Práctica 05.- Implementación de un PID discreto en una planta de segundo orden.

Francisco Javier Díaz Galindo
José Ángel Ortiz León

19 de octubre de 2013

Índice general

1. Introducción **2**

2. Marco Teórico **3**
2.1. Convertidor Analógico-digital (A/D) 3
2.2. Matlab 3
2.3. Arduino MEGA ADK 4
2.4. Método Ziegler-Nichols 4

3. Desarrollo y resultados **6**

4. Conclusiones **9**

Bibliografía **10**

Lista de Figuras **11**

Capítulo 1

Introducción

Prácticamente todos los sistemas de control que se implementan hoy en día están en control computacional. Por lo cual es importante entender perfectamente los sistemas controlados por computadora. Un PID es un mecanismo de control por realimentación que calcula la desviación o error entre un valor medido y el valor que se quiere obtener, para aplicar una acción correctora que ajuste el proceso.

Capítulo 2

Marco Teórico

En este capítulo se presenta información sobre la teoría utilizada para el desarrollo de esta práctica.

2.1. Convertidor Analógico-digital (A/D)

Un convertidor analógico-digital, también conocido como codificador es un dispositivo que convierte una señas analógica en una señal digital, usualmente una señal codificada numéricamente. Dicho convertidor se necesita como una interfaz entre un componente analógico y uno digital. Con frecuencia un circuito de muestre y retención es una parte integral de un convertidor A/D disponible comercialmente. La conversión de una señal analógica en la señal digital correspondiente (número binario) es una aproximación, **ya que la señal analógica puede adoptar un número infinito de valores, mientras la señal digital puede formar un conjunto de dígitos finitos**. Este proceso de aproximación se pueden formar mediante un conjunto finito de dígitos está limitada. [1]

2.2. Matlab

MATLAB es un acrónimo de *Matrix Laboratory*. Actualmente es uno de los entornos de programación más usados en el desarrollo de aplicaciones de ingenieria e investigación.

Su lenguaje de programación esta basado en C y C++. Además cuenta con un entorno gráfico amigable, visualización de datos, funciones, gráficas 2D y 3D, procesamiento de imágenes, video, computación numérica, etc. Particularmente, es una herramienta muy poderosa para realizar aplicaciones en mecatrónica, robótica, control y automatización.[2]

Figura 2.1: Logo de Matlab

2.3. Arduino MEGA ADK

El Arduino ADK es una placa basada en el microcontrolador ATmega2560. Tiene una interfaz host USB para conectarse con celulares basados en Android, usando el IC MAX3421e. Cuenta con 54 pines de entrada/salida digital (15 de ellos pueden usarse como salida PWM), 16 entradas analógicas, 4 UARTs (hardware de puerto serial), un cristal que oscila a 16Mhz, conexión USB, conector de alimentación, un ICSP header y un botón de reset. Además el software de la tarjeta es gratuito y se puede programar desde MATLAB.

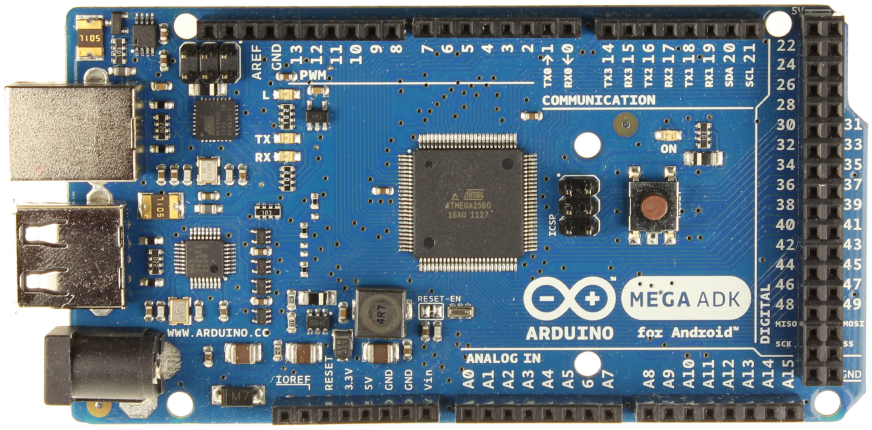

Figura 2.2: Placa Arduino MEGA ADK.

2.4. Método Ziegler-Nichols

El método de Ziegler-Nichols permite ajustar o sintonizar un regulador PID de forma empírica, sin necesidad de conocer las ecuaciones de la planta o sistema controlado. Estas reglas de ajuste propuestas por Ziegler y Nichols fueron publicadas en 1942 y desde entonces es uno de los métodos de sintonización más ampliamente difundido y utilizado. Los valores propuestos por este método

intentan conseguir en el sistema realimentado una respuesta al escalón con un sobrepulso máximo del 25 por ciento, que es un valor robusto con buenas características de rapidez y estabilidad para la mayoría de los sistemas. El método de sintonización de reguladores PID de Ziegler-Nichols permite definir las ganancias proporcional, integral y derivativa a partir de la respuesta del sistema en lazo abierto o a partir de la respuesta del sistema en lazo cerrado. Cada uno de los dos ensayos se ajusta mejor a un tipo de sistema.

Usando el método 1 de Ziegler-Nichols se obtiene lo siguiente.

$$K_p = 1{,}2 \cdot T/L \tag{2.1}$$

$$T_i = 2 \cdot L \tag{2.2}$$

$$T_d = 0{,}5 \cdot L \tag{2.3}$$

Para encontrar la función del controlador PID usamos:

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right) = K_p + \frac{K_i}{s} + K_d \cdot s \tag{2.4}$$

De donde podemos encontrar los valores de K_i y K_d, para expresar la función de PID en términos de s.

$$K_i = \frac{K_p}{T_i} \tag{2.5}$$

$$K_d = K_p \cdot T_d \tag{2.6}$$

Finalmente obtenemos la función de transferencia de lazo cerrado que involucra el control PID y la función H de la planta con retroalimentación.

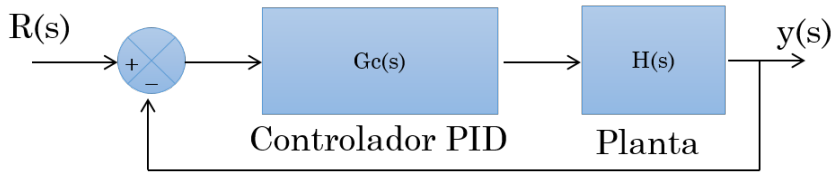

Figura 2.3: Diagrama a bloques del sistema de una planta con un control PID.

Capítulo 3

Desarrollo y resultados

Para poder aplicar el Método 1 de ZIEGLER – NICHOLS, la curva de respuesta de la función debe de tener forma de S. Con un punto de inflexión. Podemos obtener esta grafica en MatLab utilizando el siguiente código.

Listing 3.1: Creación de función de transferencia en Matlab.

```
num = [0 0 2.68];
den = [1 .916 2.68];
h=tf(num,den);
step(h)
```

Para encontrar los parámetros de tiempo de retardo L y constante de tiempo T, dibujamos una recta tangente en el punto de inflexión de la curva. El siguiente código encuentra el punto de inflexión en la curva de respuesta de H, a partir de esto se traza una línea tangente a este punto, y se utiliza la ecuación de la recta y la pendiente del punto de inflexión, para encontrar los parámetros L y T.

Listing 3.2: Código en Matlab para obtener T y L.

```
dt=0.05;
t=0:dt:8;
y=step(h,t)';
dy=diff(y)/dt;
[m,p]=max(dy);
yi=y(p);
ti=t(p);
L=ti-yi/m
T=(y(end)-yi)/m+ti-L
```

Como resultado de este programa se tiene que:

$$L = 0{,}2718$$

$$T = 0{,}8663$$

Y su gráfica es la siguiente.

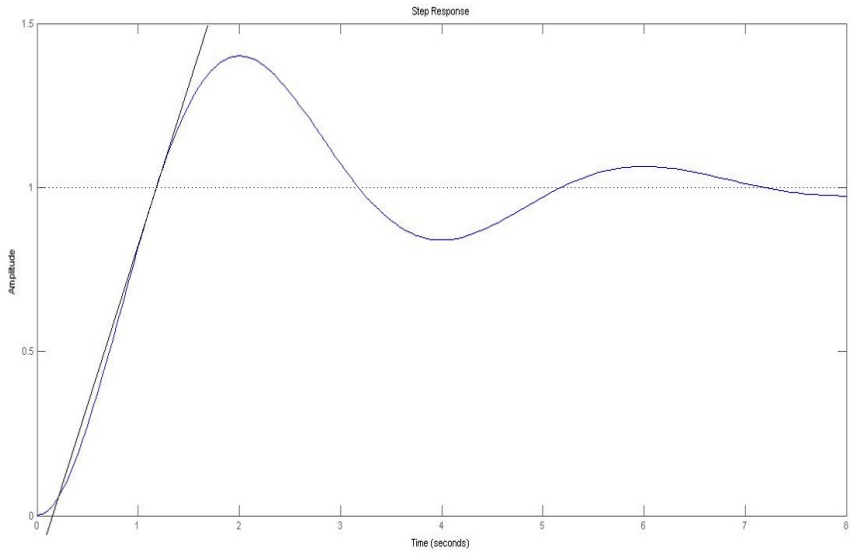

Figura 3.1: Gráfica que muestra los valores T y L.

Sustituyendo T y L en (2.1),(2.2),(2.3) se obtienen los siguientes valores.

$$K_p = 3{,}8255$$

$$T_i = 0{,}5435$$

$$T_d = 0{,}1359$$

Y sustituyendo K_p, T_i y T_d en 2.5 y 2.6 se tiene.

$$K_i = 7{,}0386$$

$$K_d = 0{,}5199$$

Lo que finalmente nos da la función del PID:

$$G_c(s) = \frac{0{,}52s^2 + 3{,}83s + 7{,}04}{s}$$

Para poder implementar el PID en su forma discreta se utiliza el siguiente pseudo código.

```
Inicio
Inicializar variables y constantes.
a1,a2,b1
e(n) = 0
e(n-1) = 0
u(n) = 0
u(n-1) = 0
Capturar el valor de la variable
c(nT) = ADC
calcular el error:
e(n) = r(nT)-c(nT)
evaluar la acción de control:
m(nT) = a1 e(n) + a2 e(n-1) + b1 u(n-1)
Emitir la acción de control:
DAC(m(n) + m(n-1))
Actualizar variables:
e(n-1) = e(n)
m(n-1) = m(n)
GOTO captura valor de la variable
```

Finalmente, aplicando el controlador PID discreto en nuestra planta y leyendo los resultados en Matlab se obtiene la siguiente gráfica:

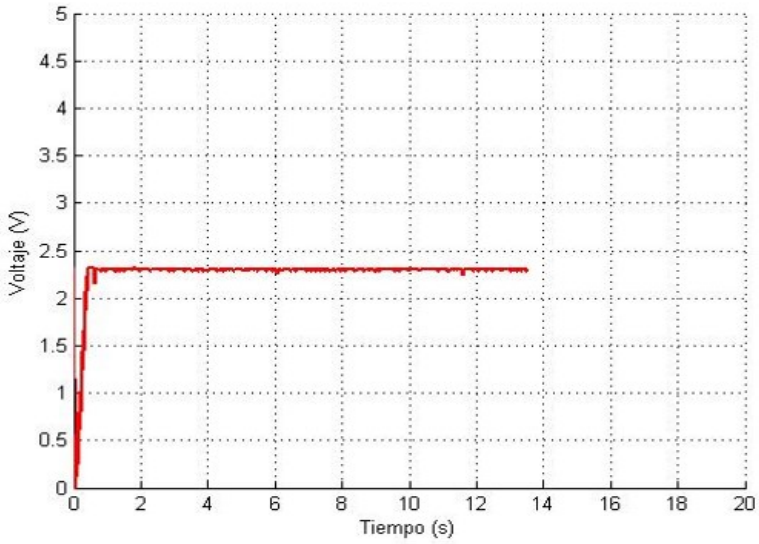

Figura 3.2: Gráfica del control PID implementado en la planta.

Capítulo 4

Conclusiones

El método 1 de Ziegler-Nichols es muy práctico al diseñar un control PID para aproximarse a los valores K_i,K_p y K_d que se necesitan para obtener las condiciones deseadas. Desarrollar el código para resolver el problema en MatLab ayuda bastante, ya que se requiere de varias pruebas experimentales con distintos valores de K_i,K_p y K_d. Para luego, observar la respuesta del sistema con estos valores, hasta encontrar algunos que cumplan con las condiciones deseadas en nuestra planta.

Bibliografía

[1] Ogata, Katsuhiko (1996). *Sistemas de Control en Tiempo discreto*. 2da. Edición. Editorial Prentice Hall Hispanoamericana S. A.

[2] Reyes Cortés, F.; Cid Monjaraz, J. & Vargas Soto, E. (2013). *Mecatrónica, Control y Automatización*. Alfaomega: México.

[3] Reyes Cortés, F. (2012). *MATLAB Aplicado a Robótica y Mecatrónica*. Alfaomega: México.

Índice de figuras

2.1. Logo de Matlab 4
2.2. Placa Arduino MEGA ADK. 4
2.3. Diagrama a bloques del sistema de una planta con un control PID. 5

3.1. Gráfica que muestra los valores T y L. 7
3.2. Gráfica del control PID implementado en la planta. 8