Departamento de
Ingeniería Industrial
Universidad de La Laguna

DISEÑO DE BMS PARA VEHÍCULO DE FÓRMULA STUDENT DE LA UNIVERSIDAD DE LA LAGUNA

Autor: Guillermo Coba Afonso

Tutor: Sergio Rodríguez Buenafuente

Titulación: Grado en Ingeniería Electrónica Industrial y Automática

Fecha de defensa:

C/ Padre Herrera s/n
38207 La Laguna
Santa Cruz de Tenerife. España

T: 900 43 25 26

ull.es

Índice

Resumen

The purpose of this final degree project is the design of a first prototype of a battery management system to be used in the future in the electric vehicle that is being built by students from the University of La Laguna. The vehicle is being built within the FSULL dynamics project to compete against other vehicles in this category developed by other universities.

The purpose of a battery management system is to try to optimize the amount of energy stored in a pack in addition to ensuring the integrity of the cells that make it up. To do this, it monitors the voltages of each of the series present, in order to make sure that each of them is within the safe voltage values declared by the cell manufacturer. To optimize the load, what it does is balance the voltages, that is, once the voltages are measured, it looks at the differences between each series, and if a serie has more voltage than normal, it balances it with respect to the others in different possible ways. , depending on the bms topology.

In the case of this design, it is a passive battery management system, which means that the balancing is done by short-circuiting the series to be discharged with a resistance to transform the excess energy into heat. In addition, its structure is such that you can have as many series as you want because it is based on modules capable of monitoring up to 12 series that communicate with a master via CAN bus, which makes it completely modular. External power is not required as the slaves are fed directly from the pack that balances. However the master must be powered via a low voltage battery up to 24V.

Introducción

Motivación

La motivación principal del trabajo es conocer lo más a fondo posible cómo funcionan los sistemas de gestión de baterías en los vehículos eléctricos pues hasta este año yo no sabía cómo era su funcionamiento. También me motivó poder aportar un granito de arena en el proyecto del Formula Student y que uno de los diseños fuese mío, permitiendo al resto del equipo conocer de mi mano el trabajo que hacen este tipo de dispositivos.

Origen del trabajo

El origen de la necesidad de este trabajo es allanar el camino en futuros años al equipo de formula student de esta universidad cuando quieran implementar un sistema de gestión de baterías de desarrollo propio con el fin de ganar puntos en el design event en las competiciones.

Requerimientos

Los requerimientos de este trabajo son extensos, pues debe cumplir la normativa que dice cómo se deben diseñar estos vehículos y que todos los años es actualizada por la organización de Formula Student Germany, con el fin de garantizar la seguridad de todos aquellos que están en los eventos.

Al ser un prototipo y desarrollarse durante unos momentos complicados debido a la situación en la que se encuentra el mundo debido al covid-19 algunos requisitos no se han conseguido llevar a cabo debido a los plazos de tiempo. Estos objetivos son todos conseguibles mediante código, el hardware está diseñado y probado para que todos los requisitos físicos se puedan cumplir sin excepción.

Aun así los requerimientos que debe cumplir un sistema de gestión de baterías para estas competiciones son:

- Each TS accumulator must be monitored by an AMS whenever the LVS is active or the accumulator is connected to a charger.

- The AMS must continuously measure:

  - all cell voltages

  - the TS current

  - the temperature of thermally critical cells

  - for lithium based cells: the temperature of at least 30% of the cells equally distributed within the accumulator container(s)

- The maximum cell temperature is 60 C or the limit stated in the cell data sheet, whichever is lower

- The AMS must switch off the TS via the shutdown circuit, if critical voltage, temperature or current values according to the cell manufacturer's datasheet or these rules persistently occurs for more than:

  - 500 ms for voltage and current values

  - 1 s for temperature values

  The accuracy, noise and sample rate of the measurement must be taken into account.

- AMS cell voltage measurement inputs, temperature measurement inputs and supply voltage of decentralized AMS slaves may be rated below the maximum TS voltage if the team has proven by calculations in the Electrical System Form (ESF), see EV9, that the input voltage rating is reasonably chosen.

- A red indicator light in the cockpit that is easily visible from inside and outside the cockpit even in bright sunlight and clearly marked with the lettering "AMS" must light up if the AMS opens the shutdown circuit. It must stay illuminated until the error state has been manually reset, see EV6.1.6. Signals controlling this indicator are SCS, see T11.9
- It must be possible to individually disconnect the current sensor, a temperature sensor and a cell voltage measurement wire during technical inspection, if any wire used.

- The AMS must be able to read and display all measured values according to EV5.8.2 e.g. by connecting a laptop to the AMS.

Objetivos del Trabajo

Diseño del BMS

Los objetivos del trabajo son el diseño de un hardware para un BMS capaz de cumplir todos los requerimientos de normativa, teniendo únicamente que mejorar el código creado y crear una base sobre la que poder desarrollar un sistema de gestión de baterías.

Como objetivo más concreto está el desarrollo de un sistema, lo más modular posible, que sea capaz de adaptarse a las necesidades del vehículo y del sistema de almacenamiento de energía del vehículo año a año . Por sí mismo, el sistema es capaz de medir la tensión de todos y cada unos de los series que componen el sistema de gestión, medir las temperaturas de las celdas, con 36 termistores por módulo, y medir la corriente total que sale del pack gracias a un sensor de corriente de efecto hall.

Con el fin de poder adaptarse a cualquier configuración del sistema de almacenamiento se ha optado por un sistema de esclavos y maestro. Los esclavos se alimentan del propio pack que balancea. De estas forma se pueden usar tantos series como se quiera y en caso de que alguno se rompa, al comunicarse por un bus paralelo, no afectaría al resto del sistema en nada pudiendo cambiarse solo la placa afectada.

Además se ha incluido como funcionalidad extra, que el maestro almacene todos los mensajes, a modo de caja negra, que le llegan por el bus. Así después de cada prueba del vehículo se pueden revisar todos los datos y se puede hacer con ellos lo que se quiera. Dando la posibilidad de un tratado de los datos y evaluar el funcionamiento del pack más a fondo, desde el punto de vista de la temperaturas como del punto de vista de los voltajes.

Determinación del Estado de la Carga (SOC)

Como la mayor parte de los sistemas de gestión de baterías una de sus principales funciones es el cálculo del estado de la carga o por sus siglas en inglés SOC (state of charge) del pack que monitoriza. Para ello hay distintos métodos que son comúnmente usados, pero debido al material del que disponíamos y que las celdas que iba a monitorizar ya estaban en la universidad se optó por sacar de forma experimental una curva que relaciona el estado de la carga con el voltaje en circuito abierto de la celda.

Este sistema para determinar el estado de la carga no es tan usado en sistemas comerciales porque es muy específico, se debe disponer de la celda y realizar descargas para obtener su comportamiento. Es por eso que es uno de los sistemas más efectivos que hay. Está demostrado que el error que se obtiene en celdas de Li-ion a la hora de determinar su estado de la carga es del 5%, pues la tensión en circuito abierto no depende de ninguna variable externa más que de su resistencia interna y del estado de la carga de la celda en cuestión, sin importar a penas la corriente a la que se descarga.

Luego el objetivo era dentro de los medios de los que se disponían en la universidad conseguir los datos para aproximar la curva de la forma más precisa posible. Normalmente estas pruebas se llevan a cabo forzando entornos de temperaturas y humedad constantes para reducir al máximo las variables que afecten a los experimentos. Dentro de nuestras posibilidades obtuvimos los datos sin ninguna clase de control de temperatura y humedad, obteniendo resultados bastante positivos, de cualquier manera. La ecuación que hace el ajuste tiene un error del 5% en las peores zonas pero determina bastante mejor la carga que otros métodos como puede ser el conteo de coulomb

Comunicaciones

El objetivo de las comunicaciones era encontrar un sistema lo suficientemente robusto para confiar las comunicaciones entre los distintos módulos con el maestro, quien se encargará de cortar en caso de que cualquier esclavo encuentre algún problema. Luego debido a la topología elegida del bms era primordial que fuese lo más fiable posible.

El sistema de comunicaciones elegido fue el CAN-bus pues es un estándar en la industria de la automoción. El hecho de que la comunicación sea a través de un bus paralelo hace que en caso de la rotura o defecto de una placa el resto de placas sigan monitoreando sus módulos ajenas a las otras obligando a que se tengan que romper todos los módulos para que deje de haber monitorización alguna sobre las celdas.

Además la elección de los microcontroladores que gestionan todas las placas estuvo fuertemente influenciado por el sistema de comunicación que se iba a usar. Los microcontroladores elegidos de hecho solo necesitan un interfaz para la comunicación por CAN, y no como la gran mayoría que también requieren de controladores que traduzcan la información a spi u otros protocolos de comunicación.

Entonces el objetivo final de la comunicaciones es que el sistema sea capaz de enviar todos los mensajes que salen de los 9 módulos que se tienen pensado implementar en el coche de forma robusta y segura. Es el maestro el que tomará todas las decisiones en base de los datos de temperatura y voltaje que reciba de los esclavos de ahí viene la necesidad de la robustez en la comunicación.

Estado del Arte

Este capítulo tratará de una revisión de lo que se está haciendo en la actualidad en el desarrollo de sistemas de gestión de baterías, que formas tienen, que funcionalidades implementan y cómo lo hacen. Serán presentadas también las ventajas que suponen unas configuraciones frente a otras. También se expondrán algunas químicas de baterías y distintos formatos existentes dentro de las mismas

Tecnologias de Baterias

La química de la batería que va a monitorizar el bms es algo de gran relevancia a la hora de diseñar un bms pues influye en cómo determinar el estado de la carga y en los requisitos eléctricos que deben tener los componentes del sistema de balanceo. Esto se debe a que los voltajes de las celdas varían en función de la química de las mismas y la capacidad que puede almacenar del formato de la celda.

Químicas

Dentro de las químicas de las baterías hay una gran cantidad de tipos:

- **Baterías de ácido plomo**: Comúnmente usadas en vehículos y motos. Estas baterías poseen dos electrodos de plomo, un cátodo de óxido de plomo y un ánodo de plomo esponjoso. Estas baterías usan un electrolito de ácido sulfúrico, y una vez estén descargadas el óxido de plomo se deposita en forma de plomo metálico
- **Baterías de níquel:** Fueron de las primeras en fabricarse pero dieron un muy mal rendimiento, pero de ellas surgieron nuevas químicas:
    - **Níquel-Hierro:** Eran tubos finos enrollados por láminas de acero niquelado. Fueron dejadas de usar por su alto coste y bajo rendimiento
    - **Níquel-Cadmio**: Compuestas por un ánodo de cadmio y un cátodo de hidróxido de níquel, con un electrolito de hidróxido de potasio. La principal desventaja de las baterías de este tipo es la baja densidad energética, además de sufrir severos efectos de memoria, lo que significa que sufren con las cargas
    - **Níquel-hidruro:** Usan hidróxido de níquel para el ánodo y una aleación de hidruro metálico como cátodo, lo que las hace idóneas desde el punto de vista del medio ambiente pues no usan cadmio, y fueron las primeras en usarse en vehículos eléctricos, pues tampoco sufren de efectos de memoria.

- **Baterías de ion litio**: Emplean como electrolito una sal de litio. Son las que actualmente se están usando en vehículos eléctricos por la densidad de energía que son capaces de almacenar, y su voltaje nominal está entre 3,6V y 3,7V.

  Entre los problemas que sufren las baterías de este tipo se encuentran el sobrecalentamiento y que al sobrecalentarse pueden llegar a explotar debido a que están fabricadas con materiales inflamables, lo que eleva sus costes de fabricación.

  Debido a estas características es recomendable que las baterías de este tipo tengan sistemas de seguridad como el que se diseña en este trabajo de fin de grado. La necesidad de los sistemas de gestión de la baterías viene de tratar de monitorizar y controlar cada uno de los elementos, para garantizar el correcto funcionamiento de las mismas y evitar accidentes que puedan resultar en importantes pérdidas materiales debido al alto costo de las celdas

- **Baterías de polímero de litio:** Son una variante de las baterías de iones de litio comunes. Presentan una densidad de carga mayor que las de ion litio comunes, y pueden descargar a corrientes mayores, pero su vida se reduce drásticamente cuando se descarga por debajo del 30% de su capacidad. Su voltaje nominal es de 3.7V

  También sufre los mismos problemas que las de ion litio existiendo la posibilidad de que exploten si se usan de forma inadecuada.

- **Baterías LiFePo4:** Son baterías derivadas de las baterías de ion litio con las que comparten muchas características. El voltaje nominal de estas baterías es de 3.4V. Estas baterías tienen mayor durabilidad que las de ion litio, y un voltaje de descarga muy constante lo que las diferencia de las baterías de ion litio normales.

  Además presentan como ventaja la estabilidad química y térmica lo que las hace más seguras que las baterías de las que provienen

Formatos

Entre las baterías de ion-litio, que son las que se miraron para el uso en el vehículo existen numerosas formas que adoptan estas baterías. Los dos grandes grupos que se encuentran son las prismáticas y las cilíndricas. Dependiendo del tamaño y no tanto de la forma podemos encontrar baterías de distintas capacidades, cuanto más grande sea la celda, normalmente si son del mismo fabricante, más energía pueden almacenar.Luego podemos empezar a ver las ventajas y desventajas de cada tipo de celda.

- **Prismáticas y Pouch:** Son las que suelen conformar los sistemas de almacenamiento de energía de los vehículos eléctricos, también las podemos encontrar en los dispositivos móviles. Entre las ventajas que presentan este tipo de celdas está su factor de forma que permite la creación de sistemas de almacenamiento más compactos. La eficiencia de uso del espacio de este tipo de celdas está entre el 90% y el 95% colocando a este tipo de celdas entre las mayores.

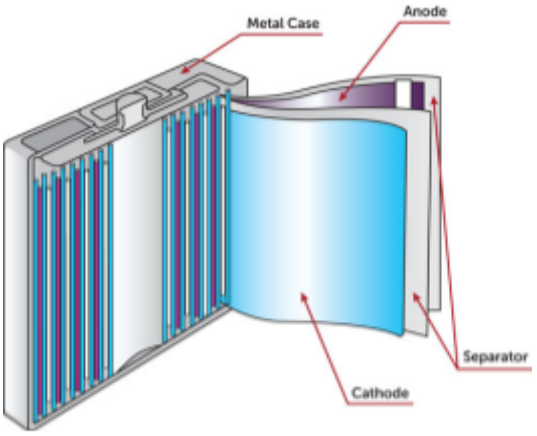


Figura 1. Batería Prismática

Una de las cosas a tener en cuenta en este tipo de celdas es la probabilidad de que se inflen lo que puede provocar que el envoltorio se rompa, pudiendo ocasionar accidentes.

Por lo general no tienen ninguna forma estandarizada y suelen ser de químicas estables como las baterías LiFePo4. Además, como ventaja frente a las celdas prismáticas o pouch las celdas de este tipo no ven sus dimensiones aumentadas con el uso.

- **Cilíndricas:** Su forma les permite ser fabricadas en masa con menores costes debido a su mayor estabilidad mecánica ya que por su forma cilíndrica puede soportar mayores presiones sin verse deformada. Aunque ahora se están empezando a emplear en vehículos eléctricos se encuentran principalmente en herramientas eléctricas, instrumentos médicos, portátiles y bicicletas eléctricas.

  Dentro de este tipo de celdas encontramos numerosos formatos de celdas estandarizadas entre los más comunes están las 18650, 20700 21700, 22700 o las recién anunciadas por tesla 4680.

  Las baterías 18650 son las más comúnmente utilizadas siendo la más optimizada ofreciendo los menores por Wh y aportando una gran fiabilidad.

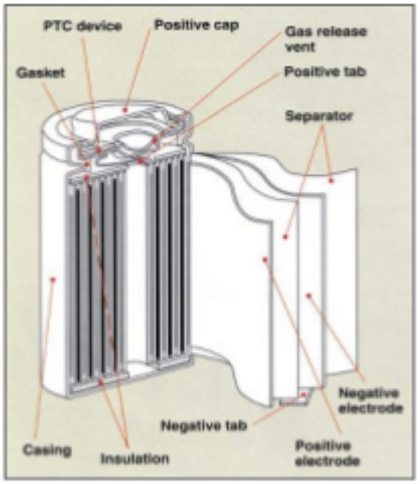


figura 2. Batería cilíndrica

  Las baterías 21700, que fueron las elegidas en el vehículo de la universidad de La Laguna están siendo foco del desarrollo pues permiten acumular mayor cantidad de energía en un formato apenas más grande que las 18650. Habiendo llegado un acuerdo Samsung Panasonic y Tesla para por su facilidad de fabricación capacidad óptima y otros beneficios

BMS

En este apartado vamos a ver que tipos de arquitecturas de BMS se suelen diseñar y utilizar en función del tipo de balanceo que realizan la forma en la que lo hacen y cómo calculan el estado de la carga.

Como ya se dijo en la introducción, la principal función de un sistema de gestión de baterías es evitar accidentes y proteger las celdas que monitoriza. Para ello se establecen límites de voltaje máximo y mínimo que evitan que las celdas se carguen o descarguen de más, lo que puede conllevar que la vida útil de las celdas se acorte o que dé lugar a accidentes más graves.

Los sistemas pueden ser tanto digitales como analógicos, presentando los primeros varias ventajas, ya que hacen mucho más que vigilar los valores de voltaje de cada celda. Estos pueden decidir si una celda tiene que ser balanceada por almacenar más energía que las demás o incluso pueden determinar el estado de las cargas de la batería. Todo esto, junto con que se puedan comunicar con el resto de sistemas del vehículo y sean capaces de almacenar los datos que recoge, los hace especialmente útiles para implementarlos en los vehículos eléctricos, pues facilita en gran medida la depuración de errores.

Como extra, los bms digitales dan la posibilidad en su mayoría de ser reconfigurables lo que permite que se puedan usar con un amplio tipo de celdas de distintos fabricantes sin problema alguno, solo teniendo que cambiar parámetros en el programa.

Tipos de Balanceo

Primero vamos a definir qué es el balanceo y por qué es tan importante dentro las funcionalidades del sistema de gestión de baterías, para que sea suficiente para clasificarlos en función de la forma en la que lo realizan.

Un pack de baterías de un vehículo consta de muchas celdas, configuradas en serie y en paralelo según las necesidades de voltaje y energía del sistema que alimenta. Normalmente los sistemas de acumulación de energía tienen que aportar tensiones que no serían capaces de dar las celdas individuales por sí mismas, luego estas se ponen en serie hasta ser capaces de dar toda la tensión necesaria.

El trabajo del balanceo de las celdas es encargarse de que todos los series estén al mismo voltaje durante la carga, cuando principalmente se producen las diferencias de tensiones entre distintos series. Estas diferencias de tensión acaban conllevando que el pack de baterías no se cargue al máximo, evitando

que este almacene toda la energía para la que fue diseñado. En el siguiente ejemplo vemos cómo podría estar las tensiones en una configuración de un pack.

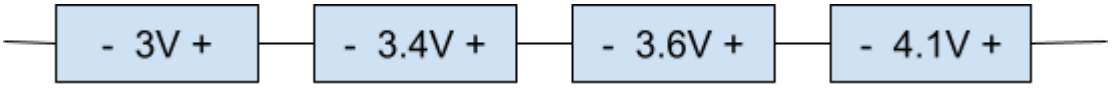


figura 3. Ejemplo celdas desbalancedas

En este caso, suponiendo que usamos las mismas celdas de ion litio que se usan en el vehículo del equipo, habría una celda que estaría a punto de estar al máximo de su capacidad, lo que para no comprometerla forzaría al sistema a parar la carga, dejando al resto de celdas bastante lejos de estar almacenando toda la energía de la que serían capaces y la tensión del pack bastante por debajo de su tensión máxima. Este tipo de casos tan exagerados se dan cuando se mezclan celdas de distintas químicas o distintas tiradas de un mismo tipo de celda, pues pese a que todas las celdas puedan ser todas fabricadas iguales y ser, supuestamente, idénticas al estar basadas en reacciones químicas pueden tener pequeñas diferencias en la cantidad de energía que son capaces de almacenar, dando lugar a que algunas se carguen antes que otras.

Para solucionar estos problemas surgió el balanceo, que lo que hace es equilibrar las tensiones de las celdas para que todas puedan llegar al mismo instante a la tensión máxima y así llegar sin problema al voltaje máximo del pack, almacenando toda la energía de la que es capaz este. Dentro de los distintos tipos de balanceo encontramos dos grandes grupos: el balanceo pasivo y el balanceo activo.

Balanceo Pasivo

El balanceo pasivo es muy simple y sencillo de implementar. Consiste simplemente en descargar aquellas celdas que estén a más tensión que el resto y en cortocircuitarlas con una resistencia en serie para que se disipe la energía hasta que esa celdas o celdas lleguen a la misma tensión que las otras.

Esta forma de balanceo permite crear sistemas más compactos y baratos por la baja necesidad de componentes, lo que lo hace la configuración elegida para la mayor parte de sistemas de gestión de baterías, como por ejemplo el sistema comercial que usa la Universidad de La Laguna actualmente.

figura 4. Orion BMS 2

La desventajas están en que la energía que disipa la convierte en calor, que puede perjudicar la refrigeración del acumulador si el BMS se encuentra en el interior del acumulador. También el mismo hecho de que disipe la energía hace que en aplicaciones en las que la energía sea todo y no se pueda desperdiciar nada no sean la mejor elección, ya que esta no se puede volver a recuperar, perdiendo por tanto carga de la batería.

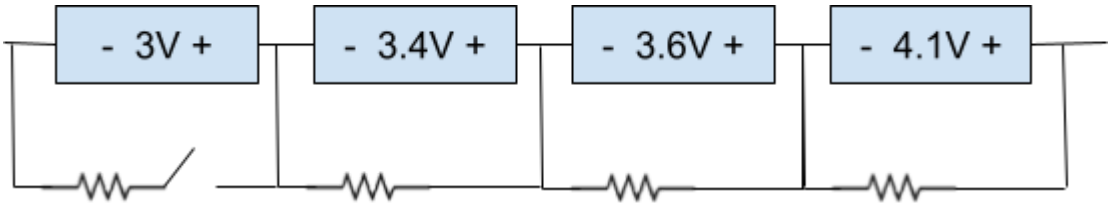


figura 5. Ejemplo celdas balanceando

En la figura podemos ver un ejemplo de cómo actuaría el balanceo. Todas las celdas que están desbalanceadas por encima del margen establecido, con respecto a la celda de menor voltaje, descargan su energía en la resistencia que tienen en paralelo. Esto se producirá hasta que la celda de menor voltaje alcance la tensión de la siguiente batería de mayor diferencia de potencial haciendo que se abra el interruptor que la cortocircuita con la resistencia, y así hasta hacerlo con todas y cada una de las celdas desbalanceadas dejando un pack perfectamente balanceado y cargado como el siguiente:

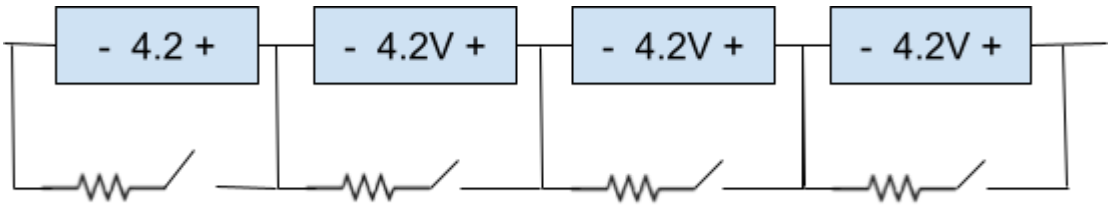


figura 6. Ejemplo celdas balanceadas

Quedando las celdas balanceadas perfectamente y todos los circuitos de balanceo abiertos, dejando al pack con la capacidad máxima que puede almacenar.

Balanceo Activo

El balanceo activo es otro tipo de balanceo existente que, por ser más complicado y necesitar más componentes para realizar el balanceo, crea sistemas más grandes y complicados. Aunque esta complicación añadida trae consigo la ventaja de que la energía que se quita de una celda pasa a las otras celdas. En estas configuraciones apenas hay pérdida de energía durante el balanceo.

Para conseguir este objetivo de evitar perder la energía se usan redes de interruptores de potencia y elementos intermedios capaces de almacenar la energía temporalmente tales como inductores y condensadores, con una forma de actuar parecida a la de los convertidores dc-dc

De esta forma de balanceo simplemente voy a mencionar algunos tipos pero no se va a hablar a fondo pues dentro de cada uno de lo tipos hay formas y formas de crear los equilibrios, pues al realizarse se debe evitar crear cortocircuitos con respecto la fuente principal y cada uno de los métodos resuelve el problema de una forma distinta.

Entre los tipos existentes podemos encontrar:

- Basados en Condensadores: Usan los condensadores como los elementos que almacenan la energía sobrante. En el primer instante el condensador es conectado a la celda de mayor voltaje hasta que es cargado al mismo voltaje que esa celda, una vez cargado es conectado a la de menor voltaje, y gracias a la diferencia de voltaje entre el condensador y la celda de menor voltaje, el condensador descarga su energía en ella.

  El proceso descrito se repetiría de forma constante descargando la carga de las baterías con mayor SOC en las de menor. Consiguiendo de esta forma minimizar las pérdidas de energía durante el balanceo en forma de calor. Los métodos basados en condensadores consiguen eficiencias durante el balanceo de alrededor del 50% debido a que se cargan los condensadores por la tensión de las celdas.

- Basados en Inductores: Alcanzan mayores niveles de eficiencia que los basados en condensadores, y usan inductores como elementos de almacenamiento intermedio de energía. Además, permite usar mayores corrientes de balanceo con inductores de menor tamaño, lo que hace que los BMS de este tipo sean notablemente más pequeños que los basados en condensadores. Algunos se basan en convertidores DC-DC como el buck-boost o el cuk

- Basados en transformadores: Los transformadores son en este caso los usados para el almacenamiento intermedio de la energía, pero además aportan el aislamiento eléctrico, en casos en los que es necesario. Entonces los transformadores son los que se encargan de transferir la energía de la celda con mayor voltaje a la de menor voltaje, luego los fuerza a estar construidos de una forma relativamente distinta a los AC-AC. Es por esto que en el núcleo ferromagnético se deja un pequeño hueco de aire que facilita el almacenamiento de energía. A este tipo de transformadores normalmente se les conoce como transformadores inductores. La mayor parte de los bms con este tipo de balanceo resuelven el balanceo activo de una forma similar al funcionamiento de un convertidor flyback.

  Las principales ventajas que aportan este tipo de bms es el balanceo aislado, y velocidades de ecualización de los voltajes muy altas. Pero estas ventajas conllevan un coste, y es que debido a la presencias de transformadores su tamaño y peso son muy superiores a los tipos anteriormente mencionados

Topologías

En este apartado de topologías vamos a describir de forma sencilla las distintas configuraciones que pueden presentarse de BMS, explicando en qué consisten y poniendo ejemplos en algunos de los casos.

Los dos grandes grupos que podemos ver son centralizados. Es decir, todas las funciones se realizan en la misma unidad y descentralizados. Lo que significa que hay funciones que se realizan en módulos auxiliares que pueden estar perfectamente en otro lugar distinto al módulo principal o maestro

Bms Centralizados

Un bms centralizado es aquel en el que un único maestro monitorea y mantiene los parámetros de todas las celdas individuales conectadas en serie dentro del pack. Este sería el caso de BMS comerciales como el que posee el equipo en la actualidad, el ORION BMS2

figura 6. Orion BMS

En este tipo de bms solo hace falta un sensor de corriente, pues la misma corriente fluye por todas las celdas. Todos los cálculos, como los del SOC, el estado de salud de la batería o state of health también son realizados por la misma unidad.

La principal ventaja de este tipo de bms es la reducción de costes que supone tenerlo todo controlado por un único microcontrolador, pero este tipo de soluciones es problemática cuando ya empiezan a usarse configuraciones de grandes cantidades de series en los que se intenta conseguir tener grandes tensiones. Este tipo de problemas suelen encontrarse en EV(electric vehicles) debido a los requerimientos de los motores.

BMS Descentralizados

Son una alternativa emergente a los bms centralizados que tienen gran cantidad de soluciones distintas, todas basadas en la presencia de múltiples microcontroladores comunicados entre sí para realizar el control del estado de las baterías.

Entre los diferentes tipos que podemos encontrar están los BMS jerárquicos, que es la solución por la que se ha optado en este trabajo, los BMS parcialmente descentralizados y un bms completamente descentralizado.

- **BMS Jerárquico**. Básicamente en este tipo de configuraciones hay módulos que monitorizan grupos de celdas, de forma que se puedan tener tantas celdas como se quiera, pero deben ser múltiplos de la cantidad de celdas que pueden monitorizar esos módulos.

  Normalmente estas unidades de monitorización, conocidas como MMU(Module Monitorization Units) se alimentan directamente del módulo de celdas. Estos módulos también realizan el balanceo entre las celdas del mismo módulo.

  Todos los MMU están conectados a una unidad central vía sistema de comunicación, que en caso de encontrarse en vehículos eléctricos suele ser CAN bus. Este sistema central (también referido en otras partes como maestro o como PMU Pack Managment Unit) se encarga de leer todos los datos que mandan las MMUs, calcular el estado actual de la carga y controlar los relés de seguridad o la velocidad a la que deben ir los ventiladores en función de los datos de temperatura que recibe.

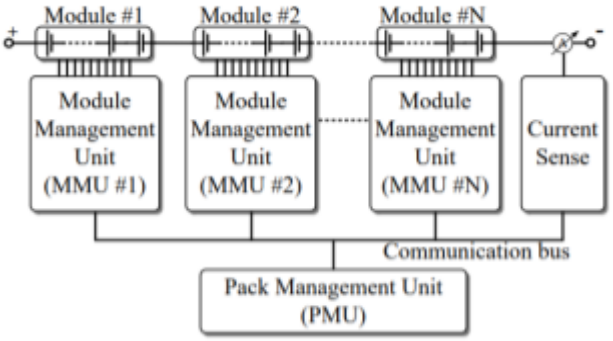


figura 7. Esquema BMS jerárquico

En comparación con los sistemas centralizados el cableado se reduce en gran cantidad pues la mayor parte va a los módulos de monitorización que comúnmente se instalan junto a cada módulo. Esto hace que los únicos cables que recorren el bus sean los cables de comunicación entre módulos y maestro.

- **BMS parcialmente distribuido:** En este tipo de sistema cada celda está equipada con un sistema local e individual de monitorización alimentado por la misma celda y se encarga de mantener los valores de la celda dentro de su rango de operación segura. Pero en este caso todas las unidades de control de celdas están conectadas a una unidad maestra, tal y como se hace en el sistema jerárquico.

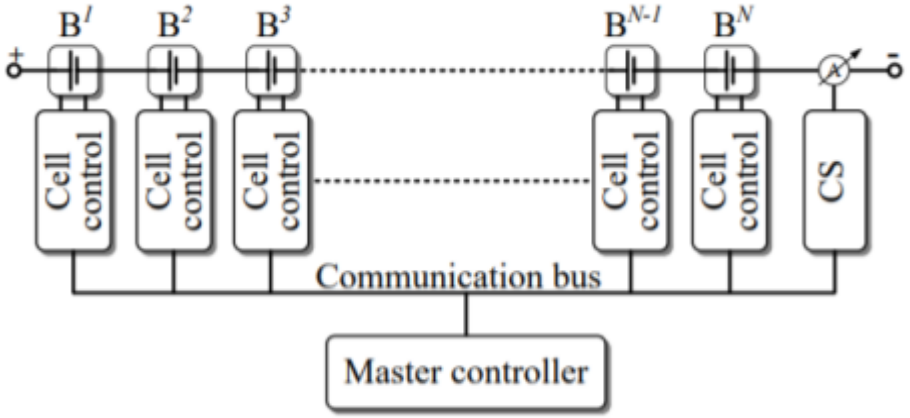


figura 8. Ejemplo BMS parcialmente distribuido

Esta unidad de control maestra tiene las mismas funciones que en el sistema jerárquico, calcula el SOC, el SOH y se encarga de accionar los relés. Con este sistema se reduce aún más la cantidad de cableado necesaria pero tiene como desventaja que hacen falta tantos microcontroladores como series hayan.

- **BMS completamente distribuido:** Es exactamente igual que el sistema parcialmente distribuido con una sola excepción, y es que no tiene un maestro y cada sistema individual es capaz de realizar las mismas funciones que realizaba el maestro antes

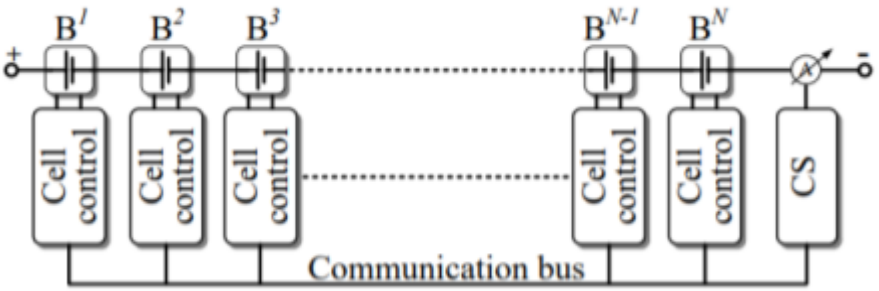


figura 9. Ejemplo bms completamente distribuido

Métodos de cálculo del SOC

El estado de la carga también conocido como SoC o state of charge (Estado de la Carga). Es el valor que representa la cantidad de energía que el pack de baterías tiene almacenado. Luego para obtener este valor, que sirve para saber la cantidad restante de energía de la que dispone el vehículo, se han desarrollado durante los años varias aproximaciones. Entre ellas están algunas más efectivas pero más laboriosas y otras que apenas requieren de trabajo previo pero que presentan problemas como derivas o grandes requisitos de computo.

Entre los métodos existentes se encuentran los siguientes:

- **Estimación por prueba de descarga:** Este método consiste en descargar la celda y estimar el estado de la carga, basado en los datos obtenidos de esta descarga.
- **Estimación por conteo de coulomb:** Este método consiste en la integración de la corriente que fluye por el pack durante un determinado espacio de tiempo. Este método conlleva derivas y normalmente da lugar a imprecisiones muy grandes.
- **OCV(Open Circuit Voltage):** El estado de la carga de una celda depende del valor de tensión del circuito abierto de una celda. Estos valores se pueden consultar en una tabla, la cual, como es el caso de este trabajo, se puede obtener mediante una prueba de descarga y haciendo la equivalencia teniendo en cuenta la resistencia interna de la celda. Este método suele dar

muy buenos resultados. Lo que se suele hacer con la tabla es sacar una ecuación polinómica que permita obtener el SOC en función del OCV.

Esta aproximación es bastante buena, pues tal y como se ve en los papers adjuntados en las referencias, la variación de la resistencia interna de una celda en una misma tirada es bastante baja. Esto permite poder estimar sin problema el SOC y obtener resultados consistentes.

Diseño del BMS

En este capítulo vamos a hablar del diseño del sistema de gestión de baterías. Discutiremos qué elementos conforman los circuitos y por qué se decidió implementar las funciones de esa forma .

Diseño del Hardware

Dentro del hardware lo más importante a mencionar es la estructura que se decidió que tuviese el sistema. Este se trata de un sistema de gestión de baterías jerárquico con esclavos o Module Management Units, capaces de monitorizar hasta 12 celdas en serie. Además estos MMU son capaces de medir la temperatura gracias a 36 termistores mediante las entradas analógicas del microcontrolador y 2 multiplexores.

El maestro o Pack Management Unit PMU tiene una entrada para el bus de comunicaciones, y puede controlar la velocidad de los ventiladores mediante señal PWM y cortar los circuitos de carga y descarga gracias a 2 salidas para los relés de control. Por último, este se encarga de la medición de la intensidad del pack mediante un sensor de corriente de efecto hall.

Maestro

El maestro se trata de la placa principal también conocida como PMU, es la placa que se encarga de procesar todos los datos recogidos por todas las MMUS, tomar decisiones en función a ellos y almacenar los datos en la tarjeta SD. Como sistema de seguridad no puede empezar a procesar datos o a funcionar cuando la tarjeta SD no está insertada.

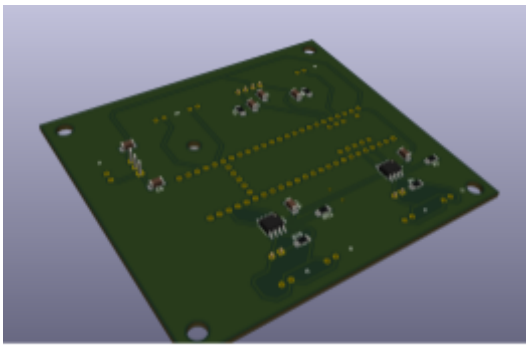


figura 10. Render PCB maestro

Componentes

El diseño de la placa gira alrededor del microcontrolador elegido. En este caso, la placa de desarrollo en la que se basa es la Teensy 3.5.

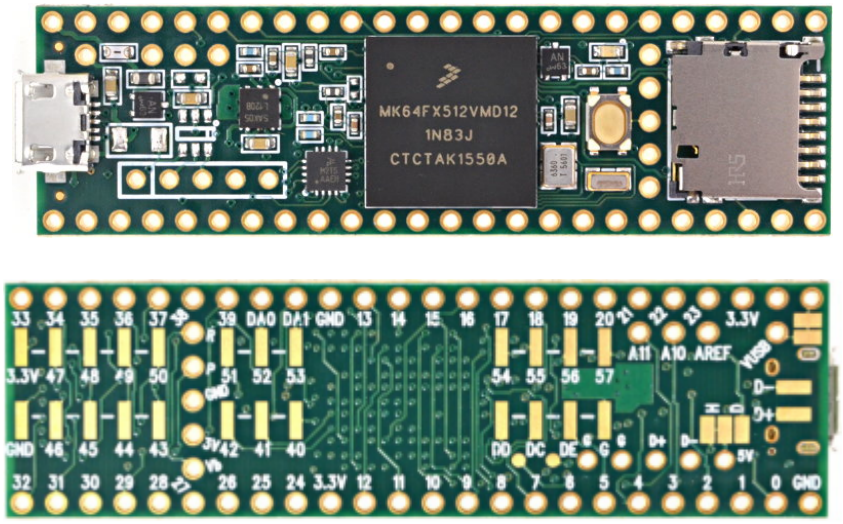


figura 11. Placa desarrollo maestro

Esta placa fue elegida para el desarrollo del maestro por 3 motivos:

- Compatibilidad nativa con el CANbus.
- Velocidad del microprocesador de 120 Mhz: lo que le permite realizar operaciones más pesadas en poco tiempo.
- Lector de tarjeta micro SD integrada.

Estas tres características hacen de él una elección perfecta, pues permitía que pudiese leer todos los mensajes de gran cantidad de MMUs y además procesar los mensajes que recibe a mayor velocidad que un Arudino Uno.

figura 12. MCP2551

El hecho de que tuviese el microcontrolador la compatibilidad con el CANbus hacía que la placa a diseñar fuese más sencilla reduciendo en gran medida la cantidad de componentes. Pues solo para el bus hacía falta un CAN interface usándose en este caso el MCP2551. Este CAN interface se eligió tras leer que estaba entre las recomendadas en el repositorio de github de collin 80

acerca de la librería FLEX CAN que sirve para comunicarse por CANbus con las placas de desarrollo de esta marca.

Figura 13. LEM DHAB S/124 Tamura L31S200S05FS

Para la medición de la corriente el sistema tiene 4 pines de 2.54mm dedicados para poder leer los valores de voltaje que devuelve el sensor de corriente elegido. En este caso, es el LEM DHAB S/124 Tamura L31S200S05FS. Fue elegido porque es capaz de medir los valores de corriente máximos que se van a producir en el sistema de tracción del vehículo, 200 A. Además este sensor no hizo falta comprarlo pues ya se tenía, ya que era el que usaba el BMS comercial del que dispone el equipo.

Este sensor de corriente tiene dos canales de medida uno para aportar más precisión en el rango de -70A a 70A. Y otro dedicado a la medición con menos sensibilidad de valores de corriente mayores con valores de -500A a 500A.

La placa del PMU además cuenta con dos salidas de control para relés de control que controlan la carga y la descarga del acumulador del vehículo. Estas salidas son simplemente salidas digitales de 3.3V que necesitan de un circuito de acondicionamiento de la señal en caso de que los relés a controlar requieran de más tensión para funcionar. No se optó por incluir el circuito de acondicionamiento de la señal para dar libertad al equipo en que relés usar y no forzarlos a que dichos relés tengan que trabajar a un voltaje determinado.

La alimentación de la placa del prototipo es sencilla, como la tensión nominal de la batería de bajo voltaje del vehículo es de 24V se ha optado por usar un LM7805 que regule la tensión hasta los 5V que requieren todos los elementos del prototipo para que funcionen.

Circuitos

Los subcircuitos que componen la placa son los siguientes:

Alimentación

El circuito de alimentación consta simplemente de un LM7805 y los condensadores de entrada y salida que según el datasheet hay que usar para su correcto funcionamiento.

figura 14. Circuito LM7805

Comunicación

Para la comunicación se ha diseñado el circuito de forma que se permita que sólo con dos cables de placa en placa se siga manteniendo el bus paralelo. Para ello se ha diseñado con dos conectores de comunicación que están conectados dentro de la placa entre sí como si fuesen el mismo punto.

Además la placa tiene la posibilidad de tener la terminación del bus CAN de 120Ω tan solo poniendo un jumper en dos dos pines dedicados para eso siendo el esquemático del circuito el siguiente.

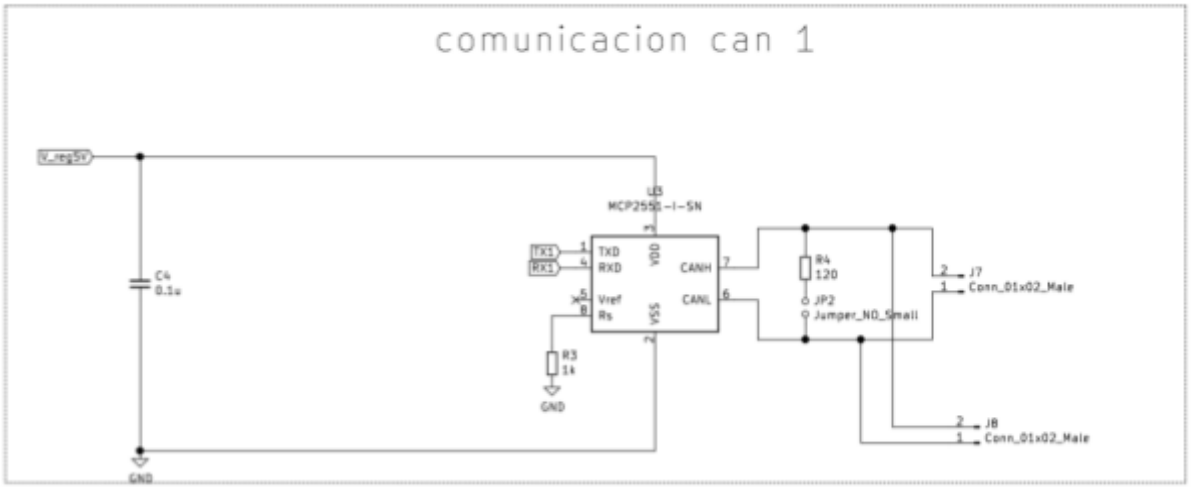


figura 15. Esquema circuito MCP2551

El integrado es alimentado directamente del regulador de 5v y no presenta ningún problema con que el integrado trabaja a 3.3V tal y como se mencionó antes se eligió comprobando anteriormente que la compatibilidad estaba asegurada por la comunidad que había desarrollado la librería para las comunicaciones

Acondicionamiento sensor de corriente

Para el sensor de corriente se ha usado el circuito de acondicionamiento de la señal recomendado por el fabricante en la hoja de datos del mismo. Siendo tan solo dos filtros RC uno por cada sensor de corriente que dispone el sensor comercial. Siendo el circuito el que podemos ver en la siguiente imagen.

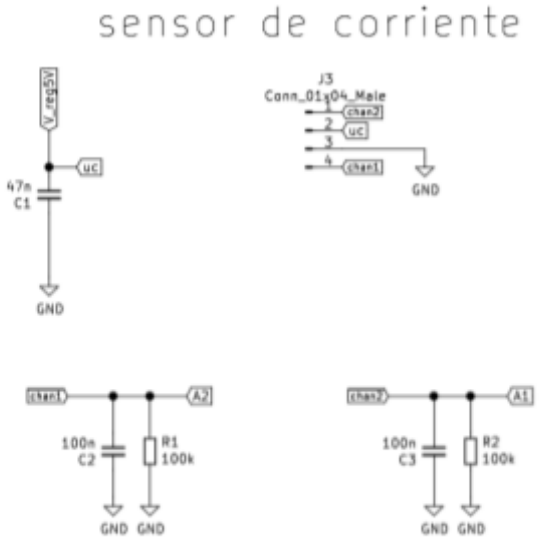


figura 16. Circuito acondicionamiento sensor de corriente

Esclavos

El diseño de los esclavos o MMU está basado alrededor de dos componentes principales. El microcontrolador elegido, esta vez una versión más sencilla del teensy 3.5, en este caso el teensy 3.2, también capaz de comunicarse por CANbus de forma nativa.

El otro componente clave del diseño es el ltc6804-1, el integrado que se encarga de la medida de tensión de todas las celdas y que gestiona el balanceo de las mismas. Este integrado realiza todas estas funciones y algunas más que no son aprovechadas por no ser necesarias, las almacena y se comunica con el microcontrolador por SPI para cargar nuevas configuraciones y enviar los datos de voltaje.

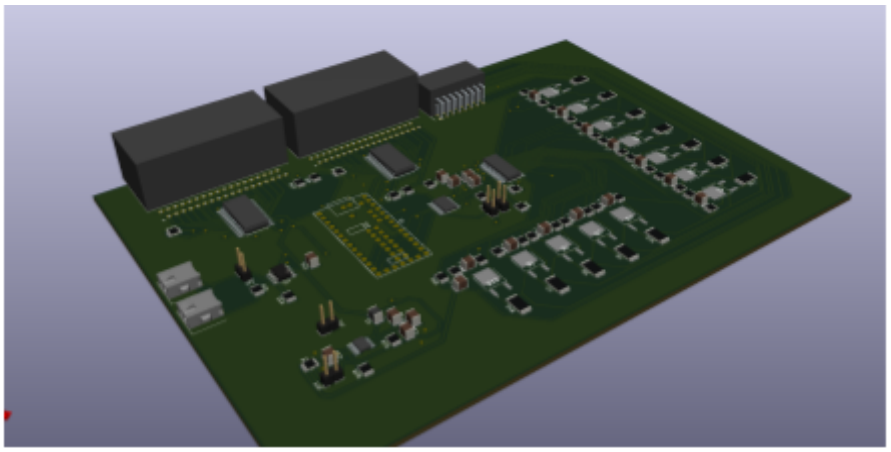


figura 17. Render PCBs esclavos

Entonces este MMU es capaz de comunicarse por el bus del sistema medir y balancear hasta 12 series y además es capaz de monitorizar la temperatura a través de 36 termistores NTC, medidos en las entradas analógicas del microcontrolador siendo 32 de ellos señales multiplexadas por dos multiplexores de 16 a 1 comerciales.

Componentes

Microcontrolador

El microcontrolador elegido para el control de esta placa es de la misma familia que el de la placa de los maestros. En este caso se trata de el teensy 3.2. Este

microcontrolador se eligió porque era el microcontrolador más barato que este fabricante hacía con compatibilidad nativa con el CANbus.

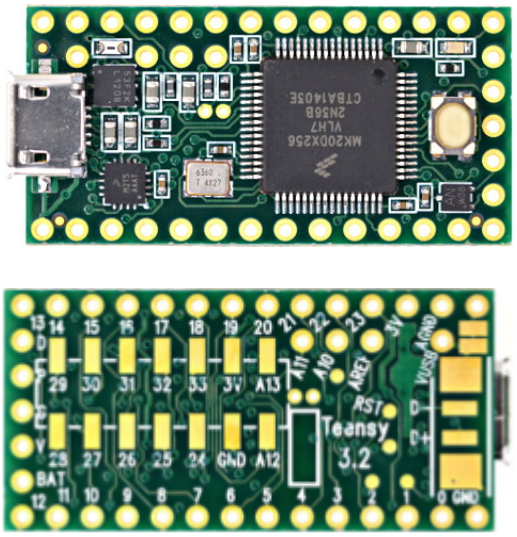


figura 18. Placa desarrollo esclavo

Entre las características de este microcontrolador encontramos:

- ARM Cortex-M4 at 72 MHz
- 256K Flash, 64K RAM, 2K EEPROM
- USB device 12 Mbit/sec
- 34 digital input/output pins, 12 PWM output pins
- 21 analog input pins, 1 analog output pin, 12 capacitive sense pins
- 3 serial, 1 SPI, 2 I2C ports
- 1 I2S/TDM digital audio port
- 1 CAN bus
- 16 general purpose DMA channels
- RTC for date/time

Integrado de monitorización de celdas

El integrado empleado es el LTC 6804-1. Este integrado puede cumplir gran cantidad de funciones. La funciones para la que es utilizado son la conversión analógica digital de la tensión de cada serie que monitoriza el integrado y la

manipulación de las señales de control que permiten el balanceo pasivo de las celdas.

Toda la información que procesa y las acciones que puede realizar son en este caso controladas por SPI mediante una librería que aporta el fabricante y que es compatible con la programación de nuestro microcontrolador.

Este integrado también es capaz de realizar más funciones de las que se llegan a utilizar destacando la capacidad de lectura y tratamiento de la señal analógica de un sensor de corriente de efecto Hall o varias entradas multipropósito para la lectura por ejemplo de las señales de hasta 5 termistores siendo de forma más detallada sus características las siguientes:

figura 19. LTC 6804-1

- Measures Up to 12 Battery Cells in Series
- Stackable Architecture Supports 100s of Cells
- Built-In isoSPI™ Interface:
    - 1Mbps Isolated Serial Communications Uses a Single Twisted Pair
    - Up to 100 Meters Low EMI Susceptibility and Emissions
- 1.2mV Maximum Total Measurement Error
- 290µs to Measure All Cells in a System
- Synchronized Voltage and Current Measurement
- 16-Bit Delta-Sigma ADC with Frequency Programmable 3rd Order Noise Filter
- Engineered for ISO26262 Compliant Systems
- Passive Cell Balancing with Programmable Timer
- 5 General Purpose Digital I/O or Analog Inputs:
    - Temperature or other Sensor Inputs
    - Configurable as an I2C or SPI Master
- 4µA Sleep Mode Supply Current

- 48-Lead SSOP Package

Este integrado debe ser alimentado a 5v para que se capaz de funcionar, y luego puede gestionar packs de hasta 75 V lo que lo hace ideal para nuestra aplicación en particular, ya que la tensión máxima que tendría el pack con la química actual suponiendo que es de 12 series, seria de 50.4V.

Regulador 60V a 5V

Como todas los componentes de la placa funcionan a 5v a excepción de la placa de desarrollo que funciona a 3.3V pero posee un regulador propio necesitamos algo que nos permita bajar la tensión hasta los 5V. La forma elegida para bajar la tensión del pack a la tensión que podemos usar en el módulo es un regulador de tensión. El regulador elegido es del mismo fabricante del que se eligió el integrado encargado del balanceo, analog devices.

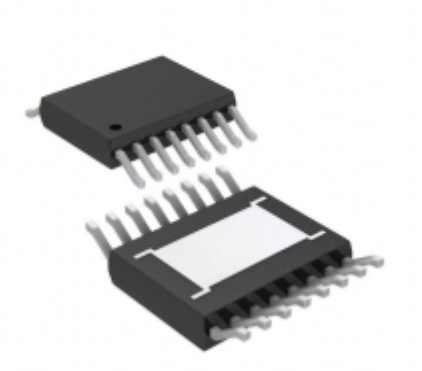


figura 20. LT 8619

El regulador en cuestión es el LT 8619-5V, que tiene una salida fija de 5V y por tanto requiere de menos componentes para funcionar que uno con salida ajustable. El integrado se trata de un regulador buck monolítico, síncrono, con una salida de hasta 1.2A suficiente para alimentar todos los elementos incluidos en la placa.

consumos circuito esclavo			
componente	ud	consumo máx/ud	consumo máx total
arm cortex m4	1	39	39
cortexm0	1	6,2	6,2
ltc6804-1	1	24,5	24,5
termistores	6	20	120
mcp2551	1	75	75
cd74hct4067m	2	50	100
		total	364,7

Multiplexores

Los multiplexores usados son circuitos integrados con 16 a 1 con 4 señales digitales de control. Estos se usan para multiplexar 32 de las señales de temperatura pues el microcontrolador tan solo dispone de 6 entradas analógicas. Por cantidad de celdas la normativa exige que usemos 36 termistores, 4 son usados directamente en las entradas analógicas y 32 son divididos en grupos de 16 por los 2 multiplexores que se montan.

figura 20. Multplexor

El multiplexor usado es el CD74HCT4067, un multiplexor muy común en la electrónica de hobby del que hay gran cantidad de documentación acerca de cómo hacerlo funcionar.

Entre las principales características del componente están:

- - 4.5V to 5.5V Operation
- Direct LSTTL Input Logic Compatibility,

  VIL= 0.8V (Max), VIH = 2V (Min)

- CMOS Input Compatibility, Il ≤ 1μA at VOL, VOH

Traductor de nivel lógico

Debido a que el spi del microcontrolador trabaja a 3.3V y el del integrado de balanceo funciona a 5V fue necesario un componente de este tipo para que fuese posible la comunicación entre ambos.

figura 21. Traductor de niveles lógicas

El circuito empleado para este propósito es TXB0108PWR que tiene las siguientes características:

- 1.2 V to 3.6 V on A Port and 1.65 V to 5.5 V on B Port (VCCA ≤ VCCB)
- VCC Isolation Feature – If Either VCC Input Is at GND, All Outputs Are in the High-Impedance State
- OE Input Circuit Referenced to VCCA
- Low Power Consumption, 4-µA Max ICC
- Ioff Supports Partial-Power-Down Mode Operation
- Latch-Up Performance Exceeds 100 mA Per JESD 78, Class II
- ESD Protection Exceeds JESD 22

Comunicación CAN

Ver la comunicación CAN del maestro, se usa el mismo integrado.

Termistores

Los termistores no son estrictamente componentes que forman parte de esta placa pero si fuerzan la decisión acerca de determinados valores. Se ha decidido usar un partidor de tensión para adaptar la señal y hacerla adecuada para la adquisición de los datos.

figura 22. NTCs

El termistor elegido para la toma de temperatura es NTCALUG03A473H que por su forma se trata de un termistor ntc de arandela. lo que simplifica bastante su montaje en el pack de baterías ya que solo hace falta un tornillo y una tuerca de métrica 2 para su instalación.

Los características de estos termistores son las siguientes:

- Resistance value at 25 °C: 47 kOhms
- Parámetro B: 3740 K
- Constante de tiempo: 5 s

Conocidos estos valores ya somos capaces de calcular la Constante A que acaba de caracterizar la ecuación que describe la variación de la temperatura con respecto a la resistencia de la NTC.

$$R_t = A \cdot e^{B/T}$$

Circuitos

En este apartado se va a hablar acerca de la forma en la que ha sido diseñado cada circuito y por que se han seleccionado los tamaños de los componentes y demás cosas relacionadas con los circuitos.

Circuito de balanceo y adquisición de voltaje

Este es el circuito principal de la placa, este circuito se encarga del control de de las celdas, tanto de monitorizar el valor de la tensión como de alternar si la celda está balanceando o no.

El principal componente, es la resistencia que se encarga de disipar la energía sobrante en las celdas que hay que balancear. Esta resistencia en este caso es una resistencia smd 2512 de 3W de potencia y 11Ω. Fue elegida esta potencia por ser un valor comercial normal para resistencias con estas medidas y además nos deja un margen de seguridad muy alto permitiendo el uso de voltajes superiores.

Este circuito también tiene un filtro RC que recomienda el fabricante usar para limpiar la señal y facilitar la labor al conversor analógico digital incorporado en el integrado de balanceo en el caso de querer emplear frecuencias de conversión muy altas.

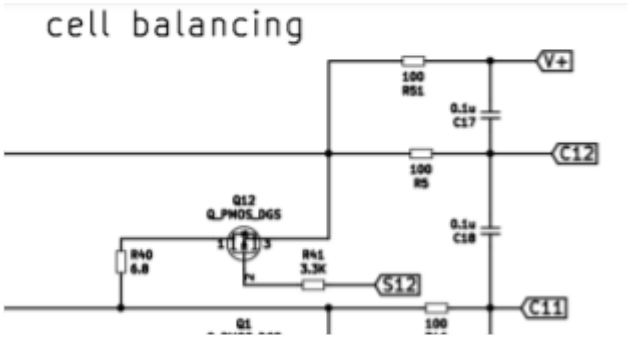


figura 23. Vista del circuito de balanceo de una celda

El recorte de esta figura es una sección del sistema ahora descrito encargado del balanceo y medición de voltajes. Con los valores escritos de los componentes

usados. Esta estructura se repite para las 12 celdas que se monitoriza el sistema tal y como se muestra en la siguiente figura.

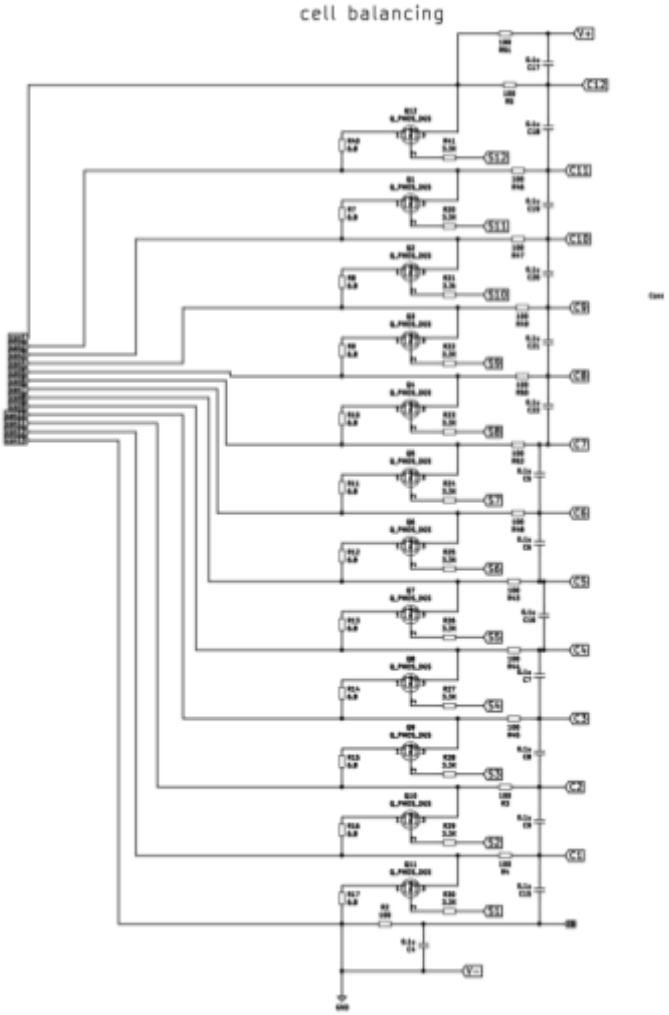


figura 24. Esquema completo balanceo

Circuito de alimentación

El circuito de alimentación está compuesto por todos aquellos componentes pasivos que la hoja de datos requiere para el funcionamiento correcto del mismo. Además por tratarse de un convertidor dc-dc el fabricante recomienda una colocación determinada de los componentes alrededor del integrado, la cual se ha tratado de tener en cuenta.

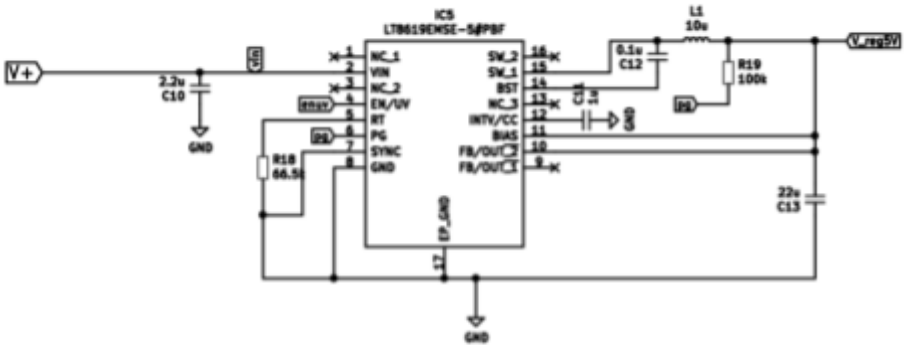


figura 25. Circuito acondicionamiento LT 8619

Configuración SPI ltc 6804-1

Como el circuito integrado en cuestión permite el uso de su propio protocolo de SPI conocido como isoSPI hay que usar el circuito que permita que este intgrado use SPI en vez del isoSPI. Para ello se ha de configurar de la manera indicada en la siguiente figura.

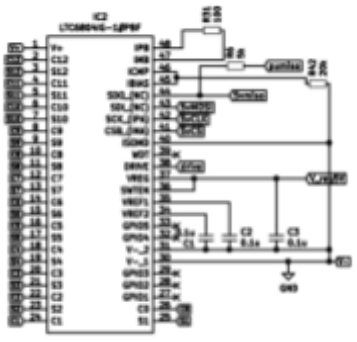


figura 26. Circuito LTC 6804-1

Además en esta figura se ve el circuito requerido para la alimentación del circuito integrado de manera segura una vez obtenidos los 5V del regulador de tensión.

Multiplexores y Medida de Temperatura

Para los multiplexores se usan 4 señales digitales de control salidas directamente del microcontrolador usado. Aunque el multiplexor funcione a 5V y las señales digitales del integrado sean a 3.3V se ha comprobado que los niveles lógicos del multiplexor acepten como HIGH los 3.3V de la señal digital del microcontrolador.

Para acondicionar la señal de los multiplexores se usan 6 resistencias una por cada termistor que va a estar conectado a la vez a las entradas analógicas, creando 6 partidores de tensión, para acondicionar la señal. tal y como se ve en la siguiente figura

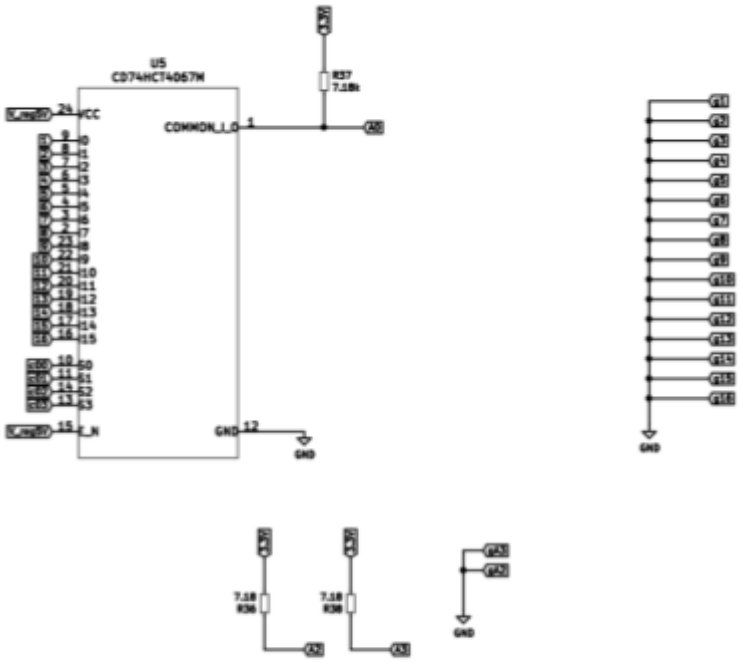


figura 27. Esquema de circuito de 18 termistores

Además la ventaja de tener multiplexadas las señales es que el consumo de corriente se disminuye en gran cantidad ya que no se está alimentando a 36 termistores a la vez si no que en este caso se está alimentando a 6

Comunicación

Como el circuito es idéntico al anterior simplemente se mostrará en la siguiente figura su forma

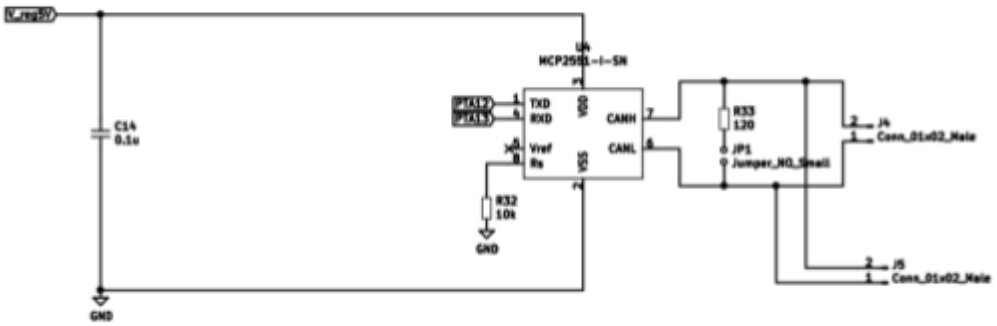


figura 29. Circuito comunicaciones.

Para ver el esquemático en su conjunto ver los anexos.

Diseño del Software

En este apartado vamos a hablar del código implementado para el funcionamiento de todos los circuitos de la toma de datos y de las comunicaciones.

Maestro

El maestro es el encargado de leer los mensajes y en función a eso tomar decisiones acerca de si cerrar un circuito o el otro, además se encarga de la adquisición de los datos del sensor de corriente , y de controlar las velocidades de los ventiladores. Además se encarga del cálculo de la estimación del estado de la carga de las celdas y el almacenamiento de datos para debugging en la tarjeta SD

Comunicaciones

Para las comunicaciones usamos la librerías de FlexCAN que sirven para poder comunicarse a través del bus can con las placas de desarrollo teensy. En concreto usamos la librería desarrollada por collin80 que permite comunicarse por can bus con funciones tan simples como las siguientes:

- begin(): sirve para inicializar el bus.
- available(): devuelve 0 si no hay ningún frame disponible, si lo hay devuelve el número de frames disponibles.
- write(): manda un buffer de 8 bytes.
- Read():lee el mensaje.

Los mensajes que se mandan se construyen de la siguiente forma:

- id: any value up to (2^(N+1) - 1) where N is either 11 or 29.
- len: dice el número de bytes contenidos en el mensaje.
- buf: es el contenido del mensaje

Esta librería incluye gran cantidad de funciones extra que por simplicidad y por no estimarse necesarias para este punto del desarrollo no han sido usadas. Para ver como ha sido codificado ver los anexos.

Control de los relés

Desde el punto de vista del circuito tal como fue explicado anteriormente las señales digitales salen directamente del microcontrolador, por lo que la mayor parte del trabajo en este aspecto es la lógica que está programada en el código.

Lo que desarrollé fue una función que tiene toda la lógica integrada. Como hay dos relés uno para el circuito de carga y otro para el de descarga vamos a diferenciar en la lógica para el caso de carga y descarga.

Para el caso de la carga asumimos la corriente como negativa y positiva para la descarga. Luego ya tenemos la forma para diferenciar entre estado de carga y descarga.

Para proteger las celdas se establecen unos valores de máxima corriente de carga y otro de máxima corriente de descarga. De esta manera se fuerza la desconexión del circuito de descarga si se da un valor menor que el límite de carga se abre el relé del circuito de carga. De la misma manera si se lee un valor mayor que el máximo se corta el de descarga. La función que implementa este comportamiento es *docurrent()* incluida en *funciones.h*

Por último estos relés también se pueden actuar para proteger las celdas en caso de que su voltaje esté por debajo o por encima de los límites establecidos. Así se puede prolongar la vida útil de las celdas. La función que implementa esta lógica es *casos()* incluida en *funciones.h*

Almacenamiento de datos

Se decidió implementar el almacenamiento de datos, pues la placa de desarrollo tiene la opción a usarla con un puerto micro sd incluida en la placa. Para poder almacenar los datos se usa la librería de arduino, SD.h. Esta librería permite con simples funciones crear archivos, leerlos y escribirlos.

Para inicializarlo hay que usar la función SD.begin() y decir que puerto se quiere usar, normalmente en arduino habría que decir los pines, pero para esta placa simplemente hay que poner BUILTIN_SDCARD.

También se debe crear una variable que almacene el nombre del archivo en que se va escribir que sea del tipo File, después hay que igualarla a SD.open("nombre_del_archivo", OPERACIÓN), usando en este caso como operación FILE_WRITE.

Después para rellenar el archivo es tan sencillo como escribir myfile.printf(), teniendo la misma estructura que el printf() de c, pudiendo rellenarlo exactamente de la misma forma.

Todas estas funcionalidades se usan en el main.cpp dentro del loop().

Cálculo de Corriente

Para el cálculo de la corriente se ha implementado una función que calcula el voltaje que le entra por la entrada analógica y aplica la ecuación que se aporta por el fabricante con los valores que da para los dos canales.

$$I_P = \left(\frac{5}{U_C} \times V_{out} - V_O\right) \times \frac{1}{G} \text{ with G in (V/A)}$$

Las sensibilidades para los sensores de 70V y 500V son respectivamente 26.7mv/A y 4mv/A. Luego en las ecuaciones Vo es el voltaje de offset que también se da en la tabla de características del sensor siendo para ambos 2.5V.

Luego para el cálculo de la corriente hay implementada una lógica bastante simple para que en el rango de -70A a 70A use la fórmula con la sensibilidad del canal uno y en los rangos de -500A a -70A y de 70A a 500A use la sensibilidad del canal dos. Esta lógica ha sido implementada usando condicionales para cada caso

Cálculo del SOC

Para el cálculo del estado de la carga primero se hicieron pruebas. La prueba de descarga a 800mA por celda nos dio los mejores resultados y es para la que se programó un código en matlab para realizar un ajuste de los datos.

Este programa manipulaba los datos del fichero .CSV que extraemos de la carga electrónica para poder incluir el valor de la resistencia interna, que ya explicaremos cómo se obtuvo en el apartado de pruebas. Una vez puesta la resistencia interna en todas las filas calculamos con la corriente, la tensión y la resistencia interna la tensión en circuito abierto de la celda con la siguiente fórmula:

$$OCV = V + I \cdot R_{int}$$

Esta fórmula es tan sencilla que lo que se hace es tener en cuenta la caída de tensión que se produce en la celda debido a la corriente que pasa por ella. De esta

forma acabamos teniendo una aproximación de la tensión que tendría si la celda estuviese en circuito abierto.

Después con esto calculado comparamos el OCV con la capacidad de la celda restante que fue obtenida restando a la capacidad total especificada por el fabricante la energía consumida y dividiéndola por la capacidad total:

$$SOC = \frac{C_t - C_a}{C_t}$$

Con estas dos cosas calculadas ya tenemos nuestra curva del SOC vs OCV que usando la función fit de matlab obtenemos la curva con el ajuste polinómico al grado que queremos. Esta vez se uso un ajuste a una ecuación de grado 9 pues es un grado que se ha visto comúnmente en este tipo de aplicaciones y dio un ajuste con 5% de error.

La ecuación polinómica obtenida fue la siguiente:

$$SOC = p_1{}^9 + p_2 x^8 + p_3 x^7 + p_4 x^6 + p_5 x^5 + p_6 x^4 + p_7 x^3 + p_8 x^2 + p_9 x + p10$$

Donde las constantes son:

x = ocv

p1 = -15.94

p2 = 481.8

p3 = -6447

p4 = 5.013e+04

p5 = -2.496e+05

p6 = 8.254e+05

p7 = -1.813e+06

p8 = 2.549e+06

p9 = -2.083e+06

p10 = 7.539e+05

Así que esta última ecuación y las constantes es lo que fue implementado en el microcontrolador para calcular el SOC en función del estado de la carga, junto con el cálculo previo de la tensión de circuito abierto, siendo así capaz de determinar el estado de la carga del pack.

Esclavo

En este apartado vamos a hablar de las funciones implementadas en el código de los MMUs. El código de los esclavos básicamente gira alrededor de la comunicación con el LTC 6804-1, la lectura de las temperaturas y el envío de los mensajes al maestro con los datos de voltaje y temperatura.

Comunicación con el LTC 6804-1

Para la comunicación con el LTC 6804-1 se ha usado la librería que aporta analog devices para el control del dispositivo. Para poder usarlas hay que incluir en el proyecto los archivos que se descargan de la página del producto.

Estos archivos son LT_SPI.h, LT_SPI.cpp, LTC68041.h y LTC68041.cpp además de incluir Linduino.h. Además para entender el funcionamiento de las librerías Analog Devices aporta un ejemplo código para programar los microcontroladores para comunicarse y configurar debidamente el integrado.

Las funciones que se usan son las siguientes:

- LTC6804_initialize(): inicializa el integrado.
- wakeup_idle(): esta función sirve para poner en estado activo al integrado antes de solicitar algo.
- LTC6804_adcv(): enciende la conversión analógica digital.
- LTC6804_rdcv():lee los últimos valores de voltaje guardados por integrado
- LTC6804_wrcfg(): Esta función escribe el vector de configuración en el integrado

Para el funcionamiento del programa te hacen usar 4 vectores:

- uint16_t cell_codes[TOTAL_IC][12]: vector que almacena los voltajes de las celdas que monitoriza.
- uint16_t aux_codes[TOTAL_IC][6]: almacena los valores de los datos de los GPIO del integrado.
- uint8_t tx_cfg[TOTAL_IC][6]: almacena los valores de la configuración que se va cargar en el integrado.
- uint8_t rx_cfg[TOTAL_IC][8]: almacena los valores de la configuración que se acaba de leer del integrado.

Para leer los datos de voltaje es tan simple como mandar a que se produzca la conversión analógica digital y después leer los datos con sus respectivas funciones.

Pero para poder gestionar el balanceo de la carga de las celdas hay que manipular la configuración del integrado escribiendo en el vector del tx_cfg[][]. Para saber qué bits de cada posición del vector debemos manipular hay que mirar la tabla aportada por Analog Devices en el datasheet del integrado.

REGISTER	RD/WR	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CFGR0	RD/WR	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	REFON	SWTRD	ADCOPT
CFGR1	RD/WR	VUV[7]	VUV[6]	VUV[5]	VUV[4]	VUV[3]	VUV[2]	VUV[1]	VUV[0]
CFGR2	RD/WR	VOV[3]	VOV[2]	VOV[1]	VOV[0]	VUV[11]	VUV[10]	VUV[9]	VUV[8]
CFGR3	RD/WR	VOV[11]	VOV[10]	VOV[9]	VOV[8]	VOV[7]	VOV[6]	VOV[5]	VOV[4]
CFGR4	RD/WR	DCC8	DCC7	DCC6	DCC5	DCC4	DCC3	DCC2	DCC1
CFGR5	RD/WR	DCTO[3]	DCTO[2]	DCTO[1]	DCTO[0]	DCC12	DCC11	DCC10	DCC9

figura 30. Tabla de referencia de bits del vector de configuración

En esta tabla vemos que en el vector de configuración los valores de los vectores 3 y 4 son los que hay que manipular para poder encender o apagar el balanceo de una celda. Si esa bit de esa posición del vector está a 1 significa que está balanceando, si está a 0 no. Luego son estos bits los que se están manipulando constantemente para balancear las celdas.

Lógica de balanceo

La lógica está implementada en funcione.h y se llama balanceo().La lógica de balanceo implementado es todo o nada, compara la tensión de cada celda con el valor de voltaje de la celda de menor valor y si la diferencia es mayor que la deseada por el usuario se rellenará un entero sin signo con un uno en el bit de la posición de la celda.

Esto se hace para pasar este entero sin signo ya relleno al vector de configuración para mandarlo directamente al integrado.

Lectura de temperaturas

Como la lectura de todas las temperaturas depende de dos multiplexores controlados por 8 señales digitales. Entonces para controlar los multiplexores se usan dos grupos de 4 bucles for() anidados para cambiar el valor de cada una de las señales de control hasta leer los 16 termistores asociados a cada multiplexor.

Para transformar los valores de voltaje leídos por el conversor analógico digital debemos saber el valor máximo de tensión que lee este adcv, en este caso 3.3V y la cantidad de bits que tiene de resolución, 1024 en este caso. Así podemos obtener el valor de la tensión que lee. Estos valores son almacenados en un vector con 36 posiciones para después convertirlos en valores de temperatura en grados celsius.

Para ello usamos los datos que se aportan en el datasheet del termistor y aplicamos la siguiente ecuación:

$$T = \frac{B}{ln(\frac{R_t}{A})} - 273.15$$

donde Rt es la resistencia del termistor obtenida de la siguiente ecuación :

$$R_t = \frac{V_m \cdot R_P}{3.3V - V_m}$$

donde en estas ecuaciones cada variable significa:

- Rt: resistencia del termistor
- Vm: voltaje medido
- Rp: valor de la otra resistencia que conforma el partidor de tensión
- A: coeficiente A del termistor
- B: coeficiente B del termistor

Pruebas

En este apartado se va a hablar de las pruebas que se han realizado. Estas se han hecho para comprobar el funcionamiento de los sistemas a implementar o para obtener datos para caracterizar las celdas y determinar la curva de estado de la carga frente a la tensión de circuito abierto.

Obtención curva SOC VS OCV

Para la obtención de las curvas se hizo uso de una carga electrónica aportada por el departamento de electrónica de la Universidad de La Laguna. Las cargas electrónicas son dispositivos que permiten la descarga de equipos electrónicos, simulando ser una resistencia.

figure 31. Fotografía prueba de descarga de la celda

En este caso la carga electrónica tiene un modo para probar baterías, que fue el usado en esta prueba. Este modo permite descargar la batería a corriente constante y te va devolviendo la energía que ha consumido durante la descarga, junto con el voltaje en ese instante. Esto hace al modo de prueba de baterías perfecto para la obtención de las curvas SOC frente OCV.

Durante las pruebas se realizaron descargas a diferentes corrientes, resultando la de 800mA la más significativa pues podríamos llegar a voltajes más bajos sin que la celda corriese peligro. Durante las descargas de mayor corriente se monitorizó la temperatura de la celda con la cámara de calor que se le había prestado al equipo para prevenir cualquier riesgo.

El montaje para realizar la prueba fue muy simple, soldamos por punto dos pletinas de níquel a cada borne, esas pletinas tenían soldadas dos cables que

daban a un conector XT60 para prevenir conexiones erróneas en la carga electrónica. La carga electrónica tenía conectado un pen drive donde se almacenaban los datos de la prueba que eran guardados como un archivo del tipo .csv.

Normalmente este tipo de pruebas se realizan en condiciones muy controladas, manteniendo constante la temperatura, la humedad y con sistemas muy complejos de gran precisión. Pero si se busca información acerca de la caracterización de celdas con curva SOC vs OCV se muestran datos y comparativas acerca del error inducido en este método para determinar el estado de la carga quedando claro que las condiciones ambientales apenas tienen influencia

El procedimiento para realizar la prueba era realizar una pequeña descarga para obtener antes de comenzar la descarga completa la resistencia interna de la celda a probar. Esta la conseguimos mirando la tensión que tenía antes de la descarga, y con la corriente que pasó por ella durante la descarga y la tensión a la que bajó. Obteniendo dicho valor de la siguiente fórmula:

$$R_{int} = \frac{OCV - V_d}{I}$$

Donde:

OCV: tensión de circuito abierto.

Vd: voltaje durante la descarga.

I: intensidad durante la descarga.

Una vez conocido el valor de la resistencia interna de la celda ya se puede comenzar la descarga completa de la celda. Los valores bajos de corriente eran ideales porque además de permitir descargar la celda hasta voltajes menores, hace que la medida de tensión en la carga electrónica sea más correcta ya que las pérdidas de tensión en los cables son menores.

Una vez terminada la prueba estos datos se deben tratar, para ellos se usó un script de matlab propio. Para obtener el ajuste de la curva se usó la función fit para conseguir una ecuación polinómica que se ajustase a la curva que describen los datos. Consiguiendo de esta forma tener una ecuación que nuestro

microcontrolador fuese capaz de calcular para determinar el estado de la carga.

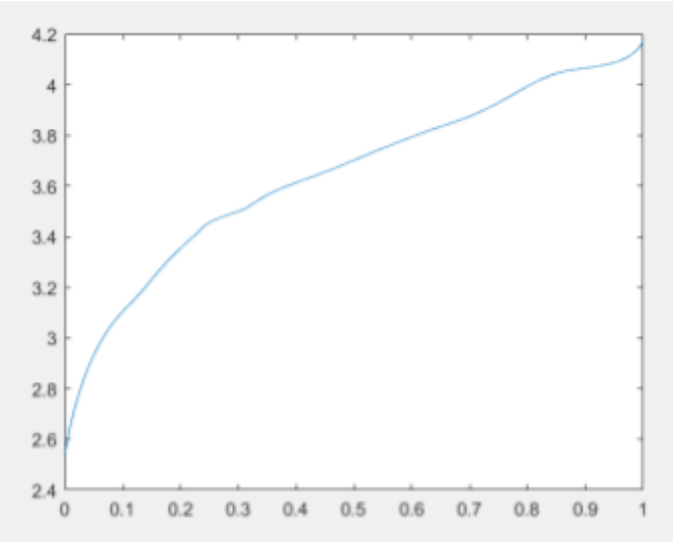


figura 32. Curva OCV vs SOC obtenida experimentalmente

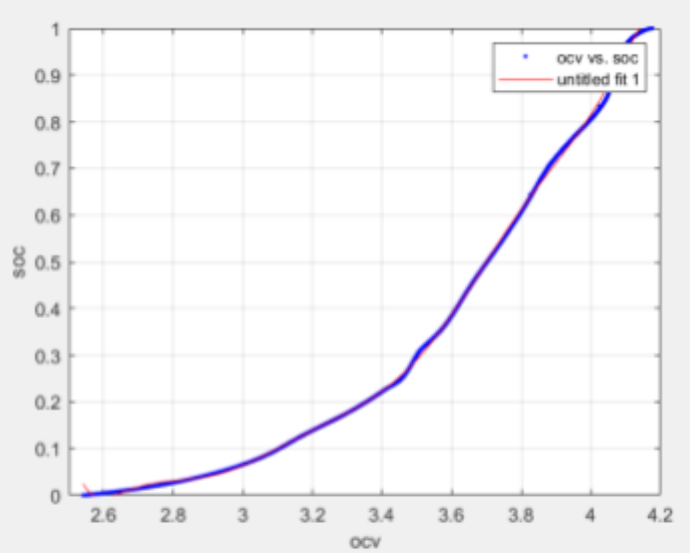


figura 33. Curva OCV vs SOC y curva sacada con la ecuación ajustada

Pruebas de Comunicaciones

Para las pruebas de comunicaciones se montaron las dos placas, la placa del maestro conectada a la fuente y con la tarjeta micro sd insertada, y la placa del esclavo conectada al portátil.

Para la comprobación se cargó en el esclavo un código que solo mandase un mensaje. Se dejó este código funcionando unos diez segundos y se extrajo la tarjeta sd.

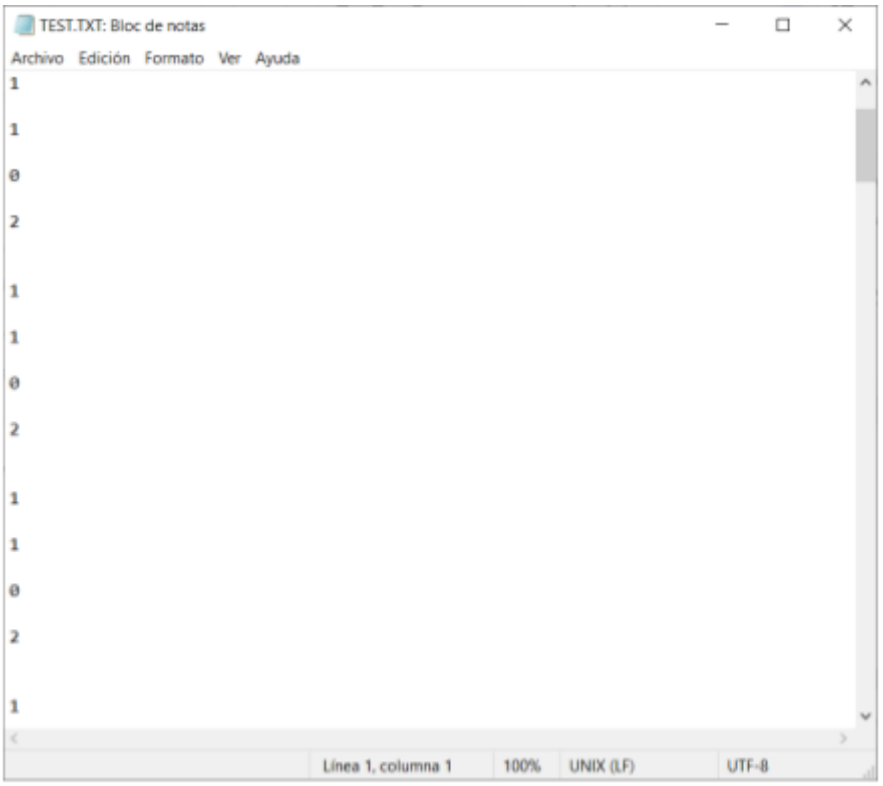


figura 34. Captura datos almacenados en la tarjeta micro SD

De esta forma podemos comprobar que no solo los mensajes se trasmiten, sino que también los datos se guardan de la forma deseada en la tarjeta de memoria

Prueba de fiabilidad del regulador de voltaje del esclavo

A pesar de que se trata de un convertidor dc-dc, y de que se espera de ellos altos niveles de eficiencia se probó alimentar durante la placa durante 3 horas a la tensión más alta que era capaz con la fuente que disponía. Esta tensión fue de 32V.

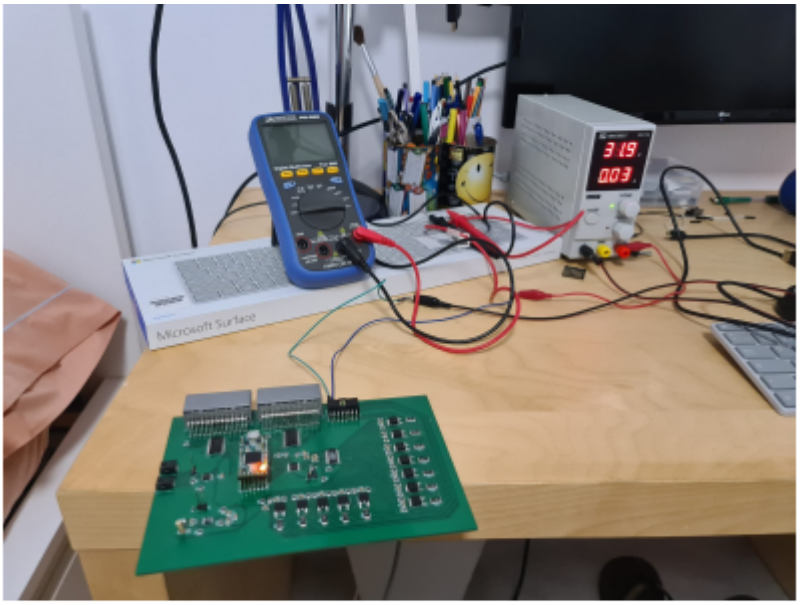


figura 35. Prueba de funcionamiento de tres horas

El resultado fue el esperado pero se estimó necesario comprobar debido a la ubicación en la que iba a estar el sistema, siendo crítico que este funcione a la perfección.

Prueba lectura y carga de configuraciones

Como última prueba se realizó la carga y lectura de configuraciones en el integrado LTC 6804-1. Para ello se usó el script aportado por el fabricante para cargar y posteriormente leer la configuración que se había cargado y comprobar que funcionaba correctamente.

De esta manera se confirmó que la placa funciona correctamente y que el integrado se comporta correctamente. Confirmando que la placa funciona según lo esperado.

Futuros estudios

En este apartado se hablará de futuras ramas de trabajo y partes en las que se deberá seguir trabajando dentro de este proyecto para conseguir un sistema completamente operacional.

Mejora del código

El código presentado en este trabajo se encuentra en una etapa muy temprana del desarrollo. Sería interesante añadir en todos los mensajes un código de tiempo y sincronizar todos los esclavos para que usen al mismo tiempo.

Por otra parte también hay que mejorar el desempeño de las funciones de lectura de temperatura que todavía se pueden hacer para que no lean todo en la misma vuelta del bucle y así permitir un desarrollo de las acciones más fluidas y más rápida, ya que en el estado actual todavía tiene algo de latencia.

Diseño de placas definitivas

El estado actual del proyecto está con unas placas de evaluación que permiten el desarrollo del software para estas placas de desarrollo. El maestro por tamaño podría perfectamente quedarse así, en cambio los esclavos por tamaño para poder hacerlos más manejables y reducir costes convendría hacer una placa que no contuviese la placa de desarrollo.

Conclusiones

El objetivo de este trabajo de fin de grado era el diseño de un sistema de gestión de baterías. Este se puede decir que ha sido cumplido pues se puede leer y escribir valores en el integrado que se encarga del balanceo, las comunicaciones funcionan, se puede leer los valores de temperatura correctamente y los datos se almacenan debidamente.

Es cierto que no se ha podido llegar a probar el sistema de gestión baterías en un pack de baterías real. Luego solo se ha podido montar un solo esclavo por motivos de costes lo que significa que todavía hay que montar un sistema completo y probar las comunicaciones y que el maestro pueda leer todos los mensajes.

Tras las satisfactorias pruebas de descarga de las baterías se ha conseguido determinar una curva que se ajuste a la descarga de las celdas. De esta forma ahora podemos determinar el valor del estado de la carga de forma satisfactoria.

Se ha conseguido que gracias a los diseños de hardware todos los requisitos de la normativa se cumplan y si en algún caso en pruebas posteriores no los cumpliese, solo habría que mejorar en aspecto de software ya que el hardware está todo cumpliendo las normativas.

Luego como conclusiones finales podemos decir que este trabajo nos ha permitido profundizar en un mundo que de otra forma no hubiera llegado a conocer. En parte ha sido gracias al equipo de la universidad que me interese por este tema después tener que configurar y informarme acerca de estos dispositivos, y lo elegí como tema del trabajo de fin de grado por ser un trabajo muy completo en el que poder aplicar todos los conocimientos de la carrera en un ámbito que ahora está en auge con la movilidad eléctrica

Referencias bibliográficas.

1. Farmann A, Sauer DU. A study on the dependency of the open-circuit voltage on temperature and actual aging state of lithium-ion batteries. Journal of power sources. 2017;347:1-13. https://dx.doi.org/10.1016/j.jpowsour.2017.01.098. doi: 10.1016/j.jpowsour.2017.01.098.

2. Zhu Q, Xiong N, Yang M, Huang R, Hu G. State of charge estimation for lithium-ion battery based on nonlinear observer: An H∞ method. Energies (Basel). 2017;10(5):679. https://search.proquest.com/docview/1910607591. doi: 10.3390/en10050679.

3. Narayanaswamy S. Design, modeling and implementation of distributed architectures for modular battery packs. ; 2018.

4. Bouzón Pousa L. BATTERY MANAGEMENT SYSTEM (BMS) PARA UN COCHE DE FORMULA STUDENT volumen I memoria autor. .

Anexos

Maestro Main.cpp

```
#include <Arduino.h>
#include <FlexCan.h>
#include <kinetis_flexcan.h>
#include <SD.H>
#include <SPI.H>
#include <string.h>
#include <funciones.h>
#include <math.h>
//definimos la funciones
void casos (int buf0,float buf2,int tvent1, int tvent2,int tcort,int pinpwm,int chenable,int disenable,float vmin,float vmax);
//definimos las variable para can bus

FlexCAN CANbus(1000000, 0);
;

static CAN_message_t msg;
static uint8_t hex[17] = "0123456789abcdef";


//definimos la variable para escribir en la sd
File myFile;

//definimos las variables de pines para el funcionamiento del maestro
int pinpwm= 23; //pin al que van conectados los ventiladores
int chenable= 36; //pin al que va conectado el charge enable
int disenable= 37; //pin al que va conectado el discharge enable
int curr1= A8;
int curr2= A7;
//definimos las variables de control del sistema
int tvent1 = 30; //temperatura a la que empiezan los ventiladores a v lenta
int tvent2 = 40; //temperatura a la ventiladores 100%
int tcort = 60; //temperatura maxima por normativa
float corrientech = 5; //corriente maxima de carga
float corrientedch = 10;//corriente maxima de descarga
float vmin = 2.5; //voltaje minimo para la celda
float vmax = 4.1; //voltaje maximo para la celda
int vvent=3;
float intern_resis=0.06;//valor calculado previamente a montar el pack
float prev_soc=1;
//varibles generales
float c=0;
float carga;
```

```
int sip=0;

void setup() {
  // put your setup code here, to run once:
  //init can bus
  CANbus.begin();

  delay(1000);
  Serial.begin(9600);
  pinMode(13,OUTPUT);
  digitalWrite(13,HIGH);
  Serial.println("Can Receiver Initialized");

  //inicializamos la sd

  if (!SD.begin(BUILTIN_SDCARD)){
    Serial.println("initialization failed");
    while(1);

  }
  Serial.println("initialization done");
//inicializamos los pines
pinMode(pinpwm,OUTPUT);
pinMode(chenable,OUTPUT);
pinMode(disenable,OUTPUT);

}

void loop()
{


    digitalWrite(13,LOW);
    delay(1000);
    digitalWrite(13,HIGH);
    delay(1000);
    c=corriente(A8,A7);
    docurrent(chenable,disenable,corrientech,corrientedch,c);

    if(CANbus.available())
    {
      CANbus.read(msg);
      prev_soc=carga;
      carga=soc(msg.buf[0],msg.buf[2],intern_resis,c,prev_soc);
  //    Serial.print("CAN bus 0: "); hexDump(8, msg.buf);
  myFile = SD.open("test.txt",FILE_WRITE);
```

```
if(myFile){
  Serial.print("Writing to test.txt...");
    //imprimo el id del mensaje
    myFile.printf("%d ",int(msg.id),HEX);


    /*imprimo el buf 0 en el que se dira que tipo de dato es
      *01 Datos de voltaje
      *02 Datos de Temperatura
    */
    myFile.printf("%f ",float(msg.buf[0]));


    /* en el buf 1 ira reflejado la posicion del termistor o de la celda
      en la que se mide la temperatura o el voltaje con el fin de poder
      reflejarla por serial y guardarlo en los datos de la SD
    */

    myFile.printf("%f ",float(msg.buf[1]));

    /* en el buf 2 se guardara los valores mencionados*/


    myFile.printf("%f",float(msg.buf[2]));

    myFile.printf("\n");
    myFile.printf("%f A",c);
    myFile.printf("\n");
    myFile.printf("%f soc\n",carga);
    // close the file:
    myFile.close();
    Serial.println("se ha guardado:\n ");
      Serial.printf("%d ",int(msg.id),HEX);

      Serial.printf("%f ",float(msg.buf[0]));

      Serial.printf("%f ",float(msg.buf[1]));

      Serial.printf("%f \n",float(msg.buf[2]));

  }

}else if(!CANbus.available()){

  digitalWrite(chenable,HIGH);
  digitalWrite(disenable,HIGH);
```

}

casos(int(msg.buf[0]),float(msg.buf[2]),tvent1,tvent2,tcort,pinpwm,chenable,disena
ble, vmin, vmax,vvent);

}

Funciones.h

```
/*esta funcion se encarga de tomar deciciones acerca de los datos que llegan por el bus*/
void casos (int buf0,float buf2,int tvent1, int tvent2,int tcort,int pinpwm,int chenable,int disenable,float vmin,float vmax,int vvent)
{

//crea el arbol de accion en funcion de si el dato es de voltaje o de temperatura
switch (buf0)
{
/*ejecuta comprobacion de temperatura y decide si cortar o no y cuando y como activar los ventiladores por la señal pwm*/
case 2:
   if (buf2>=tvent1&&buf2<=tvent2){

     analogWrite(pinpwm,vvent*25);

   }else if(buf2>=tvent2){

     analogWrite(pinpwm,100);

   }
   if(buf2>=tcort){

     digitalWrite(chenable, HIGH);
     digitalWrite(disenable, HIGH);

   }
case 1:
  if(buf2<vmin){

    digitalWrite(disenable,HIGH);

  }
  if(buf2>vmin){

    digitalWrite(chenable,HIGH);

  }

default:

break;


}
```

```
}

/*ESTA FUNCION SE ENCARGA DE CALCULAR EL VALOR DE LA CORRIENTE EN FUNCION DE LO
QUE RECIBE POR EL PUERTO ANALOGICO Y DEVUELVE EL VALOR*/
//variables para la funcion
int supplyv=5;
float uc=5;
float voffset=2.5;
float g1=26.67/1000;
float g2=4/1000;
float corrchan1=0;
float corrchan2=0;
//empezamos la funcion
float corriente(int pin1,int pin2){

  float mpin1=analogRead(pin1);
  corrchan1=((5/5)*mpin1-voffset)*(1/g1);

  float mpin2=analogRead(pin2);
  corrchan2=((5/5)*mpin2-voffset)*(1/g2);

  if(corrchan1> -75 && corrchan1< 75){
    return corrchan1;
  }else{

    return corrchan2;
  }

}
```

```
/*ESTA FUNCION DECIDE COMO ACTUAR PARA LOS DISTINTOS CASOS SEGUN EL VALOR DE LA
CORRIENTE QUE CIRCULA POR EL TS
*/
int estado=0;
void docurrent(int chenable,int disenable,int corrientech,int corrientedch, float corriente){

if(corriente>0){
estado=1;
}else{
  estado=0;
}

switch(estado)
{
   /*en este caso se supone que esta en descarga el circuito,luego no permitimos
   que por el conector de carga entre corriente asi que forzamos a que el charge
   enable este cortado*/

   case 1://DESCARGA
     digitalWrite(chenable,HIGH);
     if(corriente>=corrientedch){

       digitalWrite(disenable,HIGH);

     }else{

       digitalWrite(disenable,LOW);

     }
   case 0://CARGA

     digitalWrite(disenable,HIGH);
     if(corriente>=corrientech){
       digitalWrite(chenable,HIGH);
     }else{

       digitalWrite(chenable,LOW);

     }
}

}
```

```
//calculo del estado de la carga
float soc (uint8_t msg0,uint8_t msg2,float intern_resistance, float corriente,float prev_soc){


if(msg0==1){



float ocv= (float)msg2*0.0001 + corriente*intern_resistance;

/*Ajuste a curva obtenida experomentalmente para calcular el estado de la carga en funcion de la tension en circuito abierto*/
 float    x  =        ocv  ;
 float    p1 =     -15.94  ;
 float    p2 =      481.8  ;
 float    p3 =      -6447  ;
 float    p4 =   5.013e+04  ;
 float    p5 =  -2.496e+05  ;
 float    p6 =   8.254e+05  ;
 float    p7 =  -1.813e+06  ;
 float    p8 =   2.549e+06  ;
 float    p9 =  -2.083e+06  ;
 float    p10 =   7.539e+05 ;


float estado_carga = p1*pow(x,9) + p2*pow(x,8) + p3*pow(x,7) + p4*pow(x,6) + p5*pow(x,5) + p6*pow(x,4) + p7*pow(x,3) + p8*pow(x,2) + p9*x + p10;



return estado_carga;
}else{

return prev_soc;

}


}
```

Main.cpp esclavos

```
#include <Arduino.h>
#include <stdint.h>
#include <FlexCan.h>
#include <kinetis_flexcan.h>
#include <SoftwareSerial.h>
#include "LT_SPI.h"
#include "LTC68041.h"
#include <linduino.h>
#include "average.h"
#include "funcione.h"

#define ide 0x1


//CONFIGURACION DEL CAN

FlexCAN CANbus(1000000, 0);
;

static CAN_message_t msg;


//CONFIGURACION DEL LTC68041
#define TOTAL_IC  1              // Number of ICs in the isoSPI network LTC6804-2 ICs must be addressed in ascending order starting at 0.

/***** Pack and sensor characteristics *****/
const float CELL_BALANCE_THRESHOLD_V = 3.3;     // Cell balancing occurrs when voltage is above this value

//DEINICION DE PINES
int tempPins[] = {A2};              // Array of arduino pins used for temperature sensor inpout (REVISAR Y PONER)

//VARIABLES FOR TRACKING CELL VOLTAGES
int cellMax_i;                    // Temporary variable for holding index of cell with max voltage
int cellMin_i;                    // Temporary variable for holding index of cell with min voltage
float  cellMin_V;                   // Temporary variable for holding  min measured cell voltage
float  cellMax_V;                   // Temporary variable for holding  max measured cell voltage
float minV1 ;
float maxV1 ;
```

```
int cellmax[]={0,0};//en la posicion 1 se almacena el numero de la celda
                    //en la posicion 2 se almacena el valor del voltaje

int cellmin[]={0,100};//en la posicion 1 se almacena el numero de la celda
                    //en la posicion 2 se almacena el valor del voltaje
int diferencialimite=0;//diferencia a la que se empieza a balancear
int diferenciamax=0;//diferenciamax actual
int diferencia=0;//diferencia actual de voltaje
uint16_t balancear=0;// si se debe balancear una celda o no
//si esta a 1 se balancea
//si esta a 0 no
uint16_t vuv=2.6;
uint16_t vov=4.1;
//variables para las temperaturas
float temps[35];
float coefB=3740;
float coefA=0.16655;
float r25=47000;
//nombre de los pines
int analog[]={A0,A1,A2,A3,A4,A5};
int control1[]={17,16,15,14};
int control2[]={18,19,20,21};


/*****************************************************
  Global Battery Variables received from 6804 commands
  These variables store the results from the LTC6804
  register reads and the array lengths must be based
  on the number of ICs on the stack
 *****************************************************/
 uint16_t cell_codes[TOTAL_IC][12];
 /*!<
   The cell codes will be stored in the cell_codes[][12] array in the following format:
    |  cell_codes[0][0]| cell_codes[0][1] |  cell_codes[0][2]|    .....     |  cell_codes[0][11]|
cell_codes[1][0] | cell_codes[1][1]|  .....   |

|------------------|------------------|------------------|--------------|-------------------|-----------------
----|-----------------|----------|
   |IC1 Cell 1        |IC1 Cell 2       |IC1 Cell 3       |    .....    |  IC1 Cell 12     |IC2 Cell 1        |IC2
Cell 2       | .....    |
 ****/

 uint16_t aux_codes[TOTAL_IC][6];
 /*!<
   The GPIO codes will be stored in the aux_codes[][6] array in the following format:
       |    aux_codes[0][0]|  aux_codes[0][1]  |    aux_codes[0][2]|    aux_codes[0][3]|
aux_codes[0][4]|  aux_codes[0][5]| aux_codes[1][0] |aux_codes[1][1]|  .....    |
```

```
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------
|-----------------|---------------|-----------|
   |IC1 GPIO1        |IC1 GPIO2        |IC1 GPIO3        |IC1 GPIO4        |IC1 GPIO5        |IC1
Vref2       |IC2 GPIO1       |IC2 GPIO2      |  .....    |
 */

 uint8_t tx_cfg[TOTAL_IC][6];
 /*!<
  The tx_cfg[][6] stores the LTC6804 configuration data that is going to be written
   to the LTC6804 ICs on the daisy chain. The LTC6804 configuration data that will
be
    written should be stored in blocks of 6 bytes. The array should have the
following format:
   |  tx_cfg[0][0]| tx_cfg[0][1] |  tx_cfg[0][2]|  tx_cfg[0][3]|  tx_cfg[0][4]|  tx_cfg[0][5]|
tx_cfg[1][0] |  tx_cfg[1][1]|  tx_cfg[1][2]|  .....    |

|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--
-----------|--------------|-----------|
   |IC1 CFGR0      |IC1 CFGR1      |IC1 CFGR2      |IC1 CFGR3      |IC1 CFGR4      |IC1 CFGR5
|IC2 CFGR0      |IC2 CFGR1      | IC2 CFGR2     |  .....    |
 */

 uint8_t rx_cfg[TOTAL_IC][8];
 /*!<
   the rx_cfg[][8] array stores the data that is read back from a LTC6804-1 daisy
chain.
   The configuration data for each IC  is stored in blocks of 8 bytes. Below is an
table illustrating the array organization:

|rx_config[0][0]|rx_config[0][1]|rx_config[0][2]|rx_config[0][3]|rx_config[0][4]|rx_con
fig[0][5]|rx_config[0][6]  |rx_config[0][7] |rx_config[1][0]|rx_config[1][1]|  .....    |

|---------------|---------------|---------------|---------------|---------------|---------------|-----------
------|---------------|---------------|---------------|-----------|
   |IC1 CFGR0      |IC1 CFGR1      |IC1 CFGR2      |IC1 CFGR3      |IC1 CFGR4      |IC1 CFGR5
|IC1 PEC High     |IC1 PEC Low     |IC2 CFGR0      |IC2 CFGR1      |  .....    |
 */



 /*!*********************************************************************
  \brief  Inititializes hardware and variables
 *********************************************************************/

 //inicializamos las funciones para el ltc6804
 void init_cfg();
```

```
  void print_cells();
  void print_config();
  void print_rxconfig();
  void serial_print_hex(uint8_t data);

void setup() {
//inicializamos los pines
for(int i=0;i<4;i++){
  pinMode(control1[i],OUTPUT);
  pinMode(control2[i],OUTPUT);
}

 Serial.begin(9600);
 CANbus.begin();
 delay(1000);
 Serial.begin(9600);
 pinMode(13,OUTPUT);
 digitalWrite(13,HIGH);
 Serial.println("Can Receiver Initialized");

 msg.id = ide;
 msg.len = 4;
 msg.buf[0]=01; // 1 SI ES VOLTAJE 2 SI ES TEMPERATURA
 msg.buf[1]=00; // POSICION EN LA QUE SE MIDE
 msg.buf[2]=2.8; //VALOR DEL MENSAJE
 msg.buf[3]=0; //1 SI ESA CELDA ESTA BALANCEANDO 0 SI NO

 //INICIALIZAMOS EL LTC68041
 LTC6804_initialize();
 init_cfg();
 config_vlimites(tx_cfg,vuv,vov);
 delay(1000);
}

void loop() {
  digitalWrite(13,HIGH);



//LEEMOS LOS VOLTAJES DE LAS CELDAS
wakeup_idle();
LTC6804_adcv();//HACE LA CONVERCION AD Y RELLENA LOS REGITROS DE LAS
CELDAS
delay(10);
wakeup_idle();
```

```
int error=LTC6804_rdcv(0,TOTAL_IC,cell_codes);//leemos los valores de los registros

if(error == -1)
{
  Serial.println("A PEC error was detected in the received data");
}

//SE MANDAN MENSAJES CON LOS VOLTAJES DE LAS CELDAS Y LA POSICION DE LA MISMA
  msg.buf[0]=1;
for(int i=0;i<=12;i++){

  msg.buf[1]=i;
  msg.buf[2]=int(cell_codes[1][i]);

  if(CANbus.available())
  {
    CANbus.write(msg);
    Serial.println(int(msg.id),HEX);
    Serial.println(int(msg.buf));
    Serial.println("\n");

  }else{
    Serial.printf("bus no disponible %d\n",i);
  }
}
//BALANCEO DE CELDAS
// sacamos las celdas de menor voltaje y mayor voltaje
Serial.println("sip1");
mayor(cell_codes,cellmax);
Serial.println("sip2");
menor(cell_codes,cellmin);
Serial.println("sip3");
diferenciamax=cellmax[2]-cellmin[2];
Serial.println("sip4");
/*si la diferencia max leida es mayor que la limite se mira si hay mas celdas
que balancear*/
if(diferenciamax>diferencialimite){
balanceo(cell_codes,balancear,diferencialimite,cellmin,CELL_BALANCE_THRESHOLD_V);
  }
  Serial.println("sip5");
//una vez sabido que celdas hay que balancear se carga en los bits de config
control_balanceo(tx_cfg,balancear);
Serial.println("sip6");
//ESCRIBIMOS TODA LA CONFIGURACION CARGADA
```

```
Serial.println("sip7");
wakeup_idle();
Serial.println("sip8");
LTC6804_wrcfg(TOTAL_IC, tx_cfg);
Serial.println("sip9");
//leemos todas las temperaturas
temperaturas(temps,coefA,coefB,r25);
Serial.println("sip10");
digitalWrite(13,LOW);
//mandamos por el bus las temperaturas
msg.buf[0]=2;
for(int i=0; i<36;i++){
   msg.buf[1]=i;
   msg.buf[2]=(uint8_t)temps[i];
   if(CANbus.available()){
     CANbus.write(msg);
   }
   else{
     Serial.printf("bus no disponible %d\n",i);
   }
 }
Serial.println("sip11");
}



void init_cfg()
{
 for (int i = 0; i<TOTAL_IC; i++)
 {
   tx_cfg[i][0] = 0xFE;
   tx_cfg[i][1] = 0x00 ;
   tx_cfg[i][2] = 0x00 ;
   tx_cfg[i][3] = 0x00 ;
   tx_cfg[i][4] = 0x00 ;
   tx_cfg[i][5] = 0x00 ;
 }
}

void print_cells()
{


 for (int current_ic = 0 ; current_ic < TOTAL_IC; current_ic++)
 {
   Serial.print(" IC ");
```

```
    Serial.print(current_ic+1,DEC);
    for (int i=0; i<12; i++)
    {
      Serial.print(" C");
      Serial.print(i+1,DEC);
      Serial.print(":");
      Serial.print(cell_codes[current_ic][i]*0.0001,4);
      Serial.print(",");
    }
    Serial.println();
  }
  Serial.println();
}

void print_config()
{
  int cfg_pec;

  Serial.println("Written Configuration: ");
  for (int current_ic = 0; current_ic<TOTAL_IC; current_ic++)
  {
    Serial.print(" IC ");
    Serial.print(current_ic+1,DEC);
    Serial.print(": ");
    Serial.print("0x");
    serial_print_hex(tx_cfg[current_ic][0]);
    Serial.print(", 0x");
    serial_print_hex(tx_cfg[current_ic][1]);
    Serial.print(", 0x");
    serial_print_hex(tx_cfg[current_ic][2]);
    Serial.print(", 0x");
    serial_print_hex(tx_cfg[current_ic][3]);
    Serial.print(", 0x");
    serial_print_hex(tx_cfg[current_ic][4]);
    Serial.print(", 0x");
    serial_print_hex(tx_cfg[current_ic][5]);
    Serial.print(", Calculated PEC: 0x");
    cfg_pec = pec15_calc(6,&tx_cfg[current_ic][0]);
    serial_print_hex((uint8_t)(cfg_pec>>8));
    Serial.print(", 0x");
    serial_print_hex((uint8_t)(cfg_pec));
    Serial.println();
  }
  Serial.println();
}

void print_rxconfig()
```

```
{
  Serial.println("Received Configuration ");
  for (int current_ic=0; current_ic<TOTAL_IC; current_ic++)
  {
    Serial.print(" IC ");
    Serial.print(current_ic+1,DEC);
    Serial.print(": 0x");
    serial_print_hex(rx_cfg[current_ic][0]);
    Serial.print(", 0x");
    serial_print_hex(rx_cfg[current_ic][1]);
    Serial.print(", 0x");
    serial_print_hex(rx_cfg[current_ic][2]);
    Serial.print(", 0x");
    serial_print_hex(rx_cfg[current_ic][3]);
    Serial.print(", 0x");
    serial_print_hex(rx_cfg[current_ic][4]);
    Serial.print(", 0x");
    serial_print_hex(rx_cfg[current_ic][5]);
    Serial.print(", Received PEC: 0x");
    serial_print_hex(rx_cfg[current_ic][6]);
    Serial.print(", 0x");
    serial_print_hex(rx_cfg[current_ic][7]);
    Serial.println();
}
  Serial.println();
}

void serial_print_hex(uint8_t data)
{
  if (data< 16)
  {
    Serial.print("0");
    Serial.print((byte)data,HEX);
  }
  else
    Serial.print((byte)data,HEX);
}
```

Funcione.h esclavos

```
#include <Arduino.h>

#include <stdint.h>

#include <SoftwareSerial.h>

#include "LT_SPI.h"

#include "LTC68041.h"

//funcion para obtener la celda de mayor voltaje

void mayor(uint16_t cell_codes[1][12],int cellmax[2]){

  for (int i=0;i<12;i++){
     if(int(cell_codes[1][i])>cellmax[2]){
       cellmax[2]=int(cell_codes[1][i]);
       cellmax[1]=i;

     }

  }


}
//funcion para obtener la celda de menor voltaje
void menor(uint16_t cell_codes[1][12],int cellmin[2]){

  for (int i=0;i<12;i++){
     if(int(cell_codes[1][i])<cellmin[2]){
       cellmin[2]=int(cell_codes[1][i]);
       cellmin[1]=i;

     }
  }
}
//funcion para obtener las celdas que deben estar balanceando
void balanceo(uint16_t cell_codes[1][12],uint16_t balancear,int diferencialimite, int
cellmin[2], float CELL_BALANCE_THRESHOLD_V)
{
int diferencia=0;
   for (int i=0;i<12;i++){
     if(cell_codes[1][i]>CELL_BALANCE_THRESHOLD_V*10000){
   diferencia=cell_codes[1][i]-cellmin[2];
       if(diferencia>diferencialimite){
```

```
        balancear=(balancear & 0b0000000000000001)<<1;
      }else{

        balancear= balancear << 1;

      }


   }

}

}


void config_vlimites ( uint8_t tx_cfg[][6], uint16_t vuv,uint16_t vov){
//calcula el vuv y vov
/*
vuv value under voltage = limite minimo de voltaje
vov value over voltage = limite maximo de voltaje
*/
vuv=(vuv+1)*0.0016;
vov=vov*0.0016;
tx_cfg[1][0]=0b0000100;
tx_cfg[1][1]=vuv&(0b11111111);
tx_cfg[1][2]=(((vuv>>8) & (0b00001111))|((vov & 0b00001111<<4)));
tx_cfg[1][3]=((vov >> 4) & (0b11111111));
tx_cfg[1][5]=0b00000000;

}

void control_balanceo (uint8_t tx_cfg[][6],uint16_t balancear){

uint8_t cfg4 = balancear & 0b11111111;
uint8_t cfg5 = balancear >> 8;
 cfg5 = cfg5 & 0b00001111;

//rellenamos tx_cfg de forma que balanceen las celdas que se han calculado
tx_cfg[1][4] = (uint8_t)cfg4;
tx_cfg[1][5] = (uint8_t)cfg5;
/*con la ultima linea rellenamos nada mas los 4 primeros bits y dejamos los
ultimos
tal y como estaban*/


}
```

```
void temperaturas (float temps[35],float coefA,float coefB, float r25 ){
int analog[]={A0,A1,A2,A3,A4,A5};
int control1[]={17,16,15,14};
int control2[]={18,19,20,21};
//leemos los termistores no multiplexados
temps[0]=analogRead(analog[2])/1024 *3.3;
temps[1]=analogRead(analog[3])/1024 *3.3;
temps[2]=analogRead(analog[4])/1024 *3.3;
temps[3]=analogRead(analog[5])/1024 *3.3;
//leemos los termistores multiplexados del multiplexor 1
for (int i=4;i<20;i++){
  for(int j=0;j<2;j++){
    digitalWrite(control1[0],j) ;
      for (int k=0;k<2;k++){
        digitalWrite(control1[1],k);
          for(int l=0;l<2;l++){
            digitalWrite(control1[2],l);
              for(int m=0;m<2;m++){
                 digitalWrite(control1[3],m);

                 temps[i]=analogRead(analog[0])/1024 * 3.3;
              }
          }
      }
  }
}
//leemos los termistores multiplexados del mux 2
for (int i=20;i<36;i++){
  for(int j=0;j<2;j++){
    digitalWrite(control2[0],j) ;
      for (int k=0;k<2;k++){
        digitalWrite(control2[1],k);
          for(int l=0;l<2;l++){
            digitalWrite(control2[2],l);
              for(int m=0;m<2;m++){
                 digitalWrite(control2[3],m);

                 temps[i]=analogRead(analog[1])/1024 * 3.3;
              }
          }
      }
  }
}
//ahora transformamos el valor de voltaje en temperatura que podamos tratar
float Rt;
for(int i=0; i<36;i++){
  Rt=(temps[i]*7680)/(3.3-temps[i]);
```

```
  temps[i]=(coefB/(log(Rt/coefA)))-273.15;
}

}
```

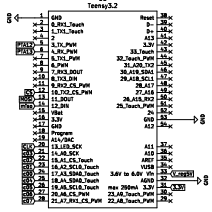
Script para el cálculo del ajuste de la curva OCV vs SOC

```
%calculo de la resistencia interna
ocvrint=4.1749;
vdescarga=4.129;
idescarga=0.8;
rint=(ocvrint-vdescarga)/idescarga


%convertimos la tabla en un array de datos
array=readtable('descarga800ma.csv');
array=removevars(array,7);
array=table2array(array);


%calculamos voltaje circuito abierto
ocv=array(:,2)+array(:,1)*rint;


%calculamos state of charge
ultvalor=length(ocv);
maxmah=array(ultvalor,5);
m=ones([ultvalor,1]);
maxmahv=m*maxmah;

soc=(maxmahv-array(:,5))/maxmah;

%graficamos los valores

plot(ocv,soc);
plot(soc,ocv);
%sacamos los valores
%% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( ocv, soc );

% Set up fittype and options.
ft = fittype( 'poly9' );

% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );

% Plot fit with data.
figure( 'Name', 'untitled fit 1' );
h = plot( fitresult, xData, yData );
legend( h, 'ocv vs. soc', 'untitled fit 1', 'Location', 'NorthEast', 'Interpreter', 'none' );
% Label axes
xlabel( 'ocv', 'Interpreter', 'none' );
ylabel( 'soc', 'Interpreter', 'none' );
```

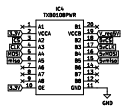
```
grid on
axis([2.5 4.2 0 1]);
```

jumpers de pruebas

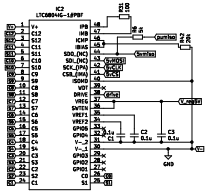
placa desarrollo



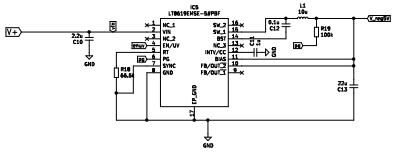
cambiador de niveles logicos



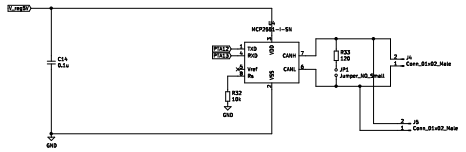
INTEGRADO BALANCEO CELDAS



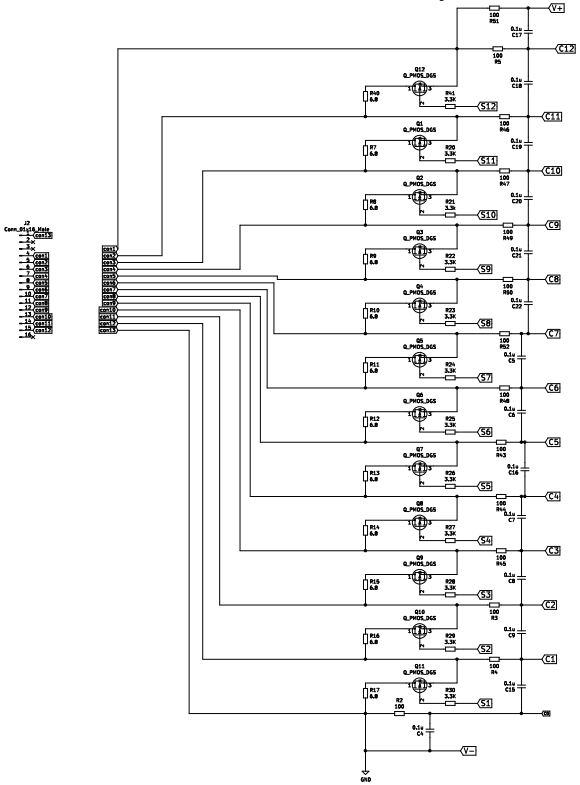
alimentaçao



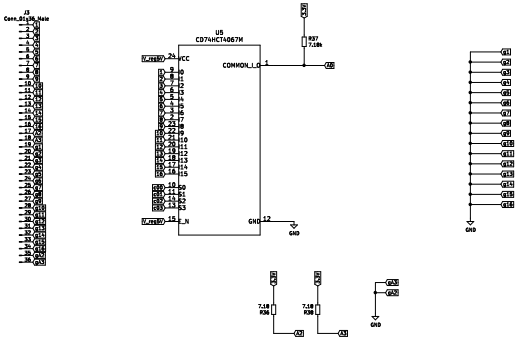
comunicacion can



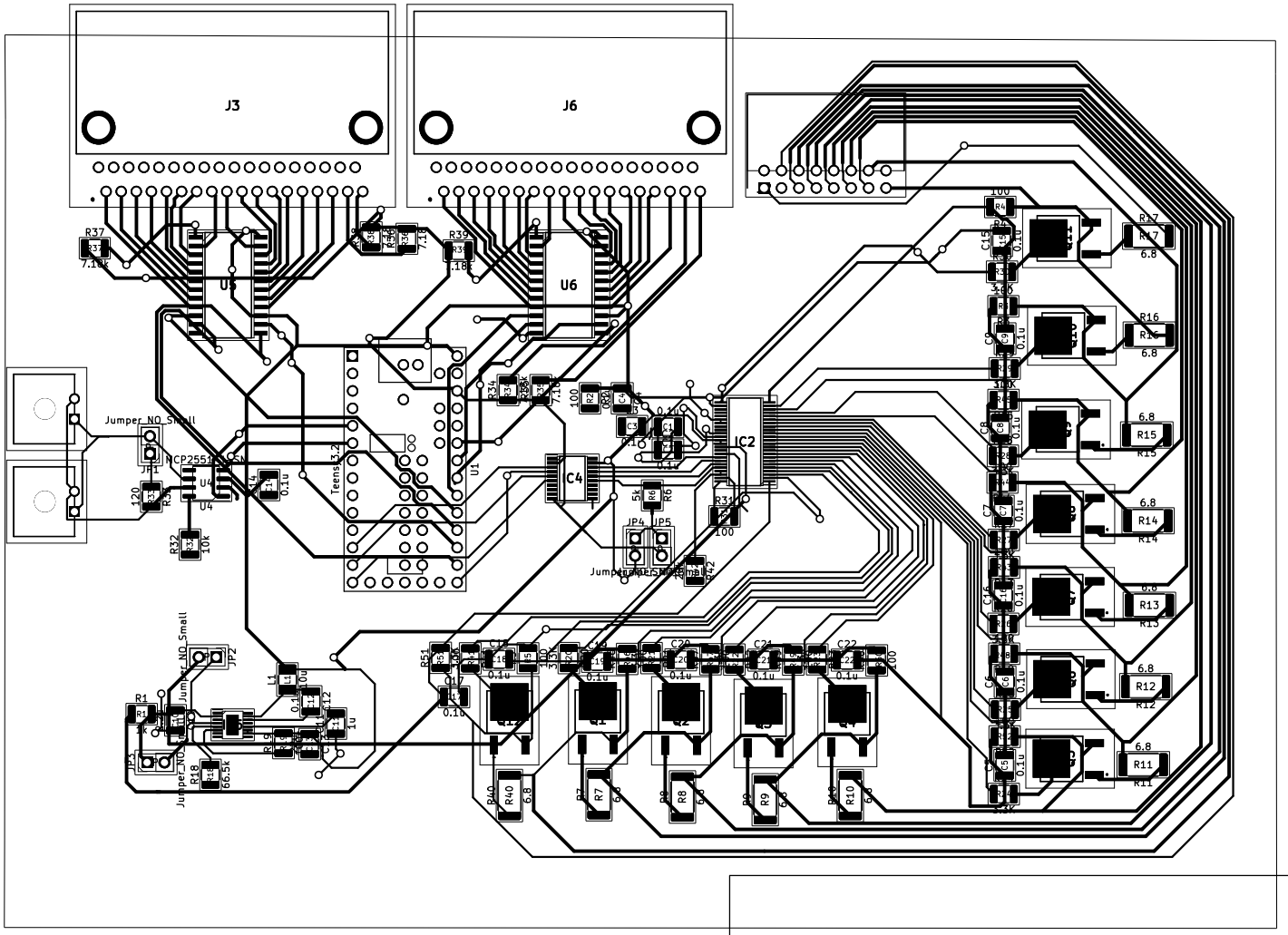
cell balancing



sistema multiplexado 36 termistores

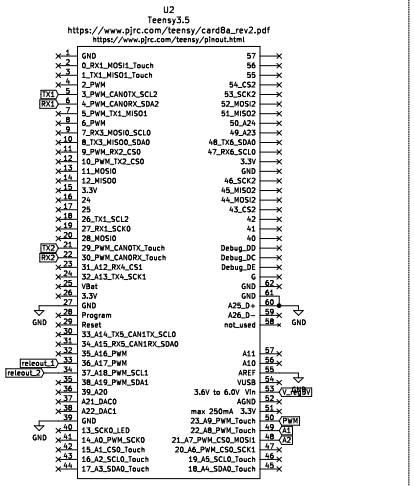


Sheet: /	
File: esclavos evaluacion.sch	
Title:	
Size: A1	Date:
KiCad E.D.A. kicad (5.1.9)-1	Rev:
	Id: 1/1

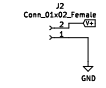


Sheet: File: esclavos evaluacion.kicad_pcb		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad (5.1.9)-1		Id: 1/1

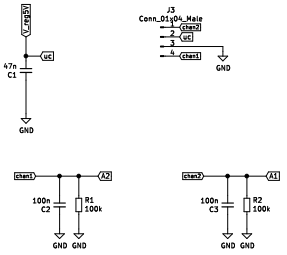
MICROCONTROLADOR CORTEX M4
micro maestro



ENTRADA 24

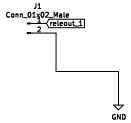


sensor de corriente

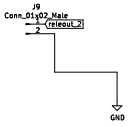


PWM ventiladores 1

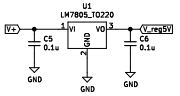
charge enable relay output



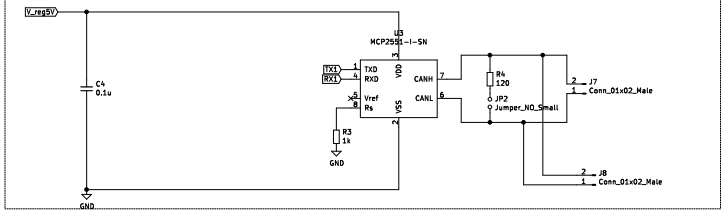
discharge enable relay output



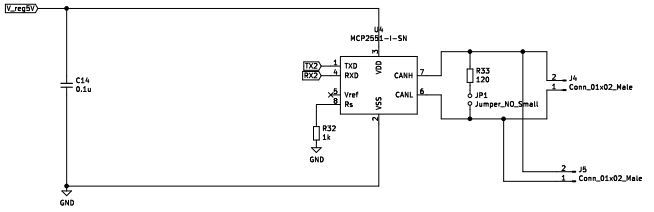
Alimentacion 5v



comunicacion can 1



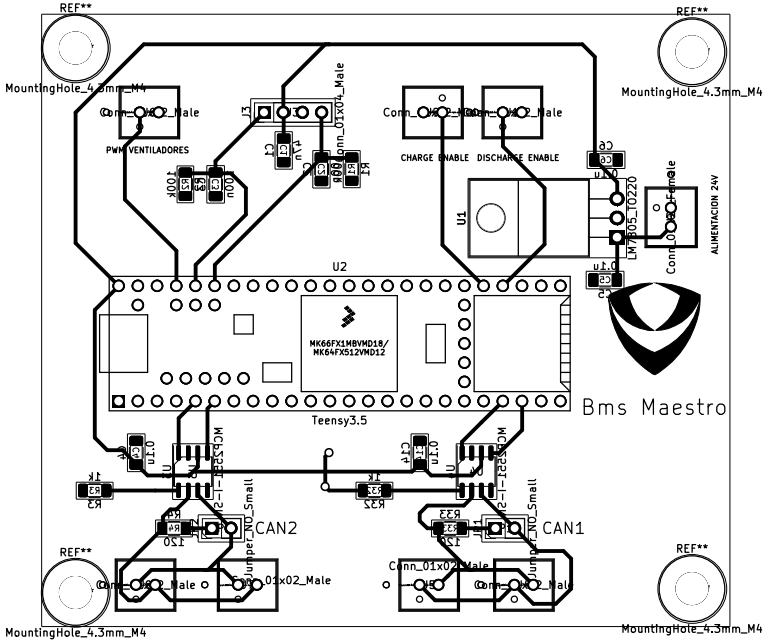
comunicacion can 2



Sheet: /
File: maestro.sch

Title:

Size: A2	Date:	Rev:
KiCad E.D.A. kicad (5.1.9)-1		Id: 1/1



Sheet:		
File: maestro.kicad_pcb		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad (5.1.9)-1		Id: 1/1