Motor & Driver

Overview

In this chapter, we will learn some knowledge about DC motor and DC motor drive, and how to control the speed and direction of motor.

Experimental Materials:

Raspberry Pi *1

Breadboard power module *1

T-type expansion board *1

Breadboard *1

PCF8591 *1

Rotary potentiometer *1

L293D *1

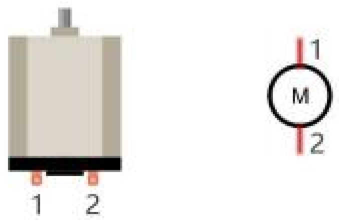
Motor *1

Some DuPont lines

Product description：

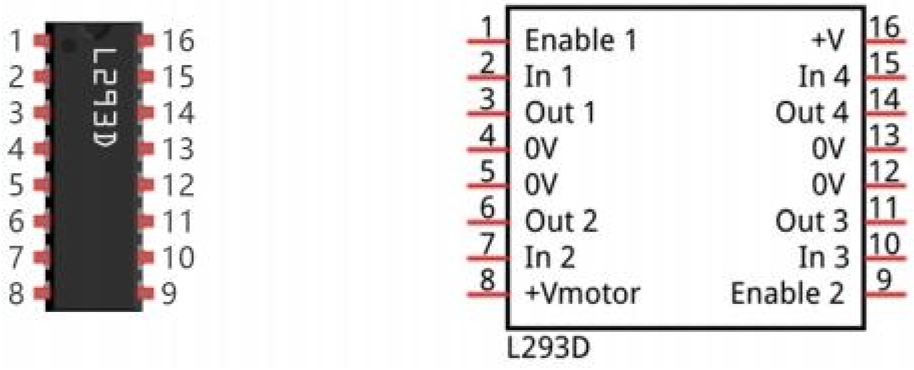
Motor

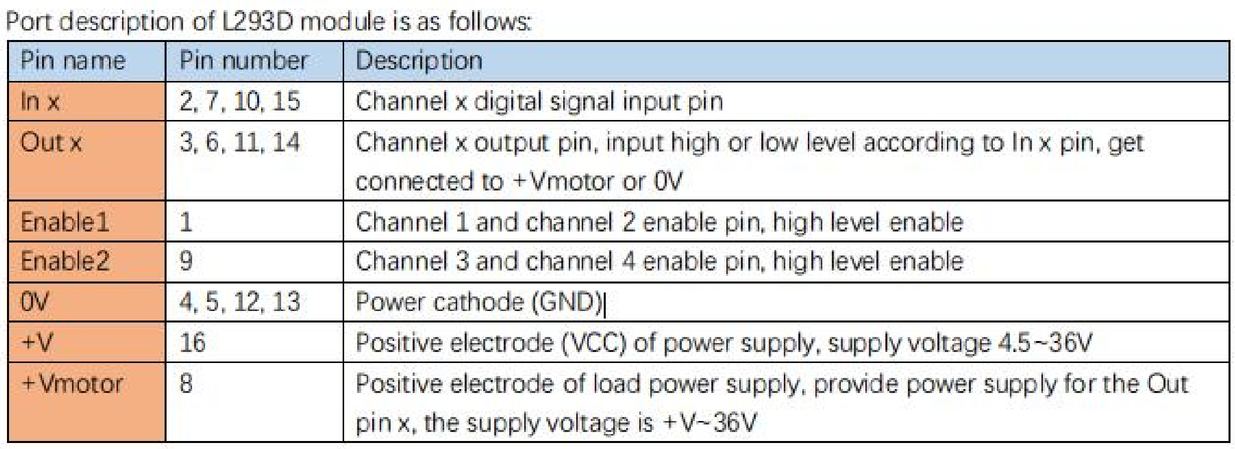
Motor is a device that converts electrical energy into mechanical energy. Motor consists of two parts: stator and rotor. When motor works, the stationary part is stator, and the rotating part is rotor. Stator is usually the outer case of motor, and it has terminals to connect to the power. Rotor is usually the shaft of motor, and can drive other mechanical devices to run. Diagram below is a small DC motor with two pin



When motor get connected to the power supply, it will rotate in one direction. Reverse the polarity of power supply, then motor rotates in opposite direction

L293D is a chip integrated with 4-channel motor drive. You can drive a unidirectional motor with 4 ports or abi-directional motor with 2 port or a stepper motor.



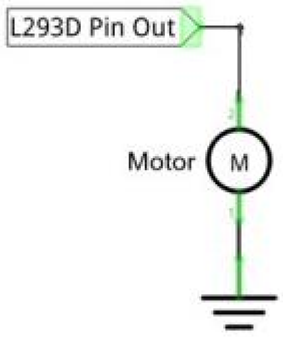


Port description of L293D module is as follows:

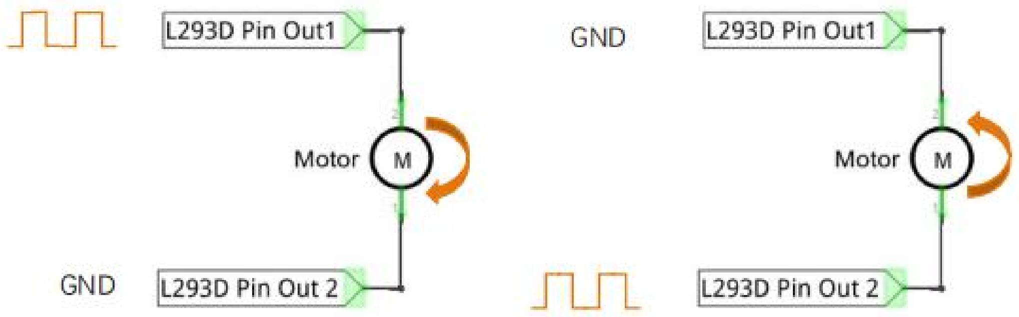
Pin name	Pin number	Description
In x	2, 7, 10, 15	Channel x digital signal input pin
Out x	3, 6, 11, 14	Channel x output pin, input high or low level according to In x pin, get connected to +Vmotor or 0V
Enable1	1	Channel 1 and channel 2 enable pin, high level enable
Enable2	9	Channel 3 and channel 4 enable pin, high level enable
0V	4, 5, 12, 13	Power cathode (GND)|
+V	16	Positive electrode (VCC) of power supply, supply voltage 4.5~36V
+Vmotor	8	Positive electrode of load power supply, provide power supply for the Out pin x, the supply voltage is +V~36V

When using L293D to drive DC motor, there are usually two kinds of connection.

Following connection uses one channel, and it can control motor speed through PWM, but the motor can only rotate in one direction

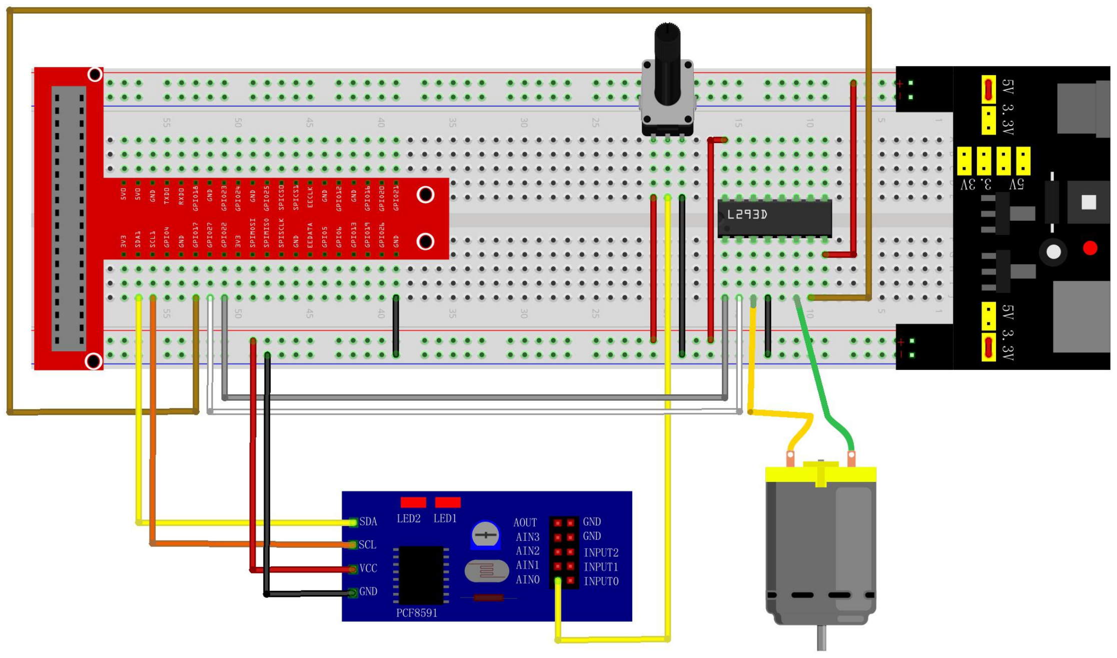


Following connection uses two channels: one channel outputs PWM wave, and another channel connects GND, so you can control the speed of motor. When these two channel signals are exchanged, the current direction of the motor can be reversed, and the motor will rotate in reverse direction. This can not only control the speed of motor, but also can control the steering of motor.



In actual use, motor is usually connected to the channel 1 and 2, output different level to in1 and in2 to control the rotation direction of the motor, and output PWM wave to Enable1 port to control the motor rotation speed. Or, get motor connected to the channel 3 and 4, output different level to in3 and in4 to control the motor's rotation direction, and output PWM wave to Enable2 pin to control the motor rotation speed.

Wiring diagram:



C code:

```
#include <wiringPi.h>
#include <pcf8591.h>
#include <stdio.h>
#include <softPwm.h>
#include <math.h>
#include <stdlib.h>

#define address 0x48       //pcf8591 default address
#define pinbase 64         //any number above 64
#define A0 pinbase + 0
#define A1 pinbase + 1
#define A2 pinbase + 2
#define A3 pinbase + 3

#define motorPin1  2       //define the pin connected to L293D
#define motorPin2  0
#define enablePin  3
//Map function: map the value from a range of mapping to another range.
long map(long value,long fromLow,long fromHigh,long toLow,long toHigh){
    return (toHigh-toLow)*(value-fromLow) / (fromHigh-fromLow) + toLow;
}
//motor function: determine the direction and speed of the motor according to
```

```
the ADC
void motor(int ADC){
    int value = ADC -128;
    if(value>0){
        digitalWrite(motorPin1,HIGH);
        digitalWrite(motorPin2,LOW);
        printf("turn Forward...\n");
    }
    else if (value<0){
        digitalWrite(motorPin1,LOW);
        digitalWrite(motorPin2,HIGH);
        printf("turn Back...\n");
    }
    else {
        digitalWrite(motorPin1,LOW);
        digitalWrite(motorPin2,LOW);
        printf("Motor Stop...\n");
    }
    softPwmWrite(enablePin,map(abs(value),0,128,0,255));
    printf("The PWM duty cycle is %d%%\n",abs(value)*100/127);//print the PMW
duty cycle
}
int main(void){
    int value;
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print messageto
screen
        printf("setup wiringPi failed !");
        return 1;
    }
    pinMode(enablePin,OUTPUT);//set mode for the pin
    pinMode(motorPin1,OUTPUT);
    pinMode(motorPin2,OUTPUT);
    softPwmCreate(enablePin,0,100);//define PMW pin
    pcf8591Setup(pinbase,address);//initialize PCF8591

    while(1){
        value = analogRead(A0);  //read A0 pin
        printf("ADC value : %d \n",value);
        motor(value);      //start the motor
        delay(100);
    }
    return 0;
}
```

Python code：

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import smbus
import time

address = 0x48
bus=smbus.SMBus(1)
cmd=0x40
# define the pin connected to L293D
motoRPin1 = 13
motoRPin2 = 11
enablePin = 15

def analogRead(chn):
    value = bus.read_byte_data(address,cmd+chn)
    return value

def analogWrite(value):
    bus.write_byte_data(address,cmd,value)

def setup():
    global p
    GPIO.setmode(GPIO.BOARD)    # set mode for pin
    GPIO.setup(motoRPin1,GPIO.OUT)
    GPIO.setup(motoRPin2,GPIO.OUT)
    GPIO.setup(enablePin,GPIO.OUT)

    p = GPIO.PWM(enablePin,1000)# creat PWM
    p.start(0)
#mapNUM function: map the value from a range of mapping to another range.

def mapNUM(value,fromLow,fromHigh,toLow,toHigh):
    return (toHigh-toLow)*(value-fromLow) / (fromHigh-fromLow) + toLow
#motor function: determine the direction and speed of the motor according to the ADC value to be input.
def motor(ADC):
    value = ADC -128
    if (value > 0):
        GPIO.output(motoRPin1,GPIO.HIGH)
        GPIO.output(motoRPin2,GPIO.LOW)
        print ('Turn Forward...')
    elif (value < 0):
        GPIO.output(motoRPin1,GPIO.LOW)
        GPIO.output(motoRPin2,GPIO.HIGH)
        print ('Turn Backward...')
    else :
        GPIO.output(motoRPin1,GPIO.LOW)
        GPIO.output(motoRPin2,GPIO.LOW)
        print ('Motor Stop...')
    p.start(mapNUM(abs(value),0,128,0,100))
    print ('The PWM duty cycle is %d%%\n'%(abs(value)*100/127))   #print PMW
duty cycle.

def loop():
    while True:
        value = analogRead(0)
        print ('ADC Value : %d'%(value))
        motor(value)
        time.sleep(0.01)

def destroy():
    bus.close()
    GPIO.cleanup()

if __name__ == '__main__':
    print ('Program is starting ... ')
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

Experimental results:

In the directory where the code file is located, execute the following command

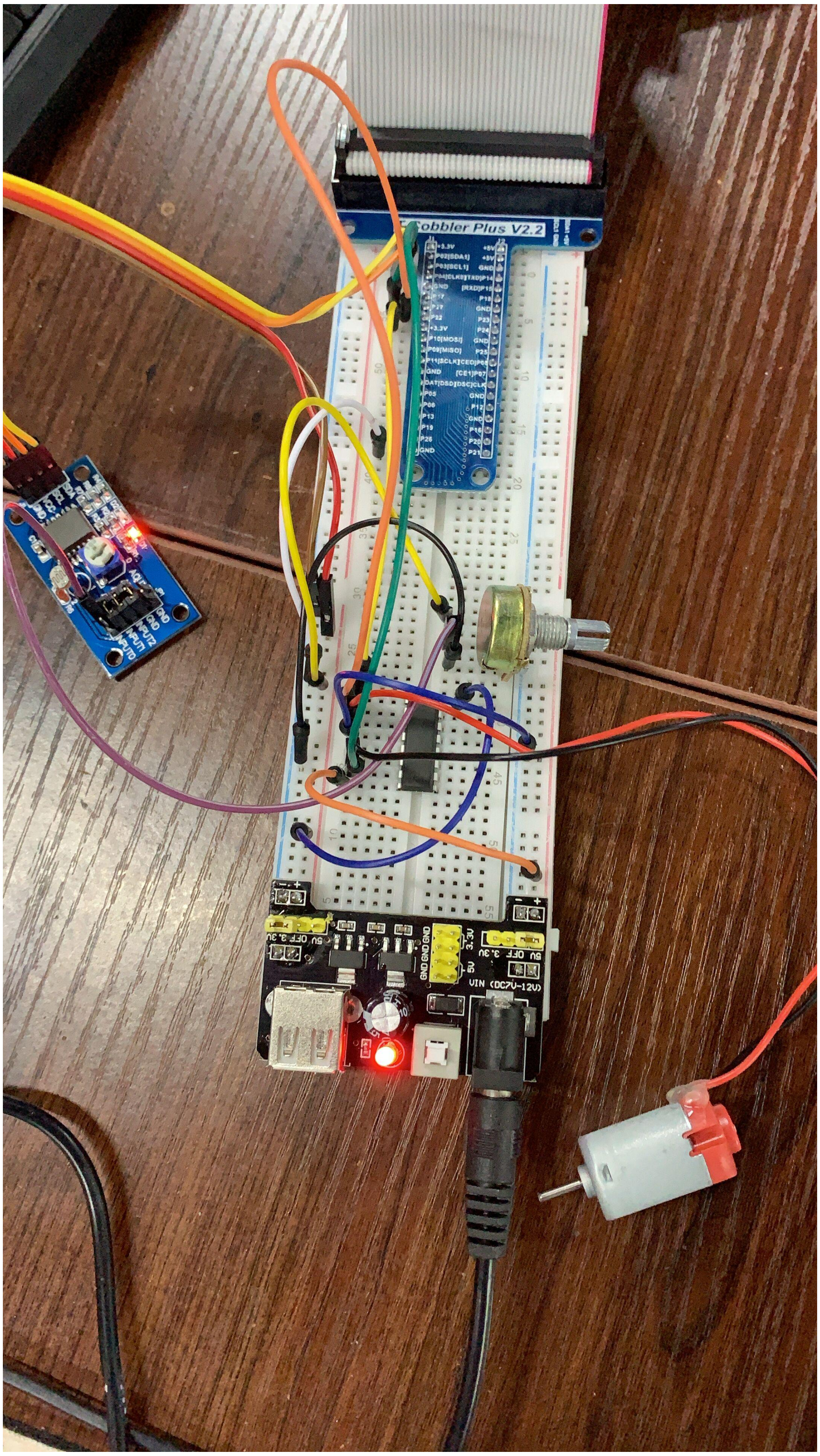
C：

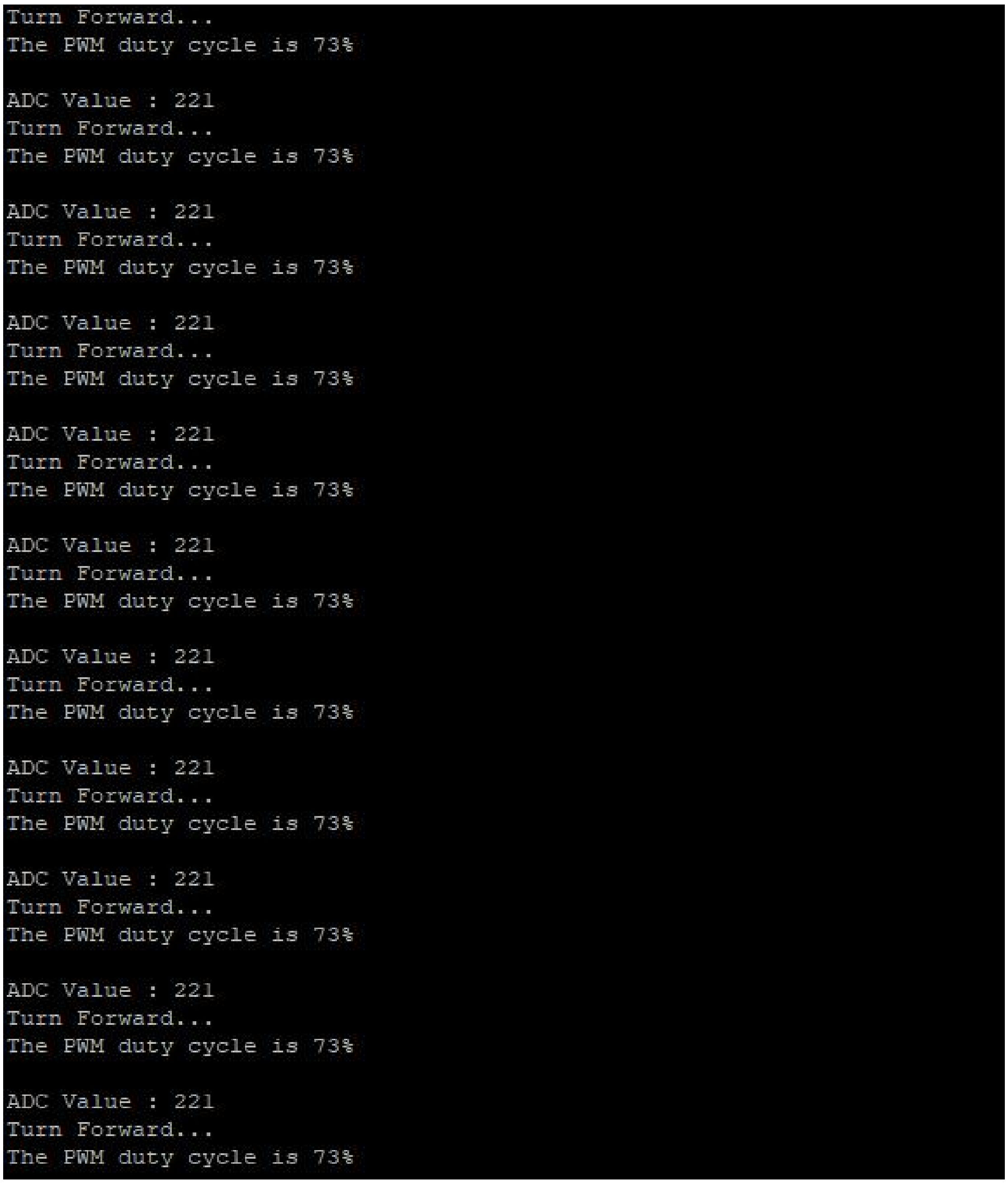
```
gcc -Wall -o Motor Motor.c -lwiringPi
sudo ./Motor
```

Python:

```
python Motor.py
```

After the program is executed, shift the potentiometer, then the rotation speed and direction of the motor will change with it. And when the potentiometer is turned to midpoint position, the motor stops running. When away from the middle position, the motor speed will increase. When to both ends, motor speed reach to maximum. When the potentiometer is turned to different side of the middle position, the motor will run with different direction. Meanwhile, the terminal will print out ADC value of the potentiometer, the motor direction and the PWM duty cycle used to control motor speed.

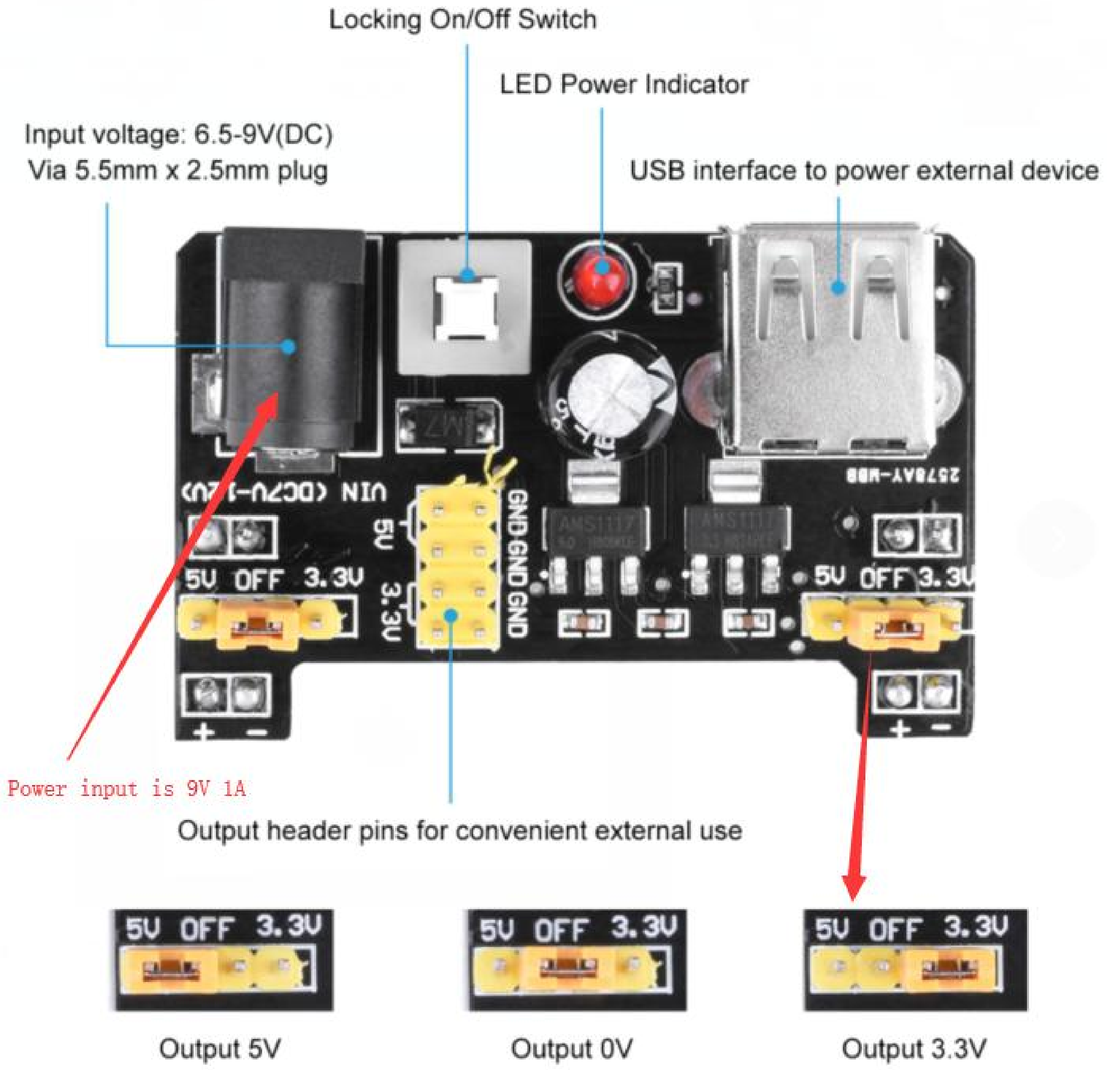

Cobbler Plus V2.2



```
Turn Forward...
The PWM duty cycle is 73%

ADC Value : 221
Turn Forward...
The PWM duty cycle is 73%

ADC Value : 221
Turn Forward...
The PWM duty cycle is 73%

ADC Value : 221
Turn Forward...
The PWM duty cycle is 73%

ADC Value : 221
Turn Forward...
The PWM duty cycle is 73%

ADC Value : 221
Turn Forward...
The PWM duty cycle is 73%

ADC Value : 221
Turn Forward...
The PWM duty cycle is 73%

ADC Value : 221
Turn Forward...
The PWM duty cycle is 73%

ADC Value : 221
Turn Forward...
The PWM duty cycle is 73%

ADC Value : 221
Turn Forward...
The PWM duty cycle is 73%

ADC Value : 221
Turn Forward...
The PWM duty cycle is 73%
```

note:

The power supply of breadboard should be 9V. The diameter of the connector is 5.5*2.1mm. The polarity of the connector is internal positive pole, external negative pole, and the current is over 1A. Please pay attention to the selection of the required voltage by using the yellow connector. The figure below is the usage method of the power module.



Note: Before using this tutorial, please open I2C and check the tutorial of PCF8591.