Ultrasonic obstacle avoidance(servo)

The purpose of the experiment:

You can make a small labyrinth, put the smart car with the uploaded program in the maze, press the start button of the tail of the car, the LCD screen of the car shows the distance measured by the ultrasonic wave. If the distance from the obstacle is less than 32 cm in front, the servo is turned to the left and right, and the distance between the two sides of the ultrasonic sensor measurement is compared. The car turns to a more spacious direction.

Precautions:

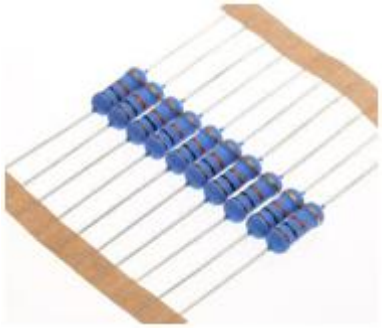
If the LCD is not displayed, use a screwdriver to adjust the adjustable resistor.

If only the ultrasonic obstacle avoidance function is used, the display distance is not required, and the 1602LCD display and the yellow adjustable resistor are not installed.

List of components required for the experiment:

Arduino Smart Car* 1
USB data cable* 1
DuPont line * n
Breadboard* 1
1602 LCD screen* 1
Adjustable resistance* 1
Active buzzer* 1
Button * 1
Ultrasonic sensor*1
10K resistor * 1


REMOVE
SEAL
AFTER
WASHING



Experimental code analysis:

```
//============================yahboom=====================================
//   Intelligent car ultrasonic obstacle avoidance(servo)
//In the program, the number part of the computer is shielded,
//and printing will affect the speed of the car's reaction to obstacles.
//When debugging, you can open the shield content Serial.print
//and print the measured distance.
// The PWM value and delay of the control speed are adjusted,
//but it is still in accordance with the actual conditions
//and the actual quantity of electricity is adjusted.
//=========================================================================
//#include <Servo.h>
#include <LiquidCrystal.h> //Declare the function library of 1602 liquid crystals
//Declare the Arduino digital port connected by the pin of the 1602 LCD,
//8 or 4 line data modes, choose one of them.
//LiquidCrystal lcd(12,11,10,9,8,7,6,5,4,3,2);    //8 data port mode connection statement
LiquidCrystal lcd(3,4,7,8,11,12,13); //4 data port mode connection statement
int Echo = A5;   // Echo(P2.0)
int Trig =A4;   //   Trig(P2.1)
int Front_Distance = 0;
int Left_Distance = 0;
int Right_Distance = 0;
int Left_motor_back=9;      //(IN1)
int Left_motor_go=5;        //(IN2)
int Right_motor_go=6;       //(IN3)
int Right_motor_back=10;    //(IN4)
```

```
int key=A0;//Define the key A0 interface
int beep=A1;//Define the buzzer A1 interface
int servopin=2;//Set the steering gear of the rudder to the digital port 2
int myangle;//Define angle variables
int pulsewidth;//Define of pulse width variable
int val;
void setup()
{
  Serial.begin(9600);     //Initialize the serial port
  //Initialize the motor drive IO for output mode
  pinMode(Left_motor_go,OUTPUT); // PIN 5 (PWM)
  pinMode(Left_motor_back,OUTPUT); // PIN 9 (PWM)
  pinMode(Right_motor_go,OUTPUT);// PIN 6 (PWM)
  pinMode(Right_motor_back,OUTPUT);// PIN 10 (PWM)
  pinMode(key,INPUT);//Define the key interface for the input interface
  pinMode(beep,OUTPUT);
  //Initialization of ultrasonic pins
  pinMode(Echo, INPUT);     // Define of ultrasonic input pin
  pinMode(Trig, OUTPUT);    // Define of ultrasonic output pin
  lcd.begin(16,2);       //Initialization of 1602 liquid crystal working mode
  //Define the 1602 LCD display range of 2 lines and 16 columns
}
//======================The basic action of
car=========================
//void run(int time)
void run()
{
  digitalWrite(Right_motor_go,HIGH);   //right motor go
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,165);//PWM ratio 0~255 speed control,
                          //the difference of left and right wheel slightly increase or
decrease
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH);   //left motor go
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,160);//PWM ratio 0~255 speed control,
```

```
                          //the difference of left and right wheel slightly increase or
decrease
  analogWrite(Left_motor_back,0);
  //delay(time * 100);          //execution time, can be adjusted
}
void brake(int time)
{
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,LOW);
  delay(time * 100);//execution time, can be adjusted
}
void left(int time)         //turn left(left wheel stop,right wheel go)
//void left()         //turn left(left wheel stop,right wheel go)
{
  digitalWrite(Right_motor_go,HIGH);   //right motor go
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,200);
  analogWrite(Right_motor_back,0);     //PWM ratio 0~255 speed control
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,0);      //PWM ratio 0~255 speed control
  delay(time * 100);    //execution time, can be adjusted
}
void spin_left(int time)         //left rotation(left wheel back，right wheel go)
{
  digitalWrite(Right_motor_go,HIGH);    //right motor go
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,150);
  analogWrite(Right_motor_back,0);     //PWM ratio 0~255 speed control
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,HIGH);   //left motor back
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,150);    //PWM ratio 0~255 speed control
```

```
  delay(time * 100);    //execution time, can be adjusted
}
void right(int time)
//void right()        //turn right (right wheel stop,left wheel go)
{
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,0);//PWM ratio 0~255 speed control
  digitalWrite(Left_motor_go,HIGH);//left motor go
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200);
  analogWrite(Left_motor_back,0);//PWM ratio 0~255 speed control
  delay(time * 100);    //execution time, can be adjusted
}
void spin_right(int time)        //right rotation(right wheel back,left wheel go)
{
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,HIGH); //right motor back
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,150);    //PWM ratio 0~255 speed control
  digitalWrite(Left_motor_go,HIGH);     //left motor go
  digitalWrite(Left_motor_back,LOW);
 analogWrite(Left_motor_go,150);
  analogWrite(Left_motor_back,0);//PWM ratio 0~255 speed control
  delay(time * 100);    //execution time, can be adjusted
}
void back(int time)
{
  digitalWrite(Right_motor_go,LOW);   //right motor back
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,200);//PWM ratio 0~255 speed control
  digitalWrite(Left_motor_go,LOW);   //left motor back
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
```

```
  analogWrite(Left_motor_back,200);//PWM ratio 0~255 speed control
  delay(time * 100);      //execution time, can be adjusted
}
//==========================================================
void keysacn()
{
  int val;
  val=digitalRead(key);//Read the value of the port 7 level to the val
  while(!digitalRead(key))//When the key is not pressed, circulate all the time
  {
    val=digitalRead(key);//This sentence can be omitted and the circulate can
run away
  }
  while(digitalRead(key))//When the key is pressed
  {
    delay(10);
    val=digitalRead(key);//Read the value of the port 7 level to the val
    if(val==HIGH)   //Judge whether the key is pressed again
    {
      digitalWrite(beep,HIGH);         //buzzer sound
      while(!digitalRead(key))            //Judge whether the key isreleased
        digitalWrite(beep,LOW);            //buzzer no sound
    }
    else
      digitalWrite(beep,LOW);          //buzzer no sound
  }
}
float Distance_test()       //Measuring the distance ahead
{
  digitalWrite(Trig, LOW);    //Give the trigger pin low level 2us
  delayMicroseconds(2);
 digitalWrite(Trig, HIGH);   //Give the trigger pin high level 10us，at least 10μs
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);     //Give the trigger pin low level Continuouly
 float Fdistance = pulseIn(Echo, HIGH);   //Reading high level time(unit：us)
 Fdistance= Fdistance/58;               //Y meter =（X second *344）/2
```

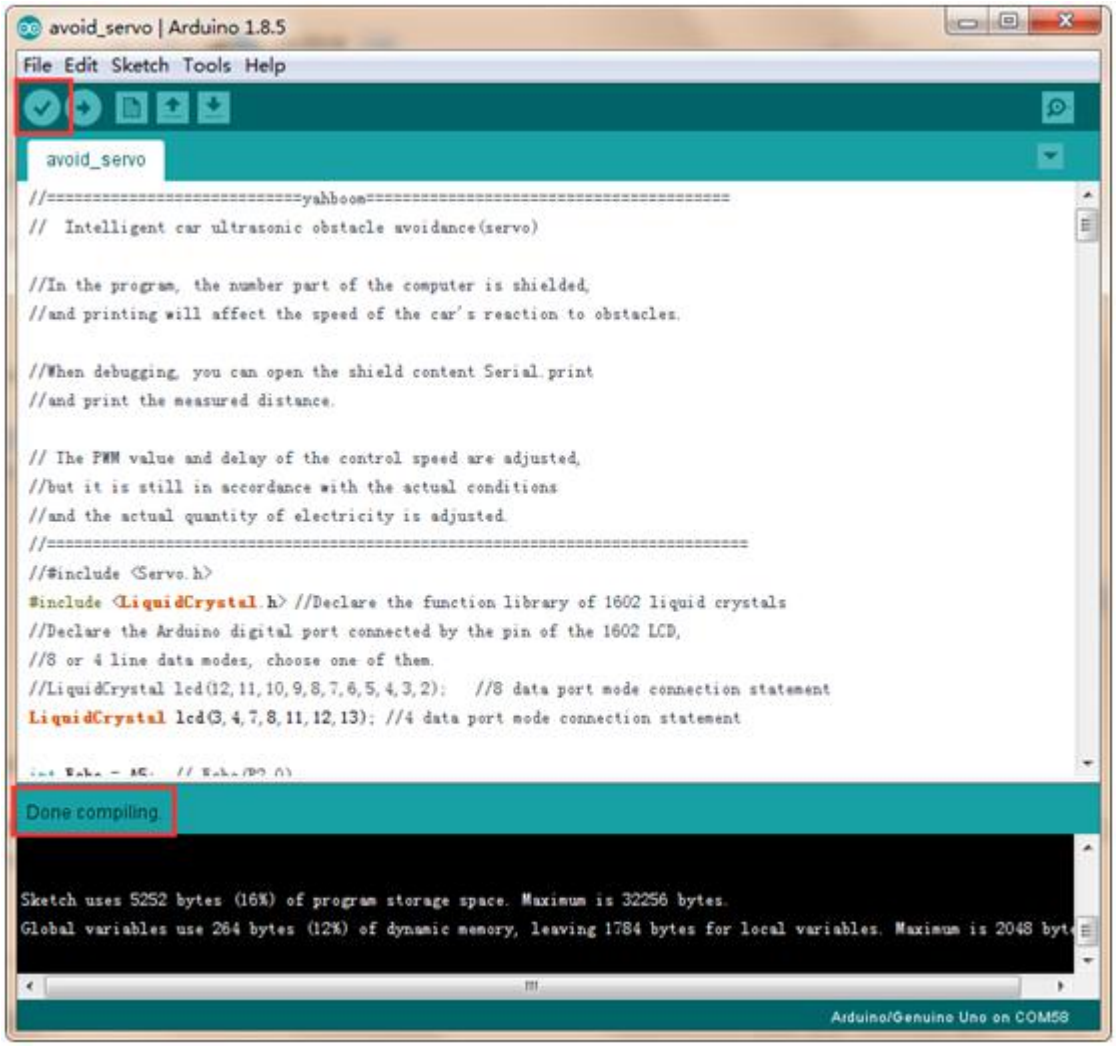
```
  //X second=（ 2*Y meter ）/344 ==》Xsecond =0.0058*Y meter ==》cm = us
/58
  //Serial.print("Distance:");            //Output distance（unit：cm）
  //Serial.println(Fdistance);            //display distanc
  //Distance = Fdistance;
  return Fdistance;
}
void Distance_display(int Distance)
{
  if((2<Distance)&(Distance<400))
  {
    lcd.home();         //Move the cursor back to the upper left corner,
                        //which is the beginning of the output
    lcd.print("     Distance: ");        //display
    lcd.setCursor(6,2);     //Position the cursor in second lines, sixth columns
    lcd.print(Distance);        //display distance
    lcd.print("cm");           //display
  }
  else
  {
    lcd.home();          //Move the cursor back to the upper left corner,
                         //which is the beginning of the output
    lcd.print("!!! Out of range");        //Display beyond distance
  }
  delay(250);
  lcd.clear();
}
void servopulse(int servopin,int myangle)
//Define a pulse function that is used to simulate a PWM value
{
  pulsewidth=(myangle*11)+500;//Turn the angle to 500-2480 of the pulse
width
  digitalWrite(servopin,HIGH);//Set high level of the servopin
  delayMicroseconds(pulsewidth);//The number of microseconds of the
delayed pulse width
  digitalWrite(servopin,LOW);//Set low level of the servopin
```

```
  delay(20-pulsewidth/1000);
}
void front_detection()
{
  //The number of circulate is reduced here to increase the speed of the car's reaction to obstacles.
  for(int i=0;i<=5;i++) //Produce PWM number, equivalent delay to ensure that it can turn to the response angle
  {
    servopulse(servopin,90);
  }
  Front_Distance = Distance_test();
}
void left_detection()
{
  for(int i=0;i<=15;i++)//Produce PWM number, equivalent delay to ensure that it can turn to the response angle
  {
    servopulse(servopin,175);
  }
  Left_Distance = Distance_test();
 //Serial.print("Left_Distance:");      //Output distance（unit:cm）
  //Serial.println(Left_Distance);       //display   distance
}
void right_detection()
{
  for(int i=0;i<=15;i++) //Produce PWM number, equivalent delay to ensure that it can turn to the response angle
  {
    servopulse(servopin,0);
  }
  Right_Distance = Distance_test();
  //Serial.print("Right_Distance:");      //Output distance（unit:cm）
  //Serial.println(Right_Distance);       //display   distance
}
//=============================================================
```

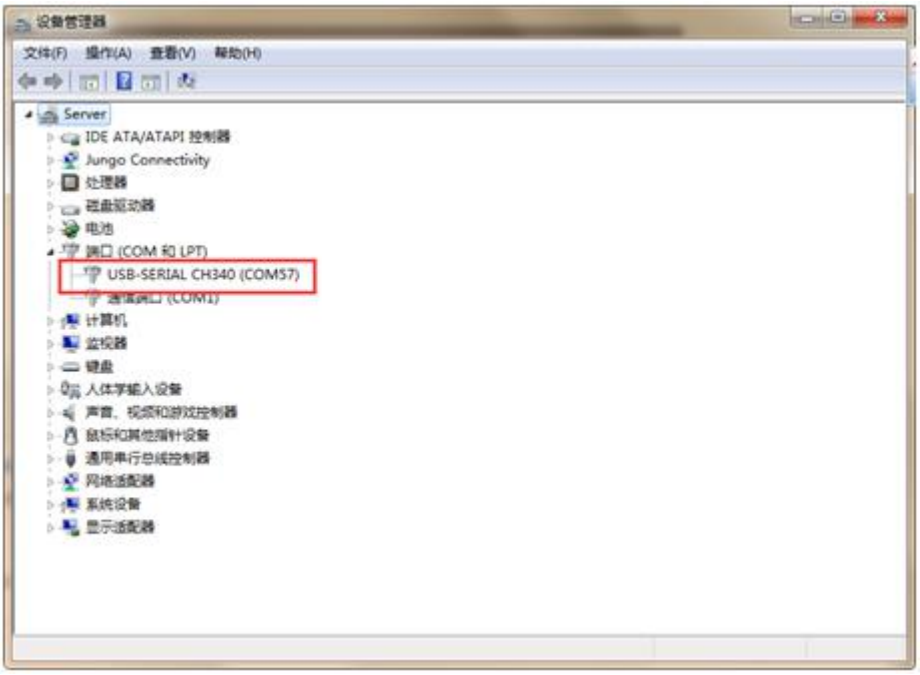
```
void loop()
{
 keysacn();
  while(1)
  {
    front_detection();//Measuring the distance ahead
    if(Front_Distance < 32)//When encounter an obstacle
    {
      back(2);
      brake(2);//Stop for distance measurement
      left_detection();//Measure the distance to the left distance obstacle
      Distance_display(Left_Distance);//LCD screen display distance
      right_detection();//Measure the distance to the right distance obstacle
      Distance_display(Right_Distance);//LCD screen display distance
      if((Left_Distance < 35 ) &&( Right_Distance < 35 ))//When there are
obstacles on both sides of the side
        spin_left(0.7);//rotate and turn around
      else if(Left_Distance > Right_Distance)//The left is more open than the
right.
      {
        left(3);
        brake(1);//brake, stable direction
      }
      else//The right is more open than the left
      {
        right(3);
        brake(1);//brake, stable direction
      }
    }
    else
    {
      run(); //No obstacle, run
    }
  }
}
```

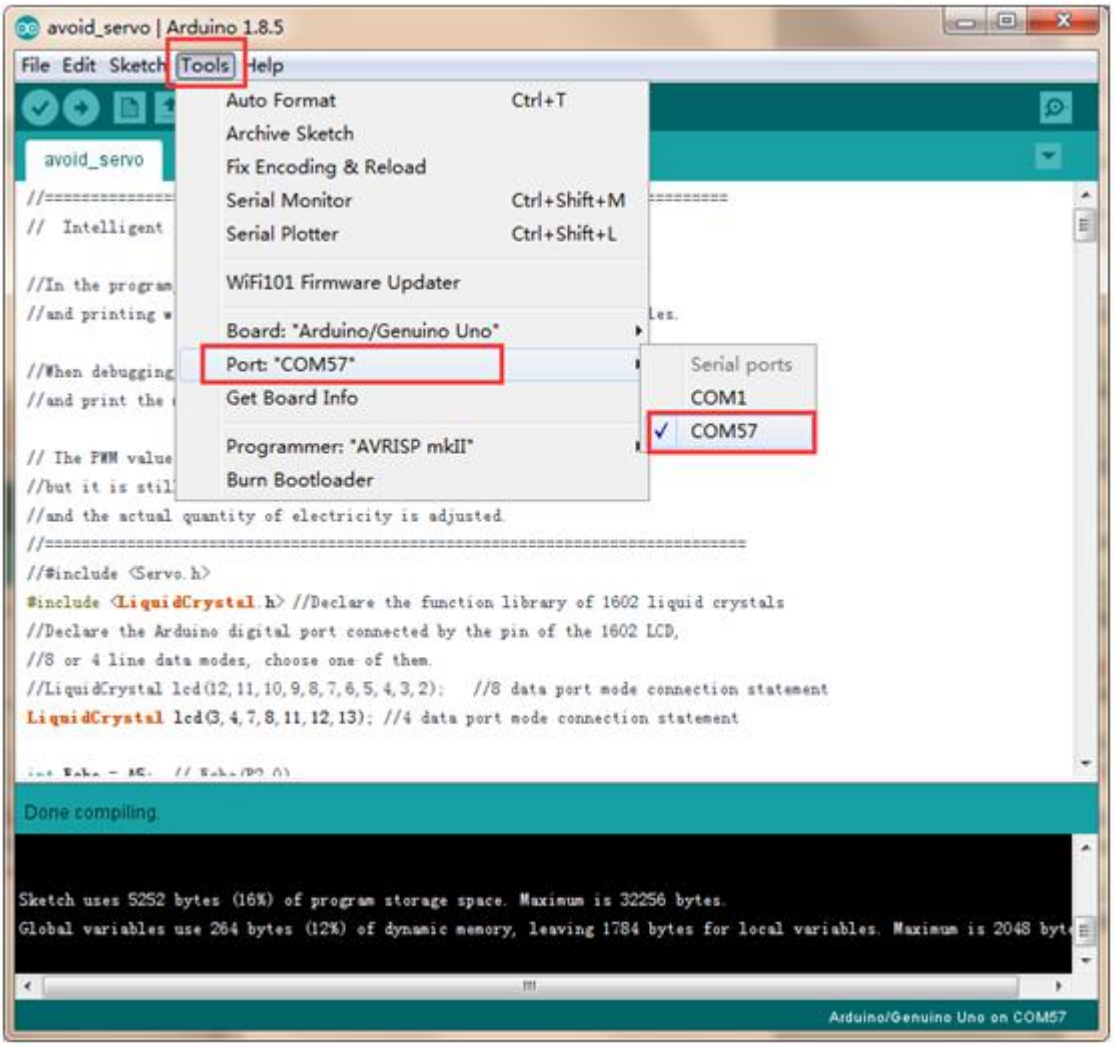
Experimental steps:

1. We need to open the code of this experiment: **avoid_servo.ino,**click“√” under the menu bar to compile the code, and wait for the word "**Done compiling** " in the lower right corner,as shown in the figure below.

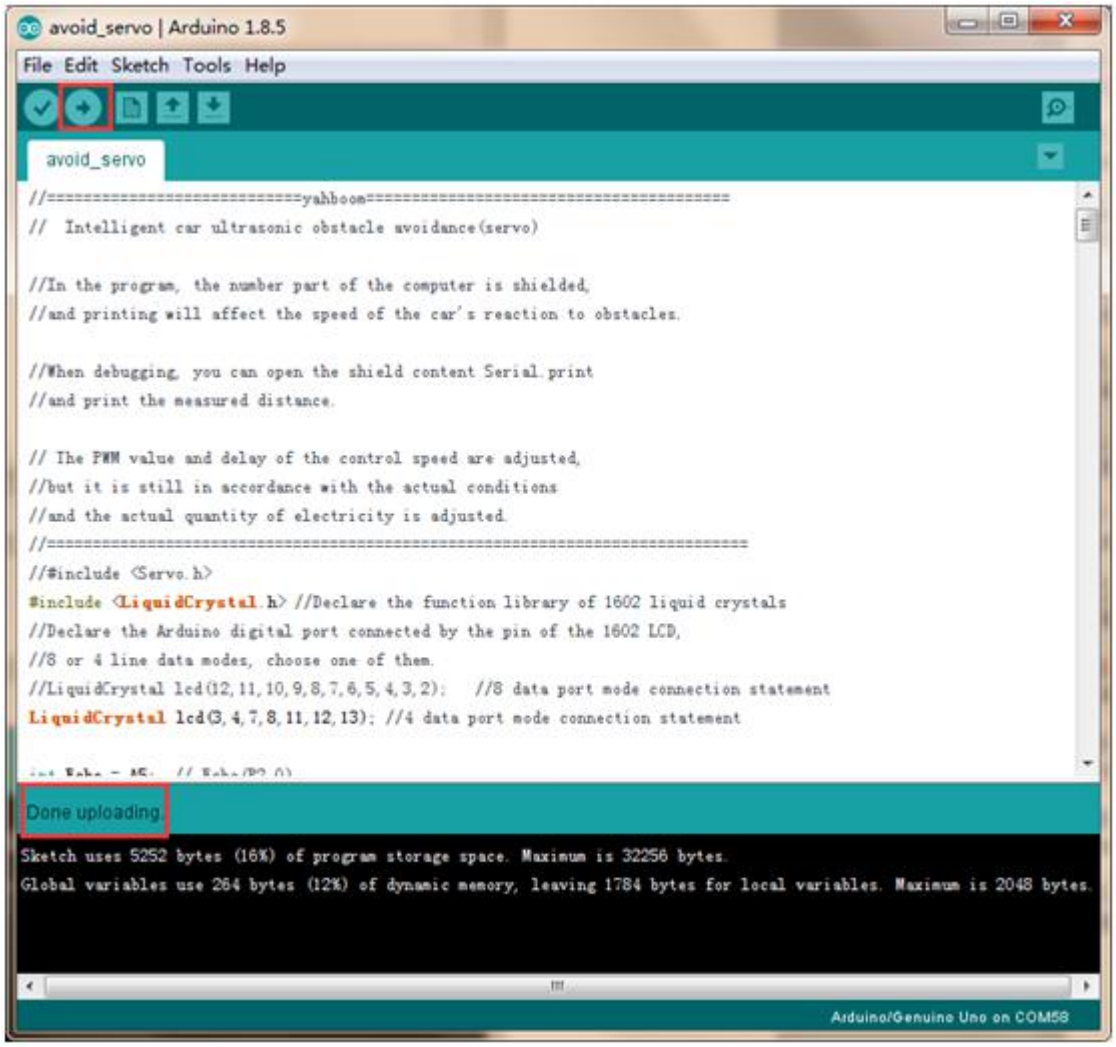


2. In the menu bar of Arduino IDE, we need to select 【Tools】---【Port】---selecting the port that the serial number displayed by the device manager just now, as shown in the figure below.

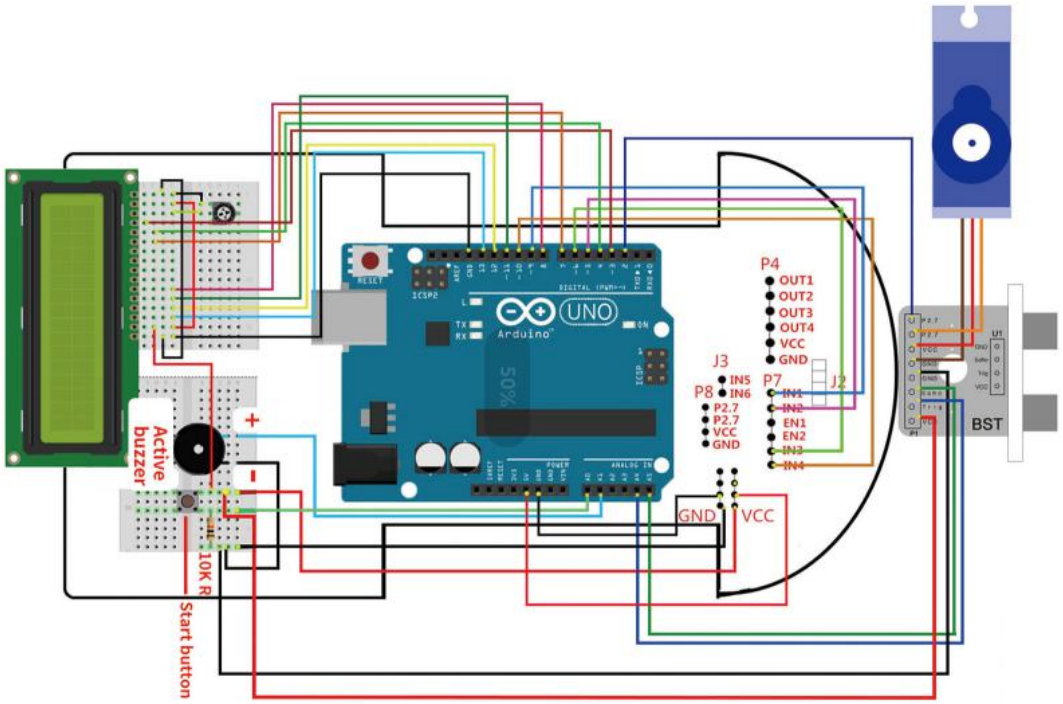

设备管理器
文件(F) 操作(A) 查看(V) 帮助(H)
Server
IDE ATA/ATAPI 控制器
Jungo Connectivity
处理器
磁盘驱动器
电池
端口 (COM 和 LPT)
USB-SERIAL CH340 (COM57)
通信端口 (COM1)
计算机
监视器
键盘
人体学输入设备
声音、视频和游戏控制器
鼠标和其他指针设备
通用串行总线控制器
网络适配器
系统设备
显示适配器


avoid_servo | 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
Board: "Arduino/Genuino Uno"
Port: "COM57"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
COM57
avoid_servo
// Intelligent
//In the program
//and printing
//When debugging
//and print the
// The PWM value
//but it is stil
//and the actual quantity of electricity is adjusted.
//#include <Servo.h>
#include <LiquidCrystal.h> //Declare the function library of 1602 liquid crystals
//Declare the Arduino digital port connected by the pin of the 1602 LCD,
//8 or 4 line data modes, choose one of them.
//LiquidCrystal lcd(12,11,10,9,8,7,6,5,4,3,2); //8 data port mode connection statement
LiquidCrystal lcd(3,4,7,8,11,12,13); //4 data port mode connection statement
Done compiling.
Sketch uses 5252 bytes (16%) of program storage space. Maximum is 32256 bytes.
Global variables use 264 bytes (12%) of dynamic memory, leaving 1784 bytes for local variables. Maximum is 2048 byt
Arduino/Genuino Uno on COM57

3. After the selection is completed, you need to click “→”under the menu bar to upload the code to the Arduino UNO board. When the word “**Done uploading**” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



4.Please wire the Smart Car as shown below.



5.Put the smart car in the labyrinth, press the start button of the tail of the car, the LCD screen of the car displays the distance measured by the ultrasonic sensor, and avoid obstacles in the labyrinth.