IMPLEMENTACIÓN DE PLATAFORMA DE *HARDWARE IN THE LOOP (HIL)* APLICADO A ELECTRÓNICA DE POTENCIA.

DANIEL RIOS RESTREPO

MARIA ALEJANDRA MORALES HIGUITA

INGENIERÍA MECATRÓNICA

ELKIN EDILBERTO HENAO BRAVO

INSTITUTO TECNOLÓGICO METROPOLITANO

Medellín, febrero 22 del 2022

RESUMEN

En el presente trabajo se plantea el desarrollo de una plataforma de *"Hardware in the loop"* (HIL) aplicada a convertidores de potencia y desarrollada para ser implementada usando como hardware de destino las DSP TMS320F28335 del fabricante *Texas Instrument.* Con el objetivo de poder desarrollar una metodología que les permita a las demás personas poder realizar este tipo de emulación y de esta manera apoyar los diferentes procesos de aprendizaje para el estudio de la electrónica de potencia y los productos de investigación relacionados.

El proyecto se realizó haciendo uso de Matlab y de la herramienta *SimCoder* del *software* PSIM el cual permite a partir de modelos esquemáticos generar códigos automáticos que se pueden programar en el hardware anteriormente mencionado. Se partió de un diseño de un convertidor DC-DC elevador tipo *FlyBack*, al cual se le realizó una modelación matemática, se encuentran las ecuaciones diferenciales que describen su comportamiento y a partir de estas se realiza un diagrama de bloques discreto para generar el código de programación de la planta. Haciendo uso del software Matlab se linealizó el sistema usando el Jacobiano y luego se diseñó un controlador discreto que se programa de igual manera usando *SimCoder* en otra DSP.

Como resultado final se logra una interconexión entre un DSP con la programación del modelo del convertidor y otra con el controlador digital diseñado, la variable evaluada fue el voltaje promedio a la salida del convertidor obteniendo una respuesta muy acorde a lo que sería el sistema implementado de manera real.

Para la programación de ambas DSP se usa el *sofware Code Compose Studio* (CCS) de *Texas instrument*, desde el cual también se realizan las perturbaciones al sistema en tiempo real.

Palabras clave: HIL, CCS, *SimCoder,* DSP, convertidor *FlyBack*, linealización, ADC, DAC, controlador, ecuaciones diferenciales.

RECONOCIMIENTOS

Queremos agradecer a nuestros familiares y amigos que nos dieron todo el apoyo moral y estuvieron siempre pendientes al desarrollo de este trabajo.

Al docente y asesor Elkin Henao por creer en nosotros para la implementación de esta idea, la paciencia por el tiempo que llevo su implementación y por todos los recursos facilitados y las asesorías brindas que fueron de gran ayuda para la obtención del resultado final.

ACRÓNIMOS

HIL Hardware in the loop.

DSP Digital signal processor.

CCS Code composer studio.

PSIM Power Sim.

FPGA Field Programmable Gate Array.

DAC Digital Analogue Converter.

ADC Analogue digital Converter.

SIMCODER Generador de código.

PWM Pulse Width Modulation.

PI Proportional and integral controller.

Vg Voltage de entrada.

V Voltage de Salida.

D Duty Cycle

IL Corriente del inductor.

Ron Resistencia de encendido del transistor

RL Resistencia serie del inductor.

TABLA DE CONTENIDO

1. INTRODUCCIÓN

La electrónica de potencia posee un sin número de aplicaciones en la industria y en el área académico debido a la importancia de su estudio los docentes e investigadores incurren en altos costos para la elaboración de prototipos y diseños de convertidores de potencia que permiten mejorar cada día el desarrollo de la humanidad.

La implementación de plataformas de *"Hardware in the loop"* permiten reducir los tiempos y costos para la elaboración de estos prototipos permitiendo prevenir toda las posibles fallas y errores de diseño antes de llevar a cabo su construcción final.

Con el fin de mejorar los procesos de enseñanza tanto virtual como presencial y apoyar los procesos de investigación que se realizan en el laboratorio de electrónica y energías renovables de la Institución, se propone desarrollar una plataforma de (HIL) aplicado a convertidores de potencia, basada en un procesador digital de señales (DSP) específicamente la tarjeta TMS320F28335 de *Texas Instrument* y apoyado en software especializado como PSIM y Matlab.

Finalmente, lo que se busca con este trabajo investigativo es poder desarrollar una metodología que permita replicar la implementación de la plataforma a cualquier convertidor de potencia.

OBJETIVO GENERAL

Desarrollar una plataforma de "*hardware in the loop"* (HIL) aplicado a electrónica de potencia, basada en un procesador digital de señales (DSP) y software especializado, para el apoyo en prácticas de laboratorio mediadas por virtualidad y los procesos de investigación del laboratorio de electrónica y energías renovables.

OBJETIVOS ESPECÍFICOS.

- Obtener el modelo dinámico de un convertidor de potencia determinado y programarlo en el hardware DSP TMS320F28335 de *Texas instrument*.
- Diseñar un controlador para el convertidor que será programado en otra DSP.
- Acondicionar las señales de entrada y salida de la DSP para poder observar la dinámica de la planta en un osciloscopio y comunicarse con la tarjeta de control.
- Validar los resultados obtenidos comparándolos con un software de simulación especializado como Matlab y PSIM.
- Elaborar un procedimiento a manera de guía de laboratorio para que los estudiantes y docentes puedan replicar dicha plataforma.

ORGANIZACIÓN DEL TRABAJO.

El siguiente trabajo se divide en cuatro secciones, las cuales describen el desarrollo para la obtención de una plataforma de *"Hardware in the loop"* (HIL) para un convertidor de potencia DC - DC.

En la primera sección, se muestra una breve introducción a la electrónica de potencia, los convertidores DC-DC, los conceptos de *"Hardware in the loop"* (HIL), procesador digital de señal y un resumen del uso de *SimCoder* de PSIM para la generación automática de código, todo esto se utiliza en la se segunda sección donde se muestra la metodología desarrollada para la obtención de la plataforma de *"Hardware in the loop"* (HIL) haciendo uso de la generación automática de código y la programación por medio de *code composer studio* (CCS) para las DSP TMS320F28335 del fabricante *Texas Instrument* a partir del modelo matemático y discretización del convertidor *Flyback*. En la tercera sección se analizan y comparan los resultados obtenidos y por último se dan las conclusiones.

2. MARCO TEÓRICO

La electrónica de potencia estudia la conversión y control de la energía eléctrica de la manera más eficiente posible. Es decir, maximizando su rendimiento y minimizando la perdida de energía (Ortega, 2002). Todo esto basado principalmente en la conmutación de dispositivos semiconductores de potencia a través de microprocesadores.

En la electrónica de potencia se combinan la potencia, la electrónica y el control. El control tiene que ver con las características de estado estable y dinámicas del sistema en lazo cerrado. La potencia tiene que ver con el equipo estático y rotatorio para la generación, transmisión y distribución de la energía eléctrica, y por su parte la electrónica se ocupa de los dispositivos y circuitos de estado sólido requeridos en el procesamiento de señales para cumplir con los objetivos de control deseados (Rashid, 2004). Los dispositivos que combinan estas tres características se denominan convertidores de potencia y se pueden encontrar diferentes clases según el tipo de energía que conviertan. A continuación, se muestran las clases de convertidores más utilizados:

Convertidor AC-DC o rectificador: Convierte una tensión de alterna en una tensión continua, los rectificadores son usados para alimentar todo tipo de sistemas electrónicos donde se necesite energía eléctrica en forma de corriente continua. Una de sus mayores aplicaciones es para el control de motores de corriente continua y en cargadores de baterías.

Convertidor AC-AC o cicloconversores: Es un convertidor que proporciona una corriente alterna a partir de otra corriente alterna, estos son usados principalmente para el control de velocidad de motores AC más conocidos como variadores de velocidad o variadores de frecuencia.

Convertidor DC-DC o troceadores: Convierte una tensión continua en otra determinada tensión continua con bajo rizado, existen convertidores DC-DC elevadores, reductores y elevador reductor de tensión. Estos convertidores tienen una gran cantidad de aplicaciones

entre las que se destaca el control de motores DC, la alimentación de dispositivos electrónicos a partir de baterías o fuentes autónomas de corriente continua.

Convertidor DC-AC o inversor: Suministra una tensión alterna a partir de una tensión continua que por lo general la entrega un convertidor DC-DC. Una de las aplicaciones mas comunes es la UPS, también es muy utilizado en aplicaciones de fuentes de energía renovable como la solar y la eólica.

El principio de funcionamiento será el mismo para todos los convertidores conmutados el cual se basa en el almacenamiento y transferencia de energía en ciclos de conmutación. Con el objetico de realizar esta transferencia de la manera más eficiente posible los convertidores de potencia ideales solo contienen elementos que no presentan grandes pérdidas, como lo son elementos reactivos (Inductores, capacitores) y elementos de conmutación rápida (IGBT, Mosfets, Diodos). También, su eficiencia dependerá de un adecuado diseño electrónico y una elección adecuada del controlador (Ortega, 2002).

El producto de investigación se basó principalmente en un convertidor DC-DC por lo que se profundiza en este tipo de convertidores.

CONVERTIDORES DC-DC

Los convertidores DC-DC son circuitos de potencia conmutados, donde el voltaje de entrada DC es convertido a un voltaje de salida DC teniendo mayor o menor magnitud dependiendo de su configuración y posiblemente con una polaridad invertida o con aislamiento de la referencia de tierra entre la entrada y la salida. Para la implementación de estos convertidores se hace uso de dispositivos cuyas pérdidas de energía por disipación calórica sean mínimas, como lo son: inductores, capacitores, transistores en modo conmutado, diodos, entre otros (Erickson, 2004), para de esta manera poder lograr un convertidor con alta eficiencia, si un convertidor tiene baja eficiencia la entrega de la potencia requerida no sería útil para la aplicación que se desee.

Dentro de los convertidores DC-DC se pueden encontrar diferentes configuraciones dependiendo la disposición de los elementos reactivos y conmutados dentro del circuito. Las tres configuraciones básicas son: *Convertidor Back* (reductor), *Convertidor Boost* (elevador) y *Convertidor Back-Boost* (Elevador – Reductor). Para el objetivo de este proyecto se estudia el convertidor elevador *FlyBack* con aislamiento galvánico de topología similar a al *Back-Boost.*

CONVERTIDOR FLYBACK.

El convertidor *FlyBack* está basado en el convertidor *Buck- Boost* con aislamiento galvánico entre la entrada y la salida que posee un transformador en lugar de un inductor, debido a esto se pueden alcanzar varias ratios de conversión. Comúnmente es usado para aplicaciones de baja potencia y por contar con una alta eficiencia.

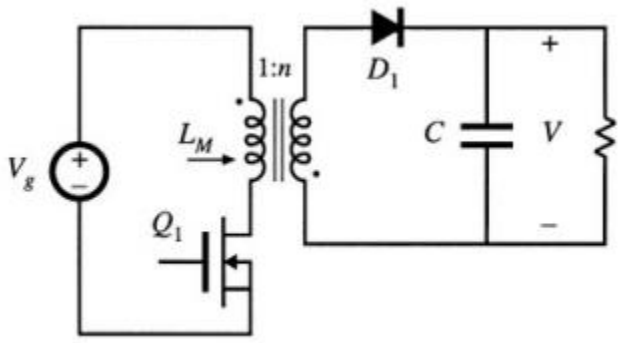


Figura 1. Circuito del convertidor *FlyBack.*

Fuente: *(Erickson, 2004, pág. 162)*

En el transformador del convertidor *Flyback* a diferencia del transformador ideal, la corriente no fluye simultáneamente en ambos devanados. La Figura 1 ilustra la configuración habitual del convertidor *Flyback*. El MOSFET está conectado a la toma de tierra del lado primario del transformador, lo que simplifica el circuito de accionamiento. La polaridad del transformador se invierte para obtener un voltaje de salida positivo. Se

introduce una relación de vueltas $(1:n)$; lo que permite una mejor optimización del convertidor. La inductancia de magnetización (L_M) funciona igual inductor (L) en un convertidor Buck-Boost. Cuando el transistor (Q_1) conduce, la energía de la fuente (V_g) se almacena en la inductancia de magnetización (L_M). Cuando el diodo (D_1) conduce, la energia almacenada es transferida a la carga con el voltaje y la corriente del inductor escalado acorde a la relación de transformación del transformador $(1:n)$. (Erickson, 2004, pág. 162).

Es utilizado en aplicaciones como fuentes de alimentación conmutadas, cargadores de baterías, sistemas de ignición en motores de combustión interna y aplicaciones con celdas fotovoltaicas.

HARDWARE IN THE LOOP (HIL).

Las pruebas de *"Hardware in the loop"* (HIL) son simulaciones en tiempo real que permiten validar modelos de sistemas y controladores sin necesidad de una implementación física con el fin de probar condiciones anormales y de fallo que pueden dañar el hardware si el código en desarrollo no opera dentro de las especificaciones. La simulación HIL muestra como el controlador responde, en tiempo real, a estímulos virtuales realistas. Se puede implementar también para determinar si el modelo físico del sistema (planta) es correcto. Esto se logra usando una plataforma que genera unas entradas y salidas analógicas y digitales que emulan las entradas y salidas del sistema real, de manera que el controlador se ejecuta como si actuara sobre el sistema real.

Un sistema HIL está constituido por tres componentes principales, el hardware del controlador, un hardware emulador para la planta y un ordenador que contine la interfaz de usuario, la Figura 2 muestra una configuración típica de una plataforma de *"Hardware in the loop"* (HIL).

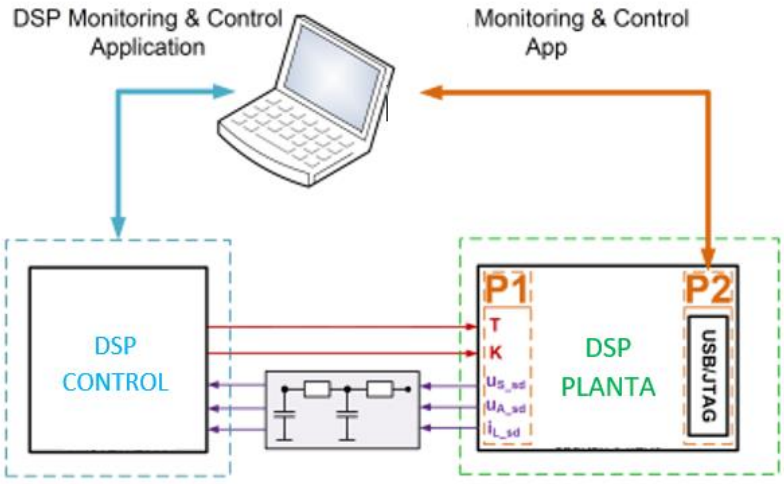


Figura 2. Configuración típica de Hardware in the loop (HIL).

Fuente: Adaptación propia tomada de *(Z. Süto, 2014)*

La simulación HIL se ha convertido en un tipo de desarrollo muy eficiente e importante porque permite visualizar la respuesta de una planta a través de un dispositivo de hardware, sin necesidad de realizar las conexiones físicas del sistema, (Trujillo, 2017). Además, permite testear el algoritmo de control diseñado sin tener en cuenta las restricciones de seguridad, coste o disponibilidad que tendría probar el control sobre el producto construido. De esta manera, probando exhaustivamente el controlador en un entorno virtual antes de proceder a realizar pruebas con el dispositivo físico, se puede asegurar la fiabilidad del sistema de control sin aumentar el "*time to-market*" o los costes asociados a las pruebas sobre el sistema real (Martin., 2021).

En la simulación HIL aplicada a convertidores de potencia, para garantizar el comportamiento en tiempo real cuando se simula el comportamiento de conmutación, es necesario ejecutar la frecuencia de muestreo de simulación 100 veces más rápido que la velocidad de conmutación esperada del sistema real, es necesario que el sistema de hardware tenga una latencia de E/S baja y un procesador que pueda alcanzar la frecuencia de funcionamiento. (Mathworks, 2021).

Cuando se realizan pruebas *"Hardware in the loop"* para sistemas de electrónica de potencia, no existe una regla absoluta sobre si una CPU o un FPGA es mejor para simular

sistemas con dinámica de conmutación. A la hora de decidir, hay que tener en cuenta la complejidad del modelo, los detalles matemáticos de los componentes de electrónica de potencia, los detalles de modelado de las cargas y las fuentes de alimentación, y el número de canales y tipos de E/S de los sistemas de prueba (Mathworks, 2021).

PROCESADOR DIGITAL DE SEÑALES (DSP).

Un sistema de procesamiento digital de señal aplica operaciones matemáticas a señales física representadas de forma digital mediante secuencias de muestras. Para realizar el procesado, las señales físicas se capturan mediante conversores análogos digitales.

Los DSP son microprocesadores diseñados con arquitecturas especiales para acelerar los cálculos matemáticos necesarios en sistemas embebidos de tiempo real, lo que hace de este dispositivo un hardware ideal para la implementación de plataformas de "*hardware in the loop".*

El microprocesador que contiene la DSP cuenta con dos unidades básicas para manipular las señales de entrada y salida, a estas unidades se le conoce como unidad de control y unidad de procesamiento central o CPU. Las actividades que realiza el microprocesador son cíclicas y se encuentran almacenadas en una memoria de código que se cargan cada vez que se cumple un ciclo de trabajo completo. La siguiente figura muestra las partes principales de un DSP. (Mendez, 2013)

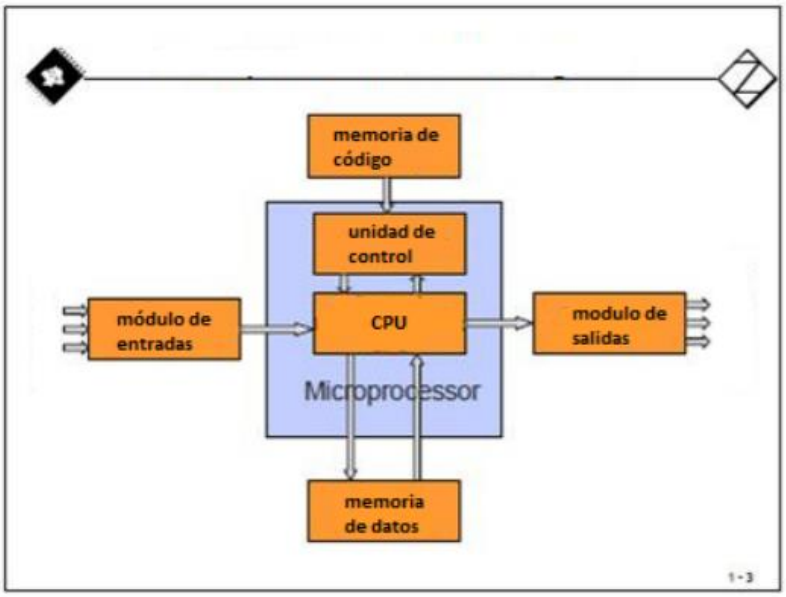


Figura 3. Partes fundamentales de un DSP.

Fuente: (Mendez, 2013, pág. 17)

DSP TMS320F28335 de *Texas Instrument*.

La DSP TMS320F28335 de *Texas Instruments* utilizada para la aplicación de este trabajo, hace parte de la gama de procesadores de 32 bits C28x Delfino de la familia C2000 creados para tener un alto rendimiento en aplicaciones de control en tiempo real. Este dispositivo cuenta con una arquitectura "*Harvad*" y aritmética de coma flotante (Mendez, 2013). A continuación, se muestran sus principales características:

- Velocidad de trabajo de 150MHz.
- Reloj de entrada de 30 MHz
- 68K bytes de memoria RAM.
- 512K bytes de memoria flash.
- CPU de 32 bits.
- 6 canales de acceso a memoria directo DMA.
- 6 módulos de modulación de ancho de pulso PWM con 12 salidas.
- 6 módulos de modulación de ancho de pulso de alta resolución HRPWM.
- 6 entradas de modo de captura eCAP.

- 2 módulos para entrada de Encoder eQEP.
- Módulos de temporizador de perro guardián.
- 16 canales de conversor análogo digital ADC de 12 bits y frecuencia de muestreo de hasta 80Khz y periodo de conversión de 80ns.
- Módulos de comunicación serial SPI.
- 3 módulos de comunicación serial SCI.
- 2 módulos de comunicación CAN.
- 1 módulo de comunicación I2C.
- 88 pines de entrada y salida de propósito general con voltaje máximo de 3.3V.
- Controlador USB JTAG.

Es importante aclarar que el DSP TMS320F28335 va montado sobre una tarjeta "*interface*" llamada "*Experimenter Kit",* esta le proporciona energía y permite la comunicación vía USB JTAG a la DSP, además contiene pines de conexión para acceder fácilmente a los módulos de entrada y salidas de propósito general (GPIO), modulo análogo digital (ADC), puertos de comunicación entre otros.

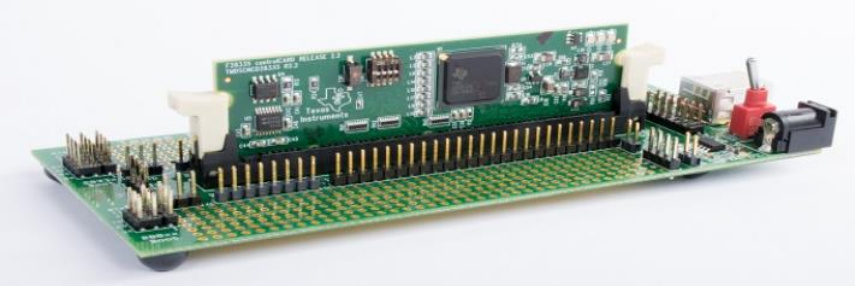


Figura 4. DSP TMS320F28335 acoplado a tarjeta interface Experimenter Kit.

Fuente: https://www.ti.com/tool/TMDSDOCK28335

Software de programación.

El software utilizado para programar las DSP de *Texas Instruments* se llama *Code Composer Studio (CCS).* Este es un entorno de desarrollo que dispone de compiladores para cada dispositivo, editor de código fuente, entorno de creación de proyectos, depurador, simuladores, sistemas de operación en tiempo real entre otras características.

El presente proyecto se desarrolló en la versión 9.3 de *CCS* de forma indirecta desde el módulo para generación de código del software PSIM de *Powersim* llamado *SimCoder,* el cual es compatible con la familia de DSP F283xx para la generación automática de códigos.

GENERACIÓN AUTOMÁTICA DE CÓDIGO CON SIMCODER DE PSIM.

SimCoder es una expansión del sofware para electrónica de potencia PSIM, el cual permite la generación automática de códigos en lenguaje C a partir de esquemáticos realizados en dicho software, estos códigos pueden ser programados directamente en un grupo de DSP´s y FPGA´s específicos de *Texas instruments.* Las familias de tarjetas compatibles con *SimCoder* son:

Tarjeta	Version de CPU que soportan *SimCoder*
F2833x	28335, 28334, 28332.
F2803x	28035, 28034, 28033, 28032, 28031, 28030.
F2806x	28069, 28068, 28067, 28066, 28065, 28064, 28063, 28062.
F2802x	28027, 28026, 28023, 28022, 28021, 28020, y 280200.
F2837x	28374S/D, 28375S/D, 28376S/D, 28377S/D, 28379S/D.
F28004x	280049, 280049C, 280048, 280048C, 280045, 280041, 280041C, 280040, 280040C.
PE-Expert4	Para el *hardware* DSP PE-Expert4. PE-Expert4 es una plataforma de desarrollo de DSP producido por Myway Co. Utiliza el DSP TMS320C6657 de punto flotante de TI y el Biblioteca PE-OS.

Tabla 1. Familia de tarjetas compatibles con SimCoder.

Fuente: *(Powersim Inc., 2020)*

El *SimCoder* se habilita desde el control de simulación en PSIM (*Simulation Control*) en la segunda pestaña como se muestra en la Figura 5:

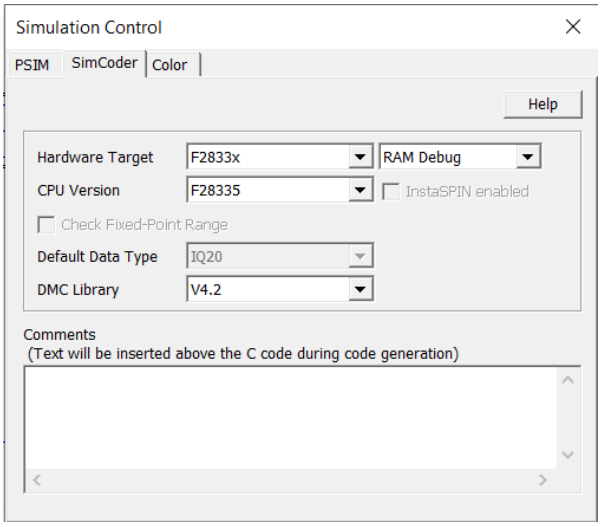


Figura 5.Ventana de configuración de SimCoder.

Fuente: ***(Powersim Inc., 2020)***

Se deben seleccionar y configurar estos parámetros correctamente para que *SimCoder* genere códigos correctamente. A continuación, se explica cada uno de los parámetros configurables:

- ***Hardware target:*** Selecciona la familia de la tarjeta a la que se le va a generar el código y posteriormente programar.
- ***CPU versión:*** Selecciona la CPU especifica que será programada.
- ***Project Configuration:*** Se pueden seleccionar cuatro configuraciones posibles para las diferentes tarjetas de destino; *RAM Debug*, *RAM Release, Flash Release, or Flash RAM Release, para la FPGA PE-Expert4* solo permite *PE-ViewX*.
- ***DMC Library:*** Selecciona la versión de macros de *Texas instruments* para generar códigos para control de motores DCM.

Elementos para la generación de códigos:

Todos los elementos para la generación de código que se encuentran en "*Elements – Event Control y Elements" – SimCoder*, son compatibles con la generación de código, además un gran número de elementos de la librería de PSIM estándar se pueden utilizar, para

diferenciar los elementos estándar que se pueden usar entre los que no se pueden, se selecciona "*options – setting – advanced*" y se chulea el recuadro "*Show image next to elements that can be used for code generation*" ver Figura 6. Todos los elementos que aparecen con el icono "CG", C en rojo G en azul, son elementos compatibles con *SimCoder*, también todos los elementos que aparecen con el icono "TI", T en rojo I en azul, son elementos de hardware de las tarjetas DSP compatibles descritas anteriormente en la Tabla 1.

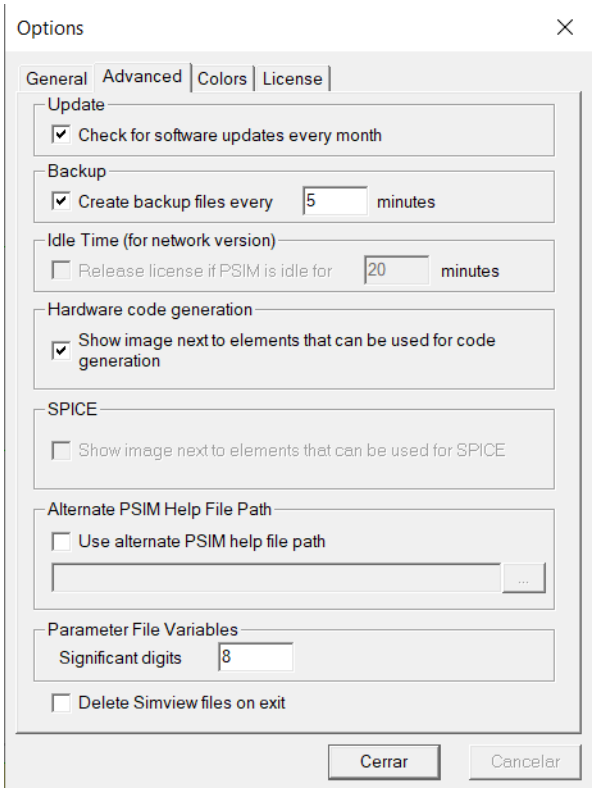


Figura 6. Opciones avanzadas de PSIM.

Fuente: ***(Powersim Inc., 2020)***

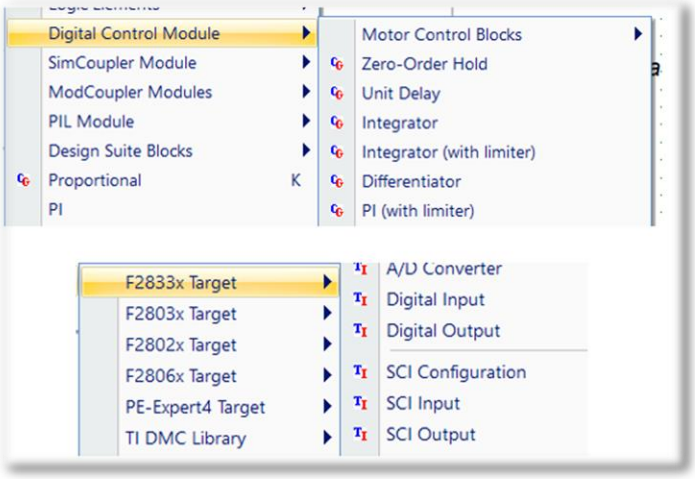


Figura 7. Elementos compatibles con generación de código.

Fuente: ***(Powersim Inc., 2020)***

Paso a paso para la generación de código con *SimCoder*.

En general, la generación automática de código mediante *SimCoder* implica los siguientes pasos:

1. Diseñar y simular un sistema en PSIM con el control en dominio continuo.
2. Convierta la sección de control del sistema en un dominio discreto y simule el sistema.
3. Si no hay un objetivo de hardware, coloque la sección de control en un subcircuito y genere el código.
4. Si hay un objetivo de hardware, modifique el sistema incluyendo elementos de hardware y ejecute la simulación para validar los resultados, luego genere el código.

Tenga en cuenta que el código solo se puede generar cuando el control está en un dominio discreto, no en un dominio continuo.

Sistema en dominio continuo:

A menudo, un sistema se diseña y simula primero en un dominio continuo. A continuación, se muestra un circuito convertidor de DC simple, el controlador PI (proporcional-integral) en el circuito de control está diseñado en el dominio continuo. Las ganancias de PI, k y la constante de tiempo T son: k = 0,4 y T = 0,0004. La frecuencia de conmutación es 20 kHz.

El objetivo de este ejercicio es generar el código C para el circuito de control en el cuadro punteado. Para realizar la generación de código, el primer paso es convertir el controlador PI analógico en el dominio s al controlador PI digital en dominio z discreto.

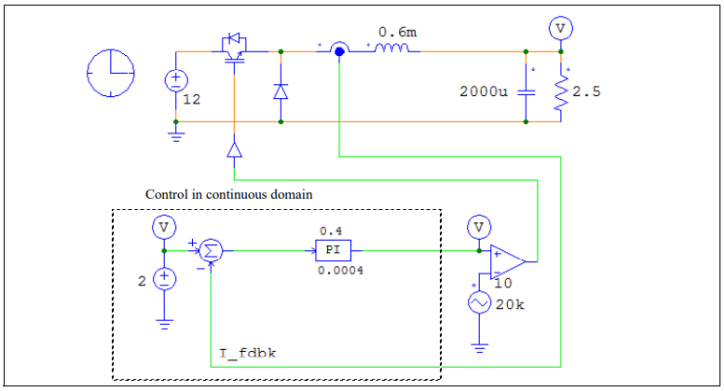


Figura 8. Sistema de convertidor CC en modo de control continuo.

Fuente: ***(Powersim Inc, 2020)***

Sistema en dominio discreto.

Para convertir un controlador analógico en un controlador digital, se puede utilizar el programa *Convertidor s2z* que viene con el módulo de control digital de PSIM. Para iniciar esta herramienta, seleccione "*Utilities >> s2z Converter",* o si prefiere diseñe su controlador o sistema en modo discreto y posteriormente lo realiza en PSIM.

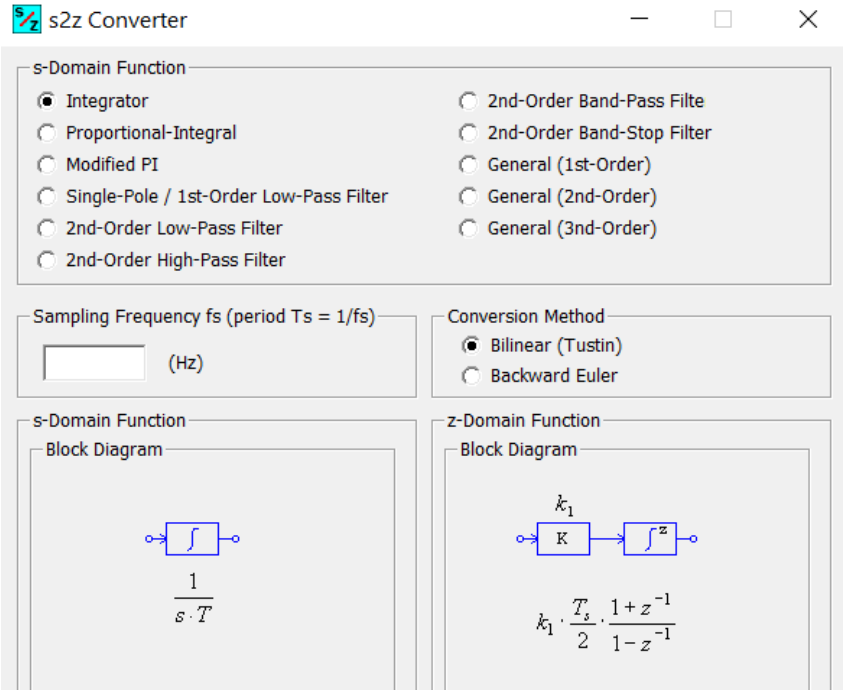


Figura 9. Convertidor s2z de sistema continuo a discreto.

Fuente: *(Powersim Inc, 2020)*

Se pueden utilizar diferentes métodos de conversión para convertir un controlador analógico en un controlador digital, los métodos comúnmente utilizados son el método Bilineal (también llamado Tustin o Trapezoidal) y el método "*Backward Euler*".

Para el circuito de la Figura 10, se convierte el controlador PI analógico al controlador PI digital y se obtienen los parámetros del controlador PI digital como: k1 = 0.4 para la acción proporcional y k2 = 1000 para la acción integral. Es importante asignar una frecuencia de muestreo por lo menos del doble de la frecuencia de conmutación del convertidor. El circuito con el controlador digital se puede observar a continuación:

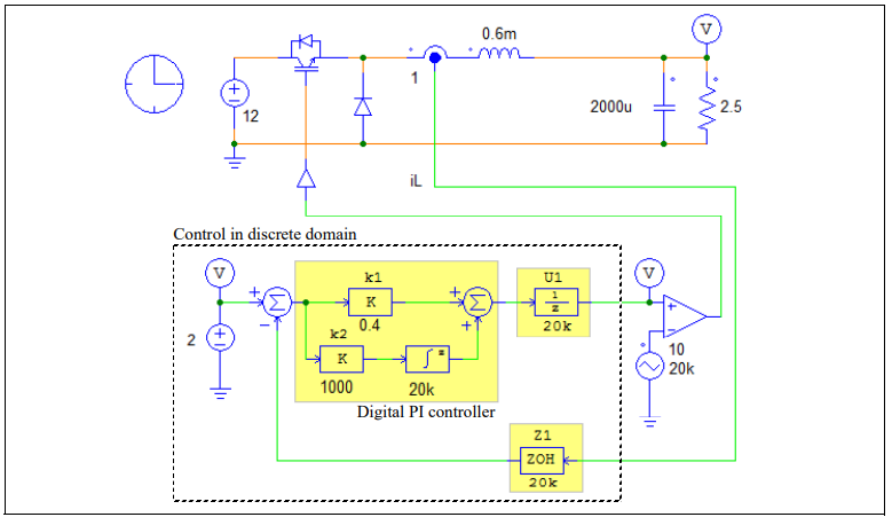


Figura 10. Sistema de convertidor CC en modo de control discreto.

Fuente: ***(Powersim Inc, 2020)***

En comparación con el circuito de control en dominio continuo, hay tres cambios en este circuito, como se destaca por las cajas amarillas. Primero, el controlador PI analógico se reemplaza por el controlador PI digital. La "bandera de algoritmo" del integrador digital se establece en 1 (para el método de "*Backward Euler*"), y la frecuencia de muestreo se establece en 20 kHz. Las ganancias k1 y k2 se obtienen del programa de conversión descrito anteriormente.

Además, se utiliza un bloque de retención de orden cero (Z1) para simular el convertidor analógico digital utilizado para el muestreo de la corriente de retroalimentación IL y se utiliza una unidad de retardo U1 para modelar ciclo por ciclo la acción de control entregada por el controlador debido a que, por lo general, se toman muestras de las cantidades al comienzo de un ciclo, y los parámetros del controlador se calculan dentro del ciclo. Pero como lleva tiempo realizar el cálculo, las cantidades recién calculadas normalmente no se utilizan hasta el comienzo del siguiente ciclo.

Tenga en cuenta que el controlador digital convertido debería dar como resultado un bucle de control estable y el rendimiento deseado. Si los resultados de la simulación con el control

digital no son estables o no son los deseados, es necesario volver al sistema de control analógico y rediseñar el controlador y repita el proceso con el método "*Backward Euler".*

Elementos de hardware para generación de código.

Luego de tener el controlador o modelo en sistema discreto, se pueden agregar elementos de hardware de *SimCoder* dentro del circuito previamente elaborado, estos elementos de hardware deben seleccionarse de acuerdo con la tarjeta que se va a programar, se debe tener presente los rangos de voltaje a los que trabajan las tarjetas y realizar una debida escalización de los valores.

A continuación, se muestra el mismo circuito de ejemplo, pero agregando elementos de hardware de destino F2833x. Para una mejor ilustración los elementos de hardware están resaltados en amarillo.

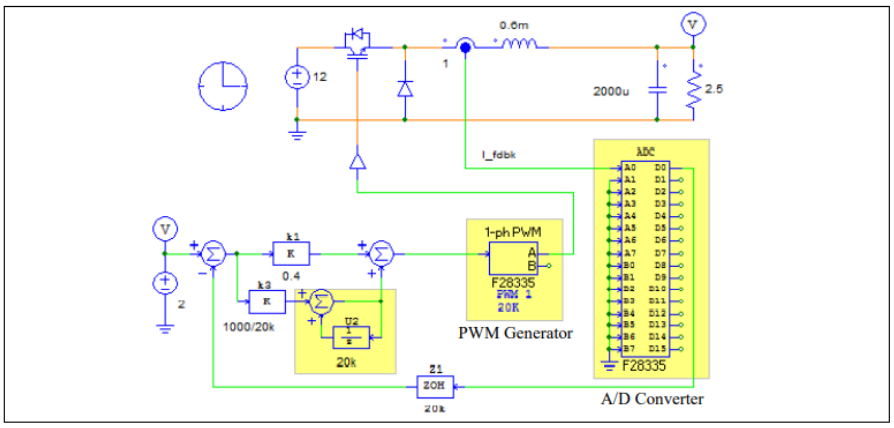


Figura 11. Convertidor CC con elementos de hardware F2833X.

Fuente: ***(Powersim Inc, 2020)***

En la Figura 11, se pueden observar dos elementos de hardware para una DSP F28335, el elemento ADC, es el conversor análogo digital con el que cuenta la DSP y para este ejemplo se utiliza el canal 0 para leer la señal de corriente I_L del convertidor CC, este conversor tiene 8 canales disponibles. Por otro lado, para la salida del controlador digital que contendrá

internamente el programa de la DSP se tiene una salida PWM de un solo canal (1-ph-PWM), este módulo PWM es configurable.

Para ampliar más la información sobre la configuración y demás elementos de *SimCoder* ver el Capítulo 6 del manual de *SimCoder* y el tutorial "*Auto Code Generation for* F2833X Target", ambos disponibles en la página web de PSIM.

Generación de código C con *SimCoder*.

Luego de tener el circuito simulado y verificado, se procede con la generación del código C seleccionando, *simulate – generate code*. El código generado para el hardware F2833x estará listo para ejecutarse sin cambios.

Después de generar el código se debe proceder con la programación de la DSP utilizando el *sofware Code Composer Studio (CCS)* de *Texas instrument*.

Durante esta sección se estudiaron los conceptos necesarios para la implementación de una plataforma de *Hardware in the loop.* Desde entender el funcionamiento del sistema a emular como lo son los convertidores de potencia, los elementos de hardware necesarios para el desarrollo del trabajo como la DSP TMS320F28335, hasta el uso de herramientas de *software* como *SimCoder* de PSIM la cual permite hacer una implementación más sencilla por medio de la generación automática de código a partir de un sistema discretizado. Todos estos elementos se utilizarán y estudiarán con mayor profundidad en el siguiente capitulo.

3. METODOLOGÍA

Para realizar la implementación de una plataforma de *"Hardware in the loop"* aplicada a la electrónica de potencia se utiliza como software principal PSIM, específicamente el módulo de *SimCoder* para la generación automática de código a partir de un diagrama con elementos compatibles para poder programar y simular el convertidor de potencia en un hardware DSP TMS320F28335 de *Texas Instrumensts*, para esto es necesario modelar y discretizar el convertidor a partir de sus ecuaciones diferenciales. También, se diseña un controlador para el convertidor y utilizando la generación automática de código se programa en otra DSP para lograr hacer la interacción entre planta y controlador. Para el cumplimiento de este objetivo se realiza una metodología de paso a paso que va desde el diseño del convertidor hasta la implementación final que consiste en la interconexión entre un DSP que contendrá la planta y otra con el controlador. Se utiliza como documentación de apoyo toda la teoría expuesta en el marco teórico, el manual de *SimCoder* y los diferentes tutoriales que ofrece PSIM para el uso adecuado de este módulo.

HARDWARE Y SOFTWARE.

El hardware empleado para la implementación de la plataforma, tanto para el modelo dinámico del convertidor (planta) como para el controlador es la DSP TMS320F28335 de *Texas Instruments,* el cual como se estudió previamente una de sus principales ventajas es la alta velocidad de procesamiento de datos que lo hace ideal para aplicaciones de control en tiempo real. También, su modulo análogo digital de 12 bits y 2 canales que permite conectar diferentes señales analógicas, las salidas PWM mejoradas para la implementación de control y su gran cantidad de puertos digitales de entrada y salida (I/0). Como desventaja al utilizar este hardware se encuentra que no tiene puerto de salidas analógicas, por lo que es necesario utilizar un conversor digital análogo para poder obtener las respuestas de las variables entregadas por la planta.

El convertidor digital análogo recomendado debe superar velocidades de 25Mhz, tener una resolución de 12 o 16 bits y debe permitir comunicación *SCI.* Debido a que en el país este tipo de dispositivos son de difícil consecución se utiliza como alternativa de solución un convertidor digital análogo PWM, el cual consiste en tomar una salida PWM y pasarla por un filtro pasa bajos RC y de esta manera obtener la señal analógica.

Un filtro pasa bajos pasivo de primer orden RC solo permite el paso de frecuencias bajas y atenúa las frecuencias altas. Este compuesto por dos elementos, una resistencia y un condensador conectados en serie. La entrada es por la resistencia y la salida por el condensador. Se conoce como pasivo porque solo está compuesto por elementos pasivos, y es de primer orden porque solo contiene un elemento reactivo (un condensador).

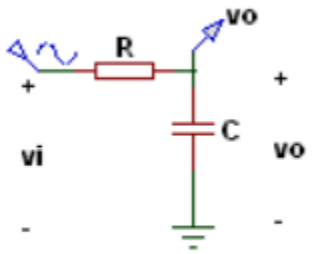


Figura 12. Filtro Pasa bajos Pasivo de 1er Orden RC.

Fuente *(Wilaeba electronica, 2018)*

La programación del modelo dinámico del convertidor y el controlador se realiza desde el módulo *SimCoder* de PSIM para luego generar automáticamente el código C del sistema y compilar y programar este en la DSP con el software *Code Composer Studio* (CCS) de *Texas Instrument,* desde este también se observa y modifican las diferentes variables que para evaluar el comportamiento del sistema en tiempo real.

MODELADELADO MATEMATICO DEL CONVERTIDOR FLYBACK.

A la hora de empezar a modelar un convertidor se tienen en cuenta los dos estados que puede tener el dispositivo de conmutación, el cual puede ser abierto (S=0) o cerrado (S=1). Se empieza obteniendo las ecuaciones del circuito generado dependiendo de qué estado

de conmutación se halle el interruptor, el tiempo de encendido (*DTs*) y el tiempo de apagado (*D'Ts*). La finalidad del análisis de las dos etapas siempre será obtener las ecuaciones para el voltaje del inductor (V_l) y la corriente del capacitor (I_C), para encontrar el modelo matemático final representando el voltaje de salida, (V_c) y corriente de salida (I_l) (Trujillo, 2017). Algo muy importante que interfiere en el valor de la salida, es el *Duty cicle* (ciclo de trabajo), el cuales es la relación existente entre el tiempo en que una señal se encuentra en estado activo con el periodo de dicha señal. En un convertidor DC-DC, el ciclo de trabajo (*Dts*), puede tomar valores entre cero y uno, por tanto, *1-D o D'*, será el complemento del ciclo de trabajo.

A continuación, se realiza el análisis para el convertidor *Flyback.*

Para Q1 en estado *ON:* Durante el primer ciclo de trabajo (*Dts*), mientras el transistor Q_1 conduce, y el diodo D_1 esta apagado, el circuito resultante del convertidor se puede apreciar en la siguiente figura:

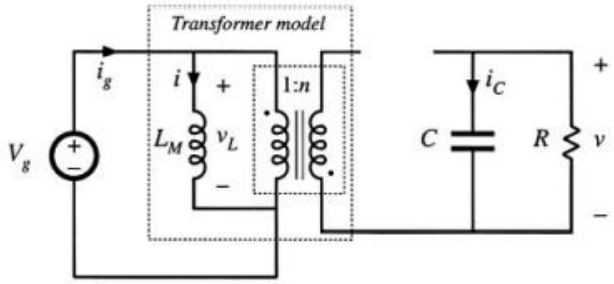


Figura 13. Convertidor Flyback con Q1 ON.

Fuente: ***(Erickson, 2004, pág. 162)***

Para el circuito de la Figura 13, el voltaje del inductor (V_l), la corriente del capacitor (I_C), y la corriente de la fuente (I_g) están dadas por:

$$V_l = V_g \qquad (1)$$

$$I_C = -\frac{v}{R} \qquad (2)$$

$$I_g = I_l \qquad (3)$$

Para Q1 en estado *OFF:* Durante el segundo ciclo de trabajo (D'ts), el transistor Q_1 esta apagado y el diodo D_1 conduce obteniendo el circuito equivalente a continuación:

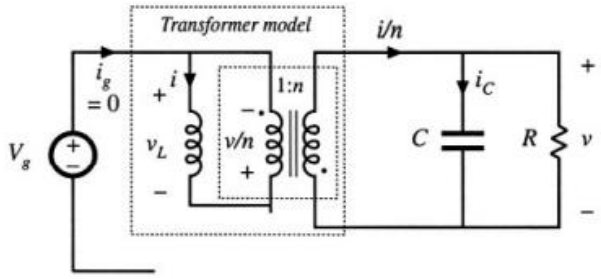


Figura 14. Convertidor Flyback con Q1 OFF.

Fuente: *(Erickson, 2004, pág. 162)*.

El voltaje del inductor (V_l), la corriente del capacitor (I_C), y la corriente de la fuente (I_g) obtenidas para este subcircuito de la Figura 14 es:

$$V_l = -\frac{v}{n} \qquad (4)$$

$$I_C = \frac{I_l}{n} - \frac{v}{R} \qquad (5)$$

$$I_g = 0 \qquad (6)$$

Luego de obtener las ecuaciones de V_l, I_C, e I_g, se grafican las diferentes señales en un periodo de conmutación (t_s) para los ciclos de trabajo de encendido *DTs* y de apagado *D'Ts*, con el fin de obtener el balance de voltio segundo en el inductor y el balance de cargas en el capacitor y con estos poder encontrar las ecuaciones para la obtención los valor promedios de voltaje y corriente de salida del convertidor, así como su relación de transformación en función del voltaje de entrada.

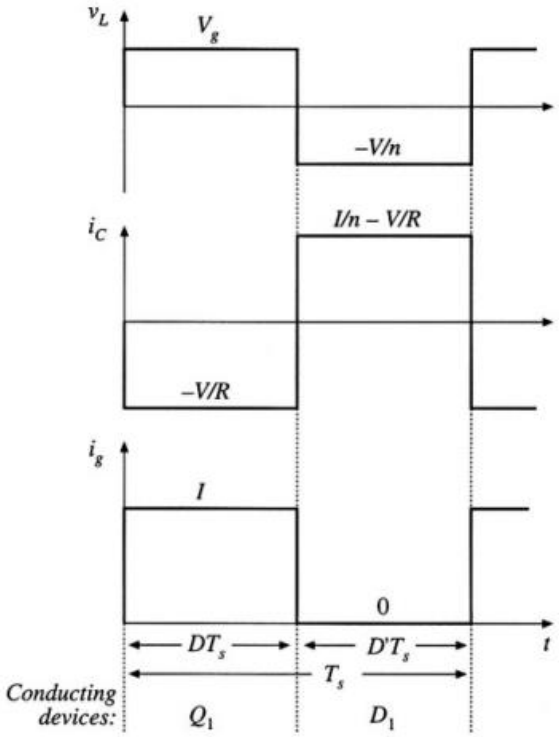


Figura 15. Grafica del convertidor FlyBack en modo de conducción continua.

Fuente: *(Erickson, 2004, pág. 164)*.

Balance de voltio segundo en el inductor.

Según el grafico que se muestra en la Figura 15, se procede a obtener el balance de voltio segundo en el inductor y así encontrar la ecuación de voltaje de salida promedio, para esto se analiza el grafico de V_l con respecto al periodo de conmutación:

$$\langle VL \rangle = D(V_g) + D'\left(-\frac{V}{n}\right) = 0 \qquad (7)$$

Teniendo en cuenta la aproximación de pequeña señal, un capacitor cargado no pide corriente y en un inductor cargado su voltaje es igual a cero, por lo que el voltaje de salida promedio en el capacitor seria:

$$V = \frac{D \cdot V_g \cdot n}{D'} \qquad (8)$$

De la ecuación (8), se obtiene la relación de transformación para el convertidor *Flyback* y la ecuación para determinar el ciclo de trabajo *(Duty).*

$$M(D) = \frac{V}{V_g} = n\frac{D}{D'} \qquad (9)$$

$$D = \frac{V}{V + V_g \cdot n} \qquad (10)$$

Balance de carga en el capacitor.

Con el fin de obtener la corriente de salida o corriente del inductor (I_l), se realiza el análisis de la segunda grafica de la Figura 15, que corresponde a la corriente del capacitor (I_C) y se obtiene:

$$\langle Ic \rangle = D\left(-\frac{V}{R}\right) + D'\left(\frac{I_l}{n} - \frac{V}{R}\right) = 0 \qquad (11)$$

De la ecuación (11), se obtiene que la corriente del inductor es:

$$I_l = \frac{D \cdot V_g \cdot n^2}{R \cdot D^2} \qquad (12)$$

Ecuación para el promedio de la corriente de entrada I_g:

$$Ig = I_l(D) \qquad (13)$$

Rizados de corriente del inductor (Δ_{il}).

Teniendo en cuenta que la corriente es triangular alrededor de un valor promedio DC con un rizado, se obtiene la gráfica del rizado de corriente del inductor a partir de la gráfica de V_l en la Figura 15, con esto podemos se podrá obtener el valor del inductor para obtener el voltaje de salida deseado.

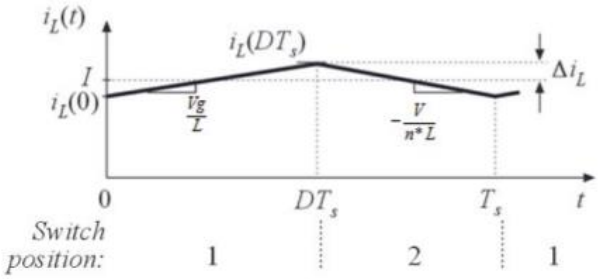


Figura 16. Corriente del inductor para convertidor Flyback.

Fuente: ***(Garcia., 2018, pág. 38)***

A partir de la gráfica de la Figura 16 y aplicando la ecuación de la recta en cualquiera de los dos estados de conmutación, se puede obtener la ecuación del inductor para el convertidor *Flyback.*

$$\frac{2\Delta i_L}{DT_S} = \frac{V_g}{L} \quad (14)$$

Despejando de la ecuación (14), se obtiene:

$$L = \frac{V_g \cdot D \cdot T_S}{2 \cdot \Delta i_L} \quad (15)$$

Rizados de voltaje del capacitor (Δ_{Vc}).

Para los cálculos del condensador se ha de tener claro que el voltaje promedio a la salida V_{out} es igual al voltaje del condensador V_C, y se analiza el rizado de voltaje del condensador para cada uno de los intervalos de funcionamiento del circuito partiendo de la gráfica de I_C de la Figura 15.

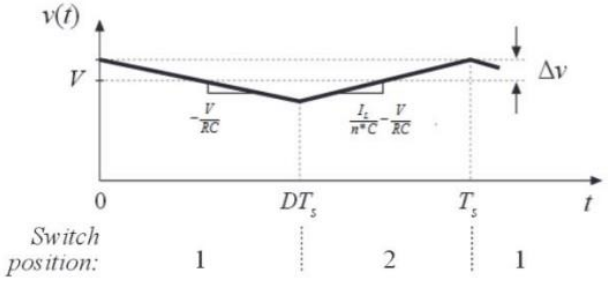


Figura 17. Rizado de voltaje en el capacitor para el convertidor Flyback.

Fuente: ***(Garcia., 2018, pág. 40)***

A partir de la gráfica de la Figura 17 y aplicando la ecuación de la recta en cualquiera de los dos estados de conmutación, se puede obtener la ecuación del capacitor para el convertidor *Flyback.*

$$\frac{2\Delta v}{DT_S} = \frac{V}{R \cdot C} \qquad (16)$$

Despejando de la ecuación (16) en términos del voltaje de entrada se obtiene:

$$C = \frac{D^2 \cdot V_g \cdot n \cdot Ts}{2 \cdot \Delta v \cdot R \cdot D'} \qquad (17)$$

Relación de transformación del transformador (n)

La relación de vueltas del transformador para el convertidor *Flyback* está dada por la siguiente ecuación:

$$n = \frac{V}{V_g} \qquad (18)$$

Modelo promedio del convertidor *Flyback.*

A partir de las ecuaciones de balance de voltio segundo para el inductor y balance de carga para el capacitor, el modelado consiste en obtener las ecuaciones diferenciales que describen el comportamiento del voltaje y la corriente de salida del convertidor.

Del balance de voltio segundo (7) se obtiene la ecuación diferencial:

$$\langle VL \rangle = L\frac{di_L}{dt} \qquad (19)$$

$$L\frac{di_L}{dt} = V_g \cdot d - \frac{v}{n} \cdot d' \qquad (20)$$

Despejando *L* de la ecuación (20) se obtiene:

$$\frac{di_L}{dt} = \frac{1}{L}\left(V_g \cdot d - \frac{v}{n} \cdot d'\right) \qquad (21)$$

Del balance de carga (11) se obtiene la siguiente ecuación diferencial:

$$\langle IC \rangle = C\frac{dv_c}{dt} \qquad (22)$$

$$C\frac{dv}{dt} = d\left(-\frac{V}{R}\right) + d'\left(\frac{I_l}{n} - \frac{V}{R}\right) \qquad (23)$$

Despejando *C* de la ecuación (23) y simplificando se obtiene:

$$\frac{dv}{dt} = \frac{1}{C}\left(\frac{i_l}{n} \cdot d' - \frac{v}{R}\right) \qquad (24)$$

Las ecuaciones diferenciales obtenidas (21) y (24) representan el modelo promedio del convertidor se utilizarán para la discretización del sistema y generación de código para ser programado en un DSP y así poder realizar la implementación de *"Hardware in the loop"* (*HIL*) para el convertidor DC-DC.

DISEÑO Y CALCULOS DEL CONVERTIDOR FLYBACK.

El primer paso para la implementación de la plataforma de *"Hardware in the loop"*es realizar el diseño del convertidor, para este caso se toma el convertidor *FlyBack* y sus ecuaciones. Las especificaciones para el diseño del convertidor son:

- Vg= 12 V (Vin),
- V= 250 V (Vout),
- Pout= 100 W,

- Rizado de voltaje 0.5% del voltaje a la salida,
- Rizado de corriente del inductor del 5 %,
- Fs= 80 Khz

Aplicando la aproximación de pequeñas señales y las ecuaciones del convertidor estudiadas anteriormente, se implementa un script en *Matlab* para facilitar los cálculos y se obtiene:

- Duty = 0.5
- I_L= 16.6667 A.
- Δ_I= 0.8333 A.
- Δ_V= 1.2500 V.
- R= 625 Ohm.
- n= 21.
- L= 47 *uH*
- C= 1 *uf*
- R_L= *6.38 mΩ.*
- *Ron= 0.150 Ω.*

Se realiza la simulación del convertidor con los parámetros de diseño obtenidos adicionando perdidas en inductor y transistor como se aprecia en la siguiente figura.

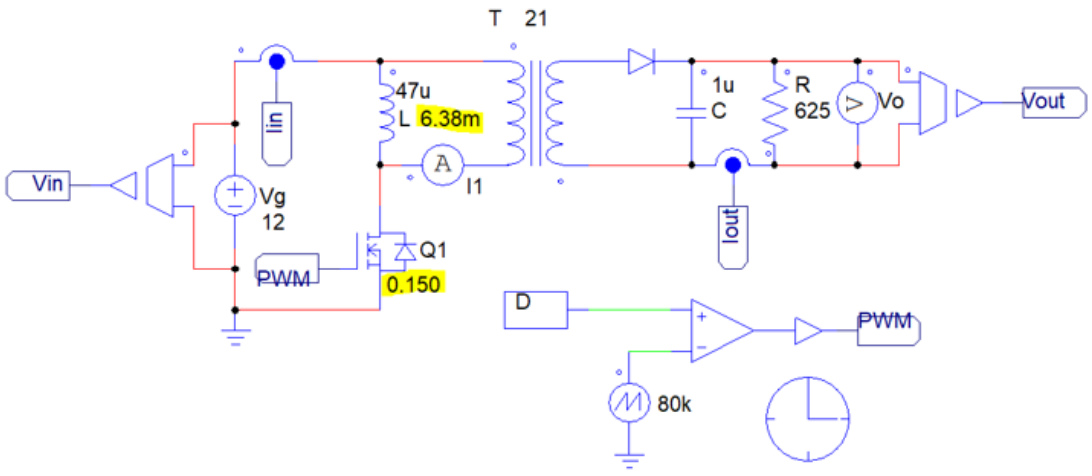
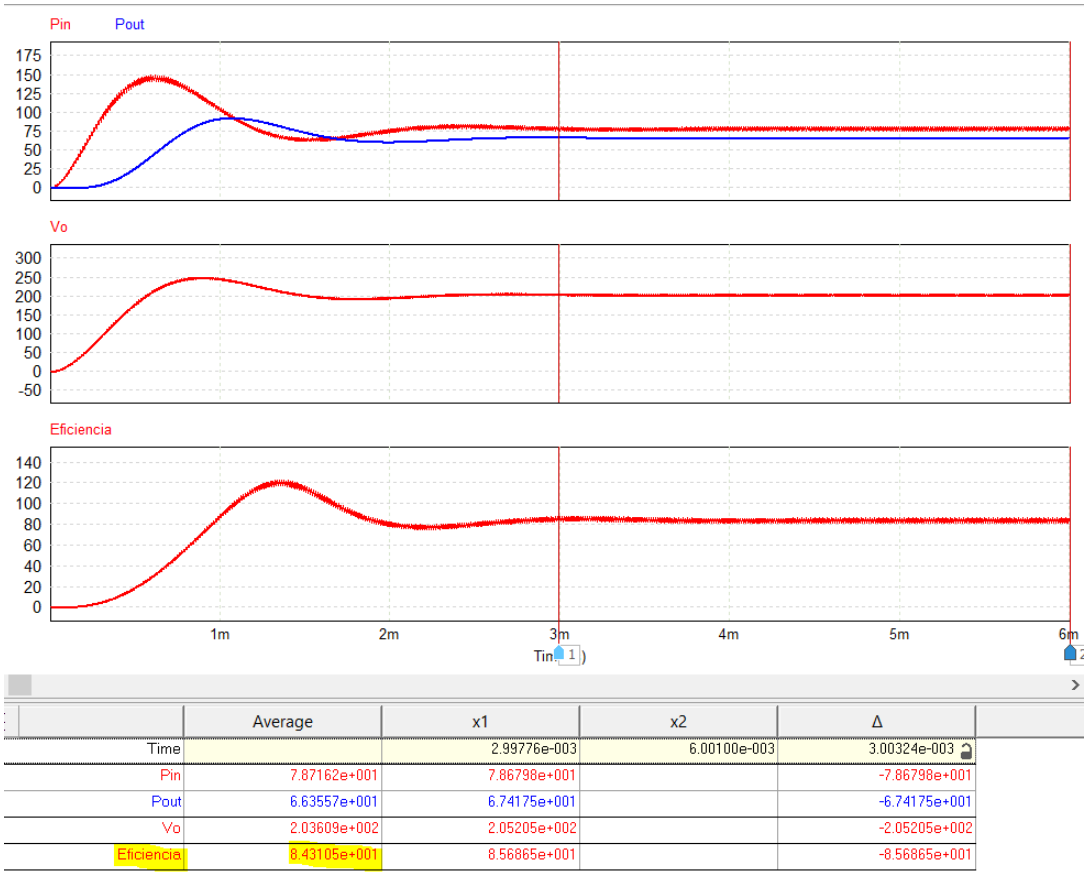


Figura 18. Convertidor FlyBack con pérdidas en inductor y transistor.

Fuente propia software PSIM

Se realiza la simulación del circuito de la Figura 19 para comprobar el funcionamiento del convertidor y que cumpla con los criterios de diseño:



	Average	x1	x2	Δ
Time		2.99776e-003	6.00100e-003	3.00324e-003
Pin	7.87162e+001	7.86798e+001		-7.86798e+001
Pout	6.63557e+001	6.74175e+001		-6.74175e+001
Vo	2.03609e+002	2.05205e+002		-2.05205e+002
Eficiencia	8.43105e+001	8.56865e+001		-8.56865e+001

Figura 19. Resultados de la simulación del convertidor FlyBack.

Fuente propia software PSIM

Según el grafico anterior se observa una buena respuesta en la simulación realizada en PSIM y se obtiene como resultado un voltaje de salida promedio de 205 V y una eficiencia del 84,3% debido a las perdidas aplicadas. Sin perdidas se tiene un voltaje de salida promedio de 250 V.

DISCRETIZACIÓN DEL CONVERTIDOR A PARTIR DEL MODELO MATEMÁTICO.

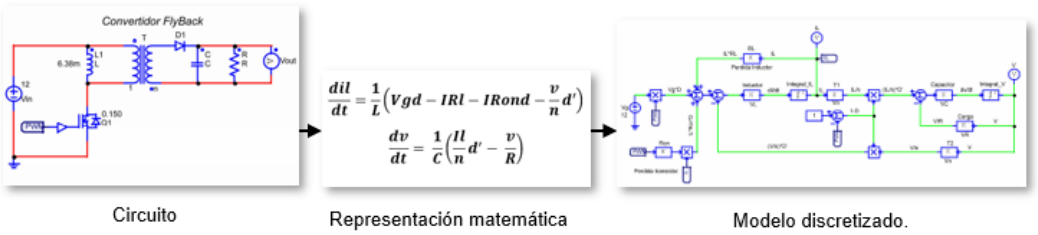


Figura 20. Modelo matemático y discretización del convertidor Flyback.

Fuente propia

El siguiente paso consiste en discretizar por medio de diagrama de bloques el convertidor diseñado para poder realizar la generación de código usando *SimCoder*. Para esto se parte del modelo matemático no lineal o modelo promedio del convertidor a partir de sus ecuaciones diferenciales:

Ecuación diferencial para i_L:

$$\frac{di_L}{dt} = \frac{1}{L}\left(V_g \cdot d - I \cdot R_l - I \cdot Ron \cdot d - \frac{v}{n} \cdot d'\right) \qquad (25)$$

Ecuación diferencial para v:

$$\frac{dv}{dt} = \frac{1}{C}\left(\frac{i_l}{n} \cdot d' - \frac{v}{R}\right) \qquad (26)$$

Se realiza la implementación del modelo matemático no lineal a partir de las ecuaciones (25) y (26) usando simulink con el objetivo de comparar la simulación de PSIM con respecto al modelo promedio.

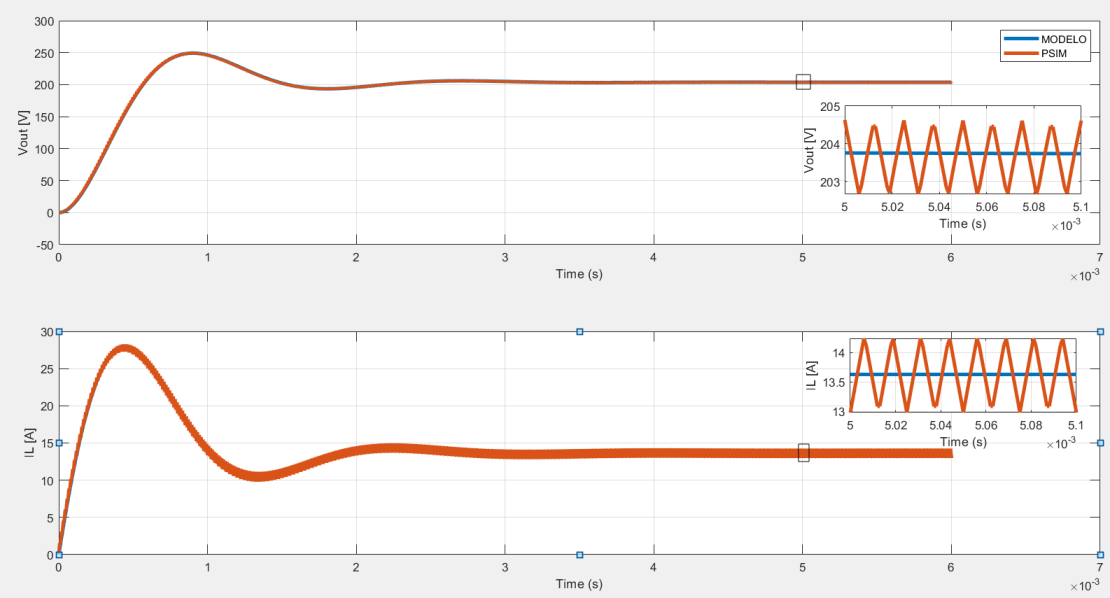


Figura 21. Modelo no lineal promediado vs simulación en PSIM.

Fuente propia software Matlab- Simulink.

En la Figura 21, la línea azul representa el modelo matemático no lineal del convertidor y se observa que este es el valor promedio del resultado de la simulación de PSIM (Línea roja), tanto para el voltaje de salida como para la corriente del inductor. Esto garantiza que las ecuaciones diferenciales (25), (26) representan el convertidor diseñado.

El siguiente paso consiste en pasar del modelo matemático a un modelo discreto en diagrama de bloques utilizando elementos de control digital compatibles con la librería de *SimCoder* para posteriormente poder generar el código y programar en el hardware de destino. Se deben tener presente las siguientes ecuaciones para la construcción del diagrama de bloques.

$$\int \frac{di_L}{dt} = i_L + i_L(0) \qquad (27)$$

$$\int \frac{dv}{dt} = v + v(0) \qquad (28)$$

A partir de las ecuaciones 25,26, 27 y 28 se genera un diagrama de bloques con componentes de control digital compatibles con *SimCoder* y se obtiene:

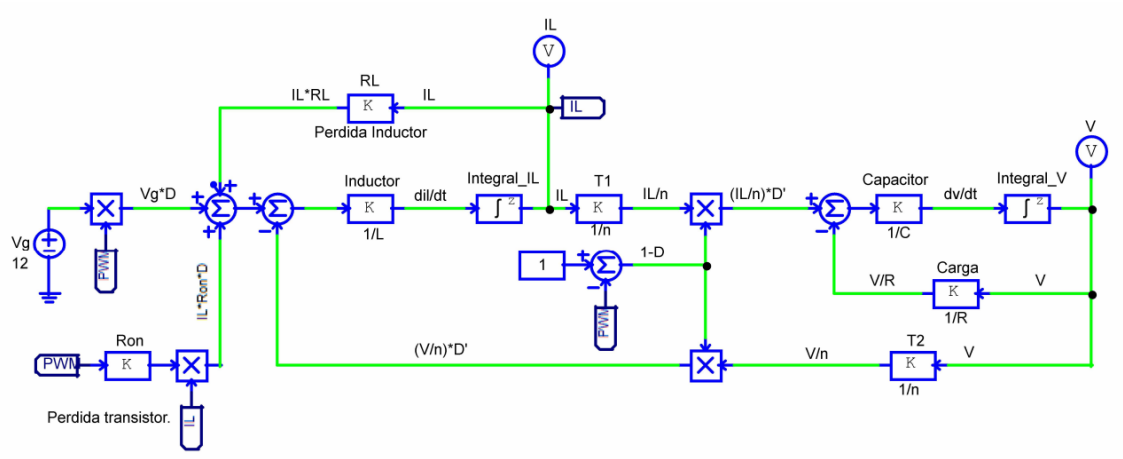


Figura 22. Modelo discreto del convertidor Flyback con pérdidas en inductor y transistor.

Fuente propia software PSIM

La Figura 22 es la representación en diagrama de bloques del convertidor *FlyBack*, partiendo del modelo matemático y utilizando elementos compatibles con *SimCoder*.

Se realiza una comparación de los resultados entre el modelo analógico o continuo (Simulación del circuito) y el modelo digital o discreto (Diagrama de bloques) y se observa que sean muy similares. Los parámetros utilizados para la simulación se encuentran en la siguiente tabla:

Parámetros del sistema			
Variable		**Valor**	**Unidad**
Resistencia de carga	R	625	*Ω*
Inductor	L	47	*uH*
Capacitor	C	1	*uf*
Relación de transformación	n	21	*A/D*
Perdida en el Mosfet.	Ron	0,15	*Ω*
Perdida en el inductor.	Rl	6,38	*mΩ*
Frecuencia de conmutación	Fs	80	*Khz*
Frecuencia de muestreo	Fm	20	*Mhz*
Voltaje de entrada	Vg	12	*V*
Duty	D	0,5	*A/D*

Tabla 2. Parámetros para la simulación del modelo.

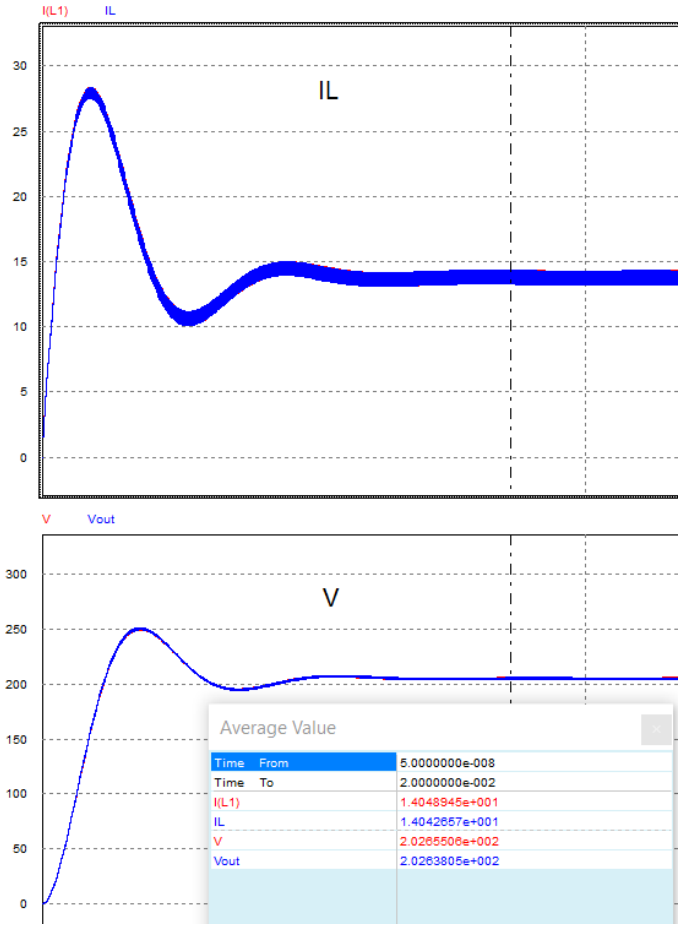


Figura 23. Comparación resultados entre circuito y modelo discreto.

Fuente propia software PSIM.

En la Figura 23, el resultado en color rojo representa a la simulación de circuito y el resultado en color azul a el modelo discreto, ambos resultados son iguales lo que valida que el modelo discreto se desarrolló correctamente.

Se utiliza una frecuencia de muestreo de 20Mhz en los integradores digitales, para poder obtener este resultado. Si se trabaja con frecuencias de muestreo inferiores la respuesta del convertido discreto es más lenta lo que genera un error en el resultado.

CONFIGURACIÓN DE ENTRADAS Y SALIDAS DEL CONVERTIDOR.

Validado el modelo discreto se procede a adicionar elementos de hardware al diagrama de bloques para luego poder hacer la programación en la DSP, para el caso de la planta (Convertidor *FlyBack*) se cuenta con una entrada y una salida. La entrada equivale a la señal PWM que conmuta al transistor y que varía el ciclo de trabajo (*Duty*), y La salida es una señala escalizada del voltaje de salida (*V*) en forma de señal PWM que se pasa por un filtro pasa bajos RC para obtener una señal analógica, esto debido a que no se contaba con un conversor digital análogo que soportara la velocidad del modelo decreto. Para el caso de la entrada se utiliza un módulo de entradas digitales de *SimCoder* y para la salida un módulo de salidas digitales respectivamente.

Debido a que la DSP trabaja con niveles de voltaje entre 0 y 3.3 V, se debe garantizar que tanto el voltaje de entrada como el de salida no supere estos niveles. Para el caso de la salida de voltaje se realiza una escalización para poder medir y controlar la respuesta de la planta.

Escalización:

La escalización consiste en reducir el valor de la variable analógica que se requiere controlar a niveles admisibles por el hardware de destino. Para el caso de la DSP el valor máximo de voltaje de salida es de 3.3 v, tomando este valor y el voltaje de salida máximo al que puede llegar el convertidor se realiza la siguiente ecuación:

$$Escalizacion = \frac{V_{maxDSP}}{V_{maxconvert}} = \frac{3.3V}{330V} = 0.01 \quad (29)$$

Según la ecuación anterior (29) el valor de voltaje de salida del modelo se multica por 0.01 y de esta manera se obtiene el valor escalizado, ejemplo.

$$Vscal = V * 0.01 = 205V * 0.01 = 2.05V \quad (30)$$

Para incluir la escalización dentro del modelo discreto y posteriormente en la generación del código, se utiliza una ganancia de 0.01 después del voltaje de salida.

Convertidor digital analógico (DAC)

En los tutoriales de PSIM y *SimCoder* se encuentran ejemplos para utilizar un convertidor digital análogo por comunicación SPI ("*Serial Peripheral Interface*"), pero como se explicó anteriormente estos convertidores son de difícil consecución por lo que se realiza un conversor digital análogo PWM. Este consiste en tomar la señal escalizada de voltaje y convertirla en una señal PWM conectada a una salida digital del DSP. Luego, se diseña un filtro pasa bajo de primer orden RC que recibe la señal PWM y la convierte en una señal analógica. La señal PWM de salida queda programada en la DSP y el filtro se implementa de manera física.

Teniendo en cuenta lo anterior se le adiciona al modelo discreto en diagrama de bloques la salida análoga de voltaje.

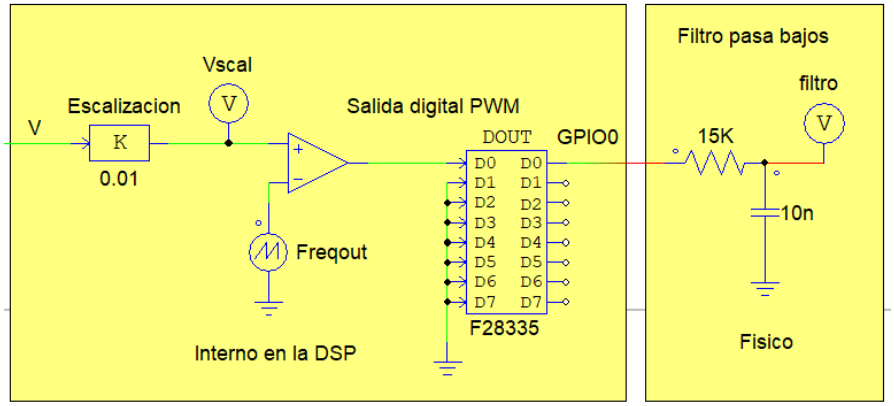


Figura 24. Salida de voltaje PWM y Filtro pasa bajos.

Fuente propia software PSIM

En la Figura 24 se puede observar el bloque de ganancia con la escalización, el generador PWM que utiliza una frecuencia igual a la frecuencia de entrada la cual es de 80KHz y el módulo de salidas digitales de la librería de *SimCoder*. Estos componentes quedan dentro de la generación del código y programados en el DSP.

Entrada PWM externa.

Luego de configurar la salida del modelo discreto, se procede con la configuración de la entrada PWM que realizara los cambios del ciclo de trabajo dentro de la planta, La entrada PWM se configura poniendo entre el modelo y el generador PWM de entrada un módulo de entradas digitales de la librería de *SimCoder*.

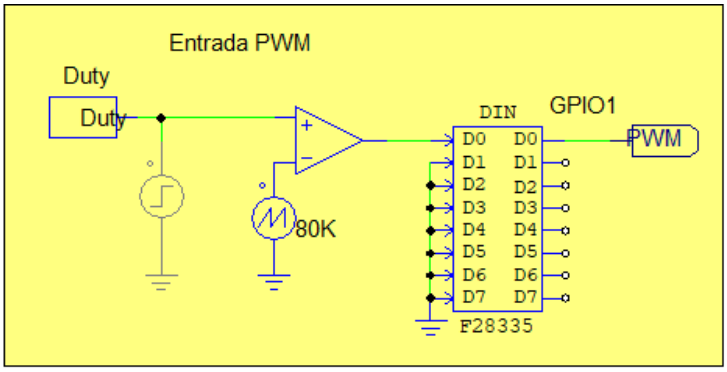


Figura 25. Configuración entrada PWM del modelo discreto.

Fuente propia software PSIM

El generador PWM que se encuentra antes de la entrada digital se programa en la DSP que contiene el controlador y se conecta al puerto programado en la DSP que contiene el modelo discreto.

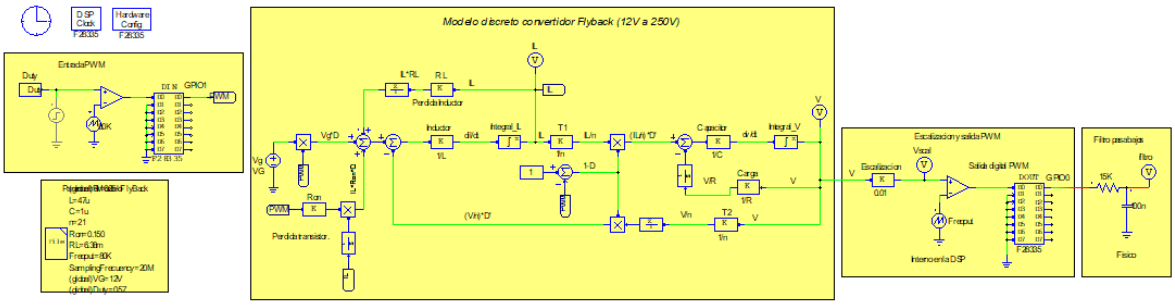


Figura 26. Modelo discreto del convertidor FlyBack con entradas y salidas.

Fuente propia software PSIM

Como se aprecia en la Figura 26. En este punto ya se encuentra diseñado todo lo que contiene el modelo discreto del convertidor *FlyBack*, se simula y valida el comportamiento del convertidor y se procede con la generación automática de código y programación de la DSP usando *Code Composer Studio*.

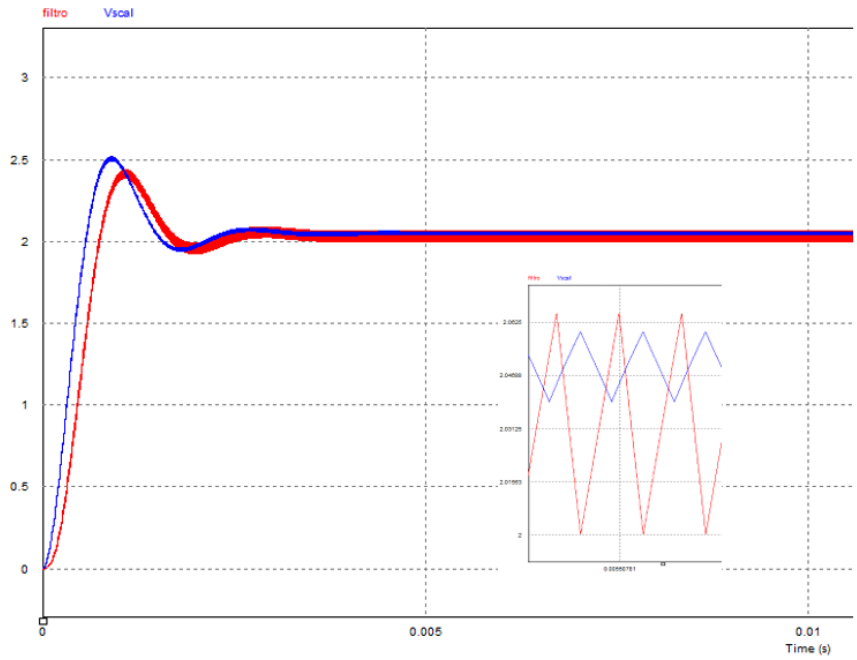


Figura 27. Simulación de señal escalizada vs señal analógica filtrada.

Fuente propia software PSIM

GENERACIÓN AUTOMÁTICA DE CÓDIGO.

En la Figura 27, la señal de color azul representa el voltaje de salida escalizado y la señal en rojo representa el voltaje escalizado filtrado, es importante comprobar que la señal filtrada tenga la misma frecuencia de la señal escalizada.

Para la generación automática de código se utiliza como guía el manual de *SimCoder* y el tutorial *"Auto Code Generation for F2833X Target".* También, como entregable de este trabajo se construye una guía donde se explica el paso a paso de como generar el código y posteriormente programar en la DSP.

A continuación, se muestra un fragmento del código generado del modelo discreto utilizando *SimCoder,* en el apéndice se encuentra el código completo.

```
/*********************************************************************************
// This code is created by SimCoder Version 11.1.5.1 for F2833x Hardware Target
//
// SimCoder is copyright by Powersim Inc., 2009-2018
//
// Date: November 04, 2021 11:01:04
*********************************************************************************/
#include  <math.h>
#include  "PS_bios.h"
typedef float DefaultType;
#define   GetCurTime() PS_GetSysTimer()
#define   PWM_IN_CHECK      // To lower PWM value setting time, comment out this




interrupt void Task();


const Uint16 PSD_CpuClock = 150;  // MHz
extern    DefaultType         fGblIL;
extern    DefaultType         fGblV;
extern    DefaultType         fGblUDELAY1;
extern    DefaultType         fGblUDELAY2;
extern    DefaultType         fGblVscal;
extern    DefaultType         fGblUDELAY3;
extern    DefaultType         fGblUDELAY4;




// Parameters in parameter file _ParamFile1
DefaultType         R = 625.0;
#define   L         (4.7e-005)
#define   C         (1e-006)
#define   n         21.0
#define   Ron       0.150
#define   RL        0.00638
#define   Freqout   80000.0
#define   SamplingFrecuency 20000000.0
DefaultType         VG = 12.0;
```

Figura 28. Fragmento de código generado a partir del modelo discreto del convertidor FlyBack.

Fuente propia software PSIM

***HARDWARE IN THE LOOP* (HIL) EN LAZO DE CONTROL ABIERTO.**

Teniendo programado el modelo discreto en una DSP, el siguiente paso en la implementación de la plataforma de *"Hardware in the loop"* (HIL), es comprobar el funcionamiento de la planta en lazo de control abierto. Esto se logra generando desde otra DSP una señal PWM con Duty configurable que se conecta al DSP que contiene el modelo. De esta manera se realizan cambios en la variable de entrada (Duty) y se comprueba la respuesta a la salida ante estos cambios.

Generador PWM.

En otra DSP se programa una salida PWM con Duty configurable usando una variable global para poder cambiarlo en tiempo real. Esta programación se hace en un archivo nuevo de PSIM diferente a donde se realizó el modelo. Se utiliza el módulo PWM de una fase que se encuentra en la librería de *SimCoder*, se genera el código y se programa la DSP.

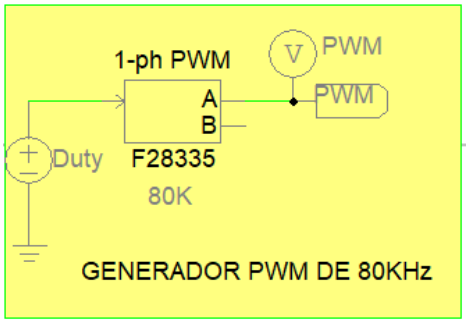


Figura 29. Generador PWM de una fase.

Fuente propia software PSIM

En la sección 6.4 del manual de *SimCoder* (Powersim Inc, 2020) se encuentra la explicación y configuración del generador PWM.

Teniendo programado en una DSP el modelo Figura 26, y en otra el generador PWM con Duty configurable Figura 29. Se realiza la conexión entre ambas tarjetas como se muestra en la siguiente figura:

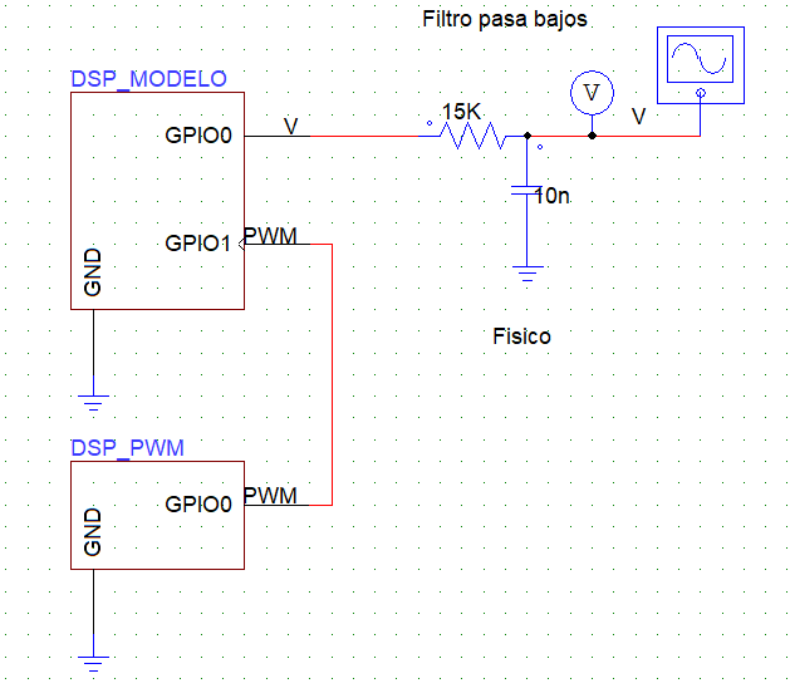


Figura 30. Diagrama de conexión en lazo abierto.

Fuente propia software PSIM

HARDWARE IN THE LOOP (HIL) EN LAZO DE CONTROL CERRADO.

Para cerrar el lazo de control y obtener la implementación final de la plataforma de *"Hardware in the loop"* (HIL), se diseña un controlador PI discreto a partir de las ecuaciones diferenciales del convertidor y se implementa en PSIM usando elementos compatibles con *SimCoder* para generar el código. Luego se programa en la DSP donde se realizó el generador PWM y se cierra el lazo de control interconectándola con la DSP que contiene modelo discreto.

Diseño del controlador.

Para el diseño del controlador discreto, se parte primero de un controlador en modo continuo, este se obtiene a partir de la linealización del sistema usando el método jacobiano, sus cálculos en el punto de operación (Tabla 3) y herramientas como SISOTOOL o PIDTOOL que tiene Matlab para el diseño de controladores.

Se debe tener presente que al estar trabajando con un sistema lineal y por tratarse de un convertidor conmutado, típicamente los controladores lineales son PI y no PID debido a la dinámica rápida con los que trabajan los convertidores de potencia. Además, los controladores PI en un convertidor DC-DC son usados para incrementar la ganancia del lazo de control en bajas frecuencias, lo que mejora la respuesta a las perturbaciones y reduce considerablemente el error en estado estable. (Erickson, 2004)

Las ecuaciones para la linealización del sistema se describen a continuación:

$$F1 = \frac{1}{L}.\left(Vgd - IRl - IRond - \frac{v}{n}d'\right) \qquad (31)$$

$$F2 = \frac{1}{C}.\left(\frac{Il}{n}d' - \frac{v}{R}\right) \qquad (32)$$

$$\dot{X} = Ax + BU \qquad (33)$$

$$Y = Cx + DU \qquad (34)$$

$$\dot{X} = \begin{bmatrix} \frac{dil}{dt} \\ \frac{dv}{dt} \end{bmatrix} \qquad (35)$$

$$X = \begin{bmatrix} IL \\ V \end{bmatrix} \qquad (36)$$

$$U = \begin{bmatrix} Vg \\ D \end{bmatrix} \qquad (37)$$

$$Y = \begin{bmatrix} V \\ IL \end{bmatrix} \qquad (38)$$

Utilizando Matlab se resuelven las ecuaciones anteriores y aparir de los resultados de las matrices jacobianas se obtiene la función de transferencia del convertidor *FlyBack*:

Gs =

$$\begin{pmatrix} -\frac{\mathrm{D}\,R\,n\,(\mathrm{D}-1)}{\sigma_1} & -\frac{R\,\mathrm{iL}\,n\,(\mathrm{RL}+L\,s+\mathrm{D}\,\mathrm{Ron})}{\sigma_1} - \frac{R\,n\,(\mathrm{D}-1)\,\sigma_2}{\sigma_1} \\ \frac{\mathrm{D}\,n^2\,\sigma_3}{\sigma_1} & \frac{n^2\,\sigma_3\,\sigma_2}{\sigma_1} - \frac{R\,\mathrm{iL}\,(\mathrm{D}-1)}{\sigma_1} \end{pmatrix}$$

where

$$\sigma_1 = R + \mathrm{D}^2\,R + \mathrm{RL}\,n^2 - 2\,\mathrm{D}\,R + \mathrm{D}\,\mathrm{Ron}\,n^2 + L\,n^2\,s + C\,R\,\mathrm{RL}\,n^2\,s + C\,L\,R\,n^2\,s^2 + C\,\mathrm{D}\,R\,\mathrm{Ron}\,n^2\,s$$

$$\sigma_2 = \mathrm{Vg} - \mathrm{Ron}\,\mathrm{iL} + \frac{v}{n}$$

Figura 31. Función de transferencia obtenida para el convertidor FlyBack.

Fuente propia software Matlab

A partir de la función de transferencia (Gs) y los valores de las variables en el punto de operación del convertidor expuestos en la tabla 3, se obtiene la función de transferencia para el controlador en modo continuo (Gvd).

```
G_v_d =

  -6.494e05 s + 8.877e09
  ----------------------
  s^2 + 3325 s + 1.492e07
```

Figura 32. función de transferencia para controlador del convertidor FlyBack.

Fuente propia software Matlab

Valores de las variables en el punto de operación.	
Variable	**Valor**
R	625
L	47 e-6
C	1 e-6
n	21
Ron	0,15
Rl	6,38 e-3
Vg	12
D	0,5

Tabla 3. Valores de las variables en el punto de operación para el cálculo de la función de transferencia.

A partir de la función de transferencia de la Figura 32 y utilizando la herramienta PIDTOOL de Matlab, se obtienen los parámetros Kp y Ki del controlador continuo.

Acción proporcional:

$$K_p = 5.7e^{-6}\ \frac{1}{v}$$

Acción Integral:

$$K_i = 1.65\ \frac{1}{v.s}$$

Con los parámetros obtenidos para el controlador, se realiza una simulación en PSIM para comprobar el funcionamiento del controlador en modo continuo:

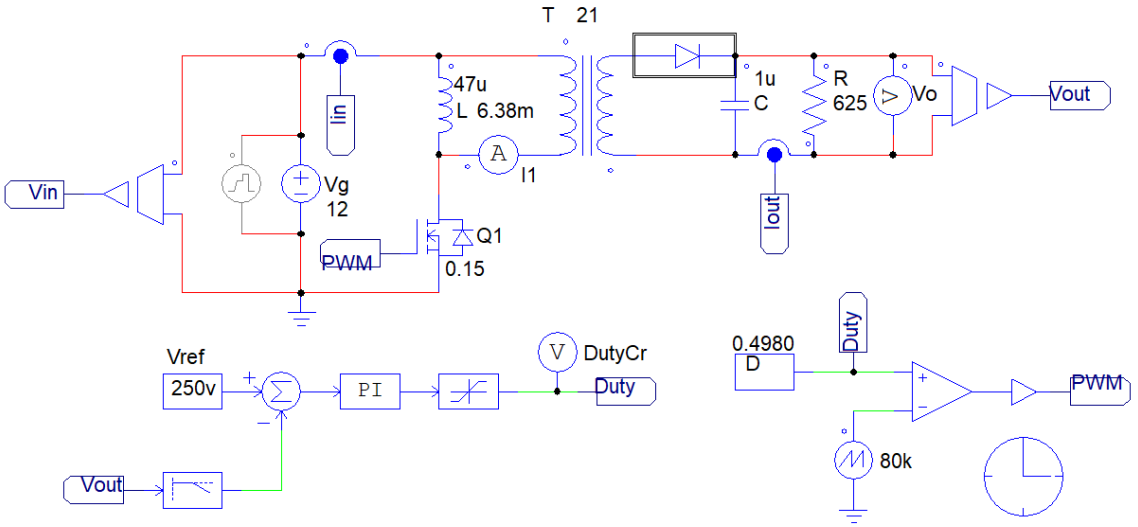


Figura 33. Convertidor FlyBack con controlador PI continuo.

Fuente propia software PSIM

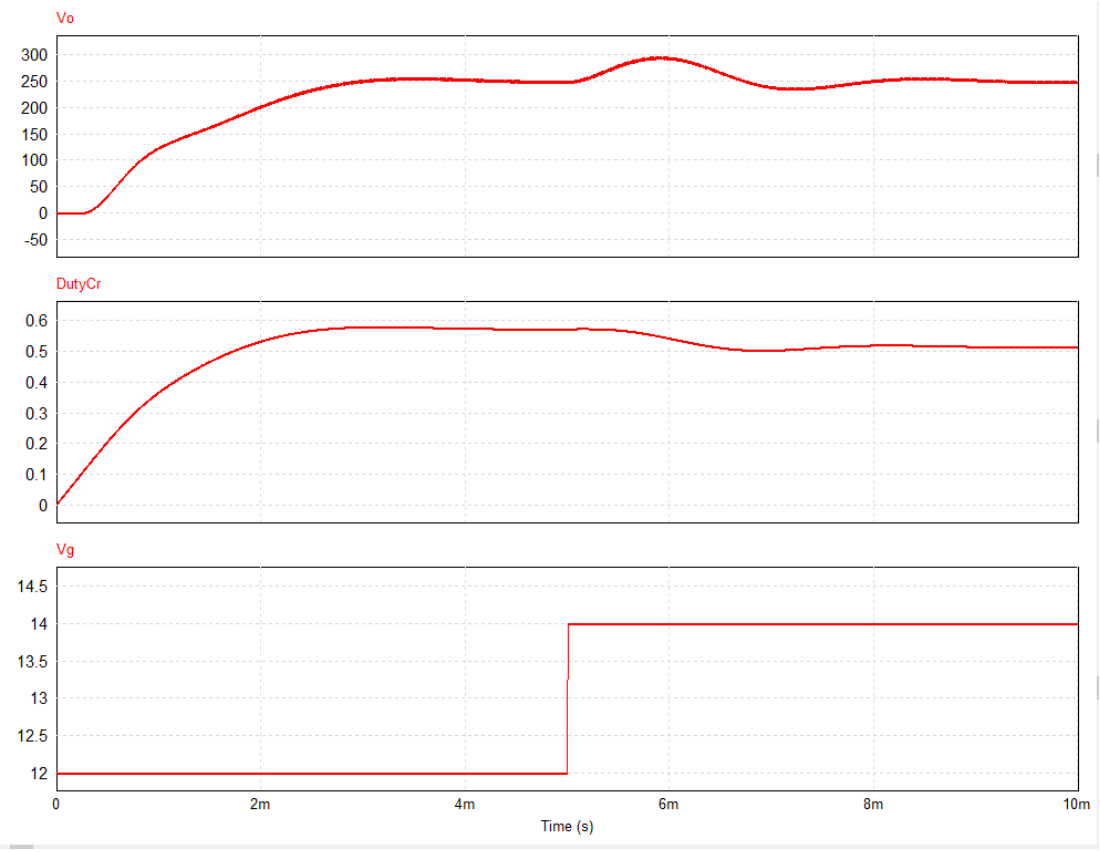


Figura 34. Respuesta de la simulación del controlador PI continúo aplicando cambios en el voltaje de entrada (Vg). Fuente propia software PSIM.

Discretización del controlador.

Luego de validar que el controlador funciona correctamente en modo continuo (Figura 34), para poder que este se pueda programar en una DSP se procede a convertirlo a un controlado discreto, para esto se usa el convertidor que trae incorporado PSIM llamado "*s2z converter*", este permite de manera rápida y sencilla pasar del modo continuo (s) al modo discreto (z) y generar el código para la programación de la DSP a través de *SimCoder*.

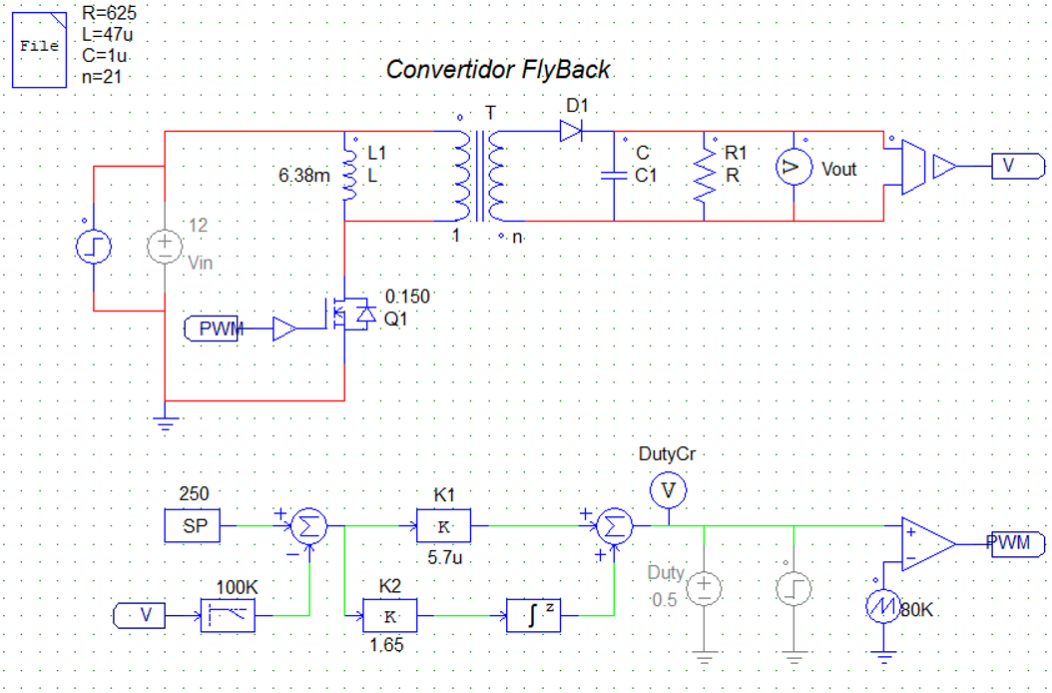


Figura 35. Convertidor FlyBack con controlador PI discreto.

Fuente propia software PSIM

A diferencia del convertidor con controlador PI continuo de la Figura 33 que contiene un bloque PI continuo y un limitador, el controlador discreto mostrado en la Figura 35 tiene dos ganancias, K1 es la acción proporcional y K2 es la acción integral integrada con un integrador digital que se suma a la ganancia proporcional, el resultado del convertidor con controlador discreto se muestra a continuación:

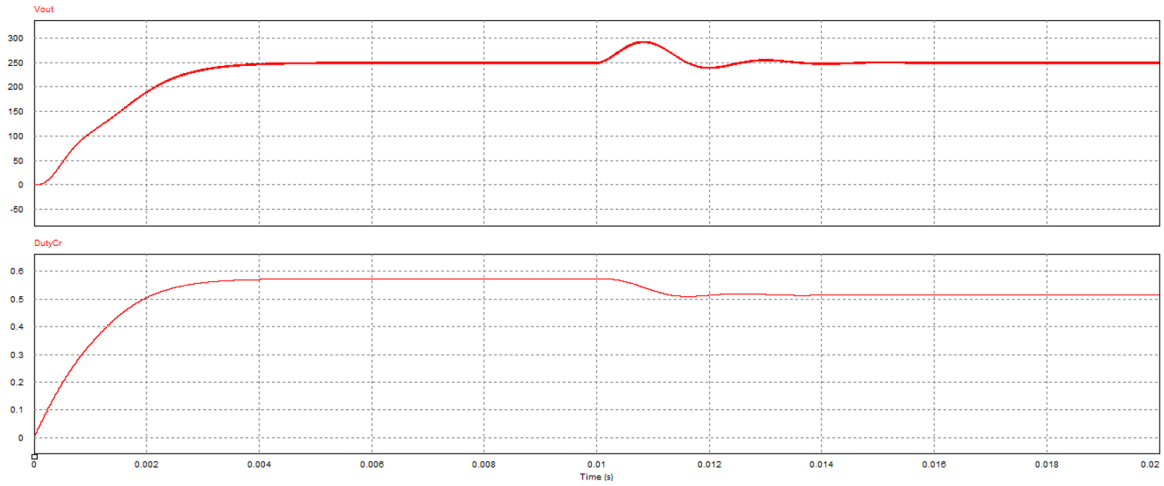


Figura 36. Respuesta de la simulación del controlador PI discreto aplicando cambios en el voltaje de entrada (Vg).

Fuente propia software PSIM

Programación del controlador PI.

Para realizar la programación del controlador en la DSP, se siguen los mismos pasos que se realizaron en la programación del modelo y el generador PWM, basta con incluir dos elementos de hardware de *SimCoder* como lo son el conversor análogo digital (ADC) y el generador PWM visto anteriormente, se genera el código, se carga y correr el programa en la DSP usando *Code Composer Studio*. A continuación, se muestra el diagrama de programación del controlador en PSIM:

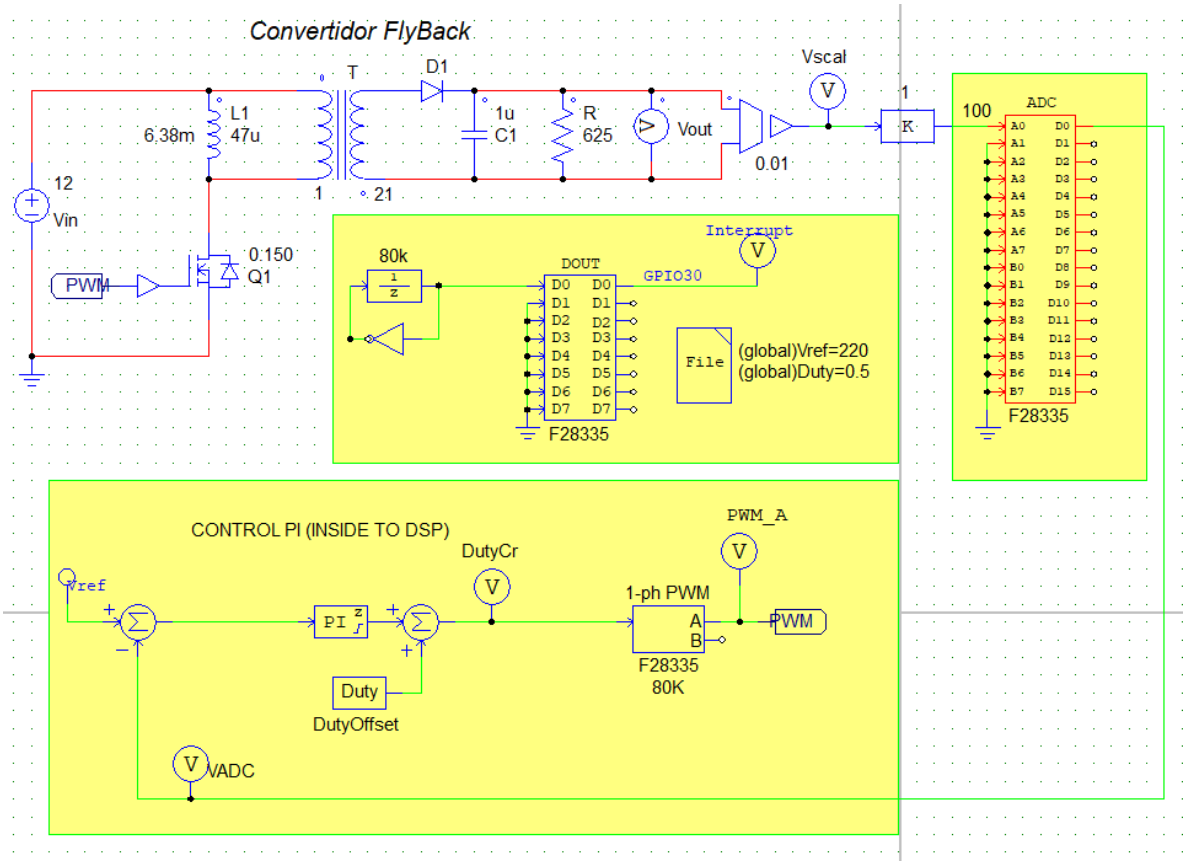


Figura 37. Diagrama de programación para el controlador PI con elementos de hardware de la DSP TMS320F28335.

Fuente propia software PSIM.

Todos los elementos resaltados en amarillo se programan en la DSP atreves de la generación del código, como se puede observar en la anterior imagen se agregan dos elementos al diagrama, uno es el conversor análogo digital (ADC) que actúa como entrada de la señal analógica y el otro es el generador PWM monofásico que actúa como salida del controlador, Además, se adiciona un bloque de interrupción para el PWM y se suma a la salida del control un valor de offset para el Duty, esto se realiza con el objetivo de que el controlador no se desborde y se pueda realizar correctamente la interacción entre los hardware.

Convertidor analógico digital (ADC)

La tarjeta F2833x proporciona un convertidor análogo digital de 12 bits y 16 canales. Se divide en dos grupos: Grupo A y Grupo B. El rango de entrada del convertidor análogo digital física es de 0 V a + 3 V, por lo que las señales análogas que se conectan a este módulo deben ser escalizadas para trabajar con estos niveles de voltaje. Esta señal es convertida a un valor

digital en la DSP y se puede utilizar un bloque de escala para volver a escalar a su valor original. Para el caso del controlador diseñado solo se utiliza el canal 0 para la lectura del voltaje de salida del convertidor.

Para la implementación final de la plataforma de *"Hardware in the loop"* se realiza el siguiente plano de conexión entre la DSP con el modelo del convertidor y la DSP con el controlador diseñado:

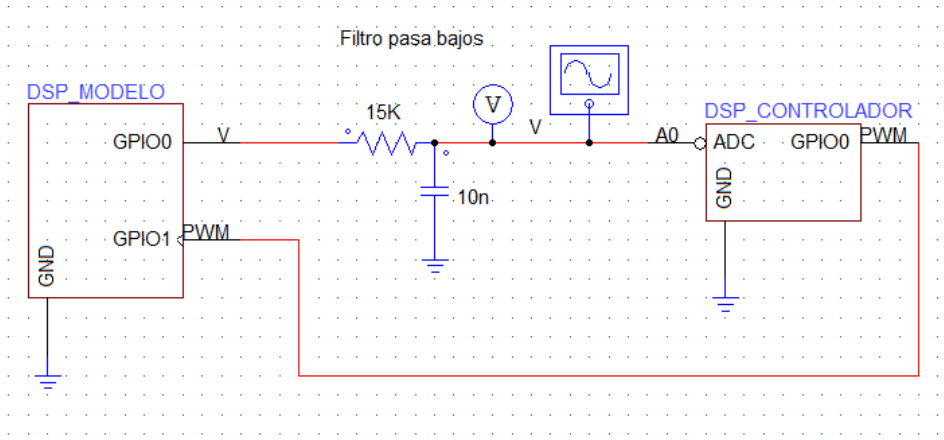


Figura 38. Diagrama de conexión en lazo cerrado.

Fuente propia software PSIM

Desarrollando esta metodología se logra la emulación del convertidor de potencia por medio del modelado matemático de la planta y la discretización del sistema para su programación en el hardware de destino usando un código generado automáticamente realizado desde la herramienta *SimCoder* de PSIM. De igual manera usando el método jacobiano para la linealización del sistema y con la ayuda de Matlab se logra diseñar un controlador para el convertidor de potencia y también generar un código automático desde PSIM para hacer programado en otra DSP. En la siguiente sección se ampliarán estos resultados y se tendrá una discusión de cada uno de estos.

4. RESULTADOS Y DISCUSIÓN

Se presentan los resultados de la implementación de la plataforma de *"Hardware in the loop"* en lazo de control abierto y posteriormente en lazo de control cerrado aplicando cambios en el voltaje de referencia (Set-Point) y comparando el valor promedio de salida de voltaje y la respuesta vista en el osciloscopio con la simulación del circuito en PSIM.

Resultados de la plataforma de *Hardware in the loop* en lazo de control abierto:

La experimentación en lazo abierto consistió en interconectar la DSP que contiene el modelo con una DSP que genera la señal PWM con Duty configurable el cual se podía cambiar en tiempo real y observar la salida de voltaje promedio de la DSP modelo en el osciloscopio. Luego se comparan estos resultados con la simulación en PSIM, esta implementación fue necesaria realizarla antes de cerrar el lazo de control para evaluar la respuesta de la planta ante los cambios en el Duty.

Los resultados de esta simulación se muestran a continuación:

Duty	Voltaje promedio simulado en PSIM (V)	Voltaje promedio en lazo abierto (V)	Error relativo (V)
0,5	2,038	1,976	3%
0,57	2,5	2,5	0%
0,4	1,485	1,452	-2%
0,3	1	0,987	-1%
0,6	2,646	2,604	-2%

Tabla 4. % de error, voltaje simulado vs voltaje en el osciloscopio ante cambios en el duty en lazo abierto de control.

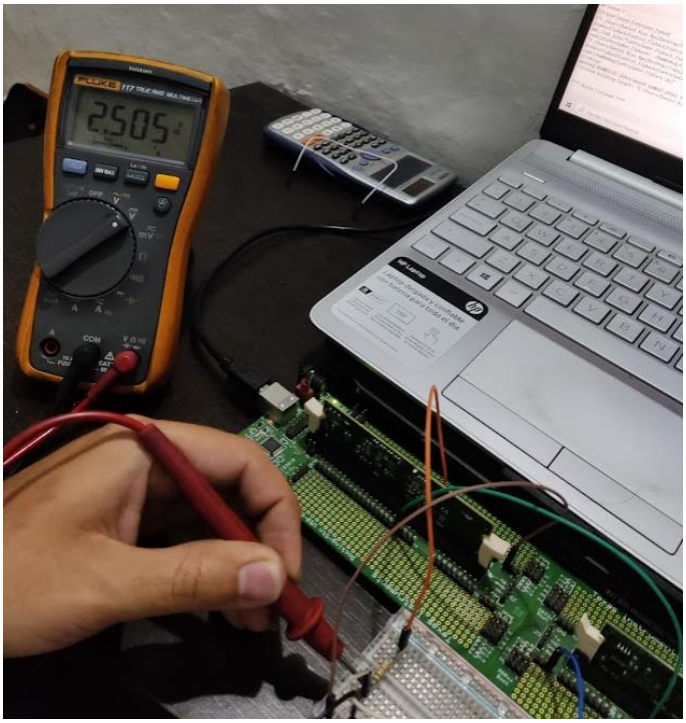


Figura 39. Lectura de voltaje promedio en lazo abierto con Duty de 0.570 y PWM externo.

Fuente propia.

El valor del Duty de 0,570 es el valor que el controlador debe mantener para lograr el valor de voltaje de referencia ("*Set Point"*) para lo que fue diseñado el convertidor.

Resultados de la implementación de la plataforma de *Hardware in the loop* (HIL) en lazo de cerrado control.

Realizando la conexión de las tarjetas DSP y el filtro pasa bajos RC según el plano de la Figura 38, se obtiene la implementación física de la plataforma *HIL* a continuación:

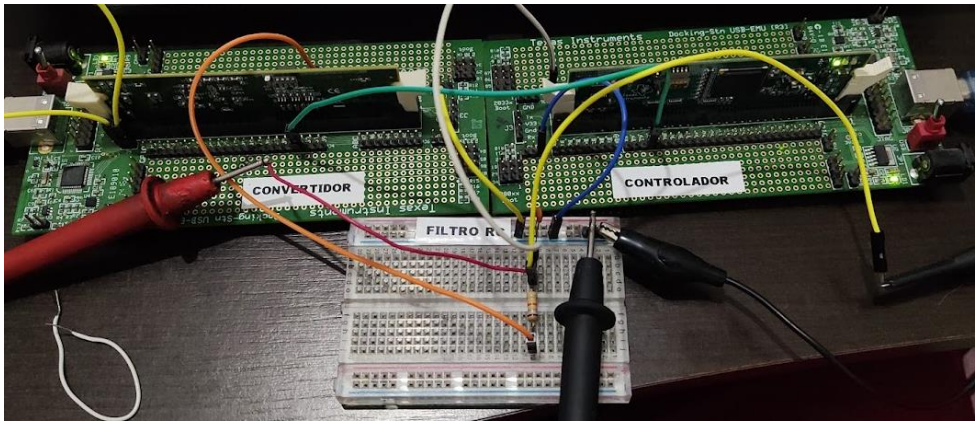


Figura 40. Montaje físico de la plataforma de *hardware in the loop* en lazo cerrado.

Fuete propio

A continuación, se realizan diferentes cambios en tiempo real en el voltaje de referencia (Vref), voltaje de entrada (V_G) y resistencia de carga (R), evaluando la respuesta de la planta y el controlador y se comparan los resultados del voltaje de salida promedio con la simulación de PSIM y se calcula el error relativo de estos.

$$\%\, error = \left|\frac{V_{HIL} - V_{simulado}}{V_{simulado}}\right| x100 \qquad (39)$$

Resultados de la plataforma *HIL* cambiando el voltaje de referencia (Vref):

Voltaje de Referencia (V)	Voltaje promedio simulado (V)	Voltaje promedio *HIL* medido. (V)	Voltaje promedio HIL escalizado (V)	Error relativo (%)
250	250	2,46	246	2%
240	237	2,34	234	1%
230	227	2,24	224	1%
220	218	2,14	214	2%
210	209	2,05	205	2%
200	204	1,98	198	3%

Tabla 5. Comparación de resultados cambiando el voltaje de referencia

Resultados de la plataforma *HIL* cambiando el voltaje de entrada (V_G):

Voltaje de entrada (V)	Voltaje promedio simulado (V)	Voltaje promedio *HIL* medido. (V)	Voltaje promedio HIL escalizado (V)	Error relativo (%)
8	0	0	0	0%
10	250	2,46	246	2%
12	250	2,46	246	2%
14	250	2,46	246	2%
15	256	2,48	248	3%
16	273	2,62	262	4%

Tabla 6. Comparación de resultados cambiando el voltaje de entrada.

Resultados de la plataforma *HIL* cambiando el valor de la resistencia de carga (R):

Resistencia de carga (Ω)	Voltaje promedio simulado (V)	Voltaje promedio *HIL* medido. (V)	Voltaje promedio HIL escalizado (V)	Error relativo (%)
625	250	2,46	246	2%
600	250	2,46	246	2%
575	250	2,46	246	2%
550	250	2,46	246	2%
500	250	2,45	245	2%
700	250	2,48	248	1%
750	250	2,48	248	1%

Tabla 7. Comparación de resultados cambiando la resistencia de carga.

En toda la simulación se observa una buena respuesta del controlador, cabe anotar que cuando se realizaron cambios en el *"set point"* de voltaje inferiores a 9 voltios, el convertidor se desborda y se va a 0, este mismo resultado también se puede ver en la simulación de PSIM. Por otro lado, se logra comprobar que el controlador es capaz de mantener el voltaje de salida en el valor de referencia mientras se hagan cambios en el voltaje de entrada o en la resistencia de carga.

Resultados en osciloscopio del valor promedio de voltaje y respuesta transitoria.

Utilizando un osciloscopio digital MDO 3024 de marca *Tektronix* facilitado por el laboratorio de electrónica y energías renovables del ITM, se analizan los resultados de la respuesta transitoria del convertidor modelado y el valor promedio de voltaje de salida aplicando cambios en el voltaje de referencia.

En la Figura 41 se muestra la implementación final de la plataforma de *"Hardware in the loop"* con cada uno de los elementos que la componen. En el ordenador (PC) se conectan la DSP con el modelo del convertidor (planta) y la DSP controladora y atreves de *code composer studio* (CCS) se visualizan las diferentes variables y se hacen los cambios en tiempo real en el voltaje de referencia, se utilizan dos puntas del osciloscopio MDO 3024, una para medir el voltaje promedio a la salida de la DSP Modelo y otra para medir el voltaje de referencia en la DSP controladora.

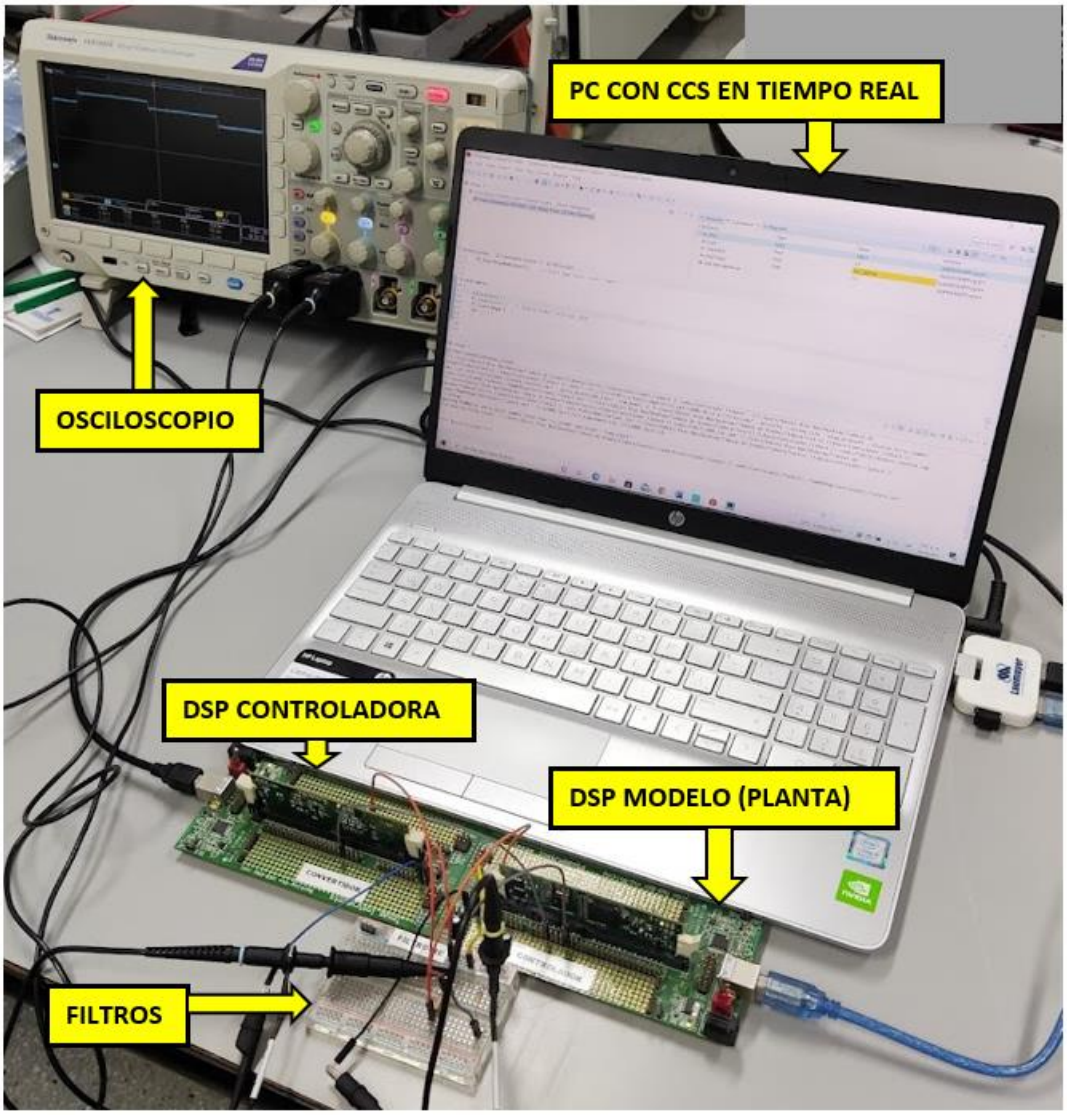


Figura 41. Implementación de plataforma de *hardware in the loop* (HIL).

Fuente propia.

Se realiza una pequeña modificación al programa esquemático del controlador con el objetivo de ver los cambios del voltaje de referencia en el osciloscopio:

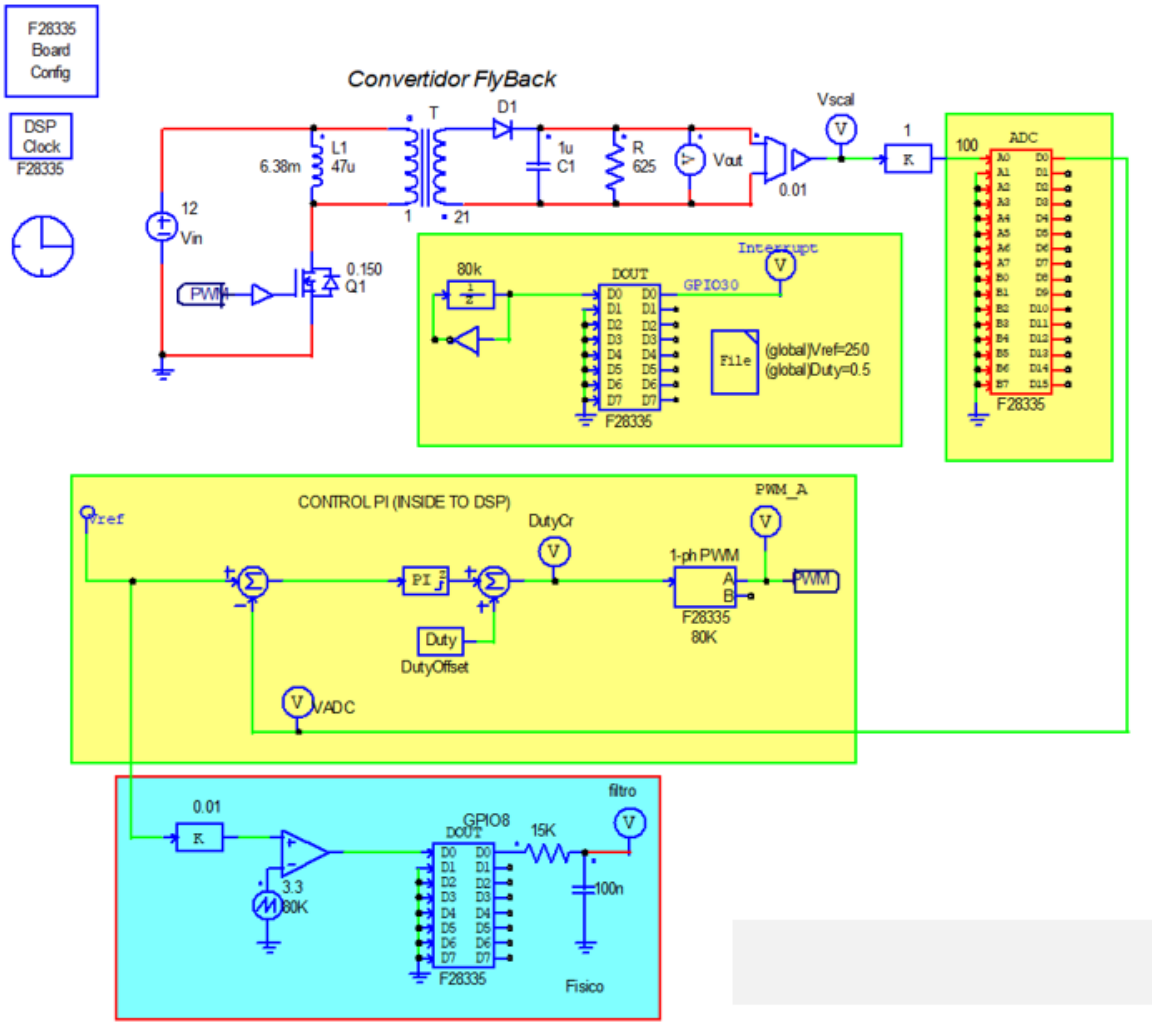


Figura 42. Diagrama de programación para el controlador PI con elementos de hardware de la DSP TMS320F28335 y salida digital PWM para observar los cambios de voltaje de referencia en el osciloscopio.

Fuente propia software PSIM

El recuadro azul de la Figura 42, representa la salida en forma de señal PWM para observar por medio de un segundo filtro pasa bajos los cambios en el voltaje de referencia en el osciloscopio, con el objetivo de poder ver la respuesta de los transitorios de manera más clara ante las perturbaciones que recibe la planta.

Respuesta de la planta ante cambio en el voltaje de referencia:

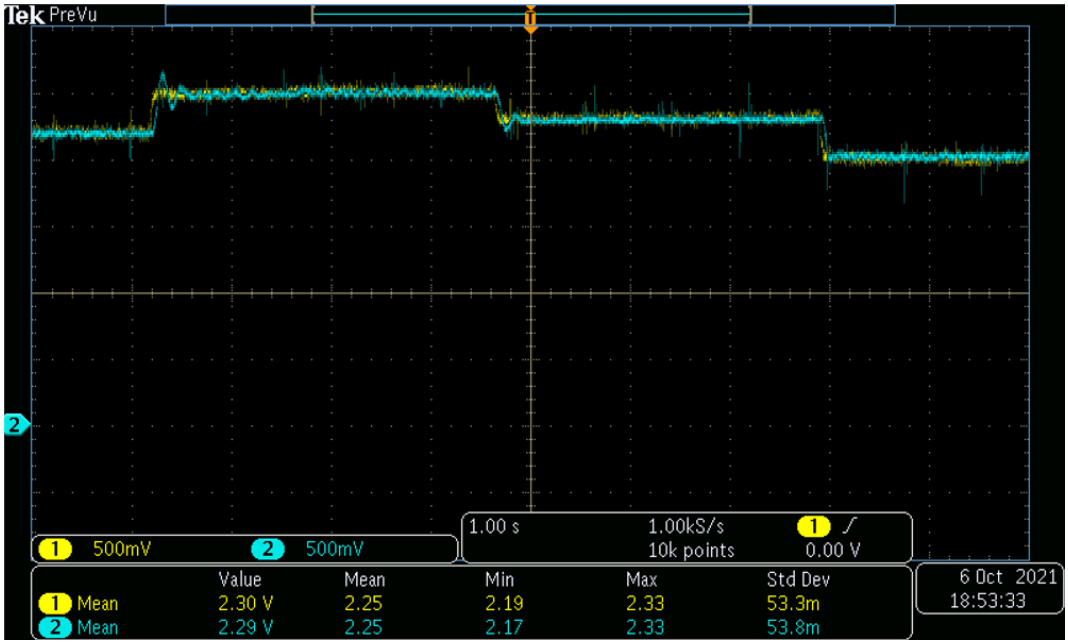


Figura 43. Lectura del voltaje de salida promedio en osciloscopio digital.

Fuente propia.

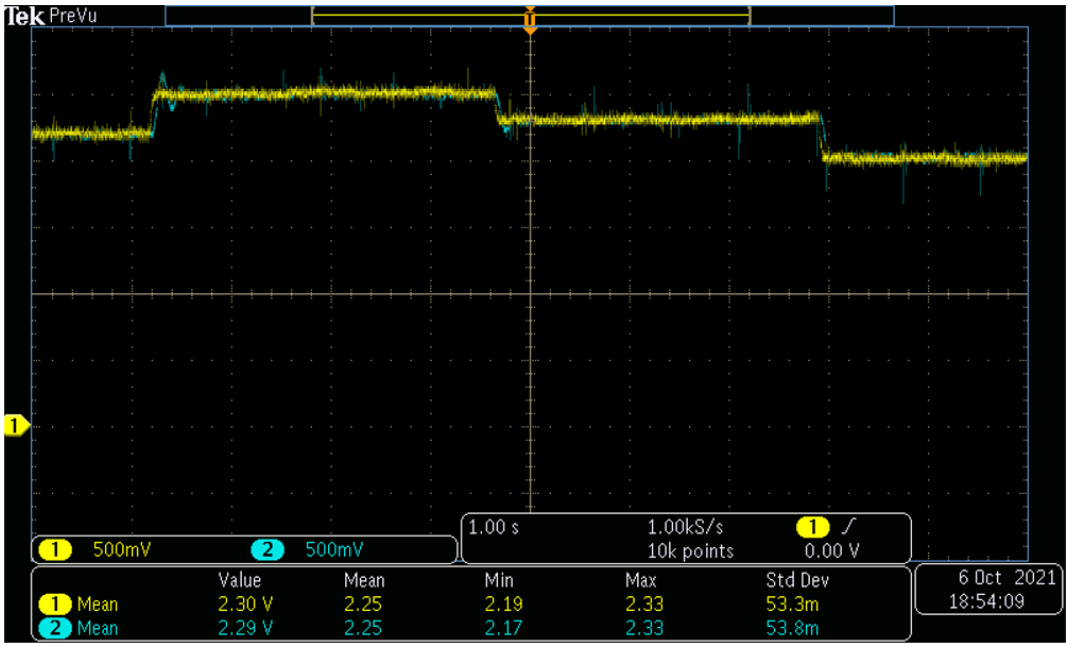


Figura 44. Lectura del voltaje de referencia en osciloscopio digital.

Fuente propia

En las Figuras 43 y 44, la línea azul representa el voltaje de salida promedio medido con el osciloscopio en la DSP que contine la planta o modelo del convertidor *FlyBack*. La línea amarilla representa el voltaje de referencia o "*Set point"* medido en la DSP que contiene el controlador. Como se puede observar se realizan varios cambios en el valor de voltaje de referencia y el controlador responde satisfactoriamente a estos manteniendo una respuesta transitoria acorde a la obtenida en la simulación y valores de voltaje de salida promedio igual al voltaje de referencia.

5. CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

Con el desarrollo de la metodología y la validación de los resultados se logró la implementación final de una plataforma de *"Hardware in the loop"* con interacción entre dos DSP´s esto es, planta y controlador, logrando obtener una respuesta a un modelo de voltaje promedio de salida y un controlador que responde adecuadamente a las perturbaciones del sistema. Este resultado marca un hito en este tipo de implementación ya que las grandes industrias que estudian la electrónica de potencia y usan plataformas de este tipo para el desarrollo de sus productos utilizando hardware como FPGA de alto costo y equipos como *Typhoon* HIL para su implementación.

Con los resultados obtenidos en este trabajo se logra concluir que es posible la implementación del convertidor *FlyBack* estudiado de manera real sin tener mayores inconvenientes en su elaboración, de igual manera aplicando esta metodología de desarrollo a través de *SimCoder* se puede llegar a emular cualquier tipo de convertidor de potencia a pequeña escala.

Toda la implementación de la plataforma de *"Hardware in the loop"* llevada a cabo en este trabajo queda documentada a manera de guía de laboratorio con el paso a paso y la explicación de cada uno de los elementos utilizados para implementar en un futuro convertidores de potencia que se deseen desarrollar.

Como recomendación para futuros trabajos o implementaciones se recomienda estudiar y conseguir un conversor analógico digital de alta velocidad (<25MHz) que permita comunicación SPI, esto con el fin de poder observar la frecuencia de conmutación a la salida del convertidor y tener un resultado aún más cercano a la realidad ya que el filtro pasa bajo reducía considerablemente esta frecuencia y al final el resultado obtenido se basa en un valor promedio de la salida de voltaje.

También, sería interesante lograr programar el controlador en otro hardware como Arduino o un microcontrolador ya que estos son los más comúnmente utilizado para controlar convertidores de potencia. Otra mejora que podría implementarse en un futuro es la implementación de una *interface* grafica de usuario que permita la interacción con la plataforma y la emulación de los convertidores de manera más sencilla y en línea con la interface se debería construir un módulo de conexiones rápidas con carcazas que protejan a los módulos y permitan conexiones rápidas y sencillas de la plataforma.

REFERENCIAS

- Erickson, R. &. (2004). *Fundamentals of power electronics 2nd ed.* New York: Klumer Academic Publishers.
- Garcia., A. J. (2018). *Convertidor tipo Flyback CFP para iluminacion Led.* Cantabria.: UNIVERSIDAD DE CANTABRIA.
- Martin., E. P. (2021). *Puesta en marcha de una plataforma de simulación en tiempo real para el estudio de la operacion de convertidores de potencia mediante tecnicas Hardware in the loop,.* Sevilla.: Universidad de Sevilla.
- Mathworks. (01 de Septiembre de 2021). *Diseño de control de electrónica de potencia con Simulink*. Obtenido de Pruebas hardware-in-the-loop (HIL) para electrónica de potencia: https://es.mathworks.com/solutions/power-electronics-control/hardware-in-the-loop.html
- Mendez, A. E. (2013). *Diseño e implementacion de un controlador digital mediante tecnicas de prototipado rapido para aplicaciones de electronica de potencia.* Bucaramanga: Universidad Pontificia Bolivariana.
- Ortega, V. G. (2002). *Simulador convertidores DC-DC.* Cataluña: Universidad Rovira I Virgili.
- Pastor, L. H. (2018). *Instalación fotovoltaica, convertidor Flyback.* Valencia.: Universidad Politecnica de Valencia.
- Pérez, Á. G. (2016). *DISEÑO Y SIMULACIÓN DE UN CONVERTIDOR FLYBACK PARA UN SISTEMA FOTOVOLTAICO.* Valencia.: Universidad Politecnica de Valencia.

- Powersim Inc. (2020, Mayo). *SimCoder User´s Guide.* Retrieved from www.powersimtech.com

- Rashid, M. H. (2004). *Electronica de potencia, circuitos, dispositivos y aplicaciones.* Mexico: Pearson educacion.

- Trujillo, J. J. (2017). *HARDWARE IN THE LOOP PARA LA SIMULACION DE CONVERTIDORES ELECTRONICOS DE POTENCIA.* Medellin.: INSTITUTO TECNOLOGICO METROPOLITANO.

- Wilaeba electronica. (11 de Septiembre. de 2018). *Filtro Pasa bajos Pasivo de 1er Orden RC*. Obtenido de Wilaeba electronica: https://wilaebaelectronica.blogspot.com/2017/01/filtro-pasa-bajos-pasivo-de-1er-orden-rc.html

- Z. Süto, T. D. (2014). *Matlab/Simulink Generated FPGA Based Real-time HIL Simulator and DSP controller: A Case Study.* Cordoba.: Department of Automation and Applied Informatics, Budapest University.

APÉNDICE

APÉNDICE A.

Manual para la implementación de una plataforma HIL utilizando generación automática de código por medio de la herramienta *SimCoder* de PSIM (Adjunta).

APÉNDICE B.

Guía de laboratorio para la simulación de un convertidor de potencia por medio de una plataforma HIL usando PSIM. (Adjunta)

APÉNDICE C.

Codigo c del modelo discreto generado automáticamente por *SimCoder*.

```
/*********************************************************************
**********
// This code is created by SimCoder Version 11.1.5.1 for F2833x Hardware Target
//
// SimCoder is copyright by Powersim Inc., 2009-2018
//
// Date: November 04, 2021 19:58:16
*********************************************************************
**********/
#include        <math.h>
#include        "PS_bios.h"
typedef float DefaultType;
#define         GetCurTime() PS_GetSysTimer()
#define         PWM_IN_CHECK        // To lower PWM value setting time, comment out
this line if PWM duty cycle values are strictly limited in the range.
interrupt void Task();
const Uint16 PSD_CpuClock = 150;  // MHz
```

```
extern DefaultType   fGblIL;
extern DefaultType   fGblV;
extern DefaultType   fGblUDELAY1;
extern DefaultType   fGblUDELAY2;
extern DefaultType   fGblVscal;
extern DefaultType   fGblUDELAY3;
extern DefaultType   fGblUDELAY4;

// Parameters in parameter file _ParamFile1
DefaultType   R = 625.0;
#define       L       (4.7e-005)
#define       C       (1e-006)
#define       n       21.0
#define       Ron     0.150
#define       RL      0.00638
#define       Freqout         80000.0
#define       SamplingFrecuency    20000000.0
DefaultType   VG = 12.0;
DefaultType   Duty = 0.57;
DefaultType   fGblIL = 0;
DefaultType   fGblV = 0;
DefaultType   fGblUDELAY1 = 0;
DefaultType   fGblUDELAY2 = 0;
DefaultType   fGblVscal = 0;
DefaultType   fGblUDELAY3 = 0;
DefaultType   fGblUDELAY4 = 0;
interrupt void Task()
{
```

```
        DefaultType    fUDELAY4, fCarga, fUDELAY3, fT2, fUDELAY2, fP7, fUDELAY1,
fCOMP2;
        DefaultType    fVSAW3, fEscalizacion, fIntegral_V, fCapacitor, fSUM2, fMULT2;
        DefaultType    fT1, fIntegral_IL, fInductor, fSUM4, fMULT3, fSUM3, fC4, fSUM32;
        DefaultType    fMULT4, fP5, fMULT1, fDIN1, fVg;

        PS_MaskIntr(M__INT13);
        fUDELAY1 = fGblUDELAY1;
        fUDELAY2 = fGblUDELAY2;
        fUDELAY3 = fGblUDELAY3;
        fUDELAY4 = fGblUDELAY4;

        fDIN1 = (PS_GetDigitInA() & ((Uint32)1 << 1)) ? 1 : 0;
        fVg = VG;
        fMULT1 = fVg * fDIN1;
        fP5 = fDIN1 * Ron;
        fMULT4 = fP5 * fUDELAY1;
        fSUM32 = fUDELAY2 * (-(1.0)) + fMULT1 * 1 + fMULT4 * (-(1.0));
        fC4 = 1;
        fSUM3 = fC4 - fDIN1;
        fMULT3 = fUDELAY3 * fSUM3;
        fSUM4 = fSUM32 - fMULT3;
        fInductor = fSUM4 * (1.0/L);
        {
                static DefaultType out_A = 0, in_A = 0.0;
                fIntegral_IL = out_A + 1.0/SamplingFrecuency * in_A;
                out_A = fIntegral_IL;in_A = fInductor;
        }
```

```
#ifdef  _DEBUG
        fGblIL = fIntegral_IL;
#endif


        fT1 = fIntegral_IL * (1.0/n);
        fMULT2 = fT1 * fSUM3;
        fSUM2 = fMULT2 - fUDELAY4;
        fCapacitor = fSUM2 * (1.0/C);
        {
                static DefaultType out_A = 0, in_A = 0.0;
                fIntegral_V = out_A + 1.0/SamplingFrecuency * in_A;
                out_A = fIntegral_V;in_A = fCapacitor;
        }
#ifdef  _DEBUG
        fGblV = fIntegral_V;
#endif


        fEscalizacion = fIntegral_V * 0.01;
        {
                static DefaultType count = 0.0;
                fVSAW3 = count;
                count += ((DefaultType)3.3 * Freqout) / 20000000L;
                if (count >= 3.3)
                        count -= 3.3;
        }
        fCOMP2 = (fEscalizacion > fVSAW3) ? 1 : 0;
        fGblUDELAY1 = fIntegral_IL;
        fP7 = fIntegral_IL * RL;
```

```
        fGblUDELAY2 = fP7;
#ifdef  _DEBUG
        fGblVscal = fEscalizacion;
#endif
        fT2 = fIntegral_V * (1.0/n);
        fGblUDELAY3 = fT2;
        fCarga = fIntegral_V * (1.0/R);
        fGblUDELAY4 = fCarga;
        (fCOMP2 == 0) ? PS_ClearDigitOutBitA((Uint32)1 << 0) :
PS_SetDigitOutBitA((Uint32)1 << 0);
        PS_ExitTimer1Intr();
}
void Initialize(void)
{
        PS_SysInit(30, 10);
        PS_InitTimer(0, 0);

        // Set initial states for those GPIO/AIO output ports.
        PS_ClearDigitOutBitA((Uint32)1 << 0);       // Reset GPIO0
        PS_ClearDigitOutBitA((Uint32)1 << 2);       // Reset GPIO2
        PS_InitDigitOut(2);    // Initialize GPIO2
        PS_ClearDigitOutBitA((Uint32)1 << 30);      // Reset GPIO30
        PS_InitDigitOut(30);   // Initialize GPIO30
        PS_InitDigitIn(1, 100);
        PS_InitDigitOut(0);
        PS_InitTimer(1,8L);
        PS_SetTimerIntrVector(1, Task);
        PS_StartStopPwmClock(2);    // start Timer1
```

```
}



void main()
{
        Initialize();
        PS_EnableIntr();   // Enable Global interrupt INTM
        PS_EnableDbgm();
        for (;;) {
        }
}
```

APÉNDICE C.

Codigo c del controlador discreto generado automáticamente por SimCoder

```
/*********************************************************************
**********
// This code is created by SimCoder Version 11.1.5.1 for F2833x Hardware Target
//
// SimCoder is copyright by Powersim Inc., 2009-2018
//
// Date: November 04, 2021 20:06:30
*********************************************************************
**********/
#include      <math.h>
#include      "PS_bios.h"
typedef float DefaultType;
#define       GetCurTime() PS_GetSysTimer()
```

```
#define         PWM_IN_CHECK       // To lower PWM value setting time, comment out
this line if PWM duty cycle values are strictly limited in the range.

interrupt void Task();
interrupt void Task_1();

const Uint16 PSD_CpuClock = 150;  // MHz
extern DefaultType    fGblSUMP4;
extern DefaultType    fGblDutyCr;
extern DefaultType    fGblVADC;
extern DefaultType    fGblUDELAY3;

// Parameters in parameter file _ParamFile1
DefaultType    Vref = 250.0;
DefaultType    Duty = 0.5;
DefaultType    fGblSUMP4 = 0;
DefaultType    fGblDutyCr = 0;
DefaultType    fGblVADC = 0;
DefaultType    fGblUDELAY3 = 0;
interrupt void Task()
{
        DefaultType    fCOMP3, fVSAW4, fP12, fDutyOffset, fS1, fSUM1, fTI_ADC1,
fVCC5;

        PS_MaskIntr(M__INT13);

        fTI_ADC1 = PS_GetDcAdc(0);
        fVCC5 = Vref;
```

```
        fSUM1 = fVCC5 - fTI_ADC1;
        {       // backward Euler
                static DefaultType out_A = 0.0;
                fS1 = out_A + ((5.7E-6)/((3.4545E-6)*20000000L)) * fSUM1;
                fS1 = (fS1 < 0.0) ? 0.0 : ((fS1 > 1.0) ? 1.0 : fS1);
                out_A = fS1;
                fS1 += (5.7E-6) * fSUM1;
                fS1 = (fS1 < 0.0) ? 0.0 : ((fS1 > 1.0) ? 1.0 : fS1);
        }
        fDutyOffset = Duty;
        fGblSUMP4 = fS1 + fDutyOffset;
#ifdef  _DEBUG
        fGblDutyCr = fGblSUMP4;
#endif
#ifdef  _DEBUG
        fGblVADC = fTI_ADC1;
#endif
        fP12 = fVCC5 * 0.01;
        {
                static DefaultType count = 0.0;
                fVSAW4 = count;
                count += ((DefaultType)3.3 * 80000) / 20000000L;
                if (count >= 3.3)
                        count -= 3.3;
        }
        fCOMP3 = (fP12 > fVSAW4) ? 1 : 0;
        (fCOMP3 == 0) ? PS_ClearDigitOutBitA((Uint32)1 << 8) :
PS_SetDigitOutBitA((Uint32)1 << 8);
```

```
        PS_ExitTimer1Intr();
}

interrupt void Task_1()
{
        DefaultType    fUDELAY3, fNOT1;

        fUDELAY3 = fGblUDELAY3;

        fNOT1 = (fUDELAY3 <= 0.3) ? 1 : 0;
        fGblUDELAY3 = fNOT1;
        // Start of changing PWM1(1ph) registers
        // Set Duty Cycle
#ifdef PWM_IN_CHECK
        if (fGblSUMP4 <= 0) {
                PWM_CMPA(1) = 0;
        } else if (fGblSUMP4 >= (1 + 0)) {
                PWM_CMPA(1) = PWM_TBPRD(1);
        } else {
#else   // PWM_IN_CHECK
        {
#endif
                DefaultType _val = ((fGblSUMP4 - 0) * (1.0/1));
                PWM_CMPA(1) = (int)(((Uint32)(PWM_TBPRD(1))+1) * _val);
        }
        // End of changing PWM1(1ph) registers
        (fUDELAY3 == 0) ? PS_ClearDigitOutBitA((Uint32)1 << 30) :
PS_SetDigitOutBitA((Uint32)1 << 30);
```

```
        PS_ExitAdcIntr(1, M__INT1);
}



void Initialize(void)
{
        PS_SysInit(30, 10);
        PS_StartStopPwmClock(0);    // Stop Pwm Clock
        PS_InitTimer(0, 0);

        // Set initial states for those GPIO/AIO output ports.
        PS_ClearDigitOutBitA((Uint32)1 << 8);         // Reset GPIO8
        PS_ClearDigitOutBitA((Uint32)1 << 10);        // Reset GPIO10
        PS_InitDigitOut(10);   // Initialize GPIO10
        PS_ClearDigitOutBitA((Uint32)1 << 30);        // Reset GPIO30
        PS_ResetAdcConvSeq();
        PS_SetAdcConvSeq(eAdcCascade, 0, 0, 100);
        PS_AdcInit(1, !2);

        PS_InitPwm(1, 0, (double)80000*1, 0*1e6, PWM_POSI_ONLY, 0);// pwnNo,
waveType, frequency, deadtime, outtype, UseHRPwm
        PS_SetPwmPeakOffset(1, 1, 0, 1.0/1);
        PS_SetPwmIntrType(1, ePwmIntrAdc, 1, 0);
        PS_SetPwmVector(1, ePwmIntrAdc, Task_1);
        PS_SetPwmTzAct(1, eTZHighImpedance);
        PS_SetPwm1RateSH(0);
        PS_StartPwm(1);
        PS_InitDigitOut(30);
```

```
        PS_InitDigitOut(8);
        PS_InitTimer(1,8L);
        PS_SetTimerIntrVector(1, Task);
        PS_StartStopPwmClock(2);    // Start Pwm Clock, start Timer1
}
void main()
{
        Initialize();
        PS_EnableIntr();   // Enable Global interrupt INTM
        PS_EnableDbgm();
        for (;;) {
        }
}
```

FIRMA ESTUDIANTES

FIRMA ASESOR Entrega de Correcciones del informe final

FECHA ENTREGA: 21/02/2022

FIRMA COMITÉ TRABAJO DE GRADO DE LA FACULTAD _____

RECHAZADO___ ACEPTADO____ ACEPTADO CON MODIFICACIONES_____

FECHA ENTREGA: _____

FIRMA CONSEJO DE FACULTAD_____

ACTA NO._____

FECHA ENTREGA: _____