



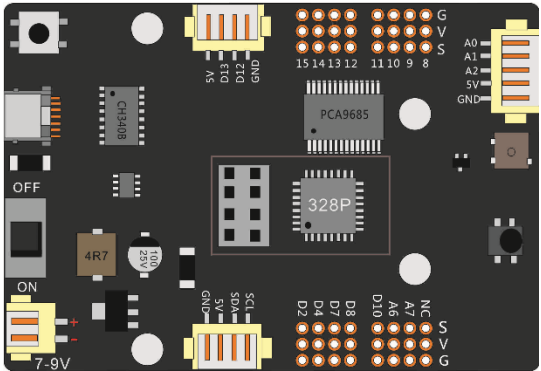
Lesson 4 Test and adjust the angle of the Servo

Table

1. What do you need to prepare.....	1
2. Introduction of the servo	2
3. Wiring	6
4. Upload the code and test	6
5. Code.....	12
6. Adjust the servo to 65 °before assembly	13
7. Any questions and suggestions are welcome	14

1. What do you need to prepare

Components	Quantity	Picture
USB cable	1	
SG90 Servo	1	

Control board	1	
---------------	---	--

2. Introduction of the servo

2.2.1 Servo Physical map

As shown in the figure below, a single SG90 Micro Servo consists of three parts: servo, servo plate, and screws



2.2.2 Pins



2.2.3 Parameters

2.2.3.1 Electrical parameters

No.	(Item)	5V
2.2.3.1.1	No-load current	120±20mA
2.2.3.1.2	load current	160±20mA
2.2.3.1.3	No-load speed	0.11sec/60 °
2.2.3.1.4	Quiescent Current	10mA
2.2.3.1.5	No-load life	50000 times
2.2.3.1.6	stall torque	1.5Kg.-cm
2.2.3.1.7	Stall current	≦ 1A
2.2.3.1.8	Temperature drift (ambient temperature 25 ℃)	≦ 1 °
2.2.3.1.9	temperature resistance	-20 ℃

2.2.3.2 Structural parameters

No.	(Item)	Specification
2.2.3.2.1	Physical dimension	32.6*12.1*22.5mm
2.2.3.2.2	Weight	12G
2.2.3.2.3	Mechanical limit angle	360 °
2.2.3.2.4	Gear type	Grade 5 Metal Gear Set
2.2.3.2.5	Output tooth spline	20T
2.2.3.2.6	Wire length	250 ±5mm
2.2.3.2.7	Gear virtual position	≦ 2 °
2.2.3.2.8	Wire	50C/0.08 OD1.0 JR Connector
2.2.3.2.9	Swing arm	One Word Rocker Cross rocker Flower rocker
2.2.3.2.10	Motor	DC motor
2.2.3.2.11	Shell material	PC

2.2.3.3 Control parameter

No.	(Item)	Specification
2.2.3.3.1	Control signal	PWM cycle 50HZ
2.2.3.3.2	Pulse width range	1000us-2000us
2.2.3.3.3	Amplifier type	Number
2.2.3.3.4	Rotation angle	90 °±3 ℃(when 1000~2000usec)
2.2.3.3.5	Midpoint	1500usec
2.2.3.3.6	Dead zone width	≦ 5usec
2.2.3.3.7	Direction of rotation	Clockwise(when 1000~2000usec)
2.2.3.3.8	Return error	≦ 1 °
2.2.3.3.9	Over-operating angle range	180 ℃(500-2500usec)
2.2.3.3.10	Angle error on both sides	≦ 5 °

2.2.4 Working Principle

The control signal of the servo is a PWM signal with a period of 20ms, in which the pulse width is from 0.5ms-2.5ms, and the corresponding position of the servo is 0-180 degrees, which changes linearly. Provide it with a certain pulse width, and its output shaft will remaster at a corresponding angle, no matter how the external torque changes, until a new pulse signal of different width is provided to it, it will change the output angle to the new corresponding position. There is a reference voltage inside the servo, which generates a reference signal with a period of 20ms and a width of 1.5ms. There is a comparator that compares the applied signal with the reference signal to determine the direction and size, thereby generating the rotation signal of the motor. The internal control circuit board of the servo receives the control signal from the signal line, and controls the rotation of the motor. The motor drives a series of gear sets, which are driven to the output steering wheel after deceleration. The output shaft of the servo is connected to the position feedback potentiometer. When the steering wheel rotates, it drives the position feedback potentiometer. The potentiometer will output a voltage signal to the control circuit board for position feedback. The control circuit board determines the rotation direction and speed of the motor according to the position of the output shaft, so that the output shaft stops when it reaches the target.

2.2.5 PWM Control

2.2.5.1 PWM Introduction

Pulse width modulation (PWM) refers to the use of the digital output of a microprocessor to control an analog circuit, and is a method of digitally encoding the level of an analog signal.

PWM (Pulse Width Modulation): Pulse Width Modulation

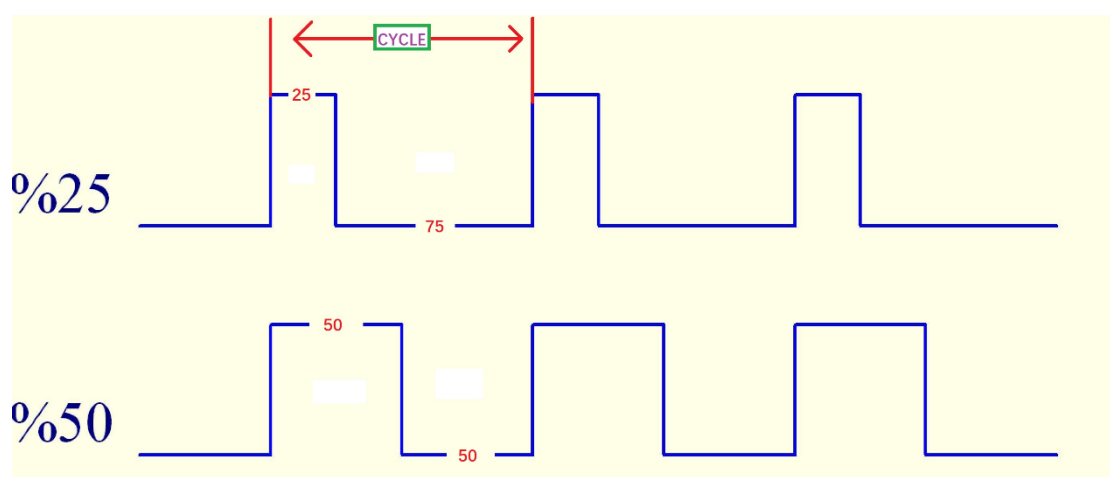
Pulse: square wave, frequency (freq)

Width: the width of the high level, the duty cycle (duty)

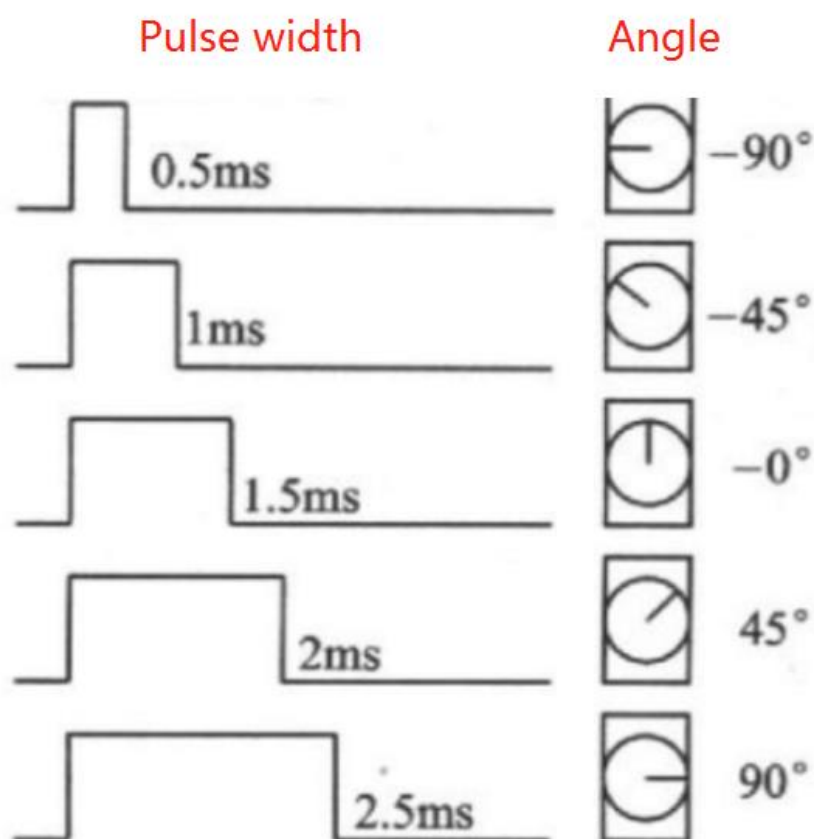
Cycle: CYCLE

Duty cycle: the proportion of high level (100&%)

Duty cycle static diagram:



The relationship between the output angle of MG90S Micro Servo and the pulse width of the input signal



CYCLE=20ms

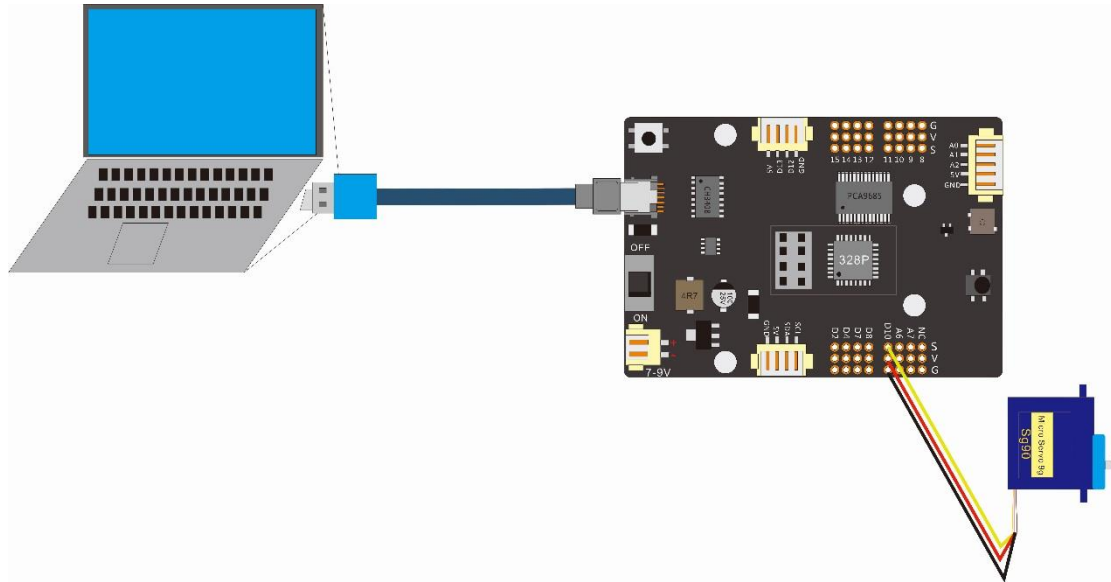
Duty_16=65532*Pulse Width/CYCLE

The relationship between the corresponding pulse width and duty_u16:

Pulse Width(ms)	Duty_16
0.5	1638
1	3276
1.5	4914
2	6552
2.5	8192

3. Wiring

The signal pin of the servo is connected to the D10 pin of the control board. Because the power of the servo is relatively small, it can be powered by the computer with a USB cable, and no external 18650 battery is needed in this lesson.



Wiring between the servo and the control board	
Connector of the Servo	Connector of the control board
+	5V
-	GND
Signal	D10

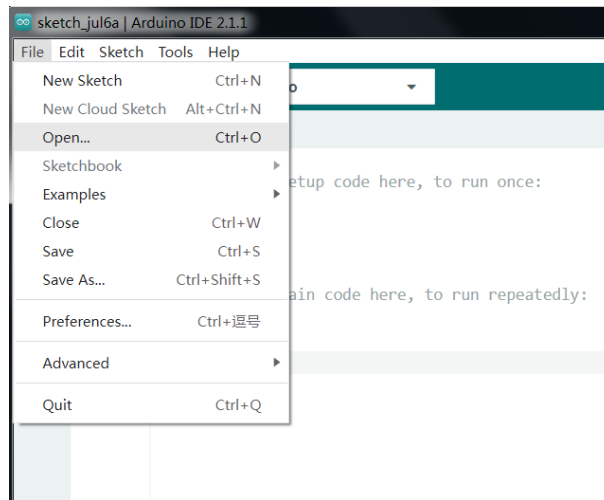
4. Upload the code and test

The code used in this lesson is placed in this folder: “[E:\CKK0002-master\Tutorial\sketches](#)”

4.1 Double-click the Arduino IDE shortcut on the desktop to open it

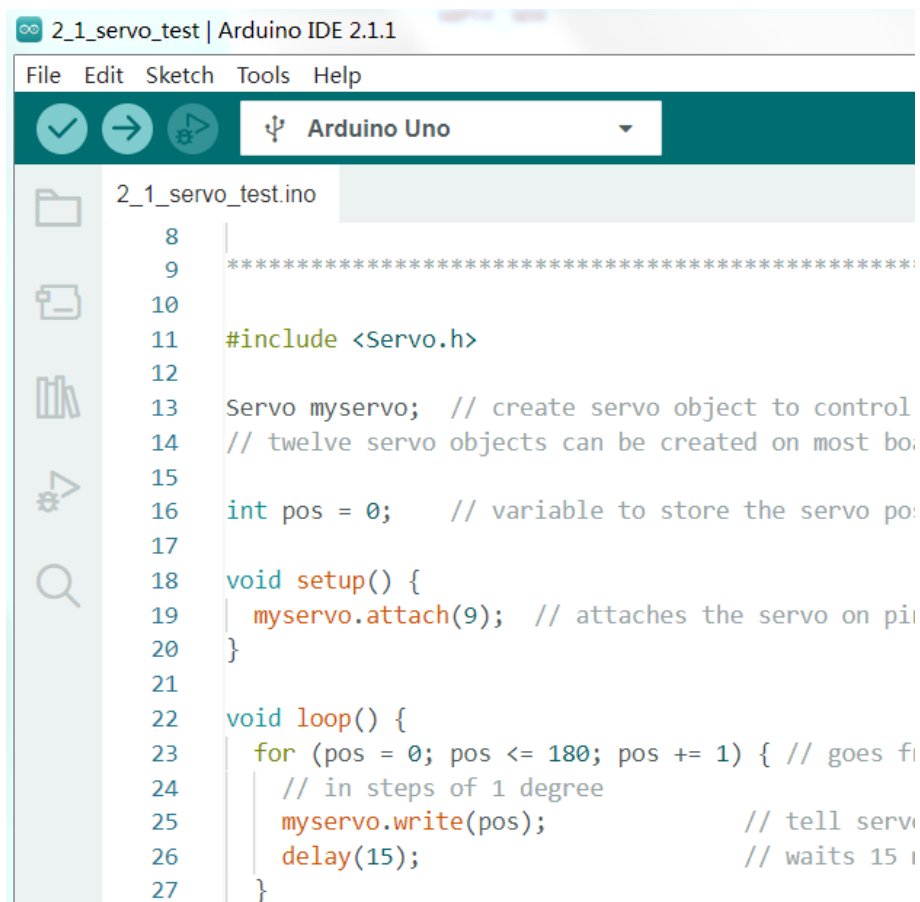


4.2 Click "File" --- "open"



4.3 Select the code in the folder named 2_1_servo_test:

E:\CKK0002-master\Tutorial\sketches\2_1_servo_test. Click "open"



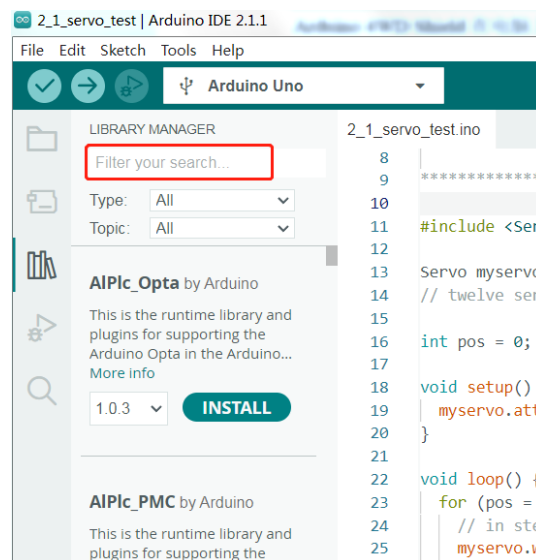
4.4 Select the board "Arduino UNO" and Port "COM3" (COM port is commonly known as an input output port for a device normally PC which enables communication between Arduino and

PC. You can check your arduino com number in device manager, the com port of our arduino board is recognized as COM3 in this tutorial)

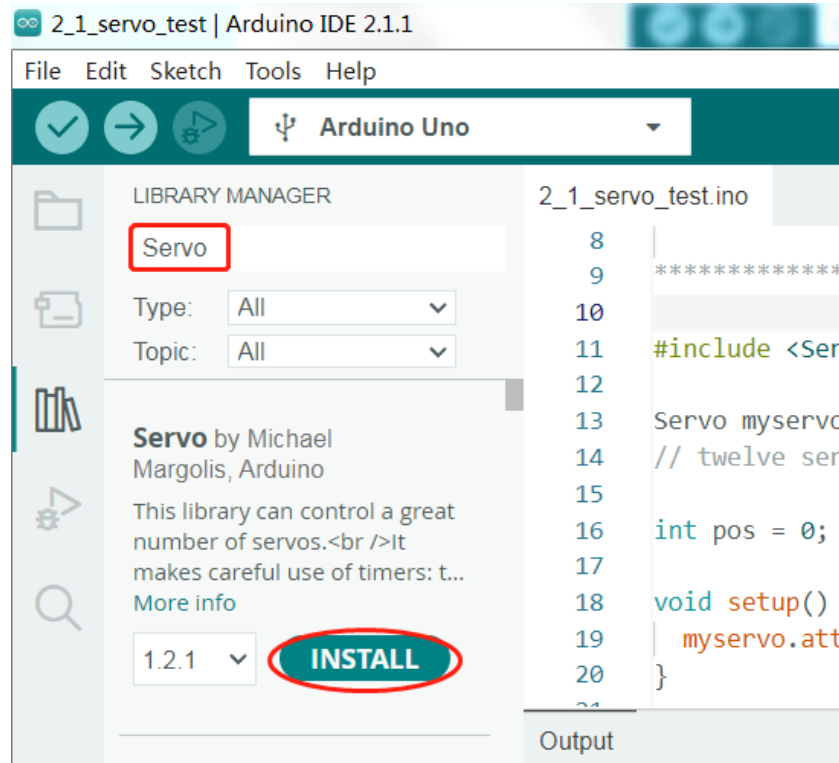


4.5 Install library [Servo.h](#)

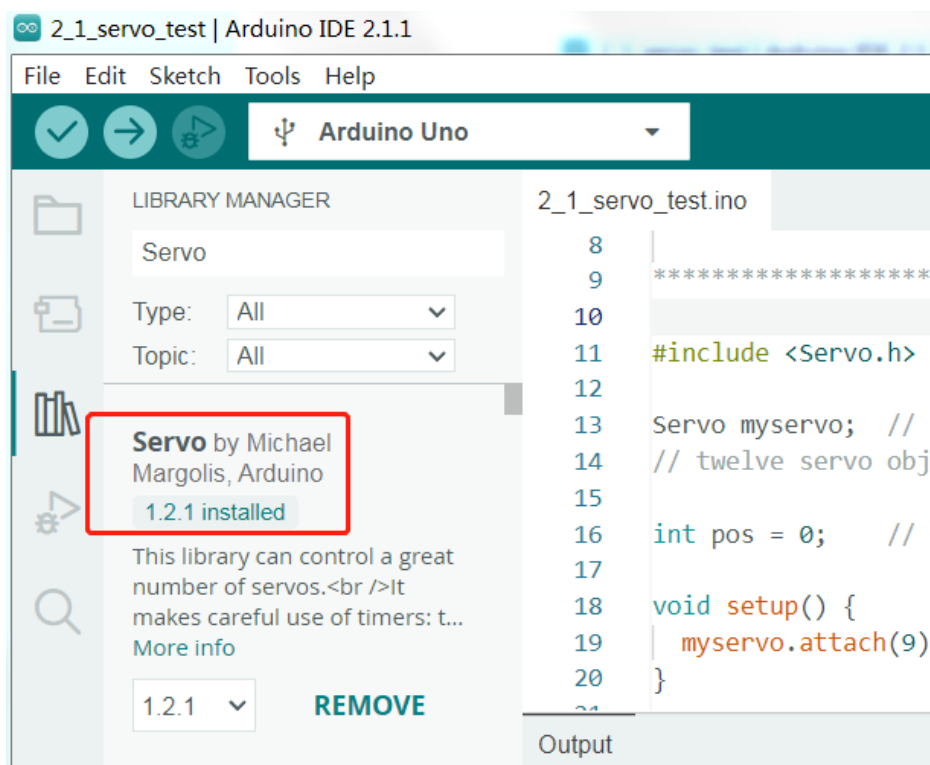
Method 1: Directly click "[Library Manager](#)" on the left side of the Arduino IDE interface, the interface is as follows



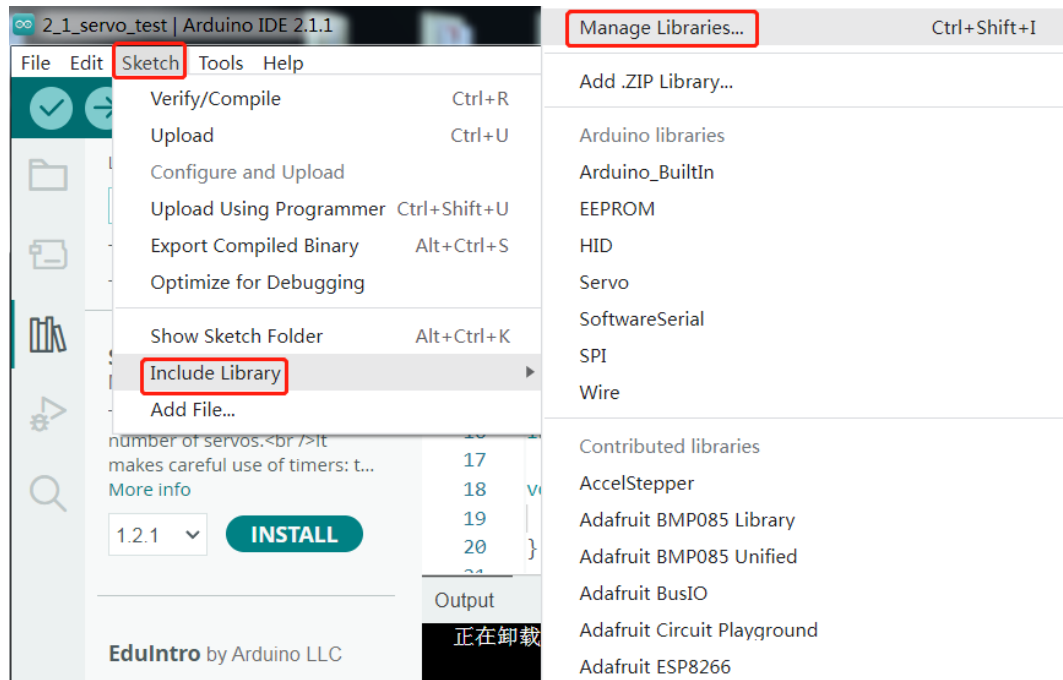
Search for Servo in the red search box, Library Manager will quickly match related library files



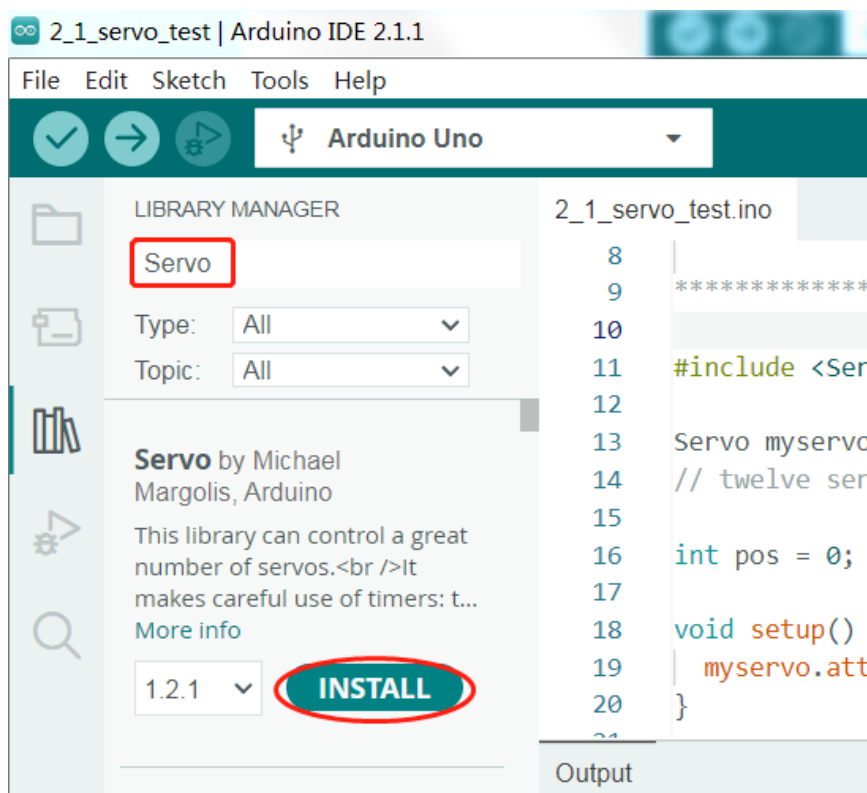
Click "INSTALL" to install the library file of Servo. After the installation is complete, it will display "installed"



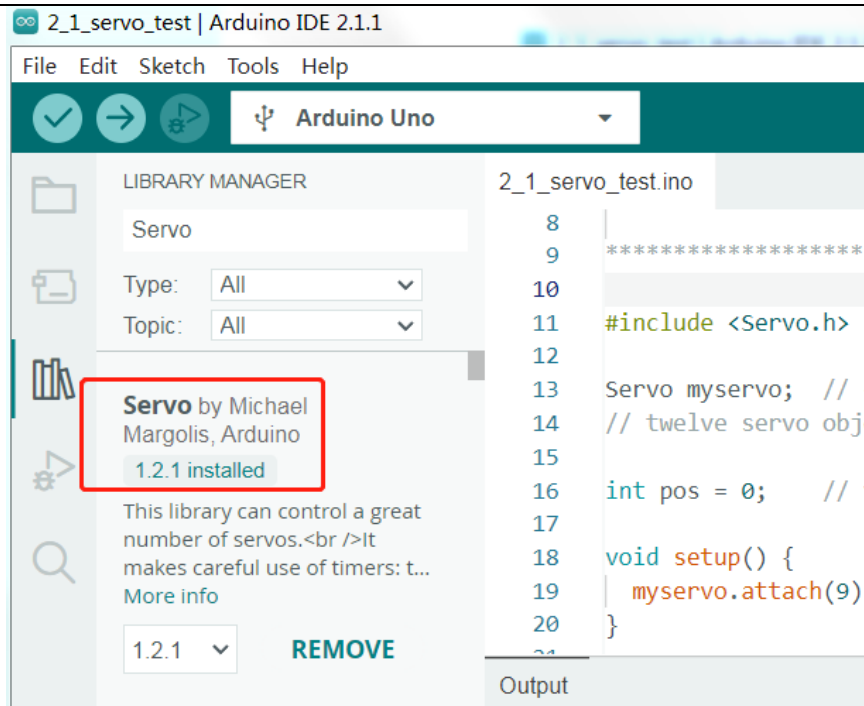
Method 2: After opening the 2_1_servo_test code, click "Sketch" --- "Include Library" --- "Manage Libraries..." to open the library management interface.



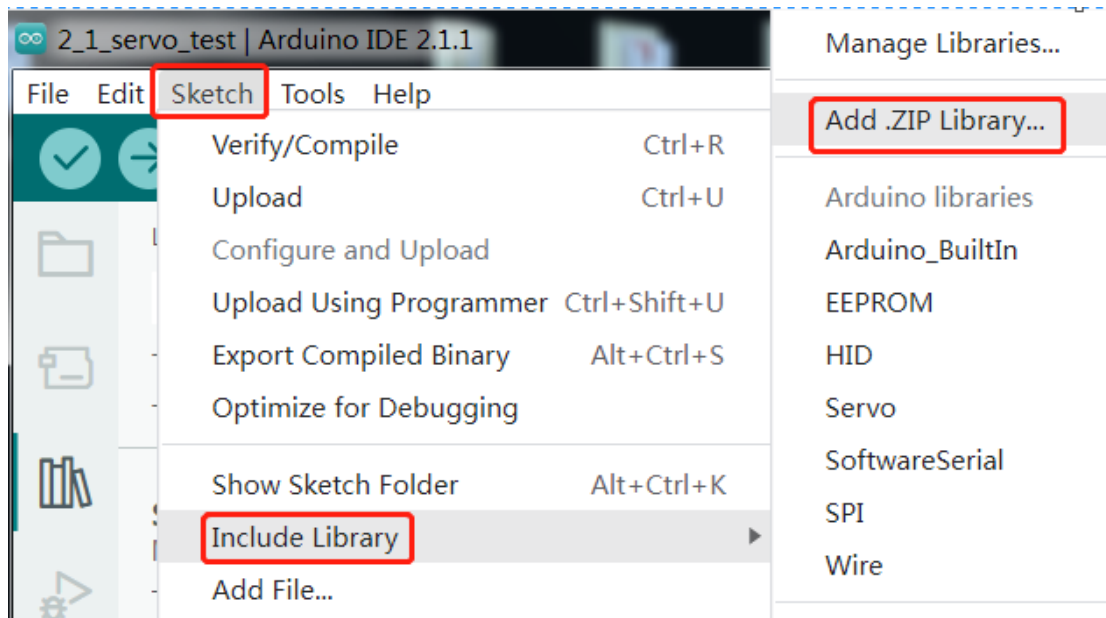
Search for Servo in the red search box, Library Manager will quickly match related library files



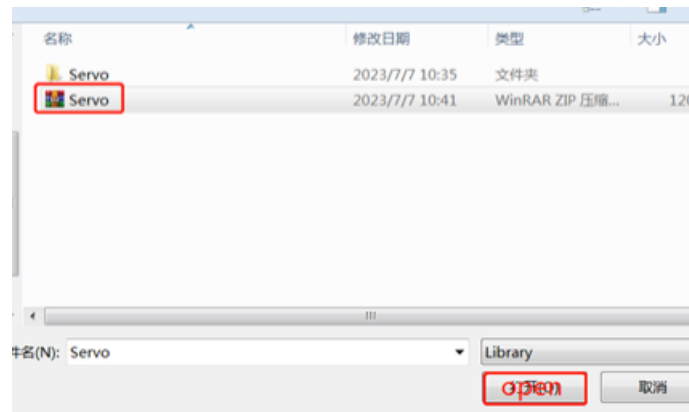
Click "INSTALL" to install the library file of Servo. After the installation is complete, it will display "installed"




Method 3: After opening the 2_1_servo_test code, click "Sketch" ---"Include Library"---"Add.ZIP Library..."




Find the Servo.ZIP file in [E:\CKK0002-master\Tutorial\Libraries](#).



Select the Servo.ZIP file and click "open", and the interface will display "installed" after the library file is installed

4.6 Click compile button , successfully compiled the code will display “Done compiling”

4.7 Before uploading the code, turn the ESP-01 switch on the control board to the side away from the "ESP-01" silk screen.

4.8 Click upload button , successfully uploading the code will display “Done uploading”. When code is uploaded successfully, the program starts to run, the servo will cycle through 180 degrees.

5. Code

Servo_test code:

```
#include <Servo.h>

Servo myservo; // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(10); // attaches the servo on pin 9 to the servo object
}
```

```
void loop() {  
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees  
    // in steps of 1 degree  
    myservo.write(pos);           // tell servo to go to position in variable 'pos'  
    delay(15);                   // waits 15 ms for the servo to reach the position  
  }  
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees  
    myservo.write(pos);           // tell servo to go to position in variable 'pos'  
    delay(15);                   // waits 15 ms for the servo to reach the position  
  }  
}
```

6. Adjust the servo to 65° before assembly

Before assembling the servo to the robot you need to adjust it to 65° so that it can work well with the ultrasonic module for the robot.

The code to adjust the servo to 65° is placed in this folder:

E:\CKK0002-master\Tutorial\sketches\Servo_65_ADJ

Or you can also directly copy and paste the code from below to the arduino IDE

```
#include<Servo.h>  
Servo myservo; // Create a servo class  
  
void setup() {  
  myservo.attach(10); //Set the servo control pin as D10  
  delay(100);         //delay 100ms  
}  
/////////////////////////////////////  
void loop() {  
  myservo.write(65); //The servo is 65 degrees  
  delay(1000);  
}
```

7. Any questions and suggestions are welcome

Thank you for reading this document!

If you find any errors and omissions in the tutorial, or if you have any suggestions and questions, please feel free to contact us:

cokoino@outlook.com

We will do our best to make changes and publish revisions as soon as possible.

If you want to learn more about smart cars, robots, learning kits and other technology products from us, please bookmark and pay attention to our website:

<http://cokoino.com/>

We will continue to launch interesting, cost-effective, innovative, user-friendly products.

LK COKOINO