Universidad de la República
Facultad de Ingeniería

# AutoBMS: Sistema de Gestión de Baterías para Vehículos Eléctricos

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Nahuel Batista, Alejo Bravo, Francisco Tutzo

en cumplimiento parcial de los requerimientos para la finalización de la carrera de Ingeniería Eléctrica.

Tutor

Juan Pedro Carriquiry ..... Universidad de la República
Mariana Siniscalchi ..... Universidad de la República

Tribunal

Virginia Echinope ..... Universidad de la República
Juan Pablo Oliver ..... Universidad de la República
Andrés Seré ..... Universidad de la República

Montevideo
viernes 19 diciembre, 2025

*AutoBMS: Sistema de Gestión de Baterías para Vehículos Eléctricos,*
Nahuel Batista, Alejo Bravo, Francisco Tutzo.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.3).
Contiene un total de 121 páginas.
Compilada el viernes 19 diciembre, 2025.
http://iie.fing.edu.uy/

La mejor forma de predecir el futuro es inventarlo.

Alan Kay

Esta página ha sido intencionalmente dejada en blanco.

# Agradecimientos

AutoBMS se desarrolló gracias al apoyo constante de numerosas personas e instituciones, cuyo acompañamiento fue fundamental. Agradecemos a nuestras familias y amigos, por su paciencia durante los meses de trabajo, por comprender las horas dedicadas al proyecto; a nuestros tutores del proyecto la Dra. Mariana Siniscalchi y el Ing. Juan Pedro Carriquiry, por su orientación técnica, su dedicación y su compromiso académico.

A la Dra. Erika Teliz, por su colaboración durante las etapas de ensayo y su valioso aporte para establecer contacto con el Dr. Unai Iraola, cuya asesoría técnica fue determinante para el desarrollo del algoritmo de estimación del SOH. Agradecemos especialmente al Dr. Iraola por su generosidad al compartir su conocimiento y experiencia, que permitió transformar ideas iniciales en criterios sólidos de implementación.

Al Instituto de Ingeniería Eléctrica (IIE) de la Facultad de Ingeniería de la Universidad de la República, por brindar el espacio, los recursos y el entorno de trabajo que hicieron posible este desarrollo. El ambiente colaborativo del instituto, así como la disponibilidad de sus laboratorios y equipamiento, resultaron esenciales para el avance del proyecto.

A todas las personas que, directa o indirectamente, contribuyeron con su tiempo, conocimiento o confianza, expresamos nuestro más sincero agradecimiento.

Esta página ha sido intencionalmente dejada en blanco.

*A nuestras familias.*

Esta página ha sido intencionalmente dejada en blanco.

# Resumen

El crecimiento sostenido de la movilidad eléctrica y del almacenamiento de energía a nivel mundial ha incrementado la necesidad de optimizar la seguridad y la vida útil de las baterías recargables. En particular, las baterías de litio presentan una alta densidad energética, pero también riesgos asociados a sobrecargas, sobredescargas o condiciones térmicas extremas, lo que exige sistemas de control confiables.

Los Sistemas de Gestión de Baterías constituyen la herramienta fundamental para enfrentar estos desafíos. Son responsables de supervisar las variables críticas, proteger ante fallas y preservar la longevidad de prácticamente todas las baterías que hoy impulsan vehículos eléctricos o alimentan aplicaciones estacionarias.

En este marco surge AutoBMS, un BMS abierto, documentado y configurable, concebido como recurso académico para la Facultad de Ingeniería. Se diseñaron el hardware y el firmware del sistema y se fabricó un prototipo funcional en un único circuito impreso. Además, se desarrolló una interfaz gráfica que permite monitorear el estado de la batería y ajustar parámetros de manera sencilla, facilitando su uso en ensayos y prácticas.

El sistema desarrollado permite medir los parámetros eléctricos y térmicos de cada celda de la batería y, a partir de dicha información, implementar mecanismos de protección, generar avisos ante condiciones de riesgo, equilibrar el estado de carga de las celdas y comunicar los datos relevantes a dispositivos externos, como la unidad de control del vehículo o un PC.

En síntesis, AutoBMS constituye una solución eficiente y autónoma, con potencial de adaptación a diferentes aplicaciones y con margen para expandir sus funcionalidades futuras. Todos los archivos de diseño se encuentran disponibles públicamente en el repositorio GitLab del proyecto.[1]

[1] `gitlab.fing.edu.uy/autobms`

# Tabla de contenidos

Esta página ha sido intencionalmente dejada en blanco.

# Capítulo 1

# Introducción

## 1.1. Motivación

El crecimiento sostenido de la movilidad eléctrica y el almacenamiento estacionario ha incrementado la necesidad de gestionar baterías recargables con altos estándares de seguridad, eficiencia y vida útil. En este escenario, los Sistemas de Gestión de Baterías o BMS (del inglés *Battery Management System*) son un componente crítico que se encarga de supervisar variables eléctricas y térmicas, proteger ante condiciones de falla y optimizar la operación del conjunto de celdas.

En el ámbito académico, particularmente en la FING–UDELAR, se ha identificado la necesidad de contar con un sistema abierto que permita estudiar y experimentar con distintos tipos de baterías de iones de litio. Las soluciones comerciales disponibles suelen ser propietarias y su funcionamiento interno no puede ser modificado.

Si bien todos los BMS comparten funciones universales como la medición de parámetros, la activación de protecciones, balanceo de celdas, no existe una arquitectura estándar ni una forma universal de configurar cómo y cuándo se ejecutan dichas tareas. Asimismo, los algoritmos de estimación del estado de carga o SOC (del inglés *State Of Charge*) y del estado de salud o SOH (del inglés *State Of Health*) difieren ampliamente entre fabricantes, cada empresa emplea métodos y modelos propios, que en general no son públicos. Esto dificulta su estudio, la comparación entre técnicas y la validación experimental en entornos académicos.

Por estas razones, el desarrollo de un BMS de diseño abierto resulta de especial interés, ya que permite comprender, modificar y optimizar cada función de éste facilitando la investigación y la enseñanza en el área de almacenamiento energético.

En respuesta a esto, el proyecto AutoBMS se desarrolla como un BMS orientado a la formación profesional e investigación.

De esa forma, se dispondrá de un sistema para:

- **Enseñanza:** brindar un recurso estable, mantenible y escalable en el tiempo que permita realizar prácticas de laboratorio y promueva el aprendizaje.
- **Investigación:** permite realizar ensayos reproducibles con registro de datos y configuraciones adaptables, facilitando la comparación sistemática de

resultados.

- **Transparencia tecnológica:** reducir la dependencia de BMS propietarios y fomentar la comprensión de los mecanismos internos de un BMS.

## 1.2. Antecedentes

En los laboratorios de FING–UDELAR se dispone de BMS comerciales orientados a aplicaciones industriales o vehiculares. Si bien estos dispositivos cumplen adecuadamente su función en escenarios de producción, presentan limitaciones para el uso académico ya que suelen estar preconfigurados para ciertas químicas de baterías y operan con *firmware* cerrado. Esto dificulta la adaptación a distintos ensayos y la comprensión profunda de sus algoritmos internos.

Si bien existen proyectos de acceso libre de BMSs, estos suelen centrarse exclusivamente en el rendimiento, por lo que emplean ASICs (del inglés *Application Specific Integrated Circuit*) en su diseño, lo que impide el estudio detallado de su funcionamiento en un entorno educativo.

El proyecto AutoBMS surge para proporcionar un BMS orientado al uso académico, con un diseño modular para facilitar su entendimiento, con flexibilidad amplia en su configuración y adaptabilidad para ensayos comparativos entre algoritmos y configuraciones de baterías.

## 1.3. Objetivos y alcance

### 1.3.1. Objetivo General

El objetivo principal de este proyecto es diseñar e implementar un BMS abierto, documentado y modificable que sirva como plataforma académica para la enseñanza e investigación en FING–UDELAR, apto para estudiar baterías.

### 1.3.2. Objetivos específicos

- **Diseño y desarrollo del Sistema Embebido:** desarrollar una placa con sensado de tensiones por celda, corriente y temperatura de la batería, llaves de potencia con configuración adecuada para cerrar/abrir la carga/descarga de la batería y con mecanismo de balanceo para asegurar un equilibrio adecuado entre las celdas de la batería.
- **Desarrollo de firmware:** desarrollo de *firmware* que contemple los componentes físicos elegidos.
- **Estimación de estados:** incorporar métodos de estimación de SOC y de SOH.
- **Realización de pruebas y validación del BMS:** el algoritmo de estimación para el SOH deberá ser ensayado y validado para asegurar que

proporciona resultados consistentes para la gestión de la batería. Se debe verificar que el BMS cumple con los requisitos de funcionalidad, precisión y seguridad establecidos.

- **Comunicación:** implementar un medio de comunicación para configuración y monitoreo desde un dispositivo externo.

### 1.3.3. Alcance

El presente trabajo comprende el diseño y la documentación de un BMS capaz de cumplir con los objetivos planteados previamente (Subsección 1.3.2). Dada la diversidad de químicas existentes en las baterías de iones de litio y sus diferencias de comportamiento electroquímico, se implementa un algoritmo de estimación de SOC y SOH para celdas de tipo NMC (níquel, manganeso y cobalto), correspondientes a las disponibles durante el desarrollo del proyecto. No obstante, el *hardware* del dispositivo cuenta con la flexibilidad necesaria para adaptarse a otras químicas, requiriendo únicamente la adecuación del *firmware* correspondiente en cada caso.

Asimismo, se incluye la implementación de una interfaz gráfica destinada al monitoreo y configuración del BMS. Si bien esta funcionalidad no formaba parte del plan inicial del proyecto, su incorporación se consideró una mejora significativa, al aportar un valor añadido al sistema y facilitar la interacción del usuario con el dispositivo.

## 1.4. Criterios de éxito

Contemplando el cumplimiento de los objetivos especificados en la Sección 1.3, a continuación se establecen los criterios de éxito y su verificación:

- **Integración del sistema embebido:** el BMS debe integrar correctamente todos los componentes de *hardware* y *firmware* (sensores, microcontrolador, algoritmos), garantizando su funcionamiento conjunto sin conflictos.
- **Cumplimiento funcional:** el sistema debe ejecutar de forma efectiva las funciones de protección, medición, balanceo y comunicación, verificadas mediante ensayos en entorno controlado.
- **Implementación del algoritmo de SOH:** se debe desarrollar un algoritmo capaz de evaluar la condición de la batería de forma consistente y estable.
- **Optimización de recursos:** el proyecto debe completarse dentro de los plazos establecidos.
- **Documentación técnica completa:** toda la documentación asociada al diseño, implementación y pruebas del BMS debe estar completa, clara y estructurada, facilitando futuras mejoras, mantenimiento o replicación del sistema.

Esta página ha sido intencionalmente dejada en blanco.

# Capítulo 2

# Solución Propuesta

El objetivo de este capítulo es ofrecer una visión general de la solución propuesta. Se busca proporcionar una primera aproximación a las estrategias adoptadas para cumplir los distintos objetivos del proyecto y a las principales decisiones de diseño que permitieron su implementación, de modo que los capítulos dedicados al desarrollo de *hardware* y, posteriormente, de *firmware*, puedan comprenderse con mayor claridad.

Además, este capítulo presenta los requerimientos detallados para el sistema, los cuales se utilizarán para medir el cumplimiento de los criterios de éxito.

## 2.1. Descripción del Sistema

Se propone el diseño de un sistema embebido que integre un microcontrolador (MCU) y el *hardware* necesario para cumplir con las funcionalidades establecidas, así como la fabricación de un prototipo funcional. El diseño se orienta a permitir la experimentación e investigación sobre el comportamiento de sistemas de gestión de baterías, por lo que se definieron corrientes nominales acordes a las de un vehículo eléctrico ligero (bicicleta o monopatín eléctrico), en lugar de las utilizadas en automóviles eléctricos. De este modo, se simplifica el diseño y la fabricación del sistema sin afectar su valor didáctico ni el alcance de los objetivos planteados.

Para poder concentrar la totalidad del sistema en una única placa de circuito impreso (PCB), se optó por una arquitectura de BMS centralizado. Esta consiste en un dispositivo central que se conecta directamente a la batería y actúa como intermediario entre esta y la carga eléctrica, como se ilustra en el diagrama de la Figura 2.1.

El sistema debe, en primer lugar, adquirir el estado de las celdas que conforman la batería, midiendo las magnitudes eléctricas y térmicas necesarias. A partir de esta información, el *firmware* estima parámetros relevantes (como el SOC y el SOH) y toma decisiones de protección en caso de condiciones anómalas. Ante una situación de riesgo, el BMS debe ser capaz de interrumpir el flujo de corriente para preservar la integridad del sistema y la seguridad del usuario.

Además, el sistema debe permitir la comunicación con una computadora ex-

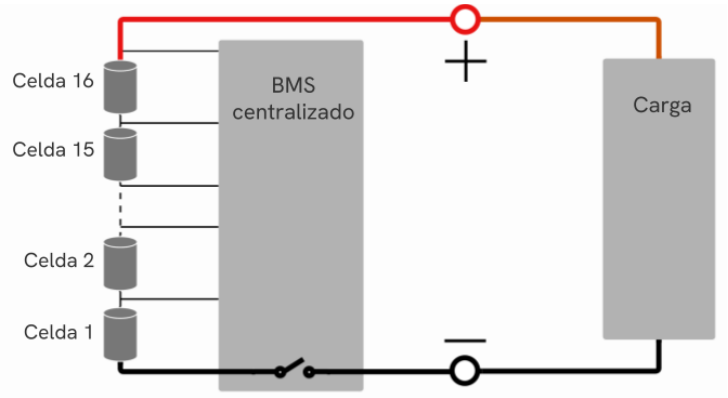


Figura 2.1: Diagrama general del sistema propuesto.

terna para el monitoreo y configuración de los parámetros de funcionamiento. Los umbrales de protección deben ser ajustables por el usuario, y el dispositivo debe operar alimentado por la misma batería que supervisa, sin generar un consumo significativo que afecte de forma perceptible su autonomía.

Durante la carga, el BMS debe ejecutar el balanceo de celdas para igualar sus estados de carga y optimizar la capacidad total del conjunto.

### 2.1.1. Hardware

El *hardware* del sistema está diseñado para cumplir con las funciones de medición, protección, balanceo y comunicación requeridas. Para organizar el diseño y facilitar la integración se adoptó una estructura modular, que agrupa los distintos componentes en bloques específicos. La conexión entre estos módulos se ilustra en el diagrama de la Figura 2.2.

El núcleo del diseño lo constituye el MCU `ESP32-S3`, encargado de coordinar el funcionamiento integral del sistema y garantizar el cumplimiento de los requerimientos funcionales y temporales establecidos. En función de las tareas que debe realizar el BMS, los periféricos asociados se organizan en cinco módulos principales, cuya arquitectura y funcionamiento se describen a continuación. En el Capítulo 4 se detalla el diseño del BMS.

#### Alimentación

El sistema debe ser alimentado directamente por la propia batería, procurando un consumo mínimo que no afecte de forma apreciable su autonomía. No todos los componentes funcionan a la misma tensión, la gran mayoría se alimenta a 3,3 V. Se utiliza un convertidor *buck* DC/DC para generar dicha tensión a partir de la batería de forma eficiente.

El convertidor se alimenta desde la tensión total del conjunto de celdas, correspondiente al punto de mayor potencial del sistema. De este modo, se garantiza que el BMS disponga siempre del voltaje necesario para su funcionamiento, incluso en situaciones donde algunas celdas se encuentren profundamente descargadas o fuera

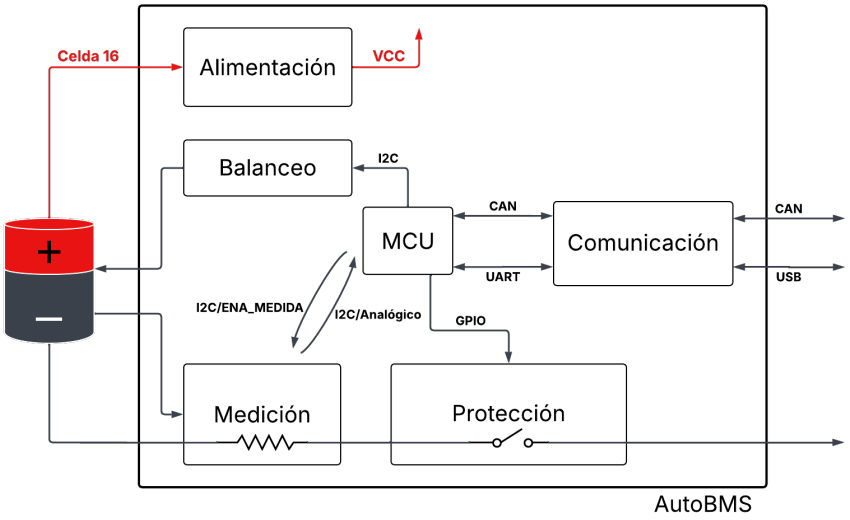


Figura 2.2: Diagrama de bloques del sistema BMS.

de servicio, evitando así la desactivación del BMS. Además, esta medida evita la descarga desigual de las celdas, consumiendo corriente de todas por igual.

### Medición

Este módulo se encarga de la adquisición de las tres magnitudes fundamentales que describen el estado de una batería: tensión, corriente y temperatura.

Para la medición de tensiones se emplea el conversor analógico-digital (ADC) interno del MCU. Se implementa una arquitectura basada en multiplexores analógicos que permite acceder a todas las celdas utilizando una cantidad reducida de pines del MCU. En el prototipo desarrollado se miden dos celdas simultáneamente, distribuidas en dos canales independientes del ADC.

La medición de corriente se realiza mediante un sensor diseñado específicamente para esta tarea, que utiliza una resistencia *shunt* como elemento de sensado. Esta configuración ofrece una elevada precisión y pérdidas mínimas de potencia. Además, el sensor cuenta con un registro interno que permite calcular la carga transferida, parámetro de gran utilidad para la estimación del SOC y el SOH.

La temperatura se mide en dos puntos internos de la batería, en contacto directo con las celdas. Se utilizan sensores de bajo consumo que no requieren calibración, lo que simplifica la implementación y asegura una lectura confiable durante todo el ciclo de operación.

La información adquirida por este módulo es procesada por el *firmware* del BMS, que se encarga de aplicar los filtros correspondientes y de evaluar las condiciones de seguridad antes de comunicar los resultados al resto del sistema.

## Protección

El módulo de protección tiene como objetivo asegurar la integridad eléctrica de la batería y de las cargas conectadas, permitiendo interrumpir el flujo de corriente en ambos sentidos de manera independiente durante la descarga (corriente saliente) y durante la carga (corriente entrante). Para ello se emplean *MOSFET*s de potencia dimensionados para soportar los niveles de tensión y corriente característicos del sistema, configurados en disposición *back-to-back* (*source* común), lo que permite bloquear la conducción a través de los diodos intrínsecos antiparalelos y asegurar un control simétrico del paso de corriente.

Adicionalmente, se implementa un circuito de predescarga que limita la corriente transitoria al conectar la batería a una carga de tipo capacitiva, como ocurre típicamente en vehículos eléctricos, con el *DC-link*. Este circuito, conformado por una resistencia en serie controlada por un transistor auxiliar, permite cargar gradualmente los condensadores del bus de corriente continua, evitando picos de corriente que podrían dañar los semiconductores o degradar las pistas de potencia [35].

Una vez completado el proceso de predescarga, el sistema activa las llaves principales, habilitando el paso normal de corriente y asegurando una transición suave hacia el régimen operativo.

## Balanceo

El balanceo de celdas es esencial para garantizar que todas las celdas de la batería mantengan un nivel de carga uniforme, condición necesaria para maximizar la capacidad utilizable y prolongar su vida útil. Debido a pequeñas diferencias en la autodescarga, resistencia interna o temperatura, algunas celdas tienden a cargarse o descargarse más rápido que otras, generando desbalances que pueden derivar en sobrecargas o sobredescargas individuales.

Existen dos estrategias principales de balanceo: la activa, que transfiere energía entre celdas mediante convertidores dedicados, y la pasiva, que disipa el exceso de energía de las celdas más cargadas a través de resistencias conectadas en paralelo. Aunque el balanceo activo ofrece mayor eficiencia energética, su complejidad y costo son significativamente mayores. Por ello, en aplicaciones donde la simplicidad y el bajo costo son prioritarios, como en este proyecto, se opta por un esquema de balanceo pasivo.

El sistema implementado utiliza una arquitectura de *resistencias conmutadas en paralelo*, activadas únicamente durante la etapa de carga, práctica común en vehículos eléctricos. El algoritmo de control se basa exclusivamente en las mediciones de tensión de cada celda, habilitando la disipación de energía en aquellas que alcancen primero el umbral de activación definido, hasta igualar su nivel con el resto del conjunto.

El control del balanceo se gestiona desde el MCU, a través de *bloques de switches* conectados a las resistencias correspondientes a cada celda.

#### Comunicación

El módulo de comunicación permite el intercambio de información entre el BMS y dispositivos externos al sistema. En aplicaciones automotrices, el protocolo CAN es el estándar predominante para la comunicación entre unidades electrónicas. Se trata de un bus diferencial robusto y tolerante a interferencias, diseñado para operar en entornos con alto nivel de ruido eléctrico y garantizar la integridad de los datos. Por su popularidad y características, su inclusión en un BMS automotriz resulta fundamental para permitir la integración con otros controladores del vehículo, como el inversor, el cargador o la unidad de control principal.

Para implementar esta interfaz, se utiliza un transceptor específico que adapta las señales diferenciales del bus CAN a niveles lógicos compatibles con el MCU, asegurando una comunicación confiable y conforme a las normas del protocolo.

Además, se incorpora una interfaz de comunicación mediante un convertidor USB–UART, que permite conectar el sistema directamente a una PC. Esta conexión se emplea para la configuración y monitoreo del BMS a través de la interfaz gráfica desarrollada.

### 2.1.2. Firmware

El *firmware* constituye la capa lógica que gobierna el funcionamiento del BMS, coordinando la interacción entre los distintos módulos de *hardware* y garantizando el cumplimiento de sus tareas. Su diseño se basa en una arquitectura modular que permite aislar las diferentes funcionalidades, facilitando así el desarrollo, el mantenimiento y la futura ampliación del sistema.

El `ESP32-S3` ejecuta las tareas de adquisición, procesamiento y control en tiempo real mediante un sistema operativo de tiempo real (RTOS), que organiza la operación en múltiples tareas concurrentes.

Cada módulo físico cuenta con su correspondiente módulo lógico en el *firmware*, encargado de gestionar las señales, aplicar los algoritmos de control y comunicar los resultados a las demás partes del sistema. En el Capítulo 5 se detalla la estructura, funcionamiento y lógica interna de este.

## 2.2. Requerimientos detallados

La Tabla 2.1 resume los requerimientos definidos para el sistema. Estos constituyen la base de referencia para la evaluación del cumplimiento de los criterios de éxito del proyecto.

| Categoría | Requerimientos Detallados |
|---|---|
| **Medición de parámetros** | Medir la tensión de cada celda con una precisión mínima de $\pm 0{,}01\,\mathrm{V}$. |
| | Medir la corriente de carga y descarga en un rango de al menos $20\,\mathrm{A}$, con precisión mínima de $\pm 0{,}2\,\mathrm{A}$. |
| | Monitorear la temperatura de la batería en al menos dos puntos, con un rango de $-20\,^{\circ}\mathrm{C}$ a $85\,^{\circ}\mathrm{C}$ y precisión de $\pm 2\,^{\circ}\mathrm{C}$. |
| | Relevar las medidas de tensión, corriente y temperatura con una frecuencia mínima de $3\,\mathrm{Hz}$. |
| **Protección de la batería** | Implementar protecciones contra sobrecarga, sobredescarga, sobrecorriente, sobrecalentamiento, cortocircuito y polaridad inversa. |
| | Desconectar la batería de fuentes o cargas externas si se detecta alguna de las condiciones anteriores, tras un tiempo configurable por el usuario. |
| | Permitir ajuste de umbrales de protección contra sobrecarga y sobredescarga entre $1{,}0\,\mathrm{V}$ y $4{,}4\,\mathrm{V}$. |
| | Permitir ajuste del umbral de sobrecorriente entre $0{,}5\,\mathrm{A}$ y $20\,\mathrm{A}$. |
| **Balanceo de celdas** | Implementar un mecanismo de balanceo pasivo de celdas. |
| **Algoritmo de SOH** | Implementar un algoritmo para el cálculo del SOC con una precisión de $\pm 5\,\%$ y del SOH. |
| **Comunicación** | Incluir interfaces UART y CAN para transmisión y recepción de datos con dispositivos externos. |
| | Permitir la configuración de todos los parámetros del sistema mediante la conexión UART. |

Tabla 2.1: Requerimientos detallados del sistema BMS.

# Capítulo 3

# Estado de una Batería

El estado de una batería describe su condición interna y su capacidad para almacenar y entregar energía de forma segura y eficiente. En el contexto de un BMS automotriz, los parámetros más relevantes para caracterizar dicho estado son el estado de carga (SOC) y el estado de salud (SOH), ya que permiten estimar la energía disponible y el grado de degradación del sistema electroquímico, respectivamente. En este capítulo se definen estos dos parámetros, se exponen los principales métodos para su estimación y se desarrolla el algoritmo implementado en el presente proyecto.

## 3.1. Conceptos Generales

El estado de carga (SOC) expresa la fracción de energía utilizable que conserva una celda respecto a su capacidad real. Una celda se considera completamente cargada cuando su voltaje de circuito abierto (OCV) alcanza el límite superior $v_h(T)$, definido por el fabricante. De manera análoga, la celda se considera completamente descargada cuando su voltaje alcanza el límite inferior $v_l(T)$. Estos estados corresponden a 100 % y 0 % de SOC, respectivamente [26].

Un método común para cargar completamente una celda consiste en ejecutar un perfil de carga de corriente constante hasta que el voltaje terminal sea igual a $v_h(T)$, seguido de un perfil de voltaje constante hasta que la corriente de carga se vuelva infinitesimal, típicamente del orden del 3 % de la corriente nominal de carga.

La capacidad real $Q_{\text{act}}$ representa la carga total que la celda puede entregar entre dichos límites, mientras que la capacidad restante $Q_{\text{res}}$ es la carga aún disponible desde el estado actual hasta la descarga completa. El SOC se define entonces como:

$$SOC = \frac{Q_{\text{res}}}{Q_{\text{act}}} \times 100. \tag{3.1}$$

Por otra parte, el estado de salud (SOH) cuantifica la pérdida de capacidad de la batería respecto a su condición nominal. Se define como la razón entre la capa-

cidad máxima que entrega la batería en condiciones nominales, en el estado actual de envejecimiento, $Q_{\max}$, y la capacidad nominal especificada por el fabricante, $Q_{\text{nom}}$.

$$SOH = \frac{Q_{\max}}{Q_{\text{nom}}} \times 100. \tag{3.2}$$

Esta métrica refleja el grado de degradación de la batería debido al envejecimiento, influenciado por factores como los ciclos de carga y descarga, la temperatura y las condiciones de operación. En aplicaciones automotrices, se considera que una batería alcanza el fin de su vida útil cuando su SOH alcanza un valor entre 80 % y 60 %, dependiendo del fabricante. Dado que la capacidad no puede medirse directamente, el BMS debe estimarla a partir de magnitudes medibles como corriente, voltaje y temperatura utilizando distintos modelos y algoritmos de estimación.

## 3.2. Métodos de Estimación

Una vez definidos los conceptos de SOC y SOH, es necesario establecer cómo pueden estimarse a partir de magnitudes eléctricas medibles. En una batería operando dentro de un sistema, dichas variables no pueden medirse directamente, por lo que deben inferirse mediante modelos o algoritmos que relacionen la corriente, el voltaje y la temperatura con el estado interno de la celda.

Existen diversos métodos para la estimación de estos parámetros, que difieren en su complejidad, precisión y costo computacional. En las siguientes secciones se presentan los principales enfoques para la estimación del SOC y del SOH, y luego se presenta el método empleado en este proyecto.

### 3.2.1. Métodos de estimación del SOC

Los enfoques para estimar el SOC de una batería pueden ser *directos* o *indirectos* [8].

#### Métodos directos

Se basan en magnitudes eléctricas medibles, sin requerir un modelo complejo de la celda. Entre ellos destacan:

- **Conteo de Coulombs (CC):** estima el SOC integrando la corriente de carga o descarga en el tiempo. Es simple y de bajo costo computacional, aunque acumula error con el tiempo y depende del conocimiento preciso del SOC inicial.
- **Voltaje en circuito abierto (OCV):** determina el SOC a partir de la relación empírica entre la tensión en reposo y el nivel de carga. Presenta buena precisión, pero requiere períodos de descanso prolongados y sufre variaciones con la temperatura y el envejecimiento.

- **Espectroscopía de impedancia electroquímica (EIS):** mide la respuesta de la celda a una excitación alterna para inferir su SOC. Brinda información detallada sobre los procesos internos, aunque su aplicación práctica es limitada a entornos de laboratorio por su complejidad y necesidad de calibración.

#### Métodos indirectos

Emplean modelos matemáticos o algoritmos de estimación para inferir el SOC a partir del comportamiento dinámico de la batería. Los principales grupos son:

- **Basados en modelos eléctricos o electroquímicos (ECM, EChM**: representan la batería mediante circuitos equivalentes o modelos fisicoquímicos que relacionan corriente, tensión y SOC. Ofrecen buena exactitud pero requieren identificación de parámetros y mayor capacidad de cómputo.

- **Basados en filtros adaptativos (Kalman, RLS)**: aplican técnicas de filtrado para actualizar el SOC en tiempo real considerando el ruido de medición y las incertidumbres del modelo.

- **Basados en inteligencia artificial y aprendizaje automático**: emplean redes neuronales, lógica difusa o aprendizaje profundo para aprender relaciones no lineales entre las variables medibles y el SOC. Alcanzan alta precisión pero demandan grandes conjuntos de datos y recursos de procesamiento.

- **Métodos híbridos y de transferencia de aprendizaje**: combinan enfoques de predicción anteriores (por ejemplo, redes neuronales con filtros de Kalman) o utilizan modelos pre-entrenados, buscando mejorar la generalización.

Aunque los métodos avanzados basados en modelos y en inteligencia artificial prometen un mejor rendimiento en condiciones dinámicas, a costa de una mayor complejidad y exigencias computacionales, el CC y el OCV siguen siendo los métodos más utilizados en aplicaciones automotrices debido a su simplicidad, bajo costo y fácil integración en sistemas embebidos [8].

En este proyecto se adopta un método directo basado en CC, pero combinándolo con correcciones periódicas por OCV. Este es descrito en la Subsección 3.3.1.

### 3.2.2. Métodos de estimación del SOH

La estimación del SOH de una batería puede resultar compleja en entornos reales, donde influyen factores como la temperatura, el número de ciclos de carga y descarga, y la antigüedad de la celda.

En general, los métodos de estimación del SOH se agrupan en dos categorías principales: *métodos basados en modelos* y *métodos experimentales* [24].

### Métodos basados en modelos

Estos métodos estiman parámetros representativos del estado interno de la batería, como la capacidad o la resistencia interna, utilizando modelos matemáticos o eléctricos. Se dividen en dos subgrupos principales:

- **Algoritmos adaptativos:** Emplean modelos que se ajustan en tiempo real según las mediciones de corriente, tensión y temperatura. Mediante técnicas como la estimación recursiva, los filtros de Kalman o los modelos electroquímicos simplificados, estos algoritmos actualizan los parámetros del modelo para reflejar el envejecimiento y las variaciones de operación. Su principal ventaja es la capacidad de seguimiento dinámico del SOH, aunque a costa de una mayor complejidad computacional y dependencia de la calibración inicial.
- **Métodos basados en datos:** Estos enfoques prescinden de un modelo físico de la batería y se apoyan únicamente en datos históricos de envejecimiento. A través de técnicas de aprendizaje automático, regresión o análisis estadístico, establecen relaciones entre las variables medibles y la pérdida de capacidad o aumento de resistencia. Su precisión puede ser elevada, pero requieren grandes conjuntos de datos experimentales y no siempre generalizan bien a condiciones no vistas durante el entrenamiento.

### Métodos experimentales

Se basan en la observación directa del comportamiento de la batería mediante ensayos controlados. Aunque su aplicación en vehículos es limitada por la necesidad de instrumentación especializada y condiciones reproducibles, constituyen la base para validar y parametrizar los métodos basados en modelos. Pueden clasificarse en dos grupos:

- **Métodos de medición directa:** Estos métodos implican la evaluación directa de los atributos físicos o químicos del sistema para determinar su salud. Incluyen ensayos de capacidad, mediciones de impedancia y el conteo de Coulomb, este último utilizado en el presente proyecto. Al requerir mediciones accesibles, son los más aplicables en sistemas embebidos, aunque menos precisos para estimaciones absolutas de degradación.
- **Métodos de análisis indirecto:** Infieren el SOH a partir de parámetros derivados de las curvas de tensión y capacidad, como el análisis de capacidad incremental (ICA), el análisis de voltaje diferencial (DVA) o técnicas ultrasónicas. Correlacionan características específicas de las curvas de carga o descarga con la pérdida de capacidad o el aumento de resistencia, permitiendo detectar tendencias de degradación sin necesidad de mediciones destructivas.

Una vez vistos los métodos más relevantes para ambos parámetros, se procede a detallar el procedimiento adoptado en este proyecto.

## 3.3. Algoritmo Implementado

El algoritmo implementado en este proyecto combina el CC con la medición del OCV para estimar el SOC y el SOH de las baterías. Se basa en el trabajo presentado en [3], adaptado para su implementación en el sistema embebido desarrollado.

El método de CC se destaca por su simplicidad, sin embargo, su precisión se ve comprometida por la acumulación de errores durante el cálculo incremental de la carga transferida en operaciones prolongadas. Además, su resultado depende del valor inicial del SOC y de la capacidad real de la batería.

El algoritmo implementado contrarresta estos efectos realizando anclajes[1] por OCV entre periodos de reposo de 25 minutos para restablecer las estimaciones del SOC, minimizar el error y proporcionar un punto de partida confiable para la estimación. La condición de reposo es necesaria para que la celda se estabilice y la lectura de dicho voltaje refleje el valor real.

Luego de estimar el SOC se estima el SOH, a partir de la carga acumulada y la diferencia de SOC registrada entre anclajes de OCV. Para ello, emplea el algoritmo de mínimos cuadrados recursivos (RLS).

En la Subsección 3.3.1 se detallan los fundamentos que respaldan el algoritmo implementado.

### 3.3.1. Estimación del SOC

La variación de carga entre dos puntos en el tiempo puede calcularse integrando la corriente de la batería, $I_b$, durante el intervalo considerado (CC). Esto es:

$$q = \int_{t_0}^{t} I_b \, dt. \tag{3.3}$$

Partiendo de la definición del SOC en la Ecuación 3.1, se puede definir el SOC en un tiempo determinado en función del SOC en un tiempo anterior y la variación de carga entre ambos instantes:

$$SOC_t = SOC_{t_0} + \frac{q}{Q_{\text{act}}}. \tag{3.4}$$

El comportamiento de la capacidad de carga y descarga varía dependiendo de la corriente y de la temperatura de la batería durante la operación. Las temperaturas bajas y las altas tasas de corriente, principalmente a bajas temperaturas, conducen a una disminución de la capacidad máxima de la batería en comparación con las condiciones nominales.

Para la compensación por OCV, se utiliza una tabla de búsqueda que relaciona el OCV y la temperatura con el SOC. Esta tabla es específica para las celdas elegidas, y se obtiene mediante pruebas experimentales de carga y descarga a bajas

[1] El término *anclaje por OCV* hace referencia a los puntos en los que la batería permanece en reposo el tiempo suficiente para medir su OCV. Estos puntos se utilizan para recalibrar la estimación del SOC, corrigiendo la deriva acumulada del conteo de carga y estableciendo un nuevo valor de referencia o "anclaje" en la curva OCV–SOC.

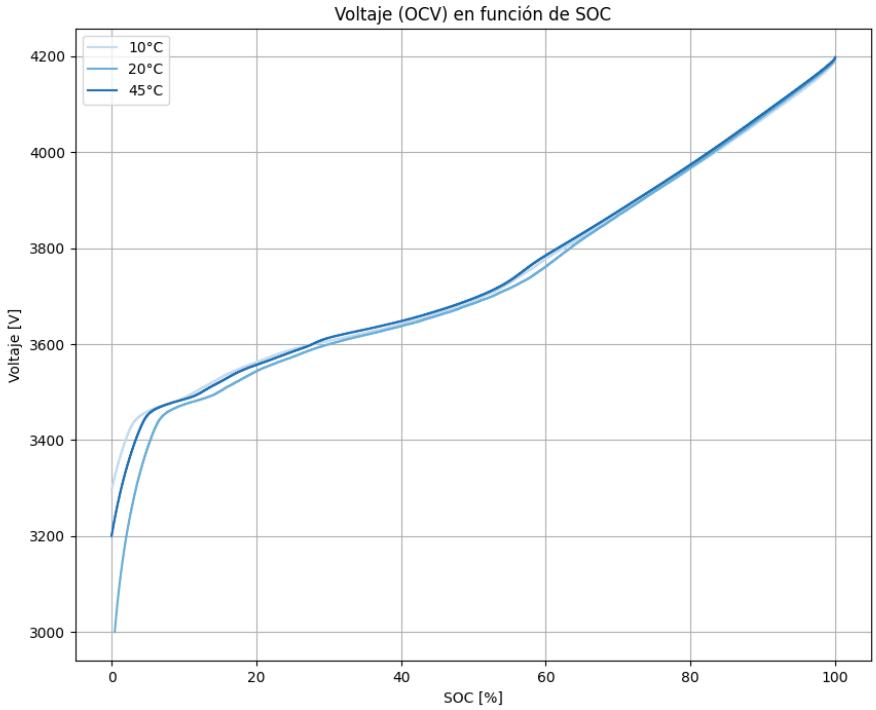


Figura 3.1: OCV en función del SOC, para tres temperaturas diferentes.

corrientes (C/50 en los ensayos realizados [2] ), registrando el OCV a diferentes niveles de carga y temperatura. El código implementado realiza una interpolación lineal entre los puntos de la tabla para estimar el SOC correspondiente a una medición de OCV y temperatura dadas. En la Figura 3.1 se pueden encontrar los datos obtenidos en los ensayos.

Para que la batería se considere en estado de reposo y se pueda realizar la medición de OCV, la corriente debe ser nula durante un tiempo mínimo de 25 minutos.

### 3.3.2. Estimación del SOH

Para la estimación del SOH se utiliza la carga acumulada $q$ entre dos anclajes de OCV y la variación del SOC estimada en dichos puntos. A partir de la definición de la Ecuación 3.4, se puede despejar la capacidad real $Q_{\text{act}}$ como:

$$Q_{\text{act}} = \frac{q}{\Delta SOC}, \tag{3.5}$$

donde,

[2]La corriente suele expresarse mediante la tasa de descarga o *C-rate*, donde 1C equivale a una descarga completa en una hora.

$$\Delta SOC = SOC_t^{\text{OCVupdate}} - SOC_{t_0}^{\text{OCVupdate}}, \tag{3.6}$$

es decir, la variación de SOC entre dos actualizaciones (*updates*) por OCV.

Cabe destacar que la precisión del método dependerá fuertemente de la magnitud de $\Delta SOC$, así como del error en la medición de $q$, que está influenciado por las condiciones de operación (corriente y temperatura) y el efecto de la autodescarga. Por este motivo, se toman la siguientes precauciones:

- Se establece un umbral mínimo de $\Delta SOC$ para realizar una estimación de capacidad; en este proyecto, se fija en 5 %. Esto quiere decir que si $\Delta SOC \leq \Delta SOC_{min}$, no se actualiza la capacidad, y se sigue acumulando carga $q$ hasta el siguiente reposo, donde se vuelve a verificar esta condición.
- Se utiliza un filtro de mínimos cuadrados recursivos (RLS) para actualizar la estimación de la capacidad real a partir de las sucesivas mediciones.

El método RLS permite actualizar una estimación en tiempo real sin necesidad de recalcularla desde cero ante cada nueva muestra. A diferencia de los métodos de promediado simple, el RLS pondera las observaciones recientes con mayor peso que las antiguas mediante un *factor de olvido* $\lambda$, comprendido entre 0 y 1. De esta forma, la estimación se adapta progresivamente a los cambios reales en la capacidad de la celda, filtrando el ruido de medición y reduciendo la influencia de eventos pasados.

En el contexto de la estimación del SOH, esta técnica resulta especialmente adecuada, ya que la capacidad de la batería evoluciona lentamente en el tiempo mientras que las mediciones de carga y tensión pueden presentar ruido considerable. El uso del RLS permite, por tanto, obtener una estimación suave y estable de $Q_{\text{act}}$, sensible a la degradación acumulada pero robusta frente a fluctuaciones instantáneas en las mediciones.

La Figura 3.2 describe el proceso general de estimación de ambos parámetros. En primer lugar, se recopilan los valores actuales de corriente, voltaje y temperatura. Si la corriente es diferente de cero, el algoritmo mide la carga acumulada, y estima el SOC a partir de ello. Si el valor de la corriente es cero durante un tiempo mínimo de 25 minutos, se considera que la batería está en estado de reposo y se utiliza la tabla de búsqueda para estimar el SOC a partir del OCV y la temperatura. Si la magnitud del incremento del SOC es suficiente para una estimación de la capacidad, se actualiza el SOH, de lo contrario, se continua acumulando la carga transferida hasta que esta condición se cumpla.

La implementación en código se detalla en la Subsección 5.2.6.

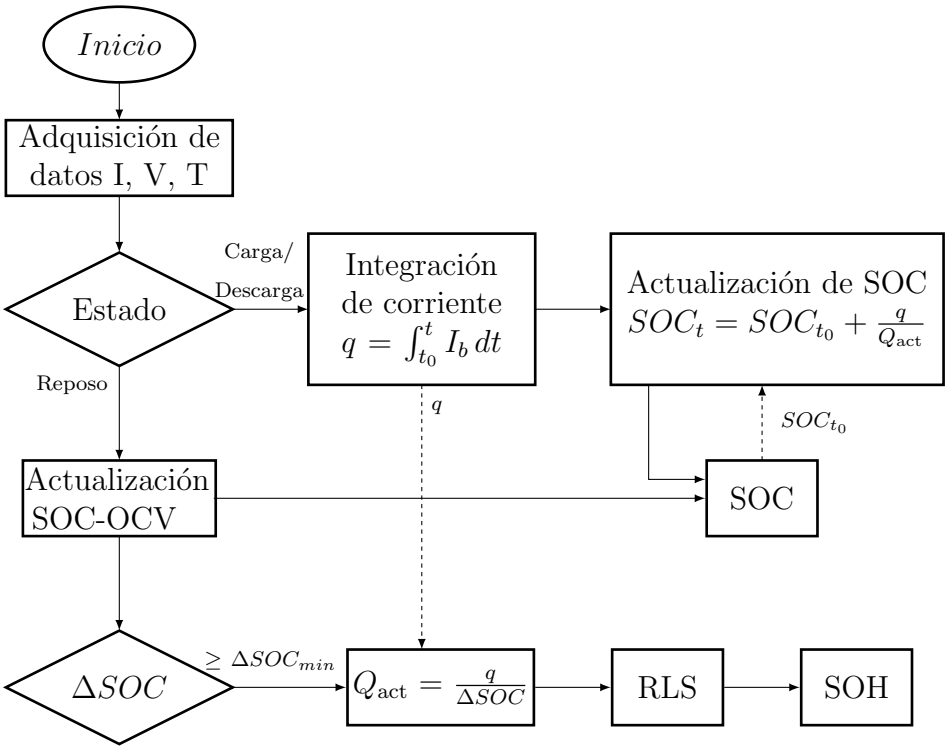


Figura 3.2: Diagrama de flujo del algoritmo implementado.

# Capítulo 4

# Diseño de Hardware

El proyecto propone diseñar un sistema con las siguientes funcionalidades:

- **Medición de parámetros relevantes:** Debe poder medir la tensión, corriente y temperatura de las celdas.
- **Protección de la batería:** El sistema debe supervisar las celdas para prevenir condiciones de riesgo, y debe ser capaz de controlar la corriente en ambos sentidos.
- **Balanceo pasivo de celdas:** Debe contar con un mecanismo de balanceo, permitiendo controlar la diferencia de voltaje entre celdas durante la carga.
- **Comunicación y monitoreo:** El sistema debe poder establecer una comunicación con dispositivos externos mediante los protocolos seriales UART y CAN.
- **Alimentación:** Debe ser capaz de alimentar todos los componentes desde la batería.

Para abordar el diseño del *hardware*, se lo dividió en seis módulos, agrupando los componentes según su finalidad. En la Figura Figura 4.1 se presenta la primera versión completa del AutoBMS, implementada sobre una única placa de circuito impreso. Esta versión integra los módulos de medición, protección, balanceo, comunicaciones y control, que se describen en detalle en las siguientes secciones.

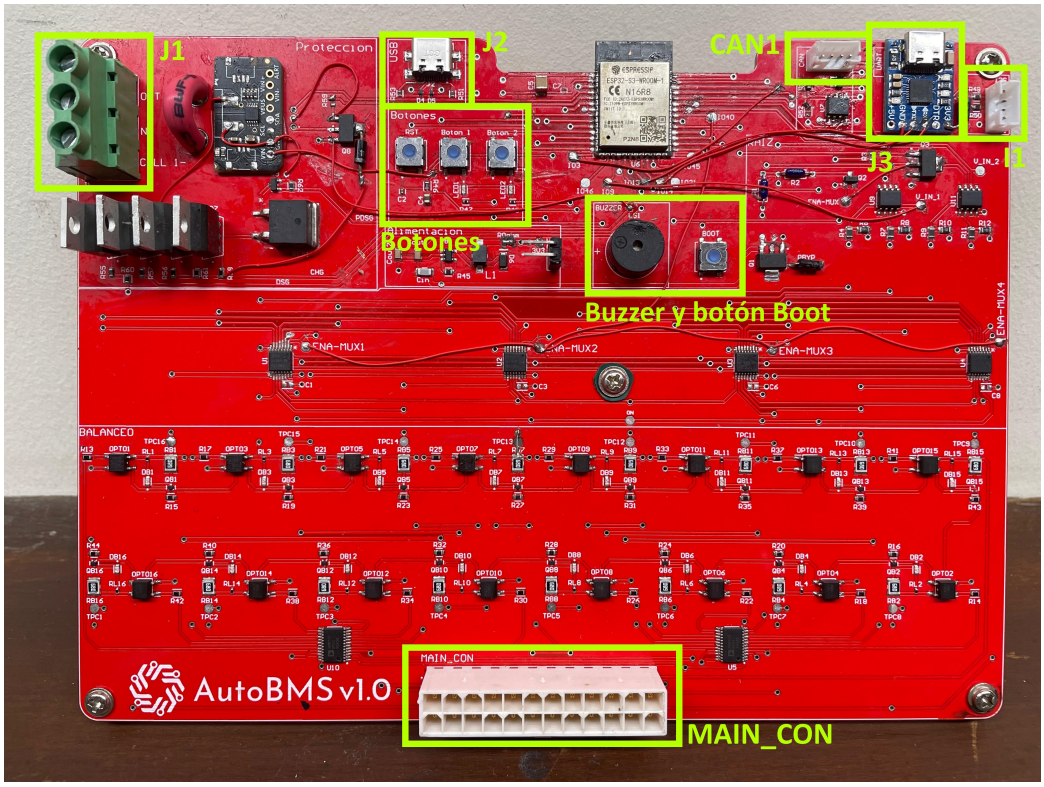


Figura 4.1: Primera versión del Sistema de Gestión de Baterías AutoBMS.

## 4.1. Módulo de Alimentación

Este módulo tiene la función de convertir la tensión de la batería, que puede alcanzar 70 V, a 3,3 V para alimentar el MCU y los demás componentes que funcionan a este voltaje. Para esto se utiliza un conversor DC/DC *step-down* Renesas `RAA211803` [13]. Este regulador está diseñado para aplicaciones de bajo consumo por lo que es compatible con esta aplicación.

Algunas características de interés de este conversor son:

- Tensión de salida fija a 3,3 V.
- Amplio rango de voltaje de entrada: 7 V a 80 V, cubriendo todos los posibles valores de la batería.
- Eficiencia mayor a 65 % para los valores de corriente esperados, en todo el rango de voltajes de entrada.

Su amplio rango de entrada permite el funcionamiento del sistema para todos los niveles de carga de la batería, y al tener tensión de salida fija, el proceso de diseño es más simple que en otros conversores con salida variable.

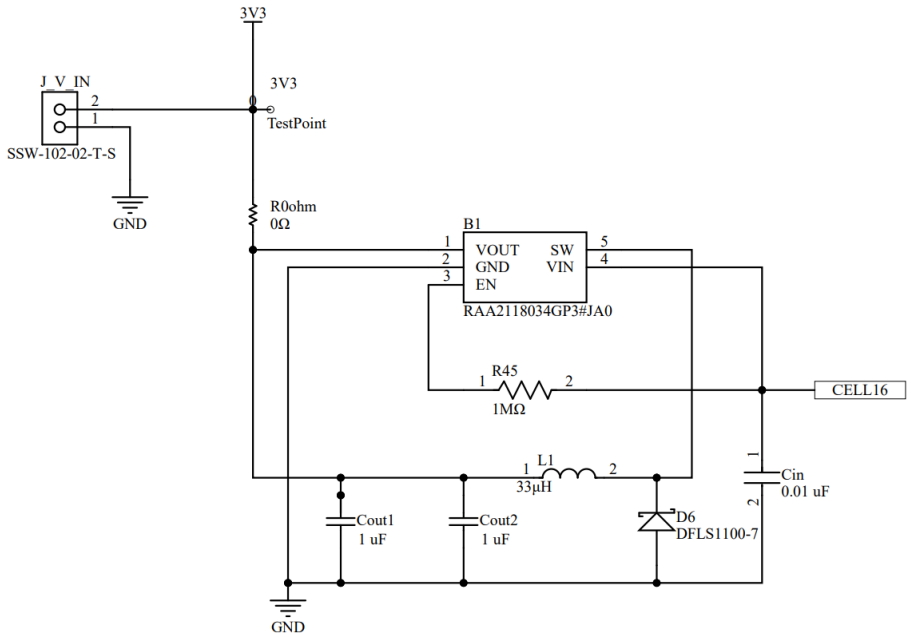


Figura 4.2: Diagrama del módulo de alimentación.

En la Figura 4.2 se encuentra el esquemático de este módulo, donde se puede observar, además de los componentes necesarios para el conversor, una resistencia de 0 Ω y un conector. Estos últimos hacen posible la medición experimental del consumo del dominio de 3,3 V, permitiendo alimentar el sistema con una fuente externa conectada al conector, luego de retirar la resistencia.

## 4.2. Módulo del Microcontrolador

En este módulo se concentran las conexiones al MCU, encargado de realizar la lectura de los sensores y realizar las funciones del BMS.

### 4.2.1. Elección del Microcontrolador

Los principales criterios utilizados en el proceso de selección fueron:

- Disponibilidad de un ADC interno con resolución de al menos 9 bits para tener un error menor a 0,01 V, como se estableció en los requerimientos detallados (ver Sección 2.2).
- Un mínimo de **26 GPIOs** disponibles para las señales digitales del sistema.
- Soporte de modos de bajo consumo (*light-sleep/deep-sleep*) para minimizar pérdidas en estado inactivo. Se estableció como objetivo un consumo promedio menor a 1 mA.

| **Atributo** | ESP32 | ESP32-S2 | ESP32-S3 | ESP32-C6 |
|---|---|---|---|---|
| **CPU / Frecuencia máx.** | Xtensa dual-core LX6 240 MHz | Xtensa dual-core LX7 240 MHz | Xtensa dual-core LX7 240 MHz | 32-bit RISC-V single-core 160 MHz |
| **GPIOs disponibles** | 34 | 43 | 45 | 30 |
| **Protocolos relevantes para el proyecto** | I2C, UART, CAN (TWAI) | I2C, UART, CAN (TWAI) | I2C, UART, CAN (TWAI) | I2C, UART, CAN (TWAI) |
| **Conectividad inalámbrica** | Wi-Fi 4; BT 4.2 BR/EDR; BLE 4 | Wi-Fi 4 | Wi-Fi 4; BLE 5 | Wi-Fi 6; BLE 5; 802.15.4 |
| **ADC (resolución / canales)** | 12 bit; 18 canales | 13 bit; 20 canales | 12 bit; 20 canales | 12 bit; 7 canales |

Tabla 4.1: Comparativa de microcontroladores ESP32 (agosto 2024).

- Ecosistema de programación en C/C++ con amplia documentación y comunidad activa.
- Buena capacidad de procesamiento, garantizando holgura en la ejecución actual y margen para futuras actualizaciones de firmware.
- Conectividad: buses I2C, UART, soporte de CAN.

Dentro de las opciones disponibles, la familia ESP32 de Espressif resultó especialmente atractiva. Esta posee amplia documentación, posee un RTOS con soporte oficial, comunidad activa y múltiples variantes que cumplen los requisitos de conectividad y rendimiento.

Se analizaron las versiones vigentes a la fecha de inicio del diseño (agosto de 2024), evaluando su factibilidad para esta implementación. El primer criterio de descarte fue la cantidad de GPIOs disponibles, reduciendo las posibilidades a los siguientes modelos: ESP32 clásico, ESP32-S2, ESP32-S3 y ESP32-C6 [32] [33] [31] [34]. En la Tabla 4.1 se presenta una comparativa de las características más relevantes de cada uno para la aplicación actual.

Dadas las opciones disponibles se seleccionó el ESP32-S3, por las siguientes razones:

- Ofrece suficiente cantidad de GPIOs con gran margen respecto al mínimo requerido.
- Dispone de Wi-Fi 4 y BLE 5, tecnologías actuales y útiles para comunicación inalámbrica en futuras versiones.

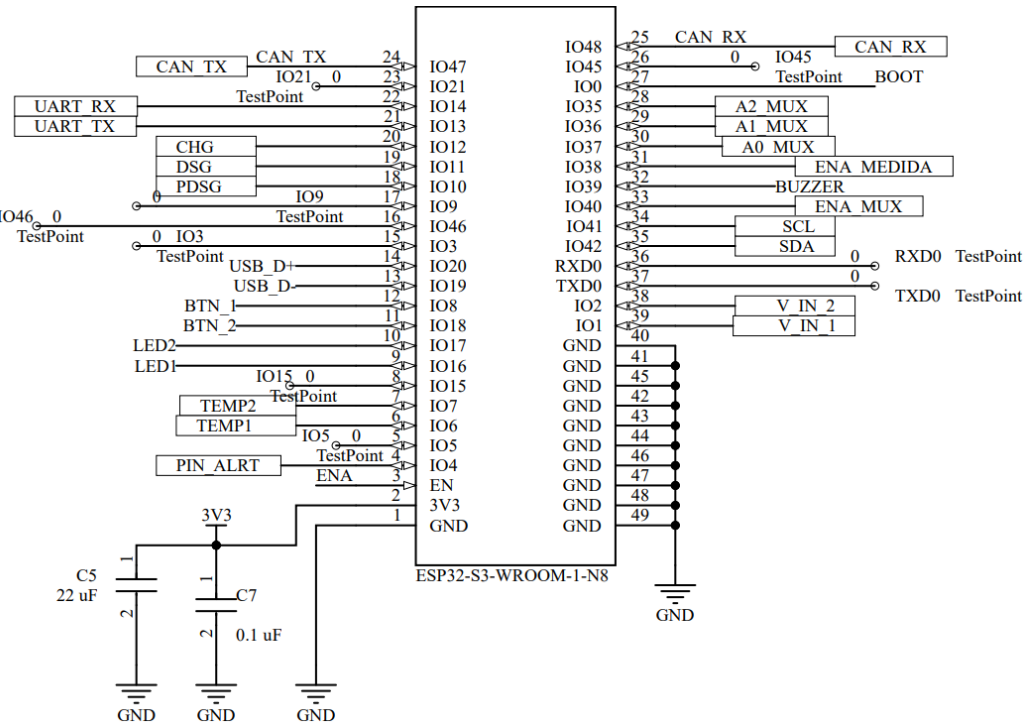


Figura 4.3: Esquemático del MCU.

- Presenta distintos modos de bajo consumo, así como posibilidad de crear un modo de bajo consumo configurado por el usuario.
- Su ADC posee una resolución de 12 bit, cumpliendo con lo requerido.

De esta forma, el `ESP32-S3` constituye un balance óptimo entre rendimiento, disponibilidad de periféricos, consumo y flexibilidad para el futuro crecimiento del proyecto.

Se optó por el módulo `ESP32-S3-WROOM-1-N8`, que posee una memoria flash Quad-SPI de 8 MB, proporcionando un margen holgado para el almacenamiento de configuraciones persistentes y funciones de registro de datos.

En la Figura 4.3 se encuentra el esquemático de la conexión del MCU. Para una lectura más fácil, se provee la lista de pines y sus respectivas señales, en la Tabla 4.2.

En la Figura 4.4 se muestra el esquemático del conector USB utilizado para la programación del *firmware* del MCU. La línea de alimentación del USB no se conecta en este caso, dado que el dispositivo es autoalimentado, es decir, tiene su propia fuente de alimentación.

### 4.2.2. Interfaz con el Usuario

Para la interacción entre el usuario y el BMS se agregan dos pulsadores (además del botón de *reset*), dos LEDs indicadores y un *buzzer*. El diagrama de las cone-

| **GPIO** | **Señal** | **Descripción** |
|---|---|---|
| GPIO0 | BOOT | Pin de arranque; debe forzarse a nivel bajo para entrar en modo descarga. |
| GPIO1 | V_IN_1 | Canal 1 de medición de voltaje de celdas. |
| GPIO2 | V_IN_2 | Canal 2 de medición de voltaje de celdas. |
| GPIO4 | PIN_ALRT | Salida de alerta del sensor de corriente. |
| GPIO6 | TEMP1 | Entrada del sensor 1 de temperatura. |
| GPIO7 | TEMP2 | Entrada del sensor 2 de temperatura. |
| GPIO8 | BTN1 | Entrada digital de pulsador de usuario. |
| GPIO10 | PDSG | Control de MOSFET de predescarga. |
| GPIO11 | DSG | Control de MOSFET de descarga. |
| GPIO12 | CHG | Control de MOSFET de carga. |
| GPIO13 | UART_TX | Salida de transmisión para la interfaz UART de comunicación. |
| GPIO14 | UART_RX | Entrada de recepción para la interfaz UART de comunicación. |
| GPIO16 | LED1 | Salida digital para controlar LED indicador 1. |
| GPIO17 | LED2 | Salida digital para controlar LED indicador 2. |
| GPIO18 | BTN2 | Entrada digital de pulsador de usuario. |
| GPIO19 | USB_D- | Línea diferencial negativa del puerto USB. |
| GPIO20 | USB_D+ | Línea diferencial positiva del puerto USB. |
| GPIO35 | A2_MUX | Bit 2 de la dirección de los multiplexores. |
| GPIO36 | A1_MUX | Bit 1 de la dirección de los multiplexores. |
| GPIO37 | A0_MUX | Bit 0 de la dirección de los multiplexores. |
| GPIO38 | ENA_MEDIDA | Señal de control de la alimentación de los periféricos de medición de voltaje y temperatura. |
| GPIO39 | BUZZER | Salida digital de control para el *buzzer*. |
| GPIO40 | ENA_MUX | Enable de los multiplexores. |
| GPIO41 | SCL | Línea de reloj para bus I2C. |
| GPIO42 | SDA | Línea de datos para bus I2C. |
| GPIO47 | CAN_TX | Señal de transmisión del controlador CAN. |
| GPIO48 | CAN_RX | Señal de recepción del controlador CAN. |

Tabla 4.2: Tabla con la conexión de cada GPIO, junto con su descripción.

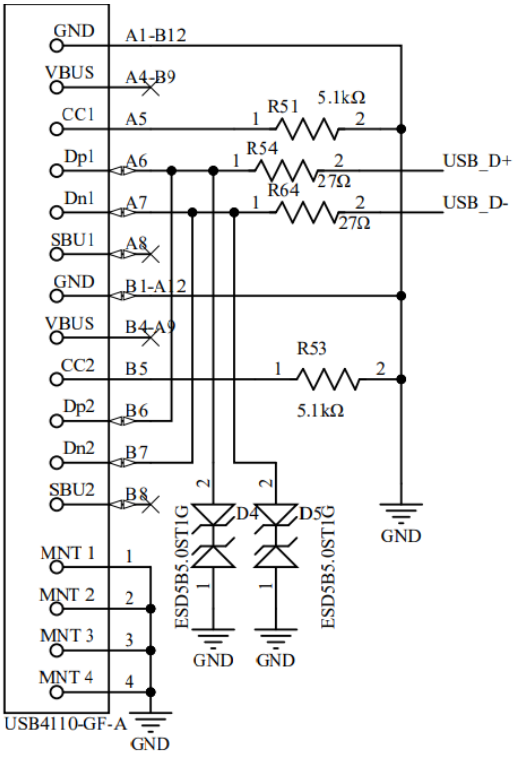


Figura 4.4: Esquemático del conector USB.

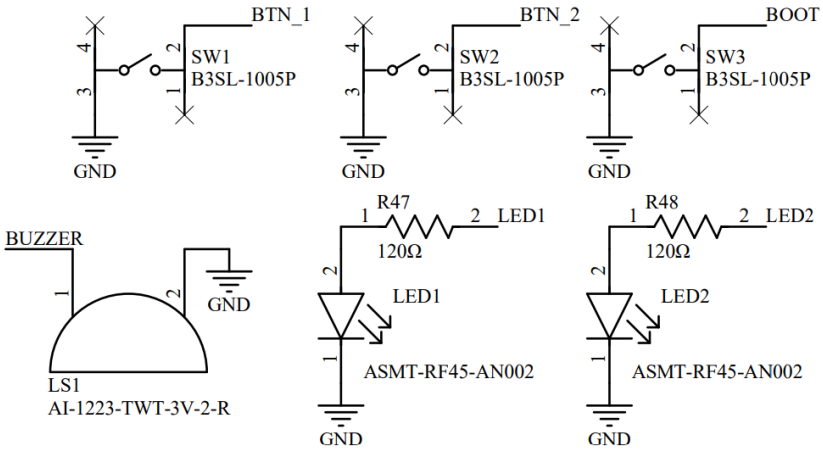


Figura 4.5: Esquemático de la conexión del buzzer, los LEDs y los pulsadores.

xiones de cada componente se encuentra en la Figura 4.5.

Se eligieron LEDs `ASMT-RF45-AN002` de color verde, encapsulado 0603. Según su hoja de datos, enciende con una corriente de $I_F = 10\,\text{mA}$, con un voltaje de $V_F = 2\,\text{V}$ [7]. Dicha corriente es suficientemente pequeña para poder alimentarlos directamente con un GPIO [34], con una resistencia limitadora dado que la tensión de salida es 3,3 V. La resistencia limitadora se calcula de la siguiente manera:

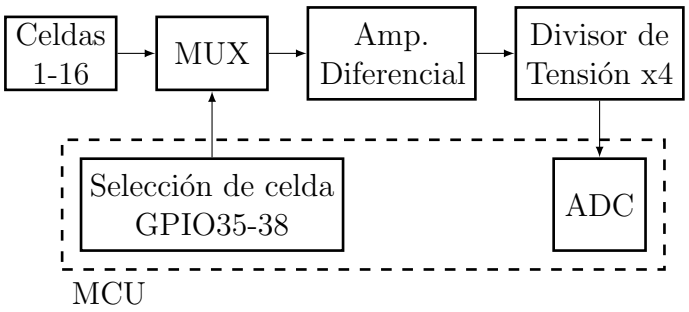


Figura 4.6: Diagrama de bloques del camino de medición de la tensión de las celdas.

$$V_{\text{GPIO}} = 3{,}3\,\text{V} = R_{\text{LED}} \cdot I_F + V_F,$$

por lo que

$$R_{\text{LED}} = \frac{3{,}3\,\text{V} - 2{,}0\,\text{V}}{10\,\text{mA}} = 130\,\Omega.$$

Se utiliza entonces $R_{\text{LED}} = 120\,\Omega$ (serie E12).

El *buzzer* elegido está diseñado para conectarse a una fuente de 3,3 V directamente, por lo que no requiere *hardware* adicional [27].

## 4.3. Módulo de Medición

Este módulo tiene como función la medición del estado del *pack* de baterías. Para ello, se mide el voltaje de cada una de las celdas, la corriente del *pack*, la carga entregada y la temperatura en dos puntos internos a la batería.

### 4.3.1. Medición de Voltajes

El objetivo de este módulo es obtener la tensión individual de cada celda del *pack* de baterías y digitalizarla para su posterior procesamiento. Dado que el sistema debe manejar un número elevado de puntos de medición (16 celdas), se implementó una arquitectura basada en multiplexores, amplificadores diferenciales y el conversor analógico-digital (ADC) integrado en el MCU.

Para medir la tensión de las celdas se empleó un sistema siguiendo el flujo del diagrama de bloques de la Figura 4.6. En la primera etapa, se selecciona la celda a medir mediante multiplexores, cuya salida se conecta a un amplificador diferencial diseñado para señales de alto modo común. De esta forma, se obtiene la tensión de la celda deseada respecto a tierra para ser medida por el MCU. El esquemático del circuito diseñado se puede observar en la Figura 4.7.

En la implementación concreta, se utilizan multiplexores analógicos `TMUX8108`. Estos dispositivos permiten conmutar los dos polos de cada celda hacia un par de amplificadores diferenciales `INA148`, que devuelven la diferencia de tensión entre sus entradas, es decir, la tensión de la celda seleccionada. La señal resultante pasa a continuación por un divisor resistivo que reduce su amplitud en un factor de 4,

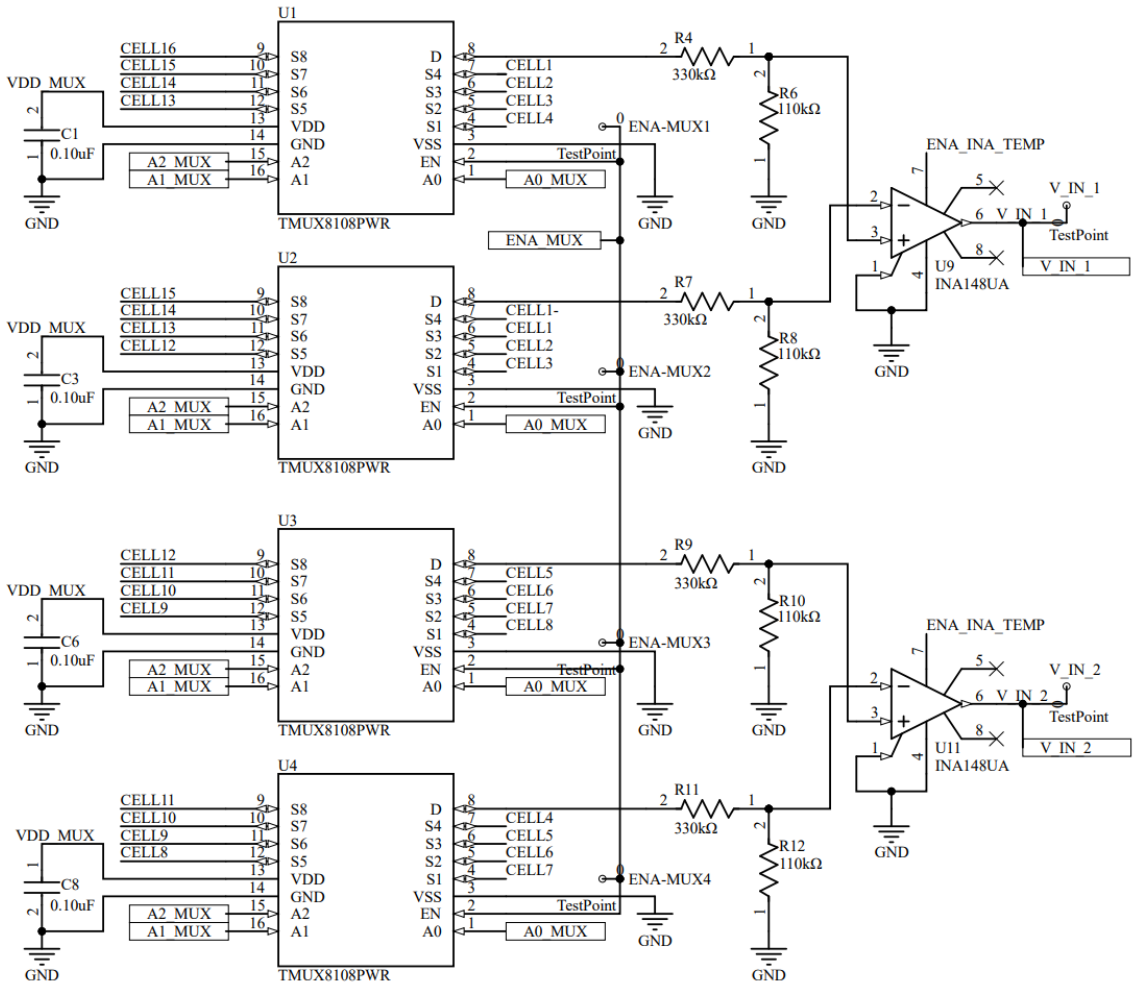


Figura 4.7: Esquemático del circuito de medida de voltaje de las celdas.

garantizando que no se supere la tensión máxima admitida por las entradas del ADC del MCU. Finalmente, la señal acondicionada es digitalizada y almacenada en memoria.

Dada la cantidad de puntos a medir (16) y que los `TMUX8108` cuentan con ocho entradas, el esquema requiere cuatro multiplexores y dos amplificadores diferenciales, lo que permite medir de a dos celdas por vez, utilizando dos de los canales del ADC. El MCU controla la selección de cada celda enviando una dirección de 3 bits correspondiente al número de celda que se desea medir. Esta dirección se transmite a través de los pines `GPIO35`, `GPIO36` y `GPIO37`.

Con el objetivo de reducir el consumo se puede controlar la alimentación de este módulo, utilizando el pin `GPIO38` del MCU (etiquetado `ENA_MEDIDA` en los esquemáticos) el cual acciona la llave de la alimentación a los multiplexores, amplificadores diferenciales y los sensores de temperatura.

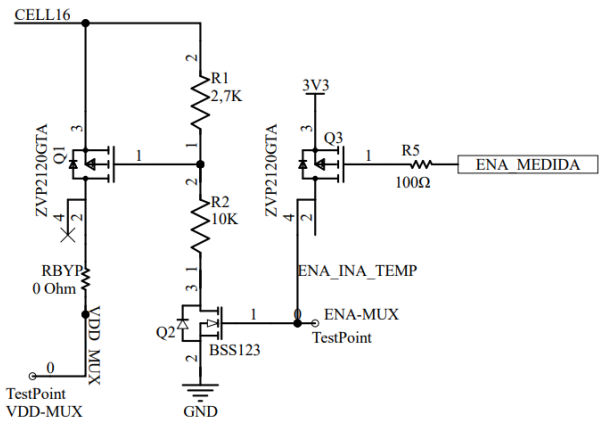


Figura 4.8: Esquemático de la llave de alimentación de los periféricos de medición.

## Llave de alimentación

Con el objetivo de optimizar el consumo, se habilita y deshabilita la alimentación de la electrónica de medida de voltaje y de temperatura de modo que sólo permanezca energizada durante los intervalos de medición. Entre los componentes que se controlan de esta forma se encuentran los sensores `TMP36` y los amplificadores diferenciales `INA148`, que operan a 3,3 V, y los multiplexores, los cuales se alimentan al potencial de la celda 16 ($V_{BAT}$). De esta manera, se reduce el consumo en reposo y se incrementa la eficiencia del sistema.

El circuito empleado para este fin se encuentra en la Figura 4.8. Se debe notar que la señal `ENA_INA_TEMP` visible en el esquemático es la que controla el encendido de los amplificadores diferenciales y los sensores de temperatura. Esta conexión se excluye en el diagrama por motivos de practicidad.

Para conmutar la rama de 3,3 V se utiliza el transistor pMOS `ZVP2120GTA` (Q3). Mientras la salida de `GPIO38` está en nivel alto, se cumple $V_{GS} \approx 0\,\text{V}$ y el dispositivo permanece en corte; cuando la salida se lleva a 0 V, se obtiene $V_{GS} \approx -3{,}3\,\text{V}$, valor suficiente para superar la tensión de umbral, que según su hoja de datos se encuentra entre $-1{,}5\,\text{V}$ y $-3{,}5\,\text{V}$ [11].

La señal de salida de Q3 se emplea también para habilitar la alimentación de los multiplexores, que trabajan al potencial de $V_{BAT}$. Dado que se requiere conmutar una tensión superior al dominio lógico, se implementó un *driver* compuesto por un transistor nMOS `BSS123` (Q2) y otro pMOS `ZVP2120GTA` (Q1). En esta configuración, la señal proveniente de Q3 se conecta al *gate* de Q2, que tiene su *source* conectado a tierra. El *drain* de Q2 se conecta a un divisor resistivo formado por $R_1$ y $R_2$, que genera, desde $V_{BAT}$, el nivel de tensión adecuado para que Q1 conduzca. Este último se ubica entre $V_{BAT}$ y los pines $V_{DD}$ de los `TMUX8108`, permitiendo cortar o habilitar la alimentación según corresponda. Con esta topología se asegura que el *gate* de Q1 nunca supere su tensión máxima y, al mismo tiempo, se garantiza un encendido confiable de los multiplexores [11, 23].

Se utilizan los `ZVP2120GTA` por su voltaje *drain-source* máximo de $V_{DS} = -200\,\text{V}$ y potencia máxima de 2 W, características holgadas para esta función de

conmutación de baja corriente. Además, su voltaje de umbral $V_{GS_{th}}$ permite su conmutación con señales de 3,3 V.

#### Elección de $R_1$ y $R_2$

Los valores de $R_1$ y $R_2$ deben cumplir con la siguiente condición:

$$|V_{GS_{th}}| \leq |V_{GS}| = V_{BAT} \cdot \frac{R_1}{R_1 + R_2}$$

para lograr que el transistor Q1 conduzca.

Esta relación debe verificarse en todo el rango de voltajes de las celdas. En particular, cuando el estado de carga es 0 %, el voltaje de la batería es aproximadamente $V_{BAT} = 16 \cdot 3\,\text{V} = 48\,\text{V}$.

Para reducir las pérdidas de potencia en Q1, la resistencia en conducción $R_{DS(on)}$ debe ser lo más baja posible. De acuerdo con la hoja de datos, esto ocurre cuando $|V_{GS}| \geq 10\,\text{V}$.

Con los valores seleccionados de $R_1 = 2{,}7\,\text{k}\Omega$ y $R_2 = 10\,\text{k}\Omega$, se obtiene $|V_{GS}| \approx 10{,}2\,\text{V}$ para SOC = 0 % y $|V_{GS}| \approx 14{,}3\,\text{V}$ para SOC = 100 %, satisfaciendo los requerimientos, y manteniendo un margen respecto del valor máximo de $V_{GS_{MAX}} = -20\,\text{V}$ [10].

### 4.3.2. Medición de Temperatura

Dada la naturaleza electroquímica de las baterías, el relevamiento de su temperatura es de suma importancia para garantizar su seguridad y proteger la vida útil de las celdas. Para seleccionar un transductor de temperatura adecuado, el primer paso es conocer en qué rango de temperatura pueden trabajar las celdas de iones de litio NMC, dado que este debe ser capaz de medir en todo este rango.

Diversas hojas de datos para celdas NMC cilíndricas comerciales establecen ventanas de operación coherentes para carga y descarga. Se estudió el caso de tres modelos de celdas muy difundidas en la industria. La LG HG2 especifica carga entre 0 y 50 °C y descarga entre −20 °C y 75 °C [20]; la LG MH1 indica carga entre 0 y 45 °C y descarga entre −20 °C y 60 °C [19]; y la Samsung INR18650-25R define carga entre 0 y 50 °C y descarga entre −20 °C y 75 °C [28].

Bajo estos requisitos, la selección del sensor `TMP36` resulta adecuada, considerando su rango de operación especificado de −40 °C a 125 °C [9].

La salida lineal con sensibilidad de 10 mV/°C y precisión típica de ±2 °C en todo el rango simplifica el acondicionamiento de señal y permite establecer umbrales de protección con márgenes adecuados para el BMS. Por otra parte, su consumo máximo de 50 µA resulta aceptable para una aplicación de bajo consumo como este sistema, especialmente considerando que la alimentación del sensor se habilita únicamente durante los intervalos de medición, con el mismo *switch* que habilita el módulo de medición de voltaje.

En la solución implementada, el BMS posee entradas para dos sensores `TMP36`, posicionados de modo de medir la temperatura en dos puntos distintos del *pack* y así reducir el tiempo de respuesta en caso de que alguna celda aumente su

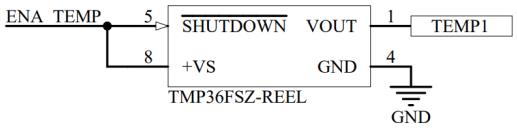


Figura 4.9: Esquemático del circuito de medida de temperatura.

temperatura de manera riesgosa. Cada sensor va montado en una PCB en contacto directo con las celdas. En la Figura 4.9 se presenta la conexión de estos sensores.

### 4.3.3. Medición de Corriente y Carga

La medición de corriente en el *pack* de baterías es fundamental tanto para la estimación de carga con CC, como para la protección del sistema ante sobrecorrientes. Para este propósito se diseñó un circuito basado en el integrado `INA228` de Texas Instruments [17].

El `INA228` es un monitor de potencia digital de alta precisión que, mediante una resistencia *shunt*, permite la medición bidireccional de la corriente del sistema. Además, integra funcionalidades adicionales que facilitan la estimación de la energía transferida y la compensación de errores. Entre sus principales características se destacan:

- Rango de alimentación de 2,7 V a 5,5 V.
- Comunicación digital I2C de alta velocidad (hasta 2,94 MHz) con 16 direcciones seleccionables por hardware.
- Resolución de 20 bits en las mediciones.
- Registro interno de acumulación de carga.
- Tiempos de conversión configurables entre 50 µs y 4,12 ms.
- Promediado programable entre 1 y 1024 muestras.
- Compensación programable de la resistencia *shunt* en función de la temperatura.

#### Selección de la Resistencia *Shunt*

Para cumplir con los requerimientos del sistema, es necesario medir corrientes en ambos sentidos con un rango amplio, ya que el diseño debe soportar corrientes de hasta 30 A.

El rango permitido de caída de tensión sobre la resistencia *shunt* en la configuración utilizada es de ±40,96 mV. A partir de esta restricción se obtiene la condición para la resistencia $R_S$:

$$R_S \cdot I_{MAX} \leq 40{,}96\,\text{mV} \quad \Longrightarrow \quad R_S \leq \frac{40{,}96\,\text{mV}}{30\,\text{A}} = 1{,}37\,\text{m}\Omega$$

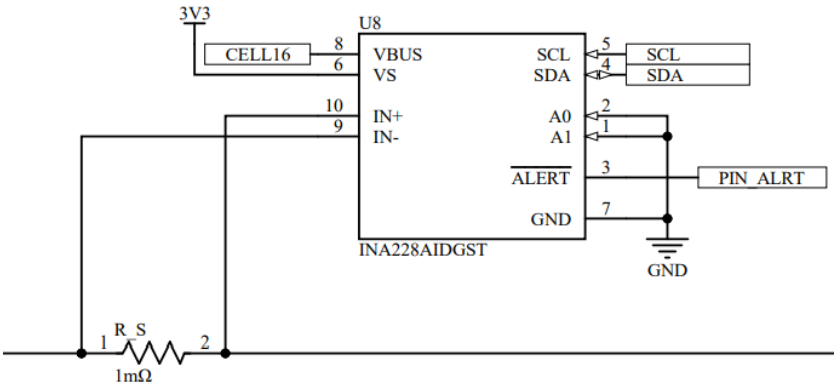


Figura 4.10: Esquemático del circuito de medida de corriente.

De acuerdo con este cálculo, se seleccionó una resistencia *shunt* de 1 mΩ, con tolerancia del 1 % y potencia nominal de 5 W. Estas características superan los requerimientos del diseño, proporcionando un margen de seguridad frente a condiciones de operación extremas. El circuito diseñado se presenta en la Figura 4.10.

## 4.4. Módulo de Protección

Este módulo integra las llaves que controlan el flujo de corriente en ambos sentidos del *pack*: saliente (descarga) y entrante (carga), e implementa el mecanismo de predescarga.

Para ambas direcciones se emplean MOSFETs en configuración *back-to-back* (*source-común*), lo cual permite bloquear la conducción a través de los diodos intrínsecos antiparalelos y controlar de manera simétrica el paso de corriente. La rama de predescarga se conecta en paralelo a la rama de descarga, pero incorpora en serie una resistencia que limita la corriente máxima; su función es cargar las capacitancias presentes en la carga antes de cerrar la rama principal, reduciendo picos de corriente y esfuerzos eléctricos.

El control se realiza desde el MCU mediante tres pines digitales:

- **GPIO12** — **CHG**: habilita la rama de carga.
- **GPIO11** — **DSG**: habilita la rama de descarga.
- **GPIO10** — **PDSG**: habilita la rama de predescarga.

Se puede visualizar el módulo completo de protección en Figura 4.11

### 4.4.1. Elección de los MOSFETs

Los parámetros de mayor relevancia para esta aplicación son:

- **Tensión *drain-source*** $V_{DS}$: debe superar con margen el máximo voltaje del *pack*.

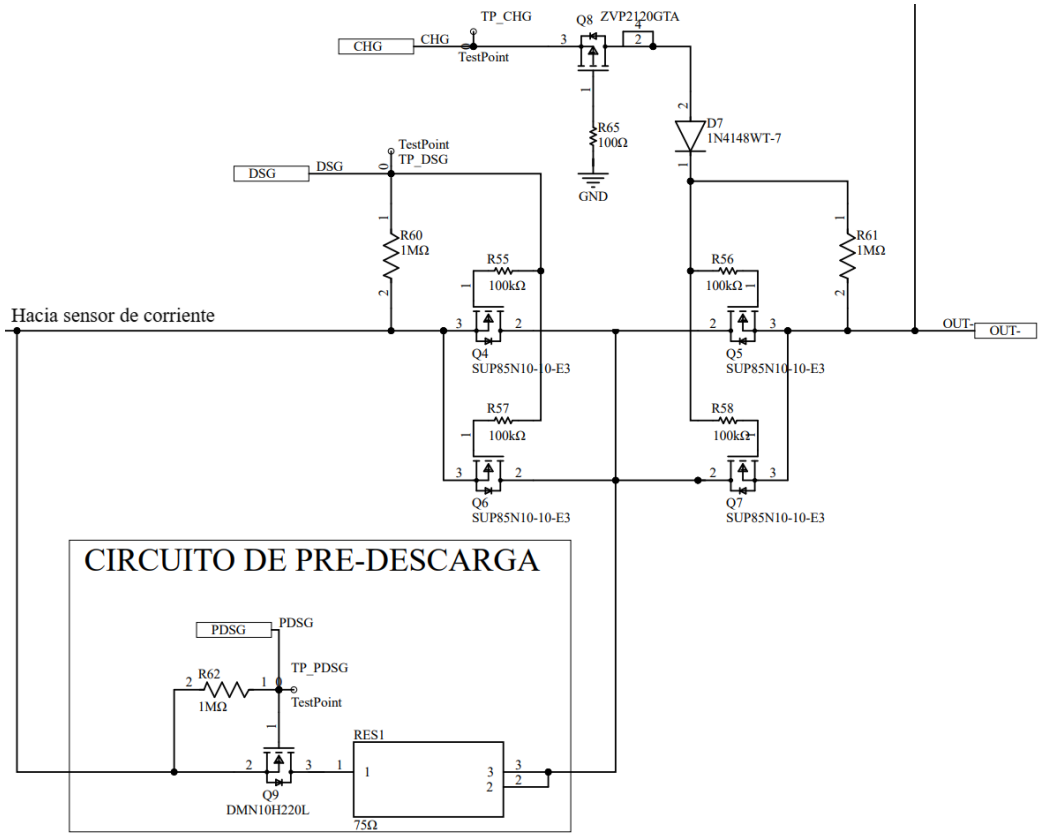


Figura 4.11: Esquemático de los circuitos de corte y predescarga.

- **Corriente de *drain* continua** $I_{\mathrm{D}}$: debe cubrir la corriente máxima de diseño en régimen constante.

- **Resistencia en conducción** $R_{\mathrm{DS(on)}}$: determina las pérdidas por conducción; cuanto menor, mejor.

## Criterios de dimensionamiento

Se adopta un margen de diseño del $\sim 20\,\%$ sobre el voltaje máximo del *pack* para $V_{\mathrm{DS}}$. Con 16 celdas de iones de litio NMC y voltaje máximo por celda de 4,2 V, resulta:

$$V_{DS} \;\geq\; 1,2 \times 16 \times 4,2\,\mathrm{V} \;=\; 80,64\,\mathrm{V}.$$

La corriente máxima en régimen se fija en $I_D = 30\,\mathrm{A}$, por diseño. Respecto de $R_{DS(on)}$, se busca minimizar las pérdidas; el uso de dispositivos en paralelo por rama reduce la resistencia efectiva y distribuye la disipación térmica (sin disipadores, con sobredimensionamiento).

| **Parámetro** | **Requerimiento mínimo del sistema** | **SUP85N10-10** |
|---|---|---|
| Tensión $V_{DS}$ | $\geq 80{,}64\,\text{V}$ | $100\,\text{V}$ |
| Corriente $I_D$ continua | $\geq 30\,\text{A}$ | $85\,\text{A}$ @ $T_C = 25\,°\text{C}$ |
| $R_{DS(on)}$ @ $V_{GS} = 4{,}5\,\text{V}$ | Minimizar | $10\,\text{m}\Omega$ typ., $12\,\text{m}\Omega$ máx. |

Tabla 4.3: Requerimientos mínimos del sistema y valores del MOSFET seleccionado.

#### Componente seleccionado

Se seleccionaron MOSFETs Vishay `SUP85N10-10`. Sus características relevantes son: $V_{DS} = 100\,\text{V}$, $I_D = 85\,\text{A}$ y $R_{DS(on)} = 10\,\text{m}\Omega$ típico y $12\,\text{m}\Omega$ máximo (a $V_{GS} = 4{,}5\,\text{V}$ e $I_D = 20\,\text{A}$) [38]. Estos valores cumplen con margen los requerimientos definidos en Sección 2.2, representado en la Tabla 4.3.

### 4.4.2. Circuito de Predescarga

El circuito de predescarga es un elemento fundamental para garantizar el correcto funcionamiento y la seguridad del sistema. Su función principal es limitar los transitorios de corriente que se producen durante la conexión inicial de la carga, causados por la presencia del condensador del *DC-link*. Este condensador actúa como un reservorio de energía, suavizando las fluctuaciones de tensión del bus de corriente continua al variar el consumo de las cargas *aguas abajo* de este.

Tal como se describe en [35], los circuitos de predescarga son ampliamente utilizados en vehículos eléctricos. El esquema típico incluye dos contactores principales de corriente elevada y un contactor de predescarga separado, con el condensador del *DC-link* en paralelo con la carga.

Durante la conexión inicial, la diferencia de potencial entre el banco de baterías y el condensador del *DC-link* puede provocar una corriente abrupta de gran magnitud, capaz de dañar los MOSFETs, contactores o pistas de potencia si no se limita adecuadamente. El circuito de predescarga mitiga este efecto mediante una resistencia de potencia conectada en serie con un transistor, que permite cargar gradualmente el condensador hasta alcanzar un nivel seguro de tensión antes de habilitar el camino principal de descarga. La implementación del sistema de predescarga se encuentra recuadrado en la Figura 4.11.

#### Resistencia del Sistema de Predescarga

Para estimar la energía y los tiempos involucrados en la etapa de predescarga, se requiere conocer la capacitancia total del *DC-link*. Con el fin de obtener valores representativos de la capacitancia en sistemas similares al presente proyecto (con voltajes de salida de hasta $70\,\text{V}$), se analizaron dos controladores comerciales de uso difundido en aplicaciones de movilidad eléctrica ligera:

1. **STMicroelectronics 48 V Low-Voltage Traction Inverter Kit:** módulo de tracción de baja tensión que incluye una placa de capacitores de bus con una capacitancia total de aproximadamente C = 6 mF [29].

2. **BorgWarner / Sevcon Gen4 Size 8 Traction Inverter:** inversor para sistemas de 80 V a 400 V, cuya documentación técnica especifica una capacitancia del *DC-link* interna de C = 1,89 mF [6].

En los cálculos que se presentan a continuación se adoptará el valor más alto de los anteriores, es decir, $C = 6\,\text{mF}$, con el objetivo de dimensionar el circuito de predescarga en la condición más exigente. Además, por el mismo motivo, se considera que el condensador comienza el ciclo completamente descargado.

La carga del condensador del *DC-link* desde el estado completamente descargado puede modelarse como un proceso de primer orden, descrito por la expresión

$$V_C(t) = V_{BUS}\left(1 - e^{-t/RC}\right),$$

donde $R$ representa la resistencia de predescarga y $V_{BUS}$ la tensión del bus de continua. El tiempo característico del proceso está dado por la constante de tiempo $\tau = RC$, que determina la velocidad de carga del condensador.

Generalmente se espera que el circuito de predescarga esté activo hasta que el condensador alcance un 90 % del voltaje del bus [2]. No obstante, algunos diseñadores adoptan un margen más conservador de $5\tau$ para considerar el ciclo de predescarga finalizado [36]. En el presente diseño se adopta este criterio para asegurar una carga plena del condensador antes de habilitar el camino principal de descarga.

Se define un tiempo máximo de predescarga de 2 s. De acuerdo con el modelo exponencial, la resistencia necesaria para cumplir con esta condición se calcula como:

$$R = \frac{t}{5C}.$$

Reemplazando $t = 2\,\text{s}$ y $C = 6\,\text{mF}$, se obtiene: $R = 66{,}6\,\Omega$.

Por disponibilidad y para reducir aún más la corriente inicial, se utiliza una resistencia de valor $R = 75\,\Omega$, resultando en un tiempo máximo de $t = 2{,}25\,\text{s}$.

De los componentes involucrados, la resistencia es quien disipa más energía.

Se eligió una resistencia `TDH35P75R0JE`, cuya potencia nominal es de 35 W y su tensión máxima de operación es de 350 V. Según su hoja de datos, admite una disipación en régimen de pulso de hasta 70 W, para una tensión de hasta $1{,}5 \times V_{nom}$ y durante un período inferior a 5 s [25].

La máxima potencia se da en el inicio, con un valor de $P = \frac{V_{BUS}^2}{R} = 65{,}3\,\text{W}$. Como la duración máxima de este pulso es de 2,25 s, como se calculó anteriormente, la resistencia está bien dimensionada para esta función.

### Transistor del Sistema de Predescarga

El dispositivo de conmutación utilizado en el sistema de predescarga es un MOSFET nMOS modelo `DMN10H220L`. Este transistor se encuentra en el lado bajo del circuito, con el *source* conectado a la tierra del sistema y el *drain* conectado a la resistencia de predescarga. El *gate* es manejado directamente desde el MCU mediante el pin `GPIO10`. Esto es posible dado su voltaje de umbral $V_{GS(th)}$ entre 1,0 V y 2,5 V, lo que garantiza una conducción adecuada con $V_{GS} = 3{,}3\,\text{V}$ [12].

El dispositivo presenta una tensión máxima *drain-source* $V_{DS(max)} = 100\,\text{V}$, una resistencia de conducción máxima $R_{DS(on)} = 250\,\text{m}\Omega$ para $V_{GS} = 4{,}5\,\text{V}$, y una corriente continua de hasta 1,3 A, admitiendo pulsos de hasta 8 A.

Durante la fase inicial de predescarga, la corriente máxima teórica es $I_0 = V_{BUS}/R = 0{,}93\,\text{A}$, valor que se encuentra dentro de los márgenes de corriente permitidos.

La potencia instantánea máxima disipada en este transistor puede calcularse como:

$$P_{max} = I_0^2\, R_{DS(on)} = 0{,}22\,\text{W},$$

muy por debajo del límite de 1,3 W a 25 °C indicado en la hoja de datos [12].

El dispositivo, por tanto, opera con amplio margen de seguridad tanto en tensión como en potencia y corriente, validando su elección para este diseño.

## 4.5. Módulo de Balanceo

Este módulo implementa el balanceo pasivo de las celdas, conectando en paralelo una carga resistiva sobre aquellas que lo requieran.

La principal dificultad radica en que cada celda se encuentra a un voltaje distinto respecto a la tierra del sistema: la celda $n$ está elevada aproximadamente $n \times V_{celda}$. Por esta razón, no es viable ni seguro controlar de forma directa las cargas con salidas GPIO del MCU.

Para resolverlo se adoptó una arquitectura en dos niveles:

1. **Selección lógica a bajo voltaje**, realizada con bloques de *switches* analógicos controlados por el MCU. A diferencia de un multiplexor, los bloques de *switches* permiten habilitar más de un canal de manera simultánea, necesario cuando varias celdas requieren balanceo a la vez.

2. **Conmutación galvánicamente aislada** en cada celda, mediante optoacopladores. La salida de cada uno queda referida al polo negativo de la propia celda, con lo cual la rama de balanceo trabaja al potencial de esa celda sin comprometer el dominio de 3,3 V del MCU.

El diagrama de bloques de la arquitectura propuesta se encuentra en la Figura 4.12.

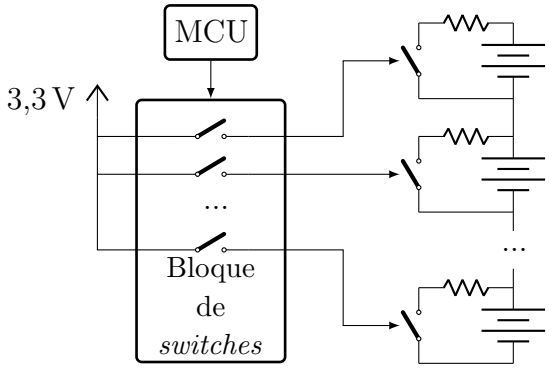


Figura 4.12: Diagrama de bloques de la arquitectura del módulo de balanceo implementado.

### 4.5.1. Implementación

#### Bloques de *Switches*

Los bloques de *switches* elegidos son los `ADG715`, de Analog Devices. Cada integrado ofrece 8 interruptores analógicos, con alimentación a 3,3 V y control mediante interfaz I2C, lo que permite una interfaz simple con el MCU. Su consumo máximo es de 20 µA [4]. Para cubrir las 16 celdas del sistema se emplean dos bloques en paralelo, lo que permite disponer de 16 líneas de control independientes.

Cada uno de los *switches* conecta o desconecta la fuente de 3,3 V a la entrada de control de un optoacoplador `CPC1001N`, a través de una resistencia limitadora. La señal de salida controla el *gate* del MOSFET que, funcionando como llave, conecta la carga resistiva de balanceo sobre su celda correspondiente (se profundiza sobre su diseño a continuación).

#### Optoacopladores

Se seleccionaron los optoacopladores `CPC1001N`. Proporcionan aislamiento galvánico de 1500 V entre entrada y salida, es decir, entre el dominio lógico (3,3 V) y cada celda. Conmutan la rama resistiva a potencial de la celda, evitando referencias a masa del sistema.

Una ventaja de este modelo de optoacoplador es su baja corriente de activación, permitiendo una corriente $I_C = 0{,}4\,\text{mA}$ con una corriente a su entrada $I_F = 0{,}2\,\text{mA}$, según curvas en su hoja de datos [21].

#### Dimensionamiento de la resistencia de límite

Según la hoja de datos de los `ADG715`, la resistencia de los *switches* en estado cerrado es de 6 Ω típicamente, para una alimentación de 3 V. La corriente de la entrada de activación del diodo interno al optoacoplador es de $I_{opt} = 0{,}2\,\text{mA}$, con una caída de tensión típica de $\Delta\text{V} = 1{,}2\,\text{V}$ [21]. Planteando la rama de la entrada de control, se tiene:

$$I_{opt}\,(R_1 + R_{\text{switch}}) + \Delta\text{V} = V_{DD} = 3{,}3\,\text{V} \implies R_1 \approx 10{,}5\,\text{k}\Omega.$$

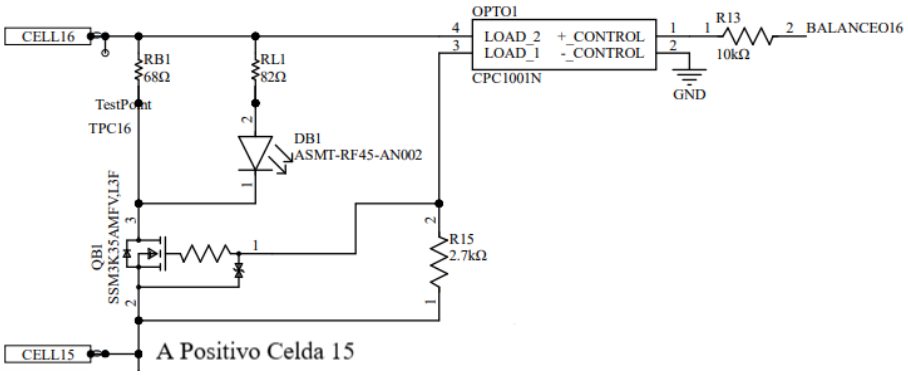


Figura 4.13: Carga resistiva de una celda para balanceo pasivo.

Se elige un valor de la serie E12, por lo que $R_1 = 10\,\text{k}\Omega$.

### Carga Resistiva de las Celdas

En la Figura 4.13 se muestra el circuito para una sola celda, el cual incluye además un diodo LED que se enciende durante el balanceo, funcionando como indicador visual de que se está balanceando cada celda. El módulo final se compone de dieciséis de dichos circuitos conectados a cada una de las celdas.

El diseño busca controlar el paso de corriente por la carga pasiva, de modo de disipar energía de la celda. Esto se logra con el MOSFET ($Q_{B1}$ en el esquemático), el cual se activa al encender el optoacoplador.

### Dimensionamiento de la rama de balanceo

Las corrientes de balanceo interno en BMS comerciales suelen encontrarse en el rango de 20 mA a 100 mA, tal como se reporta en [16]. Este rango se debe a un compromiso entre la velocidad de igualación de carga entre celdas y la disipación térmica en resistencias y transistores. Para este diseño se fijó una corriente de balanceo mínima de 50 mA en todo el rango de tensión de la celda. Este es un valor típico de corriente de balanceo, que asegura una disipación moderada en la PCB y es suficiente para corregir desbalances típicos en *packs* de iones de litio NMC [16].

Para garantizar el correcto funcionamiento de los transistores se realizaron los cálculos considerando $V_{\text{celda}} = 3{,}2\,\text{V}$, siendo este el mínimo voltaje que las celdas pueden tener en condiciones seguras de operación.

La corriente de salida del optoacoplador es $I_C = 0{,}4\,\text{mA}$ para $V_{CE} > 0{,}3\,\text{V}$ [21]. Se seleccionaron transistores `SSM3K35AMFV` para los $Q_B$, que presentan tensión de umbral bajo ($V_{th} \leq 1\,\text{V}$) y encapsulado compacto [37], condición importante dado que el circuito se replica 16 veces en la PCB.

De la curva $I_D$–$V_{DS}$ de la hoja de datos, con $V_{GS} = 1\,\text{V}$ se obtiene $I_D \in [50\,\text{mA}, 60\,\text{mA}]$ para $V_{DS} \geq 0{,}4\,\text{V}$.

Fijando $V_{GS} = 1\,\text{V}$, y dado que $V_{GS} = I_C \cdot R_{15}$, $R_{15} = 2{,}5\,\text{k}\Omega$. Se selecciona $R_{15} = 2{,}7\,\text{k}\Omega$, valor de la serie E12.

Como se requiere $V_{DS} \geq 0{,}4\,\text{V}$, con $V_{\text{celda}} = 3{,}2\,\text{V}$ se obtiene que sobre la carga habrá al menos $V_{\text{carga}} = 2{,}8\,\text{V}$. Por lo tanto, el LED seleccionado debe operar correctamente a esta tensión.

Los LEDs son del mismo modelo que los que se utilizan en las cargas resistivas de balanceo (ver Subsección 4.2.2), pero esta vez alimentados desde la celda, que tiene un voltaje distinto a la alimentación del MCU. Esto requiere una resistencia limitadora de distinto valor al utilizado en el primer caso. Considerando que los LEDs cumplen $I_f = 10\,\text{mA} \,@\, V_F = 2\,\text{V}$, el valor de dicha resistencia se determina de la siguiente forma:

$$V_{\text{celda}} = R_{\text{LED}} \cdot I_F + V_F + V_{DS},$$

con $V_{GS} = 0{,}4\,\text{V}$.

Por lo tanto, $R_{\text{LED}} = 80\,\Omega$, y se selecciona $R_{\text{LED}} = 82\,\Omega$ de la serie E12.

Considerando la corriente total de balanceo (50 mA), la corriente a través de la resistencia $R_B$ debe ser: $I_{RB} = 50\,\text{mA} - 10\,\text{mA} = 40\,\text{mA}$.

Con $V_{R_B} = V_{\text{celda}} - V_{DS} = 2{,}8\,\text{V}$, se obtiene $R_B = 70\,\Omega$.

Se selecciona $R_B = 68\,\Omega$ (E12). La potencia disipada es:

$$P_{R_B} = \frac{V_{RB}^2}{R_B} \approx 115\,\text{mW},$$

valor considerado a la hora de elegir la potencia de $R_B$. Se seleccionan así resistencias de 500 mW para tener un amplio margen de disipación.

Finalmente, se verifica que la potencia disipada en $Q_B$ para $V_{\text{celda,max}} = 4{,}2\,\text{V}$ no supere su límite:

$$P_{Q_B} = I_{bal} \cdot (V_{\text{celda,max}} - V_{\text{carga}}) = 50\,\text{mA} \cdot (4{,}2\,\text{V} - 2{,}8\,\text{V}) = 70\,\text{mW} < 150\,\text{mW}$$

, cumpliendo con lo especificado en la hoja de datos [37].

## 4.6. Módulo de Comunicación

El módulo de comunicación integra las interfaces externas necesarias para el funcionamiento del sistema. Incluye submódulos para USB–UART, I2C y CAN.

### 4.6.1. Módulo CAN

La interfaz CAN se implementa mediante el transceptor `SN65HVD233` de Texas Instruments, un dispositivo compatible con el estándar ISO 11898. Este modelo fue seleccionado por su bajo consumo de corriente en operación normal y en modo de espera, en comparación con variantes anteriores de la misma familia como el `SN65HVD230`. Además, ofrece protección contra cortocircuitos y sobrecargas, soporta velocidades de hasta 1 Mbit/s y permite operación en topologías de bus típicas de aplicaciones del mundo automotriz, incluyendo vehículos eléctricos.

El esquemático de este módulo se encuentra en la Figura 4.14.

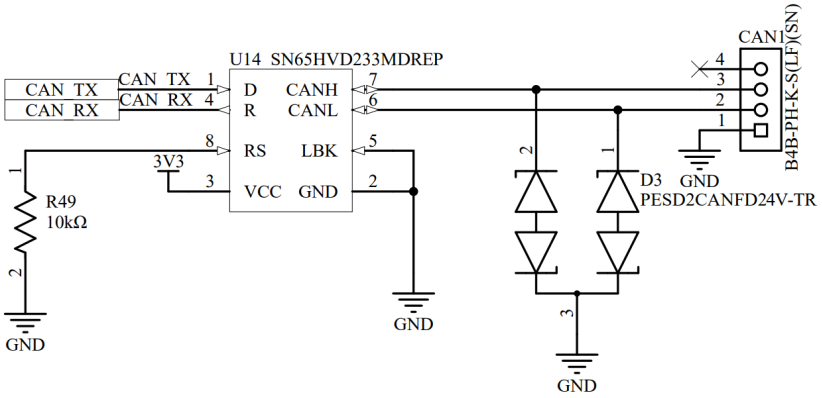


Figura 4.14: Esquemático del circuito de comunicación CAN.

### 4.6.2. Módulo USB–UART

La comunicación serie con una PC o dispositivo externo se resuelve a través del convertidor USB–UART `CP2102` de Silicon Labs. Es un integrado ampliamente utilizado para este fin, dado que es una solución sencilla y confiable para la conversión de niveles USB 2.0 a UART, requiriendo un número mínimo de componentes externos. Entre sus características más relevantes se destacan:

- Soporte de velocidades de transmisión de hasta 1 Mbit/s.
- Compatibilidad con los principales sistemas operativos mediante *drivers* oficiales, garantizando facilidad de uso y confiabilidad en la comunicación.

Una ventaja adicional es que el dispositivo toma su alimentación directamente del puerto USB del PC al que se conecta, por lo que mientras no se utiliza no genera consumo en el BMS.

### 4.6.3. Interfaz I2C

Con el fin de facilitar el diagnóstico y la depuración del sistema, se incluyó un conector para el bus I2C. Este conector permite conectar un analizador lógico externo o equipos de medición, posibilitando la observación en tiempo real de la comunicación entre el MCU y los periféricos. Esta adición, además de permitir la validación del sistema durante el desarrollo, otorga la posibilidad de agregar sensores o periféricos externos que se comuniquen a través de este protocolo a futuro. El conector y las resistencias de *pull-up* para el bus I2C se pueden ver en el esquemático de la Figura 4.16.

## 4.7. Diseño de PCB

Los módulos mencionados anteriormente se integraron en una PCB, cuyo modelo 3D se puede encontrar en la Figura 4.17.

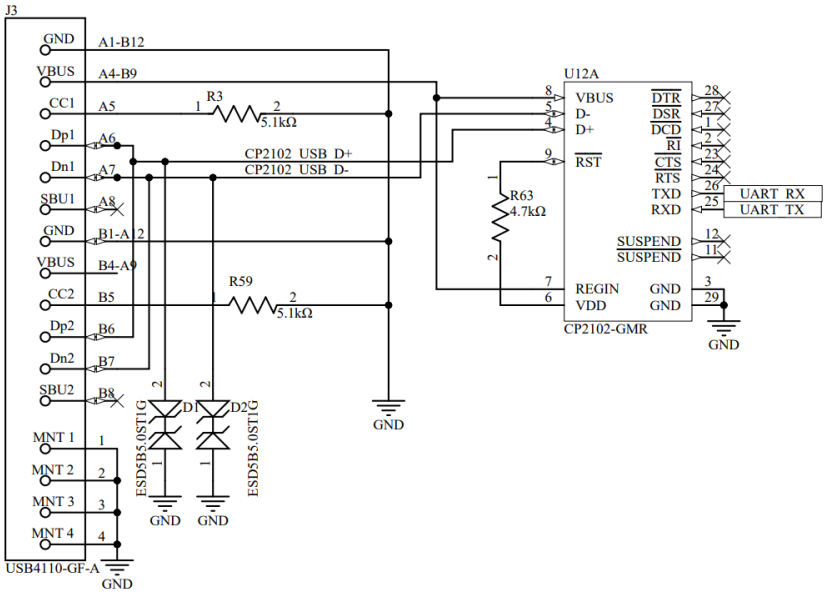


Figura 4.15: Esquemático del circuito USB–UART.

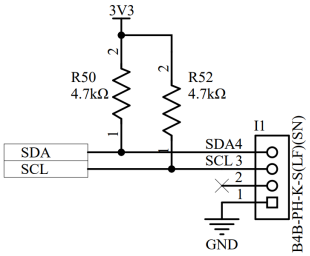


Figura 4.16: Esquemático del circuito I2C.

El diseño fue abordado prestando principal atención a que sea de fácil comprensión, visto que uno de los objetivos es que el proyecto sirva como herramienta didáctica en el futuro. Por esta razón, se priorizó la claridad por sobre la miniaturización, dejando suficiente espacio para incluir etiquetas para cada módulo y ordenando los componentes de la manera más clara posible. Por este mismo motivo, también se agregan *test points* en puntos del circuito que se consideran importantes para el entendimiento del flujo de las señales y del sistema. Ambas decisiones de diseño resultaron de gran utilidad durante el desarrollo del prototipo, ya que permitieron realizar modificaciones que en una placa más compacta o con un ruteo diferente no habrían sido posibles.

En la Tabla 4.4 se detallan función y ubicación de cada uno de los *test points*.

[1]Errata: El *test point* marcado ENA_MUX en la placa debería llamarse ENA_INA_TEMP.

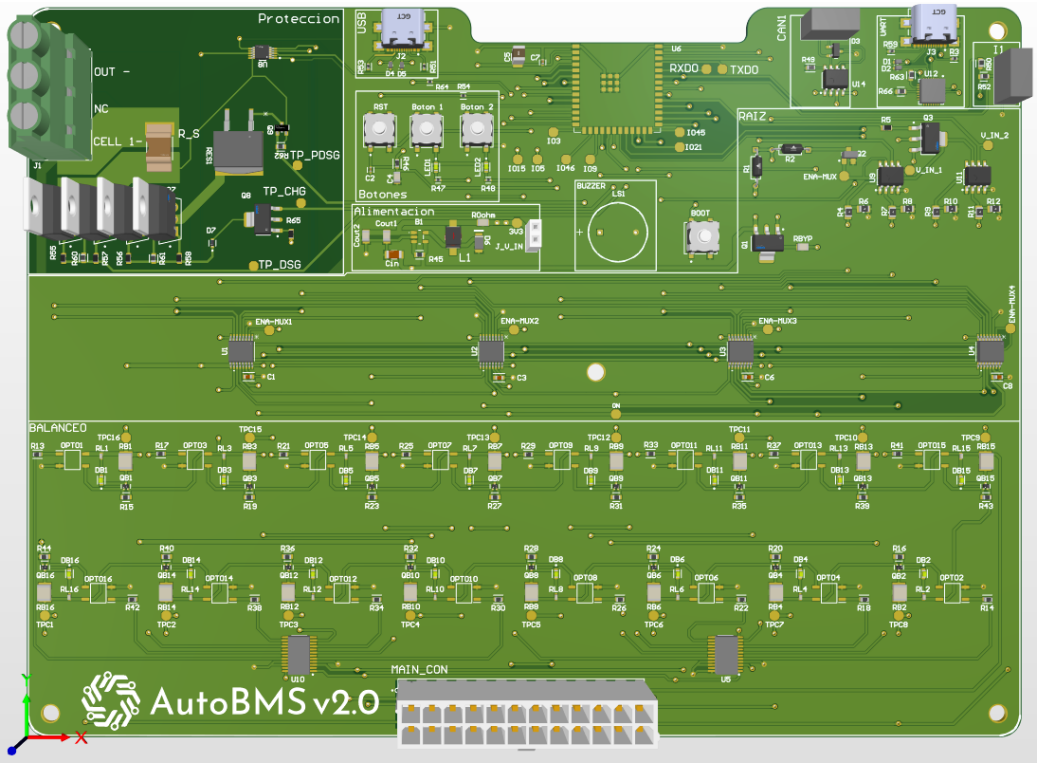


Figura 4.17: Modelo 3D de la placa diseñada.

### 4.7.1. Características Generales de la Placa

La PCB fabricada es de cuatro capas, en dieléctrico FR-4, con dimensiones de 154 mm × 210 mm y 1,6 mm de espesor. Para incrementar la tolerancia a corrientes elevadas, las capas exteriores emplean cobre de 2 oz/ft$^2$ (70 µm nominal). En la Subsección 4.7.3 se desarrolla el manejo de altas corrientes.

Por requerimiento de ciertos componentes (en particular el MCU [15]), y con el fin de facilitar el enrutamiento de las trazas, la placa incorpora un plano de tierra en la capa 2 y un plano de alimentación en la capa 3. Las capas exteriores se destinan al ruteo de señales.

También se destaca que el diseño no implementa una conexión en estrella para la distribución de alimentación, dado que las corrientes consumidas por los componentes son del orden de pocos mA, por lo que no se generan caídas de tensión significativas que justifiquen dicha topología. Este tipo de conexión resulta conveniente únicamente cuando se prevé que un consumo elevado pueda provocar caídas de tensión en los nodos ubicados *aguas abajo*.

### 4.7.2. Conectores

La placa cuenta con un total de seis conectores, cada uno destinado a una función específica dentro del sistema. Estos se resaltan en la Figura 4.17 y se describen brevemente a continuación:

| ***Test point*** | **Descripción** |
|---|---|
| 3V3 | Salida de 3,3 V del conversor DC-DC. |
| TPC# | Positivo de la celda #. |
| TP_CHG | Salida de control del MCU para el MOSFET de carga. |
| TP_DSG | Salida de control del MCU para el MOSFET de descarga. |
| TP_PDSG | Salida de control del MCU para el MOSFET de predescarga. |
| ENA_MUX# | Señal de *enable* del multiplexor número #. |
| ENA_MUX [1] | Señal de activación de la alimentación para amplificadores diferenciales, sensores de temperatura y multiplexores de medición. |
| V_IN_# | Señal a la salida de cada uno de los amplificadores diferenciales, previo a entrar al ADC. |

Tabla 4.4: *Test points* de la placa y su función.

- **Conectores USB Tipo C (`J2` y `J3`):** utilizados para las interfaces USB (Sección 4.2) y UART (Sección 4.6), respectivamente.
- **Conectores B4B-PH-K-S [18]:** se implementan dos de estos conectores, uno etiquetado como `CAN1` para la interfaz CAN, y otro como `I1`, utilizado para exponer el bus I2C para depuración o expansión.
- **Conector principal (`MAIN_CON`):** corresponde al modelo `39-29-6248` de Molex [22], de 24 pines, empleado como interfaz principal entre la placa del BMS y la batería. A través de este conector se transmiten las tensiones de las celdas, las señales de ambos sensores de temperatura y su señal de activación.
- **Conector de corriente (`J1`):** destinado a la conexión del polo negativo de la batería al BMS. Soporta corrientes de hasta 30 A y tensiones de hasta 300 V, siendo su uso apropiado para esta aplicación [39].

La tabla Tabla 4.5 detalla las asignaciones de pines correspondientes a cada uno de los conectores descritos, excluyendo los puertos USB, cuya descripción se incluye en sus respectivas secciones.

### 4.7.3. Manejo de Corrientes Elevadas

Se dimensionan las pistas de la ruta de potencia (conector J1 - MOSFETs de corte - resistencia *shunt*) para tolerar una corriente de 20 A en régimen continuo, con un aumento térmico máximo admisible de $\Delta T = 20\,^{\circ}\text{C}$. La metodología de diseño se basa en los lineamientos de la norma IPC–2221A [1] para una estimación

| Conector | Pin | Señal | Descripción |
|---|---|---|---|
| MAIN_CON | 1 | CELL16 | Positivo de la celda 16. |
| | 2 | CELL15 | Positivo de la celda 15. |
| | 3 | CELL14 | Positivo de la celda 14. |
| | 4 | CELL13 | Positivo de la celda 13. |
| | 5 | ENA_INA_TEMP | Señal de habilitación de los sensores de temperatura. |
| | 6 | TEMP2 | Salida analógica del sensor de temperatura 2. |
| | 7 | TEMP1 | Salida analógica del sensor de temperatura 1. |
| | 8 | NC | No conectado. |
| | 9 | CELL12 | Positivo de la celda 12. |
| | 10 | CELL11 | Positivo de la celda 11. |
| | 11 | CELL10 | Positivo de la celda 10. |
| | 12 | CELL9 | Positivo de la celda 9. |
| | 13 | CELL1 | Positivo de la celda 1. |
| | 14 | CELL2 | Positivo de la celda 2. |
| | 15 | CELL3 | Positivo de la celda 3. |
| | 16 | CELL4 | Positivo de la celda 4. |
| | 17 | NC | No conectado. |
| | 18 | NC | No conectado. |
| | 19 | NC | No conectado. |
| | 20 | CELL5 | Positivo de la celda 5. |
| | 21 | CELL6 | Positivo de la celda 6. |
| | 22 | CELL7 | Positivo de la celda 7. |
| | 23 | CELL8 | Positivo de la celda 8. |
| | 24 | CELL1- | Polo negativo de la celda 1 y tierra del sistema. |
| CAN1 | 1 | GND | Tierra del sistema. |
| | 2 | CANL | Línea negativa del bus CAN. |
| | 3 | CANH | Línea positiva del bus CAN. |
| | 4 | NC | No conectado. |
| J1 | 1 | OUT- | Salida negativa del BMS. |
| | 2 | NC | No conectado. |
| | 3 | CELL1- | Polo negativo de la batería. |
| I1 | 1 | GND | Tierra del sistema. |
| | 2 | NC | No conectado. |
| | 3 | SCL | Línea de reloj del bus I2C. |
| | 4 | SDA | Línea de datos del bus I2C. |

Tabla 4.5: Asignación de pines de los conectores del sistema.

conservadora del ancho mínimo de las pistas. El aumento térmico máximo admisible se estableció en $\Delta T = 20\,^{\circ}\mathrm{C}$, lo cual resulta adecuado para una aplicación automotriz donde las temperaturas de operación típicas alcanzan los $50\,^{\circ}\mathrm{C}$.

Para capas externas, la sección mínima $A$ (en $\mathrm{mil}^2$) se estima por:

$$I = k\,\Delta T^{0,44}\,A^{0,725} \quad \Rightarrow \quad A = \left(\frac{I}{k\,\Delta T^{0,44}}\right)^{1/0,725},$$

con $k = 0{,}048$ para pistas en capas externas. El ancho mínimo resulta de $w = A/t$, con $t$ en mil. Para $t = 2{,}74\,\mathrm{mil}$ (espesor de cobre correspondiente a $2\,\mathrm{oz/ft}^2$), $I = 20\,\mathrm{A}$ y $\Delta T = 20\,^{\circ}\mathrm{C}$:

$$A = 667\,\mathrm{mil}^2, \qquad w = \frac{667\,\mathrm{mil}^2}{2{,}74\,\mathrm{mil}} \approx 243\,\mathrm{mil} \approx 6{,}17\,\mathrm{mm}.$$

En comparación, para un espesor de cobre de $1\,\mathrm{oz/ft}^2$ y mismos requerimientos de corriente y temperatura, el ancho de pista mínimo hubiera sido de $12{,}3\,\mathrm{mm}$.

En el trayecto de corriente principal, la traza de menor sección presenta un ancho de $6{,}5\,\mathrm{mm}$, determinado por restricciones de espacio en el *layout*. El resto de las trazas de potencia poseen anchos superiores a $8{,}5\,\mathrm{mm}$, asegurando una capacidad de conducción acorde a los requerimientos establecidos.

### 4.7.4. Posibilidad de Expansión

Con el objetivo de permitir futuras ampliaciones o modificaciones del sistema, se incorporaron *test points* en los pines del MCU que no fueron utilizados en la presente versión del diseño. Esto facilita el acceso directo a las líneas digitales disponibles, tanto para tareas de depuración como para la implementación de nuevas funciones, sin necesidad de realizar cambios en el *layout* existente.

Particularmente, se dejaron los pines `RXD0` y `TXD0` con *test points* accesibles, ya que no se emplean en la comunicación UART principal del sistema. De esta forma, pueden destinarse a pruebas o a una interfaz de programación alternativa en futuras revisiones.

Si bien en esta versión no se implementa ningún tipo de comunicación inalámbrica, la placa contempla dicha posibilidad. Se incluyó un recorte (*cutout*) en la zona correspondiente a la antena del MCU, siguiendo las recomendaciones de diseño indicadas en [15]. Este recorte asegura el despeje de cobre necesario para un desempeño óptimo de la antena interna en caso de habilitar las interfaces Wi-Fi o Bluetooth en futuras versiones.

# Capítulo 5

# Diseño de Firmware y Software

El objetivo de este capítulo es describir en detalle el *firmware* desarrollado para el correcto funcionamiento del sistema embebido. El *firmware* se implementó utilizando el *framework* ESP-IDF en su versión 5.4, sobre el microcontrolador ESP32-S3. ESP-IDF es el *software development kit* (SDK) oficial de Espressif para microcontroladores de la familia ESP32, utiliza FreeRTOS como kernel.

Este entorno provee un sistema operativo en tiempo real basado en FreeRTOS, que permite organizar el código en tareas periódicas y coordinadas, facilitando la modularidad y el control de tiempos de ejecución.

El *firmware* se diseñó siguiendo los principios de modularidad y extensibilidad. La modularidad implica que cada bloque del sistema cumple una función específica y bien delimitada, mientras que la extensibilidad asegura que se puedan incorporar nuevas funciones o modificar el *hardware* sin necesidad de tener que hacer modificaciones profundas en el código.

Bajo estas premisas, el *firmware* se estructuró en módulos de medición, protección, balanceo, comunicación, gestión de parámetros y estimación de SOC y SOH.

Cada módulo puede ser desarrollado, probado y mantenido de manera independiente, y luego integrado a través de la lógica principal del programa.

## 5.1. Arquitectura del Firmware

El objetivo principal del sistema es ejecutar, de manera periódica y determinística, cinco mediciones completas por segundo de las magnitudes clave del BMS: tensiones de celda, corriente y temperatura.

Cada ciclo de aproximadamente 200 ms realiza la adquisición de datos, verifica las protecciones y ejecuta acciones de protección en caso de ser necesario. Cada 5 ciclos (equivalente a 1 s) transmite los datos por UART y cada 50 ciclos, o sea, cada 10 s, actualiza los parámetros de la batería (SOC y SOH).

De esta manera, el sistema mide cinco veces por segundo, verifica protecciones de forma inmediata tras cada medición, comunica periódicamente el estado de la batería y mantiene una actualización frecuente de los indicadores de capacidad y

salud de la batería.

A continuación se presenta un pseudocódigo simplificado que resume la secuencia de ejecución del *firmware*:

**Algorithm 1** Ciclo principal del BMS

```
Medir tensiones, corriente y temperatura
Verificar protecciones con las mediciones actuales
if existe evento de protección then
    Ejecutar protección correspondiente
end if
if ciclo mod 5 = 0 then
    Transmitir estado
end if
if ciclo mod 50 = 0 then
    Actualiza parámetros de la batería
end if
ciclo ← ciclo + 1
```

## 5.2. Módulos de Firmware

En esta sección se detallan los principales módulos de *firmware* (medición, protecciones, balanceo de celdas, comunicación con interfaces externas, gestión de parámetros configurables y estimación de SOC y SOH), describiendo en cada caso su funcionamiento, organización interna y los límites de su alcance.

### 5.2.1. Medición

El módulo de medición se encarga exclusivamente de adquirir los parámetros eléctricos y térmicos de la baterías, su única responsabilidad es obtener valores confiables de tensión individual de cada celda, la corriente del sistema, la carga acumulada y la temperatura en puntos críticos de la batería. Estos datos se registran en la estructura de *estado*, donde se guardan todos los parámetros asociados a la batería (incluyendo SOC y SOH), y son utilizados por otros módulos del *firmware*.

El proceso comienza con la función `init_medicion`, que se ejecuta una única vez al inicio del sistema, en ella se calibran los ADC del ESP32-S3, se configuran los pines de dirección y habilitación de los multiplexores, y se inicializa el integrado INA228, encargado de medir corriente y acumulación de carga. Posteriormente, en cada ciclo de operación, se llaman funciones específicas que procesan las mediciones:

- `medir_corriente`: obtiene la corriente que circula por la batería;
- `medir_voltajes`: procesa las medidas de tensión individual de cada celda;

- `medir_carga`: recibe la carga acumulada a lo largo del tiempo, calculada por el integrado INA228, y la guarda en la estructura *estado*; y
- `medir_temperatura`: procesa las medidas de temperatura en las dos ubicaciones térmicas definidas.

De esta manera, el módulo de medición provee una actualización del estado físico de la batería, sin almacenar ni decidir acciones sobre estos datos, tareas que son responsabilidad de otros módulos del sistema.

### Medir temperatura

La función `medir_temperatura` utiliza el ADC configurado en modo *oneshot* con canales dedicados a cada sensor. En este modo el ADC cumple con las siguientes carácteristicas:

- Solo mide cuando se lo solicita, es decir, no mide continuamente, sino que realiza una sola conversión cada vez que se llama a la función;
- No almacena automáticamente múltiples muestras ni usa interrupciones, cada llamada es una conversión puntual y directa;
- El ADC solo se activa para la medición y luego se apaga, lo que ahorra energía

Para cada punto de medición se toma un conjunto de muestras y se convierten los valores crudos de ADC a m°C usando la calibración establecida en la inicialización. De esta forma se obtienen dos lecturas de temperatura que se guardan en el vector *estado.temps[]*, perteneciente a la estructura de *estado*.

Las temperaturas actualizadas se verifican en cada ciclo por el módulo de protecciones, que decide si corresponde aplicar corte de carga, descarga o ambas en función de umbrales definidos.

**Algorithm 2** Medición de temperatura

```
1: procedure MEDIR_TEMPERATURA(estado)
2:     Leer valor ADC del canal asociado al sensor de temperatura 1
3:     Convertir valor crudo a temperatura en m°C
4:     Guardar en estado.temps[0]
5:     Leer valor ADC del canal asociado al sensor de temperatura 2
6:     Convertir valor crudo a temperatura en m°C
7:     Guardar en estado.temps[1]
8: end procedure
```

### Medir voltajes

En cada ciclo se adquieren las tensiones de las 16 celdas. Para cada lectura se toman nueve muestras por canal del ADC configurado en modo *oneshot* y se calcula la mediana de esas muestras. Se optó por utilizar la mediana de las muestras, en lugar del promedio, porque es más robusta frente a valores atípicos (*outliers*) y picos de ruido, ignora lecturas extremas que sesgarían el promedio y, en la práctica, entrega una estimación más estable en presencia de interferencias propias del entorno de potencia.

Sobre esa estimación además se aplica un filtro *exponential moving average* (EMA) que suaviza el ruido de alta frecuencia otorgando más peso a las mediciones recientes que a las antiguas; así se mantiene una actualización sensible a cambios reales sin reaccionar a fluctuaciones esporádicas.

La función `medir_voltajes` habilita la alimentación de los multiplexores y su salida (mediante los pines `GPIO38` y `GPIO40` del ESP32-S3 en niveles 0 y 1, respectivamente). Luego recorre las ocho direcciones de los multiplexores; en cada dirección se seleccionan simultáneamente dos celdas (una por cada canal de entrada, `V_IN1` y `V_IN2`), de modo que se miden dos celdas a la vez, p. ej., celda 1 y celda 7 para una misma dirección. Para cada canal, la rutina interna `leer_adc_mediana_mv` toma nueve muestras crudas del ADC, ordena las lecturas, devuelve la mediana convertida a mV y aplica la corrección de calibración correspondiente.

De vuelta en `medir_voltajes`, cada medición se asigna a su índice de celda mediante el mapeo de canales y se filtra con un EMA. El valor filtrado se copia en la estructura *estado* para que lo utilicen otros módulos (protecciones, balanceo, etc.). Al finalizar el barrido completo (16 celdas), se deshabilita la salida de los multiplexores y se corta su alimentación para reducir consumo. Esto queda resumido en el pseudocódigo descrito a continuación.

**Algorithm 3** Adquisición de tensiones de celdas

```
1: procedure MEDIR_VOLTAJES(estado)
2:     Habilitar alimentación de MUX
3:     Habilitar salida de MUX
4:     for direccion = 0 to 7 do
5:         Configurar MUX a direccion
6:         Seleccionar dos celdas asociadas a V_IN1 y V_IN2
7:         Tomar 9 muestras de ADC por canal
8:         Ordenar cada conjunto y tomar la mediana (en mV)
9:         Aplicar filtro EMA por celda
10:        Actualizar estructura estado
11:    end for
12:    Deshabilitar salida de MUX
13:    Cortar alimentación de MUX
14: end procedure
```

### Medir corriente

La medición de corriente que circula por la batería se realiza mediante el integrado INA228, conectado al bus I2C.

El integrado es un sensor especializado que entrega la corriente en unidades calibradas a través de sus registros internos. De esta forma, se evita depender exclusivamente del ADC interno del microcontrolador, y se obtiene una medición más precisa y estable de la corriente.

La función `medir_corriente` se comunica con el INA228 a través del *handle* de I2C inicializado en `init_medicion`. El procedimiento consiste en leer el registro correspondiente a la corriente, convertir el valor crudo a mA usando el factor de escala (`CURRENT_LSB`) definido en el módulo, y almacenar el resultado en la estructura *estado*.

Este valor se actualiza en cada ciclo de 200 ms junto con las tensiones y temperaturas, y se utiliza en tiempo real tanto para las protecciones por sobrecorriente como para el cálculo acumulado de carga (contador de Coulombs), que luego se emplea en la estimación del SOC. El funcionamiento de esta función queda resumido en el pseudcódigo descrito a continuación.

**Algorithm 4** Medición de corriente

```
1: procedure MEDIR_CORRIENTE(estado)
2:     Enviar dirección del registro de corriente al INA228
3:     Recibir datos crudos de corriente por I2C
4:     Convertir valor crudo a miliamperios usando CURRENT_LSB
5:     Guardar resultado en estado.corriente
6: end procedure
```

### Medir carga

Además de entregar la corriente instantánea, el integrado INA228 dispone de un acumulador interno que realiza el conteo de Coulombs (CC) a lo largo del tiempo. De esta forma, el propio sensor integra la corriente medida y genera un valor acumulado proporcional a la carga que ha circulado por la batería, lo que resulta fundamental para la estimación posterior del SOC.

La función `medir_carga`, al igual que `medir_corriente`, lee el registro *charge* del INA228 a través de I2C. Este registro tiene una resolución de 24 bits con signo, por lo que el valor recibido se compone de tres bytes, el código combina los bytes en un entero, realiza la extensión de signo para representar correctamente valores negativos (caso de descarga) y guarda el resultado en *estado.carga*.

A diferencia de las tensiones y temperaturas, la carga acumulada no se reinicia en cada ciclo, sino que refleja la integración continua de corriente desde que el sensor fue inicializado o puesto a cero. Este valor es utilizado directamente por el módulo de estimación de SOC para ajustar la estimación del estado de carga de la batería. Esta función se resume en el pseudocódigo descrito a continuación.

**Algorithm 5** Medición de carga acumulada

```
1: procedure MEDIR_CARGA(estado)
2:     Enviar dirección del registro de carga al INA228
3:     Recibir tres bytes de datos (24 bits con signo)
4:     Reconstruir valor crudo de 24 bits
5:     Extender signo a 32 bits
6:     Guardar resultado en estado.carga
7: end procedure
```

## 5.2.2. Protección

Este módulo es el encargado de ejecutar las acciones de seguridad necesarias según los parámetros adquiridos por el módulo de medición. Su funcionamiento se basa en comparar los valores de tensión, corriente y temperatura con umbrales configurables definidos en el sistema. Cuando alguno de estos parámetros supera un límite máximo o queda por debajo de un límite mínimo, el módulo activa la protección correspondiente, cortando carga, descarga o ambas según el caso.

Las protecciones que el módulo puede activar son las siguientes:

- **Sobredescarga:** Se activa cuando alguna celda cae por debajo de `umbral_sobredescarga` (por defecto 3000 mV); parámetro configurable en el módulo de parámetros. Esta protección evita daños en la celda durante la descarga, para esto deshabilita la descarga de la batería, aunque aún es posible realizar la carga. Esto asegura que la batería se pueda recargar sin ser exigida en un estado crítico.
- **Sobrecarga:** Esta condición se evalúa únicamente cuando la batería está en proceso de carga, si alguna celda supera el `umbral_sobrecarga` (por de-

fecto 4200 mV), se corta la carga pero se mantiene habilitada la descarga, actuando de manera inversa a la protección por sobredescarga.

- **Sobretemperatura:** Si la temperatura supera el umbral `sobre_temperatura` (por defecto 60 °C), se bloquean simultáneamente la carga y descarga de la batería. Esta protección previene riesgos de degradación térmica o incendio.

- **Subtemperatura:** Si la temperatura es menor que el umbral `sub_temperatura` (por defecto −20 °C), se deshabilitan tanto la descarga como la carga. Esto refleja que en condiciones de frío extremas las celdas no pueden entregar corriente de forma segura, ni se pueden cargar.

- **Temperatura de descarga superior:** Cuando la temperatura supera el umbral `temp_solo_descarga_sup` (por defecto 45 °C), la descarga se mantiene habilitada pero la carga se bloquea. Esta condición refleja que, a temperaturas elevadas, las celdas pueden entregar energía con seguridad, pero no resulta seguro continuar el proceso de carga.

- **Temperatura de descarga inferior**: Cuando la temperatura esta por debajo del umbral `temp_solo_descarga_inf` (por defecto 0 °C), se deshabilita la carga, pero se mantiene habilitada la descarga. Análogo al caso anterior, esta condición refleja que por debajo de esta temperatura las celdas pueden entregar corriente de forma segura, pero no pueden continuar con el proceso de carga [5].

- **Sobrecorriente:** Si la corriente instantánea supera el umbral `max_corriente` (por defecto 15 000 mA), el sistema corta tanto la carga como la descarga de la batería. Se trata de una protección rápida, ya que está asociada a la seguridad eléctrica y a evitar esfuerzos excesivos sobre las celdas y el *hardware*.

Además de las protecciones mencionadas, se incorporó un circuito de predescarga. La predescarga finaliza cuando la corriente cae por debajo del `UMBRAL_PREDESCARGA` (por defecto 500 mA) o cuando se alcanza el `TIMEOUT_PDSG` (por defecto 2250 ms); estos umbrales no son configurables dado que están relacionados con parámetros específicos de la implementación.

Es importante señalar que la recarga de la batería no se ve afectada por esta condición, solo la descarga está limitada durante la conexión inicial de una carga.

### Funciones de protección

De forma análoga al módulo de medición, que separa la adquisición de tensiones, corriente y temperaturas en funciones específicas, el módulo de protección divide su lógica en cuatro funciones principales:

- `pre_descarga`: controla el circuito de predescarga.

- `proteccion_carga_descarga`: evalúa las protecciones de tensión por sobredescarga y sobrecarga.

- `proteccion_temperatura`: verifica las condiciones de sobretemperatura, subtemperatura, y temperatura de descarga superior e inferior.

- `proteccion_corriente`: supervisa la protección por sobrecorriente.

Estas funciones no son independientes del resto del *firmware*, cada una utiliza los datos adquiridos por el módulo de medición, por ejemplo, la función `proteccion_temperatura` activa la protección necesaria en función de las temperaturas medidas por `medir_temperatura()`, la función `proteccion_carga_descarga` usa las tensiones obtenidas por `medir_voltajes()` para determinar si se está en el caso de sobrecarga o sobredescarga, y la función `proteccion_corriente` compara la corriente obtenida por `medir_corriente()` con el umbral de protección para sobrecorriente.

Por otro lado, la función `pre_descarga(state_t* estado)` implementa la conexión suave de la batería a la carga, es decir, cuando se conecta la batería a un sistema externo, se habilita inicialmente el camino resistivo de predescarga y se mantiene bloqueada la descarga principal, el sistema mide periódicamente la corriente y espera hasta que la corriente medida esté por debajo del `UMBRAL_PREDESCARGA` o hasta alcanzar el `TIMEOUT_PDSG`. Si la corriente cae dentro del tiempo permitido, se desactiva la predescarga y se habilita la descarga normal; de lo contrario, se desactiva igualmente la predescarga y se registra el *timeout*.

A excepción de la función de `pre_descarga(state_t* estado)`, el resto de las funciones retornan una estructura de tipo `protection_event_t`, que sirve como indicador de qué protección ha sido activada. De esta forma, el módulo permite detectar en qué categoría ocurrió el evento y aplicar la acción correspondiente sobre los MOSFETs de potencia, el buzzer o los indicadores LED.

Si bien estas tres funciones supervisan distintos parámetros, las tres siguen el mismo esquema:

1. reciben el estado de la batería,

2. comparan el parámetro medido con los umbrales configurados, y

3. activan la protección correspondiente.

Esto se ve de forma más clara en el siguiente pseudocódigo.

**Algorithm 6** Estructura general de las funciones de protección

```
1: procedure PROTECCION_X(estado, umbrales)
2:     Leer parámetro de la batería (tensión, corriente, temperatura)
3:     if parámetro fuera de rango then
4:         Actualizar estructura de eventos
5:     end if
6:     Se llama a función activar_proteccion(protection_event_t
   event, bool corte_general)
7: end procedure
```

La función `activar_proteccion()` es la rutina central del módulo de protección, que recibe la estructura completa de eventos `event`, que contiene el estado de todas las protecciones activas, y un indicador lógico `corte_general` (modo de funcionamiento descrito en Subsección 5.2.7). A partir de esta información, determina las acciones físicas a aplicar sobre el sistema: apertura o cierre de los MOSFETs de carga y descarga, encendido o apagado de los indicadores LED y activación del buzzer.

Primero evalúa qué tipo de protección se encuentra activa dentro de la estructura `event`, si se detecta una protección severa (aquella que requiere corte de carga y descarga), como sobrecorriente, sobretemperatura o subtemperatura, la función activa el buzzer de forma continua por un intervalo de 2 s. Este tiempo se controla con el contador interno `inicio_buzzer`, que guarda el tiempo de encendido del buzzer en microsegundos y apaga el buzzer una vez superado el período indicado.

Por otra parte, si la protección activa corresponde a una condición de tensión fuera de rango (sobrecarga o sobredescarga), el buzzer se activa de forma intermitente. En estas condiciones, la función genera pulsos de aproximadamente 500 ms encendido y 500 ms apagado, por un total de cuatro pulsos (aproximadamente 2 s de duración total). Para ello se utilizan dos contadores de tiempo internos:

- `beep`: mide la duración de cada pulso de encendido o apagado del buzzer; cuando transcurre el tiempo asignado, se apaga el buzzer y se reinicia para el siguiente ciclo;
- `iter_buzzer`: cuenta la cantidad total de pulsos emitidos, deteniendo la secuencia una vez alcanzados cinco pulsos.

Cuando ambos contadores se reinician (por finalización de la secuencia o ausencia de eventos de tensión), el buzzer permanece apagado.

La función también enciende los indicadores LED según el tipo de evento detectado:

- **LED1**: se enciende cuando la protección activa corresponde a una sobrecorriente;
- **LED2**: se enciende cuando se activa alguna de las cuatro protecciones térmicas;

- **LED1 y LED2**: parpadean simultáneamente durante el modo de buzzer intermitente, indicando una condición de sobrecarga o sobredescarga.

Finalmente, `activar_proteccion()` aplica las decisiones de corte sobre los MOSFETs de potencia, si `corte_general` es verdadero, o si alguna protección severa está activa, se deshabilitan simultáneamente las ramas de carga y descarga.

De lo contrario, se evalúa cada camino por separado:

- La carga se deshabilita si existe una condición térmica fuera del rango seguro, como en los casos de temperatura de descarga inferior o superior o en caso de sobrecarga;
- La descarga se deshabilita si se detecta sobredescarga;
- Si no se encuentra ninguna protección activa y no se activa el modo de `corte_general`, ambos caminos permanecen habilitados.

Se aclara que en el caso particular de sobrecarga, la protección se desactiva periódicamente cada 10 s para comprobar si la batería dejó de cargarse; de no ser así, se reactiva y repite la alarma sonora y visual, indicando que la carga está completa.

El funcionamiento de esta función se resume en el siguiente pseudocódigo:

**Algorithm 7** Activación de protecciones

```
1: procedure ACTIVAR_PROTECCION(event, corte_general)
2:     if Protección severa activa then
3:         if buzzer apagado then
4:             inicio_buzzer se carga con tiempo de inicio
5:             Se enciende el buzzer
6:         else if tiempo actual − inicio_buzzer > 2 s then
7:             Se apaga el buzzer
8:         end if
9:     end if
10:    LED1 ← (Protección de corriente activa ? 1 : 0)
11:    LED2 ← (Protección térmica activa ? 1 : 0)
12:    if Tensión fuera de rango then
13:        if iter_buzzer < 5 then
14:            if beep = 0 then
15:                Se encienden el buzzer y los LEDs
16:                beep se carga con el tiempo de inicio
17:            else if tiempo_actual − beep > 500 ms then
18:                Se apagan el buzzer y los LEDs
19:                beep se carga en 0
20:                Se incrementa en 1 el contador iter_buzzer
21:            end if
22:        end if
23:    end if
24:    if Protección severa o modo corte_general activo then
25:        Se cortan la carga y descarga
26:    else
27:        if Protección que corta solo carga activa then
28:            Se corta solo la carga
29:        else
30:            Se habilita la carga
31:        end if
32:        if Protección que corta solo descarga activa then
33:            Se cortar solo la descarga
34:        else
35:            Se habilita la descarga
36:        end if
37:    end if
38: end procedure
```

### 5.2.3. Balanceo

Este módulo tiene como responsabilidad ecualizar las tensiones de las celdas de la batería durante la carga.

Para el BMS diseñado, el balanceo se activa únicamente cuando la tensión de una o más celdas supera el umbral definido por el parámetro `voltaje_balanceo`, que por defecto tiene un valor de 3900 mV. Este umbral es configurable desde el módulo de parámetros, permitiendo adaptar el comportamiento del balanceo al tipo de celda o estrategia de carga. Activar el balanceo únicamente por encima de este umbral tiene la ventaja de reducir los tiempos de carga total de la batería, ya que el proceso de ecualización se concentra en la fase final de carga sin afectar las etapas iniciales de carga rápida.

#### Funcionamiento del balanceo

En cada ciclo, el módulo determina la tensión mínima y la tensión máxima entre las 16 celdas, si la tensión máxima supera el umbral `voltaje_balanceo`, se compara la tensión de cada celda con la tensión mínima. Cuando la diferencia entre estas supera el umbral *umbral_balanceo_encendido*, se activa la resistencia de descarga correspondiente a esa celda; cuando la diferencia desciende por debajo del umbral `umbral_balanceo_apagado`, el balanceo de esa celda se desactiva.

Ambos umbrales, `umbral_balanceo_encendido` y `umbral_balanceo_apagado`, son configurables desde el módulo de parámetros, permitiendo ajustar el comportamiento del balanceo a diferentes químicas o configuraciones de celdas. Sus valores por defecto son 40 mV y 20 mV, respectivamente, tomados como referencia de las recomendaciones de Texas Instruments para balanceo pasivo en sistemas multicelda [16]. Esta histéresis entre ambos umbrales evita conmutaciones rápidas de los MOSFET de balanceo cuando las tensiones de las celdas se aproximan al equilibrio, asegurando una operación estable y prolongando la vida útil de los componentes. Esto se detalla en el siguiente pseudocódigo.

**Algorithm 8** Balanceo pasivo de celdas

```
1: procedure BALANCEO_PASIVO(estado)
2:     if no esta cargando then
3:         return
4:     end if
5:     V_mín = mín(estado.voltajes[0..15])
6:     V_máx = máx(estado.voltajes[0..15])
7:     if V_máx < voltaje_balanceo then
8:         return
9:     end if
10:    for cada celda i = 0.,15 do
11:        ΔV = estado.voltajes[i] − V_mín
12:        if ΔV > umbral_balanceo_encendido then
13:            activar_balanceo(i)
14:        else if ΔV < umbral_balanceo_apagado then
15:            desactivar_balanceo(i)
16:        end if
17:    end for
18: end procedure
```

### 5.2.4. Comunicación

El módulo de comunicación tiene como función principal establecer y gestionar los enlaces I2C, UART y TWAI/CAN del sistema. A través de estos protocolos de comunicación el BMS intercambia comandos de configuración, transmite el estado del sistema y se comunica con una interfaz gráfica (GUI) o con dispositivos externos.

#### Protocolo de comunicación I2C

El protocolo I2C se utiliza en el BMS para la comunicación entre el microcontrolador y periféricos internos del sistema, en particular el sensor de corriente y carga INA228 y los bloques de *switches* empleados para el balanceo de celdas. El bus opera en modo *maestro–esclavo*, donde el microcontrolador actúa como maestro, las direcciones de los dispositivos esclavos son de 7 bits, siguiendo los requisitos del INA228 y los bloques de *switches* para la comunicación I2C [4] [17]. El bus se configura con una frecuencia de 100 kHz y con un tiempo máximo de espera de 1000 ms y su inicialización se realiza mediante la función `i2c_master_init()` del módulo de comunicación, que configura el modo maestro, los pines físicos y la velocidad de reloj.

Las operaciones de lectura y escritura son gestionadas por las funciones `i2c_master_read_slave()` y `i2c_master_write_slave()` respectivamente, las cuales son llamadas por los módulos de `medicion.c` y `balanceo.c`, que utilizan el bus para comunicarse con los periféricos, según se muestra a continuación:

- Módulo `medicion.c`: lo utiliza para la lectura de registros de corriente y carga acumulada del INA228.
- Módulo `balanceo.c`: envía por el bus un byte al bloque de *switches* indicando con los bits altos que *switches* cerrar para el balanceo.

### Comunicación UART

El protocolo UART se emplea para la comunicación entre el BMS y la GUI. A través de este enlace, el BMS transmite periódicamente el estado del sistema y recibe comandos de configuración enviados desde la GUI, como modificaciones en los umbrales de protección o parámetros nominales de la batería.

El módulo `comunicacion.c` se encarga de inicializar el puerto serie, definir su velocidad, formato y pines asociados, la configuración utilizada es la siguiente:

- `Baud rate:` 115 200 bps.
- `Formato:` 8 bits de datos, sin paridad y 1 bit de stop (8N1).
- `Buffers:` tamaño de 2048 bytes tanto para recepción como para transmisión.

La función `init_uart()` realiza la inicialización del puerto utilizando las funciones nativas de ESP-IDF (`uart_param_config()`, `uart_set_pin()` y `uart_driver_install()`). El puerto se configura en modo asíncrono y sin control de flujo por *hardware*, ya que la comunicación se realiza a corta distancia y a baja velocidad relativa.

### Protocolo de comunicación TWAI/CAN

Espressif denomina TWAI al controlador CAN integrado en la familia ESP32. TWAI es compatible con CAN clásico y soporta identificadores de 11 y 29 bits, pero requiere un transceptor externo para conectarse al bus. Por razones de compatibilidad, en el código las funciones se siguen usando con el prefijo/nombre "CAN", aunque el driver oficial de ESP-IDF se documenta como TWAI [34].

El módulo `comunicacion.c` implementa las funciones necesarias para inicializar y operar el CAN, demostrando que la comunicación por CAN es posible en el BMS.

En el proyecto se configura en modo estándar, con 11 bits y a una velocidad de 500 kbit/s. La inicialización se realiza mediante la función `init_can()`, que configura las colas de transmisión y recepción, con 20 elementos para cada una, habilita alertas de evento y aplica la configuración de temporización definida por el macro `TWAI_TIMING_CONFIG_500KBITS()`. El filtro se establece en modo abierto mediante `TWAI_FILTER_CONFIG_ACCEPT_ALL()`, de forma que se aceptan todos los mensajes del bus sin distinción por ID.

Las funciones principales son:

- `init_can()`: inicializa el controlador en modo normal y arranca el bus.

- `can_send(id, data, len)`: transmite una trama con identificador de 11 bits y hasta 8 bytes de datos.

- `can_receive(msg, timeout)`: recibe una trama CAN y almacena su contenido en una estructura `twai_message_t`.

Durante las pruebas se validó la transmisión y recepción con otro microntrolador ESP32-S3. Sin embargo, el bus CAN no fue integrado funcionalmente en el *firmware* final del sistema, ya que la comunicación principal con la GUI se realiza por UART. No obstante, la implementación de la comunicación CAN permanece disponible y funcional, permitiendo futuras expansiones del BMS hacia comunicación en red con otros nodos o módulos.

### 5.2.5. Parámetros del sistema

El módulo de parámetros administra todas las *variables configurables* del BMS, esto incluye los umbrales de protección, valores nominales de la batería y ajustes de balanceo. Los otros módulos hacen uso de estos valores durante su operación.

Este módulo también es el encargado de:

- Persistir los parámetros en la memoria flash interna del ESP32-S3. Para esto se utilizó la biblioteca NVS [30]. Esta es la biblioteca de ESP-IDF para almacenar pares clave–valor en la memoria flash del microcontrolador. Esta biblioteca funciona como un pequeño almacén de configuración persistente donde los datos sobreviven a reinicios y cortes de energía. La información se organiza por espacios de nombres, y cada clave guarda un valor de tipos básicos (enteros con y sin signo, cadenas, blobs, etc.).

- Recibir comandos desde la GUI por UART para cambiar parámetros y guardar los cambios en NVS.

- Transmitir el estado completo del sistema por UART, incluyendo la lista de protecciones activadas.

#### Parámetros configurables

Los parámetros configurables se declaran en `parametros.h` como *extern* y se definen en `parametros.c`. Estos se describen a continuación en la Tabla 5.1

| **Parámetro** | **Descripción** | **Valor por defecto** |
|---|---|---|
| `voltaje_nominal_celda` | Valor nominal de tensión por celda | 4100 mV |
| `capacidad_nominal_celda` | Capacidad nominal de una celda | 3000 mA h |
| `celdas_en_serie` | Número de celdas conectadas en serie | 16 |
| `celdas_en_paralelo` | Número de celdas conectadas en paralelo | 1 |
| `sobre_temperatura` | Umbral superior de temperatura; bloquea carga y descarga | 60 000 m°C |
| `temp_solo_descarga_sup` | Umbral superior de activación para corte de carga por temperatura; si se supera el umbral, permite descarga pero bloquea carga | 45 000 m°C |
| `temp_solo_descarga_sup` | Umbral inferior de activación para corte de carga por temperatura; si se esta por debajo del umbral, permite descarga pero bloquea carga | 0 m°C |
| `sub_temperatura` | Umbral inferior de temperatura; bloquea descarga y permite carga | −20 000 m°C |
| `max_corriente` | Límite máximo de corriente antes de activar protección contra sobrecorriente; bloquea la carga y descarga | 15 000 mA |
| `umbral_sobredescarga` | Umbral mínimo de tensión por celda; bloquea la descarga pero permite la carga | 3000 mV |
| `umbral_sobrecarga` | Umbral máximo de tensión por celda; bloquea la carga y permite la descarga | 4200 mV |
| `voltaje_balanceo` | Tensión mínima a partir de la cual se permite balancear | 3400 mV |
| `umbral_balanceo_encendido` | Diferencia de tensión que activa balanceo | 40 mV |
| `umbral_balanceo_apagado` | Diferencia de tensión que desactiva balanceo | 20 mV |

Tabla 5.1: Parámetros configurables del BMS y sus valores por defecto.

### Persistencia con biblioteca NVS

Para inicializar el módulo se llama a la función `init_parametros()`, la cual inicializa el NVS, abre el *namespace* ‘‘`bms_params`’’ y lee cada clave.

Si la clave no existe, asigna el valor por defecto definido por las macros (p. ej. `UMBRAL_SOBREDESCARGA` de 3000 mA) y deja el parámetro listo en la RAM para el resto del sistema. Cuando la GUI envía nuevos valores, la función `recepcion_param()` los escribe inmediatamente en NVS para que persistan entre reinicios.

La función de inicialización se puede visualizar mejor en el siguiente pseudocódigo:

**Algorithm 9** init_parametros(): NVS + defaults

```
1:  procedure INIT_PARAMETROS
2:      Inicializa la memoria flash
3:      Abre la memoria y habilita escritura y lectura del NVS
4:      for all clave de parámetro do
5:          if clave existe y tiene un valor asociado then
6:              asignar a parámetro el valor de memoria
7:          else
8:              asignar a parámetro el valor por defecto
9:          end if
10:     end for
11: end procedure
```

### Recepción de parámetros por UART

La función `recepcion_param` recibe los parámetros configurables en el mismo orden que se muestra en la Tabla 5.1, en una cadena donde cada parámetro se encuentra separado por una coma. Cada valor recibido se guarda con su clave correspondiente en NVS (p. ej. ‘‘`alta_temp`’’ para `sobre_temperatura`).

En el siguiente pseudocódigo se da un resumen del código:

**Algorithm 10** recepcion_param(): cadena → RAM + NVS

```
1:  procedure RECEPCION_PARAM(cadena)
2:      tokens ← separar por comas(cadena)
3:      for i = 0 to 12 do                          ▷ 14 parámetros
4:          valor ← atoi(tokens[i])
5:          asignar(parametro_por_indice(i), valor)
6:          guardar_en_nvs(clave_por_indice(i), valor)
7:      end for
8:  end procedure
```

### Transmisión de estado por UART

La función `transmision_estado` envía un *frame* por UART con los datos del estado de la batería y los parámetros en los que esta configurado el BMS. El *frame* se arma de la siguiente forma:

- Carácter de inicio: `X`.
- Bloque 1 (config): 14 parámetros configurables en el orden listado en la Tabla 5.1.
- Bloque 2 (*estado*): *bitmask* de balanceo (16 bits como `0/1`), tensiones, temperaturas, corriente, carga, SOC/SOH y señales de protección (`overvoltage`, `undervoltage`, etc.).
- Flag: `modo_config` (1/0).
- Carácter de fin: `Z` seguido de salto de línea.

**Algorithm 11** transmision_estado(): frame X . . . Z

```
1: procedure TRANSMISION_ESTADO(estado, prot_evt, modo_config)
2:     buffer ← X
3:     anexar(14 parámetros configurables, en orden)
4:     balanceo_str ← bitmask(estado.balanceo, 16)
5:     anexar(balanceo_str, mediciones, SOC/SOH, flags de protecciones,
   modo_config)
6:     anexar Z\n
7:     uart_write_bytes(buffer)
8: end procedure
```

## 5.2.6. Cálculo de SOH y SOC

En esta sección se describe la implementación de los algoritmos de estimación. El SOC se calcula combinando conteo de carga con correcciones a partir de tablas OCV–SOC, mientras que el SOH se obtiene mediante un esquema adaptativo de RLS que ajusta la capacidad efectiva en tiempo real. Los fundamentos teóricos de ambos métodos se presentan en el Capítulo 3.

### Funciones del módulo

Este módulo cuenta con las siguientes funciones:

`interpolate_ocv_soc`: Esta función se encarga de calcular el valor de SOC correspondiente a un OCV específico de una tabla. Para esto se utiliza el conjunto de pares calibrados `(ocv_mV, soc)` correspondientes de una tabla OCV–SOC almacenada en el *firmware*.

Cada tabla contiene 11 pares ordenados de forma ascendente por voltaje y representa la relación experimental entre el OCV y el SOC para una temperatura determinada. El procedimiento consiste en recorrer los intervalos definidos por los pares consecutivos de la tabla hasta encontrar el rango que contiene el valor de OCV medido, una vez identificado el intervalo, se realiza una interpolación lineal entre los dos puntos para estimar el SOC asociado a ese voltaje:

$$SOC = SOC_1 + (SOC_2 - SOC_1)\,\frac{OCV - OCV_1}{OCV_2 - OCV_1},$$

donde $(OCV_1, SOC_1)$ y $(OCV_2, SOC_2)$ son los límites inferior y superior, respectivamente, del intervalo encontrado.

Si el voltaje ingresado se encuentra por debajo del valor mínimo de la tabla, la función devuelve $SOC = 0\,\%$, si se encuentra por encima del máximo, devuelve $SOC = 100\,\%$.

En resumen, esta función implementa una interpolación lineal sobre los datos calibrados de la curva OCV–SOC, permitiendo estimar el estado de carga de una celda a partir de su voltaje en reposo.

`get_soc_from_ocv`: Esta función estima el valor de SOC a partir del OCV y de la temperatura medida, utilizando las tablas calibradas OCV–SOC almacenadas en el sistema.

El procedimiento consiste en identificar las dos tablas cuya temperatura de referencia se encuentra más próxima a la temperatura de operación actual, una inmediatamente inferior (`t_low`) y otra inmediatamente superior (`t_high`). Una vez localizadas las tablas, se interpola dentro de cada tabla mediante la función `interpolate_ocv_soc()` para obtener los valores `soc_low` y `soc_high` correspondientes al voltaje de entrada.

Si ambas tablas tienen la misma temperatura de referencia (es decir, la temperatura medida coincide exactamente con una tabla existente), la función devuelve directamente `soc_low`. En caso contrario, se realiza una interpolación lineal entre los dos valores obtenidos para compensar la diferencia de temperatura:

$$SOC = SOC_{\text{low}} + (SOC_{\text{high}} - SOC_{\text{low}}) \cdot \frac{T - T_{\text{low}}}{T_{\text{high}} - T_{\text{low}}}$$

Finalmente, el resultado se limita al rango válido interno del sistema mediante la macro `CLAMP()`, de modo que el valor devuelto siempre se encuentre entre 0 (batería completamente descargada) y 100000 (batería completamente cargada), equivalentes al 0 % y 100 % de SOC, respectivamente.

En resumen, la función:

1. busca las dos tablas de OCV–SOC con temperaturas más cercanas a la medida (una inferior y otra superior);

2. interpola dentro de cada tabla para obtener el SOC asociado al voltaje dado (`interpolate_ocv_soc`);
3. interpola linealmente entre ambos valores según la temperatura medida; y
4. devuelve el resultado acotado al rango 0–100 %.

De esta forma, el cálculo del SOC considera la dependencia térmica de la curva OCV–SOC, obteniendo una estimación precisa y coherente con la temperatura real de operación de la batería.

`update_capacity_rls(int32_t delta_soc)`: Actualiza la estimación de la capacidad efectiva de la batería usando el algoritmo RLS a partir del cambio de SOC en una ventana ($\Delta SOC$) y a partir de la carga acumulada medida en esa ventana.

**Variables del algoritmo RLS:**

- `rls.theta` ($\theta$): es la estimación actual de la capacidad. Es la salida que se quiere ajustar.
- `rls.P` ($P$): es la varianza de la estimación de $\theta$.
- `rls.lambda` ($\lambda$): es un "factor de olvido" acotado entre 0 y 1, que da mayor peso a los datos recientes.
- `bateria.accumulated_charge`: es la carga integrada en la ventana. Se denomina como $y$.
- `delta_soc`: es el cambio de SOC en la ventana ($\Delta SOC$, con una escala interna de $[0, 100000] \equiv 0,100\,\% \times 1000$).

La función implementa el siguiente procedimiento:

1. Si $|\Delta SOC| <$ `MIN_DELTA_SOC`, no se actualiza (evita mover $\theta$ con ruido o ventanas muy pequeñas).
2. Realiza la normalización de $\phi_{\text{frac}} = |\Delta SOC|/100000$ (fracción del 0–1); $y =$ |`accumulated_charge`|.
3. Hace la lectura de estado: Carga $\lambda$, $P$:=`rls.P`, $t$:=`rls.theta`.
4. Calcula la ganancia RLS:

$$K \;=\; \frac{P\,\phi_{\text{frac}}}{\lambda + \phi_{\text{frac}}^2\,P}$$

5. Calcula el error de predicción mediante:

$$e \;=\; y \;-\; \phi_{\text{frac}}\,t$$

6. Actualiza del parámetro (capacidad)

$$t \;\leftarrow\; t \;+\; K\,e$$

7. Actualiza la varianza mediante la ecuación:
$$P_{\text{new}} = \frac{P - K\,\phi_{\text{frac}}\,P}{\lambda}$$
y se *satura* a $P_{\text{new}} \geq 1$ para evitar colapso numérico.

8. Actualiza los parámetros pertinentes al algoritmo RLS
`rls.theta` $\leftarrow$ máx(0, $t$), `rls.P` $\leftarrow P_{\text{new}}$.

9. Resetea la ventana: `bateria.accumulated_charge` $\leftarrow 0$ (la ventana se "consume" al actualizar).

El algoritmo ajusta $\theta$ (capacidad efectiva) para que la relación carga real transferida $\approx \phi_{\text{frac}} \cdot \theta$ se cumpla. Si, para un mismo $\Delta SOC$, la carga medida $y$ es *menor* a la estimada $\phi_{\text{frac}} t$, entonces $e < 0$ y $\theta$ disminuye (reflejando *degradación*). Si $y$ es *mayor*, $\theta$ aumenta. La ganancia $K$ depende de $P$, $\phi_{\text{frac}}$ y $\lambda$: cuanto mayor la varianza $P$ o la fracción $\phi_{\text{frac}}$, mayor el ajuste; $\lambda$ hace que $P$ no crezca sin control y prioriza datos recientes.

En resumen la función `update_capacity_rls()` incorpora la relación medida entre $\Delta SOC$ y la carga realmente transferida para corregir la capacidad estimada ($\theta$), manteniendo una ganancia adaptativa y memoria finita mediante $\lambda$. Esa capacidad efectiva es la base para el SOH.

`bms_init()`: Inicializa el estado interno del estimador de SOC/SOH y del algoritmo RLS a partir de las mediciones iniciales. La función implementa el siguiente procedimiento:

1. Calcula el voltaje promedio de las celdas en mV y la temperatura promedio en m°C.

2. Obtiene el SOC inicial llamando a la función `get_soc_from_ocv`, y lo copia en `estado->soc`.

3. Inicializa el SOH en 100 % y lo copia en `estado->soh`.

4. Calcula la capacidad nominal efectiva de la batería en mC mediante la siguiente fórmula:
`C_nominal` = `celdas_en_paralelo` × `capacidad_nominal_celda (mAh)` × 3600
donde `C_nominal` representa la capacidad nominal efectiva de la batería.

5. Inicializa la carga acumulada en 0 mC.

6. Inicializa las variables del algoritmo RLS: `rls.theta` $\leftarrow$ `Capacidad_nominal`, `rls.P` $\leftarrow$ `RLS_P_INIT`, `rls.lambda` $\leftarrow$ `RLS_LAMBDA`,
donde `RLS_P_INIT` y `RLS_LAMBDA` son constantes de inicialización definidas.

En resumen, `bms_init()` arranca el sistema con un SOC coherente con el OCV y la temperatura actual, establece un SOH inicial conocido, fija la capacidad nominal de la batería en unidades de mC y deja listo el algoritmo RLS para futuras actualizaciones.

`bms_update()`: Actualiza periódicamente el SOC y, cuando corresponde, ajusta la capacidad efectiva mediante el algoritmo RLS, para luego actualizar el SOH. La función tiene el siguiente procedimiento:

1. Lee la carga actual desde la variable `estado->carga`, calcula la diferencia de carga entre la carga actual y la última carga medida:

$$\Delta C = \texttt{current\_charge} - \texttt{bateria.previous\_charge}$$

2. Actualiza la última carga medida con la carga actual en la variable `bateria.previous_charge` y la carga total acumulada en `bateria.accumulated_charge`.

3. Computa $\Delta SOC_{\mathrm{CC}} = \dfrac{\Delta C}{Q_{\mathrm{nom}}} \cdot 100000$ y aplica: `bateria.soc` $\leftarrow$ `CLAMP` (`bateria.soc` $- \Delta SOC_{\mathrm{CC}}$).

4. Si la corriente medida es menor a 40 mA, el sistema considera que la batería está en reposo y comienza a registrar el tiempo de reposo. Cuando el tiempo de reposo supera los 25 mín, se asume que el sistema alcanzó una condición estable y se ancla el SOC utilizando el voltaje en circuito abierto (OCV).

5. Calcula la temperatura promedio en m°C y, para cada celda, estima el SOC por `get_soc_from_ocv()`.

6. Promedia los 16 SOC por celda y actualiza el SOC con el resultado.

7. Si la diferencia $|\Delta SOC|$ entre el valor actual y el previo excede el umbral `MIN_DELTA_SOC`, se actualizan la capacidad estimada mediante `update_capacity_rls()`, el valor de `estado->soh` y se almacena el nuevo SOC en `bateria.previous_soc`. Si no se cumple esta condición, no se ejecuta el algoritmo RLS, pero se continúa acumulando la carga para una futura ventana de reposo válida.

8. Despúes de actualizar el SOC y SOH, se resetea a 0 el tiempo de reposo.

9. Si en algún ciclo se tiene que |`estado->corriente`| $\geq$ `CURRENT_THRESHOLD`, se reinicia el tiempo de reposo.

### 5.2.7. Main

El archivo `main.c` implementa la función principal del *firmware*, encargada de inicializar módulos y establecer la secuencia general de funcionamiento del BMS. Su propósito es coordinar la interacción entre los distintos módulos del sistema (medición, protección, balanceo, etc.) garantizando que las tareas se ejecuten de forma ordenada y con tiempos de actualización definidos. Para lograr esto, el *firmware* se estructura en tres tareas principales bajo FreeRTOS:

- **Tarea Coordinadora:** se encarga de sincronizar la ejecución de las demás tareas y de controlar el flujo general del sistema, incluyendo la gestión de modos de operación y eventos globales como el corte general.
- **Tarea General:** agrupa las funciones de adquisición de datos y control de seguridad. En cada ciclo, esta tarea realiza la medición de parámetros eléctricos y térmicos, verifica las protecciones y ejecuta las acciones correspondientes sobre los circuitos de carga y descarga.
- **Tarea de Comunicación:** maneja el intercambio de información con el entorno externo, en particular con la GUI. Se encarga de transmitir los valores medidos y el estado general de la batería, así como de recibir los comandos de configuración enviados por el usuario.

A continuación se describe lo que realiza el `app_main()` del proyecto, donde se realizan las inicializaciones de los módulos y se generan las tareas. Primero se fija la frecuencia de trabajo del microcontrolador en 40 MHz y se configura un pin como fuente de *wake-up* para el modo de bajo consumo *light sleep*, es decir, reactiva el microcontrolador mediante un evento externo originado por la GUI. Por más detalles sobre el funcionamiento en bajo consumo y la elección de frecuencia de trabajo ver Subsección 5.2.8.

Se inicializan los pines de los dos botones como entradas con *pull-up* interno, lo que habilita modos de interacción local, y se procede a llamar a las funciones de incialización de cada módulo (`init_uart()`, `init_parametros()`, `init_medicion()`, etc.), y se activa la protección de predescarga (Subsección 5.2.2).

Por último se crean las tres tareas ya mencionadas y se define la secuencia de arranque:

- Tarea de Coordinación (`taskCoordinator`) con prioridad `TASK_PRIO`;
- Tarea General (`taskGeneral`) con prioridad `TASK_PRIO+1`;
- Tarea de Comunicación (`taskComm`) con prioridad `TASK_PRIO+2`.

Donde `TASK_PRIO` es una variable no configurable con un valor por defecto de 5. La tarea General se crea inicialmente suspendida; la Coordinadora es la que la habilita cuando el sistema está listo. Se da la mayor prioridad a la tarea de Comunicación para garantizar que el enlace con la GUI no pierda eventos ni tramas, la tarea General queda con prioridad intermedia para cumplir la cadencia de adquisición y protecciones, y la Coordinadora, con prioridad base, orquesta la activación de modos y el flujo general del sistema. A continuación se describen en mayor detalle el funcionamiento de cada tarea.

### Tarea Coordinadora `taskCoordinator`

La tarea coordinadora define la secuencia general de operación del sistema y gestiona las rutinas de ahorro energético. Su función principal es mantener una

cadencia de trabajo constante y controlar los modos de funcionamiento del BMS según las acciones del usuario.

Al inicio de cada iteración, se registra el tiempo actual de funcionamiento en:

$$\texttt{modo.task\_time} \leftarrow \texttt{esp\_timer\_get\_time()},$$

el cual sirve como referencia para calcular la duración total del ciclo de tareas antes de entrar en *light-sleep*.

A continuación, la tarea coordinadora reanuda la ejecución de la tarea general (`taskGeneral`), la cual comienza suspendida, y queda en un estado de espera hasta recibir una notificación del fin del ciclo de trabajo de `taskGeneral`. De esta forma, la coordinadora marca el ritmo de ejecución, primero habilita el ciclo de medición y control, espera su finalización, y recién entonces continúa con su propio ciclo.

Una vez concluido el ciclo de `taskGeneral`, la coordinadora verifica el estado de los botones físicos para determinar si el usuario solicitó un cambio de modo de operación:

- **Botón 1 – Corte general:** al ser presionado, conmuta el estado del modo de *corte general*. Si se activa, abre los caminos de carga y descarga, y corta el balanceo de celdas si estaba activo. Si se desactiva, restablece los caminos de corriente. Al activarse o desactivarse el corte general se emite un *beep* de 500 ms.

- **Botón 2 – Configuración:** al ser presionado, conmuta el modo de *configuración*. Al activarse, se guarda el tiempo de inicio en la variable `modo.tiempo_inicio_config`, si el modo permanece activo más allá del tiempo definido por `TIMEOUT_CONFIG` (60 s por defecto), se desactiva automáticamente. De lo contrario el modo se desactiva cuando recibe un comando por GUI o cuando vuelve a presionarse el botón 2. Al activarse o desactivarse el modo de configuración se emite un *beep* de 500 ms.

Luego de procesar los botones, el sistema calcula el tiempo total consumido por las tareas mediante:

$$\texttt{modo.task\_time} \leftarrow \texttt{esp\_timer\_get\_time()} - \texttt{modo.task\_time},$$

obteniendo así la duración del ciclo completo en microsegundos. Con este valor, la tarea decide cuánto tiempo resta hasta completar el período objetivo definido por la macro `LIGHT_SLEEP_US`, la cual establece la frecuencia de operación del sistema.

En este proyecto, `LIGHT_SLEEP_US` se fija en 200 ms, lo que permite mantener una cadencia de cinco mediciones por segundo. Esta cadencia y el valor de `LIGHT_SLEEP_US` viene dado por el valor elegido en la macro `SAMPLES_PER_SECOND`, que fija la cantidad de mediciones a tomar por segundo.

Una vez calculado el tiempo total de trabajo de las tareas, la tarea evalúa si entrar en *light-sleep* bajo las siguientes condiciones:

- **Modo configuración activado:** la tarea no entra en *light-sleep*, permanece activa y espera en estado bloqueado durante el tiempo restante del período:

  $$t_{\text{espera}} = \texttt{LIGHT\_SLEEP\_US} - \texttt{modo.task\_time}.$$

  De este modo, se mantiene la cadencia de muestreo sin interrumpir la comunicación con la GUI.

- **Modo configuración desactivado:** el sistema entra en *light-sleep* durante el tiempo restante del período. Antes de dormir, los pines críticos (buzzer, LEDs, MOSFETs, etc.) se mantienen en su último estado mediante la función `rtc_gpio_force_hold_en_all()`. Luego se configura el temporizador de reactivación con:

  `esp_sleep_enable_timer_wakeup`(`LIGHT_SLEEP_US` − `modo.task_time`),

  y se ejecuta `esp_light_sleep_start()`. Al despertar, los pines se liberan (`rtc_gpio_force_hold_dis_all()`) y comienza una nueva iteración.

En resumen, la tarea `taskCoordinator` sincroniza la ejecución de las demás tareas, supervisa los modos de operación y garantiza que el microcontrolador permanezca dormido el resto del tiempo de cada ciclo completo, reduciendo así el consumo sin alterar la cadencia de muestreo. Esto se ejemplifica en el siguiente pseudocódigo.

**Algorithm 12** Tarea coordinadora

```
1: procedure taskCoordinator
2:     loop
3:     Se inicializa modo.task_time en tiempo de inicio del ciclo
4:     Se resume la Tarea General
5:     S espera al fin del ciclo de la tarea General
6:     if Se apretó botón 1 then
7:         Se emite un beep
8:         if No esta activo el corte_general then
9:             Se activa el corte_general
10:            Se desactiva el balanceo de celdas
11:        else
12:            Se desactiva el corte_general
13:        end if
14:    end if
15:    if Si se apretó botón 2 then
16:        Se emite un beep
17:        if No esta activo el modo de configuración then
18:            Se activa el modo de configuración
19:            Se registra en modo.tiempo_inicio_config el tiempo de inicio
20:        else
21:            Se desactiva el modo de configuración
22:        end if
23:    end if
24:    if Esta en modo de configuración por más de 60 s then)
25:        Se desactiva el modo de configuración
26:        Se emite un beep
27:    end if
28:    Se registra el tiempo del ciclo de trabajo en modo.task_time
29:    if Si esta en modo de configuración then
30:        Se espera un tiempo LIGHT_SLEEP_US - modo.task_time
31:    else
32:        Se mantienen fijos los pines
33:        Se setea un timer de LIGHT_SLEEP_US − modo.task_time
34:        Se entra en light-sleep
35:        Se despierta una vez transcurrido el timer
36:        Se liberan los pines
37:    end if
38: end procedure
```

### Tarea General (`taskGeneral`)

La Tarea General ejecuta el ciclo principal de adquisición, seguridad y mantenimiento del estado de la batería. En cada iteración realiza, mediciones verificación de protecciones, cada 5 iteraciones realiza la transmisión de datos y (si corresponde) balanceo pasivo, y por último cada 50 iteraciones actualiza el estado del SOC y del SOH.

Cada iteración comienza con la toma de mediciones y verificaciones de protecciones de la siguiente forma:

1. Si el corte general no está activo, se mide la corriente con la función `medir_corriente()`, se evalúa la protección de sobrecorriente con la función `proteccion_corriente()` y luego se integra la carga con la función `medir_carga()`. Por otro lado, si el *corte general* está activo, para evitar lecturas flotantes de la corriente se fuerza `estado.corriente = 0` y no se actualiza la carga en esa iteración, ni se verifica la protección de corriente.

2. Se miden las temperaturas con la función `medir_temperatura()` y se evalúan las protecciones térmicas mediante la función `proteccion_temperatura()`.

3. Se miden las tensiónes de cada celda con la función `medir_voltajes()` y se verifica que no se esté en sobrecarga o sobredescarga mediante la función `proteccion_carga_descarga()`.

Por más información sobre las funciones utilizadas en esta etapa, ver Subsección 5.2.1 y Subsección 5.2.2.

Cada 5 iteraciones (1 s), la Tarea General llama a la función `transmision_estado()` y envía por UART el estado completo de la batería como se describe en Sección 5.2.5. Esta función viene seguida por un tiempo de espera de hasta ∼50 ms para asegurar el envío de los datos.

Una vez transmitidos los datos, se verifica que no esté activo el modo de corte general ni ninguna protección que corte el camino de carga de la corriente, luego se ejecuta la función de `balanceo_pasivo()`. Si se activó una de las protecciones pertinentes, el balanceo se omite. Por más detalles sobre las función de balanceo, ver Subsección 5.2.3.

Después de las primeras 50 iteraciones (10 s), se ejecuta la función `bms_init()` para fijar el SOC inicial por OCV, la capacidad nominal y el estado inicial de las variables RLS. Luego procede a actualizar periódicamente el SOC y el SOH cada 50 iteraciones llamando a la función `bms_update()`. Por más detalles sobre estas funciones, ver Subsección 5.2.6.

Al final de cada iteración, la tarea General notifica a la tarea Coordinadora que terminó el ciclo de trabajo con `xTaskNotifyGive(hTaskCoord)` y luego se suspende con `vTaskSuspend(NULL)` hasta ser reanudada por `taskCoordinator`. Este esquema asegura la cadencia deseada y permite que el microcontrolador entre en *light-sleep* el tiempo restante del período.

El funcionamiento de esta tarea se resume en el siguiente pseudocódigo:

**Algorithm 13** Tarea General (`taskGeneral`)

**procedure** TASKGENERAL
loop
**if** No esta activo el *corte general* **then**
Se mide corriente con `medir_corriente()`
Se verifica protección de corriente `proteccion_corriente()`
Se mide la carga con `medir_carga(&estado)`
**else**
Se fija la corriente en 0 A
**end if**
Se mide temperatura con `medir_temperatura()`
Se verifican protecciones térmicas con `proteccion_temperatura()`
Se miden las tensiones de las celdas con `medir_voltajes()`
Se verifican protecciones contra sobrecarga y sobredescarga con `proteccion_carga_descarga()`
**if** Cada 10 s **then**
**if** No se inicializo el SOC y SOH **then**
Se inicializan SOC y SOH con `bms_init()`
**end if**
Actualiza los estados de SOC y SOH con `bms_update()`
**end if**
**if** Cada 1 s **then**
Se transmite el estado de la batería con `transmision_estado()`
Espera 30 ms a que se envien los datos
**if** Si no hay protección activada que corte la carga de corriente y modo *corte general* desactivado **then**
Se chequea la necesidad de hacer balanceo pasivo con `balanceo_pasivo(&estado)`
**end if**
**end if**
**if** Se activa protección severa **then**
Se activa el modo *corte general*
**end if**
Se aumenta en 1 la iteración
Si la iteración es igual a 10 veces `SAMPLES_PER_SECOND`, se resetea a 0
Notifica a la Tarea Coordinadora que termino su ciclo
Se auto suspende la Tarea General
**end procedure**

### Tarea de comunicación (`taskComm`)

Esta tarea corre en bucle continuo atendiendo el puerto `UART0`, donde `UART0` es uno de los puertos serie integrados en el microcontrolador. En este proyecto, el puerto `UART0` se utiliza como interfaz principal de comunicación entre el BMS y la GUI, tanto para enviar el estado del sistema como para recibir comandos de configuración.

En cada iteración la tarea:

1. Intenta leer hasta 255 bytes con un *timeout* de 100 ms.
2. Si no se reciben datos se espera 10 ms y vuelve a intentar.
3. Si lee datos, guarda los datos en un *buffer* y coloca `'\0'` al final del *buffer* para permitir *parsers* del tipo `strtok`/`atoi`.
4. Recorre el *buffer* ignorando delimitadores `X, Z, CR, LF, ',' y espacio` hasta hallar una de las letras de comando, `'I'` o `'C'`.
5. Si no hay comando válido comienza la siguiente iteración.

Si se recibe un comando válido:

`'I'` — **Inicialización de BMS:** desactiva los modos de *corte general* y *configuración*, emite un *beep* de 500 ms y transmite el estado actual de la batería.

`'C'` — **Configuración de parámetros:** si el *modo de configuración* está activo, le pasa los bytes posteriores a la letra `'C'` a la función de `recepcion_param()` para aplicar cambios; luego transmite el estado, desactiva el *modo de configuración* y emite un *beep* de 500 ms.

Para evitar uso excesivo de CPU, la tarea inserta un retardo de 10 ms al final de la iteración. Se destaca que la tarea sólo procesa el primer comando válido detectado en el bloque recibido, lo que simplifica sincronización y evita condiciones de carrera con la *Tarea General*. En el siguiente pseudocódigo se resume el funcionamiento de esta tarea.

**Algorithm 14** Tarea de Comunicación (`taskComm`)

**procedure** TASKCOMM
    loop
    Se trata de leer datos recibidos en el puerto `UART0` y guardarlos en un *buffer*
    **if** Si no se recibieron datos **then**; Se pasa a la siguiente iteración del loop
    **end if**
    Si se reciben datos se agrega `'\0'` al final del *buffer*
    Se lee el *buffer* buscando el comando: 'I' o 'C'
    Se guarda comando en la variable `cmd`
    **if** Si no se recibió comando **then**
        Se pasa a la siguiente iteración del loop
    **end if**
    **if** *cmd* = 'I' **then**
        Desactiva el *Corte General* y el *Modo de Configuración*
        Emite un *beep*
        Transmite el estado de la batería con `transmision_estado()`
    **else if** *cmd* = 'C' **then**
        **if** *Modo de Configuración* esta activo **then**
            Procesa los datos con `recepcion_param()`
            Transmite el estado de la batería con `transmision_estado()`
            Desactiva el *Modo de Configuración*
            Emite un *beep*
        **end if**
    **end if**
    Descanso 10 ms para no saturar CPU
**end procedure**

### 5.2.8. Reducción de consumo del microcontrolador

El *firmware* implementa una serie de estrategias orientadas a reducir el consumo del microcontrolador durante su funcionamiento normal. El `ESP32-S3` posee una frecuencia de reloj nominal de 240 MHz y dos núcleos activos. Sin embargo, en este proyecto se configuró el microcontrolador para operar en modo *single-core* y a su frecuencia mínima de 40 MHz, suficiente para mantener la cadencia de 5 mediciones por segundo con amplio margen. Esta reducción implica una disminución considerable de la corriente de operación, según las especificaciones del fabricante:

- En modo *dual-core* a 240 MHz se estima un consumo típico de 81,3 mA;
- En modo modo *single-core* a 40 MHz se estima un consumo típico de 21,8 mA.

Estos valores fueron tomados de la hoja de datos del microcontrolador y se emplean como referencia teórica [14]. Aparte de la configuración de la frecuencia

de trabajo, el `ESP32-S3` ofrece distintos modos de bajo consumo, cada uno con características específicas de retención de memoria, velocidad de recuperación y consumo de corriente. Estos modos de bajo consumo son:

- **Deep-sleep:** el microcontrolador apaga completamente la CPU, la RAM y la mayoría de los periféricos, conservando únicamente el dominio de energía del RTC y las variables almacenadas en memoria *RTC slow*. Si bien el consumo típico en este modo es inferior a 30 µA, al despertar se produce un reinicio completo del sistema, lo que implica volver a ejecutar la inicialización completa del *firmware*. Por esta razón no se utilizó este modo en el BMS, dado que requeriría reconfigurar el sistema y perdería continuidad en las tareas.

- **Light-sleep:** En este modo la CPU, la RAM y los periféricos digitales se desconectan del reloj principal y reducen su voltaje de alimentación, pero su estado interno se mantiene. Al salir de este modo (despertar), la ejecución continúa exactamente donde se detuvo, sin pérdida de contexto ni reinicio. Además, este modo es configurable, permitiendo seleccionar qué periféricos permanecen activos. En este proyecto se utiliza este modo porque ofrece un equilibrio óptimo entre ahorro energético y tiempo de recuperación, donde se estima un consumo de aproximadamente 240 µA, según los valores de referencia del fabricante.

- **Modos adicionales:** el `ESP32-S3` también dispone de otros estados intermedios, como el *modem-sleep* (donde solo se apagan los módulos Wi-Fi/BLE) y el *standby*, de uso común en aplicaciones con conectividad inalámbrica, pero irrelevantes en este diseño ya que no utiliza comunicación inalámbrica.

El control del modo *light-sleep* se realiza dentro de la Tarea Coordinadora (ver Subsección 5.2.7), que suspende el microcontrolador entre iteraciones de medición.

En resumen, la reducción de consumo del microcontrolador se basa en tres medidas complementarias:

1. reducción de la frecuencia de reloj a 40 MHz;
2. uso de un único núcleo de procesamiento;
3. suspensión periódica en modo *light-sleep* entre ciclos de medición.

### 5.2.9. Ensayos de módulos

Con el objetivo de validar el correcto funcionamiento del *firmware*, se desarrollaron ensayos individuales para cada uno de los módulos del BMS. Estos ensayos se implementaron como rutinas de ensayo independientes dentro del mismo entorno de desarrollo, de modo que cada módulo pudiera verificarse por separado antes de la integración final del sistema.[1] De esta forma fue posible depurar las

[1] Los códigos de prueba pueden consultarse en la carpeta `Firmware/tests` en el repositorio del proyecto: `gitlab.fing.edu.uy/autobms/main`.

funciones, confirmar la respuesta esperada ante distintas condiciones y evaluar la estabilidad de las mediciones y protecciones.

### Ensayo del módulo de protección

Para el módulo de protección se crearon varios conjuntos de valores artificiales de corriente, temperatura y tensión, donde cada conjunto activa una protección específica. Luego se creo un test de loop infinito donde en cada iteración, se van llamando a las funciones de protección con un nuevo conjunto de valores artificiales, donde para cada iteración se verifica que el sistema active la(s) protección(es) correspondiente(s). Para simplificar la verificación se asoció cada tipo de protección a un código visual mediante los indicadores `LED1` y `LED2`, lo que permitió confirmar la activación correcta de cada condición. Adicionalmente se verificó la actuación de los transistores de potencia, observando la apertura y cierre de los caminos de carga y descarga según las protecciones activadas. Para avanzar de una iteración a otra se utilizó el `Botón1` de la placa, que conmuta entre distintos casos de ensayo.

### Ensayo del módulo de medición

Para este módulo se ejecutaron las funciones de medición de forma secuencial, y se imprimieron por terminal los valores obtenidos. Los resultados se verificaron experimentalmente de la siguiente manera:

- Para las tensiones se midió el voltaje de cada celda con un multímetro y se comparó con el valor reportado por el BMS.
- Para la corriente: se utilizó una fuente de tensión con control de corriente y se comparó el valor indicado en la fuente con la lectura del sistema.
- Para la temperatura se contrastaron los valores leídos con un termómetro digital de referencia, verificando la calibración de los sensores TMP36.

### Ensayo del módulo de balanceo

El ensayo del módulo de balanceo consistió en simular un conjunto de tensiones de celda en las que la diferencia de tensión entre una mitad de las celdas y la otra supera el umbral de activación del balanceo, por lo que solo la parte del conjunto con mayor tensión activa el balanceo. Tras un intervalo de tiempo, se modificaron los valores de tensión simulados de forma que las condiciones se invirtieran, es decir, las celdas que inicialmente no se balancean pasaron a presentar una tensión superior y se balancean. La verificación se realizó observando el encendido de los indicadores LED correspondientes a cada celda, confirmando que las resistencias de descarga se activaban únicamente en los casos esperados según los umbrales configurados.

#### Ensayo del módulo de estimación de SOC y SOH

Para el módulo de estimación del estado de carga y salud se realizaron dos tipos de ensayos: simulaciones de software y ensayos experimentales.

En las simulaciones se introdujeron valores de corriente y carga acumulada que representaban escenarios de descarga y reposo conocidos, verificando que la lógica de cálculo de SOC y la actualización adaptativa de capacidad (RLS) respondieran consistentemente.

Posteriormente se ejecutó un ensayo práctico con una batería real, en el que se programó una secuencia cíclica de descarga → reposo de 25 min → descarga. Durante el reposo, el SOC se anclaba al OCV, y tras varios ciclos se verificó que el SOH permaneciera en torno al 100 %, como corresponde a una batería nueva.

Para la validación del SOC, se utilizaron las tablas experimentales OCV–SOC obtenidas para las celdas utilizadas, comparando los valores estimados por el sistema con los esperados según dichas tablas.

#### Ensayo del módulo de parámetros

El ensayo del módulo de parámetros tuvo por objetivo confirmar la comunicación bilateral entre el BMS y la GUI. Durante el ensayo se verificó que los comandos enviados desde la interfaz fueran correctamente recibidos y procesados por la función `recepcion_param()`, actualizando los valores almacenados en memoria NVS. Asimismo, se comprobó que los nuevos parámetros se mantuvieran tras un reinicio del microcontrolador y que el BMS transmitiera por UART el estado actualizado de la batería.

## 5.3. Interfaz Gráfica

### 5.3.1. Objetivo y alcance

La Interfaz Gráfica (GUI) Auto-BMS ofrece un entorno para visualizar en tiempo real el estado del sistema y modificar su configuración de forma segura. Desde la GUI el usuario puede observar tensiones, corriente, temperaturas, SOC, SOH y el estado del balanceo por celda. Además, puede efectuar cambios en los parámetros del sistema y ejecutar acciones de control. La aplicación, desarrollada en `Python` con `Tkinter`, se ejecuta en una PC y se encarga de gestionar la conexión con el BMS y procesar las tramas recibidas. La comunicación con el *hardware* se realiza mediante UART.

### 5.3.2. Arquitectura de la GUI

La aplicación se implementa con una clase principal llamada `AutoBMSApp` que construye una ventana con dos pestañas: *Monitoreo* y *Configuración*.

La pestaña de *Monitoreo* agrupa elementos de estado (conexión, modo del BMS), medidas en tiempo real (tensiones de las celdas, corriente, temperaturas,


Figura 5.1: Pestaña de Monitoreo de la GUI.

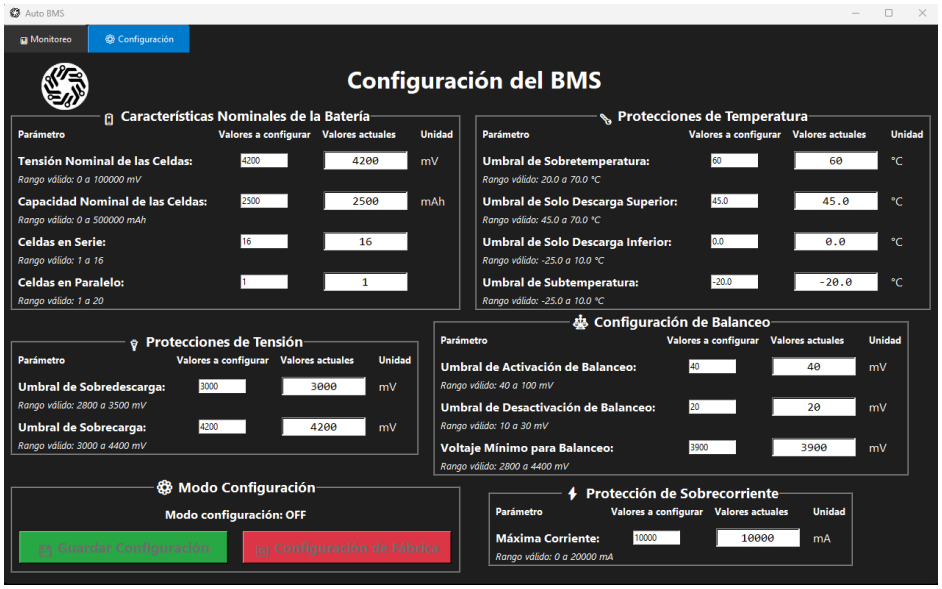


Figura 5.2: Pestaña de Configuración de la GUI.

SOC y SOH), el estado de balanceo de las celdas y el estado de las distintas protecciones. En la Figura 5.1 se puede ver la estructura de la pestaña.

La pestaña de *Configuración* organiza los parámetros en bloques temáticos: características nominales, térmicos, de corriente, de tensión y de balanceo, mostrando para cada uno valor actual, valor propuesto, unidad y rango permitido. En la Figura 5.2 se muestra la estructura de la pestaña.

El uso previsto de la interfaz se describe en la Sección B.3.

### 5.3.3. Comunicación con el *hardware*

La GUI abre el puerto serie a 115200 bps con lectura no bloqueante. La recepción se realiza mediante un bucle programado en el hilo de la interfaz que evita congelamientos de la ventana. Se estableció una tasa de refresco de 10 Hz.

En paralelo, un hilo vigila la disponibilidad física del puerto, si la conexión UART se interrumpe (por ejemplo, si se desconecta el cable), la aplicación lo detecta y guía al usuario para recuperar la conexión sin cerrar la aplicación.

Los comandos hacia el BMS son los siguientes:

- Inicio: Inicia el BMS y desactiva el corte general. Se envía la cadena `\I\n`.
- Configuración: Envía la configuración ingresada por el usuario. Comienza con el caracter `C`, seguido de cada uno de los parámetros ingresados, separados por comas, y finalizando con un salto de línea (`\n`).

Para enviar una configuración desde la GUI al BMS se construye una trama que contiene los valores establecidos en la pestaña de *Configuración*. En la Tabla 5.1 se detallan los parámetros enviados. En esta trama, los valores configurados se envían en formato punto fijo, con un factor de escala $\times 1000$, simplificando la decodificacion de parte del BMS dado que este almacena todas las cifras de la misma forma.

#### Validación y reglas de coherencia

Antes de enviar una nueva configuración al BMS, la interfaz aplica dos niveles de verificación.

Los parámetros involucrados corresponden a los umbrales de protección y control definidos por el usuario: $V_{\text{mín}}$ y $V_{\text{máx}}$ representan las tensiones mínima y máxima por celda, respectivamente; $V_{\text{bal}}$ es la tensión mínima de activación del balanceo; $T_{\text{mín}}$ y $T_{\text{máx}}$ son las temperaturas de protección por subtemperatura y sobretemperatura; $T_{\text{solo-desc,inf}}$ y $T_{\text{solo-desc,sup}}$ delimitan la zona en la que solo se permite la descarga; finalmente, $\Delta V_{\text{bal,act}}$ y $\Delta V_{\text{bal,desact}}$ corresponden a los umbrales de activación y desactivación del balanceo, definiendo una histéresis $\Delta V_{\text{hys}}$.

1. Validación individual: se comprueba que cada parámetro se encuentre dentro del rango permitido por los límites definidos.
2. Validación cruzada: se aplican reglas de coherencia global que aseguran configuraciones físicamente consistentes:
   *a*) $V_{\text{mín}} < V_{\text{máx}}$: garantiza un rango de tensión válido.
   *b*) $V_{\text{bal}} \in [V_{\text{mín}}, V_{\text{máx}}]$: el balanceo solo se habilita dentro del rango de voltajes permitido.
   *c*) $T_{\text{mín}} < T_{\text{solo-desc,inf}} < T_{\text{solo-desc,sup}} < T_{\text{máx}}$: orden de umbrales de temperatura correcto: subtemperatura < zona solo descarga inferior < zona solo descarga superior < sobretemperatura.
   *d*) $\Delta V_{\text{bal,act}} > \Delta V_{\text{bal,desact}}$: asegura histéresis de balanceo para evitar oscilaciones.

Los mensajes de error se muestran de manera explicativa, indicando qué parámetro incumple y por qué, para que el usuario pueda corregirlo en el acto. Si el usuario no resuelve los conflictos existentes, o si el BMS no habilita el modo de configuración, la GUI no permite el envío y lo informa al usuario.

### 5.3.4. Modelo de datos y mapeo de protocolo

Cada trama CSV recibida debe estar delimitada por los caracteres `X` y `Z`, y contener exactamente 45 elementos para ser considerada válida. La información transmitida por el BMS se organiza de la siguiente forma:

- 14 parámetros de configuración (valores actuales en mV, mA y °C para mostrar en la pestaña *Configuración*).
- 5 magnitudes de medición (corriente, dos temperaturas, SOC y SOH).
- 1 indicador binario del estado general del BMS.
- 1 vector de 16 bits que representa el estado de balanceo de cada celda.
- 16 tensiones de celdas.
- 7 banderas de protección.
- 1 bandera de *Modo Configuración*.

A partir de estos datos, la GUI calcula derivados útiles para diagnóstico (mínimo, máximo y diferencia de tensión entre celdas y tensión total de la batería como suma de las tensiones de las celdas), y refleja estados compuestos, por ejemplo "Cargando" o "Descargando", según las banderas activas.

### 5.3.5. Gestión de estados, errores y notificaciones

La interfaz mantiene un estado interno mínimo y observable: conexión al puerto, modo de configuración, y último estado de carga/protecciones. Los errores derivados de entrada/salida (puerto inexistente, timeouts, escritura fallida) se capturan y se traducen a notificaciones guiadas (*message boxes*) sin interrumpir el hilo principal. Para la recepción se adopta una política conservadora: el búfer se recorta si crece en exceso y las tramas incompletas se descartan sin propagar valores inválidos a la GUI. Con esto logra que el usuario nunca vea lecturas "a medias" ni una ventana bloqueada por fallos de comunicación.

# Capítulo 6

# Resultados

El presente capítulo expone los principales resultados prácticos obtenidos durante la fase de validación final del dispositivo. En esta etapa se evaluaron dos aspectos centrales del sistema: la consistencia del algoritmo de estimación del SOH y el consumo energético promedio del BMS en sus distintos modos de funcionamiento.

## 6.1. Medición de Consumo

El consumo de energía del sistema constituye un parámetro relevante a efectos de evaluar su viabilidad práctica. Un valor de consumo excesivo podría comprometer su incorporación en aplicaciones reales, dado que el BMS debe desempeñar su función de protección sin convertirse en una carga significativa para la batería.

### 6.1.1. Sistema de Ensayo

En la Figura 6.1 se presenta el sistema empleado para las mediciones de consumo. Se utilizó un Otii Arc como instrumento principal, el cual mide la corriente suministrada al dispositivo a través del BMS, utilizando para ello una resistencia *shunt* ($R_S$ en la figura). Esta configuración permite registrar de forma directa la corriente tomada desde la batería, reflejando así el consumo total del dispositivo en condiciones reales de operación.

El cable rojo en la figura simboliza el positivo de la batería, que corresponde al polo positivo de la celda 16, tensión máxima del sistema.

### 6.1.2. Comparación de Frecuencias de Muestreo

Con el objetivo de evaluar el impacto de la frecuencia de medición sobre el consumo energético, se midió la corriente promedio del dispositivo operando en modo normal para tres frecuencias distintas: 3 Hz, 5 Hz y 10 Hz. En este modo, el sistema ejecuta ciclos periódicos de medición que implican activar los multiplexores, amplificadores y otros bloques internos, por lo que se espera que una mayor

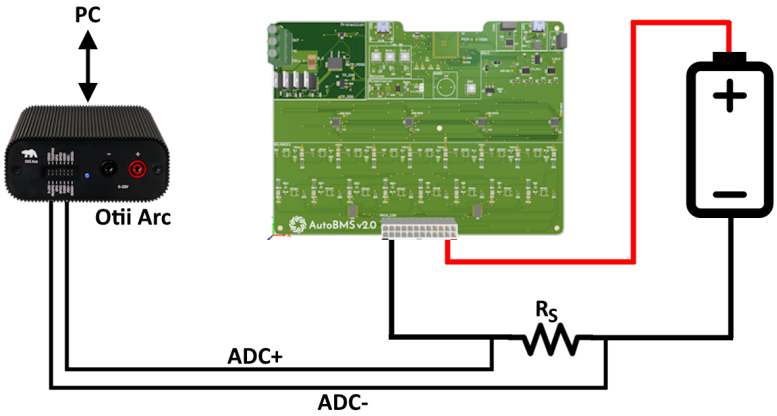


Figura 6.1: Sistema utilizado para la medición de consumo de corriente.

cantidad de mediciones por segundo incremente el consumo promedio.

La Figura 6.2 muestra los perfiles de consumo obtenidos para cada una de las frecuencias analizadas (de arriba hacia abajo: 3 Hz, 5 Hz y 10 Hz). Como se observa, el aumento en la tasa de muestreo produce una mayor densidad de picos de corriente, lo que se refleja en un incremento del consumo medio.

Los valores promedio medidos fueron:

- 3 Hz**:** 1,12 mA.
- 5 Hz**:** 1,23 mA.
- 10 Hz**:** 1,48 mA.

Estos resultados confirman una relación creciente entre la frecuencia de muestreo y el consumo total, aunque dentro de un rango acotado que sigue siendo compatible con aplicaciones de baja potencia.

### 6.1.3. Comparación de Frecuencias de la CPU

También se evaluó el impacto de la frecuencia del reloj del procesador sobre el consumo energético del sistema. Para ello, se midieron los perfiles de corriente operando en modo normal, a 5 Hz de frecuencia de muestreo, con dos configuraciones de frecuencia: el valor por defecto del microcontrolador (240 MHz) y la frecuencia reducida utilizada en la versión final del *firmware* (40 MHz).

La disminución de la frecuencia del reloj reduce las pérdidas dinámicas en el núcleo del MCU, lo que se traduce en menores consumos promedio y picos de corriente menores durante los ciclos de medición. La Figura 6.3 muestra los perfiles obtenidos para ambas configuraciones.

Los resultados cuantitativos se resumen a continuación:

- **Consumo promedio a** 240 MHz**:** 1,38 mA.

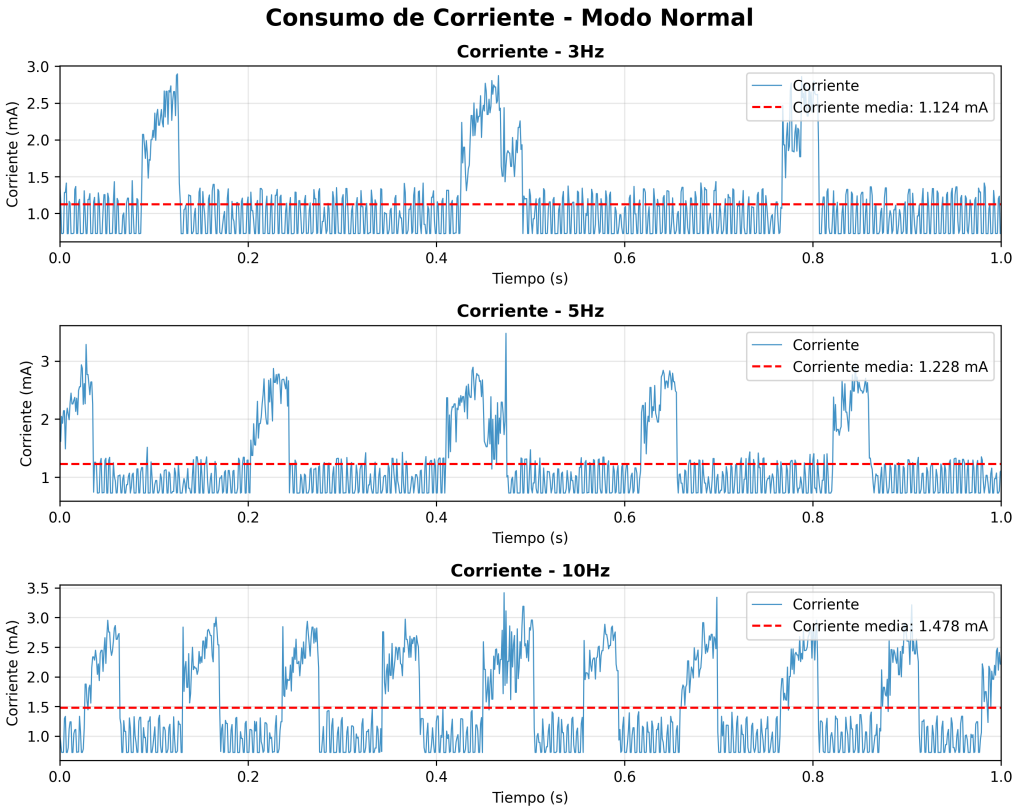


Figura 6.2: Perfil de consumo del dispositivo en modo normal para frecuencias de muestreo de 3 Hz, 5 Hz y 10 Hz (de arriba hacia abajo).

- **Consumo promedio a** 40 MHz**:** 1,24 mA.
- **Pico máximo a** 240 MHz**:** 4,44 mA.
- **Pico máximo a** 40 MHz**:** 3,22 mA.

Se observa una reducción tanto en el consumo promedio como en la magnitud de los picos de corriente al operar a 40 MHz, lo que confirma el beneficio energético de utilizar una frecuencia de CPU más baja sin afectar el desempeño requerido por las tareas del sistema. En términos cuantitativos, esta reducción de frecuencia implica un ahorro aproximado del 10,1 % en el consumo promedio.

### 6.1.4. Perfil de Consumo

Se caracterizó el consumo del sistema en dos escenarios: modo de operación normal y modo configuración. En el primero, el dispositivo ingresa periódicamente en *light-sleep*, mientras que en el segundo permanece activo de forma continua. La Figura 6.4 muestra los perfiles medidos para ambos modos. Los picos de corriente corresponden a los instantes en los que el dispositivo ejecuta el ciclo de medición, durante el cual se energizan los componentes asociados.

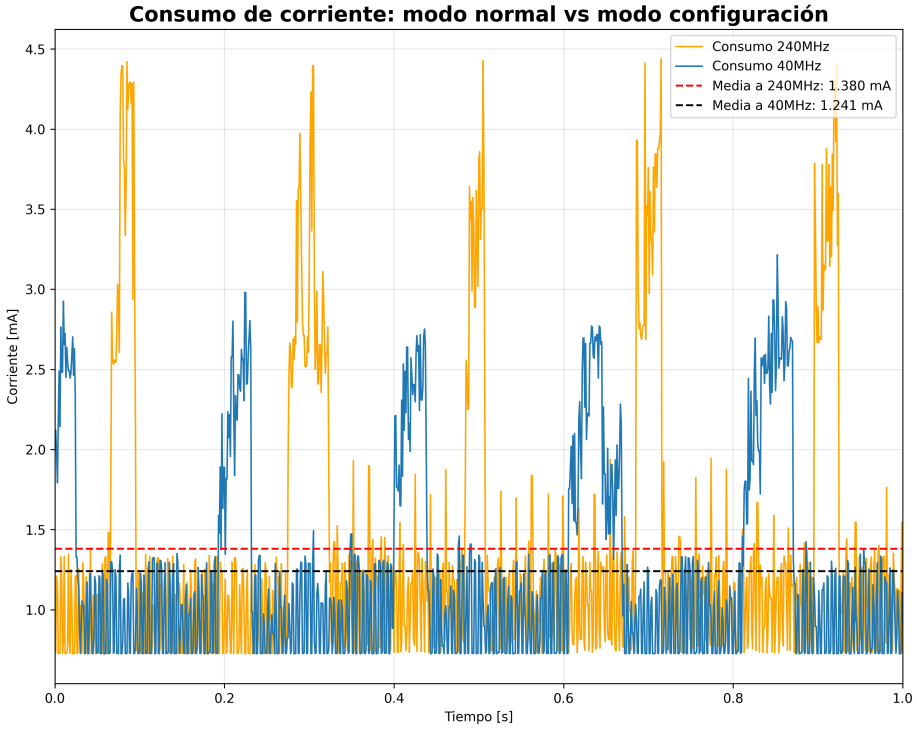


Figura 6.3: Comparación del consumo del dispositivo en modo normal para frecuencias de CPU de 240 MHz y 40 MHz.

En la versión final del *firmware*, el sistema realiza 5 mediciones por segundo, lo que explica la frecuencia de los picos observados. Los valores promedio registrados son:

- **Modo configuración:** 2,05 mA.
- **Modo normal:** 1,23 mA.

La comparación entre ambos modos muestra una reducción del consumo promedio del orden del 40 % al operar en modo normal respecto del modo configuración. Esto refleja la efectividad del uso de *light-sleep* en reducir la corriente total consumida por el dispositivo.

Considerando la capacidad de la batería construida y utilizada en el proyecto, de 2500 mA h, este consumo permite una operación continua estimada de 2033 h $\approx 84,7$ días.

## 6.2. Estimación del SOH

Debido a las limitaciones de tiempo asociadas a la etapa de validación experimental, no fue posible realizar un proceso exhaustivo de caracterización del SOH a lo largo de múltiples ciclos completos de carga y descarga. En su lugar, se optó

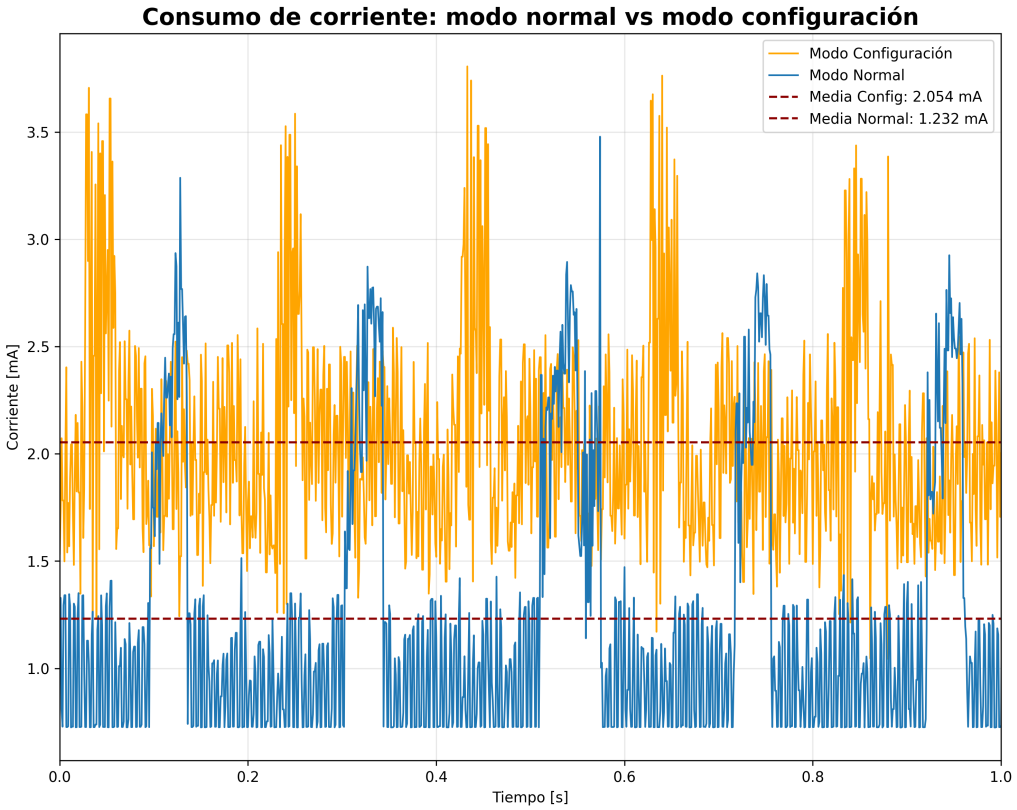


Figura 6.4: Perfil de consumo del dispositivo en los modos de operación normal y configuración.

por evaluar la consistencia del estimador implementado, verificando su estabilidad y coherencia bajo condiciones controladas de operación. Dado que el algoritmo de estimación de SOH se apoya en la estimación previa del SOC, este ensayo permite validar indirectamente el correcto funcionamiento del conjunto de estimadores.

El procedimiento de ensayo consistió en aplicar ventanas sucesivas de descarga, intercaladas con períodos de reposo de aproximadamente 25 min, durante los cuales la batería permaneció en reposo. Estos intervalos de reposo permiten la la actualización del voltaje en OCV, que es utilizado por el algoritmo para recalibrar la estimación del SOC y, consecuentemente, del SOH.

La batería utilizada en el ensayo se encontraba en condición prácticamente nueva, por lo que se esperaba una estimación de SOH cercana al 100 %. Los resultados obtenidos muestran que la estimación del SOH se mantiene estable y consistente a lo largo del ensayo. En la Figura 6.5 se presenta, en la gráfica superior, la evolución temporal de la estimación del SOH, donde se observa un valor cercano al nominal con pequeñas variaciones del orden de 0,5 %.

En la gráfica inferior de la misma figura se muestra el perfil de corriente de la batería durante el ensayo, donde se identifican claramente los intervalos de descarga y los períodos de reposo de 25 min, durante los cuales se produce la actualización del OCV. La correspondencia temporal entre estos intervalos y las pequeñas varia-

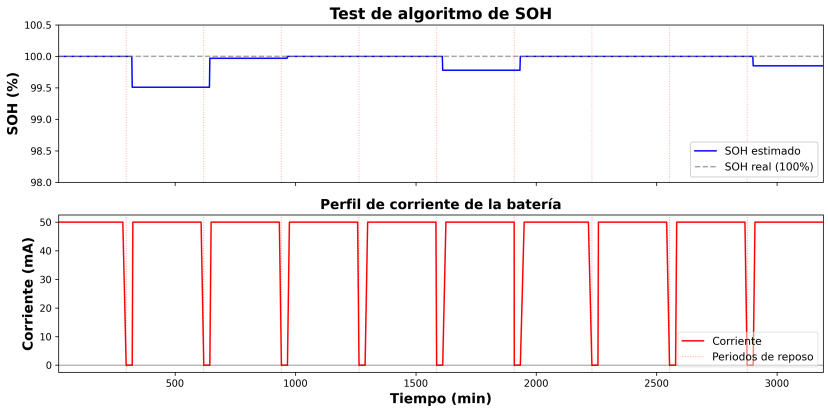


Figura 6.5: Estimación del SOH.

ciones observadas en la estimación del SOH confirma el comportamiento esperado del algoritmo y su correcta integración con el estimador del SOC.

# Capítulo 7

# Conclusiones

## 7.1. Conclusiones generales

El BMS desarrollado en el marco del proyecto cumple con los objetivos y criterios de éxito establecidos en el diseño inicial, el sistema implementado integra correctamente los distintos módulos logrando un funcionamiento estable.

El *firmware* fue diseñado bajo un enfoque modular, permitiendo su adaptación a distintas configuraciones de batería sin requerir modificaciones profundas en el código. Cada módulo opera de manera independiente, coordinado por tareas bajo el sistema operativo en tiempo real, lo que facilita la escalabilidad y el mantenimiento.

El sistema desarrollado es capaz de cumplir con las funciones esenciales de un BMS para vehículos eléctricos:

- Mide las variables eléctricas y térmicas de la batería.
- Establece las protecciones correspondientes ante condiciones de riesgo.
- Realiza balanceo de celdas durante la carga.
- Estima el estado de carga (SOC) en tiempo real.
- Intercambia información relevante con dispositivos externos.

En términos cuantitativos, el sistema logra el siguiente desempeño:

- **Tensión por celda:** apreciación de 0,01 V.
- **Corriente:** apreciación de 5 mA, y capacidad de conducir 20 A en ambos sentidos de forma sostenida.
- **Temperatura:** resolución de 2 °C, en un rango de −40 °C a 125 °C.
- **Frecuencia de muestreo:** 5 mediciones por segundo (5 Hz).

Asimismo, se incorporaron mejoras respecto al plan inicial:

- Se añadieron nuevas protecciones, como las de *temperatura de descarga superior e inferior*, extendiendo la cobertura térmica del sistema.

- Se desarrolló una interfaz gráfica completa para comunicación bidireccional con el usuario, que permite la configuración dinámica de umbrales de protección, parámetros nominales de la batería y monitoreo en tiempo real.

En la Tabla 7.1 se detallan los requerimientos establecidos y su nivel de completitud. Tres requerimientos se cumplen de manera parcial. En primer lugar, los umbrales de sobrecarga y sobredescarga tienen como valor inferior 3,0 V y 2,8 V, respectivamente, debido a limitaciones de la química utilizada y para evitar el daño irreversible de las baterías, por lo que el valor mínimo de 1,0 V no es alcanzado (si bien el dispositivo posee las capacidades). En segundo lugar, al detectar un riesgo el sistema reacciona en el mínimo tiempo posible y no tras un tiempo programado por el usuario, por motivos de seguridad. Por último, la protección contra polaridad inversa se logró de forma parcial, el conector `MAIN_CON` cuenta con una protección física para evitar la polarización inversa, pero el conector `J1` no cuenta con una protección de este tipo. Esta falta de protección para el conector `J1` se debe a la escasez de conectores polarizados mecánicamente capaces de conducir corrientes del orden deseado, pero se incluye entre las posibles mejoras a futuro.

En síntesis, el sistema constituye una plataforma abierta y flexible que cumple con los requisitos técnicos propuestos y sienta una base sólida para su aplicación en actividades de enseñanza e investigación en gestión de baterías de iones de litio.

## 7.2. Trabajos futuros y escalabilidad

Entre las principales líneas de trabajo futuro se destacan las siguientes:

- **Mejora del algoritmo de SOH:** el algoritmo actual, basado en RLS, proporciona una estimación básica de la capacidad efectiva de la batería. Puede mejorarse incorporando modelos más complejos, como el filtrado de Kalman extendido (EKF), métodos basados en modelos eléctricos equivalentes o técnicas de aprendizaje automático que consideren el historial de uso y degradación.

- **Implementación de comunicaciones inalámbricas:** el microcontrolador `ESP32-S3` dispone de interfaces integradas Wi-Fi y Bluetooth que no fueron utilizadas en esta versión. Su habilitación permitiría desarrollar una aplicación móvil o de escritorio para la GUI, con conectividad inalámbrica directa para configuración y monitoreo del BMS.

- **Soporte para número variable de celdas:** actualmente, el firmware está configurado para operar con 16 celdas fijas, se podría adaptar el firmware para detectar y trabajar con un número variable de celdas, ajustando dinámicamente las rutinas de medición, balanceo y protección según la configuración seleccionada.

- **Extensión del módulo de potencia:** se propone desarrollar una placa complementaria basada en el diseño actual, que permita manejar corrientes

superiores a los 20 A especificados para el AutoBMS. Esta placa utilizaría las mismas señales de control del sistema principal para accionar etapas de potencia externas, posibilitando su empleo en aplicaciones de mayor corriente o potencia sin modificar la arquitectura del BMS. La implementación podría realizarse mediante el uso de transistores o contactores de mayor capacidad, controlados por las salidas de `gate` existentes, y con una topología térmica adecuada para la disipación de calor.

- **Implementación de un convertidor de nivel lógico para CAN:** el bus CAN presente en la mayoría de los vehículos opera con niveles de señal de 12 V, mientras que el microcontrolador lo hace a 3,3 V. La incorporación de un convertidor de nivel lógico dedicado permitiría una interfase compatible con distintos entornos automotrices e industriales.

- **Configuración de un pin de alerta:** el sensor de corriente ofrece la posibilidad de definir limites para corriente y tensión total de la batería y generar un aviso de alerta en su salida si no se cumplen. En esta versión del dispositivo no se usa esta función, pero se puede conectar esta salida a un GPIO y configurarlo como fuente de *wake up*, de modo que el MCU pueda salir del modo *light-sleep* ante eventos críticos. Esto reduciría el tiempo de respuesta frente a condiciones de protección.

- **Reemplazo del conector de batería `J1`:** se recomienda sustituir el conector actual por uno que incluya una guía física o polarización mecánica que impida su conexión invertida, previniendo daños en el sistema ante errores de manipulación.

- **Integración de registro de datos:** incorporar almacenamiento en memoria externa o en la nube para registrar variables de operación y ciclos de carga/descarga, facilitando el análisis estadístico y el estudio de degradación a largo plazo.

En resumen, el sistema desarrollado cumple los objetivos académicos y técnicos propuestos, proporcionando una plataforma flexible y extensible para la investigación y enseñanza en BMS. Las mejoras futuras apuntan a expandir sus capacidades hacia una mayor autonomía, conectividad y precisión en la estimación del estado de las celdas, manteniendo el enfoque de diseño abierto que caracteriza al proyecto AutoBMS.

| Categoría | Requerimientos Detallados | Alcanzado |
|---|---|---|
| **Medición de parámetros** | Medir la tensión de cada celda con una precisión mínima de ±0,01 V. | Sí |
| | Medir la corriente de carga y descarga en un rango de al menos 20 A, con precisión mínima de ±0,2 A. | Sí |
| | Monitorear la temperatura de la batería en al menos dos puntos, con un rango de −20 °C a 85 °C y precisión de ±2 °C. | Sí |
| | Relevar las medidas de tensión, corriente y temperatura con una frecuencia mínima de 3 Hz. | Sí |
| **Protección de la batería** | Implementar protecciones contra sobrecarga, sobredescarga, sobrecorriente, sobrecalentamiento, cortocircuito y polaridad inversa. | Parcial |
| | Desconectar la batería de fuentes o cargas externas si se detecta alguna de las condiciones anteriores, tras un tiempo configurable por el usuario. | Parcial |
| | Permitir ajuste de umbrales de protección contra sobrecarga y sobredescarga entre 1,0 V y 4,4 V. | Parcial |
| | Permitir ajuste del umbral de sobrecorriente entre 0,5 A y 20 A. | Sí |
| **Balanceo de celdas** | Implementar un mecanismo de balanceo pasivo de celdas. | Sí |
| **Algoritmo de SOH** | Implementar un algoritmo para el cálculo del SOC con una precisión de ±5 % y del SOH. | Sí |
| **Comunicación** | Incluir interfaces UART y CAN para transmisión y recepción de datos con dispositivos externos. | Sí |
| | Permitir la configuración de todos los parámetros del sistema mediante la conexión UART. | Sí |

Tabla 7.1: Requerimientos detallados y grado de cumplimiento del sistema BMS.

# Apéndice A

# Manual de Usuario

## A.1. Introducción y Alcance

Este manual describe el uso del Sistema de Gestión de Baterías (BMS) del proyecto AutoBMS (Figura A.1), incluyendo controles físicos, puertos, conectores, procedimientos de operación y señales de alerta. Su objetivo es facilitar la puesta en marcha y la operación del dispositivo, la interacción con la interfaz gráfica (GUI) y la actualización de *firmware* del microcontrolador.

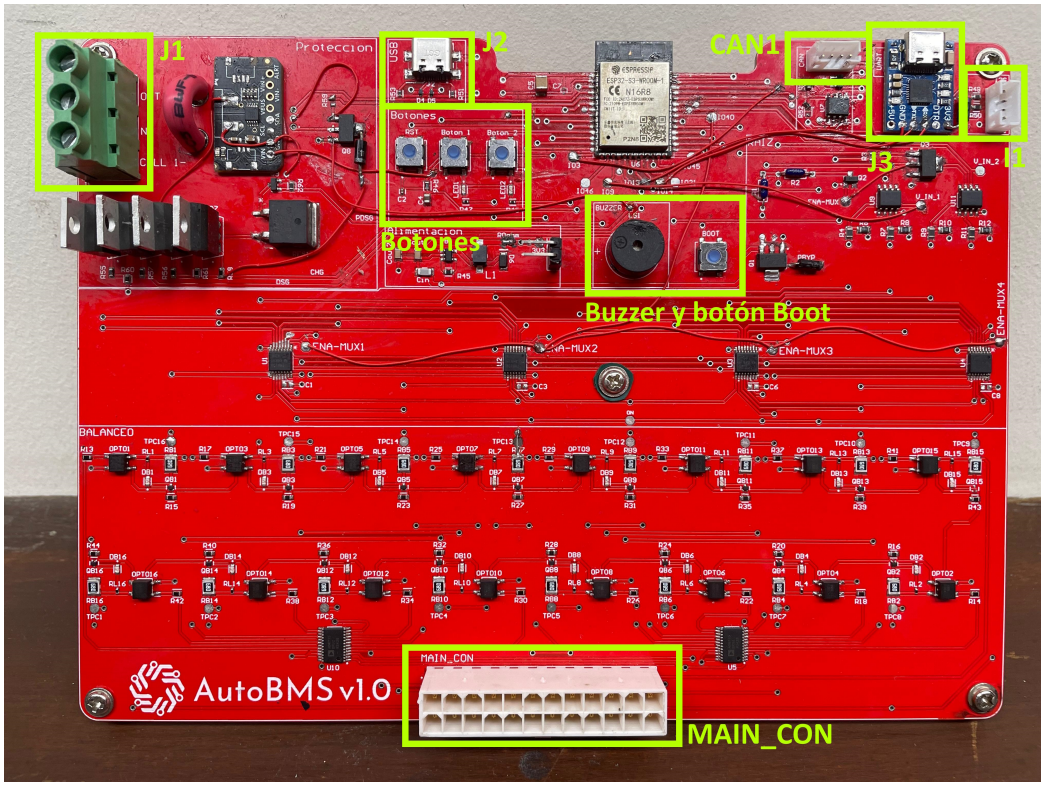


Figura A.1: Sistema de Gestión de Baterías AutoBMS, primera versión.

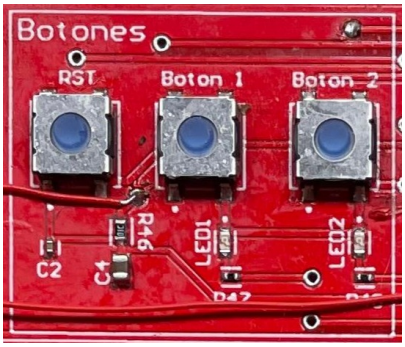


Figura A.2: Zona Botones del BMS.

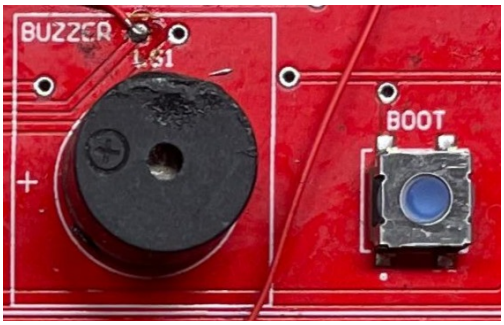


Figura A.3: *Buzzer* y botón *boot*.

## A.2. Vista General del Sistema

### A.2.1. Controles e Indicadores

En la zona Botones de la placa (ver Figura A.2) se encuentran:

- **Botón 1 – Corte General:** Inicia o sale del corte general. Cada vez que se presiona, el *buzzer* emite un pulso corto.
- **Botón 2 – Modo de Configuración:** Entra o sale del modo de configuración. Cada vez que se presiona, el *buzzer* emite un pulso corto.
- **Botón Reset:** Reinicia el microcontrolador.
- **LEDs 1 y 2:** Utilizados para indicar protecciones activas.

Además, a la derecha del *buzzer* se encuentra el botón `Boot` (ver Figura A.3). Este se utiliza únicamente para colocar al ESP32-S3 en modo descarga, para poder actualizar el *firmware* (Procedimiento explicado en la Subsección A.3.4). En la Figura A.1 se encuentran recuadradas las ubicaciones mencionadas en la placa completa.

### A.2.2. Puertos y Conectores

Los conectores presentes en la placa se encuentran indicados en la Figura A.1. A continuación se detallan sus funciones.

- **Conector principal (`MAIN_CON`):** Interfaz principal entre la placa del BMS y la batería. Conectar para alimentar el BMS.
- **Conector (`J1`):** Destinado al circuito de potencia del BMS.
  Conexión:
  `OUT-`: salida negativa del BMS.
  `CELL1-`: polo negativo de la batería.
- **Conector USB Tipo C (`J2`):** Destinado a la actualización de *firmware*. Requiere que el BMS esté conectado a la batería.

- **Conector USB Tipo C (`J3`):** Destinado a la comunicación con la GUI. Requiere que el BMS esté conectado a la batería.
- **Conector `CAN1`:** Permite la conexión con la interfaz CAN.

## A.3. Procedimientos de operación

### A.3.1. Encendido y Corte General

Al conectar el conector principal, el BMS se enciende automáticamente, iniciando con la conducción habilitada. Si en cualquier momento se requiere interrumpir la conducción de corriente, presione el `Botón 1` hasta oír el *buzzer* para entrar en modo de corte general, en este modo se cortan los caminos de corriente para la descarga y carga de la batería. Presiónelo nuevamente para salir de dicho modo.

### A.3.2. Modo de Configuración

Para alterar los parámetros configurados en el sistema o restablecerlos a los valores de fábrica, es necesario estar en el *Modo Configuración*. Para entrar o salir de este modo, presione el `Botón2`. El BMS emitirá un pulso del *buzzer* al hacerlo. El modo expira a los 60 s; tras enviar alguna configuración desde la GUI, el BMS abandona el modo automáticamente.

### A.3.3. Conexión con la interfaz gráfica

Con el sistema encendido, conecte la computadora al conector USB `J3`. Para más detalles de la conexión, ver la Sección B.3.

### A.3.4. Actualización de Firmware

Para actualizar el *firmware*, utilice el puerto USB-C `J2` y asegúrese de que el sistema esté alimentado por la batería.

#### Secuencia de Actualización

1. Mantener presionado el botón `Boot`.
2. Sin soltar el botón `Boot`, pulsar el botón `Reset` hasta que suene el *buzzer*.
3. Soltar `Reset`.
4. Soltar `Boot`.
5. Ejecutar la actualización de *firmware* desde el PC, utilizando Visual Studio Code con la extensión ESP-IDF instalada.

## A.4. Indicadores e Interpretación de Eventos

En la Tabla A.1 se detalla cómo reconocer la activación de una protección y qué acción debe tomar el usuario para corregir la situación.

| **Indicador** | **Condición detectada** | **Acción recomendada** |
|---|---|---|
| `LED1` encendido y *buzzer* activo continuo por 3 s | Sobrecorriente | Desconectar la carga y revisar posibles cortocircuitos. |
| `LED2` encendido y *buzzer* activo continuo por 3 s | Temperatura fuera del rango (sobre/subtemperatura) | Detener la operación y permitir el enfriamiento o calentamiento del sistema. |
| `LED2` encendido (sin pitido del *buzzer*) | Temperatura en rango de solo descarga (superior o inferior) | Continuar la operación únicamente en modo descarga; evitar conectar el cargador hasta que la temperatura vuelva al rango normal. |
| `LED1` y `LED2` parpadeando, y *buzzer* intermitente (cinco pulsos de 500 ms) | Sobrecarga o sobredescarga | Si se está cargando: estado de carga máximo alcanzado, desconectar el cargador.<br>Si se está descargando: estado de carga mínimo alcanzado, conectar el cargador. |

Tabla A.1: Indicadores de protección y acciones recomendadas.

# Apéndice B

# Uso de la Interfaz Gráfica

La interfaz gráfica AutoBMS es una aplicación que permite monitorear y configurar el sistema.

## B.1. Descripción de la Interfaz

### Pestaña Monitoreo

La pestaña de monitoreo se muestra en la Figura B.1. Permite visualizar el estado del BMS (p.ej., *BMS Encendido*, *UART Desconectado*), de la batería y las protecciones activas.

Además, contiene el menú de selección de puerto para la conexión con el BMS, y un botón que permite dar una señal de arranque al sistema.


Figura B.1: Pestaña Monitoreo de la GUI AutoBMS

### B.1.1. Pestaña Configuración

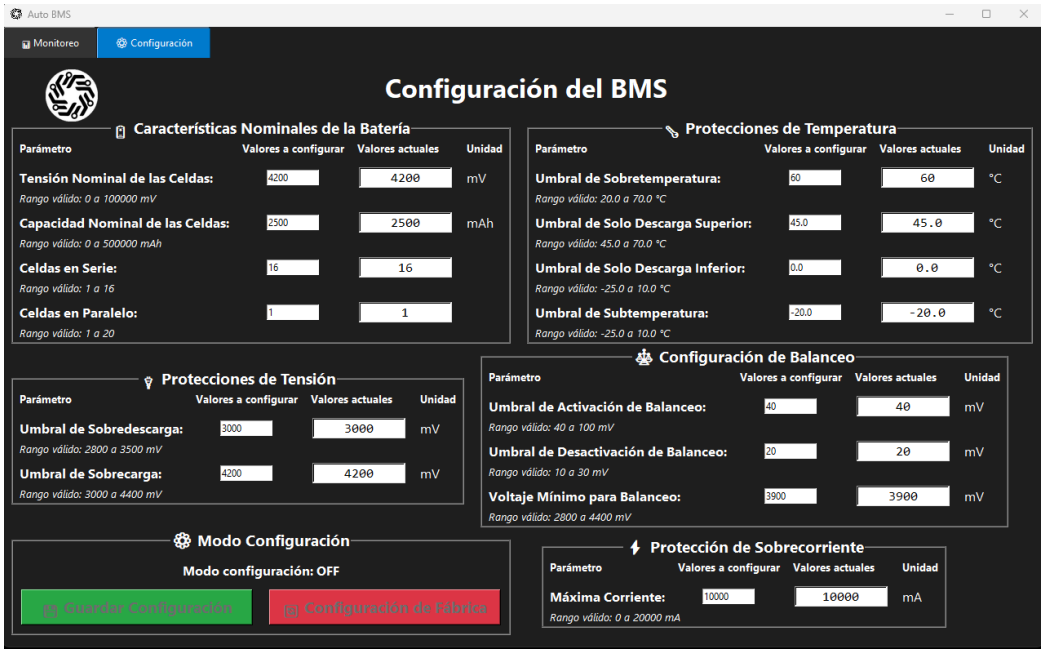


Figura B.2: Pestaña Configuración de la GUI AutoBMS

Esta pestaña se muestra en la Figura B.2. Como su nombre indica, permite configurar los parámetros del sistema, e ingresar las características de la batería a utilizar.

Posee dos botones: `Configuración de Fábrica` y `Guardar Configuración`. Estos están sombreados por defecto, y se habilitan solamente cuando el BMS se encuentra en modo de configuración.

## B.2. Requisitos y Ejecución

**Requisitos:**

- Python 3.9 o superior.
- Librerías: `time`, `serial`, `tkinter`, `threading`, `sys`, `os`.

**Ejecución**

1. Navegar hasta la carpeta que contiene el archivo `GUI_AutoBMS.py`.
2. Ejecutar en un terminal: `python GUI_AutoBMS.py`.

## B.3. Uso Paso a Paso

1. Conecte el BMS al PC.

2. Abra la aplicación.

3. Seleccione el puerto COM correcto y presione *Conectar*. Si no ve su puerto, presione el botón *Actualizar*. Un aviso le indicará cuando la conexión sea exitosa.

4. Para iniciar el BMS, presione *Start BMS*.[1]

5. Verifique que el estado muestre *Conectado* y que se actualicen medidas y protecciones.

6. Para modificar parámetros, vaya a la pestaña *Configuración*. Una vez el BMS se encuentre en *modo configuración*, la aplicación mostrará *Modo configuración: ON*. Ajuste valores y presione *Guardar configuración*.

7. Para restablecer valores predeterminados, use *Configuración de fábrica* (requiere *Modo configuración: ON*).

## B.4. Posibles Mensajes y Advertencias

Durante el uso de la interfaz gráfica pueden mostrarse los siguientes mensajes emergentes. Cada uno indica un estado o evento particular del sistema.

- **UART**: "Conectado exitosamente a COM#". Indica que la comunicación entre la GUI y el BMS se estableció correctamente.

- **UART Desconectado**: *La conexión UART se ha perdido. Por favor, verifica el cable o selecciona un nuevo puerto.* Aparece si el puerto serie se desconecta durante el monitoreo. Se recomienda revisar el cable o re-conectar el dispositivo.

- **Error UART**: *No hay puertos COM disponibles o no se seleccionó ninguno.* Se muestra cuando se intenta conectar sin haber seleccionado un puerto válido.

- **Error UART**: *No se pudo conectar a COM#: ....* Aparece si ocurre un error al abrir el puerto serie (por ejemplo, si está en uso por otra aplicación).

- **UART**: *Puerto UART no disponible.* Indica que no existe conexión activa al intentar iniciar o enviar configuración.

- **Error de Validación**: *Por favor arregle los siguientes errores: ....* Se presenta cuando uno o más parámetros ingresados están fuera del rango permitido o contienen formato inválido.

[1] Si se activa una de las protecciones severas (ver 5.2.2), se debe resetear presionando el `Botón 1` del BMS.

- **Configuración del BMS**: *Configuración Guardada*. Confirma que los valores se validaron correctamente y fueron almacenados localmente antes del envío al BMS.
- **UART**: "Configuración enviada por UART." Indica que la configuración fue transmitida correctamente al BMS por el puerto serie.
- **Error UART**: "No se pudo enviar configuración: ..." Ocurre si se pierde la conexión durante el envío de datos o se detecta un error de transmisión.
- **Error UART**: "No se pudo iniciar BMS: ..." Aparece si la orden de inicio no puede ser enviada por pérdida de conexión o error del puerto.

# Referencias


[1] Ipc-2221a: Generic standard on printed board design, 2003.

[2] SAE International. *Formula SAE® Rules 2019*, 2018.

[3] E. Schaltz & D. I. Stroe A. Gismero. Recursive state of charge and state of health estimation method for lithium-ion batteries based on coulomb counting and open circuit voltage. *Energies*, 2020.

[4] Analog Devices, Inc. *ADG714/ADG715 — CMOS, Low Voltage Serially Controlled, Octal SPST Switches*, 2018.

[5] Battery University. *Charging at High and Low Temperatures*, 2022.

[6] BorgWarner / Sevcon. *Gen4 Size 8 Traction Controller Manual*, 2020.

[7] Broadcom Inc. (Avago Technologies). *ASMT-RF45-AN002 — Surface Mount LED Indicator, Yellow, 0603 (1608 Metric)*, 2013. Datasheet.

[8] Osman Demirci, Sezai Taskin, Erik Schaltz, and Burcu Acar Demirci. Review of battery state estimation methods for electric vehicles – part i: SOC estimation. *Journal of Energy Storage*, 2024.

[9] Analog Devices. *Low Voltage Temperature Sensors TMP35/TMP36/TMP37*, 2015.

[10] Diodes Incorporated. *ZVP2120G — P-Channel Enhancement Mode Vertical DMOS FET*, 10 1995.

[11] Diodes Incorporated. *ZVP2120G — SOT223 P-Channel Enhancement Mode Vertical DMOSFET*, 03 2015. Datasheet.

[12] Diodes Incorporated. *DMN10H220L — N-Channel Enhancement Mode MOSFET*, 2023.

[13] Renesas Electronics. Raa211803 datasheet, 2023.

[14] Espressif Systems. *ESP32-S3 Datasheet*, 2023. Version 1.6.

[15] Espressif Systems. *ESP32-S3 Hardware Design Guidelines*, 2025.

[16] Texas Instruments. Simple balancing of lithium-ion batteries. Technical report, Texas Instruments, 2016. Application Report.

[17] Texas Instruments. *INA228 85-V, 20-Bit, Ultra-Precise Power/Energy/Charge Monitor With I2C Interface*, 2022.

[18] J.S.T. Mfg. Co., Ltd. *PH Series: Wire-to-Board Connectors*, 2024. Datasheet B4B-PH-K-S connector, access date: 2025-10-08.

[19] LG Chem. *Lithium Ion INR18650 MH1 3200mAh — Product Specification*, 2014.

[20] LG Chem. *Lithium Ion LG 18650 HG2 3000mAh — Product Specification (PS-HG2-Rev5)*, 2017.

[21] Littelfuse, IXYS Integrated Circuits Division. *CPC1001N — Optocoupler: Unidirectional Input, Single-Transistor Output*, 2025.

[22] Molex, LLC. *39-29-6248 Micro-Fit 3.0 Dual Row Plug Housing, 24 Circuits*, 2024.

[23] NXP Semiconductors. *BSS123 — N-channel 100 V, 0.17 A logic level MOSFET*, 01 2016. Datasheet.

[24] E. Schaltz & B. Acar Demirci O. Demirci, S. Taskin. Review of battery state estimation methods for electric vehicles-part ii: Soh estimation. *Energy Storage*, 2024.

[25] Ohmite. *35 Watt D2PAK Package Thick Film Power Surface Mount Resistor*, 2022.

[26] G. L. Plett. *Battery Management Systems Volume II Equivalent-Circuit Methods*. Artech House, 2016.

[27] PUI Audio, Inc. *AI-1223-TWT-3V-2-R Magnetic Buzzer Indicator*, 2014.

[28] Samsung SDI. *Samsung INR18650–25R — Product Specification*, 2014.

[29] STMicroelectronics. *STEVAL-LVTRACTION48V — 48 V Low-Voltage Traction Inverter Reference Design*, 2023.

[30] Espressif Systems. *Non-Volatile Storage Library*. Version 5.5.1.

[31] Espressif Systems. *ESP32-S2 Technical Reference Manual*, 2023. Version 1.6.

[32] Espressif Systems. *ESP32 Technical Reference Manual*, 2023. Version 4.9.

[33] Espressif Systems. *ESP32-C6 Technical Reference Manual*, 2024. Version 1.0.

[34] Espressif Systems. *ESP32-S3 Technical Reference Manual*, 2024. Version 1.4.

[35] Texas Instruments. *Design Considerations for Pre-Charge Circuits in High-Voltage Systems*, 2023.

[36] Texas Instruments. *Designing Pre-Charge Circuits for Low-Voltage Systems*, 2024.

[37] Toshiba Electronic Devices & Storage Corporation. *SSM3K35AMFV — MOSFETs Silicon N-Channel MOS*, 2025.

[38] Vishay Siliconix. *SUP85N10-10, SUB85N10-10 — N-Channel 100 V (D-S) 175 °C MOSFET*, 2010.

[39] Würth Elektronik eiSos GmbH & Co. KG. *WR-TBL Series 2184 — Rising Cage Clamp, THT, 7.62 mm pitch (Order Code 691218410003)*, 2024.

Esta página ha sido intencionalmente dejada en blanco.

# Índice de tablas

Esta página ha sido intencionalmente dejada en blanco.

# Índice de figuras

Esta es la última página.
Compilado el viernes 19 diciembre, 2025.
http://iie.fing.edu.uy/