Práctica 3 - Control de motores CD
Interfaces y periféricos para robots

Roberto Cadena Vega

Objetivos

Controlar la velocidad de un motor de corriente directa de acuerdo a especificaciones enviadas por el puerto serial.

Conocimientos Previos

PWM

Empezaremos esta práctica analizando el concepto de una señal digital que nos permite controlar una variable que parece analógica en cualquier otro caso, el voltaje de salida del μC.

Lo primero que haremos será modificar el código de ejemplo Fading que hemos utilizado en practicas anteriores, ya que sabemos que este genera una señal PWM para modificar la luminosidad del LED conectado a su salida:

```
1  int motor_pin = 10;
2  int motor_vel = 0;
3  int motor_accel = 5;
4
5  void setup() {
6    pinMode(motor_pin, OUTPUT);
7  }
8
9  void loop() {
10   analogWrite(motor_pin, motor_vel);
11   motor_vel = motor_vel + motor_accel;
12
13   if (motor_vel <= 0 || motor_vel >= 255) {
14     motor_accel = -motor_accel;
15   }
16   delay(30);
17 }
```

Si conectamos nuestro osciloscopio a la salida del Arduino, podremos ver una señal como la de la figura 1, de tal manera que la luminosidad del LED corresponde al porcentaje del tiempo que la señal permanece en un estado de 1 lógico.

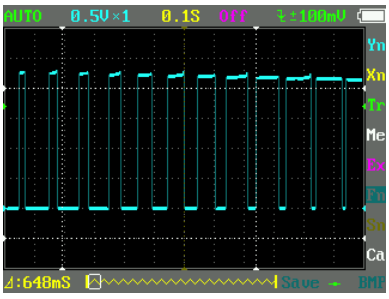

Figura 1: PWM variante con el tiempo

Si ahora, estas pensando que tienes que conectar tu motor al Arduino y hacer que gire, espera... lo único que cnoseguiras será quemar tu Arduino y probablemente el puerto USB de la computadora. Tenemos que asegurarnos que la corriente que demanda el motor no pase por el Arduino directamente, para esto podemos utilizar un circuito como el ULN2003A, que es un conjunto de arreglos Darlington, diseñado para dirigir corrientes de hasta $0.5A$, o bien circuitos como el L293D o el L298N que estan diseñados explicitamente para controlar motores de corriente directa.

En esta práctica se dará como ejemplo un circuito con el ULN2003A, sin embargo tu deberás utilizar el circuito que hayas adquirido; para esto es de primera importancia que tengas a la mano el datasheet del circuito que hayas conseguido; el datasheet es un documento que describe la correcta utilización del un circuito electrónico, por lo general estos tienen caratulas como la de la figura 2, el cual corresponde al ULN2003A.

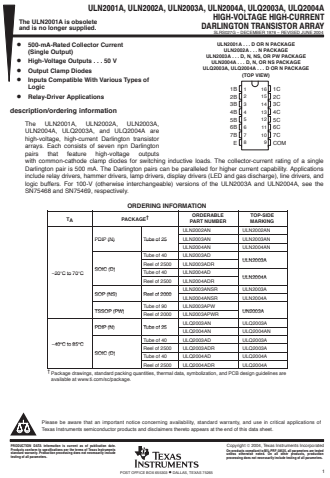
ULN2001A, ULN2002A, ULN2003A, ULN2004A, ULQ2003A, ULQ2004A
HIGH-VOLTAGE HIGH-CURRENT DARLINGTON TRANSISTOR ARRAY

Figura 2: Datasheet del circuito integrado ULN2003A

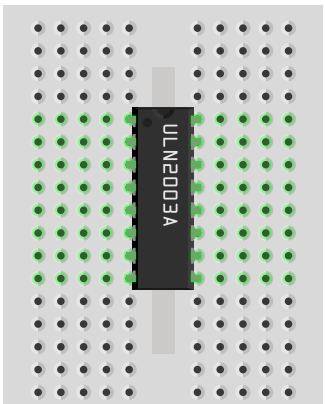

Figura 3: ULN2003A

En la figura 2 podemos ver una representación esquemática del circuito interno del CI, ten en cuenta que cuando lo coloquemos en la protoboard, el CI se verá como en la figura 3, con una muesca en la parte superior, que corresponde a la muesca en la parte superior del diagrama esquemático del datasheet.

Si ahora seguimos las especificaciones de conexion encontradas en el datasheet, podemos hacer nuestro circuito para controlar el motor por medio del Arduino. Nota que en la figura 4, existen dos cables que no tienen conexion, estos se tienen que conectar a una fuente de alimentación, de tal manera que esta sea la que alimente al motor y no la tajeta de desarrollo Arduino. Nota tambien que la figura tiene marcada una conexion de tierra desde el Arduino a la conexion de tierra del protoboard, esta conexion es necesaria siempre que se tengan multiples fuentes de alimentación, en este caso el Arduino y la fuente de alimentación del laboratorio.

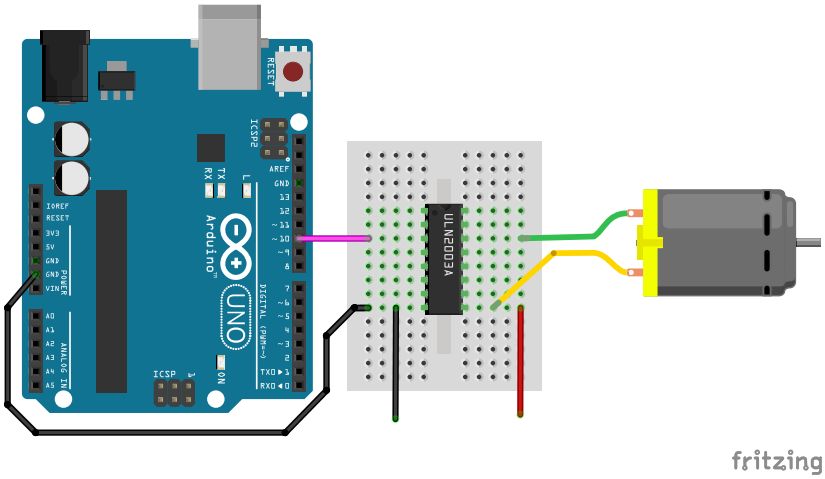

Figura 4: Conexion Arduino - ULN2003A - Motor CD

Una vez conectado nuestro circuito podemos medir con el osciloscopio la misma señal que en la figura 1 si lo conectamos a la entrada del ULN2003A, pero si lo conectamos a la salida (el punto acoplado al motor CD), veremos una señal diferente, una como la de la figura 5.

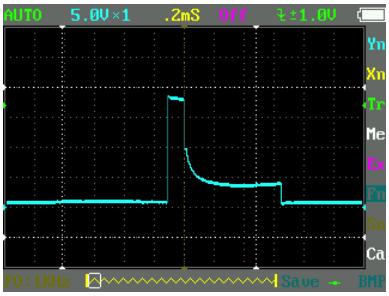

Figura 5: Señal de alimentación cuadrada acoplada a motor CD

En lugar de tener una señal cuadrada, podemos ver como se desestabiliza la señal hasta llegar a un valor mínimo, dependiendo de la construcción de tu motor de CD, pudieras ver un efecto similar o igual al de la figura 5, de cualquier manera, este efecto tiene que ver con la inductancia y reluctancia del motor, convirtiendo la señal eléctrica en una señal magnética y de regreso al sistema, otra de las razones por las que no es una buena idea conectar un motor directamente a un CI delicado como un μC.

Para este punto queremos controlar la velocidad del motor por medio de instrucciones en la computadora, por lo que pensamos acerca de las variables involucradas en esta señal; la frecuencia de la señal de control y el porcentaje de duty cycle, empezaremos modificando la frecuencia de la señal.

Desafortunadamente no hay una manera sencilla de modificar la frecuencia de la señal de control utilizando el código anterior, por lo que utilizaremos el siguiente:

```
1  int motor_pin = 10;
2  float freq = 1000;
3  float T = 1000000/freq;
4  float dc = 0.3;
5  float t_on = T*dc;
6  float t_off = T*(1-dc);
7
8  void setup()
9  {
10   pinMode(motor_pin, OUTPUT);
11 }
12
13 void loop()
14 {
15   digitalWrite(motor_pin, HIGH);
16   delayMicroseconds(t_on);
17   digitalWrite(motor_pin, LOW);
18   delayMicroseconds(t_off);
19 }
```

En las lineas 1-6, tenemos variables que configuran nuestra señal de control, `freq = 1000` nos da la frecuencia de $1kHz$, con lo que se calcula el periodo de la señal, el duty cycle es 30% y con esto se calcula el tiempo de encendido y el tiempo de apagado; estos tiempos se utilizan en las lineas 16 y 18 para poner un retraso con estos tiempos.

Si se pone a trabajar este código en el Arduino, veremos trabajar al motor con una cierta velocidad y oiremos un ruido muy especifico, exactamente un tono de $1kHz$, podemos modificar este código para replicar diferentes tonos, de acuerdo a la frecuencia que pongamos, o lo que es incluso mejor, si ponemos una frecuencia mayor a $20kHz$ dejaremos de escuchar este tono, la audición humana trabaja entre $20Hz$ y $20kHz$.

Podemos hacer algo mas, con el siguiente codigo podemos modificar la frecuencia de la señal mandando datos por medio del monitor serial:

```
1  int motor_pin = 10;
2  float freq = 1000;
3  float T = 1000000/freq;
4  float dc = 0.5;
5  float t_on = T*dc;
6  float t_off = T*(1-dc);
7  String cad_frec = "";
8  void setup()
9  {
10   Serial.begin(9600);
11   pinMode(motor_pin, OUTPUT);
12 }
13 void loop()
14 {
15   if(Serial.available() > 0){
16     cad_frec = Serial.readString();
17     modificar_tiempos(cad_frec.toFloat());}
18   digitalWrite(motor_pin, HIGH);
19   delayMicroseconds(t_on);
20   digitalWrite(motor_pin, LOW);
21   delayMicroseconds(t_off);
22 }
23 void modificar_tiempos(float frecuencia)
24 {
25   freq = frecuencia;
26   T = 1000000/freq;
27   t_on = T*dc;
28   t_off = T*(1-dc);
29 }
```

En este código se agrega una función que va a cambiar las variables de configuración de la señal de control, se manda a llamar esta función cada que hay datos nuevos mandados desde la computadora.

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 de alimentación
- Multímetro
- Pinzas

Materiales

- Protoboard
- Cables
- Motor CD
- LED
- ULN2003A, L293D, L298N, etc.

Desarrollo

Diseña los circuitos requeridos, el código necesario para que estos funciones y responde las preguntas de la hoja de anotaciones.

Conclusiones

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

HOJA DE ANOTACIONES

1. Diseña el circuito necesario para conectar tu motor al Arduino, utilizando el CI que conseguiste.

2. Escribe el código necesario para que el motor conectado al Arduino cambie de velocidad, cuando se le envie el duty cycle por medio del monitor serial.

3. Si necesito que el motor gire en el sentido contrario, ¿Que cambio tengo que hacer en el circuito que diseñaste?

4. Escribe el código necesario para que el motor conectado al Arduino empiece a girar con la velocidad requerida en la instrucción `M3 S1000`, en donde 1000 se refiere a $1000rpm$; suponga que el motor trabaja en un rango de velocidades de $0rpm$ a $5000rpm$. El motor debe parar cuando se envia la instrucción `M5` o `M3 S0`. Cuando cada instrucción sea ejecutada, el Arduino debe mandar el texto ok por el puerto serial a la computadora.

Integrantes del equipo: Revisó:

_____

_____ _____