UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

"DISEÑO Y TEST DE UN CARGADOR DE BATERÍAS CON BMS"

TRABAJO FIN DE GRADO

Junio – 2024

AUTOR: Javier Molina Berenguer

DIRECTOR: Higinio Alavés Mañogil

RESUMEN

El objetivo de este trabajo de fin de grado es la creación de un cargador de baterías que cuente con un *Battery Management System* (BMS) adaptado a la carga de baterías de 6 celdas de tipo LiPo. Para ello, se ha llevado a cabo un diseño partiendo de cero basado en las funcionalidades que ofrecen los circuitos integrados seleccionados. Con la creación de este sistema se pretende aprender sobre los métodos y procedimientos que se llevan a cabo durante el diseño y fabricación de una placa de circuitos impresa o PCB: hacer una selección de componentes adecuada, la creación del esquemático, el diseño y las limitaciones existentes a la hora de establecer la ubicación de componentes y vías, y finalmente el proceso de fabricación (soldadura, testeo, reparaciones, etc.). A continuación, se describe el diseño general planteado.

La PCB diseñada se compone de 4 circuitos distintos. La fuente de alimentación utilizada es un cargador con entrada 230V AC y salida 15V DC, la cual alimenta un circuito cargador que adapta el suministro de energía de la entrada según la necesidad de las baterías. Este circuito actúa como conversor DC-DC, alimentándose a una tensión de 15V. Puesto que la configuración de la batería es de 6 celdas, a 4.2V cada una, la salida del conversor se programará para obtener aproximadamente a la salida 25V. También es el encargado de administrar la alimentación del resto de componentes de la placa. Después, un circuito balanceador y otro monitor se complementan conectándose ambos a cada celda individualmente. El circuito balanceador utilizará la información que le aporta el monitor para gestionar las celdas según sea necesario. Por último, el monitor y el balanceador cuentan con pines de salida de comunicación SPI que se conectarán a un microcontrolador (MCU), que a su vez se comunicará por puerto serial (USB) con un ordenador en el que podremos controlar el sistema por medio de una interfaz gráfica creada a partir de la librería “tkinter” de Python.

ÍNDICE

1. INTRODUCCIÓN

1.1 Motivación

La motivación principal de este proyecto es poner en práctica los conocimientos aprendidos en la carrera y conocer mejor el mundo de los circuitos impresos, profundizando en la electrónica de potencia y en particular en los sistemas BMS. Además, un valor añadido de este proyecto es crear un diseño base de un cargador que usa circuitos integrados de interés para la empresa EMXYS, proporcionando un punto de partida para modificar e implementar en el futuro el sistema de carga.

1.2 Estado actual

Un sistema de gestión de baterías o BMS por sus siglas en inglés es un sistema utilizado principalmente en baterías de litio, que se encarga principalmente de monitorear y gestionar la carga de una batería. Estos sistemas desempeñan un papel fundamental en el contexto energético actual, caracterizado por un aumento significativo en la demanda de soluciones energéticas. Sin un BMS, las baterías están expuestas a riesgos como la sobrecarga y la sobredescarga, lo que puede resultar en daños irreparables y una vida útil reducida. Esto implica costes adicionales para reemplazar baterías de forma prematura y un alto impacto medioambiental. Con este sistema obtenemos una alta eficiencia energética, que resulta clave a la hora de aprovechar los recursos energéticos disminuyendo las pérdidas considerablemente. Un BMS también aumenta encarecidamente la seguridad global del sistema, evitando situaciones peligrosas como sobrecalentamiento o cortocircuitos gracias al monitoreo constante de las celdas de la batería.

1.3 Objetivos

- Diseñar un cargador de baterías de litio de 6 celdas con BMS utilizando los circuitos integrados de interés.
- Fabricar, montar y testear la PCB diseñada.
- Desarrollar una interfaz gráfica que permita monitorear e interactuar con el sistema.

2. ESTUDIO PREVIO

Actualmente existen multitud de tecnologías que permiten el almacenamiento de energía. En los últimos años ha crecido exponencialmente el uso de las baterías de ion de litio (Li+) y las de polímero de litio (LiPo) debido a su buen desempeño en múltiples aplicaciones, siendo una de las más importantes en el contexto actual su uso en la fabricación de vehículos eléctricos (EV) e híbridos (HEV) [1]. Sin embargo, un problema típico de estas baterías es que son muy sensibles a la sobrecarga y sobredescarga, y por ello requieren de un sistema de gestión preciso capaz de estimar el estado de carga (SOC) y monitorear-balancear las celdas manteniendo así la salud de la batería [2]. Este sistema se conoce típicamente con el nombre de *battery management system* (BMS). Para entender el diseño de un sistema BMS, se deben tener en cuenta los siguientes conceptos sobre las baterías de litio.

- **Curva de carga**

Es importante conocer los valores de tensión y corriente de nuestra batería a lo largo del proceso de carga para que nuestro sistema BMS realice una gestión adecuada en todo momento. La batería debe cargarse a unos niveles y bajo unas condiciones específicas que el sistema de gestión se encargará de alcanzar y mantener, evitando valores fuera del rango seguro que puedan dañar la batería. La figura 1 muestra la curva de carga típica CCCV, es decir, un ciclo de corriente constante seguido de un ciclo de tensión constante. Esta curva representa una celda de una batería pequeña, pero la técnica utilizada es la misma que la que se aplica en baterías más grandes como las de vehículos eléctricos [3].

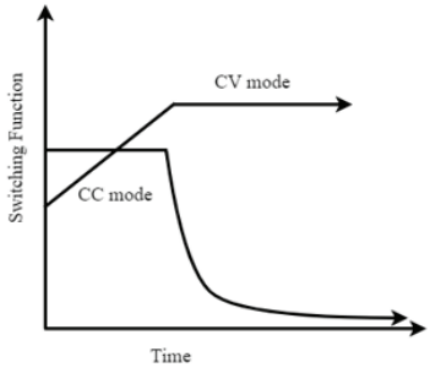


Figura 1: Curva de carga de una batería de iones de litio [3]

- **Curva de descarga**

La curva de descarga nos facilita la relación entre la tensión y la capacidad de descarga de la batería. La capacidad de descarga se mide en Amperios/hora con el parámetro C, representando los amperios que teóricamente la batería es capaz de suministrar durante una hora. La capacidad varía en cada descarga. De esta forma, 0.5C representa una descarga de 2 horas de duración mientras que 2C representa una descarga de media hora [4]. En la figura 2 se muestra un ejemplo de varias curvas de descarga de tensión frente a sus distintas capacidades de descarga para una batería de iones de litio.

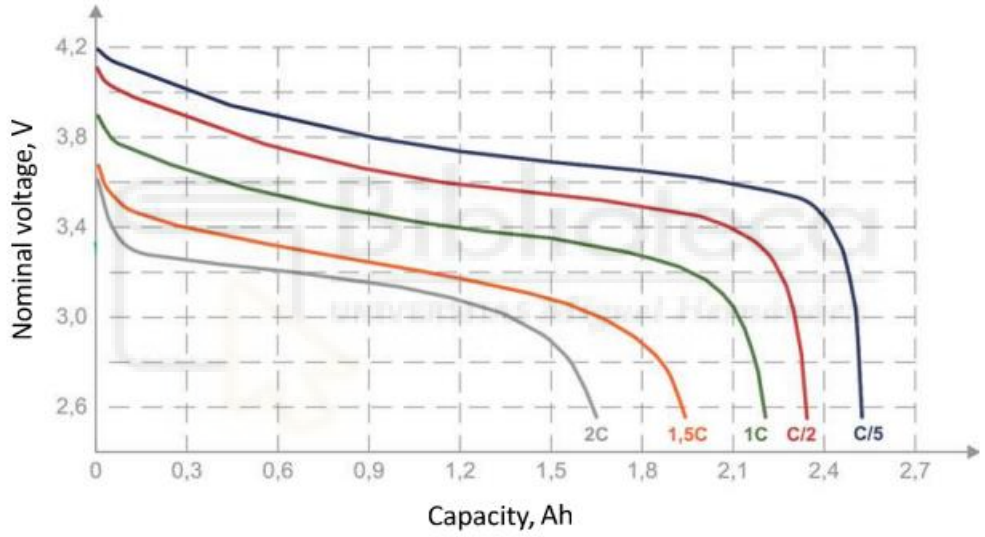


Figura 2: Dependencia de la tensión nominal sobre la capacidad [4]

El fabricante de la batería debe proporcionar la curva de descarga junto a una serie de parámetros como temperatura y corriente de descarga que no debemos sobrepasar para no dañar las celdas. La temperatura es un factor crítico a la hora de analizar una batería, y puede alterar drásticamente el comportamiento de esta [5]. También se debe tener en cuenta que las celdas pueden dañarse de forma irreparable al descargarse completamente, por lo que debemos limitar de alguna forma la descarga para que la batería se encuentre siempre por encima de un valor seguro.

- **Estado de carga (SOC)**

El estado de carga es una medida que indica la cantidad de carga almacenada en la batería en relación con su capacidad total, por lo que $SOC = \frac{C_{disponible}}{C_{total}} * 100\%$ [6]. Se expresa generalmente como un porcentaje y permite conocer cuánta energía puede almacenar la batería durante los procesos de carga y descarga, permitiendo al usuario evitar sobrecargas y sobredescargas [7]. Es un parámetro muy complejo de medir, ya que la relación entre la tensión de la batería y el SOC no es lineal y puede variar con factores como la temperatura, el tiempo que tiene la batería y el historial de carga y descarga.

- **Balanceo de celdas**

En las baterías multicelda, el balanceo es un proceso de igualación de la carga y la descarga de cada celda a nivel individual con el objetivo de mantener todas las celdas a un nivel de tensión similar. Existen dos formas de hacer el balanceo de celdas, con balanceo pasivo o activo. En ambos métodos se monitorea cada celda para mantener la salud de la batería [8]. El balanceo pasivo se basa en la disipación de la energía sobrante en forma de calor de las celdas con tensiones más altas hasta alcanzar un equilibrio con el resto de las celdas. El balanceo activo consiste en dedicar una serie de circuitos y componentes para el control de la batería, de forma que se transfiera energía de las celdas con tensiones más altas a las celdas con tensiones más bajas durante los procesos de carga o sobredescarga.

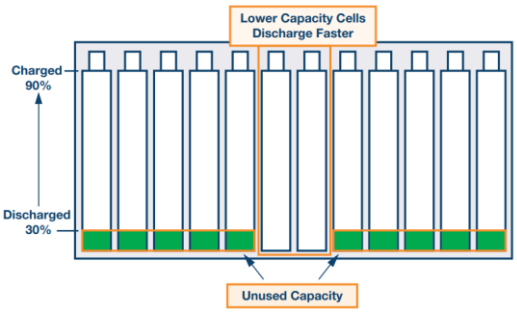


Figura 3: Descarga sin balanceo de celdas [8]

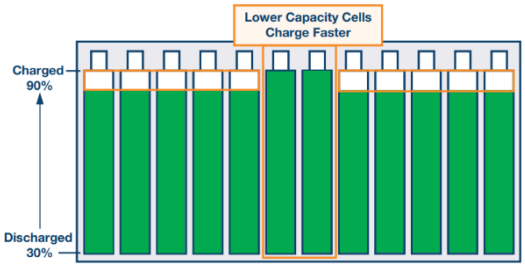


Figura 4: Carga sin balanceo de celdas [8]

3. DISEÑO

3.1 Diseño planteado

El diseño se basará en el integrado responsable del balanceo de celdas escogido, el LTC3300 de Analog Devices, adecuado para el balanceo de baterías de litio. Este integrado cuenta con la ventaja de poder trabajar fácilmente junto al monitor LTC6803, el cual utilizaremos con el fin de monitorear las celdas de la batería. El diseño se ha planteado para baterías tipo LiPo de seis celdas, para una tensión en cada celda de 4.2V, sumando un total de 25.2V. Es posible incrementar el número de celdas, ya que tanto el LTC3300 como el LTC6803 permiten la conexión de integrados del mismo tipo en serie (stack) mediante comunicación serie. A continuación, se muestra el diseño general planteado para el BMS en el diagrama de bloques de la figura 5.

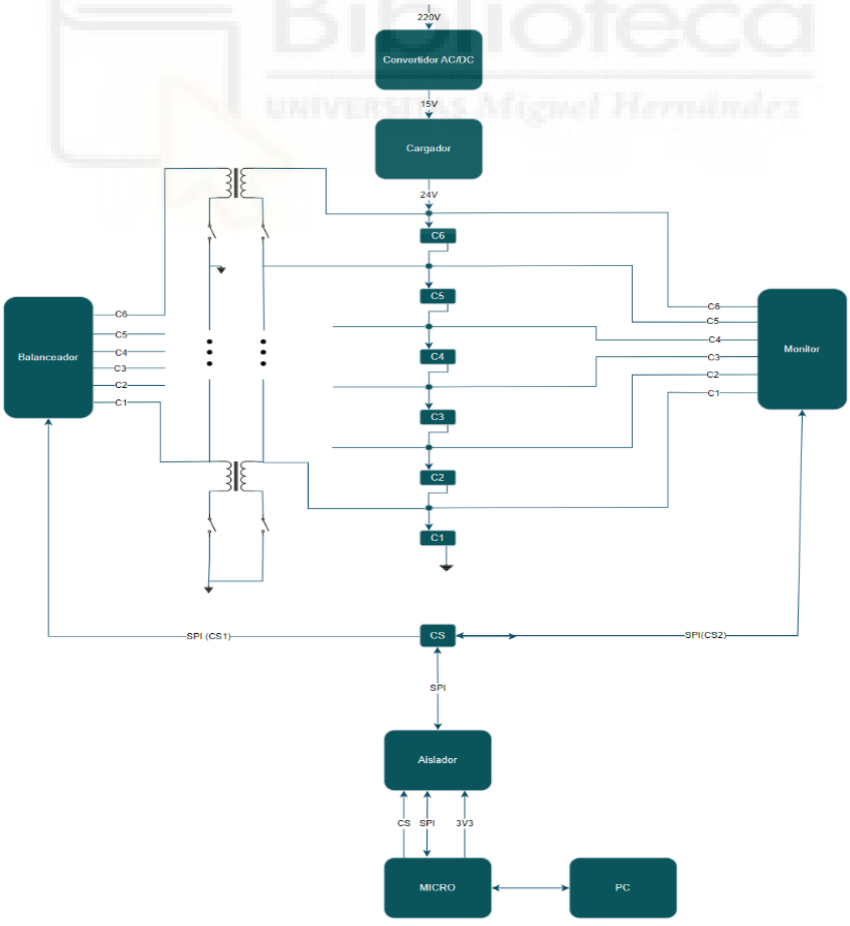


Figura 5: Diagrama de bloques del sistema BMS

En la figura 5 se muestra el esquema del sistema de alimentación y control de una batería de 6 celdas. La entrada de 220V alimenta un convertidor AC/DC que proporciona una salida de 15V. Esta tensión de 15V alimenta la placa. El bloque cargador gestiona la alimentación que recibirá la batería directamente. La serie de bloques verticales etiquetados como *Cn* representa las celdas individuales de la batería, las cuales están conectadas en serie. A los lados de estas celdas se encuentran los bloques balanceador y monitor. El balanceador regula la energía de cada celda para mantener un balance adecuado, y se conecta a los terminales positivos de las celdas a través de seis pines. De manera similar, el bloque monitor también se conecta a los terminales positivos de las celdas y se encarga de monitorear su estado. Ambos bloques, balanceador y monitor, se conectan al bloque Chip Select (CS), que maneja la lógica para la comunicación en serie con cada uno. A través de un aislador, el microcontrolador recibe las lecturas de los integrados por comunicación serie. Finalmente, el control y la monitorización del circuito se realizan desde un PC que se comunica en serie con el microcontrolador.

Teniendo en cuenta los conceptos vistos en el apartado anterior, se busca que nuestro sistema BMS sea capaz de llevar a cabo las funciones de carga completa, descarga y carga de almacenamiento configurable, basándose en un sistema de balanceo de celdas activo proporcionado por el LTC3300 que, junto al LTC6803, se comunicarán vía serie con un microcontrolador. A continuación, se describen los bloques mostrados en la figura 5.

3.2 Cargador

El cargador está alimentado por un convertidor AC/DC que proporciona 15V DC al sistema a partir de 220V AC. Este circuito cargador se basa en el integrado LTC4020, proporcionado también por Analog Devices, el cual permite gestionar la distribución de energía entre la batería y la alimentación gracias a un conversor DC/DC buck-boost, adaptándose a las variaciones en el circuito y a las limitaciones que se puedan dar en la alimentación. Incluye también una serie de protecciones interesantes como un temporizador para detener la carga y proteger la batería. Este integrado permite trabajar en rangos de 4.5V a 55V [9]. En nuestro caso, se busca obtener una tensión aproximada de 25V en la salida del cargador a partir de la entrada de 15V, por lo que se espera que el conversor trabaje en modo boost.

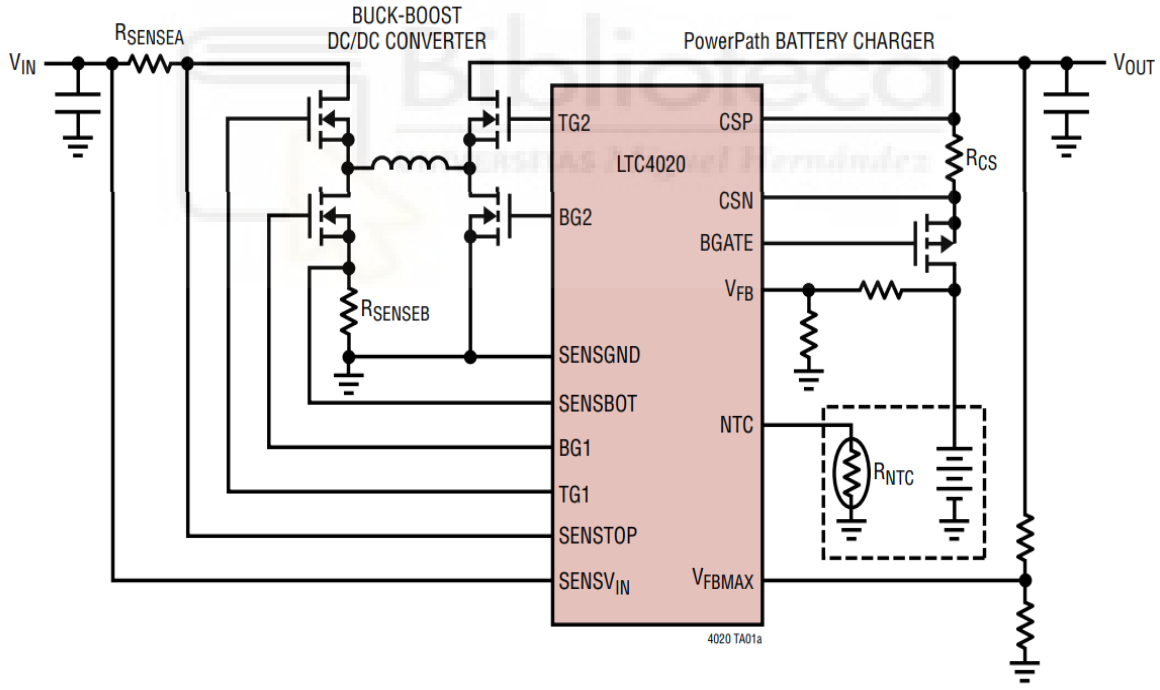


Figura 6: Aplicación típica del LTC4020 [11]

En un ciclo de carga normal, el LTC4020 proporciona una conexión de baja impedancia entre la batería y el conversor DC/DC a través del MOSFET de powerpath como vemos en la figura 6. El integrado controla la conexión de la batería a través del pin BGATE conectado a la puerta de este transistor.

3.3 Balanceador

Como se ha mencionado antes, el bloque balanceador será un circuito basado en el integrado LTC3300-1, el cual permite un balanceo activo multicelda de alta eficiencia energética. Es compatible con baterías de litio y LiFePO4 de hasta 6 celdas, y permite la comunicación SPI con un microcontrolador [10]. Se complementa con el integrado LTC6803-4 para monitorizar las 6 celdas de la batería LiPo. En la figura 7 se puede ver un esquema general de aplicación de este integrado y la posibilidad de conexión con otros integrados del mismo tipo.

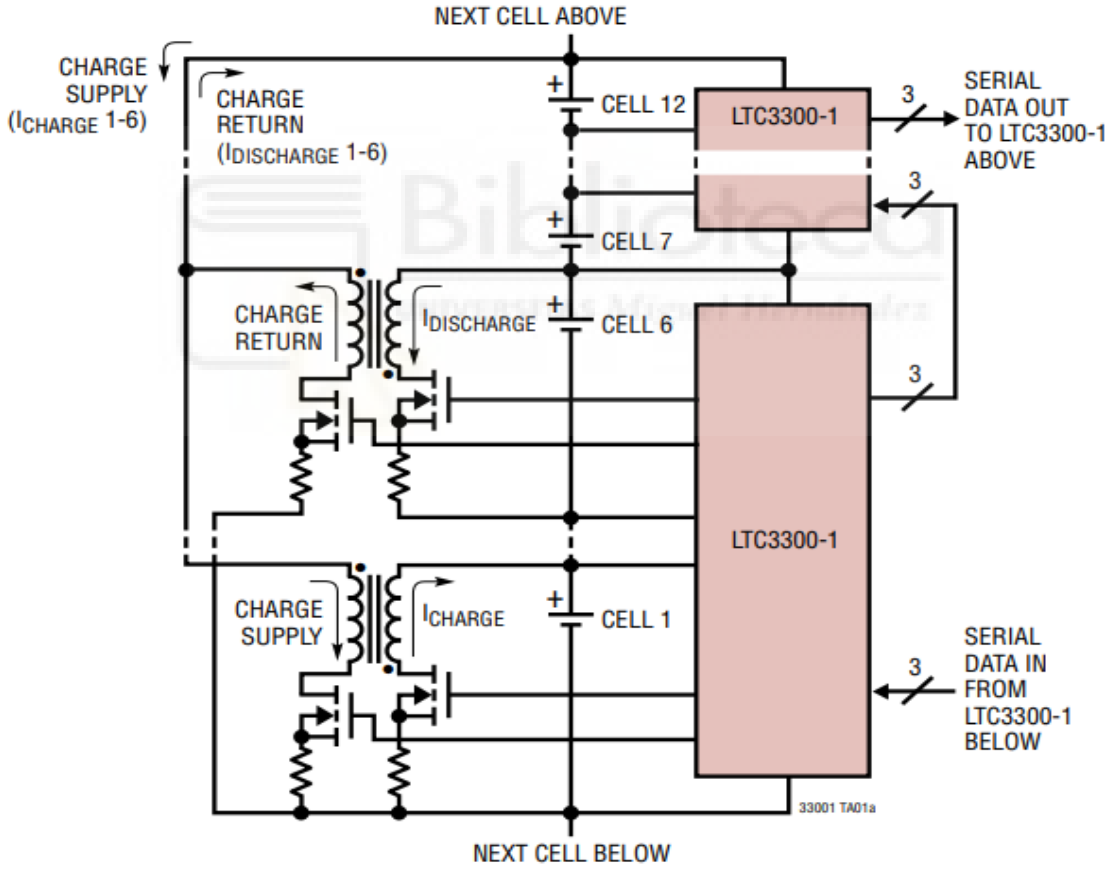


Figura 7: Esquema del LTC3300-1 [9]

3.4 Monitor

Para el circuito monitor contamos con el LTC6803-4, un integrado que permite registrar la información de hasta 12 celdas en serie de una batería. Al igual que el LTC3300-1, este integrado nos permite hacer stacks de hasta 16 integrados del mismo tipo, pudiendo adecuar el sistema a las celdas que se requieran balancear y monitorear. También cuenta con comunicación SPI a 1MHz, que utilizará para enviar la información de la batería al microcontrolador. Además, cada pin de entrada de cada celda cuenta con un transistor MOSFET que hace de switch permitiendo descargar las celdas sobrecargadas aplicando un balanceo pasivo. Esto último no será necesario ya que contamos con un bloque balanceador dedicado a esta función. También cuenta con una función de control de temperatura interna y entradas para termistores. En la figura 8 vemos el esquema general de conexión de este dispositivo.

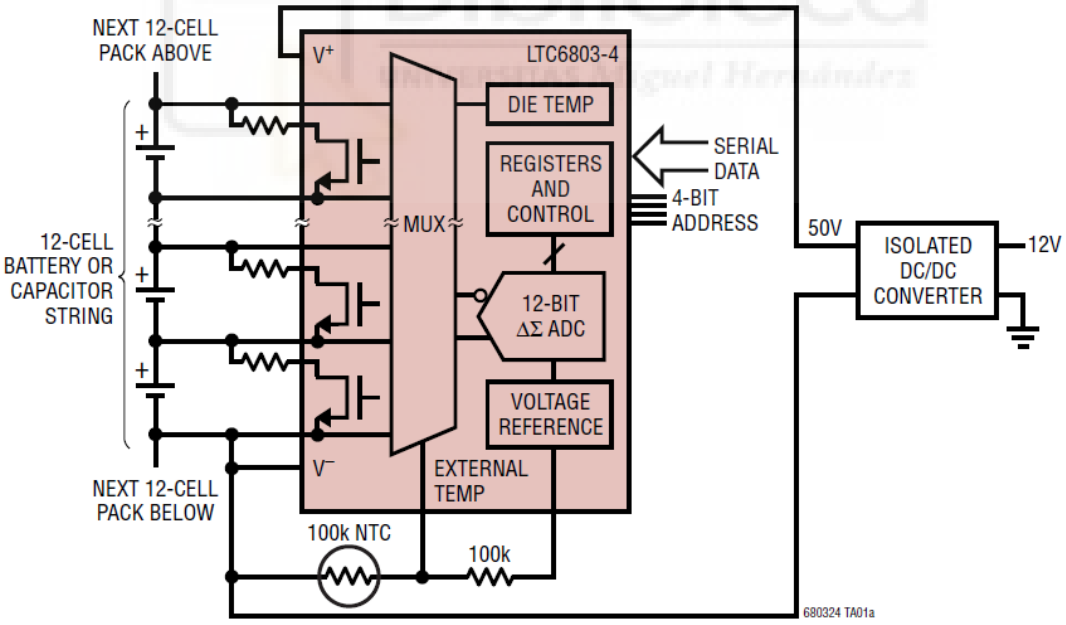


Figura 8: Esquema del LTC6803 [10]

Como se ha mencionado antes, estos dos últimos bloques balanceador y monitor trabajan de forma complementaria, permitiendo una configuración balanceador-monitor como la de la figura 10 para crear una comunicación serie con el microcontrolador gestionada por pines GPIO del LTC6893-4.

3.5 Microprocesador (MPU)

El funcionamiento del balanceador y el monitor será gestionado por un Arduino Nano conectado por SPI. Este módulo integra un microcontrolador que nos permite gestionar la comunicación serie entre un ordenador conectado al Arduino y el bloque que gestiona y monitorea las baterías conectadas a la placa. En la figura 9 podemos ver el *pinout* del Arduino Nano.

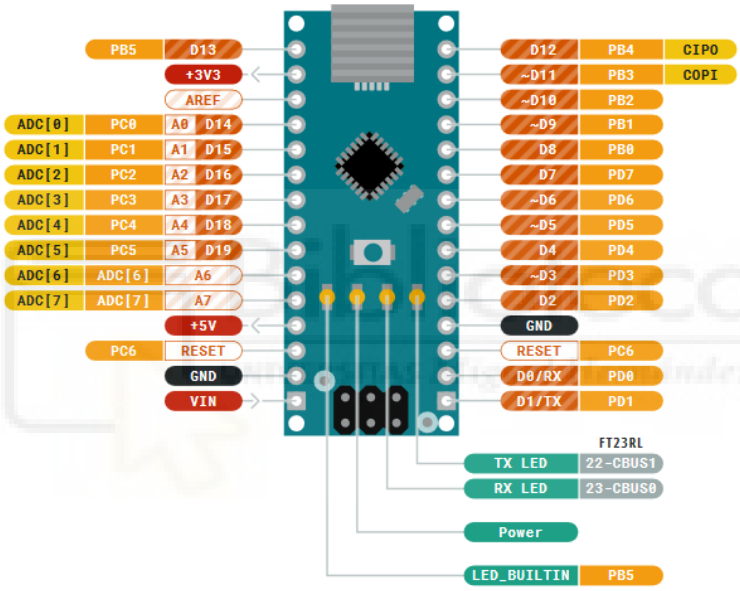


Figura 9: Diagrama de pines del Arduino Nano [13]

Para comunicarse con la PCB se utilizarán únicamente los pines D10, D11, D12 y D13, a las cuáles se asignarán las señales de CS, MOSI, MOSO y CLK respectivamente. Opcionalmente, en el apartado de software se muestra como conectar una pantalla LCD utilizando los pines A0, A1, A2, A3 y A4 para poder visualizar el estado de las celdas. Esta imagen muestra una aplicación típica que combina los integrados descritos anteriormente conectados a las celdas de las baterías y las conexiones SPI que tienen con el microcontrolador.

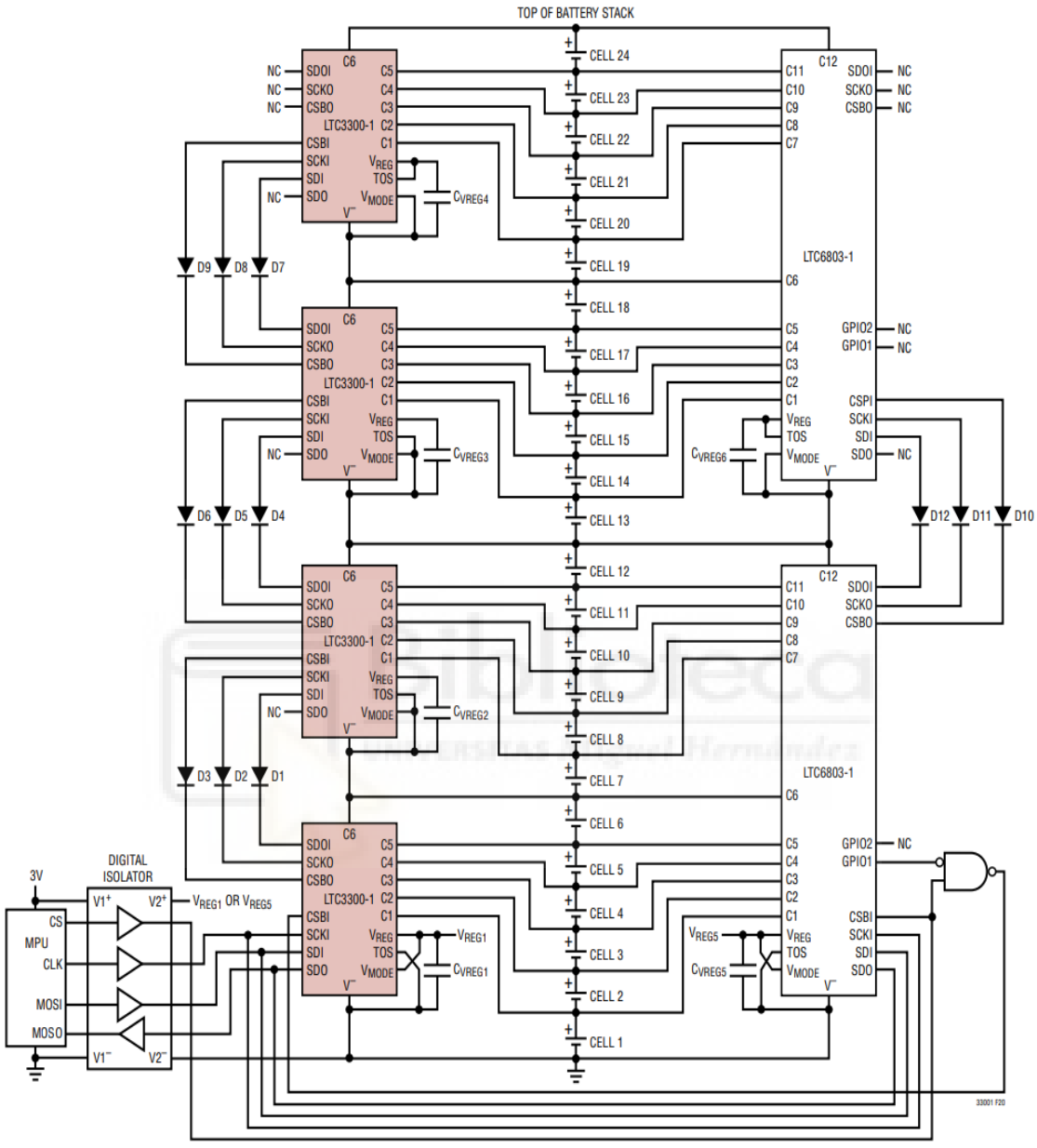


Figura 10: Uso conjunto del LTC3300-1 y el LTC6803 para la comunicación serie [9]

Para evitar dañar los componentes del circuito de control, la comunicación y la alimentación con el microcontrolador se deben separar del resto de la placa, separando la PCB en dos partes: potencia y control. Para ello necesitaremos aislar estos circuitos incluyendo un componente aislador entre ellos, como el que vemos en la figura 10.

3.6 Aislador

El componente aislador seleccionado para el proyecto es el ADUM141E. Este componente encaja perfectamente con nuestro diseño ya que ofrece las entradas y salidas necesarias. En la figura 11 podemos ver el diagrama de inputs/outputs del ADUM141. En total contamos con 3 señales de entrada (CS, CLK y MOSI) y una de salida (MOSO) para monitorear el estado del circuito desde el controlador.

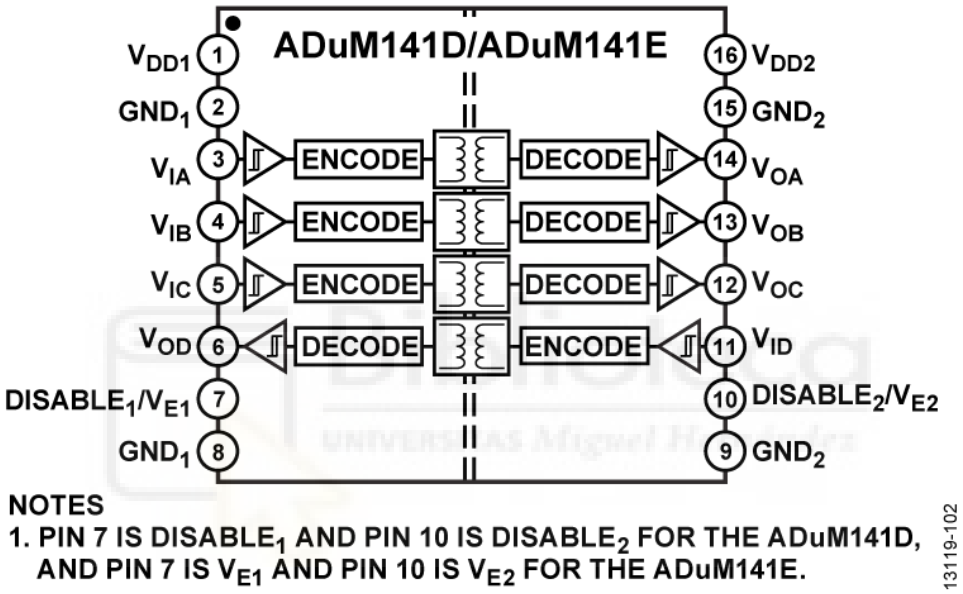


Figura 11: Diagrama del ADUM141E/D [12]

Además, este componente servirá para aislar el ordenador que se conectará al Arduino del resto de componentes de la placa. Las masas de los circuitos de control y potencia deben estar correctamente aisladas entre los pines de entrada y salida de este componente, como se muestra en la figura 12.

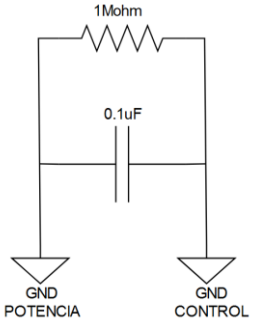


Figura 12: Separación de las masas del circuito

3.7 Esquemático propuesto del sistema

Para el diseño del circuito se ha empleado el programa KiCad, un programa ampliamente utilizado que ofrece una amplia selección de componentes y opciones para diseñar circuitos impresos y generar los archivos necesarios para su fabricación.

En la figura 13 se muestra el esquemático global del circuito. A la izquierda de la imagen están los cabezales de conexión para el microcontrolador. Como se puede ver, consta de las conexiones anteriormente mencionadas, la alimentación 3v3 y la conexión a la masa del circuito de control. En el centro de la imagen se encuentran los bloques de aislador y dos puertas lógicas, una NAND y una NOT, las cuales se encargan de gestionar la señal de *chip select* (CS). También se aprecia en la parte inferior la conexión entre las masas de los circuitos de potencia y de control. Por último, a la derecha de la imagen se encuentran los bloques cargador, balanceador y monitor conectados a las clemas de las celdas de la batería.

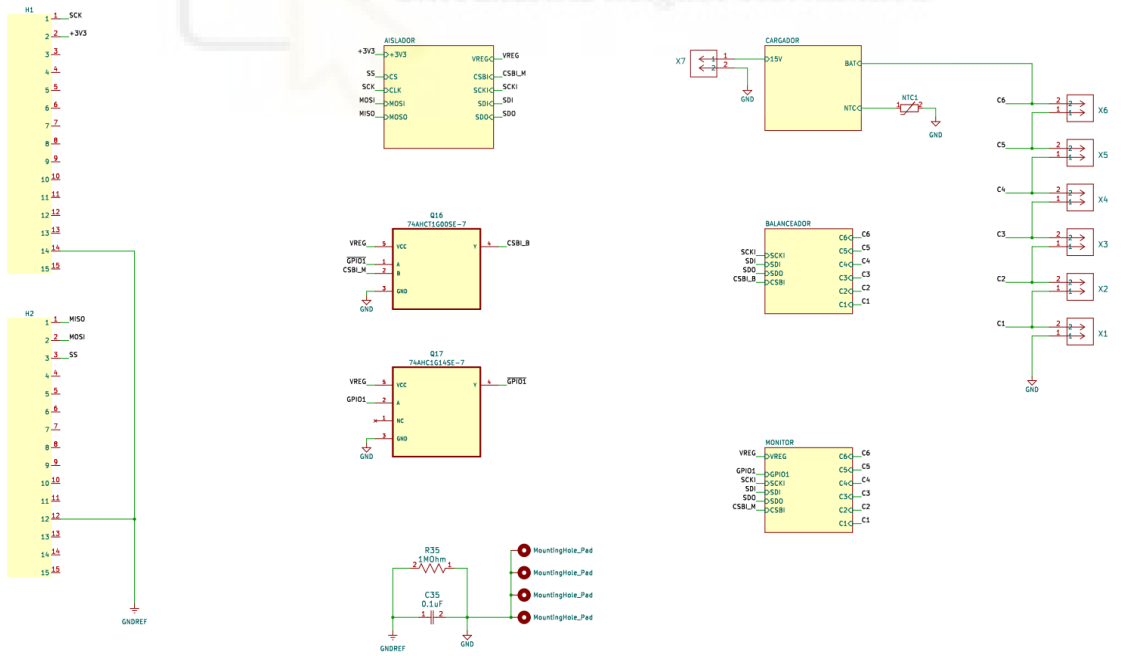


Figura 13: Esquemático global del circuito

- **Cargador**

El diseño del circuito cargador se basa en el circuito visto anteriormente en la figura 4. El objetivo es obtener una salida aproximada de 25V a partir de la alimentación de 15V. Para ello, se han seleccionado los componentes como se describe a continuación: para la selección de componentes pasivos, se han seguido los criterios de diseño establecidos en el apartado de *Application Information* del datasheet [11]. En concreto, los valores de las resistencias R11 y R15 que se utilizan para programar la tensión en la salida se han escogido según la siguiente ecuación:

$$V_{OUT} = 2.75V * \left(1 + \frac{R_{MAX1}}{R_{MAX2}}\right) ; \; 2.75V * \left(1 + \frac{215kOhm}{24.9kOhm}\right) \approx 26.5V$$

Ecuación 1: Cálculo de Vout a partir de Rmax1 y Rmax2 [11]

De esta forma, podemos suministrar hasta 26,5V. Por otro lado, tanto los MOSFET como la bobina se han escogido basándose en las propiedades de componentes propuestos por el fabricante para una aplicación similar [11]. En los transistores, las propiedades que deben tenerse en cuenta son, por una parte, la tensión VDS que deben soportar y, por otra, la resistencia RDS_{on}. También debe considerarse que se cumpla el tiempo de conmutación para que la tensión a la salida sea la adecuada. Además, este diseño cuenta con otro conmutador conectado a la entrada de la batería que permite desacoplarla si fuera necesario.

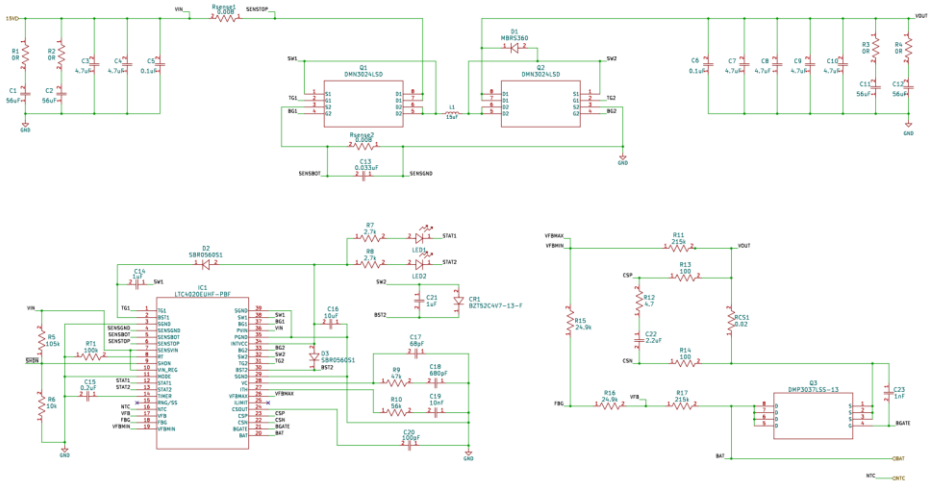


Figura 14: Esquemático del circuito cargador (LTC4020)

- **Balanceador**

Al integrado del circuito balanceador se conectan individualmente las celdas de la batería mediante una serie de conmutadores que conectan las celdas entre ellas y directamente a masa. Esto permite al integrado cargar y descargar las celdas de forma individual y gestionar de forma eficiente cada una, pudiendo transferir la carga sobrante de una celda a otra. Por último, se conectan los pines de conexión SPI que utilizaremos para controlar el circuito desde el microprocesador.

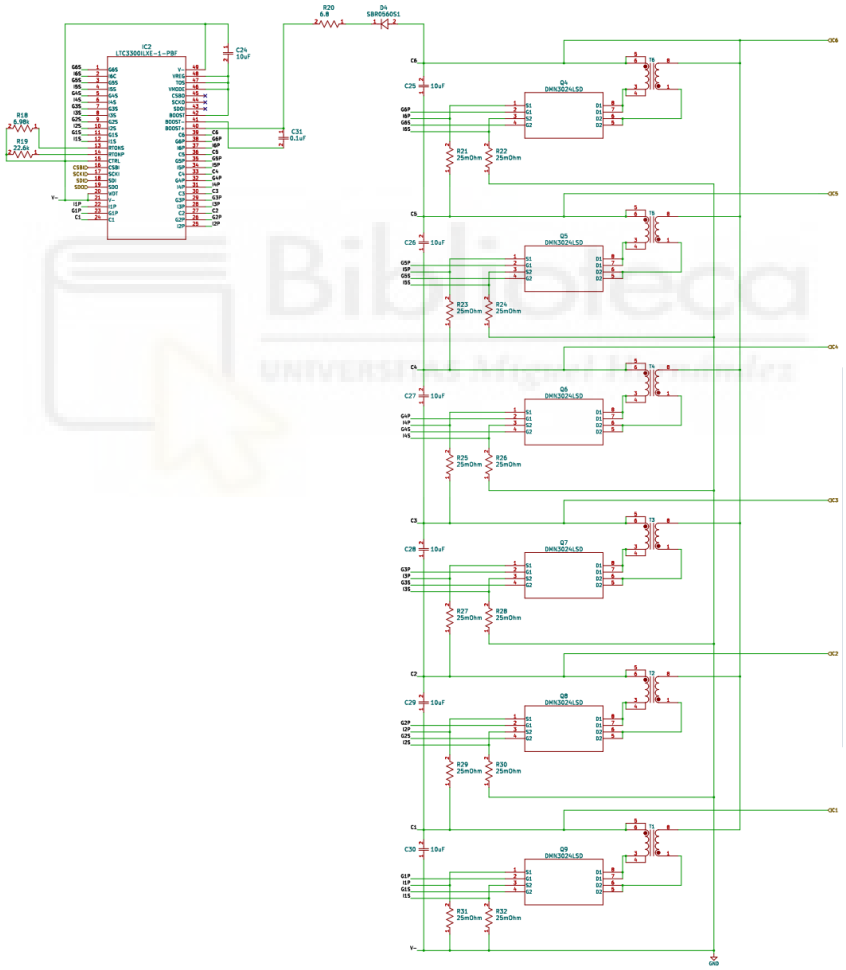


Figura 15: Esquemático del circuito cargador (LTC3300-1)

- **Monitor**

El circuito monitor se puede considerar una versión simplificada del anterior. Este circuito cuenta con pines de conexión individuales que permiten el monitoreo de cada celda. Además, cuenta con unos pines de detección de temperatura a los que conectaremos termistores NTC. Por otro lado, también cuenta con pines de comunicación SPI para enviar la información de las celdas al ordenador. Estos puertos se conectarán a los del balanceador por medio de puertas lógicas, que permitirán seleccionar la comunicación con un integrado u otro según el pin CS y el pin GPIO1 de este integrado.

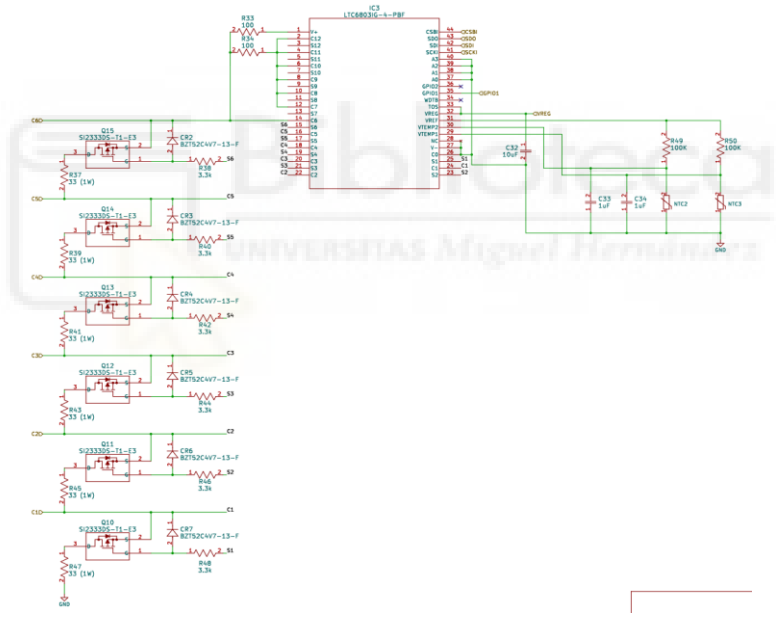


Figura 16: Esquemático del circuito monitor (LTC6803)

- **Aislador**

Finalmente, al aislador conectamos por un lado los pines de comunicación del Arduino y por otro los del balanceador y monitor. Cabe destacar que al aislar este circuito una parte de la placa de otra, cada grupo de puertos debe estar conectada a su masa de referencia correspondiente.

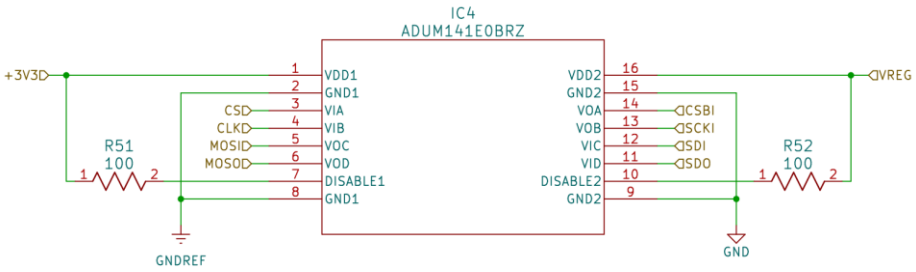


Figura 17: Esquemático del aislador (ADUM141E)

3.8 Diseño de la PCB

Para la distribución de la PCB, se ha seguido un diseño estructurado, separando cada circuito según su función descrita en el apartado anterior. En las figuras 18-21 se puede visualizar la distribución descrita a continuación. En la parte inferior izquierda se encuentra el circuito del cargador. En este circuito encontramos una clema para la conexión de alimentación de 15V. Directamente a la derecha se encuentra el circuito balanceador. Este circuito junto al circuito monitor, que se encuentra en la parte superior derecha, se comunican a través del aislador con el microcontrolador, el cual se posiciona sobre unas tiras de pines con un paso de 2,54mm en la parte inferior derecha de la placa. Además, el circuito balanceador se comunica con las celdas de la batería a través de la hilera de transformadores, mientras que el circuito monitor se conecta directamente a los bornes positivos de las celdas. En la parte superior izquierda se dispone una hilera de clemas que se utilizarán para la conexión de cada celda individual, siendo el pin situado más a la izquierda el que se conectará al borne positivo de la primera celda del conjunto. En la figura 20 se puede apreciar como la pieza 3D proporcionada por el fabricante tiene los pads algo más separados que en el diseño. Esto se debe a que, al recibir la pieza real,

se comparó con el diseño impreso en un folio para corroborar el tamaño de los pads con la realidad y este quedaba algo más pequeño, por lo que se ajustó el tamaño del diseño.

Al tratarse de un circuito de alta densidad, se utiliza tanto la cara *top* como la cara *bottom,* pasando las pistas de una cara a otra a través de vías cuando es necesario, siendo estas de 0,8mm de diámetro y 0,4mm de orificio.

Por último, se puede apreciar la separación de masas entre potencia y control en la parte inferior derecha.

- **Cara top**

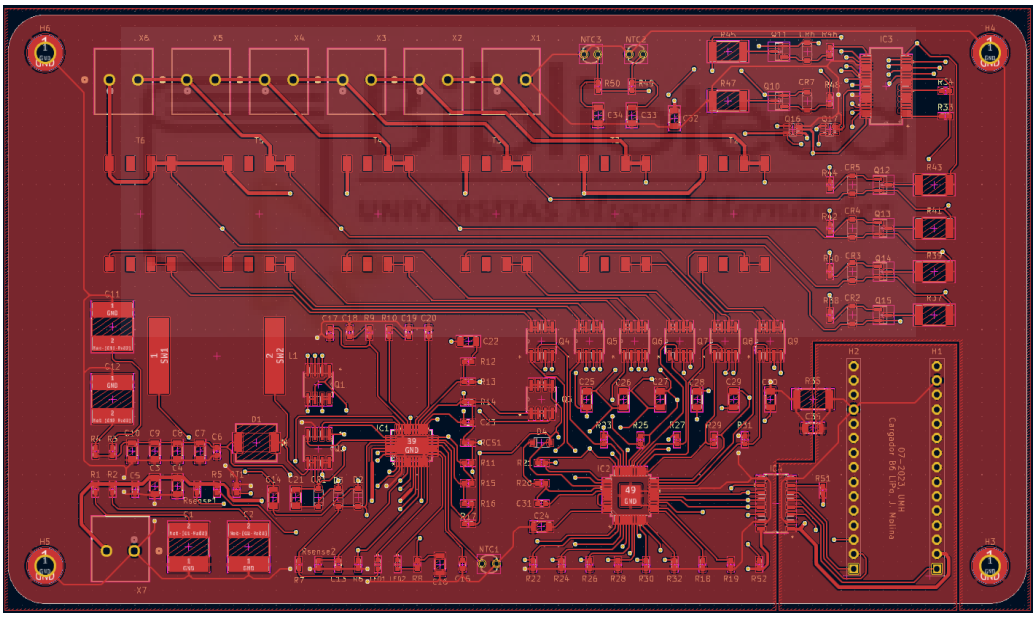


Figura 18: PCB modelo CAD (TOP)

- **Cara bottom**

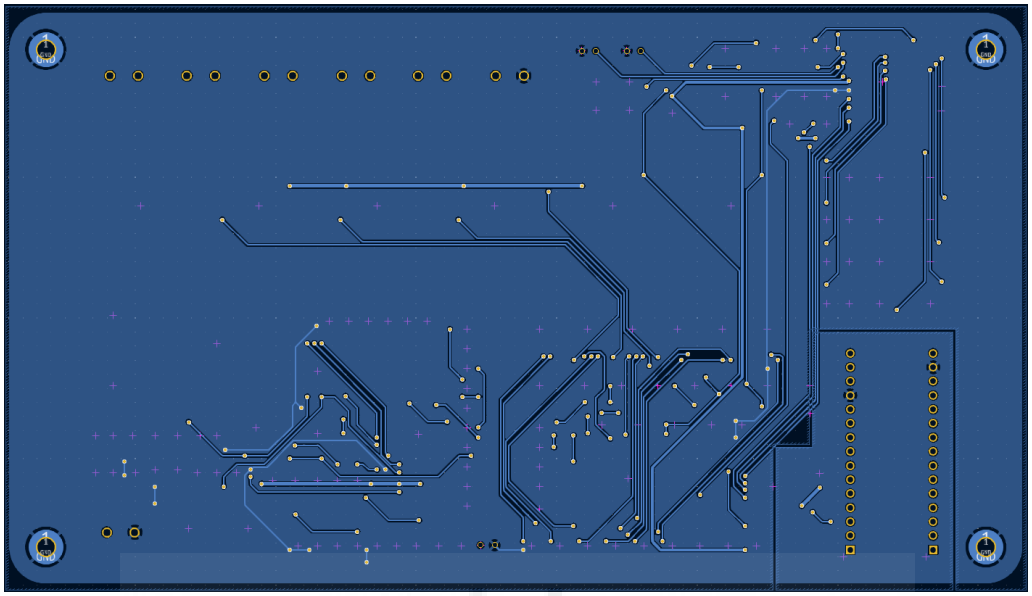


Figura 19: PCB modelo CAD (BOTTOM)

- **Modelo 3D**

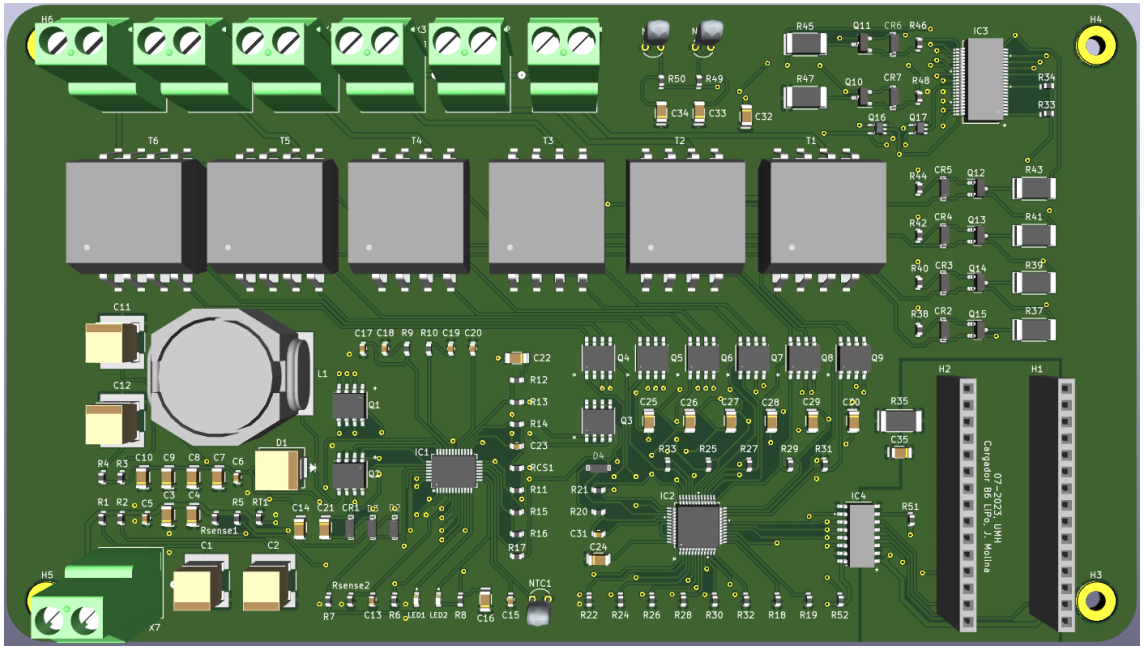


Figura 20: PCB modelo 3D (TOP)

- **Modelo 3D (reverso)**

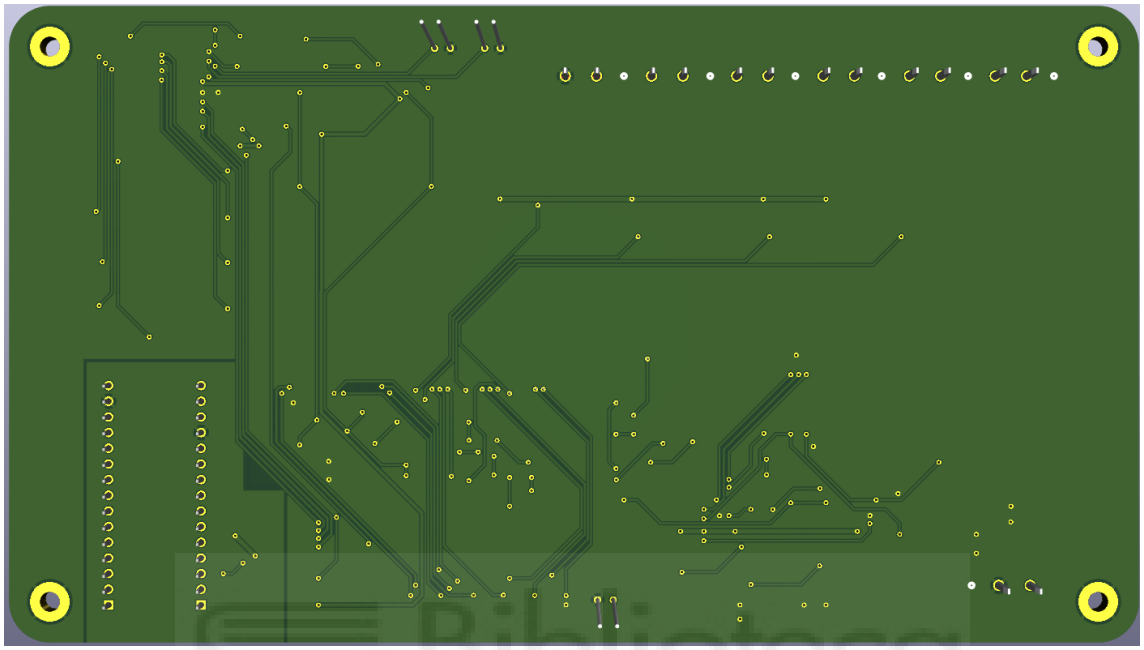


Figura 21: PCB modelo 3D (BOTTOM)

3.9 Consideraciones adicionales al diseño

- **Resistencias**: Todas las resistencias que no disipan mucha potencia son del tipo 0603 con tolerancia del 1%.
- **Diseño de los Pads**: En algunos casos, se ha aumentado el tamaño de los pads para facilitar la soldadura, en concreto en el caso del LTC4020, en el que se han alargado el doble de su longitud con el objetivo de facilitar la soldadura.
- **Conexión de Masas**: Se coloca un condensador y una resistencia en paralelo entre la masa de control y la masa de potencia.
- **Capas del PCB**: La PCB tiene un total de 2 capas, superior e inferior.
- **Vías**: Todas las vías son pasantes para evitar complejidad innecesaria en el diseño del PCB y reducimos costes.
- **Parámetros de diseño**: Tanto el *clearance* como el ancho de pista mínimos se establecen según los circuitos integrados más pequeños, ya que las dimensiones de sus pads serán las más limitantes. Estos valores son: 0,2mm para el *clearance*

y 0,305 para el ancho de las pistas. A las pistas que llevan más corriente como la de alimentación de la placa o de las baterías se les ha aumentado el grosor hasta 0,7mm.

4. DESARROLLO

4.1 Primera fase del desarrollo: simulación del circuito cargador

Una vez planteada la base del diseño, se realiza una simulación preliminar de cada circuito para estudiar su comportamiento. Para ello, se utilizará el programa de simulación LTspice, proporcionado por Analog Devices. LTspice es una herramienta de simulación de circuitos que permite simular una gran variedad de componentes, entre ellos los integrados que hemos seleccionado. Debido a la magnitud del proyecto, el desarrollo se llevará a cabo de manera escalonada, analizando cada circuito por separado para comprobar su correcto funcionamiento. En primer lugar, se realizará la simulación del circuito cargador y, si es favorable, se llevarán a cabo la fabricación y tests del mismo.

El diseño del circuito del bloque del cargador está basado tanto en aplicaciones mostradas en el datasheet del LTC4020 [9] como en la placa de desarrollo que proporciona el fabricante. Este circuito trabaja en modo boost, con una salida programada de 24V en Vout a partir de una entrada de alimentación de 15V y una frecuencia de conmutación programada con R_{RT} de 250kHz [11] página 8.

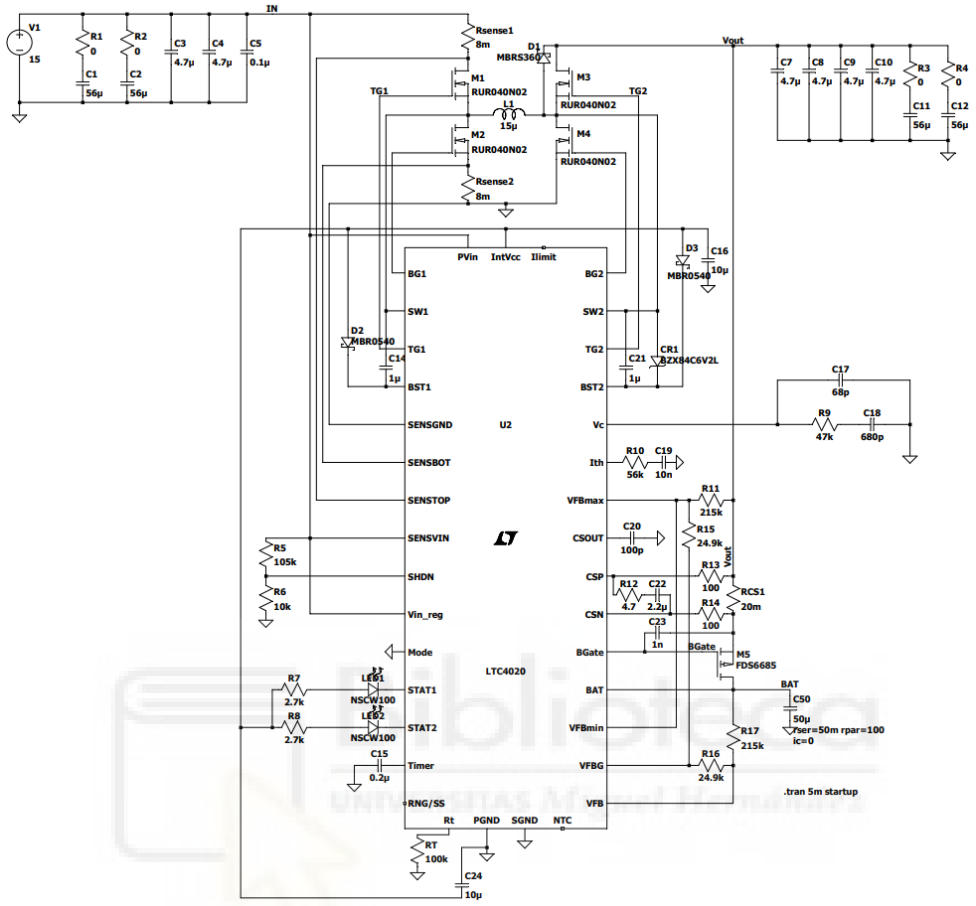


Figura 22: Esquema del circuito del bloque cargador en LTSpice

El funcionamiento de este circuito es el siguiente: la configuración representada en la figura 22 permite trabajar al LTC4020 como conversor boost en modo de carga CC/CV con baterías de litio. Si en el pin VFB se detecta una tensión inferior a 1.75V, el LTC4020 inicia un modo de preacondicionamiento, proporcionando pequeños niveles de corriente para incrementar poco a poco la tensión en el caso de tener una batería muy descargada. En este modo, la corriente que proporciona el circuito se reduce a 1/15 de la corriente [9], que en nuestro caso se ha programado para un máximo de 2.5A por medio de la resistencia R_{cs} de 20mOhm. Esta resistencia limita la corriente máxima de carga y se calcula de la siguiente forma:

$$R_{CS} = 0.05/I_{CSMAX}$$

Ecuación 2: Regulación de la corriente de carga [9]

Una vez el pin VFB alcanza 1.75V, se activa la carga normal de carga a corriente constante (CC) hasta que VFB alcanza 2.5V. Es entonces cuando se inicia la carga a tensión constante (CV) y el valor de la corriente se va reduciendo por debajo de la corriente máxima programada. En la figura 23 podemos ver la curva de carga de la batería junto a la tensión del pin VFB.

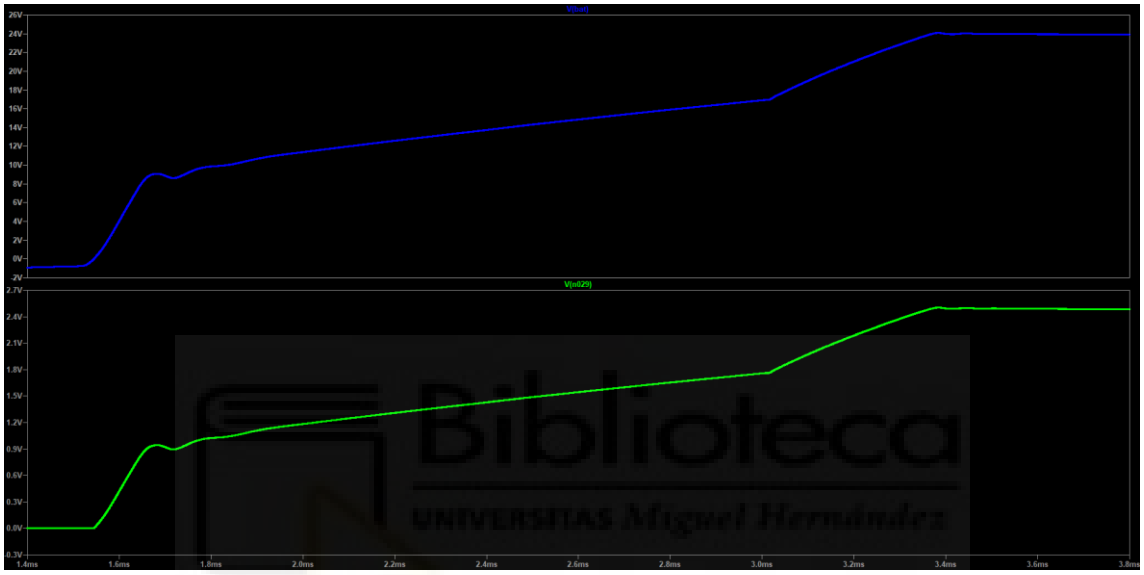


Figura 23: Curva de carga (azul) junto a curva del pin VFB (verde)

El circuito se adapta a los parámetros detectados en la batería para aportar una curva de carga adecuada mediante el LTC4020. En cuanto al conversor buck-boost, en la figura 24 se puede ver la arquitectura típica de un conversor DC/DC, similar al utilizado en nuestro circuito cargador.

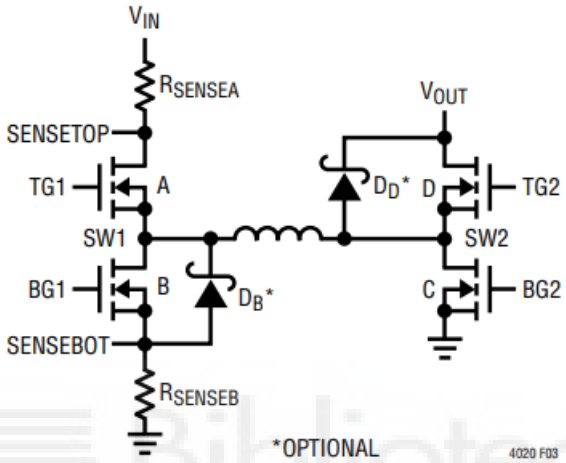


Figura 24: Diagrama del convertidor DC/DC. [9] página 16

Las puertas de estos transistores (TG1, BG1, TG2, BG2) son controladas por los pines del integrado con el mismo nombre. En la configuración boost aplicada en la figura 22, se espera que los transistores correspondientes a la parte de Vout C y D generen la señal PWM requerida para acomodar la conversión DC/DC [9] página 16. Como se ha comentado anteriormente, se ha programado una señal de alta frecuencia a 250kHz (4ps) que corresponde con la velocidad de conmutación de las puertas. En la figura 25 puede apreciarse el resultado de las señales simuladas de las puertas de los transistores C (rojo) y D (verde) superpuestas, correspondientes a los pines BG2 Y TG2 respectivamente.

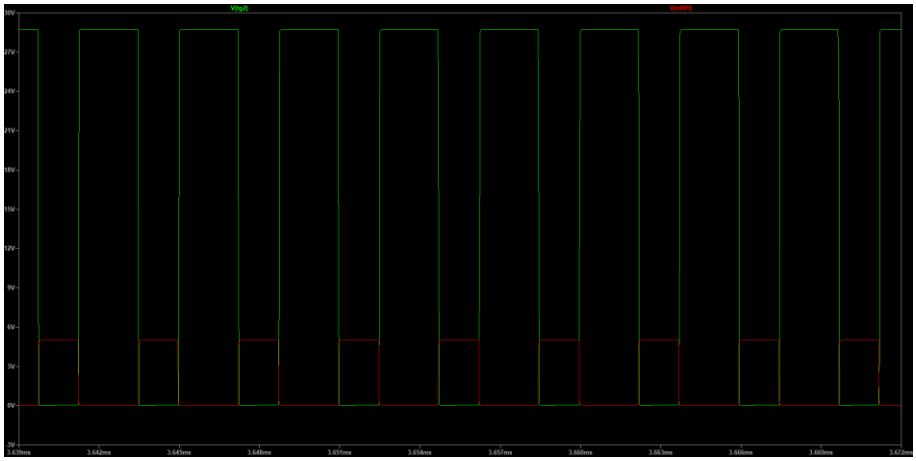


Figura 25: Conmutación de las puertas C (verde) y D (rojo)

La media de estas dos señales nos proporciona la tensión que vemos a la salida de 24V.

Por otro lado, en la parte de Vin, idealmente la puerta A debería conducir de forma continua, mientras que la puerta B no debería conducir [9] pág. 16. En la figura 26 se observa la simulación de la puerta A (verde) y la puerta B (azul). Estos resultados de la simulación se acercan bastante al modelo ideal planteado.

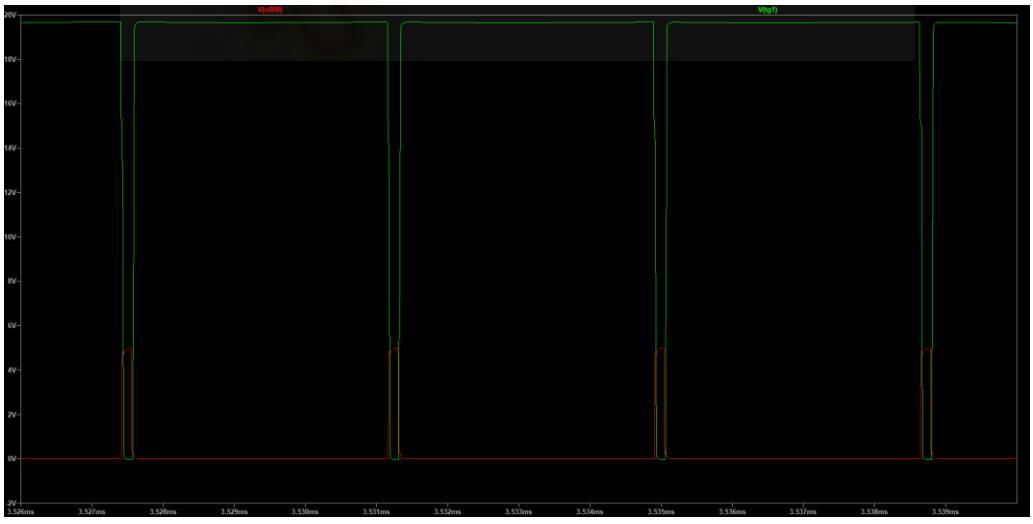


Figura 26: Conmutación de las puertas A y B

Por otro lado, como se ha mencionado antes, el pin BGATE permite controlar el transistor powerpath que conecta la batería con la salida del convertidor, como se observa en la figura 27. Esta función es útil para que el integrado gestione operaciones como el instant-on, detectando la corriente de carga a través de los pines CSP y CSN que miden la

corriente que pasa por R_{CS} y aplicando una curva acorde con el estado de la batería al encender el cargador.

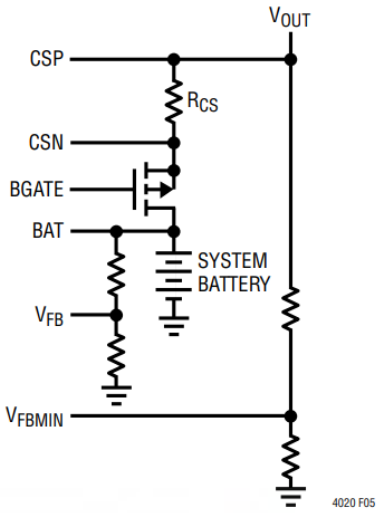


Figura 27: Transistor PowerPath de la batería

CSP BGATE está limitado internamente para que la diferencia de tensión entre estos dos pines se mantenga en 9.5V durante el proceso de carga, ajustándose así a la batería y comenzando el proceso de carga cuando esta tenga suficiente carga almacenada [9]. La tensión del pin CSP es equivalente a la tensión de salida del convertidor DC/DC. En la figura 27 se aprecia el comportamiento del *driver* BGATE (rojo) respecto a la tensión de salida del convertidor (verde).

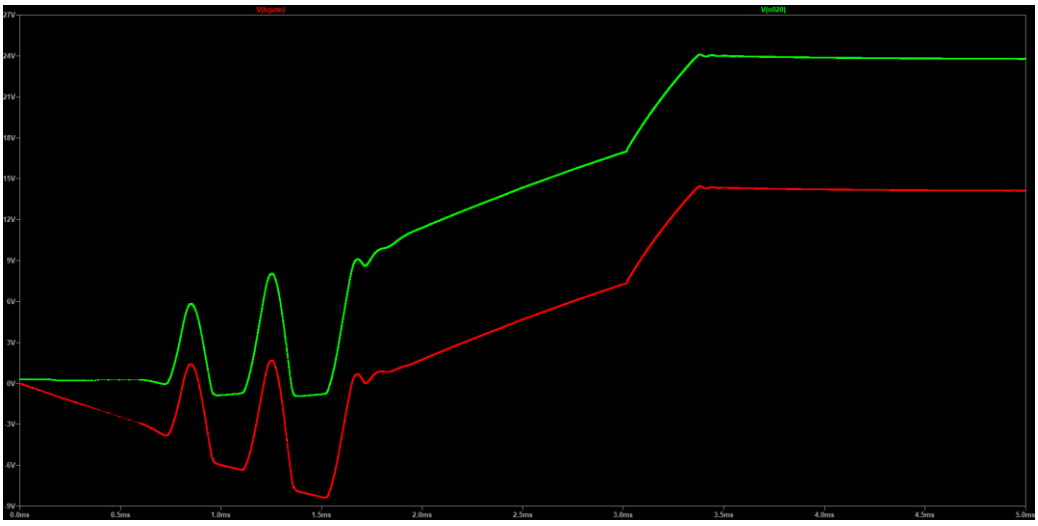


Figura 28: Comportamiento del pin BGATE vs Vout

Con este circuito no solo se consigue obtener la tensión de salida deseada para cargar la batería, sino que también se consigue regular la curva de carga generando unos valores adecuados para la batería y contando con una serie de funcionalidades adicionales que añaden un nivel de seguridad extra al sistema BMS. El LTC4020 nos proporciona más posibilidades aparte de las que se muestran en este diseño, como la opción de temporizar ciclo de carga con el pin *Timer* o medir la temperatura de la batería mediante una NTC. Además, muchos de los parámetros programados pueden modificarse fácilmente para ajustarse a las necesidades del diseño.

Una vez comprobado el correcto funcionamiento del circuito cargador, el siguiente paso será la fabricación y testeo de este, de forma que podamos comprobar su funcionamiento antes de seguir con el resto de los circuitos.

4.2 Segunda fase del desarrollo: fabricación del circuito cargador

En la figura 29 se muestra la placa fabricada para testear los distintos circuitos. En primer lugar, se procederá con el montaje y test del circuito cargador (parte inferior izquierda de la placa).

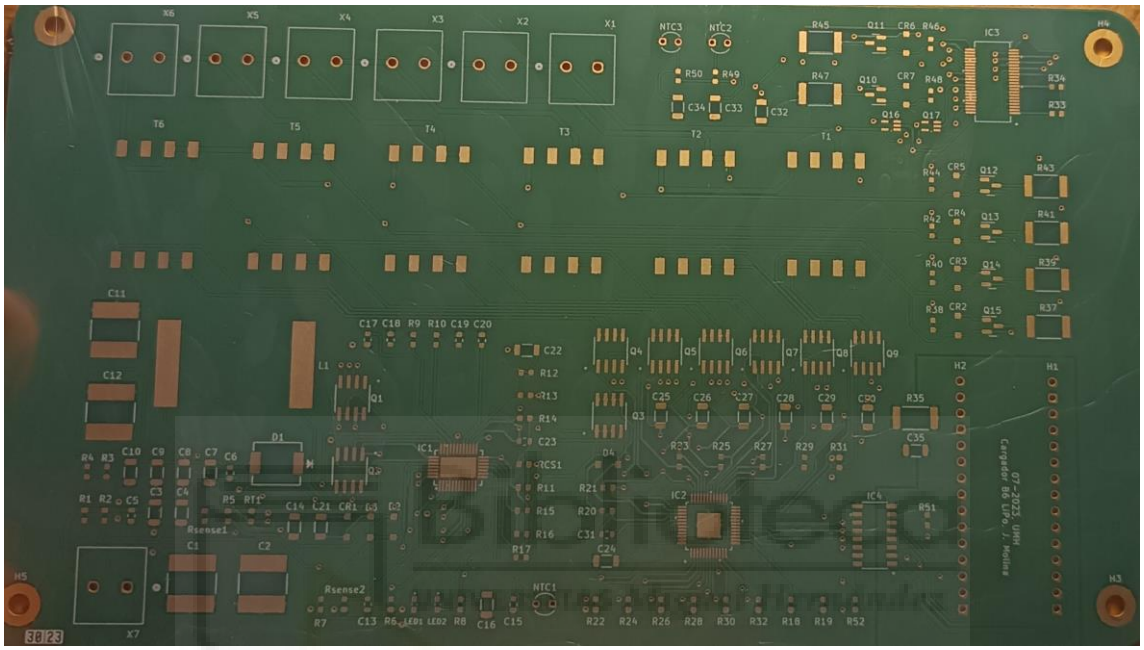


Figura 29: Placa fabricada

A continuación, se muestra una tabla de fabricación del circuito cargador, en la que se detalla cada componente por orden de fecha de montaje incluyendo un breve comentario.

FECHA	REFERENCIA	COMPONENTE	VALOR	NOTAS
14/09/2023	IC1	LTC4020EUHF#PBF		No ha podido ser soldado en el laboratorio de electrónica por el tamaño de la punta de la pistola, se ha soldado en EMXYS.
	D1	MBRS360		NA
	Q1	DMN3024		NA
	Q2	DMN3024		NA
	C17	condensador 68pF	68PF	NA
	C18	condensador 680pF	680pF	NA
	C19	condensador 10nF	10nF	Se ha quedado algo doblado, desoldado y vuelto a poner
	C20	condensador 100pF	100pF	NA
	C23	condensador 1nf	1nF	NA

	C6	condensador 0.1uF	0.1uF	NA
	C5	condensador 1nf	1nF	NA
	Rsense1	resistencia 0.008	0.008R	NA
	CR1	BZT52C4		NA
	D3	SBR0560		NA
	D2	SBR0560		NA
	C13	condensador	0,033uF	NA
	C15	condensador	0,2uF	NA
	C1	condensador	56uF	NA
	C2	condensador	56uF	NA
	C11	condensador	56uF	NA
	C12	condensador	56uF	NA
	L1	bobina	15uH	Los puntos de conexión quedan estrechos
	NTC1	NTC	10kR	NA
	Q3	DMP3037		NA
25/09/2023	C3	condensador	4.7uF	NA
	C4	condensador	4.7uF	NA
	C7	condensador	4.7uF	NA
	C8	condensador	4.7uF	NA
	C9	condensador	4.7uF	NA
	C10	condensador	4.7uF	NA
	C14	condensador	1uF	NA
	C21	condensador	1uF	NA
	C16	condensador	10uF	NA
	C22	condensador	2.2uF	NA
	RT1	resistencia	100kOhm	NA
	R6	resistencia	10kOhm	NA
	R7	resistencia	2.7kOhm	NA
	R8	resistencia	2.7kOhm	NA
	R9	resistencia	47kOhm	NA
	R10	resistencia	56kOHm	NA
	R13	resistencia	100Ohm	NA
	R14	resistencia	100Ohm	NA
	RCS1	resistencia	0.02Ohm	NA
26/092023	R5	resistencia	105kOhm	NA (SE HAN PUESTO RESISTENCIAS DE 100K)
	R15	resistencia	24.9kOhm	NA (SE HAN PUESTO RESISTENCIAS DE 24K)
	R16	resistencia	24.9kOhm	NA (SE HAN PUESTO RESISTENCIAS DE 24K)
	R17	resistencia	215kOhm	NA (SE HAN PUESTO RESISTENCIAS DE 220K)
	R11	resistencia	215kOhm	NA (SE HAN PUESTO RESISTENCIAS DE 220K)

	R12	resistencia	4.7Ohm	NA (SE HAN PUESTO RESISTENCIAS DE 4.3)
	Rsense2	resistencia	0.008	Reparación por huella de 0603
	R1	resistencia	0Ohm	NA
	R2	resistencia	0Ohm	NA
	R3	resistencia	0Ohm	NA
	R4	resistencia	0Ohm	NA
	X7	clema		NA
	X6	clema		NA
	X1	clema		NA
	LED1	led		NA
	LED2	led		NA
27/09/2023	D1	diodo		El diodo se puso al revés en el esquemático y se soldó como tal, se le ha dado la vuelta aplicando calor en ambos lados con dos soldadores
	CR1	diodo		Se ha cambiado por un diodo compatible

4.3 Tercera fase del desarrollo: test del circuito cargador

Una vez terminado el montaje del circuito cargador, se realizó un testeo preliminar del sistema considerando solo el funcionamiento de este circuito. El comportamiento esperado se muestra en el apartado 3.3. El resultado obtenido al hacer las pruebas en el laboratorio fue de 0V en el pin Vout, el cual, como se aprecia en la simulación del apartado 3.3, debe corresponder a una curva que crece hasta la tensión de salida configurada de aproximadamente 24V.

Se procedió entonces al diagnóstico de la placa. Inicialmente se comprobó el estado del cargador indicado en el circuito según el estado de los LEDs 1 y 2. En la figura 30 se puede apreciar en la parte inferior como el estado de los LEDs es el siguiente: LED1=ON, LED2=OFF.

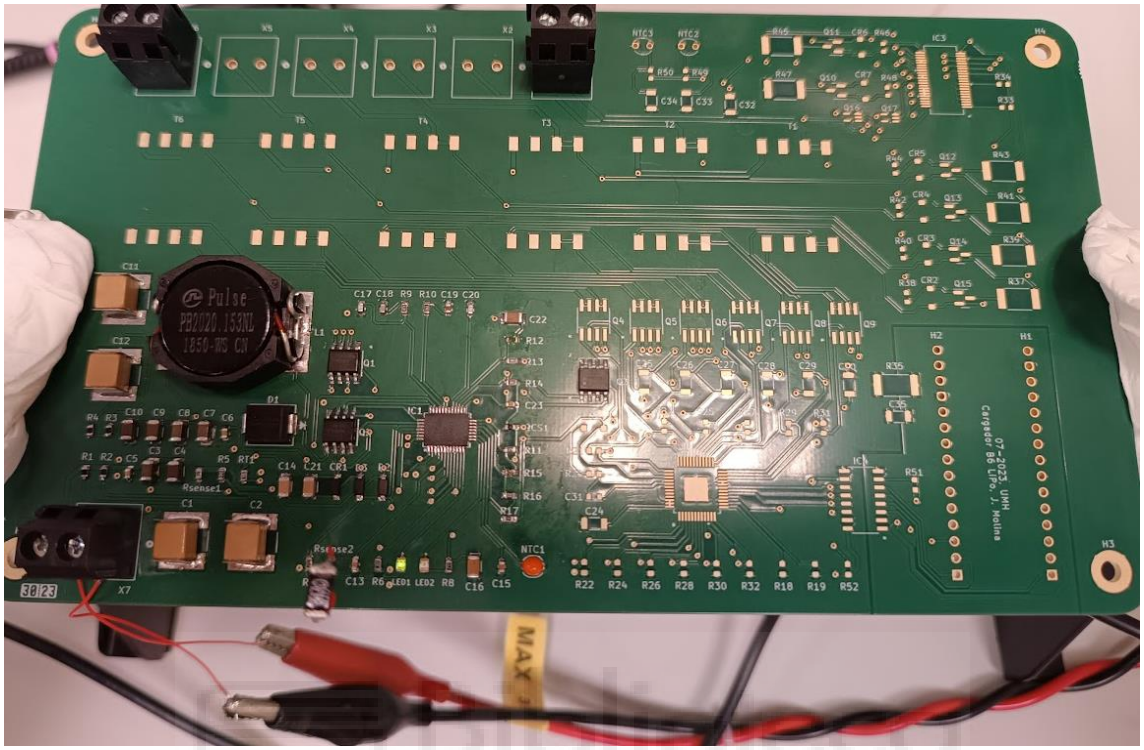


Figura 30: Test del circuito cargador

Como se comprueba en la tabla 4 de [11], en este estado el circuito se encuentra en modo de ciclo de carga correcto, por lo que, deberíamos ver la señal esperada a la salida.

Table 4. Status Pins Charging State

STATUS PINS STATE STAT1	STAT2	CC/CV (MODE = 0V)	LEAD-ACID (MODE = $INTV_{CC}$)	CC (MODE = NC)
OFF	OFF	Not Charging — Standby or Shutdown Mode, I_{CS} < C/10	Not Charging — NTC/Bad Battery Fault or Shutdown	Not Charging — Standby or Shutdown Mode, I_{CS} < C/10
OFF	ON	Bad Battery Fault	Float Charge	Not Used
ON	OFF	Charging Cycle OK: Trickle Charge or I_{CS} > C/10	Absorption Charge	Charge Cycle OK
ON	ON	NTC Fault	Bulk Charge	NTC Fault

Figura 31: Indicador de estado del LTC4020

Estando el integrado en modo de funcionamiento correcto, se procedió a investigar el funcionamiento del resto de componentes del circuito, comenzando con los transistores encargados de generar la tensión de salida. Se determinó entonces que estos transistores, integrados en los componentes Q1 y Q2, no estaban teniendo el comportamiento adecuado. Se analizó la señal en las puertas de los transistores, donde se percibía un pulso mínimo proveniente del reloj interno del LTC4020. Al no comportarse según lo esperado, se reduce el diagnóstico a estos 3 componentes en específico, ya que puede deberse a alguna de estas causas: la programación diseñada para el funcionamiento del integrado, las características de los transistores escogidos o la posibilidad de que alguno de los componentes esté defectuoso.

En el foro de Analog Devices [14] se encuentra un caso similar y una serie de cambios y propuestas de interés sobre el circuito.

Ante el problema encontrado y siguiendo las instrucciones de Analog Devices en el foro, se enumeran a continuación las posibles causas:

1. Defecto en el LTC4020.
2. Defecto en los transistores.
3. Potencia de alimentación insuficiente.
4. Parámetros y componentes en el circuito de alimentación de las puertas de los transistores inadecuados.

A continuación, se listan las soluciones ejecutadas:

- Cambio del integrado LTC4020
- Cambio de los transistores Q1 y Q2 por unos de mayores prestaciones, con tolerancia de hasta $60V_{DS}$, carga en la puerta de 2110 pF y tensión de disparo de 1V.
- Comprobación de la potencia de alimentación: Se comprobó en el laboratorio de electrónica con una fuente de alimentación programable, incrementando la corriente, sin obtener resultados.

Debido a la magnitud del proyecto y a los recursos disponibles, no es posible continuar diagnosticando el funcionamiento del cargador, por lo que queda pendiente implementar el resto de los cambios propuestos y seguir testeando el circuito en líneas futuras de este proyecto.

5. SOFTWARE

5.1 Introducción.

Como se menciona en el apartado de diseño del cargador, el sistema se controlará con un microprocesador que servirá para establecer las comunicaciones entre la placa y un software de control externo. En el esquema de la figura 7 se describen los pines necesarios del circuito integrado del balanceador y del monitor para establecer la comunicación con el microprocesador. El microprocesador seleccionado para este sistema ha sido el Arduino Nano, un dispositivo de gran versatilidad y de pequeño tamaño, adecuado para montar en una PCB. Además, este dispositivo es compatible con los protocolos de comunicación más comunes: UART, I2C y SPI. El programa se comunicará mediante el protocolo SPI (*Serial Peripheral Interface*). Para programar el controlador se utilizará Arduino IDE, un software que utiliza un lenguaje de programación basado en C/C++, el cuál dispone de una gran variedad de librerías, funciones y una amplia comunidad.

Por otra parte, dispondremos de una interfaz de usuario desde la cual se darán las siguientes instrucciones al cargador: carga, descarga, carga de almacenamiento. Para ello, se ha desarrollado en un script de Python una interfaz básica haciendo uso de la librería *tkinter*. El objetivo de esta interfaz es mandar las instrucciones al Arduino Nano por comunicación puente serie que, a su vez, utilizando el protocolo SPI, se comunicará con los circuitos integrados del cargador para establecer el modo deseado.

Por último, podrán consultarse el estado de la batería y el modo de funcionamiento actual en cualquier momento gracias a la integración de una pantalla LCD comunicada con el Arduino Nano.

A continuación, se explica el funcionamiento general del código desarrollado. Para consultar el código completo, leer el Anexo.

5.2 Arduino. Descripción general del código.

En primer lugar, se incluyen las librerías que van a utilizarse. En este programa se utilizan únicamente dos librerías, SPI.h y Adafruit_TFTLDC.h. La librería SPI permite utilizar los métodos para inicializar y programar la comunicación serial. La librería TFTLCD proporciona una serie de funciones que facilitan en gran medida la programación de la pantalla LCD.

A continuación, definimos los pines que se van a utilizar. El Arduino Uno permite el uso de los pines digitales para distintos propósitos. En este proyecto se utilizarán los pines 10, 11, 12, 13 para la comunicación SPI con los circuitos integrados de la placa. También definiremos los pines analógicos A0, A1, A2, A3 y A4 para comunicarnos con la pantalla LCD.

A continuación, se definen una seria de variables globales, inicializando el modo de la batería en “Idle” y la comunicación con la pantalla.

En el cuerpo central del programa, encontramos las funciones de “setup” y “loop”. En la función “setup” se inicializa la comunicación serial con el baud rate de 9600. Después, se inicializan los pines digitales definiendo su modo de operación (input, output) y asigna el estado lógico alto al pin CS, el cual será el encargado de seleccionar el IC con el que se establecerá la comunicación serial. Se inicializa la librería de la pantalla LCD y se llama a la función “batScreen”, la cual genera el contenido inicial de la pantalla. Por último, la función “loop” es una función bucle que comprueba el estado del puerto serial para ver si se ha establecido comunicación desde la interfaz del PC. Si se detecta comunicación, el dato recibido se compara con un algoritmo de tipo “switch” en el que según la opción recibida se establecerá el modo de funcionamiento del cargador, enviando al circuito balanceador el byte adecuado. Por otra parte, el Arduino estará constantemente leyendo la información recibida en el puerto serial de entrada (MISO) y actualizando la pantalla LCD con la información recibida sobre el estado de la batería.

La pantalla utilizada para visualizar el estado de la batería se muestra en la figura 25. En la misma, podemos apreciar los nombres de los pines. En el caso de este proyecto, únicamente se utilizarán los siguientes pines: RST, CD, RS, WR y PD, además del pin de alimentación de 5V y la masa GND.

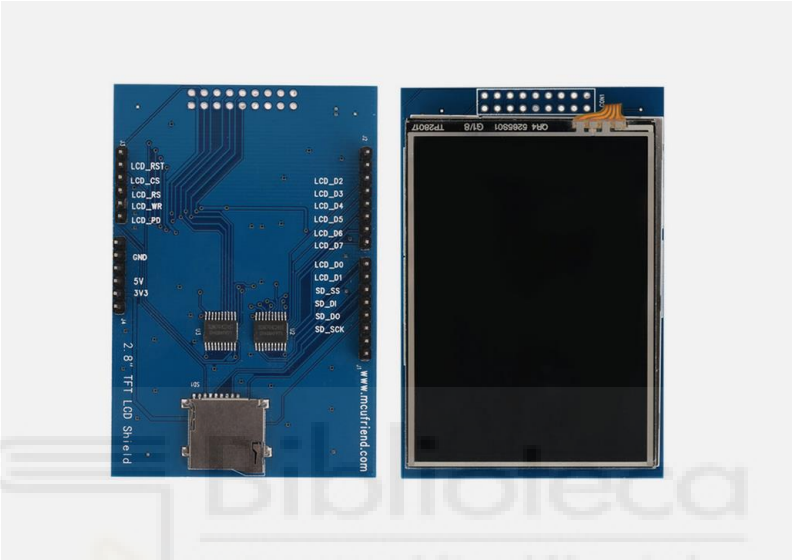


Figura 32: Pantalla LCD_TFT

En la figura 26 se puede apreciar un concepto de la aplicación desarrollada para la pantalla, en la que se muestra el modo de funcionamiento de la batería y los estados de cada celda. La información que aparece en pantalla puede editarse fácilmente desde el código para adaptarse a otras necesidades.

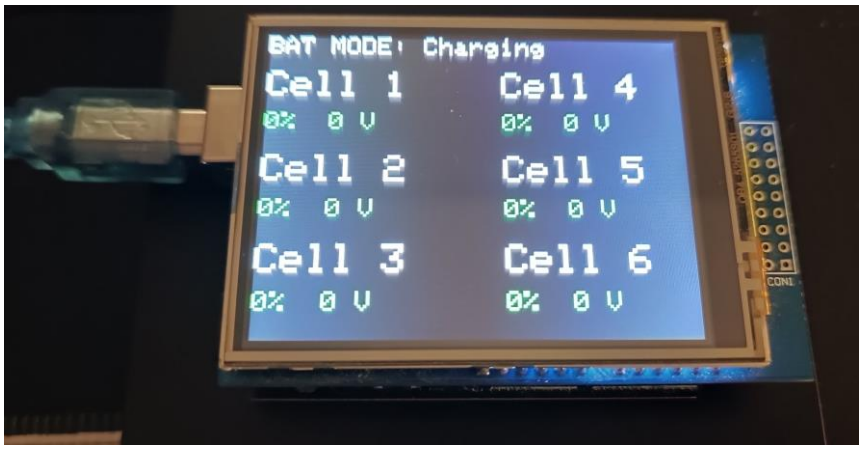


Figura 33: Información de la batería en pantalla LCD

5.3 Python. Descripción general del código.

El código de Python consiste en la creación de una interfaz gráfica sencilla para poder controlar el cargador de manera sencilla desde el PC. Establece una comunicación serie con el Arduino Nano para enviarle comandos específicos relacionados con la carga y descarga de la batería. A continuación, se presenta una descripción general del código y de su funcionalidad:

El programa se comunica el Arduino a través del puerto serie (COMX) con una velocidad de transmisión de 9600 baudios. Esta conexión permite enviar comandos desde la aplicación Python al Arduino y recibir respuestas de este.

Existen cuatro funciones principales que envían comandos al Arduino para diferentes operaciones:

- **Carga**: Inicia la carga de la batería.
- **Descarga**: Inicia la descarga de la batería.
- **Carga de almacenamiento**: Inicia una carga parcial de la batería, ideal para almacenamiento.
- **Detención**: Detiene cualquier operación de carga o descarga en curso.

Cada función envía un comando específico al Arduino y muestra en la consola cualquier respuesta recibida de este.

La interfaz gráfica se implementa utilizando la biblioteca tkinter. La interfaz incluye botones que permiten al usuario interactuar con el sistema:

- **Ventana principal**: Es la ventana que contiene todos los elementos de la interfaz.
- **Título**: Un rótulo que muestra el nombre del sistema de gestión de baterías.
- **Botones de operación**: Botones para iniciar la carga, la descarga, la carga de almacenamiento y para detener las operaciones. Cada botón está vinculado a una de las funciones de control descritas anteriormente.

- **Botón de detención**: Un botón adicional, de color rojo, que permite detener cualquier operación de carga o descarga en curso.

La aplicación inicia la ejecución abriendo la ventana principal de tkinter y esperando a que el usuario interactúe con los botones. Al presionar un botón, se envía el comando correspondiente al Arduino. Una vez que el usuario cierra la ventana, el puerto de comunicación serie se cierra adecuadamente para finalizar la conexión.

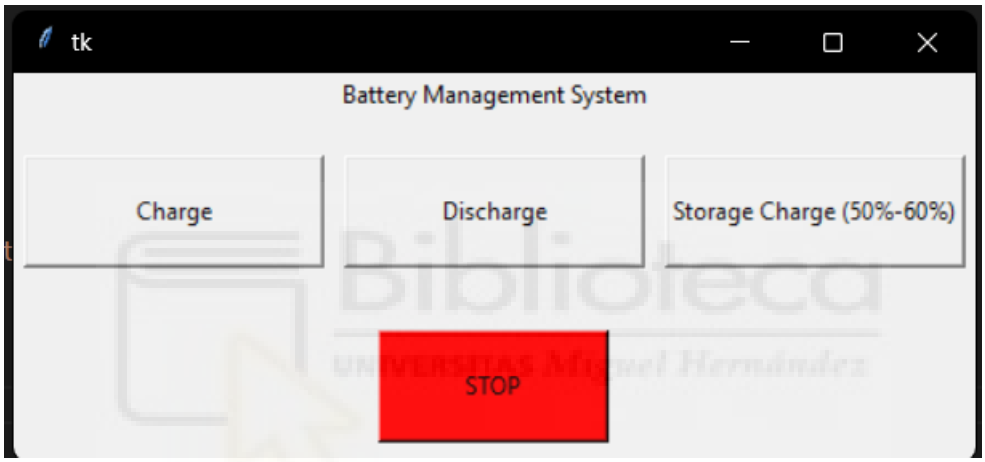


Figura 34: Interfaz creada con la librería tkinter

```
Sending charge command to arduino...
Sending discharge command to arduino...
Sending storage charge command to arduino...
Stopping battery balance...
```

Figura 35: Salida de las instrucciones de comunicación serie con el microcontrolador

6. CONCLUSIONES

El objetivo de este proyecto era diseñar un cargador de baterías 6s. Este se ha planteado como el ciclo de desarrollo de un producto, comenzando con el diseño desde cero de este y concluyendo con la fabricación y testeo. A continuación, se listan las tareas completadas:

1. Se ha analizado el estado actual de las baterías LiPo y de los sistemas BMS, entendiendo su funcionamiento.
2. Se ha planteado un diseño para un cargador de baterías 6s, escogiendo los circuitos integrados que conforman el sistema y estudiando el comportamiento de cada uno para poder implementarlos en conjunto. En base a estos circuitos integrados, se ha hecho una selección de componentes para cada uno de los circuitos para alcanzar el funcionamiento deseado.
3. Se ha llevado a cabo el diseño con KiCAD, herramienta de software de diseño de circuitos, realizando los esquemáticos y planteando la disposición de los componentes sobre la placa.
4. Se ha mandado a fabricar, montado y testeado el circuito.
5. Se ha creado una herramienta de comunicación entre el Arduino y el PC para programar el funcionamiento del cargador.

Finalmente, no ha sido posible completar el desarrollo del cargador debido al alcance del proyecto y a los recursos disponibles, por lo que queda abierta su continuación en líneas futuras.

7. ABREVIATURAS

Abreviatura	Descripción
PCB	Printed Circuit Board
BMS	Battery Management System
SPI	Serial Peripheral Interface
I2C	Inter-Integrated Circuit
UART	Universal Asynchronous Receiver-Transmitter
PC	Personal Computer
LCD	Liquid Crystal Display
TFT	Thin-Film Transistor
IC	Integrated Circuit
DC	Direct Current

8. BIBLIOGRAFÍA

[1] M. Khalid, S. S. Sheikh, A. K. Janjua and H. A. Khalid, "Performance Validation of Electric Vehicle's Battery Management System under state of charge estimation for lithium-ion Battery," 2018 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube), Quetta, Pakistan, 2018, pp. 1, doi: 10.1109/ICECUBE.2018.8610969.

[2] B. L. Alvarez, S. V. García and C. F. Ramis, "Developing an active balancing model and its Battery Management System platform for lithium ion batteries," 2013 IEEE International Symposium on Industrial Electronics, Taipei, Taiwan, 2013, pp. 1-5, doi: 10.1109/ISIE.2013.6563771.

[3] P. P. Gupta, N. Kumar and U. Nangia, "Passive Cell Balancing and Battery Charge Controller with CCCV Topology," 2022 3rd International Conference for Emerging Technology (INCET), Belgaum, India, 2022, pp. 1-5, doi: 10.1109/INCET54531.2022.9825104.

[4] I. Starostin, S. Khalyutin, A. Davidov, A. Lyovin and A. Trubachev, "The Development of a Mathematical Model of Lithium-Ion Battery Discharge Characteristics," 2019 International Conference on Electrotechnical Complexes and Systems (ICOECS), Ufa, Russia, 2019, pp. 1-4, doi: 10.1109/ICOECS46375.2019.8949976.

[5] P. Yin, N. Wang, Y. Shang, P. Gu, B. Duan and C. Zhang, "Study on the Effect of High Temperature and High-Current Rate on Fast Charging of Lithium-ion Batteries," 2021 40th Chinese Control Conference (CCC), Shanghai, China, 2021, pp. 5841-5846, doi: 10.23919/CCC52363.2021.9550628.

[6] M. Nirmala and S. Durga Shree, "Estimation of Health and Initial SOC based on Voltage Variation for LiPo battery," 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), Coimbatore, India, 2021, pp. 1670-1676, doi: 10.1109/ICAIS50930.2021.9395786.

[7] S. Yuan, H. Wu and C. Yin, "State of charge estimation using the extended Kalman filter for battery management systems based on the ARX battery model", *Energies*, vol. 6, no. 1, pp. 444-470, Jan. 2013.

[8] K. Scott, S. Nork, Active Battery Cell Balancing, Analog Devices. [Online]. https://www.analog.com/en/technical-articles/active-battery-cell-balancing.html

[9] Analog Devices. "Multicell Battery Balancer". LTC3300-1 datasheet,

[10] Linear Technology. "Multicell Battery Stack Monitor". LTC6803-4 datasheet,

[11] Analog Devices. "55V Buck-Boost Multi-Chemistry Battery Charger". LTC4020 datasheet.

[12] Analog Devices. "Quad Digital Isolator" ADUM141E datasheet

[13] Arduino Product Reference Manual.

[14] Foro Analog Devices: https://ez.analog.com/power/f/q-a/122325/ltc4020-output-fail

9. ANEXOS

9.1 ANEXO I: Listado de componentes

1. BOM

Cantidad	Lote	Referencia	Fabricante	Designador	Valor
4		CRCW06030000Z0EA	UMH stock	R1, R2, R3, R4	0
1		CRCW0603105KFKEA	UMH stock	R5	105k
1		CRCW060310K0FKEA	UMH stock	R6	10k
2		CRCW06032K70JNEA	UMH stock	R7, R8	2.7k
1		CRCW060347K0FKEA	UMH stock	R9	47k
1		CRCW060356K0FKEAC	UMH stock	R10	56k
1		CRCW0603215KFKEA	UMH stock	R11, R17	215k
1		CRCW06034R70FKEAC	UMH stock	R12	4.7
2		CRCW0603100RFKEAC	UMH stock	R13, R14, R51, R52	100
1		CRCW060324K9FKEAC	UMH stock	R15, R16	24.9k
1		CRCW06036K98FKEA	UMH stock	R18	6.98k
1		CRCW060322K6FKEA	UMH stock	R19	22.6k
1		CRCW06036R80FKEA	UMH stock	R20	6.8
12	0087802353	WSL0603R0250FEA18	Vishay Dale	R21, R22, R23, R24, R25, R26, R27, R28, R29, R30, R31, R32	25m
6		CRCW251233R0JNEG	UMH stock	R37, R39, R41, R43, R45, R47	33 (1W)
6		CRCW06033K30JNEAC	UMH stock	R38, R40, R42, R44, R46, R48	3.3k
2	2205065120	WFCP0638L000FE66	Vishay Dale	Rsense1, Rsense2	8m
1		CRCW0603100KFKEA	Vishay Dale	RT1, R49, R50	100k
1	0089331900	WSL0603R0200FEA18	Vishay Dale	RCS1	20m
4	00518-004400-23	KTJ250B476M55BFT00	United Chemi-Con	C1, C2, C11, C12	47u

Cantidad	Lote	Referencia	Fabricante	Designador	Valor
6		CL31B475KBHNFNE	UMH stock	C3, C4, C7, C8, C9, C10	4.7u
3		CL10B104KB8NNWC	UMH stock	C5, C6, C31, C33, C34	0.1u
1		CL10B333KB8NNNC	UMH stock	C13	33n
1		CL10B105KA8NNNC	UMH stock	C14, C20	1u
1		C1608X7R1H224K080AB	UMH stock	C15	0.2u
8		CL31A106KAHNNNE	UMH stock	C16, C24, C25, C26, C27, C28, C29, C30, C32	10u
1		CL10C680JB8NNNC	UMH stock	C17	68p
1		CL10C681JB8NNNC	UMH stock	C18	680p
1		CL10B103KB8NNNC	UMH stock	C19	10n
1		CL10C101JB8NNNC	UMH stock	C20	100p
1		CL31B225KBHVPNE	UMH stock	C22	2.2u
1		CL10B102KB8NNNC	UMH stock	C23	1n
2		150060RS75000	Würth Elektronik	LED1, LED2	
1	1850	PB2020	Coilcraft	L1	15u
1	MB806731.3	VS-MBRS360-M3/9AT	Vishay General Semiconductor	D1	
3	NC37607.H1	SBR0560S1-7	Diodes Incorporated	D2, D3, D4	
7	M914995.D1	BZT52C4V7-13-F	Diodes Incorporated	CR1, CR2, CR3, CR4, CR5, CR6, CR7	

Cantidad	Lote	Referencia	Fabricante	Designador	Valor
8	NA24064.D1	DMN3024LSD-13	Diodes Incorporated	Q1, Q2, Q4, Q5, Q6, Q7, Q8, Q9	
6	2058535	750032050	Würth Elektronik	T1, T2, T3, T4, T5, T6	10u
1	M954523.D1	DMP3037LSS-13	Diodes Incorporated	Q3	
6		SI2333DS-T1-E3	Vishay Siliconix	Q10, Q11, Q12, Q13, Q14, Q15	
1	2233-N830875.H1	74AHCT1G00SE-7	Diodes Incorporated	Q16	
1	P243967.H1	74AHC1G14SE-7	Diodes Incorporated	Q17	
3	0009100010	NTCLE100E3103HT1	Vishay BC Components	NTC1, NTC2, NTC3	100k
1	AX40565.6	LTC4020EUHF	Analog Devices	IC1	
1		LTC3300ILXE-1	Analog Devices	IC2	
1	2620007	LTC6803HG-4	Analog Devices	IC3	
1	AX33722.8	ADUM141E0BRZ-RL7	Analog Devices	IC4	
7		1711725	UMH stock	X1, X2, X3, X4, X5, X6, X7	
2		1-534237-3	UMH stock	H1, H2	

9.2 ANEXO II: Código Arduino.

```
#include <SPI.h>
#include <Adafruit_TFTLCD.h>

// Pin assign
const int cs = 10;
const int clk = 12;
const int mosi = 11;
const int miso = 13;

#define LCD_RESET A4 // Can alternately just connect to Arduino's reset
pin
#define LCD_CS A3 // Chip Select goes to Analog 3
#define LCD_CD A2 // Command/Data goes to Analog 2
#define LCD_WR A1 // LCD Write goes to Analog 1
#define LCD_RD A0 // LCD Read goes to Analog 0

String batmode = "Idle";

// Assign human-readable names to some common 16-bit color values:
#define BLACK   0x0000
#define BLUE    0x001F
#define RED     0xF800
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF

Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);

void setup(void) {
  Serial.begin(9600);
  pinMode(cs, OUTPUT);
  pinMode(clk, OUTPUT);
  pinMode(mosi, INPUT);
  pinMode(miso, OUTPUT);

  digitalWrite(cs,HIGH);

  //SPI.setClockDivider(SPI_CLOCK_DIV16); //sets the spi clock divider
relative to the system clock (arduino nano is 16MHz, so to get 1MHz we
set a clock div of 16)
```

```
  tft.reset();

  uint16_t identifier = tft.readID();

  tft.begin(identifier); //identifier of the tft screen

  tft.setRotation(3);
  batScreen(batmode);
}

void loop(void) {
  //Get serial from PC
  if (Serial.available()>0)
  {
    //añadir funcion que reciba info del monitor para actualizar la
pantalla

    int option = Serial.read();
    if (option >= '1' && option <= '5')
    {
      if (option == '1')
      {
        Serial.println("Charging Battery");//Devuelve al pc una
confirmación
        digitalWrite(cs,LOW);
        batmode = "Charging";
      }
      if (option == '2')
      {
        Serial.println("Discharging Battery");
        digitalWrite(cs,LOW);
        batmode = "Discharging";
      }
      if (option == '3')
      {
        Serial.println("Storage Charge Selected");
        digitalWrite(cs,LOW);
        batmode = "Storage Charge";
      }
      if (option == '4')
      {
        Serial.println("Battery Balace Stopped");
        digitalWrite(cs,LOW);
        batmode = "Idle";
      }
```

```
      batScreen(batmode);
    }
  }
}

// set display layout
void batScreen(String batstate) {
  tft.fillScreen(BLACK);
  tft.setCursor(0, 0);
  tft.setTextSize(2);
  tft.setTextColor(WHITE);
  tft.print("BAT MODE: ");
  tft.setTextSize(2);
  tft.print(batstate);
  tft.setTextSize(3);
  tft.setCursor(0, 30);
  tft.println("Cell 1");
  tft.setTextSize(1);
  tft.println();
  tft.setTextSize(2);
  tft.setCursor(0, 65);
  tft.setTextColor(GREEN);
  tft.print("0%");
  tft.print("   ");
  tft.print("0 V");
  tft.setCursor(0, 100);
  tft.setTextSize(3);
  tft.setTextColor(WHITE);
  tft.println("Cell 2");
  tft.setTextSize(1);
  tft.println();
  tft.setTextSize(2);
  tft.setCursor(0, 135);
  tft.setTextColor(GREEN);
  tft.print("0%");
  tft.print("   ");
  tft.print("0 V");
  tft.setCursor(0, 170);
  tft.setTextSize(3);
  tft.setTextColor(WHITE);
  tft.println("Cell 3");
  tft.setTextSize(1);
  tft.println();
  tft.setTextSize(2);
  tft.setCursor(0, 205);
```

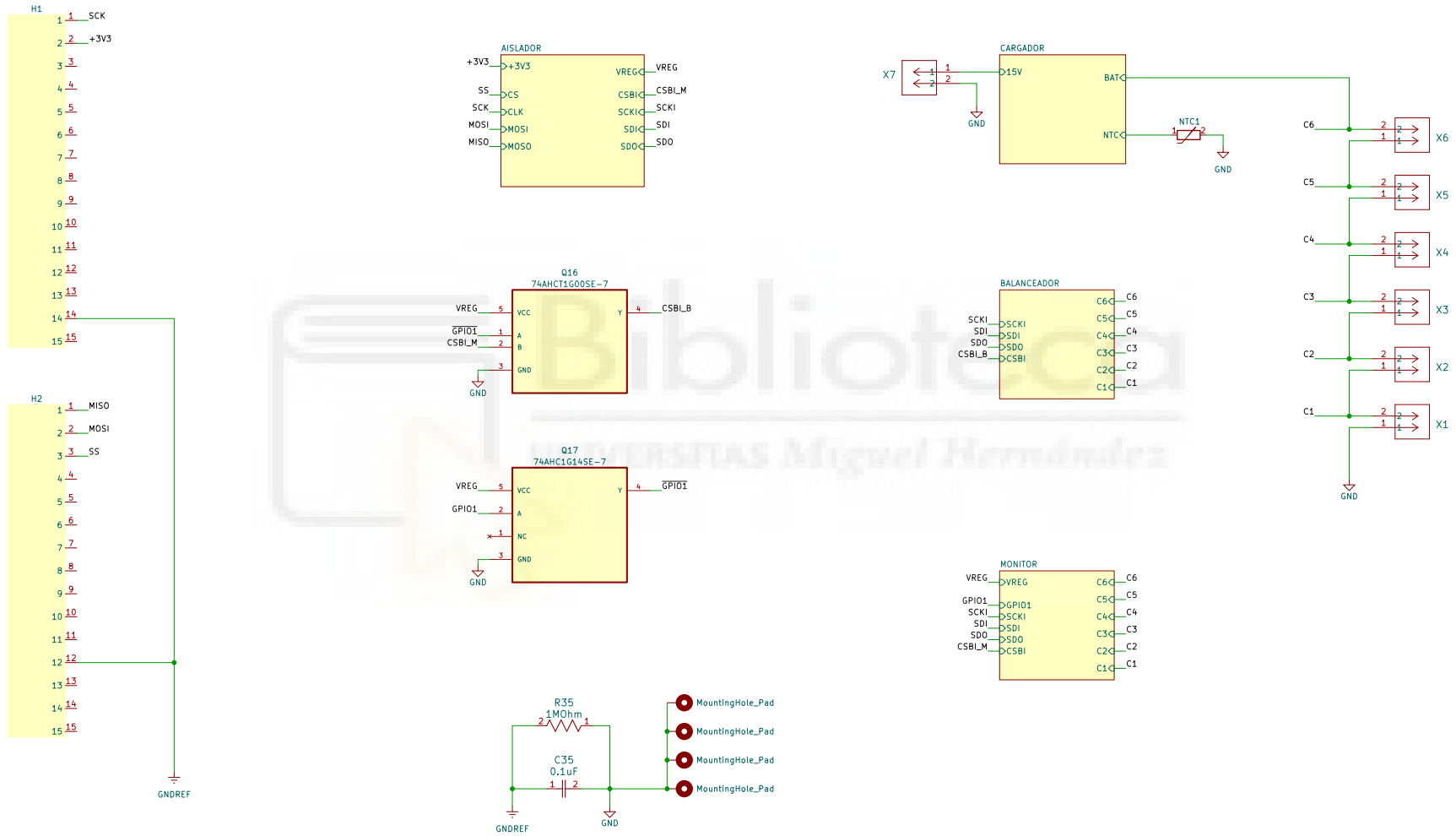
```
  tft.setTextColor(GREEN);
  tft.print("0%");
  tft.print("   ");
  tft.print("0 V");
  tft.setTextSize(3);
  tft.setTextColor(WHITE);
  tft.setCursor(180, 30);
  tft.println("Cell 4");
  tft.setTextSize(1);
  tft.println();
  tft.setTextSize(2);
  tft.setCursor(180, 65);
  tft.setTextColor(GREEN);
  tft.print("0%");
  tft.print("   ");
  tft.print("0 V");
  tft.setCursor(180, 100);
  tft.setTextSize(3);
  tft.setTextColor(WHITE);
  tft.println("Cell 5");
  tft.setTextSize(1);
  tft.println();
  tft.setTextSize(2);
  tft.setCursor(180, 135);
  tft.setTextColor(GREEN);
  tft.print("0%");
  tft.print("   ");
  tft.print("0 V");
  tft.setCursor(180, 170);
  tft.setTextSize(3);
  tft.setTextColor(WHITE);
  tft.println("Cell 6");
  tft.setTextSize(1);
  tft.println();
  tft.setTextSize(2);
  tft.setCursor(180, 205);
  tft.setTextColor(GREEN);
  tft.print("0%");
  tft.print("   ");
  tft.print("0 V");
}
```

9.3 ANEXO III: Código Python

```
import serial
import tkinter as tk

arduino = serial.Serial('COM5', 9600)   #abre puerto de comunicacion
arduino

#funciones (cargar,descargar,carga alm.,refrescar estado de las celdas)
def charge(event):
    print("Sending charge command to arduino...")
    arduino.write(b'1')
    string = arduino.readline()
    print(string)

def discharge(event):
    print("Sending Discharge command to arduino...")
    arduino.write(b'2')
    string = arduino.readline()
    print(string)

def storage(event):
    print("Sending storage charge command to arduino...")
    arduino.write(b'3')
    string = arduino.readline()
    print(string)

def stop(event):
    print("Stopping battery balance...")
    arduino.write(b'4')
    string = arduino.readline()
    print(string)

window = tk.Tk()

title = tk.Label(text="Battery Management System")
title.pack()


#BOTONES
optframe = tk.Frame(master=window)
optframe.pack(fill=tk.Y)
chargeb = tk.Button(
    master = optframe,
    text = "Charge",
```

```
    width=20,
    height=3
)
chargeb.pack(side=tk.LEFT,padx=5)
chargeb.bind("<Button-1>", charge)

dischargeb = tk.Button(
    master = optframe,
    text = "Discharge",
    width=20,
    height=3
)
dischargeb.pack(side=tk.LEFT,padx=5,pady=20)
dischargeb.bind("<Button-1>", discharge)

storagechargeb = tk.Button(
    master = optframe,
    text = "Storage Charge (50%-60%)",
    width=20,
    height=3
)
storagechargeb.pack(side=tk.LEFT,padx=5)
storagechargeb.bind("<Button-1>", storage)

refframe = tk.Frame(master=window)
refframe.pack()

stopframe = tk.Frame(master=window)
stopframe.pack(pady=10)

stopb = tk.Button(
    master = stopframe,
    text = "STOP",
    bg="red",
    width=15,
    height=3
)
stopb.pack(padx=5)
stopb.bind("<Button-1>", stop)

window.mainloop()
arduino.close()
```

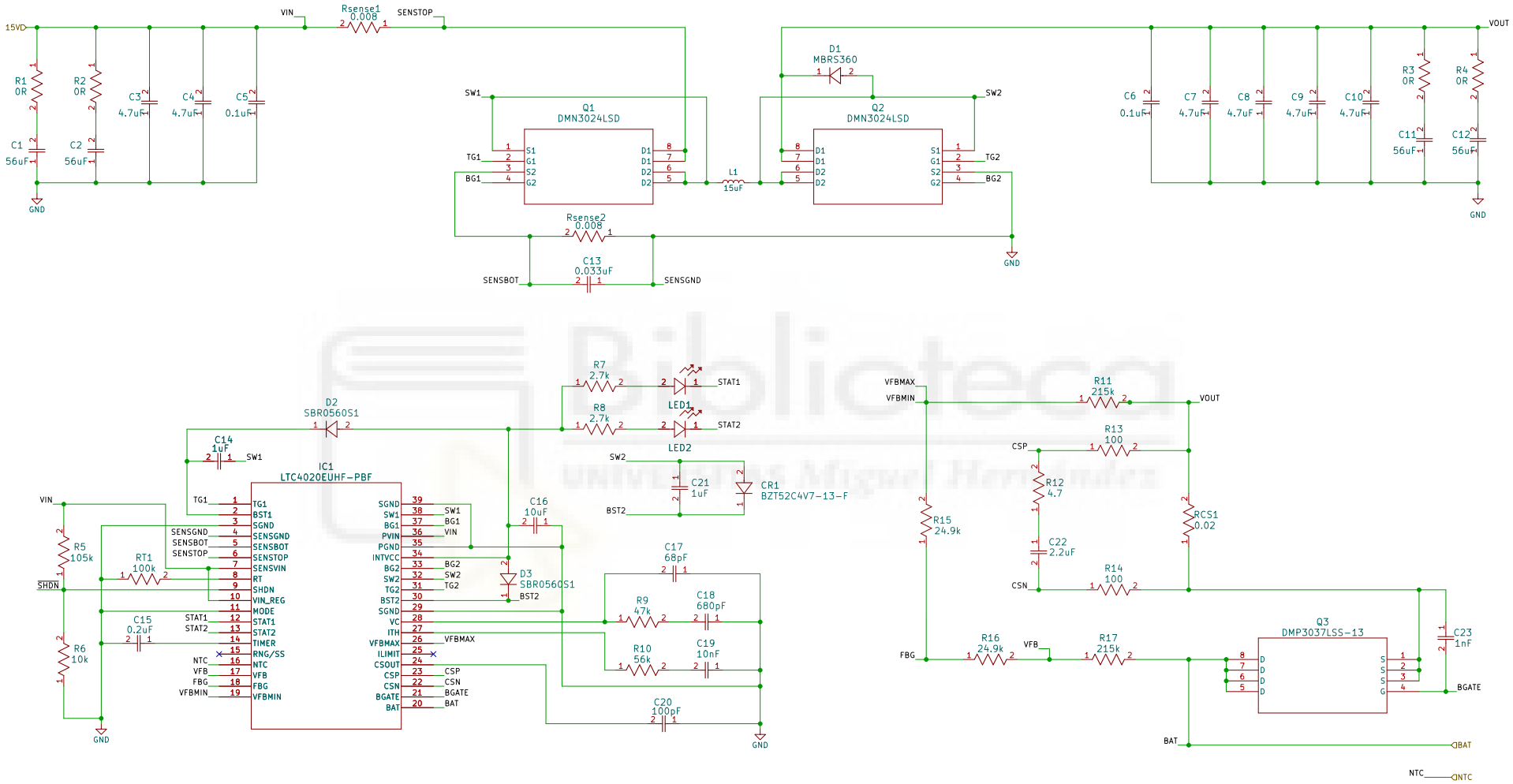
9.4 ANEXO III: PLANOS



Sheet: /
File: Cargador.kicad_sch

Title: CONJUNTO

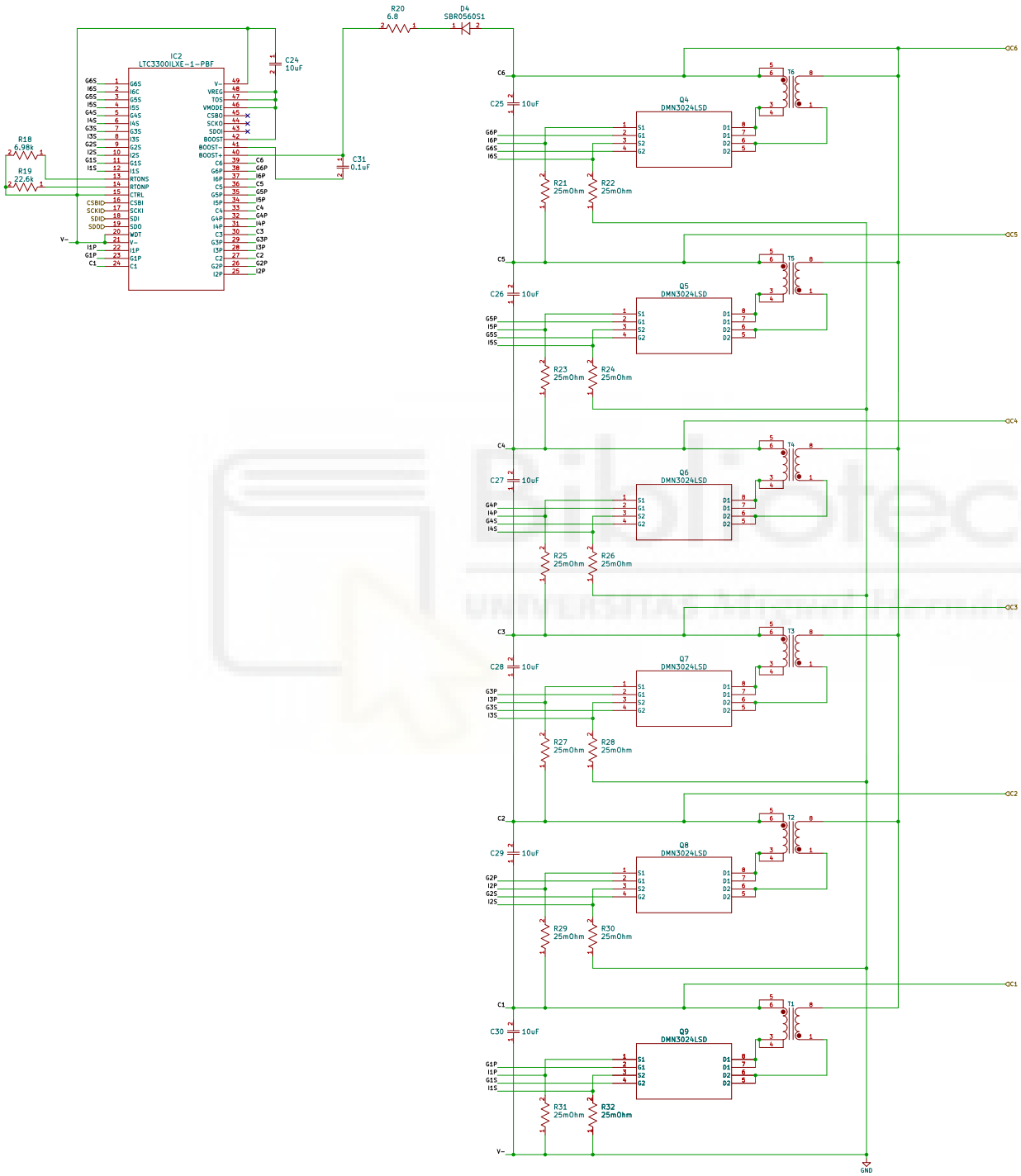
Size: A3	Date:	**Rev:**
KiCad E.D.A. 8.0.2		Id: 1/5



Sheet: /CARGADOR/
File: Cargador(DC-DC).kicad_sch

Title: CIRCUITO CARGADOR

Size: A3	Date: 1/06/24	**Rev:**
KiCad E.D.A. 8.0.2		Id: 2/5

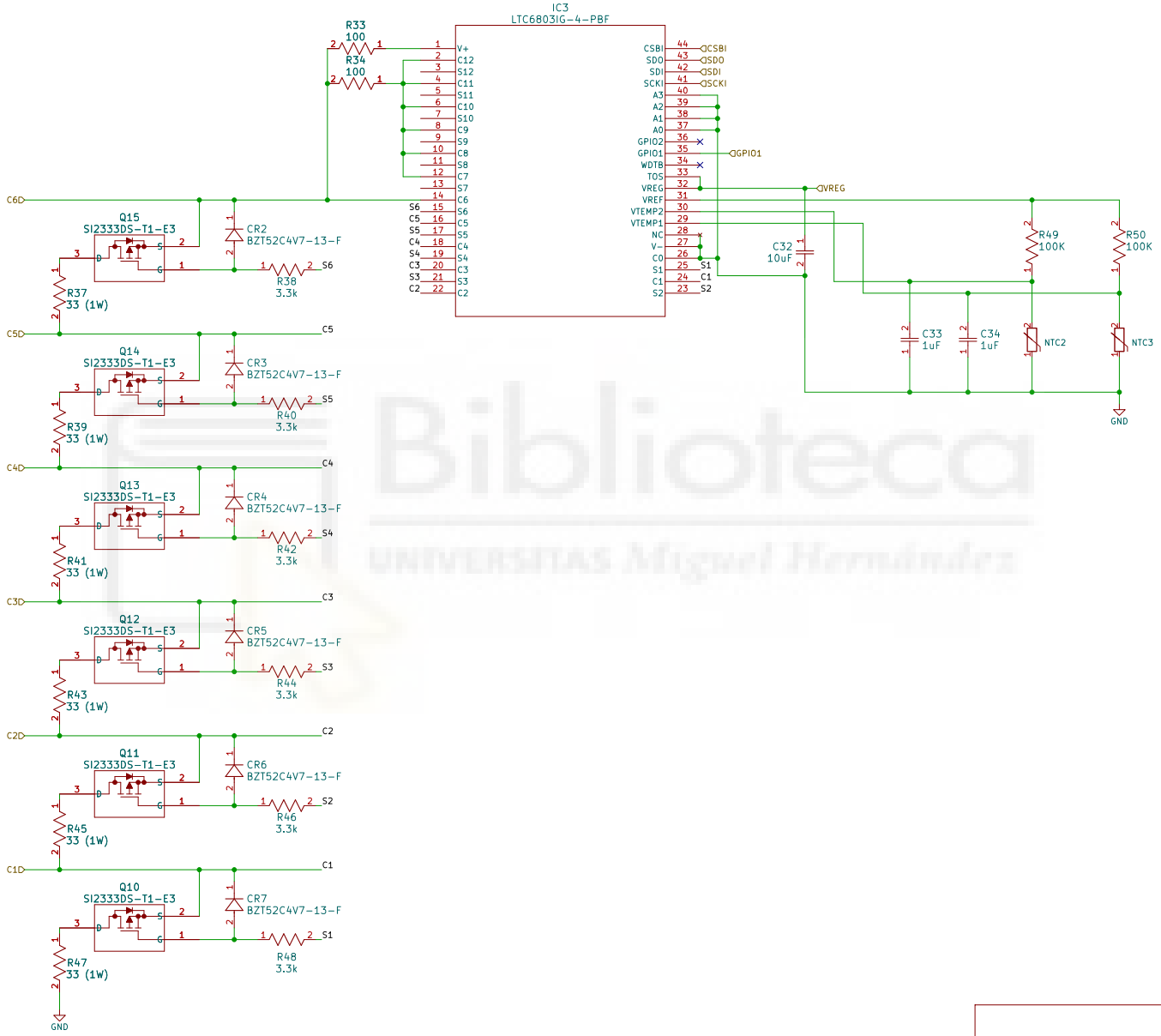


Sheet: /BALANCEADOR/
File: Balanceador.kicad_sch

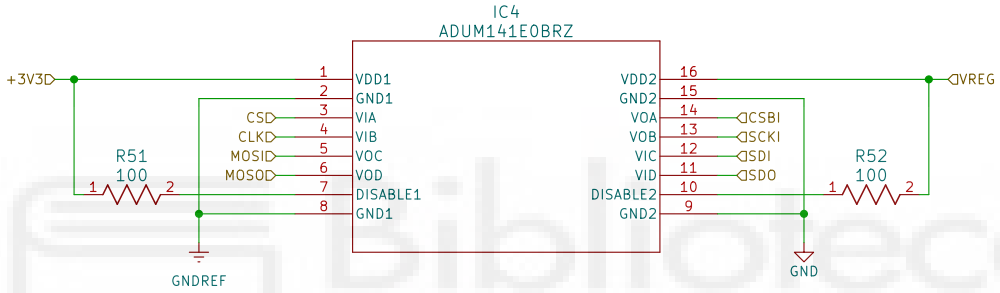
Title: BALANCEADOR

Size: A2 | Date: 1/06/24 | Rev:

KiCad E.D.A. 8.0.2 | Id: 3/5



Sheet: /MONITOR/		
File: Monitor.kicad_sch		
Title: MONITOR		
Size: A3	Date: 1/06/24	**Rev:**
KiCad E.D.A. 8.0.2		Id: 4/5



Sheet: /AISLADOR/		
File: ISOLATOR.kicad_sch		
Title: AISLADOR		
Size: A4	Date: 1/06/24	**Rev:**
KiCad E.D.A. 8.0.2		Id: 6/5