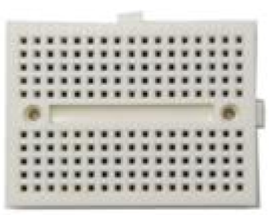
13 tracking_ultrasonic

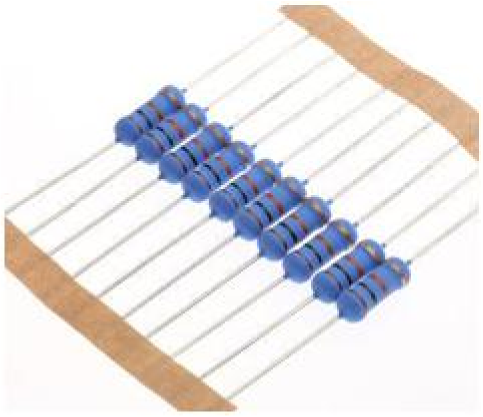
The purpose of the experiment:

This experiment is a 2in1 comprehensive experiment. The car can detect the obstacles while tracking. When encounteringan obstacle, the car stops waiting in place. After clearing obstacle, the car continues to track.

List of components required for the experiment:

Arduino Smart Car* 1
USB cable* 1
DuPont Line* 13
Breadboard* 1
Ultrasonic sensor* 1
Active buzzer* 1
Button * 1
10K resistor * 1





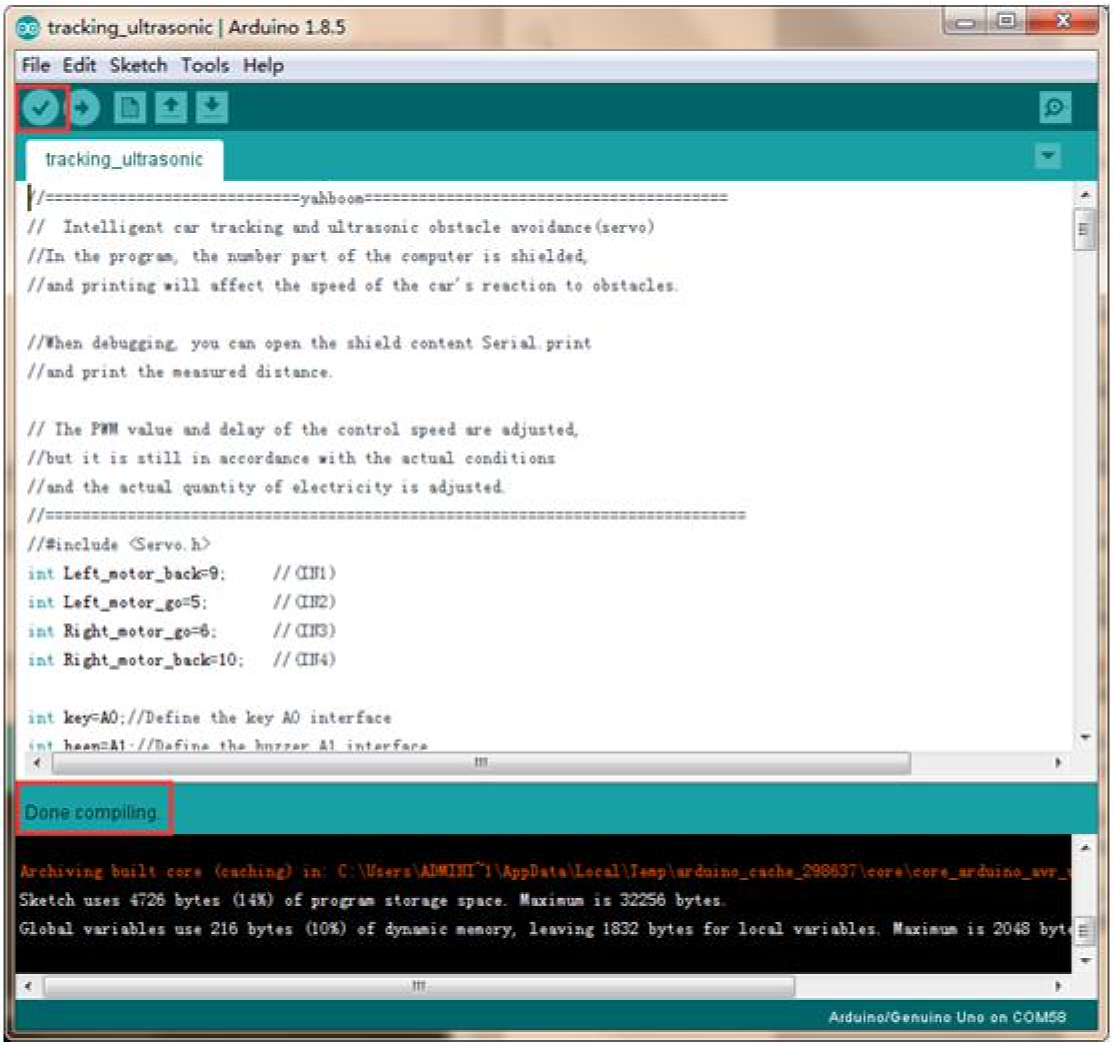
Experimental code analysis:

```
//============================yahboom=======================
=================
//   Intelligent car tracking and 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>
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
```

```
const int SensorRight = A2;         //Right tracking infrared sensor(P3.2 OUT1)
const int SensorLeft = A3;         //Left tracking infrared sensor(P3.3 OUT2)
int SL;     //Left tracking infrared sensor state
int SR;     //Right tracking infrared sensor state
int Echo = A5;   // Echo(P2.0)
int Trig =A4;   //   Trig(P2.1)
int Distance = 0;
void setup()
{
  //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);
  pinMode(SensorRight, INPUT); // Define of ultrasonic input pin
  pinMode(SensorLeft, INPUT); // Define of ultrasonic output pin
  Serial.begin(9600);        //Initialization of 1602 liquid crystal working mode
  pinMode(Echo, INPUT);       // Define of ultrasonic input pin
  pinMode(Trig, OUTPUT);      // Define of ultrasonic output pin
}
//======================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,125);//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,125);//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)
void brake()
{
  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()
{
  digitalWrite(Right_motor_go,HIGH);    //right motor go
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,125);
  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,200);
  analogWrite(Right_motor_back,0);//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 right(int time)        //turn right (right wheel stop,left wheel go)
void right()
{
  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,125);
  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);    //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,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 back(int time)
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,150);//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,150);//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
  }
}
void 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 distance
  Distance = Fdistance;
}
void loop()
{
  keysacn();
  while(1)
  {
    Distance_test();// Measuring the distance ahead
   //There is a signal is LOW , no signal is HIGH
  SR = digitalRead(SensorRight);
  //There is a signal that in the white area the L3 is bright on the car floor;
  // no signal indicates that on the black line and the L3 is extinguishing on the
car floor.
  SL = digitalRead(SensorLeft);
   //There is a signal that in the white area the L2 is bright on the car floor;
  // no signal indicates that on the black line and the L2 is extinguishing on the
car floor.
  if((Distance < 10)||(Distance > 400))
  //If the distance is less than 10cm or greater than 400cm (less than 2cm
value is also greater than 400),
  //the distance can be adjusted by yourself, otherwise the runway is more
open, and the condition of Distance > 400 can be removed.
  brake();
  else
  {
  if (SL == LOW&&SR==LOW)
    run();   //Call run function
  else if (SL == HIGH & SR == LOW)
  //Left tracking infrared sensor signal is detected,the car deviates from  track,
turn left
    left();
  else if (SR == HIGH & SL == LOW)
//Right tracking infrared sensor signal is detected,the car deviates from  track,
turn right
    right();
  else
  brake();
  }
  }
}
```

Experimental steps:

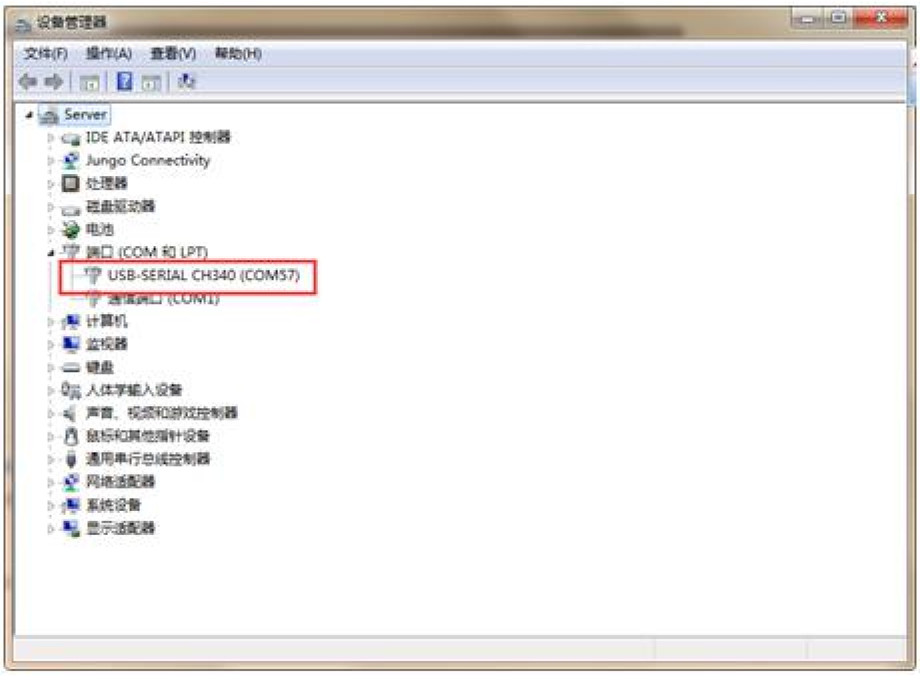
1. We need to open the code of this experiment: **tracking_ultrasonic.ino,**click " √ " under the menu bar to compile the code, and wait for the word "**Done compiling** " in the lower right corner,as shown in the figure below.

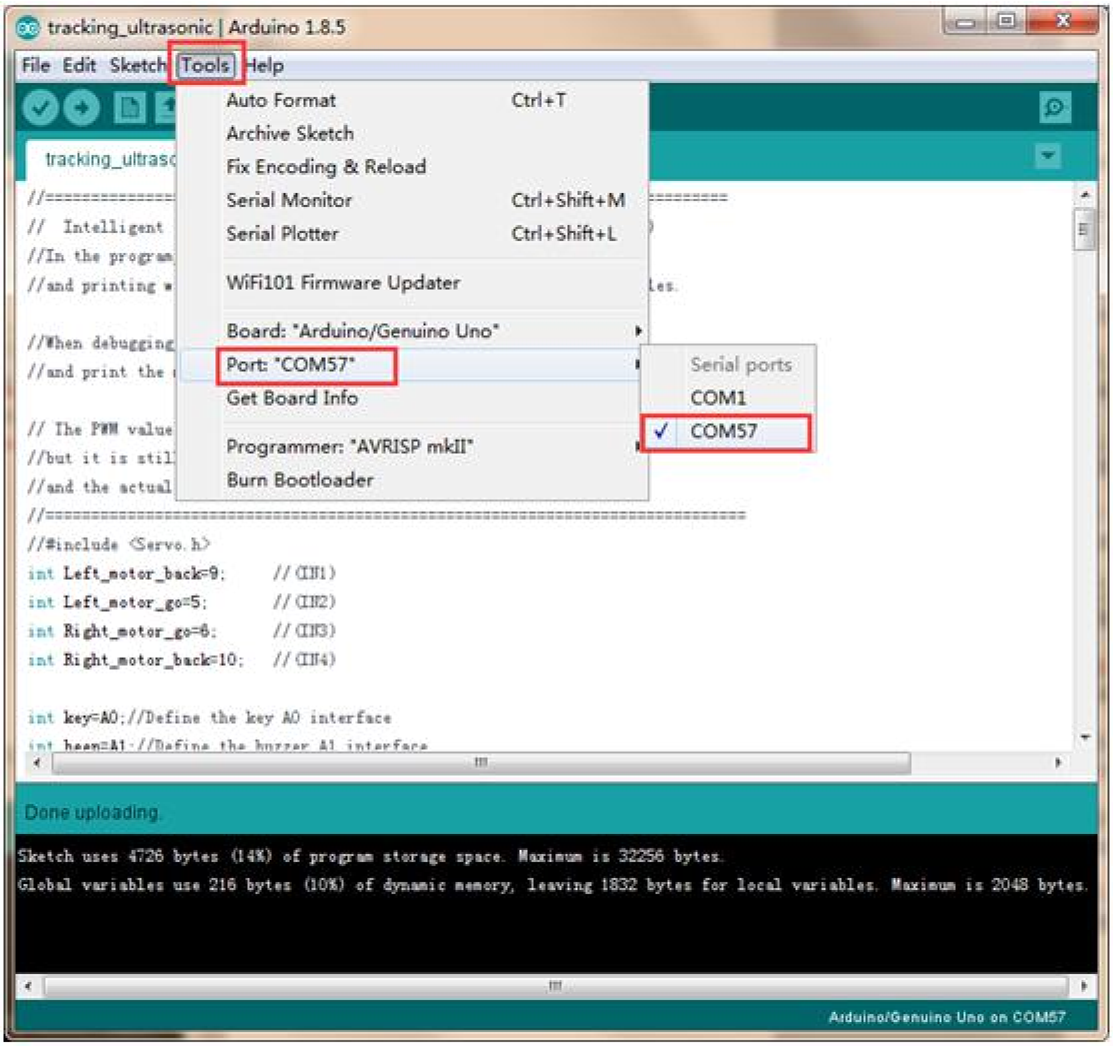

tracking_ultrasonic | Arduino 1.8.5
File Edit Sketch Tools Help
tracking_ultrasonic
Done compiling.
Arduino/Genuino Uno on COM58

```
//===========================yahboom=========================================
//  Intelligent car tracking and 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>
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
```

```
Archiving built core (caching) in: C:\Users\ADMINI~1\AppData\Local\Temp\arduino_cache_298637\core\core_arduino_avr_
Sketch uses 4726 bytes (14%) of program storage space. Maximum is 32256 bytes.
Global variables use 216 bytes (10%) of dynamic memory, leaving 1832 bytes for local variables. Maximum is 2048 byt
```

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.



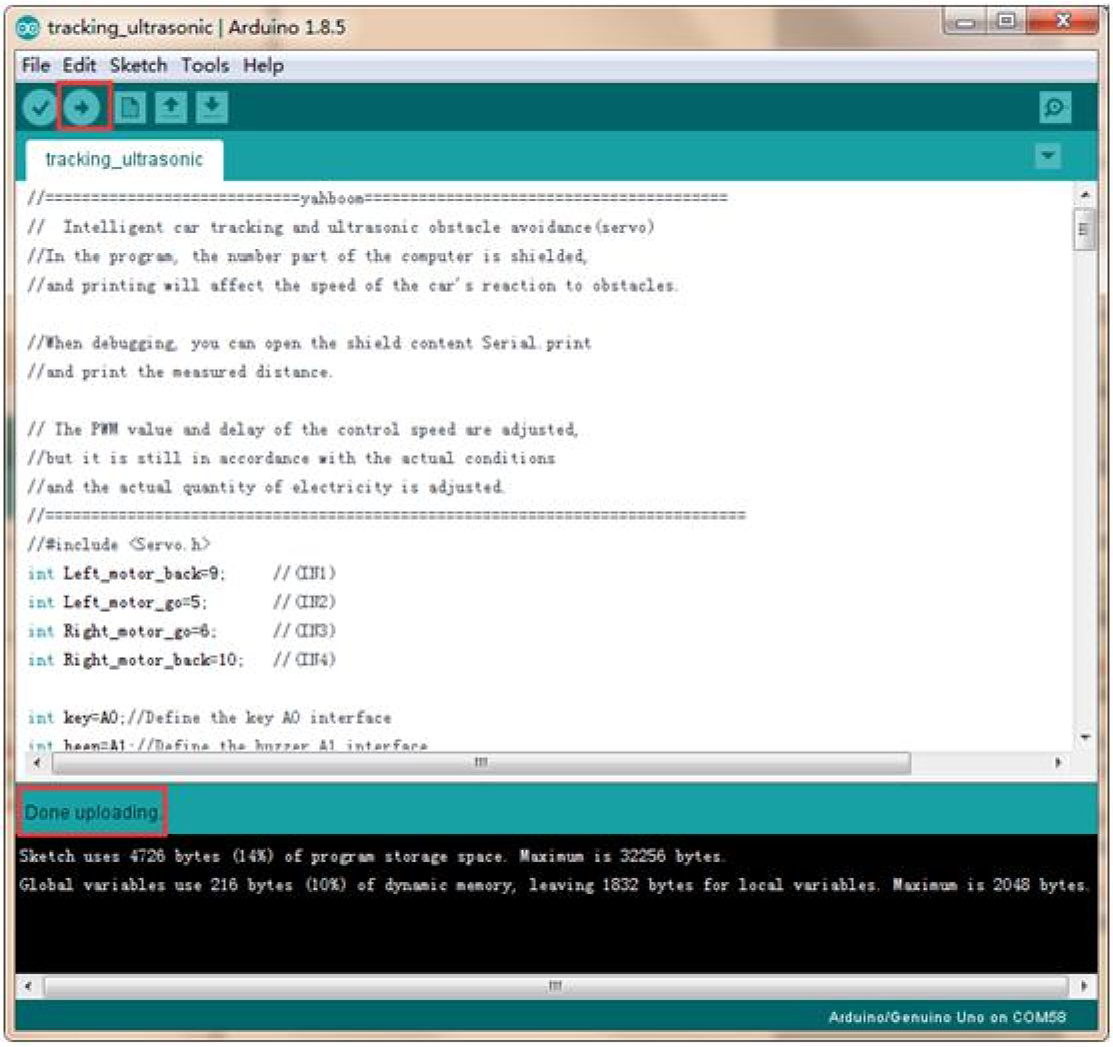

tracking_ultrasonic | Arduino 1.8.5
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM57"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
✓ COM57
tracking_ultrasc
Arduino/Genuino Uno on COM57

```
//==============                                      =========
//  Intelligent
//In the program
//and printing                                        tes.

//When debugging
//and print the

// The PWM value
//but it is stil
//and the actual
//===========================================================================
//#include <Servo.h>
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
```

Done uploading.

```
Sketch uses 4726 bytes (14%) of program storage space. Maximum is 32256 bytes.
Global variables use 216 bytes (10%) of dynamic memory, leaving 1832 bytes for local variables. Maximum is 2048 bytes.
```

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.

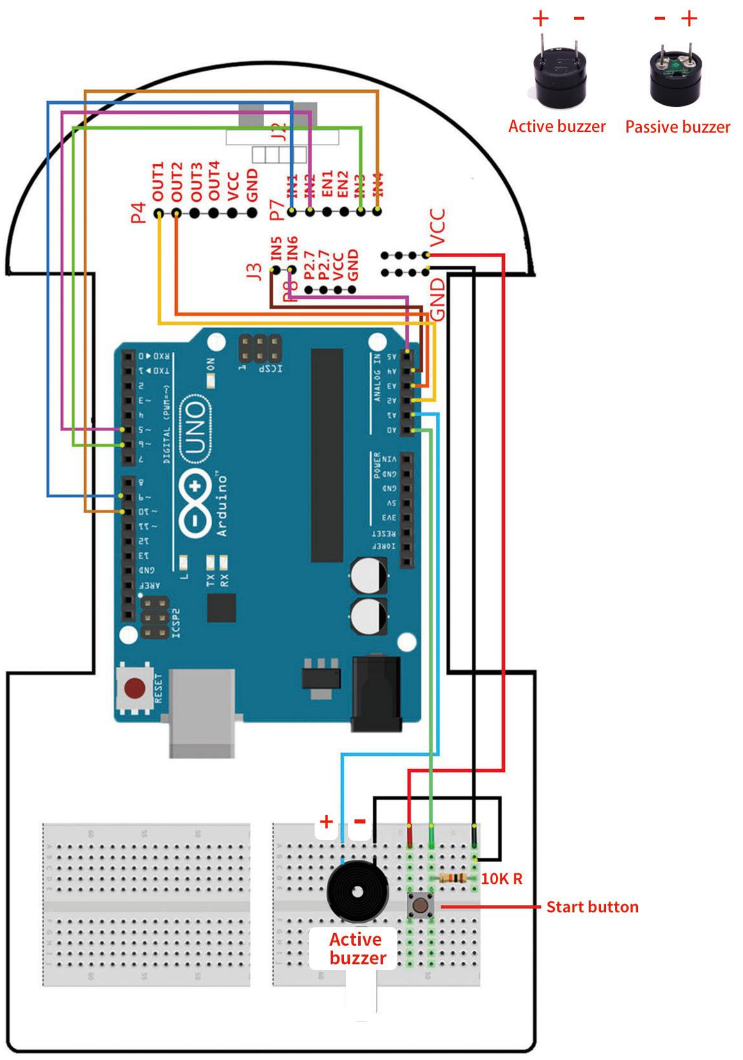

tracking_ultrasonic | Arduino 1.8.5
File Edit Sketch Tools Help
tracking_ultrasonic
Done uploading.
Arduino/Genuino Uno on COM58

```
//==========================yahboom=======================================
//  Intelligent car tracking and 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>
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
```

```
Sketch uses 4726 bytes (14%) of program storage space. Maximum is 32256 bytes.
Global variables use 216 bytes (10%) of dynamic memory, leaving 1832 bytes for local variables. Maximum is 2048 bytes.
```

4.Please wire the Smart Car as shown below.

4.7 Tracking & Ultrasonic wiring diagram



Note: At the J2 slot, insert the ultrasonic sensor as picture.

This experiment is a 2in1 comprehensive experiment. The car can detect the obstacles while tracking. When encountering an obstacle, the car stops waiting in place. After clearing obstacle, the car continues to track.

5.Place the smart car on the tracking track and press the start button. In the process of tracing, the smart car detects obstacles in the trajectory. When the obstacle car is encountered, it stops waiting in place, and after the obstacle is cleared, the car continues to trace.