16.Servo

Introduction

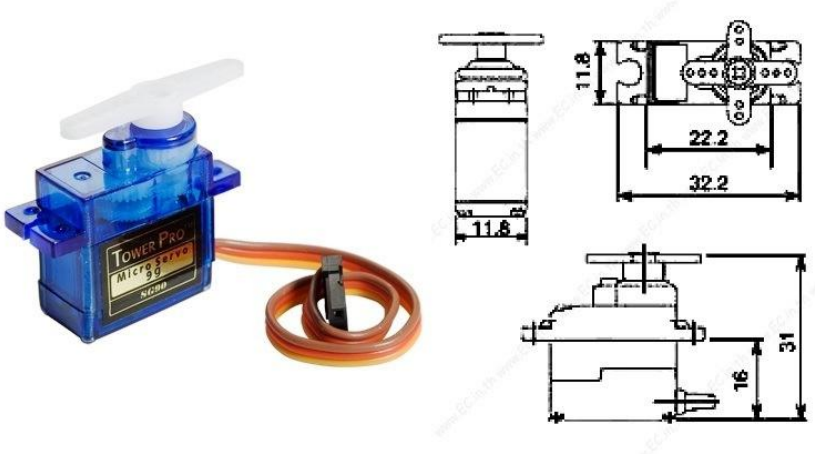
In this lesson, you will learn how to use the Servo. Servo is a type of geared motor that can only rotate 180 degrees.

Hardware Required

- ✓ 1 * Raspberry Pi
- ✓ 1 * T-Extension Board
- ✓ 1 * Servo
- ✓ 1 * 40-pin Cable
- ✓ Several Jumper Wires
- ✓ 1 * Breadboard

Principle

SG90 Servo



Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but SMALLER. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

Specifications:

•Weight: 9 g

•Dimension: 22.2 x 11.8 x 31 mm approx.

•Stall torque: 1.8 kgf·cm

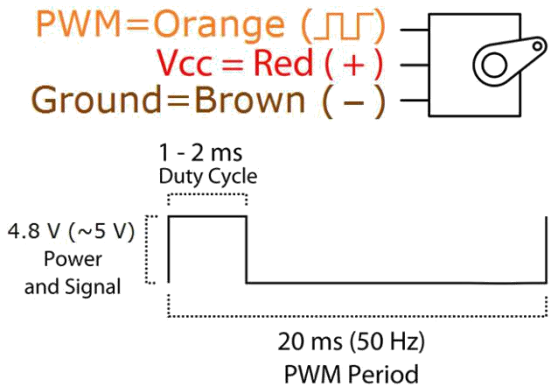
•Operating speed: 0.1 s/60 degree

•Operating voltage: 4.8 V (~5V)

•Dead band width: 10 μs

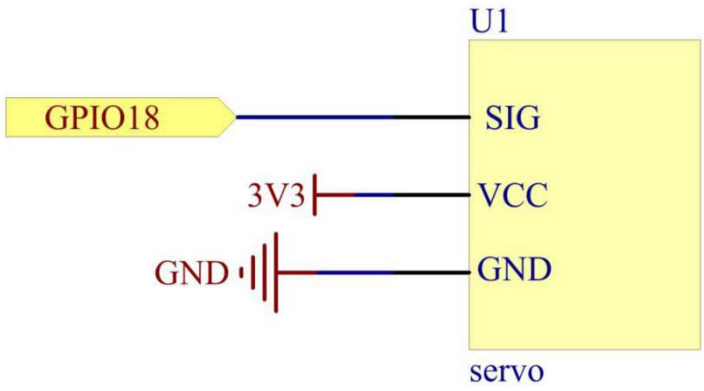
•Temperature range: 0 ºC – 55 ºC

Position "0" (1.5 ms pulse) is mi ddle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.



Schematic Diagram

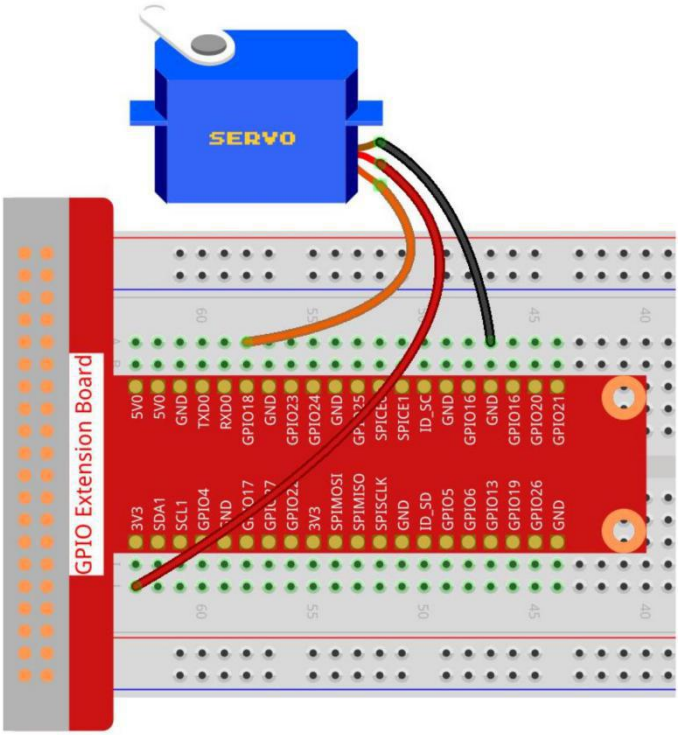
T-Board Name	physical	wiringPi	BCM
GPIO18	Pin 12	1	18



Experimental Procedures

Step 1: Build the circuit.

16.Servo



For C Language Users

Step 2: Get into the folder of the code.

```
cd /home/pi/REXQualis_Raspberry_Pi_Complete_Starter_Kit/C/16.Servo
```

Step 3: Compile the code.

```
gcc 16.Servo.c -o Servo.out -lwiringPi
```

Step 4: Run the executable file above.

```
sudo ./Servo.out
```

After the program is executed, the servo will rotate from 0 degrees to 180 degrees, and then from 180 degrees to 0 degrees, circularly.

Code

```
#include <wiringPi.h>
#include <softPwm.h> //pwm    control libs
#include <stdio.h>


#define servoPin        1               //define the GPIO number connected to servo
```

```
long map(long value,long fromLow,long fromHigh,long toLow,long toHigh){
    return (toHigh-toLow)*(value-fromLow) / (fromHigh-fromLow) + toLow;
}
void servoWrite(int pin, int angle){        //Create a funtion, servoWrite() to control the
rotate angle of the servo.
    if(angle < 0)
        angle = 0;
    if(angle > 180)
        angle = 180;
    softPwmWrite(pin,map(angle,0,180,5,25));
}

int main(void)
{
    int i;

    if(wiringPiSetup() == -1){ //when initialize wiring failed,print message to screen
        printf("setup wiringPi failed !");
        return 1;
    }
    softPwmCreate(servoPin,    0, 200);            //initialize PMW pin of servo
    while(1){
        for(i=0;i<181;i++){    //make servo rotate from minimum angle to
maximum angle
            servoWrite(servoPin,i);
            delay(1);
        }
        delay(500);
```

```
            for(i=181;i>-1;i--){    //make servo rotate from maximum angle to
minimum angle
                servoWrite(servoPin,i);
                delay(1);
            }
            delay(500);
        }
        return 0;
}
```

Code Explanation

```
long map(long value,long fromLow,long fromHigh,long toLow,long toHigh){
        return (toHigh-toLow)*(value-fromLow) / (fromHigh-fromLow) + toLow;
}
```

Create a map() function to map value in the following code.

```
void servoWrite(int pin, int angle){       //Create a funtion, servoWrite() to control the
rotate angle of the servo.
        if(angle < 0)
            angle = 0;
        if(angle > 180)
            angle = 180;
        softPwmWrite(pin,map(angle,0,180,5,25));
}
```

Create a funtion, servoWrite() to write angle to the servo.

```
softPwmWrite(pin,map(angle,0,180,5,25));
```

This function can change the duty cycle of the PWM.

To make the servo rotate to 0 ~ 180 °, the pulse width should change within the range of 0.5ms ~ 2.5ms when the period is 20ms; in the function, softPwmCreate(), we have set that the period is 200x100us=20ms, thus we need to map 0 ~ 180 to 5x100us ~ 25x100us.

```
softPwmCreate(servoPin,  0, 200);
```

The function is to use softwares to create a PWM pin, servoPin, then the initial pulse widths of them are set to 0, and the period of PWM is 200 x100us.

The prototype of this function is shown below.

```
int softPwmCreate（int pin， int initialValue， int pwmRange）;
```

Parameter pin: Any GPIO pin of Raspberry Pi can be set as PWM pin.

Parameter initialValue: The initial pulse width is that initialValue times 100us.

Parameter pwmRange: the period of PWM is that pwmRange times 100us.

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 the code

```
sudo python3 16.Servo.py
```

After the program is executed, the servo will rotate from 0 degrees to 180 degrees, and then from 180 degrees to 0 degrees, circularly.

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 python2


import RPi.GPIO as GPIO
import time


servoPin = 12


def map( value, fromLow, fromHigh, toLow, toHigh):
    return (toHigh-toLow)*(value-fromLow) / (fromHigh-fromLow) + toLow
```

16.Servo

```
def setup():
    global p
    GPIO.setmode(GPIO.BOARD)        # Numbers GPIOs by physical location
    GPIO.setup(servoPin, GPIO.OUT)    # Set servoPin's mode is output
    GPIO.output(servoPin, GPIO.LOW)    # Set servoPin to low

    p = GPIO.PWM(servoPin, 50)        # set Frequecy to 50Hz
    p.start(0)                        # Duty Cycle = 0

def servoWrite(angle):          # make the servo rotate to specific angle (0-180
degrees)
    if(angle<0):
        angle = 0
    elif(angle > 180):
        angle = 180
    p.ChangeDutyCycle(map(angle,0,180,2.5,12.5))#map the angle to duty cycle and
output it

def loop():
    while True:
        for i in range(0, 181, 1):      #make servo rotate from 0 to 180 deg
            servoWrite(i)          # Write to servo
            time.sleep(0.001)
        time.sleep(0.5)
        for i in range(180, -1, -1): #make servo rotate from 180 to 0 deg
            servoWrite(i)
            time.sleep(0.001)
        time.sleep(0.5)
```

```
def destroy():
    p.stop()
    GPIO.cleanup()


if __name__ == '__main__':        #Program start from here
    setup()
    try:
        loop()
    except KeyboardInterrupt:    # When 'Ctrl+C' is pressed, the program destroy() will be    executed.
        destroy()
```

Code Explanation

```
    p = GPIO.PWM(servoPin, 50)        # set Frequecy to 50Hz
    p.start(0)                        # Duty Cycle = 0
```

Set the servoPin to PWM pin, then the frequency to 50hz, and the period to 20ms.

p.start(0): Run the PWM function，and set the initial value to 0.

```
def servoWrite(angle):          # make the servo rotate to specific angle (0-180 degrees)
    if(angle<0):
        angle = 0
    elif(angle > 180):
        angle = 180
    p.ChangeDutyCycle(map(angle,0,180,2.5,12.5))#map the angle to duty cycle and output it
```

Create a function, servoWrite() to write angle that ranges from 0 to 180 into the servo.

```
    p.ChangeDutyCycle(map(angle,0,180,2.5,12.5))
```

This function can change the duty cycle of the PWM.

16.Servo

To render a range 0 ~ 180 ° to the servo, the pulse width of the servo is set to 0.5ms-2.5ms.

In the previous codes, the period of PWM was set to 20ms, thus the duty cycle of PWM is (0.5/20)%-(2.5/20)%, and the range 0 ~ 180 is mapped to 2.5 ~ 12.5.

Phenomenon Picture