



**UNIVERSIDAD DEL BÍO-BÍO**

FACULTAD DE INGENIERÍA  
DEPTO. INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

**“Implementación de un Controlador FOC para Motores  
Brushless con Encoder Utilizando STM32”**

AUTOR:  
RODRIGO FUENTES PEDREROS

SEMINARIO PARA OPTAR AL TÍTULO DE  
INGENIERO DE EJECUCIÓN EN ELECTRÓNICA

CONCEPCIÓN - CHILE  
AÑO 2024



**UNIVERSIDAD DEL BÍO-BÍO**

FACULTAD DE INGENIERÍA  
DEPTO. INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

**“Implementación de un Controlador FOC para Motores  
Brushless con Encoder Utilizando STM32”**

AUTOR  
RODRIGO FUENTES PEDREROS

PROFESOR GUÍA:  
ANGEL ERNESTO RUBIO

PROFESORES GUÍA ADJUNTO:  
PEDRO MELIN COLOMA

# Índice

<b>Resumen . . . . .</b>	<b>7</b>
<b>1 Introducción . . . . .</b>	<b>8</b>
1.1 Introducción general . . . . .	8
1.2 Marco teórico . . . . .	9
1.2.1 Motores <i>brushless</i> . . . . .	9
1.2.2 Control Orientado de Campo (FOC) . . . . .	10
1.2.3 Transformada de Clarke . . . . .	11
1.2.4 Transformada de Park . . . . .	11
1.2.5 Transformada inversa de Park . . . . .	12
1.2.6 Modulación de Espacio Vectorial (SVM) . . . . .	12
1.3 Motivación . . . . .	13
1.4 Objetivos . . . . .	13
1.4.1 Objetivo General . . . . .	13
1.4.2 Objetivos Específicos . . . . .	13
1.4.3 Alcances . . . . .	14
<b>2 Hardware . . . . .</b>	<b>15</b>
2.1 Introducción . . . . .	15
2.2 Desarrollo del Hardware . . . . .	16
2.2.1 Motor . . . . .	17
2.2.2 Consideraciones para el Controlador Basadas en el Motor . . . . .	18
2.2.3 Encoder . . . . .	19
2.2.4 Conjunto motor-encoder . . . . .	20
2.2.5 Puente Mosfet . . . . .	21
2.2.6 Controladores de Puerta . . . . .	22
2.2.7 Sensores de Corriente . . . . .	24
2.3 Implementación del puente MOSFET . . . . .	25
2.4 Observaciones y posibles mejoras . . . . .	27
<b>3 Firmware . . . . .</b>	<b>28</b>
3.1 Introducción . . . . .	28
3.2 Microcontrolador y periféricos . . . . .	28

3.2.1	Configuración de Timers . . . . .	29
3.2.2	Configuración de los ADC y su integración con Timers y DMA . . . . .	31
3.3	Estructura del Firmware . . . . .	32
3.3.1	Ejecución en Main . . . . .	32
3.3.2	Ejecución en Interrupción . . . . .	33
3.4	Observaciones y posibles mejoras . . . . .	34
<b>4</b>	<b>Validación . . . . .</b>	<b>35</b>
4.1	Validación de la adquisición y transformación de las mediciones de corriente . . . . .	36
4.1.1	Validación de las mediciones de corriente . . . . .	36
4.1.2	Validación de la transformada de Clarke . . . . .	37
4.1.3	Validación de la transformada de Park . . . . .	38
4.2	Validación de los Controladores PI . . . . .	39
4.2.1	Validación del controlador de velocidad . . . . .	39
4.2.2	Validación del controlador de corriente . . . . .	40
4.3	Validación señales del SVM . . . . .	41
<b>Comentarios y Conclusiones . . . . .</b>		<b>42</b>
<b>Trabajos Futuros . . . . .</b>		<b>43</b>
<b>Bibliografía . . . . .</b>		<b>44</b>

# Índice de figuras

1.1	Esquema de motores sin escobillas ( <i>BLDC</i> ) . . . . .	9
1.2	Diagrama de flujo del control FOC[7] . . . . .	10
1.3	Esquema transformada de Clarke. . . . .	11
1.4	Esquema transformada de Park. . . . .	11
1.5	Esquema transformada inversa de Park. . . . .	12
1.6	Sextantes en SVM. . . . .	12
2.1	Diagrama Resumido del sistema. . . . .	15
2.2	Diagrama resumido del hardware. . . . .	16
2.3	SunnySky X3120 V3. . . . .	17
2.4	AS5047P. . . . .	19
2.5	Montura motor-encoder. . . . .	20
2.6	Socket Encoder. . . . .	20
2.7	Socket imán axial. . . . .	20
2.8	Esquema puente MOSFET. . . . .	21
2.9	Esquema UCC27211. . . . .	22
2.10	Esquema de control PWM con las compuertas AND. . . . .	23
2.11	Esquema INA240. . . . .	24
2.12	Esquemático de la pierna C del inversor. . . . .	25
2.13	Pierna C del inversor. . . . .	25
2.14	Alimentación del Puente MOSFET. . . . .	26
3.1	Bosquejo de la estructura general del firmware en el microcontrolador. . . . .	32
3.2	Diagrama de flujo de las tareas principales en la línea de ejecución <code>main()</code> . . . . .	33
3.3	Diagrama de flujo de las tareas principales en la línea de ejecución de la interrupción. . . . .	33
4.1	Diagrama de flujo adquisición de datos. . . . .	35
4.2	Corrientes ideales en un sistema trifásico equilibrado. . . . .	36
4.3	Corrientes medidas en el sistema trifásico. . . . .	36
4.4	Corrientes ideales en el plano $\alpha\beta$ . . . . .	37
4.5	Corrientes medidas en el plano $\alpha\beta$ . . . . .	37
4.6	Corrientes ideales en el plano $dq$ . . . . .	38
4.7	Corrientes medidas en el plano $dq$ . . . . .	38
4.8	Velocidad medida por el encoder y setpoint de velocidad. . . . .	39

4.9	Corrientes medidas en el plano $dq$ .	40
4.10	Voltajes en el plano $dq$ .	40
4.11	Señales del timer.	41

# Índice de cuadros

2.1	Características del motor SunnySky X3120 V3 585Kv	17
2.2	Parámetros Requeridos para el Controlador Basados en el Motor SunnySky X3120 V3	18
2.3	Características del encoder AS5047P	19
2.4	Características del MOSFET BSC011N03LSI.	21
2.5	Tabla de Lógica del UCC27211	22
2.6	Tabla de Lógica del UCC27211 con AND	23
3.1	Configuración del Timer 1 en CubeMX.	29
3.2	Configuración del Timer 2 en CubeMX.	30

# Resumen

Este trabajo aborda el diseño e implementación de un controlador de campo orientado (*FOC*) para motores *brushless* de corriente continua, incorporando un *encoder* para la medición de la posición. En primera instancia, se revisan los fundamentos teóricos que sustentan la técnica *FOC*, destacando su capacidad para desacoplar el flujo magnético del par motor, y la relevancia de las transformadas de Clarke y Park para simplificar el control de las corrientes en el estator. Asimismo, se incluyen los conceptos principales de la Modulación por Vector Espacial (*SVM*), esencial para la síntesis de las señales de salida del inversor.

Posteriormente, se detalla el diseño y la fabricación del *hardware* de la solución, que comprende un inversor trifásico basado en MOSFETs, controladores de puerta (*gate drivers*) y sensores de corriente de tipo *shunt*, junto con los elementos de filtrado y protección necesarios. Se selecciona y acondiciona un motor *brushless*, al cual se integra un *encoder* magnético para la obtención de la posición del rotor. Además, se documentan los procesos de elaboración de la PCB y la disposición de sus pistas para manejar corrientes elevadas, considerando la disipación y la estabilidad del sistema.

En el ámbito del *firmware*, se describen las estrategias de configuración de los periféricos del microcontrolador STM32, haciendo uso de *STM32CubeMX* para la inicialización de temporizadores y convertidores A/D (*ADC*). Se explica el flujo de ejecución principal y la rutina de interrupción encargada de efectuar la lectura de sensores, la aplicación de las transformadas de Clarke y Park, los controladores de corriente y velocidad, así como la generación de las señales PWM mediante *SVM*.

Para validar el rendimiento del controlador, se efectúan pruebas de velocidad y par bajo distintas condiciones de carga, analizando la estabilidad de las corrientes y la rapidez de respuesta ante cambios en la referencia. Asimismo, se capturan datos en tiempo real para su posterior evaluación, confirmando que la implementación logra un control estable y preciso del motor. Se identifican, no obstante, algunos retos en escenarios extremos, motivando la propuesta de futuras mejoras en el diseño.

En síntesis, la solución presentada demuestra la factibilidad de implementar un controlador *FOC* de altas prestaciones utilizando un STM32 de la serie H7, combinando técnicas de transformación en el dominio rotatorio y *SVM* en el lazo de control. Las observaciones obtenidas abren la puerta a nuevas optimizaciones, tanto en el *hardware* (filtrado, disipación y protección) como en el *firmware* (estructuras de software, algoritmos de compensación y sistemas operativos en tiempo real), facilitando así el despliegue de aplicaciones en robótica competitiva y sistemas embebidos.

## Capítulo 1

# Introducción

## 1.1. Introducción general

Cuando se habla de motores eléctricos de corriente continua, los motores sin escobillas destacan por su alto desempeño, siendo la opción preferida en aplicaciones como vehículos eléctricos, drones, robótica avanzada y robótica industrial.

Para los motores *brushless*, existe una gran variedad de técnicas de control. Algunas de las más relevantes son el control trapezoidal, utilizado mayormente en drones; el control directo de torque (DTC), empleado principalmente en motores de media y alta potencia en ambientes industriales; y el control FOC, que se aplica mayormente en robótica. En este documento, se tratará únicamente esta última técnica.

Existen tres aspectos clave que diferencian cada técnica: el rizado de torque, el coste computacional y la complejidad del hardware necesario para su ejecución. En estos aspectos, el control FOC es uno de los que produce menor rizado de torque, aunque requiere un mayor coste computacional y hardware especializado.

Este proyecto presenta la implementación y validación de un controlador de campo orientado (FOC) para motores sin escobillas de corriente continua (BLDC). Además, se desarrolló la placa controladora utilizando un microcontrolador STM32 como núcleo principal.

## 1.2. Marco teórico

### 1.2.1. Motores *brushless*

Los motores *brushless*, o motores sin escobillas de corriente continua (BLDC), son un tipo de motor trifásico síncrono que utiliza imanes permanentes en el rotor y bobinas en el estator, el cual está formado por láminas de hierro con ranuras para alojar las bobinas. A diferencia de los motores de corriente continua con escobillas, los BLDC no emplean elementos mecánicos para comutar la corriente en las bobinas. En su lugar, esta comutación se realiza mediante un controlador electrónico [1], eliminando el rozamiento y el desgaste asociados a las escobillas, lo que mejora la eficiencia del sistema.

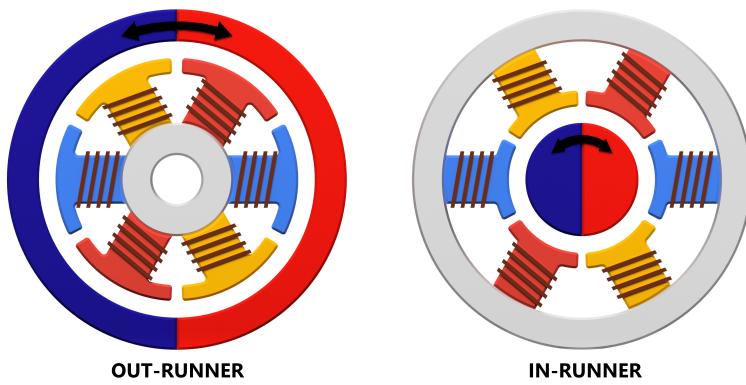


Figura 1.1: Esquema de motores sin escobillas (*BLDC*).

Similar a los motores trifásicos de corriente alterna, estos pueden tener configuraciones de bobinado en delta o estrella [2]. Además, como se ejemplifica en la figura 1.1, pueden presentar configuraciones *in-runner*, donde los imanes están en el centro del eje con las bobinas en el exterior, o configuraciones *out-runner*, donde los imanes se encuentran por fuera del motor, mientras que las bobinas están en el centro [3]. Estos motores, al requerir una sincronización entre el campo magnético del estator y el campo magnético del rotor, suelen utilizar encoders o sensores de efecto Hall para obtener información sobre la posición del eje y, de este modo, mantener una comutación adecuada. No obstante, también es posible emplear técnicas *sensorless* (sin sensor) para estimar la posición del rotor sin necesidad de sensores físicos [4].

Los *BLDC* se destacan por ofrecer una mayor densidad de potencia, mayor torque, mejores eficiencias y pueden alcanzar velocidades más altas. Sin embargo, al requerir de un controlador electrónico que se encargue de la comutación para generar el movimiento del rotor, la complejidad y los costos asociados a su implementación son mayores [5].

### 1.2.2. Control Orientado de Campo (FOC)

El control orientado de campo se basa en desacoplar el flujo electromagnético del par motor, permitiendo el control independiente de cada uno, aunque en este caso se realiza de forma indirecta, trabajando con las corrientes del estator en vez de las variables reales de flujo y par. Este desacoplamiento se logra mediante la transformación de las variables trifásicas del motor, que están en un marco de referencia estacionario  $ABC$ , hacia un marco de referencia rotatorio  $dq$  que gira en sincronía con el rotor del motor [6].

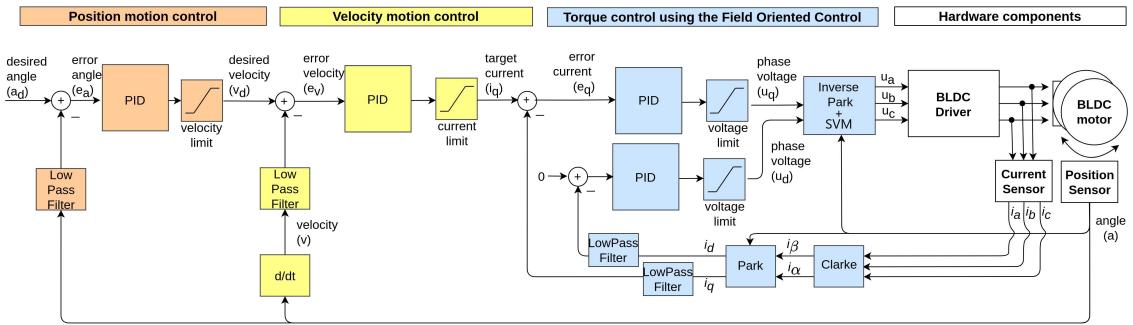


Figura 1.2: Diagrama de flujo del control FOC[7].

La figura 1.2 presenta el diagrama de flujo base de un controlador FOC, con sus elementos esenciales, donde podemos apreciar la separación por hardware, controladores de corriente, además de los controladores de velocidad y posición. La transformación entre los marcos de referencia se realiza aplicando la transformada de Clarke y la transformada de Park. De esta forma, las mediciones de corriente que presentan un comportamiento oscilatorio en el tiempo se convierten en variables de corriente continua, lo que virtualmente emula el funcionamiento de un motor con escobillas a nivel de los controladores[6]. Posteriormente, se aplican los controladores de corriente y velocidad para obtener las señales de control necesarias para el inversor trifásico.

### 1.2.3. Transformada de Clarke

La transformada de Clarke convierte un sistema trifásico de corrientes  $ABC$  en un sistema bifásico  $\alpha\beta$ , proyectando las corrientes en un sistema de coordenadas bidimensional estacionario. Esta transformación simplifica el análisis al reducir las tres corrientes de fase a dos componentes ortogonales, manteniendo la información esencial del sistema original [8].

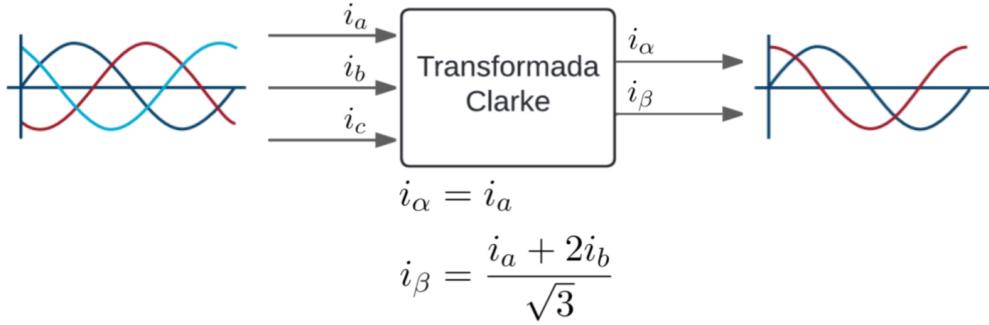


Figura 1.3: Esquema transformada de Clarke.

En la figura 1.3 se ilustra la representación gráfica de las corrientes de entrada  $i_a, i_b, i_c$  y las corrientes de salida  $i_\alpha, i_\beta$ . Además, se incluyen las ecuaciones necesarias para llevar a cabo la transformación.

### 1.2.4. Transformada de Park

La transformada de Park convierte el sistema en el marco de referencia estacionario  $\alpha\beta$  en el sistema en el marco de referencia rotatorio  $dq$  sincronizado con el ángulo del rotor[8].

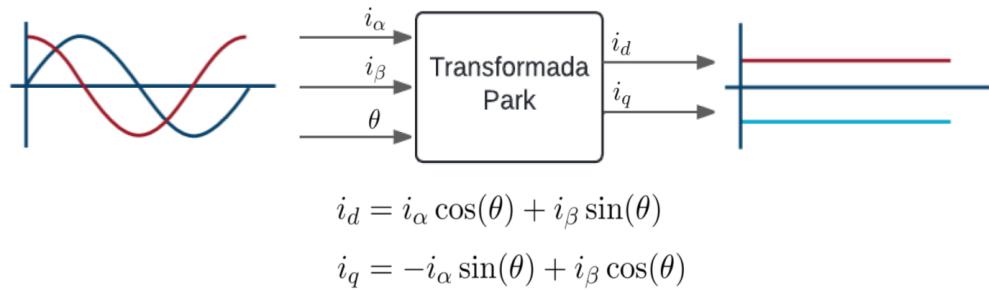


Figura 1.4: Esquema transformada de Park.

En la figura 1.4 se muestra gráficamente cómo las variables  $\alpha, \beta$  se transforman a  $d, q$ , junto con las ecuaciones empleadas para llevar a cabo esta transformación.

### 1.2.5. Transformada inversa de Park

La transformada inversa de Park convierte el sistema en el marco de referencia rotatorio  $dq$  de vuelta al marco de referencia estacionario  $\alpha\beta$  [8].

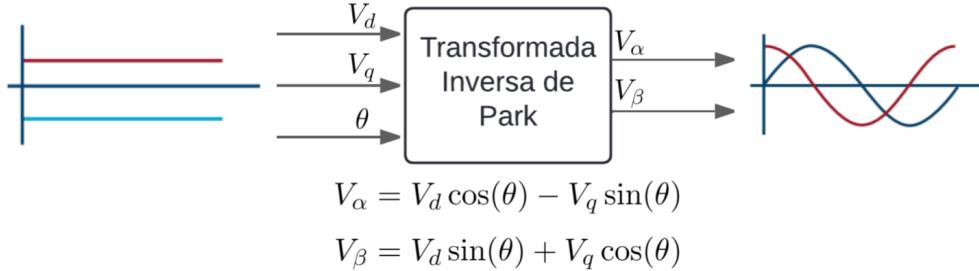


Figura 1.5: Esquema transformada inversa de Park.

En la figura 1.5 se ilustra cómo las variables  $d, q$  se transforman nuevamente a  $\alpha, \beta$ , con las ecuaciones correspondientes a este proceso inverso.

### 1.2.6. Modulación de Espacio Vectorial (SVM)

La Modulación por Vector Espacial (SVM) es una técnica utilizada para el control digital de inversores de voltaje. Representa todos los estados de conmutación del inversor como vectores de voltaje en el plano  $\alpha\beta$ , formando un hexágono regular dividido en seis sectores o sextantes, como se puede ver en la figura 1.6. Estos se utilizan para sintetizar el vector de referencia  $\vec{V}_{ref} = (V_\alpha, V_\beta)$  mediante una suma ponderada de los vectores de voltaje adyacentes en el sextante donde se encuentre, para finalmente calcular los tiempos de activación necesarios en cada fase del inversor, logrando una aproximación más precisa de la señal deseada.

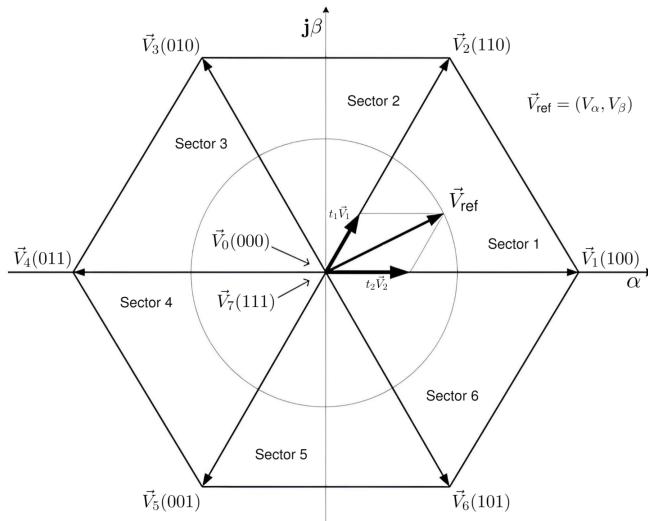


Figura 1.6: Sextantes en SVM.

## **1.3. Motivación**

Este proyecto nace de la necesidad de un controlador para motores brushless adecuado para su uso en robots de competencia en la categoría de robot sumo autónomo. En esta categoría, este tipo de motores no son normalmente utilizados debido a las limitaciones de los controladores comerciales y los riesgos que conlleva la sobrecarga de los motores en periodos cortos de acción, ya que los controladores comerciales no están pensados para esto.

## **1.4. Objetivos**

### **1.4.1. Objetivo General**

Implementar un controlador de tipo FOC (Control de Campo Orientado) para motores brushless con encoder, utilizando un microcontrolador STM32, que sirva de base para un driver especializado en la robótica competitiva.

### **1.4.2. Objetivos Específicos**

- Estudiar los principios del Control de Campo Orientado (FOC) y la modulación de espacio vectorial (SVM) para aplicarlos en el diseño del controlador.
- Diseñar el hardware para el controlador FOC, con los componentes mínimos necesarios para validar el funcionamiento.
- Configurar y programar el microcontrolador para el algoritmo FOC, utilizando las librerías HAL de STM32.
- Validar el funcionamiento del controlador y proponer posibles mejoras para su aplicación en robótica competitiva.

### **1.4.3. Alcances**

- Se configurará el microcontrolador utilizando STM32CubeMX.
- Se implementará el controlador de velocidad y corriente.
- No se implementará el control de posición.
- Se programará el firmware utilizando VScode y el lenguaje C.
- El firmware se compilará utilizando un makefile y GCC, junto con la extensión STM32 for VScode.
- Se cargará el firmware utilizando STM32CubeProgrammer y ST-link.
- Se diseñará la PCB en Eagle.
- Se diseñarán las piezas adicionales en Autodesk Inventor para su posterior fabricación mediante impresión 3D.
- La PCB se fabricará y semi ensamblará utilizando JLCPCB.
- Se obtendrán datos detallados durante el tiempo de ejecución.
- Los datos se graficarán utilizando Plotly en Python.
- Se identificarán posibles mejoras.

## Capítulo 2

# Hardware

### 2.1. Introducción

Este capítulo busca documentar el desarrollo e implementación de todo el hardware y electrónica necesaria para el controlador FOC, partiendo de la revisión de los parámetros principales y componentes utilizados, para finalmente revisar a grandes rasgos el diseño de la placa controladora.

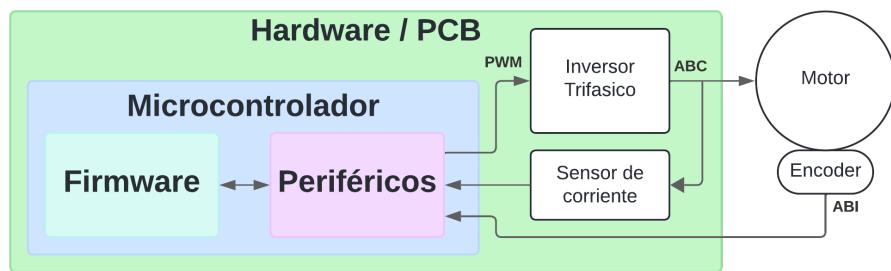


Figura 2.1: Diagrama Resumido del sistema.

En la Figura 2.1 se muestra una vista general de la interacción entre los bloques principales del controlador. El microcontrolador, programado con el firmware que utiliza los periféricos para generar las señales PWM necesarias para conducir el inversor trifásico, se encarga de suministrar la corriente al motor y adquirir las mediciones de corriente de las fases, así como la posición del rotor a través del encoder. De esta manera se cierra el lazo de control FOC.

## 2.2. Desarrollo del Hardware

En esta sección se describen de manera resumida los componentes esenciales que integran el controlador, ilustrados en la Figura 2.2.

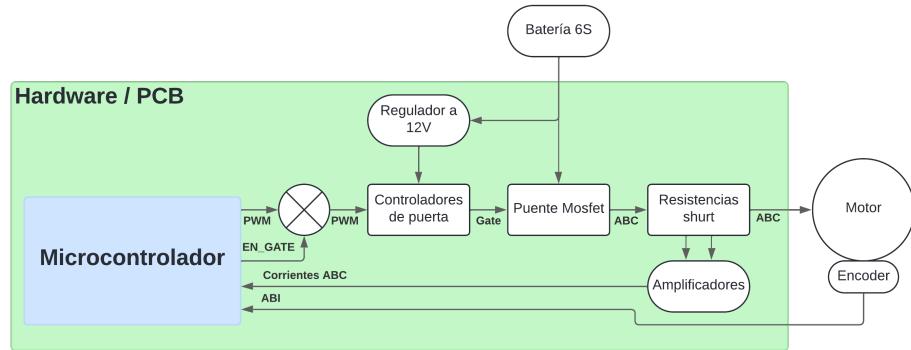


Figura 2.2: Diagrama resumido del hardware.

A continuación se presenta un listado con los principales componentes:

- **Inversor Trifásico (Puente MOSFET):** Comuta la tensión de la batería para generar señales trifásicas y controlar velocidad y par del motor.
- **Controladores de Puerta (Gate Drivers):** Amplifican las señales PWM del microcontrolador para activar y desactivar los MOSFETs de forma eficiente.
- **Resistencias Shunt:** Miden la caída de tensión en cada fase para obtener la corriente, esencial para el control FOC.
- **Motor Brushless (BLDC):** Motor síncrono trifásico sin escobillas que proporciona alta eficiencia y controlabilidad.
- **Encoder:** Sensor de posición angular que informa de la posición y velocidad del rotor para una conmutación sincronizada.
- **Regulador de Voltaje 12V:** Fuente estable de 12V para alimentar correctamente los controladores de puerta.
- **Batería:** Proporciona la potencia principal al inversor, determinando el rendimiento y la autonomía del sistema.

## 2.2.1. Motor

El punto de partida es la selección del motor, que define parámetros cruciales como corriente máxima, voltaje nominal y la velocidad máxima ( $K_v$ ). Estos influyen directamente en la elección de los MOSFETs y resistencias *shunt* para manejar la corriente requerida.

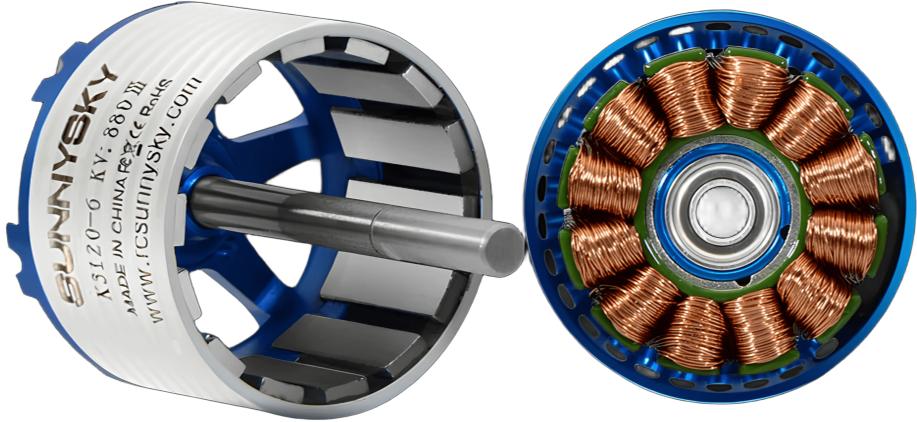


Figura 2.3: SunnySky X3120 V3.

Se ha optado por utilizar el motor **SunnySky X3120 V3** de 585Kv para el diseño del controlador debido a su disponibilidad inmediata. Este es un motor principalmente orientado al uso en drones de tipo aeroplano, pero su gran potencia y bajo peso lo hacen ideal para su uso en robots sumo, como se puede ver en la figura 2.3. Se trata de un motor tipo outrunner, en el cual su eje se extiende y sale por la base del motor.

Tabla 2.1: Características del motor SunnySky X3120 V3 585Kv

Propiedad	Valor
Número de polos del rotor	14
Número de ranuras del estator	12
Voltaje Nominal (celdas de Lipo)	6S (22.2 V)
Constante de velocidad del motor	585 $K_v$
Resistencia del motor	43.5 m $\Omega$
Corriente continua máxima	65 A/30 s
Potencia continua máxima	1625 W
Corriente en vacío	1.0 A/10 V

Algunos de los parámetros más relevantes para el diseño del hardware son el voltaje nominal, que está marcado como 6S (celdas de lipo). Las celdas de lipo varían su voltaje en función de su carga, teniendo un voltaje mínimo de 3V, nominal de 3.7V y máximo de 4.2V. Por esto, una batería de 6 celdas en serie corresponde a un voltaje mínimo de 18V, nominal de 22.2V y máximo de 25.2V. Otro parámetro importante es la corriente continua máxima de 65A.

## 2.2.2. Consideraciones para el Controlador Basadas en el Motor

Tras la selección del motor SunnySky X3120 V3, es crucial revisar los parámetros clave que este impone al diseño del controlador, asegurando que los componentes seleccionados sean adecuados para las exigencias del motor.

El **voltaje máximo del motor** es de 25.2V, correspondiente a una batería LiPo 6S completamente cargada. Para los MOSFETs del inversor, es recomendable seleccionar dispositivos con un voltaje nominal superior al máximo esperado para proporcionar un margen de seguridad. En la práctica, el valor nominal más cercano y comúnmente utilizado es de **30V** para MOSFETs. De manera similar, los condensadores en el bus de voltaje deben soportar este voltaje máximo. Siguiendo las tablas de valores nominales estándar para capacitores, se deben utilizar condensadores con un voltaje nominal de al menos **35V**.

La **corriente continua máxima** del motor es de 65A. Para el diseño del puente MOSFET, es esencial considerar un margen de seguridad para evitar el sobrecalentamiento y asegurar la fiabilidad del sistema. Un factor de seguridad común es multiplicar la corriente continua máxima del motor por un factor de al menos tres. Por lo tanto, el puente MOSFET debería ser capaz de soportar al menos **195A**. Este margen de seguridad garantiza que el controlador pueda manejar picos de corriente y operaciones continuas exigentes sin comprometer los componentes.

Finalmente, la **velocidad máxima teórica** del motor se puede estimar utilizando su constante  $K_v$  (585 RPM/V) y el voltaje máximo de operación (25.2V). Esto resulta en una velocidad teórica máxima de aproximadamente **14.742 RPM** ( $585 \frac{\text{RPM}}{\text{V}} \times 25.2\text{V} \approx 14742\text{RPM}$ ). Esta velocidad máxima es importante para la selección del encoder, asegurando que pueda operar correctamente dentro de este rango de velocidades sin perder precisión o funcionalidad.

En la siguiente tabla se resumen estos parámetros y las consideraciones para la selección de componentes del controlador:

Tabla 2.2: Parámetros Requeridos para el Controlador Basados en el Motor SunnySky X3120 V3

Parámetro del Controlador	Valor Mínimo Requerido
Voltaje Nominal de Operación	22.2V
Voltaje Drain-Source MOSFET ( $V_{DS}$ )	$\geq 30\text{V}$
Corriente Drain-Source MOSFET ( $I_D$ )	$\geq 195\text{A}$
Voltaje de los Capacitores	$\geq 35\text{V}$
Velocidad Máxima del Encoder	$\geq 15,000\text{ RPM}$

### 2.2.3. Encoder

El encoder es el encargado de proveer la información sobre la posición angular del rotor en el motor. Se utilizó el encoder magnético AS5047P (figura 2.4), el cual utiliza un imán especial con una polarización diametral en el eje del motor y un arreglo de sensores HAL internos, para poder medir e interpretar la posición angular del imán.



Figura 2.4: AS5047P.

En esencia, este es un encoder absoluto que funciona principalmente a través de una comunicación SPI, pero opcionalmente incorpora una salida que imita el comportamiento de un encoder incremental con señal index, también conocidos como ABI o ABZ por sus señales. Los microcontroladores STM32 incorporan periféricos que pueden trabajar de forma nativa con estas señales sin mayor intervención del procesador, lo que simplifica en gran medida su implementación en comparación con el uso de SPI, aunque, en consecuencia, se pierden las capacidades de encoder absoluto.

Tabla 2.3: Características del encoder AS5047P

Parámetro	Mín	Típ	Máx	Unidad
Tensión de alimentación (LDO)	4.5	5.0	5.5	V
Tensión de alimentación	3.0	3.3	3.6	V
Corriente de suministro		15		mA
Velocidad máxima			28000	RPM
Resolución del núcleo		14		bits
Resolución ABI	25	512	1024	PPR
Resolución ABI (X4)	100	2048	4096	Pasos/rev

#### 2.2.4. Conjunto motor-encoder

El encoder requiere estar fijo junto al motor y centrado respecto al eje de este para poder realizar su medición. Para ello, se diseñó en Autodesk Inventor y posteriormente se fabricó con impresión 3D un conjunto de soportes y sockets específicos para el motor-encoder.



Figura 2.5: Montura motor-encoder.

En la figura 2.5 se observa la **montura del motor y encoder**. Este conjunto de soporte sujeta el motor y se extiende hacia la parte trasera, de modo que mantiene el encoder firmemente posicionado.



Figura 2.6: Socket Encoder.

En la figura 2.6 se muestra el **socket del encoder**. Este componente asegura que el chip del encoder quede correctamente alineado con el centro del eje del motor.



Figura 2.7: Socket imán axial.

Asimismo, se diseñó un **socket para el imán** (figura 2.7) con el fin de mantener el imán necesario para el funcionamiento del encoder perfectamente centrado en el eje del motor. Este socket se atornilla al rotor y está fabricado en ABS para resistir mejor las temperaturas elevadas que puedan generarse.

## 2.2.5. Puente Mosfet

El puente mosfet es una de las partes del hardware más crítica. Este se encarga de realizar las conmutaciones en las fases del motor. Como se puede ver en la figura 2.8, se divide en 3 piernas (o fases) iguales, donde cada pierna está conformada por 2 mosfets de tipo N en configuración push-pull, es decir, se tiene un mosfet alto conectado a  $V+$  y un mosfet bajo conectado a  $V-$ . Cada mosfet tiene su propia señal de gate, las cuales se deben controlar de forma adecuada para evitar cortocircuitos. Es importante considerar que cada pierna del puente debe soportar por lo menos la corriente máxima del motor, aunque idealmente se deja un margen de seguridad de 2 a 3 veces la corriente máxima. Considerando la corriente de 65A del sunnysky X3120, los mosfets deberán soportar al menos 195A continuos y un voltaje Drain-source superior a los 25.2V para poder trabajar con la batería 6S.

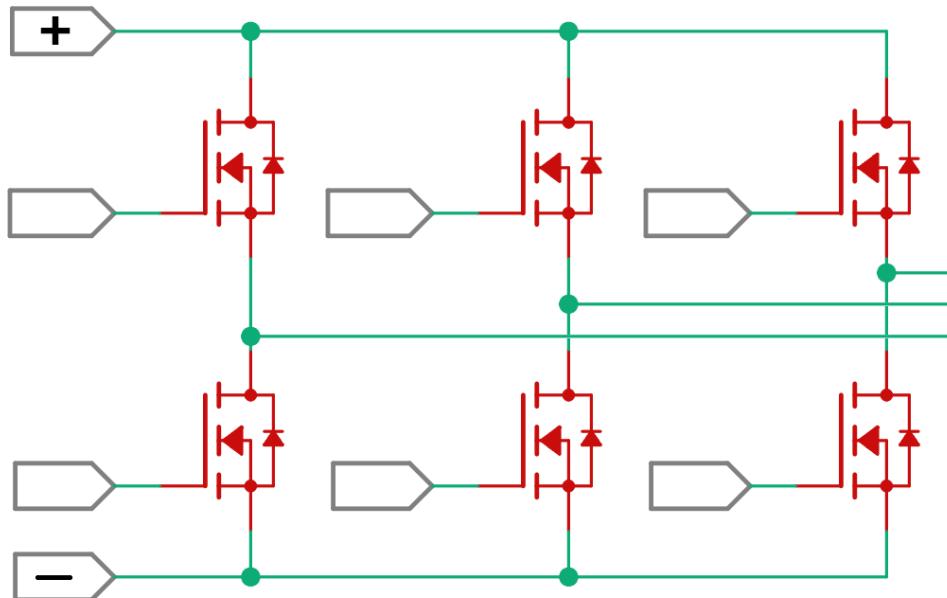


Figura 2.8: Esquema puente MOSFET.

Se utilizó el mosfet **BSC011N03LSI** de la marca Infineon. Es un Mosfet de tipo N con un voltaje Drain-source de 30V. Estos mosfets soportan una corriente continua de 100A.

Tabla 2.4: Características del MOSFET BSC011N03LSI.

Parámetro	Variable	Min	Typ	Max	Unidad
Tensión de drenaje-fuente	$V_{DS}$			30	V
Resistencia encendido	$R_{DS(on)}$	0.9	1.1	1.5	$m\Omega$
Corriente continua de drenaje	$I_D$			100	A
Corriente de pulso drenaje	$I_{D,pulse}$			400	A
Tensión umbral del gate	$V_{GS(th)}$	1.2		2	V
Carga de puerta total	$Q_g$		34	45	$nC$
Potencia de disipación	$P_{tot}$			96	W

## 2.2.6. Controladores de Puerta

El controlador de puerta (*gate driver*) se encarga de recibir las señales PWM provenientes del microcontrolador y, en consecuencia, activar o desactivar los MOSFETs en cada pierna del puente. Su función principal es proporcionar la corriente y el voltaje adecuados para encender y apagar rápidamente los MOSFETs, garantizando una conmutación eficiente y minimizando las pérdidas de energía.

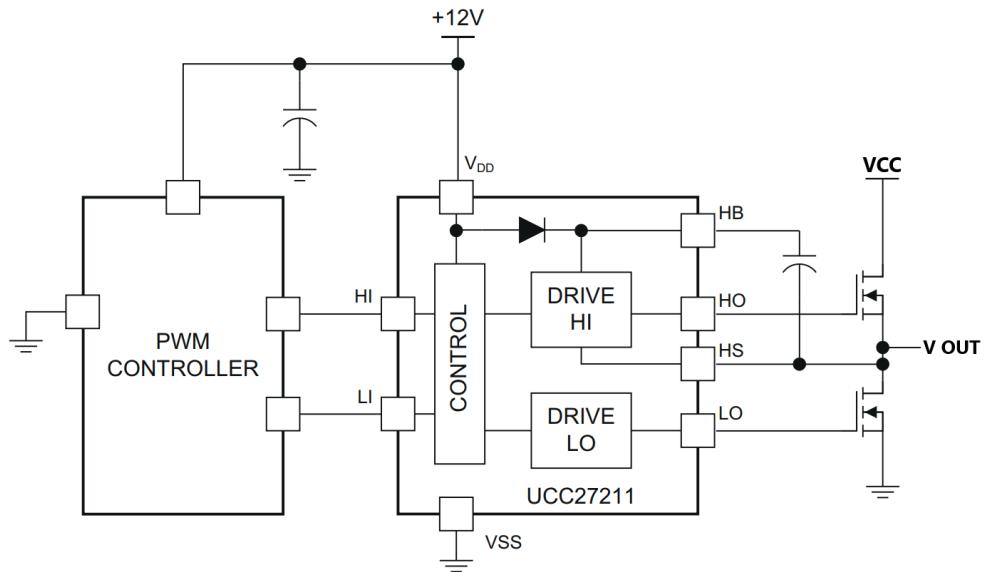


Figura 2.9: Esquema UCC27211.

Se utilizó el controlador de puerta **UCC27211** de Texas Instruments, el cual es un driver de doble canal diseñado específicamente para controlar pares de MOSFETs y, como se puede ver en la figura 2.9, este tiene un esquema de conexiones bastante simple. Este dispositivo ofrece entradas de control independientes para los MOSFETs de alta ( $H_{In}$ ) y baja ( $L_{In}$ ), lo que permite un control preciso y flexible de cada transistor de potencia.

Tabla 2.5: Tabla de Lógica del UCC27211

<b>H In</b>	<b>L In</b>	<b>H Out</b>	<b>L Out</b>	<b>V Out</b>
L	L	L	L	Z
L	H	L	H	GND
H	L	H	L	VCC
H	H	H	H	Corto

Como se puede apreciar en la tabla 2.5, el UCC2711 no incorpora protección interna contra la conducción simultánea de ambos MOSFETs, conocido como **shoot-through**. Este fenómeno ocurre cuando ambos MOSFETs de una pierna se activan y conducen al mismo tiempo, generando un cortocircuito entre el suministro de voltaje y tierra. Para evitar esta situación, es importante agregar un tiempo entre la conmutación de las señales de entrada, conocido como **Dead-Time**, el cual se estima considerando los tiempos de conmutación del controlador de puerta junto con el tiempo de conmutación del mosfet.

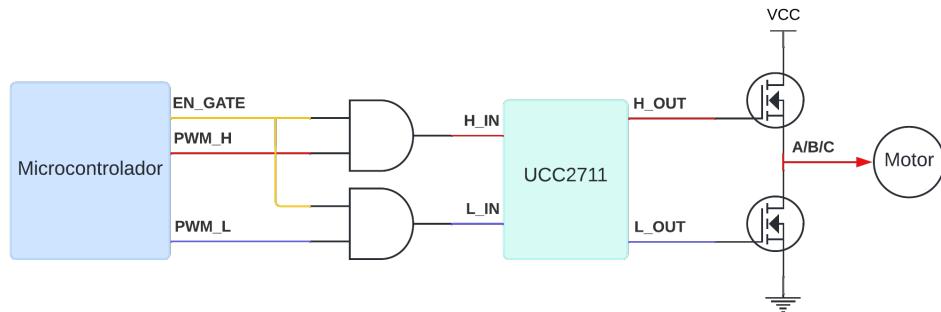


Figura 2.10: Esquema de control PWM con las compuertas AND.

El UCC27211 no dispone de un pin para activar o desactivar el control de los mosfets. Por su parte, la única forma de desactivar el puente mosfet en su totalidad es llevar las señales de entrada alta (H In) y baja (L In) a un estado bajo (L). Además, el microcontrolador, al realizar esta acción, puede llegar a perder el control sobre el estado de las señales PWM y así generar por un breve momento un estado de doble activación de los mosfet. Para evitar esto, se agregaron externamente dos compuertas AND modelo SN74LVC1G08DBVR en cada señal PWM, como se puede ver en la figura 2.10, con el fin de poder cortar las señales desde el microcontrolador de forma segura y así desactivar el puente mosfet. El comportamiento final con este agregado se puede ver en la tabla 2.6.

Tabla 2.6: Tabla de Lógica del UCC27211 con AND

<b>EN GATE</b>	<b>PWM H</b>	<b>PWM L</b>	<b>H In</b>	<b>L In</b>	<b>H Out</b>	<b>L Out</b>	<b>V Out</b>
H	L	L	L	L	L	L	Z
H	L	H	L	H	L	H	GND
H	H	L	H	L	H	L	VCC
H	H	H	H	H	H	H	Corto
L	X	X	L	L	L	L	Z

## 2.2.7. Sensores de Corriente

Para el sensor de corriente se optó por una configuración de resistencia shunt en la salida del puente mosfet, como se puede ver en la figura 2.11. Esta configuración aprovecha la caída de tensión provocada por la resistencia de valor conocido para poder calcular la corriente del circuito. Se utiliza un amplificador instrumental especializado en esta aplicación. En este caso, se utilizó el INA240A1, que tiene una ganancia de 20V/V y se le aplica un offset equivalente a la mitad del voltaje de referencia de 3.3V.

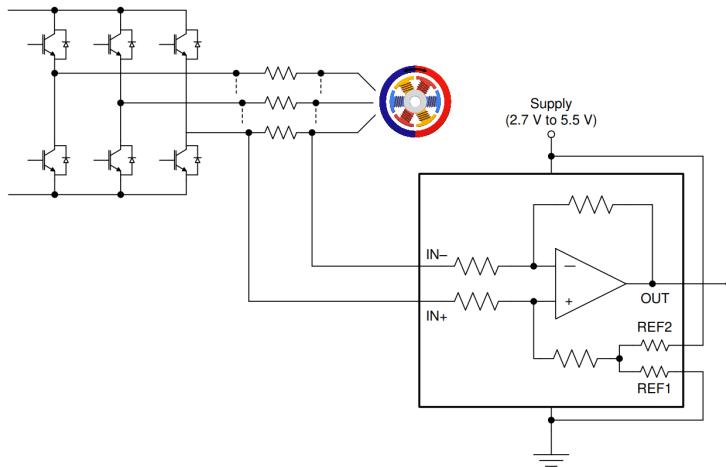


Figura 2.11: Esquema INA240.

Para la resistencia shunt se utilizó una doble resistencia en paralelo, como se ve en la figura 2.13. La resistencia es de  $1\text{m}\Omega$  y 5W de disipación, lo que da una resistencia equivalente de  $500\mu\Omega$  y 10W de disipación. Para determinar la corriente máxima que puede pasar por la resistencia conociendo su valor en ohmios  $R$  y la potencia  $P$  en vatios, se puede calcular utilizando la siguiente fórmula:

$$I_{\max} = \sqrt{\frac{P}{R}}$$

Donde:

- $I_{\max}$  es la corriente máxima,
- $P$  es la potencia en vatios (W),
- $R$  es el valor de la resistencia en ohmios ( $\Omega$ ).

De esta forma, se determina que el máximo de corriente continua que puede circular por las resistencias es de 141.42A.

## 2.3. Implementación del puente MOSFET

El diseño de la PCB está realizado en el software Eagle, ya que este entorno de diseño ofrece una interfaz amigable para el diseño de PCBs, además de una gran adaptabilidad según las necesidades del proyecto.

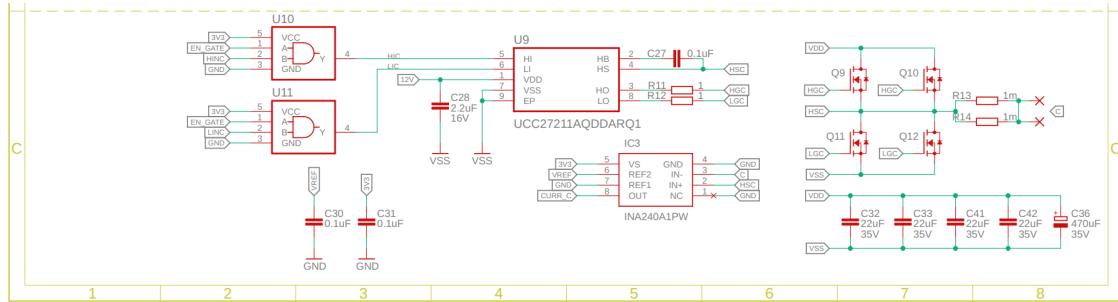


Figura 2.12: Esquemático de la pierna C del inversor.

En la Figura 2.12 se presenta el esquemático de una de las piernas del inversor, específicamente la pierna C. Como se puede apreciar, se conectaron condensadores de desacoplo de 100nF en las entradas de alimentación de los diferentes componentes para mitigar el ruido proveniente de los reguladores de voltaje. Adicionalmente, se incorporó un regulador de voltaje de referencia de 3.3V, el cual no solo sirve como referencia para los ADC, sino que también se utiliza para el ajuste del offset en los amplificadores INA240.

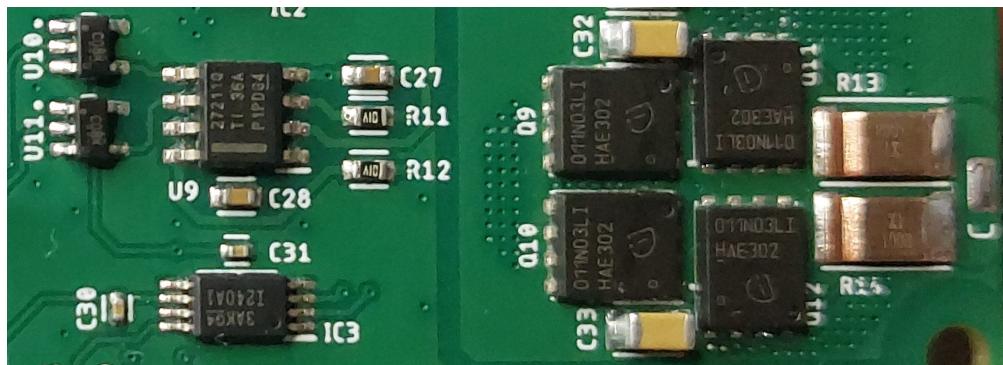


Figura 2.13: Pierna C del inversor.

La implementación física de la pierna C del inversor se muestra en la Figura 2.13. Basado en las características del MOSFET, que soporta una corriente continua de 100 A, se determinó que utilizar solo dos MOSFETs por pierna sería insuficiente. Por lo tanto, se optó por una configuración de cuatro MOSFETs, con doble MOSFET en paralelo. De esta manera, como se observa en la figura, se puede considerar una capacidad total de 200A continuos, despreciando las leves diferencias entre los MOSFETs del mismo modelo.

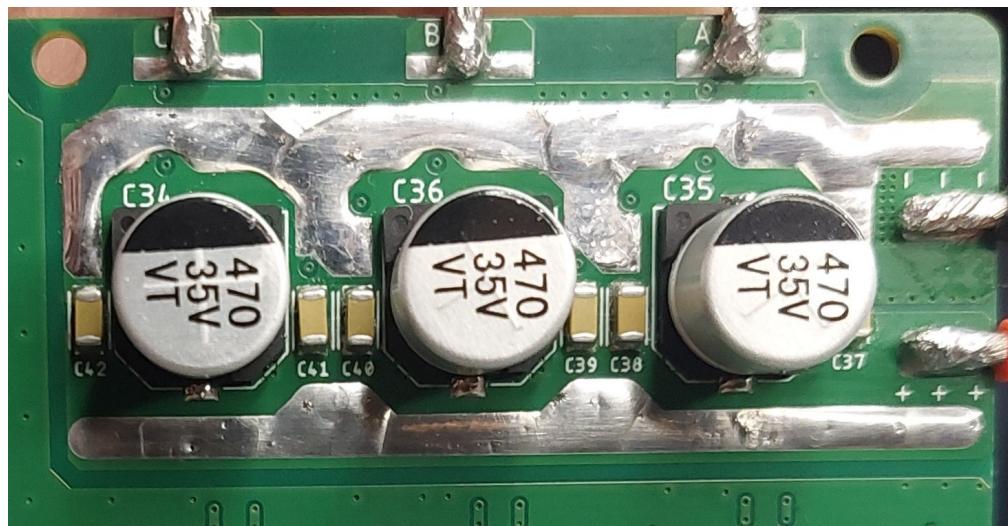


Figura 2.14: Alimentación del Puente MOSFET.

En la Figura 2.14, que muestra la alimentación del puente MOSFET en el reverso de la PCB, se puede apreciar el estañoado realizado en las pistas de alimentación de alta corriente correspondientes al puente MOSFET. Esta medida se implementó en respuesta a las altas exigencias de corriente demandadas por el motor. Conjuntamente, se dispusieron varios capacitores electrolíticos de  $470\mu\text{F}$  y capacitores cerámicos de  $22\mu\text{F}$  estratégicamente posicionados para mitigar los picos de corriente y garantizar la estabilidad del sistema.

## 2.4. Observaciones y posibles mejoras

Respecto al motor, se constató que presenta un elevado efecto de *cogging torque*, lo cual incide de manera significativa en la estabilidad de la velocidad de giro. Existen sistemas comerciales que permiten calibrar y compensar este fenómeno. Sin embargo, en el presente proyecto no se incluyó dicha funcionalidad. Además, al someter el motor a cargas elevadas, se observó un notable incremento de temperatura, pudiendo alcanzar valores superiores al punto de deformación del material PLA utilizado en la fabricación del soporte para el *encoder*. En consecuencia, se recomienda emplear materiales más resistentes al calor, como PETG o ABS, para la elaboración de este soporte.

En cuanto al puente MOSFET, el principal inconveniente radica en la activación simultánea de los transistores (conocida como “doble activación”), cuya causa podría ser una estimación inadecuada del *deadtime* o posibles errores en las señales de control. Para mitigar este problema, se sugiere incorporar compuertas lógicas adicionales que eviten la conducción simultánea o, alternativamente, utilizar controladores de puerta con protecciones integradas, como el ISL6208 de Renesas, el cual ofrece protección activa contra la doble activación y elimina la necesidad de insertar un *deadtime*. No obstante, es importante considerar que dicho controlador soporta menores voltajes de bus para los MOSFET.

Por último, se detectó que, en condiciones de carga extrema, las caídas de tensión en las pistas de alimentación de la PCB alcanzan niveles considerables, incluso cuando se aplica estañado. Para resolver esta situación, se recomienda aumentar el ancho de las pistas o soldar barras de cobre, con el fin de reducir su resistencia y mejorar la distribución de la corriente.

## Capítulo 3

# Firmware

### 3.1. Introducción

Este capítulo busca documentar el desarrollo e implementación del firmware, partiendo de la revisión de los periféricos utilizados y la configuración de estos. Finalmente, se revisará a grandes rasgos el diseño e implementación del algoritmo para el control FOC.

### 3.2. Microcontrolador y periféricos

Se utilizó un microcontrolador de la serie H7 de STM32. Esta es la serie de alto rendimiento de STM32 que incorpora un núcleo ARM Cortex M7 con unidad de cálculo para punto flotante de doble precisión (FPU). El modelo específico es STM32H743VIT6. Su núcleo trabaja a 480MHz, tiene 1MB de memoria RAM y 2MB de memoria flash, ofreciendo así una potencia de cómputo y capacidad bastante elevada para un microcontrolador.

- **Core:** Arm Cortex-M7 de 32bits a 480 MHz, FPU de doble precisión.
- **Memoria Flash:** 2 MB.
- **RAM:** 1 MB total (192 KB de RAM DTCM, 864 KB de SRAM).
- **Voltaje de operación:** 1.62 V - 3.6 V.
- **Empaquetado:** LQFP-100 (14 x 14 mm).
- **Entradas/Salidas:** 82 I/O.
- **Periféricos de comunicación:** USART, UART, SPI, CAN FD, USB, Ethernet.
- **Timers:** Hasta 22 timers (PWM, encoder, control de motores, etc.).
- **ADC:** 3x ADC tipo SAR de 16 bits (hasta 3.6 MSPS).
- **Programación y Depuración:** Interfaz SWD y JTAG.

Los periféricos en un microcontrolador son piezas de hardware especializados en una función que van integrados en el microcontrolador y permiten interactuar con el mundo exterior a través de sus señales eléctricas, ya sean entradas o salidas, digitales o análogas. En el ecosistema de

STM32 existe la herramienta **STM32CubeMX**, que permite configurar todos los periféricos del microcontrolador desde una interfaz gráfica fácil de entender para posteriormente poder generar un código C que implemente todas las configuraciones en el microcontrolador utilizando las librerías HAL de STM32. De esta forma, **STM32CubeMX** genera un código base desde el que empezar a programar el firmware con todos los periféricos ya configurados.

### 3.2.1. Configuración de Timers

Los timers en los microcontroladores STM32 permiten contar eventos y controlar pines de salida según dichos eventos. Estos eventos pueden provenir del reloj interno o de señales externas. La familia STM32 ofrece gran flexibilidad para configurar modos de conteo, fuentes de reloj y funcionalidades específicas, como la generación de PWM o la captura de entrada.

#### Timer 1

El Timer 1, al ser de tipo **Advanced-control** de hasta 16 bits, está especialmente diseñado para la generación de señales PWM complementarias con inserción de *dead time*, lo que facilita el control de puentes MOSFET. Se configura en modo *center-aligned* (cuenta ascendente y descendente), permitiendo tener salidas PWM adecuadas para su uso con el SVM.

Parámetro en CubeMX	Valor
Clock Source	Internal Clock (240 MHz)
Counter Mode	Center Aligned mode 3
Counter Period (16 bits)	5000
Trigger Event Selection TRGO	Update Event
Dead Time	50
Channel 1	PWM Generation CH1 CH1N
PWM Generation Channel 1 and 1N Mode	PWM mode 1
Channel 2	PWM Generation CH2 CH2N
PWM Generation Channel 2 and 2N Mode	PWM mode 1
Channel 3	PWM Generation CH3 CH3N
PWM Generation Channel 3 and 3N Mode	PWM mode 1

Tabla 3.1: Configuración del Timer 1 en CubeMX.

Dado que el reloj base es de 240 MHz y el Counter Period se establece en 5000, se obtiene un evento de desborde a 48 kHz ( $\frac{240 \text{ MHz}}{5000} = 48 \text{ kHz}$ ) (El desborde es cuando el contador llega a su valor mínimo o máximo). En modo *center-aligned*, cada ciclo completo de PWM comprende una fase ascendente y otra descendente, por lo que el PWM resultante es de 24 kHz. Además, las interrupciones del Timer 1 se generan en cada suceso de desborde, por lo que se dispone de una rutina de interrupción a 48 kHz para efectuar los cálculos y actualizaciones del lazo de control.

## Timer 3

El Timer 3 es un timer de uso general de 16 bits, está configurado en **modo encoder**, que permite contar los pulsos de un *encoder* incremental sin intervención directa del firmware. De este modo, el microcontrolador registra por hardware los cambios de las fases A y B, en modo de conteo por 4 (X4). Se configura un *input filter* para evitar pulsos espurios debidos al ruido, y se asigna un Counter Period de 65535 (el máximo para 16 bits), aunque este valor, al corresponder solo a unos pocos giros del motor, hace necesario implementar una gestión del desborde del timer.

## Timer 2

El Timer 2 es un timer de uso general de 32 bits, se utiliza para adquirir mediciones precisas de tiempo relacionadas con la operación del encoder y la sincronización global del sistema. En CubeMX, se configura con reloj interno (*Internal Clock* de 240 MHz) y se opera en modo *Up* con un Counter Period de 4294967295 (valor máximo de 32 bits). A continuación, se describen las conexiones de cada canal:

Parámetro en CubeMX	Valor
Trigger Source	ITR0 (Timer 1 Trigger Out)
Clock Source	Internal Clock (240 MHz)
Counter Mode	Up
Counter Period (32 bits)	4294967295
Channel 1	Input Capture triggered by TRC
Channel 2	Input Capture direct mode
Polarity	Both Edges
Channel 3	Input Capture direct mode
Polarity	Rising Edge

Tabla 3.2: Configuración del Timer 2 en CubeMX.

- **Canal 1 conectado al Trigger Out del Timer 1:** Sirve para registrar con exactitud el instante en que el Timer 1 genera un evento de disparo. También se utiliza para gestionar el desborde del Timer 2 vía software, aumentando el rango de medición de tiempo (al ser de 32 bits, por defecto podría tardar unos 17,9 segundos en desbordar).
- **Canal 2 conectado a la salida XOR de las señales A y B del encoder:** Genera un flanco de subida o bajada en cada pulso del encoder, permitiendo medir, mediante *Input Capture*, el tiempo exacto entre pulsos y así refinar el cálculo de velocidad. Este método aporta mayor exactitud de tiempo que limitarse únicamente al uso del periodo de las interrupciones.
- **Canal 3 conectado a la señal Index del encoder:** Aunque no se usa actualmente para una función concreta, permite determinar con precisión el momento en que dicha señal es activada, facilitando futuras expansiones en la detección de referencias de giro.

### 3.2.2. Configuración de los ADC y su integración con Timers y DMA

En los microcontroladores STM32, los conversores analógico-digital (ADC) se encargan de transformar voltajes analógicos en valores digitales dentro de un rango específico. En la serie H7, cada ADC ofrece una resolución de 16 bits y puede operar en dos modos principales:

- **Conversión regular:** Permite leer hasta 16 canales en una secuencia programable. Se puede combinar con el módulo DMA (Acceso Directo a Memoria) para transferir los valores de conversión directamente a la memoria RAM sin necesidad de intervención del procesador, optimizando así el rendimiento.
- **Conversión inyectada:** Interrumpe la conversión regular para realizar, de forma prioritaria, hasta 4 mediciones de canales configurables. Este modo resulta útil cuando se requiere sincronizar lecturas en instantes precisos o realizar mediciones críticas con mínima latencia.

**DMA (Acceso Directo a Memoria):** El DMA es un módulo que permite trasladar datos entre periféricos y la memoria sin intervención de la CPU. De esta manera, mientras el ADC realiza sus mediciones, el DMA copia automáticamente los resultados a la RAM, liberando al núcleo del microcontrolador de dichas tareas y mejorando la eficiencia general del sistema.

**ADC 1:** El ADC 1 se encarga de la medición de las corrientes de fase, provenientes de los amplificadores operacionales que acondicionan las resistencias *shunt*. Para sincronizar estas mediciones con las interrupciones del sistema y asegurar que las lecturas estén disponibles a tiempo, se utiliza el **modo inyectado** del ADC, asociado a un disparo (*trigger*) generado por el Timer 15.

**Timer 15:** El Timer 15 está configurado como esclavo del Timer 1 y emplea su *Channel 1* en modo de comparación, sin generar salida directa, pero conectado al (*trigger out*). De esta forma, se produce un pulso de sincronización con un pequeño desfase respecto a cuando se ejecute la interrupción del Timer 1. Este pulso activa la lectura inyectada en el ADC 1. Así, los resultados de las mediciones están listos en los registros del ADC cuando se ejecuta la rutina de interrupción.

**ADC 3:** El ADC 3 se utiliza para monitorear el voltaje de bus (por ejemplo, el suministro proveniente de la batería o la fuente principal). Dado que esta lectura no requiere una sincronización tan estricta, se configura en **modo regular**, habilitando conversiones continuas. Adicionalmente, se activa el módulo DMA para transferir los resultados al área de memoria en RAM, evitando sobrecargar la CPU con lecturas repetitivas.

**ADC 2:** Por último, el ADC 2 está destinado a la lectura de señales analógicas utilizadas, en este caso, conectados a potenciómetros utilizados para el control de los setpoints en los controladores. Al igual que en el ADC 3, se configura en **modo regular** con conversiones continuas, y se emplea el DMA para automatizar la transferencia de los datos a memoria. Con esta configuración, se pueden atender varios canales analógicos (por ejemplo, múltiples potenciómetros) sin que el procesador se vea afectado por interrupciones frecuentes.

### 3.3. Estructura del Firmware

Para la gestión del control del motor, el firmware se estructura en torno a dos líneas de ejecución principales que operan de forma pseudo-paralela. Esta arquitectura se basa en la utilización de la interrupción del Timer 1 para la ejecución de las tareas críticas de control, generando lo que se puede conceptualizar como un “*doble hilo de ejecución*”. Esta estructura permite mantener una ejecución continua en la función principal (`main()`) mientras se ejecutan las rutinas de control de motor de alta prioridad en la interrupción.

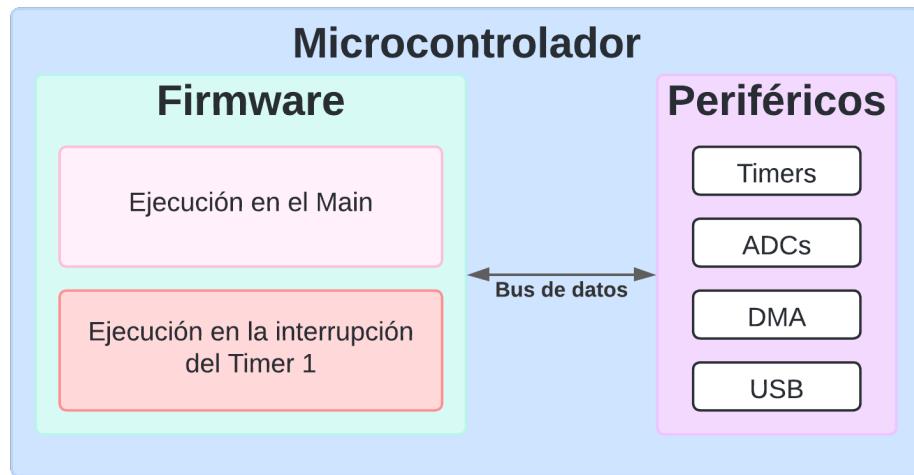


Figura 3.1: Bosquejo de la estructura general del firmware en el microcontrolador.

La Figura 3.1 ilustra esta estructura, mostrando cómo el firmware se compone de estas dos líneas de ejecución que interactúan con los periféricos del microcontrolador para llevar a cabo las funciones de control del motor y la interacción con el sistema.

#### 3.3.1. Ejecución en Main

La ejecución en `main` inicia con una etapa de **configuración del sistema**, en la cual se inicializan los periféricos (por ejemplo, los **ADCs** y los **timers**). Posteriormente, se ejecuta una **secuencia de arranque del motor**, que incluye la calibración a cero en los sensores de corriente, el proceso de *homing* y el ajuste del *offset* en el encoder.

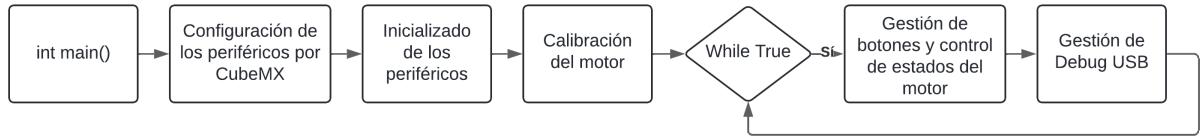


Figura 3.2: Diagrama de flujo de las tareas principales en la línea de ejecución `main()`.

La Figura 3.2 presenta un diagrama de flujo resumido de las tareas principales que se ejecutan en la función `main()`. Tras la inicialización, el programa entra en un bucle `while(true)`, que constituye el “*bucle principal*” del sistema. En este bucle se gestiona la interacción con el usuario a través de pulsadores, se controla el estado del motor y se maneja la comunicación a través del puerto USB para el envío de datos y funcionalidades de depuración.

### 3.3.2. Ejecución en Interrupción

la figura 3.3 muestra a grandes rasgos la **rutina de interrupción** asociada al Timer 1 maneja lo relacionado al **lazo de control del motor** mostrado en la figura 1.2. Su funcionamiento comienza con la obtención de datos provenientes los timers para las medisiones de tiempo y posición, así como la abstención de los datos medidos por los ADCs que son almacenados en la memoria RAM por el DMA, estos datos son procesados y normalizados para ser aplicados en las **transformadas de Clarke y Park** y así obtener las variables de estado del motor.

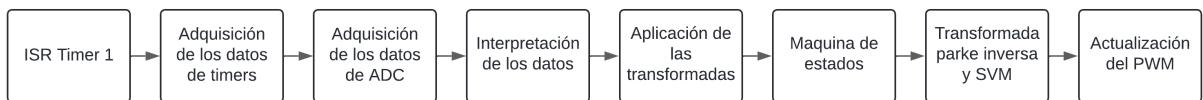


Figura 3.3: Diagrama de flujo de las tareas principales en la línea de ejecución de la interrupción.

Se utiliza una **máquina de estados** para la ejecución de diferentes tareas, esta es controlada directamente desde la ejecución `main` y tiene la capacidad de concatenar varias acciones, de esta forma es capaz de realizar las calibraciones del sistema, hacer pruebas con modos alternativos de control o ejecutar el lazo de control de velocidad. Finalmente, los valores de salida se usan para calcular la transformada inversa de Park y consecutivamente el **SVM** para obtener los valores para los PWMs.

### **3.4. Observaciones y posibles mejoras**

Durante el desarrollo del firmware, se identificó una consideración relevante para el sistema: la familia de microcontroladores STM32H7 presenta un error de arquitectura que complica el uso del DMA en condiciones normales. Este problema se debe a la división de la memoria RAM en regiones diferentes, lo que obliga a especificar la región de memoria deseada para cada variable o registro objetivo del DMA.

En lo que respecta al entorno de desarrollo, fue necesario emplear una máquina virtual con Ubuntu a través de WSL (Windows Subsystem for Linux) para compilar y escribir el código fuente utilizando VSCode, dado que la instalación del entorno en Windows no ofreció los resultados esperados.

En cuanto al lenguaje de programación, aunque el firmware se implementó en C, se observó que los microcontroladores STM32 también pueden ser programados en C++ con ligeras modificaciones al archivo `main`, lo que podría simplificar la gestión de recursos y la modularidad del código al permitir el uso de objetos.

Por último, para organizar de manera eficaz las subtareas y propiciar una mayor escalabilidad, se sugiere utilizar FreeRTOS. Este sistema operativo de tiempo real para microcontroladores puede integrarse directamente desde STM32CubeMX, proporcionando una estructura más robusta para el manejo de hilos de ejecución (threads) y facilitando la segmentación adecuada de las funciones que componen el firmware.

## Capítulo 4

# Validación

Para las validaciones, se realizó una adquisición de datos, donde estos se recopilan ciclo a ciclo en la memoria RAM del microcontrolador para su posterior envío de forma asincrónica por el puerto USB, como se muestra en la figura 4.1. Los datos son recibidos en una computadora a través de un código en Python, donde se almacenan en archivos CSV. Esta recopilación de información se realiza al final de cada ejecución de la interrupción del sistema, la cual se ejecuta a una frecuencia de 48 kHz. Debido a las limitaciones de memoria, la recopilación de datos solo puede llevarse a cabo durante un período de aproximadamente 100 ms, hasta alcanzar la capacidad máxima de la memoria.

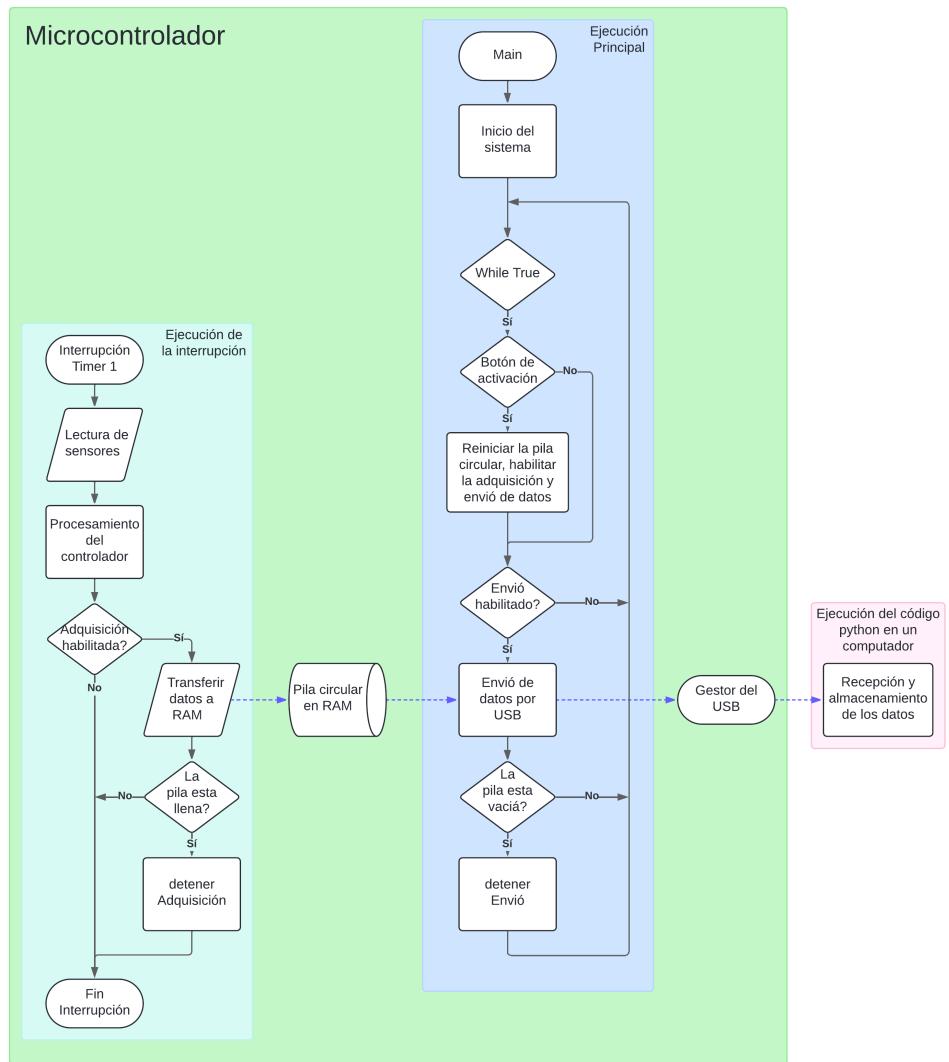


Figura 4.1: Diagrama de flujo adquisición de datos.

## 4.1. Validación de la adquisición y transformación de las mediciones de corriente

### 4.1.1. Validación de las mediciones de corriente

Se validarán las mediciones de corriente adquiridas desde el microcontrolador.

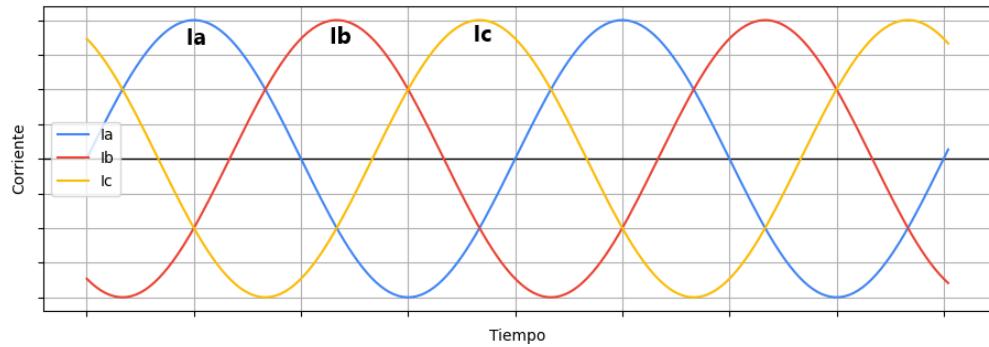


Figura 4.2: Corrientes ideales en un sistema trifásico equilibrado.

En la figura 4.2 se representan las señales ideales de un sistema trifásico equilibrado, con las tres corrientes de fase  $I_a$ ,  $I_b$  e  $I_c$ , cada una de ellas con una forma senoidal pura y un desfase de 120° entre sí.

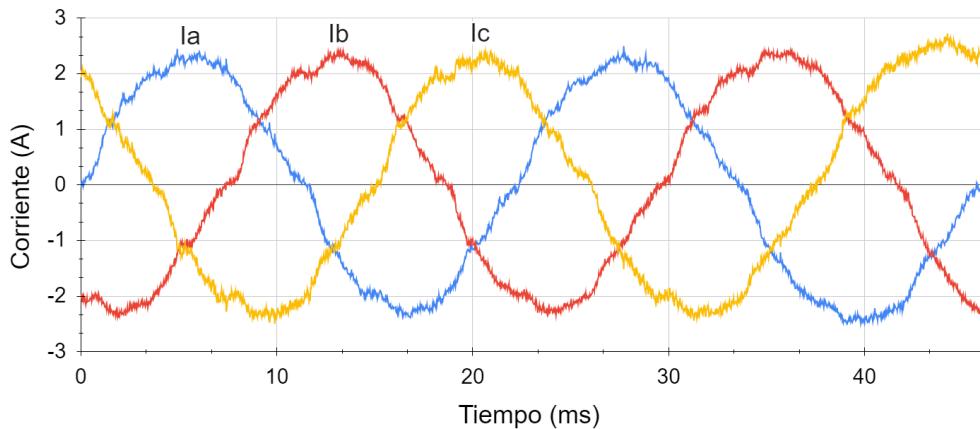


Figura 4.3: Corrientes medidas en el sistema trifásico.

En la figura 4.3 se pueden apreciar las corrientes de fase, las cuales presentan una cantidad significativa de ruido, aun cuando internamente pasan por un filtro complementario con una frecuencia de corte  $f_W = 12000\text{Hz}$ . No obstante, mantienen un comportamiento sinusoidal con el desfase de 120° entre señales, como es característico de un sistema trifásico equilibrado. Sin embargo, la cantidad de ruido podría indicar que sería necesario disminuir la frecuencia de corte en el filtro complementario o agregar un filtro pasivo en el circuito.

#### 4.1.2. Validación de la transformada de Clarke

Se validarán los resultados a la salida de la transformada de Clarke en el microcontrolador.

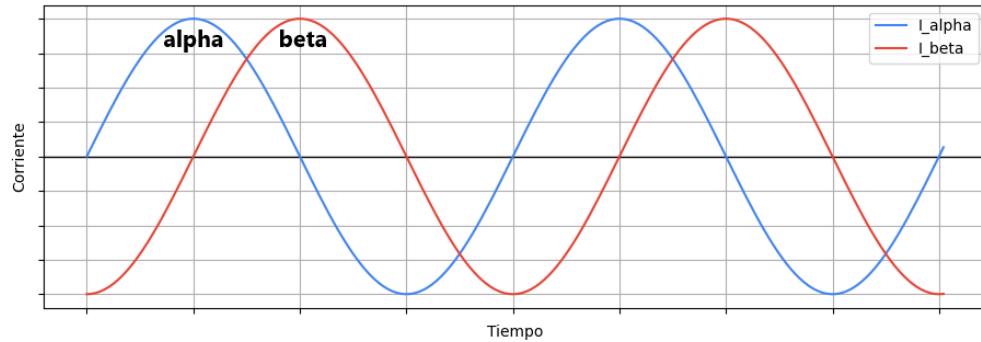


Figura 4.4: Corrientes ideales en el plano  $\alpha\beta$ .

En la figura 4.4 se representan las corrientes ideales en el plano  $\alpha\beta$ , con las formas senoidales puras con un desfase de  $90^\circ$  entre sí.

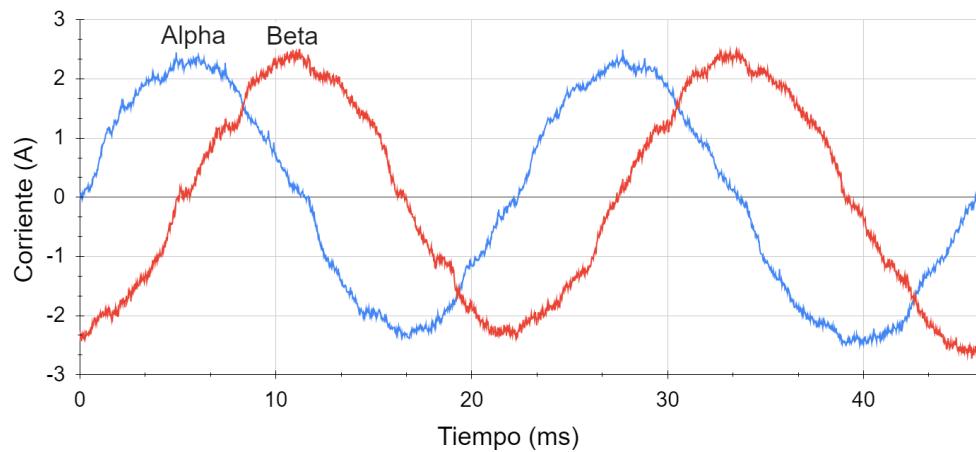


Figura 4.5: Corrientes medidas en el plano  $\alpha\beta$ .

En la figura 4.5 se puede apreciar cómo las corrientes  $\alpha\beta$  reflejan el ruido presente en las mediciones de los sensores de corriente, pero mantienen el comportamiento esperado a la salida de la transformada de Clarke, con las dos señales sinusoidales con un desfase de  $90^\circ$  entre señales, como es característico.

#### 4.1.3. Validación de la transformada de Park

Se validarán los resultados a la salida de la transformada de Park en el microcontrolador.

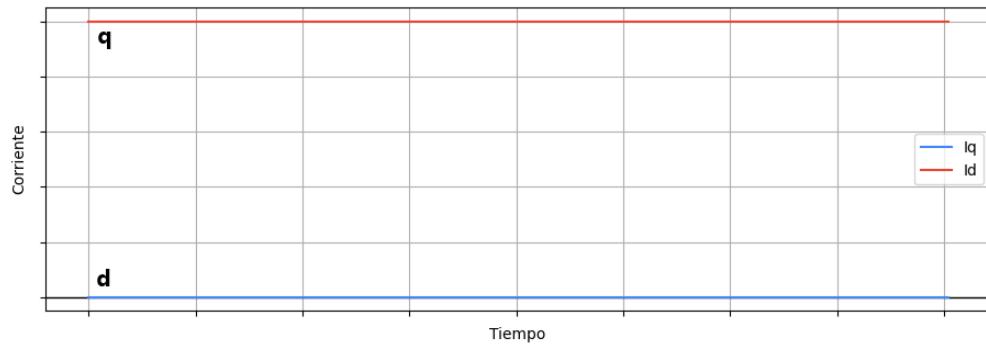


Figura 4.6: Corrientes ideales en el plano  $dq$ .

En la figura 4.6 se representan las corrientes ideales en el plano  $dq$ , donde lo ideal es que la corriente directa tenga un valor de cero, para mantener la eficiencia del sistema, mientras que solo la corriente de cuadratura tiene un valor distinto de cero.

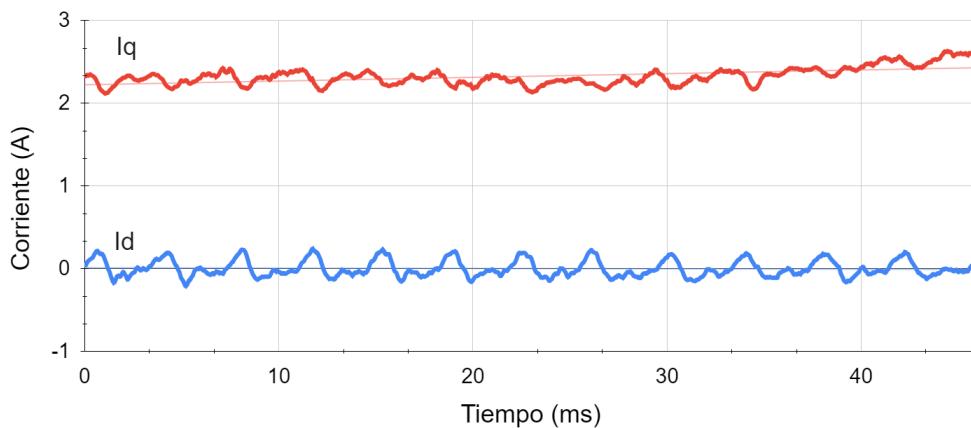


Figura 4.7: Corrientes medidas en el plano  $dq$ .

En la figura 4.7 se puede apreciar cómo las corrientes  $dq$  presentan una menor cantidad de ruido gracias a que su filtro complementario está ajustado para una frecuencia de corte de  $f_W = 800\text{Hz}$ . Aunque igualmente presentan ciertas deformaciones e inestabilidad con un patrón aparentemente constante, en términos generales, mantienen aproximadamente el comportamiento esperado a la salida de la transformada de Park.

## 4.2. Validación de los Controladores PI

En esta validación se busca comprobar si los controladores PI de velocidad y corriente son capaces de mantener sus setpoints. Las pruebas se realizaron de forma estática, aplicando una carga ligera sobre el motor y capturando datos durante este proceso.

### 4.2.1. Validación del controlador de velocidad

Para la validación, se aplicó un setpoint de 116.8 RPM utilizando uno de los potenciómetros disponibles.

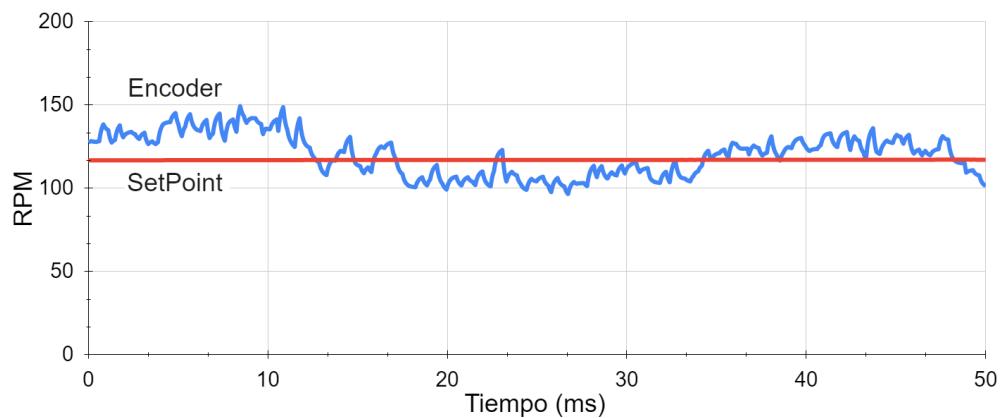


Figura 4.8: Velocidad medida por el encoder y setpoint de velocidad.

En la Figura 4.8, se observa que la velocidad medida por el encoder sigue adecuadamente el setpoint establecido de 116.8 RPM. A pesar de ligeras oscilaciones, el controlador de velocidad mantiene el régimen deseado, demostrando su capacidad para alcanzar y mantener el setpoint bajo condiciones de carga estática.

#### 4.2.2. Validación del controlador de corriente

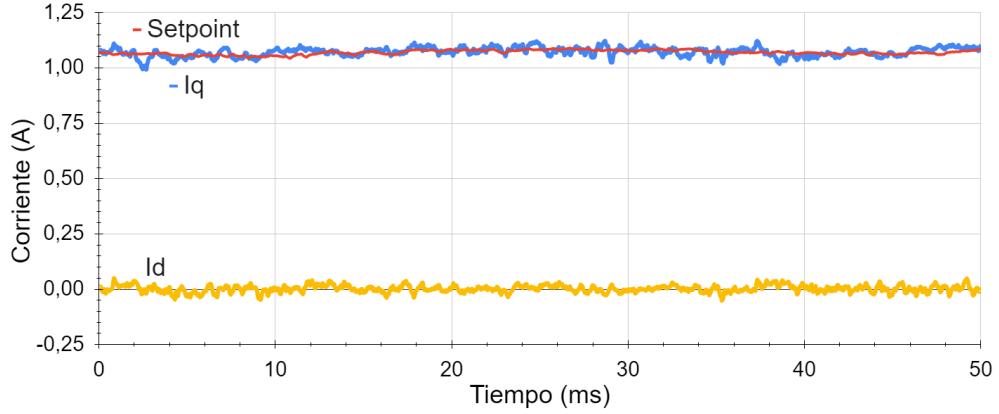


Figura 4.9: Corrientes medidas en el plano  $dq$ .

Como se muestra en la Figura 4.9, las corrientes en el plano  $dq$  indican que el controlador de corriente logra mantener la corriente de cuadratura ( $I_q$ ) cercana al valor de referencia proporcionado por el controlador de velocidad, mientras que la corriente directa ( $I_d$ ) se mantiene próxima a cero. Esto evidencia que el controlador de corriente regula eficazmente las corrientes según los setpoints establecidos.

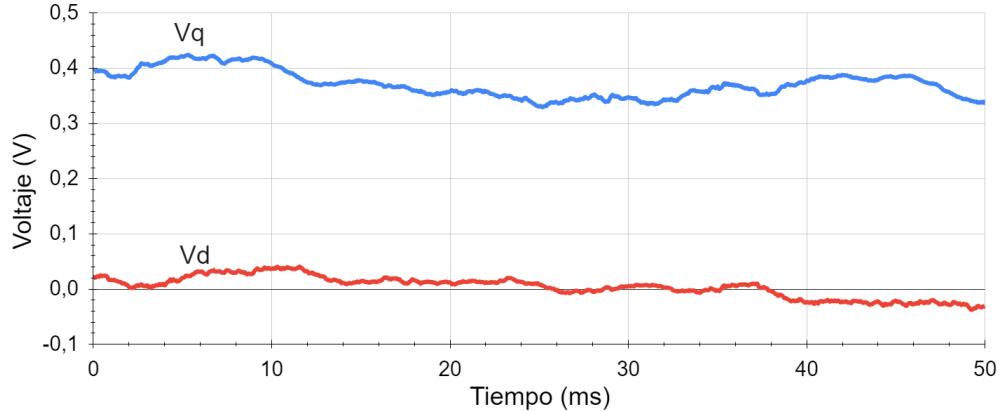


Figura 4.10: Voltajes en el plano  $dq$ .

Además, la Figura 4.10 presenta los voltajes en el plano  $dq$ , donde se aprecia que las tensiones generadas están dentro de los valores esperados para mantener las corrientes deseadas. Esto corrobora que el controlador de corriente responde adecuadamente a las demandas del sistema, contribuyendo al correcto desempeño del motor bajo condiciones de carga.

### 4.3. Validación señales del SVM

Para validar el funcionamiento del modulador por vector espacial (SVM), se realizaron pruebas virtuales aplicando diferentes valores de tensión de referencia  $V_{ref}$  al SVM para observar su comportamiento. Durante estas pruebas, se utilizó un analizador lógico para capturar las señales PWM generadas por el microcontrolador. En la captura presentada, se aisló un ciclo completo del PWM donde se puede apreciar de forma clara la secuencia de activación correspondiente al sector 1.

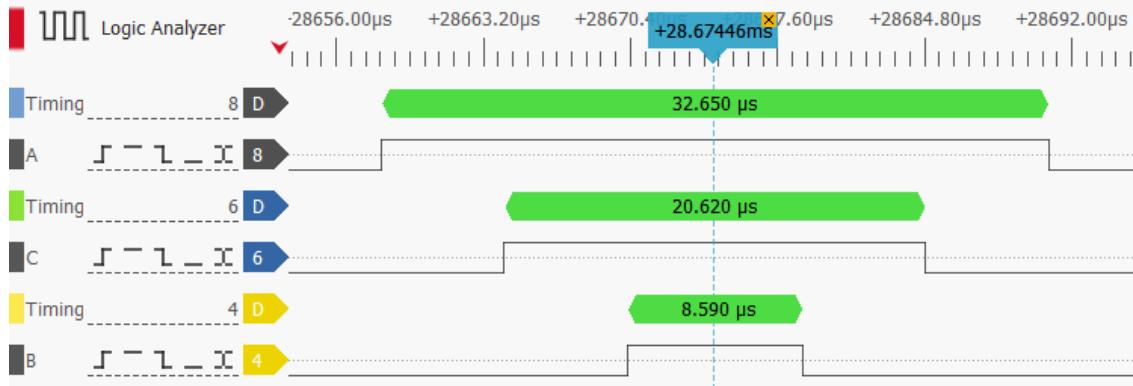


Figura 4.11: Señales del timer.

En la Figura 4.11, se observan las señales PWM correspondientes a las tres fases generadas por el microcontrolador. La secuencia de activación muestra que el SVM implementado sigue correctamente el patrón teórico esperado para el sector 1, evidenciando una conmutación precisa y sincronizada de los transistores del inversor. Esto confirma que el SVM está modulando adecuadamente las señales PWM para generar los vectores de tensión requeridos.

# Comentarios y Conclusiones

**Respecto al objetivo:** “Estudiar los principios del Control de Campo Orientado (FOC) y la modulación de espacio vectorial (SVM) para aplicarlos en el diseño del controlador” Se concluye que la base teórica relacionada con el funcionamiento de los motores *brushless* y el Control de Campo Orientado (FOC) abarca un espectro más amplio que el revisado en este proyecto. Las transformadas de Clarke y Park simplifican el análisis e implementación del sistema, aunque sería posible profundizar en metodologías de control FOC directo.

**Respecto al objetivo:** “Diseñar el hardware para el controlador FOC, con los componentes mínimos necesarios para validar el funcionamiento” Se concluye que el uso de Autodesk *Eagle* facilita de manera significativa el proceso de diseño tanto del esquemático como de la placa de circuito impreso (PCB). La disponibilidad de recursos y librerías en línea agiliza la búsqueda de los componentes específicos y su integración, garantizando la coherencia entre el símbolo de cada dispositivo y su respectivo *footprint*. Además, la compatibilidad con los servicios de *JLCPCB*, que proporciona reglas de diseño y formatos de archivos *Gerber* precisos, contribuyó a la fabricación eficiente y libre de inconvenientes de la placa.

**Respecto al objetivo:** “Configurar y programar el microcontrolador para el algoritmo FOC, utilizando las librerías HAL de STM32” Se concluye que la utilización de *STM32CubeMX* reduce de forma notable la complejidad inicial de configuración de los periféricos, puesto que automatiza gran parte del proceso y evita lidiar directamente con registros de bajo nivel y los usos avanzados de la biblioteca HAL. No obstante, el elevado número de módulos disponibles en la familia STM32H7 y la gran variedad de parámetros de configuración requirieron una curva de aprendizaje considerable para conseguir un ajuste óptimo. A pesar de ello, la configuración final aprovecha de manera significativa las prestaciones del microcontrolador.

**Respecto al objetivo:** “Validar el funcionamiento del controlador y proponer posibles mejoras para su aplicación en robótica competitiva” Se concluye que el controlador desarrollado exhibe un desempeño estable en condiciones normales de operación. Sin embargo, las pruebas evidenciaron dificultades cuando se exige al sistema con aceleraciones bruscas o cambios de dirección repentinos. Además, aunque las compuertas AND permiten deshabilitar con seguridad el puente MOSFET ante posibles fallas, persiste el inconveniente de doble activación (*shoot-through*) en determinadas circunstancias, lo que puede comprometer la integridad del prototipo. Dichas observaciones confirmán la necesidad de perfeccionar tanto el hardware como el firmware para afrontar escenarios de carga extrema.

**Respecto al objetivo principal: “Implementar un controlador de tipo FOC para motores brushless con encoder, utilizando un microcontrolador STM32, que sirva de base para un driver especializado en la robótica competitiva”** Se concluye que el controlador constituye una solución funcional para requerimientos estándar y cumple con la finalidad de servir como base para un *driver* especializado en robótica competitiva. A lo largo del documento, se sugieren mejoras potenciales en ambas vertientes, hardware y firmware, que podrían optimizar el comportamiento del sistema en condiciones extremas, habituales en competencias de robótica. Adicionalmente, el uso de la familia STM32, en particular de la serie H7, ofreció un rendimiento notable, aun considerando ciertas dificultades de arquitectura que surgieron durante la fase de desarrollo. En consecuencia, se confirma la versatilidad de estos dispositivos para aplicaciones de alto desempeño y se resalta su amplio margen de crecimiento en futuros desarrollos.

## Trabajos Futuros

- **Refinar la gestión del *deadtime*:** Emplear controladores de puerta con protecciones integradas o añadir lógica adicional para prevenir la doble activación (*shoot-through*) y preservar la confiabilidad del puente MOSFET.
- **Ampliar la robustez en el hardware:** Aumentar el ancho de las pistas o soldar barras de cobre en la PCB para soportar mejor los picos de corriente, junto con el uso de materiales termorresistentes (PETG o ABS) en piezas críticas que tengan contacto directo con el motor.
- **Optimizar la ejecución del firmware:** Migrar el código a C++ e integrar FreeRTOS para aprovechar la programación orientada a objetos y el manejo de hilos, facilitando la escalabilidad y la organización de tareas concurrentes.
- **Mejorar el filtrado de la señal de corriente:** Reducir la frecuencia de corte del filtro o añadir etapas analógicas pasivas para minimizar el ruido en las mediciones, incrementando así la precisión en el lazo de control.
- **Aumentar la versatilidad del sistema:** Explorar el soporte para un segundo motor y ampliar la capacidad de adquisición de datos, permitiendo el desarrollo de múltiples ejes de control para aplicaciones robóticas de mayor complejidad.
- **Incorporar compensaciones de *cogging torque*:** Implementar algoritmos de calibración y compensación que disminuyan la vibración producida por el entrehierro, optimizando así la estabilidad de velocidad y el rendimiento general del motor.

**Link al repositorio Github:** [https://github.com/eziron/driver\\_FOC](https://github.com/eziron/driver_FOC)

# Bibliografía

- [1] Charles Frick. *Brushless DC Motors Introduction for Next-Generation Missile Actuation Systems*. Technical Article TA20788-0-10/18. Norwood, MA, USA: Analog Devices, Inc., 2018. URL: <https://www.analog.com/media/en/technical-documentation/technical-articles/brushless-dc-motors-introduction-for-next-generation-missile-actuation-systems.pdf>.
- [2] Pete Millet. *Brushless vs. Brushed DC Motors: When and Why to Choose One Over the Other*. MonolithicPower.com. Jun. de 2022. URL: <https://MonolithicPower.com>.
- [3] Deepak Mohanraj et al. «A Review of BLDC Motor: State of Art, Advanced Control Techniques, and Applications». En: (2022). DOI: 10.1109/ACCESS.2022.3175011. URL: <https://ieeexplore.ieee.org/abstract/document/9774372>.
- [4] Matias Gualtieri Lara. «Sensorless Brushless DC Motors. Development and comparison of different fault tolerant control algorithms». Relatore: Prof. Paolo Maggiore. Tesi di Laurea Magistrale. Corso di laurea magistrale in Ingegneria Aerospaziale: Politecnico di Torino, mar. de 2018, págs. 1-49. URL: <http://webthesis.biblio.polito.it/id/eprint/6891>.
- [5] Padmaraja Yedamale. *Brushless DC (BLDC) Motor Fundamentals*. AN885. Jul. de 2003. URL: <https://www.microchip.com/en-us/application-notes/an885>.
- [6] Bin Wu. *High-Power Converters and AC Drives*. 1.<sup>a</sup> ed. Hoboken, NJ: Wiley-IEEE Press, 2006, págs. 296-306. ISBN: 978-0471731719. URL: <https://ieeexplore.ieee.org/servlet/opac?bknumber=5237895>.
- [7] Antun Skuric et al. «SimpleFOC: A Field Oriented Control (FOC) Library for Controlling Brushless Direct Current (BLDC) and Stepper Motors». En: *Journal of Open Source Software* 7.74 (jun. de 2022), pág. 4232. DOI: 10.21105/joss.04232. URL: <https://doi.org/10.21105/joss.04232>.
- [8] Jorge Zambada y Debraj Deb. *Sensorless Field Oriented Control of a PMSM using a Sliding Mode Observer (SMO)*. AN1078. Mayo de 2010. URL: <https://www.microchip.com/en-us/application-notes/an1078>.