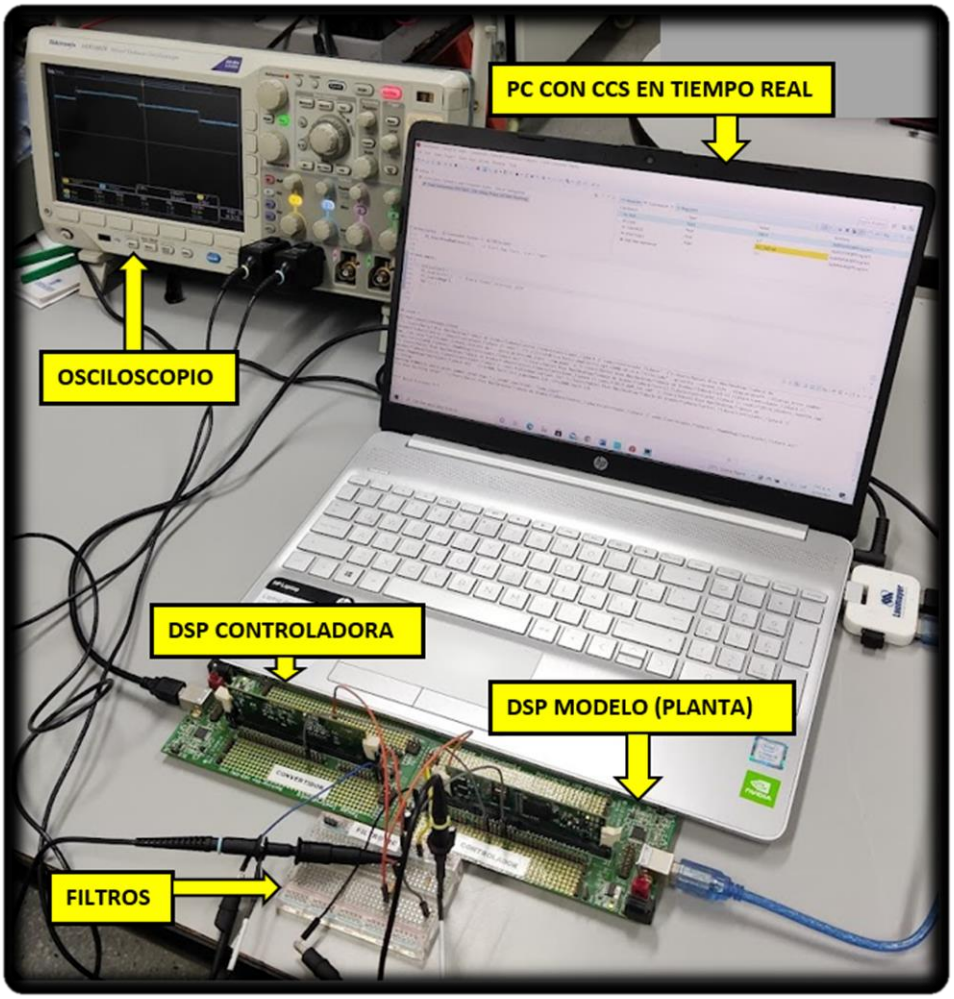
MANUAL PARA LA IMPLEMENTACIÓN DE UNA PLATAFORMA DE HARDWARE IN THE LOOP (HIL) UTILIZANDO GENERACIÓN AUTOMÁTICA DE CÓDIGO POR MEDIO DE LA HERRAMIENTA SIMCODER DE PSIM Y APLICADA A CONVERTIDORES DE POTENCIA.



Autores:
Daniel Rios Restrepo
María Alejandra Morales H

CONTENIDO

1. INTRODUCCION A LA GENERACION DE CÓDIGO.

1.1 SimCoder.

SimCoder es un módulo de PSIM que 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 instrument.* Las familias de tarjetas compatibles con SimCoder son:

Target	SimCoder Supported CPU Version
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, and 280200.
F2837x	28374S/D, 28375S/D, 28376S/D, 28377S/D, 28379S/D.
F28004x	280049, 280049C, 280048, 280048C, 280045, 280041, 280041C, 280040, 280040C.
PE-Expert4	For the PE-Expert4 DSP hardware. PE-Expert4 is a DSP development platform produced by Myway Co. It uses TI's floating-point DSP TMS320C6657 and Myway's PE-OS library.

Tabla 1. Familia de tarjetas compatibles con Simcoder.

El SimCoder se habilita desde el control de simulación en PSIM (*Simulation Control*) en la segunda pestaña como se muestra en la Figura 2:

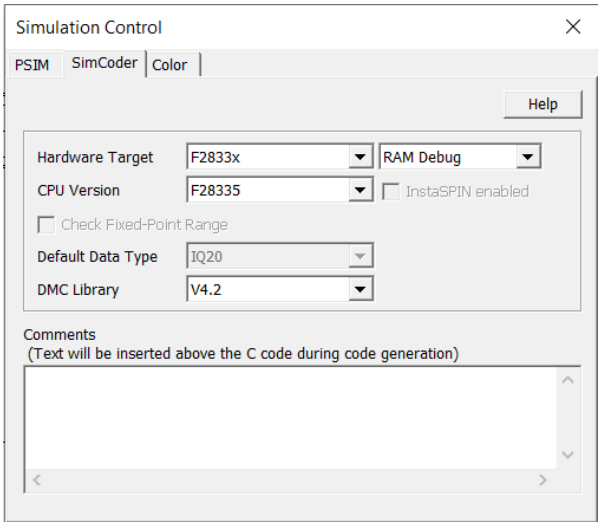


Figura 2. Ventana de configuración de SimCoder.

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 instrument* para generar códigos para control de motores DCM.

1.2 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 para la generación de código, además un gran número de elementos de la librería de PSIM estándar se pueden utilizar para la generación de código, 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 3. 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.

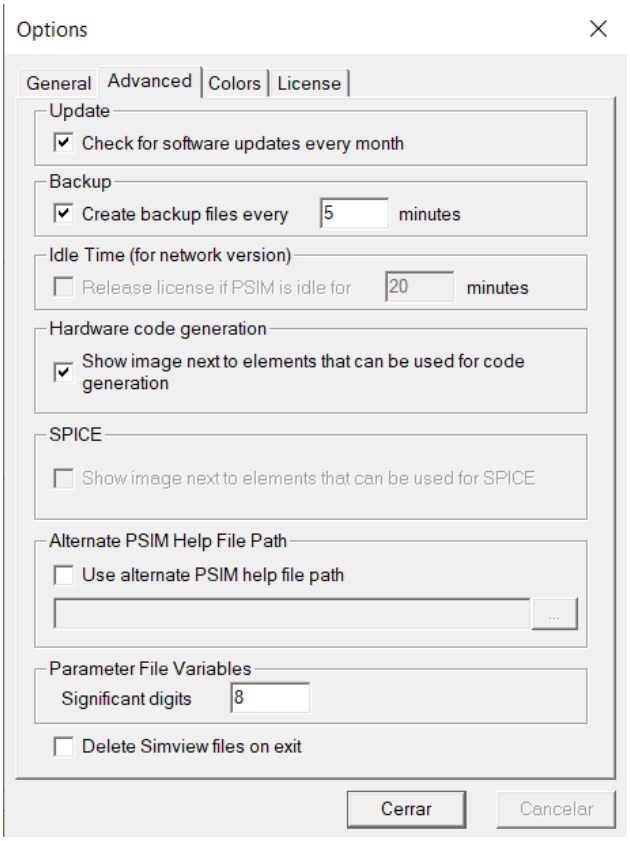


Figura 3. Opciones avanzadas.

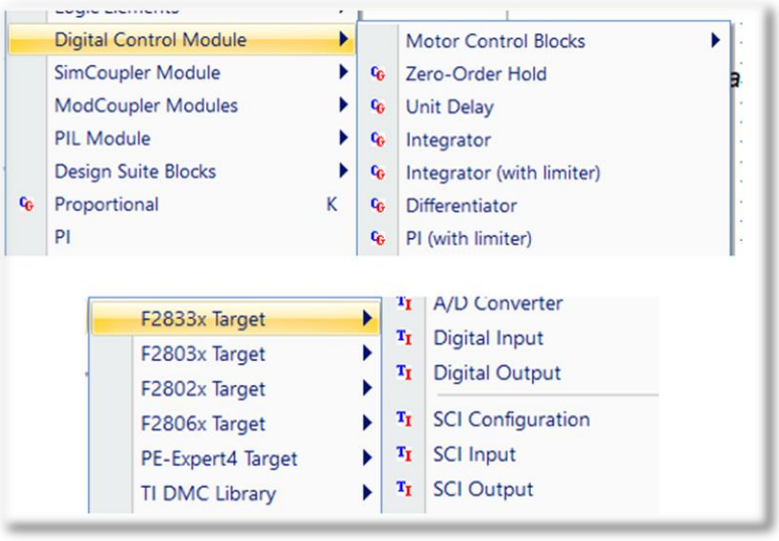


Figura 4. Elementos compatibles con generación de código.

1.3 Paso a paso para la generación de código.

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

- Diseñar y simular un sistema en PSIM con el control en dominio continuo.
- Convierta la sección de control del sistema en un dominio discreto y simule el sistema.
- Si no hay un objetivo de hardware, coloque la sección de control en un subcircuito y genere el código.
- 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. [1]

1.4 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.

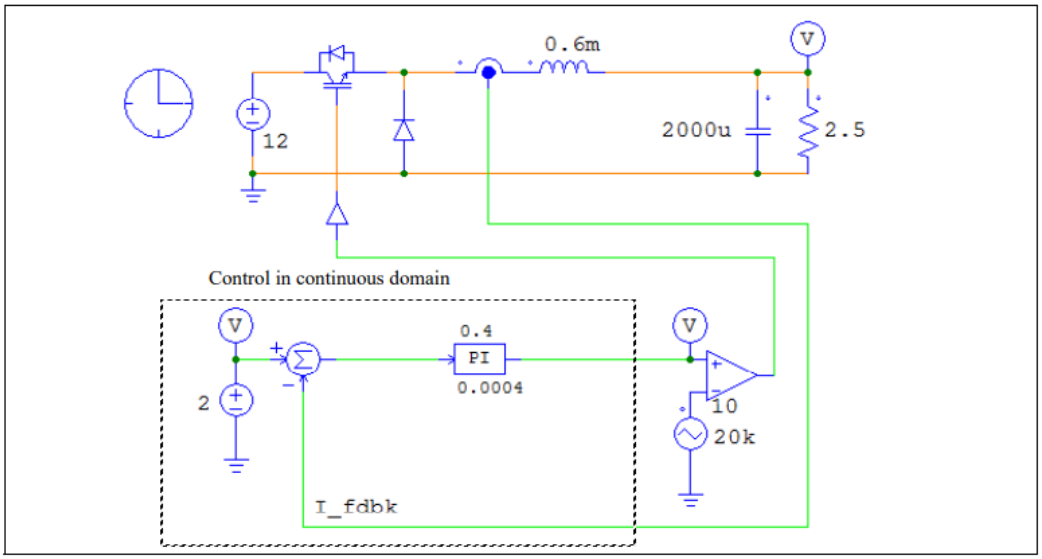


Figura 5. Sistema de convertidor CC en modo de control continuo.

1.5 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.

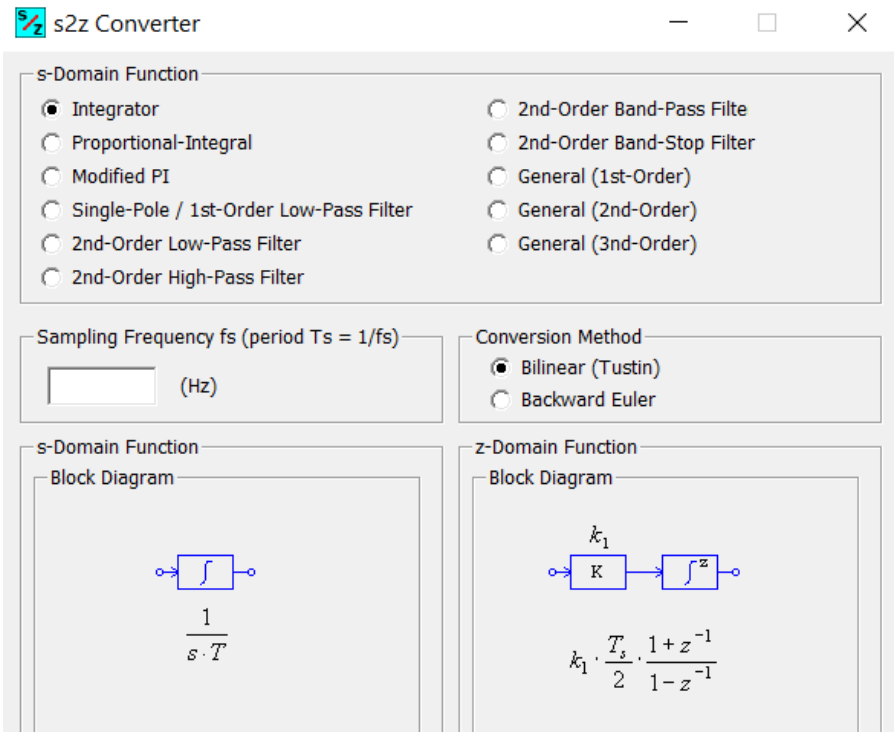


Figura 6. Convertidor control analógico a digital, *s2z Converter*

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.

En este ejemplo, se usa el método Backward Euler. Con la frecuencia de muestreo igual a la frecuencia de conmutación de 20 kHz según ejemplo de PSIM, se convierte el controlador PI analógico al controlador PI digital usando el programa de conversión, se obtienen los parámetros del controlador PI digital como: k1 = 0.4 para la acción proporcional y k2 = 1000 para la acción integral. El circuito con el controlador digital se puede observar a continuación:

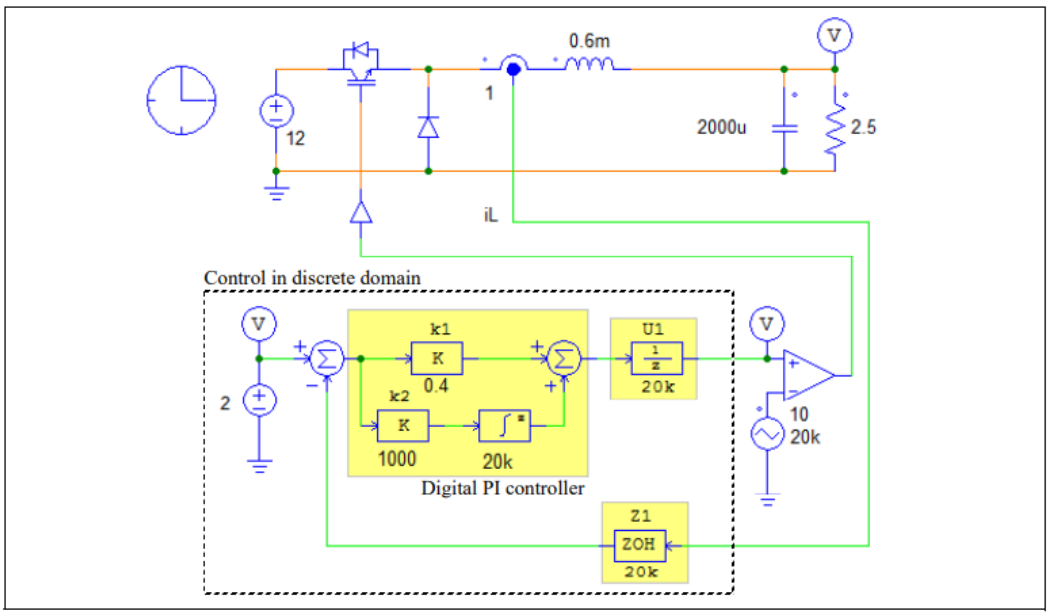


Figura 7. Sistema de convertidor CC en modo de control discreto.

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.

1.6 Generación de código para un hardware de destino.

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 ilustracion los elementos de hardware están resaltados en amarillo.

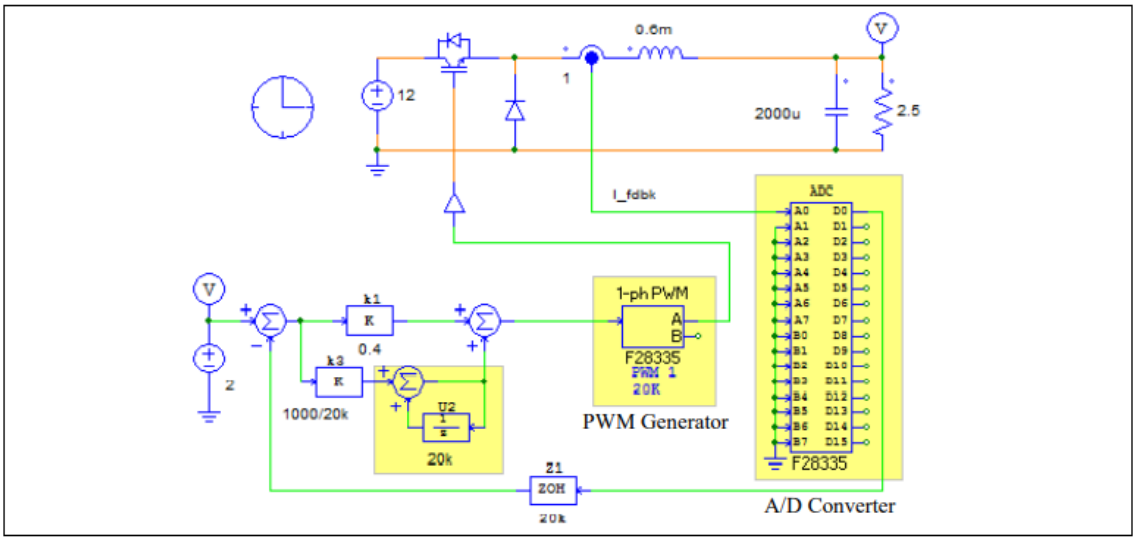


Figura 8. Convertidor CC con elementos de hardware F2833X.

En la imagen anterior 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.

1.7 Generar código.

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 está 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*, en el tutorial *Auto Code Generation for F2833X Target"* y en esta guía de laboratorio se amplía información sobre el proceso de programación y ejecución del programa en la DSP.

En el capítulo 5 de esta guía se realizará el modelo y controlador de un convertidor DC-DC *FlyBack* para dar a conocer con mayor entendimiento el uso de esta herramienta a través de una plataforma de *Hardware in the loop (HIL).*

2. PROCEDIMIENTO PARA LA IMPLEMENTACION DE UNA PLATAFORMA HIL.

Para desarrollar una plataforma de *Hardware in the loop* (*HIL)* aplicado a un convertidor de potencia y utilizando la herramienta SimCoder de PSIM se deben seguir los siguientes pasos:

1. Diseñar y simular el convertidor de potencia incluyendo las pérdidas si así se requiere.
2. Obtener el modelo matemático no lineal del convertidor de potencia.
3. A partir del modelo matemático realizar el modelo discreto en diagrama de bloques en PSIM, usando elementos de la librería compatibles con SimCoder.
4. Comparar la simulación del circuito del convertidor de potencia con la simulación del modelo discreto y ajustar la frecuencia de muestreo para dejarlo lo más similar posible al circuito simulado.
5. Escalizar la salida de voltaje o de corriente del inductor de acuerdo con la variable que se quiera controlar.
6. Tomar la señal escalizada y usar un conversor digital análogo PWM utilizando un comparador, un módulo de salidas digitales de SimCoder y un filtro pasa bajos RC.
7. Generar el código en PSIM usando un PWM incluido en el modelo, compilar y programar en la DSP en *code composer studio.*
8. Implementar el filtro pasa bajos RC físico y conectarlo a la salida del DSP para verificar la señal de voltaje entregada con el multímetro y el osciloscopio, comparar esta señal con los resultados de la simulación.
9. Incluir en el modelo discreto un módulo de entradas digitales de SimCoder después del generador PWM, de esta manera el PWM será una entrada digital a la DSP.
10. Simular, generar código y programar en otra DSP un generador PWM con Duty configurable y la frecuencia de diseño.
11. Realizar simulación HIL en lazo abierto interconectando la DSP que contiene el modelo con la DSP que genera la salida PWM.
12. Diseñar un controlador PI discreto para el convertidor de potencia,
13. Incluir elementos de hardware en el controlador, simular, generar código y programar en la DSP.
14. Realizar simulación HIL en lazo cerrado interconectando la DSP que contiene el modelo con la DSP que contiene el controlador, realizar perturbaciones en la planta (Vg, R), cambiar set-point en el controlador y comparar resultados.

Es muy importante seguir paso a paso los anteriores puntos y validar cada uno antes de seguir con el siguiente.

A continuación, se tomará de ejemplo un convertidor potencia para desarrollar cada uno de los pasos y al final obtener una implementación de una plataforma *hardware in the loop.*

2.1 Paso 1: Diseño del convertidor de potencia.

El convertidor diseñado para esta guía es un convertidor DC-DC elevador FlyBack, la metodología usada se puede encontrar en el libro *Fundamentals of Power Electronics* que se encuentra en las referencia bibliograficas.

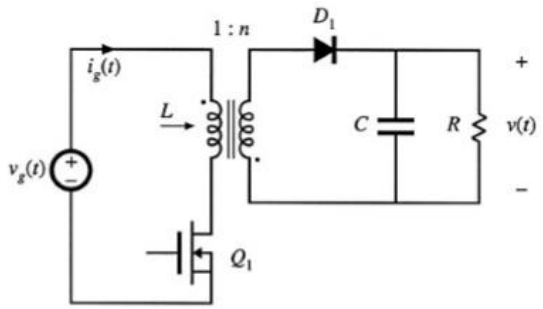


Figura 9. Convertidor elevador FlyBack.

Realizando el proceso de diseño aplicando la aproximación de pequeñas señales e implementando un script de Matlab para el cálculo del convertidor se obtiene:

- Duty = 0.5
- IL= 16.6667 A.
- ∆i= 0.8333 A.
- ∆v= 1.2500 V.
- R= 625 Ohm.
- n= 21.
- L= 45 uH
- C= 1 u F

Simulación con pérdidas en transistor e inductor.

- RL= 6.38 mΩ.
- Ron= 0.150 Ω.

A continuación, se muestra el diagrama de del convertidor FlyBack con los componentes de diseño y aplicando las perdidas las cuales se resaltan en amarillo.

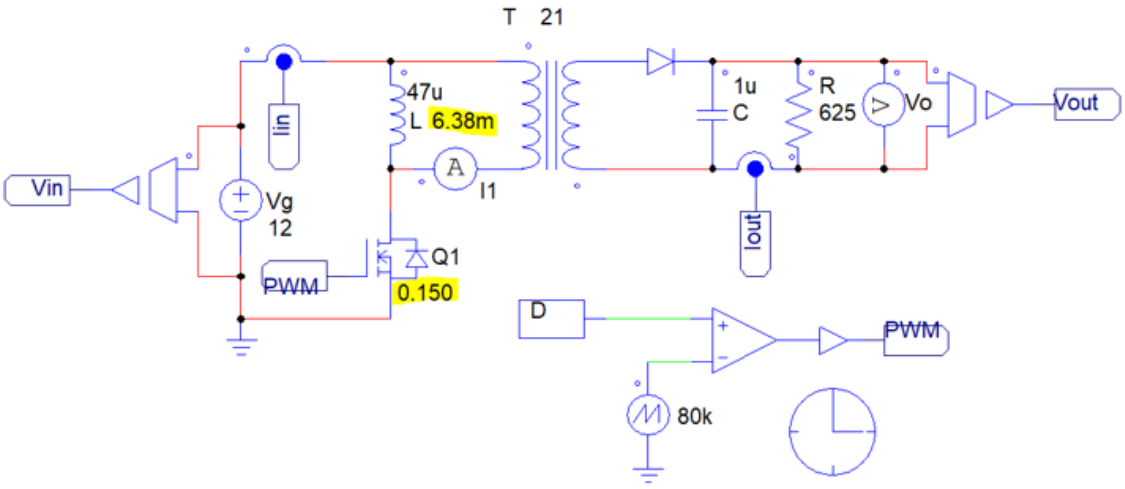
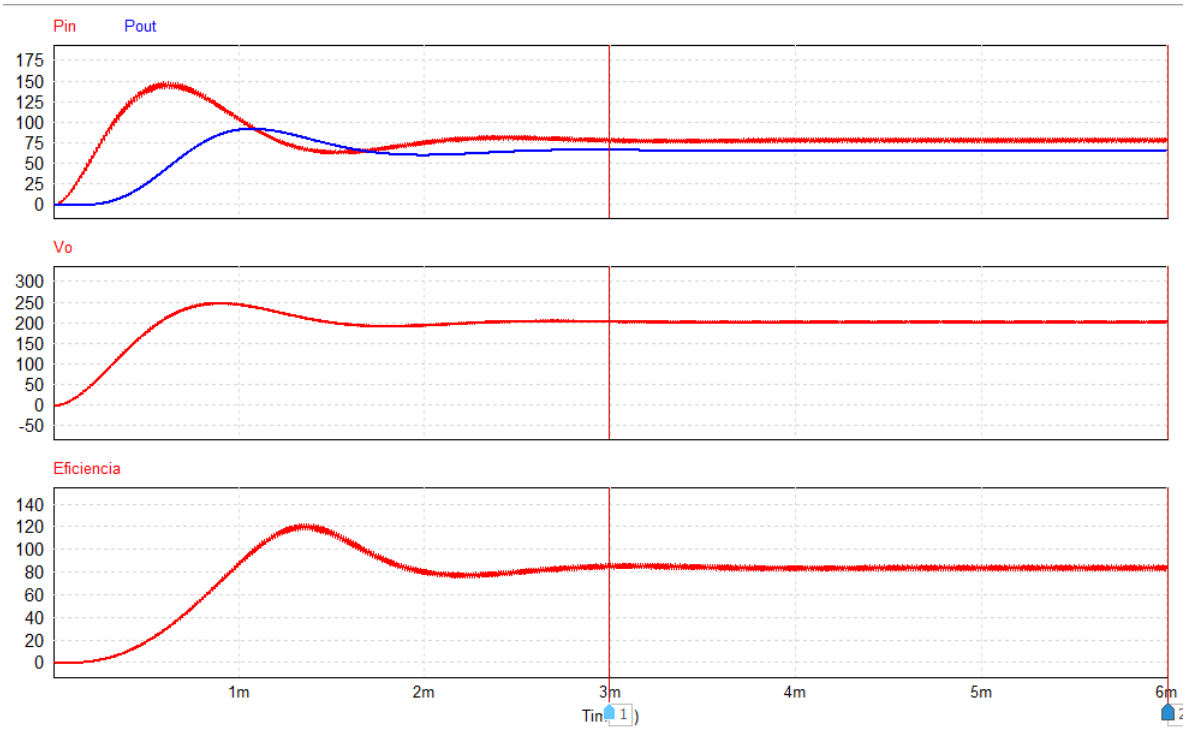


Figura 10. Simulación convertidor FlyBack con pérdidas.

Como resultado de la simulación se obtiene la siguiente respuesta:



	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 11. Resultados de la simulación del convertidor FlyBack.

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.

2.2 Paso 2: Modelo matemático no lineal.

Para realizar el modelo matemático no lineal, se deben realizar los balances de voltio segundo (1) y los balances de carga (3) a partir de la aproximación de pequeña señal realizada en el diseño.

Balance de voltio – segundo:

$$(1)\quad L\frac{di_L}{dt} = V_g * d - I * R_l - I * Ron * d - \frac{v}{n} * d'$$

$$(2)\quad \frac{di_L}{dt} = \frac{1}{L}\left(V_g * d - I * R_l - I * Ron * d - \frac{v}{n} * d'\right)$$

Balance de carga:

$$(3)\quad C\frac{dv}{dt} = -\frac{v}{R} * d + \frac{i_l}{n} * d' - \frac{v}{R} * d'$$

$$(4)\quad \frac{dv}{dt} = \frac{1}{C}\left(\frac{i_l}{n} * d' - \frac{v}{R}\right)$$

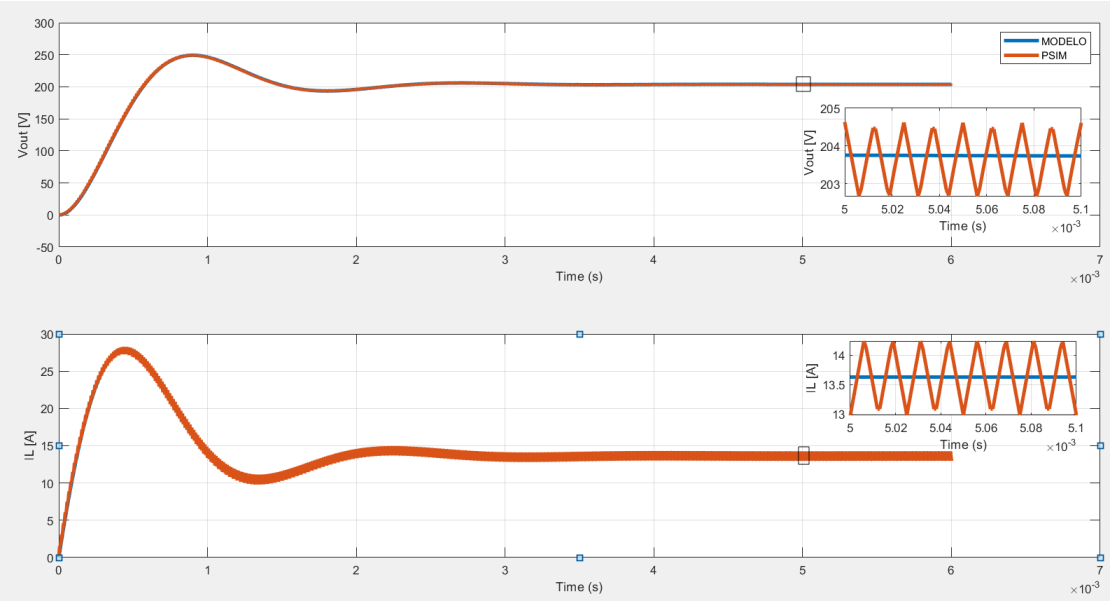


Figura 12. Modelo no lineal promediado vs simulación en PSIM.

2.3 Paso 3: Modelo discreto.

En esta etapa del proceso se debe pasar del modelo matemático a un modelo en diagrama de bloques discretizado, esto se debe hacer utilizando elementos de control digital y que sean compatibles con SimCoder para poder generar código y ser programado en la DSP.

Tener presente:

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

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

Se realiza el modelo discreto para el convertidos FlyBack en PSIM y se obtiene:

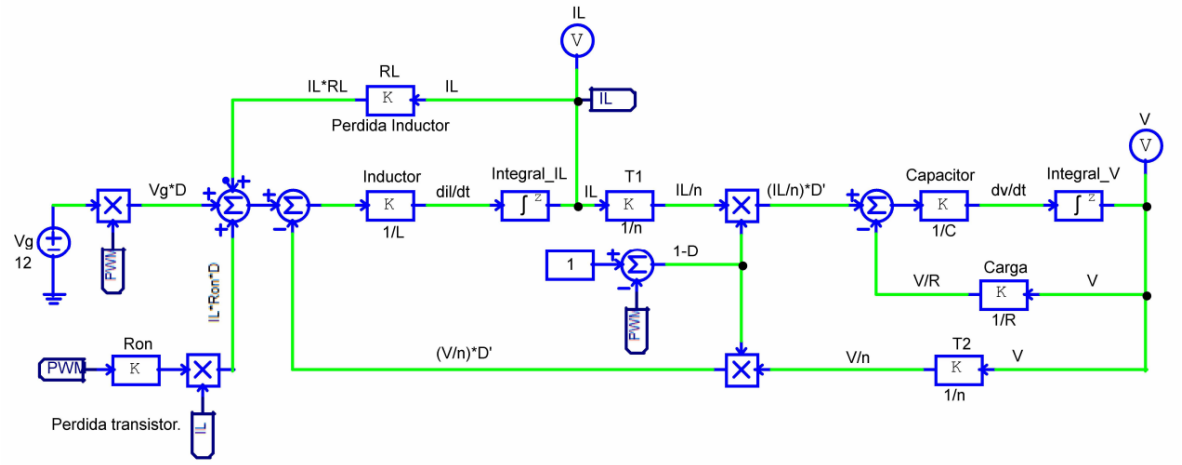


Figura 13. Modelo discreto convertidor Flyback con pérdidas en inductor y transistor.

Todos los elementos utilizados en el modelo discreto (Figura 13) son compatibles con SimCoder y se encuentran en la barra superior de PSIM en la opción *Elements*, a continuación, se explican cada uno de estos elementos:

- **Multiplicadores y divisores:** Se utilizan para hacer operaciones de multiplicar y dividir.

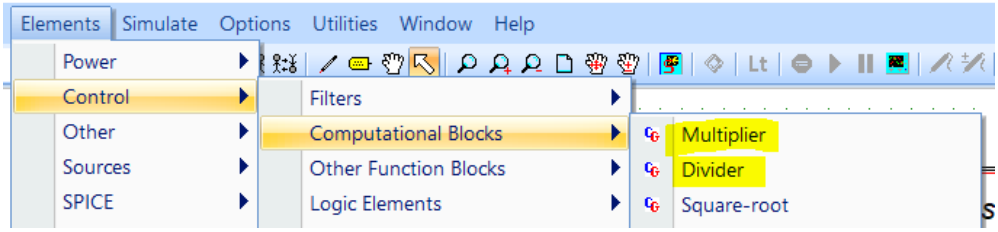


Figura 14. Multiplicador y divisor en PSIM.

Observe que estos elementos son compatibles con SimCoder.

- **Sumadores y restadores**: Sirven para hacer operaciones de suma o resta.

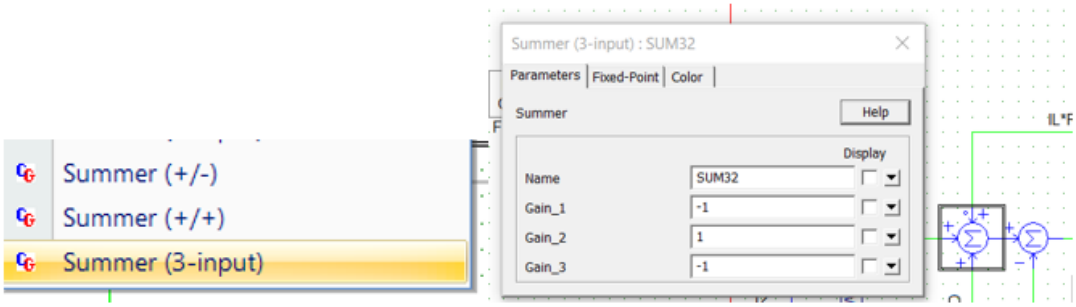


Figura 15. Sumador de 3 entradas

- **Proporcional:** La salida de este bloque entrega un resultado del producto de la entrada (V_{in}) por una ganancia dada (K).

(7) $V_{OUT} = K * V_{in}$

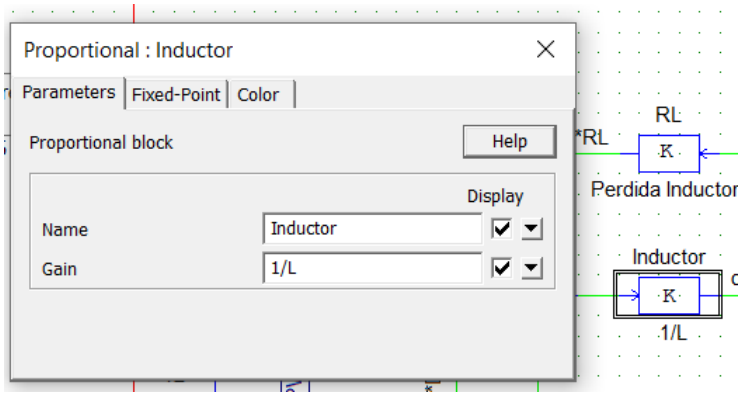


Figura 16. Proporcional.

- **Integrador digital:** Integra la variable que contenga su entrada, al ser digital es compatible con SimCoder, tiene tres posibles algoritmos de solución los cuales son: *trapezoidal rule (0), backward Euler (1) y forward Euler (2),* además es necesario definir una frecuencia de muestreo para realizar el cálculo que va a depender de cada convertidor. La frecuencia utilizada en ambos integradores es de 20MHz y el método de solución es *forward Euler,* esta configuración es la que da mejores resultados para el convertidor FlyBack.

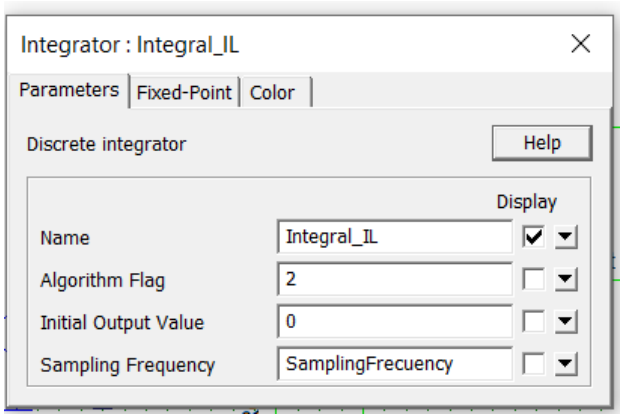


Figura 17. Integrador digital.

- **Archivo de parámetros *(Parameters file*):** Este elemento de la librería de PSIM es compatible con la generación de código y sirve para programar variables e incluso hacer algoritmos en lenguaje C para agregar al modelo o controlador que se esté diseñando, para este ejemplo se utilizó para asignar todas las variables como frecuencias, Duty, carga, inductancia, capacitancia, entre otros, lo que hace más fácil realizar cambios en el modelo.

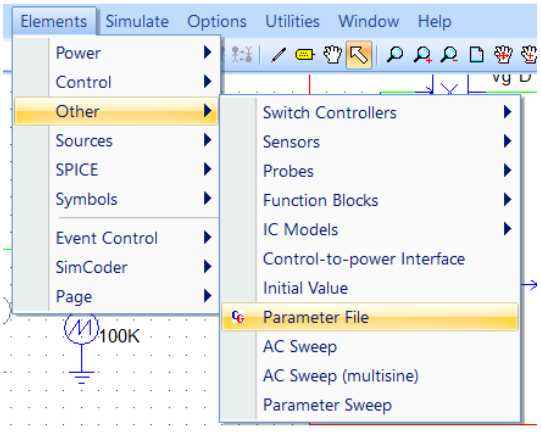


Figura 18. *Parameter File.*

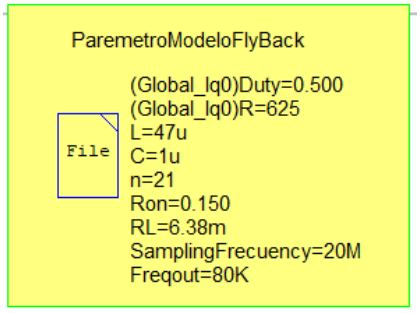


Figura 19. Parámetros del modelo discretizado en *parameter file*.

Variables gobales: Las variables globales (Global_Iq0), son variables que se pueden cambiar durante la ejecución del programa en la DSP, esto permite realizar cambios en tiempo real y poder así evaluar comportamientos y respuestas de control.

2.4 Paso 4: Comparar simulación:

En este paso se comparan 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:

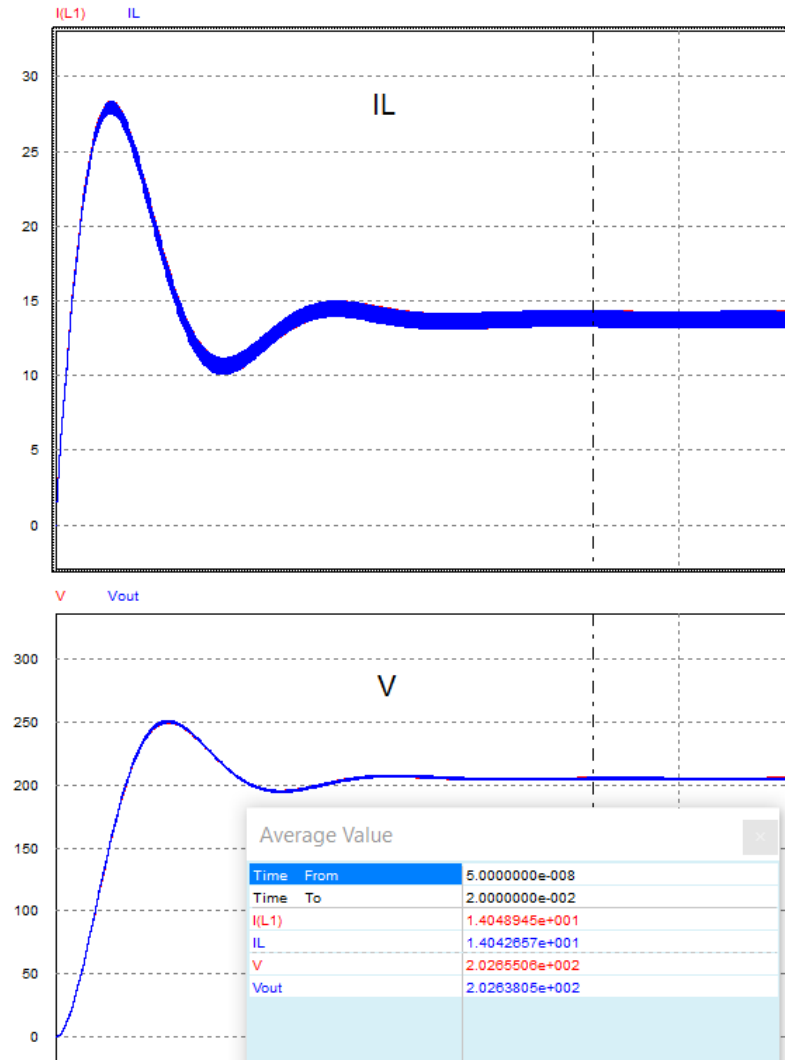


Figura 20. Comparación resultados entre circuito y modelo discreto.

Cuando la comparación no sea la esperada, se debe modificar la frecuencia de muestreo por lo general aumentándola y verificar cuál de los tres algoritmos de solución que usan los integradores da mejor respuesta.

Lo que se tiene hasta este punto se puede representar en la siguiente Figura:

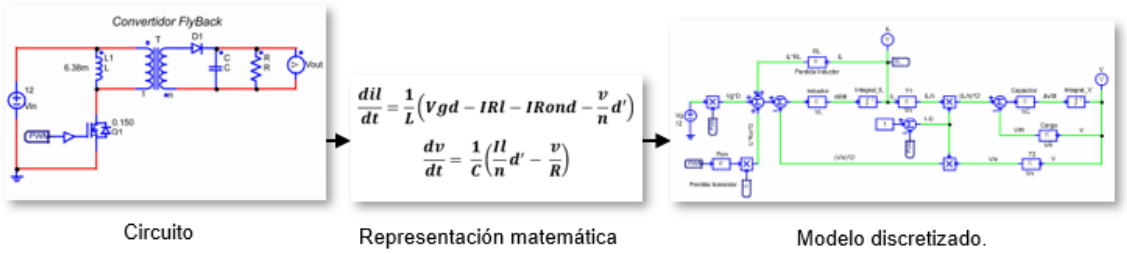


Figura 21. Modelado discreto de un convertidor.

2.5 Paso 5: Escalización.

La escalización consiste en reducir el valor de la variable analógica que se requiere controlar, la cual puede ser corriente del inductor (i_l) o el voltaje de salida (v), existen diferentes maneras de hacerlo, pero lo más importante es tener presente el voltaje máximo con el que trabaja la DSP, el cual es de 3,3 V. Para el ejemplo se escaliza la salida de voltaje y para encontrar el valor de escalización se realiza la siguiente ecuación:

$$(8)\ Escal = \frac{V_{maxDSP}}{V_{maxconvert}} = \frac{3.3V}{330V} = 0.01$$

Según la ecuación anterior (8) el valor de voltaje de salida del modelo que se multiplique por 0.01 será el valor escalizado, ejemplo:

$$(9)\ Vscal = V * 0.01 = 205V * 0.01 = 2.05V$$

2.05 V equivalen a 205V, en el modelo basta con colocar un bloque proporcional en la salida de voltaje para realizar la escalización, esta señal será la que reciba el convertidor digital análogo para generar la salida de voltaje en la DSP:

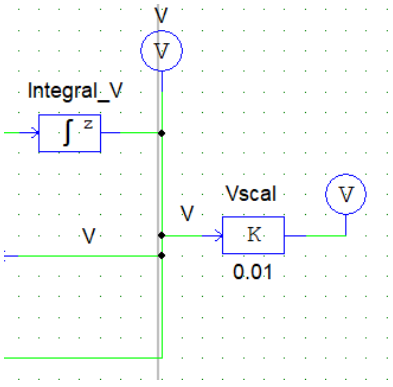


Figura 22. Escalización de señal de control en PSIM.

2.6 Paso 6: Conversor digital análogo (DAC).

En los tutoriales de PSIM y SimCoder se encuentran ejemplos para utilizar un convertidor digital análogo por comunicación *SPI* (*Serial Peripheral Interface*), pero estos convertidores

son de difícil consecución local 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 con una frecuencia determinada, luego llevarla a una salida digital del DSP y por último diseñar un filtro pasa bajos pasivo de primer orden RC que será el que reciba la señal PWM para convertirla en una señal analógica.

Nota: La señal PWM de salida queda programada en la DSP y el filtro RC es físico.

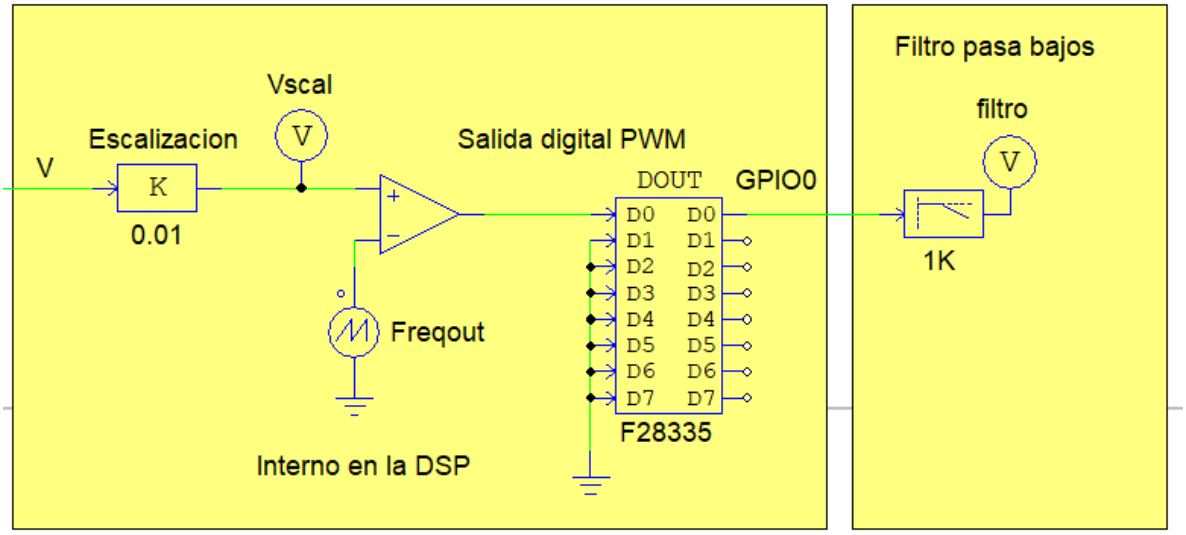


Figura 23. Salida de voltaje PWM y Filtro pasa bajos en PSIM

Filtro pasa bajos:

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).

El diseño del filtro se hace por tanteo empezando con una frecuencia de corte alta y se ajusta hasta encontrar un equilibrio entre un bajo rizado y buena precisión, la señal filtrada debe seguir la señal escalizada. Cuando se encuentre la frecuencia de corte ideal se recomienda la siguiente página para configurar el condensador y la resistencia: *https://wilaebaelectronica.blogspot.com/2017/01/filtro-pasa-bajos-pasivo-de-1er-orden-rc.html*

Para el caso del convertidor FlyBack, la frecuencia de corte es de 1Khz y el filtro RC queda configurado de la siguiente manera.

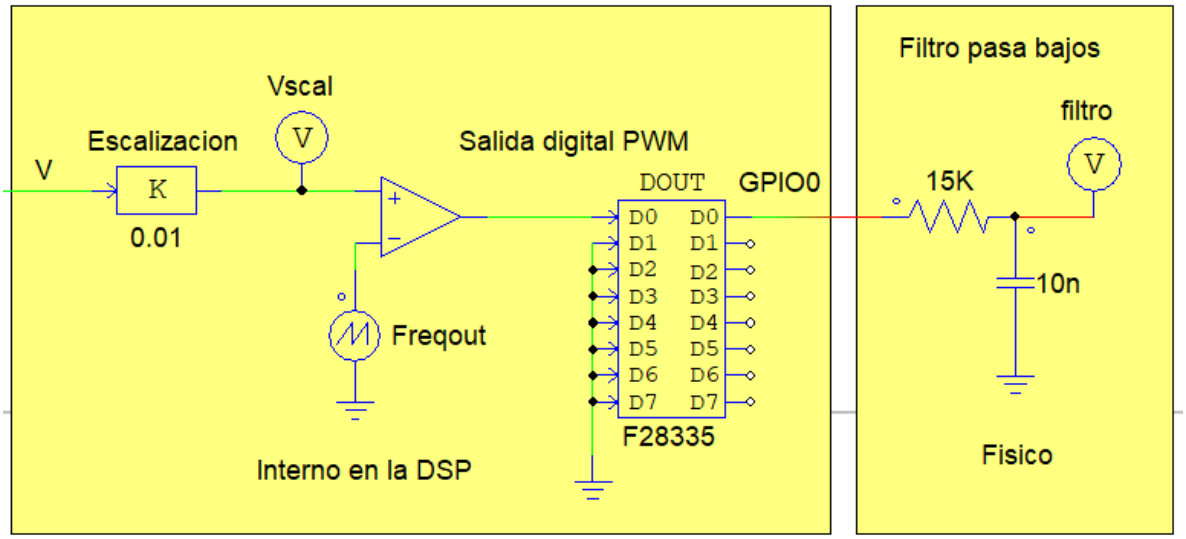


Figura 24. Salida de voltaje PWM y Filtro pasa bajos.

Salida digital:

La salida digital se encuentra en la librería de SimCoder como *digital output,* este elemento se pone después del PWM de salida, en este módulo se pueden seleccionar hasta 8 puertos GPIO de la DSP como salidas, para el caso de estudio se selecciona el GPIO0.

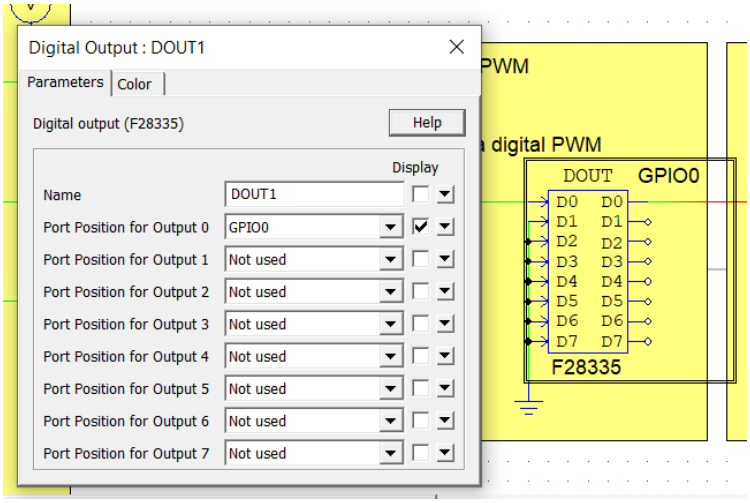


Figura 25. Salida Digital de SimCoder.

Configuración de hardware:

Es importante seleccionar el puerto GPIO0 como salida digital en el bloque de configuración de hardware del DSP que se encuentra en la librería de SimCoder como *hardware configuration.*

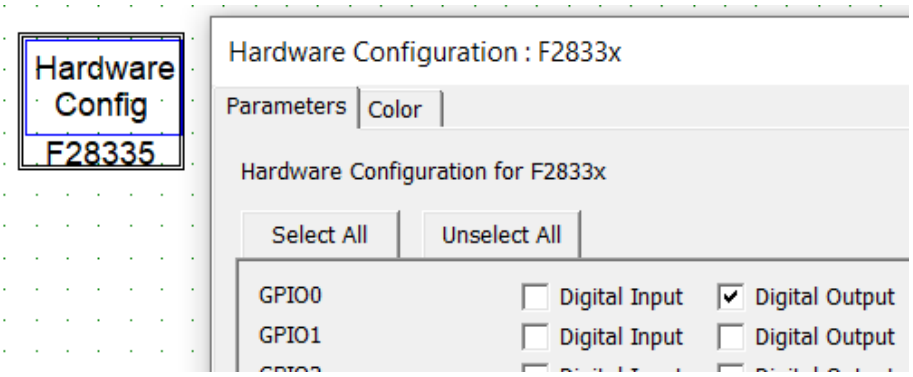


Figura 26. *Hardware configuration.*

Con este bloque se configuran todos los puertos GPIO del DSP, pudiendo seleccionar si el pin va hacer una salida digital, entrada digital o una salida PWM.

Nota: Cuando se incluyen elementos de hardware en la simulación puede aparecer el siguiente error:

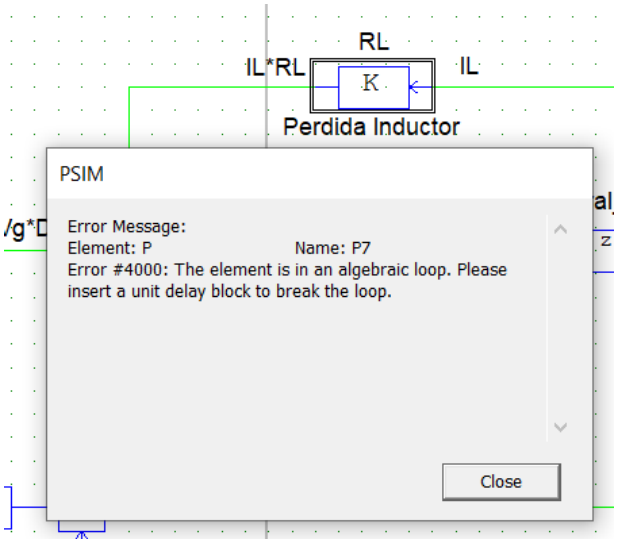


Figura 27. Error #4000 *unit delay*

Para solucionarlo, se deben colar retardadores de tiempo (*unit delay)* para los cálculos matemáticos antes de sumadores, multiplicadores y divisores, estas unidades de tiempo se encuentran en los elementos de control digital de la librería como *unit delay,* la frecuencia de muestreo debe ser igual a la puesta en los integradores digitales.

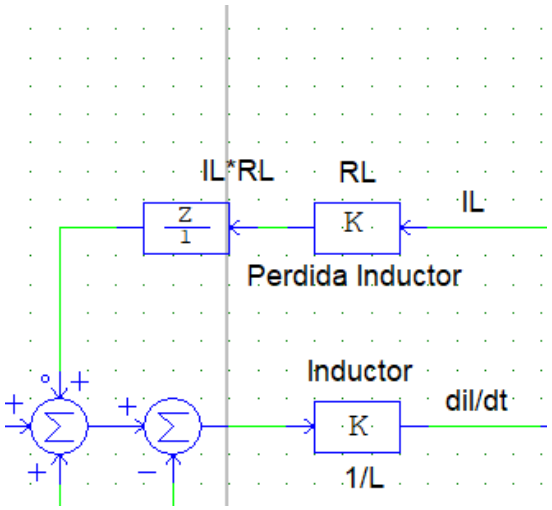


Figura 28. Aplicación de retardo *unit delay.*

Luego de eliminar el error se puede comparar la señal escalizada con la señal filtrada:

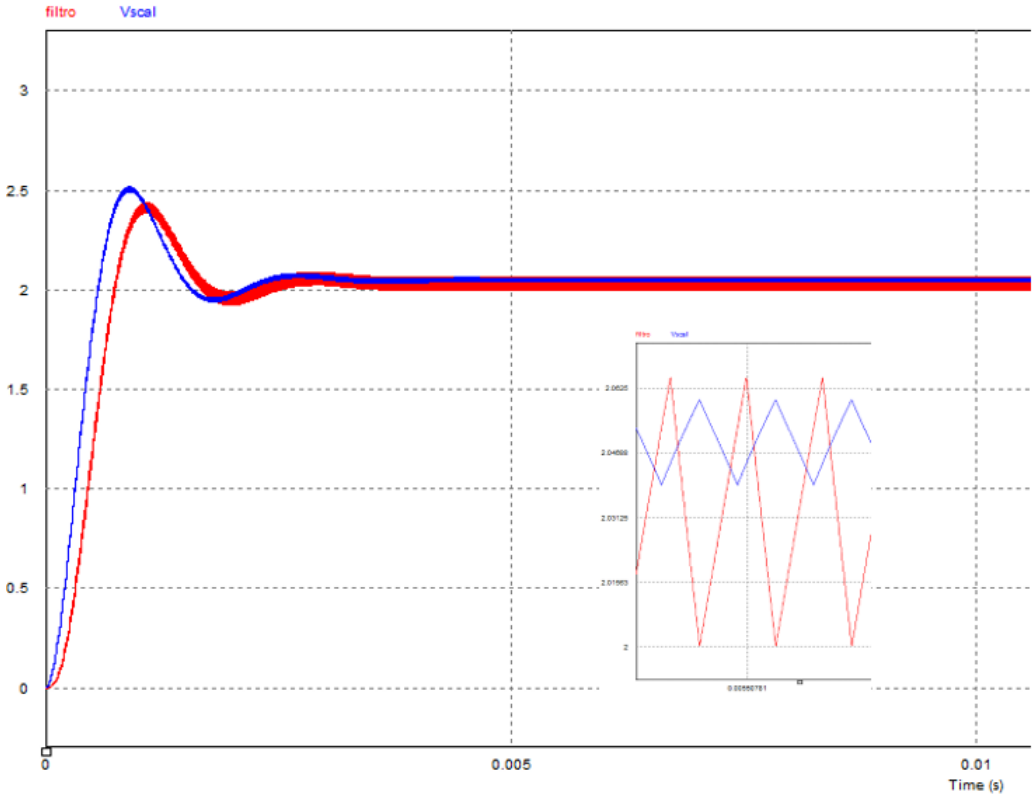


Figura 29. Señal escalizada vs señal analógica filtrada.

Es muy importante comprobar que la señal filtrada tenga la misma frecuencia de la señal escalizada, para el PWM de salida se recomienda colocar la misma frecuencia del PWM de entrada.

2.7 Paso 7: Generación y programación de código.

Hasta el paso #6 se debe tener elaborado el modelo con su respectiva salida PWM escalizada como se muestra a continuación:

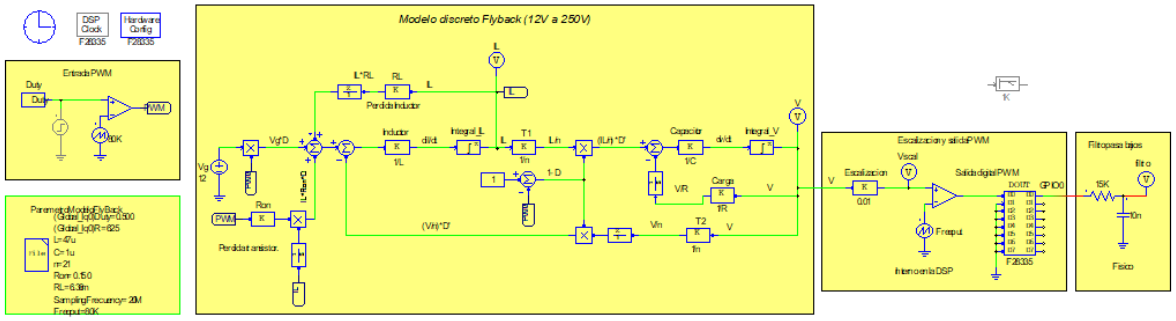


Figura 30. Modelo discreto con salida PWM.

Luego de tener lo anterior correctamente simulado, se procede con la generacion de codigo y posterior programacion en la DSP usando *Code Composer Studio*. El codigo generado desde PSIM contendra el PWM de entrada, el modelo discreto y el PWM de salida por el pin digital configurado.

Generación de código en PSIM.

Para generar código en PSIM se siguen los siguientes pasos:

1. Se configura en el control de simulación (*Simulation control*) el hardware de destino que se va a programar, en este caso la DSP F28335, esto se realiza desde la pestaña SimCoder.

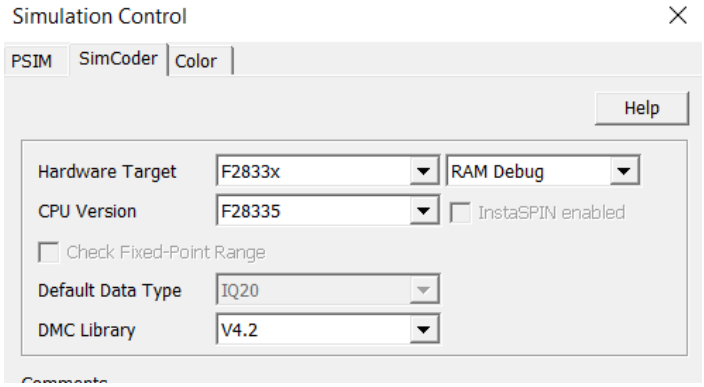


Figura 31. configuración de hardware de destino.

2. Se selecciona en la barra superior *Simulate – Generate code.*

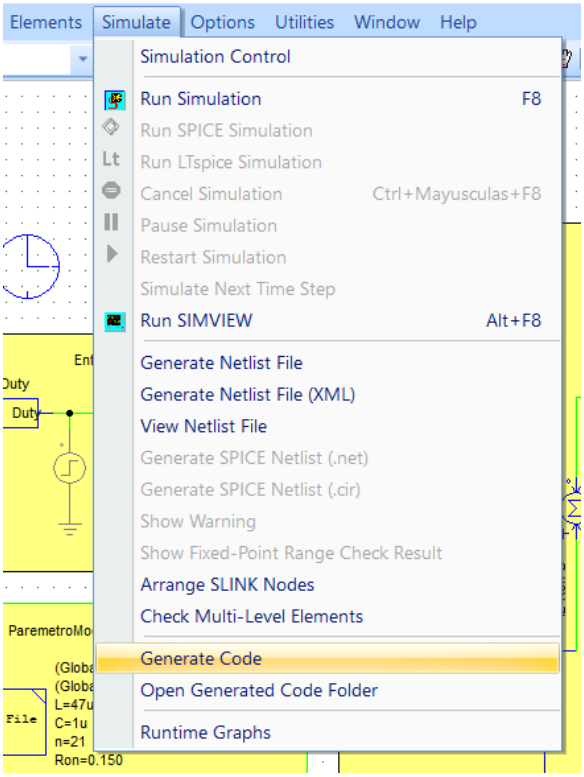


Figura 32. Generar código

3. A continuación, se abre una ventana con el código generado.

```
/*****************************************************************************
// This code is created by SimCoder Version 11.1.5.1 for F2833x Hardware Target
//
// SimCoder is copyright by Powersim Inc., 2009-2018
//
// Date: July 21, 2021 22:22:07
*****************************************************************************/
#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 l



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;
```

Figura 33. Ventana con código generado.

El Código generado se guarda de manera automática en una subcarpeta dentro de la carpeta principal donde se tiene guardado el archivo de PSIM.

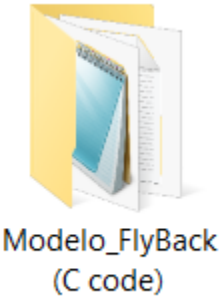


Figura 34. Carpeta con código generado.

Si le es difícil encontrar la ruta donde quedo la carpeta con el Código C, desde PSIM puede seleccionar, *Simulate – Open Generated Code Folder* y automáticamente muestra la ruta donde quedo guardado el código.

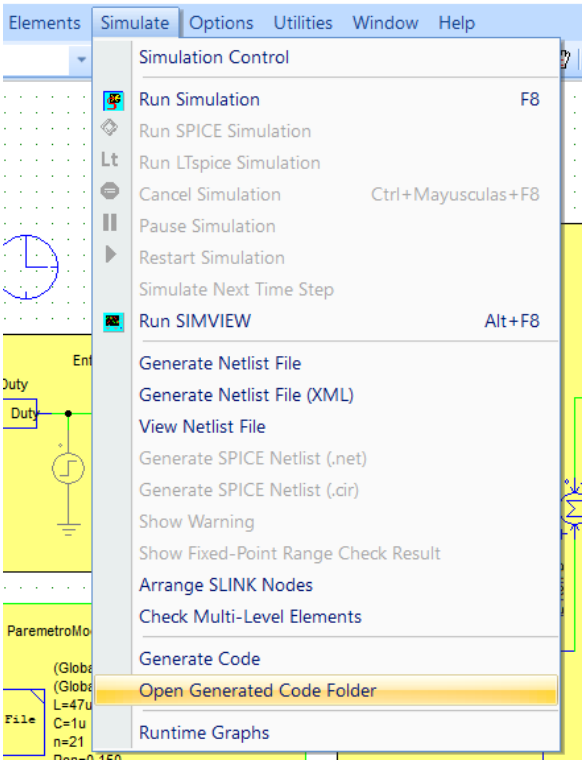


Figura 35. Encontrar carpeta con código generado.

Programación, lectura y escritura de variables de la DSP con *Code Composer Studio (CCS).*

Code Composer Studio (CCS) es entorno de desarrollo integrado para programar aplicaciones para procesadores integrados de Texas Instruments, para el caso de estudio se utiliza para programar el código generado desde PSIM en la tarjeta DSP F28335.

Code Composer Studio (CCS) se puede descargar gratis desde la página web de Texas Instrument.

Para la compilación y carga del programa en la DSP se siguen los siguientes pasos:

1. Ejecutar *Code Composer Studio (CCS).*
2. Seleccionar la carpeta donde se generó el código de PSIM dando clic en *Browse*.

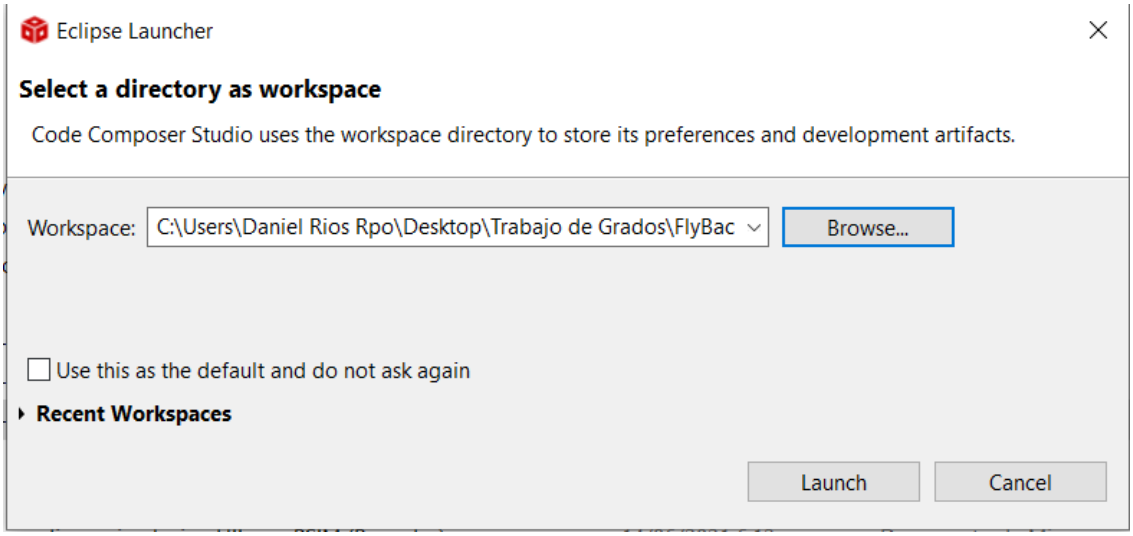


Figura 36. Directorio del programa.

3. Dar clic en *Launch* y esperar a que cargue la aplicación.
4. Seleccionar *Project, Import legancy CCSc3.3 Projects*.

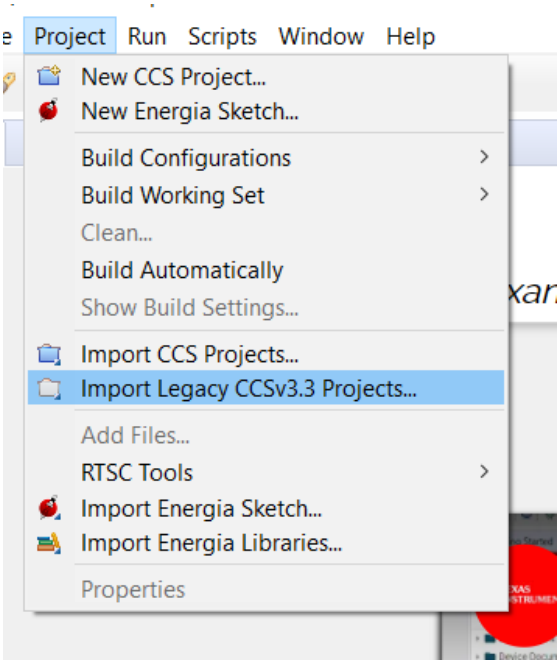


Figura 37. Importar código.

5. Se abre una ventada donde se debe buscar el código C (Archivo PJT) generado por PSIM dando clic en *browse.*

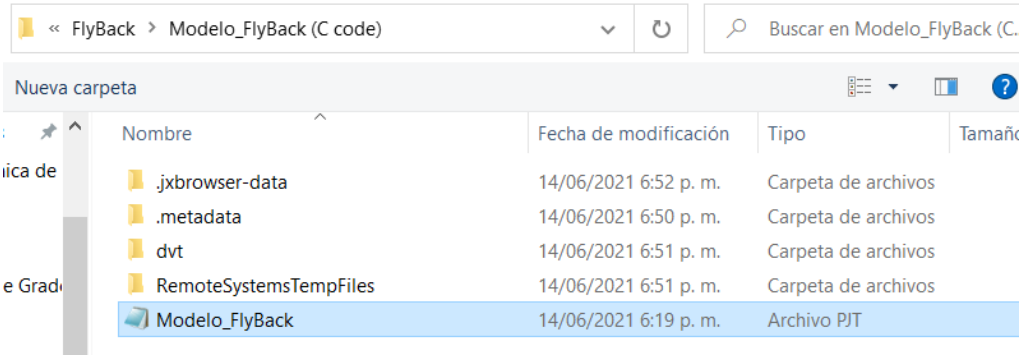


Figura 38. Selección del archivo PJT.

6. Posteriormente se da clic en siguiente *(next)* y aparece una ventana donde se debe dar clic en finalizar *(Finish*).

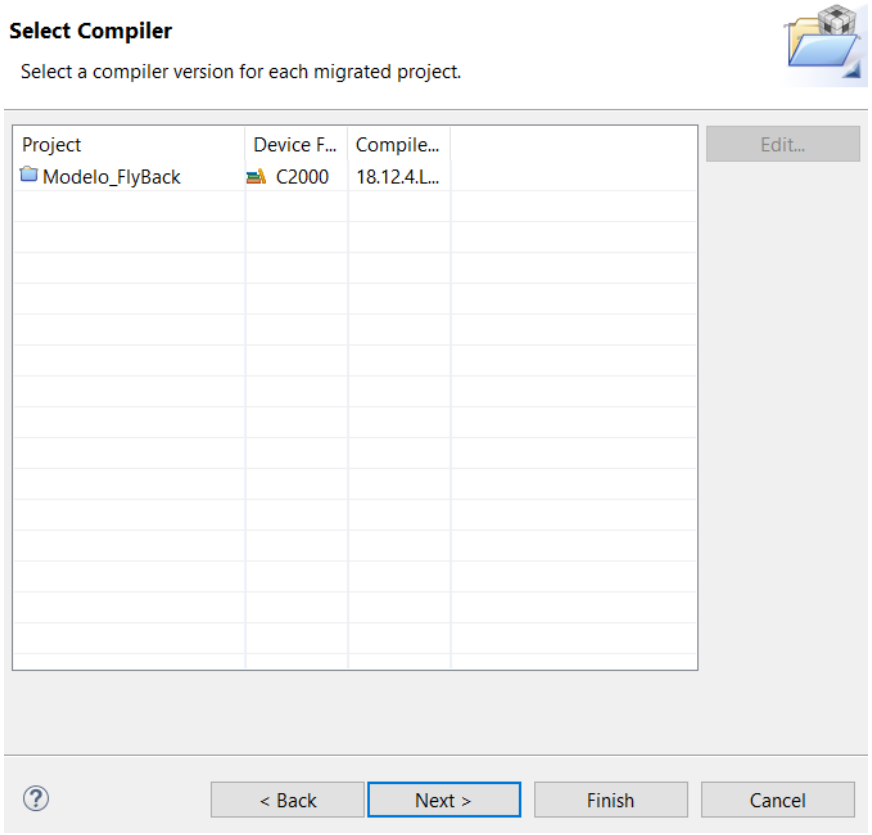


Figura 39. Selección de compilación de código.

7. Aparece un mensaje de advertencia, se da OK.

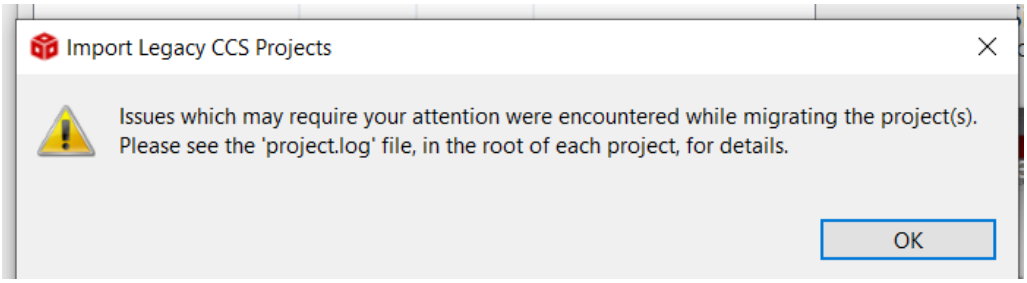


Figura 40. Mensaje de advertencia de compilación.

8. En la ventana principal al lado izquierdo y en negrita aparece el programa cargado en *CCS*.

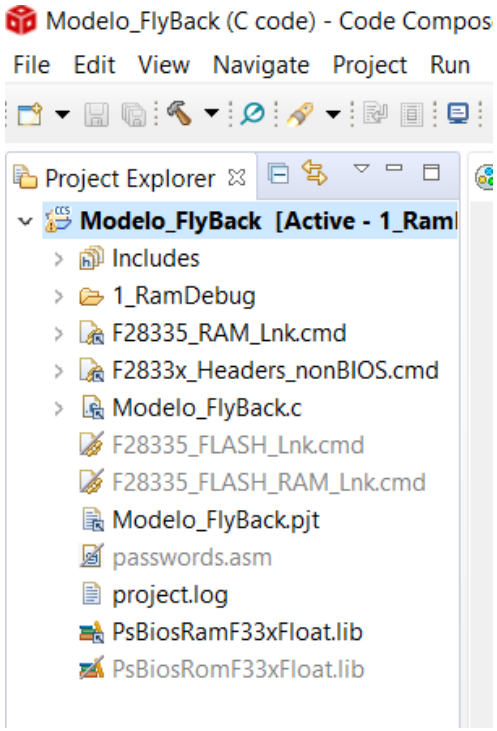


Figura 41. Programa cargado en CCS.

9. El siguiente paso es compilar el programa dando clic derecho en el nombre principal (resaltado en negrita) y luego en *Build Project*.

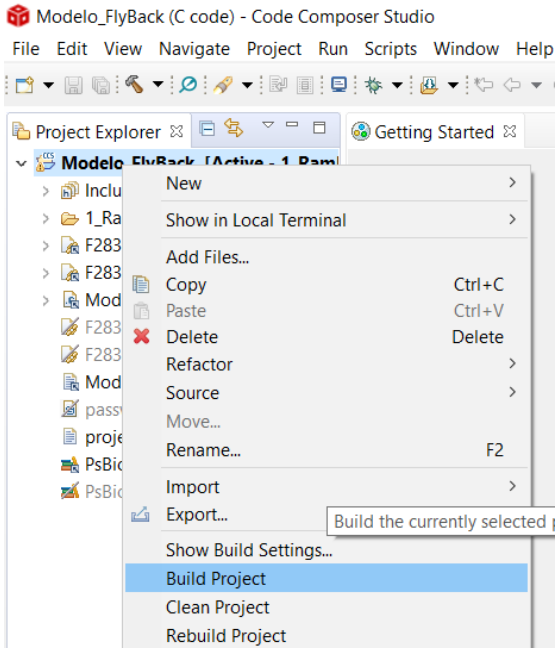


Figura 42. Compilación del programa.

10. Sale el siguiente error:

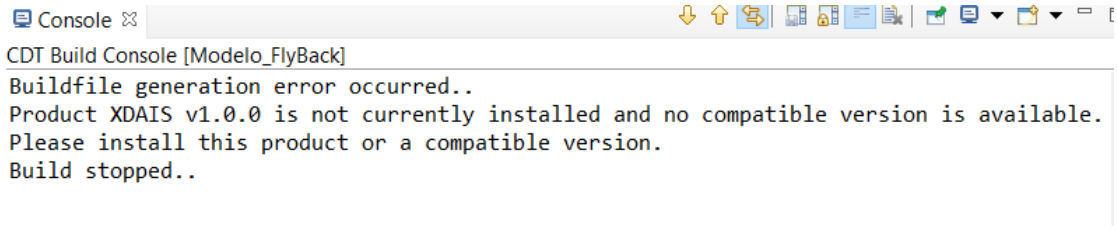


Figura 43. Mensaje de error de compilación.

Para solucionarlo se da clic en *File* y luego en *properties*.

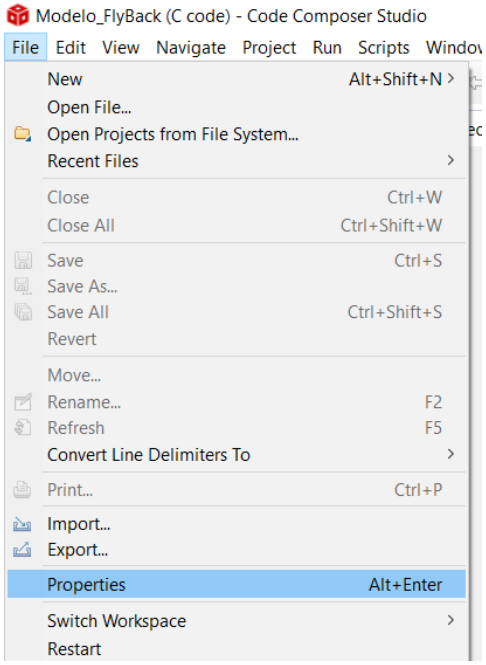


Figura 44. Acceso a propiedades.

11. A continuación, seleccionar *General, products* y se retira el chulo de *XDAIS*, luego se da clic en aplicar los cambios.

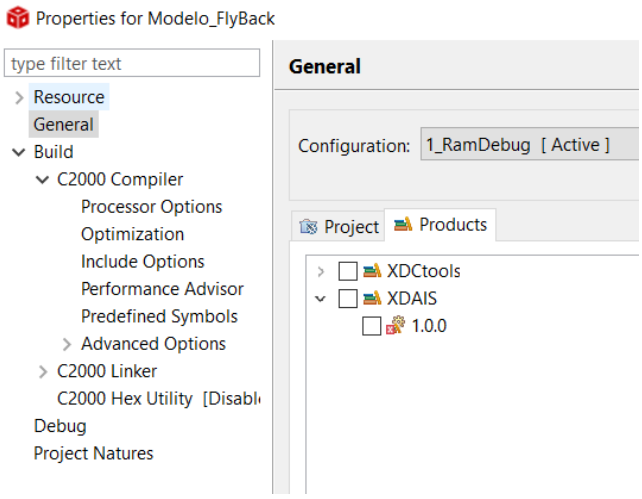


Figura 45. Ventana general de propiedades.

12. Se repite el punto 9 y se debe corregir el error.
13. El siguiente paso es configurar la tarjeta DSPF28335, para esto lo primero es conectar la DSP al computador:

Figura 46. Conexión de la DSP al computador.

14. Seleccione *View, Target Configurations*

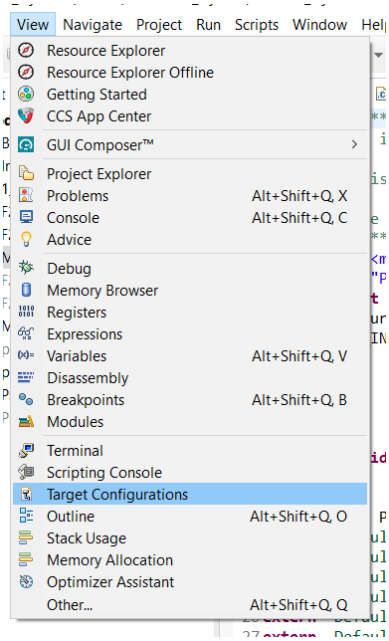


Figura 47. Ver configuración de tarjeta.

15. Hacer clic derecho en *user defined* y luego seleccionar *new target configuration*:

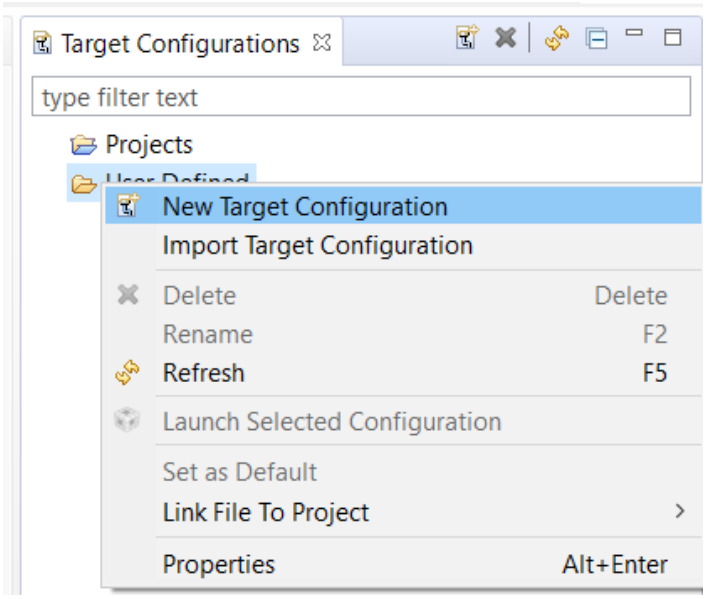


Figura 48. Nueva configuración de tarjeta.

16. Cambie el nombre del archivo como desee, en este ejemplo, se llama "F2833x" y haga clic en *finish*. La extensión del archivo será "ccxml"

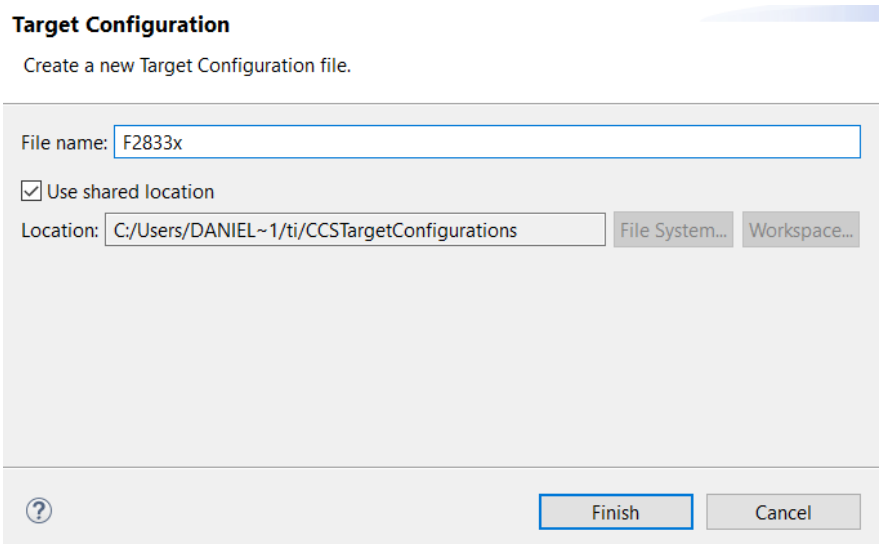


Figura 49. Nombre de configuración de tarjeta.

17. En *connection* busque la tarjeta interface *"Texas Instruments XDS100v2 USB Debug Probe",* luego selecciona la tarjeta TMS320F28335 y por último dar clic en guardar *(Save).*

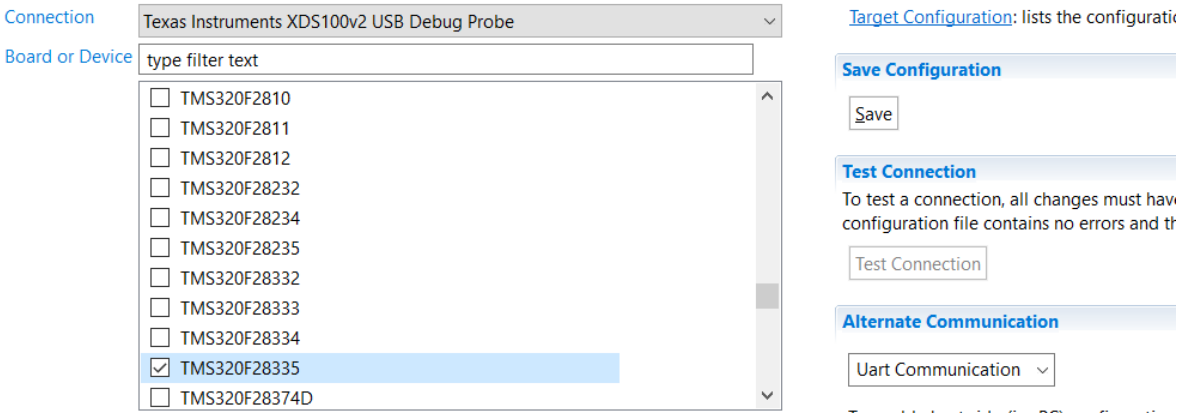


Figura 50. Selección de conexión y DSP.

18. En la ventana de configuración de tarjeta se da clic derecho en la nueva configuración y luego en "*Link File To Project*" y se selecciona el modelo desarrollado.

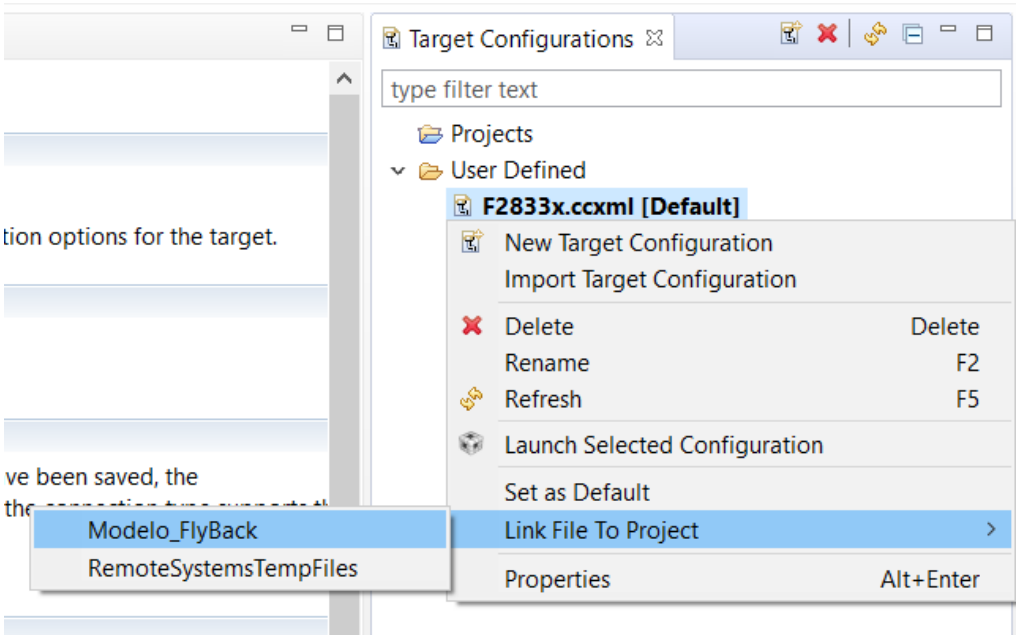


Figura 51. Selección de código en la configuración de tarjeta.

19. Para programar la DSP se selecciona el programa principal (Negrita), luego se da clic en *Run* y posteriormente en *Debug*.

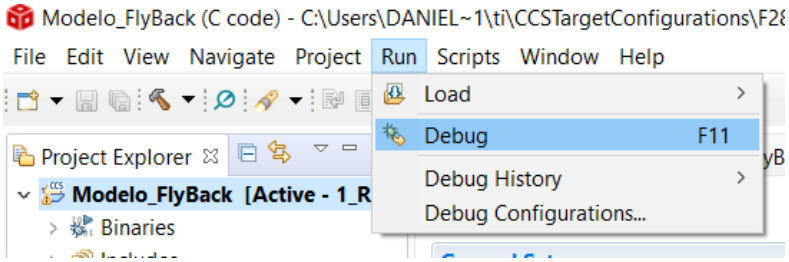


Figura 52. Programar la DSP.

20. El programa se cargará correctamente si aparece la siguiente ventana.

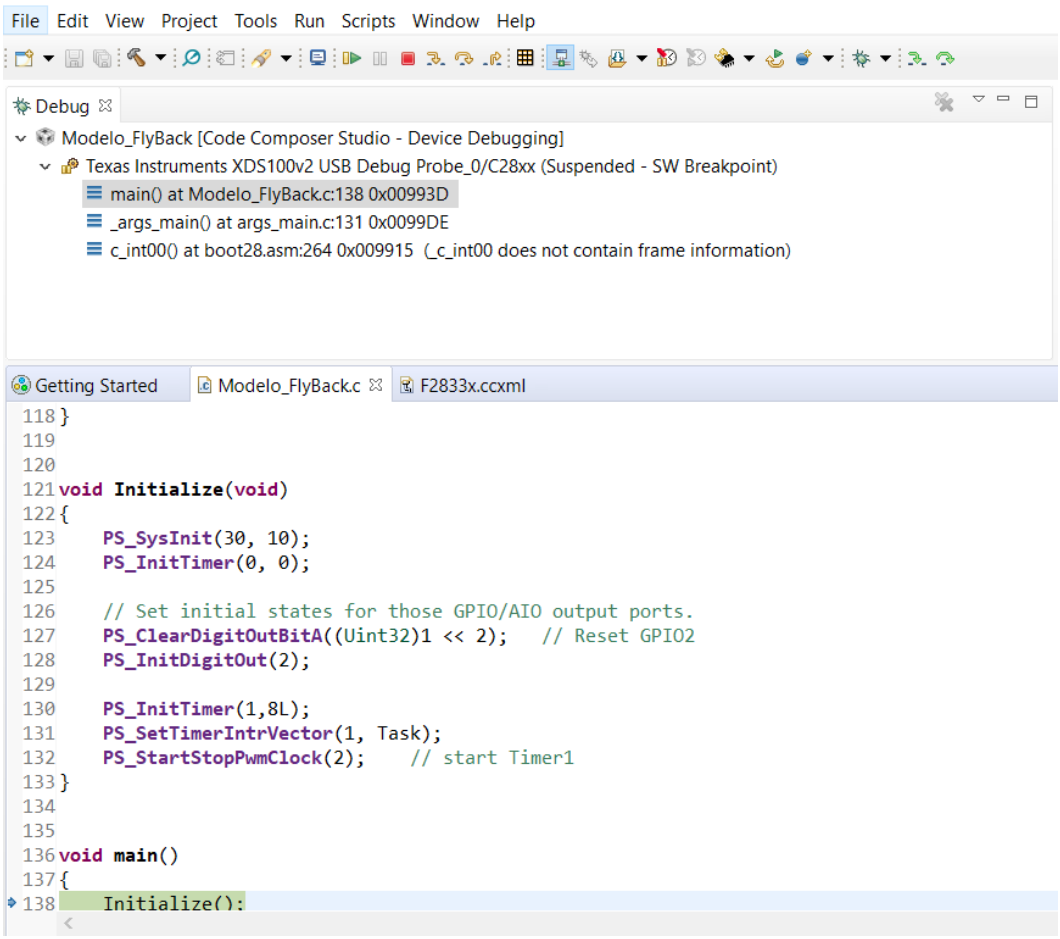


Figura 53. Ventana de depuración de programa.

21. Para ejecutarlo se da clic en *resume* y para detenerlo en *terminate.*

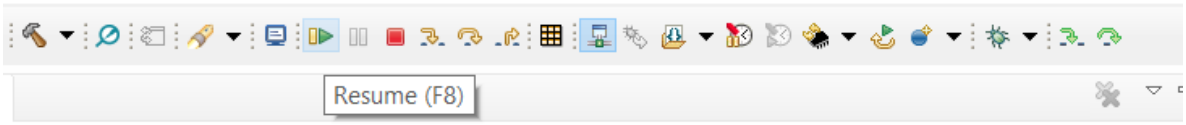


Figura 54. Barra de tareas de ventana de depuración.

22. Lectura y escritura de variables en tiempo real.

Para cambiar y leer variables en tiempo real se debe recordar que desde PSIM se generaron variables de este tipo en un archivo *File.* Por ejemplo, *(global)Duty=0.5* (Paso 3). Estas variables aparecen en el código C como *DefaultType.* Para agregarlas a la simulación se siguen los siguientes pasos:

a) Depurar el programa (Punto 19) y antes de iniciar buscar y seleccionar la variable en el código, dar clic derecho, luego clic en *add watch expresión* y por último clic en ok.

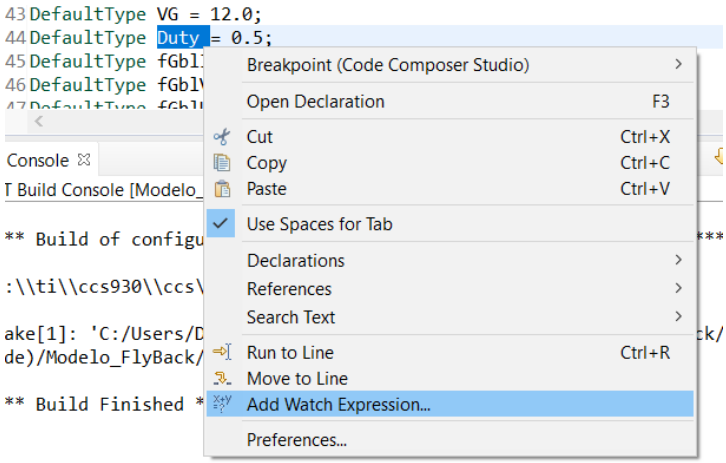


Figura 55. Adicionar variables.

b) La variable aparece en el cuadro superior derecho en el campo *expressions,* estas pueden ser de manipulación o solo visualización.

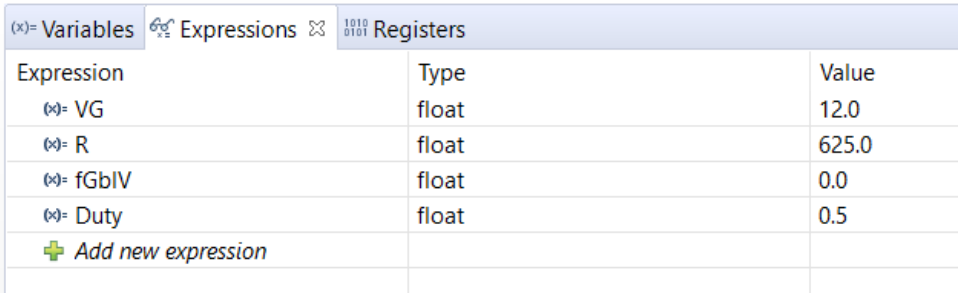


Figura 56. Cuadro de variables en CCS.

c) En la barra superior derecha seleccionar *Continuous Refresh* para que al momento de estar corriendo el programa en la DSP las variables cambien de manera inmediata.

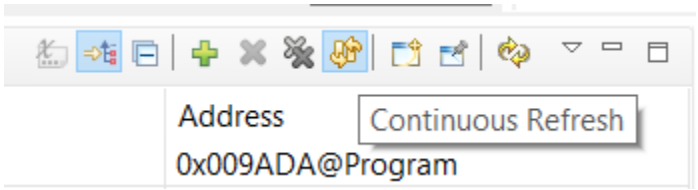


Figura 57. Selección de refrescamiento continuo.

d) Ejecutar el programa y realizar los cambios en las variables para ver los comportamientos.

2.8 Paso 8: Implementar filtro pasa bajos RC físico y verificar salida de la DSP.

Se implementa en una protoboard el filtro RC diseñado:

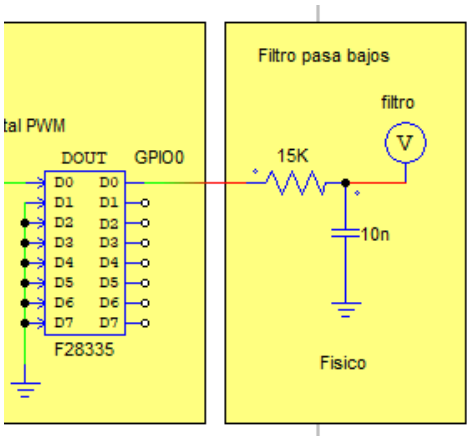


Figura 58. Filtro pasa bajos RC.

Se verifica la señal de voltaje con el multímetro y el osciloscopio, tener mucha precaución al hacer este procedimiento ya que las DSP son hardware muy delicados.

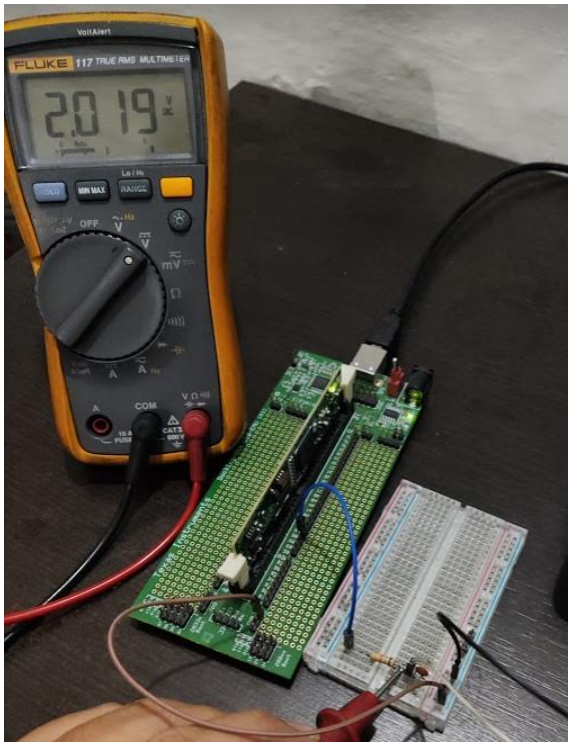


Figura 59. Salida de voltaje con duty=0.5, PWM interno.

En la Figura 59. Se puede apreciar la lectura de voltaje promedio de la salida del filtro RC aplicando un Duty de 0.5. realizando la escalización este valor equivale a 201 V lo cual es muy aproximado al resultado de la simulación en PSIM.

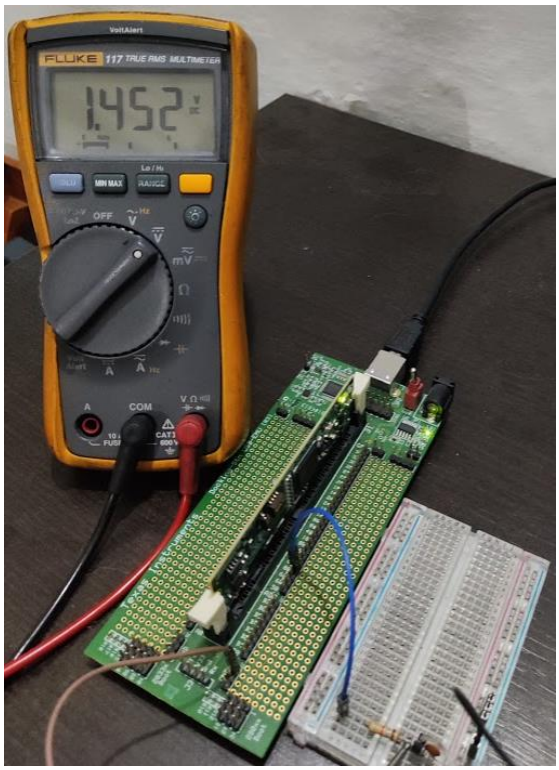


Figura 60. Salida de voltaje con duty=0.4, PWM interno

Realizando un cambio en el Duty a 0.4, se observa que la salida analógica cambia, con esto se comprueba que el modelo programado en la DSP funciona correctamente de acuerdo con lo diseñado. A continuación, se muestra la lectura de la señal análoga en el osciloscopio:

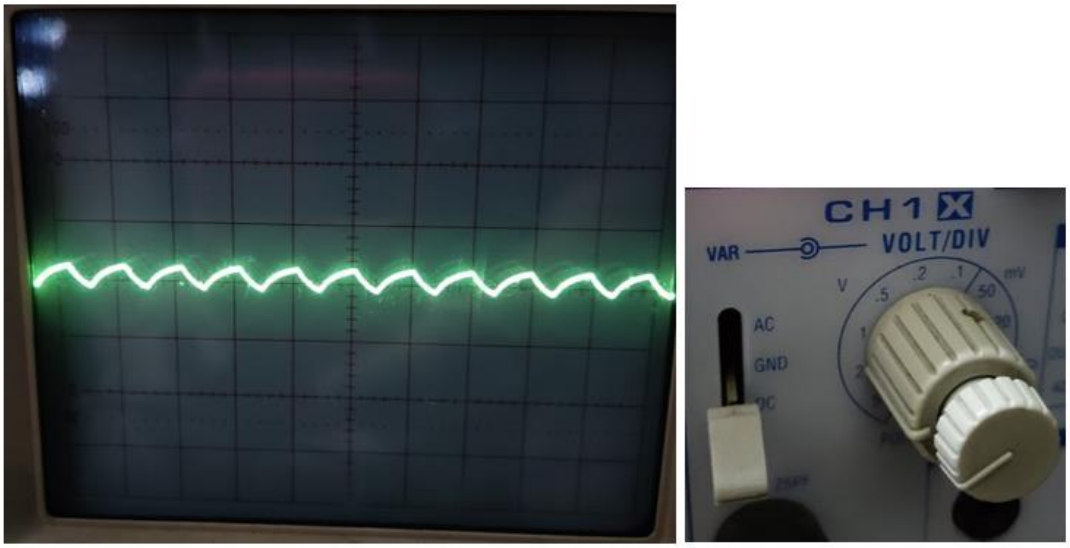


Figura 61. Señal análoga en osciloscopio.

Se realiza una comparación de los resultados y se obtiene:

Tabla 1. Comparación de resultados con PWM interno.

Duty	Voltaje promedio simulado	Voltaje promedio DSP	error relativo
0,5	2,038	2,019	-1%
0,4	1,485	1,452	-2%
0,3	1	0,987	-1%
0,6	2,646	2,604	-2%

Como se puede ver en la Tabla 1. El error de los valores de voltaje promedio entre la simulación y el voltaje real es muy pequeño.

2.9 Paso 9: Entrada PWM externa.

La entrada PWM se configura poniendo entre el modelo y el generador PWM un módulo de entradas digitales que se encuentra en la librería de *SimCoder:*

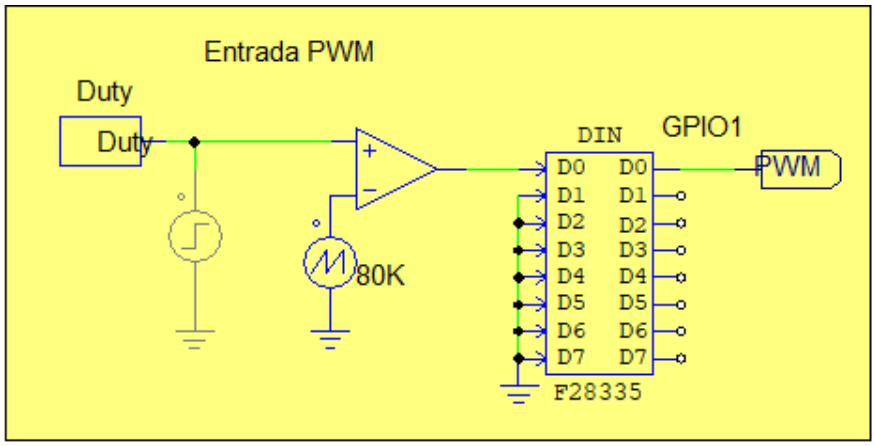


Figura 62. Configuración entrada PWM.

Configuración de la entrada digital:

1. Se da doble clic en la configuración de hardware.
2. Se selecciona el GPIO que se quiere como entrada.
3. En el módulo de entradas digitales se da doble clic y selecciona en el puerto 0 el GPIO seleccionado en el paso 2.

Luego de esto se simula el modelo, se comprueba que todo este correcto y se genera nuevamente el código.

Nota: *Code compeser studio (CCS)* actualiza el código cada que se realiza un cambio en PSIM y se genere el código.

2.10 Paso 10: Programación Generador PWM.

En otra DSP se puede programar una salida PWM con Duty configurable usando una variable global para poder cambiarlo en tiempo real. Esta programación se debe hacer en

un archivo nuevo de PSIM diferente a donde está el modelo, para la generación y programación del código se siguen los mismos pasos explicados anteriormente. A continuación, se muestra el diagrama y la explicación del módulo PWM de *SimCoder.*

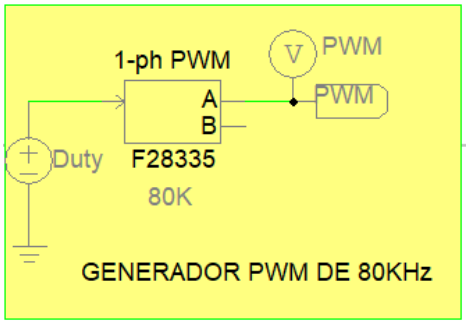


Figura 63. Generador PWM en PSIM *SimCoder.*

El generador PWM se encuentra en la librería de PSIM con el nombre *1-ph PWM* y tiene la siguiente configuración:

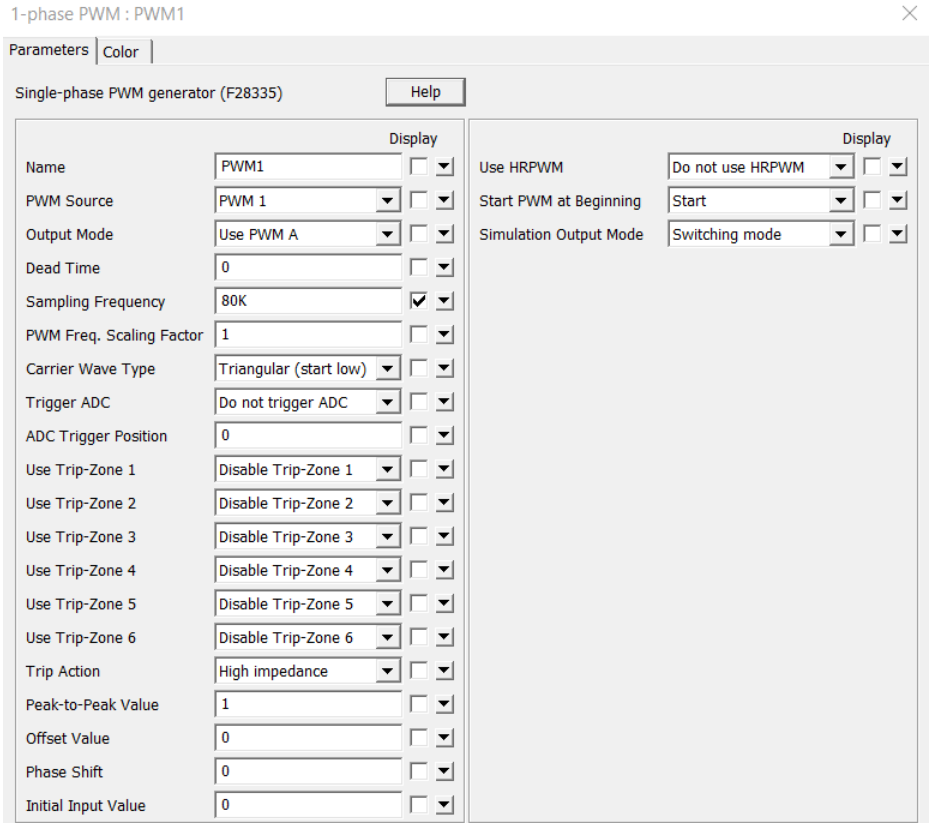


Figura 64. Configuración generadora PWM de una fase.

- ***PWM Source:*** La DSP contiene internamente 6 bloques PWM configurables cada uno con 2 salidas, en este campo se selecciona el PWM que se quiere utilizar, cada uno tiene pines GPIO asignados como se puede ver en la siguiente lista.

- PWM 1 (GPIO 0 and 1)
- PWM 2 (GPIO 2 and 3)
- PWM 3 (GPIO 4 and 5)
- PWM 4 (GPIO 6 and 7)
- PWM 5 (GPIO 8 and 9)
- PWM 6 (GPIO 10 and 11)

Nota: Dependiendo del PWM a utilizar, se deben seleccionar en el bloque de configuración de la DSP los pines como PWM, para este caso el PWM 1 serían el GPIO0 Y GPIO1.

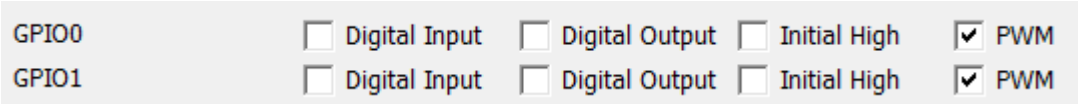


Figura 65. Configuraciones salidas PWM

- ***Output mode:*** Selecciona el canal que se va a utilizar, A, B o A&B. Cada canal corresponde a un pin de salida PWM según configuración anterior.

- ***Dead time:*** Corresponde al tiempo de retardo del generador PWM, se recomienda dejar este valor en cero para no tener errores en el resultado final del generador.

- ***Sampling frequency:*** Es la frecuencia de muestreo del PWM, esta corresponde a la frecuencia de diseño del convertidor.

- ***PWM Freq. Scaling Factor:*** Es la relación entre la frecuencia de conmutación PWM y la frecuencia de muestreo, puede ser de 1 a 100. Por ejemplo, si la frecuencia de muestreo es 50 kHz y el factor de escala es 3, la frecuencia de conmutación PWM será de 150 kHz. Es decir, los interruptores funcionarán a 150 kHz. Pero las señales de puerta se actualizarán a 50 kHz, o una vez cada 3 ciclos de conmutación. Se recomienda dejar un factor de escala de 1 para que la frecuencia de conmutación sea igual a la de muestreo.

- ***Carrier Wave Type:*** Selecciona el tipo de onda de la salida PWM, se pueden seleccionar 4 tipos de ondas:
  - *Triangular (start low):* Onda triangular con inicio en estado bajo.
  - *Triangular (start high):* Onda triangular con inicio en estado alto.
  - *Sawtooth (start low):* Onda diente de sierra con inicio en estado bajo.
  - *Sawtooth (start high):* Onda diente de sierra con inicio en estado alto.
- ***Trigger ADC:*** Establece si el generador PWM debe activar el convertidor A/D, Esta configuración aplica para la programación de controladores en la DSP, se tienen las siguientes opciones de configuración:
  - *Do not trigger ADC:* No active el convertidor A/D.
  - *Trigger ADC Group A:* Activa el grupo A del convertidor A/D (8 bits).

- *Trigger ADC Group B:* Activa el grupo B del convertidor A/D (8 bits).
- *Trigger ADC Group A&B:* Activa el grupo A y B del convertidor A/D (16 bits).

Los demás parámetros se dejan como viene por defecto. Si se desea profundizar en la configuración del generador PWM y los demás generadores disponibles, se puede encontrar en el capítulo *6,4* del manual de *SimCoder* o en la ayuda de PSIM.

2.11 Paso 11: HIL en lazo abierto.

Teniendo programado en una DSP el modelo y en otra el generador PWM con Duty configurable, se puede hacer una primera experimentación HIL en lazo abierto interconectando ambas DSP's, hacer cambios en el Duty como se realizó en el paso 8 y verificar con multímetro el voltaje promedio de salida y la señal en el osciloscopio. El diagrama de conexión se muestra a continuación:

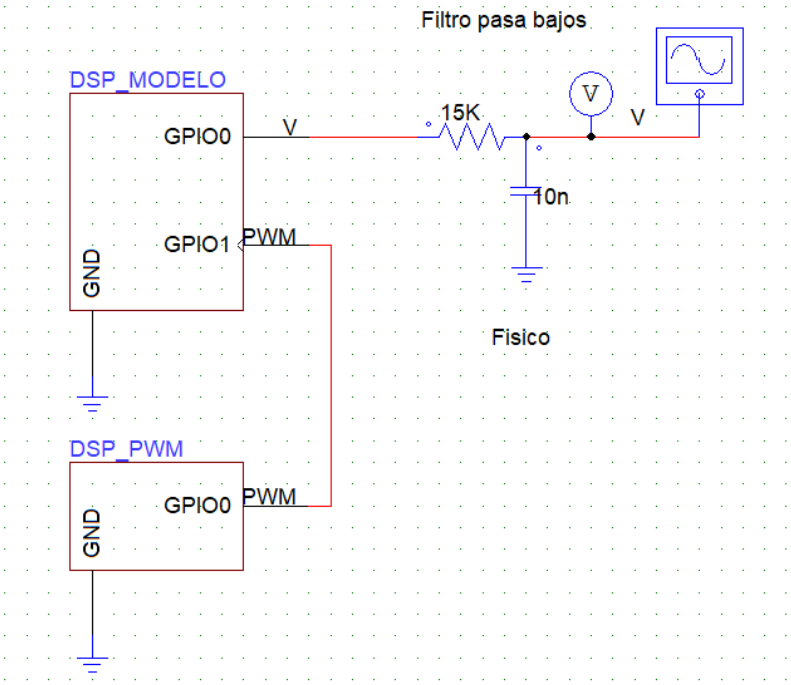


Figura 66. Diagrama de conexión en lazo abierto.

Nota: Se recomienda primero programar la DSP que contiene el modelo y luego de que esta esté corriendo en *Code Composer Studio,* proceder a programar la DSP que contiene el generador PWM, esto se hace para no generar conflictos en la comunicación USB serial entre el PC y las tarjetas.

Resultados de la simulación en lazo abierto:

Tabla 2. Comparación de resultados en lazo abierto.

Duty	Voltaje simulado	Voltaje lazo abierto	error relativo
0,5	2,038	1,976	3%
0,57	2,5	2,5	0%

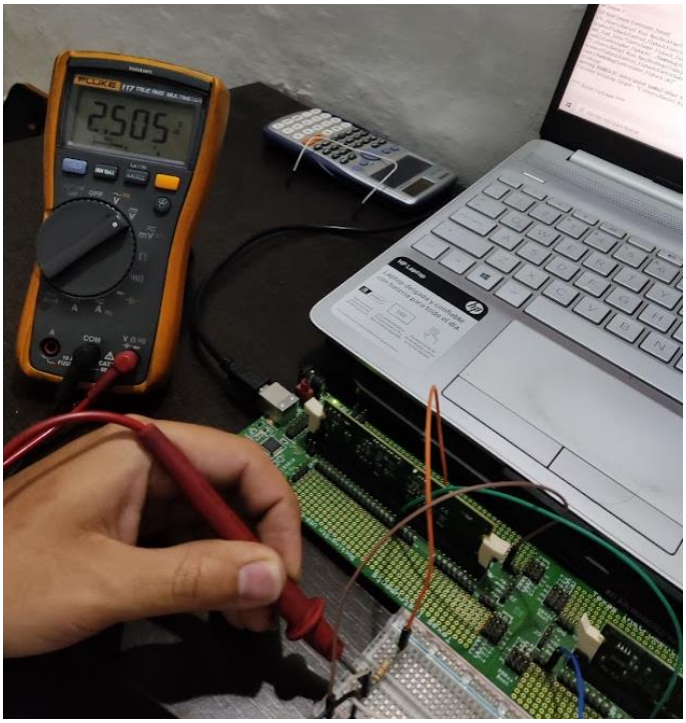


Figura 67. Lectura de voltaje en lazo abierto con Duty de 0.570 y PWM externo.

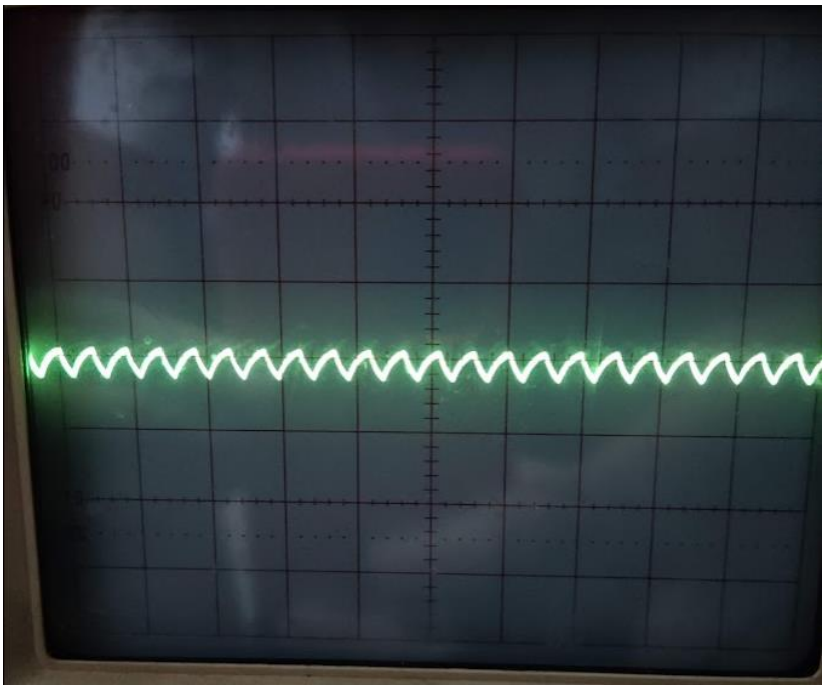


Figura 68. Lectura en el osciloscopio de salida de voltaje en lazo abierto.

Con la experimentación *HIL* en lazo abierto de control se logra una buena respuesta, luego de tener este punto validado y correctamente funcionando se procede con el diseño del controlado PI para el convertidor en estudio.

2.12 Paso 12: Diseño del controlador discreto.

Para el diseño del controlador discreto, se parte primero de un controlador en modo continuo, este se realiza a partir de la linealización del sistema y cálculos en el punto de operación usando Matlab. La función de transferencia obtenida para el convertidor FlyBack es:

```
G_v_d =

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

Figura 69. función de transferencia convertidor FlyBack.

A partir de la función de transferencia se obtienen los parámetros Kp y Ki para el controlador.

$$Kp + Ki * \frac{1}{s}$$

with Kp = 5.7e-06, Ki = 1.65

Figura 70. Parámetros del controlador PI

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

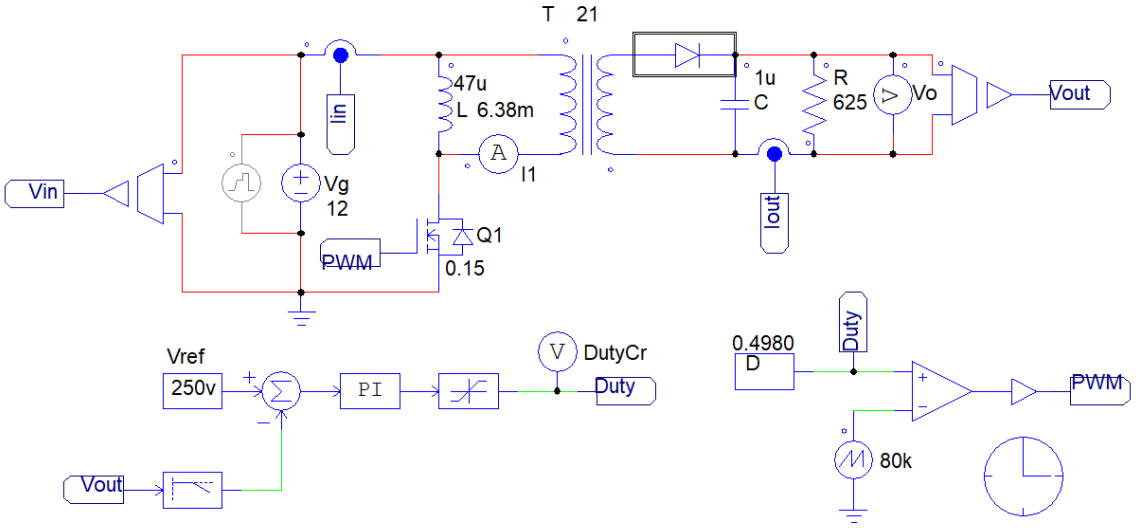


Figura 71. Convertidor FlyBack con controlador PI continuo en PSIM.

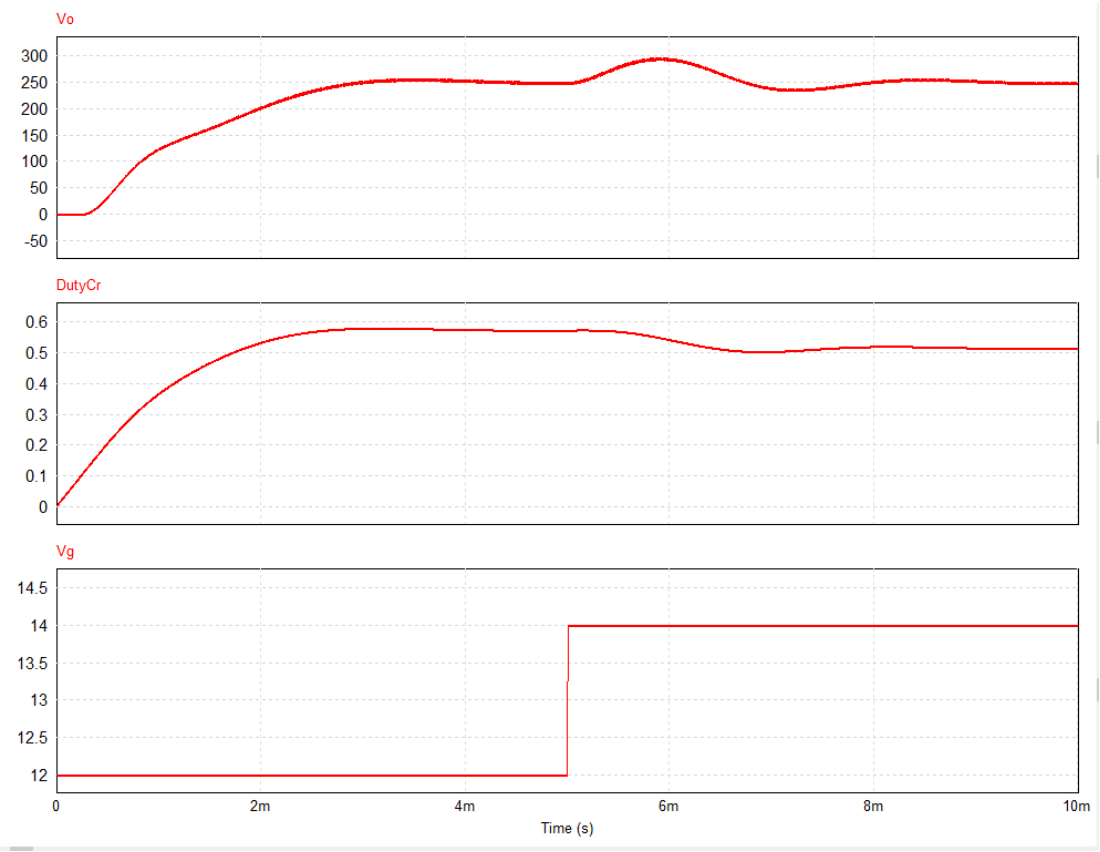


Figura 72. Respuesta simulación del controlador PI continúo aplicando cambios en Vg.

Luego de validar que el controlador funciona correctamente en modo continuo, se procede a convertirlo a modo discreto, para esto se puede usar 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), para así poder ser programado en un hardware de destino como la DSP, para el ejemplo de estudio se realiza la siguiente configuración:

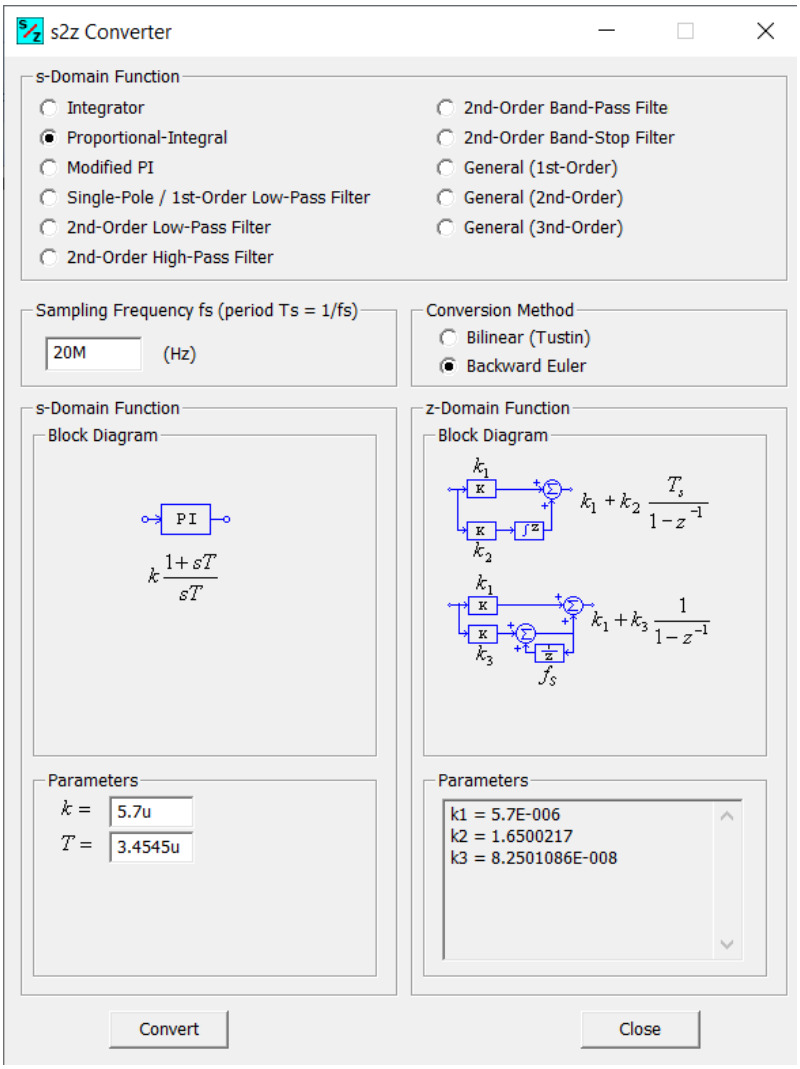


Figura 73. Configuración del convertidor de controlador PI continuo a discreto.

Es importante poner una frecuencia de muestreo mayor a la frecuencia de conmutación del convertidor, en este caso es de 20MHz ya que es la frecuencia de muestreo que se utilizó en el modelo discretizado. A continuación, se muestra el controlador discreto y su simulación:

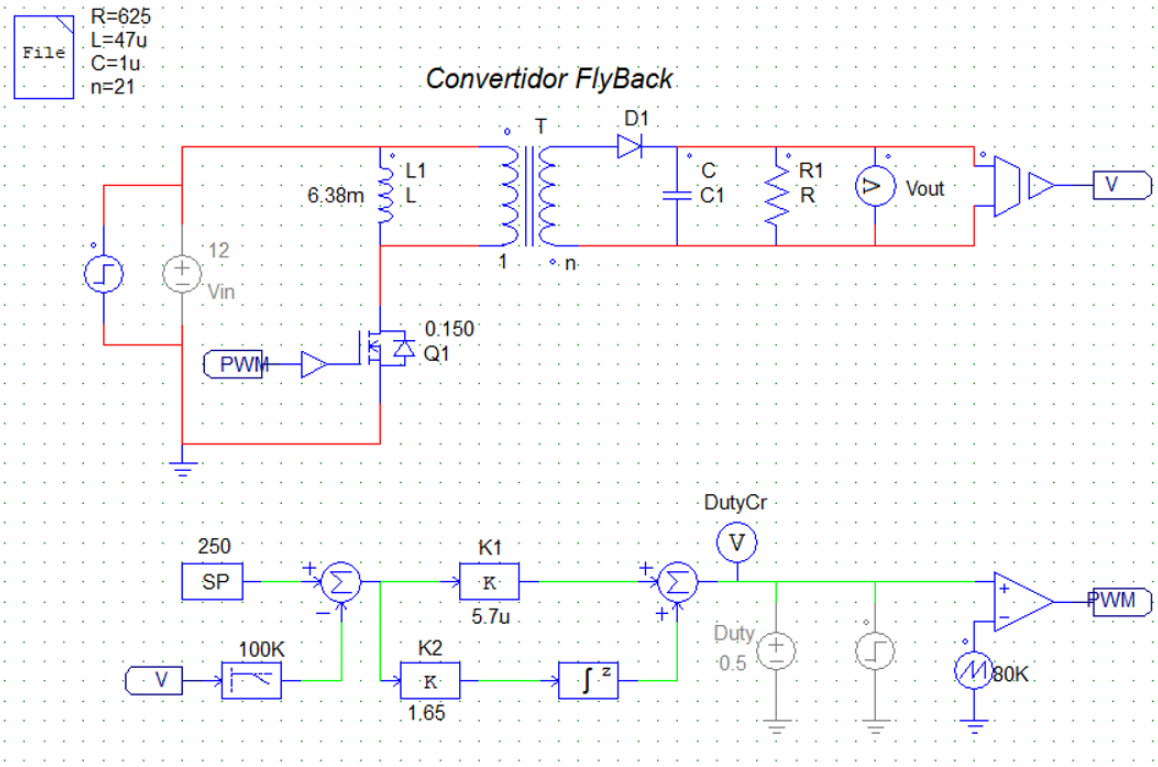


Figura 74. Convertidor FlyBack con controlador PI discreto.

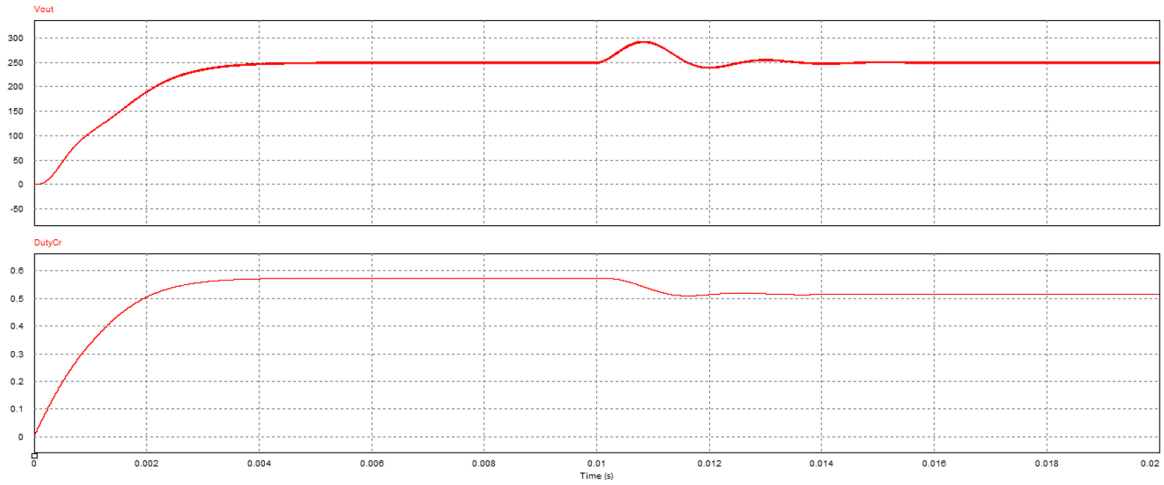


Figura 75. Respuesta simulación del controlador PI discreto aplicando cambios en Vg

Luego de comprobar la respuesta del controlador en la simulación se procede con el siguiente paso.

2.13 Paso 13: 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 PWM, basta con incluir dos elementos de hardware de SimCoder como lo son el conversor análogo digital (ADC) y el generador PWM visto anteriormente, generar el código, cargar y correr el programa en la DSP con *code composer studio.* A continuación, se muestra el diagrama de programación en PSIM:

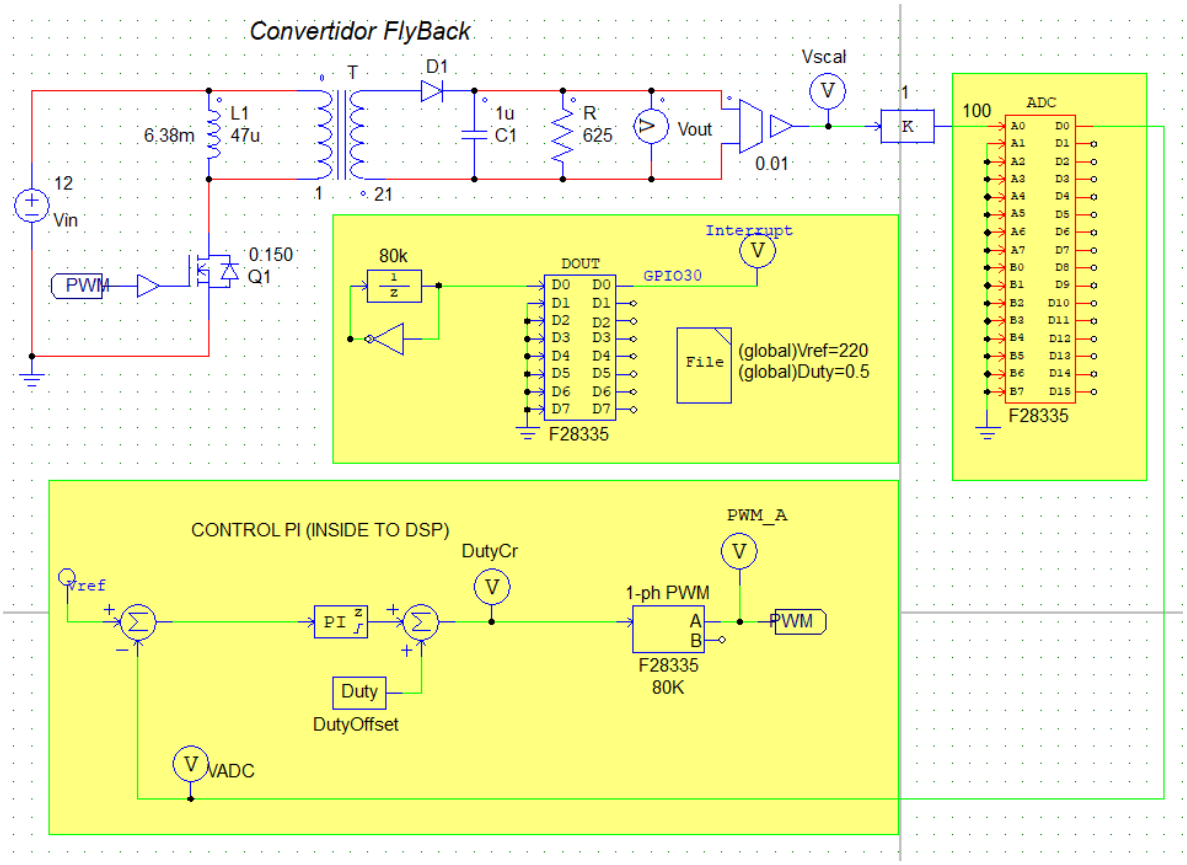


Figura. 76. Diagrama de control PI con elementos de hardware para programación en DSP F28335.

Todos los elementos resaltados en amarillo serán los programados en la DSP, 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, A continuación, se muestra la configuración y explicación de estos elementos.

Conversor análogo 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 del convertidor, la configuración del bloque ADC se muestra a continuación:

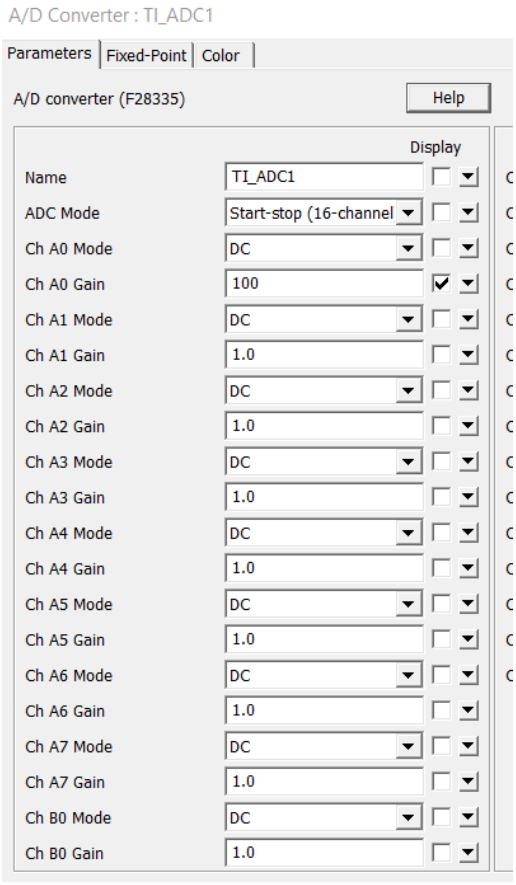


Figura. 77. Configuración modulo ADC para el controlador PI.

- *ADC Mode:* Es el modo de operación del conversor análogo digital el cual se puede seleccionar 3 opciones: Continua, *start-stop* de 8 canales (Solo canal A o B) y *start-stop* de 16 canales (Utiliza el canal A & B)
- *Ch Mode:* Cada canal se puede seleccionar si la entrada es en modo DC o AC.
- *Ch Gain:* Se puede poner el valor de escala para trabajar con la señal original, por ejemplo, si la señal original es de 250v y esta escalizada a 2.5V, al poner una ganancia de 100 en el canal del conversor ADC, se obtendrá el mismo valor de la señal original. (2.5V x100 =250v), esto es importante ya que el controlador esta diseñado para trabajar con el valor de diseño del convertidor.

Salida PWM del controlador.

Para la salida PWM del controlador se usa el mismo generador usado en el punto 11 (Figura 63), pero con un cambio en el parámetro *Trigger ADC,* el cual activa al conversor análogo

digital para trabajar de manera sincrónica. El parámetro seleccionado dependerá del modo ADC en el conversor análogo digital, si en el conversor ADC se utiliza el modo de 8 canales, en la salida PWM se debe seleccionar el canal A o B dependiendo donde se tenga la entrada, si se utiliza el modo de 16 canales, el *trigger* a usar en el módulo PWM sería el grupo A&B,

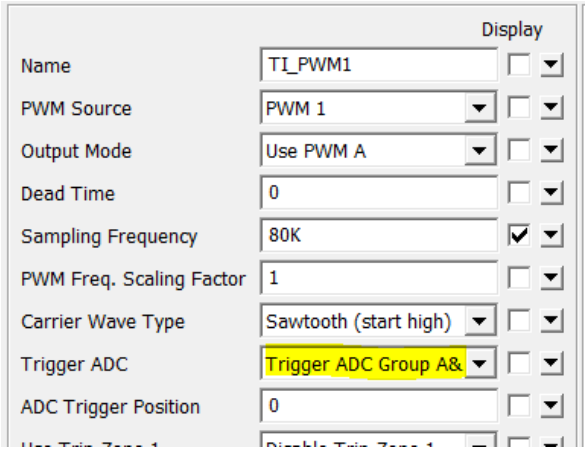


Figura. 78. Configuración salida PWM del controlador PI.

Luego de configurar y simular el controlador, se procede con la generación y programación del código en la DSP.

2.14 Paso 14: Plataforma HIL en lazo cerrado de control.

Como último paso y teniendo programado el modelo en una DSP y el controlador en otra, se procede hacer la interconexión entre las tarjetas como se muestra a continuación:

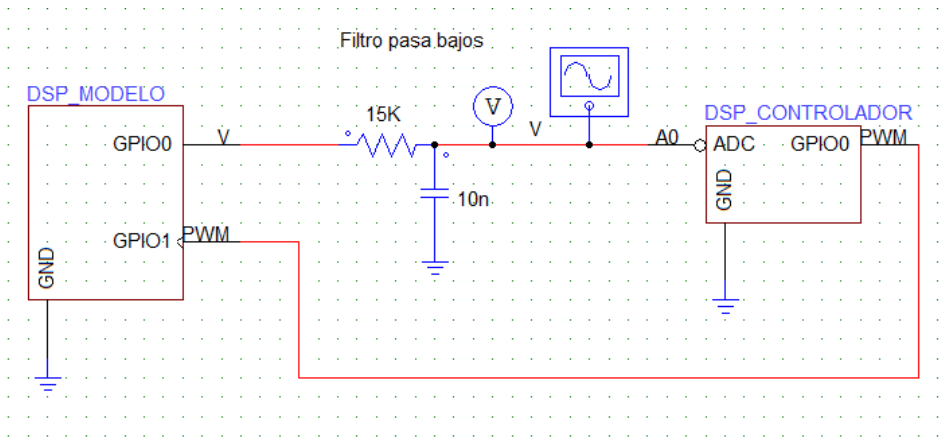


Figura. 79. Diagrama de conexión en lazo cerrado.

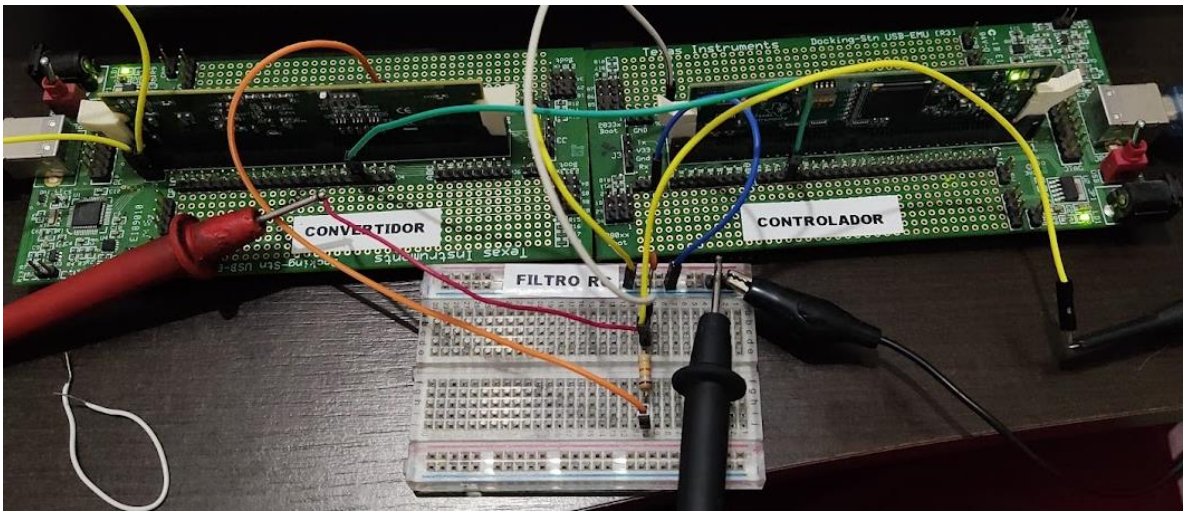


Figura. 80. Montaje físico plataforma HIL.

A continuación, se realizan diferentes cambios en el set point de voltaje, voltaje de entrada (V_g) y carga (R) y se comparan los resultados con la simulación de PSIM:

Tabla 3. Comparación de resultados cambiando el voltaje de referencia.

Voltaje de Referencia	Voltaje promedio simulado	Voltaje promedio HIL medido	Voltaje promedio HIL escalizado	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 4. Comparación de resultados cambiando el voltaje de entrada.

VG	Voltaje promedio simulado	Voltaje promedio HIL medido	Voltaje promedio HIL escalizado	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 5. Comparación de resultados cambiando la resistencia de carga.

R Carga	Voltaje promedio simulado	Voltaje promedio HIL medido	Voltaje promedio HIL escalizado	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%

En toda la simulación se observa una buena respuesta del controlador, cabe anotar que cuando se realizan 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.

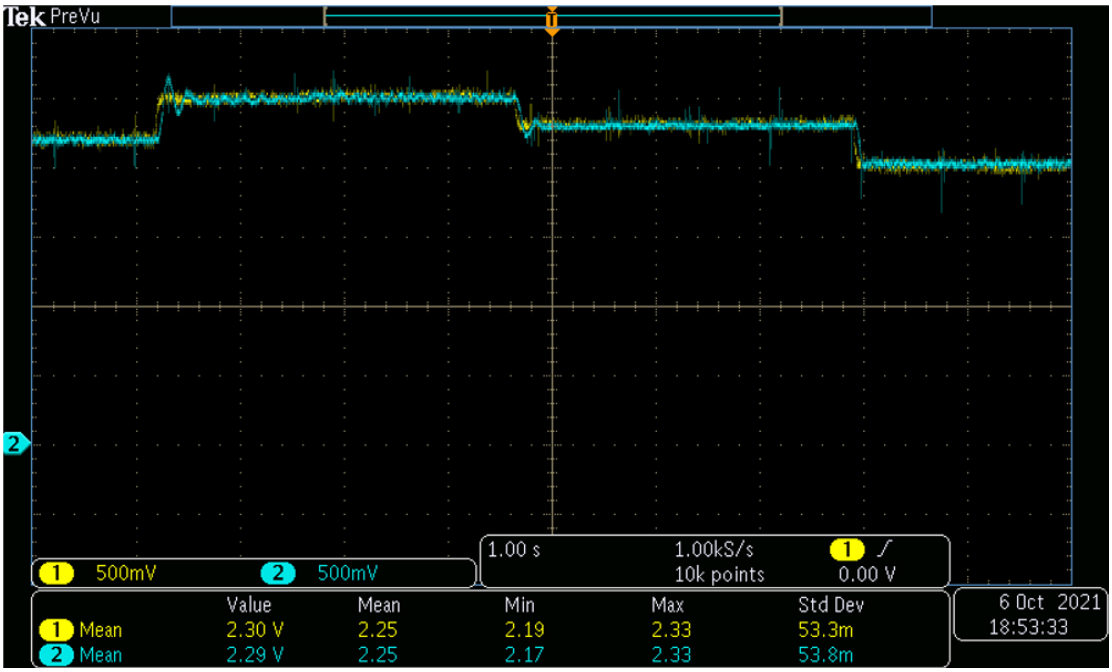


Figura 81. Lectura del voltaje de salida promedio en osciloscopio digital.

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 Setpoint 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

Con esta simulación de *Hardware in the loop (HIL),* se logró verificar que el diseño del convertidor y su controlador serian óptimos para una implementación real.

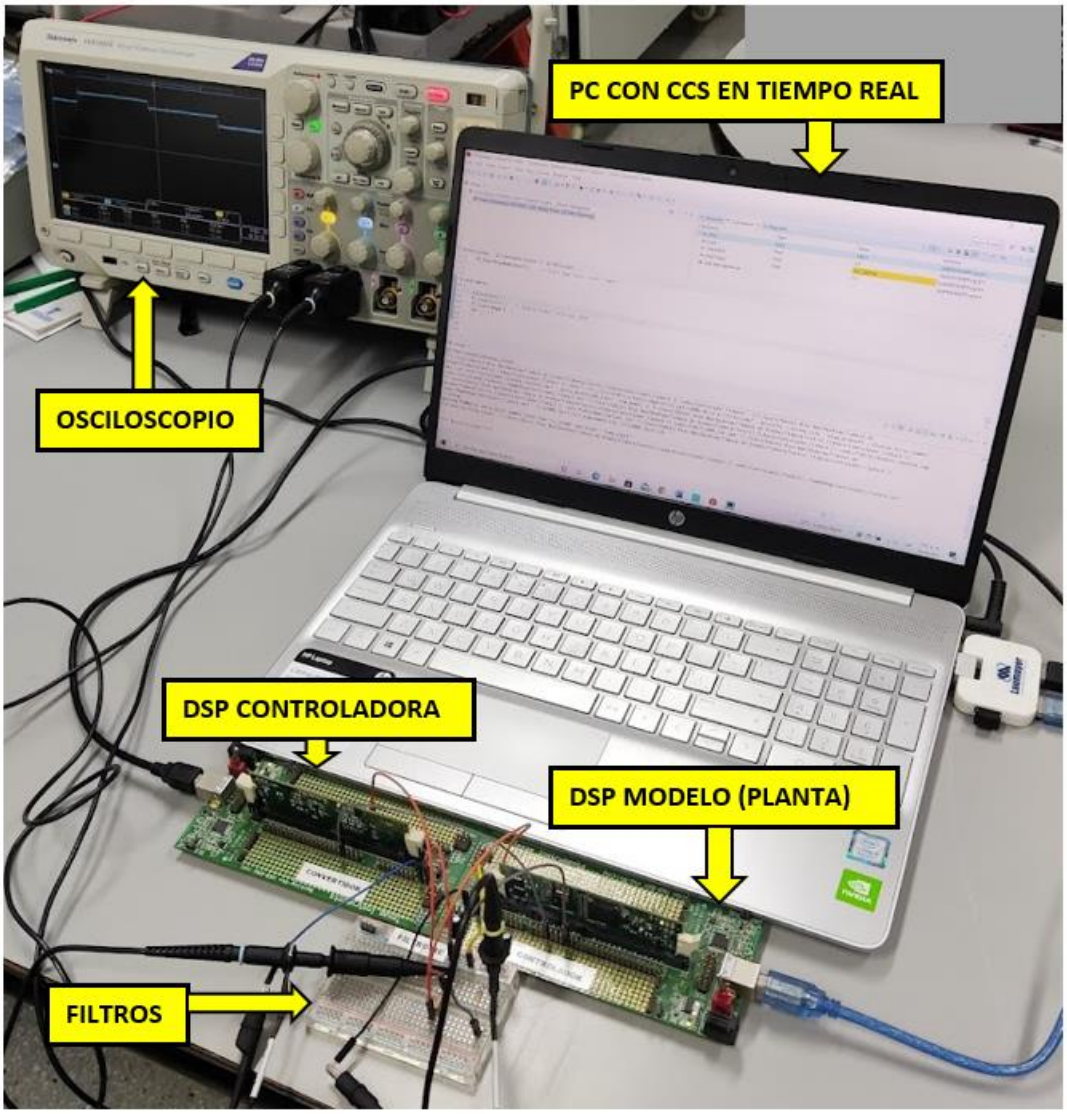


Figura 82. Implementación final de la plataforma HIL.

3. BIBLIOGRAFÍA

[1] R. W. Erickson and D. Maksimovic, *Fundamentals of Power Electronics*, 2 ed. Springer, 2005.

[2] Powersim Inc, "SimCoder User´s Guide.," Mayo 2020. [Online]. Available: www.powersimtech.com.