UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

TRABAJO DE FIN DE GRADO

CONTROL DIGITAL CON MICROCONTROLADORES

Ignacio Hernández Bardají

Julio 2014

Resumen

En este trabajo se llevará a cabo el desarrollo de un regulador en lazo cerrado sobre un microcontrolador para una aplicación de control de tensión.

Primero se realizará un estudio del mercado de los microcontroladores, teniendo como objetivo principal encontrar aquellos que cumplan las especificaciones deseadas para la implementación de dicho regulador en tiempo real.

En segundo lugar se procederá al propio diseño y desarrollo del regulador en el microcontrolador TMS320F2808 de Texas Instruments. Para ello se utilizará el entorno de trabajo adecuado para poder trabajar con él.

Por último se realizarán una serie de pruebas con el fin asegurar la funcionalidad del diseño y de demostrar mediante la comparación con resultados de otros programas su correcto funcionamiento.

Abstract

In this essay it will be developed a closed-loop system in a microcontroller for a voltage tension control application.

At first, it will be carried out a market survey on microcontrollers, being its main purpose finding those which accomplish with the desired specifications for the implementation of this regulator in real time.

Secondly, the regulator will be designed and developed in the TMS320F2808 of Texas Instruments microcontroller. Therefore, the ideal working enviroment is essential when working with it.

Last, different series of tests will be conducted to assure the functionality of the design and to prove, by the comparison with the results of other programs, its correct functioning.

Agradecimientos

Este trabajo no habría sido posible sin todas aquellas personas que me han estado apoyando y ayudando a lo largo de toda la carrera, por lo que son más que merecedores de mis agradecimientos.

A mi madre, que ha sido un referente, un orgullo y un ejemplo a seguir durante toda mi vida y sin la cuál no sería ni la mitad de lo que puedo ser a día de hoy. Gran parte del mérito es suyo.

A mi padre y a mi hermana, apoyos incondicionales que siempre han estado ahí y que no han dudado en quererme y ayudarme cuando más lo necesitaba.

A mi tutor Fernando, por tener la paciencia suficiente para realizar este trabajo conmigo y prestarme su ayuda siempre que la he necesitado. Es difícil imaginar este trabajo sin él.

A mis amigos de toda la vida Yeray, Joaquín y Miguel por ser la vía de escape necesaria durante todo este tiempo, mostrándome su apoyo y haciéndome reír cuando más lo he necesitado. Son como mis hermanos.

A mis compañeros y amigos de clase, en especial a Sergio Sánchez, Sergio Díaz, Alberto, Andrea, Bárbara, Tito, Marta, Ana y Roi, porque de una carrera presumiblemente difícil y tediosa han hecho una época divertida y amena.

A toda mi familia en general con mención especial para mi tía Ani, cuyo amor incondicional ha sido un auténtico alivio en los momentos difíciles, y a mis abuelos, mi tío Nacho, mi tía Pili y mi tía Titi que, aunque ya no estén conmigo, seguro que se sienten orgullosos de mí.

Gracias a todos de verdad.

Índice general

1. INTRODUCCIÓN 1

1.1 Dispositivos digitales para control digital 4
1.2 Estructura de la memoria 5

2. ESTADO DEL ARTE 7

3. ESTUDIO DE PRODUCTOS EXISTENTES 11

4. DESARROLLO E IMPLEMENTACIÓN 15

4.1 Características del microcontrolador 15
4.2 Lazo cerrado 17
4.3 Actuación 18
4.4 Sensado 22
4.5 Actualización de la función de transferencia del regulador 24
4.6 Montaje final del sistema 27

5. EXPERIMENTOS Y RESULTADOS 29

5.1 Pruebas PWM 29
5.2 Pruebas timer 31
5.3 Pruebas sobre circuitos RC 33
5.3.1 Circuito RC1 34
5.3.2 Circuito RC2 35
5.3.3 Circuito RC3 36
5.4 Pruebas sobre circuitos RLC 37
5.4.1 Circuito RLC1 38
5.4.2 Circuito RLC2 39
5.4.3 Circuito RLC3 40
5.5 Conclusiones sobre los experimentos 41

6. CONCLUSIONES 43

BIBLIOGRAFÍA 45

ANEXO I: LISTA DE ACRÓNIMOS 49

ANEXO II: CÓDIGO IMPLEMENTADO 51

Índice de figuras

FIGURA 1: MICROCONTROLADOR TMS320F2808 15
FIGURA 2: CAPTURA DE PANTALLA DEL CODE COMPOSER ESTUDIO 16
FIGURA 3: FOTOGRAFÍA DE LA PLACA C2000 AL QUE SE HA CONECTADO EL MICROCONTROLADOR 17
FIGURA 4: LAZO CERRADO 18
FIGURA 5: GRÁFICA PWM 19
FIGURA 6: PWM CONFIGURADO 22
FIGURA 7: MONTAJE CONTROLADOR DIGITAL 28
FIGURA 8: SEÑAL DE PWM GENERADA PARA OBTENER UNA TENSIÓN EQUIVALENTE DE 1 V 30
FIGURA 9: SEÑAL DE PWM GENERADA PARA OBTENER UNA TENSIÓN EQUIVALENTE DE 2 V 30
FIGURA 10: RESULTADO DE CONFIGURAR EL TIMER A 40 KHZ 32
FIGURA 11: RESULTADO DE CONFIGURAR EL TIMER A 400 HZ 32
FIGURA 12: CIRCUITO RC 33
FIGURA 13: RESPUESTA A UN ESCALÓN CALCULADO POR MATLAB PARA EL CIRCUITO RC1 34
FIGURA 14: CAPTURA DE OSCILOSCOPIO PARA EL CIRCUITO RC1. LA LÍNEA AMARILLA ES LA CONSIGNA, LA AZUL ES EL PWM Y LA ROSA ES LA SALIDA DE LA PLANTA 34
FIGURA 15: RESPUESTA A UN ESCALÓN CALCULADO POR MATLAB PARA EL CIRCUITO RC2 35
FIGURA 16: CAPTURA DE OSCILOSCOPIO PARA EL CIRCUITO RC2. LA LÍNEA AMARILLA ES LA CONSIGNA, LA AZUL ES EL PWM Y LA ROSA ES LA SALIDA DE LA PLANTA 35
FIGURA 17: RESPUESTA A UN ESCALÓN CALCULADO POR MATLAB PARA EL CIRCUITO RC3 36
FIGURA 18: CAPTURA DE OSCILOSCOPIO PARA EL CIRCUITO RC3. LA LÍNEA AMARILLA ES LA CONSIGNA, LA AZUL ES EL PWM Y LA ROSA ES LA SALIDA DE LA PLANTA 36
FIGURA 19: CIRCUITO RLC 37
FIGURA 20: RESPUESTA A UN ESCALÓN CALCULADO POR MATLAB PARA EL CIRCUITO RLC1 38
FIGURA 21: CAPTURA DE OSCILOSCOPIO PARA EL CIRCUITO RLC1. LA LÍNEA AMARILLA ES LA CONSIGNA, LA AZUL ES EL PWM Y LA ROSA ES LA SALIDA DE LA PLANTA 38
FIGURA 22: RESPUESTA A UN ESCALÓN CALCULADO POR MATLAB PARA EL CIRCUITO RLC2 39
FIGURA 23: CAPTURA DE OSCILOSCOPIO PARA EL CIRCUITO RLC2. LA LÍNEA AMARILLA ES LA CONSIGNA, LA AZUL ES EL PWM Y LA ROSA ES LA SALIDA DE LA PLANTA 39
FIGURA 24: RESPUESTA A UN ESCALÓN CALCULADO POR MATLAB PARA EL CIRCUITO RLC3 40
FIGURA 25: CAPTURA DE OSCILOSCOPIO PARA EL CIRCUITO RC1. LA LÍNEA AMARILLA ES LA CONSIGNA, LA AZUL ES EL PWM Y LA ROSA ES LA SALIDA DE LA PLANTA 40

Índice de código

CÓDIGO 1: INICIALIZACIÓN PWM 20
CÓDIGO 2: CONFIGURACIÓN PWM 20
CÓDIGO 3: PWM ALTA RESOLUCIÓN 21
CÓDIGO 4: INICIALIZACIÓN ADC 22
CÓDIGO 5: TABLAS DE INTERRUPCIONES ADC 22
CÓDIGO 6: CONFIGURACIÓN ADC 23
CÓDIGO 7: INTERRUPCIÓN ADC 23
CÓDIGO 8: CÓDIGO DE LA FUNCIÓN QUE CODIFICA LA CAPTURA DE CONSIGNA, VALOR DE LA PLANTA, CALCULA LA ECUACIÓN DE DIFERENCIAS Y ACTUALIZA EL CICLO DE TRABAJO 24
CÓDIGO 9: INICIALIZACIÓN TIMER 26
CÓDIGO 10: ACTUALIZACIÓN DEL CICLO DE TRABAJO 26
CÓDIGO 11: CONFIGURACIÓN BOTÓN 27

1. INTRODUCCIÓN

El término ingeniería de control se asocia normalmente con modificar el comportamiento de un sistema para que cumpla unas condiciones deseadas. En este tipo de sistemas se incluyen, entre otros, un sistema mecánico, un sistema eléctrico, un sistema de fluido, un sistema térmico, o una combinación de dos o más tipos de sistema. El comportamiento de este tipo de sistemas se describe mediante ecuaciones diferenciales. Con estas ecuaciones y conociendo las entradas del sistema es posible calcular la salida.

Los sistemas dinámicos, o plantas, son sistemas continuos en el tiempo en el que las entradas y salidas son también funciones continuas en el tiempo. De esta manera, los sistemas de control digital tratan de modificar la entrada para conseguir una respuesta deseada. Un ejemplo claro de este tipo de sistemas podría ser un motor electromagnético, donde la entrada es una señal continua (tensión) al igual que la salida (velocidad angular del motor).

Hay dos tipos de sistemas de control: en lazo abierto y en lazo cerrado. El sistema de lazo abierto no tiene en cuenta el valor de su salida a la hora de llevar a cabo el control, ya que en ningún momento el sistema modifica la regulación midiendo la salida del sistema. Si volvemos al ejemplo del motor, sería como aplicar un voltaje a la entrada y esperar que este se moviera, pero sin medir en ningún momento la velocidad. La única manera de hacerlo sería cambiar dicha tensión aplicada a la entrada. Es decir, para determinar la salida del sistema sólo se tendría en cuenta la información proporcionada por un modelo formal del motor. Para mejorar el control de la salida del sistema se introduce un sensor a la salida y una comparación con el valor que se desea alcanzar, de tal manera que se produzca una retroalimentación. A este tipo de control se le denomina de lazo cerrado. La diferencia entre el valor de salida deseado y el valor de salida real es el error. Con este error se calculará la acción necesaria que haga que la salida alcance el punto en el que su valor y el deseado sean iguales.

Una de las mayores ventajas de implementar un sistema en lazo cerrado es que su estabilidad se mantiene a pesar de las interferencias o ruido que puede sufrir. También permiten controlar el tiempo de respuesta y el error en estado estacionario.

En los últimos años los dispositivos digitales han ido sustituyendo a los analógicos de manera progresiva. Las principales causas de este cambio están relacionadas con las mayores prestaciones que ofrecen los dispositivos digitales sobre los analógicos, y con su considerable bajada de precio. Cierto es que los dispositivos analógicos todavía se encuentran presentes en una gran parte de los productos electrónicos actuales, debido principalmente a que su coste sigue siendo inferior y que su desarrollo e implementación en sistemas de mayor tamaño y complejidad es más fácil, pero el futuro se presenta cada vez más favorable para el control digital.

El desarrollo de sistemas de control no ha sido ajeno a este cambio, por lo que actualmente el uso y diseño de reguladores digitales está en franco desarrollo. Para poder entender este cambio, es necesario hacer una valoración de las ventajas e inconvenientes que posee el control digital sobre el analógico.

El control digital nos permite la posibilidad de implementar algoritmos de control más complejos, lo que conlleva una mejora de las características de los controladores lineales, permitiendo incluso abordar nuevos problemas de control que antes se hacían impensables [1]. Poseen una gran flexibilidad debido a la posibilidad de cambiar el algoritmo de control sin la necesidad de realizar ningún cambio sobre el hardware. Esto lo permite la capacidad de reprogramación del dispositivo digital. Por otro lado, la capacidad de monitorización nos permite conectarnos a otros sistemas, reduciendo, por ejemplo, la complejidad en los sistemas distribuidos. También, gracias a esta capacidad de monitorización, la realización de pruebas resulta menos compleja debido a la posibilidad de depuración que permiten los controladores digitales.

Ya que los controladores digitales utilizan un menor número de componentes discretos que los analógicos, como pueden ser resistencias o condensadores, y

emplean una menor potencia, su fiabilidad es mayor. También cabe destacar que los controladores digitales son más fáciles de integrar en un sistema mayor, ya que el único requisito para que se produzca esta integración es que los dispositivos de dicho sistema posean los recursos suficientes para llevar a cabo este control. Es decir, que en el mismo dispositivo en el que está implementado el controlador digital se puedan incluir otras funcionalidades. Por ejemplo, si se implementa en un microcontrolador, que el proceso de control no ocupe el 100% del tiempo de ejecución ni de la memoria para que este pueda realizar otras tareas.

Por último, es necesario mencionar la disminución de la variabilidad debido a cambios y la atenuación de la sensibilidad al ruido. Ya que un componente digital apenas varía al envejecer y no le afectan los cambios medioambientales, esta variabilidad de los controladores digitales se reducirá notablemente, llegando a ser casi inexistente. Por otro lado, ya que la señal digital sólo utiliza dos valores, es muy difícil que el ruido llegue a variar su valor efectivo, lo que reduce notablemente la sensibilidad frente a éste.

En oposición, hay ciertas características de los controladores digitales que caen en desventaja frente a los analógicos. La necesidad de utilizar conversores analógico-digitales (CAD) es una de ellas, ya que los sistemas a regular son analógicos. Esto introduce mayor complejidad al diseño. Otra desventaja a tener en cuenta es el problema de resolución, ya que los datos digitales vienen representados por un número finito de bits. Esto provoca que la resolución de los valores quede truncada. En el caso de los dispositivos analógicos, aunque teóricamente la resolución es infinita, ésta se ve delimitada por el ruido que presenta el sistema.

El uso de los CAD propicia la aparición de retardos en el control, al igual que el retardo introducido por los propios cálculos del control. Ambos hay que tenerlos en cuenta a la hora de calcular el control y de estimar su estabilidad. En último lugar hay que mencionar el menor ancho de banda que poseen respecto a los analógicos, causado en parte también por los retardos anteriormente mencionados, y por el tiempo de cálculo y de actuación.

En este trabajo se llevará a cabo la implementación de un sistema de regulación digital en un microcontrolador. El objetivo de este sistema será la corrección de error de tensión. Para ello diseñaremos un regulador digital basado en un lazo cerrado.

1.1 Dispositivos digitales para control digital

Una alternativa común al uso del microcontrolador es la implementación del control digital mediante el uso de dispositivos de hardware específico, como las FPGA. Para entender sus diferencias, primero acudiremos a una breve definición de cada uno.

Un microcontrolador es un dispositivo digital que contiene un microprocesador junto a elementos periféricos, como bloques de memoria, temporizadores, conversores analógico-digitales y/o contadores. Su principal característica es que opera de manera secuencial, ejecutando una instrucción tras otra [2]. Son dispositivos de propósito general, es decir, su hardware no está diseñado para realizar un tipo de tarea en concreto.

Las FPGA son dispositivos de hardware específico que contienen bloques lógicos cuya interconexión y funcionalidad se puede programar [3]. Operan de manera concurrente, de tal manera que pueden ejecutar toda su lógica a la vez, siendo capaces de realizar más de una operación al mismo tiempo.

Después de esta breve introducción, analizaremos sus características más importantes.

Los dispositivos basados en microprocesadores son reprogramables, dotándoles así de una gran flexibilidad. Dentro del campo de las FPGA la mayoría también son reprogramables, a pesar de ser hardware específico, por lo que la flexibilidad es casi la misma.

Si nos centramos en la capacidad de cálculo, los microprocesadores tienen a priori una mayor capacidad debido a que, a pesar de que las FPGA sean capaces de realizar operaciones complejas, utilizan una lógica de tipo condicional, la cual no es la óptima para el cálculo intensivo. Por el contrario, ya que las FPGA ejecutan toda su lógica simultáneamente y los microprocesadores de manera secuencial, las FPGA obtienen una velocidad de operación mucho mayor. Es decir, las FPGA son capaces de realizar varias tareas de manera simultánea a una alta velocidad mientras que los microprocesadores las realizan secuencialmente pero teniendo la capacidad de realizar tareas de alta complejidad.

El tamaño de ambos tipos de dispositivos es similar, aunque puede variar dependiendo del grado de integración en el que se encuentren. En cuanto al precio la situación es parecida, siendo un poco más económicos los microprocesadores aunque la diferencia se va reduciendo con el paso del tiempo.

Por último cabría mencionar que ambos dispositivos se pueden desarrollar en lenguaje de alto nivel, siendo C el más común para desarrollar los microprocesadores (también existe la posibilidad de usar un código ensamblador) y VHDL para las FPGA. Dado que el lenguaje C es más conocido por los programadores, los microprocesadores adquieren ventaja en este campo.

1.2 Estructura de la memoria

En la sección 2 se presenta un repaso de algunos ejemplos del uso de tecnología digital para la realización de tareas de control. A continuación, en la sección 3, se presenta el resultado de la búsqueda realizada para escoger el microcontrolador adecuado para realizar nuestro control digital, nombrando las compañías más destacadas en el mercado y sus principales familias de microcontroladores, así como sus características más importantes.

En la sección 4 se explica cómo se ha realizado la implementación del lazo cerrado en el microcontrolador, con una breve introducción teórica a esta manera de realizar

el control digital. También se explica el montaje y las características propias este trabajo.

En la sección 5 se exponen las pruebas realizadas para la verificación de nuestro proyecto y, por último, la sección 6 incluye las conclusiones obtenidas tras la realización de este trabajo fin de grado.

2. ESTADO DEL ARTE

En general, aunque casi cualquier ordenador se puede utilizar para el control digital, se tienen que cumplir unas condiciones para que dicho control se pueda llevar a cabo. A día de hoy, la mayoría de aplicaciones de control digital se basan en microcontroladores, aunque también se pueden utilizar dispositivos de hardware específico o procesadores de propósito general.

De estos procesadores, la parte más relevante es la CPU (unidad central de proceso). La CPU incluye, entre otros, la unidad aritmético/lógica (ALU), la unidad de control (CU) y los registros de propósito general. La ALU consiste en la circuitería lógica necesaria para llevar a cabo todas las operaciones aritméticas y lógicas, como pueden ser sumar, restar o comparar números. Algunas de estas unidades son capaces de llevar a cabo multiplicación y división en punto flotante. El CU es el que controla todas las operaciones dentro de la CPU, extrae y decodifica las instrucciones de memoria y controla la ALU. Por último, los registros de propósito general son un conjunto de registros rápidos que se utilizan para llevar a cabo operaciones rápidas dentro de la CPU.

Un microcontrolador consiste en un solo circuito integrado capaz de realizar varias tareas. No obstante existen distintos modelos de microcontroladores especializados según su función que se diferencian en función de la potencia del microprocesador que incluye y de sus periféricos.

Para poder programar controladores en estos dispositivos digitales también es necesario cumplir unos requisitos de software [4]:

- Capacidad de realizar *polling* o sondeo. Consiste en que el dispositivo realice una espera activa de una interrupción. Es decir, que el sistema se encuentre verificando continuamente si ha sucedido el evento que active esta interrupción. Esta técnica es usada en los sistemas de control para que el controlador no realice ninguna operación hasta que el ADC (conversor

analógico-digital) termine su muestreo. Implica un gasto del microcontrolador ya que se realiza una búsqueda activa.

- Sistema de gestión de interrupciones, conectado a un periférico externo concentrador de interrupciones. Esto permite programar el regulador como una rutina de interrupción asociada a un evento periódico. Este evento periódico se puede programar en relación al mismo reloj del microprocesador o a un reloj o contador externo del microcontrolador. Esta interrupción externa tendrá un periodo de actuación tal con el que se haya diseñado el regulador. Cada vez que se ha atendido la interrupción, el ISR (servicio de interrupciones del sistema) devuelve el control al programa principal que puede mantenerse a la espera de que se produzca otra interrupción o realizar otras tareas.

  También se podría realizar la sincronización con el controlador utilizando las propias interrupciones internas del microcontrolador. Así, el algoritmo que define el controlador estaría escrito dentro del propio servicio de rutinas del microcontrolador, el cuál es dependiente del reloj del sistema. La acción es la misma que antes, cuando se produce el evento que hace saltar la interrupción se ejecuta el algoritmo de control y posteriormente le devuelve el control al principal.

  Otra alternativa sería el uso de un reloj a tiempo real externo, de tal manera que cada vez que este alcanzase un cierto valor, como por ejemplo 50 ms, se llevase a cabo la ejecución del algoritmo y el reloj volviese a iniciar la cuenta.

Hoy en día, el control digital con microcontroladores tiene una gran variedad de aplicaciones. Se suele utilizar en sistemas de corrección de factor de potencia, en sistemas de medición, en sistemas mecánicos, en sistemas de alimentación o en sistemas que necesiten regulación ya sea de corriente o de tensión.

Como ejemplo de todas estas aplicaciones, podemos encontrar varios trabajos donde se aplica el uso de microcontroladores al control digital:

- Control digital PID (proporcional integral derivativo) para sistemas térmicos basado en un microcontrolador PIC [5]. Este trabajo se basa en la implementación de un prototipo de control PID digital de bajo coste para sistemas térmicos basado en un microcontrolador PIC que posea una interfaz fácil de usar para introducir parámetros, calibrar y fijar la referencia de temperatura y ser capaz de transferir datos del comportamiento de la temperatura en tiempo real.

- Implementación de un sistema de control digital en un sistema barra-esfera [6]. El objetivo principal de este trabajo es diseñar e implementar un sistema de control digital mediante una tarjeta de adquisición de datos basada en un microcontrolador, utilizando un sistema barra-esfera para mostrar el control.

- Implementación hardware de un sistema de control digital para un sistema de péndulo invertido [7]. En este proyecto se realiza una maqueta de péndulo invertido para experimentar con algoritmos de control en tiempo real implementados en un microcontrolador PIC. También se lleva a cabo la implementación de un sistema regulador para la posición del péndulo.

- Termómetro digital con control de temperatura [8]. En este proyecto se implementa un microcontrolador que controle una fuente de calor, un sensor de temperatura para su medición y un display para poder mostrar los datos medidos.

3. ESTUDIO DE PRODUCTOS EXISTENTES

Como hemos visto, los dispositivos digitales están en pleno desarrollo. Es por esto por lo que el mercado digital, en nuestro caso el mercado de los microprocesadores, ha sufrido una expansión notable, ofreciéndonos una amplia gama de dispositivos de diferentes características y funcionalidades, diseñados cada uno de ellos con unos objetivos concretos.

Dado que en nuestro caso vamos a proceder al desarrollo de un regulador de lazo cerrado, nos centraremos en la búsqueda de microprocesadores que nos proporcionen las características adecuadas para su correcta implementación. En concreto, deberán poder proporcionarnos una frecuencia de conversión alta del módulo ADC y una buena resolución del PWM (modulación de ancho de pulso) usado como DAC, ya que serán los elementos centrales de conversión que utilizaremos, como ya veremos más adelante. También será importante a la hora de buscar el microprocesador que disponga de una placa de desarrollo que nos permita controlarlo y programarlo de manera sencilla.

A la hora de comenzar la búsqueda, nos centraremos en las compañías más relevantes que ofrecen microprocesadores, como pueden ser Intersil Corporation, Linear Technology, Marvell, Microchip, National Semiconductor, Primarion, Silicon Labs, Texas Instruments o Wolfson Microelectrics.

Las características más importantes de las familias de algunas de estas compañías son:

- La familia dsPIC3x [10], de Microchip[1], nos ofrece microcontroladores de arquitectura Harvard modificada con un ancho de 24 bits para las instrucciones y de 16 bits para los datos. Tienen una memoria RAM de 512

[1] http://www.microchip.com

bytes y una tabla de registros de 16x16 bits. El microprocesador alcanza las 30 MIPS, además dispone de 32 fuentes de interrupción.

Poseen tres timers de 16-bit y un módulo UART. Son capaces de generar ocho salidas de PWM con cuatro generadores, cada uno de los cuales posee una frecuencia independiente. Estas salidas pueden trabajar con una resolución de 4,2 ns. Los ADC son de 10 bits de resolución con una tasa de conversión de 2000 KSPS. El máximo de canales de entrada es de 12, permitiendo la conversión simultánea. Tienen cuatro comparadores analógicos con un tiempo de respuesta de 20 ns y un DAC de 20 bits.

- La familia COP912x [11], de la compañía National Semicontuctor[2] (la cual fue absorbida por Texas Instruments), ofrece microcontroladores basados en tecnología CMOS. Tiene una memoria ROM de 768 bytes con buses de 8 bits y una memoria RAM de 64 bytes. Trabajan a 10 MHz con un ciclo de instrucción que puede ser tanto de 2,5 µs como de 2 µs. Tienen un contador multifunción de 16 bits con un PWM y 16 puertos serie I/O.

- La familia 88MC200 [12], de la compañía Marvell[3], posee microcontroladores de núcleo ARM Cortex-M3 de 32 bits, que trabaja a una frecuencia de 200 MHz. Tienen una memoria flash de 8 Mbit, una ROM de 4 KB, una memoria RAM de 512 KB y una SRAM de 4 KB. Tienen dos módulos ADC de 16 bits que trabajan a 250 KSPS y un DAC de 10 bits que opera a 500 KSPS. Poseen cuatro timers de propósito general de 32 bits que soportan 6 canales de PWM.

- La familia TMS320F280x [13], de la compañía Texas Instruments[4] , ofrece microcontroladores basados en la tecnología CMOS que nos permiten trabajar tanto a 100 MHz como a 60 MHz, con un consumo de potencia mínimo. Tienen una CPU de 32 bits, la cual ofrece operaciones MAC

[2] http://www.national.com
[3] http://www.marvell.com
[4] http://www.ti.com

(multiplicador-acumulador) tanto de 16x16 bits como de 32x32 bits con una arquitectura de bus Harvard. Las memorias Flash pueden variar tanto de 128KBx16 hasta 16KBx16 y tiene una memoria SARAM que varía de 18KBx16 hasta 6KBx16. Tienen tres timers de CPU de 32 bits y hasta 16 salidas de PWM o seis salidas de HRPWM (PWM de alta resolución) con una resolución de 150 ps. Poseen cuatro entradas para la captura y seis timers de 32 bits o de 16 bits.

Tienen 16 canales de ADC de 12 bits cada uno, con dos módulos sample-and-hold, permitiendo así conversiones simultáneas. Las tasas de conversión pueden ser de 12.5 MSPS, de 3.25 MSPS o de 3.75 MSPS. Tienen hasta 35 pins GPIO (General purpose input/output) programables individualmente.

- La familia XMC4000 [14], de la compañía Infineon[5], incluye una serie de microcontroladores con un núcleo ARM Cortex M4 con unidad de punto flotante que trabaja en un rango de 80-120 MHz. Tienen una memoria FLASH de 1 MB y una RAM de 160 KB. Poseen cuatro canales PWM de alta resolución (150 ps), cuatro ADC de 12 bits con una tasa de muestreo de 4 MSPS y dos DAC de 12 bits cada uno.

Como se ha podido observar, existe una gran variedad de productos que se adecuan a las diferentes necesidades de cada sistema. Esto permite a los ingenieros poder realizar diseños específicos para diversos fines.

Además de toda la información proporcionada, cada microcontrolador posee diferentes herramientas de desarrollo, tanto de software como de hardware, ofrecidas también por las propias compañías.

[5] http://www.infineon.com

4. DESARROLLO E IMPLEMENTACIÓN

A la hora de llevar a cabo la implementación de nuestro controlador digital en lazo cerrado, necesitamos controlar tanto el PWM, ya que será utilizado como DAC (conversor digital-analógico), como el ADC del microcontrolador. Para llevar a cabo todo este proceso, el primer paso es analizar las características de nuestro microcontrolador.

4.1 Características del microcontrolador

El microcontrolador elegido ha sido el TMS320F2808 (figura 1), de Texas Instruments [13]. Se ha tomado esta elección ya que dicho componente posee las características adecuadas para el desarrollo de nuestro regulador, tiene una placa de desarrollo y un entorno de trabajo que utiliza el lenguaje de programación C.

Figura 1: microcontrolador TMS320F2808

Este microcontrolador trabaja a una frecuencia de 100 MHz o de 60 MHz, con una tecnología CMOS que permite que el consumo de potencia sea mínimo. Su memoria Flash es de 64KBx16 y la SARAM es de 18KBx16. Este dispositivo incluye tres timers en la CPU de 32 bits. Si nos centramos en las una de las características más relevantes para desempeñar el regulador, en concreto en el PWM, este microcontrolador tiene 16 salidas y 6 salidas HRPWM con una tasa de resolución de 150 ps. En relación al PWM, es importante que tenga una buena resolución ya que va a funcionar como actuador de nuestro lazo cerrado. Fijándonos ahora en el ADC, otro de los requisitos fundamentales, este microcontrolador

incluye 16 canales de 12 bits cada uno, con dos módulos sample-and-hold, lo que permite la captura simultánea de varios canales con una tasa de conversión que puede ser tanto de 12,5 MSPS como de 6,25 MSPS o de 3,75 MSPS. El ADC es importante que tenga una alta velocidad de conversión ya que es uno de los elementos que condiciona la frecuencia de actualización del regulador.

Para el desarrollo e implementación hemos elegido el TMS320F2808 Experimenter Kit el cual nos proporciona una placa de desarrollo y un entorno de programación llamado Code Composer Studio™ IDE v3.3 (figura 2) con una versión gratuita del entorno de C28x™ Free 32K Byte. El precio del kit, junto al microcontrolador, es de 89 $.

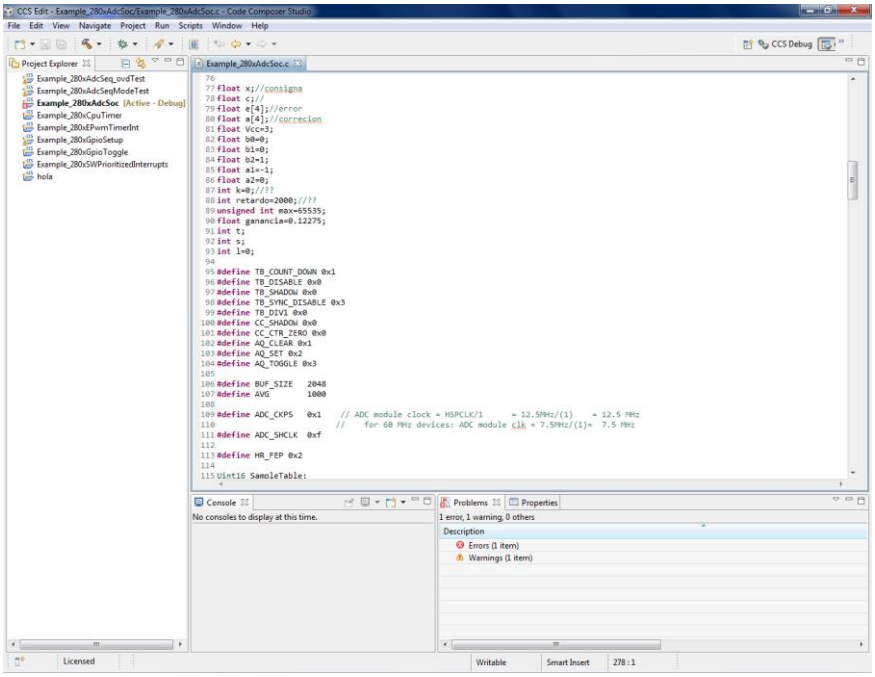

Figura 2: Captura de pantalla del Code Composer Estudio

La placa de desarrollo (figura 3) tiene un módulo *controlCARD* con acceso a todas las señales, áreas protoboard, RS-232, conectores JTAG además de la opción de la emulación JTAG sobre USB, lo que hace posible que no sea necesario tener un emulador JTAG externo. El controlCARD es el módulo a nivel de la placa que

utiliza el estándar DIMM para proporcionar el control de la tarjeta. El software proporcionado incluye algunos programas de ejemplo y algunos detalles del propio hardware.

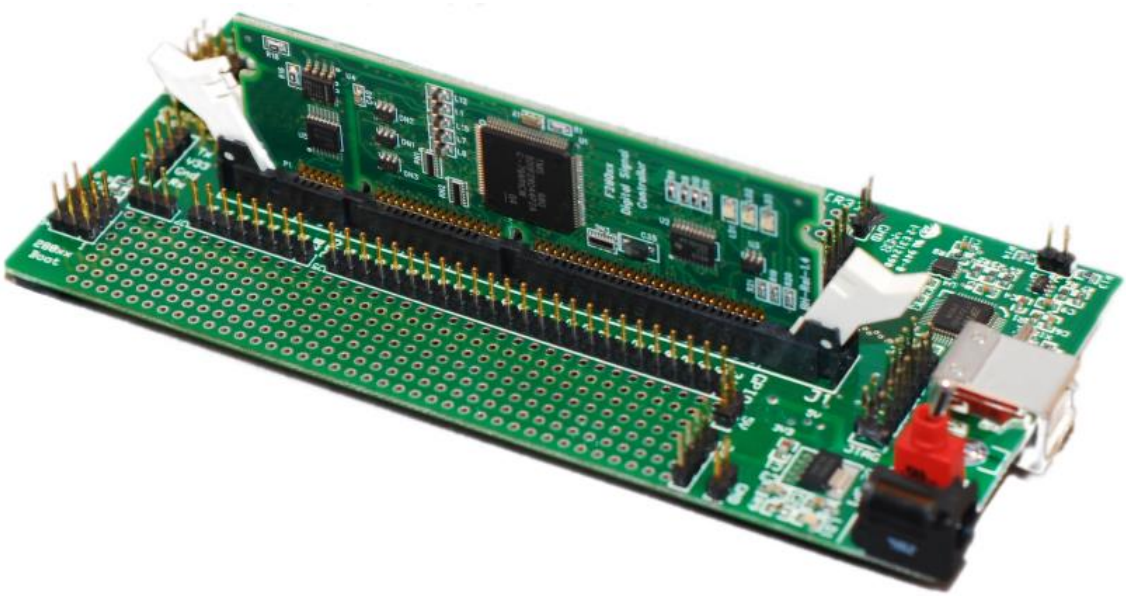

Figura 3: Fotografía de la Placa C2000 al que se ha conectado el microcontrolador

4.2 Lazo cerrado

Después de haber explicado la elección del componente, es necesario realizar una breve introducción al método que vamos a utilizar para la regulación digital: el lazo cerrado.

En este tipo de sistemas, la acción de control viene dada por la salida del sistema, utilizando la retroalimentación para corregir el propio error a la entrada. A pesar de ser un sistema más complejo que el de lazo abierto u otros similares, proporciona una mayor estabilidad y es menos susceptible a variaciones internas. Gracias a la retroalimentación, este tipo de reguladores son los utilizados en aquellos casos en los que el proceso a regular es muy complejo o requiere muchas interacciones.

El esquema general de un regulador en lazo cerrado se muestra en la figura 4.

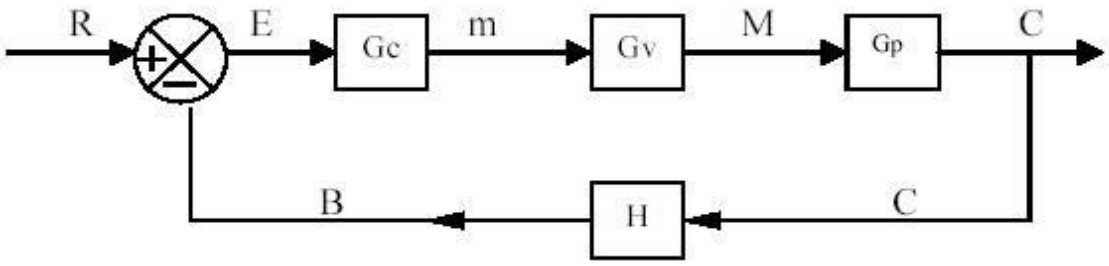

Figura 4: Lazo cerrado

A la entrada del sistema se encuentra la consigna (R), que, en nuestro caso, determina el valor de tensión que se quiere alcanzar, la cual se compara con la lectura de la salida del propio sistema (B). La diferencia existente entre ambas señales es el error cometido (E).

Esta señal de error es la entrada en nuestro regulador (Ge), el cual genera una actuación para la planta (Gp). Esta actuación se calcula en base a los valores anteriores del error y actuación, así como con el propio error actual. La actuación (m) debe pasar por un DAC (Gv) para generar la actuación (M) que llega a la planta (Gp). La señal continua resultante de la planta (C) pasa a través de un captador o ADC (H) para obtener el valor (B).

En la actuación y en el sensado es donde entran en juego los componentes principales de nuestro microcontrolador, el PWM y el ADC.

4.3 Actuación

En la fase de actuación, o el paso de la señal (m) a la (M) en la figura 4, es necesario el uso de un conversor digital-analógico. Para esta tarea se utiliza un PWM, que se presentará a continuación. Acudiendo a su nombre, *Pulse-width Modulation*, es una técnica que se basa en la modificación del ancho activo de una señal periódica. En nuestro caso, ya que utilizamos este componente como un conversor digital-analógico, el valor de la ecuación en diferencias será el que nos dé el porcentaje de ciclo de trabajo, o porcentaje activo, del PWM que necesitamos.

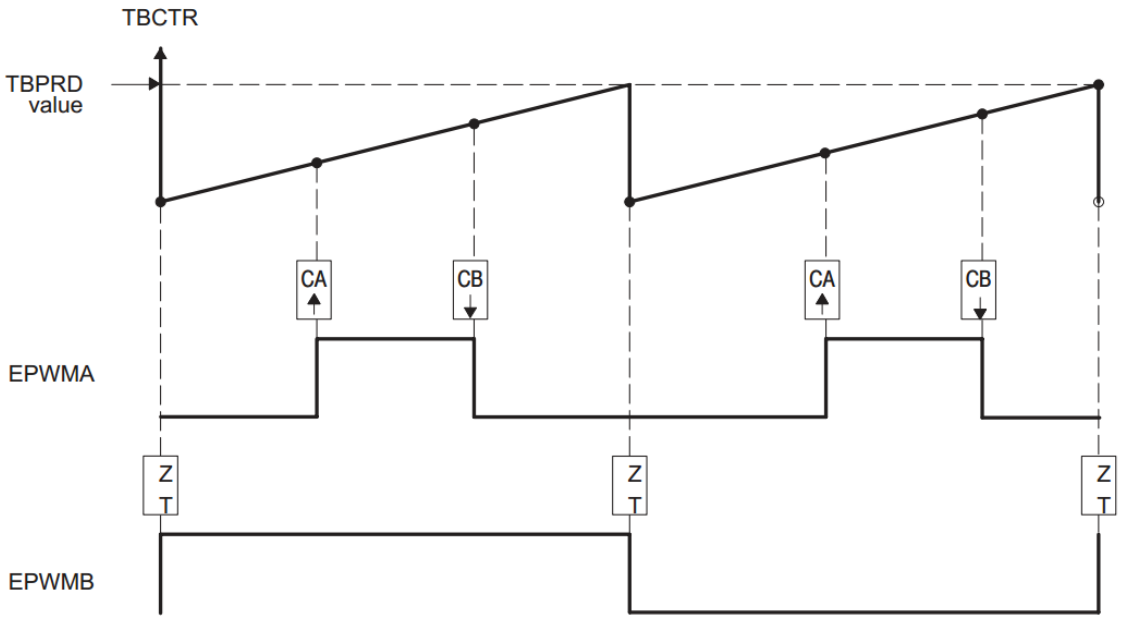

Figura 5: Gráfica PWM

El periodo del PWM (valor del TBPRD en la figura 5) es el límite de un contador que además define el máximo ciclo de trabajo que se puede alcanzar [15]. La señal de las partes activas del PWM se definen utilizando comparaciones entre el contador y distintos índices (CA, CB, ZT en la figura). Junto con los valores de estos índices, se puede configurar el comportamiento del PWM (set, reset o switch). Para que el PWM sea síncrono a flanco de subida, éste se programará de manera descendente (figura 6).

Para generar la salida, tendremos dos valores a comparar, uno que definirá el flanco de subida de nuestra señal de salida (CA en la figura 5) y otro que determinará el flanco de bajada (CB en la figura 5). El valor de CA se establecerá a de manera estática a cero, para que el flanco de subida sea regular y el cambio en el valor CB modificará el flanco de bajada y, por tanto, el ancho del pulso. De ahora en adelante, estos valores a comparar se denominarán con la misma nomenclatura. De esta manera, obtendremos una señal similar a la señal EPWMA mostrada en la figura 6.

A la hora de programar este funcionamiento, lo primero es configurar la salida del PWM en algún GPIO (código 1).

```
1. EALLOW;
2. GpioCtrlRegs.GPAPUD.bit.GPIO0 = 0;
3. GpioCtrlRegs.GPAPUD.bit.GPIO1 = 0;
4. GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1;
5. GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 1;
6. EDIS;
```

Código 1: Inicialización PWM

En el código 1, el comando de la línea 1 habilita la escritura en zonas protegidas de memoria y el comando de la línea 6 la deshabilita. En las líneas 2 y 3 hacemos que estos GPIOs estén configurados con *pull-up*, y en las líneas 4 y 5 elegimos que el multiplexor que los controla habilite ambos como salidas de PWM. De esta manera tenemos el GIPO0 como la señal EPWMA (figura 5) y el GPIO1 como la señal EPWMB (figura 5). Aunque este último no ha sido necesario para el resultado final sí que se ha utilizado para comprobar el correcto funcionamiento del PWM.

Una vez determinados los pines de salida, realizamos la propia configuración del PWM (código 2).

```
7. EPwm1Regs.TBPRD = 99.9;
8. EPwm1Regs.CMPB = 0;
9. EPwm1Regs.CMPA.half.CMPA = 0;
10. EPwm1Regs.TBCTR = 0;
11. EPwm1Regs.TBCTL.bit.CTRMODE = TB_DOWN;
12. EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;
13. EPwm1Regs.AQCTLA.bit.CBU = AQ_CLEAR;
14. EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
15. EPwm1Regs.AQCTLB.bit.ZRO = AQ_TOGGLE;
```

Código 2: Configuración PWM

El periodo del PWM se configura en la línea 7. Ya que vamos a utilizar un HRPWM (cuya configuración se mostrará a continuación), podemos definir este ciclo con decimales, lo cual permite aumentar la frecuencia a la que podemos trabajar. En las líneas 8 y 9 definimos los valores de CB y CA a comparar. En un principio los inicializamos a cero pero, una vez calculado el ciclo de trabajo que queremos, se modificará el valor de CB. En la línea 10 limpiamos el contador TB para evitar problemas con valores utilizados con anterioridad y en la 11 establecemos que el

ciclo de trabajo cuente de manera descendente, deshabilitando el uso del control de fase en la línea 12. En las últimas tres líneas definimos las condiciones que han de ocurrir cuando se cumplan las condiciones del EPWMA. En la primera de ellas establecemos que si el contador llega al valor de CB esta salida se ponga a 0, en la siguiente fijamos que si llega el valor de CA la salida se ponga a 1 y en la última línea, la 15, determinamos que si el ciclo de trabajo termina la salida cambia. Con estas tres últimas condiciones conseguimos que al comienzo de un ciclo de PWM la salida este siempre a 1 y que sea el valor de CB el que controle el ciclo de trabajo con el que se trabaja.

```
16. EALLOW;
17. EPwm1Regs.HRCNFG.all = 0x0;
18. EPwm1Regs.HRCNFG.bit.EDGMODE = HR_FEP;
19. EPwm1Regs.HRCNFG.bit.CTLMODE = HR_CMP;
20. EPwm1Regs.HRCNFG.bit.HRLOAD = HR_CTR_ZERO;
21. EDIS;
```

Código 3: PWM alta resolución

Posteriormente establecemos que esta salida sea de alta resolución (código 3). Para ello ponemos a cero todos los bits del registro de configuración, después establecemos el control de posición de los flancos, definimos que los comparadores sean los que controlen la salida y deshabilitamos cualquier valor cargado anteriormente [16]. Como podemos observar es necesario volver a escribir los comandos de habilitación de escritura. El PWM resultante de esta configuración es el mostrado en la figura 6.

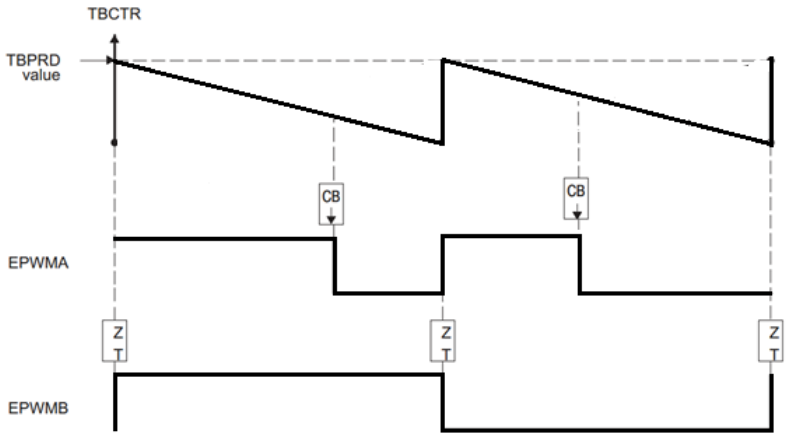

Figura 6: PWM configurado

4.4 Sensado

Una vez configurados todos los parámetros del PWM, pasaremos a establecer la configuración del ADC, ya que es necesario para entender la obtención del valor final del ciclo de trabajo. Como ha ocurrido anteriormente, lo primero es inicializarlo (código 4).

```
22. extern void DSP28x_usDelay(Uint32 Count);
23. AdcRegs.ADCTRL3.all = 0x00E0;
24. DELAY_US(ADC_usDELAY);
```

Código 4: Inicialización ADC

Con estas tres líneas presentadas en el código 4 habilitamos los relojes, los circuitos y el núcleo relacionados con el ADC. El retardo, que en nuestro caso es el definido por defecto, es necesario introducirlo para que a todos los circuitos analógicos les dé tiempo a finalizar la conversión. También es necesario inicializar las tablas de interrupciones (código 5).

```
25. PieCtrlRegs.PIEIER1.bit.INTx6 = 1;
26. IER |= M_INT1;
27. EINT;
28. ERTM;
```

Código 5: Tablas de interrupciones ADC

En el código 5 se muestra cómo habilitamos la tabla de interrupciones 1 del microcontrolador, la interrupción global INTM y la interrupción global de tiempo real DBGM. Ya inicializado, configuramos sus parámetros (código 6).

```
29. AdcRegs.ADCTRL1.bit.ACQ_PS = ADC_SHCLK;
30. AdcRegs.ADCTRL3.bit.ADCCLKPS = ADC_CKPS;
31. AdcRegs.ADCMAXCONV.all = 0x0001;
32. AdcRegs.ADCTRL1.bit.SEQ_CASC = 1;
33. AdcRegs.ADCCHSELSEQ1.bit.CONV00 = 0x0;
34. AdcRegs.ADCCHSELSEQ1.bit.CONV01 = 0x2;
35. AdcRegs.ADCTRL1.bit.CONT_RUN = 1;
36. AdcRegs.ADCTRL2.all = 0x2000;
```

Código 6: Configuración ADC

En la línea 29 establecemos que el reloj del ADC va a estar sincronizado con el reloj del sistema, configurando en la segunda que dicho reloj no este dividido por ningún factor. En la línea 31 establecemos que se van a realizar dos conversiones simultaneas en modo cascada, como indica la línea 32. En las dos siguientes determinamos qué canales vamos a utilizar. Para la primera conversión habilitaremos el canal 0 y para la segunda el canal 2. Por último en la línea 35 configuraremos un modo continuo de captura y en la 36 daremos comienzo a la conversión [17]. Ya configurados los parámetros, esperamos a la interrupción para capturar el dato (código 7).

```
37. while (AdcRegs.ADCST.bit.INT_SEQ1== 0) {}
38. AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;
39. SampleTable = ((AdcRegs.ADCRESULT0>>4) );
40. SampleTable2 = ((AdcRegs.ADCRESULT1>>4) );
```

Código 7: Interrupción ADC

En la línea 37 esperamos a que el bit de fin de conversión se ponga a 1. Cuando esto sucede, limpiamos todos los bits para volver a iniciar la secuencia de captura y guardamos los resultados de los dos buffers de conversión, en las líneas 39 y 40 respectivamente.

4.5 Actualización de la función de transferencia del regulador

Ya con el PWM y el ADC correctamente configurados y en funcionamiento, calculamos el valor que defina el ciclo de trabajo. Para ello primero se captura la consigna, después se calcula la ecuación en diferencias, a partir de ella se obtiene el valor del ciclo de trabajo (código 8) y por último se actualiza el ciclo de trabajo pasándole dicho valor a el timer (código 10).

```
41. float FT()
42. {
43.     float cicloTrabajo=0;
44.     float captura=0;
45.     captura=GpioDataRegs.GPADAT.bit.GPIO21;
46.     c=SampleTable/4096.0;
47.     if(l==0)
48.     {
49.         x=SampleTable2/4096.0;
50.         l=1;
51.     }
52.     if (captura==1)
53.         x=SampleTable2/4096.0; e[k]=x-c;
54.     a[k]=ganancia*(b0*e[k]+b1*e[k-1]+b2*e[k-2])-a1*a[k-1]-a2*a[k-2];
55.     if (a[k]>1)
56.         a[k]=1;
57.     else if(a[k]<0)
58.         a[k]=0;
59.     cicloTrabajo=a[k]*100;
60.     for (t=0;t<3;t++)
61.     {
62.         e[t]=e[t+1];
63.         a[t]=a[t+1];
64.     }
65.     t=0;
66.     return cicloTrabajo;
67. }
```

Código 8: Código de la función que codifica la captura de consigna, valor de la planta, calcula la ecuación de diferencias y actualiza el ciclo de trabajo

En la línea 43 leemos el valor de un GPIO por el que se indica con un 1 la captura de la consigna del regulador. Veremos en la sección siguiente que a este GPIO se le

ha conectado un botón, obteniendo así o un 0 o un 1. Esto nos va a servir para poder cambiar el valor de la consigna de manera dinámica. A continuación guardaremos el valor de la salida de la planta en la variable “c”, leído del primer canal del ADC, dividiéndolo antes por el máximo valor que podemos obtener para tenerlo en tanto por uno. En las siguientes siete líneas es donde vamos a capturar la consigna en la variable “x”, es decir, el valor de tensión que queremos alcanzar. Para ello, configuramos que si es la primera vez que se lee este buffer se coja directamente su valor también en tanto por uno (líneas 47-51) sin tener que pulsar el botón. Una vez leído el primer valor obtenido, habrá que pulsar el botón para volver a capturar la consigna.

Ya con el valor de la consigna y el proporcionado por el ADC, pasamos a calcular el error cometido y la ecuación de diferencias. En la línea 53, calculamos este error como la diferencia entre la consigna y el valor obtenido del sensado. Este error se irá guardando en un vector de cuatro posiciones, al igual que los valores obtenidos de la función de transferencia, la cual se calculará con los valores de ambos vectores y con la ayuda de la herramienta Sisotool de Matlab. Posteriormente se actualizarán ambos vectores de tal manera que cada valor nuevo leído entrará en la posición tercera del vector (la variable k está definida con este valor) y los valores que había anteriormente en dicho vector se desplazarán un índice inferior. Una vez calculada pasamos a obtener el valor del ciclo de trabajo, es decir, CB. En las líneas 55-58 nos aseguramos de si el valor de la actuación obtenido ha superado el máximo o el mínimo se establezca en dichos máximo o mínimo. A continuación, en la línea 59, convertimos el valor de la actuación en un valor porcentual. Esto es debido a que, como explicamos anteriormente, el valor máximo al que llega el ciclo de trabajo es 99.9, y dado que a está en tanto por uno, es necesaria esta multiplicación para que se corresponda con los valores reales de este ciclo de trabajo. En las líneas 60-64 realizamos el desplazamiento de los valores de los vectores anteriormente mencionado. Por último el valor obtenido será pasado a la función del timer, que será la que lo establezca definitivamente como el valor de CB.

```
68. void  ConfigCpuTimer(struct  CPUTIMER_VARS  *Timer,  float
    Freq, float Period)
69. {
70.     Uint32  temp;
71.     Timer->CPUFreqInMHz = Freq;
72.     Timer->PeriodInUSec = Period;
73.     temp = (long) (Freq * Period);
74.     Timer->RegsAddr->PRD.all = temp;
75.     Timer->RegsAddr->TPR.all  = 0;
76.     Timer->RegsAddr->TPRH.all  = 0;
77.     Timer->RegsAddr->TCR.bit.TSS = 1;
78.     Timer->RegsAddr->TCR.bit.TRB = 1;
79.     Timer->RegsAddr->TCR.bit.SOFT = 1;
80.     Timer->RegsAddr->TCR.bit.FREE = 1;
81.     Timer->RegsAddr->TCR.bit.TIE = 1;
82.     Timer->InterruptCount = 0;
83. }
```

Código 9: Inicialización timer

Este timer se programa para establecer la frecuencia a la cual se va a actualizar nuestro valor del ciclo de trabajo. Como todo lo utilizado anteriormente, habrá que inicializarlo (código 9). Como podemos observar, esta función se llama desde el principal estableciendo únicamente la frecuencia y el periodo al que queremos trabajar. Como resultado el timer nos da una frecuencia de interrupción proporcional a la longitud de la multiplicación de ambos parámetros introducidos anteriormente. Con este código se sincroniza este timer con el propio de la CPU estableciéndose que sus relojes sean los mismos, así como que cada vez que se produzca la interrupción se pare el timer y se reinicialice.

```
84. interrupt void cpu_timer0_isr(void)
85. {
86.     float ciclo;
87.     CpuTimer0.InterruptCount++;
88.     GpioDataRegs.GPADAT.bit.GPIO20 = CpuTimer0.InterruptCount % 2;
89.     ciclo=FT();
90.     EPwm1Regs.CMPB = ciclo;
91.     PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
92. }
```

Código 10: Actualización del ciclo de trabajo

Ya inicializado, se utilizará la función asociada a este timer para actualizar nuestro ciclo de trabajo (código 10). En la línea 87 vamos incrementando la cuenta propia del timer para que salte la interrupción. Las líneas 88 y 89 sirven para comprobar a la frecuencia al que trabaja el timer, ya que con ellas generas una onda cuadrada a través de un GPIO habilitado previamente para ello. En la siguiente línea es donde cogemos el valor obtenido en la función FT (el ciclo de trabajo) y en la 90 lo asignamos a CB, consiguiendo así modificar el ancho de pulso.

4.6 Montaje final del sistema

Para poder llevar a cabo la construcción del regulador digital, será necesario combinar los elementos proporcionados por el microcontrolador y su placa de desarrollo con componentes físicos que se encuentren conectados a entradas y salidas de esta placa.

Como generador de la consigna utilizamos un potenciómetro que nos proporciona un voltaje comprendido entre 0 y 3 V. Su salida se conecta al pin A2 del ADC, cuya configuración ya la hemos visto anteriormente. Para controlar la captura de la consigna dinámicamente utilizamos un botón. La entrada de este botón ira a Vcc y la salida se conecta al GPIO21. Su configuración se muestra en el código 11.

```
93.  EALLOW;
94.  GpioCtrlRegs.GPADIR.bit.GPIO21 = 0;
95.  GpioCtrlRegs.GPAPUD.bit.GPIO21 = 1;
96.  GpioCtrlRegs.GPAMUX2.bit.GPIO21 = 0;
97.  GpioCtrlRegs.GPAQSEL2.bit.GPIO21 = 3;
```

Código 11: Configuración botón

Primero habilitamos la escritura en zonas de memoria protegida. Después, como vemos en la línea 94, se configura para que sea un pin de entrada, se habilita con *pull-down* y en el multiplexor (línea 95) establecemos que este GPIO no tenga ninguna función especial, es decir, que actúe sólo como un GPIO [18].

La ecuación en diferencias del regulador se programa mediante software, por lo que cualquier cambio en el circuito de la planta conllevará un cambio en esta función

de transferencia. Para poder calcularla se utilizó la herramienta Sisotool de Matlab, que nos proporciona una visión previa de la respuesta que esperamos obtener, y que nos permite diseñar una respuesta más lenta o más rápida y que oscile o que por el contrario no lo haga.

Como conversor digital-analógico utilizamos una salida de PWM del propio microprocesador. Esta salida, configurada en el GPIO00, se conecta a la entrada de la planta. La planta se construye físicamente en una placa con diferentes componentes dependiendo del tipo de circuito sobre el que queramos actuar. La salida de la planta se conecta al pin A0 del ADC. El montaje ya realizado se muestra en la figura 7.

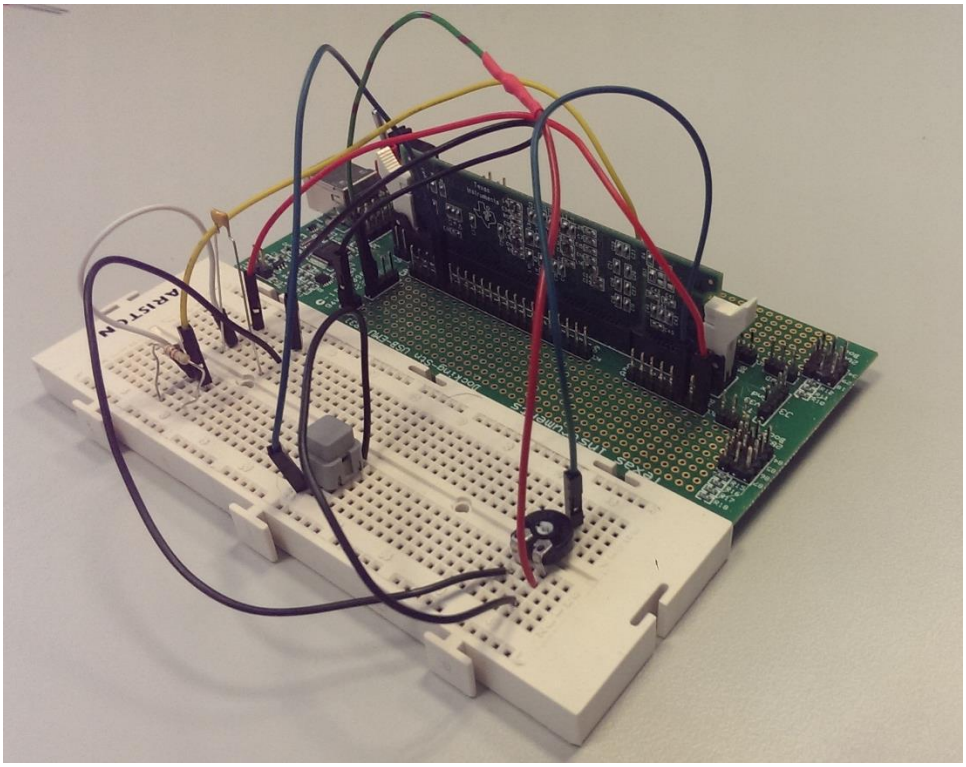

Figura 7: Montaje controlador digital

5. EXPERIMENTOS Y RESULTADOS

Durante el desarrollo del trabajo se han realizado diversas pruebas para poder entender el funcionamiento de los elementos del microcontrolador y, posteriormente, para poder comprobar que el controlador digital implementado actúa de la manera deseada.

Estas pruebas han sido realizadas alimentado la placa con el ordenador a través de un puerto USB y realizando lecturas con un osciloscopio. Una vez alimentada la placa, se utiliza el pin de alimentación de la propia placa para obtener la tensión de 3,3 V (Vcc) que se utiliza en el circuito antes presentado.

5.1 Pruebas PWM

La primera prueba realizada ha consistido en verificar el uso correcto y la configuración del PWM. Se ha configurado el PWM a una frecuencia de 1MHz y se ha probado a generar los anchos de pulso necesarios para obtener una tensión de 1 V y de 2 V. Estas tensiones se corresponden a ciclos de trabajo del 30% y del 60%. Para esta prueba se realizó una pequeña modificación en el código de inicialización del PWM estableciendo el valor de CB a estos valores de manera manual. Los resultados de esta prueba se muestran en la figura 8 (para la generación de un valor de 1 V) y en la figura 9 (para la generación de un valor de 2 V).

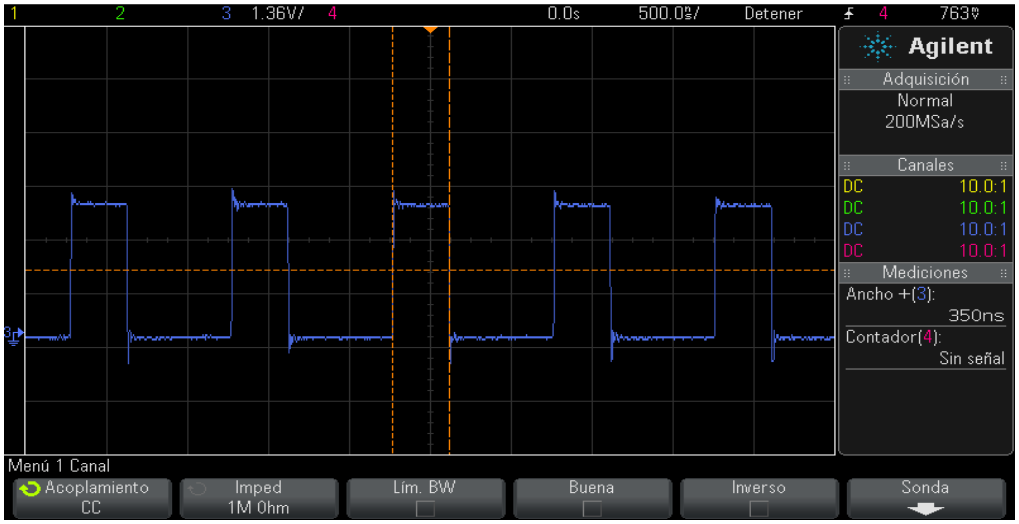

Figura 8: Señal de PWM generada para obtener una tensión equivalente de 1 V

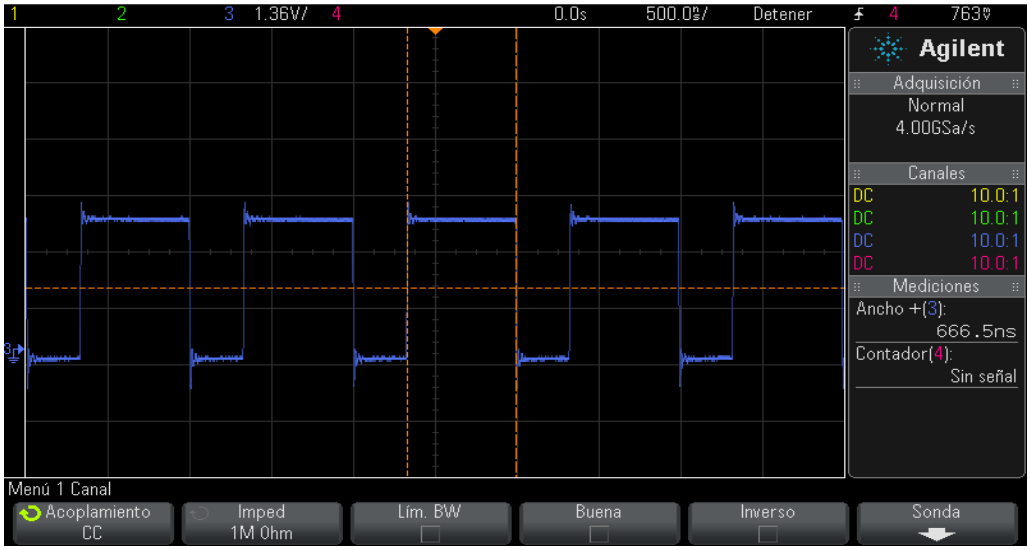

Figura 9: Señal de PWM generada para obtener una tensión equivalente de 2 V

5.2 Pruebas timer

Para establecer la frecuencia a la que se actualiza el regulador, se ha configurado un timer el cuál llama a la función que lo calcula.

A la hora de realizar la prueba se ha creado una señal cuadrada a partir del propio contador interno del timer, como se observa en las líneas 87 y 88 del código 10. Dado que cada flanco de subida o de bajada de esta señal se produce en una interrupción del timer, la frecuencia real de actualización del ciclo de trabajo será el doble de la indicada por el osciloscopio.

La frecuencia del timer se establece con la función vista en el código 9, por lo que para la realización de esta prueba sólo hace falta variar los parámetros de dicha función. Como se aprecia en la figura 10, si esta función se configura de la manera ConfigCpuTimer (&CpuTimer0, 50, 50) obtenemos una frecuencia de 40 KHz, mientras que si cambiamos los parámetros ConfigCpuTimer(&CpuTimer0, 500, 500) la frecuencia desciende hasta los 400 Hz, como muestra la figura 11.

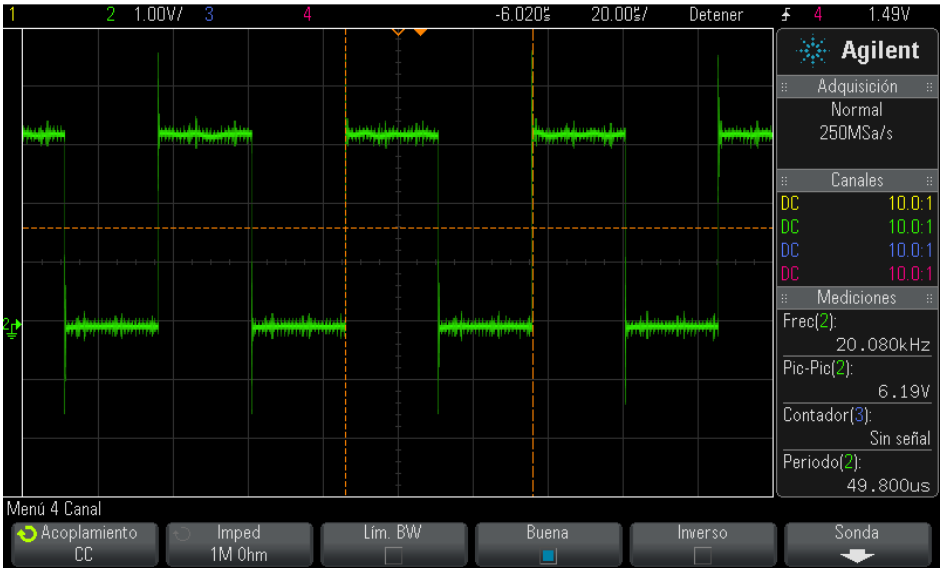

Figura 10: Resultado de configurar el timer a 40 KHz

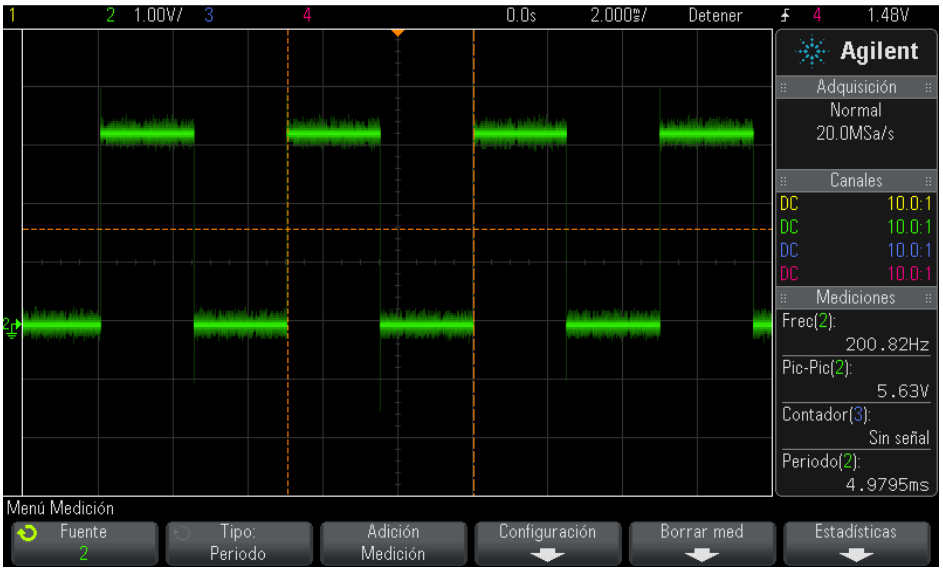

Figura 11: Resultado de configurar el timer a 400 Hz

5.3 Pruebas sobre circuitos RC

La primera batería de pruebas del funcionamiento del controlador digital se ha realizado con un circuito RC como planta (figura 12).

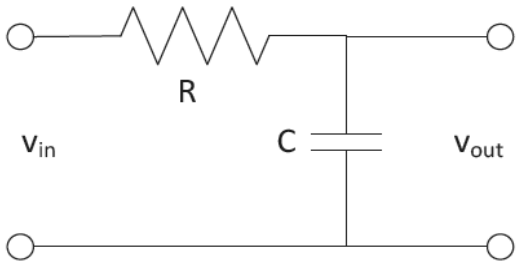

Figura 12: Circuito RC

A la hora de realizar la prueba, primero se eligen los componentes, después se simula la respuesta con la herramienta Sisotool y por último se realizan las medidas a la salida de la planta con el osciloscopio. El objetivo es comparar la respuesta teórica obtenida en la simulación con la respuesta real dada por el circuito.

En las simulaciones, la línea punteada es la planta y la línea continua es la respuesta del regulador. En las capturas del osciloscopio, la línea azul es la señal de salida del PWM, la línea amarilla es la consigna establecida y la línea rosa es la respuesta del regulador

5.3.1 Circuito RC1

El primer circuito ha sido montado con una resistencia de 1 KΩ y un condensador de 96 nF. En la simulación hemos obtenido una función de transferencia del regulador de la forma $FDT = 0{,}090856\ \frac{1}{(z-1)z}$.

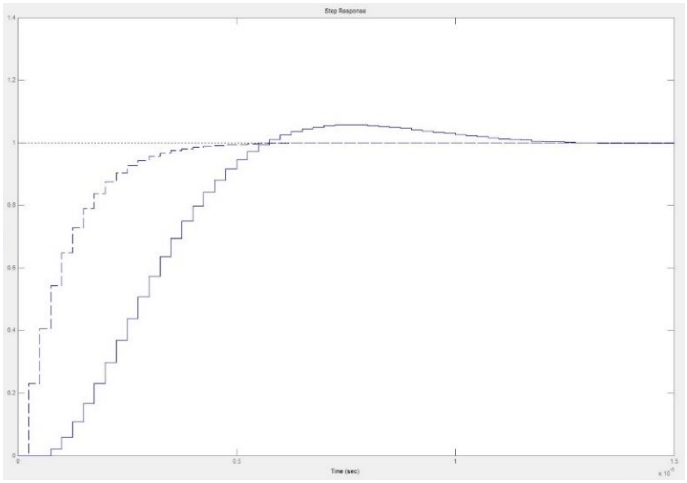

Figura 13: Respuesta a un escalón calculado por Matlab para el circuito RC1

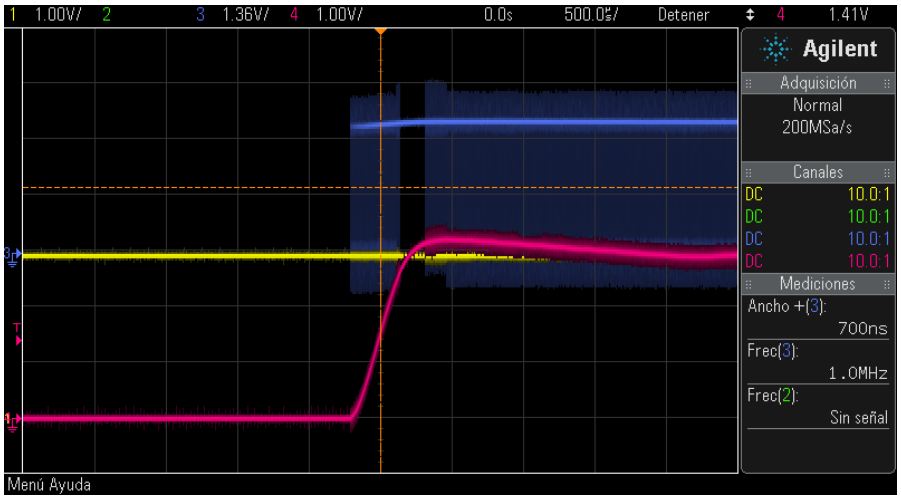

Figura 14: Captura de osciloscopio para el circuito RC1. La línea amarilla es la consigna, la azul es el PWM y la rosa es la salida de la planta

5.3.2 Circuito RC2

El segundo circuito ha sido montado con una resistencia de 9,75 KΩ y un condensador de 96 nF. En la simulación hemos obtenido una función del regulador de la forma $FDT = 0{,}05\ \frac{z-0.8}{(z-1)z}$.

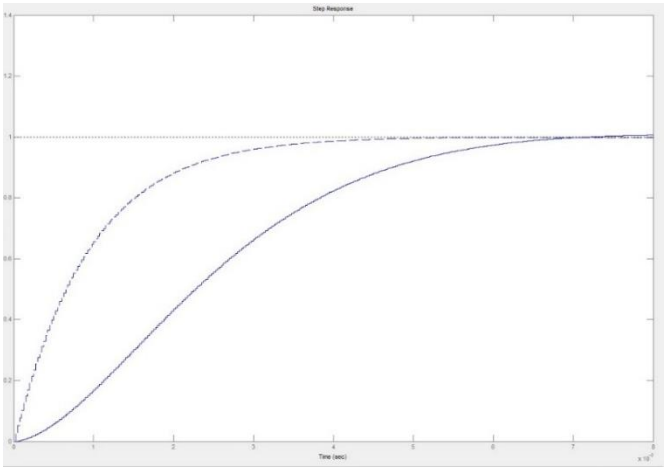

Figura 15: Respuesta a un escalón calculado por Matlab para el circuito RC2

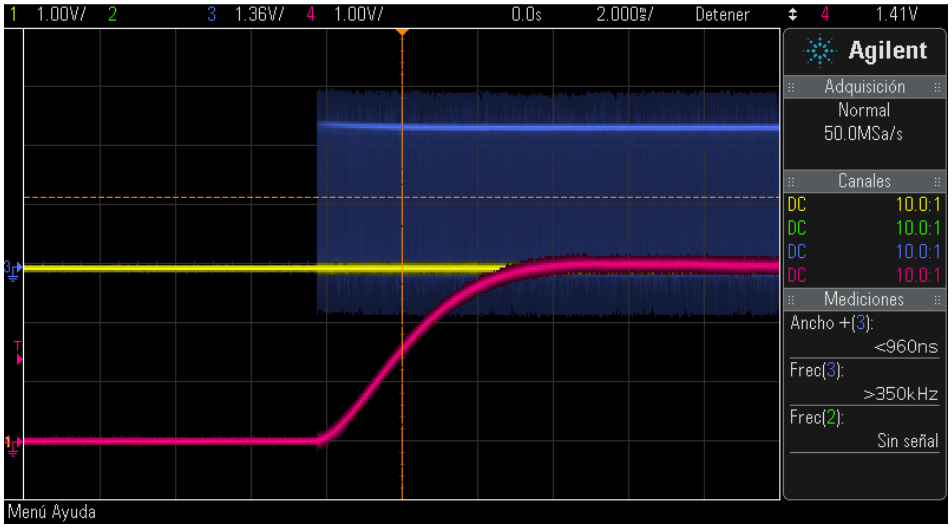

Figura 16: Captura de osciloscopio para el circuito RC2. La línea amarilla es la consigna, la azul es el PWM y la rosa es la salida de la planta

5.3.3 Circuito RC3

El tercer circuito ha sido montado con una resistencia de 3,2 KΩ y un condensador de 10 nF. En la simulación hemos obtenido una función del regulador de la forma $FDT = 0{,}12275\ \frac{1}{(z-1)z}$.

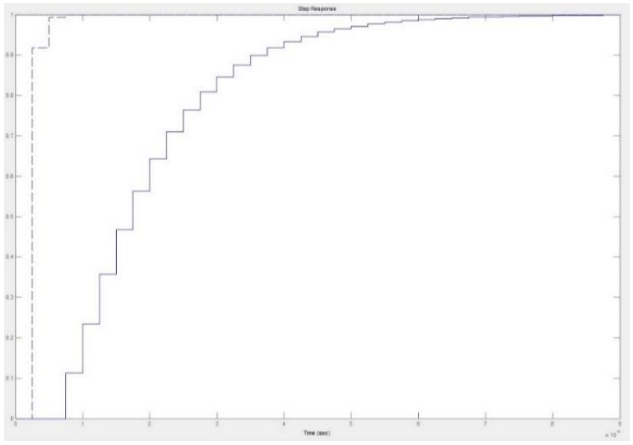

Figura 17: Respuesta a un escalón calculado por Matlab para el circuito RC3

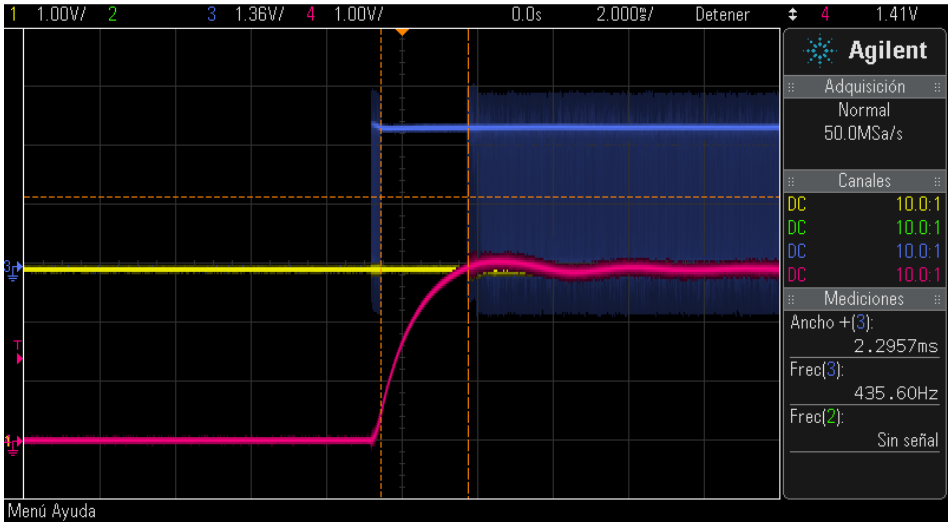

Figura 18: Captura de osciloscopio para el circuito RC3. La línea amarilla es la consigna, la azul es el PWM y la rosa es la salida de la planta

5.4 Pruebas sobre circuitos RLC

La segunda batería de pruebas del funcionamiento del controlador digital se ha realizado con un circuito RLC como planta (figura 19).

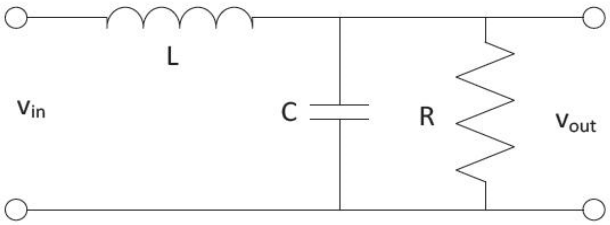

Figura 19: Circuito RLC

A la hora de realizar la prueba, primero se eligen los componentes, después se simula la respuesta con la herramienta Sisotool y por último se realizan las medidas a la salida de la planta con el osciloscopio. El objetivo es comparar la respuesta teórica obtenida en la simulación con la respuesta real dada por el circuito.

En las simulaciones, la línea punteada es la planta y la línea continua es la respuesta del regulador. En las capturas del osciloscopio, la línea azul es la señal de salida del PWM, la línea amarilla es la consigna establecida y la línea rosa es la respuesta del regulador.

5.4.1 Circuito RLC1

El primer circuito ha sido montado con una resistencia de 1 KΩ, un condensador de 10 nF y una bobina de 10 mH. En la simulación hemos obtenido una función del regulador de la forma $FDT = 0{,}15\ \frac{1}{(z-1)z}$.

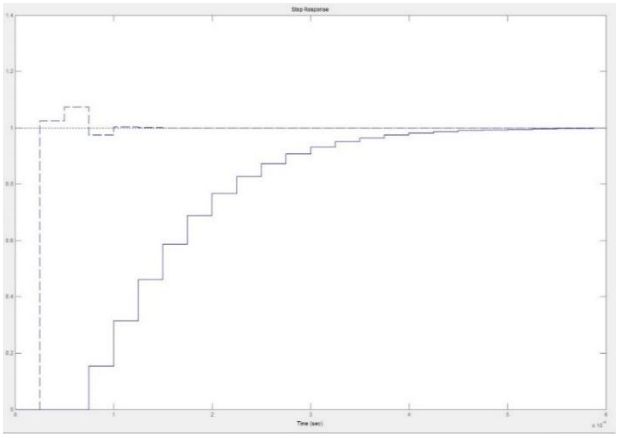

Figura 20: Respuesta a un escalón calculado por Matlab para el circuito RLC1

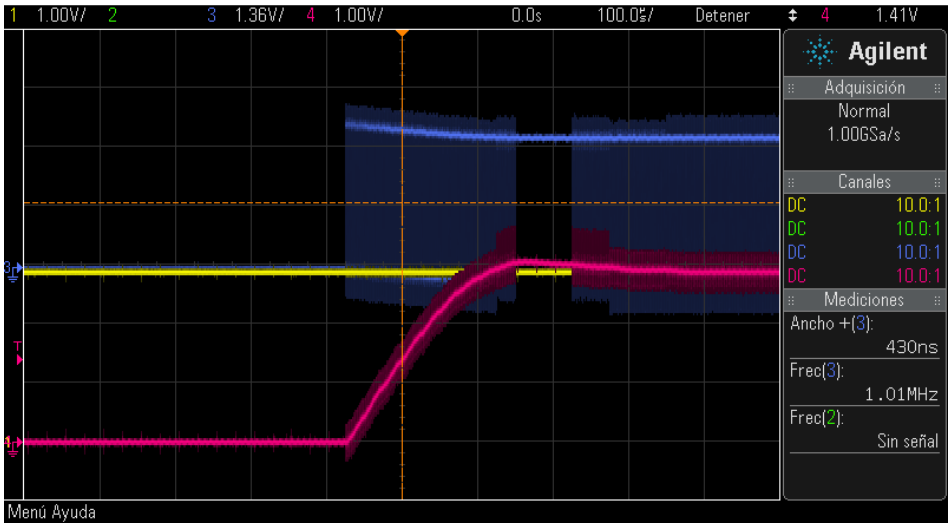

Figura 21: Captura de osciloscopio para el circuito RLC1. La línea amarilla es la consigna, la azul es el PWM y la rosa es la salida de la planta

5.4.2 Circuito RLC2

El segundo circuito ha sido montado con una resistencia de 1 KΩ, un condensador de 96 nF y una bobina de 10 mH. En la simulación hemos obtenido una función del regulador de la forma $FDT = 0{,}10142\ \frac{1}{(z-1)(z-0107)}$.

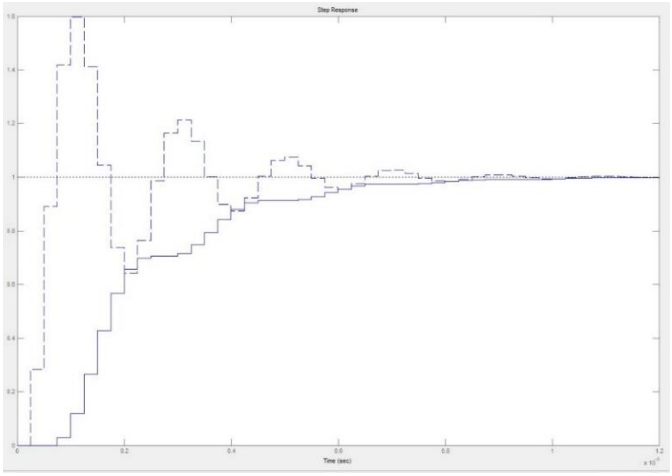

Figura 22: Respuesta a un escalón calculado por Matlab para el circuito RLC2

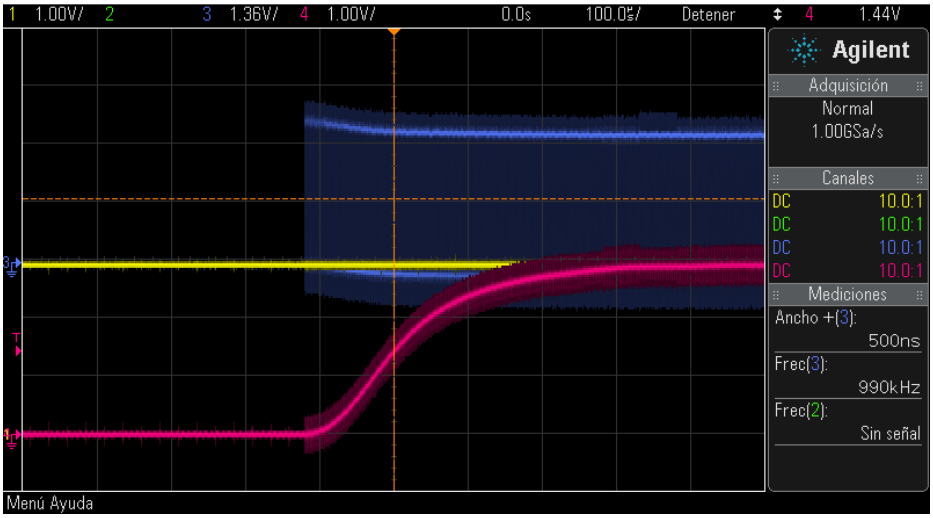

Figura 23: Captura de osciloscopio para el circuito RLC2. La línea amarilla es la consigna, la azul es el PWM y la rosa es la salida de la planta

5.4.3 Circuito RLC3

El tercer circuito ha sido montado con una resistencia de 1 KΩ, un condensador de 96 nF y una bobina de 1 mH. En la simulación hemos obtenido una función del regulador de la forma $FDT = 0{,}11352\ \frac{1}{(z-1)z}$.

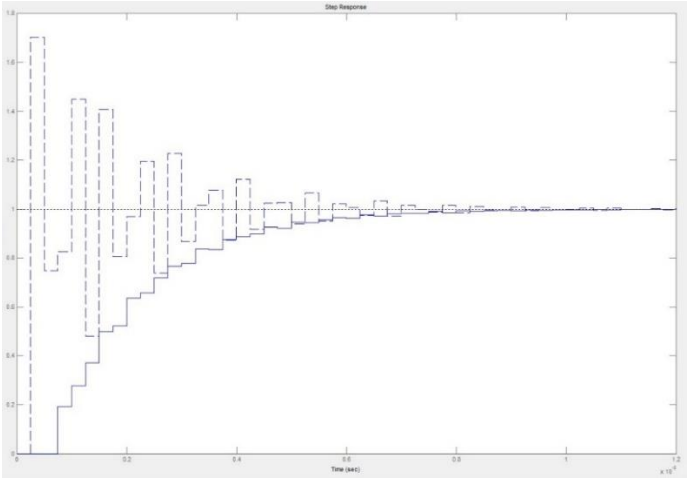

Figura 24: Respuesta a un escalón calculado por Matlab para el circuito RLC3

Figura 25: Captura de osciloscopio para el circuito RC1. La línea amarilla es la consigna, la azul es el PWM y la rosa es la salida de la planta

5.5 Conclusiones sobre los experimentos

A la hora de realizar las pruebas, como ya hemos mencionado con anterioridad, se ha utilizado la herramienta Sisotool de Matlab. Esta herramienta nos permite diseñar la función de transferencia de la planta con los valores de los componentes elegidos, visualizando la respuesta que tendría dicha planta frente a un impulso y modificándola.

Gracias a esto, podemos determinar una respuesta que nos permita observar con claridad el funcionamiento del sistema. En nuestro caso, esta respuesta intentaremos que corresponda a la de un sistema sobre-amortiguado con un tiempo de respuesta significativamente distinto al de la planta para poder apreciar con claridad que nuestro regulador digital actúa de forma correcta.

A pesar de introducir los parámetros obtenidos con Sisotool en nuestro código, la respuesta que obtendremos de nuestro sistema no siempre será exactamente igual, aunque en la mayoría de los casos será muy similar. Esto es debido a que los componentes analógicos, en nuestro caso resistencias, bobinas y condensadores, no siempre se comportan de la manera esperada. Este hecho es causado por su alta sensibilidad al ruido y las variaciones del entorno. Por ejemplo, el sistema que da lugar a la respuesta mostrada en la figura 13 es el único que sufre un poco de amortiguación, lo cual queda reflejado en la respuesta real del sistema que nos muestra el osciloscopio en la figura 14. Sin embargo, si ahora tomamos como referencia el sistema cuya respuesta viene reflejada en la figura 17, observamos que no sufre ningún tipo de amortiguación pero, en la respuesta del sistema real mostrada en la figura 18, se aprecia una ligera amortiguación aunque el tiempo de subida es similar. Este hecho es debido al comportamiento irregular de los componentes analógicos anteriormente mencionado.

6. CONCLUSIONES

En los últimos años los controladores digitales han ido adquiriendo una relevancia mayor, por lo que actualmente cualquier diseño de un controlador que requiera una alta fiabilidad o unas grandes prestaciones es desarrollado mediante la implementación de estos controladores digitales. Para las aplicaciones más sencillas se siguen utilizando los controladores analógicos debido a que su implementación es más fácil y son más baratos, aunque los digitales han disminuido considerablemente de precio.

En los sistemas en los que se necesite tener un control de la salida completo se utilizan los controladores digitales en lazo cerrado. Estos permiten al ingeniero modificar la entrada del sistema para obtener la señal de salida deseada. Esto se consigue gracias a la retroalimentación, ya que permite conocer y corregir el error que se produce entre la señal deseada y la señal de salida.

Los microcontroladores incluyen dos de los elementos más importantes de este tipo de controladores, el PWM, que se puede utilizar como DAC, y el ADC que realiza el sensado. Como consecuencia, a la hora de elegir el microcontrolador a usar es necesario tener en cuenta la resolución del PWM y la frecuencia de conversión que posee el ADC.

En la programación del controlador digital en el microcontrolador existen varias opciones a la hora de actualizar el valor del ciclo de trabajo del PWM, ya que la frecuencia a la que se realiza puede ir sincronizada con el reloj del sistema, con un reloj de tiempo real externo o con un generador de interrupciones externo. En este trabajo, se ha elegido la opción de usar una interrupción externa (un timer) ya que de esta manera resulta más sencilla la tarea de controlar la frecuencia a la que se produce.

En las pruebas hemos comprobado que se ha realizado una correcta implementación del controlador, ya que se aprecia una gran similitud entre los valores teóricos y los

resultados experimentales. Las pequeñas diferencias de comportamiento se deben a las tolerancias de los componentes analógicos.

Bibliografía

[1] A. Castro, “Aplicación de control digital basado en hardware específico para convertidores de potencia conmutados”, Tesis Doctoral, Universidad Politécnica de Madrid, 2003.

[2] A. García Talón, “Control digital de fuentes de alimentación”, Proyecto fin de carrera, Universidad Autónoma de Madrid, 2009.

[3] A. Sánchez González, “Aportaciones mediante implementación basada en sistemas embebidos al control digital de convertidores conmutados”, Tesis Doctoral, Universidad Autónoma de Madrid, 2013.

[4] D. Ibrahim, “Microcontroller Based Applied Digital Control”, Ed. John Wiley & Sons, 2006, ISBN: 0-470-86335-8.

[5] J.M. Díaz Benítez, E.S. Murcia Peraza, “Control digital PID para sistemas térmicos basado en microcontrolador PIC”, Proyecto fin de carrera, Universidad Don Bosco, 2006.

[6] D. Méndez Román, S.I. Ruiz Esparza, “Propuesta para la implementación de un sistema de control digital en un sistema barra-esfera”, Proyecto de fin de carrera, Instituto Politécnico Nacional México D.F, 2008.

[7] J.M. Berga Cabello, “Implementación hardware de un sistema de control digital para un sistema de péndulo invertido”, Proyecto de fin de carrera, Universitat Rovira Virgil, 2012.

[8] C. Girola, “Termómetro digital con control de temperatura”, Proyecto de fin de carrera, Universidad de Belgrano, 2005.

[9] J.A. Noriega Quintero, “Diseño e implementación de un controlador PID digital en un microcontrolador utilizando técnicas de prototipado rápido”, Proyecto de grado, Universidad Pontificia Bolivariana, 2011.

[10] Microchip, “dsPIC30F1010/202X Data Sheet”, 2006 (última consulta Junio 2014), “http://ww1.microchip.com/downloads/en/devicedoc/70178a.pdf”.

[11] Texas Instruments, “COP912C 8-Bit Microcontroller”, 2010 (última consulta Junio 2014), “http://www.ti.com/lit/ds/symlink/cop912c.pdf”.

[12] Marvell, “Marvell 88MC200”, 2013 (última consulta Junio 2014), “http://www.marvell.com/microcontrollers/assets/Marvell_88MC200_SoC-01_product_brief.pdf”.

[13] Texas Instruments, “TMS320F2808 Data Manual”, 2003, revisado 2012 (última consulta Junio 2014), “http://www.ti.com/lit/ds/symlink/tms320f2808.pdf”.

[14] Infineon, “XMC4000 Advanced Microcontrollers for Industrial Solutions”, 2012 (última consulta Junio 2014), “http://www.infineon.com/dgdl/XMC4000_Family_br.pdf?folderId=db3a30433580b3710135a47f3eb76c98&fileId=db3a30433580b3710135a47fff306c99”.

[15] Texas Instruments, “TMS320x280x Enhanced Pulse Width Modulator (ePWM) Module Reference Guide”, 2004 (última consulta Junio 2014), “http://www.sunist.org/shared%20documents/TRANSISTOR,%20IC,%20CIRCUIT%20...%20DATASHEETS/TMS320x280x%20Enhanced%20Pulse%20Width%20Modulator%20(ePWM)%20Module%20Reference%20Guide.pdf”.

[16] Texas Instruments, “TMS320x280x, 2801x, 2804x High Resolution Pulse Width Modulator (HRPWM) Reference Guide”, 2005, revisado 2011 (última consulta Junio 2014), “http://www.ti.com/lit/ug/spru924f/spru924f.pdf”.

[17] Texas Instruments, “TMS320x2833x Analog-to-Digital Converter (ADC) Module Reference Guide”, 2007 (última consulta Junio 2014), “http://www.ti.com/lit/ug/spru812a/spru812a.pdf”.

[18] Texas Instruments, “TMS320C645x DSP General-Purpose Input/Output (GPIO) User’s Guide”, 2005, revisado 2008 (última consulta Junio 2014), “http://www.ti.com/lit/ug/spru724a/spru724a.pdf”.

Anexo I: Lista de acrónimos

ADC	*Analog digital converter*
CPU	*Central processing unit*
DAC	*Digital analog converter*
DIMM	*Dual in-line memory module*
FPGA	*Field Programmable Gate Array*
GPIO	*General Purpose Input/Output*
ISR	*Interrupt Service Routine*
JTAG	*Join test action group*
MAC	*Multiply-Accumulator*
MIPS	*Million instructions per second*
MOSFET	*Metal oxide semiconductor field effect transistor*
MSPS	*Mega samples per second*
PID	*Proportional integral derivative*
PWM	*Pulse width modulation*
RAM	*Random access memory*
SARAM	*Sequential access and random access memory*
UART	*Universal Asynchronous Receiver-Transmitte*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuit*

Anexo II: Código implementado

```
#include "DSP280x_Device.h"
#include "DSP280x_Examples.h"

float FT(void);
interrupt void cpu_timer0_isr(void);

Uint16 LoopCount;
Uint16 ConversionCount;
Uint16 Voltage1[10];
Uint16 Voltage2[10];

float x;
float c;
float e[4];
float a[4];
float Vcc=3;
float b0=0;
float b1=0;
float b2=1;
float a1=-1;
float a2=0;
int k=0;
int retardo=2000;
unsigned int max=65535;
float ganancia=0.12275;
int t;
int s;
int l=0;

#define TB_COUNT_DOWN 0x1
#define TB_DISABLE 0x0
#define TB_SHADOW 0x0
#define TB_SYNC_DISABLE 0x3
#define TB_DIV1 0x0
#define CC_SHADOW 0x0
#define CC_CTR_ZERO 0x0
#define AQ_CLEAR 0x1
#define AQ_SET 0x2
#define AQ_TOGGLE 0x3

#define BUF_SIZE   2048
#define AVG        1000

#define ADC_CKPS   0x1
#define ADC_SHCLK  0xf

#define HR_FEP 0x2

Uint16 SampleTable;
Uint16 SampleTable2;
Uint16 i;
```

```
void main(void)
{


   InitSysCtrl();

   EALLOW;
   SysCtrlRegs.HISPCP.all = 0x4;
   EDIS;
   DINT;

   InitPieCtrl();

   IER = 0x0000;
   IFR = 0x0000;

   InitPieVectTable();

   EALLOW;
   PieVectTable.TINT0 = &cpu_timer0_isr;
   EDIS;

   InitAdc();
   InitEPwm1Gpio();
   InitCpuTimers();


   for (t=0;t<4;t++)
   {
             e[t]=0;
             a[t]=0;
   }
   t=0;

   PieCtrlRegs.PIEIER1.bit.INTx6 = 1;
   IER |= M_INT1;
   EINT;
   ERTM;

   LoopCount = 0;
   ConversionCount = 0;

   AdcRegs.ADCTRL1.bit.ACQ_PS = ADC_SHCLK;
   AdcRegs.ADCTRL3.bit.ADCCLKPS = ADC_CKPS;
   AdcRegs.ADCMAXCONV.all = 0x0001;
   AdcRegs.ADCTRL1.bit.SEQ_CASC = 1;
   AdcRegs.ADCCHSELSEQ1.bit.CONV00 = 0x0;
   AdcRegs.ADCCHSELSEQ1.bit.CONV01 = 0x2;
   AdcRegs.ADCTRL1.bit.CONT_RUN = 1;

   AdcRegs.ADCTRL2.all = 0x2000;



   ConfigCpuTimer(&CpuTimer0, 50, 50);
```

```
    StartCpuTimer0();

    IER |= M_INT1;

    PieCtrlRegs.PIEIER1.bit.INTx7 = 1;

    EINT;
    ERTM;

    EALLOW;
    GpioCtrlRegs.GPADIR.bit.GPIO20 = 1;
    GpioCtrlRegs.GPAPUD.bit.GPIO20 = 0;
    GpioCtrlRegs.GPAMUX2.bit.GPIO20 = 0;
    GpioCtrlRegs.GPADIR.bit.GPIO21 = 0;
    GpioCtrlRegs.GPAPUD.bit.GPIO21 = 1;
    GpioCtrlRegs.GPAMUX2.bit.GPIO21 = 0;
    GpioCtrlRegs.GPAQSEL2.bit.GPIO21 = 3;
    EDIS;

    EPwm1Regs.TBPRD = 99.9;
    EPwm1Regs.CMPB = 0;
    EPwm1Regs.CMPA.half.CMPA = 0;
    EPwm1Regs.TBCTR = 0;
    EPwm1Regs.TBCTL.bit.CTRMODE = TB_DOWN;
    EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;
    EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
    EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
    EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
    EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
    EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
    EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;
    EPwm1Regs.AQCTLA.bit.CBU = AQ_CLEAR;
    EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
    EPwm1Regs.AQCTLB.bit.ZRO = AQ_TOGGLE;

    EALLOW;
    EPwm1Regs.HRCNFG.all = 0x0;
    EPwm1Regs.HRCNFG.bit.EDGMODE = HR_FEP;
    EPwm1Regs.HRCNFG.bit.CTLMODE = HR_CMP;
    EPwm1Regs.HRCNFG.bit.HRLOAD = HR_CTR_ZERO;
    EDIS;
    for(;;)
      {
        for (i=0; i<AVG; i++)
        {
           while (AdcRegs.ADCST.bit.INT_SEQ1== 0) {}
           AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;
           SampleTable =((AdcRegs.ADCRESULT0>>4) );
           SampleTable2 =((AdcRegs.ADCRESULT1>>4) );

        }
      }

}
```

```
float FT()
{
		float cicloTrabajo=0;
		float captura=0;

		captura=GpioDataRegs.GPADAT.bit.GPIO21;

		c=SampleTable/4096.0;
		k=3;
		if(l==0)
		{
			x=SampleTable2/4096.0;
			l=1;
		}
		if (captura==1)
			x=SampleTable2/4096.0;
		e[k]=x-c;
		a[k]=ganancia*(b0*e[k]+b1*e[k-1]+b2*e[k-2])-a1*a[k-1]-
a2*a[k-2];

		if (a[k]>1)
			a[k]=1;
		else if(a[k]<0)
			a[k]=0;

		cicloTrabajo=a[k]*100;
		for (t=0;t<3;t++)
			{
				e[t]=e[t+1];
				a[t]=a[t+1];
			}
		t=0;

		return cicloTrabajo;
}

interrupt void cpu_timer0_isr(void)
{

	float ciclo;
	CpuTimer0.InterruptCount++;

	GpioDataRegs.GPADAT.bit.GPIO20 = CpuTimer0.InterruptCount % 2;

	ciclo=FT();

	EPwm1Regs.CMPB = ciclo;

	PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
}
```