Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

Máster en Ingeniería Industrial

MASTER EN INGENIERÍA INDUSTRIAL

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

TRABAJO FIN DE MÁSTER

Programación de Arduino mediante MATLAB/Simulink. Aplicación al control de velocidad de motores BLDC

Autor: D. Diego Cuervo Fernández

Tutor: D. Luis Carlos Herrero de Lucas

Valladolid, Junio, 2016

Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

Máster en Ingeniería Industrial

MASTER EN INGENIERÍA INDUSTRIAL

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

TRABAJO FIN DE MÁSTER

Programación de Arduino mediante MATLAB/Simulink. Aplicación al control de velocidad de motores BLDC

Autor: D. Diego Cuervo Fernández

Tutor: D. Luis Carlos Herrero de Lucas

Valladolid, Junio, 2016

Palabras clave

Motores BLDC, Arduino, MATLAB/Simulink, control Hall, control *sensorless*

Resumen

Los motores de tipo BLDC están ganando popularidad en el campo de la movilidad eléctrica debido a sus prestaciones superiores en comparación con otras tipologías. Estos motores se controlan mediante conmutación electrónica, para lo que pueden utilizarse sensores magnéticos tipo Hall o realizar el control sensorless midiendo las tensiones en las fases. En el presente proyecto, se han modelado y simulado ambos tipos de controles utilizando MATLAB/Simulink, y se ha aprovechado la herramienta de programación gráfica de la que dispone este software para realizar la implementación física del sistema sobre la plataforma Arduino. Finalmente, se han realizado pruebas en el laboratorio, extrayendo las conclusiones oportunas.

Keywords

BLDC motors, Arduino, MATLAB/Simulink, Hall control, Sensorless control

Abstract

BLDC motors are increasing their popularity in electric mobility sector because of their higher performance when they are compared with other typologies. These motors are controlled by electronic commutation, for what can be used Hall magnetic sensors or sensorless control, based in the measurement of phase voltages. In this project, the two types of control have been modeled and simulated with MATLAB/Simulink, and we have used the graphic programming tool of this software in order to implement the physic system on the Arduino platform. Finally, some tests have been made in the laboratory, drawing the appropriate conclusions.

<u>Agradecimientos:</u>

A todas las personas que han hecho posible que este proyecto sea una realidad, muchas gracias

“Si buscas resultados diferentes,

no hagas siempre lo mismo”

Albert Einstein

1. Introducción, objetivos y justificación

1.1. Introducción

La movilidad sostenible es un concepto nacido de la preocupación por los problemas medioambientales y sociales ocasionados por la generalización del uso del vehículo particular propulsado mediante combustibles fósiles como medio de transporte.

Los inconvenientes de este modelo, entre los que destacan la contaminación del aire, el consumo excesivo de energía, los efectos sobre la salud o la saturación de las vías de circulación; han generado una preocupación creciente por encontrar alternativas que ayuden a evitar o minimizar los efectos negativos de este modelo y encontrar uno nuevo.

El transporte representa la cuarta parte de emisiones de gases de efecto invernadero y el 36% del consumo de energía en España.

Se entiende por actuaciones de movilidad sostenible todas aquellas que ayudan a reducir dichos efectos negativos, ya sean prácticas de movilidad responsable por parte de personas sensibilizadas con estos problemas (desplazarse a pie, en bicicleta o en transporte público en lugar de en coche siempre que sea posible, compartir un coche entre varios compañeros para acudir al trabajo, etc.) como el desarrollo de tecnologías que amplíen las opciones de movilidad sostenible por parte de empresas o decisiones de las administraciones u otros agentes sociales para sensibilizar a la población o promover dichas prácticas.

A menudo el concepto de movilidad sostenible se vincula a las nuevas tecnologías desarrolladas en el sector de la automoción a lo largo de las últimas décadas para reducir las emisiones de CO_2 a la atmósfera. Los principales objetivos en los que se centra la implantación de la movilidad sostenible en la sociedad actual son:

- Configurar un modelo de transporte más eficiente para mejorar la competitividad del sistema productivo.
- Mejorar la integración social y accesibilidad de los ciudadanos.
- Incrementar la calidad de vida de las personas.
- No comprometer las condiciones de salud pública.
- Aportar una mayor seguridad en desplazamientos.

En este contexto, la movilidad eléctrica cobra una importancia crucial como parte de la movilidad sostenible. El vehículo eléctrico, junto con el vehículo híbrido, constituye una de las alternativas fundamentales para el desarrollo e implantación total de la movilidad sostenible en nuestra sociedad, utilizando además la única tecnología disponible que no genera ningún tipo de emisión en su funcionamiento, reduciendo así en un 100% el impacto ambiental del entorno por el que circula.

No obstante, aún existen algunos problemas asociados a la movilidad eléctrica que es necesario superar, en especial el aumento de la eficiencia de los motores para conseguir

prestaciones similares a los vehículos de combustión, la reducción de precios a través de la estandarización, el aumento de autonomía de las baterías, la proliferación de puntos de recarga, etc.

Es fundamental observar atentamente la evolución durante la historia de la movilidad eléctrica, poniendo el foco en los diferentes obstáculos que han ido surgiendo y la manera en la que la ingeniería los subsanó, para así presentar unas perspectivas de futuro para la movilidad eléctrica coherentes con la realidad y lo más precisas posibles.

1.1.1. Origen y evolución de la movilidad eléctrica

Pese a que muchas personas creen que el concepto de vehículo eléctrico es relativamente actual o novedoso, la realidad es que la idea tiene más de cien años de antigüedad.

Algunos estudios defienden que el primer vehículo eléctrico funcional se estrenó el 31 de agosto de 1894, fruto del trabajo conjunto de Henry Morris y Pedro Salom, en las calles de Filadelfia. Este vehículo tenía la apariencia de ser un carro de la época sin caballos, y tenía un peso de más de 2.000 kg, pesando las baterías más de 700 kg. Otras fuentes refieren que el primer vehículo eléctrico puro fue construido por un hombre de negocios escocés llamado Robert Anderson entre 1832 y 1839. En cualquier caso, como ha sucedido con numerosos inventos de la historia, la autoría del primer vehículo de propulsión eléctrica nunca quedará completamente clara.

Figura 1.1 – Henry Morris y Pedro Salom en su vehículo eléctrico en 1894

A principios del siglo XX, algunos automóviles de propulsión eléctrica alcanzaron un relativo éxito comercial, pero el auge del desarrollo de los vehículos de gasolina, con innovaciones como la introducción del arranque eléctrico en vehículos de combustión o la aparición de las primeras cadenas de montaje provocó la desaparición casi por

completo del vehículo eléctrico durante los siguientes 70 u 80 años, dejándolo relevado a aplicaciones industriales muy concretas.

No fue hasta 1996 cuando apareció el primer vehículo eléctrico de altas prestaciones contemporáneo, el denominado EV1 de General Motors. Este vehículo nace como consecuencia de la ley "Zero Emision Vehicle Mandatory", implantada en la década de los 90 en California. Se fabricaron unas 1.100 unidades de este vehículo biplaza durante dos fases (una primera de 1997 a 1999, y otra posterior hasta 2001). Durante este período, la mayoría de estos vehículos fueron arrendados a flotas y particulares en California y Arizona, contando con hasta 13.000 dólares de subvención ofrecida por la marca. Pese a que muchos usuarios deseaban prolongar el leasing o comprar los automóviles, General Motors se limitó a ejercer sus derechos legales de retirar el vehículo y destruirlo.

Figura 1.2 – GM EV1

Leyes como la del estado de California provocaron que, en una época en la que el vehículo híbrido tenía más viabilidad como alternativa, aparecieran varios coches eléctricos con prestaciones muy razonables y autonomía similar a los de hoy. Casi todos eran coches convencionales transformados, aunque unos pocos fueron desarrollos desde cero. Además del EV1, aparecen también vehículos como el Chevrolet S-10, Solectria Geo Metro o Ford Ranger, Toyota RAV4 EV, etc.

Estos coches ofrecían una autonomía suficiente para el 90% de los desplazamientos habituales de la población, sus prestaciones eran ya adecuadas y despertaron una contenida expectación. No obstante, los fabricantes no se esforzaron en demasía para que los conociese la opinión pública, dando apenas publicidad a este tipo de modelos. Al final, esta presión por parte de los fabricantes hizo que se acabaran saliendo con la suya, logrando rebajar las exigencias de la Ley, cambiando coches de emisión cero por coches de bajas emisiones. La industria petrolífera presionaba mucho para crear un clima desfavorable para estos coches, de forma que supusieron la práctica segunda desaparición del coche eléctrico.

No obstante, en la época actual, las normativas progresivamente más restrictivas en lo que se refiere a emisiones, el aumento del precio del petróleo y una mayor mentalización social sobre problemas medioambientales han provocado una nueva impulsión con

fuerza del desarrollo del vehículo eléctrico, haciendo emerger el vehículo híbrido, en gran parte gracias al Toyota Prius, y comenzando el vehículo eléctrico a hacerlo propio.

Figura 1.3 – Vehículo híbrido Toyota Prius

En la actualidad sumándose ya prácticamente todos los fabricantes del sector a nivel mundial a iniciativas relacionadas con el desarrollo de la movilidad eléctrica, mientras que el coche convencional de motor de combustión interna comienza ahora su declive. A la movilidad eléctrica pura y los híbridos se le han sumado los híbridos enchufables, vehículos que aúnan las mejores cualidades de ambas tecnologías.

1.1.2. Previsiones de futuro de la movilidad eléctrica

La revolución verde del automóvil está a la vuelta de la esquina. El coche eléctrico tiene futuro, casi inmediato, aunque antes hay que resolver ciertos retos, como la disposición de buenas redes de suministro y recarga, el desarrollo de baterías con más autonomía o el perfeccionamiento de los motores eléctricos.

La “chispa” se encendió hace ya más de diez años, con la llegada de los primeros vehículos híbridos al mercado, que combinaban un motor de combustión tradicional con otro eléctrico. Desde entonces, el panorama ha cambiado, y ya se habla de la electrificación total de los vehículos, de forma que algunas consultoras afirman que en el plazo de diez años el porcentaje de vehículos eléctricos podría llegar incluso al 25%. Esto, además, podría derivar en un ahorro energético y unos beneficios medioambientales muy importantes, por los datos que se manejan en el sector. En un vehículo, el 46% de la energía liberada por la batería sirve para mover el vehículo, lo que supone una eficiencia entre el 10% y el 30% superior a la de un vehículo convencional con motor de explosión.

Desde el punto de vista de la ingeniería, se trata de una verdadera revolución, según apunta el Director de la Escuela Politécnica Superior de la UC3M, el profesor Emilio Olías Ruiz. Igual que el vehículo automóvil que funciona con combustibles derivados del petróleo o biocombustibles que conocemos hoy en día supuso un cambio de paradigma en los modelos de transporte para nuestra sociedad, el vehículo eléctrico, será una de las soluciones más importantes al problema del transporte sostenible, e incorporará

nuevas soluciones tecnológicamente adecuadas y adaptadas a los requerimientos que se le exijan. La movilidad eléctrica se está empezando a introducir en la conciencia colectiva de la sociedad como una necesidad cada vez más imperiosa.

No obstante, tanto desde el sector automovilístico como desde el ámbito universitario e investigador, existe la convicción de que este mundo verde apenas acaba de comenzar. Existen aún muchos retos que alcanzar, lo que genera un mundo de investigación, desarrollo e innovación tecnológica en el que este país está sumamente implicado, apostando fuertemente por tratar de tener plantas de fabricación de vehículos eléctricos y universidades y centros de investigación trabajando activamente en proyectos en el ámbito de la movilidad eléctrica.

El vehículo eléctrico es un eje de creación de puestos de trabajo para el futuro, tanto directos como indirectos, además de potenciar la sinergia entre la universidad y la empresa, generando riqueza intelectual y económica.

Es necesario entender el vehículo eléctrico como un sistema complejo en el que se integran multitud de ramas del conocimiento y la ingeniería, desde la mecánica a la electrónica, pasando por la informática y la automática. Debido a ello, los trabajos de diseño en movilidad eléctrica se centran en contemplar aspectos diversos como son la aerodinámica, el peso, la potencia, las prestaciones, la aceleración, el frenado, la gestión energética, etc. manteniendo siempre criterios de sostenibilidad en el horizonte de los proyectos.

La tecnología electrónica, la programación y el software desempeñan un papel fundamental en la interacción entre el motor eléctrico y los sistemas de almacenamiento de energía, elementos críticos y fundamentales en el desarrollo de este tipo de vehículos.

1.2. Objetivos del proyecto

En el presente proyecto se pretende diseñar un sistema electrónico en el que se pueda controlar la velocidad de un motor eléctrico del tipo *brushless*, mediante la utilización de un microcontrolador que reciba la lectura de una referencia de velocidad y la información de la velocidad actual del motor, implemente un PID y envíe una serie de señales de control a un inversor que regule la velocidad del motor.

El diagrama de bloques del sistema se puede observar en la imagen siguiente, acompañado de una descripción breve de cada uno de ellos:

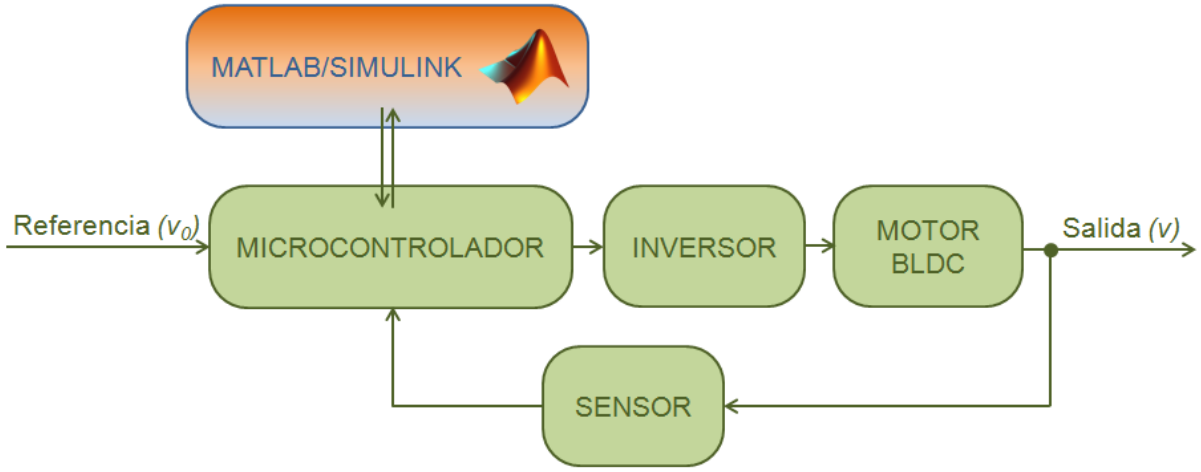

Figura 1.4 – Esquema del sistema a desarrollar durante el proyecto

- Microcontrolador: Es el cerebro del sistema, que se encargará de recibir como señales eléctricas la referencia de velocidad y la lectura del sensor, implementar un controlador PID y generar la salida correspondiente, proceso que debe ser programado.
- Inversor: El inversor es el dispositivo electrónico de potencia, que se encargará de convertir las señales digitales de Arduino en una transmisión de potencia eléctrica que llegue al motor.
- Motor BLDC: El motor es el dispositivo que transforma la potencia eléctrica que transmite el inversor en potencia mecánica en forma de movimiento.
- Sensor: El sensor es el dispositivo encargado de transformar un parámetro físico del motor como es la velocidad en señales eléctricas.

Los desarrollos del presente proyecto se va a aplicar a un motor en concreto, utilizando un inversor en concreto, seleccionado a partir de unos criterios técnicos y económicos que se acotaran durante la presente memoria. No obstante, se pretende que el sistema sea lo más estándar posible, para que se pueda aplicar a la mayoría de motores BLDC sin importar la potencia, régimen de velocidad u otros condicionantes.

Otro punto clave del presente proyecto es el de aplicar las herramientas de programación gráfica mediante bloques que ofrece MATLAB/Simulink en un proyecto real que emplee un microcontrolador.

1.3. Justificación del proyecto

Tal y como se ha mencionado en la introducción, el vehículo eléctrico se encuentra en una fase con una alta importancia de la innovación y el desarrollo, en particular para la mejora de la eficiencia de los sistemas de propulsión y su reducción de costes y estandarización.

Mediante este proyecto, se pretende participar en este crecimiento actual, tratando de abaratar por un lado y mejorar la calidad por otro de los sistemas de control de motores BLDC, que son los incorporados habitualmente en este tipo de vehículos, mediante la actuación en dos vías.

Por un lado, mediante el empleo de microcontroladores que permitan reducir el coste de los componentes del sistema, y por otro facilitando el proceso de programación de éstos mediante la utilización de herramientas de programación por bloques gráficos, que acorten el proceso de desarrollo del software de control de motores y por tanto reduzcan costes de desarrollo de estos sistemas.

1.4. Contenido del proyecto

A continuación se describirá brevemente en qué consiste cada uno de los capítulos de este documento:

- En el Capítulo 1, el que nos ocupa, se introduce el concepto de motor eléctrico, poniendo en valor su importancia en el desarrollo tecnológico a lo largo de la historia, en particular en el ámbito del transporte de personas. Además, se presenta la justificación y los objetivos perseguidos con el presente proyecto.
- En el Capítulo 2, se profundiza en el concepto de motor BLDC, analizando teóricamente su funcionamiento y las diferentes técnicas de control disponibles.
- En el Capítulo 3, se realiza una descripción del modelado y simulación que se ha llevado a cabo para poner descritas las dos técnicas de control disponibles, con el objetivo de reunir información de utilidad a la hora de implementar el sistema físico.
- En el Capítulo 4, se analizan las distintas posibilidades de implementación física, y se describe el proceso seguido y los resultados finales alcanzados.
- En el Capítulo 5, se recoge un breve resumen, las conclusiones del trabajo llevado a cabo y posibles futuros desarrollos partiendo del presente proyecto.

2. Técnicas de control de motores BLDC

2.1. Motor BLDC

2.1.1. Introducción

Los motores sin escobillas, denominados habitualmente BLDC por su nombre en inglés (*Brushless Direct Current*), son motores que ha ido ganando popularidad a lo largo del tiempo. Hoy en día, los motores BLDC son ampliamente utilizados en sectores tan diversos como son el automovilístico, aeroespacial, médico, automatización industrial, electrónica de consumo, etc.

Tal y como indica su nombre, los motores BLDC no emplean escobillas para conmutar, sino que conmutan electrónicamente. Las principales ventajas que presentan este tipo de motores respecto a los tradicionales de escobilla y los motores de inducción de corriente alterna son:

- Mejores características de velocidad frente a torque
- Alta respuesta dinámica
- Alta eficiencia
- Largo ciclo de vida
- Funcionamiento silencioso
- Mayores rangos de velocidades
- Inocuos en atmósferas explosivas debido a que no producen chispas
- Mejor relación par motor-tamaño

2.1.2. Principios constructivos

Los motores BLDC son un tipo de motor síncrono, lo que quiere decir que el campo magnético generado por el estator y el generado por el rotor giran a la misma frecuencia. Así, en estos motores no se observa el deslizamiento propio de motores asíncronos o de inducción.

Habitualmente, los motores BLDC pueden tener configuración monofásica, bifásica o trifásica, en función del número de devanados que tenga el estator. De las tres posibilidades, los motores trifásicos son los más populares y ampliamente utilizados, por lo que serán los que centren el estudio del presente proyecto.

2.1.2.1. Estator

El estator de un motor BLDC consiste en una serie de láminas de acero apiladas con devanados arrollados en torno a espacios generados por cortes axiales. Es habitual que el estator de un motor BLDC se parezca visualmente al de un motor de inducción, pero tiene las bobinas distribuidas de una manera diferente. La mayoría de los motores BLDC tienen las tres bobinas correspondientes a las fases conectadas en estrella. Cada una de esas bobinas está formada por numerosos anillos interconectados, que se distribuyen adecuadamente por la periferia del rotor para formar seis polos de potencia

uniformemente distribuidos. En la figura 2.1 se puede observar la estructura del estator de un motor BLDC.

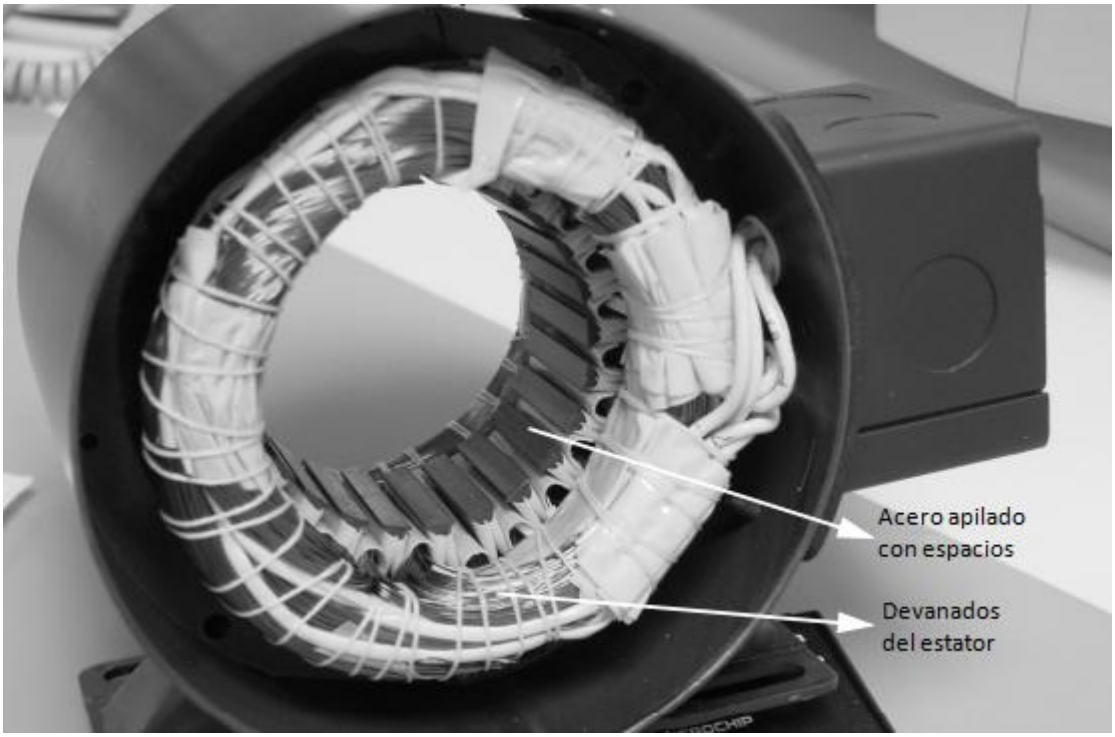

Figura 2.1 – Estator de un motor BLDC

Existen dos variantes de motores BLDC en función del tipo de bobinas que posea el estator, los motores trapezoidales y sinusoidales. La diferencia viene impuesta por el tipo de interconexión entre los anillos de las bobinas, que generan distintos tipos de forma para la fuerza contraelectromotriz, o BEMF por su denominación en inglés (*Back Electromotive Force*), concepto en el que se ahondará más adelante.

Tal y como indica su nombre, un motor trapezoidal genera una fuerza contraelectromotriz con una forma trapezoidal, mientras que un motor sinusoidal genera una fuerza contraelectromotriz con una forma sinusoidal, tal y como se puede observar en la figura 2.2.

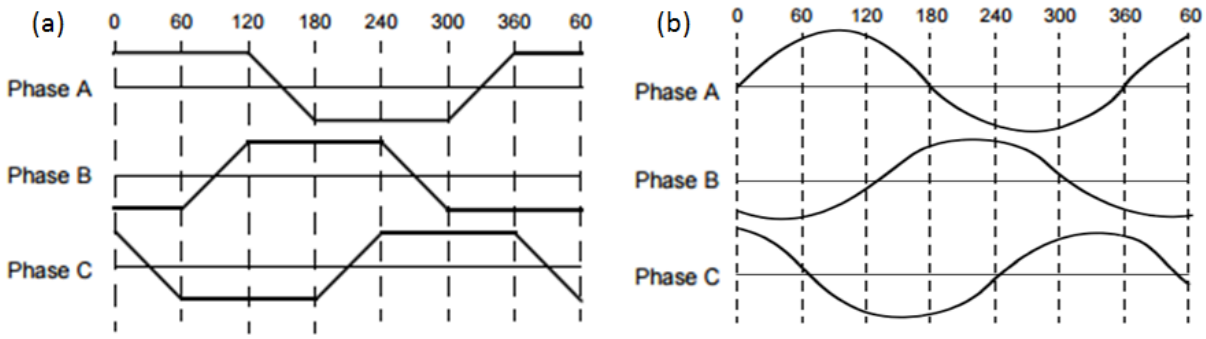

Figura 2.2 – Fuerza contraelectromotriz en un motor trapezoidal (a) y en uno sinusoidal (b)

Además de la fuerza contraelectromotriz, también la corriente que circula por las fases tiene la forma correspondiente a cada tipo de motor. De esta forma, los motores sinusoidales tienen ofrecen el par de una manera más suave que un motor trapezoidal.

Sin embargo, estos también son más costosos, debido a que requieren más interconexiones entre anillos para incrementar la distribución de polos en el estator.

En función de la tensión disponible, se deberá elegir el motor con el estator adecuado. En aplicaciones como la automoción y pequeños brazos robóticos, se emplean motores de $48V$ o menos, mientras que en grandes aplicaciones industriales se emplean motores de más de $100V$.

2.1.2.2. Rotor

El rotor de un motor BLDC está formado por imanes permanentes, pudiendo variar el número de polos magnéticos.

En función de la densidad de campo magnético que se requiera, se selecciona el material para construir el rotor. Tradicionalmente, se usaron imanes de ferrita para la construcción de imanes permanentes, y hoy en día, merced al avance de la tecnología, los imanes formados por materiales de tierras raras están ganando popularidad. Los imanes de ferrita son más baratos, pero presentan la desventaja de tener una densidad de flujo mucho menor por unidad de volumen. En contraste, los imanes de materiales de las tierras raras permiten reducir mucho el tamaño del rotor para un mismo torque suministrado.

Ejemplos de tierras raras utilizadas en la construcción de imanes permanentes son el Neodimio (Nd), la aleación de Samario y Cobalto (SmCo) y la aleación de Neodimio, Hierro y Boro (NdFeB). Existe una investigación continua en el campo de este tipo de materiales buscando incrementar la densidad de flujo para conseguir reducir aún más el rotor.

En función del número posición de los imanes del rotor se puede distinguir entre varias tipologías, tal y como se observa en la figura 2.3

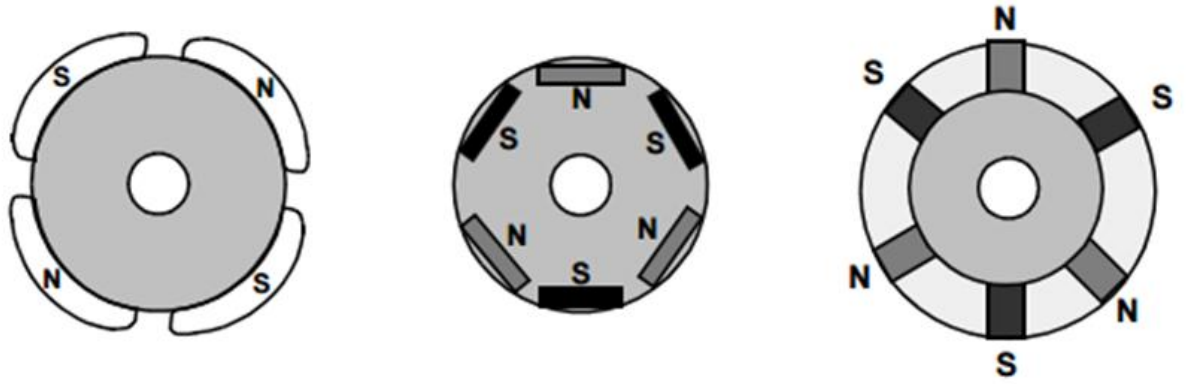

Figura 2.3 – Diferentes tipologías de rotor de motores BLDC

En la figura 2.4 se puede apreciar la estructura del rotor de un motor BLDC

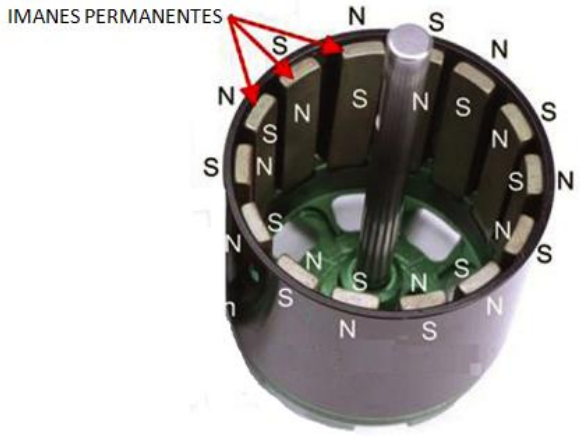

Figura 2.4 – Rotor de un motor BLDC

2.1.3. Característica torque/velocidad

Cuando se analiza cualquier tipo de motor, independientemente de su fuente de energía, principio de funcionamiento, potencia, características constructivas u otro tipo de parámetros, es interesante siempre analizar la relación que tienen torque y velocidad en la respuesta del motor.

De manera general, podemos relacionar la potencia, el torque y la velocidad mediante la ecuación 2.1.

$$P = M \cdot \omega \tag{2.1}$$

Donde P es la potencia que desarrolla el motor, M es el par motor o torque y ω es la velocidad angular de giro del motor. Tal y como se puede extraer de la ecuación, a potencia constante, el torque y la velocidad de giro son inversamente proporcionales.

En la figura 2.5 se puede observar una curva torque/velocidad típica para motores BLDC.

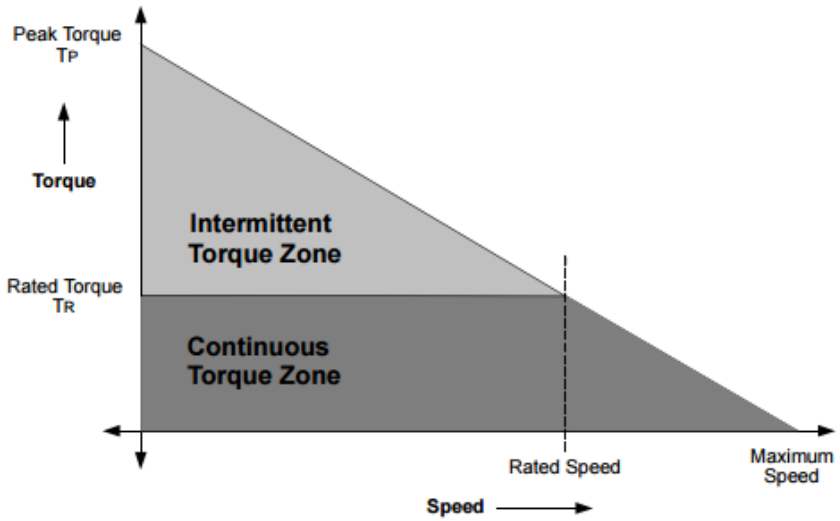

Figura 2.5 – Curva característica torque/velocidad típica para motores BLDC

De la figura 2.5 hay dos parámetros que son importantes porque se usan para definir un motor BLDC: el par máximo (T_P) y el par nominal (T_R).

Durante operaciones continuas, se observa que el torque permanece constante en un rango de velocidad acotado superiormente por la velocidad nominal. A partir de este valor, el torque comienza a caer, de forma que, aunque el motor puede llegar a una máxima velocidad que puede superar en más del 50% la velocidad nominal, el torque para esta velocidad máxima será nulo.

Por otra parte, existen aplicaciones en las que se exigen frecuentes arranques y paradas del motor, así como frecuentes cambios de sentido de giro, que pueden demandar un torque superior al nominal. Este requisito puede aparecer especialmente en periodos breves, como cuando el motor arranca desde parado durante el proceso de aceleración. Durante este periodo, el torque extra que demanda el motor se utiliza para vencer la inercia de la carga y del propio rotor, hasta alcanzar una velocidad estacionaria. El motor puede entregar un torque más alto que el nominal, hasta el torque máximo T_P, en función de la velocidad a la que se encuentre girando siguiendo la gráfica de la figura 2.5.

2.2. Teoría de operación y control

2.2.1. Principio de operación

A diferencia de un motor eléctrico de escobillas tradicional, en el que la conmutación entre devanados se produce mecánicamente provocada por el propio giro del rotor, en el caso de los motores BLDC la conmutación está controlada electrónicamente.

Cada secuencia de conmutación del motor presenta una bobina energizada con tensión positiva (la corriente entra en el devanado), otra bobina con tensión negativa (la corriente sale del devanado) y otra en estado abierto. El torque del motor se produce debido a la interacción entre el campo magnético generado por las bobinas del estator y el de los imanes permanentes que forman el rotor.

Idealmente, el pico de torque sucede cuando ambos campos están distanciados 90º, y decrece a medida que éstos se acercan. Con el objetivo de mantener el motor funcionando, es necesario que el campo magnético que se produce en las bobinas del estator cambie de posición, de forma que el rotor se mueva buscando atrapar el campo magnético generado por el estator. Para un determinado sentido de giro, existe una secuencia en la que se debe energizar las bobinas, que en un motor BLDC trifásico se denomina conmutación de seis etapas.

2.2.2. Control de posición

Para controlar el funcionamiento de un motor BLDC, tal y como se ha indicado, la clave reside en controlar en todo momento la posición en la que se encuentra el rotor. En el caso de un motor BLDC trifásico, como los que se van a manejar en el presente proyecto, se debe seguir el proceso de conmutación de seis etapas. En un ciclo eléctrico completo, cada fase estará 120º eléctricos polarizada con tensión positiva, 120º con tensión

negativa y 120º en estado flotante o abierto. Un ejemplo de una secuencia de conmutación podría ser la que se muestra en la figura 2.6.

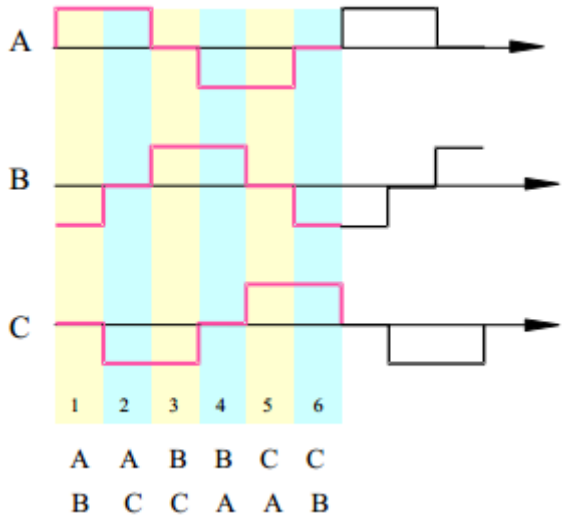

Figura 2.6 – Ejemplo de secuencia de conmutación motor BLDC trifásico

Así, al activar las bobinas siguiendo la secuencia AB-AC-BC-BA-CA-CB, se logrará que el motor gire. Tal y como se observa en el gráfico y en la secuencia, a cada instante únicamente hay dos bobinas conduciendo corriente, mientras que la tercera queda en estado flotante.

Con motivo de la interacción que ejercen los imanes del rotor con los devanados del estator, se genera una tensión en las bobinas, la fuerza contraelectromotriz o BEMF, ya mencionada anteriormente, que es el efecto medible de la resistencia al movimiento que los imanes del rotor ejercen sobre los devanados del estator, y que depende principalmente de la posición angular del rotor (fase) y de la velocidad de giro de éste (amplitud).

Idealmente, con el objetivo de conseguir el máximo torque, se deberá conmutar exactamente cada 60º eléctricos, de forma que la corriente de las bobinas se encuentre en fase con la fuerza contraelectromotriz. De esta forma, la posición del rotor determina el momento en el que es necesario realizar la conmutación.

2.2.3. Control de velocidad

Una vez se dispongan de los mecanismos necesarios para controlar que la conmutación entre bobinas del estator se produzca en el momento exacto, la problemática se traslada ahora a conseguir controlar la velocidad de giro del rotor.

Por suerte, el campo magnético que se genera en los bobinados del estator es proporcional a la tensión con la que estos se alimentan, y a su vez la velocidad del rotor será proporcional al campo magnético que se genere en el estator. Así pues, la idea es regular la tensión que se le suministra a los devanados del motor para modificar la velocidad de giro del rotor. Dicha tensión está suministrada por una batería, que alimenta un inversor trifásico controlado electrónicamente para realizar la conmutación entre fases.

Dado que la batería suministra una tensión aproximadamente constante, se utilizará en el control del inversor la técnica de la modulación por ancho de pulso, conocida también como PWM (*Pulse-Width Modulation*). Esta técnica consiste en modificar el ciclo de trabajo de una señal periódica digital de alta frecuencia, es decir, modificar el porcentaje del ciclo que se encuentra con valor alto. En la figura 2.7 se puede observar la representación gráfica de un PWM aplicado en una señal de 5V.

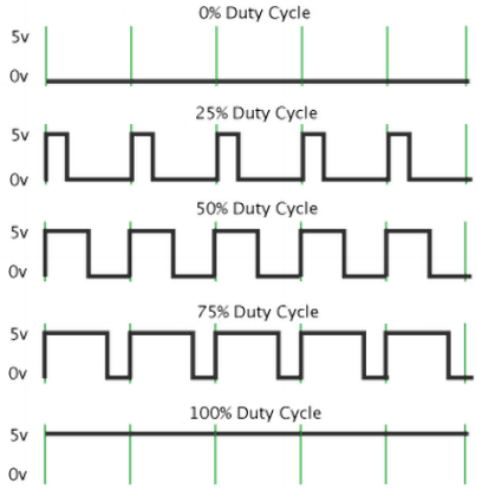

Figura 2.7 – Modulación por ancho de pulso (PWM)

Al aplicar esta señal en la entrada de los transistores que forman el inversor trifásico, se consigue modificar el valor eficaz de tensión que recibe el estator, modificando así la velocidad de giro del motor.

Para relacionar tensión con velocidad de giro, se construirá un controlador PID en lazo cerrado que, utilizando la medición de la velocidad actual del motor y una referencia de velocidad deseada, generará una señal de PWM que se aplicará directamente sobre el inversor. En la figura 2.8 se aprecia el esquema de control, tanto de posición como de velocidad, que se ha diseñado.

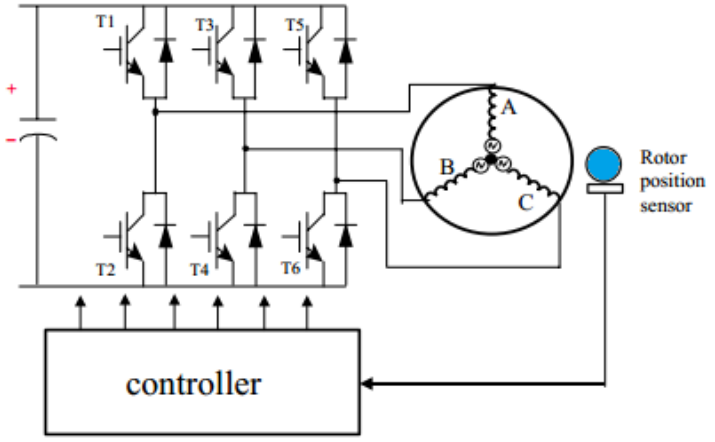

Figura 2.8 – Esquema de control clásico de un motor BLDC

2.3. *Control con sensores*

A raíz del análisis de la teoría de control de los motores BLDC, tanto para posición como para velocidad, el problema se reduce en conocer en todo momento la posición angular relativa del rotor respecto a las bobinas del estator, para así poder utilizar esta información para conmutar adecuadamente el inversor trifásico. Así, un asunto vital en el diseño de un sistema de control es definir adecuadamente cuál es el elemento marcado en color azul en la figura 2.8.

2.3.1. Sensores Hall

La primera alternativa que se va a plantear en el presente proyecto es el empleo de sensores basados en el llamado efecto Hall.

El efecto Hall consiste en la fuerza que un campo magnético ejerce sobre una corriente eléctrica que circula por un conductor que lo atraviesa, tal y como se observa en la figura 2.9. Esta fuerza generada es transversal al movimiento de los portadores de carga, lo que se traduce en una tendencia a desplazarlos hacia uno u otro lado del conductor. Este efecto se evidencia en una fina lámina conductora, en la que un desplazamiento de cargas hacia uno de los dos lados puede generar una tensión medible entre los dos terminales del conductor. La presencia de esta tensión medible transversal se denomina efecto Hall en honor a su descubridor, Edwin Herbert Hall.

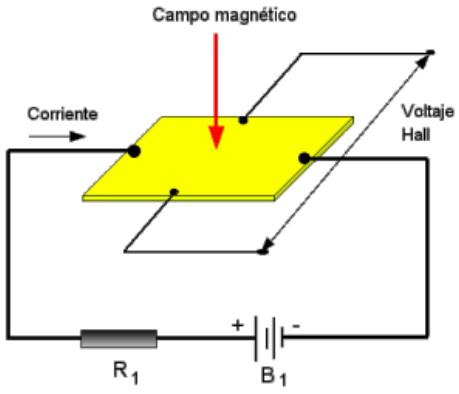

Figura 2.9 – Efecto Hall

Habitualmente, en los motores BLDC que cuentan con ellos, los sensores Hall se sitúan en el estator. En el momento que un polo magnético del rotor pasa sobre el sensor de efecto Hall, éste genera una señal que puede ser alta o baja, lo que indica si el polo que ha pasado es norte o sur. Utilizando las señales producidas por los tres sensores Hall, se puede determinar la secuencia de conmutación exacta.

Existen varias opciones constructivas para la integración de los sensores Hall en los motores BLDC. En ocasiones, éstos se incrustan dentro del estator, proceso bastante complejo porque afecta a la estructura mecánica del motor, y en el que es imprescindible un correcto alineamiento entre los tres sensores para no producir errores en la lectura de la posición del rotor. Para simplificar el diseño, algunos motores llevan los sensores Hall incrustados en el rotor, además de los imanes permanentes principales. En este caso, a

medida que gira el rotor, los sensores detectan el campo magnético que generan los devanados del estator. Cualquiera de estas dos opciones constructivas está reservada para motores BLDC empleados en aplicaciones muy específicas.

Para aplicaciones más corrientes, es habitual que los sensores Hall vayan montados sobre una placa que se fija al estator. Esta opción presenta la ventaja de que el usuario puede ajustar ligeramente la posición de los sensores para alinearlos lo mejor posible con los imanes del rotor para conseguir el mejor rendimiento posible. Como desventaja, el sistema pierde robustez al permitir holguras que no eran posibles en el caso de los sensores incrustados.

2.3.2. Teoría de operación

Dentro de un ciclo eléctrico, que corresponde con una revolución completa del rotor, un sensor Hall estará devolviendo una señal positiva durante medio ciclo, para pasar a conmutar y devolver una señal negativa durante otro medio ciclo. En función del desfase existente entre los sensores Hall y de la configuración del rotor, existen numerosas combinaciones para la correspondencia entre la lectura de los sensores Hall y el estado de excitación de las bobinas. No obstante, no son tan importantes los valores como conocer exactamente los estados de los sensores que se corresponden con cada una de las combinaciones de los sensores Hall.

En la figura 2.10 se muestran los estados de los sensores Hall y de excitación de las bobinas para el motor BLDC que viene modelado en MATLAB/Simulink, del que se hablará en el siguiente capítulo.

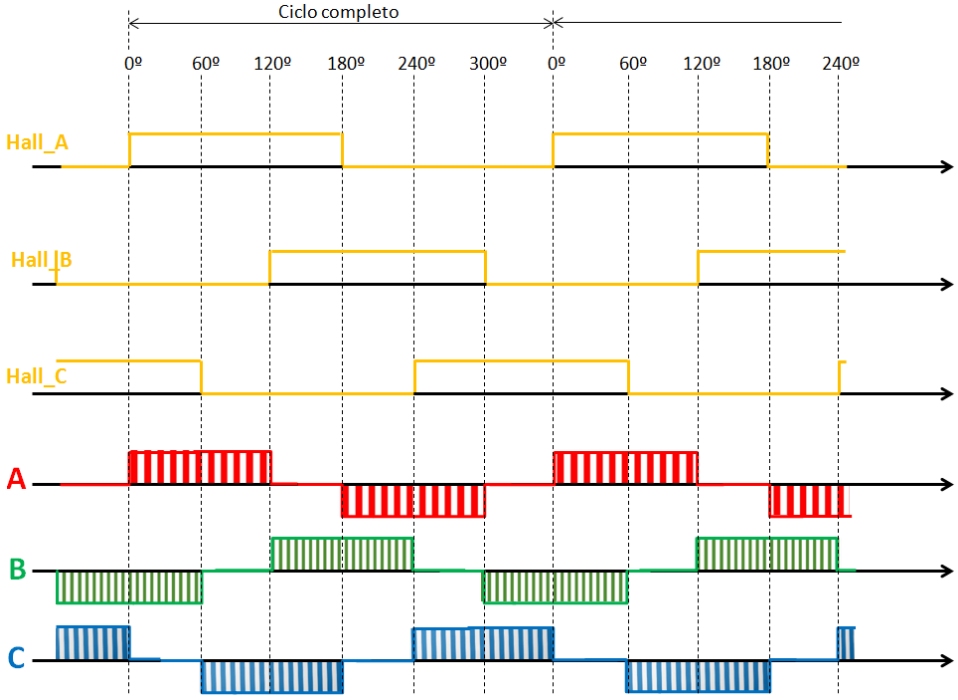

Figura 2.10 – Secuencia de conmutación de los sensores Hall

En este motor, tal y como se puede observar en la figura 2.10, los sensores tienen un desfase entre sí de 60º, y el cambio de negativo a positivo de un sensor coincide en el

tiempo con el instante en el que la bobina identificada con la misma letra que el sensor pasa de abierta a alimentada con tensión positiva.

2.3.3. Limitaciones de los sensores Hall

Es habitual el empleo de sensores tipo Hall en todo tipo de aplicaciones de control de motores BLDC, tal y como se ha mencionado. No obstante, estos sensores presentan también algunos inconvenientes y limitaciones que los hacen inviables o ineficaces en determinadas aplicaciones, haciendo que sea interesante buscar otras alternativas para la sensorización de la posición del rotor del motor BLDC.

En primer lugar, el empleo de sensores Hall incrementa el coste y el tamaño del motor, y es necesario tener en cuenta en el diseño de éste las modificaciones mecánicas que son necesarias para su integración.

Además, los sensores, y en particular los de tipo Hall, son muy sensibles a la temperatura, por lo que esto supone una limitación a los rangos de trabajo del motor, siendo habitual que el límite de temperatura se sitúe alrededor de 75ºC. Por otro lado, la robustez del sistema puede verse comprometida en el momento que se incrementan los componentes y cableados, al producirse un incremento de las probabilidades de fallo en el sistema.

Por último, existen algunas aplicaciones en las que las características del motor o la geometría del montaje impiden la utilización de ningún tipo de sensores. Debido a todas estas circunstancias, durante los últimos años se ha producido un aumento del interés en los métodos de control de motores BLDC sin sensores, denominados habitualmente *sensorless*.

2.4. Control sin sensores

2.4.1. Introducción

Dentro de los tipos de control *sensorless* que aparecen referenciados en bibliografía, podemos distinguir entre dos tipos de técnicas diferentes.

Por un lado, existen técnicas basadas en la detección de la fuerza contraelectromotriz que el giro del rotor genera sobre los devanados. En función de la forma de las ondas de la fuerza, se podrá conocer la posición del rotor respecto del estator.

Por otro lado, hay otras técnicas que se basan en la estimación de la posición del rotor empleando parámetros del motor y las tensiones y corrientes consumidas. Esta técnica tiene el inconveniente principal de que requiere procesadores digitales de señal, DSP, para realizar los complicados cálculos en tiempo real requeridos, provocando un incremento en el coste del sistema.

Debido a las ventajas e inconvenientes de cada uno de ellos, la técnica más utilizada es la detección de la fuerza contraelectromotriz, o *back EMF sensing*, técnica que será en la que centraremos el análisis del presente proyecto.

2.4.2. Detección de la fuerza contraelectromotriz

Tal y como se ha indicado previamente, en un motor brushless únicamente se encuentran conduciendo corriente dos bobinas simultáneamente, mientras que la tercera se encuentra flotante. En el caso de la figura 2.6, en el instante inicial las fases A y B se encuentran conduciendo corriente, mientras que la fase C se encuentra en estado flotante. Este estado se mantiene durante 60º eléctricos, hasta que se produce la conmutación en el inversor.

Dado que hemos dicho que la corriente en las bobinas se conmuta para que se encuentre en fase con la fuerza contraelectromotriz con el objetivo de lograr un rendimiento óptimo torque/corriente, y los instantes a los que debe producirse la conmutación vienen determinados por la posición del rotor, y sabiendo que la forma de la fuerza contraelectromotriz indica la posición del rotor, es lógico afirmar que si somos capaces de conocer la fuerza contraelectromotriz, seremos capaces de determinar los instantes de conmutación.

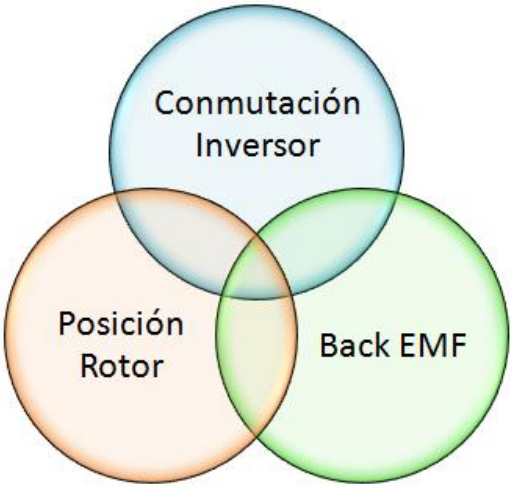

Figura 2.11 – Relación entre la fuerza contraelectromotriz y otros parámetros de interés

En la figura 2.11 se muestra simbólicamente la relación existente entre el parámetro que podemos controlar (la conmutación del inversor), el que queremos detectar (la posición del rotor) y el que podemos medir (la fuerza contraelectromotriz).

Por lo tanto, para conseguir un control del giro del motor mediante esta técnica, el paso crucial es lograr detectar la fuerza contraelectromotriz de las bobinas.

Mediante simulación, se pueden obtener gráficas ideales de la fuerza contraelectromotriz de cada una de las bobinas, en función de la etapa de control en la que nos encontremos, lo que determina la posición del rotor, tal y como se muestra en la figura 2.12

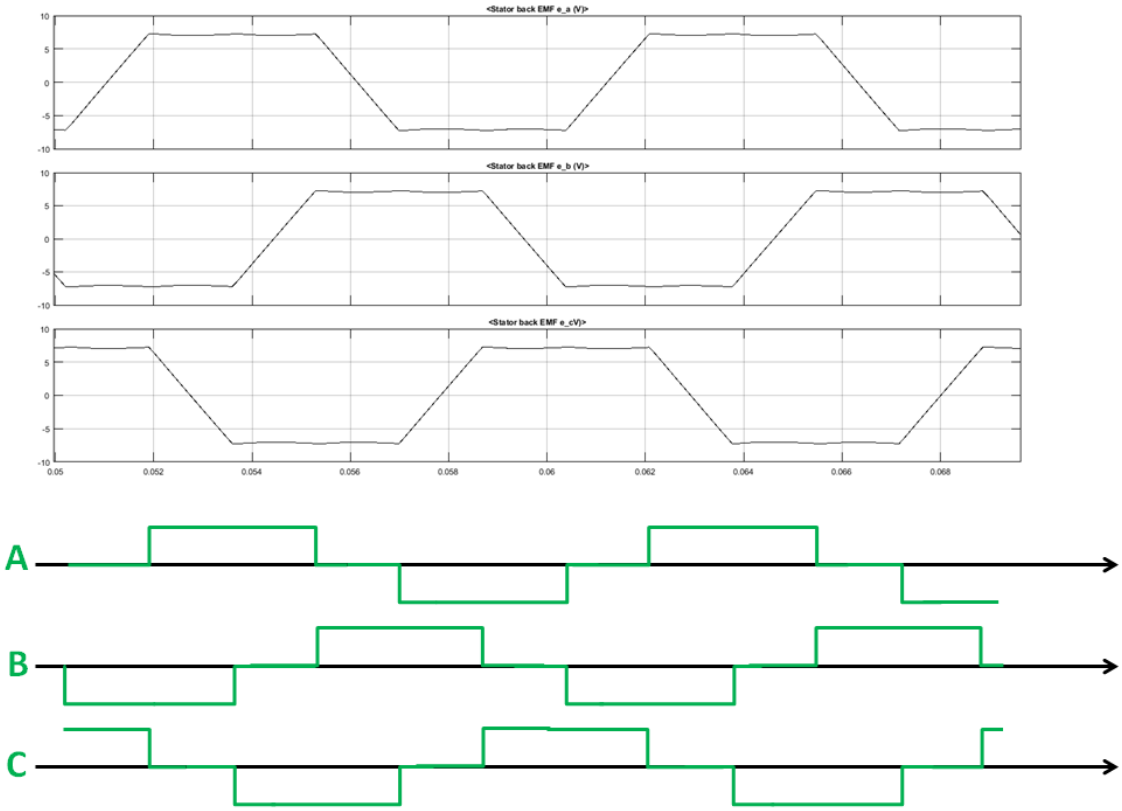

Figura 2.12 – Gráficas de la fuerza contraelectromotriz y su correspondencia con las etapas de control

Desafortunadamente, estas gráficas únicamente se pueden obtener mediante simulación, y no es posible obtenerlas midiendo magnitudes sobre un sistema real.

Podemos analizar una de las fases del motor asumiendo las siguientes hipótesis de idealidad para establecer un circuito equivalente:

- El motor no está saturado.
- Los devanados de las fases son idénticos entre sí, con idénticos componentes resistivos, las inductancias son constantes y no existen inductancias mutuas.
- Las pérdidas en el hierro son despreciables.

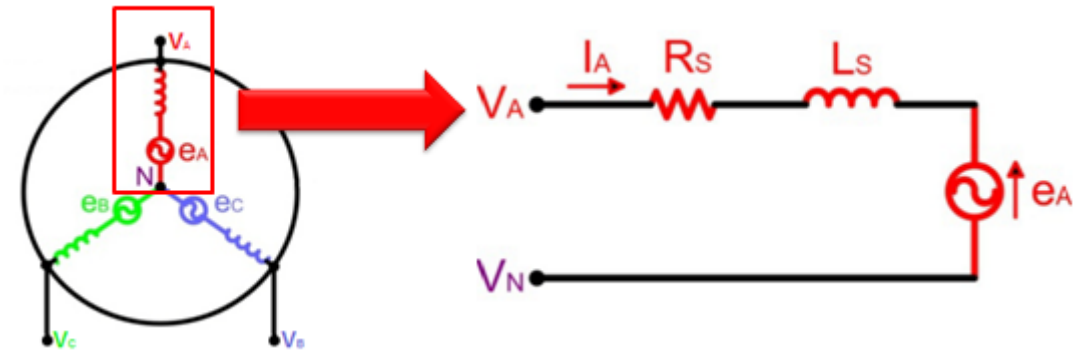

Figura 2.13 – Circuito equivalente a una de las fases del motor

A raíz de la figura 2.13, la tensión que existe entre el terminal de la fase A y el neutro puede ser obtenida por la ecuación 2.2:

$$V_{AN} = R_S \cdot I_A + L_S \cdot \frac{dI}{dt} + e_A \quad (2.2)$$

Donde:

- V_{AN} es la tensión entre el terminal de la fase A y el neutro.
- I_A es la corriente que circula a través del devanado de la fase A.
- R_S es el componente resistivo de la impedancia del devanado del estator correspondiente a la fase A.
- L_S es el componente inductivo de la impedancia del devanado del estator correspondiente a la fase A.
- e_A es la fuerza contraelectromotriz que el rotor genera en el devanado de la fase A.

A raíz de la expresión anterior, observamos que, midiendo la tensión existente entre fase y neutro obtendremos la fuerza contraelectromotriz únicamente cuando la corriente que circula por el devanado sea cero. En el momento que la fase se activa y comienza a conducir, a la tensión e_A que deseamos medir se le superpone la caída de tensión en la bobina y la resistencia.

Por si fuera poco, la situación se agrava cuando en el inversor se aplica una modulación PWM, de forma que a la fuerza contraelectromotriz se superpone una tensión que es una secuencia de pulsos de alta frecuencia, haciendo imposible obtener una gráfica como la de la figura 2.12.

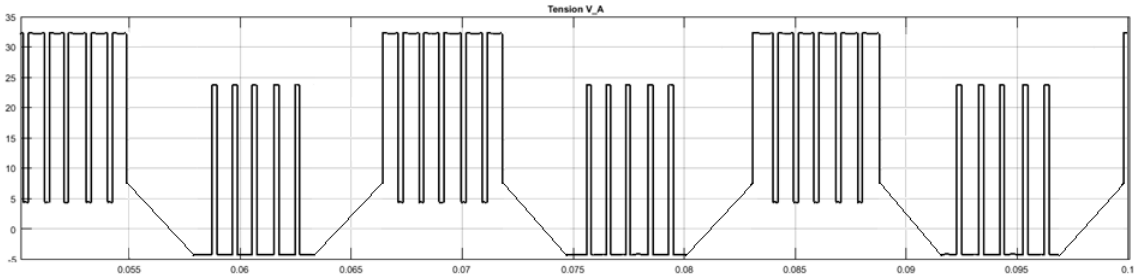

Figura 2.14 – Tensión ideal medida en la bobina A

De lo dicho anteriormente se extrae que existe una ventana en la que sí que seremos capaces de medir la fuerza contraelectromotriz midiendo la tensión entre fase y neutro, correspondiente al momento en el que la bobina se encuentra flotante, y por lo tanto la corriente que circula por ella es cero. En la figura 2.14 podemos observar de manera clara esta ventana en la que no aparece ruido de alta frecuencia provocado por el control PWM del inversor.

De este primer análisis obtenemos una valiosa conclusión a la hora de enfocar la detección de la fuerza contraelectromotriz en el sistema físico. En cada etapa de control, se deberá medir la tensión en la bobina flotante, para poder medir auténticamente la *back EMF* inducida por el rotor, sin interferencias ni tensiones superpuestas.

No obstante, existe otra razón más importante por la cual no se puede obtener directamente la gráfica de la fuerza contraelectromotriz de las bobinas. En la mayoría de motores BLDC, incluyendo los utilizados para el presente proyecto, no existe un punto de

acceso al neutro del motor, por lo que no existe la posibilidad de medir la tensión V_{AN} directamente como se haría en el circuito de la figura 2.13.

En lugar de esto, es necesario realizar las modificaciones en el esquema necesarias para medir la fuerza contraelectromotriz. Cualquiera de las técnicas que se proponen a continuación está diseñada para detectar el paso por cero de la fuerza contraelectromotriz de la bobina que se encuentra en estado abierto en cada etapa, para realizar la conmutación 30º eléctricos después.

2.4.3. Métodos convencionales

El método tradicional, propuesto por Erdman [3] y Uzuka [22], se basa en la construcción de un neutro virtual mediante tres resistencias iguales colocadas en estrella, que, en teoría, tendrá el mismo potencial que el punto neutro real del motor. Una vez construido, midiendo la tensión entre el terminal flotante y el neutro virtual se obtendrá la fuerza contraelectromotriz de la fase.

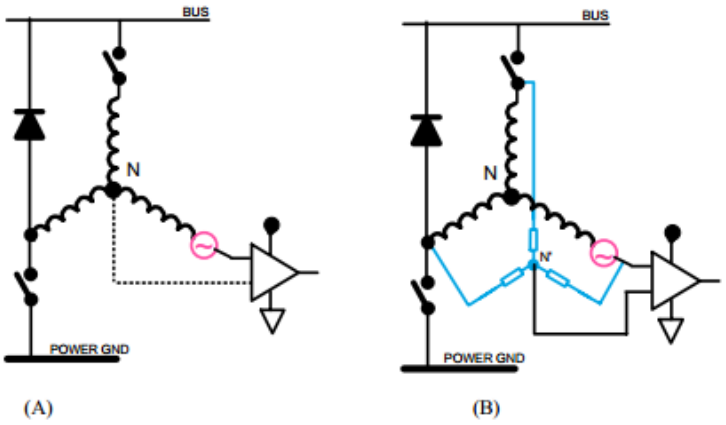

Figura 2.15 – Detección de la fuerza contraelectromotriz con el neutro del motor (A) y un neutro virtual (B)

No obstante, si medimos la tensión del punto neutral virtual N' con respecto a la tensión de referencia GND observamos que no es un punto con una tensión estable en absoluto. En lugar de eso, el potencial del neutro virtual, debido a que también está superpuesto a la señal PWM, se encuentra constantemente saltando desde cero hasta valores cercanos a la tensión del bus de alimentación, creando una señal de modo común de gran magnitud y ruido de alta frecuencia.

Por ejemplo, para un motor alimentado mediante una batería de 60 V, el potencial del neutro virtual variará entre 0V y 60V. Esto es incompatible con los circuitos de sensorización típicos como comparadores, cuyos valores admisibles de tensión son de unos pocos voltios (típicamente 5V). Por ello, será necesario realizar atenuación para reducir la señal de modo común y filtrado para eliminar el ruido de alta frecuencia.

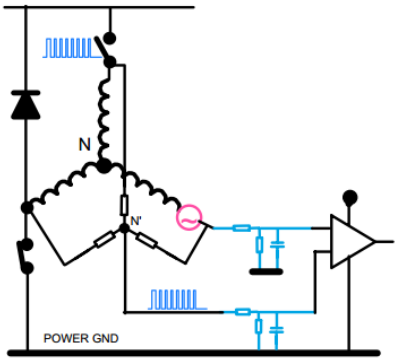

Figura 2.16 – Filtrado y atenuación de la señal medida

No obstante, el empleo del divisor de tensión presenta un inconveniente claro. Además de atenuar la señal de modo común para que sea admisible por el comparador, reducirá también proporcionalmente la fuerza contraelectromotriz que se desea medir, perdiendo así sensibilidad el sistema. Esto es especialmente preocupante en el arranque y funcionamiento a bajas velocidades, momentos en los que la magnitud de la *back EMF* es menor.

Por su lado, el filtro paso bajo para eliminar el ruido de alta frecuencia también presenta un serio inconveniente. Al incluir una capacitancia, aparecerá inevitablemente un retardo fijo en la señal detectada, con independencia de la velocidad del rotor. Esto supone que, cuando la velocidad de rotación del rotor aumenta, también lo hace el porcentaje de retardo respecto al periodo correspondiente a una vuelta completa. Este retardo, si la velocidad es suficientemente alta, puede provocar desfases importantes entre la fuerza contraelectromotriz y la corriente de las bobinas, hasta el punto de provocar serios problemas en la conmutación del inversor.

La combinación de los problemas que presentan atenuación y filtrado acota mucho la ventana de velocidad angular a la que el motor puede funcionar correctamente con un rendimiento adecuado, de forma que esta técnica ofrece un estrecho rango de velocidades de funcionamiento y unas pobres prestaciones en el arranque.

No obstante, esta técnica se ha utilizado tradicionalmente en el control *sensorless* de motores BLDC, utilizando circuitos integrados analógicos diseñados especialmente para ello, con esquemas basados en el planteado en esta técnica. Son ejemplos de estos circuitos el UC3646 de Unitrode, el ML4425 de Microlinear y el 32M595 de Silicon Systems.

Los sistemas basados en este tipo de detección debían ser cuidando especialmente la tensión de alimentación, las características del motor escogido y la velocidad y torque del motor en la aplicación, ya que la falta de flexibilidad del sistema de control no permitía realizar grandes modificaciones una vez estuviera en funcionamiento.

Adicionalmente a este método, se han planteado otros menos difundidos, como la determinación de la posición del rotor mediante la medición del tercer armónico de tensión del estator. El principal inconveniente de este método es que, a bajas

velocidades el valor del tercer armónico es muy bajo, presentando dificultades de detección.

Otro método posible es extraer la información de la posición del rotor basándose en el estado de conducción de diodos de protección de los semiconductores. El circuito de sensorización en este caso es relativamente complicado, y sigue presentando el inconveniente del funcionamiento a bajas velocidades.

2.4.4. Mejoras del método convencional

Para evitar los problemas causados por el neutro virtual en el método anterior, se va a plantear eliminar este punto, refiriendo las mediciones de la tensión de la fuerza contraelectromotriz directamente a la señal de tierra (GND). No obstante, para conseguir que esto sea válido, es necesario asumir unas ciertas consideraciones para la señal PWM que suministra el inversor.

En primer lugar, tal y como hemos indicado en numerosas ocasiones, para el control de un motor BLDC únicamente dos de las tres fases se activan simultáneamente. La fase a la que daremos el signo positivo y diremos que está polarizada directamente estará recibiendo la corriente desde el semiconductor superior correspondiente (Q3, en la figura 2.17), mientras que la fase a la que daremos el signo negativo y diremos que está polarizada inversamente estará recibiendo la corriente desde el semiconductor inferior correspondiente (Q5, en la figura 2.17).

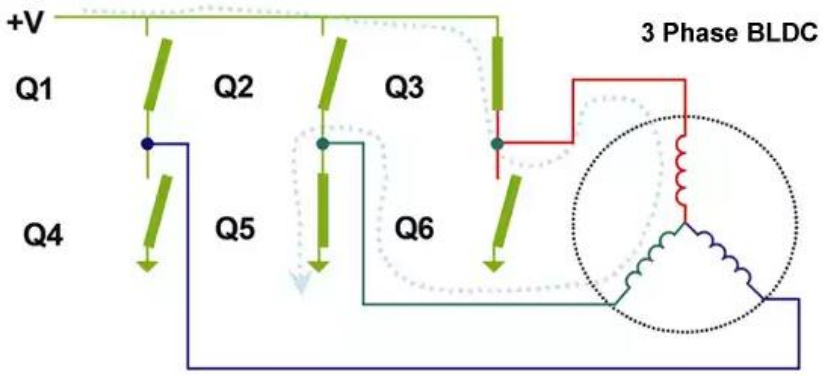

Figura 2.17 – Comportamiento del inversor en una etapa determinada de control

Dado que aplicaremos un control PWM en el inversor para regular la tensión que reciben las fases del motor, y por tanto la velocidad de giro de éste, existen tres posibilidades de aplicación:

- Superior: El PWM se aplica únicamente en el semiconductor superior, mientras que el inferior se mantiene en conducción durante todo la etapa.
- Inferior: El PWM se aplica únicamente en el semiconductor inferior, mientras que el superior se mantiene en conducción durante toda la etapa.
- Ambos: El PWM se aplica tanto en el semiconductor superior como en el semiconductor inferior.

En este método, el PWM se va a aplicar únicamente en los semiconductores superiores, y la fuerza contraelectromotriz se va a medir durante el periodo en el que la señal PWM

se encuentre en off, es decir, la situación sea la que se observa esquematizado en la figura 2.18, en la que hay uno de los semiconductores inferiores en estado de conducción y los tres semiconductores superiores están en estado abierto.

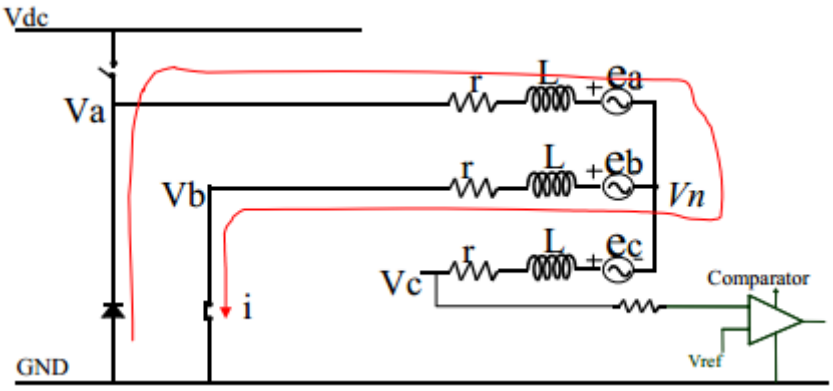

Figura 2.18 – Estado de las fases del motor en el instante en el que el PWM está en OFF

En particular en la etapa que se muestra en la figura 2.18, la fase A se encuentra controlada por el PWM en la parte superior del inversor, mientras que la fase B se encuentra controlada por la parte inferior, que se encuentra en conducción durante toda la etapa. La fase C se encuentra flotante, y por tanto es en la que deberemos medir la fuerza contraelectromotriz para detectar la posición del rotor.

La principal diferencia entre el esquema planteado en el método convencional y el planteado en este método es que ya no interviene la tensión del neutro del motor en la adquisición de señal.

El objetivo es medir la tensión existente entre el terminal del devanado C y el neutro del motor, que al estar flotante, la corriente que circula es nula, por lo que se puede deducir fácilmente que:

$$V_C = V_N + e_C \tag{2.3}$$

Donde e_C es la fuerza contraelectromotriz que necesitamos medir.

Si realizamos un análisis de la fase A, ignorando la caída de tensión en el diodo, tenemos que:

$$V_N = 0 - r \cdot i - L \cdot \frac{di}{dt} - e_A \tag{2.4}$$

Por su lado, analizando la fase B, tenemos que:

$$V_N = r \cdot i + L \cdot \frac{di}{dt} - e_B \tag{2.5}$$

Si sumamos las expresiones 2.5 y 2.6, tenemos que:

$$V_N = -\frac{e_A + e_B}{2} \tag{2.6}$$

Asumiendo que el sistema está bien equilibrado, la suma de la fuerza contraelectromotriz de las tres fases será igual a la suma de los armónicos, que por simplicidad se despreciarán todos los que tengan un orden mayor de tres:

$$e_A + e_B + e_C = e_3 \quad (2.7)$$

Considerando las expresiones anteriores, tenemos que:

$$V_N = \frac{e_C - e_3}{2} \quad (2.8)$$

Sustituyendo el valor de V_N en la expresión 2.3, tenemos que:

$$V_C = e_C + V_N = \frac{3 \cdot e_C - e_3}{2} \quad (2.9)$$

No obstante, debemos ser conscientes de que, a la hora de utilizar la fuerza electromotriz detectada, el punto principal que se desea detectar es el cruce con el cero. Como el cruce por cero coincide en la onda fundamental y en el tercer armónico, este último no va a afectar al cruce por cero de la onda fundamental, por lo que podemos afirmar que en entornos cercanos al cruce por cero se cumple que:

$$V_C = \frac{3}{2} \cdot e_C \quad (2.10)$$

2.4.5. Método propuesto

El método que se va a proponer en el presente proyecto es una evolución del método de detección basado en la comparación con GND. En el método presentado en el apartado anterior, existe un problema fundamental, que es que es necesario un mínimo tiempo en el que el PWM esté en OFF, es decir, durante el funcionamiento no se podrá llegar a un 100% de tiempo de ciclo, lo que directamente significa que no se podrá utilizar toda la tensión disponible en la batería para accionar el motor, debiendo ser ésta sobredimensionada.

Evidentemente, en una aplicación como la de un vehículo de propulsión eléctrica, para conseguir una máxima eficiencia es imprescindible que se pueda aprovechar toda la potencia de la batería, y no haya que sobredimensionarla, añadiendo peso muerto al sistema que disminuye el rendimiento.

Para resolver este problema, se plantea un sistema alternativo que mejora el anteriormente mencionado, pudiendo medirse la fuerza contraelectromotriz también durante el periodo en ON del PWM.

En este caso, analizamos lo que sucede durante el periodo en el que la zona superior del inversor se encuentra en conducción. El esquema del motor es el que se muestra en la figura 2.19.

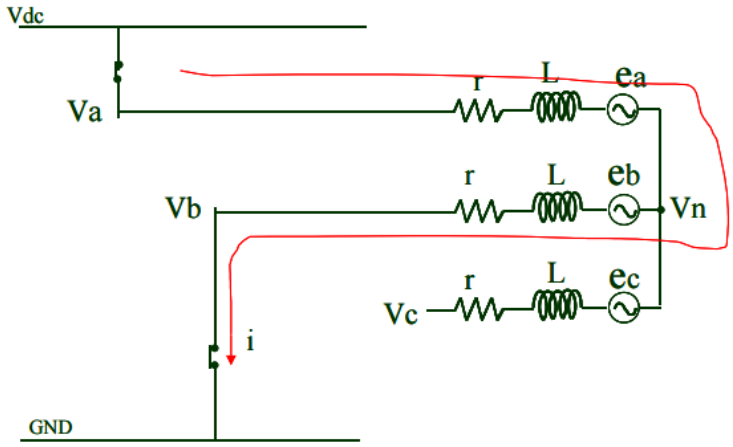

Figura 2.19 – Estado de las fases en el instante en que el PWM está en ON

Siguiendo la fase A, tenemos que la tensión en el neutro será:

$$V_N = V_{DC} - V_{mos} - r \cdot i - L \cdot \frac{di}{dt} - e_A \tag{2.11}$$

Por otro lado, en la fase B tendremos:

$$V_N = V_{mos} + r \cdot i + L \cdot \frac{di}{dt} - e_B \tag{2.12}$$

Donde V_{mos} es la caída de tensión en el transistor MOSFET cuando está en conducción.

Si sumamos las expresiones 2.11 y 2.12, obtenemos:

$$V_N = \frac{V_{DC}}{2} - \frac{e_A + e_B}{2} \tag{2.13}$$

Al igual que en el apartado anterior, dado que tenemos un sistema equilibrado, lo que significa que, ignorando los terceros armónicos al igual que en el apartado anterior:

$$e_A + e_B + e_C = 0 \tag{2.14}$$

Sustituyendo en la ecuación 2.13, tenemos que

$$V_N = \frac{V_{DC}}{2} + \frac{e_c}{2} \tag{2.15}$$

Con esto, tenemos que la tensión en el terminal de la fase C será:

$$V_C = e_c + V_N = \frac{3}{2} e_c + \frac{V_{DC}}{2} \tag{2.16}$$

De la ecuación 2.16 podemos extraer que la tensión en el terminal de la fase es igual a la fuerza contraelectromotriz que deseamos medir más la mitad de la tensión del bus. Tal y como vemos en la ecuación, si comparamos V_C con $V_{DC}/2$ seremos capaces de detectar el paso por cero.

En ocasiones, será necesario atenuar la señal para reducir la señal de modo común. No obstante, como este método se suele implementar para ciclos de funcionamiento altos, que corresponden con velocidades de giro altas que se traducen en una potente señal de fuerza contraelectromotriz producida, por lo que la atenuación no afectará a la sensibilidad de la señal.

Para la implementación de este método, no es imprescindible que el PWM se aplique únicamente en el lado superior del inversor, sino que permite que se aplique simultáneamente en ambos lados. Estudiando las dos posibilidades en la simulación, de la que se hablará en el capítulo 3, se ha llegado a la conclusión de que la respuesta del sistema es mucho mejor cuando aplicamos el PWM en ambas ramas del inversor, porque la salida presenta un mayor grado de estabilidad.

En un sistema complejo, lo ideal es implementar los dos métodos, de forma que a partir de un determinado tiempo de ciclo la detección se produzca en el instante en el que el PWM está en ON, mientras que por debajo de ese límite la detección se producirá en el estado OFF.

2.4.6. Implementación física

Una vez hemos obtenido una expresión válida que relaciona la tensión entre el terminal de cada una de las fases y la tensión de referencia con la fuerza contraelectromotriz, el objetivo es diseñar un sistema electrónico síncrono que sea capaz de detectar el paso por cero de esta señal.

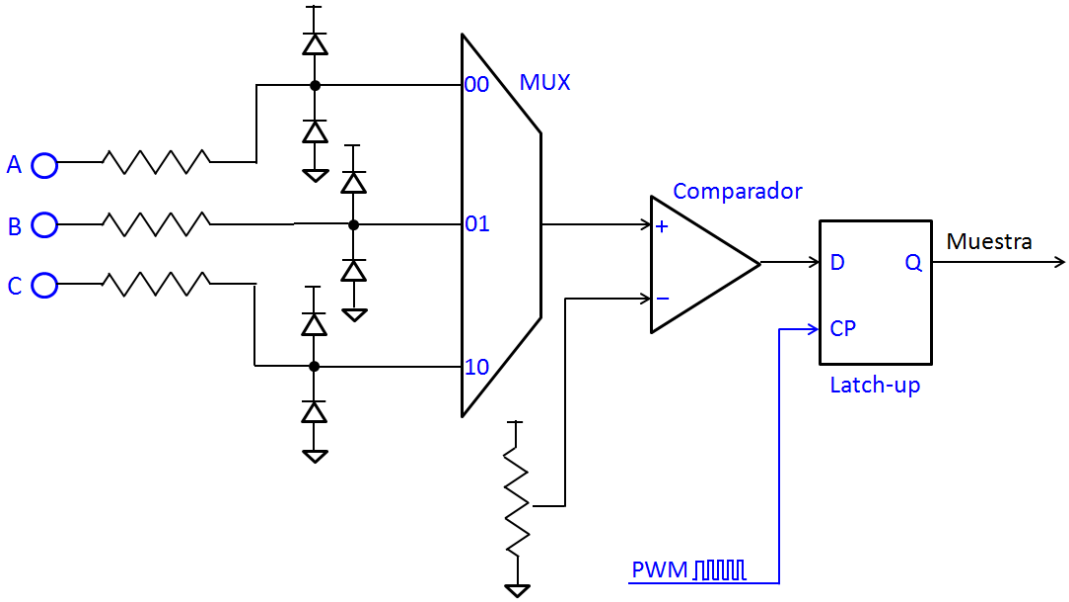

Figura 2.20 – Implementación física del circuito de detección de la fuerza contraelectromotriz

En la figura 2.20 podemos ver el esquema del detector de cruce con cero de la fuerza contraelectromotriz. La tensión medida en las fases pasa a través de un multiplexor, que selecciona cual es la fase a la que debe aplicarse la detección en función de la etapa de control en la que se encuentre el sistema.

Dado que el punto de interés de la fuerza contraelectromotriz es el punto de cruce por cero, se utilizan diodos para limitar el voltaje de pico a 5V, protegiendo así el comparador de sobretensiones manteniendo intacta la sensibilidad del sistema (la señal no se atenúa, sino que se trunca la zona sin interés). La señal medida en las fases se comparará con una tensión de referencia fija, que deberá ser próxima a 0V.

El sistema monitoriza y compara la fuerza contraelectromotriz durante el tiempo en el que el PWM está en estado OFF. En el momento que éste pasa a ON, se produce un flanco de subida en la señal que es utilizada en el Latch-Up para capturar la información acerca del cruce por cero de la fuerza contraelectromotriz.

A partir de la detección de paso por cero de la fuerza contraelectromotriz, la conmutación se producirá 30º eléctricos después. Lógicamente, esos 30º se calcularán como una estimación a partir del conocimiento de la velocidad de giro actual del motor.

2.4.7. Limitaciones del método

El método de detección de la fuerza contraelectromotriz propuesto está diseñado para funcionar adecuadamente en la mayoría de condiciones y aplicaciones. No obstante, existen algunos usos y aplicaciones que requieren de modificaciones en el método para conseguir un funcionamiento adecuado manteniendo un buen rendimiento energético del sistema.

En primer lugar, a bajas velocidades, la fuerza contraelectromotriz tiene una amplitud baja, lo que puede disminuir la sensibilidad del sistema y provocar que el paso por cero detectado no corresponda exactamente con la mitad de la etapa de control, lo que puede generar desfases en la conmutación que, con altas cargas, pueden provocar el calado del motor. Además, una mala conmutación puede provocar una regulación de velocidad deficiente si la realimentación de ésta se basa en la detección de paso por cero de la fuerza contraelectromotriz.

La principal razón de los desfases en la detección del paso por cero es que, si la fuerza contraelectromotriz es suficientemente pequeña, la caída de tensión en los diodos del inversor puede modificar significativamente las tensiones medidas.

Una posible solución para este fenómeno es aplicar el PWM en ambas ramas del inversor (superior e inferior), con el objetivo de evitar que la corriente atraviese el diodo durante el ciclo en que el PWM se encuentre en estado de OFF. Con esta estrategia, la corriente siempre será conducida a través de los transistores, cuya caída de tensión será mucho menor que la de los diodos.

Otra posible solución, para evitar tener que modificar el algoritmo, es sumar una tensión que compense la caída en el diodo previamente a introducir la señal medida en el comparador de paso por cero.

En segundo lugar, en aplicaciones de alto voltaje, aparece un retardo en la señal, debido a las grandes resistencias que son necesarias para limitar la corriente y evitar dañar el microcontrolador.

En la implementación física del método propuesto, se observa que la tensión de las fases se introduce en el microcontrolador directamente a través de resistencias que limiten la corriente inyectada. Para distintas aplicaciones, necesitaríamos distintas resistencias que mantengan la corriente en unos límites razonables que no dañen los sistemas del microcontrolador.

No obstante, en determinadas aplicaciones, la tensión del bus puede llegar a alcanzar los $300\ V$, por lo que la resistencia debe alcanzar unos $150\ k\Omega$. Resistencias tan grandes en serie con las capacidades parásitas internas que se generan en los microcontroladores producen grandes constantes de tiempo RC, que producen un retardo en la señal medida, y por tanto en la conmutación de las bobinas con el método tradicional.

La solución a este problema será reducir la constante de tiempo RC. Dado que no se puede modificar la capacitancia, habrá que modificar la resistencia, por ejemplo consiguiendo que la descarga del condensador interno se produzca a través de una resistencia más pequeña (por ejemplo, poniendo un diodo y una resistencia en serie, y a su vez en paralelo con la resistencia limitadora).

No obstante, para la aplicación pretendida en el presente proyecto, es suficiente con el método propuesto en el apartado $2.4.6$, por lo que será el que se desarrolle en simulación y físicamente. Las implementaciones alternativas fruto de las limitaciones planteadas en el presente capítulo se dejan planteada como base para nuevos desarrollos de futuros proyectos.

3. Modelado y simulación en Simulink

3.1. Introducción

Previo paso a la construcción del circuito físico real, se va a construir un modelo digital, utilizando la herramienta Simulink de MATLAB, software de MathWorks.

MATLAB, si atendemos a la descripción general que ofrece la empresa propietaria y distribuidora de este software, es un *"lenguaje de alto nivel y un entorno interactivo para el cálculo numérico, la visualización y la programación. Mediante MATLAB, es posible analizar datos, desarrollar algoritmos o crear aplicaciones. El lenguaje, las herramientas y las funciones matemáticas incorporadas permiten explorar diversos enfoques y llegar a una solución antes que con hojas de cálculo o lenguajes de programación tradicionales, como pueden ser C/C++ o Java"*.

MATLAB, acrónimo de MATrix LABoratory, o laboratorio de matrices, surge en 1984, creado por Cleve Moler, un matemático y programador estadounidense especializado en análisis numérico. Moler ya había colaborado en los años 70 en el desarrollo de dos paquetes de software escritos en FORTRAN, LINPACK y EISPACK, que contenían funciones para el cálculo de operaciones con matrices y el cálculo de autovalores y autovectores, respectivamente. Moler había desarrollado MATLAB como un lenguaje de computación numérica para que sus estudiantes de la universidad de Nuevo México pudieran tener acceso a las funciones de estos dos paquetes sin la necesidad de aprender FORTRAN. Así, en el año 1984, Cleve Moler cofunda Mathworks con Jack Little y comienza a comercializar este programa.

Se estima que en 2004 había ya más de un millón de usuarios de MATLAB, en ámbitos académicos y empresariales.

Una de las mayores virtudes de MATLAB es su modularidad, lo que le ha permitido mantener un crecimiento constante, gracias tanto a la colaboración de usuarios independientes como de los desarrolladores, cuyo objetivo es expandir el rango de aplicaciones del programa, creando toda una infinidad de módulos para hacer frente a una gran variedad de problemas de distintos ámbitos. Estos módulos se agrupan en bibliotecas denominadas toolboxes, que extienden el entorno de trabajo original de MATLAB, que era el del cálculo matricial, a una infinidad de campos de la ingeniería y la ciencia.

Así, en la actualidad existen toolboxes para campos tan diversos como el procesamiento de señales, el diseño de sistemas de control, la simulación de sistemas dinámicos, las redes neuronales, etc. Hoy en día nos encontramos en un punto en el que existen más de 50 toolboxes, preinstaladas en la versión de 2015 de MATLAB, además de otra serie de bibliotecas de usuario disponibles en la red para descargar gratuitamente.

No obstante, si hablamos de toolboxes, sin duda la que más repercusión ha tenido en la evolución de las aplicaciones de MATLAB es Simulink. Simulink es el nombre que

MathWorks dio a su programa para simulación (modelización y análisis) de sistemas dinámicos no lineales, presentado en 1990. Su aparición estuvo vinculada a la primera versión de MATLAB para Windows, en mayo de 1994, en la que la versión 1.3 de MATLAB incorporaba la opción de instalar de forma separada la toolbox Simulink.

Figura 3.1 – Logo de MATLAB Simulink

La principal diferencia de Simulink respecto a otros toolboxes de MATLAB es que Simulink ofrece un entorno gráfico de ventanas, en los que el usuario puede definir los sistemas mediante un diagrama de bloques, de manera mucho más intuitiva que otros programas de simulación.

Utilizando Simulink, el usuario puede crear sus modelos en forma de diagramas de bloques de alto nivel partiendo de librerías de componentes básicos, utilizar componentes de toolboxes para aplicaciones específicas, y programar sus propios bloques para, estableciendo las conexiones oportunas y parametrizando el modelo adecuadamente, reducir en gran medida el tiempo de desarrollo si el modelo se creara mediante lenguajes de programación convencionales. Una vez creado el modelo, Simulink también permite ejecutar su análisis, pudiendo elegir diferentes métodos de integración.

3.2. Objetivos del modelado

Acorde con las técnicas de control descritas en el capítulo anterior, se va a desarrollar un modelo utilizando bloques de Simulink que permita al usuario seleccionar entre el control utilizando sensores Hall y utilizando la detección de la fuerza contraelectromotriz, pudiendo así comparar los resultados entre ambos sistemas.

Durante los siguientes subapartados, se analizará cada uno de los bloques de Simulink utilizados para modelar el sistema físico, partiendo de un modelo con bloques básicos, al que se le incorporará en primer lugar los elementos necesarios para el control utilizando sensores Hall y a en segundo lugar los necesarios para el control sin sensores.

El modelado y la simulación del sistema es una herramienta muy valiosa para el presente proyecto. Entre sus numerosas ventajas, debemos tener en cuenta:

- Supresión del riesgo de dañar los componentes.
- Manipulación total de los parámetros del modelo.
- Medición con herramientas gráficas y numéricas con las que analizar los resultados de la simulación.
- Monitorización en tiempo real de todos los valores del modelo.

- Variación de los horizontes temporales del modelo, desde unos pocos milisegundos hasta años.
- Aislamiento de variables, pudiendo modificar los parámetros de los componentes de manera individualizada, sin afectar al resto del modelo.

3.3. Punto de partida

Para la construcción del modelo, se partirá de un modelo prediseñado en Simulink, llamado *power_brushlessDCmotor*, que representa un motor brushless controlado mediante un controlador PI, que actúa sobre una fuente de tensión variable.

El aspecto del modelo del que partimos es el que aparece en la figura 3.2.

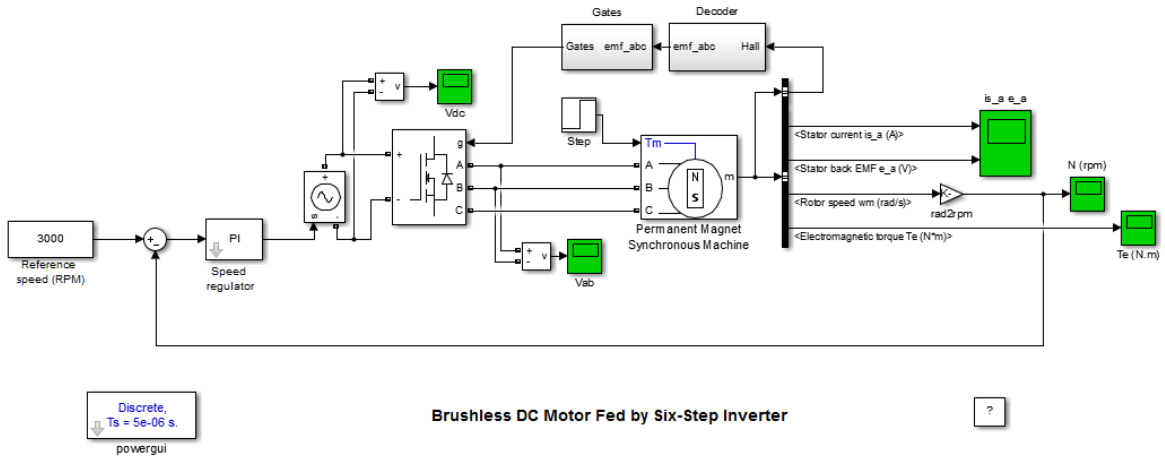

Figura 3.2 – Modelo *power_brushlessDCmotor*

Tal y como se observa en el diagrama de bloques, existe un controlador PI que, tomando una referencia de velocidad y la velocidad actual del motor, modifica la tensión suministrada por una fuente variable que alimenta el puente trifásico. Este puente trifásico genera la salida de tensión de cada una de las fases del motor BLDC, y está controlado mediante las lecturas de los sensores Hall, que atraviesan los bloques *Decoder* y *Gates*.

El bloque de la fuente de tensión variable está representando, de una manera básica y simplificada, una batería alimentando un convertidor DC/DC accionado electrónicamente mediante el controlador PI para modificar el nivel de tensión que alimenta el inversor trifásico.

El cambio más importante que hay que llevar a cabo en el modelo de control Hall respecto a éste, es el cambio de la fuente de tensión variable por una fuente de tensión fija, ya que en el sistema físico que se plantea en el presente proyecto no se dispone de ningún convertidor DC/DC, sino que la regulación de la velocidad se realiza aplicando un PWM en el inversor, modificando el porcentaje del tiempo de ciclo que conduce, variando así la tensión eficaz que recibe el motor, y por tanto su velocidad de giro.

3.4. Modelo común

El primer paso para la construcción del modelo será parametrizar adecuadamente tres elementos fundamentales del sistema que van a permanecer invariantes independientemente de la técnica de control que se desee utilizar. Estos son la batería, el inversor trifásico y el motor BLDC.

3.4.1. Batería

Como fuente de tensión fija, se va a utilizar el bloque *battery*. Este tipo de alimentación es el utilizado en numerosas aplicaciones de los motores BLDC, como es el caso de los vehículos eléctricos. De la batería, se va a extraer información de utilidad, monitorizando las variables de interés mediante un bloque *scope*. En concreto, se va a extraer la tensión y corriente suministrada, así como el porcentaje de carga de la batería (*SOC – State Of Charge*).

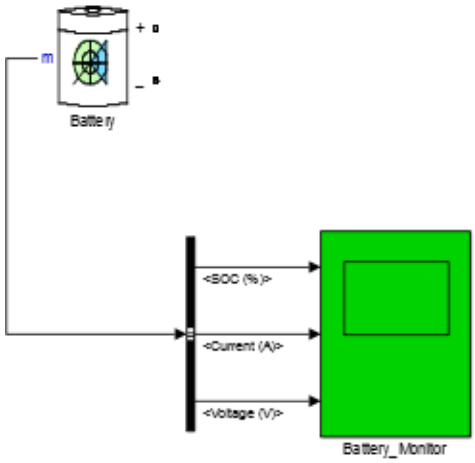

Figura 3.3 – Bloque *Battery*

En cuanto a los parámetros de la batería, se ha configurado como una batería de ión de litio, con una tensión nominal de 24 V y una capacidad de 6,5 Ah, como se muestra en la figura 3.4.

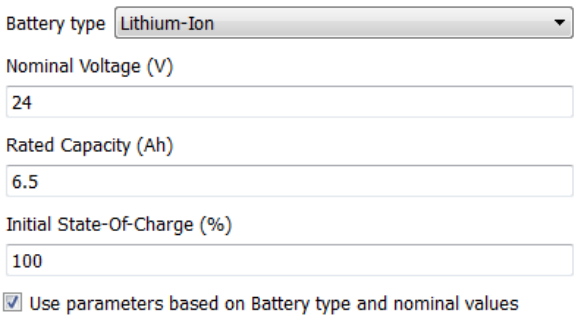

Figura 3.4 – Parámetros bloque *battery*

Dado que la selección de una batería no es objeto de el presente proyecto, y todos los ensayos de laboratorio se llevarán a cabo utilizando fuentes de alimentación externas, no se ha pensado en ningún modelo de batería en concreto, de cuyo datasheet poder extraer los parámetros avanzados para el configurar el bloque, por lo que se

seleccionará la opción de autocompletar los parámetros del bloque basándose en los valores nominales y tipo de la batería.

3.4.2. Inversor

Como inversor, se ha seleccionado el bloque *Universal bridge*, que tiene como entradas la tensión de la batería y un vector de señales digitales de control que entra en el bloque a través de la entrada *g*. Como salidas, aparecen las tensiones de las fases del motor.

El aspecto del bloque es el mostrado en la figura 3.5.

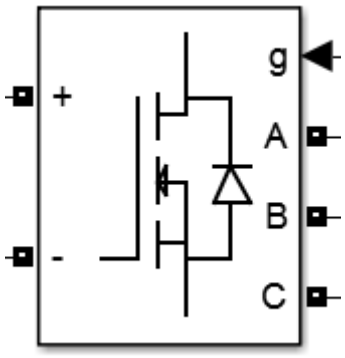

Figura 3.5 – Bloque *Universal bridge*

Las opciones que se deberán configurar en este caso son, principalmente, el número de fases del puente, que en este caso será trifásico, y el tipo de dispositivo semiconductor, que será MOSFET con sus correspondientes diodos de protección.

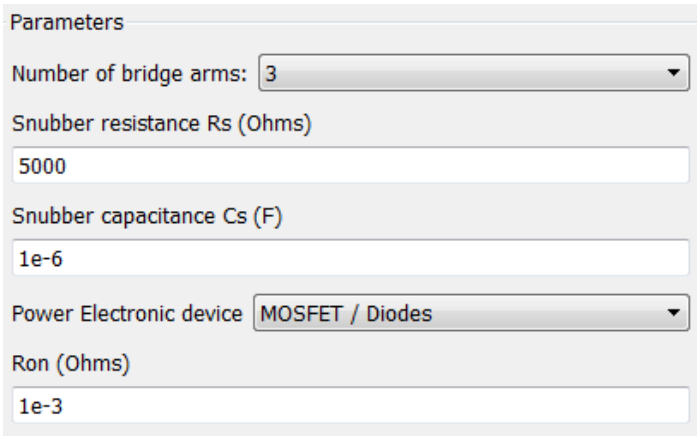

Figura 3.6 – Parámetros bloque *Universal bridge*

3.4.3. Motor BLDC

El bloque correspondiente al motor BLDC estará modelado mediante el bloque *Permanent Magnet Synchronous Machine*. Como entradas, el motor tiene el torque al que se le está sometiendo al eje, además de las tres tensiones de las fases, mientras que como salida del bloque tenemos un vector con todos los parámetros del motor, la corriente del estator, la fuerza contraelectromotriz del estator, la velocidad de giro del rotor y el torque generado.

El bloque es el que se muestra en la figura 3.7.

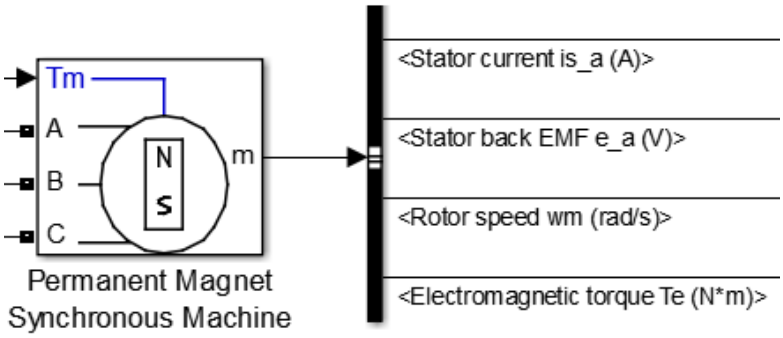

Figura 3.7 – Bloque *Permanent Magnet Synchronous Machine*

Será muy importante parametrizar adecuadamente el bloque, para que a la hora de realizar la simulación los resultados obtenidos concuerden de la mejor manera posible con los resultados del circuito físico. Además del número de fases, se va a utilizar la opción de computar los parámetros del bloque a partir de los datos estándar del fabricante, para poder introducir los datos que aporta el datasheet directamente en el bloque. Como referencia, se ha tomado el motor BLDC Hurst DMB0224C10002, un BLDC de pequeño tamaño.

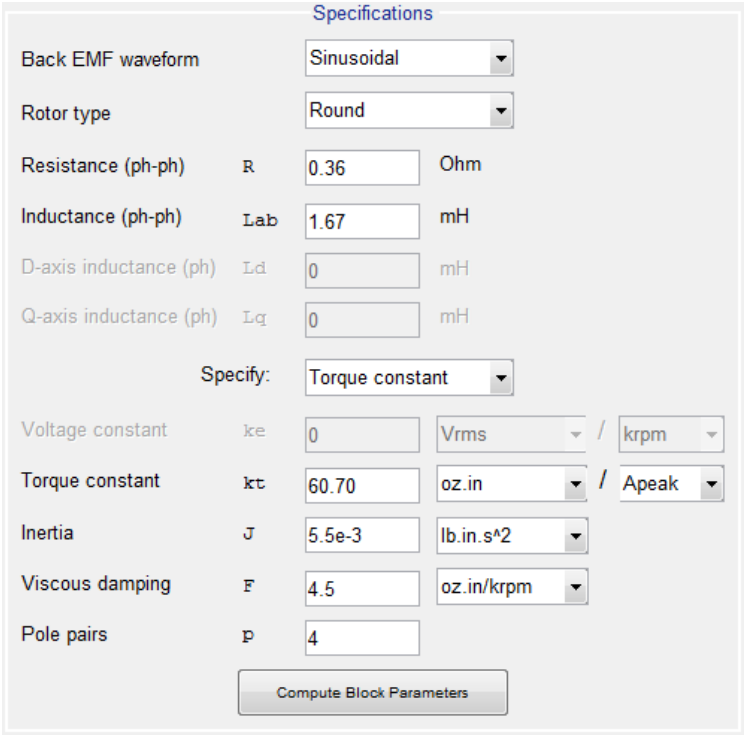

Figura 3.8 – Parámetros bloque *Permanent Magnet Synchronous Machine*

3.5. Modelo utilizando sensores

El aspecto del modelo modificado para utilizar los sensores Hall y un PWM en el inversor es el que se muestra en la figura 3.9.

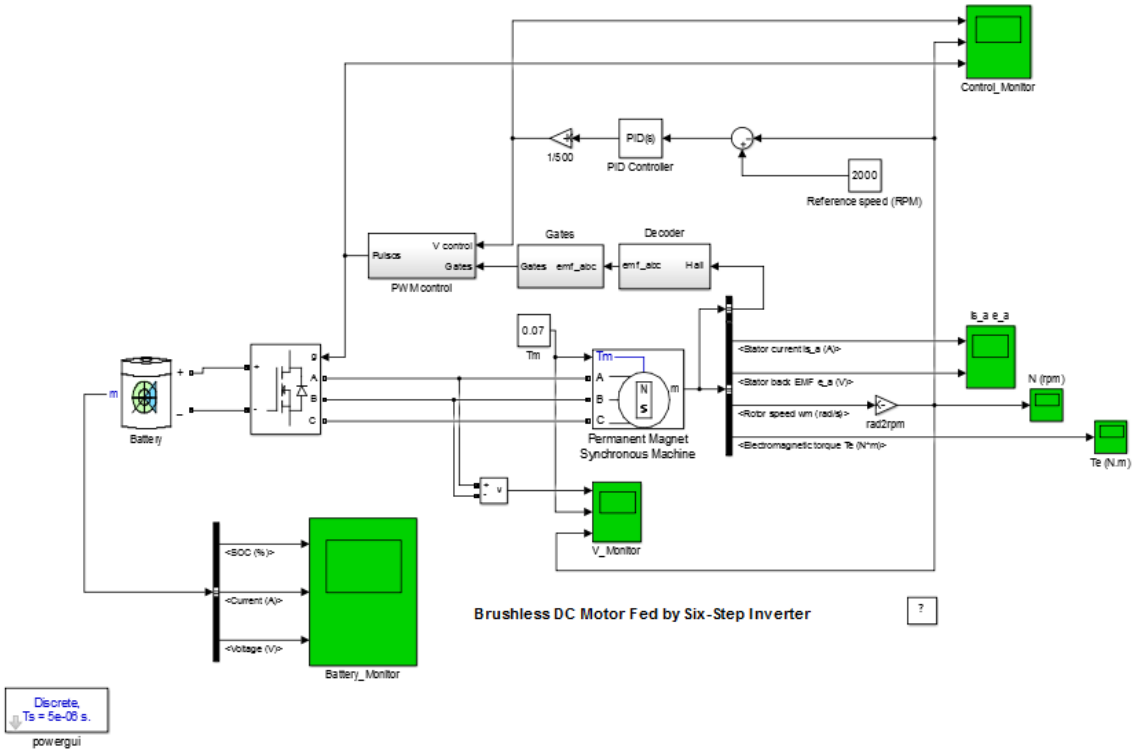

Figura 3.9 – Modelo para control utilizando sensores Hall

El sistema de control del modelo está compuesto por varios bloques, que se analizarán durante los siguientes subapartados.

3.5.1. Bloque *Decoder*

El bloque *decoder* recibe como entrada un vector con las lecturas de los sensores Hall, que denominamos ha, hb y hc, mientras que como salida aparecen las señales que activan cada fase del motor, que denominaremos emf_a, emf_b y emf_c.

El aspecto del bloque se muestra en la figura 3.10.

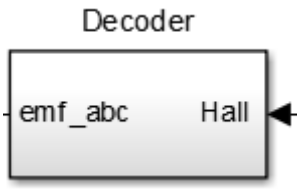

Figura 3.10 – Bloque *Decoder*

Si atendemos a la configuración interna del bloque, podemos obtener su tabla lógica de la figura 3.11.

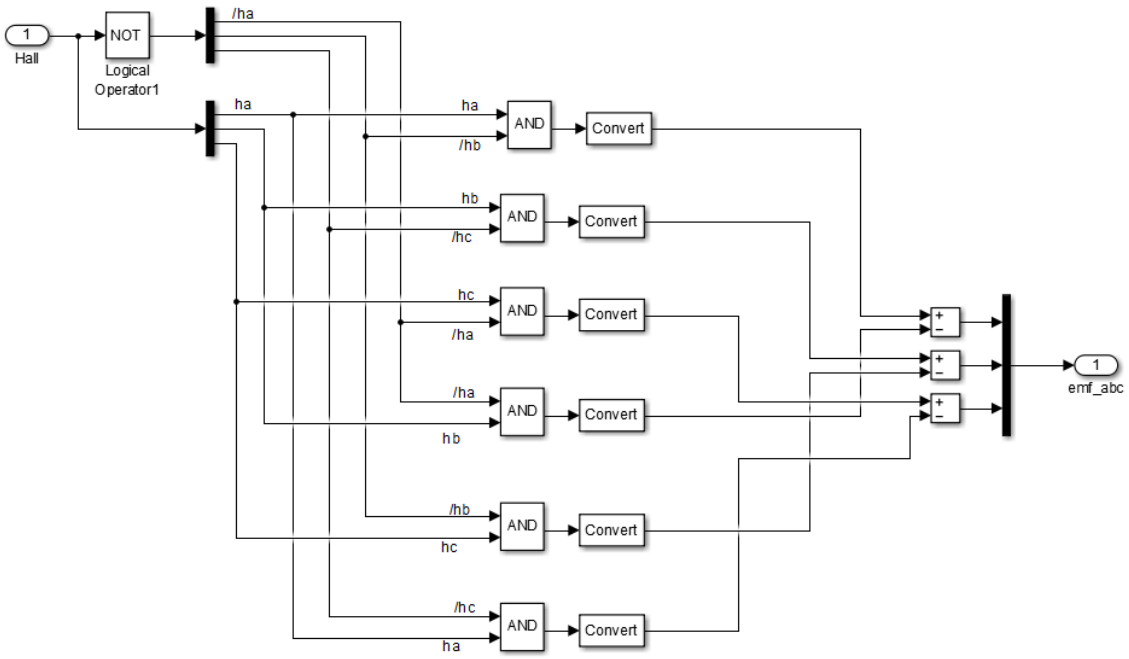

Figura 3.11 – Configuración interna del bloque *Decoder*

La tabla lógica del bloque que se extrae de la configuración interna es la mostrada en la figura 3.12.

ha	hb	hc	emf_a	emf_b	emf_c
0	0	0	0	0	0
0	0	1	0	-1	+1
0	1	0	-1	+1	0
0	1	1	-1	0	+1
1	0	0	+1	0	-1
1	0	1	+1	-1	0
1	1	0	0	+1	-1
1	1	1	0	0	0

Figura 3.12 – Tabla lógica del bloque *Decoder*

La función de este bloque es, como su propio nombre indica, de decodificar las señales de tipo Hall, y transformarlas en las señales de activación de las bobinas.

Cada una de las bobinas podrá estar alimentada con tensión positiva o negativa, o no estar alimentada, en función de la posición del imán del rotor, para que el motor gire correctamente.

3.5.2. Bloque *Gates*

El vector de salida del bloque *Decoder* se introducirá en el bloque *Gates*, que transforma las variables que indican los estados de las bobinas en señales lógicas que activan o desactivan los transistores superiores o inferiores del puente trifásico. Estas señales lógicas formarán un vector de seis señales, que representarán la salida del bloque.

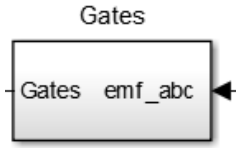

Figura 3.13 – Bloque *Gates*

El circuito interno de este bloque se muestra en la figura 3.14.

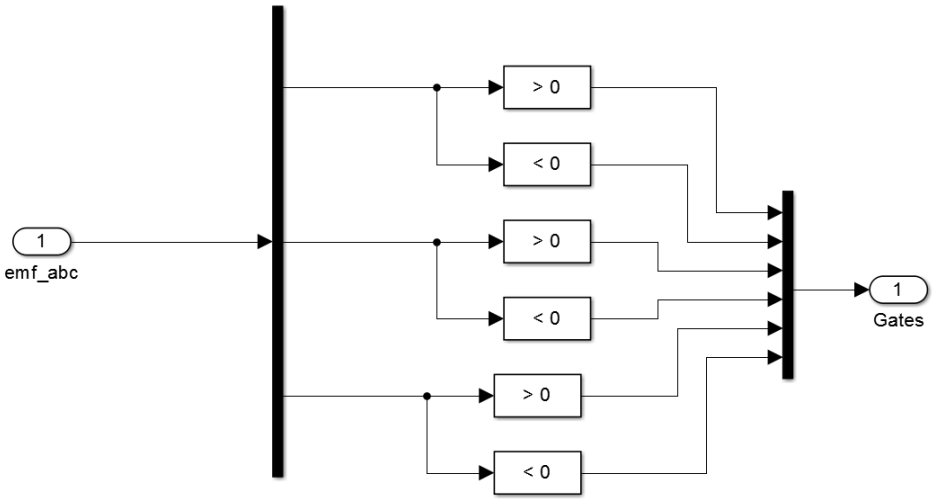

Figura 3.14 – Configuración interna del bloque *Gates*

La tabla lógica que implementa este bloque se muestra en la figura 3.15.

emf_a	emf_b	emf_c	Q1	Q2	Q3	Q4	Q5	Q6
0	0	0	0	0	0	0	0	0
0	-1	+1	0	0	0	1	1	0
-1	+1	0	0	1	1	0	0	0
-1	0	+1	0	1	0	0	1	0
+1	0	-1	1	0	0	0	0	1
+1	-1	0	1	0	0	1	0	0
0	+1	-1	0	0	1	0	0	1
0	0	0	0	0	0	0	0	0

Figura 3.15 – Tabla lógica del bloque *Gates*

Las señales Q1, Q3 y Q5 representan los polos superiores de las fases a, b y c respectivamente en el puente trifásico; mientras que las señales Q2, Q4 y Q6 representan los polos inferiores de las fases a, b y c en el puente trifásico.

3.5.3. Bloque *PWM control*

Las señales que salen del bloque *Gates*, por sí mismas ya sirven para activar las fases del motor, pero suministrando siempre toda la tensión disponible en la batería, por lo que no es posible el control de velocidad.

Para permitir el control de la velocidad, lo que se insertará es un bloque con un control PWM, que mediante el control del porcentaje tiempo de ciclo que los transistores se mantienen conduciendo, permitirá variar la tensión eficaz que reciben las bobinas del motor, y por lo tanto controlar la velocidad de giro de éste.

El bloque insertado tiene como entradas el vector de seis variables que viene del bloque *Gates*, así como la variable *V_control*, correspondiente a la salida del controlador PID implementado.

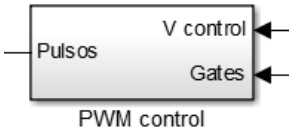

Figura 3.16 – Bloque PWM control

El aspecto interno del bloque aparece en la figura 3.17.

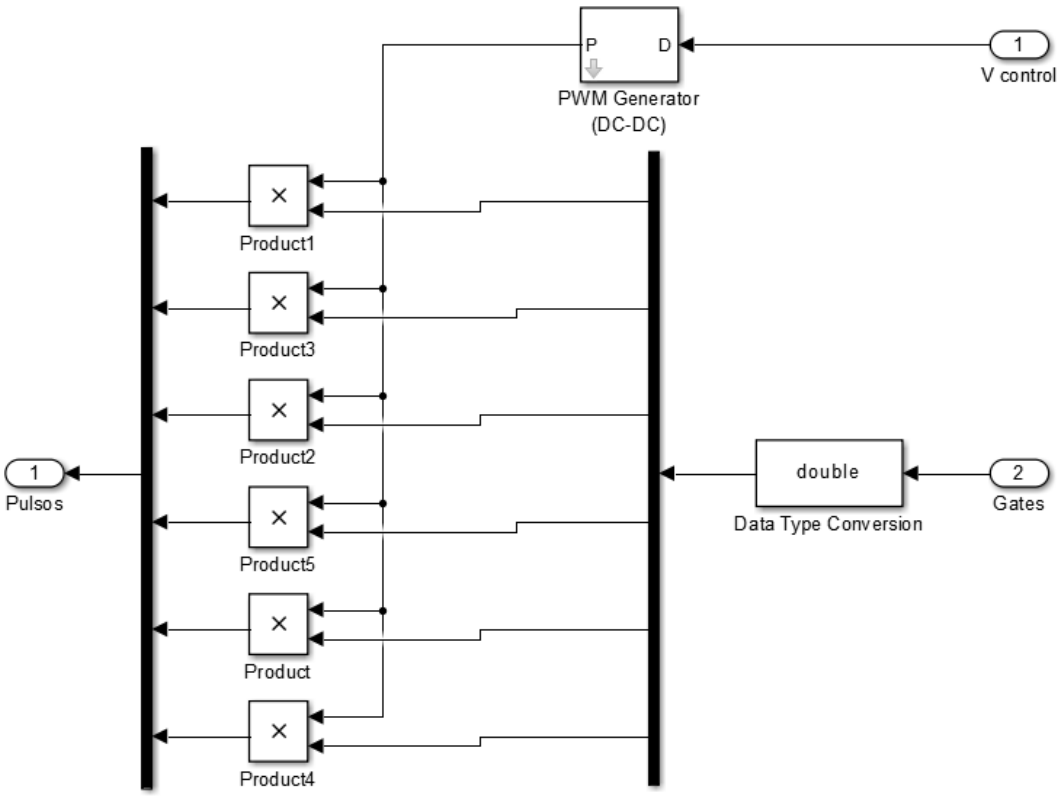

Figura 3.17 – Configuración interna del bloque *PWM control*

Para asegurar una mejor estabilidad en la velocidad de respuesta, se aplicará el PWM a todos los polos, tanto superiores como inferiores, ya que en el método propuesto no es imprescindible la aplicación del PWM únicamente en la parte superior del inversor, tal y como se indica en el apartado 2.4.5.

Debido a que las salidas del bloque *gates* son de tipo booleano, es necesario transformarlas en *double* para que no aparezcan errores al utilizar el bloque producto.

3.5.4. Controlador PID

La señal V control que se introduce en el bloque *PWM control* se genera a partir de un controlador PID, que toma como entrada la diferencia entre la referencia de velocidad dada y la actual del motor.

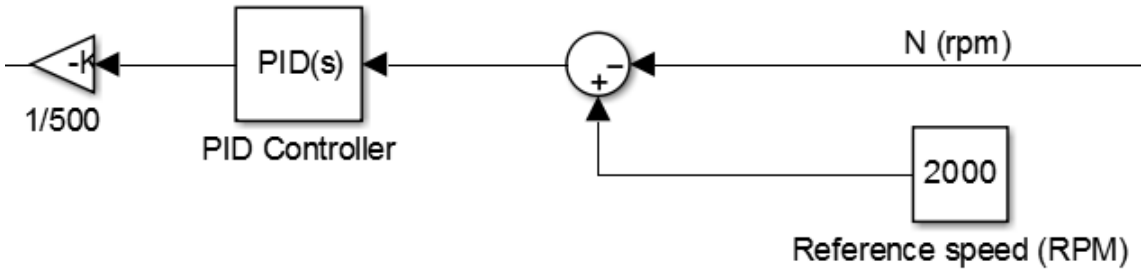

Figura 3.18 – Controlador PID

La sintonización del PID se realiza aprovechando la simulación, variando los parámetros del controlador y observando la respuesta. Los parámetros resultantes aparecen en la figura 3.19.

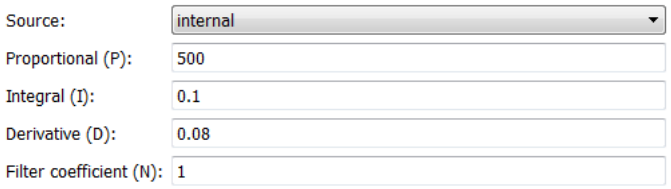

Figura 3.19 – Parámetros del PID

Estos parámetros son susceptibles de modificarse cuando se pruebe en el sistema físico real, pero suponen un punto de partida clave para iniciar estas pruebas.

3.6. Modelo utilizando la fuerza contraelectromotriz

Partiendo del modelo en el que se utiliza la lectura de los sensores Hall para controlar la posición del rotor en el motor, se va a añadir la posibilidad de controlar el motor utilizando el método sin sensores propuesto en el capítulo anterior. Al modelo existente, manteniendo aspectos como la batería, el inversor, el motor y el sistema de control mediante sensores Hall, se le han añadido bloques de usuario que conforman un nuevo sistema de control implementando el método *sensorless* propuesto.

El sistema permitirá que, mediante un selector, se pueda seleccionar entre control Hall o *sensorless*. En la figura 3.20 se muestra el aspecto del sistema completo.

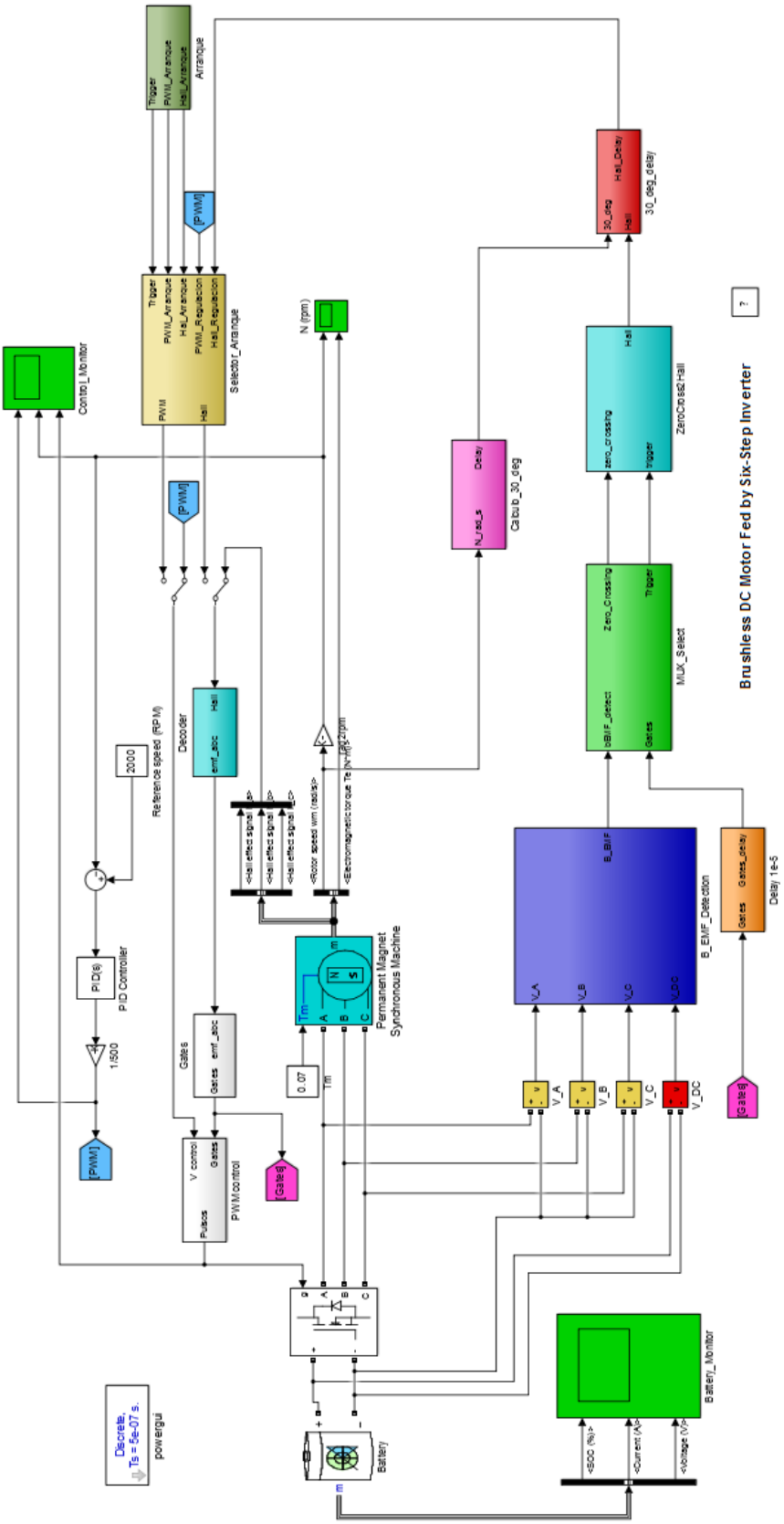

Figura 3.20 – Sistema de control de velocidad del motor BLDC completo

Las funciones que desempeñan los bloques añadidos son:

a) Detectar la fuerza contraelectromotriz de las fases a partir de la tensión de los bornes del motor.
b) Analizar cuál es la lectura que hay que realizar en función de la etapa de control en la que nos encontremos, y detectar el momento de paso por cero.
c) Activar los mecanismos de computación 30º eléctricos después de que se detecte el paso por cero.

A continuación, se analizarán los bloques uno por uno, explicando el algoritmo que implementan y su función dentro del sistema de control.

3.6.1. Bloque *B_EMF_Detection*

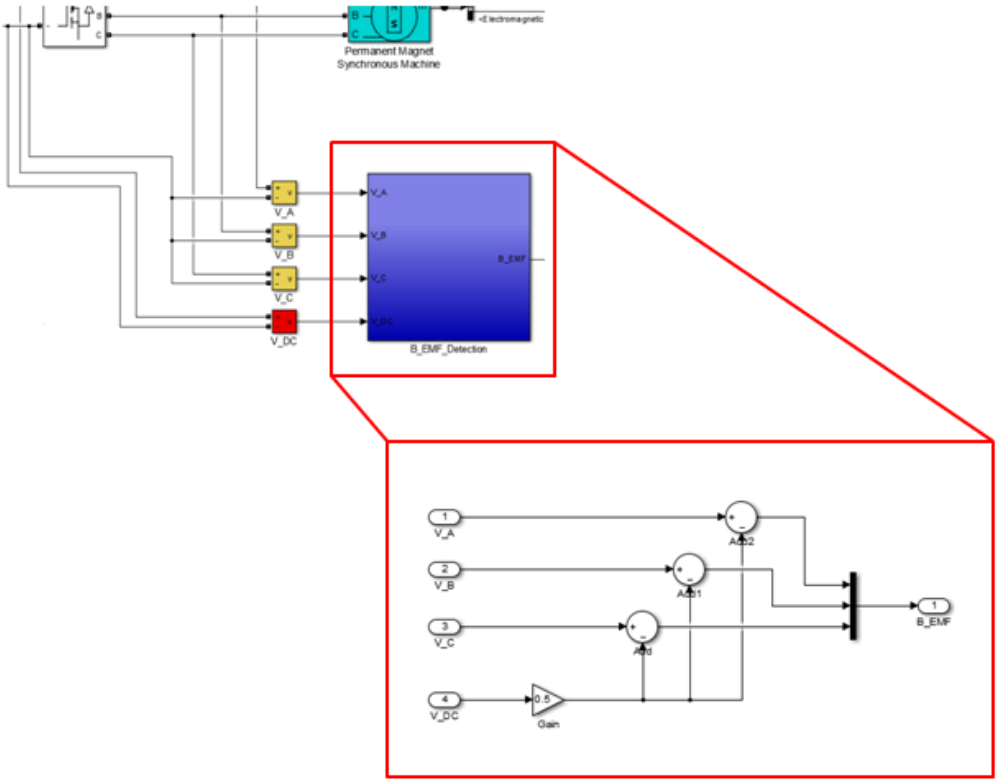

Figura 3.21 – Bloque *B_EMF_Detection* con su configuración interna

Este bloque toma las tensiones de cada una de las fases y la de la batería, medidas respecto de GND, y realiza la comparación entre cada una de las fases y $V_{DC}/2$, tal y como se explica en el modelo teórico.

En la salida de este bloque podemos apreciar la fuerza contraelectromotriz de cada una de las fases, superpuesta a la señal de PWM del controlador, de forma que se puede detectar perfectamente el paso por cero de la bobina que se encuentra abierta en cada etapa de control.

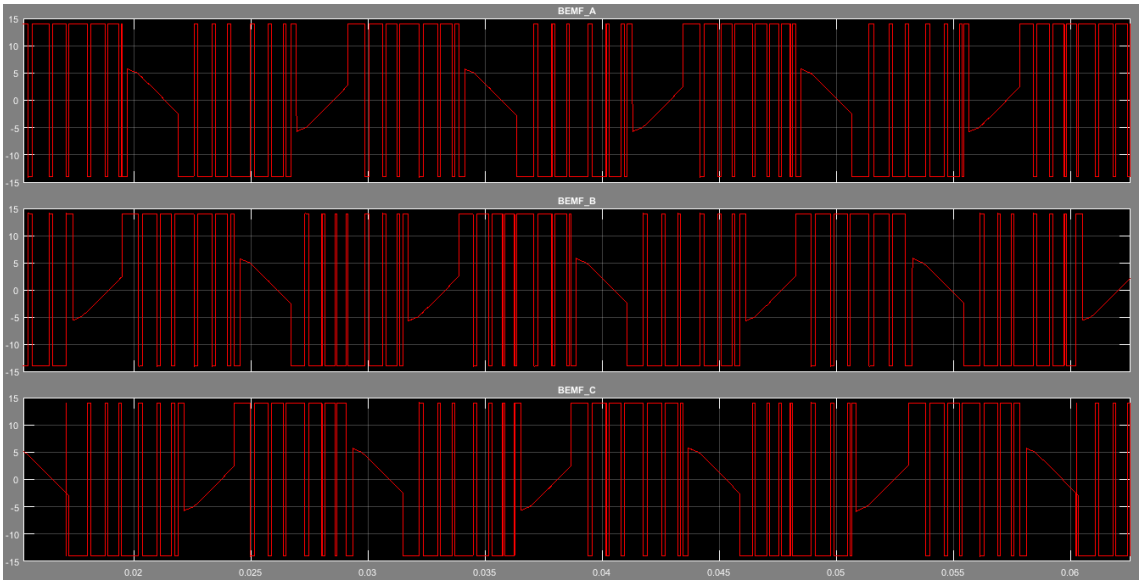

Figura 3.22 – Salida del bloque *B_EMF_Detection*

El siguiente paso será, a cada instante, detectar el cruce por cero de la señal BEMF que corresponda a cada momento.

3.6.2. Bloque *MUX_Select*

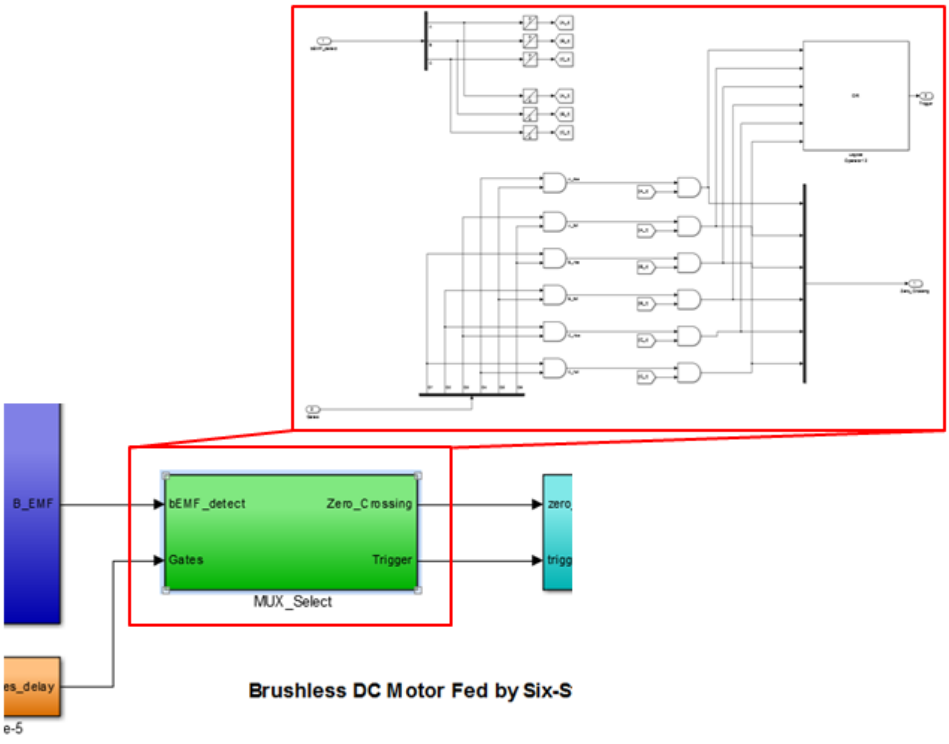

Figura 3.23 – Bloque *MUX_Select* con su configuración interna

El objetivo de este bloque es, por un lado, detectar en que etapa del control nos encontramos, mediante la observación de los estados de excitación de los transistores que forman el inversor (Q1-Q6), y por otro, utilizar esta información para decidir, en cada momento, a que fase se debe prestar atención, y si se va a medir un flanco creciente o

decreciente, en función de que la bobina abierta vaya a pasar de -1 a +1 (flanco creciente) o viceversa (flanco decreciente).

Como salidas del bloque, tenemos un vector de señales lógicas, cuya activación se corresponde con el paso por cero en cada una de las etapas de control, y por el otro una señal de activación o *trigger*, que se activa en el momento en que se ha observado algún paso por cero válido. En la figura 3.24 siguiente aparece representado el vector *zero_crossing*.

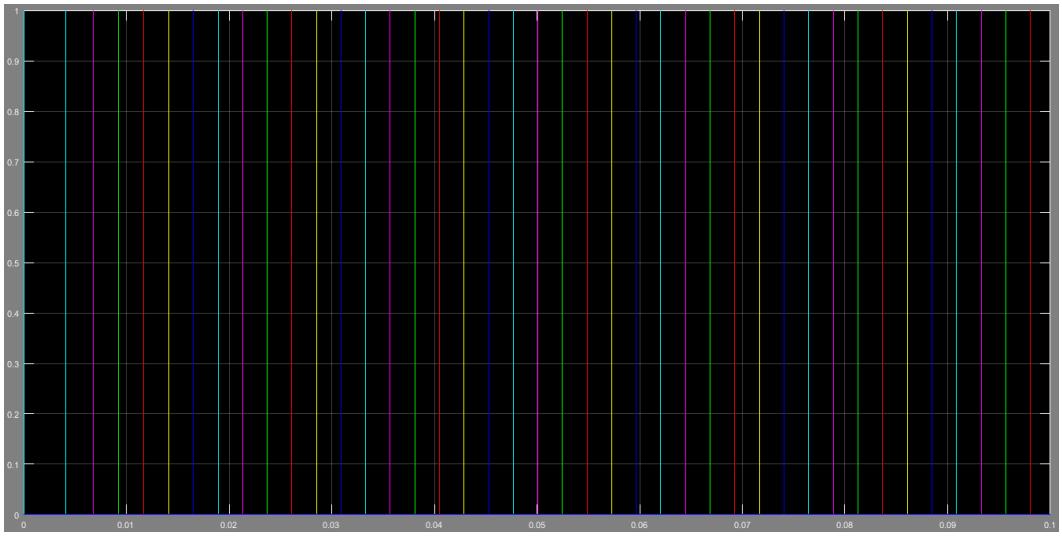

Figura 3.24 – Representación gráfica del vector *zero_crossing*

3.6.3. Bloque *Delay 1e-5*

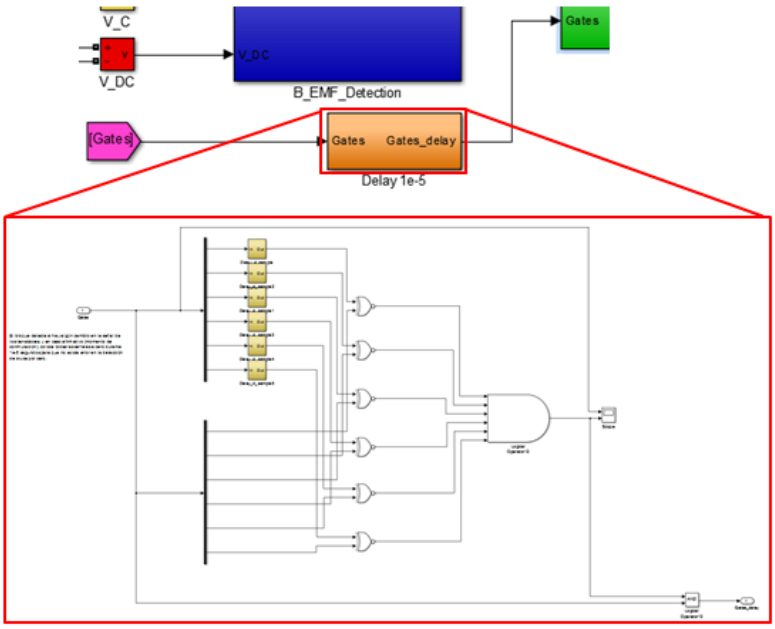

Figura 3.25 – Bloque *Delay 1e-5* con su configuración interna

El objetivo de este bloque es conseguir una pequeña zona muerta alrededor del instante de conmutación entre etapas de control, debido a que estas zonas son propensas a

producir un falso positivo (detectar un cruce por cero que no se corresponde con la fuerza contraelectromotriz).

Este bloque toma las señales de activación de los transistores, y las compara con una versión ligeramente retardada (4 unidades básicas temporales). Al detectar un cambio de etapa de control, representada por un cambio de estado en alguna de las señales de control del transistor, se crea un intervalo de tiempo de 4 unidades básicas en las que la señal normal y la retardada no coinciden. Durante este intervalo de tiempo, las señales de las puertas que entran en el bloque *MUX_Select* se anulan, evitando así los falsos positivos.

3.6.4. Bloque *ZeroCross2Hall*

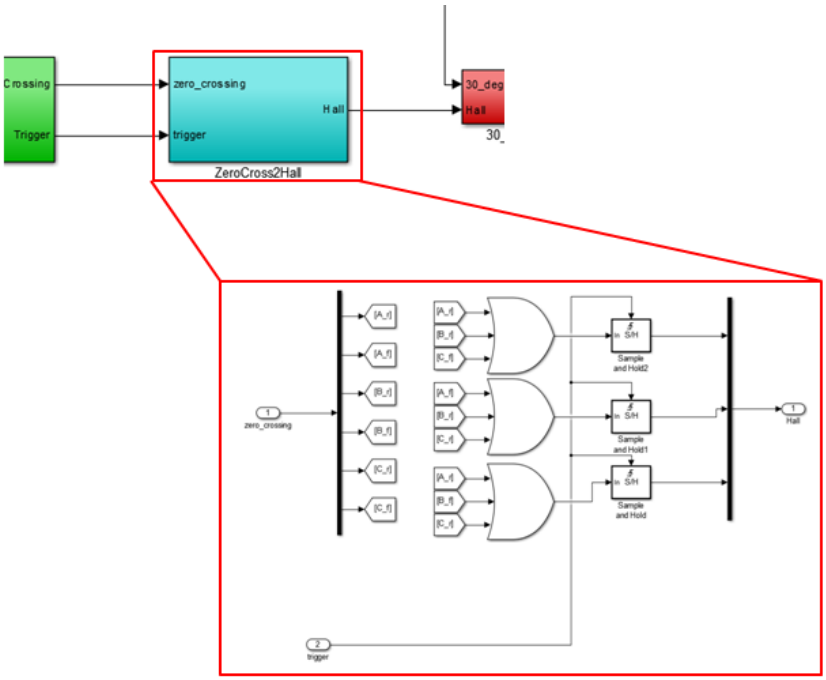

Figura 3.26 – Bloque *ZeroCross2Hall* con su configuración interna

Este bloque transforma el tren de pulsos generados en el bloque *MUX_Select* en señales digitales que se asemejen a las producidas por los sensores Hall. Para ello, combinando las seis entradas y utilizando mantenedores de señal, se consigue obtener tres señales equivalentes a las que producirían los Hall. No obstante, es necesario apuntar que estas señales Hall no han sido generadas utilizando como disparador el cambio de conmutación de las bobinas, sino el paso por cero de la fuerza contraelectromotriz, lo que provoca que estén adelantadas 30º eléctricos.

En la primera gráfica de la figura 3.27 podemos ver como los pulsos que representan la detección de paso por cero suponen un cambio en la señal Hall de alguno de las gráficas segunda, cuarta o sexta, correspondientes a las señales Hall adelantadas 30º eléctricos que estamos detectando con el bloque. Podemos comparar estas señales con las señales de los sensores Hall del motor BLDC, en las gráficas tercera, quinta y séptima.

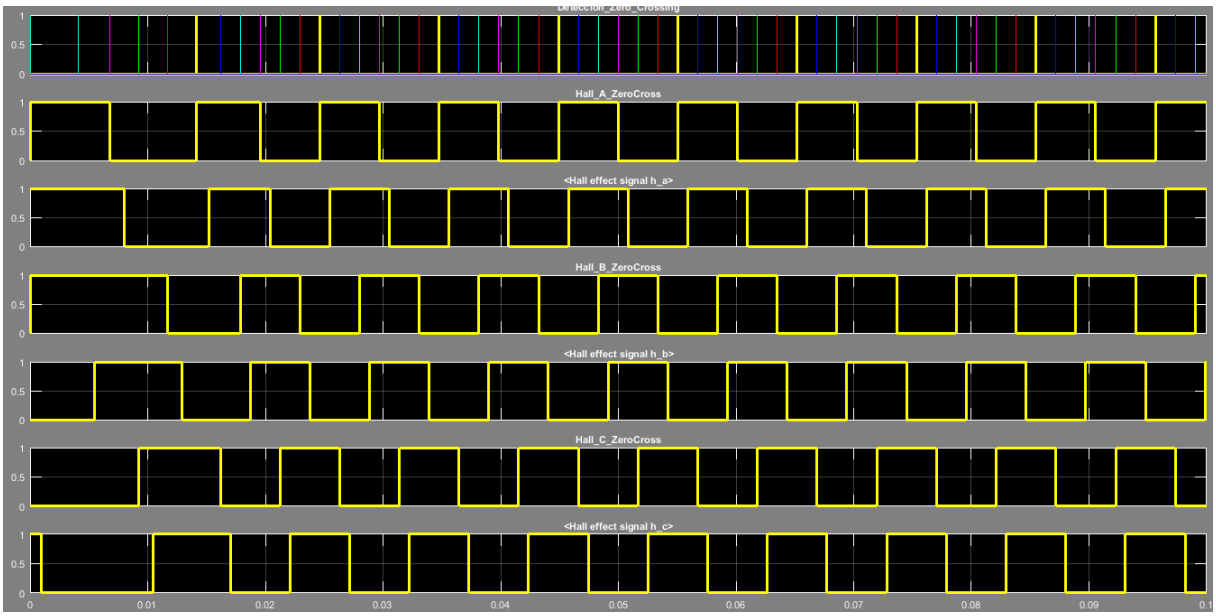

Figura 3.27 – Señales Hall retardadas producidas en el bloque *ZeroCross2Hall*

3.6.5. Bloque *Calculo_30_deg*

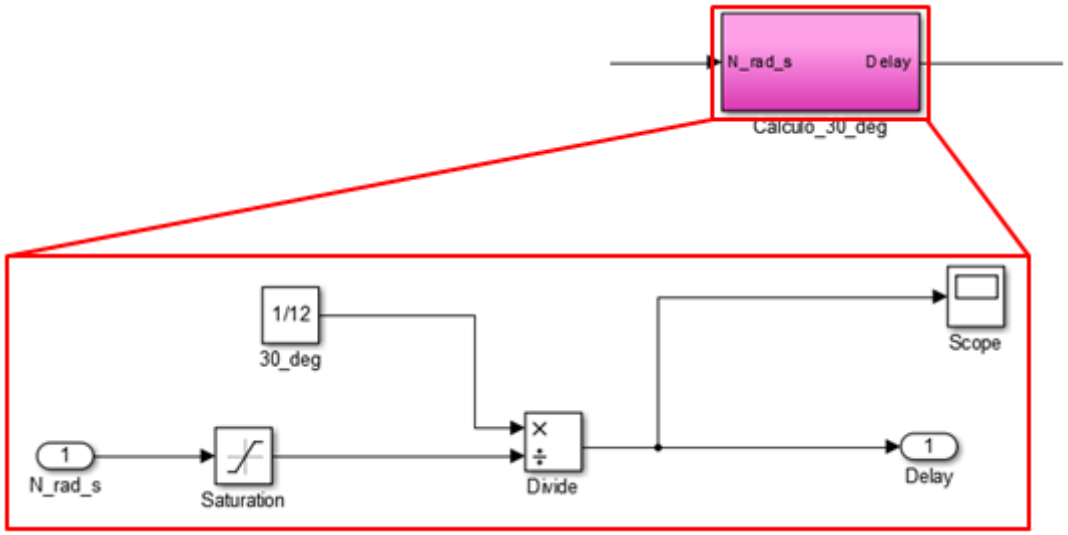

Figura 3.28 – Bloque *Calculo_30_deg* con su configuración interna

En este bloque se calcula el tiempo en segundos que representan 30º eléctricos, partiendo de la velocidad en radianes por segundo. Esta velocidad se ha saturado inferiormente para que nunca baje de 10 rad/s.

Así se consigue limitar la variable temporal *delay*, evitando así que aparezca un retraso excesivo entre la conmutación entre etapas de control consecutivas que produzca un error en el control, o una división por cero que genere un error de cálculo en el controlador en el arranque.

3.6.6. Bloque *30_deg_delay*

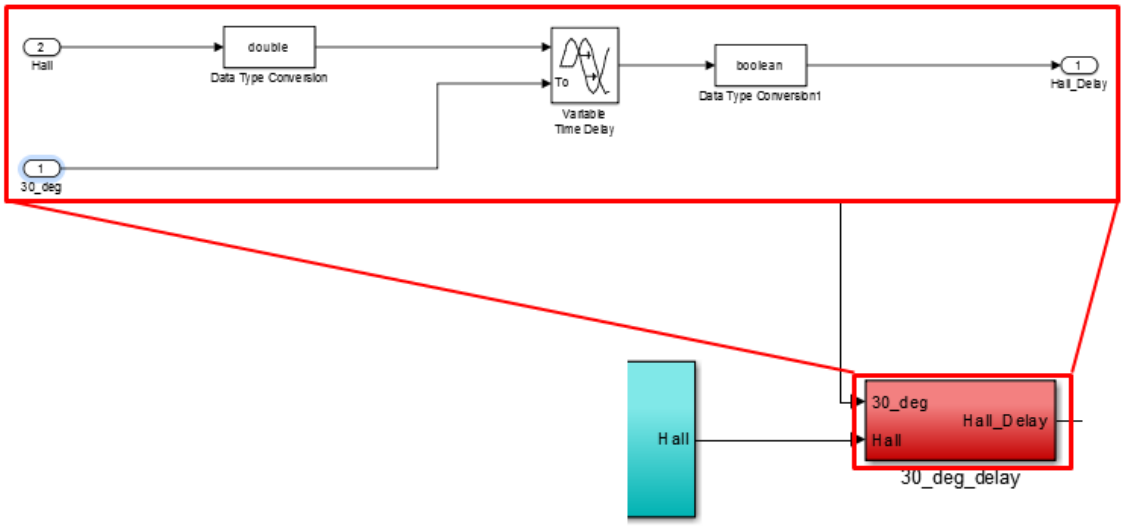

Figura 3.29 – Bloque *30_deg_delay* con su configuración interna

A través de este bloque se consigue retrasar 30º la señal generada en el bloque *ZeroCross2Hall*, para que esta se adecúe perfectamente a la señal que emitirían los sensores Hall del motor, tal y como se muestra en la figura 3.30. Las gráficas primera, tercera y quinta corresponden con las salidas del bloque *30_deg_delay*, mientras que las gráficas segunda, cuarta y sexta corresponden a las entradas del bloque, previas al retraso de 30º eléctricos.

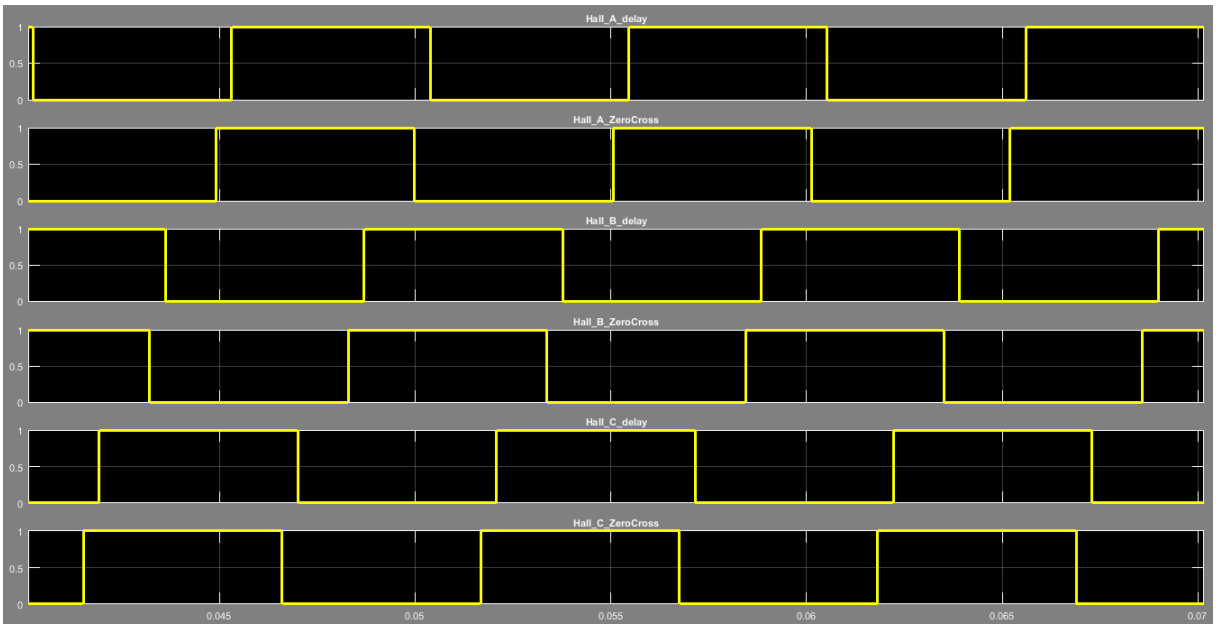

Figura 3.30 – Señales Hall producidas por el bloque *30_deg_delay*

3.6.7. Bloque *Arranque*

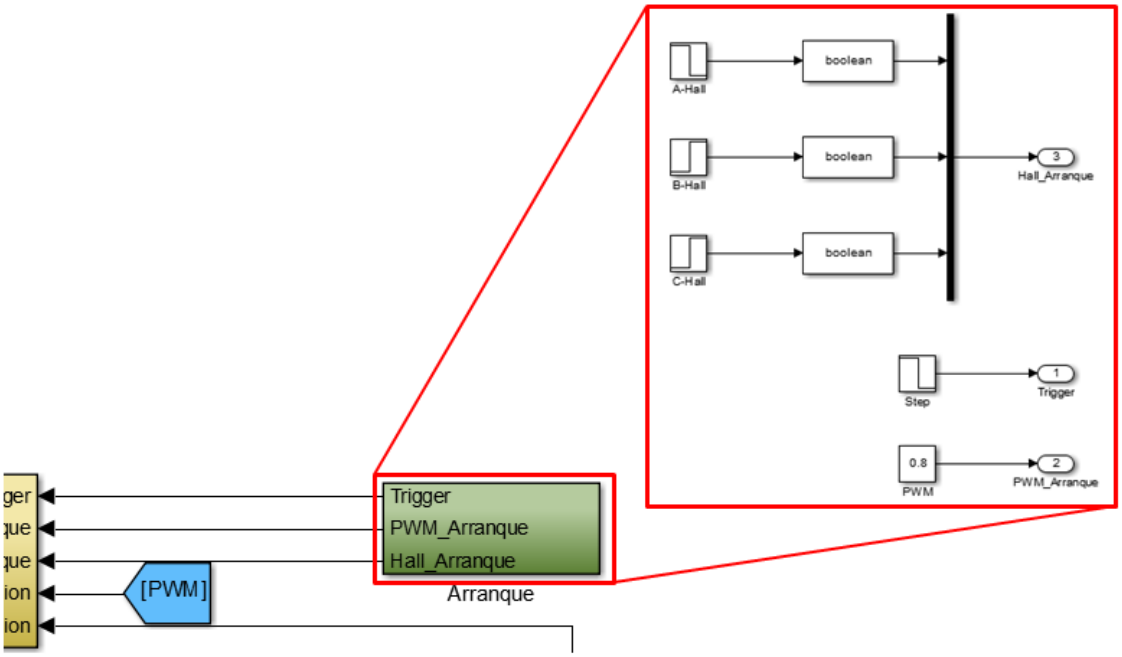

Figura 3.31 – Bloque *Arranque* con su configuración interna

Para el control *sensorless* del motor, se ha tomado la decisión de utilizar un arranque conmutando las bobinas en lazo abierto, durante el periodo en el que, debido a las bajas velocidades, la magnitud de la fuerza contraelectromotriz es menor y la sensibilidad del sistema de detección es menor.

Mediante el bloque *Arranque*, se genera un primer medio ciclo de las señales Hall, así como una señal PWM fija, que se muestra en la figura 3.32.

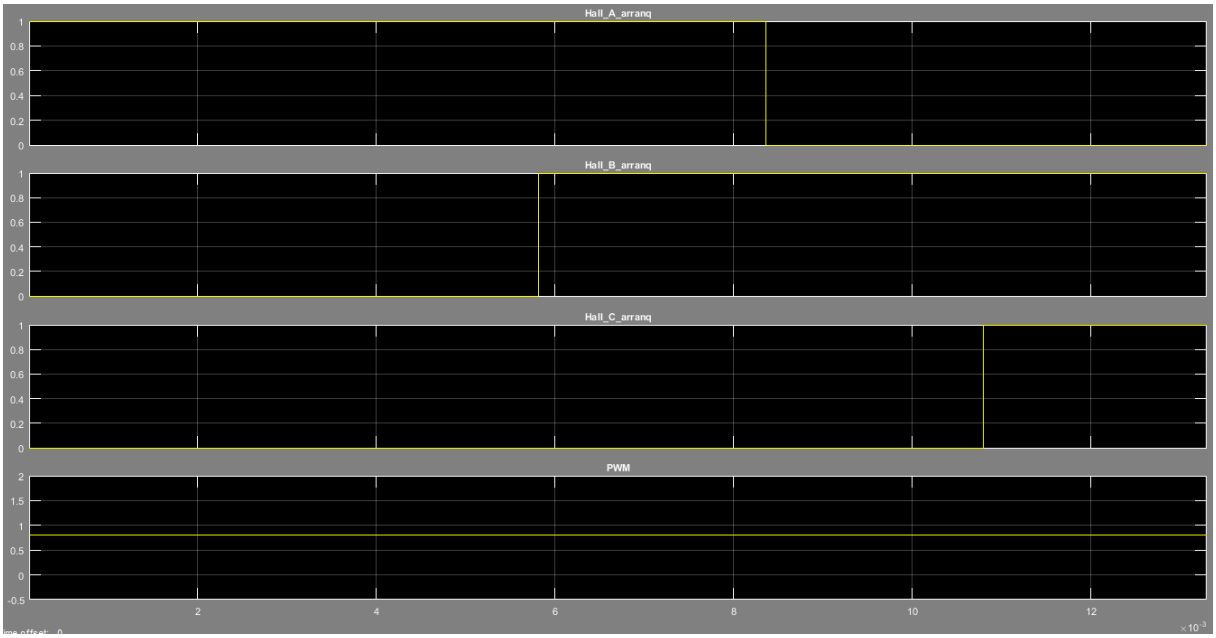

Figura 3.32 – Señales de los sensores Hall y de PWM generados en el bloque *Arranque*

Los tiempos de conmutación se han estimado utilizando el modelo con sensores Hall.

3.6.8. Bloque *Selector Arranque*

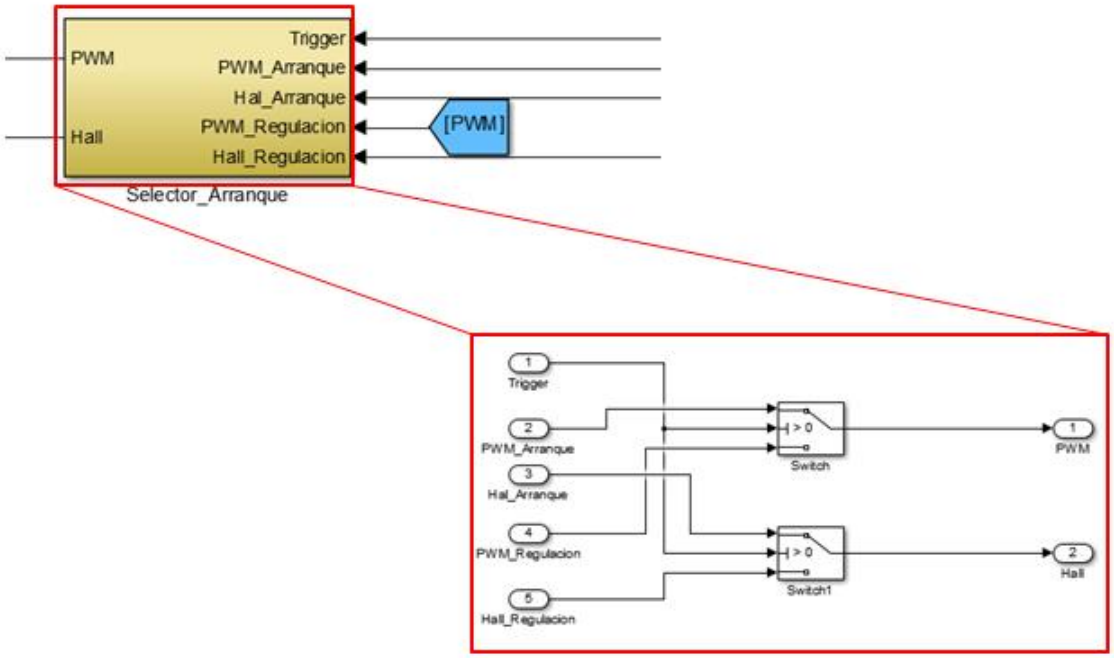

Figura 3.33 – Bloque *Selector_Arranque* con su configuración interna

Mediante el bloque *Selector_Arranque*, se selecciona si el sistema está funcionando en lazo abierto o en lazo cerrado. Se establece una señal de disparo (*Trigger*) que es la responsable de pasar del control en lazo abierto a un control en lazo cerrado.

La salida de este bloque se introducirá en un conmutador manual, que nos permite seleccionar entre funcionamiento basado en Hall o en la fuerza contraelectromotriz en la simulación.

3.7. *Resultados de la simulación*

Los resultados que arroja la simulación son adecuados para ambos tipos de control. Se ha hecho la prueba en la simulación con tres velocidades diferentes, obteniendo resultados similares en el momento que se conecta el control en lazo cerrado para el caso del control sin sensores.

La primera velocidad con la que se ha probado es de 50 rpm, una velocidad baja, para probar la sensibilidad de la detección de la fuerza contraelectromotriz en el caso del control sin sensores, así como un funcionamiento en el que el PWM genere un porcentaje de ciclo bajo. Los resultados se muestran en la figura 3.34.

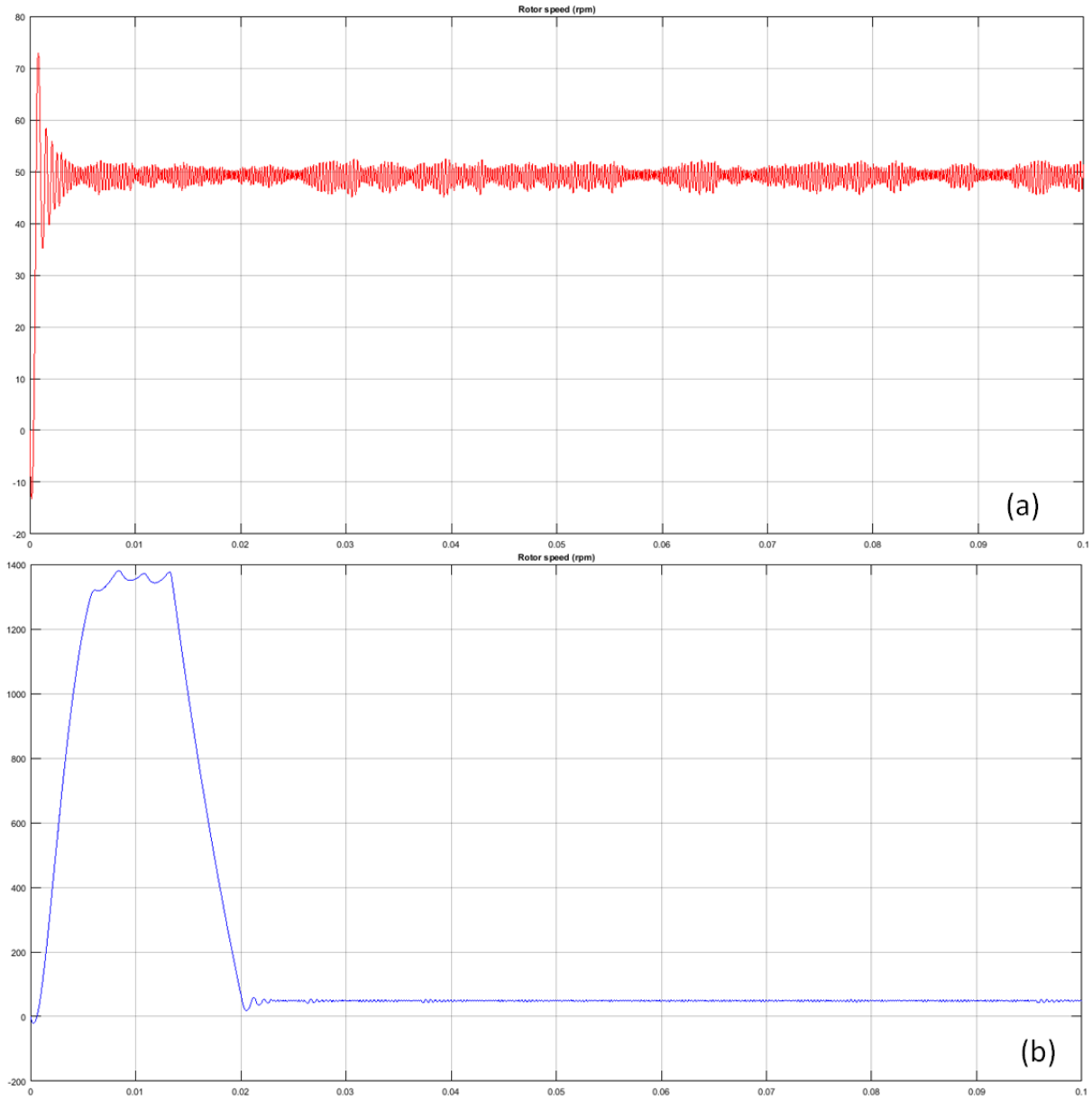

Figura 3.34 – Respuesta del sistema a 50 rpm con sensores Hall (a) o sin sensores (b)

Se ha probado una segunda velocidad, moderada, de 900 rpm, para probar un funcionamiento en el caso de que el PWM genere un porcentaje de ciclo medio. Los resultados aparecen mostrados en la figura 3.35.

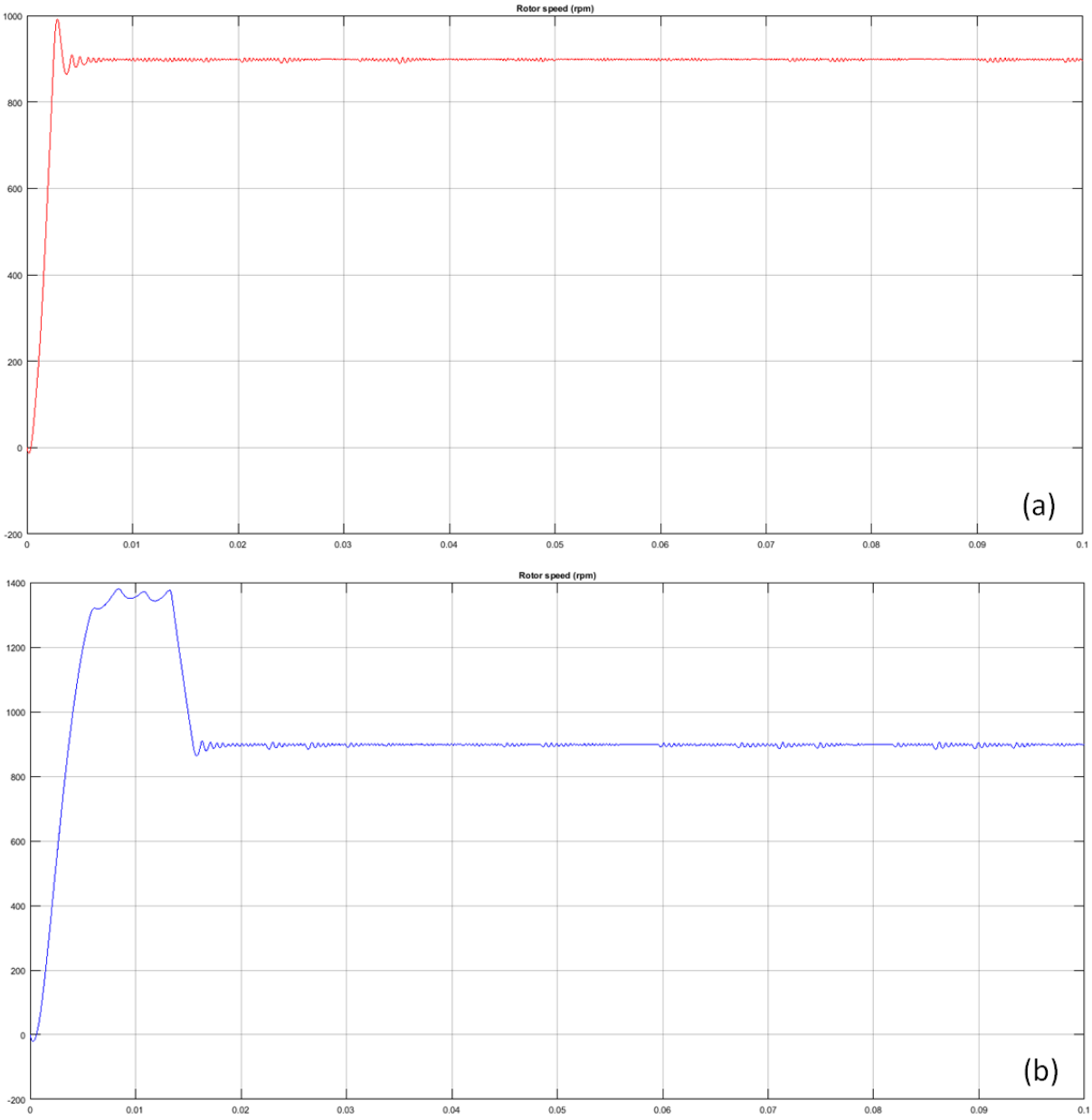

Figura 3.35 – Respuesta del sistema a 900 rpm con sensores Hall (a) o sin sensores (b)

Para la tercera velocidad, se ha probado con una velocidad alta para las características del motor y del par aplicado, de 2200 rpm, caso en el que el PWM generado tendrá un porcentaje de ciclo alto. Para observar los resultados de este caso, se puede consultar la figura 3.36.

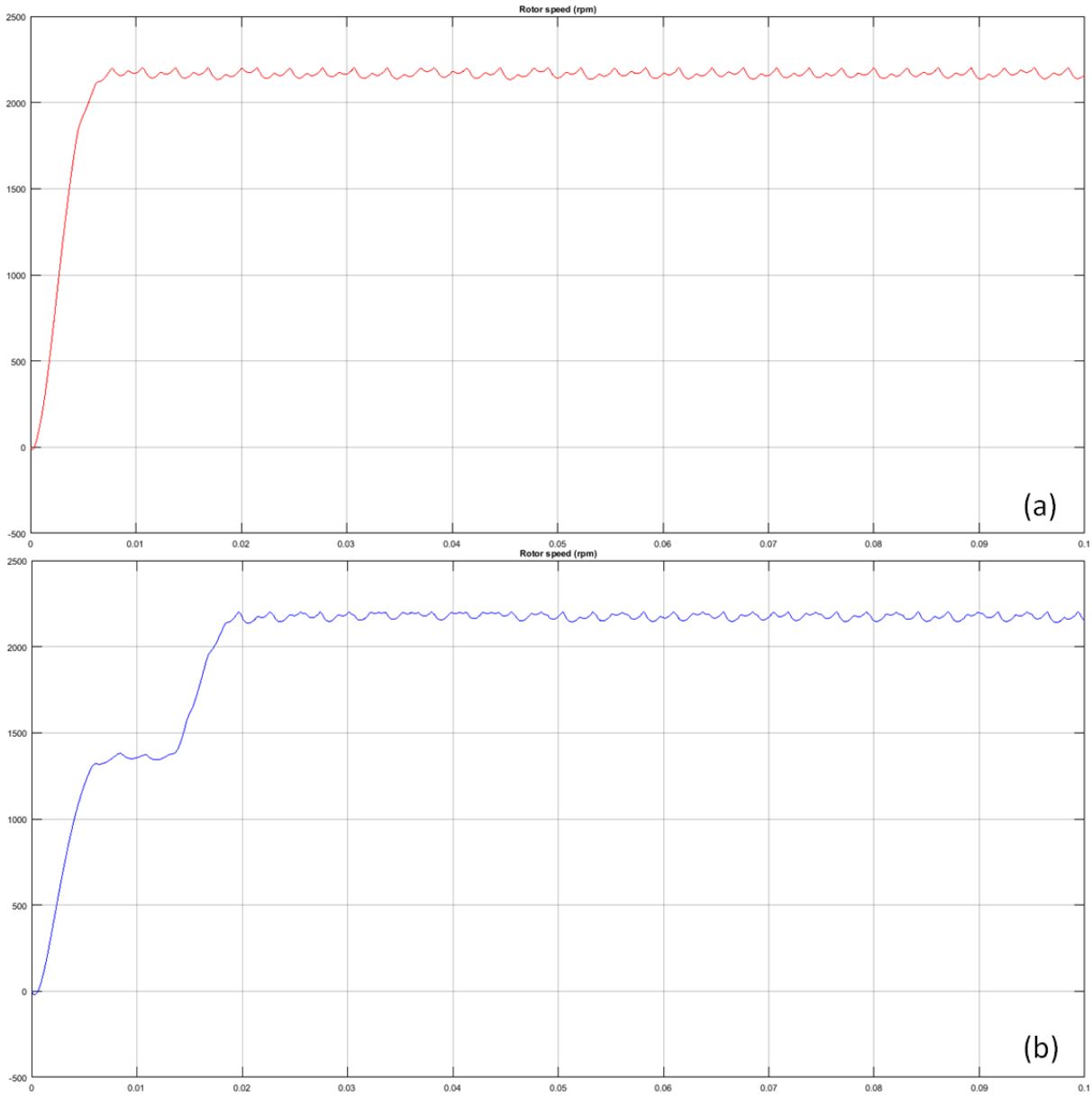

Figura 3.36 – Respuesta del sistema a 2.200 rpm con sensores Hall (a) o sin sensores (b)

Observamos que, una vez pasamos el periodo de funcionamiento en lazo abierto, la respuesta del motor BLDC en lazo cerrado es idéntica para las dos técnicas de control, independientemente de cuál sea la velocidad de funcionamiento.

4. Implementación física

4.1. Introducción

En este capítulo se analizará la implementación física de los modelos desarrollados en simulación durante el capítulo anterior.

4.2. Selección del inversor

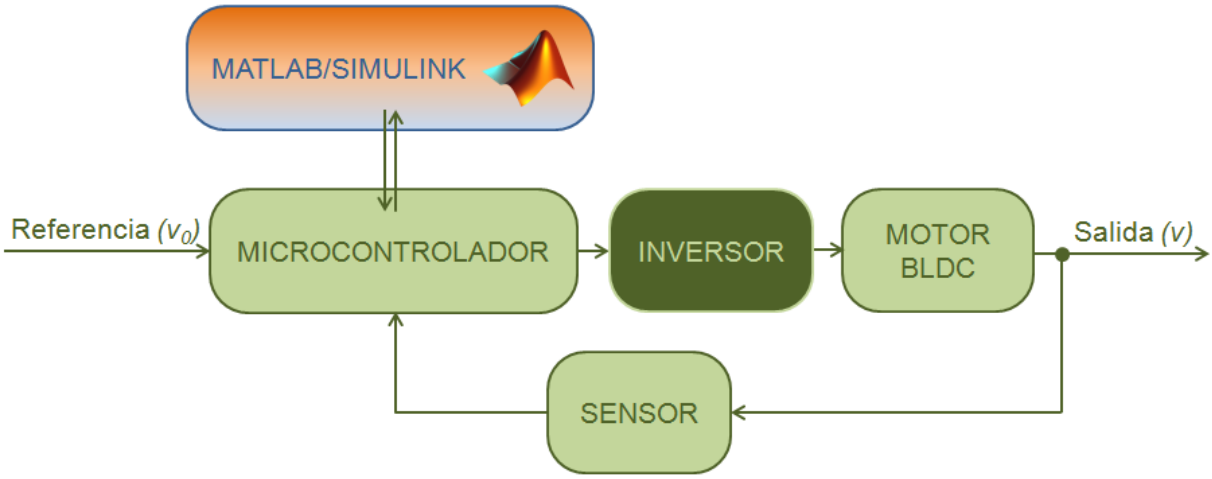

Figura 4.1 – Posición del inversor dentro del sistema

Un inversor es un dispositivo electrónico capaz de transformar una señal digital, corriente continua de baja potencia, en una señal de corriente alterna de cierta potencia, en función de los requerimientos del motor BLDC, habitualmente controlada mediante conmutación en una secuencia predeterminada de semiconductores. La tensión alterna generada a la salida del inversor es controlable en magnitud, frecuencia y fase.

4.2.1. Topologías

Existen numerosos tipos de inversores, específicos según las necesidades de la aplicación, y que se pueden clasificar por:

- Según la necesidad o no de conexión a fuente de corriente alterna:
  - No autónomos o guiados: Son convertidores AC/CC trabajando como inversores. Están controlados por ángulo de fase, de forma que solo se puede controlar la magnitud de la tensión de salida, no la frecuencia. Además, se necesita una fuente de corriente alterna que provoque la conmutación de los polos de potencia.
  - Autónomos o conmutados: No requieren la presencia de una fuente de corriente alterna, porque los polos de potencia se conmutan electrónicamente. En este caso, la señal de salida es controlable en amplitud, frecuencia y fase.
- En función de la fuente de entrada al inversor:

  - Voltage Source Inverter (VSI): Son inversores alimentados por tensión, es decir, que la tensión de entrada permanece constante.
  - Current Source Inverter (CSI): Son inversores alimentados por corriente, es decir, la corriente de entrada permanece constante.
  - Variable DC Link Inverter: Son inversores en los que la tensión de entrada es controlable, mediante un convertidor CC/CC previo.
- En función del número de niveles de salida:
  - Inversor de dos niveles: En la salida (V_a), se generan dos niveles de salida, correspondientes a V_c y 0.
  - Inversor multinivel: En la salida (V_a) se obtiene tensión alterna a partir de más de dos niveles de tensión continua.
- Teniendo en cuenta la topología de la etapa de potencia:
  - Topología en semipuente o medio puente.
  - Topología en puente.
  - Topología push-pull.
- Por el número de fases de salida generadas:
  - Monofásicos
  - Trifásicos
- En función de la técnica de control empleada:
  - Control por pulso único por semiciclo
  - Control por pulso múltiple por semiciclo:
    - Modulación SVM
    - Modulación Sliding
    - Modulación PWM
    - Control predictivo

En el presente proyecto se ha contemplado el uso de dos inversores diferentes, cuya conveniencia se determinará por criterios de costes y ergonomía, y que a partir de estas clasificaciones, podrán comenzar a ser identificados.

4.2.2. Inversor DM164130-2

En el primer caso, nos encontramos con una placa inversora de Microchip Technology capaz de manejar la gran mayoría de tipos de motores BLDC trifásicos usados en electrónica de consumo. Externamente, el dispositivo tiene el aspecto de la figura 4.2.

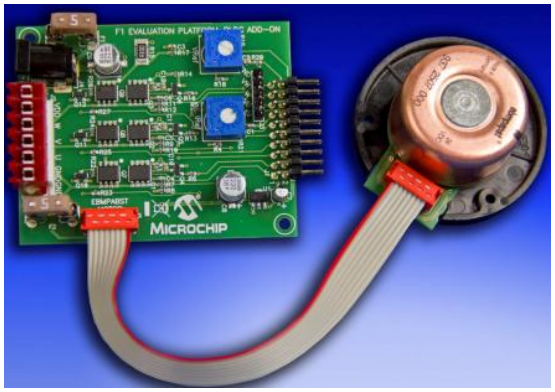

Figura 4.2 – Inversor DM164130-2

Atendiendo al datasheet del dispositivo, podemos observar un diagrama en el que se muestra un ejemplo de aplicación del inversor en el que se conecta a un motor BLDC y a un controlador PIC.

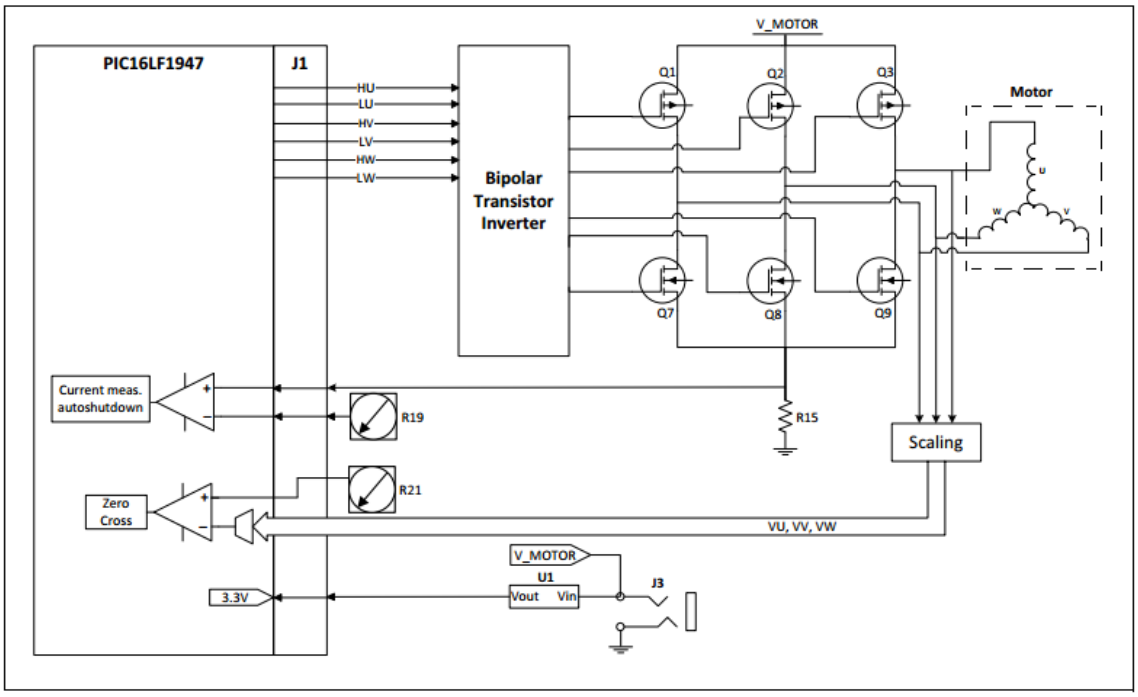

Figura 4.3 – Ejemplo de aplicación del inversor DM164130-2

Observamos que el inversor tiene una configuración en puente trifásico, formado por tres transistores MOSFET de tipo P y otros tres de tipo N, comandados mediante seis transistores bipolares. La etapa de potencia tiene seis entradas de tipo PWM (HU, LU, HV, LV, HW, LW), y tres entradas que miden la fuerza contraelectromotriz generada por las fases del motor (VU, VV, VW), que se utilizan para determinar la posición angular del rotor sin el empleo de sensores externos gracias a la comparación con la señal V_REF.

Además, incorpora una medición de la corriente global que circula por el motor, mediante la medición de la tensión que cae en la resistencia R15, y también dos fusibles de 5A (F1, F2), que se utilizan para proteger la placa en caso de la aparición de una corriente de cortocircuito. En la figura 4.4 podemos ver la estructura interna completa del inversor.

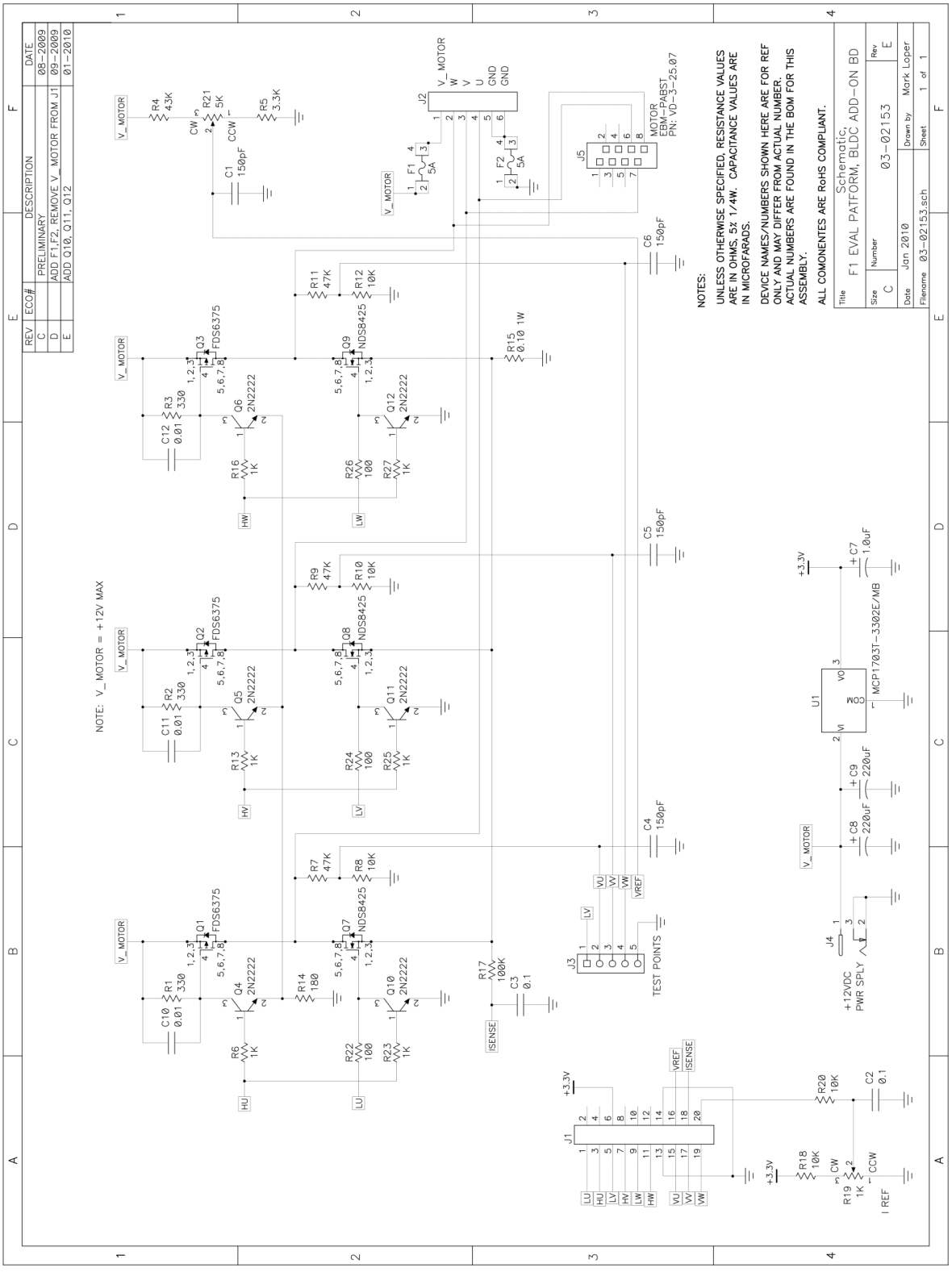

Figura 4.4 – Estructura interna del inversor DM164130-2

Para analizar más en detalle el funcionamiento del inversor, así como para obtener una tabla lógica que describa completamente el comportamiento de éste, utilizaremos el esquema de una de las fases para analizar la respuesta ante distintas entradas.

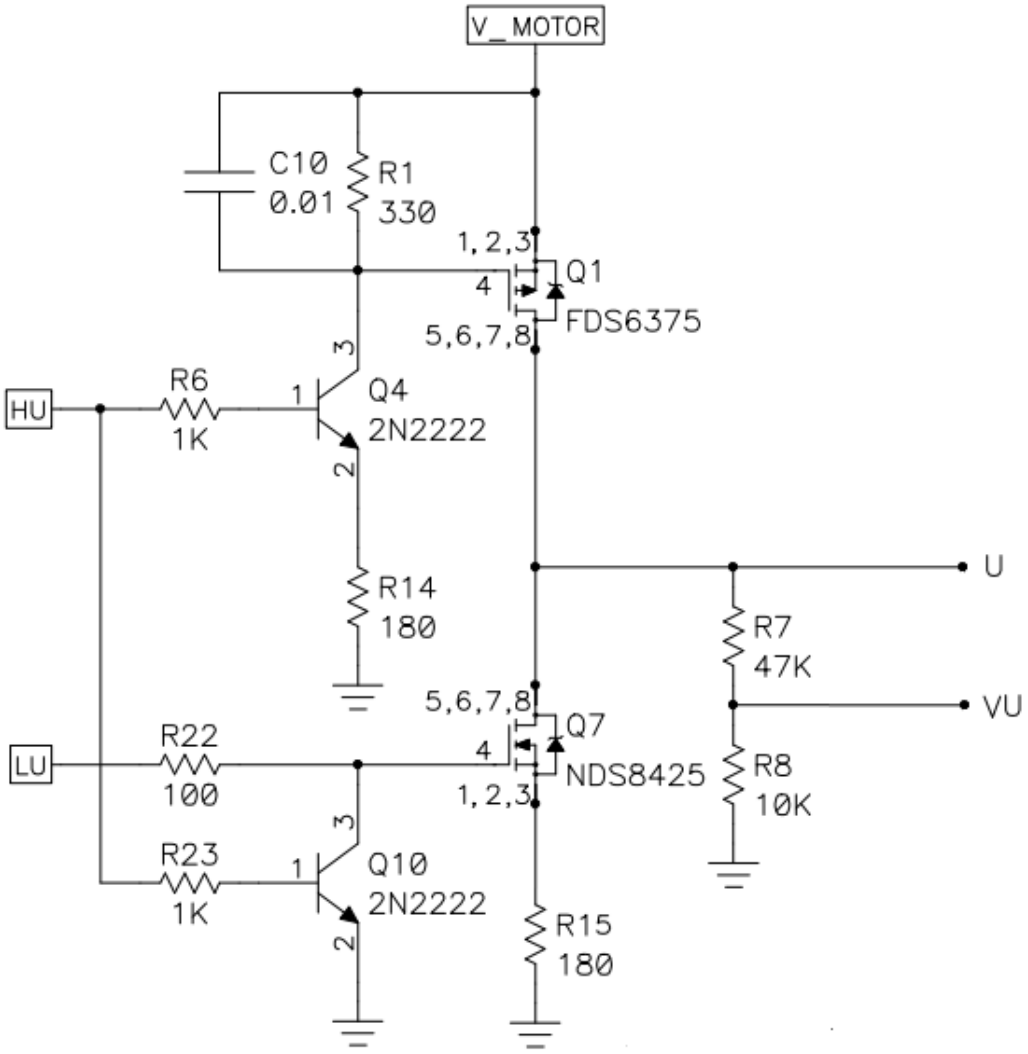

Figura 4.5 – Esquema detalle de una de las fases del inversor DM164130-2

Lo primero que se observa a simple vista es que para cada una de las fases nos encontramos con dos entradas, HX y LX, y dos salidas, X y VX, donde X es la denominación de cada una de las fases (U, V, W). El caso de la fase a analizar es la fase U.

Construiremos la tabla lógica aplicando todas las combinaciones posibles de entradas, y revisando lo que ocurre en las salidas, teniendo en cuenta que los transistores bipolares tipo NPN se activarán cuando se ponga un '1' en la base, mientras que los MOSFET tipo N se activan al aplicar una tensión positiva en la compuerta, y los MOSFET tipo P se activan al aplicar una tensión negativa en la compuerta.

- Caso 1: LU=0 y HU=0

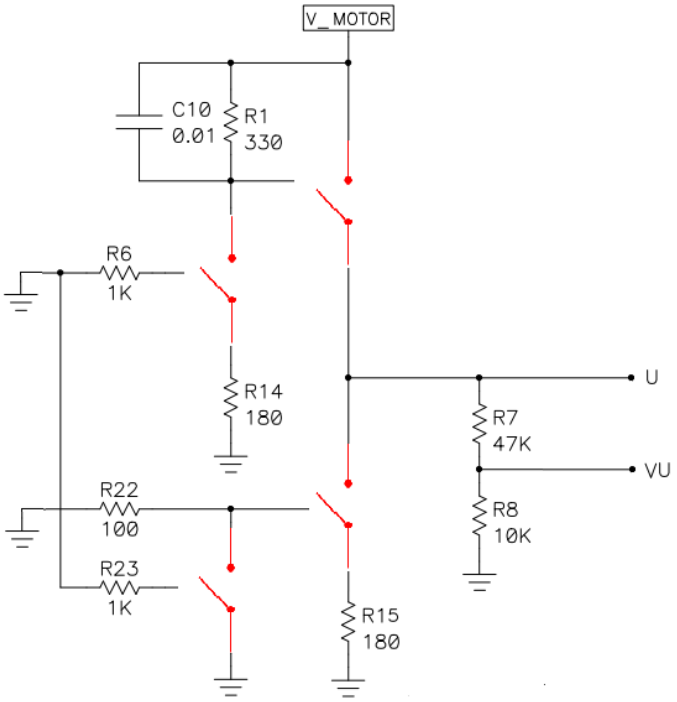

Figura 4.6 – Caso LU=0 y HU=0

Con las dos entradas con valor '0', los cuatro transistores actúan como interruptores abiertos, de forma que la salida U queda aislada y unida a la tensión de referencia, y lo mismo se puede decir de VU, por lo que podemos decir que las salidas serán U=0 y VU=0.

- Caso 2: LU=1 y HU=0

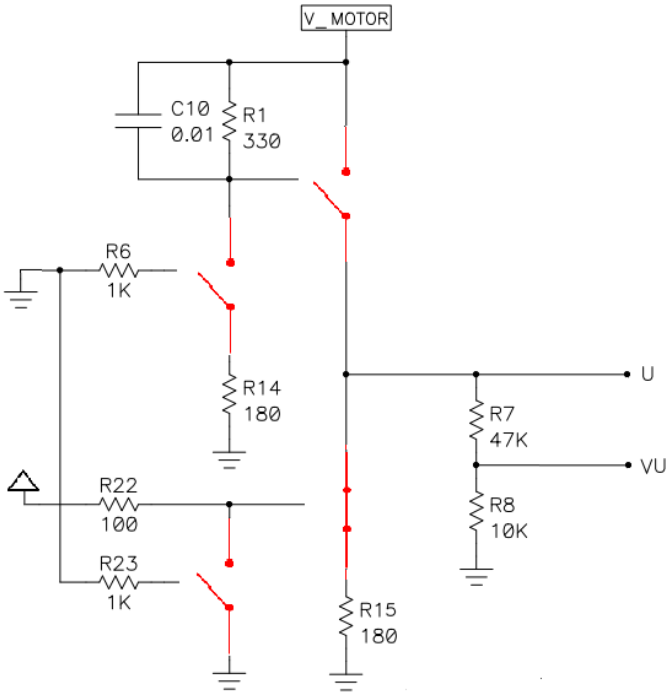

Figura 4.7 – Caso LU=1 y HU=0

Dado que la entrada HU es la que regula el estado de los transistores bipolares, estos continuarán en estado abierto. Al aplicar una tensión positiva sobre la compuerta del MOSFET de tipo N, este actúa como un interruptor cerrado, por lo que podemos observar que las salidas en este caso se mantendrán en estado de U=0, VU=0

- Caso 3: LU=0 y HU=1

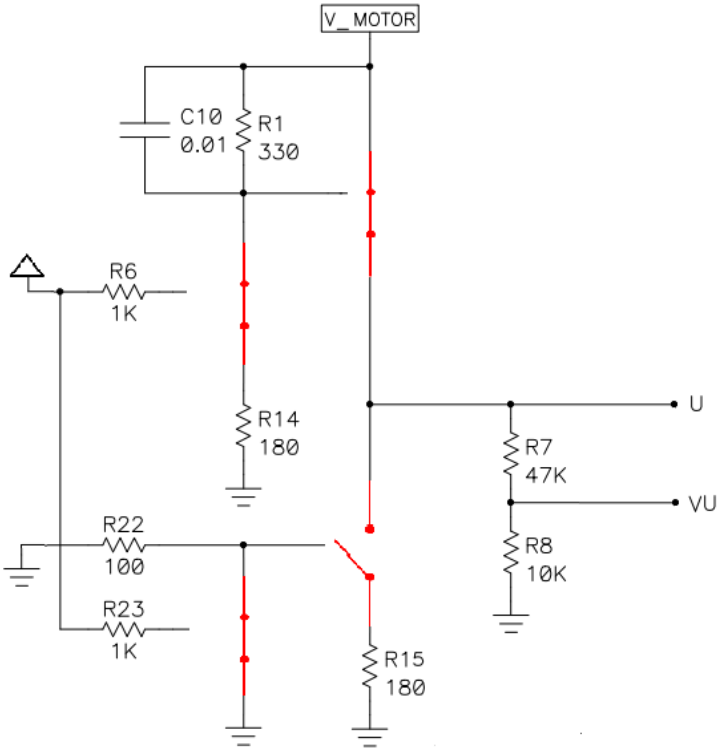

Figura 4.8 – Caso LU=0 y HU=1

Al introducir una tensión positiva en la entrada HU, activamos el estado de los transistores bipolares, comportándose éstos como interruptores cerrados. Esto provoca que apliquemos una tensión nula sobre la compuerta del MOSFET de tipo N, actuando éste como un interruptor abierto, mientras que sobre el MOSFET de tipo P actuará una tensión positiva, lo que provocará que se comporte como un interruptor cerrado que transmita la tensión V_{MOTOR} en U.

- Caso 4: LU=1 y HU=1

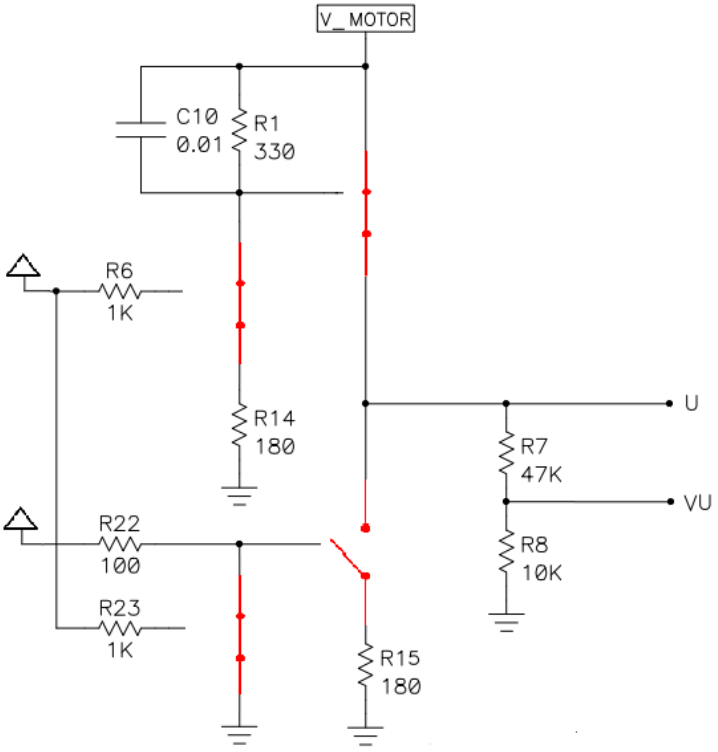

Figura 4.9 – Caso LU=1 y HU=1

En el último de los casos, en el que se introduce una tensión positiva por ambas entradas, el estado de los transistores es el mismo que en el caso anterior. Esto sucede porque al introducir un '1' lógico en la entrada HU, se activa el transistor bipolar Q10, que hace que en la puerta del MOSFET de tipo N Q7 se coloque un '0', haciendo que éste no conduzca, además de activar el transistor de tipo P Q1.

A partir del estudio de la fase U, hemos obtenido una tabla lógica, que es aplicable a las tres fases del motor:

HU	LU	U	VU
0	0	0	0
0	1	0	0
1	0	V_{MOTOR}	V_{MOTOR}·10/(47+10)
1	1	V_{MOTOR}	V_{MOTOR}·10/(47+10)

Figura 4.10 – Tabla lógica del inversor DM164130-2

Estudiando esta tabla, se procede a realizar la clasificación del inversor en función de los criterios previamente definidos:

- El inversor es autónomo, dado que trabaja sin necesidad de ninguna fuente de corriente alterna, sino que se controlan mediante conmutación de transistores.

- El inversor está alimentado por tensión (Voltage Source Inverter), ya que la tensión V_{MOTOR} proviene de una batería o fuente de alimentación cuya tensión se mantiene aproximadamente constante.
- Es un inversor de dos niveles, ya que en la salida se puede obtener la tensión V_{MOTOR} o GND.
- Es un inversor en semipuente o medio puente, controlado mediante transistores MOSFET.
- Es un inversor trifásico.
- A efectos del presente proyecto, la técnica de control que se va a emplear es control por pulso múltiple por semiciclo, con modulación por ancho de pulso (PWM).

4.2.3. Inversor L6234

El L6234 es un inversor de STMicroelectronics, cuyo aspecto externo es el siguiente, correspondiente a un encapsulado DIP20.

Figura 4.11 – Inversor L6234

A la vista de su datasheet, podemos obtener su mapa de pines, cuyo análisis será necesario a la hora de implementar el control del motor.

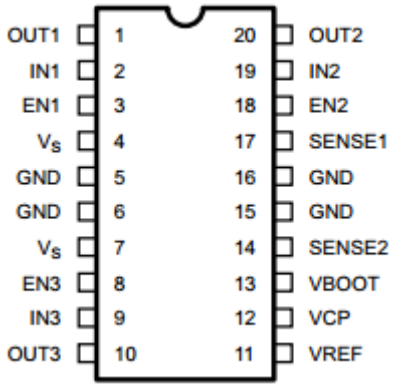

Figura 4.12 – Mapa de pines del inversor L6234

La función de cada uno de los pines se muestra en la siguiente tabla:

Pin	Nombre	Función
1 20 10	OUT 1 OUT 2 OUT 3	Salida de los canales 1, 2 y 3
2 19 9	IN 1 IN 2 IN 3	Entrada lógica a los canales 1,2 y 3. Un nivel lógico ALTO en un canal activa el transistor superior, mientras que un nivel BAJO activa el transistor inferior (siempre que el pin EN correspondiente esté en ALTO)
3 18 8	EN 1 EN 2 EN 3	Señal de activación de los canales 1, 2 y 3. Un nivel lógico BAJO en este pin desactiva ambos transistores
4, 7	V_S	Tensión de alimentación del motor
14	SENSE2	Un sensor resistivo que se conecte a este pin da una medida de la corriente que circula por el puente 3
17	SENSE1	Un sensor resistivo que se conecte a este pin da una medida de la corriente que circula por los puentes 1 y 2
11	VREF	Referencia interna de tensión. Un condensador conectada entre este pin y GND incrementa la estabilidad del circuito
12	VCP	Salida de oscilador
13	VBOOT	Entrada de sobretensión para manejar el transistor superior
5,6 15,16	GND	Señal de tierra común. Además, en los encapsulados también se usan para disipar el calor del circuito integrado hacia la PCB

Figura 4.13 – Tabla de pines del inversor L6234

Del datasheet podemos obtener también el esquema de la estructura interna del inversor, que podemos dividir en tres etapas, desde las entradas a las salidas, puertas AND, drivers y transistores MOSFET de canal N.

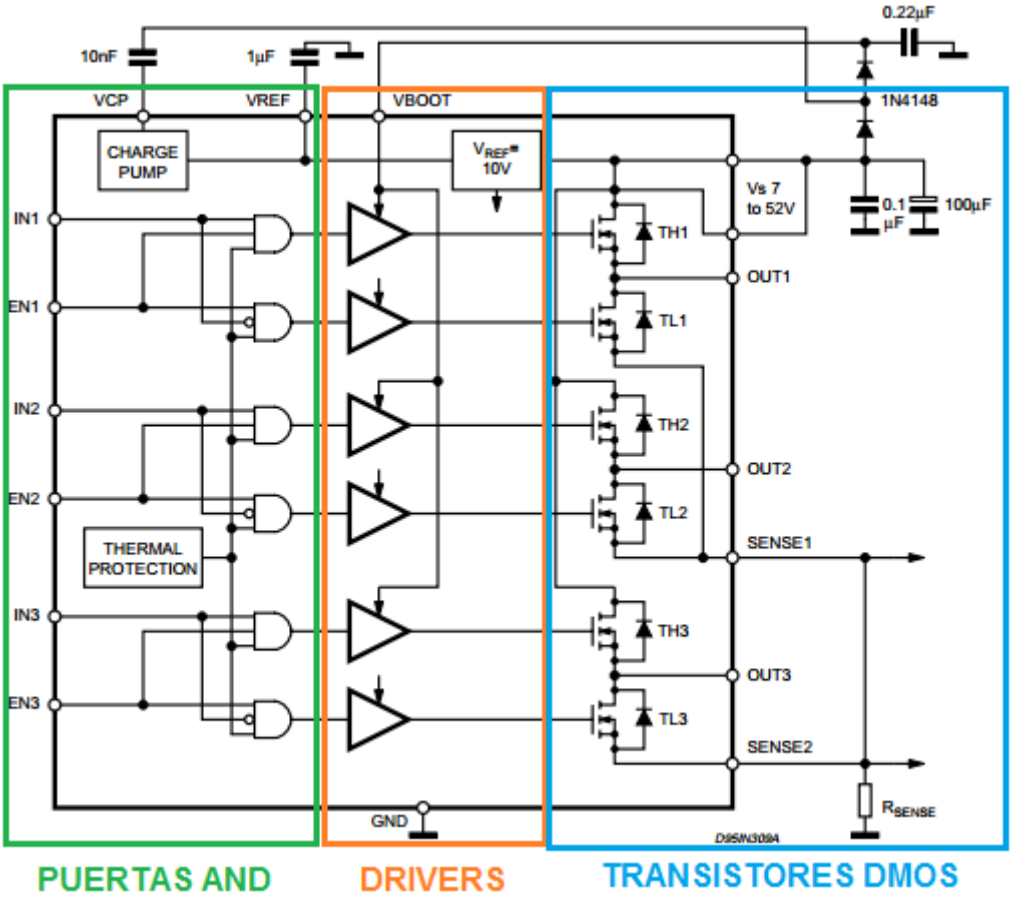

Figura 4.14 – Estructura interna del inversor L6234

- Puertas AND: Sirven para discernir, en cada fase, si se debe activar el transistor superior, el inferior o ninguno de los dos. Hay un total de 6 puertas, cuyas entradas son una combinación de las entradas IN, las entradas EN y la protección térmica, de forma que por encima de niveles térmicos que supongan un riesgo para el dispositivo el suministro de corriente al motor se interrumpe y éste se para. Las salidas de cada una de las puertas se conducen directamente a los drivers que disparan las puertas.
- Drivers: Se utilizan para disparar el transistor correspondiente a la puerta AND que lo ha activado. Se utiliza la técnica del bootstrap para que no sea necesaria la presencia de alimentaciones independientes para cada uno de los transistores superiores, así como otra más para los inferiores.
- Transistores MOSFET: Cada uno de los transistores tiene un diodo de protección, para evitar polarizaciones en inversa. Los transistores inferiores se activan cuando en su entrada IN correspondiente se coloca un '0', mientras que los superiores se activarán cuando se coloque un '1'.

Para cada una de las fases, su tabla lógica será la reflejada en la figura 4.15.

THERMAL_PROTECT	EN	IN	OUT
0	X	X	0
1	0	X	0
1	1	0	0
1	1	1	1

Figura 4.15 – Tabla lógica del inversor L6234

4.2.4. Criterios de selección

Para seleccionar entre uno y otro, se han tenido en cuenta una serie de criterios y requerimientos técnicos y económicos del proyecto para realizar una valoración.

- Simplicidad: En este apartado, el inversor de Microchip tiene ventaja, al no necesitar de diseñar una placa de circuito impreso para utilizarla en el proyecto.
- Ergonomía: En el apartado de ergonomía, entendida en este caso como tamaño y facilidad de integración, el inversor de Microchip presenta la ventaja de que ya ha atravesado todo el proceso de optimización del espacio durante su diseño, mientras que para el caso del inversor de STMicroelectronics, habría que realizar el proceso de optimización del espacio en el desarrollo de la placa de circuito impreso.
- Robustez: El inversor de Microchip, al comercializarse ya integrado en una placa de circuito impreso, presenta una menor probabilidad de fallo que una placa que se diseñe específicamente para el inversor de ST.
- Tiempo de desarrollo: Evidentemente, la utilización del inversor de Microchip conllevará un tiempo de desarrollo mucho menor que en el caso del ST, al estar ya realizado el proceso de diseño de la PCB.
- Personalización: En el apartado de personalización, el hecho de utilizar una tarjeta prediseñada como la de Microchip nos resta capacidad de personalizar el diseño y la colocación de los elementos del sistema, mientras que para el caso del inversor ST las posibilidades son mucho mayores.
- Coste económico: En lo que se refiere a coste económico, el inversor de ST tiene ventaja, ya que tiene un coste de alrededor de 7 €, frente a los 90€ que cuesta el de Microchip (nótese que por ese precio incluye también un motor BLDC y un controlador PIC). No obstante, al tratarse este de un proyecto de investigación en el que se va a desarrollar un prototipo, no consideraremos como un factor importante el coste económico.

A la vista del análisis de factores realizado, el inversor que se va a elegir para el proyecto es el Microchip DM164130-2.

4.3. Selección del microcontrolador

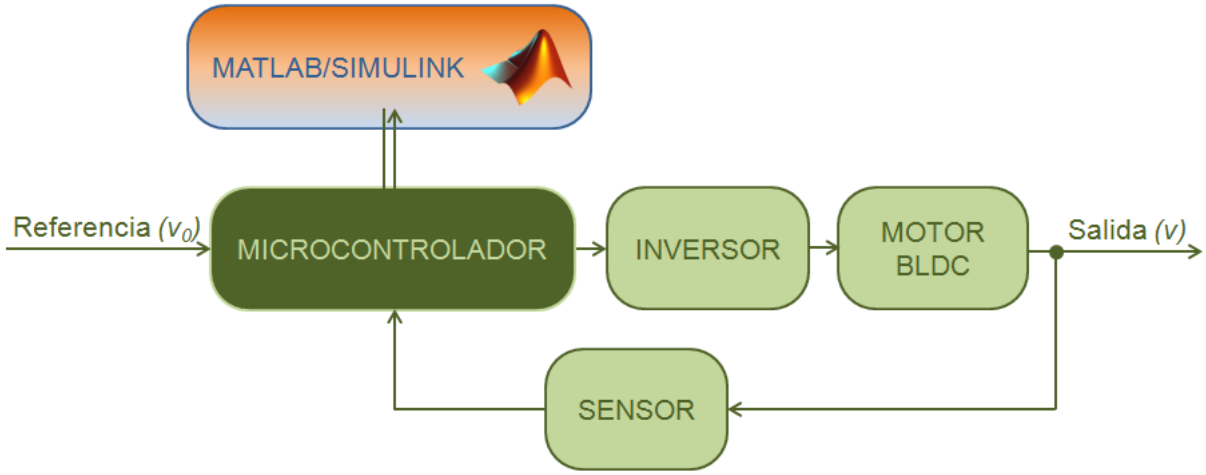

Figura 4.16 – Posición del microcontrolador dentro del sistema

Un microcontrolador es un circuito integrado programable, que permite ejecutar una serie de instrucciones almacenadas en una memoria. Un microcontrolador suele tener una arquitectura modular, con varios bloques funcionales, los cuales cumplen una tarea específica. Dentro de estos bloques, existen tres fundamentales: la unidad central de procesamiento, la memoria y los periféricos de entrada/salida.

En la aplicación al control de velocidad de un motor BLDC como la del presente proyecto, el cometido del microcontrolador es, en primer lugar, detectar la posición del rotor, ya sea mediante los sensores Hall o el paso por cero de la fuerza contraelectromotriz, así como la velocidad de giro del motor, bien recibida mediante un encoder o calculada utilizando el tiempo empleado en cada etapa de control; en segundo lugar, realizar los cálculos necesarios para determinar la etapa de control en la que nos encontramos en cada instante y aplicar el algoritmo PID a la señal de error de velocidad leída, y en tercer lugar, generar las señales de salida necesarias para controlar el inversor.

Dentro de las numerosas posibilidades para la selección del microcontrolador, durante el presente proyecto se han evaluado tres alternativas.

4.3.1. Microcontrolador Arduino

Arduino, atendiendo a la descripción que ofrece su web oficial, es "*una plataforma de prototipos electrónica de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar*". Arduino es una herramienta sencilla, flexible y económica para implementar programas digitales en sistemas reales, utilizando una amplia gama de sensores y actuadores.

La plataforma Arduino tiene su origen en el año 2005, en el instituto Interactivo de Ivrea (Italia), en el que los estudiantes utilizaban el microcontrolador BASIC Stamp, con un coste de unos 100 dólares estadounidenses, demasiado alto para los estudiantes. De la necesidad de una plataforma de bajo coste y sin problemas de compatibilidades entre sistemas operativos, surge el proyecto Arduino, cuyo nombre deriva del rey Arduino de Italia, entre los años 1002 y 1014. El instituto fue cerrado en el año 2006, habiendo sido

previamente liberada la patente del proyecto, para evitar que éste fuera embargado, y permitiendo así que el proyecto no quedara en el olvido, y desarrolladores de todo el mundo tuvieran la oportunidad de evolucionar el proyecto.

Dentro de cualquier dispositivo Arduino, existen tres componentes fundamentales y diferenciados:

- Hardware
- Software
- Lenguaje de programación

4.3.1.1. Hardware

Un dispositivo Arduino consiste en una placa de circuito impreso en la que se integran diversos elementos, siendo los más importantes a efectos de la aplicación del presente proyecto el microcontrolador, los puertos de entrada y salida y la interfaz con el ordenador.

En lo que se refiere al microcontrolador, Arduino, desde su origen, ha empleado microcontroladores Atmel AVR, siendo los más utilizados el Atmega168, Atmega328, Atmega1280 y Atmega8; aunque desde octubre de 2012 Arduino ha comenzado también a emplear microcontroladores más potentes como los Cortex M3 de ARM, de 32 bits, para aplicaciones más complejas.

En el caso del presente trabajo, la placa de Arduino seleccionada es un Arduino Uno, que cuenta con un microcontrolador Atmega 328P de 8 bits, cuyas características técnicas más importantes se reflejan en la figura 4.17.

PARÁMETRO	VALOR	UNIDAD
Tipo de CPU	8	Bits
Memoria Flash	32	kbytes
Memoria SRAM	2	kbytes
Frecuencia máxima de operación	16	MHz

Figura 4.17 – Resumen características técnicas Arduino Uno

Acerca de los puertos de entrada y salida de la placa Arduino Uno, podemos decir que existen un total de 28 pines hembra, que podemos clasificar en tres grandes grupos:

- POWER: Son pines que proporcionan salidas de tensiones continuas de distintos niveles.

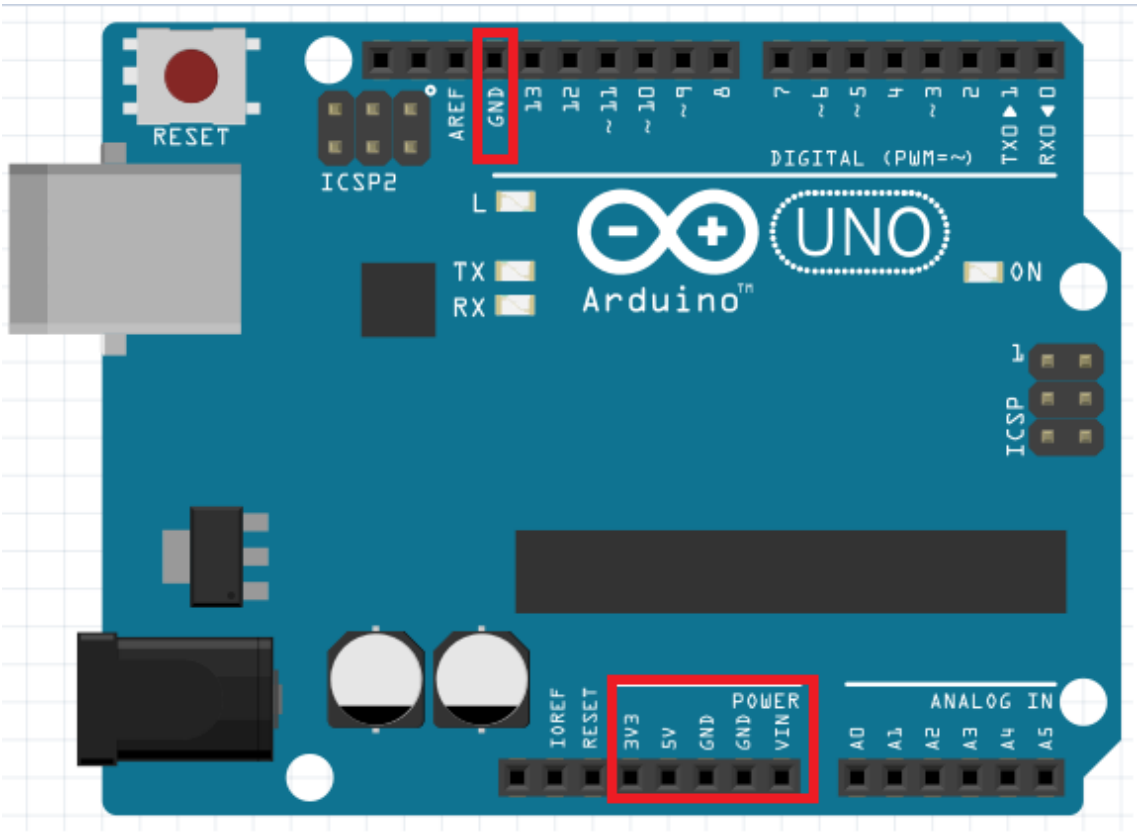

Figura 4.18 – Pines POWER del Arduino Uno

Los distintos pines son:

- Vin: Proporciona una tensión igual a la que se alimenta la placa desde una fuente externa.
- GND: Proporciona el nivel de tensión de referencia (0V) para la placa.
- 3.3V: Gracias a un regulador de tensión integrado en la placa, es capaz de suministrar una tensión constante de 3,3 V con una corriente máxima de 150 mA.
- 5V: Mediante otro regulador, suministra una tensión constante de 5 V, capaz de suministrar hasta 1 A (límite máximo del regulador). No obstante, esta corriente provoca un calentamiento que supone que la corriente máxima esté también limitada térmicamente, disminuyendo ésta cuando la temperatura crece por encima de niveles peligrosos, de forma que el datasheet asigna una máxima corriente admisible de 200 mA.

- DIGITAL: Denominados también GPIO (General Purpose Input/Output), pueden ser utilizados indistintamente como entradas y salidas digitales, con una tensión de 5 V y con una corriente que en ningún caso debe superar los 40 mA.

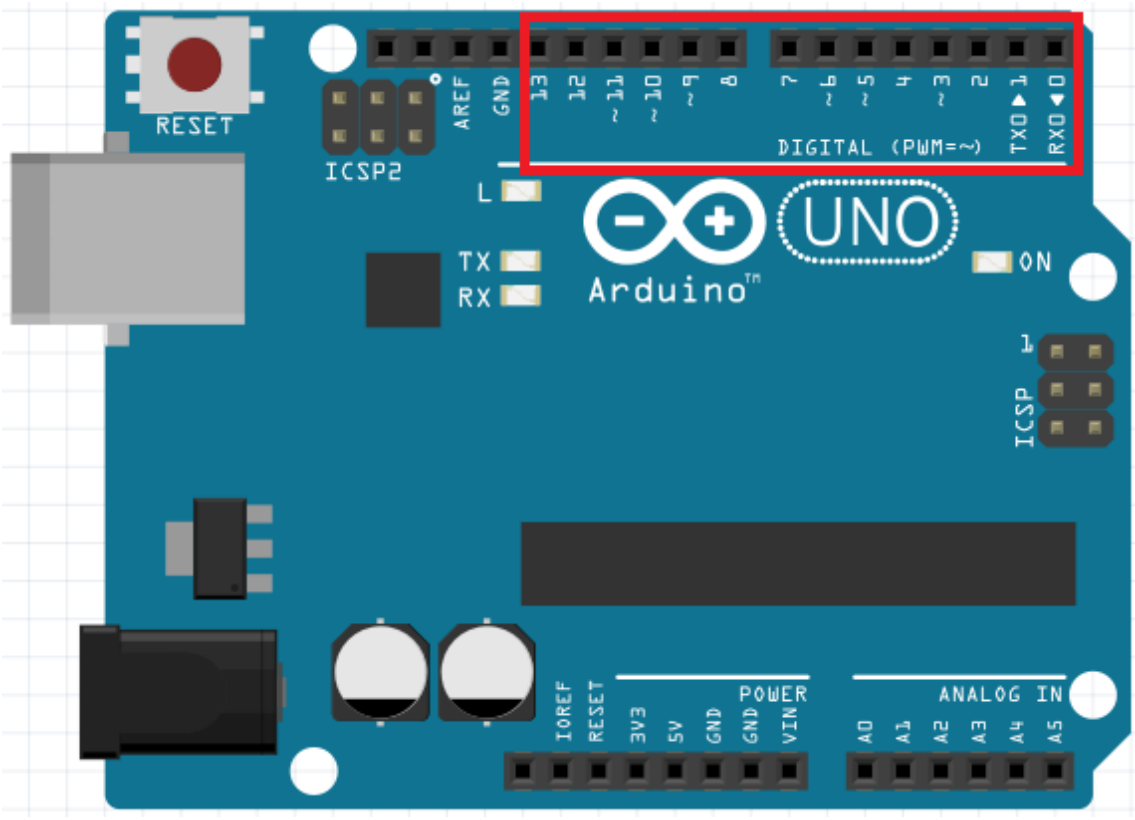

Figura 4.19 – Pines digitales del Arduino Uno

Además de sus funciones como entradas y salidas digitales de propósito general, existen algunos con peculiaridades sobre las que conviene llamar la atención:

- Pin 0 y 1: Son pines utilizados para comunicación serial, siendo el pin "0" el RX (recibe datos) y el pin "1" el TX (transmite datos). Se utilizan a la hora de conectar la placa a diferentes dispositivos de comunicación como Bluetooth, Bluetooth LE, ZigBee o Ethernet, así como para comunicarse con el ordenador a través del puerto USB.
- Pin 2 y 3: Son pines que pueden activar una interrupción del flujo de programa cuando tome un cierto valor, o se produzca un determinado flanco.
- Pin 3, 5, 6, 9, 10 y 11: Son puertos que admiten una salida del tipo PWM (Pulse-Width Modulation), que consiste en una señal digital variable de alta frecuencia (en el caso de Arduino, unos 500Hz) en el que se regula el porcentaje del ciclo que la salida se encuentra a '1'. De esta forma, Arduino es capaz de simular una salida analógica, que puede tomar valores entre 0 V y 5 V.
- Pin 10, 11, 12 y 13: Son pines que se pueden utilizar para implementar una comunicación SPI (Serial Peripheral Interface), un modo síncrono de comunicación entre un microcontrolador y uno o más dispositivos periféricos rápidamente en distancias cortas, habitualmente cuando ambos dispositivos se encuentran en la misma placa.

- ANALOG: Son pines que son capaces de leer señales analógicas cuyo valor se encuentre entre 0 V y 5 V.

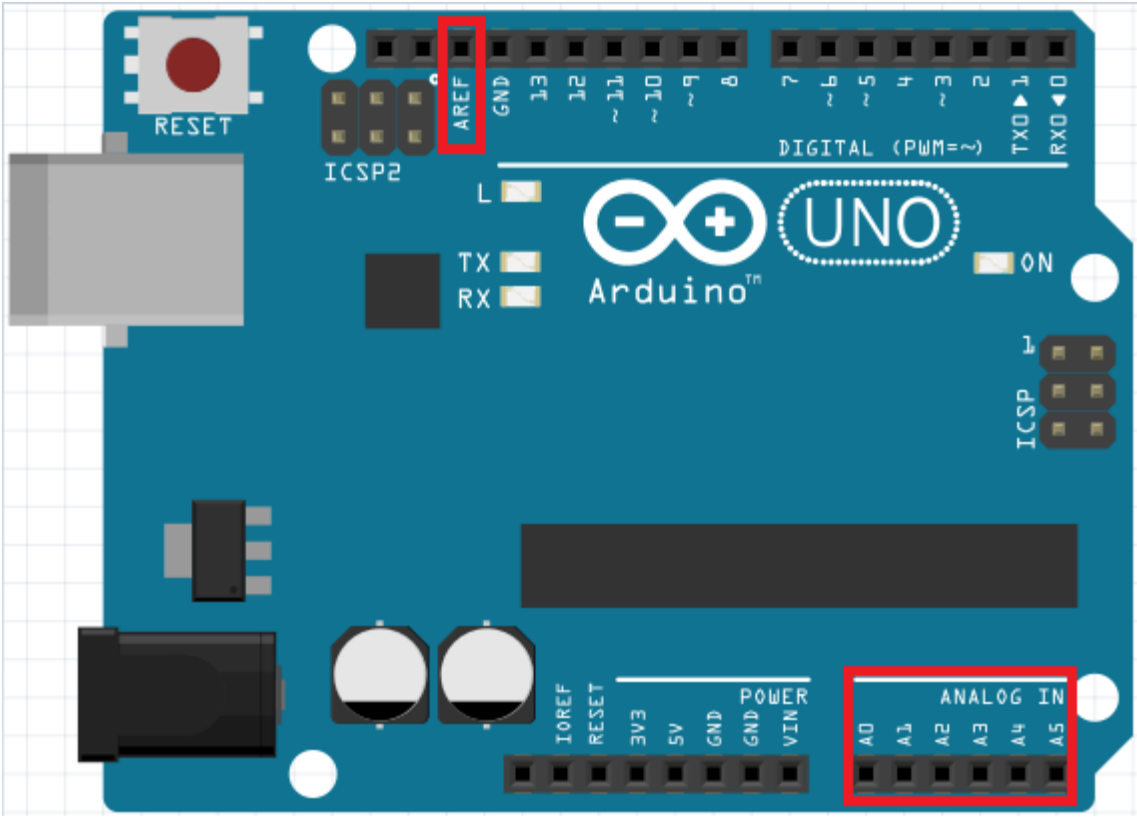

Figura 4.20 – Pines analógicos del Arduino Uno

Según su función, existen dos tipos de pines analógicos:

- Pines desde A0 hasta A5: Estos puertos son entradas analógicas directamente conectadas a un convertidor A/D con 10 bits de resolución (1024 posibles niveles de tensión)
- Pin AREF: Este puerto es una entrada por la que podremos introducir el valor máximo que queremos que admita el convertidor A/D (que debe coincidir con el valor máximo que puedan tomar las señales que se monitoricen a través de los pines analógicos, y ser menor de 5 V). De esta forma, se consigue que aumente la resolución de la lectura efectuada, al encontrarse los niveles de tensión permitidos entre 0 V y AREF, en lugar de entre 0 V y 5 V.

En lo que se refiere a la interfaz con el PC, la conexión se realiza mediante un cable USB, a través del cual se realiza la comunicación serial para cargar el programa en el Arduino, que posteriormente se ejecutará de forma autónoma. A través de este cable, es posible alimentar también el Arduino, ya que el USB actúa como una fuente de alimentación de 5 V limitada a 500 mA. Otra posibilidad es utilizar una fuente de alimentación externa, cuya tensión puede tener un valor entre 7 V y 12 V, ya que la placa de Arduino cuenta con un regulador de tensión de alimentación que genera unos niveles de tensión internos adecuados.

4.3.1.2. Software

En lo que al software respecta, Arduino tiene su propio IDE (Entorno de Desarrollo Integrado), llamado Arduino Development Environment, de código abierto y de funcionamiento sencillo.

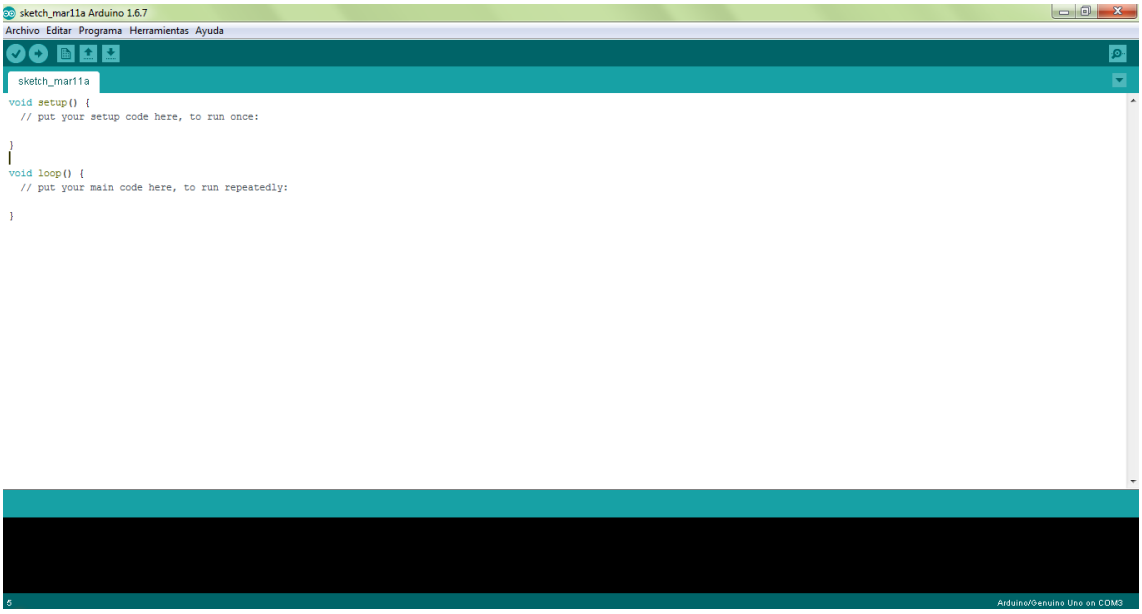

Figura 4.21 – Pantalla principal de Arduino Development Environment

No obstante, Arduino permite sustituir este IDE, en el que la programación se realiza mediante código, por MATLAB/Simulink, un entorno de desarrollo gráfico en el que la programación del Arduino se realice por bloques funcionales.

4.3.1.3. Lenguaje de programación

El lenguaje de programación que implementan todas las versiones de Arduino se denomina Arduino Programming Language, está basado en Wiring, y tiene una sintaxis muy similar a C.

Un programa de Arduino, que dentro del IDE se denomina también sketch, consta de tres secciones separadas:

- Cabecera: Al inicio del sketch, se declaran las variables globales y se incluyen las bibliotecas de instrucciones que vayan a ser necesarias para el desarrollo del programa.
- void setup(): Son instrucciones que se ejecutan una única vez al cargar el programa en la placa, o bien cuando se conecta a una fuente de alimentación. Es una sección en la que se suele inicializar algunas variables, establecer modos de funcionamiento de algunos pines, crear interrupciones, iniciar el uso de bibliotecas, etc.
- void loop(): Las instrucciones dentro de esta sección se ejecutan en un bucle infinito hasta que se apaga o resetea la placa de Arduino. Contiene el conjunto de

instrucciones principales para que el sistema electrónico interactúe con el mundo físico.

Conviene comentar algunas de las características comunes que tiene el lenguaje de Arduino y C, debido a que son factores a tener en cuenta a la hora de implementar o entender un programa de Arduino:

- La sintaxis distingue entre mayúsculas y minúsculas.
- Las tabulaciones sirven para aclarar y ordenar el código visualmente, pero a efectos de compilación no tienen ningún efecto.
- Todas las instrucciones terminan con un punto y coma, para declarar el fin de dicha instrucción.
- Se puede utilizar la misma sintaxis que en C para colocar comentarios clarificadores en el código; bien /*comentario*/, o bien //comentario.

Una de las partes fundamentales que tiene un programa de Arduino, y todo programa en casi cualquier lenguaje de programación, son las variables. Podemos distinguir entre variables globales, declaradas en la cabecera y que pueden ser utilizadas en cualquier sección del código, y variables locales, que se encuentran en una determinada sección y solo pueden ser empleadas en ella.

La sintaxis para declarar una variable es análoga a la de C, utilizando la sintaxis *tipoVariable nombreVariable;*.

Podemos darle cualquier nombre a la variable siempre que no contenga espacios o caracteres especiales, siendo recomendable siempre utilizar nombres representativos de la función que la variable desempeña en el programa.

En cuanto a los tipos de variables disponibles en Arduino, son bastante similares a los disponibles en C. Para el caso del presente proyecto, los tipos más importantes son *boolean*, que almacena un valor cierto o falso, e *int*, que almacena un número entero. Además de estos tipos, Arduino también ofrece otros tipos como *char*, *byte*, *short*, *float*, *double*, *array*, *long*, *unsigned long*, etc. que pueden ser consultados en la bibliografía.

Las principales instrucciones que se van a utilizar en el presente proyecto se detallan a continuación, acompañadas de una pequeña descripción.

- *pinMode(pin,mode)*: Se utiliza habitualmente en la sección *void setup()* para definir pines digitales como entradas o salidas. El primero de los parámetros es el número del pin que queremos modificar, mientras que el segundo parámetro se utiliza para definir si va a ser entrada (*INPUT*) o salida (*OUTPUT*).
- *digitalWrite(pin,value)*: Se emplea para cambiar el estado de un pin digital que ha sido inicializado como salida. El primer parámetro es el número del pin a modificar, mientras que el segundo parámetro puede tomar los valores de HIGH (‘1’) o LOW (‘0’).
- *analogWrite(pin,value)*: Se emplea para generar una simulación de una salida analógica utilizando uno de los pines con posibilidad de PWM, que previamente debe haber sido definido como salida. El primer parámetro es el pin a utilizar, que

debe ser el 3, 5, 6 , 9, 10 u 11, mientras que el segundo parámetro es un valor numérico entero que oscila entre 0 y 255, y que determina el ancho del pulso obtenido. Así, un valor de 255 dará como resultado un pulso del 100%, equivalente a una tensión de 5 V, un valor de 0 dará como resultado un pulso del 0%, equivalente a una tensión de 0 V, mientras que un valor de 127 dará lugar a un pulso de aproximadamente el 50%, lo que representa un valor medio de tensión a la salida de 2,5 V.

- *analogRead(pin)*: Con esta instrucción se puede leer el valor de tensión que entra en un determinado pin analógico. La instrucción devuelve un valor entre 0 y 1023, correspondiente al valor de tensión leído. Un parámetro importante a efectos del presente proyecto, y que podemos obtener de las referencias bibliográficas de Arduino, es el máximo número de lecturas por segundo. El proceso de conversión analógico/digital dura unos 100 microsegundos, lo que significa que, en el mejor de los casos, la máxima frecuencia de lectura analógica que podemos utilizar es de 10.000 lecturas por segundo.
- *delay(ms)*: Pausa la ejecución del programa durante *ms* milisegundos. Existe también la función *delayMicroseconds(us)*, para indicar el tiempo que debe detenerse el programa en microsegundos.
- *map(value, fromLow, fromHigh, toLow, toHigh)*: Esta instrucción transforma un cierto valor de una escala a otra. Se suele utilizar, por ejemplo, con una lectura analógica, que tiene 10 bits de resolución (toma valores entre 0 y 1023) que queremos pasar a una señal de salida PWM, que tiene únicamente 8 bits de resolución (toma valores de 0 a 255). Los límites entre los que se encuentra acotado inicialmente el valor se colocarán en los parámetros *fromLow* y *fromHigh*, mientras que los nuevos límites se colocarán en los parámetros *toLow* y *toHigh*.
- *if(condicion){}/else{}*: De sintaxis similar a C, sirve para condicionar la ejecución de una serie de instrucciones al cumplimiento o no de una serie de premisas.
- *switch(var)/case*: Al igual que para la instrucción anterior, condiciona la ejecución de una determinada instrucción al cumplimiento de una premisa, siendo ésta el valor que toma la variable *var*. Cada uno de los valores se coloca como un *case*, y tiene unas instrucciones independientes. Además, la opción *default* indica las instrucciones que ejecutar en caso de que el valor no coincida con ninguno de los previstos. Para acabar el set de instrucciones de cada *case*, es necesario colocar la instrucción *break;*.

Por último, es necesario tener también algunos conocimientos complementarios acerca de los operadores que el lenguaje de Arduino permite, que se pueden dividir en operadores aritméticos, lógicos y de comparación:

- Operadores aritméticos:
  - Operador suma (+)
  - Operador resta(-)
  - Operador multiplicación (*)

  - Operador división (/)
  - Operador módulo (%)
- Operadores lógicos:
  - Operador AND (&&)
  - Operador OR (||)
  - Operador NOT (!)
  - Operador AND bit a bit (&)
  - Operador OR bit a bit (|)
  - Operador NOT bit a bit (~)
- Operadores de comparación:
  - Operador de igualdad (==)
  - Operador de no igualdad (¡=)
  - Operador menor que (<)
  - Operador menor o igual que (<=)
  - Operador mayor que (>)
  - Operador mayor o igual que (>=)

Finalmente, es necesario hacer mención a los *timers*, o interrupciones programadas. La razón de ser de los *timers* es que, cuando ejecutamos un *delay* para temporizar, detenemos por completo la ejecución del programa de Arduino, no pudiendo ejecutar ninguna otra instrucción durante el tiempo que dura la interrupción. Esto es admisible en programas muy simples, o bien cuando los periodos a temporizar son muy pequeños.

Sin embargo, cuando deseamos ejecutar en Arduino varias tareas con distinta periodicidad, o bien poder seguir ejecutando instrucciones mientras se lleva a cabo un proceso de temporización largo, la opción más adecuada es la utilización de un *timer*, que permite llevar a cabo una temporización en segundo plano, mientras seguimos ejecutando otras instrucciones, de forma que cuando se cumpla el tiempo de la temporización se active una interrupción que despierta la ejecución de las instrucciones temporizadas.

4.3.2. Microcontrolador PIC16LF1937

Los microcontroladores PIC pertenecen a una familia de microcontroladores fabricados por Microchip Technology Inc., derivados del PIC1650, desarrollado por la división de microelectrónica de General Instrument.

El PIC original se creó para ser empleado como controlador de la interfaz con los periféricos de la CPUde 16 bits CP16000, que pese a tener unas buenas prestaciones de cálculo, obtenía unos pobres resultados cuando se trataba de la gestión de entradas y salidas, por lo que el PIC de 8 bits se desarrolló en 1975 para mejorar el rendimiento del sistema.

En 1985, la división de microelectrónica de General Instrument se separa como una compañía independiente, cambiando su nombre a Microchip Technology en 1987 y siendo adquirida por un nuevo propietario en 1989 que canceló gran parte de los desarrollos que se estaban desarrollando en ese momento.

Sin embargo, el PIC se mejoró añadiendo una memoria EEPROM, creando un controlador programable. Hoy en día, PIC está presente en numerosos módulos y kits con diferentes periféricos, con una gran variedad de tamaños de memoria y de palabras.

El PIC16LF1937, que viene incluido en el kit de evaluación del inversor DM164130-2 junto a una pantalla LCD e interfaz para control sin sensores de motores BLDC, es un microcontrolador de gama media, con una frecuencia de funcionamiento de 32MHz y 16 bits de longitud de palabra.

Esto lo convierte en un microcontrolador más potente que el Arduino Uno, previamente analizado, pese a que presenta un menor tamaño de espacio de almacenamiento.

Figura 4.22 – Kit de evaluación DV164132, con microcontrolador PIC16LF1937

El microcontrolador PIC tiene la ventaja de ser más potente que el Arduino Uno, pero tiene como desventaja ser más costoso, menos extendido y, sobre todo, la incompatibilidad para la programación gráfica utilizando MATLAB/Simulink, uno de los objetivos básicos del presente proyecto.

4.3.3. Controladores BLDC integrados

Como alternativa a la utilización de un microcontrolador conectado a un inversor, se ha planteado la utilización de un controlador BLDC integrado. Estos dispositivos hardware están diseñados específicamente para el control de velocidad de motores BLDC, por lo que incluyen, en un mismo circuito integrado, la computación necesaria para establecer la secuencia de conmutación de las bobinas, y la etapa de potencia para suministrar al motor.

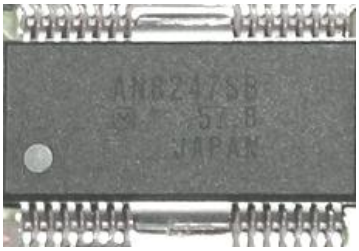

Figura 4.23 – Controlador integrado de motor BLDC AN8247SB

Los controladores BLDC integrados llegan hasta tal punto que existen motores que llevan el controlador de velocidad ya incorporado, de forma que se conectarán directamente a la batería de alimentación, como los de MicroMo Electronics, que se muestran en la figura 4.24.

Figura 4.24 – Motor con el controlador de velocidad integrado

El control en este tipo de dispositivos se realiza de manera muy simple para el usuario, ya que solo precisan que se les introduzca una serie de señales digitales para controlar algunos parámetros básicos de funcionamiento del motor, el bus de alimentación y una señal de tensión para regular la velocidad. En la figura 4.25 aparece el esquema de control perteneciente a un motor BLDC de Anaheim Automation con controlador de velocidad integrado.

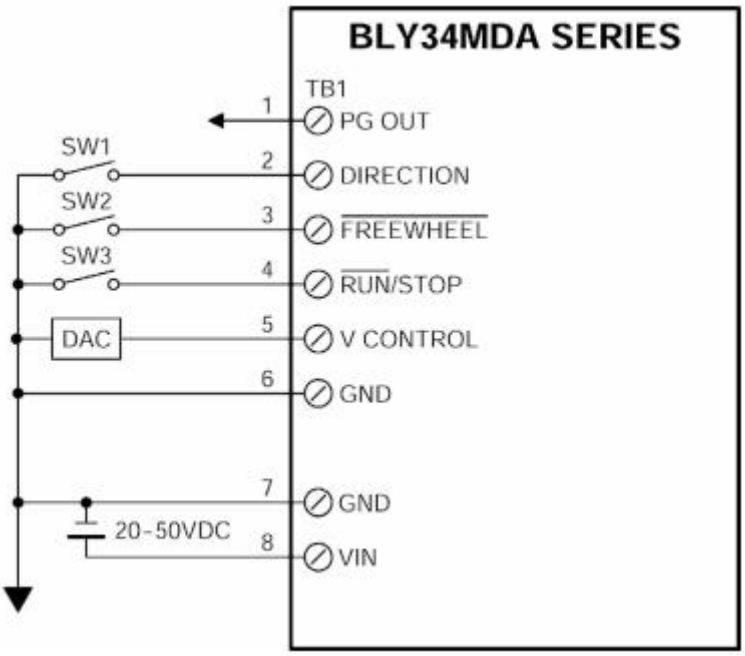

Figura 4.25 – Esquema de control de un motor BLDC con controlador integrado

Este tipo de controladores presentan la principal ventaja de la robustez del sistema y el pequeño espacio que se ocupa con respecto a controladores tradicionales, pero tienen como inconvenientes el alto coste de los sistemas integrados y la falta de personalización y el hermetismo del funcionamiento interno, lo que los hacen poco recomendables cuando de lo que se trata es de acometer un proyecto de investigación como en ante el que nos encontramos.

4.3.4. Criterios de selección

Para el presente proyecto, se va a seleccionar el microcontrolador Arduino Uno, por una razón fundamental. Uno de los objetivos principales que se persiguen con este trabajo es el de desarrollar la programación de microcontroladores mediante bloques gráficos utilizando la herramienta MATLAB/Simulink, aplicándola al control de velocidad de motores BLDC.

Dado que Arduino es el único dispositivo de los estudiados que nos permite desarrollar este objetivo, se han descartado el resto de opciones. Otras ventajas que tiene Arduino en relación a los requisitos del presente proyecto son:

- Arduino es una plataforma de código y hardware abierto, es decir, nos permite acceder a la descripción del completo funcionamiento de las placas, de forma que se puede tener un conocimiento global del funcionamiento del circuito, pudiendo adaptar las especificaciones de éste en caso de que sea necesario para cumplir las especificaciones del proyecto, ya que además la compañía distribuye gratuitamente los archivos Eagle, diagramas y componentes de sus placas.
  Además, Arduino ofrece diferentes modelos de placa, con especificaciones distintas que se puedan adaptar a proyectos de múltiples niveles de complejidad, pudiendo reutilizar código entre placas con apenas unos pocos cambios, lo que incrementa la flexibilidad y versatilidad de la plataforma. Arduino también distribuye una gran variedad de periféricos o *shields*, que amplían las capacidades de la placa con tecnologías específicas como Ethernet, GSM, Wi-Fi, ZigBee, etc.
- Arduino presenta una gran facilidad y sencillez de programación, en parte por sus similitudes con C, en parte por ser un lenguaje de muy alto nivel, con funciones preestablecidas que reducen la lógica a lectura de entradas, control de tiempos y salidas de una manera muy intuitiva. Además, no precisa de ningún tipo de tarjeta de programación, sino que la misma placa se conecta vía USB al ordenador.
- Arduino actualmente está extendido a nivel mundial, lo que implica que existe una gran cantidad de contenido fruto de desarrollos y trabajos difundidos a través de la comunidad en forma de bibliotecas que pueden ser aprovechadas en nuevos proyectos.
- El bajo coste en comparación con otros microcontroladores y la posibilidad de reutilización en múltiples proyectos hacen de Arduino una plataforma especialmente adecuada para desarrollar prototipos en proyectos de investigación.

4.4. Integración de Arduino con Simulink

4.4.1. Introducción

Conociendo la intención de Mathworks de desarrollar una gran variedad de toolboxes para expandir el rango de aplicaciones de MATLAB, la simplicidad y lo intuitivo de su modo de desarrollar modelos gráficos mediante bloques de Simulink y la gran difusión y grado de utilización que tiene Arduino en la actualidad a nivel global, era cuestión de tiempo que se integraran ambas plataformas.

Este proceso ocurre a partir del año 2012, con la aparición de dos paquetes de soporte para hardware Arduino:

- *Paquete de soporte de Matlab para Arduino*: Paquete que precisa de MATLAB como producto base, y que permite adquirir y enviar señales mediante Arduino Uno, Due o Mega, conectando directamente MATLAB a la placa de Arduino mediante el cable USB. Utilizando las funciones de este paquete, toda la ejecución del programa diseñado se lleva a cabo en la computadora que tiene instalado MATLAB, mientras que Arduino actúa exclusivamente como una interfaz con el mundo físico, constituyendo un bloque de entradas y salidas.
- *Paquete de soporte de Simulink para Arduino*: Paquete que precisa de Simulink como producto base, y que está disponible para Windows a partir de la versión r2012b, dividido en dos subpaquetes, uno para Arduino Uno, Mega, Leonardo y más, y otro para Arduino Due. Los algoritmos desarrollados utilizando este paquete de software sí que se ejecutarán directamente en el propio Arduino, mientras que la computadora se empleará exclusivamente para desarrollar el código, pudiendo crear un modelo de diagramas de bloques en Simulink para diseño de sistemas de control, aplicaciones robóticas, etc.; y llevar a cabo simulaciones que permitan verificar el correcto funcionamiento del programa. Una vez testeados, con un simple clic de ratón se arrancará la generación de código fuente para transmitir a Arduino, ejecutando el código de manera embebida en la placa.

En función de las necesidades de cada proyecto, será interesante descargar uno u otro paquete de software, en función de lo que se desee desarrollar y el uso que se quiera dar a la placa Arduino dentro del modelo. En el ámbito del presente Trabajo de Fin de Máster, lo interesante será utilizar un modelo de Simulink que se pueda ejecutar embebido en la placa de Arduino, aún estando esta desconectada del ordenador, por lo que el paquete a instalar será el de Simulink para Arduino.

La versión de MATLAB empleada para el presente proyecto será la r2015a para Windows.

4.4.2. Instalación del paquete de soporte y puesta a punto

Para instalar la toolbox de soporte para Arduino en MATLAB r2015a, se han seguido las instrucciones del fabricante, desplegando el menú *Add-Ons* en la consola de MATLAB. En

concreto, el paquete de Simulink para Arduino se encuentra dentro de los paquetes de soporte hardware, entre los que habrá que seleccionar el tipo de hardware que se dispone (en el caso del presente trabajo, Arduino Uno). Siguiendo el procedimiento de instalación, el programa estará listo para crear modelos en Simulink ejecutables en Arduino.

Para comenzar a trabajar, se lanza el explorador de bibliotecas de Simulink colocando el comando 'simulink' en el terminal de MATLAB. Los módulos pertenecientes al paquete de soporte de Simulink para Arduino se encuentran en la ruta '*Simulink Support Package for Arduino*', combinándolos con los bloques convencionales de Simulink.

4.4.3. Contenido del paquete de soporte de Simulink para Arduino

El paquete de soporte de Simulink para Arduino contiene los bloques que se muestran en la figura 4.26.

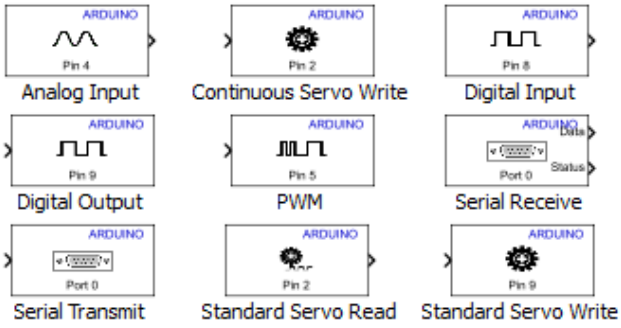

Figura 4.26 – Bloques del paquete de Simulink para Arduino

- Analog Input: Permite asignar uno de los pines analógicos del Arduino Uno, que puede ser desde el A0 hasta el A5. En el sistema genera un valor entero de entre 0 y 1023, en función del valor de tensión analógica que se mida en el pin correspondiente (resolución de 10 bits)
- Continuous Servo Write: Permite asignar un valor de tensión analógico de salida construido mediante un control PWM, para alimentar un servomotor. La tensión que recibe el servo será proporcional a la velocidad de giro de éste.
- Digital Input: Permite asignar uno de los pines digitales del Arduino Uno, desde el 0 hasta el 13, como entrada digital. El sistema recibe un uno o un cero en función de la entrada que dicho pin reciba.
- Digital Output: Permite asignar uno de los pines digitales del Arduino Uno, desde el 0 hasta el 13, como salida digital. El sistema puede generar en este bloque un 1 o un 0, que será generado en el pin.
- PWM: Permite seleccionar como salidas PWM con una resolución de 8 bits (el bloque recibe una señal entera entre 0 y 255). Los pines que admiten este bloque son los mencionados en el apartado 4.3.1.1.
- Serial Receive: Este pin se asignará al pin 0 (RX, que es común con el cable RX que comunica el Arduino con el ordenador a través del cable USB), y permite recibir datos mediante comunicación serial a través del conector USB de la placa.

- **Serial Transmit:** Este pin se asignará al pin 1 (TX, que es común con el cable TX que comunica el Arduino con el ordenador a través del cable USB), y permite transmitir datos mediante comunicación serial a través del conector USB de la placa.
- **Standard Servo Read:** Permite leer la posición de un servo, a través de una entrada digital, devolviendo la posición en grados sexagesimales.
- **Standard Servo Write:** Permite establecer la posición de un servo mediante un PWM, por lo que este bloque se le debe asignar únicamente a pines con esta funcionalidad. El módulo recibe el dato de la rotación en grados sexagesimales.

4.4.4. Programa para el control utilizando sensores Hall

Partiendo del modelo de simulación pura desarrollado en Simulink, se van a sustituir todos aquellos elementos ajenos al microcontrolador por algún bloque de entrada o salida de Arduino. El esquema correspondiente al programa para utilizar sensores Hall se muestra en la figura 4.27.

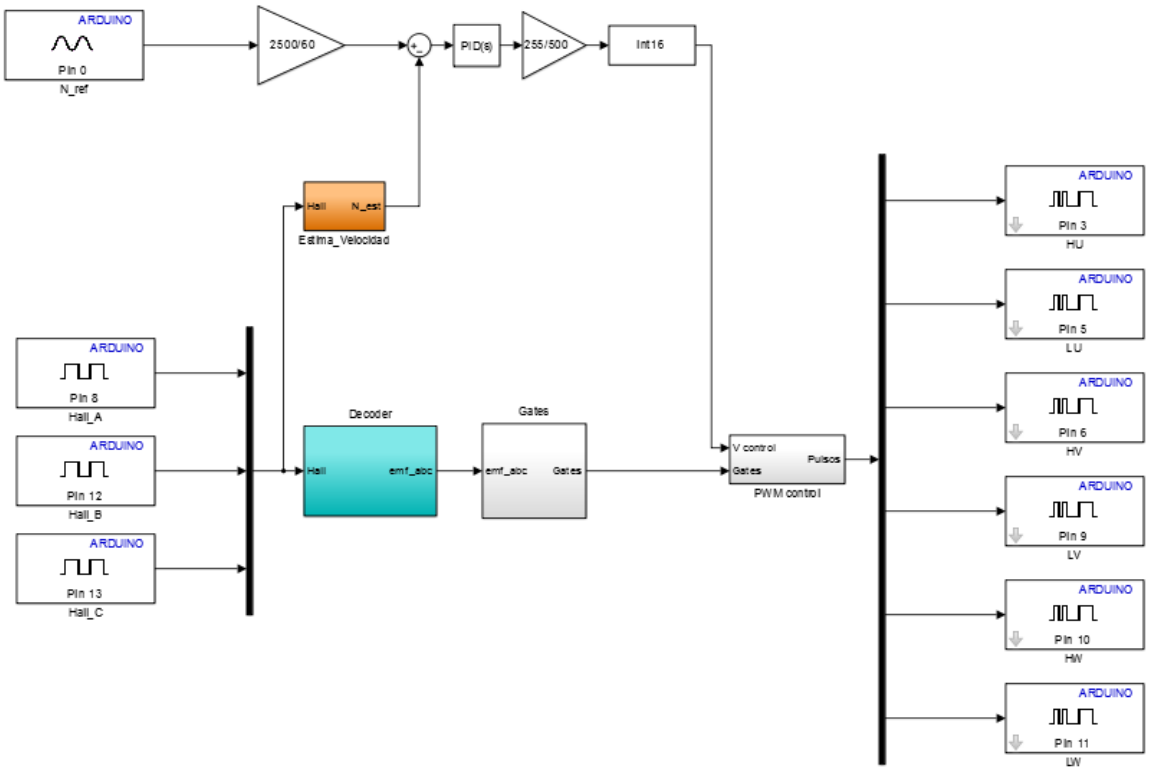

Figura 4.27 – Programa desarrollado para el control utilizando sensores Hall

Podemos observar en el diagrama que las señales Hall que en el modelo puro eran generadas por el bloque motor BLDC ahora serán entradas digitales al Arduino. Estas señales pasan, secuencialmente, por los bloques *Decoder, Gates* y *PWM control*. Además, la velocidad ahora se calcula mediante el bloque *Estima_velocidad*, a partir del tren de pulsos generado por las señales Hall, a diferencia del modelo puro, en el que la velocidad era directamente suministrada por el motor a través de un dispositivo encoder o similar.

La estructura interna del bloque *Estima_velocidad* aparece reflejado en la figura 4.28.

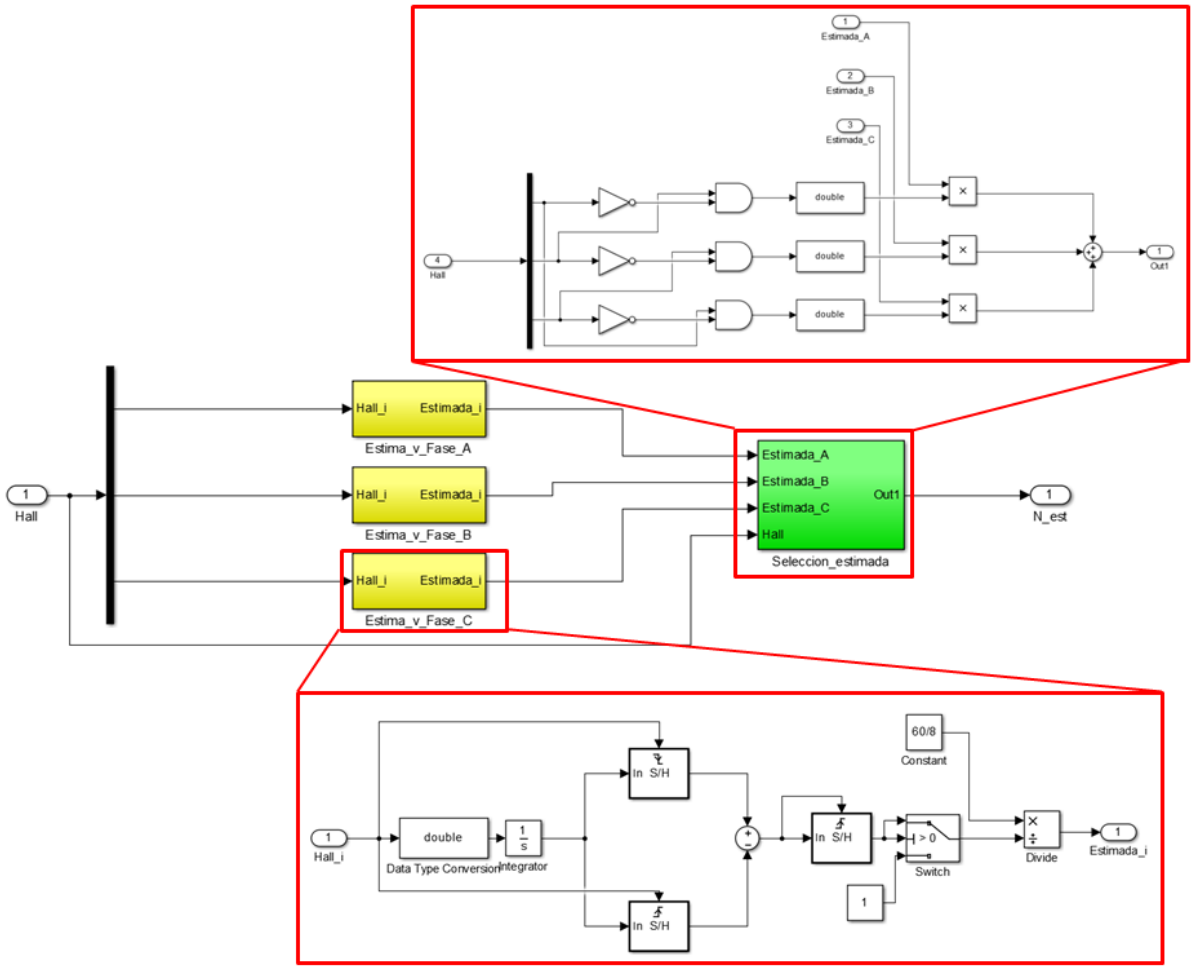

Figura 4.28 – Estructura interna bloque *Estima_velocidad*

En este bloque, se aplica un integrador a cada una de las señales Hall, gracias a lo cual se puede medir el tiempo que transcurre durante medio periodo, correspondiente al tiempo en el que cada uno de los sensores Hall se encuentra en estado activo. Para multiplicar por tres la resolución de la estimación, se ha recurrido a aplicar el proceso de estimación al vector completo de señales Hall.

En la figura 4.29 se muestra las gráficas de velocidad del motor, estando la gráfica superior estimada mediante este bloque, mientras que la gráfica inferior es la suministrada por el bloque del motor BLDC directamente. Observamos que la concordancia de resultados entre ambas gráficas es casi total, salvando los pequeños escalones por pasar de un sistema continuo a uno discreto.

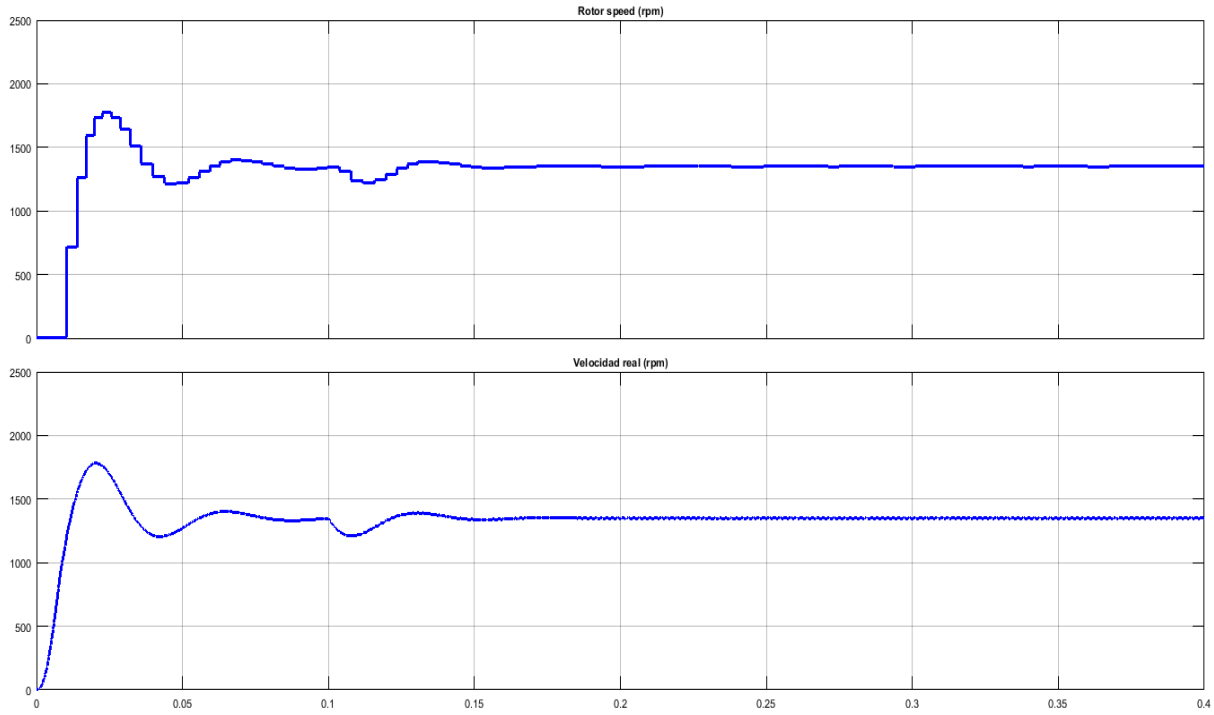

Figura 4.29 – Gráfica de la velocidad estimada (superior) frente a la velocidad real (inferior)

Por su lado, la tensión que servirá como referencia para la velocidad se toma de un potenciómetro (R19) incorporado en la tarjeta inversora, (pin 20 del conector J1 de la figura 4.4), que divide una tensión de 3,3V proporcionada por un regulador. Esta señal entrará al Arduino como entrada analógica leído por el pin A0, bajo el nombre N_ref. En la figura 4.30 podemos ver el detalle del esquema del divisor de tensión.

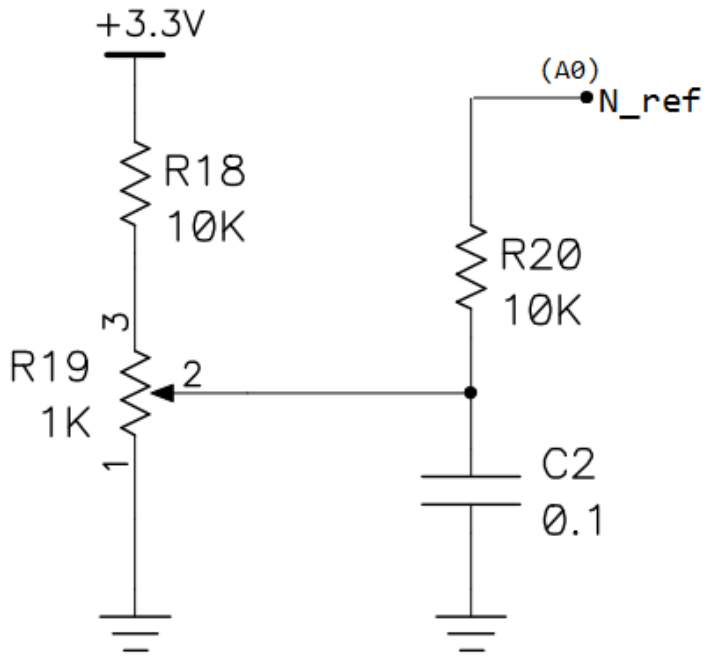

Figura 4.30 – Selector de la velocidad de referencia incorporado en la placa inversora

En la figura 4.31, se muestra el esquema de conexiones del Arduino Uno con el inversor y el motor utilizando los sensores tipo Hall.

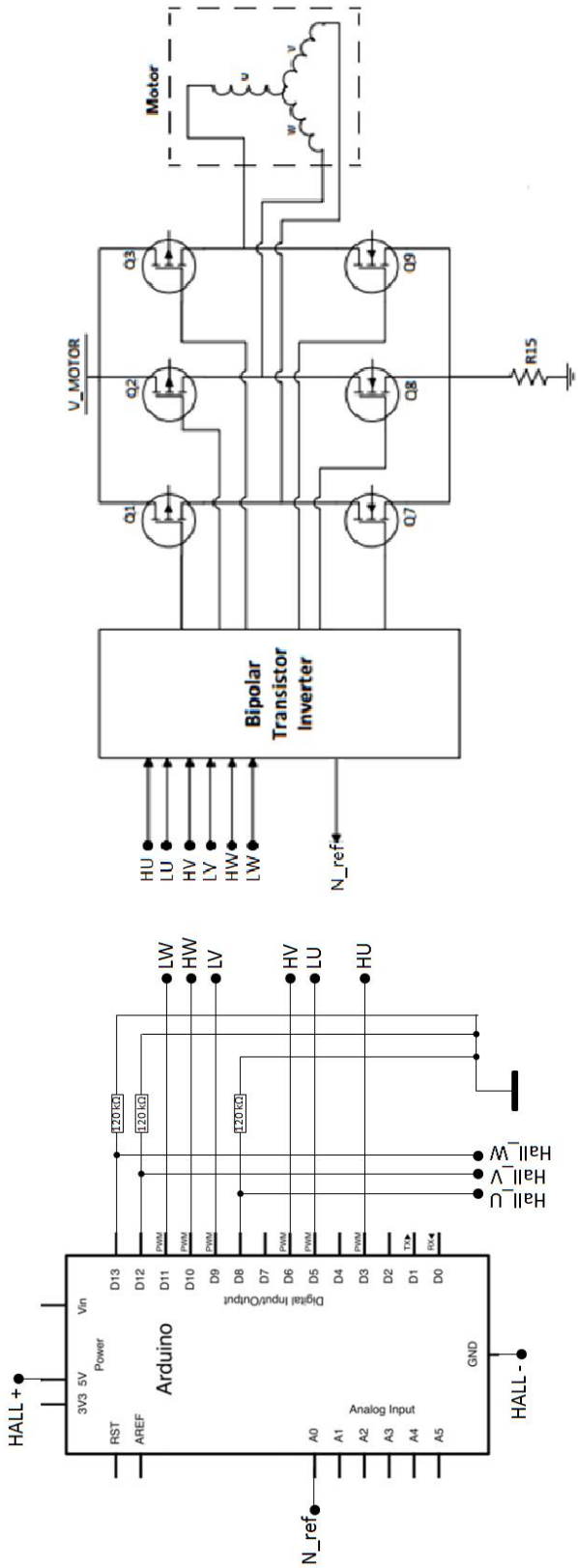

Figura 4.31 – Esquema de conexiones del Arduino con el inversor y el motor usando sensores Hall

4.4.5. Programa para el control sin sensores

En la figura 4.32 aparece reflejado el programa realizado con bloques para el control sin sensores.

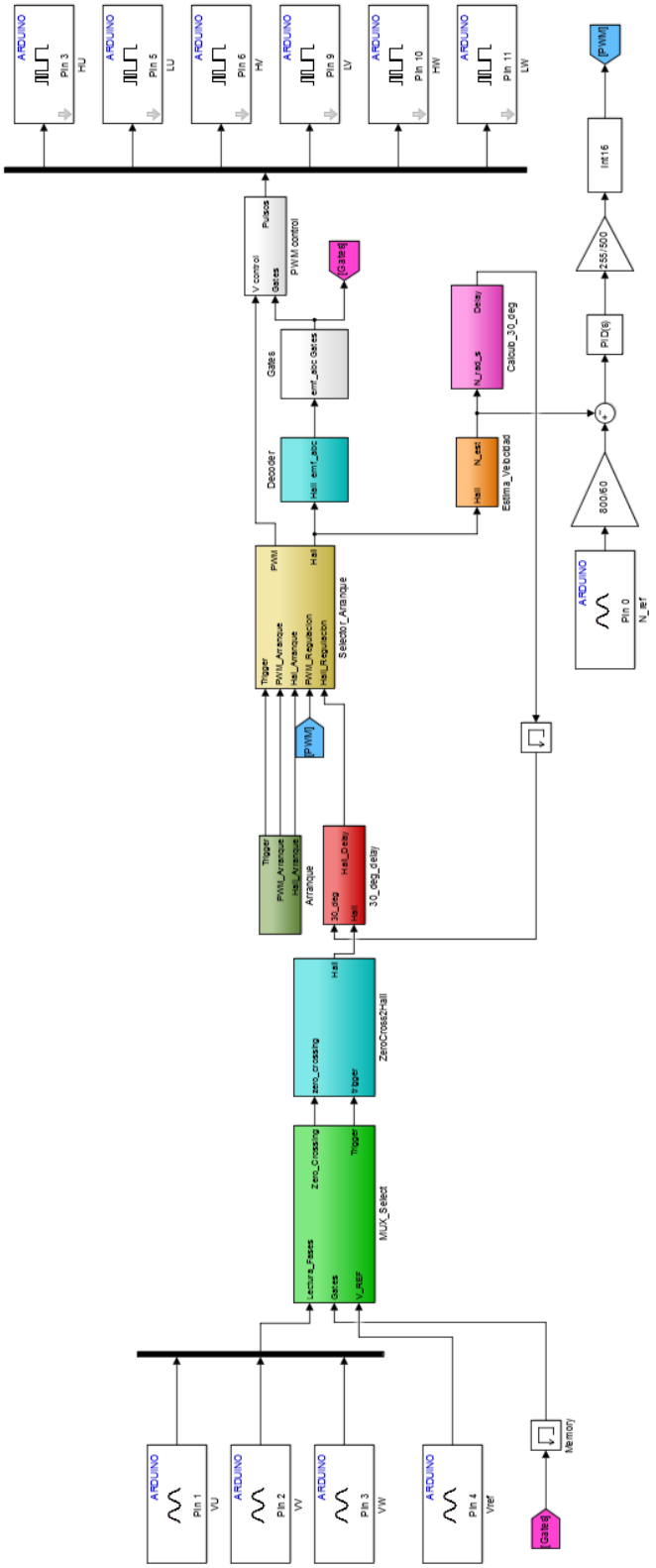

Figura 4.32 - Programa desarrollado para el control sin sensores

En esta ocasión, se realiza una lectura analógica de las tensiones en los bornes de las fases, que pasan por un divisor de tensión integrado en la placa inversora (ver figura 4.4). Estas señales son VU, VV y VW, que se reciben a través de las entradas A1, A2 y A3 respectivamente. Además, también se lee por la entrada A4 la tensión de referencia Vref, que proviene de la tensión a la que se alimenta el motor (12V) pasando por un divisor de tensión variable mediante un potenciómetro (R21). En la figura 4.33 aparece el esquema de las tensiones tomadas.

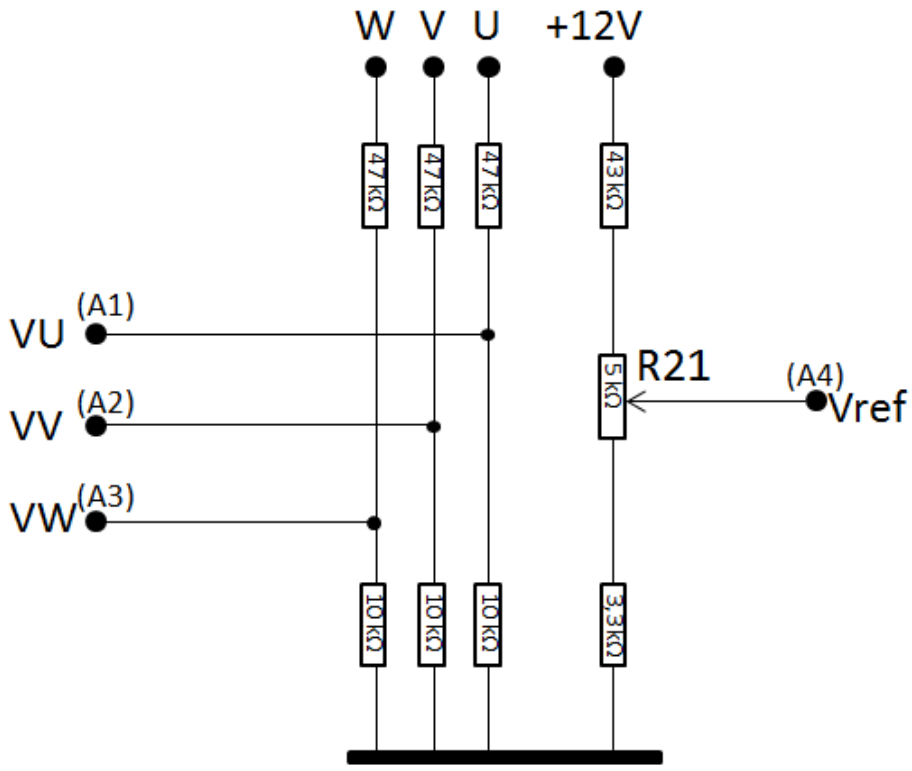

Figura 4.33 – Divisores de tensión integrados en la placa inversora

La tensión Vref se ha incorporado como entrada al bloque MUX_select, que ha variado ahora ligeramente respecto al modelo realizado para simulación. En lugar de detectar los cruces de las tensiones de las fases por cero, ahora detectaremos el cruce con la tensión de referencia.

Por su lado, para la lectura de la velocidad de referencia N_ref se realiza el mismo procedimiento que en el caso del circuito con los sensores Hall, utilizando el potenciómetro incorporado en la tarjeta inversora y leyendo su tensión correspondiente a través de la entrada A0

En lo que se refiere al resto del circuito implementado, el procedimiento que se ha seguido es el mismo que para el caso del programa de control con sensores Hall, colocando los bloques que se habían diseñado previamente en el modelo de simulación.

En la figura 4.34, se muestra el esquema de conexiones del Arduino Uno con el inversor y el motor siguiendo el modelo sin sensores.

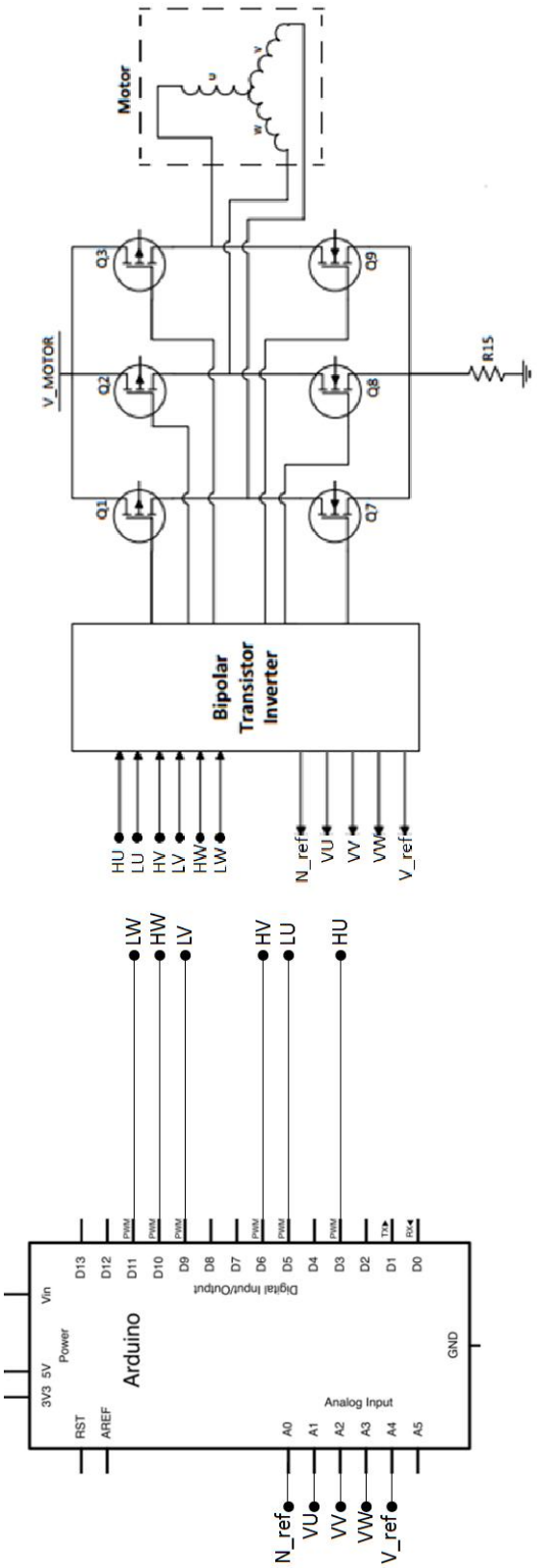

Figura 4.34 – Esquema de conexiones del Arduino con el inversor y el motor sin sensores

4.5. Validación del modelo y puesta a punto

A partir de los programas ejecutados, se ha procedido a su compilación y ejecución sobre el hardware. Para ello, nos hemos dirigido al menú *Tools/Run on Target Hardware/Options*, seleccionando Arduino Uno como microcontrolador objetivo.

Al compilar y cargar el programa en el Arduino, se pueden detectar algunos problemas y limitaciones. El principal es un mensaje que advierte que todos los *timers* están ocupados por las salidas PWM, por lo que la detección de los cambios en las entradas se realizará mediante instrucciones del código en lugar de interrupciones Hardware.

4.5.1. Primeras comprobaciones

Se han realizado pruebas iniciales con el objetivo de verificar la conexión y funcionamiento del inversor sin dañar el motor, para lo cual se ha sustituido este por tres resistencias de $1{,}2k\Omega$ tal y como se muestra en la figura 4.35.

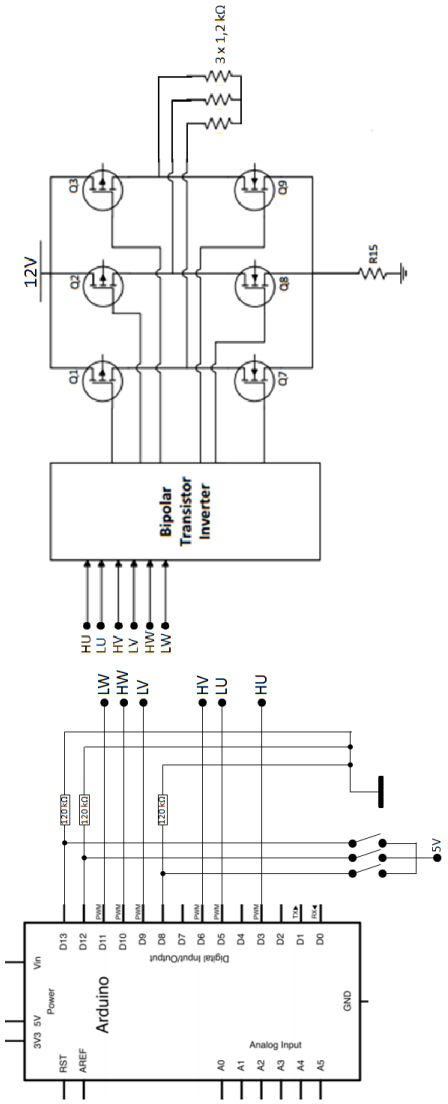

Figura 4.35 – Esquema diseñado para las primeras comprobaciones

Es necesario analizar el conector de la placa inversora, preparada de inicio para la conexión directa de un microcontrolador PIC, y que en el presente proyecto se emplearán pequeños cables o *jumpers* hembra-macho. La denominación del conector de la placa en los esquemas es J1, y su detalle se muestra en la figura 4.36, fragmento extraído del esquema de la figura 4.4.

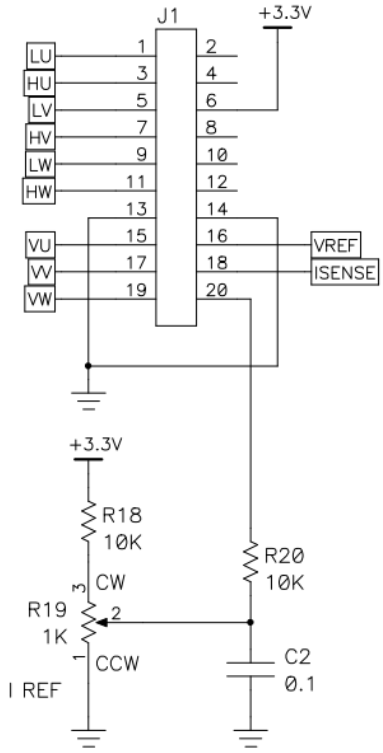

Figura 4.36 – Detalle del conector de la placa inversora con el microcontrolador

En la figura 4.37 se muestra una fotografía del montaje que se preparó para las primeras pruebas.

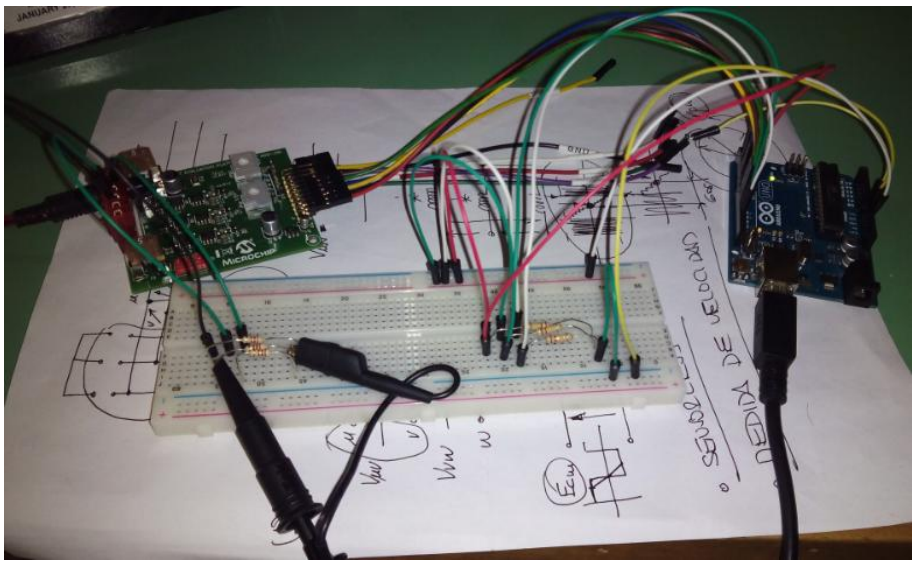

Figura 4.37 – Montaje para las primeras pruebas

Con este montaje, se han realizado tres verificaciones. En primer lugar, se ha comprobado que, conectando los elementos del sistema tal y como se indica en la gráfica de la figura 4.31, el sistema alimentará las fases del motor adecuadamente, sin dañar ninguno de los componentes del sistema. Es necesario mencionar que la tensión con la que se ha alimentado el inversor es de 12V, que es la máxima recomendada por el fabricante del dispositivo, y la corriente se ha limitado a 5A para no dañar el inversor.

En segundo lugar, mediante el empleo de un multímetro, se ha comprobado que, para las diferentes combinaciones posibles de los sensores Hall, la secuencia de activación de las fases que arroja el Arduino es correcta.

Por último, mediante el empleo de un osciloscopio se ha verificado el funcionamiento del inversor actuado mediante el controlador PWM, tal y como se muestra en la figura 4.38 para distintas fases y valores del PWM.

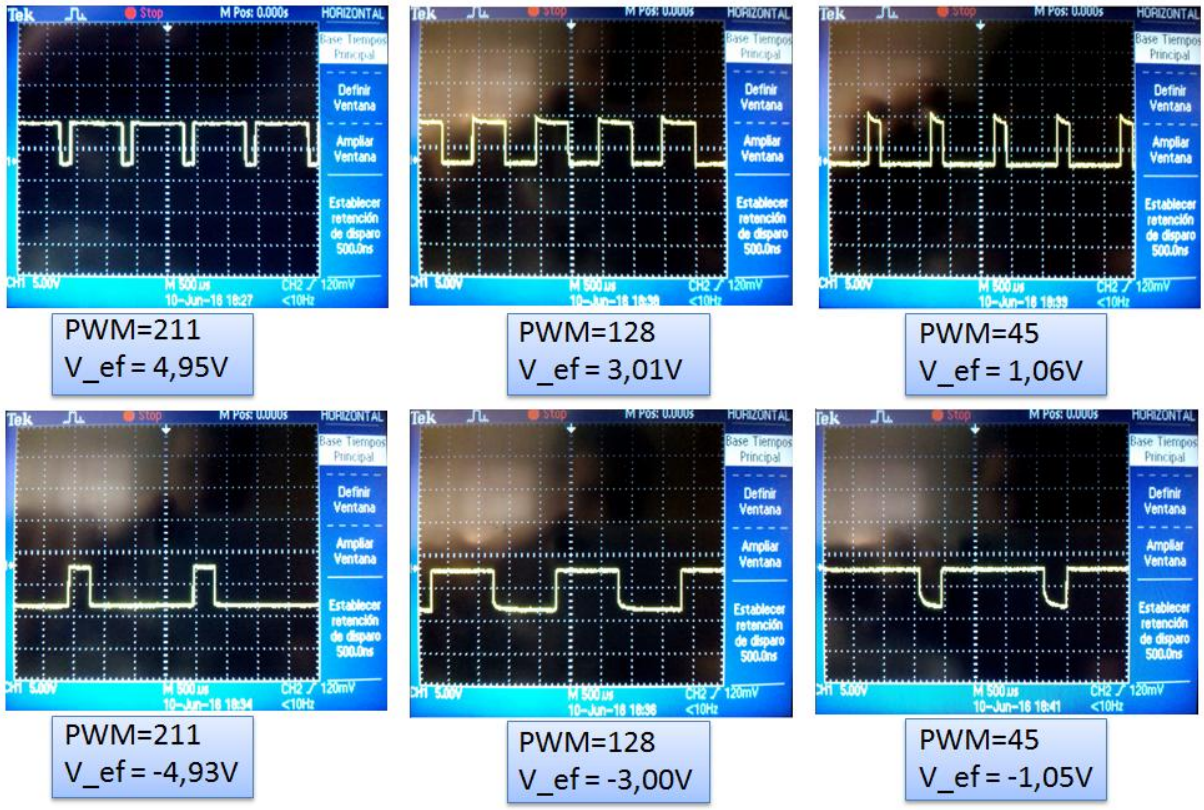

Figura 4.38 – Resultados de la verificación del funcionamiento del inversor mediante el osciloscopio

4.5.2. Implementación del modelo *sensorless*

Una vez se ha comprobado que el montaje con los sensores tipo Hall funciona correctamente, se va a probar el montaje con el motor incorporado junto a la placa inversora. Dado que éste no dispone de sensores tipo Hall montados, se va a utilizar el modelo *sensorless* desarrollado para Arduino.

El motor se conecta a través del bus del motor, cuyas señales se pueden ver en la figura 4.39, extraída del esquema del inversor de la figura 4.4.

Figura 4.39 – Montaje y esquema de conexiones del bus del motor con la placa inversora

El montaje que se ha realizado es el correspondiente al esquema de la figura 4.34, y se puede observar en la fotografía de la figura 4.40.

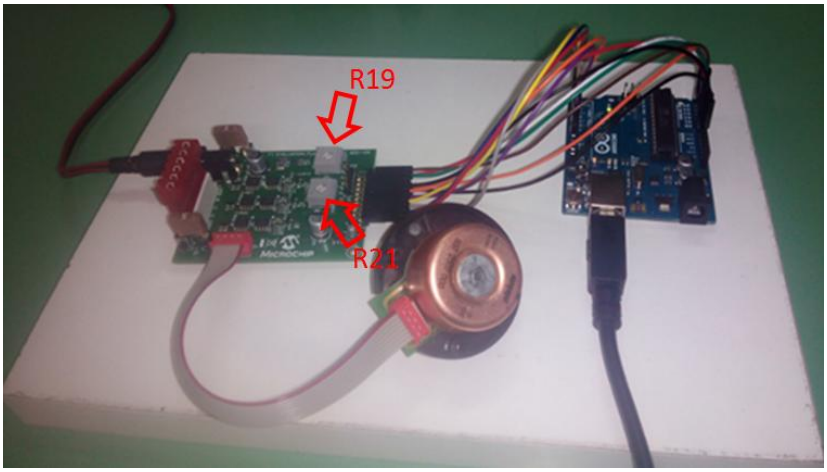

Figura 4.40 – Montaje con el esquema *sensorless* y el Arduino Uno

4.5.2.1. Primeras limitaciones detectadas

Una vez montado el sistema de la figura 4.40, se ha tratado de transferir el código construido mediante bloques de Simulink para el control *sensorless* (figura 4.32 al Arduino Uno), detectándose una serie de problemas.

En primer lugar, si tratamos de compilar y transferir el código al Arduino Uno sin realizar ningún tipo de configuración, saltará un mensaje de error, el de la figura 4.41, que indica que no es posible escribir el código utilizando temporizadores hardware (Arduino Uno tiene únicamente dos), porque todos están ocupados en generar las señales PWM. El mensaje sugiere cambiar el modo de compilación, a *SingleTasking*, es decir, a un código en el que las acciones se realicen mediante instrucciones secuenciales.

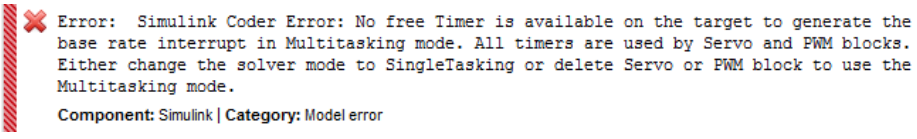

Figura 4.41 – Mensaje de error cuando se usan interrupciones hardware

Si realizamos las acciones pertinentes para corregir el error, seleccionando el modo *SingleTasking,* a la hora de compilar y transferir nos encontramos con otro mensaje de error, el de la figura 4.41, que advierte de un nuevo problema. Al no utilizar los timers, necesita emplear gran parte de la memoria interna en almacenar datos correspondientes

a variables temporales, lo que provoca que se exceda el tamaño de la escasa memoria disponible para datos en el Arduino Uno, de 2KB, en más del 300%.

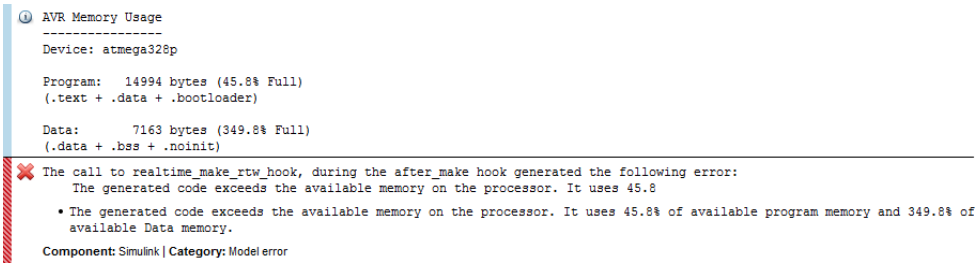

Figura 4.42 – Mensaje de error al tratar de introducir el modelo *sensorless* de Simulink en el Arduino Uno

Dado que los dos errores que ha devuelto Simulink tienen que ver con las características del hardware en el que se está descargando el código, y en particular el último deja muy claro que, por razón de tamaño de memoria no es posible utilizar el algoritmo de control *sensorless* mediante un Arduino Uno, trataremos de cambiar a su hermano mayor: el Arduino Mega 2560.

4.5.2.2. Puesta a punto y pruebas con Arduino Mega 2560

El Arduino Mega 2560 es un microcontrolador basado en el chip Atmega2560, de mayores prestaciones que el Atmega 328P que incorpora el Arduino Uno.

Las características más interesantes del Arduino Mega 2560 son, en cuanto a periféricos, 54 entradas/salidas digitales (frente a las 14 del Uno), de las cuales 15 pueden ejecutar un PWM (6 en el Uno), 16 entradas analógicas (6 en el Uno) y 4 puertos serie (solo uno en el Uno).

En lo que se refiere a potencia de procesamiento y capacidad de almacenamiento, el Mega 2560 tiene una velocidad de reloj de 16 MHz, la misma que en el Uno. Además, dispone de una memoria compuesta por 256 KB de memoria flash (32 KB del Uno), 8 KB de SRAM (2 KB en el Uno) y 4 KB de EEPROM (1 KB en el Uno). A efectos del presente proyecto, un dato muy importante a destacar es la posibilidad de utilizar hasta cinco *timers* distintos, por dos que permitía el Arduino Uno.

Figura 4.43 – Arduino Mega 2560

Para cambiar el Arduino Uno por el Mega 2560 en el sistema *sensorless* del presente proyecto, se ha modificado el código de Simulink y el esquema del montaje, cambiando unos pines por otros más convenientes, según se muestra en la figura 4.44.

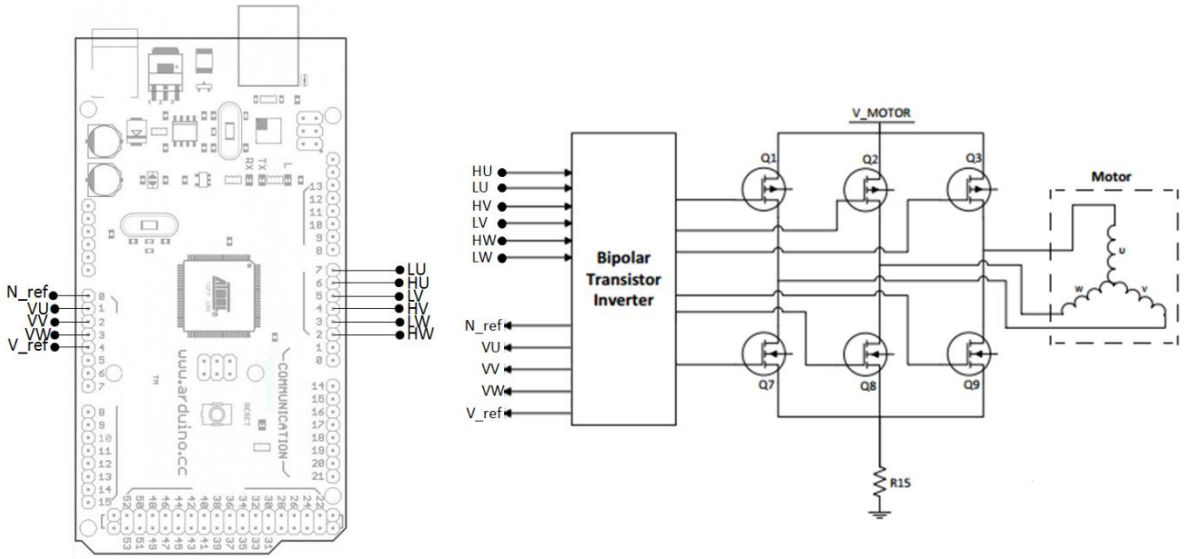

Figura 4.44 – Esquema de conexiones del sistema sin sensores con el Arduino Mega 2560

El montaje físico que se ha realizado siguiendo el esquema anterior se muestra en la fotografía de la figura 4.45.

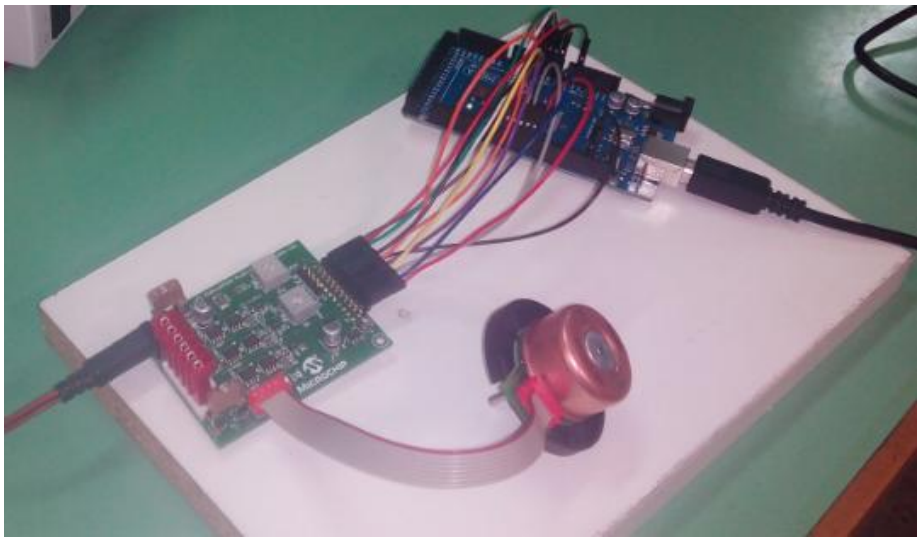

Figura 4.45 – Montaje con el esquema *sensorless* y el Arduino Mega 2560

En la figura 4.46 se muestra el código que se ha desarrollado para probar que Arduino Mega 2560 cumple con los requerimientos que comprometían la continuidad del proyecto en el caso del Arduino Uno. Para las pruebas, no se ha utilizado el regulador PID para controlar la velocidad de giro, sino que se empleará el potenciómetro R19 para modificar el porcentaje de ciclo del PWM.

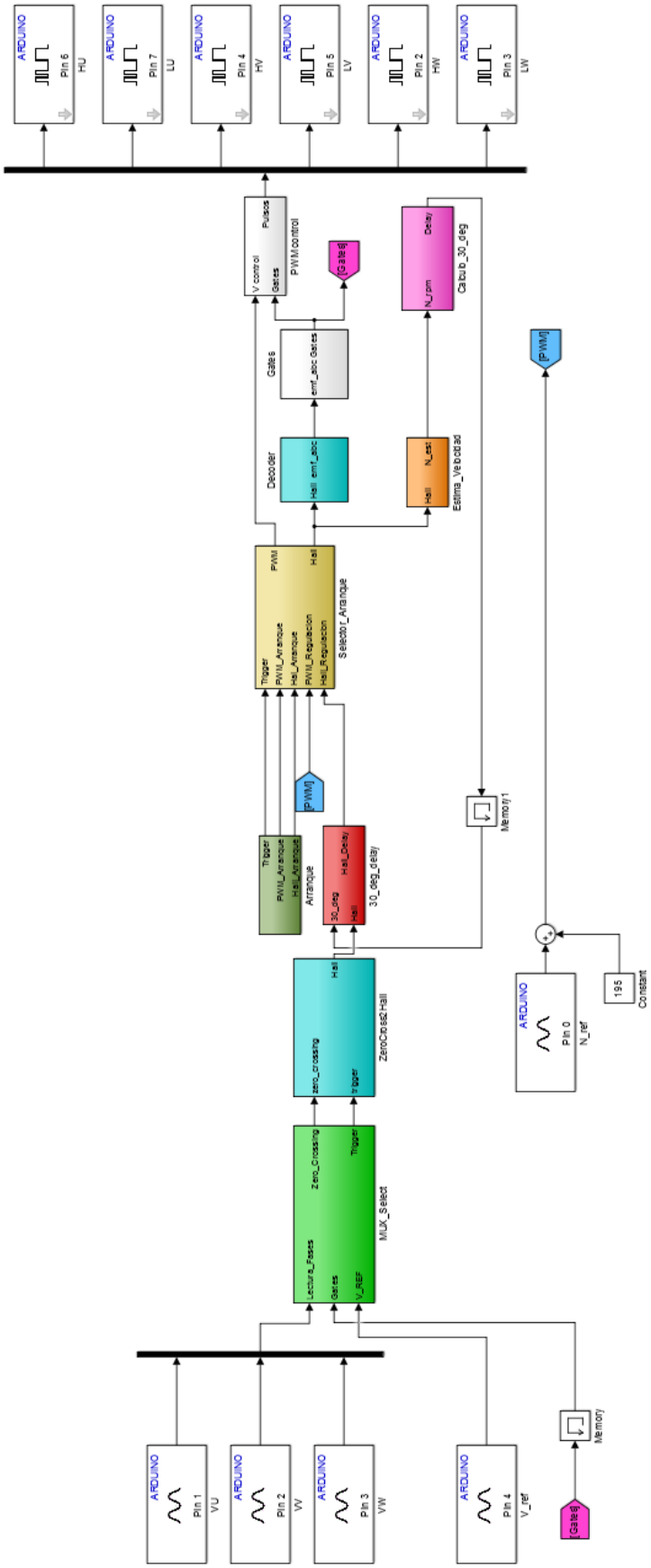

Figura 4.46 – Programa desarrollado para el control sin sensores con el Arduino Mega 2560

Al conectar el Arduino Mega 2560 al ordenador, y transferir el programa de la figura 4.46, Simulink permite compilar en modo *MultiTasking*, permitiendo así realizar cálculos y operaciones en paralelo, reduciendo la carga computacional y el peso a ocupar en la memoria de datos del microcontrolador. Pese a que los datos ocupan alrededor del 95% de la memoria del Arduino Mega 2560, en este caso sí es posible cargar el programa en el dispositivo para su ejecución.

4.5.2.3. Resultados de las pruebas

La primera prueba se ha realizado con el PWM con un tiempo de ciclo del 100%, porque, tal y como hemos analizado en la teoría, representa una mayor magnitud para la fuerza contraelectromotriz, por lo que su detección para el control *sensorless* será más sencilla en este supuesto. Para ello, se ha llevado el potenciómetro R19 a su límite.

Al conectar el sistema, se ha realizado la calibración fina del sistema mediante el giro del potenciómetro R21, ajustando lo máximo posible la detección de paso por cero. En la posición en la que mejor funcionamiento ofrecía el sistema, se detectó que el motor giraba a velocidad aparentemente constante, pero presentando una excesiva vibración provocado por un giro algo atrancado, con un efecto de saltos demasiado marcados entre las fases del motor BLDC.

Analizando la tensión de una de las fases mediante el osciloscopio, obtenemos la gráfica de la figura 4.47

Figura 4.47 – Tensión de una fase con el 100% del tiempo de ciclo

En la pantalla del osciloscopio podemos observar de manera bien marcada los periodos en los que la fase se encuentra conduciendo en directa y en inversa. En los tramos intermedios, observamos que la forma de la fuerza contraelectromotriz no se corresponde demasiado bien con las formas esperadas según la teoría, sino que aparece una forma cóncava o convexa no esperada, que afecta a la detección del paso por cero, lo que indudablemente tiene como consecuencia un excesivo atrancamiento en el giro del motor.

Las pruebas subsiguientes son reduciendo el tiempo de ciclo del PWM, mediante el giro del potenciómetro R19, manteniendo el R21 en la misma posición siempre, ya que fue calibrada para un 100% de tiempo de ciclo, observando que la respuesta del motor va empeorando conforme se hace más pequeño el tiempo que los MOSFET se encuentran en conducción.

Se llega al punto de que, si descendemos desde un cierto nivel de tiempo de ciclo del PWM, el motor llega a pararse e incluso a cambiar de sentido, un efecto no previsto en el presente proyecto.

En la figura 4.48 podemos ver la tensión de una de las fases para distintos niveles del PWM.

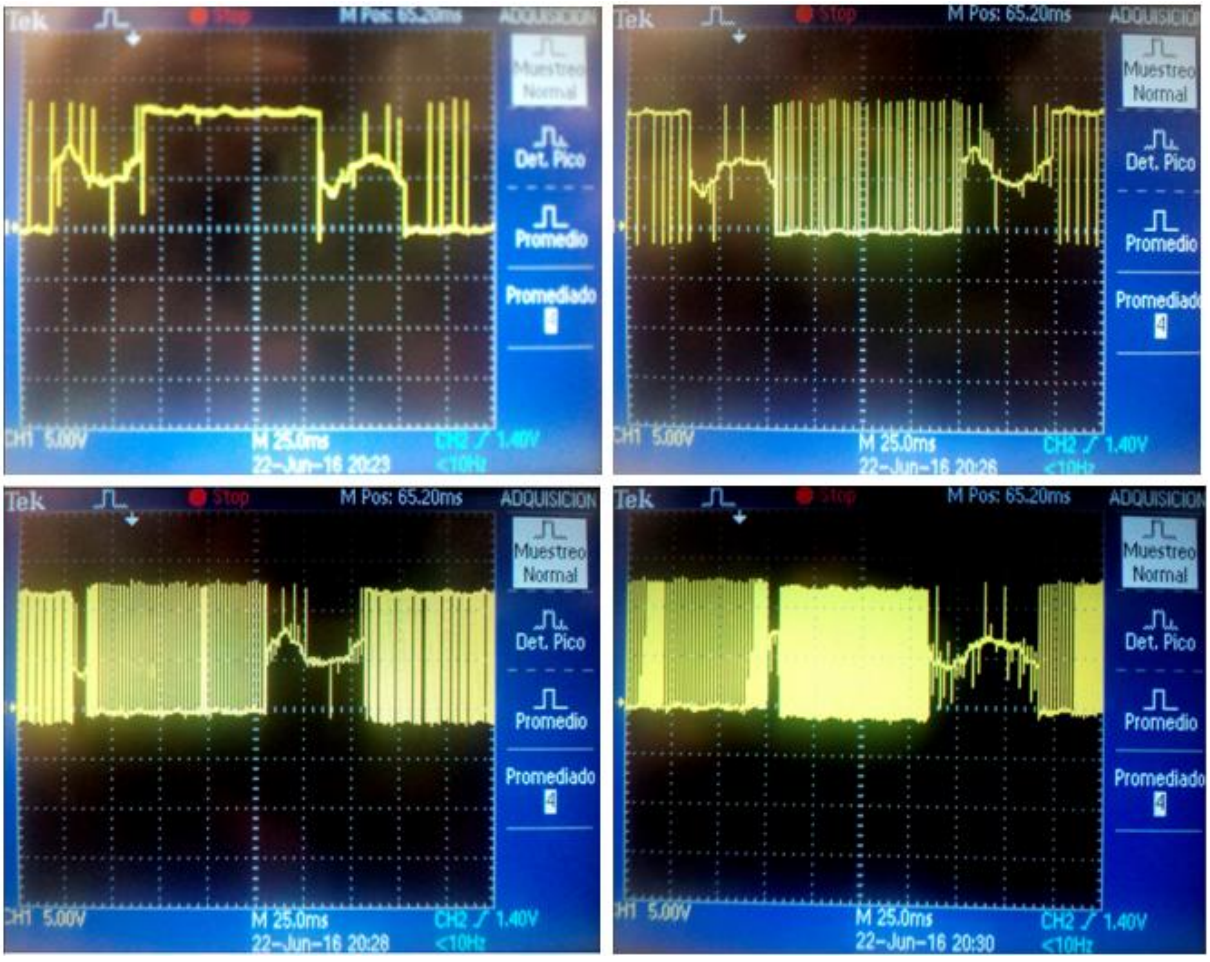

Figura 4.48 – Tensión de una fase para distintos tiempos de ciclo del PWM

4.5.2.4. Análisis y diagnóstico de limitaciones detectadas

Una vez realizadas las pruebas y habiendo realizado una recogida de datos, llega el momento de analizar los problemas y limitaciones que han surgido durante las pruebas, y tratar de diagnosticar sus causas y proponer posibles soluciones.

En primer lugar, si analizamos la manera en la que detecta la tensión de las fases la tarjeta inversora, nos damos cuenta de que, en una alimentación de motor de 12V como la que tenemos en el presente proyecto, las tensiones VU, VV y VW que se miden con el Arduino, tal y como está diseñado el divisor de tensión, oscilarán entre 2,10V y 0V, mientras que la tensión de referencia, en función de la posición que tenga el potenciómetro R21 entre 0,77V y 2,31V.

Estos niveles de tensión, evidentemente están preparados para que puedan ser adquiridos a través de la entrada analógica de un PIC, cuyo límite de tensión es de 3,3V, sin dañar el componente. No obstante, en el caso del presente proyecto, en el que el Arduino Mega 2560 es capaz de leer señales en un rango de 0V a 5V estamos perdiendo más de la mitad de la resolución debido a los divisores de tensión del inversor.

Para este problema, una solución sería redimensionar la adquisición de la señal mediante la aplicación de un divisor de tensión externo a la placa inversora en lugar de utilizar el que viene en ella integrado, en el que se pueda aprovechar la totalidad, o al menos la mayor parte, de la resolución del convertidor analógico/digital del Arduino Mega.

No obstante, existe una solución aún mejor, ya que al aplicar un divisor de tensión cualquiera se está realizando una atenuación de la señal, que reduce la sensibilidad a la hora de detectar el paso por cero, en especial a velocidades bajas donde la magnitud de las tensiones a detectar es menor. En lugar de esto, la aplicación de diodos zener para limitar la tensión máxima a los 5V que admite el microcontrolador es una mejor opción, porque trunca cualquier tensión que sobrepase los 5V pero mantiene intacta la sensibilidad en el entorno del cruce por cero.

Una aplicación incluso todavía mejor respecto a el empleo de diodos zener es utilizar circuitos amplificadores que, alimentados a 5V para que saturen a tensiones superiores a ésta, permitan incrementar la sensibilidad de la señal en el paso por cero, sin comprometer la integridad del convertidor analógico/digital.

Un segundo problema detectado, que está relacionado en parte con el anterior, es el del exceso de ruido que presenta la señal de la fuerza contraelectromotriz. Este problema se maximiza a velocidades de giro bajas, que corresponden con menores tensiones de fuerza contraelectromotriz, porque a estas tensiones el ruido representa un mayor error porcentual respecto a la señal real, incrementando así la sensibilidad del sistema al ruido. Esto daría una explicación al hecho de que el comportamiento del motor empeore conforme bajamos el tiempo de ciclo del PWM.

La solución a este problema sería introducir algún tipo de circuito con histéresis, como una báscula de Schmitt que permita eliminar el ruido de la señal real, mejorando así la detección del paso por cero y correspondientemente el comportamiento del motor.

Por último, nos encontramos con que, en la ejecución del programa desarrollado para el control *sensorless*, se está llevando al límite al Arduino Mega 2560, en lo que se refiere a tiempos de muestreo de los A/D (10 kHz), la memoria de datos (alrededor del 95% de ocupación), una gran carga de computación en el proceso de estimar la velocidad de giro y retardar la señal, etc.

Esto provoca, en primer lugar, que puedan aparecer errores en la estimación de la velocidad, o que ésta no se realice lo suficientemente rápido para que la conmutación se realice en el instante correcto, de ahí que se noten los saltos entre bobinas. Por otro lado, queda poco margen de maniobra para introducir un controlador PID que regule la velocidad en tiempo real.

La solución es utilizar otro tipo de dispositivos de una mayor capacidad de procesamiento como PIC32, dsPIC o Arduino Due (que lleva un ARM Cortex-M3 de 32 bits como cerebro).

5. Conclusiones y futuros desarrollos

5.1. Introducción

En el presente capítulo de la memoria, se van a presentar las conclusiones a las que se ha llegado tras todo el trabajo realizado en el presente proyecto, además de proponer algunas líneas de trabajo que podrían seguir algunos futuros desarrollos que partan del presente proyecto.

Previamente a las conclusiones y los futuros desarrollos, se va a presentar además un breve resumen del proyecto desarrollado.

El objetivo principal de este proyecto era diseñar un sistema electrónico en el que se pueda controlar la velocidad de un motor BLDC mediante la utilización de un algoritmo de control en lazo cerrado. Otro punto muy importante del proyecto es el de aplicar las herramientas de programación gráfica mediante bloques que ofrece MATLAB/Simulink en un proyecto real.

Para ello, inicialmente se ha realizado un estudio teórico completo del motor BLDC, analizando en primer lugar los principios constructivos de este tipo de motores, pasando a continuación a explicar su modo de funcionamiento, y refiriéndonos por último a las dos técnicas de control disponibles: mediante el uso de sensores Hall y sin sensores.

A partir de este estudio, se ha procedido a modelar en Simulink los sistemas planteados, realizando la simulación con diferentes parámetros y obteniendo resultados satisfactorios.

A continuación, se ha procedido a analizar el hardware a emplear en la implementación física de los modelos desarrollados, analizando alternativas para el inversor y el microcontrolador.

Por último, se han realizado dos programas para Arduino utilizando bloques de Simulink, uno para control con sensores Hall y otro para control sin sensores, y se han implementado físicamente, realizando análisis con la ayuda del osciloscopio y obteniendo las limitaciones del sistema como conclusiones de todo el trabajo realizado.

5.2. Conclusiones

La experiencia acumulada durante el desarrollo del presente trabajo, así como los resultados obtenidos en el mismo, permiten llegar a las conclusiones generales a continuación expuestas:

- Los motores BLDC presentan el suficiente número de ventajas como para convertirse en un estándar de la movilidad eléctrica en un futuro a corto plazo. En particular, su bajo desgaste y ruido, unidos a su mejor relación potencia/peso respecto a los motores eléctricos tradicionales hacen de este tipo de motores la

mejora alternativa de las disponibles hoy en día para la propulsión eléctrica de vehículos de todo tipo.

- Las interfaces gráficas de desarrollo de código mediante bloques representan un gran avance respecto a las interfaces de programación tradicionales, porque permite establecer relaciones entre variables de una manera mucho más intuitiva, se minimizan los errores en la elaboración del código y, como en el caso del presente proyecto, si se parte de un modelo del sistema a programar la elaboración del código se simplifica en gran medida. Como punto a mejorar de este tipo de herramientas, aún no han alcanzado el grado de personalización que permite la creación de código tradicional.
- Cuando se realiza control *sensorless* en un motor BLDC, un factor clave que se debe tener en cuenta de manera prioritaria es el dimensionamiento de los circuitos de adquisición de la tensión de las fases, que debe aprovechar al máximo la resolución de los convertidores A/D, imprescindible para conseguir la máxima precisión posible en la detección del cruce por cero y no tener retardos que generen que el motor se atranque.
- Arduino, ya sea en la versión Uno o en la más potente Mega 2560, presenta una cierta limitación para el control *sensorless* de motores BLDC, debido a la necesidad de la realización de cálculos complejos en tiempo real y, sobre todo, conversiones de señales analógicas a digitales a una frecuencia muy alta para que no aparezcan errores en la detección del paso por cero de la fuerza contraelectromotriz. No obstante, la reducción de costes de los microcontroladores en general, y en particular de los procesadores de tipo DSP, hace posible plantear que los dispositivos de bajo coste tengan futuro como controladores de motores BLDC.

5.3. Futuros desarrollos

Se presentan tres líneas de trabajo a continuación, que tratarán de hacer servir el presente proyecto como base a otros que persigan objetivos más ambiciosos.

- Una primera línea de trabajo consistiría en trabajar en superar las limitaciones detectadas en la implementación física del capítulo 4. Se podría plantear el empleo de otro microcontrolador con una mayor capacidad de procesamiento y adquisición de datos (por ejemplo tipo dsPic), con un disparador de Schmitt para evitar el ruido y un dimensionamiento de la fuerza contraelectromotriz basado en el empleo de amplificadores que no reduzcan la sensibilidad de la señal a detectar. Además, se podría realizar el diseño de una placa para integrar todas estas mejoras, incrementando la robustez del sistema.
- Una segunda línea consistiría en llevar a cabo alguna de las mejoras del algoritmo de control *sensorless* que se proponen en el apartado 2.4.7, realizando el proceso completo de desarrollo teórico, modelado, simulación e implementación física. De esta forma, se podría adaptar el desarrollo propuesto en el presente proyecto para aplicaciones concretas que requieran de métodos de control más avanzados.

- Como tercera línea de trabajo, se propone la aplicación de otro desarrollo interesante podría ser el de desarrollar un sistema de frenado regenerativo basado en el sistema de control *sensorless*, que permita recargar la batería o generar electricidad en aplicaciones que así lo permitan mediante la energía cinética del giro del rotor del motor.

Índice de figuras

Bibliografía

[1] DHANASEKARAN, Surender; BHEEMAIAH, Nandineravanda. *Intelligent control methods for sensorless control of BLDC motor drive used in high speed applications.* International Education & Research Journal [IERJ], Vol.1 – Issue 4, Nov 2015

[2] *El futuro del coche eléctrico, analizado desde la universidad.* I Cumbre Universitaria del Vehículo Eléctrico, Universidad Carlos III de Madrid

[3] ERDMAN, David M. *Control system, method of operating an electronically commutated motor and laundering apparatus.* US Patent No. 4654566, General Electric Company, 1987

[4] FULLEA, Jose; TRINIDAD, Francisco; AMASORRAIN, J. Carlos; SANZBERRO, Mikel. *El vehículo eléctrico: tecnología, desarrollo y perspectiva.* McGraw-Hill Interamericana, 1997

[5] GAMAZO-REAL, J. Carlos; VÁZQUEZ-SÁNCHEZ, Ernesto; GÓMEZ-GIL, Jaime. *Position and Speed Control of Brushless DC Motors Using Sensorless Techniques and Application Trends.* Universidad de Valladolid, 2010

[6] GARCÍA, J.Miguel. *Desarrollo de un controlador para motores DC brushless basado en CompactRIO y LabVIEW de National Instruments para el studio de nuevos algoritmos de control.* Universidad Carlos III de Madrid, Leganés, 2011

[7] GRASBLUM, Pavel. *3-Phase BLDC Motor Sensorless Control using MC9S08AW60 Designer Reference Manual, Rev. 0.1*. Freescale Czech Systems Center, República Checa, 2007

[8] GURUNATHAN, C.; MURUGAN, M.; JEYABHARATH, R. *A back EMF detection method is implemented for sensorless BLDC motor.* K.S.Rangasamy College of Technology, India. IJAICT, Vol.1 – Issue 11, Mar 2015

[9] HERRERO, L. Carlos, VÁZQUEZ, J. Antonio, RUIZ, J. Miguel. Apuntes *Electrónica Industrial*. Universidad de Valladolid

[10] KRISHNAN, R.; GHOSH, R. *Starting algorithm and performance of a PM DC brushless motor drive system with no position sensor.* Power Electronics Specialists Conference, 1989

[11] LÁJARA, J. Rafael. *Sistemas Integrados con Arduino.* Marcombo Ediciones Técnicas, Madrid, 2013

[12] *Mapa tecnológico de la Movilidad Eléctrica*. Observatorio Tecnológico de la Energía, 2012

[13] MARTÍNEZ-BROCAL, Luis. *Controlador de Motor Brushless DC para Arduino*. Universidad Pontificia de Comillas, Madrid, 2014

[14] MOSCAT, Alejandro. *Desarrollo de un controlador de presión de combustible basado en bombas trifásicas sin escobillas.* Universidad Politécnica de Madrid, 2014

[15] NAGADEVEN, A.; KARUNAKARAN, L. *Implementation of DSP based Sensorless Control with Direct Back-EMF Detection Method for BLDC Motor*. India, 2008

[16] RODRÍGUEZ, J. Carlos. *Control de velocidad de motores BLDC mediante sistemas basados en Arduino para aplicaciones en vehículos eléctricos.* Universidad de Valladolid, 2015

[17] ROELANTS, Stef. *Light regulation with a PID simulink and an Arduino.* Escuela de Ingenierías Industriales, Universidad de Valladolid, 2012

[18] SHAO, Jianwen. *Direct Back EMF Detection Method for Sensorless Brushless DC (BLDC) Motor Drives.* Virginia Polytechnic Institute, 2003

[19] SHEEL, Soumya; KUMAR, Vinay. *Optimized Design of the BLDC Motor for Higher Efficiency*. International Journal of Engineering Research & Technology, Vol.2 – Issue 5, May 2013

[20] TARA KALYANI, S.; SYFULLAH KHAN, Md. *Simulation of sensorless operation of BLDC motor based on the zero-cross detection form the line voltage.* International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering [IJAREEIE], Vol.2 – Issue 12, Dec 2013

[21] TORRENTE, Óscar. *El mundo GENUINO-ARDUINO. Curso práctico de formación.* RC Libros, Madrid, 2016

[22] UZUKA, K.; UZUHASHI, H. *Microcomputer Control for Sensorless Brushless Motor.* IEEE Trans. Industry Application, vol. IA-21, 1985

[23] VOGEL, Carl. *Build Your Own Electric Motorcycle.* McGraw-Hill, 2009

[24] WHEAT, Dale. *Arduino Internals*. Apress, 2011

[25] YEDAMALE, Padmaraja. *Brushless DC (BLDC) Motor Fundamentals.* Microchip Technology Inc., 2003

[26] ZACHARIA, Geethu; RAINA, Annai. *A survey on Back EMF Sensing Methods for Sensorless Brushless DC Motor Drives.* International Journal of Emerging Trends in Engineering Research, Vol.2 – No. 2, Feb 2014