21.DC_Motor_Control

Introduction

In this lesson, we will learn to how to use L293D to drive a DC motor and make it rotate clockwise and counterclockwise. Since the DC Motor needs a larger current, for safety purpose, here we use the Power Supply Module to supply motors.

Hardware Required

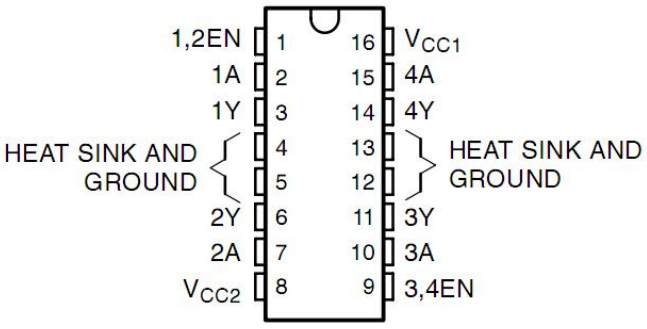
- ✓ 1 * Raspberry Pi
- ✓ 1 * T-Extension Board
- ✓ 1 * Power Module (with 9VPower Adapter or 9V battery and buckle)
- ✓ 1 * 40-pin Cable
- ✓ 1 * L293D
- ✓ Several Jumper Wires
- ✓ 1 * Breadboard
- ✓ 1 * DC Motor

Principle

L293D

This is a very practical chip that can independently control two DC motors. In this experiment, just half of the chip is used. Since most pins on the right side of the chip are used to control the second motor, they will not be used here.

L293D has two pins (Vcc1 and Vcc2) for power supply. Vcc2 is used to supply power for the motor, while Vcc1 to supply for the chip. Since a small-sized DC motor is used here, connect both pins to +5V. If you use a high power motor, connect Vcc2 to an external power supply. At the same time, the GND of L293D should be connected to that of the RexQualis Uno board.



21.DC_Motor_Control

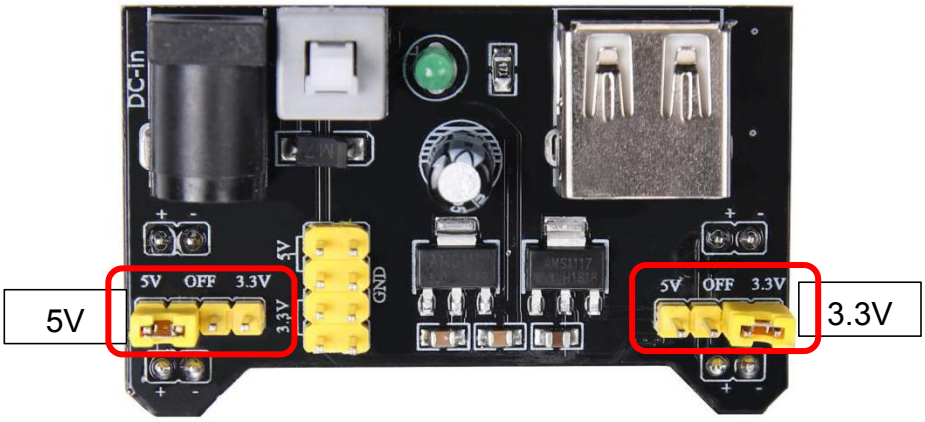
Breadboard Power Supply

The small DC motor is likely to use more power than an UNO R3 board digital output can handle directly. If we tried to connect the motor straight to an UNO R3 board pin, there is a good chance that it could damage the UNO R3 board. So we use a power supply module provides power supply.

Product Specifications:

- ✓ Locking On/Off Switch
- ✓ LED Power Indicator
- ✓ Input voltage:6.5-9v(DC) via 5.5mm x 2.1mm plug
- ✓ Output voltage:3.3V/5V
- ✓ Maximum output current:700 mA
- ✓ Independent control rail output.0v, 3.3v, 5v to breadboard
- ✓ Output header pins for convenient external use
- ✓ Size:2.1 in x 1.4 in
- ✓ USB device connector on board to power external device

Setting up output voltage:



The left and right voltage output can be configured independently. To select the output voltage, move jumper to the corresponding pins.

Note: power indicator LED and the breadboard power rails will not power on if both jumpers are in the “OFF” position.

Important note:

21.DC_Motor_Control

Make sure that you align the module correctly on the breadboard. The negative pin(-) on module lines up with the blue line(-) on breadboard and that the positive pin(+) lines up with the red line(+).Failure to do so could result in you accidently reversing the power to your project

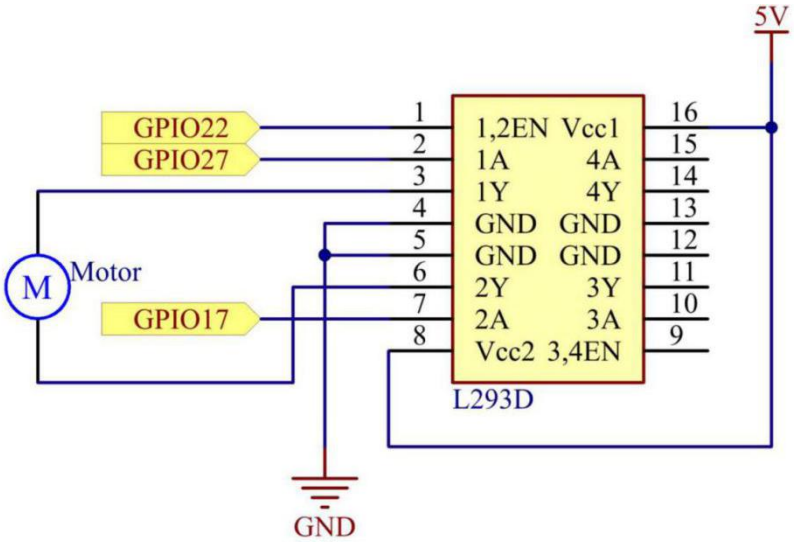
DC Motor Specifications

- ✓ Voltage：3-6V
- ✓ Main Size:length 25mm,thickness 15 mm,width 20mm
- ✓ Motor Shaft Length:9mm,Shaft Diameter 2mm
- ✓ Rated Voltage:3V
- ✓ Reference Current:0.35-0.4A
- ✓ 3V Rotating Speed:13000ROM

Schematic Diagram

Plug the power supply module in breadboard, and insert the jumper cap to pin of 5V, then it will output voltage of 5V. Connect pin 1 of L293D to GPIO22, and set it as high level. Connect pin2 to GPIO27, and pin7 to GPIO17, then set one pin high, while the other low. Thus you can change the motor's rotation direction.

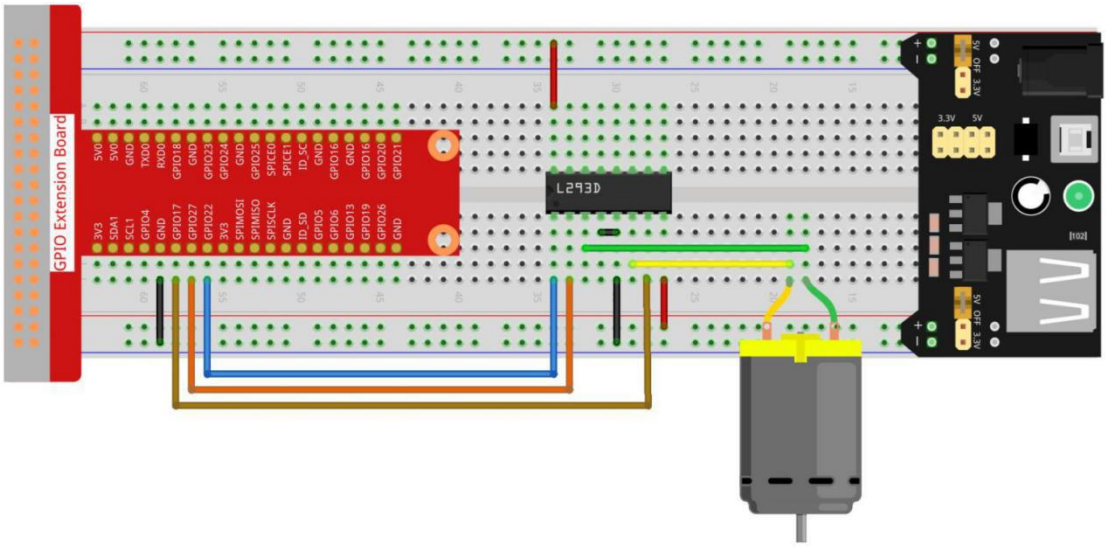
T-Board Name	physical	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO27	Pin 13	2	27
GPIO22	Pin 15	3	22



21.DC_Motor_Control

Experimental Procedures

Step 1: Build the circuit.



Note: The power module can apply a 9V battery with the 9V Battery Buckle in the kit. Insert the jumper cap of the power module into the 5V bus strips of the breadboard.

For C Language Users

Step 2: Get into the folder of the code.

```
cd /home/pi/REXQualis_Raspberry_Pi_Complete_Starter_Kit/C/21.DC_Motor_Control
```

Step 3: Compile the code.

```
gcc 21.DC_Motor_Control.c -o DC_Motor_Control.out -lwiringPi
```

Step 4: Run the executable file above.

```
sudo ./DC_Motor_Control.out
```

As the code runs, the motor first rotates clockwise for 5s then stops for 5s, after that, it rotates anticlockwise for 5s; subsequently, the motor stops for 5s. This series of actions will be executed repeatedly.

21.DC_Motor_Control

Code

```
#include <wiringPi.h>
#include <stdio.h>

#define MotorPin1       0   //2A
#define MotorPin2       2   //1A
#define MotorEnable     3   //enable

//main control the L293SD

int main(void){
    int i;
    if(wiringPiSetup() == -1){ //when initialize wiring failed, print message to screen
        printf("setup wiringPi failed !");
        return 1;
    }

    pinMode(MotorPin1, OUTPUT);
    pinMode(MotorPin2, OUTPUT);
    pinMode(MotorEnable, OUTPUT);

    while(1){
        printf("Clockwise\n");
        digitalWrite(MotorEnable, HIGH);
        digitalWrite(MotorPin1, HIGH);//2A is high
        digitalWrite(MotorPin2, LOW);//1A is low
        for(i=0;i<3;i++){
            delay(1000);    //delay time 1000
        }
```

```
        printf("Stop\n");
        digitalWrite(MotorEnable, LOW);      // not enable
        for(i=0;i<3;i++){
            delay(1000);    //delay 10000
        }

        printf("Anti-clockwise\n");
        digitalWrite(MotorEnable, HIGH);    //enable
        digitalWrite(MotorPin1, LOW);      //2A is low
        digitalWrite(MotorPin2, HIGH);    //1A is high
        for(i=0;i<3;i++){
            delay(1000);    //delay 1000
        }

        printf("Stop\n");
        digitalWrite(MotorEnable, LOW);    //not enable
        for(i=0;i<3;i++){
            delay(1000);
        }
    }
    return 0;
}
```

Code Explanation

```
        digitalWrite(MotorEnable, HIGH);
```

Enable the L239D.

```
        digitalWrite(MotorPin1, LOW);      //2A is low
        digitalWrite(MotorPin2, HIGH);    //1A is high
```

Set a high level for 2A(pin 7); since 1,2EN(pin 1) is in high level, 2Y will output high level.

Set a low level for 1A, then 1Y will output low level, and the motor will rotate.

```
for(i=0;i<3;i++){
    delay(1000);    //delay 10000
}
```

this loop is to delay for 3*1000ms.

```
digitalWrite(MotorEnable, LOW);
```

If 1,2EN (pin1) is in low level, L293D does not work. Motor stops rotating.

```
digitalWrite(MotorPin1, LOW);      //2A is low
digitalWrite(MotorPin2, HIGH);    //1A is high
```

Reverse the current flow of the motor, then the motor will rotate reversely.

For Python Language Users

Step 2: Get into the folder of the code.

```
cd /home/pi/REXQualis_Raspberry_Pi_Complete_Starter_Kit/Python
```

Step 3: Run.

```
sudo python3 21.DC_Motor_Control.py
```

As the code runs, the motor first rotates clockwise for 5s then stops for 5s, after that,it rotates anticlockwise for 5s; subsequently, the motor stops for 5s. This series of actions will be executed repeatedly.

Code

The code here is for Python3, if you need for Python2, please open the code with the suffix py2 in the attachment.

```
#!/usr/bin/env python3


import RPi.GPIO as GPIO
import time
```

```
# Set up pins
MotorPin1    = 17
MotorPin2    = 27
MotorEnable = 22

def setup():
    # Set the GPIO modes to BCM Numbering
    GPIO.setmode(GPIO.BCM)
    # Set pins to output
    GPIO.setup(MotorPin1, GPIO.OUT)
    GPIO.setup(MotorPin2, GPIO.OUT)
    GPIO.setup(MotorEnable, GPIO.OUT, initial=GPIO.LOW)

# Define a motor function to spin the motor
# direction should be
# 1(clockwise), 0(stop), -1(counterclockwise)
def motor(direction):
    # Clockwise
    if direction == 1:
        # Set direction
        GPIO.output(MotorPin1, GPIO.HIGH)
        GPIO.output(MotorPin2, GPIO.LOW)
        # Enable the motor
        GPIO.output(MotorEnable, GPIO.HIGH)
        print ("Clockwise")
    # Counterclockwise
    if direction == -1:
        # Set direction
```

```
            GPIO.output(MotorPin1, GPIO.LOW)
            GPIO.output(MotorPin2, GPIO.HIGH)
            # Enable the motor
            GPIO.output(MotorEnable, GPIO.HIGH)
            print ("Counterclockwise")
        # Stop
        if direction == 0:
            # Disable the motor
            GPIO.output(MotorEnable, GPIO.LOW)
            print ("Stop")


def main():


    # Define a dictionary to make the script more readable
    # CW as clockwise, CCW as counterclockwise, STOP as stop
    directions = {'CW': 1, 'CCW': -1, 'STOP': 0}
    while True:
        # Clockwise
        motor(directions['CW'])
        time.sleep(5)
        # Stop
        motor(directions['STOP'])
        time.sleep(5)
        # Anticlockwise
        motor(directions['CCW'])
        time.sleep(5)
        # Stop
        motor(directions['STOP'])
        time.sleep(5)
```

```
def destroy():
    # Stop the motor
    GPIO.output(MotorEnable, GPIO.LOW)
    # Release resource
    GPIO.cleanup()

# If run this script directly, do:
if __name__ == '__main__':
    setup()
    try:
        main()
    # When 'Ctrl+C' is pressed, the program
    # destroy() will be executed.
    except KeyboardInterrupt:
        destroy()
```

Code Explanation

```
def motor(direction):
    # Clockwise
    if direction == 1:
        # Set direction
        GPIO.output(MotorPin1, GPIO.HIGH)
        GPIO.output(MotorPin2, GPIO.LOW)
        # Enable the motor
        GPIO.output(MotorEnable, GPIO.HIGH)
        print ("Clockwise")
...
```

Create a function, motor() whose variable is direction. As the condition that direction=1 is met, the motor rotates clockwise; when direction=-1, the motor rotates

anticlockwise; and under the condition that direction=0, it stops rotating.

```
def main():
    # Define a dictionary to make the script more readable
    # CW as clockwise, CCW as counterclockwise, STOP as stop
    directions = {'CW': 1, 'CCW': -1, 'STOP': 0}
    while True:
        # Clockwise
        motor(directions['CW'])
        time.sleep(5)
        # Stop
        motor(directions['STOP'])
        time.sleep(5)
        # Anticlockwise
        motor(directions['CCW'])
        time.sleep(5)
        # Stop
        motor(directions['STOP'])
        time.sleep(5)
```

In the main（）function, create an array, directions[], in which CW is equal to 1, the value of CCW is -1, and the number 0 refers to Stop.

As the code runs, the motor first rotates clockwise for 5s then stop for 5s, after that, it rotates anticlockwise for 5s; subsequently, the motor stops for 5s. This series of actions will be executed repeatedly.

Now, you should see the motor blade rotating.

Phenomenon Picture

21.DC_Motor_Control