencias de alta frecuencia que puedan acompañar a la señal de alimentación, y, además, se utiliza un condensador de 100 nF. Al voltaje de alimentación resultante se le ha llamado VUSB, y se ha colocado un punto de prueba para poder medir cómodamente el voltaje durante la comprobación de funcionamiento.

Para proteger frente a descargas electrostáticas, se ha elegido el circuito integrado USBLC6, aunque, para realizar pruebas y prevenir posibles fallos del integrado, se han colocado resistencias de 0 Ω en paralelo que se pueden colocar o retirar según sea necesario.

La carcasa del conector se ha dejado desconectada de masa para asegurar la protección frente al contacto.

### 4.1.3. Cargador de batería

El circuito del cargador de batería y los transistores de protección se ha mantenido similar al diseño original del CurrentRanger R3, aunque se ha cambiado el circuito integrado por el BQ21040, de Texas Instruments, ya que es un producto más reciente y se adapta a las necesidades del proyecto [8].

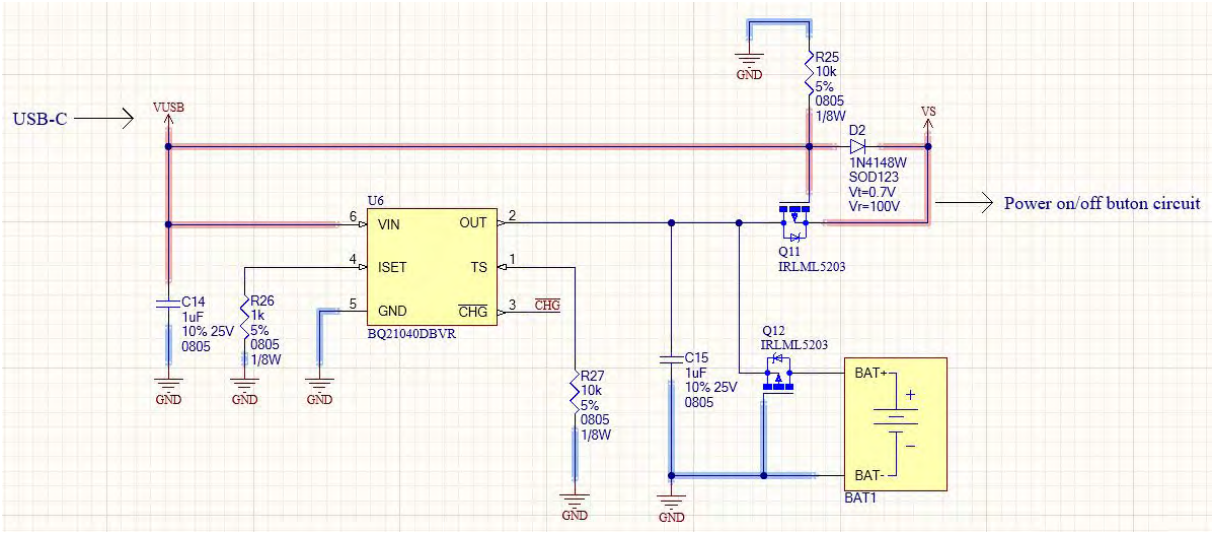
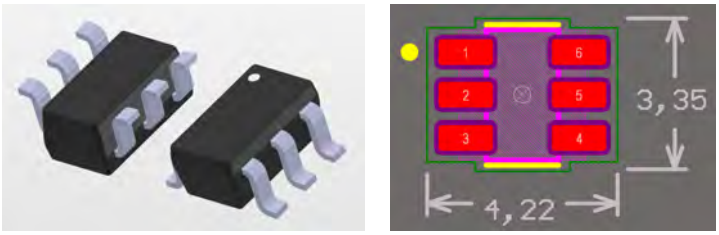



**Figura 4.6** – *Esquemático del cargador de la batería*

El BQ21040 es un cargador de 4.2 V para una única celda de Li-Ion o Li-Pol, con corriente regulable mediante una resistencia en el pin ISET hasta 0.8 A, aviso de fin de carga y protección frente a sobretensiones y a sobrecargas.




**(a)** *Modelo 3D* **(b)** *Huella*

**Figura 4.7** – *BQ21040*

En este caso, la resistencia de 1 kΩ fija la corriente máxima de carga en 500 mA. Esto se ha elegido por seguridad, ya que algunos puertos USB de ordenadores no pueden entregar más de 0.5 A.

Los condensadores de 1 $\mu$F en la entrada y la salida están recomendados por el fabricante en la hoja de datos del cargador.

El transistor PMOS Q12 se encarga de proteger frente a una conexión inversa de la batería, y el Q11 separa la batería del resto del circuito mientras esta se está cargando, ya que el voltaje VUSB es el encargado de alimentarlo. Ambos transistores son IRLML5203, capaces de conducir corrientes de hasta 3 A.

Por último, el pin CHG se encarga de indicar el fin de la carga. Este pin se pone a masa mientras la batería se está cargando, y pasa a ser flotante cuando se ha completado la carga. Como se verá más adelante, la indicación se realizará visualmente mediante un LED.

### 4.1.4. Botón de encendido/apagado

Como el sistema está diseñado para usar batería, es importante que cuente con un botón de encendido y apagado. En este caso, se ha cambiado completamente el circuito implementado en el CurrentRanger y se ha partido de cero con un nuevo diseño.

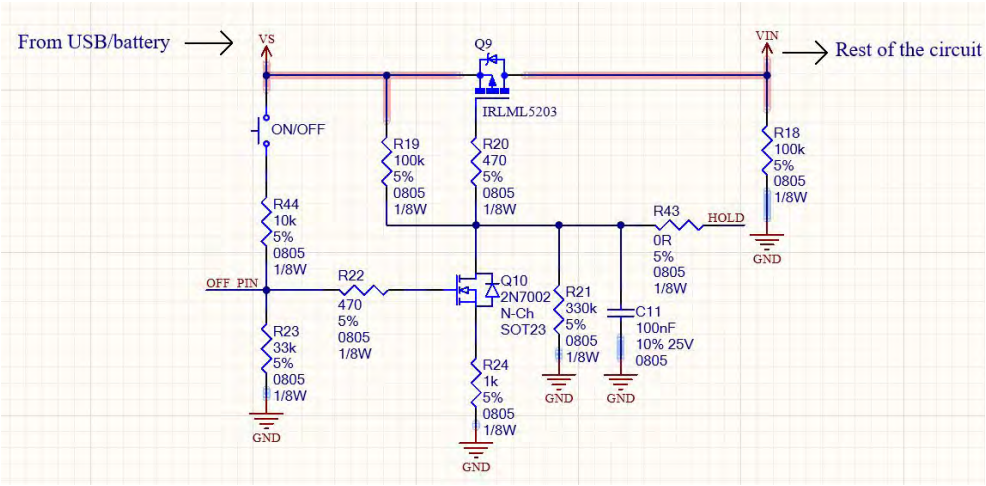



**Figura 4.8** – *Esquemático del botón de encendido/apagado*

Al conectar el sistema a una fuente de energía, ya sea batería o USB, el transistor Q9 se encuentra en corte y no se alimenta el circuito. Cuando se mantiene pulsado el botón ON/OFF, en la puerta del transistor NMOS Q10 pasa a haber un voltaje superior a 3 V, por lo que el transistor empieza a conducir, lo cual lleva la puerta del Q9 a masa. Esto, a su vez, permite el paso de corriente hacia el circuito.

Cuando el ESP32 se enciende al recibir energía, se encarga de mantener, a través del pin HOLD, la puerta de Q9 a masa, por lo que ya se puede soltar el botón y el circuito permanecerá encendido.

A su vez, el ESP32 lee continuamente el voltaje del pin OFF_PIN. Cuando detecta que el botón se vuelve a pulsar, convierte el pin HOLD de entrada a flotante, cortando así la alimentación y apagando el sistema.

La función de R44 y R23, así como de R19 y R21, es formar divisiones de tensión para proteger los pines del ESP32 frente a voltajes superiores a 3.3 V. El condensador C11, por su parte, se encarga de garantizar la estabilidad y que no se apague el sistema por una bajada momentánea de la tensión de alimentación. Las resistencias de las puertas de los MOSFET y R24 se encargan de proteger frente a corrientes altas cuando se incia la conducción de los transistores.

### 4.1.5. Regulador de tensión

El ESP32 es un microcontrolador que funciona a 3.3 V, por lo que hay que contar con un regulador que garantice que el voltaje se mantiene estable.

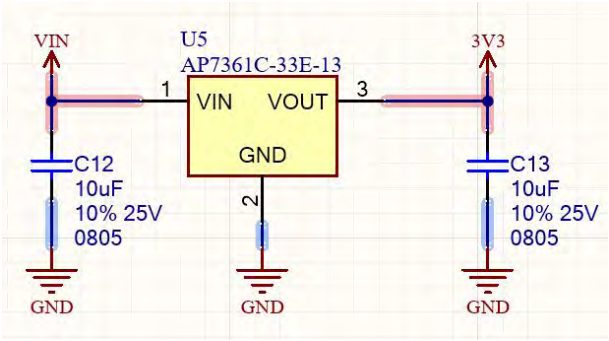
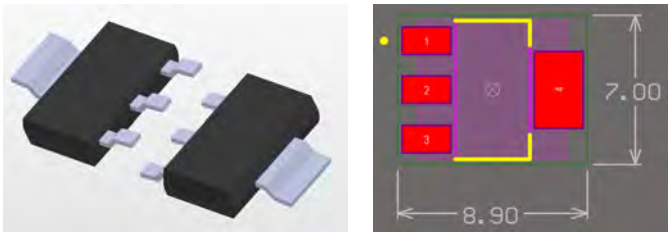



**Figura 4.9** – *Esquemático del regulador de 3.3 V*

El AP7361C, de Diodes Incorporated, es un regulador lineal de tensión fija que puede proporcionar hasta un 1 A de corriente y tiene una caída de voltaje muy baja, de 340 mV como máximo, lo cual lo hace idóneo para aplicaciones con baterías de una celda, ya que permite aprovechar una gran parte de la carga [9].




**(a)** *Modelo 3D* **(b)** *Huella*

**Figura 4.10** – *AP7361C*

El ESP32 puede funcionar, según su fabricante, con alimentación entre 2.3 V y 3.6 V, por lo que el sistema funcionaría mientras la batería proporcionara más de 2.7 V. La mayoría de celdas de litio proporcionan más de 3 V durante el 80 % de su carga, así que este regulador es una buena opción para el sistema.

Los condensadores de 10 $\mu$F en la entrada y la salida están recomendados en la hoja de datos del AP7361C.

### 4.1.6. Convertidor USB a UART

Para poder cargar el programa en la memoria del ESP32, enviar comandos y obtener los datos de las medidas, es necesario utilizar un convertidor de USB a UART, ya que el ESP32 no cuenta con interfaz USB integrada.

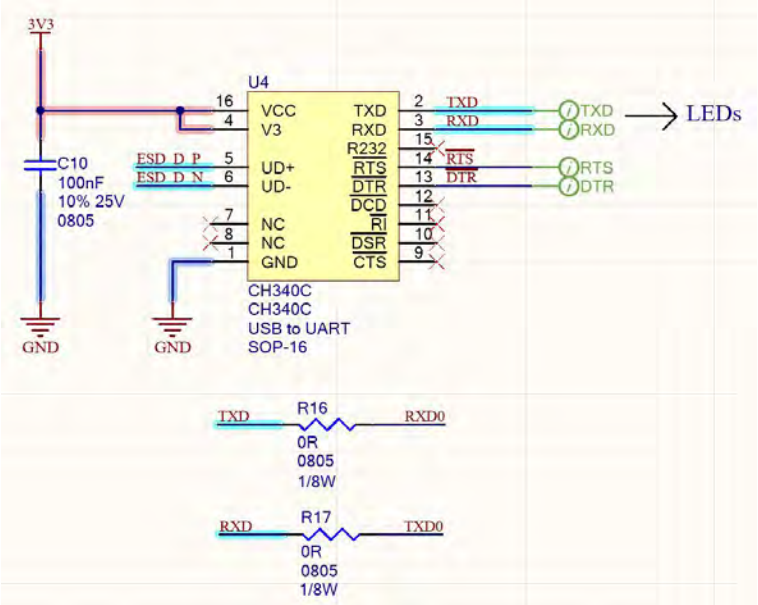
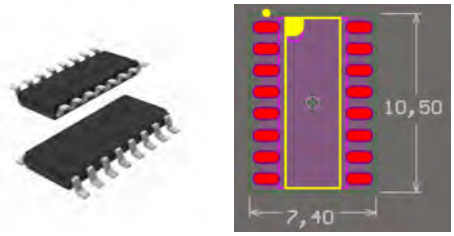



**Figura 4.11** – *Esquemático del convertidor USB a UART*

El CH340 es uno de los convertidores de USB a UART más utilizados debido a su facilidad de uso, ya que solo requiere alimentación a 5 V o 3.3 V y un condensador de estabilización, además de las señales diferenciales de datos del USB, D+ y D-.




**(a)** *Modelo 3D* **(b)** *Huella*

**Figura 4.12** – *CH340*

A partir del CH340 se obtienen las señales RX y TX para transmitir información, así como RTS y DTR, las cuales son utilizadas por el ESP32 para pasar a modo *bootloader* y reiniciar cuando estas siguen una secuencia concreta.

### 4.1.7. Pines de programación

El ESP32 debe recibir una secuencia a través de los pines ENABLE y GPIO0 para pasar a modo de carga de programa y reiniciarse cuando se haya completado la subida. Esto se puede realizar manualmente, utilizando pulsadores, o automáticamente con las señales RTS y DTR [10].

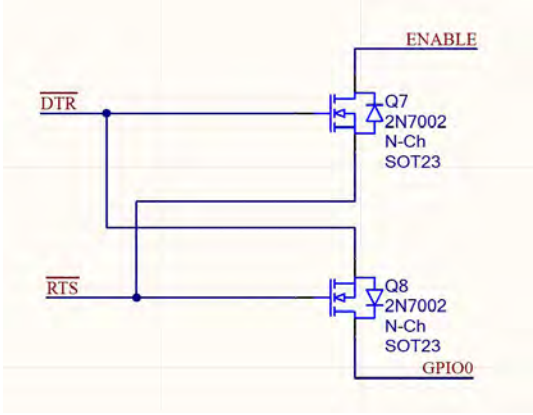



**Figura 4.13** – *Circuito de programación y reinicio automático*

Los pines ENABLE y GPIO tienen una resistencia de *pull-up* interna, por lo que, cuando DTR y RTS se encuentran a 0, ambos pines están a 1. La tabla de verdad del circuito de la figura 4.13 se muestra en la tabla 4.1.

| $\overline{\text{DTR}}$ | $\overline{\text{RTS}}$ | ENABLE | GPIO |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |

**Tabla 4.1** – *Tabla de verdad de los pines de programación*



Para pasar al modo programación, GPIO0 debe tener un valor de 0 mientras se reinicia la placa llevando el pin ENABLE a 0.

Para iniciar con el modo normal, es decir, con el arranque del programa almacenado en la memoria no volátil, GPIO0 debe valer 1 durante el reinicio.

### 4.1.8. Control de medida

Para seleccionar la rama por la que debe pasar la corriente, se ha tomado como referencia el diseño del CurrentRanger R3 y se han cambiado los transistores PMOS por NMOS para reducir su número y coste total del sistema.

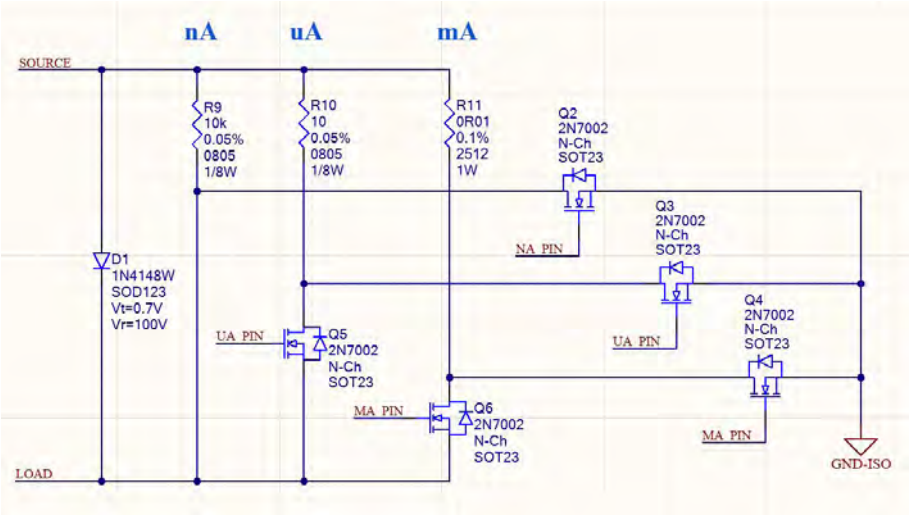



**Figura 4.14** – *Circuito de control de medida*

Al igual que se expuso en el apartado 3.1, cada rama está controlada por transistores MOS. En este caso, en cada rama se ha utilizado un NMOS para permitir el paso de corriente (Q5 y Q6), excepto en la rama nA, y otro NMOS para permitir el paso de voltaje sin paso de corriente (Q2, Q3 y Q4). Esto se ha realizado para poder medir el voltaje que cae en las resistencias sin tener en cuenta el que cae en los transistores de paso, ya que puede afectar bastante al resultado, sobre todo en la rama mA, donde la resistencia es de tan solo 10 mΩ.

El diodo D1 tiene la función de proteger al circuito frente a corrientes que causen una caída de tensión mayor a 0.6 V.

La señal GND-ISO, como se explicó para el CurrentRanger R3, tiene la función de elevar ligeramente el voltaje de referencia de la medida, con el objetivo de poder medir caídas de voltaje muy pequeñas.

### 4.1.9. Amplificador

En lugar de utilizar dos amplificadores MAX4239 como en el CurrentRanger, se ha reducido a solamente uno tras comprobar en la hoja de datos que es capaz de manejar ganancias de 100, en lugar de dos ganancias de 10 en cascada [11].

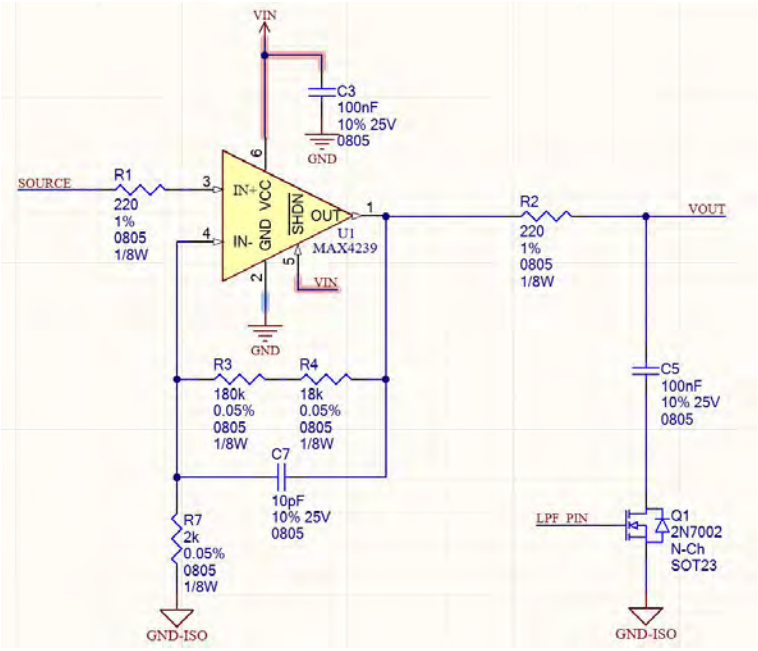
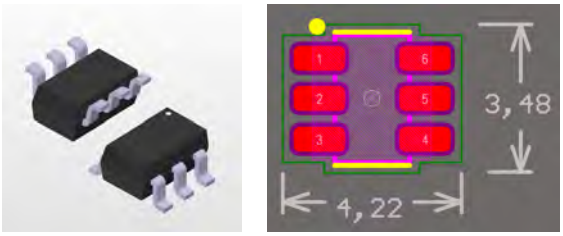



**Figura 4.15** – *Circuito amplificador*




**(a)** *Modelo 3D* **(b)** *Huella*

**Figura 4.16** – *MAX4239*

Para lograr la ganancia 100, se han elegido valores de resistencias apropiados, además de escoger modelos de precisión con muy baja tolerancia. Las resistencias de 220 Ω colocadas en la entrada y la salida tienen el

propósito de asegurar la estabilidad frente a las pequeñas corrientes de entrada y salida.

La alimentación del amplificador se realiza desde el voltaje VIN, no desde el regulador de 3.3 V, con el objetivo de no aumentar la demanda de corriente proporcionada por el regulador.

El condensador C7, de 10 pF, ayuda en la estabilización del amplificador, ya que filtra picos de corriente de frecuencia muy elevada que pueden afectar a la entrada del mismo. Por otra parte, el condensador C5, de 100 nF tiene la función de filtrar la medición de voltaje en la salida del amplificador, para reducir de esta forma interferencias y ruido que afecte al funcionamiento del ADC. Este condensador se controla con un pin del microcontrolador.

### 4.1.10. Pulsadores y *pads* táctiles

Además del botón de encendido/apagado, el medidor de corriente cuenta con pulsadores y *pads* táctiles para controlar el funcionamiento del sistema.

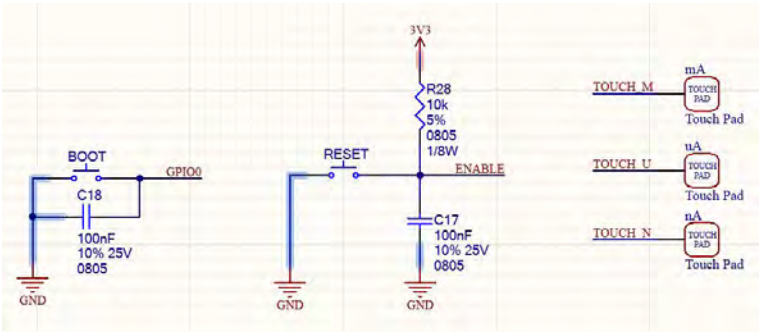



**Figura 4.17** – *Pulsadores y pads*

Los pulsadores BOOT y RESET están conectados a los pines GPIO0 y ENABLE, respectivamente, y sirven para controlar manualmente el arranque en modo *bootloader* o modo normal.

En este caso, se ha realizado una conexión con resistencia externa de *pull-up* para el pin ENABLE, aunque también se podría haber realizado para el pin GPIO0.

Los condensadores tienen la función de evitar rebotes durante la pulsación de los botones, lo cual puede afectar al proceso de arranque del ESP32.

Los tres *pads* táctiles, al igual que ocurría con el CurrentRanger R3, controlan el modo de funcionamiento del medidor.

### 4.1.11. ADC y DAC

Para conectar las señales a medir a los distintos pines del ESP32, se han empleado resistencias de 0 Ω, las cuales se pueden colocar o retirar con facilidad en caso de ser necesario.

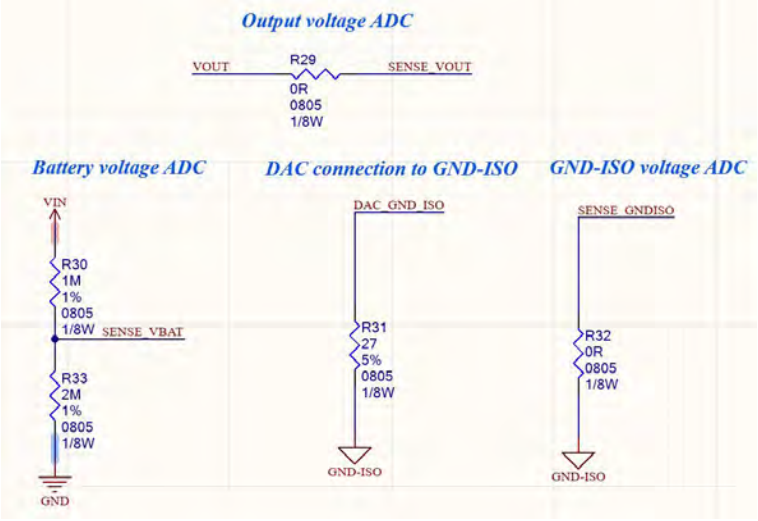



**Figura 4.18** – *ADC y DAC*

En primer lugar, las resistencias R30 y R31 forman un divisor de tensión para ajustar el voltaje de la batería a un nivel admitido por los pines del ESP32 y poder realizar la medición.

Las resistencias de 0 Ω se han utilizado para conectar VOUT y GND-ISO a sus respectivos pines, donde se realiza su medida utilizando el ADC interno del microcontrolador.

Para generar el voltaje de GND-ISO, se utiliza un pin que se conecta mediante una resistencia de bajo valor, 27 Ω. La función de esta resistencia es proteger al pin frente algún aumento puntual del voltaje que provenga de otra parte del circuito y pueda dañarlo.



### 4.1.12. Pantalla OLED

La pantalla del sistema es la SH1106, una pantalla OLED controlada por I2C, por lo que la conexión con el circuito se realiza mediante un conector hembra de 4 pines.

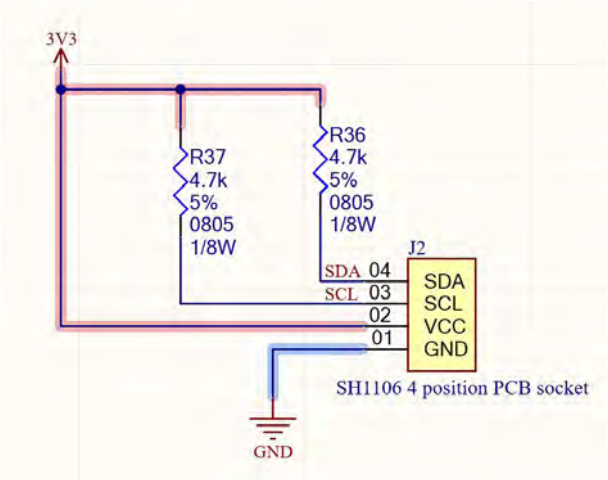



**Figura 4.19** – *Esquemático de conexión de la pantalla por I2C*

**Figura 4.20** – *SH1106*

R37 y R36 son las resistencias de terminación para las líneas de comunicación I2C: SDA para los datos y SCL para el reloj. Su función es actuar como *pull-up*, y su valor de 4.7 kΩ es el estándar utilizado en la mayoría de aplicaciones con este protocolo de comunicación.

### 4.1.13. LEDs

El sistema cuenta con diferentes LEDs para representar distinta información sobre el funcionamiento del mismo.

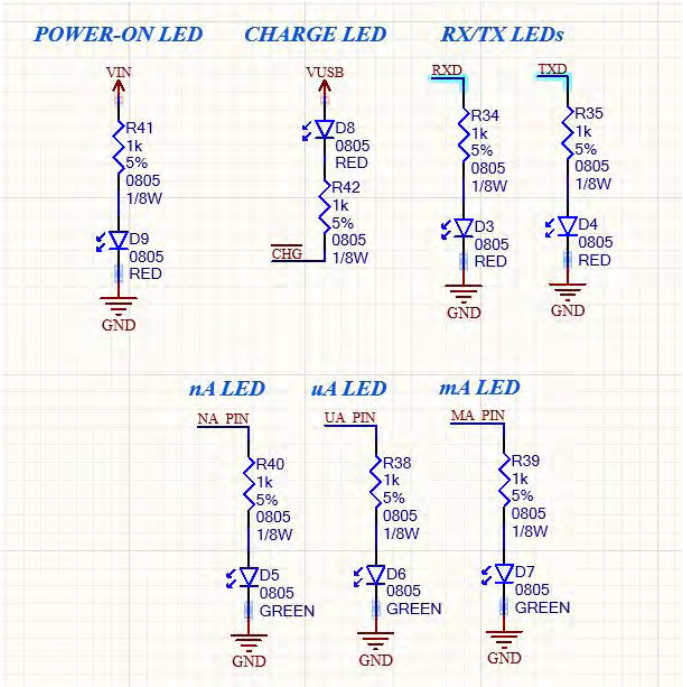



**Figura 4.21** – *LEDs del sistema*

Cuando se pulsa el botón de encendido, se ilumina el LED rojo D9, indicando que el sistema tiene energía y se encuentra funcionando, ya sea conectado a USB o con batería.

El LED rojo D8 indica que la batería está siendo cargada, y su estado depende del cargador BQ21040. Este pone el pin CHG a 0 cuando está entrando corriente en la batería, y lo mantiene flotante cuando la carga se ha completado o el USB no está conectado.

Los LEDs rojos RX y TX proporcionan una confirmación visual de que se está realizando la comunicación

UART correctamente.

Los LEDs verdes D5, D6 y D7 indican si el modo de medición es nanoamperios, microamperios o miliamperios.

Las resistencias son todas de 1 kΩ, ya que así la corriente que circula a través de los LEDs es pequeña y se evita tener un gasto de energía excesivo.

### 4.1.14. Conectores tipo banana

Las conectores que se van a utilizar para la entrada de la corriente a medir y la salida de voltaje son de tipo banana.

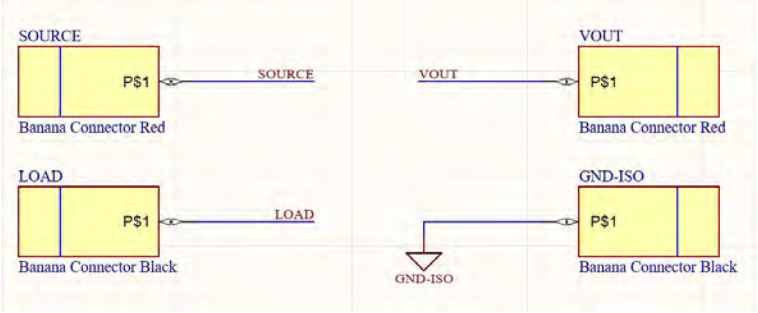
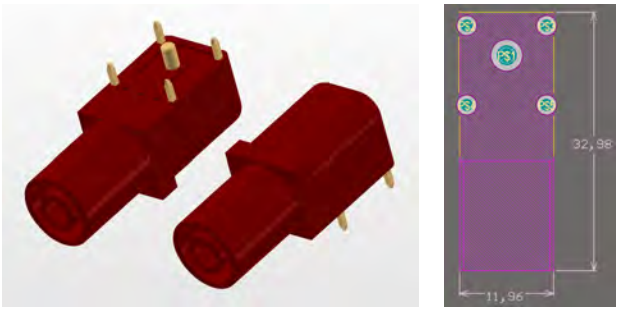



**Figura 4.22** – *Conexiones de los conectores*




**(a)** *Modelo 3D* **(b)** *Huella*

**Figura 4.23** – *Conectores tipo banana*

La entrada de la corriente se realiza por un conector de color rojo y la salida por uno de color negro, para facilitar su comprensión. A su vez, el voltaje de salida, VOUT, se mide también por otro conector de color rojo y la referencia, GND-ISO, por uno de color negro.

### 4.1.15. Diagrama de bloques

En la figura 4.24 se puede observar el diagrama de bloques de todas las partes del sistema, donde se representan las distintas señales y conexiones entre ellos.

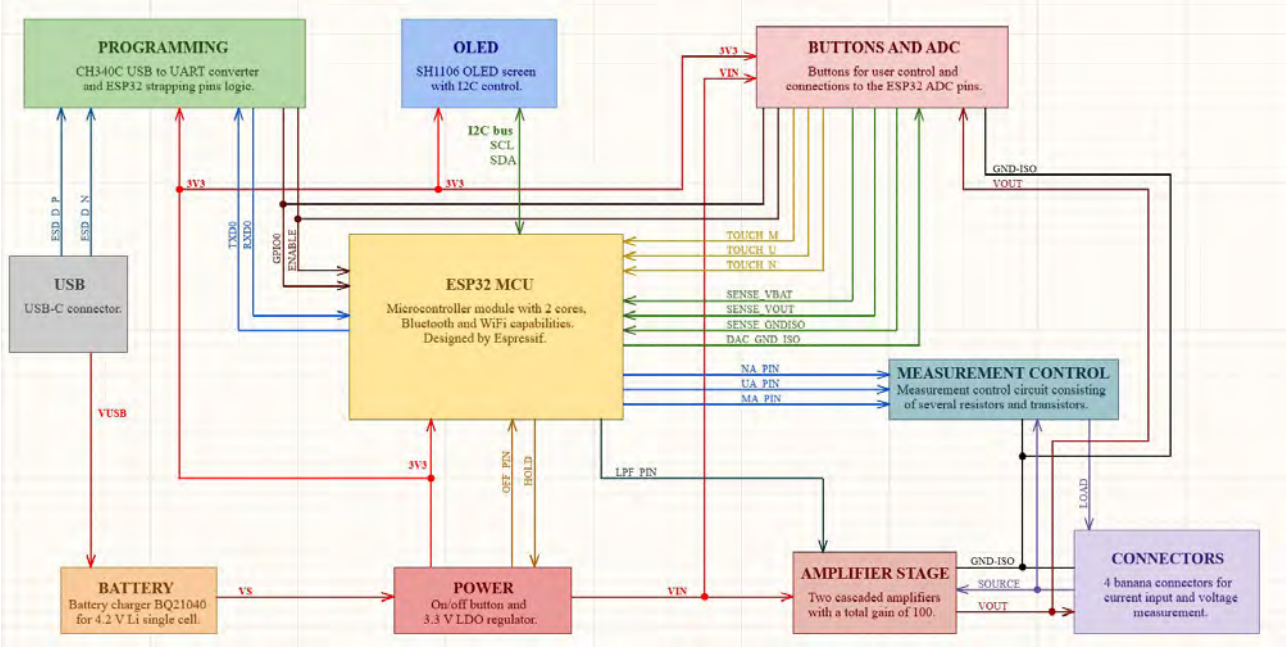



**Figura 4.24** – *Diagrama de bloques del sistema*

4

### 4.1.16. Consumo estimado del sistema

En la tabla 4.2 se muestran los consumos estimados del sistema, tanto de corriente como de potencia.

| Componente | Corriente máxima (mA) | Corriente media (mA) | Voltaje (V) | Potencia media (mW) |
|---|---|---|---|---|
| ESP32-WROOM-32E | 500 | 130 | 3.3 | 429.0 |
| BQ21040 | 500 | 1 | - | 5.0 |
| AP7361C | - | - | 0.4 | 150.0 |
| MAX4238 | 0.6 | 0.6 | 4 | 2.4 |
| CH340C | 20 | 7 | 3.3 | 23.1 |
| SH1106 | 27 | 27 | 3.3 | 89.1 |
| 7 LEDs | 9.1 | 9.1 | 2 | 18.2 |
| 1N4148W | - | - | 0.7 | 100.0 |
| **Total** | **1056.7** | **174.7** | **-** | **816.8** |

**Tabla 4.2** – *Consumo estimado de potencia y corriente del sistema*

La corriente máxima estimada del sistema es de alrededor de 1 A, dándose en el caso de que la batería esté cargándose y el ESP32 se encuentre enviando información a través de Bluetooth. En un funcionamiento normal donde no haya recarga de la batería, la corriente puede variar entre 175 mA y 500 mA.

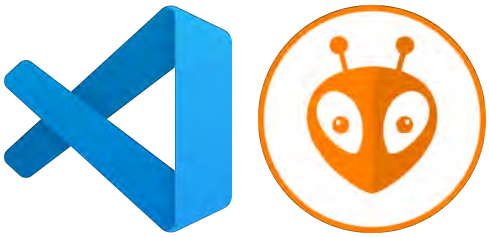
El consumo de potencia, por otro lado, es cercano a 1 W en total, concentrándose el mayor consumo en el microcontrolador, el regulador lineal de tensión y el diodo de protección de la línea USB.

## 4.2. Diseño del firmware

Para el diseño del firmware se ha partido del código del CurrentRanger R3 que se mejoró en el apartado 3.4, y se ha adaptado para lograr su funcionamiento en el microcontrolador ESP32. Para ello, ha sido necesario modificar o eliminar algunas funciones, además de crear otras nuevas. Todo el proceso se detalla a continuación.

El desarrollo del código se ha realizado con el programa Visual Studio Code, el cual es un editor de código desarrollado por Microsoft con soporte para multitud de lenguajes de programación. Dentro de VS Code, se ha utilizado la extensión PlatformIO, diseñada específicamente para la programación de microcontroladores y sistemas embebidos, que cuenta con una gran cantidad de modelos de microcontroladores existentes para seleccionar, incluidos todos los de la familia ESP32.

Los sistemas con ESP32 pueden ser programados con dos marcos diferentes. En primer lugar, utilizando el marco ESP-IDF, el cual está basado en C y está desarrollado por el fabricante Espressif. También se puede usar el marco de Arduino, el cual es una adaptación para igualar los nombres de las funciones básicas a los de la conocida plataforma Arduino, basada en C++. Para este proyecto se ha utilizado el marco de Arduino.



**(a)** *VS Code* **(b)** *PlatformIO*

**Figura 4.25** – *Software para la programación del sistema*



### 4.2.1. Librerías utilizadas

En la figura 4.26 se muestran las librerías empleadas en el código del medidor de corriente.

```
#include "../lib/esp32/Arduino.h"
#include "../lib/U8g2/src/U8g2lib.h"
#include "../lib/driver/adc.h"
#include "../lib/esp_adc_cal/esp_adc_cal.h"
#include "../lib/BluetoothSerial/src/BluetoothSerial.h"
#include "../lib/Preferences/src/Preferences.h"
#include "../lib/Wire/src/Wire.h"
#include "../include/serial.h"
#include "../include/common.h"
#include "../include/utils.h"
#include "../lib/driver/i2s.h"
```

**Figura 4.26** – *Librerías utilizadas*

La librería *U8g2lib* se utiliza para controlar el funcionamiento de la pantalla OLED. El resto de librerías son propias del microcontrolador ESP32, creadas por Espressif, y se utilizan para controlar las funcionalidades ADC, I2S, I2C y Bluetooth.

Se han creado *common.h*, *utils.h* y *serial.h* como archivos de cabecera, y contienen las definiciones de las funciones y variables utilizadas en el código.

### 4.2.2. Funcionalidades eliminadas

- Reinicio en *bootloader*: Esta función no es necesaria para el funcionamiento del ESP32, ya que no cuenta con una función de reinicio en modo *bootloader*, sino que el programador es capaz de realizar

este proceso mediante hardware.

- Velocidad, sincronización y corrección del ADC: Para estas funciones, el CurrentRanger empleaba registros del microcontrolador SAMD21, por lo que no pueden utilizarse para el ESP32.
- Actualización del LDO: El ESP32 tiene guardado en su memoria el valor del regulador de 3.3 V, por lo que no es necesario medirlo de forma externa y almacenar su valor.
- Aviso sonoro: No se ha añadido aviso con *buzzer* al sistema, por lo que su función no es necesaria.

### 4.2.3. Funcionalidades modificadas

- Calibración del ADC: El ADC del ESP32 debe ser calibrado antes de realizar mediciones. Cuenta con varios modos de atenuación, los cuales permiten medir un menor rango de voltajes con más precisión, o un mayor rango con menos precisión. En este caso se ha utilizado la mayor atenuación, de 11 dB, para poder medir el máximo rango de voltajes, el cual cuenta con mayor precisión entre 150 y 2450 mV. Además, la función *esp_adc_cal_raw_to_voltage()* permite la conversión directa del valor del ADC a mV, utilizando para ello el valor exacto de la referencia interna de voltaje del microcontrolador. Los argumentos de esta función son el valor de lectura del ADC y una variable tipo *struct* que almacena las características de calibración.

```
esp_adc_cal_characteristics_t *adc_chars = (esp_adc_cal_characteristics_t *)calloc(1, sizeof(esp_adc_cal_characteristics_t));
esp_adc_cal_value_t val_type = esp_adc_cal_characterize(ADC_UNIT_1, ADC_ATTEN_DB_11, ADC_WIDTH_BIT_12, ESP_ADC_CAL_VAL_EFUSE_VREF, adc_chars);
```

**Figura 4.27** – *Calibración del ADC*

- Lectura del ADC para señal con componente AC: El ESP32 puede utilizar funcionalidades del protocolo I2S para leer el ADC y almacenar los valores con acceso directo a memoria (DAM), lo cual aumenta considerablemente la velocidad frente a la lectura convencional. La memoria RAM del ESP32 es de 520 kB, frente a los 32 kB del SAMD21. Esto permite que el búfer pueda almacenar hasta 30000 muestras en 30 búferes de 1000 muestras cada uno, un número mucho mayor a las 5000 muestras del CurrentRanger R3.

```
int gnd_ac_voltage;
void ACmeasure()
{
  if (!VoltageWarning)
  {
    gnd_ac_voltage = readGNDISO();

    i2s_adc_disable(I2S_NUM_0);
    i2s_set_adc_mode(ADC_UNIT_1, SENSE_OUTPUT);
    i2s_adc_enable(I2S_NUM_0);

    volatile uint32_t time1 = millis();
    for (int i = 0; i < AC_BUFFER_NUMBER; i++)
    {
      i2s_read(I2S_NUM_0, &V_OUT_AC[i], sizeof(V_OUT_AC[i]), &bytes_read, portMAX_DELAY);
    }
    volatile uint32_t time2 = millis();

    int measurement_time = time2-time1;

    if (USB_LOGGING_ENABLED) Serial.println(measurement_time);
    if (BT_LOGGING_ENABLED) SerialBT.println(measurement_time);
  }
}
```

**Figura 4.28** – *Lectura en AC*

- Impresión del búfer de almacenamiento para la medición con componente AC: Para la impresión de los valores AC almacenados, se realiza un barrido por cada posición de cada búfer en el mismo orden en el que se almacenaron los datos. En la figura 4.29 se muestra el caso de la impresión por USB,

pero con Bluetooth se realiza de la misma manera. En función del formato de registro, los datos se imprimen con el formato correspondiente.

```
void printACbuffer()
{

  if (USB_LOGGING_ENABLED)
  {
    for (int i = 0; i < AC_BUFFER_NUMBER; i++)
    {
      for (int j = 0; j < I2S_DMA_BUF_LEN; j++)
      {
        int16_t voltage = esp_adc_cal_raw_to_voltage((V_OUT_AC[i][j]&0x0FFF),adc_chars)-gnd_ac_voltage;
        if(LOGGING_FORMAT == LOGGING_FORMAT_EXPONENT) { Serial.print(voltage); Serial.print("e"); Serial.println(RANGE_NA ? -9 : RANGE_UA ? -6 : -3); } else
        if(LOGGING_FORMAT == LOGGING_FORMAT_NANOS) Serial.println(voltage * (RANGE_NA ? 1 : RANGE_UA ? 1000 : 1000000)); else
        if(LOGGING_FORMAT == LOGGING_FORMAT_MICROS) Serial.println(voltage * (RANGE_NA ? 0.001 : RANGE_UA ? 1 : 1000)); else
        if(LOGGING_FORMAT == LOGGING_FORMAT_MILLIS) Serial.println(voltage * (RANGE_NA ? 0.000001 : RANGE_UA ? 0.001 : 1));
      }
    }
  }
```

**Figura 4.29** – *Impresión por USB del búfer AC*

- Mediciones individuales de la salida, GND-ISO y el voltaje de la batería: Al igual que para el búfer AC, las mediciones de corriente continua y de la batería se realizan con I2S, aunque en primer lugar hay que configurar de nuevo el I2S para leer el pin correspondiente. Se llena un búfer de 1000 muestras y se hace el promediado para tener una gran exactitud. Gracias a la velocidad del DMA, este proceso resulta mucho más rápido que utilizar las mediciones convencionales en las que los datos deben pasar por el microcontrolador.

```
void DCmeasure()
{

  int gnd_voltage = readGNDISO();
  if (gnd_voltage >= 2000) {
    VoltageWarning = true;
    Serial.println("Warning: Vin > 0.6 V. Protection diode active");
  }
  else VoltageWarning = false;

  i2s_adc_disable(I2S_NUM_0);
  i2s_set_adc_mode(ADC_UNIT_1, SENSE_OUTPUT);
  i2s_adc_enable(I2S_NUM_0);

  i2s_read(I2S_NUM_0, &V_OUT_DC, sizeof(V_OUT_DC), &bytes_read, portMAX_DELAY);

  int total = 0;
  for (int i = 0; i < I2S_DMA_BUF_LEN; i++)
  {
    total = total + (esp_adc_cal_raw_to_voltage(V_OUT_DC[i]&0x0FFF,adc_chars) - gnd_voltage);
  }
  VOUT = total/float(I2S_DMA_BUF_LEN);
}
```

**Figura 4.30** – *Lectura del voltaje en DC*

- Watchdog: Para implementar un watchdog y proteger al sistema frente a errores que lo dejen bloqueado, se ha utilizado uno de los contadores del ESP32, el cual utiliza el contador del reloj interno de 80 MHz y funciona de forma completamente independiente al programa principal. Cuando llega hasta el valor deseado, se genera una interrupción que activa la función de reinicio. En este caso, el tiempo de espera del watchdog es de 8 s.

```
void WDTset()
{
  timer = timerBegin(0,80,true);                  //timer 1Mhz resolution
  timerAttachInterrupt(timer, &resetModule,true);  //attach callback
  timerAlarmWrite(timer, WDT_TIMEOUT * 1000, true);  //set time in us
  timerAlarmEnable(timer);
}
```

**Figura 4.31** – *Configuración del temporizador para el watchdog*

- Sensores táctiles capacitivos: Para poder utilizar los *pads* táctiles, en lugar de utilizar una librería externa se ha empleado una función propia del ESP32. Estos valores se monitorizan cada 100 ms. Si se ha detectado un toque, la variable correspondiente se define como *true*, mientras que, si no hay toque, se define como *false*.

  Al igual que ocurría en el CurrentRanger, un toque en el *pad* correspondiente cambia la unidad de medida de la corriente. Al pulsar al mismo tiempo los *pads* NA y UA, se conmuta el filtro paso bajo. Al tocar al mismo tiempo UA y MA, se conmuta el modo BIAS. Por último, al tocar NA y MA, se activa el modo auto rango.

```
void handleTouchPads()
{

  if (millis() > touchTime + TOUCH_WAIT_TIME){
    touchTime = millis();
    uint8_t ReadingN = touchRead(TOUCH_N);
    uint8_t ReadingU = touchRead(TOUCH_U);
    uint8_t ReadingM = touchRead(TOUCH_M);

    MA_PRESSED = (ReadingM < TOUCH_THRESHOLD? true : false);
    UA_PRESSED = (ReadingU < TOUCH_THRESHOLD? true : false);
    NA_PRESSED = (ReadingN < TOUCH_THRESHOLD? true : false);
```

**Figura 4.32** – *Lectura de los pads táctiles*

- Comandos: Los comandos de funcionamiento del medidor de corriente son prácticamente los mismos que los comandos del CurrentRanger que se modificaron en el apartado 3.4. Las únicas diferencias en este sentido han consistido en la eliminación o variación de algunos de los comandos por no afectar al funcionamiento del microcontrolador ESP32. Los comandos eliminados han sido: BOOTLOADER, por no tener el ESP32 ninguna forma de reiniciar en modo *bootloader* desde software, LDO, por tener el valor guardado desde la fabricación del chip, y ADC, por no funcionar el conversor de la misma manera en el ESP32 que en el SAMD21. El comando modificado ha sido GAIN, ya que, mientras que con el SAMD21 afectaba al funcionamiento del ADC, en el ESP32 se ha utilizado su valor para modificar el resultado final mostrado en la pantalla. Por ello, el cálculo de la corriente medida se realiza del siguiente modo:

$$\textbf{Current} = (\textbf{Medición} + \textbf{Offset}) \cdot (\mathbf{1} + \mathbf{0{,}001} \cdot \textbf{Gain}) \tag{4.2.1}$$

### 4.2.4. Funcionalidades nuevas

- Inicialización del ADC por I2S: Se realiza una primera configuración de la medición del voltaje por I2S. Se define el modo de comunicación por I2S, que en este caso es maestro por ser el único dispositivo involucrado en la medición. Se utiliza la tasa de muestreo más alta posible, que es de 300000 muestras por segundo. Las muestras son de 12 bits, y se inicializa el búfer de 1000 muestras donde se almacenarán los valores. Este número está cerca del límite permitido como tamaño de búfer.

```
void i2sInit()
{
  i2s_config_t i2s_config = {
    .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX | I2S_MODE_ADC_BUILT_IN),
    .sample_rate =  I2S_SAMPLE_RATE,            // The format of the signal using ADC_BUILT_IN
    .bits_per_sample = I2S_BITS_PER_SAMPLE_16BIT, // is fixed at 12bit, stereo, MSB
    .channel_format = I2S_CHANNEL_FMT_ALL_LEFT,
    .communication_format = I2S_COMM_FORMAT_STAND_I2S,
    .intr_alloc_flags = 0,
    .dma_buf_count = 8,
    .dma_buf_len = I2S_DMA_BUF_LEN,
    .use_apll = false,
    .tx_desc_auto_clear = false,
    .fixed_mclk = 0
  };

  i2s_driver_install(I2S_NUM_0, &i2s_config, 0, NULL);
  i2s_set_adc_mode(ADC_UNIT_1, SENSE_OUTPUT);
  i2s_adc_enable(I2S_NUM_0);
  adc1_config_channel_atten(SENSE_OUTPUT, ADC_ATTEN_DB_11);
  adc1_config_channel_atten(SENSE_GNDISO, ADC_ATTEN_DB_11);
  adc1_config_channel_atten(SENSE_VBAT, ADC_ATTEN_DB_11);
}
```

**Figura 4.33** – *Configuración de la lectura por I2S*

- Bluetooth: Se ha creado una función para imprimir datos o enviar comandos al sistema, utilizando la funcionalidad integrada de Bluetooth del ESP32. De igual forma que para la comunicación serial, cuando se detecta que se están recibiendo caracteres, estos se almacenan en una variable de tipo *string*, con la cual se realizan las comparaciones de los comandos.



```
void checkSerial(){
  if (Serial.available()>0){
    lastKeepAlive = millis();
    inString = Serial.readString();
    inString.trim();
    //Serial.println(inString); SerialBT.println(inString);
  }
  else if (SerialBT.available()>0){
    lastKeepAlive = millis();
    inString = SerialBT.readString();
    inString.trim();
    //Serial.println(inString); //SerialBT.println(inString);
  }
```

**Figura 4.34** – *Envío de comandos por USB o Bluetooth*

- Función de reinicio: Cuando el watchdog llega al valor final, se ejecuta un comando propio del ESP32 que reinicia el módulo, pero antes se envía un mensaje de aviso por puerto serie o Bluetooth como elemento de seguridad.

```
void resetModule()
{
  Serial.println("reboot");
  SerialBT.println("reboot");
  esp_restart();
}
```

**Figura 4.35** – *Reinicio del sistema*

- Lectura del botón de encendido/apagado: El microcontrolador monitoriza continuamente el valor del pin OFF_PIN, conectado al botón. Cuando el sistema está encendido, si se detecta que se pulsa el botón y se suelta, se define como flotante pin que mantiene la alimentación y el sistema se apaga completamente.

```
void handleOffButton()
{
  if (millis() > power_on_time)
  {
    if (millis() > offButtonTime + OFF_BUTTON_INTERVAL)
    {
      offButtonTime = millis();
      uint8_t pinValue = digitalRead(OFF_PIN);
      //Serial.println(pinValue);
      if (pinValue==1)
      {
        offButtonState = true;
      }
      if (pinValue==0 && offButtonState == true){
        pinMode(HOLD, INPUT);
      }
    }
  }
}
```

**Figura 4.36** – *Lectura del botón de encendido/apagado*

### 4.2.5. FreeRTOS



Los microcontroladores ESP32 utilizan el sistema operativo FreeRTOS. Este sistema operativo permite organizar la estructura del código en tareas y aprovechar con mayor eficiencia la arquitectura de dos núcleos del ESP32. Esto permite que los programas se ejecuten con mayor rapidez y proporciona al diseñador mayor control sobre los mismos.

En el caso del sistema medidor de corriente, se han creado cuatro tareas para monitorizar los *pads* táctiles, el botón de apagado, la función de auto apagado y el reinicio del watchdog.

```
xTaskCreate(handleTouchPads, "handleTouchPads()", 1024, NULL, 0, &handleTouch);
xTaskCreate(handleOffButton, "handleOffButton()", 1024, NULL, 0, &handleOff);
xTaskCreate(handleAutoOff, "handleAutoOff()", 1024, NULL, 0, &handleAutoff);
xTaskCreate(WDTclear, "WDTclear()", 1024, NULL, 0, &ClearWDT);
```

**Figura 4.37** – *Creación de tareas con FreeRTOS*

Para crear una tarea, esta debe contar con el nombre de la función que describa la propia tarea, un nombre en forma de variable *string*, un tamaño de pila adecuado, el número de prioridad y un *handler* para poder referirse a la tarea más adelante. En este caso, todas las tareas tienen un tamaño de pila de 1024, ya que son tareas relativamente simples, y una prioridad de 0, la más baja, debido a que todas tienen la misma prioridad.

### 4.2.6. Prueba de funcionamiento del firmware

Una vez se ha escrito el firmware del sistema, se ha comprobado su funcionamiento empleando una placa de desarrollo de ESP32. Estas placas de pruebas permiten acceder a todos los pines del microcontrolador y conectar las diferentes partes del sistema de forma cómoda con una placa de pruebas.

**Figura 4.38** – *Placa de desarrollo ESP32*

Con la compilación y subida del programa al ESP32, se ha comprobado el espacio ocupado por el mismo, así como las memorias RAM y ROM disponibles.

```
Archiving .pio\build\esp32dev\libFrameworkArduino.a
Linking .pio\build\esp32dev\firmware.elf
Retrieving maximum program size .pio\build\esp32dev\firmware.elf
Checking size .pio\build\esp32dev\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [===       ]  32.7% (used 107176 bytes from 327680 bytes)
Flash: [========= ]  90.0% (used 1179133 bytes from 1310720 bytes)
Building .pio\build\esp32dev\firmware.bin
```

**Figura 4.39** – *Compilación y memoria ocupada por el programa*

La memoria ROM, identificada como Flash en la figura 4.39, está casi completamente ocupada. A pesar de que aún queda un 10 % libre, ocupar ese espacio final puede ocasionar problemas en el funcionamiento del sistema. Por lo tanto, se está utilizando toda la capacidad disponible del ESP32.

Los pines de la placa de evaluación se han conectado, utilizando una placa de pruebas, a diferentes LEDs, botones y una pantalla OLED, con el objetivo de comprobar visualmente el funcionamiento del programa.

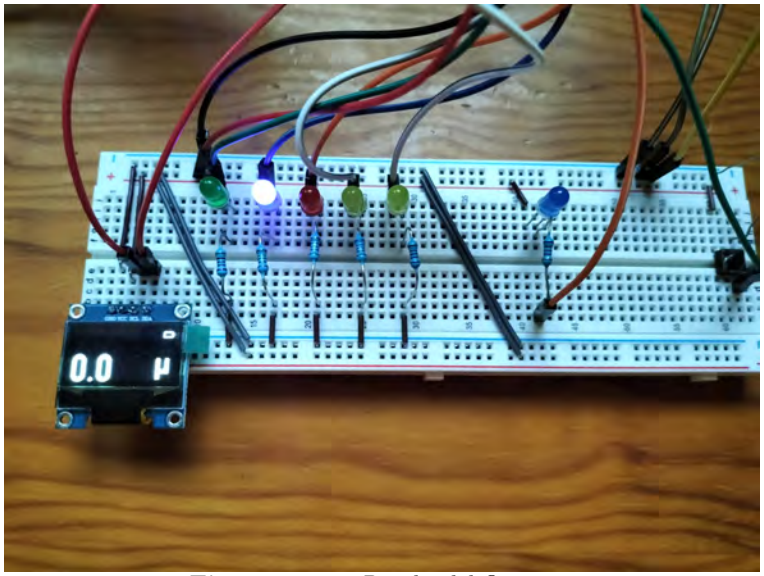



**Figura 4.40** – *Prueba del firmware*

Aunque no ha sido posible probar todas las funcionalidades, se ha podido confirmar que el programa se ejecuta correctamente, sin detenerse o bloquearse. Se han comprobado los cambios de modo de medida empleando los *pads* táctiles, con LEDs para confirmar visualmente el estado del sistema. Se ha conectado

una pantalla OLED, confirmando que su valor se actualiza correctamente, y un pulsador y un LED para representar el botón de apagado. Por lo tanto, tras comprobar que el código funciona, se ha pasado al diseño de la PCB, la compra de los materiales necesarios y el proceso de montaje.

## 4.3. Diseño de la placa de circuito impreso

Partiendo del diseño electrónico del apartado 4.1, una vez se ha comprobado el funcionamiento del software de forma general, se ha procedido con el diseño de la placa de circuito impreso del sistema.

Las huellas de los componentes se han obtenido de la página web SnapEDA, donde hay multitud de símbolos, huellas y modelos 3D de casi todos los componentes electrónicos existentes en el mercado.

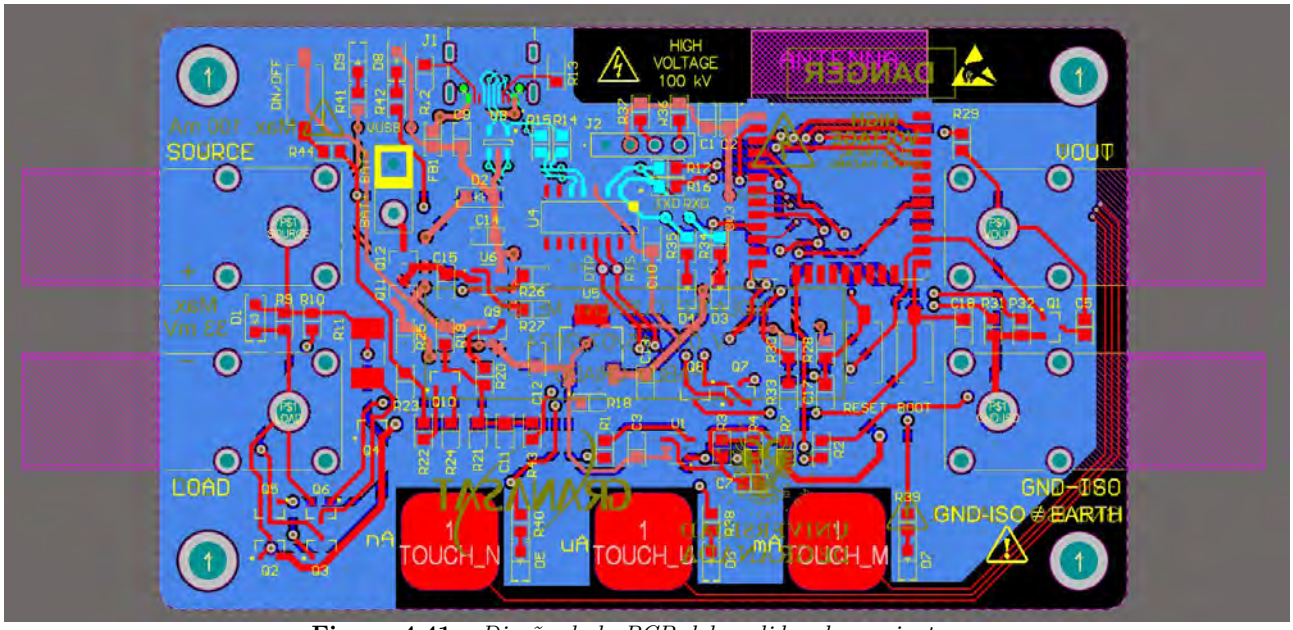



**Figura 4.41** – *Diseño de la PCB del medidor de corriente*

La placa tiene unas medidas de 10 x 6 cm, con agujeros para tornillos M4. En la parte superior se encuentra el conector USB-C, así como el microcontrolador ESP32. Cerca de la esquina superior izquierda se encuentran el botón de encendido/apagado y los LEDs de encendido y carga de la batería.

Las resistencias y condensadores se han seleccionado con tamaño 0805 para permitir una soldadura más fácil en comparación con tamaños como 0603 o menores. Destaca la resistencia de 10 mΩ, de tamaño 2512, mucho mayor a las demás, por su valor de resistencia tan reducido.

Las pistas de comunicación diferencial USB y su conversión a UART se reconocen en color azul claro, mientras que las pistas de alimentación se distinguen en color rojo claro.

Las pistas de alimentación del circuito tienen una anchura de 0.8 mm, adecuada para el paso de corriente de hasta 1 A. Las pistas de señales tienen una anchura de 0.5 mm, mayor de lo necesario pero aprovechando de ese modo el hueco entre los componentes, lo cual asegura un funcionamiento adecuado. Las pistas de los sensores táctiles, ya que deben presentar la menor capacidad posible, son mucho más finas, de 0.25 mm, y, en la medida de lo posible, no tienen plano de masa a su alrededor.

Los *pads* táctiles tienen unas medidas de 1 x 1 cm, y no tienen plano de masa a su alrededor para permitir su uso como sensor capacitivo. Su diseño se ha basado en las indicaciones encontradas en [12]. Los avisos de precaución se han colocado tanto en la capa superior como inferior, recordando el uso previsto de la placa

cerca de una fuente de voltaje de -100 kV, la intensidad y voltaje máximos de entrada y la función de cada *pad* táctil.

Los conectores para la batería, que consisten en dos agujeros donde se van a soldar los cables del porta baterías, se encuentran situados cerca del USB, por su conveniencia para alimentar el circuito desde ese punto.

Las resistencias para medir los distintos rangos de corriente se encuentran colocadas entre los conectores tipo banana SOURCE y LOAD, para reducir la distancia que debe recorrer la corriente y aumentar la precisión en la medida. Por conveniencia, los transistores de selección de rama se encuentran colocados bajo el conector LOAD.

Se han colocado puntos de prueba para comprobar el voltaje de alimentación del USB, las señales TX, RX, DTR y RTS del protocolo UART, y el voltaje de 3.3 V de alimentación.

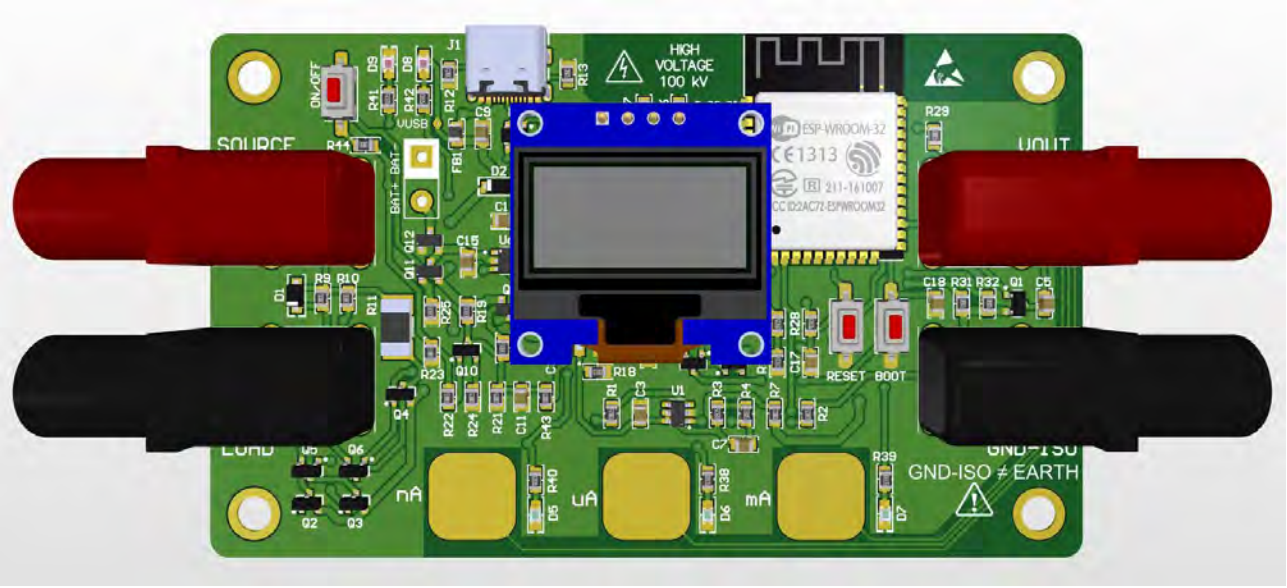



**Figura 4.42** – *Parte superior de la PCB*

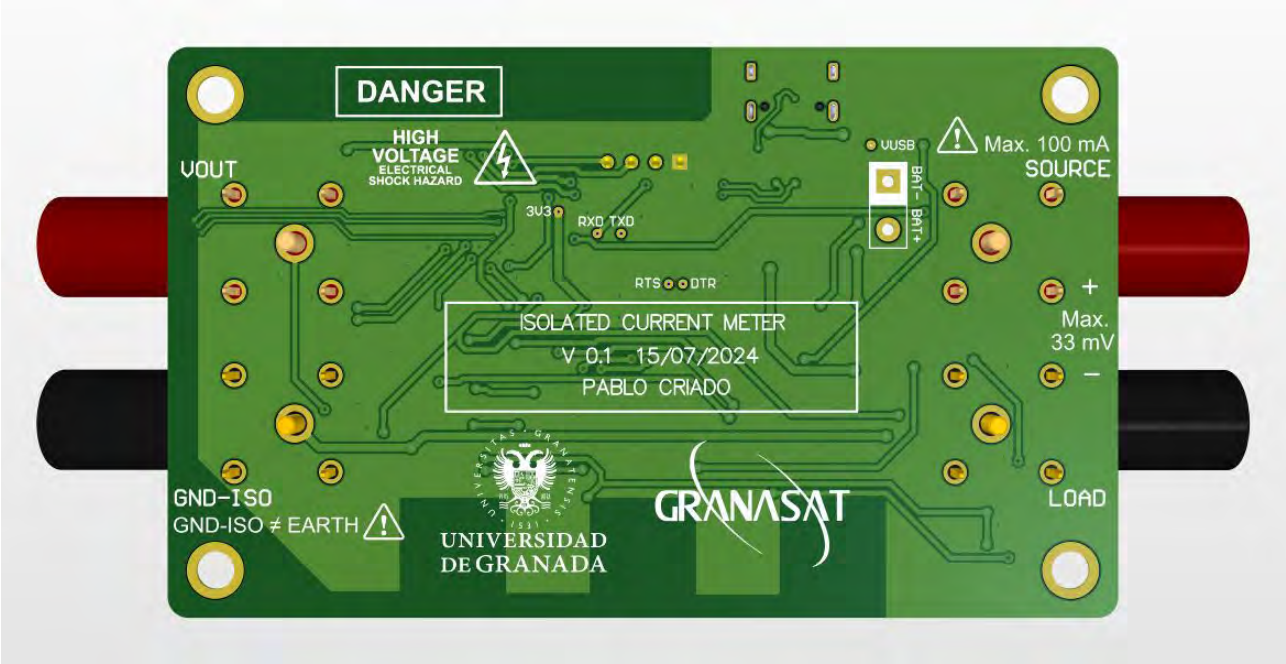



**Figura 4.43** – *Parte inferior de la PCB*

Los archivos Gerber para la fabricación de la placa se muestran en el Apéndice D, y la lista de materiales en el Apéndice C.

## 4.4. Montaje de la placa de circuito impreso

Los componentes necesarios para el montaje de la PCB se han adquirido en la tienda LCSC. La fabricación de la placa se ha pedido a JLCPCB.

**(a)** *Logo de LCSC* **(b)** *Logo de JLCPCB*

**Figura 4.44** – *Tiendas de componentes y fabricación*

Las opciones seleccionadas para la fabricación de la PCB son las siguientes:

- Material: FR4
- Capas: 2
- Dimensiones: 100 x 60 mm
- Cantidad de PCB: 5 unidades
- Grosor de la placa: 1.6 mm
- Color: verde

- Acabado superficial: HASL sin plomo

- Peso de cobre exterior: 1 onza

- Vías cubiertas con máscara de soldadura

- Tolerancia del borde de la placa: $\pm 0.2\,\text{mm}$

- Se incluye un *stencil* para facilitar la soldadura

En la figura 4.45 se pueden observar una de las placas y el *stencil* tras haber recibido el pedido.

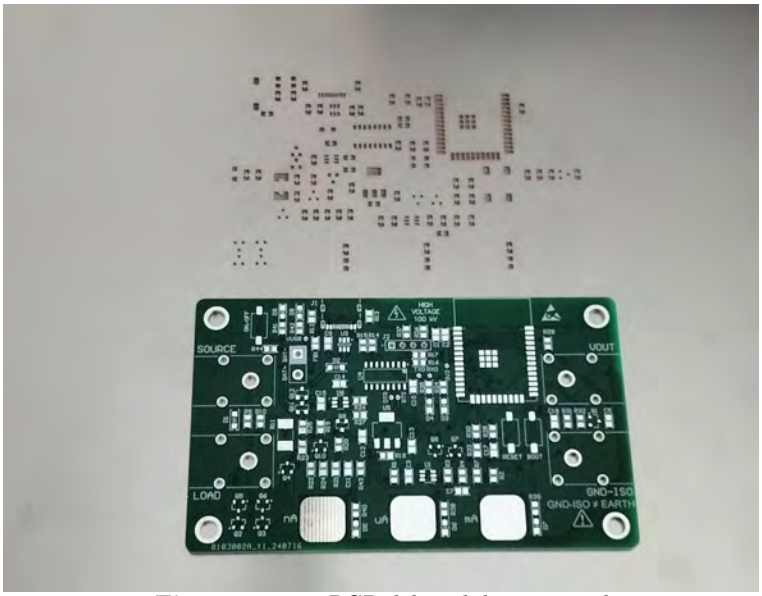



**Figura 4.45** – *PCB del medidor y stencil*

El proceso de montaje de la placa ha consistido en el uso de pasta de soldadura y horno infrarrojo. El primer paso ha sido aplicar la pasta de soldadura con ayuda del *stencil* y una espátula. Para ello, se han sujetado tanto la placa como el *stencil* con cinta adhesiva, con el objetivo de evitar desplazamientos que pudieran estropear el proceso.

**Figura 4.46** – *Aplicación de la pasta de soldadura*

Tras la pasta de soldadura, el siguiente paso ha consistido en la localización de los componentes según

su referencia y el código de la tienda. Una vez localizados, se han colocado en sus respectivas posiciones con ayuda de una pinza y un microscopio electrónico.

**Figura 4.47** – *Desempaquetado de componentes*


**Figura 4.48** – *Microscopio electrónico*

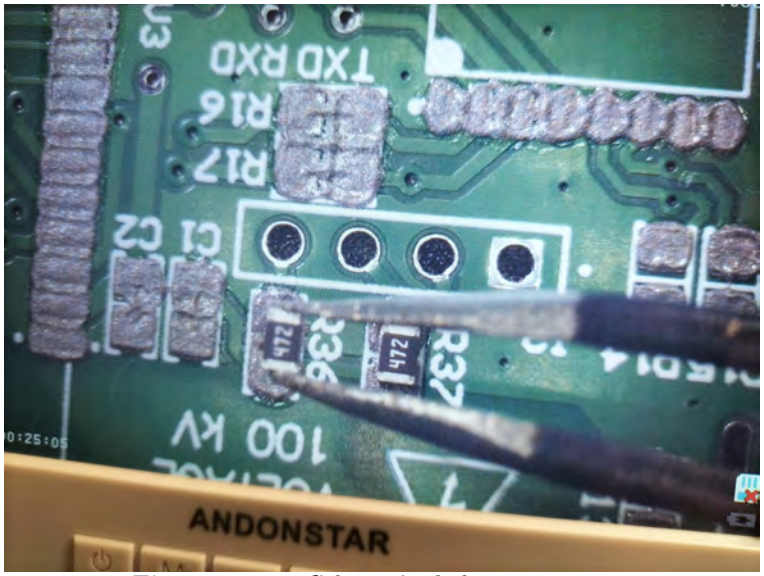



**Figura 4.49** – *Colocación de los componentes*

Tras la colocación de todos los componentes, en la figura 4.50 se puede observar la posición de la placa en la bandeja del horno.

**Figura 4.50** – *Componentes colocados antes de soldar*

Introduciendo la placa en el horno y activando el proceso, en la figura 4.51 se puede comprobar que este no dura más de seis minutos. Es necesario contar con buena ventilación, pues durante el calentamiento se desprenden gases de la pasta de soldadura que pueden ser perjudiciales para la salud.


**Figura 4.51** – *Proceso de soldadura en horno infrarrojo*

Este proceso se ha llevado a cabo para dos de las cinco placas pedidas. El resultado, tras sacar las placas del horno, dejar enfriar y limpiar con alcohol isopropílico del 99.9 %, se puede observar en la figura 4.52.



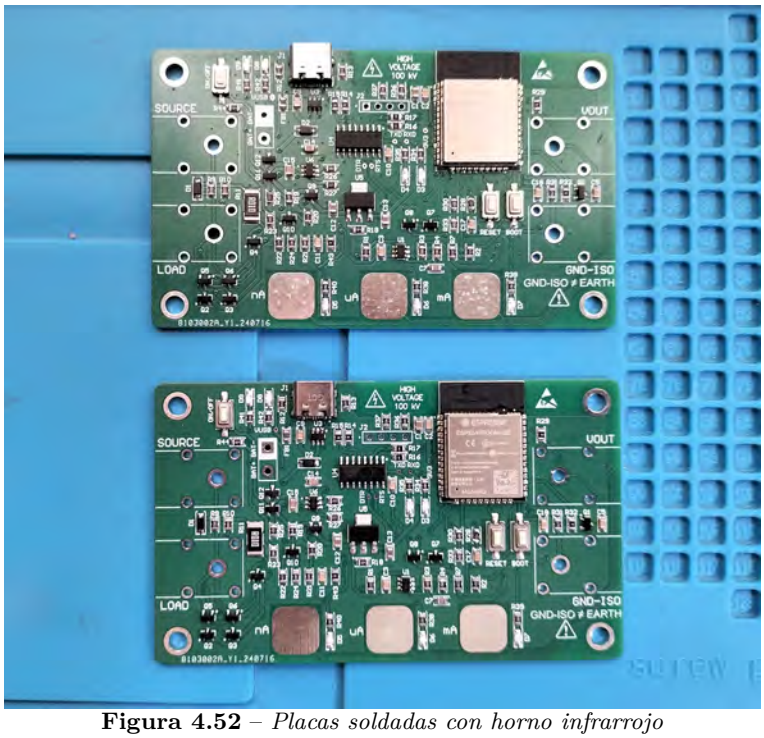



**Figura 4.52** – *Placas soldadas con horno infrarrojo*

Ha sido necesario arreglar el conector USB-C de una de las placas, ya que algunas patillas quedaron conectadas entre sí durante la soldadura. Para ello, se ha empleado fundente de soldadura junto con malla de cobre. Se ha aplicado calor utilizando un soldador manual y se han logrado eliminar los contactos.

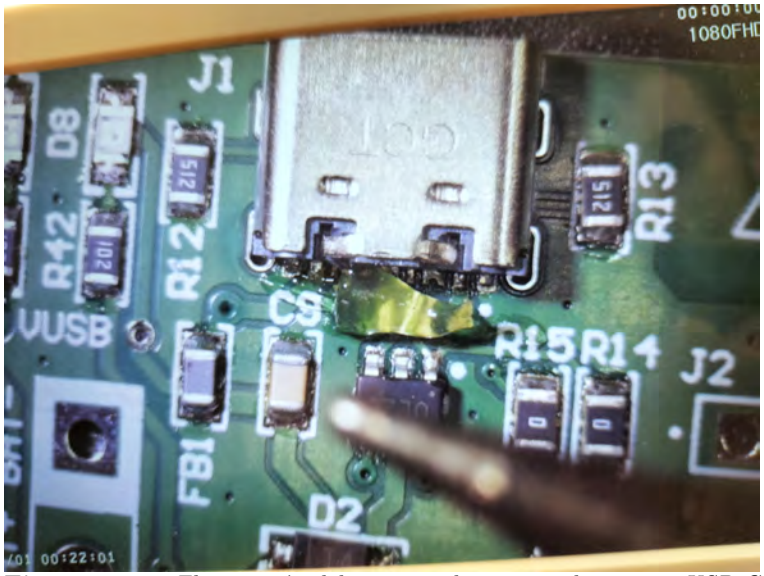



**Figura 4.53** – *Eliminación del estaño sobrante en el conector USB-C*

Tras comprobar con un polímetro la continuidad entre diferentes pistas y contactos, se ha soldado el soporte de 4 pines para la pantalla OLED y los conectores de entrada de la corriente.

Se ha conectado el cable USB al ordenador y se ha comprobado el botón de encendido. Al ser el programa el encargado de mantener el sistema encendido mediante un pin, ha sido necesario mantener el botón pulsado hasta lograr cargar el programa en la memoria del ESP32. Tras ello, se ha podido soltar el botón.

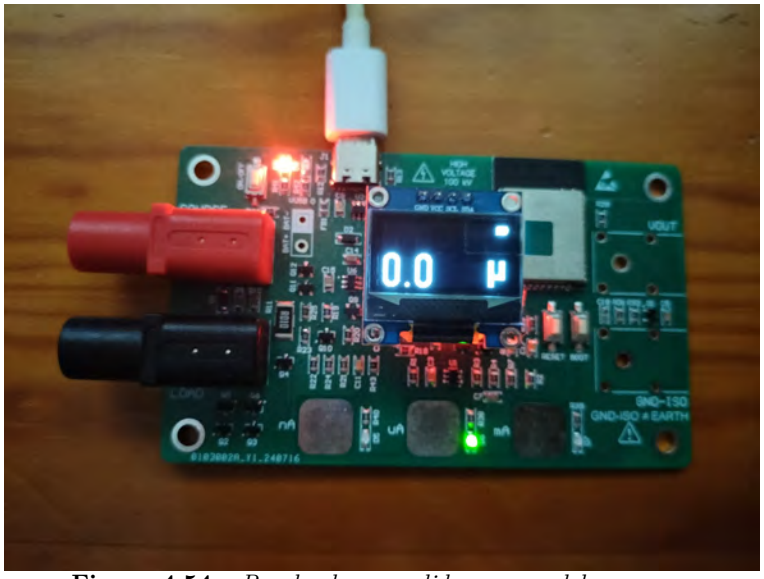



**Figura 4.54** – *Prueba de encendido y carga del programa*

De esta forma, se ha podido comprobar el funcionamiento correcto de la comunicación USB y la conversión a UART, del microcontrolador ESP32-WROOM-32E, del arranque y apagado del sistema, y de los *pads* táctiles.

El siguiente paso ha consistido en la prueba de medición de corriente continua. En este ocasión, a diferencia de las pruebas con el CurrentRanger R3, no se ha podido contar con una fuente de corriente. Para generar

una corriente constante y realizar las mediciones, se ha utilizado una placa de evaluación de ESP32. Se ha conectado uno de sus pines DAC a una serie de resistencias, con valores mucho mayores a las resistencias de medición de la placa y se han programado distintos valores de voltaje. Además, para comprobar los resultados mostrados en la pantalla, se ha calculado la corriente con un polímetro, el cual se ha conectado entre los bornes de la resistencia externa para medir el voltaje.

### 4.4.1. Diseño de la carcasa

Con el objetivo de poder manejar el dispositivo con mayor seguridad y protegerlo frente a golpes y daños externos, se ha diseñado una carcasa que englobe la placa de circuito impreso y permita acceder a los diferentes conectores, pulsar el botón de encendido y comprobar el estado de los LEDs. Para ello, se ha empleado el programa SolidWorks.

**Figura 4.55** – *Logo de SolidWorks*

La carcasa se ha diseñado con espacio suficiente para contener la batería del sistema y se han añadido cuatro soportes con agujeros para tornillos M4 para atornillar la PCB. En la parte superior, se ha dejado una ventana para la pantalla, así como un hueco para alojar el botón de encendido/apagado y otros tres huecos para poder acceder a los *pads* táctiles.

**Figura 4.56** – *Carcasa del sistema*

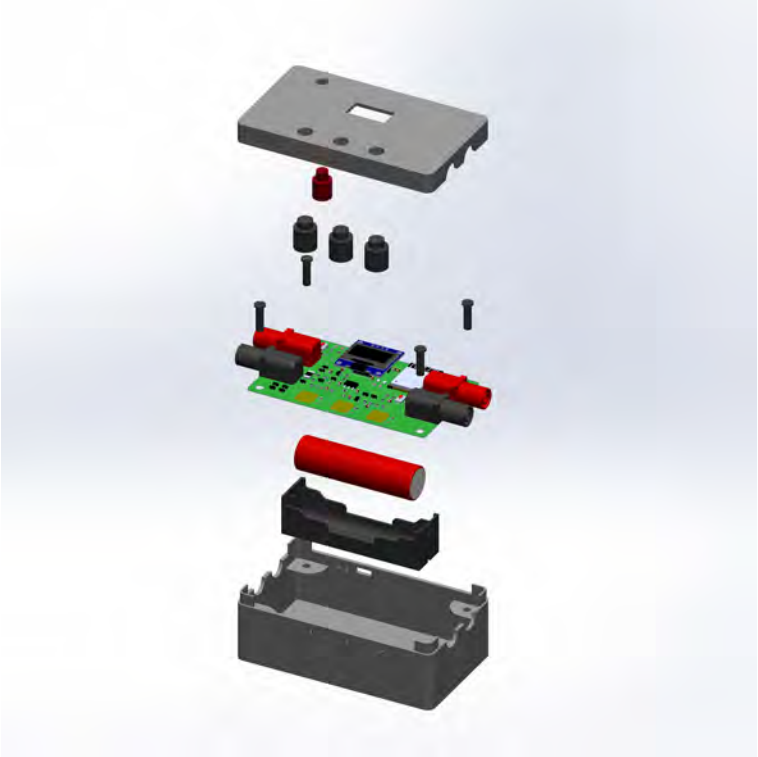


**Figura 4.57** – *Vista explosionada*

El botón de encendido es una pieza de plástico que se apoya sobre el pulsador de la PCB y sobresale ligeramente por encima de la carcasa para facilitar su uso. Debido a la estructura, cuenta con unos soportes que impiden que se pueda desplazar de su posición.

Los *pads* táctiles deben ser tocados directamente o a través de un conductor metálico para funcionar. Por ello, el material de los pulsadores de la carcasa debe ser completamente metálico o contener una parte metálica que los recorra longitudinalmente.

La carcasa cuenta con un hueco para el contector USB-C, cuatro agujeros para los conectores tipo banana y pequeñas ventanas para poder ver el estado de los LEDs que indican el modo de funcionamiento.

### 4.4.2. Librería de Python

Se ha diseñado una librería de Python para poder enviar comandos y recibir datos desde el medidor de corriente por puerto serie. Los entornos de desarrollo empleados han sido Spyder, para creación de la librería, y JupyterLab, para realizar las pruebas de comunicación y representar gráficamente los resultados. A continuación, se muestran sus funciones fundamentales.

En primer lugar, se incluyen los módulos *pySerial* y *time*, y se realiza la inicialización de la librería, como se muestra en la figura 4.58. Se declara una variable que contiene la longitud de caracteres máximos a leer en una transmisión serie.

```python
import serial
import time

class CurrentMeterESP32:
    """
    Funciones del CurrentMeter ESP32.

    Autor: Pablo Criado Asensio
    Basado en la librería CableTVAnalyzer_E8591C creada
    por José Luis Pérez Barbero para GranaSAT
    """

    def __init__(self, length_string_read = 5000):
        """
        Constructor.
        """
        self.length_string_read = length_string_read
```

**Figura 4.58** – *Inicialización de la librería*

El siguiente paso consiste en crear funciones para abrir y cerrar la comunicación con el puerto. Esto es fundamental, ya que si no se cierra el puerto al finalizar la comunicación, este se mantiene bloqueado.

```python
def Open(self, Baudrate = 115200, Port = "COM11", Timeout = 1, verbose=True):
    """

    """
    #CurrentMeterESP32_Serial_BaudRate=self.BaudRate
    self.controller= serial.Serial(Port,        # Puerto
             Baudrate,              # Baudios
             serial.EIGHTBITS,     # Payload de 8 bits
             serial.PARITY_NONE,   # Sin paridad
             serial.STOPBITS_ONE,  # 1 bit de parada
             timeout=Timeout,      # tiempo de dropout
             rtscts=False)         # Señales RTS->CTS
    time.sleep(0.5)
    self.controller.setRTS(False)
    self.controller.setDTR(False)
    if verbose:
        print("Puerto serie abierto: " + Port + ", " + str(Baudrate) + " baud, 8N1, timeout="+str(Timeout)+"s")

def Close(self, verbose=True):
    """
    Cerramos el puerto serie
    """
    self.controller.close()
    if verbose:
        print("Puerto serie cerrado")
```

**Figura 4.59** – *Funciones de pertura y cierre de la comunicación con el puerto serie*

El envío de comandos se realiza siempre empleando la misma función, mostrada en la figura 4.60. Se establecen los caracteres del comando del medidor como argumento de entrada de la misma.

```python
def _send(self, cmd, verbose=True):
    """
    Envío de mensajes al mediador. Resetea el buffer en cada llamada.
    :param cmd: comando a enviar
    :type cmd str
    """
    self.controller.reset_input_buffer()  # reset del buffer
    self.controller.write(cmd.encode() + b'\r\n') #comando+terminación

    if verbose:
        print(f"Sent: {cmd}")
```

**Figura 4.60** – *Función de envío de comandos*

Para recibir información del dispositivo, se han creado dos funciones diferentes. En primer lugar, se puede realizar una lectura de un determinado número de caracteres, tras lo cual finaliza dicha lectura. Para el caso de la lectura del búfer AC, al tener una gran cantidad de valores, se ha creado una función específica que permite realizar una lectura por líneas mucho mayor.

```python
def _read(self, verbose=True):
    """
    Recepción de mensajes del mediador.
    :returns out: mensaje captado.
    :type out str
    """
    out = self.controller.read(self.length_string_read)
    #if verbose:
        #print(f"Read: {str(out)}")
    return out.decode("utf-8")

def _read_buffer(self, verbose=True):
    """
    Recepción de mensajes del mediador.
    :returns out: mensaje captado.
    :type out str
    """
    values = [0]*self.sample_number
    i = 0
    while i < self.sample_number:
        line = self.controller.readline()
        values[i] = float(line.decode())
        i += 1
    #if verbose:
        #print(f"Read: {str(out)}")
    return values
```

**Figura 4.61** – *Funciones de lectura de datos del medidor*

Por último, en la figura 4.62 se muestran dos ejemplos de funciones para enviar comandos y recibir la respuesta del dispositivo. En el primer caso, se envía el comando "ACMEAS", el cual devuelve solo una línea con el número de milisegundos que se han necesitado para hacer la medición. La segunda función envía "ACBUFFER", y se devuelve el búfer completo de mediciones.



```python
def ACmeasurement(self, sample_number = 20000, verbose=True):
    """
    Esta función envía el comando "ACMEAS" y devuelve el tiempo de medición en ms

    """
    self.sample_number = sample_number
    cmd = "ACMEAS"
    self.controller.reset_input_buffer()
    self._send(cmd, verbose)
    return self._read()

def ACbuffer(self, verbose=True):
    """
    Esta función envía el comando "ACBUFFER" y devuelve todos los valores
    del buffer AC

    """
    cmd = "ACBUFFER"
    self.controller.reset_input_buffer()
    self._send(cmd, verbose)
    return self._read_buffer()
```

**Figura 4.62** – *Ejemplo de envío de comandos*

## 4.5. Pruebas del medidor de corriente

### 4.5.1. Pruebas de medición

El primer paso ha consistido en la realización de una medición con corriente continua. En este caso, al no disponer de un generador de corriente continua, se ha utilizado una fuente de voltaje en serie con una resistencia de gran valor para poder obtener corrientes muy pequeñas. Se ha comparado el valor mostrado en la pantalla con el medido por un polímetro en los conectores de salida, obteniendo valores muy similares y confirmando el funcionamiento del sistema.

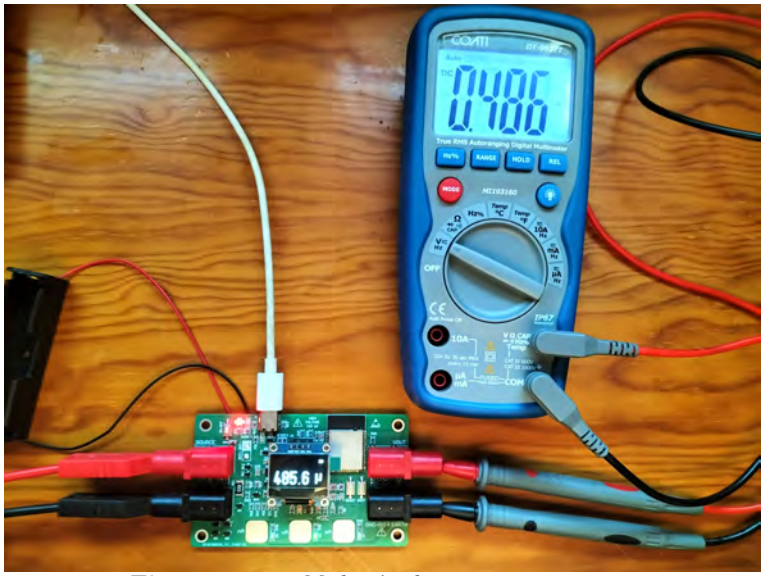



**Figura 4.63** – *Medición de corriente continua*

Además, se ha probado el modo de operación "BIAS", el cual permite medir corrientes negativas. Esto se muestra en la figura 4.64.

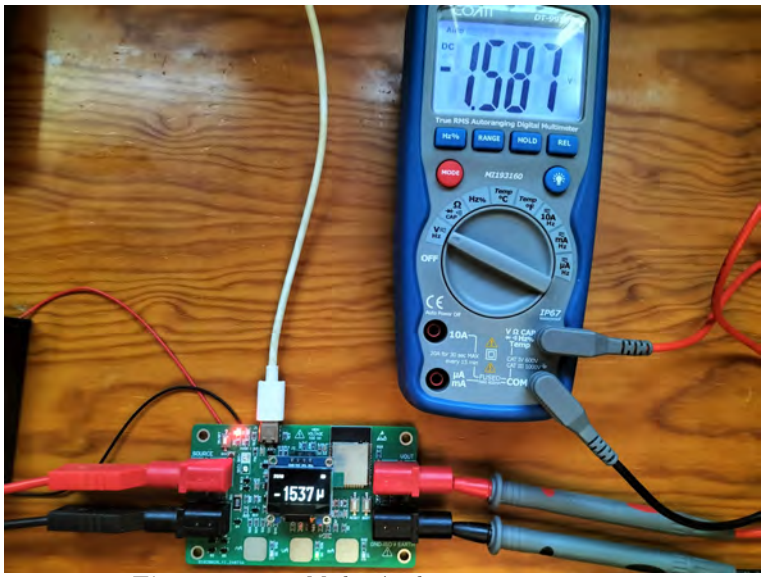



**Figura 4.64** – *Medición de corriente negativa*

De la misma manera que se realizaron las pruebas con el CurrentRanger R3, se ha utilizado el módulo *Plotly* de Python para hacer una representación gráfica de los valores medidos.

Al no contar con una fuente de corriente para realizar las pruebas, se ha utilizado una placa de evaluación con ESP32 para generar formas de onda similares a las generadas por la fuente de corriente. De esta manera, las mediciones para una corriente con forma de onda de sierra, senoidal y cuadrada se muestran a continuación.

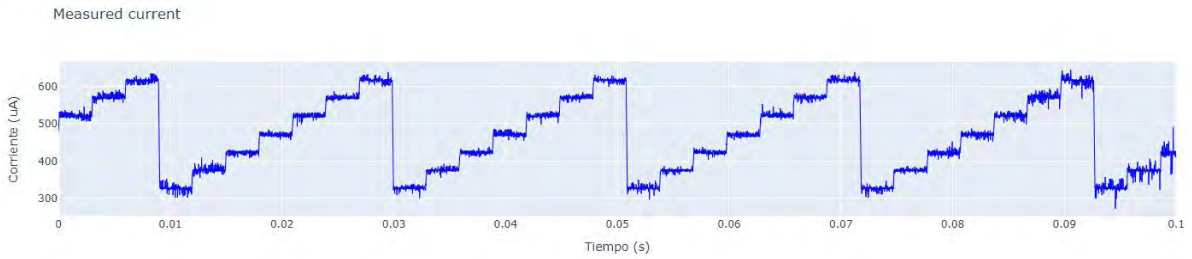



**Figura 4.65** – *Medición diente de sierra*

En la figura 3.29 se mostró el búfer de las mediciones del CurrentRanger. Si se realiza una comparación, se puede observar que el medidor de corriente con ESP32 es capaz de almacenar un número de datos mucho mayor, permitiendo de esta forma visualizar las señales con más información. La frecuencia de medida también es más elevada, ya que se miden 30000 muestras en 100 ms, es decir, de 300000 muestras por segundo, un 56 % más que las 192000 muestras por segundo del CurrentRanger.

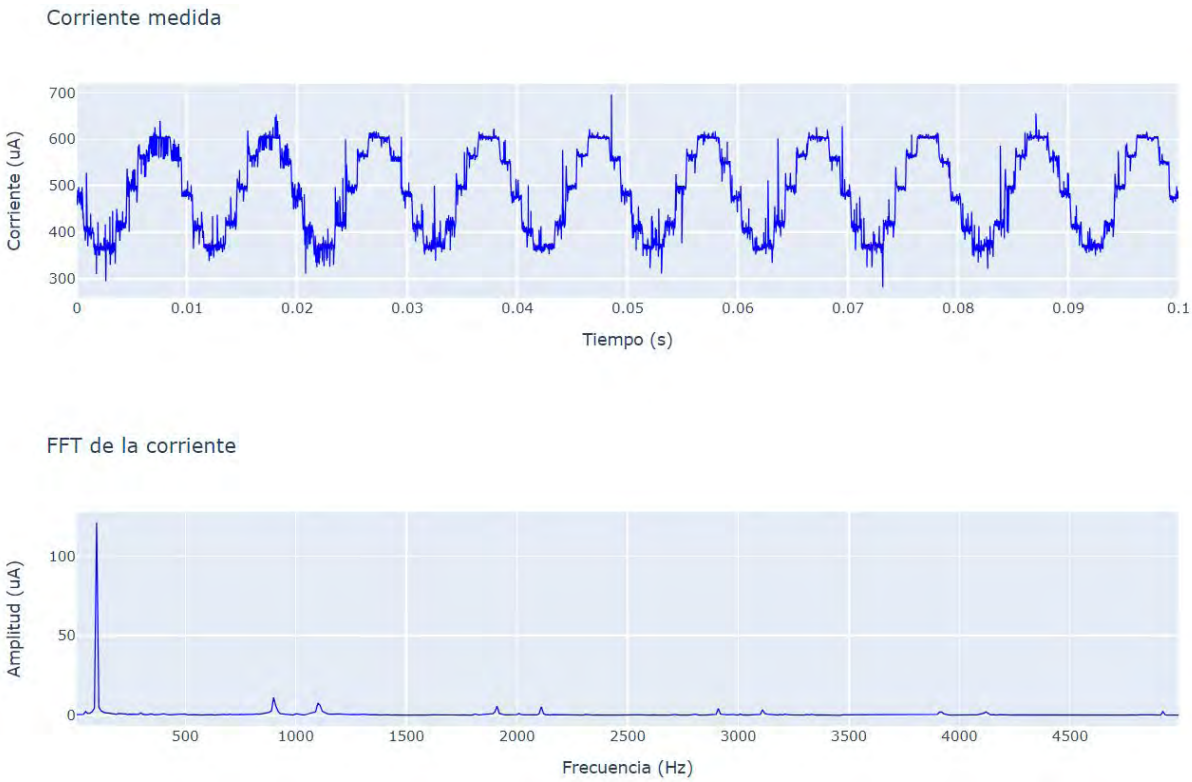



**Figura 4.66** – *Medición onda senoidal*

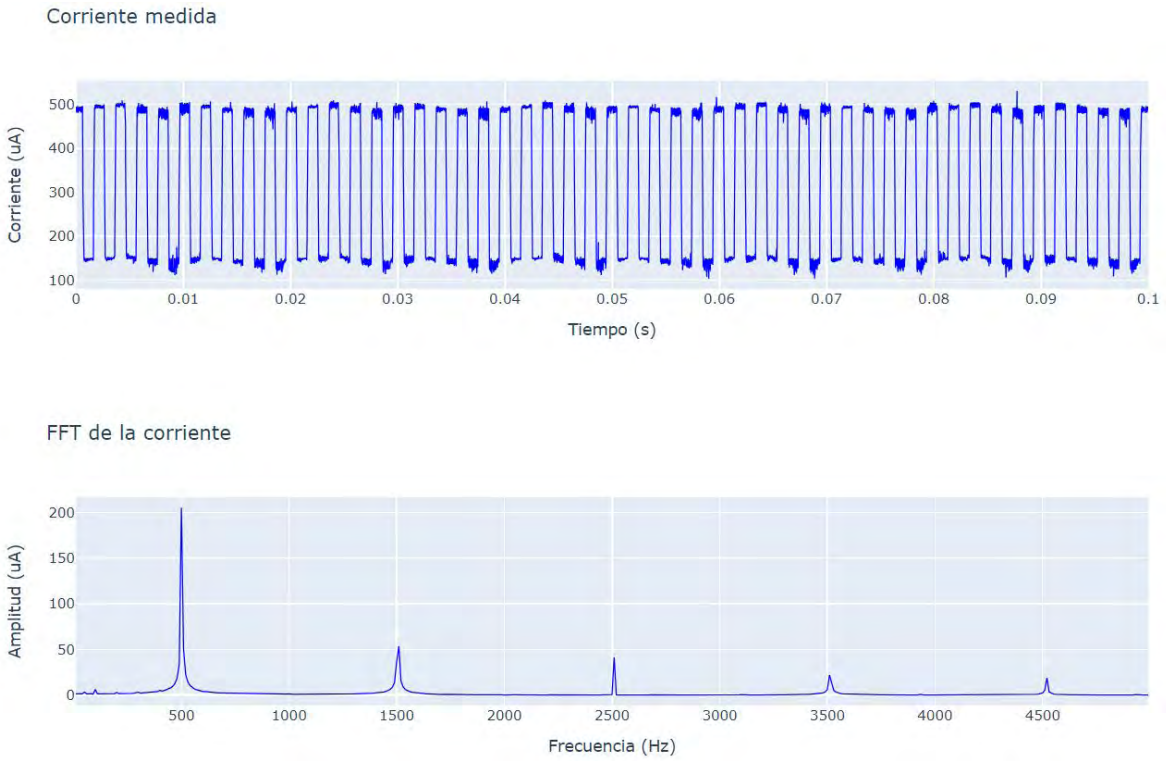


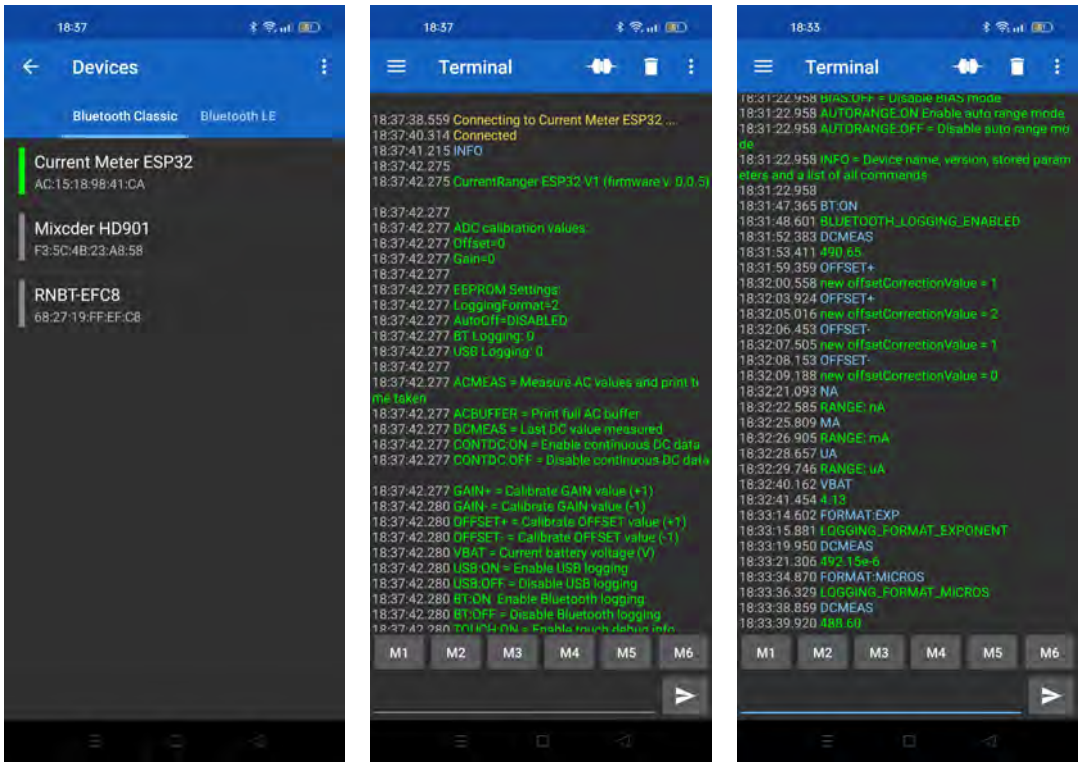

**Figura 4.67** – *Medición onda cuadrada*

Al realizar la transformada rápida de Fourier de las formas de onda senoidal y cuadrada, se puede obtener una gran cantidad de información sobre las componentes en frecuencia de las mismas, tanto de la componente fundamental como de los armónicos. Además, las interferencias generadas por la placa de evaluación ESP32 son perfectamente captadas por el medidor de corriente, lo cual ayuda a confirmar su utilidad.

El resto de comandos del medidor de corriente funcionan del mismo modo que los comandos del CurrentRanger modificados en el apartado 3.4. Si se habilita el registro por puerto serie con USB, se pueden enviar los valores de las mediciones DC, voltaje de la batería, valores de los sensores táctiles, etc. Los únicos comandos que presentan diferencias son los que modifican los valores de *offset* y *gain*.

### 4.5.2. Comunicación por Bluetooth

La aplicación de *smartphone* utilizada para realizar las pruebas de comunicación por Bluetooth ha sido Serial Bluetooth Terminal. Esta aplicación permite conectarse a un dispositivo Bluetooth y enviar o recibir información a través de una consola sencilla. Se ha seleccionado el dispositivo en el menú, se ha establecido la conexión y se han enviado distintos comandos para recibir las respuestas del sistema.




**(a)** *Descubrimiento en Bluetooth del medidor* **(b)** *Conexión y prueba de envío y recepción* **(c)** *Pruebas con distintos comandos*

**Figura 4.68** – *Pruebas de comunicación Bluetooth*

En la figura 4.68 se pueden observar las respuestas del sistema ante distintos comandos de variación de parámetros, cambios de modo y mediciones. Los comandos enviados se muestran en azul, mientras que la información recibida desde el medidor se muestra en verde.

La comunicación con Bluetooth permite controlar el medidor de la misma forma que la comunicación por USB. Todos los comandos son válidos en ambos modos.

### 4.5.3. Pruebas de la batería

Con el objetivo de poder utilizar el medidor de corriente sin necesidad de cable USB, como ya se ha comentado anteriormente, se ha implementado un sistema de carga de una celda de litio. Por ello, se ha seleccionado una batería del laboratorio y se han realizado pruebas con un dispositivo cargador de baterías.

El instrumento empleado para poder comprobar el estado y carga de la batería seleccionada para el trabajo ha sido el ALC-8500, un centro de carga de baterías de 4 canales. EL ALC-8500 se puede observar en la figura 4.69.


**Figura 4.69** – *Cargador de baterías ALC-8500*

Para controlar el cargador de baterías, el fabricante proporciona un software, llamado ChargeProfessional, el cual permite configurar todos sus parámetros de funcionamiento. Para acceder al software, es necesario realizar una conexión USB entre el ALC-8500 y el ordenador donde se encuentra instalado, así como elegir el puerto COM correspondiente.




**(a)** *Icono del software* **(b)** *Conexión con el equipo*

**Figura 4.70** – *Software ChargeProfessional para el ALC-8500*

El ALC-8500 cuenta con 2 canales que permiten la carga de baterías hasta 30 V y 5 A, y 2 canales que permiten la carga hasta 15 V y 1 A. Para cada canal, es posible configurar el tipo de batería y su capacidad, número de celdas, función de carga o descarga, corrientes diferentes para carga y descarga, y factor de capacidad. Además, cuenta con una base de datos donde se puede guardar la configuración para usos posteriores. La pantalla de configuración se muestra en la imagen 4.71.

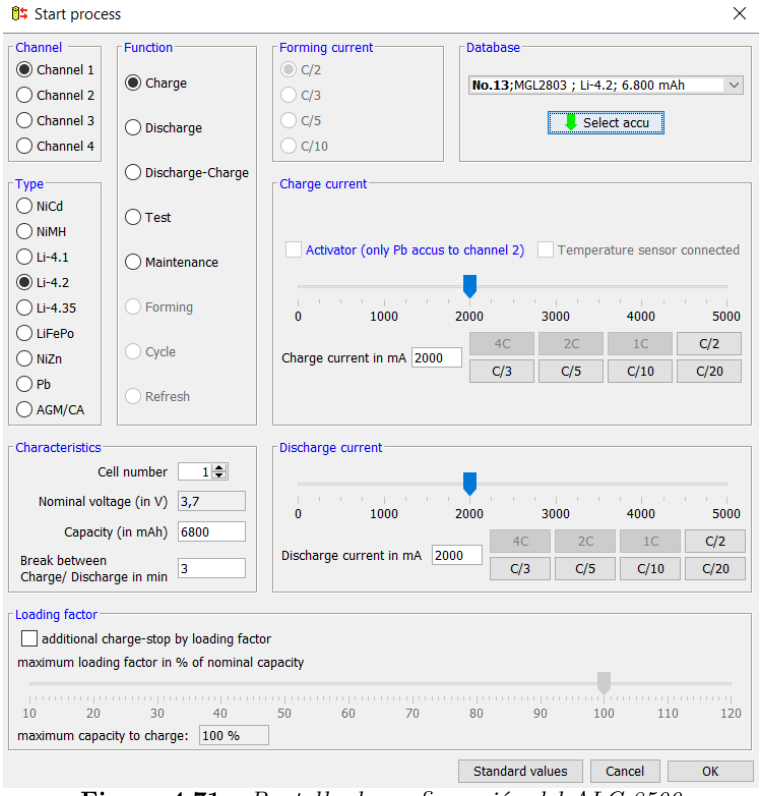
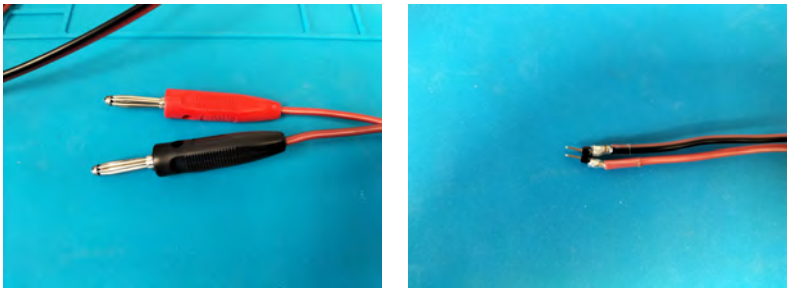



**Figura 4.71** – *Pantalla de configuración del ALC-8500*

Para hacer pruebas del funcionamiento del cargador, se ha empleado una batería de Li-Ion de 3.75 V y capacidad de 6800 mAh, con voltaje de carga de 4.2 V. La batería se puede observar en la figura 4.72.


**Figura 4.72** – *Batería de prueba del ALC-8500*

La batería se ha conectado al cargador utilizado dos cables con conectores tipo banana en uno de los extremos, mientras que en el otro se han soldado dos pines con estaño y se han asegurado con tubos de plástico termorretráctil, empleando una estación de soldadura de aire caliente, para evitar cortocircuitos. Estos pines han sido necesarios para poder realizar la conexión con la batería.



**(a)** *Conectores tipo banana* **(b)** *Conector con pines soldados*

**Figura 4.73** – *Cables para la batería*

Finalmente, se ha elegido el canal 1 del ALC-8500, ya que permite una corriente lo suficientemente elevada para cargar la batería. En la figura 4.74 se puede observar la situación de la batería durante la carga, así como la pantalla del cargador, donde se muestran los valores de voltaje, corriente y modo de funcionamiento.


**Figura 4.74** – *Conexión de la batería al ALC-8500*

En primer lugar, se ha realizado una descarga de la batería a corriente constante de 2 A. El software monitoriza continuamente la tensión y corriente durante su funcionamiento, permitiendo así crear una gráfica con sus valores en función del tiempo.

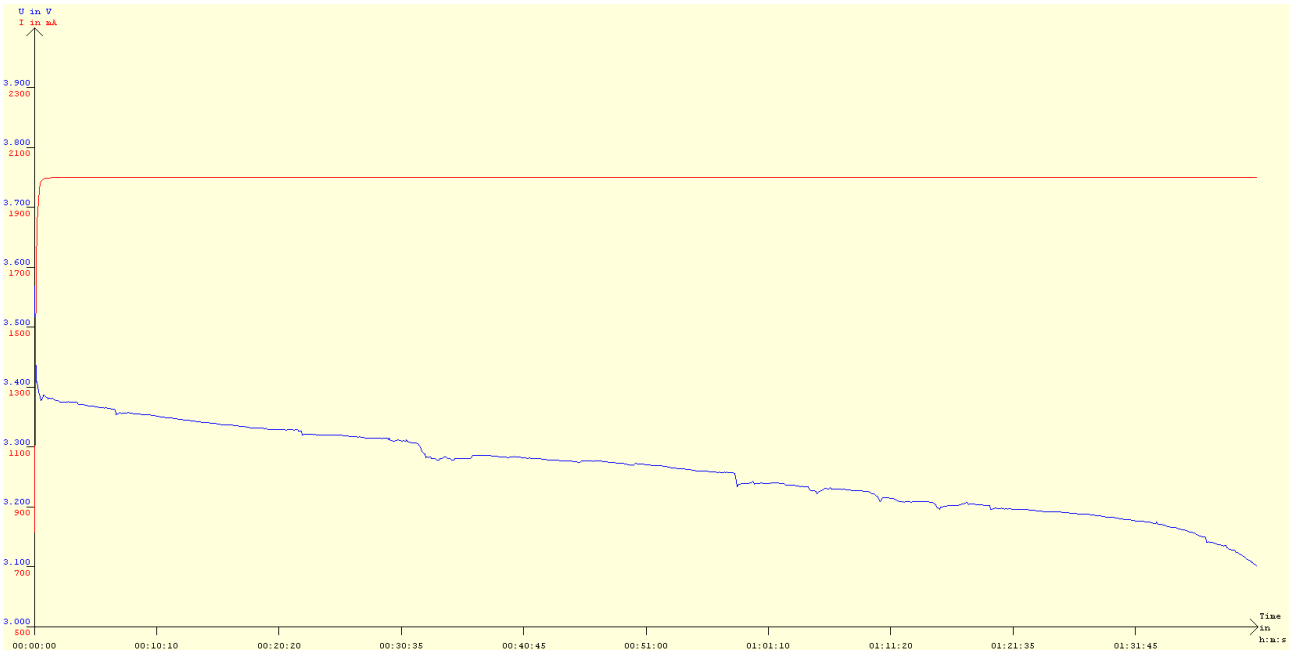



**Figura 4.75** – *Descarga de la batería*

Como se observa en la figura 4.75, el voltaje de la batería se va reduciendo con el tiempo, mientras que la corriente permanece constante. A pesar de no estar completamente cargada, la gran capacidad de la batería ha permitido generar una gran cantidad de corriente durante más de una hora y media.

Una vez finalizado el proceso de descarga, se ha probado la carga de la batería, con el objetivo de obtener su capacidad real. En este caso, debido al gran tiempo necesario para la carga completa, ha sido necesario realizar dos intervalos de carga en diferentes momentos. El comienzo de la carga se muestra en la figura 4.76, mientras que la carga completa se alcanza en la figura 4.77.


**Figura 4.76** – *Inicio de la carga de la batería*

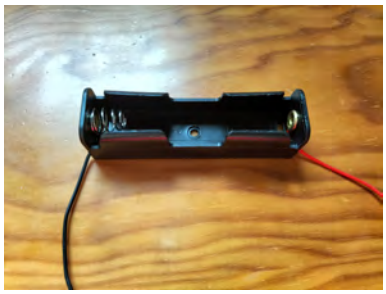

**Figura 4.77** – *Final de la carga de la batería*

Se puede observar que el tiempo total de carga ha superado 5 horas. Para obtener la capacidad total de la batería, hay que integrar las curvas de las corrientes. El software ChargeProfessional permite la descarga de todos los valores del proceso en formato CSV, por lo que, si se realiza en Python el cálculo de la suma de todos los valores de corriente y la división entre el tiempo total, se obtiene una capacidad de 4977 mAh. Este número es bastante menor a los 6800 mAh nominales, pero hay una serie de motivos que explican esta

diferencia. En primer lugar, la batería, según la información mostrada en su etiqueta, fue fabricada en el año 2015, por lo que tiene una antigüedad de casi 10 años. Además, ha sido utilizada habitualmente para distintas aplicaciones en el laboratorio, lo cual reduce aún más su capacidad. Por último, hay que tener en cuenta que, por protección, el software ha cortado la carga cuando la corriente se ha reducido a menos de 250 mA, cerca del 10 %. de la corriente máxima de carga establecida. Si se hubiera permitido continuar la carga, la capacidad final de la batería obtenida habría sido mayor.

Tras estas pruebas, aunque la celda se encuentra en condiciones aceptables para su uso, se ha decidido descartarla debido a su gran tamaño, peso y excesiva capacidad. En su lugar, se ha optado por utilizar una batería de nueva adquisición de litio BRC 18650 de 3000 mAh, con un porta baterías adecuado.


**(a)** *Batería de litio 18650*



**(b)** *Porta baterías*

**Figura 4.78** – *Batería del medidor de corriente*



Como se vio en el apartado 4.1.16, el consumo medio de corriente del sistema es de 175 mA. Teniendo en cuenta este valor y la capacidad de la batería, el tiempo total de funcionamiento con batería se puede estimar de la siguiente manera:

$$\textbf{Duración}_{\textbf{batería}} = \frac{\textbf{3000 mAh}}{\textbf{175 mA}} = \textbf{17 h} \tag{4.5.1}$$

Por lo tanto, el medidor podría mantenerse funcionando durante 17 h sin necesidad de recargar la batería si se mantuviera encendido todo el tiempo.

El siguiente paso consistiría en comprobar la capacidad real de la celda elegida para el sistema. Sin embargo, después de tomar la decisión del cambio de batería, por cuestión de fecha no se ha podido utilizar el ALC-8500 para hacer pruebas de carga y descarga. El proceso a llevar a cabo habría sido el mismo que para la batería estudiada anteriormente, realizando ciclos de descarga y carga para comprobar la capacidad real de la misma y su estado.

Tras realizar la soldadura del porta baterías a la placa de circuito impreso, se ha colocado la celda de litio y se ha encendido la placa.

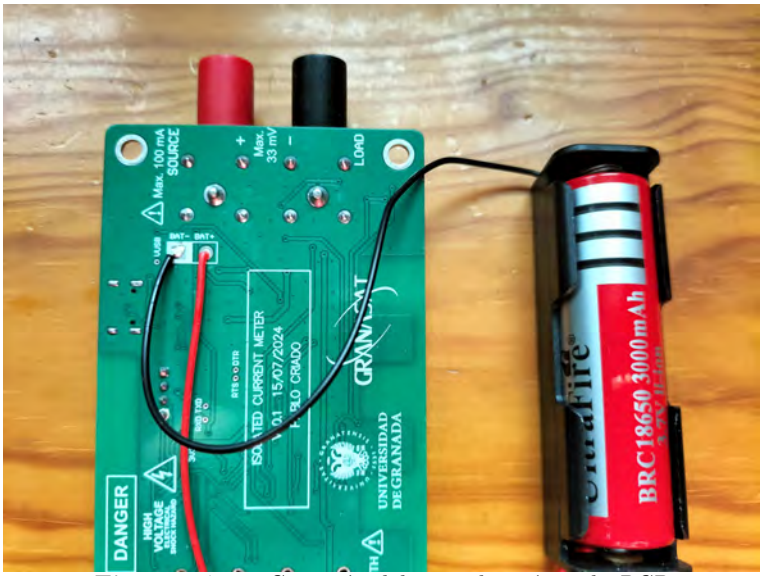



**Figura 4.79** – *Conexión del porta baterías a la PCB*

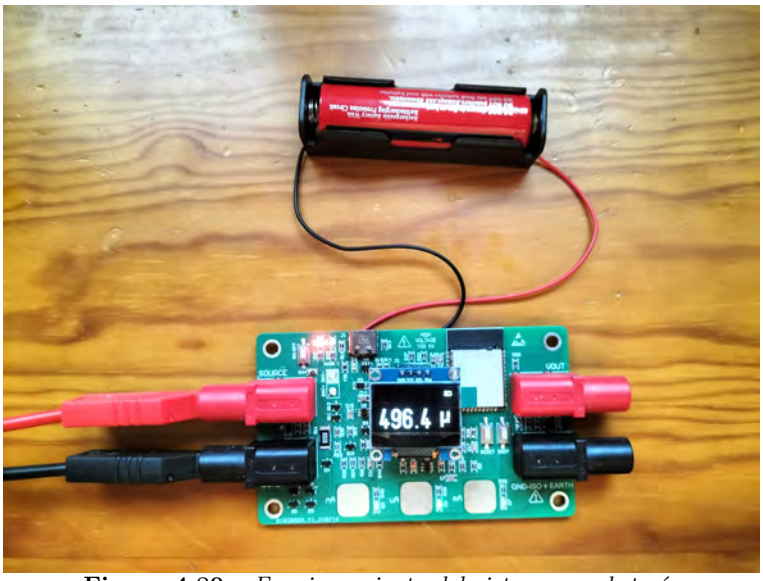



**Figura 4.80** – *Funcionamiento del sistema con batería*

Si se conecta el cable USB, la batería pasa a cargarse y se activa el LED de aviso de carga situado junto al LED de encendido.

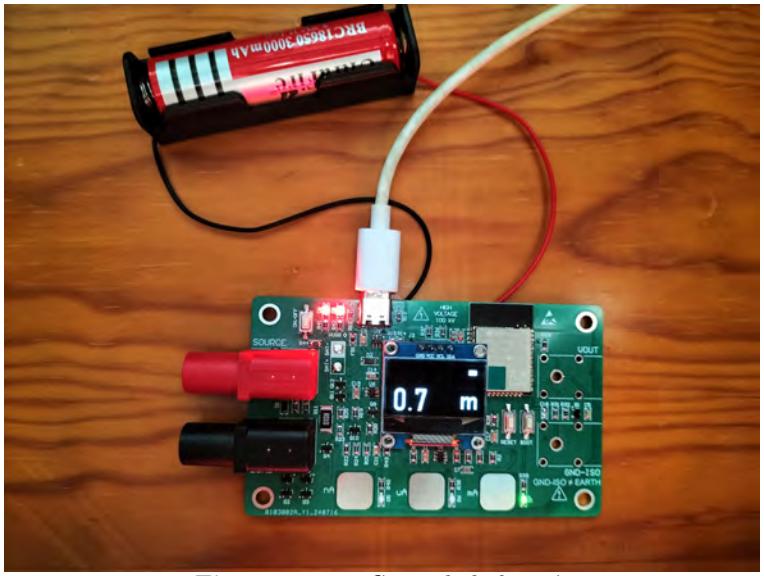



**Figura 4.81** – *Carga de la batería*

Midiendo el voltaje en los terminales de la batería con un polímetro, se ha visto que la carga finaliza cuando el voltaje de esta se encuentra muy próximo a 4.2 V. Por lo tanto, se ha comprobado que el cargador funciona correctamente.

Al utilizar el medidor con la batería, ha surgido un problema al intentar encender el sistema. Cuando no está conectado el cable USB y se pulsa el botón de encendido, hay ocasiones en las que el medidor no enciende. En estos casos, es necesario conectar el cable USB, encender el sistema y volver a desconectar el cable, permaneciendo de este modo encendido. Las posibles causas y soluciones a este problema se explican más adelante, en el apartado de soluciones y mejoras futuras del sistema.



### 4.5.4. Comparación entre CurrentRanger y medidor de corriente con ESP32

Como se ha visto a lo largo de este capítulo y en el capítulo 3, tanto el CurrentRanger R3 como el rediseño del medidor de corriente con ESP32 tienen una serie de ventajas y desventajas. En este apartado se van a comentar cada una de sus características y se va a realizar una comparación de sus prestaciones.

En primer lugar, antes de comparar el funcionamiento de los sistemas, son destacables las diferencias de tamaño y disposición de las placas de circuito impreso. El CurrentRanger tiene un tamaño de 8 x 5 cm, mientras que el medidor con ESP32 mide 6 x 10 cm. Aunque un mayor tamaño puede suponer una desventaja a la hora de usar el dispositivo en algunas aplicaciones, también permite realizar un proceso de soldadura más sencillo con componentes más grandes, buscar componentes defectuosos en caso de avería, y manejar la PCB con más facilidad durante las pruebas. Respecto a la pantalla OLED, el CurrentRanger no la presenta centrada en la placa, sino que se encuentra ligeramente desplazada hacia un lado. Esto puede suponer un problema a la hora de fabricar una carcasa, además de resultar menos estético. En el rediseño se ha colocado la pantalla en el eje central de la placa para solucionarlo.

Los conectores de tipo banana del CurrentRanger no presentan una buena sujeción, ya que las tuercas de sujeción se aflojan con el uso y pueden conllevar una caída de los conectores, lo cual presenta un peligro en algunas circunstancias. En el rediseño del medidor, en cambio, se han empleado conectores tipo banana soldados a la placa y dirigidos hacia los lados, para evitar que los cables estorben al usar el dispositivo con las manos.

Mientras que el CurrentRanger utiliza un conector USB Micro-B, en el rediseño se ha empleado un conector USB-C, ya que es el estándar actual y prácticamente todos los productos comerciales de nuevo

diseño incluyen este tipo de conectores y cables.

El medidor de corriente con ESP32 incorpora Bluetooth de forma integrada gracias al microcontrolador. Sin embargo, en el CurrentRanger habría sido necesario adquirir un módulo de Bluetooth y realizar la soldadura de las conexiones para hacerlo funcionar. Esto añade una dificultad mayor a su uso después de la compra.

Una vez encendido el sistema, la principal diferencia se encuentra en el funcionamiento del ADC. El SAMD21 del CurrentRanger incorpora más opciones y parámetros para configurar sus ADCs que el ESP32, lo cual significa una mayor precisión de las medidas si se realiza una calibración adecuada. Sin embargo, si se utiliza el procolo I2S, el ESP32 es capaz de medir voltajes con una velocidad significativamente mayor que el SAMD21, 300000 muestras por segundo frente a 200000.

La memoria interna del ESP32 es mayor que la del SAMD21. Como se ha visto anteriormente, se permite de este modo almacenar una gran cantidad de datos en el búfer, hasta 30000, para poder visualizar correctamente las formas de onda de alta frecuencia. El CurrentRanger, en cambio, no es capaz de almacenar más de 5000 valores.

El doble núcleo del ESP32 permite utilizar las tareas del sistema operativo FreeRTOS con eficiencia, lo cual aumenta la velocidad de ejecución del programa. Además, la frecuencia de reloj es más rápida que la del SAMD21, 80 MHz frente a 48 MHz. Esto hace que el ESP32 ejecute su programa de manera mucho más rápida.



### 4.5.5. Soluciones y mejoras futuras del medidor de corriente

- El primer error detectado al montar la placa y probar su funcionamiento ha consistido en que el microcontrolador no ha sido capaz de iniciar el programa automáticamente al recibir alimentación. Tras analizar los esquemáticos y buscar información al respecto, se ha decidido desoldar el condensador C18, el cual se puede observar en la figura 4.17. Este condensador influye en los pulsos que indican al microcontrolador que debe entrar en modo *bootloader*. Al no tener el pin GPIO0 resistencia de *pull-up* externa, como es el caso del botón ENABLE, el condensador afecta a los tiempos de los pulsos. Realizando la desoldadura, se ha comprobado que el microcontrolador inicia el programa correctamente al recibir alimentación.
- Por diseño, es el microcontrolador el que mantiene encendido el sistema al poner la puerta del transistor de alimentación a masa. A pesar de que esto facilita el diseño de la función de encendido y apagado, supone un problema a la hora de realizar la programación del microcontrolador, ya que hay que mantener el botón de encendido pulsado para que no se corte la alimentación durante el proceso. Una posible solución para un diseño futuro, manteniendo este mismo sistema de encendido, podría ser añadir un puente para pines, como el mostrado en la figura 4.82, conectado en paralelo con el botón de encendido. De esta forma, se podría programar el microcontrolador sin necesidad de mantener pulsado el botón.

**Figura 4.82** – *Puente para pines*

- El circuito del botón de encendido/apagado, mostrado anteriormente en la figura 4.8, no es capaz de

iniciar el sistema cuando se está utilizando la batería. Con cable USB, a pesar de funcionar en la mayoría de las ocasiones, hay veces donde tampoco logra permitir la alimentación. En la figura 4.83 se muestra una posible solución a este problema. Anteriormente, el divisor de tensión formado por las resistencias R44 y R23 se empleaba tanto para la puerta del NMOS como para el pin del ESP32. En este caso, se ha cambiado para que el divisor proteja solo al pin y no afecte al voltaje de la puerta del transistor. La comprobación de esta solución se realiza al conectar momentáneamente con un cable la pista VS y la resistencia R22 en la PCB fabricada, confirmando que el circuito enciende de esta manera.

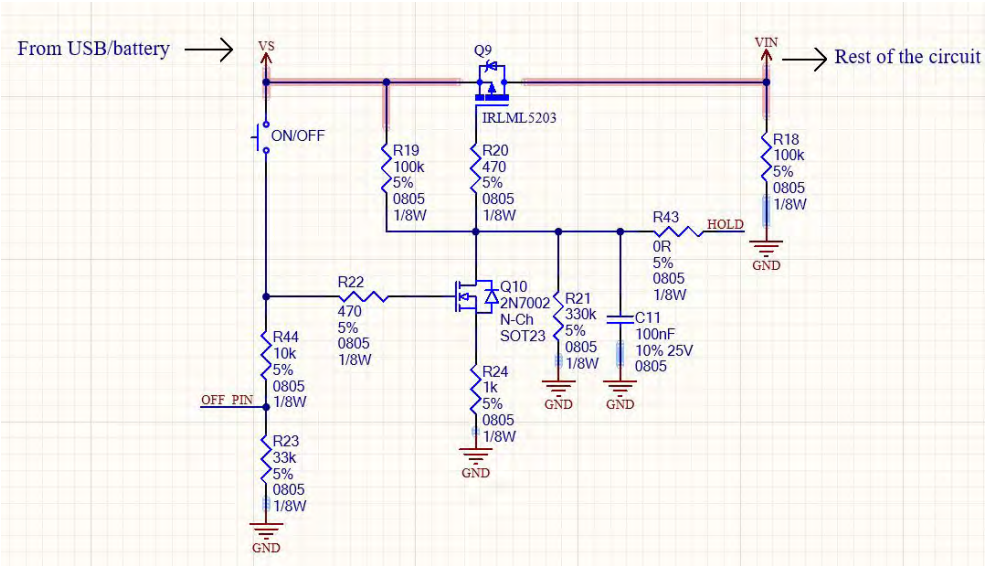



**Figura 4.83** – *Solución para el botón de encendido/apagado*

- El medidor de corriente tiene poca precisión en las zonas donde el voltaje que cae en las resistencias es muy bajo. Esto ocurre cuando se desean medir corrientes menores a 20 nA, $\mu$A o mA. Para solucionar esto, se propone la implementación de ramas con resistencias de medida de valores intermedios a las actuales. Por ejemplo, se puede elegir un conjunto de resistencias de 300 k$\Omega$, 10 k$\Omega$, 300 $\Omega$, 10 $\Omega$, 300 m$\Omega$ y 10 m$\Omega$. De esta forma cada rango pasa a tener dos ramas, empleada cada una con diferentes intensidades y seleccionada automáticamente por el microcontrolador, con un amplio margen para evitar cambios continuos. Si circula una corriente de 1 $\mu$A, con la resistencia de 300 $\Omega$ caen 300 $\mu$V, mucho mayor que los 10 $\mu$V actuales. La principal desventaja de esta solución es la necesidad de utilizar más pines del microcontrolador para controlar las nuevas ramas, además de requerir un mayor gasto en componentes.

# Capítulo 5

# Conclusiones y líneas futuras

Este Trabajo Fin de Máster ha permitido estudiar con profundidad un medidor comercial para pequeñas corrientes, hacer un análisis de su hardware y software y buscar las posibilidades de mejora del mismo. Con la información obtenida, se ha hecho un rediseño del medidor, donde se ha cambiado el microcontrolador original por un ESP32 con capacidades inalámbricas, se ha implementado un conector USB-C y se ha logrado almacenar una mayor cantidad de datos de medida. Además, se ha probado un sistema de carga y funcionamiento con baterías de litio de 3.7 V.

Los errores detectados durante las pruebas del sistema, a pesar de afectar a algunas funcionalidades del mismo, principalmente en lo relativo al botón de encendido, son fácilmente resolubles en versiones posteriores de la placa de circuito impreso.

Por lo tanto, se han cumplido todos los requerimientos solicitados al comienzo de este trabajo. A continuación, se realiza una recapitulación de los mismos.

| **Ref.** | **Requerimientos de hardware** |
|---|---|
| H.1 | Comunicación Bluetooth. ✓ |
| H.2 | Comunicación USB 2.0. ✓ |
| H.3 | Botón para encendido/apagado. ✓ |
| H.4 | Pantalla OLED con interfaz I2C. ✓ |
| H.5 | Alimentación mediante USB-C (5 V). ✓ |
| H.6 | Alimentación mediante batería Li-Ion (3.7 V) y cargador USB-C. ✓ |
| H.7 | Medición de corrientes entre nA y mA. ✓ |
| H.8 | Protección contra alto voltaje de entrada. ✓ |
| H.9 | Conexiones para medir de forma externa el voltaje de salida. ✓ |

**Tabla 5.1** – *Recapitulación de requerimientos de hardware*

| Ref. | Requerimientos de firmware |
|---|---|
| F.1 | Control y lectura por puerto serie. ✓ |
| F.2 | Control y lectura por Bluetooth. ✓ |
| F.3 | Buffer de mediciones para obtener señales de alta frecuencia. ✓ |
| F.4 | Almacenamiento de variables de funcionamiento en memoria no volátil. ✓ |
| F.5 | Librería de Python para controlar el dispositivo y descargar los valores de las mediciones. ✓ |

**Tabla 5.2** – *Recapitulación de requerimientos de firmware*

| Ref. | Requerimientos mecánicos |
|---|---|
| M.1 | Carcasa para proteger la PCB que facilite las conexiones y el manejo por el usuario. ✓ |
| M.2 | Conectores tipo banana para entrada de corriente y salida de voltaje. ✓ |
| M.3 | Conector USB-C. ✓ |
| M.4 | Pantalla que muestre el estado del medidor. ✓ |
| M.5 | Tornillos M4 para sujeción de la PCB a la carcasa. ✓ |
| M.6 | Soporte para batería en la carcasa. ✓ |

**Tabla 5.3** – *Recapitulación de requerimientos mecánicos*

Para continuar el proyecto en un futuro, es necesario realizar una revisión de los errores detectados durante las pruebas de funcionamiento, con el fin de poder fabricar una segunda versión del medidor. Por otro lado, el sistema no ha podido ser probado en una aplicación real de acelerador de partículas, lo cual demostraría su verdadera utilidad. Por ello, sería muy interesante llevar a cabo todas las pruebas necesarias en este sentido cuando el grupo GranaSAT tenga a su disposición el acelerador de partículas mencionado en la introducción de este trabajo. Al tener acceso directo por cable USB-C al sistema de programación del microcontrolador ESP32, se facilita de este modo realizar actualizaciones y cambios al firmware en el futuro, a fin de mejorar el funcionamiento del sistema.

5

# Bibliografía


[1] H. Wiedemann, *Particle Accelerator Physics*. Springer, 4th ed., February 2015.

[2] LowPowerLab, "Current Ranger," 2018. Disponible en https://lowpowerlab.com/guide/currentranger/.

[3] Microchip, *ATSAMD21G18*, 2021. Disponible en https://www.microchip.com/en-us/product/atsamd21g18.

[4] Infineon, *IRLML5203*, 2004. Disponible en https://www.infineon.com/cms/en/product/power/mosfet/p-channel/irlml5203/.

[5] "Keysight IO Libraries Suit," 2024. Disponible en https://www.keysight.com/zz/en/lib/software-detail/computer-software/io-libraries-suite-downloads-2175637.html.

[6] Espressif, "ESP Modules," 2024. Disponible en https://www.espressif.com/en/products/modules.

[7] Espressif, *ESP32-WROOM-32E*, 2020. Disponible en https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf.

[8] Texas Instruments, *BQ21040*, 2016. Disponible en https://www.ti.com/product/es-mx/BQ21040.

[9] Diodes Incorporated, *AP7361C*, 2020. Disponible en https://www.diodes.com/assets/Datasheets/AP7361C.pdf.

[10] Espressif, *ESP32_DevKitc_V4*, 2017. Disponible en https://dl.espressif.com/dl/schematics/esp32_devkitc_v4-sch.pdf.

[11] Analog Devices, *MAX4239*, 2020. Disponible en https://www.analog.com/media/en/technical-documentation/data-sheets/max4238-max4239.pdf.

[12] costaud (GitHub), *Touch Sensor Application Note*, 2019. Disponible en https://github.com/ESP32DE/esp-iot-solution-1/blob/master/documents/touch_pad_solution/touch_sensor_design_en.md.

# Apéndice A

# Diagrama de Gantt

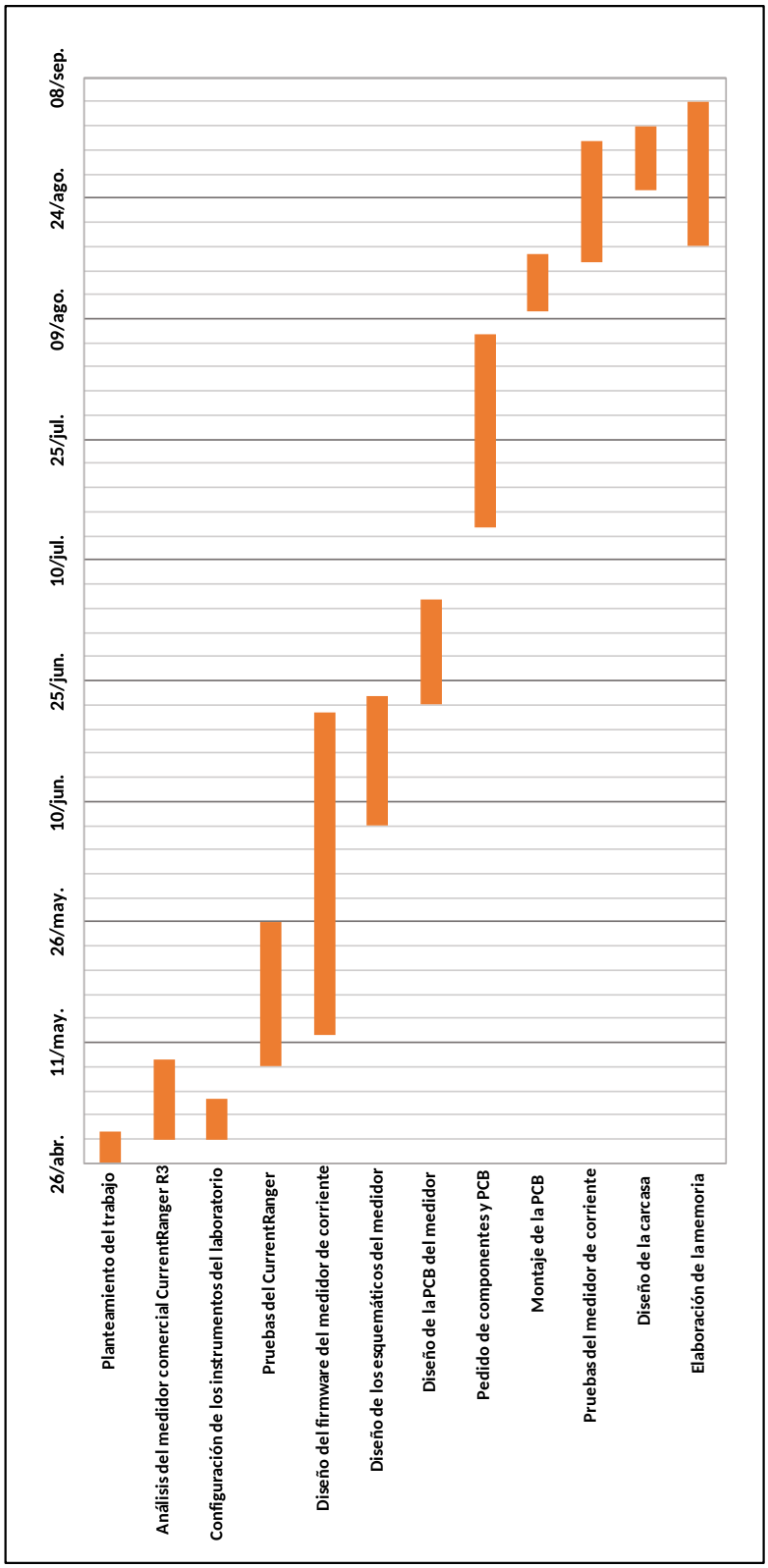



**Figura A.1** – *Diagrama de Gantt del proyecto*

# Apéndice B

# Presupuesto del proyecto

## B.1. Material

#### B.1.0.1. Instrumentación

| **Concepto** | **Precio (€)** |
|---|---|
| Probador de baterías | 100 |
| Horno infrarrojo | 200 |
| Osciloscipio | 180 |
| Polímetro | 50 |
| Pasta de soldadura | 20 |
| Generador de corriente | 200 |
| **Total** | 650 |

**Tabla B.1** – *Presupuesto de los instrumentos y equipos necesarios para el desarrollo del proyecto*

#### B.1.0.2. Fabricación

| **Concepto** | **Precio (€)** |
|---|---|
| 5 PCBs | 50 |
| *Stencil* | 10 |
| Componentes y conectores | 70 |
| Carcasa | 10 |
| **Total** | 205 |

**Tabla B.2** – *Presupuesto de fabricación y ensamblaje del sistema*

## B.2. Recursos humanos

| **Concepto** | **Precio (€)** |
|---|---|
| Ingeniero Junior (350 horas) | 5000 |
| Ingeniero Senior (Supervisión) | 2000 |
| **Total** | 7000 |

**Tabla B.3** – *Presupuesto de recursos humanos*

# Apéndice C

# Lista de materiales

| **Componente** | **Cantidad (1 placa)** | **Código de LCSC** | **Cantidad total** |
|---|---|---|---|
| Pulsador | 3 | C318938 | 20 |
| Cap 22 $\mu$F | 1 | C98190 | 20 |
| Cap 100 nF | 8 | C476766 | 50 |
| Cap 10 pF | 1 | C344177 | 100 |
| Cap 10 $\mu$F | 2 | C5189822 | 50 |
| Cap 1 $\mu$F | 2 | C376929 | 100 |
| Diodo 1N4148W | 2 | C909967 | 50 |
| Diodo LED rojo | 4 | C19171391 | 100 |
| Diodo LED verde | 3 | C19171393 | 100 |
| Núcleo de ferrita | 1 | C304328 | 100 |
| USB-C | 1 | C3025063 | 5 |
| ESP32-WROOM-32E | 1 | C5361945 | 5 |
| NMOS 2N7002 | 9 | C916396 | 100 |
| PMOS IRLML5203 | 3 | C5364310 | 20 |
| Res 220 $\Omega$ | 2 | C2933537 | 100 |
| Res 180 k$\Omega$ | 1 | C2828755 | 20 |
| Res 18 k$\Omega$ | 1 | C2828729 | 20 |
| Res 2 k$\Omega$ | 1 | C706274 | 10 |
| Res 10 k$\Omega$ | 4 | C706400 | 5 |
| Res 10 $\Omega$ | 1 | C319927 | 10 |
| Res 0.01 $\Omega$ | 1 | C5127860 | 5 |
| Res 5.1 k$\Omega$ | 2 | C2930296 | 100 |
| Res 0 $\Omega$ | 7 | C2907288 | 100 |
| Res 100 k$\Omega$ | 3 | C2907293 | 100 |
| Res 470 $\Omega$ | 2 | C2907329 | 100 |

| | | | |
|---|---|---|---|
| Res 330 kΩ | 2 | C2907320 | 100 |
| Res 1 kΩ | 9 | C2907295 | 100 |
| Res 1 MΩ | 1 | C17514 | 100 |
| Res 27 Ω | 1 | C2907314 | 100 |
| Res 2 MΩ | 1 | C26112 | 100 |
| Res 4.7 kΩ | 2 | C2907326 | 100 |
| MAX4238 | 1 | C239900 | 5 |
| USBLC6-2SC6 | 1 | C2827654 | 10 |
| CH340C | 1 | C84681 | 5 |
| AP7361C | 1 | C500795 | 5 |
| BQ21040 | 1 | C202311 | 5 |

**Tabla C.1** – *Lista de materiales de la PCB*

# Apéndice D

# Archivos Gerber para la fabricación

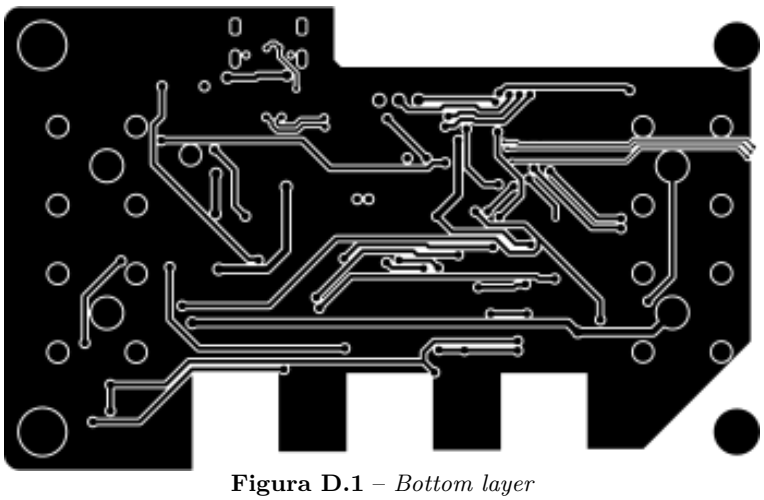


**Figura D.1** – *Bottom layer*

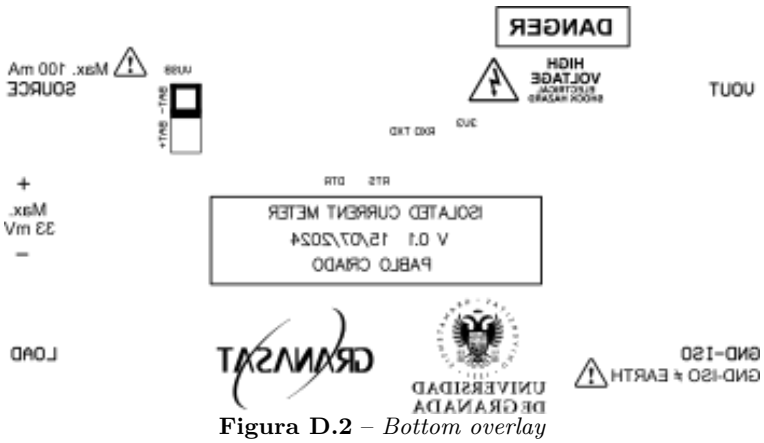



**Figura D.2** – *Bottom overlay*

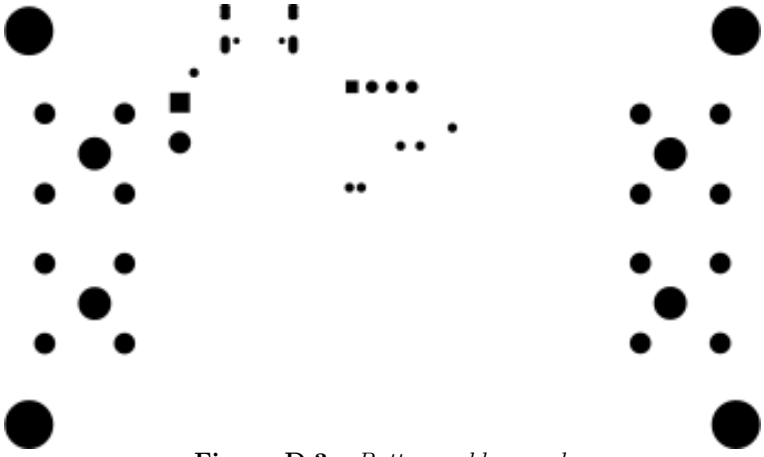


**Figura D.3** – *Bottom solder mask*

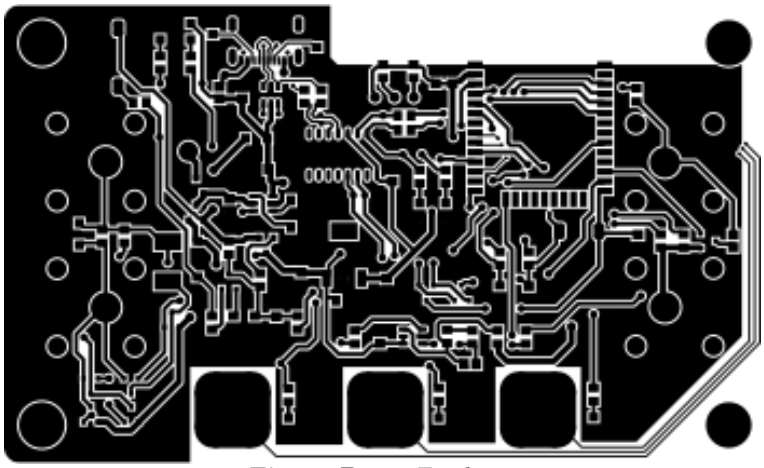


**Figura D.4** – *Top layer*

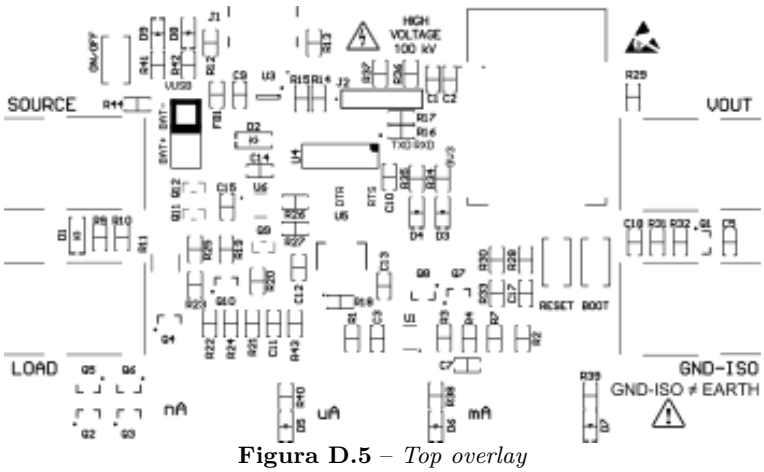



**Figura D.5** – *Top overlay*

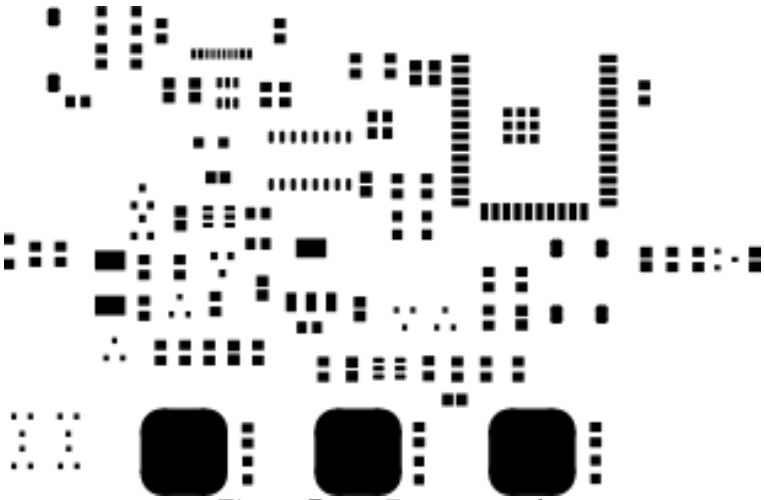


**Figura D.6** – *Top paste mask*

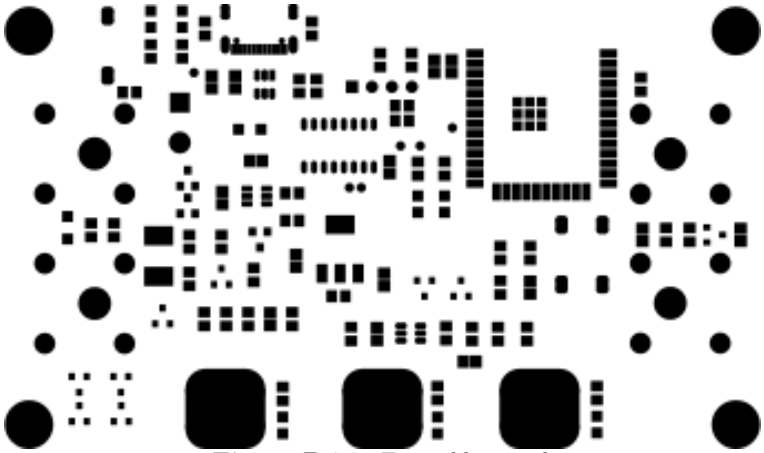


**Figura D.7** – *Top solder mask*