Instituto Tecnológico de Buenos Aires

22.85 - Sistemas de Control

Trabajo de Laboratorio N°4: Control de Carrito mediante controlador PID

Grupo 1

Máspero, Martina	57120
Mestanza, Joaquín Matías	58288
Nowik, Ariel Santiago	58309
Panaggio Venerandi, Guido Martin	56214
Parra, Rocío	57669
Regueira, Marcelo Daniel	58300

Profesor

Nasini, Víctor Gustavo

Presentado: 27/11/2019

Índice

1. PID: Introducción teórica

Los controladores PID (proporcional, integrador y derivativo) proveen un control de lazo empleando feedback que es utilizado en la industria del control. Un controlador PID calcula el error e(t) como la diferencia entre el setpoint (deseada) y una variable medida del proceso (la salida de la planta).

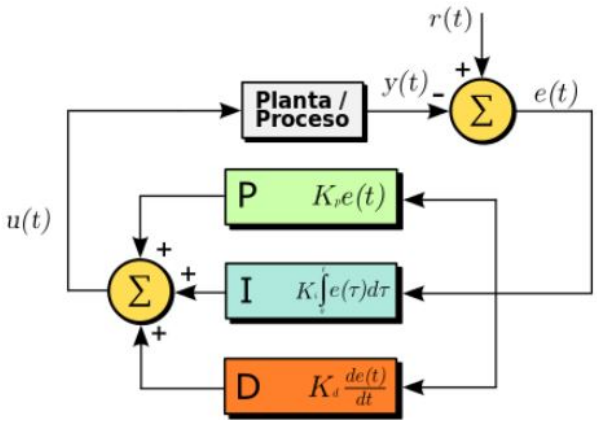

Figura 1: Controlador PID: Esquema

Como se puede observar en la figura 1 en los controladores PID se dispone de 3 constantes.

- K_p: constante que acompaña al error
- K_i: constante que acompaña a la integral del error
- K_d: constante que acompaña a la derivada del error

2. Obtención de datos con carrito

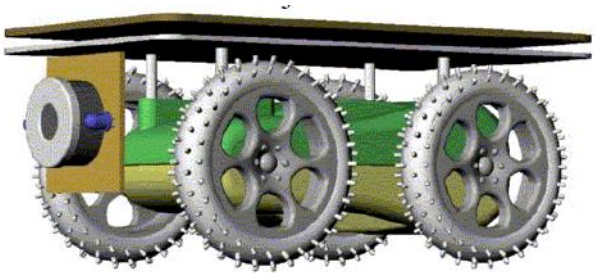

Figura 2: Carrito

Se dispone del carrito de la Figura 4, el cual posee un fototransistor y un par de leds en frente. Los leds generan radiación infraroja que se ve reflejada en la superficie que se encuentre enfrentada. Esta reflexión le da información al fototransistor sobre a qué distancia está el frente del carrito de la superficie.

El fototransistor se encuentra encerrado en un housing con una película oscura para reducir la perturbación de la luz visible, siendo esta película transparente a la radiación infrarroja.

El carrito posee un motor de corriente continua que le permite desplazarse en una sola dirección pero en ambos sentidos dependiendo de la polaridad de la tensión entregada al motor.

Se utilizó un microcontrolador tipo Arduino como ADC para la lectura del fototransistor, cuya respuesta de tensión en función de la distancia tiene forma de gaussiana. Es decir, estando junto a la pared, la señal de salida es de aproximadamente 0V, y comienza a aumentar hasta llegar a un máximo a medida que se aleja de la pared, y luego disminuye nuevamente hasta 0V si se lo sigue alejando.

3. Modelo del carrito

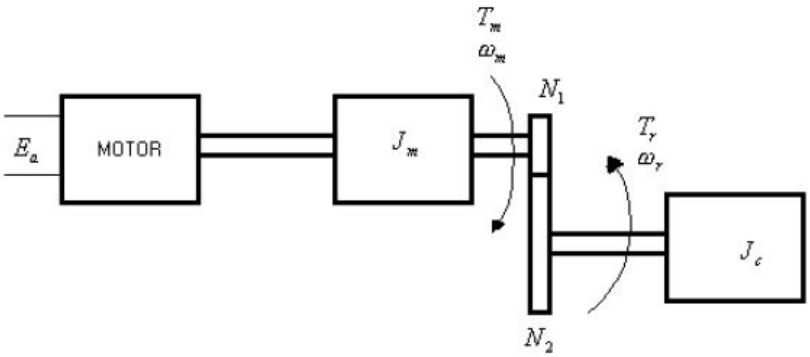

Figura 3: Representación del sistema físico

Se busca encontrar la transferencia $\frac{Y(S)}{E_a(S)}$, siendo $Y(S)$ el desplazamiento lineal del carrito sobre una superficie plana. Del desarrollo y despeje provisto por la cátedra, se obtiene la función:

$$G(S) = \frac{Y(S)}{E_a(S)} = \frac{N \cdot K_m \cdot r}{R_a \cdot J_{eq} \cdot \left(S + \frac{K_b \cdot N^2 \cdot K_m}{R_a \cdot J_{eq}}\right) \cdot S}$$

La cual caracteriza al sistema como segundo orden con un polo en el origen.

4. Control PID

El sistema realimentado con el control PID se muestra a continuación:

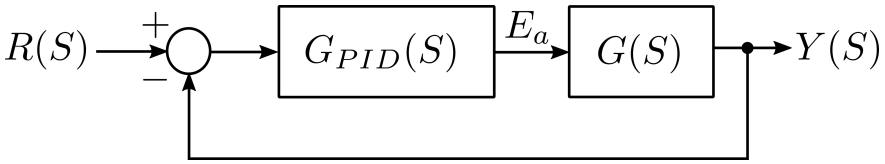

Figura 4: Sistema realimentado con compensador PID

Donde $G(S)$ es la transferencia del carrito $\frac{Y(S)}{E_a(S)}$ (siendo la posición respecto a la tensión de alimentación del motor), y la transferencia del controlador:

$$G_{PID}(S) = K \cdot \frac{(S + Z_1)(S + Z_2)}{S}$$

Para aplicar el control PID, se trabaja con el primer tramo inyectivo de la respuesta del fototransistor, es decir desde 0 hasta el máximo, buscando que el carrito se mantenga a una distancia intermedia entre esos dos puntos. El valor de tensión (que depende de la distancia) se muestrea con el ADC del Arduino, y es con el cual se realimenta al sistema.

Para el control del motor, se utiliza la Placa Stepper para Arduino (AR-L298SHIELD), cuya implementación se trata con la librería L298.h. La señal realimentada que va al controlador PID restada a $R(S)$, se convierte luego en un valor de duty para la señal de PWM para la Placa Stepper.

Las constantes K_p, K_d y K_i del controlador PID se ajustaron con la metodología descripta en la sección siguiente (dado que no se conocen los valores de la planta).

5. Método manual de ajuste

Se ajustaron las constantes mediante el método manual:

- Primero establecer $K_i = 0$ y $K_d = 0$.
- Incrementar K_p hasta que la salida oscile (es decir, el carrito presente un movimiento oscilatorio lo más armónico posible).
- Establecer K_p a aproximadamente la mitad del valor configurado previamente.
- Incrementar K_i hasta que el proceso se ajuste en el tiempo requerido (precaución: aumentar mucho K_i puede causar la inestabilidad del sistema, ya que los polos a lazo cerrado se mueven por el lugar de raíces y puede pasar al semiplano derecho).
- Finalmente, incrementar K_d si se necesita hasta que el lazo sea lo suficientemente rápido para alcanzar su referencia tras una variación brusca de la carga. Para el sistema bajo análisis se corresponde con, por ejemplo, tener al carrito fijo a una distancia, y de manera repentina colocar la mano delante del fototransistor, lo cual cambia abrutamente la medición de distancia.

6. Código

```
1
2 #include <L298N.h>
3 #include<PID_v1.h>
4
5 void setMotorSpeed(int s);
6
7 #define PH_EN 10
8 #define PH_INA 9
9 #define PH_INB 8
10 #define KP 1.4 // Valor de oscilatorio 3.2 con KD y KI en 0
11 #define KI 0.4
12 #define KD 0.2
13
14 double sp = 307; // Set point del sistema
15 double in = 0; // Medicion de la salida del sistmea - la realimentacion
16 double out=0; // Entrada a la planta
17
18 L298N canalA(PH_EN,PH_INA,PH_INB);
19 PID myPID(&in, &out, &sp,KP,KI,KD, REVERSE);
20
21 void setup() {
22  myPID.SetMode(AUTOMATIC);//turn the PID on
23  myPID.SetOutputLimits(-255,255);
24  myPID.SetSampleTime(10);//10ms de sampleo
25 }
26
27 void loop() {
28   in = analogRead(0);
29   myPID.Compute();
30   setMotorSpeed(out);
31   Serial.println(out);
32
33 }
34
35 void setMotorSpeed(int s){
36   if(s>128){
37     s=128;
38   }else if(s<-128){
39     s=-128;
40   }
41
42   // Seteo velocidad y direccion
43   if(s<0){
44     canalA.setSpeed(-s);
45     canalA.forward();
46   }else if(s<256){
47     canalA.setSpeed(s);
48     canalA.backward();
49   }
50 }
```