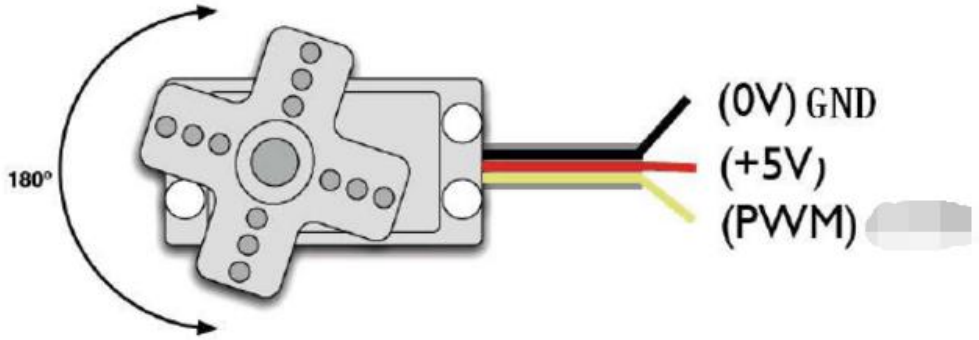
CKB0002 COKOINO 180 Servo Motor

1Overview：

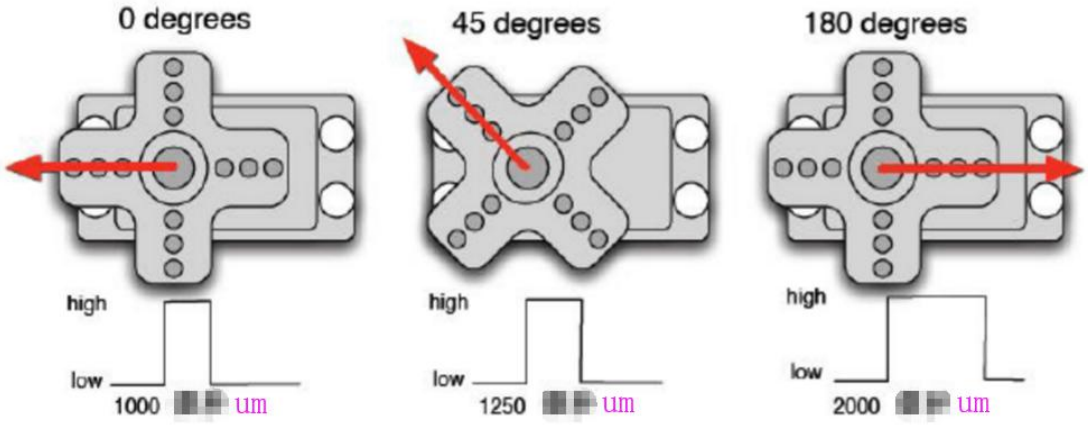
A Servo is a small device that incorporates a two wire DC motor, a gear train, a potentiometer, an integrated circuit, and an output shaft. Of the three wires that stick out from the motor casing, one is for power, one is for ground, and one is a control input line. The shaft of the servo can be positioned 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. Normally a servo is used to control an angular motion of between 0 and 180 degrees.

The servo we provide, the brown wire is the grounding wire, the red is the positive line of the power supply, and the orange is the signal line.



The rotation angle of the servo 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, the pulse width can be between 0.5ms and 2.5ms, and the pulse width corresponds to the steering angle of the servo from 0° to 180°. It is worth noting that the angle of rotation of the steering gear of different brands will also be

different.

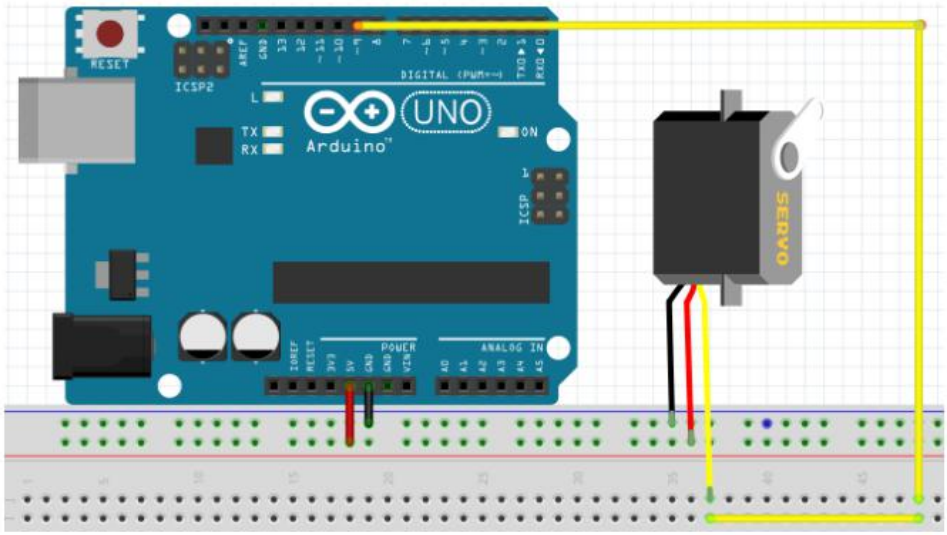


标准 1ms---2ms 示意图

Experiment:

The components required for this experiment only require one servo and one jumper.
There are two ways to control the servos with the Arduino. One is to generate square waves with different duty cycles through the Arduino's common digital sensor interface, and simulate PWM signals for servo positioning.
The second is to directly use Arduino's own Servo function to control the servo. Arduino's own function can only use the digital 9, 10 interface, so when you need to control more than one servo, you need an external power supply.
Wiring diagram:

UNO R3	9G servo
5V	red
G	back
9	yellow



Explanation of common functions of Servo.h library files：

1、attach（interface）——The interface used to set the servo. Only 9 or 10 interfaces are available.
2、write（angle）——Statement for setting the steering angle of the servo. The range of angles that can be set is 0° to 180°.
3、read（）——a statement for reading the angle of the servo, can be understood as reading the value in the last write() command
4、attached（）——Determine if the servo parameters have been sent to the interface where the servo is located.
5、detach（）——Separating the servo from its interface, the interface (9 or 10) can continue to be used as a PWM interface.

Method one：

Code：

```
int servopin=11;      //Define digital interface 9 to connect servo servo signal line
int myangle;           //Define the Angle variable 0-180
int pulsewidth;       //Define the pulse width variable
int val;               //0-9

void servopulse(int servopin,int myangle)    //Define an impulse function
{
pulsewidth=(myangle*11)+500;                  //Convert Angle to 500-2480 pulse width

digitalWrite(servopin,HIGH);                 //Set the interface level of steering gear to high

delayMicroseconds(pulsewidth);               //Delay millisecond

digitalWrite(servopin,LOW);                  //Lower the interface level of the steering gear

delay(20-pulsewidth/1000);
}

void setup()
{
pinMode(servopin,OUTPUT);            //Set servo interface as output interface set servo interface as
output mode

Serial.begin(9600);                  //The baud rate is 9,600

Serial.println("servo=o_seral_simple ready" ) ;
}

void loop()                          //Main loop function
{
val=Serial.read();                   //Read the value of the serial port

if(val>'0'&&val<='9')
{
val=val-'0';
val=val*(180/9);                     //Convert Numbers into angles
Serial.print("moving servo to ");
                                     //DEC:Converts the number to an angular decimal
representation that outputs the ASCII encoded value of b,
```

```
                                        //followed by a carriage return and a newline character
Serial.print(val,DEC);
Serial.println();
for(int i=0;i<=50;i++)
{
servopulse(servopin,val);                  //Call the impulse function
}
}
}
```

Experimental operation and results:

Copy the above code to the Arduino IDE, connect the UNO R3 motherboard to the PC with a USB cable, select the corresponding board type and port in the IDE, and upload the code to UNO R3.
It can be seen that the servo starts to run from 0 to 180 degrees, then run from 180 degrees to 0 degrees.

Method two：

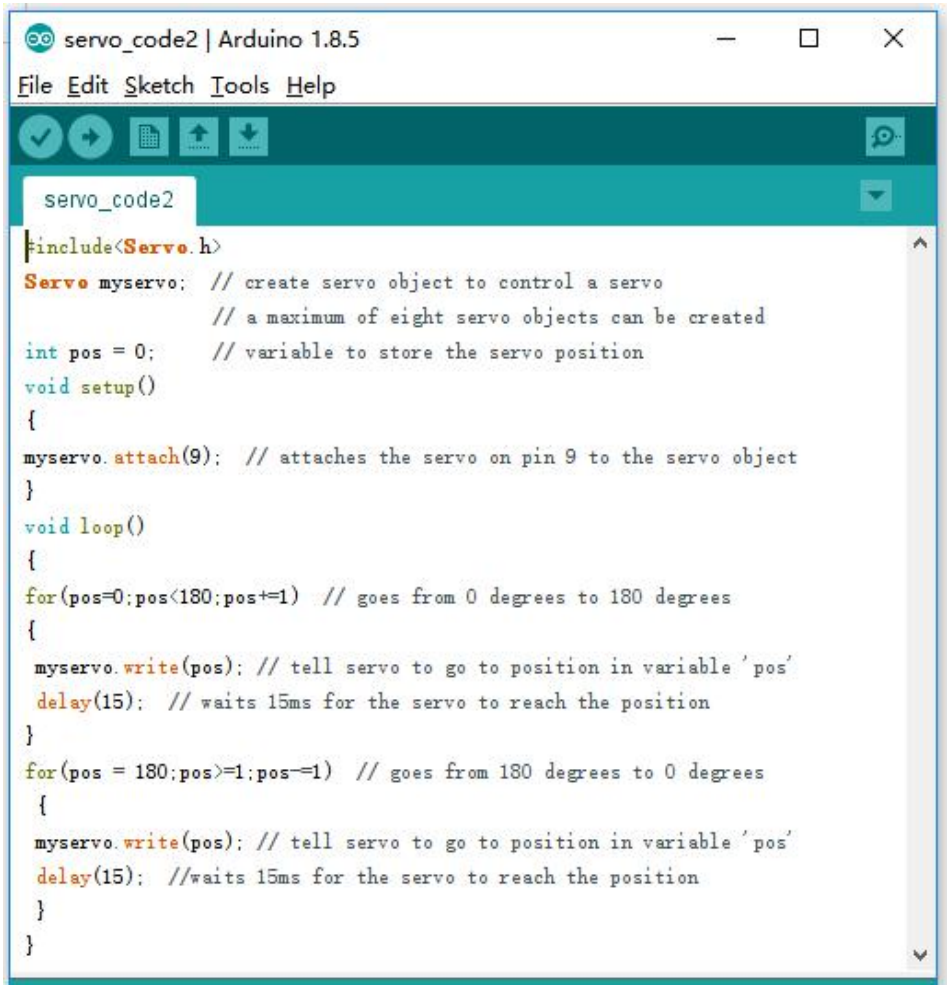
Code：

```
#include<Servo.h>
Servo myservo;    // create servo object to control a servo
                        // a maximum of eight servo objects can be created
int pos = 0;          // variable to store the servo position
void setup()
{
myservo.attach(9);    // 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
{
 myservo.write(pos); // tell servo to go to position in variable 'pos'
 delay(15);    // waits 15ms for the servo to reach the position
}
for(pos = 180;pos>=1;pos-=1)    // goes from 180 degrees to 0 degrees
  {
 myservo.write(pos); // tell servo to go to position in variable 'pos'
 delay(15);    //waits 15ms for the servo to reach the position
  }
}
```

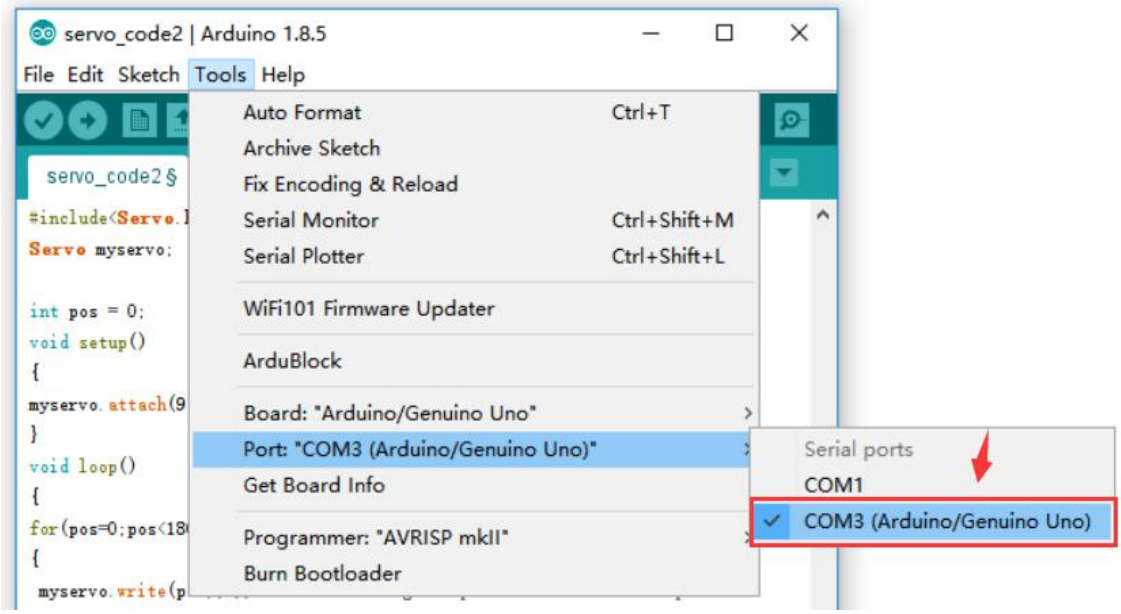
Experimental operation and results:

Copy the above code to the Arduino IDE, connect the UNO R3 motherboard to the PC with a USB cable, select the corresponding board type and port in the IDE, and upload the code to UNO R3.
It can be seen that the steering gear starts to run from 0 to 180 degrees, then run from 180 degrees to 0 degrees.

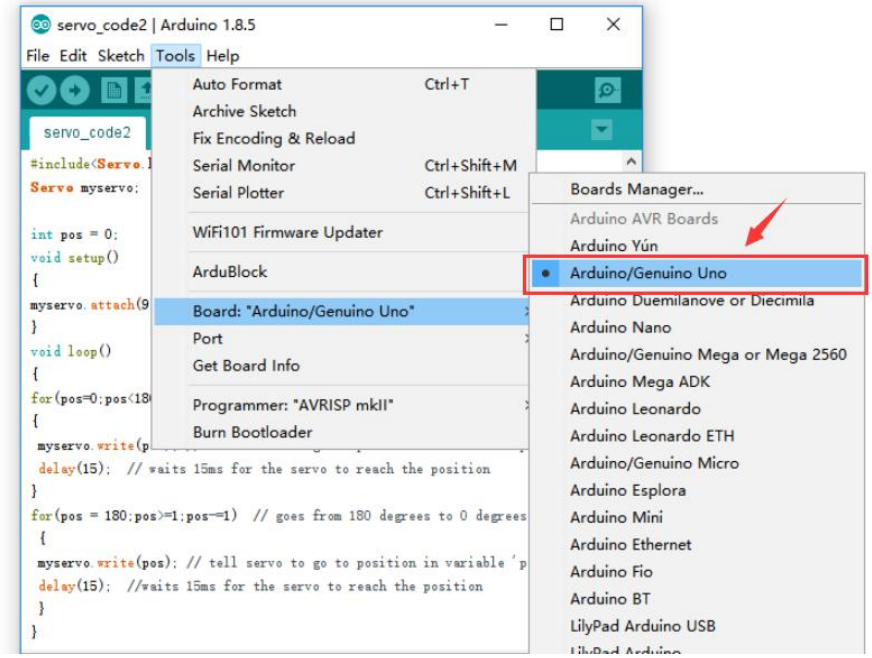
1、


servo_code2 | Arduino 1.8.5
File Edit Sketch Tools Help
servo_code2
#include<Servo.h>
Servo myservo; // create servo object to control a servo
                // a maximum of eight servo objects can be created
int pos = 0;   // variable to store the servo position
void setup()
{
myservo.attach(9); // 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
{
 myservo.write(pos); // tell servo to go to position in variable 'pos'
 delay(15); // waits 15ms for the servo to reach the position
}
for(pos = 180;pos>=1;pos-=1) // goes from 180 degrees to 0 degrees
 {
 myservo.write(pos); // tell servo to go to position in variable 'pos'
 delay(15); //waits 15ms for the servo to reach the position
 }
}

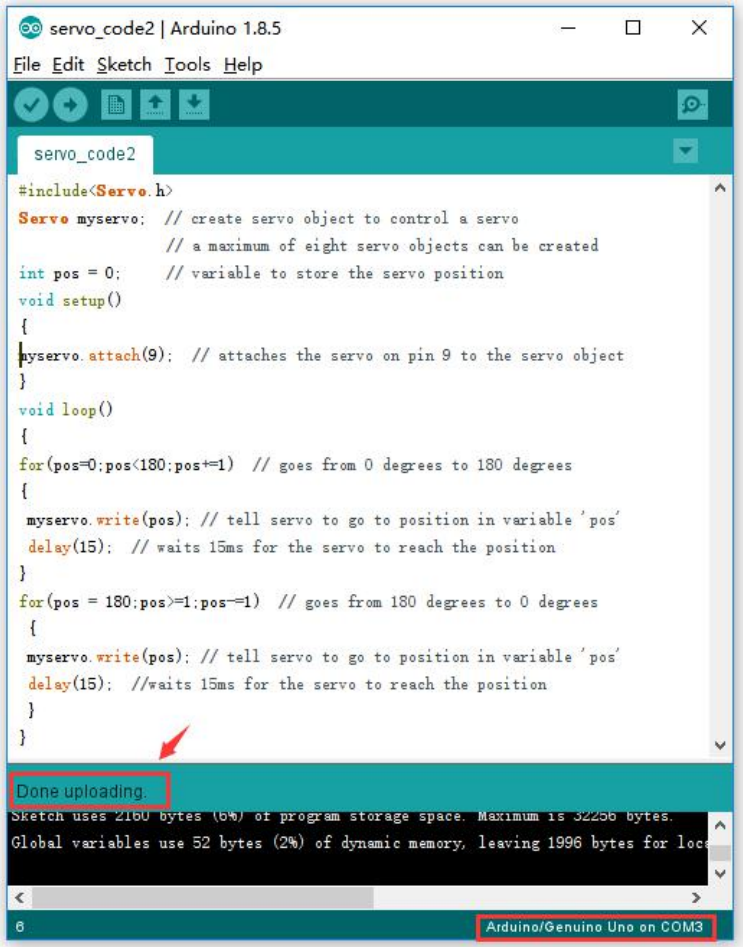
2、


servo_code2 | Arduino 1.8.5
File Edit Sketch Tools Help
servo_code2 §
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
ArduBlock
Board: "Arduino/Genuino Uno"
Port: "COM3 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
COM3 (Arduino/Genuino Uno)

3、


servo_code2 | Arduino 1.8.5
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
ArduBlock
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
Arduino Yún
Arduino/Genuino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino/Genuino Mega or Mega 2560
Arduino Mega ADK
Arduino Leonardo
Arduino Leonardo ETH
Arduino/Genuino Micro
Arduino Esplora
Arduino Mini
Arduino Ethernet
Arduino Fio
Arduino BT
LilyPad Arduino USB

4、


servo_code2 | Arduino 1.8.5
File Edit Sketch Tools Help
servo_code2
#include<Servo.h>
Servo myservo; // create servo object to control a servo
// a maximum of eight servo objects can be created
int pos = 0; // variable to store the servo position
void setup()
{
myservo.attach(9); // 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
{
myservo.write(pos); // tell servo to go to position in variable 'pos'
delay(15); // waits 15ms for the servo to reach the position
}
for(pos = 180;pos>=1;pos-=1) // goes from 180 degrees to 0 degrees
{
myservo.write(pos); // tell servo to go to position in variable 'pos'
delay(15); //waits 15ms for the servo to reach the position
}
}
Done uploading.
Sketch uses 2160 bytes (6%) of program storage space. Maximum is 32256 bytes.
Global variables use 52 bytes (2%) of dynamic memory, leaving 1996 bytes for loc
6
Arduino/Genuino Uno on COM3