CONVERSION OF AN ANALOG TO DIGITAL PID CONTROLLER AND ITS IMPLEMENTATION WITH PIC MICROCONTROLLERS

CONVERSIÓN DE UN CONTROLADOR PID ANALÓGICO A DIGITAL Y SU IMPLEMENTACIÓN CON MICROCONTROLADORES PIC

MSc. Carlos Alberto Cardona Coy

Universidad Santo Tomás
Facultad de Ingeniería Electrónica
Semillero de Investigación en Automatización, Robótica y Control "ARCON"
Tunja, Boyacá, Colombia.
Móvil: 3118472170
E-mail: carlos.cardona@usantoto.edu.co

Abstract: In the daily work in the control classes as well as in the control seedbed "ARCON" of the USTA Tunja, it has been identified that despite the fact that there is a broad theoretical framework around the control theory, to find developments that show in depth the implementation of the digital controller becomes complicated, so this article shows in detail the digitization, simulation, and programming of the digital controller and the handling of the ADC and PWM modules in PIC microcontrollers.

Keywords: PID Controller, microcontroller PIC, Pulse Width Modulation PWM, Digitization.

Resumen: En el trabajo cotidiano en la clases de control así como en el semillero de control "ARCON" de la USTA Tunja se ha identificado que a pesar de que existe un amplio marco teórico alrededor de la teoría de control, encontrar desarrollos que muestren a profundidad la implementación del controlador digital se hace complicado, así el presente artículo muestra en detalle la digitalización, simulación, y programación del controlador digital y el manejo de los módulos de conversión analógico digital ADC y el de modulación por ancho de pulso PWM en microcontroladores PIC de gama media y alta.

Palabras clave: Controlador PID, microcontrolador PIC, modulación por ancho de pulso PWM, Digitalización.

1. INTRODUCION

El control automático ha desempeñado un papel vital en el avance de la ingeniería y la ciencia. El control automático se ha convertido en una parte e integral en los sistemas de vehículos espaciales, en los sistemas robóticos, en los procesos modernos de fabricación y en cualquier operación industrial que requiera el control de temperatura, presión, humedad, flujo, etc. (Ogata, 2010). La teoría de control ha sido ampliamente desarrollada por múltiples autores, dando como resultado un marco teórico muy completo, sin embargo, en lo que se refiere a la implementación del controlador digital, la información disponible puede quedarse algo corta, ya que es posible encontrar las ecuaciones y métodos matemáticos para obtener un algoritmo base, pero, desarrollos con microcontroladores que muestren el manejo de los módulos de Conversión Analógica-Digital (ADC) y las salidas mediante la modulación por ancho de pulso (PWM), los cuales son la base para la puesta en marcha del controlador en aplicaciones cotidianas, son difíciles de encontrar.

Así, el presente artículo muestra, a partir de simulaciones (Proteus® V8.8), como diseñar, simular, e implementar un código en lenguaje C (PIC C Compiler® V5.015), que permita la realización de un controlador PID digital a partir de un PID analógico, utilizando microcontroladores PIC, explicando detalladamente cómo se configura el módulo CCP como PWM, y cómo éste afecta el funcionamiento del algoritmo del controlador PID digital.

2. EJEMPLO BASE DE DISEÑO

Se tomará como referencia el "Example 9.5 PID Controller Design" (Nise, 2015), el cual muestra el diseño de un controlador PID en el dominio de *s* utilizando el método del Lugar Geométrico de la Raíces LGR, la función de transferencia de la planta es:

$$G(s) = \frac{121.5(s+8)}{(s+3)(s+6)(s+10)} \quad (1)$$

Utilizando MATLAB® podemos observar el comportamiento de la planta sin compensar, como se muestra en la figura 1, la señal de salida presenta un tiempo pico Tp = 0.298 s, con un sobrepaso OS% = 20.7 %, y un tiempo de asentamiento Ts = 0.701 s.

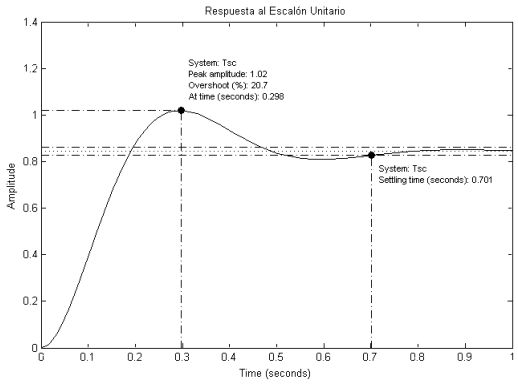


Fig. 1. Respuesta al escalón de la función de transferencia de la planta sin compensar

Se requiere que la planta compensada presente un Tp=0.198 s, con un sobrepaso del 20% y un error en estado estable de cero, tras diseñar el controlador (en tiempo continuo) se obtiene la siguiente función de transferencia para el controlador PID:

$$G_c = \frac{4.6(s+55.92)(s+0.5)}{s} \quad (2)$$

La respuesta de la planta contralada se puede observar en la figura 2, la planta compensada presenta los siguientes valores de operación, tiempo pico Tp = 0.196 s, con un sobrepaso OS% = 14.1%, y un tiempo de asentamiento Ts = 2.56 s.

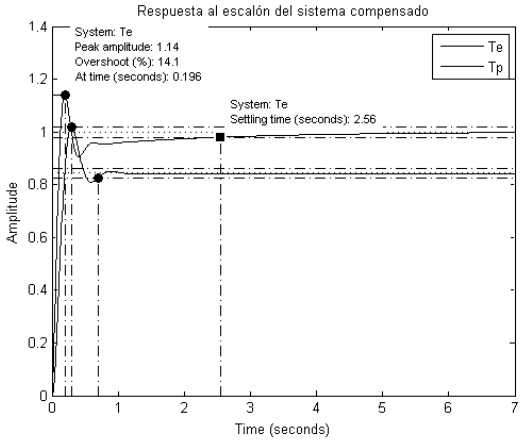


Fig. 2. Respuesta al escalón del sistema compensado (Te).

Podemos concluir que efectivamente se logró obtener una planta compensada con un tiempo pico y un sobrepaso dentro de los valores de diseño requeridos, aunque el tiempo de asentamiento es más lento que el de la planta sin compensar, esto se puede obviar ya que éste no era un criterio de diseño requerido en el ejemplo.

Si se quiere observar a profundidad el ejemplo tratado remítase a la obra de N. Nise, en ella podrá encontrarse en detalle la teoría concerniente al método del LGR, su desarrollo "manual", con del uso de una herramienta de software proveída por el autor, así como a través del Rltool/Sisotool de MATLAB.

3. CONVERSIÓN DEL CONTROLADOR DEL DOMINIO S AL DOMINIO Z

Hasta ahora se han mostrado resultados asociados a desarrollos triviales dentro del estudio del control y el método del LGR, abordaremos ahora una problemática más compleja, como lo es la conversión de la función de transferencia del controlador en el dominio de s al dominio de z, es decir, como digitalizar un controlador analógico llegando a la obtención del algoritmo de control que pueda implementarse en un microcontrolador.

Si se requiere convertir un sistema de control analógico en un sistema de control digital, y se usan las mismas leyes de control, se necesita encontrar algoritmos que se aproximen a la

función de transferencia, G(s), del controlador analógico (Bolton, 2001).

La ecuación (3) presenta la función de trasferencia general del controlador analógico en términos de la ganancia proporcional Kp, la ganancia integral Ki, y la ganancia derivativa Kd.

$$G_{PID}(s) = K_p + K_i\frac{1}{s} + K_d s \qquad (3)$$

Así, ahora debemos llevar la función de transferencia de la ecuación (2) a tener la forma general de la función de transferencia asociada a un controlador PID que se muestra en la ecuación (3), tras realizar las operaciones algebraicas pertinentes se obtiene la función de transferencia que se observa en la ecuación (4), entonces, para el ejemplo tratado en la sección 2 tendríamos: Kp = 259.5, Ki = 128.6, y Kd = 4.6.

$$\frac{4.6(s^2 + 56.42s + 27.96)}{s} = 259.5 + 128.6\,\frac{1}{s} + 4.6\,s \qquad (4)$$

Para digitalizar la función de transferencia del controlador analógico se analizan de manera individual los tres compontes del controlador PID. Un controlador proporcional, G(s) = K, se puede convertir en G(z) = K. Si se tiene un control integral, entonces G(s)=1/s. Para un control integral, se puede usar una ecuación de diferencias para aproximar la integración; así, la salida y[k], a partir del controlador, es y[k]= y[k-1]+Te(k) (Bolton, 2001). Desarrollando la transformada z se obtiene:

$$Y(z) = z^{-1}Y(z) + Te(z) \Rightarrow Y(z) = \frac{Tz}{(z-1)}e(z) \qquad (5)$$

Con un control derivativo, se usa el gradiente de la línea que une el valor actual del error y el valor previo del error para dar la derivada, es decir, salida [k] = (error[k] - error[k-1]) / T (Bolton, 2001); así, tomando su transformada z:

$$Y(z) = \frac{z-1}{Tz}e(z) \qquad (6)$$

De esta manera la función de transferencia general equivalente para el controlador digital corresponde a la aplicación de las ecuaciones (5) y (6), en la ecuación (3), obteniéndose

$$G(z) = K_p + K_i\frac{Tz}{z-1} + K_d\frac{Z-1}{Tz} \qquad (7)$$

Donde e(k) se refiere a la señal de error en tiempo discreto y e(z) corresponde a su transformada z, y *T*, es el periodo de muestreo de la señal, cuya selección es de vital importancia, ya que de éste depende la estabilidad del sistema, la selección del microcontrolador y su cristal de operación.

La ecuación (7) se puede implementar mediante un controlador digital o un microprocesador. Por su puesto, se puede obtener un controlador PI o PID fijando la ganancia apropiada a cero (Dorf y Bishop, 2008).

Se puede reescribir la ecuación (7) para que quede en términos de la constante de tiempo integral, τi = Kp / Ki, y la constante de tiempo derivativa, τd = Kd / Kp,

$$G(z) = K_p + \frac{Tz}{(z-1)\tau_i} + \tau_d\frac{Z-1}{Tz} \qquad (8)$$

La función de transferencia G(z), es la interacción de la salida del controlador U(z) y el error E(z),

$$G(z) = \frac{U(z)}{E(z)} \qquad (9)$$

A partir de la ecuación (9) la ecuación (8) se puede reescribir como

$$U(z) = \left\{K_p + \frac{Tz}{(z-1)\tau_i} + \tau_d\frac{Z-1}{Tz}\right\}E(z) \qquad (10)$$

Al realizar la transformada inversa Z para cada componente de la ecuación (10) se obtiene la ecuación (11), la cual presenta u algoritmo general para un controlador PID digital que se puede implementar por medio cualquier lenguaje de programación.

$$u(k) = K_p e(k) + [K_i T e(k) + i(k-1)] + \frac{K_d}{T}[e(k) - e(k-1)] \qquad (11)$$

Donde *i(k-1)* se refiere a la constante integral en un instante pasado, y *e(k-1)* al error en un instante pasado.

4. CÁLCULO DEL PERIÓDO DE MUESTREO

Para determinar el periodo de muestreo existen varios criterios, uno de los más conocidos es el teorema del muestreo de Shannon, de este se llega a determinar como regla general que el sistema debe ser muestreado entre 8 y 12 veces durante un ciclo de la frecuencia amortiguada de

la respuesta transitoria si el sistema es subamortiguado o entre 8 y 12 veces durante el tiempo de establecimientos de la respuesta si el sistema es sobre amortiguado (García, L., 2012).

Otro criterio útil en la selección del periodo de muestreo plantea que el valor del periodo de muestreo en segundos debe estar entre $0.15 / \omega\phi M$ y $0.5 / \omega\phi M$, donde $\omega\phi M$ es la frecuencia de corte con cero dB en la curva de la respuesta en frecuencia de la magnitud de la función de transferencia equivalente de la planta y el controlador (Nise, 2015).

En el presente desarrollo se utilizó este último método por ser un método sencillo de llevar a cabo en MATLAB y su herramienta LTI Viewer, la cual permite realizar el diagrama de Bode e interactuar con éste para hallar la frecuencia de corte.

Retomando de la sección 2, la función de transferencia equivalente de la planta y el controlador diseñado es:

$$G_{eq}(s) = \frac{4.6(s + 55.92)(s + 0.5)(s + 8)}{s(s + 3)(s + 6)(s + 10)} \quad (12)$$

Como se observa en la figura 3, $\omega\phi M \approx 15.1$ rad⁄s, entonces el periodo de muestreo, *T*, debe estar entre $0.15/\omega\phi M = 0.010$ s y $0.5/\omega\phi M = 0.033$ s; se tomó T=10 ms.

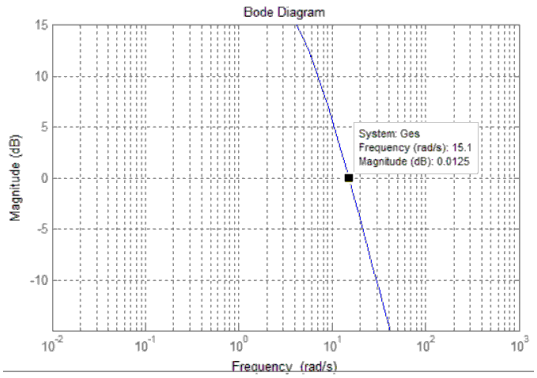


Fig. 3. Diagrama de bode para la función de transferencia equivalente planta/controlador.

5. CONFIGURACIÓN DE LOS PINES DE SALIDA PWM CON PIC C COMPILER®

Una vez obtenido el algoritmo de control, la cuestión es cómo implementarlo sobre un microcontrolador, se trabajó con un microcontrolador PIC por ser la familia más común a nivel mundial, utilizando la referencia 16F877A de gama media; como plataforma de desarrollo se escogió PIC C Compiler, por presentar una interfaz intuitiva para programadores aprendices y su capacidad de interacción para simulación con Proteus®, el lenguaje que maneja PIC C Compiler es C.

Antes de comenzar a presentar el desarrollo del código del controlador PID, es necesario conocer cómo funcionan y se configuran las salidas PWM del microcontrolador con el compilador escogido.

La familia PIC16F87X contiene el llamado módulo CCP, que son pines de entrada/salida que pueden ser configurados como i) Captura, permite temporizar o contar eventos a partir de la captura de flancos de subida o bajada presentes en la entrada del pin en operación; ii) Comparación, permite comparar el valor del registro CCPRx con el del temporizador TMR1, cuando ambos valores son iguales se pueden producir cambios en la salida del pin en operación, o realizar una ADC de forma periódica; iii) PWM, permite obtener, en el pin configurado para ello, un tren de pulsos periódico al que se le pude modificar su ciclo de útil (Duty Cicle DC), es decir, podemos variar el tiempo en el cual la señal está en alto (5V) contra el tiempo en que está en bajo (0V).

El módulo PWM es de nuestro interés porque es la base para el control de cargas, como motores, electroválvulas, hornillos, bombillas, etc. En control digital poder ampliar o reducir el ciclo útil de un PWM de forma automática es el objetivo del algoritmo de control, y se relaciona directamente con el error presente y las constantes del PID, por lo que conocer las características y configuración de los pines PWM del microcontrolador a utilizar es una de los primeros pasos para diseñar y construir el controlador digital.

Con el fin de aplicar del algoritmo obtenido en la ecuación (11), se configuró el módulo PWM para trabajar con un periodo de 1ms, en microcontroladores de la familia PIC16F87x, el periodo de la señal PWM se obtiene de configurar el temporizador 2 (TMR2), y se calcula mediante (García, E., 2014):

$$T_{PWM} = T_{CM} * [Prescaler * (Carga\,TMR2 + 1) * Postscaler] \quad (13)$$

Donde *TPWM* es el periodo del PWM, y *TCM* es el ciclo máquina, con el reloj del microcontrolador trabajando con un cristal de 4 MHz:

$$T_{CM} = \frac{4}{f_{osc}} = \frac{4}{4\,MHz} = 1\,ms \quad (14)$$

El prescaler se refiere a una función que les permite a los temporizadores dividir su frecuencia de operación para obtener mayores periodos de trabajo, el TMR2 permite dividir la frecuencia en 4 o 16.

El postscaler es una función que permite multiplicar la frecuencia de operación del temporizador, para el trabajar con el PWM, el postscaler por defecto es igual a 1.

CargaTMR2 es el valor que se debe guardar en el TMR2 y que equivale, en ciclos máquina, al periodo TPWM, es decir el número que se debe calcular para obtener un periodo (tiempo) determinando, despejando *CargaTMR2* de la ecuación (13), para un periodo de 1 ms la carga debe ser:

$$Carga\,TMR2 = \left[\frac{T_{PWM}}{(T_{CM} * Prescaler * Postscaler)}\right] - 1 = \left[\frac{1\,ms}{(1ms * 4 * 1)}\right] - 1 = 249 \quad (15)$$

Es importante reconocer que la carga no puede ser mayor a 255 ya que el TMR2, para la familia PIC escogida, es de 8 bits, de esto se desprende que el prescaler sea de 4, ya que, con el prescaler de 1 la carga excede 255; aplicando la ecuación (15) con el prescaler=1, la carga para 1 ms da 999.

6. PROGRAMA CONTROLADOR PID Y SIMULACIÓN EN PROTEUS®

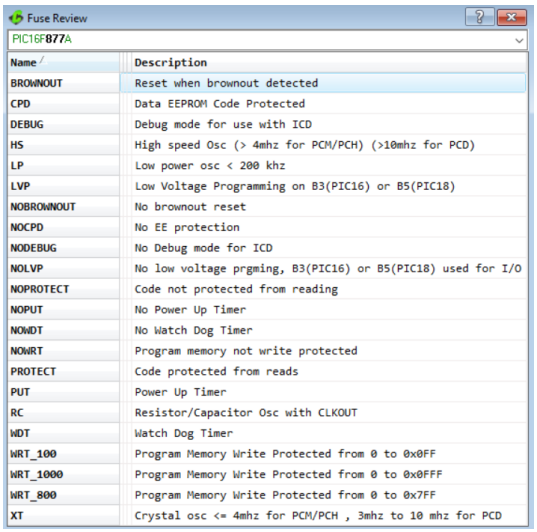
Ya se obtuvo el algoritmo de control y las constantes Kp, Ki, y Kd, así como el periodo de muestreo T y el valor de la carga equivalente a un periodo de trabajo del PWM de 1 ms para el TMR2 asociado a un PIC16F877A, ahora podemos elaborar el código de programa para el controlador PID.

Se comienza por las directivas de preprocesado, donde se incluye la librería asociada al microcontrolador a utilizar, se indica que se va a trabajar el ADC de 10 bits para obtener la máxima resolución posible de este microcontrolador, se continúa configurando los llamados "fuses", donde XT indica que se trabajar con un cristal de entre 3MHz y 10 MHz, y NOWDT indica la Deshabilitación del WactchDogTimer para evitar el reseteo continuo del microcontrolador, finalmente se define el cristal del reloj como uno de 4MHZ.

```
//DIRECTIVAS
#include <16F877A.h>
#device adc=10
#fuses XT,NOWDT
#use delay(clock=4MHz)
```

Fig. 4. Configuración de las directivas de preprocesado.

Es importante conocer las distintas opciones que el microcontrolador en cuestión nos ofrece como "fuses", ya que existen herramientas tan importantes como la protección del código contra sobre escritura entre otras.


Fuse Review

PIC16F877A

Name	Description
BROWNOUT	Reset when brownout detected
CPD	Data EEPROM Code Protected
DEBUG	Debug mode for use with ICD
HS	High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
LP	Low power osc < 200 khz
LVP	Low Voltage Programming on B3(PIC16) or B5(PIC18)
NOBROWNOUT	No brownout reset
NOCPD	No EE protection
NODEBUG	No Debug mode for ICD
NOLVP	No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
NOPROTECT	Code not protected from reading
NOPUT	No Power Up Timer
NOWDT	No Watch Dog Timer
NOWRT	Program memory not write protected
PROTECT	Code protected from reads
PUT	Power Up Timer
RC	Resistor/Capacitor Osc with CLKOUT
WDT	Watch Dog Timer
WRT_100	Program Memory Write Protected from 0 to 0x0FF
WRT_1000	Program Memory Write Protected from 0 to 0x0FFF
WRT_800	Program Memory Write Protected from 0 to 0x7FF
XT	Crystal osc <= 4mhz for PCM/PCH , 3mhz to 10 mhz for PCD

Fig. 5. Cuadro resumen de los "fuses" disponibles para el PIC16F877.

Se prosigue con declarando las variables del programa como se muestran en la fig. 6. El set point, o valor donde se desea que se estabilice la salida está configurado en 1V (S_P=1.0).

```
//FUNCIÓN PRINCIPAL
void main()
{ //Definición de la Variables Locales
int16 valor;
int16 control;
float a,b,c;
float S_P= 1.0;
float max=1000.0;
float min=0.0;
float eT,iT,dT,yT,uT,iT0=0,eT0=0;
float kp= 259.5, kd=4.6, ki=128.6, T=0.010;
```

fig. 6. Declaración de las variables del programa.

La variable "max = 1000.0" corresponde al máximo valor de carga para el DC del PWM trabajando con un periodo de 1 ms, es decir es el valor para el cual la señal del PWM está 100% en alto y no hay señal en bajo, el máximo valor de carga para el PIC 16F877A es 1024, por lo que cargas superiores a 1000 no tendrán ningún efecto sobre la señal de PWM y valores mayores

a 1024 ocasionan error en el funcionamiento del programa, el cual sería difícil de encontrar. La ecuación (16) muestra el cálculo del máximo valor de carga para un PWM con un periodo de trabajo de 1 ms.

$$Carga = DC * [4 * (PR2 + 1)] = 1.0 * [4 * (249 + 1)] = 1000 \quad (16)$$

Donde DC = 1.0, representa un ciclo útil del 100%, y PR2=249, es el valor de carga del TMR2 para obtener un tiempo de 1ms.

La variable "min=0.0" asegura que valores de error negativos no tengan efecto sobre el DC de salida del PWM ya que la carga del DC no puede ser negativa.

A continuación, se introducen las constantes del PID, de acuerdo a la ecuación (11) y los valores obtenidos en la ecuación (4).

```
//Cáculo de las constantes del PID
a=kp;
b=(ki*T);
c=(kd/T);
```

Fig. 7. Constantes de controlador PID, en términos de periodo de muestreo T, y las constantes de tiempo Ti=b y Td =c.

Se prosigue con la configuración de la salida PWM con un periodo de trabajo de 1ms y la entrada ADC por el canal 0.

```
//Configuración del PWM
setup_timer_2(t2_div_by_4,249,1);
setup_ccp1(ccp_pwm);
//Configuración del ADC
setup_adc_ports(AN0);
setup_adc(ADC_CLOCK_INTERNAL);
set_adc_channel(0);
```

Fig. 8. Configuración del PWM y el ADC.

Finalmente se escribe la rutina de control, comenzando por la lectura de la entrada ADC y la conversión de dato en binario a voltaje (ya que el set point está en voltios), se calcula el error, el cual corresponde a la salida deseada menos la salida real (set point menos la realimentación), se prosigue con el cálculo de los términos integral y derivativo, para luego sumarlos con el término proporcional, obteniendo así la salida del PID, que corresponde al valor de carga del ciclo útil del PWM.

El código continúa evaluando si la salida del PID calculada ha excedido el máximo valor de carga del ciclo útil de PWM, o si por el contrario la salida del PID es negativa, si alguna de las condiciones es verdadera se procede a cargar los valores por defecto, max=1000.0 y min = 0.0, para evitar errores en la carga de DC.

La salida del PID (uT) se encuentra guardada en un registro para números con punto flotante, ya que su cálculo requería de la realización de divisiones, pero hay que tener en cuenta que uT equivale al valor de carga del ciclo útil, y éste sólo trabaja con números enteros, por lo que con la instrucción "control=uT", se convierte la información en formato punto flotante a información en formato entero, realizando automáticamente la conversión de los formatos numéricos, despreciando los valor decimales del cálculo de la salida del PID , dejando solo los valores enteros, que ahora estarán guardados en el registro "control".

Se carga el valor del ciclo útil del PWM, para de esta forma actuar sobre la planta, se procede a guardar la constante de integración y el error calculados, convirtiéndolos en la salida integral pasada y el error pasado, esto para poder hacer los cálculos de las constantes del PID en el instante presente para el siguiente ciclo del programa, finalmente se introduce un retardo de 10 ms correspondiente al periodo de muestreo.

```
//Rutina de control
while(true) {

  valor=read_adc();
  yT=valor*5.0/1023.0;
  eT=S_P-yT;
  iT=b*eT+iT0;
  dT=c*(eT-eT0);
  uT=(a*eT)+iT+dT;

  if (uT>max)
  {   uT=max;}
  else if(uT<min)
  {  uT=min;}

  control=uT;
  set_pwm1_duty(control);
  iT0=iT;
  eT0=eT;

  delay_ms(1); //Ajustar
  }
}
```

Fig. 9. Rutina de control del programa.

Para simular el funcionamiento se utilizó Proteus® y sus bloques "Laplace Primitives", los cuales permiten introducir funciones de transferencia en el dominio de s de forma que se pueden simular los controladores analógicos y digitales sin necesidad de tener una planta física, la función de transferencia de la planta sin

compensar (Nise, 2015) para realizar la simulación es:

$$G(s) = \frac{121.5 * (s+8)}{(s+3)(s+6)(s+10)} \quad (17)$$

En la figura 10 puede observarse la conexión de la planta simulada mediante los bloques P+Z REAL y P REAL del "Laplace Primitives" de Proteus®.

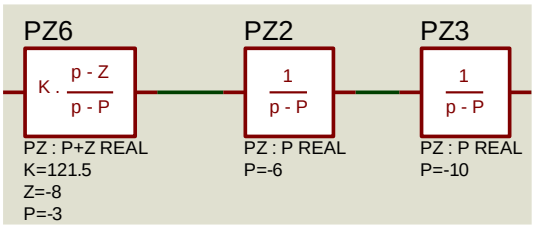


Fig. 10. Conexión de los bloques "Laplace Primitives" para simular la planta.

Finalmente se utilizó la herramienta "Analogue Analisys" de Proteus® para graficar la salida de la planta compensada, como se ve en la fig. 11, el sistema se estabiliza en 1V, con un sobrepaso menor al 20%, comprobando así un funcionamiento aceptable del controlador en la simulación.

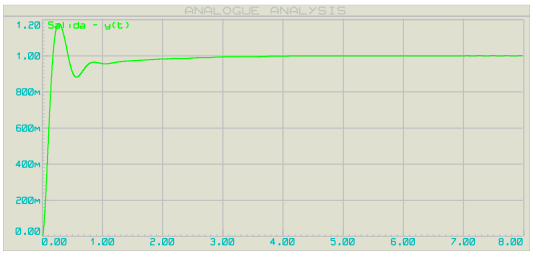


Fig. 11. Respuesta de la planta compensada mediante un controlador digital.

7. CONCLUSIONES

El periodo de muestreo, *T*, es vital para lograr el buen funcionamiento de un controlador PID digital, ya que una selección errónea de *T* puede llevar al sistema a la inestabilidad, además de conducir a una mala selección del cristal de operación del microcontrolador y por tanto de todos los parámetros de configuración del PWM, se aconseja utilizar el criterio de selección de la frecuencia de corte, descrito en la obra de N. Nise, ya que al utilizarla en ejemplos como el aquí descrito se han obtenido buenos resultados para determinar con precisión el periodo de muestreo, mientras que con otras teorías determinar el valor de *T* se vuelve prácticamente una tarea de prueba y error.

De manera similar el periodo de trabajo del PWM, *TPWM*, es muy importante ya que de éste valor depende que la señal de salida puede operar en las condiciones de diseño establecidas, en el ejemplo mostrado en la sección 2 se requería que el tiempo pico, Tp, estuviera cercano a los 196 ms, así trabajar con un *TPWM* de 1 ms permitió trabajar adecuadamente ya que se da el suficiente tiempo para que el microcontrolador ejecute el programa del controlador y ofrezca una señal de salida apta para controlar la planta a las velocidades (tiempos) requeridos, así para planta que operen a tiempo pico o de establecimiento menores a 20 ms se aconsejaría reducir el *TPWM* al orden de los microsegundos, y en consecuencia estudiar la posibilidad de aumentar la velocidad del cristal del microcontrolador a 10 o 20 MHz, e inclusive de usar un microcontrolador de gama alta o un dsPIC con mejores prestaciones.

El procedimiento descrito ha funcionado con eficacia en aplicaciones con controladores PID y PI, sin embargo, presenta problemas en la implementación de los controladores PD, donde se observan problemas para poder operar en procesos con velocidades altas, del orden de os cientos de microsegundos a unos cuantos milisegundos.

En la rutina de control del programa se debe ajustar el retardo final que debería ser equivalente al periodo de muestreo, para el ejemplo 10 ms, sin embargo, hay que tener los tiempos de procesamiento del microcontrolador, para este caso se obtuvo una buena respuesta bajando el retardo a 1 ms, si se utiliza un microcontrolador de gama alta se puede programar usando Sistemas Operativos en Tiempo Real -RTOS- y reducir éste problema.

El uso de tarjetas Arduino es una opción interesante para la implementación de controladores PID digitales, el procedimiento tratado no ha sido probado en estos dispositivos, pero se pronostica que se podría realizar sin mayor dificultad, en la web existe información alrededor del uso de éstas tarjetas y de librerías ya diseñadas, sin embargo por lo general estos procedimientos se refieren métodos de sintonización como el de Ziegler-Nichols, que si bien son ampliamente utilizados, para tareas donde se requieran funcionar a valores de tiempo de asentamiento, tiempo pico, o sobrepaso específicos, requerían de la llamada sintonización fina (Ogata, 2010), cuya aplicación pude llegar a ser engorrosa.

El simulador Proteus se erige como una instrumento muy poderoso en el estudio del control digital, ya que mediante los objetos

"*Laplace Primitives"* permite emular la funciones de transferencia en el dominio de *s*, interactuando en tiempo real con los microcontroladores PIC o con tarjetas Arduino, esto mediante los archivos *.hex* o *.cof* que sus compiladores generan, además los usuarios pueden a través de herramientas como el "*Interactive Analysis*" o el "*Analogue Analaysis*" graficar y comparar las señales de entrada y salida, entre otras; así los diseños de plantas reales, o los ejemplos "ideales" que presentan libros, pueden ser probados, comprobados y depurados de una manera antes casi impensada, disminuyendo los riegos de daños y accidentes que se podrían ocasionar al probar los controladores directamente sobre los prototipos o sistemas físicos.

REFERENCIAS

Bolton, W. (2001). *Ingeniería de Control*, Alfaomega, Segunda edición, México D.F.

Dorf, R, y Bishop, R. (2008). *Sistemas de control moderno*, Pearson - Prentice Hall, Décima edición, Madrid.

García, E. (2014). *Compilador C CCS y simulador PROTEUS para microcontroladores PIC*, Alfaomega, Primera edición, México D.F.

García, L. (2012). *Control digital teoría y práctica*, Politécnico Colombiano, Tercera edición, Medellín.

Nise, N. (2015). *Control sistems Engineering*, Wiley, Séptima edición, California.

Ogata, K. (2010). *Ingeniería de Control Moderna*, Prentice Hall, Quinta edición, Madrid.