EGR 597: Internet of Things: Standards, Protocols, and Applications
Lab 5: DC Motor and Servo Motor

Central Michigan University
2018-2019

Objective: The objective of this lab is to learn how to operate DC motors and Servo motor, build H-bridges, and vary speed of the motors using PWM signals.

In part 1 we will learn the fundamental principles of DC motors a negative temperature coefficient thermistor and how to calibrate them. In part 2 we will learn to use BME680, a digital temperature and humidity sensor.

Materials Needed:

1) Raspberry Pi 3
2) DC Motor
3) Servo Motor
4) MOSFETs
5) Prototyping breadboard
6) Resistors
7) Jumper wires.

Part 1: DC Motors

Direct current (DC) motors are commonly used in toys, power tools, and even appliances. DC motors convert electrical energy into mechanical energy, their operation is based on the principle that when a current carrying conductor is placed in a magnetic field, the conductor experiences a mechanical force.

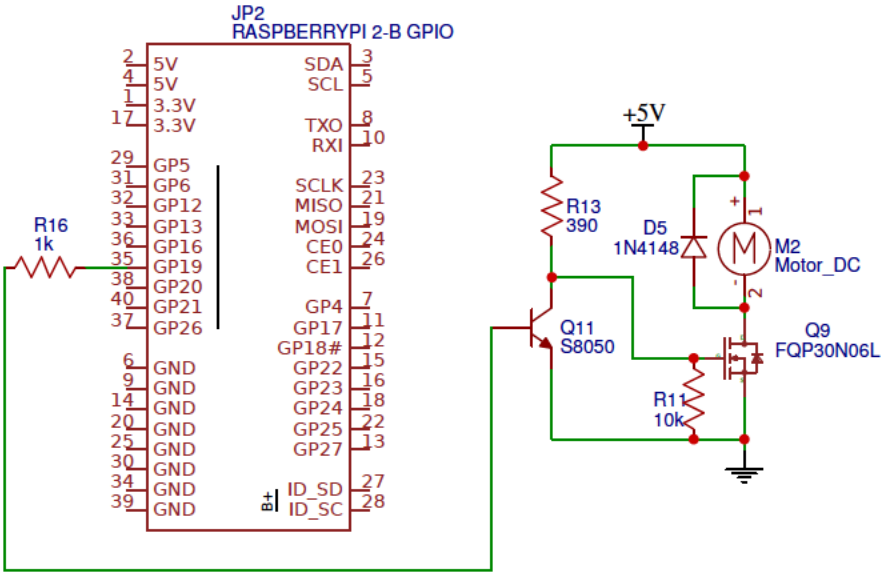


Fig. 1: Schematic to operate a DC motor from Raspberry Pi

Fig. 1 shows a simple schematic to operate a DC motor using a Raspberry Pi. A combination of resistors, diodes, and transistors are necessary to provide enough power for the motor to operate.
Write a simple Python code to turn ON and turn OFF the motor at a rate of 0.50 Hz.

Direction Control of DC motors:

The direction of the DC motor can be reversed just by controlling the polarity of our voltage source. Let's say a DC motor as in Fig. 2. has two terminal called A and B runs in the clockwise direction when terminal A is connected to positive of the voltage source and B is connected to the negative terminal. We can make the motor run in the opposite direction just by connecting the voltage positive to terminal B and negative to terminal A. So basically we are controlling the flow of current through the motor coil to change directions. But how can we switch the voltage polarity on the fly? This is where H-bridge comes in. H bridge circuit enables us to control the flow of current by using four electronic switches (here they are MOSFETS).

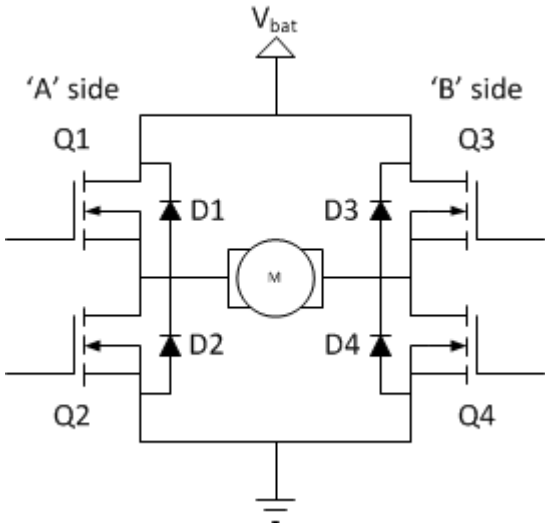


Fig.2: H-Bridge Circuit

Let's analyse the circuit given above. Q1 - Q4 are the electronic switches used to control the flow of current through the motor coil. If we provide control signal to Q1 and Q4, they start conducting and current flows from left to ride side or A to B side of the motor so the motor starts running in a certain direction. When we provide control signal to Q3 and Q2 the direction of the current flow through the motor coils changes and current flows from right to left or B to A side, changing the direction of the motor's motion. Switches on the same side must not be conducting because it will effectively short the voltage supply to the ground. The diodes across the drain and source of the MOSFETs are called Back EMF or Freewheeling diodes and are necessary for safe operation of the motor.

Utilizing these principles, build the H-Bridge circuit as shown in Fig. 3. Write a simple Python code to turn ON and turn OFF the motor at a rate of 0.50 Hz moving forward, and then backward at the same rate.

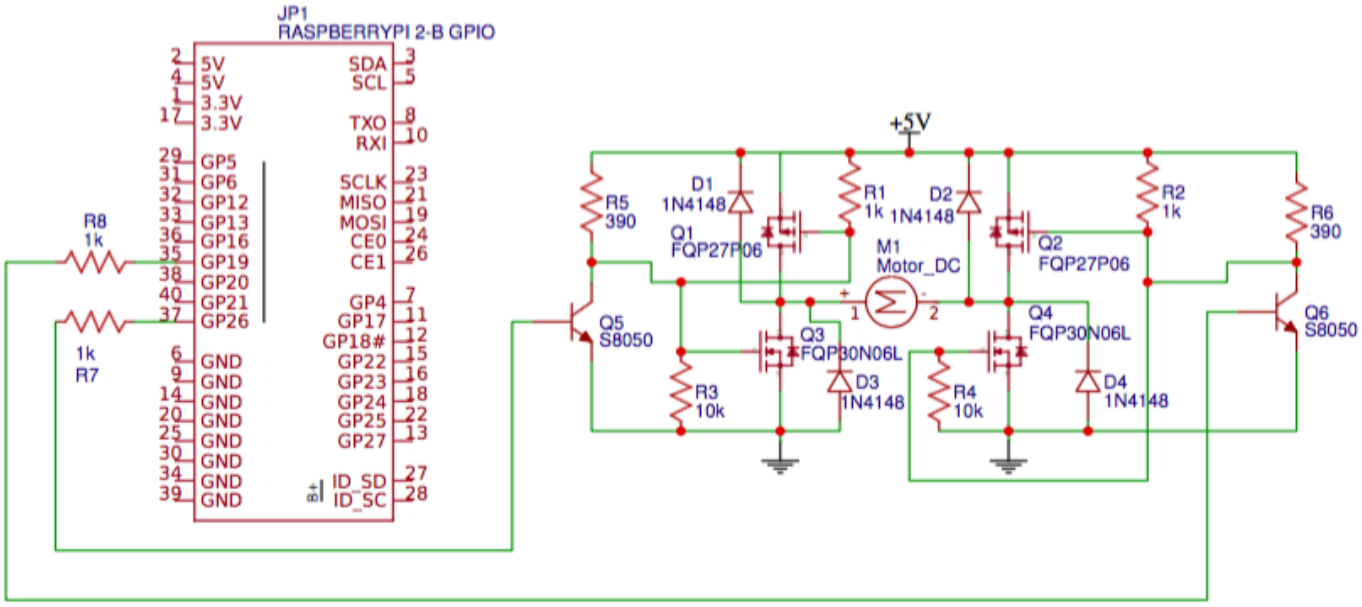
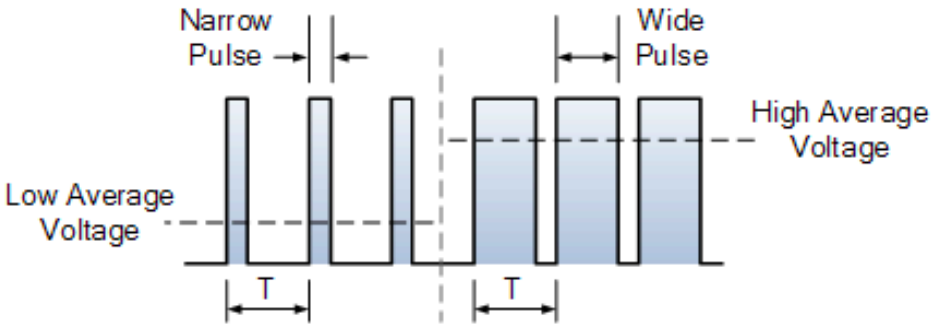


Fig. 3. H-Bridge Wiring Schematic for Raspberry Pi

Speed Control of DC motors:

The speed of a DC motor can be controlled in several different ways, today we will be using a simple method, pulse width modulation. Other methods of controlling the speed often generate a lot of heat and wasted power in the resistance, with PWM this does not happen. The power applied to the motor can be controlled by varying the width of applied pulses. By changing the timing of these pulses the speed of the motor can be controlled. The longer the pulse is “ON”, the faster the motor will rotate, the shorter the pulse is “ON” the slower the motor will rotate. The graphic below shows the difference between a narrow and wide pulse.




```
import RPi.GPIO as GPIO

import time                #calling time to provide delays in program

GPIO.setmode (GPIO.BCM)

GPIO.setup(19,GPIO.OUT)        # initialize GPIO19 as an output.
GPIO.setup(26,GPIO.OUT)
GPIO.output(26, GPIO.HIGH)


p = GPIO.PWM(19,100)        #GPIO19 as PWM output, with 100Hz frequency
p.start(0)                #generate PWM signal with 0% duty cycle

try:

        while 1:                #execute loop forever

                for x in range (50):      #execute loop for 50 times, x being
incremented from 0 to 49.
                p.ChangeDutyCycle(x)  #change duty cycle for varying the
brightness of LED.
                time.sleep(0.1)      #sleep for 100m second
```

```
            for x in range (50):      #execute loop for 50 times, x being
incremented from 0 to 49.
            p.ChangeDutyCycle(50-x)
            time.sleep(0.1)

except KeyboardInterrupt:
      pass
      p.stop()
      GPIO.cleanup()
```

Part 2: Servo Motors

Fig. 4. Servo Motor

The heart of a servo is a small direct current (DC) motor, so servos are controlled by sending them a pulse of variable width, just like how we controlled the DC motor in the previous section of this lab. It also has several gears and wings that can move to different positions. Servo motors can be rotated from 0 to 180 degrees by controlling the width of the electrical pulse delivered. Servos check the pulse delivered every 20 milliseconds, a pulse of 1 millisecond width can rotate the servo to 0 degrees, 1.5 milliseconds can rotate it to 90 degrees (neutral position), and a 2 millisecond pulse can rotate it to 180 degrees.

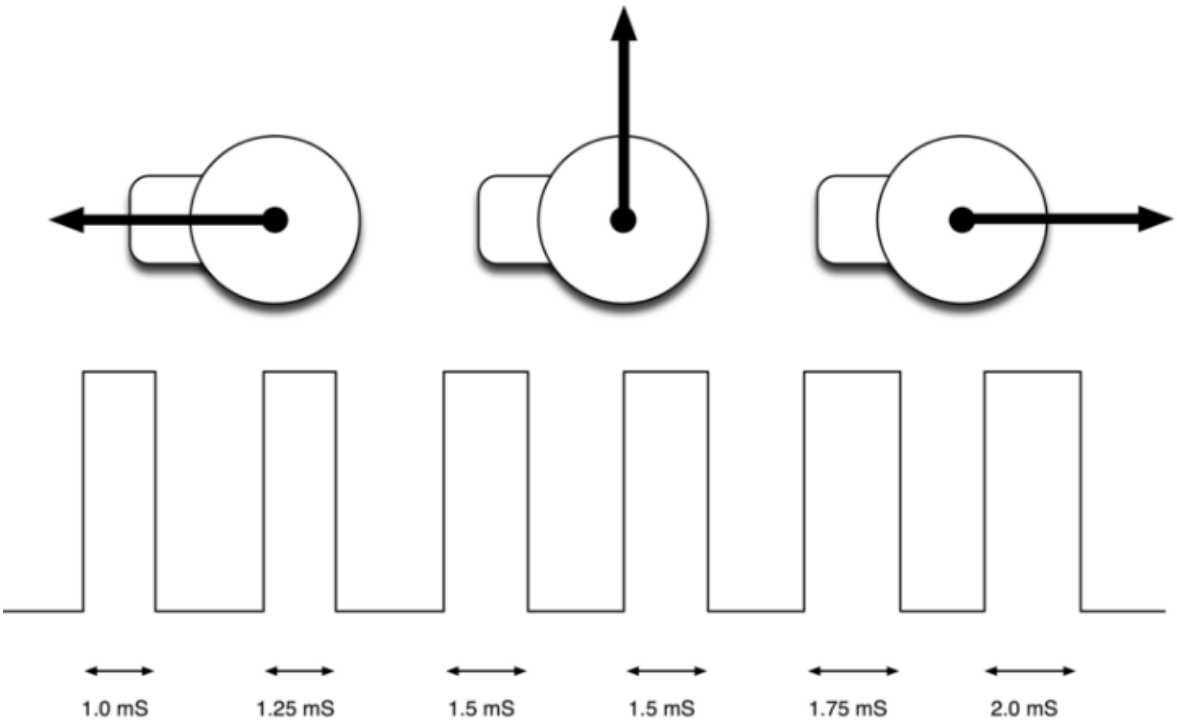


Fig. 5. Relation between PWM Signal and Direction of Motor

The pin configurations are as follows:

RED – 5 V
BROWN – GND
YELLOW – GPIO 17

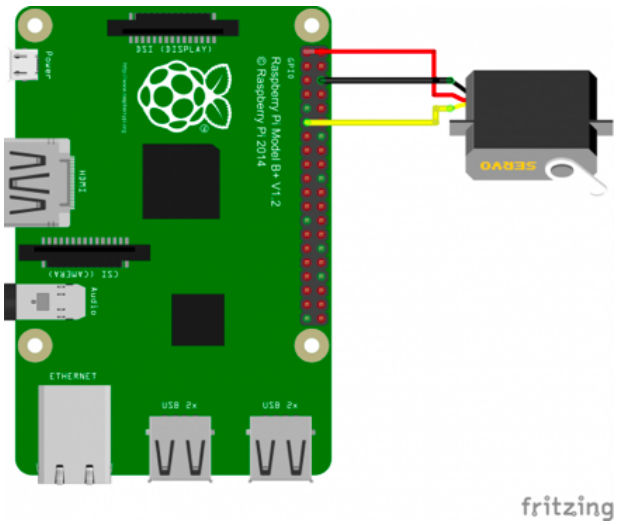


Fig. 6. Wiring Schematic for a Servo Motor

Servo Motor Code

```
import RPi.GPIO as GPIO
import time
servoPIN = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(servoPIN, GPIO.OUT)
p = GPIO.PWM(servoPIN, 50) # GPIO 17 for PWM with 50Hz
p.start(2.5) # Initialization
try:
  while True:
        p.ChangeDutyCycle(5)
        time.sleep(0.5)
        p.ChangeDutyCycle(7.5)
        time.sleep(0.5)
        p.ChangeDutyCycle(10)
        time.sleep(0.5)
        p.ChangeDutyCycle(12.5)
        time.sleep(0.5)
        p.ChangeDutyCycle(10)
        time.sleep(0.5)
        p.ChangeDutyCycle(7.5)
        time.sleep(0.5)
        p.ChangeDutyCycle(5)
        time.sleep(0.5)
        p.ChangeDutyCycle(2.5)
        time.sleep(0.5)
except KeyboardInterrupt:
  p.stop()
  GPIO.cleanup()
```

Lab Report:

Include the following in your lab report

1. Lessons learned
2. Schematics and hookups.
3. Python code to operate the motor in Fig. 1 at a rate of 0.5 Hz.
4. Python code to operate the motor and H-Bridge as in Fig. 3.

5. What are the pros and cons of using Servo Motor
6. What are the pros and cons of using DC Motor