E.T.S. de Ingeniería Industrial, Informática
y de Telecomunicación

Tarjeta para la implementación de la técnica tensión-frecuencia en un motor de inducción

Grado en Ingeniería
en Tecnologías Industriales

Trabajo Fin de Grado

Román Cebollada Cavero

Eugenio Gubia Villabona

Pamplona, 20 de octubre de 2020

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

RESUMEN

En este proyecto se va a diseñar desde cero una tarjeta electrónica que controle la velocidad de un motor de inducción. El resultado final será una versión de la tarjeta lista para fabricarse así como el código que se implementará en el microcontrolador de la misma.

El proyecto consistirá en el diseño de los algoritmos de control del motor a través de un sistema inversor PWM mediante PSIM, la descripción y diseño del hardware que se implementará en la tarjeta en EAGLE y la programación necesaria para el microcontrolador en MPLAB.

Mediante este trabajo se pretende estudiar y aplicar el conocimiento de los diferentes campos de la electrónica que se han aprendido a lo largo del grado.

LISTA PALABRAS CLAVE

Diseño de tarjeta electrónica

Motor de inducción

Inversor PWM

Algoritmo control

Microcontrolador

Programación motores

Contenido

1. INTRODUCCIÓN Y OBJETIVOS

1.1 Introducción

Hoy en día y desde ya hace mucho tiempo los motores eléctricos son de una gran importancia en la sociedad. Los motores eléctricos obtienen energía eléctrica y la convierten en un par que puede ser utilizado para distintos fines. Un campo de aplicación en auge en la actualidad es el vehículo eléctrico. En la *Figura 1* se puede ver la evolución de la venta de coches eléctricos en Europa en los últimos años.

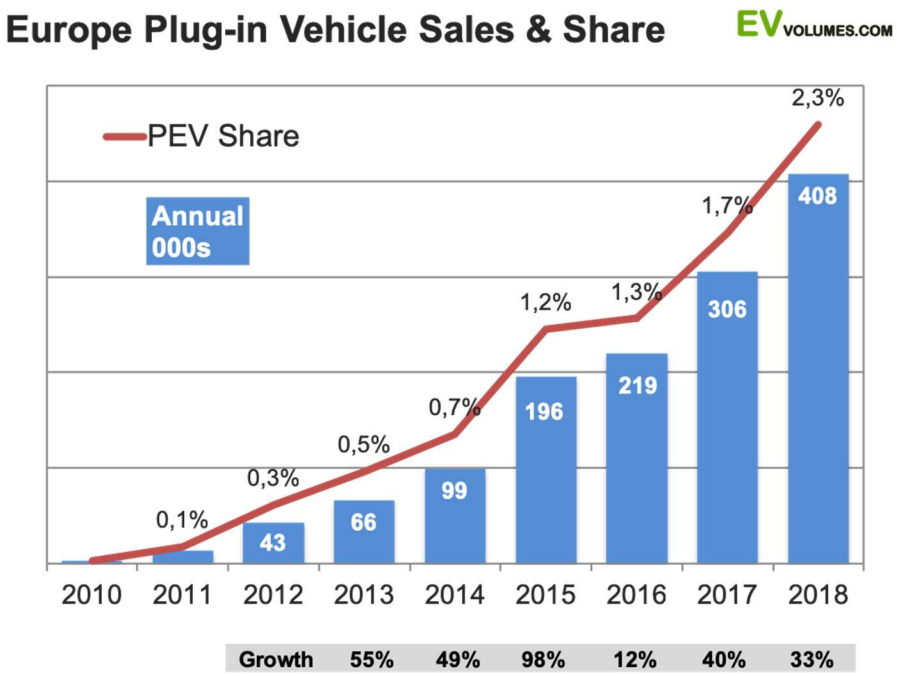

Figura 1. Venta coches eléctricos en Europa 2010-2018
Fuente: ev-volumes.com

En la empresa NTDD, en la que realicé una pequeña estancia de prácticas se dedicaban en parte a diseñar y prototipar cajas de cambios para coches eléctricos. Para poder conocer los parámetros reales y rendimientos de las cajas de cambios, las ensayaban en un banco de pruebas. Para rodar la caja y poder comprobar su funcionamiento se utilizaba un motor de inducción, el cual se quería controlar desde una tarjeta electrónica.

Es por este motivo que el trabajo se va a centrar en el diseño de esta tarjeta o PCB con la que se va a poder controlar la velocidad de un motor, así como comprobar ciertos indicadores de riesgo. La dinámica de un motor se determina mediante un conjunto de ecuaciones que definen su comportamiento. Para la aplicación descrita la dinámica no es muy alta, ya que el objetivo es solamente controlar con exactitud la velocidad de giro.

Cabe mencionar que debido a la situación extraordinaria causada por el COVID-19 este proyecto es puramente teórico ya que no ha resultado posible realizar los ensayos en un laboratorio. Mediante el programa PSIM se han realizado múltiples simulaciones para comprobar el funcionamiento de las soluciones aplicadas.

1.2 Objetivos

El principal objetivo de este Trabajo de Fin de Grado es diseñar y programar una tarjeta electrónica que sea capaz de controlar la velocidad de un motor de inducción. Esta tarjeta debe ser del menor tamaño posible así como garantizar el correcto funcionamiento ante ciertos parámetros de riesgo.

Diseñar una tarjeta electrónica es un proceso complejo en el que se han de tener en cuenta muchos factores. Para definir sus componentes el punto de partida es saber que se tiene que controlar y cómo hacerlo. Una vez se conoce el motor sobre el que se ha de actuar y su rango de operación se puede comenzar a diseñar la tarjeta con sus especificaciones al completo. Es por esto que el trabajo se va a estructurar de la siguiente manera:

- Sistema de potencia: En esta parte se va a explicar el principio de funcionamiento de un motor eléctrico y su integración en el resto de las partes necesarias para la etapa de potencia del sistema.
- Etapa de control: Conociendo el motor y sus características se puede proceder al diseño de los bloques y parámetros necesarios para diseñar un lazo de control de velocidad.
- Implementación física: Se describen y analizan todos los componentes que formarán parte en el diseño de la tarjeta electrónica.

- Programación: El último paso será el de programar el microcontrolador de la tarjeta electrónica para que realice las funciones establecidas en la etapa de control. Para realizar el control de la velocidad se ha empleado la técnica de tensión/frecuencia (V/f=cte).

2. DESCRIPCIÓN DE LA ETAPA DE POTENCIA

Para poder diseñar una tarjeta que sea capaz de controlar un motor de inducción hay que conocer, en primer lugar, cómo funcionan estos motores. Partiendo de ese análisis, se estudia de qué manera es posible controlar su velocidad.

2.1 Motor

El motor asíncrono, o más comúnmente llamado motor de inducción, tiene la característica de que es capaz de producir par a cualquier velocidad excepto a la de sincronismo. En la *Figura 2* se ve la sección de un motor de este tipo.

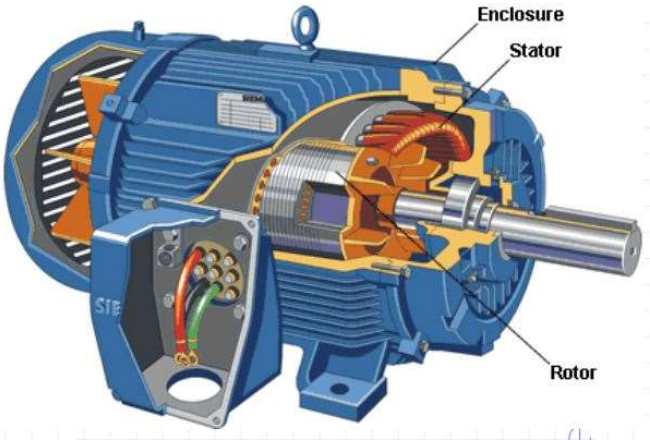

Figura 2. Sección motor de jaula de ardilla
Fuente: ticgalicia.com

Como cualquier maquina rotativa, ésta cuenta con un estator (inductor) y un rotor (inducido). El estator está conformado por una serie de chapas ferromagnéticas apiladas. Estas chapas cuentan con unas ranuras interiores en las que se encuentra situado un devanado trifásico de paso acortado con un determinado número de polos. Cuando se alimenta al devanado con un sistema trifásico de tensiones, se genera un sistema trifásico de corrientes, encargado de crear un campo magnético rotativo en el entrehierro.

El rotor se sitúa en el interior del estator y está constituido por un conjunto de chapas magnéticas que se juntan para formar un cilindro. Estas

chapas cuentan con un ranurado exterior donde va alojado el devanado. Como la velocidad de giro es diferente a la velocidad de sincronismo, que es a la que gira el campo magnético, el rotor ve un flujo variable gracias al cual se genera un sistema trifásico de corrientes en el rotor. La interacción entre los campos creados por las corrientes del estator y las del rotor da lugar a un par de giro.

En el motor de jaula de ardilla el rotor tiene unas ranuras longitudinales en las que se colocan unas barras conductoras cuyos extremos están cortocircuitados. Su comportamiento es semejante al motor de rotor bobinado.

El motor es uno de los pilares más importantes del sistema, es el que girará a una velocidad determinada, así como sobre el que se implementaran los cambios realizados por el control. Para conocer mejor su comportamiento, se va a estudiar el principio de funcionamiento del motor asíncrono de jaula de ardilla mediante sus devanados equivalentes. La explicación que sigue se basa en información ofrecida en [1].

2.1.1 Principio de funcionamiento

Se alimenta el estator con un sistema trifásico equilibrado de corrientes. Según el Teorema de Ferraris, al estar las corrientes desfasadas 120º en el tiempo, y los devanados están desfasados 120º en el espacio, se genera un campo magnético rotativo Fmms que gira a la velocidad Ω_s.

$$i_A = \sqrt{2}\, I_s \cos\left(\omega_s t\right) \quad (1)$$

$$F_{mms}(\theta, t) = \frac{3 N_{seq} \sqrt{2}\, I_s}{\pi}\, sen\left(\theta - \omega_s t\right) \quad (2)$$

$$\Omega_s = \frac{\omega_s}{p} \quad (3)$$

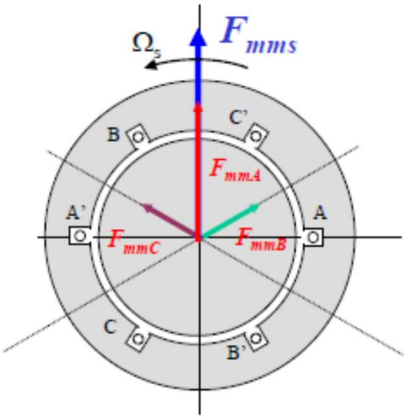

Figura 3. Campo magnético estator.
Fuente:[1]

La tensión en el devanado A, despreciando resistencias internas y flujos de fugas será:

$$v_A = \frac{d\psi_A}{dt};\ \psi_A = N_{seq}\Phi_A \qquad (4)$$

$$\Phi_A = \int_s B\ ds = \Phi_{max}\cos(\omega_s t) \qquad (5)$$

$$v_A = -N_{seq}\ \Phi_{max}\ sen\ (\omega_s t) \qquad (6)$$

Para las fases B y C, las ecuaciones son equivalentes pero desfasadas 120º y -120º.

En el rotor, se supone inicialmente que los devanados están en circuito abierto. Debido al capo magnético rotatorio del estator, el rotor, que gira a una velocidad Ω_m distinta a la de sincronismo Ω_s , ve un campo magnético variable.

Debido a este campo magnético, se inducen tensiones en los devanados del rotor. Se desprecian resistencias internas y flujos de fugas.

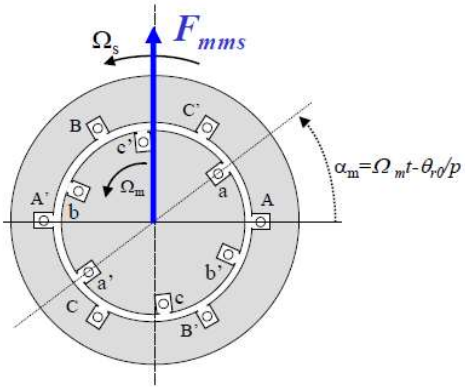

Figura 4. Velocidad mecánica y de sincronismo
Fuente:[1]

$$v_a = \frac{d\psi_a}{dt} \quad \psi_a = N_{req}\Phi_a \;\; \omega_s = p\,\Omega_m \tag{7}$$

$$\Phi_A = \int_s B\,ds = \Phi_{max}\cos\left((\omega_s - \omega_m)\,t + \theta_0\right) \tag{8}$$

$$v_a = -N_{req}\,\Phi_{max}\,(\omega_s - \omega_m)\,sen\left((\omega_s - \omega_m)\,t + \theta_0\right) \tag{9}$$

Para las fases B y C, las ecuaciones son equivalentes pero desfasadas 120º y -120º.

La pulsación de las variables eléctricas del rotor es la diferencia en las velocidades del del campo magnético giratorio y la mecánica del rotor. Se define el deslizamiento s como la relación entre la pulsación del rotor y la velocidad eléctrica de sincronismo.

$$s = \frac{\omega_r}{\omega_s} = \frac{\omega_s - \omega_m}{\omega_s} \tag{10}$$

El deslizamiento indica la distancia relativa entre la velocidad a la que gira la maquina y la velocidad de sincronismo. Es un parámetro fundamental de la maquina asíncrona y puede expresarse en tanto por ciento.

Si la maquina girase a la velocidad de sincronismo, el deslizamiento tendría un valor de 0 y el rotor no vería variación del flujo magnético, por lo que la tensión inducida seria 0. Por el contrario, si el rotor está bloqueado, el deslizamiento tendría un valor de 1, y las ecuaciones serían similares a las de un transformador.

En función del deslizamiento, la ecuación para la tensión de la fase a del rotor es la siguiente:

$$v_a = -N_{req}\, \Phi_{max}\, s\, \omega_s\, sen\, (s\, \omega_s\, t + \theta_0) \tag{11}$$

En el siguiente paso se cortocircuitan los devanados del rotor. Esto hace circular un sistema trifásico de corrientes en los que la fase b y c están desfasadas 120º y 240º.

$$i_a = \sqrt{2}\, I_r sen(\omega_r t + \varphi_r + \theta_0) = \sqrt{2}\, I_r sen\, (s\, \omega_s t + \varphi_r + \theta_0) \tag{12}$$

Al estar las corrientes desfasadas 120º en el tiempo y los devanados 120º en el espacio, se crea un campo magnético rotatorio F_{mmr} según el Teorema de Ferraris.

El campo magnético gira a la pulsación del rotor, $\boldsymbol{\Omega_r}$, respecto de este. A su vez el rotor gira a una velocidad mecánica $\boldsymbol{\Omega_m}$. Por lo que la velocidad del campo magnético generado es $\Omega_m + \Omega_r = \boldsymbol{\Omega}$.

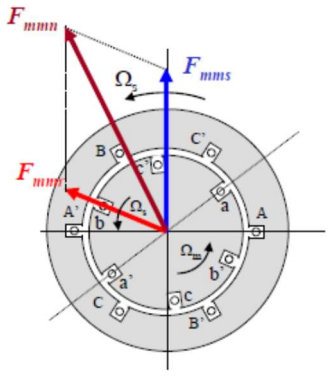

Figura 5. Campo magnético total
Fuente:[1]

Los campos magnéticos generados por el rotor y por el estator, F_{mmr} y F_{mms} giran a la misma velocidad (Ω_s) para cualquier velocidad Ω_m del motor. Debido a la interacción de ambos campos magnéticos, se genera un par que es proporcional a las amplitudes de los campos y al seno del ángulo de desfase entre ellos en radianes eléctricos:

$$T = -\frac{p\,\mu_0\,\pi\,R\,l}{g}\,F_{mmr}F_{mms}\,sen\,(\theta_e) = -k\,|F_{mmr}\text{^}F_{mms}| \qquad (13)$$

Este par es generado para cualquier velocidad diferente a la de sincronismo, dado que no se inducirían tensiones en el rotor y estas no generarían corrientes ni a su vez estas campo magnético.

Para el modelo completo del motor asíncrono en régimen permanente habría que tener en cuenta las resistencias internas, R_s y R_r, y los flujos de fugas, L_s y L_r, así como el flujo en el entrehierro.

2.1.2 Balance de potencias

A partir del circuito equivalente del motor asíncrono se va a analizar el balance de potencias en cada parte del motor desde que entra hasta que se consigue convertir esta potencia en el par que hará girar la máquina.

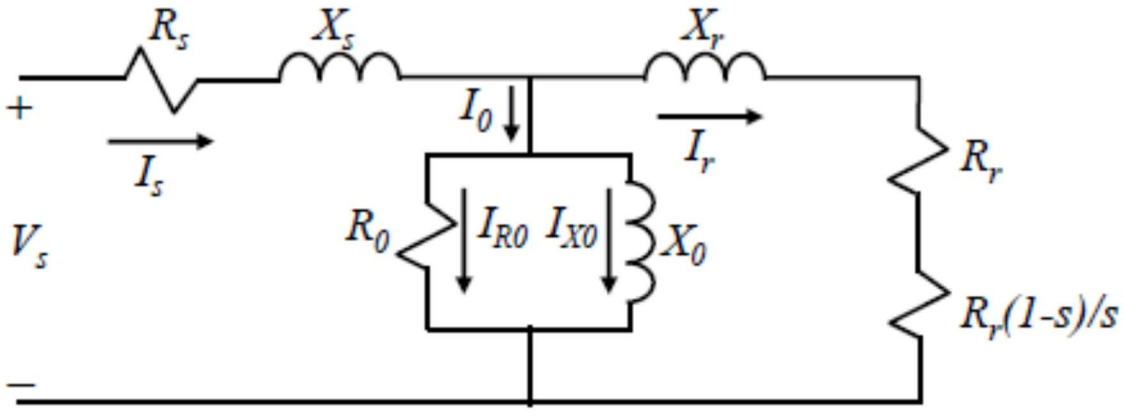

Figura 6. Circuito equivalente máquina inducción
Fuente:[1]

Para entender la potencia perdida en cada una de las partes del motor, se ha desarrollado en la *Figura 7* un diagrama de flujo de potencias.

Se muestra como la potencia que entra a la maquina sufre diversas perdidas en cada una de las etapas del motor, hasta llegar a convertir la potencia que se obtiene en par.

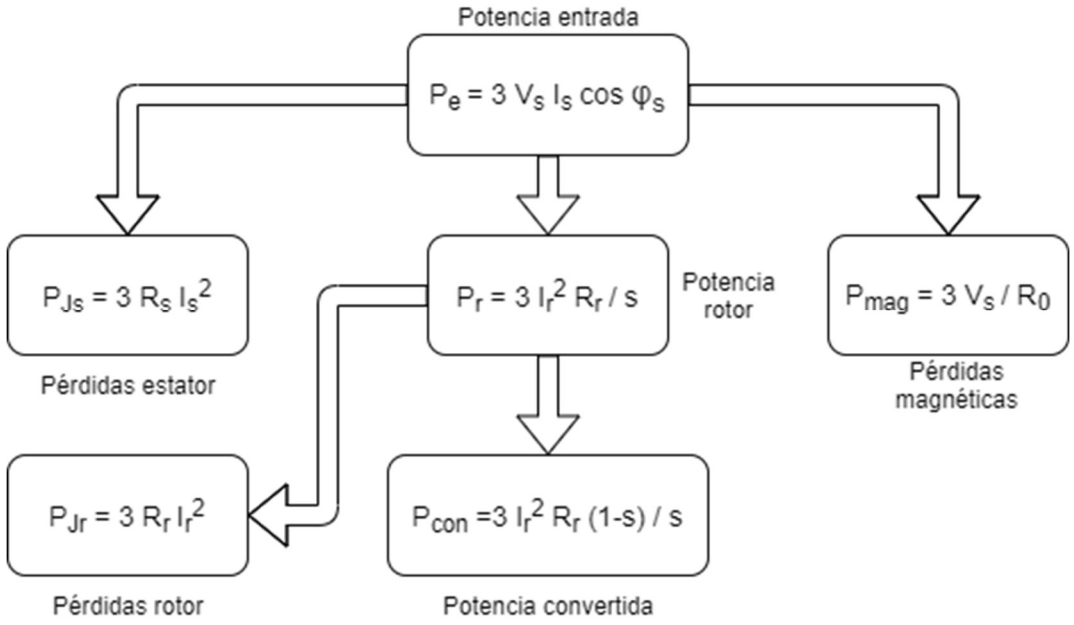

Figura 7. Flujo de potencias

A partir de la potencia convertida que se obtiene del diagrama y con las ecuaciones que se deducen del circuito equivalente, se puede obtener la expresión del par electromagnético generado por el motor y de cómo esta depende de variables que se pueden conocer.

$$T = \frac{P_{conv}}{\Omega_m} = \frac{3\ {I_r}^2\ R_r \frac{(1-s)}{s}}{\Omega_s\ (1-s)} \tag{14}$$

$$T = \frac{3\ p\ {I_r}^2\ R_r}{s\ \omega_s} = \frac{3\ p\ R_r/s\ V_s}{\omega_s \left[\left(\frac{R_r}{s} + R_s\right)^2 + {X_e}^2\right]} \tag{15}$$

Una vez se conoce la expresión del par que genera el motor se puede calcular gráficamente para cada uno de los valores de deslizamiento o de velocidad de giro. En la *Figura 8* se muestra el par que es generado, así como los puntos de especial interés.

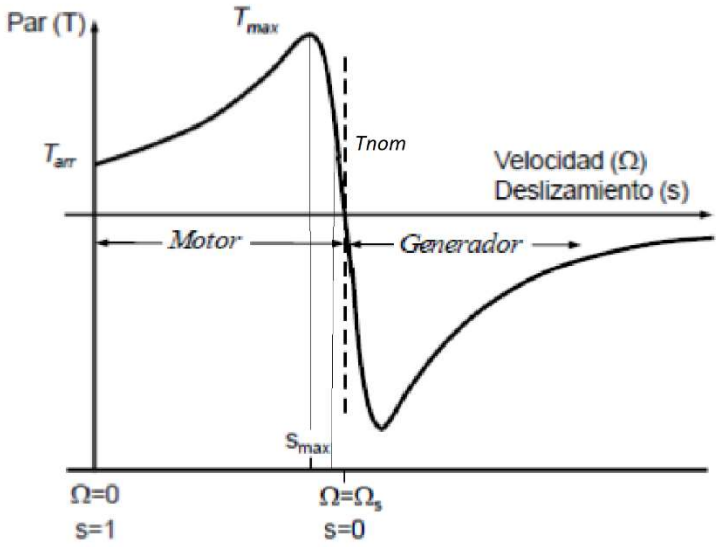

Figura 8. Curva par-velocidad general

Con esta curva se pueden obtener puntos característicos del motor como el par de arranque (T_{arr}), cuando el deslizamiento es s=1, así como el valor del par máximo (T_{max}).

Los motores asíncronos suelen trabajar con valores nominales de deslizamiento de entre 4% y 7%. En esta zona de la curva, la expresión se aproxima bastante a una recta que pasa por s=0.

A su vez se pueden definir dos zonas de funcionamiento: estable e inestable. La zona a la a la izquierda del T_{max} se define como zona inestable, ya que el sistema se desestabiliza ante cualquier perturbación. Esto se debe a que en esta zona de la curva si el par de carga aumenta, la velocidad aumentaría también en vez de disminuirse.

Contrariamente, a derecha del T_{max} es la zona estable, donde además de semejarse a una recta, el sistema se equilibra ante cualquier perturbación. En esta zona al aumentar el par de carga el motor se frena, aumentando a su vez el par ejercido para vencer a la carga.

Los motores de inducción trabajan en la zona estable. Para ello el par motor en la zona inestable debe ser mayor que el par de carga en todo momento. Todas las vibraciones de una máquina al arrancar, así como las excesivas corrientes que se consumen se deben a estar trabajando en la

zona inestable. En la *Figura 9* se muestran ambas regiones de trabajo del motor.

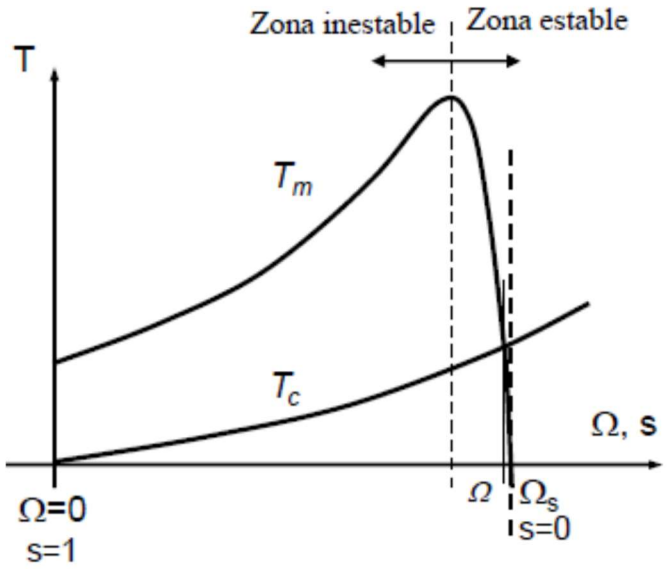

Figura 9. Curva par-velocidad y par de carga

2.1.3 Variación de velocidad

El objetivo de este proyecto es, en última instancia, controlar la velocidad del motor. La velocidad de un motor asíncrono se define en la siguiente expresión:

$$\Omega_m = \Omega_s(1-s) = \frac{\omega_s}{p}(1-s) = \frac{2\pi f_s}{p}(1-s) \tag{16}$$

Para que la velocidad de giro varíe, se ha de modificar alguna de las variables: el número de pares de polos (p), el deslizamiento (s) o la frecuencia de alimentación (f_s).

Para este trabajo se va a emplear el método de control **V/f=cte.** para variar tanto la frecuencia como el voltaje de alimentación. Para que no varíe la corriente de magnetización, y de esta manera el flujo, se debe mantener:

$$I_m = \frac{V_s}{X_0} = \frac{V_s}{2\pi\, f_s\, L_0} \longrightarrow \frac{V_s}{f_s} = cte \tag{17}$$

Con este control se consigue trasladar la curva par-velocidad a lo largo del eje horizontal para diferentes valores de *fs*. De esta manera se consiguen curvas paralelas a la original, con lo que se genera el mismo par, pero para diferentes velocidades.

Sin embargo, al aumentar la frecuencia por encima de su valor nominal, comienza a disminuir la capacidad de par del motor. Esto se debe a que no es posible aumentar el valor de V_s por encima del nominal y, por lo tanto, el valor de V_s/f_s deja de ser constante. Por lo que al aumentar la frecuencia por encima de su valor nominal, la relación comienza a disminuir y con ella el par que ejerce el motor.

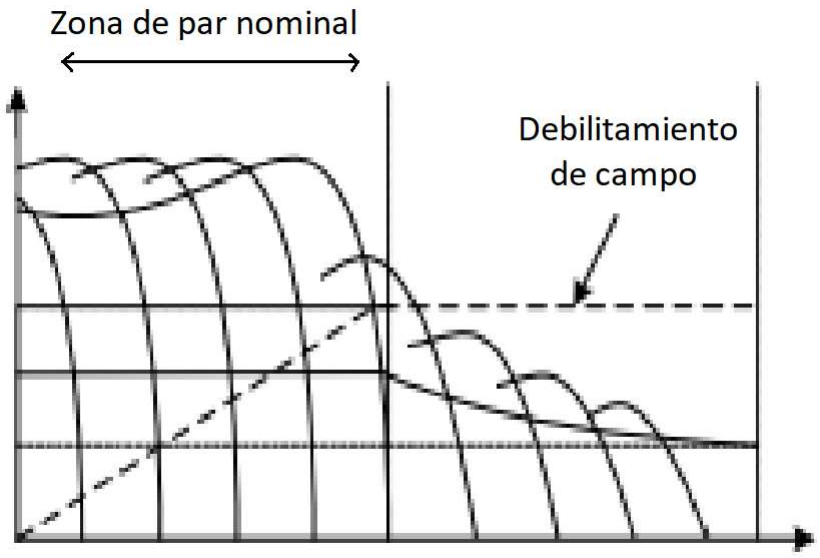

Figura 10. Curvas paralelas y debilitamiento par motor

2.1.4 Características del motor

El motor empleado tanto para los cálculos como para las simulaciones es un motor de jaula de ardilla cuyos parámetros relevantes en el presente trabajo se muestran en la Tabla I.

Tabla I. Parámetros motor de inducción

Motor de inducción de jaula de ardilla	
Tensión nom. fase-neutro	230 V
Potencia nom.	48 kW
Par nom.	200 N·m
Resistencia estator	0,294 Ω
Resistencia rotor	0,156 Ω
Inductancia estator	1,36 mH
Inductancia rotor	0,74 mH
Inductancia magnetización	41 mH
Nº de polos	4
Momento de inercia	0,4 $kg \cdot m^2$

Como se observa en la Tabla I.

Tabla I, el par nominal es de 200 N·m. Este tipo de sistemas están preparados para aguantar un par 50% mayor que el nominal durante un breve periodo de tiempo, en caso contrario las excesivas corrientes que circulan por el motor podrían quemarlo. Es por este motivo que en control que se va a realizar el máximo par que se va a permitir es de 300 N·m. En la *Figura 11* se muestra la curva característica par-velocidad de este motor, así como los puntos de especial interés.

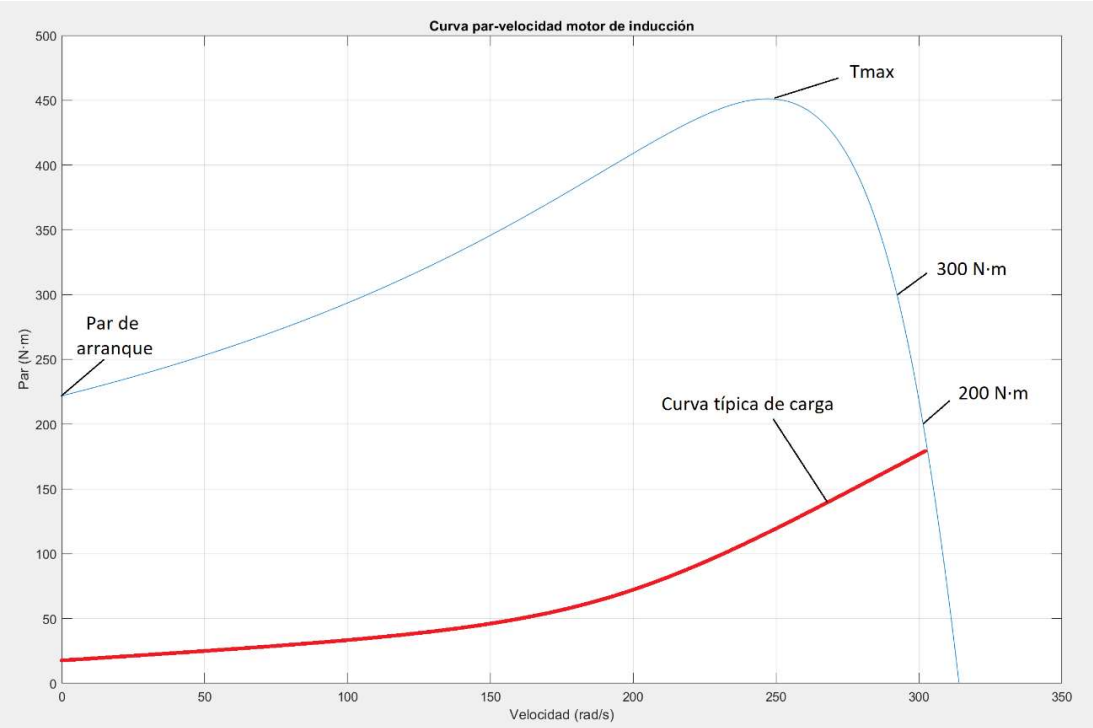

Figura 11. Curva par-velocidad motor

Se han indicado en la figura los puntos de mayor interés como lo son el par nominal (200 N·m), el par máximo de funcionamiento (300 N·m) y el par de arranque (210 N·m). Para que el motor sea capaz de arrancar, la carga debe ofrecer un par estático menor de 210 N·m. En la aplicación objeto de este TFG la carga típica ofrece un par de arranque casi nulo que crece proporcional a la velocidad, esta curva es la indicada en la *Figura 11.*

2.2 Convertidor de potencia

Como se ha visto anteriormente, para variar la velocidad del motor será necesario modificar tanto la tensión como la frecuencia de alimentación. Para lograr el control tensión/frecuencia se va a utilizar un inversor trifásico PWM. La función de los inversores PWM es controlar la frecuencia y amplitud de la tensión trifásica a partir de una tensión CC.

La modulación por ancho de pulsos (PWM) consiste en comparar una única onda triangular con 3 señales sinusoidales (moduladoras) desfasadas 120º entre sí. La señal de salida de cada uno de los comparadores es la señal de control de un puente rectificador de IGBTs. En la *Figura 12* se muestra un esquema simple de un módulo inversor PWM, donde las salidas a, b y c

simulan cada una de las fases del motor. [2]

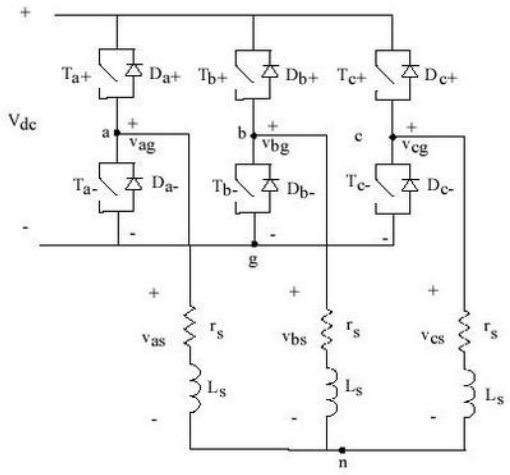

Figura 12. Esquema inversor PWM
Fuente: Wikipedia.com

Para conseguir la tensión DC que alimente al inversor PWM se va a utilizar un puente rectificador de onda completa. Este dispositivo es capaz de convertir la corriente alterna en corriente continua empleando un puente de diodos. En la *Figura 13* se muestra el esquema típico de configuración de estos sistemas.

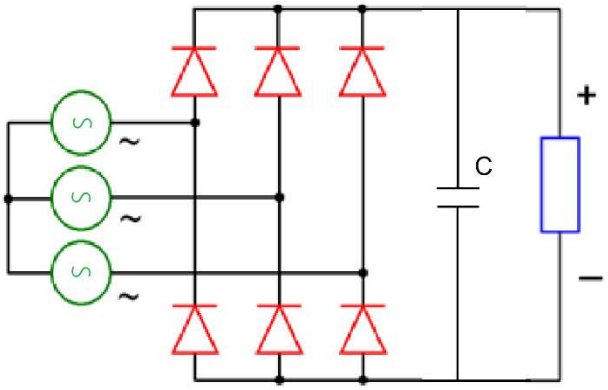

Figura 13. Esquema puente rectificador de diodos

El principio de funcionamiento de este puente es que se conecta la duente de más tensión a la parte positiva de la carga la de menos tensión a la negativa, la fuente restante queda desconectada. Mediante los diodos y añadiendo un condensador en la salida se consigue una onda prácticamente continua. En la *Figura 14* se muestran las ondas de los diodos y el pequeño rizado de la salida, el cual se atenúa en gran medida mediante

un condensador. Queda marcado en rojo la onda de salida del rectificador.

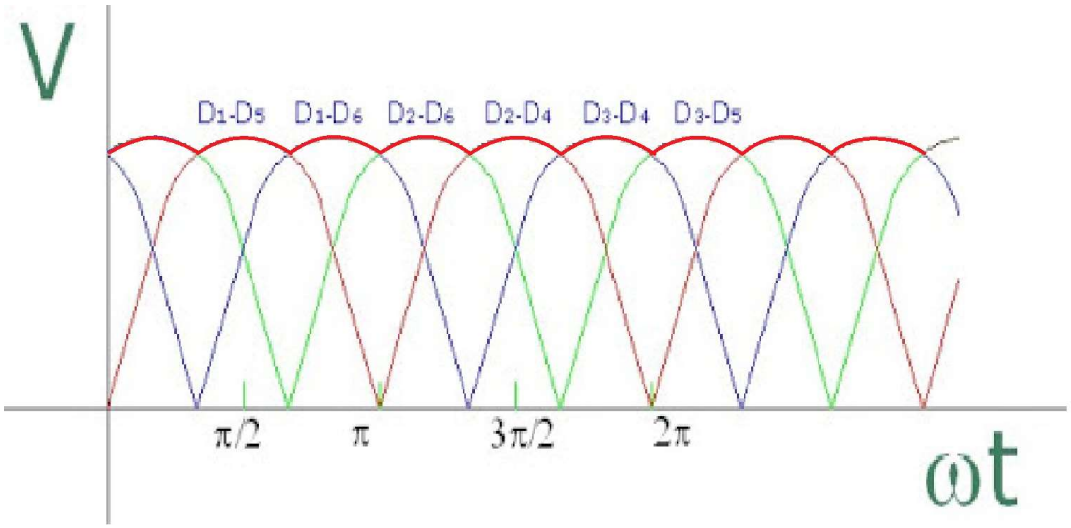

Figura 14. Formas de onda de puente rectificador de diodos

Ambos convertidores se conectan según el esquema de la *Figura 15* para conformar junto con el motor la etapa de potencia al completo.

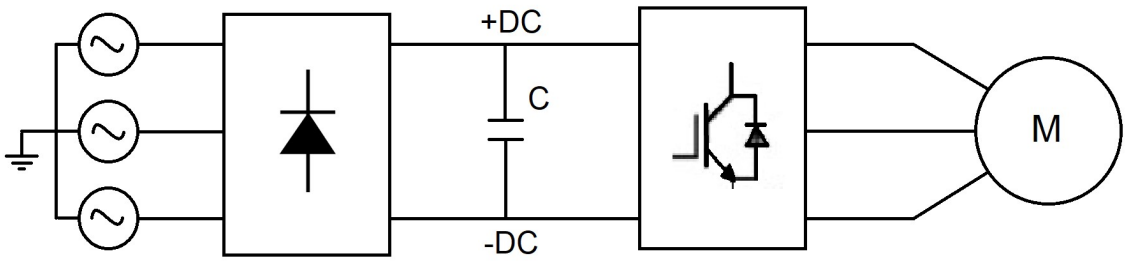

Figura 15. Esquema de potencia completo

3. ALGORITMOS COTROL

Una vez se conoce el motor que se quiere controlar, el método que se va a emplear para controlarlo y el convertidor de potencia que se debe gobernar, se va a proceder a diseñar los algoritmos de control necesarios para que el motor gire a la velocidad que se establezca.

Para el diseño y simulación de los algoritmos de control se ha empleado el software PSIM. Mediante este programa se ha conseguido adecuar tanto el voltaje como la frecuencia de entrada del motor asíncrono empleando diferentes elementos electrónicos. Se han colocado varios voltímetros en el esquema para poder realizar simulaciones y cálculos más precisos.

3.1 Motor

El motor empleado tanto para los cálculos como para las simulaciones es un motor de jaula de ardilla cuyos parámetros han descrito anteriormente. Para poder realizar un control más real se han añadido un par resistente, simulando una carga real, así como sensores de velocidad y par en el tren mecánico del motor.

El par resistente es proporcional a la velocidad (rad/s) con un coeficiente de 1,33. Además el momento de inercia de la carga es el mismo que el del motor: 0,4 $kg \cdot m^2$. El sensor de velocidad genera una señal de voltaje equivalente a la velocidad en rad/s. La señal generada por el sensor de par equivale a la medida en N·m.

En la *Figura 16* se muestra el tren mecánico al completo configurado en PSIM, incluyendo el motor, el par resistente y los sensores mencionados.

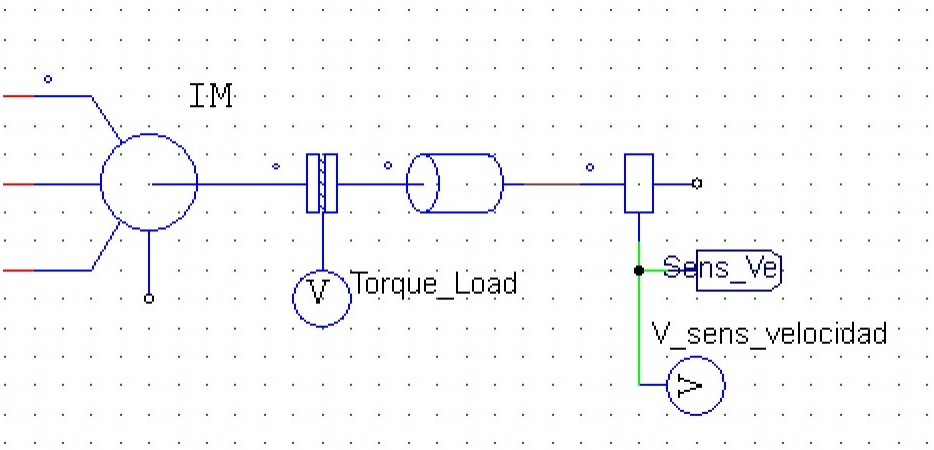

Figura 16. Tren de mecánico en PSIM

3.2 Sistema PWM

Para lograr el control tensión/frecuencia se emplea en primer lugar un inversor trifásico PWM. Su función es la de controlar tanto la tensión como la frecuencia de salida a partir de una tensión DC de +- 400 V de entrada. La señal de control, para cada fase desfasada 120º, se compara con una señal triangular de amplitud 1 y 5 kHz de frecuencia.

La salida de este comparador es llevada a un puente rectificador de IGBTs, en el que cada fase cuenta con dos de ellos, de los cuales uno es el negado del otro. Mediante la conmutación de los transistores se consigue modular la tensión de manera que se asemeja casi por completo a una señal sinusoidal.

Se incorporan además dos sensores de corriente en las fases A y B, así como un voltímetro en la fase A y otro entre las fases A y B. En *Figura 17* se puede ver la configuración que se ha utilizado en PSIM.

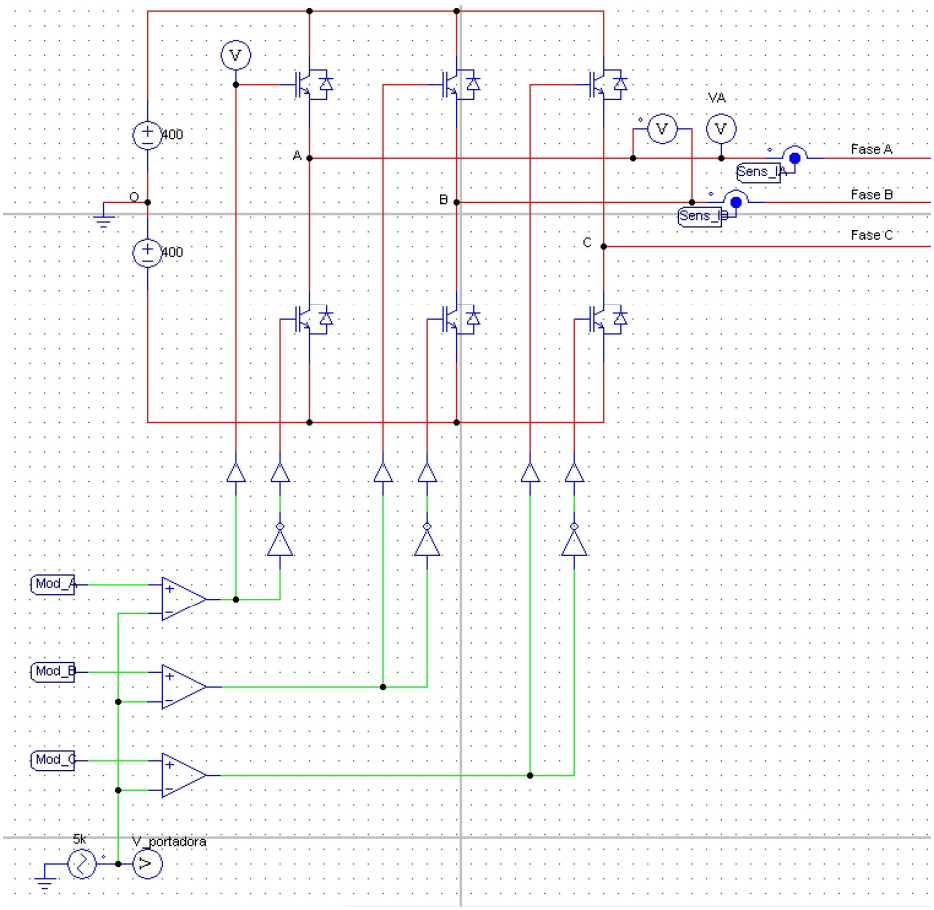

Figura 17. Esquema PWM en PSIM

Para entender de una manera más grafica el funcionamiento del inversor PWM se incluye la *Figura 18* con la simulación de las señales moduladoras, triangular y voltaje de salida en un intervalo de apenas 1 milisegundo.

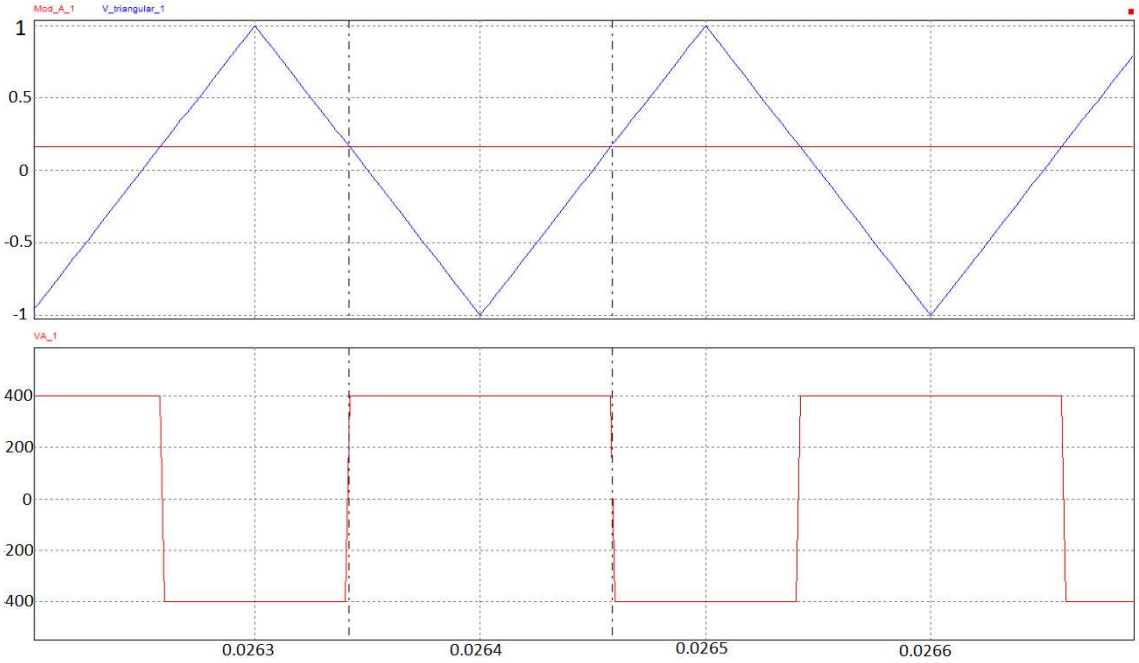

Figura 18. Simulación señales moduladora y triangular de PWM

Se puede observar como cuando la señal moduladora es mayor que la señal triangular, se obtiene un voltaje negativo en la fase A. Por el contrario, cuando la señal moduladora es menor que la triangular, el voltaje en la fase A es positivo.

De esta manera se consigue modificar el ancho de los pulsos en la fase A, ya que la señal moduladora es sinusoidal, como indican las siglas PWM: Pulse Width Modulation.

3.3 Lazo abierto

El primer paso fue diseñar un sistema de control en lazo abierto. Se trata de un sistema simple en el que la salida no forma parte del sistema, es decir, no se comprueba ni se corrige en ningún momento si la salida que se ha obtenido es la deseada o no. En la *Figura 19* se muestra el diagrama de bloques que se va a utilizar para diseñar el control en lazo abierto.

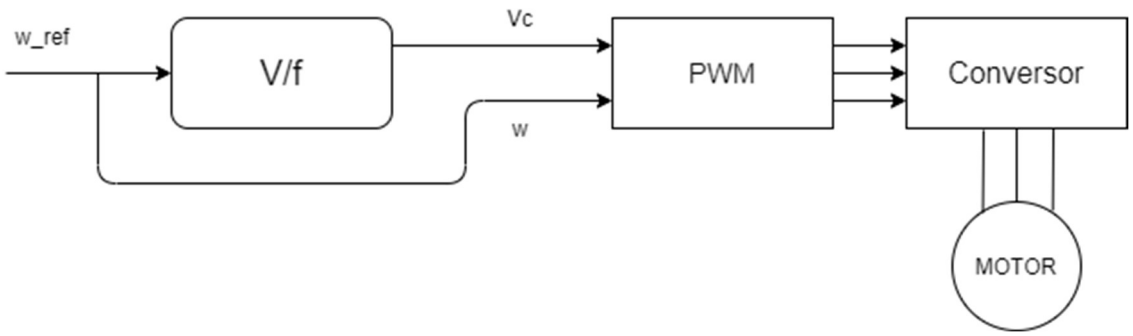

Figura 19. Diagrama bloques lazo abierto

El bloque "V/f" se encarga de convertir la referencia de velocidad en una señal moduladora que entre en la PWM. Para ello se va a estudiar los requisitos de la señal modulante.

La señal moduladora para la fase C se puede obtener restando las señales de la fase A y la fase B, ya que en un sistema equilibrado de tensiones se ha de cumplir en todo momento que:

$$V_A(t) + V_B(t) + V_C(t) = 0 \tag{18}$$

La modulante debe ser una señal sinusoidal de amplitud máxima 1. Para obtener la frecuencia y amplitud deseadas en cada fase del motor, esta señal debe ser semejante a la siguiente expresión:

$$Mod_A = k\, w \sin(w\, t) \tag{19}$$

La referencia de velocidad se introduce en grados por segundo (deg/s), esto se debe a que en PSIM a la función seno entra una señal en grados y sale el seno de los grados que han entrado al bloque. Por ello se integra la referencia de velocidad para después introducirla en el seno. Para la fase B se han restado 120º ya que ambas fases deben estar desfasadas ese valor.

A continuación se multiplica la salida del seno y la velocidad de referencia. De esta manera solo queda restante introducir una constante **k**.

Esta constante es la encargada de adecuar la señal moduladora para que haya un valor adecuado en la tensión de salida.

Para esto hay que tener en cuenta las siguientes consideraciones:

- El bus de DC es de +- 400 V y la tensión nominal del motor es de 325 V.
- La máxima frecuencia de funcionamiento es de 50 Hz.

Como se quiere que el valor de la moduladora sea máximo para la frecuencia de 50 Hz, y el máximo de la moduladora será el equivalente a 325 V, la constante k toma el siguiente valor:

$$\boldsymbol{k = \frac{325}{400} \cdot \frac{1}{360 \cdot 50} = 4.51 \cdot 10^{-5}}$$

En la *Figura 20* se detalla el diagrama de bloques en PSIM con todas las consideraciones que se han mencionado.

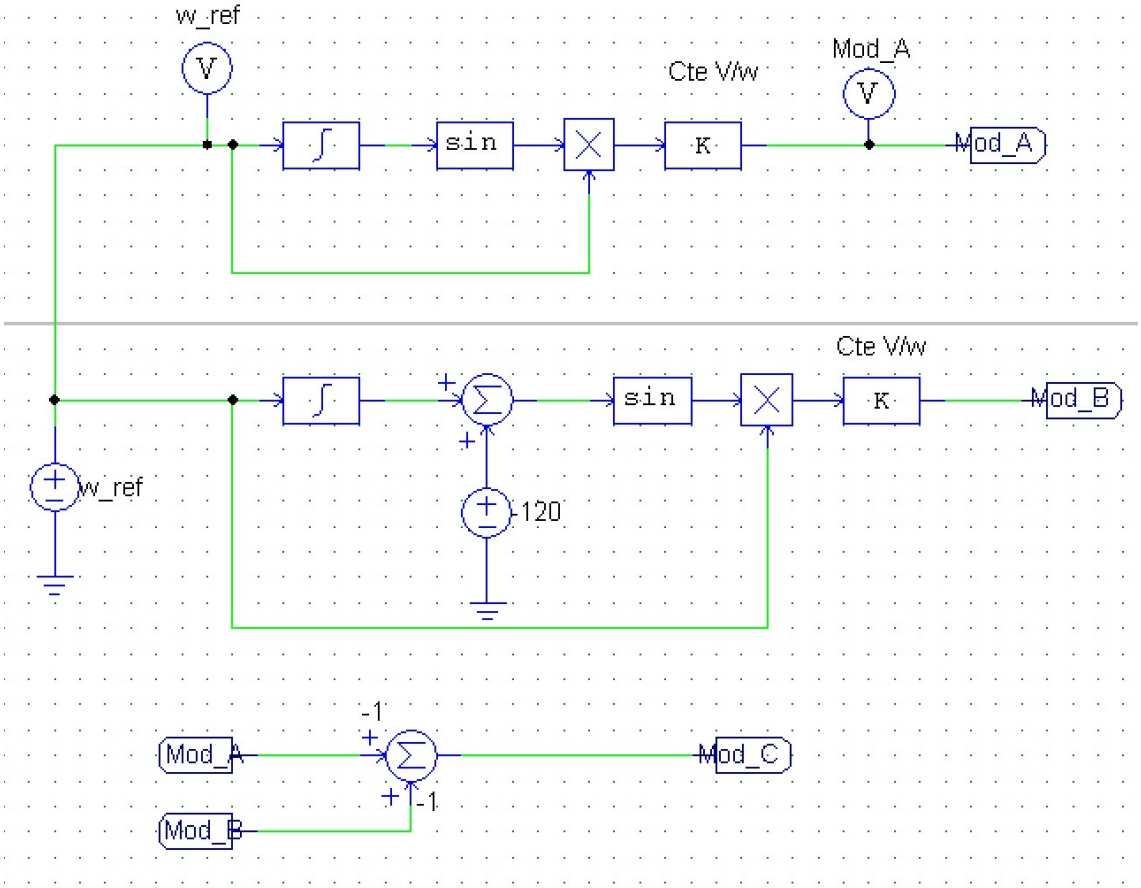

Figura 20. Esquema lazo abierto en PSIM

A continuación, en la *Figura 21* se muestran en una gráfica los resultados tanto de velocidad de referencia como las medidas por el sensor de velocidad de varias simulaciones en PSIM para las distintas las distintas referencias de 10, 20, 30, 40 y 50 Hz. El eje Y de velocidad se mide en rpm mecánicas.

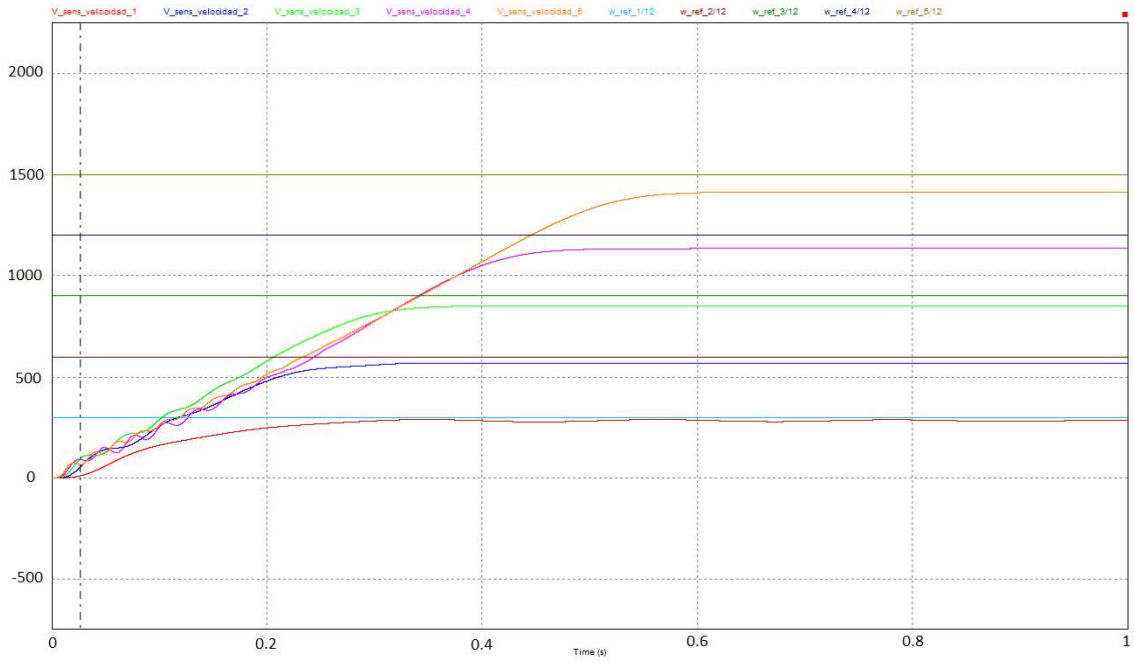

Figura 21. Simulación velocidad real y referencia lazo abierto en PSIM

Se puede observar como el control en lazo abierto no es suficiente para esta aplicación. En ningún caso la velocidad real llega a ser la de referencia, a pesar de que se aproximan. Esto es debido a que no se tiene en cuenta la velocidad real para ajustar el sistema, por lo que simplemente el sistema obedece a la referencia, aunque sigue sin ser suficiente para alcanzar la velocidad deseada.

Otra aspecto que cabe mencionar de esta grafica es el mal comportamiento que ocurre a bajas velocidades. La caída de tensión en la resistencia del estator aumenta a medida que disminuye la tensión. Esto se debe a que apenas hay caída en las reactancias ($X_e = 2\ \pi\ f_s\ L_e$), y toda la tensión cae en la resistencia, lo que provoca una disminución del par. Este problema se tratará de solucionar en uno de los siguientes apartados.

3.4 Lazo cerrado

A diferencia de los sistemas en lazo abierto, el lazo cerrado se realimenta de la señal de salida. Se restan la señal de entrada y la salida, de esta manera se genera un error, y es sobre este error sobre el que trabaja el sistema, con la intención de lograr que el error sea 0. Como se ha podido comprobar, con un sistema de lazo abierto no es suficiente, por lo que se va a estudiar el sistema con realimentación.

El diagrama de bloques incluye, un sensor de velocidad y un controlador, así como el lazo de realimentación. El esquema se muestra en la *Figura 22.*

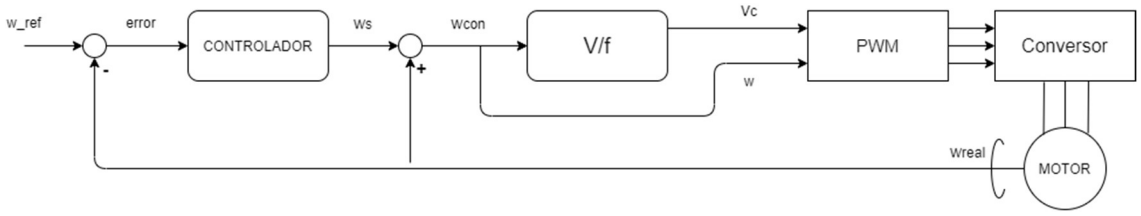

Figura 22. Diagrama bloques lazo cerrado

A partir del error generado entre la velocidad de referencia y la velocidad real, el controlador va a decidir la cantidad de ayuda necesaria para que el error se reduzca hasta ser 0. En este caso, la señal que llega al bloque "V/f" no va a ser directamente la velocidad de referencia, sino que será la velocidad real más o menos la salida del controlador. De esta manera se conseguirá una mayor precisión en la salida.

Se han incluido dos preajustes en el diseño del lazo:

- Se ha añadido un **filtro de 1er orden** con **fc=0.4 Hz** en la referencia. Esto se debe a que las ecuaciones del control V/f son válidas únicamente en régimen permanente, es decir, cuando $\mathrm{dw/dt} = 0$. Esto no se cumple al controlar la velocidad, ya que hay transitorio de una consigna a otra. Mediante el filtro de 1er orden se consigue disminuir la derivada respecto al escalón. De esta manera las ecuaciones del control son más o menos precisas.

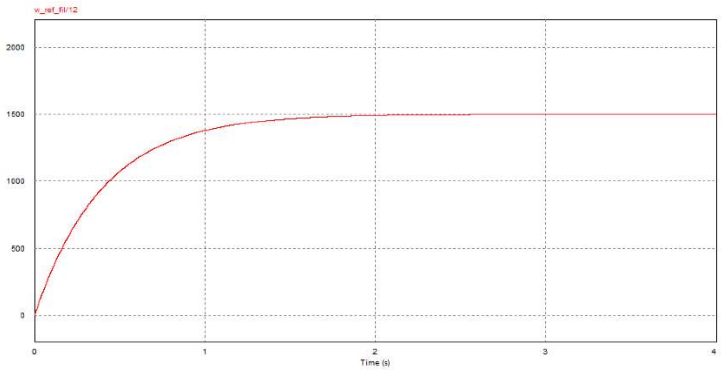

Figura 23. Efecto del filtro en la referencia

- La señal del sensor de velocidad está en rpm mecánicos, por lo que se ha añadido una constante para que sus unidades sean deg/s eléctricos.

$$K_{ud} = 2 \cdot \frac{360}{60}$$

El controlador está formado por un PI, un limitador y una constante. El **limitador** es la parte más sencilla de configurar. La salida del PI es el par que debería de ejercer el motor para alcanzar la velocidad deseada. Como se ha mencionado anteriormente al hablar del motor, el máximo par de trabajo que se permite es de 300 N/m, por lo que el limitador se configura entre **-300 y 300.**

El valor de la **constante** es crucial para el funcionamiento del controlador, es la encargada de relacionar el par que proviene del PI y del limitador, con la velocidad correspondiente según la curva característica propia del motor.

Como ya se ha comentado anteriormente, en la curva de par-velocidad del motor, la zona de trabajo se puede llegar a aproximar a una recta. Esta aproximación no es del todo verídica, pero puede darse por válida para puntos de la curva cercanos a los valores escogidos. En la *Figura 24* se pueden ver las rectas que pasan por el origen y por los puntos de par máximo y nominal.

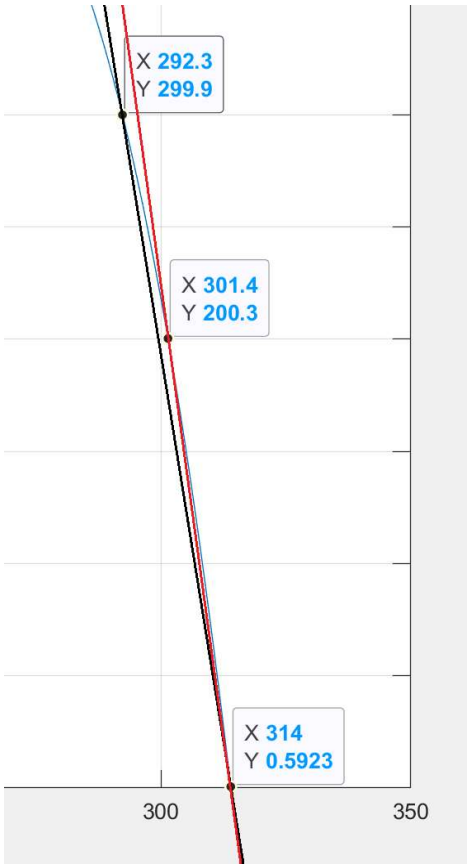

Figura 24. Ampliación curva par-velocidad y recta aproximada

Para el cálculo de la recta aproximada se han elegido los puntos de par nominal y de corte con el eje horizontal, ya que como puede observarse, de esta manera se consigue generar la recta que más se asemeja a la curva en la mayor cantidad posible de puntos.

El eje de velocidad está en rad/s, pero en el diagrama se trabaja en deg/s, por lo que se debe hacer el cambio de unidades.

$$\Delta w = (301{,}4 - 314) \cdot \frac{1}{2\pi} \cdot 360 = 721{,}92\ deg/s$$

Por lo que la constante toma el valor de:

$$\boldsymbol{k} = \frac{\Delta w}{\Delta T} = \frac{721{,}92\ deg/s}{200\ N \cdot m} = \mathbf{3{,}61}$$

Gracias a esta constante, se consigue convertir el par que ha de ejercer el motor, proveniente del PI, en la velocidad correspondiente para entrar en el bloque “V/f”.

Para configurar los parámetros del controlador proporcional e integral es necesario analizar el sistema al completo para conseguir un diagrama de bloques cuya función de transferencia sea posible poder resolver. Este diagrama incluye al controlador, el motor y el par resistente.

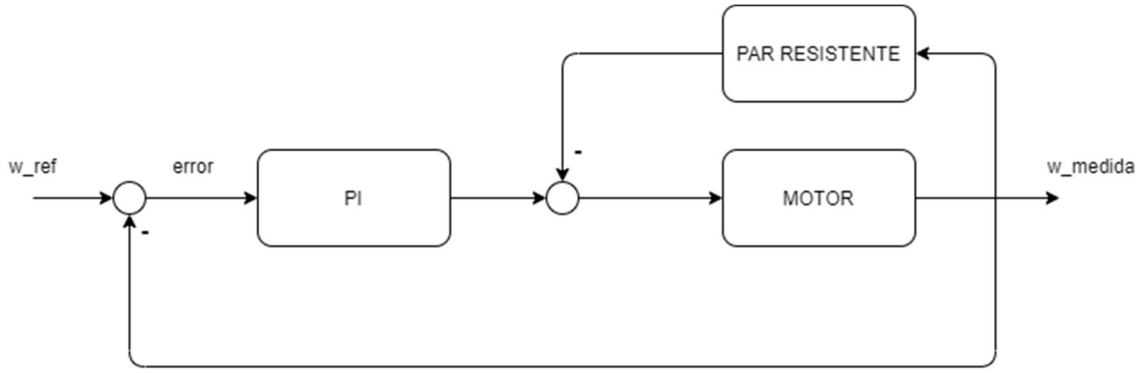

Figura 25. Diagrama bloques lazo cerrado

El par resistente ejerce un par debido a la fricción proporcional a la velocidad del eje, por lo que su función de transferencia es una contante:

$$Pr(s) = k_f$$

La función de transferencia de un controlador PI es la siguiente:

$$PI(s) = k_p \frac{1 + sT}{sT} \tag{20}$$

Donde k_p es la constante proporcional y T es la constante de tiempo del integrador.

En la función de transferencia del motor de inducción se tiene en cuenta la inercia del propio motor. La función queda de la siguiente manera:

$$M(s) = \frac{1}{J_m \cdot s}$$

Donde J_m es la inercia propia del motor inercias que se han mencionado.

El último ajuste que se ha de tener en cuenta, es que el sensor de velocidad lee la velocidad mecánica del eje, mientras que la velocidad de referencia se refiere a la velocidad eléctrica. Este ajuste se tendrá en cuenta en la programación del software, para el diseño en PSIM se ha ajustado mediante la constante de unidades en el sensor.

Con todos estos ajustes, en la *Figura 26* se muestra el diagrama de bloques sobre el que se va a trabajar.

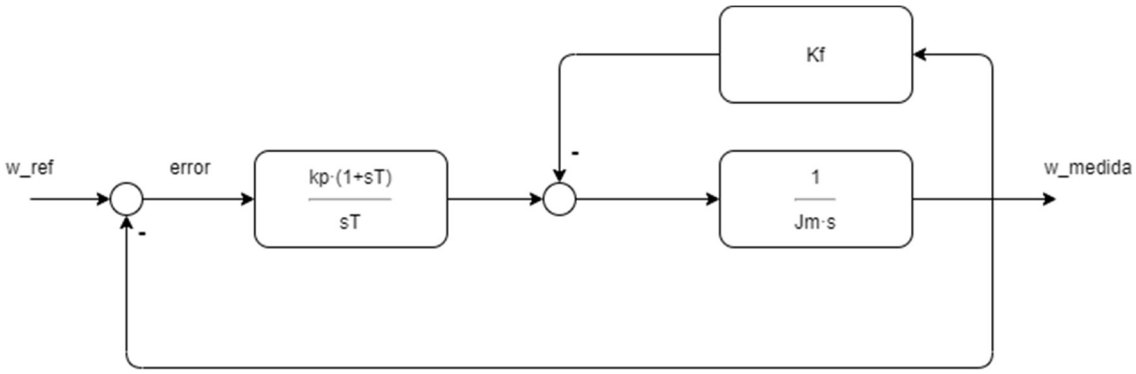

Figura 26. Diagrama bloques control-funciones

A continuación se va a simplificar el diagrama resolviendo el lazo de realimentación del par resistente. Se muestran a continuación la simplificación y el diagrama de bloques simplificado en la *Figura 27.*

$$P(s) = \frac{\frac{1}{J_m \cdot s} \cdot k_f}{1 + \frac{1}{J_m \cdot s} \cdot k_f} = \frac{1}{J_m \cdot s + k_f}$$

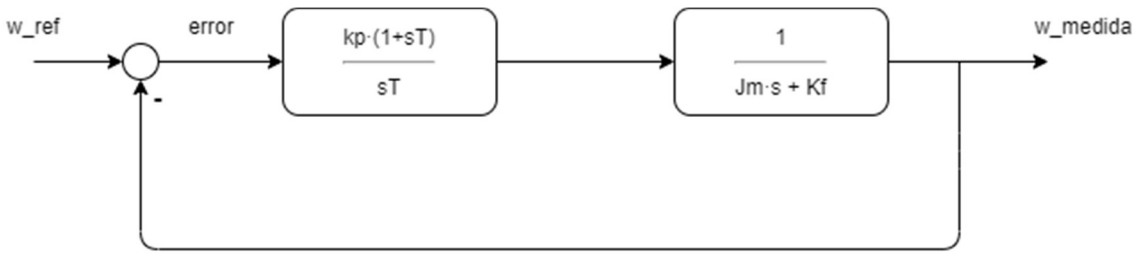

Figura 27. Diagrama bloques control-simplificación

Los únicos valores que quedan por definir y que servirán para ajustar el controlador son la constante proporcional (kp) y la constante de tiempo (T) del controlador PI. Los demás valores están predefinidos por el motor y por el tipo de carga:

- Inercia del sistema: $\mathbf{J_m}$ = **0,4 kg·m²**
- Constante de la carga: $\mathbf{K_f}$ **= 2,16**

La función de transferencia del controlador PI puede expresarse en función de la constante proporcional kp y la constante de integración ki. En la siguiente ecuación se muestra la nueva función de transferencia, así como su equivalencia con la constante de tiempo.

$$C(s) = k_p \cdot \frac{1 + sT}{sT} = \frac{k_p s + k_i}{s}; \qquad T = \frac{k_p}{k_i} \tag{21}$$

A partir del diagrama de bloques de la *Figura 27* se obtiene la función de transferencia en lazo abierto del sistema con la que se trabajará para conseguir un margen de fase y una frecuencia de corte deseados analizando el comportamiento en el dominio de la frecuencia. [3]

$$FT_{LA}(s) = \frac{k_p(1 + sT)}{sT} \cdot \frac{1}{J_m s + k_f}$$

El margen de fase está relacionado con la estabilidad del sistema, además cuanto más pequeño es el margen de fase, más rápido es el sistema, pero menos amortiguado. Son valores típicos de margen de fase para este tipo de operación comprendidos entre 40º y 60º. Para calcular el margen de fase nos queda la siguiente expresión:

$$MF = 180º + arctg(T \cdot w_c) \; - \; 90º \; - \; arctg(J_m w_c)$$

Una de las opciones más utilizadas consiste en colocar el cero del controlador PI una década antes de w_c. Con esta imposición el cero aporta exactamente 84,29º a la fase.

$$T \cdot w_c = 10 \; \rightarrow \; \text{arctg}(10) = 84.29º$$

$$MF = 180º + 84.29º - 90º - arctg(J_m/kf \cdot w_c) \rightarrow$$

$$\rightarrow MF = 174.29 - arctg(J_m/kf \cdot w_c) \rightarrow$$

$$\rightarrow MF = 174.29 - arctg(0.185 \cdot w_c)$$

Para obtener un margen de fase de 60º w_c debe ser igual a 12, con lo que el valor de **T = 0,83**. Con esto se procede a calcular kp mediante la siguiente expresión:

$$\frac{k_p \sqrt{1 + (w_c T)^2}}{w_c \cdot T} \cdot \frac{1}{\sqrt{(J_m w_c)^2 + {k_f}^2}} = 1$$

Resolviendo el sistema se obtiene un valor de kp = 5.24.Con todos los parámetros calculados, se introduce el sistema completo, incluyendo la etapa de control en lazo cerrado en PSIM para proceder a la simulación. En la *Figura 28* se puede observar el diagrama completo del lazo cerrado.

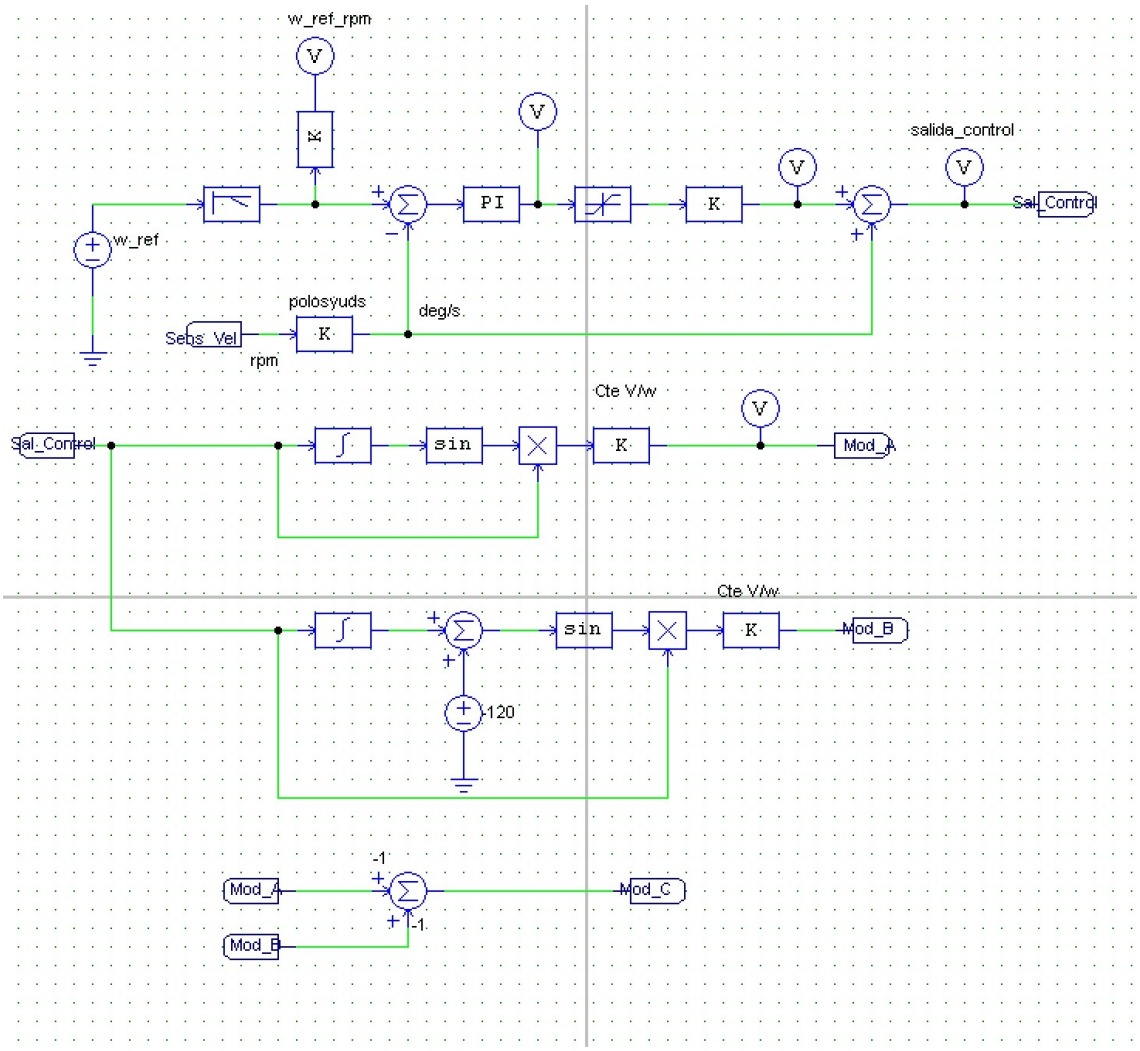

Figura 28. Esquema lazo cerrado en PSIM

El siguiente paso es comprobar mediante la simulación del sistema el efecto del sistema en lazo cerrado. Para ello en la *Figura 29* se muestran en una gráfica los resultados tanto de velocidad de referencia como las medidas por el sensor de velocidad de varias simulaciones en PSIM para las

distintas las distintas referencias de 10, 20, 30, 40 y 50 Hz. El eje Y de velocidad se mide en rpm mecánicas.

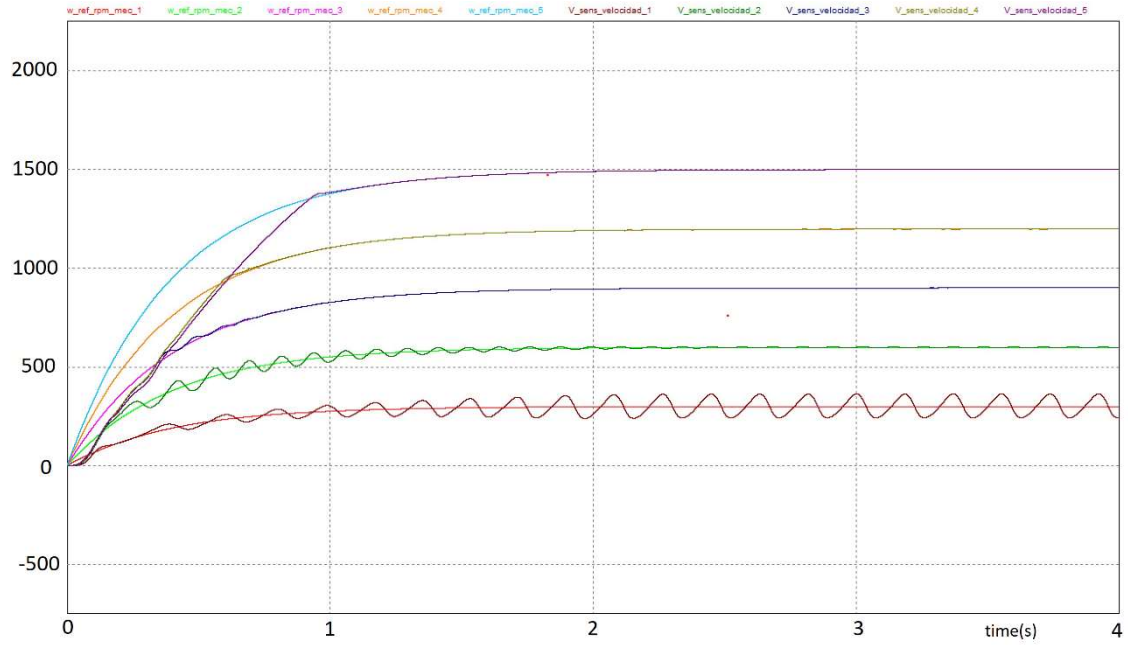

Figura 29. . Simulación velocidad real y referencia lazo cerrado en PSIM

Se puede observar como en el lazo cerrado la velocidad real alcanza la velocidad de referencia en todos los casos. Cabe mencionar que el sistema no se comporta correctamente en bajas velocidades. Como ya se ha mencionado en el apartado de lazo abierto, este mal funcionamiento es debido a que las resistencias del motor cobran importancia a bajas frecuencias y no se consideran en la relación V/f=cte.

3.5 Control digital

En la gran mayoría de aplicaciones de electrónica, las estrategias de control se implementan digitalmente y no analógicamente, como se ha calculado hasta ahora. Esto se debe a que el control digital presenta un gran número de ventajas como por ejemplo la flexibilidad de modificar el control, cambiar los parámetros del regulador y permite realizar una gran cantidad de operaciones lógicas. [4]

Al incluir el control digital, el sistema ahora incluye una parte analógica (se analiza en el dominio de Laplace) y el controlador digital (se analiza en

el dominio Z). Para analizar y trabajar con estos sistemas se puede proceder de dos maneras distintas:

- Convertir la parte analógica al dominio Z y calcular el controlador directamente con la teoría de control digital.
- Convertir el controlador al dominio de Laplace y calcularlo. Una vez calculado se transforma de nuevo a Z para implementarlo.

En este proyecto se va a emplear el segundo método ya que es el más utilizado en este tipo de aplicaciones. Para tener en cuenta los efectos del control digital, se van a incluir en el lazo de control una serie de términos que modelizan estos efectos:

- Digitalización: este debido al tiempo de cálculo del procesador se añade un retraso de un periodo de muestreo (T_m).

$$R(s) = e^{-T_m \cdot s} \tag{22}$$

- Muestreo: para cada periodo de muestreo se toma únicamente un valor para el control.

$$M(s) = \frac{1}{T_m} \tag{23}$$

- Retención: el valor tomado en cada periodo de muestreo permanece constante. El retenedor de orden cero (ZOH) modeliza este efecto.

$$ZOH(s) = \frac{1 - e^{-T_m \cdot s}}{s} \tag{24}$$

Donde para esta aplicación el tiempo de muestreo es **T_m=0,0002 s**, este periodo coincide con el periodo de conmutación de los transistores. Para cada periodo de conmutación el microprocesador aplicará el lazo de control para calcular las modulantes que se aplicarán en el siguiente periodo de

conmutación. Es por esto que los datos se tienen que tomar en este tiempo. En la *Figura 30* se puede observar este efecto.

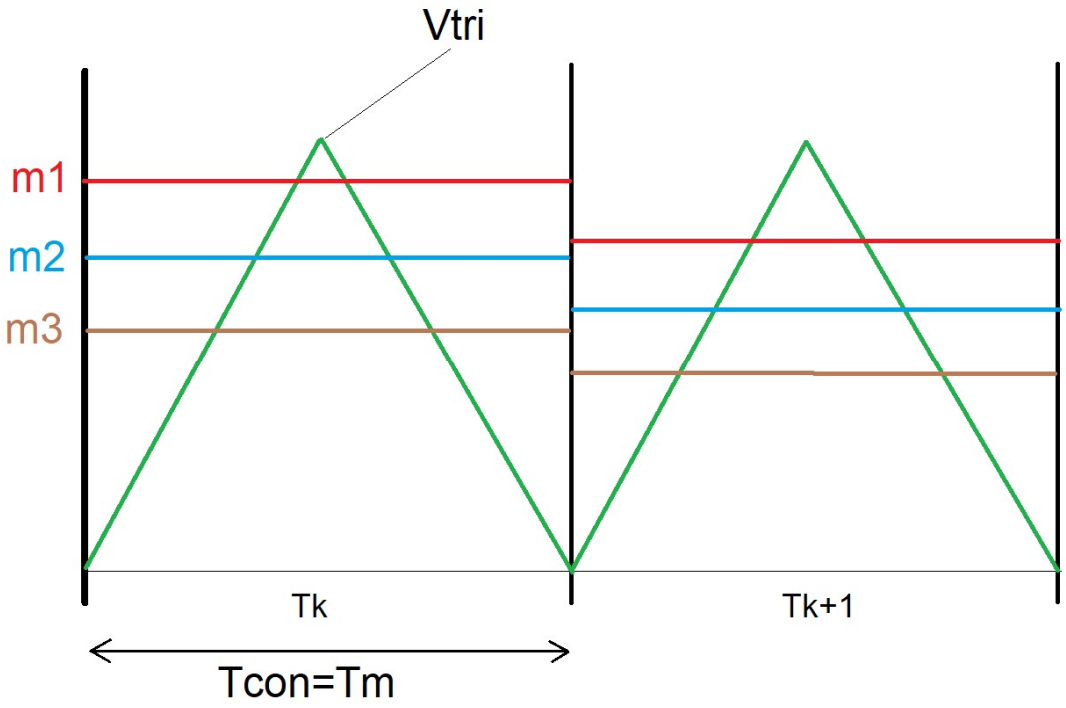

Figura 30. Periodo conmutación y modulantes

Para proceder con el cálculo del controlador, en la *Figura 31* se han incluido los elementos debidos a la digitalización en el lazo de control.

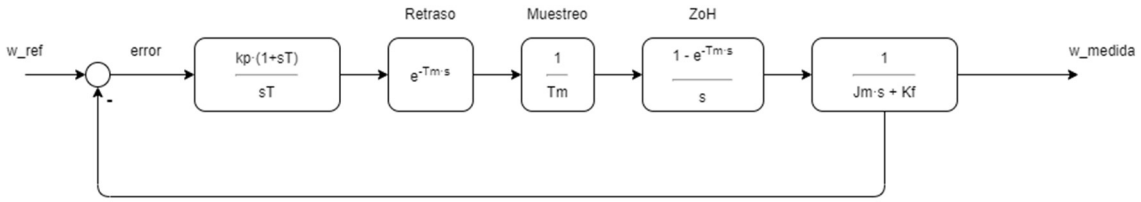

Figura 31. Diagrama de bloques lazo control digital

Para simplificar los cálculos en el lazo de control se va a emplear una aproximación que consiste en modelizar un retraso de $1{,}5 \cdot T_m$ por un primer orden. Debido a esta la función de transferencia es menos precisa, pero sigue siendo válida para frecuencias menores que $T_m/20$, en este proyecto no supone ningún problema. En este sistema la frecuencia de muestreo será la frecuencia de conmutación de los transistores: 5 kHz.

$$e^{-T_m \cdot s} \cdot \frac{1}{T_m} \cdot \frac{1 - e^{-T_m \cdot s}}{s} \approx \frac{1}{(1{,}5 \cdot T_m \cdot s + 1)} \quad (25)$$

Con esta aproximación el diagrama de bloques queda bastante simplificado. En la *Figura 32* se muestra el diagrama con el que se va a trabajar.

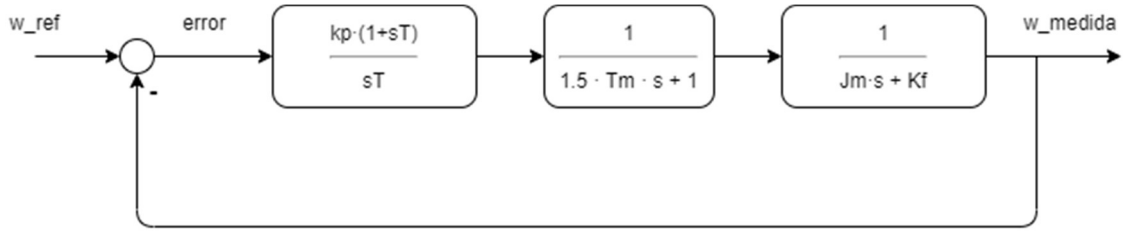

Figura 32.Diagrama de bloques lazo cerrado digital-simplificado

A partir del diagrama de bloques obtiene la función de transferencia en lazo abierto del sistema. A partir de la función en lazo abierto se trabajará para conseguir un margen de fase y una frecuencia de corte deseados analizando en el dominio de la frecuencia.

$$FT_{LA}(s) = \frac{k_p(1 + sT)}{sT} \cdot \frac{1}{1.5 \cdot T_m \cdot s + 1} \cdot \frac{1}{J_m s + k_f}$$

El margen de fase está relacionado con la estabilidad del sistema, además cuanto más pequeño es el margen de fase, más rápido es el sistema, pero menos amortiguado. Se suele trabajar con un margen de fase de entre 40º y 60º. Para calcular el margen de fase nos queda la siguiente expresión:

$$MF = 180º + arctg(T \cdot w_c) - 90º - arctg(1.5 \cdot Tm \cdot w_c) - arctg(J_m w_c)$$

Una de las opciones más utilizadas consiste en colocar el cero del controlador PI una década antes de w_c. Con esta imposición el cero aporta exactamente 84,29º a la fase.

$$T \cdot w_c = 10 \rightarrow \text{arctg}(10) = 84.29º$$

$$MF = 180º + 84.29º - 90º - arctg(1.5 \cdot Tm \cdot w_c) - arctg(J_m/kf \cdot w_c) \rightarrow$$

$$\rightarrow MF = 174.29 - arctg(1.5 \cdot Tm \cdot w_c) - arctg(J_m/kf \cdot w_c) \rightarrow$$

$$\rightarrow MF = 174.29 - arctg(0.0003 \cdot w_c) - arctg(0.185 \cdot w_c)$$

Para obtener un margen de fase de 60º w_c debe ser igual a 1520, con lo que el valor de **T = 0,0066**. Con esto se procede a calcular kp mediante la siguiente expresión:

$$\frac{k_p \sqrt{1 + (w_c T)^2}}{w_c \cdot T} \cdot \frac{1}{\sqrt{(1.5 \cdot T_m \cdot w_c)^2 + 1}} \cdot \frac{1}{\sqrt{(J_m w_c)^2 + {k_f}^2}} = 1$$

Resolviendo la ecuación se obtiene un valor de kp = 30,8. Con esto ya se han calculado los parámetros del controlador PI en el dominio de Laplace, solo falta convertirlos al dominio Z mediante la discretización del mismo. Dado que la transformación $z = e^{s \cdot Tm}$ da lugar a expresiones irracionales, se emplean las aproximaciones de Euler (forward y backward) y de Tustin. Se va a emplear por ser la más utilizada la fórmula de Euler forward en la cual se sustituye **s** por la siguiente expresión:

$$s \rightarrow \frac{z - 1}{T_m} \qquad (26)$$

$$C(z) = k_p \frac{\left(1 + T \cdot \frac{z - 1}{T_m}\right)}{T \cdot \frac{z - 1}{T_m}} = k_p \frac{\left({T_m}/{T} + 1\right) - z}{1 - z}$$

Para realizar la acción del controlador PI es necesario calcular la integral del error pero digitalmente esto no es posible, por lo que habrá que adaptar la ecuación del PI para que la realice el micro controlador. La acción del controlador analógico PI se obtiene de la siguiente manera:

$$u(t) = k_p \cdot e(t) + k_i \int e(t) \tag{27}$$

Para implementar esta ecuación en el sistema digital se va aproximar mediante rectángulos, como puede verse en la FIGURA. De esta manera el microcontrolador si que es capaz de interpretar los datos, ya que todos ellos son discretos.

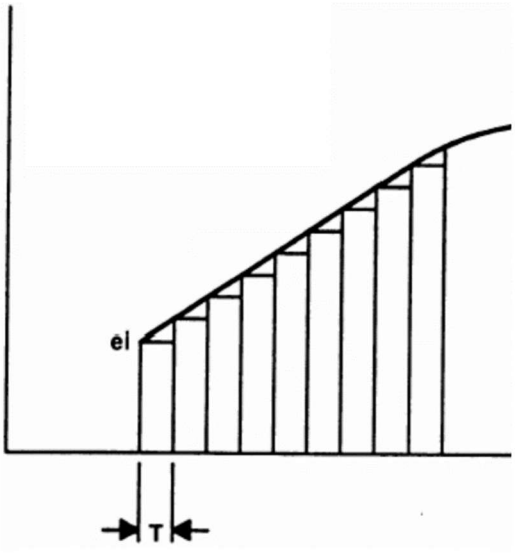

Figura 33. Aproximación rectangular del PI

Donde T es el periodo de muestreo. Con esta aproximación los términos proporcional e integral del controlador quedan de la siguiente manera:

$$k_p \cdot e(t) \rightarrow k_p \cdot e(n) \tag{28}$$

$$k_i \cdot \int e(t) \;\; \rightarrow \;\; k_i \cdot T \cdot \sum_i e_i \tag{29}$$

De esta manera queda definido por completo el controlador que se implementara digitalmente en un apartado posterior.

3.6 Compensación R·I

Para solucionar el problema del control en bajas velocidades que se ha podido observar y que ya se había mencionado anteriormente, se ha añadido al sistema una compensación RI.

El objetivo de esta compensación es el de añadir un pequeño voltaje extra en las velocidades más bajas para así evitar que todo el voltaje sea consumido por la resistencia del estator. La idea de lo que se pretende conseguir queda reflejada en la *Figura 34.*

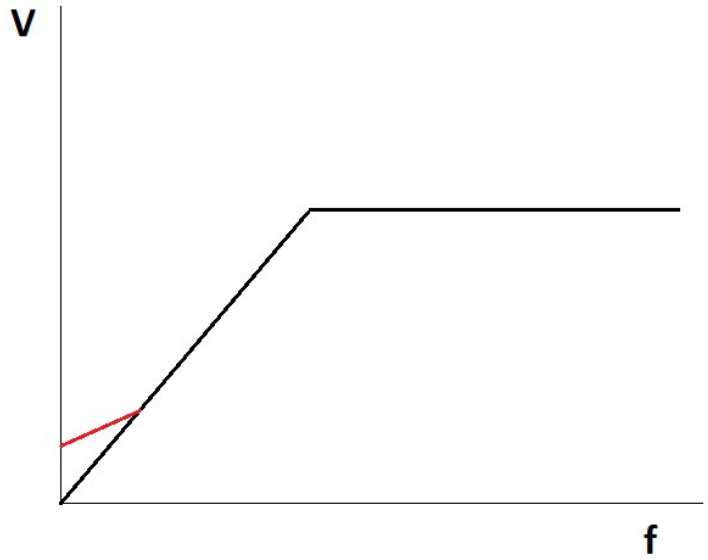

Figura 34. Efecto compensación R·I

Se ha establecido que solamente se quiere realizar la compensación para velocidades menores a 10 Hz, ya que es donde se ha observado el mal funcionamiento, y que la máxima ayuda sea de un 7,5% del máximo.

Para lograr este objetivo, se toma una lectura del sensor de velocidad del motor y es restado a un valor constante de **10**. La señal obtenida se multiplica por una constante de valor **0,0075** (de esta manera para una velocidad de 0 se ejercería una ayuda de 0,075). Para evitar posibles errores, se ha colocado un **limitador entre 0 y 0,075.** La *Figura 35* representa el diagrama de bloques de este pequeño sistema.

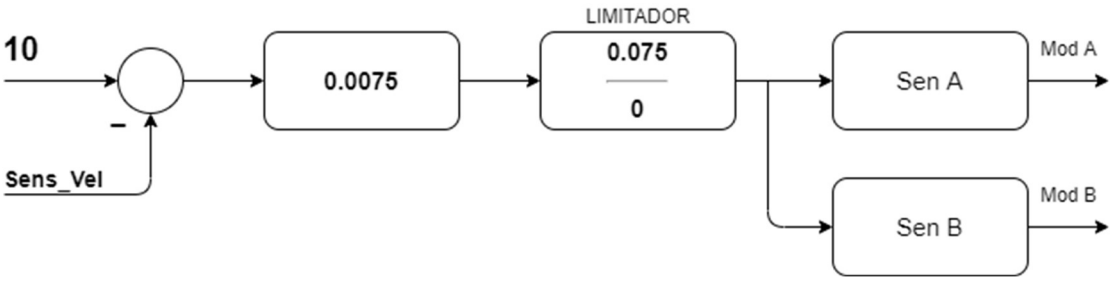

Figura 35. Diagrama bloques compensación R·I

De esta manera se asegura que la ayuda solo ocurrirá para velocidades de entre 0 y 10 Hz. En la *Figura 36* queda reflejada la ayuda que se aplicaría para cada caso.

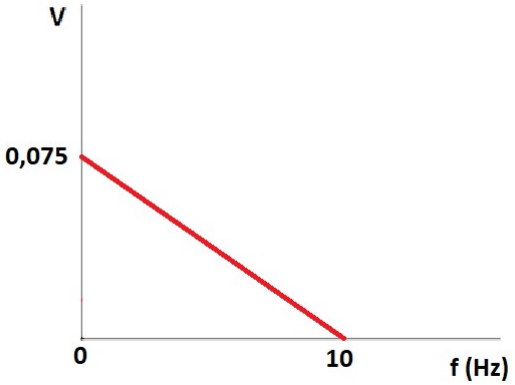

Figura 36. Voltaje adicional R·I

El último paso es añadir este valor a la señal moduladora de la PWM para que llegue al motor. Para lograr esto se multiplica el seno de cada fase por el valor obtenido, y se suma a la señal moduladora senoidal. En se muestra como se ha incluido en el sistema.

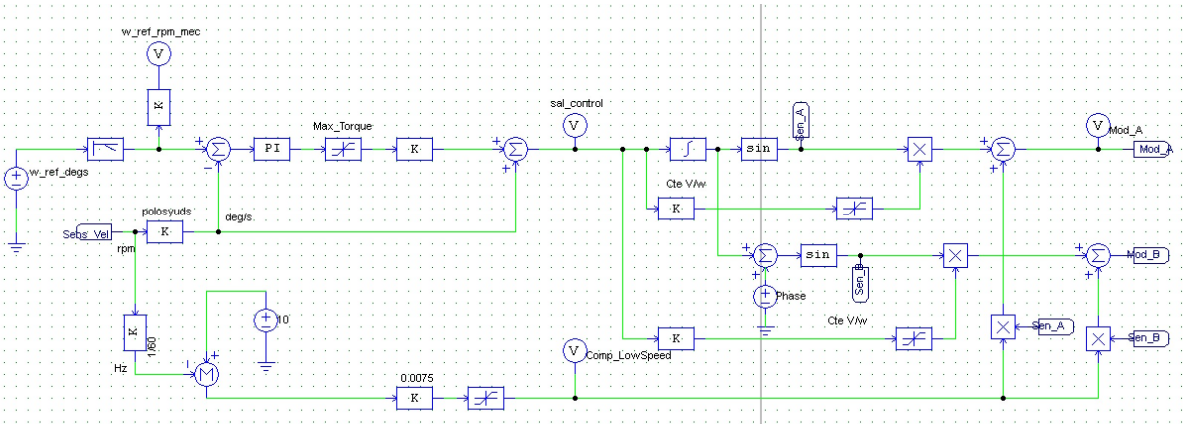

Figura 37. Esquema compensación R·I en PSIM

Con esta pequeña ayuda que se ha incluido se va a poder observar como el motor se comporta mejor en el arranque, y cuando las velocidades son bajas. En la figura tal se muestra muestran en una gráfica los resultados tanto de velocidad de referencia como las medidas por el sensor de velocidad de varias simulaciones en PSIM para las distintas las distintas referencias de 10, 20, 30, 40 y 50 Hz. El eje Y de velocidad se mide en rpm mecánicas.

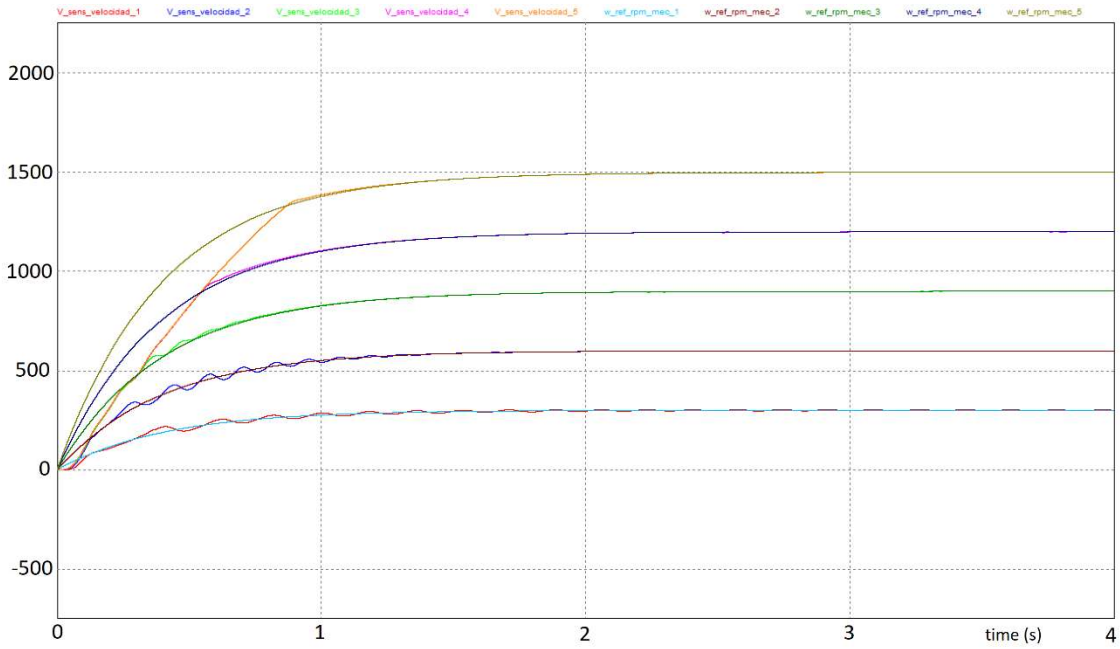

Figura 38.Simulación velocidad real y referencia con compensación en PSIM

A diferencia del caso anterior, sin incluir la compensación, se puede observar como el sistema se adapta mucho mejor a velocidades de referencia bajas, así como muestra una respuesta mas suave a las velocidades iniciales. Consigue adaptarse perfectamente a cada una de las referencias.

4. DISEÑO PCB

En esta segunda parte del proyecto se va a diseñar y dejar lista para fabricación una tarjeta electrónica que sea capaz de incluir todos los elementos mencionados en el apartado anterior. El hardware necesario para implementar los algoritmos de control, así como el convertidor de potencia se describirá en este apartado.

El esquema general que se va a seguir en la tarjeta se muestra en la *Figura 39*, así como las conexiones entre los diferentes componentes que la forman.

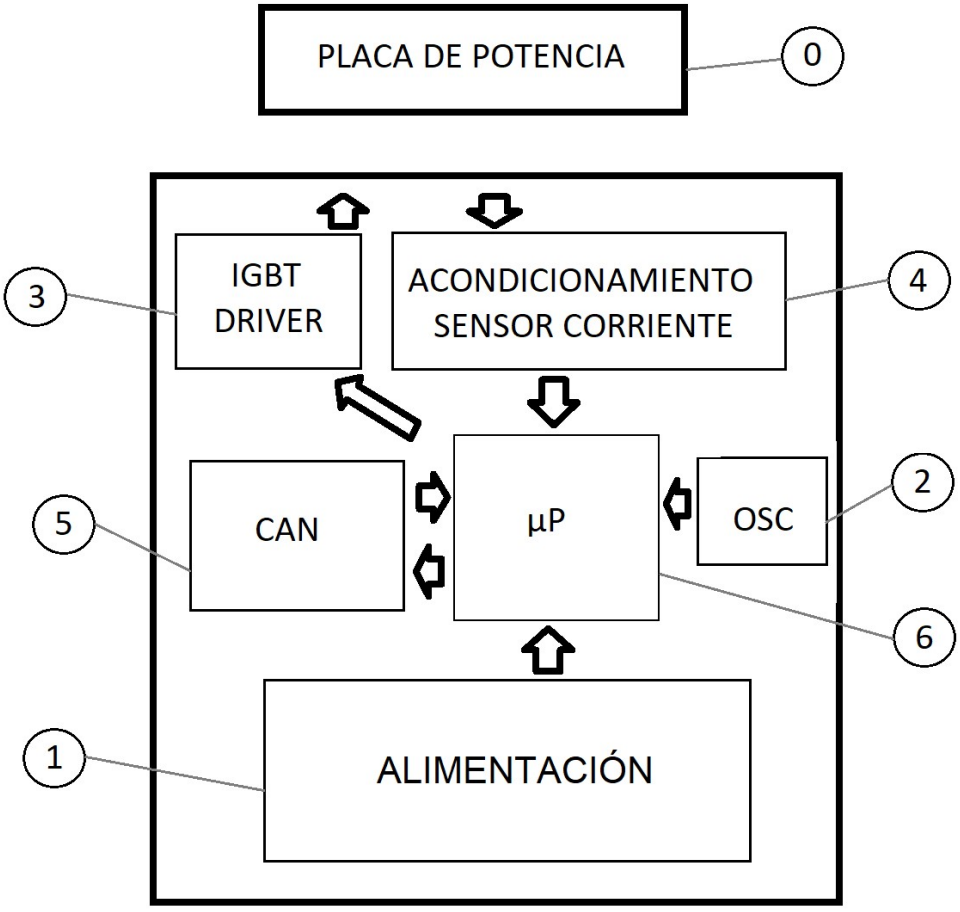

Figura 39. Esquema general de tarjeta

Se han enumerado los componentes que formaran parte de la tarjeta electrónica:

0. **Placa de potencia**: Esta placa viene ya determinada de la empresa e incluye los módulos de IGBTs, el sensor de velocidad e intensidad y el motor.
1. **Alimentación de la placa**: Este bloque incorpora la alimentación de la placa y la conversión DC/DC para alimentar el micro y demás componentes.
2. **Oscilador**: Genera la señal de reloj para sincronizar el micro.
3. **IGBT Driver:** Acondicionamiento de la señal proveniente del micro para gobernar los transistores.
4. **Acondicionamiento sensor de corriente**: Circuito necesario para preparar la señal del sensor de corriente.
5. **Can**: Módulo de interconexión entre el operario de la placa y el micro.
6. **Microprocesador**: Elemento principal de la placa. Encargado de analizar y generar señales.

Para realizar esta parte del proyecto se ha utilizado el software EAGLE de Autodesk. En este programa se trabaja mediante un esquemático en el que se insertan los componentes reales que se quiere en la tarjeta, así como todas las conexiones necesarias.

Una vez se ha terminado el esquemático, se procede a trabajar sobre lo que será la tarjeta. En esta parte del programa se colocan todos los elementos en la posición que tendrán en la tarjeta. Una vez esta todo en su sitio se procede a trazar las pistas de la PCB en la capa más adecuada.

A continuación se van a analizar los diferentes componentes que se han enumerado anteriormente. En los anexos del documento se incluyen las hojas de datos de todos los elementos que se detallarán a continuación así como de donde se han obtenido las imágenes de los componentes.

4.1 Placa de potencia

La placa de potencia que viene impuesta por la empresa y cuenta con el sensor de corriente, el sensor de velocidad y el módulo de IGBTs.

Como parte principal del inversor PWM se emplea un puente de IGBTs para generar las ondas sinusoidal que gobiernan al motor. Para realizar esta

función se emplean los módulos **SKM100GB12T4**. Estos módulos se emplean normalmente en convertidores de AC, por lo que mediante el empleo de 3 módulos se conseguiría controlar cada una de las 3 fases del motor.

Según la ficha técnica, que se incluirá en los anexos, el máximo voltaje para $\mathbf{V_{CES}}$ es de **1200 V**, un superior al de +- 400 V que se emplea como fuente de continua. El máximo para $\mathbf{I_C}$ es de **160 A**, siendo su valor nominal de $\mathbf{I_{Cnom}}$**= 100 A**, valores que concuerdan con los simulados para esta aplicación.

Por otro lado, el rango de voltaje para $\mathbf{V_{GES}}$ es de **-20..20 V**, lo que da un amplio margen para el gobierno de estos módulos IGBTs. Además el tiempo de conmutación para valores nominales es de 10 µs, lo que es compatible con la frecuencia de conmutación de 5 kHz del PWM.

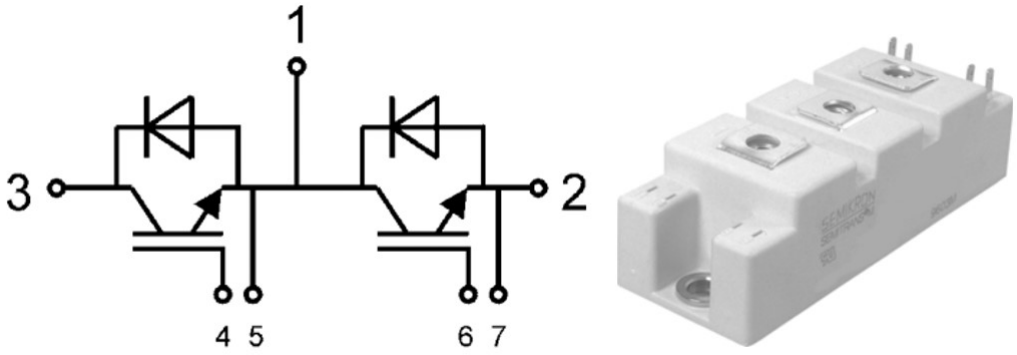

Figura 40. Módulo IGBTs SKM100GB12T4

Como sensor de velocidad en la placa se incluye un encoder incremental de 128 ranuras. Este dispositivo proporciona dos salidas iguales pero desfasadas cuyo valor es un pulso cada vez que se alcanza una de las ranuras.

El sensor de corriente consiste en una resistencia Rshunt que mide el paso de la corriente a través suya. Midiendo el voltaje antes y después de esta resistencia se obtiene la medida de corriente.

4.2 Alimentación

La alimentación de la placa se realiza mediante un conector externo de 12 V. Estos 12 V se convierten mediante un DC/DC a 3,3 V para alimentar

el microcontrolador y el resto de los elementos. El esquema de la alimentación se muestra en la *Figura 41*. [5]

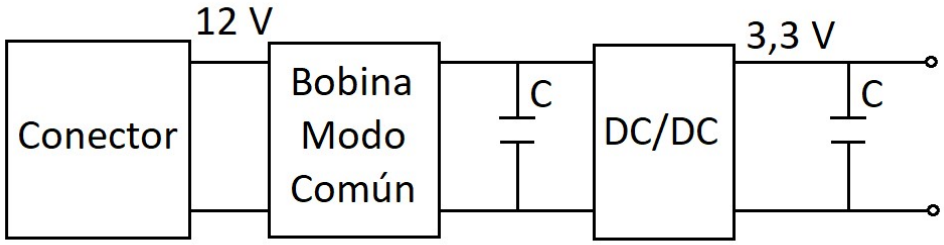

Figura 41. Esquema de alimentación de la placa

A continuación se van a detallar cada uno de los elementos necesarios en la alimentación de la placa.

- Jack alimentación**:**

Para alimentar la placa electrónica se utiliza una fuente de tensión DC de 12V. Esta tensión se introducirá en la tarjeta a través de un conector **JACK CON-SOCJ-2155 de 2.1mm**.

- Bobina modo común:

Un problema que surge con la fuente de alimentación es que pueden introducir interferencias electromagnéticas (EMI) en modo común, generando excesivas corrientes por la tierra. Esto puede ser un grave problema ya que estas corrientes pueden interferir con las altas frecuencias de la entrada.

Para evitar estos problemas se introduce una bobina de choque de modo común. Estos dispositivos presentan una alta impedancia en serie a las corrientes de interferencia en modo común en el circuito y esta impedancia no afecta a la señal funcional diferencial.

Para este proyecto se ha elegido la bobina **SC-10-30J** con una inductancia de 3 mH y capaz de soportar corrientes de hasta 10 A.

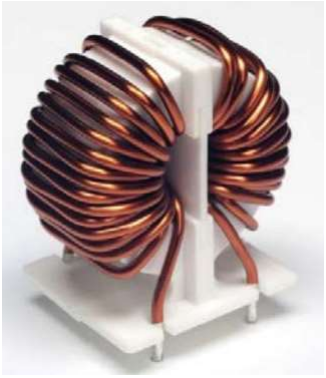

Figura 42. Bobina SC-10-30J

- Condensadores de desacoplo:

Otro problema que surge en la tensión de alimentación es que se introducen pequeñas componentes de AC que se superponen a la señal DC. Para tratar de eliminar este problema se introduce un condensador de desacoplo.

Las señales de AC son llevadas a tierra a través de este condensador, mientras que para las señales DC el condensador tiene prácticamente reactancia infinita. Este condensador se coloca en paralelo a la fuente de alimentación y su efecto ideal teórico se muestra en la *Figura 43.*

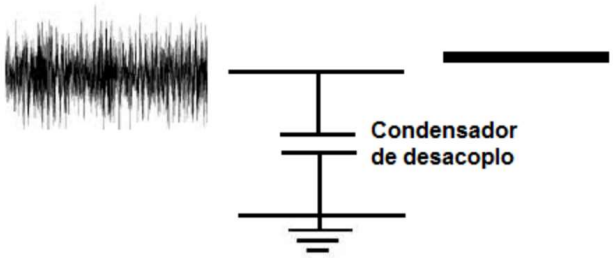

Figura 43. Efecto condensador de desacoplo

Para este proyecto se van a emplear 3 **condensadores electrolíticos de 470 µF** y hasta 25 V. Dos de ellos se colocan a continuación de la bobina mencionada anteriormente (entre 12 V y DGND) y el tercero de ellos para la salida (entre 3,3 V y DGND). Adicionalmente se ha incluido un **banco** de 6 **condensadores** cerámicos de 100 nF para el desacoplo del microcontrolador y el cristal.

- Convertidor DC/DC

Se han obtenido 12 V de la alimentación, pero el microcontrolador y gran parte de la placa se alimentan a 3,3 V. Para ello se ha incluido un **TRACO-TEL5-1210.** Este dispositivo es un convertidor DC/DC que permite una entrada entre 8 y 12 V y proporciona en la salida 3,3 V de una manera muy fiable. Además proporciona aislamiento entre entrada y salida y eficiencias de hasta el 86 %.

-

Figura 44. Convertidor DC/DC TRACO-TEL5-1210

En la *Figura 45* se muestran el diagrama de la realizado en EAGLE.

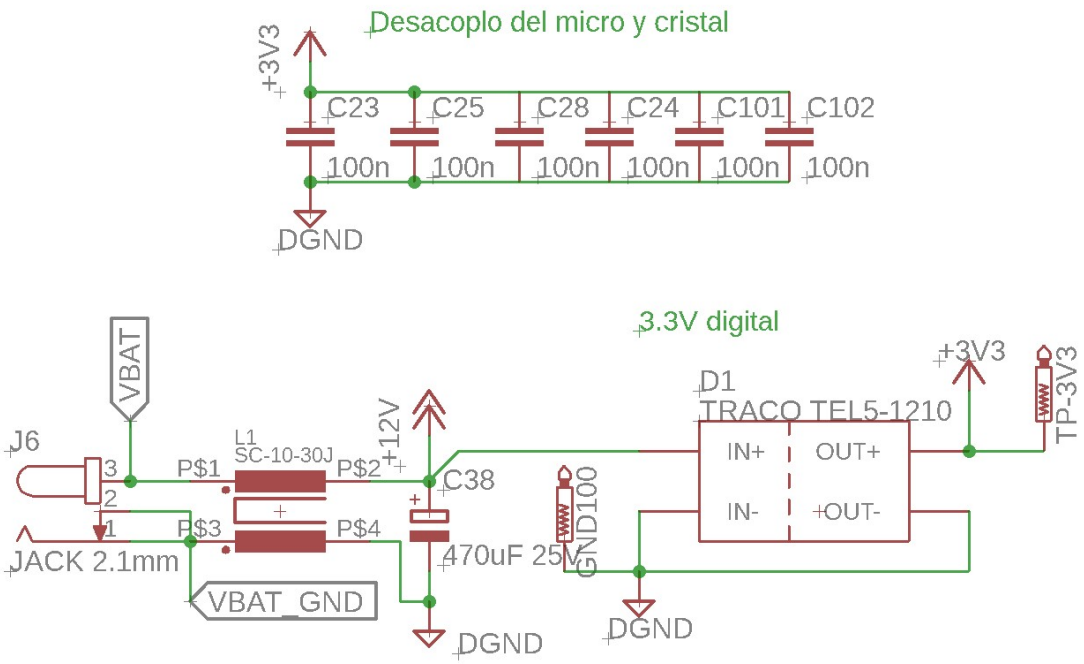

Figura 45. Alimentación y desacoplo de la placa en EAGLE

4.3 Oscilador

Los microcontroladores necesitan una señal externa para sincronizarse y de esta manera realizar sus tareas correctamente. Para generar una señal de reloj estable para el microcontrolador se va a emplear el oscilador externo **Q33519E40003912** de **10 MHz**. Se trata de un oscilador de cristal de pequeñas dimensiones y una estabilidad en la frecuencia de salida de +-50 ppm.

Figura 46. Oscilador externo de 10 MHz Q33519E40003912

4.4 IGBT Driver

La señal generada por el microcontrolador para gobernar el encendido y apagado de los IGBTs se tiene que acondicionar correctamente antes de conectarse a los transistores. En la *Figura 47* se muestra el esquema de conexión de este acondicionamiento.

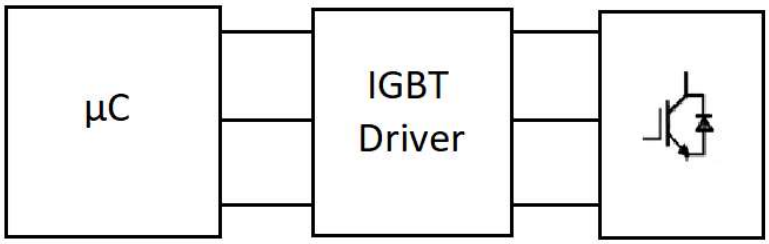

Figura 47. Esquema conexionado IGBT Driver

Para gobernar el encendido y el apagado de los transistores IGBTs se ha empleado el UCC21540. Este dispositivo es un controlador de doble canal con aislamiento que se emplea para controlar transistores de potencia.

Figura 48. IGBT Driver UCC21540

Entre sus principales aplicaciones se encuentra el gobernar los accionamientos de un motor. Por este motivo y por su combinación de aislamiento y controlador, hacen perfecto a este dispositivo para esta aplicación. [6]

En las condiciones recomendadas de operación se establecen los voltajes de alimentación para la entrada, VCCI, entre 3 y 5.5 V y para la salida, VDDA y VDDB, entre 6 y 18 V. Para esta implementación se han escogido para VCCI un voltaje de 3.3 V y para VDD un voltaje de 15 V. De esta manera se consigue disminuir el número de fuentes de tensión al usar el mismo voltaje que el microcontrolador en la entrada.

Para conseguir una fuente de tensión de 15 V se va a utilizar el convertidor DC-DC **TMR 6-1213**. Este convertidor asegura un aislamiento entre la entrada y la salida, además de una protección continua ante cortocircuitos. Por estos motivos y por su reducido tamaño se ha escogido para proporcionar una fuente de tensión continua y fiable.

Figura 49. Convertidor DC-DC TMR 6-1213

En la *Figura 50* se muestra el esquema de funcionamiento por bloques que viene indicado en el manual de usuario del UCC21540.

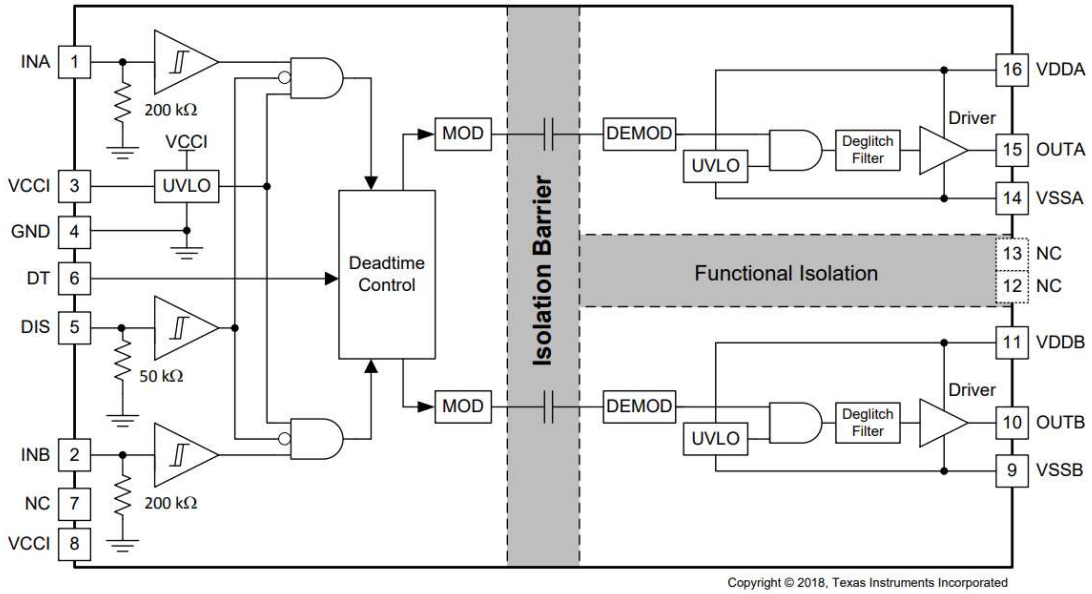

Figura 50. Esquema interno funcionamiento UCC21540

En la etapa de entrada, los inputs INA, INB y DIS están totalmente aislados del voltaje de alimentación VDD. Estos inputs se controlan fácilmente con señales de nivel lógico, el comparador de histéresis estabiliza el funcionamiento y sirve para inmunizar frente a posible ruido.

Por otro lado, si alguno de los inputs esta se deja abierto, las resistencias de pull-down fuerzan a estos pines en bajo. Cabe destacar que la amplitud de estas señales de entrada no puede superar el voltaje de alimentación de VCCI.

En la etapa de salida sigue una estructura pull-up que permite entregar la máxima corriente cuando es necesario. Para esto incluye un P-channel MOSFET junto a un N-channel MOSFET en paralelo. La función del transistor N-channel es la de entregar un aumento en el pico de corriente permitiendo un rápido encendido. Esto se consigue encendiendo brevemente el transistor cuando el estado de salida cambia de bajo a alto. Tal y como se muestra en la *Figura 51*, voltaje de salida cambia entre VSS y VSS, permitiendo un modo de operación rail-to-rail.

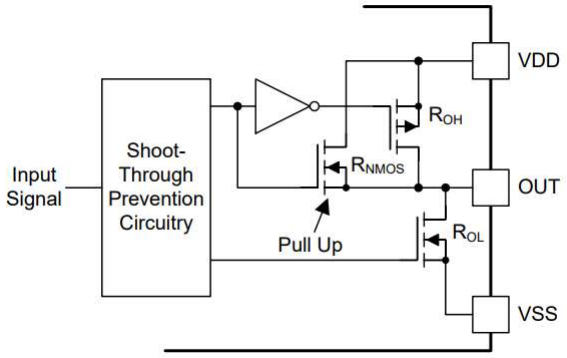

Figura 51. Etapa de salida del UCC21540

A continuación se va a detallar el proceso de diseño en la implementación de este dispositivo. Para el control de un motor de 3 fases, en la *Figura 52* se muestra el esquema de configuración típico para este dispositivo.

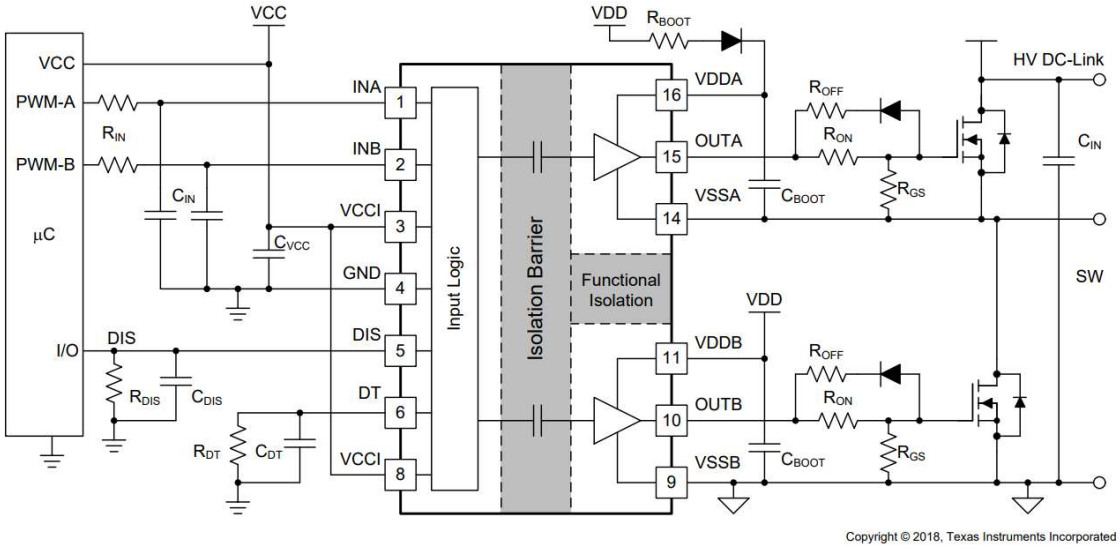

Figura 52. Esquema configuración típico UCC21540

Es posible emplear un pequeño filtro R_{IN} C_{IN} en la entrada para filtrar la señal de las interferencias introducidas por no idealidades o por pistas largas en el PCB. Sin embargo para esta aplicación no van a ser necesarios ya que se diseñará la placa en el menor espacio posible.

Para elegir un tiempo muerto de 10 ns se ha elegido una resistencia de 1 kΩ. Esta relación viene definida por la siguiente expresión:

$$t_{DT}(ns) = 10 \cdot R_{DT}(k\Omega) \tag{30}$$

Un diodo se carga por VDD a través de un diodo cada ciclo cuando el transistor de abajo conduce. Esta carga del condensador conlleva grandes picos de corriente, por lo que el diodo ha de ser capaz de disipar esa energía.

Para este diseño, el voltaje del bus DC es de +-400 V, por lo que se ha escogido el diodo **CGRM4001-G**, capaz de soportar hasta 1000 V.

Para reducir la corriente en el diodo se introduce una resistencia, R_{BOOT}. Se recomienda que el valor de esta resistencia se encuentre entre 1 y 20 Ω dependiendo del diodo utilizado. El caso más desfavorable de pico de corriente a través del diodo es el siguiente:

$$I_{DBOOT} = \frac{V_{DD} - V_{BDF}}{R_{BOOT}} = \frac{12V - 1.1}{11} = 1$$

Donde VDF es la caída de tensión en el diodo cuando está conduciendo, que según su ficha técnica es de 1,1 V para una intensidad de 1 A. Con una resistencia **R_{BOOT}** de **11 Ω** se obtiene el valor de 1 A.

En la salida del dispositivo se va a incluir una resistencia de encendido, R_{ON}. La función de esta resistencia es la de reducir cualquier ruido o interferencia que pueda introducirse, además de ajustar parámetros como la corriente, para así reducir perdidas en la conmutación.

Debido a la estructura pull-up de la etapa de salida, con un P-channel MOSFET y un N-channel adicional en paralelo. El pico de corriente en la fuente se puede estimar de la siguiente manera:

$$I_{OA} = \min\left(4, \frac{V_{DD} - V_{BDF}}{R_{NMOS} \| R_{OH} + R_{ON} + R_{GFET}}\right) \tag{31}$$

$$I_{OB} = \min\left(4, \frac{V_{DD}}{R_{NMOS} \| R_{OH} + R_{ON} + R_{GFET}}\right) \tag{32}$$

Donde todos los valores están detallados en la ficha técnica a excepción de R_{ON}, que es introducida por el usuario. El pico de corriente de la fuente será el mínimo valor entre 4 A y el valor que se calcula en la expresión anterior.

En esta ocasión se ha escogido un valor para $\mathbf{R_{ON}}$ de **2.2 Ω**. Con este valor las corrientes resultantes son las siguientes:

$$I_{OA} = \left(\frac{12\ V\ -\ 0.8\ V}{1.47\ \Omega \| 5\ \Omega + 2.2\ \Omega + 1.5\ \Omega}\right) =\ 3.1\ A$$

$$I_{OB} = \left(\frac{12\ V}{1.47\ \Omega \| 5\ \Omega + 2.2\ \Omega + 1.5\ \Omega}\right) = 3.27\ A$$

En esta aplicación no se ha incluido R_{OFF} ya que no es necesaria. Solamente con una resistencia R_{ON} de 2.2 Ω se consiguen unas corrientes menores al máximo de 4 A como establece la hoja de especificaciones de este dispositivo

El último paso para terminar el diseño del UCC21540 es el de escoger condensadores de bypass para VCCI, VDDA y VDDB. El condensador que se conecta en VCCI ha de ser capaz de entregar la corriente necesaria para la puerta lógica así como todo el consumo, que será solamente de unos pocos mA. Por esta razón se ha escogido un condensador **C-EUC0805** de **10 µF** para $\mathbf{C_{VCCI}}$. Si la fuente de tensión se encuentra a una gran distancia del pin VCCI, se deberá incluir un condensador de 1 µF en paralelo.

Como se ha mencionado anteriormente, un condensador C_{BOOT} se carga en cada ciclo. Este condensador en VDDA permite los picos de corriente necesarios, así como mantiene un voltaje estable para el transistor de potencia. La carga necesaria en cada ciclo se puede aproximar de la siguiente manera:

$$Q_{tot} = Q_G + \frac{I_{VDD}}{f_{SW}} = 0{,}154\ \mu\text{C}$$

Donde Q_G es la carga de la puerta del IGBT de potencia, I_{VDD} es el consumo de corriente a carga cero y f_{SW} es la frecuencia de conmutación. Por lo que el mínimo valor de C_{BOOT} para este requisito es el siguiente:

$$C_{BOOT} = \frac{Q_{tot}}{\Delta V_{VDDA}} = 0{,}23\ \mu\text{F}$$

Donde ΔV_{VDDA} es el rizado en VDDA. En la practica el valor de C_{BOOT} es mayor que el calculado. Con esto se permiten situaciones en las que de otra manera se saltarían pulsos debido al transitorio de carga.

Se ha de tener cuidado ya que C_{BOOT} no debe ser demasiado elevado. Esto podría causar que el condensador no se cargase en los primeros ciclos, con lo que los IGBTs no seguirían la señal de entrada correctamente. Se ha escogido para C_{BOOT} un condensador **C-EUC0805** de **10 µF.**

Con esto quedan definidos todos los componentes necesarios para la implementación de los controladores UCC21540 de los IGBTs. En la *Figura 53* se muestra el esquemático de este elemento, así como toda la instrumentación que se acaba de describir. Se emplean tres controladores, cada uno de ellos gobierna el encendido y apagado de uno de los módulos de IGBTs

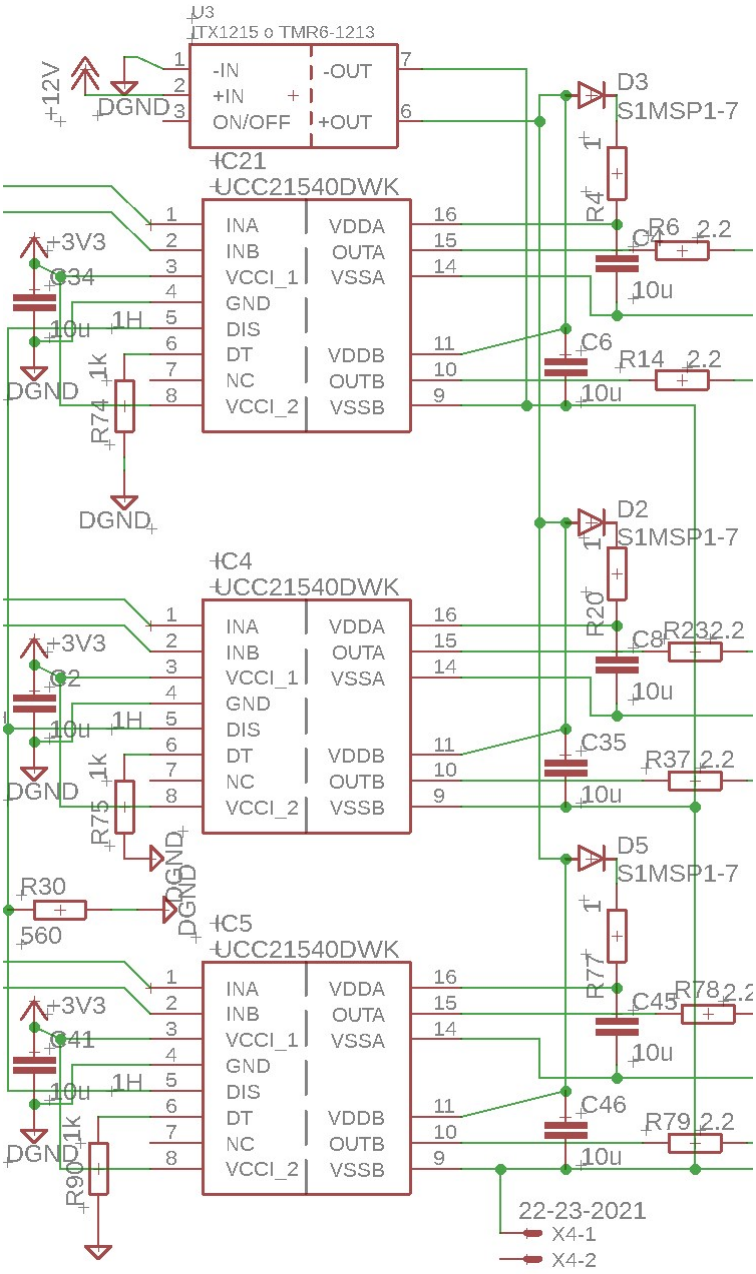

Figura 53. Diagrama componentes IGBT Driver en EAGLE

4.5 Acondicionamiento sensor de corriente

Como sistema de seguridad adicional del sistema se incluye un sensor de corriente. Con esta medida se podrá comprobar si las corrientes en el motor son adecuadas, o si por el contrario se ha producido algún fallo y las corrientes son excesivas. Lo que se pretende es que este sensor de corriente actúe como protector de sobre corrientes en el circuito, deteniéndolo si las corrientes son demasiado elevadas.

El sensor establecido para esta placa es una resistencia R_{shunt} capaz de medir hasta 200 A y cuyo voltaje está limitado a 20 mV, el cual se aumentará mediante una ganancia. El esquema de este sensor y el acondicionamiento se muestra en la *Figura 54*.

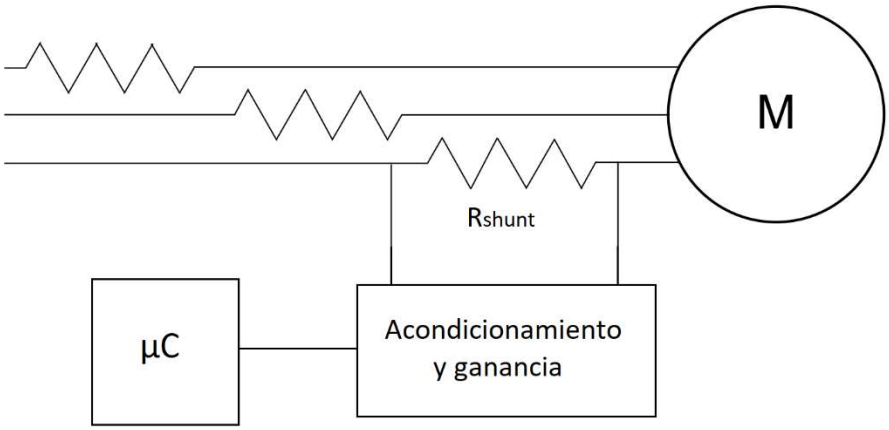

Figura 54. Esquema acondicionamiento sensor de corriente

Para este objetivo se ha escogido el dispositivo **Si8920**. Se trata de un amplificador analógico con aislamiento galvánico. Su entrada diferencial de bajo voltaje le permite medir sin problemas la caída de tensión a través de una resistencia de derivación. Es por esto, y por su aislamiento de hasta 6500 Vrms entre el sensor y el circuito de control que se ha elegido para este proyecto. En la *Figura 55* se muestra el esquema de bloques interno del Si8920, así como sus pines de entradas y salidas. [7]

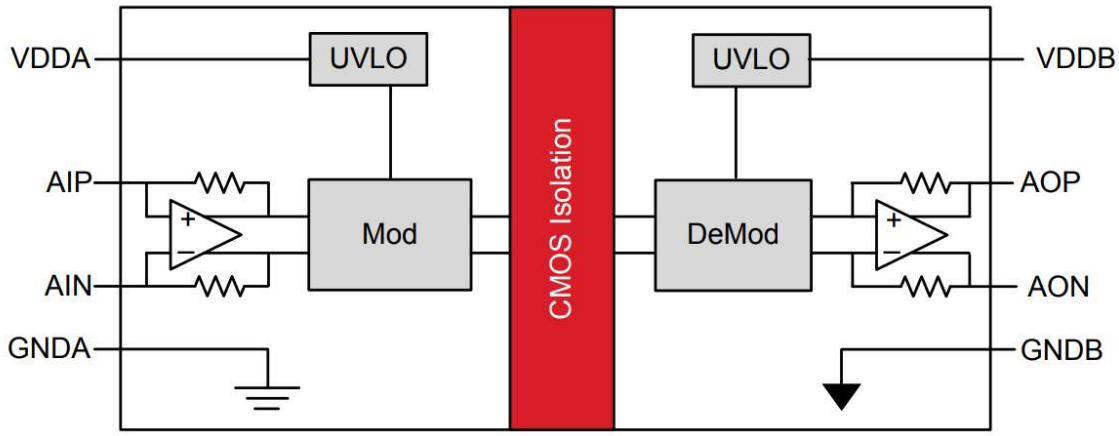

Figura 55. Diagrama bloques interno Si9020

Para poder introducir estas señales en el circuito del controlador es necesaria la implementación de un circuito de gobierno que adecue las señales.

En primer lugar son necesarias dos fuentes de tensión distintas para la entrada y la salida, ya que están aisladas. El rango de tensión para ambas se encuentra entre 3 y 5 V. En la salida, que se conecta con el circuito de control, se va a utilizar la misma fuente de 3,3 V que alimenta el microcontrolador.

Para alimentar la parte aislada se va a emplear un **IML0212D03**. Este dispositivo es un convertidor DC-DC que genera una tensión de **3,3 V** a la salida, además de garantizar aislamiento entre entrada y salida. Este dispositivo se alimentará de la fuente de alimentación de 12 V.

El voltaje en modo común de AIN y AIP se debe encontrar entre -0,2 y 1 V respecto a GNDA. Para conseguir este requisito, se va a conectar el pin AIP al pin COMMON del IML0212D03. Los pines -VOUT y +VOUT del convertidor se conectan respectivamente a VDDA y GNDA.

Para asegurar que la salidas no se sobrecargan, se recomienda emplear dos resistencias iguales y de valor mayor a 5 kΩ en las salidas AOP y AON. En el siguiente paso se detalla el valor de estas resistencias.

La etapa de salida se conecta al microcontrolador. Este será el encargado de decidir si las corrientes están dentro del rango de funcionamiento, o si estas son demasiado grandes debido a algún fallo.

El **Si8920** cuenta con una entrada diferencial, por lo que la etapa de salida necesita un acondicionamiento para entrar al microcontrolador. En la *Figura 56* se muestra la etapa intermedia que se ha diseñado.

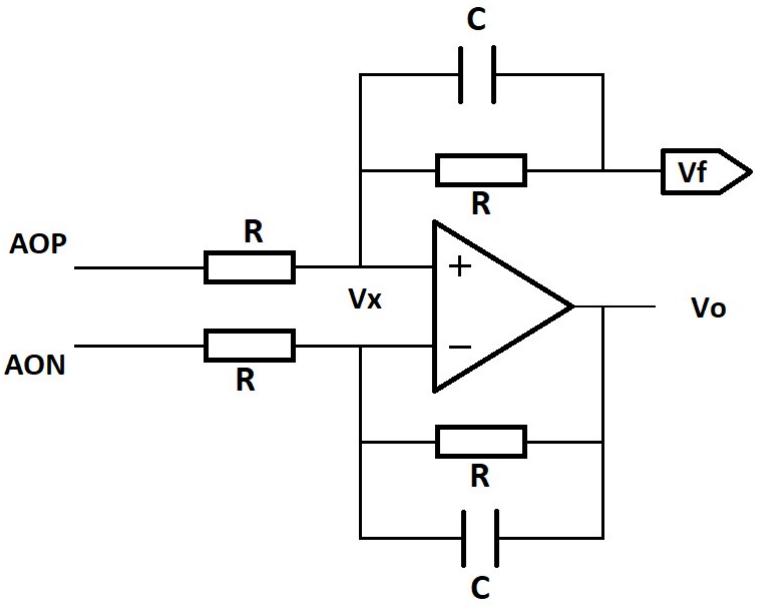

Figura 56. Circuito acondicionamiento sensor corriente

Mediante esta configuración se pretende aumentar la salida diferencial del Si8920 además de referenciarla respecto a un voltaje fijo, Vf. Para poder estudiar más detenidamente este circuito se va a trabajar sobre el circuito simplificado de la *Figura 57*.

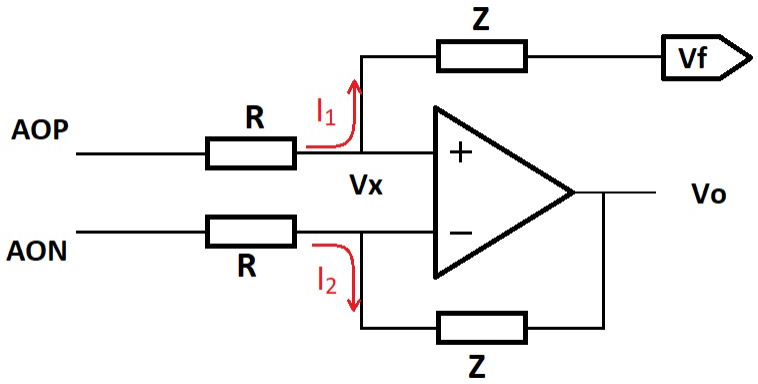

Figura 57. Circuito acondicionamiento sensor corriente simplificado

Donde Z es el paralelo de R y X_C:

$$X_C = \frac{1}{2\pi f \cdot C}$$

$$Z = \frac{1}{\sqrt{\left(\frac{1}{R}\right)^2 + \left(\frac{1}{X_C}\right)^2}}$$

Debido a la realimentación negativa, ambos terminales positivo y negativo del amplificador se encuentran teóricamente al mismo voltaje, V_x. Se procede a analizar por separado ambos nodos de las corrientes I_1 e I_2:

$$I_1: \frac{V_{AOP} - V_x}{R} = \frac{V_x - V_f}{Z}$$
$$I_2: \frac{V_{AON} - V_x}{R} = \frac{V_x - V_o}{Z}$$

A continuación se despeja V_x en ambas ecuaciones y se igualan ambos resultados:

$$I_1: V_x = \frac{Z \cdot V_{AOP} + R \cdot V_f}{R + Z}$$
$$I_2: V_x = \frac{Z \cdot V_{AON} + R \cdot V_o}{R + Z}$$
$$V_o = \frac{Z}{R} \cdot (V_{AOP} - V_{AON}) + V_f$$

Mediante la implementación de este circuito se consigue una ganancia para la entrada diferencial, así como que la salida este referenciada respecto a un voltaje elegido V_f.

El valor de esta ganancia diferencial depende de los valores de las resistencias y los condensadores. Para conseguir una ganancia unidad y teniendo en cuenta que las resistencias de salida para AOP y AON deben ser mayores de 5 kΩ se han elegido los siguientes valores para los componentes: **R = 5,6 kΩ** y **C = 330 pF**

$$\frac{Z}{R} = \frac{\dfrac{1}{\sqrt{\left(\dfrac{1}{5{,}6 \cdot 10^3}\right)^2 + (2\pi \cdot 50 \cdot 330 \cdot 10^{-1})^2}}}{5{,}6 \cdot 10^3} \approx 10$$

En la hoja de especificaciones del Si8920 se indica que la **máxima salida** a escala completa es de **1,65 Vpk** para el máximo en el rango de entrada, ±200 mV. Con esta ganancia de 10 se amplifica el valor de entrada de 20 mV para que encaje con el driver. Para que la señal pueda entrar al microcontrolador sin exceder los 3,3 V se va a emplear un voltaje de referencia de 1,6 V.

Para conseguir este voltaje fijo se va a emplear un **MAX6018AEUR16**. Este dispositivo es un voltaje de referencia de precisión que proporciona a la salida un voltaje de 1,6 V con una precisión de ±0,2 %.

El MAX6018AEUR16 se puede alimentar desde 1,8 V hasta 5,5 V, por lo que se alimentara directamente con los 3,3 V que ya se han dispuesto para el microcontrolador. Además en la salida se va a incluir un condensador de **100 nF** para evitar cualquier ruido que se pueda añadir a la referencia.

Con todo esto queda definida la instrumentación necesaria para adquirir y adecuar las medidas que tome el sensor de corriente. En la *Figura 58* se muestra el diagrama implementado en PSIM para este sensor.

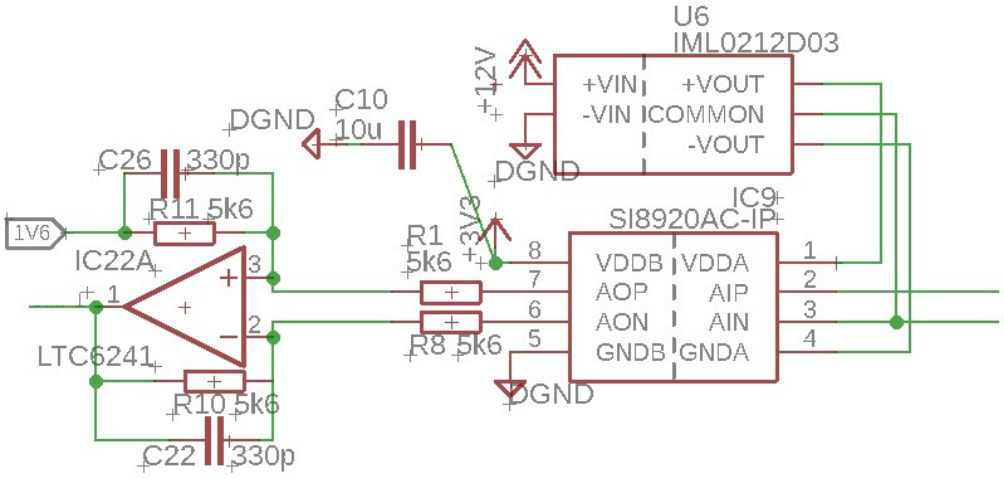

Figura 58. Diagrama componentes sensor corriente en EAGLE

4.6 CAN

La comunicación del sistema con el exterior se va a realizar mediante el protocolo de comunicaciones CAN (Controller Area Network). Se trata de un protocolo esta normalizado, por lo que además de abaratar los costes, simplifica su configuración. Además este protocolo proporciona una alta inmunidad a interferencias, así como ser capaz de detectar fallos en los datos y corregirlos.

Para realizar la comunicación mediante el protocolo CAN bus han de estar presentes 3 componentes principales.

- Microcontrolador con un software capaz de manejar las capas más altas del protocolo, el microcontrolador será abordado más adelante.
- Controlador CAN, este dispositivo tiene que tener implementado el protocolo can en él; en este proyecto el controlador está incorporado en el microcontrolador.
- Transceptor de alta velocidad que convierta la señal de voltaje en una señal diferencial dentro de los estándares de CAN.

Como hardware adicional al microcontrolador se va a utilizar el transceptor **MAX3051**. Este dispositivo cuenta con 4 modos de operación: “high-speed”, “slope-control”, “standby” y “shutdown”. En este proyecto el modo “slope-control” no se va a emplear, con lo que se simplifica la instrumentación necesaria. En la *Figura 59* se muestra el diagrama interno de bloques de este transceptor.

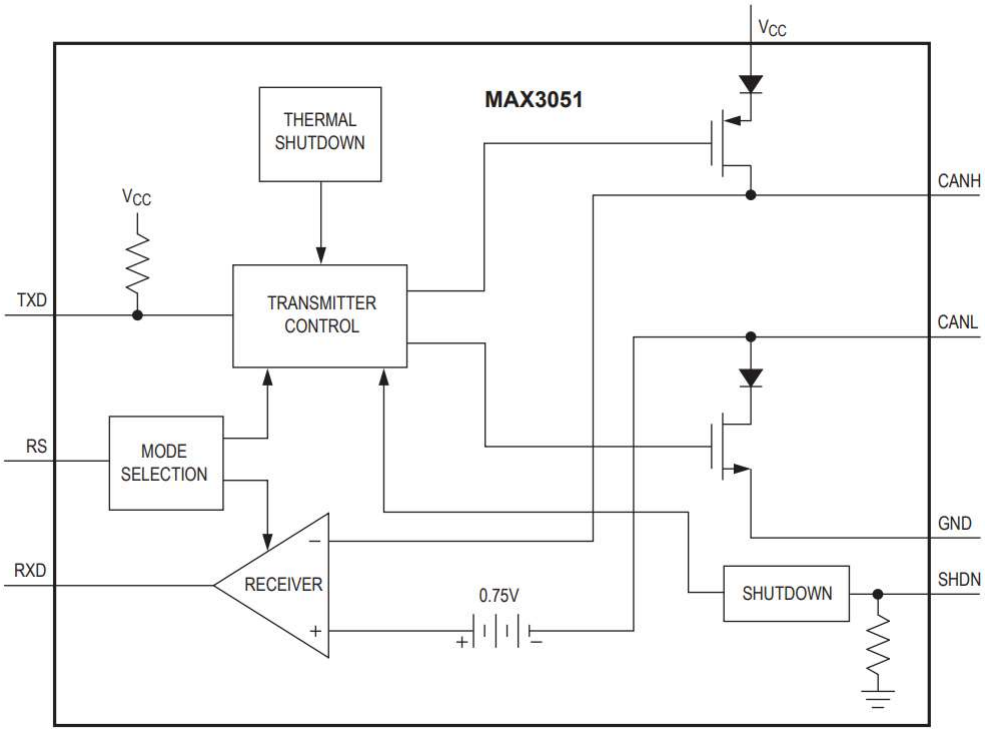

Figura 59. Diagrama funcional de MAX3051

En los buses de comunicación los ruidos e interferencias pueden afectar negativamente a los mensajes. Para proteger el bus CAN de descargas electroestáticas se van a incluir dos **ESDCAN02-2BWY**. Estos dispositivos funcionan como supresores del voltaje transitorio, especialmente diseñados para aplicaciones con CAN bus. Esta protección resulta indispensable por el hecho de que la tarjeta va a estar trabajando cerca de un motor, pudiendo ser afectada por descargas del motor. En la *Figura 60* se puede observar el diagrama interno de este dispositivo.

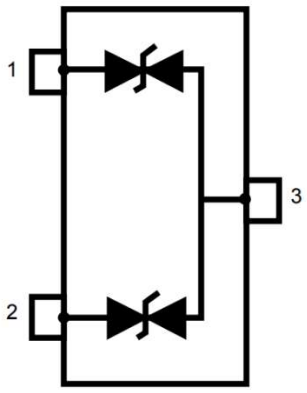

Figura 60. Diagrama interno de ESDCAN02-2BWY

Se han seguido los consejos de configuración típica incluidos en la hoja de especificaciones del transceptor, y se han añadido ambos dispositivos supresores. Para conectarse con el exterior se han añadido un conector de 9 pines **F09VB** y un conector de 3 pines **22-23-2031**. En la FIGURA se muestra el diagrama de conexiones para el módulo CAN. Se han incluido etiquetas para las señales importantes: SDA, SCL, CAN_TX, CAN_RX,CANL y CANH. Los detalles de estas señales se estudiarán con detenimiento en el apartado de programación. En la línea de transmisión se recomiendan incluir resistencias de fin de línea de 120 Ω, en este caso se han escogido 2 resistencias de 56 Ω para poder incluir un condensador a tierra de 22 nF. En la Figura 61 se muestra el diagrama implementado en EAGLE.

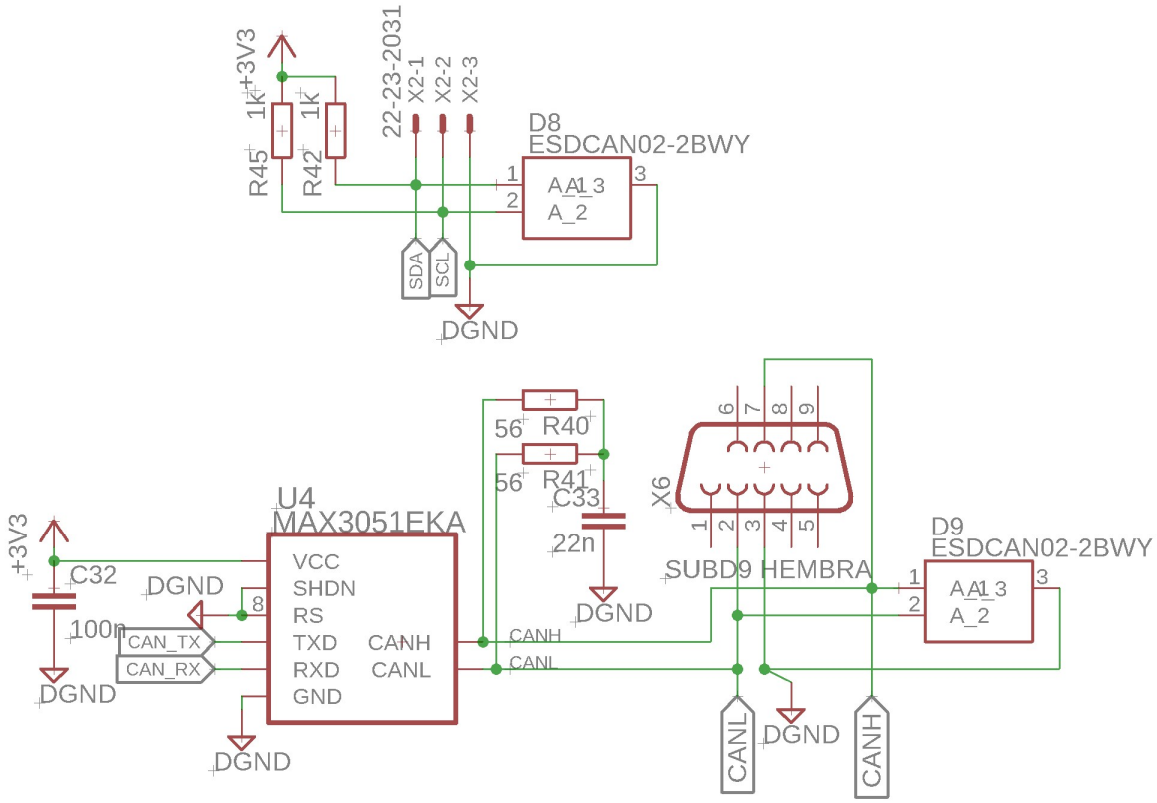

Figura 61. Diagrama componentes del módulo CAN en EAGLE

4.7 Microcontrolador

El microcontrolador que se ha escogido para este trabajo es el **DSPIC33EP512GM710-I_PF**. Se ha elegido este dispositivo ya que fue el utilizado por la empresa de prácticas y contar con todos los módulos

necesarios para esta aplicación. En la *Figura 62* se muestra un esquema de los módulos internos del microprocesador, así como los requeridos para este proyecto. [8]

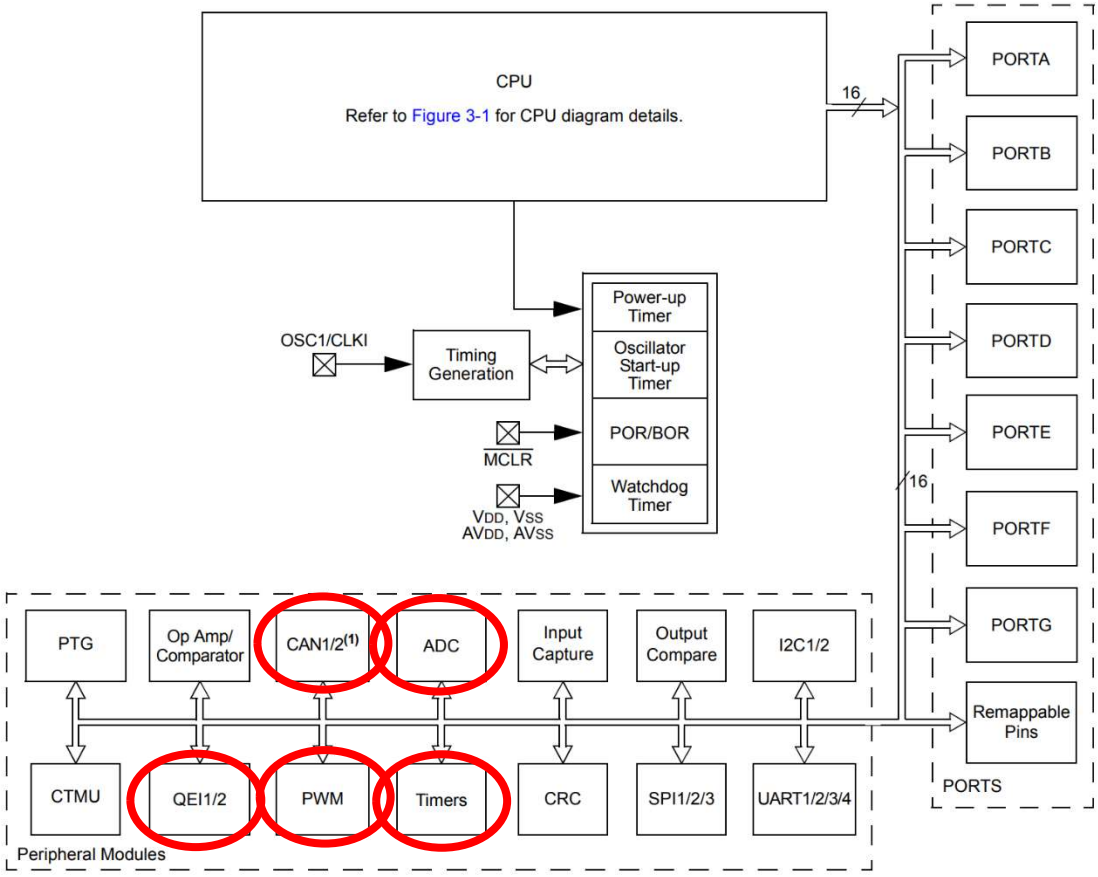

Figura 62. Diagrama de bloques internos del microcontrolador

Este micro controlador consta de 100 pines, de los cuales se van a utilizar una pequeña parte, ya que en una primera instancia se pretendía controlar la velocidad de 4 motores diferentes.

El dispositivo cuenta con 12 salidas PWM de alta velocidad, así como de 21 contadores de carácter general. Además cuenta con 7 puertos (PORTA-PORTG) de 15 pines cada uno. En la *Figura 63* se muestra la disposición de los pines en este microcontrolador:

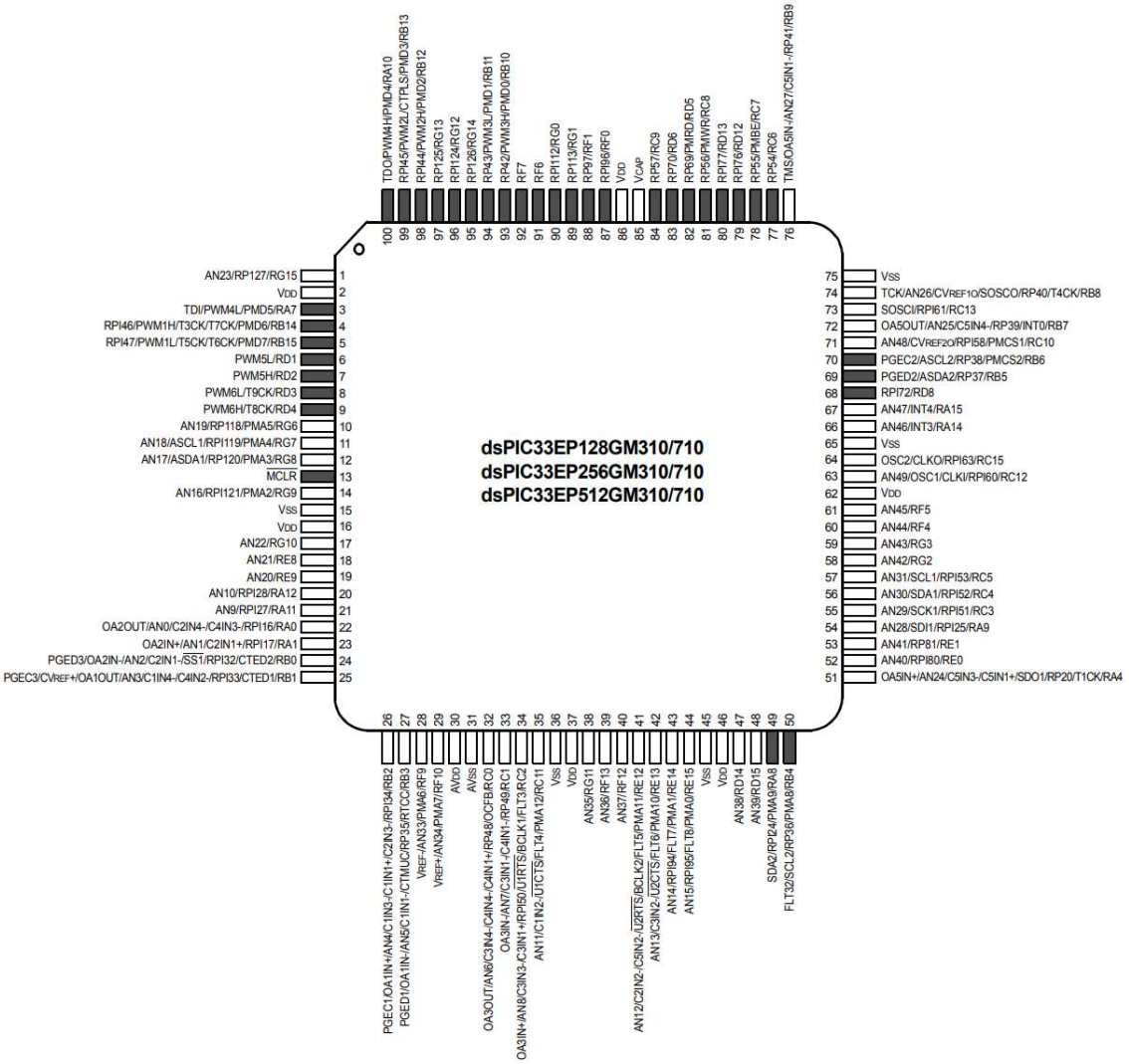

Figura 63. Disposición pines DSPIC33EP512GM710-I_PF

El microcontrolador se alimenta a **3,3 V** y contará con la señal de reloj de un oscilador externo de 10 MHz que se conectará en el pin 63 **OSC1.** Además es necesario incluir un **condensador** de **10 µF** en el pin V_{CAP}, mediante este pin se estabiliza el voltaje del regulador de salida de voltaje. Todos los pines de alimentación, masa y la conexión del oscilador se muestran en la *Figura 64*.

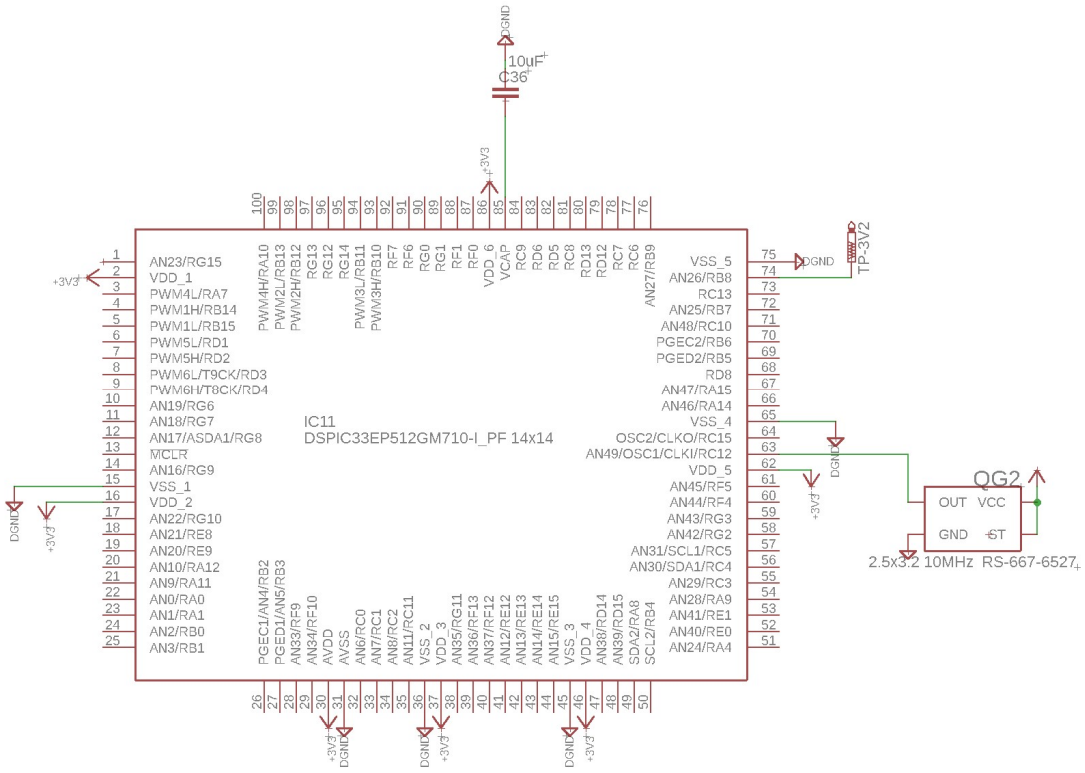

Figura 64. Conexiones alimentación, masa y oscilador

Para programar el microcontrolador se emplean los pines PGEC y PGED, se ha de tener en cuenta que la "Communication Channel Select" que se ha programado en el dispositivo coincide con las conexiones físicas. Además también se emplea el pin $\overline{MCLR}$, este pin tiene las funciones de "Reset" y "Programming and Debugging". Para la programación se recomienda incluir una resistencia de máximo 10 kΩ entre alimentación y este pin, para este caso se ha escogido una resistencia de 5,6 kΩ.

Para introducir la señal se va a hacer por medio de un **RJ11 6P6S**, es un conector modular para una corriente de 1,5 A y voltajes hasta 120 VAC.

Figura 65. Conector RJ11 para programación

En la *Figura 66* se muestra el diagrama en EAGLE de este conector y su configuración.

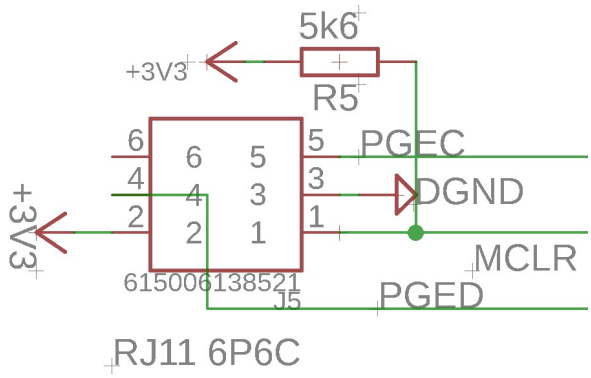

Figura 66. Conector RJ11 y pines programación en EAGLE

Además se van a utilizar una serie de pines para la adquisición y envío de señales los cuales se describen en la ***¡Error! No se encuentra el origen de la referencia.***. En el siguiente apartado se explicará la configuración para cada una de las entradas.

Tabla II. Referencia pines-señales

SEÑAL	PIN
Transistor A1	**PWM1H**
Transistor A2	**PWM1L**
Transistor B1	**PWM2H**
Transistor B2	**PWM2L**
Transistor C1	**PWM3H**
Transistor C2	**PWM3L**
Sensor corriente	**AN25**
Encoder QEA1	**RPI124**
Encoder QEB1	**RPI125**
CAN_TX	**RP48**
CAN_RX	**RP49**
SDA	**SDA2**
SCL	**SCL2**

4.8 Diseño tarjeta electrónica

Una vez elegidos todos los componentes y diseñada la instrumentación necesaria para cada uno se puede proceder a diseñar la placa electrónica. En la *Figura 67* se muestra un diagrama completo que incluye todos los elementos que forman parte de la placa.

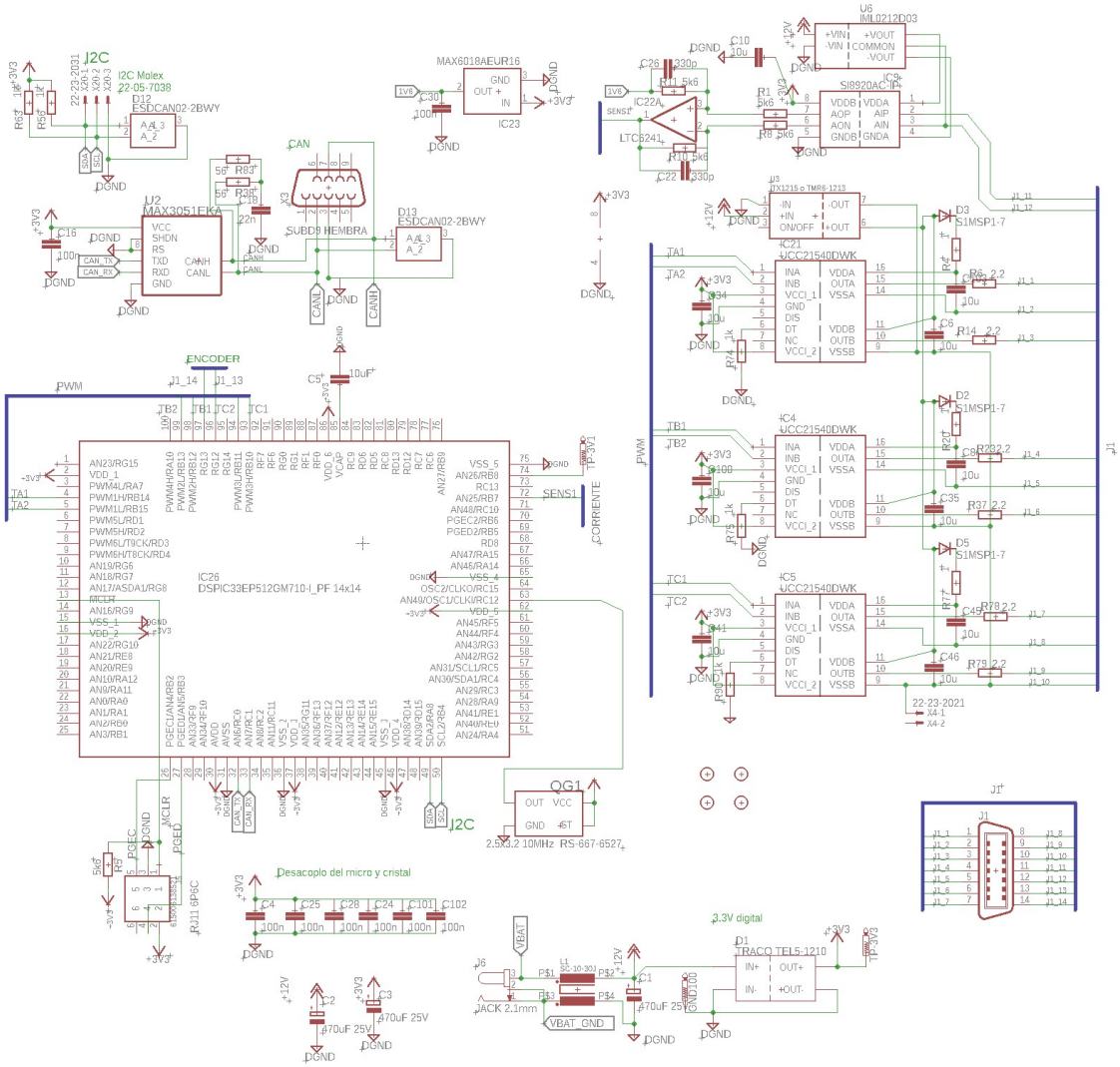

Figura 67. Diagrama completo de elementos en EAGLE

Con todos los elementos bien definidos y todas las conexiones entre ellos definidas en el esquemático se puede proceder a diseñar la placa con todos sus componentes. En la *Figura 68* se puede ver la disposición inicial de los componentes que proporciona el software EAGLE. A partir

de este diagrama el diseño de la placa consiste en ubicar de manera correcta todos los componentes y trazar las pistas necesarias.

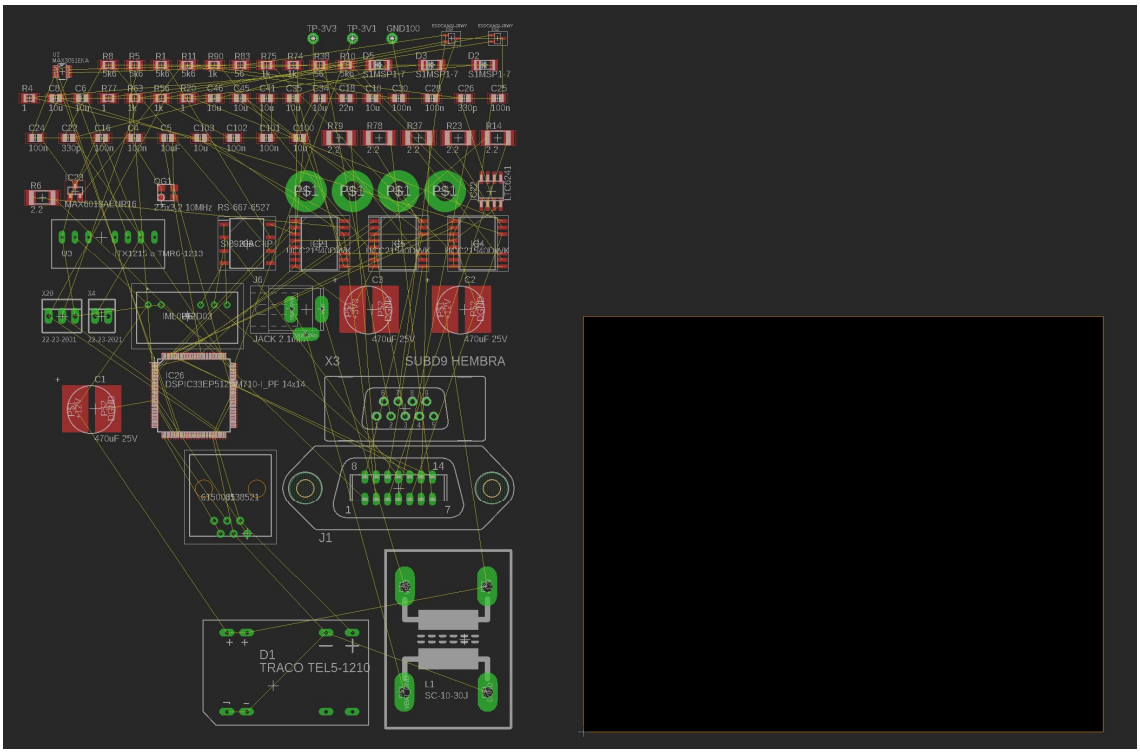

Figura 68. Disposición inicial de componentes en placa en EAGLE

Los bordes naranjas delimitan la zona de la placa, por lo que todos los componentes que se quiera que formen parte de la placa deberán estar en su interior. Los elementos de los distintos bloques se van a organizar según del diagrama que se muestra en la *Figura 69*.

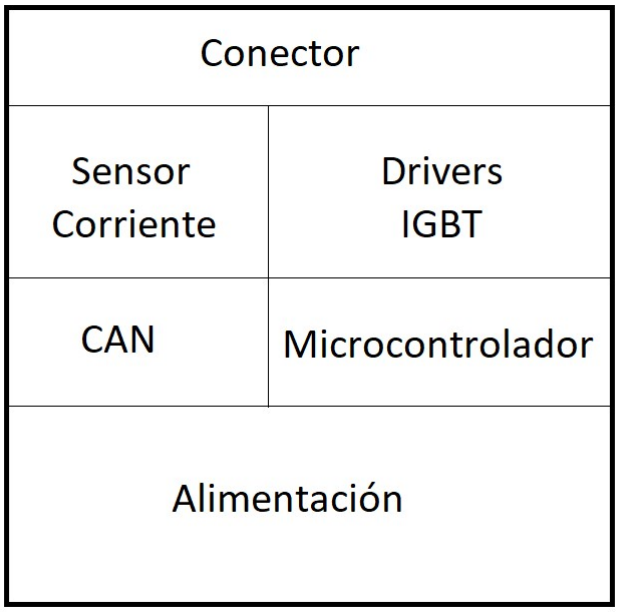

Figura 69. Disposición bloques en placa

El motivo de colocar los elementos de esta manera es principalmente el de acortar la longitud de las pistas entre elementos. Esto es importante ya que cuanto mayor sea esta distancia mayor probabilidad de que el ruido e interferencias distorsionen las señales. La placa va a constar de 3 capas (superior, inferior y central) para facilitar en gran medida la conexión entre elementos y los planos de tierra.

En primer lugar, el sensor de corriente y los drivers de los IGBTs se colocan cerca del conector, ya que las señales que provienen del exterior de la placa entran directamente en ellos. De esta manera además se consigue separar la parte aislada de estos elementos del resto del circuito, lo que es muy recomendable ya que las masas son diferentes.

La alimentación de la placa se ubica en la parte inferior para estar lo más alejada posible del resto de elementos. Se dispondrá una pista de mayor grosor para alimentar las demás partes del circuito.

Por último el módulo CAN y el micro controlador se ubicarán en la parte central de la tarjeta. Es importante que el micro este accesible al resto de elementos, ya que es la parte principal de la placa.

El primer paso en el diseño de la placa consiste en ubicar cada uno de los componentes dentro del área de fabricación. A su vez se han colocado los planos de tierra de la placa. Los planos de tierra sirven como retorno de

corriente de muchos elementos sin la necesidad de crear pistas largas en la placa, además de reducir posibles interferencias y ruido EMI.

En la *Figura 70* se puede ver la ubicación de todos los elementos así como los planos de tierra. Los planos de tierra se encuentran en la capa superior e inferior cubriendo todos los elementos del circuito. Se puede ver como no llegan a la parte superior de la placa, esto se debe a que estos elementos presentan aislamiento galvánico, por lo que su masa es diferente a la del resto del circuito.

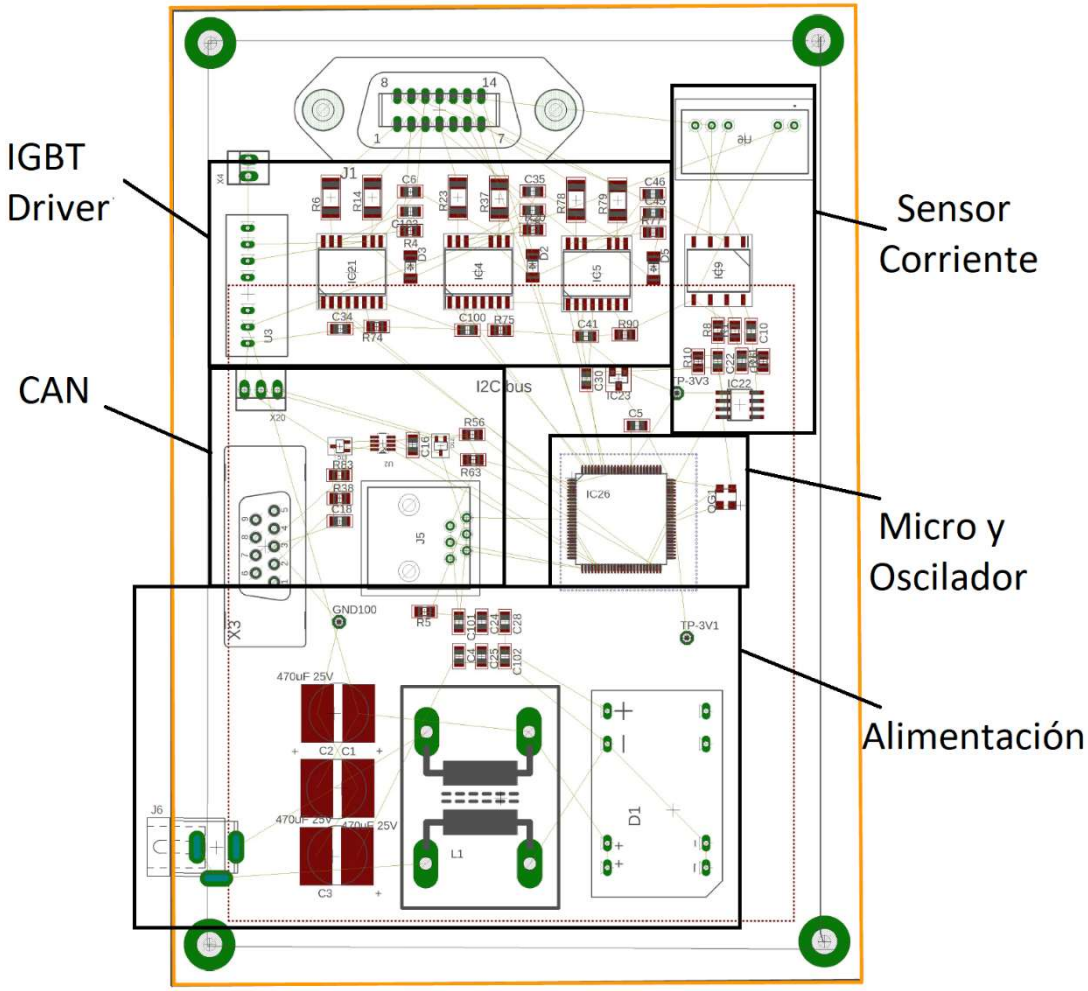

Figura 70. Ubicación elementos y planos de tierra en EAGLE

Se ha añadido un plano de tierra adicional para el microcontrolador en la capa central. Esto es muy recomendable, ya que el microcontrolador es la parte más importante y sensible del circuito.

El siguiente y último paso es trazar las pistas entre los elementos del circuito. En el programa las conexiones vienen indicadas por líneas

amarillas, por lo que se trata de trazar las pistas del tamaño adecuado y en la capa adecuada. En la *Figura 71* se muestra el resultado final de la placa con todas las conexiones y planos de tierra.

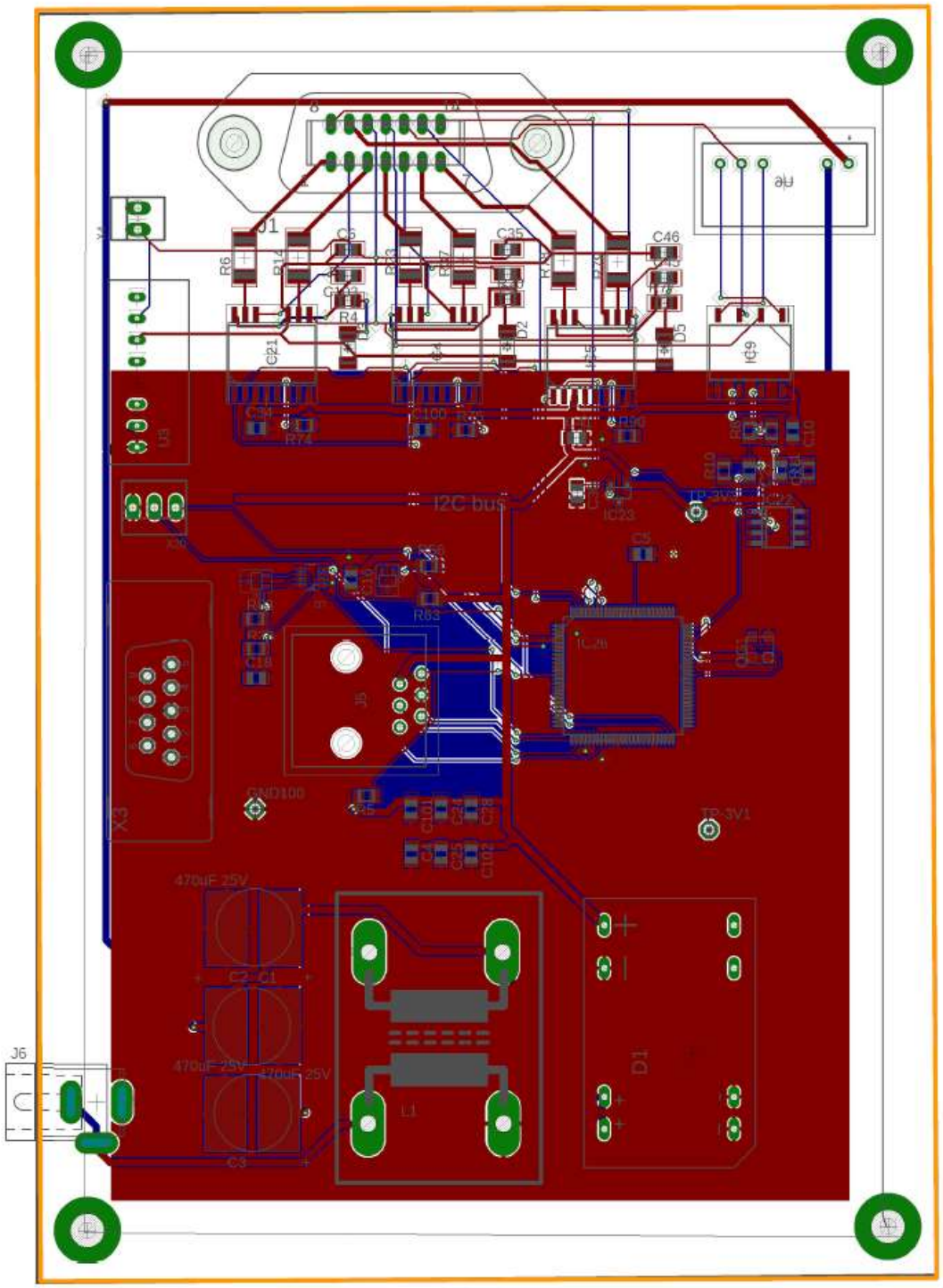

Figura 71. Diseño final de la placa

5. PROGRAMACIÓN DEL MICROPROCESADOR

El microprocesador es el encargado de realizar la gran mayoría de funciones de este proyecto. Todos los bloques de control, así como el control y análisis de entradas y salidas pasan por el micro.

Como ya se ha mencionado en el apartado anterior, para esta aplicación se ha escogido el microprocesador **DSPIC33EP512GM710-I_PF** ya que incluye todos los módulos necesarios así como una velocidad de procesamiento elevada. Los pines y funciones no utilizados se guardan para una posible ampliación de la tarjeta, ya sea para controlar más de un motor o añadir diferentes sensores.

En este apartado se va a estudiar todo lo necesario para programar el micro mediante el software MPLAB X IDE. Esto es entender la estructura interna, comprender las funciones que tiene que desarrollar en cada ciclo y finalmente el código encargado de que todo esto se cumpla.

5.1 Tareas y lazo de control

Para que el control funcione correctamente, el micro deberá realizar múltiples tareas en cada periodo de conmutación, es por esto que se deben ordenar y preparar cada una de las tareas para que estén sincronizadas entre sí.

5.1.1 Tareas

En esta sección se va a analizar las principales tareas que se deben realizar en cada periodo para que el micro sea capaz de controlar perfectamente el motor sin poner en riesgo el circuito.

- Lectura de sensores: En este sistema se cuenta únicamente con dos sensores: sensor de velocidad y sensor de corriente. El sensor de velocidad es el encargado de que el control se realice correctamente, en cada periodo se debe tomar el valor de este sensor para que sea llevado al lazo de control. El sensor de intensidad es un método de protección, si no es leído en cada periodo el circuito podría

someterse a sobreintensidades y resultar dañado.

- Cálculo de tensiones: El motor generará un par según la tensión y la frecuencia a la que sea sometido. En cada periodo se debe obtener el valor de la tensión a aplicar en el motor. Mediante el lazo de control se puede obtener este valor.

- Traducir tensiones: Cada valor de tensión que se ha obtenido se debe traducir a una secuencia de unos y ceros que el sistema sea capaz de interpretar. El inversor PWM funciona con secuencias de encendido y apagado en cada uno de los IGBTs.

- Comunicaciones: En cada ciclo el sistema debe de enviar un mensaje si ha ocurrido algún fallo en el sistema. Además se enviarán los valores actuales de velocidad e intensidad.

- PWM: en cada inicio de cada ciclo, se envía la señal de conmutación para los drivers de los transistores. Es importante hacerlo en cada ciclo, ya que el control se aplica a su vez en cada periodo de conmutación.

El esquema que se va a llevar en la programación se muestra en la *Figura 72****¡Error! No se encuentra el origen de la referencia.***. Mediante un diagrama de flujo se pretende mostrar de qué manera funcionará la tarjeta, como se tomarán las decisiones importantes y como se plantea el paso de una tarea a otra.

El programa se inicia habilitando todos los módulos y el timer que marcará el periodo de conmutación. La lectura de los sensores, el lazo de control, el envío de mensajes y la carga de PWM se aplicarán en una rutina de interrupción, de tal manera que en cada periodo de conmutación se hallan tomado todas las lecturas y realizado todos los cálculos necesarios.

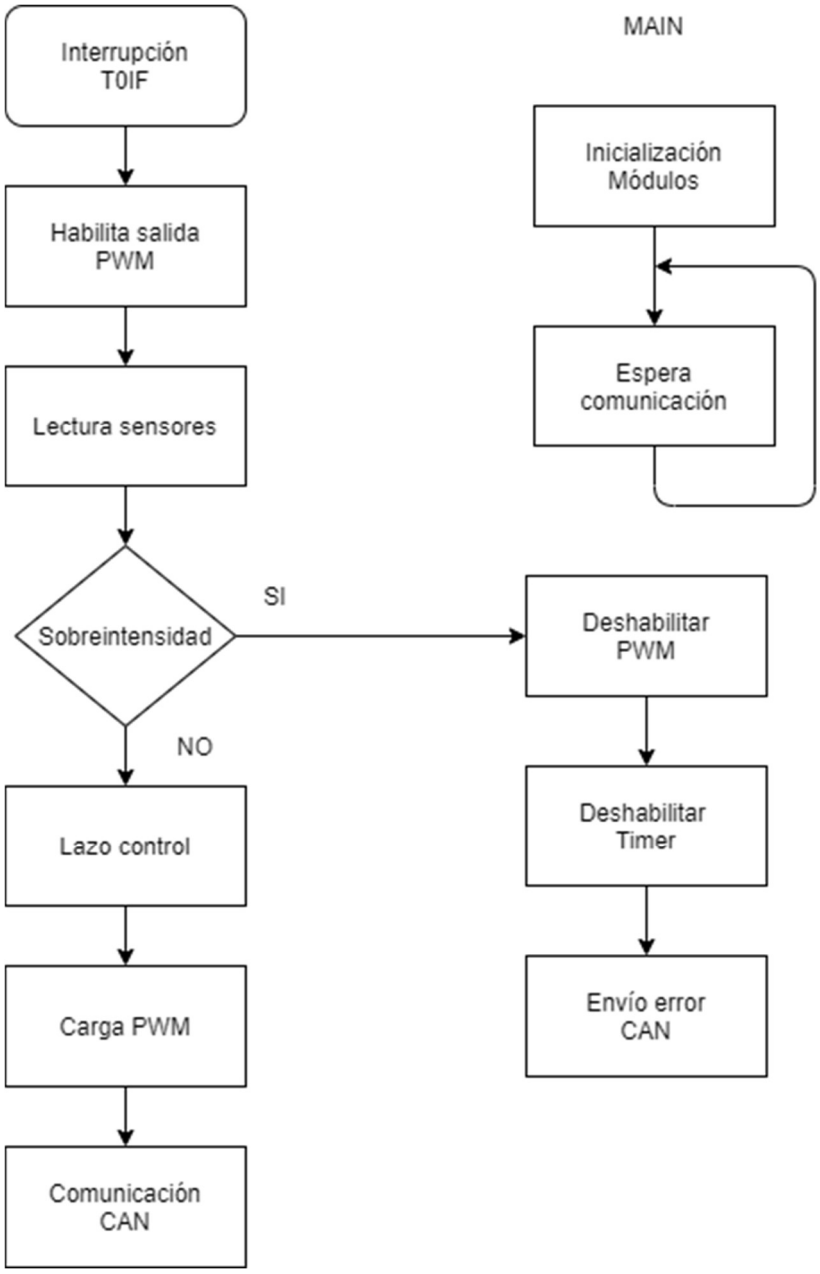

Figura 72. Diagrama de flujo de tareas microprocesador

5.1.2 Sincronización

Para que las tareas se realicen correctamente y en el orden adecuado es necesaria la sincronización de todas ellas. Con esto se consigue estructurar el programa dentro del micro para que todo funcione de manera correcta y en el momento correcto.

Como sincronización se va a emplear el periodo de conmutación de los IGBTs del sistema PWM. Se va a trabajar de este modo ya que aunque lo

que de verdad se quiere controlar la velocidad del motor, es mediante la conmutación de los transistores como se va a aplicar la tensión al motor.

La frecuencia de conmutación de los transistores de 5 kHz, la misma frecuencia que la señal triangular generada. El oscilador del microprocesador es de 10 MHz, por lo que habrá que generar ciclos de trabajo para cada periodo de conmutación (T_{CON}).

$$T_{CON} = \frac{1}{f_{CON}} = \frac{1}{5000\ Hz} = 200\ \mu s$$

En cada T_{CON} el programa deberá tomar lecturas de ambos sensores, calcular la tensión que se debe aplicar en el motor y traducirlo a la secuencia de disparo de los transistores IGBTs. En el siguiente periodo, el sistema envía la secuencia de disparo para después repetir los pasos mencionados.

La comunicación con los periféricos se realizará después del lazo de control en cada ciclo de conmutación o cuando se quiera introducir un nuevo parámetro que se activará un flag para iniciar la comunicación. Esta tarea adicional deberá estar a su vez sincronizada con el programa para así no perder la estructura.

Es por esto que todas las tareas de control se van a realizar dentro de una rutina de interrupción. De esta manera se asegura que todas las tareas estén sincronizadas correctamente. En el main del programa se va a estar únicamente a la espera de recibir nuevas instrucciones.

5.1.3 Inicialización

Hasta ahora se ha analizado el funcionamiento del programa, pero es igual de importante que se inicialice correctamente. Al encender la placa todas las variables que se hubieran almacenado deberán volver a cero, así como el módulo PWM, ya que de otra manera mandaría señales erróneas a los controladores. En el inicio se han de configurar a su vez todos los registros necesarios para garantizar el correcto funcionamiento de cada uno de los módulos.

Al ser el módulo PWM la parte principal de la aplicación, será el que comience la sincronización. Al iniciarse este módulo la señal de sincronización se activará e iniciará con ella todas las tareas restantes.

5.2 Programación de los módulos

El microcontrolador **DSPIC33EP512GM710-I_PF** cuenta con diversos módulos y registros, de los cuales solamente algunos se van a emplear en este trabajo. En este apartado se van a describir y estudiar los bloques más importantes para este proyecto.

En los siguientes apartados se estudiará la estructura y funcionamiento de cada uno de los módulos que se van a emplear en este proyecto, además de describir los registros de configuración para cada uno de ellos [8]. Por último se indicará la configuración que se establecerá para cada uno de ellos.

5.2.1 Convertidor analógico digital

En el **DSPIC33EP512GM710** se cuenta con dos módulos de conversión: ADC1 y ADC2. El primer módulo admite hasta 49 canales de entrada analógica, mientras que el segundo admite hasta 32. Cada uno de ellos permite además la conversión a 10 bits, para 4 “Sample-and-Hold”, o bien conversión a 12 bits para 1 solo canal.

En este proyecto se va a emplear el conversor de 10 bits ya que la resolución que se obtiene es más que suficiente. En este modo de trabajo se obtienen velocidades de conversión de hasta 1,1 Msps así como la conversión simultanea de hasta 4 entradas analógicas. Las entradas analógicas que se quieren leer son la señal del sensor de velocidad, y la señal del sensor de corriente.

La lectura del sensor de intensidad se realizara mediante el módulo de conversión AD del micro procesador.

Para la adquisición de datos se ha de tener en cuenta que se requiere un tiempo de muestreo antes de comenzar la conversión, para esta aplicación con un tiempo de **2 μs** es suficiente. Por otro lado se han de configurar los

puertos analógicos que se van a utilizar, así como los registros de configuración que se han desarrollado en el apartado anterior.

REGISTROS DE CONFIGURACION

ADxCON1: Configuración del modo de conversión de 10 o 12 bits, así como el formato del dato de salida o la fuente de reloj. Además cuenta con el bit DONE que indica cuando ha terminado la conversión y con el bit ADON, encargado de encender o apagar el módulo.

ADxCON2: Elección de los voltajes de referencia para la conversión y la selección de canal para la conversión; si se quiere leer solamente 1 canal, 2 o 4 canales.

ADxCON3: Configuración de la fuente del reloj de conversión, así como el tiempo de adquisición de los datos.

ADxCON4: Elección del lugar de almacenamiento de datos; los datos se pueden guardar en el buffer del conversor A/D o si se transfieren al DMA del sistema.

ADxCHS123: Configuración de los pines de entrada para los canales de conversión 1, 2 y 3.

ADXCHS0: Configuración del pin de entrada para el canal 0 del módulo.

ADxCSSH y **ADxCSSL**: Selección del escaneo de cada uno de los inputs; si se quiere escanear o saltarse. En caso de escanearse y no tener una entrada se convierten en V_{REF}.

5.2.2 Módulo ECAN

En este microcontrolador se permite el uso de hasta dos módulos CAN, en este proyecto se va a emplear solamente uno de ellos. Este protocolo está diseñado para trabajar en situaciones en las que haya mucho ruido. Permite el empleo de ambos protocolos CAN 2.0 A y B (pasivo o activo). [9]

Un frame de datos standard de CAN cuenta con los siguientes campos:

- SOF: bit de inicio de frame (Start-of-Frame)
- Arbitration: 11-bits de identificador (SID) y 1-bit tipo de dato (RTR)

- Control: 6-bits con información del tipo de mensaje
- Data: contiene los datos del mensaje (de 1 a 8 bytes)
- CRC: Secuencia de datos que comprueba errores
- ACK: indica si el mensaje se ha recibido
- EOF: 7-bits recesivos para terminar el mensaje

En la *Figura 73* se muestra un frame de datos standard de CAN.

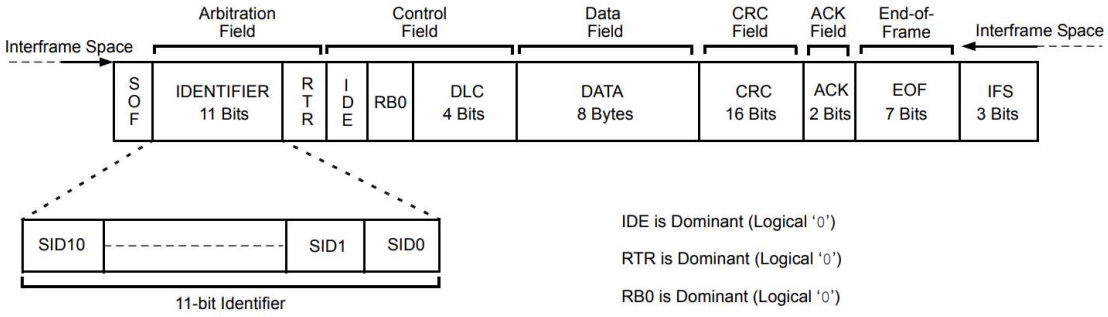

Figura 73. Frame de datos standard en CAN bus

El protocolo CAN maneja todas las funciones de transmitir y recibir mensajes en el bus CAN. Los mensajes se transmiten cargándolos primero en los registros de datos, mientras que los recibidos se comparan con los filtros para ver si se han de almacenar en uno de los registros. En la *Figura 74* se muestra el diagrama de bloques interno de uno de los módulos.

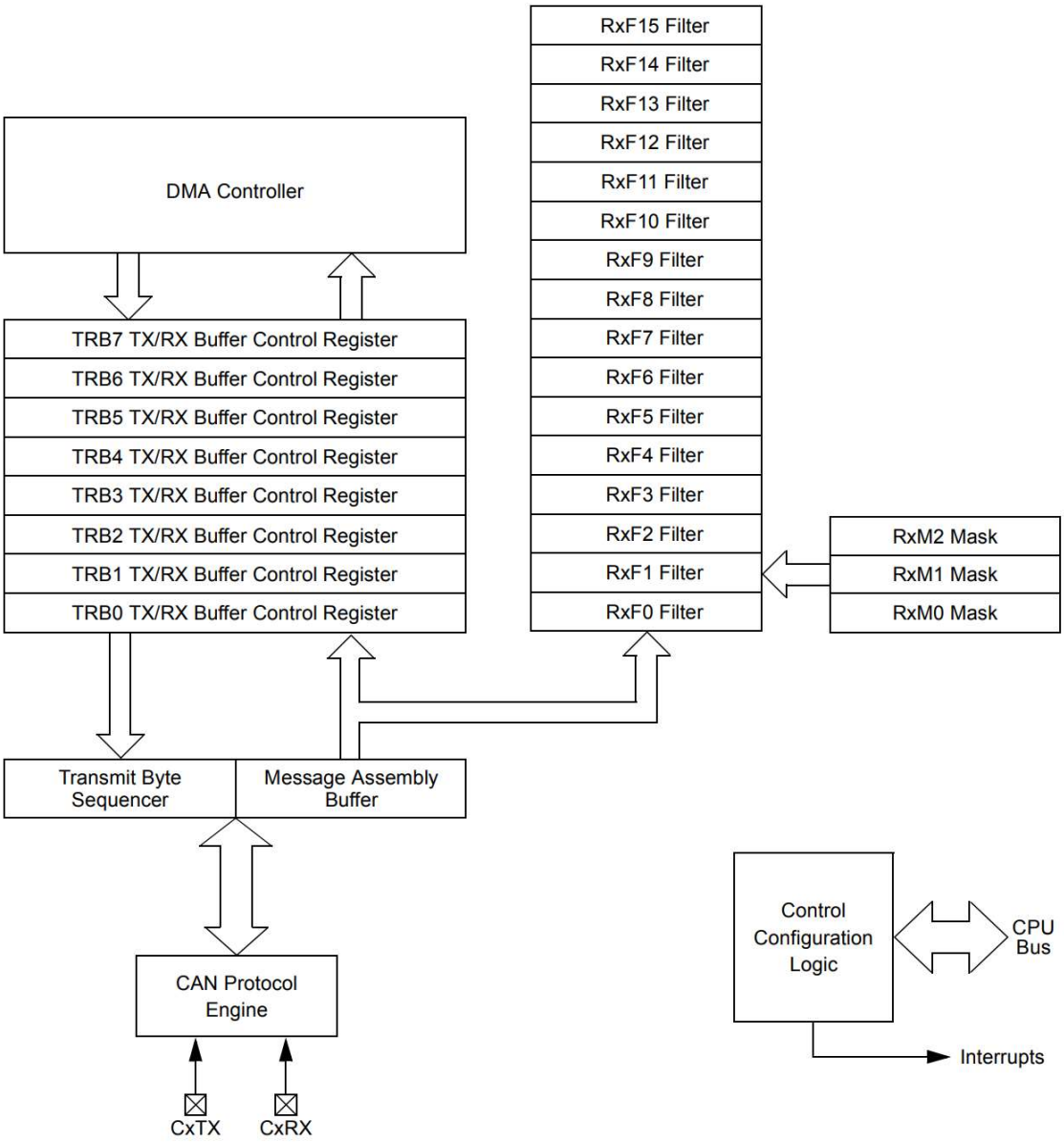

Figura 74. Diagrama de bloques interno del módulo CANx

Mediante los filtros de aceptación los mensajes recibidos se almacenaran en el buffer del DMA o serán descartados. El DMA funciona como interfaz entre los buffers de mensajes y el módulo ECAN.

Para configurar este módulo se va a utilizar el archivo fuente proporcionado por Microchip Technology Inc. para la configuración de los buffer y de los filtros..

Para terminar de configurar el módulo can se ha generado otro archivo fuente con funciones que inicializan el DMA, el reloj y el propio módulo ECAN.

REGISTROS DE CONFIGURACION

CxCTRL1: Configuración el reloj interno del módulo y el modo de funcionamiento en el que se va a trabajar: “Listen All Messages”, “Configuration”, “Listen Only”, “Loopback”, “Disable” o “Normal Operation”.

CxCTRL2: Número de bits del filtro para DeviceNet™ que está incorporado en el módulo CAN. En caso de utilizarse

CxVEC: Configuración para elegir el filtro CAN y para configurar el flag de interrupción del módulo.

CxFCTRL: Configuración del tamaño del buffer del DMA así como el control del FIFO (buffer de 8 palabras).

CxFIFO: Elección del puntero del buffer del FIFO, así como cuál será el siguiente.

CxINTF: Configuración si se activan cada una de las diferentes interrupciones que puede generar el módulo CAN.

CxINTE: En este registro se habilitan o no cada una de las interrupciones generales por el módulo.

CxEC: En este registro se almacena la cuenta de los errores de transmisión o recepción de mensajes.

CxCFG1: Configuración del “baud rate” en función de la frecuencia del reloj escogido para el módulo.

CxCFG2: En este registro se escogen opciones adicionales para los tiempos y esperas del módulo.

CxFEN1: En este registro se habilitan o no cada uno de los filtros del módulo.

CxBUFPNT1-2-3-4: En este registro se escoge en que buffer se van a aplicar los filtros del módulo, así como la máscara para los filtros 0-3 (1), 4-7 (2), 8-11 (3) y 12-15 (4) .

CxRXFnSID: Configuración del SID para filtro de aceptación **n**. En caso de que el identificador sea extendido (EID) se configura en este registro y se incluyen 2 bits del filtro.

CxRXFnEID: En caso de haber habilitado el identificador extendido, se configuran el resto de los bits del filtro.

CxFMSKSEL1-2: En este registro se escoge la máscara para cada uno de los filtros 0-7 (1) o 8-15 (2).

CxRXMnSID: Confiiguracion el SID de la máscara n, además se configuran dos bits para el EID.

CxRXMnEID: En este registro se configuran el resto de bits para la máscara en caso de que el identificador sea extendido.

CxRXFUL1-2: Configuración si cada uno de los buffers está lleno o se ha vaciado por el usuario.

CxRXOVF1-2: Configuración si los buffers se escriben hasta llenarse o si no se escoge la condición de "overflow".

CxTRmnCON: En este registro se controla si cada uno de los buffers mn es de transmisión o recepción de mensajes, así como varios parámetros de control para cada uno de ellos.

5.2.3 Módulo PWM de alta velocidad

Este microcontrolador incorpora un módulo PWM con hasta 12 salidas. Este módulo cuenta con 6 generadores PWM en los que en cada uno de ellos se generan 2 salidas: PWMxH y PWMxL. Para cada uno de los 6 generadores se puede escoger un periodo, un tiempo muerto y una fase diferentes.

Cada una de las PWM puede generar un trigger para un módulo de conversión ADC, para muestrear la señal analógica en un momento específico del periodo de la PWM. En la *Figura 75* se muestra el diagrama interno de bloques del módulo PWM.

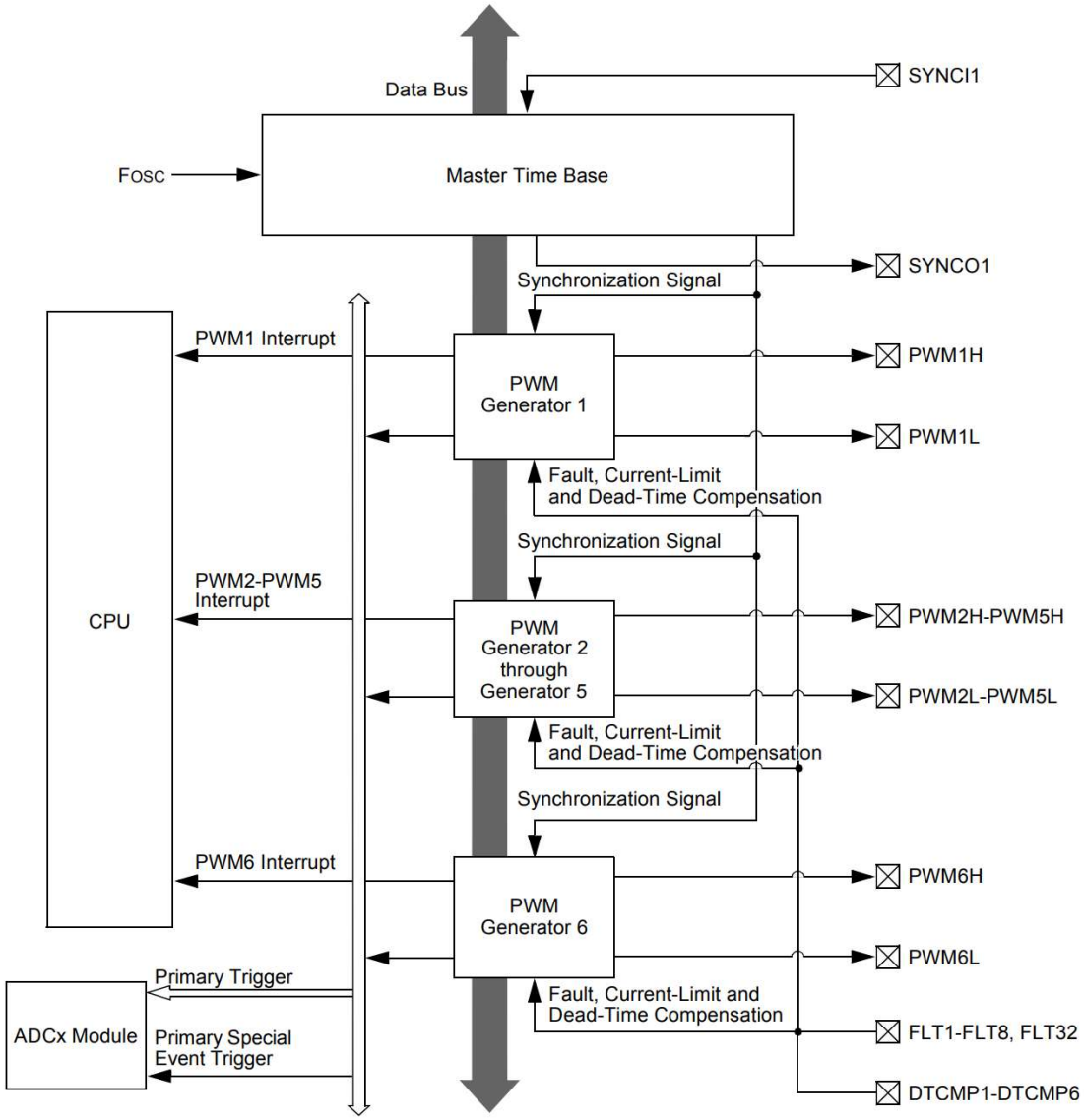

Figura 75. Diagrama de bloques del módulo PWM

EL módulo pwm es el encargado de generar las señales que encenderán y apagaran los transistores que entregan la energía al motor. Además se va a utilizar una función especial que genera un trigger al módulo ADC.

El periodo de conmutación de los transistores es de $\mathbf{T_{CON}=200\ \mu s}$, por lo que habrá que configurar el periodo de la señal PWM para que coincida. La señal de reloj nominal para el módulo es de 120 MHz para garantizar una funcionalidad óptima, para ello se va a configurar el registro ACLKCON para que utilice el reloj interno FRC de 7,37 MHz.

El tiempo base maestro se utiliza por todos los generadores de PWM. Este reloj de entrada cuenta con un preescalar (1..64) y se configura en el registro PTCON2. Este preescalar afecta a su vez a la resolución tanto del

periodo como del “Duty Cycle” de las PWM, pero igualmente disminuye el consumo del módulo. Un preescalar de 1 entrega la máxima resolución de 1.04 ns, mientras que un preescalar de 4 da una resolución de 4,16 ns, pero disminuye el consumo en un 75%.

Para que el módulo ADC inicie justo después de que comience un nuevo periodo de la PWM se generará el trigger en el 5% de cada periodo. Para calcular el registro para el periodo de conmutación se va a utilizar la siguiente expresión:

$$PHASEx = \frac{ACLK \cdot 8 \cdot Tcon}{PCLKDIV} - 8 = \frac{117.93MHz \cdot 8 \cdot 200\mu s}{2} - 8 = 94336$$

El “Duty Cycle se puede configurar independientemente para generador mediante el registro PDCx. Este registro se calculará y enviará automáticamente en el lazo de control. Esto se consigue activando el bit IUE del registro PWMCONx, que habilita la sincronización inmediata del registro PDCx.

Como modo de funcionamiento del módulo PWM se va a seleccionar el modo complementario mediante el bit PMOD del registro IOCON. Esta funcionalidad permite generar salidas complementarias para los pines PWMxH y PWMxL. De esta manera cada una de las salidas gobernará directamente uno de los transistores.

REGISTROS DE CONFIGURACION

PTCON: Configuración de la sincronización de los eventos que puede generar el módulos. Además habilita o deshabilita el módulo PWM.

PTCON2: En este registro se escoge la preescala para el reloj primario del módulo.

PTPER: Configuración del periodo base primario del módulo.

SEVTCMP En este registro se almacena la cuenta de eventos especiales.

STCON: Configuración de la sincronización de los eventos secundarios del módulo.

STCON2: En este registro se escoge la escala para el reloj secundario.

STPER: Configuración del periodo base secundario.

SSVTCMP: En este registro se almacena la cuenta de los eventos especiales secundarios generados por el módulo.

CHOP: Configuración del generador de reloj de cambio (chop) del módulo.

MCD: Configuración del "Duty Cycle" primario del módulo.

PWMCONx: En este registro se habilitan las interrupciones, el limite de corriente y los trigger. También se configura el tiempo muerto, el modo de la señal PWM (alineada en el centro o en el borde) y de donde proviene el "Duty Cycle".

PDCx: En este registro también se configura el "Duty Cylce" del módulo.

SDCx: Configuración del "Duty Cycle" secundario.

PHASEx: Configuración de la fase o el tiempo base independiente para el módulo.

SPHASEx: Configuración del offset de fase secundario.

DTRx: Configuración del tiempo muerto del módulo.

ADTRx: Configuración alternada del tiempo muerto.

TRGCONx: configuración para generar la señal trigger del módulo. Se escoge cuando se genera el evento trigger, y cada cuantos eventos se genera la señal de salida trigger.

IOCONx: Configuración de las salidas de cada PWM. Se puede escoger si cada par de salida es independiente, redundante o complementario.

TRIGx: Valores de comparación que pueden generar el trigger mara el módulo ADC.

FCLCONx: Configuración del modo "fault" para el limitador de corriente. Se escoge si se habilita o no y la polaridad.

LEBCONx: Configuración de qué borde de señal genera el trigger.

AUXCONx: Configuración secundaria del módulo. Se selecciona la fuente de tachado entre las posibles salidas del módulo, además se escoge cual será la fuente del reloj de cambio (chop).

PEMCAPx: Almacena la captura del tiempo base del PWM cuando un borde se detecta en el limitador de corriente.

5.2.4 Timer 1

Como señal que sincronice los ciclos de conmutación se va a emplear el Timer 1 del microcontrolador. Se trata de un timer de 16 bits que puede funcionar como contador, donde la frecuencia interna de instrucción se utiliza como fuente para el contador.

Mediante este módulo se generará la interrupción que dará comienzo a un nuevo ciclo en el microcontrolador. En la *Figura 76* se muestra el diagrama de bloques de este módulo.

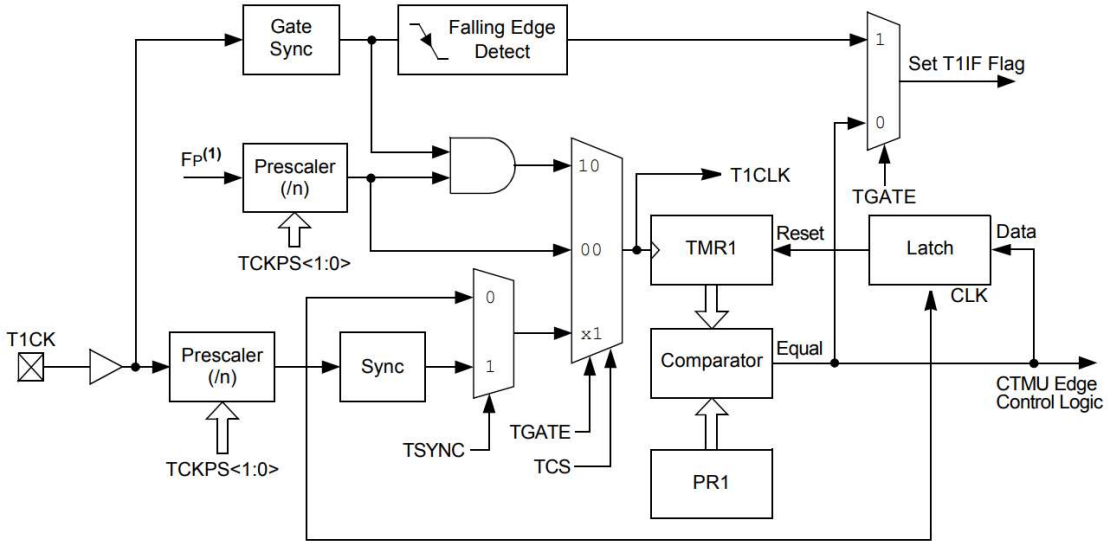

Figura 76. Diagrama de bloques interno del Timer 1

El Timer 1 se encargará de contabilizar el tiempo entre los periodos de conmutación de los transistores. Antes de la conmutación de los transistores, el Timer 1 generará una interrupción que se empleará para enviar la señal PWM previamente cargada a los drivers de los IGBTs.

El periodo de conmutación de los transistores es de $\mathbf{T_{CON}=200\ \mu s}$, y un ciclo de instrucción del microcontrolador es de $\mathbf{T_{CY}=2\cdot T_{OSC}}$. Para este proyecto se ha incorporado un oscilador externo de 10 MHz, por lo que un ciclo de instrucción será de $\mathbf{T_{CY}=200\ ns}$.

Para configurar el Timer 1 hay que escoger un predivisor (1, 8, 64 o 216) y el periodo. Para que timer se desborde cada 200 μs hay que resolver la

siguiente expresión:

$$200 \cdot 10^{-6}\, s = 200 \cdot 10^{-9} \ * \ TCKPS * TMR1$$

Si se escoge un **predivisor** de **8** y un **periodo** de **125**, el Timer se desbordará y generará el flag exactamente cada 200 μs.

REGISTROS DE CONFIGURACION:

T1CON: Registro de configuración del Timer 1. En este registro se inicia el funcionamiento del timer, se selecciona la preescala y se escoge la fuente del reloj: interno o externo.

PR1: En este registro se configura el periodo del módulo.

5.2.5 Módulo QEI

El microcontrolador que se va a incorporar incluye dos módulos QEI (Quadrature Encoder Interface). Estos módulos son una interfaz con los encoder incrementales, permitiendo obtener datos de posición y de velocidad.

Este módulo se va a emplear para medir la posición y velocidad del motor en cada ciclo para aplicar así el lazo de control. En la *Figura 77* se muestra el diagrama de bloques de este módulo.

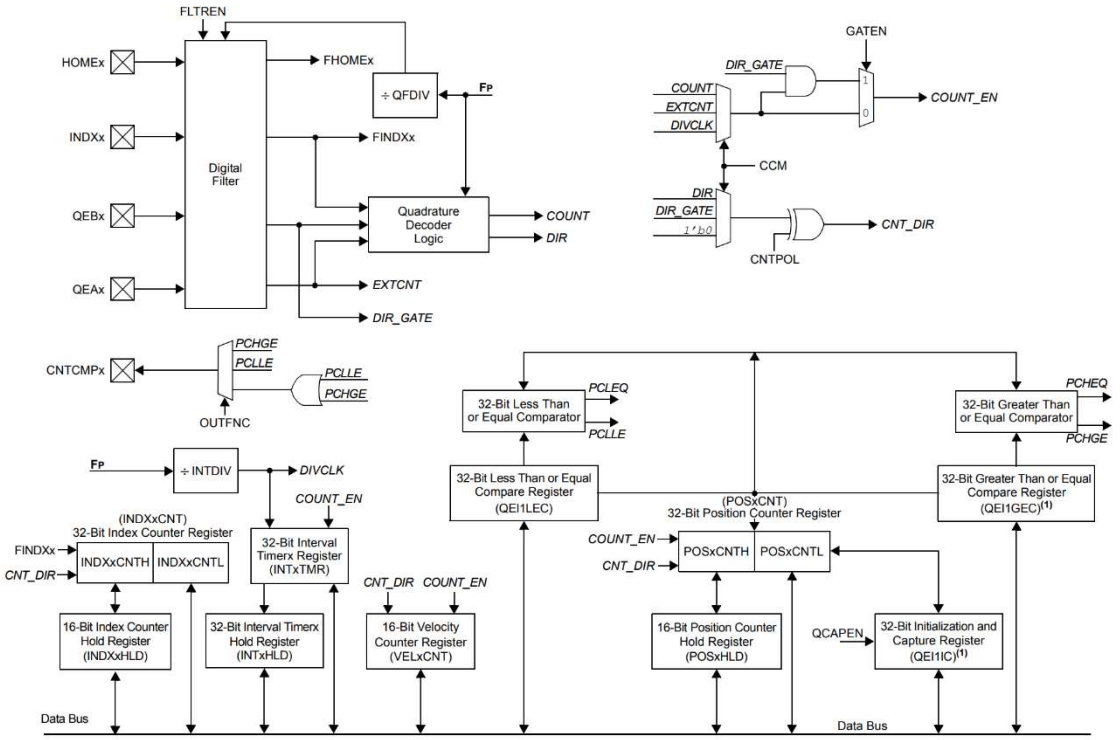

Figura 77. Diagrama de bloques módulo QEI

El módulo QEI1 se va a utilizar para medir la posición y velocidad del motor mediante un encoder incremental de 128 ranuras. Se va a configurar el módulo para que la posición se resetee cada vez que el motor da una vuelta, de esta manera se podrá conocer la posición exacta para realizar el control.

Para obtener el valor de la velocidad actual se cuenta con un registro que almacena las pulsaciones desde la última lectura, por lo que se dividirá ese valor entre el tiempo de adquisición para obtener la velocidad real.

Las entradas para este módulo se configuran mediante el mapeo de inputs. Los registros RPINRx sirven para mapear los pines RPx para cada periférico que se quiera configurar. En concreto para las entradas QEA1 y QEA2 se tiene que configurar el registro RPINR14.

Para obtener los valores de velocidad y posición en valores reales (se trabaja en radianes y dg/s), y no como cuenta de pulsos se van a emplear las siguientes expresiones:

$$Posicion = medida(pulsos) \cdot \frac{2\pi\ (rad/vuelta)}{256\ (pulsos/vuelta)} = \_\_rad$$

$$Velocidad = medida\ (pulsos) \cdot \frac{360\ (deg/vuelta)}{256\ (pulsos/vuelta)} \cdot \frac{1}{T_{con}} = \_deg/s$$

REGISTROS DE CONFIGURACION

QEIxCON: Configuración de reloj, modo de adquisición y habilitación del módulo.

QEIxIOC: Configuración de las entradas y salidas, y predivisores para las entradas.

QEIxSTAT: Configuración de interrupciones para posición y velocidad.

POSxCNTH-L: Contador de posición (2 registros).

POSxHLD: Almacena valor de POSxCNTH durante lectura o escritura.

VELxCNT: Contador de velocidad.

INDXxCNTH-L: Almacenan el contador índice (2 registros).

INDXxHLD: Almacena valor de INDXxXNTH durante lectura o escritura.

QEIxICH-L: Almacenan la inicialización/captura (2 registros).

QEIxLECH-L: Almacenan comparador menor que o igual (2 registros).

QEIxGECH-L: Almacenan comparador mayor que o igual (2 registros).

INTxTMRH-L: Contador de intervalo de pulsos (2 registros).

INTxHLDH-L: Almacena los valores del intervalo de puslos (2 registros).

5.3 Programación main

En este último apartado de la programación se van a ir describiendo cada una de las tareas ya mencionadas para el archivo main del programa. Como se ha mencionado anteriormente, el programa main solamente inicia todos los módulos y se queda a la espera de alguna instrucción. Todas las demás tareas se ejecutarán mediante una interrupción cada vez que se desborde el Timer1.

El programa inicia llamando a las cabeceras de los demás archivos de configuración de los módulos y declarando las funciones y variables que se van a emplear.

En la función main se inicializan los módulos necesarios y se configura el Watchdog. A continuación se entra en un bucle "while" en el que se va a ejecutar la función de leer el bus CAN.

La función de lectura interpreta el ID del mensaje recibido mediante una estructura "switch..case" y en función de éste puede realizar 3 funciones: actualizar la referencia de velocidad, parar el sistema o reiniciarlo tras una parada. Mediante el empleo de esta estructura se pueden añadir funciones adicionales que un operario podría introducir al sistema.

El resto del programa se ejecutará en la rutina de interrupción del Timer1. En la programación del lazo de control se obtiene como resultado el ciclo de trabajo que se aplicará en al módulo PWM para que envíe el valor a los drivers de los transistores. Dado que el microprocesador trabaja sobre datos digitales y no es capaz de comparar señales analógicas en el tiempo, se ha establecido la siguiente relación para obtener el valor del ciclo de trabajo a partir de la señal modulante.

$$Ciclo\ trabajo\ = 0{,}5 + 0{,}5 \cdot modulante$$

Como la frecuencia de la señal triangular con la que idealmente se compararía la modulante es muy superior (5 kHz) a la máxima frecuencia de la modulante (50 Hz) se puede aplicar la expresión anterior sin perder apenas precisión. De esta manera, al calcular el valor instantáneo de la señal modulante se podrá calcular el ciclo de trabajo correspondiente.

Para obtener los valores de velocidad se leerá el registro VEL1CNT, que se resetea cada vez se lee. Para el valor de posición, que se empleará para calcular el seno para la modulante, se obtendrá de la integración de la salida del controlador. Para realizar esta integral se hará uso de la aproximación por rectángulos que ya se ha mencionado anteriormente.

Para realizar el lazo de control se aplicará el controlador PI junto con la constante y el limitador. Le señal que salga del control se adecuará para obtener la señal modulante, de la que se obtendrá el ciclo de trabajo. El ciclo de trabajo se carga en los registros PDCx del módulo PWM. Este

registro es de 16 bits, por lo que el valor 65535 equivaldrá al 100% y el 0 al 0% del ciclo de trabajo.

En el bloque adecuación hay que utilizar la función seno, por lo que será necesario incluir la librería ***math.h*** para este proyecto. Dentro del archivo *main* se van a utilizar funciones que se han programado en otros archivos, por lo que será importante incluir las cabeceras en las que se encuentren dichas funciones.

6. CONCLUSIONES Y LINEAS FUTURAS

Para este trabajo se ha buscado un microcontrolador que en primer lugar tuviese un módulo PWM, ya que esta modulación ha sido la parte principal del proyecto. La comunicación de la tarjeta se realiza mediante el protocolo CAN con un sistema “switch ..case” de tal manera que la tarjeta esta siempre pendiente de si ha llegado alguna petición nueva. Con estos requisitos se ha diseñado tarjeta electrónica al completo capaz de trabajar junto a un banco de potencia con un motor de inducción.

El diseño de la tarjeta electrónica lamentablemente no se ha podido implementar ya que debido al estado de alarma por COVID-19 se interrumpió el tiempo de prácticas en la empresa en la que se estaba desarrollando.

En el último paso del diseño se incluye la propuesta programación con un algoritmo control V/f=cte que se ejecuta mediante la interrupción que se genera en cada periodo de conmutación de la PWM. Incluye un programa principal que funciona constantemente y que se encarga de estar atento a las comunicaciones para gobernar el encendido o apagado de la tarjeta, además de actualizar la velocidad de referencia.

Finalmente indicar que el algoritmo control diseñado se ha verificado mediante sistema simulación específico para sistemas de potencia, con una etapa de potencia y se han obtenido resultados fiables, lo cual valida la propuesta de este TFG.

LINEAS FUTURAS

Una posible mejora para este sistema de control sería introducir el control vectorial de velocidad, ya que permite un control más preciso en el transitorio de las señales. Como ya se dispone en la tarjeta de medidas de corriente, solamente sería necesaria la reprogramación del microcontrolador para poder incluir un control de este tipo

Como sensor de corriente se podría implementar una célula lem. Estos sensores son más avanzados que las R_{shunt} ya que funcionan como transceptores de corriente, obteniendo así el valor de corriente tanto AC, DC como pulsada.

Gracias a la estructura “switch..case” en la lectura del bus CAN se podrían incluir diferentes funcionalidades para la tarjeta. Una de ellas y con una gran utilidad sería tener precargados los parámetros de diferentes motores de tal manera que un operario de la tarjeta pudiera introducir el modelo de la máquina y automática se ajustasen los parámetros del control.

7. BIBLIOGRAFÍA

[1] L. M. y. P. Sanchis, «Máquina asíncrona,» Pamplona, Apuntes de clase para 242401, Dpto. Ingeniería Eléctrica, Electrónica y de Comunicación, UPNA, 2018.

[2] J. D. González, «Inversores PWM,» Gijón, 1999.

[3] E. G. Villabona, «Control escalar del inversor trifásico,» Dpto. Ingeniería Eléctrica, Electrónica y de Comunicación, UPNA, Pamplona, 2020.

[4] E. G. Villabona, «Control digital,» Dpto. Ingeniería Eléctrica, Electrónica y de Comunicación, UPNA, Pamplona, 2020.

[5] F. D. Luna, «Redeweb,» Marzo 2015. [En línea]. Available: https://www.redeweb.com/ficheros/articulos/cemdal_1209500509.pdf. [Último acceso: Junio 2020].

[6] Texas Instruments, «Texas Instruments,» Septiembre 2019. [En línea]. Available: https://www.ti.com/lit/ds/symlink/ucc21540.pdf?HQS=TI-null-null-mousermode-df-pf-null-wwe&DCM=yes&ref_url=https%3A%2F%2Fwww.mouser.es%2F&distId=26. [Último acceso: Junio 2020].

[7] Silicon Labs, «Silicon Labs,» Enero 2019. [En línea]. Available: https://www.silabs.com/documents/public/data-sheets/si8920-datasheet.pdf. [Último acceso: Julio 2020].

[8] Microchip, «Microchip,» 25 Marzo 2014. [En línea]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/70000689d.pdf. [Último acceso: Julio 2020].

[9] I. d. Villar, Bus CAN, Devicenet y Ethernet/IP, Apuntes de clase para 242715, Dpto. Ingeniería Eléctrica, Electrónica y de Comunicación, UPNA, Pamplona, 2019.

7.1 Código de programación

7.2 Listado de datasheets

7.3 Código de programación

Configuración módulo ADC

```
#include <p33Exxxx.h>
/** CONFIGURACION ******************************************************/
_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSECMD & POSCMD_XT & OSCIOFNC_OFF & IOL1WAY_OFF);
_FWDT(FWDTEN_OFF);
_FPOR(FPWRT_PWR128 & BOREN_ON & ALTI2C1_ON & ALTI2C2_ON);
_FICD(ICS_PGD1 & RSTPRI_PF & JTAGEN_OFF);
void initAdc1(void);
void Delay_us(unsigned int);
void lee_sensores(uint16_t ,uint16_t)
int ADCValues[2] = {0, 0};
int i;
int main(void)
{
__builtin_write_OSCCONH(0x03); //Iniciar reloj Oscilador Principal con PLL
__builtin_write_OSCCONL(0x01);
while (OSCCONbits.COSC != 0x3);
while (_LOCK == 0); /
initAdc1();
}
void initAdc1(void)
{
/* Configuracion de puertos */
ANSELA = ANSELB = ANSELC = ANSELD = ANSELE = ANSELG = 0x0000;
ANSELBbits.ANSB0 = 1; // Configurar AN0/RB0 como analogico
ANSELBbits.ANSB5 = 1; // Configurar AN5/RB5 como analogico
/* Inicializar modulo ADC */
AD1CON1 = 0x000C; // Habilita muestreo simultaneo
AD1CON2 = 0x0300; // Muestreo de 2 canales
AD1CON3 = 0x000F;
AD1CON4 = 0x0000;
AD1CSSH = 0x0000;
AD1CSSL = 0x0000;
AD1CHS0bits.CH0SA = 5; // Seleccion de AN5 para CH0 +ve input
AD1CHS0bits.CH0NA = 0; // Seleccion de Vref- para CH0 -ve input
AD1CHS123bits.CH123SA = 0; // Seleccion de  AN0 para CH1 +ve input
AD1CHS123bits.CH123NA = 0; // Seleccion de Vref- para CH1 -ve input
}
void Delay_us(unsigned int delay) // funcion para generar delay en us
{
for (i = 0; i < delay; i++)
{
__asm__ volatile ("repeat #39");
__asm__ volatile ("nop");
}
}
```

Figura 78. Configuración ADC en MPLAB

Configuración Timer1

```
T1CONbits.TON = 0; // Apagado del Timer
T1CONbits.TCS = 0; // Seleccion de reloj: ciclo interno de instruccion
T1CONbits.TGATE = 0; // Seleccion modo Timer
T1CONbits.TCKPS = 0b011; // Preescalar de 1:8
TMR1 = 0x00; // Registro del timer a 0
PR1 = 125; // Carga del periodo del timer
IPC0bits.T1IP = 0x01; // Configurar prioridad de interrupccion del Timer 1
IFS0bits.T1IF = 0; // Limpiar flag de interrupccion del Timer 1
IEC0bits.T1IE = 1; // Habilitar interrupccion del Timer1
T1CONbits.TON = 1; // Encendido del Timer
```

Figura 79. Configuración Timer1 en MPLAB

Configuración módulo PWM

```
/*((FRC* 16) / APSTSCLR) = (7.37 * 16) / 1 = 117.92 MHz */
ACLKCONbits.FRCSEL = 1; // FRC como input
ACLKCONbits.SELACLK = 1; // ACLK genera la fuente de reloj
ACLKCONbits.APSTSCLR = 7; // Divisor de 1 */
ACLKCONbits.ENAPLL = 1; // Activa PLL (preescalar 16)
while(ACLKCONbits.APLLCK! = 1) /* Wait for Auxiliary PLL to Lock */

 PTCON2bits.PCLKDIV = 1  // preescalar de 1:2
 PWMCON1bits.CAM = 0; // Modo alineado con final

/* Configuracion del ADC Special Event Trigger  */
SEVTCMP = 250; // Special Event Trigger configurado al 5% del periodo (4999
PTCONbits.SEVTPS = 0; /* Special Event Trigger posescalar a 1:1 seleccion
                      * (trigger generado cada ciclo) */
PTCONbits.SEIEN = 0; /* Special event interrupt desactivada */
while (PTCONbits.SESTAT == 0); /* Wait for special event status change */

PWMCON1bits.ITB = 1; // periodo generado por PHASEx
PHASE1 = 94336;   // periodo de 200 us para cada generador
PHASE2 = 94336;
PHASE3 = 94336;

PWMCON1bits.MDCS = 0; // Duty Cycle independiente para cada uno
PWMCON1bits.IUE = 1;  // Actualización automatica de Duty Cycle
PWMCON2bits.IUE = 1;  // para cada generador
PWMCON3bits.IUE = 1;

IOCON1bits.PMOD = 0; // Seleccion de modo complementario
```

Figura 80. Configuración PWM en MPLAB_1

```
/*((FRC* 16) / APSTSCLR) = (7.37 * 16) / 1 = 117.92 MHz */
ACLKCONbits.FRCSEL = 1; // FRC como input
ACLKCONbits.SELACLK = 1; // ACLK genera la fuente de reloj
ACLKCONbits.APSTSCLR = 7; // Divisor de 1 */
ACLKCONbits.ENAPLL = 1; // Activa PLL (preescalar 16)
while(ACLKCONbits.APLLCK! = 1) /* Wait for Auxiliary PLL to Lock */

 PTCON2bits.PCLKDIV = 1  // preescalar de 1:2
 PWMCON1bits.CAM = 0; // Modo alineado con final

/* Configuracion del ADC Special Event Trigger  */
SEVTCMP = 250; // Special Event Trigger configurado al 5% del periodo (4999
PTCONbits.SEVTPS = 0; /* Special Event Trigger posescalar a 1:1 seleccion
                      * (trigger generado cada ciclo) */
PTCONbits.SEIEN = 0; /* Special event interrupt desactivada */
while (PTCONbits.SESTAT == 0); /* Wait for special event status change */

PWMCON1bits.ITB = 1; // periodo generado por PHASEx
PHASE1 = 94336;   // periodo de 200 us para cada generador
PHASE2 = 94336;
PHASE3 = 94336;

PWMCON1bits.MDCS = 0; // Duty Cycle independiente para cada uno
PWMCON1bits.IUE = 1;  // Actualización automatica de Duty Cycle
PWMCON2bits.IUE = 1;  // para cada generador
PWMCON3bits.IUE = 1;

IOCON1bits.PMOD = 0; // Seleccion de modo complementario
```

Figura 81. . Configuración PWM en MPLAB_2

Configuración ECAN

```
#include <xc.h>
#include <stdint.h>
#include "ecan1_config.h"


/* Configuracion del filtro de aceptacion n
 inputs:    n: numero de filtro
            id: SID
            bufPnt: buffer para almacenar el mensaje aceptado
            maskSel: seleccion opcional de mascara*/
void Ecan1WriteRxAcptFilter( int16_t n, int32_t id, uint16_t bufPnt,
        uint16_t maskSel )
{
    uint32_t    sid10_0 = 0;

    uint32_t    eid15_0 = 0;

    uint32_t    eid17_16 = 0;
    uint16_t    *sidRegAddr;
    uint16_t    *bufPntRegAddr;
    uint16_t    *maskSelRegAddr;
    uint16_t    *fltEnRegAddr;

    C1CTRL1bits.WIN = 1;

    // Obtain the Address of CiRXFnSID, CiBUFPNTn, CiFMSKSELn
    //and CiFEN register for a given filter number "n"
    sidRegAddr = ( uint16_t * ) ( &C1RXF0SID + (n << 1) );
    bufPntRegAddr = ( uint16_t * ) ( &C1BUFPNT1 + (n >> 2) );
    maskSelRegAddr = ( uint16_t * ) ( &C1FMSKSEL1 + (n >> 3) );
    fltEnRegAddr = ( uint16_t * ) ( &C1FEN1 );

    // Bit-filed manupulation to write to Filter identifier register


       // Filter Standard Identifier
        sid10_0 = ( id & 0x7FF );
        *sidRegAddr = ( sid10_0 ) << 5; // Write to CiRXFnSID Register
        *( sidRegAddr + 1 ) = 0;        // Write to CiRXFnEID Register


    *bufPntRegAddr = ( *bufPntRegAddr ) & ( 0xFFFF - (0xF << (4 * (n & 3))) );
    // limpiar nibble
    *bufPntRegAddr = ( (bufPnt << (4 * (n & 3))) | (*bufPntRegAddr) );
    // Escribir en C1BUFPNTn
    *maskSelRegAddr = ( *maskSelRegAddr ) & ( 0xFFFF - (0x3 << ((n & 7) * 2)) );
```

Figura 82. Configuración ECAN en MPLAB_1

```
    // limpiar 2 bits
    *maskSelRegAddr = ( (maskSel << (2 * (n & 7))) | (*maskSelRegAddr) );
    // Write to C1FMSKSELn Register
    *fltEnRegAddr = ( (0x1 << n) | (*fltEnRegAddr) );
    // Write to C1FEN1 Register
    C1CTRL1bits.WIN = 0;
}

/*  Configuracion de mascara de aceptacion m
    Inputs:     m: numero de mascara
                id: identificador de la mascara*/
void Ecan1WriteRxAcptMask( int16_t m, int32_t identifier, uint16_t mide,
        uint16_t exide )
{
    uint32_t    sid10_0 = 0;

    uint32_t    eid15_0 = 0;

    uint32_t    eid17_16 = 0;
    uint16_t    *maskRegAddr;

    C1CTRL1bits.WIN = 1;

    // Se obtiene la diereccion del registro CiRXMmSID para mascara m
    maskRegAddr = ( uint16_t * ) ( &C1RXM0SID + (m << 1) );



      // Filter Standard Identifier
        sid10_0 = ( identifier & 0x7FF );

            *maskRegAddr = ( sid10_0 ) << 5; // Escribir en CiRXMnSID Register

        *( maskRegAddr + 1 ) = 0;            // Escribir en CiRXMnEID Register


    C1CTRL1bits.WIN = 0;
}

/* Configuracion de los buffer de mensajes
    Inputs:     buf: numero de buffer
                txId: SID del mensaje
                remoteTransmit: mensaje es normal o remoto*/
void Ecan1WriteTxMsgBufId( uint16_t buf, int32_t txIdentifier,
        uint16_t remoteTransmit )
{
```

Figura 83. Configuración ECAN en MPLAB_2

```
    uint32_t    word0 = 0;

    uint32_t    word1 = 0;

    uint32_t    word2 = 0;
    uint32_t    sid10_0 = 0;
    uint32_t    eid5_0 = 0;
    uint32_t    eid17_6 = 0;


    sid10_0 = ( txIdentifier & 0x7FF );


    if( remoteTransmit == 1 )
    {   // Transmit Remote Frame
        word0 = ( (sid10_0 << 2) | ide | 0x2 );
        word2 = ( (eid5_0 << 10) | 0x0200 );
    }
    else
    {
        word0 = ( (sid10_0 << 2) | ide );
        word2 = ( eid5_0 << 10 );
    }

    // Obtain the Address of Transmit Buffer in DMA RAM for
    // a given Transmit Buffer number


    ecan1msgBuf[buf][0] = word0;


    ecan1msgBuf[buf][1] = word1;
    ecan1msgBuf[buf][2] = word2;
}

/* Transmision de datos CAN
    Input:  buf: numero de buffer
            dataLength: numero de bytes de datos
            datos: data1, data2, data3, data4*/
void Ecan1WriteTxMsgBufData( uint16_t buf, uint16_t dataLength, uint16_t data1,
        uint16_t data2, uint16_t data3,uint16_t data4 )
{
    ecan1msgBuf[buf][2] = ( (ecan1msgBuf[buf][2] & 0xFFF0) + dataLength );

    ecan1msgBuf[buf][3] = data1;
    ecan1msgBuf[buf][4] = data2;
```

Figura 84. Configuración ECAN en MPLAB_3

```
    ecan1msgBuf[buf][5] = data3;
    ecan1msgBuf[buf][6] = data4;
}

/* Deshabilitar filtro n
    Input:  n: numero de filtro*/
void Ecan1DisableRXFilter( int16_t n )
{
    uint16_t    *fltEnRegAddr;
    C1CTRL1bits.WIN = 1;
    fltEnRegAddr = ( uint16_t * ) ( &C1FEN1 );
    *fltEnRegAddr = ( *fltEnRegAddr ) & ( 0xFFFF - (0x1 << n) );
    C1CTRL1bits.WIN = 0;
}
```

Figura 85.. Configuración ECAN en MPLAB_4

Inicialización ECAN

```
#define CAN_prog_500KHZ

#include <xc.h>
#include "ecan1_config.h"

void DMA0Init( void ) // Inicializacion DMA0 para transmitir
{
    DMAPWC = 0;
    DMARQC = 0;
    DMA0CON = 0x2020;
    DMA0PAD = ( int ) &C1TXD;   /* ECAN 1 (C1TXD) */
    DMA0CNT = 0x0007;
    DMA0REQ = 0x0046;           /* ECAN 1 Transmit */

    DMA0STAL = (uint16_t)(&ecan1msgBuf);
    DMA0STAH = 0;

    DMA0CONbits.CHEN = 1;
}

void DMA2Init( void ) // Inicializacion DMA2 para recibir
{
    DMAPWC = 0;
    DMARQC = 0;
    DMA2CON = 0x0020;
    DMA2PAD = (volatile unsigned int) &C1RXD;   /* ECAN 1 (C1RXD) */
    DMA2CNT = 0x0007;
    DMA2REQ = 0b00100010;           /* ECAN 1 Receive */

    DMA2STAL = __builtin_dmaoffset(&ecan1msgBuf);
    DMA2STAH = 0x0000;

    DMA2CONbits.CHEN = 1;
}


void Ecan1ClkInit( int can_speed ) /* Inicializacion del reloj
                                    * can_speed: 500 o 250
{
    /* FCAN se selecciona como FCY */
    C1CTRL1bits.CANCKS = 0x0;

    /* Bit Time = (Sync Segment + Propagation Delay + Phase Segment 1 +
       + Phase Segment 2) = 14*TQ
       Phase Segment 1 = 6TQ
       Phase Segment 2 = 6Tq
```

Figura 86. Funciones inicialización ECAN en MPLAB_1

```
       Propagation Delay = 1Tq
       Sync Segment = 1TQ
       CiCFG1<BRP> =(FCAN /(2 × N × FBAUD))□ 1 */

    C1CFG2 = 0x1A8;
    //BRP TQ = (2 x 5)/FCAN; SJW 1 x TQ;
    if (can_speed==500)
        C1CFG1 = 0x05;//1Mhz=0x2
    else if (can_speed==250)
        C1CFG1 = 0x0B;
    else C1CFG1 = 0x02; //1MHz
}


void Ecan1Init( int canspeed ) /* Inicializacion de can y configuracion
                                * de filtros y mascaras
{
    /* Modo configuracion  */
    C1CTRL1bits.REQOP = 4;
    while( C1CTRL1bits.OPMODE != 4 );

    Ecan1ClkInit(canspeed);

    C1FCTRLbits.FSA = 2;           /* FIFO empieza en buffer 2 */
    C1FCTRLbits.DMABS = 0b000;     /* 4 CAN buffer de mensajes in DMA RAM */

    /*    Configuracion de filtro y mascara  */
    int32_t id1=0x1A0,mask1=0x1FFFFFF0; //solo mensajes 1Ax (hex))

    Ecan1WriteRxAcptFilter( 1, id1, 0, 1, 1 );//standard 0x1FF
    Ecan1WriteRxAcptMask( 1, mask1, 1, 0 );

    /* Modo Normal  */
    C1CTRL1bits.REQOP = 0;
    while( C1CTRL1bits.OPMODE != 0 );

    /* ECAN transmit/receive message control */
    C1RXFUL1 = C1RXFUL2 = C1RXOVF1 = C1RXOVF2 = 0x0000;
    C1TR01CONbits.TXEN0 = 1;        /* ECAN1, Buffer 0 para transmitir */
    C1TR01CONbits.TXEN1 = 0;        /* ECAN1, Buffer 1 para recibir */
    C1TR01CONbits.TX0PRI = 0b11;    /* Message Buffer 0 prioridad */
    C1TR01CONbits.TX1PRI = 0b11;    /* Message Buffer 1 prioridad */

    /* Configuracion I/O pins */
```

Figura 87. Funciones inicialización ECAN en MPLAB_2

```
    RPOR1bits.RP36R = 0x000E;     //RB4->ECAN1:C1TX
    RPINR26bits.C1RXR = 0x0018;     //RA8->ECAN1:C1RX   CAN RX a pin RPI24!

}
```

Figura 88. Funciones inicialización ECAN en MPLAB_3

Configuración módulo QEI

```
#include <p33EP512GM710.h>


QEI1CONbits.QUEIEN=0;    // Deshabilita modulo para config
QEI1CONbits.QEISIDL=0;   // Funciona en modo Idle
QEI1CONbits.PIMOD=5;     // Contador resetea al llegar a QEI1GEC
QEI1GECH=0;              // Reset al llegar a 256 pulsos
QEI1GECL=256;            // encoder tiene 128 ranuras (64pulsos)

RPINR14bits.QEA1R=0b1111100; // Entrada QEA1 en RPI124
RPINR14bits.QEB1R=0b1111011; // Entrada QEb1 en RPI123

QEI1CONbits.QEIEN=;      // Habilita el módulo
```

Figura 89. Configuración módulo QEI

Código archivo main

```
#include "xc.h"
#include "main.h"
#include "ADC1CON.h"
#include "ECAN_conf.h"
#include "PWMCON.h"
#include "QEICON.h"
#include "TMR1CON.h"

//Declaracion de funciones
void RxECAN1( mID *message );
void Ecan1WriteMessage_Data();
void Ecan1WriteMessage_Warning()
//Declaracion de variables

float ref_Vel=0;        // Referencia de velocidad
float sens_Vel;         // Valor velocidad real
float sens_Int;         // Valor sensor de intensidad
float error=0;          // Error
float errork=0;         // Acumulación de errores
float ts=0.0002;        // Periodo de muestreo
float u;                // Accion del control
float kp=3;             // Constante proporcional
float ki=0.1;           // Constante integral
float cons_PV=3.61;     // Constante par-velocidad
float salidaControl;    // Salida de la etapa de control
float salidaControlk=0; // Acumulacion para integral
float posicion;
float intControl;
float salidaA, salidaB, salidaC;
int warning=0;
float dutyA=0, dutyB=0, dutyC=0;


int main(void) {
    WATCHDOG_TimerSoftwareDisable();// Deshabilita watchdog
    SYSTEM_Initialize();           // Inicializa dispositivo

/*  Inicializacion ECAN1    */
    int can_bus_speed=500;//500-> 500khz
    Ecan1Init(can_bus_speed);
    DMA0Init();
    DMA2Init();
    /* Habilira interrupcciones ECAN1  */
    IEC2bits.C1IE = 1;
```

Figura 90. Código main_1

```
    C1INTEbits.TBIE = 1;
    C1INTEbits.RBIE = 1;
    IEC0bits.T1IE = 1; // Habilitar interrupccion del Timer1
    T1CONbits.TON = 1; // Encendido del Timer

while (1){
    void RxECAN1();     // Funcion de lectura CAN
}

void __attribute__((__interrupt__, no_auto_psv)) _Lazocontrol(void)
{
    sens_Int = lee_adc(0);           // Se asigna el valor del converidor ADC
    sens_Vel = VEL1CNT*360/(ts*256); /* Se asigna el valor del converidor ADC
                                        y adecuan unidades pulsos> deg/s */

    if (sens_Int<150){                    // Comprobacin límite intensidad
        error = ref_Vel-sens_Vel;         // Calculo de error actual
        errork+=error;                    // Acumulacion error para integral
        u=kp*error+ki*ts*errork;              // Acción del control

        if (u>300){                       // Se aplica limitador
            u=300;
        }

        salidaControl=u*cons_PV+sensVel;  // Salida=u*k+vel real

 /// Control V/f
        salidaControlk+=salidaControl;
        intControl=ts*salidaControlK*2*3.141592/360;  /* Calculo (posicion)en
                                                         * radianes*/
        senA=sin(intControl);                         // Seno salida control
        senB=sin(intControl+2/3*3.141592);            // seno desfassado 120°
        salidaA=senA*salidaControl*4.514E-7;   /* Obtienen salidas para las
                                                  * 3 fases (sin compensacion)*/
        salidaB=senA*salidaControl*4.514E-7;

// Compensacion
        if (sens_Vel<=10*360){                 // Ayuda si velocidad<10Hz
            ayudaA=sens_Vel*0.0075*senA;       // Calculo de ayudas
            ayudaB=sens_Vel*0.0075*senB;
        }
        else {
            ayudaA=ayudaB=0;
        }

// Modulantes
```

Figura 91. Código main_2

```
        modA=salidaA+ayudaA;                    // Calculo de modulantes
        modB=salidaB+ayudaB;
        modC=-modA-modB-ayudaA-ayudaB;

        Ecan1WriteMessage_Data();       //Envia valores de velocidad y
                                        //corriente actuales

    }
    else {
        modA=modB=modC=0;               // Modulantes a cero para apagar PWM
        errork=salidaControlk=0;        // Variables de acumulacion a 0
        T1CONbits.TON = 0;              // Timer1 OFF para parar el ciclo
        Ecan1WriteMessage_Warning();    // Envio mensaje de alarma
    }
    // Ciclo trabajo para el siguiente periodo

    PDC1=(0.5+modA*0.5)*65535;
    PDC2=(0.5+modB*0.5)*65535;
    PDC3=(0.5+modC*0.5)*65535;

    IFS0bits.T1IF = 0; // Se limpia el flag de interrupcion
}



uint16_t lee_adc (ADC1_CHANNEL canal)
{
    AD1CHS0 = canal;
    AD1CON1bits.SAMP = 1;       // Start sampling
    delay_us(2);               // Wait for sampling time (2 us)
    AD1CON1bits.SAMP = 0;       // Start the conversion
    while (!AD1CON1bits.DONE);  // Wait for the conversion to complete
    return ADC1BUF0;            //Read the ADC conversion result
}


void RxECAN1( mID *message ) //funcion lectura CAN
{
    unsigned int    ide = 0;
    unsigned int    srr = 0;
    unsigned long   id = 0;

    /* Se identifica el tipo de mensaje */
    ide = ecan1msgBuf[message->buffer][0] & 0x0001;
    srr = ecan1msgBuf[message->buffer][0] & 0x0002;
```

Figura 92. Código main_3

```
/* Mensaje con SID */
if( ide == 0 )
{
    message->id = ( ecan1msgBuf[message->buffer][0] & 0x1FFC ) >> 2;
    message->frame_type = CAN_FRAME_STD;
}

/* Mensaje con EID */
else
{
    id = ecan1msgBuf[message->buffer][0] & 0x1FFC;
    message->id = id << 16;
    id = ecan1msgBuf[message->buffer][1] & 0x0FFF;
    message->id = message->id + ( id << 6 );
    id = ( ecan1msgBuf[message->buffer][2] & 0xFC00 ) >> 10;
    message->id = message->id + id;
    message->frame_type = CAN_FRAME_EXT;
}

/* Comprobar tipo de mensaje */
/* Mensaje RTR */
if( srr == 1 )
{
    message->message_type = CAN_MSG_RTR;
}

/* Mensaje normal  */
else
{
    message->message_type = CAN_MSG_DATA;
    message->data[0]=(unsigned char) ecan1msgBuf[message->buffer][3];
    message->data[1]=(unsigned char) ((ecan1msgBuf[message->buffer][3]
                                       & 0xFF00) >> 8 );
    message->data[2]=(unsigned char) ecan1msgBuf[message->buffer][4];
    message->data[3]=(unsigned char) ((ecan1msgBuf[message->buffer][4]
                                       & 0xFF00) >> 8 );
    message->data[4]=(unsigned char) ecan1msgBuf[message->buffer][5];
    message->data[5]=(unsigned char) ((ecan1msgBuf[message->buffer][5]
                                       & 0xFF00) >> 8 );
    message->data[6]=(unsigned char) ecan1msgBuf[message->buffer][6];
    message->data[7]=(unsigned char) ((ecan1msgBuf[message->buffer][6]
                                       & 0xFF00) >> 8 );
    message->data_length=(unsigned char) (ecan1msgBuf[message->buffer][2]
                                          & 0x000F );
}
```

Figura 93. Código main_4

```
    switch (id)
      {
         case 1:    /* Actualizacion de la referencia de velocidad */
            refVel = ecan1msgBuf[message->buffer][3];
            break;
        case 2:     /* Parada del sistema*/
            T1CONbits.TON = 0;          // Apagado del Timer1
            errork=salidaControlk=0;    // Variables de acumulacion a 0
            break;
        case 3:     /* Inicio del sistema (tras parada)*/
            T1CONbits.TON = 1; // Encendido del Timer1
            break;
        default:
            break;
}

void Ecan1WriteMessage_Data()//Envia valores de velocidad y corriente actuales
{
    uint16_t w1=(uint16_t) (sens_Vel), w2=(uint16_t) (sens_Int), w3=0, w4=0;
    Ecan1WriteTxMsgBufId( 0, 0x1BC, 1, 0 );
    Ecan1WriteTxMsgBufData( 0, 4, w1, w2, w3, w4 );
    C1TR01CONbits.TXREQ0 = 1;
}
void Ecan1WriteMessage_Warning()//Aviso de fallo
{
    Ecan1WriteTxMsgBufId( 0, 0x1BB, 1, 0 );
    Ecan1WriteTxMsgBufData( 0, 8, 10, 10, 10, 10 );
    C1TR01CONbits.TXREQ0 = 1;
    delay_us(1);
    Ecan1WriteTxMsgBufData( 0, 8, 10, 10, 10, 10 );
    C1TR01CONbits.TXREQ0 = 1;
}
```

Figura 94.. Código main_5

8.1 Listado de datasheets

- Módulo IGBTs SKM100GB12T4:
  https://docs.rs-online.com/9aa4/0900766b80daac4b.pdf

- JACK Alimentación CON-SOCJ-2155:
  https://www.mouser.es/ProductDetail/Gravitech/CON-SOCJ-2155?qs=fkzBJ5HM%252BdCcpvFQyQZHtA%3D%3D

- Bobina modo común SC-10-30J:
  https://www.mouser.es/datasheet/2/212/1/KEM_LF0006_SC-J-1104520.pdf

- Convertido DC/DC TRACO-TEL5-1210:
  https://www.mouser.es/datasheet/2/687/tel5-519367.pdf

- Oscilador Q33519E40003912:
  https://es.rs-online.com/web/p/osciladores-crystal/6676527/

- IGBT Driver UCC21540:
  https://www.ti.com/lit/ds/symlink/ucc21540.pdf?HQS=TI-null-null-mousermode-df-pf-null-wwe&DCM=yes&ref_url=https%3A%2F%2Fwww.mouser.es%2F&distId=26

- Diodo CGRM4001-G:
  https://www.mouser.es/datasheet/2/80/CGRM4001-G%20Thru.%20CGRM4007-G%20RevD-476722.pdf

- Acondicionamiento sensor de corriente Si8920:
  https://www.silabs.com/documents/public/data-sheets/si8920-datasheet.pdf

- Convertidor DC/DC IML0212D03:
  https://www.mouser.es/datasheet/2/942/SF_IML02-1508758.pdf

- Referencia voltaje MAX6018AEUR16: https://www.mouser.es/datasheet/2/256/MAX6018-MAX6018B-1515781.pdf

- Transceptor CAN MAX3051: https://datasheets.maximintegrated.com/en/ds/MAX3051.pdf

- Supresor voltaje ESDCAN02-2BWY: https://www.mouser.es/datasheet/2/389/dm00551472-1799515.pdf

- Microcontrolador DSPIC33EP512GM710: https://ww1.microchip.com/downloads/en/DeviceDoc/70000689d.pdf