ESCOLA SUPERIOR DE TECNOLOGIA I CIÈNCIES EXPERIMENTALS
Grado en Ingeniería Eléctrica

TRABAJO FIN DE GRADO

Desarrollo y programación de una herramienta para ensayo de baterías y aplicación a simulación de sistemas fotovoltaicos con almacenamiento

Realizado por
Javier González Barreda
Castellón, julio de 2023

Dirigido por
Emilio Pérez Soler

Realizado en el departamento de
Ingeniería de Sistemas Industriales y Diseño

A mi tutor, Emilio. Por todo el tiempo dedicado y el apoyo mostrado en todo momento, transformando este proyecto en un reto personal.

A mis padres, Javier y Carmina. Por su sacrificio y paciencia todos estos años.

Resumen

El presente trabajo, llevado a cabo bajo la beca de investigación asociada al proyecto: "Disseny d'estratègies optimitzades de gestió de bateries per participar en els serveis complementaris del sistema elèctric", se enfoca en el desarrollo de una herramienta versátil y fácil de usar para realizar ensayos de carga y descarga en baterías, así como para simular sistemas fotovoltaicos con almacenamiento de energía.

La herramienta propuesta está compuesta por diferentes componentes clave. En primer lugar, se cuenta con una fuente de alimentación programable que actúa como una fuente de intensidad, permitiendo controlar la carga de la batería de manera precisa. De la misma forma, se incluye una carga programable que actúa como un sumidero de intensidad, lo que posibilita controlar la descarga de la batería de forma exacta.

Para medir la tensión en el sistema, se utiliza una tarjeta Arduino y un convertidor analógico/digital (ADC) que garantizan una medición precisa y confiable. Esta configuración asegura que la medida de tensión sea común a todo el sistema, evitando posibles errores o interferencias.

Todos estos componentes se comunican con un ordenador que actúa como interfaz hombre-máquina (HMI), proporcionando una interfaz simple para el usuario final.

La versatilidad de la herramienta radica en la posibilidad de conectar los dispositivos de diferentes modos, lo que permite al sistema adoptar diferentes configuraciones según los requisitos específicos de cada tarea. Esto garantiza una flexibilidad óptima para llevar a cabo una amplia variedad de ensayos y simulaciones.

El software desarrollado en Python desempeña un papel crucial en la funcionalidad del sistema. A través de este software, se pueden realizar diferentes funciones según las indicaciones del usuario. Además, se ha llevado a cabo una serie de experimentos con diversos tipos de baterías utilizando esta herramienta, lo que ha permitido caracterizar su comportamiento y validar todas las funcionalidades implementadas.

En resumen, este proyecto ofrece una solución integral para realizar ensayos de carga y descarga en baterías, así como para simular sistemas fotovoltaicos con almacenamiento de energía. La herramienta desarrollada es intuitiva y fácil de utilizar, y se ha validado mediante pruebas que demuestran su eficacia, funcionalidad y fiabilidad. Se espera que esta herramienta sea de gran utilidad para futuras investigaciones en el campo del almacenamiento de energía, permitiendo al usuario aprovechar todas sus capacidades sin necesidad de conocer los detalles técnicos de su implementación.

Índice general

Índice de figuras

Índice de cuadros

Parte I

Memoria

Índice

1. Introducción

1.1. Antecedentes y motivación

Actualmente, las energías renovables y, en concreto, la tecnología fotovoltaica están en un momento de auge debido a varios factores. Por un lado, el desarrollo de esta tecnología esta haciendo que cada vez sea más rentable y atractiva tanto para autoconsumo, como para las grandes productoras.

Por otro lado, el aumento de los precios de la energía está repercutiendo no solo en el consumo de los hogares y la industria, sino también en toda la producción energética. La subida del precio del gas ha incrementado igualmente el precio de la electricidad, lo cual ha terminado impulsando el uso de la energía solar en todo el país.

Según los datos oficiales de Red Eléctrica Española,la potencia solar fotovoltaica se ha triplicado en los últimos 3 años, pasando de 4767 MW a principios de 2019 a 15190 MW a finales del 2021, una cifra que se estima será superior en 2022, pues a finales de agosto ya se contabilizaban 13100 MW. En otras palabras, la energía solar ha pasado de representar 3.55 % del total de energía producida a un 8.05 %. Y parte de ese incremento tiene como base esencial el autoconsumo.

Una de las claves de este aumento ha sido la derogación del conocido como "impuesto al Sol"aprobado en 2018. El tributo, establecido 3 años antes, grababa la conexión a la red eléctrica, una decisión que se tradujo en pérdidas irrecuperables para inversores e importantes gastos para los consumidores particulares que habían optado por esta energía limpia.

Otro de los factores que ha propulsado el autoconsumo ha sido el incremento de ayudas y subvenciones gubernamentales, unas medidas que originalmente estaban encaminadas a alcanzar los objetivos de generación renovables marcados por las instituciones comunitarias. Los fondos Next Generation de la UE iniciados a finales de 2021, están siendo el principal catalizador para la financiación de proyectos de instalaciones fotovoltaicas, tanto en lo que hace referencia a grandes inversores como a pequeños usuarios particulares.

Sin embargo, como ya se ha comentado, si hay un detonante que ha favorecido la energía solar, ha sido el incremento del precio de la electricidad, cuyo promedio se ha multiplicado por 5 en los últimos 3 años, como puede verse en la figura 1.1.

Esa escalada de los precios de la energía también ha provocado un incremento importante del autoconsumo energético, una práctica que puede acarrear un importante ahorro en la factura energética y que cada vez gana más adeptos. Tal y como informan desde la empresa SolarProfit, dedicada a la instalación de paneles fotovoltaicos, "la energía solar fotovoltaica se ha optimizado mucho durante los últimos años y cada vez es más económica".

La reciente guerra en Ucrania y la importancia del gas ruso para toda la Unión Europea ha provocado que los gobiernos de la Unión se planteen alternativas más limpias (y económicas a largo plazo). Lo cierto es que la implantación de energías renovables

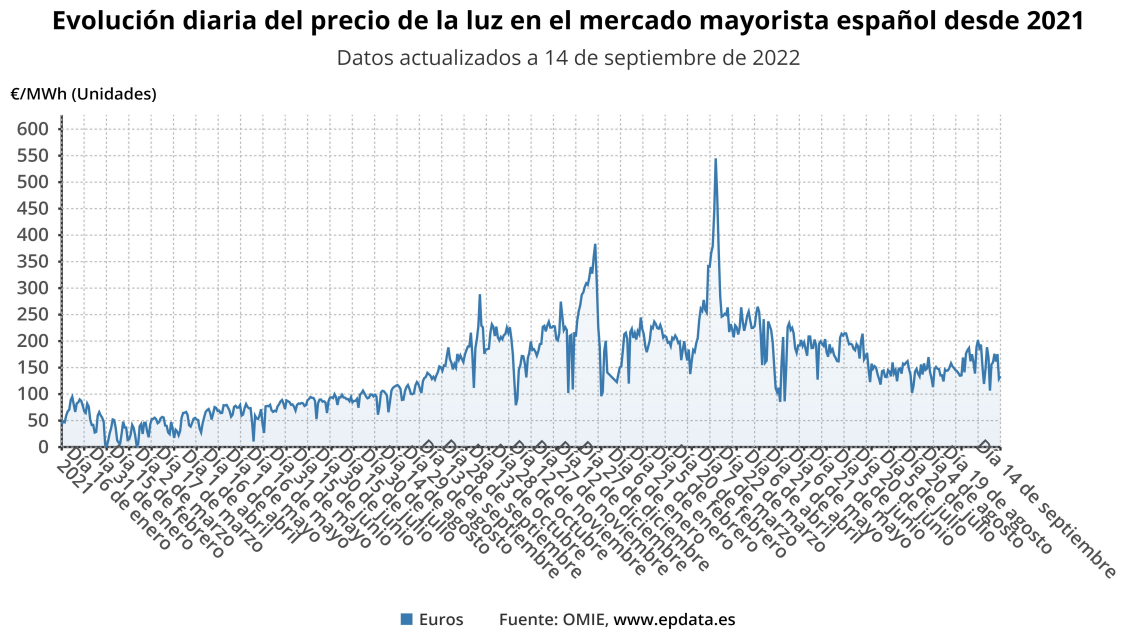


Figura 1.1: Evolución diaria del precio de la luz en el mercado mayorista español desde 2021

ha experimentado un auge en toda la Unión Europea, no solo en España. En 2021, el peso de estas energías creció un 30 por ciento en toda Europa, un dato que confirma la tendencia alcista que se materializó en 2020, cuando las energías no contaminantes superaron por primera vez a los combustibles fósiles en el mix energético. La descarbonización de la economía no tiene vuelta atrás, y la energía del Sol adquiere cada vez un papel más determinante.

Sin embargo, uno de los principales problemas que presenta la tecnología fotovoltaica es su intermitencia. Es decir, la producción fotovoltaica no puede estar totalmente controlada ya que depende de factores externos como la ubicación, el clima y la hora del día, lo cual la hace incapaz de sustentar una red eléctrica por sí sola.

Una de las soluciones a este problema es el almacenamiento de energía. Hay diversos sistemas de almacenamiento de energía, pero el uno de los más utilizados y que actualmente está en continuo en desarrollo, es el almacenamiento con baterías.

Una batería eléctrica o acumulador eléctrico (figura 1.2) es un dispositivo que consiste en dos o más celdas electroquímicas que pueden convertir la energía química almacenada en corriente eléctrica. Cada celda consta de un electrodo positivo, o cátodo, un electrodo negativo, o ánodo, y electrolitos que permiten que los iones se muevan entre los electrodos, permitiendo que la corriente fluya fuera de la batería para alimentar un circuito eléctrico.

Y aunque las baterías ofrecen una serie de ventajas muy importantes, actualmente, todavía poseen una serie de carencias o desventajas (vida media corta, número limitado de cargas, elevado coste, sobrecalentamiento, rendimiento de trabajo en frío...) que hacen que no sea una tecnología rentable en la aplicación a sistemas fotovoltaicos.

Sin embargo, las baterías son una tecnología con mucho potencial y mucho margen de

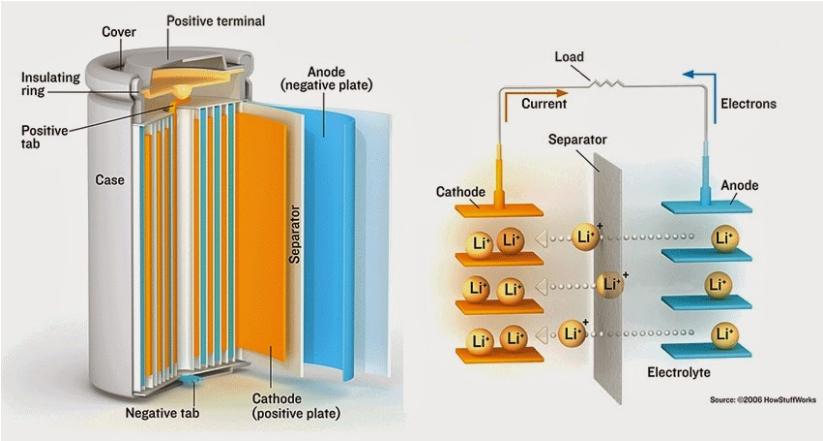


Figura 1.2: Celda ion-litio

mejora. Por esta razón, la importancia de la investigación y el desarrollo de este tipo de tecnología y el motivo de la realización de este proyecto.

Durante mi estancia en el Departamento de Ingeniería de Sistemas Industriales y Diseño de la Universitat Jaume I, primero realizando las prácticas y, más tarde, colaborando como becario de investigación, mis tareas se han centrado en desarrollar y programar una herramienta que permite realizar ensayos de carga y descarga de manera totalmente controlada en baterías, y aplicarlo a la simulación de sistemas fotovoltaicos con almacenamiento de energía.

De esta forma, esta herramienta puede ser un instrumento muy útil para el departamento en futuras investigaciones de estrategias optimizadas de gestión de baterías para participar en los servicios complementarios del sistema eléctrico, de algoritmos de gestión de sistemas de fotovoltaica con almacenamiento o de estimación del estado de carga de baterías, entre otras. También, en aplicaciones didácticas de la universidad e, incluso, en otras aplicaciones más prácticas, como la previsión del comportamiento de instalaciones fotovoltaicas.

1.2. Objetivo

El presente proyecto tiene como objetivo principal configurar y programar una herramienta que permita realizar ensayos de carga y descarga en cualquier tipo de batería y simular sistemas de fotovoltaica (FV) con almacenamiento de energía de forma intuitiva y sencilla, de modo que pueda ser usada en futuras investigaciones por un personal que no necesariamente tendrá que conocer cómo está implementado el sistema para poder utilizarlo. Igualmente, realizar una serie de experimentos, empleando la misma herramienta, con los que caracterizar algunos tipos de batería y validar todas las funcionalidades de la herramienta, las cuales se enumeran a continuación.

- Ensayo de descarga
- Ensayo de carga
- Descarga siguiendo un perfil
- Carga siguiendo un perfil

- Descarga constante
- Carga constante
- Simulación de un sistema de FV con curvas
- Simulación de un sistema de FV con perfiles
- Lectura de tensión
- Calibración de la fuente de alimentación
- Ensayo de calibración

1.3. Descripción general del proyecto

El sistema para ensayo de baterías y simulación de sistemas fotovoltaicos está formado por una fuente de alimentación programable que actúa como una fuente de intensidad; una carga programable que actúa como un sumidero de intensidad; un sistema de medida de tensión formado, a su vez, por una tarjeta Arduino y un convertidor analógico/digital (ADC), de modo, que la medida de tensión sea común a todo el sistema; y una batería. Estos dispositivos se comunican con un ordenador, el cual se utiliza para programar los diversos dispositivos y, además, actúa como interfaz hombre-máquina (HMI). Con el objetivo de realizar las diferentes funciones indicadas por el usuario, estos equipos se pueden conectar de distintas maneras, de forma, que el sistema adopte diferentes configuraciones según la tarea que se quiera llevar a cabo, la cual realizará el sistema a través de un software desarrollado en Python.

1.4. Tecnología

En este apartado, se presentan y describen las diversas tecnologías empleadas en el desarrollo de este proyecto. El objetivo es ofrecer una visión general de los equipos, dispositivos, lenguajes de programación y plataformas que han sido utilizados para la implementación y puesta en marcha de la solución propuesta. A través de esta exposición, se pretende proporcionar al lector una visión completa de las tecnologías involucradas, sus características principales y su relevancia en el contexto del proyecto. Se explorarán aspectos como la elección de tecnologías específicas, su aplicabilidad, ventajas y desafíos asociados, con el fin de comprender en su totalidad el ecosistema tecnológico que ha dado forma al proyecto final.

1.4.1. Fuente de alimentación programable

Se trata de la fuente de alimentación de corriente continua (CC) totalmente programable SL600-10 de Magna-Power Electronics (figura 1.3) con las siguiente especificaciones generales (ver más en la ficha técnica: punto 13.1).

- Potencia: 6 kW
- Tensión máxima: 600 V

- Intensidad máxima: 10 A

Esta fuente cuenta con configuraciones programables de voltaje, corriente y protección junto con mediciones de alta precisión. Las funciones y características de la serie SL son accesibles y configurables desde una variedad de métodos de control, que incluyen:

- Interfaz del panel frontal con control manual
- E/S de usuario analógico-digital externo de 37 pines
- Interfaz de computadora RS232 con software y controladores (RIS panel)
- API de programación remota SCPI
- Mediciones de alta precisión
- Funcionalidad maestro-esclavo
- Sensores remotos
- Ethernet y GPIB disponibles
- Entradas analógicas externas de 0-10 V
- Límites de protección programables
- Respuesta transitoria rápida
- Software de interfaz remota
- Controlador NI LabVIEW™ e IVI
- Entrada de apagado de enclavamiento

Hay varias interfaces de programación adicionales disponibles, como LXI TCP/IP Ethernet (+LXI), IEEE-488 GPIB (+GPIB), accesorio Edgeport USB (+USB), accesorio RS485 (+RS485).

Figura 1.3: Fuente de alimentación de CC SL600-10 de Magna-Power Electronics

A continuación se introducen las características de interés para este proyecto de la gran variedad que ofrece la máquina.

- **Protección de salida programable**: el disparo por sobrevoltaje (OVT) y el disparo por sobrecorriente (OCT) programables permiten al usuario programar disparos cuando se excede el umbral. Los ajustes de OVT y OCT se pueden programar del 10 % al 110 % de los valores máximos de la unidad.
- **Programación y medición de alta precisión**: la precisión de programación de ± 0,075 % del voltaje nominal máximo garantiza que la salida siga el set point programado deseado. La precisión de lectura de ± 0,2 % de la tensión nominal máxima garantiza mediciones de alta precisión.
- **Interfaces de medición simultánea**: las mediciones de voltaje y corriente están disponibles simultáneamente desde los medidores del panel frontal, la salida analógica externa de 0-10 V y por comando de computadora.

- **API de programación remota SCPI**: compatible con los comandos estándar para instrumentos programables (SCPI), lo que permite que los comandos de texto ASCII sin procesar controlen todas las características, funciones y configuraciones del dispositivo.

La fuente de alimentación puede funcionar con una regulación de salida de voltaje constante (CV) o de corriente constante (CC) y cambia automáticamente entre los dos estados de regulación en función de los set point programados y la impedancia de carga. Es decir, la fuente de alimentación realiza un cruce automático continuo desde el control de modo de voltaje hasta el control de modo de corriente, según lo determinen los set point de voltaje y corriente y/o la impedancia de carga. Si cualquiera de los set points se establece en cero, el otro control de set point tendrá poco o ningún efecto, lo que forzará un límite de voltaje en cero o un límite de corriente en cero.

- **Tensión constante**: Cuando se indica el estado de regulación de voltaje constante, la fuente de alimentación mantiene un voltaje fijo en el set point programado, mientras que la corriente de salida fluctúa en función de la impedancia de carga, como se ilustra en la figura 1.4.

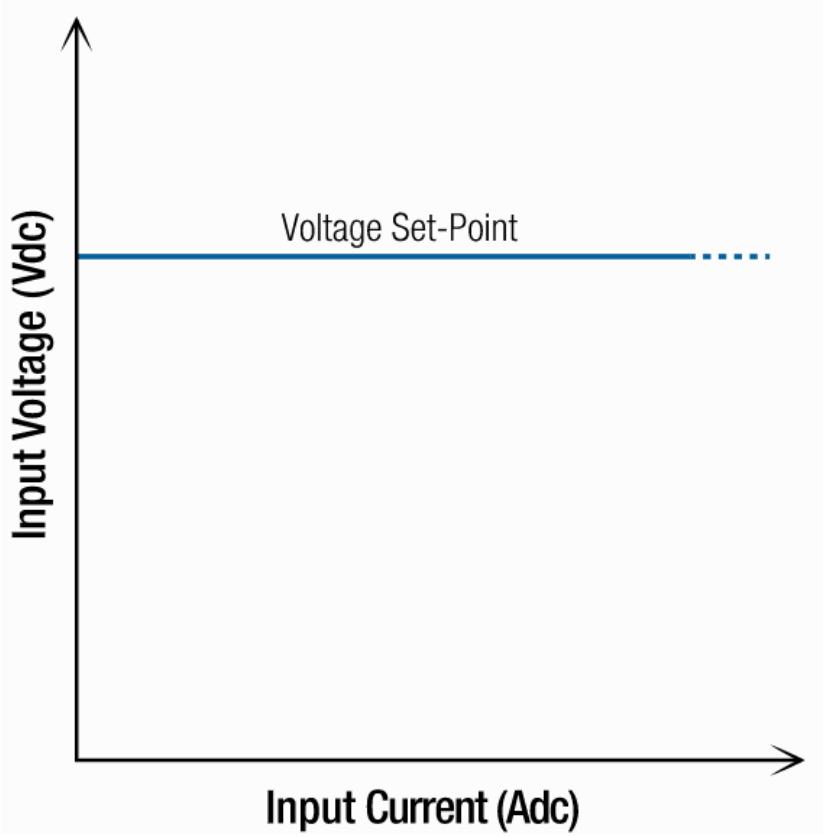


Figura 1.4: Modo de funcionamiento en tensión constante

- **Corriente constante**: Cuando se indica el estado de regulación de corriente constante, la fuente de alimentación mantiene una corriente fija en el set point programado, mientras que el voltaje de salida fluctúa en función de la impedancia de carga, como se ilustra en la figura 1.5.

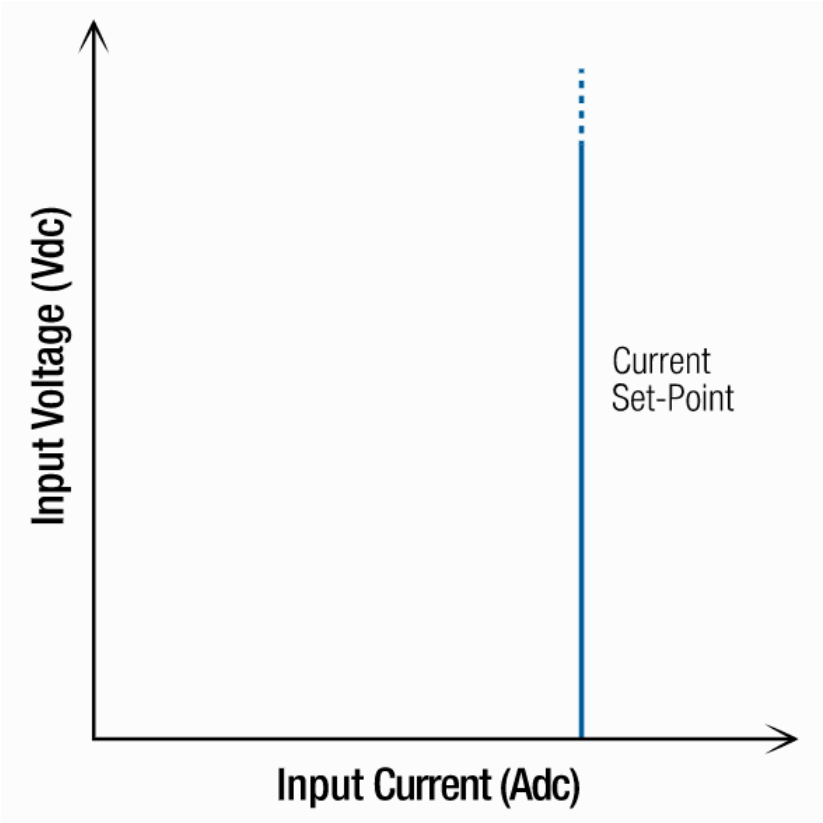


Figura 1.5: Modo de funcionamiento en corriente constante

1.4.2. Carga programable

Se trata de la carga programable de CC y CA 3A300-04 de APS (Adaptive Power Systems) (figura 1.6) de los que se disponen 3 módulos integrados en la 34M04 Mainframe (unidad central) de APS (figura 1.7) con las siguientes especificaciones (ver más en la ficha técnica: punto 13.2).

- Rango de potencia por módulo: [0, 300] VA
- Rango de tensión: [30, 300] V
- Rango de intensidad: [0, 4] A

La carga está diseñada para probar y evaluar fuentes de alimentación y baterías de CA o CC. Se puede operar desde el panel frontal (modo manual) o usando un control remoto RS232 o GPIB. Los 3 módulos de carga están instalados en un mainframe de cuatro ranuras 34M04 para que funcionen. El mainframe proporciona suministros de polarización y refrigeración para los módulos de carga y posee las siguientes características principales.

- Módulo de carga electrónico de CA y CC completamente programable con configuración flexible y capacidades de doble rango
- Control totalmente remoto de todos los ajustes de carga y lectura de medición mediante API de programación remota SCPI
- Medidores duales de tensión y corriente de alta precisión y alta resolución

Figura 1.6: Módulo de carga programable 3A300-04 de APS

Figura 1.7: Mainframe de carga programable 3A300-04 de APS

- Compatibilidad con un rango de frecuencia de CC o de 0.1 a 400 Hz (modos CC y LIN)
- Control de factor de potencia (PF) y factor de cresta (CF) (modos CC y LIN)
- Cambio del interruptor ON/OFF de la carga y encendido de la fuente de alimentación
- Detección de voltaje interno o externo
- Prueba automática Go/NoGo
- Protección total contra exceso de potencia, exceso de temperatura, exceso de voltaje y polaridad inversa
- Salida analógica de monitor de corriente (I-Monitor)
- Entrada de sincronización externa
- Control de ventilador de velocidad variable para un funcionamiento silencioso

A continuación se introducen las características de interés para este proyecto de la gran variedad que ofrece la máquina.

- **Protección de salida programable**: el disparo por sobrevoltaje (OV) y el disparo por sobrecorriente (OC) programables permiten al usuario programar disparos cuando se excede el umbral. Los ajustes de OV y OC se pueden programar hasta 315 V y 4.2 A, respectivamente.
- **Programación y medición de alta precisión**: resolución de 1 mA en el modo de CC en la programación de la corriente para garantizar que la salida siga el set point programado deseado. Resolución de 0.1 V y de 0.001 A en la lectura de tensión y corriente, respectivamente, que garantiza mediciones de alta precisión.
- **Interfaces de medición simultánea**: las mediciones de voltaje y corriente están disponibles simultáneamente desde los medidores del panel frontal y por comando de computadora.
- **API de programación remota SCPI**: compatible con los comandos estándar para instrumentos programables (SCPI), lo que permite que los comandos de texto ASCII sin procesar controlen todas las características, funciones y configuraciones del dispositivo.

La carga programable posee tres modos de funcionamiento distintos:

- **Modo de corriente constante lineal (LIN)** Cuando se opera en el modo de corriente constante lineal, el nivel de corriente que absorbe la carga constante depende del voltaje, es decir, la corriente de la carga seguirá la forma de onda de la tensión de entrada en tiempo real, tal y como se observa en la figura 1.8.

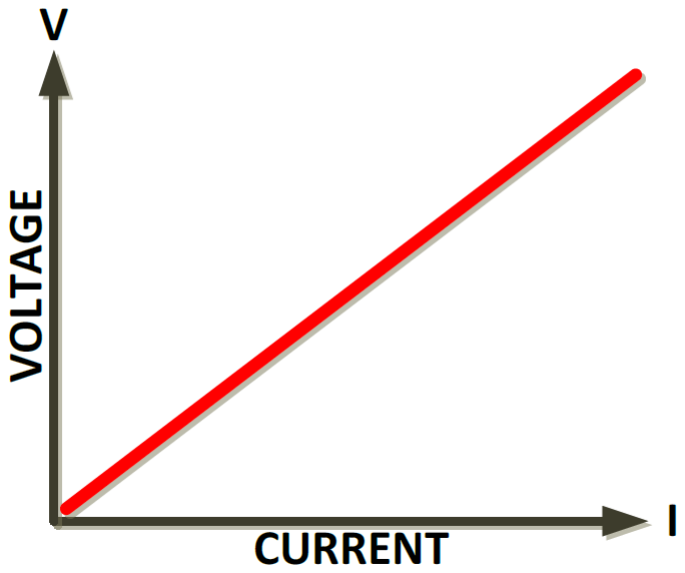


Figura 1.8: Modo de funcionamiento de corriente constante lineal

- **Modo de resistencia constante (CR)** En el modo de resistencia constante, la carga absorberá una corriente directamente proporcional al voltaje de entrada de CC detectado. La relación entre el voltaje de CC y la corriente es lineal por ley de Ohm y el usuario puede configurarla dentro del rango operativo de la carga de CA y CC. La corriente se define mediante $I = V/R$, donde R es el valor establecido en el modo CR y V es el voltaje de entrada de CC de la unidad bajo prueba.
- **Modo de corriente constante (CC)** El modo CC es el único modo de funcionamiento de interés para este proyecto. En este modo de operación, la carga absorbe un nivel constante de corriente según lo establecido por el usuario, independientemente de las variaciones de voltaje. Un circuito de retroalimentación en tiempo real asegura una corriente estable bajo cualquier variación de voltaje del suministro de CC o de la batería (ver figura 1.9).

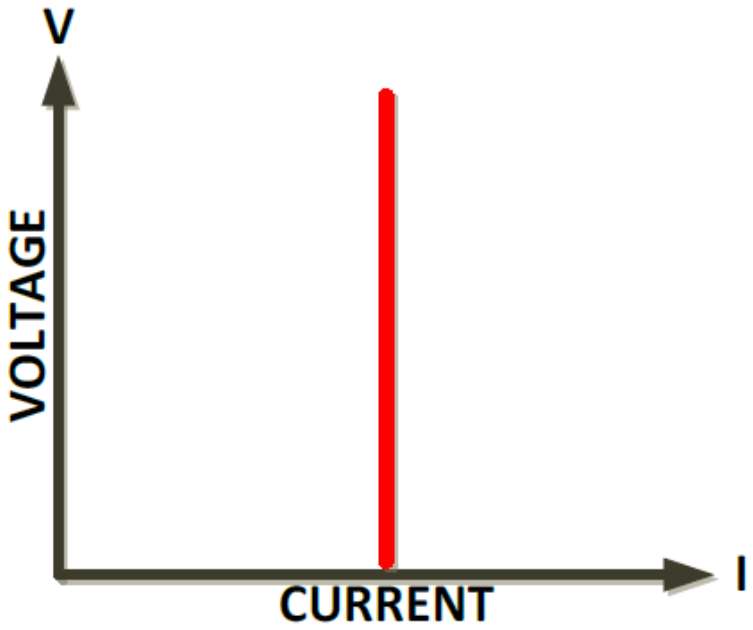


Figura 1.9: Modo de funcionamiento en corriente constante

1.4.3. Batería

En este proyecto se han usado dos tipos de baterías diferentes para demostrar que la herramienta es útil para cualquier clase de acumulador.

Por un lado, se ha usado la batería de las motos eléctricas del laboratorio del departamento (figura 1.10), en concreto, se trata de la Next NX1, que utiliza una batería de la cual no se especifica por que tipo de celdas está compuesta. Sin embargo, se realizado la hipótesis de que contiene celdas de tipo 18650 del fabricante Samsung (figura 1.11) por las especificaciones de la batería:

- Tensión nominal: 60 V
- Capacidad: 20 Ah

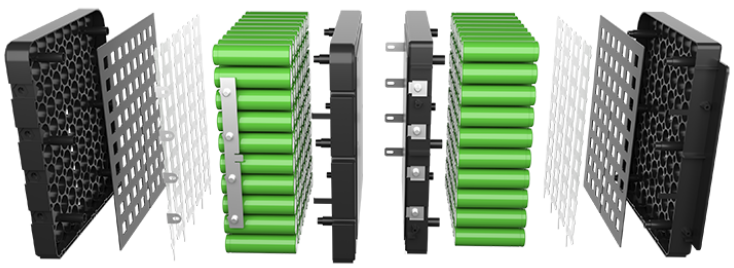


Figura 1.10: Batería de la moto Next NX1

Figura 1.11: Celdas Samsung tipo 18650

Desgraciadamente, no se tiene más información técnica sobre las características de la batería, pero como se verá en el punto 4.1, se ha podido estimar, a partir de una serie de pruebas haciendo uso de la misma plataforma y tomando la anterior hipótesis, el modelo de las celdas, el número total y como están conectadas entre sí.

Por otro lado, se ha usado una celda Panasonic PA-L154.K01 (figura 1.12) con las siguientes características entre otras que se especifican en la ficha técnica (ver punto 13.4).

- Tensión nominal: 3.6 V
- Capacidad: 2250 mAh

Figura 1.12: Celda Panasonic PA-L154.K01

1.4.4. Sistema de medida de tensión

El sistema de medida de tensión está formado por los siguientes componentes: microcontrolador, convertidor analógico-digital, divisor de tensión y regulador de tensión.

- **Microcontrolador**: Se trata de una tarjeta Arduino UNO R3 (figura 1.13), la cual posee las siguientes características técnicas (ver más en ficha técnica: punto 13.6).
  - Microcontrolador: Microchip ATmega328P6
  - Voltaje de funcionamiento: 5 voltios
  - Voltaje de alimentación: 7 a 20 voltios
  - Pines de E/S digitales: 14 (de los cuales 6 proporcionan salida PWM)
  - Pines de entrada analógica: 6
  - Corriente DC por Pin de E/S: 20 mA
  - Corriente CC para Pin de 3.3V: 50 mA
  - Memoria Flash: 32 KB de los cuales 0.5 KB utilizados por el gestor de arranque
  - SRAM: 2 KB
  - EEPROM: 1 KB
  - Velocidad del reloj: 16 MHz

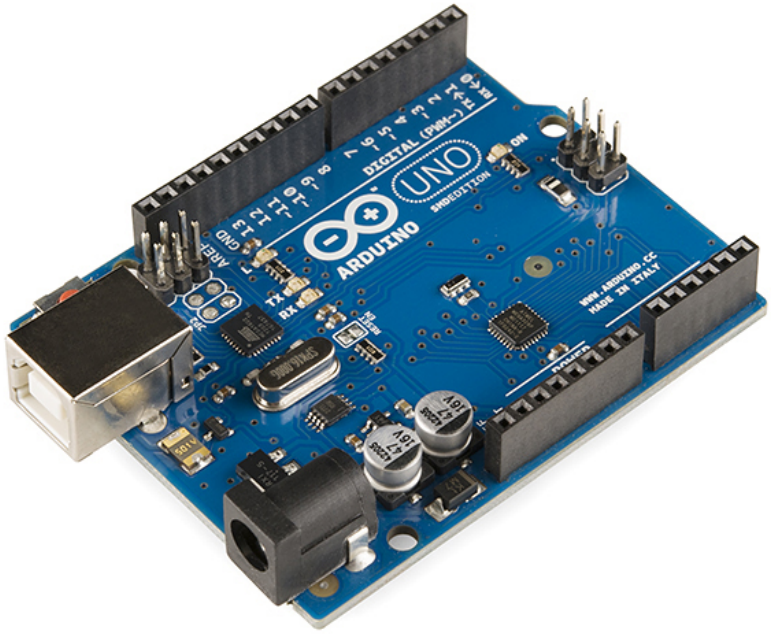


Figura 1.13: Placa Arduino UNO

- **Convertidor Analógico Digital (ADC)**: Se trata del módulo ADC de 16 bits DFRobot I2C ADS1115 (ver figura 1.14) que puede adquirir y convertir con precisión señales analógicas en digitales. Este módulo utiliza un chip ADC ADS1115 de 16 bits y admite una tensión de alimentación de 3,3 - 5V. El chip

tiene un voltaje de referencia de precisión y un ajuste de ganancia programable (PGA), de modo que, se pueden realizar mediciones y conversiones muy precisas para señales débiles y altamente variables. Además, también es aplicable a todo tipo de aplicaciones que la placa de control principal necesita para recopilar con precisión las señales analógicas.

El ADC puede leer señales analógicas de hasta 4 canales. Sin embargo, debido al interruptor de selección de dirección I2C incorporado, admite la cascada de dos módulos ADC y puede leer señales analógicas de hasta 8 canales.

A continuación, se detallan las principales características del módulo ADC (ver más en la ficha técnica: punto 13.5).

- Admite un voltaje de suministro amplio de 3,3 - 5,0 V
- Señal de nivel digital I2C de 3,3 V
- Interfaz Gravity I2C, plug and play
- Interruptor de selección de dirección I2C integrado, admite cascada de dos módulos ADC
- Headers de 3 pines codificados por colores, plug and play con sensores analógicos Gravity
- Voltaje de suministro (VCC): 3,3 - 5,0 V
- Rango de detección de señal analógica: 0 VCC
- Número de canales analógicos: 4
- Bits ADC: 16 bits
- Corriente de funcionamiento: 2 - 3 mA (no incluye módulo de sensor)
- Nivel de interfaz: alto 3,3 V/ bajo 0 V

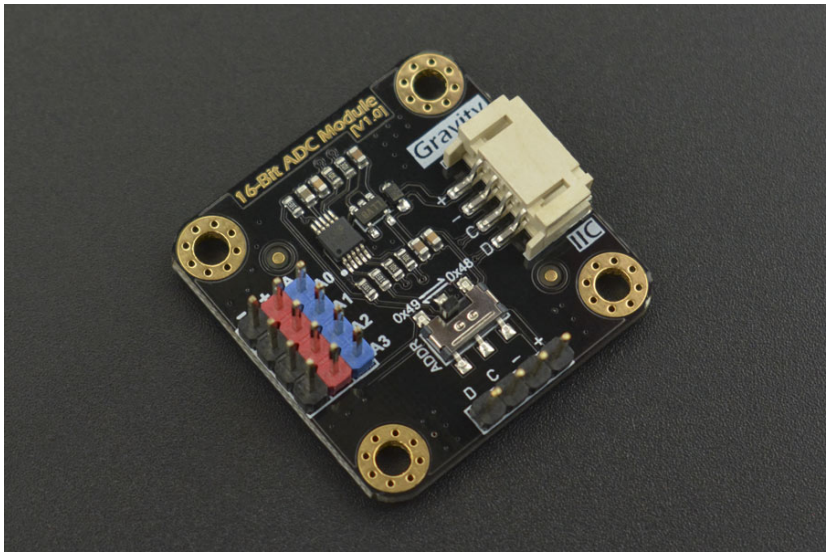


Figura 1.14: Convertidor Analógico Digital ADS1115

- **Divisor de tensión**: Se trata de un divisor resistivo formado por dos resistencias como se puede observar en la figura 1.15, las cuales variarán en función de la batería que con la que se quiera trabajar en la herramienta.

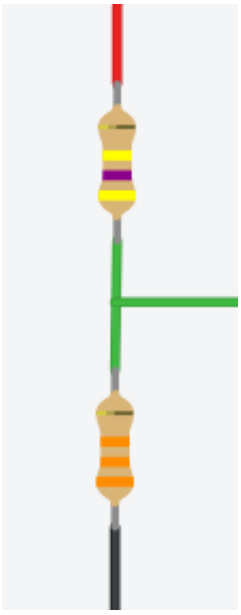


Figura 1.15: Divisor resistivo

- **Regulador de tensión**: Se trata de un regulador zener básico, formado un diodo zener y una resistencia como se puede observar en la figura 1.16, que variarán en función de la batería que con la que se quiera trabajar en la herramienta.

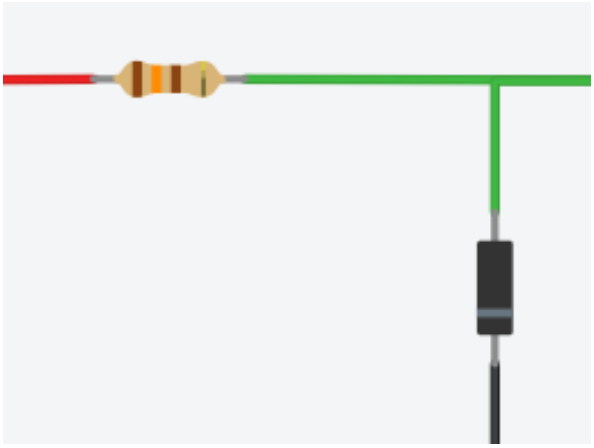


Figura 1.16: Regulador zener

1.4.5. Computadora

Se ha utilizado un ordenador de escritorio convencional con un sistema operativo Windows 7, el cual funciona como unidad central del proyecto, donde se ejecuta el software permitiendo la comunicación entre el resto de elementos del sistema.

1.4.6. Programación

El software de la herramienta se ha desarrollado principalmente en Python 3, concretamente con la versión 3.8. Python es un lenguaje de programación de alto nivel interpretado, orientado a objetos y libre. Además, es un lenguaje que se caracteriza por la legibilidad y simplicidad de su código; y el enorme número de librerías disponibles hacen que sea una de las mejores opciones para programar aplicaciones que manejen grandes cantidades de datos y se comuniquen con diversos dispositivos al mismo tiempo, como es el caso de este proyecto. También se ha usado C, para la programación de la placa Arduino en el desarrollo del sistema de medida.

Figura 1.17: Logo Python

Figura 1.18: Logo C

Por otro lado, se ha usado Visual Studio Code como entorno de desarrollo para todo el software a excepción de la parte correspondiente al Arduino, para la cual se ha usado el propio entorno de desarrollo integrado de Arduino (Arduino IDE). Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto y compatible con multitud de lenguajes de programación entre los cuales se encuentra Python.

1.5. Alcance del proyecto

La herramienta desarrollada dentro de este proyecto es capaz de realizar todas las funcionalidades con total seguridad, previniendo siempre posibles fallos que puedan producir algún efecto peligroso. El sistema adapta y restringe conforme a sus limitaciones las capacidades de todas las funcionalidades e informa convenientemente al usuario de las configuraciones que se pueden llevar a cabo y las restricciones del sistema.

El sistema maneja varios tipos de comunicación con los distintos dispositivos que integra con un control de errores que impide que si una de estas falla el sistema quede bloqueado o se caiga.

Además, es una plataforma dinámica y intuitiva para el usuario. Es decir, existe una interfaz entre el usuario y la herramienta que permite utilizarla sin necesidad de tener

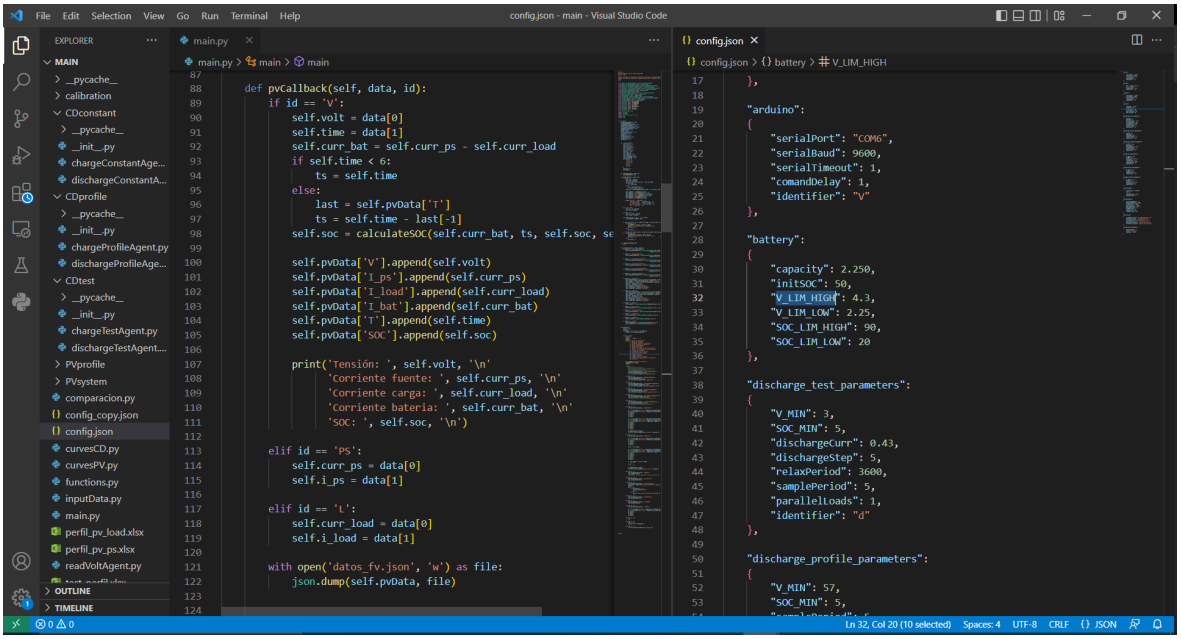


Figura 1.19: Entorno de desarrollo VS Code

Figura 1.20: Arduino IDE

conocimiento de su diseño interno. De modo que, respecto a la entrada de datos, el usuario solo tiene que seguir las instrucciones de la plataforma para realizar una tarea y respecto a la salida de datos, se pueden visualizar fácilmente todos los resultados quedando estos guardados de forma automática en una misma ubicación.

La entrada de datos implementa un control de errores que imposibilita que el usuario haga caer el sistema por equivocación o de forma deliberada.

1.6. Metodología y planificación

Para la realización de este proyecto, se ha empezado estudiando los manuales y fichas técnicas de todos los dispositivos involucrados.

Una vez se ha tenido claro el funcionamiento y las limitaciones de cada equipo, se ha procedido a realizar una serie de pruebas en cada uno de ellos individualmente, comprobando, de esta forma, su correcto funcionamiento y analizando los problemas que han surgido para poder evitarlos más adelante.

A continuación, se han ido implementando los diferentes sistemas a desarrollar, tanto a nivel físico como de software. Y posteriormente, se ha comprobado su funcionamiento realizando una serie de ensayos y pruebas, los cuales han servido para detectar los errores de la herramienta y poder solucionarlos.

Finalmente, se han sacado una serie de conclusiones, tanto sobre el funcionamiento de la propia herramienta como de los mismos experimentos y pruebas.

2. Normas y referencias

En este apartado se detallan las normas relativas al proyecto, la herramientas software empleadas para cada una de las partes del mismo y las referencias de las que se ha obtenido la información bibliográfica necesaria para la consecución de los objetivos establecidos.

2.1. Disposiciones legales y normas aplicadas

2.1.1. Elaboración del documento

- **Norma UNE 157001:2014**: Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico [UNE, 2014].

2.1.2. Equipos y electrónica en general

- **Directiva 2011/65/UE**: La Directiva 2011/65/UE, de las siglas en inglés de "Restriction of hazardous substances", es una directiva que rige el uso de materiales peligrosos en la fabricación de equipos eléctricos y electrónicos. A esta directiva se refiere como RoHS [Sierra Molina, 2006].
- **REBT**: El Reglamento Electrotécnico para Baja Tensión (REBT) es un reglamento de obligado cumplimiento que prescribe las condiciones de montaje, explotación y mantenimiento de instalaciones de baja tensión.

2.2. Entornos de desarrollo y software empleado

- **Visual Studio Code**: Entorno de desarrollo empleado para la programación del software en Python.
- **Arduino IDE**: Entorno de desarrollo propio de Arduino empleado para la programación del Arduino.
- **Python 3.8**: Lenguaje de programación Python empleado para el desarrollo del software de la herramienta
- **C**: Lenguaje de programación empleado para la programación del software del sistema de medida de tensión dentro del Arduino.

2.3. Bibliografía

[1] Norma UNE 157001:2014. *Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico.* AENOR.

[2] Directiva 2011/65/UE del Parlamento Europeo y del Consejo de 8 de junio de 2011. *Restricciones a la utilización de determinadas sustancias peligrosas en aparatos eléctricos y electrónicos*. En «*DOUE*» *núm. 174, de 1 de julio de 2011* (páginas 88 a 110).

[3] Mª Teresa Sierra Molina (2006). *La Directiva RoHS y sus implicaciones en la industria eléctrica/electrónica*. En *Universidad Carlos III de Madrid*.

[4] REBT (2023). *Reglamento electrotécnico para baja tensión e ITC*. Agencia Estatal Boletín Oficial del Estado.

[5] National Geographic (2022). *España produce el triple de energía solar que hace 3 años*. Recuperado de https://www.nationalgeographic.com.es/ciencia/espana-produce-triple-energia-solar-que-hace-3-anos_18737.

[6] Wikipedia (2023). *Batería (electricidad)*. Recuperado de https://es.wikipedia.org/wiki/-Bater%C3%ADa_(electricidad)#Principios_de_funcionamiento.

[7] Magna-Power Electronics (2012). *SL Series 1 User Manual*.

[8] Adaptive Power Systems (2014). *34M04 Load Mainframe Operation Manual*.

[9] Adaptive Power Systems (2014). *3A Series AC & DC Load Operation Manual*.

[10] DFRobot. *DFR0553 Gravity I2C ADS1115 16-Bit ADC Module Arduino & Raspberry Pi Compatible*. Recuperado de https://wiki.dfrobot.com/Gravity__I2C_ADS1115_16-Bit_ADC_Module_Arduino_%26_Raspberry_Pi_Compatible__SKU__DFR0553.

[11] @programarfacilc (2018). *ADS1115 convertidor analógico digital ADC para Arduino y ESP8266*. Recuperado de https://programarfacil.com/blog/arduino-blog/ads1115-convertidor-analogico-digital-adc-arduino-esp8266.

[12] Texas Instruments (2016). *ADS111x Ultra-Small, Low-Power, I2C-Compatible, 860-SPS, 16-Bit ADCs With Internal Reference, Oscillator, and Programmable Comparator*.

[13] Next Electric Motors. *Batería moto eléctrica: Funcionamiento*. Recuperado de https://nextelectricmotors.com/bateria-moto-electrica/.

[14] Samsung SDI Co., Ltd., Energy Business Division (2014). *SPECIFICATION OF PRODUCT: Lithium-ion rechargeable cell for power tools. Model name : INR18650-25R*. SAMSUNG SDI Confidential Proprietary.

[15] Panasonic. *Ficha técnica Panasonic PA-L154.K01*.

[16] Panasonic. *Handling Instructions Li-Ion Battery Soft Pack (NOT a hard case battery pack) Panasonic PA-L154.K01*.

[17] Wikipedia (2023). *Arduino Uno*. Recuperado de https://es.wikipedia.org/wiki/-Arduino_Uno.

[18] Arduino (2023). *Arduino UNO R3 Product Reference Manual SKU: A000066*. Recuperado de https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf.

3. Diseño del sistema

3.1. Alternativas

Dentro de las posibilidades del proyecto se ha realizado un análisis exhaustivo de las diferentes alternativas disponibles para la solución del objetivo planteado. Por tanto, se ha requerido evaluar las opciones para el desarrollo del software y el sistema de medida. A continuación, se presenta un análisis detallado de las alternativas propuestas, destacando las ventajas y desventajas de cada una de ellas, justificando cual de ellas ha sido escogida en casa caso.

Alternativas para el desarrollo del software:

1. **Python**, como ya se ha comentado en el punto 1.4.6, es un lenguaje de programación de alto nivel ampliamente utilizado en el ámbito de la ingeniería y la ciencia de datos. Presenta una sintaxis clara y legible, lo que facilita su aprendizaje y desarrollo de software. Algunas ventajas de utilizar Python son:
   - **Facilidad de uso y aprendizaje**: Python se caracteriza por su legibilidad y simplicidad, lo que permite a los desarrolladores escribir código de manera eficiente y comprensible. Esto, también, facilita la comprensión del código para las personas que no han estado involucradas en su desarrollo. Además, cuenta con una amplia documentación y una gran comunidad de usuarios que brindan soporte.
   - **Amplia disponibilidad de librerías**: Python ofrece una gran cantidad de librerías especializadas que facilitan el desarrollo de diferentes aplicaciones. En el ámbito de la ingeniería, existen librerías como NumPy, SciPy y Pandas, entre muchas otras, que proporcionan herramientas para el análisis numérico, la manipulación de datos y la implementación de algoritmos complejos.
   - **Versatilidad y flexibilidad**: Python es un lenguaje versátil que puede utilizarse en diferentes entornos, desde el desarrollo de aplicaciones de escritorio hasta el desarrollo web. Esto brinda flexibilidad y permite adaptarse a las necesidades específicas del proyecto. Python también es conocido por su capacidad de integración con otros lenguajes y tecnologías, es decir, si en algún momento se requiere la interacción con componentes desarrollados en otros lenguajes o con otros dispositivos (como es el caso), Python ofrece facilidades para lograr esa integración, lo cual es una ventaja adicional en términos de flexibilidad y expansión futura del proyecto.
   - **Open source**: Python es una tecnología de uso libre, lo que presenta una gran ventaja frente a otras tecnologías como Matlab, por la que hay que pagar una licencia de uso.

   No obstante, es importante tener en cuenta que Python puede ser menos eficiente en términos de velocidad de ejecución en comparación con otros lenguajes de programación como C++. Sin embargo, en este caso, la diferencia de rendimiento

no es significativa y se compensa con la productividad y facilidad de desarrollo que ofrece Python.

2. **C++** es un lenguaje de programación de propósito general ampliamente utilizado en aplicaciones que requieren un alto rendimiento y eficiencia. Algunas ventajas de utilizar C++ son:
   - **Alta velocidad y eficiencia**: C++ es conocido por su rendimiento rápido y eficiente, lo que lo convierte en una opción adecuada para aplicaciones que requieren un procesamiento intensivo y un control de recursos preciso.
   - **Acceso a la programación de bajo nivel**: C++ permite acceder a características de bajo nivel, como la gestión de memoria, lo que brinda un mayor control sobre los recursos del sistema y facilita la optimización de algoritmos.
   - **Amplia comunidad y bibliotecas**: C++ cuenta con una gran comunidad de desarrolladores y una amplia selección de bibliotecas y frameworks disponibles, lo que facilita la implementación de funcionalidades complejas y la reutilización de código.

   Sin embargo, el desarrollo en C++ puede ser más complejo y propenso a errores en comparación con Python, ya que requiere una mayor atención a los detalles y un mayor nivel de experiencia en programación. Además, el tiempo de desarrollo suele ser más largo debido a la necesidad de escribir y depurar código de manera más minuciosa.

3. **Matlab** es un entorno de programación especialmente diseñado para el cálculo científico y técnico. Es ampliamente utilizado en campos como la ingeniería, las ciencias exactas y la investigación. Algunas ventajas de utilizar Matlab son:
   - **Facilidad en el manejo de datos y cálculos matemáticos**: Matlab proporciona una sintaxis específica y funciones especializadas que simplifican el manejo de datos, la implementación de algoritmos matemáticos y el análisis numérico.
   - **Amplia colección de herramientas y librerías**: Matlab ofrece una amplia gama de herramientas y librerías específicas para diferentes áreas de aplicación, lo que facilita el desarrollo de proyectos técnicos y científicos. Además, cuenta con una comunidad activa que comparte soluciones y recursos.
   - **Visualización y gráficos**: Matlab es reconocido por su capacidad para generar gráficos y visualizaciones de datos de manera sencilla y efectiva. Esto es especialmente útil en aplicaciones donde la representación visual de los resultados es importante.

   No obstante, es importante considerar que Matlab es un entorno propietario y su licencia puede ser costosa en comparación con otras alternativas. Además, su enfoque se centra principalmente en cálculos científicos y técnicos, por lo que puede resultar menos adecuado para aplicaciones más generales o de mayor escala, como podría ser el caso de este proyecto.

Después de evaluar las diferentes alternativas para el desarrollo del software, se concluye que Python es la opción más adecuada. Python proporciona una com-

binación de facilidad de uso, amplia disponibilidad de librerías especializadas y versatilidad que se ajusta a los requerimientos del proyecto. Su legibilidad y simplicidad permiten un desarrollo eficiente y comprensible, y la amplia comunidad de usuarios brinda soporte adicional, además, de que es de uso libre.

Alternativas para el sistema de medida:

1. **Arduino**. En primer lugar, entre la gran gama de microcontroladores disponibles en el mercado se ha escogido el Arduino UNO R3, ya que es el que el departamento dispone en mayor cantidad, y siempre tiene unidades disponibles. Además, Arduino es una plataforma de hardware de código abierto ampliamente utilizada en proyectos de electrónica y sistemas embebidos que proporciona una forma accesible y muy económica de realizar mediciones y controlar dispositivos electrónicos. Algunas ventajas de utilizar Arduino son:
   - **Facilidad de uso**: Arduino se caracteriza por su simplicidad y facilidad de uso, lo que permite realizar proyectos como el que se trata de manera muy sencilla y rápida.
   - **Amplia disponibilidad de sensores y módulos**: Arduino cuenta con una amplia gama de sensores y módulos disponibles en el mercado, lo que facilita la conexión y el uso de diferentes tipos de dispositivos.
   - **Comunidad activa**: Arduino cuenta con una comunidad activa de usuarios que comparten proyectos, tutoriales y soluciones, lo que brinda soporte adicional y fomenta el aprendizaje colaborativo.

   Sin embargo, es importante tener en cuenta que Arduino tiene limitaciones en cuanto a su capacidad de procesamiento y almacenamiento de datos. También puede tener limitaciones en términos de precisión y resolución en comparación con soluciones más avanzadas, lo que en este caso, supone un problema importante, ya que es fundamental para el proyecto conocer en la tensión con una buena precisión para después obtener resultados válidos a la hora de obtener conclusiones.
2. **Arduino + ADC**. Para mejorar la precisión y la resolución en las mediciones, se puede utilizar un convertidor analógico-digital (ADC) en conjunto con Arduino. Un ADC permite convertir señales analógicas en digitales con mayor precisión. Algunas ventajas de esta combinación son:
   - **Mayor precisión en las mediciones**: Al utilizar un ADC externo, se puede mejorar la precisión y la resolución de las mediciones realizadas con Arduino. Esto es especialmente útil cuando se requiere una alta precisión en las mediciones o se trabaja con señales de entrada más sensibles.
   - **Flexibilidad y compatibilidad**: La combinación de Arduino con un ADC externo permite ampliar las capacidades de medición y adaptarse a diferentes requisitos. Existen varios modelos de ADC disponibles en el mercado, lo que brinda opciones para seleccionar el más adecuado según las necesidades específicas del proyecto.
   - **Integración con Arduino**: Arduino ofrece una integración sencilla con ADC externos a través de sus pines analógicos, lo que facilita la conexión y el uso conjunto de ambos componentes.

Por tanto, por todas estás razones queda justificada la utilización de Arduino en conjunto con un ADC externo en este proyecto. Sin embargo, ¿Porque se ha elegido el ADC DFRobot I2C ADS1115 y no otro?

En primer lugar, como se explicará en más detalle en los puntos 3.2.1 y 3.3, cumple con las características técnicas necesarias para el proyecto con una resolución de 15 bits. Es compatible con Arduino, el cual dispone de una librería especifica para el manejo de este ADC, lo que facilita enormemente la implementación y la integración del mismo. Además, tiene un modo llamado, modo diferencial (puntos 3.2.1 y 3.3), que resulta convenientemente útil para la medición de tensión en baterías, como es el caso de este proyecto. Finalmente, entre los ADC que cumplen con todas estás condiciones y que estaban disponibles en el mercado, este modelo era el más económico.

En conclusión, la elección de Python para el desarrollo del software y de Arduino junto con un ADC externo para el sistema de medida se basa en una combinación de varios factores como la facilidad de uso, la disponibilidad de recursos y soporte, la flexibilidad, la precisión de las mediciones, la capacidad de expansión y el coste económico. Estas opciones proporcionan un equilibrio adecuado entre eficiencia, simplicidad y funcionalidad, permitiendo el desarrollo exitoso del proyecto.

3.2. Análisis del sistema y requisitos de diseño

En este punto se pretende realizar un análisis completo del sistema. En primer lugar, se analizará las distintas formas de comunicación, y los requisitos y restricciones de cada uno de los dispositivos involucrados en la herramienta.

En segundo lugar, se analizará cada una de las funcionalidades del sistema y los diferentes casos de uso, profundizando en las posibilidades y limitaciones del mismo.

Finalmente, se procederá a realizar un análisis preliminar del software que se ha de diseñar para que la plataforma cumpla con todas las funcionalidades que se han definido, mediante un diagrama de Clases/Agentes.

3.2.1. Análisis del hardware

Fuente de alimentación programable

La fuente de alimentación se comunica con la computadora a través de una conexión Ethernet mediante una comunicación tipo socket, concretamente, mediante el protocolo TCP/IP haciendo uso de los comandos estándar para instrumentación programable o Standard Commands for Programmable Instrumentation (SCPI). Este protocolo de comunicación permite a la fuente realizar varias acciones, las cuales se detallan en el cuadro 3.1, que utiliza la siguiente sintaxis:

- <**NR1**> representa dígitos con un punto decimal implícito asumido a la derecha del dígito menos significativo.
- <**NR2**> representa dígitos con punto decimal explícito.

- **<NRf+>** es el formato decimal expandido que incluye <NR1>; <NR2>; dígitos con un punto decimal explícito y un exponente; y MIN y MAX, donde MIN y MAX son los valores límite mínimo y máximo en la especificación de rango para el parámetro.
- **<bool>** representa datos booleanos, pueden ser 0/1 o OFF/ON.

Comando SCPI	**Descripción**
MEAS:VOLT?	Mide y devuelve el voltaje medido
MEAS:CURR?	Mide y devuelve la corriente medida
VOLT <NRf+>	Establece el set point de voltaje
CURR <NRf+>	Establece el set point de la corriente
VOLT:PROT <NRf+>	Establece el set point de disparo por sobrevoltaje (OVT)
CURR:PROT <NRf+>	Establece el set point de disparo por sobrecorriente (OCT)
CONT:INT <bool>	Configura la capacidad de iniciar, detener, armar y borrar a través del panel frontal
CONT:EXT <bool>	Configura la capacidad de iniciar, detener, armar y borrar a través de entradas digitales y comandos por computadora
CONF:SETPT <NR1>	Establece el modo operativo de la fuente de alimentación (0: ROTARY, 1: KEYPAD, 2: EXT PGM, or 3: REMOTE)
OUTP:START	Permite que el circuito de procesamiento de energía del dispositivo comience a producir salida
OUTP:STOP	Deshabilita el circuito de procesamiento de energía del dispositivo para dejar de producir salida
CAL:PASS <NR1>	Protege contra la corrupción de la calibración del sistema e inicia la secuencia de calibración
CAL:POT <NR1>,<NR2>	Establece el valor para el potenciómetro especificado. La primera variable especifica el potenciómetro que se va a ajustar (del 1 al 5) y la segunda variable especifica la configuración del potenciómetro (del 0 al 255)
CAL:DEF	Establece la calibración de los potenciómetros a los valores predeterminados de fábrica
CAL:STOP	Termina el subsistema de calibración

Tabla 3.1: Lista de comandos SCPI de la fuente de alimentación

Para enviar una orden correctamente a la máquina se debe de enviar un string codificado en bytes formado por el comando correspondiente más el carácter '\n' que representa el fin del comando. De la misma forma, si se quiere leer el valor devuelto por la máquina como respuesta a un comando, esta se recibe como un string codificado en bytes formado por la respuesta más el carácter '\n' que representa el fin de la respuesta.

A continuación, se muestran dos ejemplos en Python para entender mejor la sintaxis que se emplea con la fuente de alimentación.

Establecer la corriente en 2.5 A:

```
powerSupply.sendall(('CURR␣' + str(2.5) + '\n').encode())
```

Leer la tensión:

```
powerSupply.sendall('MEAS:VOLT?\n'.encode())
resp = float(powerSupply.recv(16).decode().replace('\n', ''))
```

Además, se ha de tener en cuenta que la fuente de alimentación puede entregar una tensión de entre 0 y 600 V y una intensidad de entre 0 y 10 A.

También, se ha confirmado que la máquina funciona de forma óptima a tensiones mayores de 12 V, pero para tensiones menores de este valor no está bien calibrada, ya que los valores de tensión del display de la fuente o los de la respuesta por comandos SCPI (que son los mismos) no se corresponden con el valor de referencia introducido por comandos, ni con el valor real (medido con un voltímetro) al que está la fuente, los cuales sí coinciden. Por tanto, el valor real y el de referencia sí coinciden pero hay un desfase de 0.5 V por debajo, aproximadamente, con la medición que devuelve la máquina. Esto se ha intentado solucionar recalibrando la máquina pero no se ha conseguido (ver más en detalle en el punto 4.6).

Por otro lado, se ha comprobado que tiene que existir un tiempo mayor a 1 segundo entre el envió de dos comandos o entre el envió de un comando y una respuesta para que la comunicación con la máquina sea la correcta.

Finalmente, a continuación se especifica la configuración del socket de la comunicación TCP/IP de la fuente de alimentación.

- **Dirección IP**: 150.128.87.59.
- **Puerto**: 50505

Carga programable

Todos los módulos de la carga se comunican con la computadora a través del puerto serie RS232 ubicado en el mainframe, mediante el protocolo de comunicación serie y haciendo haciendo uso de los comandos estándar para instrumentación programable o Standard Commands for Programmable Instrumentation (SCPI), al igual que la fuente de alimentación. Este protocolo de comunicación permite configurar los ajustes de la carga de forma remota y recuperar los datos de medición para el análisis, tal y como se detalla en el cuadro 3.2, que utiliza la siguiente sintaxis.

- <**NR2**>: Valor numérico con punto decimal con el formato ###.#####. Por ejemplo: 30.12345. Es decir, la carga lee hasta cinco dígitos significativos después del punto decimal. El punto decimal se puede omitir si no es necesario.

Comando SCPI	Descripción
MEAS:VOLT?	Mide y devuelve el voltaje medido en V
MEAS:CURR?	Mide y devuelve la corriente medida en A
CURR:A <NR2>	Establece el set point de la corriente de carga en el set-point A en A
CURR:B <NR2>	Establece el set point de la corriente de carga en el set-point B en A
LIM:VOLT:LOW <NR2>	Establece el valor de disparo por subtensión. La operación fuera de este hará que se genere una señal NG (NO GOOD)
LIM:VOLT:HIGH <NR2>	Establece el valor de disparo por sobrevoltaje. La operación fuera de este hará que se genere una señal NG (NO GOOD)
LOCAL	Deshabilita el control remoto, cierra la interfaz de control RS232
REMOTE	Establece el control remoto de la unidad a través de RS232
MODE CC	Establece el modo de funcionamiento de la carga en corriente continua
LOAD ON	Enciende la carga
LOAD OFF	Apaga la carga
FREQ <NR2>	Establece el valor de la frecuencia
CHAN <NR2>	Selecciona un módulo de carga específico. El rango disponible es de 2 a 4
GLOB:MEAS:VOLT?	Consulta el voltaje total en todos los módulos de carga
GLOB:MEAS:CURR?	Consulta la corriente total en todos los módulos de carga
GLOB:LEV A	Establece el nivel seleccionado en todos los módulos de carga en A
GLOB:LEV B	Establece el nivel seleccionado en todos los módulos de carga en B
GLOB:LOAD ON	Enciende todos los módulos de carga
GLOB:LOAD OFF	Apaga todos los módulos de carga
GLOB:MODE CC	Establece el modo de operación en corriente continua en todos los módulos de carga

Tabla 3.2: Lista de comandos SCPI de la carga

Para enviar una orden correctamente a la máquina se debe de enviar un string codificado en bytes formado por el comando correspondiente más el carácter '\n' que representa el fin del comando. De la misma forma, si se quiere leer el valor devuelto por la máquina como respuesta a un comando, esta se recibe como un string codificado en bytes formado por la respuesta más el carácter '\n' que representa el fin de la respuesta.

A continuación, se muestran dos ejemplos en Python para entender mejor la sintaxis que se emplea con la CARGA.

Establecer la corriente en 2.5 A:

```
load.write(('CURR:A␣' + str(2.5) + '\n').encode())
```

Leer la tensión:

```
load.write('MEAS:VOLT?\n'.encode())
resp = float(load.read_until().decode().replace('\n', ''))
```

Además, se ha de tener en cuenta que cada módulo de carga puede soportar tensiones de hasta 300 V e intensidades entre 0 y 4 A.

Todos los comandos de control remoto son generalmente específicos del canal. Eso significa que se aplican a un módulo de carga individual. El módulo de carga seleccionado se configura usando el comando "CHAN". Una vez seleccionado, el módulo permanece seleccionado y todos los comandos emitidos se aplican a ese módulo hasta que se emite un comando CHAN para seleccionar módulos diferentes. Los módulos están numerados del 2 al 4 de izquierda a derecha. El único comando no específico del canal es el comando "GLOB" que se aplica a todos los módulos instalados.

De esta forma, se pueden conectar los tres módulos disponibles en paralelo (figura 3.1) para obtener una carga más potente que podría soportar hasta 300 V y 12 A, usando los comandos "GLOB", para controlar todos los módulos al mismo tiempo.

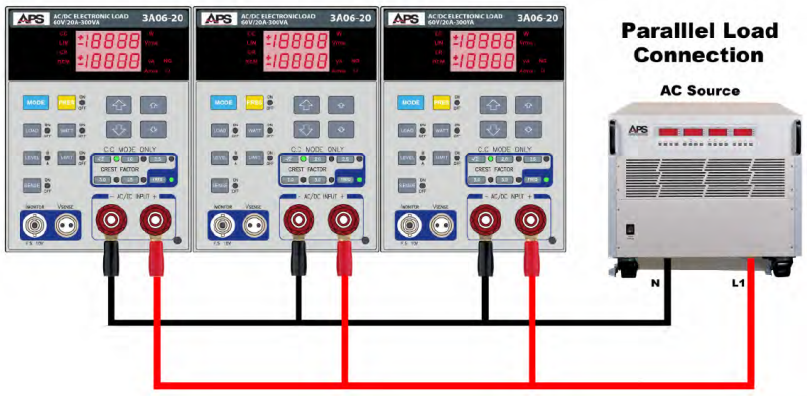


Figura 3.1: Celda ion-litio

Sin embargo, se ha confirmado a través de múltiples pruebas iniciales realizadas en cada uno de los módulos de la carga que solo el canal correspondiente al primer módulo empezando por la izquierda (canal 2) funciona correctamente. En el caso del canal 3, la comunicación no funciona debidamente por motivos que se desconocen y no se han podido averiguar. En el caso del canal 4, el módulo está inhabilitado y no responde ante ninguna acción tanto manual como remota debido a la activación del bit 2 del registro de protección del módulo por sobrevoltaje, OVP (Over Voltage Protection). Se ha intentado poner este bit a 0 para restablecer el normal funcionamiento del módulo sin éxito.

Finalmente, la interfaz RS232 del mainframe APS 34M04 está configurada de la siguiente manera:

- **Tasa de baudios**: 9600 bps
- **Paridad**: Ninguna

- **Bit de datos**: 8 bits
- **Bit de parada**: 1 bit
- **Retraso de comando**: se requieren 20 ms entre comandos sucesivos para permitir el análisis y procesamiento de comandos

Batería

Para el caso de la batería de la moto se ha realizado una primera estimación de como estaría esta formada a partir del tipo de celda que utiliza (celdas 18650 de Samsung) y de las pocas especificaciones técnicas que se tienen de la batería (60 V y 20 Ah).

De esta manera, se ha estimado que la batería esta compuesta por 17 módulos conectados en serie de 8 celdas de ion-litio INR18650-25R de Samsung conectadas en paralelo. Estas celdas son de 3.6 V y 2.5 Ah por lo que se cumpliría lo siguiente.

$$V_{nominal} = 3.6V \cdot 17 = 61.2V \tag{3.1}$$

$$Capacidad = 2.5Ah \cdot 8 = 20Ah \tag{3.2}$$

El resto de características técnicas de estas celdas se encuentran en su ficha técnica. De forma que, como se puede ver a partir del punto 4, se han realizado una serie de pruebas con la propia herramienta para comprobar que, realmente, se trata de este tipo de celdas y de esta configuración.

Según la ficha técnica (ver punto 13.3) estas celdas pueden descargarse hasta un límite de 2.5 V, pero se ha preferido ser conservador y se ha decidido fijar el límite de descarga en una tensión de 3.35 V. Además, se ha comprobado que la tensión cuando la batería está cargada al 100 % del SOC es de 71 V. Por lo que se ha fijado los siguientes límites.

- **Tensión límite máxima**: 71.0 V
- **Tensión límite mínima**: 57.0 V ($3.35V \cdot 17 = 56.95V$)

Por otro lado, para las celdas Panasonic PA-L154.K01 se pueden ver el resto de características en su ficha técnica (punto 13.4), además de su tensión nominal y su capacidad de 3.6 V y 2.25 Ah, respectivamente, de la cual se han extraído los siguientes límites.

- **Tensión límite máxima**: 4.3 V
- **Tensión límite mínima**: 2.25 V

Sistema de medida de tensión

En primer lugar, el sistema de medida de tensión está compuesto por un Arduino UNO, el cual se comunica con la computadora a través de una conexión USB que utiliza un protocolo de comunicación serie. Para ello, el Arduino emplea un chip FT232RL que es capaz de emular un puerto serie RS232 utilizando un puerto USB. Se configura del siguiente modo:

- **Tasa de baudios**: 9600
- **Bit de paridad**: EVEN (E)

- **Bits de datos**: 8 bits
- **Bits de parada**: 1 bit
- **ID**: 1
- **Pin indicador**: 13

Además, cuenta con 6 entradas analógicas de 10 bits, las cuales soportan señales de entre $+V_{CC}$ y $-V_{CC}$, siendo V_{CC} la tensión de alimentación. En este caso 5 V, ya que se alimenta a través del puerto USB.

Teniendo en cuenta la naturaleza de las entradas analógicas del Arduino, las cuales tienen una resolución de 10 bits, es importante considerar su aplicabilidad en el sistema de medida de tensión de las baterías. Dado que la herramienta se utilizará específicamente para medir la tensión de la batería en uso, se puede afirmar que la tensión medida será igual a la tensión de la batería. Además, se puede establecer que la tensión de la batería siempre será mayor o igual a cero, ya que las baterías no pueden proporcionar voltajes negativos.

Antes de llegar al Arduino, la tensión de la batería pasa a través de un divisor de tensión, el cual tiene como objetivo ajustar la tensión de entrada de modo que la señal analógica resultante se encuentre dentro del rango aceptable para el Arduino, que en este caso es de 0 a 5 V. Al utilizar un divisor de tensión, se logra adecuar la tensión máxima que el sistema puede manejar a 5 V. Por lo tanto, la señal analógica de entrada estará confinada en el rango de 0 a 5 V.

Considerando que el Arduino cuenta con una resolución de 10 bits, lo cual implica 2^{10} valores digitales o 1024 valores, la transformación de la señal analógica de 0 a 5 V a un valor digital de 0 a 1023 se realiza de manera lineal. En otras palabras, cada incremento de 5 mV en la señal analógica se traduce en un incremento de 1 en el valor digital.

Por ejemplo, si estamos trabajando con la batería de la moto y se ha establecido previamente que su límite de tensión máxima es de 71 V, podemos determinar la precisión de medida de tensión. Considerando la relación entre la tensión medida y el valor digital, se obtiene una precisión de 69 mV por incremento en el valor digital (según se muestra en la ecuación 3.5). Esta información es relevante para comprender la capacidad de resolución del sistema de medida y evaluar su idoneidad para el propósito deseado.

$$[0, 71]V \rightarrow [0, 5]V \rightarrow [0, 1023] \tag{3.3}$$

$$resolución = \frac{1}{2^{10}} = 0.0009765 \tag{3.4}$$

$$\frac{71}{2^{10}} = 0.069V \rightarrow 69mV \tag{3.5}$$

Esta precisión sería muy baja para las aplicaciones para las que se quiere emplear la plataforma ya que se toman medidas con un tiempo de muestreo de 5 segundos y con una precisión de 69 mV no se distinguen cambios entre medida y medida, es por ello, que se ha decidido utilizar un ADC para aumentarla.

El ADC ADS1115 se comunica con el Arduino mediante el protocolo I2C a través de los pines SCL (clock) y SDA (data) y posee una resolución de 16 bits, de los cuales,

uno esta reservado para el signo (positivo o negativo) y los otros quince para el valor, ya que la salida que se obtiene del ADC es un entero con signo. De esta manera, se obtienen 32768 valores posibles (2^{15}) entre 0 y 32767 lo que resulta en una resolución de

$$\frac{1}{2^{15}} = 0.00003052 \tag{3.6}$$

Además, otra de las particularidades que dotan al ADS1115 de todavía mayor precisión y fiabilidad es el amplificador de ganancia programable o PGA que lleva incorporado. Básicamente, se trata de un amplificador operacional al que se puede modificar la ganancia a través del código, programando. El PGA establece la escala completa es decir, indica el valor de referencia. En Arduino este valor viene determinado por el voltaje de referencia que en el caso de Arduino UNO es 5V. En el ADS1115 lo establece el PGA, por defecto este valor de referencia es ±6.144 V, lo que quiere decir que el valor de 32.677 (valor máximo con 15 bits) corresponde a 6,144 V. De modo que, se calcula el factor de escala (FE) del ADC como

$$FE = \frac{6.144}{2^{15}} = 0.0001875V = 0.1875mV \tag{3.7}$$

Pero se puede cambiar el valor de referencia a valores menores. En la tabla 3.3 se muestra un resumen de todos los valores posibles del PGA.

PGA	**Referencia (V)**	**Factor de escala (mV)**
2/3	6.144	0.1875
1	4.096	0.1250
2	2.048	0.0625
4	1.024	0.0312
8	0.512	0.0156
16	0.256	0.0078

Tabla 3.3: Resumen de todos los valores posibles del PGA

Por otro lado, el convertidor posee varios modos de funcionamiento de los cuales uno en particular es oportunamente útil para aplicaciones de lectura de tensión de baterías y sistemas como el de este proyecto donde la tensión viene dada por la tensión de la batería. Se trata del modo diferencial, el cual mide la diferencia entre dos de las entradas analógicos del ADS1115 es decir, se realiza una medición en la que ninguno de los lados se encuentra en GND. En este modo, se emparejan los pines de dos en dos, A0 con A1 y A2 con A3, lo que significa que sólo se tienen dos canales. El canal 1 devuelve la diferencia entre los pines A0 y A1 y el canal 2 devuelve la diferencia entre los pines A2 y A3. Esto resulta realmente útil para mejorar la precisión en la medida de tensión de baterías ya que estas, suelen trabajar en un rango de tensión determinado muy restringido, por ejemplo en el caso de la batería de la moto con la que se trabaja en este proyecto se ha definido una tensión límite máxima (V_{LIM_MAX}) y mínima (V_{LIM_MIN}) de 71 V y 57 V, respectivamente. Es decir, solo interesa leer entre la V_{LIM_MAX} y la V_{LIM_MIN} no entre la V_{LIM_MAX} y 0, lo que haría perder mucha precisión. De esta forma, el primer pin de la pareja recibe la señal de la tensión de la

batería y el segundo, recibe la señal de la V_{LIM_MIN}. De este modo se acorta el rango de medida a $(V_{LIM\_MAX} - V_{LIM\_MIN})$ V y no se mantiene en $(V_{LIM\_MAX} - 0)$ V. Además, otro beneficio de este modo es que se pueden realizar mediciones de voltajes negativos, lo que permite se pueda escoger una referencia de tensión del PGA más pequeña y por tanto un factor de escala más óptimo.

Finalmente, debido a que el ADC está alimentado por el pin de alimentación del Arduino de 5 V, sus entradas analógicas solo podrán leer tensiones de entre 0 y 5.3 V, ya que, estas solo soportan tensiones de $V_{CC} + 0.3V$, donde V_{CC} es la tensión de alimentación del ADC. Esto ocasiona que en baterias más grandes haya que usar, por un lado, un divisor de tensión que reduzca el voltaje de entrada en el primer pin del convertidor y, por otro lado, un regulador zener alimentado de la misma forma que el ADC por el Arduino a 5V que fije la tensión en el segundo pin del convertidor para que la reste a la señal del primero.

En resumen, para sacar el máximo partido al ADS1115, en primer lugar, se ha de conocer el rango de tensiones con el que trabaja la batería que se esté empleando. En segundo lugar, se ha de diseñar un divisor de tensión y un regulador zener de tal forma que cumpla con la restricción de tensión de las entradas analógicas del convertidor y permita reducir al máximo la referencia de voltaje para poder elegir el PGA óptimo y ganar el máximo de precisión.

3.2.2. Funcionalidades del sistema

Ensayo de descarga

Los dispositivos implicados son la carga, la batería y el sistema de medida, conectados en paralelo, tal y como se muestra en la figura 3.2 (ver esquema completo en Planos 12.1). La carga electrónica descarga la batería a una intensidad determinada por el usuario dentro de los límites posibles de los dispositivos involucrados, es decir, un máximo de 4 A para la batería de la moto (el limitante es la carga) y un máximo de 1.25 A para la celda (el limitante es la misma celda). Esta descarga se realiza de forma constante hasta que el sistema de almacenamiento se descargue un porcentaje de su SOC, el cual también es introducido por el usuario. Una vez se llega a este punto, el sistema pasa a un periodo de relajación durante un tiempo indicado por el usuario, en el que la intensidad de la carga es 0 A y la tensión de la batería se estabiliza. Es justamente la tensión en el instante en el que termina el tiempo de relajación, la que se quiere obtener para después poder visualizar el comportamiento de la batería con curvas V-SOC del tipo de la figura 3.3. A continuación, se vuelve a repetir todo el proceso de descarga-relajación hasta que finalmente se detecta que la batería ha llegado a un determinado SOC límite (indicado por el usuario) o a una determinada tensión límite (indicada por el usuario), nunca menor que la tensión límite mínima del sistema de almacenamiento (57 V para la batería y 2.25 V para la celda).

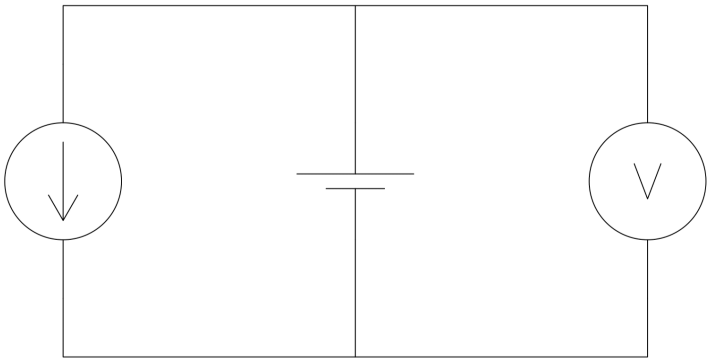


Figura 3.2: Esquema eléctrico del sistema carga-batería-sistema de medida

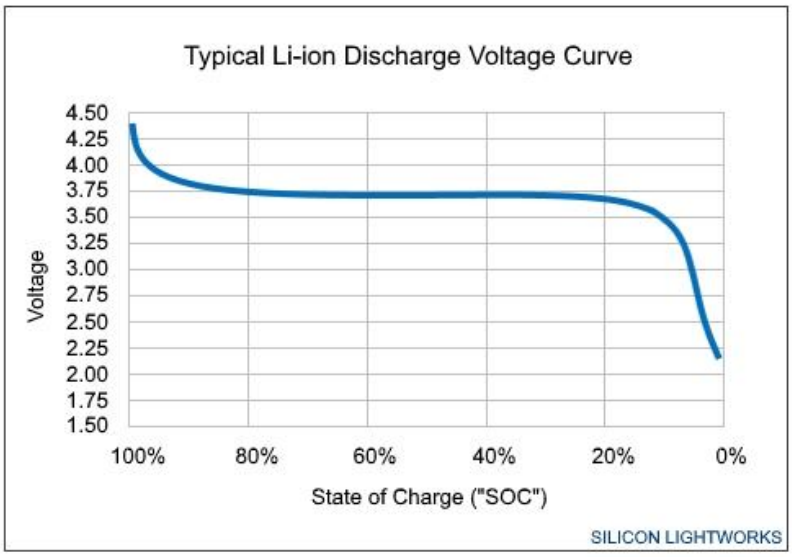


Figura 3.3: Curva típica de descarga de una batería

Por otro lado, el usuario elige un periodo de muestreo concreto, el cual debe ser mayor a 20 ms multiplicado por el numero de comandos ejecutados en ese periodo, debido a las limitaciones que presenta la carga. Durante este periodo se recogen las medidas de tensión (con el sistema de medida), de intensidad (usando la lectura de la propia carga), de tiempo, y se calcula el SOC.

Para calcular el SOC, previamente, el usuario introduce los parámetros del sistema de almacenamiento que se está empleando, entre los cuales se incluyen la capacidad y el SOC inicial. A partir de estos datos se calcula el SOC siguiendo la ecuación 3.8.

$$SOC_i = SOC_{i-1} - \frac{I_{d,i} \cdot (t/3600)}{C} \cdot 100 \quad \forall i = 1, ..., k \tag{3.8}$$

$$SOC_0 = 100 \tag{3.9}$$

Donde, i representa cada muestra; k es el número total de muestras; SOC_i es el estado de carga de una determinada muestra, dado en porcentaje; SOC_{i-1} es el estado de carga en la muestra anterior, dado en porcentaje; $I_{d,i}$ es la intensidad de descarga en una determinada muestra, en A; t es el periodo de muestreo, en segundos; C es la capacidad del sistema de almacenamiento, en Ah; SOC_0 es el estado de carga inicial.

Ensayo de carga

Los dispositivos implicados son la fuente de alimentación, la batería y el sistema de medida, conectados en paralelo tal y como se muestra en la figura 3.4 (ver esquema completo en Planos 12.2). La fuente carga la batería a una intensidad determinada por el usuario dentro de los límites posibles de los dispositivos involucrados, es decir, un máximo de 10 A para la batería de la moto (el limitante es la fuente) y un máximo de 1.2 A para la celda (el limitante es la misma celda). Esta carga se realiza de forma constante hasta que el sistema de almacenamiento se cargue un porcentaje de su SOC, el cual también es introducido por el usuario. Una vez se llega a este punto, el sistema pasa a un periodo de relajación durante un tiempo indicado por el usuario, en el que la intensidad de la fuente es 0 A y la tensión de la batería se estabiliza. Es justamente la tensión en el instante en el que termina el tiempo de relajación, la que se quiere obtener para después poder visualizar el comportamiento de la batería con curvas V-SOC. A continuación, se vuelve a repetir todo el proceso de carga-relajación hasta que finalmente se detecta que la batería llega a un determinado límite de tensión de corriente constante (indicado por el usuario). Cuando llega a esta tensión la fuente pasa de cargar a la batería en modo de corriente constante a cargarla en modo de tensión constante, de modo, que en este punto la tensión se fija en este valor límite y a medida que la batería se va cargando, su intensidad disminuye hasta llegar a un umbral de intensidad (fijado por el usuario), momento en el cual terminará el ensayo. Notar, que a pesar de que se cambie a modo de carga en tensión constante, se sigue respetando el ciclo de carga-relajación con las mismas condiciones.

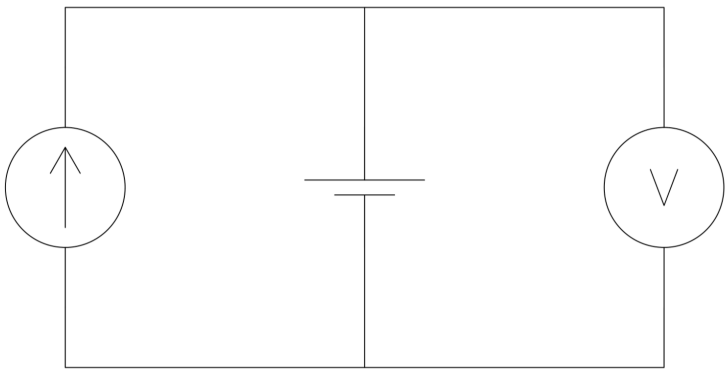


Figura 3.4: Esquema eléctrico del sistema fuente-batería-sistema de medida

Normalmente, tanto la tensión límite de corriente constante, como la intensidad umbral vienen dadas en las fichas técnicas de los sistemas de almacenamiento. En cualquier caso, estos valores deben ser lo suficientemente bajos como para no superar la tensión

máxima de la batería. En el caso de la batería de la moto, estos valores se han hallado de forma experimental midiendo la tensión y la intensidad de la carga de la batería con su propio cargador (tensión límite de corriente constante de 71 V y intensidad umbral de 0.4 A). En el caso de la celda, según su ficha técnica, la tensión limite de corriente constante es 4.2 V y la intensidad umbral es 0.1 A.

Por otro lado, el usuario elige un periodo de muestreo concreto, el cual debe ser mayor a 1 segundo multiplicado por el numero de comandos ejecutados en ese periodo, debido a las limitaciones que presenta la fuente. Durante este periodo se recogen las medidas de tensión (con el sistema de medida), de intensidad (usando la lectura de la propia fuente), de tiempo, se calcula el SOC.

Para calcular el SOC, previamente, el usuario introduce los parámetros del sistema de almacenamiento que se está empleando, entre los cuales la capacidad y el SOC inicial. A partir de estos datos se calcula el SOC siguiendo la ecuación 3.10.

$$SOC_i = SOC_{i-1} + \frac{I_{c,i} \cdot (t/3600)}{C} \cdot 100 \quad \forall i = 1, ..., k \tag{3.10}$$

$$SOC_0 = SOC_{d,k} \tag{3.11}$$

Donde, i representa cada muestra; k es el número total de muestras; SOC_i es el estado de carga de una determinada muestra, dado en porcentaje; SOC_{i-1} es el estado de carga en la muestra anterior, dado en porcentaje; $I_{c,i}$ es la intensidad de carga en una determinada muestra, en A; t es el periodo de muestreo, en segundos; C es la capacidad del sistema de almacenamiento, en Ah; SOC_0 es el estado de carga inicial; $OC_{d,k}$ es el estado de carga final de la descarga.

Descarga siguiendo un perfil

Los dispositivos implicados son la carga, la batería y el sistema de medida, conectados tal y como se muestra en la figura 3.2 (ver esquema completo en Planos 12.1). La carga descarga la batería siguiendo un perfil de intensidades proporcionado por una hoja de cálculo $xlsx$. La hoja está formada por una columna con los datos de las intensidades en A y por otra con los datos del tiempo en segundos que se debe aplicar la correspondiente intensidad. De esta forma, la carga descarga la batería aplicando la intensidades que marque la hoja de calculo, durante el intervalo de tiempo que le corresponda a cada corriente.

Las limitaciones son las mismas que las especificadas en el ensayo de descarga. Además, como en el ensayo, la descarga se realiza de la misma forma, con la diferencia de que otro modo de que termine la descarga es que llegue al final del perfil de descarga.

Igualmente, tanto el sistema de muestreo como el método de cálculo del SOC son los mismos que en el ensayo de descarga.

Carga siguiendo un perfil

Los dispositivos implicados son la fuente de alimentación, la batería y el sistema de medida, conectados tal y como se muestra en la figura 3.4 (ver esquema completo en

Planos 12.2). La fuente carga la batería siguiendo un perfil de intensidades proporcionado por una hoja de cálculo *.xlsx*. La hoja está formada por una columna con los datos de las intensidades en A y por otra con los datos del tiempo en segundos que se debe aplicar la correspondiente intensidad. De esta forma, la fuente carga la batería aplicando la intensidades que marque la hoja de calculo, durante el intervalo de tiempo que le corresponda a cada corriente.

Las limitaciones son las mismas que las especificadas en el ensayo de carga. Además, como en el ensayo, la carga se realiza de la misma forma, con la diferencia de que otro modo de que termine la descarga es que llegue al final del perfil de descarga. También, los límites de corriente constante, como la intensidad umbral, son los mismos que los mencionados en el ensayo de carga.

Igualmente, tanto el sistema de muestreo como el método de calculo del SOC son los mismos que en el ensayo de carga.

Descarga constante

Los dispositivos implicados son la carga, la batería y el sistema de medida, conectados tal y como se muestra en la figura 3.2 (ver esquema completo en Planos 12.1). La carga descarga la batería a una intensidad constante. Esta descarga se realiza de forma constante durante todo el tiempo hasta que el sistema de almacenamiento llega a un porcentaje determinado de SOC, o a una determinada tensión límite nunca menor que la tensión límite mínima del sistema de almacenamiento.

Las limitaciones son las mismas que las especificadas en el ensayo de descarga. Además, como en el ensayo, la descarga se realiza de la misma forma.

Igualmente, tanto el sistema de muestreo como el método de cálculo del SOC son los mismos que en el ensayo de descarga.

Carga constante

Los dispositivos implicados son la fuente de alimentación, la batería y el sistema de medida, conectados tal y como se muestra en la figura 3.4 (ver esquema completo en Planos 12.2). La fuente carga la batería a una intensidad constante. Esta carga se realiza de forma constante durante todo el tiempo hasta que el sistema de almacenamiento llega a la tensión límite de corriente constante, momento en el cual la tensión se fija en este valor y cambia a cargar en modo tensión constante hasta llegar a una determinada intensidad umbral, momento en el que termina la carga.

Las limitaciones son las mismas que las especificadas en el ensayo de carga. Además, como en el ensayo, la carga se realiza de la misma forma. También, los límites de corriente constante, como la intensidad umbral, son los mismos que los mencionados en el ensayo de carga.

Igualmente, tanto el sistema de muestreo como el método de cálculo del SOC son los mismos que en el ensayo de carga.

Simulación de un sistema de FV con curvas de irradiancia y consumo

Se emplean todos los dispositivos del sistema (fuente de alimentación, carga, batería y sistema de medida) conectados en paralelo, figura 3.5 (ver esquema completo en Planos 12.3). Esta función utiliza una curva de irradiancia y una curva de consumo para simular el funcionamiento de un sistema fotovoltaico con almacenamiento de energía.

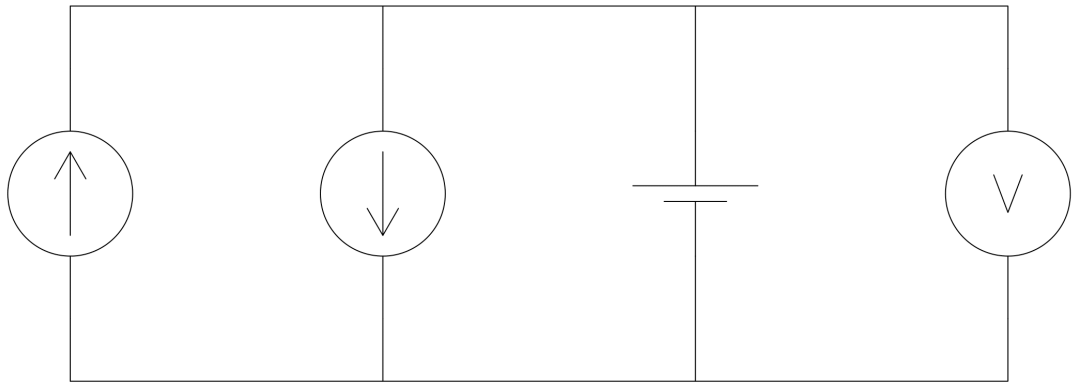


Figura 3.5: Esquema eléctrico del sistema fuente-carga-batería-sistema de medida

En primer lugar, la curva de irradiancia contiene los datos de un día de la irradiancia global de un determinado lugar. Estos datos pueden estar definidos por cualquier tipo intervalo de tiempo, siempre que sea el mismo (horas, cuartos de hora, minutos, etc.). Además, el usuario introduce una serie de parámetros referentes al sistema de FV que se quiere simular:

- Performance ratio
- Potencia instalada
- Radiación estándar

A partir de aquí, se calcula la potencia generada por el sistema y se escala debidamente para que la fuente establezca un valor de corriente equivalente de la siguiente manera (ecuaciones 3.12, 3.13 y 3.14).

$$SCL = \frac{PR \cdot G_{max} \cdot P_{inst}}{Radiacion_{STC}} / I_{max,PS} \tag{3.12}$$

$$P_{FV,i} = \frac{PR \cdot G_i \cdot P_{inst}}{Radiacion_{STC}} \rightarrow \tag{3.13}$$

$$I_{SCL,i} = \frac{P_{FV,i}}{SCL} \tag{3.14}$$

Donde, SCL es el factor de escala de la intensidad; PR es el performance ratio del sistema; G_{max} es la irradiancia máxima de la curva; P_{inst} es la potencia instalada del sistema fotovoltaico, $Radiacion_{STC}$ es la radiación estándar, comúnmente 1000 W/m^2;

$I_{max,PS}$ es la intensidad máxima que puede establecer la fuente de alimentación; $P_{FV,i}$ es la potencia generada por el sistema de FV en un determinado instante; G_i es la irradiancia en un determinado instante; $I_{SCL,i}$ es la intensidad equivalente a la potencia generada por el sistema de FV, debidamente escalada para que pueda ser establecida por la fuente de alimentación.

En segundo lugar, la curva de consumo contiene los datos de un día de un consumo determinado. Estos datos pueden estar definidos por cualquier tipo intervalo de tiempo, siempre que sea el mismo (horas, cuartos de hora, minutos, etc.).

En este punto, se calcula el factor de escala debidamente para que la carga establezca un valor de corriente equivalente a la potencia consumida de la siguiente manera (ecuaciones 3.15 y 3.16).

$$SCL = \frac{P_{max}}{I_{max,LOAD}} \tag{3.15}$$

$$I_{SCL,i} = \frac{P_i}{SCL} \tag{3.16}$$

Donde, SCL es el factor de escala de la intensidad; ; P_{max} es la potencia consumida máxima de la curva; $I_{max,LOAD}$ es la intensidad máxima que puede establecer la carga; P_i es la potencia consumida en un determinado instante; $I_{SCL,i}$ es la intensidad equivalente a la potencia consumida, debidamente escalada para que pueda ser establecida por la carga.

Una vez obtenidas las curvas el funcionamiento es similar a la carga y descarga de la batería siguiendo un perfil de intensidades. Por un lado, la fuente va estableciendo las intensidades equivalentes de la curva de irradiancia durante el tiempo correspondiente y, por otro, la carga va estableciendo las intensidades equivalentes de la curva de consumos durante el tiempo correspondiente. De forma que, el comportamiento de la batería responde a las ecuaciones 3.17 y 3.18.

$$I_{bat,i} = I_{PS,i} - I_{LOAD,i} \tag{3.17}$$

Donde, $I_{bat,i}$ es la intensidad de la batería en un cierto instante; $I_{PS,i}$ es la intensidad de la fuente de alimentación en un cierto instante; y $I_{LOAD,i}$ es la intensidad de la carga en un cierto instante.

$$SOC_i = SOC_{i-1} + \frac{I_{bat,i} \cdot (t/3600)}{C} \cdot 100 \tag{3.18}$$

Donde, SOC_i es el estado de carga de la batería en un cierto instante en porcentaje; SOC_{i-1} es el estado de carga de la batería en el instante anterior en porcentaje; $I_{bat,i}$ es la intensidad de la batería en un cierto instante en A; t es el periodo de muestreo en segundos; C es la capacidad de la batería en Ah.

La simulación terminará cuando alguna de las dos curvas termine. Además, si se detecta que la batería estuviera muy llena (la tensión supera a la tensión límite máxima de la batería) y la corriente de la fuente es mayor que la de la carga, se limita la corriente de la fuente a la de la carga, hasta que alguna de alguna de estas condiciones se deje de cumplir. De la misma forma, si se detecta que la batería estuviera muy vacía (la tensión supera a la tensión límite mínima de la batería) y la corriente de la carga es

mayor que la de la fuente, se limita la corriente de la carga a la de la fuente, hasta que alguna de estas condiciones se deje de cumplir.

Tanto las intensidades de la fuente, de la carga, de la batería, la tensión del sistema, el SOC de la batería y el tiempo se van mostrando por pantalla a medida que se actualiza la información y se va guardando en un fichero $json$, el cual al terminar la simulación volcará todos los datos para generar las siguientes gráficas.

- SOC vs t
- V vs t
- $I_{bat}, I_{PS}, I_{LOAD}$ vs t

Simulación de un sistema de FV con perfiles de intensidad

Se emplean todos los dispositivos del sistema (fuente de alimentación, carga, batería y sistema de medida) conectados en paralelo, figura 3.5 (ver esquema completo en Planos 12.3). Esta función utiliza un perfil de intensidades para la fuente y otro para la carga, similares a los definidos en la carga y descarga siguiendo un perfil de intensidades. De modo que, tanto la fuente como la carga van siguiendo sus respectivos perfiles al mismo tiempos simulando así el funcionamiento de un sistema fotovoltaico con almacenamiento de energía.

Las limitaciones y restricciones de esta funcionalidad son las mismas que para el caso de la carga y descarga siguiendo un perfil de intensidades. Además, el comportamiento de la batería es el mismo que el definido en las ecuaciones 3.17 y 3.18 y, como en el caso anterior, la simulación termina cuando uno de los dos perfiles termina y, también, implementa el mismo método de seguridad ante un llenado o vaciado excesivo de la batería.

Por último, se muestran por pantalla los mismos resultado que en la funcionalidad anterior, se guardan de la misma forma y, posteriormente, se representan las misma gráficas.

Lectura de tensión

Se puede emplear con cualquier configuración del sistema, simplemente conectándolo en paralelo al sistema de medida. En este modo, el usuario introduce el periodo de muestro con el que se quiere leer la tensión y se muestra por pantalla la medida de tensión del sistema al que este conectado con la frecuencia indicada.

Calibración de la fuente de alimentación

La fuente de alimentación es el único dispositivo involucrado en esta funcionalidad, la finalidad de la cual es calibrar debidamente los diferentes potenciómetros de la fuente para que establezca y mida la tensión correctamente.

Para ello, el usuario debe de elegir un potenciómetro numerado del 1 al 5 y, a continuación, un valor de calibración que se debe corresponder con un entero entre 0 y

255, también puede escoger la opción *default* que devuelve al sistema sus valores por defecto.

Para ver con más detalle el proceso de calibración y como realizarla correctamente, puede consultarse a la guía de fuente de alimentación.

Ensayo de calibración

El ensayo de calibración es una funcionalidad que permite comprobar si la calibración de la fuente de alimentación ha sido la correcta. El sistema está formado por la fuente de alimentación, una resistencia de 100 Ω y el sistema de medida, conectados en paralelo como se muestra en la figura 3.6 (ver esquema completo en Planos 12.4).

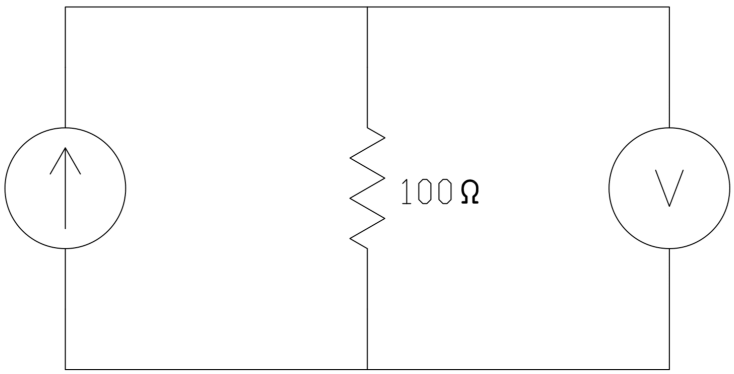


Figura 3.6: Esquema eléctrico para el ensayo de calibración

El ensayo consiste en ir estableciendo varios voltajes durante unos segundos cada uno y recoger la medida de tensión con la lectura de la propia máquina y con el sistema de medida. Estas medidas se comparan entre si y con la tensión de referencia, calculando el error absoluto y relativo.

A continuación, tanto las dos medidas como la tensión de referencia y los errores se guardan y se representan las siguientes gráficas.

- Curva de error
- Curva de error relativo
- Curvas de la tensión medida con la máquina, de la tensión medida con el sistema de medida, y de la tensión de referencia

Apagar la fuente de alimentación

Esta funcionalidad permite apagar la fuente de alimentación en el caso que por algún tipo de error que no se haya contemplado en este proyecto se quede encendida.

Apagar la carga

Esta funcionalidad permite apagar la carga en el caso que por algún tipo de error que no se haya contemplado en este proyecto se quede encendida.

3.2.3. Análisis del software

El objetivo del análisis del software es transformar la definición de requisitos en una especificación de software. La especificación de software es una documentación completa y precisa de qué tiene que realizar el sistema para cubrir los requisitos de usuario y las restricciones físicas de los componentes de la herramienta.

En este apartado se muestra el diagrama de Clases/Agentes y su documentación, este diagrama consiste en una visión general de las clases, los atributos, los métodos y las asociaciones que dan lugar a la aplicación software.

Diagrama de Clases/Agentes

El diagrama de clases representa una primera abstracción de la información que se debe empaquetar y manipular en el sistema de software y el flujo de información y ejecución del programa. Se muestran las características, las relaciones y las funcionalidades de alto nivel, para ello se debe tener en cuenta, las características del sistema y los casos de usos definidos de manera explícita en los puntos 3.2.1 y 3.2.2.

En el anexo 16, se puede observar el diagrama de clases de la aplicación, en él se pueden ver todas las clases del software y como se relacionan entre si.

Aunque, se trata de un diagrama de clases, en el aparecen una serie de módulos adicionales, que no contienen ningún tipo de clase. Son módulos auxiliares cuya labor es indispensable para el correcto funcionamiento del resto de programa.

En el diagrama se pueden observar nodos de distintos colores para distinguir fácilmente la naturaleza de cada módulo. El naranja es el módulo principal del programa; el morado son los módulos que implementan algún Agente que funciona de forma independiente, el verde son los módulos que implementan algún Agente que funciona en ejecución multithreading; el rojo es el Agente de lectura de tensión, el cual según la funcionalidad, se ejecuta de forma individual o junto a otros hilos; el azul son los módulos auxiliares; el rosa es el archivo de configuración.

Documentación del Diagrama de Clases/Agentes

La Documentación del Diagrama de Clases/Agentes proporciona una explicación de todas las clases de la aplicación, así como una vista lógica de los cuadros que incluyen todas las características, atributos, métodos y relaciones que se ven en el diagrama.

Desde el cuadro 3.4 al cuadro 3.19, se describen cada una de las clases del diagrama de clases, se definen sus atributos, se enumeran sus métodos y se establecen las asociaciones con otras clases o módulos.

Clase	chargeTestAgent()
Hardware	- Fuente de alimentación - Sistema de medida
Descripción	Este agente se encarga de realizar ensayos de carga de la batería según las especificaciones del usuario
Funcionalidad	Ensayo de carga (punto 3.2.2)
Atributos	- powerSupply: Objeto que representa la fuente de alimentación - readVolt: Objeto que representa el sistema de medida - C: Capacidad de la batería - V_LIM_HIGH: Tensión límite máxima de la batería - soc: Estado de carga de la batería - V_LIM_CC: Tensión límite de corriente continua - I_UMB: Intensidad umbral - Ic: Intensidad de carga - c: Porcentaje de carga de un escalón de carga-relajación - Ts: Periodo de relajación - tm: Periodo de muestreo - id: Identificador - data: diccionario de listas de los datos obtenidos (tensión, SOC, intensidad, tiempo)
Métodos	resetSocket(), turnOn(), firstRead), charge(), stabilize(), turnOff(), queryCURR(), response(), readCURR(), run()
Asociaciones	- Envía los datos obtenidos mediante un callback a main(), el cual se encarga de almacenarlos en un archivo *csv* - Para leer la tensión del sistema se emplea el objeto readVolt, que es un objeto de la clase readVoltAgent(9 - Llama a la función delay_seconds() ubicada en functions.py

Tabla 3.4: Clase: chargeTestAgent()

Clase	dischargeTestAgent()
Hardware	- Carga - Sistema de medida
Descripción	Este agente se encarga de realizar ensayos de descarga de la batería según las especificaciones del usuario
Funcionalidad	Ensayo de descarga (punto 3.2.2)
Atributos	- load: Objeto que representa la carga - readVolt: Objeto que representa el sistema de medida - C: Capacidad de la batería - V_LIM_HIGH: Tensión límite máxima de la batería - V_LIM_LOW: Tensión límite mínima de la batería - soc: Estado de carga de la batería - V_MIN: Tensión límite mínima a la que puede llegar la descarga - SOC_MIN: SOC minimo que debe alcanzar la batería durante la descarga - Id: Intensidad de descarga - c: Porcentaje de descarga de un escalón de carga-relajación - Ts: Periodo de relajación - tm: Periodo de muestreo - id: Identificador - data: diccionario de listas de los datos obtenidos (tensión, SOC, intensidad, tiempo)
Métodos	resetSerial(), turnOn(), firstRead), discharge(), stabilize(), turnOff(), queryCURR(), response(), readCURR(), run()
Asociaciones	- Envía los datos obtenidos mediante un callback a main(), el cual se encarga de almacenarlos en un archivo *csv* - Para leer la tensión del sistema se emplea el objeto readVolt, que es un objeto de la clase readVoltAgent() - Llama a la función delay_seconds() ubicada en functions.py

Tabla 3.5: Clase: dischargeTestAgent()

Clase	chargeProfileAgent()
Hardware	- Fuente de alimentación - Sistema de medida
Descripción	Este agente se encarga de realizar cargas de la batería siguiendo un perfil de intensidades
Funcionalidad	Carga siguiendo un perfil (punto 3.2.2)
Atributos	- powerSupply: Objeto que representa la fuente de alimentación - readVolt: Objeto que representa el sistema de medida - C: Capacidad de la batería - V_LIM_HIGH: Tensión límite máxima de la batería - soc: Estado de carga de la batería - V_LIM_CC: Tensión límite de corriente continua - I_UMB: Intensidad umbral - profile: path o nombre del archivo *xlsx* que contiene el perfil de intensidades - Ic: Lista de las intensidades de carga del perfil - interval: Intervalos de tiempo de las intensidades del perfil - tm: Periodo de muestreo - id: Identificador - data: diccionario de listas de los datos obtenidos (tensión, SOC, intensidad, tiempo)
Métodos	resetSocket(), turnOn(), firstRead), charge(), turnOff(), queryCURR(), response(), readCURR(), run()
Asociaciones	- Envía los datos obtenidos mediante un callback a main(), el cual se encarga de almacenarlos en un archivo *csv* - Para leer la tensión del sistema se emplea el objeto readVolt, que es un objeto de la clase readVoltAgent() - Llama a la función delay_seconds() ubicada en functions.py - Llama a la función parseExcel() ubicada en functions.py

Tabla 3.6: Clase: chargeProfileAgent()

Clase	dischargeProfileAgent()
Hardware	- Carga - Sistema de medida
Descripción	Este agente se encarga de realizar descargas de la batería siguiendo un perfil de intensidades
Funcionalidad	Descarga siguiendo un perfil (punto 3.2.2)
Atributos	- load: Objeto que representa la carga - readVolt: Objeto que representa el sistema de medida - C: Capacidad de la batería - V_LIM_HIGH: Tensión límite máxima de la batería - V_LIM_LOW: Tensión límite mínima de la batería - soc: Estado de carga de la batería - V_MIN: Tensión límite mínima a la que puede llegar la descarga - SOC_MIN: SOC minimo que debe alcanzar la batería durante la descarga - profile: path o nombre del archivo *xlsx* que contiene el perfil de intensidades - Id: Lista de las intensidades de descarga del perfil - interval: Intervalos de tiempo de las intensidades del perfil - tm: Periodo de muestreo - id: Identificador - data: diccionario de listas de los datos obtenidos (tensión, SOC, intensidad, tiempo)
Métodos	resetSerial(), turnOn(), firstRead), discharge(), turnOff(), queryCURR(), response(), readCURR(), run()
Asociaciones	- Envía los datos obtenidos mediante un callback a main(), el cual se encarga de almacenarlos en un archivo *csv* - Para leer la tensión del sistema se emplea el objeto readVolt, que es un objeto de la clase readVoltAgent() - Llama a la función delay_seconds() ubicada en functions.py - Llama a la función parseExcel() ubicada en functions.py

Tabla 3.7: Clase: dischargeProfileAgent()

Clase	chargeConstantAgent()
Hardware	- Fuente de alimentación - Sistema de medida
Descripción	Este agente se encarga de realizar la carga de la batería a una intensidad constante
Funcionalidad	Carga constante (punto 3.2.2)
Atributos	- powerSupply: Objeto que representa la fuente de alimentación - readVolt: Objeto que representa el sistema de medida - C: Capacidad de la batería - V_LIM_HIGH: Tensión límite máxima de la batería - soc: Estado de carga de la batería - V_LIM_CC: Tensión límite de corriente continua - I_UMB: Intensidad umbral - Ic: Intensidad de carga - tm: Periodo de muestreo - id: Identificador - data: diccionario de listas de los datos obtenidos (tensión, SOC, intensidad, tiempo)
Métodos	resetSocket(), turnOn(), firstRead), charge(), turnOff(), queryCURR(), response(), readCURR(), run()
Asociaciones	- Envía los datos obtenidos mediante un callback a main(), el cual se encarga de almacenarlos en un archivo *csv* - Para leer la tensión del sistema se emplea el objeto readVolt, que es un objeto de la clase readVoltAgent() - Llama a la función delay_seconds() ubicada en functions.py

Tabla 3.8: Clase: chargeConstantAgent()

Clase	dischargeConstantAgent()
Hardware	- Carga - Sistema de medida
Descripción	Este agente se encarga de realizar la descarga de la batería a una intensidad constante
Funcionalidad	Descarga constante (punto 3.2.2)
Atributos	- load: Objeto que representa la carga - readVolt: Objeto que representa el sistema de medida - C: Capacidad de la batería - V_LIM_HIGH: Tensión límite máxima de la batería - V_LIM_LOW: Tensión límite mínima de la batería - soc: Estado de carga de la batería - V_MIN: Tensión límite mínima a la que puede llegar la descarga - SOC_MIN: SOC minimo que debe alcanzar la batería durante la descarga - Id: Intensidad de descarga - tm: Periodo de muestreo - id: Identificador - data: diccionario de listas de los datos obtenidos (tensión, SOC, intensidad, tiempo)
Métodos	resetSerial(), turnOn(), firstRead), discharge(), turnOff(), queryCURR(), response(), readCURR(), run()
Asociaciones	- Envía los datos obtenidos mediante un callback a main(), el cual se encarga de almacenarlos en un archivo *csv* - Para leer la tensión del sistema se emplea el objeto readVolt, que es un objeto de la clase readVoltAgent() - Llama a la función delay_seconds() ubicada en functions.py

Tabla 3.9: Clase: dischargeConstantAgent()

Clase	powerSupplyPV()
Hardware	Fuente de alimentación
Descripción	Este agente se encarga de simular el funcionamiento de la parte de la generación un sistema de FV con almacenamiento a partir de una curva de irradiancia
Funcionalidad	Simulación de un sistema de FV con curvas de irradiancia y consumo (punto 3.2.2)
Atributos	- powerSupply: Objeto que representa la fuente de alimentación - PR: Preformance ratio - P_inst: Potencia instalada - STCRadiation: Radiación estándar - G_max: Irradiancia máxima de la curva - I_max: Intensidad máxima de la fuente de alimentación - timeSCL: Escala de tiempo de la curva - file: path o nombre del archivo que contiene la curva de irradiancia - tg: Periodo de cambio de la irradiancia - G: Lista con los datos de irradiancia de la curva - t: Lista de tiempos de simulación - P: Lista de la potencia generada en cada instante - I: Lista de corrientes de salida de la fuente - SCL: Factor de escala - t_load: Periodo de cambio de la intensidad de la carga - n_items: Número de elementos de la curva de consumos - id: Identificador
Métodos	resetSocket(), turnOnPS(), turnOffPS(), calculatePeriod(), cloneCurrent(), calculateScale(), isValidTime(), queryCURR(), response(), readCURR(), parseCurve(), calculateCurrent(), simulate(), run1(), run2()
Asociaciones	- Envía los datos obtenidos mediante un callback a main(), el cual se encarga de realizar una serie de cálculos con ellos y almacenarlos en un archivo *json* - Los métodos de esta clase se ejecutan al mismo tiempo que los de loadPV() y readVoltAgent() empleando multithreading - Se comunica con las clases del resto de hilos a través de funciones que usan variables globales definidas en el módulo functions.py para: ▪ Enviar y recibir errores de comunicación del software con los distintos dispositivos: communicationError() ▪ Enviar y recibir bandera de fin de la simulación: endSimulation() ▪ Enviar y recibir warnings en caso que no se cumplan los requisitos de simulación con la batería (ver sistema de seguridad en el punto xx): warningPS ▪ Enviar y recibir los periodos de cambio de intensidad de la carga como parte de la implementación del sistema de seguridad: sendTimePS y receiveTimePS() - Llama a la función delay_seconds() ubicada en functions.py

Tabla 3.10: Clase: powerSupplyPV()

Clase	loadPV()
Hardware	Carga
Descripción	Este agente se encarga de simular el funcionamiento de la parte del consumo de un sistema de FV con almacenamiento a partir de una curva de consumo
Funcionalidad	Simulación de un sistema de FV con curvas de irradiancia y consumo (punto 3.2.2)
Atributos	- load: Objeto que representa la carga - P_max: Consumo máximo de la curva - I_max: Intensidad máxima de la carga - timeSCL: Escala de tiempo de la curva - file: path o nombre del archivo que contiene la curva de irradiancia - tg: Periodo de cambio de la irradiancia - hour: Lista de cada hora de la curva - minute: Lista de cada minuto de la curva - P: Lista de la potencia consumida en cada instante - I: Lista de corrientes de entrada en la carga - SCL: Factor de escala - t_ps: Periodo de cambio de la intensidad de la fuente - n_items: Número de elementos de la curva de irradiancia - id: Identificador
Métodos	resetSerial(), turnOnLoad(), turnOffLoad(), calculatePeriod(), cloneCurrent(), calculateScale(), queryCURR(), response(), readCURR(), parseCurve(), calculateCurrent(), simulate(), run1(), run2()
Asociaciones	- Envía los datos obtenidos mediante un callback a main(), el cual se encarga de realizar una serie de cálculos con ellos y almacenarlos en un archivo *json* - Los métodos de esta clase se ejecutan al mismo tiempo que los de powerSupplyPV() y readVoltAgent() empleando multithreading - Se comunica con las clases del resto de hilos a través de funciones que usan variables globales definidas en el módulo functions.py para: ▪ Enviar y recibir errores de comunicación del software con los distintos dispositivos: communicationError() ▪ Enviar y recibir bandera de fin de la simulación: endSimulation() ▪ Enviar y recibir warnings en caso que no se cumplan los requisitos de simulación con la batería (ver sistema de seguridad en el punto xx): warningLoad() ▪ Enviar y recibir los periodos de cambio de intensidad de la fuente como parte de la implementación del sistema de seguridad: sendTimeLoad() y receiveTimeLoad() - Llama a la función delay_seconds() ubicada en functions.py

Tabla 3.11: Clase: loadPV()

Clase	powerSupplyProfile()
Hardware	Fuente de alimentación
Descripción	Este agente se encarga de simular el funcionamiento de la parte de la generación un sistema de FV con almacenamiento a partir de un perfil de intensidades
Funcionalidad	Simulación de un sistema de FV con perfiles de intensidad (punto 3.2.2)
Atributos	- powerSupply: Objeto que representa la fuente de alimentación - profile: path o nombre del archivo que contiene el perfil de intensidades - interval: Lista con los intervalos de tiempo de establecimiento de cada intensidad del perfil - I: Lista de intensidades del perfil o de la salida de la fuente - id: Identificador
Métodos	resetSocket(), turnOnPS(), turnOffPS(), queryCURR(), response(), readCURR(), simulate(), run1(), run2()
Asociaciones	- Envía los datos obtenidos mediante un callback a main(), el cual se encarga de realizar una serie de cálculos con ellos y almacenarlos en un archivo *json* - Los métodos de esta clase se ejecutan al mismo tiempo que los de loadProfile() y readVoltAgent() empleando multithreading - Se comunica con las clases del resto de hilos a través de funciones que usan variables globales definidas en el módulo functions.py para: ▪ Enviar y recibir errores de comunicación del software con los distintos dispositivos: communicationError() ▪ Enviar y recibir bandera de fin de la simulación: endSimulation() ▪ Enviar y recibir warnings en caso que no se cumplan los requisitos de simulación con la batería (ver sistema de seguridad en el punto xx): warningPS - Llama a la función parseExcel() ubicada en functions.py - Llama a la función delay_seconds() ubicada en functions.py

Tabla 3.12: Clase: powerSupplyProfile()

Clase	loadProfile()
Hardware	Carga
Descripción	Este agente se encarga de simular el funcionamiento de la parte del consumo un sistema de FV con almacenamiento a partir de un perfil de intensidades
Funcionalidad	Simulación de un sistema de FV con perfiles de intensidad (punto 3.2.2)
Atributos	- load: Objeto que representa la carga - profile: path o nombre del archivo que contiene el perfil de intensidades - interval: Lista con los intervalos de tiempo de establecimiento de cada intensidad del perfil - I: Lista de intensidades del perfil o de la entrada en la carga - id: Identificador
Métodos	resetSerial(), turnOnLoad(), turnOffLoad(), queryCURR(), response(), readCURR(), simulate(), run1(), run2()
Asociaciones	- Envía los datos obtenidos mediante un callback a main(), el cual se encarga de realizar una serie de cálculos con ellos y almacenarlos en un archivo *json* - Los métodos de esta clase se ejecutan al mismo tiempo que los de powerSupplyProfile() y readVoltAgent() empleando multithreading - Se comunica con las clases del resto de hilos a través de funciones que usan variables globales definidas en el módulo functions.py para: ▪ Enviar y recibir errores de comunicación del software con los distintos dispositivos: communicationError() ▪ Enviar y recibir bandera de fin de la simulación: endSimulation() ▪ Enviar y recibir warnings en caso que no se cumplan los requisitos de simulación con la batería (ver sistema de seguridad en el punto xx): warningLoad - Llama a la función parseExcel() ubicada en functions.py - Llama a la función delay_seconds() ubicada en functions.py

Tabla 3.13: Clase: loadProfile()

Clase	readVoltAgent()
Hardware	Sistema de medida
Descripción	Este agente se encarga de devolver la lectura de tensión tomada por el sistema de medida
Funcionalidad	- Lectura de tensión - Funcionalidades de carga-descarga - Funcionalidades de simulación de FV - Ensayo de calibración
Atributos	- arduino: Objeto que representa el sistema de medida - fde: Factor de escala del ADC - r1: Valor de la resistencia R_1 del divisor de tensión - r2: Valor de la resistencia R_2 del divisor de tensión - vz: Tensión del diodo zener - tm: Periodo de muestreo - id: Identificador
Métodos	resetSerial(), readVolt(), periodicalInterrupt(), runCD(), runPV(), run()
Asociaciones	- Se comunica con los módulos de carga-descarga y de calibración para devolver la medida de tensión - Si se ejecuta una de las funcionalidades de simulación de FV: ▪ Envía los datos obtenidos mediante un callback a main(), el cual se encarga de realizar una serie de cálculos con ellos y almacenarlos en un archivo $json$ ▪ Los métodos de esta clase se ejecutan al mismo tiempo que los de powerSupplyPV() y loadPV() o powerSupplyProfile() y loadProfile() empleando multithreading ▪ Se comunica con las clases del resto de hilos a través de funciones que usan variables globales definidas en el módulo functions.py para: • Enviar y recibir errores de comunicación del software con los distintos dispositivos: communicationError() • Enviar y recibir una bandera de fin de la simulación: endSimulation()

Tabla 3.14: Clase: readVoltAgent()

Clase	calibrationAgent()
Hardware	Fuente de alimentación
Descripción	Este agente se encarga de calibrar debidamente los diferentes potenciómetros de la fuente para que establezca y mida la tensión correctamente
Funcionalidad	Calibración de la fuente de alimentación (punto 3.2.2)
Atributos	- powerSupply: Objeto que representa la funete de alimentación - pot: Potenciometro que se quiere calibrar - value: Valor de la calibración - password: Contraseña que permite la calibración del dispositivo
Métodos	turnOn(), turnOff(), calibration(), runCD(), run()
Asociaciones	

Tabla 3.15: Clase: calibrationAgent()

Clase	calibrationTestAgent()
Hardware	- Fuente de alimentación - Sistema de medida
Descripción	Este agente se encarga de comprobar si la calibración de la fuente de alimentación ha sido la correcta
Funcionalidad	Ensayo de calibración (punto 3.2.2)
Atributos	- powerSupply: Objeto que representa la carga - readVolt: Objeto que representa el sistema de medida - v_ref: Tensión de referencia (la que se desea establecer) - v_out: Tensión medida por el sistema de medida - v_dis: Tensión medida por la propia máquina (es la misma que se muestra en el display del dispositivo) y la que se desea calibrar - e: Error absoluto entre la tensión del sistema de medida y la del display - e_rel: Error relativo entre la tensión del sistema de medida y la del display - calibration_file: path o nombre del archivo *csv* donde se guardan todos los datos obtenidos
Métodos	turnOn(), turnOff(), queryVOLT(), response(), readVOLT(), calibrationTest(), calculateError(), writeData(), representError(), run()
Asociaciones	- Para leer la tensión del sistema se emplea el objeto readVolt, que es un objeto de la clase readVoltAgent() - Llama a la función curve() y curve2() ubicada en functions.py para representar los resultados obtenidos

Tabla 3.16: Clase: calibrationTestAgent()

Clase	turnOffPSAgent()
Hardware	Fuente de alimentación
Descripción	Este agente se encarga de apagar la fuente de alimentación
Funcionalidad	Apagar la fuente de alimentación (punto 3.2.2)
Atributos	powerSupply: Objeto que representa la fuente
Métodos	remote(), turnOff(), run()
Asociaciones	

Tabla 3.17: Clase: turnOffPSAgent()

Clase	turnOffLoadAgent()
Hardware	Carga
Descripción	Este agente se encarga de apagar la carga
Funcionalidad	Apagar la carga (punto 3.2.2)
Atributos	load: Objeto que representa la carga
Métodos	remote(), turnOff(), run()
Asociaciones	

Tabla 3.18: Clase: turnOffLoadAgent()

Clase	main()
Descripción	Es la clase que se encarga de gestionar todas las funcionalidades de la aplicación, la entrada de datos, el almacenamiento de datos, así como parte de los dos tipos simulación de FV.
Atributos principales	- dischargeTestBattery: Objeto de la clase dischargeTestAgent() - chargeTestBattery: Objeto de la clase chargeTestAgent() - dischargeProfileBattery: Objeto de la clase dischargeProfileAgent() - chargeProfileBattery: Objeto de la clase chargeProfileAgent() - dischargeConstantBattery: Objeto de la clase dischargeConstantAgent() - chargeConstantBattery: Objeto de la clase chargeConstantAgent() - pvPowerSupply: Objeto de la clase powerSupplyPV() - pvLoad: Objeto de la clase loadPV() - profilePowerSupply: Objeto de la clase powerSupplyProfile() - profileLoad: Objeto de la clase loadProfile() - readVolt: Objeto de la clase readVoltAgent() - calibration: Objeto de la clase calibrationAgent() - calibrationTest: Objeto de la clase calibrationTestAgent() - turnOffLoad: Objeto de la clase turnOffLoadAgent() - turnOffPS: Objeto de la clase turnOffPSAgent()
Atributos de la simulación de FV	- curr_ps: Intensidad de la fuente de alimentación - i_ps: Intensidad de referencia de la fuente de alimentación - curr_load: Intensidad de la carga - i_load: Intensidad de referencia de la carga - curr_bat: Intensidad de la batería - volt: Tensión del sistema - soc: Estado de carga de la batería - time: Tiempo de simulación - capacity: Capacidad de la batería - V_LIM_HIGH: Tensión límite máxima de la batería - V_LIM_LOW: Tensión límite mínima de la batería - pvData: Diccionario de listas con los datos obtenidos durante toda la simulación (intensidad de la fuente, de la carga y de la batería; tensión; SOC; tiempo)
Métodos	cdCallback(), pvCallback(), parseConfig(), main()
Asociaciones	- Recibe callbacks de las clases correspondientes a la carga-descarga y a la simulación de FV - Se comunica con las clases del resto de hilos de la simulación de FV a través de funciones que usan variables globales definidas en el módulo functions.py para enviar y recibir warnings en caso que no se cumplan los requisitos de simulación con la batería (ver sistema de seguridad en el punto xx): warningPS() y warningLoad() - Llama a la función calculateSOC() ubicada en functions.py para calcular el SOC de la batería en la simulación de FV - Se comunica con el módulo inputData.py para gestionar la entrada de datos y la configuración de las distintas funcionalidades

Tabla 3.19: Clase: main()

Módulo	curvesCD.py
Descripción	Gestiona las llamadas a las funciones de representación de curvas de ensayos de carga y descarga
Asociaciones	- Recibe como entrada los archivos de salida de alguna de las funcionalidades - Llama a la función de representación de curvas correspondiente ubicada en functions.py

Tabla 3.20: Módulo: curvesCD.py

Módulo	curvesPV.py
Descripción	Gestiona las llamadas a las funciones de representación de curvas de simulación de FV
Asociaciones	- Recibe como entrada los archivos de salida de alguna de las funcionalidades - Llama a la función de representación de curvas correspondiente ubicada en functions.py

Tabla 3.21: Módulo: curvesPV.py

Módulo	inputData.py
Descripción	Gestiona la entrada de datos del usuario. Entrada estandar de Python. Interfaz de consola
Asociaciones	- Recibe como entrada la entrada que da el usuario por teclado - Modifica el fichero de configuración *json* (config.json)

Tabla 3.22: Módulo: inputData.py

Módulo	functions.py
Descripción	Sirve de nexo de comunicación entre diferentes módulos y contiene funciones empleadas en más de un único módulo
Asociaciones	- Recibe desde la clase main() los datos de salida de las diferentes funcionalidades del programa y los almacena en ficheros - Sirve de nexo de las clases powerSupplyPV(), powerSupplyProfile(), loadPV(), loadProfile() o readVoltAgent(), a través de la clase main() para comunicarse entre hilos cuando hay alguna alerta (warnings: punto 3.4.4) en alguno de ellos, cuando hay algún conflicto con la sincronización de los mismos en las funcionalidades de simulación de FV por errores de comunicación o fin de simulación o para enviar y recibir tiempos entre las máquinas - Se comunica con la clase main() para calcular el SOC de la batería en las funcionalidades de simulación de FV - Contiene la función de delay (delay_seconds()) empleada en la mayoría de clases - Contiene las funciones de representación de curvas que se invocan desde los módulos curvesCD.py, curvesPV.py y la clase calibrationTestAgent() - Contiene funciones para parsear los perfiles de definidos en todas las clases que trabajan con perfiles

Tabla 3.23: Módulo: functions.py

3.3. Diseño del sistema de medida

3.3.1. Diseño del hardware

En primer lugar, el sistema de medida esta formado por un divisor de tensión resistivo compuesto por dos resistencias R_1 y R_2 tal y como puede observarse en la figura 3.7, el cual se comporta según la ecuación 3.19.

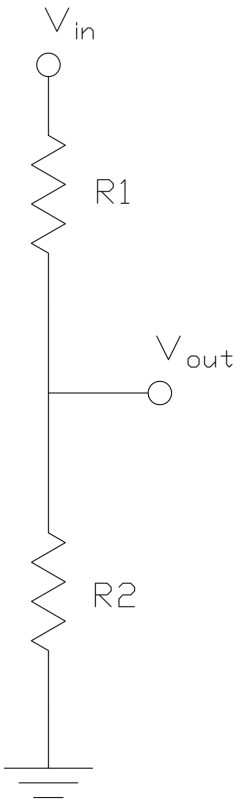


Figura 3.7: Esquema de un divisor de tensión resistivo

$$V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{in} \tag{3.19}$$

La entrada del divisor va conectada a la salida del sistema fuente-carga-batería, de forma, que V_{in} es la tensión de la batería. La salida del divisor va conectada a la primera entrada analógica del convertidor ($A0$), de modo que, V_{out} es la señal de entrada del pin $A0$ del ADC.

En segundo lugar, se tiene un regulador de tensión zener compuesto por una resistencia (R_Z) y un diodo zener, tal y como puede observarse en la figura 3.8, donde V_Z es la tensión que fija el regulador.

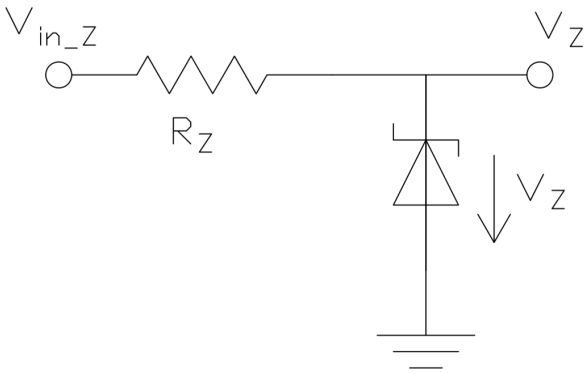


Figura 3.8: Esquema de un regulador de tensión zener

La entrada del regulador va conectada al pin de de alimentación de 5 V del Arduino, de forma, que V_{in_Z} son 5 V. La salida del regulador va conectada a la segunda entrada analógica del convertidor ($A1$), de modo que, V_Z es la señal de entrada del pin $A1$ del ADC.

En tercer lugar, se tiene el ADC ADS1115 (figura 3.9), el cual recibe las señales analógicas del divisor de tensión y del regulador zener en las entradas A0 y A1, respectivamente, las cuales se restan al programarlo en modo diferencial. El convertidor se alimenta a 5 V por el Arduino (se conecta el pin de alimentación del ADC, + o VDD, al pin de 5 V del Arduino y el pin - o GND al pin de tierra del Arduino). Para la comunicación de los datos se conecta los pines CLOCK y DATA del ADC a los respectivos pines del Arduino (ver detalle del pinout del ADC en el Anexo 14.2).

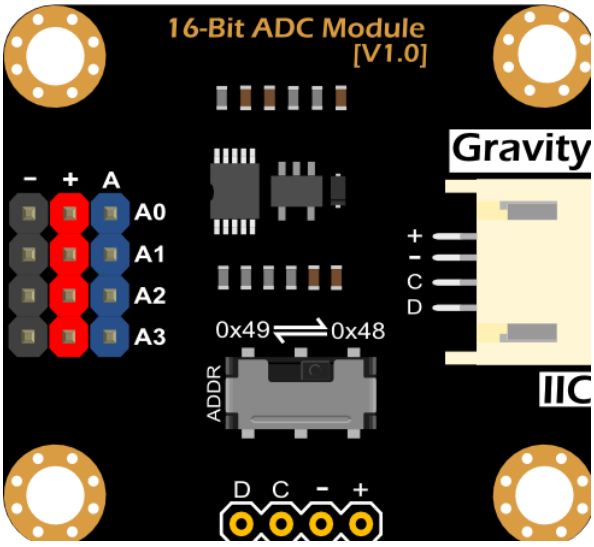


Figura 3.9: ADC ADS1115

Por último, se tiene la placa Arduino UNO R3 (figura 3.10), el cual recibe la información de la medida de tensión del ADC a través de los buses CLOCK y DATA. Además, alimenta con 5 V, tanto al regulador zener, como al convertidor; y el resto de componentes están conectados a su pin de tierra (GND). Además, la placa está alimentada y se comunica con la computadora mediante conexión USB (ver detalle del pinout del Arduino en el Anexo 14.1).

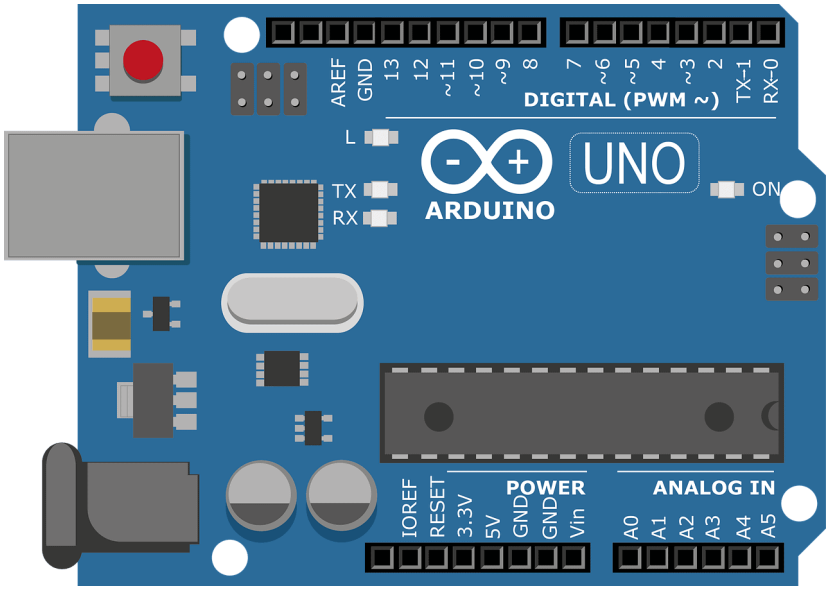


Figura 3.10: Arduino UNO R3

A continuación, puede observarse el esquema completo del circuito del sistema de medida de tensión (figura 3.11), tal y como se ha descrito anteriormente.

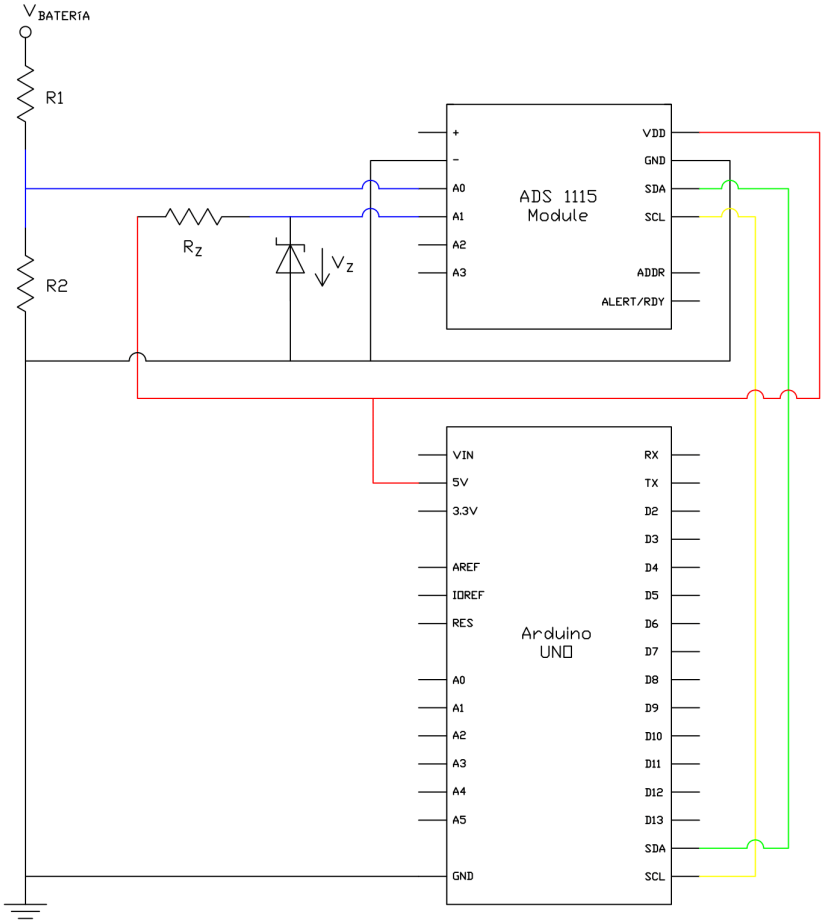


Figura 3.11: Circuito del sistema de medida

Donde, las **conexiones azules** se corresponden con las señales de entrada en los pines A0 y A1 del convertidor; la **conexión amarilla** es CLOCK; la **conexión verde** es DATA; las **conexiones rojas** son la alimentación a 5 V; y las **conexiones negras** son GND.

Según la batería que se emplee dentro de la plataforma, el diseño del divisor de tensión, del regulador zener y del software será diferente.

Seguidamente, se realiza el diseño tanto del divisor, como del regulador para los dos tipos de batería que se han estudiado en este proyecto.

Sistema de medida para la batería de la moto

Por un lado, se ha calculado un divisor de tensión a cuya salida hayan como máximo 5 V y cuya impedancia sea elevada para que la intensidad que circule por el convertidor no provoque ningún daño. De modo que atendiendo a la ecuación 3.19, se han obtenido los siguientes valores de las resistencias.

$$R_1 = 472k\Omega, R_2 = 32.7k\Omega \tag{3.20}$$

De esta forma, para el máximo valor de tensión de la batería se tiene una señal a la salida del divisor de $V_{out\_max} = 4.6$ V; y para un el mínimo valor de tensión de la batería se tiene una señal a la salida del divisor de $V_{out\_min} = 3.69$ V, como se puede comprobar en las ecuaciones 3.21 y 3.22.

$$V_{out\_max} = \frac{32.7}{32.7 + 472} \cdot 71 = 4.6V \tag{3.21}$$

$$V_{out\_min} = \frac{32.7}{32.7 + 472} \cdot 57 = 3.69V \tag{3.22}$$

Por otro lado, se ha elegido un diodo zener que fije la tensión V_Z en 4.32 V, para que la diferencia máxima y mínima entre las señales del divisor y del zener sean lo más cercana a 0, lo que permite escoger la referencia de tensión del PGA más baja posible y obtener el factor de escala óptimo. Para conseguir que el diodo zener fije la tensión en este valor, se ha escogido una resistencia $R_Z = 130\Omega$, a partir de la ficha técnica del diodo y, además, se ha comprobado experimentalmente. De modo que, así quedarían los valores del regulador zener:

$$V_Z = 4.32V, R_Z = 130\Omega \tag{3.23}$$

Esto permite obtener los siguientes valores de tensión máxima y mínima (ecuación 3.24 y 3.25, respectivamente) cuando el convertidor realiza la diferencia entre las dos entradas analógicas.

$$V_{dif\_max} = 4.6 - 4.32 = 0.28V \tag{3.24}$$

$$V_{dif\_min} = 3.69 - 4.32 = -0.63V \tag{3.25}$$

Por tanto, en este caso se elige el PGA 4 que establece la referencia de voltaje del convertidor en 1.024V, ya que es la mínima que se puede escoger para los 0.63 V máximos que necesita leer la entrada analógica. Con este PGA, se obtiene un factor de escala de 0.0312 mV, lo que permite alcanzar la precisión que se observa en la ecuación 3.32.

$$FE = 0.0312mV \tag{3.26}$$

$$V_{ref} + V_Z = 1.024 + 4.32 = 5.344V \rightarrow \tag{3.27}$$

$$5.344 = V_{medida\_max} \cdot \frac{32.7}{32.7 + 472} \rightarrow V_{medida\_max} = 82.4806V \tag{3.28}$$

$$0 + V_Z = 0 + 4.32 = 4.32V \rightarrow \tag{3.29}$$

$$4.32 = V_{medida\_min} \cdot \frac{32.7}{32.7 + 472} \rightarrow V_{medida\_min} = 66.676V \tag{3.30}$$

$$precisión = \frac{V_{medida\_max} - V_{medida\_min}}{2^{15}} \rightarrow \tag{3.31}$$

$$precisión = \frac{82.4806 - 66.676}{2^{15}} = \frac{15.8046}{2^{15}} = 0.0004823V = 0.4823mV \tag{3.32}$$

Donde, V_{medida_max} y V_{medida_min} son las tensiones máximas y mínimas que teóricamente se pueden leer con el zener y el PGA escogido con los 15 bits reservados para el valor digital de la tensión.

Esto supone una precisión de 143 veces más elevada que si solo se utiliza el Arduino para medir la tensión de la batería de la moto.

Sistema de medida para la celda Panasonic PA-L154.K01

En primer lugar, se ha prescindido del divisor de tensión, ya que la tensión máxima que entrega la celda es de 4.2 V.

De forma que, para el máximo valor de tensión de la batería se tiene una señal $V_{out\_max} =$ 4.2 V; y para un el mínimo valor de tensión de la batería se tiene una señal de $V_{out\_min} =$ 3 V.

Así que, se ha elegido un diodo zener que fije la tensión V_Z en 3.54 V, para que la diferencia máxima y mínima entre las señales de la celda y del zener sean lo más cercana a 0, lo que permite escoger la referencia de tensión del PGA más baja posible y obtener el factor de escala óptimo. Para conseguir que el diodo zener fije la tensión en este valor, se ha elegido una resistencia $R_Z = 56\Omega$, a partir de la ficha técnica del diodo y, además, se ha comprobado experimentalmente. De modo que, así quedarían los valores del regulador zener:

$$V_Z = 3.54V, R_Z = 56\Omega \tag{3.33}$$

Lo que permite obtener los siguientes valores de tensión máximo y mínimo (ecuación 3.34 y 3.35) cuando el convertidor realiza la diferencia entre las dos entradas analógicas.

$$V_{dif\_max} = 4.2 - 3.54 = 0.66V \tag{3.34}$$

$$V_{dif\_min} = 3 - 3.54 = -0.54V \tag{3.35}$$

Por tanto, en este caso se elige el PGA 4 que establece la referencia de voltaje del convertidor en 1.024V, ya que es la mínima que se puede escoger para los 0.66 V máximos que necesita leer la entrada analógica. Con este PGA, se obtiene un factor de escala de 0.0312 mV, lo que permite alcanzar una precisión que coincide con este factor, ya que no se tiene ningún divisor, como se observa en la ecuación 3.40.

$$FE = 0.0312mV \tag{3.36}$$

$$V_{medida\_max} = V_{ref} + V_Z = 1.024 + 3.54 = 4.564V \rightarrow \tag{3.37}$$

$$V_{medida\_min} = 0 + V_Z = 0 + 3.54 = 3.54V \rightarrow \tag{3.38}$$

$$precisión = \frac{V_{medida\_max} - V_{medida\_min}}{2^{15}} \rightarrow \tag{3.39}$$

$$precisión = \frac{4.564 - 3.54}{2^{15}} = \frac{1.024}{2^{15}} = 0.0000312V = 0.0312mV \tag{3.40}$$

Donde, V_{medida_max} y V_{medida_min} son las tensione máximas y mínimas que teóricamente se pueden leer con el zener y el PGA escogido con los 15 bits reservados para el valor digital de la tensión.

Esto supone una precisión de 156.5 veces más elevada que si solo se utilizará el Arduino para medir la tensión de la celda.

3.3.2. Diseño del software

Como ya se ha mencionado anteriormente en el punto 1.4.6, para el diseño del software del sistema de medida se ha utilizado el entorno de desarrollo propio de Arduino (Arduino IDE), el cual emplea C como lenguaje de programación.

En primer lugar, se han incluido las librerías para establecer la comunicación entre todos los dispositivos (extracto de código 3.1). La librería de comunicación modbus que permite la comunicación entre la tarjeta Arduino y el programa principal de la herramienta que se está ejecutando en la computadora. También, la propia librería del convertidor ADC, que además de permitir la comunicación I2C entre Arduino y ADC, también implementa el método necesario para usar el modo diferencial del convertidor.

```
#include <SimpleModbusSlave.h>
#include <Adafruit_ADS1X15.h>
```

Extracto de código 3.1: Librerías

A continuación, en el extracto de código 3.2 se declaran todas las variables, se configura el PGA correspondiente, se inicializa el convertidor y se configura la comunicación modbus tal y como se indica en el punto 3.2.1.

```
enum
{
  VOLT_ADDRESS,
  HOLDING_REGS_SIZE
};
unsigned int holdingRegs[HOLDING_REGS_SIZE];
short result;

void setup(void)
{
  Serial.begin(9600);
  ads.setGain(GAIN_FOUR);
  if (!ads.begin())
  {
    Serial.println("Failed to initialize ADS.");
    while (1);
  }
  modbus_configure(&Serial, 9600, SERIAL_8E1, 1, 13,
     HOLDING_REGS_SIZE, holdingRegs);
  modbus_update_comms(9600, SERIAL_8E1, 1);
```

```
}
```

Extracto de código 3.2: Declaración de variables y setup()

Por último, dentro de la función principal *loop*() (extracto de código 3.3), se actualiza constantemente la comunicación modbus, se guarda el valor digital leído por el convertidor en modo diferencial usando el método $ads.readADC\_Differential\_0\_1()$ en una variable y, finalmente, esta información se envía a la computadora vía modbus.

```
void loop(void)
{
  modbus_update();
  result = ads.readADC_Differential_0_1();
  holdingRegs[VOLT_ADDRESS] = result;
}
```

Extracto de código 3.3: loop()

3.4. Diseño del software e implementación

3.4.1. Diseño de la arquitectura del software

El software tiene implementado el programa principal en el módulo *main.py*, el cual contiene la clase principal (*main()*), encargada de gestionar el resto del software. De este modo, este módulo inicia la ejecución del programa instanciando a la clase *main()*, la cual, en el método *__init__()* tiene definidos una serie de atributos (detallados en el cuadro 3.19) y además llama a al método principal de la clase, también llamado *main()*. Este método, es el encargado manejar de los siguientes puntos.

- **Interfaz de usuario: Entrada/Salida Estándar**
- **Gestión de los Agentes**
- **Almacenamiento de datos**

Interfaz de usuario: Entrada/Salida Estándar

La forma de comunicación entre la plataforma y el usuario se realiza mediante la entrada/salida estándar de Python, es decir, se emplea una interfaz de consola. Por tanto, la entrada siempre se realiza a través del teclado escribiendo en la ventana de la consola y, de la misma manera, la información de salida se visualiza por pantalla, a través de la consola.

En primer lugar, en el método *main()* está implementado un menú con todas las funcionalidades que tiene la plataforma. Por tanto, es lo primero que se observa en la consola (figura 3.12) al ejecutar la aplicación. A continuación, se le pide al usuario que introduzca la tarea que quiere realizar de las que se ofrecen en el menú. Una vez, el usuario ha introducido este dato, desde *main()* se llama a la función *inputData()* ubicada en el módulo *inputData.py*, a la cual se le pasan como parámetros, la opción escogida por el usuario y el contenido del archivo de configuración.

El fichero de configuración (*config.json*), es un archivo *json* que contiene un diccionario de diccionarios, es decir, la clave de los elementos del diccionario del nivel más externo se corresponde con un string que identifica el componente o la funcionalidad, y el valor se corresponde con otro diccionario. El diccionario del nivel más interno, a su vez, contiene los elementos a configurar (ver figura 3.13).

Este fichero viene con una configuración por defecto. De esta forma, *inputData()*, a partir de la funcionalidad elegida por el usuario, extrae su configuración por defecto y la muestra por pantalla. A continuación, el usuario decide si quiere seguir con los parámetros que están por defecto o quiere cambiarlos. En caso de decidir cambiarlos, el usuario va introduciendo los valores uno por uno hasta que se terminan (figura 3.14). Finalmente, se le pregunta al usuario si quiere guardar estos parámetros, como configuración por defecto; en caso afirmativo, se modifica el archivo de configuración de forma permanente con los nuevos valores; en caso contrario, se cambian los parámetros por los nuevos en la variable que contiene el diccionario del archivo pero no en el mismo archivo que sigue conservando los valores originales.

```
______MENÚ______

1. Ensayo de descarga
2. Ensayo de carga
3. Descargar siguiendo un perfil
4. Cargar siguiendo un perfil
5. Descarga constante
6. Carga constante
7. Simular un sistema de FV con curvas
8. Simular un sistema de FV con perfiles
9. Lectura de tensión
10. Calibración de la fuente
11. Ensayo de calibración
12. Apagar carga
13. Apagar fuente
14. Apagar todo el sistema
15. Cambiar configuaración de la bateria
16. Cambiar configuaración del sistema de medida de tensión
17. Salir

Selecciona una opción:
```

Figura 3.12: Menú

```
{
    "serial":
    {
        "serialPort": "COM1",
        "serialBaud": 9600,
        "serialTimeout": 1,
        "comandDelay": 1,
        "identifier": "L"
    },

    "socket":
    {
        "IP": "150.128.87.59",
        "port": 50505,
        "comandDelay": 0.5,
        "identifier": "PS"
    },

    "arduino":
    {
        "serialPort": "COM6",
        "serialBaud": 9600,
        "serialTimeout": 1,
        "comandDelay": 1,
        "identifier": "V"
    },
```

Figura 3.13: Archivo de configuración

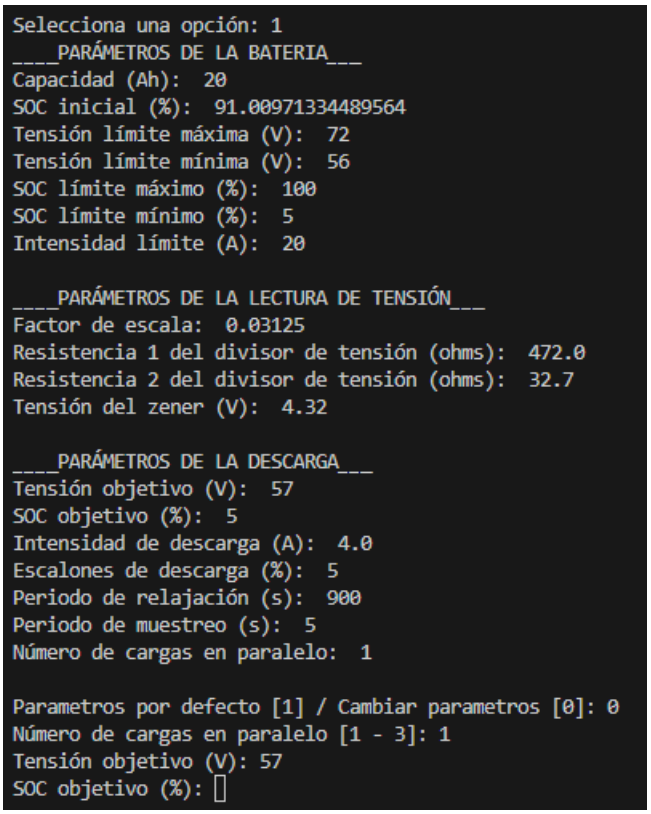


Figura 3.14: Configuración del sistema

Una vez ya está en ejecución alguna de las funcionalidades de la herramienta se puede visualizar la información de salida en tiempo real por pantalla. Por último, cuando termina la ejecución de la funcionalidad se muestran los resultados completos de forma gráfica por pantalla.

Gestión de los Agentes

El método *main()*, de acuerdo con la funcionalidad seleccionada, llama al método *parseConfig()*, el cual se encarga de parsear correctamente la configuración de la opción determinada y de crear un objeto del Agente que gestiona esa funcionalidad.

Seguidamente, desde *main()* se llama al método del objeto en cuestión que inicia la ejecución de la funcionalidad (*run()*). Una vez termina el ciclo de ejecución de la funcionalidad, nuevamente, desde *main()* se llama a la función encargada de la representación de los resultados obtenidos. Se trata de la función *run()*, ubicada en el módulo *curvesCD.py* o *curvesPV.py*, dependiendo de la funcionalidad elegida, a la cual se le pasa como parámetro el nombre del archivo donde se han almacenado los resultados o, en su defecto, el path.

Como caso especial, se debe mencionar las dos opciones de simulación de un sistema fotovoltaico con almacenamiento. En ambos casos, se hace uso de multithreading para poder emplear todos los dispositivos al mismo tiempo. Concretamente, se crean tres hilos, uno para la fuente que ejecuta el Agente de simulación de la fuente; otro para la carga que ejecuta el Agente de simulación de la carga; y otro para el sistema de medida

que ejecuta el Agente de simulación del sistema de medida.

Almacenamiento de datos

Este software gestiona el almacenamiento de datos principalmente a través de callbacks, es decir, la clase *main()* implementa el método *cdCallback()* para las funcionalidades de carga/descarga y *pvCallback* para las simulaciones de FV. Estos métodos se pasan como argumento en el momento de crear el objeto del Agente correspondiente, de modo que, durante la ejecución del Agente cuando se obtiene un dato que se desea almacenar, el Agente llama al callback pasándole como argumento el dato. De esta forma, el dato se recibe en la clase principal que lse encarga de almacenarlos.

En el caso de funcionalidades de carga/descarga, la información se guarda en un fichero *csv*, de lo cual se encarga la función *writeDataCD()* ubicada en *functions.py*. En el caso de las simulaciones de FV, es diferente debido a que se trabaja con tres hilos al mismo tiempo.

En primer lugar se recibe la información de cada uno de los hilos en el callback, para, a continuación, cruzar todos los datos y realizar los cálculos necesarios (corriente y SOC de la batería). Una vez se han obtenido todos los datos deseados, estos se almacenan directamente desde el callback en archivo *json*.

Cuando, finalmente, se ha terminado el proceso de simulación, el método principal se encarga de llamar a la función *writeDataPV()* ubicada en *functions.py* para almacenar todos los resultados en un archivo *csv*.

Por último, en el caso de el ensayo de calibración de la fuente, el almacenamiento de datos lo realiza directamente su Agente, escribiendo la información en fichero *csv*.

3.4.2. Ciclo de ejecución

A continuación, se muestra el ciclo de ejecución general del programa en pseudocódigo, que permite ver de forma global como funciona el programa:

Entrada: Funcionalidad seleccionada por el usuario, Configuración de la funcionalidad
Salida : Ficheros con los resultados obtenidos y representaciones gráficas de los mismos

```
// Programa principal
```
while *no salir del programa* **do**
    Mostrar menú al usuario y solicitar la opción deseada;
    Leer la opción seleccionada por el usuario;
    Solicitar la configuración de la funcionalidad al usuario;
    Guardar la configuración de la funcionalidad;
    if *la opción seleccionada es válida* **then**
        Crear un `Agente` correspondiente a la funcionalidad seleccionada;
        Ejecutar el `Agente`;
        while *el* `Agente` *esté en ejecución* **do**
            Obtener los resultados del `Agente` mediante callbacks;
            Escribir los resultados en ficheros;
        end
        Representar los resultados obtenidos en gráficas;
    end
end

3.4.3. Código

En el siguiente enlace se da acceso a un repositorio de GitHub donde se puede visualizar y descargar de forma libre todo el código fuente de la herramienta:

https://github.com/JavierGonzalezBarredaUJI/TFG_JavierGonzalezBarreda.git

3.4.4. Control de errores

El control de errores desempeña un papel fundamental para garantizar que las aplicaciones sean confiables, robustas y capaces de manejar situaciones inesperadas. Los errores son inevitables en cualquier sistema complejo, y su manejo adecuado se vuelve decisivo para minimizar su impacto negativo en la experiencia del usuario y en el correcto funcionamiento de la herramienta.

Este apartado, se centra en el control de errores implementado en esta herramienta, donde se exploran las estrategias y técnicas utilizadas para detectar, informar y manejar los errores que puedan surgir durante la ejecución del programa, así como las mejores prácticas para garantizar una experiencia de usuario fluida y segura.

El control de errores en esta herramienta no solo tiene como objetivo detectar y manejar los errores en tiempo de ejecución, sino también prevenirlos en la medida de lo posible durante la etapa de desarrollo. Al implementar estrategias sólidas de control de errores, se busca minimizar interrupciones inesperadas, mejorar la estabilidad y confiabilidad de la herramienta, y proporcionar una retroalimentación clara y precisa al usuario en caso de que se produzca algún error.

De esta forma, lo que se pretende es explorar en detalle las diversas técnicas y los enfoques utilizados para controlar los errores en el software de la herramienta. Para lo cual se abordan aspectos como la detección temprana de errores, el manejo adecuado de excepciones y la comunicación efectiva de los errores al usuario.

Principalmente, se pueden dar errores en tres situaciones diferentes: en la entrada de datos, en la comunicación entre dispositivos y en el cumplimiento de requisitos y restricciones.

En la entrada de datos

Los errores de entrada pueden variar desde simples equivocaciones tipográficas hasta datos incorrectos o inconsistentes que pueden comprometer la integridad hardware y la funcionalidad del sistema. Por lo tanto, se han implementado mecanismos de control de errores efectivos para validar y gestionar la entrada de datos proporcionada por el usuario mediante:

- **La validación de datos**, es decir, realizar una serie de verificaciones y comprobaciones sobre los datos ingresados, asegurando que cumplan con los requisitos y restricciones definidos por los componentes de la herramienta y con los definidos en la etapa de diseño de la misma. Los aspectos clave de la validación de datos incluyen:
  - **Tipos y formatos de datos**: Verificar que los datos ingresados por el usuario cumplan con el tipo y formato esperados. Esto implica comprobar si se trata de números, cadenas de texto u otros tipos de datos específicos, y validar que se ajusten a la estructura adecuada que espera el programa.
  - **Rangos y límites**: Validar que los valores ingresados se encuentren dentro de los rangos y límites permitidos.
  - **Coherencia y consistencia**: Verificar la coherencia y consistencia de los datos ingresados en relación con otros datos o reglas definidas por la herramienta.
- **La retroalimentación al usuario**. Además de la validación de datos, se proporciona una retroalimentación clara y útil al usuario cuando se producen errores en la entrada de datos, informando de manera comprensible qué ha causado el error y ofreciendo sugerencias o indicaciones sobre cómo actuar para corregirlo. Además, si los errores están relacionados con el formato o los requisitos de entrada, se proporcionan indicaciones específicas sobre cómo se deben ingresar los datos correctamente.

En la comunicación entre dispositivos

En este proyecto la comunicación entre dispositivos es fundamental, por tanto, es decisivo abordar los posibles errores que pueden surgir en esta etapa. La comunicación entre dispositivos involucra la transmisión de datos, comandos e información crítica entre los diferentes sistemas o componentes, que pueden tener un impacto significativo en el rendimiento, la confiabilidad y la integridad de la herramienta. A continuación,

se exploran los aspectos clave relacionados con los errores en la comunicación entre dispositivos.

La validación de datos y la verificación de los mensajes es fundamental para garantizar la integridad de los datos transmitidos y asegurar que lleguen sin alteraciones. Para ello, se emplean técnicas de reintentos y mecanismos de recuperación, es decir, ante posibles errores de comunicación, como paquetes perdidos o conexiones interrumpidas, se implementan mecanismos de reintentos y recuperación para asegurar que los datos se transmitan de manera confiable. Esto puede incluir el uso de confirmaciones de recepción, retransmisiones automáticas y estrategias de recuperación ante fallos, mediante el manejo de excepciones.

Por tanto, en la gestión de errores de comunicación es fundamental implementar mecanismos adecuados para gestionar y manejar esos errores de manera efectiva. Las prácticas que se incluyen son:

- **Mensajes de error claros**: Proporcionar mensajes de error descriptivos y comprensibles para informar a los usuarios y desarrolladores sobre la naturaleza y causa del error. Esto ayudará a identificar rápidamente los problemas y a tomar las acciones correctivas necesarias.
- **Reintentos y recuperación**: Implementar mecanismos de reintentos automáticos y recuperación ante fallos de comunicación. Esto permite volver a intentar la transmisión de datos o restablecer la conexión para minimizar las interrupciones y asegurar que la comunicación se reanude de manera confiable, todo ello evitando la pérdida de información.

En el cumplimiento de requisitos y restricciones

Los dispositivos empleados en este proyecto como ya se ha comentado anteriormente en el punto 3.2.1 tienen una serie de limitaciones físicas que han de ser cumplidas estrictamente si no se quiere que las consecuencias sean catastróficas. Concretamente, se han de tener en cuenta los límites de tensión y de intensidad de la fuente de alimentación y de la carga, y, sobretodo, los límites de tensión de la batería que al ser los menores de los tres dispositivos, son los que marcan el funcionamiento general de todo el sistema.

Por tanto, para impedir que errores de esta índole se den en este sistema se realizan pruebas y validaciones continuas. Es decir, se trata de realizar pruebas y validaciones continuamente a lo largo de la ejecución del programa para identificar cuándo en alguno de los parámetros mencionados anteriormente llega a un umbral de seguridad y, entonces, efectuar las acciones correspondientes en cada caso.

Por ejemplo, en los ensayos de carga y descarga se está constantemente comprobando que el nivel de tensión de la batería no supera el umbral de seguridad a partir del cual se podría dañar el propio dispositivo. Otro caso más complejo que merece la pena mencionar y que ya se ha explicado en el punto 3.2.2 se da cuando se está ejecutando simulaciones de FV. En este caso, se podría dar el escenario donde en un determinado instante la intensidad de la fuente sea de un determinado valor mayor que el de la intensidad de la carga y, además, la batería esté llena. O el caso contrario, donde la intensidad de la fuente sea de un determinado valor menor que el de la intensidad de la carga y, además, la batería esté vacía. Por tanto, lo que sucedería es que la

batería intentaría absorber ese exceso de intensidad estando llena (primer caso), o intentaría aportar esa falta de intensidad estando vacía (segundo caso) por lo que acabaría dañándose.

Para evitar esto se ha implementado una estrategia en la cual se monitoriza continuamente las intensidades de la fuente y de la carga y cuando se detecta que el sistema esta en alguna de estas situaciones manda una alerta al resto de hilos que están ejecutando la simulación para modificar el valor de estas intensidades. En el primer caso, la intensidad de la fuente se iguala al valor de la intensidad de la carga, y en el segundo caso, la intensidad de la carga se iguala al valor de la intensidad de la fuente. De está forma, la corriente que fluye por la batería es cero y ya no existe ningún peligro de dañar el dispositivo.

4. Pruebas y resultados

En esta sección se abordarán las pruebas y los resultados obtenidos durante el desarrollo del proyecto. Aquí se presentará un análisis exhaustivo de las pruebas realizadas para evaluar la funcionalidad, rendimiento y fiabilidad del sistema implementado.

El objetivo principal de esta etapa es verificar que el software cumple con los requisitos establecidos previamente y se encuentra en condiciones óptimas para su despliegue y uso en un entorno real. Además, se busca identificar posibles fallos, errores o limitaciones, a fin de realizar las correcciones necesarias y garantizar la calidad del producto final.

En esta parte del trabajo se describirán detalladamente las estrategias de prueba utilizadas, incluyendo los casos de prueba diseñados, las herramientas empleadas y los criterios de aceptación definidos.

Asimismo, se presentarán los resultados obtenidos de las pruebas realizadas, junto con un análisis crítico de los mismos. Se mostrará cómo se evaluó el cumplimiento de los requisitos establecidos, así como los criterios de éxito definidos para cada prueba. Además, se destacarán los aspectos positivos del sistema, como su desempeño eficiente, la detección temprana de posibles problemas y la robustez alcanzada, así como las áreas de mejora identificadas y las acciones tomadas para abordarlas.

En resumen, en esta sección se proporcionará una visión general de las pruebas llevadas a cabo y los resultados obtenidos, lo que permitirá evaluar la calidad del software desarrollado y tomar decisiones fundamentadas para su posterior implementación y mejora.

4.1. Ensayo de descarga con la batería

Esta prueba pretende mostrar el correcto funcionamiento del ensayo de descarga con la batería de la moto conforme a las especificaciones definidas anteriormente. En este caso, se han fijado los siguientes parámetros de entrada:

- Tensión objetivo (V): 57
- SOC objetivo (%): 10
- Intensidad de descarga (A): 4
- Escalones de descarga (%): 5
- Periodo de relajación (s): 7200
- Periodo de muestreo (s): 5
- Número de cargas en paralelo: 1

4.1.1. Resultados

A continuación, se muestran las gráficas obtenidas del ensayo de descarga:

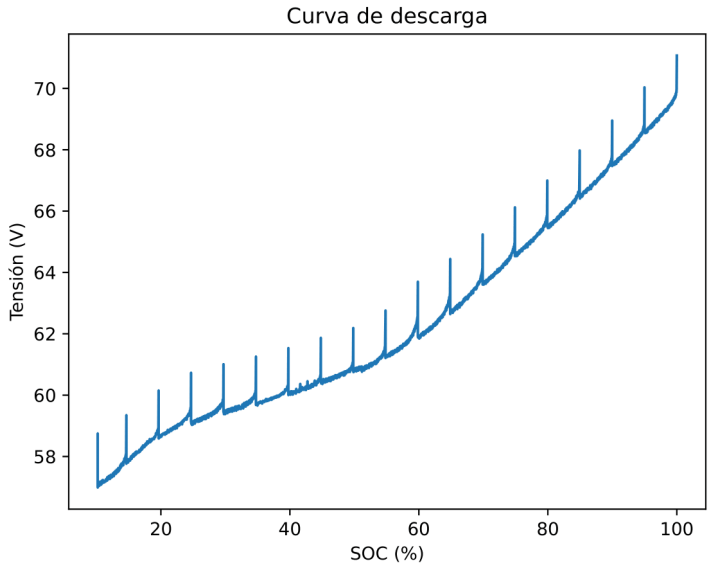


Figura 4.1: Curva de descarga de la batería ($V - SOC$)

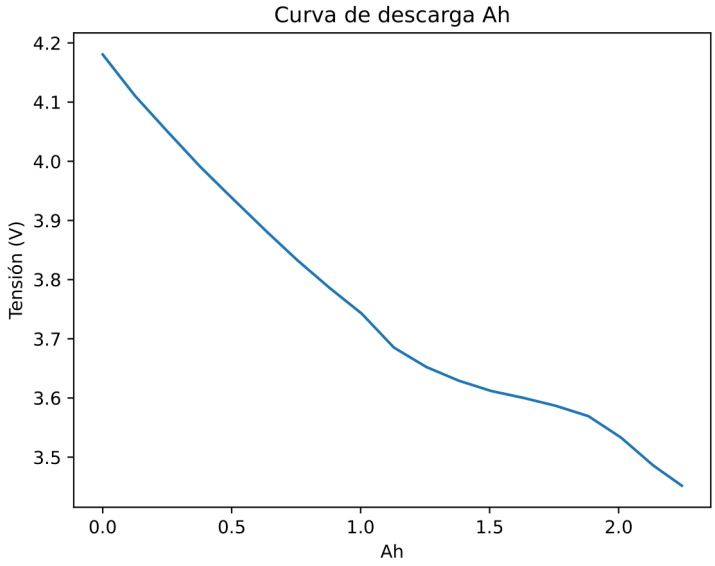


Figura 4.2: Curva de descarga de una celda de la batería ($V - Ah$)

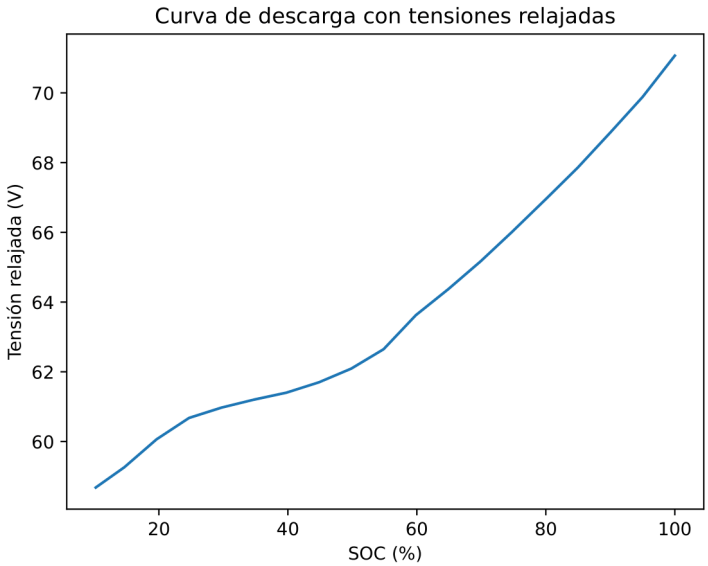


Figura 4.3: Curva de descarga de la batería ($V_{relajada} - SOC$)

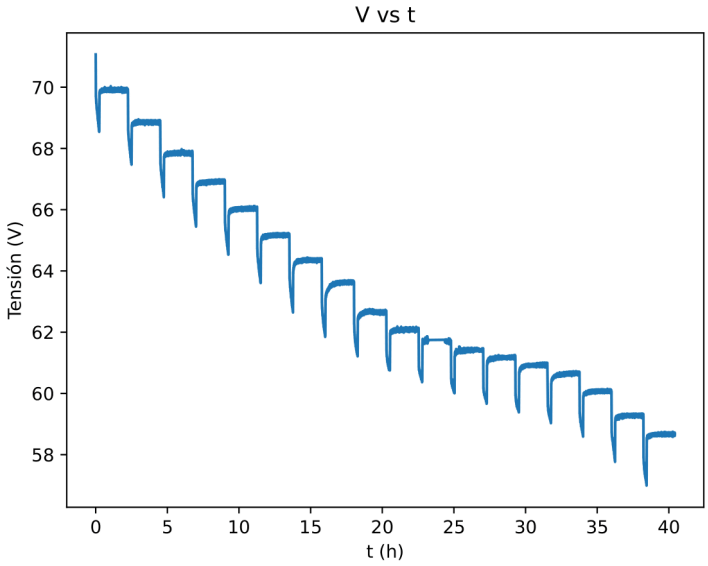


Figura 4.4: Curva de descarga de la batería ($V - t$)

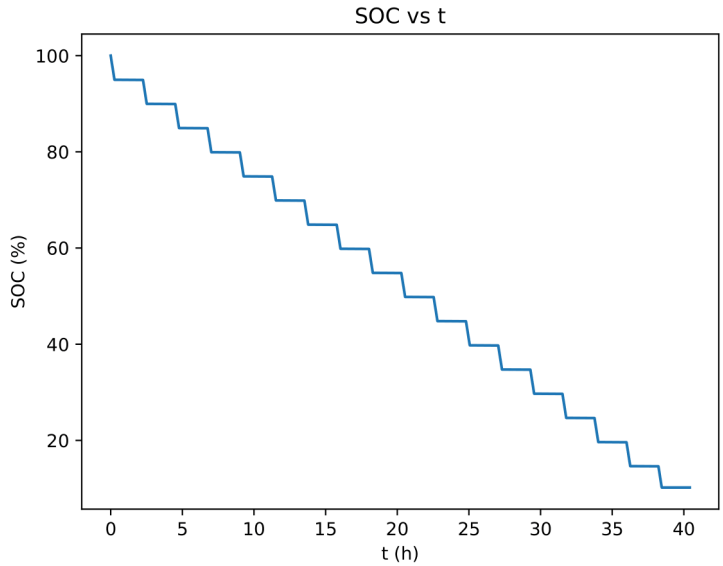


Figura 4.5: Curva de descarga de la batería ($SOC - t$)

4.1.2. Análisis y conclusiones

En primer lugar, como se puede observar en la figura 4.1, la herramienta realiza el ensayo correctamente, tal y como se define en el punto 3.2.2.

Además, cumple con la condición de parada ya que se detiene al llegar a una tensión de 56.988 V. También, se observa que los escalones de descarga son del 5 %.

Por otro lado, en las figuras 4.4 y 4.5 el tiempo de relajación es de 2 horas, tal y como se ha definido.

Igualmente, se ha aprovechado el ensayo para realizar múltiples test de errores que cubran todos los posibles fallos descritos en la sección 3.4.4 teniendo en cuenta las especificaciones y limitaciones concretas de esta funcionalidad, y se puede afirmar que la herramienta maneja todos los tipos de errores ampliamente.

Por tanto, se concluye que la aplicación responde de forma fiable y segura a esta funcionalidad, atendiendo a todas las condiciones y limitaciones definidas en el proceso de desarrollo y a los requisitos introducidos por el usuario, y cumpliendo con su objetivo principal que es mostrar el comportamiento de una batería durante un determinado ensayo de descarga.

Finalmente, si se compara la figura 4.2, que muestra la curva de descarga de una celda de la batería de estudio, con la figura 4.6, que muestra la curva de descarga típica de una celda Samsung INR18650-25R (Green), se observa que las curvas son muy similares. Hay que tener en cuenta que la curva de la figura 4.2 está incompleta, ya que no se ha llegado a descargar totalmente, por lo que solo habría que observar hasta valores de tensión de entre 4.2 V y 3.5 V para realizar la comparación. Por tanto, se puede establecer la hipótesis que la batería de estudio esta formada por celdas de este tipo, sin embrago, esto no supone una prueba definitiva que lo confirme.

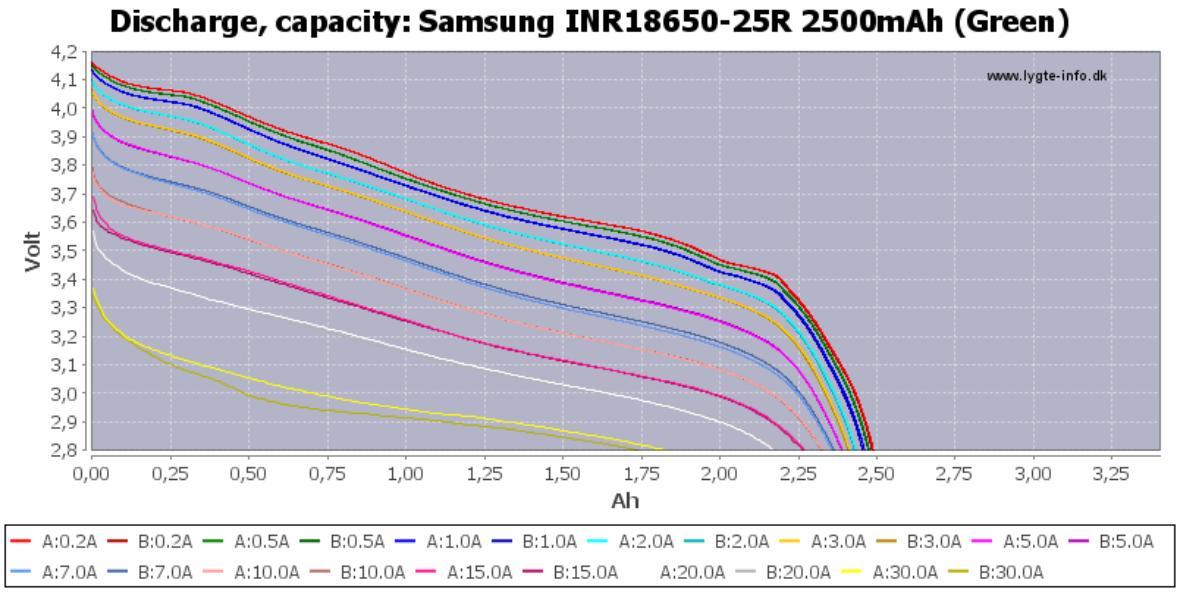


Figura 4.6: Curva de descarga de una celda Samsung INR18650-25R (Green)

4.2. Ensayo de carga con la batería

Esta prueba pretende mostrar el correcto funcionamiento del ensayo de carga con la batería de la moto conforme a las especificaciones definidas anteriormente. En este caso, se han fijado los siguientes parámetros de entrada:

- Tensión límite de carga a corriente constante (V): 71
- Intensidad umbral de fin de carga (A): 0.4
- Intensidad de carga (A): 4
- Escalones de carga (%): 5
- Periodo de relajación (s): 7200
- Periodo de muestreo (s): 5

4.2.1. Resultados

A continuación, se muestran las gráficas obtenidas del ensayo de carga:

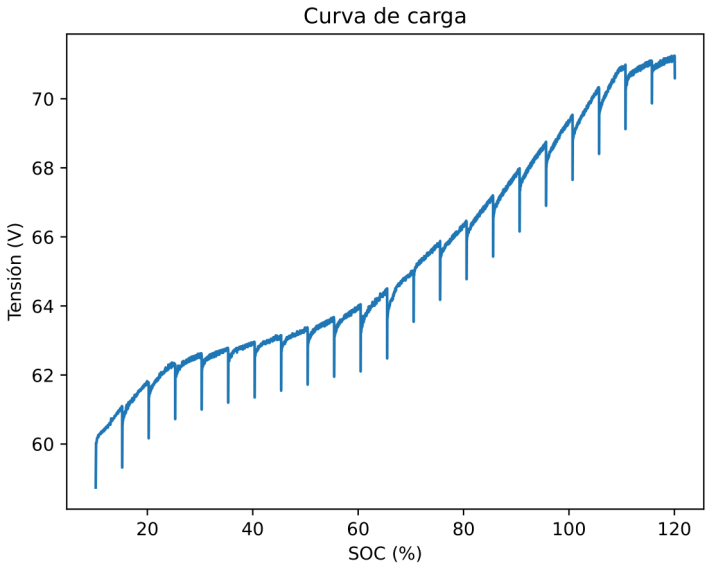


Figura 4.7: Curva de carga de la batería ($V - SOC$)

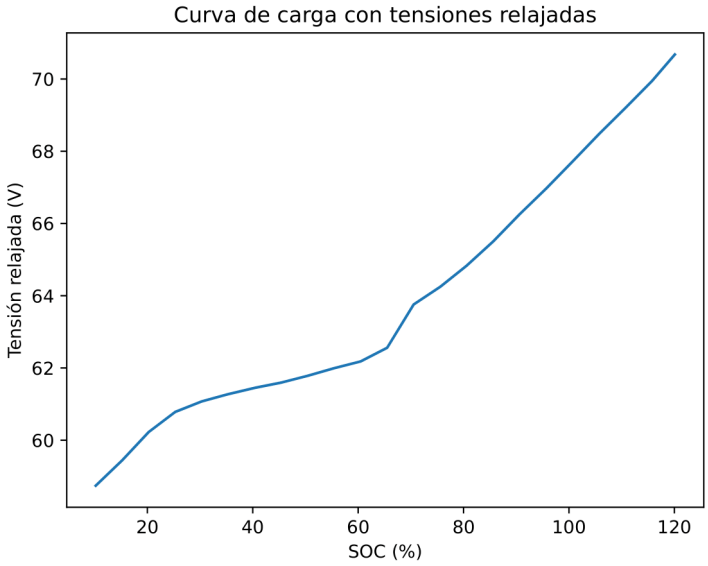


Figura 4.8: Curva de carga de la batería ($V_{relajada} - SOC$)

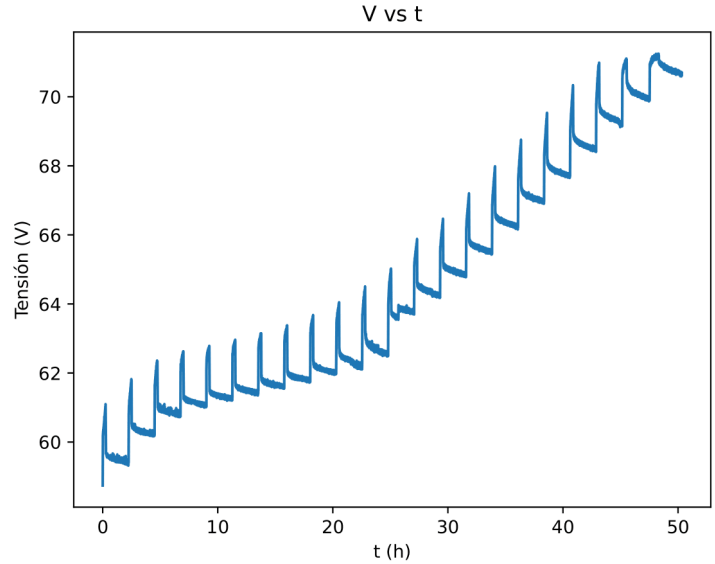


Figura 4.9: Curva de carga de la batería $(V - t)$

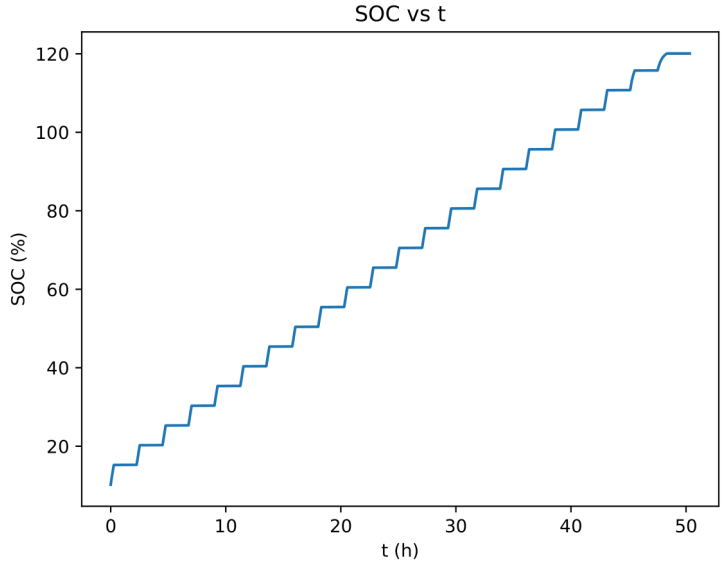


Figura 4.10: Curva de carga de la batería $(SOC - t)$

4.2.2. Análisis y conclusiones

En primer lugar, como se puede observar en la figura 4.7, la herramienta realiza el ensayo correctamente, tal y como se define en el punto 3.2.2.

Además, cumple con la condición de parada ya que se detiene al llegar a la intensidad umbral de 0.4 A. También se observa, que los escalones de descarga son del 5 %.

Por otro lado, en las figuras 4.9 y 4.10 el tiempo de relajación es de 2 horas, tal y como se ha definido.

Igualmente, se ha aprovechado el ensayo para realizar múltiples test de errores que cubran todos los posibles fallos descritos en la sección 3.4.4 teniendo en cuenta las especificaciones y limitaciones concretas de esta funcionalidad, y se puede afirmar que la herramienta maneja todos los tipos de errores ampliamente.

Por tanto, se concluye que la aplicación responde de forma fiable y segura a esta funcionalidad, atendiendo a todas las condiciones y limitaciones definidas en el proceso de desarrollo y a los requisitos introducidos por el usuario, y cumpliendo con su objetivo principal que es mostrar el comportamiento de una batería durante un determinado ensayo de carga.

4.3. Corrección de las curvas de los ensayos de carga y descarga

Si se presta atención a las curvas con tensiones relajadas de la figura 4.3 para la descarga y de la figura 4.8 para la carga, se puede observar que la descarga llega hasta un 10 % del SOC y la carga hasta un 120 % del SOC. Esto es debido a las perdidas que presenta la batería, es decir, la batería tiene un cierto rendimiento que tiene que ser tenido en cuenta para obtener unos resultados correctos.

Para calcular el rendimiento, en primer lugar, ha de obtenerse la energía saliente de la batería al descargarse y la energía entrante al cargarse. Si se considera que la energía de carga o descarga se define como:

$$E = \int_{t_0}^{t_k} v(t) \cdot i(t)\, dt \tag{4.1}$$

Donde, E es la energía total que entra o sale de la batería; t es el tiempo; t_0 es el instante inicial de carga o descarga; t_k es el instante final de carga o descargas; $v(t)$ es la curva de tensión de carga o descarga; $i(t)$ es la curva de intensidad de carga o descarga.

Es decir, lo que representa en realidad la ecuación 4.1 es el área bajo la curva $v(t) \cdot i(t)$. Considerando que, la intensidad siempre toma el valor de la corriente de carga/descarga o cero, se puede realizar la siguiente aproximación para su calculo:

$$E_i = E_{i-1} + \frac{v_i + v_{i+1}}{2} \cdot i_i \cdot (t_{i+1} - t_i), \quad \forall i = 1, ..., k \tag{4.2}$$

$$E_0 = \frac{v_0 + v_1}{2} \cdot i_0 \cdot (t_1 - t_0) \tag{4.3}$$

Donde, i representa cada muestra; k es el número total de muestras; E_i es la energía en una determinada muestra; E_{i-1} es la energía en la muestra anterior; v_i es la tensión en una determinada muestra; v_{i+1} es la tensión de la muestra siguiente; i_i es la intensidad de una determinada muestra; t_i es el tiempo transcurrido hasta la una determinada muestra; t_{i+1} es el tiempo transcurrido hasta la siguiente muestra; E_0 es la energía inicial; v_0 y v_1 son las tensiones de la primera y la segunda muestra, respectivamente;

i_0 es la corriente de la primera muestra; t_0 y t_1 son los tiempos transcurridos hasta la primera y la segunda muestra, respectivamente.

Una vez se han obtenido las energías de carga y descarga, se hallan las perdidas de la siguiente forma:

$$perdidas = E_c - E_d \tag{4.4}$$

Donde, E_c y E_d son las energías de carga y descarga, respectivamente.

Ahora, se obtiene el rendimiento considerando que el este es el mismo en la carga que en la descarga:

$$\eta = \sqrt{\frac{E_d}{E_d + perdidas}} \tag{4.5}$$

Finalmente, se recalculan las curvas del ensayo de carga según la ecuación 4.8, y del ensayo de descarga según as ecuación 4.6.

$$SOC_i = SOC_{i-1} - \frac{I_{d,i} \cdot (t/3600)}{C \cdot \eta} \cdot 100 \quad \forall i = 1, ..., k \tag{4.6}$$

$$SOC_0 = 100 \tag{4.7}$$

Donde, i representa cada muestra; k es el número total de muestras; SOC_i es el estado de carga de una determinada muestra, dado en porcentaje; SOC_{i-1} es el estado de carga en la muestra anterior, dado en porcentaje; $I_{d,i}$ es la intensidad de descarga en una determinada muestra, en A; t es el periodo de muestreo, en segundos; C es la capacidad del sistema de almacenamiento, en Ah; η es el rendimiento de la batería; SOC_0 es el estado de carga inicial.

$$SOC_i = SOC_{i-1} + \frac{I_{c,i} \cdot (t/3600) \cdot \eta}{C} \cdot 100 \quad \forall i = 1, ..., k \tag{4.8}$$

$$SOC_0 = SOC_{d,k} \tag{4.9}$$

Donde, i representa cada muestra; k es el número total de muestras; SOC_i es el estado de carga de una determinada muestra, dado en porcentaje; SOC_{i-1} es el estado de carga en la muestra anterior, dado en porcentaje; $I_{c,i}$ es la intensidad de carga en una determinada muestra, en A; t es el periodo de muestreo, en segundos; C es la capacidad del sistema de almacenamiento, en Ah; η es el rendimiento de la batería; SOC_0 es el estado de carga inicial; $SOC_{d,k}$ es el estado de carga final de la descarga.

Dando como resultado, la curva corregida del ensayo de descarga con tensiones relajadas (figura 4.11) y la curva corregida del ensayo de carga con tensiones relajadas (figura 4.12).

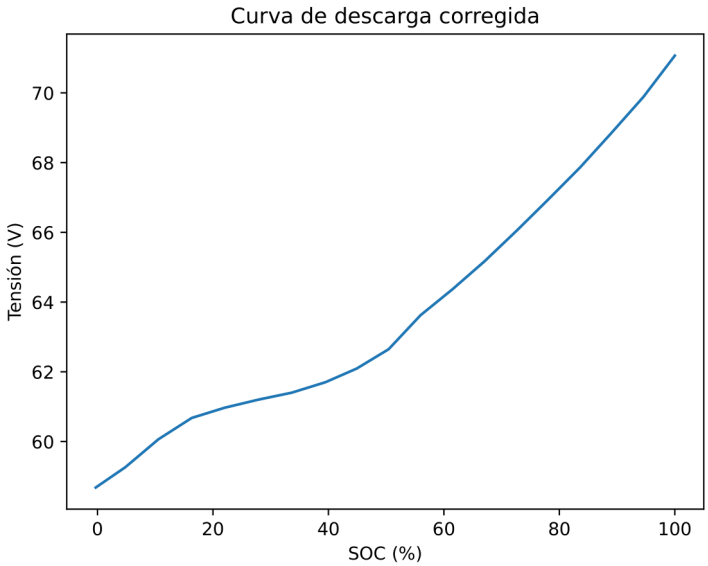


Figura 4.11: Curva de descarga corregida de la batería ($V_{relajada} - SOC$)

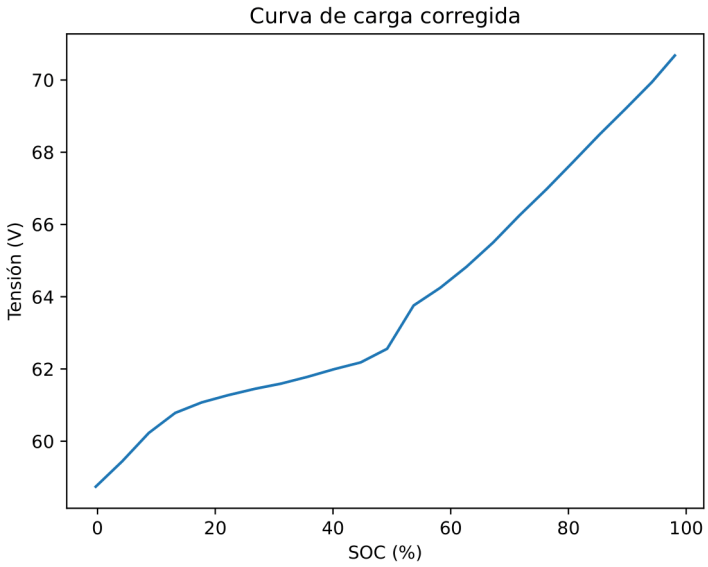


Figura 4.12: Curva de carga corregida de la batería ($V_{relajada} - SOC$)

Ahora, como se puede observar, las curvas de carga y descarga están dentro de los límites esperados.

4.4. Simulación de un sistema de FV con curvas de irradiancia y consumo en la batería

Esta prueba pretende mostrar el correcto funcionamiento de la simulación de un sistema de FV con curvas de irradiancia y consumo en la batería conforme a las especificaciones

definidas en la sección 3.2.2. En este caso, se han fijado los siguientes parámetros de entrada:

- Performance Ratio: 1
- SOC objetivo (%): 10
- Potencia instalada (W): 1000
- Radiación estandar (W): 1000
- Irradiancia máxima (W/m2): AUTO
- Consumo máximo (W): AUTO
- Intensidad máxima de la carga (A): 4
- Intensidad máxima de la fuente (A): 4
- Escala de tiempo de la fuente (s): 3600
- Escala de tiempo de la carga (s): 60
- Curva de irradiancia de 24 horas
- Curva de consumo de 24 horas

4.4.1. Resultados

A continuación, se muestran las gráficas obtenidas en la simulación:

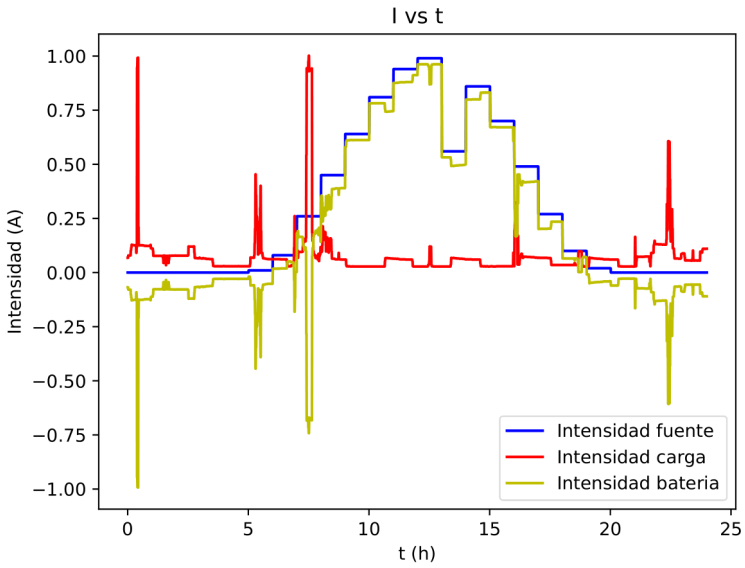


Figura 4.13: Curvas de intensidades de la simulación ($I - t$)

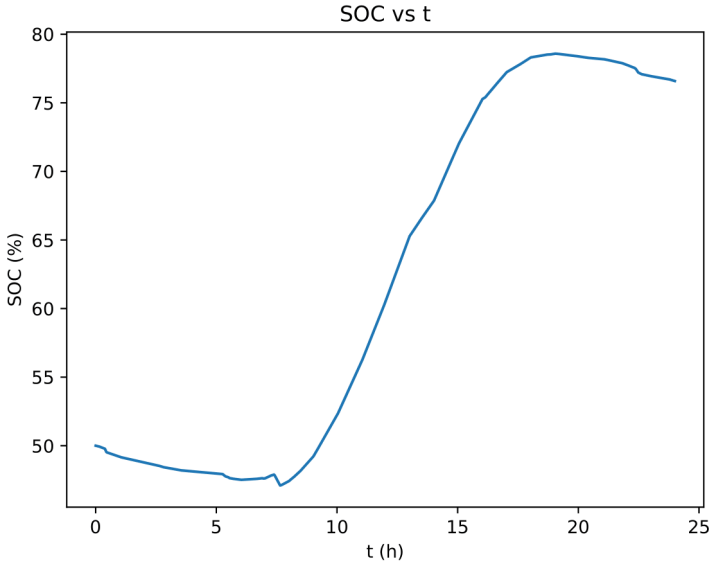


Figura 4.14: Curva del SOC de la batería durante la simulación ($SOC - t$)

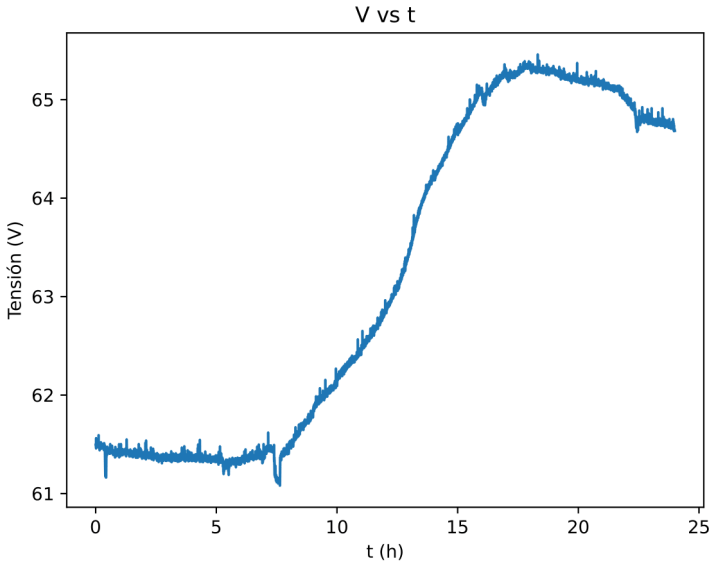


Figura 4.15: Curva de tensión de la batería durante la simulación ($V - t$)

4.4.2. Análisis y conclusiones

En primer lugar, como se puede observar en la figura 4.13, la herramienta realiza la simulación correctamente, tal y como se define en el punto 3.2.2.

Por otro lado, las figuras 4.14 y 4.15 muestran la evolución de la batería en SOC y tensión, respectivamente, a lo largo de la simulación, lo que da una descripción del comportamiento de la batería en sistemas fotovoltaicos.

Igualmente, se ha aprovechado el ensayo para realizar múltiples test de errores que cubran todos los posibles fallos descritos en la sección 3.4.4 teniendo en cuenta las especificaciones y limitaciones concretas de esta funcionalidad, y se puede afirmar que la herramienta maneja todos los tipos de errores de forma correcta.

Por tanto, se concluye que la aplicación responde de forma fiable y segura a esta funcionalidad, atendiendo a todas las condiciones y limitaciones definidas en el proceso de desarrollo y a los requisitos introducidos por el usuario, y cumpliendo con su objetivo principal que es mostrar el comportamiento de una batería durante una simulación de un sistema de FV a partir de curvas de irradiancia y consumo.

4.5. Simulación de un sistema de FV con perfiles de intensidad en la batería

Esta prueba pretende mostrar el correcto funcionamiento de la simulación de un sistema de FV con perfiles de intensidad en la carga y en la fuente, en la batería conforme a las especificaciones definidas en la sección 3.2.2. En este caso, se han fijado los siguientes parámetros de entrada:

- Perfil de intensidad para la fuente de 1.6 horas en escalones de 1 minuto
- Perfil de intensidad para la carga de 1.6 horas en escalones de 4 minutos

4.5.1. Resultados

A continuación, se muestran las gráficas obtenidas en la simulación:

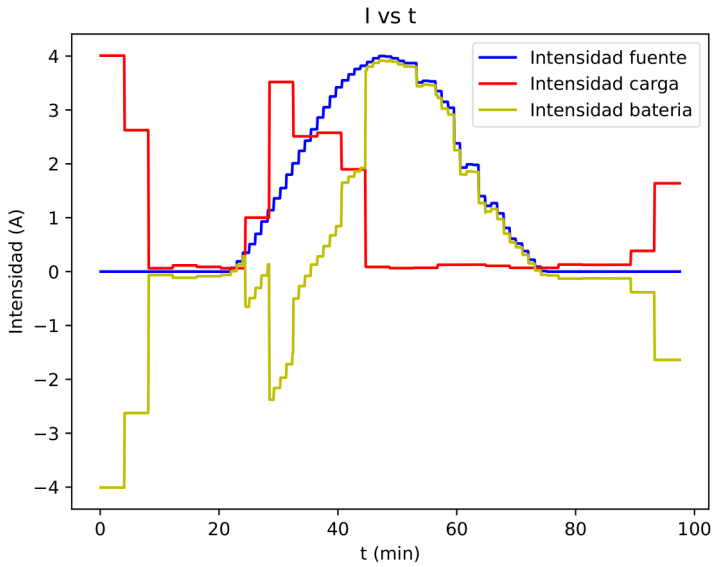


Figura 4.16: Curvas de intensidades de la simulación $(I - t)$

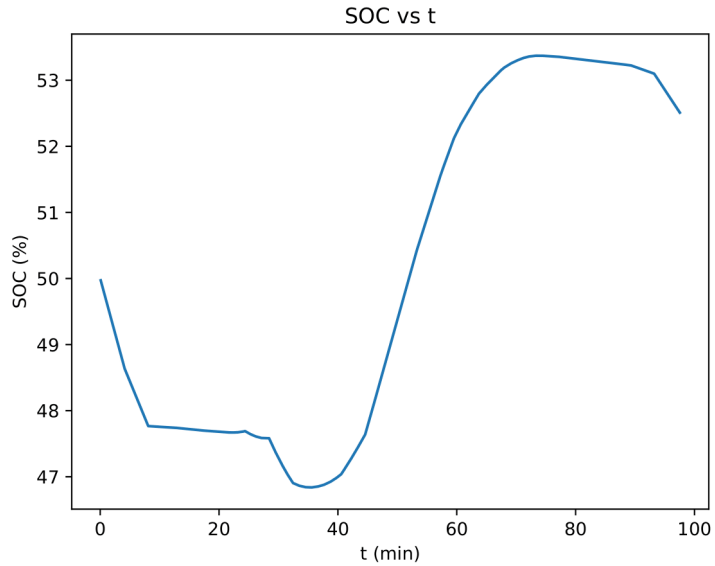


Figura 4.17: Curva del SOC de la batería durante la simulación ($SOC - t$)

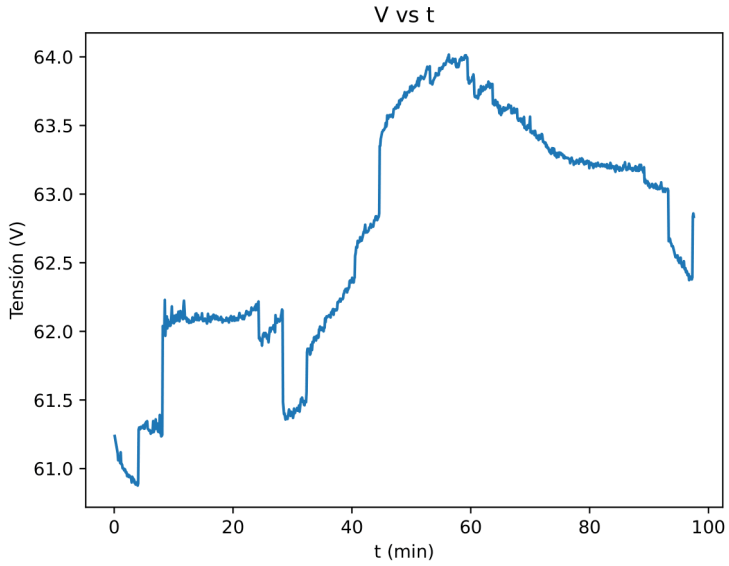


Figura 4.18: Curva de tensión de la batería durante la simulación ($V - t$)

4.5.2. Análisis y conclusiones

En primer lugar, como se puede observar en la figura 4.16, la herramienta realiza la simulación correctamente, tal y como se define en el punto 3.2.2.

Por otro lado, las figuras 4.17 y 4.18 muestran la evolución de la batería en SOC y tensión, respectivamente, a lo largo de la simulación, lo que da una descripción del comportamiento de la batería en sistemas de FV.

Igualmente, se ha aprovechado el ensayo para realizar múltiples test de errores que cubran todos los posibles fallos descritos en la sección 3.4.4 teniendo en cuenta las especificaciones y limitaciones concretas de esta funcionalidad, y se puede afirmar que la herramienta maneja todos los tipos de errores de forma correcta.

Por tanto, se concluye que la aplicación responde de forma fiable y segura en esta funcionalidad, atendiendo a todas las condiciones y limitaciones definidas en el proceso de desarrollo y a los requisitos introducidos por el usuario, y cumpliendo con su objetivo principal que es mostrar el comportamiento de una batería durante una simulación de un sistema de FV a partir de curvas de intensidad definidas por el usuario.

4.6. Ensayo de calibración de la fuente de alimentación

Como ya se ha mencionado en la sección 3.2.1 se ha detectado que la fuente de alimentación no muestra el valor real de tensión cuando está es menor de 12 V. En principio esto podría deberse a un error de calibración de la fuente, por lo que se ha calibrado la fuente de diferentes maneras y, posteriormente, se ha realizado un ensayo de cada una para ver cómo afecta la calibración al error de medida de la tensión.

Para las diferentes calibraciones se ha escogido el potenciómetro 1, que según la guía de usuario de la fuente de alimentación es la que podría estar causando este error, y se ha modificado su valor entre $default$ (valor de fábrica), 0 (valor mínimo) o 255 (valor máximo). Además, se han hecho pruebas sin carga y con una carga de 100 Ω.

En las gráficas que se van a mostrar a continuación se denota la tensión de referencia como V_{ref}, la tensión del display de la fuente como V_{dis} y la tensión medida con el sistema de medida como V_{volt}.

4.6.1. Ensayo de calibración 1

Configuración de la calibración:

- Potenciómetro 1 = default
- Carga = 0

Resultados de la calibración:

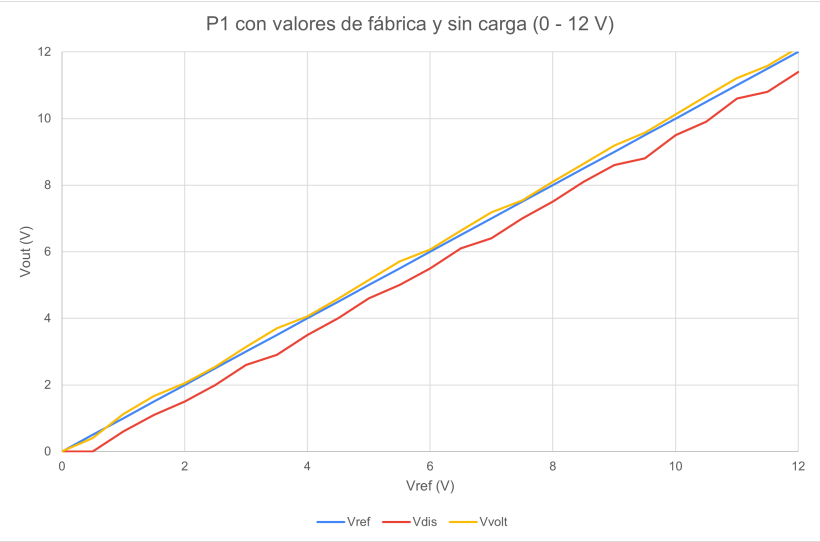


Figura 4.19: P1 con valores de fábrica y sin carga (0 - 12 V)

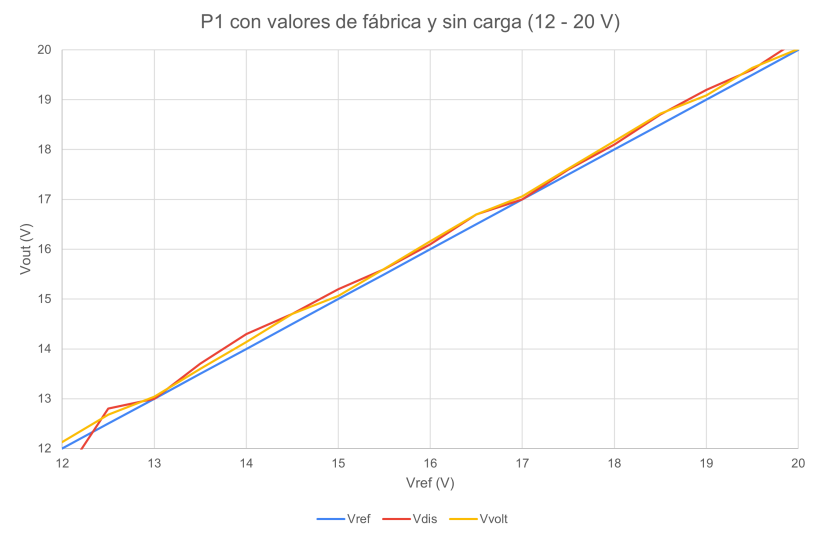


Figura 4.20: P1 con valores de fábrica y sin carga (12 - 20 V)

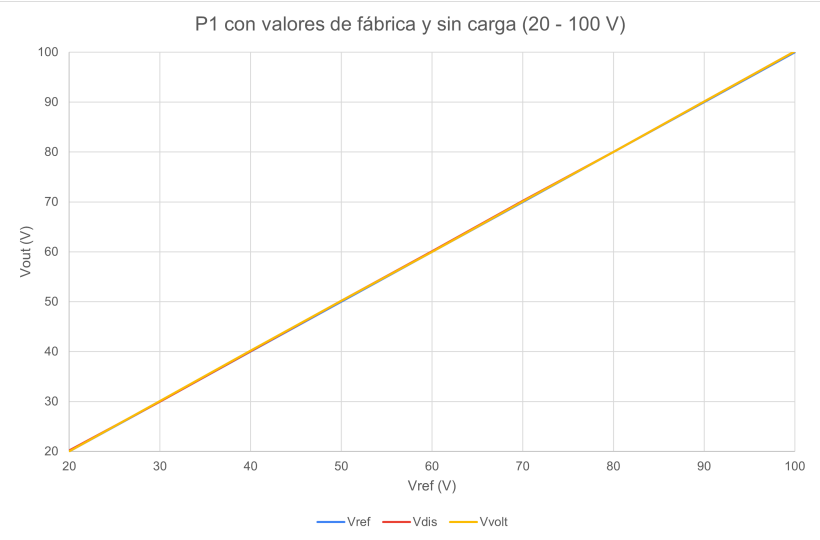


Figura 4.21: P1 con valores de fábrica y sin carga (20 - 100 V)

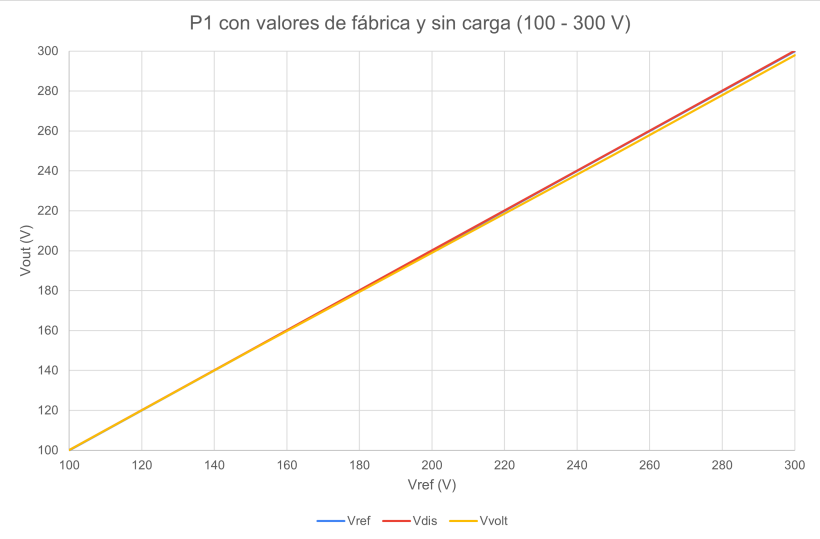


Figura 4.22: P1 con valores de fábrica y sin carga (100 - 300 V)

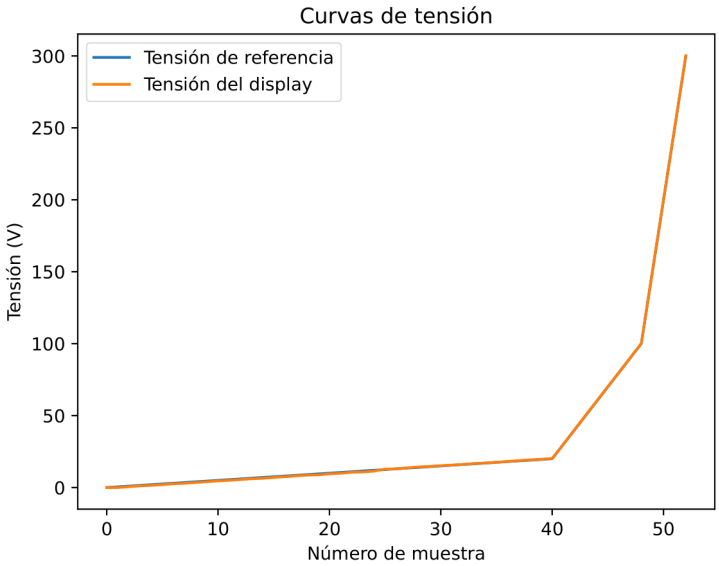


Figura 4.23: Curvas de tensión

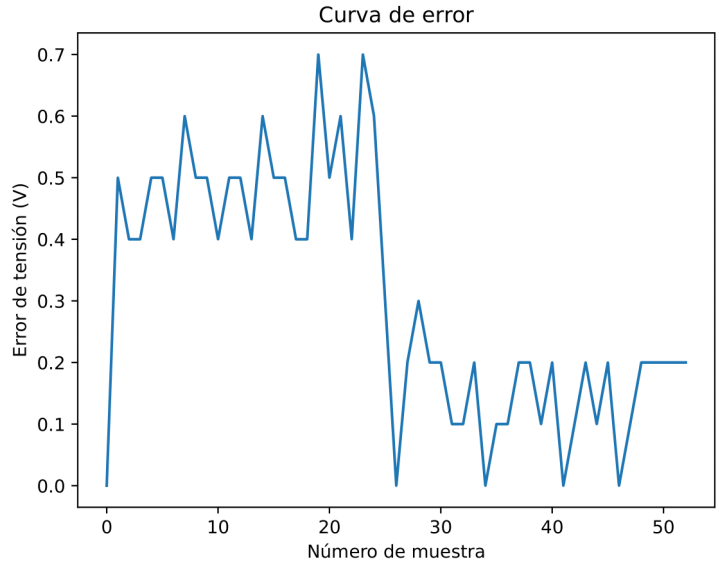


Figura 4.24: Error

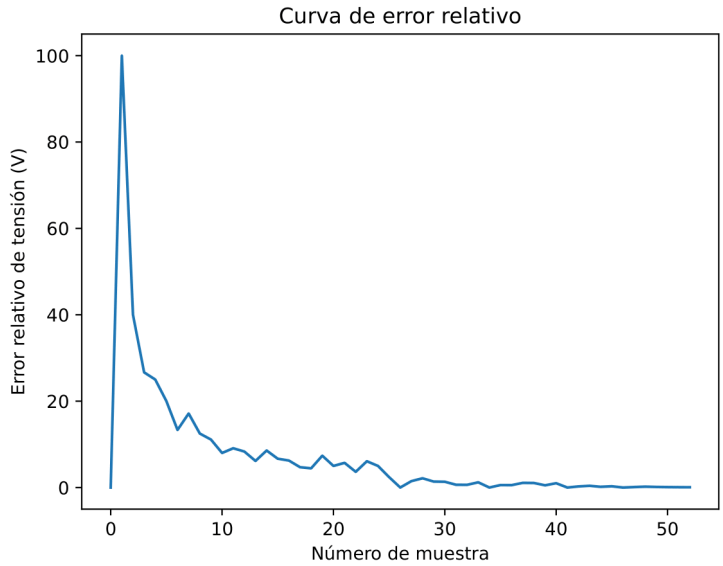


Figura 4.25: Error relativo

4.6.2. Ensayo de calibración 2

Configuración de la calibración:

- Potenciómetro 1 = default
- Carga = 100 Ω

Resultados de la calibración:

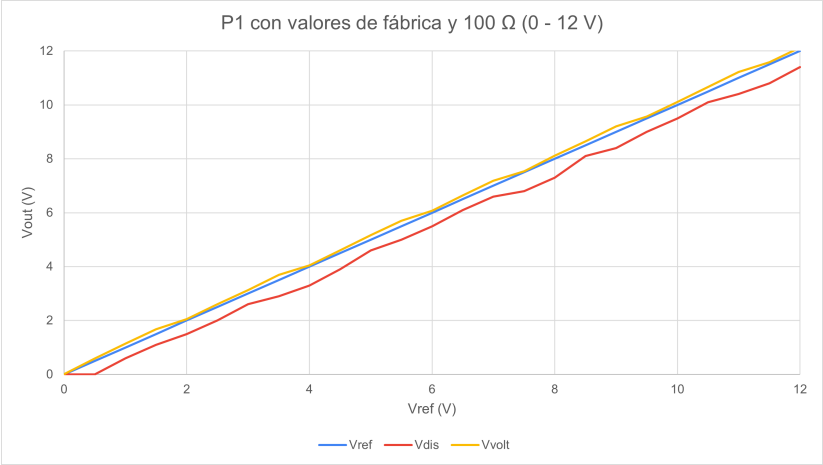


Figura 4.26: P1 con valores de fábrica y 100 Ω (0 - 12 V)

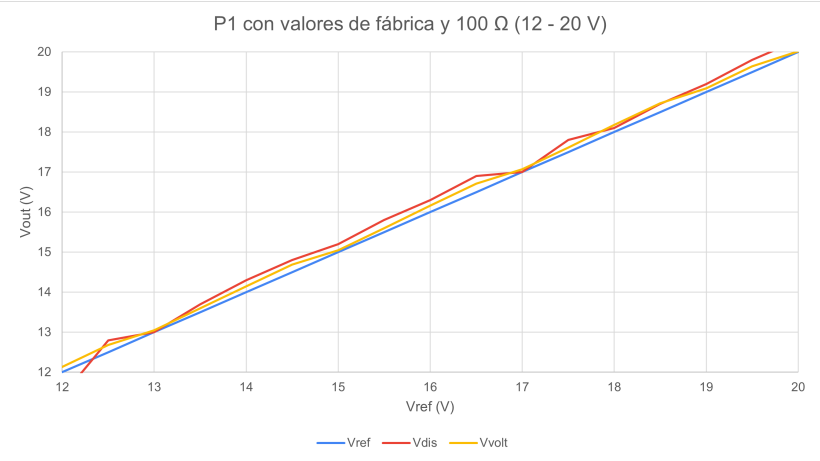


Figura 4.27: P1 con valores de fábrica y 100 Ω (12 - 20 V)

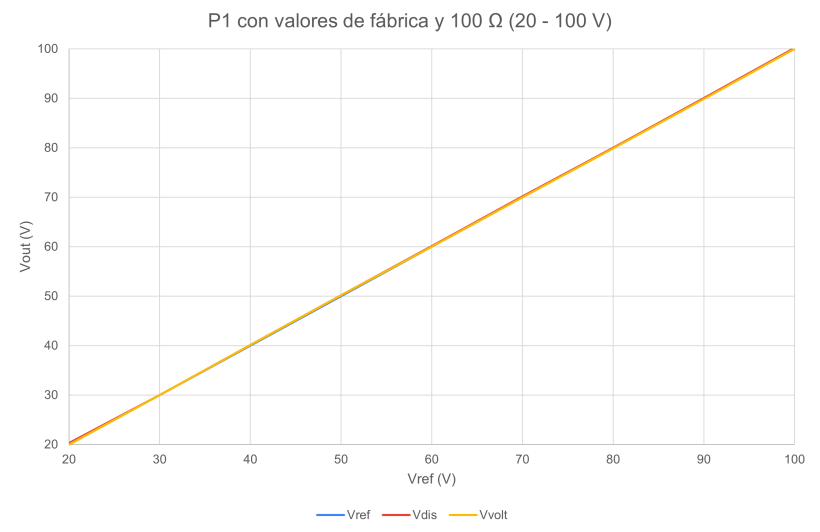


Figura 4.28: P1 con valores de fábrica y 100 Ω (20 - 100 V)

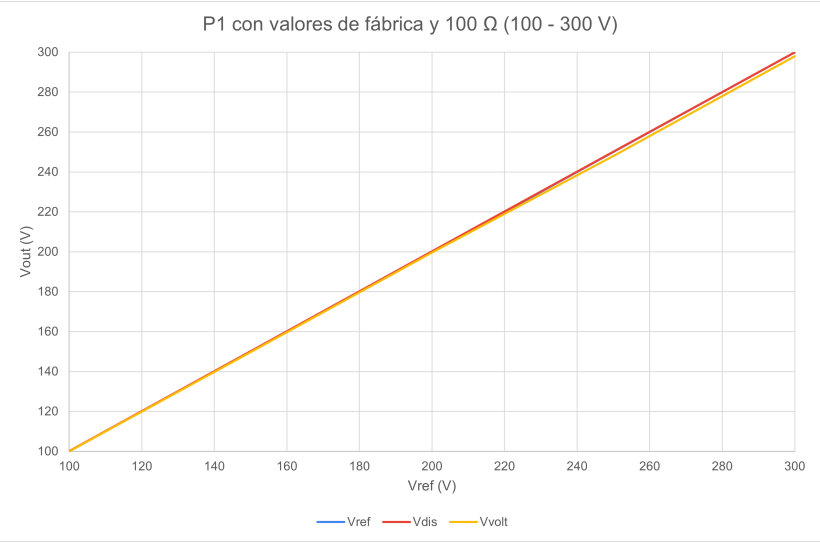


Figura 4.29: P1 con valores de fábrica y 100 Ω (100 - 300 V)

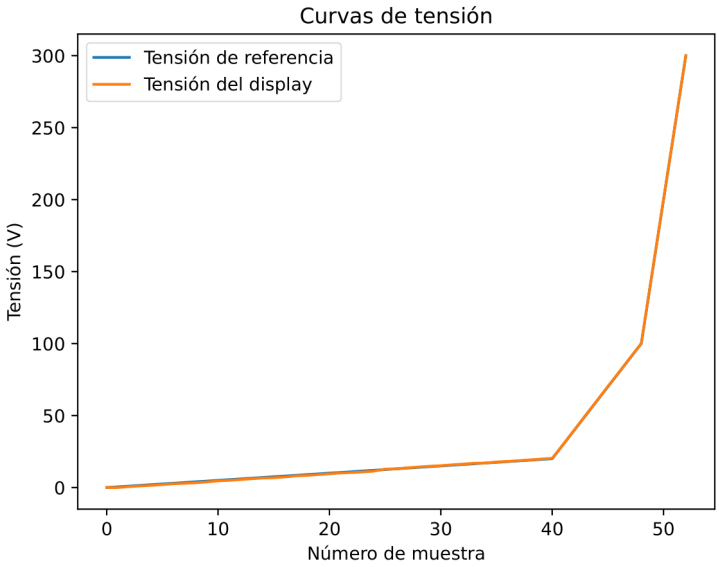


Figura 4.30: Curvas de tensión

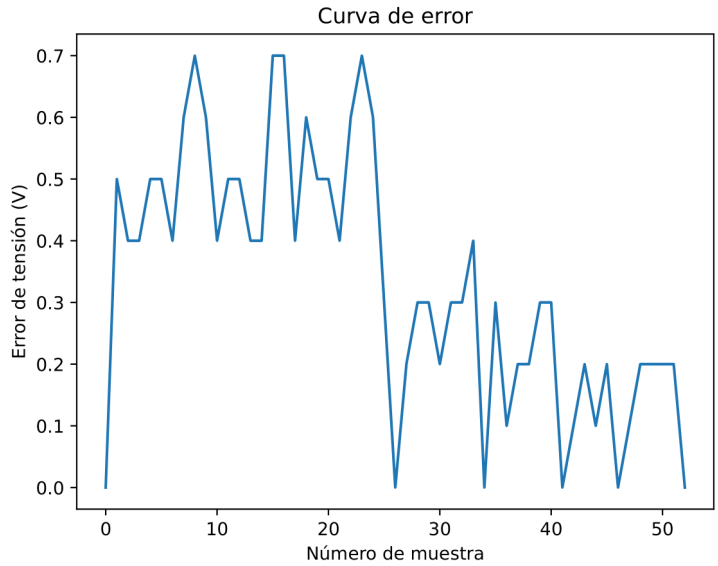


Figura 4.31: Error

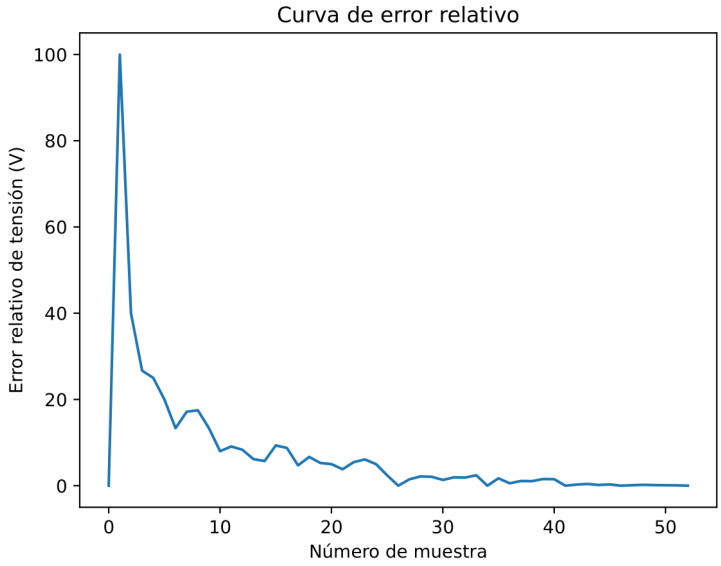


Figura 4.32: Error relativo

4.6.3. Ensayo de calibración 3

Configuración de la calibración:

- Potenciómetro 1 = 0
- Carga = 0

Resultados de la calibración:

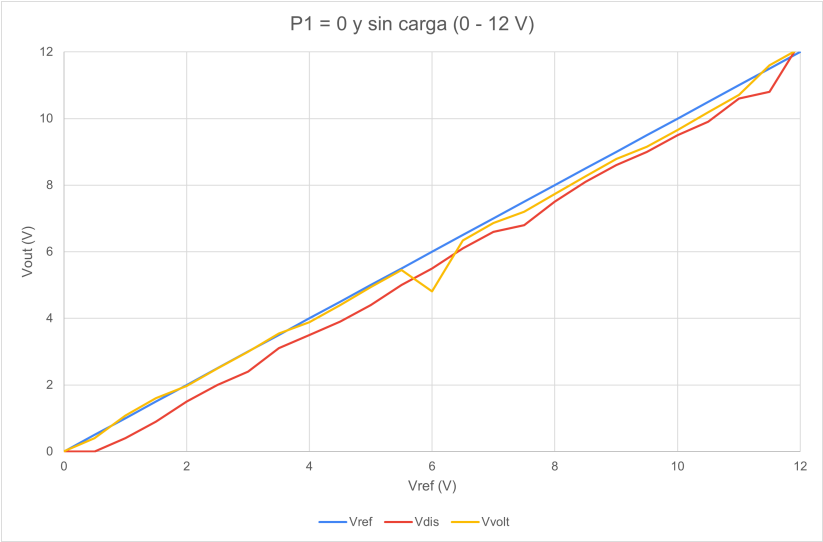


Figura 4.33: P1 = 0 y sin carga (0 - 12 V)

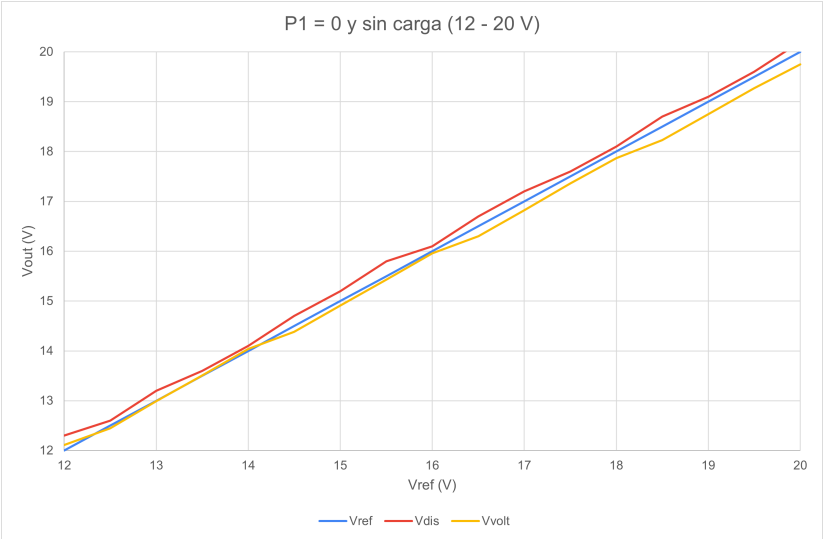


Figura 4.34: P1 = 0 y sin carga (12 - 20 V)

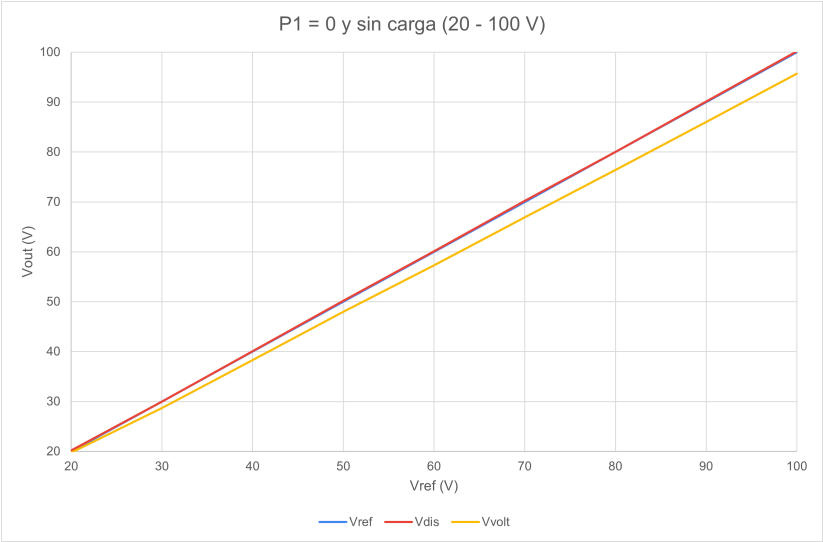


Figura 4.35: P1 = 0 y sin carga (20 - 100 V)

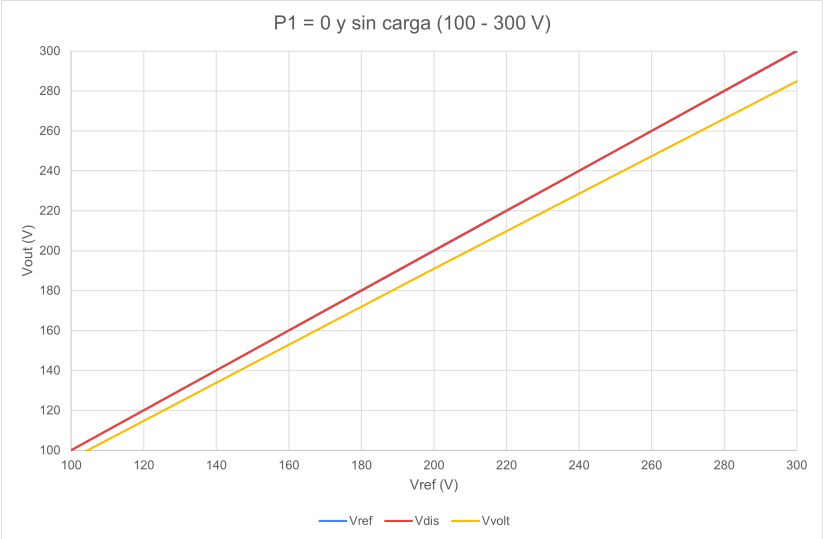


Figura 4.36: P1 = 0 y sin carga (100 - 300 V)

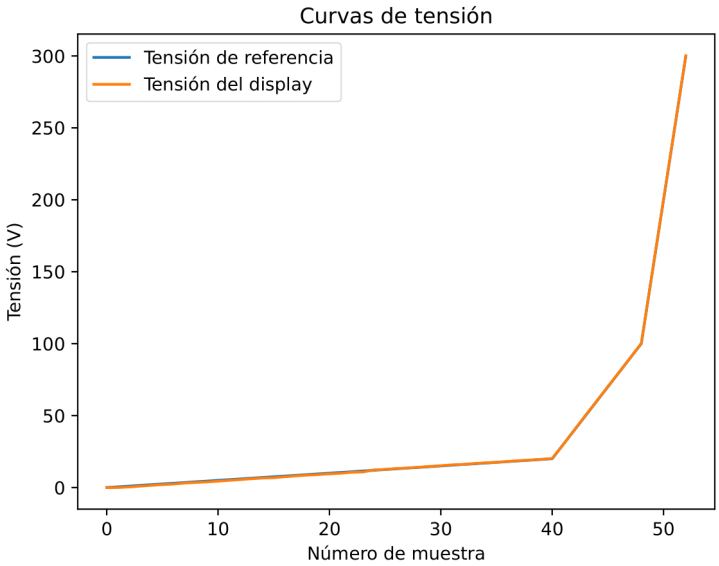


Figura 4.37: Curvas de tensión

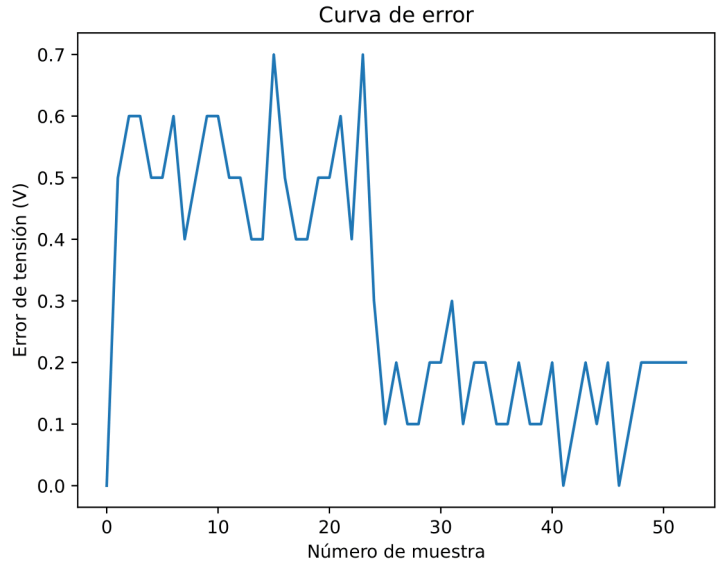


Figura 4.38: Error

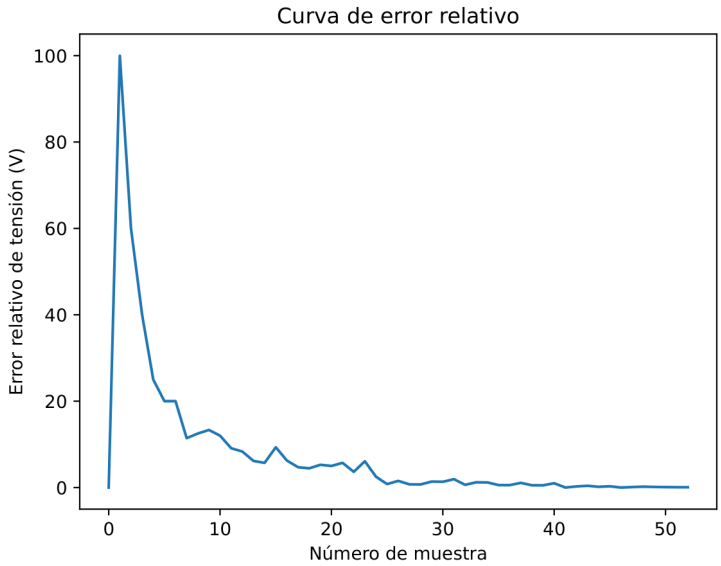


Figura 4.39: Error relativo

4.6.4. Ensayo de calibración 4

Configuración de la calibración:

- Potenciómetro 1 = 0
- Carga = 100 Ω

Resultados de la calibración:

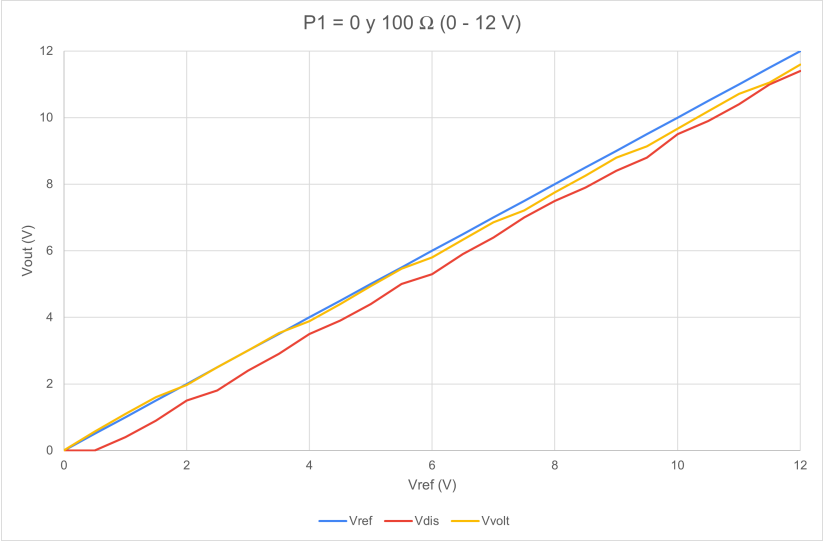


Figura 4.40: P1 = 0 y 100 Ω (0 - 12 V)

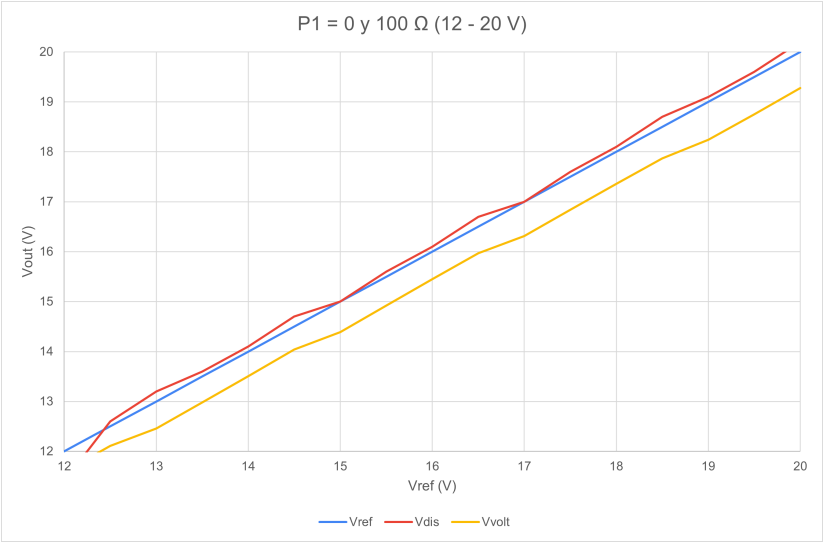


Figura 4.41: P1 = 0 y 100 Ω (12 - 20 V)

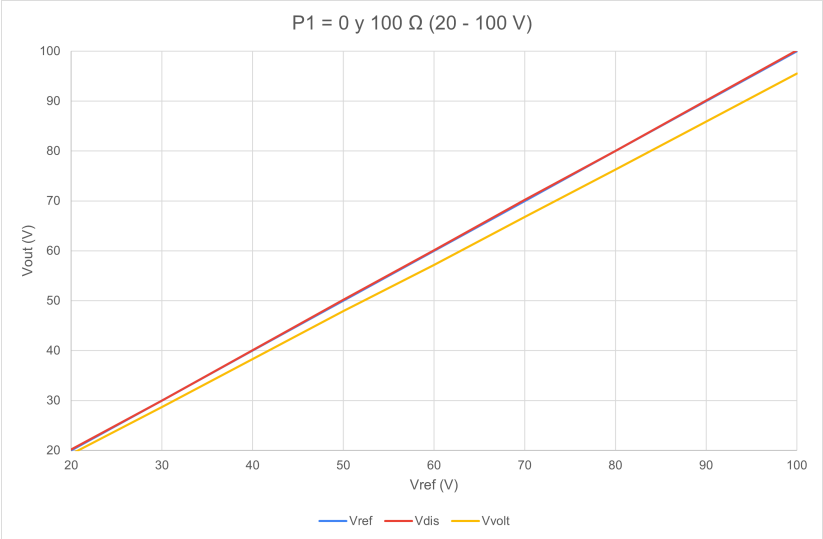


Figura 4.42: P1 = 0 y 100 Ω (20 - 100 V)

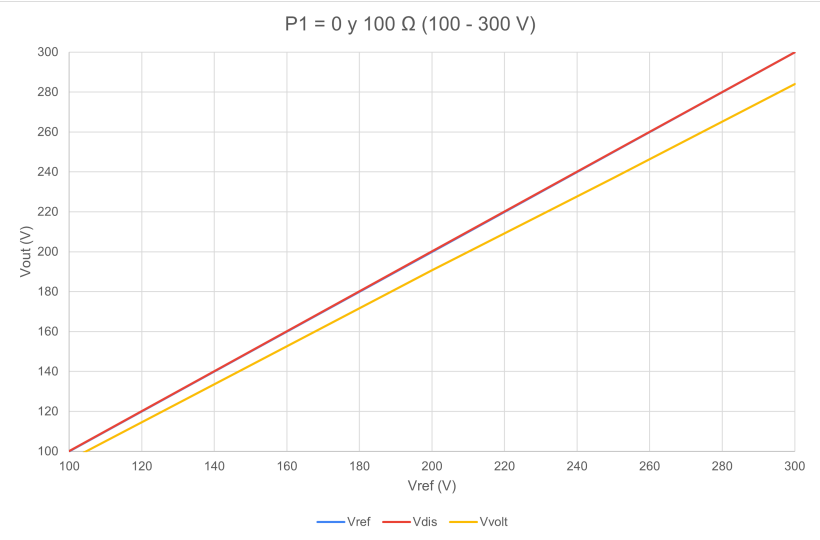


Figura 4.43: P1 = 0 y 100 Ω (100 - 300 V)

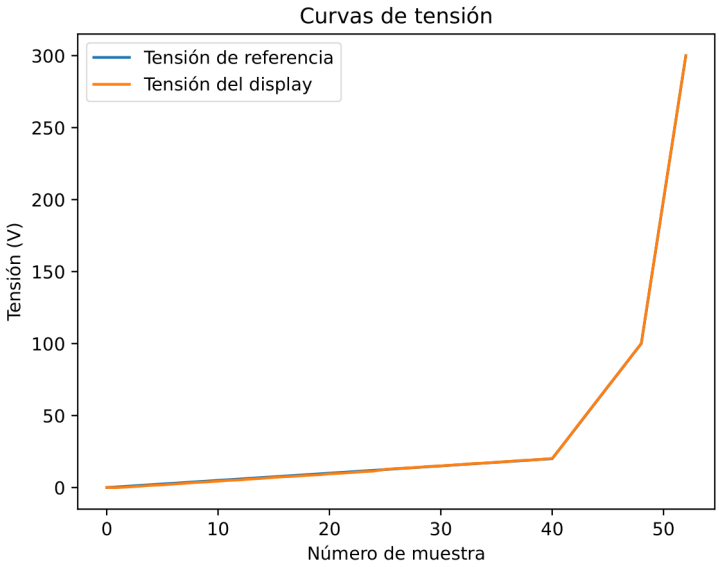


Figura 4.44: Curvas de tensión

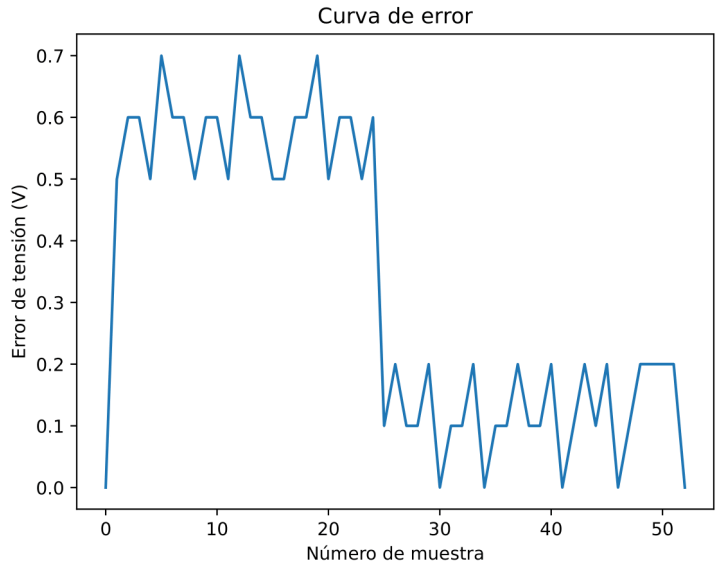


Figura 4.45: Error

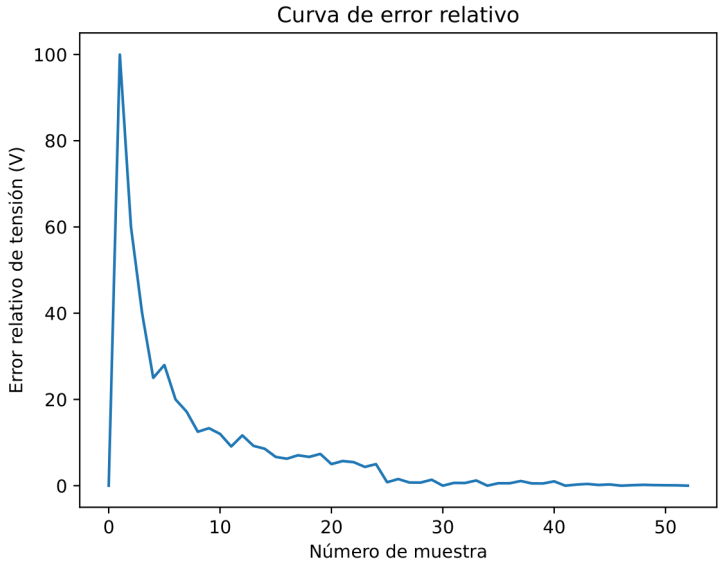


Figura 4.46: Error relativo

4.6.5. Ensayo de calibración 5

Configuración de la calibración:

- Potenciómetro 1 = 255
- Carga = 0

Resultados de la calibración:

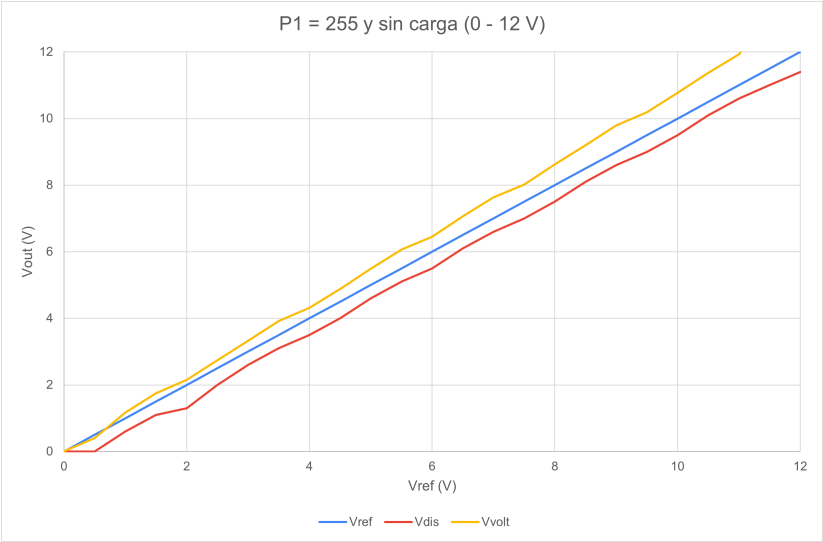


Figura 4.47: P1 = 255 y sin carga (0 - 12 V)

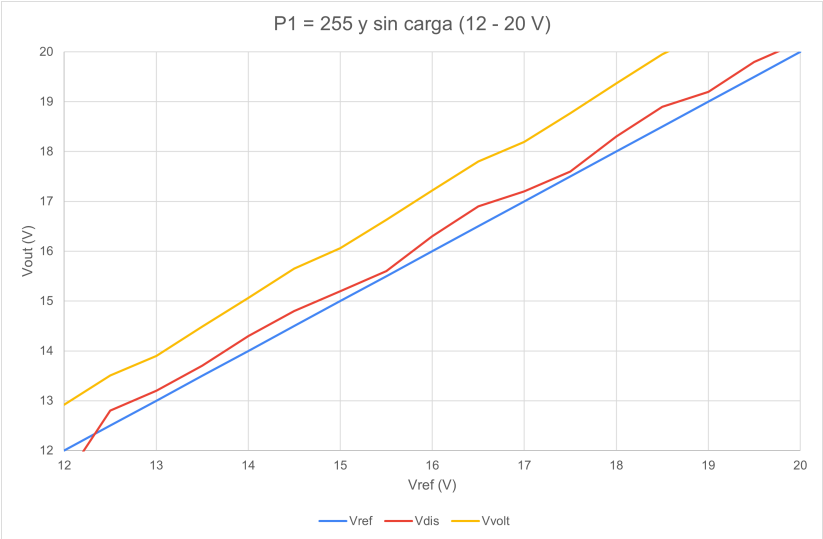


Figura 4.48: P1 = 255 y sin carga (12 - 20 V)

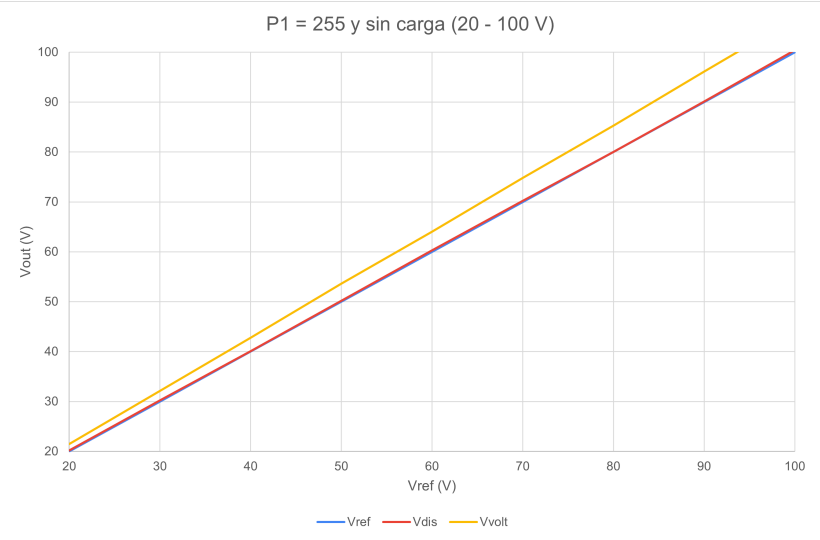


Figura 4.49: P1 = 255 y sin carga (20 - 100 V)

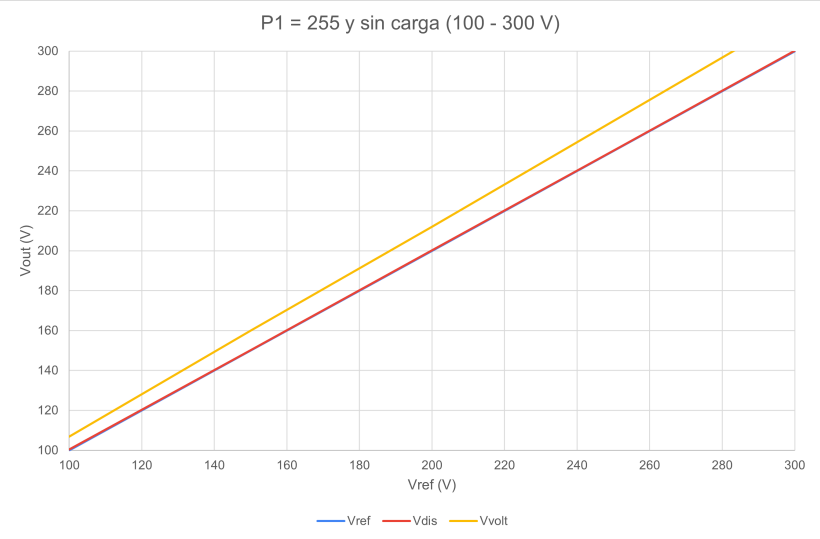


Figura 4.50: P1 = 255 y sin carga (100 - 300 V)

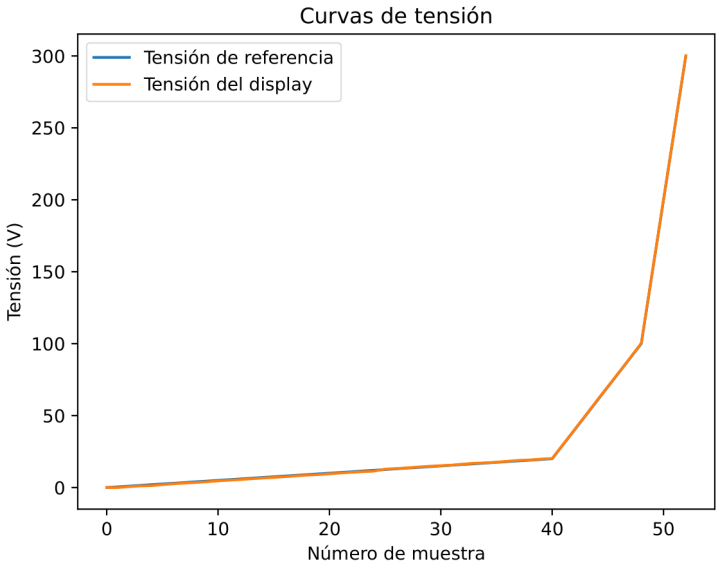


Figura 4.51: Curvas de tensión

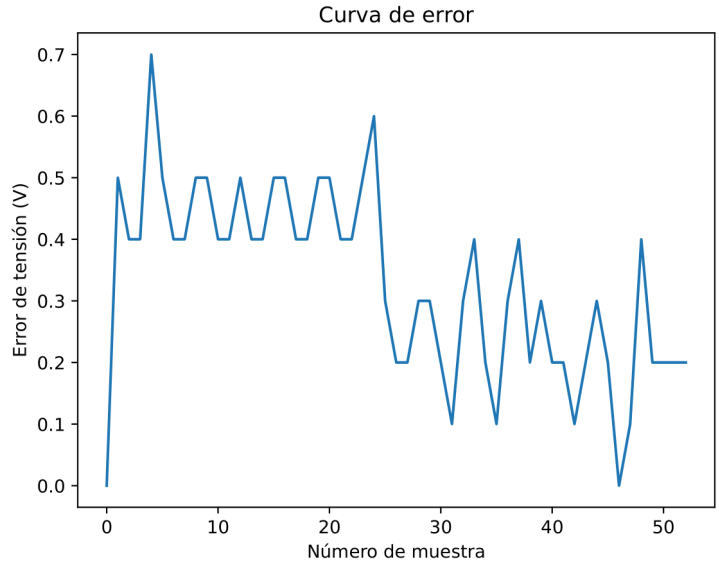


Figura 4.52: Error

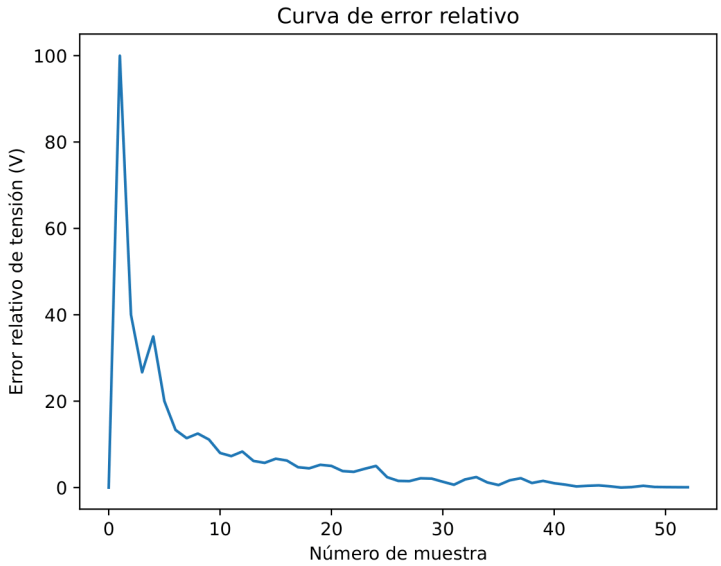


Figura 4.53: Error relativo

4.6.6. Ensayo de calibración 6

Configuración de la calibración:

- Potenciómetro 1 = 255
- Carga = 100 Ω

Resultados de la calibración:

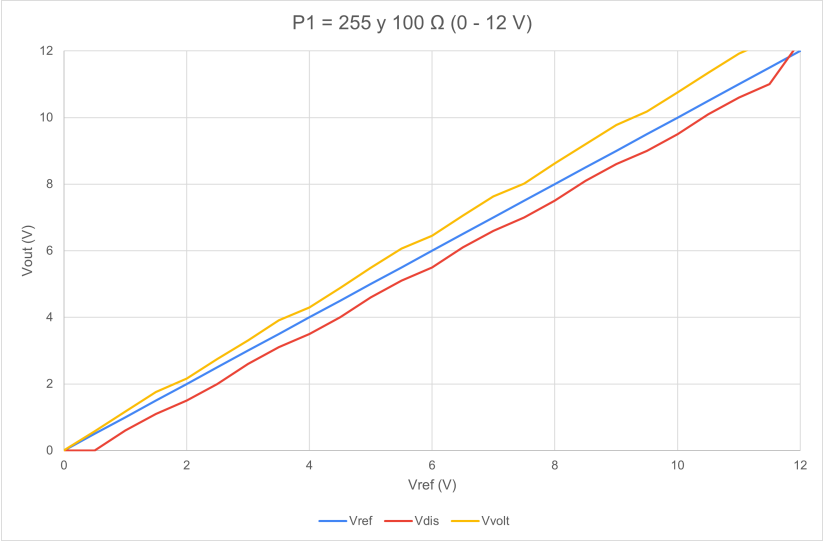


Figura 4.54: P1 = 255 y 100 Ω (0 - 12 V)

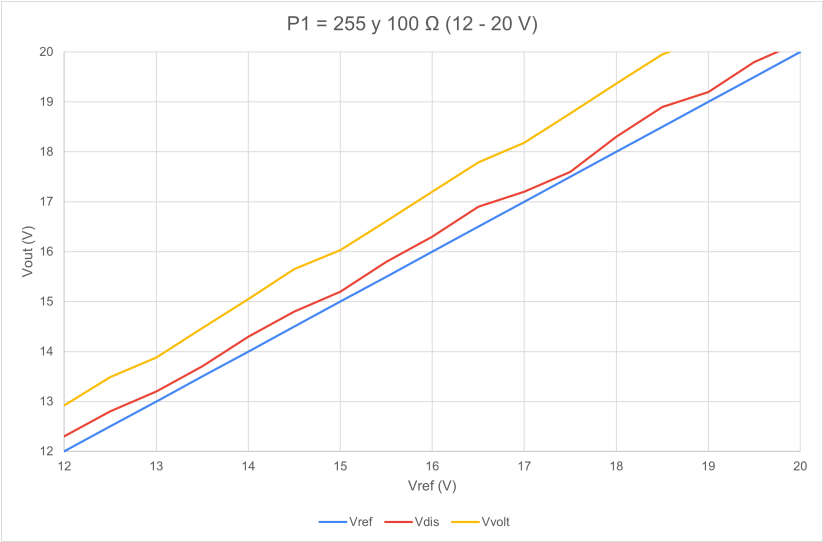


Figura 4.55: P1 = 255 y 100 Ω (12 - 20 V)

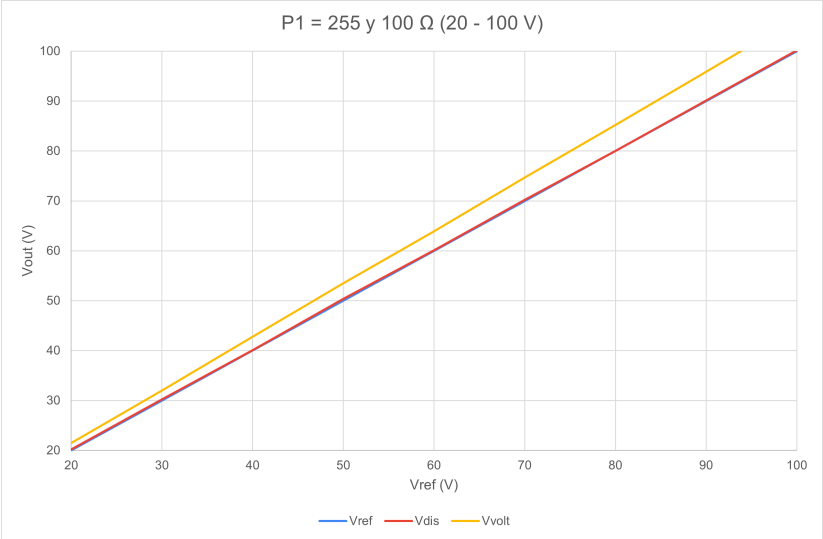


Figura 4.56: P1 = 255 y 100 Ω (20 - 100 V)

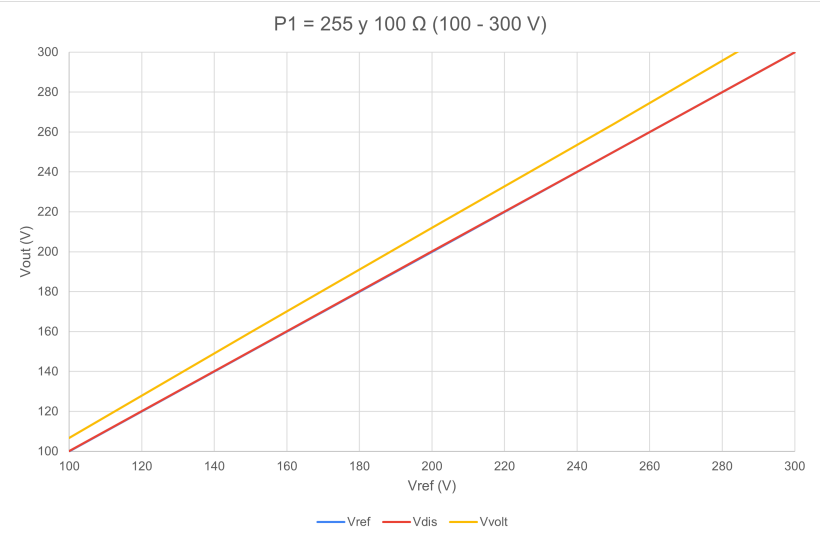


Figura 4.57: P1 = 255 y 100 Ω (100 - 300 V)

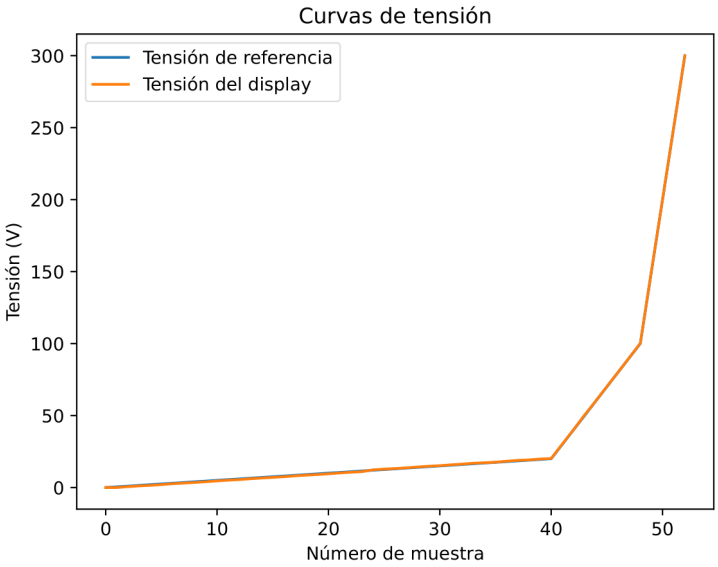


Figura 4.58: Curvas de tensión

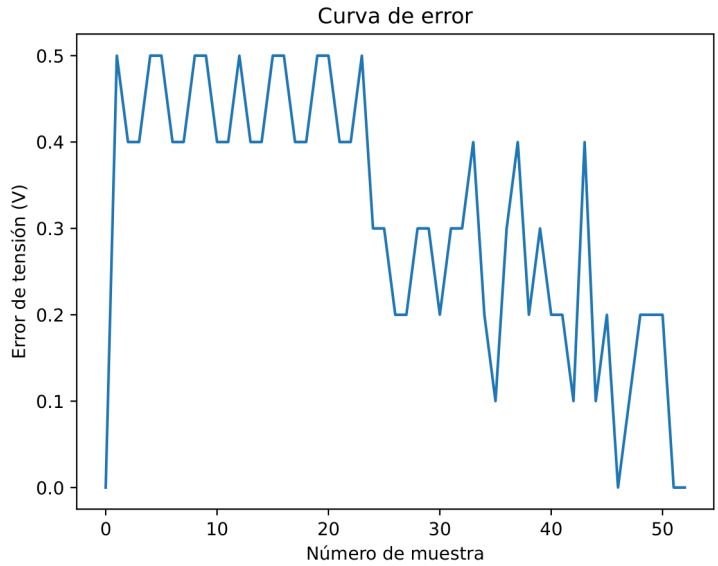


Figura 4.59: Error

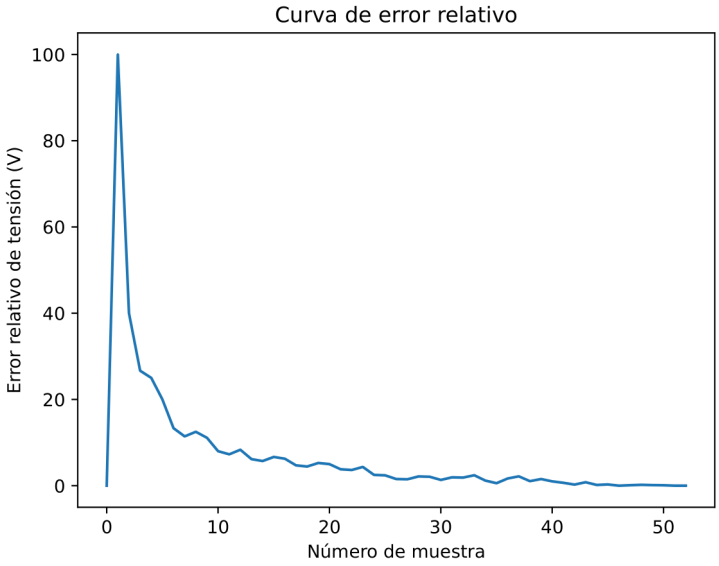


Figura 4.60: Error relativo

4.6.7. Análisis y conclusiones

De los experimentos anteriores se puede extraer que en ningún caso se observa ninguna mejora respecto de la configuración por defecto en el primer tramo de 0 a 12 V (donde se ha detectado el error) entre la tensión mostrada por la fuente y la tensión real. De hecho, a partir de los 12 V en la configuración por defecto se observa claramente como todas las tensiones coinciden y el error disminuye conforme aumenta la tensión. Sin embargo, en las otras configuraciones empiezan a aparecer otros errores entre las tensiones a

voltajes mayores. De lo cual, se deduce que tampoco configuraciones intermedias entre 0 y $default$ o $default$ y 255 mejorarán los resultados y que el mejor de los casos es dejar la calibración que lleva la fuente por defecto.

Por tanto, se concluye que el error de aproximadamente 0.5 V (figuras 4.24 y 4.31) que existe entre la tensión que da la fuente de alimentación y la tensión real hasta los 12 V no se puede solucionar realizando una calibración de la fuente. Es por eso que, se recomienda tener en cuenta este error cuando se emplee esta máquina para leer la tensión en cualquier aplicación y se intente, en la medida de lo posible, utilizar la fuente en rangos de tensión mayores a 12 V. Sin embargo, en lo que a este proyecto se refiere, este defecto no afecta en absoluto ya que se trata de un error en la lectura de tensión del display de la fuente y la herramienta emplea el sistema de medida de tensión externo desarrollado exclusivamente para la misma.

4.7. Resto de funcionalidades

Por una parte, en el resto de funcionalidades no nombradas en esta sección de pruebas y resultados, es decir, carga y descarga siguiendo un perfil, carga y descarga constante, lectura de tensión, apagado de la carga, la fuente y todo el sistema, también se han realizado una serie de pruebas que han demostrado que la herramienta las realiza correctamente, tal y como se define en la sección 3.2.2. Además, se trata de funciones secundarias que en realidad ya están implementadas dentro de las otras funcionalidades que se han visto hasta ahora.

Igualmente, se han aprovechado estas pruebas para realizar múltiples test de errores que cubran todos los posibles fallos descritos en la sección 3.4.4 teniendo en cuenta las especificaciones y limitaciones concretas de estas funcionalidades, y se puede afirmar que la herramienta maneja todos los tipos de errores de forma correcta.

Por tanto, se concluye que la aplicación responde de forma fiable y segura a estas funcionalidades, atendiendo a todas las condiciones y limitaciones definidas en el proceso de desarrollo y a los requisitos introducidos por el usuario, cumpliendo con su objetivo principal.

Por otra parte, además de las pruebas realizadas con la batería de la moto NX1, se ha evaluado la capacidad de la herramienta para adaptarse a diferentes modelos de sistemas de almacenamiento. En este sentido, se han repetido las pruebas anteriores utilizando la celda Panasonic con el objetivo de verificar la capacidad de la herramienta para adaptarse y funcionar correctamente con diferentes configuraciones de baterías.

A pesar de las limitaciones de tiempo, se realizaron los ensayos y simulaciones de la forma más adecuada posible para evaluar el funcionamiento del sistema con la celda. Si bien no se pudieron llevar a cabo los ensayos y simulaciones de forma convencional debido a restricciones temporales, se implementaron los casos de prueba más representativos y críticos para verificar el correcto funcionamiento de la herramienta. Esto garantiza que los resultados obtenidos sean confiables y respalden la validez y el correcto funcionamiento del sistema en diferentes configuraciones de baterías.

De esta forma, los resultados parciales obtenidos demuestran que el sistema maneja y simula adecuadamente el sistema con la celda. Lo que sugiere que la herramienta tiene

la flexibilidad y la capacidad de adaptación necesarias para ser utilizada con éxito en diversos modelos de sistemas de almacenamiento.

Siendo conscientes de las limitaciones en la realización de las pruebas con la celda, se recomienda en futuras investigaciones y desarrollos extender los ensayos y simulaciones a múltiples modelos de sistemas de almacenamiento, incluyendo una variedad de celdas de batería con diferentes características técnicas. Esto permitiría una validación más exhaustiva y una mayor confianza en la capacidad del sistema para adaptarse y simular con precisión diversos sistemas de almacenamiento de energía.

4.8. Pruebas de fiabilidad

A lo largo de esta sección se ha hablado de que en las diferentes pruebas realizadas para comprobar el correcto funcionamiento de la herramienta, se han realizado varios test de errores que cubren todos los posibles fallos descritos en la sección 3.4.4. A continuación, se va a explicar en que consisten estos test de fiabilidad.

- **En la entrada de datos**: Se han realizado multitud de pruebas de tipo y formato de datos para asegurar que cuando cualquier entrada por teclado no es del tipo o formato adecuados, la herramienta lo valide correctamente, introduciendo a propósito entradas invalidas. Del mismo modo, se han introducido múltiples valores fuera de los límites definidos en el sistema para comprobar que, efectivamente, la aplicación maneja correctamente estas entradas.
- **En la comunicación entre dispositivos**: Para confirmar que el sistema es capaz de aplicar correctamente los mecanismos de reintentos y recuperación en los fallos de comunicación, se han realizado diversas pruebas en las que se interrumpía deliberadamente la comunicación de uno de los dispositivos involucrados en la funcionalidad que se estuviera ejecutando.
- **En el cumplimiento de requisitos y restricciones**: Para verificar que tanto los requisitos de usuario, como los límites físicos de los dispositivos se respetan en todo momento de la ejecución del sistema, se han realizado una serie de pruebas consistentes en definir, de manera momentánea, unos requisitos y restricciones mucho menores que los reales, de forma que, se ejecute el programa comprobando que el sistema toma las acciones correspondientes en cada caso sin poner en peligro la integridad del mismo.

  Finalmente, para el caso especial de la limitación de la intensidad en las simulaciones de FV (punto 3.4.4), se han introducido curvas donde se cumplen las condiciones para que salte el sistema de seguridad que limita la intensidad de alguno de los dispositivos para evitar dañar la batería, manipulando el código fuente para hacer creer al sistema que la batería esta en un SOC límite, cuando en realidad esta entorno al 50 %, y así realizar la prueba con total seguridad.

5. Resumen del presupuesto

A continuación, en la tabla 5.1, se muestra el resumen del presupuesto del proyecto, el cual asciende a un total de 17229.99 euros.

	PEM	**IVA (21 %)**	**Total**
Presupuesto	14239.66	2990.33	17229.99

Tabla 5.1: Resumen del presupuesto en euros

6. Estudio de viabilidad

Por la propia naturaleza del proyecto desarrollado únicamente se presenta un estudio de viabilidad técnica.

La viabilidad técnica del proyecto ha sido ampliamente demostrada a través del desarrollo y las pruebas realizadas. La herramienta para ensayos con baterías y su aplicación a la simulación de sistemas fotovoltaicos ha mostrado un desempeño sólido y consistente, cumpliendo con los requisitos y restricciones establecidos.

En primer lugar, la herramienta ha demostrado ser capaz de realizar ensayos con baterías y de simular de manera precisa y eficiente el comportamiento de los sistemas fotovoltaicos en combinación con baterías. Los resultados obtenidos de los ensayos y simulaciones han mostrado una concordancia satisfactoria con datos y modelos de referencia, validando la capacidad del sistema para proporcionar resultados confiables y representativos.

Además, la herramienta ha demostrado ser adaptable y compatible con diferentes modelos de sistemas de almacenamiento de energía. Esto se ha comprobado mediante pruebas con distintas celdas de batería, donde la herramienta ha sido capaz de manejar y simular correctamente las características y parámetros específicos de cada modelo. Esta capacidad de adaptación confirma la versatilidad de la herramienta y su capacidad para ser utilizada en una amplia gama de aplicaciones y configuraciones de sistemas de almacenamiento.

La eficiencia computacional de la herramienta también ha sido evaluada y ha arrojado resultados satisfactorios. Durante las pruebas, se ha observado que la herramienta puede llevar a cabo simulaciones complejas en un tiempo razonable, lo que indica que su implementación es viable desde el punto de vista computacional.

En términos de usabilidad, aunque la interfaz actual de la herramienta es una aplicación de consola, se ha demostrado que es funcional y fácil de usar. Sin embargo, se reconoce la necesidad de una mejora en la interfaz para hacerla más amigable y visualmente atractiva, lo que permitiría una experiencia de usuario más intuitiva y agradable.

En resumen, la viabilidad técnica del proyecto ha sido confirmada a través del desarrollo y las pruebas realizadas. La herramienta para ensayos con baterías y su aplicación a la simulación de sistemas fotovoltaicos ha demostrado ser capaz, eficiente y adaptable, cumpliendo con los objetivos y requisitos establecidos. Los resultados obtenidos respaldan la capacidad de la herramienta para proporcionar ensayos y simulaciones precisas y confiables, lo que la convierte en una solución tecnológicamente viable para el análisis de baterías y el diseño de sistemas de energía solar con almacenamiento de energía.

7. Conclusiones

En el presente trabajo, se ha abordado el desarrollo de una herramienta para ensayos con baterías y su aplicación a la simulación de sistemas fotovoltaicos con almacenamiento de energía. A lo largo de este estudio, se han explorado diferentes aspectos relacionados con la implementación y utilización de la herramienta, con el objetivo de analizar su funcionalidad, rendimiento y aplicabilidad en el campo de las energías renovables. En esta conclusión, se presentarán las principales conclusiones técnicas derivadas de la investigación, así como las mejoras identificadas para el proyecto, el futuro potencial de la herramienta y las conclusiones personales obtenidas durante el desarrollo del proyecto.

7.1. Conclusiones técnicas

Durante el desarrollo de este proyecto, se ha demostrado que la herramienta es una solución eficaz para el análisis y evaluación de sistemas energéticos basados en energías renovables. Las pruebas realizadas han confirmado la funcionalidad y precisión de la herramienta, permitiendo obtener resultados consistentes y fiables en diferentes escenarios de estudio.

Además, se ha observado que la herramienta proporciona una interfaz intuitiva y de fácil uso, lo que la hace accesible tanto para expertos en el campo de la ingeniería y la energía solar como para aquellos usuarios menos familiarizados con la tecnología. Esto la convierte en una herramienta versátil y adaptable a diferentes entornos de aplicación.

En términos de rendimiento, se ha constatado que la herramienta es capaz de realizar ensayos y de simular sistemas fotovoltaicos complejos en tiempos razonables, lo que la hace adecuada para su utilización en proyectos de investigación, diseño y optimización de sistemas energéticos. Además, se ha comprobado su capacidad para generar gráficos visuales que facilitan la interpretación de los resultados y la toma de decisiones informadas.

7.2. Mejoras del proyecto

A pesar de los avances logrados, se han identificado algunas áreas en las que se podrían realizar mejoras significativas en el proyecto para aumentar su eficiencia, usabilidad y seguridad. En primer lugar, sería beneficioso crear una base de datos de baterías utilizada por la herramienta, con el fin de abarcar una gran variedad de modelos y fabricantes. Esto permitiría una mayor precisión en las simulaciones y un mayor grado de personalización de los sistemas estudiados.

Asimismo, se podría explorar la posibilidad de integrar la herramienta con tecnologías de inteligencia artificial y aprendizaje automático, con el objetivo de optimizar los algoritmos utilizados y mejorar la predicción del rendimiento energético. Estas técnicas podrían ayudar a identificar patrones y tendencias en los datos de entrada, así como a

ajustar automáticamente los parámetros de simulación para maximizar la eficiencia y el rendimiento de los sistemas fotovoltaicos.

Actualmente, la herramienta utiliza una interfaz de consola, que aunque es simple y intuitiva, puede resultar engorrosa para algunos usuarios. Por tanto, una mejora importante sería implementar una interfaz visual más amigable para el usuario, transformando la herramienta en una aplicación de escritorio. Esto permitiría una interacción más intuitiva y cómoda, con elementos gráficos, menús desplegables y opciones de arrastrar y soltar. Una interfaz más visual facilitaría la configuración de los ensayos, la visualización de los resultados y el manejo de los datos.

En la versión actual de la herramienta, los resultados de los ensayos y las simulaciones se presentan en gráficas una vez que su ejecución ha finalizado. Sin embargo, sería interesante incorporar la capacidad de visualizar estas gráficas durante la propia ejecución de los ensayos. Esto permitiría a los usuarios monitorear y analizar los datos en tiempo real, lo que facilitaría la detección temprana de posibles problemas o anomalías en el sistema. La visualización en tiempo real mejoraría la eficiencia de los ensayos y proporcionaría una retroalimentación inmediata a los usuarios.

Por otro lado, el sistema no genera ningún archivo de logs. Sería recomendable implementar un sistema de generación de logs para registrar eventos importantes durante la ejecución de los ensayos. Estos registros proporcionarían información detallada sobre el progreso de los ensayos, posibles errores o advertencias, y podrían ser útiles para el análisis posterior y la depuración de problemas. La generación de archivos de logs contribuiría a mejorar la trazabilidad y la capacidad de diagnóstico del sistema.

Uno de los puntos clave de la herramienta es la comunicación entre los distintos dispositivos involucrados. Sin embargo, no se ha implementado ningún sistema de seguridad en la comunicación, lo que podría dar lugar a posibles errores de vulnerabilidad. Sería fundamental introducir mecanismos de seguridad, como la encriptación de los datos transmitidos o el uso de protocolos seguros, tanto para garantizar la integridad de la información intercambiada entre los dispositivos. Estas mejoras en la seguridad de la comunicación brindarían una mayor protección contra posibles ataques y mitigarían los riesgos de manipulación o filtración de datos sensibles.

En conjunto, estas mejoras podrían elevar la herramienta a un nivel superior de eficiencia, usabilidad y confiabilidad, y aumentar su potencial de aplicación en el campo de los sistemas fotovoltaicos y el almacenamiento de la energía.

7.3. Futuro del proyecto

El proyecto presenta un gran potencial para futuras investigaciones y desarrollos. En primer lugar, se podrían realizar estudios adicionales para evaluar la viabilidad económica de los sistemas fotovoltaicos simulados con la herramienta, considerando los costos de instalación, mantenimiento y vida útil de los componentes. Esto permitiría a los usuarios tomar decisiones informadas sobre la implementación de sistemas de energía solar con almacenamiento en diferentes contextos.

Además, sería interesante que el proyecto explorara la integración de tecnologías emergentes, como la Internet de las Cosas (IoT) y la computación en la nube. La combinación

de la herramienta con sensores IoT permitiría la recopilación de datos en tiempo real de los sistemas fotovoltaicos en funcionamiento, mejorando así la precisión de las simulaciones y la capacidad de monitorización. Además, la integración con plataformas en la nube permitiría el acceso remoto y la colaboración entre diferentes usuarios, facilitando el intercambio de información y conocimientos en el campo de las energías renovables.

Finalmente, una etapa crucial en el futuro del proyecto sería la validación experimental de la herramienta a través de pruebas en entornos reales. Esto permitiría comparar los resultados obtenidos mediante simulación con datos medidos, lo que proporcionaría una validación adicional y aumentaría la confianza en los resultados simulados. Además, la colaboración con la industria energética y los actores clave en el campo de las energías renovables sería fundamental para garantizar que la herramienta responda a las necesidades y desafíos reales de la industria, y para fomentar su adopción y aplicabilidad en el sector.

7.4. Conclusiones personales

El desarrollo de esta herramienta ha sido una experiencia muy enriquecedora y gratificante a nivel personal. A través de la investigación y el desarrollo de la herramienta he podido adquirir un profundo conocimiento sobre las energías renovables, los sistemas de almacenamiento de energía y la programación de herramientas de simulación con múltiples dispositivos.

Este proyecto me ha permitido comprender la importancia de la eficiencia energética y la utilización de fuentes renovables en la actualidad, así como las oportunidades y desafíos asociados con su implementación. Además, he desarrollado habilidades técnicas y de investigación que considero fundamentales para mi futura carrera como ingeniero.

En resumen, este trabajo ha demostrado la relevancia y utilidad de la herramienta desarrollada en el campo de la investigación de los sistemas de almacenamiento y las energías renovables. Las pruebas y los resultados obtenidos respaldan su funcionalidad y eficacia, mientras que las mejoras identificadas y el futuro potencial del proyecto indican que hay espacio para el crecimiento y la innovación continua en este campo. Personalmente, esta experiencia ha sido fundamental para mi formación académica y me ha inspirado a seguir explorando y contribuyendo al desarrollo de soluciones tecnológicas sostenibles.

Parte II

Presupuesto

Índice

8. Presupuesto

En esta parte de la memoria se muestra el presupuesto total del proyecto desglosado:

Descripción de equipos	**Ud.**	**Precio/Ud.**	**Subtotal**	**IVA (21 %)**	**Total**
Fuente de alimentación SL600-10 de Magna-Power Electronics	1	4182.47	4182.47	878.32	5060.79
3 Módulos de carga 3A300-04 de APS + unidad central 34M04 Mainframe	1	4262.39	4262.39	895.10	5157.49
Batería de la moto Next NX1	1	825.62	825.62	179.38	999.00
Celda Panasonic PA-L154.K01	1	7.00	7.00	1.47	8.47
Arduino UNO R3	1	23.95	23.95	5.029	28.98
Módulo ADC DFRobot I2C ADS1115	1	16.63	16.63	3.49	20.12
Estudio y desarrollo de la herramienta	320	15.38	4921.6	1033.54	5955.14
PEM			14239.66		
IVA (21 %)				2990.33	
TOTAL					17229.99

Tabla 8.1: Presupuesto en euros

Como se puede observar en la tabla 8.1, el presupuesto suma un total de 17229.99 euros.

Parte III

Pliego de condiciones

Índice

9. Introducción

El presente pliego de condiciones establece los requisitos y condiciones generales, técnicas y operativas para el desarrollo y funcionamiento de la herramienta de ensayos con baterías y su aplicación a la simulación de sistemas fotovoltaicos con almacenamiento. El objetivo principal es definir los parámetros necesarios para garantizar el correcto desempeño y cumplimiento de los objetivos del proyecto.

10. Condiciones generales

10.1. Prescripciones generales

- Se establece que el proyecto deberá cumplir con todas las normativas, leyes y regulaciones vigentes en el ámbito de la ingeniería [2], [3] y [4]. Asimismo, se deben seguir los principios éticos y de integridad en la investigación y desarrollo del proyecto.
- Para alcanzar los resultados exactos, se deben cumplir los criterios recogidos en los apartados de este Pliego de Condiciones.
- El correcto funcionamiento de la aplicación viene condicionado por los equipos y dispositivos utilizados y por lo tanto es necesario cumplir con las especificaciones técnicas que estos exigen en sus respectivos manuales.

10.2. Lugar de trabajo

El desarrollo del proyecto se llevará a cabo en un entorno de laboratorio o en un espacio de trabajo adecuado, equipado con los recursos necesarios para la programación, montaje, simulación y pruebas de la herramienta.

También es de suma importancia en la manipulación de elementos electrónicos, que el espacio de trabajo sea un lugar limpio, completamente seco y con buena iluminación. Además, se aconseja ubicar, tanto los dispositivos electrónicos, como los equipos eléctricos en un lugar fijo, estable y estático, alejado de cualquier otro componente evitando así que estos puedan ponerse en contacto y provocar daños.

11. Condiciones técnicas y operativas

11.1. Requisitos de entrada y salida de datos

- **Formato de las curvas de entrada**: fichero *csv* con el formato especificado en el anexo 15.1 para curvas de irradiancia, fichero *csv* con el formato especificado en el anexo 15.2 para curva de consumo, fichero *xlsx* con el formato especificado en el anexo 15.3 para perfiles de intensidad.
- **Unidades de los datos de entrada**: Intensidad (A), tensión (V), tiempo (s), SOC (%), capacidad (Ah), resistencia (Ω).
- **Ubicación de los ficheros de entrada**: Directorio input_files
- **Ubicación de los ficheros de salida**: Directorio output_files.
- **Permisos de escritura y lectura**: Tanto el directorio desde el que se leen los datos de entrada como el de salida de los resultados deben ser accesibles para la aplicación.
- **Tratamiento de la salida**: Los ficheros de salida cuentan con un formato y estructura establecidos y para interpretarlos adecuadamente se debe conocer.

11.2. Software

- Es necesario tener instalado en la computadora donde se vaya a ejecutar el la aplicación Python con la versión 3.8 (es posible que también funcione correctamente con algunas versiones anteriores o posteriores pero no se asegura).
- Es necesario tener instalado Arduino-IDE, en el caso que se quiera cargar el programa de lectura de tensión en el Arduino.
- Es necesario tener instalada una terminal para poder ejecutar la aplicación.
- Se recomienda usar Windows 7 o superior como sistema operativo, pero también es totalmente viable usar cualquier distribución de Linux o MAC OS.

11.3. Hardware

1. **Computadora** Especificaciones mínimas:
   - **Procesador**: Cualquier Intel o AMD de x64 bits
   - **Disco**: 2 GB libres
   - **RAM**: 4 GB
   - **GPU**: No requerida

2. **Microcontrolador**: Arduino UNO R3
3. **Convertidor analógico digital (ADC)**: Módulo ADC DFRobot I2C ADS1115
4. **Fuente de alimentación programable**: DC Power Supply SL600-10 de Magna-Power Electronics
5. **Carga programable**: De 1 a 3 módulos DC & AC Load 3A300-04 más la unidad central 34M04 Mainframe de APS (Adaptive Power Systems)
6. **Sistema de almacenamiento de energía eléctrica**: Cualquier batería o celda cuyas características permitan ser cargadas y descargadas por la fuente y la carga, descritas anteriormente.

11.4. Prescripciones para el correcto funcionamiento de la herramienta

- Las entradas o salidas de tensión y intensidad nunca deben ser sometidas a valores superiores a los marcados en las especificaciones de este proyecto (ver sección 3.2.1), pudiendo dañar severamente los distintos dispositivo si esto ocurriera. Por tanto, hay que tener especial cuidado cuando se modifique el archivo de configuración de no establecer valores por fuera de los límites físicos de los equipos que componen la herramienta.
- El montaje eléctrico tanto del sistema de medida, como de los equipos involucrados en una determinada funcionalidad de la herramienta es fundamental para el funcionamiento correcto del sistema, ya que un error de este tipo podría suponer poner en riesgo todo el sistema y provocar daños irreparables.

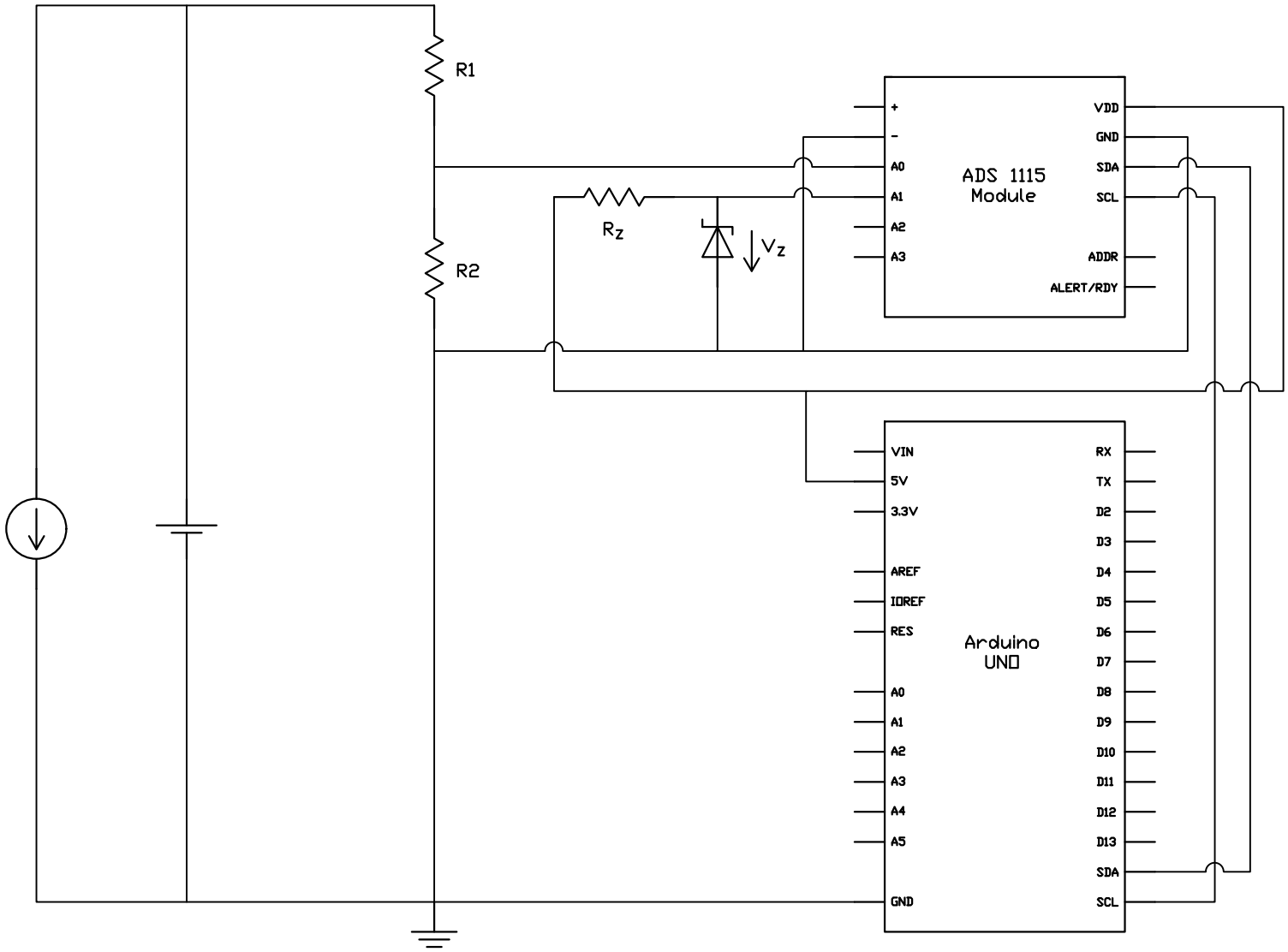
Parte IV

Planos

Índice

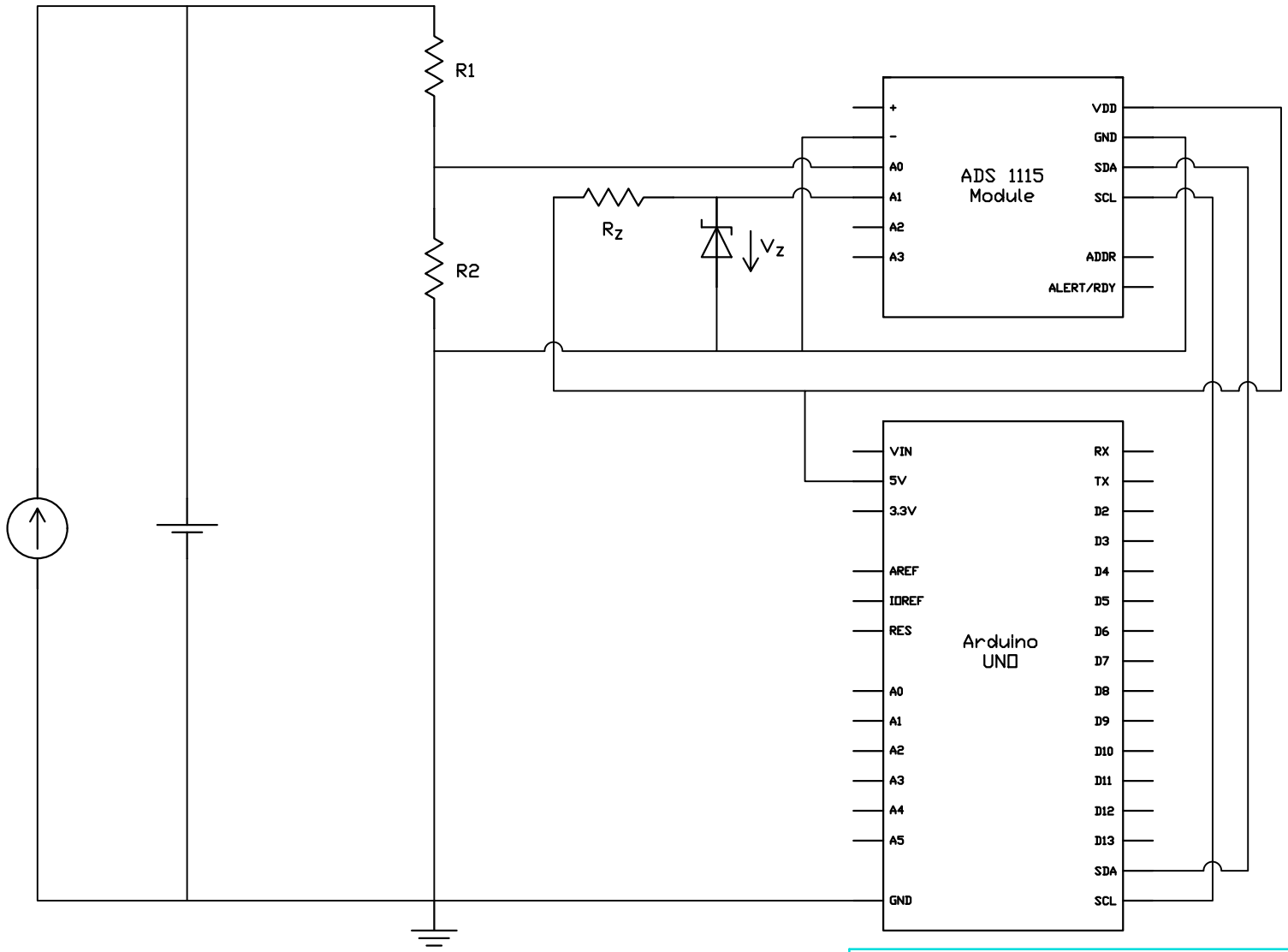
12. Planos

12.1. Esquema unifilar para descargas



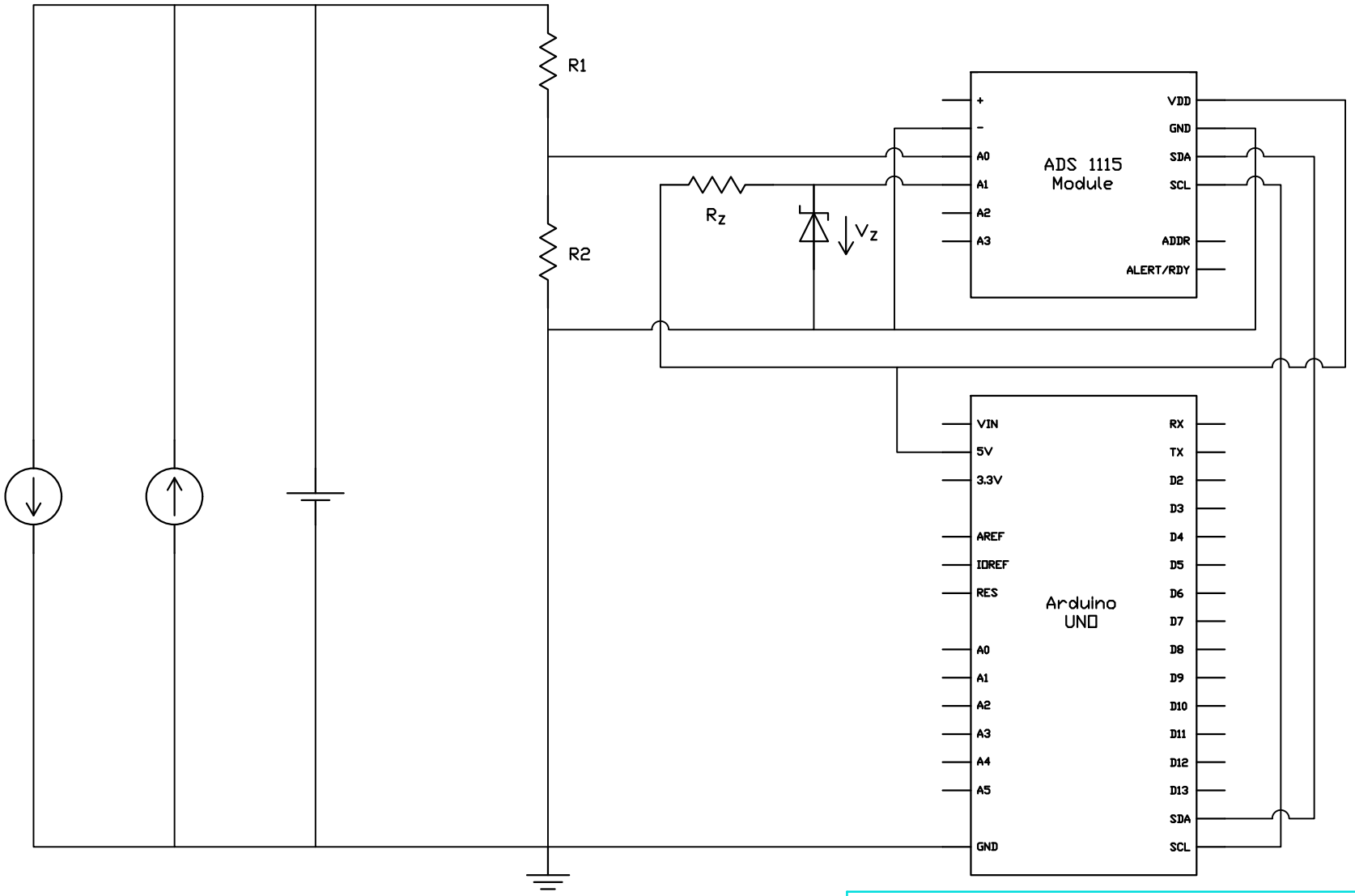
PROYECTO	DESARROLLO Y PROGRAMACIÓN DE UNA HERRAMIENTA PARA ENSAYO DE BATERÍAS Y APLICACIÓN A SIMULACIÓN DE SISTEMAS FOTOVOLTAICOS CON ALMACENAMIENTO.				
CONTENIDO	ESQUEMA UNIFILAR ENSAYO DE DESCARGA				Nº DE PLANO 1
UBICACIÓN UNIVERSITAT JAUME I		FORMATO A4	IDIOMA es	HOJA 1/1	
AUTOR JAVIER GONZÁLEZ BARREDA				ESCALA	FECHA JUL-2023

12.2. Esquema unifilar para cargas



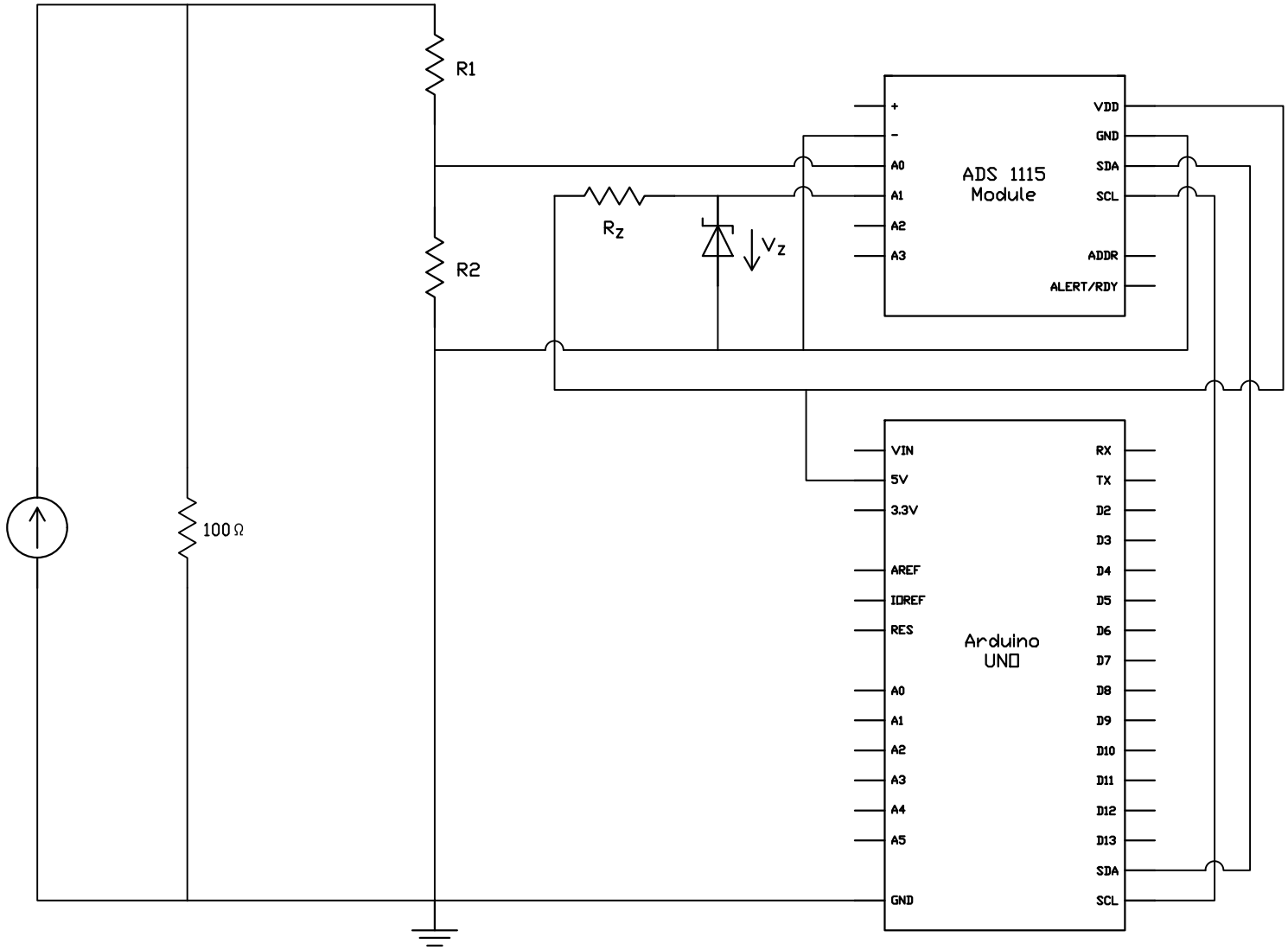
PROYECTO	DESARROLLO Y PROGRAMACIÓN DE UNA HERRAMIENTA PARA ENSAYO DE BATERÍAS Y APLICACIÓN A SIMULACIÓN DE SISTEMAS FOTOVOLTAICOS CON ALMACENAMIENTO.				
CONTENIDO	ESQUEMA UNIFILAR ENSAYO DE CARGA				Nº DE PLANO 2
UBICACIÓN	UNIVERSITAT JAUME I	FORMATO A4	IDIOMA es	HOJA 1/1	
AUTOR	JAVIER GONZÁLEZ BARREDA			ESCALA	FECHA JUL-2023

12.3. Esquema unifilar para simulaciones de FV



PROYECTO	DESARROLLO Y PROGRAMACIÓN DE UNA HERRAMIENTA PARA ENSAYO DE BATERÍAS Y APLICACIÓN A SIMULACIÓN DE SISTEMAS FOTOVOLTAICOS CON ALMACENAMIENTO.				
CONTENIDO	ESQUEMA UNIFILAR SIMULACIÓN FV				Nº DE PLANO 3
UBICACIÓN UNIVERSITAT JAUME I		FORMATO A4	IDIOMA es	HOJA 1/1	
AUTOR JAVIER GONZÁLEZ BARREDA				ESCALA	FECHA JUL-2023

12.4. Esquema unifilar para la calibración de la fuente de alimentación



PROYECTO	DESARROLLO Y PROGRAMACIÓN DE UNA HERRAMIENTA PARA ENSAYO DE BATERÍAS Y APLICACIÓN A SIMULACIÓN DE SISTEMAS FOTOVOLTAICOS CON ALMACENAMIENTO.				
CONTENIDO	ESQUEMA UNIFILAR CALIBRACIÓN FUENTE				Nº DE PLANO 4
UBICACIÓN	UNIVERSITAT JAUME I	FORMATO A4	IDIOMA es	HOJA 1/1	
AUTOR	JAVIER GONZÁLEZ BARREDA			ESCALA	FECHA JUL-2023

Parte V

Anexos

Índice

13. Fichas técnicas

13.1. Ficha técnica de la fuente de alimentación

SL Series

1U Programmable DC Power Supply • Rugged Current-Fed Power Processing

Overview

The SL Series builds on nearly 40 years of power supply innovation at Magna-Power, designed from the ground up to meet the highly reliable power dense demands of ATE system integrators through Magna-Power's signature current-fed power processing topology. Utilizing state-of-the-art semiconductors and innovative internally designed and manufactured heat sinks, the SL Series offers industry-leading 1U (1.75" height) programmable power levels with models at 1.5 kW, 2.6 kW, 4 kW, 6 kW, 8 kW, and 10 kW, while still maintaining an ambient operating temperature rating up to 50°C.

Models

	1.5 kW	2.6 kW	4 kW	6 kW	8 kW	10 kW		
Max Voltage (Vdc)	**Max Current (Adc)**						**Ripple[1] (mVrms)**	**Efficiency**
5	250	N/A	N/A	N/A	N/A	N/A	50	84%
10	150	250	N/A	N/A	N/A	N/A	40	89%
16	93	162	250	N/A	N/A	N/A	35	89%
20	75	130	200	250	N/A	N/A	40	90%
25	60	104	160	240	N/A	N/A	40	91%
32	46	81	125	186	250	N/A	40	91%
40	37	65	100	150	200	250	40	91%
50	30	52	80	120	160	200	50	92%
60	25	43	66	100	133	166	60	93%
80	18	32	50	75	100	125	60	93%
100	15	26	40	60	80	100	60	93%
125	12	20	32	48	64	80	100	93%
160	9	16	25	36	50	60	120	93%
200	7.5	13	20	30	40	50	125	94%
250	6	10.4	16	24	32	40	130	94%
300	5	8.6	13.2	20	26.4	33.3	160	94%
375	4	6.9	10.4	16	21.3	26.5	170	94%
400	3.7	6.5	10	15	20	25	180	95%
500	3	5.2	8	12	16	20	220	95%
600	2.5	4.3	6.4	10	13.3	16.5	250	95%
800	1.8	3.2	5	7.5	10	12.5	300	95%
1000	1.5	2.6	4	6	8	10	350	95%
1250	1.2	2	3.2	4.8	6.4	8	375	95%
1500	1	1.7	2.6	4	5.3	6.6	400	95%

AC Input Voltage (Vac)	**Input Current Per Phase (Aac)**					
UI (85-265 Vac, 1Φ)	21-7	N/A	N/A	N/A	N/A	N/A
UI2 (187-265 Vac, 1Φ)	N/A	16-12	N/A	N/A	N/A	N/A
208/240 Vac, 3Φ	6	11	16	24	32	39
380/415 Vac, 3Φ	5	8	11	16	19	22
440/480 Vac, 3Φ	4	6	9	14	17	19

Key Features

- SCPI Remote Programming API
- High Accuracy Measurements
- Master-Slave Functionality
- Remote Sensing
- 37-Pin External User I/O
- RS232 Interface
- Ethernet and GPIB Available
- 0-10V External Analog Inputs
- Programmable Protection Limits
- Fast Transient Response
- Remote Interface Software
- NI LabVIEW™ and IVI Driver
- Interlock Shutdown Input
- Designed and Manufactured in the USA

Available Options

- High Slew Rate Output (+HS)
- IEEE-488 GPIB Communications (+GPIB)
- LXI TCP/IP Ethernet Communications (+LXI)
- Ruggedized (+RUG)

SL Series Model Ordering Guide

There are 127 different models in the SL Series spanning power levels: 1.5 kW, 2.6 kW, 4 kW, 6 kW, 8 kW, 10 kW.

To determine the appropriate model:

1. Select the desired Max Voltage (Vdc) from the left-most column.
2. Select the desired Max Current (Adc) from the same row that contains your desired Max Voltage.
3. Construct your model number according to the model ordering guide.

SL 800-10/208+HS+LXI

Product Series

Max Voltage

Max Current

Front Panel Type
(Empty): Standard
C: Blank
Note: Computer control and 37-pin isolated I/O included with all front panel types

Input Voltage
UI: 85 - 265 1Φ (1.5 kW Models)
UI2: 187 - 265 1Φ (2.6 kW Models)
208: 208 Vac 3Φ
240: 240 Vac 3Φ
380: 380/400 Vac 3Φ
415: 415 Vac 3Φ
440: 440 Vac 3Φ
480: 480 Vac 3Φ

Option Codes

[1] Ripple specified for standard models. Ripple will be higher for models with the High Slew Rate Output (+HS). Refer to option page for more details.

Specifications

AC Input Specifications

1Φ AC Input Voltage 1Φ, 2-wire + ground	85 to 265 Vac (UI: Universal input; 1.5 kW Models) 187 to 265 Vac (UI2: Universal input; 2.6 kW Models)
3Φ AC Input Voltage 3Φ, 3-wire + ground Available on all models	208 Vac (operating range 187 to 229 Vac) 240 Vac (operating range 216 to 264 Vac) 380/400 Vac (operating range 342 to 440 Vac) 415 Vac (operating range 373 to 456 Vac) 440 Vac (operating range 396 to 484 Vac) 480 Vac (operating range 432 to 528 Vac)
AC Input Current	Refer to chart of available models
AC Input Frequency	50-400 Hz
Power Factor	0.99 at max power; models with 1Φ AC input >0.82 at max power; models with 3Φ AC input
AC Input Isolation	±2500 Vac, maximum input voltage to ground

Output Specifications

Voltage Ripple	Refer to chart of available models.
Line Regulation	Voltage mode: ± 0.004% of full scale Current mode: ± 0.02% of full scale
Load Regulation	Voltage mode: ± 0.01% of full scale Current mode: ± 0.04% of full scale
Load Transient Response	2 ms to recover within ±1% of regulated output with a 50% to 100% or 100% to 50% step load change
Stability	± 0.10% for 8 hrs. after 30 min. warm-up
Efficiency	84% to 95%; refer to chart of available models
DC Output Isolation Models Rated ≤1000 Vdc	±1000 Vdc, max output voltage to ground
DC Output Isolation Models Rated >1000 Vdc	±1500 Vdc, max output voltage to ground

Programming Specifications

Programming Accuracy	Voltage: ±0.075% of max voltage rating Current: ±0.075% of max current rating
Measurement Accuracy	Voltage: ±0.2% of max voltage rating Current: ±0.2% of max current rating
Maximum Slew Rate Standard Models	100 ms, output voltage change from 0 to 63% 100 ms, output current change from 0 to 63%
Maximum Slew Rate Models with High Slew Rate Option (+HS)	4 ms, output voltage change from 0 to 63% 8 ms, output current change from 0 to 63%
Trip Settings Range	Over Voltage: 10% to 110% max voltage rating Over Current: 10% to 110% max current rating
Computer Command Protocol	Standard Commands for Programmable Instruments (SCPI)
Remote Sense Limits Wired; Available on Models Rated ≤1000 Vdc	3% maximum voltage drop from output to load

Connectivity Specifications

Communication Interfaces (Standard)	RS232: DB-9, Female External User I/O: DB-37, Female
Communication Interfaces (Optional)	LXI TCP/IP Ethernet: RJ-45 GPIB: IEEE-488

External User I/O Specifications

Digital Inputs	5 V, 10 kΩ impedance
Digital Monitoring Signals	5 V, 5 mA capacity
Digital Reference Signal	5 V output, 25 mA capacity
Analog Programming Input	0-10 V
Analog Programming Impedance	10 kΩ
Analog Monitoring Signals	0-10 V, 5 mA capacity
Analog Monitoring Impedance	100 Ω
Analog Monitoring Accuracy	0.2% of max rating
Analog Reference Signal	10 V, 5 mA capacity, 1 Ω impedance

Physical Specifications

Racking Standard	EIA-310
Rear Support Rails	Included

Power Level	Rack Units	Size	Weight
1.5 kW	1U	1.75" H x 19" W x 24" D (4.4 x 48.3 x 61.0 cm)	32 lbs (14.52 kg)
2.6 kW	1U	1.75" H x 19" W x 24" D (4.4 x 48.3 x 61.0 cm)	34 lbs (15.42 kg)
4 kW	1U	1.75" H x 19" W x 24" D (4.4 x 48.3 x 61.0 cm)	35 lbs (15.88 kg)
6 kW	1U	1.75" H x 19" W x 24" D (4.4 x 48.3 x 61.0 cm)	35 lbs (15.88 kg)
8 kW	1U	1.75" H x 19" W x 24" D (4.4 x 48.3 x 61.0 cm)	36 lbs (16.33 kg)
10 kW	1U	1.75" H x 19" W x 24" D (4.4 x 48.3 x 61.0 cm)	37 lbs (16.78 kg)

Environmental Specifications

Ambient Operating Temperature	0°C to 50°C
Storage Temperature	-25°C to +85°C
Humidity	Relative humidity up to 95% non-condensing
Air Flow	Side air inlet, rear exhaust
Temperature Coefficient	0.04%/°C of maximum output voltage 0.06%/°C of maximum output current

Regulatory Compliance

EMC	Complies with 2014/30/EU (EMC Directive) CISPR 22 / EN 55022 Class A
Safety	NRTL Listed, Intertek Control Number 5014353 Conforms to UL61010-1:2012 Ed.3+R:29Apr2016 Certified to CSA C22.2#61010-1-12:2012 Ed.3 Complies with EN61010-1 Complies with 2014/35/EU (Low Voltage Directive)
CE Mark	Yes
RoHS Compliant	Yes

Note: Specifications are subject to change without notice. Input voltage specifications are line-to-line.

13.2. Ficha técnica de la carga

4 Technical Specifications

Technical specifications shown here apply at an ambient temperature of 25° C ± 5°. Refer to V-I curve and Very Low Voltage V-I Curve charts by models for operating envelope.

4.1 Operating Ranges

MODEL	3A060-20		3A150-08		3A300-04	
OPERATING RANGES						
Power Ranges	0 - 300 VA		0 - 300 VA		0 - 300 VA	
Current Ranges	0 -10 Arms	10 -20 Arms	0 - 4 Arms	4 - 8 Arms	0 - 2 Arms	2 - 4 Arms
Voltage Range	10 - 60 Vrms		15 - 150 Vrms		30 - 300 Vrms	
Frequency	DC, 40 - 400Hz (CC Mode) / DC - 400Hz (LIN,CR Mode)					
AC Waveforms	Sine, Square, Step, DC					

4.2 Operating Modes

MODEL	3A060-20		3A150-08		3A300-04	
OPERATING MODES						
CC Mode - High Range	0 -10 Arms	10 -20 Arms	0 - 4 Arms	4 - 8 Arms	0 - 2 Arms	2 - 4 Arms
Resolution	2.5 mA	5 mA	1 mA	2 mA	0.5 mA	1 mA
Accuracy	50Hz & 60Hz: ± 0.5% OF (SETTING + RANGE) / > 60 Hz: ±(0.5% OF SETTING + 1% OF RANGE)					
CC Mode - Low Range	0.000- 1.000 A		0.000- 0.400 A		0.000- 0.200 A	
Accuracy	±2% OF (SETTING + RANGE)					
CC Linear Mode - .	Refer to CC Mode data					
CF Mode - Range	√2 - 3.5 / 1.5 - 1.9 / 3.0 - 3.4					
Resolution	0.5 / 0.1 / 0.1					
Lagging	-0.30 to -0.85 for CF 2.0 to 3.5					
Leading	+0.30 to +0.85 for CF 2.0 to 3.5					
CR Mode Range	0.3 - 1.2 KΩ	12.3 - 4.8 KΩ	1.875 - 7.5 KΩ	7.5 - 30 KΩ	7.5 - 30 KΩ	30 - 120 KΩ
Resolution[1]	0.83 mS	0.2083 mS	0.13 mS	0.033 mS	0.033 mS	0.0083 mS
Accuracy	50Hz & 60Hz: ± 0.5% OF (SETTING + RANGE) / > 60 Hz: ±(0.5% OF SETTING + 2% OF RANGE)					

4.3 Protection Modes

MODEL	3A060-20		3A150-08		3A300-04	
PROTECTION						
Over Power (OP)	315 VA		315 VA		315 VA	
Over Current (OC)	10.5 A	21 A	4.2 A	8.4 A	2.1 A	4.2 A
Over Voltage (OV)	63 V		157.5 V		315 V	
Over Temperature (OT)	+85° C / +185° F					

[1] Note: S = Siemens or mho, unit of conductance. 1S = 1/Ω = A/V

4.4 Power Factor Range

MODEL	3A060-20	3A150-08	3A300-04
POWER FACTOR RANGE			
Lagging PF	CF: √2 to 3.5 . PF: - 0.30 to -0.85		
Leading PF	CF: √2 to 3.5 . PF: + 0.30 to +0.85 or 1.00		

4.5 Metering

MODEL		3A060-20	3A150-08	3A300-04
METERING				
Voltage	Range	0 - 60 V	0 - 150 V	0 - 300 V
	Resolution	0.01 V	0.01 V	0.1 V
	Accuracy	±(0.5% OF SETTING + 0.2% OF RANGE)		
Current	Range	0 - 20 A	0 - 8 A	0 - 4 A
	Resolution	0.01 A	0.001 A	0.001 A
	Accuracy	50Hz & 60Hz: ± 0.5% OF (READING + RANGE) / > 60 Hz: ±(0.5% OF READING + 2% OF RANGE)		
Power	Range	0 - 300 W		
	Resolution	0.1 W		
	Accuracy	50Hz & 60Hz: ± 0.5% OF (READING + RANGE) / > 60 Hz: ±(0.5% OF READING + 2% OF RANGE)		
Apparent Power	Range	0 - 300 VA		
	Resolutions	0.1 VA		
	Accuracy	Derived from Volt and Current Measurement		

4.6 Analog I/O

MODEL	3A060-20	3A150-08	3A300-04
ANALOG I/O			
Current Monitor Out Scale	5 A/V	2 A/V	1 A/V
Range	0 – 4 V full scale	0 – 4 V full scale	0 – 4 V full scale
Accuracy	± 0.5% OF (SETTING + RANGE)		
External Sync In	Optically Isolated Input, TTL level, Zin = 330 Ohm. Requires 50% duty cycle signal-10/330		

4.7 Power & Cooling

MODEL	3A060-20	3A150-08	3A300-04
AC INPUT AND COOLING SPECIFICATIONS			
Power	Supplied by 34M01 or 34M04 mainframe		
Cooling	Supplied by 34M01 or 34M04 mainframe		

4.8 Dimensions & Weight

MODEL	3A060-20	3A150-08	3A300-04
DIMENSIONS AND WEIGHT			
Dimensions (H x W x D)	143 x 108 x 405 mm / 5.6” x 4.25” x 15.9		
Weight (Net)	3.5 kg / 7.7 lbs	3.5 kg / 7.7 lbs	3.5 kg / 7.7 lbs

4.9 Environmental

MODEL	3A060-20	3A150-08	3A300-04
ENVIRONMENTAL			
Operating Temperature	0 - 40° C / 32 - 104° F		
Relative Humidity	80% max. non-condensing		
Environmental	Indoor Use Only, Pollution Degree 2		
Altitude	2000 meter / 6500 feet max. Operating		
EMC & Safety	CE Mark		

13.3. Ficha técnica de las celdas de la batería de la moto

Spec. No.	INR18650-25R	Version No.	1.0	In-Young Jang

1.0. Scope

This product specification has been prepared to specify the rechargeable lithium-ion cell ('cell') to be supplied to the customer by Samsung SDI Co., Ltd.

2.0. Description and model name

2.1 Description lithium-ion rechargeable cell

2.2 Model name INR18650-25R

3.0. Nominal specifications

Item	**Specification**
3.1 Nominal discharge capacity	2,500mAh Charge: 1.25A, 4.20V,CCCV 125mA cut-off, Discharge: 0.2C, 2.5V discharge cut-off
3.2 Nominal voltage	3.6V
3.3 Standard charge	CCCV, 1.25A, 4.20 ± 0.05 V, 125mA cut-off
3.4 Rapid charge	CCCV, 4A, 4.20 ± 0.05 V, 100mA cut-off
3.6 Charging time	Standard charge : 180min / 125mA cut-off Rapid charge: 60min (at 25℃) / 100mA cut-off
3.7 Max. continuous discharge (Continuous)	20A(at 25℃), 60% at 250 cycle
3.8 Discharge cut-off voltage End of discharge	2.5V
3.9 Cell weight	45.0g max
3.10 Cell dimension	Height : 64.85 ± 0.15mm Diameter : 18.33 ± 0.07mm
3.11 Operating temperature (surface temperature)	Charge : 0 to 50℃ (recommended recharge release < 45℃) Discharge: -20 to 75℃ (recommended re-discharge release < 60℃)
3.12 Storage temperature (Recovery 90% after storage)	1.5 year -30~25℃(1*) 3 months -30~45℃(1*) 1 month -30~60℃(1*)

Note (1): If the cell is kept as ex-factory status (50±5% SOC, 25℃),
the capacity recovery rate is more than 90% of 10A discharge capacity
100% is 2,450mAh at 25℃ with SOC 100% after formation.

13.4. Ficha técnica de la celda

PA-L154.K01

Powered by Panasonic CGR-18650

Dimensions

length	71 mm	+2 / -1 mm
diameter	18,5 mm	+1 / -0 mm

Data for pack

Nominal Voltage		3,6V	3,0V - 4,2V
Nominal capacity		2250mAh	typical
Used cell in Pack		1x	Panasonic CGR-18650CG
internal resistance pack		220mOhm	190 - 250mOhm
Max charge voltage		4,2V	
Charge current	standard	430mA	0℃ < T < 45℃
	rapid	1200mA	10℃ < T < 45℃
Discharge	standard	430mA	-20℃ < T < 60℃
	max.cont.	1250mA	-20℃ < T < 45℃
	max.peak	1300mA	-20℃ < T < 45℃
Short circuit current		≈20A	<500µs
NTC		10 kOhm	Tolerance 5%; B-value 25℃/85℃ = 3980K
Connector	JST	XHP-3	Pin 1 : GND black
			Pin 2: NTC yellow
			Pin 3: PLUS red
Cable length		45mm	±5mm
Weight		47g	±2,5g
Lithium content		0,68g	8,1 Wh

Limitations by Safety Unit (SU)

over voltage	cut off	4,3V	±25mV
over voltage	release	4,05V	±100mV
under voltage	cut off	2,25V	±100mV
under voltage	release	3,0V	2,25V – 3,45V
Current limit 1 by SU		1250mA	continuous (typical)
Current limit 2 by SU		>1300mA	< 150ms (typical)
Current limit 3 by SU		≤20A	< 4ms (typical)
Power consumption	active	12,5µA	-0/+7,5µA
	sleep	1,5µA	-0/+1,0µA
ESD Protection		no	

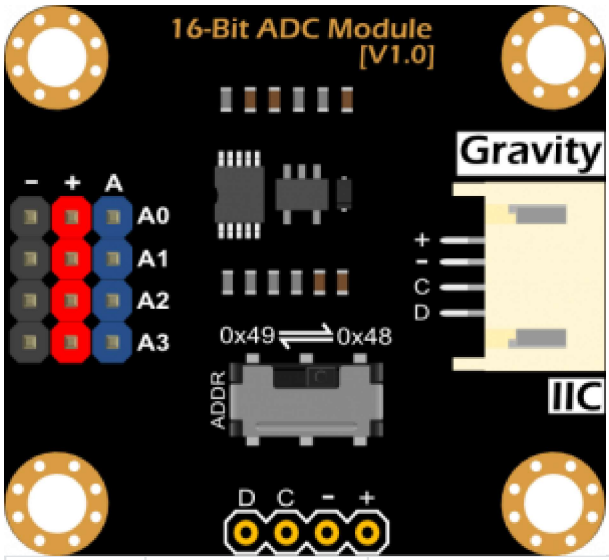
- Constructed by approved Panasonic Pack assembler -

13.5. Ficha técnica del ADC

Specification

- Supply Voltage (VCC): 3.3~5.0V
- Analog Signal Detection Range: 0~VCC
- Number of Analog Channels: 4
- ADC Bits: 16 Bit
- Operating Current: 2~3mA (Not include sensor module)
- Interface Type: Gravity I2C
- Interface Level: High 3.3V, Low 0V
- Product Size: 32mm * 32mm(1.26in*1.26in)

Board Overview



Num	Label	Description
\	VCC	Power VCC(3.3~5.0V)
-	GND	Power GND(0V)
C	SCL	I2C Bus Clock Line
D	SDA	I2C Bus Data Line
A	Analog In	Analog Input Channels: A0, A1, A2, A3
ADDR	I2C Address	I2C address selection switch,I2C address: 0x48, 0x49

Tutorial

⚠	**The voltage on analog input pins must be less than VCC 0.3V! That is, Analog Signal ≤ VCC 0.3V**

13.6. Ficha técnica del Arduino

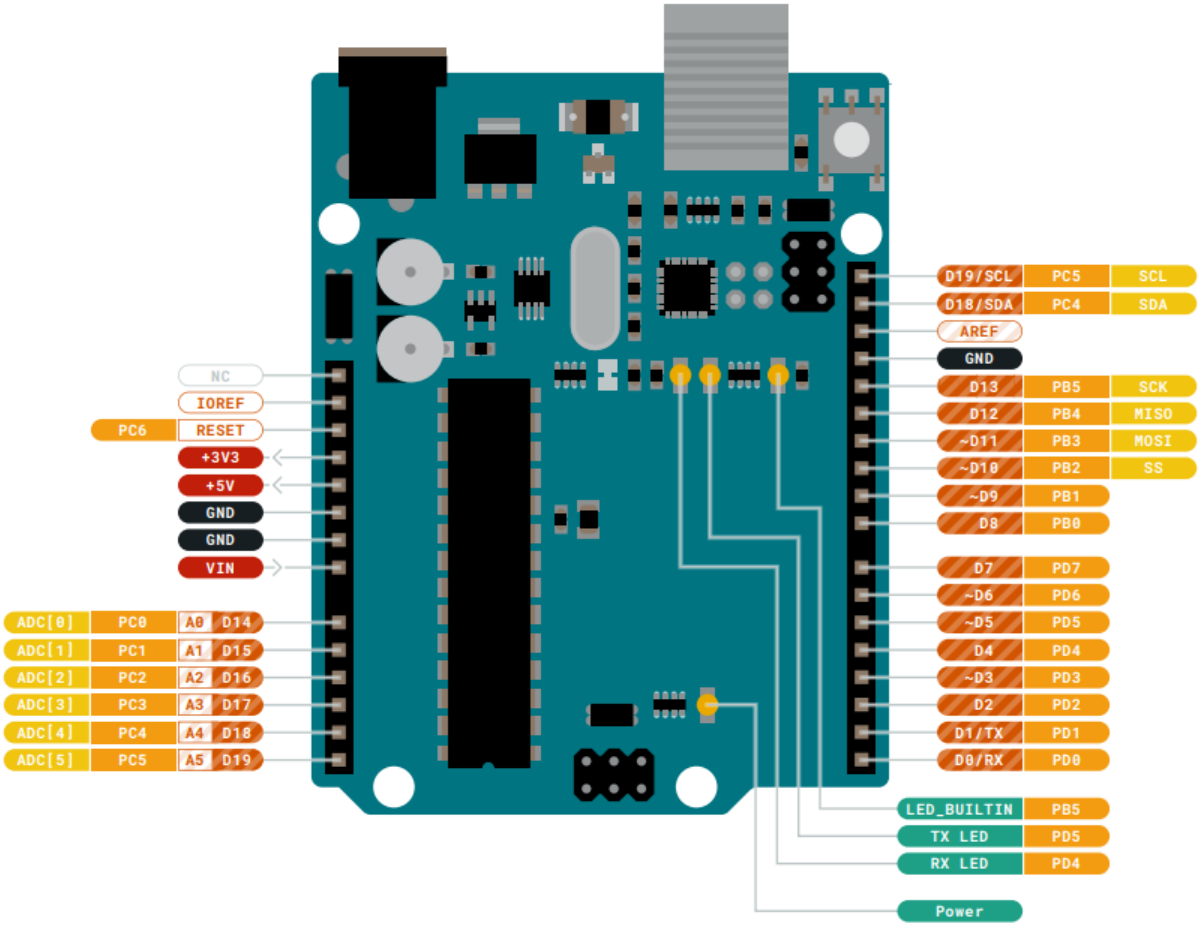
Features

- **ATMega328P** Processor
  - **Memory**
    - AVR CPU at up to 16 MHz
    - 32KB Flash
    - 2KB SRAM
    - 1KB EEPROM
  - **Security**
    - Power On Reset (POR)
    - Brown Out Detection (BOD)
  - **Peripherals**
    - 2x 8-bit Timer/Counter with a dedicated period register and compare channels
    - 1x 16-bit Timer/Counter with a dedicated period register, input capture and compare channels
    - 1x USART with fractional baud rate generator and start-of-frame detection
    - 1x controller/peripheral Serial Peripheral Interface (SPI)
    - 1x Dual mode controller/peripheral I2C
    - 1x Analog Comparator (AC) with a scalable reference input
    - Watchdog Timer with separate on-chip oscillator
    - Six PWM channels
    - Interrupt and wake-up on pin change
  - **ATMega16U2 Processor**
    - 8-bit AVR® RISC-based microcontroller
  - **Memory**
    - 16 KB ISP Flash
    - 512B EEPROM
    - 512B SRAM
    - debugWIRE interface for on-chip debugging and programming
  - **Power**
    - 2.7-5.5 volts

14. Pinouts

14.1. Pinout del Arduino

5 Connector Pinouts



Pinout

5.1 JANALOG

Pin	Function	Type	Description
1	NC	NC	Not connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog/GPIO	Analog input 0 /GPIO
10	A1	Analog/GPIO	Analog input 1 /GPIO
11	A2	Analog/GPIO	Analog input 2 /GPIO
12	A3	Analog/GPIO	Analog input 3 /GPIO
13	A4/SDA	Analog input/I2C	Analog input 4/I2C Data line
14	A5/SCL	Analog input/I2C	Analog input 5/I2C Clock line

5.2 JDIGITAL

Pin	Function	Type	Description
1	D0	Digital/GPIO	Digital pin 0/GPIO
2	D1	Digital/GPIO	Digital pin 1/GPIO
3	D2	Digital/GPIO	Digital pin 2/GPIO
4	D3	Digital/GPIO	Digital pin 3/GPIO
5	D4	Digital/GPIO	Digital pin 4/GPIO
6	D5	Digital/GPIO	Digital pin 5/GPIO
7	D6	Digital/GPIO	Digital pin 6/GPIO
8	D7	Digital/GPIO	Digital pin 7/GPIO
9	D8	Digital/GPIO	Digital pin 8/GPIO
10	D9	Digital/GPIO	Digital pin 9/GPIO
11	SS	Digital	SPI Chip Select
12	MOSI	Digital	SPI1 Main Out Secondary In
13	MISO	Digital	SPI Main In Secondary Out
14	SCK	Digital	SPI serial clock output
15	GND	Power	Ground
16	AREF	Digital	Analog reference voltage
17	A4/SD4	Digital	Analog input 4/I2C Data line (duplicated)
18	A5/SD5	Digital	Analog input 5/I2C Clock line (duplicated)

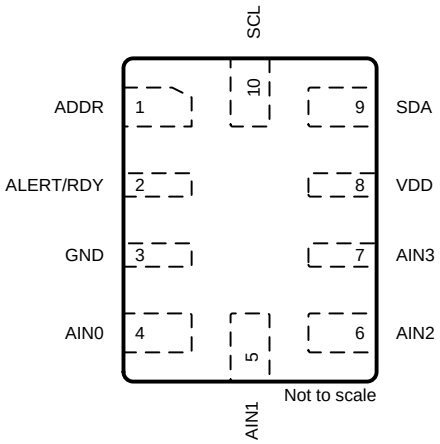
14.2. Pinout del ADC

5 Device Comparison Table

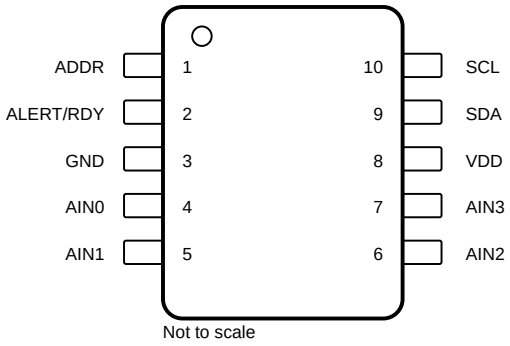
DEVICE	RESOLUTION (Bits)	MAXIMUM SAMPLE RATE (SPS)	INPUT CHANNELS Differential (Single-Ended)	PGA	INTERFACE	SPECIAL FEATURES
ADS1115	16	860	2 (4)	Yes	I^2C	Comparator
ADS1114	16	860	1 (1)	Yes	I^2C	Comparator
ADS1113	16	860	1(1)	No	I^2C	None
ADS1015	12	3300	2 (4)	Yes	I^2C	Comparator
ADS1014	12	3300	1 (1)	Yes	I^2C	Comparator
ADS1013	12	3300	1 (1)	No	I^2C	None
ADS1118	16	860	2 (4)	Yes	SPI	Temperature sensor
ADS1018	12	3300	2 (4)	Yes	SPI	Temperature sensor

6 Pin Configuration and Functions

RUG Package
10-Pin X2QFN
Top View



DGS Package
10-Pin VSSOP
Top View



Pin Functions

PIN (1)				TYPE	DESCRIPTION
NAME	ADS1113	ADS1114	ADS1115		
ADDR	1	1	1	Digital input	I^2C slave address select
AIN0	4	4	4	Analog input	Analog input 0
AIN1	5	5	5	Analog input	Analog input 1
AIN2	—	—	6	Analog input	Analog input 2 (ADS1115 only)
AIN3	—	—	7	Analog input	Analog input 3 (ADS1115 only)
ALERT/RDY	—	2	2	Digital output	Comparator output or conversion ready (ADS1114 and ADS1115 only)
GND	3	3	3	Analog	Ground
NC	2, 6, 7	6, 7	—	—	Not connected
SCL	10	10	10	Digital input	Serial clock input. locks data on SDA
SDA	9	9	9	Digital I/O	Serial data. Transmits and receives data
VDD	8	8	8	Analog	Power supply. Connect a 0.1-μF, power-supply decoupling capacitor to GND.

(1) See the *Unused Inputs and Outputs* section for unused pin connections.

15. Formatos de los ficheros empleados por la herramienta

15.1. Formato del fichero csv con la curva de irradiancia para la simulación de un sistema de FV

time(UTC)	G(i)	Gb(i)	Gd(i)
00:00	0.0	0.0	0.0
01:00	0.0	0.0	0.0
02:00	0.0	0.0	0.0
03:00	0.0	0.0	0.0
04:00	0.0	0.0	0.0
05:00	25.77	0.0	25.24
06:00	96.72	22.16	71.37
07:00	262.26	136.07	119.99
08:00	451.14	281.57	160.33
09:00	620.63	427.86	180.93
10:00	773.38	559.06	200.16
11:00	861.83	638.64	207.71
12:00	891.86	669.34	206.58
13:00	851.64	632.22	204.06
14:00	735.18	529.97	191.61
15:00	584.53	394.17	179.0
16:00	395.87	241.32	146.13
17:00	211.15	99.48	106.29
18:00	61.67	0.44	59.5
19:00	7.05	0.0	6.9
20:00	0.0	0.0	0.0
21:00	0.0	0.0	0.0
22:00	0.0	0.0	0.0
23:00	0.0	0.0	0.0

Figura 15.1: Formato de la curva de irradiancia

15.2. Formato del fichero csv con la curva de consumo para la simulación de un sistema de FV

```
"id_sm,""solo_hora"",""minuto"",""dia"",""media"""
"1,""0"",""0"",""2016-06-01"",""125.344000"""
"1,""0"",""1"",""2016-06-01"",""153.266500"""
"1,""0"",""2"",""2016-06-01"",""152.160500"""
"1,""0"",""3"",""2016-06-01"",""154.413000"""
"1,""0"",""4"",""2016-06-01"",""154.565500"""
"1,""0"",""5"",""2016-06-01"",""154.870000"""
"1,""0"",""6"",""2016-06-01"",""154.756000"""
"1,""0"",""7"",""2016-06-01"",""210.903500"""
"1,""0"",""8"",""2016-06-01"",""240.962500"""
"1,""0"",""9"",""2016-06-01"",""275.789000"""
"1,""0"",""10"",""2016-06-01"",""270.982500"""
"1,""0"",""11"",""2016-06-01"",""270.792000"""
"1,""0"",""12"",""2016-06-01"",""269.915000"""
"1,""0"",""13"",""2016-06-01"",""269.226000"""
"1,""0"",""14"",""2016-06-01"",""268.730000"""
"1,""0"",""15"",""2016-06-01"",""268.578000"""
"1,""0"",""16"",""2016-06-01"",""268.046000"""
"1,""0"",""17"",""2016-06-01"",""268.198000"""
"1,""0"",""18"",""2016-06-01"",""267.284500"""
"1,""0"",""19"",""2016-06-01"",""269.573000"""
"1,""0"",""20"",""2016-06-01"",""181.799500"""
"1,""0"",""21"",""2016-06-01"",""241.800500"""
"1,""0"",""22"",""2016-06-01"",""2142.033000"""
"1,""0"",""23"",""2016-06-01"",""2237.891500"""
"1,""0"",""24"",""2016-06-01"",""2239.378500"""
"1,""0"",""25"",""2016-06-01"",""464.684000"""
"1,""0"",""26"",""2016-06-01"",""265.451500"""
"1,""0"",""27"",""2016-06-01"",""265.033500"""
"1,""0"",""28"",""2016-06-01"",""265.452000"""
"1,""0"",""29"",""2016-06-01"",""265.833500"""
"1,""0"",""30"",""2016-06-01"",""265.757000"""
```

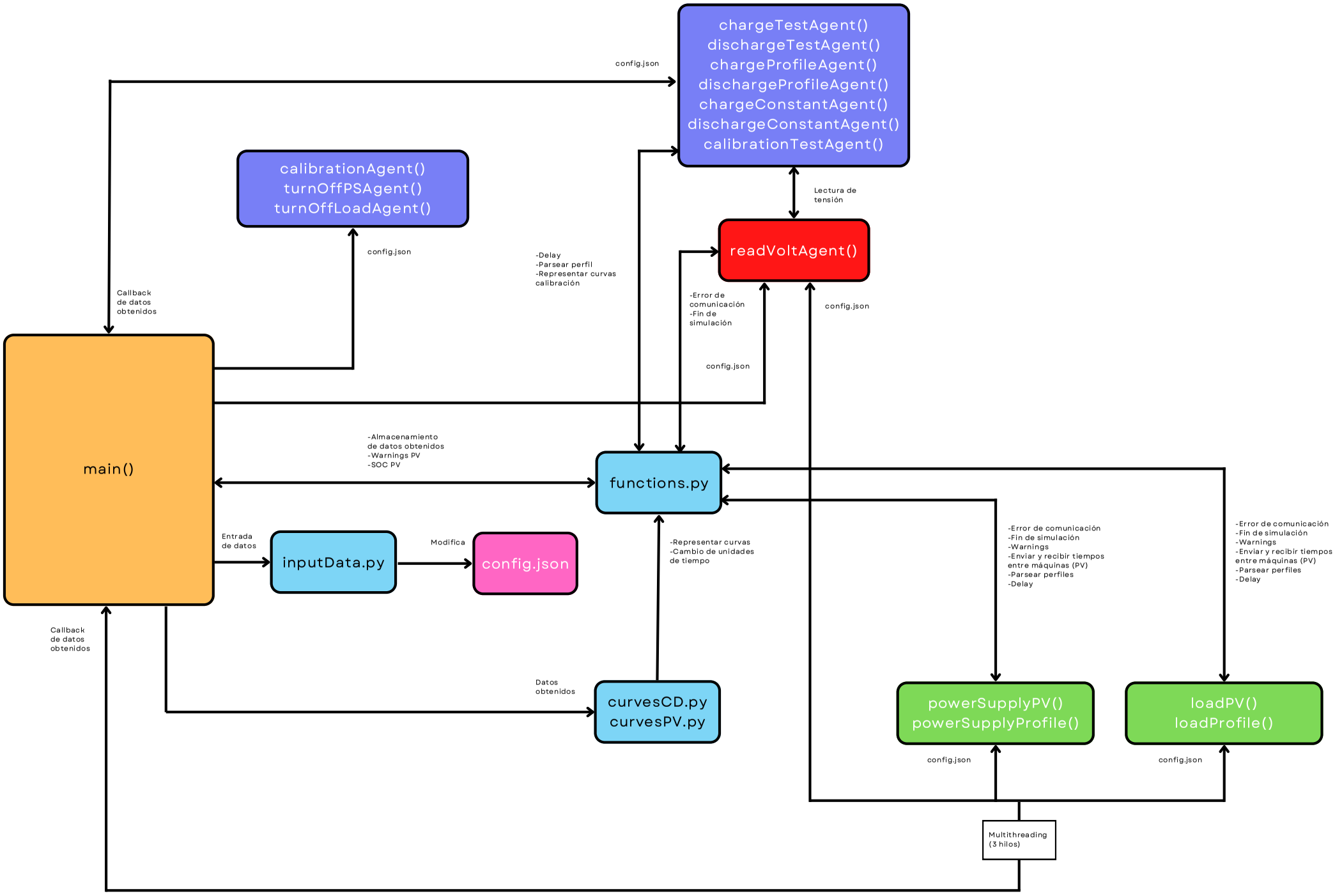
Figura 15.2: Formato de la curva de consumo

15.3. Formato de la hoja de datos (fichero xlsx) para perfiles de intensidad

Intensidad (A)	Tiempo (s)
4	240
2.611960764	240
0.04134585	240
0.104630313	240
0.075097564	240
0.048939985	240
1.002004008	240
3.511022044	240
2.497204936	240
2.565974053	240
1.889252189	240
0.068769117	240
0.048518089	240
0.056534121	240
0.118974792	240
0.122349963	240
0.097036178	240
0.05779981	240
0.058643603	240
0.123615652	240
0.117287206	240
0.120240481	240
0.383081953	240
1.636114334	240

Figura 15.3: Formato de los perfiles de intensidad

16. Diagrama de Clases/Agentes


chargeTestAgent()
dischargeTestAgent()
chargeProfileAgent()
dischargeProfileAgent()
chargeConstantAgent()
dischargeConstantAgent()
calibrationTestAgent()
config.json
calibrationAgent()
turnOffPSAgent()
turnOffLoadAgent()
config.json
Lectura de tensión
readVoltAgent()
-Delay
-Parsear perfil
-Representar curvas calibración
-Error de comunicación
-Fin de simulación
config.json
config.json
Callback de datos obtenidos
main()
-Almacenamiento de datos obtenidos
-Warnings PV
-SOC PV
functions.py
Entrada de datos
inputData.py
Modifica
config.json
-Representar curvas
-Cambio de unidades de tiempo
-Error de comunicación
-Fin de simulación
-Warnings
-Enviar y recibir tiempos entre máquinas (PV)
-Parsear perfiles
-Delay
-Error de comunicación
-Fin de simulación
-Warnings
-Enviar y recibir tiempos entre máquinas (PV)
-Parsear perfiles
-Delay
Callback de datos obtenidos
Datos obtenidos
curvesCD.py
curvesPV.py
powerSupplyPV()
powerSupplyProfile()
loadPV()
loadProfile()
config.json
config.json
Multithreading (3 hilos)


UNIVERSITAT
JAUME I