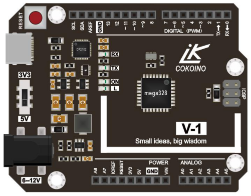
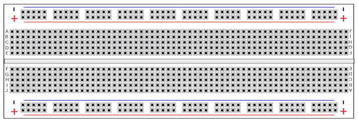
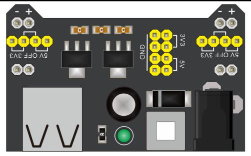
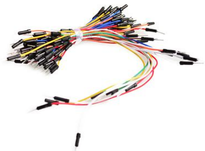
15.Two methods to drive the servo

ABOUT THIS PROJECT：

You will learn：

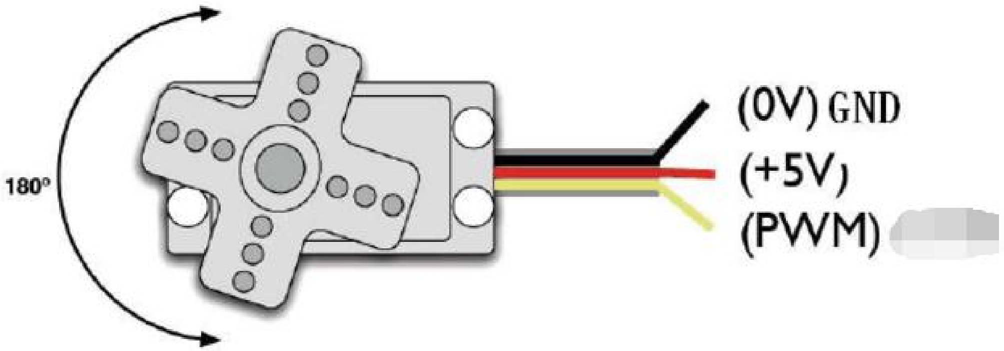
- How to drive the servo to rotate different degree
- How to drive the servo to rotate from 0 degree to 180 degree, and from 180 degree to 0 degree

1、Things used in this project：

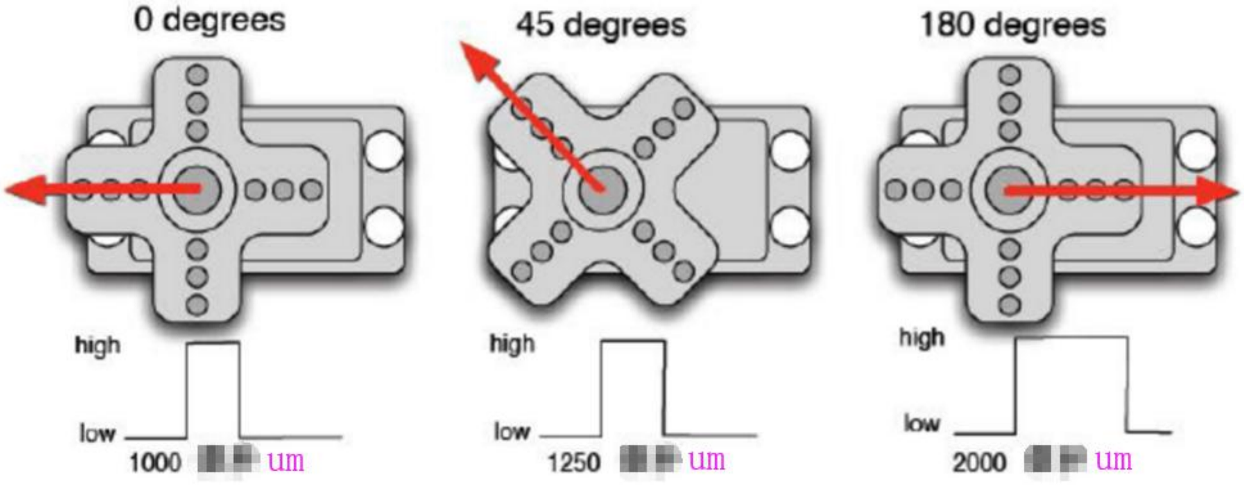
Hardware components	Picture	Quantity
V-1 board		1 PCS
Breadboard		1 PCS
9V Battery Snap Connector (you need to buy 9V battery yourself)		1 PCS
Breadboard power module		1 PCS
Male to Male DuPont Cable		5 PCS
30 CM USB Cable		1 PCS
Servo		1 PCS

2、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°.



3、Two ways to control the servo

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

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

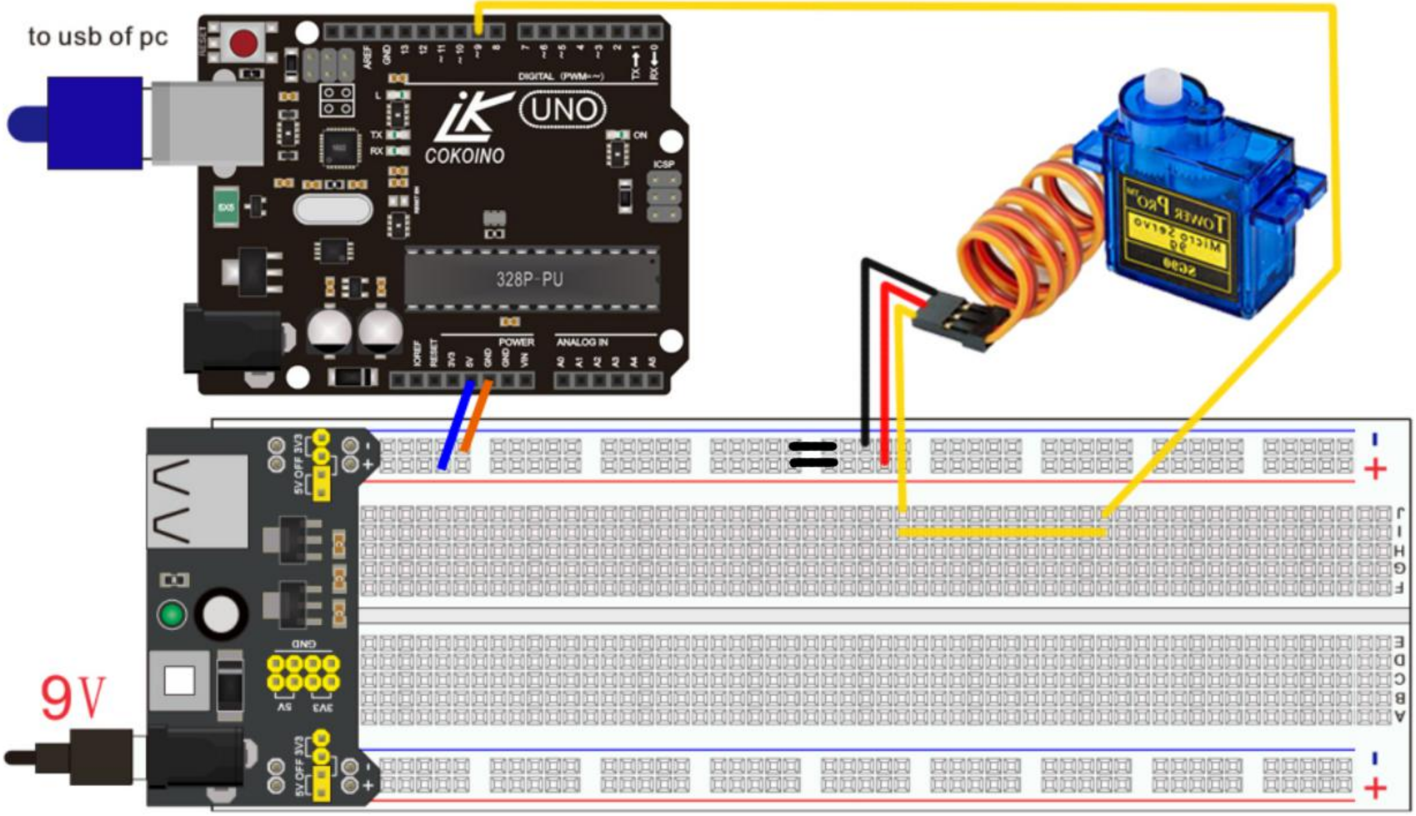
The second method is to directly control the steering gear by using the Arvoino's own Servo function. The advantage of this control method is programming. The Arduino has limited drive capability, so an external power supply is required when it is necessary to control more than one servo.

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.

Wiring table:

V-1 board	9G servo
5V	red
G	back
9	yellow



3.1、Method one:

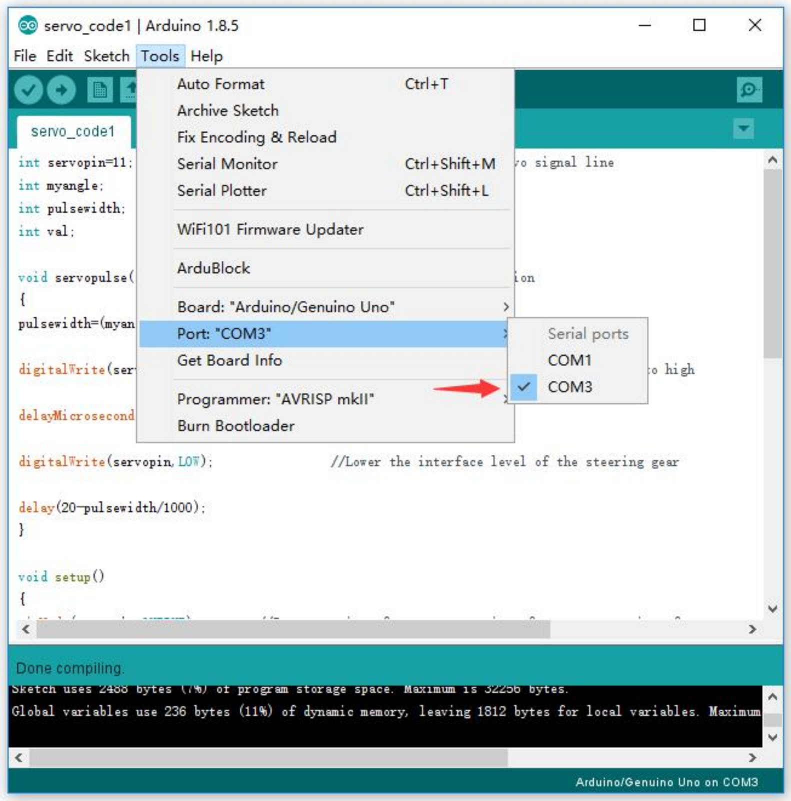
Sketch：

```
int servopin=9;//Define digital interface 9 to connect signal line of the servo
int myangle;//Defines the angle variable 0-180.
int pulsewidth;//Define pulse width variables
int val; //0-9

void servopulse(int servopin,int myangle)//Define a pulse function
{
pulsewidth=(myangle*11)+500;//Convert the angle to the pulse width value of 500 ≤ 2480

digitalWrite(servopin,HIGH);//The servo interface level will be high.

delayMicroseconds(pulsewidth);//Microseconds of delay pulse width

digitalWrite(servopin,LOW);//Lower the servo interface level

delayMicroseconds(2500-pulsewidth);
}

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

Serial.begin(9600);//Connect to serial port, baud rate is 9600.

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

void loop()//Convert 0 to 9 number to 0 to 180 angles and let the LED blink the corresponding number of times.
{
val=Serial.read();//Read the value of the serial port

if(val>'0'&&val<='9')
{
val=val-'0';//Convert feature quantities into numerical variables
val=val*(180/9);//Convert numbers to angles
Serial.print("moving servo to ");
//DEC:Outputs the ASCII encoded value of b in decimal form, followed by a carriage return and line feed symbol
Serial.print(val,DEC);
Serial.println();
for(int i=0;i<=50;i++) //Give the servo enough time to turn it to the specified angle
{
servopulse(servopin,val);//Reference pulse function
}
}
}
```

3.1.1、Step：

Connect the computer and V-1 board with a USB cable and copy the above sample code to the Arduino IDE as shown below：

1. Select the board type and port

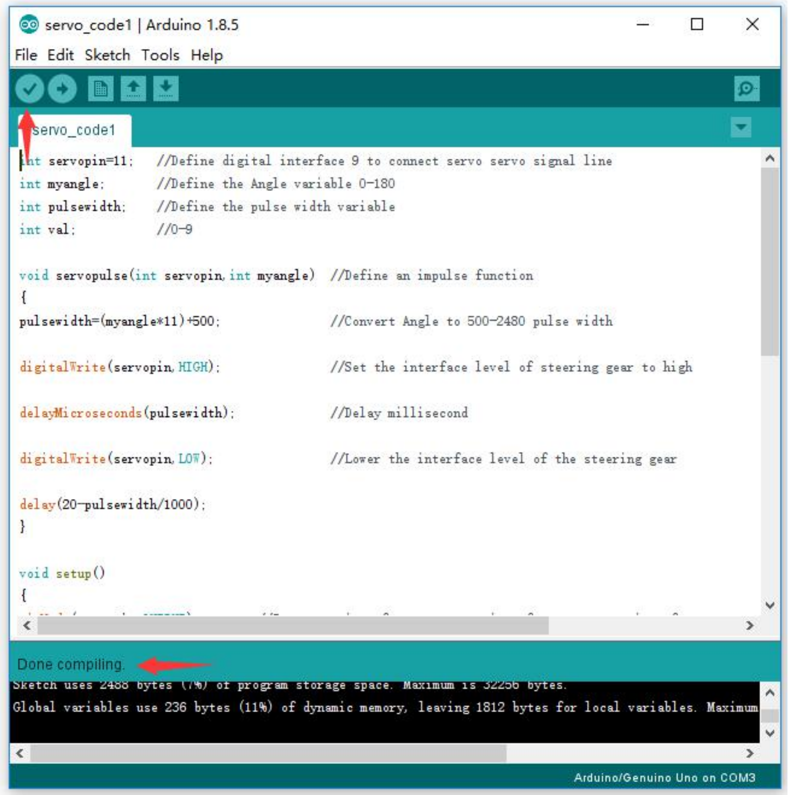

servo_code1 | Arduino 1.8.5
File Edit Sketch Tools Help
servo_code1
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
ArduBlock
Board: "Arduino/Genuino Uno"
Port: "COM3"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
✓ COM3
Arduino/Genuino Uno on COM3

```
int servopin=11;                                    o signal line
int myangle;
int pulsewidth;
int val;

void servopulse(                                   ion
{
pulsewidth=(myan

digitalWrite(ser                                   o high

delayMicrosecond

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

delay(20-pulsewidth/1000);
}

void setup()
{
```

Done compiling.

```
Sketch uses 2488 bytes (7%) of program storage space. Maximum is 32256 bytes.
Global variables use 236 bytes (11%) of dynamic memory, leaving 1812 bytes for local variables. Maximum
```

2. Verification code

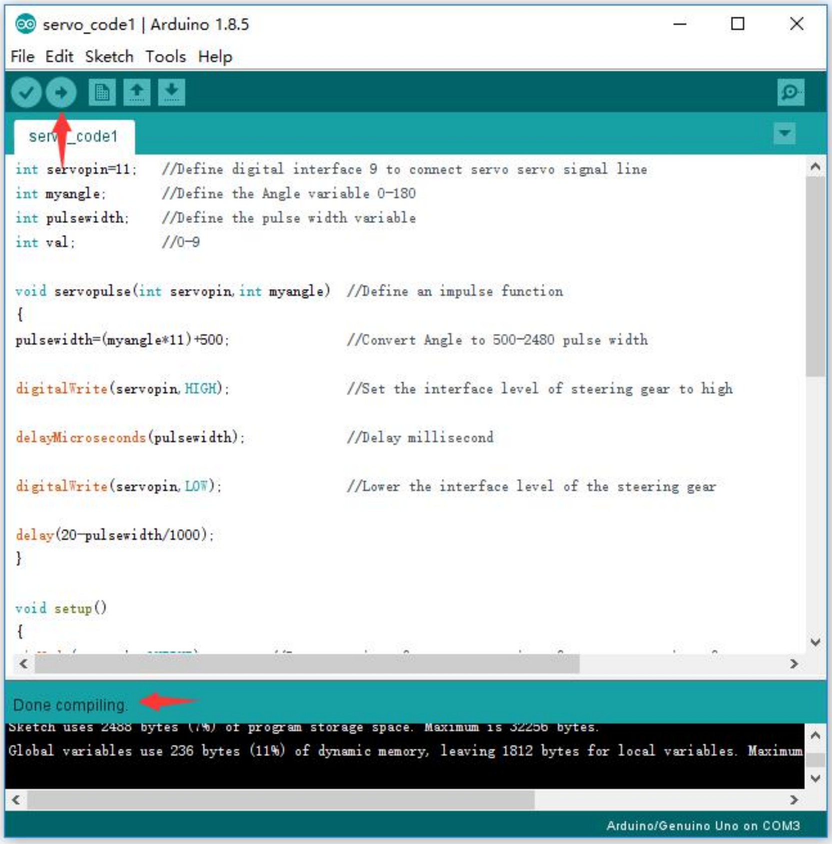

servo_code1 | Arduino 1.8.5
File Edit Sketch Tools Help
servo_code1
Arduino/Genuino Uno on COM3

```
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()
{
```

Done compiling.

```
Sketch uses 2488 bytes (7%) of program storage space. Maximum is 32256 bytes.
Global variables use 236 bytes (11%) of dynamic memory, leaving 1812 bytes for local variables. Maximum
```

4、Upload the code



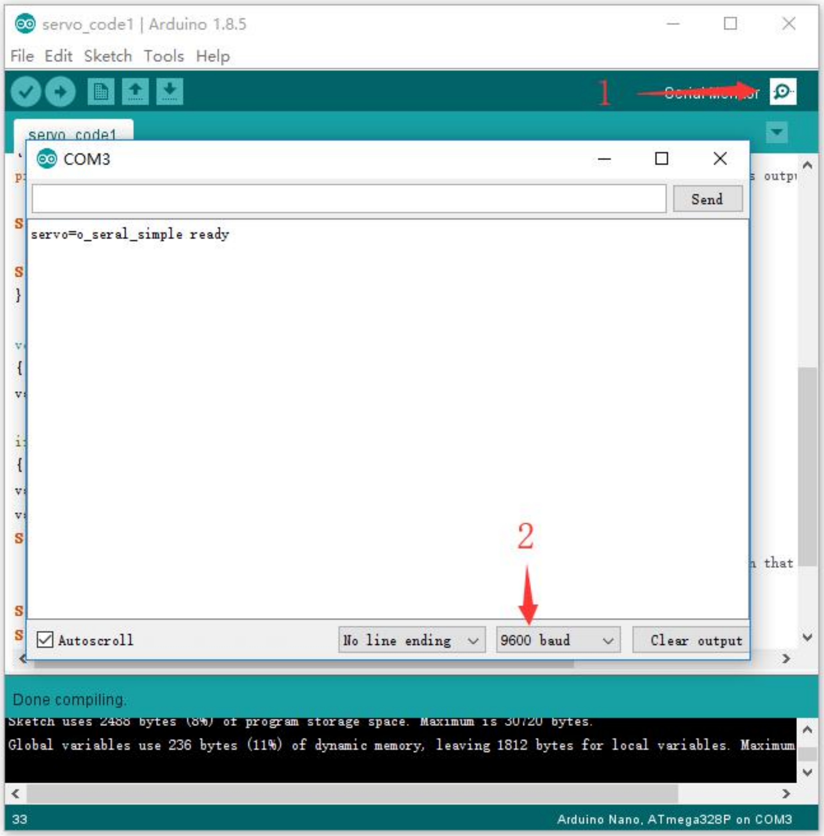
```
int servopin=11;    //Define digital interface 9 to connect servo servo signal line
int myangle;        //Define the Angle variable 0-180
int pulsewidth;     //Define the pulse width variable
int val;            //0-9

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()
{
```

Done compiling.

```
Sketch uses 2488 bytes (7%) of program storage space. Maximum is 32256 bytes.
Global variables use 236 bytes (11%) of dynamic memory, leaving 1812 bytes for local variables. Maximum
```

Arduino/Genuino Uno on COM3

5. Open the serial monitor and set the baud rate



```
servo=o_seral_simple ready
```

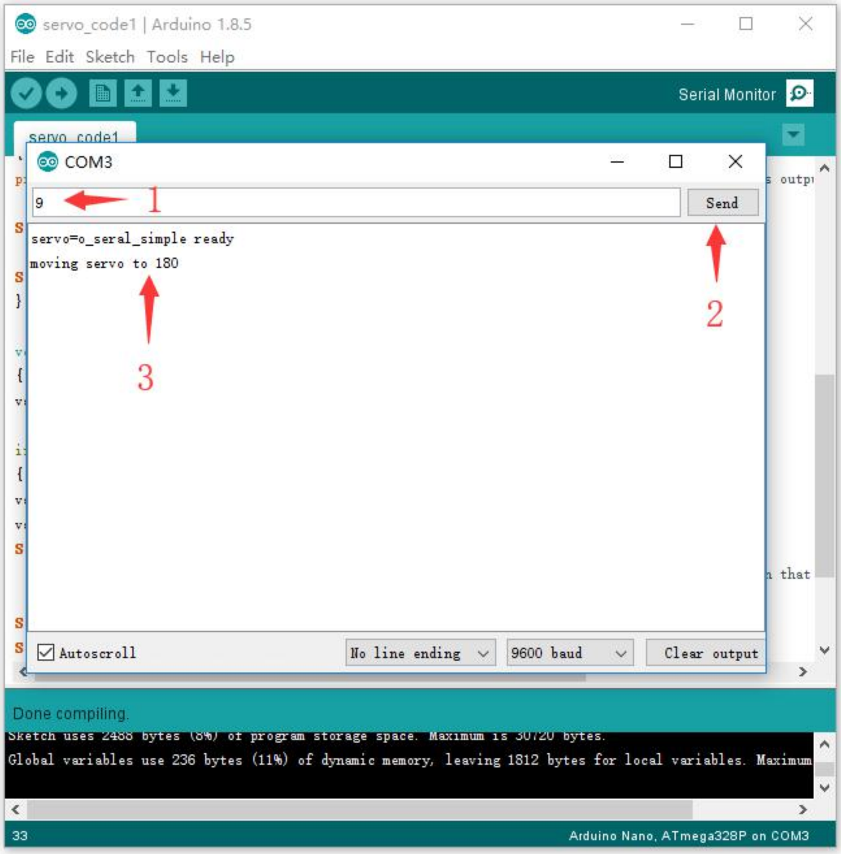
☑ Autoscroll   No line ending   9600 baud

Done compiling.

```
Sketch uses 2488 bytes (8%) of program storage space. Maximum is 30720 bytes.
Global variables use 236 bytes (11%) of dynamic memory, leaving 1812 bytes for local variables. Maximum
```

33   Arduino Nano, ATmega328P on COM3

6. Set the serial port baud rate to 9600, send number 0, 1, 2, 3, 4, 5, 6, 7, 8 or 9 to the V-1 board, Note that the number 0-9 corresponds to the steering angle 0-180 of the servo. For example, send the number 9 and let the servo rotate 180 degrees：


servo_code1 | Arduino 1.8.5
File Edit Sketch Tools Help
Serial Monitor
servo_code1
COM3
9
1
Send
2
3
☑ Autoscroll
No line ending
9600 baud
Clear output
33
Arduino Nano, ATmega328P on COM3

```
servo=o_seral_simple ready
moving servo to 180
```

Done compiling.

```
Sketch uses 2488 bytes (8%) of program storage space. Maximum is 30720 bytes.
Global variables use 236 bytes (11%) of dynamic memory, leaving 1812 bytes for local variables. Maximum
```

Method 2：

Sketch：

```
#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
  }
}
```

Copy the above sketch into the Arduino IDE，copy the serco library to the libraries folder of the Arduino IED，connect the PC to the V-1 board with a USB cable, select the corresponding board type and port in the IDE, upload the sketch to the V-1 board, and you can see that the servo starts to run from 0 to 180 degrees. Then run from 180 degrees to 0 degrees.