Universidad
Carlos III de Madrid
www.uc3m.es

Departamento de Tecnología Electrónica

Grado en Ingeniería Electrónica Industrial y Automática.

TRABAJO FIN DE GRADO

Diseño e Implementación con un Microprocesador del Lazo de Control de un Convertidor de Potencia

Autora: Celia Carrasco Braojos

Tutoras: Clara Marina Sanz García

Marta Portela García

Agradecimientos:

Para terminar el trabajo me gustaría agradecer el apoyo y la ayuda de todas las personas que me han ayudado todos estos años.

A toda mi familia, en especial a Milagros y Jesús, mis padres, a Nuria, mi hermana, por el cariño, apoyo y confianza en todo este tiempo y en especial durante la realización del Trabajo Fin de Grado ya que sin ellos no hubiera sido posible.

A mis amigos de siempre, Rober, Irene, Aroa,… por estar ahí siempre preocupándose y ayudándome a desconectar. Y en especial a Nacho por su ayuda y apoyo a lo largo de toda esta etapa.

A todos los amigos conocidos durante la carrera, Víctor, Elena, Nerea, Álvaro, Juli, Javi, Marco, Arturo, Jorge, Rober,… por hacer de etapa una de las mejores, y compartir conmigo todos estos momentos.

Por último a mis tutoras Marta y Marina, por el apoyo recibido durante la realización del trabajo, y por todos los conocimientos que he adquirido gracias a ellas.

Índice

Índice de Figuras:

Índice de Tablas:

Capítulo 1 Introducción

1.1 Ámbito del proyecto

El Trabajo Fin de Grado "*Diseño e Implementación con un Microprocesador del Lazo de Control de un Convertidor de Potencia*", es un proyecto que se ha realizado en la Universidad Carlos III de Madrid, en el departamento de Tecnología Electrónica, haciendo uso de las instalaciones de laboratorios del grupo de Sistemas Electrónicos de Potencia y bajo la tutela de sus profesores.

Los convertidores de potencia se usan para adecuar la energía eléctrica que reciben en la necesitada por una determinada carga. Existen varios tipos de convertidores en función del tipo de energía con la que se trabaje. Para este caso se utilizara un convertidor CC/CC [1].

Los convertidores CC/CC suelen estar conectados a una red que ofrece una tensión continua no regulada, esta tensión, que se convertirá en la tensión de entrada al convertidor, puede fluctuar debido a la naturaleza senoidal de la tensión rectificada.

Por consiguiente los convertidores CC/CC son utilizados para convertir la tensión de entrada no regulada en la tensión de salida requerida por la carga.

En este proyecto se va a utilizar un convertidor reductor CC/CC que se usa cuando la fuente primaria de energía tiene un nivel de magnitud más elevado que el demandado por la carga ver Figura 1.1

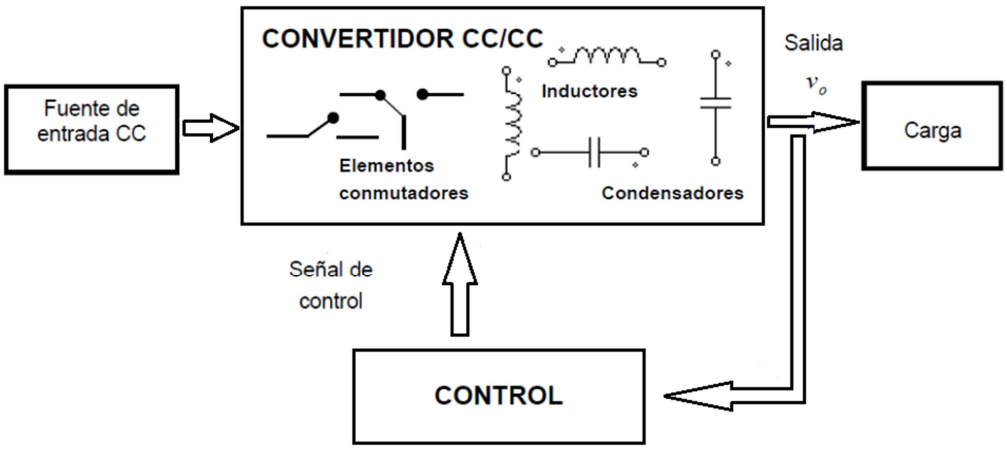


Figura 1.1 Esquema general de un convertidor CC/CC.

1.2 Necesidad de control

El convertidor debe comportarse según las especificaciones requeridas y para ello es necesario hacer un control durante su funcionamiento, como se puede observar en la Figura 1.1.

Para conseguir que el rango de la señal en la salida del convertidor tenga el menor error posible y que no se vea afectada por perturbaciones es necesario diseñar un lazo de control que regule el comportamiento del convertidor de potencia. Dicha regulación se hace mediante un circuito que de forma automática a través de la generación de una señal que ajustará el tiempo de encendido del interruptor que controla el convertidor, abriéndolo o cerrándolo durante un determinado periodo de tiempo.

El módulo PWM ("*Pulse-Width Modulation*"), es el encargado de generar la señal de gobierno del convertidor que define el tiempo que tiene que estar cerrado el interruptor frente al tiempo total de conmutación en la Figura 1.2 se pueden observar dos señales de gobierno para el interruptor, una de ellas con un ciclo de trabajo (D) del 50% y otro del 20% (indica el porcentaje que el interruptor está cerrado respecto al tiempo total de conmutación)

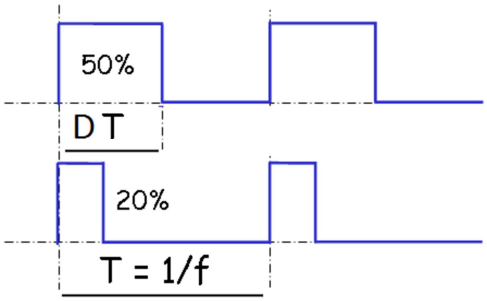


Figura 1.2: Señal generada por el módulo PWM

En este trabajo se va a diseñar e implementar el lazo de control en un microprocesador de un convertidor de potencia CC/CC.

Para obtener un buen lazo de control para el convertidor se debe hacer un buen análisis del comportamiento de dicho control. En este proyecto en primer lugar se va hacer un análisis del control en analógico, ya que se tienen los conocimientos necesarios para poder hacer el análisis. Y A continuación se procede a convertir los resultados obtenidos en el sistema analógico a su equivalente en digital. El paso de analógico a digital se debe a que este presenta múltiples ventajas que se detallan a lo largo de este capítulo.

1.2.1 Tipos de implementación del lazo de Control

Como se ha indicado anteriormente, existen dos tipos de control: analógico y digital. La diferencia principal entre los dos sistemas es que el sistema analógico utiliza un sistema continuo y el digital un sistema discreto.

En la actualidad el control digital se impone al analógico ya que presenta muchas ventajas respecto a éste [2]:

- Un control digital puede implementarse en sistema digital programable y esto proporciona flexibilidad a la hora de reconfigurar las operaciones de procesamiento digital, simplemente cambiando la configuración del dispositivo. En cambio en un sistema analógico un cambio implicaría el rediseño del hardware, lo que requiere un mayor coste.

- Un sistema digital tiene una reproducibilidad de resultados muy alta. Para una misma entrada siempre tendrá la misma salida, mientras que los sistemas analógicos presentan una mayor sensibilidad a las condiciones externas como la temperatura, la antigüedad de los componentes, etc.

- El procesamiento digital de señales también posibilita la implementación de algoritmos de procesado de señal más sofisticados. Estos algoritmos son difíciles de conseguir mediante implementación analógica.

- Un sistema digital es más económico, puede proporcionar mucha funcionalidad en un espacio muy pequeño, y un mismo dispositivo sirve para realizar varias tareas, siempre y cuando se programe para ello.

El control digital puede llevarse a cabo mediante distintos dispositivos, presentando cada uno de ellos diferentes características. A continuación se describen el control mediante FPGA y mediante un microprocesador.

- **Control mediante FPGA**

Dentro de los posibles controles que ofrece el campo digital se puede utilizar una FPGA (“*Field Programmable Gate Array”*). Este dispositivo permite hacer una programación del hardware implementado mediante lenguajes de descripción de hardware de alto nivel como el lenguaje *VHDL*, lo que permite una gran flexibilidad y la posibilidad de reprogramar el sistema fácilmente.

Además, permiten la paralelización de tareas lo que incrementa las prestaciones del sistema implementado.

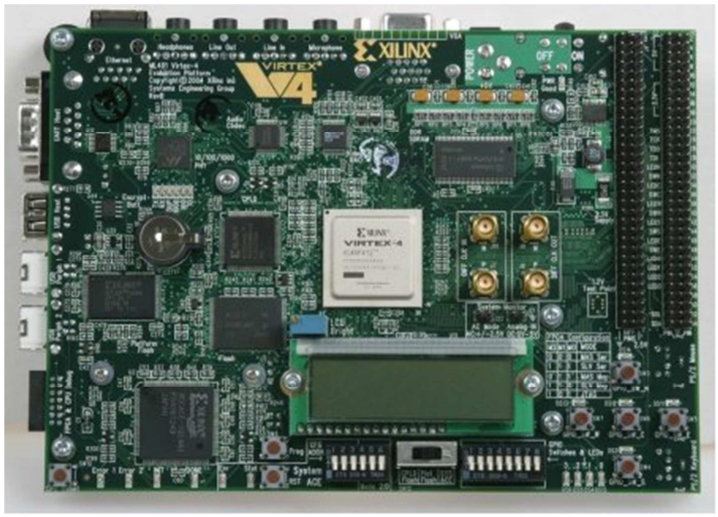


Figura 1.3: Imagen de una FPGA

- **Control mediante microprocesador**

Otra posibilidad para realizar el control digital es el uso de un microprocesador. Este es el método que se va a utilizar en este proyecto. Se trata de un circuito integrado que puede ser programado con lenguajes de alto nivel, como por ejemplo C, lo que facilita su uso. El uso de estos lenguajes es posible porque existen compiladores cruzados que traducen este lenguaje a lenguajes de más bajo nivel, como el ensamblador.

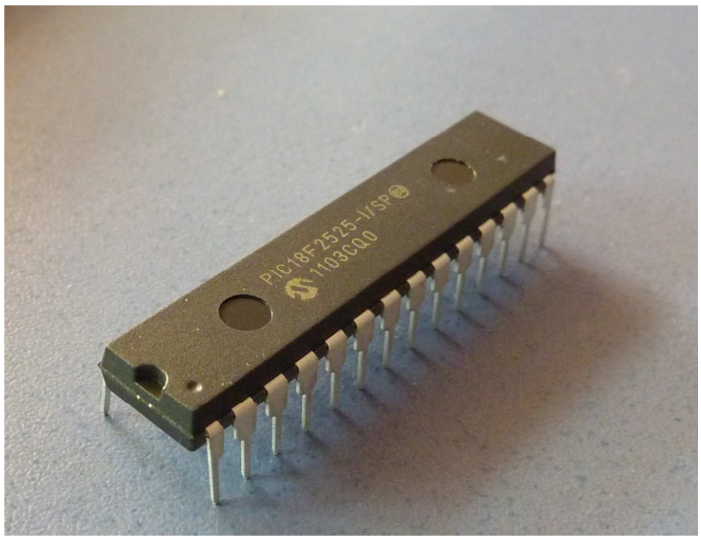


Figura 1.4: Imagen de un PIC18F2525

1.3 Herramientas utilizadas para el diseño e implementación

Para la realización de este proyecto se han utilizado las siguientes herramientas:

- **Psim:** Es una herramienta de simulación de circuitos eléctricos y electrónicos. Es muy intuitiva ya que se hace por medio de una interfaz gráfica para crear el/los sistema/s que se quieran analizar. Esta herramienta incluye a su vez otra llamada ***SmartCtrl*** que permite realizar el diseño del lazo de control de un convertidor de potencia de una forma sencilla. En este proyecto se utilizaran las dos herramientas, se utiliza el PSIM para analizar el convertidor de potencia y el Smartctrl para diseñar el lazo de control analógico.

- **Matlab:** Es una herramienta matemática de altas prestaciones. En este proyecto se va a utilizar para discretizar las ecuaciones del plano continuo (s) al discreto (z)

- **MPLAB IDE:** Es un software gratuito ofrecido por Microchip para la programación y depuración de sus microprocesadores. Esta herramienta ha sido utilizada para la implementación del lazo de control en el microprocesador.

- **ORCAD:** es un programa que sirve para el diseño de circuitos electrónicos. Consta de dos bloques:

  - *PSPICE*: sirve para ver el comportamiento de los circuitos electrónicos. En este proyecto se va utilizar para definir el circuito del hardware necesario para el funcionamiento del convertidor para posteriormente exportarlo a la herramienta LAYOUT

  - *LAYOUT*: sirve para realizar el diseño de placas de circuitos impresos PCB. Este diseño se realiza a través de la exportación del circuito de PSPICE. La Herramienta LAYOUT tiene definidas librerías con las huellas de los componentes y en caso de tener una huella asociada a un componente concreto te permite crearla.

1.4 Objetivos y alcance del proyecto

Los objetivos planteados para este proyecto son los que se exponen a continuación:

- El objetivo principal es diseñar el lazo de control digital para un convertidor CC/CC e implementarlo en un microprocesador.

- Adaptar el proceso de diseño de la herramienta "*SmartCtrl*" a los controles digitales, teniendo en cuenta las características de estos últimos.

- Diseñar la placa de circuito impreso (PCB) para el montaje del convertidor y los componentes necesarios para su funcionamiento.

- Aprender y aplicar una metodología de diseño del lazo de control digital a partir de su diseño analógico

- Documentar el procedimiento realizado para su utilización como referencia en futuros trabajos.

1.5 Breve descripción de la memoria

La finalidad de este proyecto es diseñar e implementar el lazo de control digital. Esto se hará a partir del diseño de un lazo analógico, ya que como se ha explicado anteriormente es el diseño que se conoce en detalle.

Para llevar a cabo este diseño será necesario conocer cada bloque del lazo analógico y digital, y cómo están relacionados unos bloques con otros. Después será necesario hacer el cambio correctamente del sistema analógico al digital. Para este proyecto la implementación del lazo digital se hará en un microprocesador. Los bloques en los que se ha dividido la memoria son los siguientes:

- Capítulo 2: en el capítulo 2 se hace una breve descripción de los conceptos teóricos necesarios para llevar a cabo el diseño de ambos sistemas de control, tanto el analógico como el digital.

- Capítulo 3: en el siguiente capítulo se explican los pasos que se deben seguir para realizar el diseño del lazo en el caso concreto de nuestro convertidor. Se explican las características de todos los elementos que se van a utilizar, tanto

del convertidor como del microprocesador. De estas especificaciones dependen los pasos a seguir para realizar un buen diseño.

- Capítulo 4: en el capítulo 4 se explica el diseño de la placa impresa que necesita el convertidor, ya que es necesario conectar algunos elementos para que éste funcione correctamente.

- Capítulo 5: en el capítulo número 5 se describe la validación del sistema. Se simulan ambos tipos el control para comprobar que se comportan de la misma manera, ver si se obtiene el resultado deseado, y en caso contrario analizar el fallo.

- Capítulo 6: por último, después de analizar los resultados obtenidos del diseño se exponen las conclusiones a las que se ha llegado y se analizan los posibles trabajos futuros que se pueden realizar.

- Capítulo 7: en este capítulo se presenta el presupuesto del proyecto.

Capítulo 2 Diseño del lazo de control de un convertidor de potencia

En este capítulo se va a realizar un análisis de los bloques de los que consta un sistema completo [3]:

- Convertidor :
  - Planta
- Control:
  - Regulador
  - Modulador
  - Sensado.

Como se explicó anteriormente en primer lugar, se estudia el diseño teórico de estos 4 bloques en analógico, y a continuación se hará el análisis equivalente para el diseño en digital.

2.1 Descripción de los bloques del sistema Analógico

Los sistemas electrónicos de potencia controlan y transforman la energía eléctrica mediante circuitos electrónicos y se usan para:

- Convertir eficientemente la energía
- Controlar la energía
- Acondicionar la energía

Para conseguir lo anterior es necesario hacer un buen diseño del sistema, esto implica que además de una buena elección del convertidor adecuado para nuestro proyecto, es importante realizar un control de la salida de éste. De esta forma se asegura que en todo el rango de funcionamiento del convertidor la salida es la deseada.

2.1.1 Descripción de los bloques de control

Como se ha destacado anteriormente un buen diseño del lazo de control es de vital importancia para alcanzar los objetivos que se deben cumplir en el sistema de potencia. Entre los objetivos principales de la etapa de control se encuentran los siguientes:

- Garantizar la estabilidad del sistema para todos los puntos de trabajo.
- Modular el ciclo de trabajo para compensar las variaciones de la entrada y de la carga, así como posibles variaciones en los valores de los elementos del circuito.
- Conseguir que el sistema presente una respuesta dinámica adecuada.

Como se ha mencionado al principio del capítulo en el caso de un control analógico se observan los siguientes bloques: sensado, modulador y regulador (ver Figura 2.1).

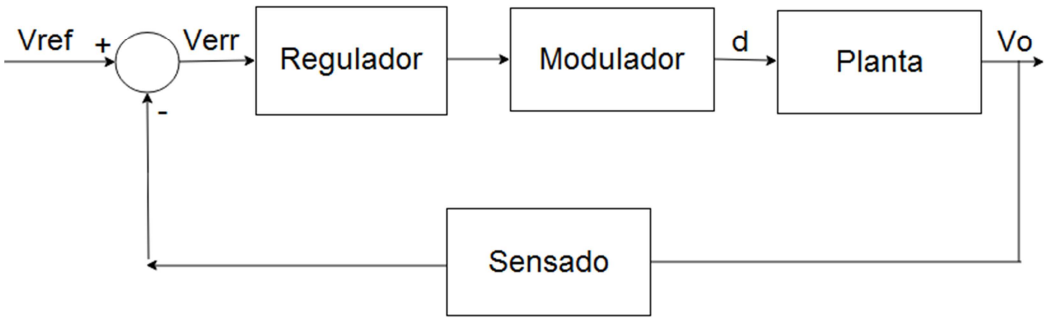


Figura 2.1: Diagrama de bloques de un sistema de potencia general

- **Sensado:** es el bloque en el que se mide la tensión de salida del convertidor y se acondiciona para la etapa siguiente del lazo. En esa etapa se obtiene el error (Verr) entre la señal de salida deseada (Vref) y la señal de salida que tenemos en ese momento (Vo), restando ambas señales.
- **Regulador**: es el encargado de compensar las variaciones que se producen en la tensión de salida, ya sean provocadas por variaciones en la entrada o en la carga. Y garantizar la estabilidad del sistema
- **Modulador**: es el encargado de generar la señal de gobierno del convertidor. Dicha señal (d) es una señal cuadrada que se construye mediante la comparación de una rampa (señal triangular), de la frecuencia de conmutación del convertidor y una tensión continua (tensión de salida del regulador). Esta señal cuadrada toma valores de 0 y 1, de manera que el interruptor permanecerá abierto cuando la señal que le llegue sea 1 y permanecerá cerrado cuando la señal que le llegue sea 0. En la Figura 2.2 podemos observar la comparación de las dos señales y la obtención de la

señal del ciclo de trabajo, donde Vcomp es la señal obtenida tras el bloque del regulador.

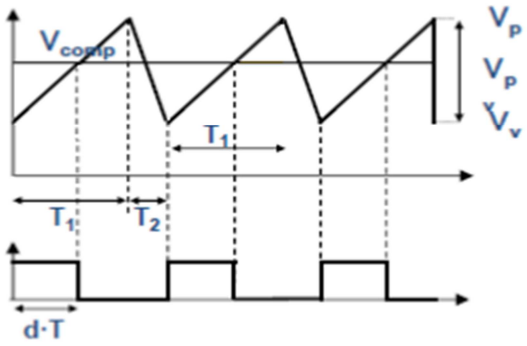


Figura 2.2: Modulación PWM

Para realizar el diseño de control analógico se utilizaran las técnicas de cálculo de regulación y estabilidad, el requisito para poder aplicar estas técnicas es que el sistema sea lineal. Debido que los convertidores de potencia tienen elementos no lineales se debe hacer el modelado del sistema para la obtención de un sistema lineal.

2.2 Modelado del sistema

Para realizar un buen diseño del lazo de control es necesario conocer la dinámica del sistema para ver qué variables pueden sufrir perturbaciones y cómo afectan a la salida. El problema de los convertidores es su no linealidad, ya que debido a la acción del interruptor tiene dos estados diferentes de funcionamiento, y las técnicas de cálculo de regulación y estabilidad son métodos desarrollados para sistemas lineales. Por lo que es necesario realizar un modelo lineal invariante en el tiempo. Esto se realiza mediante el método de modelado, que puede llevarse a cabo mediante simulación o de forma analítica mediante ecuaciones matemáticas.

A continuación se realiza el modelado del sistema mediante ecuaciones matemáticas para describir el comportamiento físico de la planta y con ello ver la dinámica de ésta.

Realizar este análisis permite:

- Conocer la evolución del sistema ante distintas entradas de control y perturbaciones.

- Y diseñar los algoritmos de control que permiten estabilizar y mejorar las características dinámicas del sistema.

Para la realización del modelado se han de promediar, linealizar y perturbar para obtener el sistema invariante en el tiempo, mediante al implementación de las técnicas obtendremos los siguientes modelos: modelo conmutado, modelo promediado y modelo en pequeña señal (ver Figura 2.3).

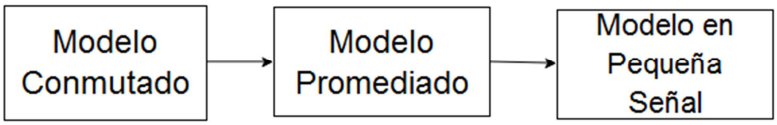


Figura 2.3: Tipos de Modelado

El circuito conmutado es circuito del convertidor.

Tras promediar obtenemos el modelo promediado que elimina la información del rizado, quedándonos solo con los valores medios de tensión o corriente de los elementos no lineales del sistema. A pesar de haber perdido la información del rizado de conmutación no tenemos un sistema lineal todavía.

Para obtener el sistema lineal se tiene que linealizar y perturbar el modelo promediado en torno a un punto de trabajo, y tras aplicar estas técnicas obtendremos el modelo en pequeña señal. Y ahora si tenemos un sistema lineal, en el cual poder estudiar la respuesta en frecuencia del sistema y ver a si su estabilidad.

2.2.1 Modelado de la Planta

Como se explicó anteriormente durante el modelado de la planta hay que estudiar los componentes no lineales del sistema. En la Figura 2.4 se observan las intensidades y tensiones de la bobina, el interruptor y el diodo respectivamente. A partir de estas figuras se obtienen las ecuaciones que rigen el comportamiento de cada uno de estos componentes (modelo conmutado):

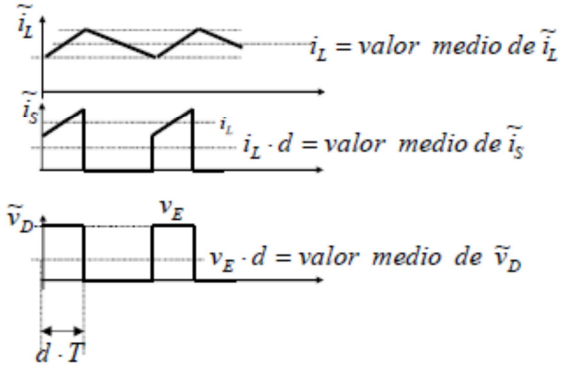


Figura 2.4: Gráficas de intensidades y tensiones.

A continuación se realiza un promedio de la intensidad y tensión. Mediante la técnica del promediado se pretende reproducir el comportamiento más significativo del sistema mediante el uso de aproximaciones. De esta forma se mejora la compresión del sistema proporcionando una visión física más fácil de entender y facilitando un buen diseño del control. Durante este proceso se elimina la información del rizado de conmutación.

$$i_S = i_S(t) = \frac{1}{T}\int_0^{dT} \tilde{i_S}(\tau)d\tau = i_L \cdot d$$

$$V_D = V_D(t) = \frac{1}{T}\int_0^{dT} \widetilde{V_D}(\tau)d\tau = V_E \cdot d$$

Para calcular el modelo en pequeña señal y poder realizar un análisis de estabilidad se linealiza y se perturba. Considerando solo pequeñas perturbaciones en el entorno de un punto de trabajo se obtiene las siguientes ecuaciones:

$$\widehat{i_S} = \left.\frac{\partial i_S}{\partial d}\right|_{i_L=I_L} \cdot \hat{d} + \left.\frac{\partial i_S}{\partial i_L}\right|_{d=D} \cdot \widehat{i_L} = I_L \cdot \hat{d} + D \cdot \widehat{i_L}$$

$$\widehat{v_D} = \left.\frac{\partial v_D}{\partial d}\right|_{v_S=V_S} \cdot \hat{d} + \left.\frac{\partial v_D}{\partial v_E}\right|_{d=D} \cdot \widehat{v_S} = V_S \cdot \hat{d} + D \cdot \widehat{v_S}$$

Para calcular las variaciones en pequeña señal de la tensión de salida, $\hat{v}_o$, con respecto al ciclo de trabajo, $\hat{d}$, solo se considera la perturbación en el ciclo de trabajo y por tanto se consideran nulas las variaciones de la tensión de entrada, $\widehat{v_S}$:

$$\hat{v}_o = V_S \cdot \hat{d} \cdot \frac{1}{1 + \frac{L}{R} \cdot S + L \cdot C \cdot S^2}$$

Finalmente se obtiene la función de transferencia de la planta:

$$G_{vd} = \frac{\hat{v}_o}{\hat{d}} = V_S \cdot \frac{1}{1 + \frac{L}{R} \cdot S + L \cdot C \cdot S^2}$$

Este procedimiento es tedioso y puede llegar a ser muy complejo en función de la topología del convertidor. En estos casos resulta interesante poder caracterizar la planta mediante la medida de respuesta en frecuencia.

2.3 Medida de la respuesta en frecuencia

Un aspecto muy importante para conseguir un buen lazo de control ya sea digital o analógico, es garantizar la estabilidad del sistema, para ello se tendrá que hacer un estudio de medida de respuesta en frecuencia.

La respuesta en frecuencia es una representación de la respuesta del sistema a entradas sinusoidales a frecuencia variable. Esta respuesta pude representarse mediante dos métodos [4],[5]:

- **Diagrama de bode**: Mediante esta representación obtenemos dos gráficas para el análisis de la estabilidad (ver Figura 2.5). En las cuales obtenemos información de la amplitud de salida y en la otra del desfase de salida respecto a la entrada del sistema, respectivamente.

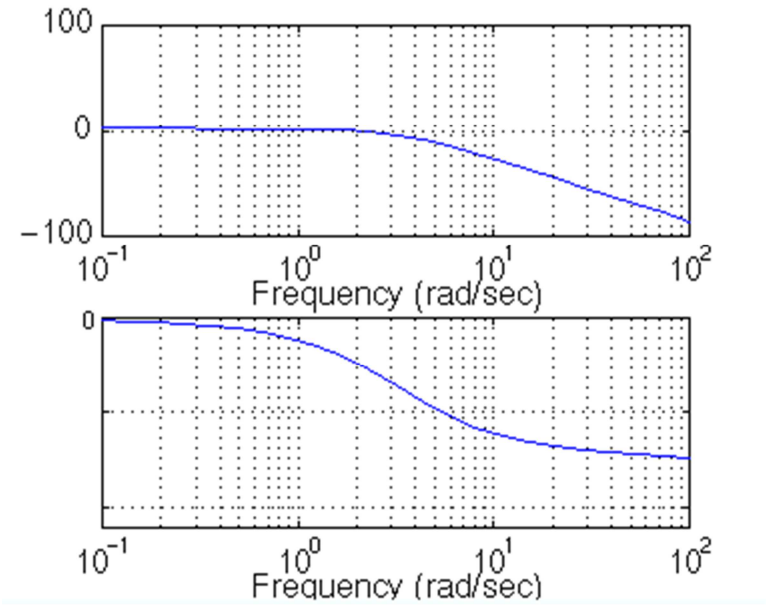


Figura 2.5: Diagrama de Bode

- **Diagrama de Nyquist:** En este diagrama se relaciona el modulo y la fase en un diagrama polar. Se puede estudiar la estabilidad del sistema en lazo cerrado sin necesidad de conocer la ecuación matemática que define el sistema. Ver Figura 2.6

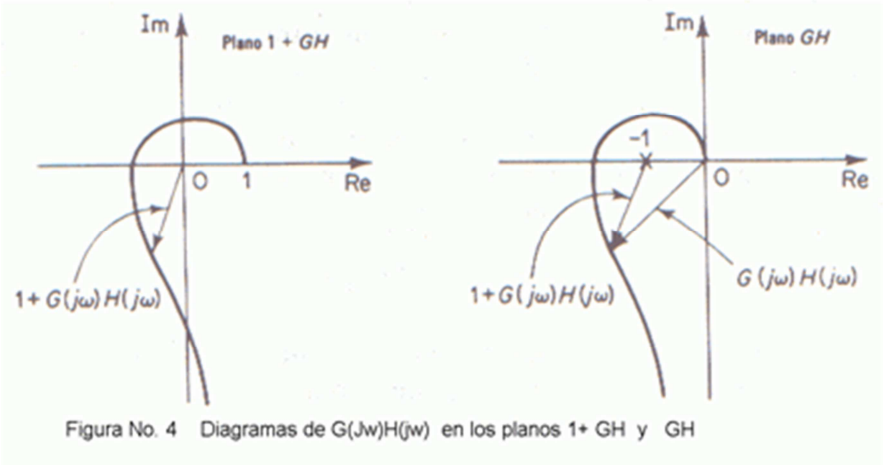


Figura 2.6: Diagrama de Nyquist

2.4 Descripción de los bloques del sistema Digital

Una vez obtenido el lazo de control analógico, se procede a la descripción de los bloques digitales y ver como se relacionan los bloques del lazo digital y del analógico.

El lazo digital se compone de un convertidor analógico-digital (para discretizar la señal de salida), el regulador (que está formado por la ecuación en diferencias) y por último el bloque para la generación de la DPWM. Ver Figura 2.7

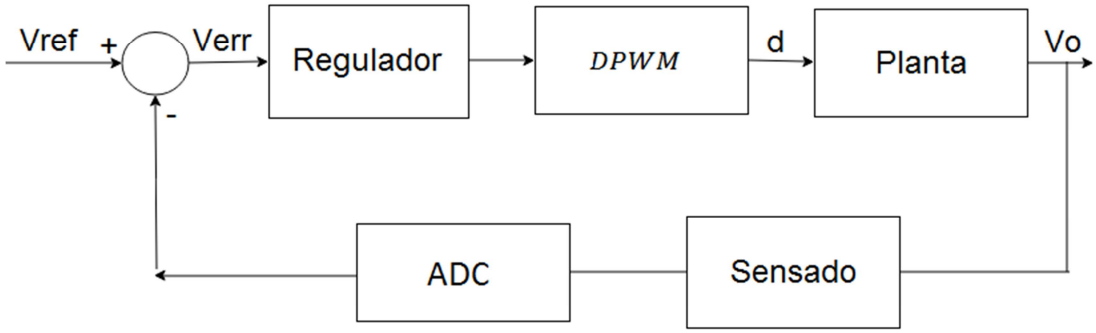


Figura 2.7: Esquema de bloques del sistema con el control digital

Para poder comparar los dos tipos de lazo se tiene que tener en cuenta las ganancias que introducen los bloques del lazo digital y que no están en la implementación analógica. Por ejemplo, tanto los microprocesadores como las FPGAs añaden ganancias al realizar el proceso de conversión del ADC y del DPWM, por lo que hay que calcular cuales son y tenerlas en cuenta a la hora de hacer el lazo analógico para que sea lo más parecido posible al digital.

2.4.1 Bloque ADC

Como se ha explicado anteriormente el bloque ADC introduce una ganancia en el sistema de realimentación (lazo de control). La ganancia del conversor A/D, para el caso del microprocesador que se va a utilizar para la realización de este proyecto vienen dadas por las siguientes ecuaciones donde las variables Vref+ y Vref-, son parámetros que utiliza el microprocesador para la conversión de analógico a digital y se especificarán en el apartado 3.3.2.1.2 del capítulo 3.

$$resolucion_{adc} = \frac{V_{ref+} - V_{ref-}}{2^n}$$

$$G_{ADC} = \frac{1}{resolucion_{adc}} = \frac{1}{\frac{V_{ref+} - V_{ref-}}{2^n}}$$

2.4.2 Bloque DPWM

Como se ha explicado anteriormente el bloque DPWM introduce una ganancia en el sistema de realimentación (lazo de control). La ganancia de este bloque es:

$$G_{DPWM} = \frac{1}{2^n}$$

Siendo n en ambos casos el número de bits.

2.4.3 Regulador del lazo de control digital

Una vez que se ha calculado el valor de las ganancias, hay que proceder a calcular el regular del lazo digital, dicho regulador estará formado por la ecuación en diferencias que define el sistema.

El cálculo de la ecuación en diferencias es fundamental ya que de ella depende el valor del ciclo de trabajo que se va a obtener, por lo que es importante no perder información en el redondeo y ser lo más precisos posible.

Para hallar la ecuación en diferencias del lazo de control digital se necesita pasar la ecuación del regulador del plano continuo (S) al plazo discreto (Z). Para ello se discretizan los parámetros de la función de transferencia H(S).

2.4.3.1 Discretización y obtención de la ecuación en diferencias

En este apartado se va a explicar de forma genérica como efectuar la discritización de un sistema continuo mediante la herramienta Matlab y como obtener posteriormente la ecuación en diferencias.

Para discretizar la ecuación como se ha mencionado anteriormente se utiliza la herramienta de *Matlab*.

Otro parámetro muy importante a la hora de discretizar es el tiempo que transcurre entre dos actualizaciones del ciclo de trabajo, T_s. Los motivos por los que es necesario escoger un tiempo adecuado son los siguientes:

- T_s alto: si cogemos un tiempo muy alto, obtendremos un comportamiento dinámico pobre, que puede dar lugar a que el control no actúe tan rápido como el sistema requiere.

- T_s bajo: si cogemos un tiempo muy pequeño se requiere una alta precisión en los coeficientes, ya que en caso contrario acumularíamos mucho error al obtener el ciclo de trabajo.

Una buena referencia para la elección del tiempo de discretización es el periodo de muestreo, que da una aproximación del tiempo adecuada para el sistema.

La ecuación que representa la función de transferencia (FDT) del sistema continuo será la siguiente:

$$H(s) = \frac{Y(s)}{X(s)} = \frac{b_0 \cdot s^n + \cdots + b_{n-1} \cdot s + b_n}{a_0 \cdot s^n + \cdots + a_{n-1} \cdot s + a_n}$$

En el caso un de regulador tipo 3 se tiene una FDT de orden 3, y por tanto la ecuación queda de la siguiente forma:

$$H(s) = \frac{Y(s)}{X(s)} = \frac{b_0 \cdot s^3 + b_1 \cdot s^2 + b_2 \cdot s + b_3}{a_0 \cdot s^3 + a_1 \cdot s^2 + a_2 \cdot s + a_3}$$

Después de discretizar los parámetros con *Matlab,* se obtiene la función de transferencia en discreto:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 \cdot z^3 + b_1 \cdot z^2 + b_2 \cdot z + b_3}{a_0 \cdot z^3 + a_1 \cdot z^2 + a_2 \cdot z + a_3}$$

Para conseguir que la función de transferencia dependa de los valores anteriores, se multiplican denominador y numerador por z^n , con un valor de $a_0 = 1$. Se obtiene la siguiente ecuación:

$$H(z) = \frac{Y(z^{-1})}{X(z^{-1})} = \frac{b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2} + b_3 \cdot z^{-3}}{1 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2} + a_3 \cdot z^{-3}}$$

Esta es la ecuación general del sistema discretizado. El siguiente paso es convertirla a la ecuación en diferencias para implementarla en un sistema digital. La ecuación en diferencias para un regulador tipo 3 se muestra a continuación:

$$d_n = b_0 \cdot e_n + b_1 \cdot e_{n-1} + b_2 \cdot e_{n-2} + b_3 \cdot e_{n-3} - a_1 \cdot d_{n-1} - a_2 \cdot d_{n-2} - a_3 \cdot d_{n-3}$$

Donde e_n es el error actual y d_n es el ciclo de trabajo actual.

Otro parámetro a tener en cuenta a la hora de diseñar el control es el retraso introducido por la implementación del sistema de control en digital. Esto es debido a que la implementación digital del lazo es más lenta que la implementación analógica, ya que el sistema digital requiere un tiempo para que el bloque ADC realice la conversión con éxito, posteriormente tiene que calcular la ecuación en diferencias lo que puede introducir un retraso en función de cómo se implemente y una vez obtenido el resultado se procede a la obtención de la señal DPWM. Este retraso afecta a la fase del lazo y puede provocar que la fase disminuya hasta producir inestabilidad en el sistema.

Capítulo 3 Diseño e Implementación digital del lazo de control

Una vez conocidos todos los requisitos necesarios para realizar un lazo de control adecuado, se procede a su diseño y a su implementación.

Para ello en primer lugar es necesario conocer bien las características de los elementos que se van a utilizar en este trabajo y que son:

- Convertidor PTD08A010W

  Se ha sido elegido este convertidor ya que tiene accesible el terminal de control, lo que permite que nos centremos sólo en el diseño e implementación del lazo de control

- Microcontrolador PIC18F2525

  Ha sido elegido el PIC para la implementación del lazo de control debido a su bajo coste, disponibilidad de herramientas de fácil uso para su programación y depuración y a que incluye los periféricos necesarios para los requisitos del sistema. En concreto se ha usado el PIC18F2525 debido a que se disponía de unidades en la universidad.

Una vez conocidas todas las especificaciones de dichos componentes se procede al diseño del lazo y posteriormente a su implementación.

3.1 Descripción del convertidor

Para poder realizar el lazo de control del convertidor el primer paso es estudiar las características de dicho convertidor.

El convertidor que se va a estudiar es el PTD08A010W de Texas Instruments (Ver Figura 3.1).

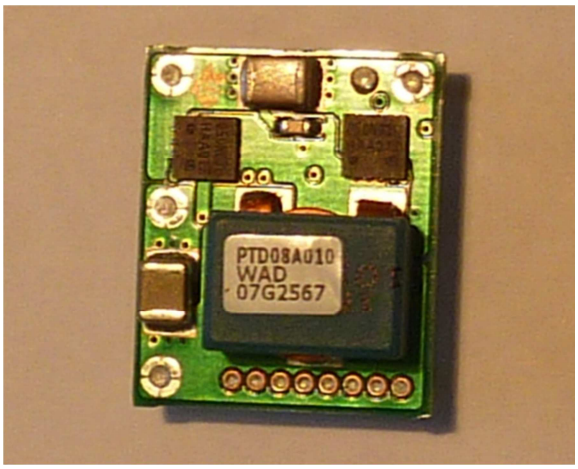


Figura 3.1: Convertidor PTD08A010W

3.1.1 Características del convertidor

Para conocer las características del convertidor hay que acudir a la hoja de catálogo que facilita el fabricante [7].

Los datos más significativos son las especificaciones de los valores de entrada y salida que podemos obtener con el convertidor, el rango de temperaturas, etc.

Los datos que se obtiene a partir de la hoja de características proporcionada por el fabricante son los siguientes:

- $V_{Imin} = 4.75\,V$ y $V_{Imax} = 14\,V$
- $V_{omin} = 0.7\,V$ y $V_{omax} = 3.6\,V$

Los valores de los condensadores, que son necesarios añadir para el correcto funcionamiento del convertidor, son los siguientes:

- Condensadores de entrada: uno de 330 µF y otro de 22 µF,
- Condensadores de salida: uno de 330 µF y de 47µF.

Los condensadores de 330 µF no son necesarios si se trabaja a una frecuencia mayor de 500KHz. Pero como en nuestro caso vamos a trabajar a una frecuencia de 300KHz es necesario poner todos los condensadores.

Las resistencias parasitas de todos los condensadores tienen un valor de 1.5mΩ.

3.1.2 Patillaje del convertidor

Es necesario analizar cada uno de los terminales del convertidor. La información sobre ellos está recogida en la página 5 de la hoja de características del fabricante (ver Figura 3.2 y

Tabla 3.1).

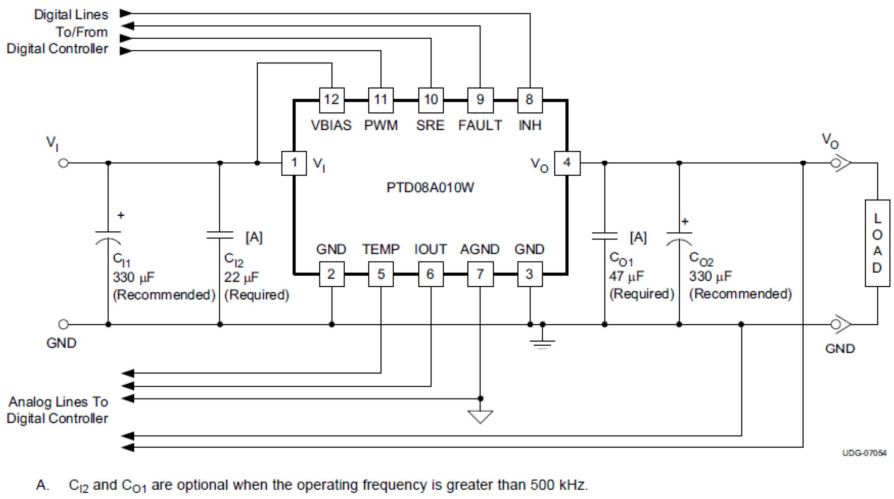


Figura 3.2: Patillaje del convertidor

Nº	Nombre	Descripción
1	V_I	Terminal para la tensión de entrada.
2 3	GND	Terminal de tierra común para V_I y V_o.
4	V_o	Terminal para la tensión de entrada.
5	TEMP	El nivel de voltaje de este terminal representa la temperatura del módulo.
6	IOUT	El nivel de voltaje de este terminal representa la intensidad de salida media del módulo.
7	AGND	Es la referencia de 0V para las señales de entradas.
8	INH	Este pin que inhabilita la salida del módulo si está a nivel bajo, requiere una resistencia de Pull-up conectada a 3.3 o 5V para que no sea usado
9	FAULT	Pin que limita la corriente al convertidor
10	SRE	Pin que hace que el módulo sea un sumidero de corriente si está a nivel alto, y si está a nivel bajo actuara como fuente de corriente.
11	DPWM	Pin de entrada para la señal DPWM
12	VBIAS	Para un rendimiento óptimo se debe conectar VBIAS a V_I.

Tabla 3.1: Descripción del patillaje del convertidor

A partir de esta información se obtiene el esquema general del convertidor, con el hardware necesario para su funcionamiento, como se muestra en la Figura 3.3

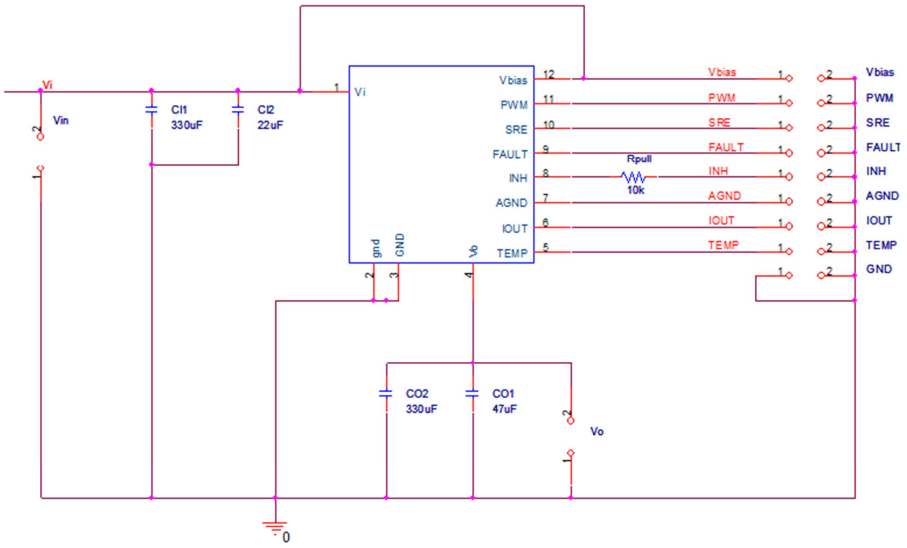


Figura 3.3: Esquema general del convertidor

3.2 Características del hardware a utilizar:

Antes de implementar el lazo del control se realiza un análisis de las características del PIC utilizado en este trabajo. Este análisis servirá de base para los cálculos posteriores que se realizan en el capítulo.

Una forma de realizar el control digital es mediante un microcontrolador o DSP (Digital Signal Processor). Se trata de dispositivos programables que ejecutan un código de forma secuencial. Dicho código puede ser programado en C o en código maquina (ensamblador), la diferencia entre un lenguaje y otro es la velocidad de procesamiento, que es menor si se programa en ensamblador.

Los microcontroladores además de la unidad de procesamiento incluyen distintos periféricos en función del modelo que sea. Para el caso del control de convertidores es muy útil que el microcontrolador incluya los siguientes periféricos:

- Conversor analógico – digital (ADC).
- Módulo específico para la generación de señales PWM (DPWM)
- Temporizadores (*timers*).

Estos bloques son importantes para la implementación del control digital. Del módulo ADC dependerá la capacidad para detectar cambios en la salida del

convertidor con mayor o menor precisión. Este aspecto es importante ya que cuando se trata de tener una magnitud controlada en un convertidor es imprescindible conocer en todo momento el valor de esa magnitud, ya sea corriente o tensión, con una precisión suficiente y con capacidad para detectar cambios bruscos en ellas.

Del mismo modo el módulo DPWM se encarga de actuar sobre los semiconductores del convertidor con rapidez y precisión para adecuar el ciclo de trabajo si se producen alteraciones en el sistema.

3.2.1 Microcontrolador PIC 18F2525

Para la realización de este proyecto se escoge un microcontrolador de microchip, en concreto el modelo 18F2525, ya que le tenemos disponible en la universidad.

A continuación se puede observar el diagrama de pines (Figura 3.4) de dicho microcontrolador:

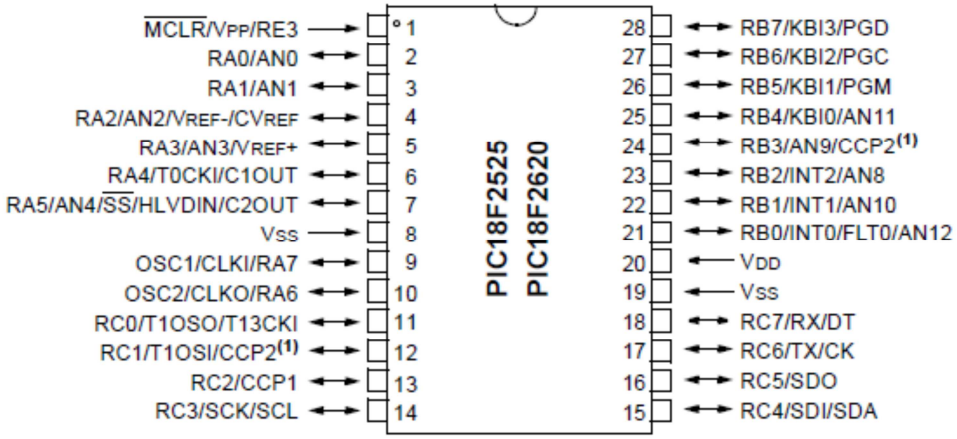


Figura 3.4: Diagrama de pines

Como se ha comentado con anterioridad, el PIC elegido para el proyecto tiene los bloques ADC, DPWM y Timers. Estos bloques son necesarios para poder realizar un buen control del convertidor.

3.2.1.1 Características del PIC

En primer lugar para que el PIC funcione hay que montar la arquitectura hardware necesaria en función de las especificaciones que se necesiten. La información necesaria para saber los componentes necesarios la encontramos en el data Sheets del PIC [8].

El primer componente a elegir es el oscilador externo. Este componente tiene gran importancia, ya que de él depende la velocidad de ejecución del microcontrolador. Debido a que la frecuencia de conmutación del convertidor es de 300KHz se ha escogido un oscilador de 25MHz, ya que es la máxima frecuencia que puede tener el oscilador.

Al elegir dicho oscilador tenemos que poner dos condensadores de 15pF, al poner un cristal externo. Dicha especificación viene en la siguiente tabla del manual (Ver Figura 3.5):

Osc Type	Crystal Freq	Typical Capacitor Values Tested:	
		C1	C2
LP	32 kHz	30 pF	30 pF
XT	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
	10 MHz	15 pF	15 pF
	20 MHz	15 pF	15 pF
	25 MHz	15 pF	15 pF

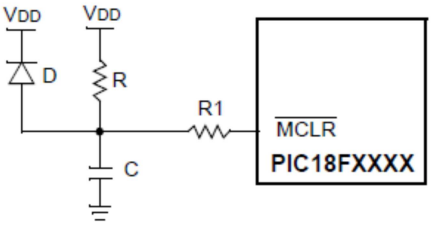
Capacitor values are for design guidance only.

Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application.

See the notes following this table for additional information.

Figura 3.5: Osciladores posibles para el PIC

Para evitar que por fluctuaciones en la tensión de alimentación se active el microcontrolador, ponemos el siguiente circuito en la entrada MCLR. Esta entrada filtra la tensión de entrada para evitar que se active con valores menores. El esquema de dicho circuito viene especificado en la Figura 3.6



Note 1: External Power-on Reset circuit is required only if the VDD power-up slope is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.

2: R < 40 kΩ is recommended to make sure that the voltage drop across R does not violate the device's electrical specification.

3: R1 ≥ 1 kΩ will limit any current flowing into MCLR from external capacitor C, in the event of MCLR/VPP pin breakdown, due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS).

Figura 3.6: Circuito señal MCLR

El valor de la resistencia R elegido es de es de 470Ω, el valor de R1 es de 10KΩ y el valor del condensador es de 100nF.

El esquema final del circuito es el siguiente:

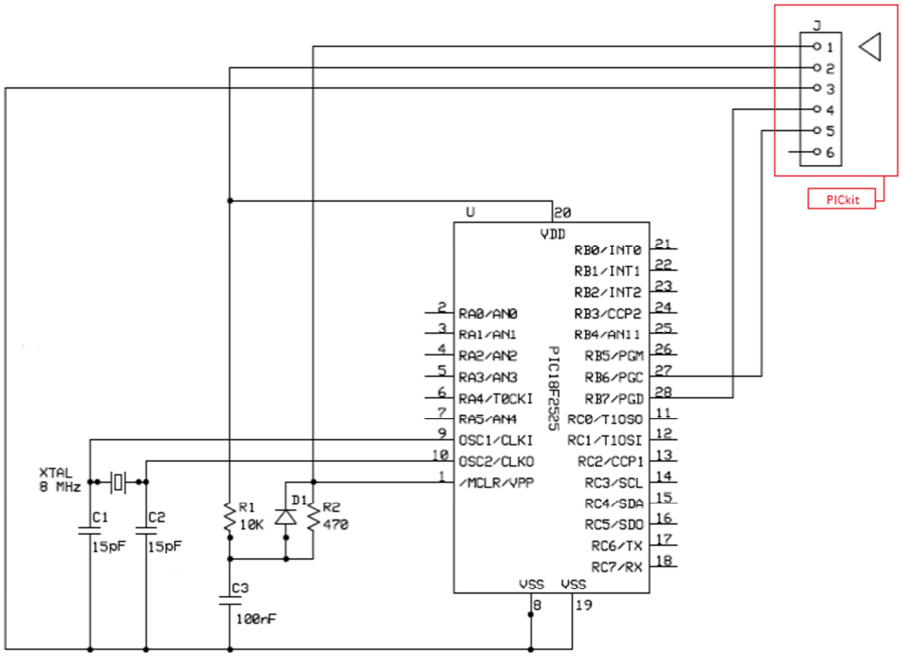


Figura 3.7: Esquema de conexión del PIC

En la figura se puede observar los componentes necesarios para el correcto funcionamiento del PIC así como a las patillas a las que tienen que ir conectados. A demás se observa donde tiene que ir conectado el Pickit que se utiliza para conectar el ordenador a mircrocontrolador para la programación y la depuración.

3.2.1.1.1 Bloque DPWM

El PIC tiene 3 módulos llamados CCP (Capture/Compare/PWM) que funcionan en combinación con los timers disponibles y se utilizan para implementar funciones relacionadas con temporización, como por ejemplo la generación de señales PWM. Con el módulo CCP configurado para generación de señales PWM se puede generar una señal con una resolución máxima de 10bits. En Figura 3.8 se muestran las características de la señal generada en función del valor de los registros a configurar. La estructura interna del periférico DPWM se muestra en la Figura 3.9.

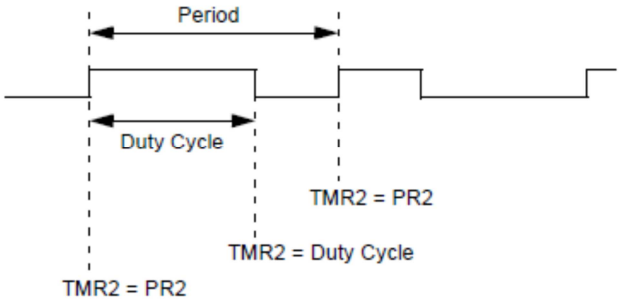


Figura 3.8: Esquema del funcionamiento del PWM

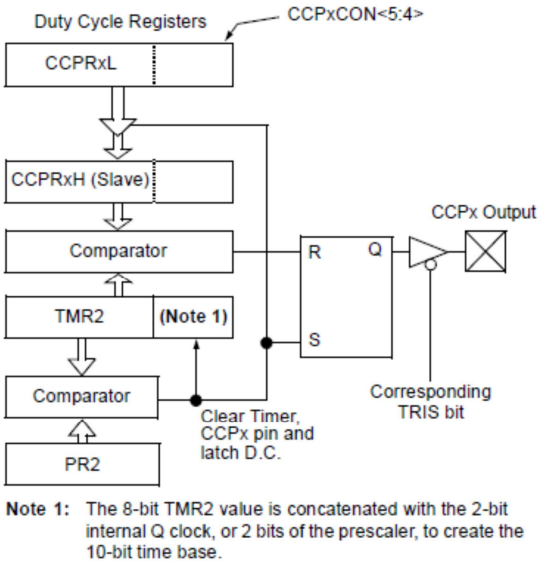


Figura 3.9: Diagrama de bloques DPWM

Como se puede observar en la Figura 3.9, configurar el PWM tenemos que usar el timer 2 y el registro PR2.

El primer paso es saber la resolución que se puede obtener con este módulo para la generación de la señal PWM, para ello aplicamos la siguiente formula:

$$PWM\ resolution = \frac{\log\left(\frac{Fosc}{Fpwm}\right)}{\log(2)} = \frac{\log\left(\frac{25{\cdot}10^{6}}{300{\cdot}10^{3}}\right)}{\log(2)} = 6.38$$

Al sustituir en la fórmula los valores particulares para el caso de estudio se obtiene un valor de $PWM\ resolution$ de 6.38. Por lo tanto, se necesitan 7 bits para cubrir todos los casos.

El siguiente parámetro a calcular es el *PR2*, que es el número de ciclos del *timer* que van a pasar en un periodo. Se desea una frecuencia de 300KHz, por tanto un periodo de 3.3µs. Hay que tener en cuenta que para el PIC18, la ejecución de una instrucción básica requiere de 4 ciclos de reloj, por lo que el periodo de un ciclo máquina (T_{ciclo}) es cuatro veces el periodo del oscilador (T_{osc}). Teniendo en cuenta lo anterior se calcula el tiempo de un ciclo del timer efectuando la siguiente operación:

$$T_{ciclo} = \frac{1}{Fosci} \cdot 4 = \frac{1}{25 \cdot 10^6} \cdot 4 = 160\ ns$$

$$n^{o}_{ciclo} = \frac{\frac{1}{300 \cdot 10^3}}{160 \cdot 10^{-9}} = 20\ Ciclos$$

El resultado no es exacto y es necesario truncar para que el periodo de la señal PWM que se genere no sea menor a 300KHz. Con 20 ciclos de funcionamiento del timer (el valor de *PR2* seria 19 ciclos) el periodo de la señal generada será de 312.5KHz.

Una vez efectuados estos cálculos tenemos todos los valores necesarios para la configuración de la función PWM

3.2.1.1.2 Bloque ADC

El modulo conversor analógico-digital (A/D) del PIC18F2525, para poder realizar la conversión adecuadamente necesita tener una referencia positiva y una negativa que pueden ser internas (alimentación a 5V y masa respectivamente) o externas conectándose a través de pines específicos (AN3 y AN2 respectivamente). En este trabajo se van a tomar las referencias internas.

Una vez conocidas las referencias vemos la resolución que va a tener nuestro ADC, puede tiene una resolución máxima de 10bit.

La resolución del convertidor A/D en el caso que nos ocupa va a ser de 6 bits ya que para que posteriormente se obtenga un ciclo de trabajo adecuado es necesario que se cumpla la siguiente condición:

$$V_e \cdot \Delta d < q_{ADC} \rightarrow 6 \cdot \frac{1}{2^7} < \frac{5}{2^n}$$

Para que se cumpla la ecuación el valor de n debe ser 6, por lo tanto nuestro A/D será capaz de leer datos de 78mV $\left(\frac{5}{2^6}\right)$. Cualquier variación de voltaje que está por debajo de ese valor pasará desapercibido para el convertidor.

En la siguiente figura podemos ver un esquema del funcionamiento del conversor A/D (Figura 3.10):

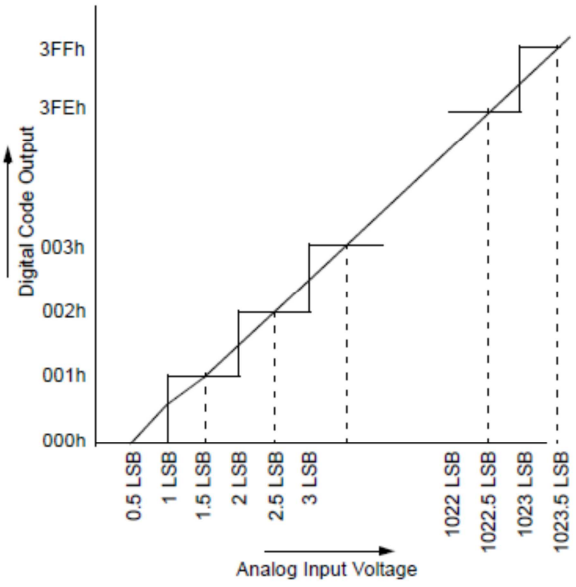
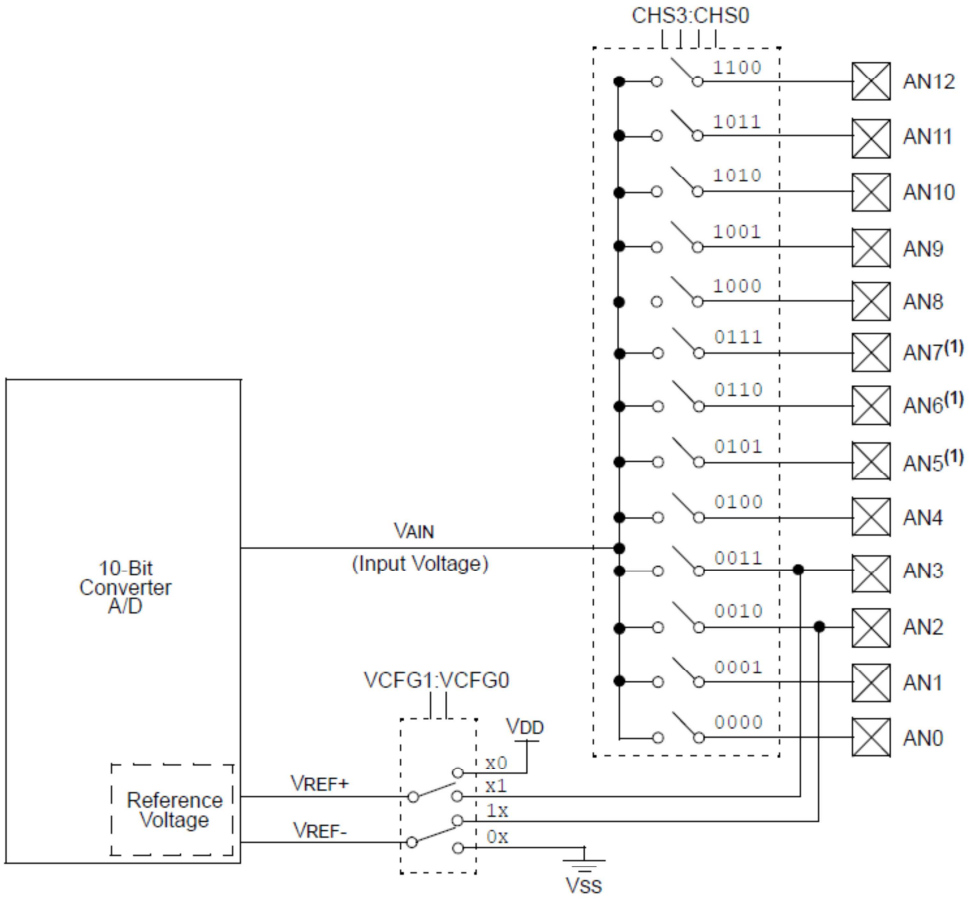


Figura 3.10: Esquema del funcionamiento del ADC

El diagrama de bloques del A/D es el correspondiente a la Figura 3.11.



Note 1: Channels AN5 through AN7 are not available on 28-pin devices.
2: I/O pins have diode protection to VDD and VSS.

Figura 3.11: Diagrama de bloques del ADC

3.3 Diseño del lazo de control

A lo largo de este apartado se explicará la metodología utilizada para la obtención de los parámetros necesarios para realizar el control del convertidor.

3.3.1 Definición del convertidor

Lo primero es definir en el programa *"Psim"* (herramienta que se va a utilizar para el análisis del lazo de realimentación) las características de nuestro convertidor. Para ello es necesario conocer los datos proporcionados por el

fabricante mediante la hoja de características del convertidor en cuestión, para ver en que rangos puede trabajar. Dichas características son las descritas en el apartado 3.1.1. Hemos elegido una tensión de entrada de 6V y una salida de 2V, la potencia del convertidor será de 10W.

El esquema del convertidor queda de la siguiente forma (ver Figura 3.12)

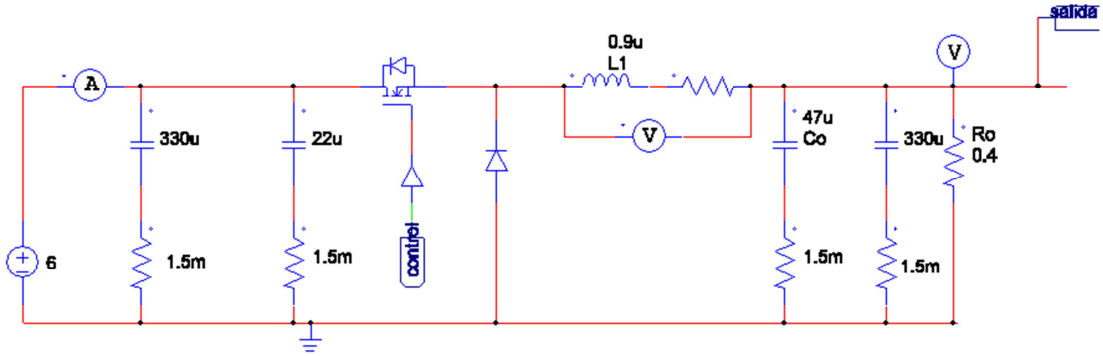


Figura 3.12: Esquemático del convertidor

Todos los datos de los componentes vienen dados por el fabricante excepto la resistencia de salida, que depende de la tensión que deseemos a la salida. Para calcular la resistencia que vamos a tener en la salida aplicamos las siguientes fórmulas:

$$P_o = V_o \cdot I_o \rightarrow P_o = \frac{{V_o}^2}{R_o}$$

$$R_o = \frac{{V_o}^2}{P_o} \rightarrow R_o = \frac{2^2}{10} = 0.4\ \Omega$$

Una vez obtenidos todos los datos del convertidor pasamos a realizar el lazo de control analógico para posteriormente calcular el equivalente en digital y poder implementarlo en el microcontrolador como se explicó en el apartado 1.2.

3.3.2 Lazo analógico

El segundo paso es implementar el lazo de control analógico para más tarde a partir del lazo en tiempo continuo el obtener el lazo digital (tiempo discreto) a implementar en el microcontrolador.

Para ello utilizamos la herramienta *"SmartCtrl"* para hacer el lazo en tiempo continuo. Esta herramienta permite definir de forma rápida y sencilla reguladores tipo 3, tipo 2, tipo 1 y PI, centrándose principalmente en su margen de fase y frecuencia de cruce del lazo abierto del sistema.

Tipos de regulador

Existen varios tipos de control que pueden ser utilizados para regular el convertidor: control tipo 1, control PI, control tipo 2 y control tipo 3

- **Tipo 1 o Single Pole:**

  Este control tiene un único polo en el origen. Se emplea ocasionalmente en sistemas de primer orden, sin ceros en el semiplano positivo.

- **PI o proporcional integral:**

  Este control tiene un polo en el origen y un cero. Se emplea para sistemas de primer orden.

- **Tipo 2:**

  Este control tiene un polo en el origen, un polo y un cero. Se emplea en sistemas de primer orden. La diferencia entre este y el PI es que al añadir el tipo 2 un polo, presenta atenuación a alta frecuencia.

- **Tipo 3:**

  Este control tiene un polo en el origen, dos polos y dos ceros. Se emplea en sistemas de segundo orden. Al Introducir dos ceros es capaz de proporcionar 180^{o} de fase, mientras que el resto proporcionan 90^{o}.

En este proyecto se utiliza un sistema de control tipo 3, ya que lo que se busca es obtener la máxima respuesta dinámica del sistema y por qué se tiene un sistema de segundo orden

Por tanto, el circuito resultante en el caso de un regulador analógico con el lazo de control tipo 3 será el siguiente

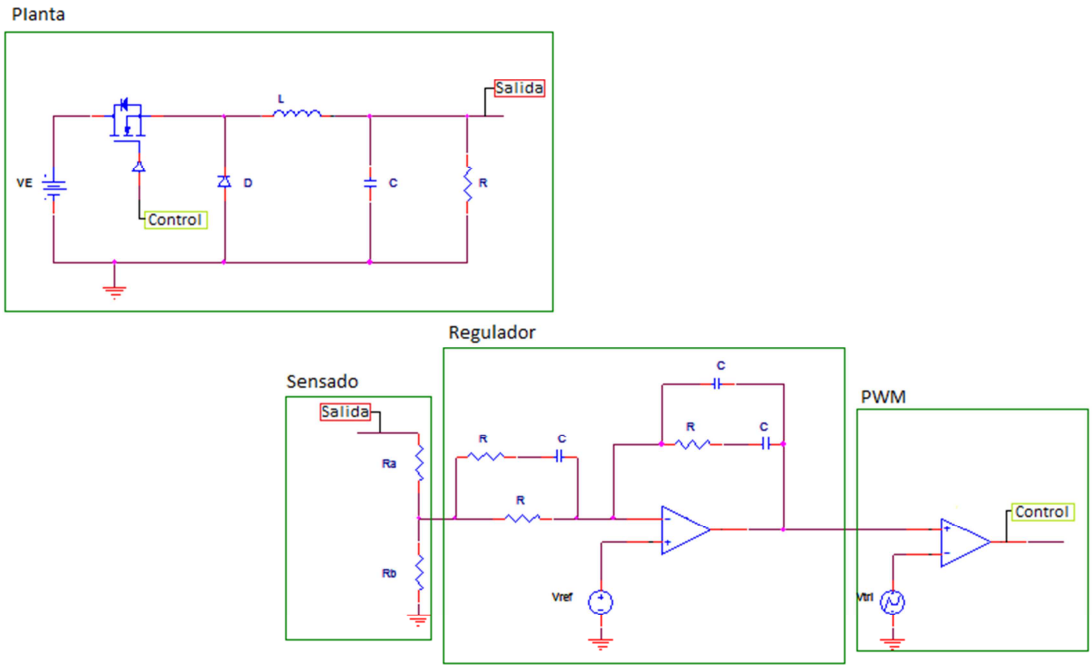


Figura 3.13: Esquema del convertidor reductor y lazo analógico

El proceso de diseño del control de un convertidor con *"SmartCtrl"* es el siguiente:

- Definición de la planta
- Definición del sensor
- Definición de los objetivos dinámicos (margen de fase y frecuencia de corte)
- Obtención del regulador y análisis del mismo.

Hacer bien el diseño del regulador analógico es clave ya que a partir de él se va a obtener el regulador digital. Por lo tanto, es muy importante que este regulador tenga las mismas características que el digital, para que la simulación sea lo más parecida posible una de otra. Esto implica que hay que tener en cuenta todos los factores que tienen los reguladores digitales y que por lo tanto afectan al modo de operar del control. Estos factores son los que se mencionan en el punto 2.3 del capítulo 2.

- Ganancias del microcontrolador.
- Retrasos que introduce el microcontrolador.

3.3.2.1 Definición de la planta

El programa *"SmartCtrl"* permite introducir la planta mediante topologías predefinidas (reductor, elevador, reductor elevador, Flyback, etc.), o importar su respuesta en frecuencia en formato numérico, ya sea desde un barrido en frecuencia de una simulación (ej. *AC sweep* del mismo PSIM) o mediante la medida experimental de la planta mediante un analizador de ganancia y fase.

En este caso se va a importar la respuesta en frecuencia del barrido en frecuencia que realiza mediante simulación, ya que debido a los retrasos que introduce el microcontrolador es necesario tenerlos en cuenta en la simulación, para que el sistema no se vuelva inestable. Si el sistema digital no tuviera retrasos se utilizarían las topologías predefinidas que nos proporciona la herramienta.

La elección de hacerlo de una forma u otra es muy importante, ya que como se explicó anteriormente si hay retrasos en el sistema y no se tienen en cuenta cuando se realice la puesta en marcha el sistema puede volverse inestable debido a que un retraso en el control implica una fase más baja.

Por lo tanto, para realizar el diseño del lado lo primero que se debe hacer es un barrido en frecuencia de la planta.

3.3.2.1.1 Barrido en frecuencia.

El barrido en frecuencia se realiza en el *"Psim"* mediante la simulación AC. Para ver cómo afecta el retraso a la planta se debe poner a trabajar al regulador con el valor del ciclo de trabajo necesario para que la salida sea la deseada. En este caso, asignando al convertidor un ciclo de trabajo de 0.33 y una entrada de 6V se obtiene una tensión de salida de 2V, que es la tensión de salida deseada para el sistema, es decir sin realimentación.

El retraso será introducido a la salida del ciclo de trabajo para hacer la simulación en frecuencia. El circuito estará en lazo abierto, es decir, sin realimentación ya que el convertidor estará trabajando con un valor del ciclo de trabajo constante.

El esquema de nuestro circuito para realizar el análisis en frecuencia teniendo en cuenta en retraso será el que se muestra en la Figura 3.14. El retraso introducido que vamos a tener es el tiempo de ejecución y calculo que necesita el PIC para poder obtener un nuevo ciclo de trabajo. La explicación detallada de este cálculo se presenta en el apartado 3.3.3.2

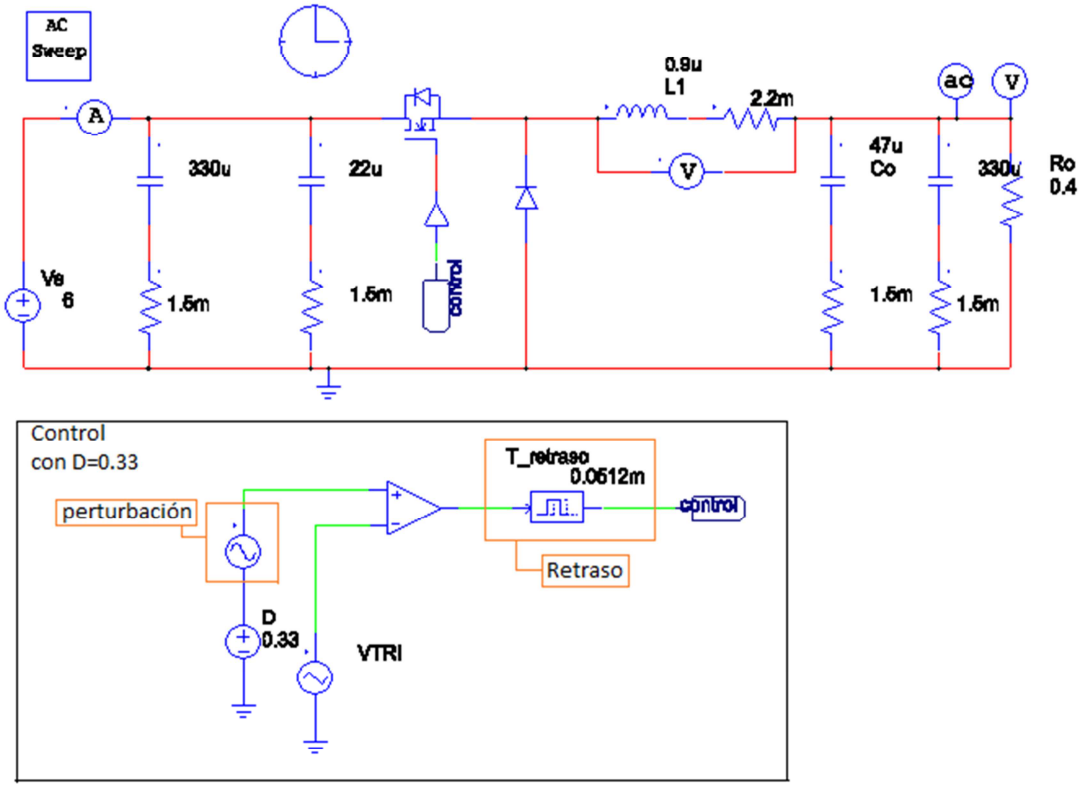


Figura 3.14: Circuito para la simulación en frecuencia

Como se puede observar, para ver la frecuencia de la planta se perturba en el ciclo de trabajo (control) y se mide la respuesta en frecuencia a la salida (ac). Mediante el bloque *AC Sweep* se indica cual es la fuente de perturbación y el rango de frecuencias que se quiere analizar, así como el número de puntos de paso de simulación.

Para ver cómo afecta el retardo introducido se hace una comparación del análisis en frecuencia con y sin retraso. En la Figura 3.15 la simulación de color rojo es la que no tiene retraso y la simulación de color azul tiene en cuenta el retraso.

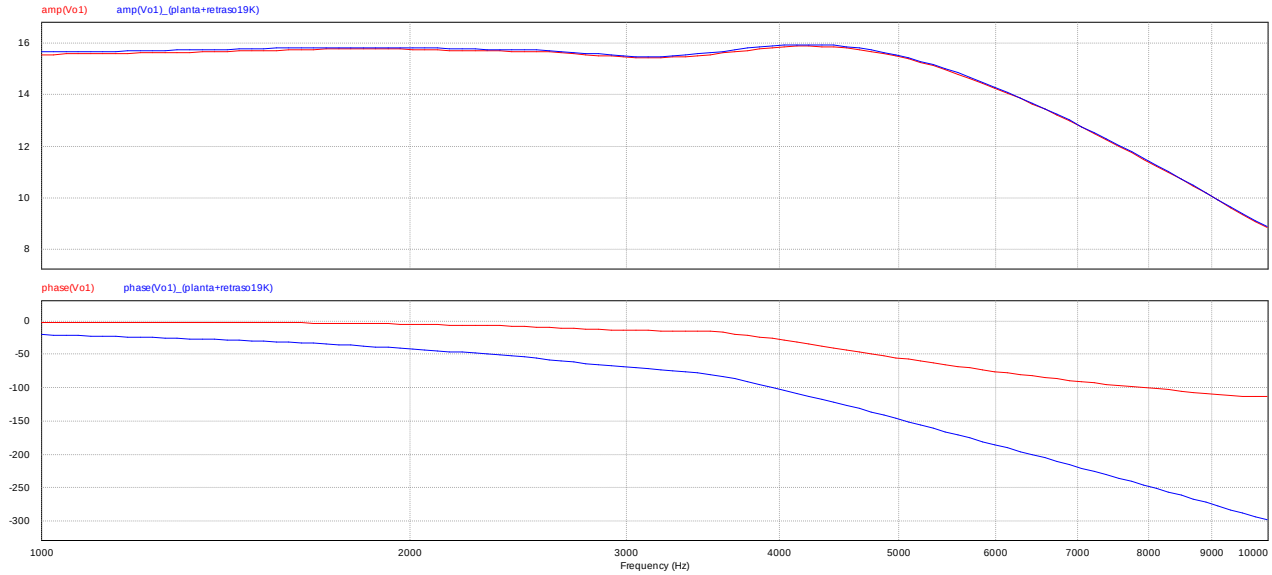


Figura 3.15: Comparación de la amplitud y la fase del sistema con y sin retraso

En esta grafica se puede observar como el retraso modifica la fase y no la ganancia. Al bajar tanto la fase si no se tiene en cuenta durante el diseño del lazo de control, el sistema se puede volver inestable.

3.3.2.1.2 Diseño de lazo de control analógico

Una vez que tenemos la simulación en frecuencia con el retraso incluido se importa la simulación al programa "*SmartCntrl"* para diseñar el lazo de control analógico siguiendo los siguientes pasos a través de las ventanas del programa:

Design → Importer transfer funcion → single loop → Voltage mode controlled.

A continuación se siguen los siguientes pasos:

- **Paso 1**:

El programa pide que se importe la simulación y se especifiquen los valores de la salida y la frecuencia de conmutación (2V y 300KHz) (ver Figura 3.16):

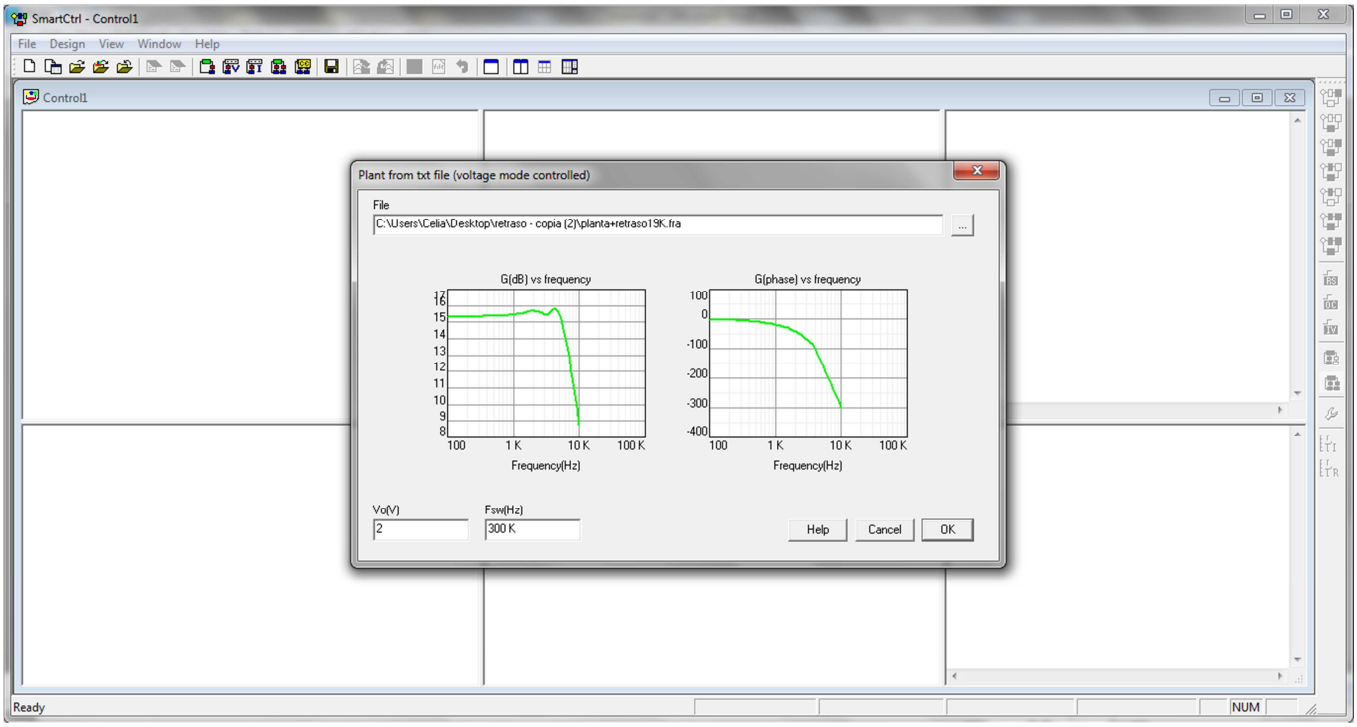


Figura 3.16: Captura de pantalla Smartctrl: introducción del analisis en frecuencia

- **Paso 2**:

En este paso se introducen los datos del sensor, tal y como se explicó anteriormente es muy importante que aquí se tenga en cuenta las ganancias del conversor A/D y del DPWM del microcontrolador ya que estas ganancias no existen en el lazo de control analógico.

Hay varias formas de introducir estas ganancias, una de ellas es calcular la ganancia total del control digital. Se diferencian los siguientes casos:

- Ganancia total menor que uno: en este caso se puede añadir dicha ganancia en la parte del sensor.

- Ganancia total mayor que uno: en este caso se debe añadir la ganancia en la parte del regulador.

- Ganancia del ADC menor que uno: en este caso se introduce la ganancia del ADC en el sensor y la del DPWM se debe introducir en el regulador.

En este caso la forma elegida para añadir las ganancias es calculándonos la ganancia total del sistema. A continuación se procede a dicho cálculo según la teoría explicada en el apartado 2.3 del capítulo 2:

- *Ganancia del conversor A/D:*

$$resolucion_{adc} = \frac{V_{ref+} - V_{ref-}}{2^n}$$

$$G_{ADC} = \frac{1}{resolucion_{adc}} = \frac{1}{\frac{V_{ref+}-V_{ref-}}{2^n}} = \frac{1}{\frac{5-0}{2^6}} = 12.8$$

- *La ganancia del DPWM :*

$$G_{DPWM} = \frac{1}{2^n} = \frac{1}{2^7} = \frac{1}{128}$$

- *Ganancia total del sistema:*

$$G_T = G_{DPWM} \cdot G_{ADC} = 12.8 \cdot \frac{1}{128} = 0.1$$

Al obtener una ganancia total menor que uno, se introduce dicho valor en la etapa del sensado tal y como se observa en la Figura 3.17.

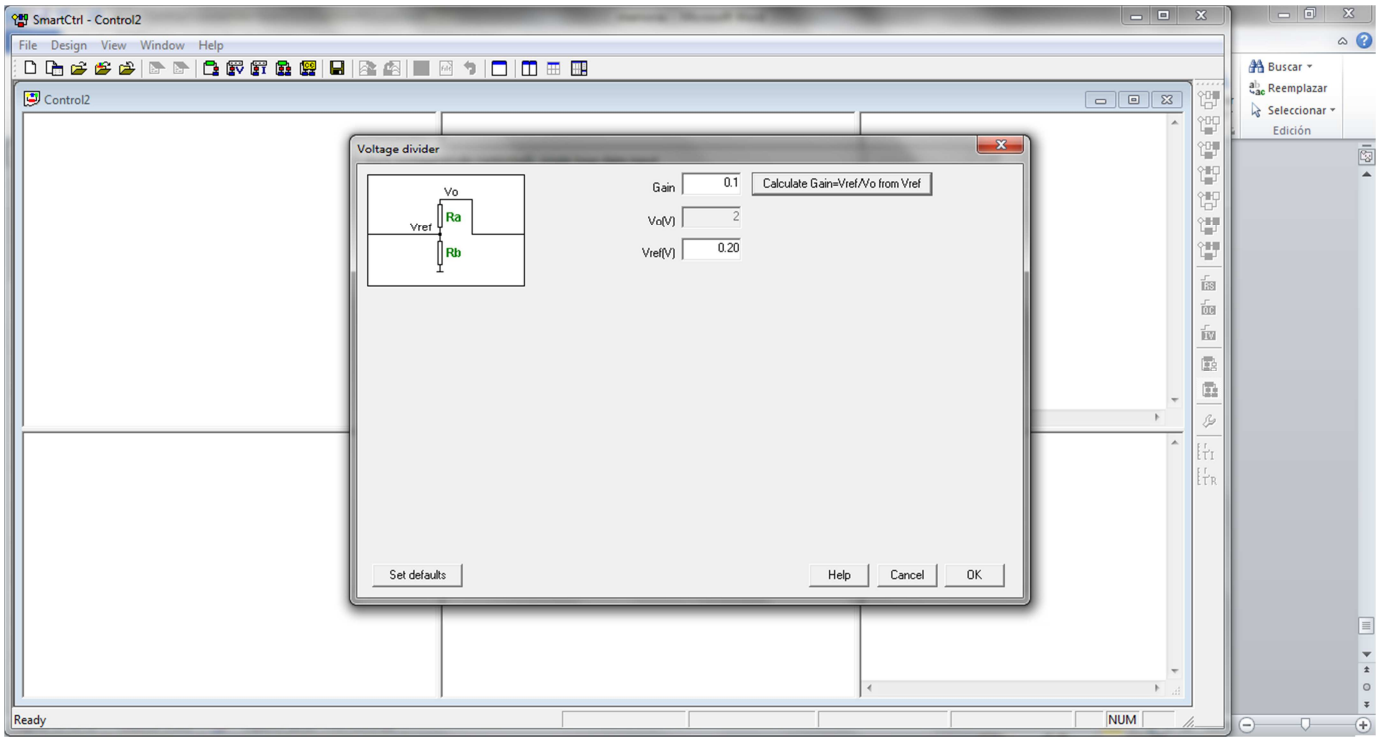


Figura 3.17: Captura de pantalla Smartctrl: sensado

- **Paso 3**:

Después de introducir los datos del sensor, se procede a introducir los datos correspondientes al bloque del regulador.

Si la ganancia no se ha introducido anteriormente la meteremos ahora en el campo *Gmod*. En este caso *Gmod* tendrá valor 1, porque ya ha sido tenida en cuenta en el bloque de sensado (Figura 3.18).

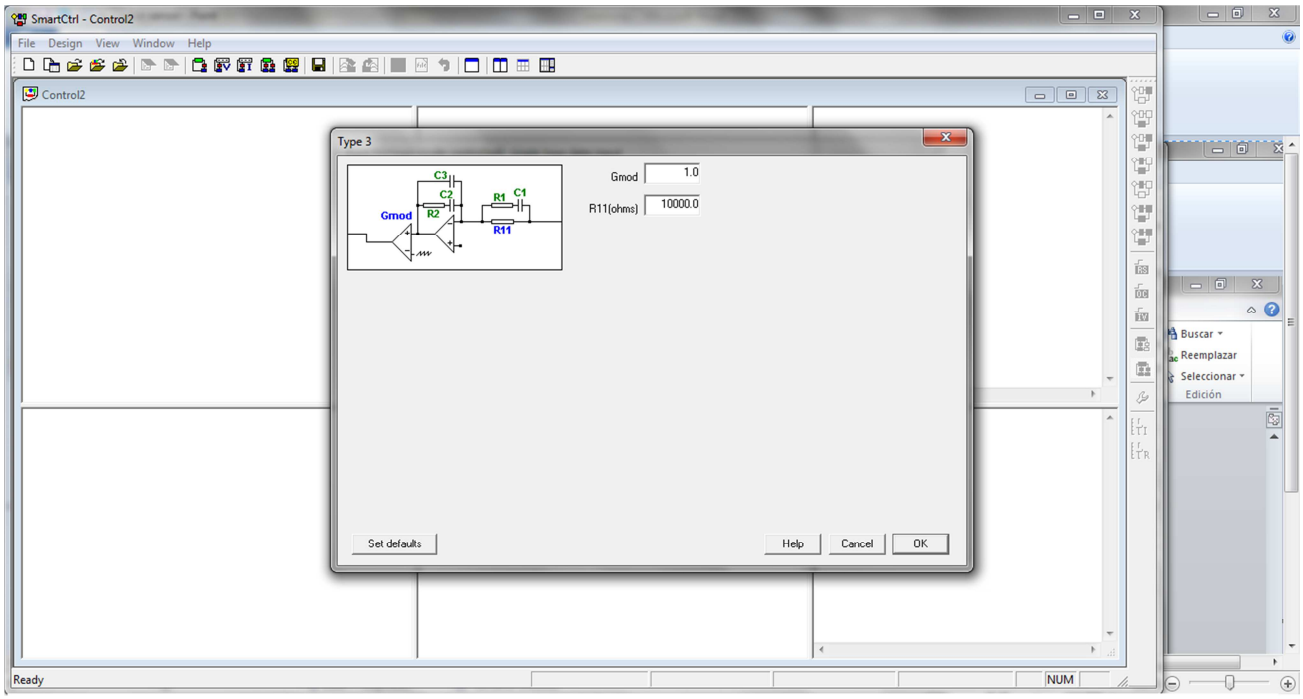


Figura 3.18: Captura de pantalla Smartctrl: regulador

- **Paso 4**:

El siguiente paso es elegir la frecuencia de cruce y la fase del sistema para que este sea estable.

Si elegimos una frecuencia alta tendremos una buena regulación dinámica, y si escogemos una frecuencia baja filtraremos el rizado de conmutación, por lo que en general lo adecuado es elegir al menos una frecuencia de cruce (fc) como mínimo de una década antes de la frecuencia de conmutación para así eliminar el rizado y tener una buena regulación dinámica.

En cuanto al margen de fase tiene que ser lo suficientemente bajo para una buena respuesta dinámica y lo suficientemente alto para proporcionar una respuesta amortiguada y seguridad frente a desfases adicionales en el convertidor. Un buen margen de fase suele estar comprendido entre los 30 y 90º.

En el caso de estudio al introducir un retraso tan grande se tiene que aumentar el margen de fase para compensar el efecto de dicho retraso.

La frecuencia de cruce será elegida para que el sistema sea estable. Para ver si el sistema es estable se recurre a los diagramas de *bode* y de *nyquits* que nos proporciona el propio programa "*SmartCtrl*" (ver Figura 3.19). Los valores escogidos para este caso en concreto son:

- fc = 221.82Hz
- margen de fase = 144º.

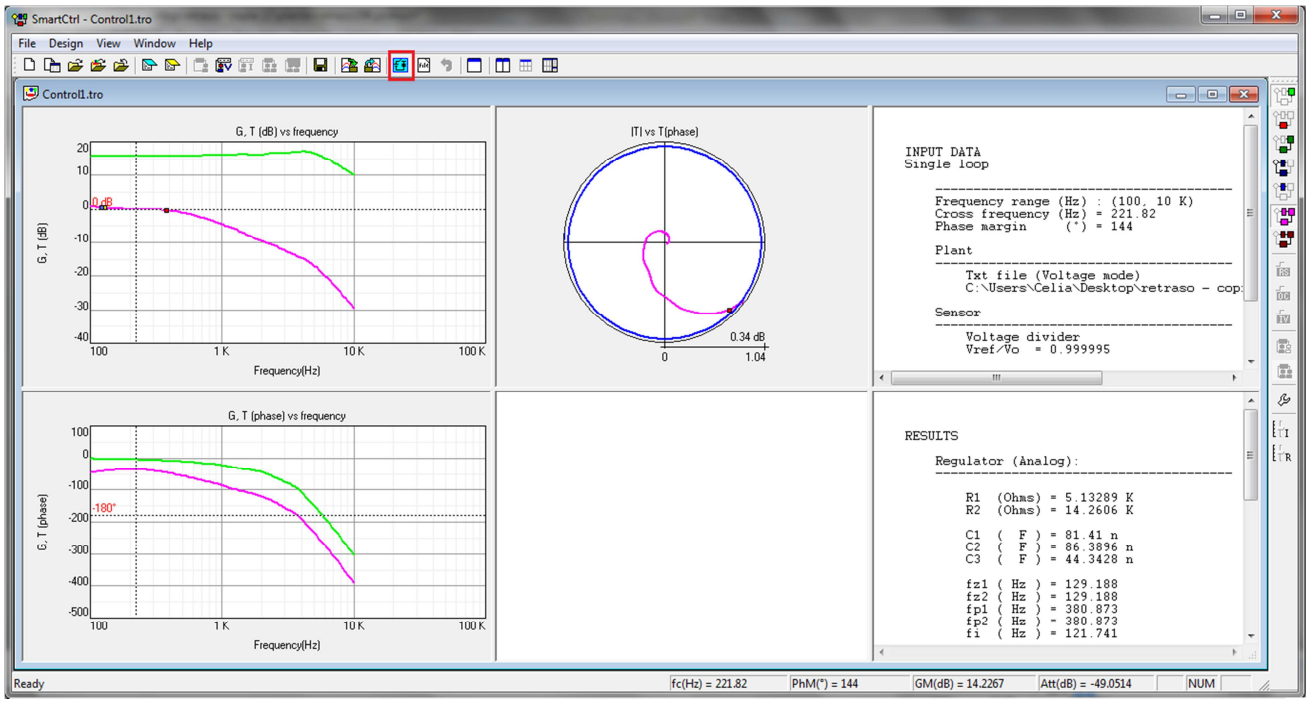


Figura 3.19: Respuesta en frecuencia del lazo

3.3.2.2 Sistema Analógico

Una vez elegidas la frecuencia de cruce y el margen de fase, se exporta el lazo de control al esquemático del programa "*Psim*". Para ello se utiliza el icono señalado en rojo de la figura anterior (Figura 3.19).

Cuando se exporta aparecen dos posibilidades:

- La primera seria importar el lazo a analógico
- La segunda exportarlo en el dominio de S.

Para comprobar el lazo diseñado es correcto se realiza la simulación con los dos casos, definiendo escalones a la salida de subida y bajada para comprobar que el sistema se estabiliza y por lo tanto el lazo actúa correctamente (Ver Figura 3.20)

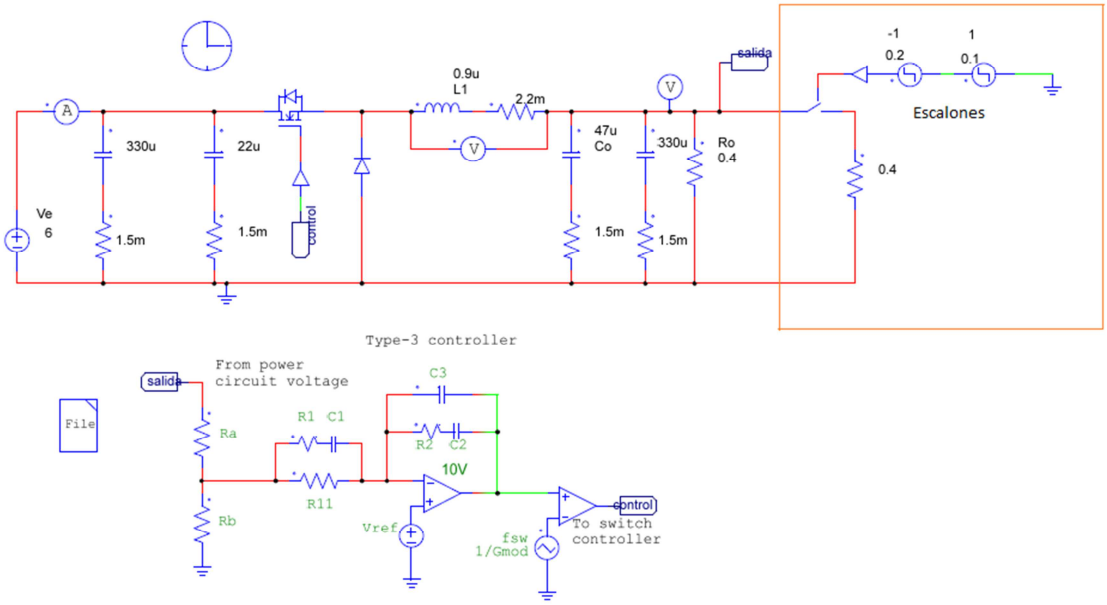


Figura 3.20: Imagen del Convertidor y el lazo analógico

Como se observa en la Figura 3.20 los escalones están puestos para un tiempo 0.1 y 0.2 s. En la simulación se puede ver como el sistema se estabiliza en el arranque y en los escalones (Figura 3.21)

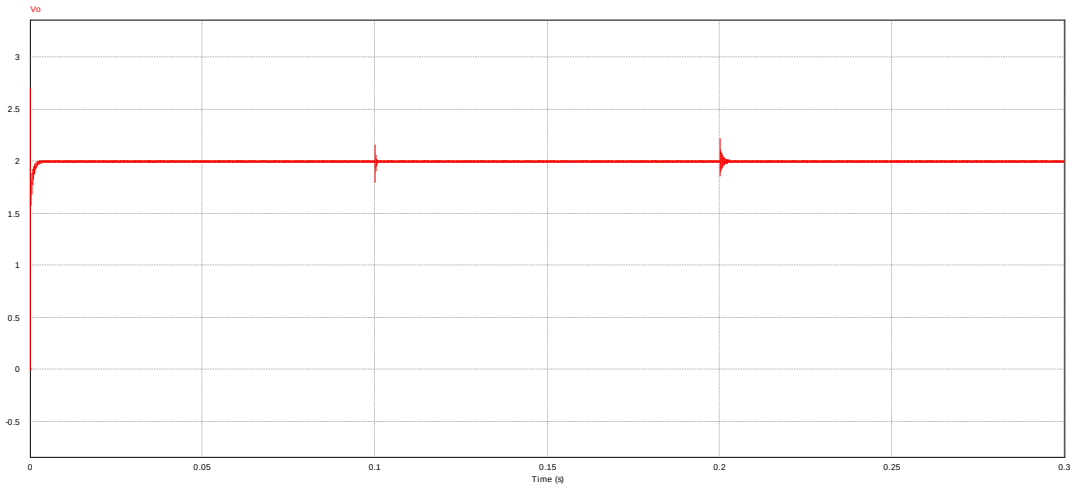


Figura 3.21: Simulación del sistema con el lazo analógico

Si se exporta el control en el domino de S, se puede observar que el sistema se estabiliza correctamente después de ambos escalones (ver Figura 3.22 y Figura 3.23)

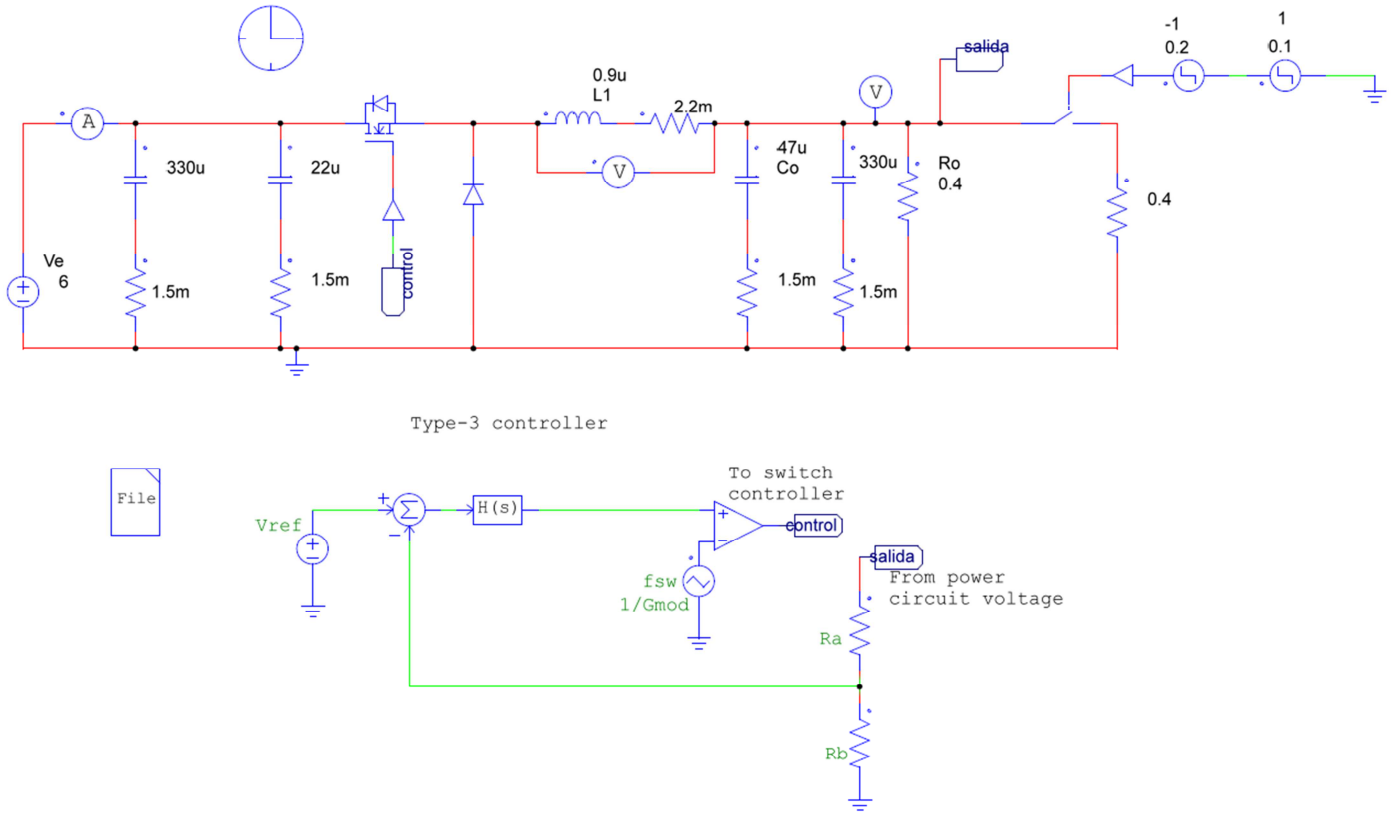


Figura 3.22: Sistema con el lazo de control H(s)

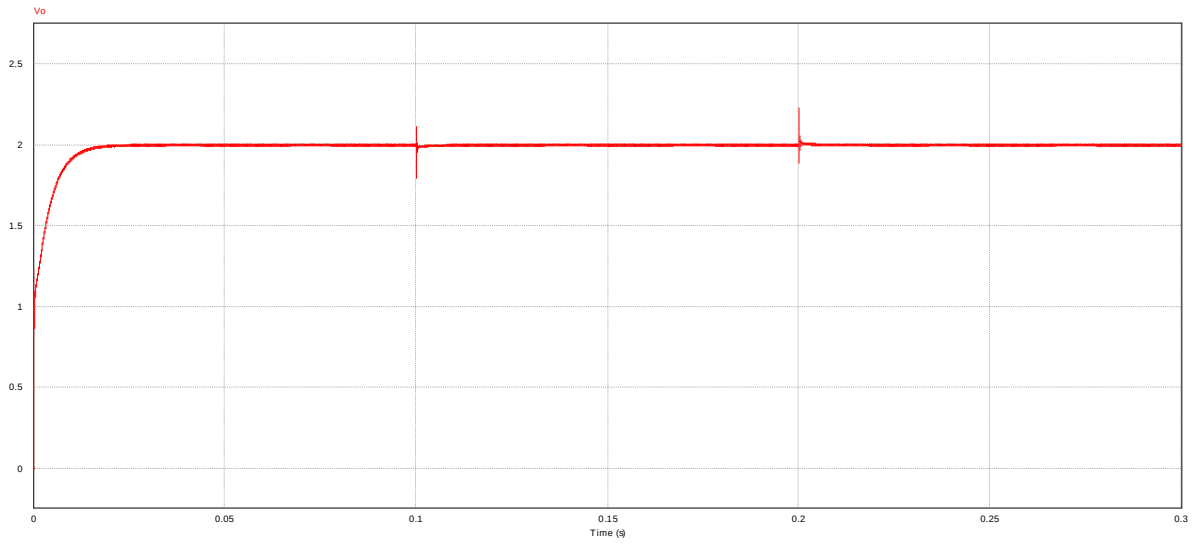


Figura 3.23: Simulación del sistema ante los dos escalones con el lazo H(s)

3.3.3 Calculo del sistema Digital

A continuación se procede a discretizar los parámetros del sistema continuo para obtener la ecuación en diferencias.

Para obtener la ecuación en diferencias el primer paso es conocer los valores de la Función de Transferencia continua. Estos valores son proporcionados la propia herremienta "*Smartcntrl*". Ver Figura 3.24.

```
RESULTS

    Regulator (Analog):
    ------------------------------------

        R1   (Ohms) = 5.13289 K
        R2   (Ohms) = 14.2606 K

        C1   (   F ) = 81.41 n
        C2   (   F ) = 86.3896 n
        C3   (   F ) = 44.3428 n

        fz1  ( Hz ) = 129.188
        fz2  ( Hz ) = 129.188
        fp1  ( Hz ) = 380.873
        fp2  ( Hz ) = 380.873
        fi   ( Hz ) = 121.741


        b2   ( s^2) = 1.51774e-006
        b1   ( s  ) = 0.00246394
        b0          = 1

        a3   ( s^3) = 2.28277e-010
        a2   ( s^2) = 1.09258e-006
        a1   ( s  ) = 0.00130732
        a0          = 0

    Sensor:
    ------------------------------------

        Ra   (Ohms) = 847.974 u
        Rb   (Ohms) = 169.594

        Pa  (Watts) = 117.928 n
        Pb  (Watts) = 0.0235855

    Loop performance parameters:
    ------------------------------------

        PhF  ( Hz ) = 3.77787 K
        GM   ( dB ) = 14.2267
        Atte( dB ) = -49.0514
```

Figura 3.24: Valores del sistema en H(s)

Una vez obtenidos los datos, se procede a su discretización mediante la herramienta *Matlab*, se discretizan tal y como se explicó en el apartado 2.3.4 del capítulo 2. Los comandos introducidos en *Matlab* son los siguientes:

```
>> num = [1.51774e-006 0.00246394 1]

num =0.0000    0.0025    1.0000

>> den = [2.28277e-010 1.09258e-006 0.00130732 0]

den =0.0000    0.0000    0.0013         0

>> gs = tf(num,den)

Transfer function:

1.518e-006 s^2 + 0.002464 s + 1
-------------------------------------------
2.283e-010 s^3 + 1.093e-006 s^2 + 0.001307 s

>> gz = c2d (gs, 16/312500, 'tustin')

Transfer function:

0.1575 z^3 - 0.1446 z^2 - 0.1572 z + 0.1449
-----------------------------------------
z^3 - 2.769 z^2 + 2.552 z - 0.7824

Sampling time: 5.12e-005
```

Figura 3.25: Comandos de Matlab

El tiempo de discretización empleado es el más próximo al tiempo de ejecución del microcontrolador y siendo múltiplo de la frecuencia de DPWM.

Por lo tanto los datos para la ecuación en diferencias son los siguientes:

$$b_0 = 0.1575$$

$$b_1 = -0.1446$$

$$b_2 = -0.1572$$

$$b_3 = 0.1449$$

$$a_0 = 1$$

$$a_1 = -2.769$$

$$a_2 = 2.552$$

$$a_3 = -0.7824$$

Una vez obtenidos los parámetros de la ecuación, se multiplican por 2^n para realizar las operaciones con enteros disminuyendo la pérdida de información, ya que el microprocesador a utilizar no dispone de una unidad de punto flotante para operar con números reales. Para hacer operaciones con reales sería necesario realizar muchas instrucciones en el software y esto aumentaría considerablemente el tiempo de cálculo.

En este caso se va a multiplicar los parámetros por $2^9 = 512$, por lo que efectuando esta multiplicación los parámetros que se obtiene son los siguientes:

$$b_0 = 81$$

$$b_1 = -74$$

$$b_2 = -80$$

$$b_3 = 74$$

$$a_0 = 512$$

$$a_1 = -1418$$

$$a_2 = 1306$$

$$a_3 = -400$$

Hay que tener cuidado con el redondeo de los coeficientes para acumular el menor error posible durante el cálculo de la ecuación en diferencias.

Una vez que se han redondeado los coeficientes se comprueba, mediante el programa *"Psim",* que la simulación es correcta cambiando en el esquemático el bloque H(s) por H(z) (ver Figura 3.26)

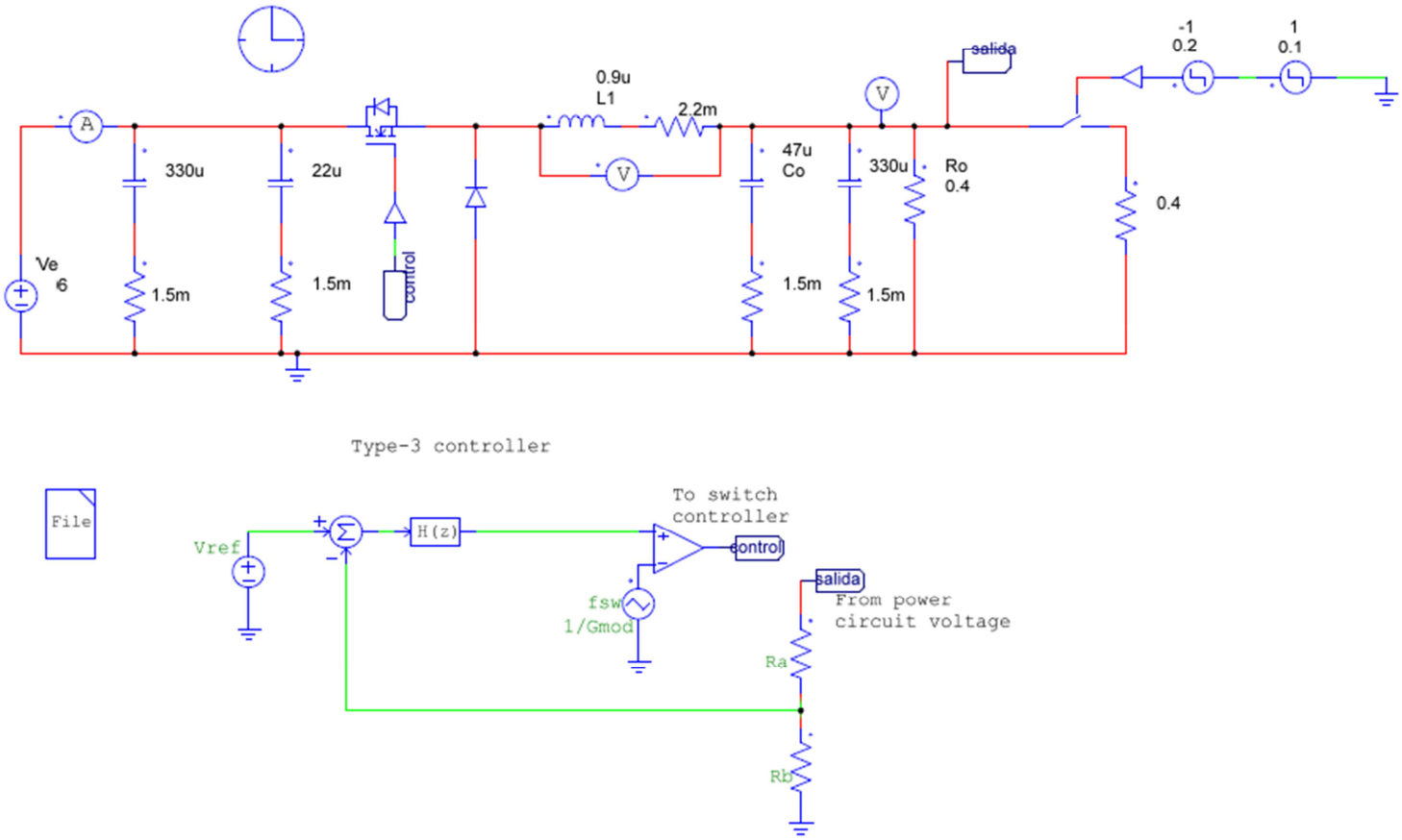
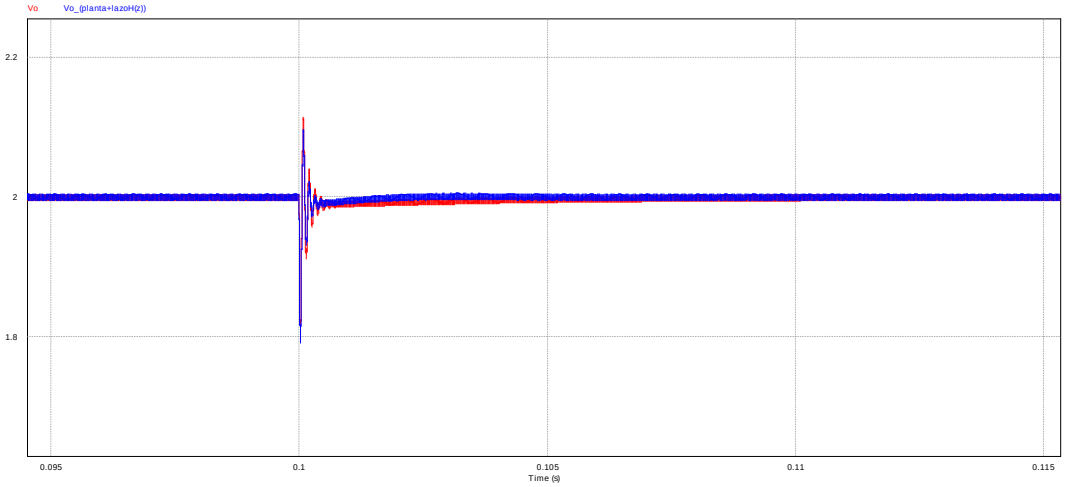


Figura 3.26: Sistema con el lazo de control H(z)



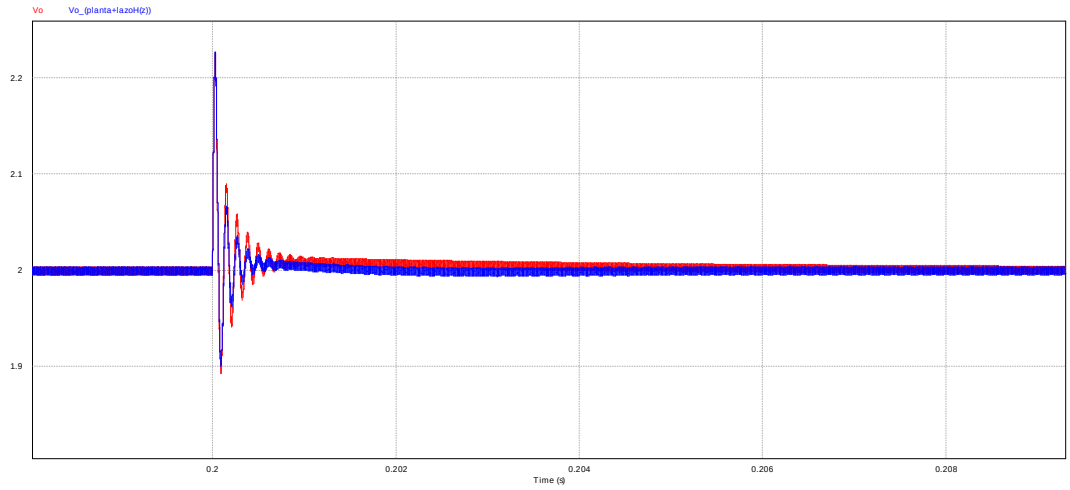


Figura 3.27: Comparación de la respuesta antes los escalones del lazo de control H(z) y H(S)

Como se puede observar en la Figura 3.27, la salida del convertidor es correcta, y la respuesta del sistema continuo (rojo) y la del sistema discreto (azul) son similares, por lo tanto el redondeo de los coeficientes está bien realizado.

3.3.3.1 Implementación del control

Como se ha visto en el punto 3.2.1.1 las características concretas que van a tener los distintos bloques a utilizar del PIC se va proceder a su configuración para poder implementar el lazo de control de forma correcta.

3.3.3.1.1 Programación módulo DPWM

Para configurar el módulo CCP como funcionamiento para DPWM hay que hacer los siguientes pasos:

1. Establecer el período de DPWM mediante el registro PR2.

2. Ajuste el ciclo de trabajo de DPWM mediante los registros CCPRxL y CCPxCON<5:4>

3. Configurar como salida el pin CCPx mediante el registro TRIS.

4. Habilitar y establecer el valor pre-escalado TMR2.

5. Configure el módulo CCPx para que opere en modo DPWM.

Por lo que los pasos de nuestra función de configuración para el bloque PWM los podemos ver en el la Figura 3.28,

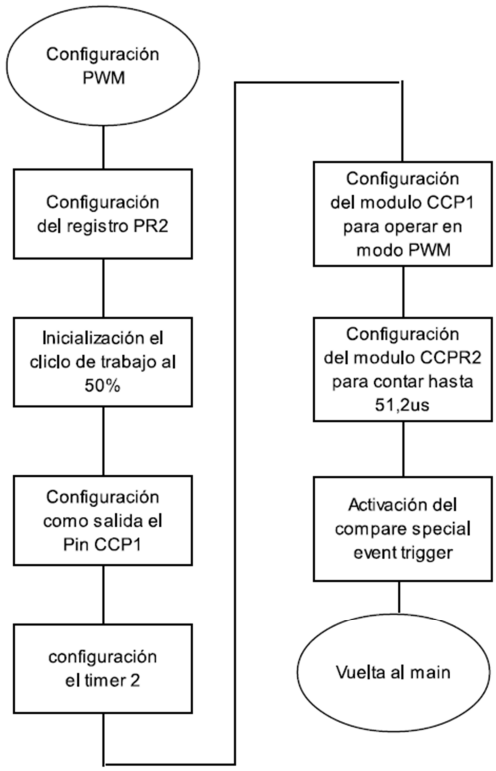


Figura 3.28: Flujograma de la configuración PWM

3.3.3.1.2 Configuración ADC

Para configurar el conversor A/D hay que tener en cuenta 5 registros:

1. *A/D Result High Register (ADRESH):* variable en la que se guarda la conversión de analógico a digital.
2. *A/D Result Low Register (ADRESL):* variable en la que se guarda la conversión de analógico a digital.
3. *A/D Control Register 0 (ADCON0):* registro que controla el funcionamiento del A/D

4. *A/D Control Register 1 (ADCON1):* registro que configura las funciones de los pines.
5. *A/D Control Register 2 (ADCON2):* registro que configura el reloj del A/D.

Por lo que los pasos de nuestra función de configuración para el bloque PWM los podemos ver en el la Figura 3.29

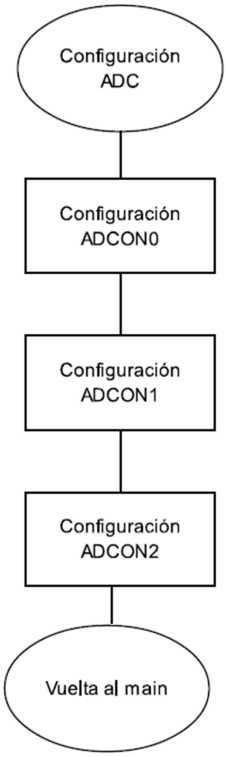


Figura 3.29: Flujo grama ADC

Una vez que se tienen los datos de los distintos módulos procedemos a la programación del PIC. Para ello usamos la herramienta *MPLAB IDE*. Con esta herramienta se puede configurar, programar y depurar el PIC.

3.3.3.2 Programación PIC

Para programar el PIC se utiliza el lenguaje C. Durante el desarrollo del código de deben seguir los siguientes pasos:

1. *Definición de los coeficientes*: lo primero que se debe hacer en el código es la definición global de los coeficientes a y b de la ecuación en diferencias.

   Los valores de estas variables son los obtenidos en la discretización (apartado 3.3.3 de capítulo 3) multiplicados por 512, para que el PIC no trabaje con decimales y sea más rápida su ejecución.

   Al haber multiplicado los coeficientes se debe dividir el resultado obtenido en la ecuación en diferencias por 512 (valor por el que se multiplicaron dichos coeficientes) para obtener el valor correcto del ciclo de trabajo. Por lo tanto, crearemos también una variable global, divisor, que almacene dicho valor.

2. *Esperar a que se calculen los nuevos valores de "d"*: La conversión del ADC se lanza automáticamente, por lo que es necesario crear un contador para que se lance cuando se haya terminado el cálculo del nuevo valor del ciclo de trabajo (d), y no antes para no poder información.

3. *Actualización de los valores anteriores de la conversión de "Vo":* lo primero que ejecuta el PIC es almacenar los valores anteriores de las conversiones, de manera que la información pasa del registro 1 al registro 2 y así sucesivamente hasta el registro 4.

4. *Almacenamiento de la nueva conversión de "Vo"*: a continuación se guarda el nuevo valor en el registro 1. El valor guardado en el registro 1 se desplaza dos posiciones, para que queden así los 6 bits más significativos. Este paso es necesario ya que como se ha calculado anteriormente el conversor A/D tiene una resolución de 6 bits.

5. *Obtención del error:* posteriormente se resta el valor de referencia con el dato guardado en la primera posición del registro. El valor de referencia en este caso serían 2V. Para calcular el valor equivalente a los 2 voltios en digital, y para restar adecuadamente los dos números empleamos la siguiente fórmula:

$$V_{ADC} = \frac{1}{\frac{V_{refAD+} - V_{refAD-}}{2^n}} \cdot V_{ref} = \frac{2}{\frac{5}{2^6}} = 25.6$$

   Analizamos solo la parte real y la convertimos a hexadecimal quedando un valor de 0x19.

6. *Cálculo de la ecuación en diferencias*: el siguiente paso es calcular la ecuación en diferencias. El valor obtenido se divide entre la variable divisor.

7. *Actualización de los valores anteriores y del nuevo valor de "d":* antes de guardar el nuevo valor se almacenan los valores obtenidos anteriormente de forma que la información pasa del registro 1 al registro 2 y así sucesivamente hasta el registro 4. Y se guarda el nuevo valor obtenido en la ecuación de diferencias en la primera posición del registro d.

8. *Cálculo de la señal DPWM*: con el valor actual de d se calcula mediante la función "SetDCPWM1" la señal DPWM.

Estos pasos se pueden ver en el flujograma de la Figura 3.30.

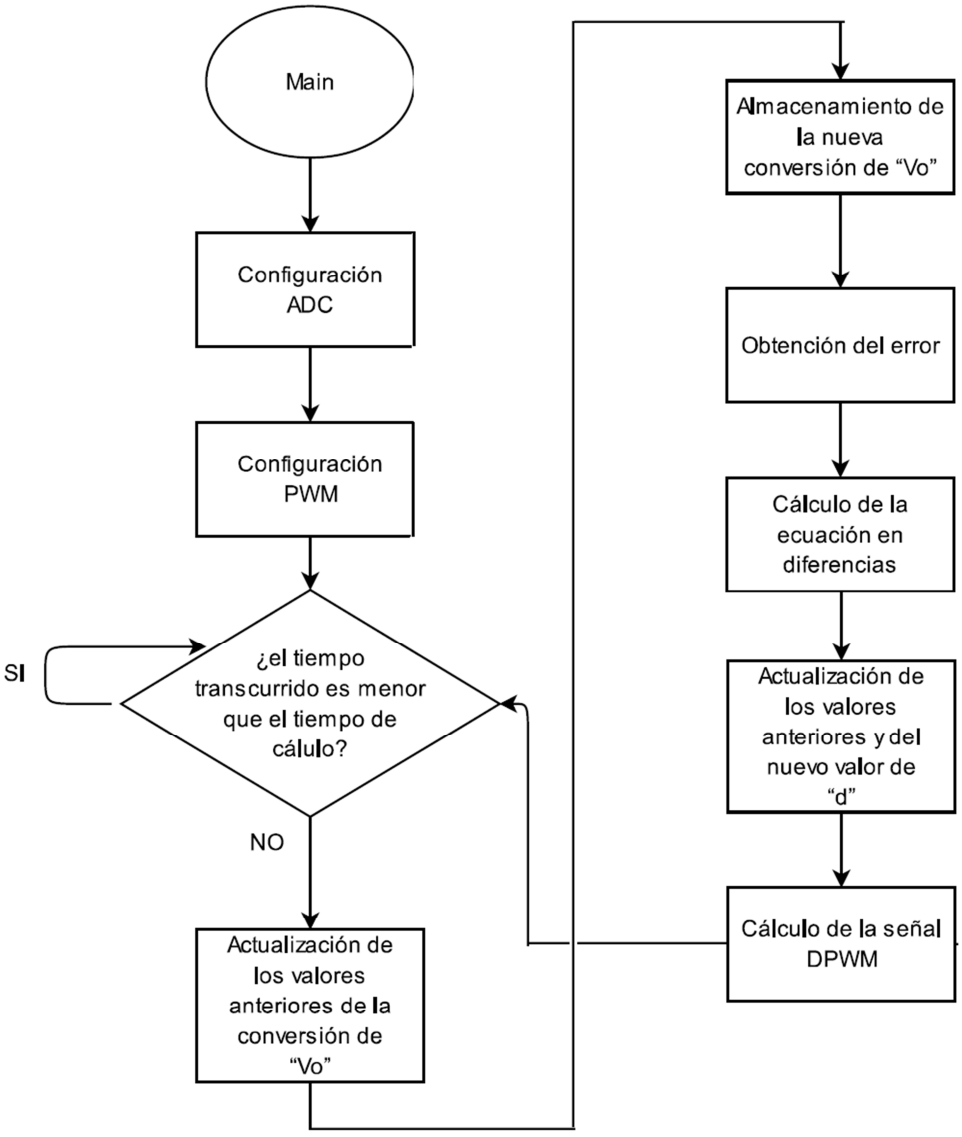


Figura 3.30: Flujograma del main

Para poder ejecutar el código se tienen que haber definido las funciones ADC y DPWM, se deben haber incluido las siguientes librerías:

```
#include <p18f2525.h>

#include <adc.h>

#include <pwm.h>
```

Una vez que se tiene el código se crea una lista de estímulos para poder comprobar que se guardan correctamente los datos en el registro y comprobar el tiempo que tarda en ejecutar el programa mediante la ventana Stop Watch señalada en la Figura 3.31:

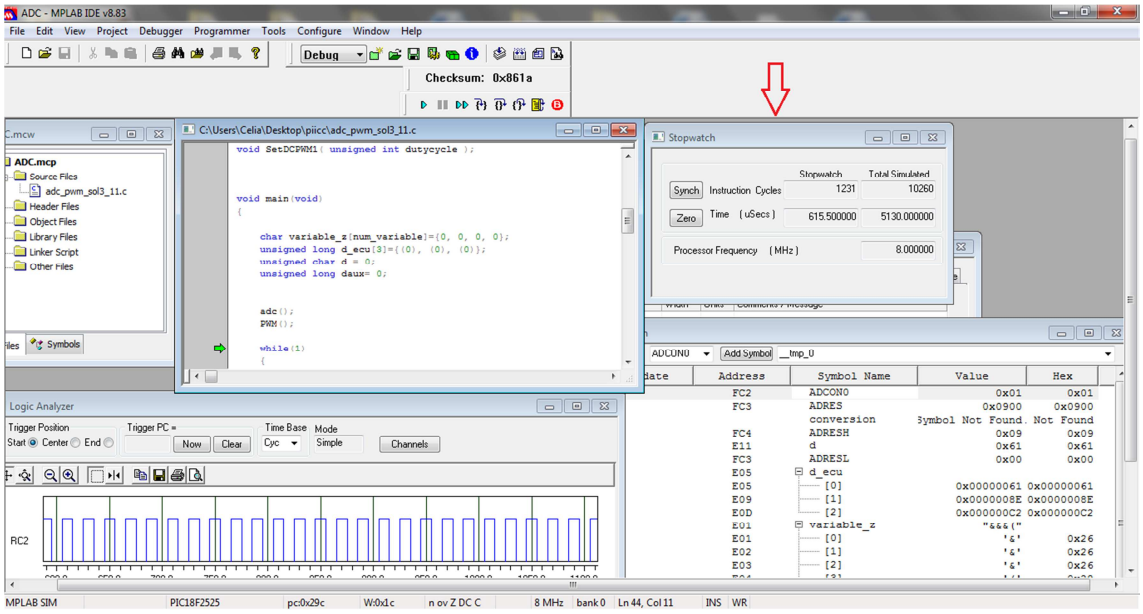


Figura 3.31: Imagen MPLAB con el código C

El tiempo de cálculo para la implementación de la ecuación en diferencias sería de 615,5 µs, por lo que sería a frecuencia de $\frac{1}{615{,}5\ \mu s} = 1{,}6KHz$. Esta frecuencia es demasiado baja para poder realizar un buen control sobre el convertidor ya que conmuta a 300KHz.

Una de las posibilidades para aumentar la velocidad es aumentar el oscilador, pero en este caso ya se ha elegido el oscilador más alto que permite

el PIC, por lo que la siguiente opción es programar el PIC en ensamblador para ahorrar el paso de conversión de C a ensamblador que tiene que realizar el PIC.

El código en ensamblador consta de las mismas partes que el código C. Los cambios necesarios para programar el PIC en este lenguaje son:

- Asignar a las variables una dirección de memoria mediante las directivas adecuadas (por ejemplo, "*pragma idata MisDatos*")
- Y definir la posición de cada variable.

Esto es necesario ya que para programar en ensamblador se trabaja con las direcciones de memoria.

Para conocer las direcciones de las variables propias del PIC que se tienen que utilizar se puede mirar o bien en el catálogo o bien en la ventana "*Watch"* del programa *MPLAB* donde se pueden ver las variables seleccionadas.

Por ejemplo, para dividir el resultado de la ecuación en diferencias entre 2^9 (el valor por el que se multiplican los coeficientes a y b de la ecuación) lo que hacemos es desplazar el registro 9 posiciones.

Como nuestra variable *d* tiene reservados 4 registros de 8 bits cada uno. Para desechar los 9 primeros bits se desplaza con acarreo la posición 3 y a continuación se hace lo mismo con la posición 2, de forma que el último bit de la posición 3 pasa a ser el primero de la posición 2 y ese valor es el de la ecuación en diferencias dividido entre 512 (2^9).

Haciendo el código en ensamblador el tiempo obtenido es de 32,8µs por lo tanto obtenemos una frecuencia que ha aumentado hasta un valor de 28KHz ver Figura 3.32. Al tiempo de cálculo hay que sumarle el tiempo que tarde en coger el dato el ADC que se calcula de la siguiente forma:

$$T_{ADC} = 11 \cdot T_{ad} + T_{acq} = 11 \cdot 32 \cdot T_{osc} + 1{,}2\mu s = 11 \cdot 32 \cdot \frac{1}{25Mhz} + 1{,}2\mu s$$

$$T_{ADC} = 15.28\mu s$$

Finalmente la frecuencia del PIC es la siguiente:

$$F = \frac{1}{15.28\mu s + 32.8\mu s} = 20.7KHz$$

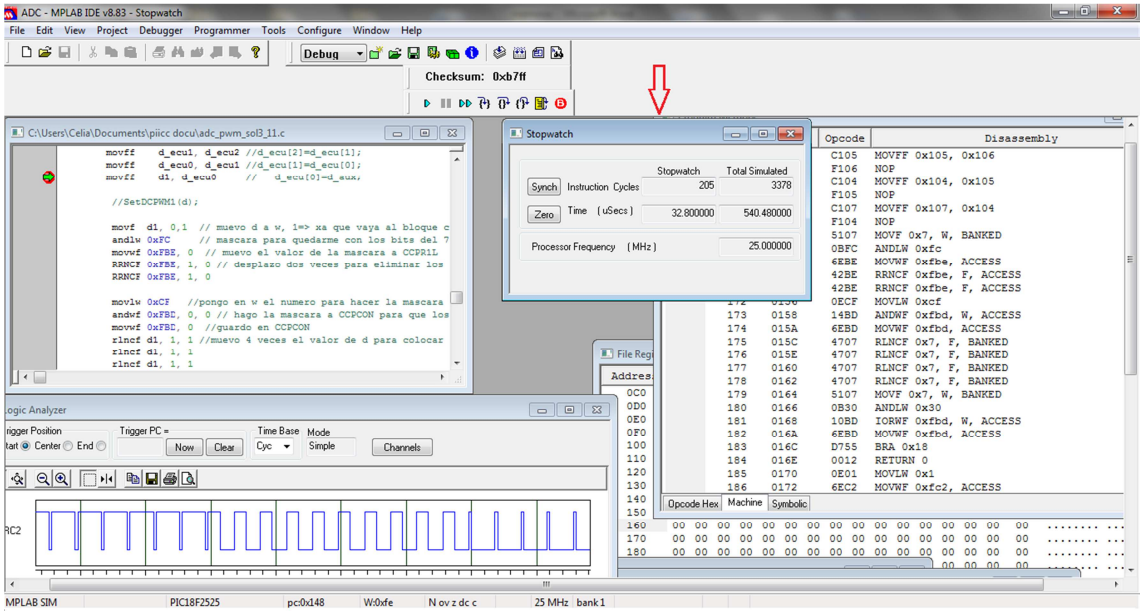


Figura 3.32: Imagen MPLAB código ensamblador

Una vez que se ha calculado el tiempo que tarda el PIC en ejecutar todo, se configura el valor del CCPR2H y CCPR2L para que la conversión empiece cada 19.5KHz., De esta forma se consigue tener tiempo suficiente para calcular el valor de la señal de PWM correctamente. Además este tiempo es múltiplo de la frecuencia a la que trabaja el DPWM.

Una vez obtenido el código se comprueba, introduciendo los mismos estímulos en el programa *MPLAB,* que el programa hace lo mismo tanto progamado en lenguaje el C como en lenguaje ensamblador.

Para hacer esta comprobación se recurre a las siguientes pantallas:

- "*file registers*" : en esta pantalla se ve el contenido de la memoria del PIC y se pueden observar los cambios de las variables,

- *"Watch"*: esta pantalla permite seleccionar las variables que se deseen para ver el registro en el que se guardan y el valor que va tomando.

Estas pantallas son de gran utilidad a la hora de comprobar la programación en ensamblador.

Otra pantalla utilizada es el *"Stopwatch"* que como hemos visto anteriormente nos sirve para medir el tiempo de ejecución del programa.

También se ha utilizado *"Logic Analyzer"* que muestra la señal DPWM que se obtiene en cada momento.

Capítulo 4 Diseño de la PBC

La realización de la placa de circuito impreso se ha hecho en colaboración con otro alumno. [9]

Para el diseño de la placa del circuito impreso se ha utilizado la herramienta *ORCAD*.

La herramienta *ORCAD* es un programa que sirve para el diseño de circuitos electrónicos. Consta de dos bloques:

- *PSPICE* : sirve para ver el comportamiento de los circuitos electrónicos
- *LAYOUT*: sirve para realizar el diseño de placas de circuitos impresos PCB

En este proyecto se va a realizar el diseño para el convertidor y los elementos externos necesarios para su correcto funcionamiento (hardware). Para ello usaremos la herramienta *LAYOUT* que contiene las librerías de los distintos componentes con sus dimensiones. Sin embargo, para este diseño se han tenido que crear en algunos casos nuevas librerías, ya que no se disponía de ellas en las librerías que tiene por defecto el programa. Por ejemplo, se ha tenido que crear una huella específica para el convertidor.

Como paso previo al diseño de la PCB es necesario realizar la captura del esquema del circuito que se quiere analizar. En este caso el circuito queda de la siguiente forma, ver Figura 4.1:

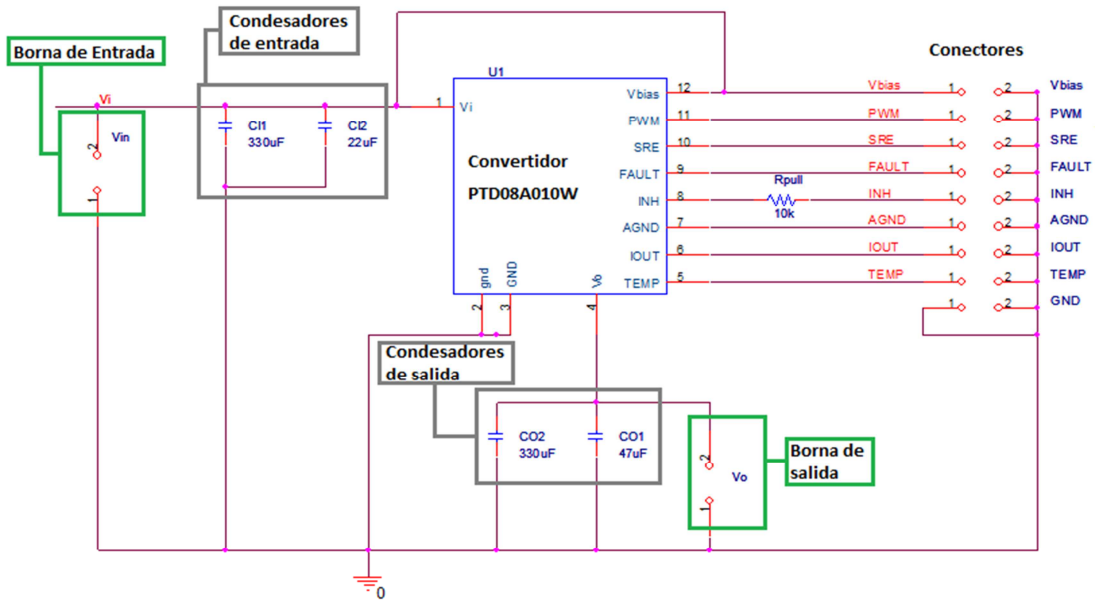


Figura 4.1: Esquemático para el prototipo del convertidor

En el esquemático se pueden ver los diferentes componentes que van a formar la placa:

- Para el convertidor, tal y como se ha mencionado anteriormente, ha sido necesario crear una huella con sus dimensiones. Dichas dimensiones han sido obtenidas de la hoja de características proporcionada por el fabricante, donde se especifica la distancia entre pines y el ancho de ellos.

- Las bornas de entrada y salida han sido elegidas en base a las especificaciones estudiadas en el convertidor, ya que debido a la elevada intensidad que pasa por ellas no se puede poner el mismo conector que a la salida de los pines del convertidor. Por no ser un conector tipo se han tenido que crear las huellas de estos conectores mediante la herramienta "*LAYOUT*" como se especificó anteriormente.

- Los condensadores de entrada y salida electrolíticos no disponen de huella en las librerías, por lo que ha tenido que ser creada. Para obtener las medias se han usado las hojas de características del condensador que se eligió.

- Los condensadores cerámicos y las resistencias sí tienen una huella asociada con sus dimensiones, por lo que no ha sido necesario crear ninguna adicional.

- Para las salidas del convertidor se han usado conectores convencionales, por lo que no ha sido necesario crear ninguna huella.

En la Figura 4.2 podemos ver la huella del convertidor y de las bornas:

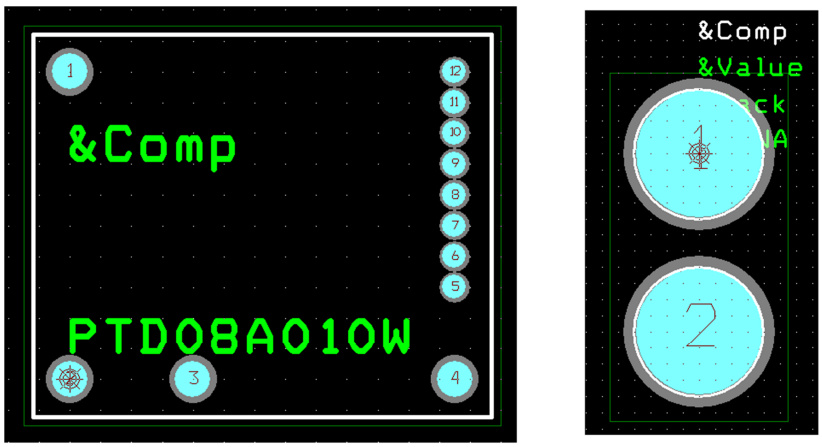


Figura 4.2: Huellas del convertidor y de las bornas

Una vez que se tiene el esquemático y todas las huellas de los componentes, se exporta dicho esquemático al *"LAYOUT"* para crear el diseño de la placa a fabricar. Ver Figura 4.3

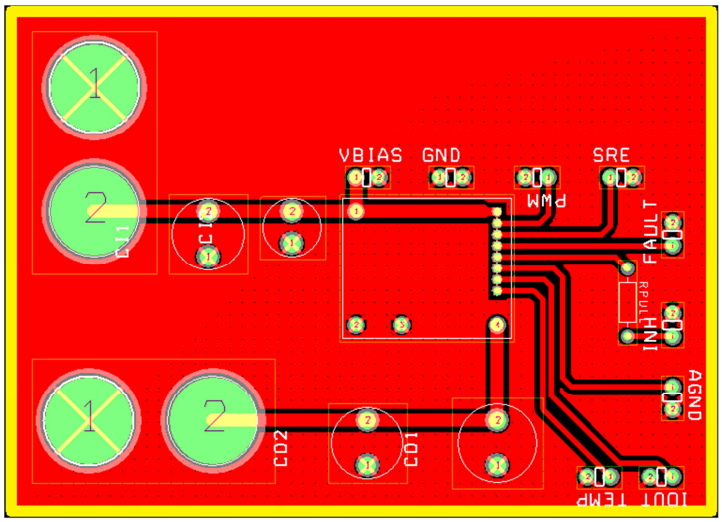


Figura 4.3: Rutado del prototipo del convertidor

Capítulo 5 Validación del Sistema

A lo largo de este capítulo vamos a realizar las comprobaciones necesarias mediante simulación y experimentalmente para ver si el código implementado es correcto y si somos capaces de logar la salida deseada con los elementos que disponemos.

5.1 Comprobación de la generación del ciclo de trabajo.

Para comprobar que el PIC genera bien el ciclo de trabajo se conecta la placa al ordenador mediante el conector para depuración y programación "*PICkit 2*". De esta forma se graba el programa en el PIC. Es necesario cambiar el modo de depuración del *MPLAB SIM* al *PICkit2*, ya que si no es imposible realizar la operación.

A continuación, se conecta al PIC una fuente de tensión que equivaldría a la tensión de salida del convertidor. Además, se conecta el osciloscopio a la salida del *DPWM* para ver que el ciclo de trabajo varía. Ver Figura 5.1.

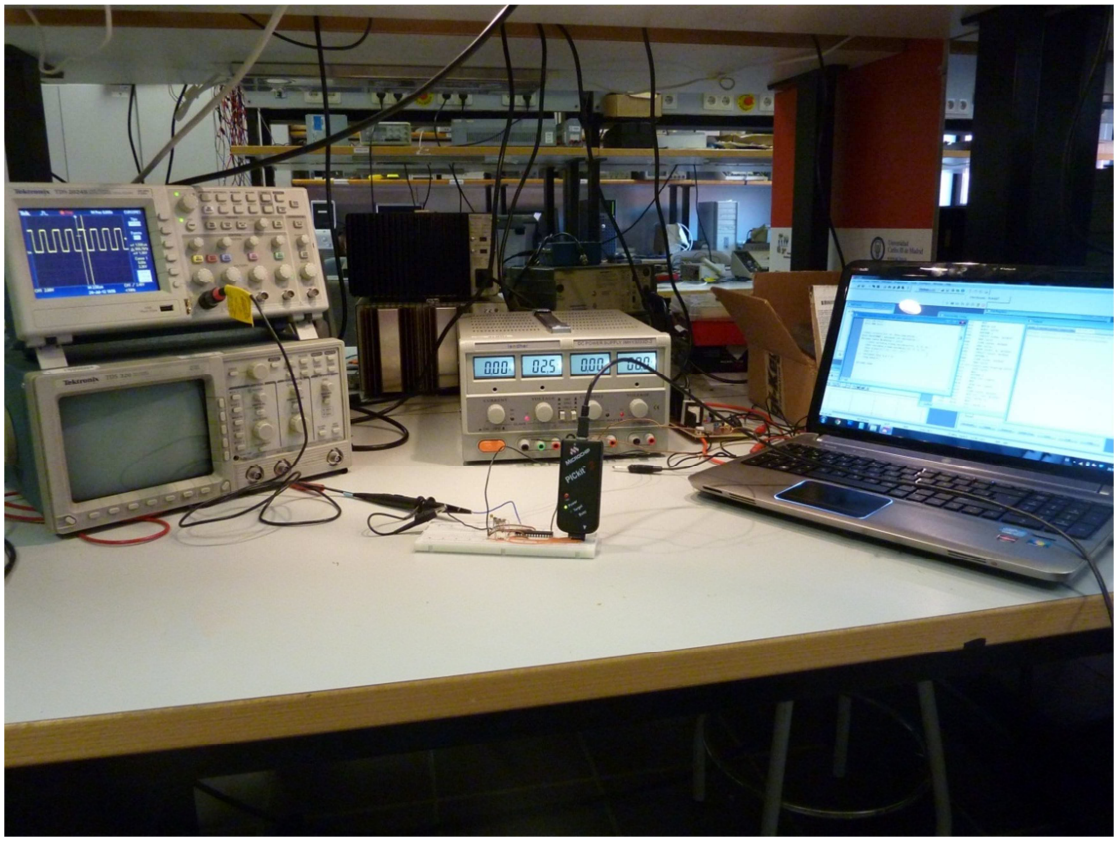


Figura 5.1: Imagen de los elementos utilizados en el laboratorio

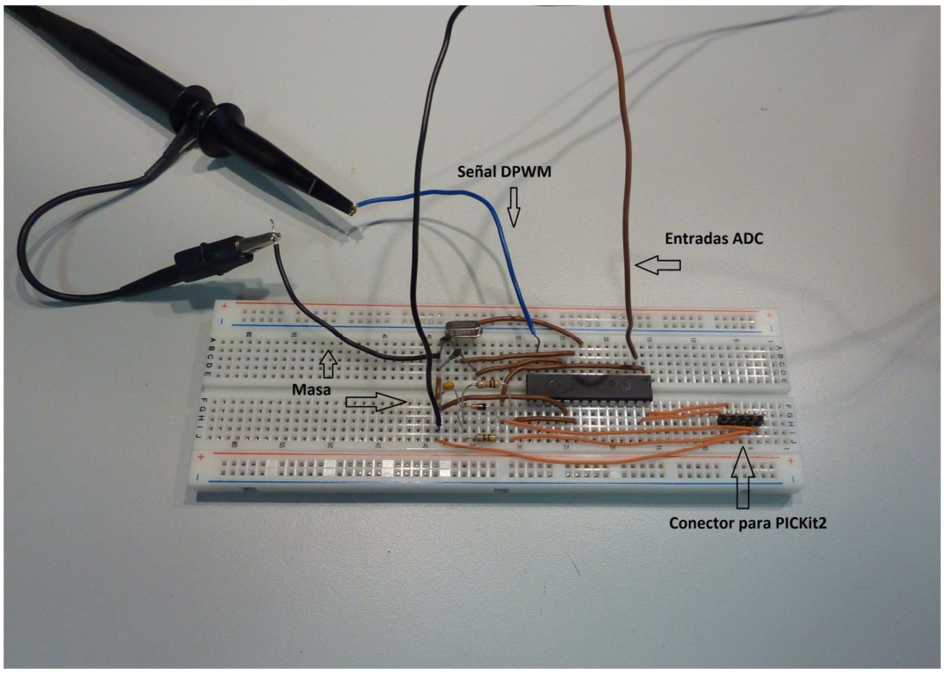


Figura 5.2: Conexiones realizadas en la Placa

En la Figura 5.2 se pueden ver:

- conexiones realizadas al osciloscopio
- fuente de alimentación
- PICkit2 encargado de conectar el PIC y el ordenador.

Tras la conexión se comprueba con el osciloscopio que la señal del ciclo de trabajo es correcta. Para saber si la señal se genera bien comprobamos el valor de su frecuencia y que la señal se modifica en función de las distintas entradas que le llegan de la fuente.

En las siguientes figuras observamos la señal DPWM que medimos con el osciloscopio:

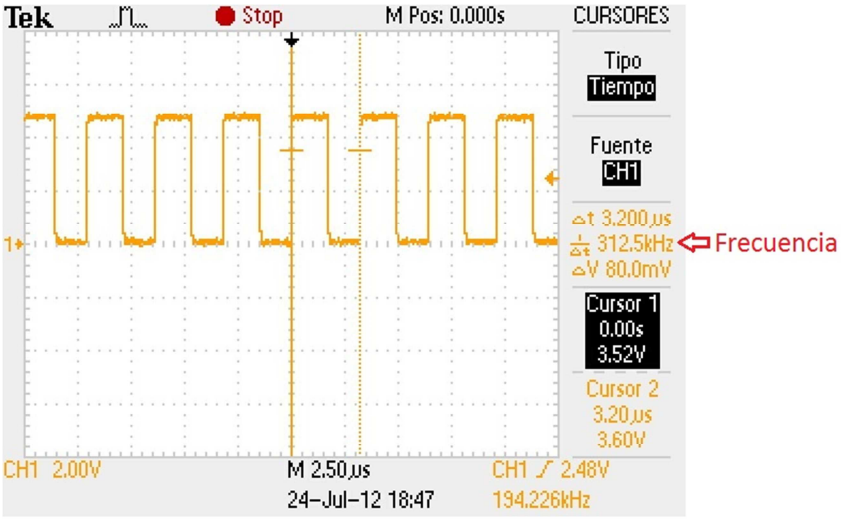


Figura 5.3: Señal DPWM

En la Figura 5.3 se observa que la frecuencia del DPWM es de 312.5KHz. Esta frecuencia coincide con la calculada teóricamente en el apartado 3.2.1.1.1

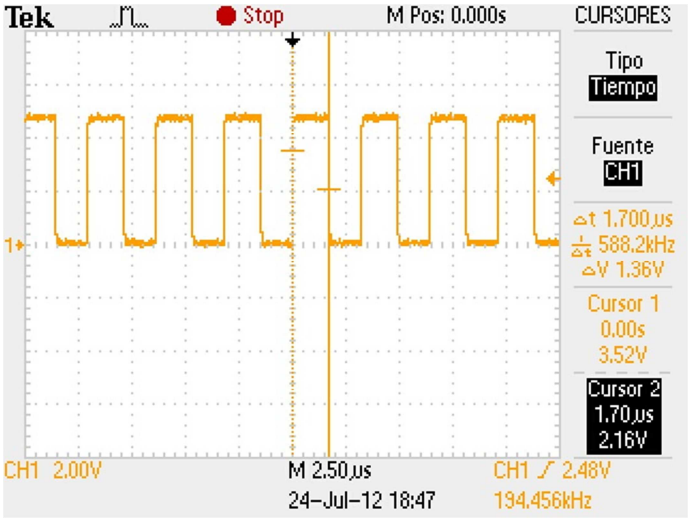


Figura 5.4: Señal DPWM con un ciclo de trabajo del 53%

En la Figura 5.4 se observa un ciclo de trabajo del 53%. Para calcular el valor del ciclo de trabajo, dividimos el tiempo que está a 1 entre el tiempo total. El tiempo total se puede observar en la Figura 5.3 y el tiempo que esta encendido Figura 5.4.

Por lo tanto el ciclo de trabajo es el siguiente:

$$D = \frac{T_{on}}{T} \cdot 100 = \frac{1.7}{3.2} \cdot 100 = 53\ \%$$

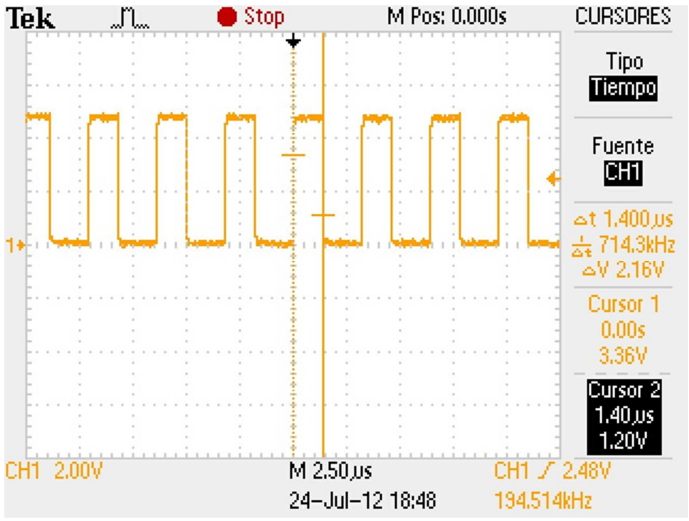


Figura 5.5 Señal DPWM con un ciclo de trabajo del 43%

En la Figura 5.5 vemos que el ciclo de trabajo ha disminuido. Para saber el nuevo valor aplicamos la misma fórmula:

$$D = \frac{T_{on}}{T} \cdot 100 = \frac{1.4}{3.2} \cdot 100 = 43\ \%$$

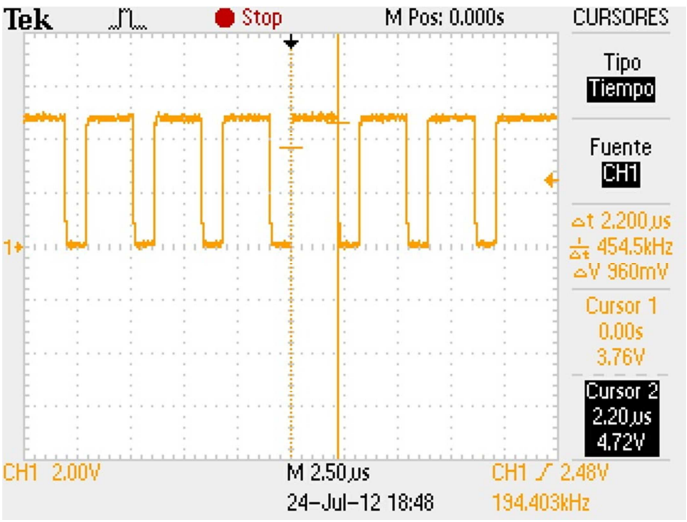


Figura 5.6: Señal DPWM con un ciclo de trabajo del 69%

Por último, en la Figura 5.6 vemos que el valor del ciclo de trabajo ha aumentado:

$$D = \frac{T_{on}}{T} \cdot 100 = \frac{2.2}{3.2} \cdot 100 = 69\ \%$$

De esta manera se comprueba que el valor de D varía según las entradas que vaya tomando el PIC.

5.2 Simulación en Psim del sistema real.

El siguiente paso es simular el regulador implementado con un programa en C en la herramienta PSIM para comprobar que el código programado es correcto. Comprobamos además qué salida es la que vamos a tener cuando lo conectemos al convertidor.

Para simular el sistema final de la forma más real posible se tiene que cuantificar tanto la salida como la señal de referencia (2V). Esto se ha realizado mediante los bloques de cuantificación del "*Psim*". La señal triangular necesaria para la obtención del DPWM también se tiene que cuantificar. Ver Figura 5.7

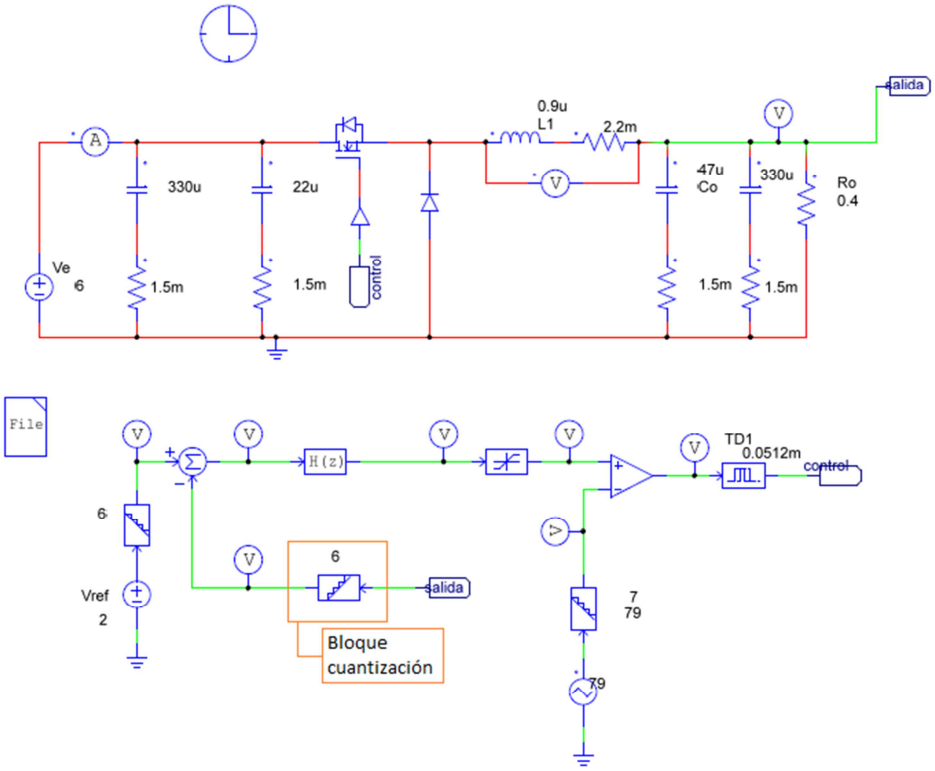


Figura 5.7: Esquemático del convertidor con el control H(z) cuantificado

La función del bloque de cuantización es discretizar la señal de continua. Este comportamiento se asemeja a lo que haría el ADC del PIC. Los parámetros necesarios para este bloque son:

- Nº of bit → El nº de bits en el caso del bloque señalado se corresponde al nº de bits del ADC. El bloque de cuantización de la referencia deberá ser igual al bloque de cuantización del ADC para poder realizar una buena comparación. En el caso del cuantificador de la rampa el nº de bits será idéntico a los del DPWM.
- Vin-min → valor mínimo de entrada que va a tener el bloque. Es 0 en todos los casos.
- Vin-max → valor máximo de entrada que va a tener el bloque. En el caso del bloque del ADC este valor es de 5V (tensión de referencia del ADC) y en el caso del cuantificador de la rampa es de 79 (ya que es el máximo valor que puede medir DPWM del PIC).
- Vo-min → valor mínimo de salida que va a tener el bloque. Es 0 en todos los casos.
- Vo-max → valor máximo de salida que va a tener el bloque. En el caso del ADC este valor es de 63 (ya que al tener 6 bits va de

0 a 63) y en el caso de DPWM es de 79 (valor máximo que va alcanzar el PIC).

- Sampling frecuency → valor de frecuencia de muestreo. Para los bloques del ADC este valor es de 19531.25 Hz (múltiplo de la frecuencia del DPWM y por debajo de la que tarda el PIC. Condiciones necsarias para que exista tiempo suficiente para realizar todas las operaciones). La frecuencia del DPWM es 312.5KHZ*80 (pasos).

Como se puede observar en la figura además de los bloques para la discretización de las señales también se ha añadido el bloque "*delay*" para tener en cuenta el retraso del PIC y que sea lo más semejante a la realidad. La simulación obtenida del este circuito es la representada en la Figura 5.8.

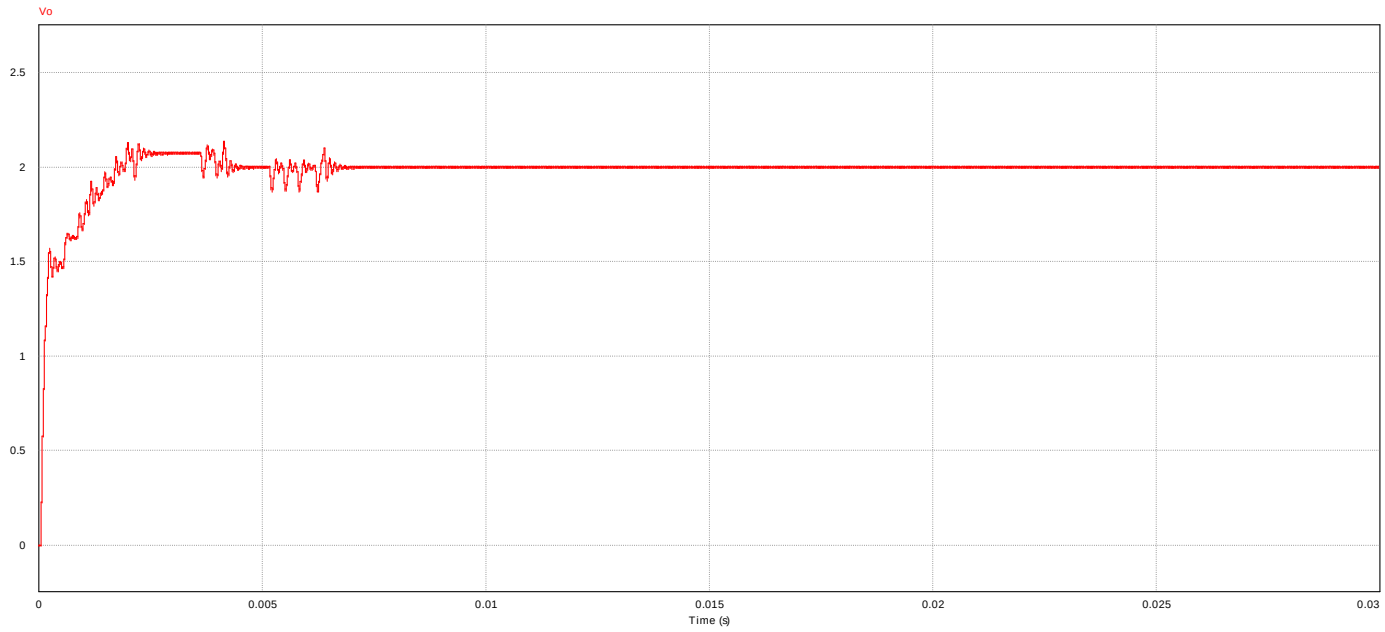


Figura 5.8: Simulación con el control H(z) cuantificado

Debido a la poca precisión del PIC y al retraso introducido, la señal tarda más en estabilizarse. Pero como se observa en la imagen, finalmente se obtiene la salida deseada.

El último paso a seguir es realizar la simulación con el código C. Esta simulación se realiza para ver si con el código del PIC se obtiene la salida deseada y comprobar que se ha implementado bien la ecuación en diferencias.

Para esta nueva comprobación se cambia el bloque H(z) (regulador) por nuestro código C, comprobando si el resultado es el mismo. Para obtener una buena simulación es necesario poner un bloque *Zero-order hold* con la frecuencia del PIC. En este caso dicha frecuencia toma un valor de 19531.25 Hz, para que todo el bloque de realimentación se ejecute en el mismo tiempo. Ver Figura 5.9

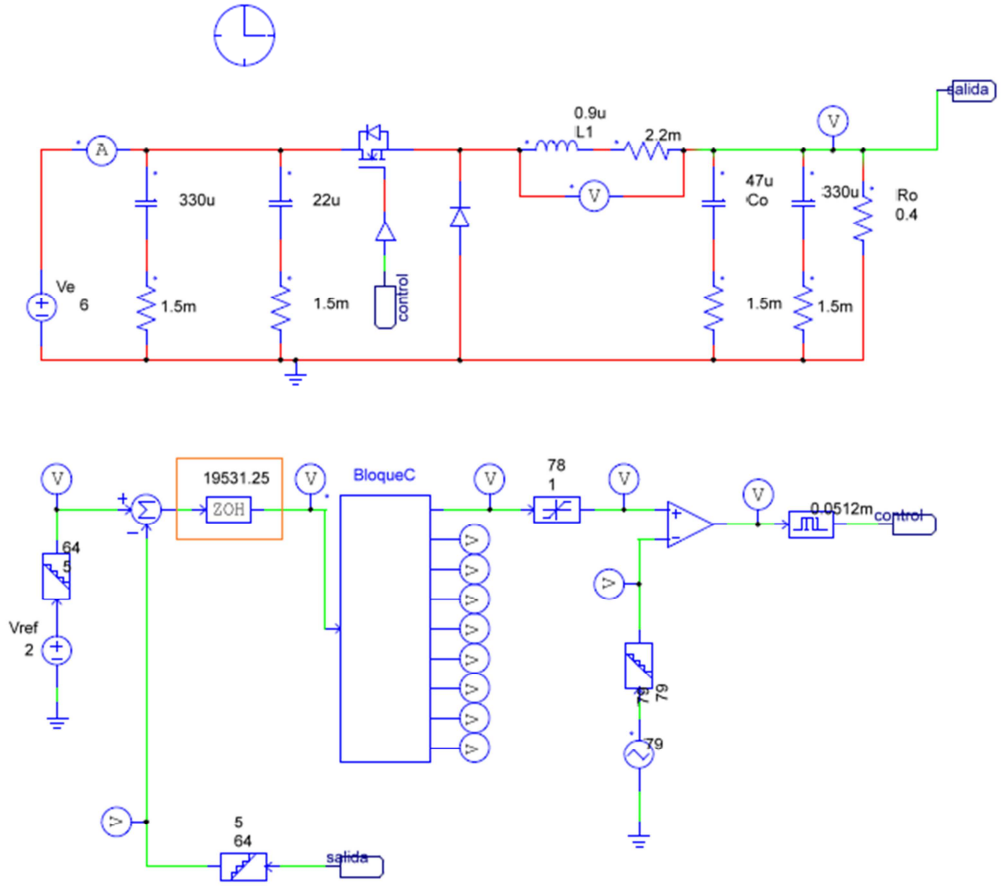


Figura 5.9: Esquemático del convertidor con el control en C cuantificado

Para poder utilizar el bloque C de forma correcta hay que crear las variables con el comando *"static"* delante para que siempre las guarde en la misma dirección y no la cambie cada vez que se ejecute el código.

Llegados a este punto se observa la verdadera importancia que tiene el redondeo de los coeficientes de la ecuación en diferencias, ya que en función de la precisión que se tenga la salida se estabiliza en 2V o no.

Utilizando el valor de los coeficientes multiplicados por 2^9 como dijimos anteriormente, la salida obtenida no es válida (ver Figura 5.10, donde el color rojo representa la simulación con el bloque C y el color azul representa la simulación con el bloque H(z)):

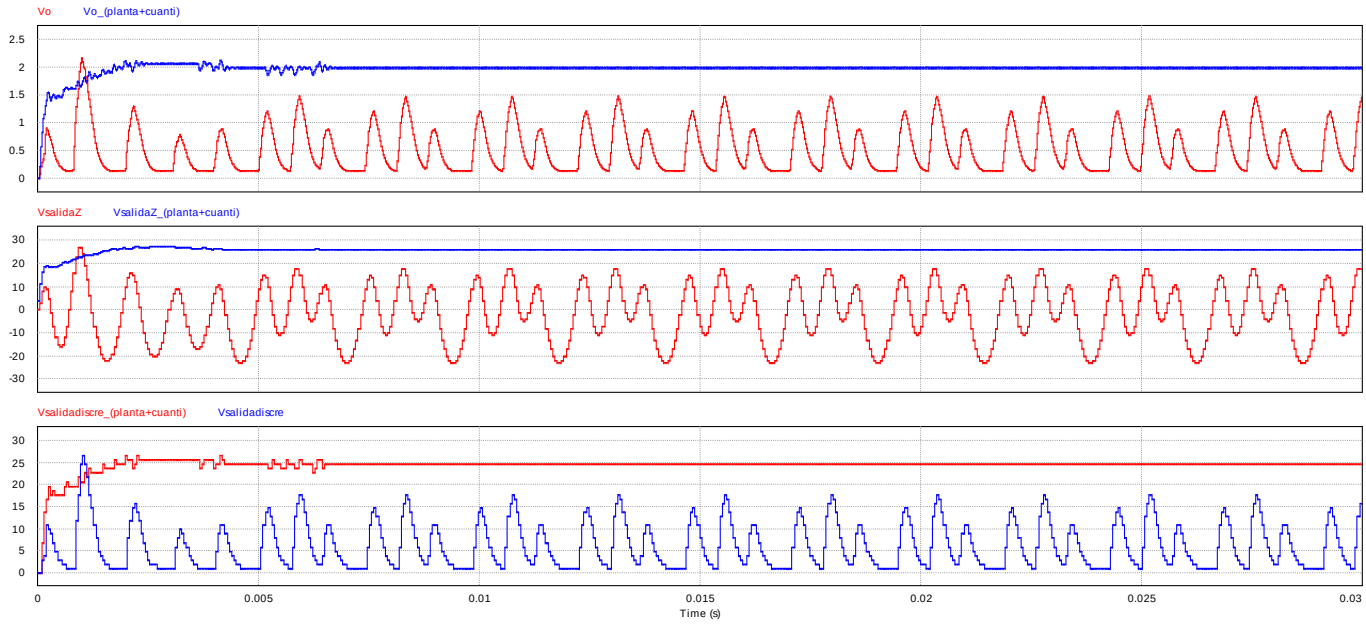


Figura 5.10: Simulación con el control C cuantificado

En lugar de lo anterior, se modifican las siguientes variables del código C:

- *d* : pasa a ser de tipo *float* en lugar de tipo *long*
- *daux:* pasa a ser de tipo *float* en lugar de tipo *long*

De esta forma se obtiene la salida deseada (ver Figura 5.11, donde el color rojo representa la simulación con el bloque C y el color azul representa la simulación con el bloque H(z)):

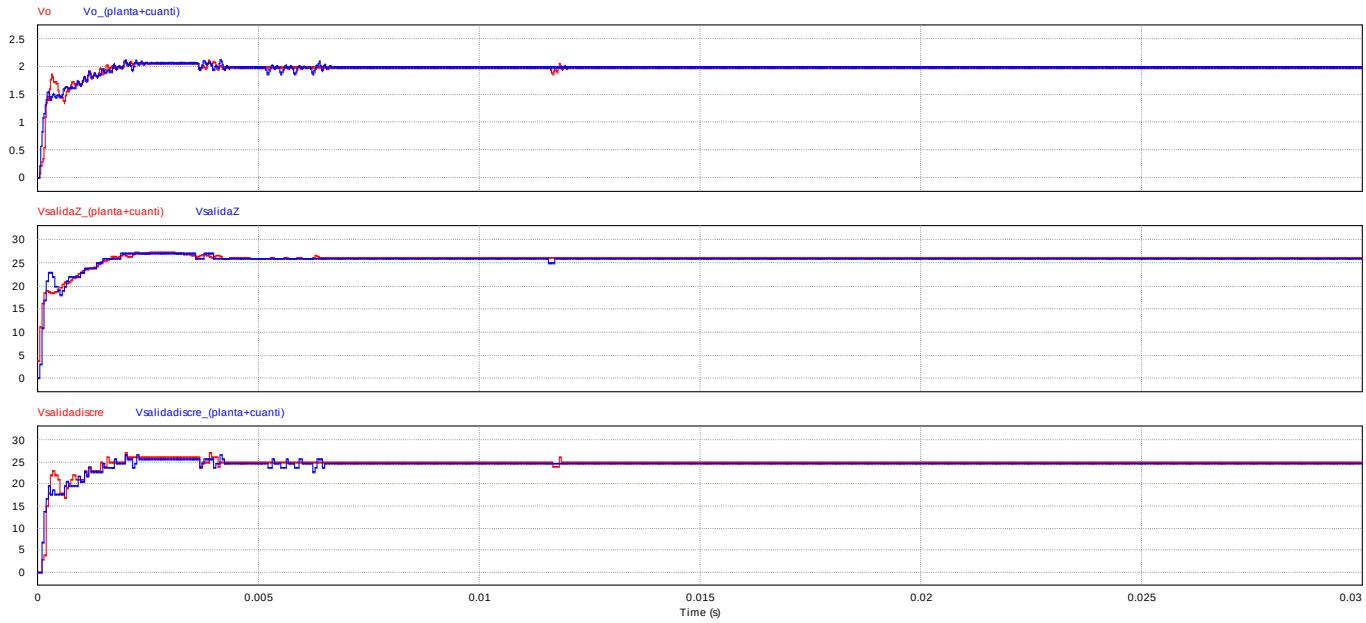


Figura 5.11: Simulación con el control C cuantificado, cambiando e tipo de variables

En estas simulaciones se ha comprobado la magnitud de error que se comente con un mal redondeo de los coeficientes de la ecuación en diferencias.

Debido a que al trabajar con variables *float* la frecuencia del PIC sería menor de 19KHz, no se pueden hacer tan precisos los cálculos con el PIC del que disponemos. Esto significa que no se puede probar el sistema de forma física ya que la salida que se obtendría sería parecida a la de la Figura 5.10, y por tanto no es válida.

Capítulo 6 Conclusiones y trabajos futuros

Después de haber desarrollado todo el trabajo se procede a realizar una valoración global del trabajo fin de grado.

6.1 Conclusiones

En el capítulo 1 se indicaron los objetivos principales que pretendía cubrir el trabajo. A continuación se procede a analizar los problemas encontrados y los resultados obtenidos para cada uno de ellos.

El objetivo principal del trabajo es desarrollar el software y hardware necesario para la correcta implementación del control en un convertidor comercial CC/CC. A lo largo del trabajo se han explicado los pasos seguidos para el desarrollo tanto el software como el hardware, adoptando las medidas necesarias para solucionar los problemas que se han ido encontrando durante el desarrollo del mismo.

Respecto al objetivo de adaptar el proceso de diseño de la herramienta "*SmartCtrl*" a los controles digitales, teniendo en cuenta las características de estos últimos, tal y como se ha visto en el capítulo 5 se ha obtenido un regulador digital con la misma respuesta que el analógico, por lo tanto se puede concluir que se han tenido en cuenta todas las características de ambos tipos de control.

Sin embargo no ha sido posible realizar la implementación real del lazo del control digital debido a que el PIC18F2525 no era lo suficientemente rápido como para regular el convertidor comercial de Texas Instruments. Aunque se ha realizado una optimización en tiempo de ejecución del código, mediante la utilización del código ensamblador en lugar de código C, no se ha conseguido obtener resultados adecuados en este caso.

Otro de los objetivos de este proyecto era documentar las técnicas y características a seguir para diseñar a partir de un lazo de control analógico uno digital. A lo largo del desarrollo de la memoria del proyecto se ha explicado la teoría necesaria para poder desarrollar el lazo de control digital, que era el principal objetivo del trabajo. En el capítulo 3 se ha documentado los pasos a seguir para el diseño.

6.2 Trabajos futuros

Como posibles trabajos futuros se pude utilizar la opción que tiene el microcontolador PIC18F2525 de utilizar el PLL para aumentar la frecuencia del oscilador 4 veces.

Otra opción es implementar el lazo con un microprocesador que permita un oscilador de mayor frecuencia. De esta forma el control se realizaría de una forma más rápida y se obtendría la respuesta adecuada.

Capítulo 7 Presupuesto

En este capítulo se van a desarrollar los costes que son necesarios para la realización de este proyecto, dichos costes se dividen en dos grandes bloques:

- Coste de material
- Coste del desarrollo de ingeniería y documentación

7.1 Coste del material

En este bloque se incluirá el coste de los materiales físicos utilizados. No se va a incluir el coste de los aparatos de media usados en el laboratorio, ya que no solo son utilizados para este proyecto y son parte del material que aporta la Universidad.

Las licencias necesarias de software para el uso de los programas empleados tampoco se van a añadir en el presupuesto ya que se tiene acceso a dichos programas en los laboratorios y en las aulas informáticas de la Universidad.

Tampoco se van a incluir los costes de los materiales relativos al hardware del convertidor ni a la fabricación de la PCB, ya que debido al retraso producido por el PIC no se ha fabricado.

A continuación en la Tabla 7.1 podemos observar el coste de los materiales:

Elemento	Coste (€)	Número de Unidades	Coste Total (€)
PIC	3,66	3	10,98
PICkit	28,44	1	28,44
Condensadores	0,362/0,219	2/1	0,724/0,219
Resistencias	0,10	2	0,20
Diodo	0,28	1	0,28
Oscilador	0,64	1	0,64
Conectores	0,528	1	0,528
Placa Protoboard	10	1	10
Total:			52,011

Tabla 7.1: Costes de material

7.2 Costes de desarrollo

A continuación se van a desarrollar los costes del estudio y diseño necesarios para alcanzar los objetivos del Trabajo (Ver Tabla 7.2). Estos costes se dividen en:

- Diseño y validación
- Documentación ver tabla

Actividad	€/hora	Horas	Coste
Diseño y Validación	35	228	7.980€
Documentación	20	90	1.800€
Total		318	9.780€

Tabla 7.2: Costes de desarrollo

7.3 Presupuesto Total del Proyecto

En la

Tabla ***3.1*** se puede observar el coste total del Trabajo.

Tipo de Coste	Coste (€)
Coste de Material	52,01
Coste de Desarrollo	9.780
Total antes del I.V.A	9.832,01
I.V.A (16%)	1.573,12
Total	11.405,13

Tabla 7.3: Coste Total

Capítulo 8 Bibliografía

[1] Documentación de la asignatura Electrónica de Potencia 2010/2011.

[2] Gus_wolvering "Sistema Digital y Sistema Analógico: concepto, ventajas y ejemplos" disponible en : http://www.monografias.com/trabajos27/analogico-y-digital/analogico-y-digital.shtml , Julio 2012

[3] Documentación de la asignatura Sistemas Electrónicos de Potencia 2011/2012.

[4] Universidad de Michigan "Análisis y diseño de la respuesta en frecuencia" http://www.ib.cnea.gov.ar/~instyctl/Tutorial_Matlab_esp/freq.html, Julio 2012

[5] Tiapa Jonatan "Sistemas Lineales" disponible en : http://www.monografias.com/trabajos46/sistemas-lineales/sistemas-lineales.shtml , Julio 2012

[6] Pablo Zumel, Cristina Fernández , Marina Sanz, Antonio Lázaro, Andrés Barrado, "Step-By-Step Design of FPGA-Based Digital Compensator for DC/DC Converters Oriented to an Introductory Course"

[7] Caracteristicas Convertidor PTD08A010, disponible en: http://www.ti.com/lit/ds/symlink/ptd08a010w.pdf

[8] Data Sheets PIC18F2525 disponible en: http://ww1.microchip.com/downloads/en/DeviceDoc/39626e.pdf

[9] Álvaro Granados "Diseño e Implementación en FPGA del Lazo de Control de un Convertidor de Potencia". Trabajo Fin de Grado 2012.

Capítulo 9 Anexos

En este capítulo se va desarrollar el código utilizado para implementar el lazo de control en el PIC. A continuación se expondrán tanto el código C, como el código Ensamblador.

9.1 Codigo C

```
#include <p18f2525.h>

#include <adc.h>

#include <pwm.h>

#define num_variable 4

// Definición de las variables de la ecuación en diferencias multiplicadas por 512

#define b0 81

#define b1 -74

#define b2 -80

#define b3 74


#define a0 512

#define a1 -1418

#define a2 1306

#define a3 -400


#define divisor 512


// Definición de la variable referencia (2V) pasada a digital

#define referencia 0x19
```

```
// Definición de las funciones del PIC

void adc(void);

void PWM(void);

void SetDCPWM1( unsigned int dutycycle );

// Programa principal

void main(void)

{


        char variable_z[num_variable]={0, 0, 0, 0};

        unsigned long d_ecu[3]={(0), (0), (0)};

        unsigned char d = 0;

        unsigned long daux= 0;


        adc();

        PWM();


        while(1)

        {


                while (!PIR2bits.CCP2IF);

                PIR2bits.CCP2IF=0;

                while(ADCON0bits.GO);

                //desplazamiento de datos

                variable_z[3]=variable_z[2];

                variable_z[2]=variable_z[1];

                variable_z[1]=variable_z[0];

                // Se guarda la conversión y se desplazan dos posiciones con el fin de
                quedarnos con 6bits en lugar de 8.
```

```
                variable_z[0]=ADRESH>>2;

                // Se resta la referencia (2V) al resultado del ADC para obtener el error

                variable_z[0]=(referencia-variable_z[0]);

                // Se calcula el valor de la ecuación en diferencias

                daux=(((b0*variable_z[0])+(b1*variable_z[1])+(b2*variable_z[2])+(b3*
                variable_z[3]))-((a1*d_ecu[0])+(a2*d_ecu[1])+(a3*d_ecu[2])));


                // Se divide entre 512

                d=daux/divisor;

                //desplazamiento de datos

                d_ecu[2]=d_ecu[1];

                d_ecu[1]=d_ecu[0];

                d_ecu[0]=d;

                 // Se calcula la señal DWPM

                SetDCPWM1(d);

        }

}


// Función para definir el ADC

void adc(void)

{

        //configuración del AD

        ADCON0 = 0x1; // canal 0, enable A/D

        ADCON1 = 0x0B;  // V-= VSS, V+ = VDD,del AN0 al AN3 analógico

        ADCON2 = 0X2A; // justificado a la izquierda, 12TAD, Fosc/32 ((Fosc=25MHZ))

}


// Función para definir el PWM
```

```
void PWM(void)
{       //configuración del PWM
        PR2=19;
        CCP1CONbits.DC1B0=0;
        CCP1CONbits.DC1B1=0;
        // Para un ciclo de trabajo inicial 50%
        CCPR1L=0x9;
        //Configuración como salida del pin del CCP1
        TRISCbits.TRISC2=0;
        //Timer 2 arranca, preescalado 1:1
        T2CON=0b00000100;
        //Configuración de CCP1 para operación PWM para 50%
        CCP1CON=CCP1CON |0x3C;
        //configuración del CCPR2 para que active el ADC cada 51.52 µs
        CCPR2H=0x01;
        CCPR2L=0x40; //0x40
        T1CON=0b10000001;
        //Configuración del Special event trigger
        CCP2CON=0x0B;
}
```

9.2 Código Ensamblador

```
#include <p18f2525.h>

#include <adc.h>

#include <pwm.h>

#define num_variable 4

// Definición de las variables de la ecuación en diferencias multiplicadas por 512

#define b0 81

#define b1 -74

#define b2 -80

#define b3 74


#define a0 512

#define a1 -1418

#define a2 1306

#define a3 -400


#define divisor 512


// Definición de la variable referencia (2V) pasada a digital

#define referencia 0x19

// Definición de las variables asignándoles un seudónimo con la dirección.

#define var3 0x103

#define var2 0x102

#define var1 0x101

#define var0 0x100

#define d_ecu2  0x106
```

```
#define d_ecu1 0x105

#define d_ecu0 0x104

#define d1 0x107


// Definición de las funciones del PIC

void adc(void);

void PWM(void);

void SetDCPWM1( unsigned int dutycycle );


// Asignación de memoria a las variables

#pragma idata MisDatos = 0x100

        char variable_z[num_variable]={0, 0, 0, 0};

        char d_ecu[3]={(0), (0), (0)};

        char d = 0;

        long daux=0;

#pragma code


// Programa Principal

void main(void)

{

        adc();

        PWM();


        while(1)

        {

                while (!PIR2bits.CCP2IF);

                PIR2bits.CCP2IF=0;
```

```
            while(ADCON0bits.GO);
// Código ensamblador
    _asm
//Bloque de memoria 1
  movlb 1
// Desplazamiento de datos
 movff    var2,var3  //variable_z[3]=variable_z[2];
 movff    var1, var2  //variable_z[2]=variable_z[1];
movff    var0, var1  //variable_z[1]=variable_z[0];
// Se guarda la conversión y se desplazan dos posiciones con el fin de
quedarnos con 6bits en lugar de 8.
MOVF 0xfc4, 0, 0
RRNCF 0xfe8, 0, 0
RRNCF 0xfe8, 0, 0
ANDLW 0x3f
MOVWF 0x100, 1
// Se resta la referencia (2V) al resultado del ADC para obtener el error
MOVLW 0x19
BSF 0xfd8, 0, 0
SUBFWB var0, 0, 1
MOVWF var0, 1

//Se calcula de la ecuación en diferencias
MOVLW 0xc
MULWF 0x3, 1
MOVFF 0xff3, 0x18
MOVLW 0xf1
MULWF 0x2, 1
```

```
MOVFF 0xff3, 0x17

MOVLW 0xf5

MULWF 0x1, 1

MOVFF 0xff3, 0x16

MOVLW 0xf

MULWF 0, 1

MOVF 0xff3,0, 0

ADDWF 0x16, 0, 0

ADDWF 0x17, 0, 0

ADDWF 0x18, 0, 0

MOVWF 0x14, 0

CLRF 0x15, 0

BTFSC 0x14, 0x7, 0

SETF 0x15, 0

MOVLW 0x73

MULWF 0x5, 1

MOVF 0xff3, 0, 0

MOVWF 0x1f, 0

CLRF 0x20, 0

MOVF 0x4, 0, 1

MOVWF 0x1d, 0

CLRF 0x1e, 0

MOVLW 0x93

MOVWF 0x8, 0

MOVLW 0xfd

MOVWF 0x9, 0

MOVFF 0x1d, 0xd
```

```
MOVFF 0x1e, 0xe
MOVFF 0x9, 0x13
MOVFF 0x8, 0x12
MOVF 0x8, 0, 0
MULWF 0xd, 0
MOVFF 0xff4, 0x7
MOVFF 0xff3, 0x6
MOVF 0x9, 0, 0
MULWF 0xe, 0
MOVFF 0xff4, 0x9
MOVFF 0xff3, 0x8
MULWF 0xd, 0
MOVF 0xff3, 0, 0
ADDWF 0x7, 1, 0
MOVF 0xff4, 0, 0
ADDWFC 0x8, 1, 0
CLRF 0xfe8, 0
ADDWFC 0x9, 1, 0
MOVF 0x12, 0, 0
MULWF 0xe, 0
MOVF 0xff3, 0, 0
ADDWF 0x7, 1, 0
MOVF 0xff4, 0, 0
ADDWFC 0x8, 1, 0
CLRF 0xfe8, 0
ADDWFC 0x9, 1, 0
BTFSS 0xe, 0x7, 0
```

```
        GOTO  h
        MOVF 0x12, 0, 0
        SUBWF 0x8, 1, 0
        MOVF 0x13, 0, 0
        SUBWFB 0x9, 1, 0
h:      BTFSS 0x13, 0x7, 0
        MOVF 0xd, 0, 0
        SUBWF 0x8, 1, 0
        MOVF 0xe, 0, 0
        SUBWFB 0x9, 1, 0
        MOVF 0x1f, 0, 0
        SUBWF 0x6, 0, 0
        MOVWF 0x1b, 0
        MOVF 0x20, 0, 0
        SUBWFB 0x7, 0, 0
        MOVWF 0x1c, 0
        MOVLW 0xfa
        MULWF 0x6, 1
        MOVF 0xff3, 0, 0
        MOVWF 0x21, 0
        CLRF 0x22, 0
        MOVF 0x21, 0, 0
        SUBWF 0x1b, 0, 0
        MOVWF 0x19, 0
        MOVF 0x22, 0, 0
        SUBWFB 0x1c, 0, 0
        MOVWF 0x1a, 0
```

```
        MOVF 0x19, 0, 0
        SUBWF 0x14, 0, 0
        MOVWF 0x9, 1
        MOVF 0x1a, 0, 0
        SUBWFB 0x15, 0, 0
        MOVWF 0xa, 1
        CLRF 0xb, 1
        CLRF 0xc, 1
        BTFSS 0xa, 0x7, 1
        BRA sig
        SETF 0xb, 1
        SETF 0xc, 1

        // Se divide entre divisor=> pero desplazando, con el registro C (acarreo)
sig:    rrcf  0x0A,0,1
        rrcf  0x09,0,1
        movwf d1, 1

        // Desplazamiento de datos
         movff    d_ecu1, d_ecu2
        movff    d_ecu0, d_ecu1
        movff    d1, d_ecu0

        //Cálculo de la señal DPWM
        movf  d1, 0,1          // Se mueve d a w
        andlw 0xFC             // Máscara para que queden los bits del 7 al 2
        movwf 0xFBE, 0       // Se mueve el valor de la máscara a CCPR1L
```

```
            RRNCF 0xFBE, 1, 0   //Se  desplaza dos veces para eliminar los bits 0 y 1

            RRNCF 0xFBE, 1, 0

            movlw 0xCF              //Se pone en w el número para hacer la máscara

            andwf 0xFBD, 0, 0     // Se hace la máscara a CCPCON para que los bits 5 y 4 =0

            movwf 0xFBD, 0      // Se guarda en CCPCON

            rlncf d1, 1, 1       // Se mueve 4 veces el valor de d para colocar los bits -
                                    significativos en 5 y 4

            rlncf d1, 1, 1

            rlncf d1, 1, 1

            rlncf d1, 1, 1

            movf d1, 0 ,1          // Se mueve d a W

            andlw 0x30          // Se hace una máscara a d para eliminar todo menos bit 5 y 4

            iorwf 0xFBD, 0, 0   // Se hace una or para modificar solo los bits 5 y 4 del
                                    CCPCON

            movwf 0xFBD, 0     // Se guarda en CCPCON

            // Fin del código ensamblador

        _endasm

            }

}


// Función para definir el ADC

void adc(void)

{

            //configuración del AD

            ADCON0 = 0x1; // canal 0, enable A/D

            ADCON1 = 0x0B;  // V-= VSS, V+ = VDD,del AN0 al AN3 analógico

            ADCON2 = 0X2A; // justificado a la izquierda, 12TAD, Fosc/32 ((Fosc=25MHZ))

}
```

```
// Función para definir el PWM

void PWM(void)

{       //configuración del PWM

        PR2=19;

        CCP1CONbits.DC1B0=0;

        CCP1CONbits.DC1B1=0;

        // para un ciclo de trabajo inicial 50%

        CCPR1L=0x9;

        //configuración como salida del pin del CCP1

        TRISCbits.TRISC2=0;

        //Timer 2 arranca, preescalado 1:1

        T2CON=0b00000100;

        //Configuración de CCP1 para operación PWM para 50%

        CCP1CON=CCP1CON |0x3C;

        //configuración del CCPR2 para que active el ADC cada 51.52 µs

        CCPR2H=0x01;

        CCPR2L=0x40; //0x40

        T1CON=0b10000001;

        //configuración del Special event trigger

        CCP2CON=0x0B;

}
```