Laboratorio de Principios de Mecatrónica

11 de marzo de 2022

Práctica #4

Actuadores

Grupo: 1

L002

Estudiante:

- García Cortez Alejandro
- Hermida Flores Manuel Joaquín
- Chicatti Avendaño Josué Doménico

Profesor:

Benito Granados–Rojas

Índice

1. Introducción

Así como un microcontrolador tiene diferentes formas de recibir información del mundo, también tiene diferentes formas de actuar sobre el mundo exterior. Para el primer caso tenemos los sensores, que nos permiten leer el exterior, y para responder tenemos los actuadores, que nos permiten actuar sobre el exterior. En esta práctica trabajamos con un motor que, al recibir comandos, giraba a la izquierda, derecha o se detenía.

Si bien un microcontrolador Arduino tiene suficente potencia para controlar un pequeño motor, cuando nos enfretamos a dispositivos más grandes necesitamos de un Puente H. El puente H es un circuito electrónico que nos permite controlar el sentido de giro de un motor de corriente directa, además de que nos permite dar más potencia con una fuente externa [1].

Otra herramienta útil para el control de los motores, además del Puente H, son los PWM. PWM signifca "Pulse Wave Modulation" y consiste en mandar una señal cuadrada con diferente ciclo de trabajo. De acuerdo a este ciclo de trabajo es la forma en que reacciona el motor. A mayor ciclo de trabajo, más rápido girará el motor.

Apoyado en estos recursos, la práctica tiene como objetivo implementar un motor, junto con su correspondiente Puente H y analizar como diferentes ondas generadas por PWM afectan el funcionamiento del motor.

2. Experimentos y Simulaciones

2.1. Sentido de giro

Para este experimento utilizamos un puente H. Un puente H nos permite alimentar con una fuente externa mayor a 5V a los motores. Al mismo tiempo mantenemos el voltaje apropiado en el microcontrolador.

Realizamos las conexiones desde la fuente externa hasta el puente H. Nuestros motores operaron a 8V y utilizamos la salida de 5V del puente H para alimentar al Arduino.

En este primer experimento controlamos la dirección de giro del motor con dos push buttons. Para demostrar los cambios en la dirección de giro encendimos uno de dos LEDs. Además para observar la dirección fácilmente le agregamos una bandera al eje del motor.

El loop del código se ve de esta manera:

```
  void loop() {
if(digitalRead(btn1)==LOW && digitalRead(btn0)==LOW)
```

```
  {
    digitalWrite(led1, LOW);
    digitalWrite(led0, LOW);
    digitalWrite(Izq, LOW);
    digitalWrite(Der, LOW);
   }
  else if(digitalRead(btn1)==LOW && digitalRead(btn0)==HIGH)
  {
    digitalWrite(led1, LOW);
    digitalWrite(led0, HIGH);
    digitalWrite(Izq, HIGH);
    digitalWrite(Der, LOW);
  }
  else if(digitalRead(btn1)==HIGH && digitalRead(btn0)==LOW)
  {
    digitalWrite(led1, HIGH);
    digitalWrite(led0, LOW);
    digitalWrite(Izq, LOW);
    digitalWrite(Der, HIGH);
  }
  else if(digitalRead(btn1)==HIGH && digitalRead(btn0)==HIGH)
  {
    digitalWrite(led1, HIGH);
    digitalWrite(led0, HIGH);
    digitalWrite(Izq, HIGH);
    digitalWrite(Der, HIGH);
  }
}
```

Se puede ver que cuando ningun botón está presionado no hay movimiento. Cuando uno de los dos está presionado, el motor gira en una dirección. Y cuando ambos se presionan el motor se frena. Todo esto se logra con una condición que checa el estado de los botones mediante la función de *digitalRead(boton)*.

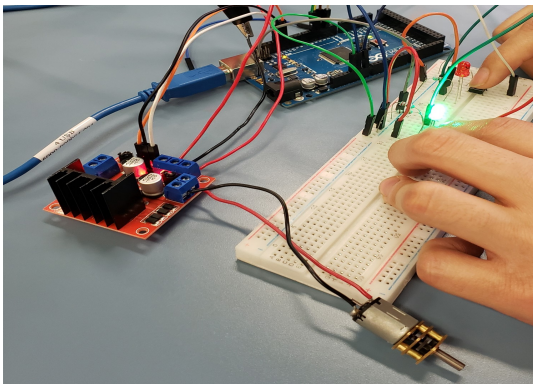

Figura 1: Implementación de giro con dos botones

2.2. Control proporcional de velocidad

El segundo ejercicio consistió en intercambiar los push button previamente usados por un potenciómetro. Se conectó a la protoboard tal como se ha hecho en prácticas pasadas. Su función es hacer girar el motor con cierta dirección y velocidad de acuerdo a su posición.

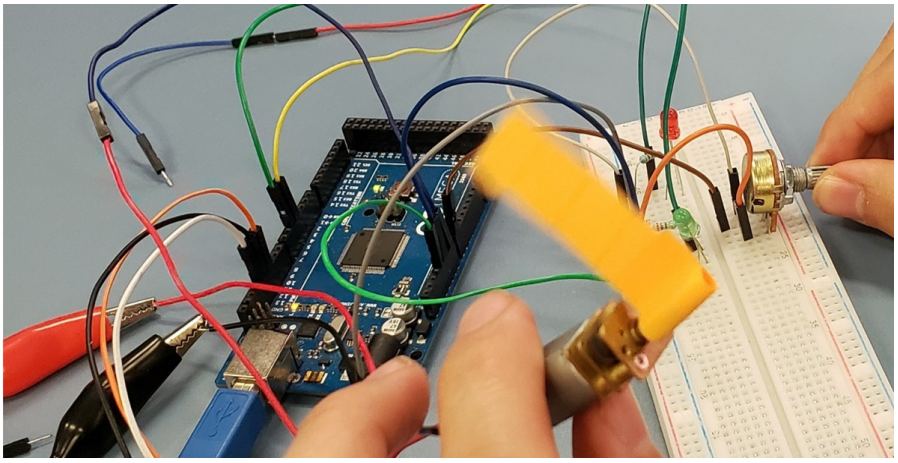

Figura 2: Implementación de giro proporcional a potenciómetro

Sabemos que su rango de valores está entre 0 y 1023, los cuales son recibidos con la función de *analogRead()*. Junto a la lectura del valor habilitamos el enable para permitir el giro.

```
val = analogRead(Pot);
digitalWrite(Enb, HIGH);
```

Posteriormente vimos que el centro se encontraba en 512, por lo que si recibíamos valores menores al mismo giramos a izquierda; en el caso contrario, a la derecha. El la posición central tenemos el estado de freno activo. Usamos la función *digitalWrite()* cuando queremos apagar algún "giro". Para girar con cierta proporción usamos *analogWrite()* para enviar una onda PWM al pin, que hará que la velocidad varíe. Aquí es donde hacemos la conversión con el rango de la función que va de 0 a 255. Para hacer correcta la proporción hay que tomar en cuenta si estamos en un giro a la izquierda o a la derecha. La parte del código que decide el giro se pone a continuación:

```
if(val<512) {
    digitalWrite(Der, LOW);
    analogWrite(Izq, (512 - val)/2);
}
else if(val>512){
    digitalWrite(Izq, LOW);
    analogWrite(Der, (val - 512)/2);
}
```

```
else {
    digitalWrite(Izq, LOW);
    digitalWrite(Der, LOW);
}
```

Cabe añadir que inicialmente habíamos realizado esta conversión antes de checar las condiciones y mandarlo a la función. Esto generó que nuestro primer giro se realizará más lento, ya que los valores mandados eran más pequeños.

3. Conclusiones

Aunque la práctica fue corta, los ejercicios realizados fueron ilustrativos. Los Puente H son dispositivos muy útiles, por lo que aprender a usarlos y conectarlos, así como conocer sus aplicaciones para prácticas futuras fue educativo. Controlar el sentido de giro es tan sencillo como mandar una señal alta en uno u otro canal. También fue interesante ver cómo el motor se puede detener de una forma ya sea "activa", ya sea "pasiva".

Para la segunda práctica fue interesante analizar también como los diferentes pulsos afectaban la velocidad del motor. Si bien todavía necesitabamos activar uno u otro canal para determinar el giro del motor, lo que realmente cambiaba era la señal. Entre mayor fuera el ciclo de trabajo, más rápido giraba el motor. De aquí aprendimos como la velocidad del motor se puede modificar desde la programación, sin preocuparnos por el cableado físico.

Durante esta práctica no solo aprendimos a controlar un motor de corriente directa, sino también a cablear un Puente H y controlar la velocidad con una señal cuadrada generada por PWM.

4. Enlaces externos

`https://github.com/ManoHF/lab-mecatronica`

5. Referencias

[1]Rslicing 3D, "Tipos de Puentes H," *Rslicing3d.com*, Feb. 21, 2021. https://www.rslicing3d.com/programacion-arduino-complementos/puente-h-arduino/ (accessed Mar. 12, 2022).