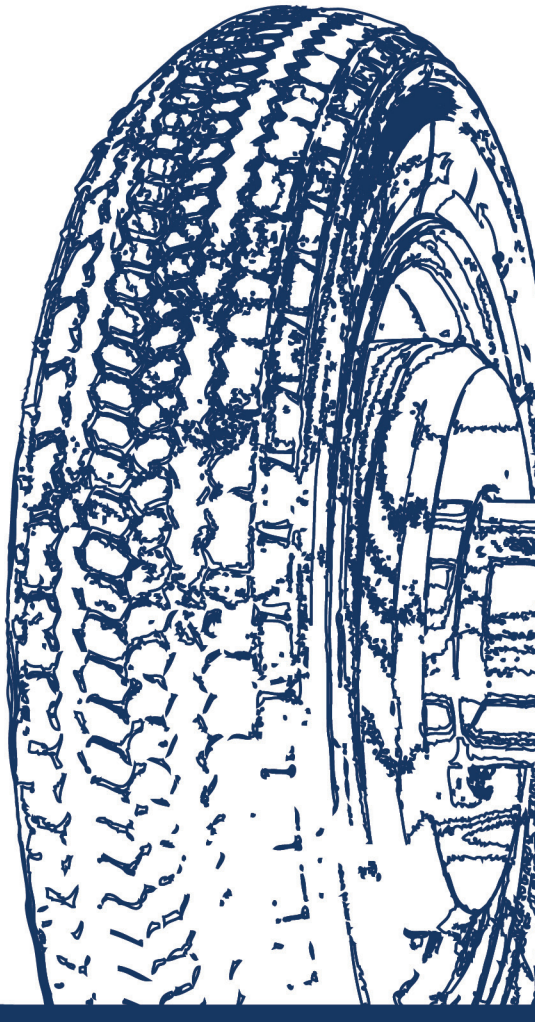
UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada
Sistemas de información y Comunicaciones

IUMA

TRABAJO FIN DE MÁSTER

SISTEMA DE CONTROL DE MOTORES BLDC BASADO EN COMUNICACIONES CAN PARA VEHÍCULOS ELÉCTRICOS



Titulación: Máster en Tecnologías de Telecomunicación
Autor: Himar A. Fabelo Gómez
Tutor: Dr. Aurelio Vega Martínez
Fecha: Julio de 2014

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada
Sistemas de información y Comunicaciones

Máster en
Tecnologías de Telecomunicación

Trabajo Fin de Máster

SISTEMA DE CONTROL DE MOTORES BLDC BASADO EN COMUNICACIONES CAN PARA VEHÍCULOS ELÉCTRICOS

Autor: Himar A. Fabelo Gómez
Tutor: Dr. Aurelio Vega Martínez

Fecha: Julio de 2014

IUMA
t +34 928 451 086
f +34 928 451 083
iuma@iuma.ulpgc.es
www.iuma.ulpgc.es
Campus Universitario de Tafira
35017 Las Palmas de Gran Canaria

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada
Sistemas de información y Comunicaciones

Máster en
Tecnologías de Telecomunicación

Trabajo Fin de Máster

SISTEMA DE CONTROL DE MOTORES BLDC BASADO EN COMUNICACIONES CAN PARA VEHÍCULOS ELÉCTRICOS

HOJA DE FIRMAS

Alumno:	Himar A. Fabelo Gómez	Fdo.:
Tutor:	Dr. Aurelio Vega Martínez	Fdo.:
Fecha:	**Julio de 2014**	

IUMA
t +34 928 451 086
f +34 928 451 083
iuma@iuma.ulpgc.es
www.iuma.ulpgc.es
Campus Universitario de Tafira
35017 Las Palmas de Gran Canaria

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada
Sistemas de información y Comunicaciones

Máster en
Tecnologías de Telecomunicación

Trabajo Fin de Máster

SISTEMA DE CONTROL DE MOTORES BLDC BASADO EN COMUNICACIONES CAN PARA VEHÍCULOS ELÉCTRICOS

HOJA DE EVALUACIÓN

Calificación: .....

Presidente	Dr. Javier García García	Fdo.:
Secretario	Dr. Gustavo Marrero Callicó	Fdo.:
Vocal	Dr. Fernando de la Puente Arrate	Fdo.:

Fecha: **Julio de 2014**

IUMA
t +34 928 451 086
f +34 928 451 083
iuma@iuma.ulpgc.es
www.iuma.ulpgc.es
Campus Universitario de Tafira
35017 Las Palmas de Gran Canaria

Versión: 2.0 – 14 junio 2011

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. Introducción

1.1. Resumen

Desde hace algunos años, la industria del automóvil ha comenzado a centrar sus esfuerzos en la investigación y desarrollo de vehículos eléctricos. Este tipo de vehículos posee algunas ventajas frente a los vehículos tradicionales de combustión interna, tales como el ahorro económico en el consumo de combustible y la nula emisión de gases contaminantes. Es por ello por lo que estos vehículos, a pesar de presentar, de momento, una autonomía menor que la de los vehículos de combustión interna, pueden ser en un futuro cercano muy habituales en nuestras carreteras.

Puesto que los vehículos eléctricos requieren de una alta eficiencia para conseguir las mejores prestaciones, es requisito indispensable el uso de motores BLDC (*Brushless Direct Current Motors*). Este tipo de motores ofrecen excelentes características de par, unas altas prestaciones y un rango de velocidades muy amplio, además de una gran vida útil. Al no poseer escobillas de conmutación, se reduce la necesidad de un mantenimiento periódico. Sin embargo, se hace necesaria la utilización de un complejo sistema de control basado en un microcontrolador que gestione la conmutación de las bobinas del motor.

En este proyecto se ha desarrollado una controladora para motores BLDC de 3 fases basada en un microcontrolador ATmega64M1. Gracias a esta controladora y un sistema de gestión de baterías (BMS) de Ión-Litio desarrollado en otro proyecto, se ha implementado un sistema de control de motores BLDC aplicado a vehículos eléctricos. Para permitir la intercomunicación del sistema mediante el protocolo CAN se ha utilizado, en ambas placas, un microcontrolador que posee un controlador CAN. Además, se ha incorporado al diseño (tanto de la controladora como de la BMS) el tansceptor CAN que permite la conexión directa a este tipo de bus.

Por otra parte, para poder comprobar el correcto funcionamiento de la controladora BLDCM fabricada, se ha desarrollado el software de control del motor BLDC para el microcontrolador ATmega64M1. La librería C se ha elaborado siguiendo una metodología de programación modular. Uno de los beneficios de esta metodología de desarrollo es obtener la posibilidad de reutilizar el código elaborado, consiguiéndose una gran productividad y reducción de tiempo en futuros trabajos en los que se puedan utilizar los módulos creados anteriormente. El desarrollo de la librería de funciones se ha basado en la técnica de control de lazo abierto con ajuste de velocidad.

Por último, se presenta un software de control y monitorización del motor BLDC desde un PC. Este software ha sido desarrollado en LabVIEW y está basado en el protocolo de comunicaciones RS-232.

1.2. Abstract

In recent years, the automotive industry has begun to focus its efforts on the research and development of electric vehicles. This type of vehicle has some advantages over traditional internal combustion vehicles, such as the cost savings in fuel consumption and zero emissions of polluting gases. For this reason the use of these vehicles, despite having less autonomy than internal combustion vehicles, may increase in the near future.

Due to electric vehicles require high efficiency to achieve the best performance, it is required to use BLDC motors (Brushless Direct Current Motors). These motors have excellent torque characteristics, high performance, and a very wide range of speeds, plus a lifetime. They have no brushes switching, so the need for periodic maintenance is reduced. However, the use of a complex control system based on a microcontroller that manages the switching of the motor coils is necessary.

This project has developed a controller for 3-phase BLDC motors based on a microcontroller ATmega64M1. With this BLDCM controller and battery management system (BMS) for Li-ion batteries developed in another project, a control system of BLDC motors applied to electric vehicles has been implemented. To allow the communication of the system using the CAN protocol in a future project, a microcontroller that has a CAN controller has been used in both boards. Has also been incorporated into the design (both the controller and the BMS) a CAN transceiver that allows direct connection to this type of bus.

Moreover, in order to check the correct operation of the BLDCM controller manufacturer, software for BLDC motor control based on a microcontroller ATmega64M1 has been developed. The C library has been made using a modular programming methodology. One advantage of this development methodology is to obtain the ability to reuse code developed achieving high productivity and reduced time in future work. The development of the function library for controlling BLDC motors is based on the open-loop control with speed adjustment technique.

Finally, a control and monitoring software of the BLDC motor from a PC is presented. This software has been developed in LabVIEW and is based on the RS-232 communication protocol.

1.3. Objetivos del Proyecto

El objetivo de este proyecto es implementar un sistema de control de motores BLDC para vehículos eléctricos. Para ello se han desarrollado las siguientes tareas:

- Diseño y fabricación de una controladora de motores BLDC de tres fases basada en un microcontrolador ATmega64M1 con formato estándar Simple Europa y adaptada para permitir la utilización del protocolo de comunicaciones CAN.
- Desarrollo de la arquitectura de una librería de funciones en C basada en la técnica de lazo abierto con ajuste de velocidad que permita la comprobación del correcto funcionamiento de la controladora.
- Desarrollo de un software de control y monitorización remota del motor BLDC mediante comunicación RS-232 en LabVIEW.
- Proponer una implementación con el protocolo CAN mediante la cual se permita controlar y monitorizar el sistema de control de motores BLDC propuesto.
- Fabricación de un prototipo para las pruebas del sistema de control completo.

2. Estado del Arte

2.1. Vehículos Eléctricos

2.1.1. Historia de los Vehículos Eléctricos

El uso del vehículo eléctrico beneficia el medio ambiente al no emitir monóxido de carbono y otros gases nocivos para la salud humana. Las limitaciones energéticas y ambientales existentes a nivel mundial indican que se deben encontrar soluciones viables que sustituyan el empleo de combustibles fósiles. El transporte eléctrico es una de esas soluciones a la contaminación global.

La construcción del primer vehículo eléctrico se le atribuye a Robert Anderson en 1839 en Aberdeen, Escocia, pero oficialmente se reconoce a Gustave Trouvé como el primero en construir un triciclo eléctrico, exhibido en París, Francia en 1881. Un año después, en 1882, W. Ayrton y J. Perry presentaron en Inglaterra otro triciclo (Figura 2-1). El desarrollo de este vehículo eléctrico tenía la forma de un carruaje sin caballos. Este estilo fue imitado en muchos países de forma simultánea, particularmente en Francia, en 1902, por Jeantaud y Krieger.

Figura 2-1: Vehículo eléctrico creado por W. Ayrton y J. Perry en 1882

En la primera década del siglo XX existía una fuerte competencia por dominar el mercado automovilístico entre las compañías de vehículos eléctricos y las compañías de vehículos de combustión interna. El bajo costo del petróleo en aquel entonces y el largo periodo de carga de las baterías eléctricas, propició que el consumidor se inclinara por los vehículos de combustión interna. La escasa demanda de vehículos eléctricos provocó un decremento considerable en el desarrollo tecnológico de éstos.

2.1.2. Vehículos Eléctricos en la Actualidad

Con los coches eléctricos, aparte de la contaminación medioambiental, también se reduce la contaminación acústica, ya que son vehículos silenciosos. Además, si se hace referencia a la inferior autonomía de los vehículos eléctricos actuales frente a los de combustión interna, es necesario conocer el dato de que en Estados Unidos, el 80% de los desplazamientos diarios son inferiores a 80km., y más de la mitad son inferiores a 40km. En la Unión Europea, en 2007, según *Eurostat*, 460 millones de ciudadanos realizan en promedio tres desplazamientos, que totalizan 27km diarios en coche. Por lo tanto, podemos afirmar con seguridad que existe un mercado potencial para los vehículos eléctricos, y que además cubriría las necesidades de una gran parte de los requerimientos personales de movilidad.

Existen muchas empresas que se han aventurado a fabricar vehículos eléctricos. A continuación se presentan algunos ejemplos de ello.

2.1.2.1. Tesla Motors

La empresa Tesla Motors [1], situada en Sillicon Valley, California, es una de las pioneras en la construcción y comercialización de coches eléctricos. Actualmente, tiene dos vehículos eléctricos en el mercado, el Tesla Model S y el Tesla Model X.

2.1.2.1.1. Tesla Model S

El Tesla Model S es un sedán eléctrico fabricado por Tesla Motors, que inició sus entregas en el mercado estadounidense el 22 de junio de 2012.2 La autonomía del Model S equipado con una batería de 85 kWh es de 426 km, convirtiéndose en el automóvil eléctrico con la mayor autonomía disponible en el mercado en ese momento. En diciembre de 2013 el Model S alcanzó el hito de 25.000 unidades vendidas en Estados Unidos, Canadá y Europa.

El modelo base, con una batería de 60 kWh tiene una autonomía de 370 km y una aceleración de 0 a 100 km/h de 6.2 segundos mientras que el modelo con la batería de 85 kWh Performance (PM85) acelera de 0 a 100 km/h en 4.4 segundos con una autonomía de 480 km.

TABLA 2-1. ESPECIFICACIONES TESLA MODEL S

Especificaciones	Tesla Model S
Carburante	Energía eléctrica
Potencia máxima kW (C.V.)	225 (302) a 310 (416)
Par máximo Nm	430 a 600
Capacidad de Batería (kWh)	60 a 85
Velocidad máxima (km/h)	190 a 210
Autonomía (km)	334 a 426
Precio (€)	Desde 46.600,00

Figura 2-2: Tesla Model S

2.1.2.1.2. Tesla Model X

El Tesla Model X es un monovolumen eléctrico fabricado por Tesla Motors basado en la plataforma del Model S. Iniciará sus entregas en el mercado estadounidense a principios de 2015.

Figura 2-3. Tesla Model X

2.1.2.2. Mahindra Reva

Reva es el coche eléctrico urbano más vendido del mundo, y se autodefine como eléctrico, económico y ecológico [2]. Es fabricado por la empresa india *Reva Electric Car Co.* Se trata de un vehículo pequeño de tres puertas, diseñado para uso urbano, y con capacidad para tres personas. Actualmente, se puede elegir entre el modelo T01 Base y el T2 Premium que ofrecen una autonomía máxima de 80km y 100km, respectivamente.

TABLA 2-2. ESPECIFICACIONES MAHINDRA REVA

Especificaciones	MaindraReva T01 Base	MaindraReva T2 Premium
Carburante	Energía eléctrica	
Potencia máxima kW	14,8	19
Par máximo Nm	53,9	53
Velocidad máxima (km/h)	75	81
Autonomía (km)	80	100
Precio (€)	5.900,00	6.850,00

Figura 2-4: Modelos de los vehículos eléctricos Mahindra Reva

2.1.2.3. Renault

Actualmente la marca Renault dispone de 3 modelos de vehículos eléctricos en el mercado [3].

2.1.2.3.1. Renault Twizy

El Renault Twizy es un vehículo eléctrico biplaza del producido por Renault en su planta de Valladolid desde 2011. Es uno de los cuatro vehículos Renault ZE que la marca está fabricando para adentrarse en este segmento de mercado. En concreto, el Twizy es un vehículo de dos plazas en tándem sin puertas, aunque cuenta en opción con protectores laterales, que son una especie de puertas sin ventanilla.

TABLA 2-3. ESPECIFICACIONES RENAULT TWIZY

Especificaciones	Twizy Urban 45	Twizy Urban
Carburante	Energía eléctrica	
Potencia máxima KW CEE (c.v.)	004 (005)	008 (013)
Par máximo Nm CEE (m.kg)	33	57
Consumo Z.E. (Wh/km)	58	63
Velocidad máxima (km/h)	45	80
Precio (€)	7.220,00	7.930,00

Figura 2-5. Renault Twizy

2.1.2.3.2. Renault Zoe

El *Renault Zoe* fue presentado en el salón de París de 2010 y salió a la venta en el año 2012. El Renault Zoe está equipado con un motor eléctrico de 88 CV, con 222 Nm de par motor desde las 250 r.p.m.

TABLA 2-4. ESPECIFICACIONES RENAULT ZOE

Especificaciones	Zoe Life	Zoe Intens	Zoe Zen
Carburante	Energía eléctrica		
Potencia máxima KW CEE (c.v.)	065 (088)		
Par máximo Nm CEE (m.kg)	222		
Consumo Z.E. (Wh/km)	146		
Velocidad máxima (km/h)	135		
Autonomía Z.E. (km)	210		
Precio (€)	21.250,00	23.050,00	

Figura 2-6. Renault Zoe

2.1.2.3.3. Renault Fluence Z. E.

El *Renault Fluence* es un turismo diseñado y producido por el fabricante francés Renault. Fue presentado como concepto en junio del 2004 y finalmente la versión de producción se presentó durante el Salón de Frankfurt del 2009. Durante el Salón del Automóvil edición 2012 de Estambul fue presentado una renovación del modelo que se lanzó en 2013.

TABLA 2-5. ESPECIFICACIONES FLUENCE Z. E.

Especificaciones	Zeo Expression	Zeo Dynamique
Carburante	Energía eléctrica	
Potencia máxima KW CEE	70	
Par máximo Nm CEE (m.kg)	226	
Consumo Z.E. (Wh/km)	140	
Velocidad máxima (km/h)	135	
Autonomía Z.E. (km)	185	
Precio (€)	26.600,00	27.700,00

Figura 2-7. Renault Fluence Z. E.

2.1.2.4. Nissan

El fabricante de vehículos Nissan lanzó al mercado en 2010 el Nissan LEAF [4]. Este modelo se diseñó desde cero para ser un vehículo eléctrico. No es una adaptación de otro vehículo existente. Es el automóvil eléctrico apto para carretera más vendido en la historia, con más de 100.000 unidades vendidas desde diciembre de 2010 hasta febrero de 2014, lo que representaba el 45% de las ventas de coches eléctricos. En 2013 Nissan sacó al mercado una nueva versión del vehículo en el que se realizaron unas 100 mejoras técnicas respecto a la versión del 2010.

TABLA 2-6. ESPECIFICACIONES NISSAN LEAF

Especificaciones	Nissan LEAF
Carburante	Energía eléctrica
Potencia máxima KW CEE (c.v.)	80
Par máximo Nm CEE (m.kg)	254
Capacidad de Batería (kWh)	24 (360 V / 192 Celdas)
Velocidad máxima (km/h)	145
Autonomía (km)	135
Precio (€)	Desde 17.500,00

Figura 2-8. Nissan LEAF

2.1.2.5. Smart

El fabricante de vehículos urbanos Smart, desarrolló en 2007 una versión eléctrica de su modelo Smart ForTwo denominada Smart ForTwo Electric Drive [5]. Se trata de un modelo biplaza de tres puertas y de tan solo 2,69 m de largo. Las especificaciones más relevantes de este modelo de vehículo eléctrico se muestran en la siguiente tabla.

TABLA 2-7. ESPECIFICACIONES SMART FORTWO ELECTRIC DRIVE

Especificaciones	Smart ForTwo Electric Drive
Carburante	Energía eléctrica
Potencia máxima KW (c.v.)	55 (75)
Par máximo Nm	130
Capacidad de Batería (kWh)	17,6
Velocidad máxima (km/h)	125
Autonomía (km)	145
Precio (€)	Desde 11.932,00

Figura 2-9. Smart For Two Electric Drive

2.1.2.6. Volkswagen

2.1.2.6.1. Volkswagen e-up!

El *Volkswagen e-up!* es el primer automóvil en serie totalmente eléctrico fabricado por Volkswagen [6]. Es un vehículo urbano que incorpora un motor eléctrico de 82 caballos de potencia y se alimenta de una batería de iones de litio de 18,7 kw/h de capacidad.

Tabla 2-8. Especificaciones Volkswagen e-up!

Especificaciones	Volkswagen e-up!
Carburante	Energía eléctrica
Potencia máxima C.V.	82
Velocidad máxima (km/h)	130
Autonomía (km)	160
Precio (€)	26.300,00

Figura 2-10. Volkswagen e-up!

2.1.2.6.2. Volkswagen e-Golf

En el año 2015 este fabricante prevé lanzar al mercado el *Volkswagen e-Golf.* Este modelo tiene un motor eléctrico síncrono de imanes permanentes que desarrolla una potencia de 85 kW (casi 116 CV) y 270 Nm de par motor. Gira hasta 12.000 rpm y tiene una eficiencia ligeramente superior al 95%.

Tabla 2-9. Especificaciones Volkswagen e-Golf

Especificaciones	Volkswagen e-Golf!
Carburante	Energía eléctrica
Potencia máxima C.V.	75 a 116
Par máximo Nm	175 a 270
Capacidad de Batería (kWh)	
Velocidad máxima (km/h)	90 a 140
Autonomía (km)	130 a 190
Precio (€)	--

Figura 2-11. Volkswagen e-Golf

2.1.2.7. BMW

2.1.2.7.1. BMW i3

El fabricante alemán BMW [7] lanzó al mercado en 2013 el BMW i3, el primer BMW de serie con cero emisiones debido a su motor eléctrico. Es un coche de 5 puertas para 4 pasajeros con carrocería de fibra de carbono para mejorar el consumo de energía.

TABLA 2-10. ESPECIFICACIONES BMW I3

Especificaciones	BMW i3
Carburante	Energía eléctrica
Potencia máxima C.V.	170
Par máximo Nm	250
Velocidad máxima (km/h)	150
Autonomía (km)	130 a 160
Precio (€)	Desde 35.500,00

Figura 2-12. BMW i3

2.1.2.7.2. BMW i8

Además del modelo i3, en este mismo año 2014 el fabricante alemán comenzó a producir en serie el modelo híbrido BMW i8. Este modelo tiene dos motores eléctricos, uno en el eje delantero con una potencia de 60 kW (max. 104 kW), y el otro en una caja de cambios de doble embrague de 6 marchas con 25 kW (max. 38 kW). Además, dispone de un motor de gasolina BMW TwinPower Turbo de 3 cilindros y de 1,5 litros, con una potencia de 231 CV. La potencia total asciende a 262 kW (356 CV), que permiten acelerar de 0 a 100 km/h en 4,8 s. El alcance del vehículo asciende a 650 km a partir del consumo del depósito. Con la activación de los motores eléctricos se aumenta la autonomía en 50 km hasta un total de 700 km.

TABLA 2-11. ESPECIFICACIONES BMW I8

Especificaciones	BMW i8
Carburante	Energía eléctrica y Gasolina
Potencia máxima kW (C.V.)	262 (356) total
Par máximo Nm	320
Capacidad de Batería (kWh)	5
Velocidad máxima (km/h)	250
Autonomía (km)	700
Precio (€)	129.900,00

Figura 2-13. BMW i8

2.2. El Motor BLDC

Los motores BLDC son uno de los tipos de motores que más popularidad ha ganado en los últimos años. Estos se emplean en sectores industriales tales como: automovilístico, aeroespacial, consumo, médico, equipos de automatización e instrumentación. Gracias a sus numerosas ventajas frente a otro tipo de motores.

La principal característica de los motores BLDC es que no emplean escobillas en la conmutación para la transferencia de energía. Puesto que éstas producen rozamiento, disminuyen el rendimiento, generan calor, son ruidosas y requieren una sustitución periódica y, por tanto, un mayor mantenimiento.

Algunas de las ventajas de los motores BLDC con respecto a los motores DC convencionales son:

- Mejor relación velocidad - par motor.
- Mayor respuesta dinámica.
- Mayor eficiencia.
- Mayor vida útil.
- Menor ruido.
- Mayor rango de velocidad.

También, la relación par-motor/tamaño es mucho mayor, por lo que son muy útiles en ambientes de trabajo con espacio reducido.

2.2.1. Tipos de motores BLDC

Los motores BLDC se fabrican de dos tipos, *inrunner* y *outrunner*. Los primeros desarrollan mayor velocidad y suelen ser más pequeños, entregan su torque máximo a muy altas revoluciones por minuto, por lo que se usan siempre con engranajes reductores. En estos motores el elemento móvil es el eje, sobre el cual se encuentran instalados los imanes permanentes. Por otra parte, los motores *outrunner* consiguen su torque máximo a velocidades más bajas, por lo que usualmente no necesitan reducción, y se pueden acoplar directamente a una hélice. En éstos los imanes permanentes están instalados en la carcasa externa del motor, que en este caso es la que gira y el bobinado se encuentra acoplado al eje.

Estos motores trabajan por medio de controladoras de velocidad, que transforman la corriente continua de las baterías en una tensión alterna trifásica y la alimentan a los bobinados en cierta secuencia dependiendo de la posición del rotor.

Para controlar los motores se precisa el conocimiento de la posición del rotor en cada momento. Para ello, existen dos técnicas dependiendo de la existencia o no de sensores en el motor, lo que los divide en dos familias: con sensores (*sensored*) y sin sensores (*sensorless*)

Sensored: Disponen de sensores de efecto hall o de *encoders* que indican la posición del rotor. Es habitual que tengan 3 sensores separados 120^{o}, uno para cada bobinado del motor.

Sensorless: No tienen sensores. La posición se determina mediante la medición del efecto de la fuerza contraelectromotriz sobre las bobinas.

2.2.2. Métodos de conmutación de motores sin escobillas

Las técnicas de control de los motores con sensores hall se clasifican según el algoritmo de control utilizado [8]. Los más usados son los siguientes, en orden creciente de eficiencia y complejidad:

- Conmutación trapezoidal o "*six step mode*".
- Conmutación sinusoidal.
- Control vectorial *o Field Oriented Control*.

2.2.2.1. Conmutación trapezoidal o six steps mode

Es el método más simple de control de los motores sin escobillas. En este esquema se controla la corriente que circula por los terminales del motor, excitando un par simultáneamente y manteniendo el tercer terminal desconectado. Sucesivamente se va alternando el par de terminales a excitar hasta completar las seis combinaciones posibles. Las 6 direcciones de las corrientes se muestran en la Figura 2-14.

Este esquema puede ser empleado tanto en motores con sensores de efecto Hall como en motores sin sensores, donde se usa para conocer la posición del rotor el censado de la fuerza contraelectromotriz en la bobina que está sin excitar.

Tiene como ventajas su sencillez y facilidad de implementación por lo cual es el método más usado en motores pequeños. Pese a esto, tiene un problema inherente a la conmutación del vector de corrientes que es un rizado en el par de salida. En aplicaciones donde se requieren fuerzas uniformes o bajas velocidades, esto puede llegar a ser inconveniente. En la Figura 2-15 se muestran las corrientes por cada una de las fases, la secuencia de conmutación y el par.

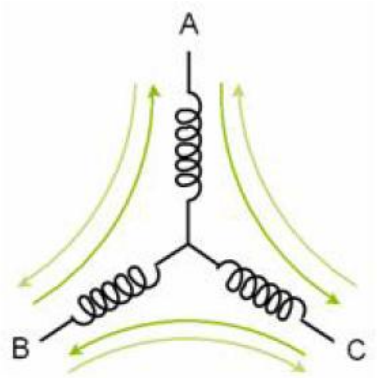


Figura 2-14: Diagrama de circulación de corriente en el control trapezoidal

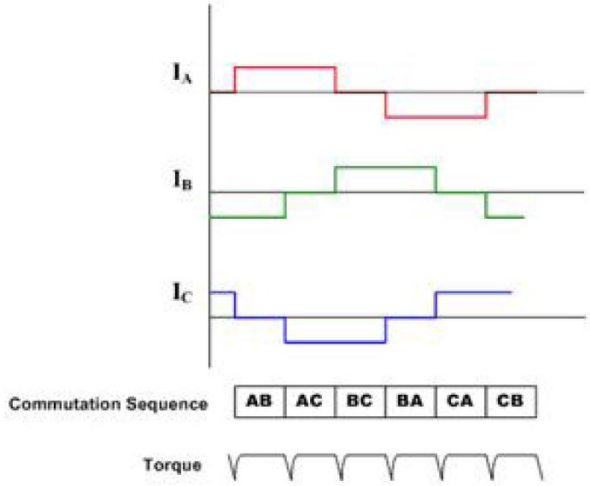


Figura 2-15: Corrientes en las bobinas y par del motor

2.2.2.2. Conmutación sinusoidal

La conmutación sinusoidal es un control más avanzado y exacto que el trapezoidal, ya que intenta controlar la posición del rotor continuamente.

Esta continuidad se consigue aplicando simultáneamente tres corrientes sinusoidales desfasadas 120^{o} a los tres bobinados del motor. La fase de estas corrientes se escoge de forma que el vector de corrientes resultante siempre esté en cuadratura con la orientación del rotor y tenga un valor constante. Como consecuencia de este procedimiento se obtiene un par más preciso y sin el rizado típico de la conmutación trapezoidal. Sin embargo, para poder generar dicha modulación sinusoidal es necesaria una medida precisa de la posición del rotor, que

difícilmente se logra con sensores de efecto Hall, por lo cual se requiere de un *encoder* absoluto de alta resolución.

A bajas velocidades este método de control presenta una gran eficiencia y suavidad del torque, sin embargo, a altas frecuencias no responde tan bien debido a la necesidad de procesar señales sinusoidales de frecuencias altas y a que los controladores PI empleados para generar estas señales tienen una respuesta limitada en ganancia y frecuencia. Cuando la frecuencia es suficientemente alta, la eficiencia decrece y el error aumenta, tendiendo a un punto de cero torque.

2.2.2.3. Control vectorial o Field Oriented Control

El problema principal que presenta la conmutación sinusoidal es que intenta controlar directamente las corrientes que circulan por el motor, las cuales son intrínsecamente variantes en el tiempo. Al aumentar la velocidad del motor, y por tanto la frecuencia de las corrientes, empiezan a aparecer problemas.

El control vectorial o *Field Oriented Control* (FOC) soluciona el problema controlando el vector de corrientes directamente en un espacio de referencia ortogonal y rotacional, llamado espacio D-Q (*Direct-Quadrature*). Dicho espacio de referencia está normalmente alineado con en el rotor de forma que permite que el control del flujo y del par del motor se realice de forma independiente. La componente directa permite controlar el flujo y la componente en cuadratura el par. Para este fin se requiere no solamente una muy buena medición de la orientación del rotor, sino un tratamiento matemático previo de las señales para transformarlas del marco trifásico estático de los bobinados en el estator al marco rotacional D-Q del rotor. Este es el control que presenta mejor respuesta en todos los rangos de velocidad pero resulta ser el más costoso de implementar, lo cual lo hace inadecuado para toda aplicación en la que no sea estrictamente necesario.

2.2.3. Los Sensores de Efecto Hall

Los sensores de posición de efecto Hall tienen numerosas aplicaciones y una gran ventaja, que es la invariabilidad frente a suciedad (no magnética) y frente al agua. En la industria del automóvil el sensor Hall se utiliza de forma frecuente. Algunas de sus aplicaciones son:

- Sensores de posición del cigüeñal (CKP).
- Sistema de cierre del cinturón de seguridad.
- sistemas de cierres de puertas.
- Reconocimiento de posición del pedal o del asiento.
- Para el cambio de transmisión.
- Reconocimiento del momento de arranque del motor (inmovilizador).

Una de sus aplicaciones más destacadas últimamente es su uso en los motores de corriente continua sin escobillas o motores BLDC.

Como ya se explicó antes, a diferencia de un motor de escobillas de corriente continua, la conmutación de un motor BLDC se controla electrónicamente. Para girar el motor BLDC, los

bobinados del estator deben estar activados en una secuencia concreta. Es importante conocer la posición del rotor para poder entender cómo deben ser activadas las bobinas según la secuencia de activación. La posición del rotor se detecta mediante sensores de efecto Hall integrados en el estator.

La mayoría de estos motores tienen tres de estos sensores integrados en el estator en el extremo opuesto al rotor del motor. Cada vez que los polos magnéticos del rotor pasan cerca de los sensores, éstos generan una señal de nivel alto o nivel bajo, según si se trata de el polo Norte o Sur el que está pasando cerca de los sensores. A través de la programación de un microcontrolador y basándose en la combinación de estas tres señales, se determina la secuencia exacta de conmutación de las bobinas del motor.

La Figura 2-16 muestra una sección transversal de un motor BLDC con un rotor que tiene imanes alternativos Norte y Sur permanentes. La incorporación de los sensores Hall en el estator es un proceso complejo, ya que cualquier desajuste en éstos, con respecto a los imanes del rotor, generaría un error en la determinación de la posición del mismo.

Para simplificar el proceso de montaje de los sensores Hall en el estator, algunos motores pueden tener sensores magnéticos Hall en el rotor, además del rotor magnético principal. Estos son una versión reducida de la réplica del rotor. Por lo tanto, cada vez que el rotor gira, los sensores magnéticos Hall proporcionan el mismo efecto que los imanes principales. Los sensores Hall se montan normalmente en una placa de circuito impreso y se fija a la tapa de caja en el extremo de no conducción. Esto permite a los usuarios ajustar el conjunto completo de sensores, para alinearse con los imanes del rotor, con el fin de lograr el mejor rendimiento.

Sobre la base de la posición física de los sensores Hall, hay dos versiones de la producción. Los sensores de Hall pueden estar en 60º o 120º de variación de una fase a la otra. Sobre esta base, el fabricante del motor define la secuencia de conmutación, que se deben seguir cuando se controla el motor.

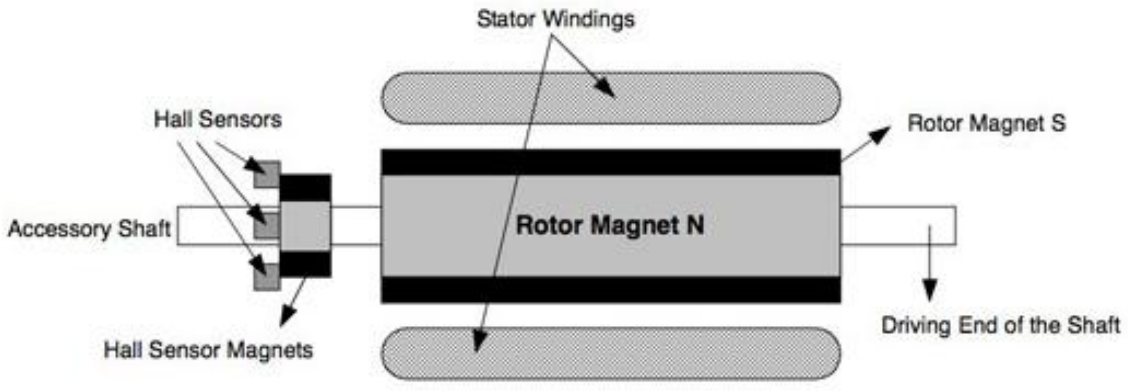


Figura 2-16: Distribución de los sensores de efecto Hall de un motor BLDC

Los sensores de posición Hall en los motores BLDC requieren una fuente de alimentación que puede variar de 4 a 24 voltios y la corriente necesaria puede variar desde 5 hasta 15 miliamperios. Durante el diseño del controlador del motor BLDC, es necesario consultar las

especificaciones técnicas propias del motor, para poder trabajar con rangos de tensiones y corrientes exactas requeridas por los sensores de efecto Hall utilizados.

2.3. El Bus CAN

2.3.1. Introducción

El protocolo de comunicación CAN (*Controller Area Network*) fue propuesto y desarrollado por Robert Bosh, fabricante alemán de componentes para automóviles. Se basa en una topología bus para la transmisión de mensajes en entornos distribuidos y ofrece una solución a la gestión de la comunicación entre múltiples CPUs (unidades centrales de proceso) [9][10].

Las especificaciones originales de Bosch no predeterminaban una capa física concreta. Por ello, posteriormente se normalizó el estándar mediante la norma ISO 11898, para aplicaciones de alta velocidad (hasta 1 Mbps), que define la capa física a través de la transmisión diferencial sobre par trenzado. Es a esta combinación de especificaciones y norma lo que se denomina "Bus CAN".

Existen multitud de dispositivos controladores CAN (ya sean autónomos o integrados como periféricos) y transceptores para dar soporte a la norma ISO 11898. El primero de ellos fue el 82526, desarrollado por Intel, que posteriormente sería sustituido por el 82527.

Este estándar despertó el interés de los fabricantes debido a sus prestaciones de robustez, filosofía multimaestro, técnicas de arbitraje en acceso múltiple no destructivas, comunicación por eventos, etc. Y en 1992 varias compañías se unieron para crear la organización sin ánimo de lucro CiA (*CAN in Automation*) con el fin de ofrecer información técnica de productos y marketing acerca del Bus CAN, consiguiendo con ello garantizar el futuro del protocolo. Actualmente alrededor de 560 empresas son miembros de esta organización internacional. La CiA se encarga de la normalización de especificaciones tanto de la capa física del protocolo como de las capas de aplicación.

Los principales fabricantes mundiales de componentes ya integran controladores CAN como periféricos de sus principales familias de microcontroladores, e incluso existen multitud de modelos VHDL para su incorporación en módulos ASIC y FPGA.

2.3.2. Características del Bus CAN

El Bus CAN se basa en el modelo productor y consumidor, el cual es un concepto que describe una relación entre un nodo productor y uno o más nodos consumidores. Es un protocolo orientado a mensajes, es decir, la información que se va a intercambiar se descompone en mensajes, a los cuales se les asigna un identificador y se encapsulan en tramas para su transmisión. Cada mensaje tiene un identificador único dentro de la red, con el cual los nodos deciden aceptar o no dicho mensaje.

Sus principales características, que se detallarán más adelante, son las siguientes:

- La comunicación está basada en mensajes y no en direcciones.

- Un mensaje es diferenciado por el campo llamado identificador, que no indica el destino del mensaje, pero sí describe el contenido del mismo.
- No hay un sistema de direccionamiento de los nodos en el sentido convencional. Los mensajes se envían según su prioridad.
- La prioridad de los mensajes para acceder al bus la define el identificador.
- Es un sistema multimaestro. Cuando el bus está libre, cualquier nodo puede empezar la transmisión de un mensaje y el mensaje con mayor prioridad es el que accede al bus.
- Todos los nodos CAN son capaces de transmitir y recibir datos y varios pueden acceder al bus de datos simultáneamente.
- Un nodo emisor envía el mensaje a todos los nodos de la red, cada nodo, según el identificador del mensaje, lo filtra y decide si debe procesarlo inmediatamente o descartarlo. Como consecuencia el sistema se convierte en *multicast* en el cual un mensaje puede estar dirigido a varios nodos al mismo tiempo (Figura 2-17).

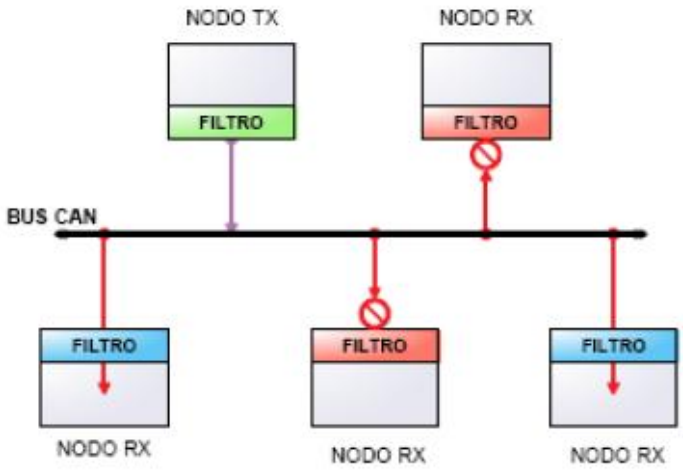


Figura 2-17: Sistema *Multicast* Bus CAN

- Gran fiabilidad y robustez en la transmisión. Detecta errores, los señaliza, envía mensaje de error y reenvía el mensaje corrupto una vez el bus vuelva a estar activo. Además, puede operar en ambientes con condiciones extremas de ruido e interferencias gracias a que es un bus diferencial.
- Es un protocolo de comunicaciones normalizado, con lo que se simplifica y economiza la tarea de comunicar subsistemas de diferentes fabricantes sobre una red común o bus.
- Al ser una red multiplexada, reduce considerablemente el cableado y elimina las conexiones punto a punto.
- El procesador principal delega la carga de comunicaciones a un periférico inteligente (controlador), por lo tanto, el procesador principal dispone de mayor tiempo para ejecutar sus propias tareas.

2.3.3. Componentes del Bus CAN

Un Bus CAN consta de un controlador, un transceptor, dos elementos finales del bus y dos cables para la transmisión de datos (Figura 2-18). Con excepción de los cables del bus, todos los componentes están alojados en las unidades de control.

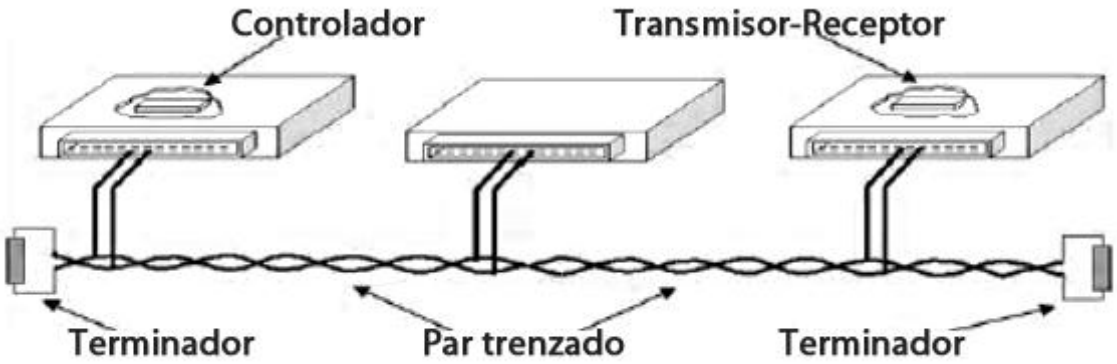


Figura 2-18: Componentes de un Bus CAN

2.3.3.1. Controlador

Recibe del microprocesador, en la unidad de control, los datos que han de ser transmitidos. Los acondiciona y los pasa al transceptor CAN. Asimismo, recibe los datos procedentes del transceptor CAN, los acondiciona y los traslada al microprocesador en la unidad de control.

2.3.3.2. Transceptor

Es un transmisor y receptor (*transceiver*) que transforma los datos del controlador en señales eléctricas y transmite éstas sobre los cables del Bus CAN. Además, realiza el proceso inverso, recibiendo los datos del Bus y transformándolos para el controlador.

2.3.3.3. Impedancias de carga

Se trata de una resistencia (típicamente 120Ω) que se coloca en cada uno de los extremos del Bus y que evita que los datos transmitidos sean reflejados en los extremos produciéndose errores en la trasmisión.

2.3.3.4. Cables del bus de datos

Es un par trenzado que funciona de manera bidireccional y sirve para la transmisión de los datos. Se denominan con las designaciones *CAN-High* (señales de nivel lógico alto) y *CAN-Low* (señales de nivel lógico bajo).

2.3.3.5. Conectores

Son varios los tipos de conectores especificados para los buses CAN. Suelen ser los protocolos de la capa de aplicación los que indican el tipo de conector. Los más utilizados son los DB-9, *MiniDIN* de 5 patillas y el de tipo regleta (Figura 2-19).

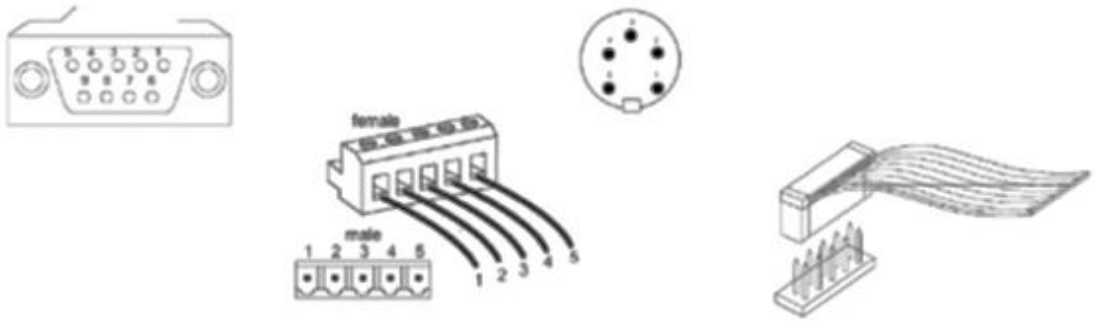


Figura 2-19: Tipos de conectores del Bus CAN

2.3.4. Tipos de Implementación

Teniendo en cuenta los elementos básicos que conforman el Bus CAN, existen tres tipos de implementaciones donde la comunicación es igual para todas. La diferencia radica en los filtros de aceptación, en la capacidad de almacenamiento de las tramas, en la responsabilidad que asume el microcontrolador o el controlador CAN, etc., es decir, en el hardware del nodo.

Las tres implementaciones incorporan un microcontrolador, ya que este componente representa una herramienta de hardware ideal para el desarrollo de aplicaciones con conexión CAN.

2.3.4.1. Basic CAN

Se trata de la arquitectura más simple, en la que existe un vínculo muy fuerte entre el controlador CAN y su microcontrolador asociado.

El controlador dispone de muy pocos registros de recepción de tramas (*buffers* o buzones) y es el microcontrolador quién lleva el peso de las tareas, siendo interrumpido para tratar cada trama CAN que reciba y evitar sobreescrituras en el *buffer* del controlador. Esto hace que el controlador CAN sea más simple y, por tanto, más barato. Se definen registros de máscara y filtro para dejar pasar selectivamente sólo aquellas tramas que sean de interés para el microcontrolador.

Este método es bueno para nodos encargados de manejar informaciones esporádicas, disminuyendo la ocupación del bus.

2.3.4.2. Full CAN

En esta arquitectura, el controlador CAN dispone de un número elevado de buzones de entrada y salida (típicamente 15). Cada buzón puede estar configurado para recibir

únicamente mensajes con un identificador concreto y puede transmitir y recibir mensajes sin ayuda del microcontrolador.

Este sistema de buzones actúa como una memoria compartida con la red que flexibiliza mucho los requerimientos de acceso desde el microcontrolador. En definitiva, el controlador reduce la carga de trabajo de éste. Además, se pueden habilitar interrupciones en el microcontrolador para notificarle la llegada de un mensaje.

Este tipo de arquitectura consiste en un microcontrolador que incluya no sólo sus características propias, sino, además, un módulo CAN con las características de un microcontrolador CAN. El *transceiver* se sitúa de manera separada.

2.3.4.3. Serial Linked I/O

Los dispositivos *Link Input/Output (SLIOS)* son controladores de bajo coste e inteligencia y sin capacidad de programación. Son interfaces preconfiguradas que requieren de un nodo CAN programable para controlarlo y son usados para entradas y salidas lejanas del bus. Son dispositivos esclavos físicamente direccionados con *jumpers* o con *switches DIP*.

2.3.5. Protocolo de Comunicaciones CAN

CAN fue desarrollado inicialmente para aplicaciones en los automóviles y, por lo tanto, la plataforma del protocolo es resultado de las necesidades existentes en el área de la automoción. La Organización Internacional para la Estandarización (ISO, *International Organization for Standarization*) define dos tipos de redes CAN:

La red de alta velocidad

- Con una velocidad de hasta 1 Mbps, bajo el estándar ISO 11898-2, es destinada para controlar el motor e interconectar las unidades de control electrónico (ECU).
- El establecimiento de una red CAN para interconectar los dispositivos electrónicos internos de un vehículo tiene la finalidad de sustituir o eliminar el cableado. Las ECUs, sensores, sistemas antideslizantes, etc. se conectan mediante una red CAN a velocidades de transferencia de datos de hasta 1 Mbps.

La red de baja velocidad

- Con una velocidad menor o igual a 125 Kbps, bajo el estándar ISO 11519-2/ISO 11898-3, es tolerante a fallos y, dentro del área de la automoción, es dedicada a la comunicación de los dispositivos electrónicos internos de un automóvil como son control de puertas, techo corredizo, luces y asientos.

De acuerdo al modelo de referencia OSI (*Open Systems Interconnection*) (Figura 2-21), la arquitectura de protocolos CAN incluye tres capas: física, de enlace de datos y aplicación, además de una capa especial para gestión y control del nodo llamada capa de supervisor.

TABLA 2-12: RELACIÓN ENTRE VELOCIDAD Y LONGITUD MÁXIMA DEL BUS CAN SEGÚN LA NORMA ISO 11898

Velocidad	Tiempo de Bit	Longitud máx.
1 Mbps	1 µs	30 m
800 Kbps	1.25 µs	50 m
500 Kbps	2 µs	100 m
250 Kbps	4 µs	250 m
125 Kbps	8 µs	500 m
50 Kbps	20 µs	1000 m
20 Kbps	50 µs	2500 m
10 Kbps	100 µs	5000 m

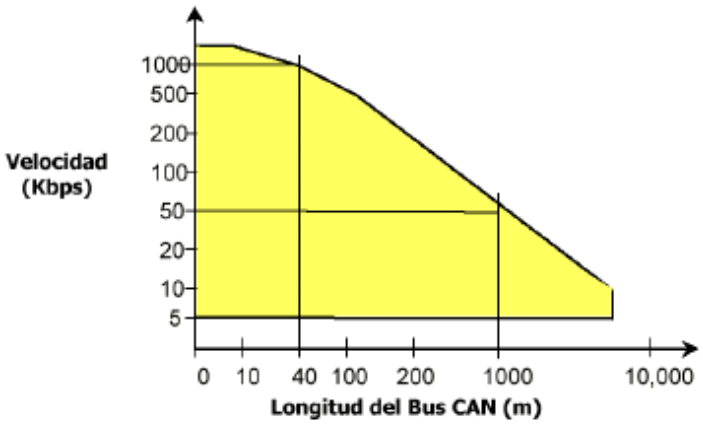


Figura 2-20: Relación entre velocidad y longitud máxima del Bus CAN

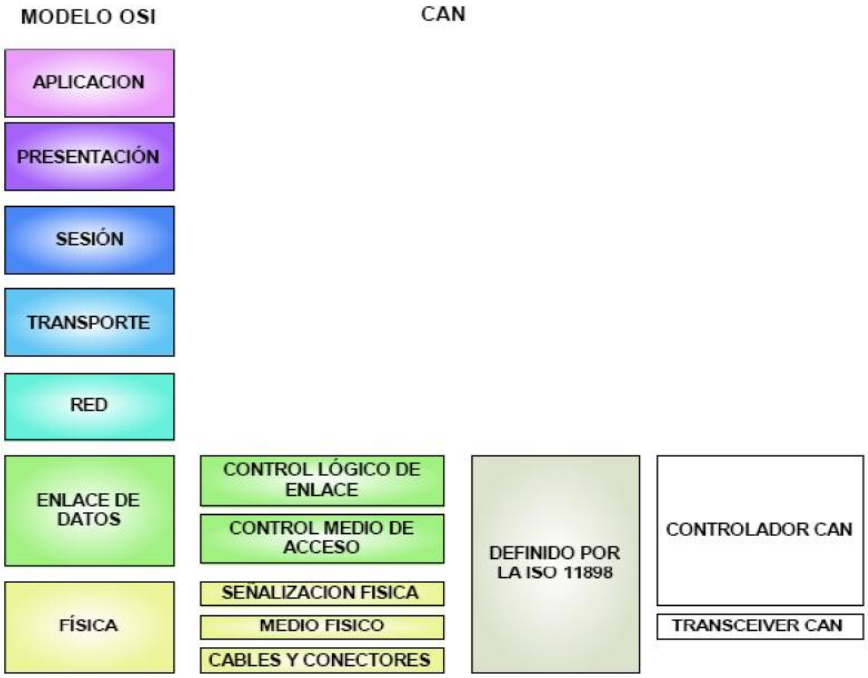


Figura 2-21: Relación entre el modelo OSI y CAN

2.3.5.1. Capa Física

Define los aspectos del medio físico para la transmisión de datos entre nodos de una red CAN, los más importantes son niveles de señal, representación, sincronización y tiempos en los que los bits se transfieren al bus. La especificación del protocolo CAN no define una capa física, sin embargo, los estándares ISO 11898 establecen las características que deben cumplir las aplicaciones para la transferencia en alta y baja velocidad.

La topología básica del bus CAN es de tipo *bus*, con derivaciones de longitud corta para las conexiones de los nodos. En algunos casos se puede adoptar una topología tipo *estrella*, pero a costa de sacrificar prestaciones de longitud máxima y ancho de banda.

El cable de bus es un par trenzado que debe estar finalizado en ambos extremos con impedancias de carga (típicamente 120Ω) para evitar las reflexiones de señal (Figura 2-22). En cuanto al número máximo de nodos que se pueden conectar al bus, se trata de una limitación de los transceptores y no de una especificación básica del bus CAN.

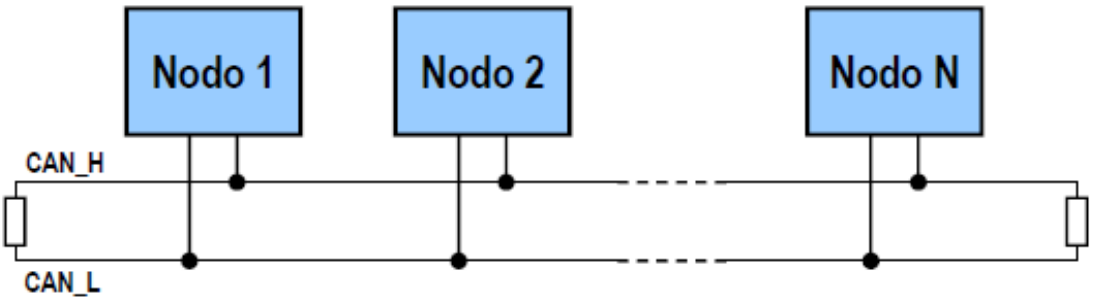


Figura 2-22: Conexión de nodos y terminaciones en un Bus CAN

El Bus CAN utiliza un método de señalización asíncrono NRZ (*Non Return* to Zero) y para mantener la sincronización realiza una técnica de relleno de bits (*bit stuffing*) por la que se inserta un bit de estado opuesto siempre que haya 5 bits seguidos de igual polaridad.

De todas las características eléctricas que define la capa física, es importante conocer los denominados niveles lógicos del bus. Al tratarse de un bus diferencial, éste está formado por dos señales (CAN_H y CAN_L) y la diferencia que existe entre ellas, que determinan el estado del bus. Por tanto, el Bus CAN dispone de dos niveles lógicos (Figura 2-23).

Normalmente en los sistemas digitales de dos niveles se conocen estos dos estados por nivel alto y nivel bajo, sin embargo, en este caso se denominan nivel dominante y nivel recesivo.

2.3.5.1.1. Nivel Dominante

Corresponde con la tensión diferencial entre los pines de comunicación (V_{CAN_H} - V_{CAN_L}), que ha de ser del orden de 2 V. Para conseguir esto es necesario que V_{CAN_H} tenga 3,5 V y V_{CAN_L} sea de 1,5 V (nominales). De hecho, si el voltaje de la línea CAN_H es al menos 0,9 V mayor que CAN_L, entonces ya se detectará la condición de bit dominante.

2.3.5.1.2. Nivel Recesivo

Corresponde con la tensión diferencial entre los pines de comunicación (V_{CAN_H} - V_{CAN_L}) que ha de ser del orden de 0 V. Para conseguir esto es necesario que V_{CAN_H} y V_{CAN_L} tengan 2,5 V (nominales). Aunque realmente el bus detectará una condición de recesivo si el voltaje de la línea CAN_H no es más alto que el voltaje de la línea CAN_L más 0,5 V.

TABLA 2-13: VALORES DE LOS BITS DOMINANTE Y RECESIVO EN EL BUS CAN

	Dominante	**Recesivo**
V_{CAN_H}	3'5V	2'5V
V_{CAN_L}	1'5V	2'5V

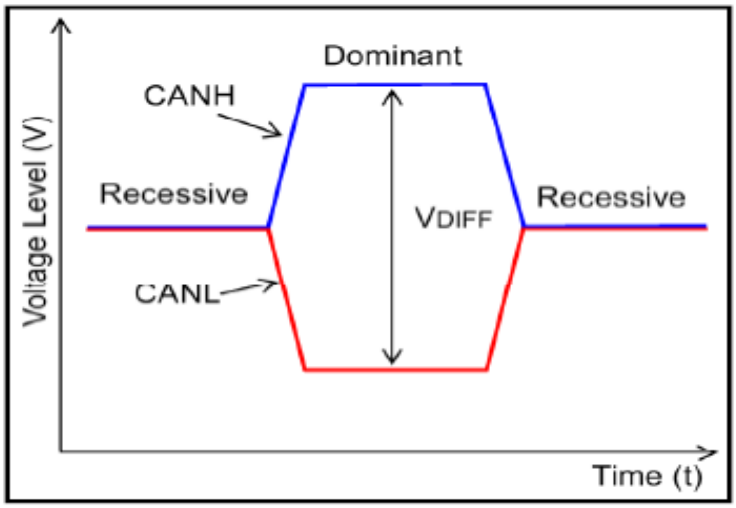


Figura 2-23: Identificación de las señales del Bus CAN

2.3.5.2. Capa de enlace de datos

Define las tareas independientes del método de acceso al medio, además debido a que una red CAN brinda soporte para procesamiento en tiempo real a todos los sistemas que la integran, el intercambio de mensajes que demanda dicho procesamiento requiere de un sistema de transmisión a frecuencias altas y retrasos mínimos. En redes multimaestro, la técnica de acceso al medio es muy importante ya que todo nodo activo tiene los derechos para controlar la red y acaparar los recursos. Por lo tanto, la capa de enlace de datos define el método de acceso al medio así como los tipos de tramas para el envío de mensajes.

Cuando un nodo necesita enviar información a través de una red CAN, puede ocurrir que varios nodos intenten transmitir simultáneamente. CAN resuelve lo anterior al asignar prioridades mediante el identificador de cada mensaje, donde dicha asignación se realiza durante el diseño del sistema en forma de números binarios y no puede modificarse dinámicamente. El identificador con el menor número binario es el que tiene mayor prioridad.

El método de acceso al medio utilizado es el de Acceso Múltiple por Detección de Portadora, con Detección de Colisiones y Resolución de Colisión (CSMA/CD+CR, *Carrier Sense*

Multiple Access with Collision Detection and Collision Resolution). De acuerdo con este método, los nodos en la red que necesitan transmitir información deben esperar a que el bus esté libre (detección de portadora). Cuando se cumple esta condición, dichos nodos transmiten un bit de inicio (acceso múltiple) y cada nodo lee el bus bit a bit durante la transmisión de la trama y comparan el valor transmitido con el valor recibido. Mientras los valores sean idénticos, el nodo continúa con la transmisión y si se detecta una diferencia en los valores de los bits, se lleva a cabo el mecanismo de resolución de colisión (arbitraje), sobreviviendo siempre la trama de mayor prioridad (que posee el identificador con el menor número binario) (Figura 2-24).

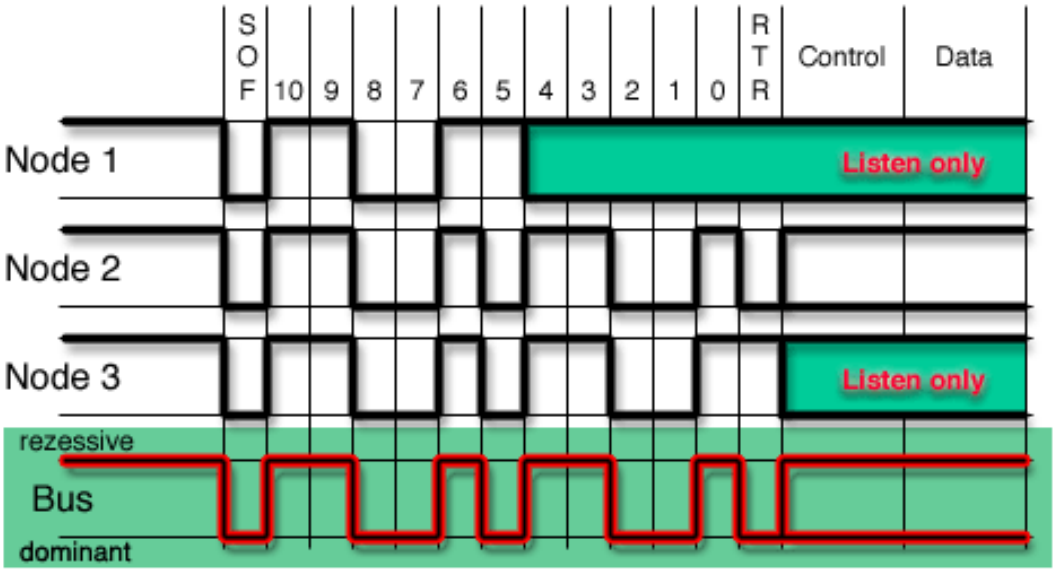


Figura 2-24: Sistema de resolución de colisiones Bus CAN

CAN establece dos formatos de tramas de datos (*data frame*) que difieren en la longitud del campo del identificador, las tramas estándares (*standard frame*) con un identificador de 11 bits definidas en la especificación CAN 2.0A, y las tramas extendidas (*extended frame*) con un identificador de 29 bits definidas en la especificación CAN 2.0B. Dentro de la especificación CAN 2.0B existe la versión Pasiva, donde sólo se aceptan y manejan tramas estándar del CAN, pero toleran la presencia de tramas extendidas sin generar errores (simplemente las ignoran), y la versión Activa, que aceptan y manejan ambos tipos de tramas estándar y extendida.

Para la transmisión y control de mensajes CAN, se definen cuatro tipos de tramas: de datos, remota (*remote frame*), de error (*error frame*) y de sobrecarga (*overload frame*). Las tramas remotas también se establecen en ambos formatos, estándar y extendido, y tanto las tramas de datos como las remotas se separan de tramas precedentes mediante espacios entre tramas (*interframe space*).

En cuanto a la detección y manejo de errores, un controlador CAN cuenta con la capacidad de detectar y manejar los errores que surjan en una red. Todo error detectado por un nodo, se notifica inmediatamente al resto de los nodos.

2.3.5.3. Capa de supervisor

La sustitución del cableado convencional por un sistema de bus serie presenta el problema de que un nodo defectuoso puede bloquear el funcionamiento del sistema completo. Cada nodo activo transmite un *flag* de error cuando detecta algún tipo de error que pueda ocasionar que un nodo defectuoso pueda acaparar el medio físico.

Para eliminar este riesgo el protocolo CAN define un mecanismo autónomo para detectar y desconectar un nodo defectuoso del bus, dicho mecanismo se conoce como aislamiento de fallos o tratamiento de errores, que se definirá más adelante.

2.3.5.4. Capa de aplicación

Existen diferentes estándares que definen la capa de aplicación del Bus CAN. Algunos son muy específicos y están relacionados con sus campos de aplicación. Los más populares que ofrece el mercado son: *CAN Open*, *Smart Distributed System*, *Device Net*, OSEK, J1939 o *CAN Kingdom*.

2.3.6. CAN Open

Es uno de los HLP (*Higher Layer Protocol*) más utilizados en las aplicaciones basadas en el bus CAN. Está desarrollado y supervisado por CiA [11], y originalmente estaba pensado para sistemas de control industrial orientados a control de movimiento, aunque su uso se ha extendido a aplicaciones diversas (vehículos comerciales, equipamiento médico, automatización de edificios, etc).

Las especificaciones cubren el nivel de aplicación, el perfil de la comunicación, el armazón de los aparatos programables de los nodos y recomendaciones de cables y conectores.

La definición de dispositivos *CANopen* se realiza en base a perfiles y sus especificaciones cubren varios aspectos recogidos en distintos estándares:

- **CiA DS-301:** capa de aplicación y perfil de comunicación.
- **CiA DSP-302:** entorno para dispositivos programables.
- **CiA DRP-303-1:** recomendaciones para cables y conductores.
- **CiA DRP-303-2:** unidades SI y representación de prefijos.
- **CiA DS-4XX**: familias de perfiles de dispositivo.

2.3.6.1. Perfiles de Dispositivo

Un perfil de dispositivo define su funcionalidad y los mecanismos de comunicación estandarizados que emplea para el intercambio de sus datos con otros dispositivos.

El uso de perfiles proporciona ventajas como:

- Facilitar la integración de dispositivos procedentes de fabricantes distintos.
- Ayudar al fabricante a crear productos estandarizados.

La funcionalidad obligatoria permite que sea posible al menos un modo de operación no específico del fabricante y existe la posibilidad de especificar funcionalidad opcional no indicada en el perfil (específica de cada fabricante), aunque el perfil sí indica la forma que esa funcionalidad opcional debe ser especificada.

Todo dispositivo *CANopen* debe implementar el perfil de comunicación definido en el estándar DS-301, que describe cómo debe usar este nodo los servicios de comunicación del protocolo. La especificación completa del dispositivo se consigue mediante la implementación de un perfil de dispositivo.

Existen varios perfiles de dispositivo definidos en *CANopen* por sus respectivos estándares, algunos de los cuales son los mostrados en la Tabla 2-14.

TABLA 2-14: PERFILES *CANOPEN*

Perfil	Uso
DS-401	Módulos Genéricos de E/S
DS-402	Motores y Control de Movimiento
DS-403	Máquinas con Interfaz Humano
DS-404	Dispositivos de Medida y Controladores de Bucle-Cerrado
DS-405	Dispositivos Programables IEC 61131-3
DS-406	Codificadores de Posición (Encoders)

2.3.6.2. Diccionario de Objetos

Se trata de un grupo de objetos definido en cada nodo de la red (y que es accesible a través de ésta) con una estructura pre-definida. Cada objeto del diccionario se accede mediante un índice de 16 bits, por lo que se pueden definir hasta 65536 objetos distintos.

La estructura general del diccionario de objetos CANopen es la que se puede ver en la Tabla 2-15.

TABLA 2-15: ESTRUCTURA GENERAL DEL DICCIONARIO DE OBJETOS CANOPEN

Índice (hex)	Objeto
0000	No usado
0001-001F	Tipos de datos estáticos
0020-003F	Tipos de datos complejos
0040-005F	Tipos de datos específicos del fabricante
0060-007F	Tipos de datos estáticos específicos del perfil
0080-009F	Tipos de datos complejos específicos del perfil
00A0-0FFF	Reservados para futuro uso
1000-1FFF	Área del perfil de comunicación
2000-5FFF	Área del perfil específico del fabricante
6000-9FFF	Área del perfil de dispositivo estándar
A000-FFFF	Reservado para futuro uso

2.3.6.3. Acceso al objeto

Para acceder al dato contenido en un objeto del diccionario se usa siempre una referencia formada por un índice y un subíndice.

El dato contenido en cada objeto puede estar representado por:

- **Variable simple**: se referencia directamente por el índice del objeto, y el subíndice siempre valdrá 0.
- **Estructura compleja**: el índice referencia a toda la estructura, mientras que el subíndice permite acceder a un elemento concreto dentro de ella.

Por ejemplo, en un módulo de interfaz RS-232 se podría definir el objeto de configuración de parámetros de comunicación en Tabla 2-16.

TABLA 2-16: OBJETO DE CONFIGURACIÓN DE UN MÓDULO DE INTERFAZ RS-232 EN CANOPEN

Índice	Subíndice	Variable Accedida	Tipo de dato
6092h	0	Nº de entradas	Unsigned8
	1	Baudios	Unsigned16
	2	Nº de bits de datos	Unsigned8
	3	Nº de bits de stop	Unsigned8
	4	Paridad	Unsigned8

2.4. Baterías de Ión-Litio

2.4.1. Características generales

Una batería es un conjunto de celdas electroquímicas, dispuestas generalmente en serie para aumentar el voltaje final, que es capaz de almacenar la energía liberada por una reacción química y luego suministrarla como energía eléctrica [12]. Cada celda electroquímica está formada por un electrodo positivo, un electrodo negativo, un electrolito y un circuito externo que pone en contacto a los electrodos. En el diseño primitivo de una batería de ión-litio, y como ocurre de modo general en las baterías recargables a temperatura ambiente, los electrodos son materiales de inserción. La razón de su amplio y eficaz uso se debe a que las reacciones de inserción electroquímica son simples y reversibles. Una reacción de inserción consiste en una reacción de estado sólido en la que la especie denominada huésped reacciona ocupando sitios vacantes en la estructura de otra especie a la que se denomina anfitrión.

La especie huésped puede ser tanto iónica como molecular (Li+, Na+, H2O, etc.), siendo la capacidad donadora de electrones una propiedad común a todas ellas. Por otro lado, las especies anfitrión suelen ser un sólido, por lo general cristalino (óxidos, calcogenuros, oxohaluros, grafito y haluros). Éstos tienen una estructura “abierta”, es decir, con localizaciones asequibles para el huésped, interconectadas para permitir su difusión. La red anfitrión debe reunir una serie de requisitos:

- Estabilidad termodinámica y cinética del sólido a la temperatura de reacción.

- Posiciones vacantes que pueden ser ocupadas por la especie huésped.
- Alta movilidad de la especie huésped dentro de la red anfitrión a la temperatura de reacción.
- Propiedades conductoras que permitan la movilidad de los electrones del circuito externo.

El funcionamiento de las baterías de ión-litio se basa en el proceso de inserción-desinserción de iones Li+ utilizando dos compuestos de intercalación como electrodos. El potencial de salida de la batería es el correspondiente a la diferencia de potencial entre ambos compuestos de intercalación, que puede estimarse de la diferencia de potencial de cada uno de ellos por separado respecto del par Li+/Li. Con la intención de que el potencial de la batería de ión-litio sea lo más alto posible, debe seleccionarse como electrodo positivo un material que tenga un potencial alto de intercalación respecto del litio, y como electrodo negativo uno que lo tenga lo más bajo posible.

Por tanto, uno de los electrodos de la batería debe contener previamente el litio en su estructura de tal manera que durante las sucesivas etapas de carga-descarga sean estos iones *Li*+ los que salgan de un electrodo para poder insertarse en el otro.

2.4.2. Proceso de carga

Durante este proceso se suministra energía en forma de corriente eléctrica a la batería, de tal manera que los iones *Li*+ fluyen del electrodo positivo al electrolito y de éste al electrodo negativo donde se produce la reducción de la especie anfitrión y la inserción del ión huésped (Li+). En el electrodo positivo se oxida el material de inserción. El electrolito permite el paso de iones pero no de electrones. Durante el proceso de carga, éstos últimos fluyen también del electrodo positivo al negativo pero a través del circuito externo. En consecuencia, el electrodo negativo se vuelve más negativo y el electrodo positivo más positivo, aumentando la diferencia de potencial entre ellos y, por tanto, el voltaje de la celda.

2.4.3. Proceso de descarga

En este proceso ocurre justamente contrario que en el proceso de carga. La batería suministra corriente eléctrica. Los electrones fluyen por el circuito externo hacia el electrodo positivo saliendo del electrodo negativo. Este proceso fuerza a los iones litio a salir (desinserción) del electrodo negativo produciéndose la oxidación del mismo a la vez que se insertan en el electrodo positivo, lo que origina la reducción de este último material. A medida que avanza el proceso de descarga, se modifica el potencial suministrado por cada electrodo, lo que origina una disminución del voltaje de salida de la batería.

La movilidad de los iones Li+ entre los electrodos es posible gracias a que ambos están en contacto con un electrolito formado por una sal de litio disuelta en un disolvente no acuoso. Entre las sales de litio más comunes destacan el $LiClO_4$ y $LiPF_6$, mientras que los disolventes más utilizados son el carbonato de etileno (EC), carbonato de propileno (PC), dimetoxietano (DME), carbonato de dietilo (DEC) y carbonato de dimetilo (DMC).

2.4.4. Construcción

La Figura 2-25 muestra la estructura de una batería de ión-litio. Una lámina de aluminio actúa de colector de corriente del cátodo y una lámina de cobre es el colector del ánodo, ya que este metal no reacciona con el ión litio a bajos voltajes. Entre las láminas se dispone el separador, constituido por unas delgadas películas porosas de propileno empapado del electrolito orgánico.

Como regla general, en el diseño de una batería de ión-litio, los compuestos que se utilizan como electrodos han de ser capaces de insertar litio siguiendo un proceso reversible. Este proceso ha de mantener una alta capacidad específica (energía liberada por unidad de masa de material activo, Ah/kg) durante los ciclos de carga-descarga. Además, sus estructuras tienen que ser estables a lo largo de estos ciclos sucesivos, de lo contrario, las tensiones continuadas acabarían destruyéndolas.

1. Cubierta del cátodo
2. Junta
3. Aislante
4. Contacto externo del cátodo
5. Orificio de seguridad
6. PTC (resistencia de seguridad)
7. Separador
8. Separador
9. Aislante
10. Pin central
11. Contenedor del ánodo
12. Cátodo
13. Contacto externo del ánodo
14. Ánodo

Figura 2-25: Estructura de una batería de ión-litio

2.4.5. Características y rendimiento

En la Figura 2-26 se detallan las principales características relativas al funcionamiento de las baterías de ión-litio.

Characteristic	Performance range
Operational cell voltage	4.2 to 2.5 V
Specific energy	100 to 158 Wh/kg
Energy density	245 to 430 Wh/L
Continuous rate capability	Typical: 1C High rate: 5C
Pulse rate capability	Up to 25C
Cycle life at 100% DOD	Typically 3000
Cycle life at 20 to 40% DOD	Over 20000
Calendar life	Over 5 years
Self discharge rate	2 to 10%/month
Operable temperature range	−40°C to 65°C
Memory effect	None
Power density	2000 to 3000 W/L
Specific power	700 to 1300 W/Kg

Figura 2-26: Características generales del funcionamiento de las baterías de ión-litio

Figura 2-27: Pack de baterías de Ión-Litio de 3 celdas

2.5. Sistema de Gestión de Baterías

Un BMS (*Battery Management System*) es un sistema cuya función principal es la de monitorizar, proteger y estimar el estado de una serie de baterías conectadas a él [13]. Gracias a la utilización de un BMS se consigue maximizar el rendimiento y vida de las celdas que componen una batería. Por seguridad y por el bien de las celdas de la batería, un BMS debe cumplir al menos las siguientes especificaciones:

- Evitar que la tensión de la celda supere el límite permitido, deteniendo la corriente de carga o solicitando la detención. Esto es un problema que ocurre a menudo en las baterías de ión-litio.
- Evitar que la temperatura de alguna de las celdas o del conjunto supere una temperatura determinada. Las baterías de ión-litio son propensas a este tipo de fugas térmicas.
- Evitar que el voltaje de alguna celda se encuentre por debajo de la tensión umbral mínima, deteniendo la corriente de carga o dando la orden de parada.
- Evitar que la corriente de carga supere el límite (dicho límite varía con la tensión de la celda, temperatura de la celda y nivel anterior de carga) solicitando que la corriente sea reducida o detenida.
- Evitar que la corriente de descarga sobrepase un límite, descrito en el punto anterior.

Un BMS es esencial en la carga de las baterías de ión-litio. En el momento en que alguna de las celdas alcance el máximo voltaje de carga, deberá desconectarse del cargador (Figura 2-28). Un BMS también debe balancear las celdas para maximizar su capacidad. El balanceo puede realizarlo desconectando la celda con más carga hasta que el voltaje sea suficientemente bajo para que el cargador vuelva a suministrar la corriente de carga. Después de algunos ciclos de este proceso todas las celdas deberían encontrarse a la misma tensión. Cuando el conjunto de celdas se encuentra balanceado significa que la carga se ha completado. Por otro lado, en cuanto alguna celda alcance la tensión mínima de corte permitida también se ha de desconectar de la carga (Figura 2-29).

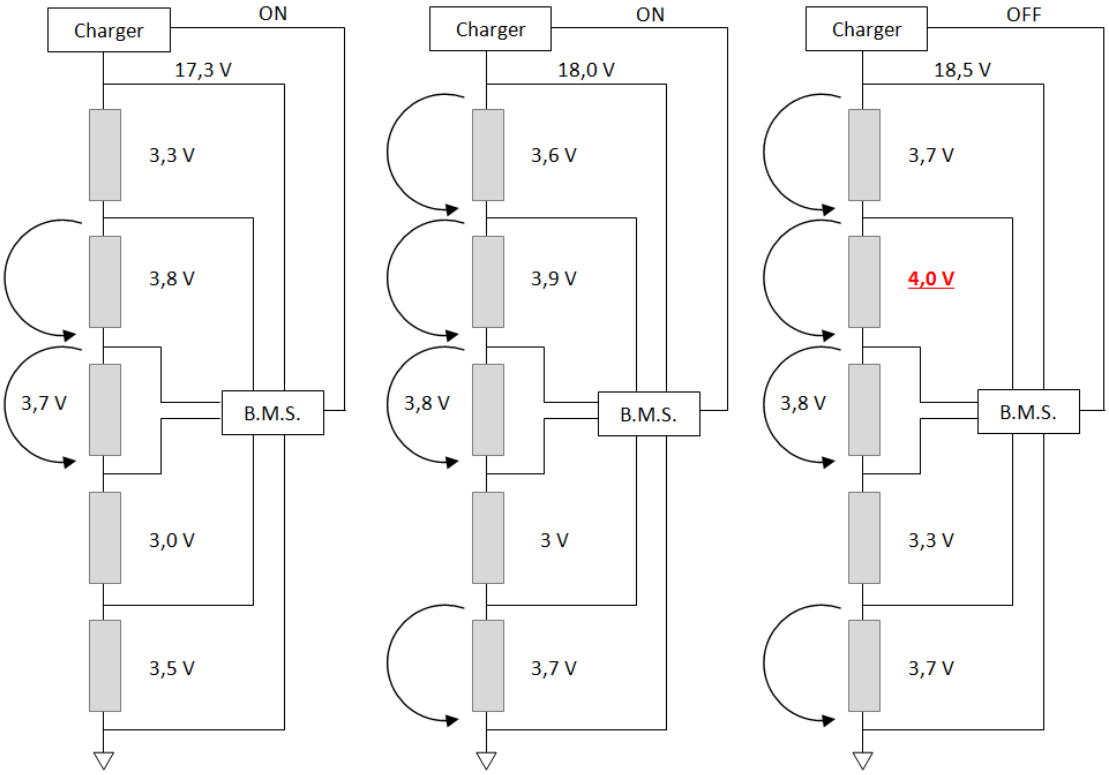


Figura 2-28: Proceso de balanceo de la carga de un conjunto de celdas

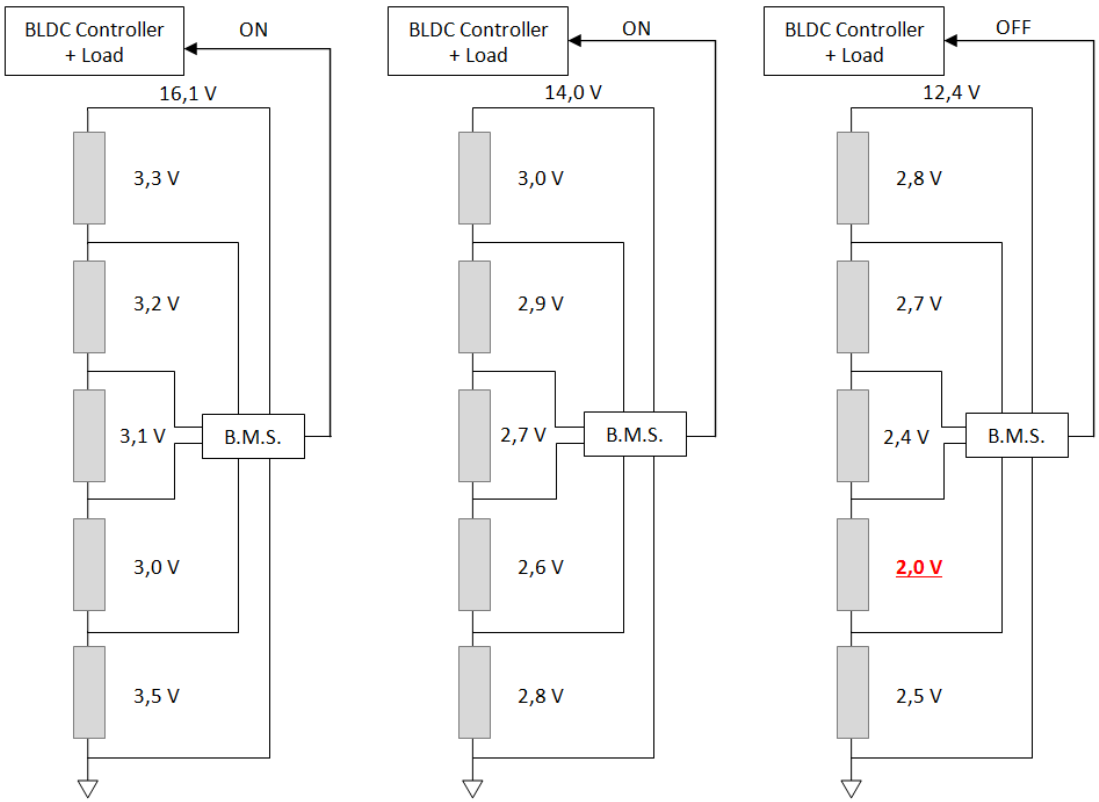


Figura 2-29: Proceso de balanceo en la descarga de un conjunto de celdas

3. Diseño de la Controladora BLDC

El diseño de la controladora BLDC se ha basado en el microcontrolador (µC) ATmega64M1 [14], el cual, por sus características, resulta idóneo para el control de este tipo de motores. Las principales características de este µC que son requeridas para este proyecto son las siguientes:

- Una CPU AVR de 8 bits.
- 11 canales ADC (*Analog to Digital Converter*) de 10 bits de resolución, uno de los cuales será utilizado para capturar la señal analógica del potenciómetro de ajuste de velocidad.
- Una interfaz CAN y otra RS-232 (*Recommended Standard 232*) utilizadas para realizar la comunicación de la controladora con el PC de test y monitorización.
- Un módulo PSC (*Power Stage Controller*) de alta velocidad con una resolución de 12 bits. Este módulo será el encargado de realizar el control de la etapa de potencia y de la conmutación de las señales de las bobinas del motor BLDC mediante la técnica de PWM (*Pulse Width Modulation*) [15][16][17].

3.1. Diseño del Circuito

Las etapas principales en las que se divide el diseño de la controladora son tres: la etapa de control, la etapa de comunicación y la etapa de potencia. En la etapa de control se encuentran las configuraciones de las entradas y salidas del µC y, en la etapa de comunicación, la electrónica con la que interconectamos el µC con el PC. En la etapa de potencia, se localizan los drivers encargados de elevar la tensión de salida del módulo PSC del µC hasta conseguir la tensión adecuada para la activación de los transistores MOSFET. Estos transistores son los encargados de excitar las bobinas del motor con la tensión correcta.

En la Figura 3-1 se observan los diferentes módulos de los que consta el diseño de la controladora.

3.1.1. Etapa de control

En la etapa de control se encuentra uno de los elementos más importantes de todo el sistema: el µC ATmega64M1. Éste se encarga de leer los valores obtenidos de los sensores de efecto Hall que posee el motor BLDC, que determinan la posición en la que se encuentra el rotor del motor. Mediante la lectura de estos valores, el microcontrolador genera unas señales PWM con una frecuencia de 16 kHz y una tensión de pico de 5 V. Estas señales se envían a los drivers de los MOSFETs, que forman parte de la etapa de potencia del circuito, desfasadas 120º entre sí. Este desfase viene dado por las señales de los sensores Hall (Figura 3-2).

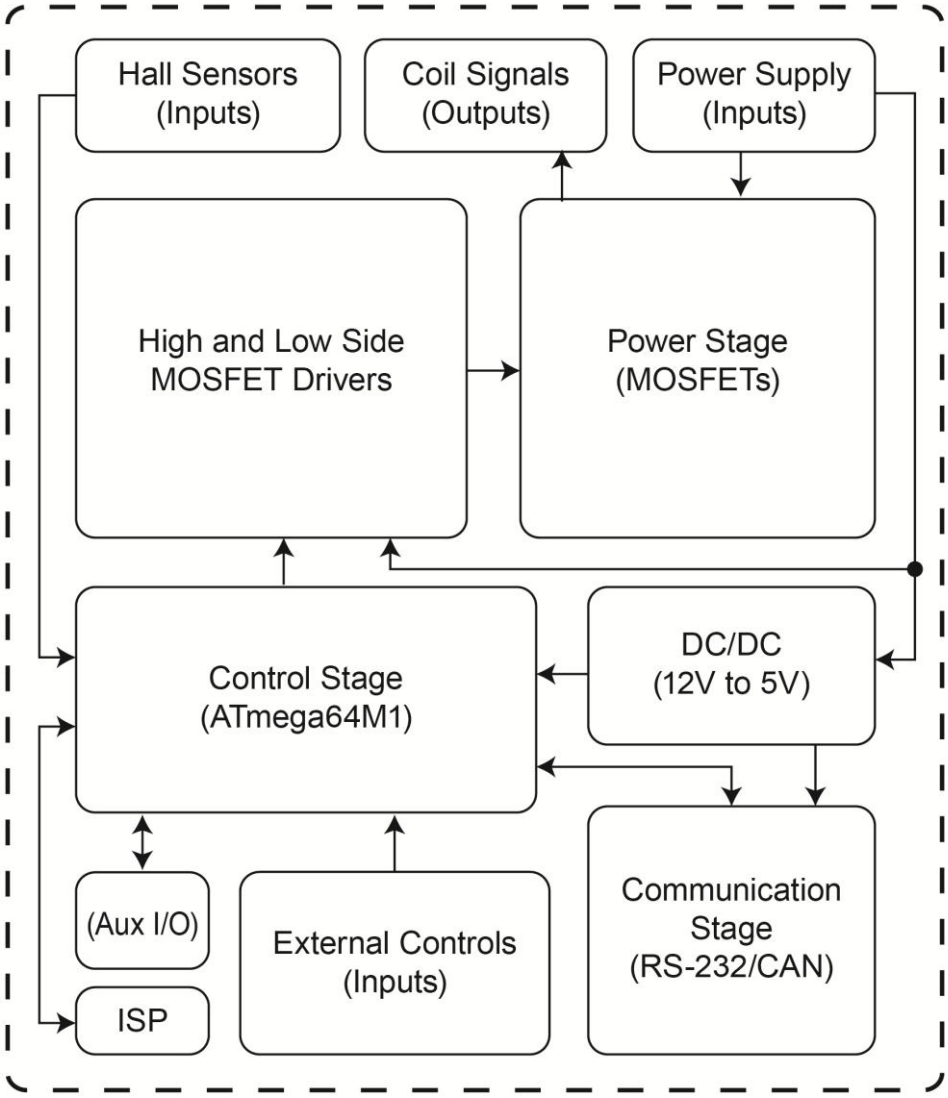


Figura 3-1: Diagrama de las partes que consta el diseño de la controladora BLDCM

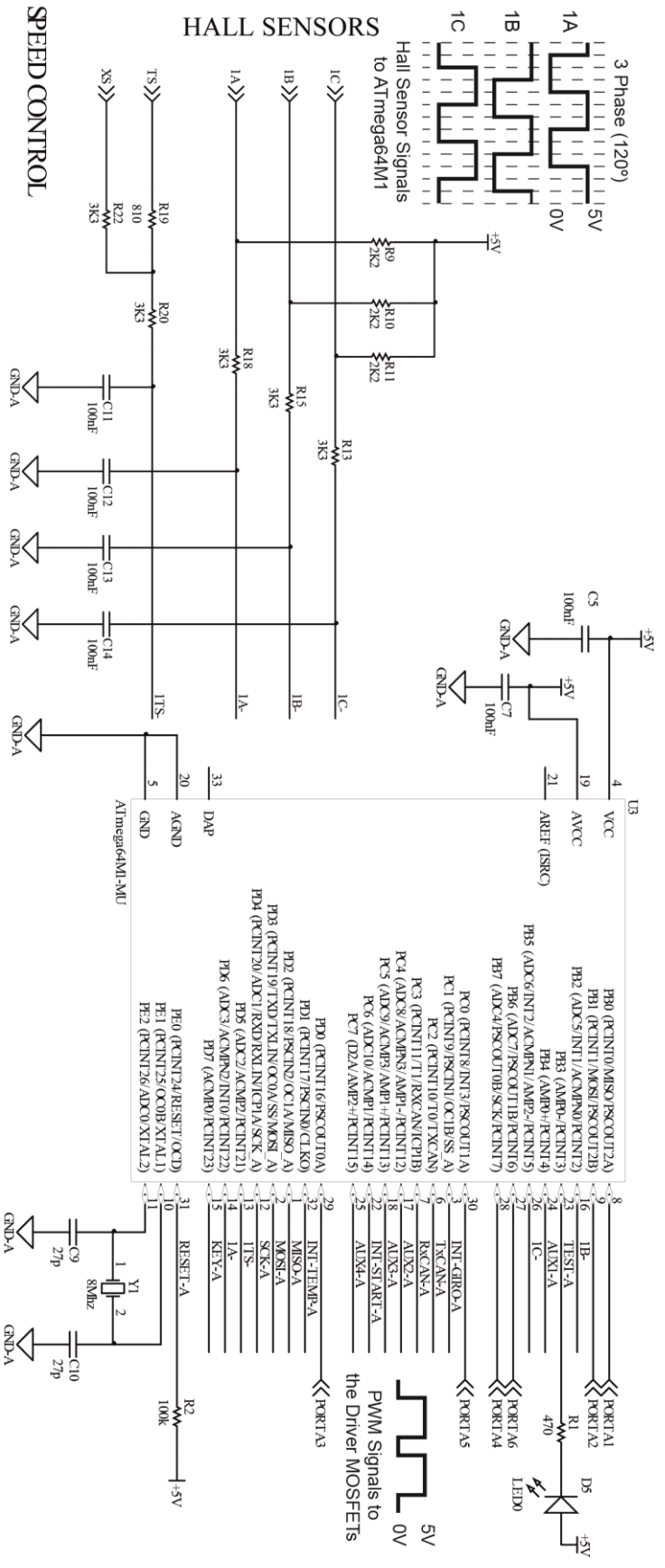


Figura 3-2: Etapa de control de la controladora BLDC

3.1.2. Etapa de potencia

En esta etapa se sitúan los drivers de los MOSFETs, que son alimentados con una tensión de 12 V y elevan la tensión de las señales PWM generadas por el µC a dicho valor. Con estas señales se atacan las puertas de los transistores de potencia MOSFETs (IRFP4568), habilitando el paso a la tensión de alimentación (Vcc) hacia el motor BLDC. Esta tensión de alimentación y corriente que fluye a través de los MOSFETs, siguiendo la lógica de conmutación establecida por el µC, excita las bobinas del motor haciendo que el mismo gire. La velocidad y el par de giro del motor lo determina el ciclo de trabajo de las señales PWM establecido por el potenciómetro de ajuste de velocidad.

En la Figura 3-3 se muestra una de las tres partes de las que se compone esta etapa de potencia. La bobina del motor puede encontrarse en 3 estados provocados por las señales *high*/*low* del driver y el desfase de 120º entre ellas:

- **VCC**: bobina conectada a la tensión de alimentación.
- **GND**: bobina conectada a tierra.
- **NC**: bobina no conectada.

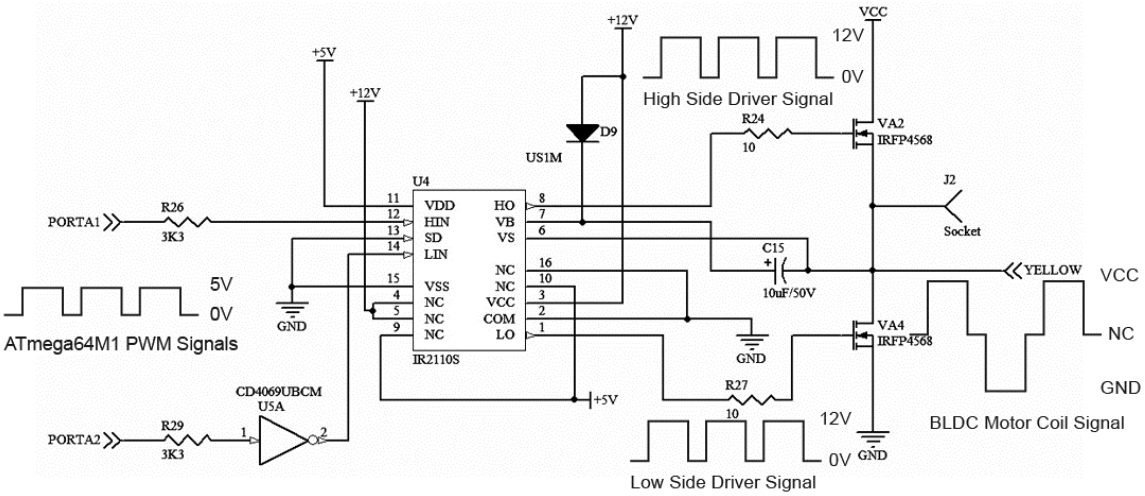


Figura 3-3: Esquemático de una de las tres partes de la etapa de potencia

3.1.3. Etapa de comunicación

Por otro lado, el diseño de la controladora dispone de una etapa de comunicación mediante la cual se conecta la controladora con una estación de trabajo (*PC o display*) a través de una comunicación serial (RS-232). Además, el microcontrolador ofrece la posibilidad de realizar esta conexión por medio del protocolo de comunicaciones CAN, que garantiza una mayor seguridad y robustez ante interferencias y colisiones en la transmisión de los datos. Es por ello, por lo que se decide habilitar la posibilidad de utilizar también este protocolo. Para este fin, se han incorporado al diseño dos conmutadores y dos *jumpers* con los que, según la combinación utilizada, permite seleccionar que al conector DB9 de salida esté conectado el

transceiver CAN o el *transceiver* RS-232. En la Figura 3-4 se muestra el esquemático de la etapa de comunicación de la controladora BLDCM.

Por último, cabe destacar el bloque en el que se realizan las conexiones de controles y sensores externos para la controladora. Éstos pueden ser:

- El interruptor de encendido del sistema.
- El potenciómetro de ajuste de velocidad del motor.
- El interruptor de cambio de sentido de giro del motor.
- El sensor de temperatura del motor.

Además, se sitúa de manera accesible el conector ISP (*In-System Programming*) para llevar a cabo la programación del µC con el software que se ha desarrollado para el control de este tipo de motores. En este proceso, se utiliza el software que ofrece el fabricante del µC (Atmel) de descarga gratuita desde su página web, el Atmel Studio 6.0 [18]. Con este software y el dispositivo programador AVRISP mkII [19] (Figura 3-5), se realiza la programación del µC de la controladora a través de la interfaz ISP.

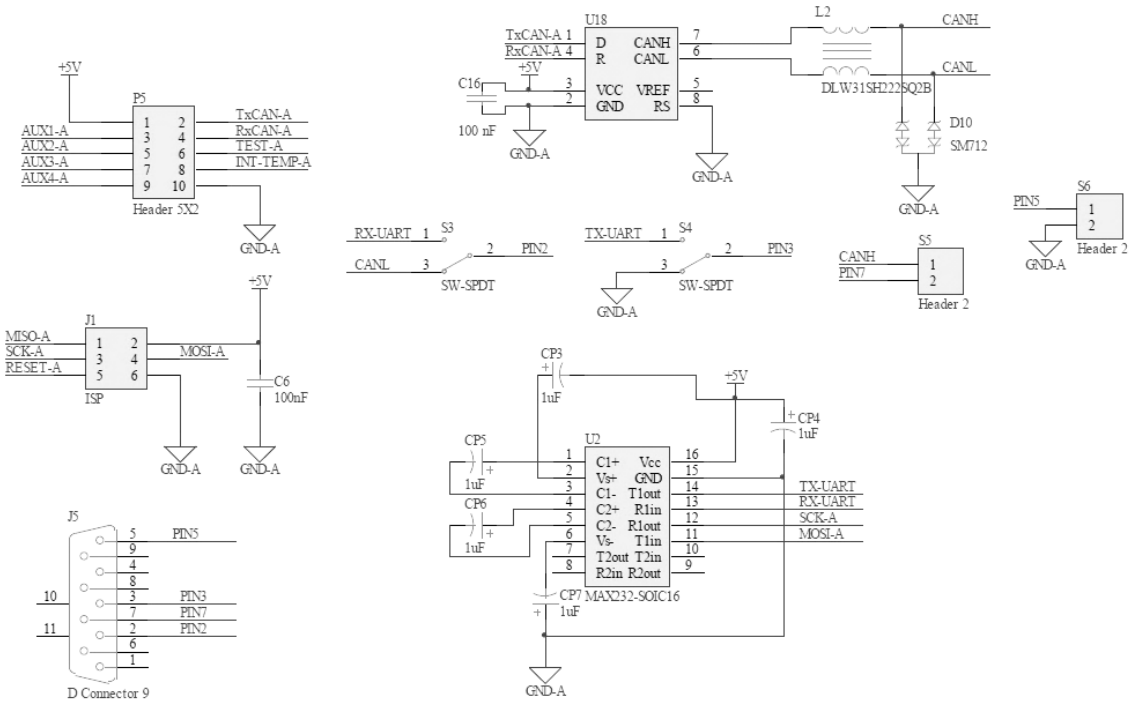


Figura 3-4: Etapa de comunicación de la controladora BLDC

Figura 3-5: Programador AVRISP mkII de Atmel

3.2. Diseño y Fabricación de la PCB

El diseño y fabricación de la PCB (*Printed Circuit Board*) de la controladora se ha realizado en el Laboratorio SFP del IUMA de la ULPGC (Servicio de Fabricación de Prototipos del Instituto Universitario de Microelectrónica Aplicada de la Universidad de Las Palmas de Gran Canaria) [20]. Se ha utilizado el formato *Eurocard* Simple de 100 mm x 160 mm, que permite que la controladora se pueda conectar en racks de una forma estandarizada.

3.2.1. Diseño en Altium Designer

Una vez realizados los esquemáticos del circuito completo de la controladora BLDC en el software *Altium Designer* [21], se procede al diseño de la PCB de la controladora. Para ello se asigna a cada componente su *footprint* (huella) en el esquemático. El *footprint* determina la forma física del componente. En él se especifica el número de pines, el tipo de tecnología (componente de montaje superficial o de inserción), así como sus dimensiones. En este diseño se han usado componentes de inserción y componentes de montaje superficial (o SMD).

El siguiente paso en el diseño de la PCB es establecer las dimensiones de la misma, como se explicó anteriormente, el tamaño será de 160 mm x 100 mm que corresponde al formato *Eurocard* o Simple Europa. Además, se establece el número de capas de las que constará la PCB. En este caso se trata de un diseño de 2 capas con el montaje de los componentes en la capa superior (*top layer*).

A continuación, se importan las conexiones pertenecientes a los diversos esquemáticos realizados, así como los *footprints* de los propios componentes. Una vez hecho esto, hay que proceder a situar los *footprints* de los componentes teniendo en cuenta que algunos de ellos deben colocarse en una posición fija, como son los conectores de la parte trasera y frontal de la PCB. Es por esto por lo que serán los primeros componentes en ser ubicados.

3.2.1.1. Reglas de diseño

Las reglas de diseño establecen las normas a seguir durante el proceso de diseño de la placa de circuito impreso. A continuación, se muestran las características más relevantes de dichas reglas:

- *Electrical clearance*: Determina la distancia mínima que debe existir entre pistas y entre un plano de alimentación y una pista. Se ha empleado una distancia mínima de 0.254 mm.

Figura 3-6: *Electrical clearance* mínimo establecido para el diseño de la PCB

- *Routing width*: Especifica el tamaño mínimo, máximo y preferente para el ruteado de las pistas. El tamaño mínimo de ancho de pista empleado es de 0.254 mm y el límite máximo de 5 mm.

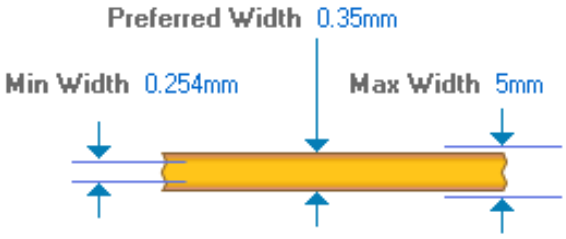


Figura 3-7: *Routing width* establecido para el diseño de la PCB

- *Routing vias style*: Establece el tamaño de las vías empleadas en el diseño de la PCB, tanto para el taladro pasante como para su corona. Las vías se han especificado con un ancho de taladro de 0.7 mm y 1.27 mm de corona como mínimo.

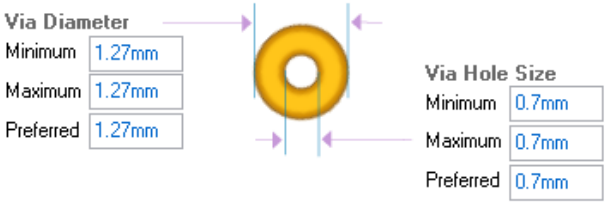


Figura 3-8: *Routing vias style* establecido para el diseño de la PCB

3.2.1.2. Recomendaciones de diseño

A la hora de realizar el ruteado de la PCB, se prescinde de utilizar la herramienta de *autorouting* (autoruteado) que posee el programa. Para obtener una mayor eficacia en el ruteo y distribución de los componentes, se decide realizar un ruteado manual.

Se han sobre dimensionado las pistas de la zona de conmutación y alimentación, de manera que no se encuentren afectadas por un exceso de corriente.

Se realiza un chequeo de errores con el fin de detectar posibles alteraciones en el circuito así como, pistas sin conectar, pistas en circuito abierto, etc.

3.2.1.3. Resultado del diseño

En la Figura 3-9 se muestra el resultado final obtenido del ruteo PCB. Se trata de un diseño de 2 capas con el montaje de los componentes en la capa superior. En la Figura 3-10 se muestra una vista en 3D del diseño final del prototipo. En el Anexo I se detallan los esquemáticos y capas de la PCB utilizados para la fabricación de la controladora BLDCM.

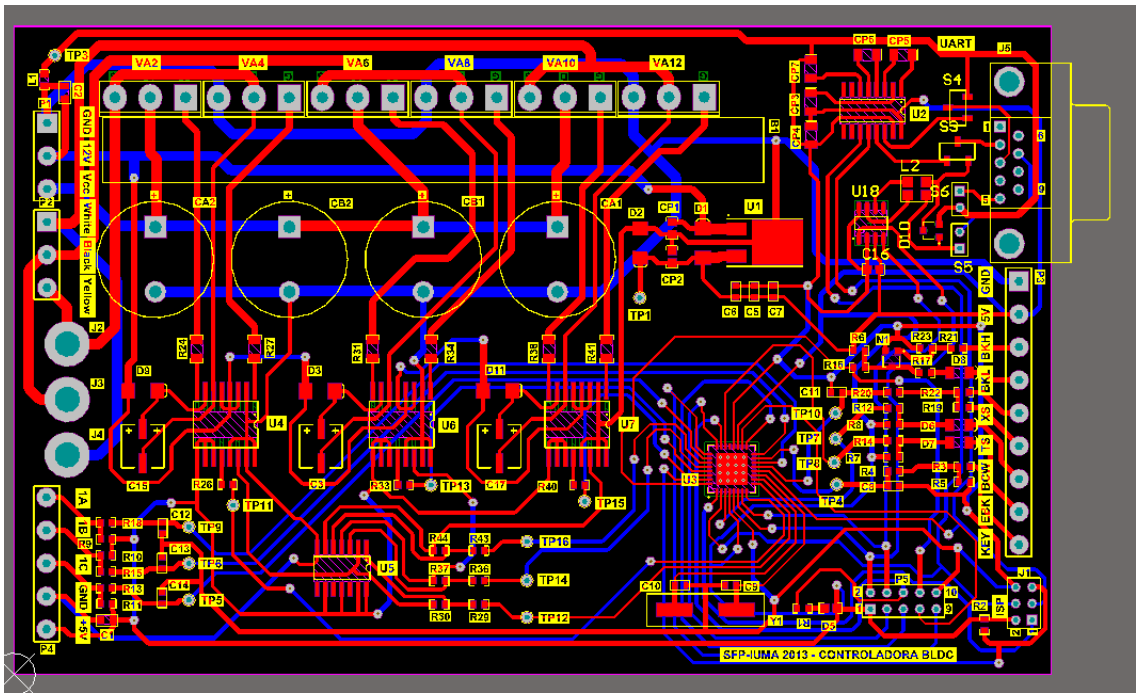


Figura 3-9. Diseño final ruteado de la PCB de la controladora

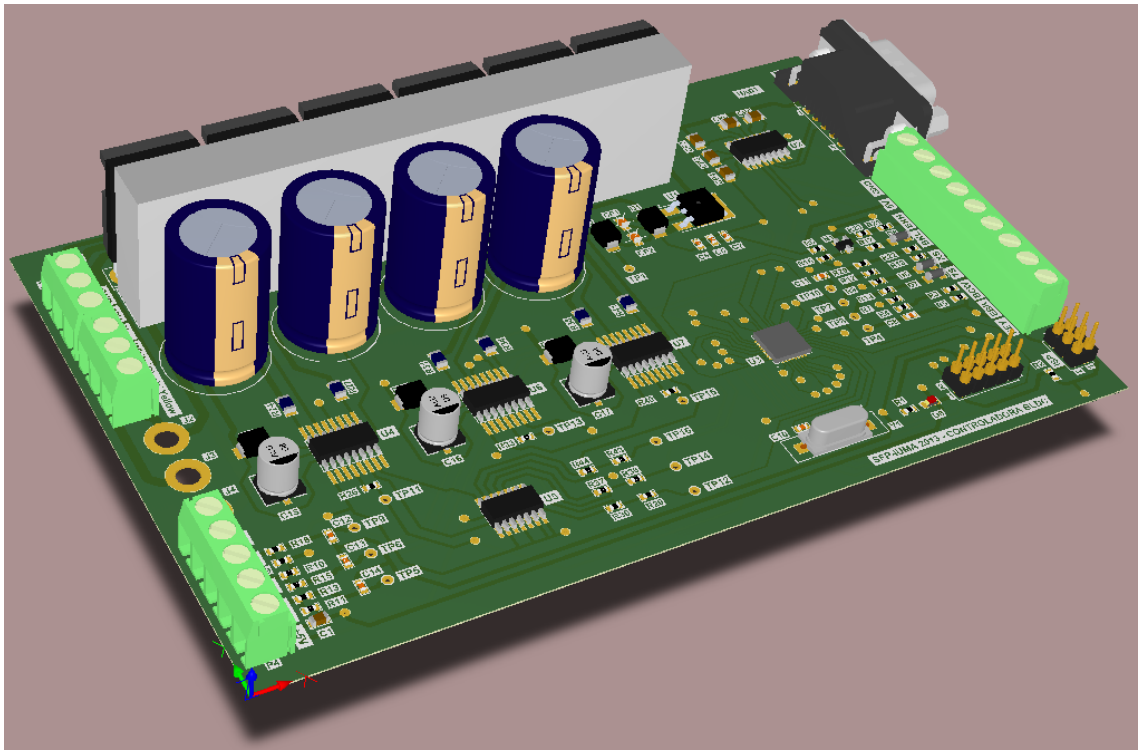


Figura 3-10: Vista 3D de la controladora BLDCM diseñada

3.2.2. Fabricación de la PCB

Para la fabricación de la PCB se generan, desde *Altium Designer*, una serie de ficheros conocidos como *Gerber Files* (Figura 3-11). Estos ficheros son empleados para imprimir los fotolitos de cada capa utilizada en el diseño de la PCB mediante una *photoplotter* (Figura 3-12). Por otro lado, desde el software *Altium* se exportan los ficheros utilizados para realizar el taladrado de la placa y para fabricar la máscara de soldadura (*stencil*) de la PCB.

Una vez finalizado el proceso de fabricación se procede a realizar el montaje, de forma manual, de los componentes SMD (*Surface Mount Devices*). Este proceso se lleva a cabo aplicando pasta de soldadura mediante una espátula sobre los PAD de montaje superficial de los componentes. Para ello, se utiliza una pantalla de soldadura (*stencil*) que evita que se aplique pasta en lugares no deseados (Figura 3-13). Posteriormente, se colocan los componentes SMD utilizando una máquina *Pick&Place* manual (Figura 3-14).

Cuando se encuentran todos los componentes colocados en su correcta posición, se introduce la placa en un horno de refusión para soldarlos (Figura 3-15). Finalmente, se sueldan manualmente los componentes de inserción de la placa y se coloca en su caja de protección correspondiente (Figura 3-16).

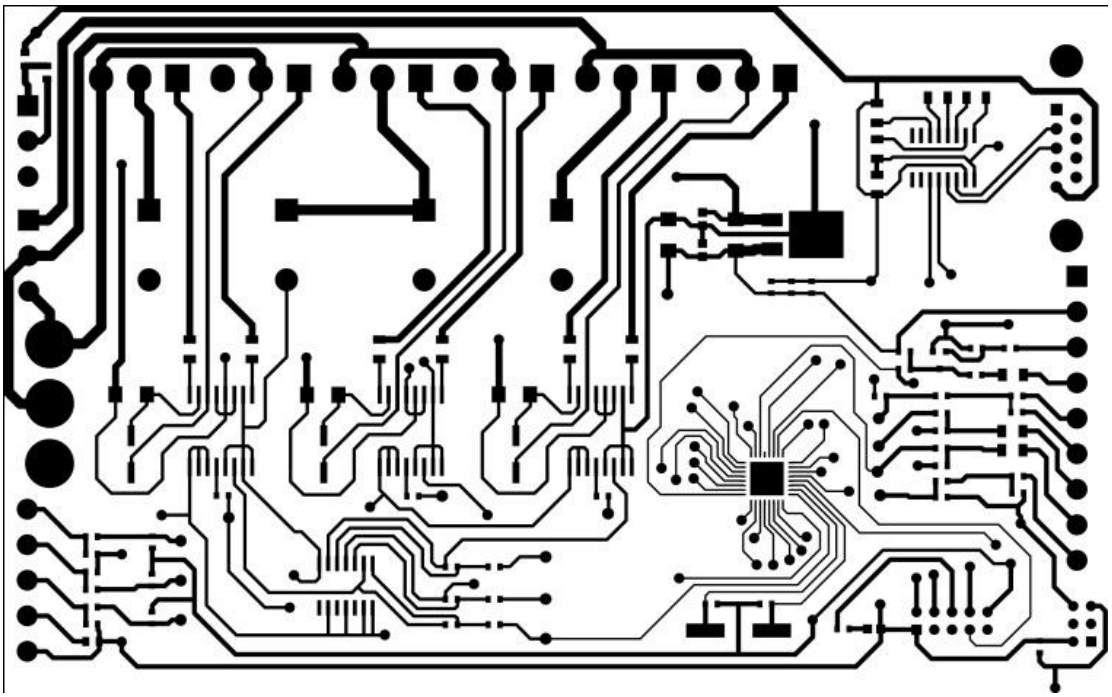


Figura 3-11: *Gerber file* de la capa *top layer* de la controladora BLDC

Figura 3-12: *Photoplotter* utilizada para la impresión de los fotolitos de la PCB

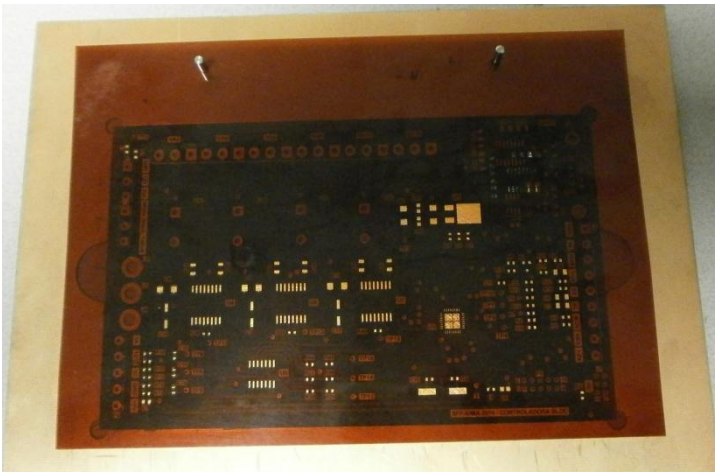


Figura 3-13. PCB y máscara de soldadura (*stencil*) lista para empastar

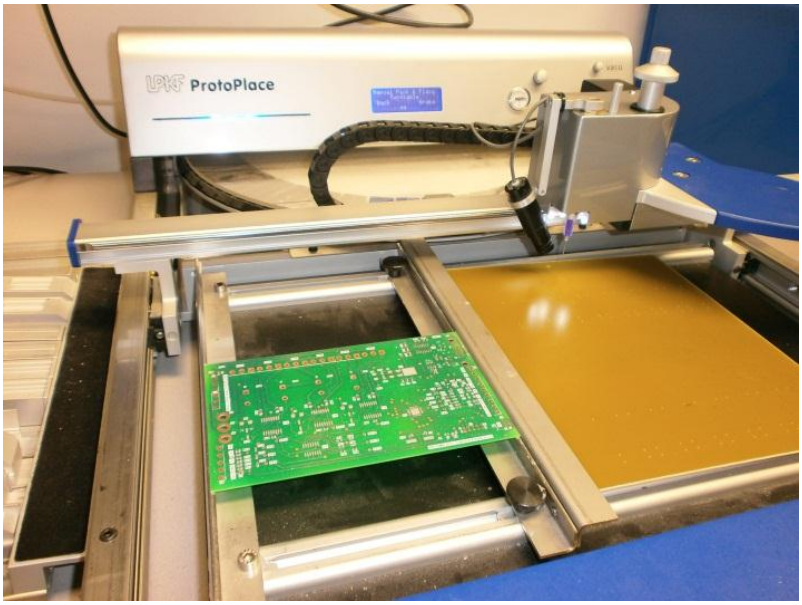


Figura 3-14. PCB y *Pick&Place* manual para la colocación de los componentes SMD

Figura 3-15. PCB y horno de refusión

Gracias a que el Laboratorio SFP del IUMA permite la fabricación de placas de 2 capas, el coste de las mismas se reduce a los materiales y componentes utilizados. Además, éstos últimos son de uso habitual y de fácil adquisición. Por otro lado, cabe destacar que el diseño modular de la controladora permite su uso en diferentes tipos de aplicaciones. Así pues, dados todos estos aspectos, obtenemos una controladora de motores BLDC de bajo coste.

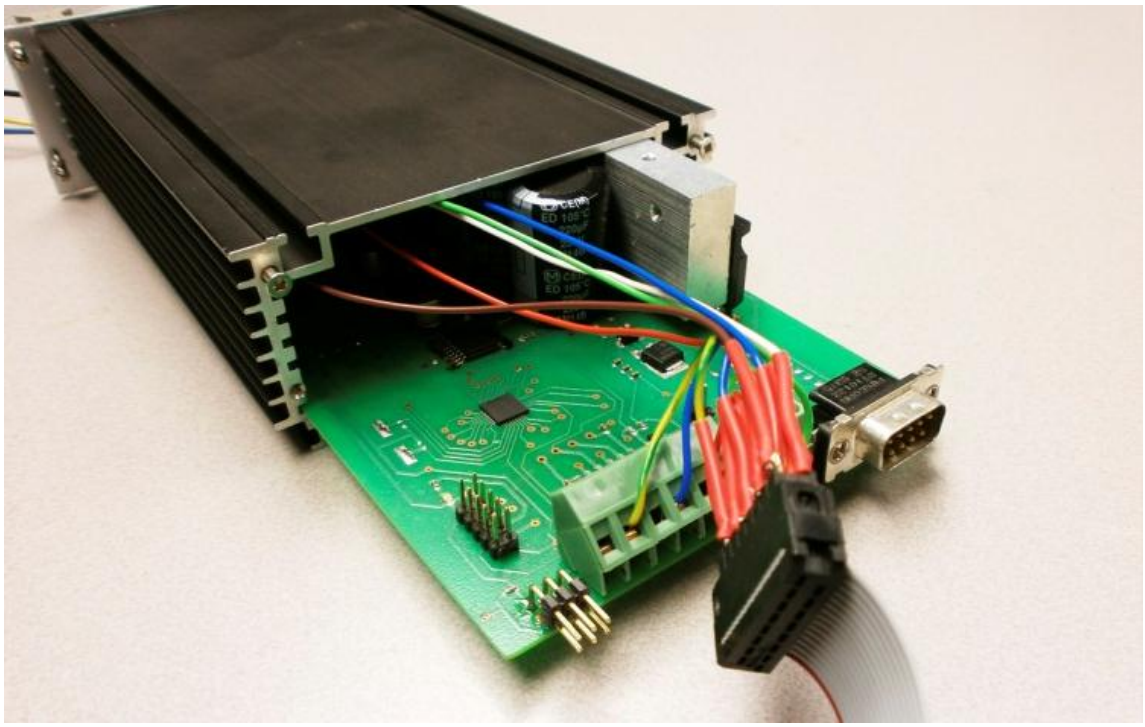


Figura 3-16: Vista real de la controladora BLDCM fabricada

4. Librería C para el control del motor BLDC

4.1. Introducción

Como ya se mencionó anteriormente, los motores BLDC reemplazan la conmutación de las escobillas por una conmutación electrónica señalizada por sensores Hall de estado sólido. Estos sensores indican al microcontrolador de la controladora la posición en la que se encuentra el rotor del motor. La controladora, a partir de estas señales, debe provocar la excitación de las bobinas del estator del motor siguiendo una lógica de conmutación correcta. Es por esto, por lo que se hace necesario disponer de un software que realice estas operaciones. Además, dicho software debe tener en cuenta otros aspectos como el ajuste de velocidad del sistema, el sensor de temperatura del motor (en caso de poseerlo) o las comunicaciones hacia el exterior.

En la actualidad, la tendencia a la hora de realizar casi cualquier tipo de software es la de desarrollarlo de forma modular. Uno de los beneficios de esta metodología de desarrollo es obtener la posibilidad de reutilizar el código elaborado, consiguiéndose una gran productividad y reducción de tiempo en futuros trabajos en los que se puedan utilizar los módulos creados anteriormente.

La programación modular se basa en dividir un problema grande y complejo en varios subproblemas más pequeños y simples. Estos subproblemas, a su vez, deben ser divididos en otros más simples. Así se debe repetir estos pasos hasta obtener elementos lo suficientemente simples como para poder ser resueltos fácilmente. Esta técnica se denomina “refinamiento sucesivo o análisis descendente”.

Para poder comprobar el correcto funcionamiento de la controladora fabricada en este proyecto y desarrollar el sistema de control completo, se ha elaborado el software en C para programar el microcontrolador ATmega64M1. Este µC posee una capacidad de memoria *flash* de 64 kB y una memoria interna SRAM (*Static Random Access Memory*) de 4 kB junto con las características mencionadas en el apartado del diseño de la controladora BLDC. Dentro de esta serie ATmega, también existen el Atmega16M1 y el ATmega32M1 con una capacidad de memoria *flash* y SRAM de 16 kB y 1 kB, y 32 kB y 2 kB respectivamente. La elección del modelo de µC se realiza en función de la aplicación y del tamaño del programa. La razón de escoger el ATmega64M1 para este proyecto es la de poseer mayor capacidad de memoria *flash* donde alojar futuras ampliaciones del software.

4.2. Desarrollo de la Librería C

El desarrollo de la librería de funciones para el control de motores BLDC se ha basado en la técnica de control en lazo abierto (*Open Loop*) con ajuste de velocidad [22][23]. En el Anexo II se presenta el código de programa elaborado. En el diagrama de bloques de la Figura 4-1 se muestran los elementos más importantes utilizados del µC ATmega64M1.

Las señales de los sensores Hall (A, B y C), provenientes del motor, se introducen en el µC a través de los puertos que poseen las interrupciones externas prioritarias. Estas interrupciones actúan sobre el proceso principal del sistema activando la conmutación de las salidas Q1 a Q6 del módulo PSC en el orden correcto. La interrupción del sensor Hall A se encarga de realizar la medición de velocidad del motor mediante el temporizador 0.

La señal de entrada "*Key*" se encarga de arrancar o parar el motor. Esta señal, activa a nivel bajo, se introduce a través de la interrupción por cambio de pin número 23. Esta interrupción es ejecutada cuando se produce un cambio de nivel en la señal. Dentro del proceso principal, es la encargada de desactivar o activar el módulo PSC. Este módulo se encarga de generar las 6 señales PWM, Q1 a Q6, que atacan los drivers de los transistores MOSFETs. Estos transistores son los responsables de excitar las bobinas del motor BLDC.

El ajuste de velocidad del motor se realiza mediante una señal analógica digitalizada a través de uno de los canales del ADC. Esta señal digitalizada es la encargada de establecer el ancho de pulso (*Duty Cycle*) de la señal PWM que utiliza el módulo PSC. Esta señal PWM de 16 kHz se genera a partir de la señal de reloj del PLL (*Phase-Locked Loop*) interno del µC, establecida a 32 MHz.

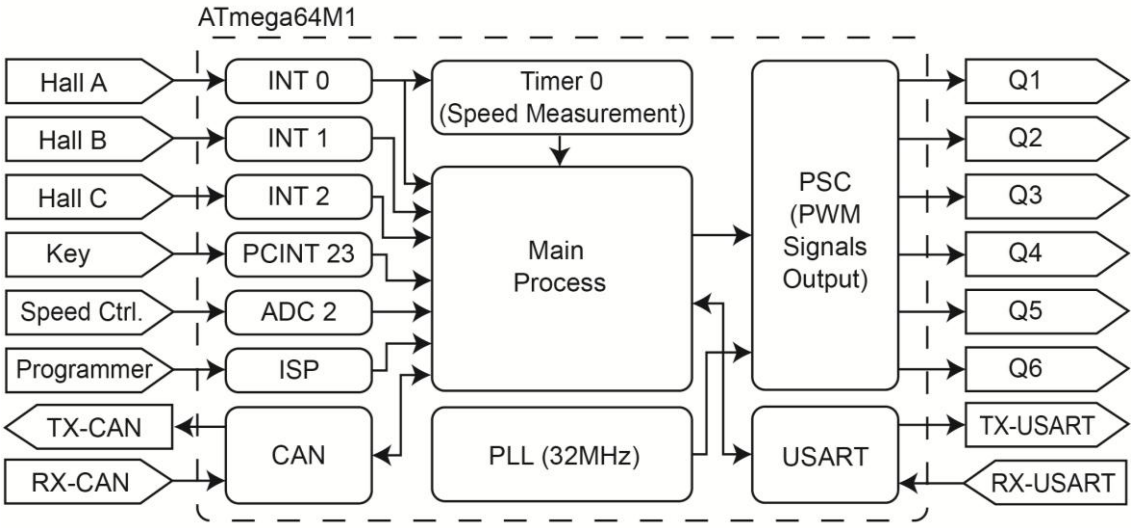


Figura 4-1: Diagrama de las partes utilizadas en el diseño de la controladora BLDCM

Por otra parte, se dispone de los módulos de comunicaciones CAN y UART (*Universal Asynchronous serial Receiver and Transmitter*) con sus respectivas señales de transmisión y recepción de datos.

4.2.1. Proceso Principal

En este proceso se realiza la inicialización de las variables y del hardware necesario del microcontrolador. Mediante la función *micro_modules_initialization()*, se realiza la configuración inicial de los puertos de entrada y salida del µC y se ejecutan las siguientes funciones:

- *adc_initialization()*: Configura y habilita el módulo ADC para la lectura del potenciómetro analógico de ajuste de velocidad por el canal ADC2.
- *timer1_initialization()*: Inicializa el temporizador 1 para realizar la tarea de muestreo de la velocidad y ajuste del ancho de pulso de la salida PWM. Se habilita la interrupción por comparación para que se ejecute cada 256 µs.
- *timer0_initialization()*: Configura el temporizador 0 como contador para realizar la medición de velocidad del motor. Habilita la interrupción por desbordamiento para convertir el temporizador 0 de 8 bits en uno de 16 bits. Esto se consigue gracias a una variable auxiliar que se incrementa cada vez que se ejecute esta interrupción. Con esta conversión se consigue una mayor precisión en la medición de la velocidad del motor.
- *uart_initialization()*: Configura el módulo UART del µC como transmisor y receptor. Para ello, utiliza los parámetros especificados por el usuario en el fichero de configuración.
- *start_pll_32mhz()* y *wait_pll_ready()*: Corresponde a las funciones encargadas de configurar y activar el PLL a una frecuencia de 32 MHz. Una vez activado, se mantiene a la espera de que el PLL esté preparado y se continúe ejecutando el programa.
- *psc_initialization()*: Realiza la configuración del módulo PSC según los parámetros detallados por el usuario en el fichero de configuración.
- *external_interrupt_initialization()*: Habilita las interrupciones externas utilizadas por los sensores Hall (INT1, 2 y 3). Además, habilita las interrupciones ejecutadas por cambio de pin para el interruptor de arranque del motor y para el interruptor de cambio de sentido de giro (en el caso de estar habilitada esta opción).

Una vez inicializado el sistema, se ejecuta un bucle infinito donde, cada 256 µs, se ejecuta la función *launch_adc().* Esta función es la encargada de realizar la lectura del potenciómetro externo de ajuste de velocidad. Este valor es almacenado y utilizado para establecer la velocidad de referencia del motor. Mediante esta velocidad de referencia, se ejecuta el bucle de regulación en lazo abierto para establecer el valor del ancho de pulso de la señal PWM.

Obtenido el valor del ancho de pulso, se actualizan los registros del módulo PSC con ese valor. Posteriormente, se envía el valor de la velocidad medida a través del módulo UART para su visualización en el software de monitorización (Figura 4-2).

4.2.1. Interrupciones de los Sensores Hall

Estas interrupciones son las encargadas de detectar los flancos de subida de las señales de los sensores Hall. Son las interrupciones con más alta prioridad de todo el programa. En cada una, la función *output_psc_switch_commutations(value)* es ejecutada. Esta función requiere como parámetro el valor en ese instante de los sensores Hall, correctamente formateado. Mediante ese valor y el sentido de giro establecido en ese momento, se realiza la activación de las salidas Q correspondientes del módulo PSC. Estas señales de salida PWM tendrán el ancho de pulso establecido por el potenciómetro de ajuste de velocidad y serán las encargadas de realizar la conmutación de las bobinas del motor BLDC.

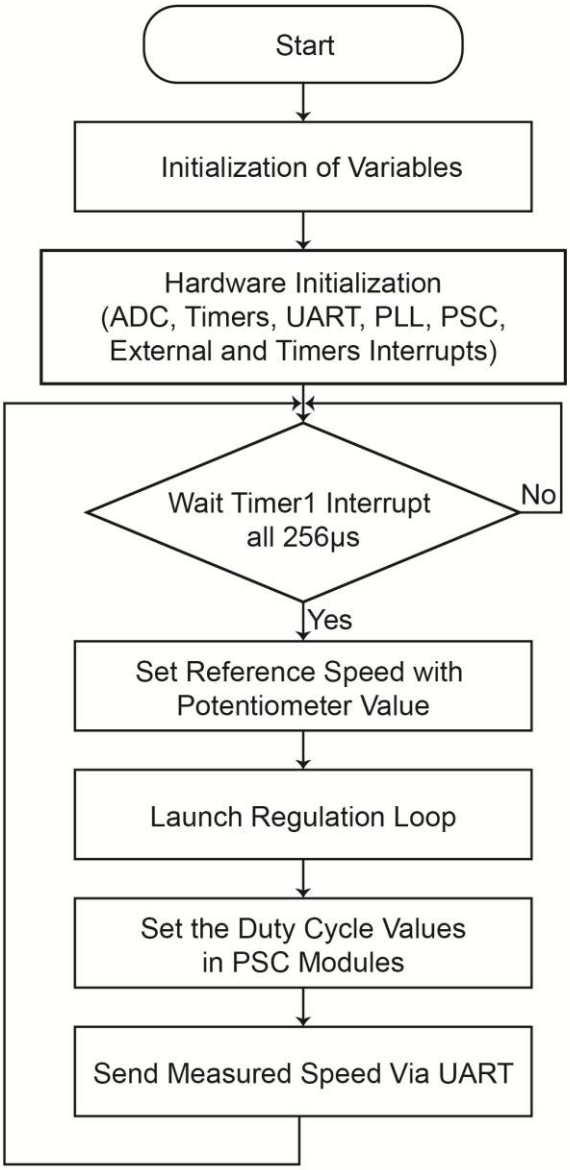


Figura 4-2: Diagrama de flujo del proceso principal

En la Figura 4-3 se muestra la secuencia de conmutación de las salidas del módulo PSC en función de las señales de los sensores Hall cuando el sentido de giro es CW (*ClockWise*). Los tres estados posibles en los que se pueden encontrar las bobinas U, V y W del motor son los siguientes:

- **VCC**: bobina conectada a la tensión de alimentación.
- **GND**: bobina conectara a tierra.
- **NC**: bobina no conectada.

En la Tabla 4-1 se detalla la secuencia de conmutación de las salidas PSC en relación con el valor de los sensores Hall y el sentido de giro del motor, CCW (*CounterClockWise*) o CW.

En el caso particular de la interrupción del sensor Hall A (INT 0), por cada flanco de subida se realiza el cálculo de las revoluciones por minuto del motor mediante la función *calculate_estimated_speed()*. Esta función utiliza el valor de 16 bits obtenido del temporizador 0 y una constante, definida por el usuario y que varía en función del motor utilizado, para calcular las revoluciones por minuto del motor (Figura 4-5) (Figura 4-4).

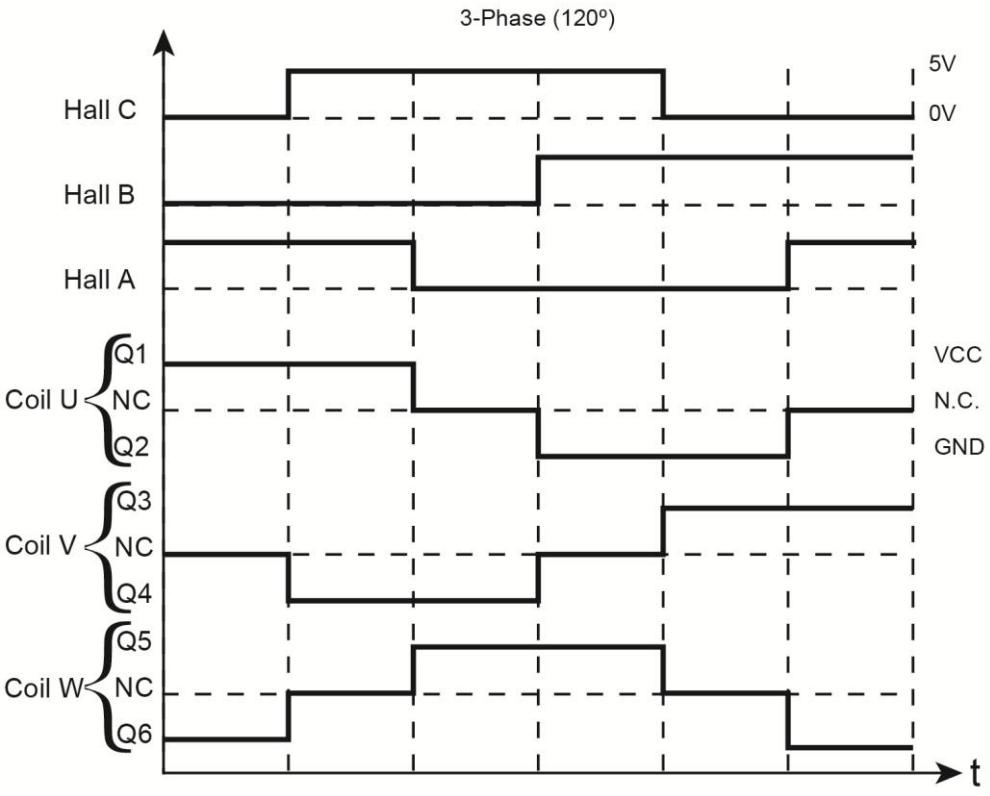


Figura 4-3: Señales de los sensores Hall para la rotación CW

TABLA 4-1: SECUENCIA DE CONMUTACIÓN DE LAS SALIDAS PSC

Hall Sensors Value (C,B,A)	PSC Outputs Active (CCW)	PSC Outputs Active (CW)
001	Q5 – Q2	Q1 – Q6
101	Q3 – Q2	Q1 – Q4
100	Q3 – Q6	Q5 – Q4
110	Q1 – Q6	Q5 – Q2
010	Q1 – Q4	Q3 – Q2
011	Q5 – Q4	Q3 – Q6

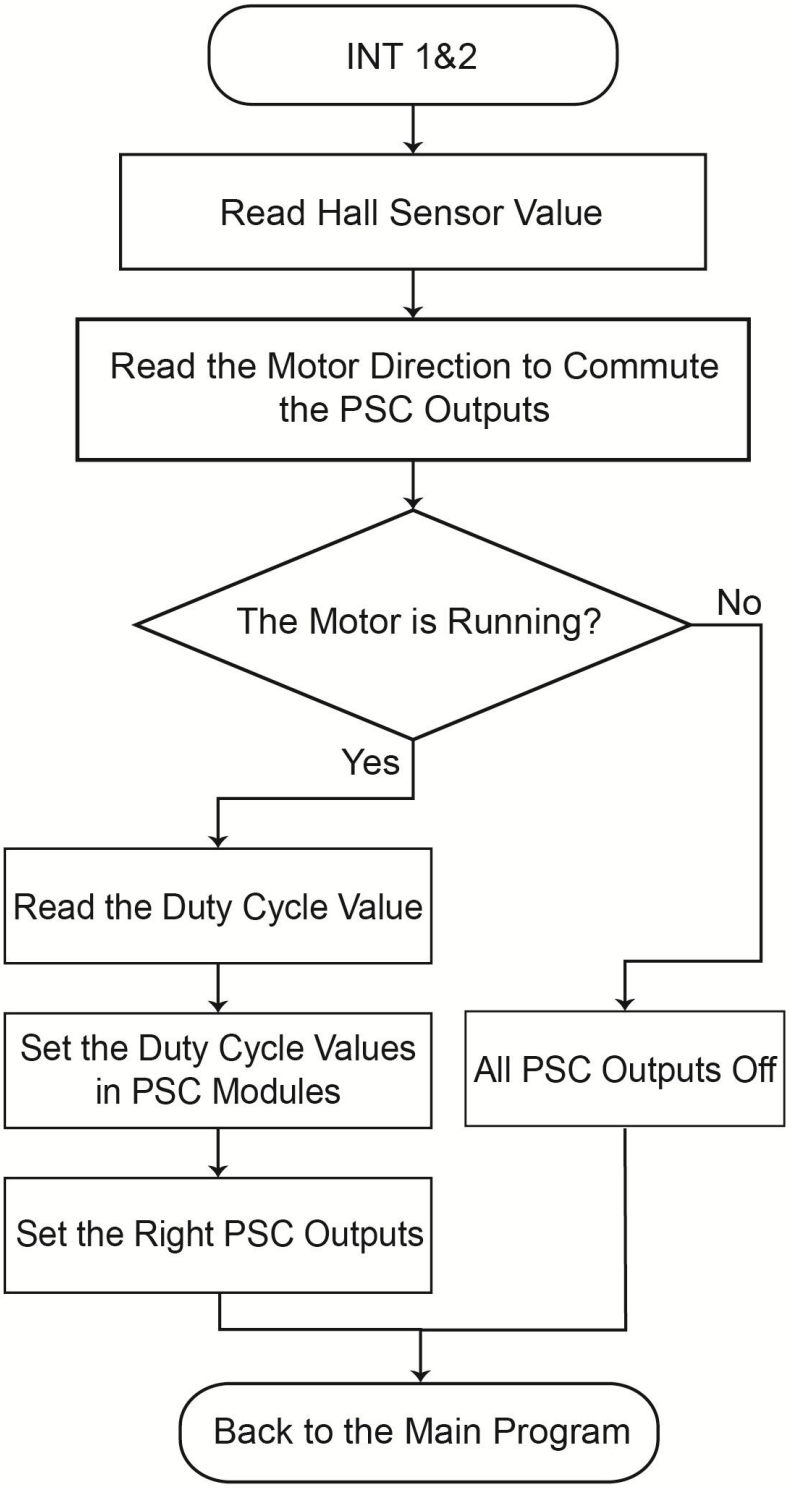


Figura 4-4: Diagrama de flujo de las interrupciones 1 y 2 de los sensores Hall

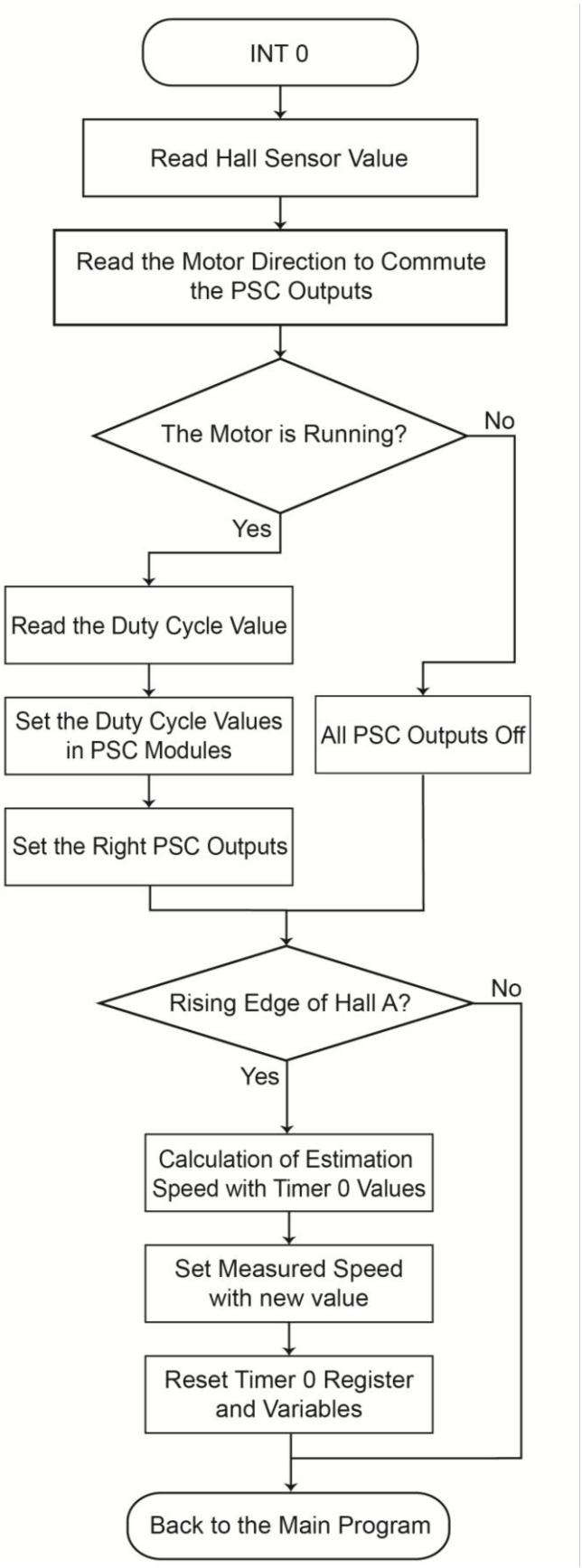


Figura 4-5: Diagrama de flujo de la interrupción 0 de los sensores Hall

4.2.1. Interrupción Convertidor Analógico/Digital (ADC)

Es la interrupción encargada de realizar la lectura del potenciómetro analógico externo de ajuste de velocidad y convertir su valor a un valor digital (Figura 4-6). Esta interrupción es ejecutada cuando se completa la conversión de la señal del potenciómetro conectada al canal 2 del ADC.

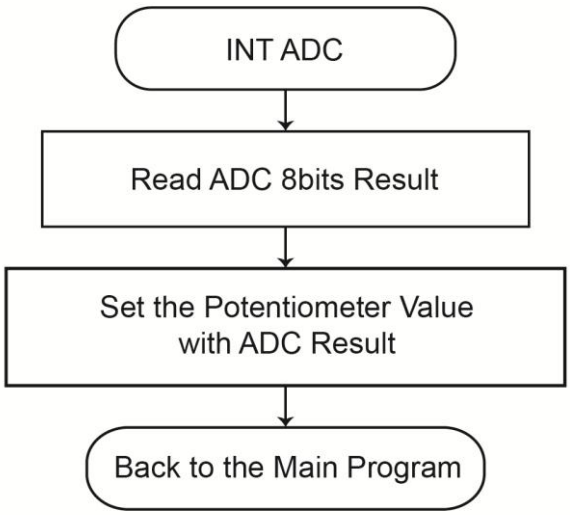


Figura 4-6: Diagrama de flujo de la interrupción ADC

4.2.2. Interrupción *Timer* 0 por desbordamiento

Como ya se ha explicado anteriormente, se encarga de incrementar la variable auxiliar que convierte el temporizador 0 de 8 bits en uno de 16 bits (Figura 4-7). En el caso de que se detecte que la variable auxiliar sea mayor de un cierto valor calculado, se resetea la variable y el valor de la velocidad de referencia. Si se detecta que la variable que indica si el motor está activo tiene un valor verdadero, se intenta reanudar la ejecución del motor. Para ello, se llama a la función *retry_run_motor()* que se encarga de lanzar el bucle de regulación, actualizar el valor del ancho de pulso y ejecutar la función de activación de las salidas del módulo PSC.

4.2.1. Interrupción por cambio de pin 1 (PCINT 1)

Es la interrupción encargada de controlar el arranque o la parada del motor. Cuando la interrupción detecta un cambio en la señal del interruptor de arranque, comprueba si la señal se encuentra a nivel alto o bajo. En el caso de que el nivel sea alto, se realiza la desactivación de todas las salidas del módulo PSC. Cuando la señal se encuentra a nivel bajo, se realiza una llamada a la función *output_psc_switch_commutations(value)* con el valor de los sensores Hall en ese instante (Figura 4-8).

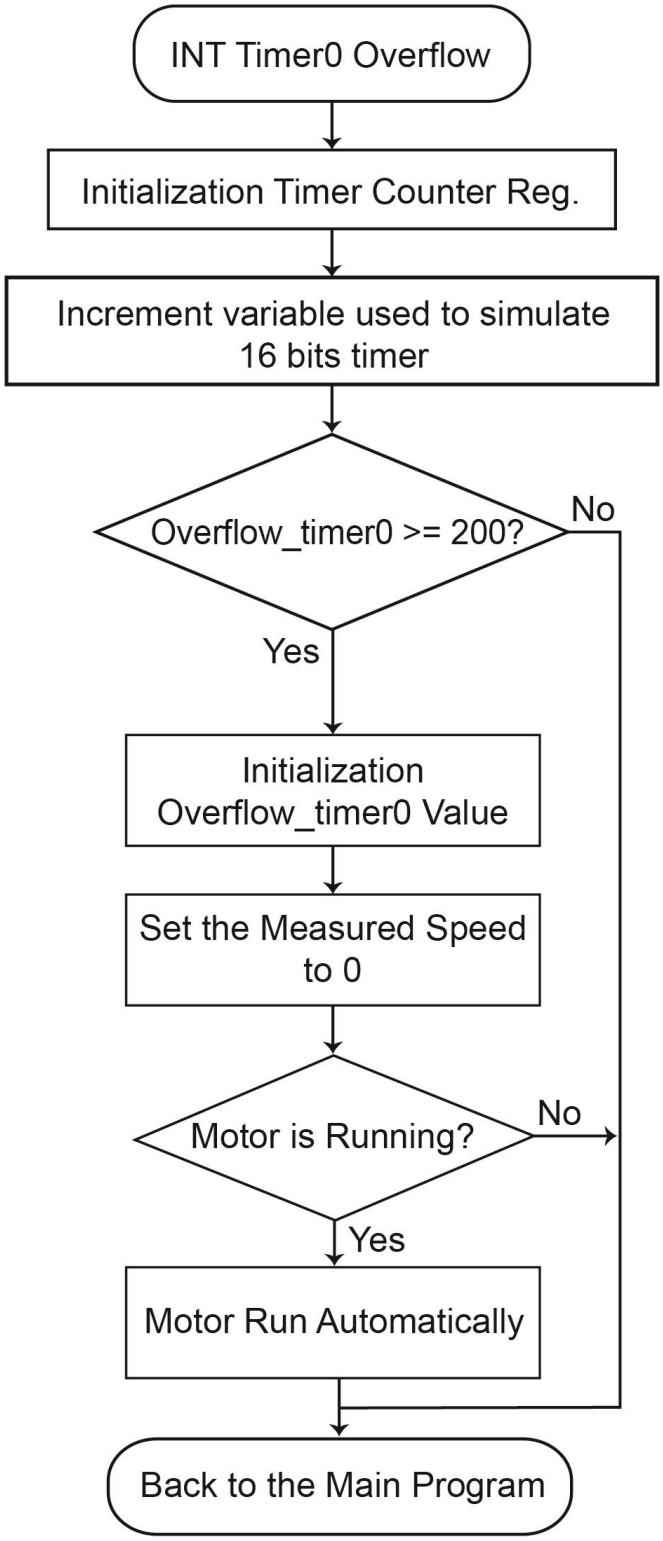


Figura 4-7: Diagrama de flujo de la interrupción del *Timer* 0 por *Overflow*

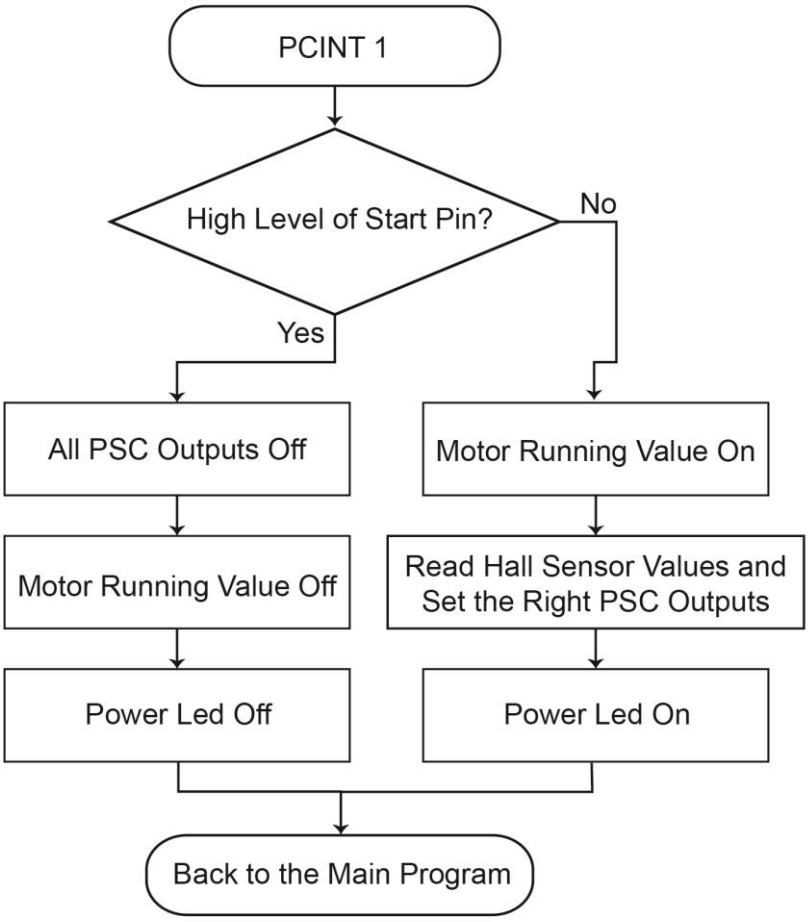


Figura 4-8: Diagrama de flujo de la interrupción arranque y parada del motor

4.2.1. Fichero de configuración

En este fichero se encuentran las constantes que el usuario debe configurar en función de sus necesidades. Para el módulo UART, ha de configurar el *baudrate*, el número de bits de stop y el número de bits a transmitir que utilizará en la conexión serie con el PC.

Otro aspecto que se debe configurar es la constante de velocidad del motor. Esta constante se calcula siguiendo la ecuación (1). Dependerá del tiempo, en segundos, configurado para el temporizador 0 (*t_timer0*), la velocidad máxima, en revoluciones por minuto, (*max_speed*) y el número de pares de polos del motor BLDC utilizado.

$$speed\_const = \frac{60 \cdot 255}{n \cdot t\_timer0_{(s)} \cdot \max\_speed_{(rpm)}} \quad (1)$$

Esta constante se utiliza para convertir la velocidad del motor medida a un dato acotado en un rango de 0 a 255 (8 bits). Este dato será el valor enviado al software de monitorización para visualizar la velocidad del motor.

4.3. Programación del ATmega64M1

En la Figura 4-9 se muestra el mapa de memoria *flash* del µC ATmega64M1 utilizado en el proyecto. El código de la aplicación ocupa un espacio de 10,8 kB. Se observa que existe un espacio libre en la memoria para las futuras ampliaciones del proyecto.

La programación del microcontrolador se realiza mediante el software Atmel Studio 6.0. Este software es de descarga gratuita desde la web del fabricante del µC y permite el desarrollo y compilación de códigos elaborados en C/C++ o lenguaje ensamblador.

A través de la interfaz ISP del µC y el módulo programador AVRISP mkII (Figura 4-10), se realiza la carga del programa en memoria.

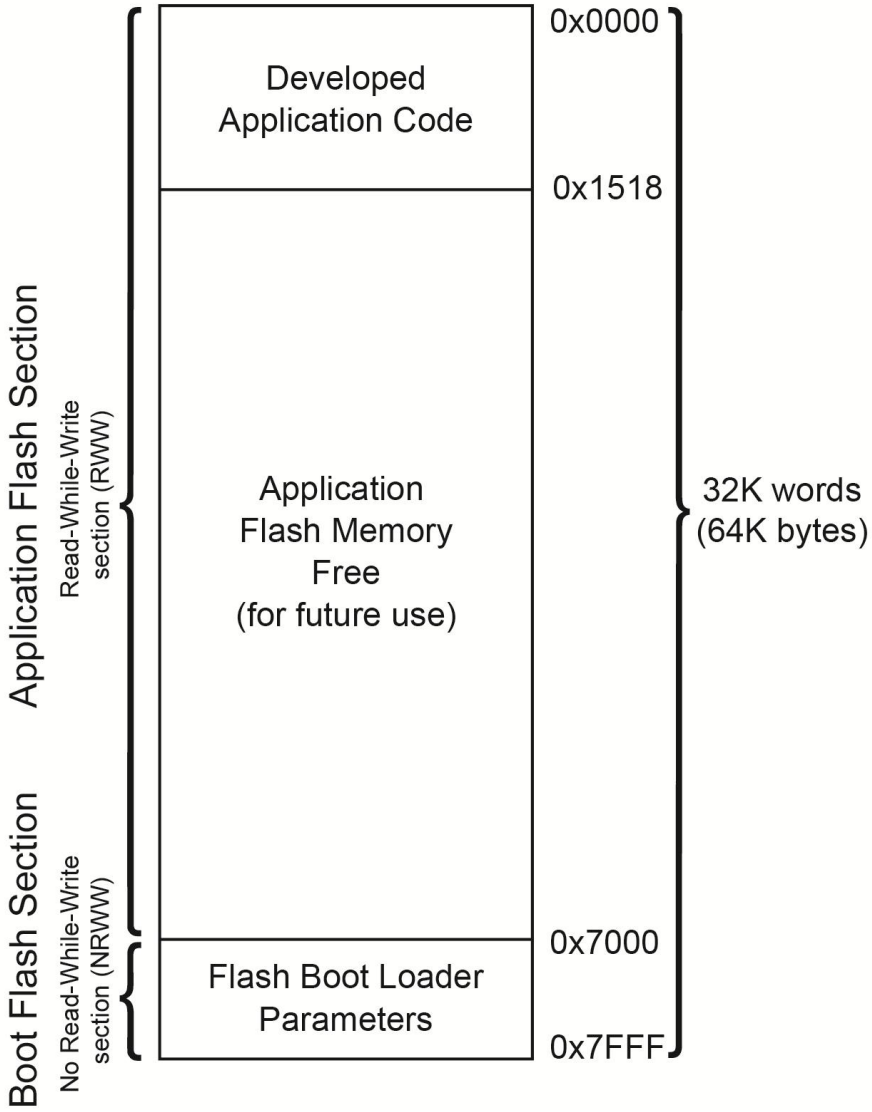


Figura 4-9: Mapa de memoria flash del µC ATmega64M1 utilizado

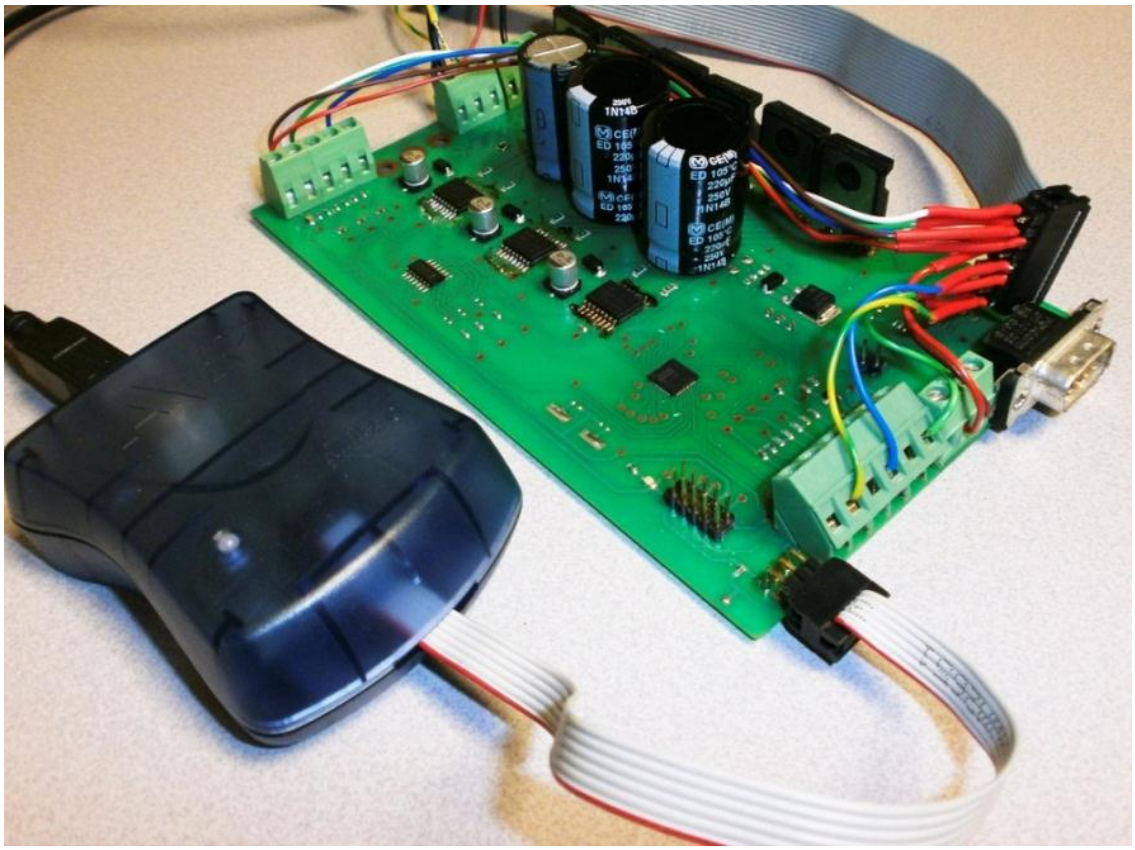


Figura 4-10: Vista de la controladora BLDC y el programador AVRISP mkII

5. Software de control y monitorización en LabVIEW

En este punto, se describe el software elaborado, utilizando NI LabVIEW [24], para realizar el control y monitorización del motor BLDC desde un PC. El protocolo de comunicaciones empleado es RS-232. El software se compone de dos partes: un panel de control y monitorización del motor BLDC, y un panel para la configuración de la conexión UART y de la constante de cálculo de velocidad en revoluciones por minuto. En el Anexo III se muestra el diagrama de bloques de la aplicación.

5.1. Control y monitorización

En este panel se encuentran los controles y el visualizador de las revoluciones por minuto del motor, además de un indicador para mostrar la velocidad, en kilómetros por hora, del prototipo desarrollado (Figura 5-1). Dispone de los controles de encendido y apagado (*ON/OFF*), cambio de sentido de giro (*CCW/CW*) y un deslizador para el ajuste de velocidad. Estos controles virtuales se han incorporado para dar la posibilidad de controlar el motor desde la aplicación. En el panel de configuración se dispone de un interruptor que habilita estos controles. Si este interruptor está activado, los controles físicos del banco de pruebas quedarán anulados.

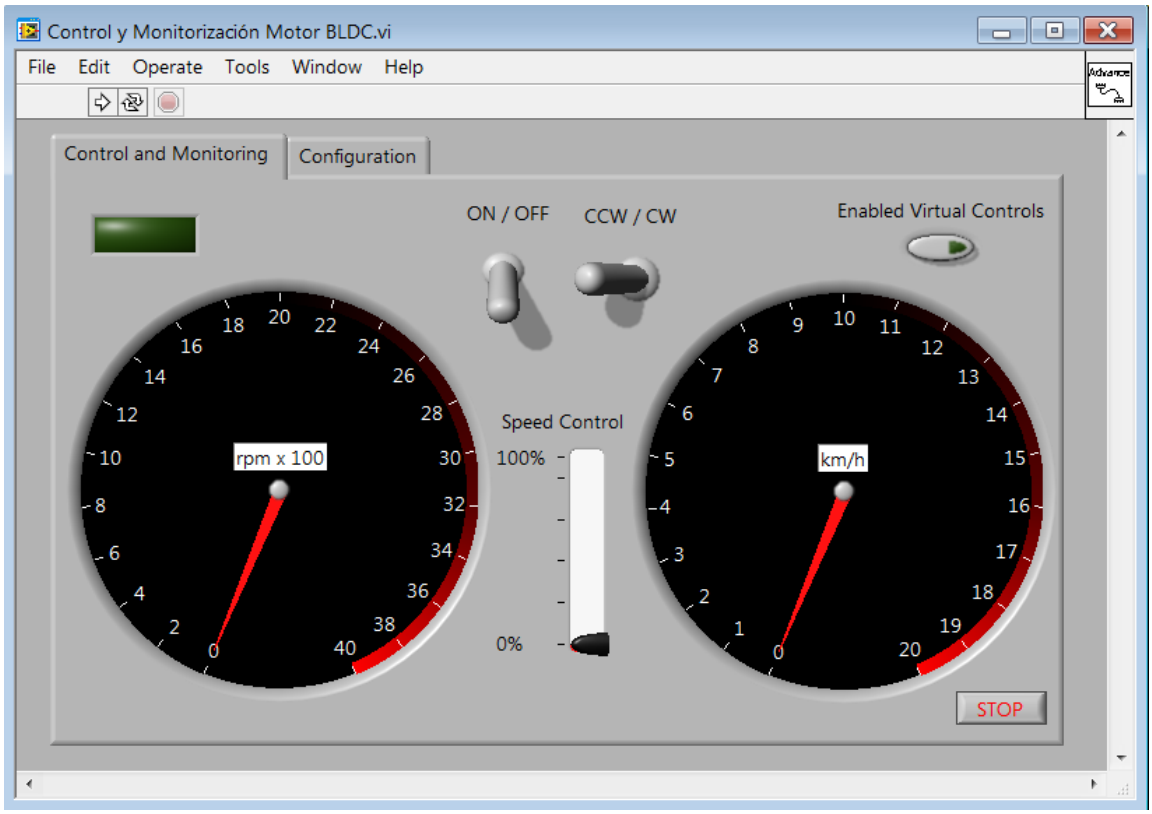


Figura 5-1: Vista del panel de control y monitorización del motor BLDC

5.2. Configuración UART

En este panel se muestran los controles para configurar la conexión del puerto serie del PC con el µC (Figura 5-2). Los parámetros a configurar de este protocolo son los siguientes:

- Selección del puerto serie usado en el PC.
- Selección del *baudrate* utilizado en la conexión.
- Número de bits por paquete en la conexión.
- Tipo de paridad establecida.
- Número de bits de stop utilizados.
- Control de flujo empleado en la conexión.
- Carácter de inicio de transmisión (*XON*).
- Carácter de parada de transmisión (*XOFF*).
- Carácter de fin de trama.
- Tiempo de espera máximo en milisegundos (*Timeout*).

Además, se dispone de un campo donde ha de introducirse la constante *alpha* utilizada para hallar el valor real de la velocidad del motor. Esta constante se calcula siguiendo la expresión (2), donde *n* es el número de pares de polos del motor BLDC, *speed_const* la constante calculada anteriormente y *t_timer0* el valor del temporizador 0 en segundos.

$$alpha = \frac{60}{n \cdot speed\_const \cdot t\_timer0_{(s)}} \tag{2}$$

La velocidad real del motor BLDC se calcula mediante la expresión (3), donde *alpha* es la constante hallada en la ecuación (2) y *measured_speed* es el valor de 8 bits de la velocidad medida y recibida por el puerto serie.

$$real\_speed_{(rpm)} = alpha \cdot measured\_speed \tag{3}$$

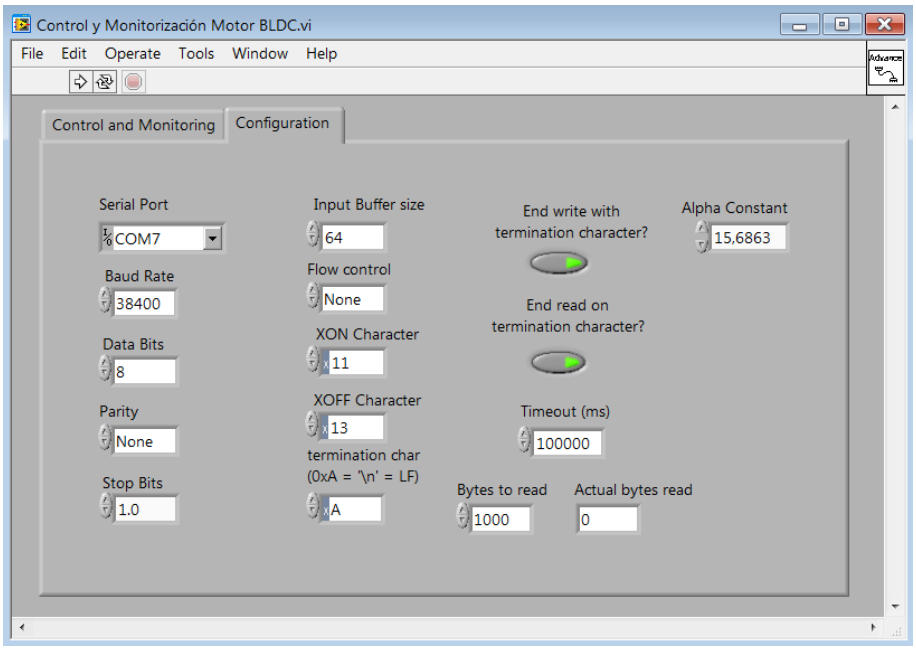


Figura 5-2: Vista del panel de configuración del software de control y monitorización

6. Sistema de Comunicaciones CAN

Como ya se comentó anteriormente, el protocolo de comunicaciones CAN ofrece una mayor seguridad y robustez ante interferencias y colisiones en la transmisión de los datos que el protocolo RS-232. Es por ello, por lo que se decide proponer una implementación con este protocolo mediante la cual se permita controlar y monitorizar el sistema de control de motores BLDC propuesto.

6.1. Interconexión del sistema

En la Figura 6-1 se muestra el diagrama de bloques del sistema y las interconexiones existentes entre ellos. El PC o *display* del sistema es el encargado de alojar la capa de aplicación CAN. Éste es el nodo encargado de controlar y supervisar el estado del motor y del pack de baterías a través de la controladora y el BMS, respectivamente.

La controladora BLDCM es el nodo encargado de comunicar la velocidad de giro del motor al nodo maestro. Además, recibirá las señales de control de arranque, cambio de sentido de giro y de ajuste de velocidad desde la aplicación alojada en el nodo maestro.

El BMS es el nodo encargado de comunicar a la aplicación el estado del pack de baterías. El BMS enviará un mensaje especial al nodo maestro cuando el nivel de voltaje de las baterías sea crítico, y éste deberá actuar en consecuencia.

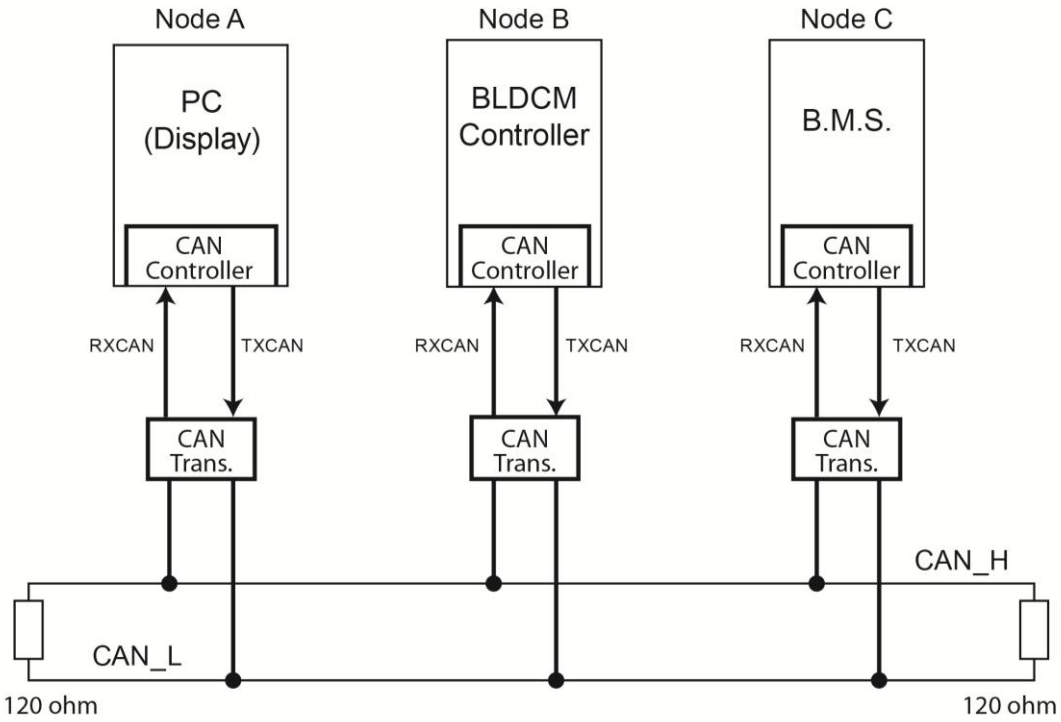


Figura 6-1: Esquema de interconexión del sistema de control mediante Bus CAN

TABLA 6-1: NODOS Y DISPOSITIVOS ASOCIADOS

Nodo	Dispositivo
A	PC o *display* (Unidad Central de control)
B	Controladora BLDCM
C	BMS

6.2. Formato de tramas

Puesto que el número de mensajes utilizados en el sistema de control es reducido, se utilizará la versión de protocolo CAN estándar, que corresponde al CAN original (versiones 1.0, 1.2 y 2.0A) con un identificador de 11 bits. En la Figura 6-2 se muestra el formato de la trama de datos y de la trama remota.

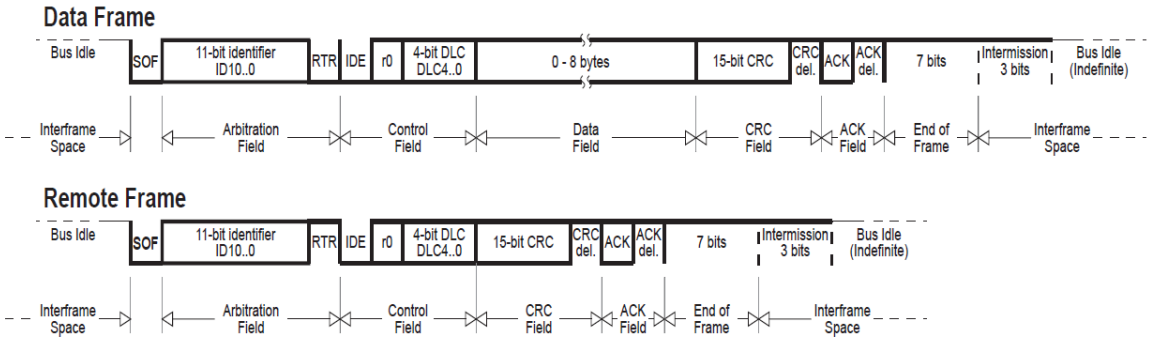


Figura 6-2: Formato de tramas CAN versión 2.0 A

6.3. Prioridad de mensajes

En la Tabla 6-2 se especifican los mensajes utilizados en las comunicaciones CAN del sistema de control, estableciendo los niveles de prioridad según el valor del identificador (ID).

TABLA 6-2: NIVELES DE PRIORIDAD ENTRE LOS MENSAJES DE LOS NODOS

Nodo Origen	Nodo Destino	Prioridad	ID (Hex)	Mensaje
PC	Controladora BLDCM	1	0x530	Parada/arranque del Sistema
PC	Controladora BLDCM	2	0x531	Cambio del sentido de giro del motor
BMS	PC	3	0x532	Voltaje de batería mínimo
PC	Controladora BLDCM	4	0x533	Valor deseado de velocidad del motor
Controladora BLDCM	PC	5	0x534	Valor actual de velocidad del motor
BMS	PC	6	0x535	Valor de tensión del pack de baterías

6.4. Implementación de las funciones C

Las siguientes funciones que se presentan, permiten la configuración y utilización del protocolo CAN para el microcontrolador ATmega64M1 en la controladora BLDCM. La función de inicialización del protocolo (*can_initialization()*) establece un *baudrate* de 125 kbps utilizando una frecuencia de reloj de 8 MHz y el formato de trama CAN 2.0 A.

```
void can_initialization(void){

	// Software reset
	CANGCON = ( 1 << SWRES );
	// CAN timing prescaler set to 0
	CANTCON = 0x00;

	// Set baud rate to 125kb with FclkIO = 8Mhz
	CANBT1 = 0x0E;
	CANBT2 = 0x04;
	CANBT3 = 0x13;

	//ATmega64M1 has 6 Message Object (MOb)
	for ( int8_t mob=0; mob<6; mob++ )
	{
		// Selects Message Object 0-5
		CANPAGE = ( mob << 4 );
		// Disable MOb
		CANCDMOB = 0x00;
		// Clear MOb status register
		CANSTMOB = 0x00;
	}
	// Select MOb1
	CANPAGE = ( 1 << MOBNB0 );
	// Enable interrupts on MOb1
	CANIE2 = ( 1 << IEMOB1 );
	// Enable interrupts on receive
	CANGIE = ( 1 << ENIT ) | ( 1 << ENRX );

	// Clear Mask, let all IDs pass
	CANIDM1 = 0x00;
	CANIDM2 = 0x00;
	CANIDM3 = 0x00;
	CANIDM4 = 0x00;

	// CONMOBn = Enable Reception - IDE = 11 bits identifier CAN 2.0 A
	// (MAX IDTAG = 0x7FF)
	// DLCn = 1 Byte in the data field of the message
	CANCDMOB = (1 << CONMOB1) | (0 << CONMOB0) | (0 << IDE) | (1 << DLC0);

	// Enable mode:
	// The CAN channel enters in enable mode once 11 recessive bits has
	been read
	CANGCON |= ( 1 << ENASTB );

	// General Interrupts Enable
	sei();
}
```

La función *can_tx_speed(id_tag, can_message)* permite la transmisión de un mensaje CAN de 1 byte con el identificador enviado por parámetros. En el caso de la controladora BLDCM se utiliza para el envío de la velocidad actual del motor.

```
void can_tx_speed(int id_tag, int8_t can_message)
{

	// Select MOb1 for transmission
	CANPAGE = ( 1 << MOBNB0 );

	// Wait for MOb1 to be free
	while ( CANEN2 & ( 1 << ENMOB1 ) );

	// Clear mob status register
	CANSTMOB = 0x00;

	// Set CAN Identifier Tag Registers
	CANIDT4 = 0x00;
	CANIDT3 = 0x00;

	// Shift Left id_tag 5 bits
	CANIDT2 = (5 << id_tag);
	// Shift Right id_tag 3 bits
	CANIDT1 = (3 >> id_tag);

	// Set message data with the data to transmit
	CANMSG = can_message;

	// CONMOBn = Enable Transmission - IDE = 11 bits identifier CAN 2.0 A
	// (MAX IDTAG = 0x7FF)
	// DLCn = 1 Byte in the data field of the message
	CANCDMOB = (0 << CONMOB1) | (1 << CONMOB0) | (0 << IDE) | (1 << DLC0);

	// Wait for Transmission Complete (TXOK flag set)
	while ( ! ( CANSTMOB & ( 1 << TXOK ) ) );

	// Disable Transmission
	CANCDMOB = 0x00;
	// Clear TXOK flag
	CANSTMOB = 0x00;
}
```

Por último, tras haber habilitado la interrupción CAN por recepción del microcontrolador, se realiza en ella la recepción de los mensajes CAN destinados a la controladora BLDCM. Según el identificador de mensaje recibido se realizan las siguientes acciones:

- Arranque o parada del motor.
- Cambio del sentido de giro del motor.
- Ajuste de la velocidad del motor.

```
//***** CAN Reception Interrupt***************//
ISR ( CAN_INT_vect )
{
       int8_t  length, savecanpage;
       int    id_tag;

       // Save current MOb
       savecanpage = CANPAGE;

       // Selects MOb with highest priority interrupt
       CANPAGE = CANHPMOB & 0xF0;

       // Interrupt caused by receive completed
       if ( CANSTMOB & ( 1 << RXOK) )
       {
              //Get the ID Tag of the message
              id_tag = (3 << CANIDT1)+(5 >> CANIDT2);

              // DLC, number of bytes to be received
              length = ( CANCDMOB & 0x0F );

              for ( int8_t i = 0; i < length; i++ )
              {
                     // Get message in data array
                     data[i] = CANMSG;
              }

              // CONMOBn=Enable Reception - IDE = 11 bits identifier CAN 2.0 A
              // (MAX IDTAG = 0x7FF)
              // DLCn=Number of bytes to be received in the data field
              CANCDMOB = (1 << CONMOB1) | (0 << CONMOB0) | (0 << IDE) |
                         (length << DLC0);

          // Operations to be performed according to the ID Tag
          switch(id_tag)
          {
                 case START_STOP_MOTOR:

                     // Star/Stop the Motor conditions and functions
                     break;

                 case MOTOR_ROTATION_DIRECTION:

                     // Motor Rotation Direction conditions and functions
                     break;

                 case REMOTE_MOTOR_SPEED:

                     // Motor Speed Adjust functions
                     break;

              default:
                     break;
       }

       // Clear RXOK flag
       CANSTMOB = 0x00;
       // Restore original MOb
       CANPAGE = savecanpage;
}
```

6.5. Software de control y monitorización CAN

La empresa National Instruments, proveedora del entorno de desarrollo LabVIEW, dispone de diversos productos hardware y software que actúan como interfaz entre aplicaciones programadas en LabVIEW y un bus de comunicaciones CAN. Entre los productos software se encuentra el NI-CAN. Este módulo de desarrollo ofrece la posibilidad de realizar un software en LabVIEW que permita la recepción o transmisión de mensajes de un PC conectado a un bus CAN.

Como línea futura de desarrollo, se propone la modificación del software de control y monitorización que se ha basado en el protocolo RS-232 para que realice la comunicación mediante un bus CAN. Las funciones propuestas del protocolo CAN serían implementadas en el microcontrolador de la controladora BLDCM, y una adaptación de ellas se programarían en el microcontrolador de la BMS. Los niveles de prioridad de los mensajes, según sus identificadores, para una correcta resolución de los conflictos en el bus CAN, serían establecidos tanto en el código de los microcontroladores como en el software desarrollado en LabVIEW. Con este software y el sistema de comunicaciones propuesto para interconectar la controladora BLDCM, la BMS y un PC, se obtendría un sistema de control de motores BLDC para vehículos eléctricos basado en una comunicación robusta y fiable.

7. Resultados Obtenidos

A partir del diseño de esta controladora BLDCM y su librería de funciones para el control de motores BLDC, se ha desarrollado un prototipo para comprobar el correcto funcionamiento de todo el sistema. Además se ha incorporado al sistema una BMS digital desarrollada en otro proyecto [25] con la que gestionar el pack de baterías de ión-litio utilizado. Por otro lado, se han propuesto otras dos aplicaciones en la que sería factible utilizar tanto la controladora BLDCM y su librería como la BMS.

7.1. Sistema de control de motores BLDC de 3 fases

Este sistema de control de motores BLDC se ha diseñado y fabricado con el objeto de utilizarse para comprobar el correcto funcionamiento de la controladora BLDCM y la librería C elaborada (Figura 7-1). Para ello, se ha construido un prototipo de pruebas donde se aloja un motor BLDC conectado mecánicamente a un eje a través de una correa de transmisión (Figura 7-2). En un extremo del prototipo se encuentran los controles externos de la controladora (interruptor de arranque, interruptor de cambio de sentido de giro del motor y potenciómetro de ajuste de velocidad) (Figura 7-3).

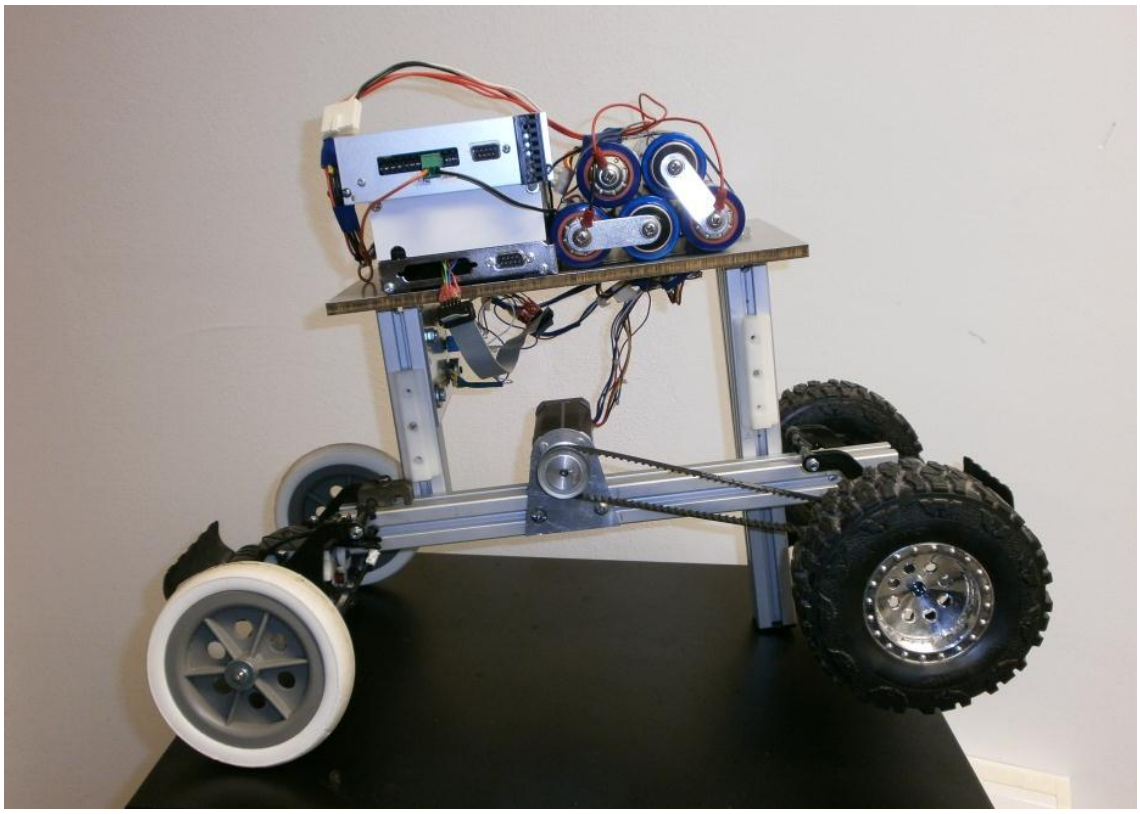


Figura 7-1: Vista del sistema de control de motores BLDC de 3 fases fabricado

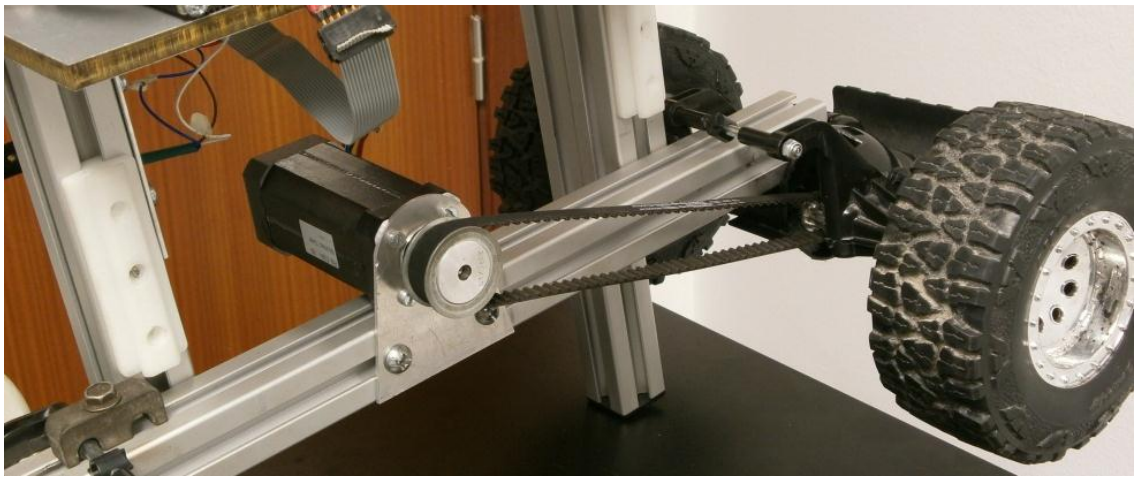


Figura 7-2: Motor BLDC y correa de transmisión

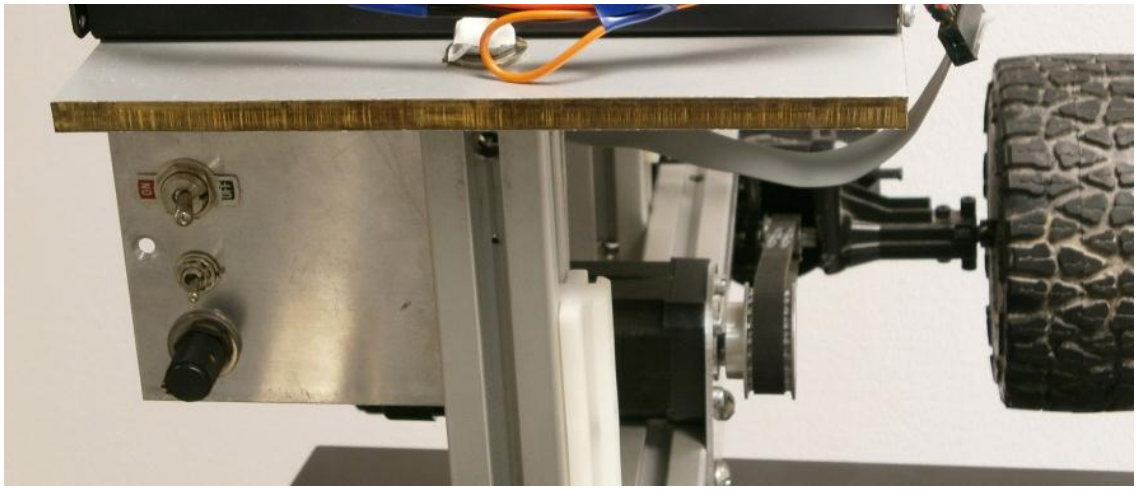


Figura 7-3: Controles externos del prototipo

En la Figura 7-4 se muestra el sistema de interconexión de los elementos que forman el sistema de control para motores BLDC de 3 fases. Este motor envía las 3 señales de los sensores Hall a la controladora y ésta, a su vez, envía las señales de conmutación de las bobinas al motor. Estas señales de conmutación se envían con un ancho de pulso determinado por el valor del potenciómetro de ajuste de velocidad conectado a la controladora. Además, se dispone de un interruptor de arranque que pone en marcha el motor y otro que permite cambiar el sentido de giro del mismo.

La controladora y el motor son alimentados con una tensión de 24 V suministrada por un pack de baterías Li-Ion que es gestionado a través de una BMS. Esta BMS digital, que forma parte de otro proyecto desarrollado por el IUMA, se encarga de monitorizar el estado de un pack de baterías de 7 celdas de 3,6 V cada una, garantizando la máxima durabilidad del mismo. Además de esto, permite realizar una carga segura de las baterías a partir de un cargador conectado a la red. En el caso del prototipo desarrollado se ha utilizado un pack de baterías de 5 celdas de 3,6 V cada una, obteniéndose una tensión final de 18 V.

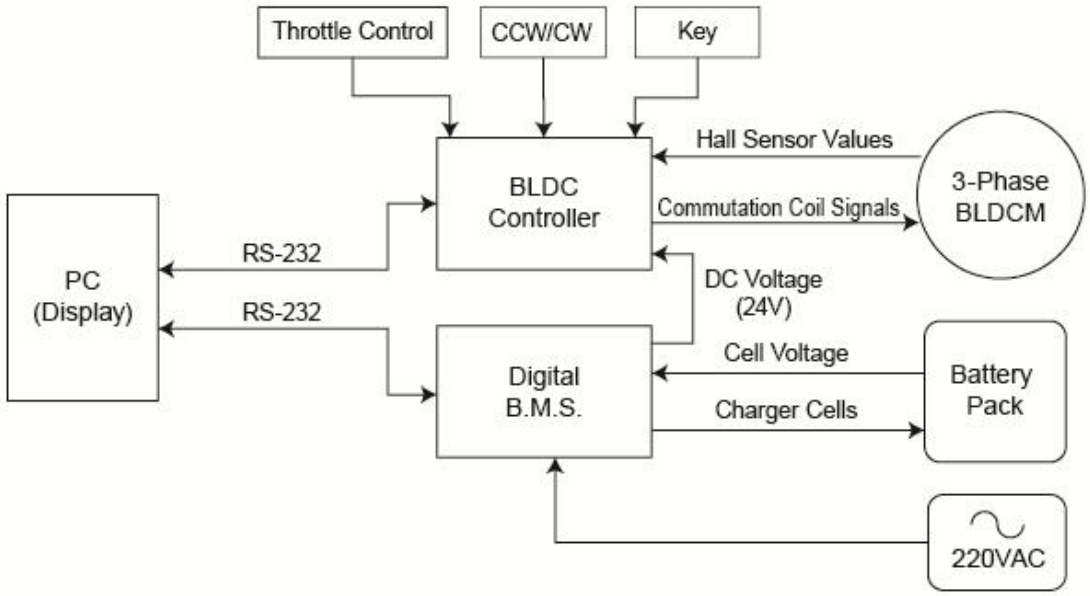


Figura 7-4: Configuración de un sistema de control de motores BLDC de 3 fases

La controladora y la BMS están comunicadas con una estación de trabajo (PC) vía serial mediante el protocolo RS-232. Esta comunicación, junto con el programa realizado en LabView expuesto anteriormente y un software desarrollado en Visual Basic, permiten monitorizar, en todo momento, la velocidad y estado del motor, así como el estado de cada una de las celdas del pack de baterías.

El motor BLDC empleado en el prototipo es el 42BL100, cuyas características principales se muestran en la Tabla 7-1.

TABLA 7-1: DATOS DEL MOTOR BLDC 42BL100

Nº of Pole	8
Nº of Phase	3
Rated Voltage (V)	24
Rated Speed (rpm)	4000
Rated Torque (Nm)	0,25
Max Peak Torque (Nm)	0,75
Torque Constant (Nm/A)	0,036
Max Peak Current (A)	20
Mass (kg)	0,8

7.2. Sistema de control para motores BLDC de 6 fases

En este apartado se presenta una configuración con la que realizar un sistema de gestión para un motor BLDC de 6 fases de 5 kW (Figura 7-5). Este tipo de motor BLDC es empleado para la propulsión de motos eléctricas y se gestiona mediante dos controladoras independientes pero sincronizadas entre sí a través de los controles externos (potenciómetro

de ajuste de velocidad e interruptor de encendido) y los sensores Hall provenientes del motor (Figura 7-6). Cada controladora se encarga de excitar 3 de las bobinas del motor siguiendo el orden preestablecido por el software y sus señales correspondientes de los sensores Hall.

La alimentación de la electrónica y del motor se realiza, al igual que en el sistema anterior, con un pack de baterías, pero esta vez de 96 V (26 celdas de 3,6 V). Igualmente, el pack está gestionado por una BMS y, tanto ésta como las 2 controladoras, se conectan a un Bus CAN. Este protocolo es el encargado de gestionar la comunicación de las placas con un *display* que muestra las variables de estado del motor y de las baterías. Por otro lado, existe la posibilidad de conectar al bus un PC para testear y monitorizar el sistema.

Figura 7-5: Motor BLDC de 6 fases para motos eléctricas

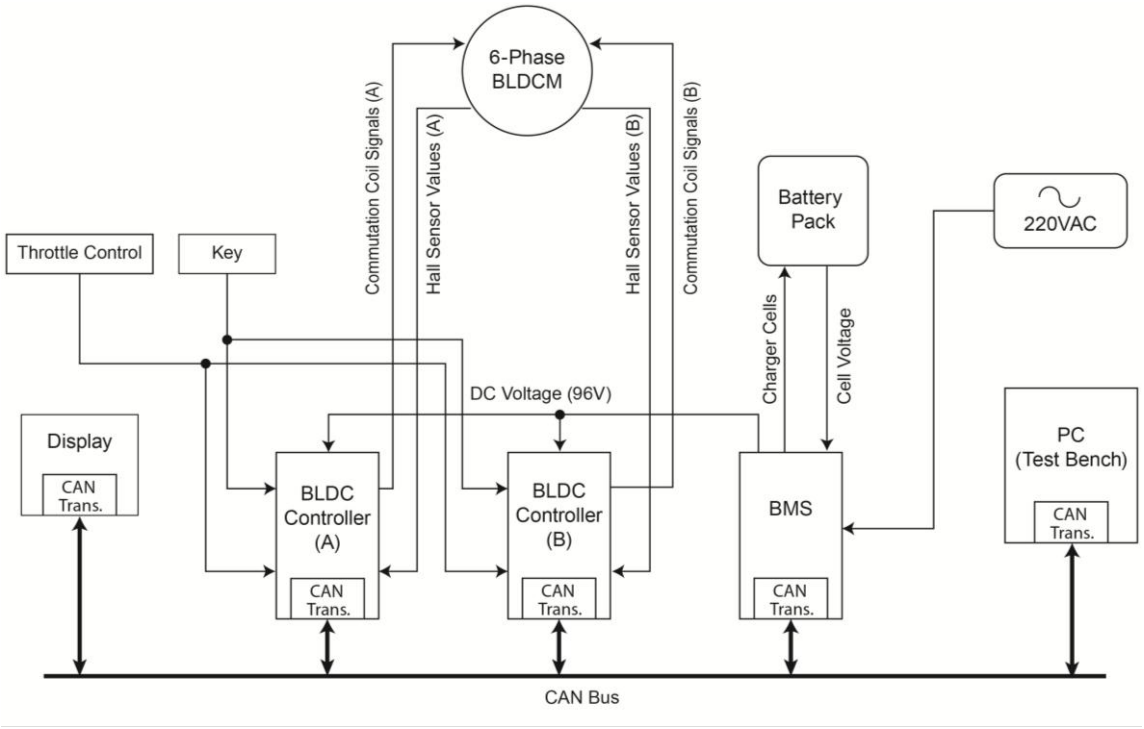


Figura 7-6: Configuración del sistema de control de un motor BLDC de 6 fases

7.3. Sistema de control para ROVs

Actualmente se está desarrollando en la ULPGC un proyecto relacionado con los ROVs (*Remotely Operated Vehicles*) y AVs (*Autonomous Vehicles*), el proyecto AVORA (*Autonomous Vehicle for Operation and Research in Aquatic environments*) [26]. Este proyecto está siendo llevado a cabo por un equipo de estudiantes e investigadores de esta universidad en cooperación con PLOCAN (Plataforma Oceánica de Canarias), cuyo objetivo es el de diseñar y fabricar la electrónica, mecánica y software de un vehículo submarino autónomo. Con él participarán en la competición europea SAUC-E 14 (*Student Autonomous Underwater Vehicle Challenge – Europe*), una competición de referencia en el campo de la robótica submarina, que desde 2010 se celebra anualmente en el CMRE (*Center for Maritime Research and Experimentation*) del Nato Underwater Research Center, en La Spezia, Italia.

El sistema de propulsión que utiliza el AV AVORA consta de cuatro motores *thruster* DC con escobillas. El sistema de control de éstos está basado en dos placas de control (*RoboClaw Boards*). Cada una se encarga de controlar dos motores. Además, una placa Arduino Mega se encarga de controlar las placas *RoboClaw* y comunicarse vía *ethernet* con la CPU (*Central Processing Unit*). Esta CPU se conecta mediante WIFI a la estación de control para recibir las órdenes necesarias cuando el robot está en la superficie (Figura 7-7). Una vez sumergido, éste es completamente autónomo.

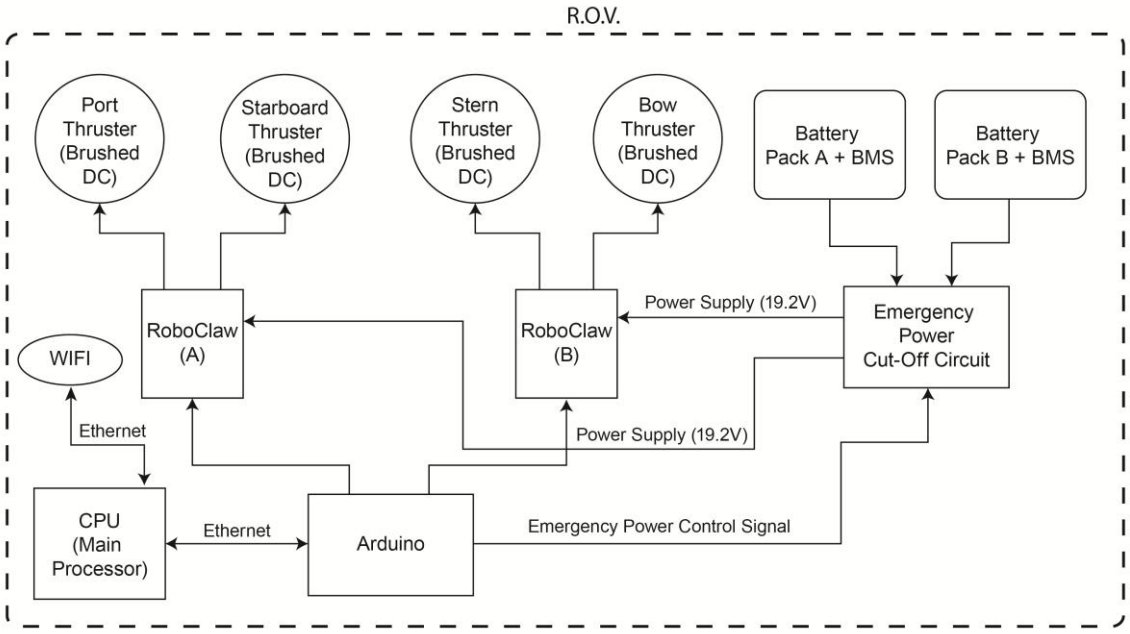


Figura 7-7: Configuración del sistema de control actual del AV AVORA

Una vía futura dentro del proyecto AVORA es la de sustituir el sistema de propulsión actual por un sistema basado en motores BLDC. Con ello, se conseguiría una mayor eficiencia del vehículo y una reducción de tamaño y peso del mismo. Es por esto por lo que se desarrolla el prototipo de un sistema de control de ROVs con motores BLDC basado en la controladora desarrollada en este proyecto.

Este sistema se basa en 4 motores BLDC de 3 fases totalmente independientes (Figura 7-8). Cada motor dispone de una controladora independiente asociada y conectada a un bus CAN. Así mismo, al igual que en las configuraciones anteriores, el sistema dispone de un pack de baterías que suministra la tensión necesaria para el funcionamiento de los motores. Este pack está gestionado por una BMS también conectada a un bus CAN.

Al igual que en el sistema de control actual, se utiliza una CPU para gestionar el sistema. La CPU estará integrada en el vehículo y recibirá las órdenes de configuración vía WIFI cuando esté en la superficie. Además, se podría implementar la posibilidad de controlar el ROV en tiempo real mediante un mando de control o un PC. Este controlador remoto estará comunicado con el vehículo gracias a un cable umbilical que transmitirá los datos utilizando el protocolo CAN. La velocidad de transmisión de datos podrá variar entre 1 Mbps y 10 kbps en función de la longitud del cable utilizado.

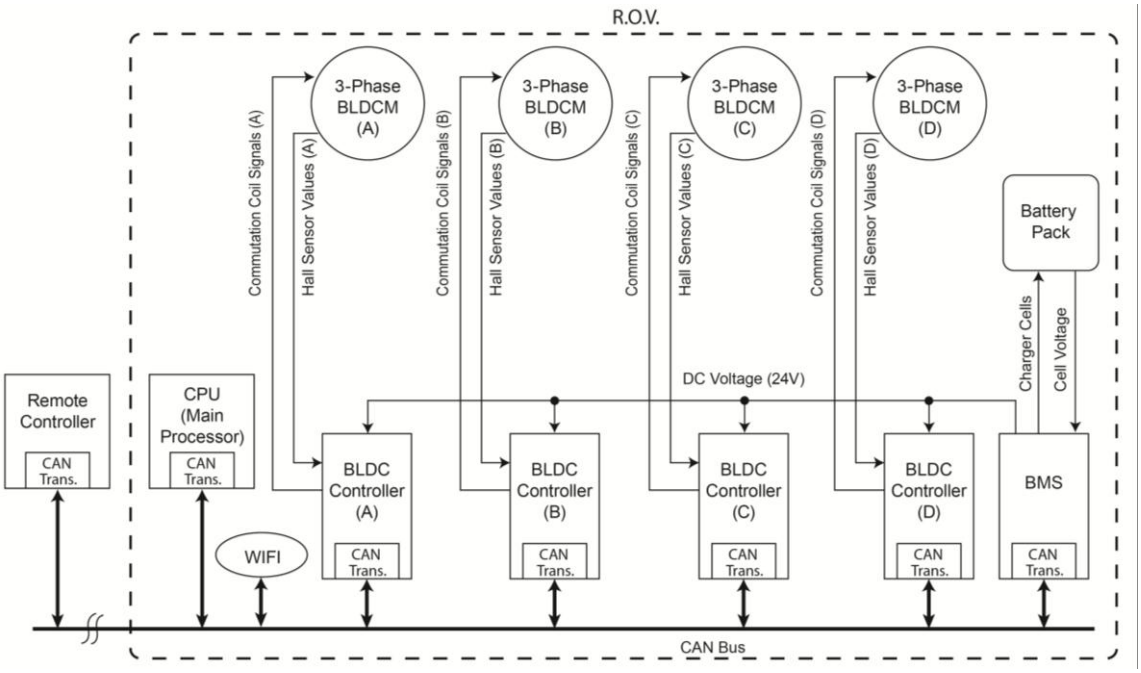


Figura 7-8: Sistema de control para ROVs con motores BLDC

8. Conclusiones y Trabajos Futuros

Este proyecto se ha basado en la experiencia previa adquirida en el laboratorio SFP del IUMA, tras la realización de una beca de colaboración en un proyecto de investigación concedida por el MEC y la realización de las prácticas en empresa correspondientes al Itinerario de Adaptación al Grado en Ingeniería en Tecnologías de la Telecomunicación.

Gracias a la elaboración de este proyecto, se han redactado dos artículos que han sido presentados el pasado mes de Junio en el XI Congreso de Tecnologías, Aprendizaje y Enseñanza de la Electrónica (TAEE 2014) [27]. Uno de ellos detalla el diseño y la fabricación de la controladora de motores BLDC y su posible aplicación en la docencia de los sistemas de control para este tipo de motores [29]. El otro artículo presenta la arquitectura empleada para el desarrollo de la librería C para la programación del microcontrolador ATmega64M1 [28]. Además, se ha colaborado en el desarrollo de otro artículo presentado en el congreso relacionado con el diseño de una BMS analógica reconfigurable [30][31]. Por otra parte, estos artículos han sido traducidos al inglés, lo que ha permitido su indexación con *IEEE Xplore*.

Tras el montaje del prototipo y sus pruebas se ha demostrado el correcto funcionamiento tanto de la controladora y la librería C desarrollada, como del sistema completo con la BMS incorporada. Este desarrollo presenta un punto de partida para futuras ampliaciones de este proyecto o nuevas aplicaciones relacionadas con los motores BLDC.

Una línea de trabajo futuro relacionado con la librería C es la de implementar las funciones en el código para el control del motor BLDC en modo *Current-Loop* y *Speed-Loop*. Además, otra ampliación sería implementar y probar las funciones para utilizar el protocolo de comunicaciones CAN para el control y monitorización del motor en el microcontrolador, así como desarrollar la capa de aplicación utilizando *CANOpen* o el entorno de desarrollo de LabVIEW.

Respecto a la controladora BLDCM, un futuro proyecto sería llevar a cabo la implementación del sistema de control de ROVs con motores BLDC. Posteriormente, se realizaría el rediseño de la PCB de la controladora para integrar las cuatro controladoras necesarias en una sola placa de control para los cuatro motores. Así se obtendría una reducción del tamaño de las mismas y, en consecuencia, una disminución del tamaño y del peso del vehículo.

9. Listado de Acrónimos

ADC	Analog to Digital Converter
ASIC	Application-Specific Integrated Circuit
AVORA	Autonomous Vehicle for Operation and Research in Aquatic environments
BLDCM	Brushless Direct Current Motor
BMS	Battery Management System
CAN	Controller Area Network
CCW	Counter Clock Wise
CiA	CAN in Automation
CMRE	Center for Maritime Research and Experimentation
CPU	Central Processing Unit
CSMA/CD+CR	Carrier Sense Multiple Access with Collision Detection and Collision Resolution
CW	Clock Wise
DC	Direct Current
DIP	Dual In-line Package
D-Q	Direct-Quadrature
ECU	Engine Control Unit
FOC	Field Oriented Control
FPGA	Field Programmable Gate Array
HLP	Higher Layer Protocol
ISO	International Organization for Standarization
ISP	In-System Programming
NRZ	Non Return to Zero
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
OSI	Open Systems Interconnection
PCB	Printed Circuit Board
PLL	Phase-Locked Loop
PLOCAN	Plataforma Oceánica de Canarias
PSC	Power Stage Controller
PWM	Pulse Width Modulation
ROV	Remotely Operated Vehicle
RS-232	Recommended Standard 232
UART	Universal Asynchronous Receiver-Transmitter
SAUC-E	Student Autonomous Underwater Vehicle Challenge – Europe
SLIOS	Serial Linked Input/Output
SMD	Surface Mount Device
SRAM	Static Random Access Memory
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

10. Presupuesto

En este apartado detallamos el presupuesto requerido para la elaboración de este proyecto. Se han tenido en cuenta los componentes y materiales empleados para la fabricación de la controladora BLDC, los materiales empleados en la fabricación del prototipo y la mano de obra correspondiente a las distintas fases del proyecto.

10.1. Materiales

TABLA 10-1: PRESUPUESTO DE MATERIALES DE LA CONTROLADORA BLDC

Componente	Unidades	Precio Ud. (€)	Precio Total (€)
Resistencia 0603	32	0,080	2,560
Resistencia 1206	6	0,100	0,600
Condensador 0603	10	0,100	1,000
Condensador 0805	3	0,100	0,300
Condensador 1206	8	0,110	0,880
Condensador Elect. 220uF/250V	4	2,150	8,600
Bobina 0805	1	0,630	0,630
Diodo 1N914	3	0,140	0,420
Diodo LED 0805	1	0,320	0,320
Diodo SM712	1	1,080	1,080
Diodo US1M	5	0,249	1,245
Cristal Oscilador 8 MHz	1	0,560	0,560
Conector *Terminal Block* 3	2	0,760	1,520
Conector *Terminal Block* 5	1	0,900	0,900
Conector *Terminal Block* 9	1	1,980	1,980
Header 1X2	2	0,330	0,660
Header 2X3	1	0,400	0,400
Header 2X5	1	0,550	0,550
Toggle Switch	2	0,690	1,380
Conector DB9	1	1,570	1,570
uC ATmega64M1-MU	1	8,360	8,360
Inversor CD4069UBCM	1	0,440	0,440
Bobina CAN Transceiver	1	1,020	1,020
High/Low Side Driver IR2110S	3	1,697	5,091
N-Channel MOSFET IRFP4568	6	7,920	47,520
RS-232 Transceiver MAX232	1	3,260	3,260
CAN Tranceiver SN65HVD230QD	1	0,800	0,800
Regulador MC78L05CG	1	0,510	0,510
		Total:	**94,16**

10.2. Fabricación PCB

TABLA 10-2: PRESUPUESTO FABRICACIÓN PCB CONTROLADORA

Material	Precio (€)
Placa Circuito Impreso 1,6mm	9,08
Productos químicos	20,00
Pintura	20,00
Materiales varios	25,00
Mano de obra	75,00
Total:	**149,08**

10.3. Fabricación Prototipo

TABLA 10-3: PRESUPUESTO FABRICACIÓN PROTOTIPO

Material	Precio (€)
Barra Aluminio	9,08
Motor BLDC 42BL100	91,36
Materiales Varios	25,00
Mano de obra	50,00
Total:	**175,44**

10.4. Mano de Obra

TABLA 10-4: PRESUPUESTO MANO DE OBRA

Fase	Precio/hora (€)	Tiempo (Horas)	Precio Total (€)
Diseño y Fabricación Controladora	20,00	100	2000,00
Desarrollo Software	20,00	120	2400,00
Desarrollo de prototipo	20,00	50	1000,00
Test del prototipo	20,00	30	600,00
	Total:	**300**	**6000,00**

10.5. Presupuesto Final

En la Tabla 10-5 se muestra el presupuesto total del proyecto. La duración total del proyecto ha sido de 300 horas.

TABLA 10-5: PRESUPUESTO FINAL

Fase	Precio (€)
Materiales	94,16
Fabricación PCB	149,08
Fabricación Prototipo	175,44
Mano de Obra	6000,00
I.G.I.C. (7%)	449,30
Total:	**6867,98**

11. Bibliografía

[1] Tesla Motors: *http://www.teslamotors.com/*, último acceso Mayo 2014.

[2] Mahindra Reva: *http://mahindrareva.com/*, último acceso Mayo 2014.

[3] Renault Vehículos Eléctricos: *http://www.renault.es/gama-renault/gama-vehiculos-electricos/*, último acceso Mayo 2014.

[4] Nissan LEAF: *http://www.nissan.es/ES/es/vehicle/electric-vehicles/leaf.html*, último acceso Mayo 2014.

[5] Smart Dor Two Electric Drive: *https://www2.smart.com/es/es/index/smart-fortwo-electric-drive.html*, último acceso Mayo 2014.

[6] Volkswagen: *http://www.volkswagen.es/es.html*, último acceso Mayo 2014.

[7] BMW: *https://www.bmw.es/home/home.html*, último acceso Mayo 2014.

[8] Juanpere Tolrà, Roger, "Técnicas de control para motores Brushless, Comparativa entre conmutación Trapezoidal, conmutación Sinusoidal y Control Vectorial", ingenia-cat S.L., Motion Control Department, Barcelona, España.

[9] Carlos Andrés Mesa Montoya, *"CAN BUS Introducción a los sistemas de comunicación del vehículo"*, SENA Virtual, Contact Center Risaralda, 2008.

[10] Aurelio Vega Martínez, *"Buses de Campo"*, Asignatura Intrumentación Electrónica, Máster en Tecnologías de la Telecomunicación, Instituto de Microelectrónica Aplicada, Universidad de Las Palmas de Gran Canaria, 2013.

[11] CANOpen protocol: *http://www.can-cia.org/*, último acceso Marzo 2014.

[12] Thomas B. Reddy, "Linden´s handbook of batteries", McGraw-Hill, ISBN 0-07-135978-8.

[13] Davide Andrea, "Battery Management Systems for Large Lithium-ion Battery Pack", ARTECH HOUSE,. ISBN-13 978-1-60807-104-3, 2010.

[14] Atmel ATmega16/32/64M1 microcontrollers family: *http://www.atmel.com/devices/ATMEGA64M1.aspx*, último acceso Marzo 2014.

[15] Alphonsa Roslin Paul and Prof. Mary George (2011) "Brushless DC Motor Control Using Digital PWM Techniques" Proceedings of 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies (ICSCCN 2011), ISBN 978-1-61284-653-8, pp: 733 – 738.

[16] Hai-tao Wang, Ze Zhang, Xiang-yu Liu (2011) "Design of Control System for Brushless DC Motor Based on TMS320F28335" Third International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), ISBN 978-1-4244-9010-3, pp: 954 – 958.

[17] Radu Duma, Petru Dobra, Mirela Dobra, Ioan Valentin Sita (2011) "Low Cost Embedded Solution for BLDC Motor Control" 2011 15th International Conference on System Theory, Control, and Computing (ICSTCC), ISBN 978-1-4577-1173-2, pp: 1 – 6.

[18] Atmel Studio 6.0 Manual. [Online]. Available: *http://atmel.no/webdoc/atmelstudio/*, último acceso Marzo 2014.

[19] Atmel AVRISP mkII microcontroller programmer: *http://www.atmel.com/tools/avrispmkii.aspx*, último acceso Marzo 2014.

[20] SFP – IUMA – ULPGC: *http://www.iuma.ulpgc.es/services/sfp/*, último acceso Abril 2014.

[21] Altium Designer Software: *http://www.altium.com/en/products/altium-designer/overview*, último acceso Abril 2014.

[22] Wang Dongmei, Guo Haiyan, and Yu Jing (2011) "Modeling and Simulation Research of Brushless DC Motor open-loop Speed-adjustment System" The 2nd International Conference on Intelligent Control and Information Processing, ISBN 978-1-4577-0816-9, pp: 394 – 398.

[23] AVR194: Brushless DC Motor Control using ATmega32M1: *http://www.atmel.com/webdoc/atmel.docs/atmel.docs.33085.20306.html,* último acceso Enero 2014.

[24] NI LabVIEW: http://www.ni.com/labview/esa/, último acceso Mayo 2014.

[25] H. Garrido, J. M. Cabrera, V. Déniz, "Diseño de un sistema de gestión de baterías Li-Ion", PFC, Universidad de Las Palmas de Gran Canaria, Julio 2014.

[26] A. Mahtani, L. Sanchez, A. Martínez, D. García, D. Morales, E. Fernandez, F. Maniscalco, and J. Cabrera, "AVORA I : The SAUC-E'12 challenge", The C Continuum, 2012.

[27] Web TAEE 2014: http://www.taee2014.es/index.php/es/, último acceso Enero 2014.

[28] H. A. Fabelo, A. Vega, J. M. Cabrera, and V. Déniz, "A Brushless DC Motor Control Software C Library based on ATmega64M1 Applied to Teaching", IEEE Xplore XI TAEE, 2014.

[29] H. A. Fabelo, J. M. Cabrera, A. Vega, and V. Déniz, "A Low-Cost Control System for BLDC Motors Applied to Teaching",IEEE Xplore XI TAEE, 2014.

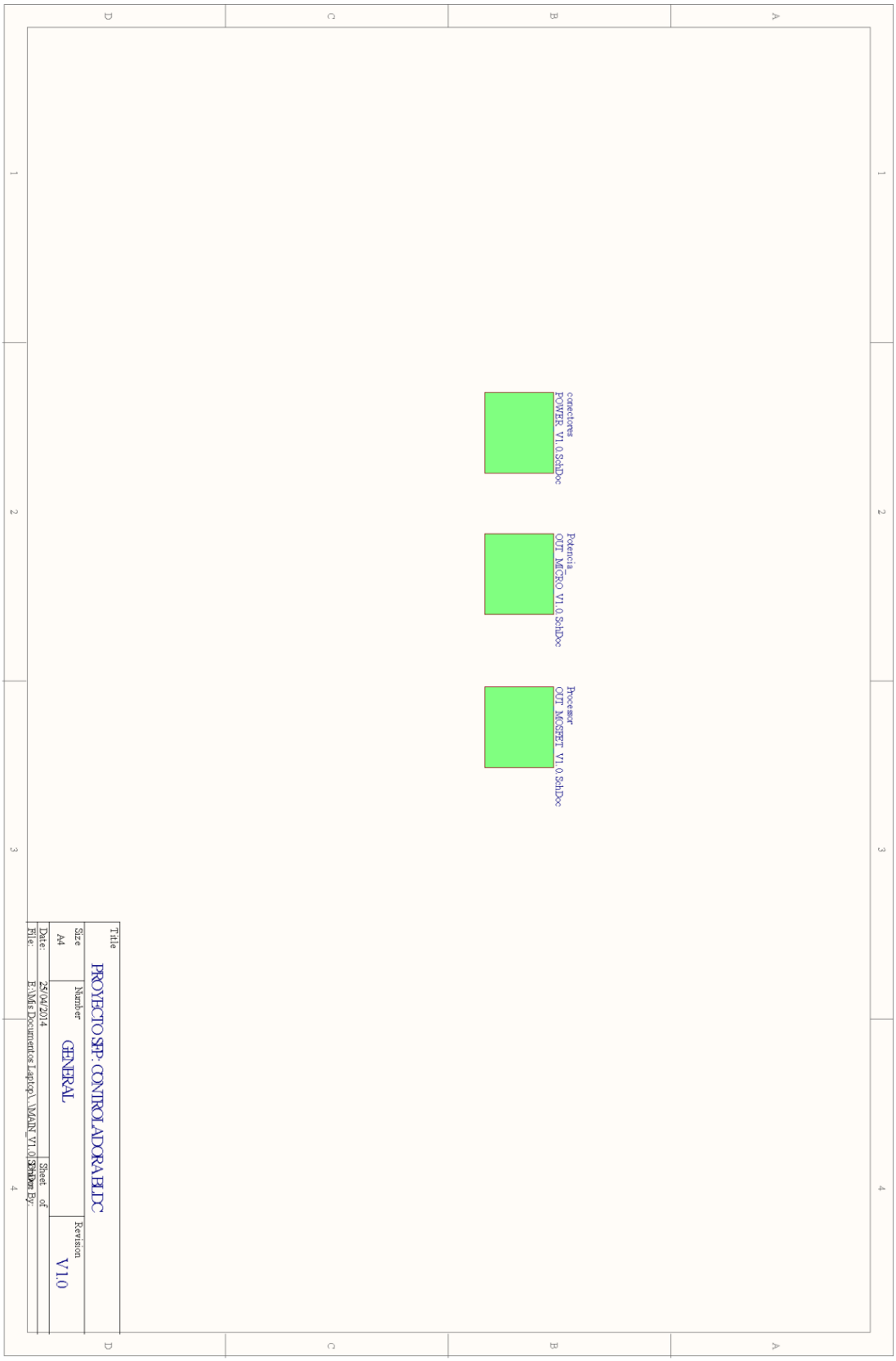
[30] V. Déniz, A. Vega, J. M. Cabrera, "Diseño de un sistema de gestión de baterías Li-Ion", PFC Universidad de Las Palmas de Gran Canaria, Enero 2014.

[31] J. M. Cabrera, A. Vega, F. Tobajas, V. Déniz and H. A. Fabelo, "Design of a Reconfigurable Li-Ion Battery Management System (BMS)",IEEE Xplore XI TAEE, 2014.

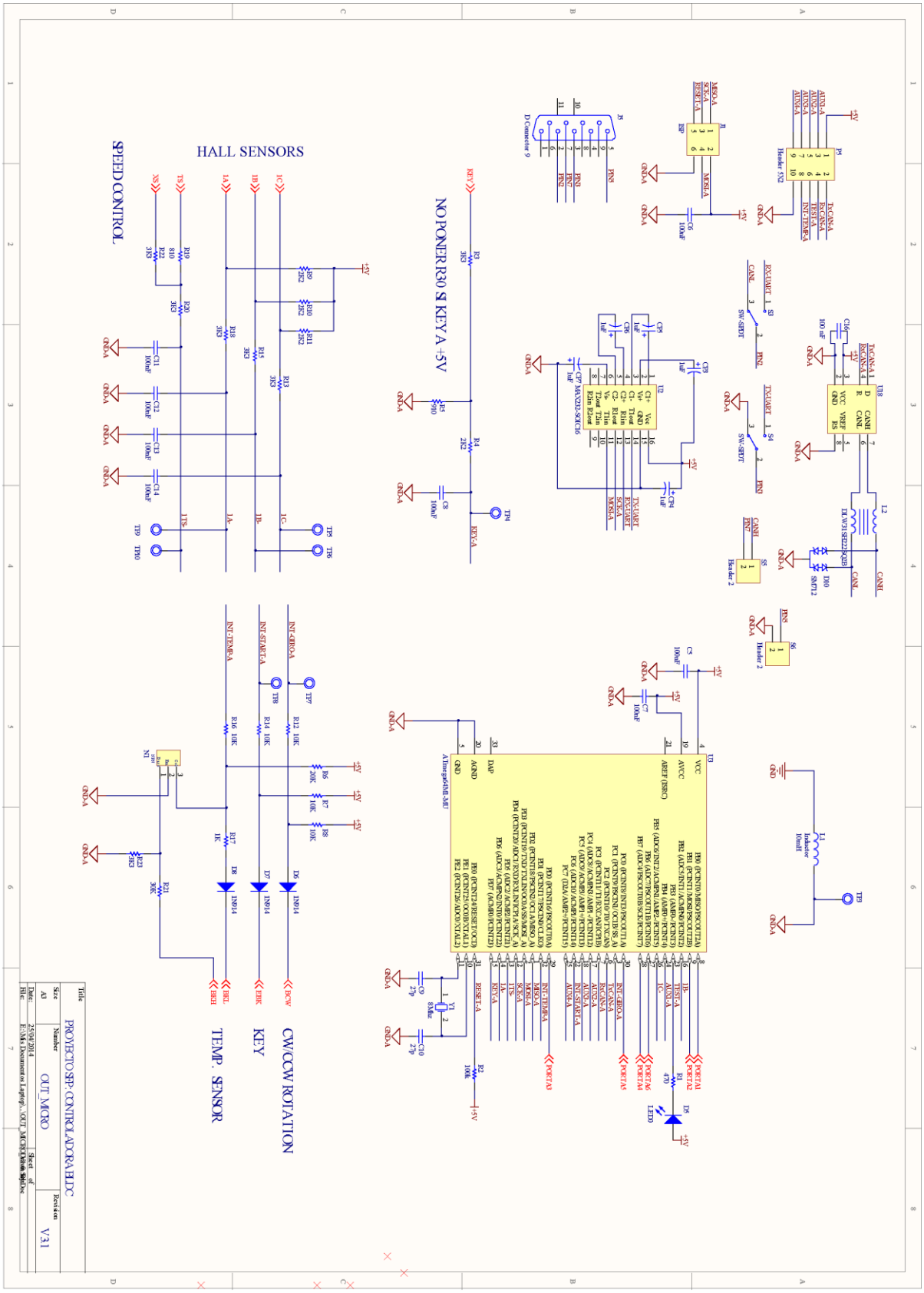
ANEXO I: FICHEROS PCB CONTROLADORA BLDCM

En este anexo se presentan los esquemáticos y las capas empleadas para la fabricación de la PCB mediante el software *Altium Designer*.

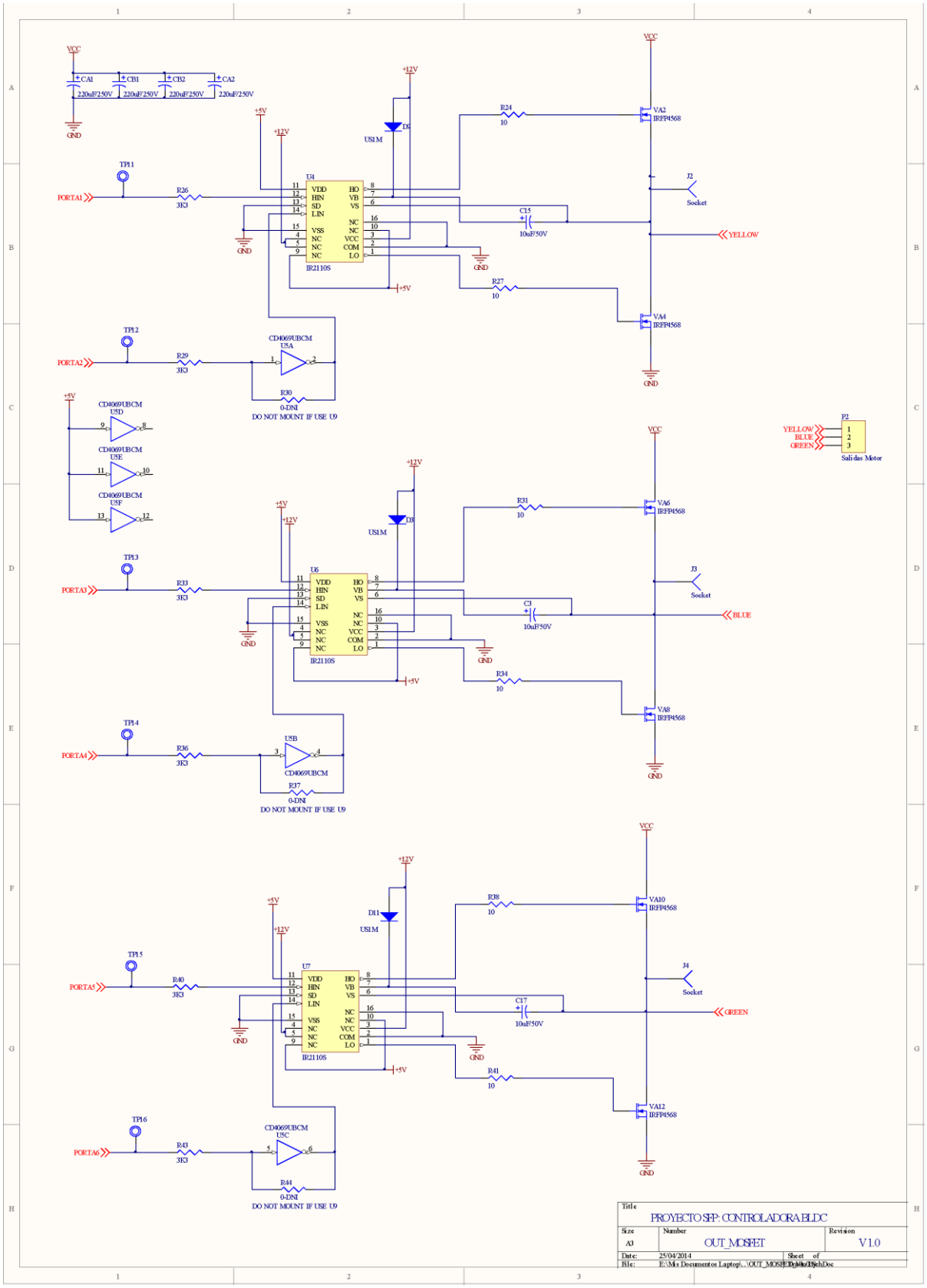
Fichero General



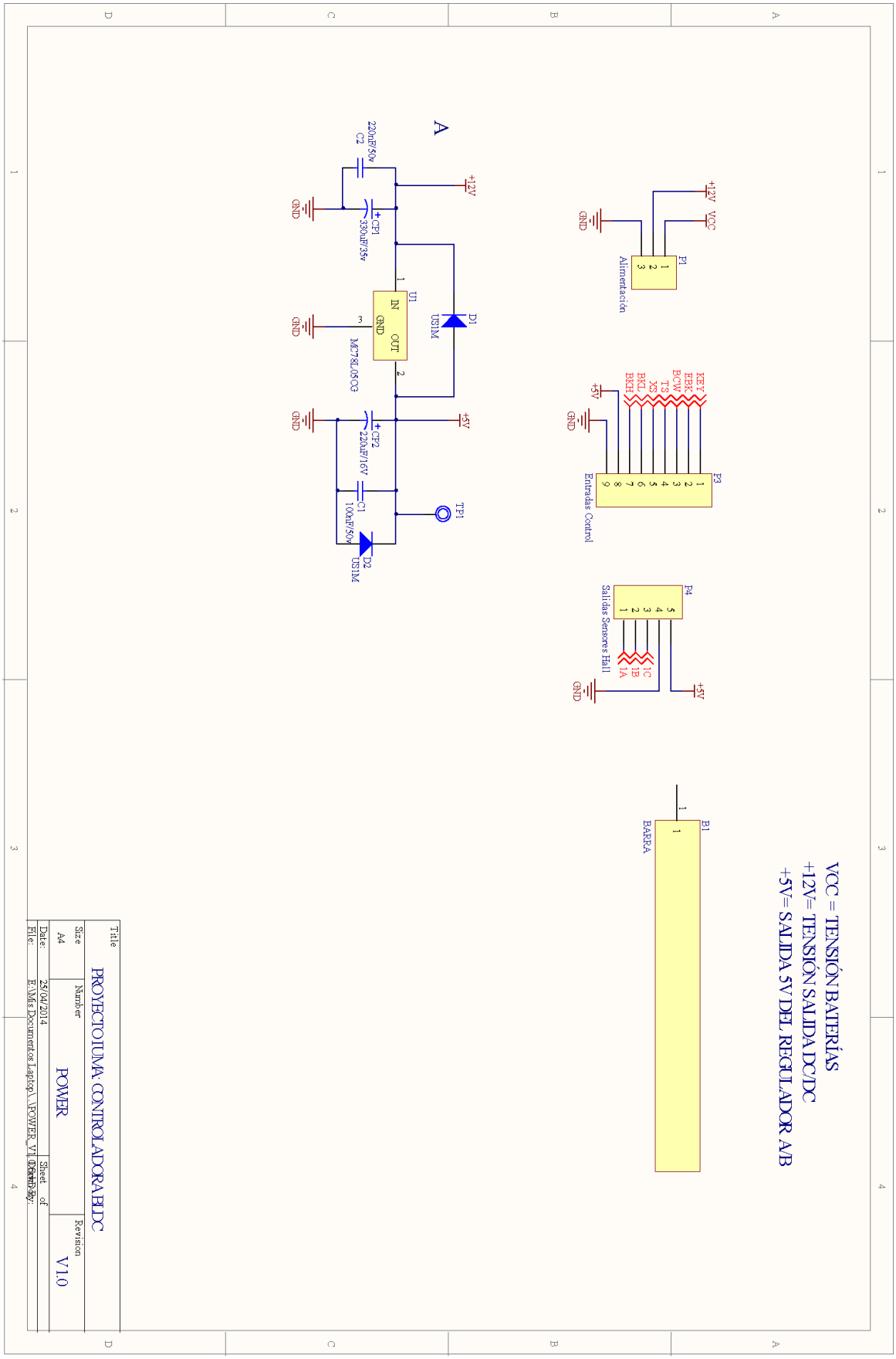
Esquemático Etapa de Control y Comunicación



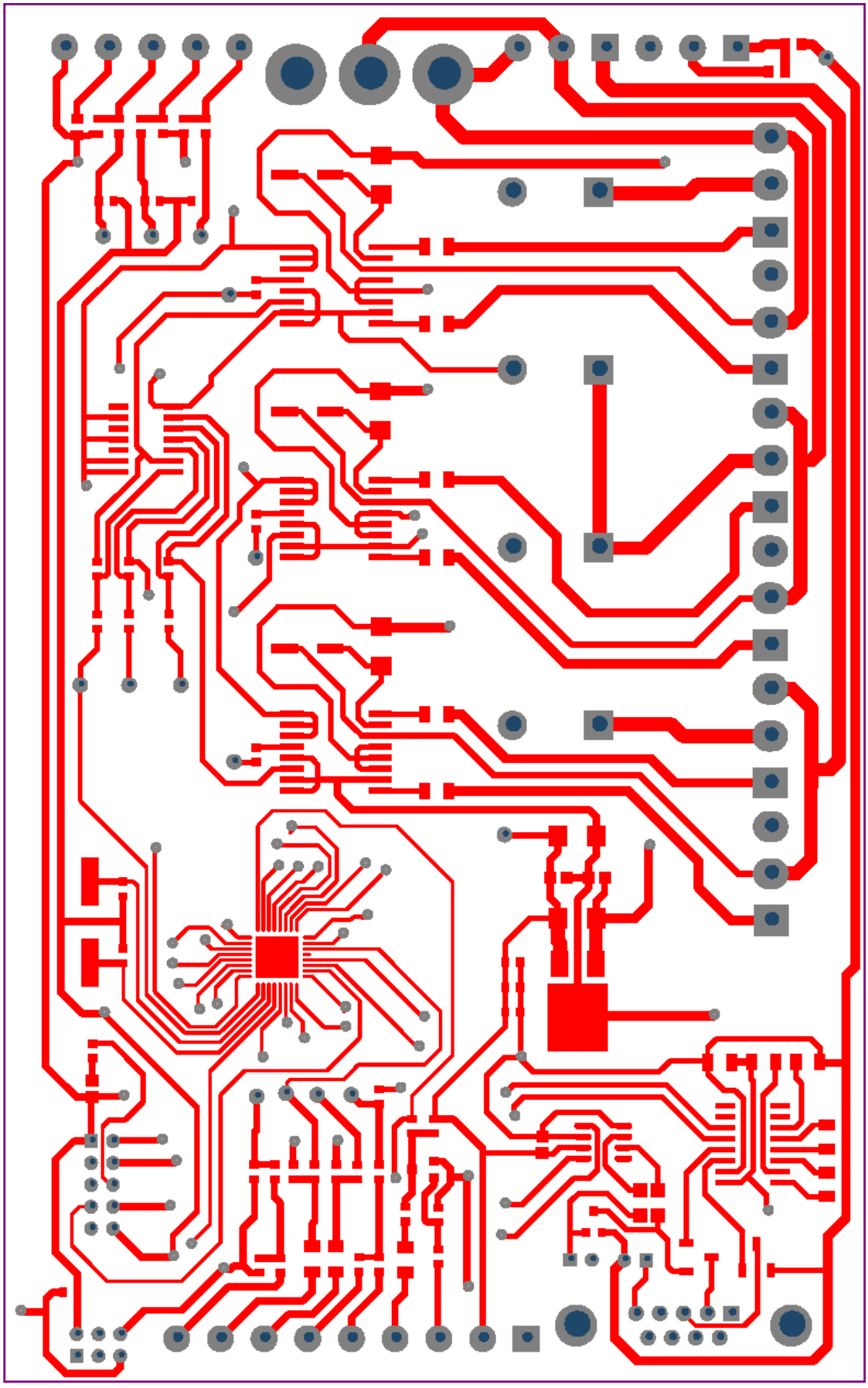
Esquemático Etapa de Potencia



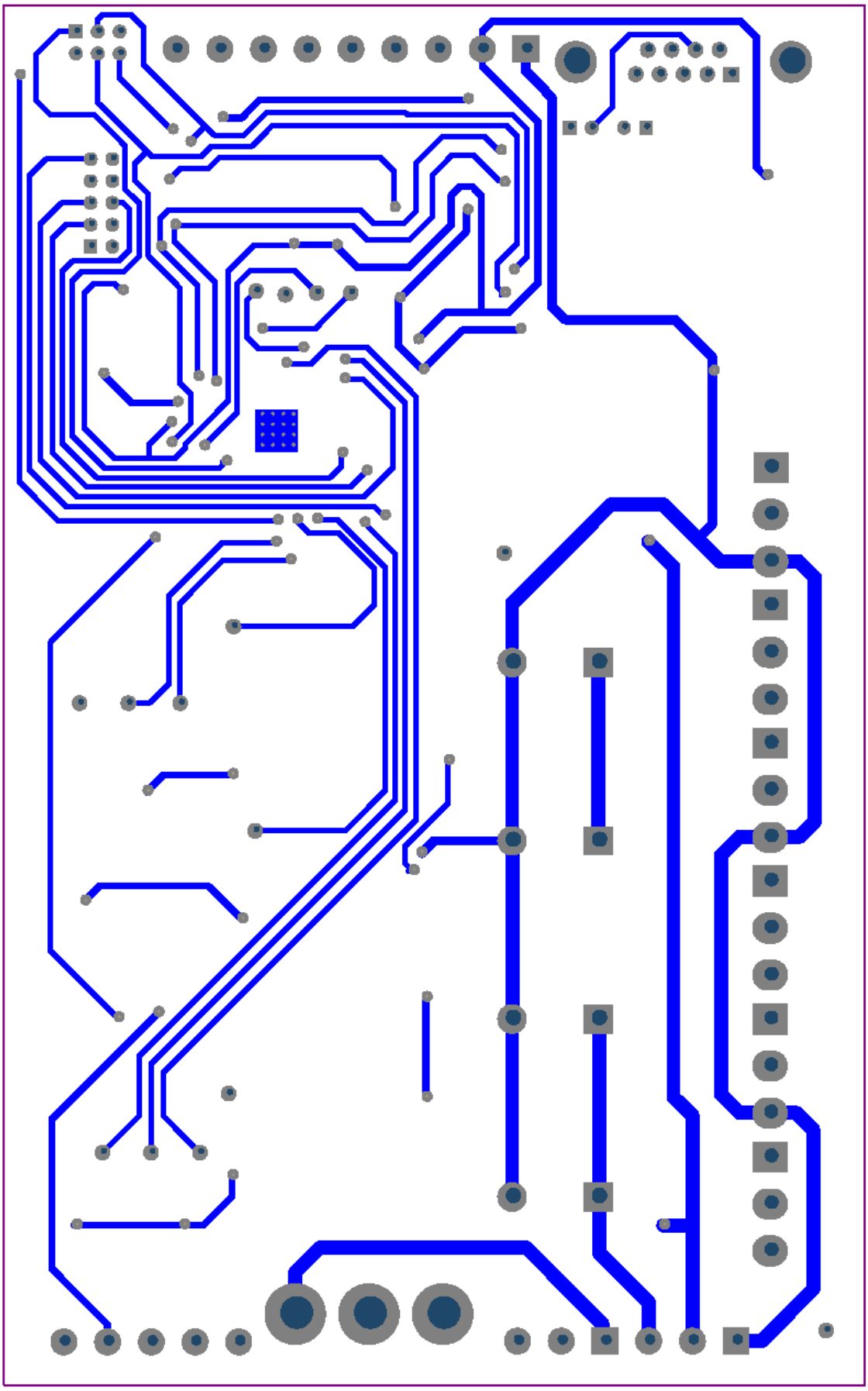
Esquemático Etapa de Alimentación



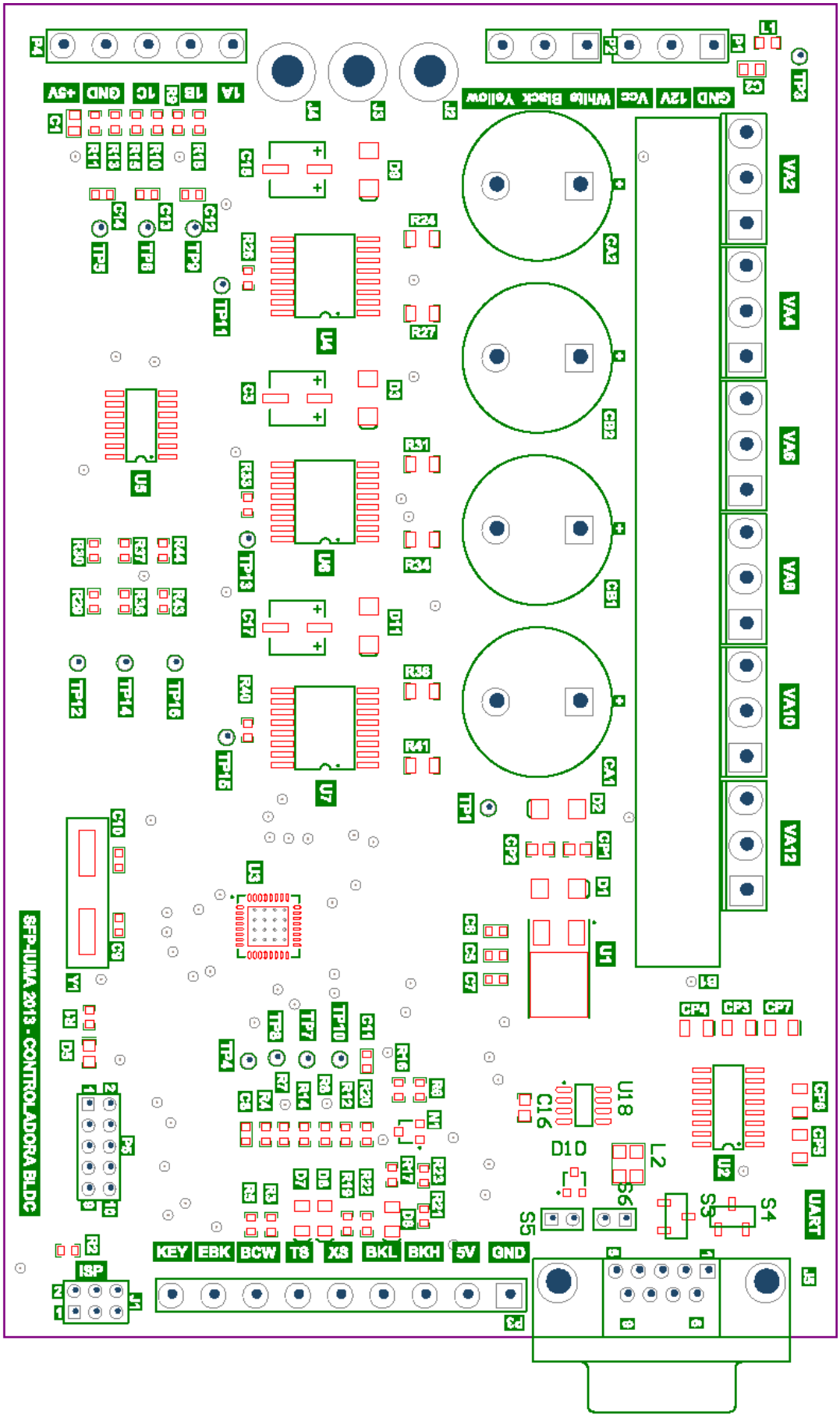
Top Layer



Bottom Layer



Distribución de los componentes y Overlay Layer



Listado de Componentes

Comment	Footprint	Designator	Qty.	Description
0-DNI	RES0603	R30, R37, R44	3	Resistor
1K	RES0603	R17	1	Resistor
1N914	MELF-1	D6, D7, D8	3	High Conductance Fast Diode
1uF	CAP1206	CP3, CP4, CP5, CP6, CP7	5	Polarized Capacitor (Radial)
2K2	RES0603	R4, R9, R10, R11	4	Resistor
3K3	RES0603	R3, R13, R15, R18, R19, R20, R22, R23, R26, R29, R33, R36, R40, R43	14	Resistor
8Mhz	XTAL	Y1	1	Crystal Oscillator
10	RES1206	R24, R27, R31, R34, R38, R41	6	Resistor
10K	RES0603	R7, R8, R12, R14, R16	5	Resistor
10uF/50V	F80_ELECTROLYTIC	C3, C15, C17	3	Polarized Capacitor (Surface Mount)
20K	RES0603	R6	1	Resistor
27p	CAP0603	C9, C10	2	Capacitor
30K	RES0603	R21	1	Resistor
100k	RES0603	R2	1	Resistor
100 nF	0805 CAP	C16	1	Condensador
100nF	CAP0603	C5, C6, C7, C8, C11, C12, C13, C14	8	Capacitor
100nF/50v	CAP0805	C1	1	Capacitor
220nF/50v	CAP0805	C2	1	Capacitor
220uF/16V	0805 CAP	CP2	1	Polarized Capacitor (Radial)
220uF/250V	CAPPRS10-18X20	CA1, CA2, CB1, CB2	4	Polarized Capacitor (Radial)
330uF/35v	0805 CAP	CP1	1	Polarized Capacitor (Radial)
470	RES0603	R1	1	Resistor
910	RES0603	R5	1	Resistor
Power Supply Connector	Block Connector-3	P1	1	Header, 3-Pin
ATmega64M1-MU	QFN7X7-32_N	U3	1	8-bit AVR Microcontroller, 64KB Flash, 2KB EEPROM, 4KB SRAM, Industrial Grade (-40° to +85°C), 32-Pin MLF
Dissipation Bar	BAR-10X105MM	B1	1	
CD4069UBCM	SOIC14	U5	1	Hex Inverter, 14-Pin SOIC
D Connector 9	DSUB1.385-2H9	J5	1	Receptacle Assembly, 9 Position, Right Angle
DLW31SH222SQ2B	Can Transceiver Inductor	L2	1	EMI Suppresion Filter - Common Mode Choke Coil

Control Signals In	Block Connector-9	P3	1	Header, 9-Pin
G1	SOT-23	N1	1	
Header 2	HDR1X2	S5, S6	2	Header, 2-Pin
Header 5X2	HDR2X5	P5	1	Header, 5-Pin, Dual row
Inductor	RES0805	L1	1	Inductor
IR2110S	Wide SOIC16	U4, U6, U7	3	High and Low Side Driver
IRFP4568	SUPER-247	VA2, VA4, VA6, VA8, VA10, VA12	6	N-Channel MOSFET
ISP	HDR2*3	J1	1	Header, 3-Pin, Dual row
LED0	Led_0805	D5	1	Typical INFRARED GaAs LED
MAX232-SOIC16	SOIC16	U2	1	
MC78L05CG	D2-PAK	U1	1	3-Terminal Low-Current Positive Fixed Voltage Regulator
Motor Out	Block Connector-3	P2	1	Header, 3-Pin
Hall Sensors Out	Block Connector-5	P4	1	Header, 5-Pin
SM712	SOT-23C	D10	1	Asymmetrical TVS Diode for RS-485
SN65HVD230QD	SOIC8_L	U18	1	IC 3.3V CAN TXRX W/STNDBY 8-SOIC
Socket	CON-1X	J2, J3, J4	3	Socket
SW-SPDT	CJS 1200TA	S3, S4	2	SPDT Subminiature Toggle Switch, Right Angle Mounting, Vertical Actuation
TP-POWER	TP-POWER	TP1, TP3, TP4, TP5, TP6, TP7, TP8, TP9, TP10, TP11, TP12, TP13, TP14, TP15, TP16	15	Test points
US1M	DIODE	D1, D2, D3, D9, D11	5	Default Diode

ANEXO II: CÓDIGO LIBRERÍA C

En este anexo se muestra el código de programa desarrollado para el microcontrolador ATmega64M1.

main_module.c

```
/*
 * main_module.c
 *
 * A BLDCM Control System Based on CAN Communications Project
 * For ATmega16/32/64M1
 * Created: 20/03/2014 12:28:58
 * Author: Himar A. Fabelo Gómez
 */

//////////////////////INCLUDES//////////////////////////////
#include "generic_configuration.h"
#include "get_set_variable.h"
#include "pid_control_module.h"
#include "uart_module.h"
#include "control_module.h"

//////////////////////VARIABLES//////////////////////////////
U8 regulation_period = 0;
U32 wait_send_speed = 0;
U8 sampling_period_factor = 40;

//////////////////FUNCTIONS////////////////////////////////
int main(void)
{

	micro_modules_initialization();

	while(1)
	{
		// Launch regulation loop
		// Timer 1 generates an IT (g_tick) all 256us
		// Sampling period = sampling_period_factor * 256us
		if ( ( get_sampling_time() == TRUE ) && ( get_motor_start() == RUN ) )
		{
			set_sampling_time(FALSE);

			regulation_period += 1;

			//sampling_period_factor * 256us = Te
			if ( regulation_period >= sampling_period_factor )
			{
				regulation_period = 0;

				if ( get_virtual_controls() == FALSE )
				{
					// Get external potentiometer value
					launch_adc();
				}

				// Set User Speed Command with potentiometer
				set_reference_speed( get_potentiometer_value() );

				launch_regulation_loop();
				duty_cycle_update( get_duty_cycle() );
			}

			if( wait_send_speed >= 500 )
			{
```

```
                        wait_send_speed = 0;

                        //Send the measured speed by UART
                        send_by_uart( get_measured_speed() );
                    }
                    wait_send_speed++;
                }
            }
}
```

---

generic_configuration.h

```
/*
 * generic_configuration.h
 *
 * A BLDCM Control System Based on CAN Communications Project
 * For ATmega16/32/64M1
 * Created: 20/03/2014 12:28:58
 * Author: Himar A. Fabelo Gómez
 */

#ifndef GENERIC_CONFIGURATION_H_
#define GENERIC_CONFIGURATION_H_

//////////////////////INCLUDES///////////////////////////////
#include <avr/io.h>
#include "avr/interrupt.h"
#include "stdio.h"
#include "stdlib.h"

//////////////////////TYPES DEFINITIONS/////////////////////////
typedef unsigned char       Bool;
typedef unsigned char       U8;
typedef unsigned short      U16;
typedef unsigned long       U32;
typedef signed char         S8;
typedef signed short        S16;
typedef long                S32;
//Sensor Hall Position Values
typedef  enum {HS_001=1,HS_010=2,HS_011=3,HS_100=4,HS_101=5,HS_110=6} Hall_Position;

//////////////////////CONSTANTS/////////////////////////////
//-------------------USER DEFINITIONS-----------------------//
//--------------------------UART----------------------------//
#define F_CPU  8000000      // Clock frequency
#define BAUD   38400        // Baud rate

// Speed Constant (KSPEED) Calculation
// K_SPEED = (60 * 255)/(n * t_timer0 * speed_max(rpm))
// with n=4 (8/2 pair of poles).
// and t_timer0 : 32us for Fclk=8Mhz
// Max RPM motor = 4000
#define K_SPEED 7470
// absolute_speed = alpha * measured_speed
// with alpha = 60 / (n * K_SPEED * t_timer0) = 15,6862

//----------------SYSTEM DEFINITIONS-------------------------//
#define FALSE               (0==1)
#define TRUE                (1==1)

#define MINDUTYCYCLE  90

#define Low(data_w)                 ((U8)data_w)
#define High(data_w)                ((U8)(data_w>>8))
```

```
//--------------------------ADC-----------------------------//
#define FREE                0
#define BUSY                1
#define ADC_INPUT_2     2
#define ADC_INPUT_GND   18

#define Adc_get_8_bits_result()          ((int)(ADCH))
#define Adc_get_10_bits_result()         ((int)(ADCL+((int)(ADCH<<8))))

#define Adc_select_channel(channel) \
              (ADMUX = ( ADMUX &
(~((1<<MUX4)|(1<<MUX3)|(1<<MUX2)|(1<<MUX1)|(1<<MUX0)))) \
              | (channel) )

#define Adc_right_adjust_result()        (ADMUX  &= ~(1<<ADLAR))
#define Adc_left_adjust_result()         (ADMUX  |=  (1<<ADLAR))

#define Adc_start_conv()                 (ADCSRA |= (1<<ADSC) )
#define Adc_start_conv_channel(channel)  (Adc_select_channel(channel),
Adc_start_conv() )

//--------------------------PSC-----------------------------//
//Hall sensor values to 8 bits register in the correct position
//Hall_Position = 00000xxx - 00000H3H2H1
#define HALL_SENSOR_VALUE()   (Hall_Position) \
       ( ( (PIND & (1<<PIND6)) >> 6 ) | ( (PINB & (1<<PINB2)) >> 1 ) | ( (PINB &
(1<<PINB5)) >> 3 ) )

//Low level to the PSC output ports
#define Clear_Port_Q1() (PORTD &= (~(1<<PORTB0)))
#define Clear_Port_Q2() (PORTB |= ((1<<PORTB1)))
#define Clear_Port_Q3() (PORTC &= (~(1<<PORTD0)))
#define Clear_Port_Q4() (PORTB |= ((1<<PORTB7)))
#define Clear_Port_Q5() (PORTB &= (~(1<<PORTC0)))
#define Clear_Port_Q6() (PORTB |= ((1<<PORTB6)))

//Enable/disable the correct output ports in the commutation
#define Set_Q1Q6()  POC =
(0<<POEN0A)|(0<<POEN0B)|(0<<POEN1A)|(1<<POEN1B)|(1<<POEN2A)|(0<<POEN2B);\
                                  Clear_Port_Q2();              \
                                  Clear_Port_Q3();              \
                                  Clear_Port_Q4();              \
                                  Clear_Port_Q5();

#define Set_Q1Q4()  POC =
(0<<POEN0A)|(1<<POEN0B)|(0<<POEN1A)|(0<<POEN1B)|(1<<POEN2A)|(0<<POEN2B);\
                                  Clear_Port_Q2();              \
                                  Clear_Port_Q3();              \
                                  Clear_Port_Q5();               \
                                  Clear_Port_Q6();

#define Set_Q5Q4()  POC =
(0<<POEN0A)|(1<<POEN0B)|(1<<POEN1A)|(0<<POEN1B)|(0<<POEN2A)|(0<<POEN2B);\
                                  Clear_Port_Q1();              \
                                  Clear_Port_Q2();              \
                                  Clear_Port_Q3();              \
                                  Clear_Port_Q6();

#define Set_Q5Q2()  POC =
(0<<POEN0A)|(0<<POEN0B)|(1<<POEN1A)|(0<<POEN1B)|(0<<POEN2A)|(1<<POEN2B);\
                                  Clear_Port_Q1();              \
                                  Clear_Port_Q3();              \
                                  Clear_Port_Q4();              \
                                  Clear_Port_Q6();

#define Set_Q3Q2()  POC =
(1<<POEN0A)|(0<<POEN0B)|(0<<POEN1A)|(0<<POEN1B)|(0<<POEN2A)|(1<<POEN2B);\
```

```
                                    Clear_Port_Q1();                \
                                    Clear_Port_Q4();                \
                                    Clear_Port_Q5();                 \
                                    Clear_Port_Q6();

#define Set_Q3Q6()  POC =
(1<<POEN0A)|(0<<POEN0B)|(0<<POEN1A)|(1<<POEN1B)|(0<<POEN2A)|(0<<POEN2B);\
                                    Clear_Port_Q1();                \
                                    Clear_Port_Q2();                \
                                    Clear_Port_Q4();                \
                                    Clear_Port_Q5();

//Stop motor. All PSC outputs in low level
#define Set_none()    POC =
(0<<POEN0A)|(0<<POEN0B)|(0<<POEN1A)|(0<<POEN1B)|(0<<POEN2A)|(0<<POEN2B);\
                                    Clear_Port_Q1();                \
                                    Clear_Port_Q2();                \
                                    Clear_Port_Q3();                \
                                    Clear_Port_Q4();                \
                                    Clear_Port_Q5();                \
                                    Clear_Port_Q6();

//Configuration of the PSC module registers
#define Psc_set_module_A(sa_val,ra_val,sb_val) \
              POCR0SAH = High(sa_val); \
              POCR0SAL = Low(sa_val);  \
              POCR0RAH = High(ra_val); \
              POCR0RAL = Low(ra_val);  \
              POCR0SBH = High(sb_val); \
              POCR0SBL = Low(sb_val);

#define Psc_set_module_B(sa_val,ra_val,sb_val) \
              POCR1SAH = High(sa_val); \
              POCR1SAL = Low(sa_val);  \
              POCR1RAH = High(ra_val); \
              POCR1RAL = Low(ra_val);  \
              POCR1SBH = High(sb_val); \
              POCR1SBL = Low(sb_val);

#define Psc_set_module_C(sa_val,ra_val,sb_val) \
              POCR2SAH = High(sa_val); \
              POCR2SAL = Low(sa_val);  \
              POCR2RAH = High(ra_val); \
              POCR2RAL = Low(ra_val);  \
              POCR2SBH = High(sb_val); \
              POCR2SBL = Low(sb_val);

#define Psc_set_register_RB(rb_val) \
              POCR_RBH = High(rb_val); \
              POCR_RBL = Low(rb_val);

//Configuration the PSC
#define A_SA_VAL 0 //!< POCR0SA = 0 for no pulse at init
#define A_RA_VAL 1 //!< POCR0RA = 1 to synchronize the ADC at the center of the
waveform
#define A_SB_VAL 0 //!< POCR0SB = 0 for no pulse at init
#define B_SA_VAL 0 //!< POCR1SA = 0 for no pulse at init
#define B_RA_VAL 1
#define B_SB_VAL 0 //!< POCR1SB = 0 for no pulse at init
#define C_SA_VAL 0 //!< POCR2SA = 0 for no pulse at init
#define C_RA_VAL 1
#define C_SB_VAL 0 //!< POCR2SB = 0 for no pulse at init

#define RB_VAL 255 //!< POCR_RB = 255 => PWM freq = PLL freq / 255
//PSC Cycle = (32MHz/8)/256*2 = 15,625kHz

//Configure the PMIC0 register
```

```
#define Psc_config_input_0(v1,v2,v3,v4,v5,v6)\
              PMIC0 = ((v1)<<POVEN0)| \
              ((v2)<<PISEL0)| \
              ((v3)<<PELEV0)| \
              ((v4)<<PFLTE0)| \
              ((v5)<<PAOC0)| \
              ((v6)<<PRFM00);

//Configure the PMIC1 register
#define Psc_config_input_1(v1,v2,v3,v4,v5,v6)\
              PMIC1 = ((v1)<<POVEN1)| \
              ((v2)<<PISEL1)| \
              ((v3)<<PELEV1)| \
              ((v4)<<PFLTE1)| \
              ((v5)<<PAOC1)| \
              ((v6)<<PRFM10);

//Configure the PMIC2 register
#define Psc_config_input_2(v1,v2,v3,v4,v5,v6)\
              PMIC2 = ((v1)<<POVEN2)| \
              ((v2)<<PISEL2)| \
              ((v3)<<PELEV2)| \
              ((v4)<<PFLTE2)| \
              ((v5)<<PAOC2)| \
              ((v6)<<PRFM20);

/* PMIC register definitions */
#define PSC_OVERLAP_DISABLE         0
#define PSC_OVERLAP_ENABLE          1

#define PSC_USE_PIN                 0
#define PSC_USE_COMPARATOR          1

#define PSC_USE_LOW_LEVEL           0
#define PSC_USE_HIGH_LEVEL          1

#define PSC_INPUT_FILTER_DISABLE    0
#define PSC_INPUT_FILTER_ENABLE     1

#define PSC_ASYNCHRONOUS_OUTPUT_CONTROL 0
#define PSC_SYNCHRONOUS_OUTPUT_CONTROL  1

#define PSC_INPUT_NO_ACTION         0
#define PSC_DISACTIVATE_OUTPUT_A    1
#define PSC_DISACTIVATE_OUTPUT_B    2
#define PSC_DISACTIVATE_OUTPUT_AB   3
#define PSC_DISACTIVATE_ALL_OUTPUTS 4
#define PSC_INPUT_HALT              6

//////////////////////////ENUMERATIONS//////////////////////////
//Define the two states of the Key
enum {ON = TRUE, OFF = FALSE};
//Define the two states of the motor
enum {RUN = TRUE, STOP = FALSE};
//Define the rotor direction: CCW (counter clock wise) and CW (clock wise)
enum {CCW = TRUE, CW = FALSE};
//------------------------SPEED MEASUREMENT-----------------------------//
//Regulation loop modes
enum {OPEN_LOOP  = 0, SPEED_LOOP = 1, CURRENT_LOOP = 2};

#endif /* GENERIC_CONFIGURATION_H_ */
```

---

control_module.h

```
/*
```

```
 * control_module.h
 *
 * A BLDCM Control System Based on CAN Communications Project
 * For ATmega16/32/64M1
 * Created: 20/03/2014 12:28:58
 * Author: Himar A. Fabelo Gómez
 */

#ifndef CONTROL_MODULE_H_
#define CONTROL_MODULE_H_

///////////////FUNCTION PROTOTYPES////////////////////////
void micro_modules_initialization(void);
void launch_adc(void);
void duty_cycle_update(U8 level);
void adc_initialization(void);
void timer1_initialization(void);
void timer0_initialization(void);
void rotation_direction_initialization(void);
void start_pll_32mhz(void);
void wait_pll_ready(void);
void psc_initialization(void);
void external_interrupt_initialization(void);
void output_psc_switch_commutations(Hall_Position position);
void calculate_estimated_speed(void);
void retry_run_motor(void);

#endif /* CONTROL_MODULE_H_ */
```

---

control_module.c

```
/*
 * control_module.c
 *
 * A BLDCM Control System Based on CAN Communications Project
 * For ATmega16/32/64M1
 * Created: 20/03/2014 12:28:58
 * Author: Himar A. Fabelo Gómez
 */

////////////////////////INCLUDES//////////////////////////////
#include "generic_configuration.h"
#include "control_module.h"
#include "get_set_variable.h"
#include "pid_control_module.h"
#include "uart_module.h"

////////////////////////VARIABLES//////////////////////////////
static U8 ovf_timer = 0; // variable "ovf_timer" is use to simulate a 16 bits timer
with 8 bits timer
Bool read_enable = FALSE;  // the speed can be read
U8 virtual_speed = 0;
U8 digit = 0;
U8 digit_count = 0;
U8 ms4_t0 = 0;
U16 Te = 0;

////////////////////////////////////FUNCTIONS//////////////////////////////////
void start_pll_32mhz(void)
{
	PLLCSR = (0<<PLLF)|(1<<PLLE);
}

void wait_pll_ready(void)
{
```

```
	while ( !( PLLCSR & (1<<PLOCK) ) );
}

//---------------Launch ADC to Capture the Potentiometer Value----------------//
void launch_adc(void)
{
	if( get_adc_state() == FREE )
	{
		set_adc_state(BUSY);
		Adc_left_adjust_result();
		Adc_start_conv_channel(ADC_INPUT_2);
	}
}

//---------------------Duty Cycle PSC Register Update ----------------------//
void duty_cycle_update(U8 level)
{
	U8 duty = level;

	//Set PULOCK Bit to 0 to modify de registers
	PCNF = (1<<PULOCK)|(1<<PMODE)|(0<<POPB)|(0<<POPA);

	Psc_set_module_A( duty, A_RA_VAL, duty );
	Psc_set_module_B( duty, B_RA_VAL, duty );
	Psc_set_module_C( duty, C_RA_VAL, duty );

	//Set PULOCK Bit to 1
	PCNF = (0<<PULOCK)|(1<<PMODE)|(0<<POPB)|(0<<POPA);
}

//---------PSC Output Commutation According to Hall Sensor Values------------//
void output_psc_switch_commutations(Hall_Position position)
{
	// Get the motor direction to commute right switches.
	char direction = get_motor_rotation_direction();

	// The switches are commuted only if the user start the motor
	if ( get_motor_start() )
	{
		duty_cycle_update( get_duty_cycle() );
		switch(position)
		{
		case HS_001:	if (direction==CCW)		{Set_Q5Q2();}
					else
	{Set_Q1Q6();}
					break;
		case HS_101:	if (direction==CCW)		{Set_Q3Q2();}
					else
	{Set_Q1Q4();}
					break;
		case HS_100:	if (direction==CCW)		{Set_Q3Q6();}
					else
	{Set_Q5Q4();}
					break;
		case HS_110:	if (direction==CCW)		{Set_Q1Q6();}
					else
	{Set_Q5Q2();}
					break;
		case HS_010:	if (direction==CCW)		{Set_Q1Q4();}
					else
	{Set_Q3Q2();}
					break;
		case HS_011:	if (direction==CCW)		{Set_Q5Q4();}
					else
	{Set_Q3Q6();}
					break;
			default :	break;
```

```
                }
        }
        else
        {
                // All switches are switched OFF
                Set_none();
        }
}

//-----------Timer 1 Initialization - CTC Mode to Launch ADC----------//
void timer1_initialization(void)
{
        // Interrupt every 256us [Frecuency (foc1a) = 1953,125hz (512us)]
        // Normal port operation + Mode CTC
        TCCR1A = 0;
        // Mode CTC + prescaler 64 - Value 8Mhz fclk
        TCCR1B = (1<<WGM12)|(0<<CS12)|(1<<CS11)|(1<<CS10);
        TCCR1C = 0;
        OCR1AH = 0;
        OCR1AL = 31;
        // Output compare A Match interrupt Enable
        TIMSK1=(1<<OCIE1A);
}

//-----Timer 0 Initialization - CTC Mode to Speed Measurement--------//
void timer0_initialization(void)
{
        TCCR0A = 0;
        // 1024 prescaler (32us) 8Mhz (fclk=31.250khz)
        // (Every 65ms the OvfInt is executed)
        TCCR0B = (1<<CS02)|(0<<CS01)|(1<<CS00);
        TIMSK0 = (1<<TOIE0);
}

//----PSC Initialization according to the settings in config.h------//
void psc_initialization (void)
{
        Psc_set_module_A(A_SA_VAL,A_RA_VAL,A_SB_VAL);
        Psc_set_module_B(B_SA_VAL,B_RA_VAL,B_SB_VAL);
        Psc_set_module_C(C_SA_VAL,C_RA_VAL,C_SB_VAL);
        Psc_set_register_RB(RB_VAL);

        // PMODE=Center Aligned Mode
        // PSC B Output Polarity(Low or High)
        // The PSC outputs A are active High
        PCNF = (0<<PULOCK)|(1<<PMODE)|(0<<POPB)|(0<<POPA);

        Psc_config_input_0(PSC_OVERLAP_ENABLE,\
        PSC_USE_PIN,\
        PSC_USE_LOW_LEVEL,\
        PSC_INPUT_FILTER_ENABLE,\
        PSC_SYNCHRONOUS_OUTPUT_CONTROL,\
        PSC_INPUT_NO_ACTION);

        Psc_config_input_1(PSC_OVERLAP_ENABLE,\
        PSC_USE_COMPARATOR,\
        PSC_USE_HIGH_LEVEL,\
        PSC_INPUT_FILTER_ENABLE,\
        PSC_SYNCHRONOUS_OUTPUT_CONTROL,\
        PSC_INPUT_NO_ACTION);

        Psc_config_input_2(PSC_OVERLAP_ENABLE,\
        PSC_USE_PIN,\
        PSC_USE_LOW_LEVEL,\
        PSC_INPUT_FILTER_ENABLE,\
        PSC_SYNCHRONOUS_OUTPUT_CONTROL,\
        PSC_INPUT_NO_ACTION);
```

```
        PIFR = (1<<PEV2)|(1<<PEV1)|(1<<PEV0)|(1<<PEOP);
        PIM = (1<<PEVE1);

        // No divider on PSC input clock
        // No Fast clock input (CLKPLL)
        // PLL/8 (prescaler)
        // PSC completes the entire waveform cycle before
        // halt operation requested by clearing PRUN
        // Start PSC
        PCTL = (1<<PPRE0)|(1<<PCLKSEL)|(0<<PCCYC)|(1<<PRUN);
}

//-------Interrupts for The Hall Sensors and External Controls----------//
void external_interrupt_initialization(void)
{
        // INT2(Hall C)-INT1(Hall B)-INT0(Hall A)
        EICRA = (1<<ISC20)|(1<<ISC10)|(1<<ISC00);
        EIMSK = (1<<INT2)|(1<<INT1)|(1<<INT0);

        // Pin Change Interrupt
        PCICR = (1<<PCIE1)|(1<<PCIE2);
        // Enable START Interrupt
        PCMSK1 = (1<<PCINT14);
        // Enable KEY y Motor Rotation Interrupt
        PCMSK2 = (1<<PCINT17)|(1<<PCINT23);

        sei(); //Enable the Interrupts
}

//---------ADC Initialization to Speed Control------------//
void adc_initialization(void)
{
        // VRef 2,56V - Left Adjust - ADC2(PD5) Input
        ADMUX=(3<<REFS0)|(0<<ADLAR)|(2<<MUX0);

        //ADC Enable - No start - Autotrigger Disable
        // Flag to 0 - Interrupt Enable - Prescaler=16
        ADCSRA = (1<<ADEN)|(0<<ADSC)|(0<<ADATE)|(0<<ADIF)|(1<<ADIE)|(4<<ADPS0);

        // ADC Hight Speed Mode - 100uA to the AREF
        // Disconnected internal AREF circuit from the AREF - Free Running Mode
        ADCSRB = (1<<ADHSM)|(1<<ISRCEN)|(0<<AREFEN)|(0<<ADTS0);
}

void rotation_direction_initialization(void)
{
        if( PIND&(1<<PORTD1) )
        {
                set_motor_rotation_direction(CW);
        }
        else
        {
                set_motor_rotation_direction(CCW);
        }
}

//-----------Microcontroller Hardware Initialization------------//
void  micro_modules_initialization(void)
{

        //Configuration and initialization Start Led
        DDRB = (1<<DDB3);
        PORTB |=(1<<PORTB3);

        // Output Pin configuration (used by PSC outputs)
        // PB0 => UH     PB1 => UL
```

```
        // PD0 => VH     PB7 => VL
        // PC0 => WH     PB6 => WL

        // Output Low for MOSFET Drivers
        PORTB &= ~(1<<PORTB7 | 1<<PORTB6 | 1<<PORTB1 | 1<<PORTB0);
        PORTC &= ~(1<<PORTC0);
        PORTD &= ~(1<<PORTD0);

        // PORT B :
        DDRB = (1<<DDB7)|(1<<DDB6)|(1<<DDB3)|(1<<DDB1)|(1<<DDB0);
        // PORT C :
        DDRC = (1<<DDC0);
        // PORT D :
        DDRD = (1<<DDD0);

        // PULL-UPS
        PORTB &= (1<<PORTB4);
        PORTC &= (1<<PORTC1 | 1<<PORTC4 | 1<<PORTC5 | 1<<PORTC7);
        PORTE &= (1<<PORTE1 | 1<<PORTE2);

        adc_initialization();

        timer1_initialization();

        timer0_initialization();

        rotation_direction_initialization();

        uart_initialization();

        start_pll_32mhz();

        wait_pll_ready();

        psc_initialization();

        external_interrupt_initialization();
}

//-------Retry to Run the Motor if Speed is Null--------//
void retry_run_motor(void)
{
        set_motor_start(RUN);
        launch_regulation_loop();
        duty_cycle_update( get_duty_cycle() );
        output_psc_switch_commutations( HALL_SENSOR_VALUE() );
}

//---Calculate Estimated Speed with Hall Sensor A Rising Edge-------//
void calculate_estimated_speed(void)
{
        U16 timer_value;
        U32 new_measured_speed;

        if ( read_enable == TRUE )
        {
                // Two 8 bits variables are use to simulate a 16 bits timers
                timer_value = ( ovf_timer<<8 ) + TCNT0;

                if (timer_value == 0)
                {
                        timer_value += 1;
                } // warning DIV by 0

                new_measured_speed = K_SPEED / timer_value;

                if( new_measured_speed > 255 )
```

```
                {
                        // Saturation Variable
                        new_measured_speed = 255;
                }

                set_measured_speed( new_measured_speed );

                // Reset Timer 0 register and variables
                TCNT0=0x00;
                ovf_timer = 0;
                read_enable=FALSE;
        }
}

/////////////////INTERRUPTS//////////////////////////

//--------Hall Sensors External Interrupts----------//
ISR(INT0_vect)
{
        //Hall A Interruption
        output_psc_switch_commutations( HALL_SENSOR_VALUE() );

        // Speed calculation on rising edge of Hall A
        if ( PIND&(1<<PORTD6) )
        {
                        calculate_estimated_speed();
                        read_enable=FALSE; // Wait 1 period
        }
        else
        {
                read_enable=TRUE;
        }
}

ISR(INT1_vect)
{
        // Hall B Interrupt
        output_psc_switch_commutations( HALL_SENSOR_VALUE() );
}

ISR(INT2_vect)
{
        // Hall C Interrupt
        output_psc_switch_commutations( HALL_SENSOR_VALUE() );
}

//------------Pin Change Interruptions---------------//

//------------START/STOP Interruption---------------//
ISR(PCINT1_vect)
{
        if ( get_virtual_controls() == FALSE )
        {
                if( PINC&(1<<PORTC6) )
                {
                        // Disable PSC Outputs
                        Set_none();
                        set_motor_start(STOP);
                        // Led off
                        PORTB |=(1<<PORTB3);
                }
                else
                {
                        set_motor_start(RUN);
                        // Led On
                        PORTB &=~(1<<PORTB3);
                        // Enable PSC Outputs
```

```
                        output_psc_switch_commutations( HALL_SENSOR_VALUE() );
                }
        }
}

//--------Motor Rotation Interrupt----------//
ISR(PCINT2_vect)
{
        if ( get_virtual_controls() == FALSE )
        {
                // Motor Rotation control
                if( PIND&(1<<PORTD1) )
                {
                        // PSC outputs off - Motor stop
                        Set_none();
                        set_motor_start(STOP);
                        // Change the motor rotation direction
                        set_motor_rotation_direction(CW);
                        _delay_ms(200);
                        set_motor_start(RUN);
                        // Motor start
                        output_psc_switch_commutations( HALL_SENSOR_VALUE() );
                }
                else
                {
                        Set_none();
                        set_motor_start(STOP);
                        set_motor_rotation_direction(CCW);
                        _delay_ms(200);
                        set_motor_start(RUN);
                        output_psc_switch_commutations( HALL_SENSOR_VALUE() );
                }
        }
}

//------------ADC Interruption--------------//
ISR(ADC_vect)
{
        if ( get_virtual_controls() == FALSE )
        {
                Adc_select_channel(ADC_INPUT_GND);
                set_potentiometer_value( Adc_get_8_bits_result() );
                set_adc_state(FREE);
        }
}

//-----Timer 0 Overflow Interrupt to Speed Measurement-----//
ISR(TIMER0_OVF_vect)
{
        // Every 4ms executed the Interrupts Operations
        ms4_t0++;
        if ( ms4_t0 == 2 )
        {
                TCNT0=0x00;
                ovf_timer++;
                if( ovf_timer >= 100 )
                {
                        ovf_timer = 0;
                        set_measured_speed(0);
                        // If the motor was turning and no stop order
                        // was given, motor run automatically.
                        if(get_motor_start())
                        {
                                retry_run_motor();
                        }
                }
                ms4_t0 = 0;
```

```
	}
}
//-------Timer 1 Interruption for Regulation Loop-------//
ISR(TIMER1_COMPA_vect)
{
	set_sampling_time(TRUE);
}
//-------UART Reception Interrupt-------//
ISR(LIN_TC_vect)
{
	int y = 0;
	y=uart_rx();
	switch(y)
	{
		//Stop the Motor
		case 's':

			if ( get_virtual_controls() == TRUE )
			{
				Set_none();
				set_motor_start(STOP);
				PORTB |=(1<<PORTB3);
			}
			break;

		// Start the Motor
		case 'r':

			if ( get_virtual_controls() == TRUE )
			{
				set_motor_start(RUN);
				PORTB &=~(1<<PORTB3);
				output_psc_switch_commutations( HALL_SENSOR_VALUE() );
			}
			break;

		// Virtual controls activation
		case 'z':

			set_virtual_controls(TRUE);
			break;
		case 'v':

			set_virtual_controls(FALSE);
			break;

		// Change rotation of the motor to CCW
		case 'x':

			if ( get_virtual_controls() == TRUE )
			{
				Set_none();
				set_motor_start(STOP);
				set_motor_rotation_direction(CW);
				_delay_ms(200);
				set_motor_start(RUN);
				output_psc_switch_commutations( HALL_SENSOR_VALUE() );
			}
			break;

		// Change rotation of the motor to CW
		case 'y':

			if ( get_virtual_controls() == TRUE )
			{
				Set_none();
				set_motor_start(STOP);
```

```
                        set_motor_rotation_direction(CCW);
                        _delay_ms(200);
                        set_motor_start(RUN);
                        output_psc_switch_commutations( HALL_SENSOR_VALUE() );
                }
                break;

        default :
                break;
    }

    if( y >= '0' && y <= '9' )
    {
        if ( get_virtual_controls() == TRUE )
        {
                digit = y-48;

                virtual_speed = MINDUTYCYCLE + (digit*15);
                if ( digit == 9 )
                {
                        set_potentiometer_value(250);
                }
                else
                {
                        set_potentiometer_value(virtual_speed);
                }
        }
    }
}
```

---

pid_control_module.h

```
/*
 * pid_control_module.h
 *
 * A BLDCM Control System Based on CAN Communications Project
 * For ATmega16/32/64M1
 * Created: 20/03/2014 12:28:58
 * Author: Himar A. Fabelo Gómez
 */

#ifndef PID_CONTROL_MODULE_H_
#define PID_CONTROL_MODULE_H_

void launch_regulation_loop();

#endif /* PID_CONTROL_MODULE_H_ */
```

---

pid_control_module.c

```
/*
 * pid_control_module.c
 *
 * A BLDCM Control System Based on CAN Communications Project
 * For ATmega16/32/64M1
 * Created: 20/03/2014 12:28:58
 * Author: Himar A. Fabelo Gómez
 */

//////////////////////INCLUDES//////////////////////////////
#include "generic_configuration.h"
#include "pid_control_module.h"
#include "get_set_variable.h"
```

```
#include "pid_control_module.h"

//////////////////////VARIABLES///////////////////////////////
U8 regulation_type = OPEN_LOOP; // Define the type of regulation (OPEN_LOOP)

//////////////////////////FUNCTIONS//////////////////////////
//-------------Selection of Regulation Loop-----------------//

// Launch regulation Loop
// Apply new duty cycle on PWM
void launch_regulation_loop()
{
	switch(regulation_type)
	{
		case OPEN_LOOP:
			set_duty_cycle(get_reference_speed());
			break;
			// More cases for more regulation types
		default :
			break;
	}
}
```

Get_set_variable.h

```
/*
 * get_set_variable.h
 *
 * A BLDCM Control System Based on CAN Communications Project
 * For ATmega16/32/64M1
 * Created: 20/03/2014 12:28:58
 * Author: Himar A. Fabelo Gómez
 */

#ifndef GET_SET_VARIABLE_H_
#define GET_SET_VARIABLE_H_

////////////////FUNCTION PROTOTYPES////////////////////////
Bool get_sampling_time(void);
void set_sampling_time(Bool bValue);

Bool get_key_state(void);
void set_key_state(Bool bValue);

Bool get_motor_start(void);
void set_motor_start(Bool bValue);

Bool get_motor_rotation_direction(void);
void set_motor_rotation_direction(Bool bValue);

U8 get_reference_speed(void);
void set_reference_speed(U8 u8Speed);

U8 get_measured_speed(void);
void set_measured_speed(U8 u8Speed);

U8 get_potentiometer_value(void);
void set_potentiometer_value(U8 u8Value);

U8 get_duty_cycle(void);
void set_duty_cycle(U8 u8Value);

Bool get_virtual_controls(void);
void set_virtual_controls(Bool bValue);
```

```
#endif /* GET_SET_VARIABLE_H_ */
```

---

get_set_variable.c

```
/*
 * get_set_variables.c
 *
 * A BLDCM Control System Based on CAN Communications Project
 * For ATmega16/32/64M1
 * Created: 20/03/2014 12:28:58
 * Author: Himar A. Fabelo Gómez
 */

//////////////////////INCLUDES//////////////////
#include "generic_configuration.h"
#include "get_set_variable.h"

///////////////VARIABLES////////////////////////
volatile Bool g_tick;
volatile Bool key_state = OFF;
volatile Bool motor_start = STOP;
volatile Bool rotation_direction = CW;
volatile Bool virtual_controls = FALSE;

U8 ref_speed = 0;
U8 measured_speed = 0;
U8 potentiometer_value = 0;
U32 measured_current = 0;
U16 measured_speed_dc = 0;
U8 duty_cycle = 0;

// ADC State : running = BUSY not running = FREE
char ADC_State = FREE;

////////// GETTER & SETTER FUNCTIONS/////////////
//------------ADC State FREE/BUSY-----------------//
Bool get_adc_state(void)
{
	return ADC_State;
}
void set_adc_state(Bool bValue)
{
	ADC_State = bValue;
}
//----------Sampling Time TRUE/FALSE---------------//
Bool get_sampling_time(void)
{
	return g_tick;
}
void set_sampling_time(Bool bValue)
{
	g_tick = bValue;
}
//--------------Key State ON/OFF -----------------//
Bool get_key_state(void)
{
	return key_state;
}
void set_key_state(Bool bValue)
{
	key_state = bValue;
}
//------------Motor Start RUN/STOP---------------//
Bool get_motor_start(void)
{
```

```
	return motor_start;
}
void set_motor_start(Bool bValue)
{
	motor_start = bValue;
}
//--------Motor Rotation Direction CW/CCW---------//
Bool get_motor_rotation_direction(void)
{
	return rotation_direction;
}
void set_motor_rotation_direction(Bool bValue)
{
	rotation_direction = bValue;
}
//-----------Reference Speed U8------------------//
U8 get_reference_speed(void)
{
	return ref_speed;
}
void set_reference_speed(U8 u8Speed)
{
	ref_speed = u8Speed;
}
//--------------Measured Speed U8---------------//
U8 get_measured_speed(void)
{
  return measured_speed;
}
void set_measured_speed(U8 u8Speed)
{
  measured_speed = u8Speed;
}
//------------Potentiometer Value U8------------//
U8 get_potentiometer_value(void)
{
	return potentiometer_value;
}
void set_potentiometer_value(U8 u8Value)
{
	potentiometer_value = u8Value;
}
//----------Duty Cycle U8----------------------//
U8 get_duty_cycle(void)
{
	return duty_cycle;
}
void set_duty_cycle(U8 u8Value)
{
	if( u8Value > MINDUTYCYCLE)
	{
		duty_cycle = u8Value;
	}
	else
	{
		duty_cycle = MINDUTYCYCLE;
	}
}
//----------Virtual Controls-----------------//
Bool get_virtual_controls(void)
{
	return virtual_controls;
}
void set_virtual_controls(Bool bValue)
{
	virtual_controls = bValue;
}
```

uart_module.h

```
/*
 * uart_module.h
 *
 * A BLDCM Control System Based on CAN Communications Project
 * For ATmega16/32/64M1
 * Created: 20/03/2014 12:28:58
 * Author: Himar A. Fabelo Gómez
 */

#ifndef UART_MODULE_H_
#define UART_MODULE_H_

void uart_initialization(void);
U8 uart_tx (U8 byte_data);
U8 uart_rx (void);
void send_by_uart(U16 value);
U8 bin_to_ascii (U8 c);

#endif /* UART_MODULE_H_ */
```

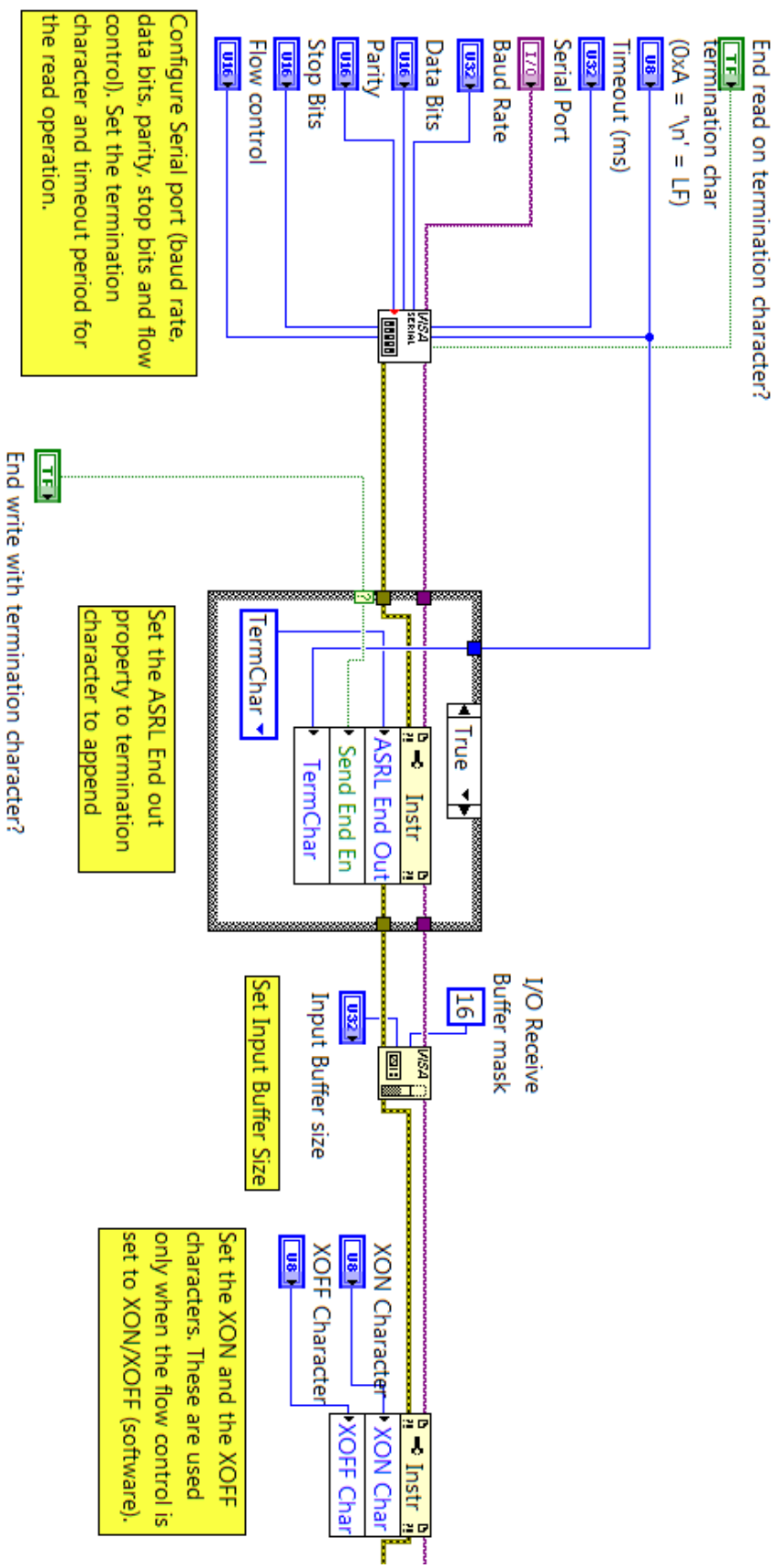
uart_module.c

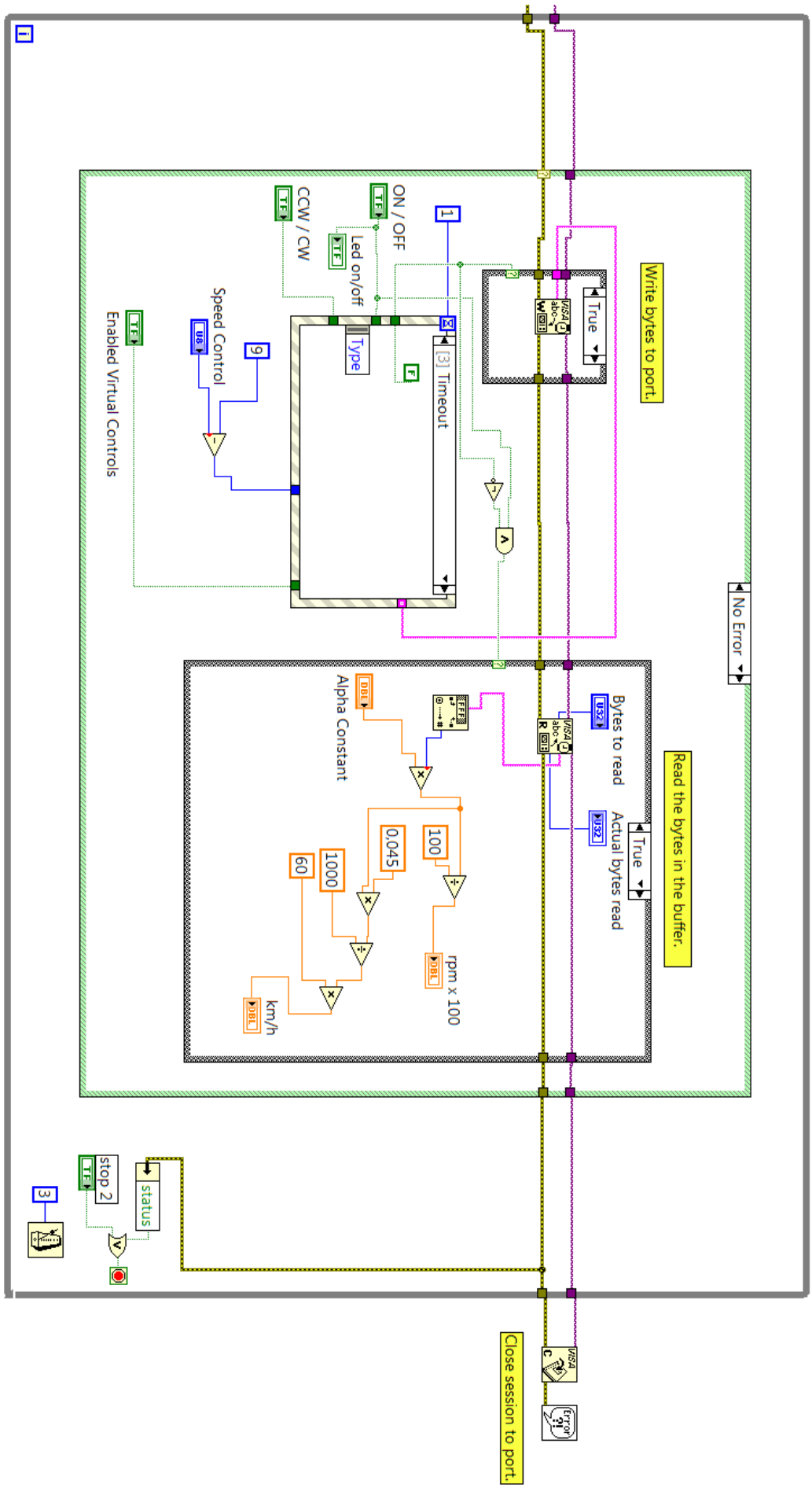
```
/*
 * uart_module.c
 *
 * A BLDCM Control System Based on CAN Communications Project
 * For ATmega16/32/64M1
 * Created: 20/03/2014 12:28:58
 * Author: Himar A. Fabelo Gómez
 */

#include "generic_configuration.h"
#include "uart_module.h"

//////////////////////VARIABLES//////////////////////////////
volatile unsigned int suma = 0, num = 0, contador = 0;
volatile unsigned char resto = 0;
volatile unsigned char digitos[5];

//------------------------UART-------------------------//
void uart_initialization(void)
{
	//UART/LIN Configuration Rx-Tx (8 bit + 1 stop bit - 38400 baudrate)
	LINCR = (1 << LSWRES);
	LINBRRH = (((F_CPU/BAUD)/26)-1)>>8;
	LINBRRL = (((F_CPU/BAUD)/26)-1);
	// 8MHz configuration option (LBT=26)
	LINBTR = (1 << LDISR) |
(0<<LBT5)|(1<<LBT4)|(1<<LBT3)|(0<<LBT2)|(1<<LBT1)|(0<<LBT0);
	LINCR = (1<<LENA)|(1<<LCMD2)|(1<<LCMD1)|(1<<LCMD0);
	LINENIR = (1<<LENRXOK);
}

int uart_tx (unsigned char byte_data)
{
	// Wait while the UART is busy.
	while (LINSIR & (1 << LBUSY));
	LINDAT = byte_data;
	return 0;
}
```

```
int uart_rx (void)
{
	// Wait while the UART is busy.
	while (LINSIR & (1 << LBUSY));
	return LINDAT;
}

U8 bin_to_ascii (U8 c)
{
	// Returns the ASCII value of a quartet
	if (c>=0x0A) return (c+('A'-0x0A));
	return (c+'0');
}

void send_by_uart(U16 value)
{
	// Send  by UART
	U8 c;
	U8 d4;
	U8 d3;
	U8 d2;
	U8 d1;

	d4 = (U16)(value >> 12) & 0x0F;
	d3 = (U16)(value >> 8) & 0x0F;
	d2 = (U16)(value >> 4) & 0x0F;
	d1 = (U16)(value) & 0x0F;

	if (d4 != 0)
	{
		c = bin_to_ascii(d4);
		uart_tx(c);

		c = bin_to_ascii(d3);
		uart_tx(c);

		c = bin_to_ascii(d2);
		uart_tx(c);
	}
	else
	{
		if (d3 != 0)
		{
			c = bin_to_ascii(d3);
			uart_tx(c);

			c = bin_to_ascii(d2);
			uart_tx(c);
		}
		else
		{
			if (d2 != 0)
			{
				c = bin_to_ascii(d2);
				uart_tx(c);
			}
		}
	}

	c = bin_to_ascii(d1);
	uart_tx(c);
	uart_tx(0xA);
}
```

ANEXO III: DIAGRAMA DE BLOQUES SOFTWARE LABVIEW

En este anexo se muestra el diagrama de bloques del software de control y monitorización desarrollado en LabVIEW.




Write bytes to port.
Read the bytes in the buffer.
Close session to port.
No Error
ON / OFF
CCW / CW
Led on/off
Speed Control
Enabled Virtual Controls
Type
[3] Timeout
Alpha Constant
Bytes to read
Actual bytes read
rpm x 100
km/h
stop 2
status
True
0.045
100
60
1000

ANEXO IV: ARTÍCULOS PRESENTADOS EN TAEE 2014

Artículos presentados en el XI congreso TAEE 2014 celebrado en Bilbao en Junio de 2014.

1. H. A. Fabelo, A. Vega, J. M. Cabrera, and V. Déniz, "A Brushless DC Motor Control Software C Library based on ATmega64M1 Applied to Teaching", IEEE Xplore - XI TAEE, 2014.

2. H. A. Fabelo, J. M. Cabrera, A. Vega, and V. Déniz, "A Low-Cost Control System for BLDC Motors Applied to Teaching",IEEE Xplore - XI TAEE, 2014.

3. J. M. Cabrera, A. Vega, F. Tobajas, V. Déniz and H. A. Fabelo, "Design of a Reconfigurable Li-Ion Battery Management System (BMS)",IEEE Xplore - XI TAEE, 2014.

A Brushless DC Motor Control Software C Library based on ATmega64M1 Applied to Teaching

Himar A. Fabelo, José Cabrera, Aurelio Vega, Víctor Déniz
Dept. of Electronics Engineering and Automatics (DIEA)
Institute for Applied Microelectronics (IUMA)
University of Las Palmas de Gran Canaria (ULPGC), Spain
hfabelo@iuma.ulpgc.es, jose.cabrera@ulpgc.es, avega@iuma.ulpgc.es, vdgonzalez@iuma.ulpgc.es

***Abstract*— This paper describes the architecture of a C library developed for the control of a brushless direct current motor. The library has been made using a modular programming methodology. The control system is based on ATmega64M1 microcontroller integrated into a controller for 3-phase brushless direct current motors. The controller has been especially designed and manufactured for this project. This library has been mainly created to be used as an educational resource in teaching of practical sessions of microcontrollers programming and motor control systems. With it, students can learn the structure and the operation of the most used control systems currently that are replacing to the traditional direct current motors with brushes.**

Keywords*— brushless direct current motor (BLDCM), BLDC motor control C library, microcontroller ATmega64M1, analog to digital converter (ADC), Power Stage Controller (PSC).

I. INTRODUCTION

BLDC motors (Brushless Direct Current Motors) operate by an electronic commutation signalized by solid state Hall sensors instead of a brushes commutation. These sensors indicate to the microcontroller (µC) the position of the motor rotor. The controller, from these signals, must excite the motor coils by a right switching logic. Software to perform these functions is needed. It must also control the speed adjustment of the system, the motor temperature sensor (if the motor has one) or the communications to the outside.

At present the trend at the time of performing almost any type of software, is to develop it in a modular way. One advantage of this development methodology is to obtain the ability to reuse code developed, achieving high productivity and reduced time in future work.

By the modular programming, large and complex problems are divided into several smaller and simple subproblems. These subproblems, in turn, are divided into simpler ones. These steps must be carried out again and again to get items that are simple enough to be easily resolved. This technique is called "successive refinement" or "top-down design".

By this project, students of electronics and robotics will understand and study the modular programming and the BLDC motor control systems. Thus, by the experiments with this software and the complete control system, they will understand its functioning and, moreover, they will learn the C programming of µCs.

II. THE MOTOR CONTROL SOFTWARE C LIBRARY

The µC chosen for the development of this project is the ATmega64M1 [1], which has a 64 kB flash memory and a SRAM (Static Random Access Memory) of 4 kB. Within this series of ATmega, there are also the Atmega16M1 and ATmega32M1 with a flash memory capacity and a SRAM of 16 kB and 1 kB, and 32 kB and 2 kB respectively. The µC model is chosen according to the application and the size of the software. The ATmega64M1 was chosen for this project due to its larger flash memory capacity in which future software upgrades will be able to be placed.

The development of the function library for controlling BLDC motors is based on the open-loop control with speed adjustment technique [2][3][4][5]. Fig. 1 shows the most important components of the µC ATmega64M1 for this application.

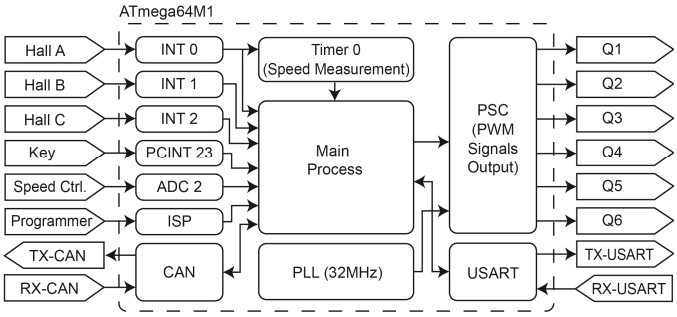


Fig. 1. Diagram of the parts of the µC used in the BLDCM controller

The Hall sensor signals (A, B and C) from the motor enter in the µC through the ports that have the external interrupts with higher priority levels. These interrupts act on the main system process activating the switching of the Q1 to Q6 outputs of the PSC (Power Stage Controller) module in the correct order. The interrupt of the A Hall sensor carries out the measurement of motor speed by the timer 0.

The input signal "Key" starts or stops the motor. This active low signal is introduced through the pin change interrupt 23 (PCINT 23). This interrupt is executed when a signal level

978-1-4799-6002-6/14/$31.00 ©2014 IEEE

A Low-Cost Control System for BLDC Motors Applied to Teaching

Himar A. Fabelo, Aurelio Vega, José Cabrera, Víctor Déniz
Dept. of Electronics Engineering and Automatics (DIEA)
Institute for Applied Microelectronics (IUMA)
University of Las Palmas de Gran Canaria (ULPGC), Spain
hfabelo@iuma.ulpgc.es, avega@iuma.ulpgc.es, jose.cabrera@ulpgc.es, vdgonzalez@iuma.ulpgc.es

***Abstract*— In this paper, the process carried out to design and manufacture the hardware of a low-cost control system for brushless direct current motors based on a ATmega64M1 is described. The proposed control system, applied to teaching, consists of the controller presented in this paper and a C library specially created for this project. Three types of different control systems are detailed. The first is the basic system, which can be used for teaching microcontroller programming within the scope of brushless motors. The second system presented is used to monitor and manage electric vehicles powered by a 6-phase brushless motor. This system offers the possibility to be used as a practice teaching module for electromechanical students. Finally, we detail a system that enables the control of remotely operated vehicles. This configuration can also be oriented to the control of any type of robotic vehicle powered by brushless motors. These three systems use the controller proposed in this article. The difference between them is the number of controllers that are required for their operation and how the system components are interconnected.**

Keywords*— *brushless direct current motor (BLDCM), BLDC control system, electric vehicle control system, remotely operated vehicle control system.

I. INTRODUCTION

BLDC motors (Brushless Direct Current Motors) have excellent torque characteristics, high performance, and a very wide range of speeds, plus a lifetime. They have no brushes switching, so the need for periodic maintenance is reduced. However, the use of a complex control system based on a µC (microcontroller) that manages the switching of the motor coils is necessary.

Nowadays, these motors are a viable alternative to traditional DC (Direct Current Motors) motors in almost any application. BLDC motors are used in industries such as automotive, aerospace, consumer electronics, medical, automation equipment and instrumentation, etc.

This project has focused primarily on applications for electric or hybrid cars, electric motorcycles, ROVs (Remotely Operated Vehicles) and similar applications within the field of robotics.

For the reasons given above and the degree of complexity of these new systems versus traditional systems of DC motors, students of electronics, electromechanical and robotics should know the features and operation of the control systems for BLDC motors. With a broad base of knowledge in this field, they may confront the working world in the coming years in the above sectors.

II. THE CONTROLLER

The BLDCM controller design is based on µC ATmega64M1 [1], which, by its characteristics is suitable for the control of this type of motors. The main features of this µC that are required for this project are as follows:

- An 8-bit AVR CPU.
- Up to 11 single ended channels 10-bit ADC (Analog to Digital Converter), one of which will be used to capture the analog signal from the speed adjustment potentiometer.
- A CAN (Controller Area Network) and RS-232 (Recommended Standard 232) interfaces used to carry out the communication between the controller and the testing and monitoring computer.
- A 12-bit high speed PSC (Power Stage Controller) module. This module controls the power stage and the switching of the BLDC motor coil signals by the technique of PWM (Pulse Width Modulation) [2][3][4].

A. Controller Design

The main stages in the design of the controller are three: the control stage, the communication stage and the power stage. At the control stage the configurations of the inputs and outputs of the µC are performed. In the communication stage, it is located the electronics used to interconnect the µC with the PC. Finally, in the power stage, they are located the drivers that raise the output voltage of the PSC module of the µC to get the right voltage for activating the MOSFET transistors. These transistors excite the motor coils with the correct voltage.

Fig. 1 shows the different modules of the controller's design.

978-1-4799-6002-6/14/$31.00 ©2014 IEEE

Design of a Reconfigurable Li-Ion Battery Management System (BMS)

José Cabrera, Aurelio Vega, Félix Tobajas, Víctor Déniz, Himar A. Fabelo
Dept. of Electronics Engineering and Automatics (DIEA)
Institute for Applied Microelectronics (IUMA)
University of Las Palmas de Gran Canaria (ULPGC), Spain
jose.cabrera@ulpgc.es, avega@iuma.ulpgc.es, tobajas@iuma.ulpgc.es, vdgonzalez@iuma.ulpgc.es, hfabelo@iuma.ulpgc.es,

***Abstract*— In this paper, the design of a Battery Management System for a battery pack composed of Lithium-Ion cells is described. It specifies which lithium-ion technology is used for monitoring control signals such as the high voltage per cell, the start voltage balancing, the low voltage shutdown, and the maximum temperature of battery cells pack. The proposed system allows performing management practices in controlled charging and discharging of batteries.**

Keywords*— *Battery Management System (BMS), Metal Oxide Field Effect Transistor (MOSFET), Brushless Direct Current (BLDC), Printed Circuit Board (PCB).

I. INTRODUCTION

Nowadays, the implementation of a Battery Management System (BMS) is necessary because of the rising demand for electric vehicles. The main characteristic of BMS is the balance at the charge and discharge process of the battery pack. This increases the lifetime of the batteries and optimizes their performance [1]. In the market there are various technologies of lithium-ion cells with different control voltages, so that by varying only the voltage comparator in the design, it is possible cover almost all existing batteries [2].

II. BMS HARDWARE DESIGN

The proposed hardware design consists of an analogic BMS with centralized topology, so that the system has full control of the state of the cells, thus performing tasks of monitoring, protection and remote control over it.

The student develops the BMS schematic determining which type of lithium-ion cell technology will be used to make the layout design later. Finally, the student will manufacture the printed circuit board. In this way, the student will develop all phases of a real project, starting from the initial specifications to the phase of analysis and verification of the right operation of the BMS.

The student must specify which lithium-ion technology will be used, because the four voltage levels necessary for the design of comparators-samplers depend on it:

- Maximum voltage per cell.
- Starting voltage balancing or burned.
- Low voltage disconnecting cell.
- Hysteresis undervoltage shutdown after.
- Maximum temperature of the cell pack.

Once the schematic has been completed and documented we proceed to design and make the printed circuit board. This design includes modules for charging, discharging, balancing by burning excess energy, overcurrent protection, over temperature protection, and communications bus for future expansion.

Fig. 1 shows the location of the BMS in a generic control system. This system consists of a charger, a battery pack, the BMS, a control stage and the system load [3].

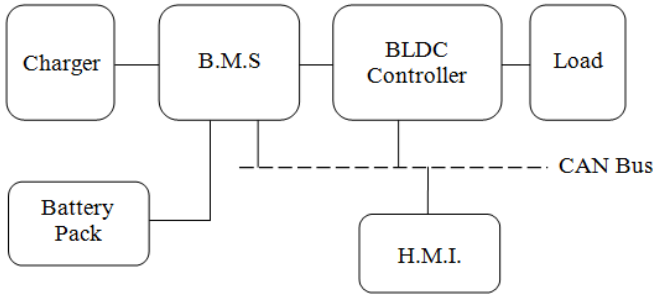


Fig. 1. Full system would be located the BMS

When any cell voltage reaches the maximum load, the BMS should be disconnected from the charger (Fig. 2). The BMS must also balance the cells to maximize their capacity. The system can make the balancing by disabling the load of the cell that holds the greatest voltage until it is low enough so that the charger supplies the load current again. After some cycles of this process, all the cells must be at the same voltage, which would mean that the pack is balanced.

Furthermore, as a cell reaches the minimum cutting voltage allowed, the BMS is disconnected from the load (Fig. 3).

978-1-4799-6002-6/14/$31.00 ©2014 IEEE

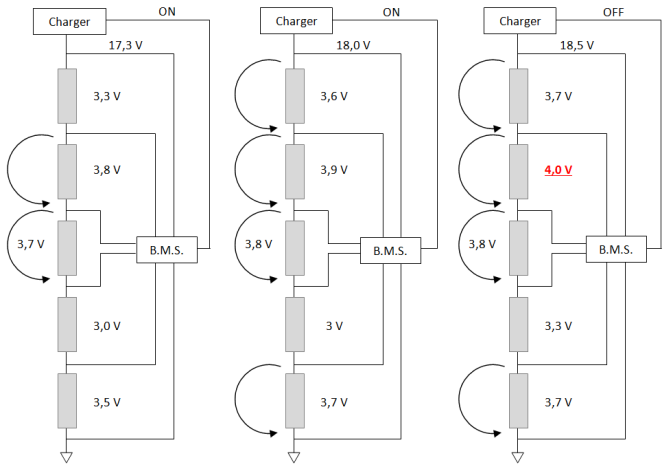


Fig. 2. Battery pack charge process

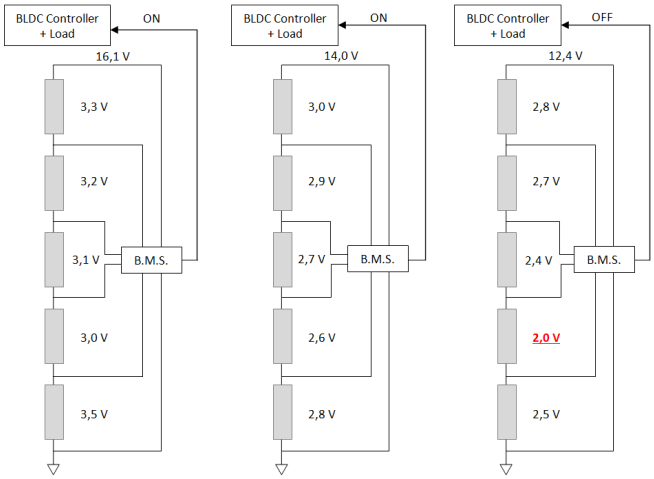


Fig. 3. Battery pack discharge process

III. CIRCUIT ANALYSIS

According to the initial considerations to the design of the BMS, its study can be divided into the stages described below.

The full circuit has several parts and each one has its function. All parts are connected to the battery pack. Fig. 4 shows a part of the schematic in which you can see that the cells are monitored by several control circuits at once. The monitoring is performed cell by cell. The connectors marked from B0 to B11 are consecutively connected from cell to cell as they are located in the battery pack

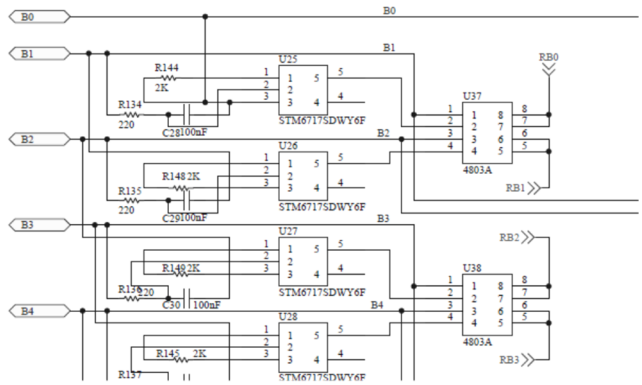
A. Charge Phase

At the charge phase, is carried out separately the monitoring of critical voltages of each cell. These voltages vary according to the type of technology used in the manufacture of the cells. The choice of the supervisor is based on the voltages, although the BMS can be reconfigured for each technology.

Fig. 6 shows part of the design of the phase in which the monitoring of the two cell control voltages (2 V and 4 V) is carried out.

The integrated circuit G4EN monitors these voltages. If the cell voltage is above 4 V, this device stops the charging and activates the burning phase. The cell charging is interrupted by the transistor Q28, which is controlled by the transistors associated with the supervised voltage, in this case, Q2.

a)



b)

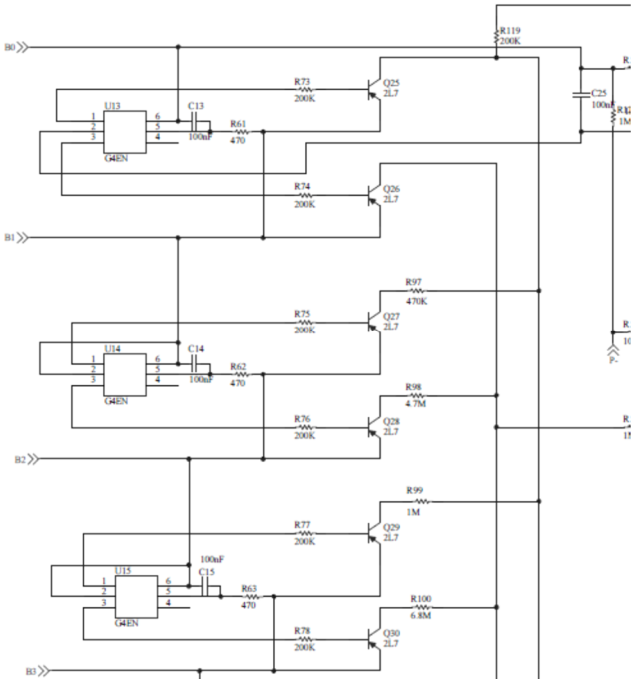


Fig. 4. a) A part of the circuit which carries out, cell by cell, the burning phase b) Minimum and maximum voltage supervisor circuit

The switching stage consists of four n-channel MOSFET transistors in parallel (Fig. 5), which connects and disconnects the charger to the system.

When Q28 (Fig. 6) is switched, it sets the voltage VGS=0 of the MOSFET transistors associated to the load, thus cutting off the conduction of current from the drain to the source (P-CH) [4]. If the cell voltage is lower to 3.6 V, then the current will flow again. In this way, any cell will operate with a voltage above the 4 V set up by the control circuit.

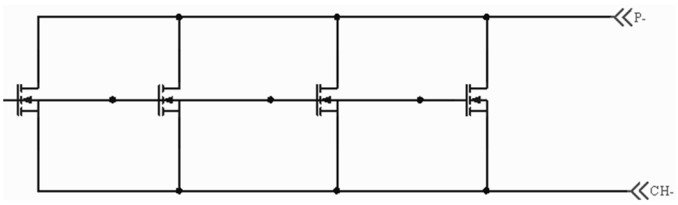


Fig. 5. Switching stage of the BMS

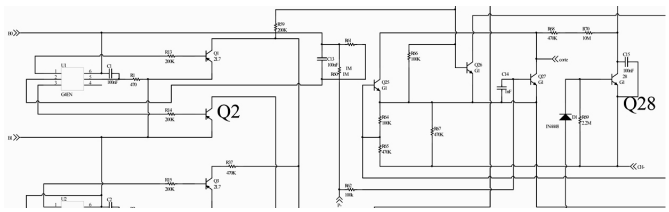


Fig. 6. Charge monitoring circuit

B. Discharge Phase

Fig. 7 shows the discharge phase of the BMS. The discharge occurs when the BMS is connected to a load, resulting in the discharge of cells. The function of this circuit is to monitor continuously the voltage of each cell and prevent these are below 2 V.

In the discharge process, something similar to the load one occurs: the transistor Q26 is controlled by the transistors associated to the supervision voltage of 2 V of the G4EN device, Q1 in this case. When the voltage is below 2 V, the transistor Q26 switch off the quartet of MOSFETs disposed between P-and B-, stopping the flow of the discharge current (Fig. 8).

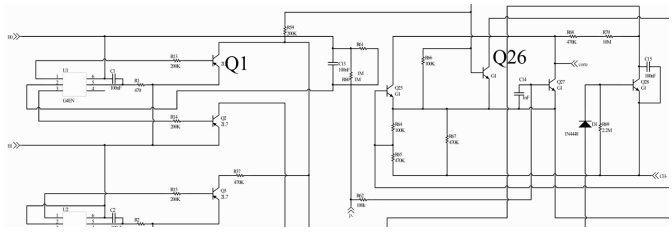


Fig. 7. Discharge supervisor circuit

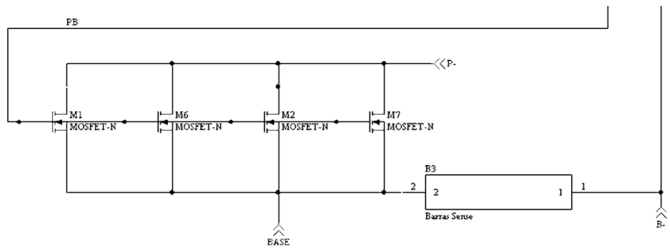


Fig. 8. The power MOSFETs in parallel with high current sensor bar

If a cell voltage is below 2 V, the load of the battery pack will be instantly disconnected, thus preventing any possible error in the process and reducing the inherent risk of destruction of the cell. As the integrated monitoring circuit has a hysteresis voltage of 2.2 V [5], the intermittent operation of the batteries is prevented.

As a protective measure, and in order to limit the current from the cells that flows by the discharge phase to the output, the BMS includes current monitoring bars. These bars consist of six steel strips between 24 and 30 mm connected in parallel. Together, they form a total resistance of 3.6 mΩ, so that taking into account the VBE = 0.5 V of the transistor G1, the sensed current will be about 140 A.

C. Burning Phase

Although the cells have been manufactured by the same producer under the same manufacturing process, these are different to each other. These are not been charged or discharged at the same speed, and their lifetime it is not equal. For this reason, the charging system performs a balancing operation thereof.

In the charge phase of the battery pack, a cell balancing process is performed in which, from a certain voltage, the excess of current of each cell is burnt.

This process is performed again and again until all cells are uniformly charged and they reach the maximum voltage value, which depends on the type of technology used. Fig. 9 shows the supervisor that monitors the reference voltage and the associated part for switching the burnt cells (Fig. 10).

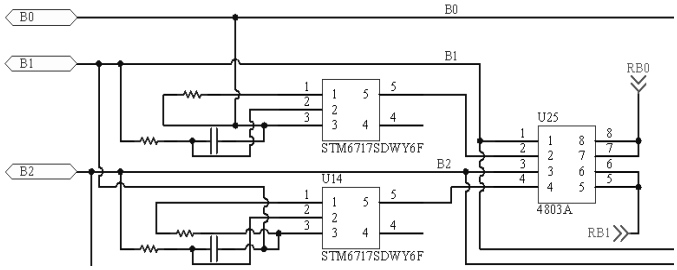


Fig. 9. Supervisor and switch burned

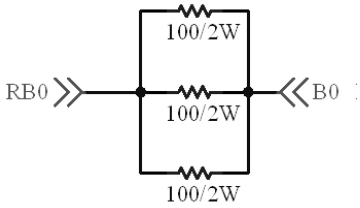


Fig. 10. The burnt cell

D. Protection Circuit against the Overtemperature

The temperature of the system, specifically the MOSFET transistors one, is controlled by a thermostat with an encapsulated TO 220. The associated circuit allows it to act both in the charge phase of the battery pack as in the discharge phase (Fig. 11). In this particular case, the monitored control temperature is 80 ^{0}C. The transistor Q29 acts on the two gates of the two MOSFETs lines, disconnecting both the charge and the discharge.

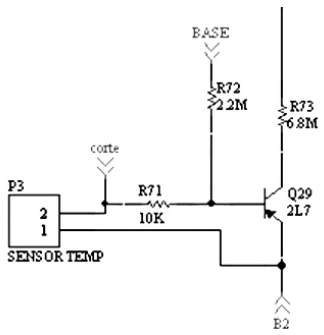


Fig. 11. Temperature sensing circuit

E. CAN BUS Interface

A completely analogic BMS does not provide the user with the status of the battery pack and the cells, or the remaining capacity of these. However, in a digital BMS, the user can get this information and monitors the charge and the discharge of the battery pack. For this reason, a CAN interface was implemented in order to provide the analogic BMS with some relevant characteristics of a digital one such as:

- Monitoring the voltage of each cell.
- Monitoring the voltage of the battery pack.
- Charging status of each cell (SOC).
- Health status of each cell (SOH).
- Battery Capacity.

This task is performed by a microcontroller, which adapts the output signals to the CAN bus protocol (Fig. 12).

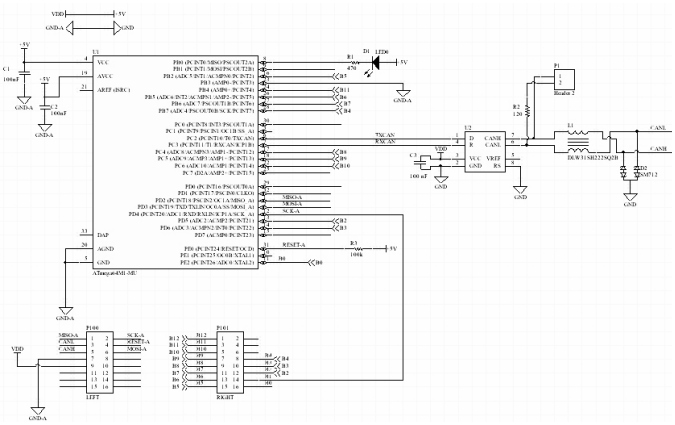


Fig. 12. Bus CAN schematic

IV. Designs of The Printed Circuit Boards

The BMS consists of three printed circuit boards:

- The BMS (Fig. 13).
- Burnt resistors (Fig. 14).
- The CAN bus (Fig. 15).

The design of the circuit schematics and the wiring diagram to form the copper tracks was carried out in two phases. First, according to the specifications for the functionality of the circuit, the components and the necessary interconnections between them were established. The wiring diagram was made by the schematic editor of the software. After, the mask, which is a virtual representation of the components on the plate, was defined by the PCB editor. Additionally, the physical form of the connections between the components was established too. To make an electrical or electronic circuit, the designer needs documentation, data sheets from manufacturers and support of computational tools.

A. Techniques for the design of printed circuit boards

At the process of development and manufacture of electronic prototypes, a previous design of the circuit board must be performed. For this, it is necessary to use special tools and apply theoretical and technical considerations, so that the developer can make and study in detail the models and electronic schemes to be implemented.

The parameters to be considered to design the board are as following:

- The resistive effect of tracks. The tracks must be designed taking into account the length, the thickness and the maximum current which they should lead. Software tools should be used to identify and calculate the dimensions.
- The thermal effect. The location of all components must be studied according to the interconnection and the thermal and electromagnetic interferences.
- The capacitive and inductive effect. The power section must be as far from the control section of the BMS as possible.
- In order to prevent electromagnetic disturbances (EMI) and provide electrical protection, the analogic plane must be separated from the digital one. Also, for this purpose, couplings such as optocouplers, RF isolators, transformers, etc. can be used.

All circuits were integrated under the Single European format, as shown in Fig. 16.

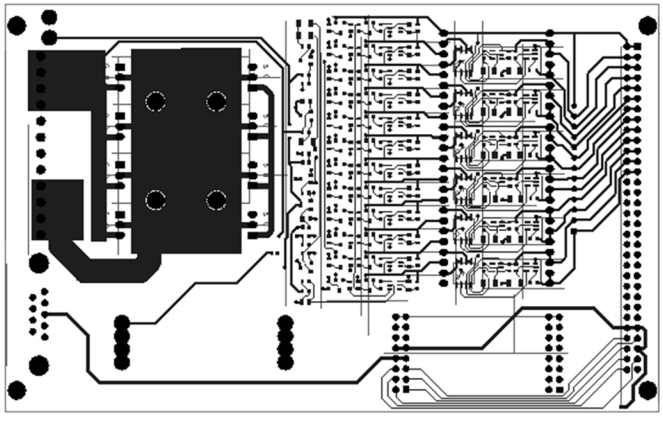


Fig. 13. BMS printed circuit board

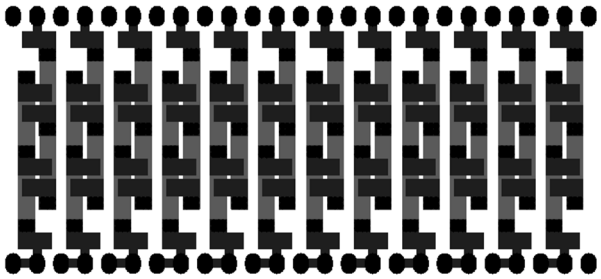


Fig. 14. Burnt printed circuit board

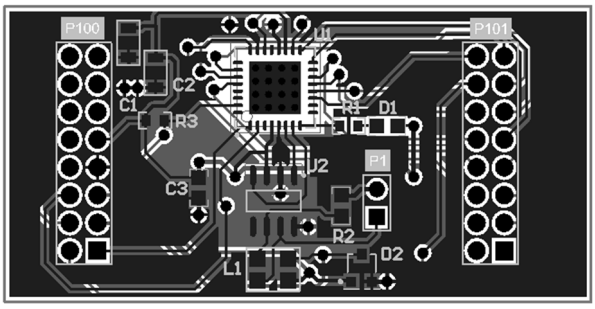


Fig. 15. CAN bus printed circuit board

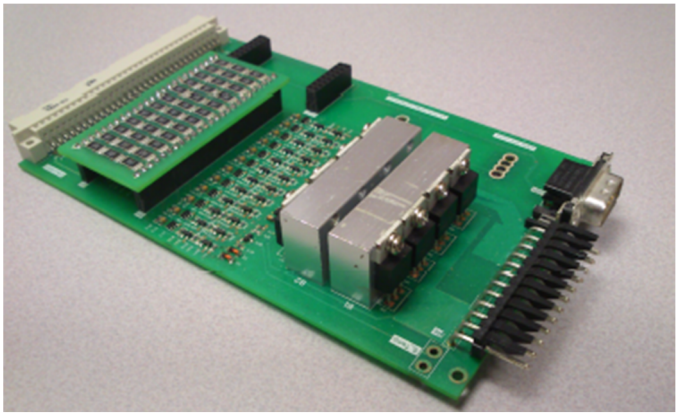


Fig. 16. View of full BMS

V. TESTS

In order to carry out the verification of the operation of BMS, a test bench was developed. By this, the status of all cells and the switching between states of charge and discharge, as well as the balancing on the charge phase, were constantly monitored.

A. *Discharge Test*

The right operation of the BMS is check by this test. The cells are discharged using a resistive power. If a cell voltage is below 2 V (the minimum cell voltage, $Vcell_{min}$), the system is deactivated and it will be reactivated when all cell voltages are above 2.2 V.

Table I shows a discharge process of the battery pack. When a cell voltage is below 2 V in a battery pack, the system is completely disconnected.

B. *Charge Test*

In this test, a current of 2 A is supplied to the system. The aim of this test is to check that when a cell voltage is 3.6 V (the burnt voltage, $V_{burning}$), the excess energy burning process stars, thus balancing the load. When a cell voltage is 4 V, its charge is disconnected, and when it is below 3.8 V, this will be reactivated.

Table II shows a charge process of the batteries through the implemented BMS. When a cell voltage is 4 V, the system is disconnected, but the process of burning the cell voltages that are below 3.6 V continues.

TABLE I. DISCHARGE PROCESS OF THE BATTERY PACK

BMS Status	**On**	**On**	**Off**
Time (min)	**0'**	**10'**	**70'**
Cell	**Vcell (V)**	**Vcell (V)**	**Vcell (V)**
B0	3,60	3,42	2,30
B1	3,55	3,38	2,28
B2	3,50	3,32	2,23
B3	3,60	3,40	2,29
B4	3,60	3,45	2,32
B5	3,60	3,42	2,30
B6	3,45	3,40	2,29
B7	3,55	3,35	2,25
B8	3,60	3,42	2,30
B9	3,50	3,38	2,28
B10	3,55	3,39	2,28
B11	3,60	3,40	2,29
B12	3,50	3,30	**1,98**
Battery pack	46,20	44,03	29,31

TABLE II. CHARGE PROCESS OF THE BATTERIES THROUGH THE BMS

Time (min)	**0'**	**10'**	**60'**	**70'**
Cell	**V_{cell} (V)**	**V_{cell} (V)**	**V_{cell} (V)**	**V_{cell} (V)**
B0	2,30	2,47	3,39	**4,01 Off**
B1	2,28	2,45	3,37	3,57
B2	2,23	2,40	3,32	3,52
B3	2,29	2,46	3,38	3,58
B4	2,32	2,60	**4,02 Off**	3,90
B5	2,30	2,49	3,39	3,59
B6	2,29	2,47	3,38	3,58
B7	2,25	2,46	3,34	3,54
B8	2,30	2,42	3,39	**4,01 Off**
B9	2,28	2,47	3,37	3,57
B10	2,28	2,45	3,37	**4,01 Off**
B11	2,29	2,45	3,38	3,58
B12	1,98	2,46	2,98	3,18
Battery pack	29,31	31,62	44,08	47,91

CONCLUSIONS AND ONGOING WORK

A version of the proposed BMS for charging the battery of an electric motorbike was made for a Master Thesis, thus showing how this kind of system performs. This project was implemented in the Power Electronics courses taught by the Department of Electrical Engineering and Automation in various degrees of the University of Las Palmas de Gran Canaria (ULPGC).

This BMS is being used in conjunction with a controller for a BLDC motor (Fig. 17) mounted on a test bench (Fig. 18).

Thanks to this design, students can acquire the knowledge and the skills necessary for the complete development of a Battery Management System. They can also analyse the initial specifications and determine the best topology to use in each case. Moreover, students learn to manufacture and assembly the PCBs, and to analyse the data.

Based on this project, a BMS digital will be developed in the future to facilitate the interconnection with other devices and perform the monitoring tasks by a microcontroller. This system allows to program different control signals according to the technology used to manufacture the lithium-ion cells.

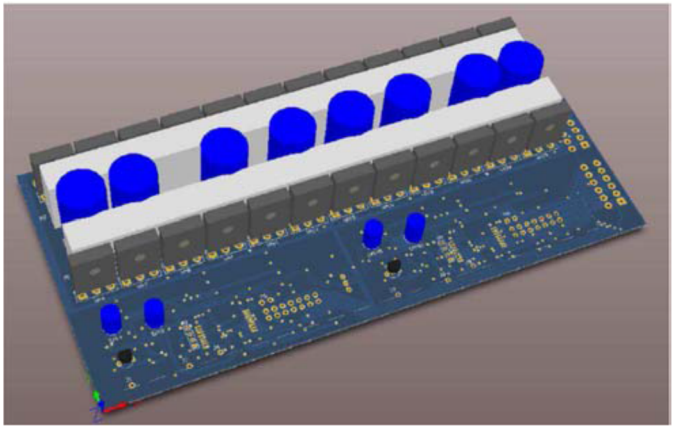


Fig. 17. 3D front view of the controller for 6-phase BLDC motors

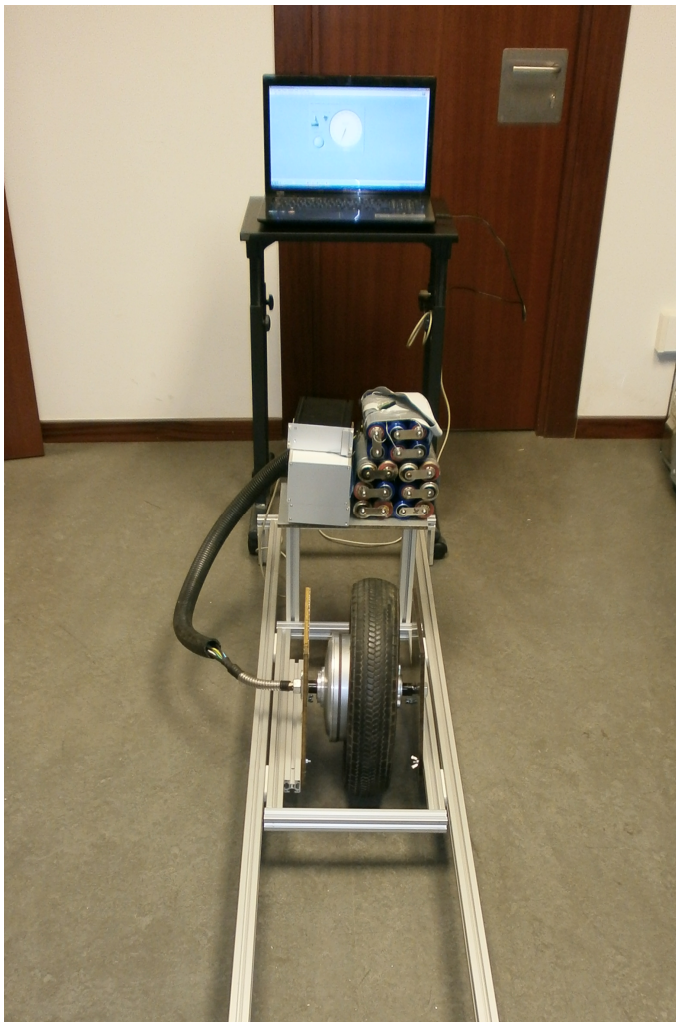


Fig. 18. Test bench of BLDC motors

REFERENCES

[1] Battery Management Systems for large lithium-ion battery pack. Davide Andrea, Artech House Inc; Edition: New. (September 1, 2010)

[2] Linden´s handbook of batteries. McGraw-Hill Professional; Thomas B. Reddy, 4 edition (October 27, 2010)

[3] Designing a New Generalized Battery Management System. John Chatzakis, Kostas Kalaitxakis, Nicholas C. Vulgaris, and Stefanons N. Manias, Senior Menber, IEEE Transactions on Industrial Electronics, Vol. 50, Nº. 5, October 2003

[4] Electrónica de Potencia. Daniel W. Hart, First edition. Prentice Hall.

[5] Battery Management System Based on Battery Nonlinear Dymanics Modeling, Antoni Szumanowski and Yuhua Chang. Vehicular Transaction On. May 2008

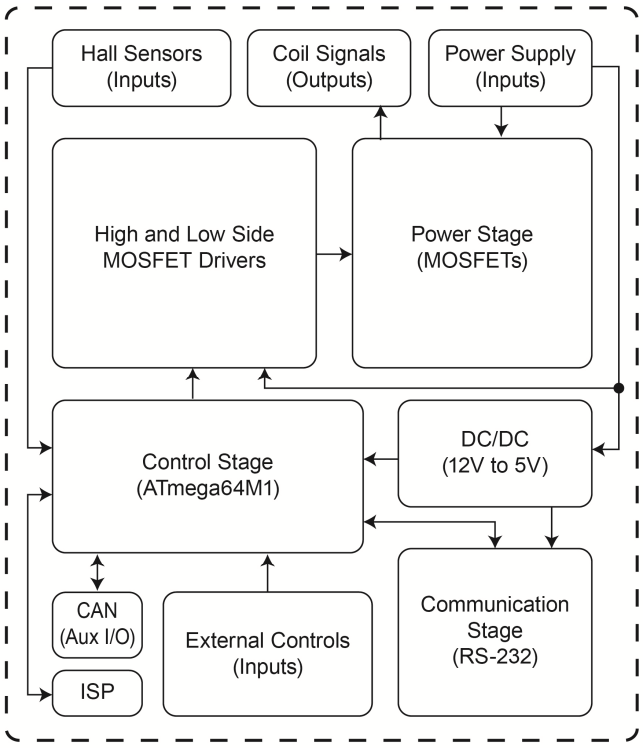


Fig. 1. Diagram of the parts of BLDCM controller

1) Control stage

In the control stage it is located one of the most important elements of the system: the µC ATmega64M1. It reads the values obtained from the Hall sensors of the BLDC motor. These sensors determine the position of the rotor. By reading these values, the µC generates a PWM signals with a frequency of 16 kHz and a peak voltage of 5 V. The PWM signals are sent to the drivers of the MOSFETs, located in the power stage, phase-shifted 120° from each other. This offset is established by the signals of the Hall sensors (Fig. 2).

2) Power stage

At this stage the MOSFET drivers are located. These drivers are supplied with a voltage of 12 V and raise the voltage of the PWM signals generated by the µC to that value. These signals attack the gates of the power MOSFET (IRFP4568), so that the supply voltage (Vcc) can pass to the BLDC motor. The voltage and the current that flow through the MOSFETs, according to the switching logic set up by the µC, excites the motor coils causing the motor rotate. The duty cycle of the PWM signals, set up by the speed adjust potentiometer, determine the speed of the motor.

Fig. 3 shows one of the three parts of this power stage. The motor coil can be found in 3 different states caused by the high/low driver signals:

- VCC: coil is connected to the supply voltage.
- GND: coil is connected to the ground.
- NC: coil is not connected.

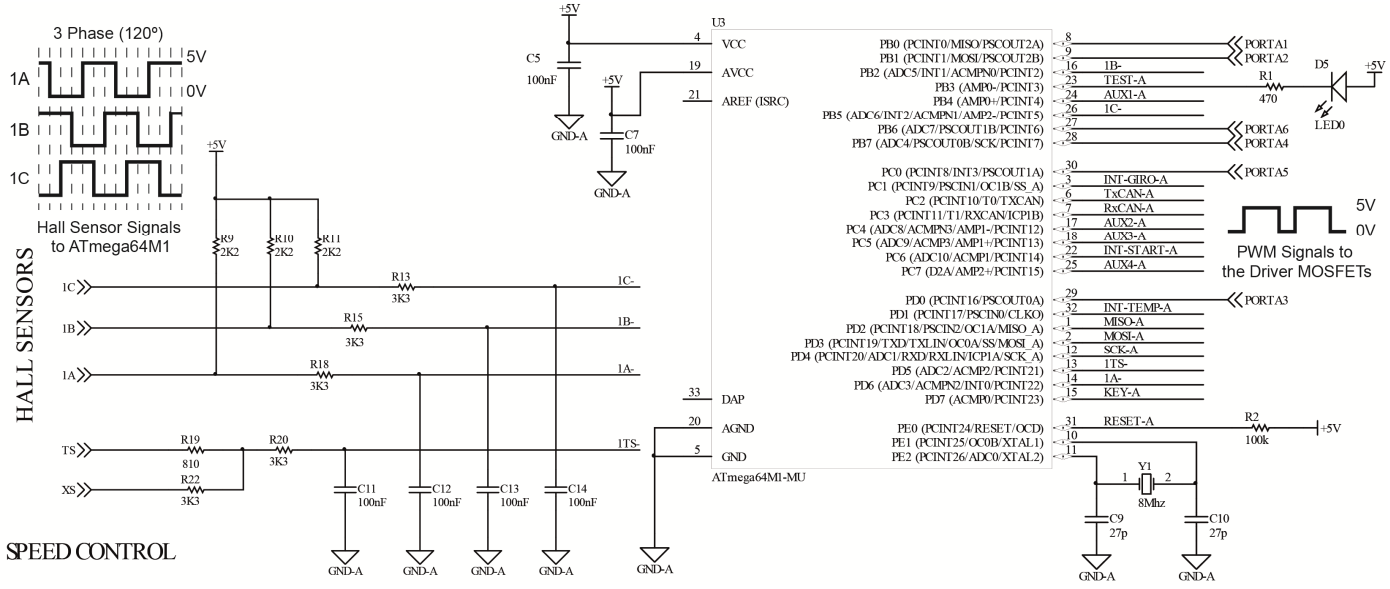


Fig. 2. Control stage of the BLDC controller

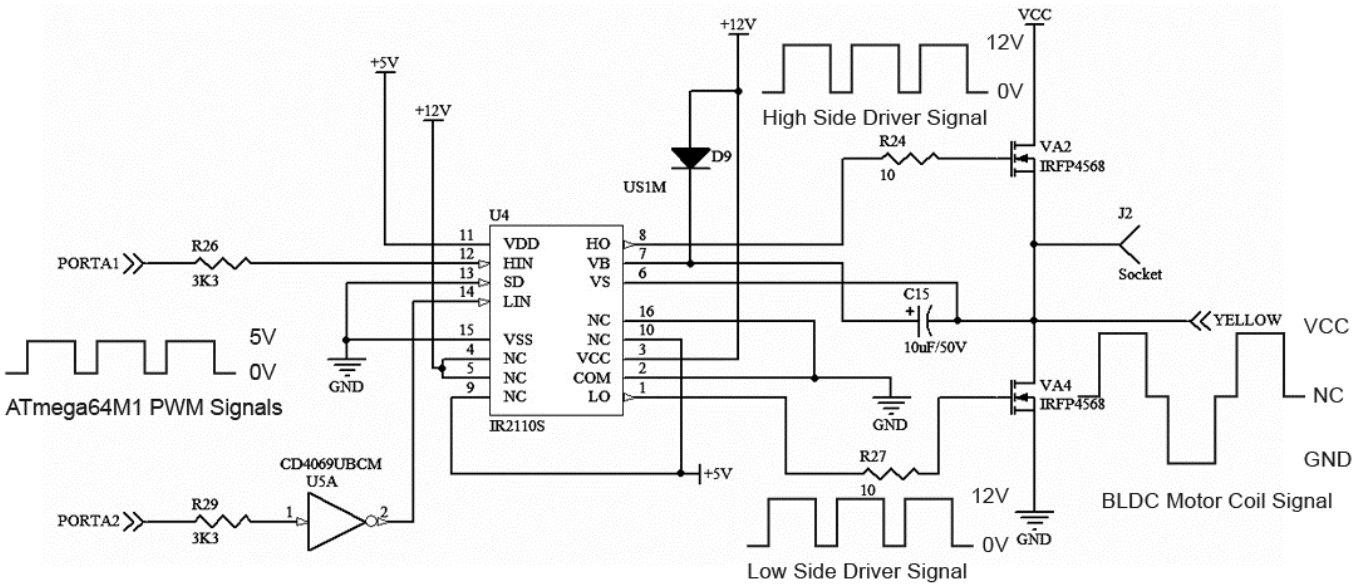


Fig. 3. One of the three parts of the power stage

3) Communication stage

On the other hand, the design of the controller has a communication stage in which the controller is connected to a workstation (PC or display) via a serial communication (RS-232). In addition, the µC enables to make this connection through the CAN protocol (Controller Area Network), which ensures greater security and robustness against interferences and collisions in data transmission. This is why, so we decide to have the possibility of also using this protocol.

Finally, at this stage note the part where the connections of external controls and sensors for the controller are performed. These controls and sensors may be:

- The ignition switch of the system.
- The motor speed adjustment potentiometer.
- The motor rotation direction switch.
- The motor temperature sensor.

In addition, the ISP (In-System Programming) connector is located in an accessible way to carry out the µC programming with the C library that has been developed to control this type of motors [5]. In this process, it is used the free software provided by the manufacturer of the µC (Atmel) from their website, the Atmel Studio 6.0 [6]. The programming of the µC is performed using this software and the AVRISP mkII programmer device [7] through the ISP interface.

B. Design and Manufacture of the PCB

The design and the manufacture of the PCB (Printed Circuit Board) controller have been performed at the ULPGC-IUMA-SFP Laboratory (Prototype Manufacturing Service of the Institute for Applied Microelectronics at the University of Las Palmas de Gran Canaria). We used the simple Eurocard format (100 mm x 160 mm), so that the controller can be connected in standard racks.

The PCB controller was designed using the Altium Designer software [8]. This is a 2-layer design with the component mounting in the top layer. Fig. 4 shows a 3D view of the final design of the prototype, developed under the simple Eurocard format. Likewise, to manufacture the PCB, a set of files known as Gerber Files are generated from the software (Fig. 5). These files are used to print the photolithos of each layer, in addition to the files used to drill the plate and to manufacture the solder mask.

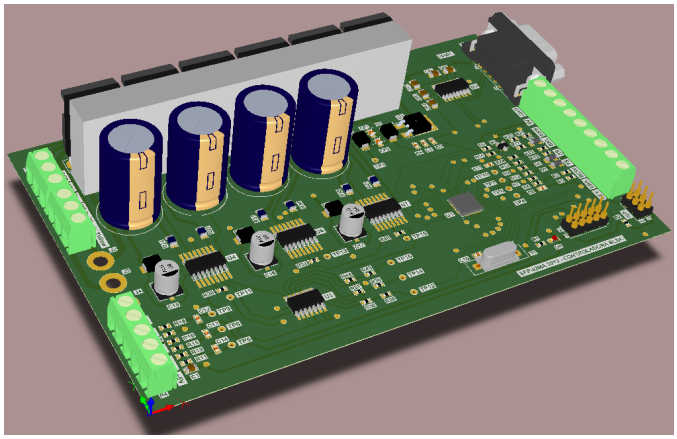


Fig. 4. 3D view of the BLDCM controller

Once the PCB manufacturing process is completed, the manually placement of the SMD (Surface Mount Devices) components is performed. This process is carried out by applying solder paste on the surface mount PADs of the components using a spatula. In order to apply the paste correctly, a welding screen is used. Subsequently, SMD components are placed using a manual Pick & Place machine.

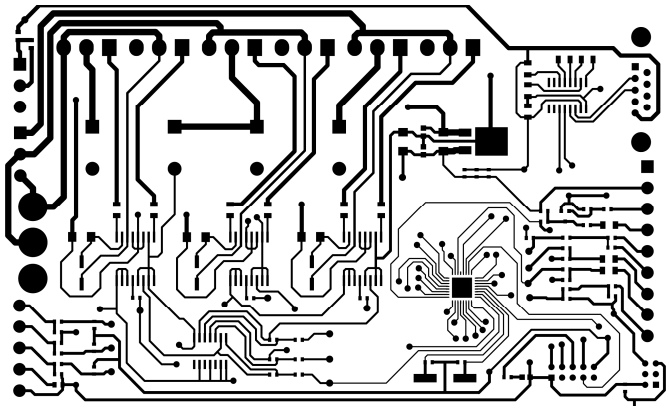


Fig. 5. Top layer gerber file of the BLDCM controller

When all components are placed in the right position, the PCB is introduced into a reflow oven to solder them. Finally, the through-hole components are soldered by hand and the PCB is placed in its enclosure (Fig. 6).

Because the 2-layer PCBs can be manufactured at the IUMA SFP Lab, the cost thereof is reduced, since you have only to spend on the materials and the components used. Moreover, these elements are commonly used and readily available. On the other hand, due to the modular design of the controller, it can be used in different applications. Thus, because of all these aspects, a low-cost BLDC motor controller is obtained.

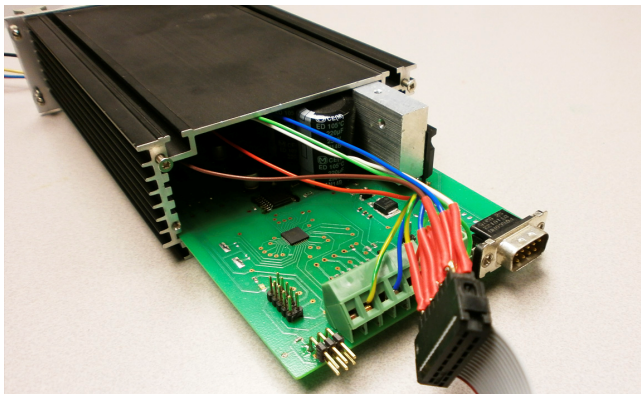


Fig. 6. Real view of the manufactured BLDCM controller

III. The Test Bench

From the design of this low-cost controller, 3 possible applications in which integrate it have been developed.

A. Basic control system

This control system of BLDC motors is a learning kit designed and manufactured in order to be used to unify microcontroller programming practices with its direct application to control BLDC motors (Fig. 7). For this purpose, has been built a test bench where it is placed a BLDC motor mechanically connected to a conventional DC motor via a transmission belt. By this system, measurements of the motor torque can be taken applying a back electromotive force through the DC motor. On the side of the test bench, the external controls of the controller (the speed adjustment potentiometer, the motor rotation direction switch and the ignition switch) are located.

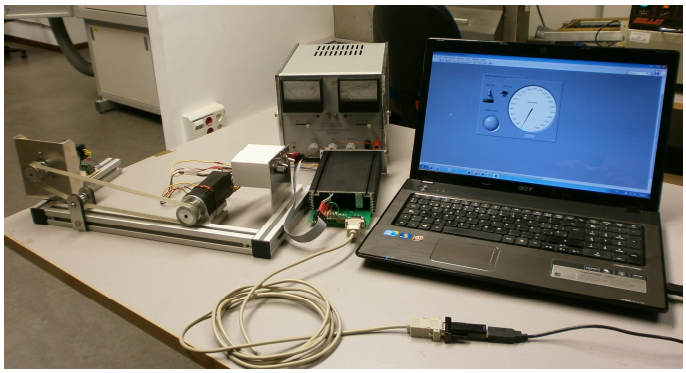


Fig. 7. View of the manufactured basic control system

Fig. 8 shows the interconnection of the elements of the control system for 3-phase BLDC motors. The motor sends the three Hall sensor signals to the controller, and this in turn sends the switching signals to the motor coils. These switching signals are sent with a duty cycle determined by the value of the speed adjustment potentiometer connected to the controller. In addition, there is an ignition switch that turns on the motor.

The controller and the motor are fed with a voltage of 24 V supplied by a Li-Ion battery pack which is managed by a BMS (Battery Management System). This analog BMS is part of another project developed by the IUMA [9]. The BMS monitor the state of a 7 cell battery pack (of 3.6 V each cell), ensuring the maximum durability thereof. In addition, it enables to make an accurate battery charging from a charger connected to the mains.

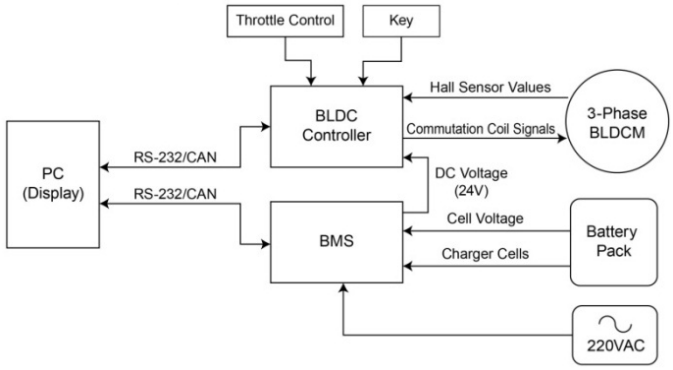


Fig. 8. Configuration of a basic BLDC motor control system

The controller and the BMS are connected to a workstation (PC) via serial under the RS-232 protocol. This communication, along with a program developed in LabView, allows monitoring the speed and the status of the motor, and the status of each cell in the battery pack.

In the proposed basic learning kit, the use of the battery pack and the BMS is optional. You can power the motor and the controller using a standard power supply that provides a 24 V of voltage and a minimum of 2 A of current to the motor

used in the trials. The reference number of the BLDC motor used is 42BL100. Its main characteristics are shown in Table I.

TABLE I: 42BL100 BLDC MOTOR DATASHEET

Nº of Pole	8
Nº of Phase	3
Rated Voltage (V)	24
Rated Speed (rpm)	4000
Rated Torque (Nm)	0,25
Max. Peak Torque (Nm)	0,75
Torque Constant (Nm/A)	0,036
Max. Peak Current (A)	20
Mass (kg)	0,8

B. Control system for electric vehicles

In this section, the configuration to perform a management system for electric vehicles is presented (Fig. 9). To make this system, two BLDC controllers are used with the goal of manage a 6-phase BLDC motor with a power of 5 kW.

This kind of BLDC motor is used in electric motorcycles and it is managed by two independent controllers but synchronized with each other through external controls (the speed adjustment potentiometer and the ignition switch) and the Hall sensors from the motor. Each controller excites three of the six motor coils according to the order predetermined by the software and the Hall sensor signals.

The power supply of this system is a battery pack of 96 V (26 cells of 3.6 V). The battery pack is managed by a BMS and both it and the two controllers are connected to a CAN bus. This protocol manages the communication between the boards and a display that shows the variables of the motor state and the battery pack. Furthermore, there is the possibility of connecting to the bus a computer for testing and monitoring the system.

With this control system, a prototype of a test bench applied to teaching in electromechanical practical sessions has been developed (Fig. 10). In this prototype the external controls have been replaced by software developed in LabVIEW, which enables the control of the motor from a PC.

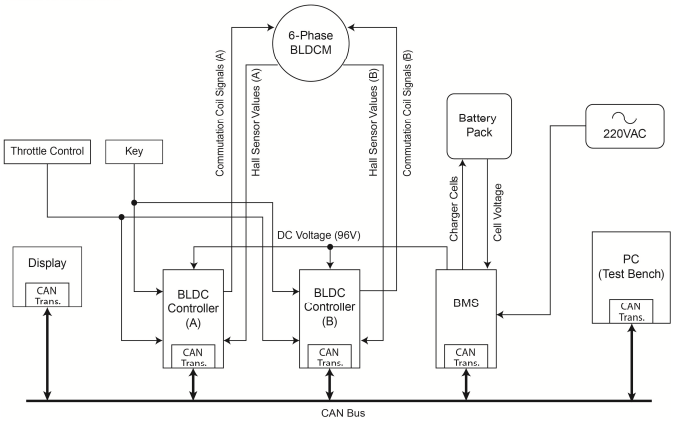


Fig. 9. Configuration of the control system for electric vehicles

It is noteworthy that it has been opened a new line of research to integrate the two controllers needed to control the 6-phase motors into a single controller. This controller has been specifically developed for use in motor management of electric motorcycles (Fig. 11).

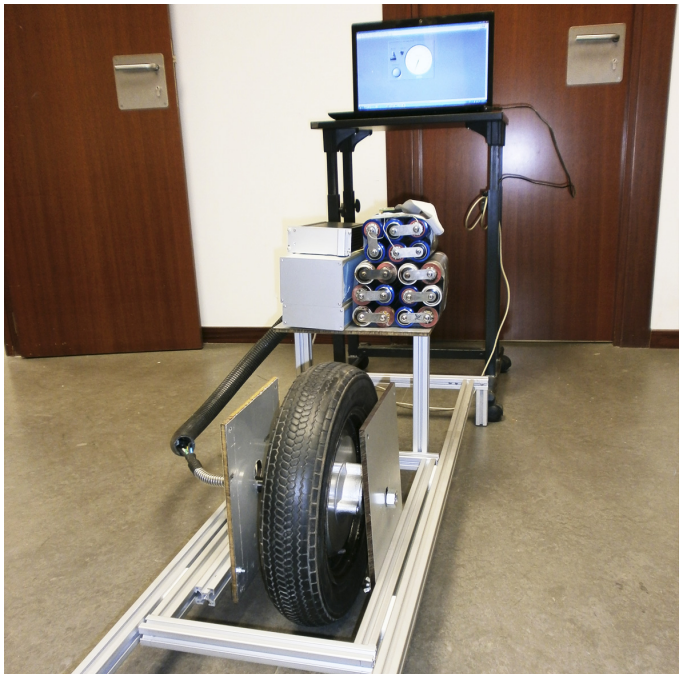


Fig. 10. View of the test bench of BLDCM control system for electric vehicles

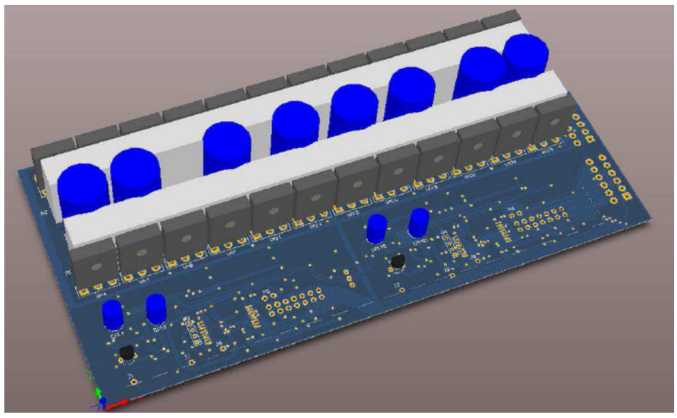


Fig. 11. 3D front view of the 6-phase BLDCM controller

C. ROVs control system

A project related to the ROVs, the AVORA project (Autonomous Vehicle for Operation and Research in Aquatic environments) is being currently developed in ULPGC by a team of students and researchers from this university in cooperation with PLOCAN (Canary Islands Oceanic Platform) [10]. The objective of the project is to design and manufacture electronics, mechanics and software of an autonomous underwater vehicle. With it the team will participate in the European competition SAUC-E 14 (Student Autonomous Underwater Vehicle Challenge - Europe), a prestigious event in the field of underwater robotics, which is annually held since

2010 at the RSMC (Center for Maritime Research and experimentation) at Nato Underwater Research Center in La Spezia, Italy.

The propulsion system that is used by the AVORA ROV consists of four thruster DC motors with brushes. The control system of these motors is based on two control boards (RoboClaw Boards). Each one controls two motors. On the other hand, an Arduino Mega board controls the RoboClaw boards and communicates with the CPU (Central Processing Unit) via Ethernet. This CPU is connected via WIFI to the control station for receive the necessary commands when the robot is on the water surface (Fig. 12). When the robot is submerged, it will be completely autonomous.

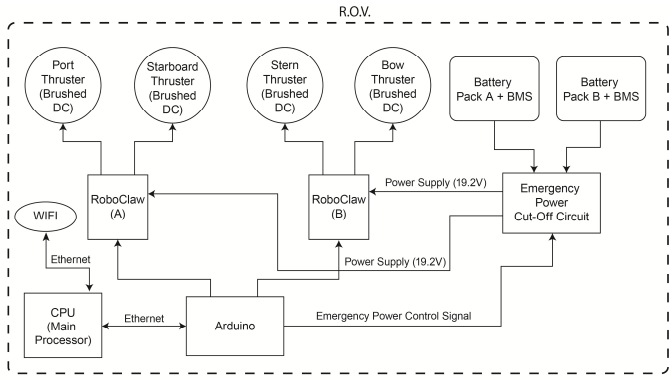


Fig. 12. Configuration of the current AVORA ROV control system

A future goal within this project is replacing the current propulsion system for a system based on BLDC motors. Thus, a greater efficiency of the vehicle, apart from the reduction in size and weight thereof, would be got. It is for this reason that we design a prototype of a ROV control system with BLDC motors based on the low-cost controller proposed in this paper.

This system is based on four 3-phase BLDC motors (Fig. 13). Each motor has a separate controller connected to a CAN bus. Also, as in the previous configurations, the system includes a battery pack that supplies the required voltage for the operation of the motors. This pack is managed by a BMS also connected to the CAN bus.

As in the current control system of ROV, a CPU is used to manage the system. The CPU will be integrated into the vehicle and it will receive the configuration commands via WIFI when it is in the water surface. In addition, the ability to control the ROV in real time using a remote control or PC will be implemented. This remote controller will be connected to the vehicle through an umbilical cable that will transmit data using the CAN protocol. The data transmission speed may vary between 500 kbps and 10 kbps according to the length of the cable used.

Another goal within this project in the future is redesigning the PCB controller to integrate these four independent controllers into a single control board for the four motors. Thus, a reduction in the size thereof would be obtained, and consequently, a decrease in size and weight of the vehicle.

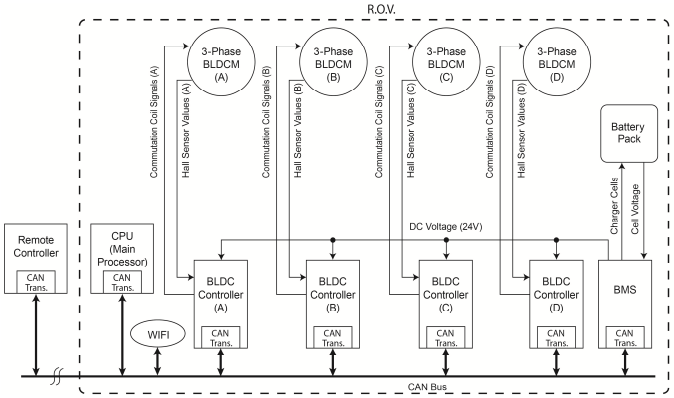


Fig. 13. ROV Control system with BLDC motors

CONCLUSIONS

The goal of this project is that students of electronics, electromechanical and robotic know both the software and hardware elements of a BLDC motors control system and their operation. By the use of the controller and test benches proposed in this paper, we believe that the teaching of this field can be successful and engaging for both the student and the teacher. Moreover, thanks to the RS-232 and CAN system connections, these control systems for teaching can be also used to teach the communication protocols.

At this moment, multiple controllers for use in practice at the next academic year 2014/2015 are being performed.

REFERENCES

[1] Atmel ATmega16/32/64M1 microcontroller's family: http://www.atmel.com/devices/ATMEGA64M1.aspx, Last accessed 2013, December.

[2] Wang Dongmei, Guo Haiyan, and Yu Jing, "Modeling and simulation research of brushless DC motor open-loop speed-adjustment system", The 2nd International Conference on Intelligent Control and Information Processing, ISBN 978-1-4577-0816-9, pp: 394 – 398, 2011.

[3] Alphonsa Roslin Paul, and Prof. Mary George, "Brushless DC motor control using digital PWM techniques", Proceedings of 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies (ICSCCN 2011), ISBN 978-1-61284-653-8, pp: 733 – 738, 2011.

[4] K. R. Rajagopal, and Ajay Nair, "Design and development of a TMS320F2812 DSP controller based PM BLDC motor drive", International Conference on Electrical Machines and Systems (ICEMS), ISBN 978-1-4244-7720-3, pp: 776 – 780, 2010.

[5] H. A. Fabelo, A. Vega, J. M. Cabrera, and V. Déniz, "Librería C para el control de un motor BLDC basado en un Atmega64M1 para docencia", XI TAEE, 2014.

[6] Atmel Studio 6.0 Manual. [Online]. Available: http://atmel.no/webdoc/atmelstudio/, Last accessed 2013, December.

[7] Atmel AVRISP mkII microcontroller programmer: http://www.atmel.com/tools/avrispmkii.aspx, Last accessed 2013, December.

[8] Altium Designer Software: http://www.altium.com/en/products/altium-designer/overview, Last accessed 2013, December.

[9] V. Déniz, A. Vega, and J. M. Cabrera, "Diseño de un sistema de gestión de baterías Li-Ion", PFC, University of Las Palmas de Gran Canaria, 2013.

[10] A. Mahtani, L. Sanchez, A. Martínez, D. García, D. Morales, E. Fernandez, F. Maniscalco, and J. Cabrera, "AVORA I : The SAUC-E'12 challenge", The C Continuum, 2012.

change occurs. In the main process, it disables or enables the PSC module. This module generates the 6 PWM (Pulse Width Modulation) signals, from the Q1 to the Q6, which attack the drivers of MOSFETs transistors. Then these transistors excite the BLDC motor coils.

The motor speed adjustment is performed by a digitized analog signal through one of the ADC (Analog to Digital Converter) channels. This digitalized signal establishes the duty cycle of the PWM signal used by the PSC module. This PWM signal of 16 kHz is generated from the 32 MHz PLL (Phase-Locked Loop) inside the µC.

Moreover, there are both the CAN (Controller Area Network) and the UART (Universal Asynchronous Receiver and Transmitter serial) communication modules with their respective transmit and receive data signals.

A. Main Process

This process performs the initialization of variables and the necessary hardware of the microcontroller. By *micro_modules_initialization()* function, the initial configuration of input and output ports of the µC is realized and the following functions are executed:

- *adc_initialization()*: It configures and enables the ADC module to read the analog speed adjustment potentiometer by the ADC2.
- *timer1_initialization()*: It initializes the timer 1 to perform the task of sampling speed and the PWM duty cycle adjustment. The compare match interrupt is enabled to run every 256 microseconds.
- *timer0_initialization()*: It configures the timer 0 as a counter for the motor speed measurement. It enables the overflow interrupt to convert the 8-bit timer in a 16-bit timer by an auxiliary variable that is incremented each time the interrupt is executed. Thus, higher accuracy in the motor speed measurement is obtained.
- *uart_initialization()*: It sets the µC UART module as transmitter and receiver. It uses the parameters set up by the user in the configuration file.
- *start_pll_32mhz()* y *wait_pll_ready()*: Functions that configures and activates the PLL to a frequency of 32 MHz. Once they are activated, they keep waiting until the PLL is ready to continue executing the program.
- *psc_initialization()*: It makes the PSC module configuration according to the parameters detailed in the user configuration file.
- *external_interrupt_initialization()*: It enables the external interrupts used by the Hall sensors (INT1, 2 and 3). Also, it enables the pin change interrupts for the ignition switch and the motor rotation direction switch (if this option is enabled).

When the system has been initialized, the *adc_launch()* function is executed every 256 µs in an infinite loop. This function reads the speed adjustment potentiometer. The value is stored and used to set the motor reference speed. By this reference speed, the control loop is running in open loop to set the value of the PWM duty cycle.

When the value of the PWM duty cycle has been obtained, the PSC module registers are updated with that value. After, the value of the measured speed is sent through the UART module for its display on the monitoring software (Fig. 2).

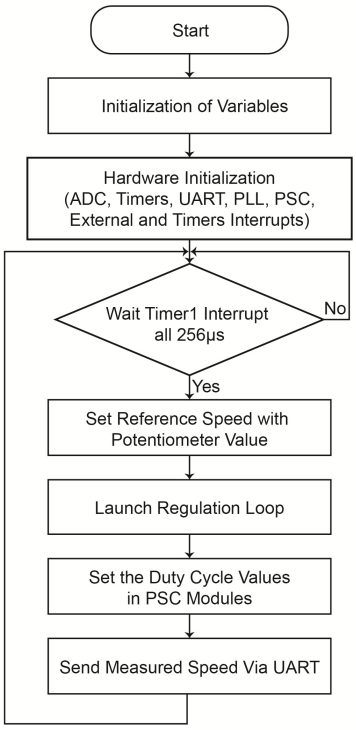


Fig. 2. Flow diagram of the main process

B. Hall Sensors Interrupts

These interrupts detect the rising edges of the Hall sensors signals. They are the highest priority level external interrupts of the program. In each one, the *output_psc_switch_commutations (value)* function is executed. This function requires as parameter the value at that time of the Hall sensors properly formatted. Using this value and the rotation direction variable at that time, the activation of the corresponding Q outputs of the PSC module is performed. These PWM output signals make the switching of the BLDC motor coils and their duty cycle is set by the speed adjustment potentiometer.

Fig. 3 shows the switching sequence of PSC module outputs according to the Hall sensor signals when the rotation is CW (*ClockWise*). The three possible states in which the U, V and W motor coils can be found are the following:

- VCC: coil is connected to the supply voltage.
- GND: coil is connected to the ground.
- NC: coil is not connected.

Table I details the switching sequence of PSC outputs in relation to the value of the Hall sensors and the CCW (*CounterClockWise*) or CW rotation motor.

In the A Hall sensor interrupt (INT 0) the calculation of revolutions per minute of the motor for each rising edge is performed by the *calculate_estimated_speed ()* function. This function uses the 16-bit value obtained from the timer 0 and a constant, which is defined by the user based on the motor used, to calculate the revolutions per minute of the motor (Fig. 4).

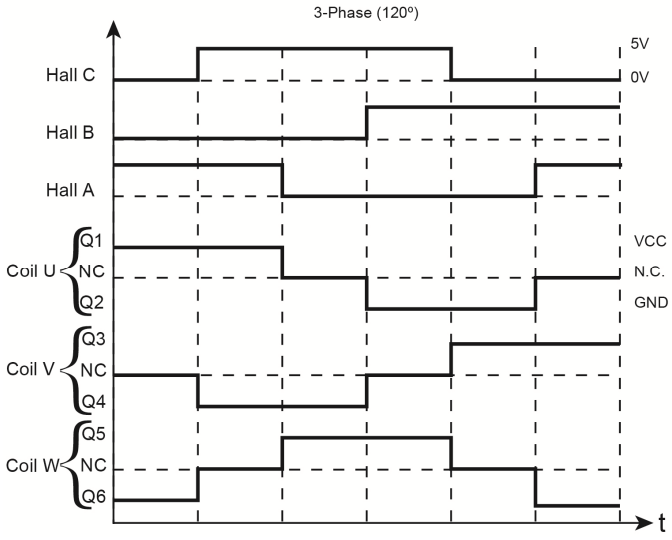


Fig. 3. Hall sensors signals and PSC outputs for CW rotation

TABLE I. SWITCHING SEQUENCE OF PSC OUTPUTS

Hall Sensors Value (C,B,A)	PSC Outputs Active (CCW)	PSC Outputs Active (CW)
001	Q5 – Q2	Q1 – Q6
101	Q3 – Q2	Q1 – Q4
100	Q3 – Q6	Q5 – Q4
110	Q1 – Q6	Q5 – Q2
010	Q1 – Q4	Q3 – Q2
011	Q5 – Q4	Q3 – Q6

C. ADC interrupt

This interrupt reads of the speed adjustment potentiometer and converts the value obtained to a digital value (Fig. 5). This interrupt is executed when the potentiometer signal conversion connected to the analog input channel 2 (ADC2) is finalized.

D. Timer 0 overflow interrupt

As already explained above, this interrupt increases the value of auxiliary variable that converts the 8-bit timer 0 in a 16-bit timer (Fig. 6). If the auxiliary variable is above than a calculated value, then the variable and the value of the reference speed are reset. If the variable that indicates whether the motor is active has a true value, the motor is tried to run again. To do this, it is used the *retry_run_motor()* function, which launches the control loop, updates the value of the PWM duty cycle and runs the activation function of the PSC outputs.

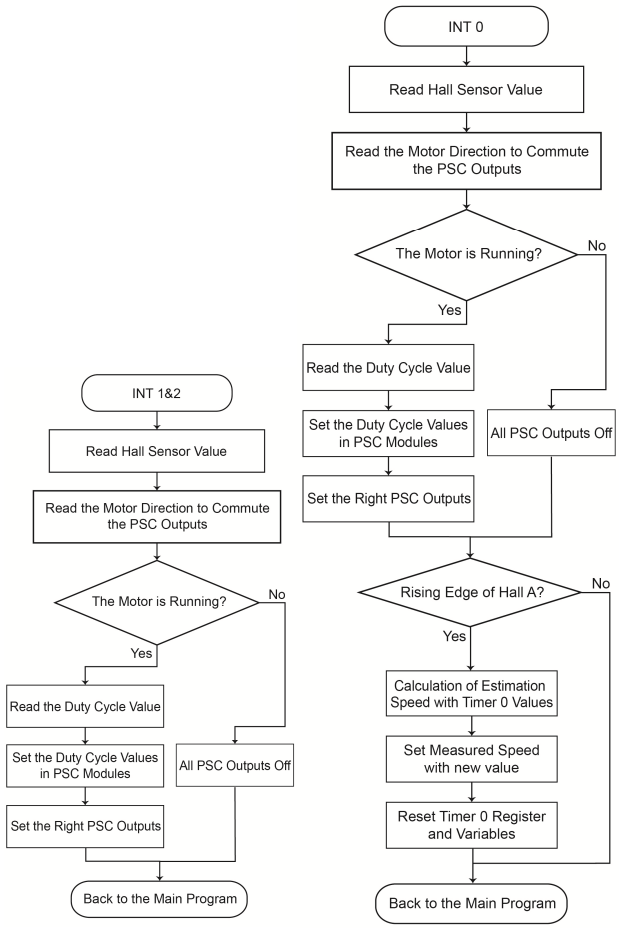


Fig. 4. Flow diagram of the Hall sensors interrupts

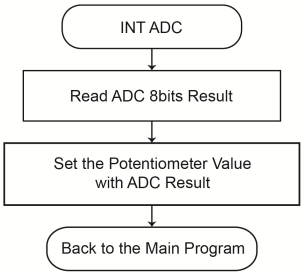


Fig. 5. Flow diagram of the ADC interrupt

E. Pin change interrupt 1 (PCINT 1)

It controls the starting and the stopping of the motor. When the interrupt detects a change in the signal level from the ignition switch, it checks if the level is high or low. When a high level is present on the pin, all PSC outputs are disabled. On the contrary, when a low level is present, the *output_psc_switch_commutations(value)* function is called with the value of the Hall sensors in that instant (Fig. 7).

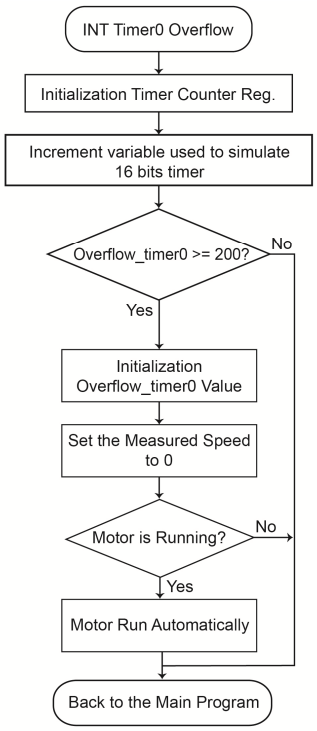


Fig. 6. Flow diagram of the Timer 0 overflow interrupt

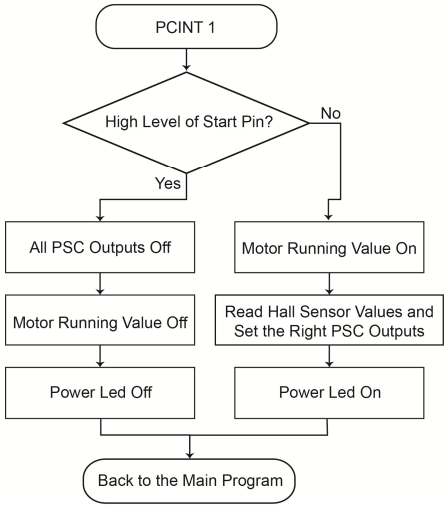


Fig. 7. Flow diagram of start and stop motor interrupt

F. *Configuration file*

In this file there are the constants that the users must set depending on their needs. For the UART module, they must set the baud rate, the number of stop bits, and the number of bits to transmit that they will use in the serial connection to the PC.

Another aspect that must be configured is the speed constant of the motor. This constant is calculated using the equation (1). This equation depends on the value, in seconds, set for the timer 0 (*t_timer0*), the maximum speed (*max_speed*), in revolutions per minute, and the number of pairs of poles of the BLDC motor used.

$$speed\_const = \frac{60 \cdot 255}{n \cdot t\_timer0_{(s)} \cdot \max\_speed_{(rpm)}} \quad (1)$$

A limited data in a range of 0 to 255 (8 bits) from the measured value of the motor speed is got by this constant. This data will be the value sent to the monitoring software to display the motor speed.

G. *ATmega64M1 programming*

Fig. 8 shows the flash memory map of the ATmega64M1 µC used in the project. The application code occupies a memory space of 10.8 kB. It is observed that there is clearance in memory for future expansion of the project.

Microcontroller programming is done by the Atmel Studio 6.0 software [6]. This software can be downloaded for free from the µC manufacturer's web and it is used to develop and compile codes made in C/C++ or assembly language.

A low-cost BLDC motor controller has been developed to carry out the tests in this project [7]. The application testing has been performed by this controller (Fig. 9). On the other hand, the load of the program has been carried out through the ISP (In-System Programming) interface of the µC and the AVRISP mkII programmer device [8].

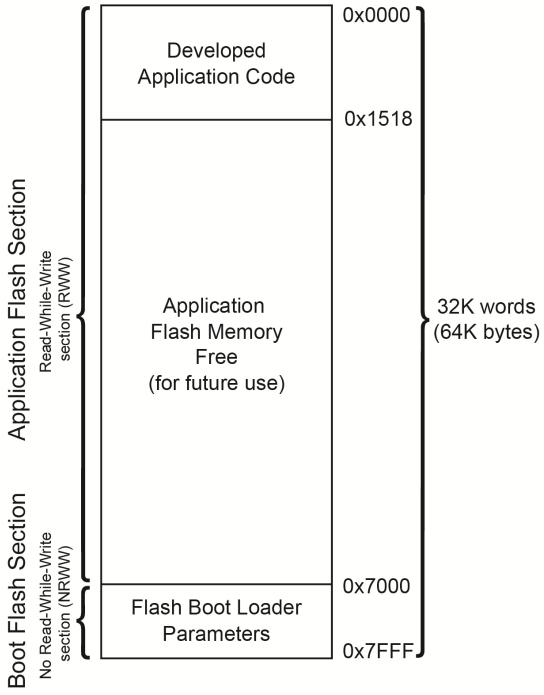


Fig. 8. Flash memory map of the ATmega64M1

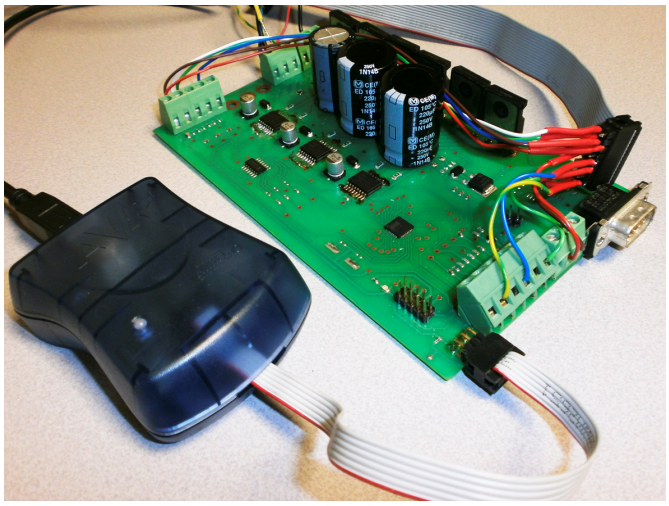


Fig. 9. View of the BLDCM controller and the AVRISP mkII programmer

III. Control and Monitoring Software

In this section, the developed software, using NI LabVIEW [9], for the control and monitoring of the BLDC motor from a PC is described. The communication protocol used is the RS-232. The software consists of two parts: a control and monitoring panel of the BLDC motor and a configuration panel of the UART connection and the constant for calculating the motor speed in revolutions per minute.

A. Control and monitoring

In this panel there are the controls and the display of revolutions per minute of the motor (Fig. 10). It has controls on and off (ON / OFF), change of direction of rotation (CCW / CW) and a virtual potentiometer for speed adjustment. These virtual controls have been incorporated to give the possibility to control the motor from the application. The configuration panel has a switch that enables these controls. If this switch is set, the physical controls of the test bench will be locked.

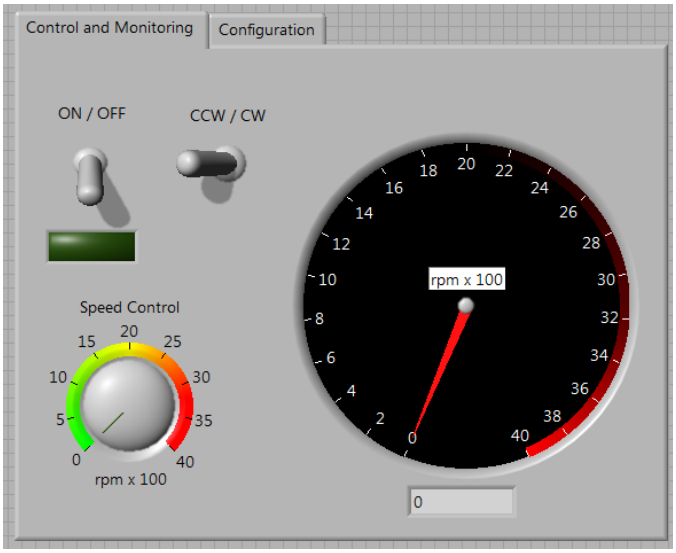


Fig. 10. View of the control and monitoring panel

B. UART configuration

This panel shows the controls to configure the serial port connection of the PC to the µC (Fig. 11). The parameters to configure this protocol are the following:

- Serial port used on the PC.
- Baudrate used in the connection.
- Number of bits per packet in the connection.
- Type of parity.
- Number of stop bits.
- Flow control used in the connection.
- Start transmission character (*XON*).
- Stop transmission character (*XOFF*).
- End of frame character.
- Maximum wait time in milliseconds (*Timeout*).

Furthermore, there is a field where must be entered the *alpha* constant used to find the real value of the motor speed. This constant is calculated by the expression (2), where *n* is the number of pairs of poles of the BLDC motor, *speed_const* is the constant calculated above and *t_timer0* is the timer0 value in seconds.

$$alpha = \frac{60}{n \cdot speed\_const \cdot t\_timer0_{(s)}} \quad (2)$$

The real speed of the BLDC motor is calculated by the expression (3), where *alpha* is the constant obtained in equation (2) and *measured_speed* is the 8-bit value of the measured speed received by the serial port.

$$real\_speed_{(rpm)} = alpha \cdot measured\_speed \quad (3)$$

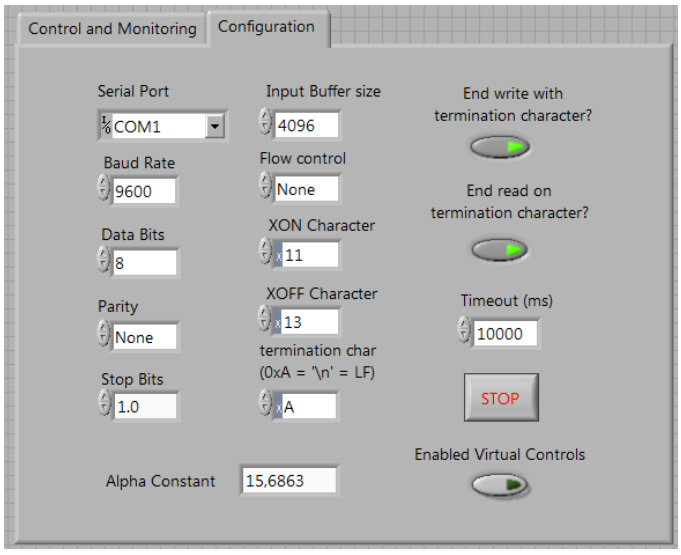


Fig. 11. View of the configuration panel

CONCLUSIONS AND ONGOING WORK

Nowadays this type of BLDC motors are increasingly used due to their high performance and excellent features, replacing traditional DC motors (Brushed Direct Current Motors). Hence, students must acquire knowledge in this field. By this system, they can learn to make mathematical calculations of the BLDC motor control and, moreover, settings and calculations to program the μC simultaneously. In this way, students will be more motivated and get a greater benefit from the practices.

A possible future action to expand this project is implementing the functions for speed and current loop control of BLDC motors. Thus, students will be able to learn everything about the PID control (Proportional Integral Derivative controller) of a BLDC motor.

On the other hand, the basic functions of the CAN protocol may be used in future as communication system for control and monitoring of the motor. This protocol is more robust and reliable than the RS-232 one. Students will be able to do internships and understand the operation of this communication bus by implementing the necessary functions to configure the CAN module of the microcontroller. After, the application layer would be added to the project using CANopen [10].

Multiple controllers are being currently manufactured for use in practice the next academic year 2014/2015.

REFERENCES

[1] Atmel ATmega16/32/64M1 microcontroller's family: http://www.atmel.com/devices/ATMEGA64M1.aspx, Last accessed 2013, December.

[2] Wang Dongmei, Guo Haiyan, and Yu Jing, "Modeling and Simulation Research of Brushless DC Motor open-loop Speed-adjustment System", The 2nd International Conference on Intelligent Control and Information Processing, ISBN 978-1-4577-0816-9, pp.: 394 – 398, 2011.

[3] Alphonsa Roslin Paul, and Prof. Mary George, "Brushless DC motor control using digital PWM techniques", Proceedings of 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies (ICSCCN 2011), ISBN 978-1-61284-653-8, pp.: 733 – 738, 2011.

[4] Hai-tao Wang, Ze Zhang, and Xiang-yu Liu, "Design of control system for brushless DC motor based on TMS320F28335", Third International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), ISBN 978-1-4244-9010-3, pp.: 954 – 958, 2011.

[5] Radu Duma, Petru Dobra, Mirela Dobra, and Ioan Valentin Sita, "Low cost embedded solution for BLDC motor control", 15th International Conference on System Theory, Control, and Computing (ICSTCC), ISBN 978-1-4577-1173-2, pp.: 1 – 6, 2011.

[6] Atmel Studio 6.0 Manual. [Online]. Available: http://atmel.no/webdoc/atmelstudio/

[7] H. A. Fabelo, J. M. Cabrera, A. Vega, and V. Déniz, "A Low-Cost Control System for BLDC Motors Applied to Teaching", XI TAEE, 2014.

[8] Atmel AVRISP mkII microcontroller programmer: http://www.atmel.com/tools/avrispmkii.aspx, Last accessed 2013, December.

[9] NI LabVIEW: http://www.ni.com/labview/esa/, Last accessed 2013, December.

[10] CANOpen protocol: http://www.can-cia.org/, Last accessed 2013, December.