UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Control Vectorial del par motor de un motor brushless

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Matevosyan, Rafael

TUTORIZADO POR

Sánchez Díaz, Carlos

CURSO ACADÉMICO: 2020/2021

RESUMEN

Hoy en día los motores brushless -BLDC- se encuentran presentes en diferentes ámbitos como en la industria automóvil, la robótica, instrumentos médicos, vehículos eléctricos, etc. Esto se debe a la gran cantidad de ventajas que presentan frente a otro tipo de motores, como los DC con escobillas o los motores de inducción, como por ejemplo su alta eficiencia, mayor rango de velocidad, o su mayor vida útil, por enumerar algunos. Sin embargo, tienen la principal desventaja de ser más costosos, ya que requieren del uso de un inversor trifásico para su control.

Idealmente, un BLDC motor presenta una fuerza contraelectromotriz trapezoidal y una fuente de tensión constante. Por lo tanto, en teoría, la corriente generada por la fuente debe de aumentar instantáneamente a un estado estable, en el motor, y producir un par motor sin rizado. Sin embargo, en la práctica las características de la corriente difieren del caso ideal, ya que la corriente está influenciada por la resistencia y la inductancia del bobinado del motor, y por lo tanto no puede alcanzar un estado estable de forma instantánea. En consecuencia, se genera una corriente de rizado que afecta directamente al par motor.

Para solucionar este problema se ha planteado para este trabajo de fin de grado usar el método de control vectorial del par motor, que también es conocido como Control de Campo Orientado, FOC. Cuyo objetivo es controlar el par motor y a su vez la velocidad del motor. Para ello, se va a diseñar un circuito de control en lazo cerrado que estará formado por una etapa de control, y otra de medición, los cuales permitirán disminuir el rizado del par motor y en consecuencia aumentar el rendimiento del motor.

RESUM

Hui dia els motors brushless -BLDC- es troben presents en diferents àmbits com en la indústria automòbil, la robòtica, instruments mèdics, vehicles elèctrics, etc. Això es deu a la gran quantitat d'avantatges que presenten en comparació amb altres tipus de motors, com els DC amb escombretes o els motors d'inducció, com per exemple la seua alta eficiència, major rang de velocitat, o la seua major vida útil, per enumerar alguns. No obstant això, tenen el principal desavantatge de ser més costosos, ja que requereixen de l'ús d'un inversor trifàsic per al seu control.

Idealment, un BLDC motor presenta una força contraelectromotriu trapezoidal i una font de tensió constant. Per tant, en teoria, el corrent generat per la font ha d'augmentar instantàniament a un estat estable, en el motor, i produir un parell motor sense arrissat. No obstant això, en la pràctica les característiques del corrent difereixen del cas ideal, ja que el corrent està influenciat per la resistència i la inductància del bobinatge del motor, i per tant no pot aconseguir un estat estable de manera instantània. En conseqüència, es genera un corrent d'arrissat que afecta directament el parell motor.

Per a solucionar aquest problema s'ha plantejat per a aquest treball de fi de grau usar el mètode de control vectorial del parell motor, que també és conegut com a Control de Camp Orientat, FOC. L'objectiu del qual és controlar el parell motor i al seu torn la velocitat del motor. Per a això, es dissenyarà un circuit de control en llaç tancat que estarà format per una etapa de control, i una altra de mesurament, els quals permetran disminuir l'arrissat del parell motor i en conseqüència augmentar el rendiment del motor.

ABSTRACT

Nowadays brushless motors -BLDC- are present in different areas such as in the automobile industry, robotics, medical instruments, electric vehicles, etc. This is due to the large number of advantages they have compared to other types of engines, such as brushed DC or induction motors, for example its high efficiency, greater speed range, or longer service life, to list a few. However, they have the main disadvantage of being more expensive since they require the use of a three-phase inverter.

Ideally, a BLDC motor exhibits a trapezoidal back EMF and a constant voltage source. Therefore, in theory, the current generated by the source should instantly rise to a steady state, in the motor, and produce a torque motor without any ripple. However, in practice the characteristics of the current differ from the ideal case, since the current is influenced by the resistance and inductance of the motor winding, and therefore cannot reach a steady state instantly. Consequently, a ripple current is generated that directly affects the engine torque.

To solve this problem, it has been proposed for this final degree project to use the vector control method to control the motor torque, which is also known as Field Oriented Control, FOC. The objective of this method is to control the motor torque and in turn the motor speed. To do this, we are going to design a closed-loop control circuit that will consist of a control stage with feedback, which will reduce the ripple of the motor torque and consequently increase engine performance.

Índice

Contenidos del TFG

- Memoria
- Pliego de condiciones
- Presupuesto
- Planos

Índice de figuras

Índice de tablas

Memoria

Índice de la memoria

1. Introducción

El uso de motores se ha extendido a todo tipo de campos en el último siglo y supone el principal dispositivo electrónico-mecánico de conversión de energía desde hace más de dos. Uno de los más usados son los motores sin escobillas (BLDC) y debido a varios factores el precio de estos motores ha bajado significativamente y su disponibilidad ha aumentado con creces. Sin embargo, las técnicas de control para este tipo de motores se han mantenido bastante anticuados. Principalmente se controlan mediante un control escalar en bucle abierto sin ningún tipo de realimentación. Esto es debido a que los motores BLDC son usados en aplicaciones donde no se requiere de un control preciso de la velocidad o de posición, como por ejemplo para mover las hélices de un helicóptero.

Los motores BLDC, en general, son diseñados para que su potencia de salida sea tan alta como sea posible. La razón es para aumentar su autonomía. Un aspecto que no se ha considerado es que este tipo de motores también se pueden llegar a usar en los campos de la robótica y automatización usando técnicas de control más sofisticadas. Las ventajas del uso de los motores BLDC en estos campos son varias frente a los motores con escobillas o los motores de inducción: presentan mayor eficiencia, mayor vida útil, menor ruido, mayor rango de velocidad, etc.

La principal motivación de este trabajo fin de grado es aprovechar su alta potencia por ratio de masa de este tipo de motores para controlarlos usado la técnica de control de campo orientado, de aquí en adelante FOC, para proporcionar un control preciso de velocidad. Con este método las ecuaciones del motor son transformadas en un sistema de coordenadas que rota en sincronía con el flujo de los imanes permanentes. De esta manera se separan las ecuaciones del motor y con ello se consigue controlar indirectamente tanto el flujo como el torque mediante un controlador PI de corriente.

1.1 Objetivos

Los objetivos principales del presente trabajo se podrían resumir en:

- Demostrar, mediante las simulaciones, cómo la técnica de FOC permite realizar un control preciso de velocidad.
- Implementar el control FOC en un sistema real.
- Determinar los valores correctos de las constantes de los dos controladores PI que se usan en el FOC.
- Ensayar el motor en régimen de carga mecánica para poder validar el comportamiento del control.

1.2 Estructura

El presente trabajo se estructura en capítulos donde de forma progresiva se adentra en cada parte relevante y que en su conjunto conforman el proyecto.

En el primer capítulo se muestra una introducción de la memoria y trata sobre los objetivos, la principal motivación por la que se decido realizar este trabajo, su metodología y su justificación.

En el segundo capítulo hace un breve repaso de las partes que componen un motor sin escobillas, de su principio de funcionamiento.

En el tercer capítulo, se describen las técnicas de control más habituales para mover los motores BLDC y de la razón por la que se ha decidido usar la técnica de FOC.

En el cuarto capítulo, se detalla el funcionamiento del control vectorial y de cada una de las partes que lo forman y de algunas características que tiene esta técnica.

En el quinto capítulo se muestra una simulación en MATLAB del control vectorial y se muestra cada una de las partes que lo forman.

En el sexto capítulo se realiza una muestra experimental donde se puede observar el funcionamiento del FOC.

Con los datos del capítulo anterior se detallará en los capítulos siete y ocho las conclusiones obtenidas y las posibles futuras líneas de trabajo.

Finalmente, se incluye un presupuesto del sistema implementado, los planos del proyecto unos anexos donde se incluye imágenes y el código usado y una bibliografía.

1.3. Justificación

El control trapezoidal, a pesar de ser la técnica que se ha usado tradicionalmente, presenta el gran problema de que genera oscilaciones en el par motor. Por lo que para conseguir una mejor respuesta dinámica se requieren de técnicas de control mucho más avanzadas y una de ellas es el FOC que permite controlar el par motor y la velocidad basándose en el estado electromagnético del motor, al igual que un motor DC. FOC es la primera tecnología que realiza un control “real” de las variables de par motor y flujo. Con el desacoplamiento entre las componentes de la corriente del estator (flujo magnético y par motor) el torque del motor se puede controlar independientemente. FOC proporciona un control suave en velocidades bajas al igual que un rendimiento alto en velocidades altas. Por lo que no es de extrañar que se considere la principal técnica de control usada en la industria del futuro.

2. Motores BLDC

En general los motores BLDC son cada vez más populares en sectores, como la industria automotriz o en electrodomésticos ya que, al no necesitar escobillas, que tienden al desgaste, no requieren de un mantenimiento continuo. Además, un motor BLDC puede ser más pequeño que un motor con escobillas con la misma potencia de salida por lo que los hace ideales en aplicaciones donde no se tenga mucho espacio.

Se definen como motores eléctricos AC síncronos. Esto quiere decir que el campo magnético generado por el estator y el campo magnético generado por el rotor rotan a la misma frecuencia. Además, en este tipo de motores no existe una diferencia entre la velocidad síncrona y la velocidad de funcionamiento, que se conoce comúnmente como "deslizamiento", propia de los motores de inducción. La velocidad de sincronismo es directamente proporcional a la frecuencia de línea e inversamente proporcional al número de polos pares del motor, como se puede apreciar en la siguiente fórmula:

$$n = \frac{60 * f}{P} \qquad (1)$$

Siendo

- n la velocidad de sincronismo (rpm),
- f, la frecuencia de alimentación,
- p, el número de pares de polos.

De esta fórmula, se puede observar que existen únicamente dos modos de realizar un control de velocidad en un motor síncrono: variando el número de polos pares o variando la frecuencia de alimentación. La primera opción, aparte de ser poco práctica, permite únicamente velocidades discretas, por ejemplo, para una frecuencia de 60 Hz solo se podrían conseguir velocidades de 3600, 1800, 1200, 900, etc. Para conseguir un control de velocidad más continuo se debe de actuar sobre la frecuencia de alimentación. Por lo que se precisa de algún dispositivo capaz de transformar una señal continua en otra alterna, para tal objetivo se suele usar un inversor capaz de generar una señal de excitación alterna de frecuencia deseada con la que se alimenta el motor.

2.1 Partes de un motor BLDC

Los motores BLDC, principalmente se pueden dividir en dos partes: el estator y el rotor, y dependiendo del bobinado del estator pueden presentar una configuración de una fase, dos fases o tres fases. Siendo la más común la de tres fases [1], configuración que se va a usar para el desarrollo de este trabajo. La construcción de este tipo de motor tiene muchas similitudes a los motores de inducción de tres fases al igual que a un motor DC convencional.

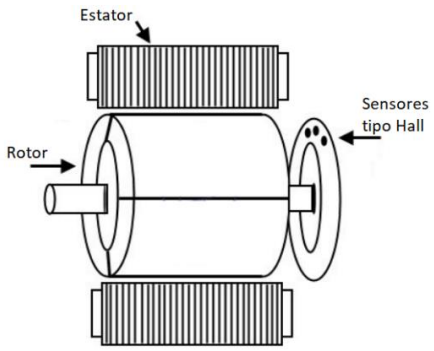


Figura 1. Estructura del motor BLDC

2.1.1 Estator

El estator constituye la parte fija del motor, está formado por unas láminas de acero, que suelen estar cubiertas por un material aislante que tiene la función de disminuir las pérdidas del motor. Además, cuentan con una serie de ranuras axiales donde se arroyan las bobinas que generan el campo magnético necesario para atraer o repeler el campo magnético creado por el rotor. Dichas ranuras dependen de las características del motor. Aunque no existe una relación entre el número de ranuras y el tamaño del motor se suele cumplir que cuanto mayor es el motor mayor es el número de ranuras. Aunque dependiendo de si es un motor de una fase, dos fases o tres fases se debe de cumplir que las ranuras sean múltiplo de 1, 2 o 3 en función de cuantas fases tenga el motor.

Si bien el estator de un motor BLDC se parece al de un motor de inducción, su bobinado se distribuye de una manera diferente. La mayoría de los motores BLDC tienen tres devanados en el estator formando una conexión tipo estrella. Cada una de estas está formada por varias bobinas que están interconectas entre ellas. Una o más bobinas están situadas en cada una de las ranuras del estator y están conectadas para formar un único devanado.

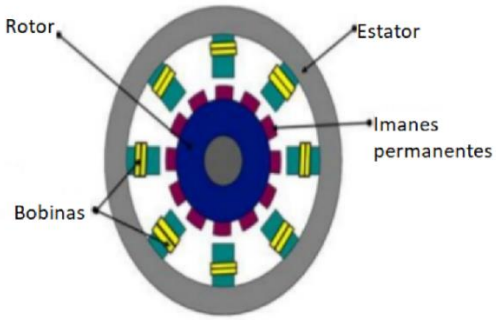


Figura 2. Partes de un motor BLDC

Existen dos variantes del devanado de un estator: trapezoidal y sinusoidal, que hace referencia a la forma de onda que tiene la fuerza contraelectromotriz del motor (back-EMF). La forma del Back-EMF se determina por el tipo arrollamiento que se hace en las bobinas y el entrehierro [2]. Además, la corriente de fase también sigue una forma de onda sinusoidal o trapezoidal. Un motor con una back-EMF sinusoidal presenta la ventaja de que produce un par electromagnético más suave que uno trapezoidal, aunque implica un mayor coste debido a que necesita un mayor número de bobinados.

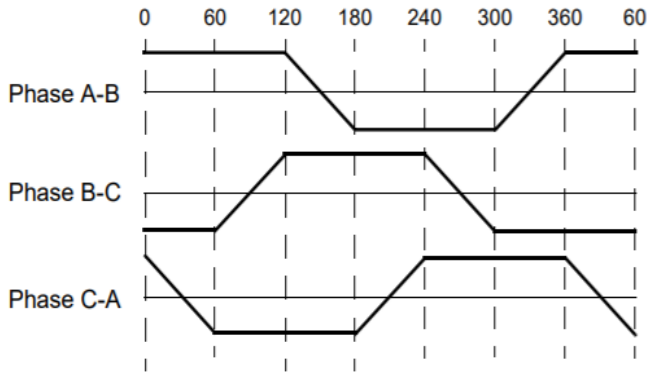


Figura 3. Back-EMF trapezoidal de un motor BLDC

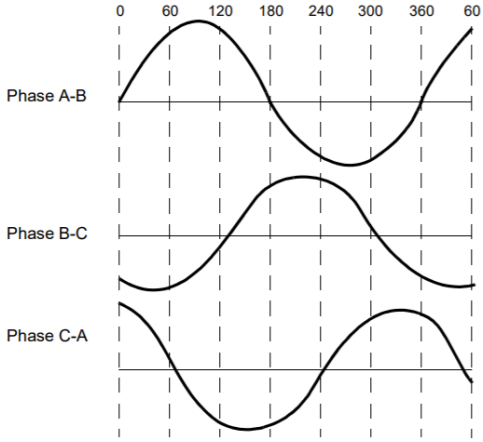


Figura 4. Back-EMF sinusoidal de un motor BLDC

2.1.2 Rotor

El rotor, como su nombre indica, constituye la parte del motor que realiza el giro y está formado por imanes permanentes con una disposición alterna de polo norte- polo sur. La posición del rotor en referencia al estator no es fija, existe la posibilidad de que el rotor se encuentre en el interior, conocido como inrunner, donde los tres devanados del estator rodean al rotor, este tipo de configuración permite que el motor gire a una mayor velocidad, llegando hasta los 11000RPM por voltio [3] pero genera menor torque. Por otro lado, se encuentra el outrunner donde es el rotor quien rodea al estator, cesta distribución aporta mayor par motor pero menor velocidad.

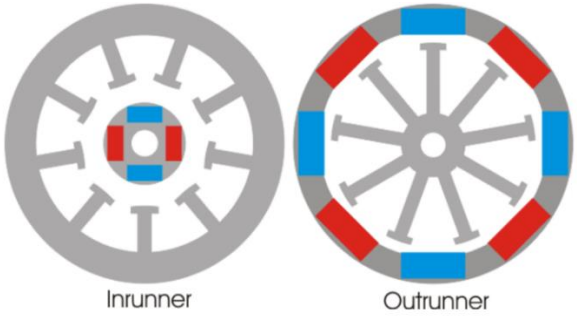
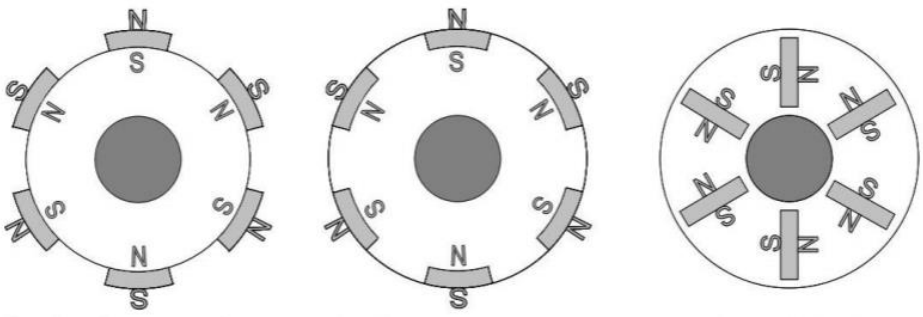


Figura 5. Motor inrunner vs motor outrunner

Un aspecto clave a la hora de construir un motor son el tipo de imanes que se van a usar ya que depende de las necesidades de densidad de campo magnético. Los imanes de ferrita son los más usados, tradicionalmente, sin embargo, a medida que la tecnología ha avanzado han aparecido nuevas aleaciones, como por ejemplo el neodimio (Nd), samario-cobalto (SmCo) o la aleación de neodimio, ferrita y boro (NdFeB). Los imanes de ferrita son menos caros, pero presentan la desventaja de que proporcionan un bajo flujo de densidad eléctrico por volumen. Por otro lado, las nuevas aleaciones tienen una mayor densidad de flujo eléctrico por volumen, lo cual permite que el rotor sea más pequeño mejorando la relación tamaño-peso y producen mayor par motor en comparación con un motor de imanes de ferrita.

En cuanto a la posición de los imanes existen varias posibilidades, sin embargo, tres de ellas son las más usadas, que son las que aparecen en la figura 6.



a) cara externa b) cara interna c) en el interior

Figura 6. Posiciones más habituales de los imanes permanentes

La configuración a) posee la característica de crear un campo magnético isotrópico, y también, por su estructura mecánica el espacio en el entrehierro suele ser mayor, lo que da lugar a menor capacidad de debilitamiento de flujo. Por otro lado, los imanes en la parte interna del motor muestran anisotropía, leve en la configuración b) y mayor en la configuración c), y el hueco en el entrehierro suele ser menor.

2.2 Principio de funcionamiento

Los motores eléctricos transforman potencia del marco eléctrico al marco mecánico usando la interacción magnética. En los motores BLDC, está interacción magnética ocurre entre los cables de las bobinas del estator y los imanes permanentes del rotor.

En un motor BLDC en cada secuencia de conmutación, ver figura 7, se tiene uno de los devanados energizados con tensión positiva (la corriente entra en la bobina), una segunda bobina negativa (la corriente sale de esta bobina) y una tercera bobina no energizada. En consecuencia, en la bobina con tensión positiva se genera un campo magnético, formándose de este modo un electroimán, que da lugar a un par de giro sobre la parte móvil, y se produce la rotación del eje del motor. Para que la rotación continúe es necesario conmutar la excitación de las bobinas, de esta forma se consigue que el campo magnético asociado a los imanes permanentes persiga al campo magnético generado por el estator.

Idealmente el pico de par motor se produce cuando los dos campos se encuentran a 90º uno respecto al otro y es mínima cuando se encuentran alineados, en el capítulo 4 se dará una explicación más detallada. Con el objetivo de mantener el motor girando

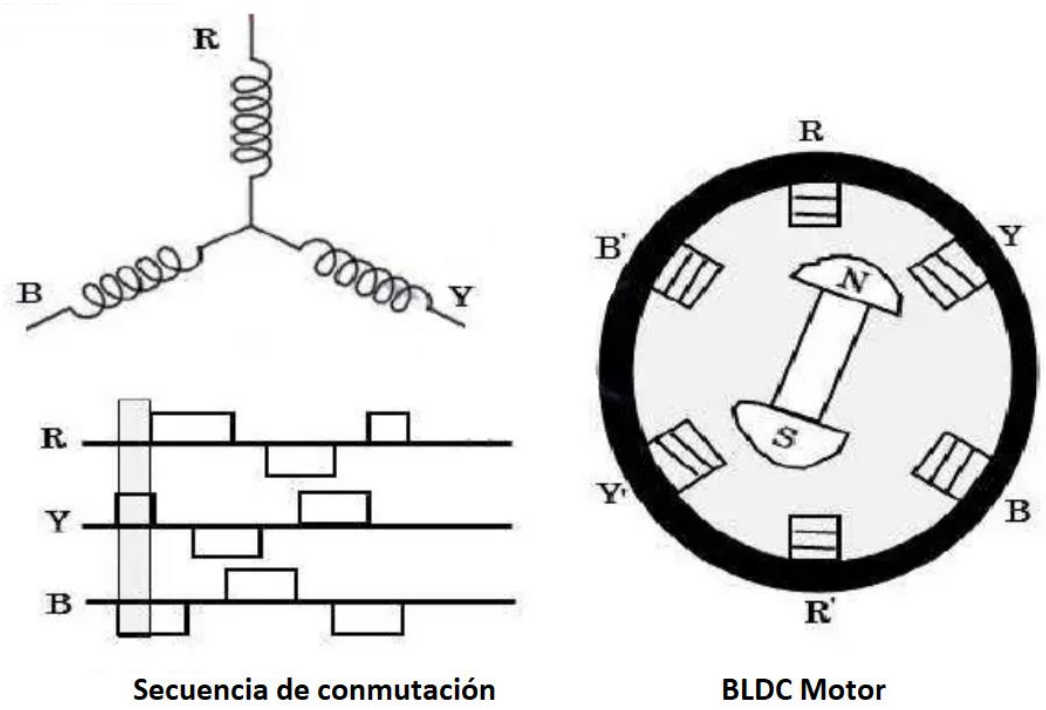


Figura 7. Principio de funcionamiento de un motor BLDC

2.3 Modelo matemático

El modelo matemático de cada devanado del motor es expresado de la siguiente forma [4]:

$$v_a = Ri_a + L\frac{di_a}{dt} + e_a \quad (2)$$

$$v_b = Ri_b + L\frac{di_b}{dt} + e_b \quad (3)$$

$$v_c = Ri_c + L\frac{di_c}{dt} + e_c \quad (4)$$

donde: L es la inductancia de la armadura [H],

R – la resistencia de la armadura [Ω],

V_a, V_b, V_c - voltaje de la fase de la terminal [V],

i_a, i_b, i_c - corriente de salida del motor [A],

y e_a, e_b, e_c - la back-EMF del motor [V].

El circuito equivalente para una de las fases es representado en la figura 8.

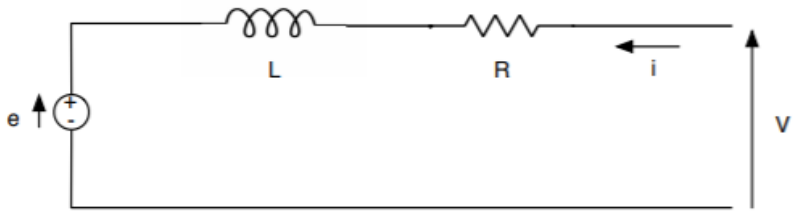


Figura 8. Circuito equivalente de una de las fases del motor BLDC

En las tres fases del motor, el back-EMF es función de la posición del rotor y existe un desfase de 120º de ángulo entre cada una de las fases del motor, sus ecuaciones son:

$$e_a = K_e f(\theta_e)\omega \quad (5)$$

$$e_b = K_e f(\theta_e - 120º)\omega \quad (6)$$

$$e_c = K_e f(\theta_e + 120º)\omega \quad (7)$$

donde K_e es la constante de la back-EMF [V/m - RPM],

θ_e - es el ángulo eléctrico del rotor [e - grados],

ω - la velocidad del rotor [m - RPM]

La constante de la back-EMF se expresa en voltios por mecánica revolución por minuto. La m- y la e- antes de cada unidad significan respectivamente mecánica y eléctrica, haciendo referencia a ángulos mecánicos y eléctricos los cuales son relacionados por el número de polos pares del motor.

El ángulo eléctrico del rotor es igual al ángulo mecánico del rotor multiplicado por el número de polos pares p:

$$\theta_e = p\theta_m \quad (8)$$

donde θ_m ,el ángulo mecánico del rotor [m - grados].

El torque total de salida puede representarse como una suma de las tres fases. La siguiente ecuación representa el par motor de salida total, o el torque electromagnético:

$$T_e = \frac{e_a i_a + e_b i_b + e_c i_c}{\omega} = K_T \frac{3}{2} i_q \qquad (9)$$

donde T_e es el par motor total de salida [Nm],

K_T –constante del motor [Nm/A],

I_q – la corriente de cuadratura [A].

La ecuación de la parte mecánica se representa como:

$$T_e - T_l = J \frac{d\omega}{dt} + T_f \qquad (10)$$

donde T_l es la carga de torque [Nm],

J – la inercia del rotor [kgm^2],

T_f – la fricción del torque [Nm].

2.4 Relación velocidad/par motor

La figura 9 muestra la relación entre el par motor y la velocidad. Existen dos parámetros de par motor utilizados para definir un motor BLDC: el pico de torque (T_P) y el par nominal (T_R). Tal como se observa en la figura, en funcionamiento continuo, el motor es cargado con el par nominal manteniéndose constante hasta la velocidad nominal. Sin embargo, El motor es capaz de funcionar a velocidades superiores a la nominal, pudiendo llegar a funcionar hasta el 150% de la misma, con la consecuente disminución del par.

En aplicaciones en las que aparecen arranques y paradas de manera frecuente y cambios habituales de rotación con carga en el motor, la demanda superará el par nominal. Esto se debe a que durante un breve periodo de tiempo (especialmente cuando el motor arranca desde parado y durante una aceleración) es necesario disponer de un par adicional para poder superar la inercia de la carga y el propio rotor.

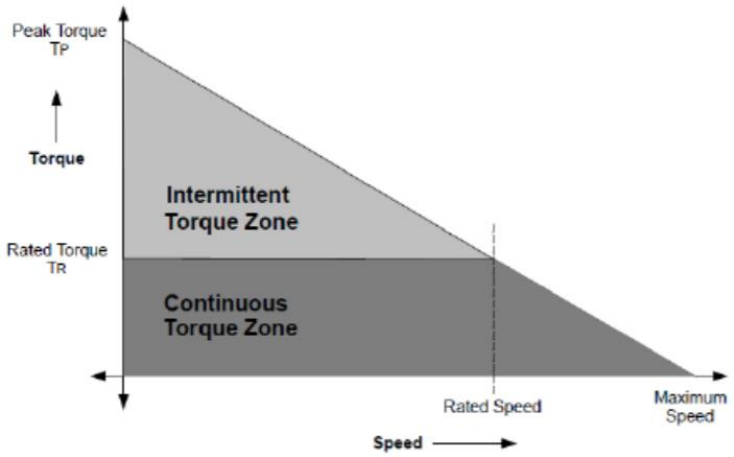


Figura 9. Relación par/velocidad de un motor sin escobillas

3. Técnicas de control

Los motores sin escobillas necesitan de un elemento de control que les proporcione la corriente necesaria a cada uno de los devanados del motor. Existen varias técnicas de control de los motores sin escobillas encargas de realizar dicha función, su objetivo es determinar la excitación óptima de cada una de las fases del motor con la finalidad de alcanzar los parámetros de funcionamiento determinados.

Hay varias técnicas de control cuyo grado de complejidad varia. Los métodos más sencillos se encargan, únicamente, de conmutar las fases del motor accionadas, mientras que otras más complejas pueden incluir un sistema de control de tensión o corriente a través de una realimentación en lazo cerrado. Generalmente, cuanto mayor es la complejidad del algoritmo de conmutación mejor es el control que hay sobre el motor, en consecuencia, se consiguen mejores resultados.

A continuación, se tratarán las tres técnicas más usadas hoy en día: Conmutación Trapezoidal, Conmutación Sinusoidal y el Control Vectorial.

3.1 Conmutación trapezoidal

Uno de los métodos más simples se usa con los motores dc sin escobillas es el llamado Conmutación Trapezoidal. Con esta técnica la corriente es controlada a través de los terminales del motor, uno por uno, con el tercer terminal desconectado eléctricamente de la fuente de tensión. Tres sensores de tipo Hall situados en el motor son usados para proporcionar una señal digital que mide la posición del rotor y proporcionan esa información al controlador del motor. Debido a que en cualquier instante las corrientes en dos de los devanados son iguales en magnitud y el tercero es cero, este método solo puede producir vectores de corriente en el espacio teniendo una de las seis posiciones existentes.

A medida que el motor gira la corriente de los terminales del motor conmutan cada 60º de su rotación, logrando de este modo que el vector se aproxime siempre a 30º de la dirección de cuadratura, dando como resultado un vector de corriente en el espacio de una rotación suave [5].

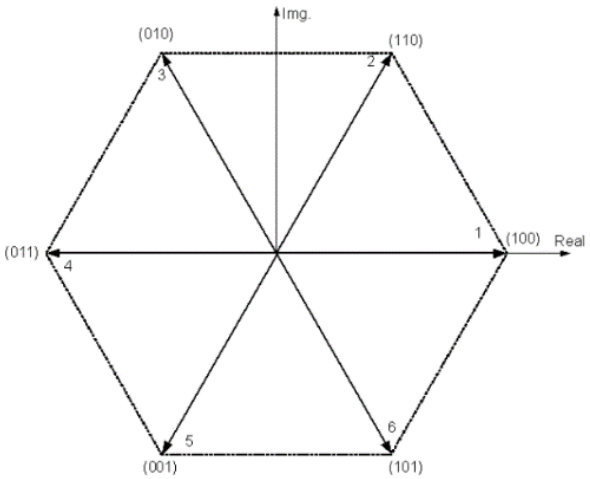


Figura 10. Conmutación trapezoidal

Un diagrama de bloque del método de conmutación trapezoidal se muestra en la figura 11.
Un controlador PI es utilizado para el control de corriente. El torque deseado es comparado con la corriente medida para producir la señal de error. El error es integrado y amplificado para producir una salida, cuyo objetivo es reducir ese error. La salida del controlador PI se modula posteriormente en el PWM y proporciona una salida hacia el puente.

La conmutación se realiza de manera independiente al control de corriente. Las señales de posición procedentes de los sensores Hall del motor son usados para seleccionar los terminales que deben de energizarse.

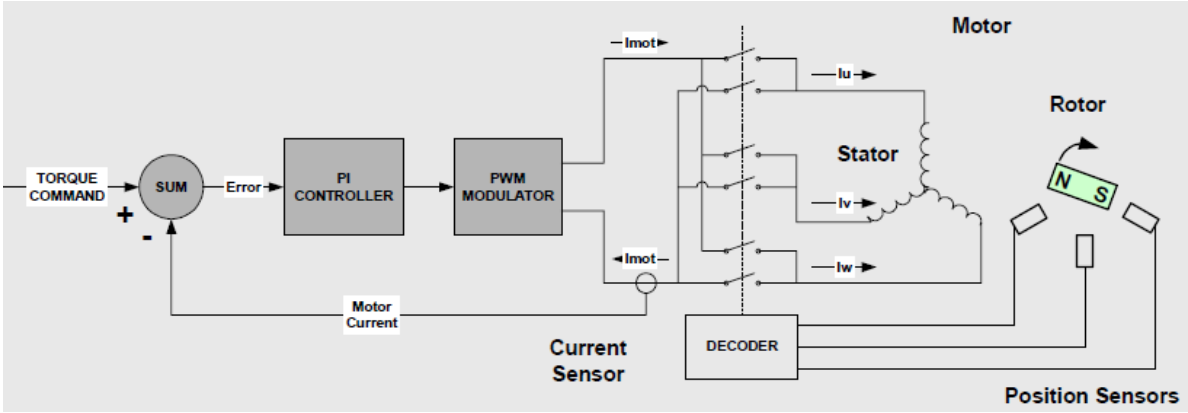


Figura 11. Circuito simplificado de control trapezoidal de un motor BLDC

Dado que el vector de corrientes solo puede apuntar a una de las seis direcciones se produce una desalineación entre éstas y la posición real del rotor. En el peor caso, es decir cuando el rotor se encuentra en la posición intermedia de uno de los 6 sectores, la desalineación puede llegar a ser de 30º. Generando un rizado en el par motor de uno 15% a una frecuencia seis veces de la velocidad de rotación del motor. Esto a su vez genera pérdidas en la eficiencia, ya que cierta parte de la corriente del motor no se traduce en par motor.

3.2 Conmutación sinusoidal

La conmutación trapezoidal no es la técnica de control adecuada cuando se busca un control preciso de un motor, especialmente a velocidades bajas. La conmutación sinusoidal soluciona este problema, ya que intenta controlar la posición del rotor continuamente.

Esta continuidad se consigue aplicando simultáneamente tres corrientes sinusoidales desfasadas 120º a las tres bobinas del motor. La fase de estas corrientes se escoge de forma que el vector de corrientes resultante siempre este en la cuadratura con la orientación del rotor y que tenga un valor constante. Para conseguir tal objetivo se precisa conocer de forma exacta la posición del rotor, por lo que los sensores Hall no son adecuados ya que solo proporcionan una medida aproximada, por lo tanto, se necesitan otro tipo de sensores, como por ejemplo los encoders, que sí dan una medida precisa de la posición del rotor.

En la figura 12 se puede apreciar un diagrama de bloques de control sinusoidal. Este esquema usa un bucle de corriente separada para cada uno de los dos bobinados del motor.

Desde que el motor tiene una configuración estrella y siguiendo la ley de Kirchoff se puede conocer la corriente de la tercera bobina.

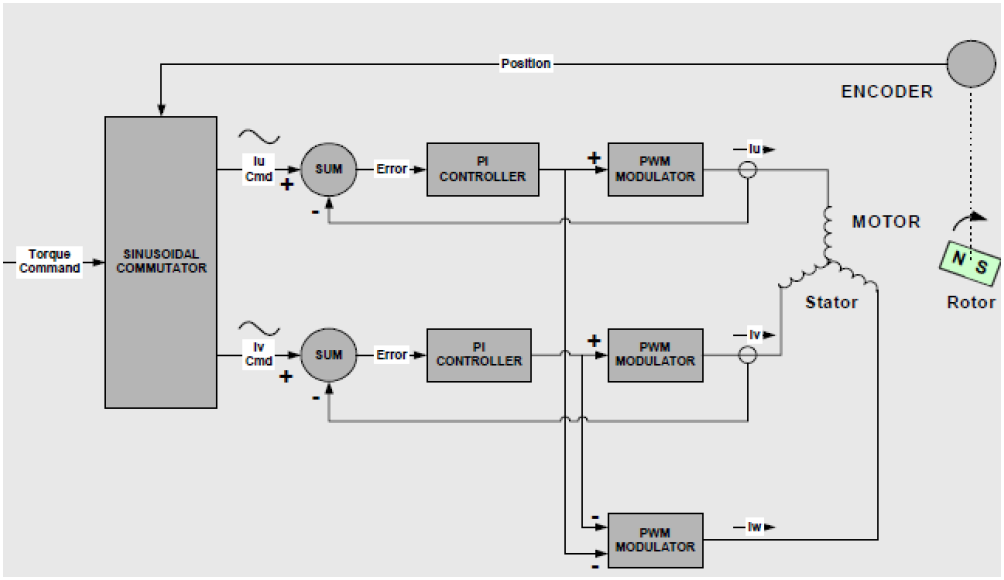


Figura 12. Diagrama de un control sinusoidal de un motor BLDC

Gracias a la información de la posición del rotor, se sintetizan las dos ondas sinusoidales deseadas, normalmente mediante el uso de una Look-Up-Table, LUT, (Técnica computacional que permite la optimización del tiempo de cálculo de una función mediante la sustitución de ésta por la consulta a un array). Estas son comparadas con las medidas de las corrientes que circulan por el motor y el error resultante se aplica a dos controladores PI que intentan corregir las desviaciones. La salida de cada uno de los controladores PI son introducidas a un modulador PWM y después de pasan a los dos terminales del motor. El voltaje aplicado al tercer terminal es el resultado del valor negativo de la suma de los otros dos terminales.

La conmutación sinusoidal resuelve los problemas de eficiencia que presenta la conmutación trapezoidal. Pero, presenta problemas a altas velocidades debido a la limitación de frecuencia del bucle de corriente. Cuanto más aumenta la velocidad, mayor error y por lo tanto mayor desalineación entre el vector de corrientes y la dirección de cuadratura del rotor. En consecuencia, se produce una disminución progresiva del par motor.

Para poder mantener el par constante se necesita aumentar la corriente del motor produciendo una disminución de la eficiencia. Este deterioro aumenta al incrementarse la velocidad hasta llegar a un punto en el que el desfase entre el vector de corrientes y la dirección de cuadratura puede llegar a 90º produciendo un par motor nulo.

3.3 Control vectorial

El principal problema de la conmutación sinusoidal es que intenta controlar directamente las corrientes que circulan por el motor, las cuales varían con el tiempo. Al aumentar la velocidad del motor, y en consecuencia la frecuencia de las corrientes, empiezan a aparecer problemas, ya que los controladores PI tienen uno ancho de banda límite.

El control vectorial o también conocido como Control de Campo Orientado (FOC) soluciona el problema controlado el vector de corrientes directamente en un espacio de referencia ortogonal y rotacional llamado espacio D-Q (Directo-Cuadratura). Debido a que el vector de corrientes en el espacio de referencia D-Q es estático, los controladores PI trabajan en continua, en vez de señales senoidales. Esto permite aislar a los controladores de la variación en el tiempo de las corrientes y tensiones de las bobinas y, por lo tanto, elimina la limitación de los controladores de la frecuencia de respuesta y el desfase de fase en el torque y velocidad del motor.

Para poder realizar este control es necesario transformar matemáticamente las medidas de las corrientes de los tres devanados del sistema de referencia estator al sistema de dos ejes rotativos del rotor. Una vez que se han hecho las transformadas, se aplican dos controladores PI que se utilizan para controlar la componente directa y la cuadratura de forma independiente.

La componente en cuadratura es la única que proporciona par útil, por tanto, la directa suele fijarse a cero, se explicará con más detalle en el capítulo 4. De este modo, se consigue maximizar la eficiencia del sistema, ya que se fuerza al sistema en la dirección de la componente de la cuadratura.

Posteriormente se realizan una serie de transformadas para regresar al espacio estacionario de las bobinas y se energiza cada una de las fases mediante modulación.

El diagrama de bloque de un FOC se puede ver en la figura 13, donde se muestran cada uno de los pasos que se debe de seguir:

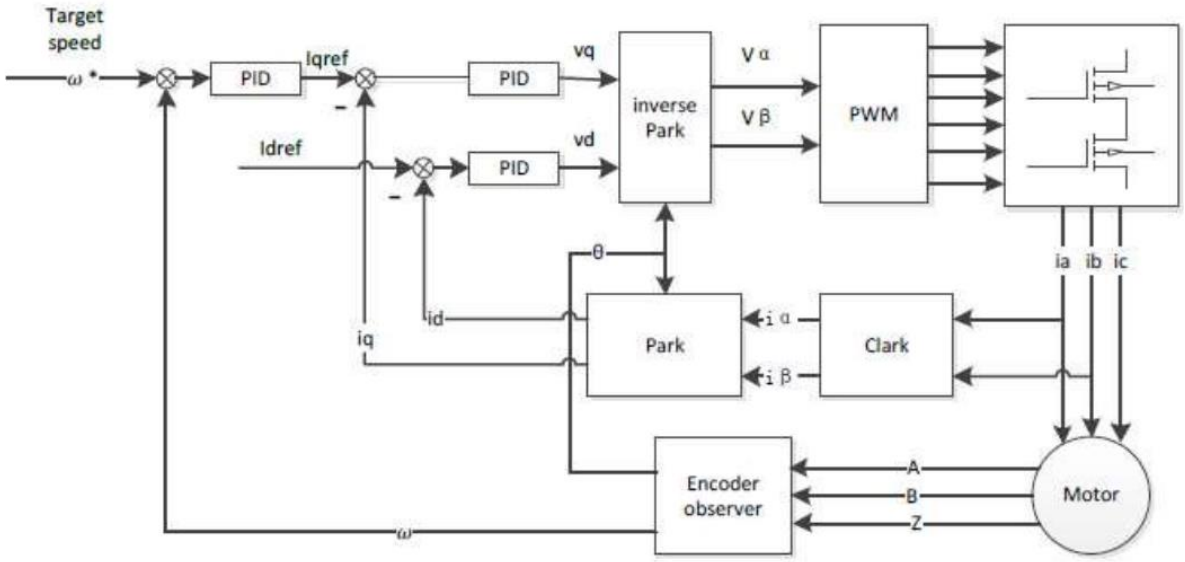


Figura 13. Diagrama de bloque de FOC

La característica importante que diferencia el control vectorial de la conmutación sinusoidal son los procesos de la secuencia de conmutación y el control de corriente. En la conmutación sinusoidal, se realiza la conmutación en primer lugar y luego se aplica el controlador PI de la corriente sinusoidal resultante. Los controladores PI en un sistema sinusoidal son, por lo tanto, expuestos a la variación de tiempo en las corrientes y voltajes del motor y el motor tiene un ancho de banda limitado y el desfase de fase de los controladores. En el Control de Campo Orientado, el control PI de corriente se realiza en primer lugar y después es seguido por el proceso de conmutación. En consecuencia, los controladores PI están aislados de las variaciones de tiempo de la corriente y el voltaje y, además no están limitados en frecuencia.

3.4 Elección de la técnica de control

La conmutación trapezoidal proporciona una primera aproximación al control de motores sin escobillas. Gracias a su sencillez y los pocos recursos que necesita, permite que sea utilizado en varias aplicaciones, donde no sea necesario un control preciso de velocidad o posición. Sin embargo, debido al alto rizado que produce no es aconsejable usarlo en aplicaciones donde se requiera una alta eficiencia.

La conmutación sinusoidal resuelve el problema del rizado producido en el par motor a cambio de aumentar la complejidad del control y de colocar un sensor mucho más preciso. Pero, debido a que trabaja en un espacio donde el tiempo varia, presenta una limitación de velocidad.

Por otro lado, el control vectorial soluciona los problemas de las dos técnicas obteniendo una alta eficiencia y control, tanto a velocidades bajas como altas. En consecuencia, se ha elegido este último sistema de control para el desarrollo del presente trabajo.

4. Control de campo orientado (FOC)

4.1 Definición de FOC

El Control de Campo Orientado (FOC), es un método de control que es usado para dirigir un inversor de frecuencia variable (VFD), cuyo objetivo es desacoplar las componentes de flujo y par de un motor BLDC, facilitando el control de este tipo motores.

A través de esta técnica las corrientes del estator son transformadas a un sistema ortogonal de dos componentes, permitiendo ver dichas corrientes como un vector, de ahí que también se denomine control vectorial. Si, además, este sistema ortogonal está constantemente girando con una velocidad angular igual a la frecuencia síncrona de la máquina se puede separar, el vector, en sus componentes reales e imaginarios, normalmente uno proporcional al par generado y el otro proporcional al flujo del motor, que puede ser del rotor, estator o del entrehierro [6]. Para este proyecto el flujo del rotor es considerado como uno de los componentes.

Como consecuencia de que el vector se encuentre en un sistema ortogonal que gira de forma constante a una determinada velocidad constante, sus componentes darán la sensación de que son constantes. Bajo esta perspectiva se pueden obtener dos componentes del fasor de corriente, de manera que cada componente controla independientemente el par y el flujo del motor. La particularidad del control vectorial en los motores sin escobillas es que la corriente del flujo del estator es establecida a cero. Los imanes en el rotor producen flujo de enlace ψ_m, a diferencia de un motor AC el cual necesita un corriente de flujo positiva para generar la corriente de magnetización y por lo tanto producir el flujo de enlace del rotor.

El flujo generado en el entrehierro es el resultado de la suma de los flujos del rotor y del estator. En un motor BLDC el flujo de enlace del rotor es generado por los imanes permanentes y el flujo del estator por las corrientes que circula por las bobinas. Por debajo de la velocidad nominal, no se genera el flujo de enlace del estator ya que su valor se ha establecido a cero y por lo tanto el flujo del entrehierro es igual al flujo del rotor. Sin embargo, por encima de la velocidad nominal el valor del

flujo del estator es puesta a un valor negativo, el flujo del estator se opone al flujo del rotor, dando lugar a un debilitamiento del flujo del entrehierro.

FOC puede ser implementado usando un sensor de velocidad o puede realizarse sin sensores. Para un control preciso es preferible el control con sensor. En un FOC con sensor, la posición del rotor y la velocidad mecánica del son determinados usando un encoder o resolver. Para este proyecto se usará un encoder incremental.

Para realizar un control FOC se deben de seguir los siguientes pasos:

- En primer lugar, se deben de medir dos de las tres corrientes del motor, ya que siguiendo la ley de Kirchoff se puede calcular la corriente de la tercera fase.

$$I_a + I_b + I_c = 0 \qquad (11)$$

- Las tres corrientes son transformadas al sistema estacionario de dos ejes. Esta conversión permite obtener las variables I_α e I_β a partir de las corrientes medidas en el estator. Los valores I_α e I_β son valores de corriente de cuadratura que varían con el tiempo vistos desde la perspectiva del motor.

- El sistema estacionario de dos ejes es rotado para alienarse con el flujo del rotor usando la transformada del ángulo realizado en el anterior bucle de control. Mediante está conversión se obtienen los valores I_d e I_q. Estos valores son corrientes de cuadratura que ha sido transformadas al sistema de coordenadas rotatorio.

- Las variables I_d e I_q se comparan con los valores de referencia que son:
  - I_d de referencia: controla el flujo magnético del rotor.
  - I_q de referencia: controla el par de salida del motor.

- El error obtenido de comparar estas variables es introducido en los controladores PI. La salida de los controladores genera dos vectores de voltaje V_d y V_q los cuales serán aplicados al motor.

- Mediante los valores V_d y V_q, obtenidos de los controladores PI, se realiza una transformada inversa para volver al sistema de referencia estacionario. A partir de este paso se obtienen los valores V_α y V_β.

- Los valores V_α y V_β son usados para calcular el ciclo de trabajo de los PWMs los cuales generarán el voltaje deseado.

- Se realiza una nueva medición del ángulo eléctrico del motor a partir de los pulsos del encoder.

4.2 Principios de las transformadas de coordenadas

A través de una serie de transformaciones de coordenadas, se puede llegar a determinar y controlar indirectamente los valores de par motor y flujo, invariantes en el tiempo, con un controlador PI. En este apartado se comentarán cada una de las transformadas que hacen posible controlar las dos variables.

4.2.1 Transformada de Clarke

La transformada de Clarke convierte el sistema de corrientes trifásico, con referencia el estator, en un sistema de coordenadas de dos ejes manteniendo la misma referencia.

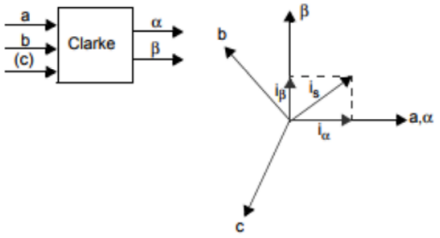


Figura 14. Transformada de Clarke

$$i_a + i_b + i_c = 0 \qquad (12)$$

$$i_a = i_\alpha \qquad (13)$$

$$i_\beta = \frac{i_a + 2i_b}{\sqrt{3}} \qquad (14)$$

4.2.2 Transformada de Park

La transformada de Park convierte el sistema de dos ejes, con referencia el estator, a un sistema rotatorio de dos ejes que gira en sincronía con el flujo del rotor.

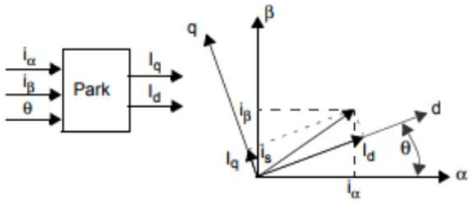


Figura 15. Transformada de Park

$$i_d = i_\alpha \cos(\theta) + i_\beta \sin(\theta) \qquad (15)$$

$$i_q = -i_\alpha \sin(\theta) + i_\beta \cos(\theta) \qquad (16)$$

4.2.3 Transformada inversa de Park

La transformada inversa de Park convierte el sistema de referencia de dos ejes giratorio, referenciado al rotor, en un sistema estacionario de dos ejes.

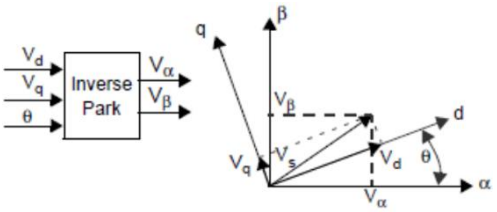


Figura 16. Transformada inversa de Park

$$v_\alpha = v_d \cos(\theta) - v_q \sin(\theta) \qquad (17)$$

$$v_\beta = v_d \sin(\theta) + v_q \cos(\theta) \qquad (18)$$

4.2.4 Transformada inversa de Clarke

La transformada inversa de Clarke convierte el sistema de referencia de dos ejes, estacionario, a un sistema de tres ejes, con la misma referencia.

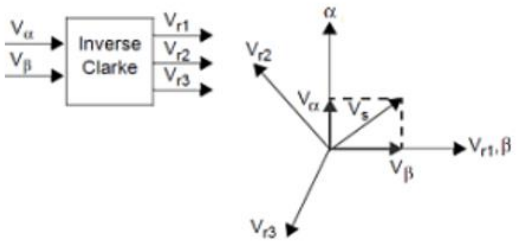


Figura 17. Transformada inversa de Clarke

$$v_{r1} = v_\beta \qquad (19)$$

$$v_{r2} = \frac{-v_\beta + \sqrt{3}v_\alpha}{2} \qquad (20)$$

$$v_{r3} = \frac{-v_\beta - \sqrt{3}v_\alpha}{2} \qquad (21)$$

4.3 Modulación de espacio vectorial

El control vectorial suele ir acompañado por su propia técnica de modulación, llamada SVPWM. En este apartado se va a profundizar en esta técnica para dar una idea de cómo funciona.

La modulación de espacio vectorial consiste en la aplicación de una combinación de estados de conmutación básicos durante un tiempo concreto para generar un vector resultante idéntico al vector de tensión deseado.

El modelo típico de un inversor trifásico PWM se muestra en la figura 18. Donde existen seis interruptores, S1 - S6, que están controlados con las variables a, a', b, b', c y c'. Cuando uno de los interruptores de la rama de arriba está encendido, es decir que tiene un nivel lógico de 1, el interruptor correspondiente de la rama de abajo está apagado, esto es, que tiene un 0. Por lo tanto, controlando el estado de los interruptores de arriba S1, S3 y S5 podemos controlar el voltaje de salida.

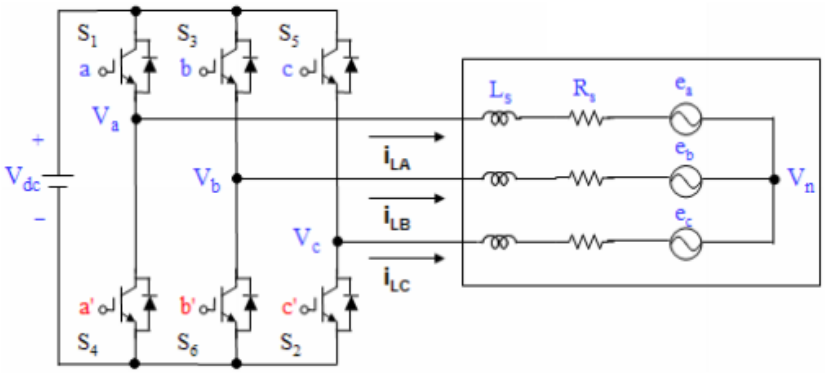


Figura 18. Inversor trifásico PWM

En consecuencia, se obtiene 8 posibles estados, $2^3 = 8$. Sin embargo, los estados en los que los interruptores S1, S3 y S5 están conectados a V_{dc} o a masa se consideran nulos, ya que no hay diferencia de potencial entre los bobinados. Los 8 posibles se representan como vectores de voltaje divididos en seis sectores dando como resultado una distribución tipo de estrella, como se observa en la figura 19, donde los estados nulos se encuentran en el centro. La distribución de los vectores se ha realizado con el propósito de que la diferencia entre dos vectores adyacentes sea de tan solo 1 bit, por lo que

cambiando únicamente el estado de un de los interruptores el vector de referencia se puede mover de un sector a otro, reduciéndose de este modo las pérdidas por conmutación.

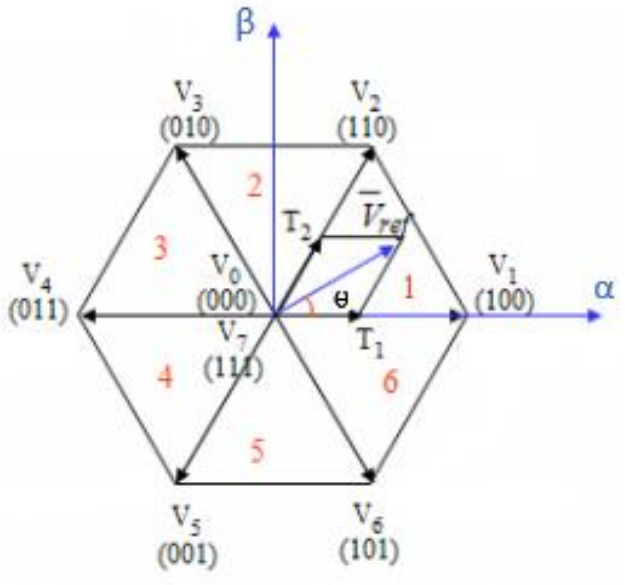


Figura 19. Representación espacial de los 8 vectores

Para implementar la modulación de espacio vectorial, el primer paso consiste calcular el voltaje de referencia.

Después de aplicar la transformada inversa de Park obtenemos dos vectores V_α y V_β, a partir de estos dos valores calculamos su módulo y ángulo:

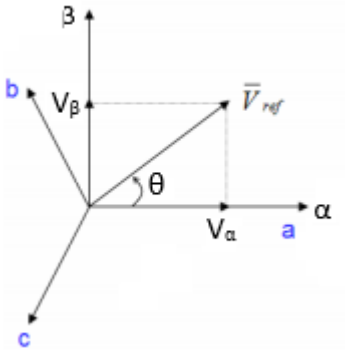


Figura 20. Componentes Vα y Vβ

$$|V_{ref}| = \sqrt{V_\alpha^2 + V_\beta^2} \tag{22}$$

$$\theta = tam^{-1}\left(\frac{V_\beta}{V_\alpha}\right) = \omega t = 2\pi f t \; donde \; f \; es \; la \; frecuencia \; fundamental \tag{23}$$

Una vez calculado el voltaje de referencia y su ángulo, se conoce el sector en el que se encuentra y en consecuencia se escogen los dos vectores adyacentes. Por ejemplo, si el voltaje de referencia estuviera en el sector 1, los vectores adyacentes serian S1 y S2.

El siguiente paso consiste en calcular el tiempo de apertura de cada uno de los vectores, para ello tenemos las siguientes ecuaciones:

$$T_1 = \frac{\sqrt{3}\cdot T_s \cdot \left|V_{ref}\right|}{V_{dc}} \sin\left(\frac{n}{3}\pi - \alpha\right) \tag{24}$$

$$T_2 = \frac{\sqrt{3}\cdot T_s \cdot \left|V_{ref}\right|}{V_{dc}} \sin\left(\alpha - \frac{n-1}{3}\pi\right) \tag{25}$$

$$T_0 = T_s - (T_1 + T_2) \tag{26}$$

donde n es el sector, α el ángulo entre $0 \leq \alpha \leq \pi/3$ y T_1, T_2 y T_0 son los tiempos de cada uno de los vectores de voltaje. T_s es el tiempo de conmutación.

Una vez calculados los tiempos de apertura de cada uno de los vectores, mediante la siguiente tabla se determina a que vectores se debe de aplicar cada tiempo de apertura:

Sector	S_1, S_3, S_5	S_4, S_6, S_2
1	$S_1 = T_1 + T_2 + T_0/2$ $S_3 = T_2 + T_0/2$ $S_5 = T_0/2$	$S_4 = T_0/2$ $S_6 = T_1 + T_0/2$ $S_2 = T_1 + T_2 + T_0/2$
2	$S_1 = T_1 + T_0/2$ $S_3 = T_1 + T_2 + T_0/2$ $S_5 = T_0/2$	$S_4 = T_2 + T_0/2$ $S_6 = T_0/2$ $S_2 = T_1 + T_2 + T_0/2$
3	$S_1 = T_0/2$ $S_3 = T_1 + T_2 + T_0/2$ $S_5 = T_2 + T_0/2$	$S_4 = T_1 + T_2 + T_0/2$ $S_6 = T_0/2$ $S_2 = T_1 + T_0/2$
4	$S_1 = T_0/2$ $S_3 = T_1 + T_0/2$ $S_5 = T_1 + T_2 + T_0/2$	$S_4 = T_1 + T_2 + T_0/2$ $S_6 = T_2 + T_0/2$ $S_2 = T_0/2$
5	$S_1 = T_2 + T_0/2$ $S_3 = T_0/2$ $S_5 = T_1 + T_2 + T_0/2$	$S_4 = T_1 + T_0/2$ $S_6 = T_1 + T_2 + T_0/2$ $S_2 = T_0/2$
6	$S_1 = T_1 + T_2 + T_0/2$ $S_3 = T_0/2$ $S_5 = T_1 + T_0/2$	$S_4 = T_0/2$ $S_6 = T_1 + T_2 + T_0/2$ $S_2 = T_2 + T_0/2$

Tabla 1. Tiempo de conmutación en cada sector

Los PWMs generados mediante está técnica son simétricos respecto al centro del periodo de cada PWM. Por ejemplo, en el caso de que el vector de referencia se encontrará en el sector 1 los PWMs generados tendrían la siguiente forma, la cual se muestra en la figura 21:

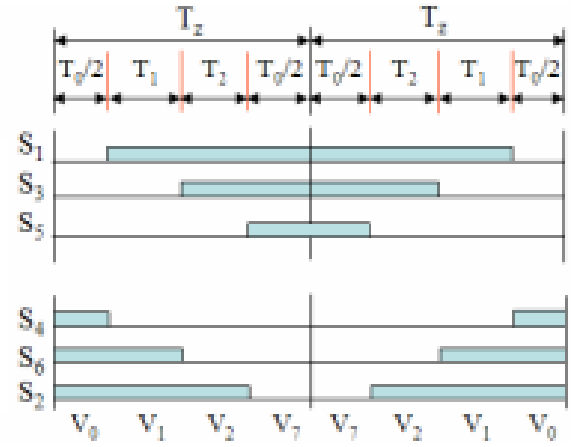


Figura 21. Formas de onda en el sector 1

La modulación de espacio vectorial permite la composición de cualquier vector del espacio por medio de la suma de dos de los vectores adyacentes. Además, presenta menos distorsión de harmónicos en la salida y utiliza de una forma mucho más eficiente la fuente de tensión comparada a la técnica de modulación sinusoidal. [8]

4.3.1 Comparación de SVM con la modulación sinusoidal

A parte de la modulación de espacio vectorial también existe la posibilidad de generar 3 señales sinusoidales desfasadas 120º, obtenidas a través de la transformada inversa de Clarke y mediante una señal triangular generar las señales PWMs, técnica conocida como modulaciónde ancho de pulso sinusoidal (SPWM), correspondientes a la apertura y cierre de los interruptores, tal como se observa en la figura 22.

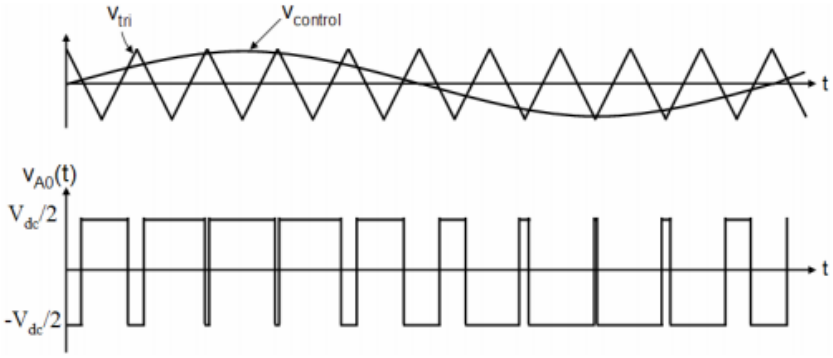


Figura 22. Modulación sinusoidal

La modulación se logra empleando una señal de control sinusoidal $V_{control}$ a la frecuencia de salida deseada f_1, que es comparada con una onda portadora triangular para generar las señales de disparo. La frecuencia de la forma de onda triangular establece la frecuencia de conmutación del inversor f_s. La relación de amplitud es:

$$m_a = \frac{V_{control}}{V_{tri}} \qquad (27)$$

Donde $V_{control}$ es la amplitud pico de la señal de control y V_{tri}, es el pico de la onda portadora triangular.

La relación de modulación de frecuencia es:

$$m_f = \frac{f_s}{f_1} \tag{28}$$

La principal ventaja de la modulación SVM es que genera menos harmónicos de distorsión en la corriente de los bobinados de cada una de las fases del motor. Además, permite usar la fuente de tensión de una manera mucho más eficiente en comparación con la modulación sinusoidal.

Mediante la técnica de modulación de SVPWM, la longitud de cada vector espacial discreto es de $\frac{2}{3}V_{DC}$. Cada lado del punto medio del hexágono es tangencial al círculo inscrito. La mayor magnitud posible del voltaje de referencia se puede calcular como:

$$OM = OL * cos\left(\frac{\pi}{6}\right) = \frac{2}{3}V_{dc} * \frac{\sqrt{3}}{2} = \frac{V_{dc}}{\sqrt{3}} \tag{29}$$

Mientras que en la modulación de ancho de pulso sinusoidal el vector de referencia se encuentra en un círculo de radio $\frac{1}{2}V_{DC}$, ya que como NM es perpendicular a OL se tiene que:

$$ON = OM * cos\left(\frac{\pi}{6}\right) = \frac{V_{dc}}{\sqrt{3}} * \frac{\sqrt{3}}{2} = \frac{V_{dc}}{2} \tag{30}$$

Por lo tanto, la máxima tensión de salida del SVM teóricamente es de $\frac{2}{\sqrt{3}}(\frac{OM}{ON})$ veces superior a la técnica de SPWM habitual. En consecuencia, mediante el SVM se puede sintetizar una onda fundamental de amplitud 15,47% mayor a la obtenida con un SPWM convencional.

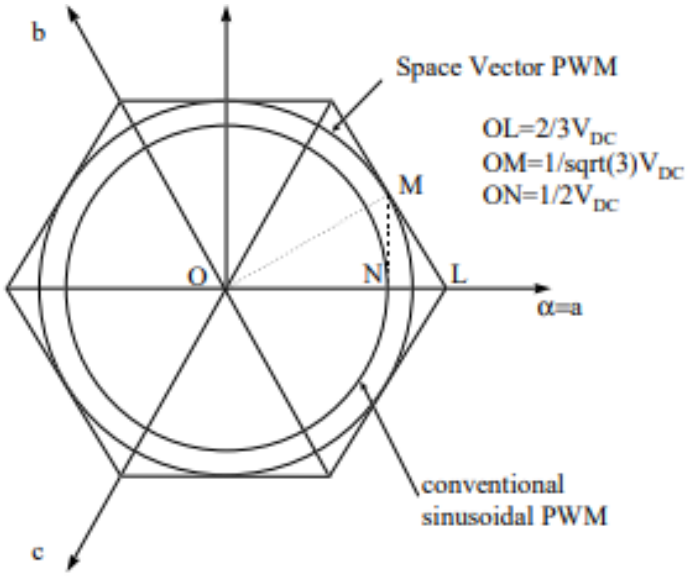


Figura 23. Comparación entre la modulación sinusoidal y SVM

4.4 Controladores PI

La función de un controlador PI consiste en responder a una señal de error en un bucle de control cerrado e intentar ajustar la cantidad controlada para lograr la respuesta deseada del sistema. El parámetro controlado puede ser cualquier cantidad medible del sistema, como puede ser la velocidad o el flujo. La ventaja de estos controladores es que se pueden ajustar empíricamente variando uno o más valores de ganancia y observando el cambio en la respuesta del sistema.

Un controlador PI digital se ejecuta en un intervalo de muestreo periódico. Se supone que el controlador se ejecuta con frecuencia, por lo tanto, el sistema puede ser controlado. La señal de error se forma restando el valor deseado del parámetro a controlar del valor real medido. El signo del error indica la dirección del cambio requerida por la entrada del control.

El término Proporcional (P) del controlador se forma multiplicando la señal de error por una ganancia 'P', haciendo que el controlador PI produzca una respuesta de control, que es una función de la magnitud del error. A medida que la señal de error se hace más grande, el término 'P' del controlador se hace más grande para proporcionar una mayor corrección.

El efecto del término 'P' tiende a reducir el error a medida que transcurre el tiempo. Sin embargo, el efecto del término 'P' disminuye a medida que el error se acerca a cero. En la mayoría de los sistemas, el error del parámetro controlado se acerca a cero, pero no converge. El resultado es un pequeño error en el estado estacionario.

El término Integral (I) del controlador se utiliza para eliminar los pequeños errores en el estado estacionario. El término 'I' responde a un error acumulado en el tiempo. Por lo tanto, un pequeño error en el estado estacionario da lugar a un error grande. Esta señal de error acumulada se multiplica por un factor de ganancia 'I' y se convierte en el término de salida 'I' del controlador PI.

Por otro lado, se encuentra la acción derivativa que tiene como objetivo mejorar la velocidad del controlador y responder al cambio de la señal de error. La entrada del término ‘D’ se calcula restando el valor de error actual respecto a un valor anterior. Este valor se multiplica por un factor de ganancia ‘D’ dando lugar al termino de salida ‘D’ de los controladores típicos PID. Sin embargo, esta acción para el control de motores no se tiene en cuenta ya que puede amplificar el ruido generado en el motor, lo que a su vez puede causar cambios excesivos en el ciclo de trabajo del PWM que podrían dañar al motor.

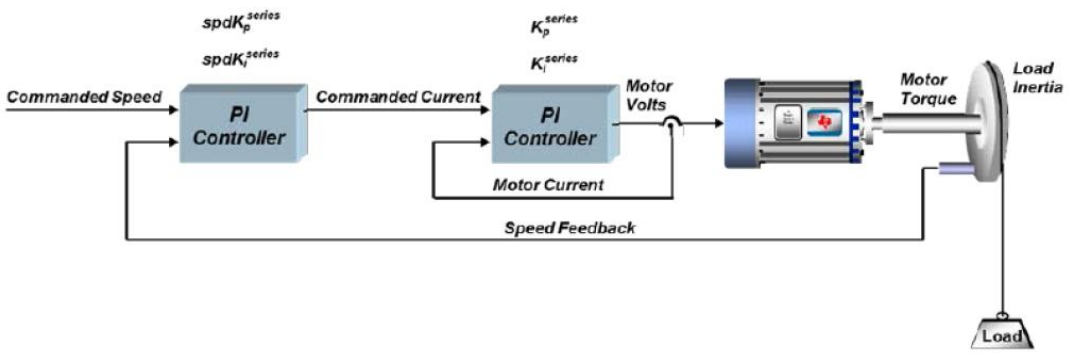


Figura 24. Controlador de velocidad en cascada con un controlador de corriente

Cabe destacar que para el cálculo de las constates Kp y Ki del controlador de velocidad se ha seguido el método de iteración, tal como se ha comentado al principio de este apartado, mientras que para el caso de las corrientes I_d e I_q, en su lugar, se han usado una serie de fórmulas para determinar sus constantes en función de la resistencia e inductancia del motor.

4.4.1 Controlador PI del par y flujo del motor

Para el control del par y flujo del motor se va a implementar un regulador PI en serie, como el que se puede observar en la figura 25.

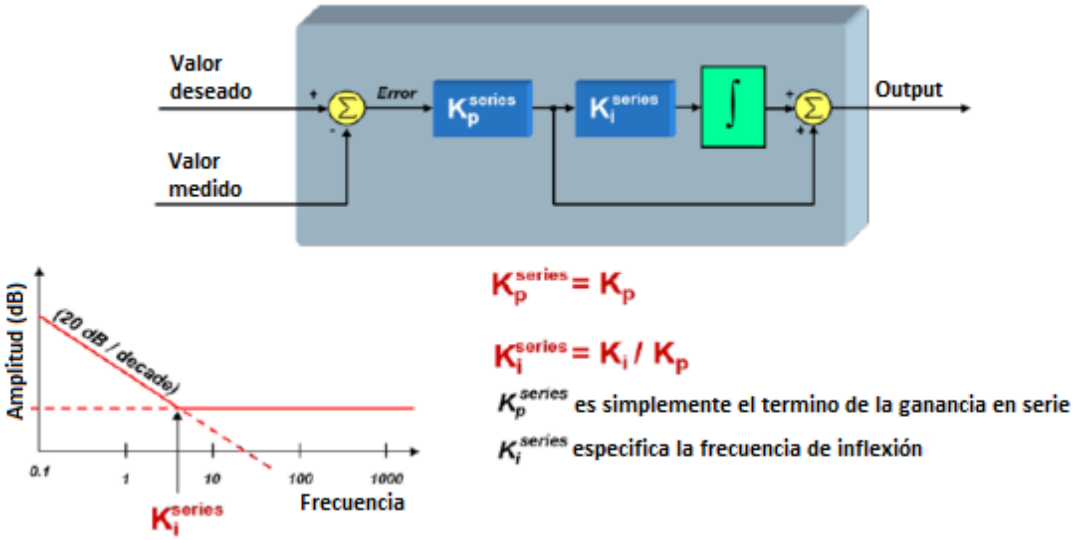


Figura 25. Regulador PI en serie genérico

La ganancia del controlador PI, K_p^{series} y K_i^{series} tienen un efecto sustancial en la actuación del controlador. K_p^{series} determina la frecuencia para todas las frecuencias mientras que K_i^{series} define el punto de inflexión del controlador en rad/sec. [10]

La función de transferencia del controlador en el dominio s es:

$$PI(s) = \frac{K_p^{series} * K_i^{series}}{s} + K_p^{series} = \frac{K_p^{series} * K_i^{series} * \left(1 + \frac{s}{K_i^{series}}\right)}{s} \qquad (31)$$

De esta fórmula, se puede observar que existe un polo en s=0 y un cero en s=K_i^{series}. La figura 26 muestra el controlador PI y el circuito de la bobina del estator equivalente. (primer orden de aproximación del bobinado del estator usando una inductancia, una resistencia y una fuente de tensión de back-EMF en serie).

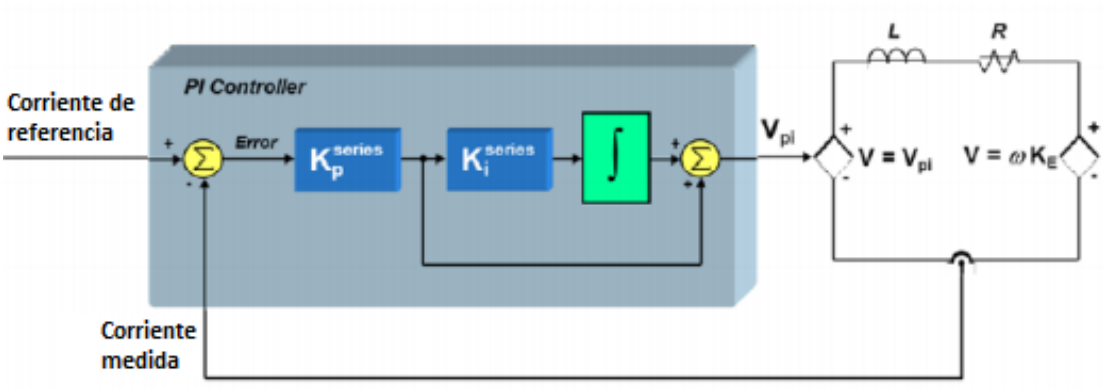


Figura 26. Sistema de control del par y flujo del motor

La señal de la función de transferencia del voltaje del motor a la corriente del motor es:

$$\frac{I(s)}{V(s)} = \frac{\frac{1}{R}}{1+\frac{L}{R}s} \qquad (32)$$

La ganancia del bucle de control es el producto de la función de transferencia del controlador PI definido y la función de que relaciona el voltaje y la corriente del motor, definida anteriormente.

$$G_{bucle}(s) = PI(s) * \frac{I(s)}{V(s)} = \left(\frac{K_p^{series} * K_i^{series} * \left(1+\frac{s}{K_i^{series}}\right)}{s}\right)\left(\frac{\frac{1}{R}}{1+\frac{L}{R}s}\right) \qquad (33)$$

La respuesta del bucle cerrado es:

$$G(s) = \frac{G_{bucle}(s)}{1+G_{bucle}(s)} \qquad (34)$$

Asumiendo que H(s)=1.

Sustituyendo la ecuación 33 en la ecuación 34, la función de transferencia del bucle cerrado queda como:

$$G(s) = \frac{\left(\frac{K_p^{series} * K_i^{series} * \left(1+\frac{s}{K_i^{series}}\right)}{s}\right)\left(\frac{\frac{1}{R}}{1+\frac{L}{R}s}\right)}{1+\left(\frac{K_p^{series} * K_i^{series} * \left(1+\frac{s}{K_i^{series}}\right)}{s}\right)\left(\frac{\frac{1}{R}}{1+\frac{L}{R}s}\right)} \qquad (35)$$

Después de simplificar la función, la función de transferencia puede ser definida como:

$$G(s) = \frac{1 + \frac{s}{K_i^{series}}}{\left(\frac{L}{K_p^{series} K_i^{series}}\right)s^2 + \left(\frac{R}{K_p^{series} K_i^{series}} + \frac{1}{K_i^{series}}\right)s + 1} \quad (36)$$

Observando la ecuación 36, se puede ver que el denominador es una expresión de segundo orden, lo que significa que tiene dos polos. Por lo tanto, la correcta selección de los valores K_p^{series} y K_i^{series} es crítica para evitar polos complejos y para mantener un control estable del motor. Es necesario elegir los polos que no estén situados cerca del eje jω para evitar picos resonantes.

El denominador de la ecuación 36 se puede reducir en la siguiente expresión:

$$G(s)denominador = \left(\frac{L}{K_p^{series} K_i^{series}}\right)s^2 + \left(\frac{R}{K_p^{series} K_i^{series}} + \frac{1}{K_i^{series}}\right)s + 1 \quad (37)$$
$$= (1 + Cs)(1 + Ds)$$

$$\frac{L}{K_p^{series} K_i^{series}} = C * D \quad (38)$$

$$\frac{R}{K_p^{series} K_i^{series}} + \frac{1}{K_i^{series}} = C + D \quad (39)$$

$$\frac{R}{K_p^{series} K_i^{series}} = C \; y \; \frac{1}{K_i^{series}} = D \quad (40)$$

Sustituyendo los valores C y D obtenidos a través de la ecuación 40 en la ecuación 36:

$$G(s) = \frac{1 + \frac{s}{K_i^{series}}}{\left(1 + \left(\frac{R}{K_p^{series} K_i^{series}}\right)s\right)\left(1 + \left(\frac{1}{K_i^{series}}\right)s\right)} \quad (41)$$

Tal como observamos en la ecuación 41**,** la sustitución de 'D' ha dado como resultado en la cancelación de un polo con uno de los ceros de la función de transferencia original. Por lo tanto, con la correcta selección de 'C' y 'D', un sistema de bucle cerrado con ningún cero y un polo real puede alcanzarse.

Además, sustituyendo las expresiones de C y D establecidas en la ecuación 40 en la ecuación 38 da como resultado la siguiente relación:

$$K_i^{series} = \frac{R}{L} \quad (42)$$

Por otro lado, también se tiene que:

$$G(s) = \frac{1}{\frac{L}{K_p^{series}}s + 1} => \; K_p^{series} = L * Bandwith \quad (43)$$

Considerando la ecuación 42 y 43, se tiene que establecer el valor de K_i^{series} al valor del polo de la planta y el K_p^{series} establecerlo al valor del ancho de banda (bandwidth) del controlador de corriente.

Una vez definidos los valores de las constantes K_p y K_i se procede a implementar el controlador PI en serie en el lenguaje C, que es el utilizado para escribir el código del FOC. El controlador PI está dotado de un sistema de *antiwindup* que limita la acción de integral, así como un limitador del controlador en sí, para evitar que el controlador aporte un valor superior al que puede aportar la parte de control del FOC.

```
pi_error = ref - real;
         error_sum += pi_error;

        /* Limiting Error */
        if (error_sum < ERRORMIN)
           {
              error_sum = ERRORMIN;
           }
        else if (error_sum > ERRORMAX)
           {
              error_sum = ERRORMAX;
           }

        out = Kp*(pi_error + Ki*error_sum);

        /* Limiting output */
        if (out < OUTMIN)
           {
              out = OUTMIN;
           }

        else if (out > OUTMAX)
           {
              out = OUTMAX;
            }
```

4.4.2 Controlador PI de velocidad

En la figura 27 se puede observar la estructura de un controlador PI en paralelo:

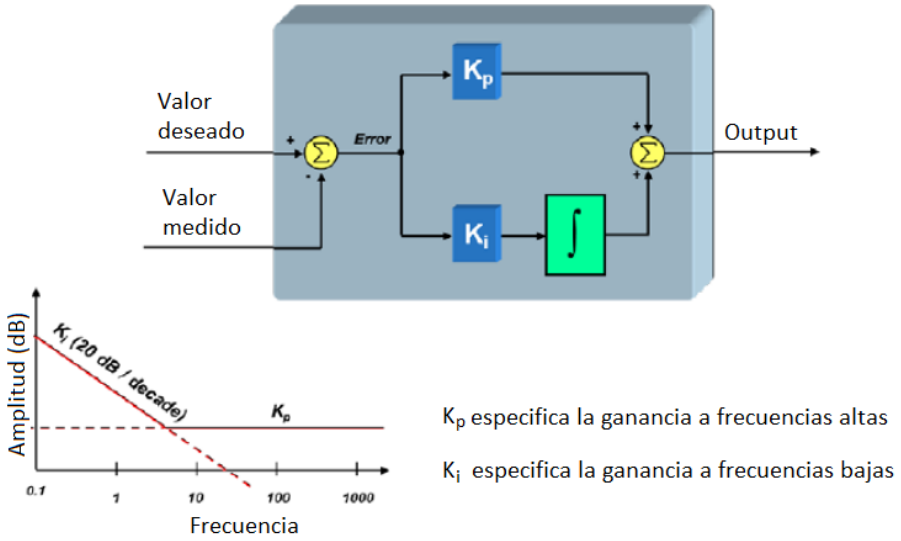


Figura 27. Controlador en paralelo

Como se puede ver, el controlador de velocidad está en cascada con el controlador de corriente. Esto es una característica por destacar ya que el controlador de velocidad reside en el dominio mecánico donde la constante de tiempo es mucho más baja. La ecuación del controlador PI en paralelo es:

$$output(t) = K_P * e(t) + K_I \int_0^t e(t) * dt \qquad (44)$$

Para el controlador PI de velocidad se utilizará el método de la iteración donde se elegirán los valores de K_p y K_i que mayor tiempo de reacción y menor amortiguamiento presenten.

Para determinar los valores de K_p y K_i idóneos en primer lugar se tiene que igualar el valor de la acción integral a cero. Entonces, incrementar el valor del valor de la ganancia 'P' hasta que el sistema responda a los cambios sin excesivas oscilaciones ni sobreamortiguamientos. Usar valores bajos de 'P' dará lugar a que el controlador actúe de forma muy lenta, mientras que valores altos darán a un control más agresivo.

Después de seleccionar un valor de 'P' razonable, el siguiente paso consiste en aumentar suavemente el valor de la ganancia 'I' para forzar al error del sistema a cero. De normal se necesitan valores de 'I' pequeños. El efecto de la ganancia 'I', si es lo suficientemente grande, puede superar la acción del término 'P', ralentizar la respuesta de control y hacer que el sistema oscile alrededor del punto de referencia.

Una vez definidos los valores de las constantes del controlador, se pasa a su implementación en el lenguaje C, el cual se muestra a continuación:

```
pi_error = ref - real;
 error_sum += pi_error;

/* Limiting Error */
if (error_sum < ERRORMIN)
   {
      error_sum = ERRORMIN;
   }
else if (error_sum > ERRORMAX)
   {
      error_sum = ERRORMAX;
   }

out = Kp*pi_error + Ki*error_sum;

/* Limiting output */
if (out < OUTMIN)
   {
      out = OUTMIN;
   }

else if (out > OUTMAX)
   {
      out = OUTMAX;
    }
```

4.5 Arranque del motor y alineación

Para conseguir el par máximo del motor PMSM, el ángulo entre el flujo del rotor y estator debe estar alienado a 90 grados. Para mantener los 90 grados entre los dos flujos es importante conocer la posición inicial del rotor. Debido a que la posición inicial del motor no puede controlarse, por lo que debe de alienarse a una posición conocida. Esto se puede conseguir mediante la excitación de solo uno de los ejes q o d (cuando nos encontramos en el sistema de referencia del rotor, transformada de Park). Excitar el eje q es preferible ya que de este modo se mantiene el eje d en '0'.

Después de la alienación inicial mediante la excitación de q, se debe de desplazar el ángulo eléctrico 90 grados para obtener el máximo par. Sin está acción, el motor se detendrá ya que no habrá ninguna diferencia de ángulo entre el flujo del estator y el rotor.

Las siguientes imágenes muestran la posición del flujo del rotor en reposo, después de la excitación del eje q y después del desplazamiento de 90 grados.

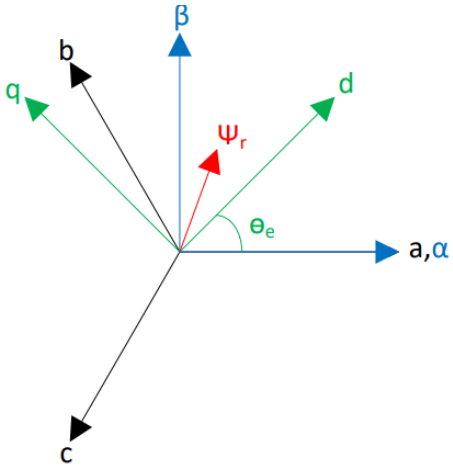


Figura 28. Posición del flujo del rotor en reposo

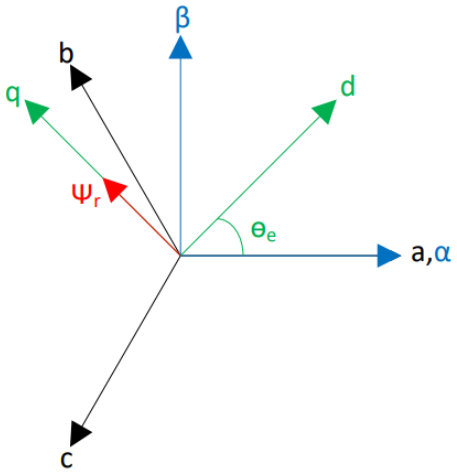


Figura 29. Posición del rotor después de la excitación del eje q

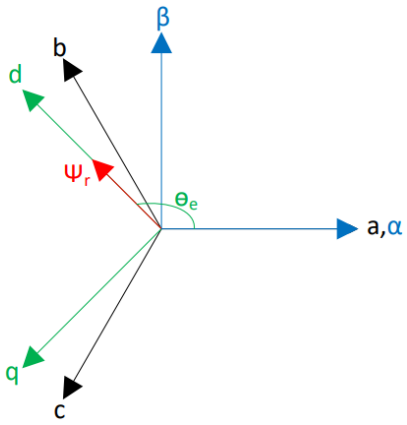


Figura 30. Posición del flujo del rotor después de desplazarse 90 grados

4.6 Debilitamiento de campo

El debilitamiento de campo en un motor BLDC consiste en la imposición de un valor negativo para la corriente del estator en el eje-d del marco giratorio, produciendo, en consecuencia, un debilitamiento del flujo del entrehierro.

La caída de tensión en un motor se produce sobre la resistencia del estator y la reactancia de los devanados y el resto se utiliza para contrarrestar el Back-EMF. El Back-EMF es proporcional a la velocidad del motor y a la constante de voltaje, K_Φ, del motor [10]. Teniendo en cuenta la limitación de la tensión del inversor, se puede lograr un aumento de la velocidad disminuyendo la contante del motor K_Φ, el cual es proporcional al flujo del entrehierro. Aunque se debe de tener en cuenta que una disminución del flujo implica una disminución de par.

El efecto de la corriente de armadura del eje d sobre el debilitamiento de campo depende de la forma y las propiedades magnéticas del motor, desde los dientes de la armadura hasta el núcleo del rotor, por lo que su implementación es un tanto compleja. Sin embargo, esta característica puede llegar a ser útil en los casos donde se necesite una velocidad superior y en el par del motor no sea una característica importante. En la figura 31 se puede observar en qué momento se produce el efecto de debilitamiento de campo.

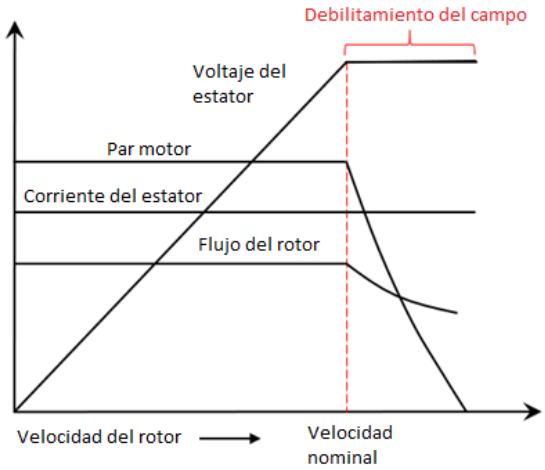


Figura 31. Debilitamiento de campo de un motor BLDC

4.7 Aplicaciones del FOC

FOC tiene múltiples características que lo presentan como una solución muy útil en distintos sectores de la industria. Por ejemplo, es usado en el control de motores de un brazo robot debido a su capacidad de situarse en la posición deseada de forma precisa o en el control de cintas transportadoras y ventiladores donde se requiere un control preciso de velocidad. También se usa en aplicaciones donde se tenga que mantener el calor generado tan bajo como sea posible, como por ejemplo en aplicaciones de refrigeración y procesos sensibles a la temperatura [11].

Además, debido a que hoy en día es una de las técnicas de control más eficientes, lo hace ideal en aplicaciones de tracción eléctrica, como por ejemplo en los coches eléctricos que cada vez son más populares, donde la eficiencia juega un papel importante ya que permite aumentar la vida útil de la batería.

5. Simulación

Para comprobar el funcionamiento de la técnica de control de campo orientado, se ha realizado una simulación, antes de proceder a realizar un montaje experimental. Cada parte se va a explicar en cada uno de los apartados siguientes, comenzando por la presentación del puente inversor de PWM junto con el motor y terminando en las transformadas inversas de Clarke y Park que son las que alimentan las puertas de los mosfets que se usan como interruptores. En el apartado Planos de este trabajo se puede encontrar una imagen de la simulación entera.

5.1 Motor y puente inversor de tres fases

En la figura 32 se puede observar el motor junto con el puente inversor y dos bloques que tienen el nombre de “sensors” y “load”.

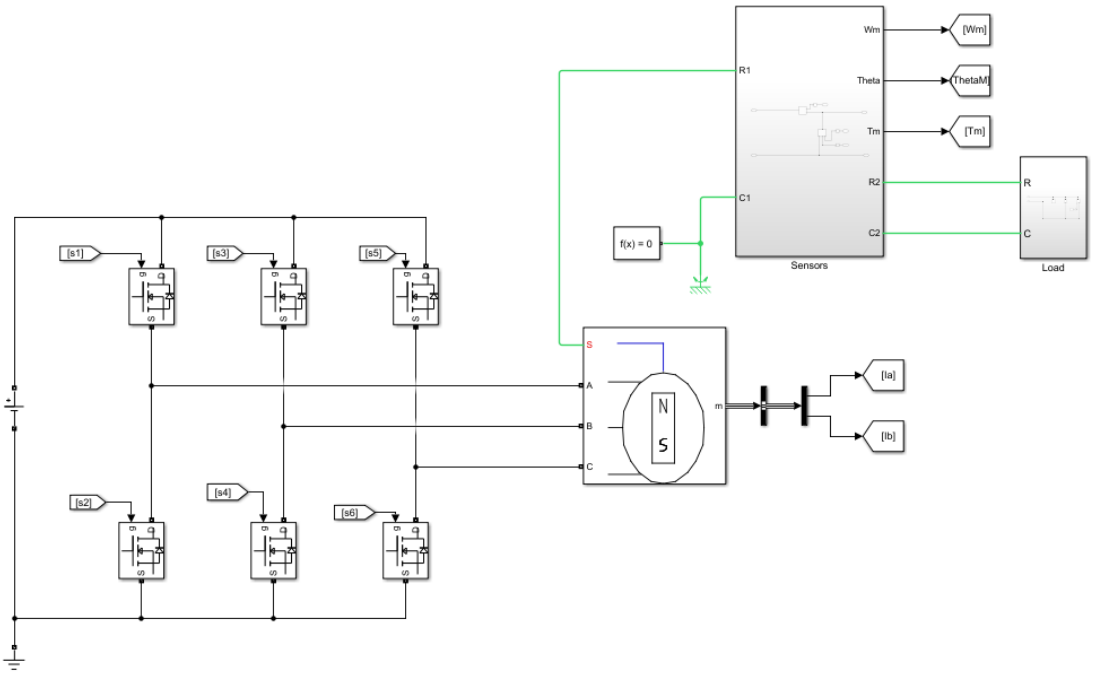


Figura 32. Motor BLDC con puente inversor y carga mecánica para el desarrollo del control vectorial

Tal como se puede ver, en primer lugar, se encuentra el puente inversor, formado por seis mosfets que se encargan de controlar el motor, alimentado con una fuente de tensión constante de 100V. Cada mosfet está controlado a través de su puerta, gate, e irá cambiando de estado entre cerrado y abierto con el objetivo de generar las 3 señales obtenidas mediante la técnica de modulación espacial. Dichas señales se utilizarán para alimentar el motor, y mediante las dos salidas del motor se puede medir la corriente de las fases A y B, ya que para realizar la transformada de Clarke se requiere únicamente la medición de dos fases de las tres fases.

El motor presenta algunos parámetros requeridos para la computación del modelo. Algunos de estos valores son:

```
Phi = 0.1;        % Flujo de enlace los imanes permanentes [Wb]
Ld = 0.01;        % Inductancia del estator del eje d [H]
Lq = 0.02;        % Inductancia del estator del eje q [H]
Rs = 0.38;        % Resistencia del estator de cada fase [ƒ¶]
p = 2;           % Número de polos pares
```

El motor tiene 4 entradas, tres de ellas proceden del puente inversor y la cuarta es el par mecánico aplicado al motor, de normal se le aplica un par constante, pero en este caso se ha introducido una rampa a mitad de la simulación con el objetivo de observar el tiempo de reacción del motor en situaciones donde aparezca una carga de frenado. A parte de aplicar una rampa, se ha aprovechado esta entrada para medir el par motor, la velocidad angular y el ángulo del motor.

En las figuras 33 y 34 se puede observar con mayor detalle los bloques “sensors” y “load”.

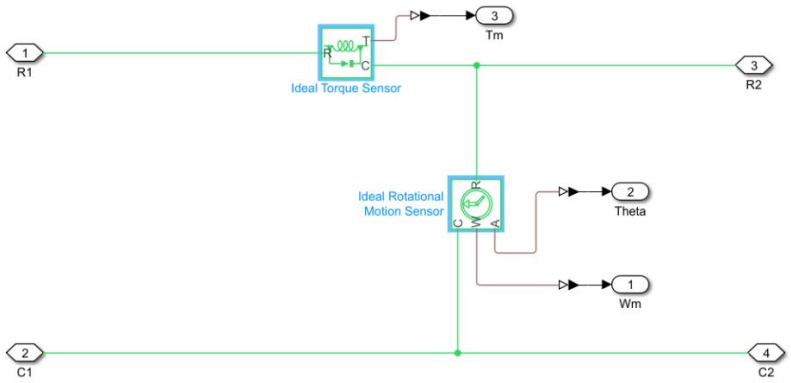


Figura 33. Bloque sensors

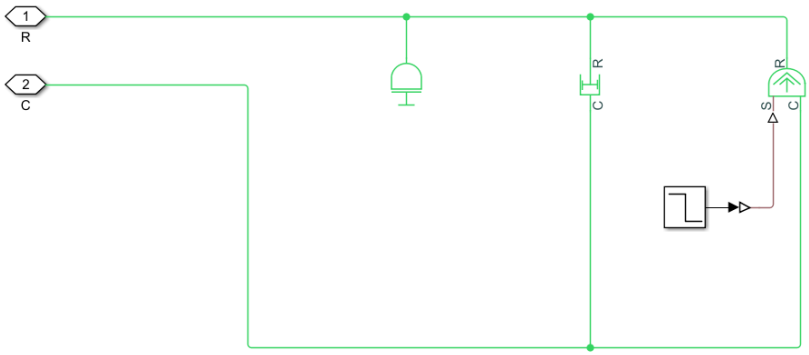


Figura 34. Bloque load

En la figura 34 se puede observar la rampa cuyo objetivo es aplicar una carga negativa al motor. Sin embargo, también podría darse el caso de que en vez introducir una rampa negativa se introdujese una rampa positiva, en tal caso el inversor en vez de generar mayor corriente generaría menor corriente con el objetivo de mantener la velocidad deseada.

Una vez definidas las entradas y salidas del motor, el siguiente paso consiste en realizar las transformadas de Clarke y Park, las cuales se explicarán en el apartado siguiente.

5.2 Transformadas

Tal como se observa en la figura 35, una vez obtenidas la corriente de dos de las fases del motor, se procede a aplicar la transformada de Clarke. Una vez obtenidas las corrientes I_α e I_β del sistema de referencia estacionario, el siguiente paso consiste en aplicar la transformada de Park, sin embargo, se necesitan los valores del seno y coseno del ángulo del motor antes de poder realizar la transformada.

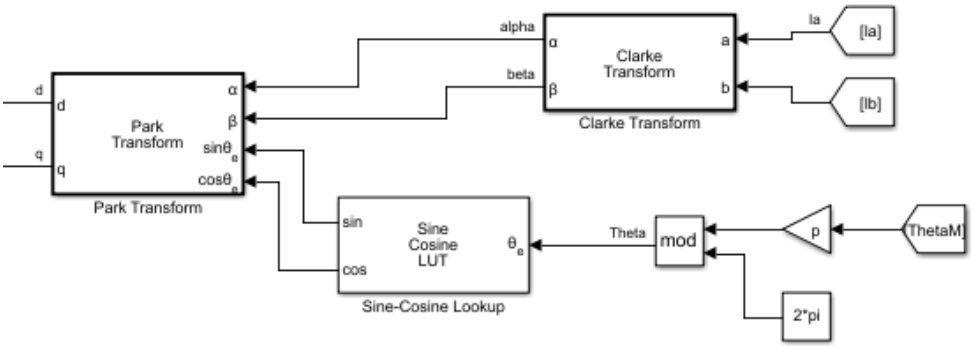


Figura 35. Transformada de Clarke y tabla de seno y coseno

Para calcular los valores del seno y coseno del ángulo se necesita saber la posición del motor, dicho dato se obtiene a partir del sensor que se puede observar en la figura 33, denominado Ideal Rotational Motion Sensor. Sin embargo, tal valor es el ángulo mecánico del motor y como se comentó en el apartado 4.1 se necesita el ángulo eléctrico. Para calcular el ángulo eléctrico se puede aplicar la ecuación 1.7 del apartado 1.3, que básicamente consiste en multiplicar el ángulo mecánico por el número de polos pares del motor, que en este caso es de 2. Una vez multiplicado por el número de polos pares se obtiene el ángulo eléctrico, pero aún se debe de realizar un cálculo adicional antes de poder utilizarlo ya que este valor aumenta constantemente y se debe de ajustar para que se encuentre en el rango de 0 a 2π radianes. Para ello se utiliza el bloque denominado "mod" cuya función consiste en calcular el resto de una división, cuyas entradas son el ángulo eléctrico y 2π. En las figuras 36 y 37 se puede observar el ángulo eléctrico antes y después del bloque mod.

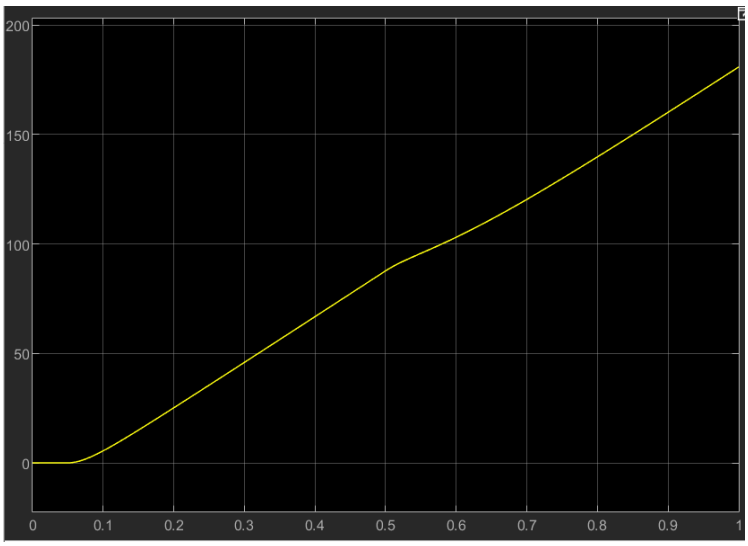


Figura 36. Ángulo eléctrico antes del bloque mod

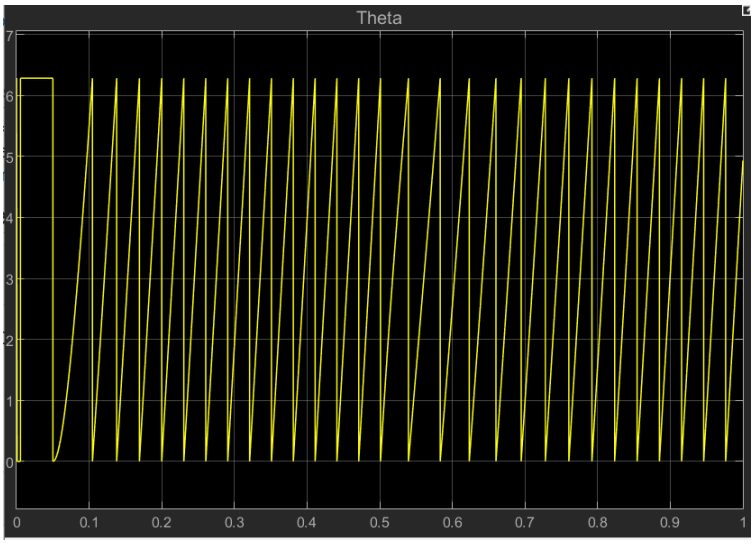


Figura 37. Ángulo eléctrico después del bloque mod

Tal como se observa, una vez aplicado el bloque mod se puede ver como el valor aumenta hasta llegar al valor de 2π y una vez alcanzado vuele al valor 0. Mientras que en la figura 36 el valor del ángulo aumenta constantemente. Una vez obtenido el ángulo eléctrico del motor se procede a calcular los valores I_d e I_q a través de la transformada de Park.

Una vez obtenidos los valores I_d e I_q, el siguiente paso consiste en comparar dichos valores con los valores de referencia, y el error obtenido de esa comparación introducirlos en los controladores PI, tal como se observa en la figura 38.

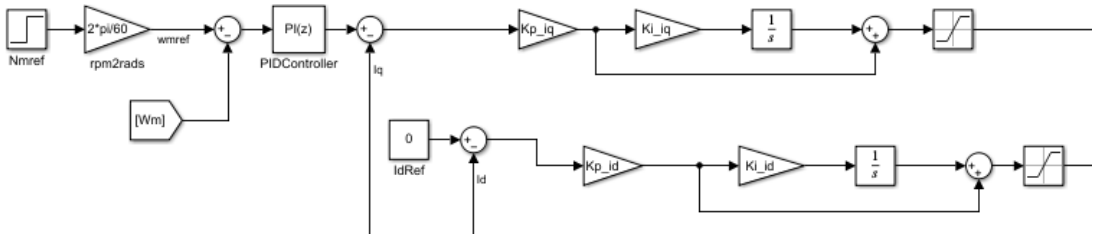


Figura 38. Comparación de los valores Id e Iq

En el apartado 4.1 se explicó la razón por la cual la corriente de referencia, I_d es igual a 0. Sin embargo, la corriente de par, I_q, se debe de ajustar en función de la velocidad deseada, ya que esta se encarga de generar la corriente necesaria para mantener la velocidad objetivo. Por lo tanto, el valor de referencia I_q procede de la salida del controlador PI encargado de ajustar su salida para mantener la velocidad de referencia.

Los valores de las constantes del Kp y Ki del controlador PI de las corrientes I_q e I_d son:

```
wc_c = 1e3;            % Frecuencia del ancho de banda [rad/s]
Kp_id = wc_c*Ld;       % ganancia proporcional del eje d
Ki_id = Rs/Ld;         % ganancia integral del eje d
Kp_iq = wc_c*Lq;       % ganancia proporcional del eje q
Ki_iq = Rs/Lq;         % ganancia integral del eje q
```

Mientras que los valores de Kp y Ki del controlador PI de velocidad son:

```
Kp_s = 0.0330;          % ganancia proporcional de la velocidad
Ki_s = 0.2925;          % ganancia integral de la velocidad
```

Además, tal como se observa en la figura 38 el bloque "Nmref" determina la velocidad deseada, sin embargo, el sensor de velocidad funciona en rad/s por lo que en primer lugar se debe de ajustar el valor de referencia a rad/s.

5.3 Transformadas inversas

Una vez realizadas las transformadas de Clarke y Park y de haber generado las señales de control a través de los controladores PI, el último paso consiste en realizar las transformadas inversas y aplicar la técnica de modulación espacial para poder alimentar con las señales generadas las puertas de los mosfets que controlan el motor BLDC.

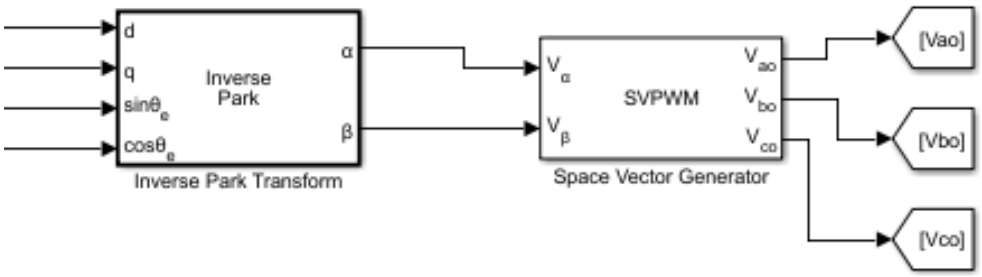


Figura 39. Transformadas inversas y la técnica SVPWM

Tal como se observa en la figura 39, en primer lugar, se tiene el bloque "Inverse Park", este se encarga de generar las señales V_α y V_β del marco de referencia estacionario de dos ejes, para ello utiliza los valores V_d y V_q obtenidos a través de los controladores PI y los valores del seno y coseno del ángulo eléctrico del motor adquirido anteriormente. Una vez que se tienen los valores V_α y V_β el siguiente paso consiste en aplicar la técnica de modulación espacial, la cual se realiza por medio del bloque denominado "SVPWM". De este modo obtenemos las tres señales necesarias para alimentar el motor.

Finalmente, las tres señales obtenidas se comparan con una señal triangular centrada, cuya frecuencia es de 10kHz ya que para esta técnica de modulación se requiere únicamente usar un PWM centro alienado, tal como demostró Peter Pinewski en 1996[12]. La comparación consiste en que si la señal generada con la técnica SVM es mayor que la señal triangular se genera un nivel lógico de 0 en el mosfet correspondiente de la rama de arriba y un 1 en el mosfet de la rama de abajo.

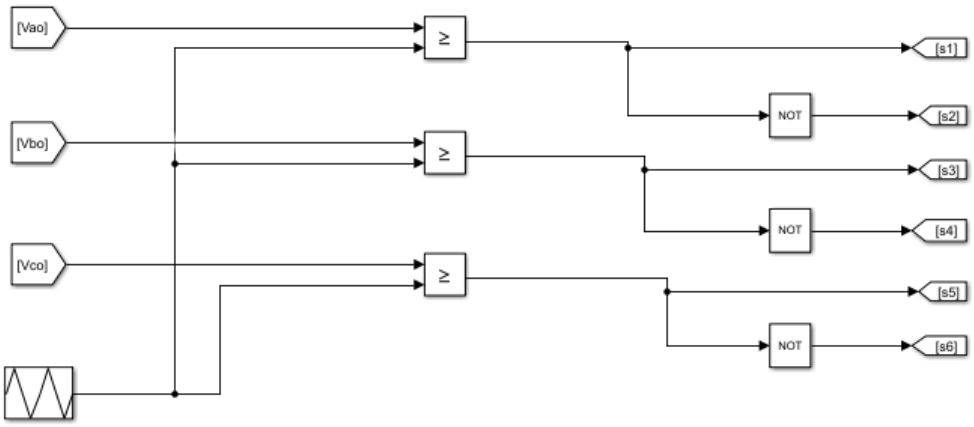


Figura 40. Comparación de las señales con una señal triangular

De este modo, el bucle de control FOC se cierra, permitiendo un control total de la velocidad del motor. En el siguiente apartado se muestra los resultados de la simulación realizada.

5.4 Resultados de la simulación

Para comprobar el correcto funcionamiento del FOC se ha introducido una carga de frenado, de valor -0.2, en el motor, a través del bloque "Ideal Torque Source", tal como se comentó en el apartado 5.1, que actuará a mitad de la simulación. Su efecto se puede observar en la figura 41.

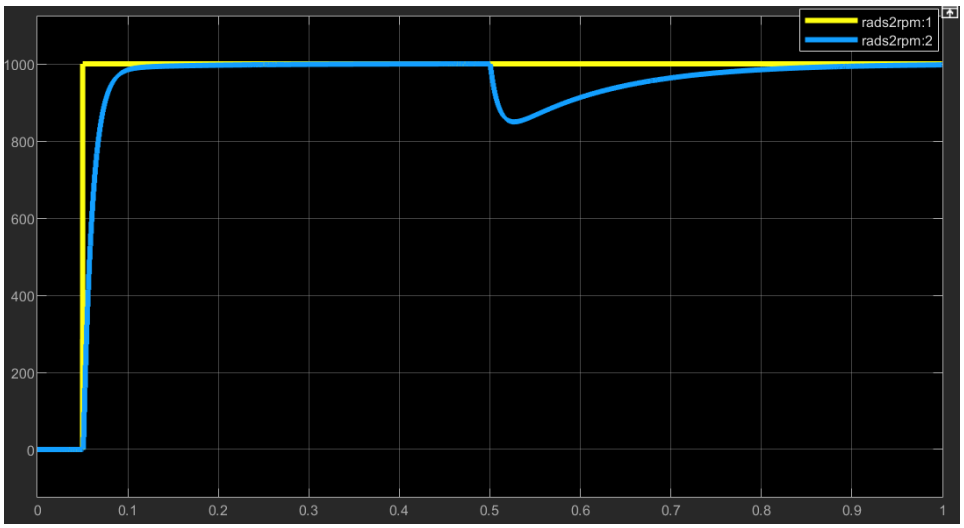


Figura 41. Resultado de la simulación con carga de frenado

Como se puede observar al principio de la simulación el motor trata de alcanzar la velocidad de referencia y una vez alcanzada se tiene que mantener en dicha velocidad. Sin embargo, al aparecer una fuerza negativa, el motor debe de aumentar su corriente para superar dicha fuerza. Esto se consigue a través de los reguladores PI que controlan las corrientes I_d e I_q, que aumentan o disminuyen su salida en función de las necesidades del motor. Esta característica se observa en la figura 42.

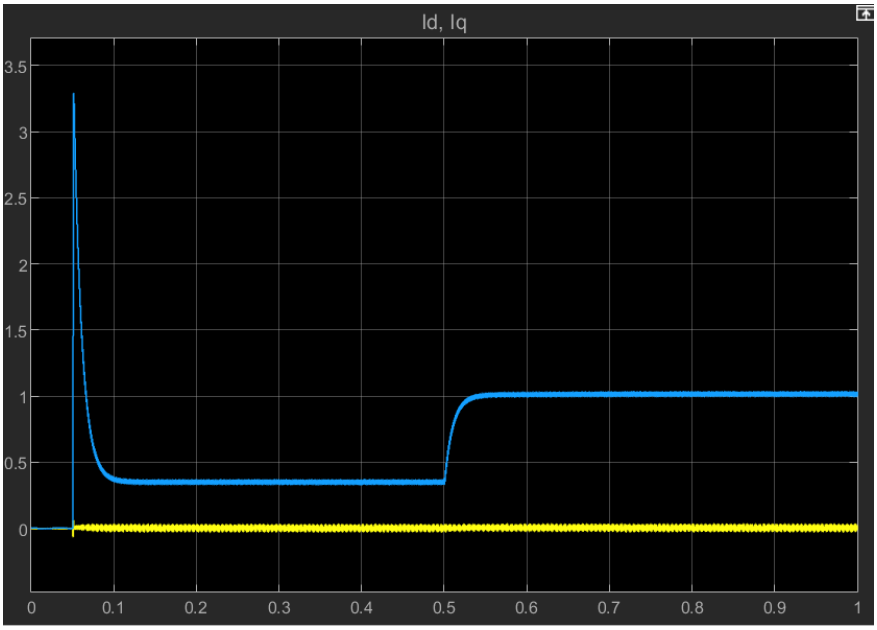


Figura 42. Corrientes Id e Iq en la transformada de Park

Como se puede ver en la primera mitad de la simulación la corriente I_q se mantiene constante ya que no existe ninguna fuerza que haga cambiar la salida del controlador PI de velocidad. Sin embargo, en la segunda mitad el motor sufre de una carga de frenado constante que puede dar en consecuencia una reducción de velocidad, por tanto, dicho controlador aumenta la corriente de referencia I_q, lo que conlleva que el regulador PI encargado de controlar la corriente I_q cambie también. Por ello se observa el cambio de nivel de I_q.

Este fenómeno también se puede observar en la transformada de Clarke.

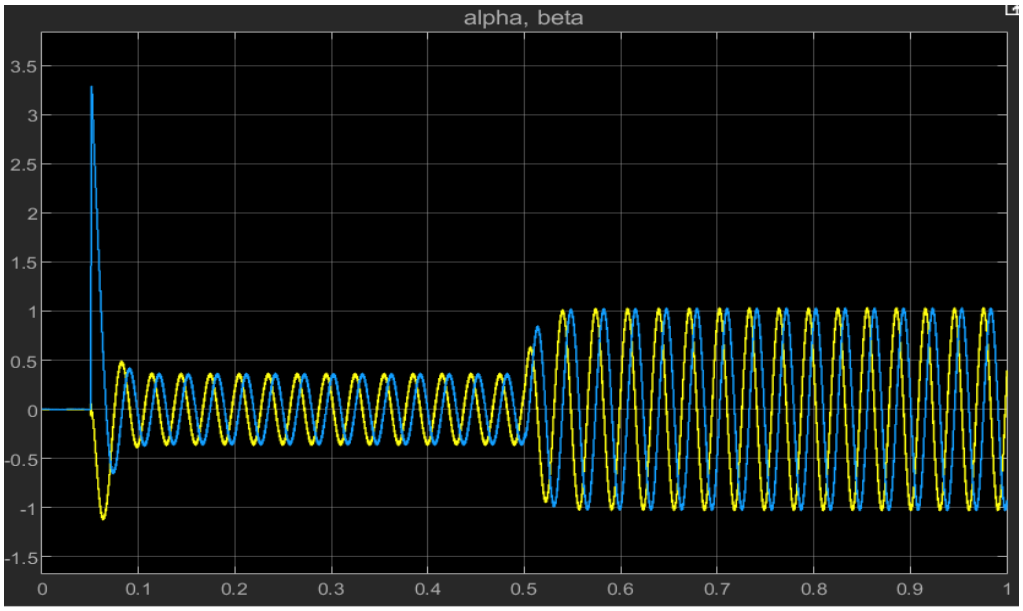


Figura 43. Corrientes Iα e Iß en la transformada de Clarke

A través de la figura 44, se puede observar el aumento de la corriente en cada una de las fases del motor en la segunda mitad de la simulación.

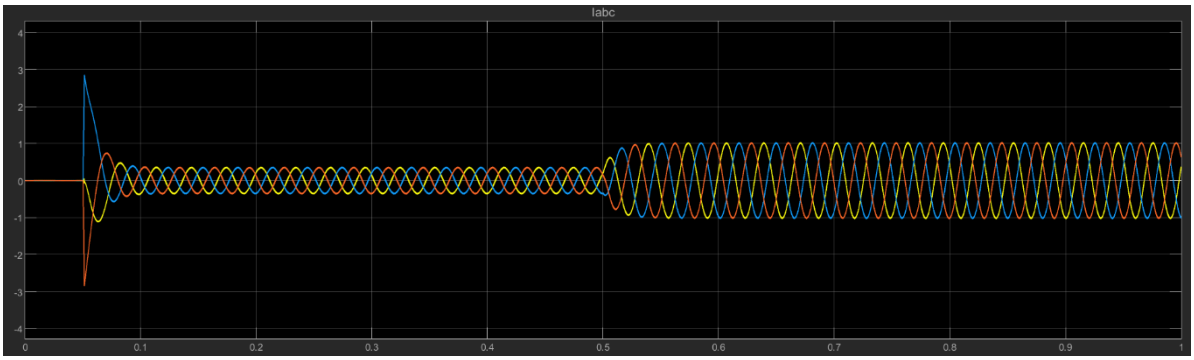


Figura 44. Corrientes de cada una de las fases del motor

Cabe destacar que los picos de corriente que se observan al principio son debidos al arranque del motor ya que, como se indicó en el apartado 2.4, se necesita un par adicional para poder superar la inercia del propio motor.

El mismo fenómeno se puede observar en el caso de que en vez de haber una fuerza de frenado aparezca una fuerza, de valor 0.05, que impulse el motor hacia delante. Por lo que la corriente debe de disminuir para mantener la velocidad deseada.

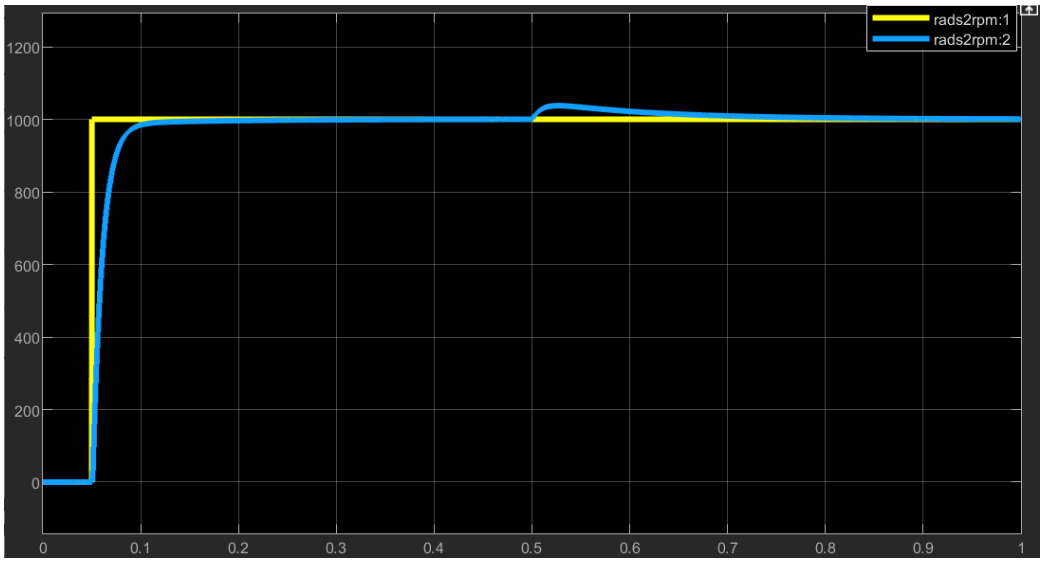


Figura 45. Resultado de la simulación del control vectorial donde el motor recibe una carca que aumenta su velocidad

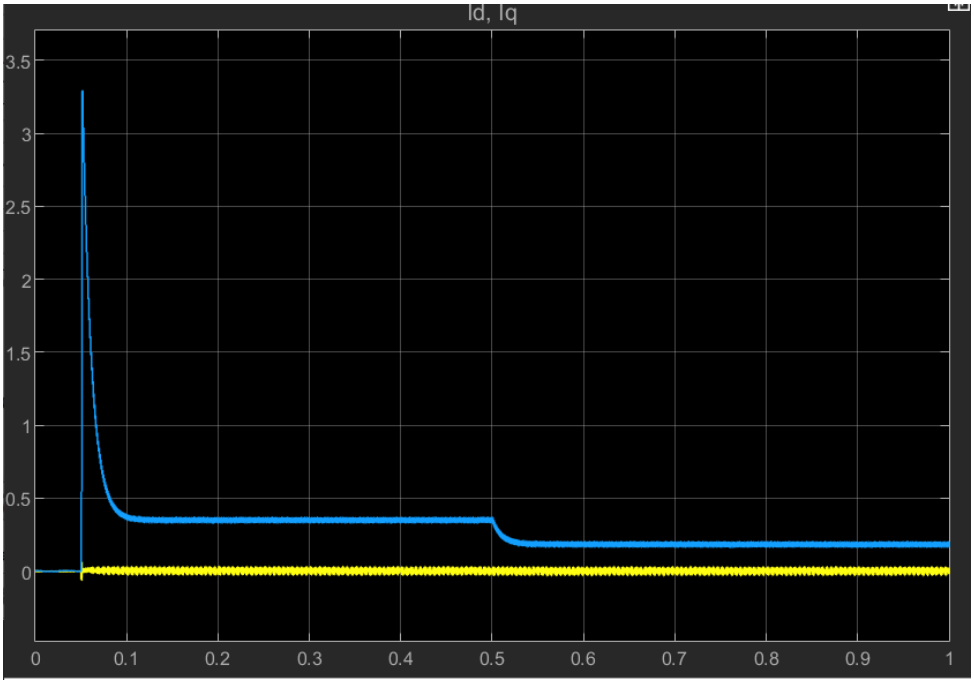


Figura 46. Corrientes I_q e I_d donde el motor recibe una carga que acelera su velocidad

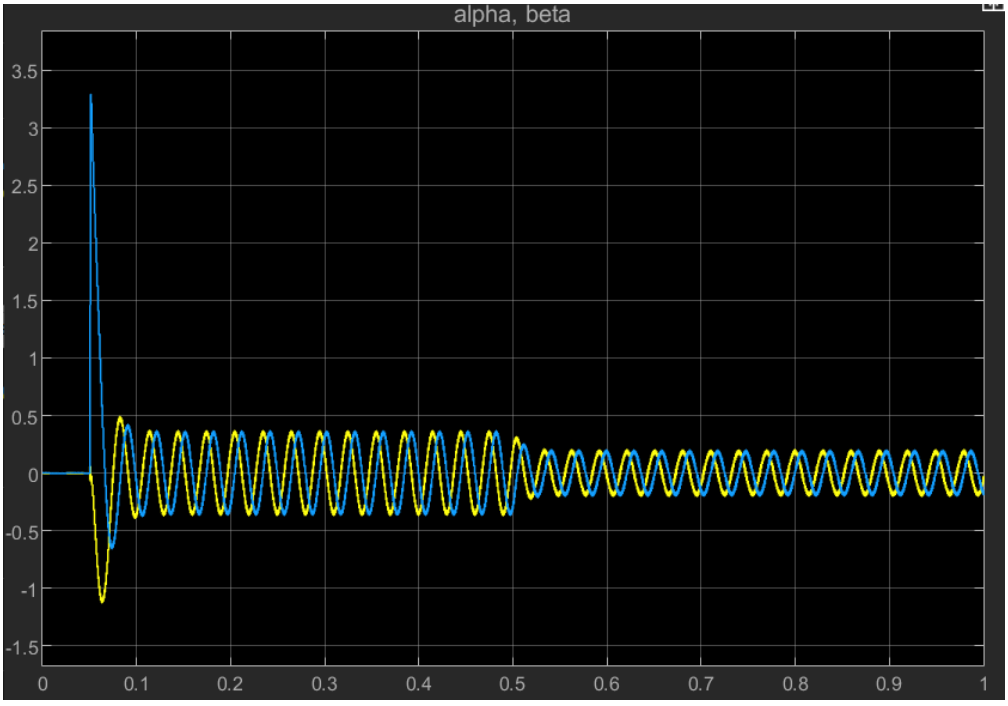


Figura 47. Corrientes I_α e I_β con carga de impulso donde el motor tiene que disminuir su corriente debido a que el motor es impulsado

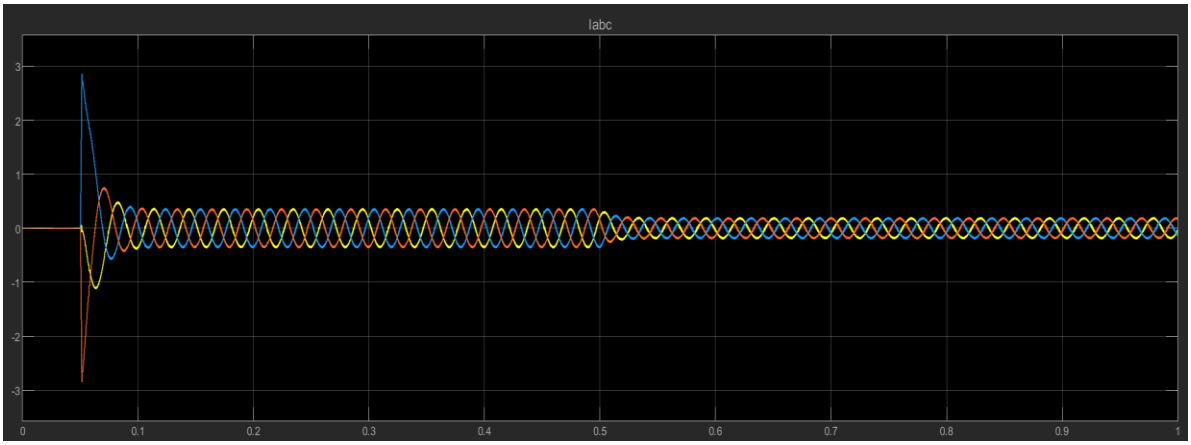


Figura 48. Corrientes del motor con carga de impulso

6. Montaje experimental

El funcionamiento de los motores sin escobillas no sería posible sin un controlador eléctrico por las razones que se han explicado en los capítulos anteriores. Dicho controlador se encarga de aplicar la excitación precisa sobre cada una de las fases del motor en todo momento, su objetivo es mantener el motor en movimiento con los parámetros determinados por el usuario.

Como puede suponer la complejidad de un controlador de motor eléctrico es elevada, pues está formada por una parte de hardware y otra de software que haga posible la ejecución de los algoritmos de control. Debido a esta complejidad, se va a separar cada una de las partes que forman el controlador, y se dará una explicación de cada parte que ayudará a comprender el funcionamiento del conjunto. Las partes que componen el controlador son:

- Etapa de alimentación.
- Etapa de potencia.
- Etapa de medición de corriente.
- Etapa de control.
- Etapa de medición de posición.

6.1 Partes del controlador

6.1.1 Etapa de alimentación

La función de la etapa de alimentación es la de suministrar la potencia eléctrica necesaria para hacer funcionar todos los elementos electrónicos que componen el controlador del motor. Por un lado, se va a utilizar una fuente de alimentación de 12V y 5A para alimentar la etapa de potencia. Debido a ello, se tendrá en cuenta en las pruebas para no sobrepasar dicha corriente ya que podría dañar la fuente. Por otro lado, la etapa de control se va a alimentar con la conexión USB que proporciona el microcontrolador, de manera que estará conectada a una fuente de 5V y 0.9A.

6.1.2 Etapa de potencia

La función de la etapa de potencia es la generación de las señales de excitación del motor de acuerdo con las instrucciones marcadas por la etapa de control.

El elemento principal de esta etapa es el inversor trifásico, circuito utilizado para conseguir las tres señales de corriente alterna de valor eficaz y frecuencia deseada.

Sería posible crear un inversor a partir de transistores (bipolares, MOSFET, IGBT, etc.), sin embargo, para facilitar el montaje de este proyecto, ya que el objetivo de este trabajo no es construir un inversor trifásico, se ha decidido adquirir una placa de control que contenga el inversor y los drivers necesarios para accionar cada interruptor. En este proyecto se utilizará una placa formada por MOSFETs y el driver de potencia DRV8302.

6.1.2.1 Controlador DRV8302

La placa que se ha decidido usar para la etapa de potencia se puede observar en la figura 49:

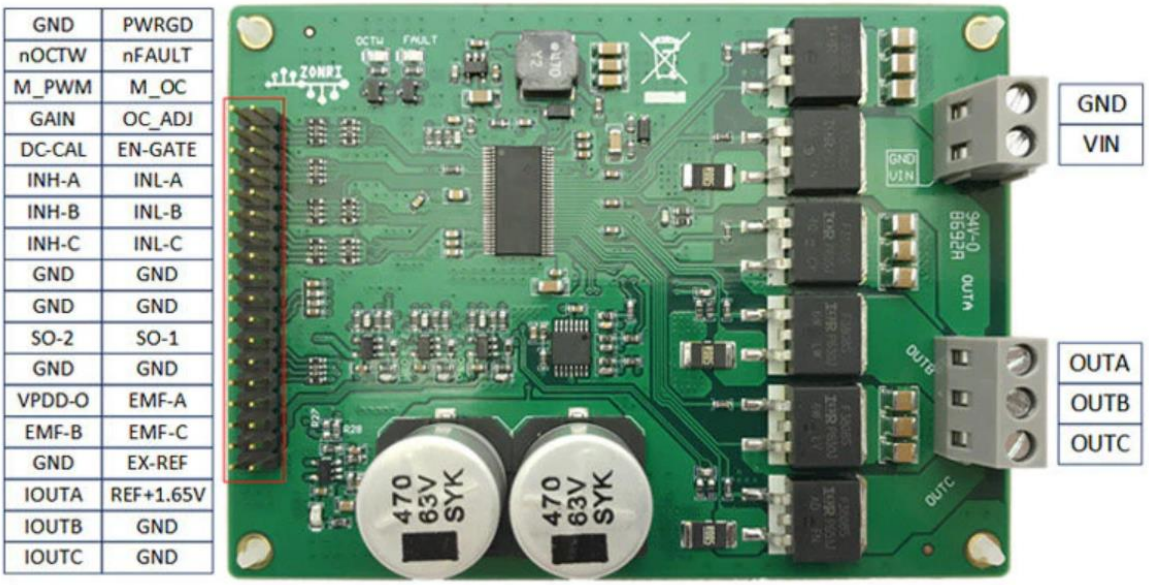


Figura 49. Placa DRV83202 (Texas Instruments)

Se ha decidido utilizar esta placa, cuyas características se encuentran en el Anexo 6 y 7, porque contiene el chip DRV8302, un controlador de puerta, que se caracteriza por su versatilidad, ya que es adecuado para un amplio rango de tensiones y es compatible con distintas técnicas de control.

La principal razón por la cual se necesita el DRV8302 es que no se pueden controlar los MOSFETs de un inversor directamente con la tensión de salida proporcionada por un microcontrolador, ya que esta suele ser normalmente de 3.3V. Para controlar los MOSFETs se necesita un driver que sea capaz de aumentar la tensión de salida del microcontrolador y para ello se usa el DRV8302 que lleva integrado 3 drivers para cada una de las fases del motor.

Además, en su interior incorpora dos sensores de corriente, varios métodos de protección de sobrecorriente y un sistema de alerta de fallo, el cual se encarga de apagar todos los MOSFETs para

mantenerlos en estado de alta impedancia [13]. Además, posee protecciones contra tensiones de bajas y de excesos de temperatura.

Mediante el pin M_PWM se puede seleccionar si se desea controlar cada MOSFET de manera independiente o bien controlar los MOSFETs de cada fase al mismo tiempo.

M_PWM	INL_x	INH_x	GL_x	GH_x
0	0	0	L	L
	0	1	L	H
	1	0	H	L
	1	1	H	H
1	X	0	H	L
	X	1	L	H

Tabla 2. Modos de funcionamiento del PWM

En el mismo documento se ilustra un diagrama de cómo debería diseñarse el circuito para el control de un motor.

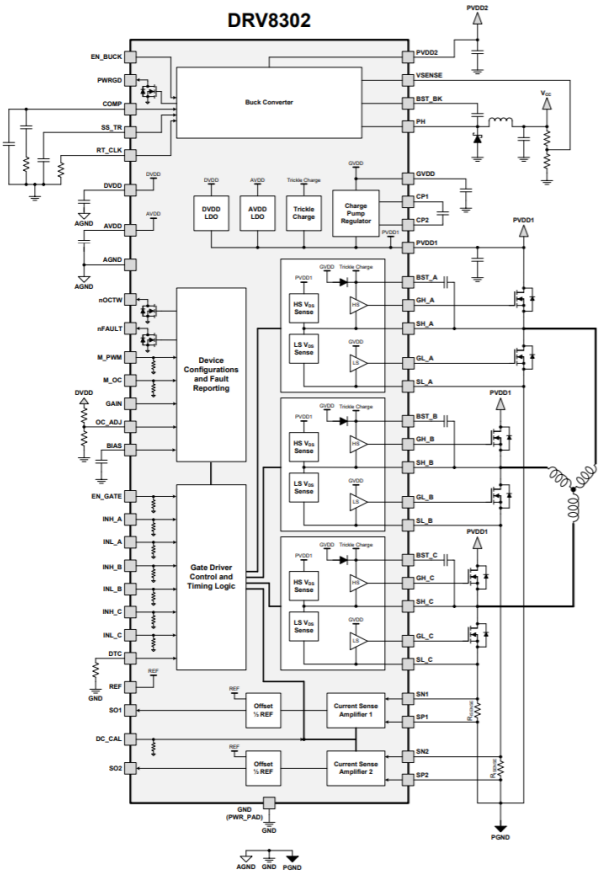


Figura 50. Diagrama del DRV8302

En la figura 50 se pueden distinguir algunas zonas importantes. En la parte de la derecha se encuentran los drivers de los MOSFETs conectados al motor y dos resistencias shunt para medir la corriente. Mientras que en el lado izquierdo se encuentran los pines que estarán conectados a la etapa de control, mediante los cuales se podrán controlar el motor y la salida de los sensores de corriente que estarán conectados a los ADCs del microcontrolador.

6.1.3 Etapa de medición de corriente

Su función es la medición de las corrientes de fase del motor, la información obtenida ha de transferirse a la etapa de control para que con estos datos se pueda ejecutar el algoritmo de control.

De normal para medir la corriente de cada una de las fases se usa resistencias shunt, la ubicación de estas puede diferir. Pueden encontrarse entre el MOSFET del lado bajo y masa, en la salida de cada una de las fases, o bien se puede usarse una única resistencia que junte los MOSFETs del lado bajo, esta técnica presenta la ventaja de ser más barata, pero aumenta considerablemente la complejidad de la etapa de control. Para este proyecto se van a utilizar dos resistencias shunt conectadas en el lado bajo de dos de las fases, ya que esta es la configuración que sigue la placa DRV8302.

El lazo de realimentación de corriente estará compuesto por dos amplificadores, los cuales convertirán la intensidad de corriente que atraviesa cada resistencia de detección en una tensión. La función del amplificador es la de adecuar el valor de tensión para ser medido por el ADC del microcontrolador.

En la figura 51 se puede observar el amplificador con la resistencia shunt.

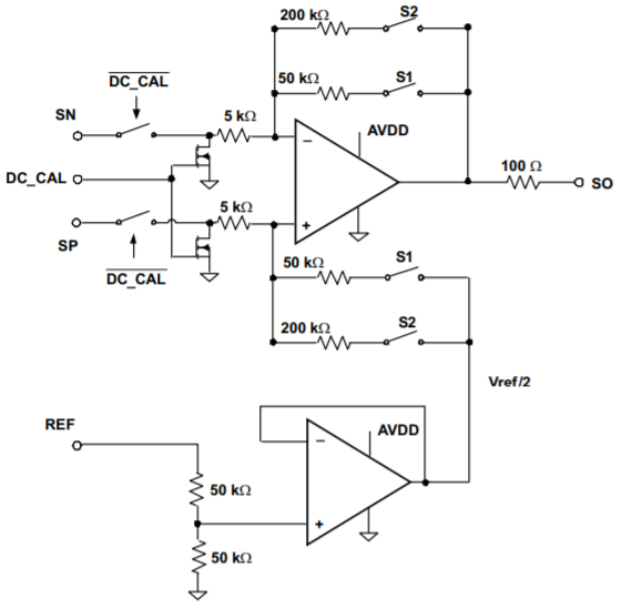


Figura 51. Amplificador con resistencia shunt

Los amplificadores para el sensado de corriente poseen dos valores de ganancia programables mediante el pin GAIN. Estos pueden ser de 10V/V o bien 40V/V.

Además, proporcionan un offset de hasta 3V que permite el sensado de la corriente en ambas direcciones. El offset se establece a la mitad de la tensión aplicada en el pin X-REF.

Mediante el pin DC_CAL se puede realizar una calibración. Cuando el pin de calibración DC este habilitado, el dispositivo corta la entrada de los amplificadores y desconecta la carga. La calibración se puede realizar en cualquier momento, incluso cuando los MOSFETs están conmutando, ya que la carga está desconectada.

La salida del amplificador para el sensado de corriente se puede calcular como:

$$V_O = \frac{V_{REF}}{2} - G * (SN_X - SP_X) \qquad (45)$$

donde:

- VREF es la corriente de referencia del pin X-REF.
- G es la ganancia del amplificador.
- SNX y SPX son las entradas del canal x.

6.1.4 Etapa de control

La etapa de control es la encargada de recibir y procesar la información proveniente de otros periféricos, ejecutar el algoritmo de control y transferir los comandos de control a la etapa de potencia.

Para realizar todas estas funciones se va a utilizar un microcontrolador de la compañía ST Microelectronics. Un microcontrolador es un circuito integrado capaz de ejecutar un conjunto de operaciones en un orden con un fin concreto, por lo tanto, el microcontrolador debe de consistir el núcleo del procesamiento, en una memoria que contenga el algoritmo a ejecutar y almacene el valor de los registros, así como una serie de dispositivos periféricos útiles para las tareas que deba de realizar.

Para este experimento se ha decidido usar el microcontrolador STM32F407-Discovery de la familia ARM-CORTEX-M4 32-bit, que es de propósito general.

Figura 52. Microcontrolador STM32F407-Discovery

La placa STM32F407-Discovery ofrece las siguientes características:

- Microcontrolador STM32F407VGT6 con 1MB de memoria flash, 192KB de RAM, encapsulado LQFP100.
- ST-LINK/V2 incorporado con selector usar el kit como un ST-LINK/V2 independiente (con conector SWD para programación y depuración).
- Fuente de alimentación: a través del bus USB o desde una fuente de alimentación externa de 5V.
- Sensor de movimiento ST MEMS LIS302DL, acelerómetro con salida digital de 3 ejes
- Audio DAC CS43L22 con controlador integrado de altavoz clase D
- Ocho LEDs: - LD1 (rojo / verde) para la comunicación USB - LD2 (rojo) alimentación 3,3 V - Cuatro LEDs de usuario, LD3 (naranja), LD4 (verde), LD5 (rojo) y LD6 (azul) - 2 LEDs USB OTG LD7 (verde), VBus y LD8 (rojo)
- Dos pulsadores (usuario y Reset)
- USB OTG con conector micro-AB

Y el microcontrolador STM32F407VGT6 incorporado a la tarjeta se caracteriza por:

- Procesador ARM Cortex-M4 32-bit.
- FPU tiene 210 DMIPS.
- Memoria de 1 MB Flash, 196 KB RAM.
- Conexiones USB OTG HS/FS, Ethernet.
- Timers 17 TIMs.
- 3 ADCs.

De todas estas características las más significativas para la aplicación es la posibilidad de generar pulsos PWM con los Timers para controlar el motor y los ADCs medir la corriente.

6.1.5 Etapa de medición de la posición

La medición de la posición es una parte indispensable para el control vectorial, ya que es necesario conocer la posición del rotor en todo momento. Para ello, existen dos posibilidades o bien medir la posición a través de un sensor o bien estimarlo midiendo la fuerza contra electromotriz inducida (Back-EMF) producida por el propio motor, este método es más económico, pero a su vez aumenta la complejidad y el tiempo de computación del algoritmo de control del motor.

Para este proyecto se ha decidido usar un encoder incremental para determinar la posición del motor, concretamente el modelo E6B2-CWX1X 360p/r.

Figura 53. Encoder incremental

En la figura 54 se puede observar el funcionamiento del encoder incremental.

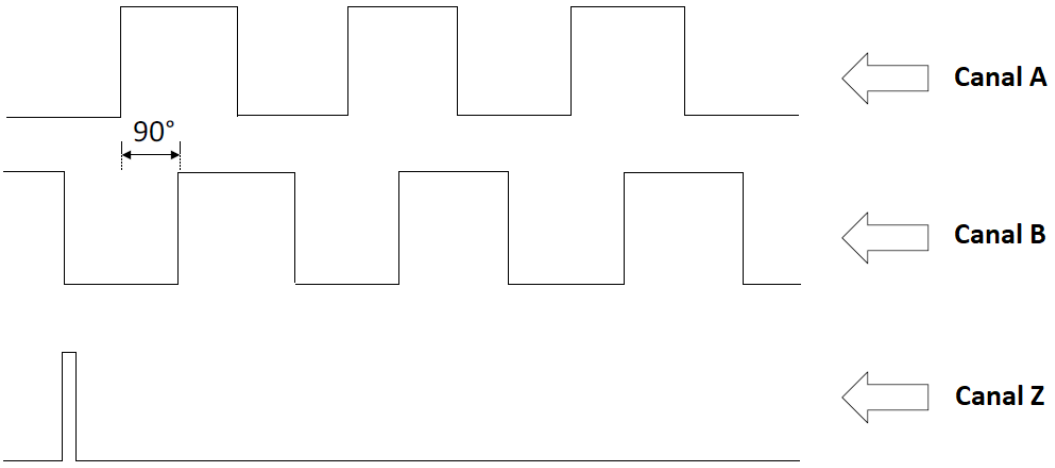


Figura 54. Representación gráfica de los canales A, B y Z

El encoder incremental proporciona normalmente dos formas de ondas cuadradas y desfasadas entre sí 90º eléctricos, los cuales por lo general son “canal A” y “canal B” [14]. El encoder mide la posición en tics, sin embargo, conociendo la resolución (tics por vuelta), es posible calcular el ángulo mecánico. A parte de la posición también es posible conocer la velocidad de rotación y el sentido de giro del motor. Algunos encoders, como el que se va a usar, también disponen de otra señal denominada canal Z o Cero, que proporciona la posición absoluta de cero del eje del encoder.

La precisión de un encoder incremental depende de factores mecánicos y eléctricos entre los cuales, e error de división del retículo, la excentricidad del disco, la de los rodamientos, el error introducido por la parte electrónica de lectura, etc.

La unidad de medida para definir la precisión de un encoder es el grado eléctrico, este valor se determina a partir de la división de 360º eléctricos que corresponde a una rotación mecánica entre el número de pulsos que el encoder tenga.

$$Grado\ eléctrico = \frac{360º\ mecánicos}{Nº\ pulsos} \qquad (46)$$

6.2 Motor

El motor de la marca GARTT model ML4108 es el que se va a usar para realizar este proyecto. Es un motor sin escobillas outrunner, formado por un rotor de imanes permanentes y un estator formado por 3 bobinas. La característica de outrunner implica que los bobinados se disponen en la parte interior, mientras que los imanes están situados en una campana exterior que rodea al bobinado y a la que se conecta el eje. Este tipo de motores producen menor número de revoluciones, pero posee mayor par motor.

Figura 55. Motor sin escobillas ML4108

Las características del motor se muestran a continuación:

Motor KV	500KV RPM/V
Resistencia del motor (RM)	0,1088Ω
Corriente de reposo (Io/10V)	0,6A/10V
Corriente máxima en continua	23A
Máxima potencia en continua	530W
Número de brazos de estator	24
Número de polos	22

Tabla 3. Características del motor

6.3 Software

En este apartado se describirá la estructura que se ha seguido para desarrollar el programa encargado de controlar el motor y además se indicará los programas utilizados.

6.3.1 Diagrama de flujo

En la figura 56 se muestra el diagrama de estados de la ejecución del firmware.

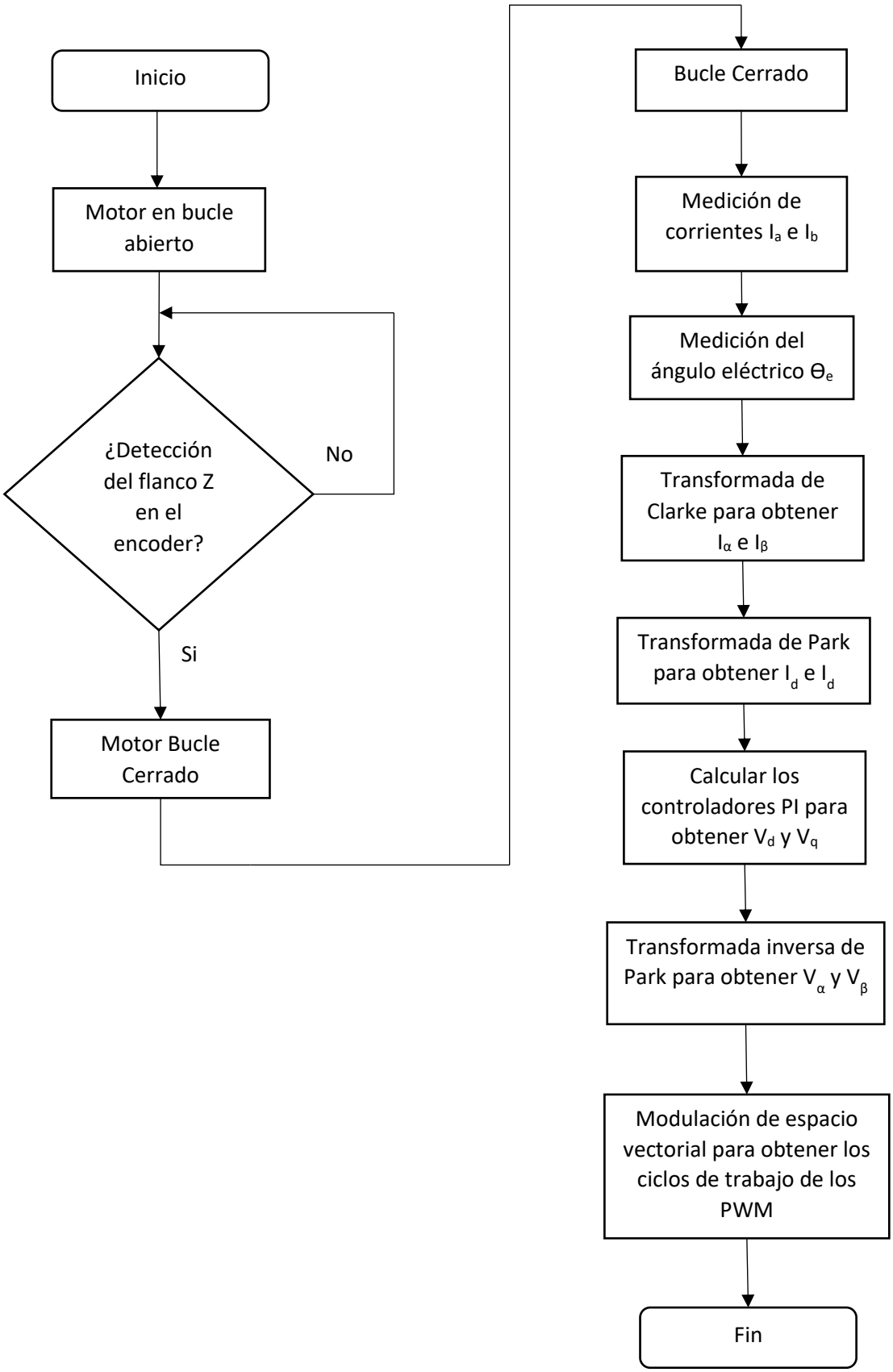


Figura 56. Diagrama usado para la programación del Control Vectorial

6.3.2 Bucle abierto

Tal como se ha indicado en el diagrama de flujo, el primer paso consiste en mover el motor en bucle abierto, es decir, sin tener en cuenta los sensores del motor. Para ello, se va a emplear el programa MATLAB, que va a generar tres vectores cuyos datos representen tres ondas sinusoidales desfasadas 120º. El programa se muestra a continuación:

```
Ns = 50;            %Número de muestras
Rs = 8;             %Resolución

T = 0:((2*pi/(Ns-1))):(4*pi);
%Sine waves
Y1 = sin(T);
Y1 = Y1+1;
Y1 = Y1*((2^Rs-1))/(2);
Y1 = round(Y1);

Y2 = sin(T + ((2*pi)/3));
Y2 = Y2 + 1;
Y2 = Y2*((2^Rs-1))/(2);
Y2 = round(Y2);

Y3 = sin(T + ((4*pi)/3));
Y3 = Y3 + 1;
Y3 = Y3*((2^Rs-1))/(2);
Y3 = round(Y3);

fprintf('%d, %d,%d, %d,%d, %d,%d, %d,%d, %d,%d, %d,%d, %d,%d, \n', Y1);
fprintf('%d, %d,%d, %d,%d, %d,%d, %d,%d, %d,%d, %d,%d, %d,%d, \n', Y2);
fprintf('%d, %d,%d, %d,%d, %d,%d, %d,%d, %d,%d, %d,%d, %d,%d, \n', Y3);
plot(T,Y1,T,Y2,T,Y3);
grid

```

Al establecer una resolución de 8 bits, el valor máximo que se va a obtener como punto de muestreo de las ondas es de 255. Este valor se va a emplear más adelante para determinar la resolución de los ADCs que se van a usar para la lectura de la corriente. Se ha elegido este valor para demostrar que no es necesario emplear un ADC con una resolución alta de 12 ó 16 bits para poder realizar el control vectorial, sino que con un ADC de 8 bits o incluso uno de 6 bits puede funcionar perfectamente.

6.3.3 STM32CubeMX

Con el objetivo de facilitar la programación de los pines necesarios para el desarrollo del control FOC se ha decidido usar el programa STM32CubeMX que permite configurar cada uno de los puertos de una manera más sencilla. Además, es capaz de generar el código de inicialización, los controladores STM32 HAL compatibles, las pilas de middleware necesarias para la configuración del usuario y todos los archivos necesarios para abrir y construir el proyecto en el IDE seleccionado.

En los siguientes apartados se muestran los pasos necesarios para crear la base del programa. Para ello, una vez que se ha abierto el programa el primer paso consiste en pulsar en “**New project**” y después en la ventana de “**Board Selector**” seleccionamos la placa que se va a usar.

Figura 57. Selección de la placa

Una vez seleccionado la placa, en la siguiente ventana aparece el encapsulado del microcontrolador seleccionado.

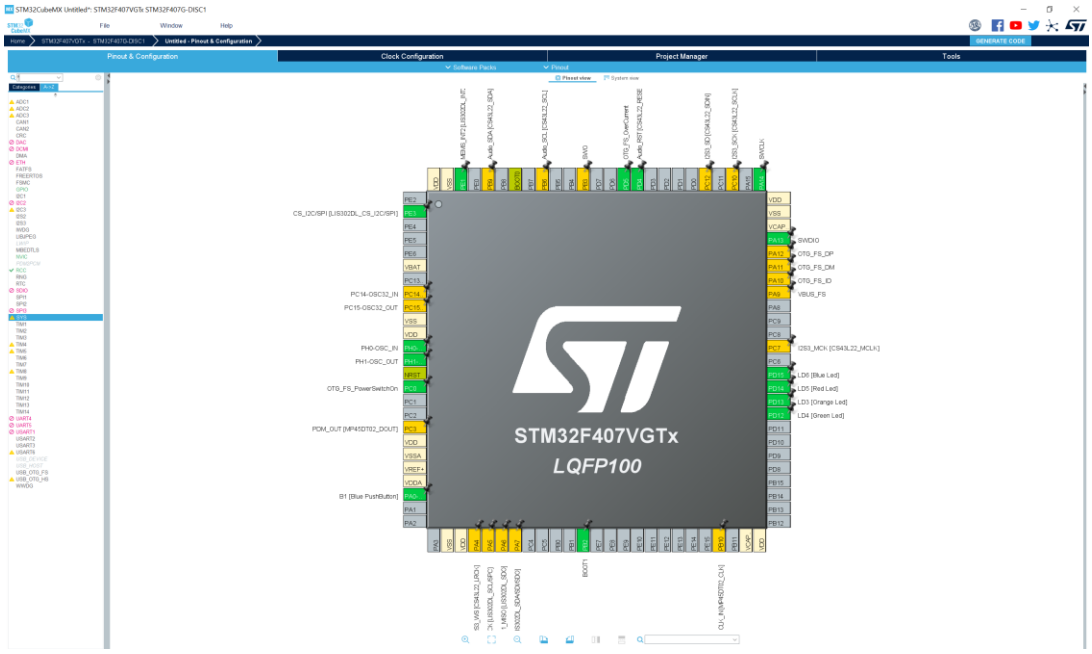


Figura 58. Entorno de programación del microcontrolador

Haciendo clic sobre cada uno de los pines, se selecciona la funcionalidad que se le quiere dar, ya sea de entrada/salida de propósito general o la de un periférico concreto. Para este proyecto se va a necesitar dos ADCs, un pin de interrupción y tres Timers (los microcontroladores STM32, poseen una configuración específica para los encoders que se encuentra en los timers, su configuración se va a realizar a través de registros, por lo que no se verá en este apartado).

En primer lugar, se va a configurar el TIM3 para que genere los tres PWMs que se encargarán de mover el motor BLDC, tanto en bucle abierto como cerrado. Para calcular la frecuencia de funcionamiento del timer se puede emplear la siguiente fórmula:

$$f_{CLK\_CNT} = \frac{f_{APBx}}{(TIM_{ARR} + 1) * (Prescaler + 1)} \qquad (47)$$

donde:

- f_{APBx} depende del periférico al que esté conectado el timer, por ejemplo, TIM3 está conectado a f_{APB1}.
- TIM_{ARR} representa el valor máximo el cual el timer va a contar.
- Presecaler, como su nombre indica, es un preescalar que se usa para dividir la frecuencia de f_{APB1}.

La frecuencia de funcionamiento de este timer se ha establecido a 15kHz por ello si establecemos que el valor máximo es de 255, se ha elegido este valor para que coincida con la resolución de 8 bits que se ha visto en el apartado anterior, obtenemos que el valor del Prescarler es igual a 10.

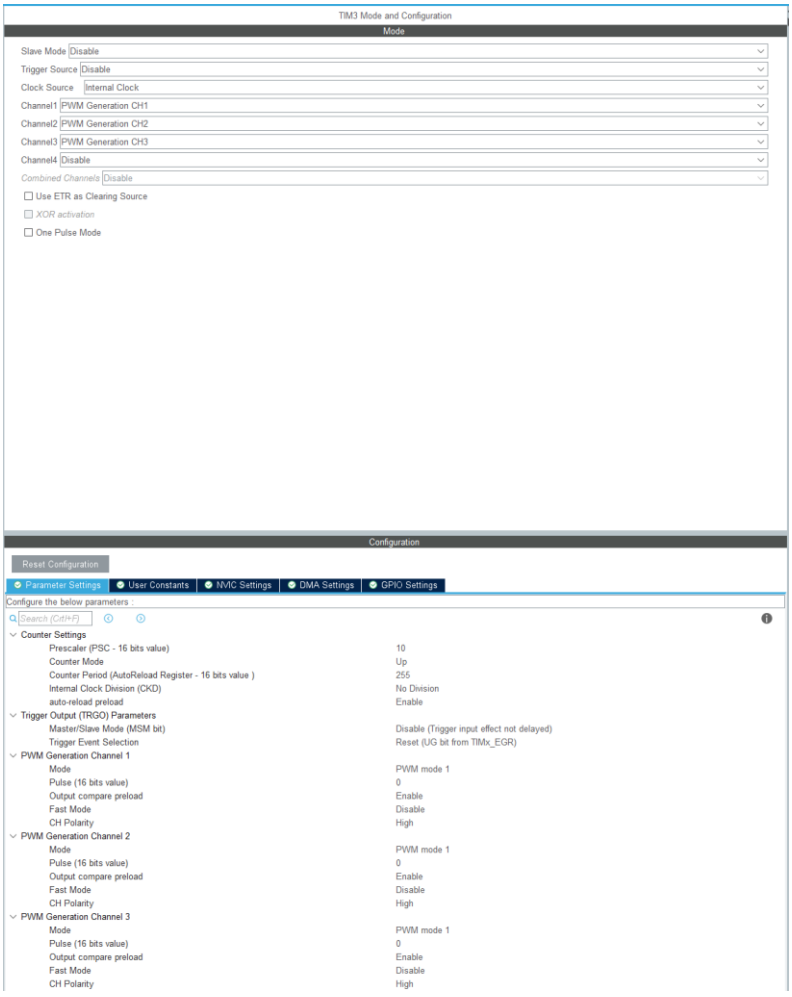


Figura 59. Configuración del TIM3

El segundo paso consiste en configurar dos ADCs para que midan la corriente de dos de las tres fases cada cierto tiempo. Para ello se selecciona el pin IN2 del ADC1 e IN3 del ADC2, la razón por la cual se han seleccionado dos ADCs en vez de seleccionar dos pines del mismo ADC, es porque el ADC funciona como un multiplexor, es decir para leer el segundo valor se debe de esperar hasta que se haya leído el primero, por lo que la lectura de las dos corrientes podría estar desfasada y el objetivo es leer ambas en el mismo instante, por ello se han seleccionado dos ADCs. Además, la resolución de ambos ADCs se ha establecido a un valor de 8 bits, tal como se ha comentado en el apartado 6.3.2.

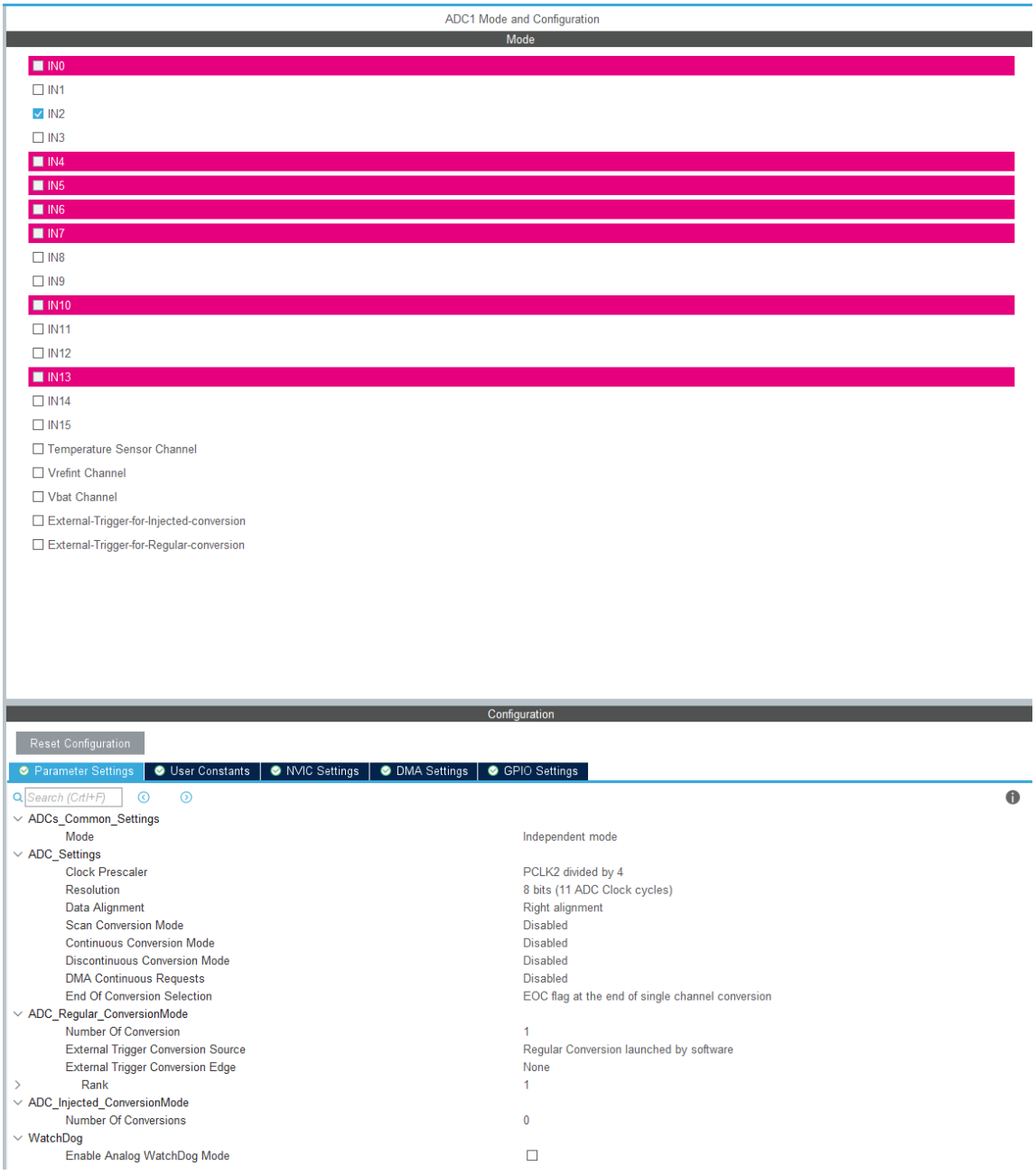


Figura 60. Configuración del ADC1

En la configuración del ADC únicamente se selecciona el pin del ADC que se va a usar y la configuración se deja con los valores por defecto. No se muestra la configuración del ADC2 ya que es idéntica al ADC1.

Una vez configurados ambos ADCs, el siguiente paso consiste en asegurar que ambos reciban la señal de la corriente al mismo tiempo, para ello se va a usar el TIM4 para que "*interrumpa*" el programa cada cierto periodo y mida la corriente. Para ello se va a configurar el timer del siguiente modo.

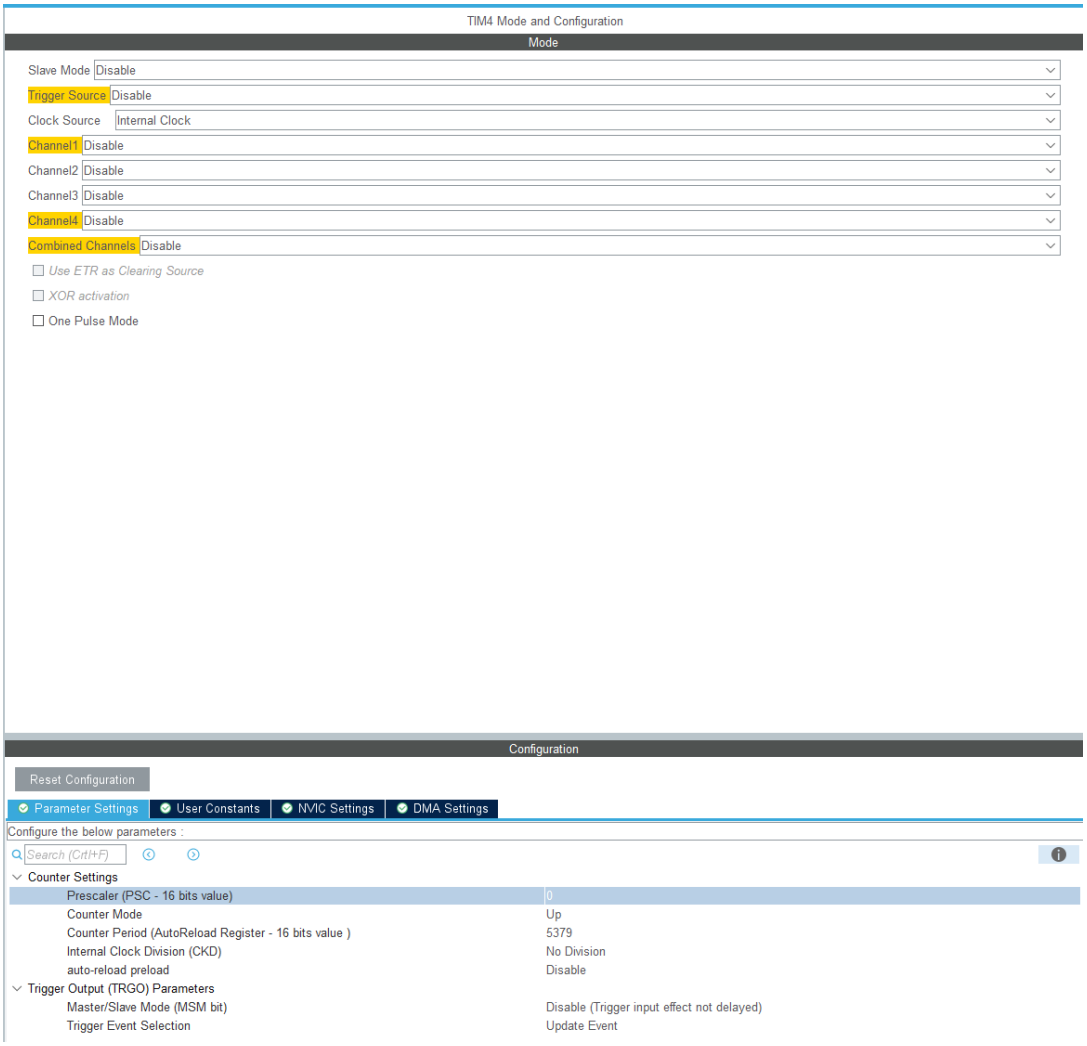


Figura 61. Configuración del TIM4

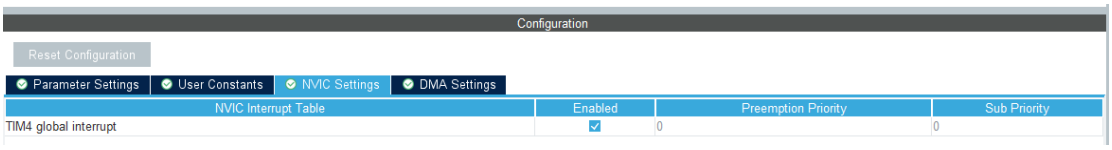


Figura 62. Habilitar el timer como interrupción

El TIM4 se ha configurado de tal manera que tenga la misma frecuencia que el TIM3. Por ello siguiendo la ecuación 47 y teniendo en cuenta que el TIM3 funciona a 15kHz y que ambos timers están conectados al mismo periférico de reloj, f_{APB1}, el valor de ARR que se obtiene si PSC es igual a 0 es de 53799.

Una vez configurados todos los parámetros necesarios, se procede a generar el código fuente que servirá de esqueleto del proyecto. Para ello se va al apartado de "**Project Manager**" y se establece que el IDE que se va a usar es MDK-ARM.

Al terminar el proceso, la aplicación STM32CubeMX genera la siguiente estructura de directorios:

- Core
- Drivers
- MDK-ARM

Dentro del directorio "MDK-ARM" se encuentra el fichero con la extensión "**.uvprojx**" que abre el proyecto en Keil µVision5.

6.3.3 Keil µKeil 5 IDE

Una vez abierto el proyecto, el primer paso consiste en generar el código necesario para que el motor se mueva en bucle abierto. Para ello, se debe de almacenar en tres vectores los valores muestreo obtenidas en MATLAB.

```
/*Ondas senoidales*/
/*Onda senoidal desfasada 0°*/
uint32_t pwmSin1[] = {128, 111,95, 80,65, 51,39, 28,18, 11,5, 2,0, 1,3,
8, 14,23, 33,45, 58,72, 87,103, 119,136, 152,168, 183,197,
210, 222,232, 241,247, 252,254, 255,253, 250,244, 237,227, 216,204,
190, 175,160, 144,128};

/*Onda senoidal desfasada 120°*/
uint32_t pwmSin2[] = {238, 245,250, 254,255, 254,251, 246,239, 230,220, 208,195, 180,165,
149, 133,117, 100,85, 70,56, 43,31, 21,13, 7,3, 0,0,
2, 6,12, 20,30, 41,53, 67,82, 98,114, 130,147, 162,178,
192, 206,218, 229,238};

/*Onda senoidal desfasada 240°*/
uint32_t pwmSin3[] = {17, 26,37, 49,63, 77,93, 108,125, 141,157, 173,188, 202,214,
225, 235,243, 249,253, 255,255, 252,248, 242,234, 224,212, 199,185,
170, 155,138, 122,106, 90,75, 60,47, 35,25, 16,9, 4,1,
0, 1,5, 10,17};
```

Una vez declaradas, el siguiente paso consiste en introducir cada uno de estos valores en el registro "**capture/compare register**" del TIM3.

```
  while (1)
  {

    if(FirstLoop == 0){
      TIM3->CCR1 = pwmSin1[currentStep]*0.15;
      TIM3->CCR2 = pwmSin2[currentStep]*0.15;
      TIM3->CCR3 = pwmSin3[currentStep]*0.15;

      currentStep += 1;

      if(currentStep >= 50){
        currentStep = 0;
      }
      WaitForAMoment(20000);
    }
  }

}
void WaitForAMoment(int Moment)
{
  volatile int i, j=0;

  for(i = 0; i < Moment; i++)
  {
    j++;
  }
}
```

Tal como se observa, mediante el comando ***TIM3->CCRx*** se introduce el ciclo de trabajo de cada uno de los PWMs, además este valor se multiplica por 0.15, con el objetivo de reducir el par del motor.

Una vez que se han utilizado todos los valores se realiza un ***reset*** de la variable ***currentStep*** para que utilice de nuevo todos los valores.

La razón por la cual se necesita un bucle abierto al principio del control es debido a que el encoder desconoce cuál es su posición inicial, ya que al arrancar el programa el contador del encoder comienza a incrementar su valor a medida que el motor va girando. Con el objetivo de tener un punto de referencia al que se pueda llamar posición inicial se va a utilizar el canal Z del encoder, el cual genera un único pulso cada 360º.

El bucle abierto se utiliza para mover el motor a una velocidad constante hasta que el microcontrolador detecte un flanco de bajada en el canal Z y con ello pueda resetar el contador. De este modo el encoder está sincronizado y el programa ya puede pasar al bucle cerrado.

```
/*Interrupción para resetear el Encoder*/
void EXTI4_IRQHandler(void)
{
  if(EXTI->PR & EXTI_PR_PR4)
    {

      FirstLoop = 1;
      TIM1->CNT = 0;
      EXTI->PR |= EXTI_PR_PR4;
    }
}
```

Esta parte del código se encarga de detectar flanco de bajada en el pin PC4, al cual está conectado el canal Z y una vez detectado resetea el contador e iguala el valor de *FirstLoop* a 1, por lo que ya no se vuelve a entrar en el bucle abierto.

```
static void init_hardware_encoder(void)
{
      //Encoder A and B
      RCC->AHB1ENR |= RCC_AHB1ENR_GPIOEEN;                          //Habilita el puerto E
      RCC->APB2ENR |= RCC_APB2ENR_TIM1EN;                           //Habilita el TIM1

      //PA11 and PA9
      GPIOE->MODER |= (GPIO_MODER_MODE9_1 | GPIO_MODER_MODE11_1);   //Modo de función alternativo
      GPIOE->AFR[1] = 0x1010;                                       //Habilita TIM1_CH1 y TIM1_CH2

      TIM1->ARR = 0xFFFF;                                           //Valor maximo permitido por el timer

      TIM1->CCMR1 |= (TIM_CCMR1_CC1S_0 | TIM_CCMR1_CC2S_0);         //Canal 1 y 2 como entradas
      TIM1->SMCR |= (TIM_SMCR_SMS_1);                               //Encoder modo 2
      TIM1->CR1 |= TIM_CR1_CEN;                                     //El contador de TIM1 habilitado

       //Interrupcion del canal Z en el pin PC4
      RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN;                          //Habilita el puerto C
      GPIOC->MODER &= ~(GPIO_MODER_MODE4);                          //Pin 4 como input
      GPIOC->PUPDR &= ~(GPIO_PUPDR_PUPD4);                          //Sin resistencias de pull-up o pull-down

      SYSCFG->EXTICR[1] = SYSCFG_EXTICR2_EXTI4_PC;                  //Selección de la fuente de input para el EXTIx
      EXTI->IMR |= EXTI_IMR_MR4;                                    //Habilita la interrupción
      EXTI->FTSR |= EXTI_FTSR_TR4;                                  //Modo flanco de bajada

      NVIC_EnableIRQ(EXTI4_IRQn);                                   //Habilita la interrupcion en NVIC
      NVIC_SetPriority(EXTI4_IRQn, 0);                              //Selecciona un nivel de prioridad alto
}
```

Esta parte del código presenta la configuración del encoder, la cual se encarga de leer la señal de los canales A y B, al leer la señal en los dos canales la precisión en la medición del ángulo aumenta el doble, permitiendo medir la posición de una forma mucho más precisa. En el apartado 6.1.5 se mostró el encoder que se va a usar y este presenta 360 pulsos por rotación, por lo que, tiene una precisión de 1º, sin embargo, al haber configurado el encoder de tal modo que se lean ambas señales se obtienen 720 pulsos por rotación por lo que se ha pasado a tener una precisión de 0.5º. Además, se ha realizado la configuración de los registros necesarios para habilitar el pin PC4 como interrupción.

Una vez configurado el encoder y reseteado el contador mediante el pin PC4, queda como último paso leer el valor de la corriente de dos de las tres fases y realizar las transformadas.

```
204  /*Lectura de los valores del ADC*/
205  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
206  {
207      Park result;
208
209      HAL_ADC_Start(&hadc1);
210      HAL_ADC_Start(&hadc2);
211
212      if(HAL_ADC_PollForConversion(&hadc1, 1) ==  HAL_OK){
213          Ia = HAL_ADC_GetValue(&hadc1);
214        }
215
216      if(HAL_ADC_PollForConversion(&hadc2, 1) ==  HAL_OK){
217          Ib = HAL_ADC_GetValue(&hadc2);
218        }
219
220      HAL_ADC_Stop(&hadc1);
221      HAL_ADC_Stop(&hadc2);
222
223      Ia = (Ia - 140);  //Eliminar offset
224      Ib = (Ib - 140);
225
226      encoder = TIM1->CNT;                          //Lectura del encoder
227      angulo = (float)encoder/2.0f;                 //Transformacion a angulo mecanico
228      theta = fmod(angulo*11.0f, 360.0f);           //Transformacion a angulo electrico
229
230      result =  transformation(Ia, Ib, theta);      //Transformada de Clarke y Park
231
232      Ud = PI_controlD(0.0f, result.Id);            //Controlador PI para el eje D
233      Uq = PI_controlQ(100.0f, result.Iq);          //Controlador PI para el eje Q
234
235      /*With SVM*/
236      arm_inv_park_f32(Ud, Uq, &Ualpha, &Ubeta, result.sine, result.cos); //Transformada inversa de Park
237
238      sSVPWM.fUdcCCRval = Uq;                       //Situar el valor Uq como el valor máximo del SVM
239      sSVPWM.fUal = Ualpha;
240      sSVPWM.fUbe = Ubeta;
241      sSVPWM.m_calc(&sSVPWM);                       //Enviar los valores al módulo svpwm.c
242
243
244      if (FirstLoop == 1){
245        /*Igual los valores calculados en el modulo svpwm.c a los pines del timer 3*/
246        TIM3->CCR1 = sSVPWM.fCCRB;
247        TIM3->CCR2 = sSVPWM.fCCRC;
248        TIM3->CCR3 = sSVPWM.fCCRA;
249      }
250  }
```

Esta función representa la interrupción del TIM4 que se ha configurado en el apartado anterior, la cual se ejecuta cada 0.06ms. En primer lugar se leen los valores del ADC1 y ADC2, para ello se inicializan los ADCs con la función ***HAL_ADC_Start()*** después se espera a que se complete la conversión mediante la función ***HAL_ADC_PollForConversion()*** el cual devolverá un HAL_OK cuando se haya completado y una vez terminado se recogen los valores a través de la función ***HAL_ADC_GetValue()*** y como último paso se detiene el ADC, ***HAL_ADC_Stop()***. La placa DRV8302, tal como se mostró anteriormente, envía el valor de corriente en voltios con un rango de 0V - 3V, sin embargo, para aplicar correctamente las transformadas es necesario eliminar el offset, por ello se realiza una resta en las líneas 223 y 224 antes de aplicar las transformadas.

Otra característica importante por conocer es el ángulo eléctrico del motor, para ello según el manual de referencia de la placa de desarrollo STM32F407-Discovery se tiene que acceder al contador del timer donde se haya configurado el encoder. Sin embargo, el valor que proporciona varía entre 0 y 720 por lo que es necesario realizar una transformación y convertirlo en ángulo eléctrico.

Una vez obtenidos los valores de las dos corrientes y el ángulo eléctrico, el siguiente paso consiste en realizar las transformadas de Clarke y Park, que se realizan mediante la función ***transformation()***.

```
/*Transformada de Clarke y Park*/
Park transformation(float32_t Ia, float32_t Ib, float32_t angulo){

  float32_t Ialpha, Ibeta;
  float32_t Id, Iq;

  Park p;

  arm_clarke_f32(Ia, Ib, &Ialpha, &Ibeta);


  arm_sin_cos_f32(angulo,&sinevalue, &cosinevalue);

  arm_park_f32(Ialpha, Ibeta, &Id, &Iq, sinevalue, cosinevalue);

  p.Id = Id;
  p.Iq = Iq;
  p.sine = sinevalue;
  p.cos = cosinevalue;

  return(p);
}
```

Una vez obtenidos los valores I_d e I_q el siguiente paso consiste en aplicar dos controladores PI para ajustar la salida del motor a los valores de referencia. En este caso, se ha seleccionado el valor de referencia de I_q = 100.0 (tal como se ha explicado en apartados anteriores el valor de I_d se debe de mantener a 0).

```
/*Definiciones*/
#define OUTMIND 1.0f
#define OUTMAXD 255.0f
#define ERRORMIND 1.0f
#define ERRORMAXD 1.0f
#define KpD 0.1f
#define KiD 0.01f

#define OUTMINQ 0.0f
#define OUTMAXQ 255.0f
#define ERRORMINQ 3.0f
#define ERRORMAXQ 3.0f
#define KpQ 1.42f
#define KiQ 0.6f
```

Los valores K_p y K_i de los dos controladores se han obtenido mediante el método de iteración, ya que no se ha tenido acceso a los valores de inductancia del motor.

```
/*PI Control del eje D*/
float32_t PI_controlD(float32_t refD, float32_t realD){

  float32_t out, pi_error;

  pi_error = refD - realD;
  error_sumD += pi_error;
  /*Limitar el error*/
  if(error_sumD < ERRORMIND){
    error_sumD = ERRORMIND;
    }
  else if(error_sumD > ERRORMAXD){
    error_sumD = ERRORMAXD;
    }

    out = KpD*(pi_error + (KiD*error_sumD));

    /*Limitar el output*/
  if(out < OUTMIND){
    out = OUTMIND;
    }
  else if(out > OUTMAXD){
    out = OUTMAXD;
    }
  return(out);
}
```

```
/*PI Control del eje Q*/
float32_t PI_controlQ(float32_t refQ, float32_t realQ){

  float32_t out, pi_error;

  pi_error = refQ - realQ;
  error_sumQ += pi_error;
  /*Limitar el error*/
  if(error_sumQ < ERRORMINQ){
    error_sumQ = ERRORMINQ;
    }
  else if(error_sumQ > ERRORMAXQ){
    error_sumQ = ERRORMAXQ;
    }

    out = KpQ*(pi_error + (KiQ*error_sumQ));

    /*Limitar el output*/
  if(out < OUTMINQ){
    out = OUTMINQ;
    }
  else if(out > OUTMAXQ){
    out = OUTMAXQ;
    }

  return(out);

}
```

Finalmente, una vez aplicados los controladores PI, se realiza la transformada inversa de Park y mediante los valores de tensión de U_α y U_β se introducen en el módulo ***svpwm.c*** y a partir de este se obtienen los ciclos de trabajo de los tres PWMs conectados al TIM3.

Función del módulo ***svpwm.c*** para calcular el ciclo de trabajo de los tres PWMs.

```
void tSVPWM_calc(tSVPWM* ptSVPWM)
{
  uint8_t u8Sector;
  float fMaxUs, fScaledUs, fBeta, fTb1, fTb2, afTi[4];

  fMaxUs = ptSVPWM->fUdc * (1.0f/sqrtf(3.0f));

  switch(ptSVPWM->enInType)
  {
    case AlBe:
      ptSVPWM->fUs = hypotf(ptSVPWM->fUbe, ptSVPWM->fUal);
      ptSVPWM->fAngRad = atan2f(ptSVPWM->fUbe, ptSVPWM->fUal);
    break;

    case UsAng:
      ptSVPWM->fUs = fabsf(ptSVPWM->fUs);
    break;

    default: return;
  }

  if(ptSVPWM->fUs > fMaxUs) ptSVPWM->fUs = fMaxUs;

  fScaledUs = ptSVPWM->fUs/fMaxUs;

  ptSVPWM->fAngRad += M_Pi;

  u8Sector = (uint8_t)(ptSVPWM->fAngRad * (1.0f/M_Pi_3));

  fBeta = ptSVPWM->fAngRad - M_Pi_3 * u8Sector;

  fTb1 = fScaledUs * sinf(M_Pi_3 - fBeta);
  fTb2 = fScaledUs * sinf(fBeta);

  afTi[0] = (1.0f - fTb1 - fTb2)*0.5f;
  afTi[1] = fTb1 + fTb2 + afTi[0];
  afTi[2] = fTb2 + afTi[0];
  afTi[3] = fTb1 + afTi[0];

  ptSVPWM->fCCRA = ptSVPWM->fUdcCCRval * afTi[au8PermuataionMatrix[u8Sector][0]];
  ptSVPWM->fCCRB = ptSVPWM->fUdcCCRval * afTi[au8PermuataionMatrix[u8Sector][1]];
  ptSVPWM->fCCRC = ptSVPWM->fUdcCCRval * afTi[au8PermuataionMatrix[u8Sector][2]];
}
```

6.3 Resultados del experimento

Una vez mostrado todas las partes que contiene el control vectorial, se procede a su montaje, el cual se puede observar en la figura 63.

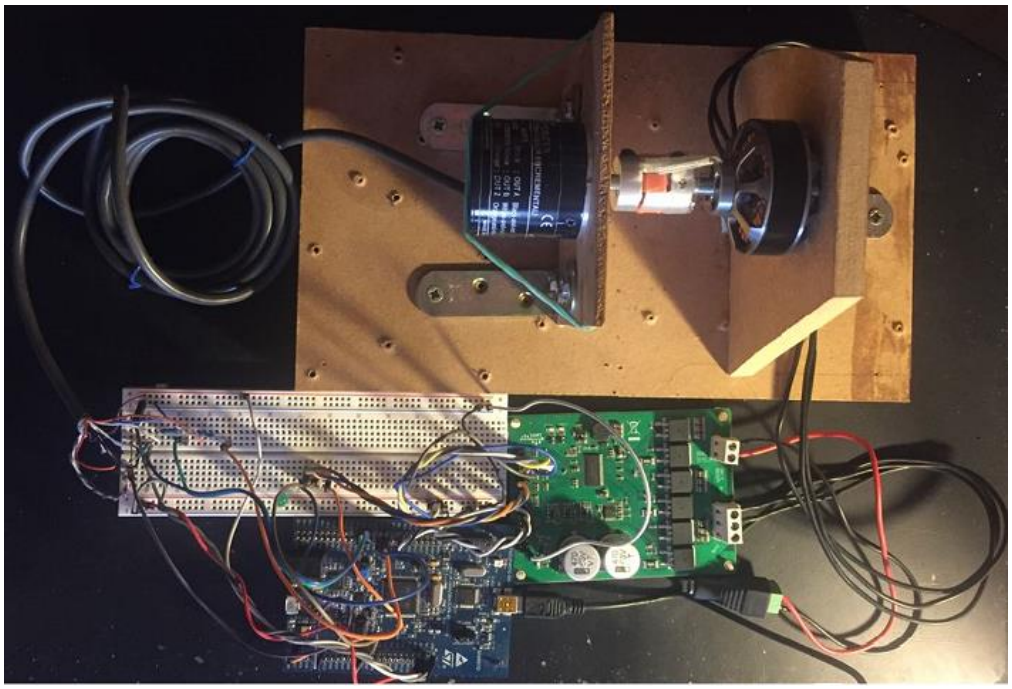


Figura 63. Montaje del proyecto de Control Vectorial

Después de montar el motor junto al encoder y las dos placas, el siguiente paso consiste en exponer los resultados que se han obtenido. Para ello se va a usar el programa STM Studio, que permite mostrar, en tiempo real, la evolución de las variables del programa.

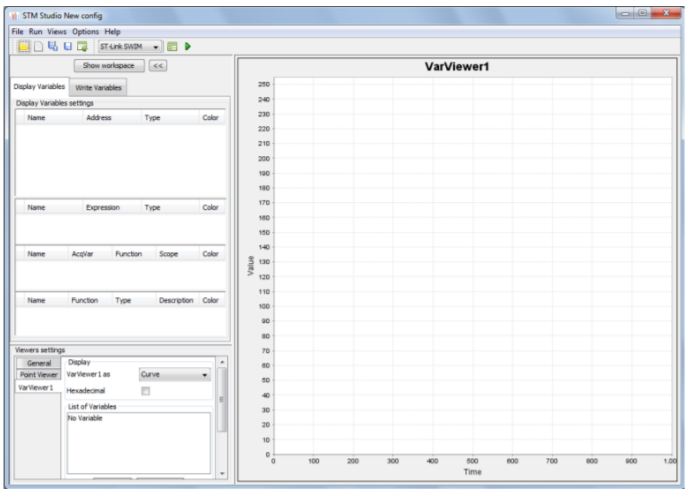


Figura 64. Interface del programa STM Studio

Tal como se explicó anteriormente, el programa debe de funcionar en bucle abierto para que el encoder incremental se pueda sincronizar con el motor. Una vez conseguido, se comienza a recibir los valores analógicos de dos de las tres corrientes del motor y el valor del encoder en cada instante, que posteriormente se usará para crear las variables *ángulo* y *theta* que representan una vuelta completa mecánica y eléctrica respectivamente.

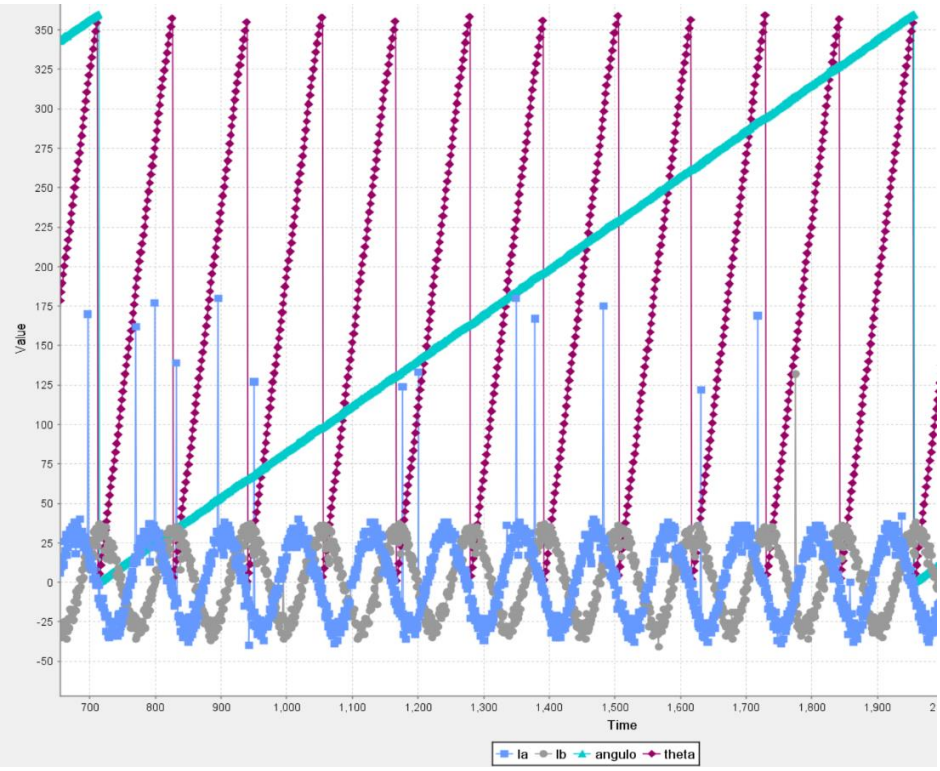


Figura 65. Valores de las corrientes del motor y del encoder

Posteriormente, se realizan las transformadas de Clarke y Park, las cuales se pueden observar en las figuras 66 y 67.

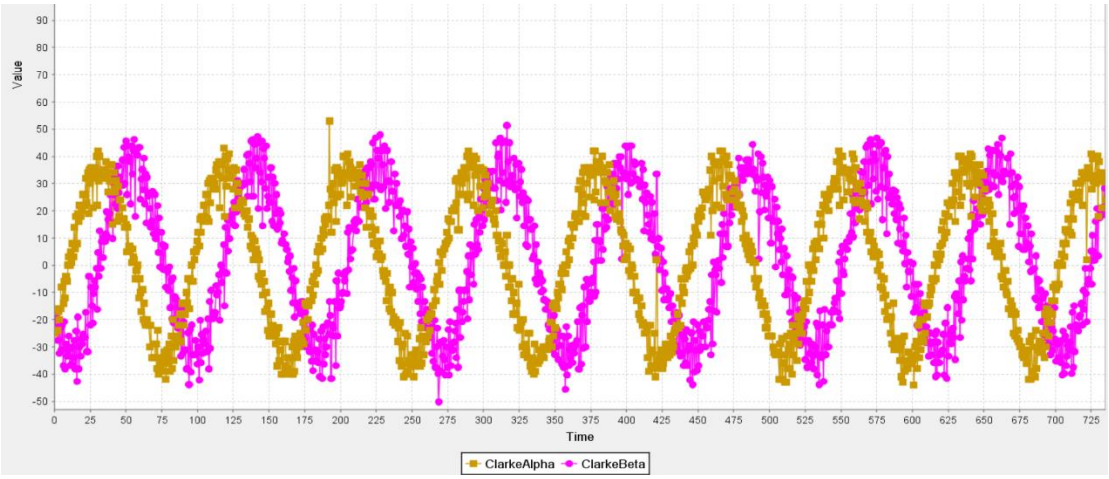


Figura 66. Transformada de Clarke

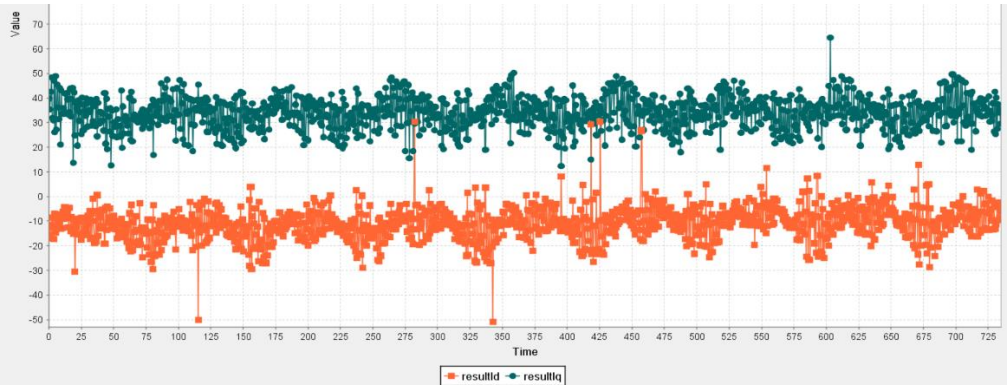


Figura 67. Transformada de Park

Una vez realizada la transformada de Park el siguiente paso consiste en aplicar los controladores PI en las corrientes I_d e I_q, para que el valor de I_d se mantenga a 0 y la I_q al valor deseado, en este caso se ha decidió fijar su valor a 100.

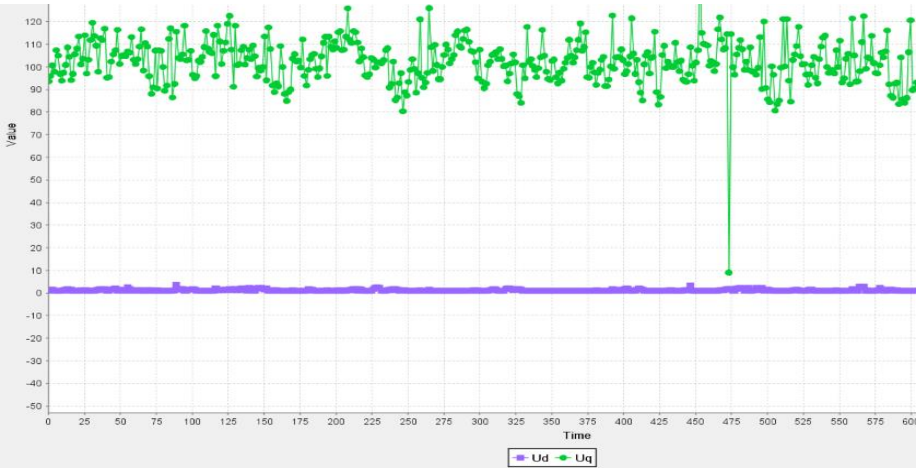


Figura 68. Valores Id e Iq después de los controladores PI

Una vez que se han ajustado los valores de Id e Iq el siguiente paso consiste en realizar la transformada inversa de Clarke y aplicar el SVM. Las señales generadas mediante el SVM son las que se introducen en el motor.

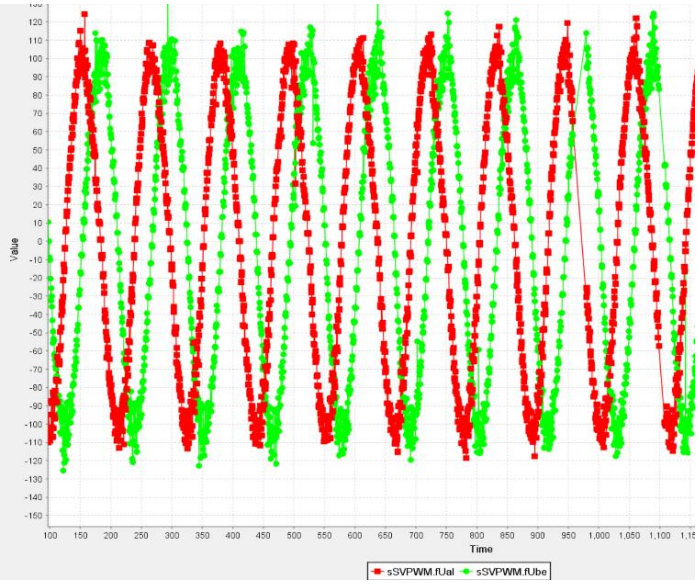


Figura 69. Transformada inversa de Clarke

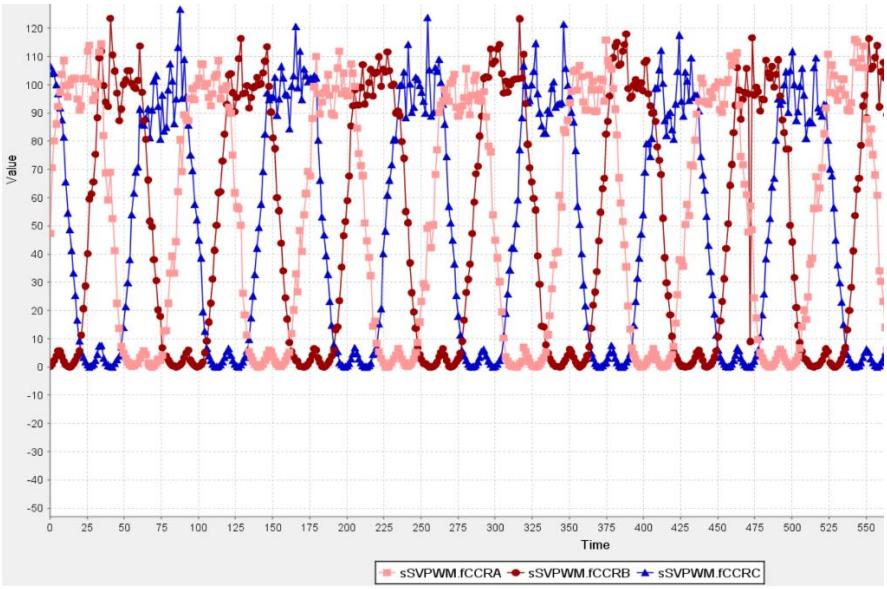


Figura 70. Modulación de espacio vectorial

De este modo se ha podido observar que, mediante el programa descrito en el apartado anterior, cada una de las transformaciones matemáticas funcionan correctamente. Sin embargo, hasta ahora se han mostrado los resultados donde el motor no posee ningún tipo de fuerza de frenado, por ello, se ha aplicado al motor una carga resistiva que dificulte su movimiento y se han recogido los resultados. Para ello, se debe de prestar atención a la evolución de los valores Ud y Uq que son los valores que controlan el comportamiento del motor, además se ha decidido mostrar los valores del SVM para una mejor comprensión.

En primer lugar, tal como se puede observar en la figura 71, se ha introducido una carga de frenado que aumenta lentamente, dicha carga desde el punto de vista del programa significa una disminución del valor Uq y para compensar dicha bajada el programa aumenta el valor de Uq. En las figuras 72 y 73, se muestra el comportamiento del motor en caso de que la carga de frenado tenga un valor constante y en caso de que vaya disminuyendo hasta desaparecer.

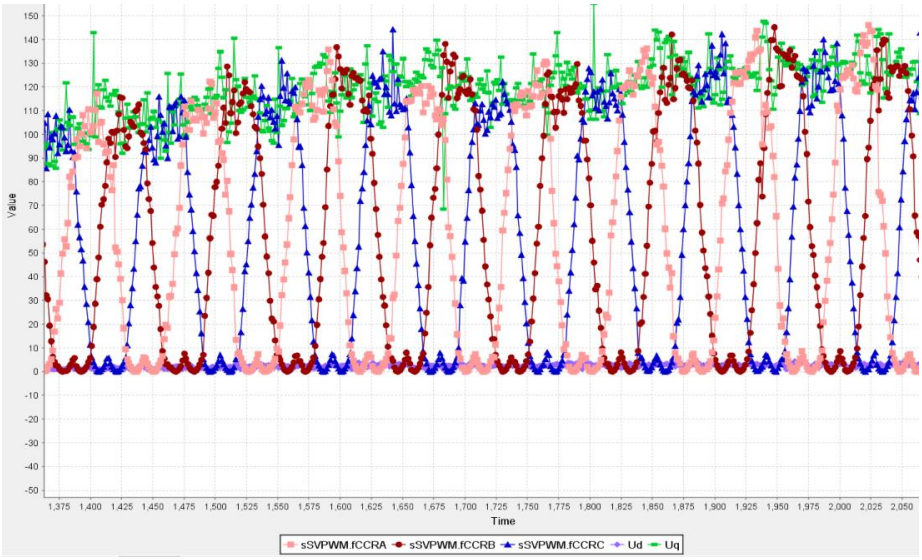


Figura 71. Valores de Ud y Uq cuando la carga de frenado va aumentando

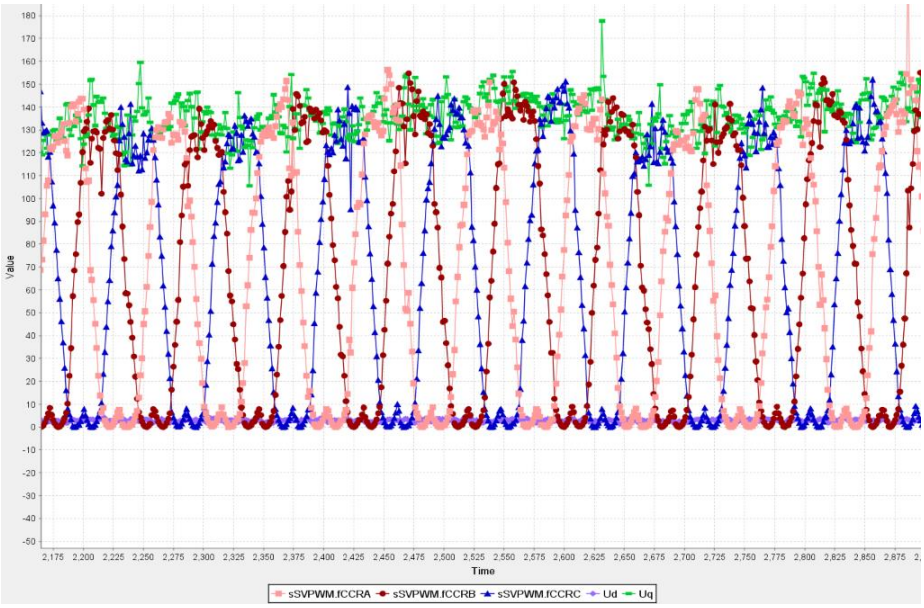


Figura 72. Valores de Ud y Uq con carga de frenado constante

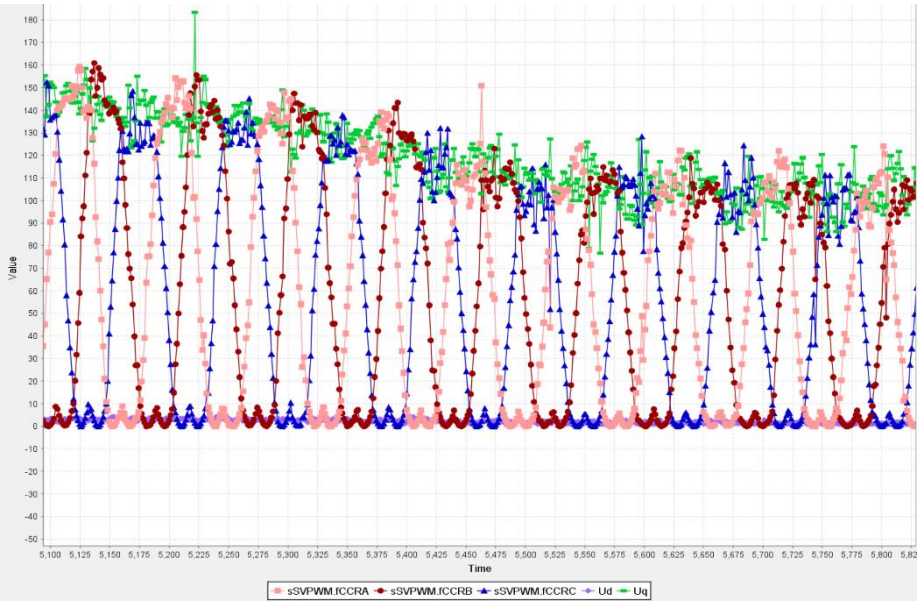


Figura 73. Valores de Ud y Uq con carga de frenado que va disminuyendo

De este modo, se consigue que el motor este continuamente funcionamiento al mismo valor de Uq y en caso de que aparezca una carga de frenado en el motor el programa sea capaz de ajustar rápidamente su valor de Uq para compensar dichas variaciones de frenado.

7. Conclusiones y futuras líneas de trabajo

Finalmente, con la implementación del bucle abierto y cerrado, el proyecto está completo. La solución obtenida cumple con todos los objetivos expuestos al principio del trabajo.

La implementación de módulos y funciones en el programa ha permitido aumentar la fiabilidad del proyecto, ya que ha facilitado la búsqueda de los problemas que han ido surgiendo a lo largo del proyecto. Además, ha facilitado su integración.

Con el objetivo de realizar el control FOC se han necesitado dos tipos de sensores, uno de posición y dos sensores de corriente, el cual estaba implementado en el driver DRV8302. Gracias al encoder incremental se ha podido conocer la posición del motor en todo momento, la cual ha sido usada para la transformada de Park.

Con este metodo se puede controlar la corriente que se le introduce al motor, por lo que, en consecuencia, se controla el par motor del rotor. La evolución del motor en caso de que exista algún tipo de cambio en la carga es suave gracias a los controladores PI implementados, que además evitan las oscilaciones en la corriente, uno de los principales objetivos a controlar.

A pesar de que el controlador vectorial funciona correctamente, se puede llegar a mejorar el programa, como por ejemplo implementado un sistema de compensación del error de la posición del motor, al principio del programa. De este modo no se requeriría que el motor estuviese funcionando en bucle abierto.

Además, se podría diseñar una placa PCB que juntara la placa de desarrollo STM32F407Discovery con el driver DRV8302, ya que uno de los principales problemas que ha surgido es que al mover una de las placas el motor dejaba de funcionar correctamente, y esto es debido a las interferencias que se generaban en la placa de potencia. Por ello uniendo ambas placas se evitarían estos problemas y daría una mayor fiabilidad al montaje.

Por otro lado, otras posibles mejoras que se podrían realizar al proyecto para hacerlo más completo es crear un bucle externo para controlar la velocidad o posición del motor. Esto se puede realizar fácilmente ya que una de las principales ventajas del control vectorial es que es de tipo cascada por lo tanto una vez implementado el control de par motor solo se necesita conocer la velocidad o posición del motor.

A pesar de que el uso del encoder facilita la programación del control vectorial, es verdad que aumenta su coste. Para reducir la fabricación del control FOC se puede desarrollar un control sin sensores que disminuiría el coste y el mantenimiento, pero aumentaría la complejidad. Esto se puede conseguir calculando la posición del motor a partir de las formas de onda de las corrientes en vez de usar el encoder.

Pese a todas estas posibles mejoras que se pueden aplicar al proyecto, el programa actual funciona correctamente y cumple con todas las partes necesarias para demostrar la eficiencia del Control de Campo Orientado (FOC). Además, tal como se ha comentado al principio de este trabajo, esta es una de las mejores técnicas de control para controlar, y al mismo tiempo fácil de implementar, los parámetros de un motor BLDC.

8. Bibliografía

[1] Brushless DC Motor (BLDC) – Construction, Working & Applications: (Consultado el 12/04/2021)

https://www.electricaltechnology.org/2016/05/bldc-brushless-dc-motor-construction-working-principle.html

[2] Brushless DC (BLDC) Motor Fundamentals: (Consultado el 24/04/2021)

https://ww1.microchip.com/downloads/en/appnotes/00885a.pdf

[3] Inrunner: (Consultado el 31/04/2021)

https://en.wikipedia.org/wiki/Inrunner

[4] Modelling and Simulation Analysis of the Brushless DC Motor by using MATLAB (Consultado el 09/05/2021)

https://www.academia.edu/7002076/Modelling_and_Simulation_Analysis_of_the_Brushless DC_Motor_by_using_MATLAB

[5] What is 'Field Oriented Control' and what good is it?: (Consultado el 15/05/2021)

https://www.maccon.de/fileadmin/redaktion/downloads/Produkte/Antriebselektronik/Copley_drives/Field-Oriented-Control.pdf

[6] Control vectorial: (Consultado el 16/05/2021)

https://es.wikipedia.org/wiki/Control_vectorial

[7] Sensores (Encoder-Based) Field Oriented Control of Three-Phse Permanent Magnet Synchronous Motor (PMSM): (Consultado el 20/05/2021)
https://www.microchip.com/content/dam/mchp/documents/OTH/ApplicationNotes/ApplicationNotes/Sensored-Encoder-Based)-Field-Oriented-Control-of-Three-Phase-Permanent-Magnet-Synchronous-DS00002757A.pdf

[8] PROJECT #2 SPACE VECTOR PWM INVERTER: (Consultado el 12/05/2021)

http://www2.ece.ohio-state.edu/ems/PowerConverter/SpaceVector_PWM_Inverter.pdf

[9] Field Oriented Control of 3-Phase AC.Motors: (Consultado el 12/04/2021)

https://www.ti.com/lit/an/bpra073/bpra073.pdf

[10] High Performance Brushless DC Motor Control: (Consultado el 02/05/2021)

https://www.ti.com/lit/an/sprt703/sprt703.pdf

[10] Sensorless Field Oriented Control (FOC) for a Permanent Magnet Synchronous Motor (PMSM) Using a PLL Estimator and Field Weakening (FW): (Consultado el 01/06/2021)

http://ww1.microchip.com/downloads/en/Appnotes/01292A.pdf

[11] Field-oriented Control(Vector Control) for Brushless DC Motors: (Consultado el 12/04/2021)

https://control.com/technical-articles/field-oriented-control-vector-control-for-brushless-dc motors/

[12] Teaching Old Motors New Tricks: (Consultado el 05/06/2021)

https://training.ti.com/teaching-old-motors-new-tricks-part-3-space-vector-modulation-field-weakening-d-q-axis-decoupling?context=1137615-31562-31459

[13] DRV8302 datasheet: (Consultado el 18/05/2021)

https://www.ti.com/lit/ds/symlink/drv8302.pdf?ts=1621349903590&ref_url=https%253A%252F252Fwww.google.com%252F

[14] Encoder incremental: (Consultado el 17/05/2021)

https://www.guemisa.com/sicod/docus/ENCODER-TEC.pdf

Anexos

Anexo 1: Código en MATLAB para la simulación

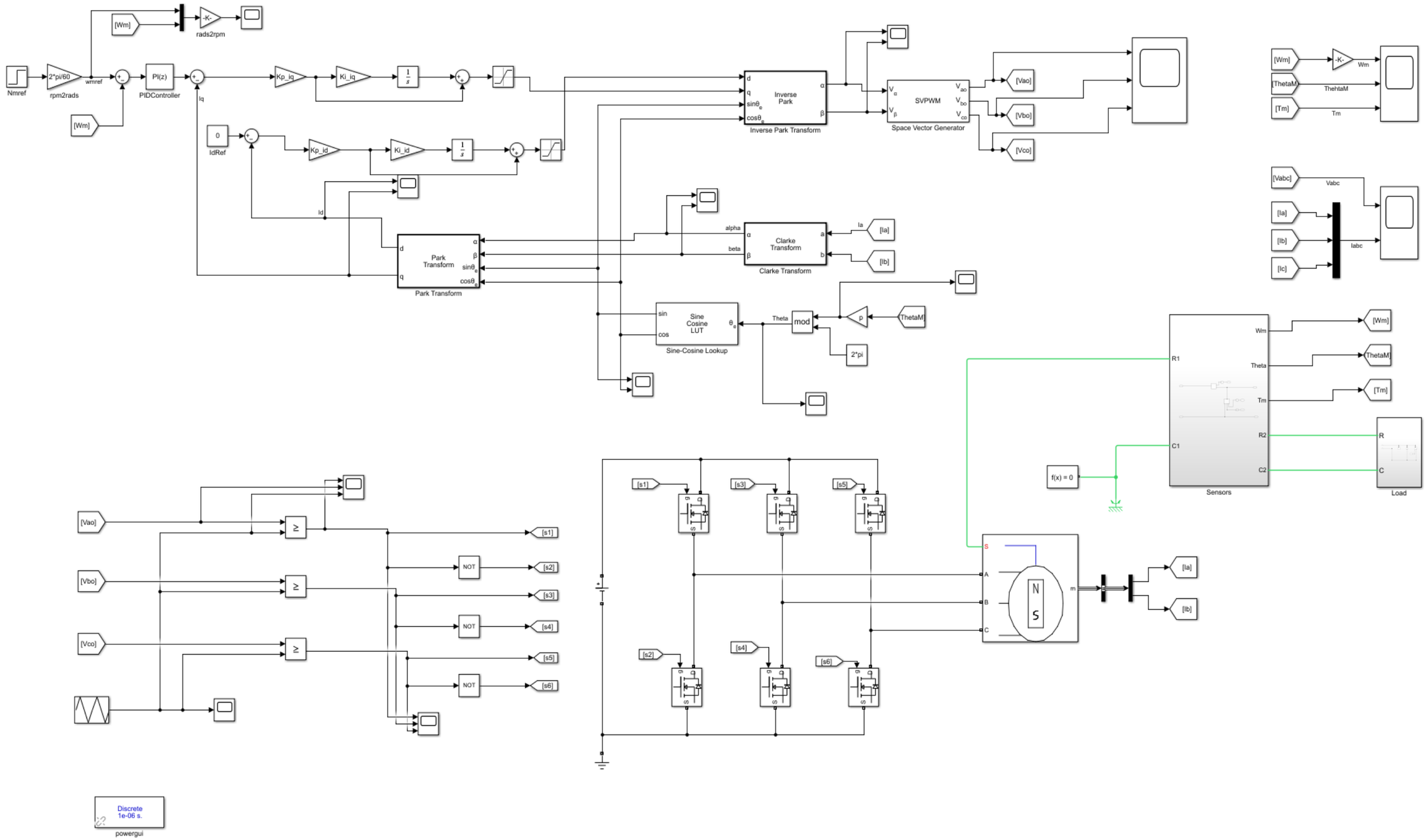
```
%% Parametros del Field Oriented Control para el motor sin
escobillas


%% BLDC
Phi = 0.1;       % Permanent magnet flux linkage [Wb]
Ld = 0.01;       % Stator d-axis inductance [H]
Lq = 0.02;       % Stator q-axis inductance [H]
Rs = 0.38;       % Stator resistance per phase [ƒ¶]
p = 2;           % Number of pole pairs
Jm = 0.1e-3;     % Rotor inertia [kg*m^2]
Dm = 1e-3;       % Rotor damping [N*m/(rad/s)]
%% Inversor
Vdc = 100;       % DC voltage [V]
fc = 10e3;       % Carrier Frequency [Hz]
Ts = 1/fc/100;   % Sampling time [sec]

%% Current controllers of dq-axis (PI controller)
%  Target response time constant: 1e-3[sec]
wc_c = 1e3;        % Target response frequency [rad/s]
Kp_id = wc_c*Ld; % d-axis proportional gain 10 wc_c*Ld
Ki_id = Rs/Ld;  % d-axis integral gain 38  Rs/Ld
Kp_iq = wc_c*Lq;  % q-axis proportional gain 20 wc_c*Lq
Ki_iq = Rs/Lq;  % q-axis integral gain 19 Rs/Lq
% Tcc = 100e-6;      % Sample time of current control [sec]

%% Velocity controller (PI controller)
Kp_s = 0.0330;     % Velocity proportional gain
Ki_s = 0.2925;      % Velocity integral gain
```

Anexo 2: Simulación del Control Vectorial



main.c

```
#include "main.h"
#include "stdint.h"
#include "arm_math.h"
#include "math.h"
#include "svpwm.h"


/*Definitions*/
#define OUTMIND 1.0f
#define OUTMAXD 255.0f
#define ERRORMIND 1.0f
#define ERRORMAXD 1.0f
#define KpD 0.1f
#define KiD 0.01f

#define OUTMINQ 0.0f
#define OUTMAXQ 255.0f
#define ERRORMINQ 3.0f
#define ERRORMAXQ 3.0f
#define KpQ 1.42f
#define KiQ 0.6f

ADC_HandleTypeDef hadc1;
ADC_HandleTypeDef hadc2;

TIM_HandleTypeDef htim2;
TIM_HandleTypeDef htim3;
TIM_HandleTypeDef htim4;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------
---*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
static void MX_ADC2_Init(void);
static void MX_TIM2_Init(void);
static void MX_TIM3_Init(void);
static void MX_TIM4_Init(void);
static void init_hardware_encoder(void);
void WaitForAMoment(int Moment);
float32_t PI_controlD(float32_t refD, float32_t realD);
float32_t PI_controlQ(float32_t refQ, float32_t realQ);


float32_t Ualpha = 0, Ubeta = 0;
```

```
volatile uint32_t encoder = 0;
volatile float angulo = 0.0;
float32_t sinevalue = 0, cosinevalue = 0;
int currentStep = 0;
volatile float theta;

float32_t Uq = 0, Ud = 0;
/*Variables para el monitor*/
float32_t ClarkeAlpha = 0, ClarkeBeta = 0;
float32_t resultIq = 0, resultId = 0;



float32_t error_sumD = 0;
float32_t error_sumQ = 0;

float32_t Ia = 0, Ib = 0;
volatile uint32_t FirstLoop = 0;

tSVPWM sSVPWM = SVPWM_DEFAULTS;

float32_t ParkInverseTransformationD, ParkInverseTransformationQ,
ClarkeInverseTransformationA, ClarkeInverseTransformationB;

struct ParkTransformations {
        float32_t Id;
        float32_t Iq;
        float32_t sine;
        float32_t cos;

};

struct ParkInverseTransformation {
        float32_t IaInv;
        float32_t IbInv;
        float32_t IcInv;

};


typedef struct ParkTransformations Park;

typedef struct ParkInverseTransformation ParkInv;

Park transformation(float32_t Ia, float32_t Ib, float32_t angulo);
ParkInv invtransformation(float32_t Ud, float32_t Uq, float32_t Usine,
float32_t Ucos);

/*Ondas senoidales*/
/*Onda senoidal desfasada 0°*/
uint32_t pwmSin1[] = {128, 111,95, 80,65, 51,39, 28,18, 11,5, 2,0, 1,3,
8, 14,23, 33,45, 58,72, 87,103, 119,136, 152,168, 183,197,
210, 222,232, 241,247, 252,254, 255,253, 250,244, 237,227, 216,204,
190, 175,160, 144,128};


/*Onda senoidal desfasada 120°*/
```

```
uint32_t pwmSin2[] = {238, 245,250, 254,255, 254,251, 246,239, 230,220,
208,195, 180,165,
149, 133,117, 100,85, 70,56, 43,31, 21,13, 7,3, 0,0,
2, 6,12, 20,30, 41,53, 67,82, 98,114, 130,147, 162,178,
192, 206,218, 229,238};

/*Onda senoidal desfasada 240°*/
uint32_t pwmSin3[] = {17, 26,37, 49,63, 77,93, 108,125, 141,157, 173,188,
202,214,
225, 235,243, 249,253, 255,255, 252,248, 242,234, 224,212, 199,185,
170, 155,138, 122,106, 90,75, 60,47, 35,25, 16,9, 4,1,
0, 1,5, 10,17};


/*Interrupción para resetear el Encoder*/
void EXTI4_IRQHandler(void)
{
        if(EXTI->PR & EXTI_PR_PR4)
                {

                        FirstLoop = 1;
                        TIM1->CNT = 0;
                        EXTI->PR |= EXTI_PR_PR4;
                }
}

/*PI Control del eje D*/
float32_t PI_controlD(float32_t refD, float32_t realD){

        float32_t out, pi_error;

        pi_error = refD - realD;
        error_sumD += pi_error;
        /*Limitar el error*/
        if(error_sumD < ERRORMIND){
                error_sumD = ERRORMIND;
                }
        else if(error_sumD > ERRORMAXD){
                error_sumD = ERRORMAXD;
                }

                out = KpD*(pi_error + (KiD*error_sumD));

                /*Limitar el output*/
        if(out < OUTMIND){
                out = OUTMIND;
                }
        else if(out > OUTMAXD){
                out = OUTMAXD;
                }
        return(out);
}

/*PI Control del eje Q*/
float32_t PI_controlQ(float32_t refQ, float32_t realQ){

        float32_t out, pi_error;
```

```
        pi_error = refQ - realQ;
        error_sumQ += pi_error;
        /*Limitar el error*/
        if(error_sumQ < ERRORMINQ){
                error_sumQ = ERRORMINQ;
                }
        else if(error_sumQ > ERRORMAXQ){
                error_sumQ = ERRORMAXQ;
                }

                out = KpQ*(pi_error + (KiQ*error_sumQ));

                /*Limitar el output*/
        if(out < OUTMINQ){
                out = OUTMINQ;
                }
        else if(out > OUTMAXQ){
                out = OUTMAXQ;
                }

        return(out);

}

/*Transformada de Clarke y Park*/
Park transformation(float32_t Ia, float32_t Ib, float32_t angulo){

        float32_t Ialpha, Ibeta;
        float32_t Id, Iq;

        Park p;

        arm_clarke_f32(Ia, Ib, &Ialpha, &Ibeta);

        ClarkeAlpha = Ialpha;
        ClarkeBeta = Ibeta;

        arm_sin_cos_f32(angulo,&sinevalue, &cosinevalue);

        arm_park_f32(Ialpha, Ibeta, &Id, &Iq, sinevalue, cosinevalue);

        p.Id = Id;
        p.Iq = Iq;
        p.sine = sinevalue;
        p.cos = cosinevalue;

        return(p);
}

/*Lectura de los valores del ADC*/
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
          Park result;

                HAL_ADC_Start(&hadc1);
                HAL_ADC_Start(&hadc2);
```

```
                if(HAL_ADC_PollForConversion(&hadc1, 1) ==  HAL_OK){
                                Ia = HAL_ADC_GetValue(&hadc1);
                        }

                if(HAL_ADC_PollForConversion(&hadc2, 1) ==  HAL_OK){
                        Ib = HAL_ADC_GetValue(&hadc2);
                  }

                HAL_ADC_Stop(&hadc1);
                HAL_ADC_Stop(&hadc2);

                Ia = (Ia - 152);        //Eliminar offset
                Ib = (Ib - 152);

                encoder = TIM1->CNT;
                                                        //Lectura del encoder
                angulo = (float)encoder/2.0f;
                                //Transformacion a angulo mecanico
                theta = fmod(angulo*11.0f, 360.0f);
                //Transformacion a angulo electrico

                result =        transformation(Ia, Ib, theta);
        //Transformada de Clarke y Park
                resultId = result.Id;
                resultIq = result.Iq;

                Ud = PI_controlD(0.0f, result.Id);
                //Controlador PI para el eje D
                Uq = PI_controlQ(100.0f, result.Iq);        //Controlador
PI para el eje Q

                /*With SVM*/
                arm_inv_park_f32(Ud, Uq, &Ualpha, &Ubeta, result.sine,
result.cos); //Transformada inversa de Park

                sSVPWM.fUdcCCRval = Uq;
                                                        //Situar el valor Uq
como el valor máximo del SVM
                sSVPWM.fUal = Ualpha;

                sSVPWM.fUbe = Ubeta;

                sSVPWM.m_calc(&sSVPWM);
                                                        //Enviar los valores
al módulo svpwm.c


                if (FirstLoop == 1){
                        //Igual los valores calculados en el modulo svpwm.c
a los pines del timer 3
                        TIM3->CCR1 = sSVPWM.fCCRB;
                        TIM3->CCR2 = sSVPWM.fCCRC;
                        TIM3->CCR3 = sSVPWM.fCCRA;
                }

}
```

```c
int main(void)
{

        sSVPWM.enInType = AlBe;     // set the input type
        sSVPWM.fUdc = 3.0f;                         // set the DC-Link voltage
in Volts

  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_ADC1_Init();
  MX_ADC2_Init();
  MX_TIM2_Init();
  MX_TIM3_Init();
  MX_TIM4_Init();

        /* Initialize PWMs */
        HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);
        HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2);
        HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_3);

        /*Initialize Timer 4 Interrupt*/
        HAL_TIM_Base_Start_IT(&htim4);

        /* Frequency Encoder */
        HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);

        /* Encoder */
        init_hardware_encoder();



  while (1)
  {

                if(FirstLoop == 0){
                        TIM3->CCR1 = pwmSin1[currentStep]*0.15;

                        TIM3->CCR2 = pwmSin2[currentStep]*0.15;
                        TIM3->CCR3 = pwmSin3[currentStep]*0.15;

                        currentStep += 1;

                        if(currentStep >= 50){
                                currentStep = 0;
                        }
                        WaitForAMoment(20000);
                }
  }
```

```
}

void WaitForAMoment(int Moment)
{
        volatile int i, j=0;

        for(i = 0; i < Moment; i++)
        {
                j++;
        }
}


static void init_hardware_encoder(void)
{
                        //Encoder A and B
                        RCC->AHB1ENR |= RCC_AHB1ENR_GPIOEEN;

                        //Habilita el puerto E
                        RCC->APB2ENR |= RCC_APB2ENR_TIM1EN;

                                //Habilita el TIM1

                        //PA11 and PA9
                        GPIOE->MODER |= (GPIO_MODER_MODE9_1 |
GPIO_MODER_MODE11_1);     //Modo de función alternativo
                        GPIOE->AFR[1] = 0x1010;


        //Habilita TIM1_CH1 y TIM1_CH2

                        TIM1->ARR = 0xFFFF;


        //Valor maximo permitido por el timer

                        TIM1->CCMR1 |= (TIM_CCMR1_CC1S_0 |
TIM_CCMR1_CC2S_0);                                          //Canal 1 y 2
como entradas a los inputs 1 y 2 del timer
                        TIM1->SMCR |= (TIM_SMCR_SMS_1);

                                        //Encoder modo 2




                        TIM1->CR1 |= TIM_CR1_CEN;

                                                                //El contador
de TIM1 habilitado

              //Interrupcion del canal Z en el pin PC4
                        RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN;

                        //Habilita el puerto C
```

```
                        GPIOC->MODER &= ~(GPIO_MODER_MODE4);

                        //Pin 4 como input
                        GPIOC->PUPDR &= ~(GPIO_PUPDR_PUPD4);

                        //Sin resistencias de pull-up o pull-down

                        SYSCFG->EXTICR[1] = SYSCFG_EXTICR2_EXTI4_PC;

        //Selección de la fuente de input para el EXTIx interrupción
externa
                        EXTI->IMR |= EXTI_IMR_MR4;

                                                    //Habilita la
interrupción
                        EXTI->FTSR |= EXTI_FTSR_TR4;

                                                //Modo flanco de bajada

                        NVIC_EnableIRQ(EXTI4_IRQn);

                                                    //Habilita la
interrupcion en NVIC (Nested Vectored Interrupt Controller)
                        NVIC_SetPriority(EXTI4_IRQn, 0);

                                                //Selecciona un nivel de prioridad
alto
}
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 168;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 7;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
```

```
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief ADC1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC1_Init(void)
{

  /* USER CODE BEGIN ADC1_Init 0 */

  /* USER CODE END ADC1_Init 0 */

  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC1_Init 1 */

  /* USER CODE END ADC1_Init 1 */
  /** Configure the global features of the ADC (Clock, Resolution, Data
Alignment and number of conversion)
  */
  hadc1.Instance = ADC1;
  hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
  hadc1.Init.Resolution = ADC_RESOLUTION_8B;
  hadc1.Init.ScanConvMode = DISABLE;
  hadc1.Init.ContinuousConvMode = DISABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 1;
  hadc1.Init.DMAContinuousRequests = DISABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure for the selected ADC regular channel its corresponding rank
in the sequencer and its sample time.
  */
  sConfig.Channel = ADC_CHANNEL_2;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
```

```
  /* USER CODE BEGIN ADC1_Init 2 */

  /* USER CODE END ADC1_Init 2 */

}

/**
  * @brief ADC2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC2_Init(void)
{

  /* USER CODE BEGIN ADC2_Init 0 */

  /* USER CODE END ADC2_Init 0 */

  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC2_Init 1 */

  /* USER CODE END ADC2_Init 1 */
  /** Configure the global features of the ADC (Clock, Resolution, Data
Alignment and number of conversion)
  */
  hadc2.Instance = ADC2;
  hadc2.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
  hadc2.Init.Resolution = ADC_RESOLUTION_8B;
  hadc2.Init.ScanConvMode = DISABLE;
  hadc2.Init.ContinuousConvMode = DISABLE;
  hadc2.Init.DiscontinuousConvMode = DISABLE;
  hadc2.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc2.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc2.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc2.Init.NbrOfConversion = 1;
  hadc2.Init.DMAContinuousRequests = DISABLE;
  hadc2.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  if (HAL_ADC_Init(&hadc2) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure for the selected ADC regular channel its corresponding rank
in the sequencer and its sample time.
  */
  sConfig.Channel = ADC_CHANNEL_3;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc2, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN ADC2_Init 2 */

  /* USER CODE END ADC2_Init 2 */

}
```

```c
/**
  * @brief TIM2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM2_Init(void)
{

  /* USER CODE BEGIN TIM2_Init 0 */

  /* USER CODE END TIM2_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_IC_InitTypeDef sConfigIC = {0};

  /* USER CODE BEGIN TIM2_Init 1 */

  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 0;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 0xffffffff;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_IC_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) !=
HAL_OK)
  {
    Error_Handler();
  }
  sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_FALLING;
  sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
  sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
  sConfigIC.ICFilter = 0;
  if (HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC, TIM_CHANNEL_1) !=
HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */
```

```
  /* USER CODE END TIM2_Init 2 */

}

/**
  * @brief TIM3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM3_Init(void)
{

  /* USER CODE BEGIN TIM3_Init 0 */

  /* USER CODE END TIM3_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM3_Init 1 */

  /* USER CODE END TIM3_Init 1 */
  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 10; //20
  htim3.Init.CounterMode = TIM_COUNTERMODE_CENTERALIGNED1;
  htim3.Init.Period = 255;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
  if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim3) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) !=
HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_1) !=
HAL_OK)
  {
    Error_Handler();
```

```
  }
  if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_2) !=
HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_3) !=
HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */
  HAL_TIM_MspPostInit(&htim3);

}

/**
  * @brief TIM4 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM4_Init(void)
{

  /* USER CODE BEGIN TIM4_Init 0 */

  /* USER CODE END TIM4_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM4_Init 1 */

  /* USER CODE END TIM4_Init 1 */
  htim4.Instance = TIM4;
  htim4.Init.Prescaler = 0; //0
  htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim4.Init.Period = 53799; //8399 52499 5375
  htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) !=
HAL_OK)
  {
    Error_Handler();
```

```
  }
  /* USER CODE BEGIN TIM4_Init 2 */

  /* USER CODE END TIM4_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOE_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(CS_I2C_SPI_GPIO_Port, CS_I2C_SPI_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(OTG_FS_PowerSwitchOn_GPIO_Port,
OTG_FS_PowerSwitchOn_Pin, GPIO_PIN_SET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD, LD4_Pin|LD3_Pin|LD5_Pin|LD6_Pin
                          |Audio_RST_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : CS_I2C_SPI_Pin */
  GPIO_InitStruct.Pin = CS_I2C_SPI_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(CS_I2C_SPI_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : OTG_FS_PowerSwitchOn_Pin */
  GPIO_InitStruct.Pin = OTG_FS_PowerSwitchOn_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(OTG_FS_PowerSwitchOn_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : PDM_OUT_Pin */
  GPIO_InitStruct.Pin = PDM_OUT_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
  HAL_GPIO_Init(PDM_OUT_GPIO_Port, &GPIO_InitStruct);
```

```
/*Configure GPIO pin : B1_Pin */
GPIO_InitStruct.Pin = B1_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : I2S3_WS_Pin */
GPIO_InitStruct.Pin = I2S3_WS_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.Alternate = GPIO_AF6_SPI3;
HAL_GPIO_Init(I2S3_WS_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pins : SPI1_SCK_Pin SPI1_MISO_Pin SPI1_MOSI_Pin */
GPIO_InitStruct.Pin = SPI1_SCK_Pin|SPI1_MISO_Pin|SPI1_MOSI_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.Alternate = GPIO_AF5_SPI1;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/*Configure GPIO pin : BOOT1_Pin */
GPIO_InitStruct.Pin = BOOT1_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(BOOT1_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : CLK_IN_Pin */
GPIO_InitStruct.Pin = CLK_IN_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
HAL_GPIO_Init(CLK_IN_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pins : LD4_Pin LD3_Pin LD5_Pin LD6_Pin
                        Audio_RST_Pin */
GPIO_InitStruct.Pin = LD4_Pin|LD3_Pin|LD5_Pin|LD6_Pin
                      |Audio_RST_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

/*Configure GPIO pins : I2S3_MCK_Pin I2S3_SCK_Pin I2S3_SD_Pin */
GPIO_InitStruct.Pin = I2S3_MCK_Pin|I2S3_SCK_Pin|I2S3_SD_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.Alternate = GPIO_AF6_SPI3;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

/*Configure GPIO pin : VBUS_FS_Pin */
GPIO_InitStruct.Pin = VBUS_FS_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
```

```
  HAL_GPIO_Init(VBUS_FS_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : OTG_FS_ID_Pin OTG_FS_DM_Pin OTG_FS_DP_Pin */
  GPIO_InitStruct.Pin = OTG_FS_ID_Pin|OTG_FS_DM_Pin|OTG_FS_DP_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF10_OTG_FS;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : OTG_FS_OverCurrent_Pin */
  GPIO_InitStruct.Pin = OTG_FS_OverCurrent_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(OTG_FS_OverCurrent_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : Audio_SCL_Pin Audio_SDA_Pin */
  GPIO_InitStruct.Pin = Audio_SCL_Pin|Audio_SDA_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  GPIO_InitStruct.Alternate = GPIO_AF4_I2C1;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : MEMS_INT2_Pin */
  GPIO_InitStruct.Pin = MEMS_INT2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(MEMS_INT2_GPIO_Port, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
state */
  __disable_irq();

  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
```

```
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line
number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

svpwm.c

```
/* Includes ------------------------------------------------------------
--------*/
#include "svpwm.h"

/* Typedef privado -----------------------------------------------------
--------*/
/* Definiciones privadas -----------------------------------------------
---------------*/
#define M_Pi    3.1415926535897f // Valor de la constante Pi
#define M_Pi_3  (M_Pi/3.0f)

/* Constantes privadas -------------------------------------------------
----------*/

/**
  * @brief Matriz con celdas que indican en una tabla de intervalors de
tiempo para
        * la tecnica de calculo del ciclo de trabajo del SVPWM
  */
const uint8_t au8PermuataionMatrix[6][3] =
{
        { 1, 2, 0 },
        { 3, 1, 0 },
        { 0, 1, 2 },
        { 0, 3, 1 },
        { 2, 0, 1 },
        { 1, 0, 3 }
};

/* Macro privado -------------------------------------------------------
--------*/
/* Variables privadas --------------------------------------------------
---------*/
/* Funciones privadas --------------------------------------------------
*/

/**
  * @brief  Funcion de calculo del SVPWM. Esta funcion calcula tres valores
de
        *         ciclo de trabajo para generar un vector de voltaje de
tres fases
        *                                        con amplitud - ptSVPWM->fUs y
angulo - ptSVPWM->fAngRad.
  * @param  ptSVPWM: puntero para la estrucura de datos del usuario de tipo
"tSVPWM".
  * @retval None
  */
void tSVPWM_calc(tSVPWM* ptSVPWM)
{
        uint8_t u8Sector;
        float   fMaxUs, fScaledUs, fBeta, fTb1, fTb2, afTi[4];

        fMaxUs = ptSVPWM->fUdc * (1.0f/sqrtf(3.0f));
```

```
        switch(ptSVPWM->enInType)
        {
                case AlBe:
                        ptSVPWM->fUs = hypotf(ptSVPWM->fUbe, ptSVPWM->fUal);
                        ptSVPWM->fAngRad = atan2f(ptSVPWM->fUbe, ptSVPWM-
>fUal);
                break;

                case UsAng:
                        ptSVPWM->fUs = fabsf(ptSVPWM->fUs);
                break;

                default: return;
        }

        if(ptSVPWM->fUs > fMaxUs) ptSVPWM->fUs = fMaxUs;

        fScaledUs = ptSVPWM->fUs/fMaxUs;

        ptSVPWM->fAngRad += M_Pi;

        u8Sector = (uint8_t)(ptSVPWM->fAngRad * (1.0f/M_Pi_3));

        fBeta = ptSVPWM->fAngRad - M_Pi_3 * u8Sector;

        fTb1 = fScaledUs * sinf(M_Pi_3 - fBeta);
        fTb2 = fScaledUs * sinf(fBeta);

        afTi[0] = (1.0f - fTb1 - fTb2)*0.5f;
        afTi[1] = fTb1 + fTb2 + afTi[0];
        afTi[2] = fTb2 + afTi[0];
        afTi[3] = fTb1 + afTi[0];

        ptSVPWM->fCCRA = ptSVPWM->fUdcCCRval *
afTi[au8PermuataionMatrix[u8Sector][0]];
        ptSVPWM->fCCRB = ptSVPWM->fUdcCCRval *
afTi[au8PermuataionMatrix[u8Sector][1]];
        ptSVPWM->fCCRC = ptSVPWM->fUdcCCRval *
afTi[au8PermuataionMatrix[u8Sector][2]];
}
```

svpwm.h

```
/* Para evitar la inclusión recursiva ---------------------------------
-----*/
#ifndef __SVPWM_H__
#define __SVPWM_H__

#ifdef __cplusplus
extern "C" {
#endif

/* Includes -----------------------------------------------------------
--------*/
#include <stdint.h>
#include <math.h>

/* Tipos exportados ---------------------------------------------------
----------*/

/**
  * @brief SVPWM modulo
  */
typedef enum
{
        AlBe    = 0x00U,                // Componente de entrada Alpha-Beta
        UsAng   = 0x01U                 // Componenten de entrada Magnitud-
Angulo
} tInType;

/**
  * @brief "SVPWM Modulo" estructura de datos
  */
typedef struct sSVPWM
{
//Entradas:
        tInType enInType;               // Tipo de dato de entrada SVPWM
modulo
        float   fUdc;                                   // Voltaje DC de
entrada, Voltios
        float   fUdcCCRval;             // Valor del Counter compare register
que equivale a

                                                        // a la tensión
completa de la fuente de alimentacion DC
        // enInType == AlBe:
        float   fUal;                   // Entrada Alpha, Voltios
        float   fUbe;                   // Entrada Beta, Voltios
        // enInType == UsAng:
        float   fUs;                        // Magnitud de entrada, Voltios
(enInType == UsAng)
        float   fAngRad;                  // Angulo de entrada, Radianes
(enInType == UsAng)
//Salidas:
```

```
        float   fCCRA;                  // Valor del Counter compare register
A
        float   fCCRB;                  // Valor del Counter compare register
B
        float   fCCRC;                  // Valor del Counter compare register
C
// Funciones:
        void (*m_calc)(struct sSVPWM*);       // Puntero a la funcion que
calcula el SVPWM
} tSVPWM;

/* Constantes exportadas ---------------------------------------------------
-----------*/

/**
  * @brief Initialization constant with defaults for user variables
  *        with "tSVPWM" type
  */
#define SVPWM_DEFAULTS {                \
        .enInType       = AlBe,         \
        .fUal           = 0.0f,         \
        .fUbe           = 0.0f,         \
        .fUs            = 0.0f,         \
        .fAngRad        = 0.0f,         \
        .fUdc           = 0.0f,         \
        .fUdcCCRval     = 0.0f,         \
        .fCCRA          = 0.0f,         \
        .fCCRB          = 0.0f,         \
        .fCCRC          = 0.0f,         \
        .m_calc         = tSVPWM_calc   \
}

/* Macro exportado ---------------------------------------------------------
---------*/
/* Funciones exportadas ----------------------------------------------------
----------*/

/* Funcion que calcula el ciclo de trabajo
*******************************/
void tSVPWM_calc(tSVPWM*);

#ifdef __cplusplus
}
#endif

#endif /* __SVPWM_H__ */
```

life.augmented

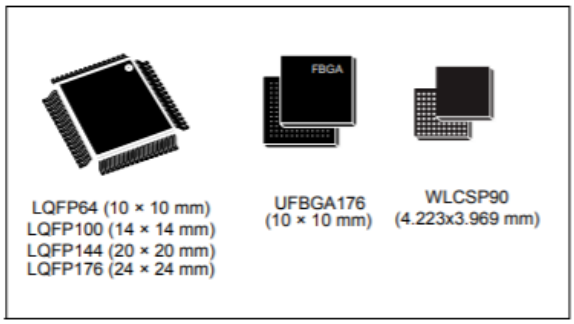
STM32F405xx STM32F407xx

Arm® Cortex®-M4 32b MCU+FPU, 210DMIPS, up to 1MB Flash/192+4KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 15 comm. interfaces & camera

Datasheet - production data

Features

- Core: Arm® 32-bit Cortex®-M4 CPU with FPU, Adaptive real-time accelerator (ART Accelerator) allowing 0-wait state execution from Flash memory, frequency up to 168 MHz, memory protection unit, 210 DMIPS/ 1.25 DMIPS/MHz (Dhrystone 2.1), and DSP instructions
- Memories
  - Up to 1 Mbyte of Flash memory
  - Up to 192+4 Kbytes of SRAM including 64-Kbyte of CCM (core coupled memory) data RAM
  - 512 bytes of OTP memory
  - Flexible static memory controller supporting Compact Flash, SRAM, PSRAM, NOR and NAND memories
- LCD parallel interface, 8080/6800 modes
- Clock, reset and supply management
  - 1.8 V to 3.6 V application supply and I/Os
  - POR, PDR, PVD and BOR
  - 4-to-26 MHz crystal oscillator
  - Internal 16 MHz factory-trimmed RC (1% accuracy)
  - 32 kHz oscillator for RTC with calibration
  - Internal 32 kHz RC with calibration
- Low-power operation
  - Sleep, Stop and Standby modes
  - V_{BAT} supply for RTC, 20×32 bit backup registers + optional 4 KB backup SRAM
- 3×12-bit, 2.4 MSPS A/D converters: up to 24 channels and 7.2 MSPS in triple interleaved mode
- 2×12-bit D/A converters
- General-purpose DMA: 16-stream DMA controller with FIFOs and burst support



LQFP64 (10 × 10 mm)
LQFP100 (14 × 14 mm)
LQFP144 (20 × 20 mm)
LQFP176 (24 × 24 mm)

UFBGA176
(10 × 10 mm)

WLCSP90
(4.223x3.969 mm)

- Up to 17 timers: up to twelve 16-bit and two 32-bit timers up to 168 MHz, each with up to 4 IC/OC/PWM or pulse counter and quadrature (incremental) encoder input
- Debug mode
  - Serial wire debug (SWD) & JTAG interfaces
  - Cortex-M4 Embedded Trace Macrocell™
- Up to 140 I/O ports with interrupt capability
  - Up to 136 fast I/Os up to 84 MHz
  - Up to 138 5 V-tolerant I/Os
- Up to 15 communication interfaces
  - Up to 3 × I^2C interfaces (SMBus/PMBus)
  - Up to 4 USARTs/2 UARTs (10.5 Mbit/s, ISO 7816 interface, LIN, IrDA, modem control)
  - Up to 3 SPIs (42 Mbits/s), 2 with muxed full-duplex I^2S to achieve audio class accuracy via internal audio PLL or external clock
  - 2 × CAN interfaces (2.0B Active)
  - SDIO interface
- Advanced connectivity
  - USB 2.0 full-speed device/host/OTG controller with on-chip PHY
  - USB 2.0 high-speed/full-speed device/host/OTG controller with dedicated DMA, on-chip full-speed PHY and ULPI
  - 10/100 Ethernet MAC with dedicated DMA: supports IEEE 1588v2 hardware, MII/RMII

Anexo 5: STM32F4Discovery

life.augmented

UM1472
User manual

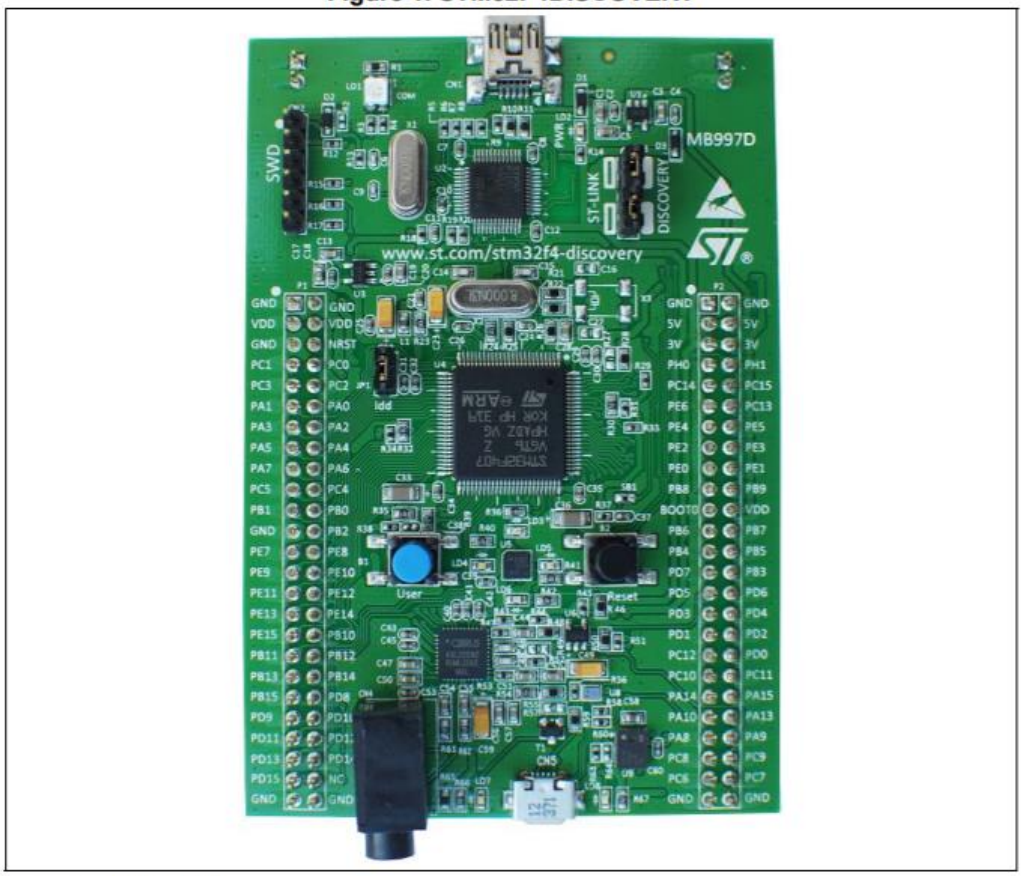
Discovery kit with STM32F407VG MCU

Introduction

The STM32F4DISCOVERY Discovery kit allows users to easily develop applications with the STM32F407VG high-performance microcontroller with the Arm® Cortex®-M4 32-bit core. It includes everything required either for beginners or experienced users to get started quickly.

Based on STM32F407VG, it includes an ST-LINK/V2-A embedded debug tool, one ST-MEMS digital accelerometer, one digital microphone, one audio DAC with integrated class D speaker driver, LEDs, push-buttons and a USB OTG Micro-AB connector. Specialized add-on boards can be connected by means of the extension header connectors. The STM32F4DISCOVERY Discovery kit comes with the STM32 comprehensive free software libraries and examples available with the STM32CubeF4 MCU Package.

Figure 1. STM32F4DISCOVERY



Picture is not contractual.

TEXAS INSTRUMENTS

DRV8302

SLES267C – AUGUST 2011 – REVISED MARCH 2016

DRV8302 Three Phase Gate Driver With Dual Current Shunt Amplifiers and Buck Regulator – Hardware Controlled

1 Features

- 8-V to 60-V Operating Supply Voltage Range
- 1.7-A Source and 2.3-A Sink Gate Drive Current Capability
- Bootstrap Gate Driver With 100% Duty Cycle Support
- 6 or 3 PWM Input Modes
- Dual Integrated Current Shunt Amplifiers With Adjustable Gain and Offset
- 3.3-V and 5-V Interface Support
- Hardware Control Interface
- Protection Features:
  - Programmable Dead Time Control (DTC)
  - Programmable Overcurrent Protection (OCP)
  - PVDD and GVDD Undervoltage Lockout (UVLO)
  - GVDD Overvoltage Lockout (OVLO)
  - Overtemperature Warning/Shutdown (OTW/OTS)
  - Reported through nFAULT and nOCTW pins

2 Applications

- 3-Phase BLDC and PMSM Motors
- CPAP and Pumps
- E-Bikes
- Power Tools
- Robotics and RC Toys
- Industrial Automation

3 Description

The DRV8302 is a gate driver IC for three phase motor drive applications. It provides three half bridge drivers, each capable of driving two N-channel MOSFETs. The device supports up to 1.7-A source and 2.3-A peak current capability. The DRV8302 can operate off of a single power supply with a wide range from 8-V to 60-V. It uses a bootstrap gate driver architecture with trickle charge circuitry to support 100% duty cycle. The DRV8302 uses automatic hand shaking when the high side or low side MOSFET is switching to prevent current shoot through. Integrated VDS sensing of the high and low side MOSFETs is used to protect the external power stage against overcurrent conditions.

The DRV8303 includes two current shunt amplifiers for accurate current measurement. The amplifiers support bi-directional current sensing and provide and adjustable output offset up to 3 V.

The DRV8302 also includes an integrated switching mode buck converter with adjustable output and switching frequency. The buck converter can provide up to 1.5 A to support MCU or additional system power needs.

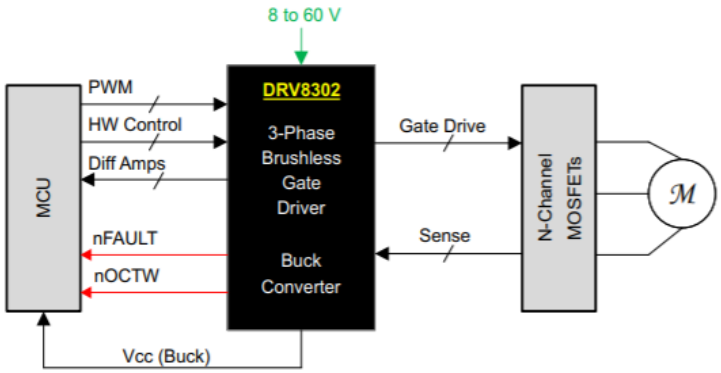
A hardware interface allows for configuring various device parameters including dead time, overcurrent, PWM mode, and amplifier settings. Error conditions are reported through the nFAULT and nOCTW pins.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
DRV8302	HTSSOP (56)	14.00 mm × 6.10 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

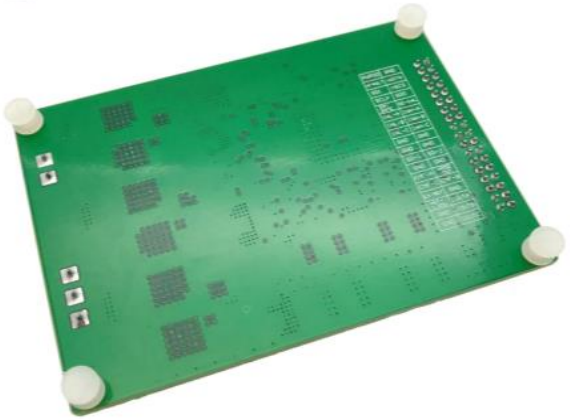
Simplified Schematic



Anexo 7: Módulo de controlador de motor DRV8302

ZONRI

Manual del módulo de controlador de motor DRV8302



Visión general:

▶ Rango de entrada de potencia nominal:5,5 V - 45 V CC, Corriente de impulsión nominal de salida 15A(No hay radiador instalado a temperatura ambiente)

▶ Circuito amplificador de corriente externo trifásico a bordo, la ganancia es 12.22, La corriente de amplitud total es ±27A.

▶ A bordo de tres circuitos de muestreo de fuerza electromotriz inversa y circuito de muestreo de potencia, la relación de atenuación de la señal es 5.1 / 73.1.

▶ Nivel de referencia del amplificador de corriente diferencial trifásico opcional, que se puede comparar con 5V U otro nivel de conexión del sistema.

▶ Mantener y tener DRV8302 Salida de amplificador dual interno.

▶ Soporte para tres amplificadores diferenciales de corriente externos ST FOC Aplicación de biblioteca.

▶ apoyo BLDC Desarrollo de aplicaciones de accionamiento de motor sin escobillas no inductivo.

▶ Cuatro pisos PCB FR4 Junta, en línea con ROHS.

contenido:

https://zonritech.taobao.com

Incremental 40-mm-dia. Rotary Encoder

E6B2-C

CSM_E6B2-C_DS_E_6_1

General-purpose Encoder with External Diameter of 40 mm

- Incremental model
- External diameter of 40 mm.
- Resolution of up to 2,000 ppr.

Be sure to read *Safety Precautions* on page 4.

For the most recent information on models that have been certified for safety standards, refer to your OMRON website.

Ordering Information

Encoders [Refer to *Dimensions* on page 4.]

Power supply voltage	Output configuration	Resolution (pulses/rotation)	Model
5 to 24 VDC	NPN open-collector output	10, 20, 30, 40, 50, 60, 100, 200, 300, 360, 400, 500, 600	**E6B2-CWZ6C (resolution) 0.5M** Example: E6B2-CWZ6C 10P/R 0.5M
		720, 800, 1,000, 1,024	
		1,200, 1,500, 1,800, 2,000	
12 to 24 VDC	PNP open-collector output	100, 200, 360, 500, 600	**E6B2-CWZ5B (resolution) 0.5M** Example: E6B2-CWZ5B 100P/R 0.5M
		1,000	
		2,000	
5 to 12 VDC	Voltage output	10, 20, 30, 40, 50, 60, 100, 200, 300, 360, 400, 500, 600	**E6B2-CWZ3E (resolution) 0.5M** Example: E6B2-CWZ3E 10P/R 0.5M
		1,000	
		1,200, 1,500, 1,800, 2,000	
5 VDC	Line-driver output	10, 20, 30, 40, 50, 60, 100, 200, 300, 360, 400, 500, 600	**E6B2-CWZ1X (resolution) 0.5M** Example: E6B2-CWZ1X 10P/R 0.5M
		1,000, 1,024	
		1,200, 1,500, 1,800, 2,000	

Accessories (Order Separately) [Refer to *Dimensions* on *Rotary Encoder Accessories*.]

Name	Model	Remarks
Couplings	**E69-C06B**	Provided with the product.
	E69-C68B	Different end diameter
	E69-C610B	Different end diameter
	E69-C06M	Metal construction
Flanges	**E69-FBA**	---
	E69-FBA02	E69-2 Servo Mounting Bracket provided.
Servo Mounting Bracket	**E69-2**	---

Note: 1. Refer to *Rotary Encoders Accessories* on your OMRON website for details.
2. Refer to *Precautions For Correct Use of Rotary Encoders* on your OMRON website when using the Rotary Encoders together with a Coupling.

Pliego de condiciones

Índice de pliego de condiciones

1. Definición y alcance del pliego

En el presente proyecto se realiza la implementación de un sistema de control en bucle cerrado para un motor de corriente continua sin escobillas, cuya aplicación consiste en que el motor se mantenga en el mismo par con independencia de la carga mecánica que se le aplique. A pesar del carácter académico de este proyecto, se han considerado todos los elementos necesarios para poder obtener un sistema de control vectorial funcional que pueda implementarse en el mundo industrial.

El objetivo de este trabajo está orientado a los criterios técnicos que permiten la reproducción, desarrollo o implementación del proyecto "Control Vectorial del par motor de un motor brushless", de acuerdo con las especificaciones recogidas en este pliego.

El ámbito de aplicación de este documento es en la reproducción del estudio y las pruebas que se realizan con el control vectorial. Sin embargo, en caso de querer implementar dicho control en otros ámbitos se tendrá que estudiar cada una de las condiciones particulares que en tales estados se produzcan.

2. Condiciones y normas de carácter general

En este capítulo se presentan las condiciones y normas de carácter general que el proyecto debe de cumplir.

2.1 Instalación

Se debe de comprobar que se cumplen con los requisitos necesarios para la correcta instalación del hardware y software.

Para la correcta implementación del proyecto se requieren conocimientos de electrónica analógica, electrónica de potencia, máquinas eléctricas y programación de microcontroladores.

2.2 Seguridad

La instalación y los ensayos que se realicen mediante el hardware y software descrito en la Memoria, deberá de respetar en todo momento las normas de seguridad y, además, tendrá que seguir las pautas descritas.

Debido a que el proyecto hace uso de fuentes de alimentación, se tendrá que hacer uso de todo elemento de protección que sea necesario, como pueden ser fusibles o diferenciales.

Además, se debe de prestar especial atención a las conexiones eléctricas que se realizan en el proyecto y se deben de revisar antes de encender la fuente de alimentación, ya que una conexión incorrecta puede dañar las placas de control o incluso el motor.

Cualquier daño realizado debido a una incorrecta conexión, manipulación de los elementos del proyecto o utilización de estas en condiciones no controladas recae exclusivamente en el usuario.

2.3 Utilización

La utilización que se le puede dar a este proyecto está sometida a las hojas de características técnicas proporcionadas por los fabricantes. Por lo que, el proyecto no se encontrará en condiciones que no hayan sido validadas y estudiadas para todos los elementos que intervienen en el presente trabajo.

2.4 Mantenimiento

Antes de cada prueba y como norma general, se debe realizar una inspección de forma ocular en busca de defectos en el motor de tipo mecánico o fallos de conexionado eléctrico en las placas de control. Además, es recomendable limpiar después de cada uso para garantizar el correcto funcionamiento.

Cuando vaya a darse un periodo de inactividad se tiene que desconectar la fuente de alimentación y guardar el proyecto, para evitar su deterioro por elementos como la humedad o el polvo al que podría estar sometido.

3. Condiciones particulares

3.1 Microcontrolador

Las características técnicas del microcontrolador STM32F407VGT6 se encuentran ubicadas en el Anexo 4 de este documento.

Las características técnicas de la placa de desarrollo STM32F4Discovery, donde se encuentra implementado el microcontrolador STM32F407VGT6, se pueden encontrar en el Anexo 5.

Esta placa de desarrollo cumple con las siguientes normas de seguridad para equipos eléctricos de medida:

- IEX 61010-1, EN61010-1
- UL 61010-1, CSA 61010-1
- UNE-EN 60745
- UNE-EN 61800

Control de calidad

Se debe de realizar una revisión visual de la placa de desarrollo para comprobar que no posee ningún defecto de fábrica, como por ejemplo que no se hayan formado bolas de soldadura en los pines o que

no exista ningún puente de soldadura, y determinar que la serigrafia de todos los componentes sea la que se indica en el citado manual.

3.2 Motor

Las características del motor model ML4108 utilizado se encuentran descritos en la tabla 6.

Fabricante	GARTT
Motor KV	500KV RPM/V
Resistencia del motor (RM)	0,1088Ω
Corriente de reposo (Io/10V)	0,6A/10V
Corriente máxima en continua	23A
Máxima potencia en continua	530W
Peso	112g
Celda de Lipo	4-6S
Motor Diámetro	46mm
Motor Longitud Del Cuerpo	25,5mm
Número de estator armas	24
Número de polos	22
Longitud del eje	33,4mm
Diámetro del mango	3,175mm
Diámetro entre agujeros de perno	19mm/25mm
Rosca de tornillo	M3x6

Tabla 4. Especificaciones del motor sin escobillas ML4108 utilizado en este proyecto

Control de calidad

Debe de comprobarse el buen estado físico del motor, el cual se puede realizar mediante un giro manual para evaluar la suavidad con la que se produce el giro sin roces.

Se ha de comprobar el movimiento del motor sin carga para comprobar que el motor funciona correctamente en giro libre.

Se puede examinar la forma de onda de la fuerza electromotriz (Back-EMF) con un osciloscopio para su verificación.

3.3 Controlador DRV8302

Las características técnicas del driver DRV8302 se encuentran ubicadas en el Anexo 6 de este documento.

Las características técnicas del controlador de motor donde se encuentra implementado el driver DRV8302, se puede encontrar en el Anexo 7.

Control de calidad

Debe comprobarse el buen estado físico del controlador visualmente, se debe de asegurar que el modelo corresponde con la misma versión de la cual se están comprobando las especificaciones. Ha de observarse que los valores de tensión de los condensadores electrolíticos sean superiores a las tensiones de entrada del circuito.

Asegurarse que la salida del controlador corresponde con la señal de entrada, lo cual se puede comprobar conectándolo a una fuente de alimentación de 12V. Posteriormente se tiene que conectar el pin M_PWM a 3.3V y poner uno de los pines INH-x a 3.3V o a GND y observar que en su salida se obtenga la misma señal, pero intensificada, en caso de que esté conectado a tensión.

Además, se debe de comprobar que los pines encargados de proteger el driver DRV8302 funcionan correctamente, para ello se debe conectar el pin M_OC a 3.3V y el pin OC_ADJ a 3.0V y observar si en caso de conectar una tensión superior a los 3.0V en los pines INH-x o INL-x se apaga el canal donde se ha detectado el exceso de corriente. También se tiene que prestar atención a los pines nOTCTW y nFAULT ya que estos pines se encargan de advertir al usuario en caso de fallo.

3.4 Sensor de posición

Las características técnicas del encoder incremental encargado de determinar la posición del motor se encuentran en el Anexo 8 de este documento.

Control de calidad

Para determinar el correcto funcionamiento del encoder incremental se debe hacer girar el encoder a una velocidad constante y mediante un osciloscopio observar la salida en cada uno de sus canales, principalmente se tiene que observar dos ondas cuadradas desfasadas entre sí en los canales A y B, que dependiendo del sentido de giro uno de los canales precederá al otro. Además, se debe observar el canal Z el cual dará un único pulso por vuelta.

Presupuesto

Índice del presupuesto

1. Descripción

En este apartado se realiza el estudio del presupuesto del proyecto FOC con el objetivo de determinar la viabilidad económica mediante el cálculo del coste que requeriría el proyecto. Para ello, se ha presupuestado la mano de obra y los materiales empleados durante la realización de este trabajo, obteniéndose el cuadro de precios parcial de cada una.

A partir de estos recursos, se ha determinado el presupuesto de ejecución de material, el presupuesto de ejecución por contrata y el presupuesto general del proyecto o también conocido como presupuesto base de licitación.

Para la elaboración de cada uno de estos presupuestos se han considerado las *Recomendaciones en la Elaboración de Presupuestos en Actividades de I+D+I* de la UPV de acuerdo al art.83 de la Ley Orgánica de Universidades, que contempla la contratación de trabajos de carácter científico, técnico o artístico por parte de los profesores universitarios.

2. Coste directo

2.1 Mano de obra directa

La tabla 3 muestra los precios de mano de obra para la realización de este proyecto. En la cantidad de horas empleadas se han considerado aspectos como: conexión de las dos placas empleadas, pruebas con cada parte del código empleado, recopilación y análisis de los resultados, las reuniones de planificación, seguimiento y corrección con el tutor del proyecto.

Descripción	Importe		
	Coste (€/h)	Cantidad (h)	Precio (€)
Graduado en Ingeniería electrónica	15	300	4500,00
Tutor responsable del proyecto	30	65	1950,00
Precio Total de la mano de obra directa			6450,00

Tabla 5. Presupuesto de la mano de obra directa

2.2 Material

Para el material utilizado se ha calculado la amortización según la ecuación 48, considerando un periodo de amortización indicado en la tabla 4.

$$Coste\ amort. = \frac{A}{B} * C * D \qquad (48)$$

donde:

A = número de meses que el equipo se va a usar en el proyecto, después de la fecha de compra.

B= periodo de amortización, que varía según la naturaleza del bien.

C= coste del equipo.

D= porcentaje de uso del equipo en el proyecto.

Descripción	Importe					
	Cantidad (ud)	Uso (meses)	Periodo de amor. (años)	Coste (€)	Uso del equipo (%)	Precio (€)
STM32F407Discovery	1	6	6	19,92	85	16,93
DRV8302	1	6	6	42,69	85	36,29
Encoder incremental	1	6	6	27,58	100	27,58
Motor	1	6	8	24,84	100	18,63
Fuente de alimentación	1	6	12	13,99	70	4,90
Cables eléctricos	21	6	12	7,99	70	2,80
Cable USB	1	6	12	6,99	85	2,97
Acoplamiento flexible	1	6	6	2,51	100	2,51
Tornillos	18	6	12	2,49	100	1,25
Pack de destornilladores	1	1	12	12,49	30	1,87
Placa protoboard	1	6	12	3,95	70	1,38
Resistencias de 330Ω	1	6	10	0,10	100	0,06
Tabla de madera	3	6	12	4,50	100	6,75
Sierra	1	1	6	2,20	10	0,22
Taladro	1	1	12	36,54	10	1,83
MATLAB	1	2	1	69,00	50	207,00
Ordenador portátil	1	6	6	388,95	85	330,61
Precio Total de Materiales						663,57

Tabla 6. Coste de materiales

3. Presupuesto de ejecución de material, por contrata y base de licitación

En primer lugar, se calcula el presupuesto de ejecución material, que es el resultado de sumar el coste directo más un 5% de mano de obra indirecta del coste directo, este coste consiste en el equipo que ha apoyado en el proyecto como puede ser el personal de administración, mantenimiento. limpieza, etc., más un 12,5% como coste indirecto, que procede de la suma del coste directo más la mano de obra indirecta.

1.Presupuesto de la mano de obra ---------- 6450,00€

2.Presupuesto del material ---------- 663,57€

Total parcial ---------- 7113,57€

Mano de obra indirecta (5%) ---------- 355,68€

Costes indirectos (12,5%) ---------- 933,65€

Total del Presupuesto de Ejecución Material ---------- **8402,90€**

El presupuesto de ejecución por contrata resulta de la suma del presupuesto de ejecución material, los gastos generales aplicables al presupuesto (12,1% del presupuesto de ejecución de material) y un beneficio industrial que al igual que los gastos generales procede del presupuesto de ejecución de material.

Presupuesto de Ejecución Material ---------- 8402,90€

Beneficio Industrial (5%) ---------- 420,15€

Total Ejecución por Contrata ---------- **8823,25€**

Por último, para obtener el presupuesto base de licitación se debe de sumar el porcentaje de 21% del IVA sobre el presupuesto anterior.

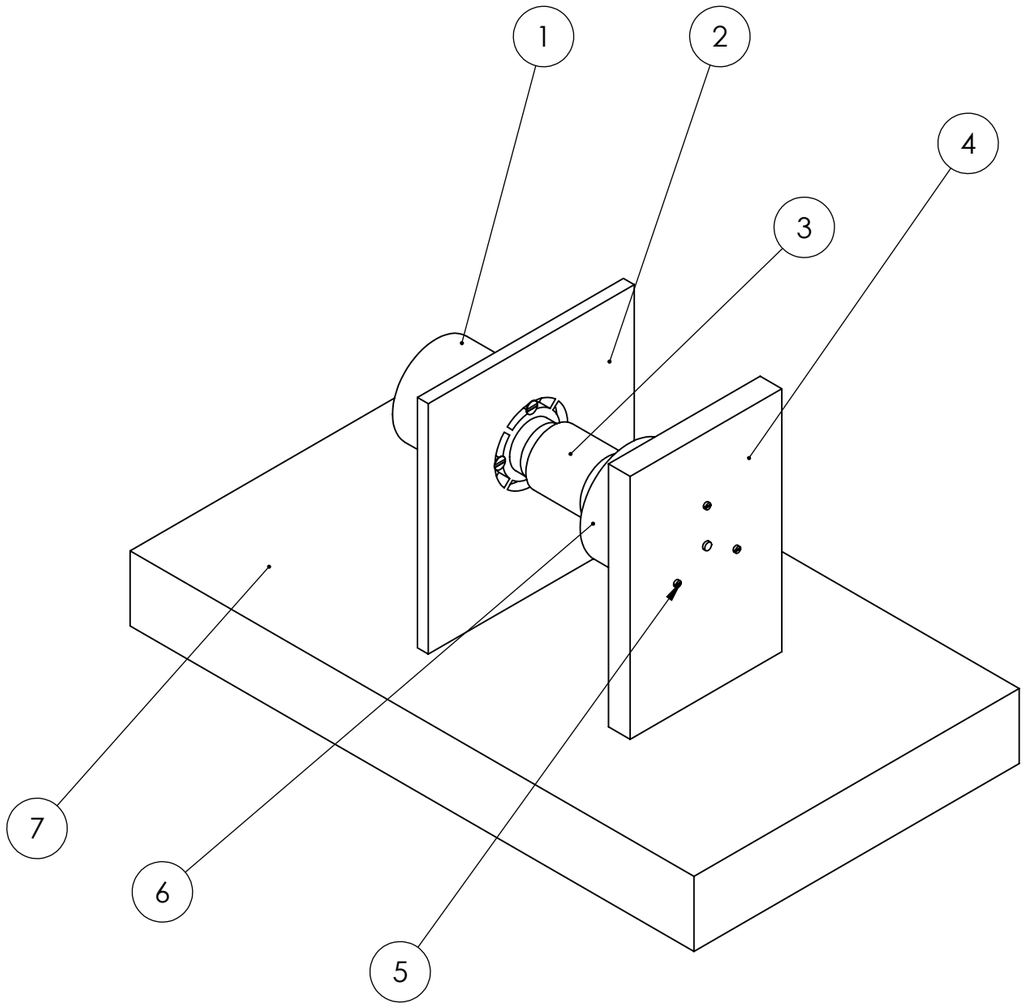
Presupuesto Ejecución por Contrata ---------- 8823,05€

IVA (21%) ---------- 1852,84€

Total Base de Licitación ---------- **10675,89€**

Planos

Índice de planos



N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	CANTIDAD
1	Encoder	Sensor para medir el ángulo del motor	1
2	Tabla de madera	10.5x9.5x0.5cm	1
3	Acoplamiento flexible	Encargado de juntar el enconder con el motor	1
4	Tabla de madera	10.5x7x1cm	1
5	Tornillos	Tornillos de cabeza plana	1
6	Motor	500kV	1
7	Tabla de madera	3x26x15cm	1

TRABAJO FINAL DE GRADO DE INGENIERIA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Proyecto

Control Vectorial de motor BLDC

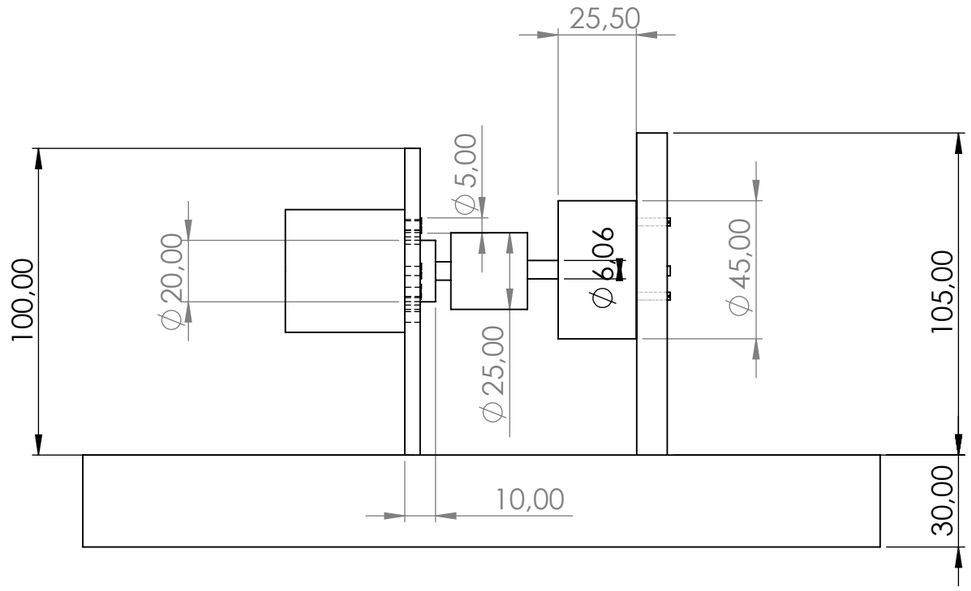
Fecha: Agosto 2021

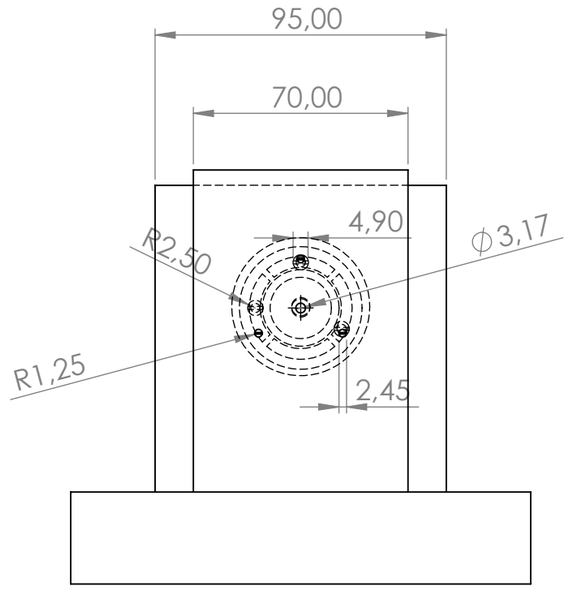
Escala: 1:2

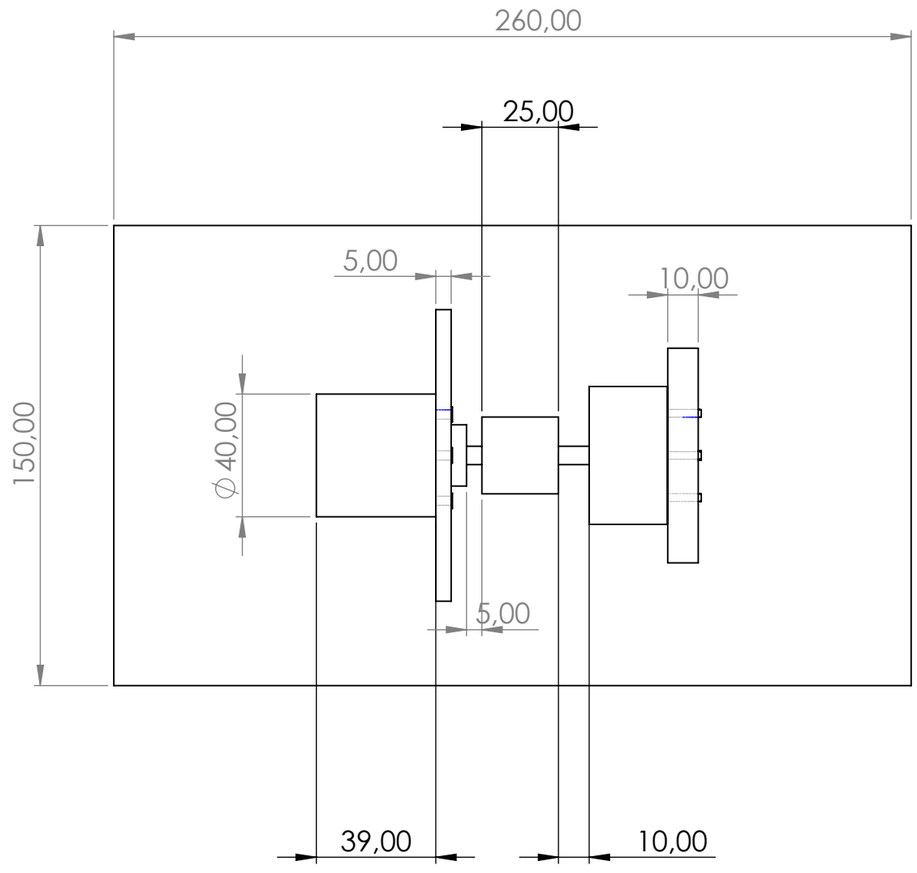
Plano: Vista isometrica

Nº de plano 1

Rafael Matevosyan
Autor del proyecto







TRABAJO FINAL DE GRADO DE INGENIERIA
ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Rafael Matevosyan
Autor del proyecto

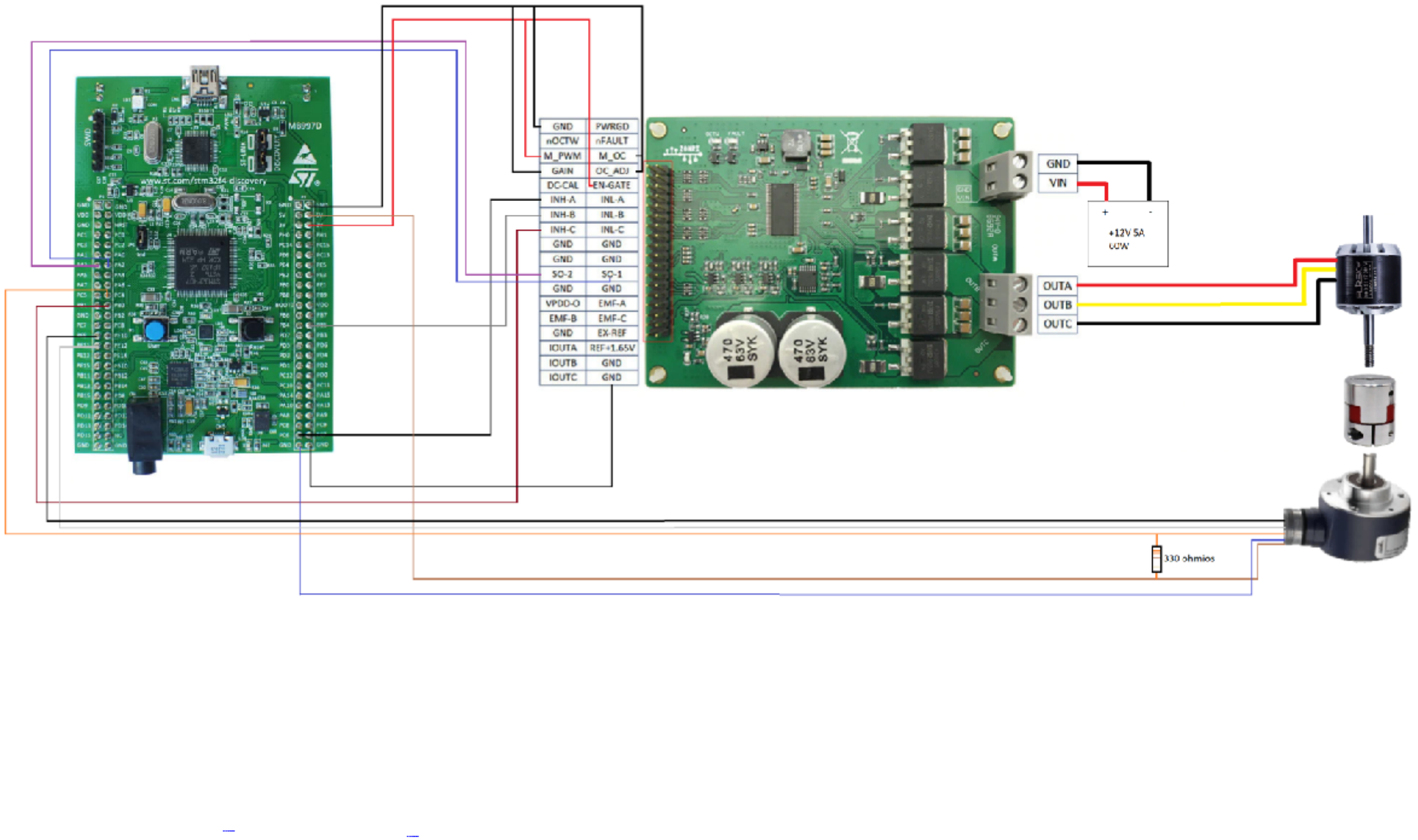
Proyecto
Control Vectorial de motor BLDC

Fecha:
Agosto 2021

Escala:
1:2

Plano:
Estructura del montaje

Nº de plano
2



TRABAJO FINAL DE GRADO DE INGENIERIA
ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Proyecto

Control Vectorial de motor BLDC

Fecha: Agosto 2021

Escala: 1:2

Plano: Conexión de las placas y el motor

Nº de plano: 3

Rafael Matevosyan
Autor del proyecto