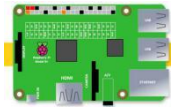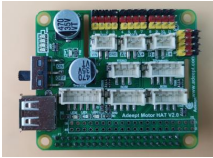# Lesson 5 How to Control 180° Servo

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

5.1 Components used in this course

| Components	Quantity	Picture
Raspberry Pi	1	
Motor HAT	1	
180°Servo	1	

5.2 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 Robot HAT, 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:
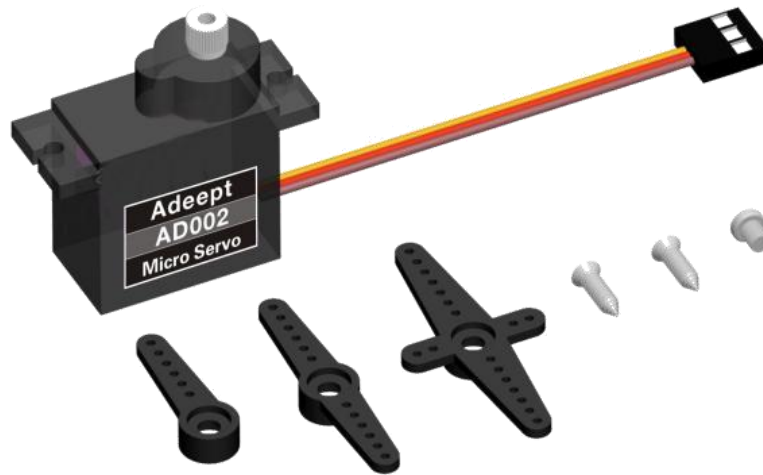
5.3 Wiring diagram (Circuit diagram)

When the 180°Servo module is in use, it needs to be connected to the servo interface on the RobotHAT driver 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):
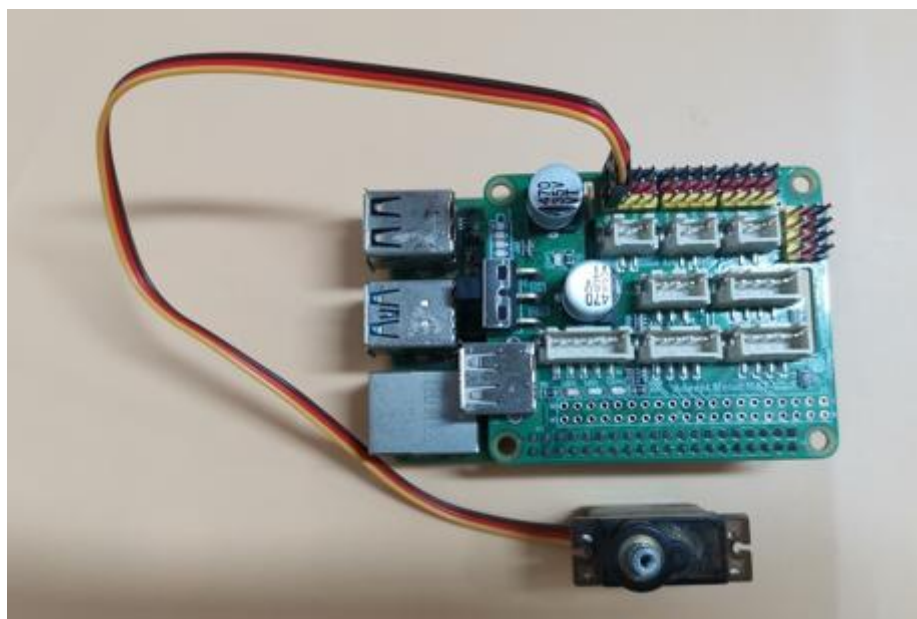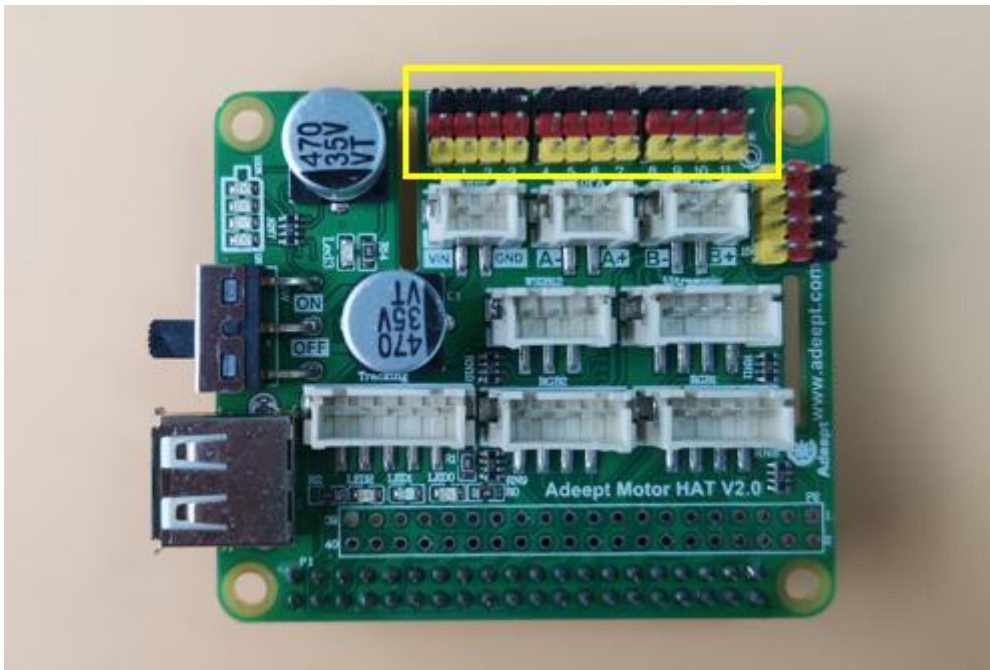
5.4 How to control 180°Servo

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

cd adeept_rasptank/server/

```
pi@raspberrypi:~ $
pi@raspberrypi:~ $ cd adeept_rasptank/server/
pi@raspberrypi:~/adeept_rasptank/server $ ▮
```

3. View the contents of the current directory file:

ls

```
pi@raspberrypi:~/adeept_rasptank/server $
pi@raspberrypi:~/adeept_rasptank/server $ ls
app.py              FPV.py            move.py            server.py
appserverAP.py      FPVtest.py        OLED.py            serverTest.py
appserver.py        functions.py      PID.py             servo.py
base_camera.py      info.py           __pycache__        switch.py
camera_opencv.py    Instruction.txt   raspi-config.py    ultra.py
config.txt          Kalman_filter.py  requirements.txt   webServer.py
dist                LEDapp.py         robotLight.py
findline.py         LED.py            RPIservo.py
pi@raspberrypi:~/adeept_rasptank/server $ ▮
```

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

sudo python3 servo.py

```
pi@raspberrypi:~/adeept_rasptankpro/server $
pi@raspberrypi:~/adeept_rasptankpro/server $ sudo python3 servo.py
▮
```

5. After running the program successfully, you will observe that the servo will rotate regularly.

6. When you want to terminate the running program, you can press the shortcut key "Ctrl + C" on the keyboard.

5.5 The main code program

Complete code refer to servo.py or RPIservo.py .

Control the servo to rotate to a certain angle

```
1.  import Adafruit_PCA9685 # Import the library used to communicate with PCA9685
2.  import time
3.
4.  pwm = Adafruit_PCA9685.PCA9685()    # Instantiate the object used to control the PWM
5.  pwm.set_pwm_freq(50)    # Set the frequency of the PWM signal
6.
7.  while 1：    # Make the servo connected to the No. 3 servo port on the RobotHAT drive board reciprocate
8.      pwm.set_pwm(0, 0, 300)
9.      time.sleep(1)
10.     pwm.set_pwm(0, 0, 400)
11.     time.sleep(1)
```

In the above code, set_pwm_freq(50) is used to set the PWM frequency to 50Hz. This setting depends on the model of the servo. The servo used by our robot products needs to be controlled by a 50Hz PWM signal. If you use For other servos, this value needs to be set by referring to the specific servo documentation.

Pwm.set_pwm(0, 0, 300) This method is used to control the rotation of a servo to a certain position, where 3 is the port number of the servo, which corresponds to the number marked on the RobotHAT driver board, but pay attention When the servo is connected to the drive board, do not insert the ground wire, VCC and signal wire in the reverse direction, brown to black, red to red, and yellow to yellow; 0 is the deviation value for controlling the rotation of the servo, which is not used in our program This function is used to correct the deviation (the cause of the error of the servo can

be referred to the precautions for structural assembly); 300 is the value of the PWM duty cycle you want to set. Depending on the servo, this value represents a different servo angle. The PWM duty cycle range of the servo we use is about 100 to 560, which corresponds to a rotation range of about 0° to 180°.

Using the above code to control the servo does not control the rotation speed of the servo. If we want a servo to slowly swing back and forth between two positions, we need to use the method of increasing or decreasing variables to control the servo.

Control the servo to move slowly

```
1.  import Adafruit_PCA9685
2.  import time
```

Import dependent libraries.

```
1.  pwm = Adafruit_PCA9685.PCA9685()
2.  pwm.set_pwm_freq(50)
```

Set_pwm_freq(50) is used to set the PWM frequency to 50Hz. This setting depends on the model of the servo. The servos used in our robot products need to be controlled by a 50Hz PWM signal. If you use other servos, this A value needs to be set by referring to the specific servo document.

```
1.  while 1:
2.      for i in range(0, 100):
3.          pwm.sety_pwm(0, 0, (300+i))
4.          time.sleep(0.05)
5.      for i in range(0,100):
6.          pwm.sety_pwm(0, 0, (400-i))
7.          time.sleep(0.05)
```

pwm.set_pwm(0, 0, 300+i) This method is used to control the rotation of a servo to a certain position, where 3 is the port number of the servo, which corresponds to the number marked on the Robot HAT drive board, but pay attention to the direction of the ground wire, VCC and signal wire when the servo is connected to the drive board. Don't plug in the opposite direction. Brown to black, red to red, and yellow to yellow; 0 is the deviation value that controls the rotation of the servo, 300+i

is the PWM duty cycle value to be set depends on the different servos. This value represents the angle of the servo. The PWM duty cycle range of the servos we use is about 100 to 560, corresponding to about 0° to 180° Rotation range.

By the formula: (duty ratio -100)/2.55, the angle corresponding to the current duty ratio can be calculated.

Using the above code can make the servo rotate slowly back and forth between 300 and 400, but this method of controlling the servo also has great drawbacks. When the program is executed to the slow motion part of the servo, it will be blocked, which will seriously affect the program. Therefore, a multi-threaded solution is provided in our robot product program to solve this problem.

Non-blocking control method

You can find the RPIservo.py file in the server folder of the robot product, copy it to the same folder as the program you want to run, and then you can use this method in your program.

```python
1.  import RPIservo #  Import a library that uses multiple threads to control the servo
2.  import time
3.
4.  sc = RPIservo.ServoCtrl()   # Instantiate the object that controls the servo
5.  sc.start()  # Start this thread, when the servo does not move, the thread is suspended
6.
7.  while 1:
8.      sc.singleServo(3, -1, 2)
9.      time.sleep(1)
10.     sc.stopWiggle()
11.
12.     sc.singleServo(3, 1, 2)
13.     time.sleep(1)
14.     sc.stopWiggle()
```

Use the above code to control the servo to reciprocate, except for time.sleep(), it will not block the running of the context program.

Call singleServo() to start motion, call stopWiggle() to stop motion, singleServo() requires three parameters, which are the port number of the servo to be controlled (3 is to control the No. 3 servo connected to RobotHAT) , The direction of the servo (1 or -1), and the speed of the servo (the larger the value, the faster the movement).

Note:

- When the load of the servo is too large, it will cause the servo to heat up obviously and even cause the device to restart. If the servo is blocked, please cut off the power immediately to avoid damage to the equipment or even fire.

- Possible reasons for the servo stall:

    1. Whether the power supply voltage of the device is too low (check whether the battery is fully charged, if it is powered by USB, the voltage is low, and sometimes the servo cannot be driven, and a fully charged battery can be used instead).

    2. Whether the servo is damaged (it is determined that the servo is damaged if the driving servo cannot rotate under the condition of no load).