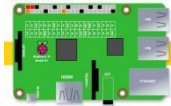
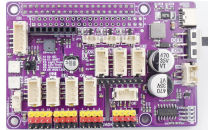



## Lesson 7 How to Control 180° Servo

In this lesson, we will learn how to control 180° Servo.

### 7.1 Components used in this course

Components	Quantity	Picture
Raspberry Pi	1	
Aadept Robot HAT V3.0	1	
180°Servo	1	

### 7.2 Introduction of 180° Servo

What is a servo?

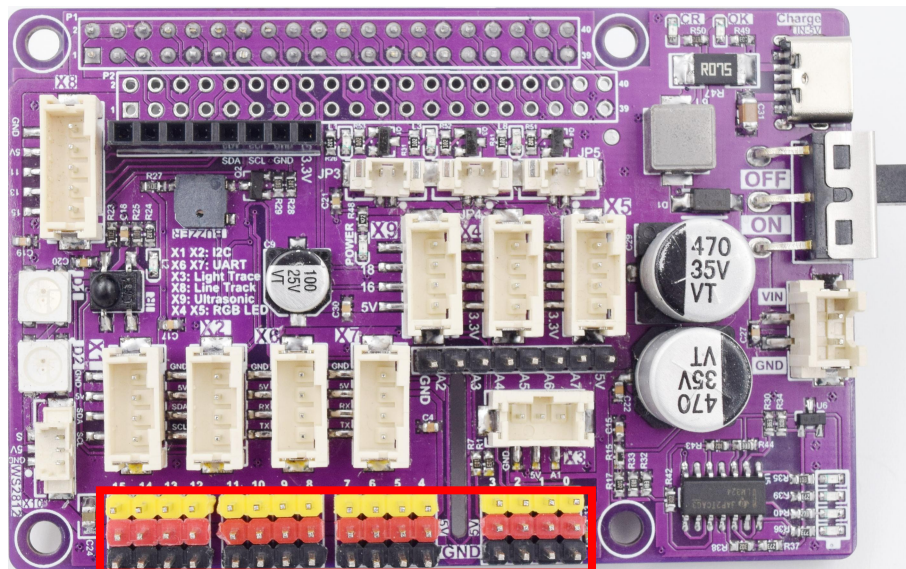
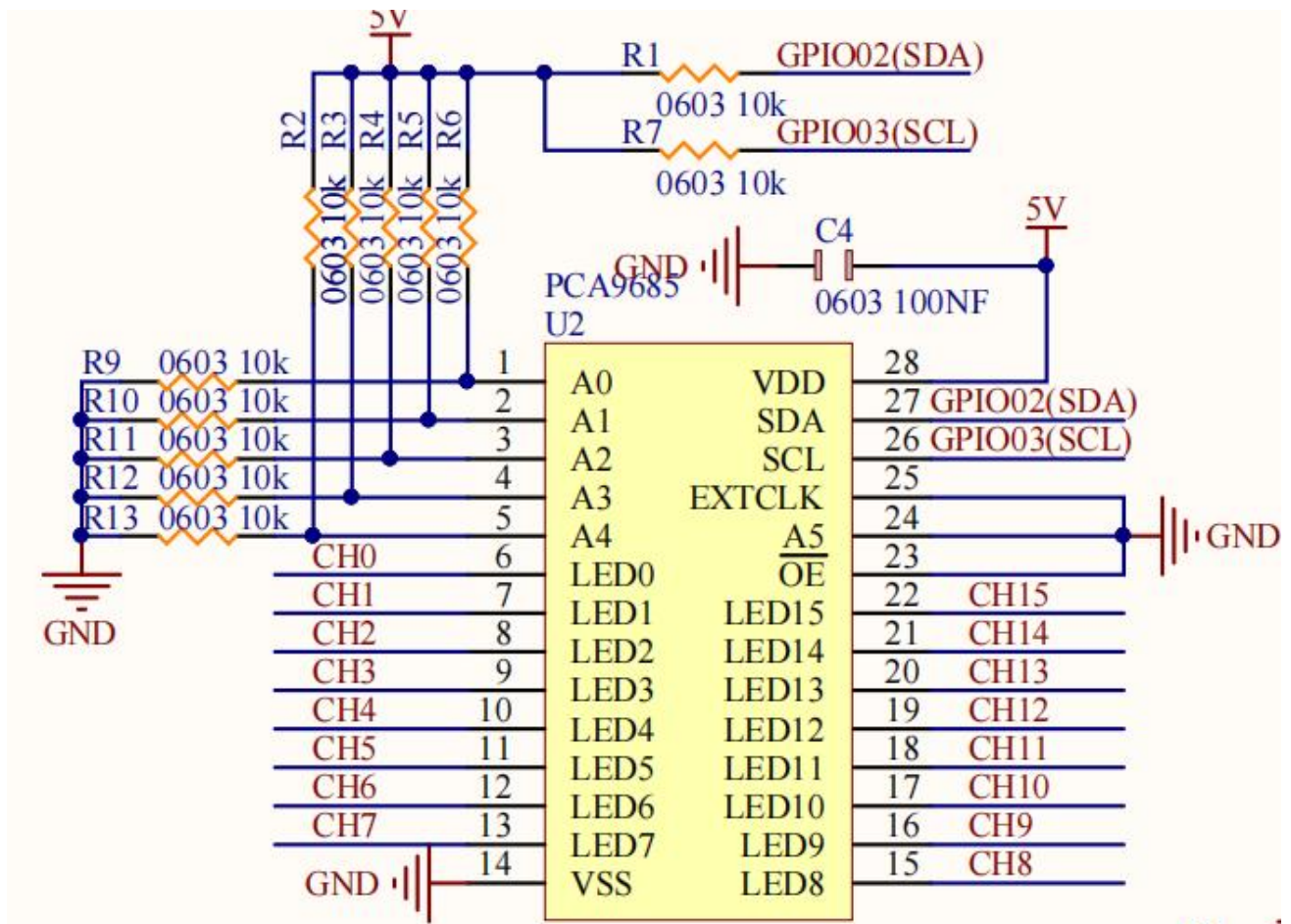
The servo is a position (angle) servo driver, which is suitable for those control systems that require constant angle changes and can be maintained. It has been widely used in high-end remote control toys, such as airplanes, submarine models, and remote control robots.

We use a 180° servo in this lesson, which can move between 0° and 180°. Since the 180° servo can use the PWM signal to control the rotation angle of a certain mechanism, it is a more commonly used module in robot products.

On the Raspberry Pi driver board Adept Robot HAT V3.0, there is a PCA9685 chip specially used to control the servo. The Raspberry Pi uses I2C to communicate with the PCA9685. It controls the servo by sending pulse signals from the microcontroller. These pulses tell the servo mechanism of the servo where to move. The picture of the 180° servo is as follows:



The PCA9685 chip is connected to the I2C interface of the Raspberry Pi and can control the 16-pin signals extended by the PCA9685 module through the I2C address. The I2C address of Adept Robot HAT V3.0 is **0x5f**. You can enter "**i2cdetect -y 1**" in the Raspberry Pi command line to see all i2C addresses on the Raspberry Pi.



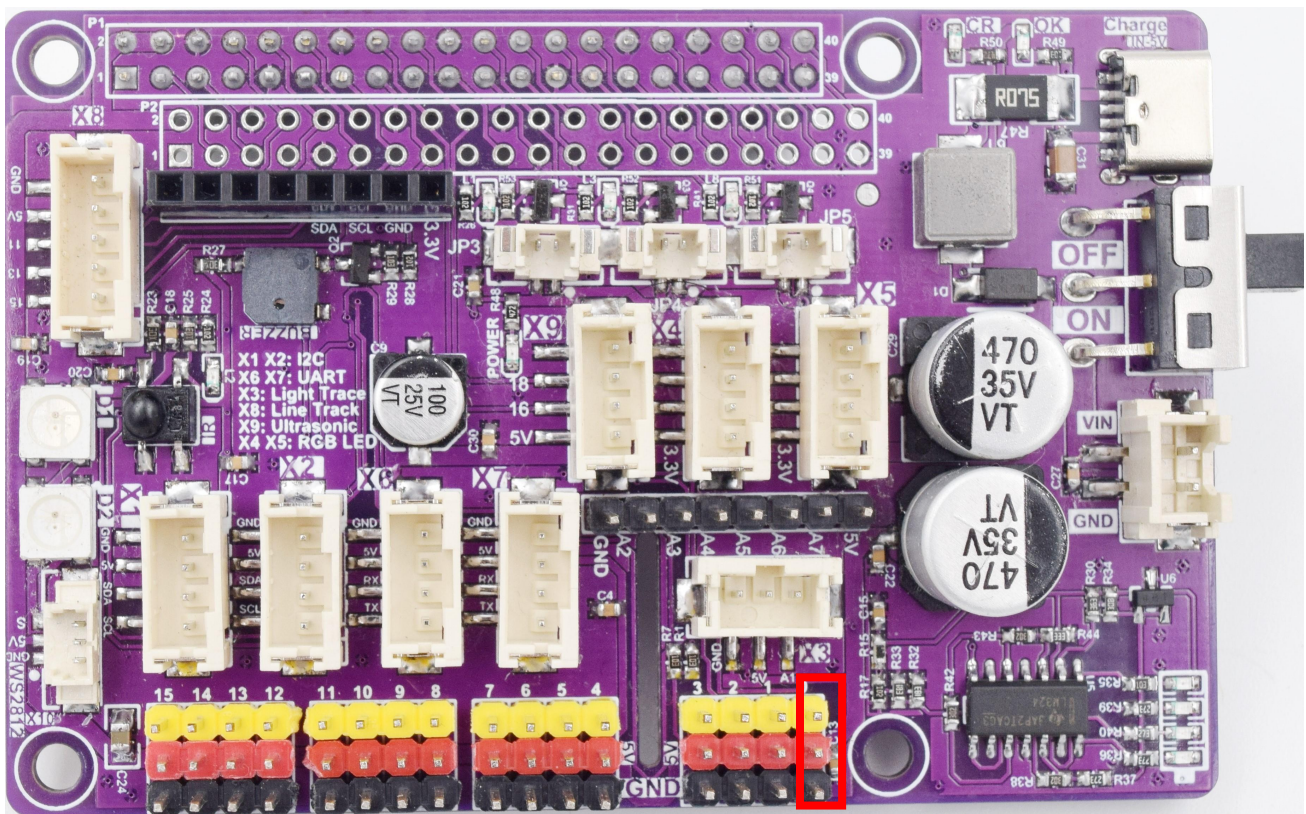
**Note:** Some of the servo pins and motors of the PCA9685 extension are shared.



The extended pins 0-7 of PCA9685 are only used for servos. The extended pins 8-15 of PCA9685 are used by both the motor and the servo. If you need to use the motor while using the servo, it is recommended to use pins 0-7 as the servo control pins and pins 8-15 as the motor control pins. If you don't need to use a motor, you can use pins 0-15 to control the servo.

### 7.3 Wiring diagram (Circuit diagram)

When the 180° Servo module is in use, it needs to be connected to the servo interface on the Robot HAT V3.0 expansion board. The yellow wire is connected to the yellow pin, the red wire is connected to the red pin, and the brown wire is connected to black pin, as shown below (connect to pin 0):



## 7.4 How to control 180°Servo

### Run the code

1. Remotely log in to the Raspberry Pi terminal.

```
Linux raspberrypi 4.19.118-v7l+ #1311 SMP Mon Apr 27 14:26:42 BST 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Aug 29 08:17:49 2020 from 192.168.3.208

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~ $
```

2. See the occupied I2C address.

```
i2cdetect -y 1
```

```
pi@raspberrypi:~ $
pi@raspberrypi:~ $ i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  48  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  5f  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
pi@raspberrypi:~ $
```

3. Enter the command and press Enter to enter the folder where the program is located:

```
cd aadept_rasptank2/examples/
```

```
pi@raspberrypi:~ $  
pi@raspberrypi:~ $ cd adeept_rasptank2/examples/  
pi@raspberrypi:~/adeept_rasptank2/examples $  
pi@raspberrypi:~/adeept_rasptank2/examples $
```

4. View the contents of the current directory file:

```
ls
```

```
pi@raspberrypi:~/adeept_rasptank2/examples $ ls  
01_LED.py      03_servo.py   05_ws2812.py  07_lineTracking.py  
02_buzzer.py   04_motor.py   06_ultra.py  
pi@raspberrypi:~/adeept_rasptank2/examples $
```

5. Enter the command and press Enter to run the program:

```
sudo python3 03_servo.py
```

```
pi@raspberrypi:~/adeept_rasptank2/examples $  
pi@raspberrypi:~/adeept_rasptank2/examples $ sudo python3 03_servo.py  
█
```

6. After running the program successfully, you will observe that the servo will rotate regularly.
7. When you want to terminate the running program, you can press the shortcut key "**Ctrl + C**" on the keyboard.

## 7.5 The main code program

Complete code refer to [03\\_servo.py](#).

```
1. #!/usr/bin/env/python3  
2. import time  
3. from board import SCL, SDA  
4. import busio  
5. from adafruit_motor import servo  
6. from adafruit_pca9685 import PCA9685  
7.  
8. i2c = busio.I2C(SCL, SDA)  
9. # Create a simple PCA9685 class instance.
```

```
10. pca = PCA9685(i2c, address=0x5f) #default 0x40
11.
12. pca.frequency = 50
13.
14. # servo7 = servo.Servo(pca.channels[7], min_pulse=580, max_pulse=2350)
15. # servo7 = servo.Servo(pca.channels[7], min_pulse=500, max_pulse=2600)
16. # servo7 = servo.Servo(pca.channels[7], min_pulse=400, max_pulse=2400)
17. # servo7 = servo.Servo(pca.channels[7], min_pulse=600, max_pulse=2500)
18. # servo7 = servo.Servo(pca.channels[7], min_pulse=500, max_pulse=2400)
19. # The pulse range is 750 - 2250 by default. This range typically gives 135 degrees of
20. # range, but the default is to use 180 degrees. You can specify the expected range if you wish:
21. # servo7 = servo.Servo(pca.channels[7], actuation_range=135)
22. def set_angle(ID, angle):
23.     servo_angle = servo.Servo(pca.channels[ID], min_pulse=500, max_pulse=2400, actuation_range=180)
24.     servo_angle.angle = angle
25. ....
26. # You can also specify the movement fractionally.
27. fraction = 0.0
28. while fraction < 1.0:
29.     servo7.fraction = fraction
30.     fraction += 0.01
31.     time.sleep(0.03)
32. pca.deinit()
33. '''
34.
35. def test(channel):
36.     for i in range(180): # The servo turns from 0 to 180 degrees.
37.         set_angle(channel, i)
38.         time.sleep(0.01)
39.     time.sleep(0.5)
40.     for i in range(180): # The servo turns from 180 to 0 degrees.
41.         set_angle(channel, 180-i)
42.         time.sleep(0.01)
43.     time.sleep(0.5)
44.
45. if __name__ == "__main__":
46.     channel = 1
47.     while True:
48.         test(channel)
```