

Lesson 5 - Test The Servo

Need to prepare:

- ◆ Three SG90 servos, one MG90S servo
- ◆ A LK COKOINO Shield
- ◆ A LK COKOINO Nano
- ◆ Board A USB Cable
- ◆ A Battery Case and Two 18650 batteries(Batteries are not included in the kit)

Goal: Test if each servo is working properly and understand how it works

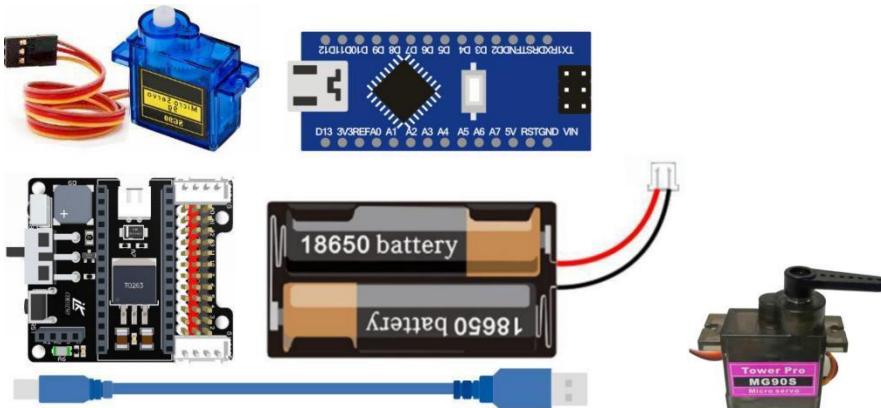


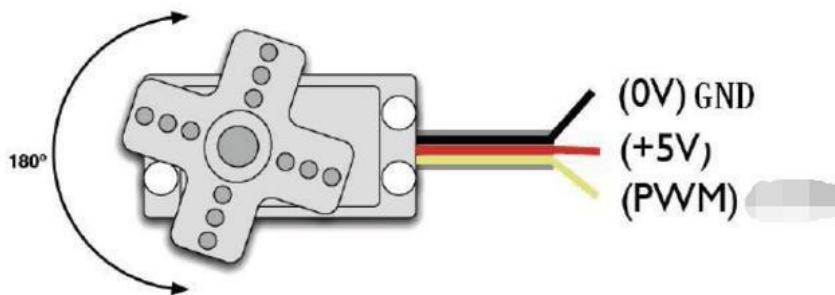
Table of Content

1. Overview :	2
2. The ways to control the servo	3
2.1 Wiring diagram	3
2.2 Method one for testing:	5
2.4 You can proceed to the next step.	8

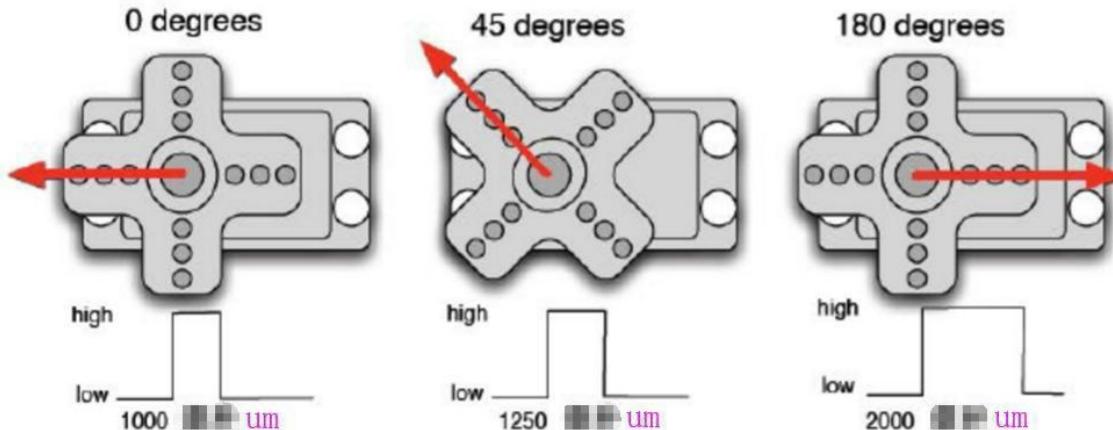
1. Overview:

A Servo is a small device that incorporates a two-wire DC motor, a gear train, a potentiometer, an integrated circuit, and an output shaft. Of the three wires that stick out from the motor casing, one is for power, one is for ground, and one is a control input line. The shaft of the servo can be positioned to specific angular positions by sending a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. If the coded signal changes, then the angular position of the shaft changes.

Servos are constructed from three basic pieces; a motor, a potentiometer (variable resistor) that is connected to the output shaft, and a control board. The potentiometer allows the control circuitry to monitor the current angle of the servo motor. The motor, through a series of gears, turns the output shaft and the potentiometer simultaneously. The potentiometer is fed into the servo control circuit and when the control circuit detects that the position is correct, it stops the motor. If the control circuit detects that the angle is not correct, it will turn the motor the correct direction until the angle is correct. Normally a servo is used to control an angular motion of between 0 and 180 degrees.



The angle of rotation of the steering gear is achieved by adjusting the duty cycle of the PWM (Pulse Width Modulation) signal. The period of the standard PWM (Pulse Width Modulation) signal is fixed at 20ms (50Hz). Theoretically, the pulse width distribution should be 1ms to Between 2ms. However, in fact, the pulse width can be between 0.5ms and 2.5ms, and the pulse width corresponds to the rotation angle of the steering gear from 0° to 180°.



2. The ways to control the servo

For the Arduino, there are two ways to control the servos.

One is that the Arduino's common digital pin generates square waves with different duty cycles to simulate PWM signals for servo positioning.

The second method is to directly control the steering gear by using the Arvoino's own Servo function. The advantage of this control method is programming. **The Arduino has limited drive capability, so an external power supply is required when it is necessary to control more than one servo.**

Explain the common functions of the Servo.h library file:

1, attach (interface) - set the interface of the servo.

2, write (angle) - set the steering angle of the servo, the range of angles that can be set is 0° to 180° .

3, read () - read the steering angle, can be understood as reading the value of the last write () command.

4. attached() - Determines whether the servo parameters have been sent to the interface where the servo is located.

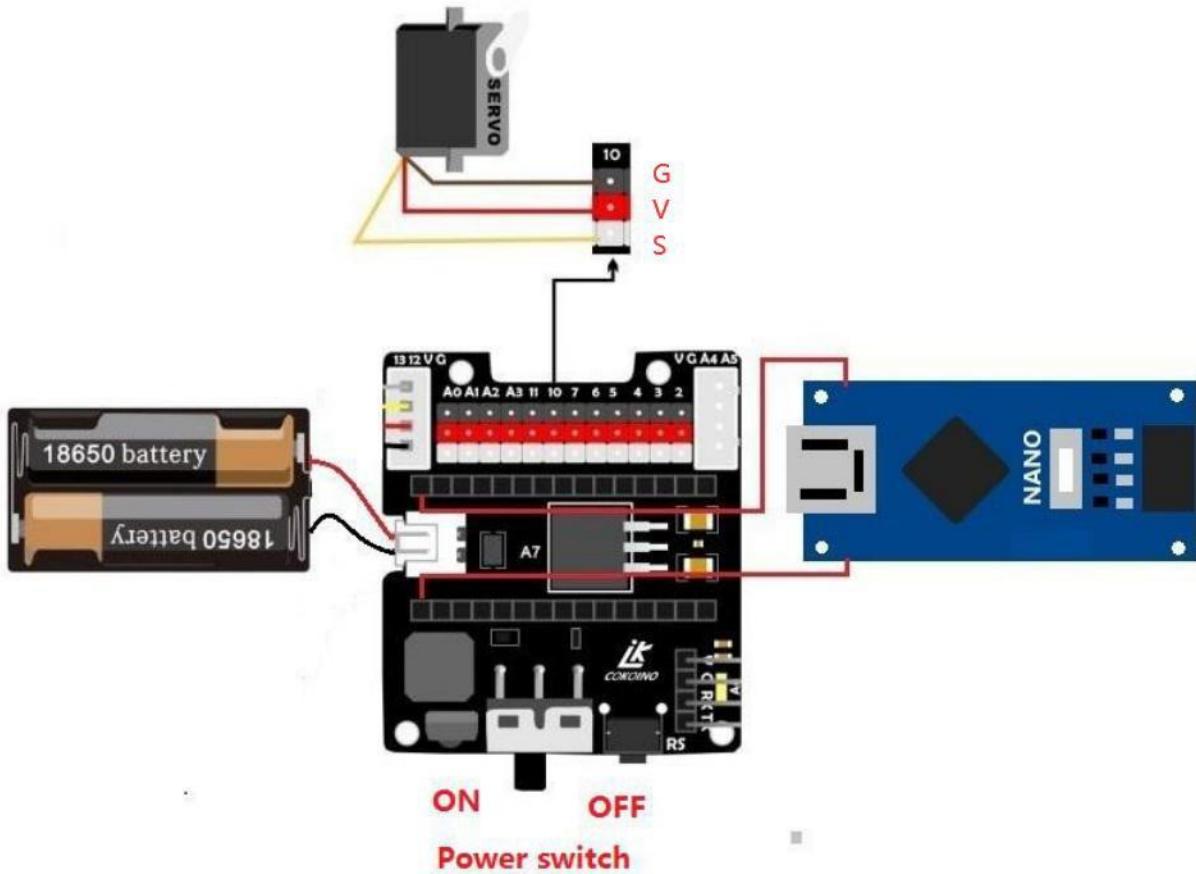
5, detach () - the servo is separated from the interface, the interface (9 or 10) can continue to be used as a PWM interface.

2.1 Wiring diagram

NOTE:

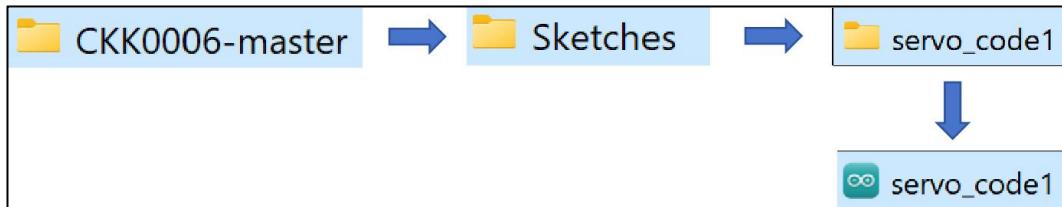
Turn on the power switch of the shield

The brown, red and orange wires of the servo are connected as shown below



2.2 Method one for testing:

Find the "servo_code1" code from the following path, open it with the Arduino IDE and upload it.



connect the PC to the NANO Shield board with a USB cable, select the corresponding board type and port in the IDE, upload the code to the NANO board.

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** servo_code1 | Arduino IDE 2.3.4
- Menu Bar:** File Edit Sketch Tools Help
- Tool Bar:** Checkmark, Refresh, Save, Board Selection (set to Arduino Nano), Upload
- Code Editor:** The code for `servo_code1.ino` is displayed:

```
1 int servopin=10; //Define digital interface 10 to connect
2 int myangle; //Define the Angle variable 0-180
3 int pulselength; //Define the pulse width variable
4 int val; //0-9
5
6 void servopulse(int servopin,int myangle) //Define an impulse
7 {
8     pulselength=(myangle*11)+500; //Convert Angle to pulse length
9
10    digitalWrite(servopin,HIGH); //Set the interface high
11
12    delayMicroseconds(pulselength); //Delay milliseconds
13
14    digitalWrite(servopin,LOW); //Lower the interface
```
- Output Panel:** Shows the compilation message:

```
Sketch uses 2472 bytes (8%) of program storage space. Maximum is 32256 bytes.
Global variables use 236 bytes (11%) of dynamic memory, leaving 18300 bytes free.
```
- Status Bar:** Done uploading.

Open the serial monitor and set the baud rate:9600

Input the number 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 in the serial monitor in turn, you will find that the servo rotates 20°, 10°, 60°, 80°, 100°, 120 °, 140°, 160°, 180°

For example, if you enter 9, the rudder turns 180 degrees, if you enter 3, the rudder turns 60 degrees.



Install the servo arm on the servo to see its angle change more clearly.

The screenshot shows the Arduino IDE interface. The top bar says "servo_code1 | Arduino IDE 2.3.4". The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar has icons for file operations and a "Arduino Nano" dropdown. The left sidebar shows a folder icon and a sketch named "servo_code1.ino". The code editor contains the following code:

```
1 int servopin=10; //Define digital interface 10 to connect servo serv
2 int myangle; //Define the Angle variable 0-180
3 int pulsewidth; //Define the pulse width variable
4 int val; //0-9
5
6 void servopulse(int servopin,int myangle) //Define an impulse function
7 {
8     pulsewidth=(myangle*11)+500; //Convert Angle to 500-2480
9
10    digitalWrite(servopin,HIGH); //Set the interface level to high
11
12    delayMicroseconds(pulsewidth); //Delay millisecond
13
14    digitalWrite(servopin,LOW); //Lower the interface level to low
```

The bottom section shows the "Serial Monitor" tab selected. The output window displays the following text:

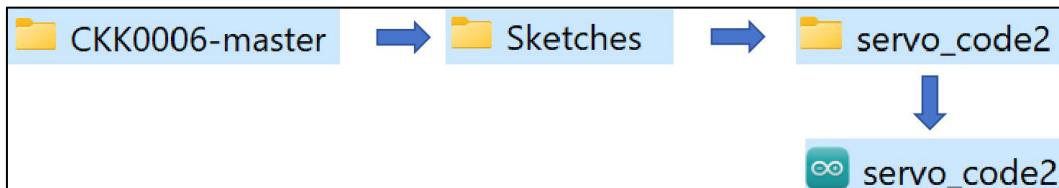
```
servo=0 serial simple ready
moving servo to 180
```

Annotations in red are overlaid on the image:

- A red arrow points to the input field in the Serial Monitor window, with the text "Enter the number 9 and press enter".
- A red box highlights the "moving servo to 180" line in the output window, with the text "Rotate the servo to the corresponding angle" below it.

2.3 Method Two for

Find the "servo_code2" code from the following path, open it with the Arduino IDE.



Upload the code:

Connect the PC to the NANO Shield board with a USB cable, select the corresponding board type and port in the IDE.

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

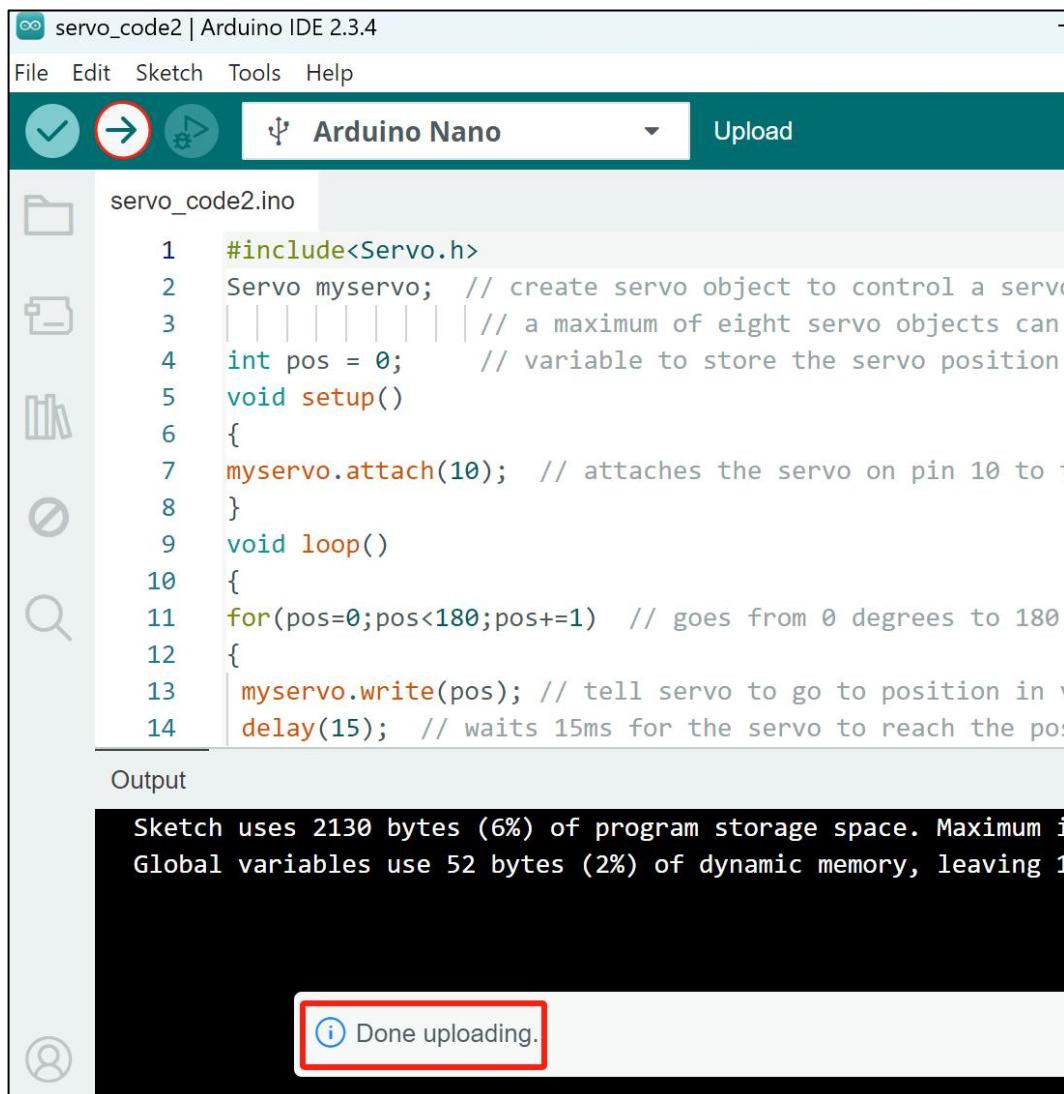
The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** servo_code2 | Arduino IDE 2.3.4
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Open, Save, Print, and Verify/Compile. The Verify/Compile button is highlighted with a red circle and a checkmark.
- Board Selection:** Set to Arduino Nano
- Sketch Area:** Displays the code for "servo_code2.ino":

```
1 #include<Servo.h>
2 Servo myservo; // create servo object to control a se
3 | | | | | | | | // a maximum of eight servo objects c
4 int pos = 0; // variable to store the servo positio
5 void setup()
6 {
7   myservo.attach(10); // attaches the servo on pin 10 t
8 }
9 void loop()
10 {
11   for(pos=0;pos<180;pos+=1) // goes from 0 degrees to 1
12   {
13     myservo.write(pos); // tell servo to go to position i
14     delay(15); // waits 15ms for the servo to reach the
```
- Output Area:** Shows the compilation results:

```
Sketch uses 2130 bytes (6%) of program storage space. Maximum
Global variables use 52 bytes (2%) of dynamic memory, leaving
```
- Status Bar:** At the bottom right, a message "Done compiling." is displayed inside a red-bordered box.

Click upload button , successfully uploading the code will display “Done uploading”



The screenshot shows the Arduino IDE interface. The title bar reads "servo_code2 | Arduino IDE 2.3.4". The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar features icons for upload (circled in red), refresh, and other functions. A dropdown menu shows "Arduino Nano". The main area displays the code for "servo_code2.ino":

```
1 #include<Servo.h>
2 Servo myservo; // create servo object to control a servo
3 // a maximum of eight servo objects can be attached
4 int pos = 0; // variable to store the servo position
5 void setup()
6 {
7 myservo.attach(10); // attaches the servo on pin 10 to the servo object
8 }
9 void loop()
10 {
11 for(pos=0;pos<180;pos+=1) // goes from 0 degrees to 180 degrees in steps of 1 degree
12 {
13 myservo.write(pos); // tell servo to go to position in variable 'pos'
14 delay(15); // waits 15ms for the servo to reach the position
}
```

The "Output" window shows the compilation results:

```
Sketch uses 2130 bytes (6%) of program storage space. Maximum memory usage is 32768 bytes.
Global variables use 52 bytes (2%) of dynamic memory, leaving 1544 bytes for local variables.
Global variables use 52 bytes (2%) of dynamic memory, leaving 1544 bytes for local variables.
```

A message at the bottom of the output window is highlighted with a red box:  Done uploading.

Results:

Turn on the power switch on the shield, you can see that the servo starts running from 0 to 180 degrees, and then from 180 degrees to 0 degrees.



2.4 You can proceed to the next step.