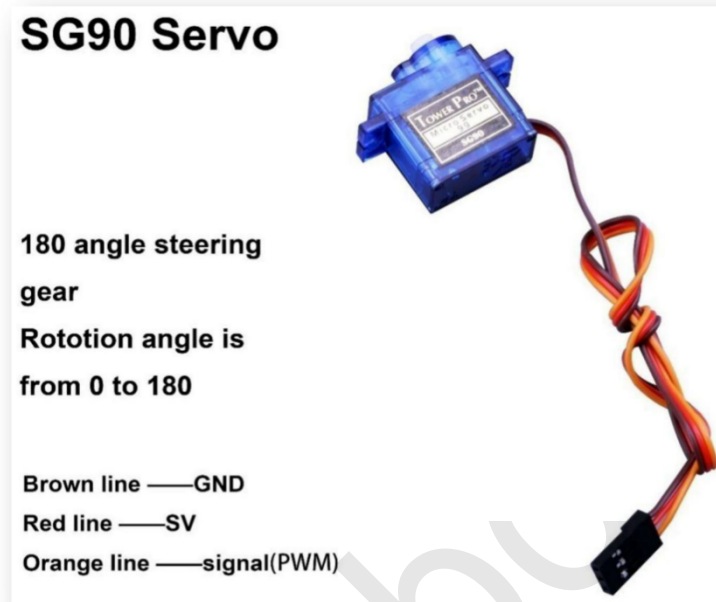


## 1.About MG90S servo

First, let's learn about SG90 servo:



### Classification: 180° servo

Generally, the servo has three control lines: power line, ground line and signal line.

Servo pin definition: brown line - GND, red line - 5V, orange line – signal.

### How the servo works:

The signal modulation chip in the servo receives the signal from the control board, and then the servo will obtain the basic DC voltage. There is also a reference circuit inside the servo, which can generate standard voltage. The two voltages will be compared with each other and output the difference. Then the motor chip will receive the difference and determine the speed, direction and angle. When there is no difference between the two voltages, the servo will stop.

### How to control servo:

To control the servo rotation, it is necessary to make the time pulse about 20ms and the high level pulse width about 0.5ms~2.5ms, which is consistent with the servo angle limit.

Taking 180 degree servo as an example, the corresponding control pulse time is as follows

0.5ms	0
1.0ms	45
1.5ms	90
2.0ms	135
2,5ms	180

## 2.Example

Open the tutorial provided by us, and open the Servo\_Test.ino file under "03\_Tutorial\_&\_Code → Lesson1 Drives a Single Servo → Servo\_Test". Connect the ESP32 development board and computer with USB cable, select the correct development board, processor and port. Download the code into the ESP32, as shown in the figure below.

```

Servo_Test | Arduino 1.8.19
File Edit Sketch Tools Help
Servo_Test
36 * if you are particular, adjust the min and max values to match your needs.
37 */
38
39 #include <ESP32Servo.h>
40
41 Servo myservo; // create servo object to control a servo
42 // 16 servo objects can be created on the ESP32
43
44 int pos = 0; // variable to store the servo position
45 // Recommended PWM GPIO pins on the ESP32 include 2,4,12-19,21-23,25-27,32-33
46 // Possible PWM GPIO pins on the ESP32-S2: 0(used by on-board button),1-17,18(used by on-board LED),19-2.
47 #if defined(ARDUINO_ESP32S2_DEV)
48 int servoPin = 17;
49 #else
50 int servoPin = 18;
51 #endif
52
53 void setup()
54 {
55     // Allow allocation of all timers
56     ESP32PWM::allocateTimer(0);
57     ESP32PWM::allocateTimer(1);
58     ESP32PWM::allocateTimer(2);
59     ESP32PWM::allocateTimer(3);
60     myservo.setPeriodHertz(50); // standard 50 hz servo
61     myservo.attach(servoPin, 1000, 2000); // attaches the servo on pin 18 to the servo object
62     // using default min/max of 1000us and 2000us
63     // different servos may require different min/max settings
64     // for an accurate 0 to 180 sweep
65 }
66
67 void loop()
68 {
69     for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees

```

Done uploading.

Leaving...

Hard resetting via RTS pin...

ESP32 Dev Module, Disabled, Default 4MB with spiiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, None on COM5

After the code is downloaded, pull out the USB cable and connect the MG90S servo and expansion board. The connection relationship is shown below.

MG90S servo	servo shield
Brown line	-----G
Red line	-----V
Orange line	-----18

Insert the battery into the expansion board. If the servo is normal, it should rotate from 0 degrees to 180 degrees, and then rotate from 180 degrees to 0 degrees, and keep cycling. Disassemble the 12 servos provided in the kit and test them according to the same operation to ensure that each servo is normal.。