Práctica 5 - Control de servomotores
Interfaces y periféricos para robots

Roberto Cadena Vega

Objetivos

Controlar la posición de un motor de corriente directa con un sensor de realimentación (servomotor).

Conocimientos Previos

Controladores

En esta práctica utilizaremos primero el servomotor, como el de la figura 1, de la manera en que el fabricante esperaba que lo usaramos, despues lo modificaremos de tal manera en que podamos controlarlo completamente desde el Arduino.

En primer lugar debemos saber como funciona normalmente; para esto lo conectaremos de acuerdo a las instrucciones del fabricante, tomando cuidado de conectar el cable de señal al pin 13 de la tarjeta de desarrollo Arduino, para darle la señal de control por medio de este.

Ahora podemos subir el siguiente código a nuestro Arduino, con el que podremos modificar su posición por medio de comandos en el monitor serial del IDE de Arduino.

```
1  const int pin_servo = 13;
2  float pos = 90;
3  float t_on = 0;
4  float t_off = 0;
5
6  void setup() {
7    Serial.begin(9600);
8    pinMode(pin_servo, OUTPUT);
9  }
10 void loop() {
11   recibir_datos();
12   mandar_senal_servo();
13 }
14 void recibir_datos(){
15   if (Serial.available()) {
16     pos = Serial.readString().toFloat();
17     definir_senal_servo();
18   }
19 }
20
21
22
23 void definir_senal_servo(){
24   t_on = pos/150.0 + 0.9;
25   t_off = 3 - t_on;
26 }
27
28
29
30 void mandar_senal_servo(){
31   digitalWrite(pin_servo, HIGH);
32   delayMicroseconds(t_on*1000);
33   digitalWrite(pin_servo, LOW);
34   delayMicroseconds(t_off*1000);
35   delay(17);
36 }
```

En este código, la estructura general es que intentará recibir datos del puerto serial, y después mandará la señal al servomotor. Cuando recibe datos del puerto serial supondrá que es un número entre 0^o y 180^o y hará el cálculo de los retardos necesarios para dar la señal correcta al servomotor. Si analizamos este cálculo, podremos ver que cuando la posición es 0^o, el tiempo que la señal permanecerá encendida es $0.9ms$ y el tiempo que permanecerá apagada es $2.1ms + 17ms = 19.1ms$, es decir, una señal de $20ms$ en total; cuando la posición es 180^o, el tiempo de encendido es $2.1ms$, el tiempo de apagado es $0.9ms + 17ms = 17.9ms$, es decir, una señal de $20ms$ en total.

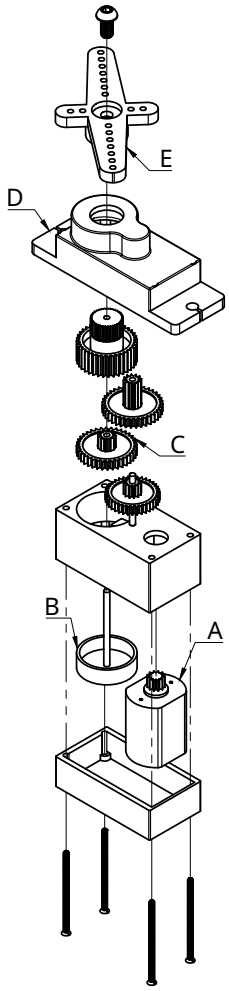


Figura 1: Explosion de un servomotor comercial

Cabe añadir que si esta señal dura $20ms$ en su totalidad, se tiene que enviar cada $20ms$, dependiendo de cada fabricante, es posible que el servomotor mantenga esta posición hasta que llegue una señal nueva, o bien, que si no llega una nueva señal despues de $20ms$, el controlador del servomotor suponga que el motor debe de ser apagado y el servomotor no conservará su posición al aplicarsele un par de torsión. Tambien es posible que el servomotor utilice un esquema diferente de excitación con tiempos diferentes a los establecidos en este código, en este caso será necesario modificar estos tiempos para adecuarse a la información del datasheet del servomotor.

En este punto deberás llenar la primer tabla de la hoja de anotaciones, con los tiempos que le toma a tu servomotor moverse de la posición 0^o a la posición 180^o o de la posición 180^o a la posición 0^o, para obtener un tiempo promedio de la operación realizada por el servomotor.

Trayectorias dependientes del tiempo

Si ahora queremos medir el error que va a tener este servomotor en una cierta trayectoria, necesitamos cargar el siguiente código el cual dará una señal senoidal al servomotor con una frecuencia variable:

```
1  const int pin_servo = 13;
2  float pos = 0;
3  float t_on = 0;
4  float t_off = 0;
5  float frec = 1;
6  float omega = TWO_PI*frec;
7  float t = 0;
8
9  void setup() {
10   Serial.begin(9600);
11   pinMode(pin_servo, OUTPUT);
12 }
13 void loop() {
14   recibir_datos();
15   trayectoria();
16   definir_senal_servo();
17   mandar_senal_servo();
18 }
19 void recibir_datos(){
20   if (Serial.available()) {
21     frec = Serial.readString().toFloat();
22     Serial.println(frec);
23     definir_frecuencia();
24   }
25 }
26
27
28 void definir_senal_servo(){
29   t_on = pos/150 + 0.9;
30   t_off = 20 - t_on;
31 }
32
33
34 void mandar_senal_servo(){
35   digitalWrite(pin_servo, HIGH);
36   delayMicroseconds(t_on*1000);
37   digitalWrite(pin_servo, LOW);
38   delayMicroseconds(t_off*1000);
39   delay(17);
40 }
41
42
43 void definir_frecuencia(){
44   omega = TWO_PI*frec;
45 }
46
47
48 void trayectoria(){
49   t = millis()/1000.0;
50   pos = (sin(omega*t) + 1)*90;
51 }
```

Este código empezará dando una señal senoidal al servomotor, haciendo que se mueva de 0^o hasta 180^o y de regreso, con una frecuencia de $1Hz$ para empezar, y con la posibilidad de enviar la frecuencia requerida por medio del monitor serial. Este código reutiliza elementos del pasado, pero antes de definir la posición a la que se debe mover el servomotor, primero calcula la posición por medio de una función senoidal que depende del tiempo que el Arduino ha permanecido encendido.

Si jugamos con este código, enviando valores por medio del puerto serial, podremos notar que mientras mayor sea la frecuencia de la señal senoidal, menor es el angulo que el servomotor esta recorriendo, no puede moverse lo suficientemente rápido antes de que tenga que regresar de nuevo.

En este punto deberás llenar la segunda tabla con el angulo que recorre el servomotor dependiendo de la frecuencia dada.

HACKING

¡Peligro! - A partir de este punto se harán modificaciones importantes al servomotor las cuales harán dificil el volver a resolver los puntos anteriores, por lo que es importante que hayas terminado las primeras dos tablas y hayas validado tu trabajo con el profesor, antes de empezar a trabajar en esta sección de la práctica.

Una vez que hemos caracterizado el desempeño de nuestro servomotor, podemos seguir adelante con el diseño de un controlador por medio de la tarjeta de desarrollo Arduino, para esto necesitaremos abrir la parte inferior del servomotor: debes tener cuidado con no abrir el compartimiento de la transmisión de engranes, o bien tratarlos con el suficiente cuidado. Como se ve en la figura 1, la transmisión de engranes identificada con la etiqueta (C), esta acoplada al motor DC (A) y al potenciometro (B) a través de una carcasa plástica la cual esta fija con la tapa inferior y la tapa superior (D) por medio de tornillos, estos deben ser removidos para acceder a las conexiones en la tapa inferior, pero a su vez pueden aflojar la conexion de la tapa superior; mas aún si la transmisión es desarmada, es facil volver a armarla, pero comúnmente estara cubierta de grasa de litio (sustancia blanca y viscosa), por lo que te mancharas las manos y probablemente la ropa si es que no tienes cuidado. Muchas veces ayuda comectar el eslabon de salida (E) a la tranmisión para evitar que la transmisión tenga juego y se desensamble en el proceso de abrir la tapa inferior del servomotor.

Una vez que tengas el servomotor abierto deberás retirar el controlador actual, desoldando o cortando los cables que entran a este y añadiendo longitud a los cables, los cuales deben ser, dos conductores del motor CD y tres conductores del potenciometro. Una vez que estos cables han sido alargados, se puede cerrar el servomotor de nuevo.

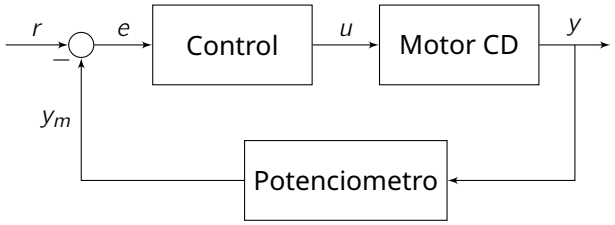
Los dos conductores del motor CD deben ser conectados al circuito de potencia adquirido (L293D o L298N), las señales de control para este circuito vendrán del Arduino y estaran marcadas como `m1` y `m2`. Los tres conductores del potenciometro serán conectados a $0V$, $5V$ y al pin marcado como `pot_pin` en el código; de tal manera que el Arduino detecte un voltaje en este pin entre $0V$ y $5V$ dependiendo de la posición del servomotor.

Esta parte de la práctica es tardada y delicada por lo que la recomendación general es que se haga en casa, con tiempo suficiente para corregir errores que pudieran suceder, previamente se deben tomar recomendaciones del profesor dependiendo del tipo de servomotor que se adquirió para esta práctica.

Una vez con nuestro circuito conectado, es momento de probar el primer código:

```
const int pot_pin = A0;
const int m1 = 10;
const int m2 = 11;

float kp = 1;
float sensor = 0;
float pos_des = 100;
float error = 0;
int senal = 0;

void setup(){
  Serial.begin(9600);
  pinMode(m1, OUTPUT);
  pinMode(m2, OUTPUT);
}

void loop(){
  recibir_datos();
  sensor = analogRead(pot_pin);
  Serial.println(sensor);
  calcular_error();
  calcular_senal();
  if(senal < 0) reversa();
  else avance();
  delay(2);
}

void recibir_datos(){
  if(Serial.available() > 0){
    pos_des = Serial.readString().toFloat();
  }
}

void calcular_error(){
  error = pos_des - sensor;
}

void calcular_senal(){
  senal = int(max(min(kp*error, 255), -255));
}

void reversa(){
  analogWrite(m1, 0);
  analogWrite(m2, abs(senal));
}

void avance(){
  analogWrite(m2, 0);
  analogWrite(m1, senal);
}
```

Este código intentará recibir la referencia de la posición deseada del puerto serial, leerá la posición del potenciometro, mandará este dato por el puerto serial, calculará el error entre la posición deseada y la posición real, calculará la señal necesaria para mover el motor a la posición deseada, y por ultimo, ira hacia adelante o hacia atras, dependiendo de si la señal necesaria es positiva o negativa. Si dibujamos un diagrama de bloques de la secuencia de información se verá asi:



En el código, el control esta determinado por `kp`, de tal manera que esta cantidad determinará el comportamiento del sistema.

Si jugamos con el sistema asi, podemos enviar la posicion de refencia por medio del monitor serial, tambien podemos modificar la ganancia `kp`, de tal manera en que el sistema sea mas rapido o lento, estable o criticamente estable, subamortiguado o sobreamortiguado.

En este punto deberás llenar la tercer tabla, sintonizando el controlador k_p para obtener el menor tiempo posible en mover el motor de la posición 0^o a 180^o o biscversa.

CONTROL PD

Hasta el momento hemos utilizado un control proporcional simple, pero podemos implementar un control un poco mas complejo con pocos calculos; el control PD tan solo requiere que calculemos el error y la velocidad del error que es equivalente a la velocidad del motor.

```
const int pot_pin = A0;
const int m1 = 10;
const int m2 = 11;

float kp = 10;
float kd = 2.5;
float sensor = 0;
float pos_des = 100;
float error = 0;
float error_anterior = 0;
int senal = 0;

void setup(){
  Serial.begin(9600);
  pinMode(m1, OUTPUT);
  pinMode(m2, OUTPUT);
}

void loop(){
  recibir_datos();
  sensor = analogRead(pot_pin);
  Serial.println(sensor);
  calcular_error();
  calcular_senal();
  if(senal < 0) reversa();
  else avance();
  delay(2);
}

void recibir_datos(){
  if(Serial.available() > 0){
    pos_des = Serial.readString().toFloat();
  }
}

void calcular_error(){
  error_anterior = error;
  error = pos_des - sensor;
}

void calcular_senal(){
  float u = kp*error + kd*(error - error_anterior);
  senal = int(max(min(u, 255), -255));
}

void reversa(){
  analogWrite(m1, 0);
  analogWrite(m2, abs(senal));
}

void avance(){
  analogWrite(m2, 0);
  analogWrite(m1, senal);
}
```

En este punto deberás llenar la tabla cuatro, la cual te ayudará a sintonizar el controlador PD utilizando el mismo procedimiento de la anterior.

Equipo

El siguiente equipo será proporcionado por el laboratorio, siempre y cuando lleguen en los primeros 15 minutos de la práctica, y hagan el vale conteniendo el siguiente equipo (exceptuando las pinzas).

- Fuente de alimentación
- Osciloscopio
- Cables BNC - caimán
- Cables banana - caimán
- Cables de alimentación
- Multímetro
- Pinzas
- Transportador

Materiales

- Protoboard
- Cables
- Motor CD
- LED
- L293D, L298N, etc.

Desarrollo

Diseña los circuitos requeridos y toma las mediciones necesarias para llenar la hoja de anotaciones y realiza el análisis cuantitativo del error de cada controlador.

Conclusiones

El alumno deberá describir sus conclusiones al final de su reporte de práctica.

HOJA DE ANOTACIONES

1. Llena la tabla de abajo, utilizando el primer código para validar el tiempo necesario para que tu servomotor se mueva de la posición 0^o a la posición 180^o o de la posición 180^o a la posición 0^o.

Intento	1	2	3	4	5	6	7	8	9	10
t										

2. Llena la tabla de abajo, utilizando el segundo código para validar el angulo recorrido por el servomotor en cada una de las frecuencias descritas abajo.

f (Hz)	0.25	0.5	0.75	1.00	1.25	1.5	1.75	2.0	4.00	8.00
$\Delta\theta$										

3. Diseña el circuito necesario para conectar tu servomotor al Arduino por medio del circuito de potencia que hayas adquirido.

4. Llena la tabla de abajo, utilizando el tercer código para sintonizar el controlador, minimizando el tiempo necesario para que tu servomotor se mueva de la posición 0^o a la posición 180^o o de la posición 180^o a la posición 0^o.

Intento	1	2	3	4	5
kp					
t_1					
t_2					
t_3					
t_4					

5. Llena la tabla de abajo, utilizando el tercer código para sintonizar el controlador, minimizando el tiempo necesario para que tu servomotor se mueva de la posición 0^o a la posición 180^o o de la posición 180^o a la posición 0^o.

Intento	1	2	3	4	5
kp					
kd					
t_1					
t_2					
t_3					
t_4					

Integrantes del equipo: Revisó:

_____

_____ _____