Práctica 4 - Control de motores a pasos
Interfaces y periféricos para robots

Roberto Cadena Vega

Objetivos

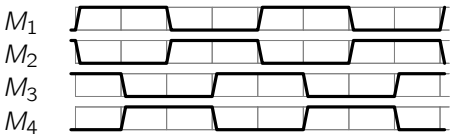
Controlar un motor a pasos por medio de instrucciones de código G, para el posicionamiento de un mecanismo de un grado de libertad.

Conocimientos Previos

Motores a pasos

En esta práctica controlaremos un motor a pasos por medio de instrucciones proporcionadas a la tarjeta de desarrollo Arduino por la computadora, primero empecemos con el circuito; en primer lugar, recuerda que debes tener el datasheet de tu CI de control, sin embargo, propongo un circuito para el L293D en la figura 1.

Lo siguiente que tenemos que hacer es describir las señales que vamos a darle al motor a pasos, a continuación analizamos la siguiente gráfica:



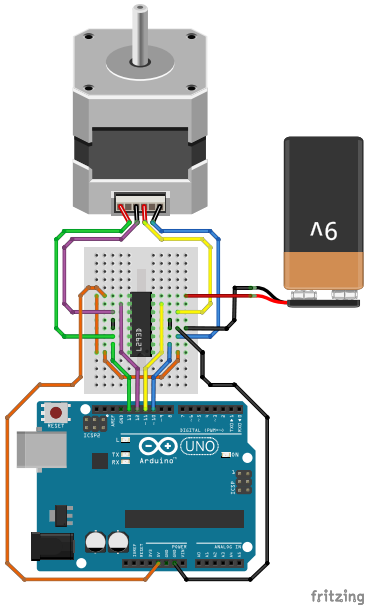


Figura 1: Diagrama representativo de Arduino controlando un motor a pasos bipolar

con lo que, en primera instancia, desarrollamos el siguiente código:

```
 1 int subpaso = 0;
 2 int pos = 0;
 3 const int m1 = 10;
 4 const int m2 = 11;
 5 const int m3 = 12;
 6 const int m4 = 13;
 7
 8 void setup() {
 9   Serial.begin(9600);
10   pinMode(m1, OUTPUT);
11   pinMode(m2, OUTPUT);
12   pinMode(m3, OUTPUT);
13   pinMode(m4, OUTPUT);}
14
15 void loop() {
16   if (Serial.available())
17     if(Serial.readString() == "p\n")
18       avanzar_paso();}
19
20 void avanzar_paso(){
21   if (subpaso == 0){
22     digitalWrite(m1, HIGH);
23     digitalWrite(m2, LOW);
24     digitalWrite(m3, HIGH);
25     digitalWrite(m4, LOW);
26     delay(10);}
27   if (subpaso == 1){
28     digitalWrite(m1, HIGH);
29     digitalWrite(m2, LOW);
30     digitalWrite(m3, LOW);
31     digitalWrite(m4, HIGH);
32     delay(10);}
33   if (subpaso == 2){
34     digitalWrite(m1, LOW);
35     digitalWrite(m2, HIGH);
36     digitalWrite(m3, LOW);
37     digitalWrite(m4, HIGH);
38     delay(10);}
39   if (subpaso == 3){
40     digitalWrite(m1, LOW);
41     digitalWrite(m2, HIGH);
42     digitalWrite(m3, HIGH);
43     digitalWrite(m4, LOW);
44     delay(10);}
45   subpaso++;
46   subpaso = subpaso%4;}
```

En las lineas 1 a 6, se definen como constantes los pines en que se conectan los puertos de control del motor a pasos, y se define como variable el paso en el que se encuentra el motor, asi como la posición. Adentro de la función `setup` se manda a llamar a la función `avanzar_paso` cada que se recibe el caracter `p`. En la función `avanzar_paso` se prenden o apagan los pines necesarios para mandar la señal descrita anteriormente, de tal manera que en el subpaso `0` mandamos los estados de las señales en el primer instante de tiempo descrito en la gráfica, en el subpaso `1` las señales en el segundo instante de tiempo y asi sucesivamente; despues de esto se incrementa el contador para `subpaso` y se calcula el modulo `4`, de tal manera que subpaso siempre tenga un valor entre `0` y `3`.

Si este programa es cargado en la tarjeta de desarrollo Arduino y se le manda un mensaje `p` el motor a pasos dará un paso. Esto mismo lo podemos lograr con otra estructura de programación que se llama `switch`:

```
int subpaso = 0;
int pos = 0;
int retraso = 10;
String inst;
const int m1 = 10;
const int m2 = 11;
const int m3 = 12;
const int m4 = 13;

void setup() {
  Serial.begin(9600);
  pinMode(m1, OUTPUT);
  pinMode(m2, OUTPUT);
  pinMode(m3, OUTPUT);
  pinMode(m4, OUTPUT);}

void loop() {
  if (Serial.available()) {
    inst = Serial.readString();
    if(inst == "p\n"){
      avanzar_paso();
      Serial.println(subpaso);}}}

void avanzar_paso(){
  switch(subpaso){
    case 0:
      digitalWrite(m1, HIGH);
      digitalWrite(m2, LOW);
      digitalWrite(m3, HIGH);
      digitalWrite(m4, LOW);
      delay(retraso);
      break;
    case 1:
      digitalWrite(m1, HIGH);
      digitalWrite(m2, LOW);
      digitalWrite(m3, LOW);
      digitalWrite(m4, HIGH);
      delay(retraso);
      break;
    case 2:
      digitalWrite(m1, LOW);
      digitalWrite(m2, HIGH);
      digitalWrite(m3, LOW);
      digitalWrite(m4, HIGH);
      delay(retraso);
      break;
    case 3:
      digitalWrite(m1, LOW);
      digitalWrite(m2, HIGH);
      digitalWrite(m3, HIGH);
      digitalWrite(m4, LOW);
      delay(retraso);
      break;}
  subpaso++;
  subpaso = subpaso%4;}
```

Sin embargo este código solo puede mover el motor a pasos en un sentido, para hacer que el motor se mueva en el sentido contrario tenemos que mandar la señal anterior, esencialmente disminuir el subpaso del programa, por lo que podemos crear una función que regrese el motor al subpaso anterior y modificar la función principal del programa de tal manera que le mandemos la instrucción `d` para mover el motor hacia adelante o `r` para mover el motor hacia atras:

```
void loop() {
  if (Serial.available()) {
    inst = Serial.readString();
    if(inst == "d\n")
      avanzar_paso();
    if(inst == "r\n")
      retroceder_paso();
    Serial.println(subpaso);}}
```

y agregamos la función para reversa:

```
void retroceder_paso(){
  switch(subpaso){
    case 2:
      digitalWrite(m1, HIGH);
      digitalWrite(m2, LOW);
      digitalWrite(m3, HIGH);
      digitalWrite(m4, LOW);
      delay(retraso);
      break;
    case 3:
      digitalWrite(m1, HIGH);
      digitalWrite(m2, LOW);
      digitalWrite(m3, LOW);
      digitalWrite(m4, HIGH);
      delay(retraso);
      break;
    case 0:
      digitalWrite(m1, LOW);
      digitalWrite(m2, HIGH);
      digitalWrite(m3, LOW);
      digitalWrite(m4, HIGH);
      delay(retraso);
      break;
    case 1:
      digitalWrite(m1, LOW);
      digitalWrite(m2, HIGH);
      digitalWrite(m3, HIGH);
      digitalWrite(m4, LOW);
      delay(retraso);
      break;}
  subpaso--;
  subpaso = (subpaso + 4)%4;}
```

Pero no es suficiente con hacer que el motor se mueva un paso cada que se le dice, sería mejor poder decirle a que posición se tiene que mover. Para esto utilizaremos la variable `pos` para llevar un registro de en donde esta, tambien modificaremos la función principal para que el mensaje que le llegue de la computadora, lo convierta a entero y lo guarde en la variable `inst`:

```
int subpaso = 0;
int pos = 0;
int retraso = 5;
int inst;
const int m1 = 10;
const int m2 = 11;
const int m3 = 12;
const int m4 = 13;

void setup() {
  Serial.begin(9600);
  pinMode(m1, OUTPUT);
  pinMode(m2, OUTPUT);
  pinMode(m3, OUTPUT);
  pinMode(m4, OUTPUT);}

void loop() {
  if (Serial.available()) {
    inst = Serial.readString().toInt();
    Serial.println(inst);}
  if (inst < pos)
    retroceder_paso();
  if (inst > pos)
    avanzar_paso();}

void avanzar_paso(){
  switch(subpaso){
    case 0:
      digitalWrite(m1, HIGH);
  subpaso++;
  subpaso = subpaso%4;
  pos++;}
```

y agregaremos un paso a la posición del motor cada que avanzemos el motor o disminuiremos la posición cuando retrocedamos:

```
  subpaso--;
  subpaso = (subpaso + 4)%4;
  pos--;}
```

Si hacemos pruebas con el código de esta manera, podremos ver que el motor funciona o no... esto dependerá de las necesidades de energía de tu motor, si tu motor no funciona puedes intentar cualquiera de las dos siguientes:

- Incrementar el retraso
- Incrementar el voltaje de suministro al motor

Cualquiera de las dos debe hacer que tu motor funcione, sin embargo, si tu motor ya esta en el voltaje nominal no te recomiendo subir aun mas el voltaje de alimentación, ya que pudieras arruinar sus embobinados.

Aun podemos hacer cambios a este código, por ejemplo, podemos hacer que el motor trabaje a la velocidad que deseemos; para esto implementare un subconjunto de instrucciones de código G, de tal manera que cuando envie la instrucción `G00 100`, el motor se moverá a la posición 100 a la velocidad máxima, y cuando envíe la instrucción `G01 100 050`, se moverá a la posición 100 con una velocidad de 50 pasos por segundo. El código para esto queda:

```
int subpaso = 0;
int pos = 0;
int vel_max = 150;
int velocidad;
int retraso;
int pos_des;
String inst;
const int m1 = 10;
const int m2 = 11;
const int m3 = 12;
const int m4 = 13;

void setup() {
  Serial.begin(9600);
  pinMode(m1, OUTPUT);
  pinMode(m2, OUTPUT);
  pinMode(m3, OUTPUT);
  pinMode(m4, OUTPUT);
  conf_ret(vel_max);}

void loop() {
  if (Serial.available()) {
    inst = Serial.readString();
    if (inst.substring(0,3) == "G01"){
      pos_des = inst.substring(4,7).toInt();
      velocidad = inst.substring(8,11).toInt();
      conf_ret(velocidad);}
    if (inst.substring(0,3) == "G00"){
      pos_des = inst.substring(4,7).toInt();
      conf_ret(vel_max);}
    Serial.println(inst);}
  if (pos_des < pos)
    retroceder_paso();
  if (pos_des > pos)
    avanzar_paso();}

void conf_ret(int vel){
  retraso = 1000/vel;}

void avanzar_paso(){
  switch(subpaso){
    case 0:
      digitalWrite(m1, HIGH);
      digitalWrite(m2, LOW);
      digitalWrite(m3, HIGH);
      digitalWrite(m4, LOW);
      delay(retraso);
      break;
    case 1:
      digitalWrite(m1, HIGH);
      digitalWrite(m2, LOW);
      digitalWrite(m3, LOW);
      digitalWrite(m4, HIGH);
      delay(retraso);
      break;
    case 2:
      digitalWrite(m1, LOW);
      digitalWrite(m2, HIGH);
      digitalWrite(m3, LOW);
      digitalWrite(m4, HIGH);
      delay(retraso);
      break;
    case 3:
      digitalWrite(m1, LOW);
      digitalWrite(m2, HIGH);
      digitalWrite(m3, HIGH);
      digitalWrite(m4, LOW);
      delay(retraso);
      break;}
  subpaso++;
  subpaso = subpaso%4;
  pos++;}

void retroceder_paso(){
  switch(subpaso){
    case 2:
      digitalWrite(m1, HIGH);
      digitalWrite(m2, LOW);
      digitalWrite(m3, HIGH);
      digitalWrite(m4, LOW);
      delay(retraso);
      break;
    case 3:
      digitalWrite(m1, HIGH);
      digitalWrite(m2, LOW);
      digitalWrite(m3, LOW);
      digitalWrite(m4, HIGH);
      delay(retraso);
      break;
    case 0:
      digitalWrite(m1, LOW);
      digitalWrite(m2, HIGH);
      digitalWrite(m3, LOW);
      digitalWrite(m4, HIGH);
      delay(retraso);
      break;
    case 1:
      digitalWrite(m1, LOW);
      digitalWrite(m2, HIGH);
      digitalWrite(m3, HIGH);
      digitalWrite(m4, LOW);
      delay(retraso);
      break;}
  subpaso--;
  subpaso = (subpaso + 4)%4;
  pos--;}
```

En este caso las funciones `avanzar_paso` y `retroceder_paso` no cambian, agrego una función `conf_ret` que va a calcular el retraso necesario para una cierta velocidad, en la función principal se revisa que instrucción se mandó y configura la velocidad a la que se debe de mover el motor, así como la posición, y por ultimo se mueve un paso (hacia adelante o hacia atras), dependiendo de si ya se llego a la posición deseada o no. Si no quiero que la velocidad que trate de lograr el motor sea mayor a la velocidad maxima establecida, puedo cambiar la función principal por esta:

```
void loop() {
  if (Serial.available()) {
    inst = Serial.readString();
    if (inst.substring(0,3) == "G01"){
      pos_des = inst.substring(4,7).toInt();
      velocidad = inst.substring(8,11).toInt();
      if (velocidad > vel_max) conf_ret(vel_max);
      else conf_ret(velocidad);}
    if (inst.substring(0,3) == "G00"){
      pos_des = inst.substring(4,7).toInt();
      conf_ret(vel_max);}
    Serial.println(inst);}
  if (pos_des < pos)
    retroceder_paso();
  if (pos_des > pos)
    avanzar_paso();}
```

Sin embargo no vamos a parar aqui, lo que realmente queremos es cambiar la velocidad poco a poco, hasta llegar a una velocidad adecuada, sin que el motor necesite demasiada energía. Pensando en lo que pasó cuando intentamos correr el motor desde el reposo hasta una velocidad rápida, le exigimos una aceleración instantanea demasiado grande, por lo que si limitamos la aceleración a la que sometemos el motor, podremos conseguir velocidades mucho mas grandes que con el metodo anterior.

Para conseguir esto, primero definiré una función para configurar el retraso dandole la velocidad y guardando este valor como la velocidad actual:

```
void conf_vel(int vel){
  velocidad = vel;
  retraso = 1000/velocidad;}
```

Segundo, ahora no solo estamos variando la posición del motor, tambien estamos cambiando la velocidad del motor, por lo que agregamos una condición similar a la hora de mover el motor, pero considerando que ahora queremos mover la posición del motor con una velocidad diferente cada vez, esta diferencia es lo que consideramos la aceleración, por lo que esta parte del código queda:

```
  if (pos_des < pos) {
    if (velocidad < vel_des)
      conf_vel(velocidad+aceleracion);
    retroceder_paso();}
  if (pos_des > pos) {
    if (velocidad < vel_des)
      conf_vel(velocidad+aceleracion);
    avanzar_paso();}}
```

Por lo que al final el código queda:

```
int subpaso = 0;
int pos = 0;
int vel_max = 500;
int vel_min = 100;
int velocidad;
int vel_des;
int aceleracion = 1;
int retraso;
int pos_des;
String inst;

const int m1 = 10;
const int m2 = 11;
const int m3 = 12;
const int m4 = 13;

void setup() {
  Serial.begin(9600);
  pinMode(m1, OUTPUT);
  pinMode(m2, OUTPUT);
  pinMode(m3, OUTPUT);
  pinMode(m4, OUTPUT);
  conf_ret(vel_min);}

void loop() {
  if (Serial.available()) {
    inst = Serial.readString();
    if (inst.substring(0,3) == "G01"){
      pos_des = inst.substring(4,7).toInt();
      vel_des = inst.substring(8,11).toInt();
      if (vel_des > vel_max) vel_des = vel_max;}
    if (inst.substring(0,3) == "G00"){
      pos_des = inst.substring(4,7).toInt();
      vel_des = vel_max;}
    conf_vel(vel_min);
    Serial.println(inst);}
  if (pos_des < pos) {
    if (velocidad < vel_des)
      conf_vel(velocidad+aceleracion);
    retroceder_paso();}
  if (pos_des > pos) {
    if (velocidad < vel_des)
      conf_vel(velocidad+aceleracion);
    avanzar_paso();}}

void conf_vel(int vel){
  velocidad = vel;
  retraso = 1000/velocidad;}

void avanzar_paso(){
  switch(subpaso){
    case 0:
      digitalWrite(m1, HIGH);
      digitalWrite(m2, LOW);
      digitalWrite(m3, HIGH);
      digitalWrite(m4, LOW);
      delay(retraso);
      break;
    case 1:
      digitalWrite(m1, HIGH);
      digitalWrite(m2, LOW);
      digitalWrite(m3, LOW);
      digitalWrite(m4, HIGH);
      delay(retraso);
      break;
    case 2:
      digitalWrite(m1, LOW);
      digitalWrite(m2, HIGH);
      digitalWrite(m3, LOW);
      digitalWrite(m4, HIGH);
      delay(retraso);
      break;
    case 3:
      digitalWrite(m1, LOW);
      digitalWrite(m2, HIGH);
      digitalWrite(m3, HIGH);
      digitalWrite(m4, LOW);
      delay(retraso);
      break;}
  subpaso++;
  subpaso = subpaso%4;
  pos++;}

void retroceder_paso(){
  switch(subpaso){
    case 2:
      digitalWrite(m1, HIGH);
      digitalWrite(m2, LOW);
      digitalWrite(m3, HIGH);
      digitalWrite(m4, LOW);
      delay(retraso);
      break;
    case 3:
      digitalWrite(m1, HIGH);
      digitalWrite(m2, LOW);
      digitalWrite(m3, LOW);
      digitalWrite(m4, HIGH);
      delay(retraso);
      break;
    case 0:
      digitalWrite(m1, LOW);
      digitalWrite(m2, HIGH);
      digitalWrite(m3, LOW);
      digitalWrite(m4, HIGH);
      delay(retraso);
      break;
    case 1:
      digitalWrite(m1, LOW);
      digitalWrite(m2, HIGH);
      digitalWrite(m3, HIGH);
      digitalWrite(m4, LOW);
      delay(retraso);
      break;}
  subpaso--;
  subpaso = (subpaso + 4)%4;
  pos--;}
```

Tomemos en cuenta que este código tiene ahora muchos parametros para configurar, en mi caso pude subir la velocidad maxima hasta 500 pasos por segundo, con una velocidad minima de 100 pasos por minuto y una aceleración de 1 paso por segundo al cuadrado.

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
- Cables de alimentación
- Multímetro
- Pinzas

Materiales

- Protoboard
- Cables
- Motor a pasos
- 4 LEDs
- L293D, L298N, etc.

Desarrollo

Sintoniza las variables en el código de control de motores a pasos para minimizar el tiempo requerido para moverse desde la posición `0` hasta la posición `999` o bien de la posición `999` hasta la posición `0` y responde las preguntas de la hoja de anotaciones.

Utiliza 4 mediciones de tiempo para cada intento de sintonización (2 de ida y 2 de regreso) y 10 mediciones de tiempo para el intento final. Registra todos estos tiempos en la hoja de anotaciones.

Conclusiones

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

HOJA DE ANOTACIONES

1. ¿Cual es el retraso mínimo necesario para que tu motor a pasos funcione confiablemente cuando se trabaja a velocidad constante?

2. Llena la tabla de abajo, con los tiempos medidos en el proceso de sintonización de parametros:

Intento	1	2	3	4	5
`vel_max`					
`vel_min`					
`aceleracion`					
t_1					
t_2					
t_3					
t_4					
t_5					
t_6					
t_7					
t_8					
t_9					
t_{10}					

Intento	6	7	8	9	10
`vel_max`					
`vel_min`					
`aceleracion`					
t_1					
t_2					
t_3					
t_4					
t_5					
t_6					
t_7					
t_8					
t_9					
t_{10}					

Integrantes del equipo: Revisó:

_____

_____ _____