Trabajo de Fin de Grado
Ingeniería en Tecnologías Industriales

Diseño y evaluación mediante técnicas Processor in the Loop de la operación de un convertidor de potencia conectado a la red eléctrica

Autor: Lorenzo Prats Caparrós
Tutor: Sergio Vázquez Pérez

Departamento de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Proyecto Fin de Carrera
Ingeniería en Tecnologías Industriales

Diseño y evaluación mediante técnicas Processor in the Loop de la operación de un convertidor de potencia conectado a la red eléctrica

Autor:
Lorenzo Prats Caparrós

Tutor:
Sergio Vázquez Pérez
Profesor titular

Departamento de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2024

Proyecto Fin de Carrera: Diseño y evaluación mediante técnicas Processor in the Loop de la operación de un convertidor de potencia conectado a la red eléctrica

Autor: Lorenzo Prats Caparrós

Tutor: Sergio Vázquez Pérez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2024

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Mis agradecimientos quiero que vayan dedicados a todas aquellas personas que me han apoyado a lo largo de estos años, puesto que cada uno ha aportado su granito de arena para hacer de mí la persona que soy. En especial quiero dedicárselos:

A mi familia, por estar en las buenas y en las malas, educarme y hacerme entender que todo esfuerzo tiene su recompensa, el grado es una "carrera de fondo", donde lo importante es aprender y formarse para poder ser buen ingeniero y persona.

A Almudena, por ser la persona con la que más he compartido estos últimos años, por su paciencia, por ayudarme en lo posible, y por ser mi refugio en las épocas de estrés.

A mis amigos de la infancia y a los que he ido haciendo durante estos años, por tomarse la vida con humor y demostrando que la vida consiste en algo más que en sentarse todo el día con la mirada puesta en un papel.

A los docentes de la Escuela, puesto que son quienes me han ido guiando y orientando a lo largo de esta etapa, enseñando lo necesario para poder ser un buen ingeniero.

Lorenzo Prats Caparrós

Sevilla, 2024

Resumen

En este Trabajo de Fin de Grado se trata el uso de Processor In The Loop (PIL) con el software de PLECS para validación de algoritmos de control en el campo de la Electrónica de Potencia. En concreto, consiste en la implementación del control de corriente de un convertidor de potencia conectado a la red. Se verá el funcionamiento de los elementos de dicho convertidor, siendo necesario transformar las tensiones y las corrientes que provienen de la red eléctrica a ejes síncronos dq para facilitar los cálculos y ejecución del modelo simulado, aplicando para ello las transformadas de Clarke y Park.

Para la realización de este trabajo se ha empleado el software PLECS para la simulación del sistema, además del uso del microcontrolador C2000 TMS320F28069 de Texas Instruments, y el software de Code Composer Studio (CCS) para su programación.

Se hará por tanto la implementación en PLECS del convertidor de potencia conectado a la red, así como del bloque de control de corriente con las transformaciones necesarias y del modulador PWM encargado de gestionar las señales de disparo de los IGBTs del convertidor, para posteriormente usar el microcontrolador que es quien realizará PIL para esta aplicación.

Abstract

This Final Degree Project deals with the use of Processor In The Loop (PIL) with the PLECS software for validation of control algorithms in the field of Power Electronics. Specifically, it consists of the implementation of current control of a power converter connected to the grid. The operation of the elements of said converter will be seen, being necessary to transfer the voltages and currents that come from the electrical network to synchronous dq axes to facilitate the calculations and execution of the simulated model, applying the Clarke and Park transforms.

To carry out this work, the PLECS software has been used to simulate the system, in addition to the use of the C2000 TMS320F28069 microcontroller from Texas Instruments, and the Code Composer Studio (CCS) software for its programming.

Therefore, the implementation in PLECS of the power converter connected to the grid will be carried out, as well as the current control block with the necessary transformations and the PWM modulator in charge of managing the trigger signals of the IGBTs of the converter, to later use the microcontroller which is the one who will perform PIL for this application.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

CC	Corriente Continua
CA	Corriente Alterna
V	Voltios
A	Amperios
Ω	Ohmios
W	Watios
var	Voltiamperio radioactivo
cos	Función coseno
sin	Función seno
tg	Función tangente
arctg	Función arco tangente
θ	Theta
∞	Infinito
X^*	Magnitud X de referencia
H	Henrios (unidad de inductancia)
Hz	Herzios
s	segundos
μ	"mu" o "micro" (magnitud por 10^{-6})
$\partial y\ \partial x$	Derivada parcial de *y* respecto
Xº	Notación de grado, X grados.
<	Menor o igual
>	Mayor o igual

1 INTRODUCCIÓN

"La mitad de lo que separa a los emprendedores exitosos de los que no lo son es la perseverancia".

- Steve Jobs -

La Electrónica de Potencia ha experimentado un importante crecimiento en los últimos años, sobre todo debido al desarrollo de las nuevas tecnologías, que han permitido la fabricación e integración de nuevos dispositivos y semiconductores, capaces de aumentar la velocidad de los sistemas para los cuales se necesitan, a la vez de una notable mejora en la eficiencia de los mismos.

Se puede definir por tanto la Electrónica de Potencia como una rama de la Electrónica encargada del estudio de los dispositivos, circuitos y sistemas electrónicos que transforman y controlan la energía eléctrica, de continua a alterna o viceversa. El objetivo es manejar y transformar la energía de forma eficiente, por lo que se justifica el uso mayormente de bobinas y condensadores, así también como semiconductores tales como el SCR, tiristores o IGBT, y evitando el uso de elementos resistivos, ya que son los que más pérdidas producen.

El origen de la Electrónica de Potencia se remonta al año 1900, con la introducción del rectificador de arco de mercurio. Hasta la década de los años 50 se fueron introduciendo nuevos elementos y dispositivos como el ignitrón, el tiratrón o el fanotrón, que se aplicaban para el control de potencia, pero no fue hasta 1948 que se inventó el primer transistor de silicio en los Bell Telephone Laboratories, comenzando así una revolución en la industria de la Electrónica. También en estos laboratorios se inventó años más tarde el tiristor (SCR).

La segunda revolución electrónica se dio en 1958 con el desarrollo del tiristor comercial, lo que supuso el comienzo de una nueva era para la Electrónica de Potencia. Posteriormente, con la invención del MOSFET por Kahng y Atalla en 1959, y del IGBT en 1980 por Hans W. Becke y Carl F. Wheatley, se abrió la puerta a la creación de circuitos electrónicos más pequeños, con mayores velocidades de conmutación y por supuesto, más potentes y capaces de trabajar con mayor capacidad de soportar voltajes y corrientes mayores. [1]

Se introduce en el ámbito de la Electrónica de Potencia el concepto de interruptor ideal, que puede definirse como "*aquel que no presenta pérdidas en ninguno de sus dos estados y es capaz de conmutar de un estado a otro de forma instantánea"*. [2] Estos dos estados que se mencionan son el de interruptor abierto o cerrado, no permitiendo el primero el paso de corriente, mientras que el segundo sí que permite la circulación de ésta.

Un interruptor ideal en estado de abierto es aquel que presenta una una $R_{off} = \infty, v_{off} = \infty \; y \; T_{on} = 0$, mientras que el interruptor ideal cerrado presenta $R_{on} = 0$, $i_{on} = \infty \; y \; T_{off} = 0$. Los tiempos T_{off} y T_{on} son aquellos tiempos en los que el interruptor tarda en pasar de cerrado a abierto, o de abierto a cerrado, respectivamente. [2]

Con estos conceptos como base, se comprenderán mejor los apartados descritos a continuación.

1.1. IGBT

Tal y como se verá más adelante, el convertidor de potencia utilizado basa su funcionamiento en las señales de disparo de los IGBTs que lo conforman. Es por ello, que se dedica esta parte del primer capítulo para definir qué son y cómo funcionan estos dispositivos.

Introducido en los años 80, este dispositivo que podemos definir como una combinación entre los MOSFET y los BJT, capaz de mejorar a los dos anteriores en cuanto a características, y que consta de tres partes (colector, puerta y emisor), surgió como una necesidad de obtener un transistor más potente y eficiente al no ser capaces los MOSFET de aquella época de trabajar eficazmente con altos voltajes. [3]

Como ya se ha mencionado, combina las características del BJT y del MOSFET, pudiendo destacar del IGBT que tiene la entrada como el MOS, y la salida de un BJT. Si lo comparamos con un MOS, tiene tensiones y corrientes bastante mayores (1700V – 400A). En cambio, la frecuencia de conmutación es menor (se sitúa entre la del MOS y la del BJT). En cuanto a la geometría y dopado, es análogo al MOS, con la diferencia de tener una capa n^- más ancha y menos dopada. [4]

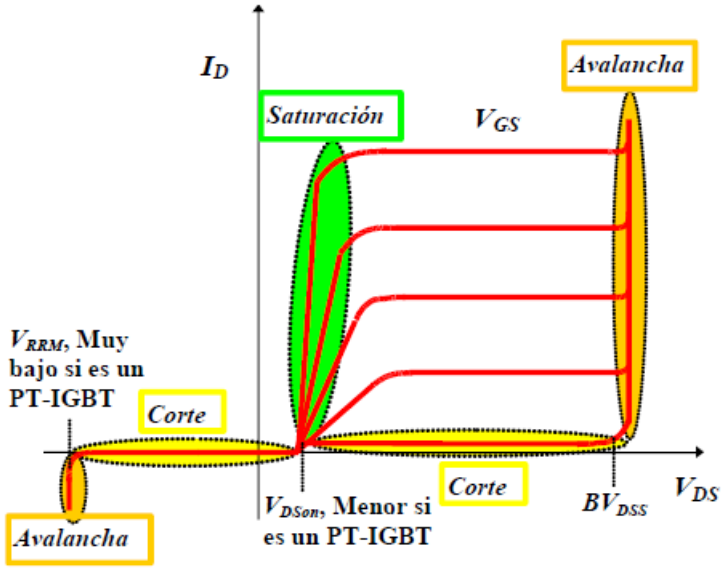

Figura 1. Curva característica estática de un transistor canal n

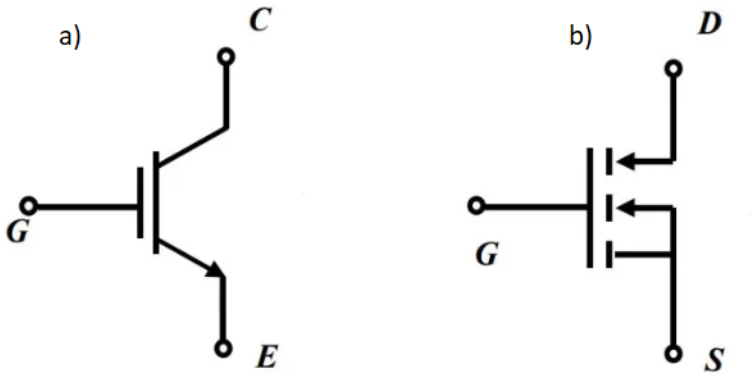

Figura 2. Representación simbólica del IGBT: a) Como BJT b) Como MOSFET

1.2. Convertidor de potencia

Dentro de la Electrónica de Potencia, un elemento fundamental es el convertidor de potencia, que se puede definir como un dispositivo que, gracias a su circuitería interna, es capaz de convertir el tipo de energía eléctrica según las necesidades o el uso para el cual se ha instalado. Los convertidores de potencia están presentes en multitud de procesos industriales, siendo esenciales para todos aquellos procesos que requieran del uso de la corriente eléctrica.

Como ya se ha mencionado anteriormente, su principal uso es el de convertir la corriente de alterna a continua, o viceversa, ya que muchos circuitos electrónicos están alimentados en continua, mientras que otros tantos obtienen la energía de la red.

Además de convertir la energía de una forma a otra, estos dispositivos permiten también controlar la velocidad de diferentes máquinas, motores y generadores eléctricos. En el sector industrial es habitual modificar las características de la tensión y la corriente para el uso específico de cada proceso, por lo que permiten transformar la potencia eléctrica obtenida de la red para su uso en un aparato o máquina con unas características y condiciones específicas

Estos dispositivos cuentan además con sistemas de seguridad y de protección, haciendo frente a sobrecargas y sobrecalentamientos, y garantizando en todo momento una completa seguridad de los procesos para los cuales son requeridos. [5]

El convertidor que se utilizará en este proyecto es un convertidor trifásico de dos niveles, o lo que es lo mismo, un puente completo trifásico conformado por IGBTs.

1.2.1. Tipos de convertidores

Para entender los tipos de convertidores que hay, es necesario antes diferenciar entre corriente alterna y corriente continua:

La corriente continua (CC) se da cuando los electrones circulan en una sola dirección. Se le atribuye su descubrimiento a Thomas Edison, quien descubrió la posibilidad de generar corriente eléctrica a través de un material metálico conductor, que repelía los electrones por un lado y los atraía por el otro. Al contrario que en la corriente continua, en la corriente alterna (CA) la electricidad que circula puede llevar varios sentidos. El objetivo de ésta es transportar la mayor cantidad de energía a una larga distancia. Su descubridor fue el científico austriaco Nikola Tesla.

Una vez expuestos los conceptos y diferencias de la corriente continua y alterna, se puede entender con mayor facilidad la necesidad de tener los siguientes cuatro tipos de conversión de energía:

- Rectificadores: convierten la potencia de CA a CC y son unos de los más habituales debido a la cantidad de dispositivos que se encuentran en nuestras casas, y que necesitan de la corriente continua para funcionar o cargarse. Basa su funcionamiento a una serie de diodos que se encargan de transformar la onda sinusoidal de la corriente alterna a una serie de picos positivos, y ajustando posteriormente la corriente de la fuente a una tensión más manejable.
- Inversores: menos habituales que los anteriores. Transforman la corriente continua a corriente alterna. Algunas de sus aplicaciones son la carga de baterías de los vehículos o accionamiento de motores de CA de baja potencia.
- Convertidores CC a CC (choppers): toman una tensión CC a la entrada y entregan una tensión CC diferente, necesarios si la tensión de salida es superior o inferior a la de entrada.
- Convertidores CA a CA (matriciales): convierten una forma de onda de CA a otra con distinta tensión y frecuencia.

[5]

1.3. Transformadas de Clarke y Park

El objeto de este Trabajo de Fin de Grado se basa en el control de corriente para el convertidor trifásico de dos niveles conectado a la red eléctrica. Este control de corriente es algo bastante importante para conseguir unos buenos niveles de funcionamiento del sistema. Para ello, es necesario conocer el concepto de las transformadas de Clarke y Park, ya que nos permiten simplificar y realizar los cálculos de manera más sencilla, permitiéndonos trabajar en los ejes síncronos, facilitando así también el análisis del sistema trifásico. Lo que se pretende con el uso de estas transformadas es pasar de las variables sinusoidales que obtenemos de la red eléctrica a unos vectores de componentes continuas.

1.3.1. Transformada de Clarke

La transformada de Clarke consiste en pasar de magnitudes en ejes naturales *abc* a ejes estacionarios *αβ*, es decir, nos permite reducir el número de componentes del sistema. Con esto, pasamos del sistema trifásico de tres hilos a trabajar con dos componentes, ya que en el sistema de tres hilos las componentes no son linealmente independientes (esto es, conociendo dos de ellas podemos llegar a obtener la tercera).

La ecuación que modela el cambio de unos ejes a otros es la siguiente

$$v_{\alpha\beta} = T_{abc}^{\alpha\beta} v_{abc} \qquad (1.1.)$$

Donde

$$T_{abc}^{\alpha\beta} = k_{\alpha\beta} * \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \qquad (1.2.)$$

La constante $k_{\alpha\beta}$ puede ser de 2 tipos:

"Power invariant"	"Non – power invariant"
$k_{\alpha\beta} = \sqrt{2/3}$	$k_{\alpha\beta} = 2/3$

La constante usada para el caso que atañe a este trabajo es la constante "Power invariant"

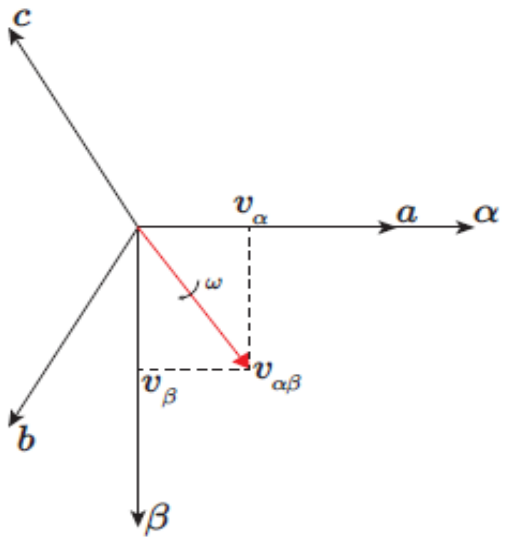

Figura 3. Representación de los ejes de la Transformada de Clarke

1.3.2. Transformada de Park

La transformada de Park permite trabajar con magnitudes continuas. Esto es, se usan los ejes síncronos dq, que giran solidariamente con la tension de la red a una velocidad ω, y que tiene una sola componente (los términos multiplicados por v_q son 0) y que ocupan la posición angular θ = ωt a lo largo del tiempo.

La ecuación que modela esta transformación es la siguiente

$$v_{dq} = T_{\alpha\beta}^{dq} v_{\alpha\beta} \quad (1.3.)$$

Donde

$$T_{\alpha\beta}^{dq} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \quad (1.4.)$$

Donde θ es la fase del vector de tensión de red en el plano $\alpha\beta$

Como se podrá observar en capítulos posteriores, se emplearán tanto las transformadas de Clarke y Park como sus anti - transformadas, según las magnitudes que se quieran obtener del modelo de simulación.

[6]

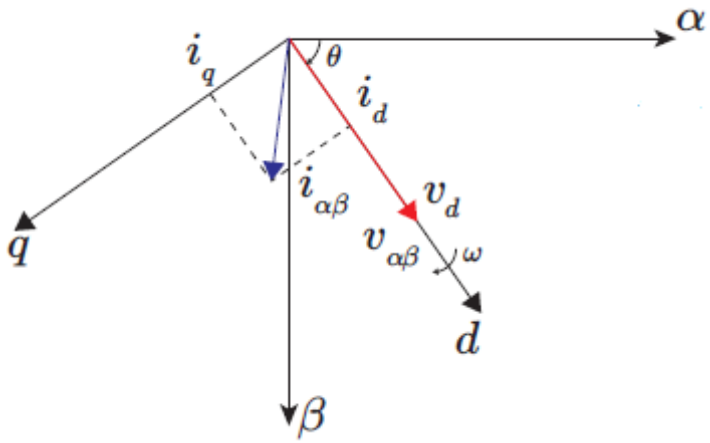

Figura 4. Representación de los ejes de la Transformada de Park

1.4. Punto de partida

En este Trabajo de Fin de Grado se empleará el convertidor de potencia trifásico de dos niveles conectado a la red. En la Figura 5. Convertidor de potencia conectado a la red se puede observer que la Fuente de energía removable (FER) genera cierta energía que se podrá usar posteriormente. La idea es poder mantener constante la tension del dc – link (v_{dc}), para lo cual es necesario el convertidor, que evacúa la energía suministrada a la red, ya que no se puede almacenar en el condensador para que, como se ha mencionado, se mantenga dicha tensión constante. Esto se consigue cumpliendo, además, con los criterios de calidad de la energía inyectada en la red, THD, suministro de reactiva, pérdidas, etc.

El objetivo es simular el sistema usando PLECS, que es un software desarrollado por Plexim que permite simular circuitos eléctricos, sobre todo en el ámbito de la electrónica de potencia. El motivo por el cual se elige esta herramienta de simulación es debido a que cuenta con un paquete de PIL, por lo que se nos permite conectar nuestro MCU al modelo de simulación y ejecutarlo en pseudo – tiempo real.

2 ESTADO DEL ARTE

"La tecnología moderna le debe a la ecología una disculpa".

-Alan M. Eddison-

Desde el principio de los tiempos, el ser humano ha necesitado de la energía para sobrevivir, para buscar el bienestar social y evolucionar, desde el descubrimiento del fuego hasta la obtención de las energías renovables. Se ha buscado siempre el mejor aprovechamiento del medio que nos rodea, lo cual con el paso del tiempo ha derivado en la necesidad de obtener dicha energía de manera limpia y sostenible, pero empecemos con un poco de contexto histórico.

A partir del sigo XVIII, con el comienzo de la Revolución Industrial, siendo un pilar fundamental el desarrollo de la máquina de vapor, el ser humano comienza un proceso de progreso constante en lo que se refiere a evolución de maquinaria y procesos industriales. Seguido de este suceso, comienzan a llegar los combustibles fósiles, como el carbón y el petróleo, usados para la combustión en varios procesos, y protagonistas de gran parte de la obtención de energía. Surgirían además nuevas formas de obtención y transformación de energía, como la transformación de energía eléctrica, de la ayuda del electromagnetismo, para obtener energía mecánica. Allá por el siglo XX se asentarían también las bases de la Energía Nuclear, con la primera fisión artificial del átomo de Uranio en 1938. [7]

Todo ello nos lleva a hoy día, donde se nos presenta un problema bastante importante: los combustibles fósiles no son ilimitados, y además de ello, emiten gran cantidad de gases contaminantes a la atmósfera, que pueden provocar efectos nocivos para la salud de la población y otras especies. A raíz de esto se están implantando cada vez más las energías renovables, que son una forma sostenible y limpia de obtener energía de la naturaleza.

La Unión Europea lanzó el objetivo de tener menos del 55% de emisiones de gases de efecto invernadero para 2030, comparándose con las emisiones de los mismos en el año 1990, y de al menos un 32% del consumo de energía debe proceder de energías renovables. [8] Además de esto, *"La Unión Europea (UE, en adelante) pretende ser neutra en términos climáticos de cara al año 2050. Es decir, la UE se ha fijado el objetivo de tener una economía con cero emisiones netas de gases de efecto invernadero"*. [9]

El presente Trabajo de Fin de Grado consiste, como bien ha podido observarse, en el control de corriente de un convertidor de potencia que va conectado a la red eléctrica. Entonces, ¿qué tiene esto que ver con lo comentado en este capítulo? Pues bien, el convertidor de potencia conectado a red tiene varias aplicaciones en el ámbito de la integración de energías renovables, como puede ser para la integración de energía eólica, puesto que es capaz de evacuar a la red la energía del giro de la turbina del generador, o también para la energía fotovoltaica, pudiendo transformar la corriente continua generada en los paneles PV a corriente alterna, que es la usada por la red eléctrica.

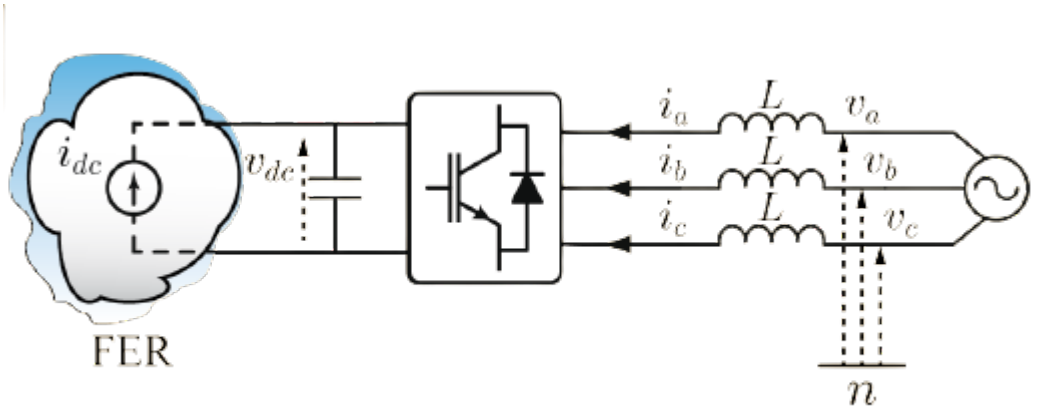

Figura 5. Convertidor de potencia conectado a la red

De esta forma, se entiende mejor la finalidad y el motivo del proyecto, pues es una forma de estudiar y comprender el convertidor de potencia como elemento esencial para la integración de este tipo de energía que tanto interesa hoy en día para mejorar la calidad de vida y asegurar un buen futuro, además de ahondar más en otras tecnologías, permitiendo más avances en el campo de la Electrónica de Potencia, y quien sabe, si futuros descubrimientos e invenciones.

3 MODELADO

El objetivo de este capítulo es exponer el modelado del sistema, así como proporcionar una breve explicación de los bloques y subsistemas que lo componen. Se pretende dar a conocer el funcionamiento básico y general del modelo de simulación bajo estudio para facilitar la comprensión de capítulos posteriores, en los cuales se desarrollará la implementación y los pasos necesarios a seguir para alcanzar un modelo que nos permita ejecutar la idea por la cuál se realiza este trabajo.

El proyecto se organiza en una estructura jerárquica, compuesta por bloques y subsistemas conectados entre sí. En este capítulo se desglosará dicha estructura desde el más alto nivel hasta los subsistemas de más bajo nivel, avanzando paso a paso por los subsistemas intermedios que componen dicha estructura, pretendiendo así asegurar una mayor comprensión del funcionamiento del sistema completo.

En el nivel superior de dicho modelo, encontramos tres grandes bloques a partir de los cuales se podrá desglosar y exponer el resto de las partes que lo conforman. Los tres bloques, por tanto, en los que se puede dividir este primer nivel son:

- Convertidor de potencia: como se ha mencionado anteriormente, se va a considerar el convertidor trifásico de dos niveles (o puente completo trifásico) formado por IGBTs.
- Control de Corriente: este bloque se va a encargar del control de corriente. Como se verá más adelante, necesita de las medidas de tensión y corriente de la red, así como unas referencias en las potencias activa y reactiva.
- Modulador PWM: este bloque se encarga de generar las señales de disparo de los IGBTs del convertidor mediante la comparación de una referencia de tensión con una portadora triangular.

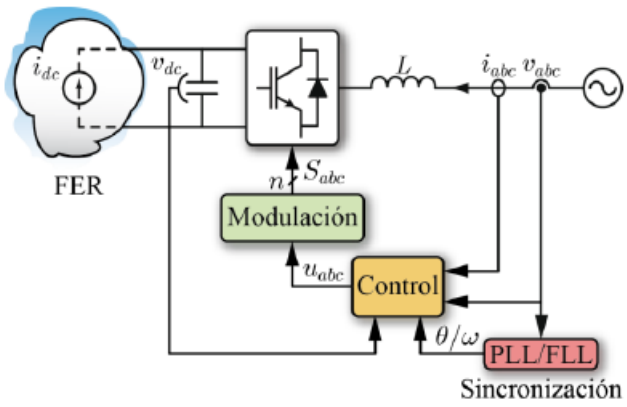

Figura 6. Esquema básico de un convertidor conectado a red y el diagrama de bloques para su control

3.1. Convertidor de potencia

Puesto que ya se hizo mención al convertidor de potencia en el capítulo anterior, nos centramos ahora en lo relacionado con este Proyecto. Como bien se puede observar en la siguiente figura, el convertidor utilizado es un convertidor trifásico de dos niveles conectado a la red. Está formado por seis transistores bipolares de puerta aislada, más comúnmente conocidos por sus siglas en inglés, IGBT, y una fuente de tensión continua de 700V.

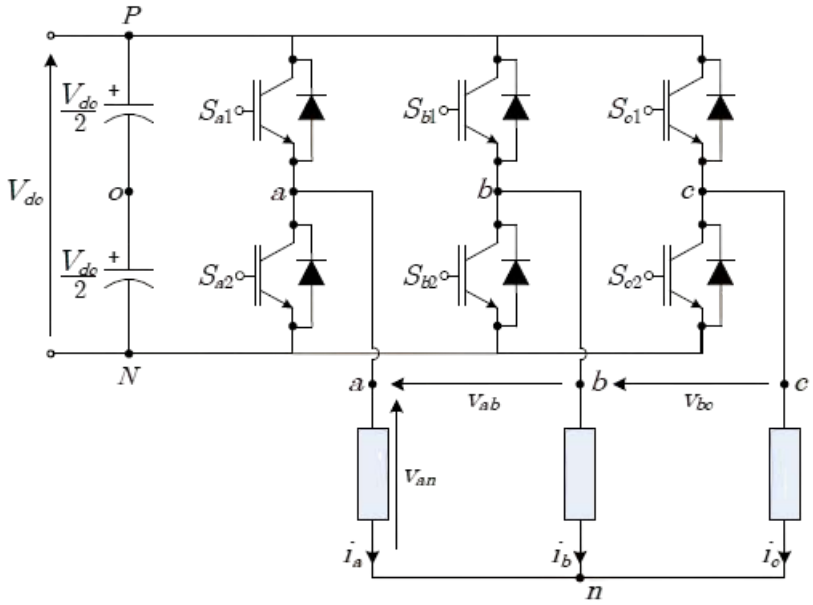

Figura 7. Convetidor trifásico de dos niveles

Asimismo, el convertidor de potencia implementado en el software de simulación quedaría de la siguiente manera:

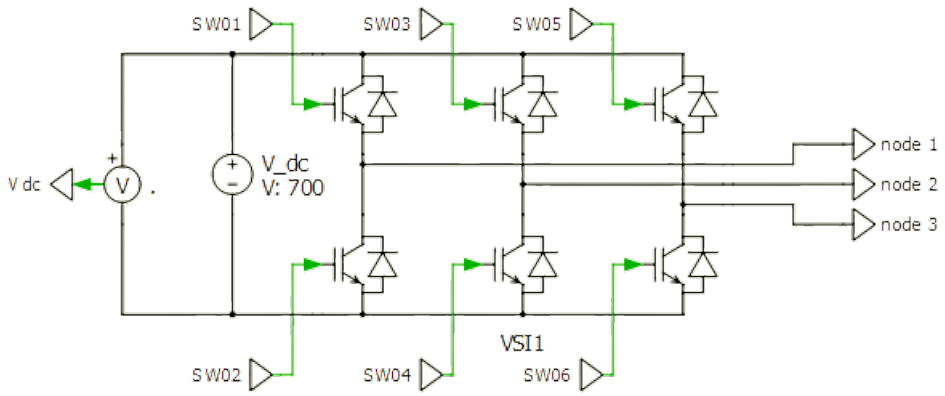

Figura 8. Convertidor de potencia implementado en PLECS

Los puntos entre cada pareja de transistores en serie (también llamados nodos) están conectados a un filtro RL de conexión a la red, cuyos valores son de R = 0.1Ω y de L = 5mH. Además de esto, como se aprecia en la siguiente figura, se tiene una fuente AC de 400V de amplitud rms fase-fase, 0º de fase y 50Hz, que hace las veces de red eléctrica.

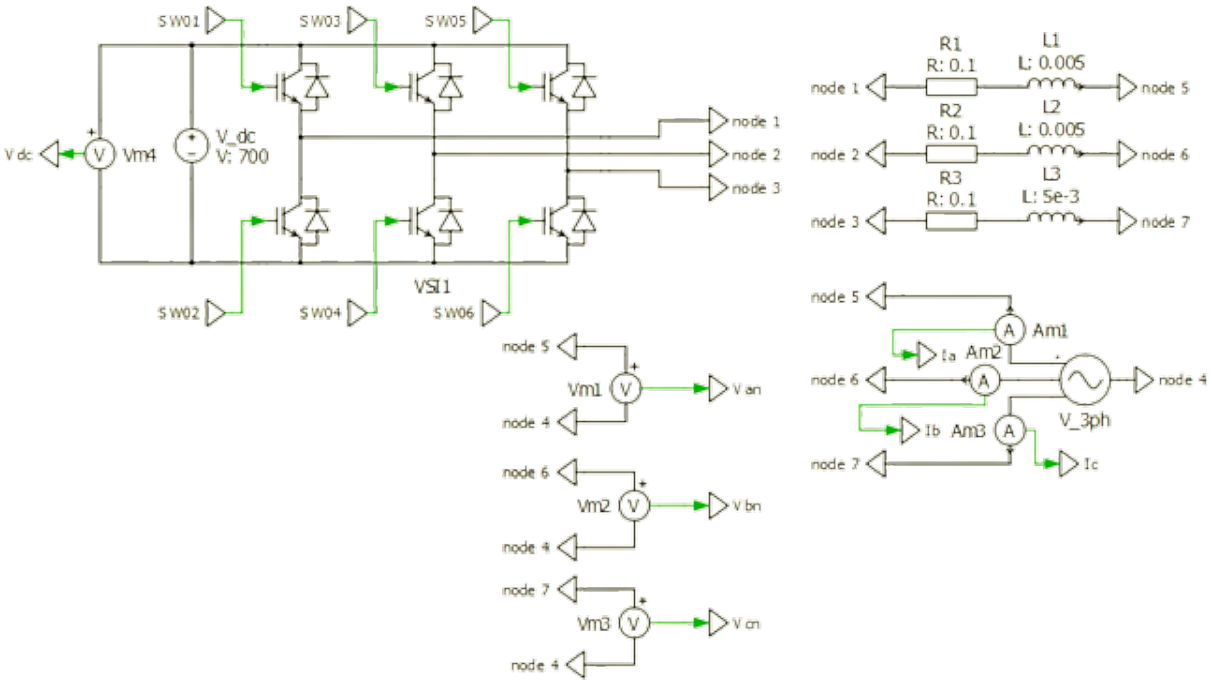

Figura 9. Esquema completo de convertidor de potencia conectado a red con filtro RL de conexión en PLECS

Para configurar la fuente trifásica que simula la red eléctrica, se debe escribir la amplitud como $400\sqrt{2}/\sqrt{3}$. Esto es así debido a que los 400 rms son de tensión de línea y se debe pasar la amplitud a tensión de fase (la fuente se configura en estrella, de ahí el dividir entre la $\sqrt{3}$), y multiplicar por $\sqrt{2}$ se debe a buscar que sea de pico.

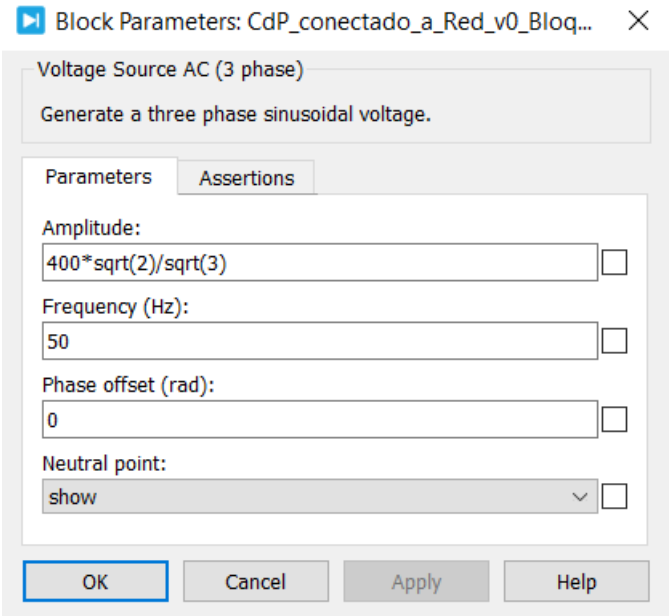

Figura 10. Configuración de la fuente AC trifásica

En este bloque es donde se miden, usando amperímtetros y voltímetros, los valores de corriente y tensión en las tres fases del sistema, respectivamente. Se muestran a continuación las siguientes imágenes, donde pueden verse los valores de corriente y tensión obtenidos mediante el uso del bloque “scope”, que es una herramienta del software que va a permitir visualizar los valores de estas magnitudes.

3.2. Control de Corriente

Este bloque es el encargado de realizar el control de corriente. Como Podemos ver en la figura, es un bloque que tiene seis entradas y tres salidas. Entre las entradas destacamos las medidas de corriente y tensión trifásicas medidas en el convertidor de potencia, las referencias de potencia activa y reactiva, la tensión dc – link y un enable.

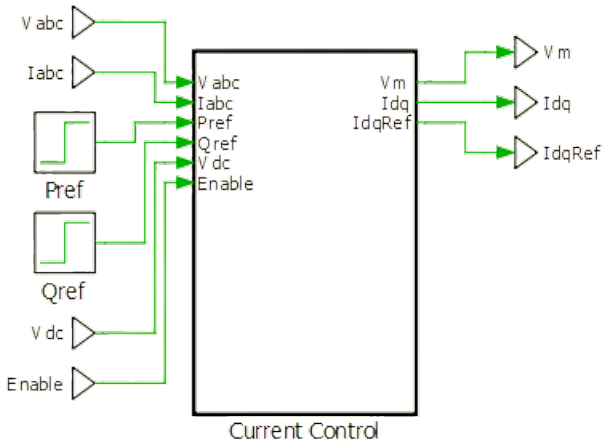

Figura 11. Esquema bloque Current Control

Dentro de este subsistema podemos diferenciar entre 4 grandes divisiones: Inner Control Loop, antitransformadas de Clarke y Park, transformadas de Clarke y Park junto al bloque PLL en un mismo subsistema, y por último, un c – script con el cálculo de las referencias de corriente necesarias para el control de las corrientes.

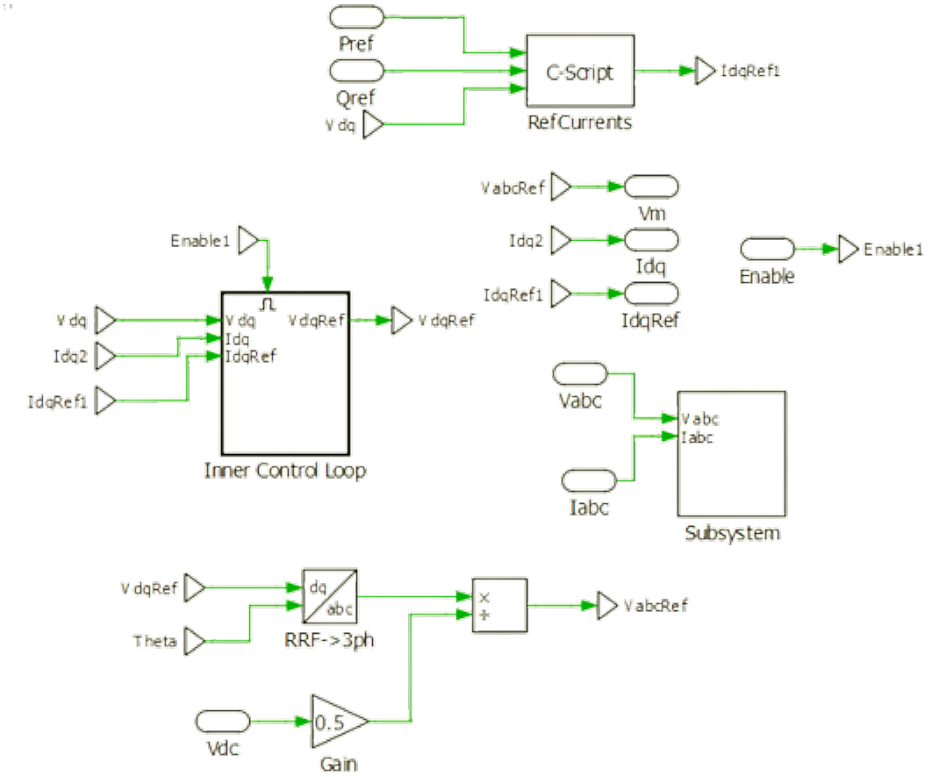

Figura 12. Diagrama de bloques del Current Control

3.2.1. Subsistema: transformadas de Clarke y Park. PLL

Empezando por este subsistema, es destacable que no tiene ninguna salida aparente, siendo sus entradas las corrientes y tensiones medidas del convertidor conectado a red. En realidad, mirandos dentro del subsitema, es fácil identificar cuáles son sus salidas: tensión y corriente en ejes síncronos (V_{dq} e I_{dq}, respectivamente), además de la fase (Theta).

Este capítulo sirve únicamente para presentar el modelo de simulación, por lo que tendrá una visión general del mismo. Es por este mismo motivo por el cual en las imágenes se mostrarán los bloques "3ph-> SRF" y "SRF-RRF", que realizan directamente la transformación de ejes naturales a ejes alpha-beta, y de éstos a ejes síncronos, esto es, realizan de forma conjunta las transformaciones de Clarke y Park de la tensión y la corriente. En realidad, conforme se avance en el desarrollo de este trabajo, se irán sustituyendo estos bloques por c – scripts, ya que es necesario que estén codificados para que puedan ser leídos por el microcontrolador más adelante. Cabe recordar que la transformada de Clarke se realiza con la constante denominada "Power Invariant", sin embargo, buscando en cómo están configurados internamente éstos bloques, se puede observar que PLECS usa la constante "Non-power invariant", es decir, se ha tenido que pasar a la constante deseada mediante multiplicación por bloques para obtener los resultados deseados.

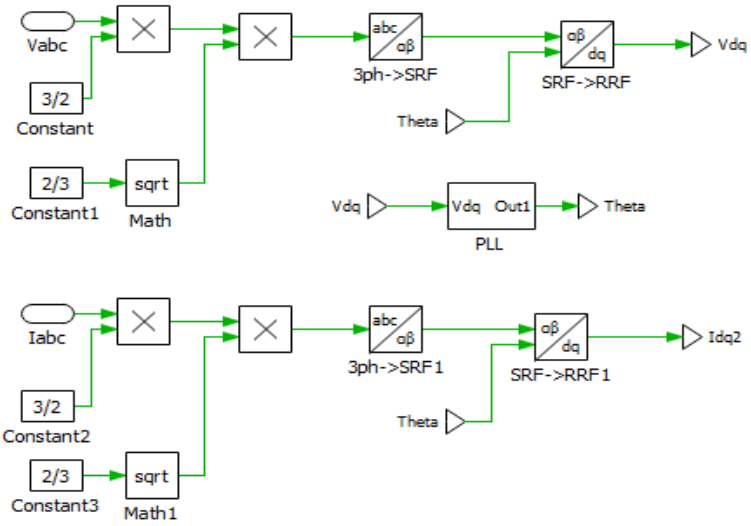

Figura 13. Diagrama bloques de las transformadas de Clarke y Parke

Por último, se hace mención al bloque PLL (Phase Locked Loop), que recibe la tensión en ejes síncronos dq y devuelve la fase "Theta". El funcionamiento de este bloque consiste en sincronizar sobre uno de los ejes del Sistema, en este caso sobre el eje q (ya que la tensión tiene solo componente d en ejes síncronos). La fase obtenida se realimenta sobre la propia transformada de Park, pudiéndose observar cómo es entrada para el bloque "3ph-> RRF". Dentro de este bloque PLL se puede encontrar, además, un bloque PI, siendo la constante proporcional e integral de 2 y 250, respectivamente, y un tiempo de muestreo de 100µs.

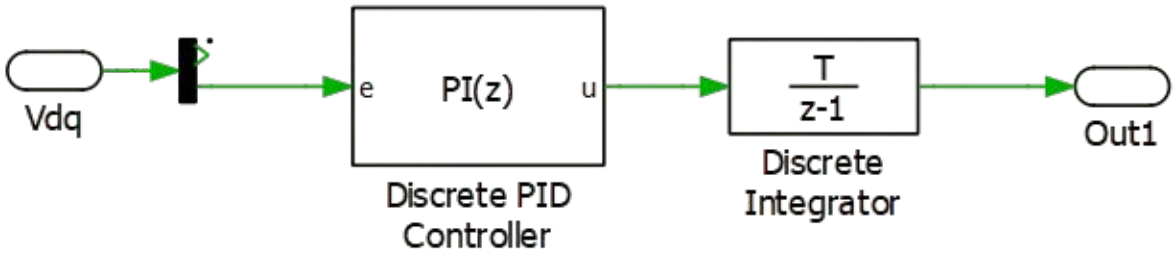

Figura 14. Diagrama de bloques PLL

3.2.2. Cálculo de las corrientes de referencia

Como bien se puede deducir del nombre del apartado, en este bloque se realiza el cálculo de las corrientes de referencia (I_{dqRef} en el modelo de simulación) necesarias para el control de corriente. Se puede apreciar que sus entradas son las referencias de potencia activa y reactiva y las tensiones en ejes síncronos obtenidas del apartado anterior.

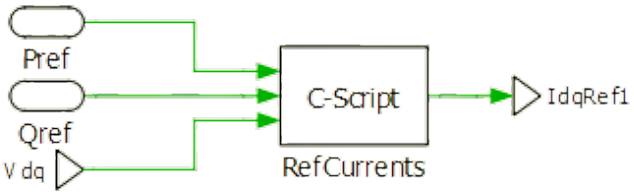

Figura 15. Cálculo de corrientes de referencia mediante c - script

Empezando por las referencias de potencia mencionadas, éstas se configuran de tal forma que para la referencia de potencia activa, temenos un escalón de 0 a 5kW en el instante 0.03s, mientras que para la potencia reactiva el escalón es negativo, de 0 a -5kW, y en el instante 0.12s. Estos escalones se verán reflejados en el resultado de la simulación, cuando gracias a un par de bloques scope obtengamos I_d, I_q y sus referencias.

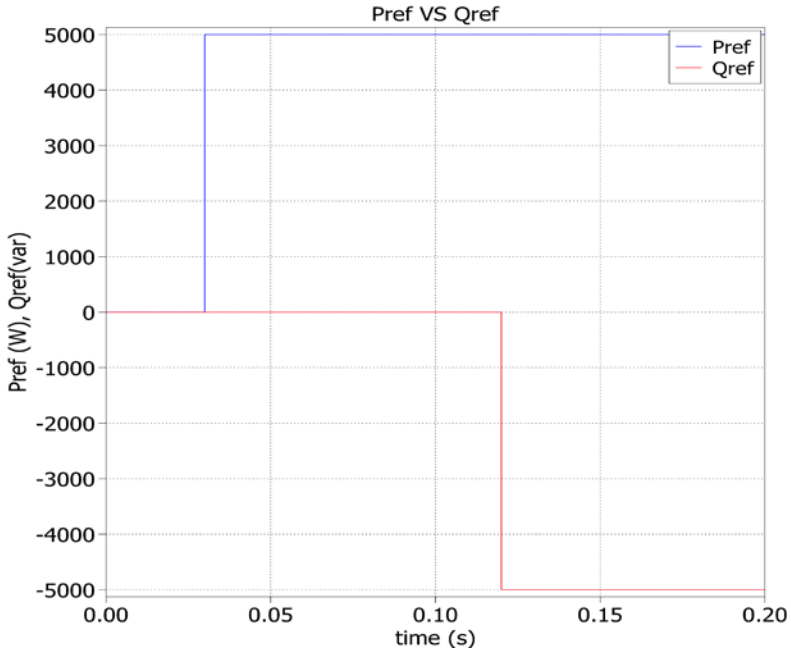

Figura 16. Potencias activas y reactiva de referencia

Por último, queda la configuración de este c – script llamado “RefCurrents”. Este bloque es el encargado de realizar el cálculo de las corrientes de referencia mencionado anteriormente.

El funcionamiento de este bloque se basa en la “*Instantaneous Power Theory*”, que podemos encontrar en las referencias [6] y [10]. Vemos cómo se presentan dos nuevos conceptos para las potencias activa y reactiva, tal que:

- La potencia activa, p: es la “*transferencia de energía instantánea entre la fuente y la carga*”. [6]
- La potencia reactiva, q: es la “*transferencia de energía instantánea entre las fases del sistema*”. [6]

Se encuentran, además, una serie de expresiones para la transformación empleada en este proyecto, es decir, para la transformación de ejes “Power Invariant”. Estas expresiones son:

- Para las potencias activa y reactiva instantáneas en ejes estacionarios:

$$p = v_{\alpha\beta}^{T} i_{\alpha\beta} \quad (3.1.)$$

$$q = v_{\alpha\beta}^{T} J\, i_{\alpha\beta} \quad (3.2.)$$

- Para las potencias activa y reactiva instantáneas en ejes síncronos:

$$p = v_{dq}^{T} i_{dq} \rightarrow p = v_d i_d \quad (3.3.)$$

$$q = v_{dq}^{T} J\, i_{dq} \rightarrow q = -v_d i_q \quad (3.4.)$$

Siendo J = $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ una matriz que gira el vector 90º.

Teniendo esto en cuenta, el término $J \cdot i_{dq} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} id \\ iq \end{bmatrix} = \begin{bmatrix} -i_q \\ i_d \end{bmatrix}$ origina como se puede apreciar, términos cruzados, lo que deriva en que una variación en el eje d afecte al eje q y viceversa, formando en la gráfica una serie de picos que habría que tener en cuenta.

Gracias a estas expresiones de las potencias activa y reactiva instantáneas, se obtienen las referencias en ejes dq:

$$\begin{bmatrix} i_d^* \\ i_q^* \end{bmatrix} = \frac{1}{v_d^2 + v_q^2} \begin{bmatrix} v_d & v_q \\ v_q & -v_d \end{bmatrix} \begin{bmatrix} p^* \\ q^* \end{bmatrix} \rightarrow \begin{bmatrix} i_d^* \\ i_q^* \end{bmatrix} = \frac{1}{v_d} \begin{bmatrix} p^* \\ q^* \end{bmatrix} \quad (3.5.)$$

Esta expresión es la que está codificada en el bloque c – script de la siguiente manera para que realice el cálculo de estas corrientes de referencia:

$$InvMod = 1/(V_d^2 + V_q^2) \quad (3.6.)$$

$$I_d Ref = InvMod(V_d P_{ref} + V_q Q_{ref}) \quad (3.7.)$$

$$I_q Ref = InvMod(V_q P_{ref} - V_d Q_{ref}) \quad (3.8.)$$

[6]

3.2.3. Inner Control Loop

Siguiendo el orden de obtención de variables, una vez obtenidas las corrientes y tensiones en ejes síncronos, y tras haber calculado las corrientes de referencia necesarias, también en ejes síncronos, este bloque, gracias a una señal de enable que pasará de 0 a 1 en el instante 0.2s, se encarga de calcular las referencias de tensión en estos ejes.

En primer lugar, se definen los siguientes términos, necesarios para comprender las ecuaciones y expresiones posteriores:

VARIABLE	DESCRIPCIÓN
$\boldsymbol{v_{abc} = \{v_{an}\, v_{bn}\, v_{cn}\}^T}$	Vector de tensión de la red
$\boldsymbol{i_{abc} = \{i_a\, i_b\, i_c\}^T}$	Vector de corriente proveniente de la red
$\boldsymbol{u_{abc} = \{u_a\, u_b\, u_c\}^T}$	Vector de señales de control
R	Modela las pérdidas de Joule
L	Bobina de conexión
C	Condensador del dc – link
$\boldsymbol{V_{dc}}$	Tensión dc – link

Tabla 1. Magnitudes que intervienen en el modelo del convertidor

A continuación, se muestran las ecuaciones que modelan la dinámica del Sistema:

- Dinámica de la corriente de entrada
  - En ejes naturales abc:

$$v_{abc} = Ri_{abc} + \frac{Ldi_{abc}}{dt} + u_{abc} \quad (3.9.)$$

  - En ejes estacionarios $\alpha\beta$:

$$v_{\alpha\beta} = Ri_{\alpha\beta} + \frac{Ldi_{\alpha\beta}}{dt} + u_{\alpha\beta} \quad (3.10.)$$

  - En ejes síncronos dq:

$$v_{dq} = Ri_{dq} + \frac{Ldi_{dq}}{dt} + J\omega Li_{dq} + u_{dq} \quad (3.11.)$$

- Dinámica de la tensión del condensador
  - En ejes naturales abc:

$$C\frac{d}{dt}\left(\frac{V_{dc}^2}{2}\right) = u_{abc}^T i_{abc} - V_{dc} i_{Load} \quad (3.12.)$$

  - En ejes estacionarios $\alpha\beta$:

$$C\frac{d}{dt}\left(\frac{V_{dc}^2}{2}\right) = u_{\alpha\beta}^T i_{\alpha\beta} - V_{dc} i_{Load} \quad (3.13.)$$

  - En ejes síncronos dq:

$$C\frac{d}{dt}\left(\frac{V_{dc}^2}{2}\right) = u_{dq}^T i_{dq} - V_{dc} i_{Load} \quad (3.14.)$$

El control de corriente depende únicamente de las ecuaciones dinámicas de la corriente de entrada vistas anteriormente. Esto es, para el diseño del control de corriente en ejes síncronos debemos seguir los siguientes pasos, empezando, en primer lugar, con la ecuación de la dinámica de la corriente vista en $v_{dq} = Ri_{dq} + \frac{Ldi_{dq}}{dt} + J\omega Li_{dq} + u_{dq}$ (3.11.)

Lo que se quiere obtener es que aparezca la derivada del error, $\tilde{\imath}_{dq}$. Esto se consigue sumando y restando términos:

$$v_{dq} = Ri_{dq} + L\frac{di_{dq}}{dt} + J\omega Li_{dq} + u_{dq} \pm L\frac{di_{dq}^*}{dt} \pm K_p\tilde{\imath}_{dq} \quad (3.15.)$$

Siendo $\tilde{\imath}_{dq} = i_{dq} - i_{dq}^*$

Reordenando los términos de la siguiente forma:

$$v_{dq} = L\frac{d\tilde{\imath}_{dq}}{dt} + K_p\tilde{\imath}_{dq} + J\omega Li_{dq} + u_{dq} + Ri_{dq} - K_p\tilde{\imath}_{dq} + L\frac{d\tilde{\imath}_{dq}}{dt} \quad (3.16.)$$

Y teniendo en cuenta que la señal de control u_{dq} es tal que:

$$u_{dq} \triangleq v_{dq} - Ri_{dq} - J\omega Li_{dq} + K_p\tilde{\imath}_{dq} - L\frac{di_{dq}^*}{dt} \quad (3.17.)$$

Donde $-Ri_{dq}$ es la caída de tensión en la Resistencia. Las pérdidas de Joule del Sistema dependen del punto de operación del mismo. No conocido el valor de R, el término $-Ri_{dq}$ puede estimarse como un término integral del error en corriente $K_i\int\tilde{\imath}_{dq}dt$. Siguiendo con el resto de términos de la

expresión $u_{dq} \triangleq v_{dq}\text{-}Ri_{dq}\text{-}J\omega Li_{dq} + K_p\tilde{\imath}_{dq}\text{-}L\frac{di^*_{dq}}{dt}$ (3.17.), Podemos también identificar el término de desacoplo $-J\omega Li_{dq}$, un término de amortiguamiento que asegura la estabilidad, $K_p\tilde{\imath}_{dq}$, y el último término, $-L\frac{di^*_{dq}}{dt}$, que vale 0 en régimen permanente.

Pueden eliminarse términos y se obtiene la dinámica del error en corriente en lazo cerrado:

$$0 = \frac{Ld\tilde{\imath}_{dq}}{dt} + K_p\tilde{\imath}_{dq} \rightarrow \lim_{t\to\infty}\tilde{\imath}_{dq} = 0 \qquad (3.18.)$$

Con este breve marco teórico, es sencillo comprender el siguiente diagrama de bloques que calcula las tensiones de referencia en ejes dq. Los controladores PI están ambos configurados a constante proporcional de 8, y constante integral de 3000, además de un periodo de muestreo de 100µs. Se hace mención además al valor de la ganancia que multiplica a ambas componentes de la corriente en ejes síncronos. Este valor no es otro que ωL, siendo ω igual a $2\pi f$, con f = 50Hz como la frecuencia de la red. Este término puede apreciarse en las expresiones antes desarrolladas.

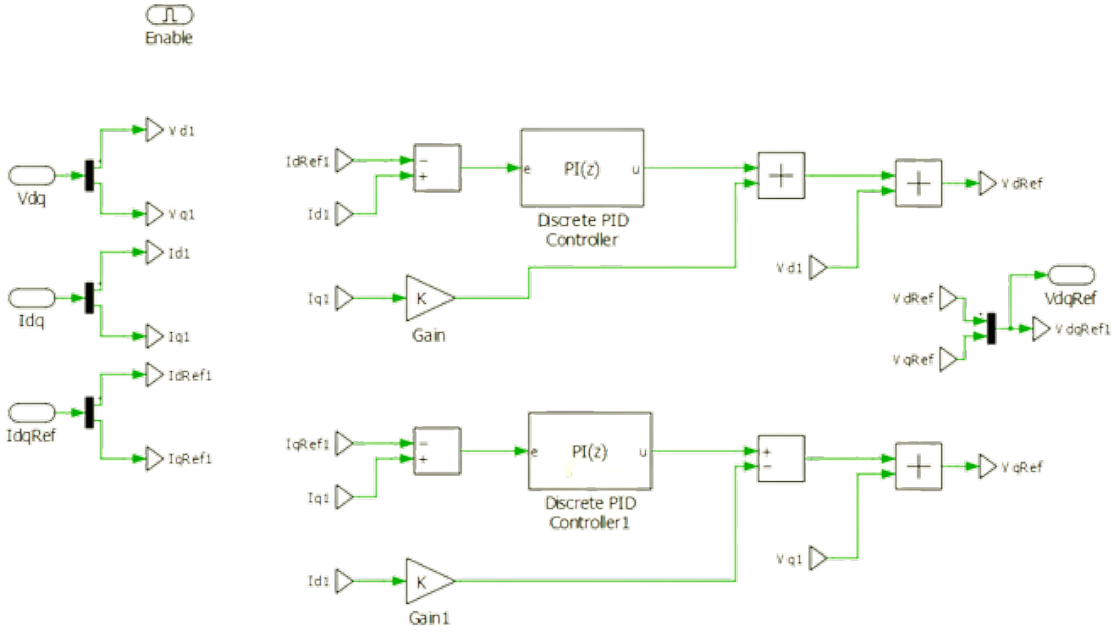

Figura 17. Diagrama de bloques del Inner Control Loop

"La respuesta del controlador depende fuertemente de una correcta sincronización. Es necesario un PLL de altas prestaciones" [6] como el diseñado para este proyecto.

[6]

3.2.4. Antitransformadas de Clarke y Park

El último conjunto de bloques destacables es el que realiza las antitransformadas de Clarke y Park, de forma directa en esta primera observación del modelo, y paso a paso conforme se avance en los capítulos de este Trabajo de Fin de Grado.

No hay mucho que decir sobre este conjunto de bloques, puesto que de forma directa con los bloques "RRF-> 3 ph" se realiza la transformada, usando para ello el vector de tensión de referencia en ejes dq y la fase "Theta". Se emplea, además, la tensión dc – link dividia por la mitad. Esto último tiene sentido gracias a conocimientos previous de Electrónica de Potencia: obtenidos los valores de referencia de tensión con las antitransformadas, es necesario que estos valores estén normalizados para que sea posible compararse con la señal portadora triangular de la modulación PWM (se explicará en el siguiente apartado). La forma que se tiene para normalizar esta referencia de tensión es la siguiente: activado SW01, SW02 estará desactivado, lo que quiere decir que solo se

verá una tensión de $V_{dc}/2$. Esta tensión es la que se usará para normalizar la tensión de referencia obtenida, dividiendo esa referencia entre este valor.

De esta forma, obtenemos el vector de tensiones abc de referencia, que aparecerá también en el siguiente apartado, puesto que se usa para la modulación PWM.

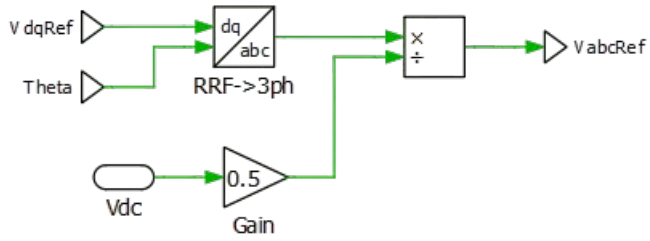

Figura 18. Diagrama de bloques de las antitransformadas de Clarke y Park

3.3. Modulador PWM

Uno de los elementos esenciales para el correcto funcionamiento de un convertidor de potencia es el modulador. Este elemento es el que se encarga de sintetizar los disparos de los IGBTs a partir de las tensiones de referencias obtenidas del apartado anterior, que se reciben en este subsistema como entradas, y siendo dichos disparos las salidas del subsistema.

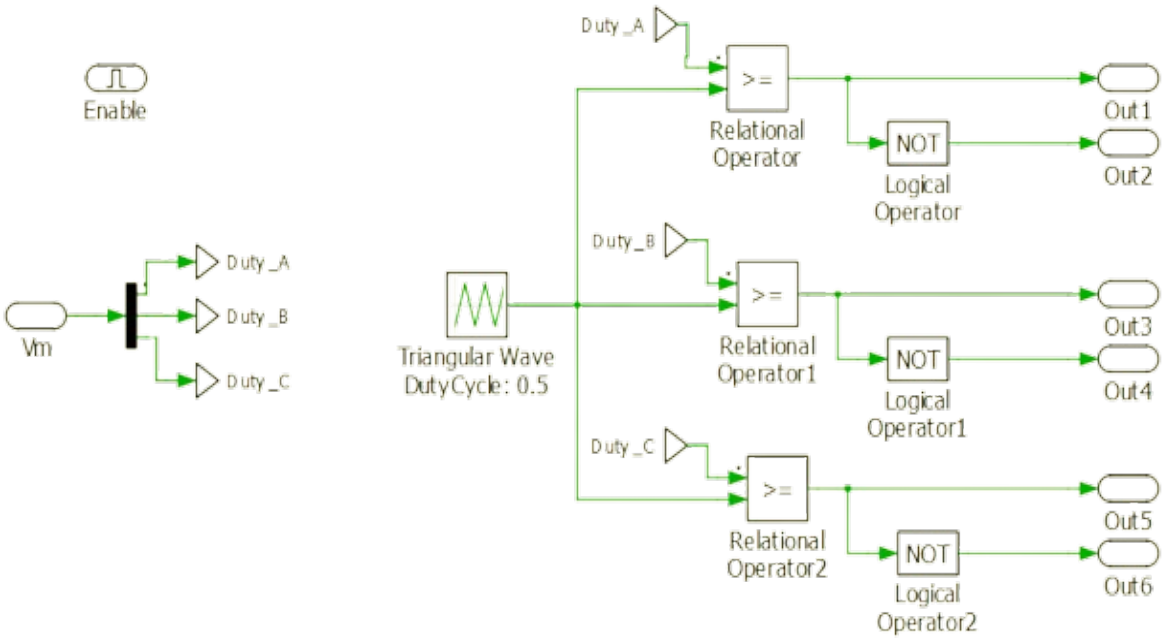

Figura 19. Diagrama de bloques del modulador

La modulación por anchura de pulsos (PWM) es una de las más simples y utilizadas, debido a que permite controlar la cantidad de corriente (por tanto, potencia) que se entrega a otro dispositivo modificando el ancho de pulso de una señal digital. Su funcionamiento se basa en comparar el valor de una señal portadora triangular, periódica y de frecuencia, en este caso de 10kHz, con las señales de referencia de las tensiones de red. A este método se le conoce como modulación sinusoidal por anchura de pulsos. De cada componente de tensión de referencia, se obtienen dos señales de disparo, siendo una de ellas la negada de la otra. De esta forma, se obtienen seis señales de disparo, una para cada IGBT del convertidor de potencia (SW01, SW02, SW03, SW04, SW05,

SW06). Por lo tanto, si la señal de referencia es mayor o igual que la portadora triangular, habrá disparo de SW01, SW03, SW05, mientras que el resto quedará inactivo, y viceversa.

En la siguiente figura se ven dos gráficas que representan lo siguiente: la gráfica superior contiene a la señal portadora triangular. De igual modo podemos observar las tres señales de tensión de red de referencia en ejes abc. Tal y como se comentó en el apartado de las antitransformadas, estas señales de tensión están normalizadas para que sea posible la modulación. En la gráfica inferior se ve un ejemplo de señales de disparo, en este caso del primer IGBT (SW01), que se consigue con la comparación de la portadora triangular con la tensión en la fase "a". Para el caso del segundo IGBT, esto es, SW02 sería al contrario.

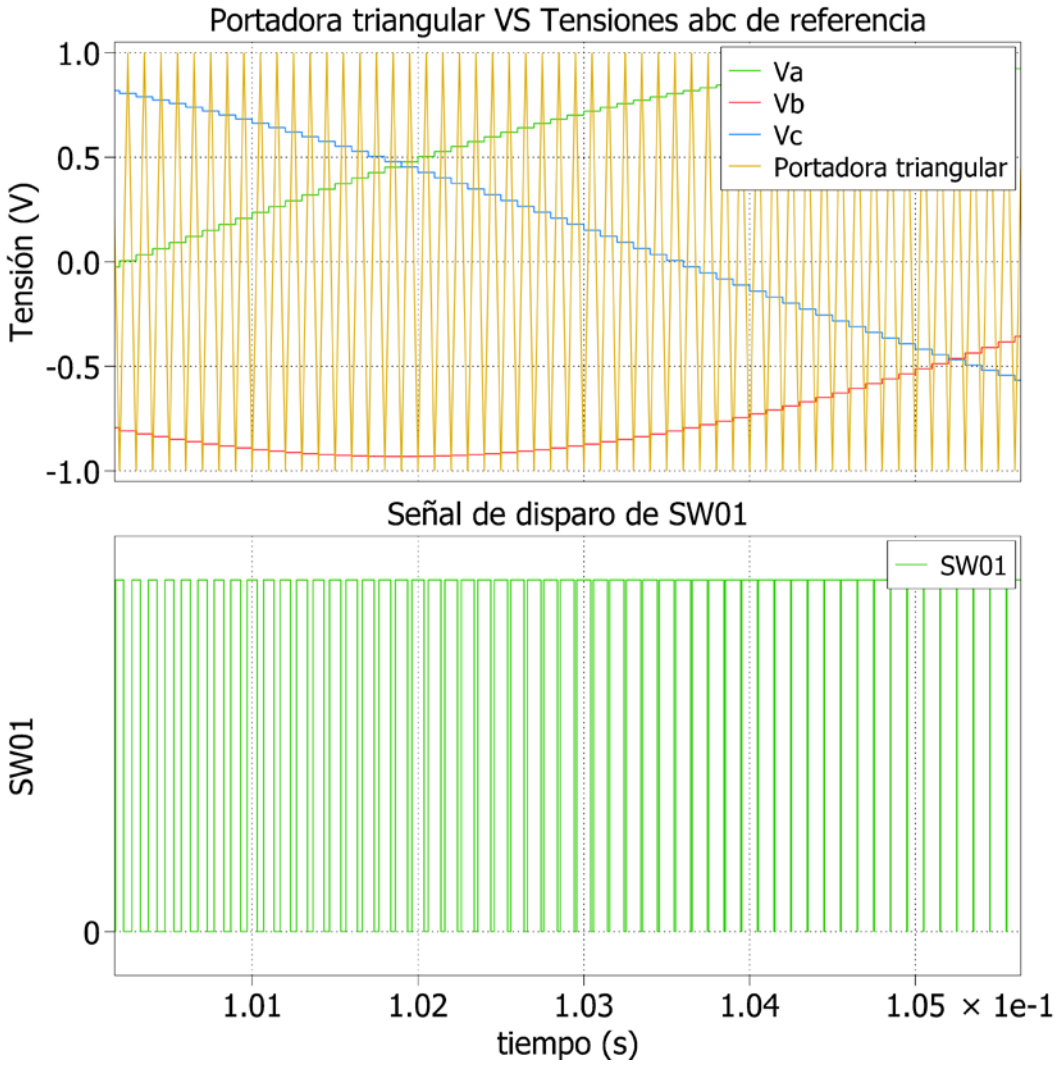

Figura 20. a) Portadora triangular con tensiones de referencia en ejes abc, b) Señal de disparo de SW01

3.4. Resultados

Los resultados típicos obtenidos son los mostrados en las siguientes figuras. Se mostrarán primero las variables de entrada al controlador (bloque “Current Control”): éste trabaja en una primera instancia recibiendo las corrientes y tensiones reales de la red (abc).

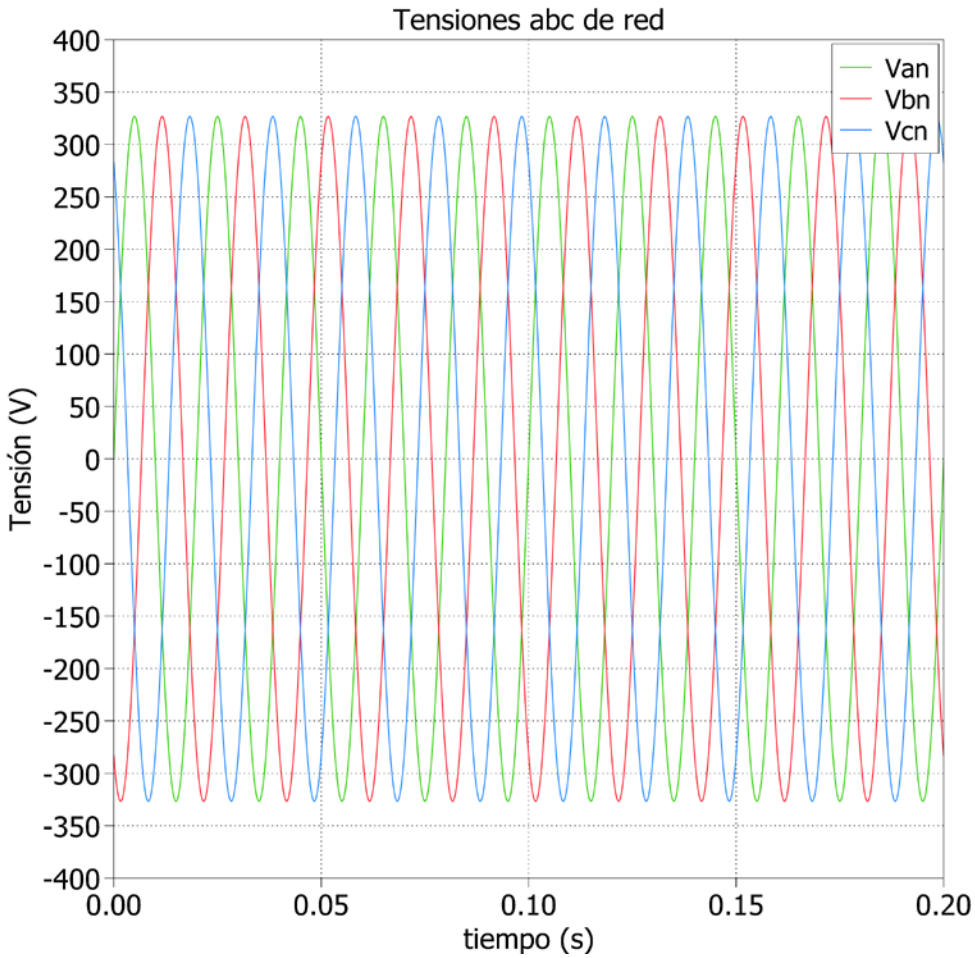

Figura 21. Tensiones obtenidas de la red (abc)

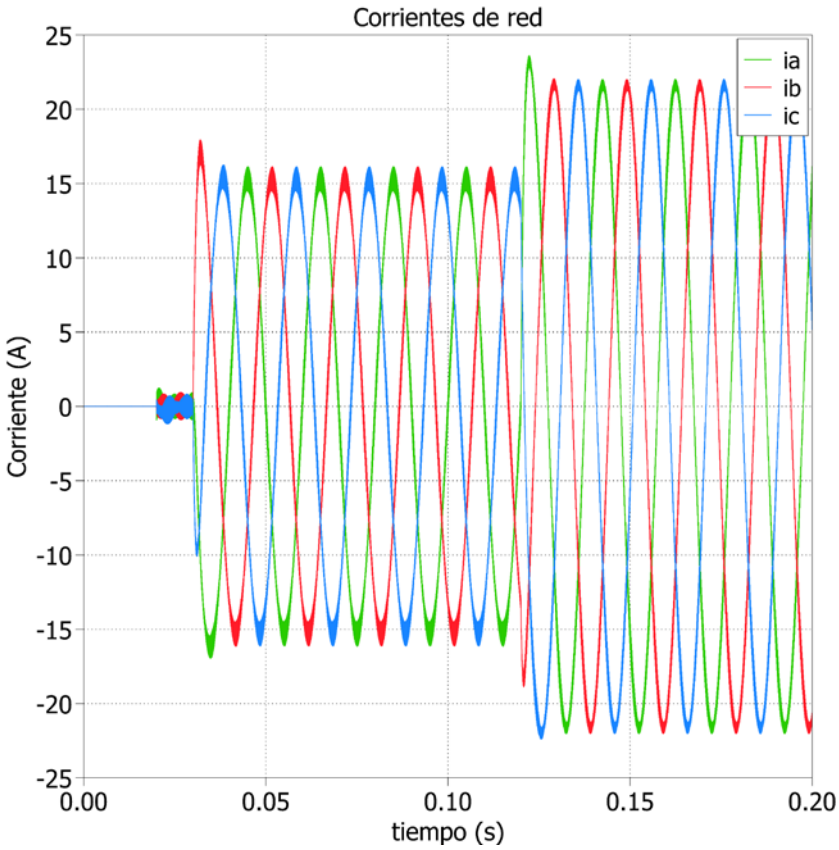

Figura 22. Corrientes obtenidas de la red (abc)

Dentro de este bloque se obtienen las tensiones en ejes dq (cuya componente q es igual a cero) y las corrientes en dq gracias al subsistema del PLL. Si a estas señales se añade la referencia de corrientes en dq (Subsistema RefCurrents), se completan así las entradas al subsistema Inner Control Loop, el cual se encargaba de obtener las tensiones de referencia en ejes dq. A continuación, se muestran las gráficas correspondientes a las señales mencionadas.

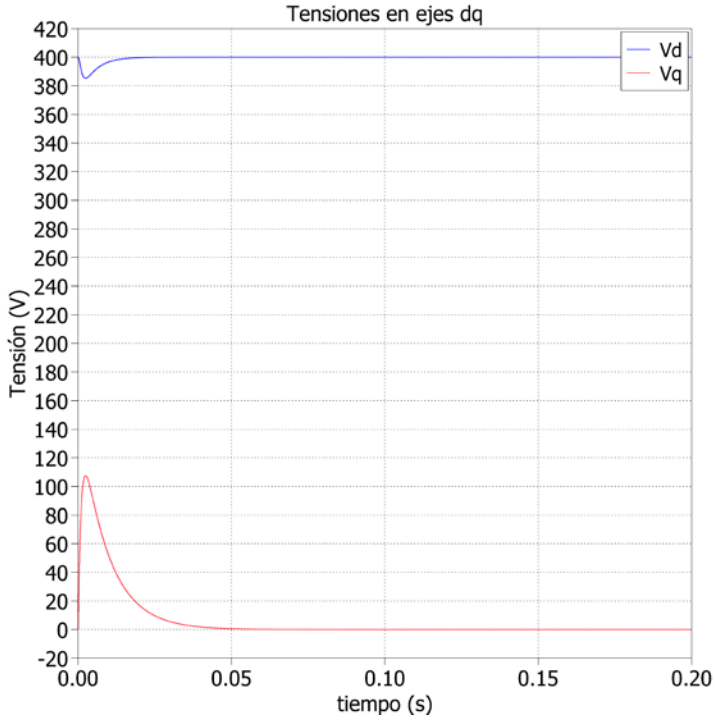

Figura 23. Tensiones en ejes síncronos dq

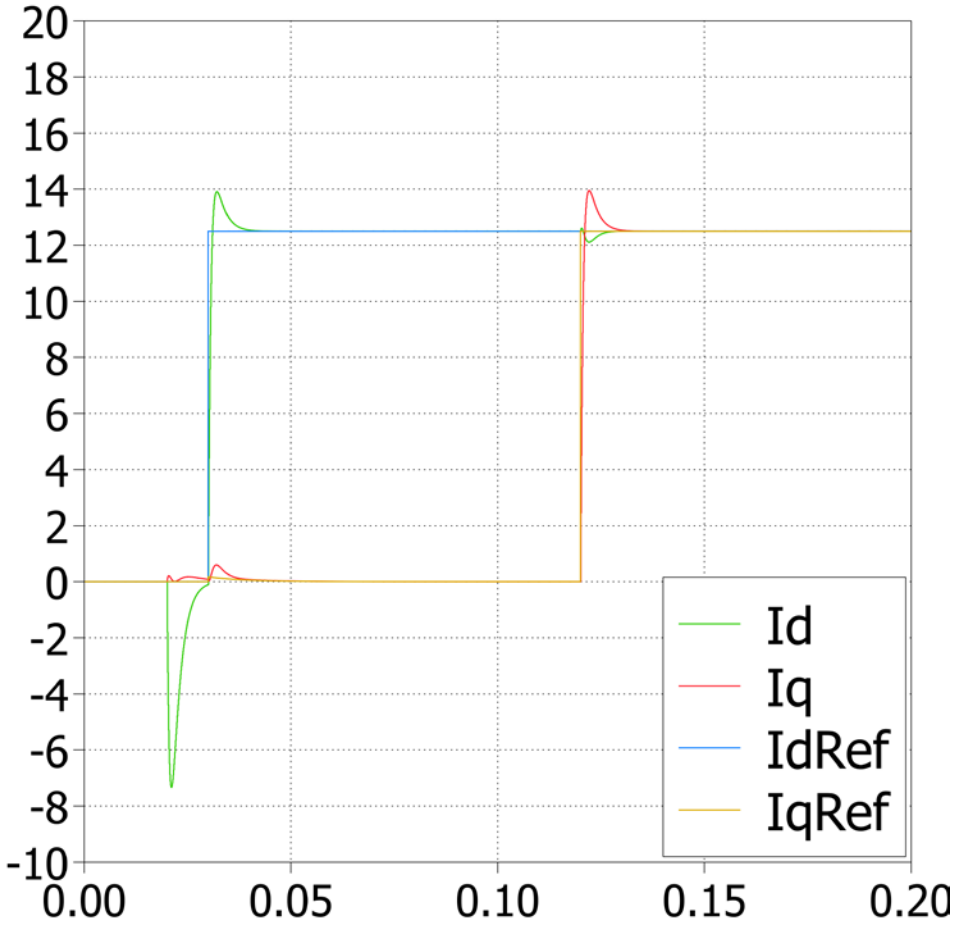

Figura 24. Corrientes en ejes dq junto con sus referencias

Idelamente, el error en regimen permanente debería de ser nulo, además de no existir acoplo entre ejes debido al término cruzado $-J\omega Li_{dq}$ de las expresiones de apartados anteriores para el control de corriente. Como se puede intuir, esto no ocurre en la práctica.

Una observación que se puede realizar viendo la gráfica es el efecto del enable en el instante 0.02s. Si centramos nuestra atención en ese instante, es fácil apreciar un pequeño pico en la corriente, a pesar de ser la referencia constante e igual a cero. Esto se debe a que el enable es el encargado de activar a los PI. Los bloques del controlador PI tienen la función de hacer que las corrientes se parezcan a las referencias, por tanto al iniciarse el enable, estos controladores empiezan a regularse hasta que consiguen alcanzar la referencia.

Otra observación que se podría hacer de estas gráficas es en el instante 0.12s, cuando se inyecta la potencia reactiva, I_d sufre una pequeña perturbación. La inyección de este tipo de potencia en el sistema hace que se tienda a alcanzar un nuevo valor. Esto se puede deducir de las expresiones $\mathrm{I_dRef} = \mathrm{InvMod}(\mathrm{V_dP_{ref}} + \mathrm{V_qQ_{ref}})$ (3.7.) y $\mathrm{I_qRef} = \mathrm{InvMod}(\mathrm{V_qP_{ref}}\text{-}\mathrm{V_dQ_{ref}})$ (3.8.); recordando la definición de ejes síncronos, se conoce que el valor de la tensión en el eje q es de cero. La componente de I_d que multiplica a $V_q \cdot Q_{ref}$ será por tanto siempre cero, lo que quiere decir que al inyectar Q, el Sistema trata de Volver a ajustarse, y esto es lo que origina dicha perturbación. Esto ecplica tambíen por qué I_q vale cero hasta ese instante (observar expresión $\mathrm{I_qRef} = \mathrm{InvMod}(\mathrm{V_qP_{ref}}\text{-}\mathrm{V_dQ_{ref}})$ (3.8.)).

Como ya se ha comentado, lo ideal sería no encontrarse ninguna perturbación. En cambio, ante los cambios de referencia es notable la aparición de éstas, debido también a la estimación que se realiza de la caída de tensión en la resistencia empleando el término integral del controlador.

Por último, se hace mención a al tiempo de establecimiento y a la S.O., de 5ms y de 2.5A, respectivamente. Se hicieron pruebas combinando distintos valores de Kp y Ki en los controladores PI, pero los obtenidos y comentados anteriormente son los que obtuvieron mejores resultados. Si analizamos de forma aislada los

cambios en cada una de las constantes obtenemos lo siguiente:

- Un valor de Kp menos al elegido haría de nuestro sistema un sistema sobreamotiguado, y no llegaría a alcanzar el valor de referencia, haciéndose inestable, mientras que un valor mayor al escogido aumentaría el pico en el transitorio y ralentizaría la estabilización del sistema en la referencia.
- Si analizamos ahora con detalle variaciones en la constante integral, un menor valor de Ki bajaría el pico del transitorio pero ralentizaría bastante que se alcanzase la referencia. Por otra parte, para un Ki mayor que el elegido, ocurriría justamente lo contrario.

Se muestra en la siguiente figura el modelo completo implementado en PLECS.

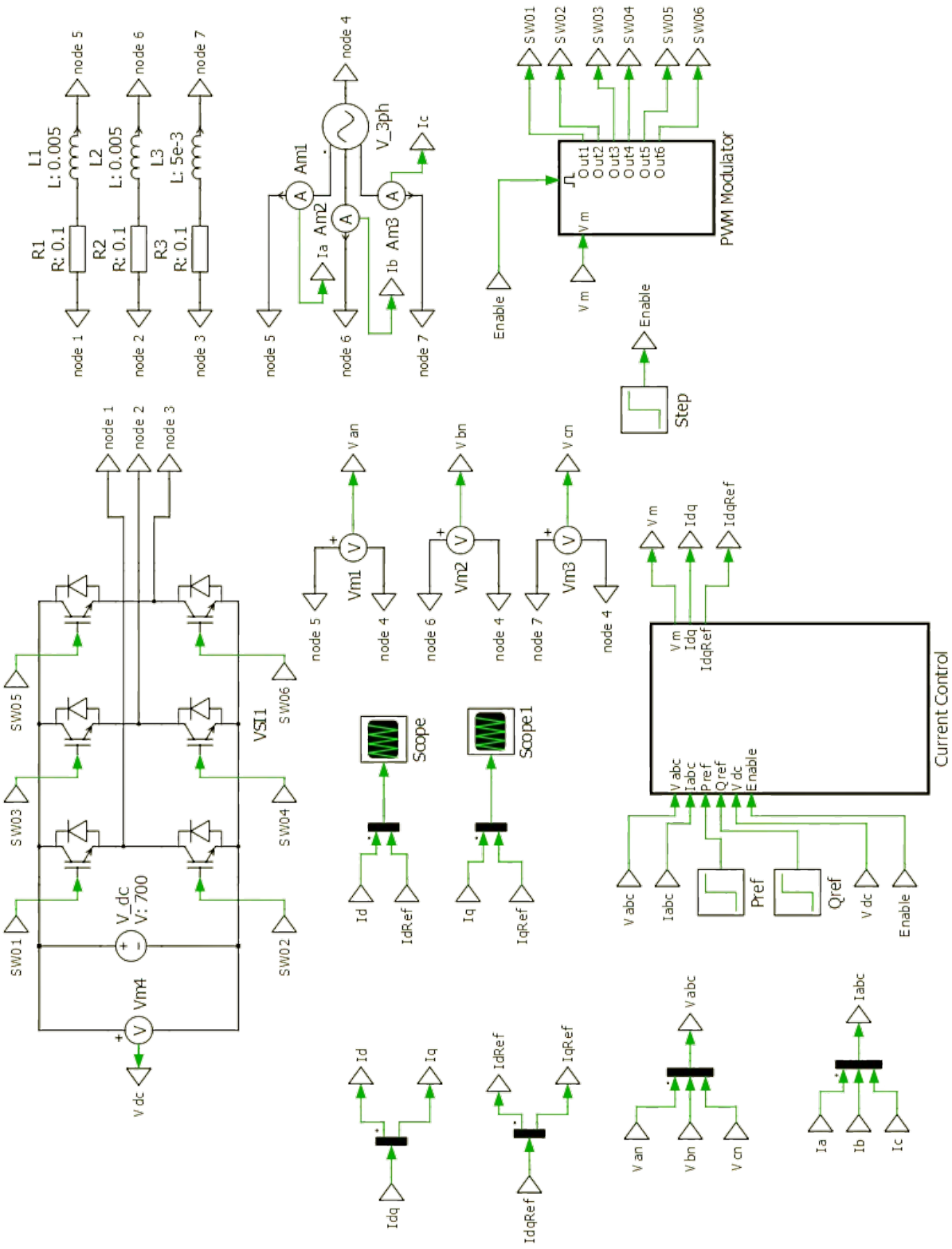

Figura 25. Captura del modelo de simulación del convertidor conectado a red, el control de corriente y el modulador PWM

4 IMPLEMENTACIÓN

Tras definir el modelado del sistema del convertidor de potencia y todos los bloques que lo conforman, llega el momento de ajustar el sistema simulado para que el microcontrolador, que se va a encargar de realizar los cálculos y operaciones, además de hacer PIL, pueda leer este modelo. Esto se consigue sustituyendo algunos bloques que hacen las operaciones directas por c – script de manera que queden codificados y puedan ser leídos por el MCU.

Un ejemplo claro de este tipo de bloques que son necesarios sustituir son aquellos que realizan las transformadas (o antitransformadas) de Clarke y de Park. El nuevo modelo de simulación quedaría tal y como puede verse en las siguientes figuras.

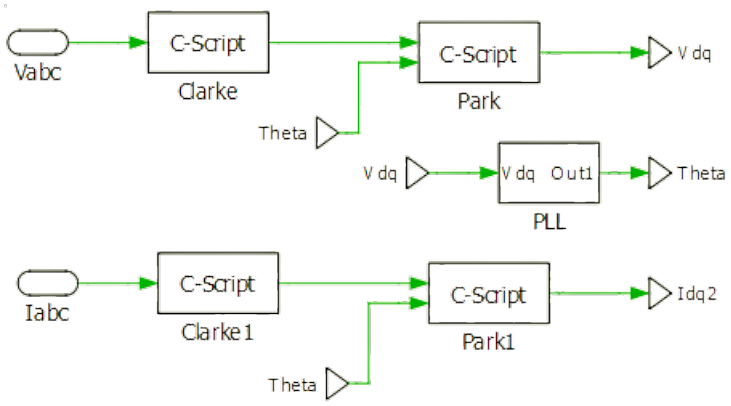

Figura 26. Transformadas de Clarke y Park implementadas con c – script

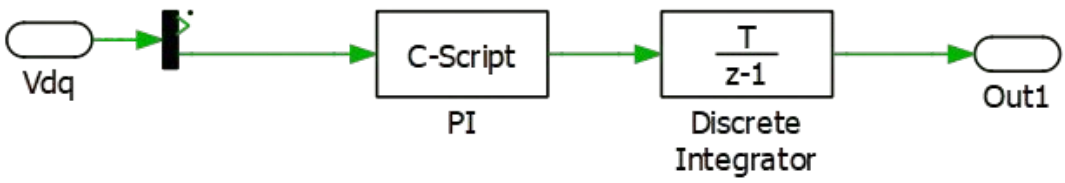

Figura 27. PLL implementado con bloque c - script

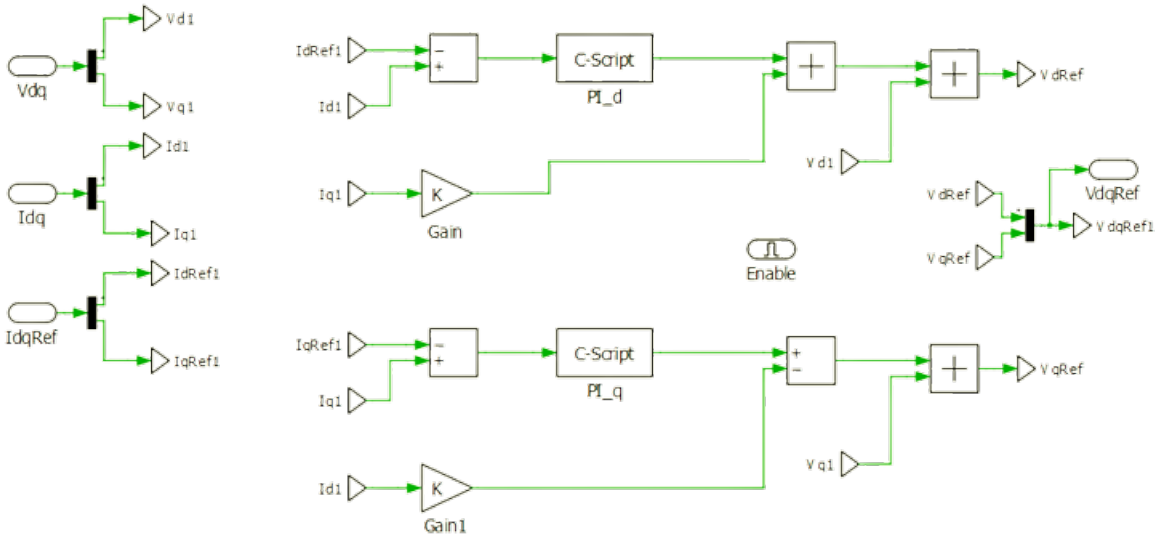

Figura 28. Inner Control Loop implementado con c – script

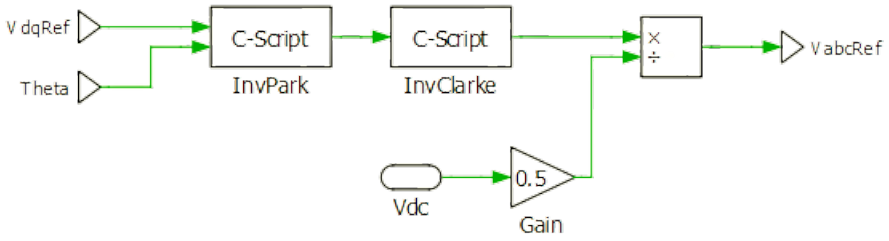

Figura 29. Antitransformadas de Clarke y Park implementadas con c – script

Tal y como se ha podido apreciar, simplemente se sustituyen los bloques mencionados anteriormente por estos bloques c – script los cuales se configuran y codifican para que tengan un comportamiento análogo a aquellos que sustituyen. Si nos centramos en la configuración, todos y cada uno de estos bloques tienen un periodo de muestreo igual a 100µs, y a cada uno se le deben asignar el número de salidas y entradas que poseen según el caso. La siguiente figura muestra un ejemplo de cómo se configuran estos c – script, en este caso, el relativo a la transformada de Clarke.

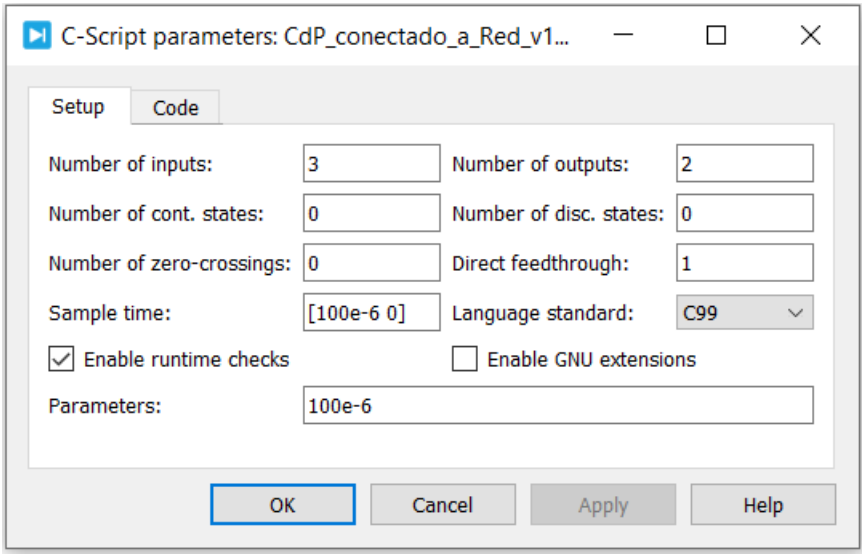

Figura 30. Configuración c - script que realiza la función de la transformada de Clarke

En este caso, se tienen tres entradas que son las tres componentes del sistema trifásico, y dos salidas, en ejes αβ.

Otro caso sería el de la transformada de Park, que como se muestra a continuación, la forma de introducirle el número de variables de entradas se realiza mediante un vector de dos componentes. El primer valor del vector hace referencia a la primera entrada, esto es, la magnitud en ejes αβ, que al ser una entrada de dos componentes hay que especificarlo numéricamente. El segundo valor del vector es una entrada de una componente y corresponde a la fase calculada en el PLL.

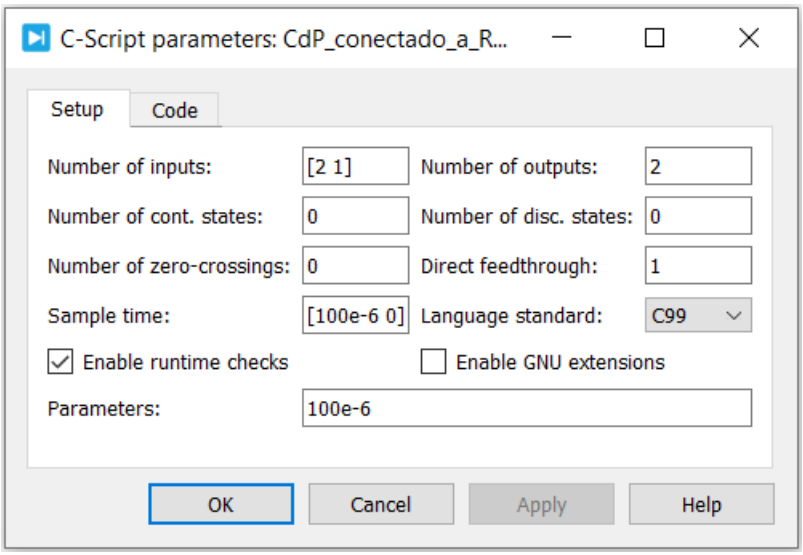

Figura 31. Configuración c - script que realiza la función de la transformada de Park

Dichos bloques de código se deben configurar tal y como se observa en las Figura 34, Figura 35 y Figura 36.

Además de los bloques mencionados para las transformadas y antitransformadas, es necesario sustituir los controladores PI por bloques c – scripts. Esta codificación es sencilla de implementar ayudándonos del ejemplo que proporciona PLECS para c – script.

Este ejemplo nos dice que la ley de control continua para PI se describe de la siguiente manera:

$$y(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau$$

Donde y(t) es la salida del controlador, e(t) es la entrada, y las constantes K_p y K_i son las constantes proporcional e integral, respectivamente. Como estamos implementando un controlador PI digital usando c – script, necesitamos discretizar la ley de control. Esto se consigue utilizando la siguiente expresión para aproximar la integral:

$$i_k = i_{k-1} + T_s e_k$$

Siendo i_k y e_k valores en el instante k de muestreo, i_{k-1} el valor en el instante anterior, y T_s el periodo de muestreo. Asimismo, la ley de control de nuestro controlador PI queda tal que:

$$y_k = K_p e_k + K_i i_k$$

Los valores de las constantes proporcional e integral son conocidos (8 y 3000, respectivamente), al igual que el valor del periodo de muestreo (100µs). El valor empleado como inicial para i_{k-1} es de cero para facilitar el cálculo.

El código de la acción de control está escrito en este caso en el apartado del c – script llamado *"Update function"*. Se hace de este modo ya que se llama una sola vez en cada periodo de muestreo. En cambio, la función de salida sí es llamada varias veces. Es por este motivo que los estados discretos no se pueden calcular en la "Output function", si no que deberemos configurar en los ajustes del c – script el número de estados discretos que necesitaremos (uno en este caso). Se crea así una variable interna llamada *"DiscState(0)"* que hace que el solver

invoque una vez cada periodo a la *"Update function*". Posteriormente se define la variable y_k, que será de tipo global ya que será usada tanto en la *"Update function"* como en la *"Output function"* y representará la salida del controlador, además de asignar el estado discreto a una variable que representará el estado anterior (i_{k-1}) tal y como se mencionó párrafos antes. Es importante actualizar el valor de i_{k-1} después de calcular i_k.

[11]

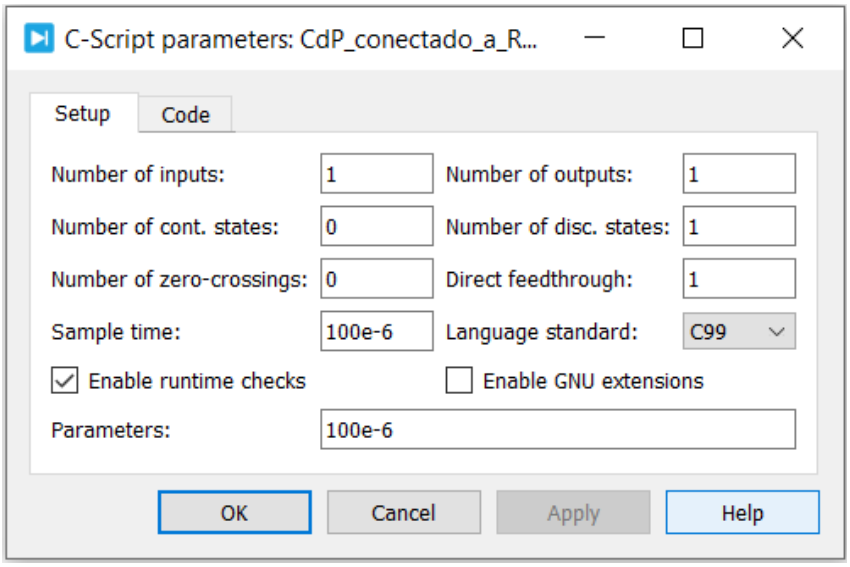

Figura 32. Configuración c - script que realiza la función de controlador PI

Esta última figura representaría el último tipo de configuración que se puede encontrar. Tal y como se ha comentado en este apartado, a diferencia del resto de c – script, los que se encargan de realizar la función de los controladores PI usan un tipo de variable nueva llamada *"DiscState"*, y es por ello por lo que está puesto ese recuadro a uno.

4.1. Resultados del modelo con c - scripts

Se muestran en este apartado los resultados obtenidos, de tal forma que puedan compararse con los resultados obtenidos en otros apartados.

En primer lugar, se muestran las gráficas correspondientes a las corrientes y tensiones reales, es decir, las correspondientes a la red (ejes abc).

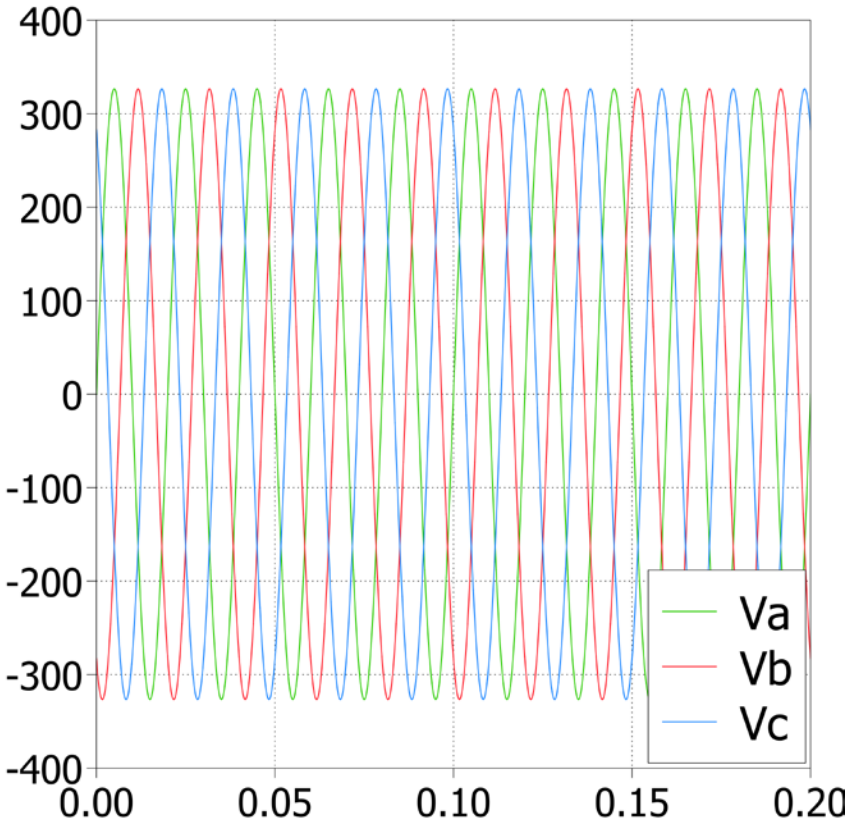

Figura 33. Tensiones en ejes abc en el modelo con c - scripts

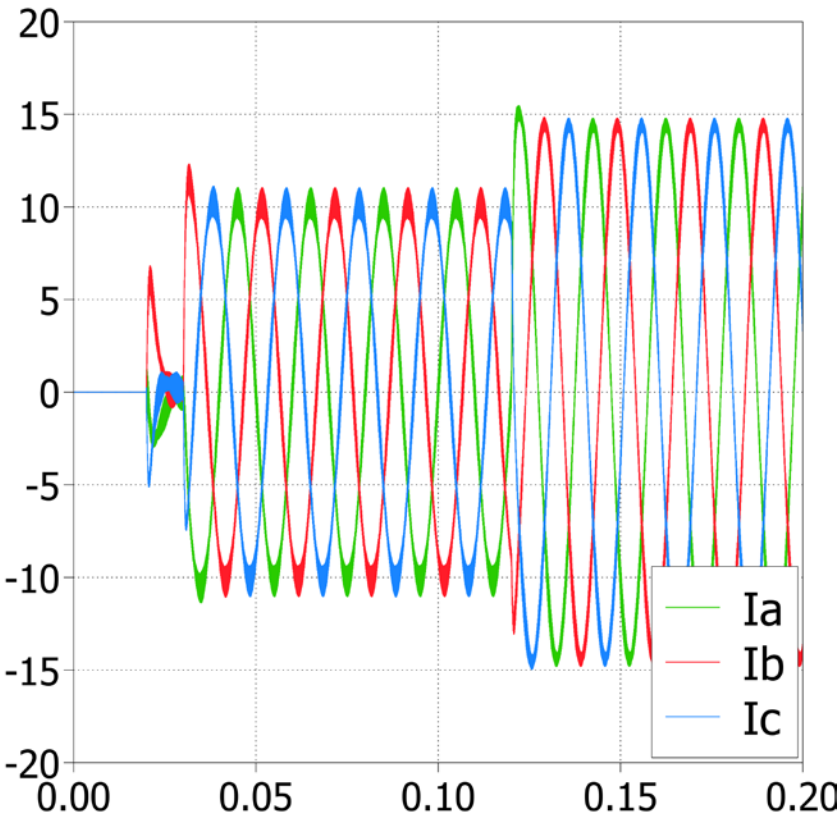

Figura 34. Corrientes en ejes abc en el modelo con c - scripts

A continuación, se adjuntan las gráficas de las señales que intervienen de forma directa en el subsistema Inner Control Loop, tal y como se mostraban en el apartado 3.4.

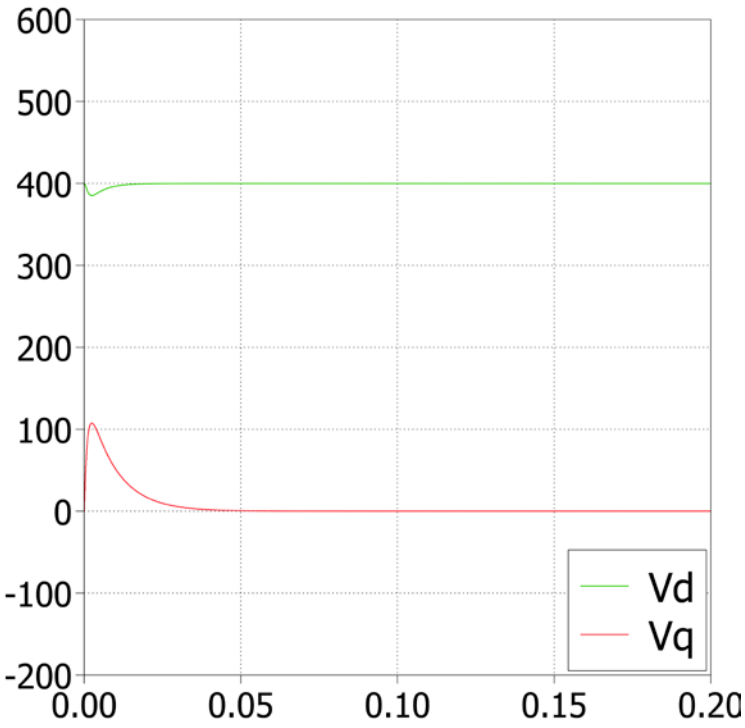

Figura 35. Tensiones en ejes dq en el modelo con c - scripts

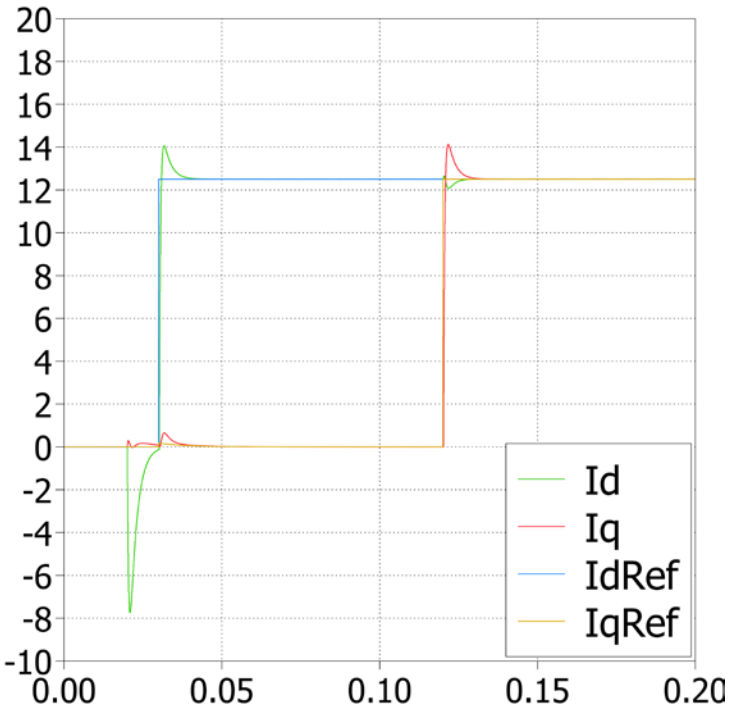

Figura 36. Corrientes en ejes dq junto con sus referencias en el modelo con c - scripts

Una vez sustituidos los bloques por c – scripts, se juntarán en un único c – script que simulará lo que posteriormente será el código en CCS, tal y como se puede ver en la siguiente figura.

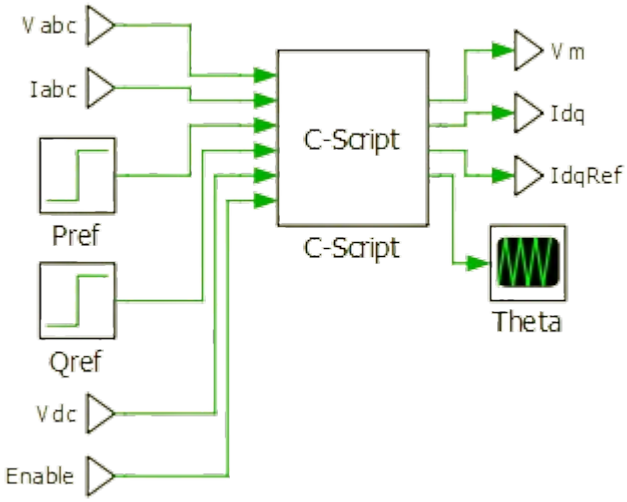

Figura 37. Modelado con un único c - script

De esta forma, quedan definidas las únicas entradas necesarias para implementar en CCS, así como las salidas V_m (para el cálculo de los disparos), I_{dq} y I_{dqRef} para comprobar y comparar que el control de corriente funciona correctamente.

En la siguiente página se muestra una captura del conjunto del modelo implementado con un único c – script. Puede verse de esta forma el cambio que ha sufrido el archivo de programa, necesario para unificar el código que posteriormente se usará (en una versión adaptada) para hacer PIL con la ayuda de los bloques correspondientes, el micro y Code Composer Studio.

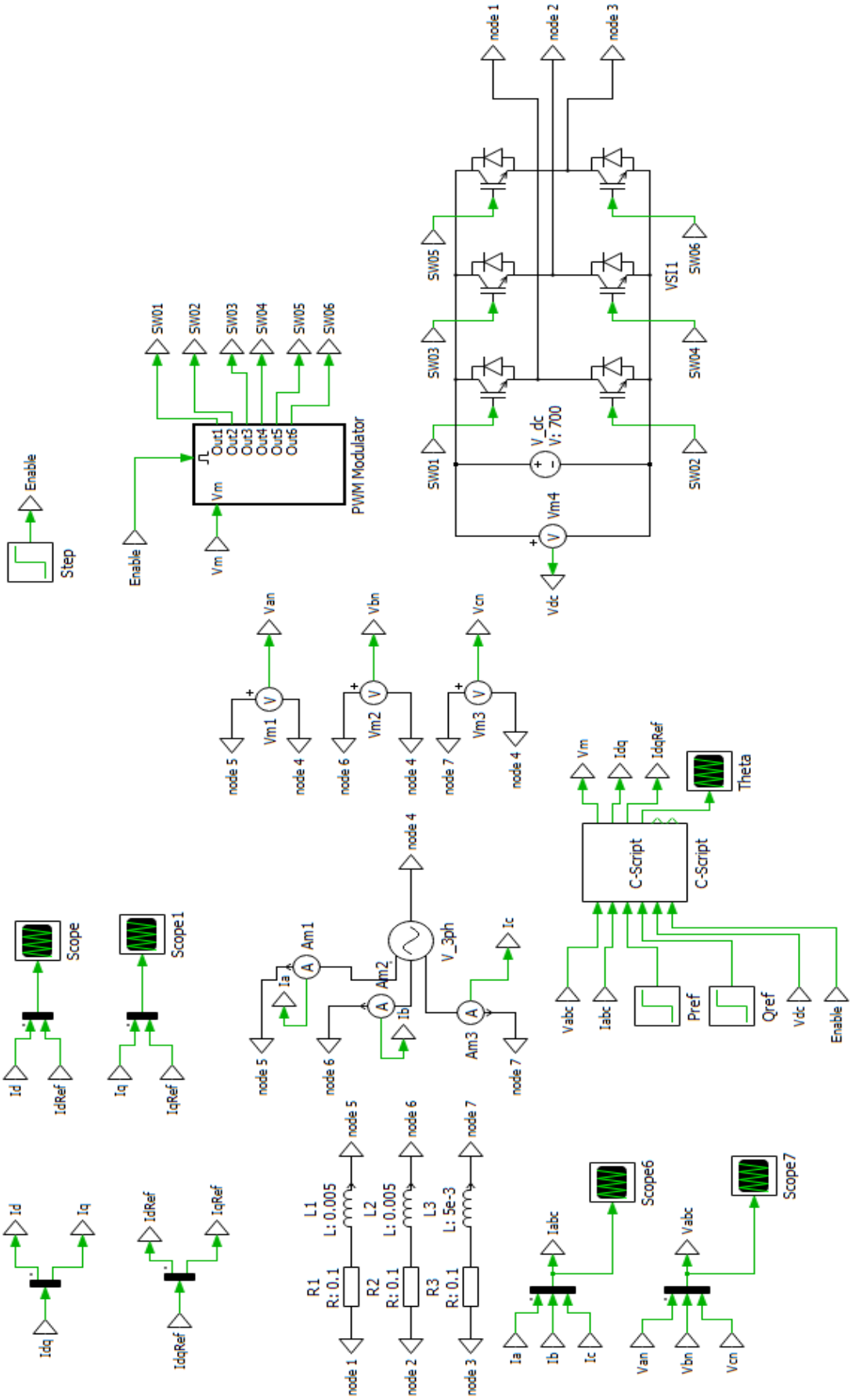

Figura 38. Captura del modelo de simulación implementado con un único c - script

4.2. Resultados del modelo con un único c – script

Siguiendo la estructura de capítulos anteriores, se muestran primero las gráficas correspondientes a las tensiones y Corrientes obtenidas de la red (abc).

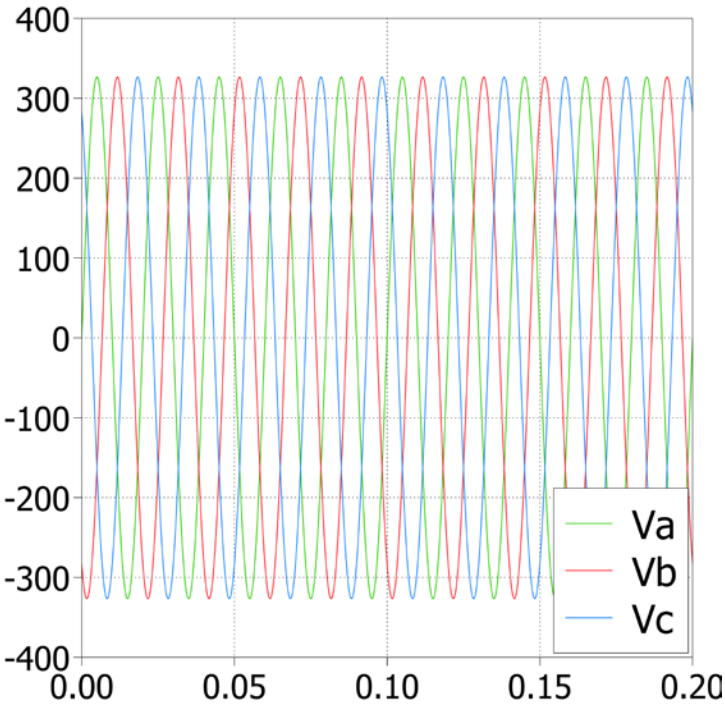

Figura 39. Tensiones obtenidas de la red en el modelo con único c - script

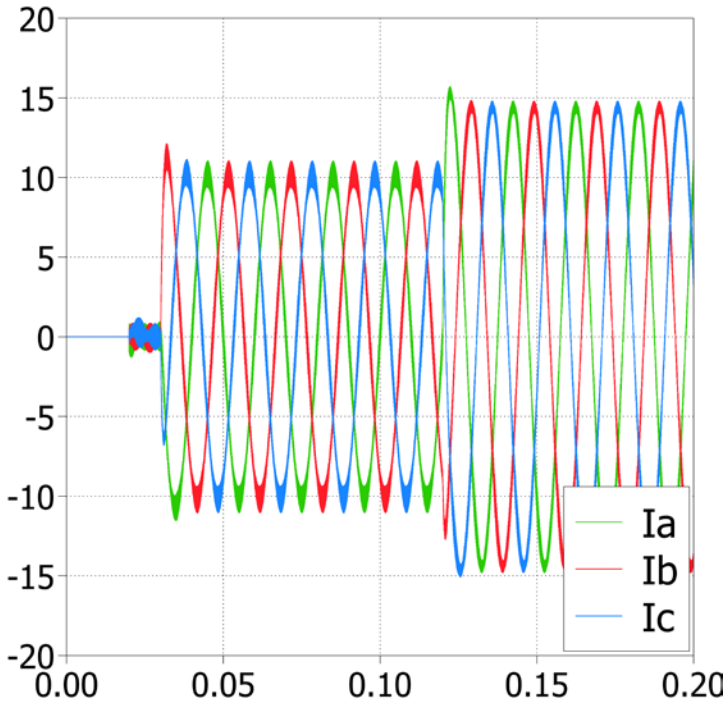

Figura 40. Corrientes obtenidas de la red en el modelo con único c - script

Se adjuntan por último las correspondientes a entradas del subsistema Inner Control Loop.

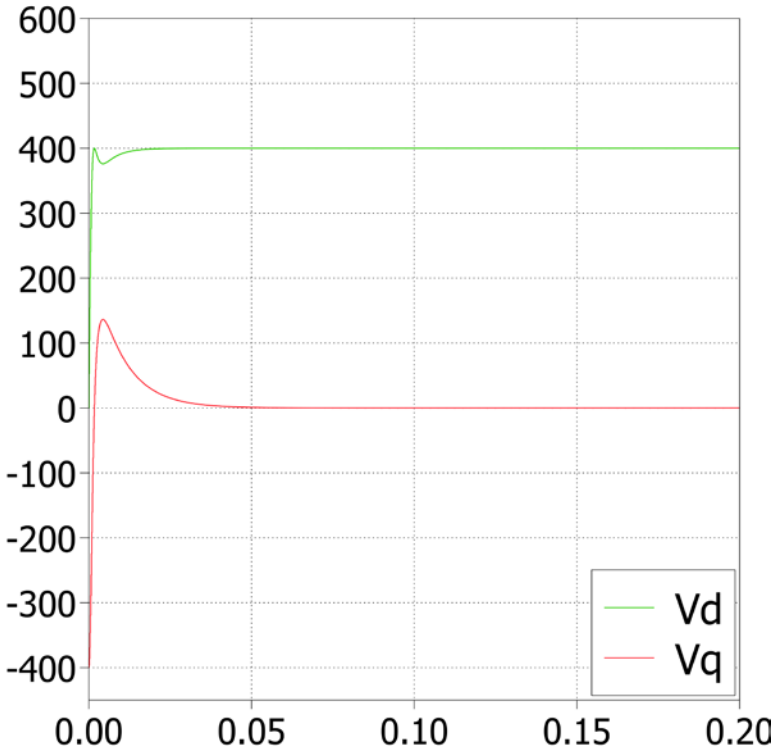

Figura 41. Tensiones en ejes dq en el modelo con único c - script

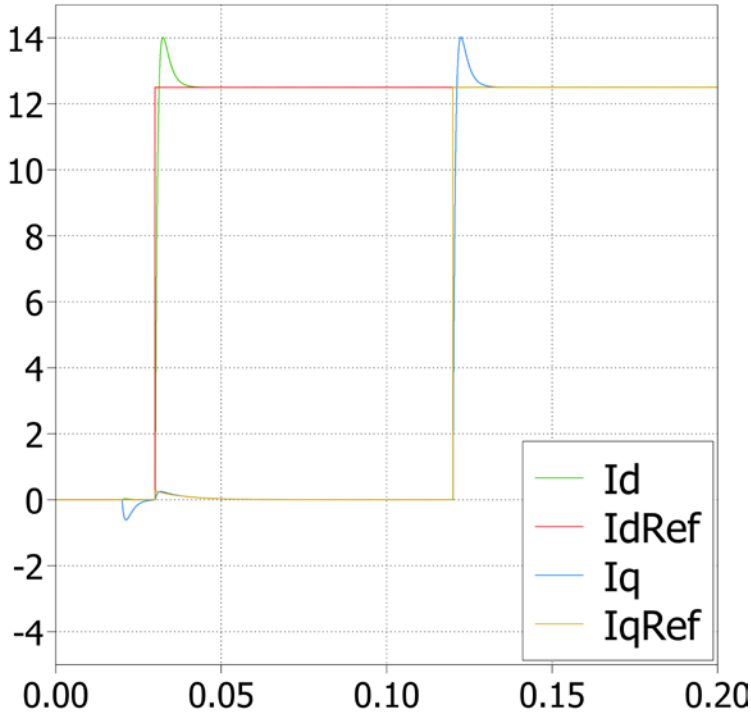

Figura 42. Corrientes en ejes dq junto con su referencia en el modelo con único c - script

5 PROCESSOR IN THE LOOP (PIL)

Processor in the Loop (en adelante PIL) es una técnica empleada cuando se quiere probar algoritmos de control en un sistema embebido real, es decir, tiene la ventaja frente a otras técnicas también bastante utilizadas (como SIL) de que es posible trabajar directamente con el hardware real. Esto facilita entre otras cosas la detección de fallos de software, como por ejemplo fallos de desbordamiento. Además, PIL permite, en lugar de leer el valor actual de los sensores del convertidor, usar valores calculados por la herramienta de simulación como entradas para el algoritmo empleado en el sistema real. De igual forma ocurre en las salidas, ya que se obtienen de dicho algoritmo de control, devolviéndose estas señales a la simulación. Cabe destacar que PIL no es una herramienta que trabaje en tiempo real debido a que necesita parar la ejecución del algoritmo de control mientras está leyendo dichas variables de entrada y salida del PIL.

Hablando con más detalle, esas variables de entrada a las que nos referíamos en el párrafo anterior se denominan *Override Probes*, y las de salida, *Read Probes*. Las *Override Probes* son aquellos valores de entrada al PIL como los medidos de corriente y voltaje que son eliminados y sustituidos por unos valores que proporciona la simulación en PLECS. De forma análoga ocurre con las *Read Probes*, que son leídas y devueltas a la simulación, como es el caso de los valores del registro del PWM.

Como bien se comentó al principio del capítulo, mientras se ajustan las *Override Probes* y se leen las *Read Probes*, debe detenerse el envío de algoritmos de control. Esto quiere decir que el control debe permanecer en pausa mientras PLECS se encarga de actualizar el modelo de simulación. En otras palabras, el algoritmo de control opera en cada paso de simulación PIL, aunque mientras se está ejecutando funciona como una simulación en tiempo real. Es por este motivo que se dice que tiene un comportamiento en pseudo – tiempo real.

En definitiva, se puede resumir el funcionamiento del PIL con la siguiente ilustración.

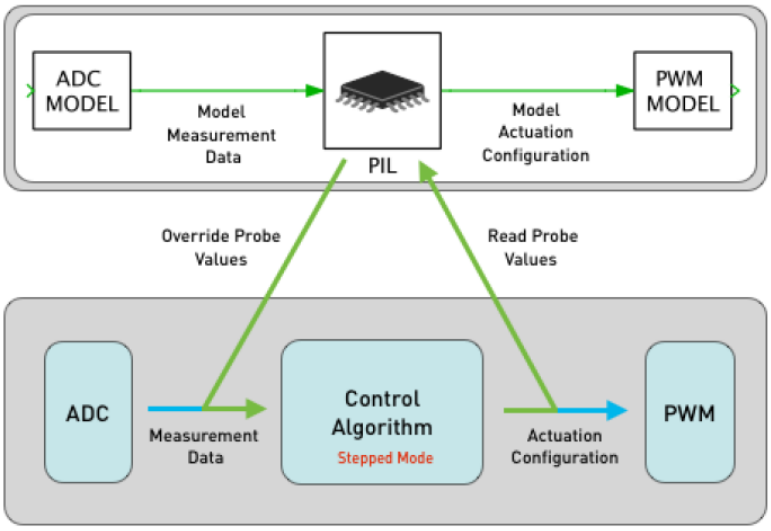

Figura 43. Esquema del funcionamiento del PIL [15]

Este capítulo trata del uso de PIL para el sistema bajo estudio, mostrándose además las modificaciones que son necesarias realizar para conseguir el correcto funcionamiento del sistema con este método. Así, una vez realizado el modelado del capítulo anterior, se organiza el nuevo modelo de programa en base a dos subsistemas: el primero, denominado “Controller”, es el encargado de recibir como entradas las magnitudes medidas de la red, y tiene como salidas las señales del PWM. Además, contiene al otro subsistema, llamado “Controller_logic”, el cual contiene el bloque PIL.

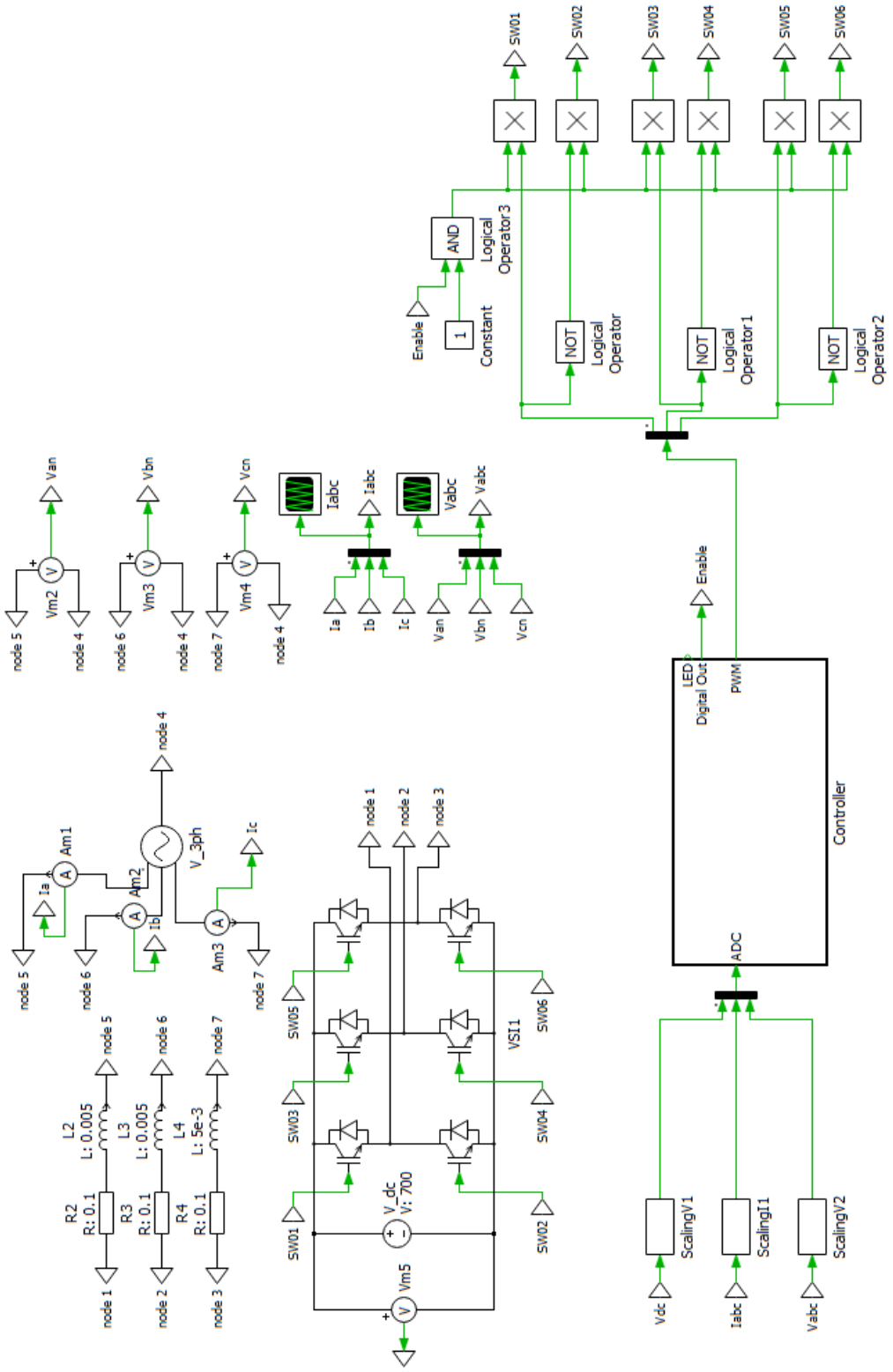

Figura 44. Captura del modelo de simulación usando PIL

Como puede verse en la Figura 44, el subsistema “Controller” que se explica a continuación corresponde con el único c – script del modelo del capítulo anterior.

5.1. Subsistema “Controller”

El primer subsistema, denominado “Controller”, cuyas entradas son las tensiones V_{dc} y V_{abc}, junto con la intensidad I_{abc}, se encarga de recibir las medidas de las corrientes y tensiones de la red, así como la tensión continua, con el objetivo de poder usarlas en los cálculos del control de corriente, como ya se ha explicado con anterioridad en apartados previos a este. Las salidas de este subsistema serán las señales del PWM y una señal de “enable” que se utilizará para habilitar los disparos a partir del instante 0.02 segundos.

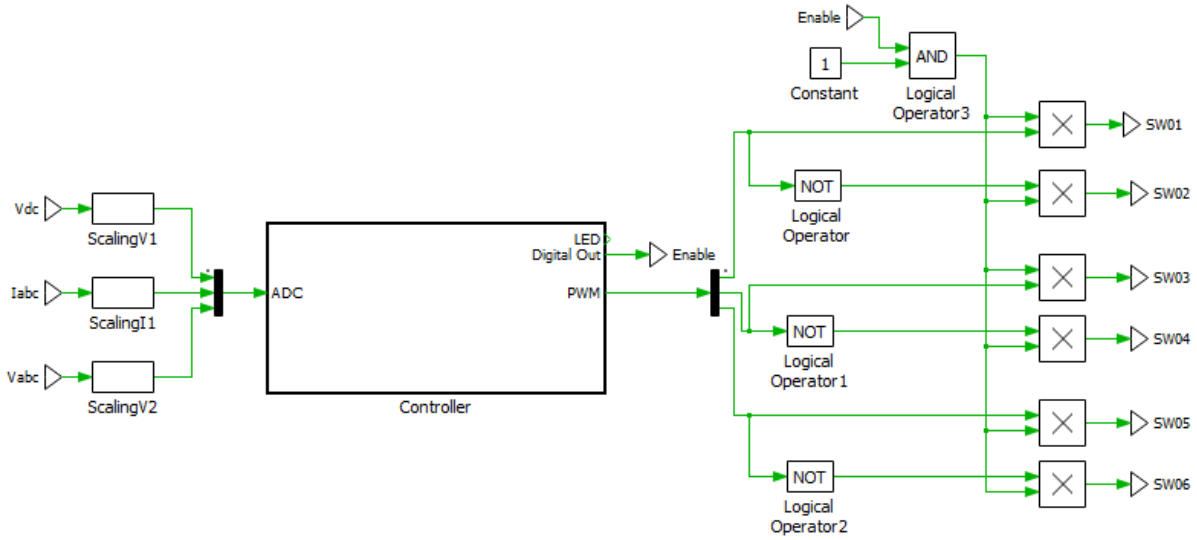

Figura 45. Vista externa del subsistema Controller

Este subsistema contiene en su interior todo lo relativo al PIL expuesto en la primera página de este capítulo. Tal y como se puede observar en la Figura 40. Interior del subsistema “Controller”, las entradas antes mencionadas, es decir, tensiones y corrientes de la red junto con la tensión en continua, son leídas por un bloque ADC que conecta con otro subsistema llamado “Control_logic”. Estas entradas son leídas por el bloque “Override Probes”, permitiendo utilizar sus valores para la simulación de PIL. El subsistema “Control_logic” cuenta con tres entradas más, las cuales se corresponden con las potencias activa y reactiva, además del “enable” mencionado.

Pese a observar varias salidas del subsistema “Control_logic”, las únicas que coinciden con el subsistema ahora bajo estudio son las correspondientes a las señales obtenidas del bloque PWM, además de la señal de “enable”, que sale de este gracias al bloque “Digital Out”, el cual permite obtener esta señal mediante el GPIO39. El resto de las salidas, al no ser necesarias para el funcionamiento de la estructura principal del programa, no es necesario sacarlas de este subsistema (es decir, sirven únicamente como medio de comprobación visual de las señales deseadas).

5.2. Subsistema “Control_logic”

El subsistema “Control_logic” actúa de una forma parecida al de la Figura 37. Modelo con un único c - script, con la diferencia de que se sustituye el c – script por un bloque PIL, en el cual se basa este trabajo. Por lo tanto, continuando con la comparación y similitudes con respecto al modelo anterior, se observan prácticamente las mismas entradas (mencionadas en el apartado anterior) y salidas: corriente en ejes dq, corriente de referencia en ejes dq y tensión de referencia utilizada para el modulador PWM, además de añadirse como salida digital la señal de “enable” para habilitar los disparos.

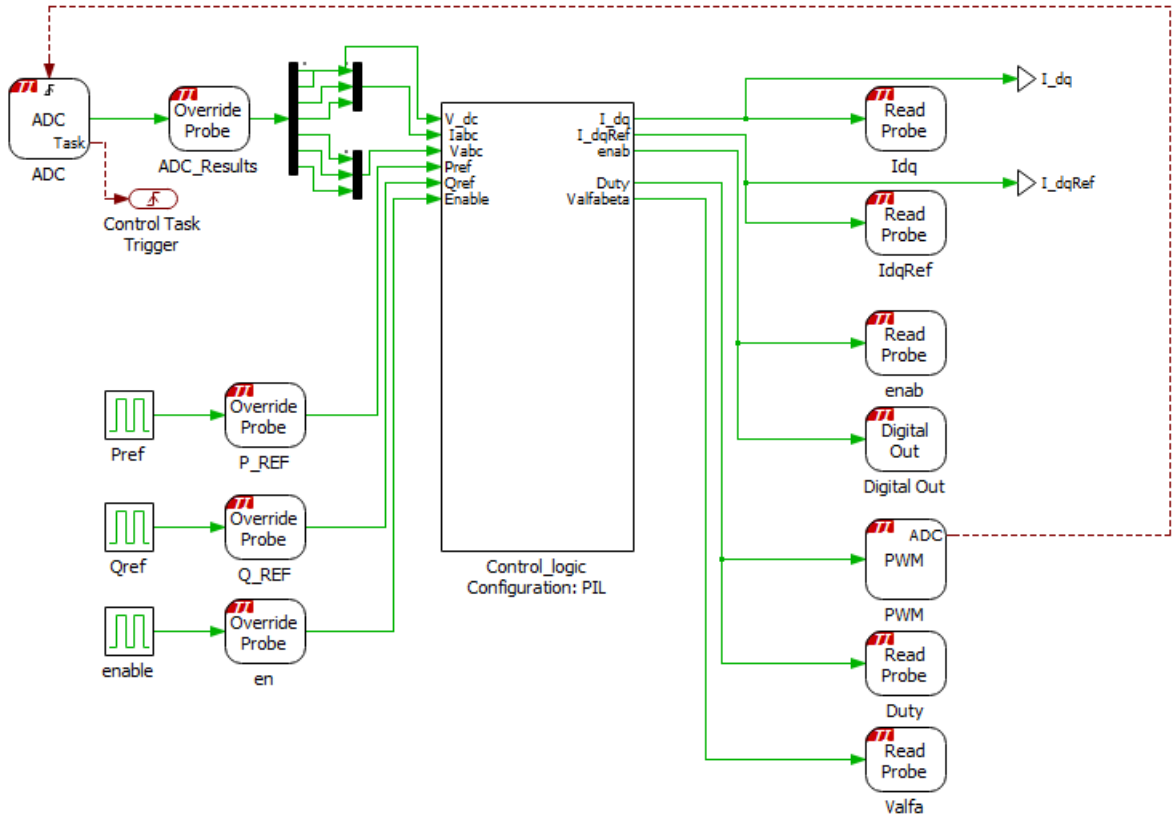

Figura 46. Vista interna del subsistema Controller

Se explican a continuación la configuración de las entradas al subsistema "Control_logic", mostradas en la Figura 46 :

Para el microcontrolador utilizado (MCU TI28069) se comprobó que tiene dos ADCs, el ADCINAx y el ADCINBx. El manual del micro especifica que para los canales correspondientes al ADCINBx, se debe configurar en el bloque ADC de la forma "ADC unit: ADC A", permitiendo escribir los canales del ADCINAx directamente y los canales correspondientes al ADCINBx como los del ADC "A" sumados a un offset de 8. Por tanto, en la pestaña de *Simulation → Simulation Parameters → Initialization* pueden verse los canales ADC utilizados, siendo estos los siguientes: [3, 3+8, 5+8, 2+8, 0, 2, 5], donde el primero se encarga de la lectura de V_{dc}, los tres siguientes para la lectura de las componentes de I_{abc} (véase que están sumados por 8, por lo que corresponderían con ADCINB3, ADCINB5 y ADCINB2) y los últimos tres corresponden a las componentes de V_{abc}. La salida de este bloque ADC se conecta con la entrada de un bloque "Override Probe", para posteriormente separar de nuevo estas entradas para poder usarlas de forma independiente en el subsistema "Control_logic".

Dentro de "Controller" se observó que los bloques "Step" no servían, por lo que se optó por la solución de usar bloques "Pulse Generator", de una frecuencia que resultase lo suficientemente grande en periodo para que superase por bastante el tiempo de simulación, de forma que no se repitiese el pulso, evitando así que pudiese generar problemas. De esta forma, se muestra en la siguiente figura la configuración de los tres bloques de este tipo utilizados, en orden (de izquierda a derecha): P_{ref}, Q_{ref} y enable.

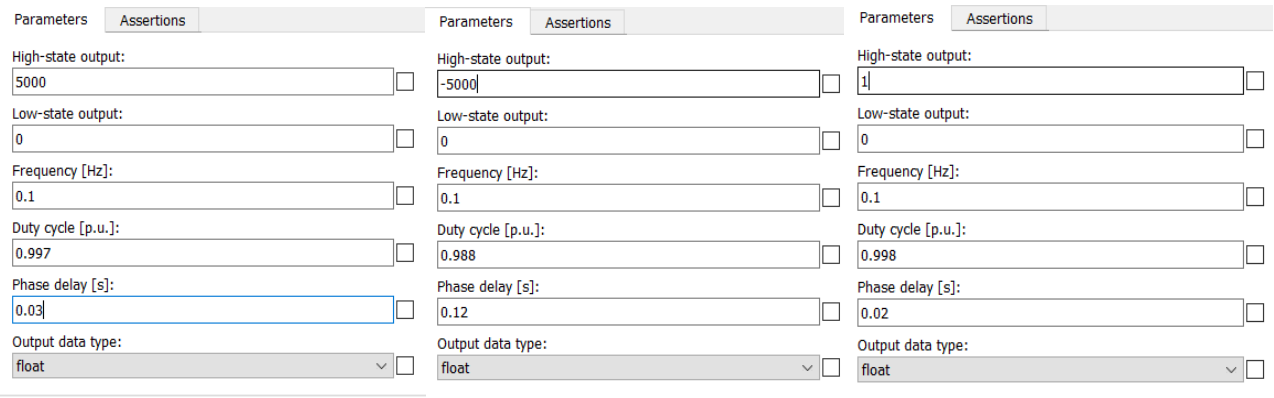

Figura 47. Configuración "Pulse Generator": Pref, Qref y enable

Una vez explicadas las entradas, se pasa a continuación a la explicación de las salidas que componen este subsistema.

Las salidas del bloque “Control_logic” se han escogido razonadamente para poder visualizar mediante “Scopes” el correcto funcionamiento del programa, comprobando que se obtengan las señales esperadas. De este modo, se crean “Read Probes” para I_{dq}, I_{dqRef}, $V_{\alpha\beta}$ y las señales de disparo de los IGBTs, pudiéndose crear otras distintas en caso de que se quiera observar el comportamiento de otras señales. Además, sale también la señal de enable como “Digital Out” para poder usarse fuera en la habilitación de esos disparos (véase Figura 39. Subsistema "Controller"

5.3. Modulador PWM

Como ya se habrá podido observar, una de las salidas del subsistema “Control_logic” está conectada a un bloque PWM. Si se observa el código utilizado en CCS, podrá verificarse que la salida del subsistema es una señal con tres componentes de tensión de referencia, los cuales se utilizaron en modelos anteriores para comparar con una portadora triangular y así realizar la llamada “Modulación por anchura de pulsos”, obteniendo de este modo los disparos de los 6 IGBTs de los que se compone el convertidor trifásico (2 niveles).

Al igual que con el ADC, en la pestaña de *Simulation* → *Simulation Parameters* → *Initialization* es necesario especificar los canales que utilizará el PWM, seleccionando para este caso los siguientes: [4 5 2].

En esta última versión del modelado del sistema, cabe destacar el bloque PWM por lo siguiente: en modelos anteriores, se esperaba una señal que debía ser modulada en el código o mediante bloques, y que se encontraba entre -1 y 1 debido a la portadora triangular usada en dichos casos. Ahora, con el uso de este bloque, la señal se obtiene ya modulada entre 0 y 1. Es por eso que el código original (el usado en el c – script) tuvo que ser modificado para cuadrar con dicha señal. Es decir, se ha adaptado el programa a los nuevos cambios impuestos por este bloque.

Importante también comentar que como las señales obtenidas eran las correspondientes a una tensión V_{abc} de referencia (es decir, 3 componentes), es necesario obtener la señal negada para así dar señal a todos los IGBTs, obteniendo así las 6 señales necesarias. El funcionamiento es prácticamente idéntico al de modelos anteriores con la salvedad de que ahora, en lugar de comparar con la portadora triangular (se recuerda que se comparaba una señal de tensión de referencia llamada V_m), únicamente es necesario habilitar los disparos en el instante deseado con el enable, tal y como se ve en la Figura 39. Subsistema "Controller", ya que esa comparación se encarga de hacerla el bloque PWM.

5.4. Etapa de adaptación de señales

Antes de poner en marcha el programa, es necesario realizar una etapa de adaptación de las señales que entran en el ADC. Para ello se utilizarán los bloques de PLECS llamados "Scaling", que calculan la salida de este bloque como $input * scale + offset$. De esta manera, se consigue adaptar las señales en el rango de tensiones entre las cuales trabaja el microcontrolador, esto es, de 0V hasta 3.3V.

Por tanto, teniendo en cuenta únicamente el valor de "scale" (es decir, la ganancia) y poniendo el offset a cero, se calcularía el valor de entrada al ADC como

$$X_{ADC} = K * X_{in} \qquad (5.1.)$$

Siendo X_{ADC} el valor que se quiere obtener como entrada al ADC, X_{in} el valor de entrada a la etapa de adaptación de señal, es decir, el valor real de tensiones y Corrientes que se obtienen en la red, y K la ganancia por la cual se multiplicará al valor anterior. El valor de esta ganancia depende de la magnitud que se vaya a transformar, siendo $K = 3.3/X\_max$, donde X_{max} es el valor máximo de tensión o corriente en cada caso. En realidad, no se ha escogido el valor máximo exacto si no un valor entero superior, puesto que los flotantes pierden precision.

Así, las ganancias que se usarán son las siguientes:

- Tensión V_{dc}: ganancia K = 3.3/700
- Tensiones V_{abc}: ganancia K = 3.3/330
- Corrientes I_{abc}: ganancia K = 3.3/16

Tras esto, es necesario deshacer dichas multiplicaciones para que el bloque PIL realice los cálculos con los valores reales. Es por ello que el bloque ADC permite introducir valores de "scale" y "offset", siendo necesario únicamente introducir en los primeros la inversa de las ganancias calculadas anteriormente, quedando como se observa en la siguiente figura.

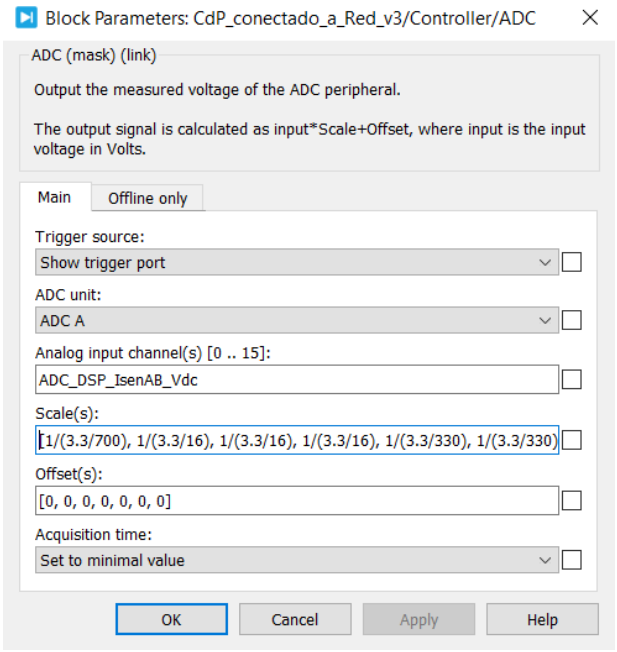

Figura 48. Configuración ADC

5.5. Ejecución del programa y Code Composer Studio

Como ha podido comprobarse, Code Composer Studio es un entorno de desarrollo para la programación de aplicaciones creado por Texas Instrument. Este entorno es el que usará para la integración del microcontrolador en el trabajo.

Para la ejecución del archivo de PLECS es necesario conocer algunos aspectos más de los subsistemas y bloques que lo componen. En primer lugar, el subsistema "Control_logic" tiene dos configuraciones posibles: Codegen y PIL. La primera configuración es la que se selecciona cuando se quiere generar código en CCS, mientras que con la segunda se consigue que se ejecute el programa haciendo PIL.

Introducido este apartado, se explican los pasos seguidos para la ejecución del programa. En este caso, si se abre el subsistema "Control_logic", la pestaña correspondiente a Codegen se dejará vacía, quedando únicamente las entradas y salidas originadas mediante las "Override Probes" y "Read Probes", puesto que a priori solamente interesa que se generen las entradas y salidas. Posteriormente, en la pestaña *Coder → Coder Options → General* se definirá, entre otras cosas, el tamaño del paso de discretización (100 μs), y después en la pestaña *Target*, dentro de *Coder Options*, se elegirá el micro que se utilizará (28069), un directorio donde se creará una carpeta, siendo el nombre de esta el nombre del archivo seguido de "_Codegen" y además es necesario seleccionar como *Build Type* la opción "Generate Code into CCS project". Se procede finalmente a hacer click sobre *Build*.

Una vez generado ese código en CCS (es necesario haber creado un proyecto en CCS tal y como se especifica en el *manual c2000* [12]), puede observarse el archivo "Controller.c" que puede verse en el apartado de Anexos, y corresponde con uno de los archivos de código que se generan tras seguir los pasos anteriores. Este archivo se genera prácticamente vacío, o mejor dicho, sin funcionalidad alguna. Es deber del alumno modificarlo y meter el código correspondiente entre los comentarios de código que indican que ahí se debe escribir, introduciéndose por tanto una adaptación del código del c – script de la Figura 37 en el que estarán los cálculos necesarios para el correcto funcionamiento del programa. Añadir también que se modificó el archivo "Controller.h", usándose como archivo de cabecera para declarar variables auxiliares. Una vez depurado el código, comprobando que no se tiene ningún error, se procede a pulsar "Run" y "Debug" y se vuelve a PLECS, donde ahora se seleccionará la otra opción de configuración del subsistema, esto es, PIL.

Ahora PLECS permite la adición de las entradas y salidas al bloque PIL, pudiendo así conectar los bloques "Signal Inport" y "Signal Outport" con dicho bloque, siguiendo el orden que se le haya definido para cada entrada y salida. Dentro de la pestaña correspondiente, clickando sobre el bloque PIL, se selecciona el "Target" y el puerto correspondiente al que esté conectado el dispositivo, y en la pestaña *Properties* se debe cambiar a la opción "Ready for PIL". Tras realizar estos pasos (se encuentran explicados en el *manual c2000* [12] y en el pdf de ejemplo *simple_PIL_model* [13]) se procede con la simulación.

Todos estos pasos mencionados se pueden verificar en los manuales que ofrece PLECS y que se encuentran en su página web (se facilitan la referencia de éstos en *Referencias*).

6 RESULTADOS

Una vez implementado el proyecto en su totalidad, y tras ejecutar el mismo, se puede verificar visualmente mediante los bloques llamados “Scopes” la correcta forma de las señales que se desean estudiar. Para poder hacer una comparación con los modelos anteriormente expuestos, se muestran a continuación las señales correspondientes a las corrientes en ejes síncronos dq, así como de sus referencias. En la columna de la izquierda se mostrarán los resultados obtenidos por SIL (es decir, los obtenidos únicamente por la simulación de bloques en PLECS, usando para ello el modelo de un único c – script) y a la derecha se mostrarán los obtenidos por PIL en este último capítulo.

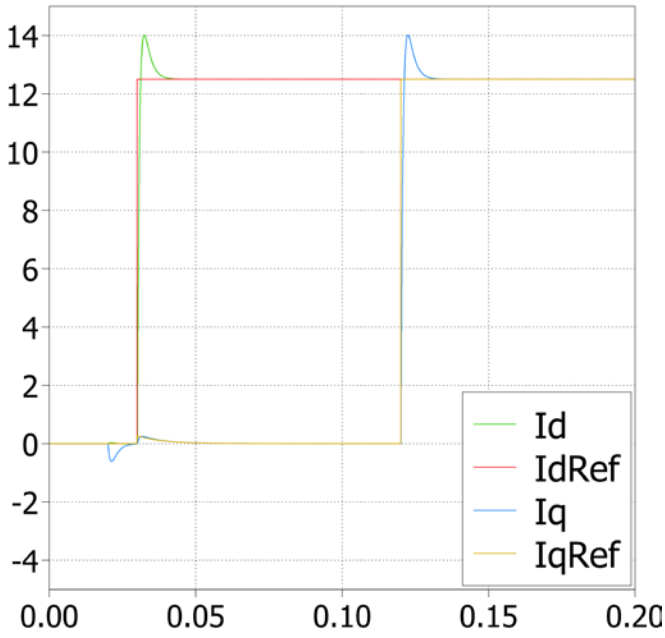

Figura 49. Resultados obtenidos en corrientes en ejes dq del modelo SIL (un único c - script)

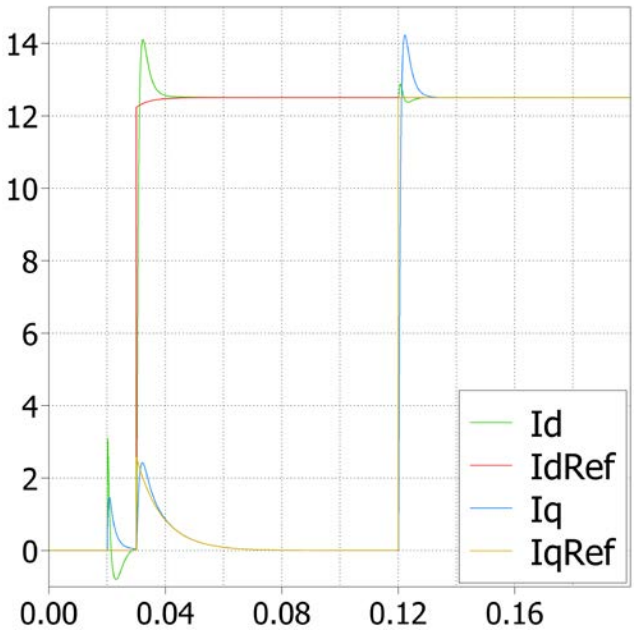

Figura 50. Resultados obtenidos en corrientes en ejes dq del modelo PIL

Se adjuntan a continuación las gráficas correspondientes a las tensiones en ejes síncronos dq, con la misma estructura que las anteriores: a la izquierda las obtenidas en el modelo de un único c – script y a la derecha las obtenidas en el modelo PIL.

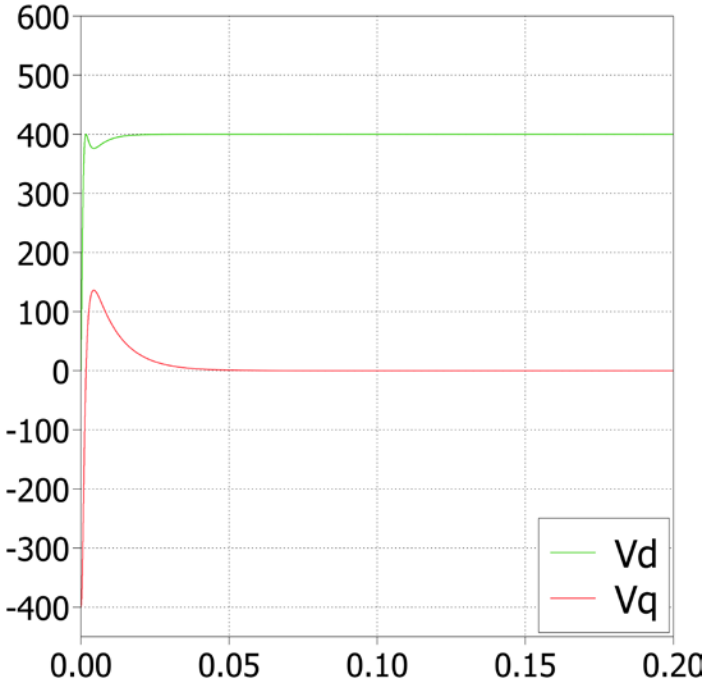

Figura 51. Resultados obtenidos de tensiones en ejes dq para el modelo SIL (un único c - script)

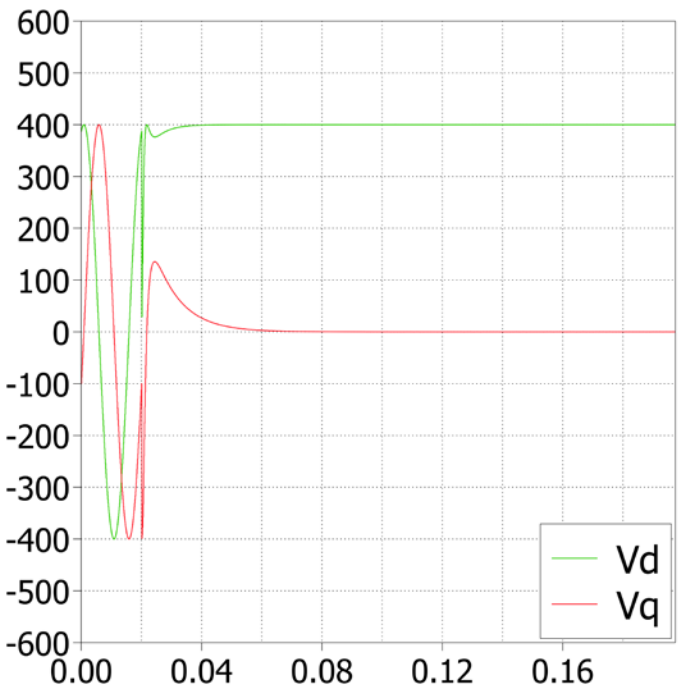

Figura 522. Resultados obtenidos de tensiones en ejes dq para el modelo PIL

Tal y como puede observarse, PIL consigue cumplir con los objetivos, obteniéndose mediante este sistema unos muy buenos resultados comparados con las simulaciones iniciales, a pesar de mostrar las pequeñas oscilaciones que se muestran al principio de las gráficas (sobre todo cuando ocurren cambios de referencia, algo relativamente común en simulaciones con elementos reales).

7 CONCLUSIÓN Y TRABAJOS FUTUROS

La forma en que se realiza este TFG permite pensar en algunas mejoras futuras que podrían servir como ampliación de este. Entre estas, está el uso del bus CAN o del SPI para transmitir mensajes al microcontrolador. Para este caso, se pensó que sería buena idea para un futuro implementar el bus CAN con los bloques que proporciona PLECS (CAN Port, CAN Pack, CAN Unpack, CAN Transmit y CAN Recieve) de modo que fuese posible enviar las señales de referencia de la potencia activa y reactiva, así como el enable, desde la vista principal del programa, es decir, sacarlas del subsistema "Controller".

El problema que se encontró es que para poder probar los ejemplos que proporciona el programa y poder hacer pruebas, era necesario disponer de otro LaunchPad, para establecer la conexión entre ambos, y no era posible hacer dicha comunicación entre uno solo. Es por esto por lo que se deja abierta esta posibilidad para implementar en un futuro próximo, explotando así las posibilidades y características que aporta el programa y este microcontrolador.

Como bien se ha ido explicando, el poder hacer PIL permite realizar la simulación de plantas de electrónica de potencia de grandes magnitudes sin la necesidad de tenerlas delante usando para ello una simulación con el sistema embebido. Esto permite analizar el comportamiento de estas, así como la detección de errores y fenómenos no deseados, con el objetivo de poder solucionarlos antes de llevarlos a la planta real, siendo por tanto una solución más económica. De esta forma, con el estudio realizado en este trabajo se ha podido aprender sobre este tipo de alternativas y se pretende que sirva para que más personas investiguen sobre el tema, con el fin de mejorar y avanzar en este campo de la electrónica.

8 ANEXOS

Anexo A. Archivos Code Composer Studio

Controller.c

```
/*
 * Implementation file for: CdP_conectado_a_Red_v3/Controller
 * Generated with         : PLECS 4.6.8
 *                          TI2806x 1.5.9
 * Generated on           : 28 Aug 2024 13:02:23
 */
#include "Controller.h"
#ifndef PLECS_HEADER_Controller_h_
#error The wrong header file "Controller.h" was included. Please check your
#error include path to see whether this file name conflicts with the name
#error of another header file.
#endif /* PLECS_HEADER_Controller_h_ */
#if defined(__GNUC__) && (__GNUC__ > 4)
#   define _ALIGNMENT 16
#   define _RESTRICT __restrict
#   define _ALIGN __attribute__((aligned(_ALIGNMENT)))
#   if defined(__clang__)
#      if __has_builtin(__builtin_assume_aligned)
#         define _ASSUME_ALIGNED(a) __builtin_assume_aligned(a, _ALIGNMENT)
#      else
#         define _ASSUME_ALIGNED(a) a
#      endif
#   else
#      define _ASSUME_ALIGNED(a) __builtin_assume_aligned(a, _ALIGNMENT)
#   endif
#else
#   ifndef _RESTRICT
#      define _RESTRICT
#   endif
#   ifndef _ALIGN
#      define _ALIGN
#   endif
#   ifndef _ASSUME_ALIGNED
#      define _ASSUME_ALIGNED(a) a
#   endif
#endif
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
#include <string.h>
#include "plx_hal.h"
#include <stdlib.h>
```

```
#define PLECSRunTimeError(msg) Controller_errorStatus = msg
static const uint32_t Controller_subTaskPeriod[3]= {
   /* [0.01, 0], Base task */
   100,
   /* [0.02, 0], Base task */
   200,
   /* [0.04, 0], Base task */
   400
};
static uint32_t Controller_subTaskTick[3];
static char Controller_subTaskHit[3];
static const float Controller_UNCONNECTED = 0;
static uint32_t Controller_D_uint32_t[3];
static uint32_t Controller_tickLo[2];
static int32_t Controller_tickHi[2];
Controller_BlockOutputs Controller_B;
#if defined(EXTERNAL_MODE) && EXTERNAL_MODE
const float * const Controller_ExtModeSignals[] = {
   &Controller_UNCONNECTED,
   &Controller_UNCONNECTED,
   &Controller_UNCONNECTED,
   &Controller_UNCONNECTED,
   &Controller_B.Q_REF,
   &Controller_B.P_REF,
   &Controller_B.Q_REF,
   &Controller_B.P_REF,
   &Controller_B.ADC_Results[0],
   &Controller_B.en,
   &Controller_UNCONNECTED,
   &Controller_UNCONNECTED,
   &Controller_UNCONNECTED,
   &Controller_UNCONNECTED,
   &Controller_UNCONNECTED,
   &Controller_UNCONNECTED
};
#endif /* defined(EXTERNAL_MODE) */
Controller_ModelStates Controller_X _ALIGN;
const char * Controller_errorStatus;
const float Controller_sampleTime[2][2] = {
   /* Task "Base task" */
   {0.0001f, 0.f},
   /* Task "Blink task" */
   {0.5f, 0.f}
};
const char * const Controller_checksum =
   "f7775154e77062818b4469595e41a2e4264e052f";
/* Target declarations */
extern void Controller_initHal();
/* Override Probe : 'Controller/Q_REF' */
ControllerProbes_t Controller_probes;

/*Declarar estructuras de variables auxiliares*/
aux1_var aux1;
aux2_var aux2;
```

```
void Controller_initialize(float time)
{
   float remainder;
   Controller_errorStatus = NULL;
   Controller_tickHi[0] = floor(
                                time/
                                (4294967296.0*Controller_sampleTime[0][0]));
   remainder = time - Controller_tickHi[0]*4294967296.0*
               Controller_sampleTime[0][0];
   Controller_tickLo[0] = floor(remainder/Controller_sampleTime[0][0] + .5);
   remainder -= Controller_tickLo[0]*Controller_sampleTime[0][0];
   if (fabsf(remainder) > 1e-6*fabsf(time))
   {
      Controller_errorStatus =
         "Start time must be an integer multiple of the base sample time.";
   }
   /* Initialize sub-task tick counters */
   Controller_subTaskTick[0] = 0;   /* Base task, [0.01, 0] */
   Controller_subTaskTick[1] = 0;   /* Base task, [0.02, 0] */
   Controller_subTaskTick[2] = 0;   /* Base task, [0.04, 0] */


   /* Offset sub-task tick counters */
   {
      uint32_t i, n, N, delta;
      N = abs(Controller_tickHi[0]);
      for (i = 0; i < 3; ++i)
      {
         delta = -Controller_subTaskPeriod[i];
         delta %= Controller_subTaskPeriod[i];
         if (Controller_tickHi[0] < 0)
         {
            delta = Controller_subTaskPeriod[i] - delta;
         }
         for (n = 0; n < N; ++n)
         {
            Controller_subTaskTick[i] =
               (Controller_subTaskTick[i] +
                delta) % Controller_subTaskPeriod[i];
         }
         Controller_subTaskTick[i] =
            (Controller_subTaskTick[i] + Controller_tickLo[0] %
             Controller_subTaskPeriod[i]) % Controller_subTaskPeriod[i];
      }
   }

   /* Target pre-initialization */
   Controller_initHal();


   /* Initialization for Pulse Generator : 'Controller/Qref' */
   Controller_D_uint32_t[0] =
      (((int32_t) floor(time/0.04f+.5) - 3) % 250 + 250) % 250;
```

```
    /* Initialization for Pulse Generator : 'Controller/Pref' */
    Controller_D_uint32_t[1] =
       (((int32_t) floor(time/0.01f+.5) - 3) % 1000 + 1000) % 1000;

    /* Initialization for Pulse Generator : 'Controller/enable' */
    Controller_D_uint32_t[2] =
       (((int32_t) floor(time/0.02f+.5) - 1) % 500 + 500) % 500;

    /* Initialization for Delay : 'Controller/Delay2' */
    Controller_X.Delay2 = 0.f;

    /*Inicializar variables auxiliares*/
            aux2.Ts = 100e-6;
            aux2.Theta = 0.0;
            aux2.ik_1 = 0.0;
            aux2.ik_1d = 0.0;
            aux2.ik_1q = 0.0;
            aux2.Zeta_1 = 0.0;

            aux1.kp = 2.5;
            aux1.ki = 250.0;
            aux1.kpRef = 8.0;
            aux1.kiRef = 3000.0;
}

void Controller_step(int task_id)
{
    if (Controller_errorStatus)
    {
       return;
    }
    switch(task_id)
    {
    case 0: /* Task "Base task" */
    {
       {
          size_t i;
          for (i = 0; i < 3; ++i)
          {
             Controller_subTaskHit[i] = (Controller_subTaskTick[i] == 0);
          }
       }
       if (Controller_subTaskHit[2])
       {
          /* Pulse Generator : 'Controller/Qref' */
          Controller_B.Qref = Controller_D_uint32_t[0] < 247 ? -5000.f : 0.f;
       }
       /* Override Probe : 'Controller/Q_REF' */
       SET_OPROBE(Controller_probes.Q_REF, Controller_B.Qref);
       Controller_B.Q_REF = Controller_probes.Q_REF;
       if (Controller_subTaskHit[0])
       {
          /* Pulse Generator : 'Controller/Pref' */
          Controller_B.Pref = Controller_D_uint32_t[1] < 997 ? 5000.f : 0.f;
```

```
        }
        /* Override Probe : 'Controller/P_REF' */
        SET_OPROBE(Controller_probes.P_REF, Controller_B.Pref);
        Controller_B.P_REF = Controller_probes.P_REF;
        if (Controller_subTaskHit[1])
        {
           /* Pulse Generator : 'Controller/enable' */
           Controller_B.enable = Controller_D_uint32_t[2] < 499 ? 1.f : 0.f;
        }
        /* Override Probe : 'Controller/en' */
        SET_OPROBE(Controller_probes.en, Controller_B.enable);
        Controller_B.en = Controller_probes.en;

        /* Override Probe : 'Controller/ADC_Results' */
        SET_OPROBE(Controller_probes.ADC_Results_1, PLXHAL_ADC_getIn(0, 0));
        SET_OPROBE(Controller_probes.ADC_Results_2, PLXHAL_ADC_getIn(0, 1));
        SET_OPROBE(Controller_probes.ADC_Results_3, PLXHAL_ADC_getIn(0, 2));
        SET_OPROBE(Controller_probes.ADC_Results_4, PLXHAL_ADC_getIn(0, 3));
        SET_OPROBE(Controller_probes.ADC_Results_5, PLXHAL_ADC_getIn(0, 4));
        SET_OPROBE(Controller_probes.ADC_Results_6, PLXHAL_ADC_getIn(0, 5));
        SET_OPROBE(Controller_probes.ADC_Results_7, PLXHAL_ADC_getIn(0, 6));
        Controller_B.ADC_Results[0] = Controller_probes.ADC_Results_1;
        Controller_B.ADC_Results[1] = Controller_probes.ADC_Results_2;
        Controller_B.ADC_Results[2] = Controller_probes.ADC_Results_3;
        Controller_B.ADC_Results[3] = Controller_probes.ADC_Results_4;
        Controller_B.ADC_Results[4] = Controller_probes.ADC_Results_5;
        Controller_B.ADC_Results[5] = Controller_probes.ADC_Results_6;
        Controller_B.ADC_Results[6] = Controller_probes.ADC_Results_7;

        /* PWM  : 'Controller/PWM' */
        {
           PLXHAL_PWM_setDuty(0, Controller_UNCONNECTED);
        }
        {
           PLXHAL_PWM_setDuty(1, Controller_UNCONNECTED);
        }
        {
           PLXHAL_PWM_setDuty(2, Controller_UNCONNECTED);
        }

------------------------------ /*CODIGO ALUMNO*/ ----------------------------

// ASIGNACION VARIABLES CODE COMPOSER A LAS USADAS EN EL CODIGO DEL ALUMNO
                aux1.Vdc = Controller_B.ADC_Results[0];
                aux1.Ia = Controller_B.ADC_Results[1];
                aux1.Ib = Controller_B.ADC_Results[2];
                aux1.Ic = Controller_B.ADC_Results[3];
                aux1.Va = Controller_B.ADC_Results[4];
                aux1.Vb = Controller_B.ADC_Results[5];
                aux1.Vc = Controller_B.ADC_Results[6];
                aux1.Pref = Controller_B.P_REF;
                aux1.Qref = Controller_B.Q_REF;
                aux1.enable = Controller_probes.en;
```

```
// VARIABLES DE SALIDA DEL CODE COMPOSER ASIGNADAS CON LAS DEL ALUMNO

                        Controller_probes.Valfa_1 = aux1.Valpha;
                        Controller_probes.Valfa_2 = aux1.Vbeta;
                        /* Read Probe : 'Controller/Idq' */
                        Controller_probes.Idq_1 = aux1.Id;
                        Controller_probes.Idq_2 = aux1.Iq;
                        /* Read Probe : 'Controller/IdqRef' */
                        Controller_probes.IdqRef_1 = aux1.IdRef;
                        Controller_probes.IdqRef_2 = aux1.IqRef;


                        /* Read Probe : 'Controller/Duty' */
                        Controller_probes.Duty_1 = aux1.VaRef;
                        Controller_probes.Duty_2 = aux1.VbRef;
                        Controller_probes.Duty_3 = aux1.VcRef;


            /* Read Probe : 'Controller/enab' */
            Controller_probes.enab = aux1.enable;
            /* Digital Out : 'Controller/Digital Out' */
                 PLXHAL_DIO_set(0, Controller_probes.enab);

----------------------------/* FIN CODIGO ALUMNO*/--------------------------

      if (Controller_errorStatus)
      {
         return;
      }
      if (Controller_subTaskHit[2])
      {
         /* Update for Pulse Generator : 'Controller/Qref' */
         Controller_D_uint32_t[0] += 1;
         if (Controller_D_uint32_t[0] > 249)
         {
            Controller_D_uint32_t[0] = 0;
         }
      }
      if (Controller_subTaskHit[0])
      {
         /* Update for Pulse Generator : 'Controller/Pref' */
         Controller_D_uint32_t[1] += 1;
         if (Controller_D_uint32_t[1] > 999)
         {
            Controller_D_uint32_t[1] = 0;
         }
      }
      if (Controller_subTaskHit[1])
      {
         /* Update for Pulse Generator : 'Controller/enable' */
         Controller_D_uint32_t[2] += 1;
         if (Controller_D_uint32_t[2] > 499)
         {
```

```
            Controller_D_uint32_t[2] = 0;
         }
      }
      /* Update for PWM  : 'Controller/PWM' */
      PLXHAL_PWM_enableAllOutputs();
      /* Increment sub-task tick counters */
      {
         size_t i;
         for (i = 0; i < 3; ++i)
         {
            Controller_subTaskTick[i]++;
            if (Controller_subTaskTick[i] >= Controller_subTaskPeriod[i])
            {
               Controller_subTaskTick[i] = 0;
            }
         }
      }
      break;
   }

   case 1:
/* Task "Blink task" */
   {

      /* Delay : 'Controller/Delay2' */
      Controller_B.Delay2 = Controller_X.Delay2;

      /* Logical Operator : 'Controller/Logical\nOperator2' */
      Controller_B.LogicalOperator2 = !Controller_B.Delay2;
      /* Digital Out : 'Controller/LED' */
      PLXHAL_DIO_set(1, Controller_B.LogicalOperator2);
      if (Controller_errorStatus)
      {
         return;
      }

      /* Update for Delay : 'Controller/Delay2' */
      Controller_X.Delay2 = Controller_B.LogicalOperator2;
      break;
   }
   }
}

void Controller_terminate()
{
}
```

9 REFERENCIAS

[1] F. D. Trujillo, A. Pozo y A. Triviño, «Tema 1. Introducción a la Electrónica de Potencia,» Málaga, Univesidad de Málaga, 2011.

[2] S. Vázquez Pérez, J. I. León Galván y L. García Franquelo, «Tema 1. Introducción a la Electrónica de Potencia,» Sevilla, Departamento de Ingeniería Electrónica de la Universidad de Sevilla, 2020.

[3] A. M. Controls, «MOSFET / IGBT,» [En línea]. Available: https://www.a-m-c.com/es/experiencia/technologies/power-devices/mosfet-igbt/#:~:text=El%20IGBT%20es%20un%20transistor,de%20salida%20de%20los%20MOSFET..

[4] S. Vázquez Pérez, J. I. León Galván y L. García Franquelo, «Tema 4. MOS, IGBT y convertidores DC/DC sin aislamiento.,» Sevilla, Departamento de Electrónica de la Universidad de Sevilla, 2022.

[5] Distron, «Convertidor de potencia, ¿cómo funciona?,» 16 enero 2023. [En línea]. Available: https://distron.es/convertidor-de-potencia/.

[6] S. Vázquez Pérez, «Tema 2. Control de convertidores de potencia conectados a red eléctrica,» Sevilla, Departamento de Ingeniería Electrónica de la Universidad de Sevilla, 2022.

[7] M. J. S. Martín, «Investigando la Energía,» Cedec, [En línea]. Available: https://descargas.intef.es/cedec/proyectoedia/fisica_quimica/contenidos/investigando_energia/un_paseo_por_la_historia_de_la_energa.html.

[8] V. T. d. Gobierno, «Objetivos de reducción de emisiones de gases de efecto invernadero,» Ministerio para la transición ecológica y el reto demográfico, [En línea]. Available: https://www.miteco.gob.es/es/cambio-climatico/temas/mitigacion-politicas-y-medidas/objetivos.html#:~:text=2021%20%2D%202030&text=Los%20principales%20objetivos%20de%20dicho,en%20el%20consumo%20de%20energ%C3%ADa..

[9] V. C. d. Gobierno, «Estrategia de Descarbonización a LargoPlazo 2050,» [En línea]. Available: https://ec.europa.eu/clima/sites/lts/lts_es_es.pdf.

[10] H. Akagi, E. H. Watanabe y M. Aredes, «Instantaneous Power Theory and Applications to Power Conditioning,» *IEEE*, Capítulo 3.

[11] PLECS, «Implementation of a digital and analog PI controler,» de *Introduction to the C-Script Block*.

[12] PLECS, «Program the MCU from CCS,» de *TI C2000 Target Support User Manual*.

[13] PLECS, «Simple PIL Model,» de *Demo Model*.

[14] S. Vázquez Pérez y A. Márquez Alcaide, «Práctica 2. Implementación del control de corriente para un convertidor de potencia conectado a la red eléctrica,» de *Prácticas de Integración de Energías Renovables*, Sevilla, Departamento de Ingeniería Electrónica de la Universidad de Sevilla.

[15] PLECS, «How PILWorks,» de *PIL User Manual*, 2018.