

First, burn the calibration program into the microcontroller (when the robot is not turned on),

```
calibration | Arduino 1.8.19
文件 编辑 项目 工具 帮助

calibration
// Locate the initial position of legs
// Register 2015-09-09

#include <ESP32Servo.h>

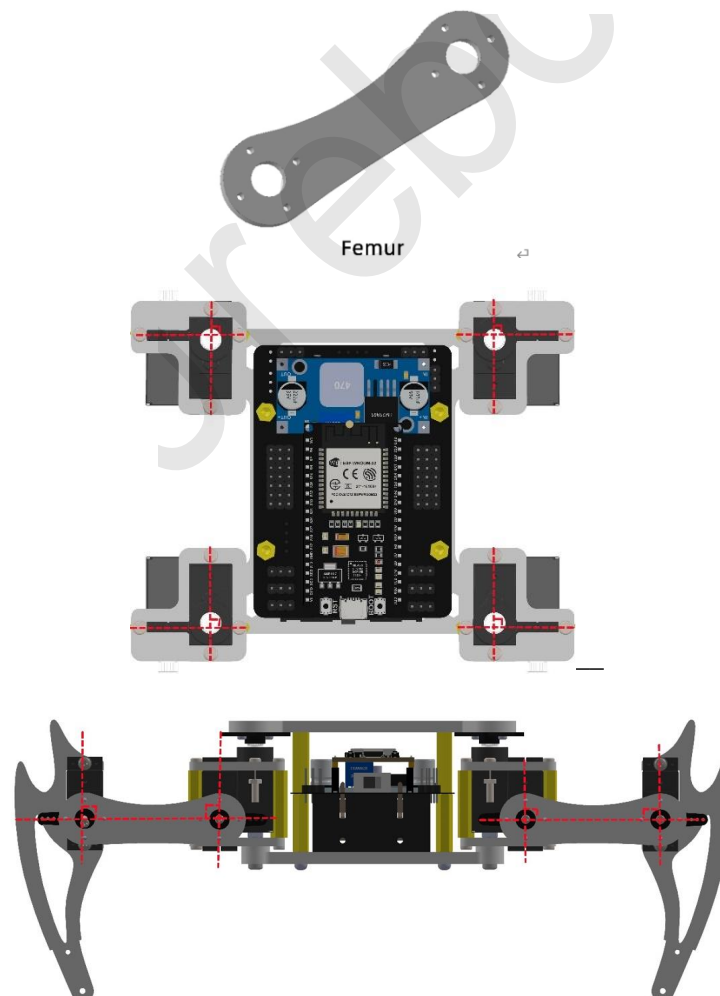
Servo servo[12];

//define servos' ports
const int servo_pin[12] = {18, 5, 19, 2, 4, 15, 33, 25, 32, 26, 14, 13};
const int offset[4][3] = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}};

void setup()
{
  Serial.begin(9600);
  //Initialize all servos
  for (int i = 0; i < 12; i++)
  {
    servo[i].attach(servo_pin[i], 500, 2500);
    delay(20);
  }
  for (int i = 0; i < 12; i++)
  {
    servo[i].write(90);
    delay(20);
  }
}

void loop(void)
{
  while(Serial.available() > 0)
  {
    char command = Serial.read();
    // SERVO Set command
    if (command == 'S') { command == 's' } // 'S' is Servo Command "S,Servo_no,Angle"
    {
      Serial.print(command);
      Serial.print(",");
      int servoNo = Serial.parseInt();
      Serial.print(servoNo);
      Serial.print(",");
      int servoAngle = Serial.parseInt();
      Serial.print(servoAngle);
      Serial.println();
    }
    // if (0 <= servoNo && servoNo < MAX_SERVO_NUM+1)
    {
      // Single Servo Move
    }
  }
}
```

After turning on the machine, when you hear the servo stop rotating, turn off the machine. After turning off the machine, install the structure of the connecting parts between the two servos as shown in the following figure



After assembly into parallel, open the calibration code (USB connection between computer and robot) (robot startup status)

```
calibration | Arduino 1.8.19
文件 编辑 项目 工具 帮助

calibration
// Locate the initial position of legs
// Begin: 2018-09-09

#include <ESP32Servo.h>

Servo servo[12];

//define servos' ports
const int servo_pin[12] = {18, 5, 19, 2, 4, 15, 33, 25, 32, 26, 14, 13};
const int offset[4][3] = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}};

void setup()
{
  Serial.begin(9600);
  //initialize all servos
  for (int i = 0; i < 12; i++)
  {
    servo[i].attach(servo_pin[i], 500, 2500);
    delay(20);
  }
  for (int i = 0; i < 12; i++)
  {
    servo[i].write(90);
    delay(20);
  }
}

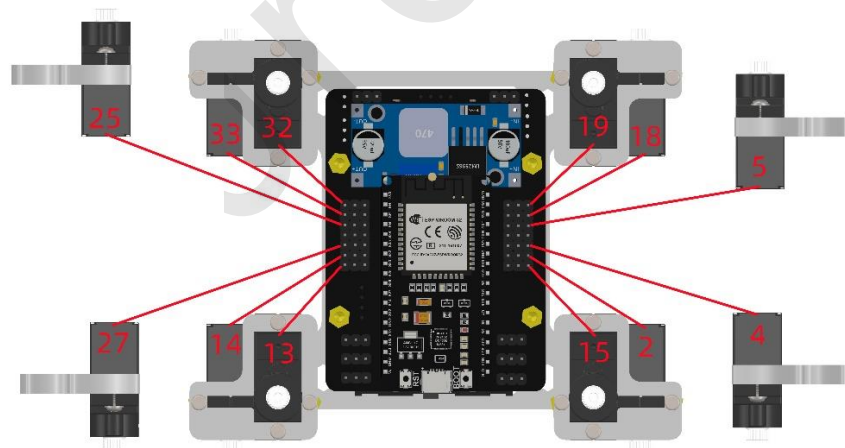
void loop(void)
{
  while(Serial.available() > 0)
  {
    char command = Serial.read();
    // SERVO set command
    if (command == 's' || command == 'S') // 's' is Servo Command "S,Servo_no,Angle"
    {
      Serial.print(command);
      Serial.print(",");
      int servoNo = Serial.parseInt();
      Serial.print(servoNo);
      Serial.print(",");
      int servoAngle = Serial.parseInt();
      Serial.print(servoAngle);
      Serial.println();
    }
    // if (0 <= servoNo && servoNo < MAX_SERVO_NUM+1)
    {
      // Single Servo Move
    }
  }
}
```

In the code, we can see two arrays

12 numbers correspond to the calibration values of 12 servos respectively

```
//define servos' ports
const int servo_pin[12] = {18, 5, 19, 2, 4, 15, 33, 25, 32, 27, 14, 13};
const int offset[4][3] = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}};
```

The first array corresponds to the twelve ports plugged into the servo on our expansion board (with numbers written next to the expansion board ports)



The second array is what we need to calibrate next



calibration

```
// Locate the initial position of legs
// RegisHsu 2015-09-09

#include <ESP32Servo.h>

Servo servo[12];

//define servos' ports
const int servo_pin[12] = {18, 5, 19, 2, 4, 15, 33, 25, 32, 26, 14, 13};
const int offset[4][3] = {{ 0, 0, 0}, { 0, 0, 0}, { 0, 0, 0}, { 0, 0, 0}};
```

1.

```
void loop(void)
{
  while(Serial.available() > 0)
  {
    char command = Serial.read();
    // SERVO Set command
    if ( command == 'S' || command == 's' ) // 'S' is Servo Command "S,Servo_no,Angle"
```

1. 首先我们定义一个 "S"

```
    {
      Serial.print(command);
      Serial.print(',');
      int servoNo = Serial.parseInt();
      Serial.print(servoNo);
      Serial.print(',');
      int servoAngle = Serial.parseInt();
      Serial.print(servoAngle);
      Serial.println();
```

2. 输入的值保存在command里然后逗号隔开

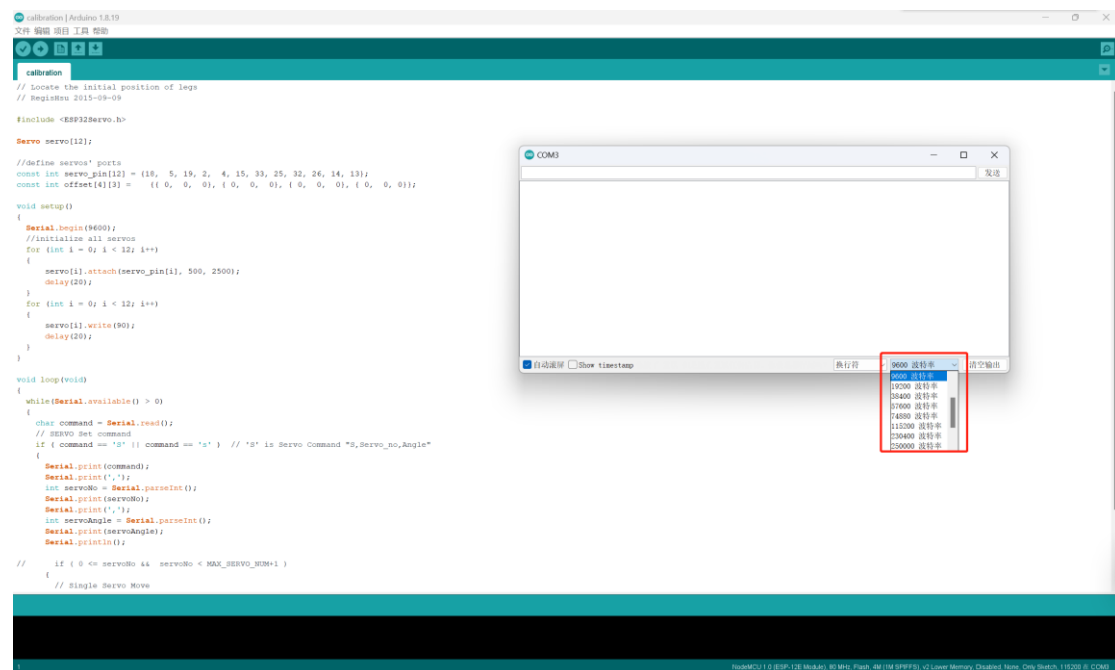
```
    }
    if ( 0 <= servoNo && servoNo < MAX_SERVO_NUM+1 )
    {
      // Single Servo Move
      servo[servoNo].write(90+servoAngle);

      // Wait Servo Move
      delay(300); // 180/60 (°/100msec) =300(msec)
    }
    // SERVO Set END
  }
}
```

3. Next, open the serial port monitor



4. Adjust the baud rate to 9600



5. After adjusting to 9600, enter s, 0,10 on it

(The 0 between s, 0, 10 represents port 18 in the first string array, input 1 represents port 5, and so on)

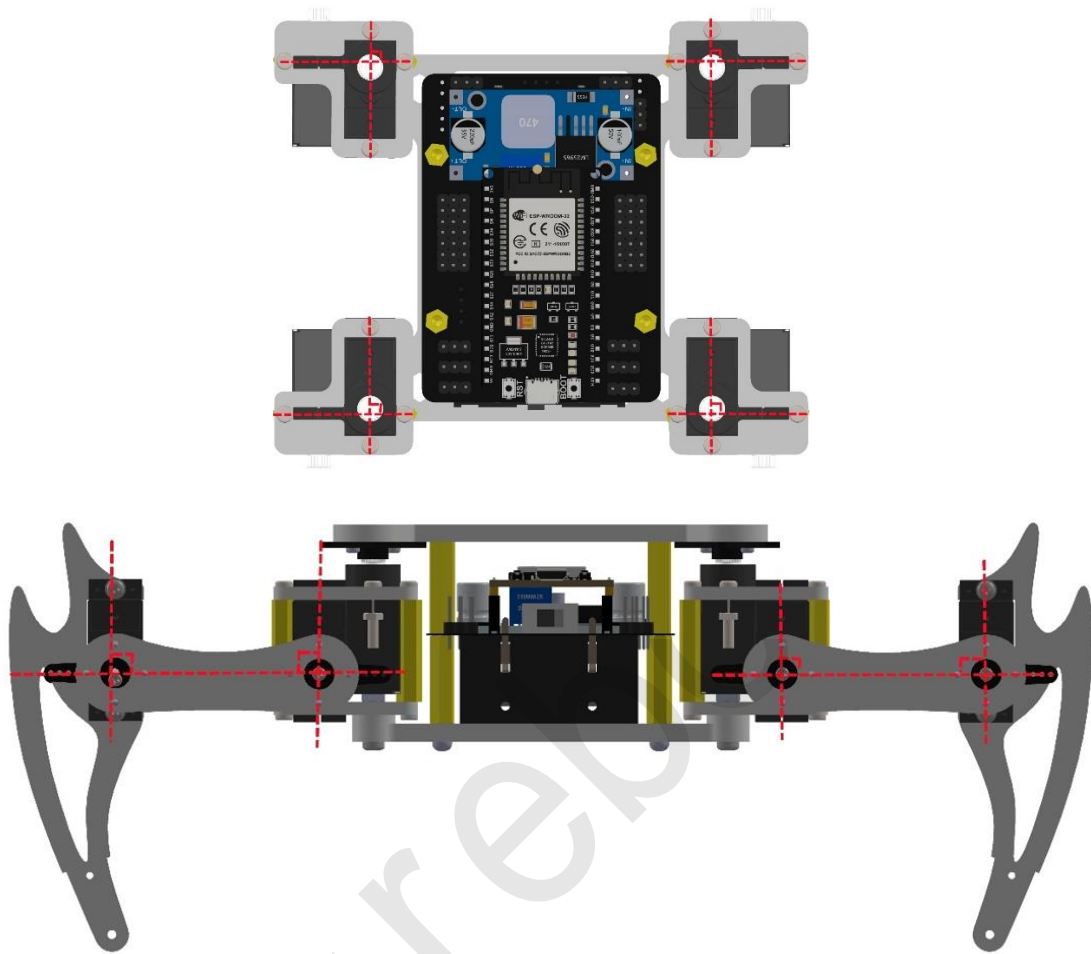
(The 10 after s, 0, and 10 represents the adjusted angle, and generally the input value is between 20 and -20)

(Enter s, 0, 10, and then click send. You will notice that the servo motor inserted in port 18 has rotated a bit, and then slowly adjust the angle (keeping parallel) by inputting the correct number again, and then record the final number.)

For example, the final number recorded below s, 0, xx is -5, s, 1, xx, and the final parallel value is -10 (the values for each device are different, and the pictures are for reference only)



After recording the twelve numbers, the spider's current state is that all four feet are parallel



Then download the mobile app control program or webpage control program

(When downloading the program, it is necessary to maintain the shutdown state, otherwise the servo will rotate randomly.)

(Use the same method to manipulate the downloaded program)

- Lesson0 Setting Development Environment
- Lesson1 Drives a Single Servo
- Lesson2 Setup_& Calibration
- Lesson3 Automatically runs the specified action
- Lesson4 Wifi Control**
- Lesson5 Automatic obstacle avoidance
- Lesson6 Mobile App Control Robot**

Then replace the twelve numbers previously recorded with the numbers in the second string array below



QuadBot_T_ESPA

config.h

```
#include <BluetoothSerial.h>
#include "EEPROM.h"

#define Addr 0x1C
#define SDA 18
#define SCL 19

#define EEPROM_SIZE 64

/* Servos -----*/
//define 12 servos for 4 legs
Servo servo[4][3];
//define servos' ports
const int servo_pin[4][3] = {{18, 5, 19}, { 2, 4, 15}, {33, 25, 32}, {26, 14, 13}};
const int offs[4][3] = {{ -5, -10, 10}, { 5, 10, -15}, { 5, 20, -15}, { 0, -5, 15}};
int offset[4][3];
//const int offset[4][3] = {{ 0, 0, 0}, { 0, 0, 0}, { 0, 0, 0}, { 0, 0, 0}};
/* Size of the robot -----*/
const int width = 55;
```

And then burn the control program into the robot