

Escuela de Ingeniería Industrial

TRABAJO FIN DE GRADO

Diseño, implementación y validación de un circuito electrónico para la medida del nivel de carga de baterías en robots móviles

Grado en Ingeniería en Electrónica Industrial y Automática

ALUMNO: Ángel Soto Boullosa

DIRECTORES: Juan José Rodríguez Andina

Universida₁deVigo

INDICE

MEMORIA	3
PLANOS	33
PLIEGO DE CONDICIONES	44
PRESUPUESTO	47
ANEXOS	51

Escuela de Ingeniería Industrial

TRABAJO FIN DE GRADO

*Diseño, implementación y validación de un circuito
electrónico para la medida del nivel de carga de baterías en
robots móviles*

Grado en Ingeniería en Electrónica Industrial y Automática

Documento

MEMORIA

Universidad de Vigo

CONTENIDO

Contenido	4
Figuras	5
Tablas	6
Memoria	7
1. Introducción	7
2. Objetivo y Alcance	8
3. Peticionario	9
4. Emplazamiento	9
5. Antecedentes	9
6. Diseño	10
1. Selección y adquisición de baterías	10
2. Selección del circuito integrado	13
3. Diseño eléctrico	15
4. Selección de componentes auxiliares	17
5. Diseño PCB	18
6. Fabricación PCB	19
7. Integración en el sistema	22
8. Programación del bq34z100	26
9. Validación del sistema	28
10. Conclusiones	31
11. Líneas Futuras	32
Bibliografía	32

FIGURAS

Figura 1.1 Diagrama general del <i>UviSpace</i> (1).....	7
Figura 1.2 Elementos del <i>UviSpace</i> (1)	7
Figura 1.3 <i>UviRobot</i> (1)	8
Figura 6.1 Batería.....	11
Figura 6.2 Curva de carga con 6.4 V	11
Figura 6.3 Curva de carga con 7 V	12
Figura 6.4 Curva de carga con 8.4 V	12
Figura 6.5 Diagrama de pines del <i>chip bq34z100</i> (3)	14
Figura 6.6 Diagrama de bloques del circuito	17
Figura 6.7 Verificación eléctrica y listado de conexiones	19
Figura 6.8 <i>CvPcb</i> – Asignación de huellas	20
Figura 6.9 <i>Pcbnew</i> – componentes y bordes de placa.	20
Figura 6.10 Pistas de cobre	21
Figura 6.11 Detalle del soldado del chip <i>bq34z100</i> al microscopio	22
Figura 6.12 Diagrama de bloques del sistema	23
Figura 6.13 Diseño 3D de la <i>PCB</i>	23
Figura 6.14 Primer prototipo de la <i>PCB</i> diseñada	24
Figura 6.15 Detalle interior del UGV	24
Figura 6.16 Detalle del soporte de la batería y cargador	25
Figura 7.1 Configuración de la memoria flash	26
Figura 7.2 Calibración del contador de culombios	27
Figura 7.3 Calibración de voltaje y temperatura.....	27
Figura 7.5 Calibración de offset y de corriente.....	28
Figura 8.1 Datos del experimento de descarga	29
Figura 8.2 Evolución del SOC en el experimento de descarga.....	30
Figura 8.3 Datos del experimento de carga	30
Figura 8.4 Evolución del SOC en el experimento de carga.....	31

TABLAS

Tabla 5.1 Especificaciones del <i>Pirate 4WD Mobile Platform</i> (2)	9
Tabla 5.2 Especificaciones del controlador	9
Tabla 6.1 Especificaciones de la <i>batería</i>	10
Tabla 6.2 Caracterización del consumo de corriente del UGV	11
Tabla 6.3 Especificaciones del cargador.....	12
Tabla 6.4 Funcionalidad de los pines del <i>bq34z100</i> (3).....	15

I. MEMORIA

1. Introducción

Este TFG forma parte del proyecto *UviSpace*. *UviSpace* es un proyecto tanto de *hardware* como de *software* de código abierto, desarrollado en el Departamento de Tecnología Electrónica de la Universidad de Vigo. Este proyecto consiste en el control autónomo de varios vehículos terrestres no tripulados o UGVs (del inglés, *Unmanned Ground Vehicles*) dentro de un espacio delimitado, mediante un sistema inteligente que consta de tres elementos principales, representados en las figuras 1.1 y 1.2: varios nodos de localización, que procesan la información proveniente de varias cámaras situadas en el techo de la sala; un controlador central, basado en un procesador; y los propios vehículos controlados por un microcontrolador instalado en cada uno de ellos. En la figura 1.3 se puede ver un detalle de uno de los vehículos.

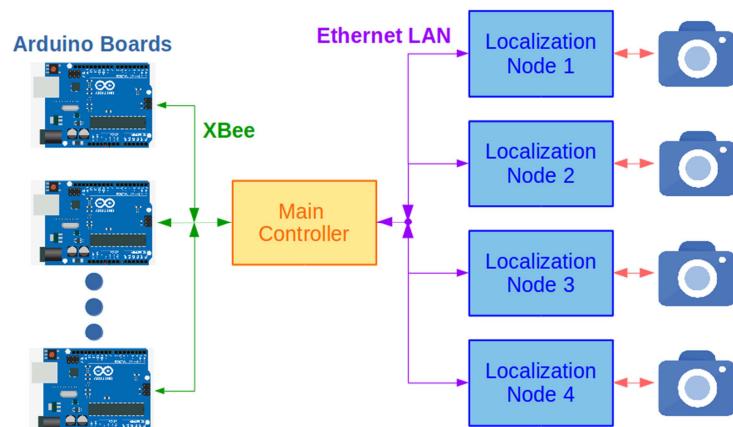


Figura 1.1 Diagrama general del *UviSpace* (1)

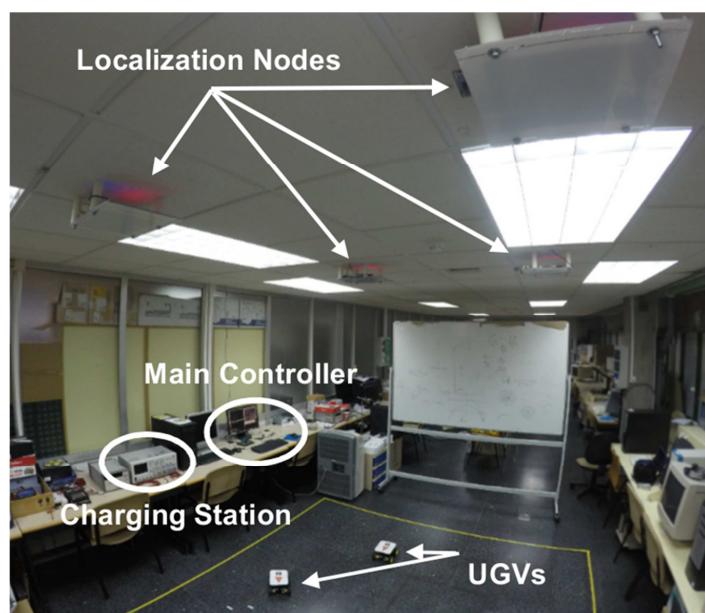


Figura 1.2 Elementos del *UviSpace* (1)

La alimentación del vehículo se puede realizar de 2 formas: la primera, mediante una fuente de alimentación cableada al vehículo; y la segunda, mediante un sistema de baterías. Respecto a la segunda alternativa, se han barajado varias opciones. Como primera opción, se ha valorado el uso de pilas alcalinas convencionales o recargables. Esta solución presenta una desventaja principal, que es su baja capacidad, lo que implica poca autonomía. Así pues, el uso de pilas alcalinas conlleva un mayor gasto económico, además del impacto ecológico que ello supone. Como segunda opción, se ha valorado el uso de baterías. Por tanto, se ha estudiado la utilización de un sistema de baterías con alta capacidad, volumen reducido y al que se le pueda acoplar un sistema de monitorización del nivel de carga.

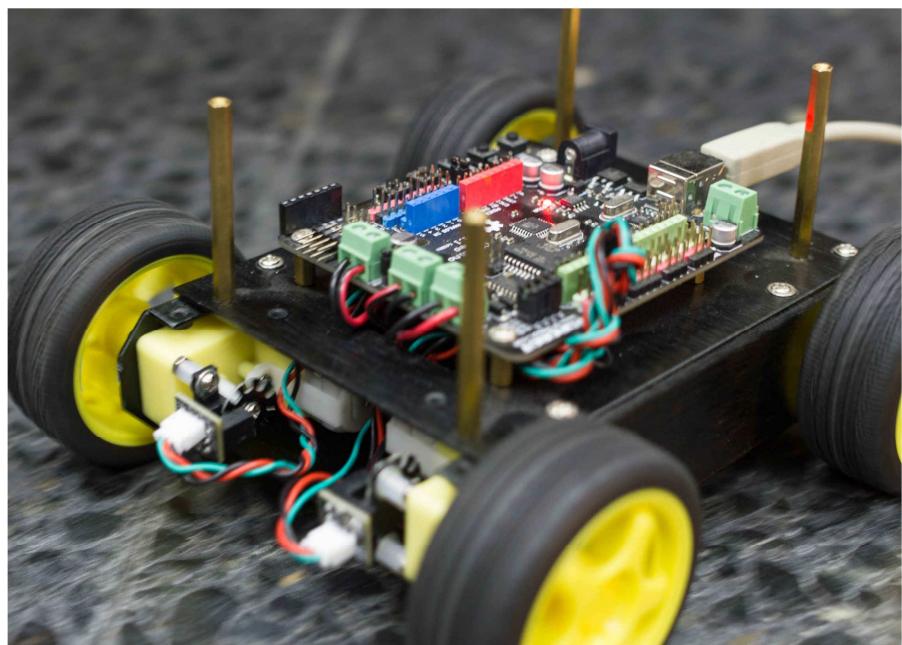


Figura 1.3 *UviRobot (1)*

2. Objetivo y Alcance

El presente TFG se ha dividido en varios objetivos intermedios. Estos son:

- ❖ Selección de las baterías.
- ❖ Selección del circuito integrado de monitorización.
- ❖ Selección de los componentes y diseño del circuito eléctrico.
- ❖ Diseño de una PCB.
- ❖ Montaje y validación de la PCB.
- ❖ Integración de la PCB en un vehículo del *UviSpace*.
- ❖ Validación del sistema.

El objetivo de este TFG es dotar a los UGVs de mayor autonomía y tener un mayor control del sistema. Al instalar la batería en los vehículos, éstos no tienen que estar conectados a la fuente de alimentación por medio de cables, que le impiden realizar los movimientos libremente. Por otra parte, debido al uso del circuito electrónico implementado en la PCB objeto

de este TFG, se obtiene un mayor control del sistema, ya que es posible monitorizar el estado de carga en todo momento.

3. Peticionario

Este proyecto es el Trabajo de Fin de Grado de la titulación de Graduado en Ingeniería Industrial en Electrónica Industrial y Automática. Su realización es intrínseca para la obtención del título universitario. Así pues, como peticionario figura la Escuela de Ingeniería Industrial de la Universidad de Vigo.

4. Emplazamiento

La realización de este proyecto se ha llevado a cabo en las instalaciones (laboratorio de microelectrónica y laboratorio de simulación) del Departamento de Tecnología Electrónica de la Sede Campus de la Escuela de Ingeniería Industrial sita en Rúa Maxwell, s/n, Campus Universitario Lagoas-Marcosende 36310 Vigo, Pontevedra.

5. Antecedentes

Previa a la realización de este TFG, en el Departamento de Tecnología Electrónica se disponía del siguiente material:

- ❖ Robot:
Pirate 4WD Mobile Platform de la marca *DFRobot*
Alimentación de los motores: Entre 3 y 12 V CC

Alimentación	Velocidad (sin carga)	Corriente (sin carga)	Corriente (carga máxima)	Par motor
3V	100 rpm	60 mA	260 mA	1.2 Kg x cm
6V	200 rpm	71 mA	470 mA	1.92 Kg x cm

Tabla 5.1 Especificaciones del *Pirate 4WD Mobile Platform* (2)

- ❖ Controlador:
Romeo All in One Controller (Arduino compatible Atmega 328)

Alimentación CC	USB o tensión entre 5 y 12 V
Salida CC	5 o 3.3 V
Microcontrolador	Atmega 328
Bootloader	Arduino Uno
Canales Analógicos E/S	8 canales de 10 bits
Comunicación	3 canales I2C, <i>Bluetooth</i> y puerto serie

Tabla 5.2 Especificaciones del controlador

❖ Alimentación:

Se estaban utilizando pilas alcalinas con una tensión igual a 7.5V (5x1.5V) o bien la fuente de alimentación cableada a los UGVs.

6. Diseño

1. Selección y adquisición de baterías

Criterios

Las características eléctricas de las baterías deben ser las adecuadas para la aplicación. Puesto que el conjunto del robot y controlador tiene que ser alimentado con una tensión de entre 5 y 12V y el consumo de los motores para una tensión de 6V es de 470mA según la tabla 5.1, se elige una batería con una tensión aproximada de 7.5 V y que proporcione una corriente suficiente.

Justificación

Se han elegido baterías de polímero de Litio (también denominadas *LiPo* o *LiPoly*) de 7.4 V CC de la marca *Turnigy* de la serie *Turnigy Blue* (figura 6.1)

Tecnología	Polímero de Litio
Tensión nominal	7.4 V CC
Rango de tensiones	6.4 a 8.4 V CC
Nº de celdas	2
Tensión de almacenamiento	Entre 3.7 y 3.8 V CC por celda
Capacidad	2200 mAh
Corriente de descarga continua máxima	20 C (equivale a 44 A)
Corriente de descarga máxima	30 C (equivale a 66 A)
Ciclos de carga-descarga	1000
Conecotor	XT60 hembra

Tabla 6.1 Especificaciones de la batería

Se ha optado por esta solución por 2 motivos principales. El primero, el tamaño. Dado el tamaño de los UGVs, estas baterías son ideales ya que no sobresalen de la planta de los vehículos. El segundo, sus características eléctricas.



Figura 6.1 Batería

Parametrización de la carga

Se han realizado 3 pruebas a diferentes tensiones de alimentación, utilizando la fuente de alimentación, simulando las condiciones de batería casi descargada (6.4 V), batería totalmente cargada (8.4 V) y batería con un valor intermedio de 7 V para obtener una caracterización de la carga (motores y controlador).

El experimento consiste en tener un UGV alimentado y parado, para luego aplicar la máxima velocidad para finalmente dejarlo en reposo de nuevo. A continuación se incluyen los datos obtenidos así como las formas de onda obtenidas con el osciloscopio para las diferentes tensiones de alimentación (figuras 6.2, 6.3 y 6.4).

Alimentación	Corriente (sin carga)	Corriente (de pico)	Corriente (carga máxima)
6.4 V	117 mA	1583 mA	700 mA
7 V	120 mA	1772 mA	810 mA
8.4 V	124 mA	2005 mA	930 mA

Tabla 6.2 Caracterización del consumo de corriente del UGV

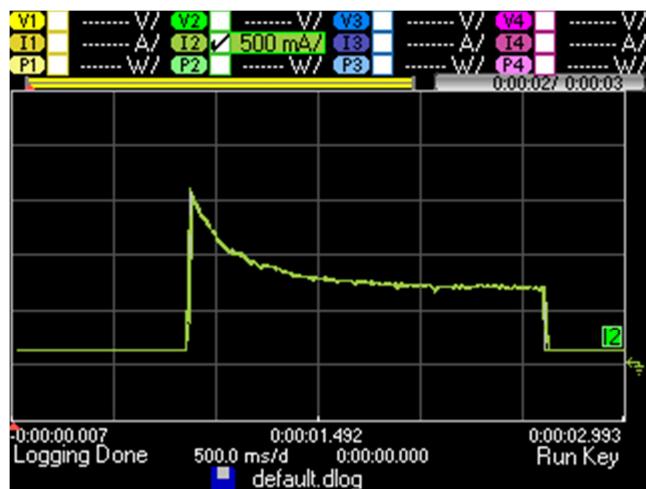


Figura 6.2 Curva de carga con 6.4 V

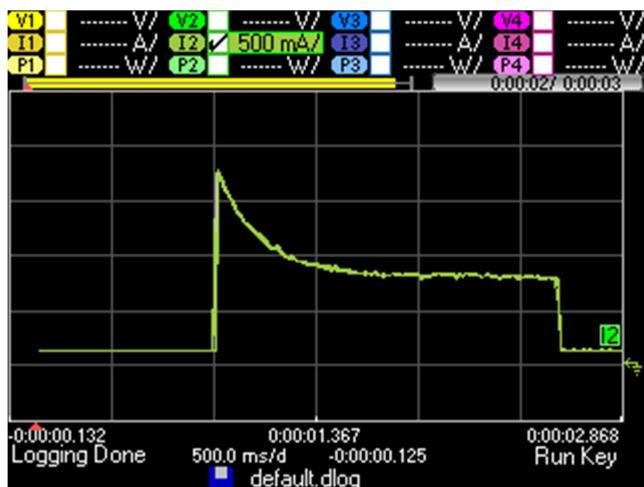


Figura 6.3 Curva de carga con 7 V

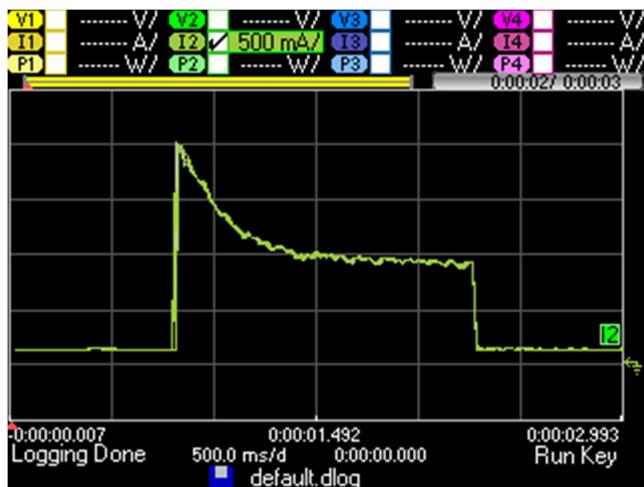


Figura 6.4 Curva de carga con 8.4 V

Cargador

Se ha adquirido un cargador balanceador de la marca *Imax* modelo B6AC. El motivo de su adquisición radica en la tecnología de las baterías. Las baterías de polímero de litio requieren de un proceso de carga controlado. Las baterías adquiridas para este proyecto tienen 2 celdas. Ambas celdas deben de tener un nivel de tensión similar en todo momento para que no se dañen. Cualquier mal uso de estas baterías puede conducir a su ignición espontánea, de ahí el uso de un cargador específico para esta tecnología.

Alimentación	11 a 18 V CC o 125 V CA
Corriente de carga	0.1 a 5 A
Corriente de descarga	0.1 a 1 A
Número de celdas (LiPo)	1 a 6

Tabla 6.3 Especificaciones del cargador

Al ser un cargador-descargador balanceador permite tanto la carga de las baterías como la descarga, simulando un consumo de corriente. El término balanceador quiere decir que en todos los procesos, el dispositivo, vigila la tensión de cada celda independientemente para asegurarse de que se encuentran con niveles de tensión similares. Este control de celdas lo realiza a través de

los terminales correspondientes a cada una de ellas. En la figura 6.1 se corresponden a los tres cables de menos diámetro agrupados en el conector blanco.

Este cargador ha sido utilizado para despertar las baterías, esto es, llevarlas desde el estado del que salen de fábrica al estado en el que se pueden utilizar normalmente. Este proceso consiste en realizar varios ciclos de carga y descarga controlados. De igual manera, el cargador sirve para llevar a las baterías desde el estado de operación al estado de almacenamiento si las mismas no van a ser utilizadas durante un tiempo.

2. Selección del circuito integrado

Criterios

El circuito integrado de monitorización del estado de baterías ha de proporcionar la información suficiente para tener controlada la batería en todo momento. La información básica que se exige del chip es obtener el estado de carga de la batería, la tensión entre sus bornes, la capacidad restante, la corriente que consume la carga así como también la temperatura de la batería. Además, es deseable que pueda programarse estados de alerta para un mejor control de los parámetros de la batería. Por otra parte, el tipo de encapsulado deberá ser el adecuado para poder integrarse en una PCB que sea posible fabricar en el departamento, es decir, que sea soldable con la tecnología de la que se dispone. Para ello ha de ser un circuito integrado de montaje superficial (SMD) o de tecnología de agujeros pasantes (*through-hole technology* o THT).

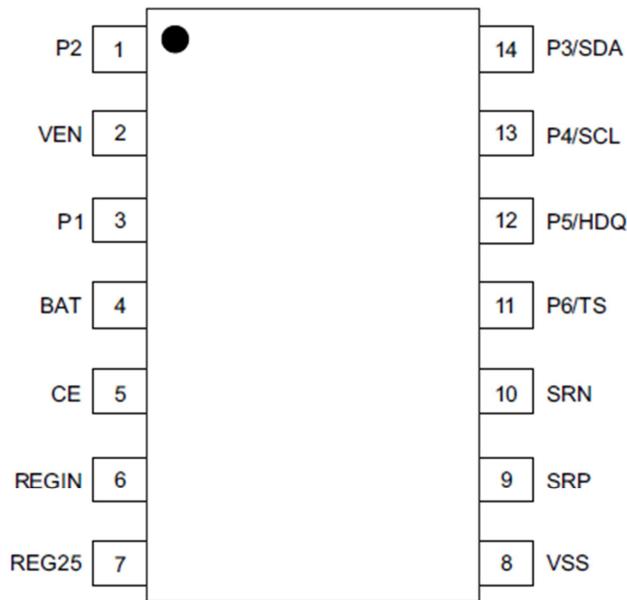
Justificación

Se ha elegido el circuito integrado *bq34z100* de *Texas Instruments*. Este circuito integrado, con su correspondiente circuito externo, proporciona la información necesaria relativa al estado de la batería como para tenerla totalmente controlada, tanto por el problema de descarga por debajo del valor de voltaje mínimo admisible como para tener un control autónomo, bajo petición de un maestro (controlador), del nivel de carga de batería y determinar así si es necesario mandar al robot a la estación de carga.

Esto es, el chip proporciona, mediante una conexión I2C con un maestro información que almacena en registros internos de memoria. Esta información es el estado de carga de la batería, el voltaje, la capacidad restante, la corriente que consume la carga, la temperatura tanto del chip como de otro punto que se desee (mediante termistor externo), entre otras.

Características

El chip *bq34z100* (figura 6.6) predice tanto la capacidad de la batería como otras características de baterías de una o varias celdas de tecnologías tanto de iones de litio como otro tipo de baterías. Para la aplicación del *UviSpace* se utiliza con una batería cuya tecnología es polímero de litio. Esta información puede ser solicitada por un procesador.

Figura 6.5 Diagrama de pines del chip *bq34z100* (3)

Nombre del pin	Número de pin	Tipo	Descripción
P2	1	Salida	Pin de ALERTA. No utilizado. Conectado a VSS.
VEN	2	Salida	Habilita la reducción de consumo en el divisor de tensión de entrada. Activo a nivel alto.
P1	3	Salida	No utilizado. Conectado a VSS.
BAT	4	Entrada	Voltaje de <i>batería</i> con el nivel adaptado después del divisor de tensión.
CE	5	Entrada	Chip Enable. El regulador CC-CC interno será desconectado de REGIN cuando CE esté a nivel bajo.
REGIN	6	Alimentación	Entrada del regulador CC-CC interno.
REG25	7	Alimentación	Salida de 2,5V del regulador CC-CC interno.
VSS	8	Alimentación	Tierra del dispositivo.
SRP	9	Entrada	Pin de entrada analógico conectado a la resistencia de sensado de corriente interna del <i>chip</i> . Terminal positivo de la tensión que se genera en la resistencia de sensado.
SRN	10	Entrada	Pin de entrada analógico conectado a la resistencia de sensado de corriente interna del <i>chip</i> . Terminal negativo de la tensión que se genera en la resistencia de sensado.
P6/TS	11	Entrada	Sensor de temperatura de la <i>batería</i> .
P5/HDQ	12	Entrada/ Salida digital	Pin de comunicación serie <i>HDQ</i> . No utilizado.
P4/SCL	13	Entrada	Pin de entrada de la señal de reloj de la comunicación serie <i>I2C</i> .
P3/SDA	14	Entrada/ Salida	Pin de datos de la comunicación serie <i>I2C</i> . Terminal

	digital	de colector abierto.
--	---------	----------------------

Tabla 6.4 Funcionalidad de los pines del *bq34z100* (3)

La información es accesible a través de una serie de comandos estándar (3). También existe la posibilidad de obtener funcionalidades extra por medio de un juego de comandos extendidos. Estos comandos sirven tanto para leer como para escribir en las posiciones correspondientes de la memoria *flash* y se envían a través de uno de los 2 canales posibles de comunicación serie de los que dispone el circuito integrado (HDQ o I2C). En la aplicación del *UviSpace* sólo se utiliza la comunicación I2C ya que también está implementada en los controladores de los UGVs.

La clave de la predicción tan precisa del nivel de carga reside en el algoritmo de seguimiento de la impedancia. Este algoritmo es propiedad de *Texas Instruments*. Tiene en cuenta medidas de voltaje y medidas instantáneas de corriente así como propiedades de la batería para crear predicciones sobre el estado de carga que pueden llegar a menos del 1% de error en una amplia variedad de condiciones de operación.

Por otra parte, la temperatura en el entorno de la batería se monitoriza mediante un termistor NTC. Esta información es utilizada por el algoritmo así como también para establecer una protección ante una sobre temperatura.

3. Diseño eléctrico

Criterios

Una vez que se ha decidido la utilización del circuito integrado de *Texas Instruments* se adquiere una placa de evaluación o EVM (4). Ésta es una solución ofrecida por *Texas Instruments* en la que se implementa un circuito que contiene el circuito integrado *bq34z100* así como el resto de circuitos auxiliares y dispositivos configurables, tales como *jumpers*, que permiten un uso genérico del chip válido para diferentes aplicaciones. Se puede utilizar con un amplio rango de tensiones de entrada e incorpora 2 circuitos integrados que permiten el uso de diodos LED para mostrar el estado de carga.

Esta placa se utiliza como placa de desarrollo para la aplicación del *UviSpace* permitiendo variar las configuraciones del circuito integrado y realizar el proceso de calibración para obtener la configuración final que se implementa en la PCB final.

Se dará información más completa en la sección 6.7 del presente documento.

Justificación

Una vez realizado un estudio de las diferentes configuraciones posibles del circuito integrado *bq34z100* de *Texas Instruments*, se ha diseñado un esquema eléctrico partiendo de los circuitos propuestos por el fabricante y adaptándolos a la aplicación concreta. Para ello se eligen las funcionalidades que se desean implementar y se sustituyen los elementos que permiten variaciones rápidas del circuito tales como los *jumpers* que propone el fabricante, por elementos definitivos.

El circuito impreso que se ha diseñado contiene el circuito integrado y los circuitos auxiliares que le dotan de las funcionalidades necesarias para su correcto funcionamiento.

Esquemático

Teniendo en cuenta todo lo señalado en puntos anteriores, se procedió al diseño del esquemático del circuito. Para ello se utiliza el software *KiCad* de libre distribución.

Se puede dividir el esquema eléctrico en los siguientes bloques (figura 6.7):

- Divisor de tensión. Consiste en un conjunto de transistores y resistencias cuyo objetivo es obtener un valor de tensión, en función del valor de tensión de la batería, tal que en el pin 4 (BAT) del circuito integrado se tenga una tensión de entre 0.8 y 1V. Estas especificaciones son recomendaciones que proporciona *Texas Instruments* en la hoja de características del circuito integrado (3). Para ello, el elemento que se debe diseñar es la resistencia R2. Este cálculo se realiza utilizando la siguiente ecuación (3):

$$RSeries = \frac{16500\Omega \cdot (Vin \text{ max mV} - VBAT \text{ mV})}{VBAT \text{ mV}}$$

Siendo *Vin max* la tensión máxima de la batería expresada en milivoltios, *VBAT* la tensión que se le aplica al terminal 4 (BAT) del circuito integrado *bq34z100* expresada en milivoltios y *RSeries* el valor de resistencia R2 de escalado de tensión.

Dado que la tensión máxima de la batería es 8400 mV y que el valor de tensión admisible en el terminal 4 del chip está en el rango de 800 mV a 1000mV se calcula el valor de *RSeries* para los extremos:

$$RSeries \text{ max} = \frac{16500 \Omega \cdot (8400 \text{ mV} - 800 \text{ mV})}{800 \text{ mV}} = 156750 \Omega$$

$$RSeries \text{ min} = \frac{16500 \Omega \cdot (8400 \text{ mV} - 1000 \text{ mV})}{1000 \text{ mV}} = 122100 \Omega$$

De esta forma se obtiene un rango de valores para la resistencia R2. Por facilidad a la hora de adquirir el componente, se decide que el valor de la R2 sea de 150 k Ω . Por lo tanto el valor de la tensión en el terminal 4 del chip será de 832.43 mV.

- Comunicación I2C. Este bloque es el encargado de permitir la comunicación del circuito integrado con el maestro. Se realiza mediante una implementación del protocolo de comunicaciones serie I2C. Este canal se utiliza tanto para programar los registros internos del circuito integrado como para leer los parámetros de la batería. Se ha decidido colocar unos interruptores en las líneas de *pull-up* del canal serie para poder disponer de 2 funcionalidades distintas seleccionables de manera cómoda por el usuario. La correcta posición de estos selectores permite, o bien, la comunicación (a través del adaptador correspondiente, que es adquirido en el mismo paquete que la EVM) con el ordenador para operaciones de configuración, calibrado y lectura de valores utilizando la interfaz gráfica propia del *software* proporcionado por *Texas Instruments* (3), o bien con el controlador del UGV a través de su canal de comunicaciones serie I2C. En el primer caso, los interruptores deben de estar en estado de cerrado para permitir que el chip determine el nivel eléctrico alto del canal.

En el segundo caso, los interruptores deben de estar en estado de abierto para permitir que sea el controlador de los UGVs quienes determinen el nivel eléctrico alto del canal I2C que en este caso es de 5V.

- Sensado de temperatura. Consiste en un termistor conectado a un pin de entrada del circuito integrado que leerá el valor de resistencia del componente. Éste variará en función de las variaciones de temperatura que existan en el entorno del termistor. Éste está alimentado directamente desde el chip con una tensión de referencia de 2,5V proveniente del regulador CC-CC interno. Para ser utilizado como una medida de la temperatura del entorno de la batería, el termistor ha de colocarse lo más cerca posible de ésta.
- Sensado de corriente. Consiste en una resistencia de un valor muy pequeño que se coloca en serie con la carga y este conjunto carga-resistencia de sensado está en paralelo con la batería. De esta manera toda la corriente consumida por la carga circula por la resistencia de sensado. Debido al bajo valor de este componente, se genera una tensión de valor muy bajo que es recogida por el circuito integrado entre sus pines 9 y 10 a los cuales está conectado internamente a un contador de culombios. Éste integra la corriente en el tiempo para predecir la capacidad restante.

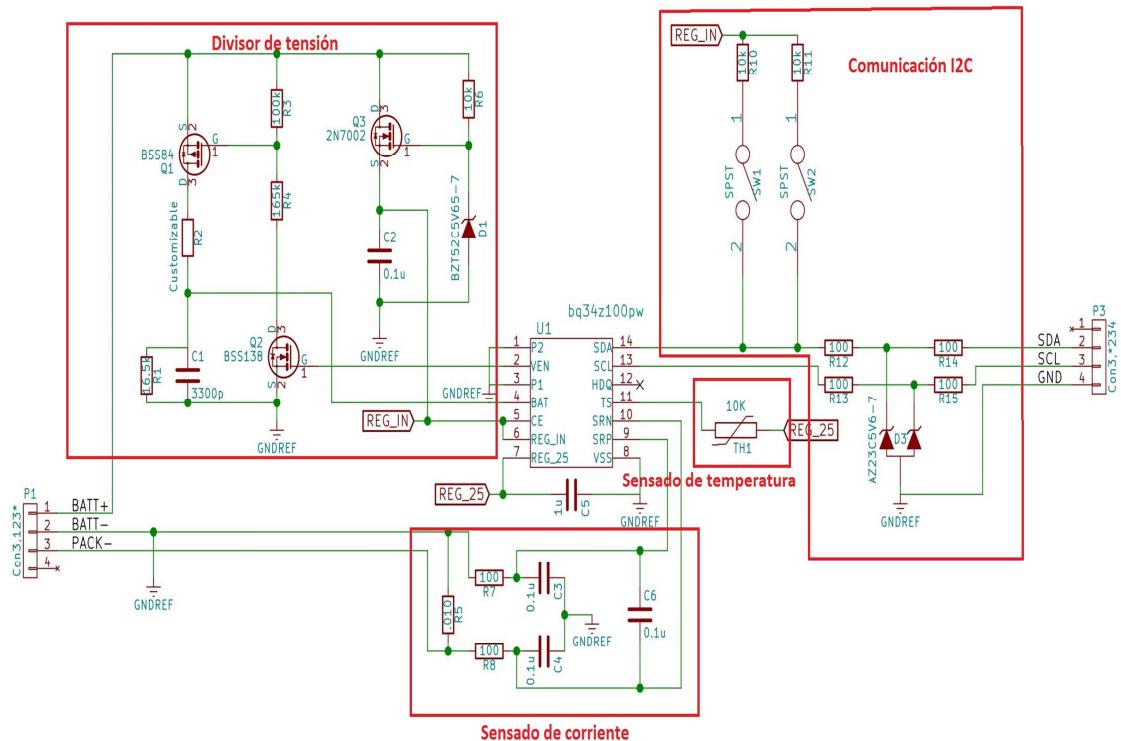


Figura 6.6 Diagrama de bloques del circuito

4. Selección de componentes auxiliares

Criterios

Tal y como sucede con el circuito integrado de *Texas Instruments*, el resto de componentes deben tener características físicas que posibiliten la fabricación de la PCB utilizando el material y herramientas disponibles en el departamento. Así pues, se decide utilizar componentes de

montaje superficial (SMD) y otros componentes de tecnología de agujeros pasantes (*through-hole technology* o THT).

Los componentes SMD se eligen del mayor tamaño posible para facilitar su soldadura manual.

Los componentes THT son, básicamente, los conectores de la PCB. Esta tecnología ya implica por sí misma mayor facilidad de instalación y soldado en la PCB.

Justificación

Los componentes elegidos son los que utiliza *Texas Instruments* para realizar su EVM (4). De este modo, se intenta reproducir lo más fielmente posible la placa de evaluación, con las modificaciones de diseño adoptadas y siguiendo el criterio de fácil fabricación, aunque siempre manteniendo los valores de referencia, para obtener un comportamiento lo más similar posible a la misma.

5. Diseño PCB

Criterios

Para que la fabricación de la PCB en el departamento sea factible se deben de tener en cuenta una serie de consideraciones:

- Ancho de pistas. Debido a que la fabricación de la PCB se hace de manera manual, para permitir que las pistas sean fáciles de fabricar, no se utilizarán anchos de pista menores de 0.5 mm.
- Capas de cobre. Si bien una PCB puede tener hasta 32 capas de cobre, esto solo es posible con métodos de fabricación sofisticados. Así pues, sólo es posible la fabricación de placas de circuito impreso de dos capas.
- Vías. Para conectar pistas de dos capas distintas es necesaria la realización de vías. El tamaño mínimo de la vía debe de ser de 2 mm de diámetro para poder luego taladrar y colocar el hilo conductor correspondiente.

Justificación

El ancho de pistas se ha respetado en todo momento a excepción de en 2 ocasiones. Una de ellas es en el entorno del circuito integrado. Debido al reducido tamaño de sus pines, se ha utilizado un ancho de pista de 0.32mm. Este ancho de pista se utiliza para realizar una separación en forma de araña entre sus pines que permite variar el ancho a su dimensión normal de 0.5mm para ser conectado al resto del circuito. La otra ocasión donde se varía el ancho de pista es en la conexión de la resistencia de sensado con el terminal de la PCB. Se ha optado por un ancho de 2mm que ofrece menos resistencia al paso de corriente lo que permite disminuir la caída de tensión en la pista antes de la resistencia. Si esta caída de tensión fuese relativamente grande, el sensado de corriente no se realiza de manera correcta. Por este motivo, además, se ha decidido colocar dicha resistencia lo más cerca posible del conector de la PCB así como también que las 2 pistas que unen la resistencia con el conector sean idénticas en cuanto a tamaño y forma. De esta manera se pretende que ambas pistas tengan el mismo valor de resistencia.

Para el tamaño de la placa se han tomado como referencia las dimensiones internas de los vehículos. Tras realizar un estudio de las posibles ubicaciones finales de la PCB en los vehículos, se determina que la mejor posición era dentro de la caja del chasis, en la parte delantera, en el

sentido de la marcha, entre los motores y la base de los vehículos. En cuanto al ancho, para favorecer el encaje de la PCB en el hueco, se decide que la PCB ocupe todo el ancho disponible en el interior de los vehículos para ser, de esa manera y a través de tornillos, fijada al chasis en sus laterales quedando así una PCB de 100mm de ancho. En lo relativo al largo, si bien se tiene un valor de referencia para que la placa no interfiera mecánicamente con el resto de elementos internos de los UGVs, después del diseño del circuito impreso, se intenta minimizar esta dimensión para compactar el diseño quedando finalmente una PCB de 70mm de largo.

6. Fabricación PCB

Fotolitos

Una vez que se tiene el diseño final del esquemático del circuito en el *KiCad* han de seguirse unos pasos para poder fabricar la PCB (5).

Primeramente, el *software* de diseño ya integra una herramienta de verificación del circuito eléctrico que realiza un control de reglas eléctricas. Una vez solucionados los posibles errores si los hubiera, es necesario general un archivo de conexiones eléctricas o *netlist* (figura 6.8).

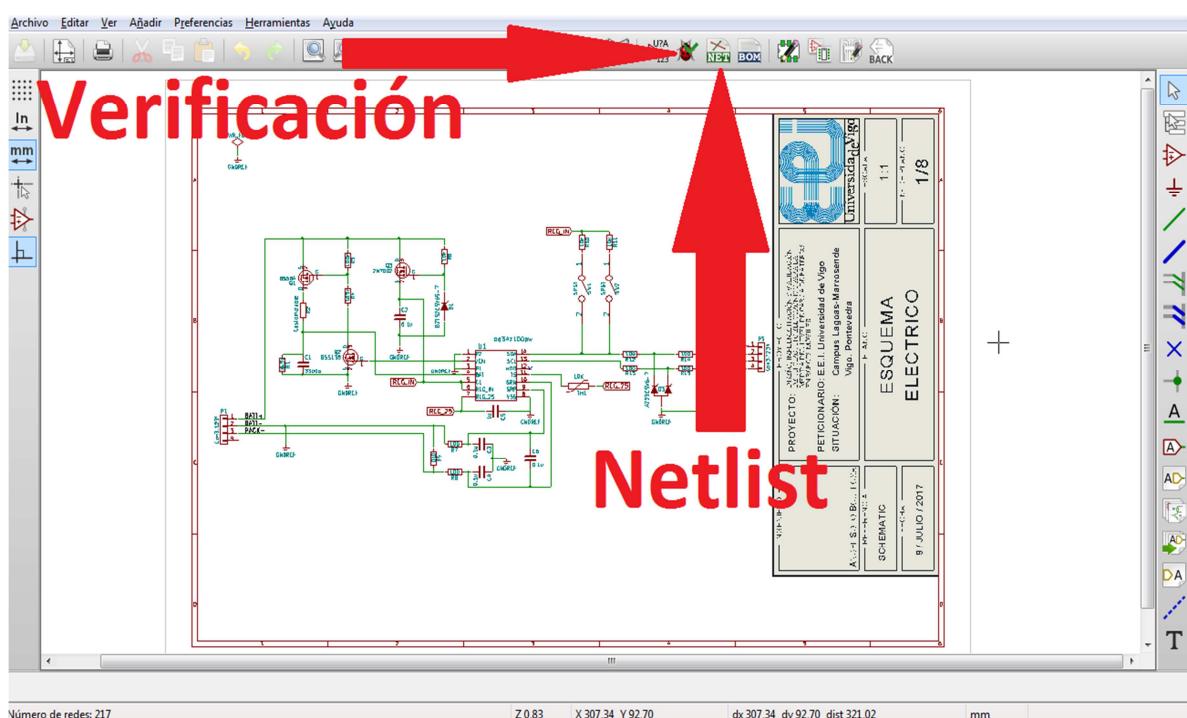


Figura 6.7 Verificación eléctrica y listado de conexiones

Para pasar definitivamente al diseño de la PCB con la herramienta *Pcbnew* que proporciona *KiCad* es necesario previamente asignar la huella de los componentes a cada uno de ellos. Estas huellas se asignan con la herramienta *CvPcb* (figura 6.9) ya sea desde librerías importadas o desde librerías propias (5).

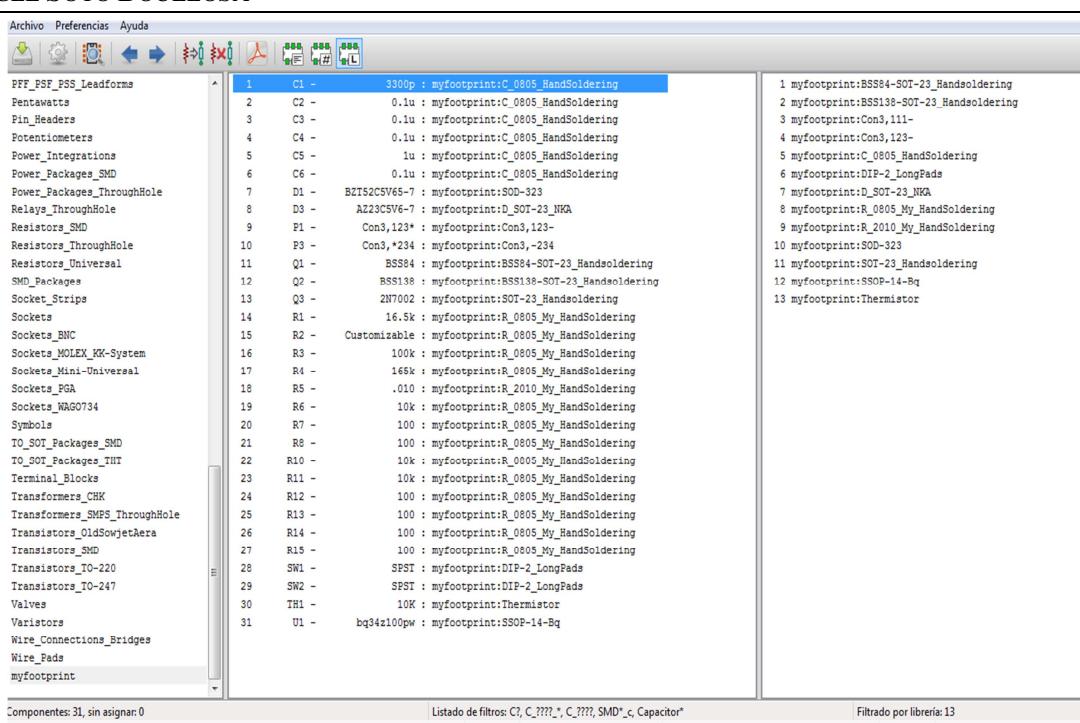


Figura 6.8 CvPcb – Asignación de huellas

Una vez que se dispone de toda esta información, se abre la herramienta *Pcbnew* (figura 6.10) y se importa el *netlist* o listado de redes. De este modo, en pantalla se verán todas las huellas de los componentes.



Figura 6.9 Pcbnew – componentes y bordes de placa.

El siguiente paso consiste en crear pistas de cobre utilizando ambas caras. En la figura 6.9 se pueden apreciar varios componentes cuya huella es de color verde. Este indica que esos componentes se han situado en la cara inferior. Habiendo componentes en ambas caras de la

PCB es necesaria la creación de vías. Una vía es un agujero en la PCB que permite conectar una pista de la cara superior de la PCB con una pista de la cara inferior. En la siguiente figura se puede ver la distribución de pistas final.

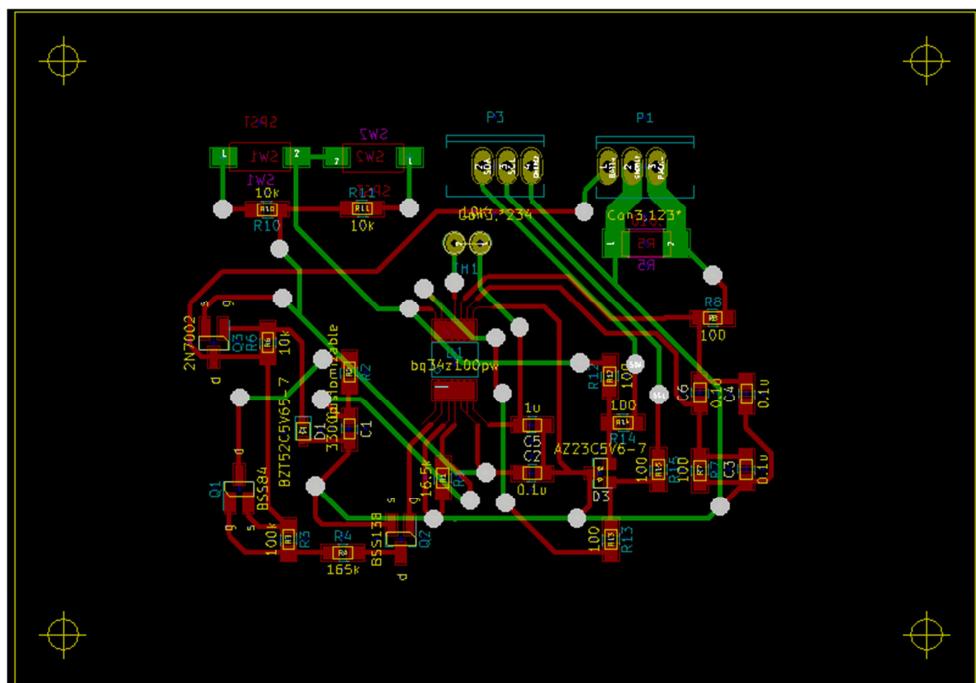


Figura 6.10 Pistas de cobre

Estas operaciones anteriores conducen a la impresión de los fotolitos. Un fotolito es una impresión de las capas de cobre sobre un soporte transparente que luego se utilizará en el proceso de insolación.

Insolación

El siguiente paso para la fabricación de la PCB consiste en introducir una placa de circuito impreso virgen en la insoladora. Entre el parte inferior de la insoladora y la placa se coloca el fotolito de la cara inferior de la PCB y entre la placa y la parte superior de la insoladora, el fotolito de la cara superior de la PCB. El conjunto de los dos fotolitos tiene que estar correctamente alineado ya que sino no coincidirían las vías y habría que desechar la PCB.

Una vez la máquina esté cerrada y preparada se insola durante un tiempo determinado.

El proceso consiste en que las partes de la PCB virgen que no están tapadas por las partes oscuras del fotolito reciben luz ultravioleta. Esta luz evapora la película de resina con la que está cubierta la PCB virgen. Debajo de esta capa se encuentra la capa de cobre.

Revelado

En esta operación se ataca la placa en proceso de fabricación con un agente revelador. Éste tiene la función de variar la composición química de la resina que no ha sido insolada para que a partir del proceso de revelado, deje de ser fotosensible.

Ataque químico

Una vez revelada la PCB se procede al ataque químico con una solución de agua oxigenada de 110 volúmenes, ácido clorhídrico y agua. Este proceso destruye el cobre que no está tapado por la capa de resina quedando así sólo cobre en las zonas donde interesa, es decir, en las vías, en las pistas y en las huellas de los componentes. Cuando ya no queda rastro de cobre en las zonas

no deseadas se procede al lavado con abundante agua para detener el proceso y eliminar los restos de la solución corrosiva.

Taladrado

Llegados a este punto se procede al taladrado de la PCB. Esta operación se realiza para crear las vías entre las 2 caras así como los agujeros para los tornillos de fijación de la PCB a los vehículos.

Soldado de componentes

Como último paso de fabricación, se sueldan los componentes en su sitio empezando por los componentes más pequeños. En este TFG se ha empezado por el soldado del circuito integrado *bq34z100* (figura 6.12). Luego se sueldan las vías y el resto de componentes y finalmente los conectores.

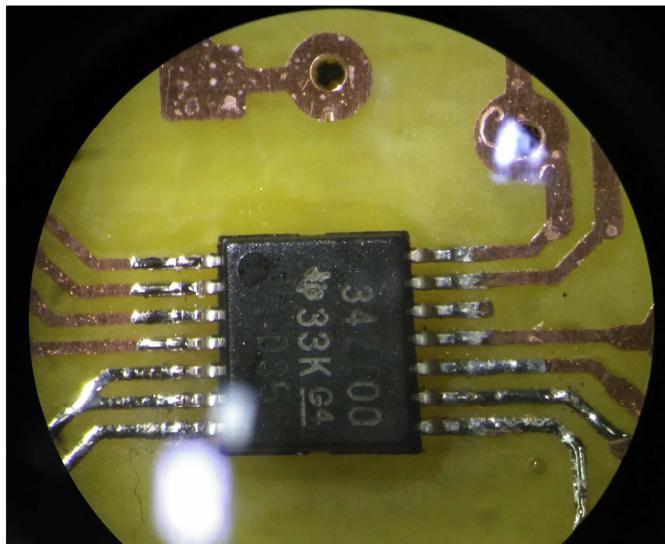


Figura 6.11 Detalle del soldado del chip *bq34z100* al microscopio

Comprobación física de la *PCB*

Una vez finalizada la fabricación de la PCB se realiza una comprobación física de la misma comprobando la continuidad de las pistas y vías con el multímetro.

Barnizado

Para darle un acabado superficial que impida la corrosión del cobre se barniza la PCB. Esto, además, crea una capa aislante que impide el contacto eléctrico de las pistas o componentes con el resto de los vehículos.

7. Integración en el sistema

Integración del hardware

Como objetivo final de este trabajo se tiene la integración de la PCB en un vehículo. Además del circuito impreso de monitorización, se ha integrado la batería y el cargador. Antes de ver el resultado final es necesario reparar en un diagrama de bloques (figura 6.13) para ver cómo se integra todo.

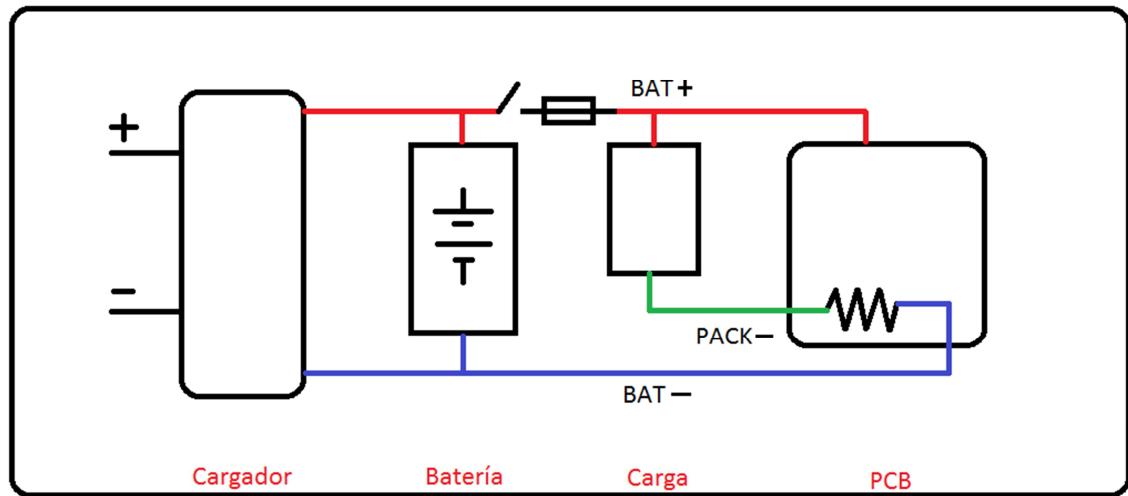
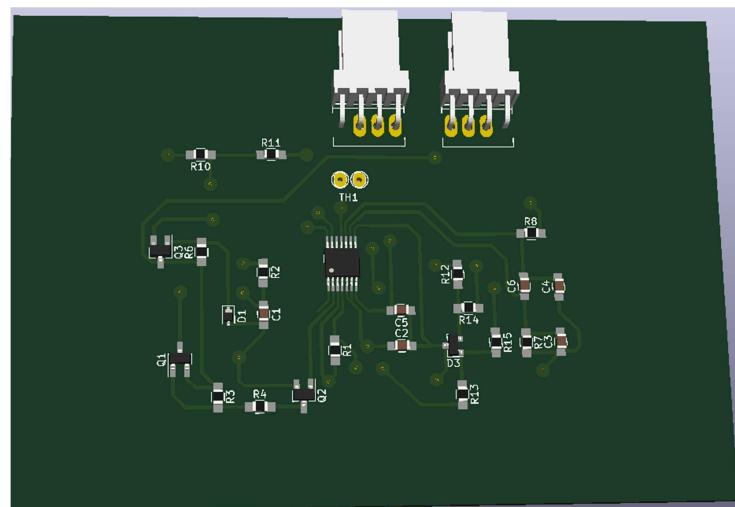


Figura 6.12 Diagrama de bloques del sistema

Como se puede ver en la figura 6.12, la batería y el conjunto carga-resistencia de sensado están en paralelo. También se pueden ver los tres terminales de alimentación de la PCB (Bat+, Bat- y Pack-). Asimismo y como medida de protección se ha instalado un fusible en serie con el interruptor general del UGV. Este fusible protege tanto la carga como la batería, ya que si existe un consumo elevado de corriente abrirá el circuito. La instalación del interruptor general se ha realizado para tener la certeza de desconectar cualquier carga que pueda generar consumo en la batería en condiciones de no utilización de los vehículos.

A continuación se muestran 3 figuras. En la figura 6.13 se puede ver el diseño 3D generado con el software *KiCad*. En la figura 6.14 se muestra el prototipo una vez fabricado y en la figura 6.15 su colocación en el interior de uno de los vehículos.



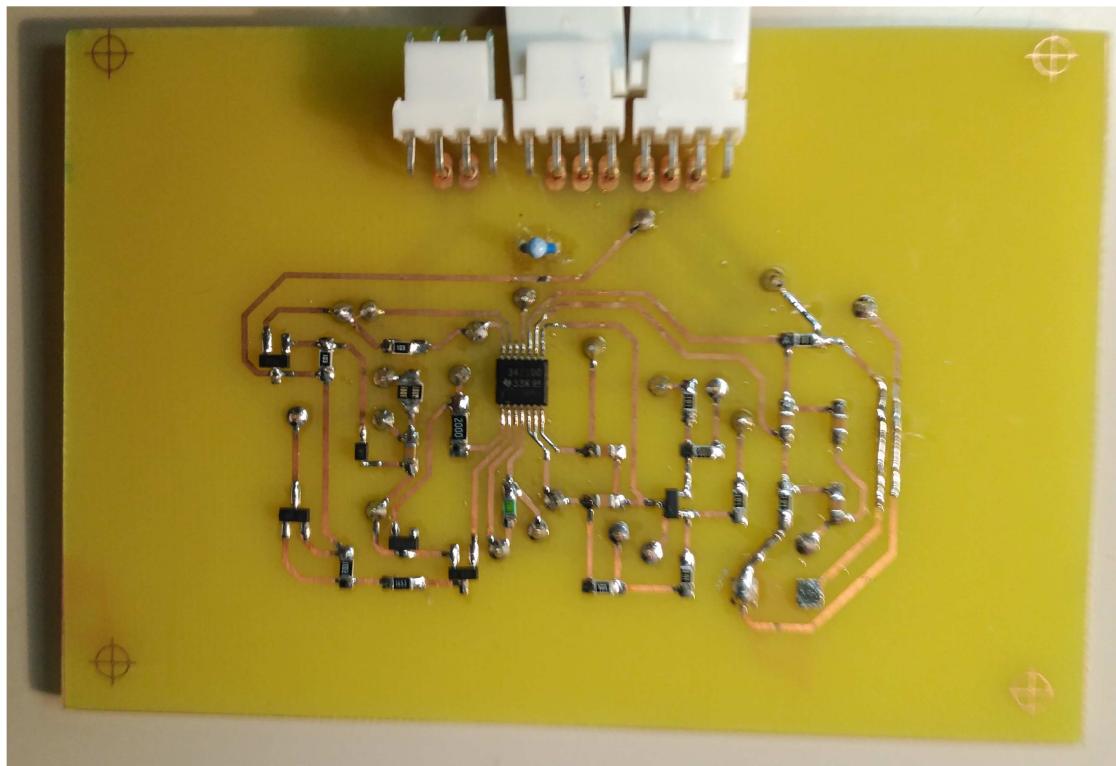


Figura 6.14 Primer prototipo de la PCB diseñada

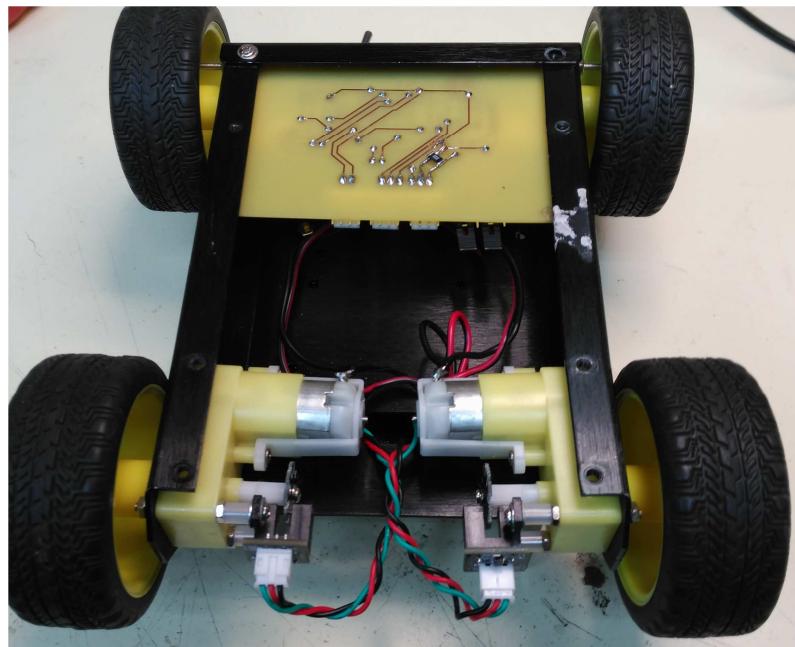


Figura 6.15 Detalle interior del UGV

Para una correcta disposición tanto de la batería como del cargador se ha diseñado un soporte (figura 6.16) utilizando el software SolidWorks 2014. Éste, se instala en la parte delantera, en el sentido de la marcha, y aloja a la batería, en la parte externa y al cargador por la parte interna. El prototipo se ha realizado en madera y posteriormente se ha pintado con spray amarillo.

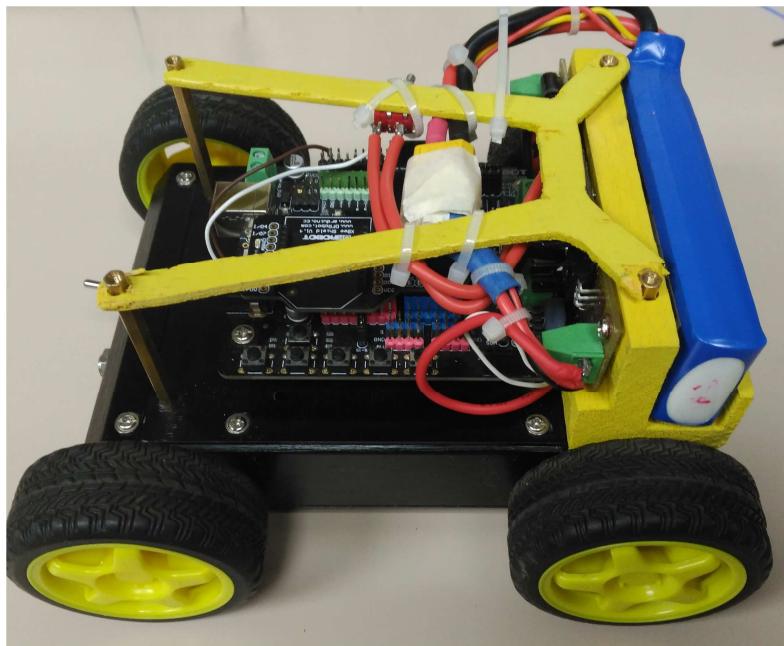


Figura 6.16 Detalle del soporte de la batería y cargador

Integración en el software

Una vez que se ha programado correctamente el chip de la PCB, se ha programado el controlador del vehículo y el controlador central. Toda la programación del sistema *UviSpace* está almacenada en Github (1) y está gestionada por la herramienta *git* de control de versiones. Para una correcta integración en el sistema *UviSpace* ya existente de toda la nueva programación se ha creado una rama específica que contiene la programación de los microcontroladores que gobiernan los UGVs llamada *arduino_fuel_gauge*. De esta manera ha sido posible realizar un seguimiento de los avances y/o problemas que han surgido, así como también de recuperar códigos antiguos para corregir errores además de que es muy útil en proyectos colaborativos como lo es el *UviSpace* ya que se mantiene a los colaboradores al corriente de los avances, mejoras y cambios.

La programación del microcontrolador del vehículo se ha realizado utilizando el *IDE* de *Arduino*. El programa completo se encuentra en el anexo II.

La programación del controlador se estructura en 4 archivos diferentes.

- *UGV.ino*: Contiene el programa principal. En él se realiza una petición de un dato de entre 5 en cada iteración del bucle principal. Los datos solicitados son estado de carga, voltaje, capacidad, temperatura y corriente.
- *BoardParams.h*: Contiene todos las constantes utilizadas por las funciones de los otros archivos.
- *BatteryFunctions.ino*: Contiene una función que se encarga de leer el dato solicitado en el *UGV.ino* realizando la correspondiente consulta a la PCB a través del bus I2C.
- *SerialFunctions.ino*: Contiene las funciones necesarias para, a partir del dato obtenido, acondicionarlo y enviarlo a través del puerto serie al que está conectado el módulo *Xbee* de comunicaciones inalámbrico.

A la hora de integrar la PCB en el sistema *UviSpace* es necesario incluir algunas líneas de código en el programa que se ejecuta en el controlador central. Éste utiliza lenguaje Python. El detalle de la programación que se ha realizado se puede consultar en el Anexo III.

7. Programación del bq34z100

Para configurar el circuito integrado *bq34z100*, se empleó un entorno gráfico suministrado por Texas Instruments, llamado *bqEVSW*. Se empleó la versión 0.9.90 de dicho entorno.

Siguiendo las directrices de uso del manual del chip (3), se han programado los registros de configuración del mismo, en los que se especifican las características eléctricas de las baterías que se han decidido emplear.

El anexo I contiene el código necesario para realizar la programación. Dicho código es migrable, por lo que se podrá emplear con el mismo objetivo en diferentes plataformas.

Fase I

Se ha comenzado por la programación de la placa de evaluación siguiendo las directrices del manual (3). Antes de conectar la placa EVM al ordenador ha sido necesario configurar los *jumpers* para la aplicación concreta. Después de esa comprobación se ha procedido a conectar la EVM al ordenador por medio del adaptador I2C-USB. Antes de arrancar el programa de *Texas Instruments* en el PC ha sido necesario alimentar la PCB con una tensión de 7 V con la fuente de alimentación para que ésta sea reconocida por el PC. Se ha elegido ese valor de tensión ya que es un valor próximo al nominal de las baterías seleccionadas.

Hecho esto se ha procedido al inicio del programa en el PC. Éste reconoce automáticamente la EVM. Una vez iniciado se ha leído la memoria flash del circuito integrado (figura 7.1) y se han verificado los campos para editarlos acorde la aplicación concreta. Los pasos están completamente detallados en la documentación del *bq34z100* (3).

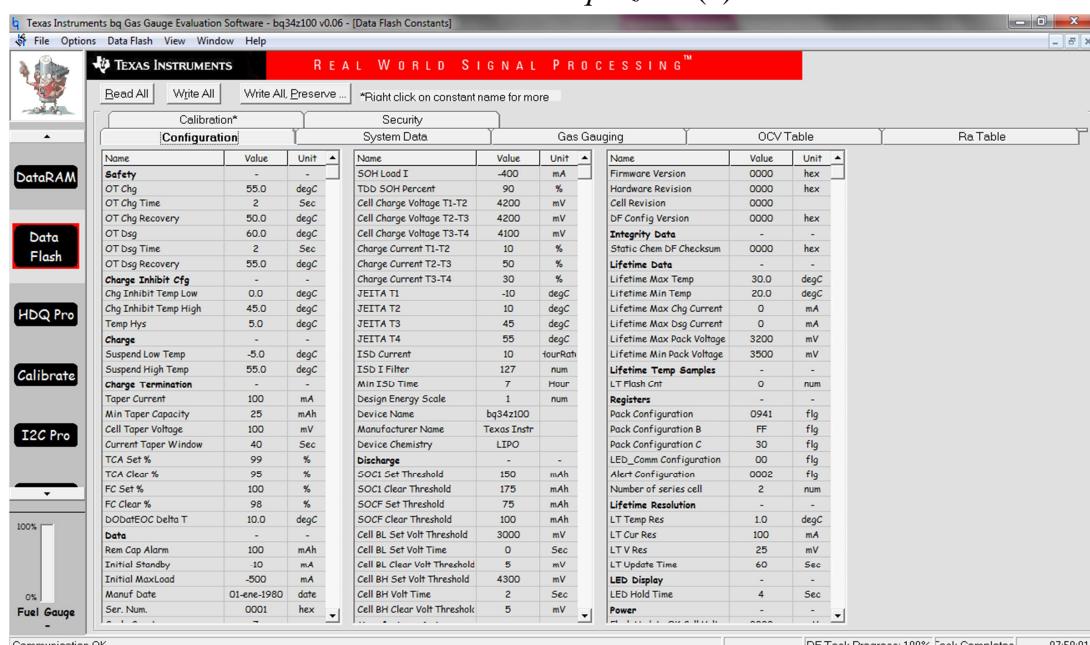


Figura 7.1 Configuración de la memoria flash

Finalizado este paso de edición de registros de la memoria flash se ha procedido a la calibración del chip para la aplicación concreta. Primeramente ha sido necesario saber que la EVM debe estar alimentada solo en tensión. Cumpliéndose esto y en la pestaña de calibración se ha iniciado el proceso de calibración del contador de culombios. En la figura 7.2 se puede ver el tick de verificación del proceso terminado.

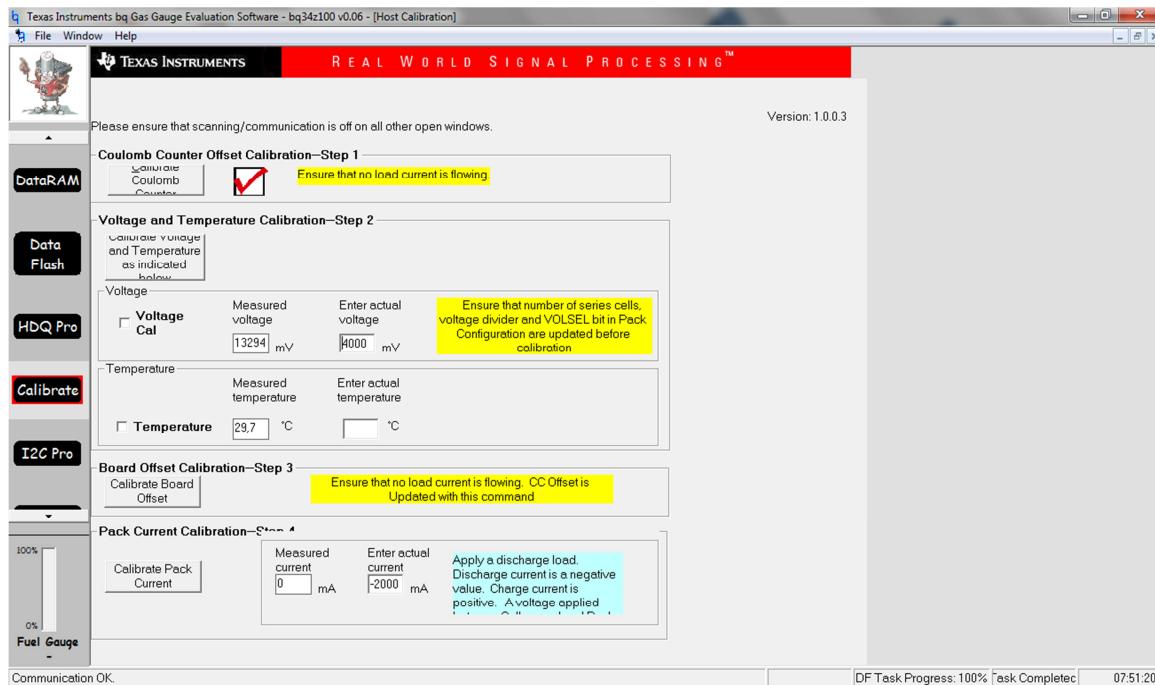


Figura 7.2 Calibración del contador de culombios

Luego ha sido necesario calibrar el voltaje y la temperatura. Para ello hay que disponer de medidas lo más precisas posibles para introducirlas en los campos de medida. El software contrastará el valor introducido con el medido (figura 7.3).

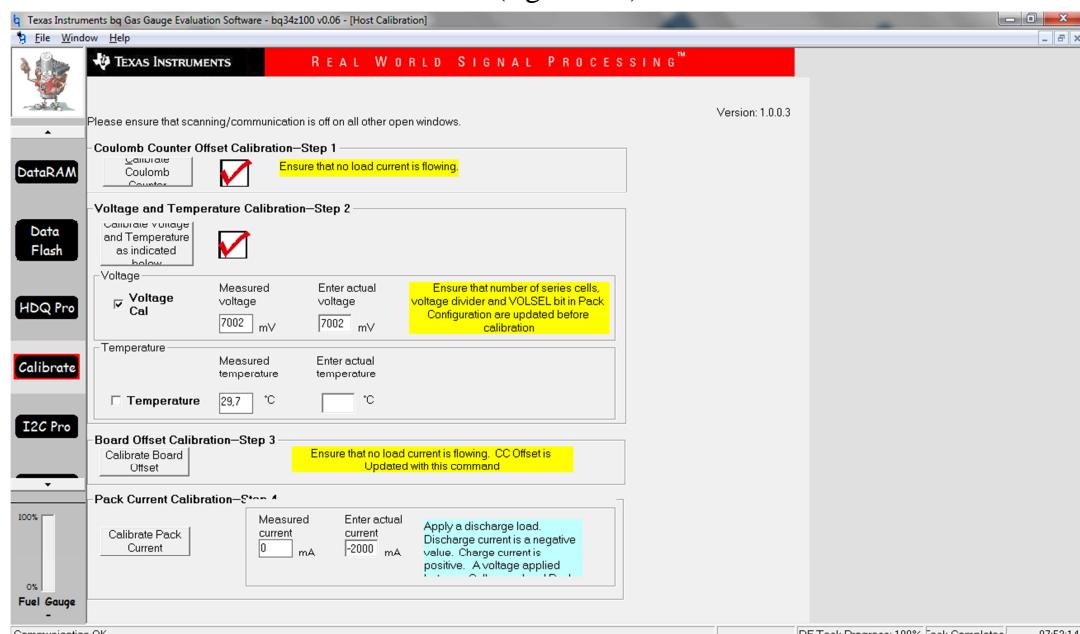


Figura 7.3 Calibración de voltaje y temperatura

Por último, se ha realizado la calibración de offset y la de corriente. Para esta última ha sido necesario simular una corriente de descarga a través de la EVM con un valor aproximado de 1A. En la figura 7.5 se puede ver el resultado de la calibración.

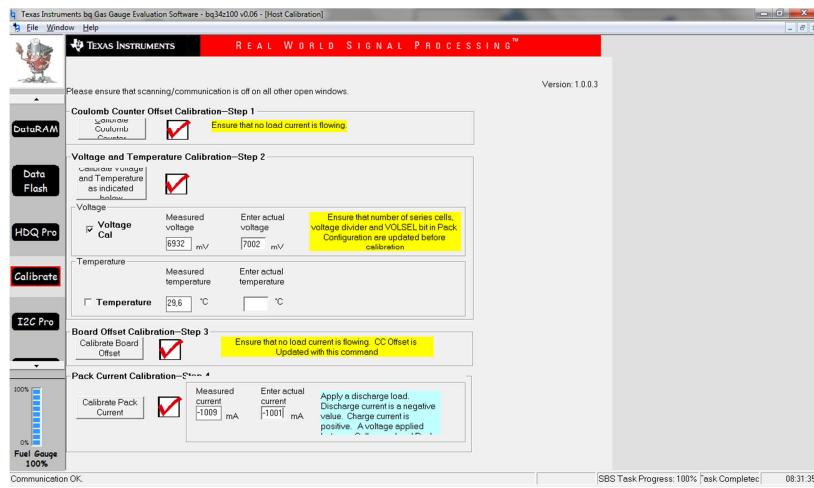


Figura 7.4 Calibración de offset y de corriente

Una vez realizados todos los pasos de calibración, el programa ha editado automáticamente los valores de los registros de la memoria flash del circuito integrado. Esta información se ha exportado en un fichero cuya extensión es “.gg” y que será útil más adelante.

Fase II

Una vez llegados a este punto no sería práctico tener que repetir todo el proceso anterior para cada placa que se quisiese programar. Así pues, se conecta, en las mismas condiciones que se ha hecho con la EVM, la PCB al ordenador y se importa el fichero de configuración citado anteriormente. De esta manera solo es necesario realizar la calibración.

Fase III

Realizada la programación y calibración de la PCB objeto de este TFG se ha procedido a conectarla a un UGV e intentar tomar una medida del estado de carga. El resultado ha sido un valor correcto que implica que la placa ha sido correctamente diseñada y fabricada.

8. Validación del sistema

Para asegurar el correcto funcionamiento del sistema se ha realizado tanto la validación de la PCB como la validación de la correcta integración en el sistema.

Validación de la PCB

Es necesario comprobar que la PCB realiza correctamente las lecturas de los valores deseados. Para ello se ha realizado el siguiente experimento.

Con la ayuda del *software* de *Texas Instruments* se ha comprobado la dispersión entre los valores leídos por la PCB y por la EVM. Esta comprobación se ha realizado para un ciclo de descarga y para un ciclo de carga.

En la tabla 8.1 se pueden ver los datos obtenidos de la realización del experimento de descarga. Se han recogido valores del estado de carga (SOC), de la capacidad restante, voltaje entre terminales de la batería, corriente circulante por el circuito de sensado y temperatura. Los valores de corriente son negativos. Esto indica que es una corriente de descarga. Se puede observar que si bien las medidas tomadas con la placa de evaluación EVM y con la PCB son distintas, ambos valores son muy próximos.

En la figura 8.2 se ha representado el estado de carga, parámetro que más interesa, obtenido por el módulo de evaluación (color azul) y por la PCB diseñada (color rojo). Se observa que ambas placas tienen lecturas muy similares y que no existe mucho grado de dispersión entre ambas.

Tiempo (minutos)	EVM	PCB	EVM	PCB	EVM	PCB	EVM	PCB	EVM	PCB
	SOC (%)		Remaining Capacity (mAh)		Voltage (mV)		Current (mA)		Temperature (°C)	
0	100	100	2125	2154	8366	8370	-140	10	30	29,5
5	98	98	2106	2098	8204	8278	-758	-804	30,2	29,2
15	93	93	1985	1984	8096	8212	-794	-847	30,2	29
20	90	91	1933	1940	8052	8066	-794	-787	30,2	29,3
25	88	88	1884	1890	8022	8024	-766	-781	30	28,9
30	86	86	1838	1852	7992	7954	-770	-775	30	29
35	84	83	1797	1781	7930	7952	-754	-708	29,9	28,8
40	81	80	1737	1714	7892	7888	-804	-822	29,5	28,7
45	78	77	1661	1642	7834	7838	-847	-856	29,6	28,5
50	75	75	1595	1604	7814	7774	-810	-821	29,6	28,5
55	73	73	1559	1556	7786	7796	-761	-777	29,6	28,5
60	71	71	1511	1508	7760	7722	-761	-787	29,6	28,5
70	64	64	1347	1373	7686	7696	-750	-764	29,4	28,2
80	59	59	1266	1265	7622	7630	-728	-742	29,3	28,3
90	53	55	1106	1181	7564	7568	-750	-769	29,2	28,2
100	50	50	1067	1074	7520	7526	-736	-746	29,1	28,1
110	45	45	959	953	7486	7492	-710	-733	28,9	27,9
120	38	39	810	832	7440	7452	-730	-744	28,6	27,8
130	33	31	699	666	7420	7418	-718	-766	28,6	27,8
140	26	25	551	524	7386	7278	-710	-742	28,4	27,7
150	19	20	396	421	7352	7352	-703	-745	28,3	27,5
155	18	19	376	395	7340	7340	-710	-724	28,3	27,3
160	17	18	356	367	7316	7326	-702	-730	28,3	27,4
165	16	16	330	339	7300	7302	-706	-736	28,3	27,4
170	14	15	296	310	7278	7284	-698	-717	28,3	27,4
175	12	11	257	231	7250	7252	-705	-736	28,2	27,2
180	9	8	182	165	7224	7224	-711	-745	28,1	27,2
185	6	8	122	153	7200	7162	-722	-764	28,1	27,2
190	6	7	125	139	7186	7186	-733	-760	28	27,3
195	5	3	93	63	7062	7074	-730	-780	28,2	27,1
200	2	2	27	36	6900	6904	-744	-776	28	27,1
205	1	1	16	19	6800	6802	-756	-778	27,9	27
210		0		0		6404		-50		29,8

Figura 8.1 Datos del experimento de descarga

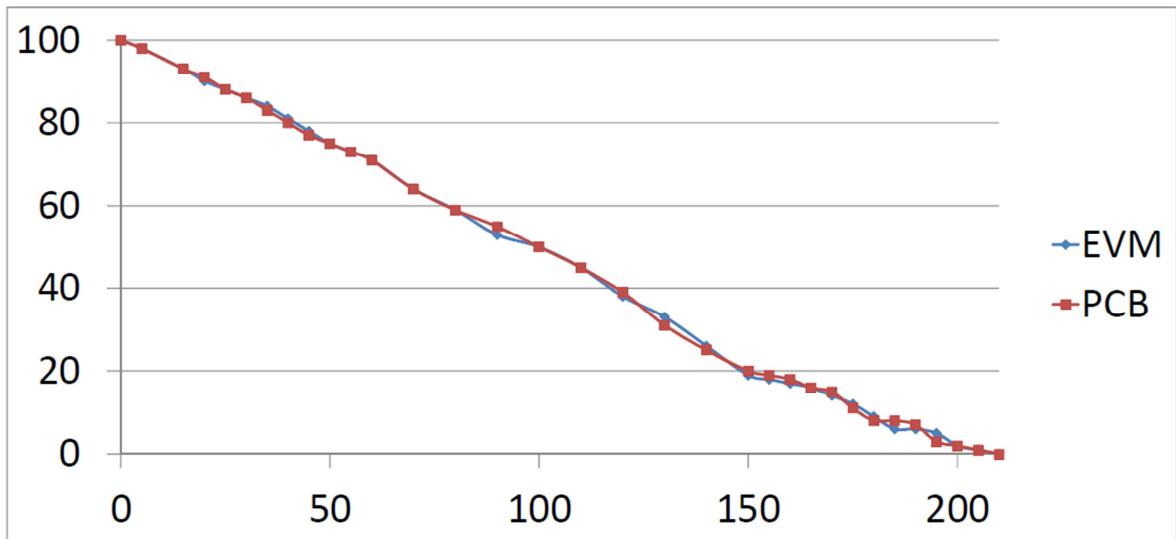


Figura 8.2 Evolución del SOC en el experimento de descarga

En la tabla 8.3 se pueden ver los datos obtenidos en el experimento de carga. Se han recogido los mismos parámetros que en experimento anterior. De igual manera se puede concluir que los valores, si bien no son idénticos, no existe mucha diferencia entre los obtenidos con la placa de evaluación y con la PCB.

En la figura 8.4 se ha representado el estado de carga obtenido por el módulo de evaluación (color azul) y por la PCB (color rojo). Se vuelve a comprobar que ambas placas tienen lecturas de valores muy similares.

Tiempo (minutos)	EVM		PCB		EVM		PCB		EVM		PCB	
	SOC (%)	Remaining Capacity (mAh)	Voltage (mV)	Current (mA)	Temperature (°C)							
0	1	1	6560	-52	29,7							
6	16	15	7408	-53	29,8							
11	17	17	7428	-54	29,6							
16	35	32	7556	-54	32							
26	51	53	7632	-54	30,7							
31	55	54	7668	-54	31,6							
37	58	57	7718	-54	31							
43	61	63	7764	-54	30,5							
48	68	67	7844	-54	30,4							
58	76	77	7956	-54	30,9							
68	82	83	8042	-54	31,7							
78	92	92	8208	-54	32							
83	97	98	8308	-54	31							
88	100	100	8388	-54	31,3							

Figura 8.3 Datos del experimento de carga

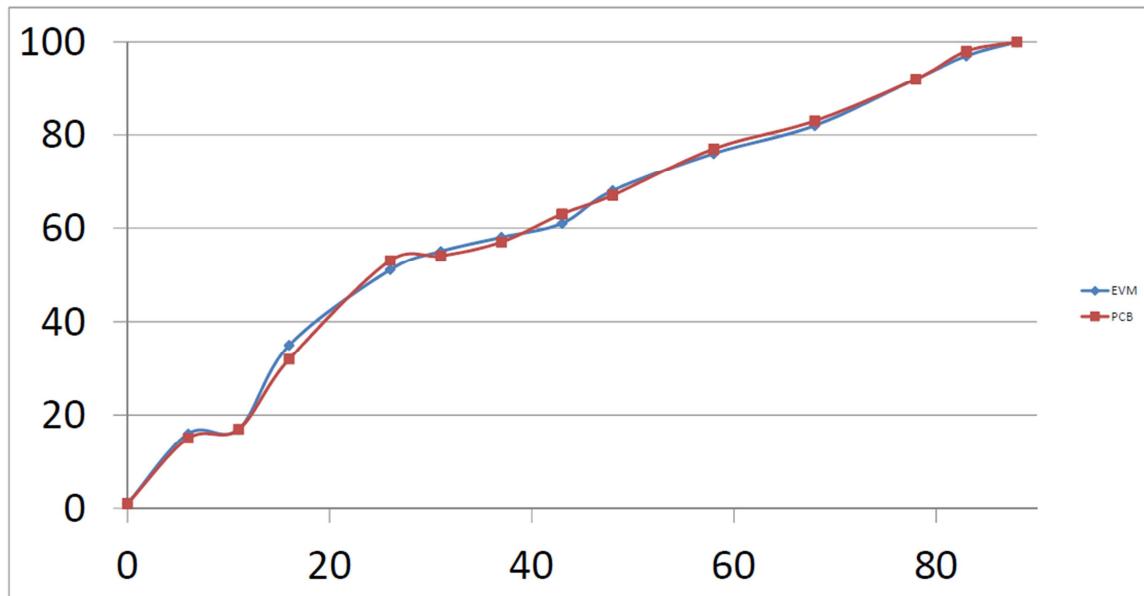


Figura 8.4 Evolución del SOC en el experimento de carga

Validación de la integración

Para validar la integración de la placa diseñada en el *UviSpace* es necesario comprobar que tanto el controlador del UGV como el controlador central pueden realizar lecturas válidas.

En el primer caso se ha generado una versión del código implementado en el controlador del UGV (arduino) en el que se imprimen valores del estado de carga, capacidad restante, voltaje entre bornes de la batería, corriente circulante por la carga y temperatura por pantalla para que sean visualizados por el usuario. Se han obtenido varias lecturas válidas con lo que ha quedado demostrado el correcto funcionamiento del controlador. Es capaz de solicitar datos y de recibir respuestas de manera correcta.

En el segundo caso, ha sido necesaria la adaptación de 2 *scripts* que se ejecutan en el controlador central (en *Python*) para poder recibir, a través del módulo *Xbee*, la información solicitada, gestionarla y mostrarla al usuario. Se han obtenido varias lecturas válidas de todos los parámetros con lo que se puede concluir que se ha realizado la integración de la PCB en el *UviSpace* correctamente.

9. Conclusiones

Tras la realización del trabajo se han conseguido alcanzar los siguientes objetivos:

- ❖ La compatibilidad entre las baterías de polímero de litio de tensión nominal 7.4V CC y los vehículos del *UviSpace* es total. Así, los UGVs pueden trabajar con al menos las mismas condiciones de operación que cuando son alimentados mediante la fuente de alimentación externa.
- ❖ La capacidad de las baterías seleccionadas, 2200 mAh, es suficiente como para dotar de alimentación a los UGVs durante al menos un hora. Este tiempo es suficiente para realizar los trabajos habituales que se llevan a cabo en el laboratorio.

- ❖ La PCB diseñada es compatible con el modelo de baterías seleccionado. Ésta es capaz de transmitir todos los parámetros eléctricos de dicho modelo durante todo el rango de trabajo así como de comunicarse correctamente con el microcontrolador que gobierna los UGVs.
- ❖ El programa de *Arduino* diseñado es capaz, una vez compilado y cargado en las placas de los vehículos, de solicitar, recibir y enviar al resto del sistema *UviSpace* los parámetros eléctricos de las baterías recogidos por la PCB implementada.
- ❖ La integración de todos los elementos diseñados en los vehículos del *UviSpace* es factible y no implica ninguna modificación de las dimensiones globales de los vehículos.

10. Líneas Futuras

Se desea independizar la gestión de la alerta, ya sea por nivel de tensión por debajo de los límites o por un exceso de temperatura, del funcionamiento normal del controlador. Para ello se puede diseñar en la PCB un circuito que lleve una señal a los interruptores generales de los vehículos. Estos deberán ser interruptores pilotados que permitan su apertura si se da la condición de alerta.

Como funcionalidad extra se desean colocan 3 diodos LED en la trasera de los vehículos que muestren el estado de carga. Para ello se ha elegido un LED verde que se enciende mientras que el estado de carga de la batería sea superior al 40%, un LED rojo que se enciende cuando el estado de carga es menor al 15% y un LED amarillo para el resto de valores del estado de carga.

Después de realizar las mejoras, se pretende implementar el sistema en uno de los UGVs de tal manera que quede totalmente integrado con una solución final.

11. Bibliografía

1. Documentación UviSpace. [En línea] <http://uvispace.readthedocs.io/en/latest/>
2. Documentación DFRobots. [En línea]
[https://www.dfrobot.com/wiki/index.php/4WD_Mobile_Platform_\(SKU:ROB0003\)](https://www.dfrobot.com/wiki/index.php/4WD_Mobile_Platform_(SKU:ROB0003))
3. Texas Instruments *bq34z100*. [En línea] <http://www.ti.com/lit/ds/symlink/bq34z100.pdf>
4. Texas Instruments EVM. [En línea] <http://www.ti.com/lit/ug/slau904a/slau904a.pdf>
5. Manual KiCad primeros pasos. [En línea]
http://docs.kicad-pcb.org/stable/en/getting_started_in_kicad.pdf

Escuela de Ingeniería Industrial

TRABAJO FIN DE GRADO

*Diseño, implementación y validación de un circuito
electrónico para la medida del nivel de carga de baterías en
robots móviles*

Grado en Ingeniería en Electrónica Industrial y Automática

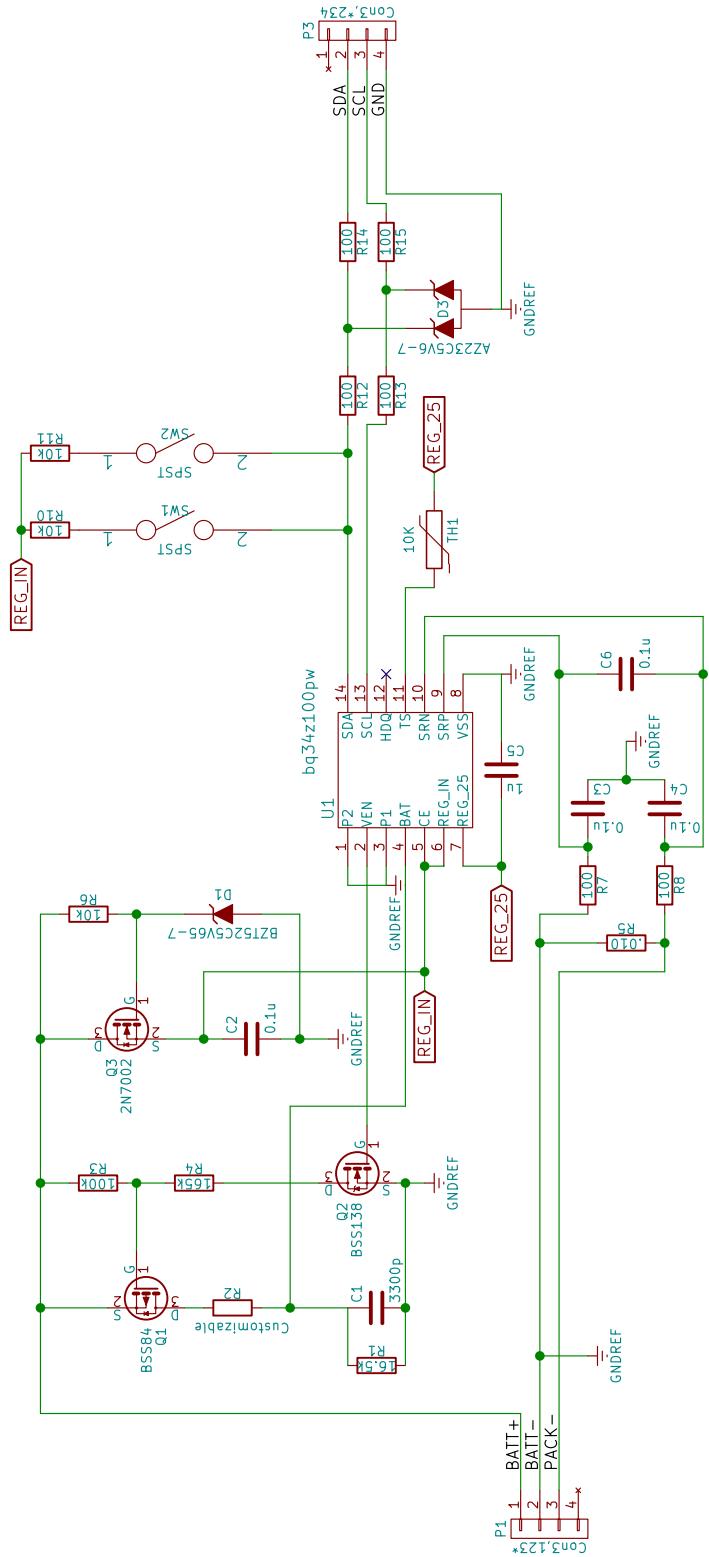
Documento

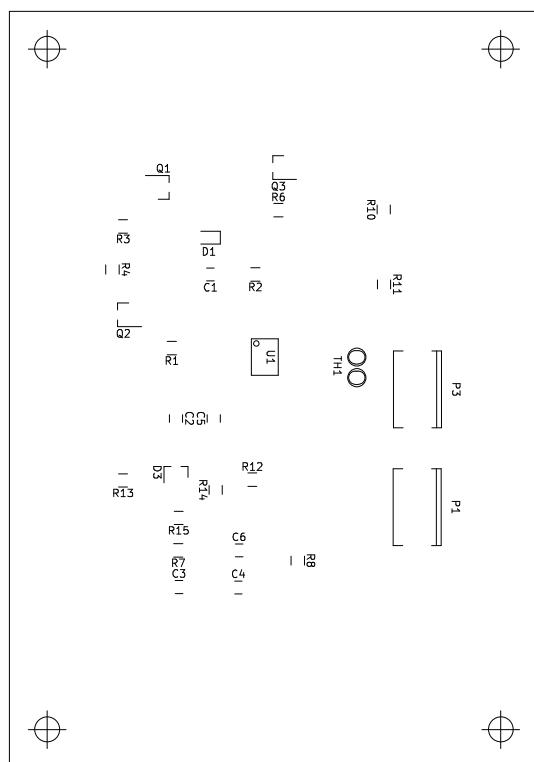
PLANOS

UniversidadeVigo

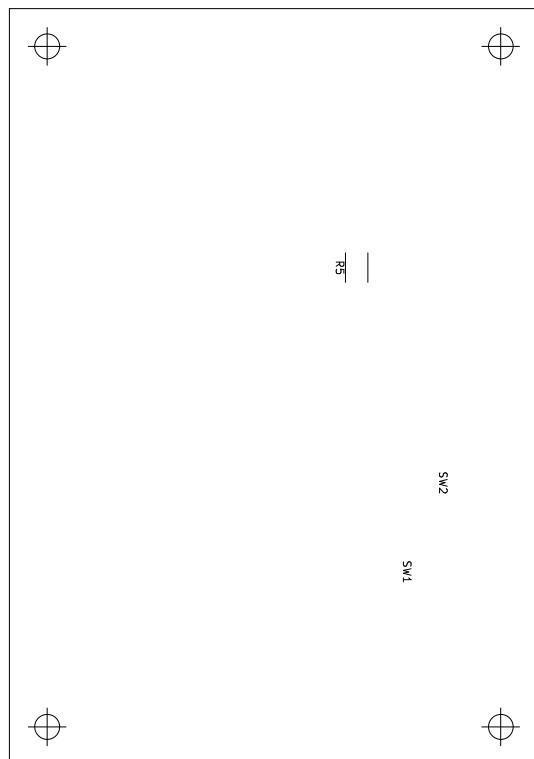
II. PLANOS

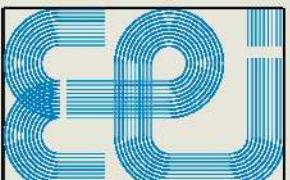
PLANO 1: ESQUEMA ELÉCTRICO.....	36
PLANO 2: SERIGRAFÍA CARA SUPERIOR.....	37
PLANO 3: SERIGRAFÍA CARA INFERIOR.....	38
PLANO 4: MONTAJE CARA SUPERIOR.....	39
PLANO 5: MONTAJE CARA INFERIOR.....	40
PLANO 6: CAPA DE COBRE CARA SUPERIOR.....	41
PLANO 7: CAPA DE COBRE CARA INFERIOR.....	42
PLANO 8: SOPORTE CARGADOR Y BATERIA.....	43

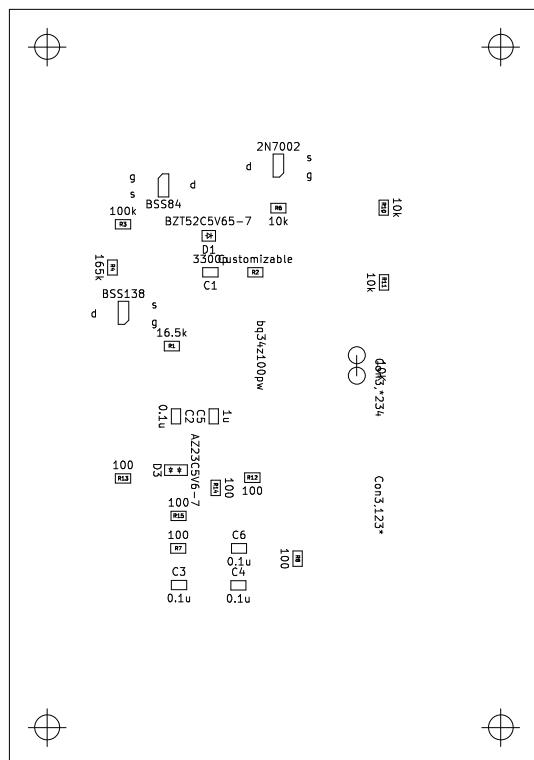




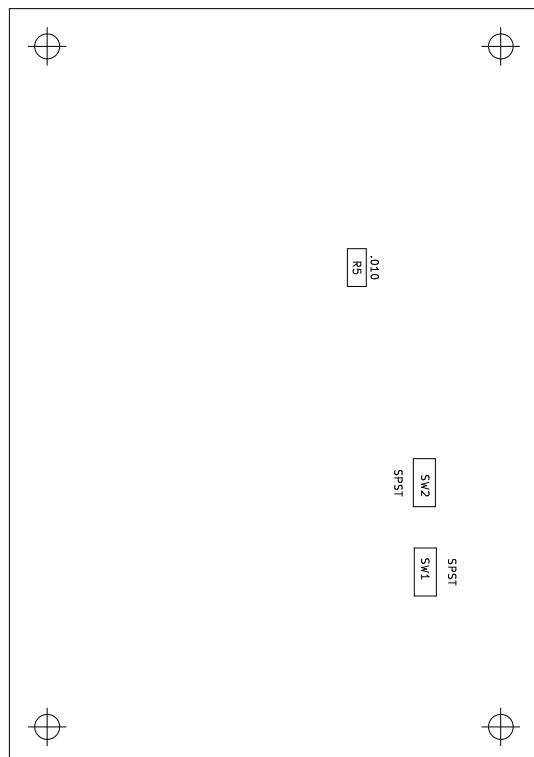
INGENIERO	PROYECTO	 Universidade Vigo
ANGEL SOTO BOULLOSA	PROYECTO: DISEÑO, IMPLEMENTACIÓN Y VALIDACIÓN DE UN CIRCUITO ELECTRÓNICO PARA LA MEDIDA DEL NIVEL DE CARGA DE BATERÍAS EN ROBOTS MÓVILES	PETICIONARIO: E.E.I. Universidad de Vigo
REFERENCIA	SITUACIÓN: Campus Lagoas-Marcosende Vigo. Pontevedra	PLANO
SILKS_F	SERIGRAFIA CARA SUPERIOR	ESCALA
FECHA	9 / JULIO / 2017	Nº DE PLANO
		2/8



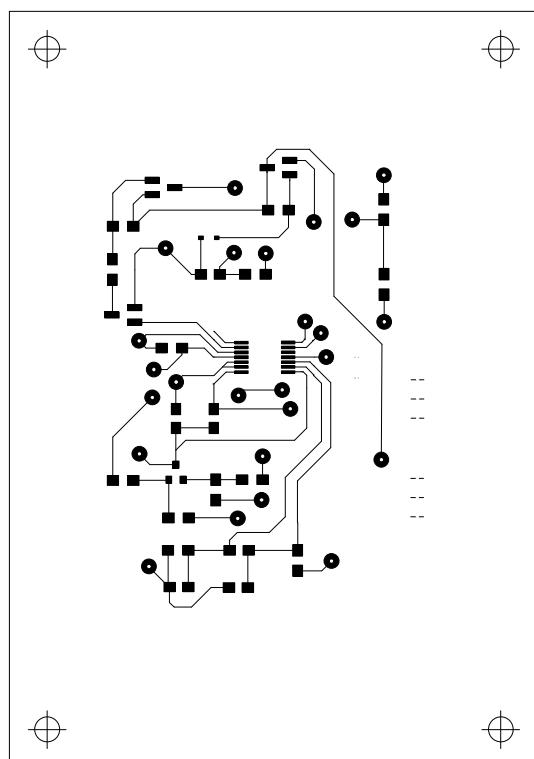
INGENIERO	PROYECTO	 Universidade Vigo
ANGEL SOTO BOULLOSA	PROYECTO: DISEÑO, IMPLEMENTACIÓN Y VALIDACIÓN DE UN CIRCUITO ELECTRÓNICO PARA LA MEDIDA DEL NIVEL DE CARGA DE BATERÍAS EN ROBOTS MÓVILES	
REFERENCIA	PETICIONARIO: E.E.I. Universidad de Vigo	
SILKS_B	SITUACIÓN: Campus Lagoas-Marcosende Vigo. Pontevedra	PLANO
FECHA	SERIGRAFIA CARA INFERIOR	ESCALA
9 / JULIO / 2017		1:1
		Nº DE PLANO
		3/8



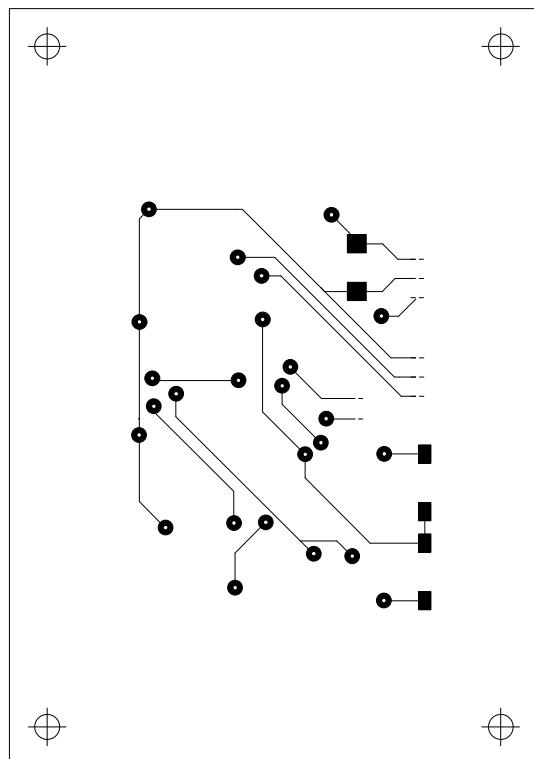
INGENIERO	PROYECTO	
ANGEL SOTO BOULLOSA	PROYECTO: DISEÑO, IMPLEMENTACIÓN Y VALIDACIÓN DE UN CIRCUITO ELECTRÓNICO PARA LA MEDIDA DEL NIVEL DE CARGA DE BATERÍAS EN ROBOTS MÓVILES	
REFERENCIA	PETICIONARIO: E.E.I. Universidad de Vigo	ESCALA 1:1 Nº DE PLANO 4/8
FAB_F	SITUACIÓN: Campus Lagoas-Marcosende Vigo. Pontevedra	
FECHA	PLANO	
9 / JULIO / 2017	MONTAJE CARA SUPERIOR	



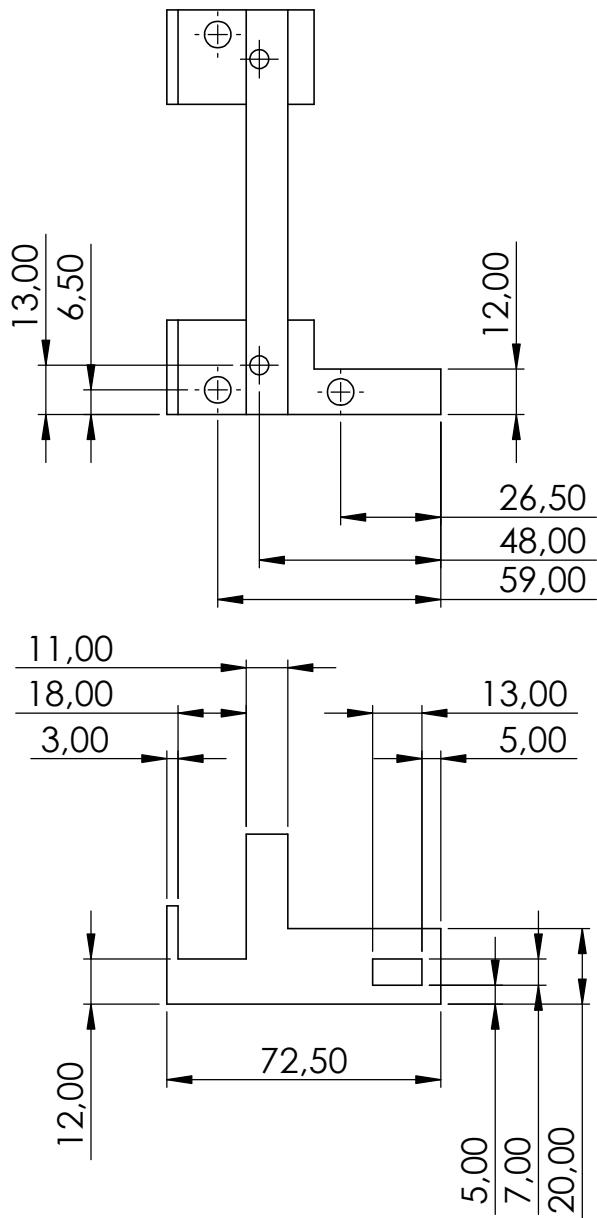
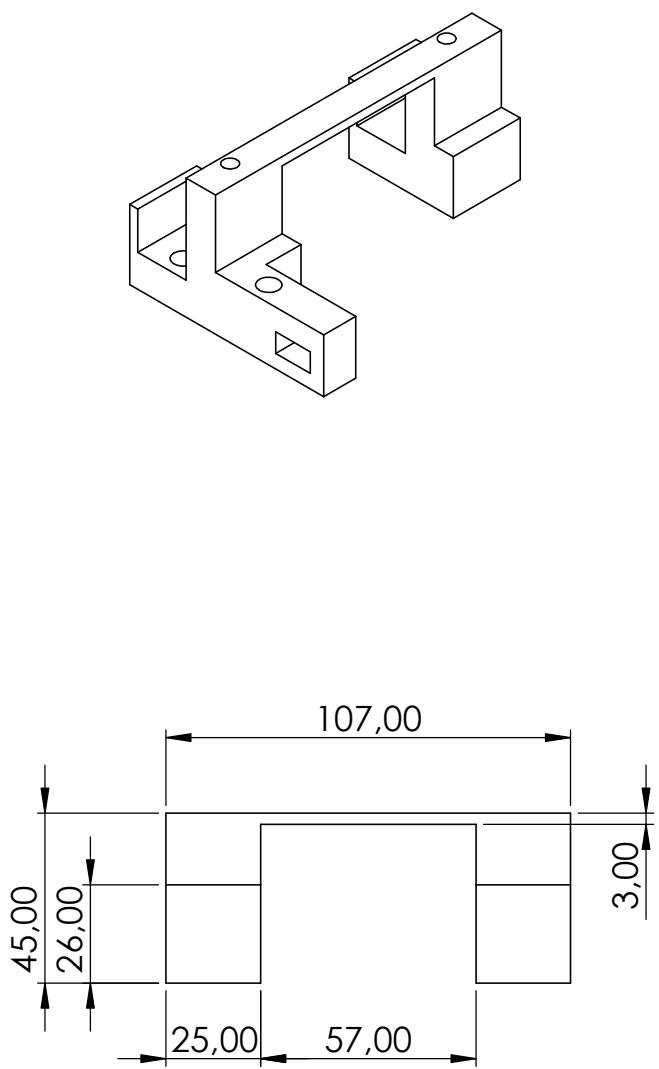
INGENIERO	PROYECTO	 Universidade de Vigo
ANGEL SOTO BOULLOSA	DISEÑO, IMPLEMENTACIÓN Y VALIDACIÓN DE UN CIRCUITO ELECTRÓNICO PARA LA MEDIDA DEL NIVEL DE CARGA DE BATERÍAS EN ROBOTS MÓVILES	
REFERENCIA	PETICIONARIO: E.E.I. Universidad de Vigo	
FAB_B	SITUACIÓN: Campus Lagoas-Marcosende Vigo. Pontevedra	
FECHA	PLANO	ESCALA
9 / JULIO / 2017	MONTAJE	1:1
	CARA INFERIOR	Nº DE PLANO
		5/8



INGENIERO	PROYECTO	
ANGEL SOTO BOULLOSA	PROYECTO: DISEÑO, IMPLEMENTACIÓN Y VALIDACIÓN DE UN CIRCUITO ELECTRÓNICO PARA LA MEDIDA DEL NIVEL DE CARGA DE BATERÍAS EN ROBOTS MÓVILES	 Universidade Vigo
REFERENCIA	PETICIONARIO: E.E.I. Universidad de Vigo	ESCALA
CU_F	SITUACIÓN: Campus Lagoas-Marcosende	1:1
FECHA	Vigo. Pontevedra	Nº DE PLANO
9 / JULIO / 2017	CAPA DE COBRE	6/8
	CARA SUPERIOR	



INGENIERO	PROYECTO	 Universidade de Vigo
ANGEL SOTO BOULLOSA	DISEÑO, IMPLEMENTACIÓN Y VALIDACIÓN DE UN CIRCUITO ELECTRÓNICO PARA LA MEDIDA DEL NIVEL DE CARGA DE BATERÍAS EN ROBOTS MÓVILES	
REFERENCIA	PETICIONARIO: E.E.I. Universidad de Vigo	ESCALA
CU_B	SITUACIÓN: Campus Lagoas-Marcosende Vigo. Pontevedra	
FECHA	PLANO	1:1
9 / JULIO / 2017	CAPA DE COBRE CARA INFERIOR	Nº DE PLANO
		7/8



INGENIERO

ANGEL SOTO BOULLOSA

REFERENCIA

SOPORTE

FECHA

9 / JULIO / 2017

PROYECTO

PROYECTO: DISEÑO, IMPLEMENTACIÓN Y VALIDACIÓN
DE UN CIRCUITO ELECTRÓNICO PARA LA
MEDIDA DEL NIVEL DE CARGA DE BATERÍAS
EN ROBOTS MÓVILES

PETICIONARIO: E.E.I. Universidad de Vigo

SITUACIÓN: Campus Lagoas-Marcosende
Vigo. Pontevedra

PLANO

SOPORTE CARGADOR Y BATERIA



ESCALA

1:2

Nº DE PLANO

8/8

Escuela de Ingeniería Industrial

TRABAJO FIN DE GRADO

*Diseño, implementación y validación de un circuito
electrónico para la medida del nivel de carga de baterías en
robots móviles*

Grado en Ingeniería en Electrónica Industrial y Automática

Documento

PLIEGO DE CONDICIONES

Universidad de Vigo

INDICE

INTRODUCCIÓN.....	46
CONDICIONES DE USO.....	46

III. PLIEGO DE CONDICIONES

1. INTRODUCCIÓN

En el pliego de condiciones se especifican las condiciones técnicas para la correcta implementación de baterías de polímero de litio en el *UviSpace* y la correcta monitorización de sus parámetros de estado.

2. CONDICIONES DE USO

El sistema ha sido diseñado para su uso en el *UviSpace*. Por tanto su utilización queda restringida a sistemas eléctricos o electrónicos con unas especificaciones eléctricas similares a las del *UviSpace*; además de unas condiciones ambientales parecidas (restringido a zonas interiores con atmósferas no agresivas).

Escuela de Ingeniería Industrial

TRABAJO FIN DE GRADO

*Diseño, implementación y validación de un circuito
electrónico para la medida del nivel de carga de baterías en
robots móviles*

Grado en Ingeniería en Electrónica Industrial y Automática

Documento

PRESUPUESTO

UniversidadeVigo

INDICE

INTRODUCCIÓN.....	49
PRESUPUESTO.....	49

IV. PRESUPUESTO

1. INTRODUCCION

En este documento se analizan los costes que se consideran en el desarrollo del diseño, implementación y validación de un circuito electrónico para la medida del nivel de carga de baterías en robots móviles.

Sólo se tendrán en cuenta los costes de materiales cuyas magnitudes serán expresadas en euros.

2. PRESUPUESTO

1. Componentes

Los componentes del circuito electrónico se relacionan a continuación:

Componente	Valor	Referencia Mouser	Precio unitario	Unidades	Subtotal
Conector	Molex 4	538-22-05-3041	0,439	2,00	0,878
Resistencia	16,5 K	667-ERJ-U06F1652V	0,242	1	0,242
Resistencia	150 K	667-ERJ-PB6B1503V	0,519	1	0,519
Resistencia	100 K	667-ERJ-PB6B1003V	0,519	1	0,519
Resistencia	165 K	667-ERA-6AEB1653V	0,564	1	0,564
Resistencia	0,01	756-LRMAT2010R01FT4	0,43	1	0,43
Resistencia	10 K	660-RN73H2ATD1002B25	0,795	3	2,385
Resistencia	100	660-RN73H2ATD1000B10	0,477	6	2,862
Condensador	3300 pF	80-C0805C332F3G	1,73	1	1,73
		80-C0805C104J5RAUTO			
Condensador	0,1 uF		0,197	4	0,788
Condensador	1 uF	80-C0805C105K4R	0,107	1	0,107
Transistor	BSS84	512-BSS84	0,206	1	0,206
Transistor	BSS138	512-BSS138	0,197	1	0,197
Transistor	2N7002	512-2N7002	0,242	1	0,242
Diodo Zener	BZT52C5V65-7	621-BZT52C5V6T-7	0,179	1	0,179
Diodo Zener	AZ23C5V6-7	621-AZ23C5V6-F	0,367	1	0,367
Thermistor	10 K	594-NTCLE203E3103GBO	0,528	1	0,528
2 interruptores	774-204211ST	774-204211ST	1,01	2	2,02
IC	bq 34z100		5,7	1	5,7
Portafusible		567-03453LS1H	2,86	1	2,86
Fusible		530-5MT3-R	0,206	1	0,206
				Total	23,529

2. Soporte batería y cargador

La fabricación del soporte de la batería y el cargador se encarga a la empresa Lupeon, S.L. de impresión en 3D cuyos datos son los siguientes:

B70367818

Polígono Industrial Porto do Molle, Nave 7^a

Oficinas Pontes 4, Buzón 29

36350 – Nigrán

Off: +34 986118131

admin@lupeon.com

El presupuesto que presenta es el siguiente:

Pieza	Tecnología	Material	Precio
Soporte batería y cargador	FDM	ABS	42,00

3. Presupuesto total

Así pues, la realización de este proyecto conlleva un coste de materiales de $42+23,529 =$
65,529 euros en total.

Escuela de Ingeniería Industrial

TRABAJO FIN DE GRADO

*Diseño, implementación y validación de un circuito
electrónico para la medida del nivel de carga de baterías en
robots móviles*

Grado en Ingeniería en Electrónica Industrial y Automática

Documento

ANEXOS

DISEÑO, IMPLEMENTACIÓN Y VALIDACIÓN DE UN CIRCUITO ELECTRÓNICO PARA LA MEDIDA DEL NIVEL

DE CARGA DE BATERÍAS EN ROBOTS MÓVILES

ÁNGEL SOTO BOULLOSA

V. ANEXOS

Anexo I: FICHERO GG DE CONFIGURACION DE LA PCB.....	54
Anexo II: CODIGO ARDUINO.....	60
Anexo III: Python.....	72

1. ANEXO I

INDICE

FICHERO GG DE CONFIGURACIÓN DE LA PCB.....	55
--	----

FICHERO GG DE CONFIGURACIÓN DE LA PCB

[Header]

bq EVSW Version = 0.9.90

DeviceName = bq34z100 v0.06

Time = 13/06/2017 19:31:35

[Safety(Configuration)]

OT Chg = 55.0

OT Chg Time = 2

OT Chg Recovery = 50.0

OT Dsg = 60.0

OT Dsg Time = 2

OT Dsg Recovery = 55.0

[Charge Inhibit Cfg(Configuration)]

Chg Inhibit Temp Low = 0.0

Chg Inhibit Temp High = 45.0

Temp Hys = 5.0

[Charge(Configuration)]

Suspend Low Temp = -5.0

Suspend High Temp = 55.0

[Charge Termination(Configuration)]

Taper Current = 100

Min Taper Capacity = 25

Cell Taper Voltage = 100

Current Taper Window = 40

TCA Set % = 99

TCA Clear % = 95

FC Set % = 100

FC Clear % = 98

DODatEOC Delta T = 10.0

[Data(Configuration)]

Rem Cap Alarm = 100

Initial Standby = -10

Initial MaxLoad = -500

Manuf Date = 01-ene-1980

Ser. Num. = 0001

Cycle Count = 7

CC Threshold = 900

Design Capacity = 2200

Design Energy = 18480

SOH Load I = -400

TDD SOH Percent = 90

Cell Charge Voltage T1-T2 = 4200

Cell Charge Voltage T2-T3 = 4200

Cell Charge Voltage T3-T4 = 4100

Charge Current T1-T2 = 10
Charge Current T2-T3 = 50
Charge Current T3-T4 = 30
JEITA T1 = -10
JEITA T2 = 10
JEITA T3 = 45
JEITA T4 = 55
ISD Current = 10
ISD I Filter = 127
Min ISD Time = 7
Design Energy Scale = 1
Device Name = bq34z100
Manufacturer Name = Texas Instr
Device Chemistry = LIPO
[Discharge(Configuration)]
SOC1 Set Threshold = 150
SOC1 Clear Threshold = 175
SOCF Set Threshold = 75
SOCF Clear Threshold = 100
Cell BL Set Volt Threshold = 3000
Cell BL Set Volt Time = 0
Cell BL Clear Volt Threshold = 5
Cell BH Set Volt Threshold = 4300
Cell BH Volt Time = 2
Cell BH Clear Volt Threshold = 5
[Manufacturer Data(Configuration)]
Pack Lot Code = 0000
PCB Lot Code = 0000
Firmware Version = 0000
Hardware Revision = 0000
Cell Revision = 0000
DF Config Version = 0000
[Integrity Data(Configuration)]
Static Chem DF Checksum = 0000
[Lifetime Data(Configuration)]
Lifetime Max Temp = 30.0
Lifetime Min Temp = 20.0
Lifetime Max Chg Current = 0
Lifetime Max Dsg Current = 0
Lifetime Max Pack Voltage = 3200
Lifetime Min Pack Voltage = 3500
[Lifetime Temp Samples(Configuration)]
LT Flash Cnt = 0
[Registers(Configuration)]
Pack Configuration = 0941
Pack Configuration B = FF
Pack Configuration C = 30
LED_Comm Configuration = 00
Alert Configuration = 0002
Number of series cell = 2

[Lifetime Resolution(Configuration)]

LT Temp Res = 1.0

LT Cur Res = 100

LT V Res = 25

LT Update Time = 60

[LED Display(Configuration)]

LED Hold Time = 4

[Power(Configuration)]

Flash Update OK Cell Volt = 3333

Sleep Current = 10

FS Wait = 0

[Manufacturer Info(System Data)]

Block A 0 = 00

Block A 1 = 00

Block A 2 = 00

Block A 3 = 00

Block A 4 = 00

Block A 5 = 00

Block A 6 = 00

Block A 7 = 00

Block A 8 = 00

Block A 9 = 00

Block A 10 = 00

Block A 11 = 00

Block A 12 = 00

Block A 13 = 00

Block A 14 = 00

Block A 15 = 00

Block A 16 = 00

Block A 17 = 00

Block A 18 = 00

Block A 19 = 00

Block A 20 = 00

Block A 21 = 00

Block A 22 = 00

Block A 23 = 00

Block A 24 = 00

Block A 25 = 00

Block A 26 = 00

Block A 27 = 00

Block A 28 = 00

Block A 29 = 00

Block A 30 = 00

Block A 31 = 00

[IT Cfg(Gas Gauging)]

Load Select = 1

Load Mode = 0

Max Res Factor = 15

Min Res Factor = 3

Ra Filter = 500

Fast Qmax Start DOD % = 92

Fast Qmax End DOD % = 96

Fast Qmax Start Volt Delta = 200

Cell Terminate Voltage = 3200

Cell Term V Delta = 50

ResRelax Time = 200

User Rate-mA = 0

User Rate-Pwr = 0

Reserve Cap-mAh = 0

Reserve Energy = 0

Max Scale Back Grid = 4

Cell Max DeltaV = 200

Cell Min DeltaV = 0

Max Sim Rate = 2

Min Sim Rate = 20

Ra Max Delta = 44

Qmax Max Delta % = 5

Cell DeltaV Max Delta = 10

Fast Scale Start SOC = 10

Charge Hys V Shift = 40

[Current Thresholds(Gas Gauging)]

Dsg Current Threshold = 50

Chg Current Threshold = 50

Quit Current = 110

Dsg Relax Time = 60

Chg Relax Time = 60

Quit Relax Time = 1

Cell Max IR Correct = 400

[State(Gas Gauging)]

Qmax Cell 0 = 2200

Cycle Count = 0

Update Status = 00

Cell V at Chg Term = 4200

Avg I Last Run = 0

Avg P Last Run = 0

Cell Delta Voltage = 2

T Rise = 20

T Time Constant = 1000

[OCVa Table(OCV Table)]

Chem ID = 0146

[R_a0(Ra Table)]

Cell0 R_a flag = FF55

Cell0 R_a 0 = 105

Cell0 R_a 1 = 100

Cell0 R_a 2 = 113

Cell0 R_a 3 = 143

Cell0 R_a 4 = 98
Cell0 R_a 5 = 97
Cell0 R_a 6 = 108
Cell0 R_a 7 = 89
Cell0 R_a 8 = 86
Cell0 R_a 9 = 85
Cell0 R_a 10 = 87
Cell0 R_a 11 = 90
Cell0 R_a 12 = 110
Cell0 R_a 13 = 647
Cell0 R_a 14 = 1500
[R_a0x(Ra Table)]
xCell0 R_a flag = FFFF
xCell0 R_a 0 = 105
xCell0 R_a 1 = 100
xCell0 R_a 2 = 113
xCell0 R_a 3 = 143
xCell0 R_a 4 = 98
xCell0 R_a 5 = 97
xCell0 R_a 6 = 108
xCell0 R_a 7 = 89
xCell0 R_a 8 = 86
xCell0 R_a 9 = 85
xCell0 R_a 10 = 87
xCell0 R_a 11 = 90
xCell0 R_a 12 = 110
xCell0 R_a 13 = 647
xCell0 R_a 14 = 1500

[Data(Calibration)]
CC Gain = 10.134
CC Delta = 10.114
CC Offset = -0.65
Board Offset = -0.24
Int Temp Offset = 0.0
Ext Temp Offset = 0.0
Voltage Divider = 19284
[Current(Calibration)]
Deadband = 5

[Codes(Security)]
Sealed to Unsealed = 36720414
Unsealed to Full = FFFFFFFF
Authen Key3 = 01234567
Authen Key2 = 89ABCDEF
Authen Key1 = FEDCBA98
Authen Key0 = 76543210

2. ANEXO II

INDICE

BoardParams.h.....	61
UGV.ino.....	63
BatteryFunctions.ino.....	66
SerialFunctions.ino.....	68

BoardParams.h

```

// Serial port configuration
#define BAUD_RATE 57600

// Motors parameters
#define PIN_PWM_R      5 // Speed control of right motor
#define PIN_PWM_L      6 // Speed control of left motor
#define PIN_MOT_R      4 // Direction control of right motor
#define PIN_MOT_L      7 // Direction control of left motor

#define MIN_PWM        58 // PWM minimum value
#define MAX_PWM        255 // PWM maximum value

// Fuel gauge pins
#define PIN_ALERT      8 // Fuel gauge negated alert pin.

// Debug LED
#define PIN_LED         13

// Comm protocol constants
#define ID_SLAVE      0x01
#define ID_MASTER      0x01
#define STX            0x02
#define ETX            0x03

// Function codes from PC
#define READY         0X04
#define MOVE          0X05
#define GET_SOC       0X06
#define GET_V          0X07
#define GET_R_CAP     0X08
#define GET_TEMP      0X09
#define GET_CURR      0X0A

// Function codes from RoMeo
#define ACK_MSG        0X01
#define SOC_MSG        0x02
#define V_MSG          0X03
#define R_CAP_MSG     0X04
#define TEMP_MSG       0X05
#define CURR_MSG       0X06

// I2C address (B1010101)
#define FUEL_GAUGE_I2C_ADDR    0x55

// I2C commands
#define READ_CONTROL_LOW    0X00

```

```
#define READ_CONTROL_HIGH      0X01
#define READ_STATE_OF_CHARGE_LOW 0x02
#define READ_STATE_OF_CHARGE_HIGH 0x03
#define READ_CAPACITY_LOW 0X04
#define READ_CAPACITY_HIGH 0X05
#define READ_VOLTAGE_LOW         0X08
#define READ_VOLTAGE_HIGH        0X09
#define READ_TEMPERATURE_LOW     0X0c
#define READ_TEMPERATURE_HIGH    0X0d
// I2C Subcommands
#define READ_CURRENT_LOW         0X00
#define READ_CURRENT_HIGH        0X18
```

UGV.ino

/*

UGV program, from UviSpace project for Arduino RoMeo

Sets the board as a slave and enables communications through serial port. Move the motors according to the master's orders

Components:

- * Serial port: Communication based on the protocol SerMesProtocol specified in the docs. The program has been tested using XBEE protocol (IEEE 802.15.4 based).
- * I2C wires: These port allows to communicate with external devices using the I2C protocol. It uses 2 wires, one for the clock synchronization signal and the other one for sending serial data.
- * Outputs: 2 Motors conected to pins 4-5 and 6-7 respectively. The first pin is for power control and the second one is for direction control.

Subroutines and functions:

- * loop: main function of the program. The board will be constantly listening for incoming messages on the serial port. When a message is received, its validity is checked and data is extracted. Finally, the data is passed to the process_message() function.
- * process_message: Function which takes the processed data and analyzes it, deciding which function to call afterwards, and passing it the necessary values. Finally, after the action is completed, it calls publish_data.
- * publish_data: Function which answers the master request through the serial port.
- * move_robot: This function receives 2 bytes of data, 1 for each motor. It resizes the bytes and obtains the direction and module of the motors' speeds.

*/

// I2C Library

#include <Wire.h>

// Robot control program and parameters

#include "BoardParams.h"

```
void setup(void) {
  // Motor pins.
  pinMode(PIN_PWM_R, OUTPUT);
  pinMode(PIN_PWM_L, OUTPUT);
  pinMode(PIN_MOT_R, OUTPUT);
  pinMode(PIN_MOT_L, OUTPUT);
  // Debug LED.
  pinMode(PIN_LED, OUTPUT);
  Serial.begin(BAUD_RATE);
```

```

// Joing I2C bus.
Wire.begin();
}

// Auxiliar subroutines declaration
void move_robot(unsigned char a,unsigned char b);
void process_message(char raw_data[]);
void publish_data(char fun_code, unsigned long int len, char* data);

// Iterations counter
unsigned int it_counter = 0;

// I2C function variables
unsigned int soc[2];
unsigned int voltage[2];
unsigned int remaining_capacity[2];
unsigned int temperature[2];
unsigned int current[2];

// Main loop (communications)
void loop(void)
{
    // Variables definition
    unsigned long int length;
    unsigned char fun_code;
    char id_slave = ID_SLAVE;
    char id_master = ID_MASTER;
    char etx = ETX;
    char stx = STX;
    char buffer[30];
    int j;
    // LED indicates that board is waiting for transmission
    digitalWrite(PIN_LED, HIGH);
    if (Serial.available())
    {
        digitalWrite(PIN_LED, LOW);
        buffer[0] = Serial.read();
        if (buffer[0]==stx)
        {
            for (j=1; j < 6; j++)
            {
                while (! Serial.available()) {}
                buffer[j] = Serial.read();
            }
            id_slave= buffer[1];
            id_master= buffer[2];
            fun_code = buffer[5];
            length = 256*((long int)buffer[4])+((long int)buffer[3]);
            char data[length];
        }
    }
}

```

```

for (j=6; j < length+6; j++)
{
    while (! Serial.available()) {}
    buffer[j] = Serial.read();
    data[j-6] = buffer[j];
}
while (! Serial.available()) {}
buffer[length+6] = Serial.read();
if (buffer[length+6]==ETX)
{
    Serial.flush();
    process_message(data, fun_code, length);
}
}
}

/* In each different iteration cycle, a different parameter is
required in order to space operations over time and reduce the
cycle time*/
if (it_counter == 0){
    it_counter++;
    ReadBatParam(READ_STATE_OF_CHARGE_LOW,
READ_STATE_OF_CHARGE_HIGH,
    false, &soc[0]);
}
else if (it_counter == 1){
    it_counter++;
    ReadBatParam(READ_VOLTAGE_LOW, READ_VOLTAGE_HIGH,
    false, &voltage[0]);
}
else if (it_counter == 2){
    it_counter++;
    ReadBatParam(READ_CAPACITY_LOW, READ_CAPACITY_HIGH,
    false, &remaining_capacity[0]);
}
else if (it_counter == 3){
    it_counter++;
    ReadBatParam(READ_TEMPERATURE_LOW, READ_TEMPERATURE_HIGH,
    false, &temperature[0]);
}
else if (it_counter == 4){
    it_counter = 0;
    ReadBatParam(READ_CURRENT_LOW, READ_CURRENT_HIGH,
    true, &current[0]);
}
}
}

```

BatteryFunctions.ino

/*

This script contains the ReadBatParam function, which has 4 parameters

- I2C_Command_Low: It is the low byte of the I2C command for communicating with the board.

- I2C_Command_High: It is the high byte of the I2C command for communicating with the board.

The two first parameters are variables that take the value of constants defined in the script BoardParams.h.

- subcommand: It is a boolean. It only indicates if the battery parameter required is the current. The current parameter in the PCB is accessed by using subcommands. It is the only parameter that UviSpace uses that has to deal with subcommands.

- *parameter: pointer dereference (*). The function call argument assigned to this one has to be a pointer reference (&). That has to be done in this way because in C language a function can not return an array of data. Instead of doing that, if you work inside the function with a pointer, that information is stored properly and you can get it outside the function.

*/

```
void ReadBatParam(unsigned char I2C_Command_Low, unsigned char I2C_Command_High,
                  boolean subcommand, unsigned int *parameter){
    if (subcommand == true){
        // Write Fuel Gauge Address.
        Wire.beginTransmission(FUEL_GAUGE_I2C_ADDR);
        // Write I2C subcommand.
        Wire.write(READ_CONTROL_LOW);
        Wire.endTransmission();
    }
    // Write Fuel Gauge Address.
    Wire.beginTransmission(FUEL_GAUGE_I2C_ADDR);
    // Ask for less significative byte.
    Wire.write(I2C_Command_Low);
    Wire.endTransmission();
    // Read requested byte.
    Wire.requestFrom(FUEL_GAUGE_I2C_ADDR, 1);
    parameter[1] = (unsigned int) Wire.read();

    if (subcommand == true){
        // Write Fuel Gauge Address.
        Wire.beginTransmission(FUEL_GAUGE_I2C_ADDR);
        // Write I2C subcommand.
        Wire.write(READ_CONTROL_HIGH);
        Wire.endTransmission();
    }
    // Write Fuel Gauge Address
    Wire.beginTransmission(FUEL_GAUGE_I2C_ADDR);
```

```
// Ask for more significative byte.  
Wire.write(I2C_Command_High);  
Wire.endTransmission();  
// Read requested byte.  
Wire.requestFrom(FUEL_GAUGE_I2C_ADDR, 1);  
parameter[0]=(unsigned int) Wire.read();  
}
```

SerialFunctions.ino

```

// Select function and call corresponding subroutine
void process_message(char raw_data[], unsigned char fun_code,
                      unsigned long int length){
    char incomming_data[length];
    char *output_data;
    char sending_function_code;
    int message_length;
    int j;
    int k;
    for (j=0; j<length; j++){
        incomming_data[j] = raw_data[j];
    }
    if (fun_code == READY){
        // Sends back an acknowledge message.
        sending_function_code = ACK_MSG;
        message_length = 0;
    }
    else if (fun_code == MOVE){
        // Writes to the motors the speed values and direction.
        // After that, sends back an acknowledge message.
        move_robot(incomming_data[0],incomming_data[1]);
        sending_function_code = ACK_MSG;
        message_length = 0;
    }
    else if (fun_code == GET_SOC){
        sending_function_code = SOC_MSG;
        message_length = 4;
        output_data = get_bat_param(soc);
    }
    else if (fun_code == GET_V){
        sending_function_code = V_MSG;
        message_length = 4;
        output_data = get_bat_param(voltage);
    }
    else if (fun_code == GET_R_CAP){
        sending_function_code = R_CAP_MSG;
        message_length = 4;
        output_data = get_bat_param(remaining_capacity);
    }
    else if (fun_code == GET_TEMP){
        sending_function_code = TEMP_MSG;
        message_length = 4;
        output_data = get_bat_param(temperature);
    }
    else if (fun_code == GET_CURR){
        sending_function_code = CURR_MSG;
    }
}

```

```

    message_length = 4;
    output_data = get_bat_param(current);
}
publish_data(sending_function_code, message_length, &output_data[0]);
}

/*
Function: move_robot
-----

```

Right and left wheels speed control.

Values from 0 to 127 are interpreted as reverse direction.

Values from 128 to 255 are interpreted as direct direction.

Once the direction is checked, the output has to be rescaled to [0..255], as it is the full range of the output.

```

*/
void move_robot(unsigned char sp_right, unsigned char sp_left)
{
    boolean right_direction;
    boolean left_direction;
    // Reverse direction
    if (sp_right < 128){
        if (sp_right < 0){
            sp_right = 0;
        }
        sp_right *= 2;
        sp_right++;
        sp_right = 255 - sp_right;
        right_direction = true;
    }
    // Direct direction
    else{
        if (sp_right > 255){
            sp_right = 255;
        }
        sp_right -= 128;
        sp_right *= 2;
        sp_right++;
        right_direction = false;
    }
    if (sp_left < 128){
        if (sp_left < 0){
            sp_left = 0;
        }
        sp_left *= 2;
        sp_left++;
        sp_left = 255 - sp_left;
        left_direction = true;
    }
}
```

```

    }
else{
    if (sp_left > 255){
        sp_left = 255;
    }
    sp_left -= 128;
    sp_left *= 2;
    sp_left++;
    left_direction = false;
}
// Send the values to the corresponding pins.
analogWrite (PIN_PWM_R, sp_right);
digitalWrite(PIN_MOT_R, right_direction);
analogWrite (PIN_PWM_L, sp_left);
digitalWrite(PIN_MOT_L, left_direction);
}

// Publish data functions
void publish_data(char fun_code, int len, char *data) {
    char partial_len[2];
    char id_slave = ID_SLAVE;
    char id_master = ID_MASTER;
    char etx = ETX;
    char stx = STX;
    int j;
    // Less significative byte.
    partial_len[1] = (char)len;
    // Most significative byte.
    partial_len[0] = (char)(len >> 8);
    // Sends through serial port the message bytes.
    Serial.print(stx);
    Serial.print(id_master);
    Serial.print(id_slave);
    Serial.print(partial_len[0]);
    Serial.print(partial_len[1]);
    Serial.print(fun_code);
    for (j=0; j<len; j++){
        Serial.print(data[j]);
    }
    Serial.print(etx);
}
}

char *get_bat_param(unsigned int parameter[]){
    char *output_data;
    output_data = (char*)malloc(2*sizeof(unsigned int));
    int j = 0;
    int k = 0;
    // Each parameter element has 2 bytes, thus each one has to be written
    // on 2 chars.
    for (j=0; j<2; j++){

```

```
output_data[k] = 0xFF00 & parameter[j] >> 8;  
output_data[k+1] = 0x00FF & parameter[j];  
k += 2;  
}  
return output_data;  
}
```

3. ANEXO III : Python

INDICE

messenger.py (l.78-l.107).....	73
messenger.py (l.139-l.150).....	74
serialcomm.py (l.35-l.71).....	75
serialcomm.py (l.153-l.165).....	76

messenger.py (l.78-l.107)

```

def listen_speed_set_points(my_serial, robot_id, robot_speed, speed_calc_times,
                           wait_times, xbee_times, soc_read_interval=5):
    """Listens for new speed set point messages on a subscriber socket."""
    logger.debug("Initializing subscriber socket")
    # Open a subscribe socket to listen speed directives
    listener = zmq.Context.instance().socket(zmq.SUB)
    # Set subscribe option to empty so it receives all messages
    listener.setsockopt_string(zmq.SUBSCRIBE, u"")
    # Set the conflate option to true so it only keeps the last message received
    listener.setsockopt(zmq.CONFLATE, True)
    listener.connect("tcp://localhost:{}".format(
        int(os.environ.get("UVISPACE_BASE_PORT_SPEED"))+robot_id))

    logger.debug("Listening for speed set points")
    # Initialize the time for checking if the soc has to be read.
    soc_time = time.time()
    # listen for speed directives until interrupted
    try:
        while True:
            data = listener.recv_json()
            logger.debug("Received new speed set point: {}".format(data))
            move_robot(data, my_serial, wait_times, speed_calc_times,
                       xbee_times, robot_speed)
            # Read the battery state-of-charge after regular seconds intervals.
            if (time.time()-soc_time) > soc_read_interval:
                read_battery_soc(my_serial)
                soc_time = time.time()
    except KeyboardInterrupt:
        pass
    return

```

messenger.py (l.139-l.150)

```
def read_battery_soc(my_serial):  
    """Send a petition to the slave for returning the battery SOC"""  
    raw_soc = my_serial.get_soc()  
    soc = ""  
    if raw_soc is not None:  
        # The soc variable are 4 bytes, but the data is stored on the last 2.  
        #soc = struct.unpack('>H', raw_soc[-2:])[0]  
        soc = struct.unpack('>H', raw_soc[1]+raw_soc[3])[0]  
        logger.info("The current battery soc is {}%".format(soc))  
    else:  
        logger.warn("Unable to get the battery state of charge")  
    return soc
```

serialcomm.py (l.35-l.71)

```
class SerMesProtocol(Serial):
    """This is a child of PySerial class and implements a comm. protocol.
```

This class implements a message-based protocol over the serial port in Master-slave mode: The master (PC) starts communication with the slave(peripheral) sending a message. The slave process the message and returns an answer.

The class uses the serial port to implement this protocol.

```
:param str port: name identifier of the port path in the PC's OS.
:param int baudrate: communications speed between the XBee modules.
:param int stopbits: Number of bits at the end of each message.
:param str parity: Message parity. *None* by default
:param float timeout: Time to wait to achieve the communication.
"""

```

----- CLASS CONSTANTS -----

```
# message fields
STX = '\x02'
ETX = '\x03'
# slave-to-master function codes
ACK_MSG = '\x01'
SOC_MSG = '\x02'
V_MSG = '\x03'
R_CAP_MSG = '\x04'
TEMP_MSG = '\x05'
CURR_MSG = '\x06'
# master-to-slave function codes
READY = '\x04'
MOVE = '\x05'
GET_SOC = '\x06'
GET_V = '\x07'
GET_R_CAP = '\x08'
GET_TEMP = '\x09'
GET_CURR = '\x0A'
```

Serialcomm.py (l.153-l.165)

```
def get_soc(self):  
    """Get the State of Charge (SoC) of the vehicle battery."""  
    soc = None  
    self.send_message(self.GET_SOC)  
    Rx_OK, fun_code, length, data = self.read_message()  
    # If the Rx_OK field was not asserted, raise an error  
    if Rx_OK is False:  
        logger.error('Unsuccessfull communication')  
    else:  
        if fun_code == self.SOC_MSG:  
            soc = data  
    return soc
```

Ángel Soto Boullosa
77460475-V

Vigo, a 10 de julio de 2017