

2019

Yahboom Arduino Robot Car

Arduino IDE Programming Tutorials



C programming

1. Advance	2
2. Car Run	7
3. ColorfulRun	12
4. KeysacnStartCar	18
5. Tracking	25
6. Infrared obstacle avoidance obstacle - basic	33
7. Infrared obstacle avoidance - follow	41
8. Infrared obstacle avoidance with backdrop	49
9. Ultrasonic distance measurement	57
10. Ultrasonic obstacle avoidance (no servo)	62
11. Ultrasonic obstacle avoidance(servo)	71
12. Infrared remote control	81
13. Tracking ultrasonic	90
14. Bluetooth control	98

1- Advance

The purpose of the experiment:

Assemble the Smart Car according to the instructions and install two 14500 rechargeable li batteries. Wire the drawing at the end of the article and upload the program advance.ino. Turn on the power switch at the rear of the Smart Car, and the Smart Car pauses for 0.5 seconds before starting.

List of components required for the experiment:

Arduino Smart Car *1

USB cable *1



Experimental code analysis:

```
int Left_motor_back=9;    //IN1
int Left_motor_go=5;     //IN2
int Right_motor_go=6;    // IN3
int Right_motor_back=10; //IN4
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)
}
void run(int time)
{
  digitalWrite(Right_motor_go,HIGH);
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,200); //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);
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200); //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
}
```

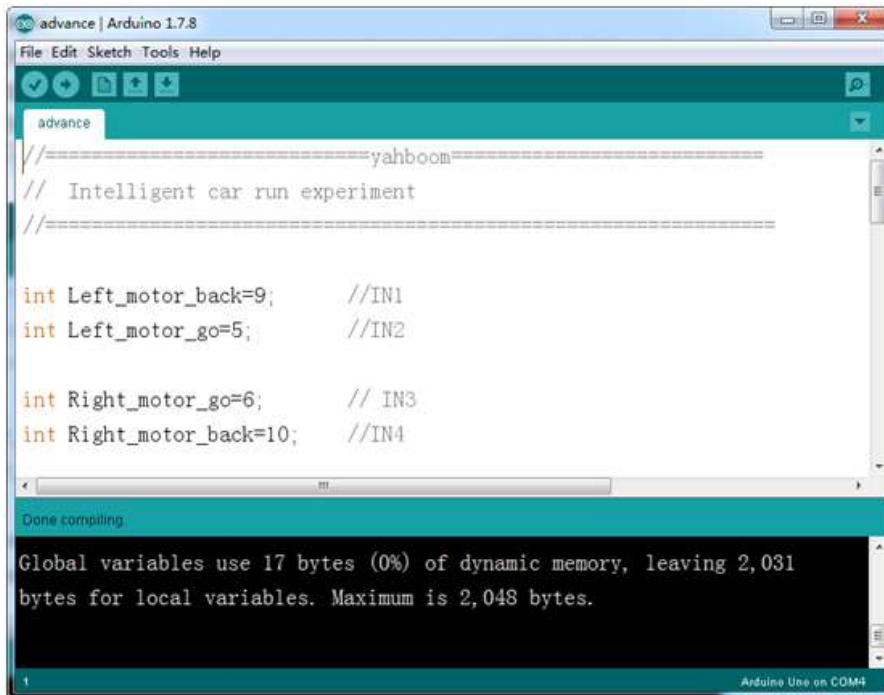
```

void loop()
{
  delay(500);
  run(10);
}

```

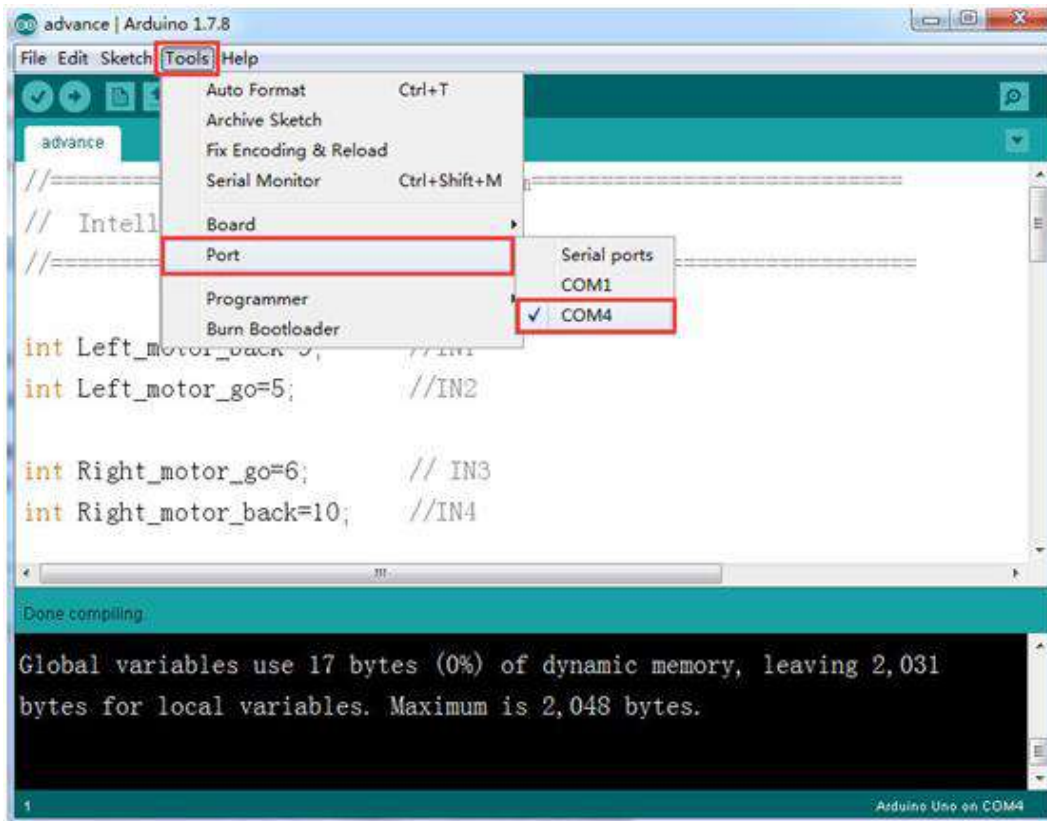
Experimental steps:

1. We need to open the code of this experiment: **advance.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.

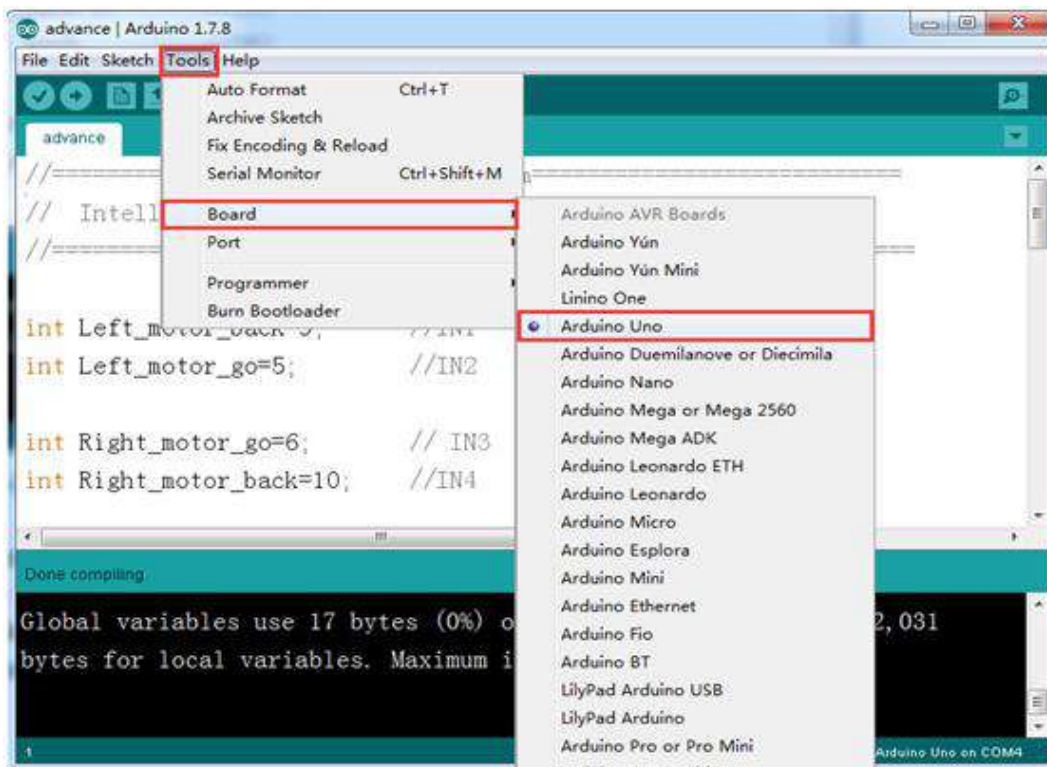


2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below.





3. Click [Tools]---[Board]---Select Arduino Uno as shown below.



4. 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.

```
advance | Arduino 1.7.8
File Edit Sketch Tools Help
advance
//=====yahboom=====
// Intelligent car run experiment
//=====

int Left_motor_back=9;    //IN1
int Left_motor_go=5;     //IN2

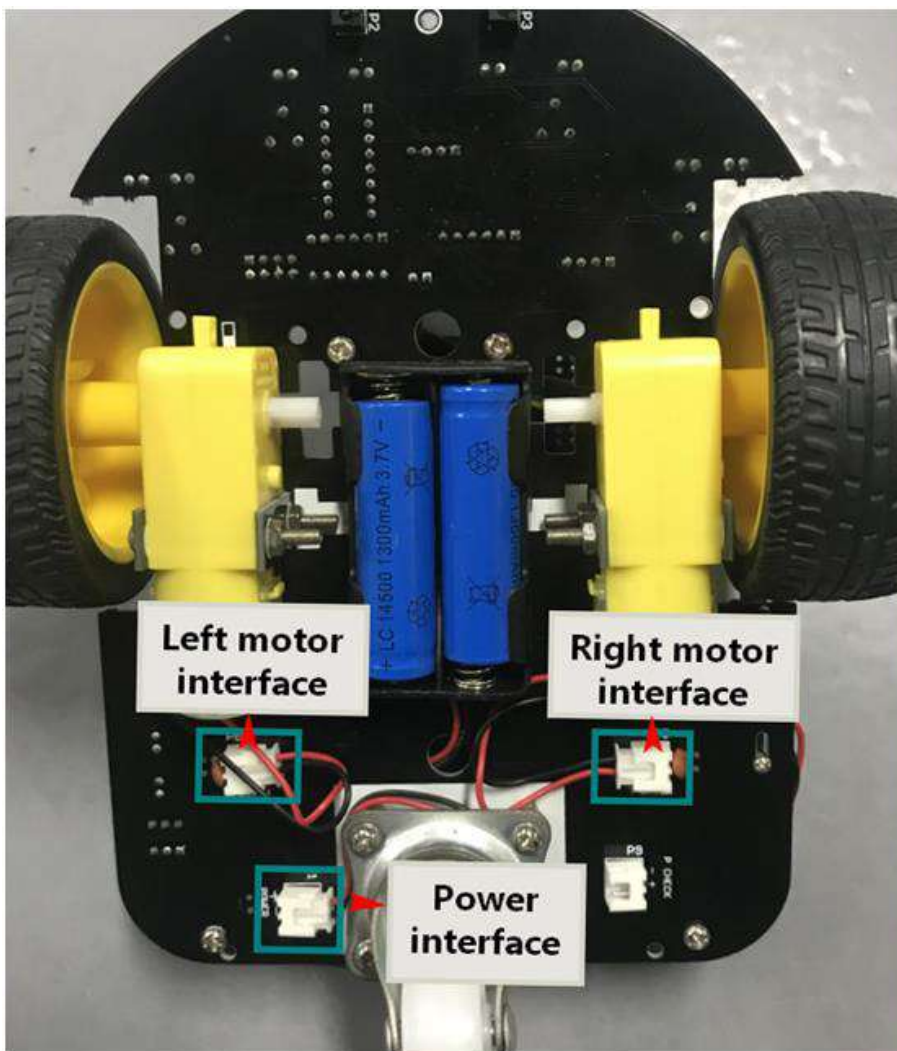
int Right_motor_go=6;    // IN3
int Right_motor_back=10; //IN4

Done uploading.

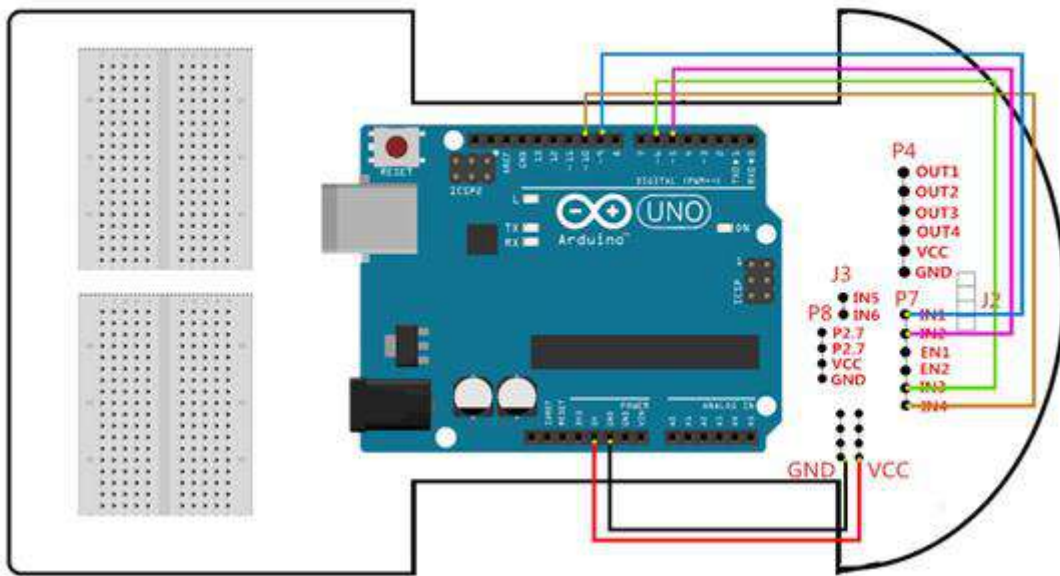
Global variables use 17 bytes (0%) of dynamic memory, leaving 2,031 bytes
for local variables. Maximum is 2,048 bytes.

Arduino Uno on COM43
```

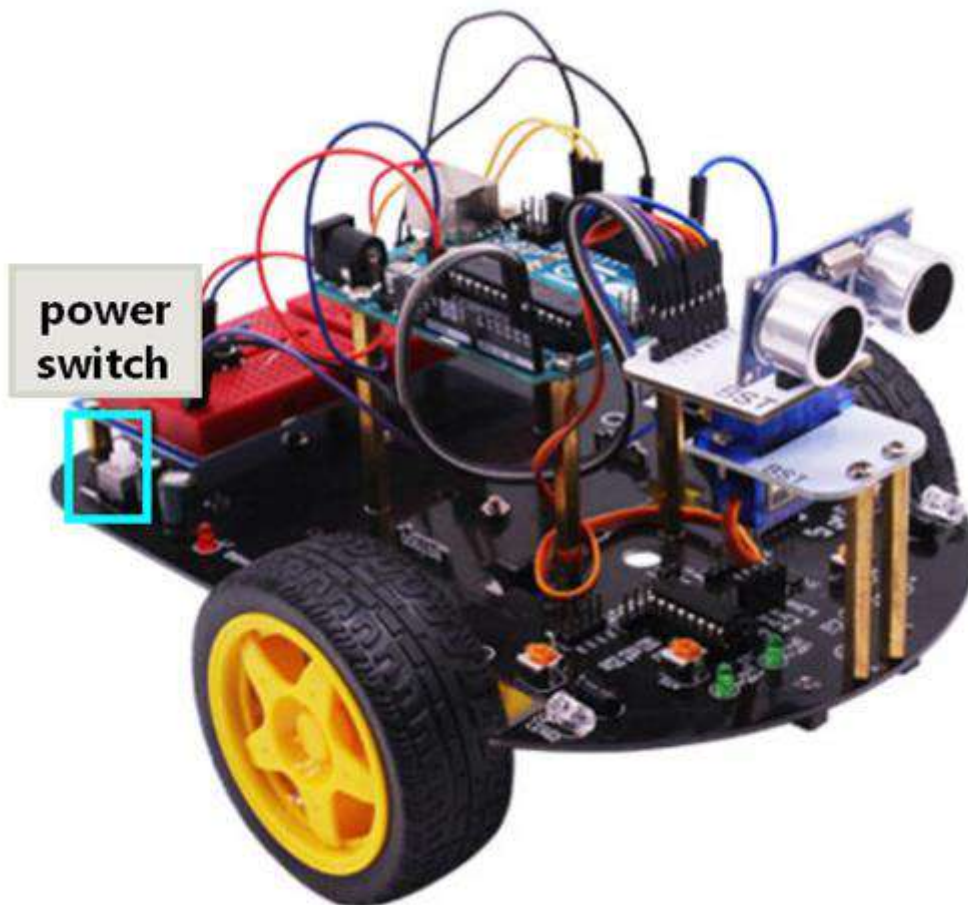
5. Please wire the Smart Car as shown below.



Motor drive wiring diagram of intelligent car



6. After the smart car is connected to the line and the program is uploaded, the power switch at the rear of the smart car is turned on, and the smart car stops after 0.5 seconds.



2- Car Run

The purpose of the experiment:

As with the previous experimental project, after uploading the program, open the power switch at the rear of the smart car and pause for 2 seconds, the Smart Car starts to move backward, forward, left turn, right turn, rotate right, rotate left.

List of components required for the experiment:

Arduino Smart Car *1

USB cable *1



Experimental code analysis:

```
int Left_motor_back=9;    //(IN1)
int Left_motor_go=5;     //(IN2)
int Right_motor_go=6;    //(IN3)
int Right_motor_back=10; //(IN4)
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)
}
void run(int time)
{
  digitalWrite(Right_motor_go,HIGH);
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,200); //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);
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200); //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 (The left wheel stop, the right wheel run)
{
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 stop
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)    //Turn left in place(The left wheel back, the right wheel run)
{
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(The right wheel stop, the left wheel run)
{
digitalWrite(Right_motor_go,LOW); //right motor stop
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)    //turn right in place(The right wheel back, the left wheel run)
{
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
}

```

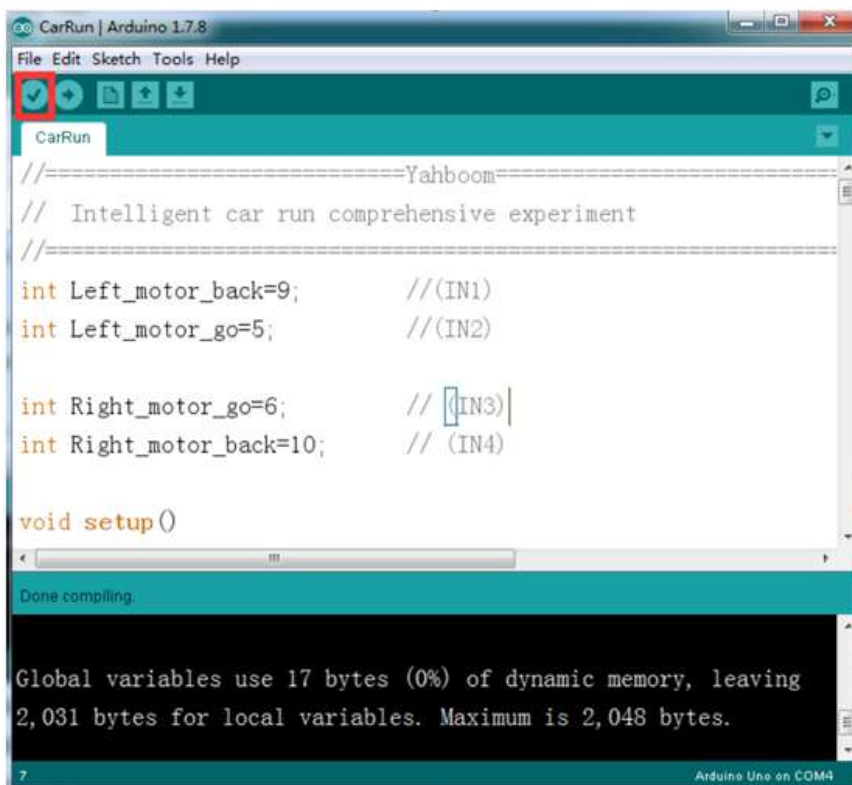
```

}
void back(int time)      //car back
{
  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 loop()
{
  delay(2000); //Start up after delay 2s
  back(10); //back 1s
  brake(5); //stop 0.5s
  run(10); //run 1s
  brake(5); //stop 0.5s
  left(10); //turn left 1s
  right(10); //turn right 1s
  spin_right(20); //turn left in place 2s
  spin_left(20); //turn right in place 2s
  brake(5); //stop
}

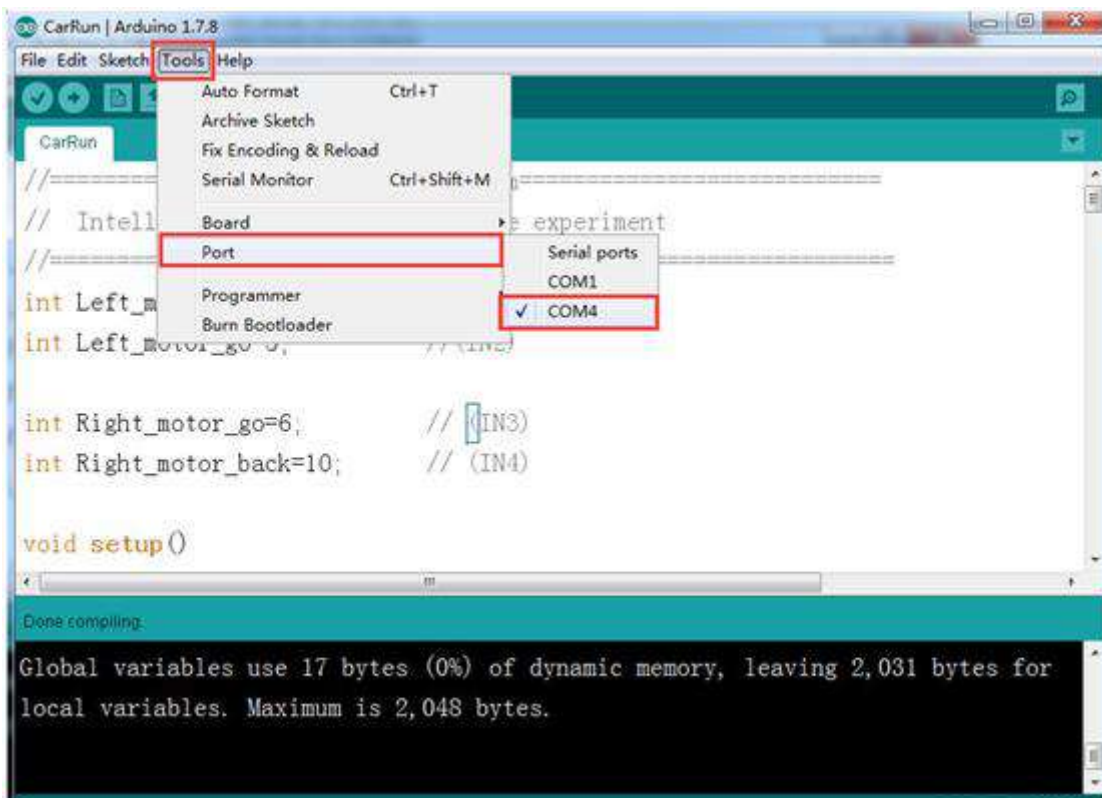
```

Experimental steps:

1. We need to open the code of this experiment: **CarRun.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.



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.

```

CarRun | Arduino 1.7.8
File Edit Sketch Tools Help
CarRun
//-----Yahboom-----
// Intelligent car run comprehensive experiment
//-----

int Left_motor_back=9;    //(IN1)
int Left_motor_go=5;     //(IN2)

int Right_motor_go=6;    //(IN3)
int Right_motor_back=10; //(IN4)

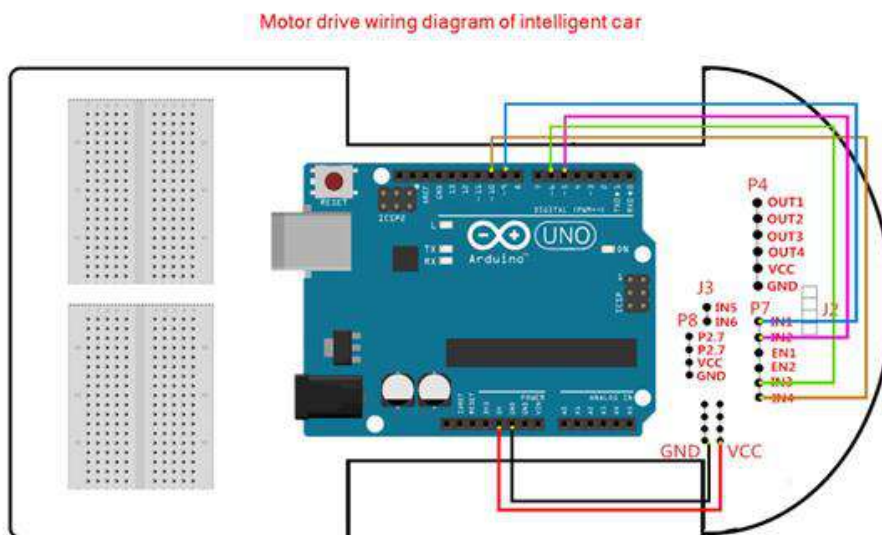
void setup()

Done uploading.

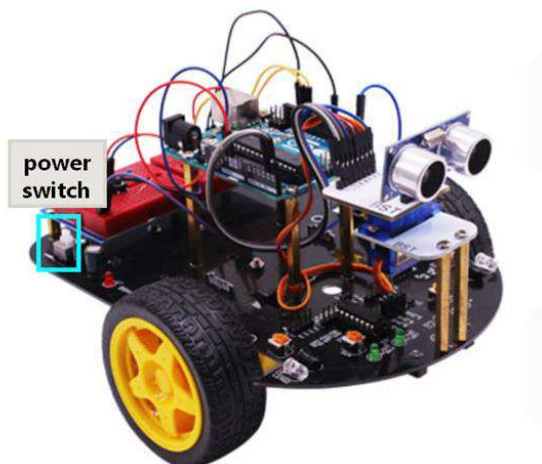
Global variables use 17 bytes (0%) of dynamic memory, leaving 2,031 bytes for
local variables. Maximum is 2,048 bytes.

Arduino Uno on COM43
  
```

4. Please wire the Smart Car as shown below.



5. Unplug the USB cable, place the smart car in a wide area, turn on the power switch, and the smart car pauses for two seconds and then starts moving backwards, moving backwards, turning left, turning right, etc.



3- ColorfulRun

The purpose of the experiment:

As with the previous experimental project, after uploading the program, turn on the power switch on the back of the smart car and pause for 2 seconds, the smart car starts the specified fancy action.

List of components required for the experiment:

Arduino Smart Car *1

USB cable *1



Experimental code analysis:

```
//=====yahboom=====
// Intelligent car colorfulrun comprehensive experiment
//=====
int Left_motor_back=9;    //(IN1)
int Left_motor_go=5;     //(IN2)
int Right_motor_go=6;    //(IN3)
int Right_motor_back=10; //(IN4)
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)
}
void run(int time) //car run
{
  digitalWrite(Right_motor_go,HIGH);
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,200); //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);
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200); //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)    //car stop
{
  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)
{
  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 wheel back
  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 wheel 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)
{
  digitalWrite(Right_motor_go,LOW); //right motor back
  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); //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) //back
{
digitalWrite(Right_motor_go,LOW); //right wheel 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 wheel 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 loop()
{
int i;
delay(2000); //Start up after delay 2S
run(10);
back(10);
brake(5);
for(i=0;i<5;i++)
{
run(10); //5 steps in the discontinuous advance of the car
brake(1);
}
for(i=0;i<5;i++)
{
back(10); //5 steps in the discontinuous back of the car
brake(1);
}
for(i=0;i<5;i++)
{
left(10); //Large sets of small bend left rotation continuously
spin_left(5);
}
for(i=0;i<5;i++)
{
right(10); //Large sets of small bend right rotation continuously
spin_right(5);
}
for(i=0;i<10;i++)
{
right(1); //turn right intermittently
brake(1);
}
for(i=0;i<10;i++)
{

```

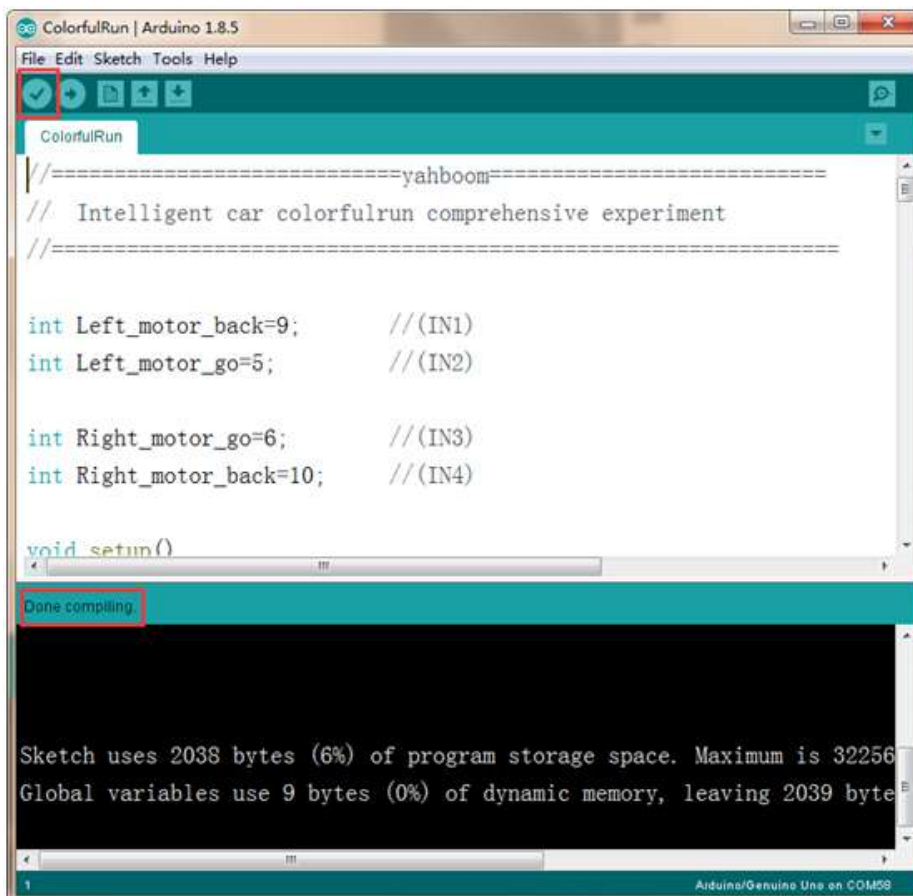
```

    left(1);//turn left intermittently
    brake(1);
}
for(i=0;i<10;i++)
{
    left(3);//Go S - shaped ways
    right(3);
}
for(i=0;i<10;i++)
{
    spin_left(3);//left rotation intermittently
    brake(3);
}
for(i=0;i<10;i++)
{
    spin_right(3);//right rotation intermittently
    brake(3);
}
}
}

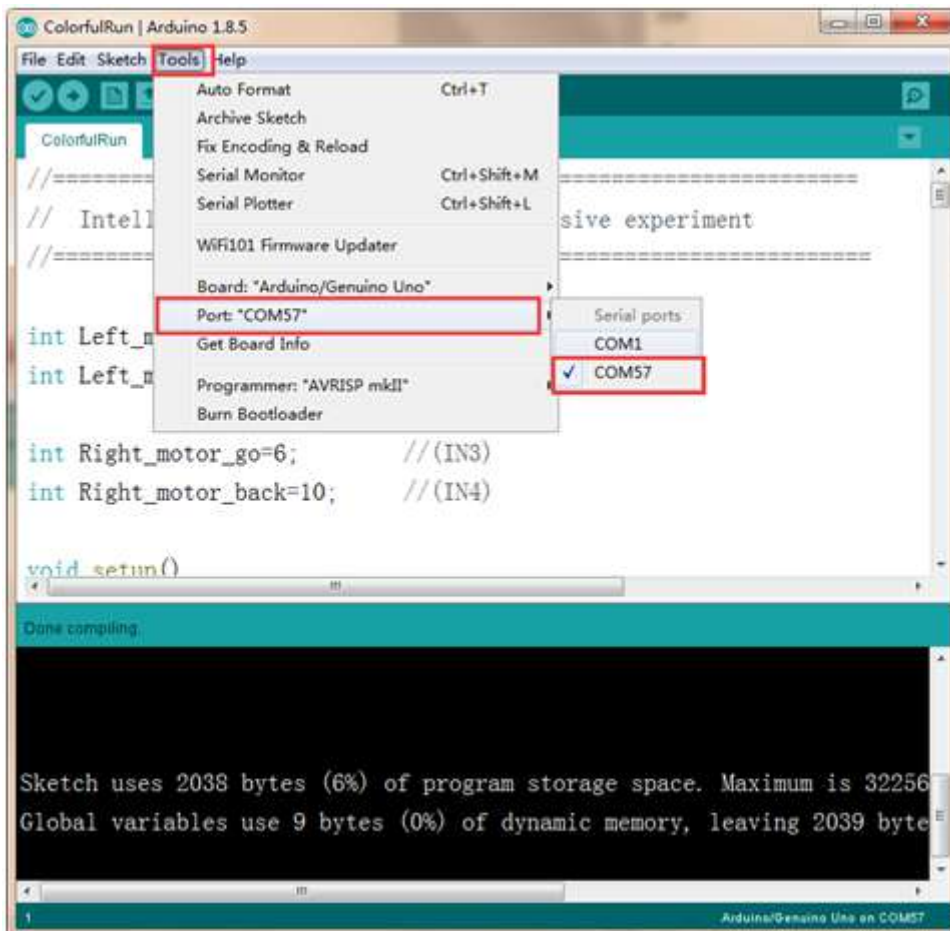
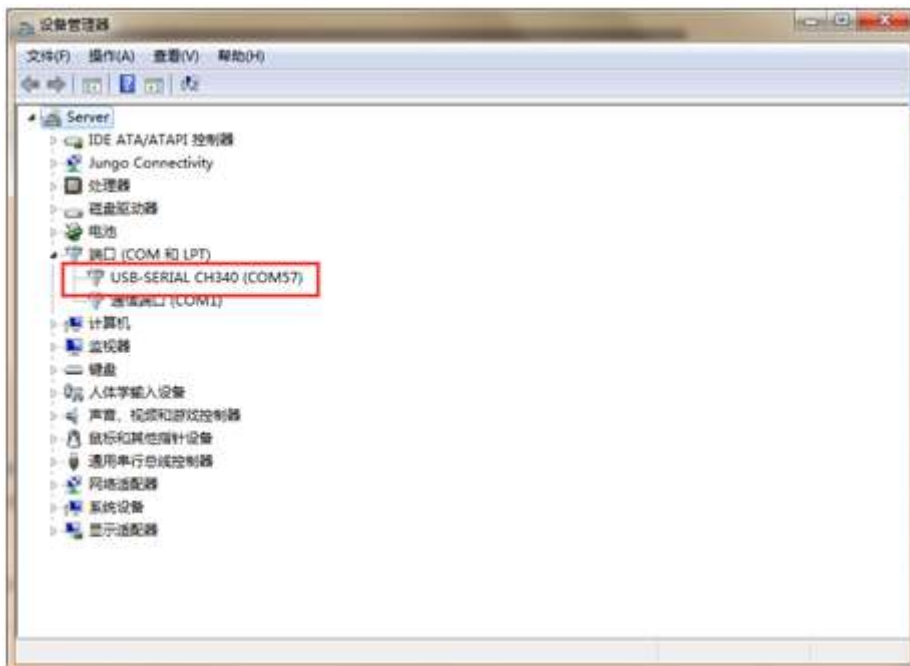
```

Experimental steps:

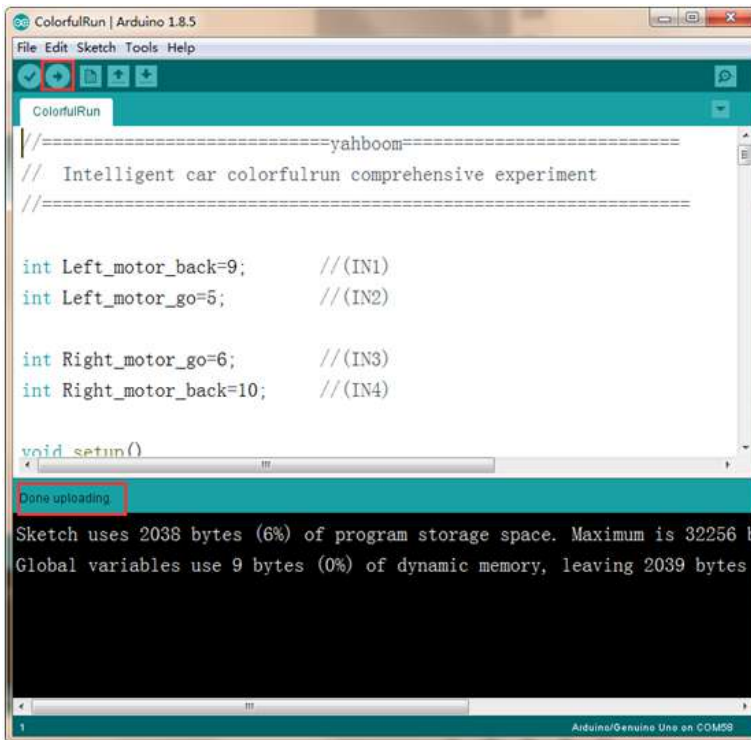
1. We need to open the code of this experiment: **ColorfulRun.ino**, click “√” under the menu bar to compile the code, and wait for the word "**Done compiling**" in the lower right corner, as shown in the figure below.



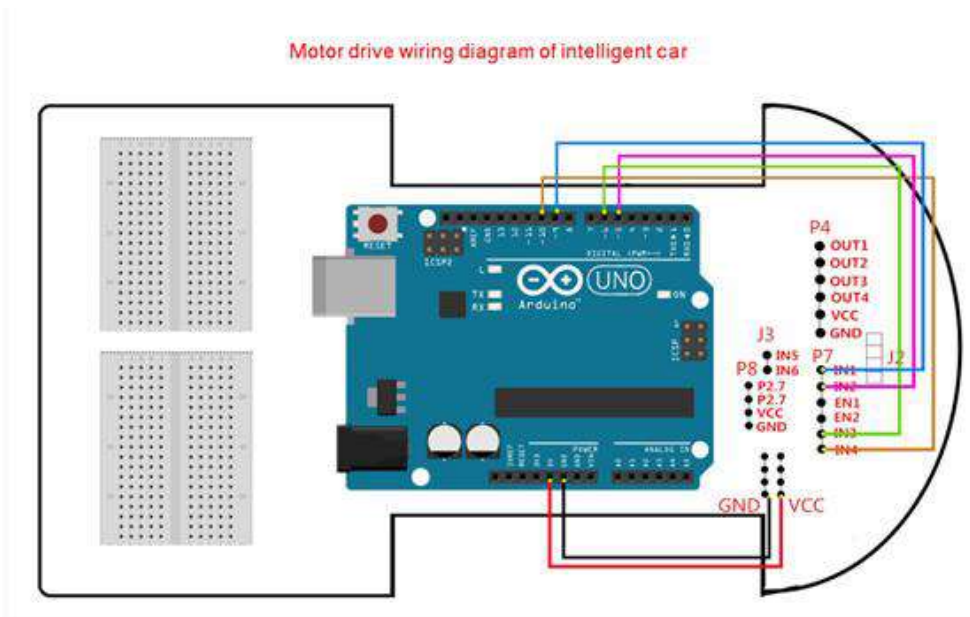
2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below.



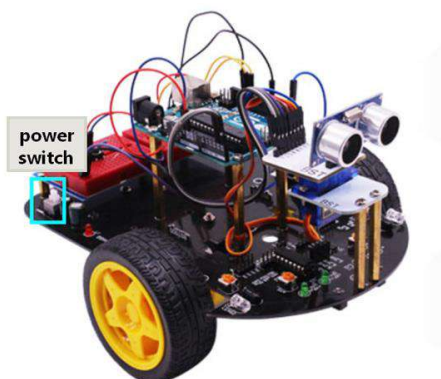
3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “**Done uploading**” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



4. Please wire the Smart Car as shown below.



5. Unplug the USB cable, place the smart car in a wide area, turn on the power switch, pause the smart car for two seconds, and then start the specified fancy action.



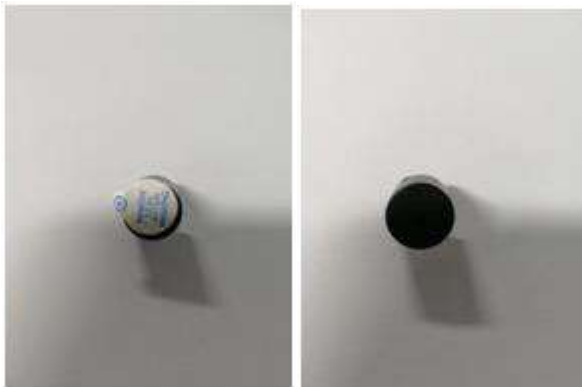
4- KeysacnStartCar

The purpose of the experiment:

As with the previous experimental project, upload the program keyssacnStartCa.ino of this lesson, put the smart car in a spacious area, turn on the power switch, and the car is still. After pressing the start button, the car starts the specified fancy action with a short whistle.

Precautions:

1.A is the active buzzer B is a passive buzzer. This experiment uses an active buzzer for experiments.



A

B

List of components required for the experiment:

Arduino Smart Car* 1

USB cable* 1

Active buzzer* 1

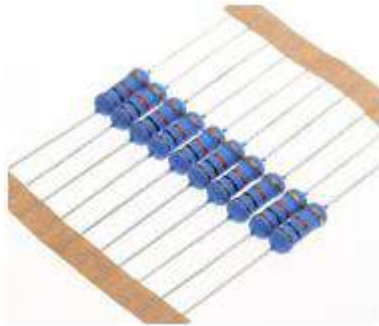
DuPont Line* 11

Breadboard* 1

Button * 1

10K resistor * 1





Experimental code analysis:

```
//=====yahboom=====
// Intelligent car key star and buzzer alarm experiment
//=====
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
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);
}
void run(int time) //car run
{
  digitalWrite(Right_motor_go,HIGH);
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,200);//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);
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200);//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) //car stop
{
  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)
{
    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 wheel back
    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 wheel 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)
{
    digitalWrite(Right_motor_go,LOW); //right motor back
    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); //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) //back
{

```

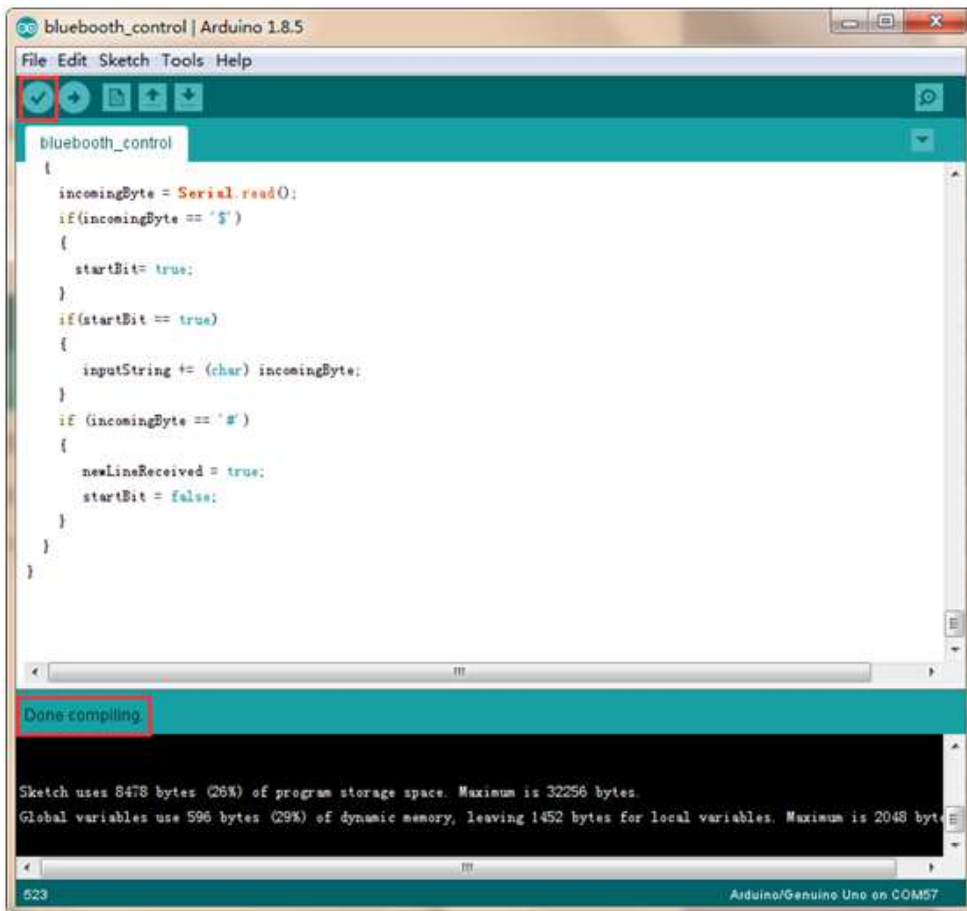
```

digitalWrite(Right_motor_go,LOW); //right wheel 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 wheel 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 loop()
{
keysacn(); //Call key scan function
back(10); //back 1s
brake(5); //stop 0.5s
run(10); //run 1s
brake(5); //stop 0.5s
left(10); //turn left 1s
right(10); //turn right 1s
spin_left(20); //left rotation 2s
spin_right(20); //right rotation2s
brake(5); //stop
}

```

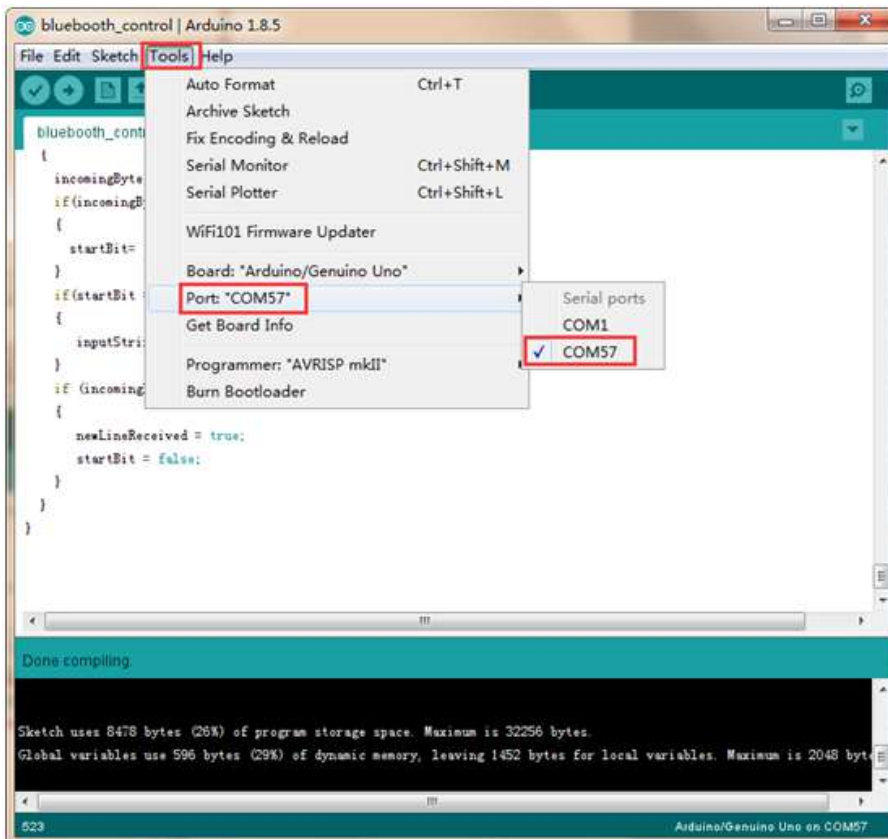
Experimental steps:

1. We need to open the code of this experiment: **keysacnStartCar.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.

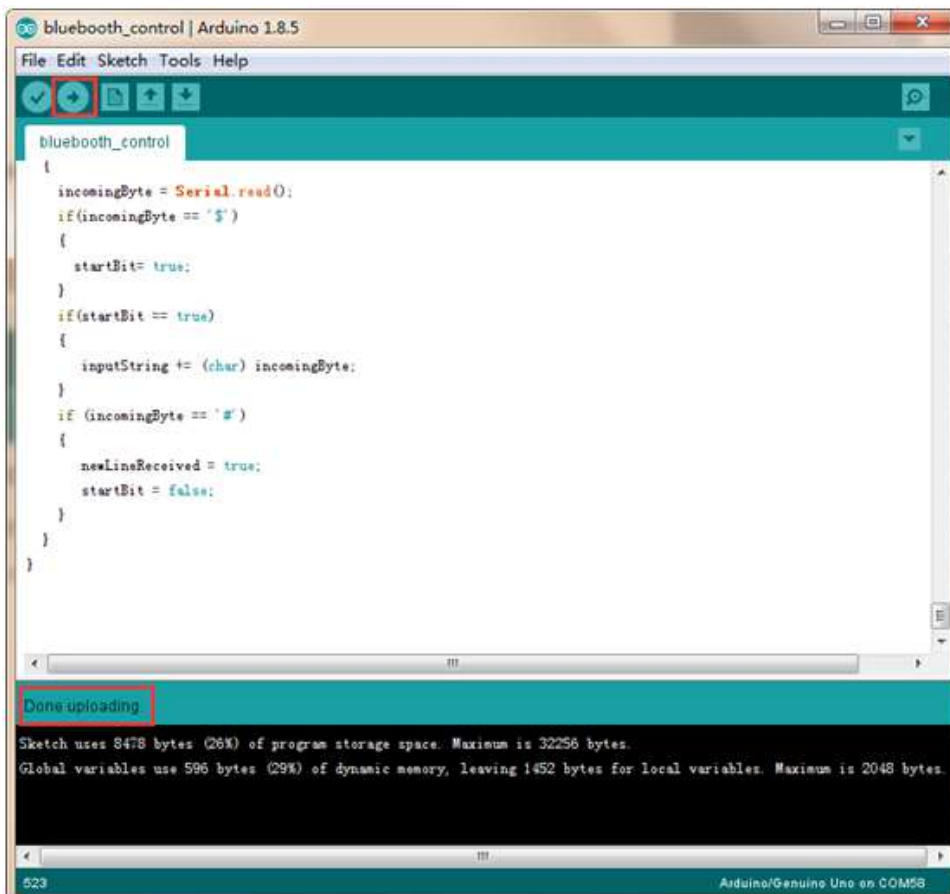


2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below.

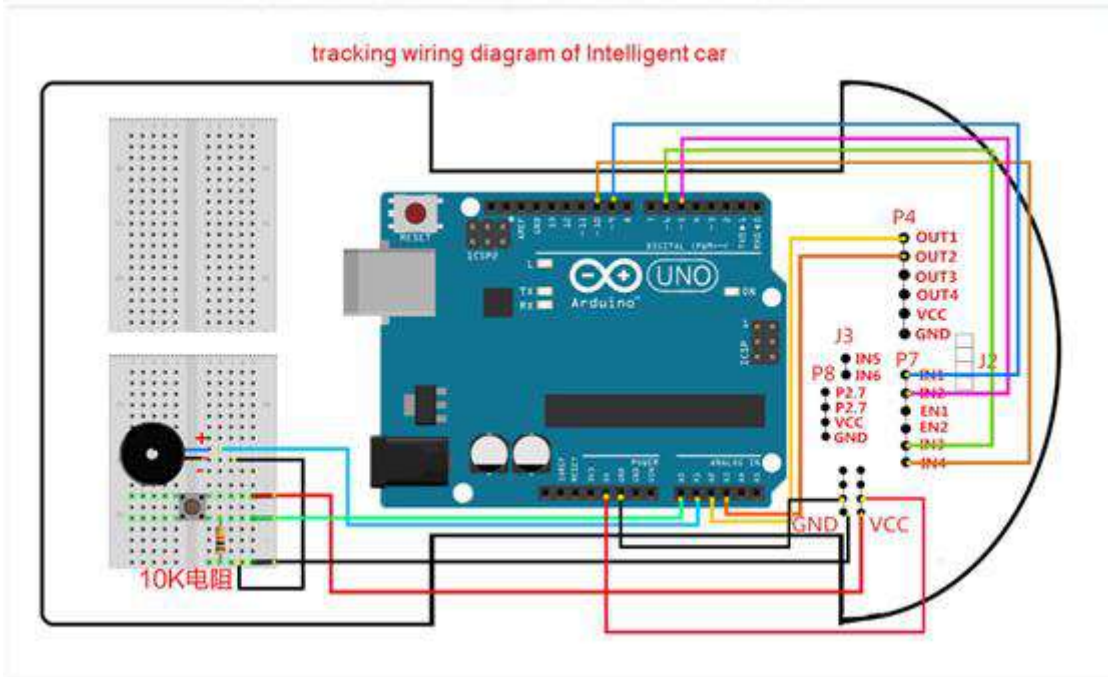




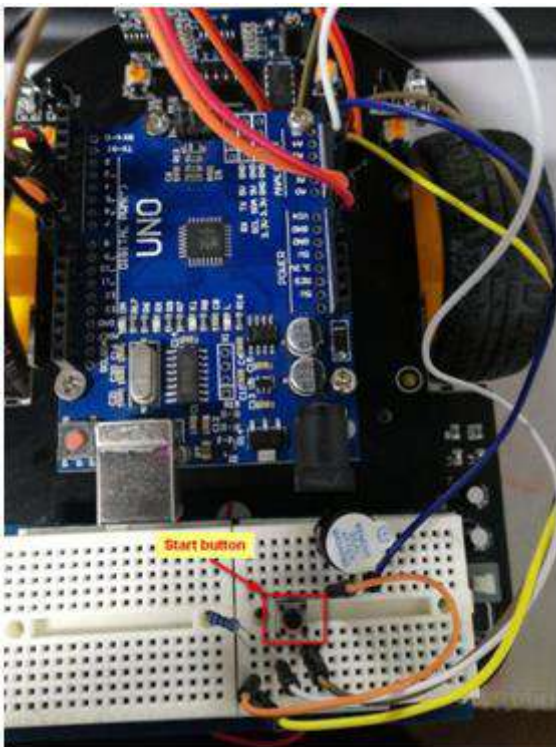
3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “**Done uploading**” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



4. Please wire the Smart Car as shown below.



5. Put the smart car in a spacious area, turn on the power switch, and the car is stationary. After pressing the start button as shown below, the car starts the specified fancy action with a short whistle.



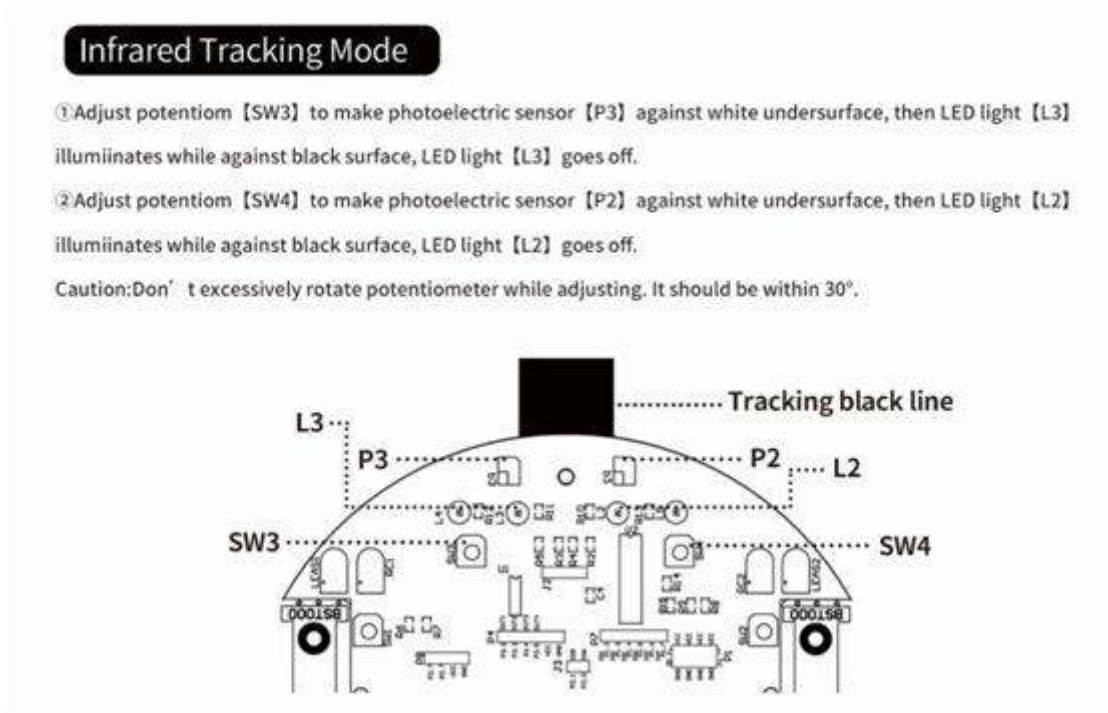
5- Tracking

The purpose of the experiment:

Place the smart car on the patrol track and press the start button. With a short whistle, the car begins to walk along the patrol track.

Precautions:

1. Before conducting the experiment, you need to debug the adjustable resistors sw3 and sw4 in front of the smart car, as shown below.

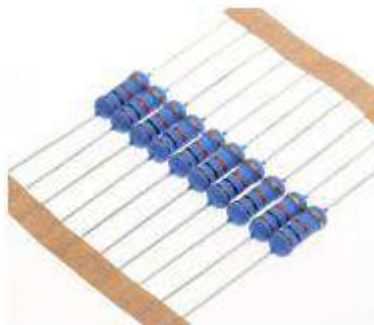


2. This experiment must be carried out in an environment without outdoor light, and it is also necessary to pull curtains to block outdoor light indoors.

List of components required for the experiment:

- Arduino Smart Car* 1
- USB data cable* 1
- Active buzzer* 1
- DuPont Line* 13
- Breadboard* 1
- Button * 1
- Electrical tape with a width of about 1.6cm* 1
- 10K resistor * 1





Experimental code analysis:

```
//=====yahboom=====
// Intelligent car tracking experiment
//=====
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
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 Right tracking infrared sensor for the input
interface
pinMode(SensorLeft, INPUT); //Define left tracking infrared sensor for the input
interface
}
//=====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,150);//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,150);//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)
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,150);
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);
digitalWrite(Left_motor_back,HIGH); //left motor back
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)
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,150);
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,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);
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 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 loop()
{
  keysacn(); //Call key scan function
  while(1)
  {
    //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.

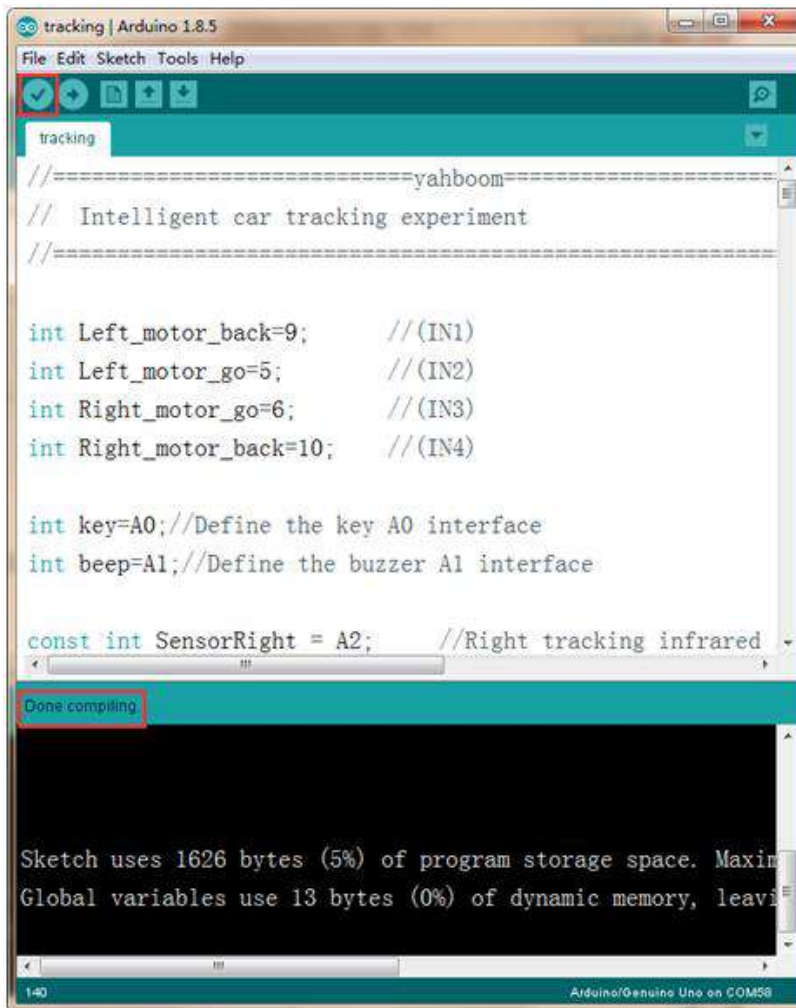
    SL = digitalRead(SensorLeft);
    //There is a signal that in the white area the L2 is bright on the car floor;
    // no signal indicates that on the black line and the L2 is extinguishing on the car floor.

    if (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 //all white, stop
      brake();
  }
}

```

Experimental steps:

1. We need to open the code of this experiment: **tracking.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 | Arduino 1.8.5
File Edit Sketch Tools Help
tracking
//-----yahboom-----
// Intelligent car tracking experiment
//-----

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

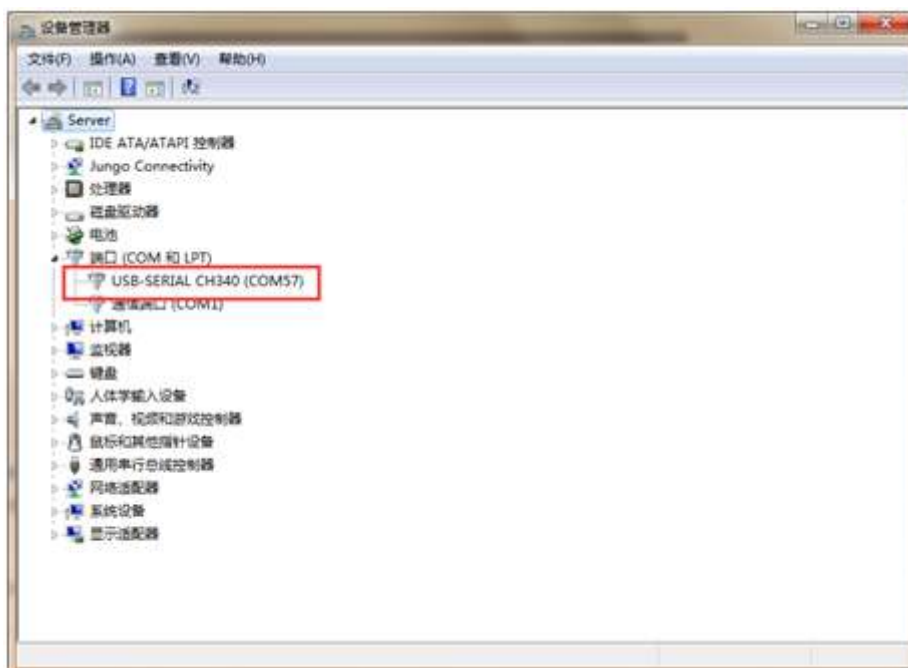
const int SensorRight = A2; //Right tracking infrared

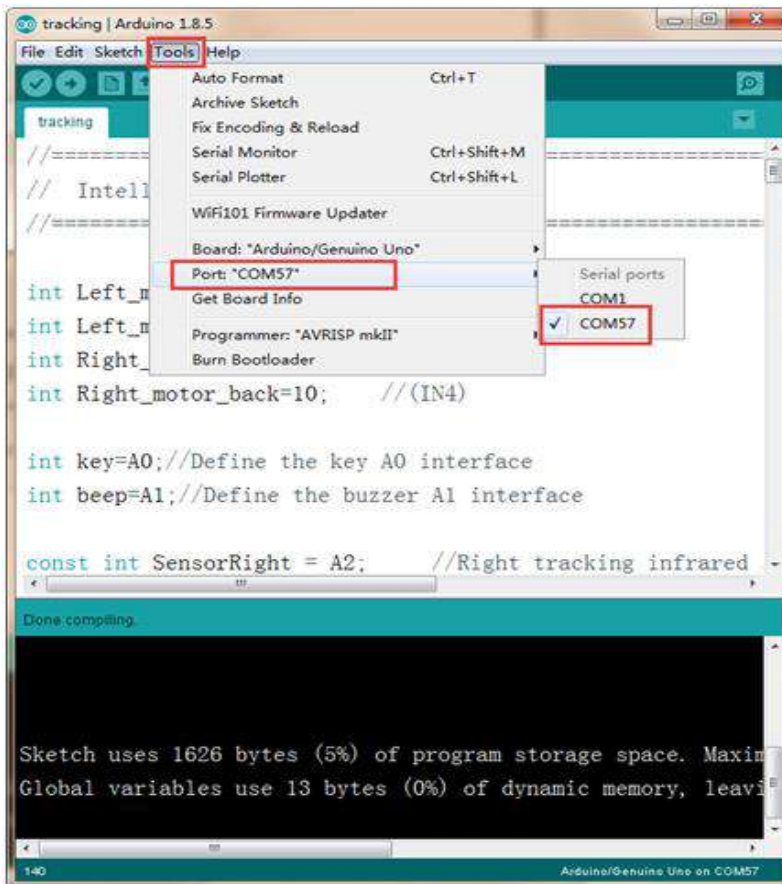
Done compiling

Sketch uses 1626 bytes (5%) of program storage space. Maximum is 32256 bytes.
Global variables use 13 bytes (0%) of dynamic memory, leaving 2047 bytes free.

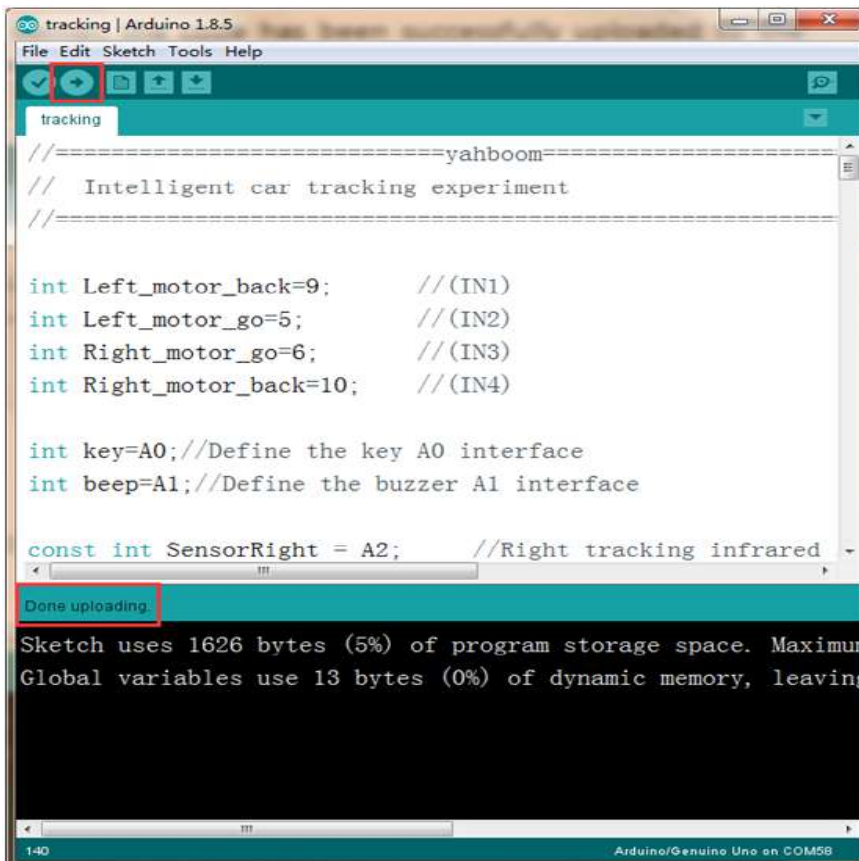
1:40 Arduino/Genuino Uno on COM58
```

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.

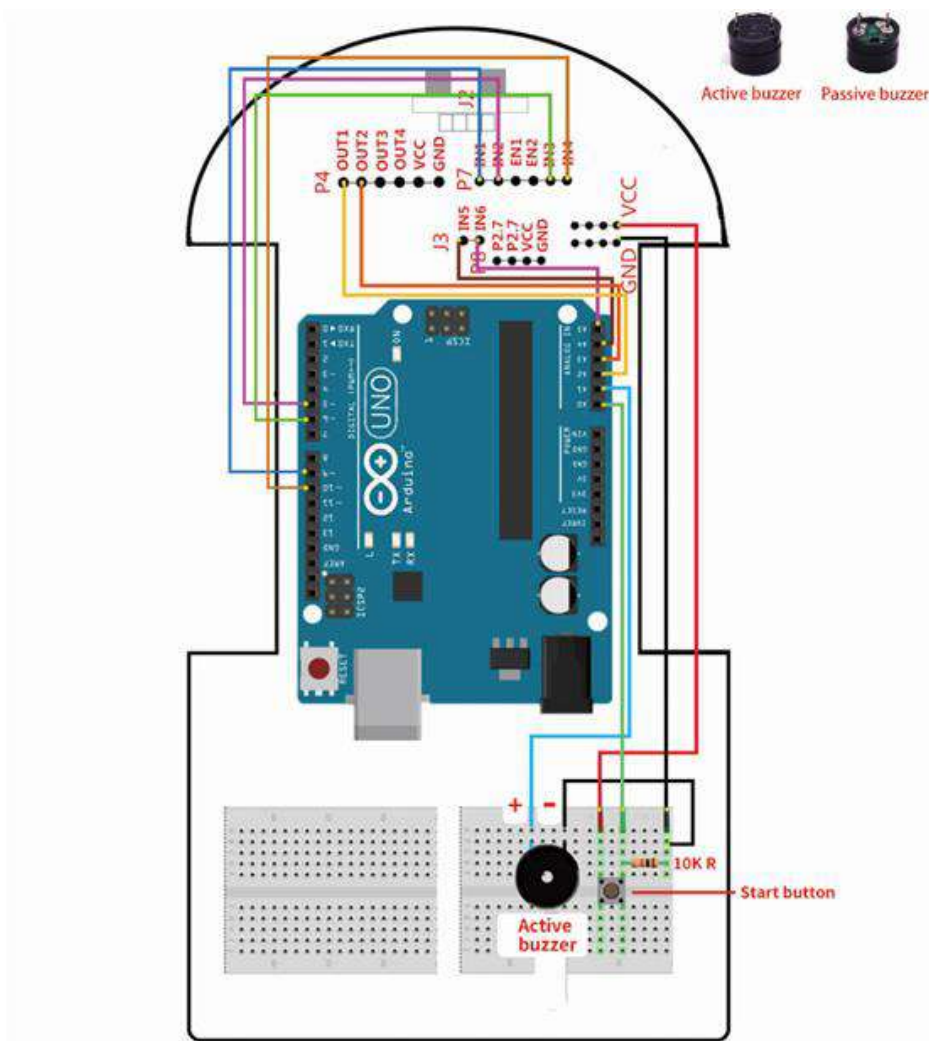




3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “**Done uploading**” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



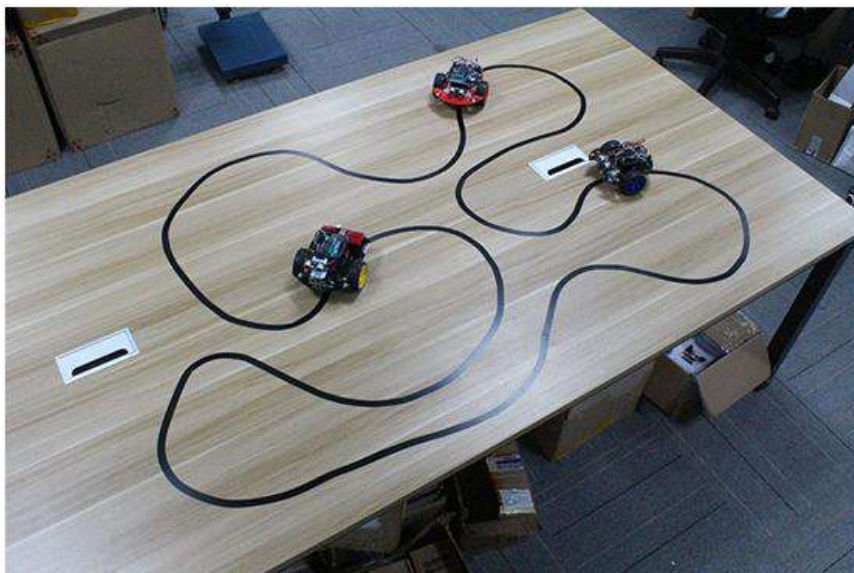
4. Please wire the Smart Car as shown below.



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. Use a 1.6cm wide black electrical tape to attach the curved track as shown in the figure below on the light ground or tabletop. Place the debugged smart car on the track and press the start button. The car starts to walk along the black line.



6- Infrared obstacle avoidance obstacle – basic

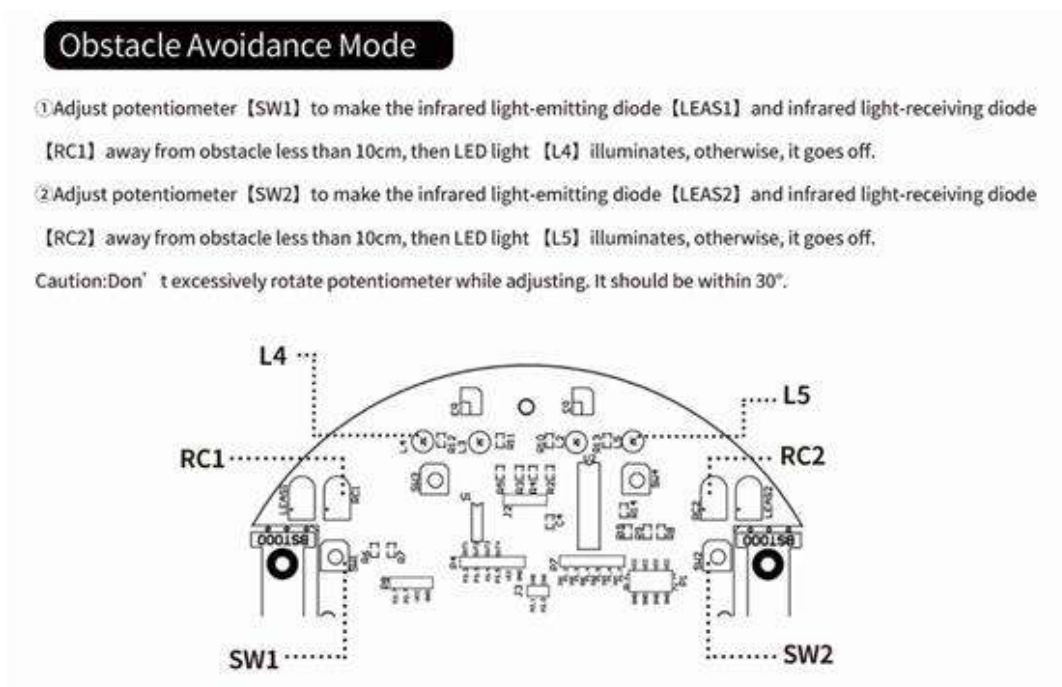
6 Infrared obstacle avoidance obstacle - basic

The purpose of the experiment:

The carton simulates obstacles on the spacious floor, and the smart car that has uploaded the program **avoid.ino** is placed on the ground. After pressing the start button of the car, the car starts to avoid the obstacles in front.

Precautions:

1. Before conducting the experiment, you need to debug the adjustable resistors SW1 and SW2 in front of the smart car, as shown below.



2. This experiment must be carried out in an environment without outdoor light, and it is also necessary to pull curtains to block outdoor light indoors.

List of components required for the experiment:

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





Experimental code analysis:

```
//=====yahboom=====
// Intelligent car infrared obstacle avoidance 1(basic obstacle avoidance)
//=====
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)
const int SensorLeft_2 = A4;   //Left infrared sensor(P3.4 OUT3)
const int SensorRight_2 = A5;  //Right infrared sensor(P3.5 OUT4)
int SL;  //Left tracking infrared sensor state
int SR;  //Right tracking infrared sensor state
int SL_2; //Left infrared sensor state
int SR_2; //Right infrared sensor state
```

```

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 Right tracking infrared sensor for the input
interface
  pinMode(SensorLeft, INPUT); //Define left tracking infrared sensor for the input
interface
  pinMode(SensorRight_2, INPUT); //Define right infrared sensor for the input interface
  pinMode(SensorLeft_2, INPUT); //Define left infrared sensor for the input interface
}
//=====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,200); //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,200); //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)
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,175);
  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,250);
    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,100); //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,175);
    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,100); //PWM ratio 0~255 speed control
    digitalWrite(Left_motor_go,HIGH); //left motor go
    digitalWrite(Left_motor_back,LOW);
    analogWrite(Left_motor_go,100);
    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,100);//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 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 is released
digitalWrite(beep,LOW); //buzzer no sound
}
else
digitalWrite(beep,LOW); // buzzer no sound
}
}
}
void loop()
{
keysacn(); //Call key scan function
while(1)
{
//There is a signal is LOW , no signal is HIGH
SR_2 = digitalRead(SensorRight_2);
SL_2 = digitalRead(SensorLeft_2);
if (SL_2 == HIGH&&SR_2==HIGH)
run(); //Call run function
else if (SL_2 == HIGH & SR_2 == LOW)
//There is an obstacle on the right,return signal,turn left.
left();
else if (SR_2 == HIGH & SL_2 == LOW)
//There is an obstacle on the left,return signal,turn right
right();
else //There are obstacles on both sides , back
back(3);
}
}
}

```

Experimental steps:

1. We need to open the code of this experiment: **avoid.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.

```

avoid | Arduino 1.8.5
文件 编辑 项目 工具 帮助
avoid
//=====yahboom=====
// Intelligent car infrared obstacle avoidance 1(basic obstacle avoidance)
//=====

int Left_motor_back=9; // (I11)
int Left_motor_go=5; // (I12)
int Right_motor_go=6; // (I13)
int Right_motor_back=10; // (I14)

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 OUI1)
const int SensorLeft = A3; //Left tracking infrared sensor (P3.3 OUI2)

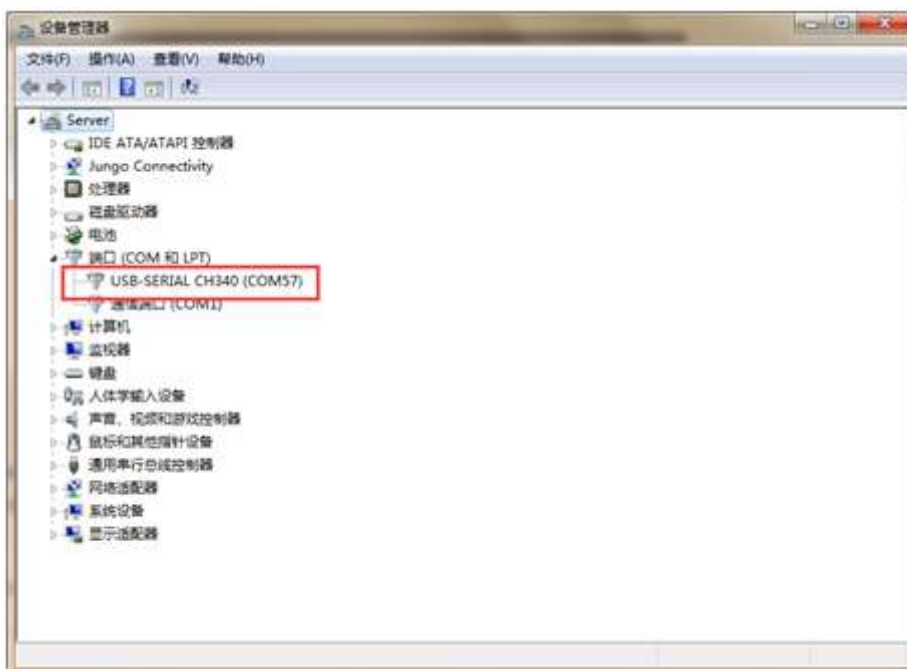
const int SensorLeft_2 = A4; //Left infrared sensor (P3.4 OUI3)
const int SensorRight_2 = A5; //Right infrared sensor (P3.5 OUI4)

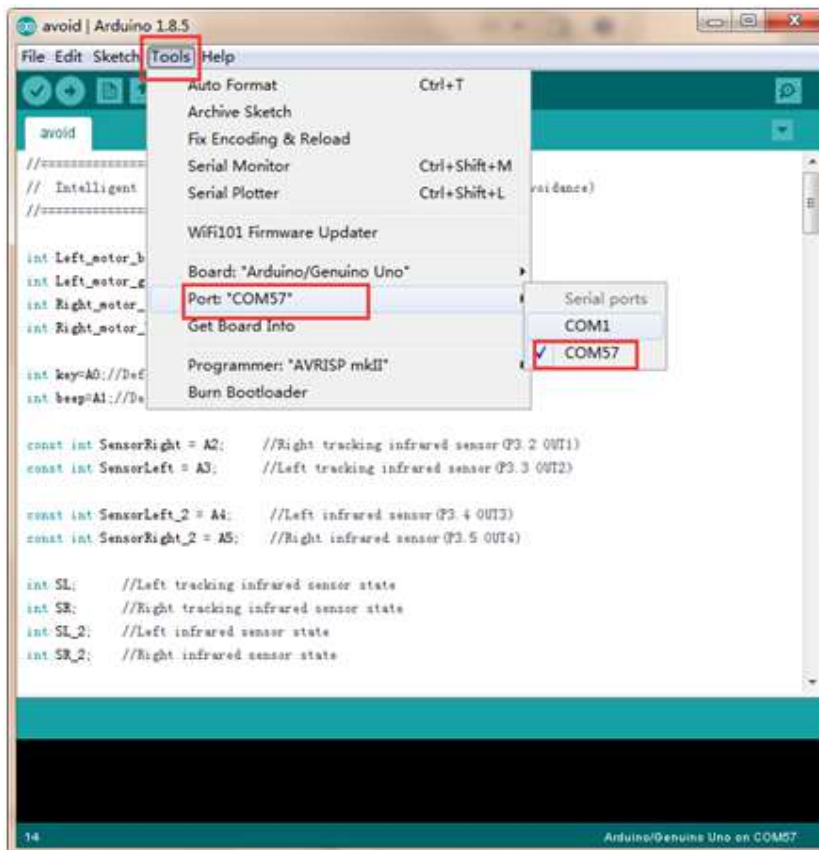
int SL; //Left tracking infrared sensor state
int SR; //Right tracking infrared sensor state
int SL_2; //Left infrared sensor state
int SR_2; //Right infrared sensor state

编译完成。
项目使用了 1776 字节, 占用了 (5%) 程序存储空间。最大为 32256 字节。
全局变量使用了13字节, (0%)的动态内存, 余留2035字节局部变量。最大为2048字节。
158 Arduino/Genuino Uno COM1

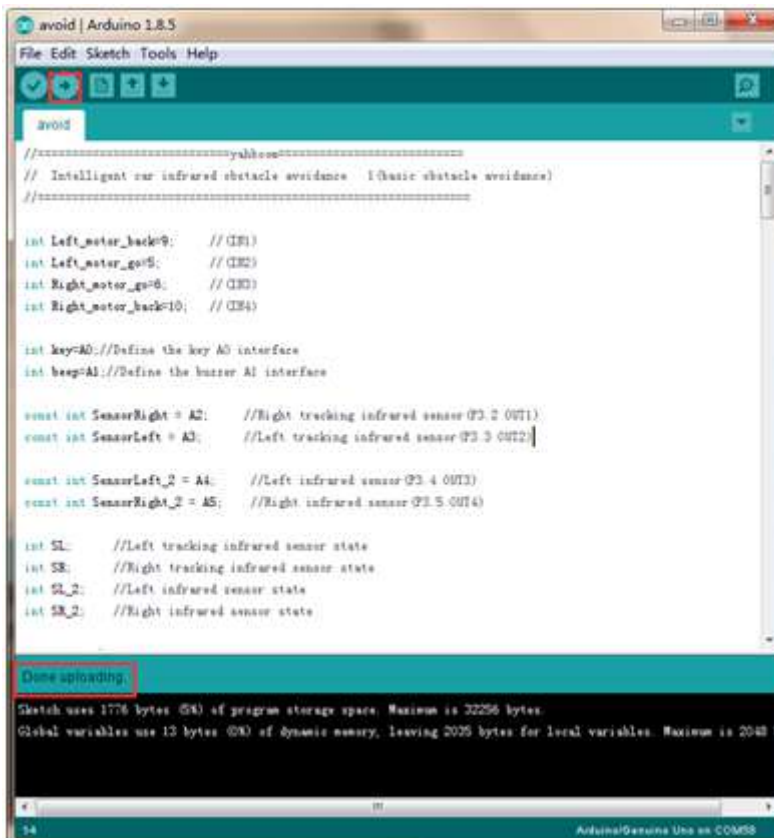
```

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.



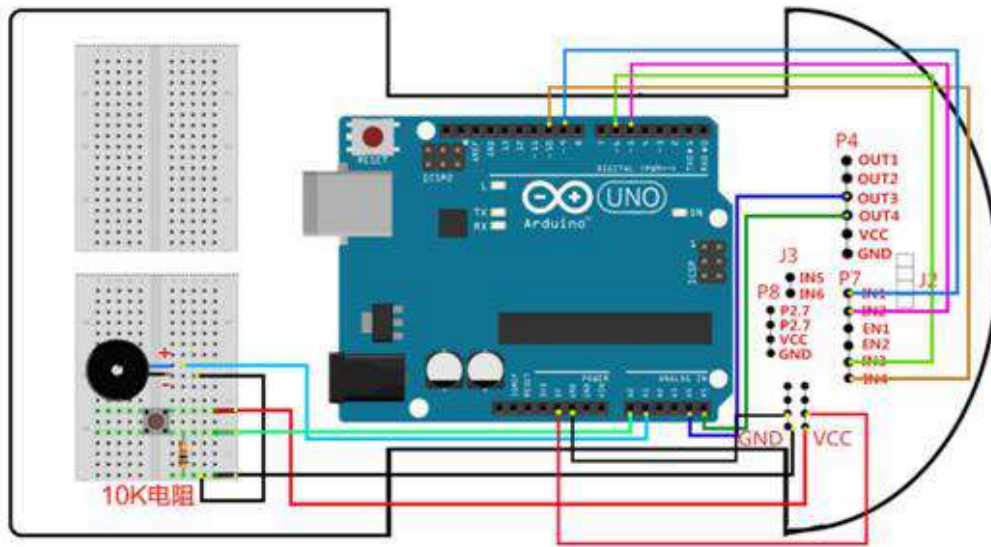


3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “**Done uploading**” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.

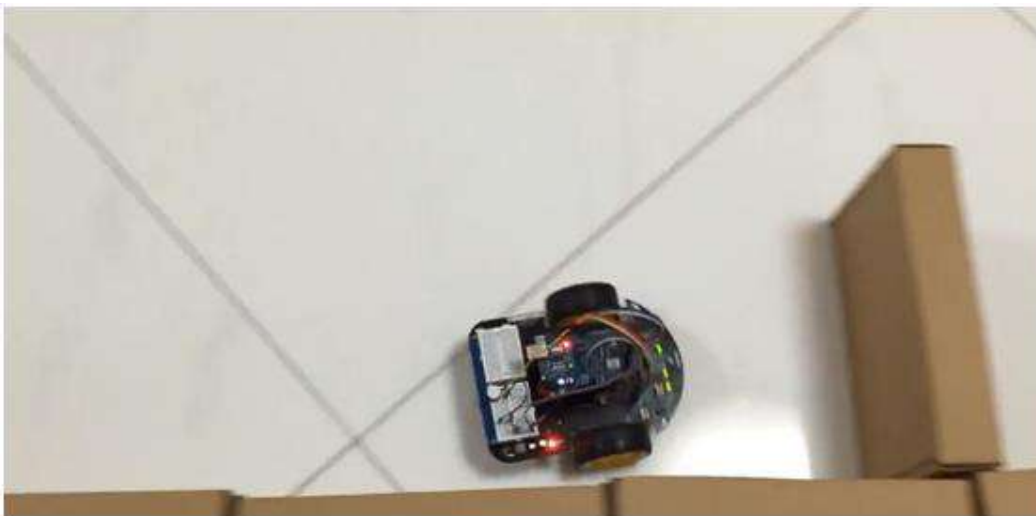


4. Please wire the Smart Car as shown below.

Infrared obstacle avoidance



5. Place the debugged smart car on the ground with obstacles and press the start button. The car will avoid obstacles.



7- Infrared obstacle avoidance – follow

The purpose of the experiment:

Using the characteristics of the infrared obstacle avoidance sensor, the obstacle is used to block the left and right infrared obstacle avoidance probes of the trolley, and when the trolley recognizes the obstacle ahead, the obstacle is followed.

Precautions:

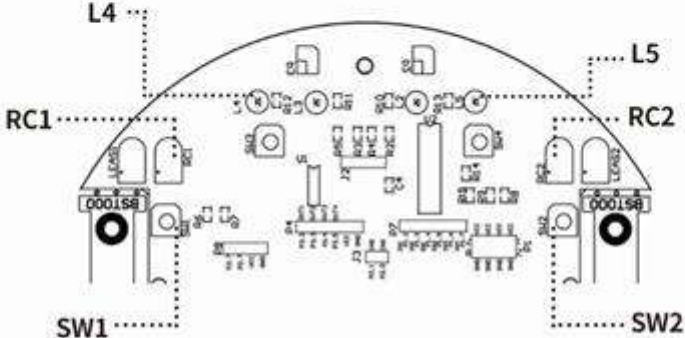
1. Before conducting the experiment, you need to debug the adjustable resistors SW1 and SW2 in front of the smart car, as shown below.

Obstacle Avoidance Mode

① Adjust potentiometer 【SW1】 to make the infrared light-emitting diode 【LEAS1】 and infrared light-receiving diode 【RC1】 away from obstacle less than 10cm, then LED light 【L4】 illuminates, otherwise, it goes off.

② Adjust potentiometer 【SW2】 to make the infrared light-emitting diode 【LEAS2】 and infrared light-receiving diode 【RC2】 away from obstacle less than 10cm, then LED light 【L5】 illuminates, otherwise, it goes off.

Caution: Don't excessively rotate potentiometer while adjusting. It should be within 30°.

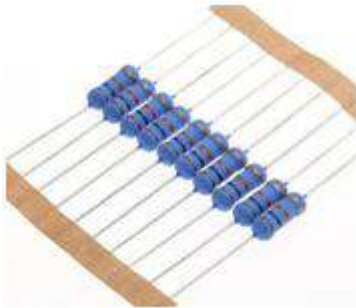


2. This experiment must be carried out in an environment without outdoor light, and it is also necessary to pull curtains to block outdoor light indoors.

List of components required for the experiment:

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





Experimental code analysis:

```
//=====yahboom=====
// Intelligent car infrared obstacle avoidance experiment2(follow avoidance)
//=====
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)
const int SensorLeft_2 = A4;   //Left infrared sensor(P3.4 OUT3)
const int SensorRight_2 = A5;  //Right infrared sensor(P3.5 OUT4)
int SL;    //Left tracking infrared sensor state
int SR;    //Right tracking infrared sensor state
int SL_2;  //Left infrared sensor state
int SR_2;  //Right infrared sensor state
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 Right tracking infrared sensor for the input
interface
pinMode(SensorLeft, INPUT); //Define left tracking infrared sensor for the input
interface
pinMode(SensorRight_2, INPUT); //Define right infrared sensor for the input interface
pinMode(SensorLeft_2, INPUT); //Define left infrared sensor for the input interface
}
//=====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,100);//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,100);//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)
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,100);
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,100);
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,100); //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,100);
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,100); //PWM ratio 0~255 speed control
digitalWrite(Left_motor_go,HIGH); //left motor go
digitalWrite(Left_motor_back,LOW);
analogWrite(Left_motor_go,100);
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,100); //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;

```

```

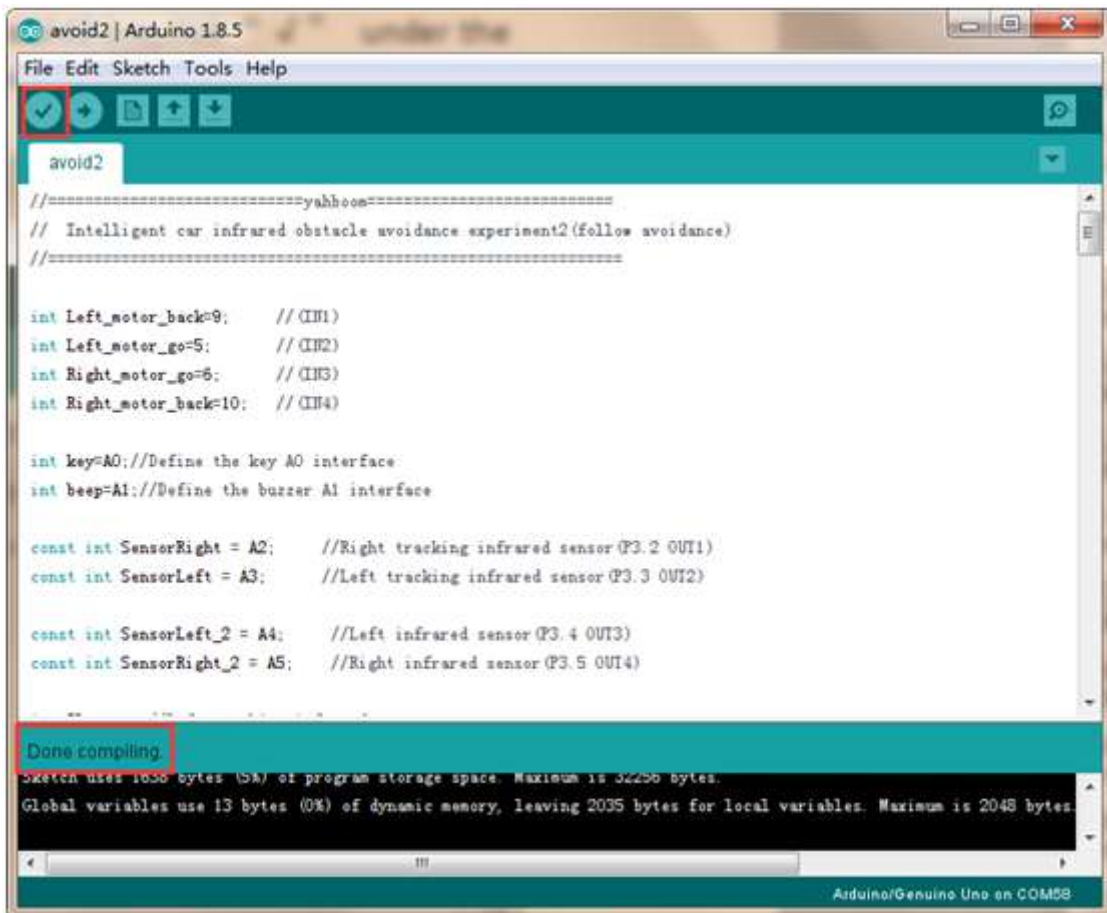
val=digitalRead(key);//Read the value of the port 7 level to the val
while(!digitalRead(key))//When the key is not pressed, circulate all the time
{
  val=digitalRead(key);//This sentence can be omitted and the circulate can run away
}
while(digitalRead(key))//When the key is pressed
{
  delay(10);
  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 is released
      digitalWrite(beep,LOW);    //buzzer no sound
  }
  else
    digitalWrite(beep,LOW);    //buzzer no sound
}
}
void loop()
{
  keysacn();
  while(1)
  {
    //There is a signal is LOW , no signal is HIGH
    SR_2 = digitalRead(SensorRight_2);
    SL_2 = digitalRead(SensorLeft_2);
    if (SL_2 == LOW&&SR_2==LOW)
      run(); //Call run function
    else if (SL_2 == HIGH & SR_2 == LOW)
      //There is an obstacle on the right,return signal,turn right.

      right();
    else if (SR_2 == HIGH & SL_2 == LOW)
      //There is an obstacle on the left,return signal,turn left.
      left();
    else //There are no obstacles on both sides,brake
      brake();
  }
}

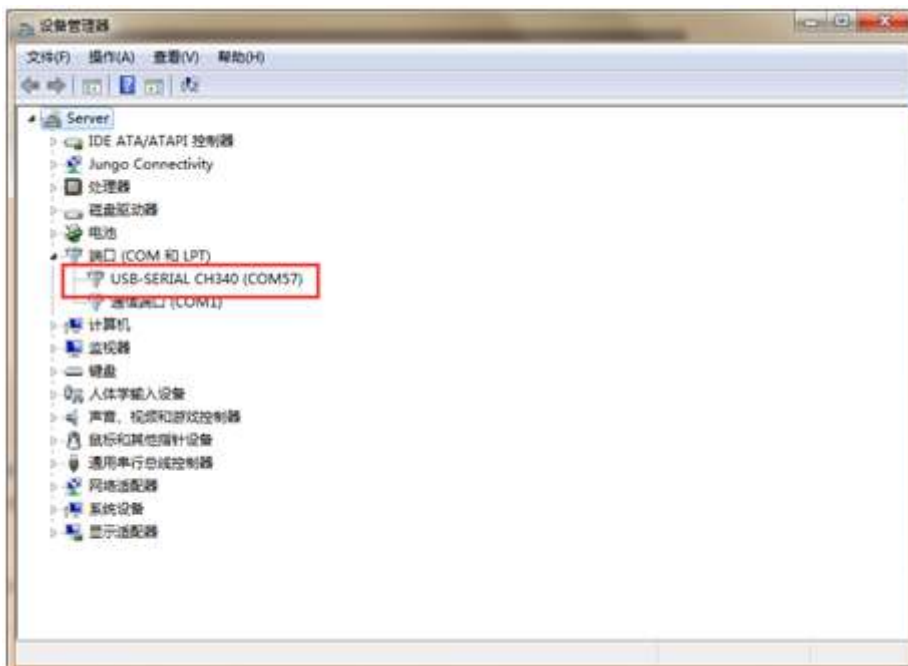
```

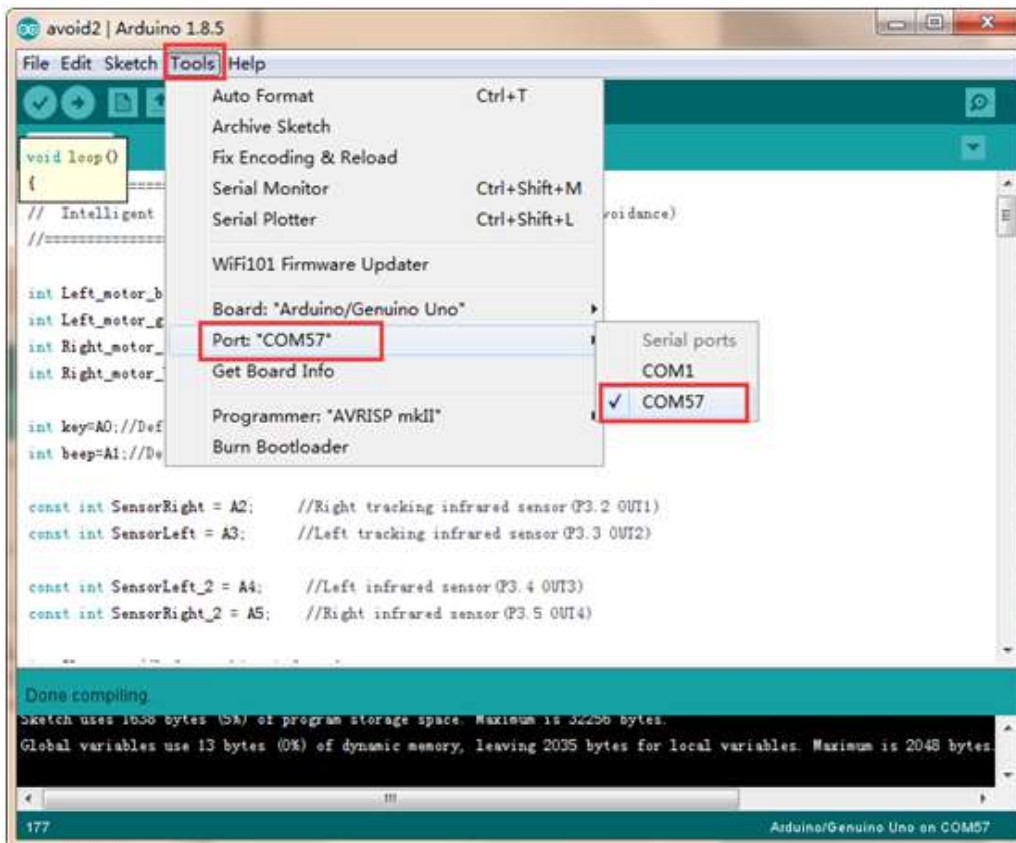
Experimental steps:

1. We need to open the code of this experiment: **avoid2.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.

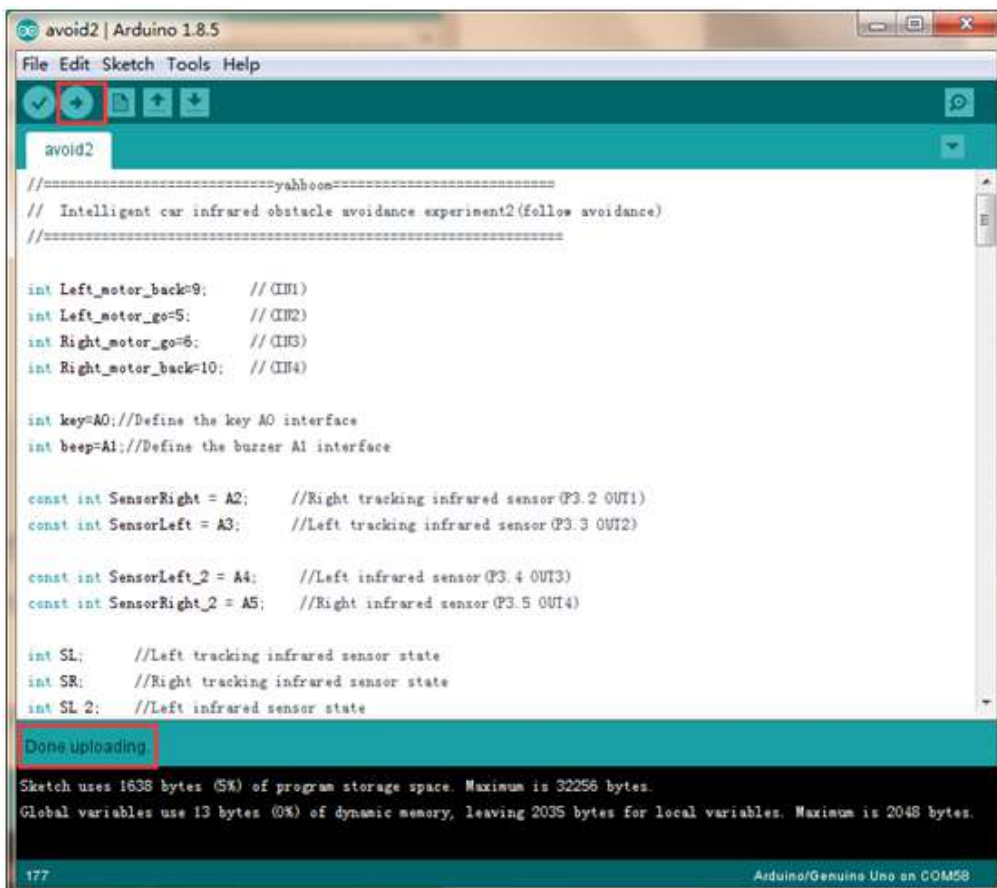


2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below.



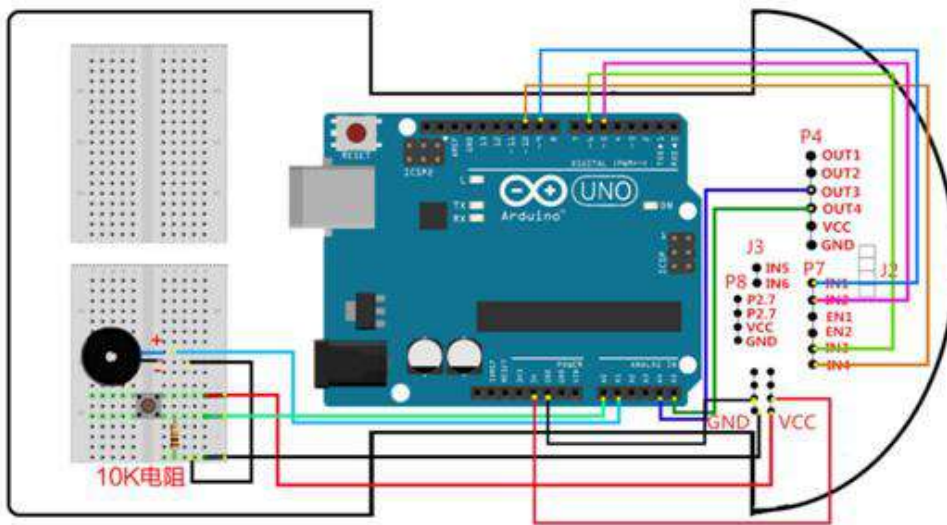


3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “**Done uploading**” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.

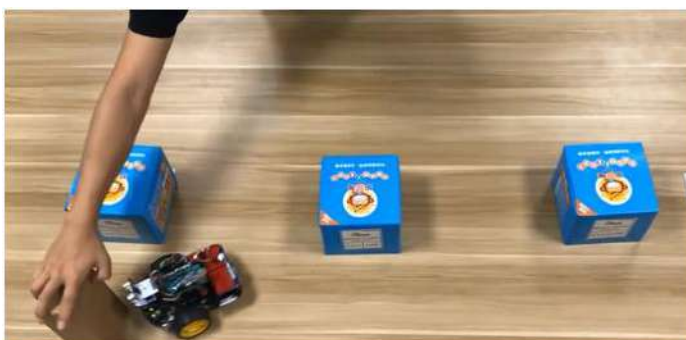


4. Please wire the Smart Car as shown below.

Infrared obstacle avoidance



5. Put the debugged smart car in a wide area, turn on the power switch, press the start button, place the obstacle in front of the left and right infrared obstacle avoidance probe of the smart car, move the obstacle, and the smart car will follow the obstacle.



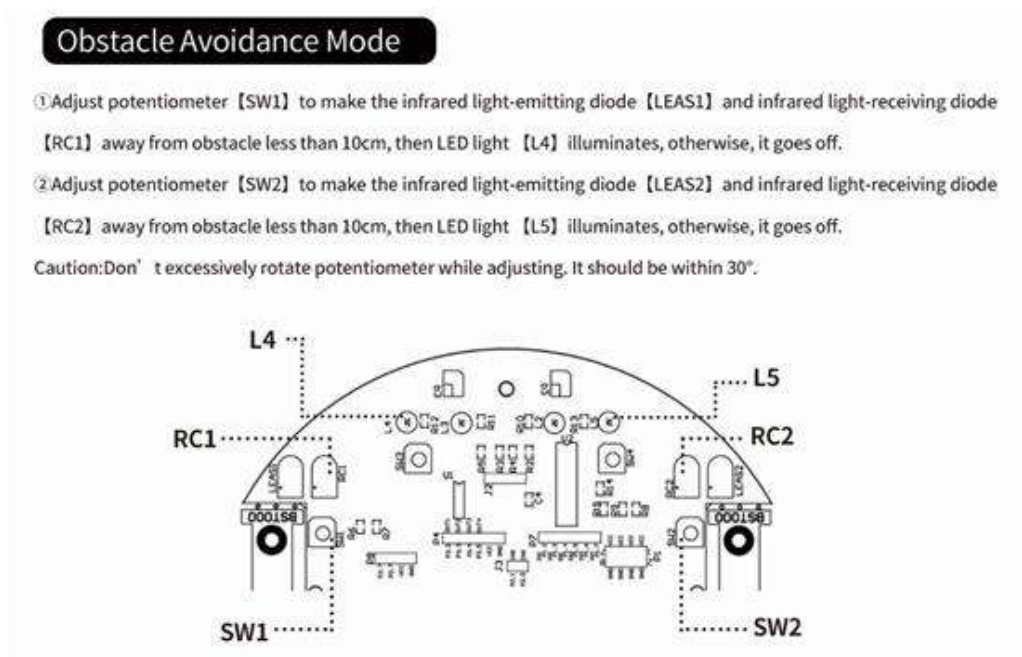
8- Infrared obstacle avoidance with backdrop

The purpose of the experiment:

The difference between this course and the basic obstacle avoidance course is that after the obstacles are detected by the left and right infrared obstacle avoidance probes, the smart car retreats and the direction of the obstacles is avoided.

Precautions:

1. Before conducting the experiment, you need to debug the adjustable resistors SW1 and SW2 in front of the smart car, as shown below.

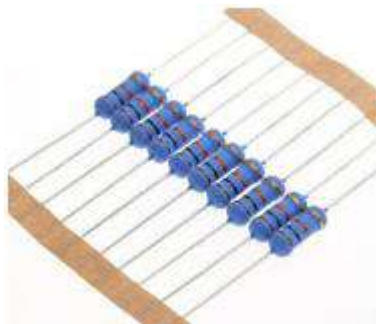


2. This experiment must be carried out in an environment without outdoor light, and it is also necessary to pull curtains to block outdoor light indoors.

List of components required for the experiment:

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





Experimental code analysis:

```
//=====yahboom=====
// Intelligent car infrared obstacle avoidance experiment3(Obstacle avoidance with
//backdrop)
//=====
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)
const int SensorLeft_2 = A4;   //Left infrared sensor(P3.4 OUT3)
const int SensorRight_2 = A5;  //Right infrared sensor(P3.5 OUT4)
int SL;    //Left tracking infrared sensor state
int SR;    //Right tracking infrared sensor state
int SL_2;  //Left infrared sensor state
int SR_2;  //Right infrared sensor state
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 Right tracking infrared sensor for the input
interface
pinMode(SensorLeft, INPUT); //Define left tracking infrared sensor for the input
interface
pinMode(SensorRight_2, INPUT); //Define right infrared sensor for the input interface
pinMode(SensorLeft_2, INPUT); //Define left infrared sensor for the input interface
}
//=====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,100);//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,100);//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)
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,100);
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,100);
  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,100); //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,100);
  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,100); //PWM ratio 0~255 speed control
  digitalWrite(Left_motor_go,HIGH); //left motor go
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100);
  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,100); //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()
{

```

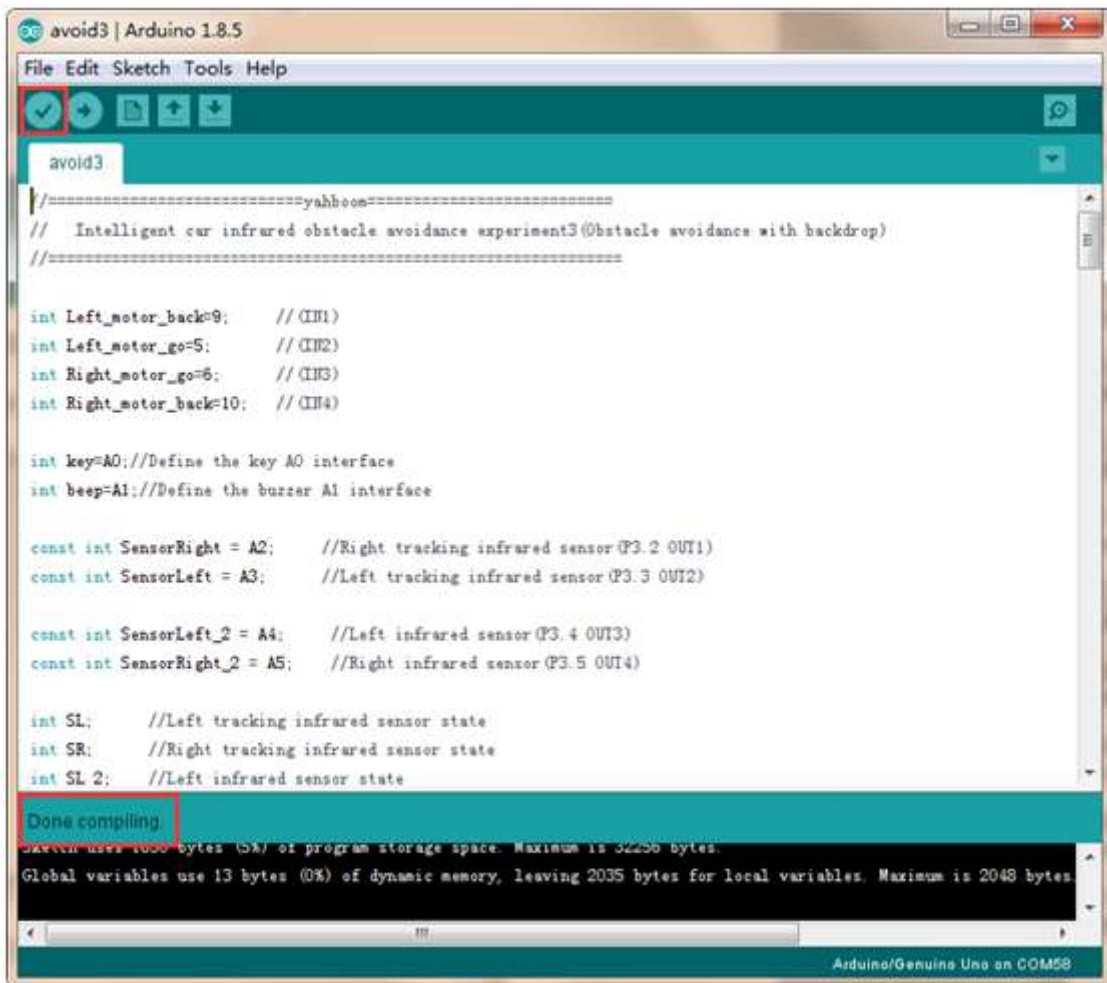
```

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))//This sentence can be omitted and the circulate can run away.
{
  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 is released
      digitalWrite(beep,LOW);    //buzzer no sound
  }
  else
    digitalWrite(beep,LOW);    //buzzer no sound
}
}
}
void loop()
{
  keysacn();    //Call key scan function
  while(1)
  {
    //There is a signal is LOW , no signal is HIGH
    SR_2 = digitalRead(SensorRight_2);
    SL_2 = digitalRead(SensorLeft_2);
    if (SL_2 == HIGH&&SR_2==HIGH)
      run(); //Call run function
    else if (SL_2 == HIGH & SR_2 == LOW)
      //There is an obstacle on the right,return signal,turn left.
      left();
    else if (SR_2 == HIGH & SL_2 == LOW)
      //There is an obstacle on the left,return signal,turn right
      right();
    else // There are obstacles on both sides , back
    {
      back(4.5);
      spin_right(4.5);//right rotation, adjust direction
    }
  }
}
}

```

Experimental steps:

1. We need to open the code of this experiment: **avoid3.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.



```

avoid3 | Arduino 1.8.5
File Edit Sketch Tools Help
avoid3
//=====yahboom=====
// Intelligent car infrared obstacle avoidance experiment3 (Obstacle avoidance with backdrop)
//=====

int Left_motor_back=9; // (III1)
int Left_motor_go=5; // (III2)
int Right_motor_go=6; // (III3)
int Right_motor_back=10; // (III4)

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)

const int SensorLeft_2 = A4; //Left infrared sensor (P3.4 OUT3)
const int SensorRight_2 = A5; //Right infrared sensor (P3.5 OUT4)

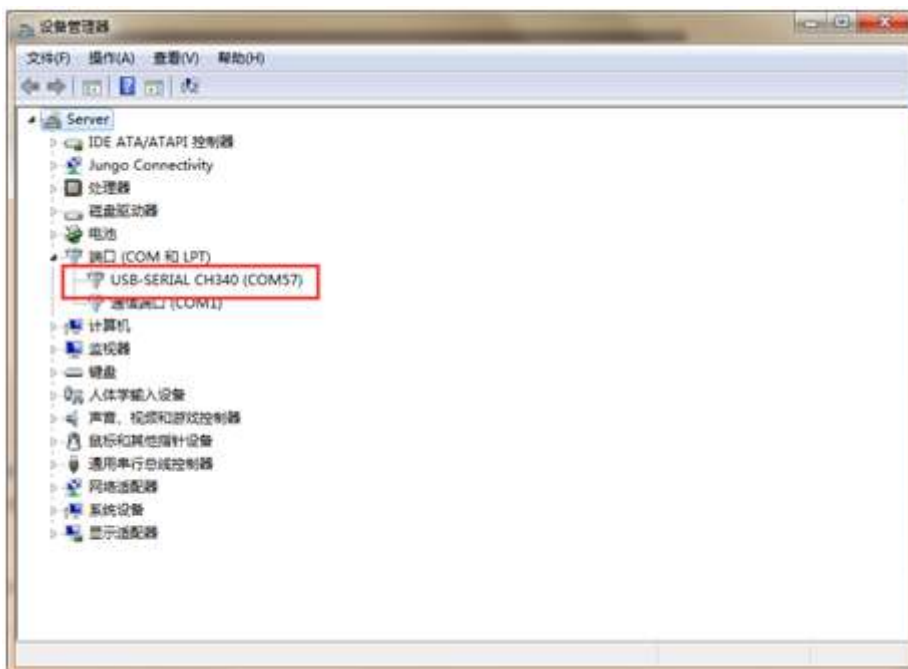
int SL; //Left tracking infrared sensor state
int SR; //Right tracking infrared sensor state
int SL 2; //Left infrared sensor state

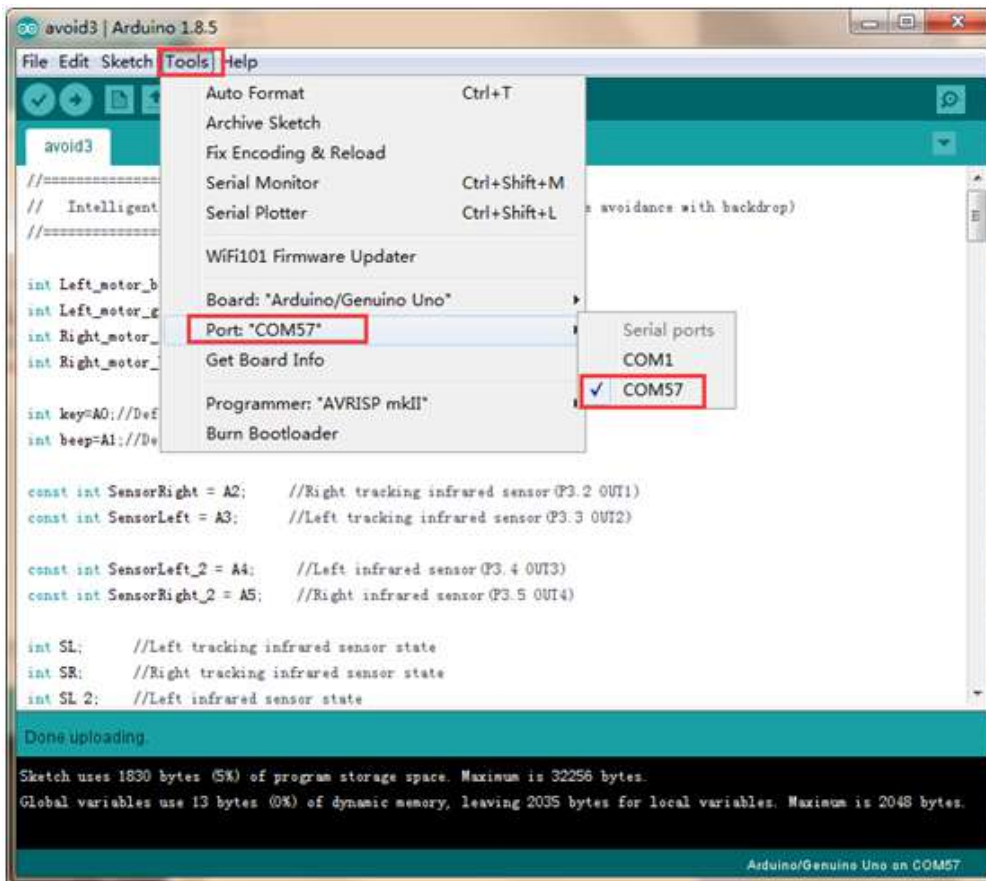
Done compiling
Sketch uses 1000 bytes (5%) of program storage space. Maximum is 32256 bytes.
Global variables use 13 bytes (0%) of dynamic memory, leaving 2035 bytes for local variables. Maximum is 2048 bytes.

Arduino/Genuino Uno on COM58

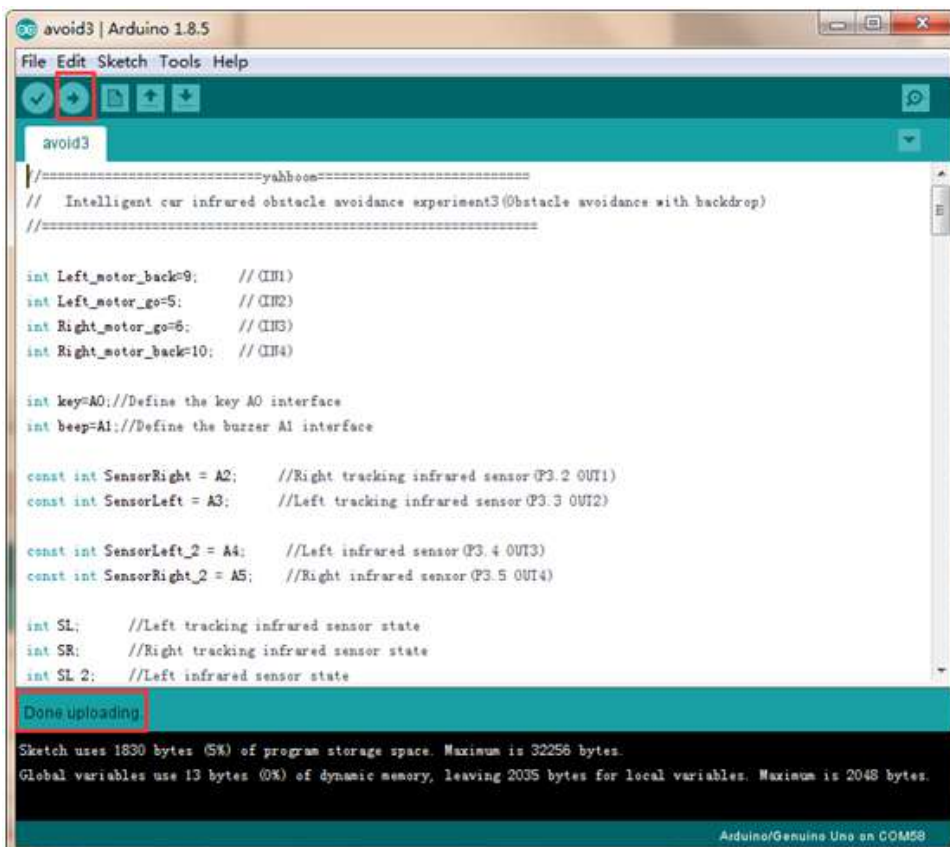
```

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.



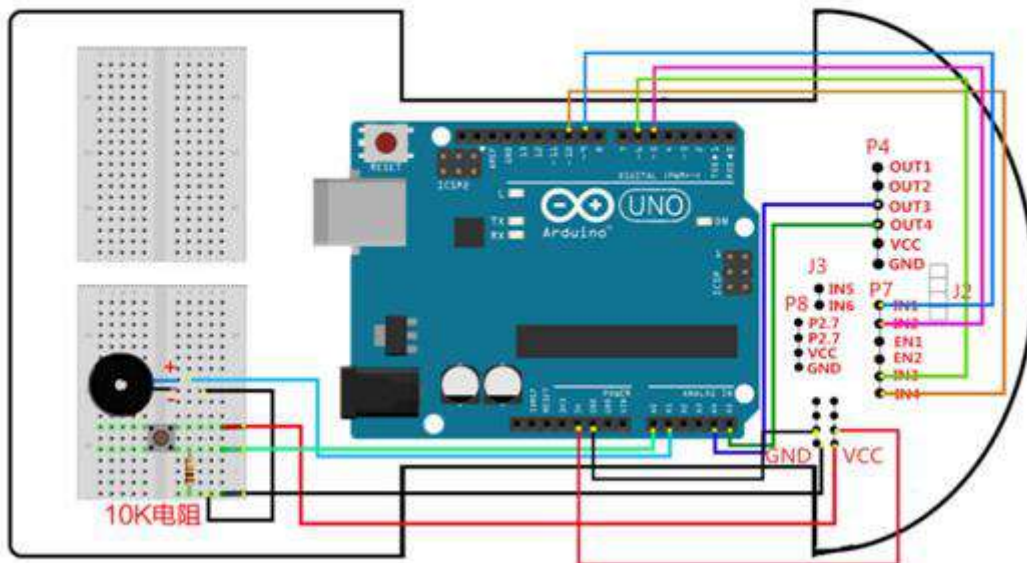


3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “**Done uploading**” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



4. Please wire the Smart Car as shown below.

Infrared obstacle avoidance



5. Use the carton to simulate the obstacles on the ground, put the smart car that has uploaded the program **avoid.ino** on the ground, press the start button of the tail of the car, if the smart car detects that there is no obstacle in front, the car goes straight. If an obstacle is detected in the left front, the car turns right to avoid obstacles. If the right front right obstacle is detected, the cart turns left to avoid obstacles. If it is detected that there are obstacles on the left and right sides, the smart car will retreat and turn around to avoid obstacles.

9- Ultrasonic distance measurement

The purpose of the experiment:

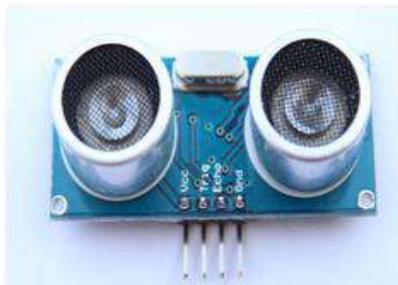
After uploading the program ultrasonic.ino of this lesson, open the power switch of the car, the ultrasonic sensor will measure the distance in front and display it on the LCD screen.

Precautions:

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

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
- Ultrasonic sensor*1



Experimental code analysis:

```
//=====yahboom=====
// Intelligent car ultrasonic distance measurement
//=====
#include <LiquidCrystal.h> //Declare the function library of 1602 liquid crystals
```

```

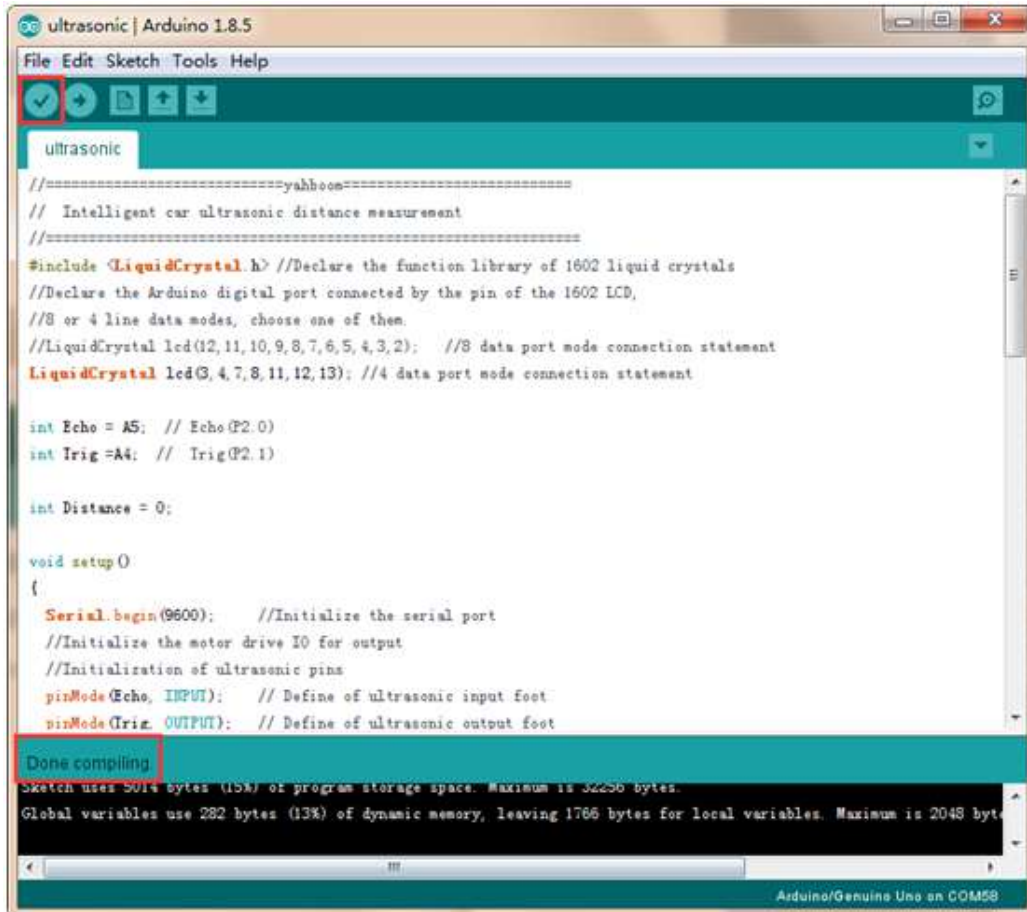
//Declare the Arduino digital port connected by the pin of the 1602 LCD,
//8 or 4 line data modes, choose one of them.
//LiquidCrystal lcd(12,11,10,9,8,7,6,5,4,3,2); //8 data port mode connection statement
LiquidCrystal lcd(3,4,7,8,11,12,13); //4 data port mode connection statement
int Echo = A5; // Echo(P2.0)
int Trig =A4; // Trig(P2.1)
int Distance = 0;
void setup()
{
  Serial.begin(9600); //Initialize the serial port
  //Initialize the motor drive IO for output
  //Initialization of ultrasonic pins
  pinMode(Echo, INPUT); // Define of ultrasonic input foot
  pinMode(Trig, OUTPUT); // Define of ultrasonic output foot
  lcd.begin(16,2); //Initialization of 1602 liquid crystal working mode
  //Define the 1602 LCD display range of 2 lines and 16 columns
}
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 Continuously
  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()
{
  Distance_test();
  if((2<Distance)&(Distance<400))//Range of ultrasonic distance ranging from 2cm to
400cm
  {
    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();
}
```

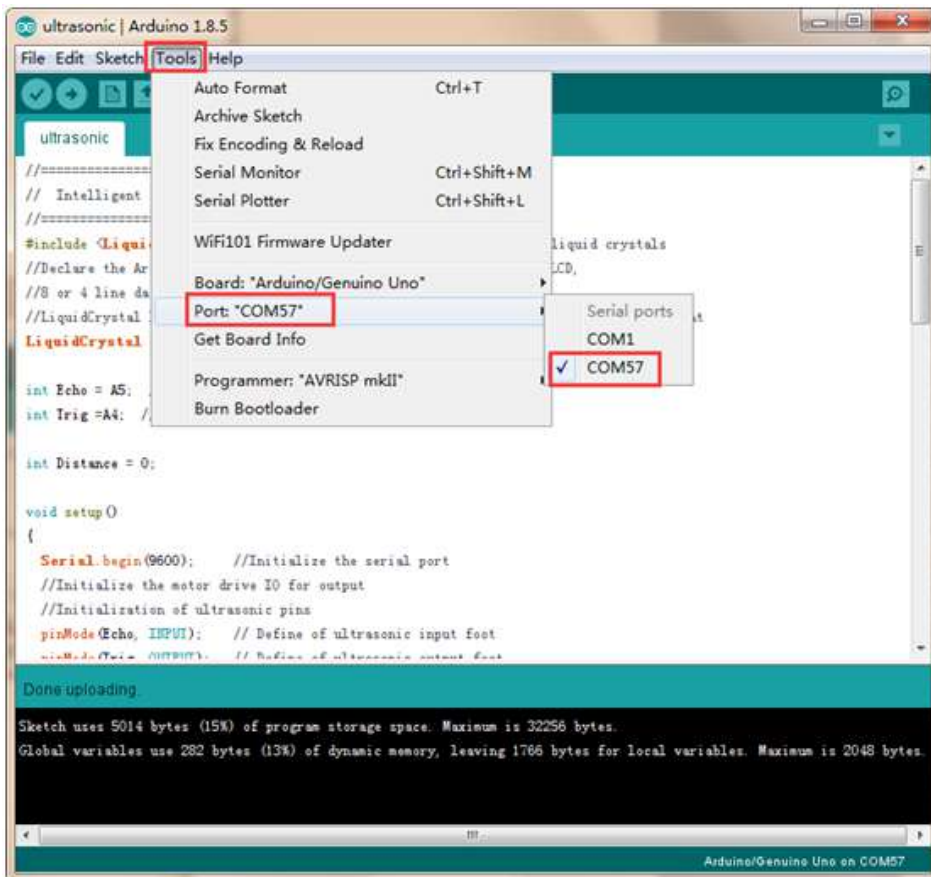
Experimental steps:

1. We need to open the code of this experiment: **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.

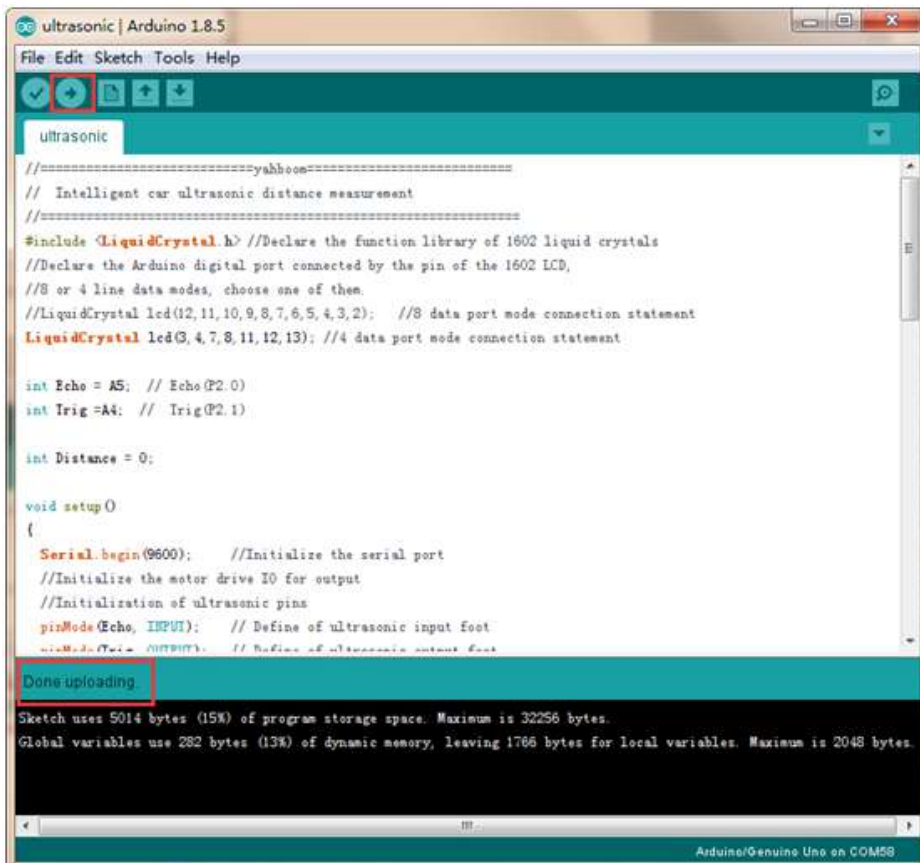


2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below.

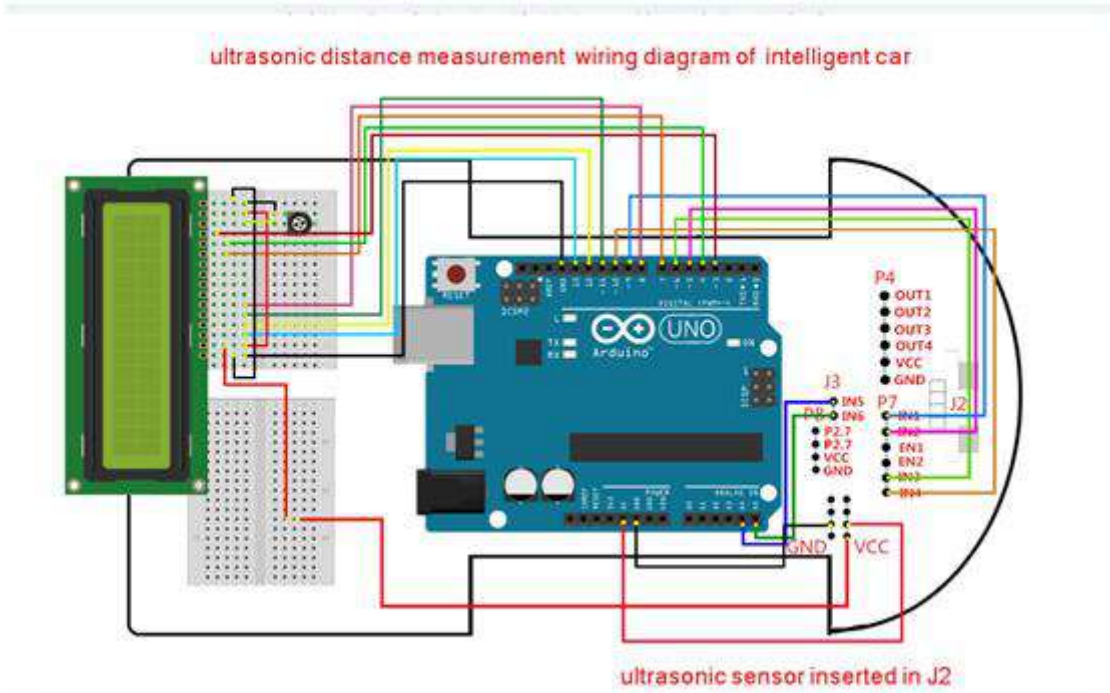




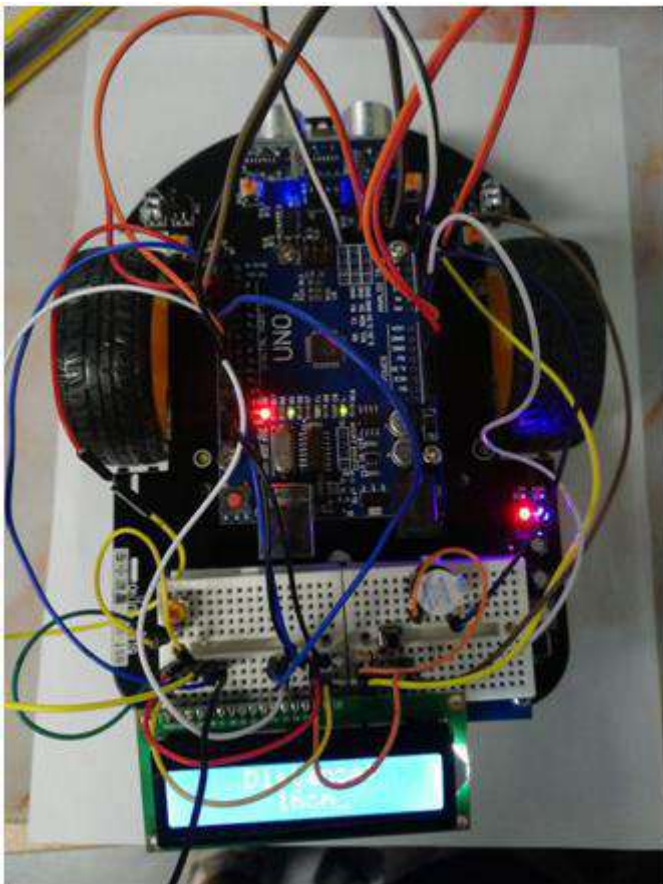
3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “**Done uploading**” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



4. Please wire the Smart Car as shown below.



5. Insert the ultrasonic sensor directly into the bottom plate of the smart car, turn on the power switch, adjust the adjustable resistance until the liquid crystal can display normally, you can see the value of the ultrasonic sensor measurement displayed on the liquid crystal.



10- Ultrasonic obstacle avoidance (no servo)

The purpose of the experiment:

Use the carton to simulate obstacles on the ground, put the smart car that has uploaded the program **avoid.ino** on the ground, press the start button of the tail of the car, the LCD screen of the car shows the distance measured by the ultrasonic sensor and starts to avoid the obstacle ahead.

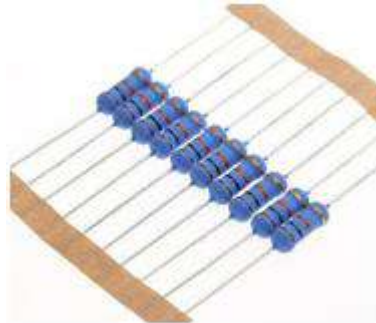
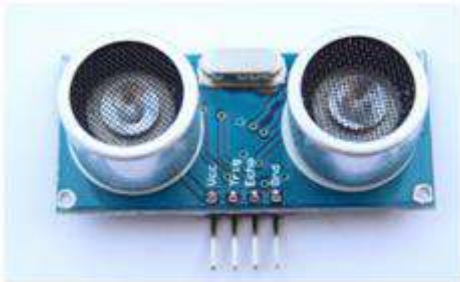
Precautions:

1. If the LCD is not displayed, use a screwdriver to adjust the adjustable resistor.
2. If only the ultrasonic obstacle avoidance function is used, the display distance is not required, and the LCD1602 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





Experimental code analysis:

```
//=====yahboom=====
//Intelligent car ultrasonic obstacle avoidance(no servo)
//=====
#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 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
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,100);//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,100);//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)
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,100);
    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,100);
    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,100); //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,100);
  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,100); //PWM ratio 0~255 speed control
  digitalWrite(Left_motor_go,HIGH); //left motor go
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100);
  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,100); //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 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 Continuously
    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 Distance_display()
{
    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 loop()
{
    keysacn();
    while(1)
    {
        Distance_test();             // Measuring the distance ahead
        Distance_display();           //LCD screen display distance
        if(Distance < 60)            //The value of the distance to the obstacle can be set according to
        the actual situation.
            while(Distance < 60)// Judge whether there is an obstacle, again

```

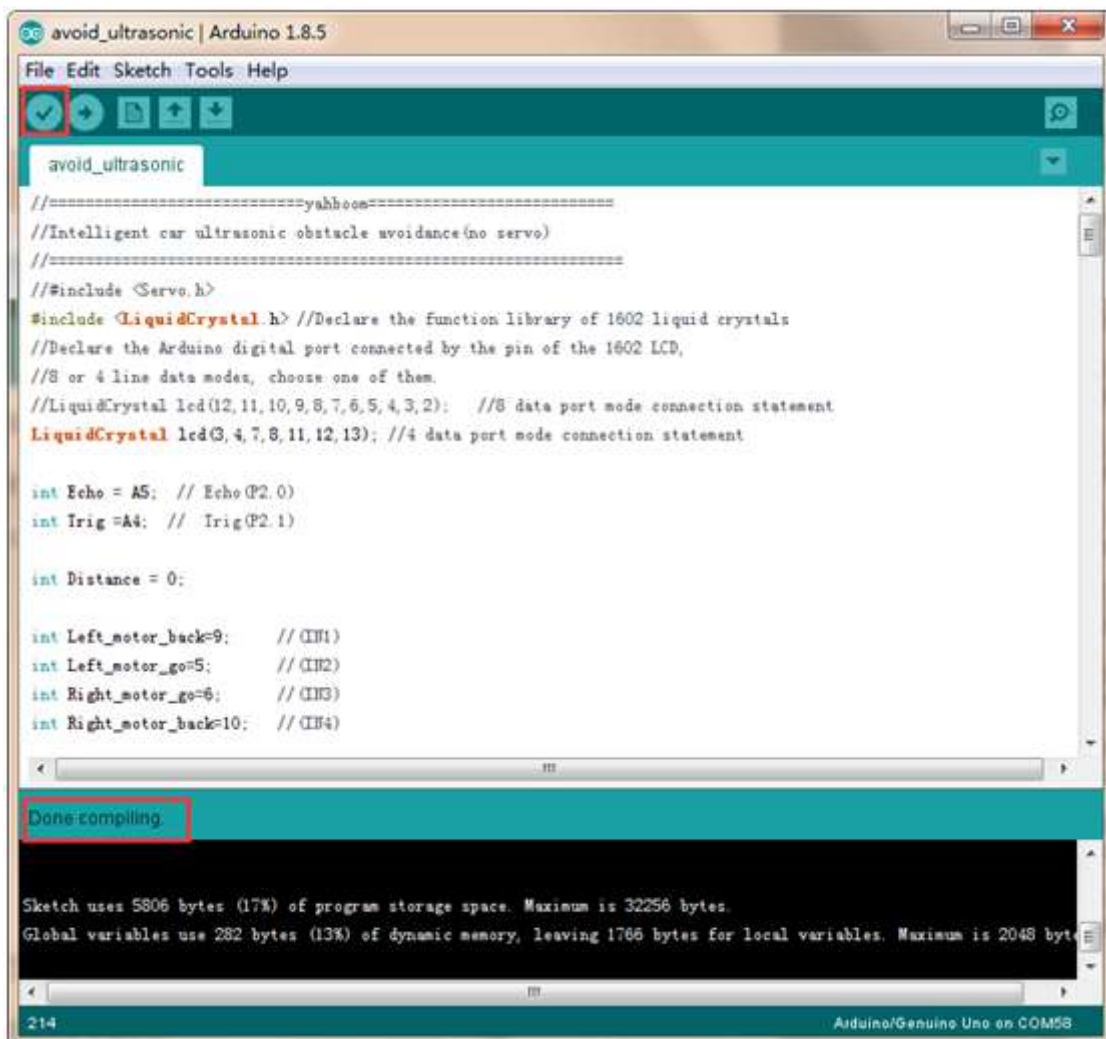
```

//if there is a turning direction, continue to judge
{
  right(1);
  brake(1);
  Distance_test(); // Measuring the distance ahead
  Distance_display();//LCD screen display distance
}
else
  run();//No obstacle, run
}
}
}

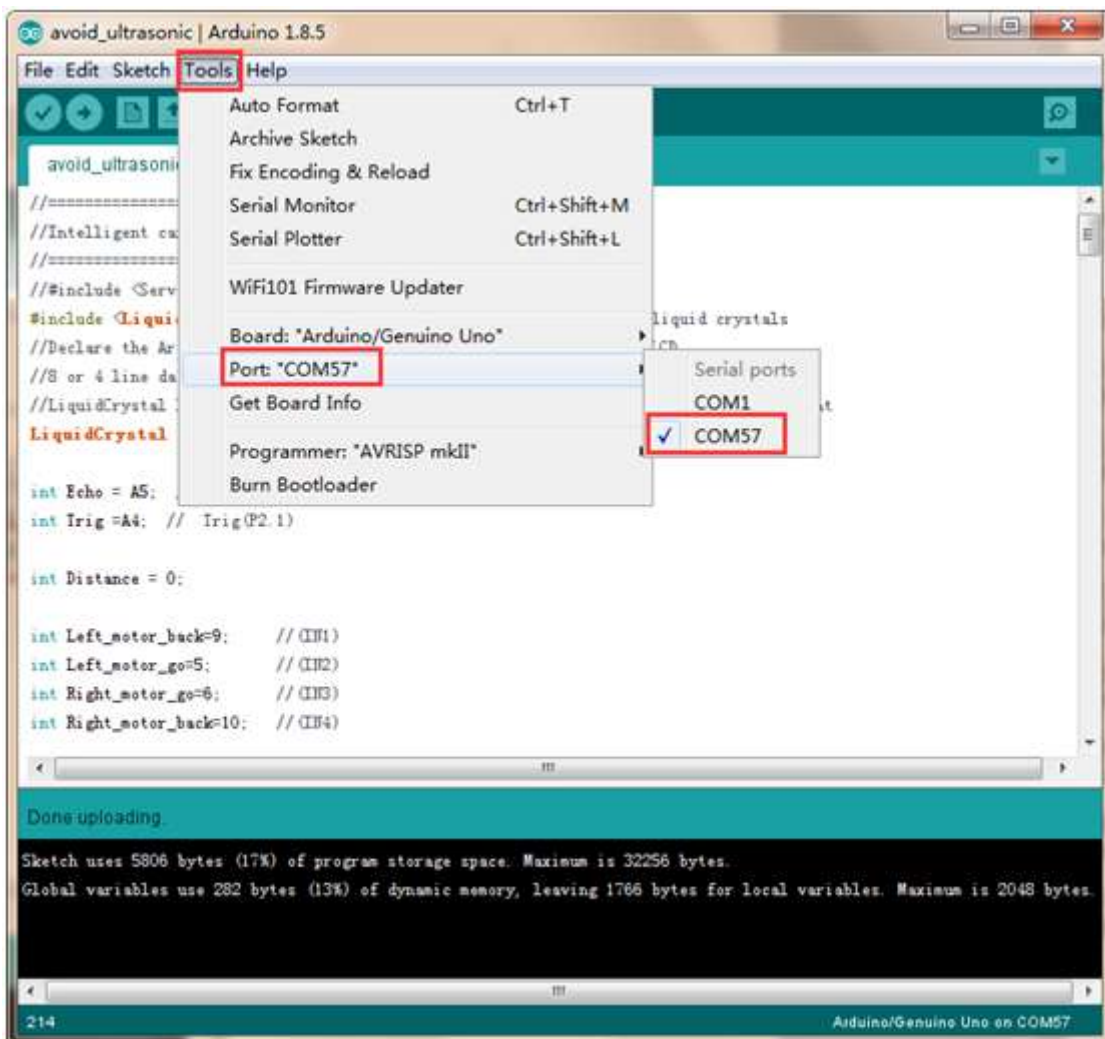
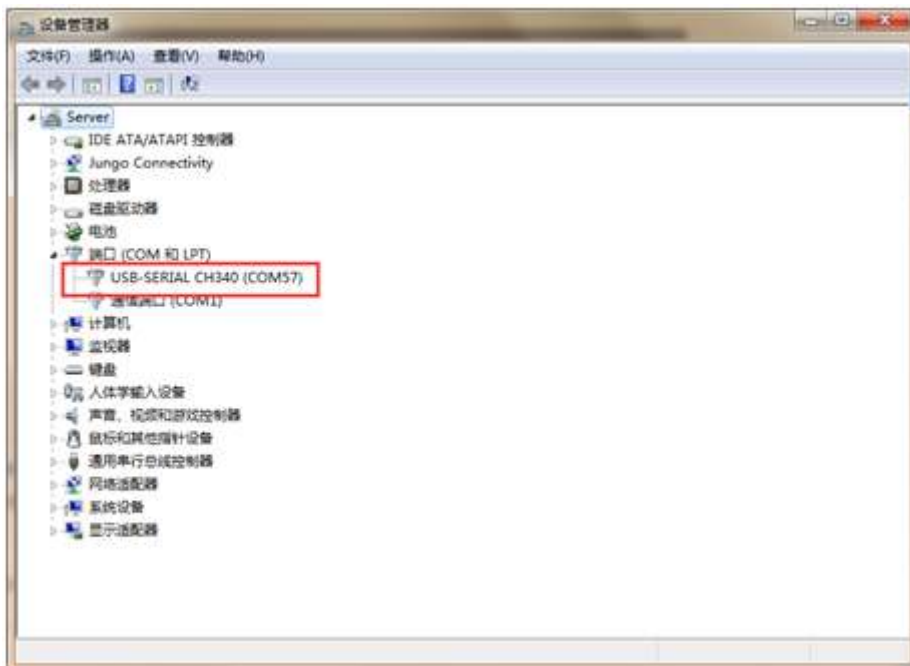
```

Experimental steps:

1. We need to open the code of this experiment: **avoid_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.



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.



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.

```

avoid_ultrasonic | Arduino 1.8.5
File Edit Sketch Tools Help
avoid_ultrasonic
//=====yahboom=====
//Intelligent car ultrasonic obstacle avoidance(no servo)
//=====
#include <Servo.h>
#include <LiquidCrystal.h> //Declares 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 Irig =A4; // Irig(P2.1)

int Distance = 0;

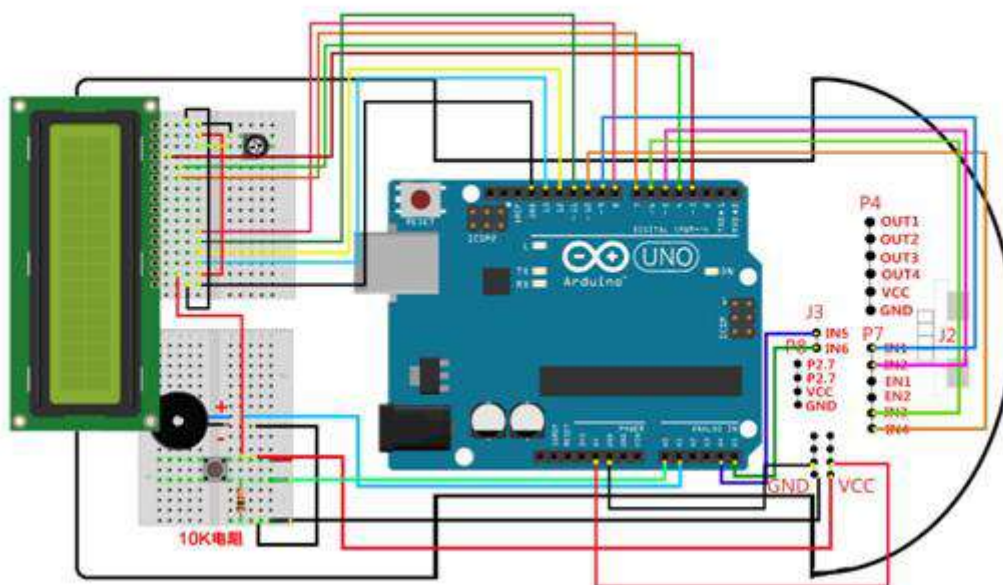
int Left_motor_back=9; // (III1)
int Left_motor_go=5; // (III2)
int Right_motor_go=6; // (III3)
int Right_motor_back=10; // (III4)

Done uploading.
Sketch uses 5806 bytes (17% of program storage space. Maximum is 32256 bytes.
Global variables use 282 bytes (13% of dynamic memory, leaving 1766 bytes for local variables. Maximum is 2048 bytes.
214 Arduino/Genuine Uno on COM5B

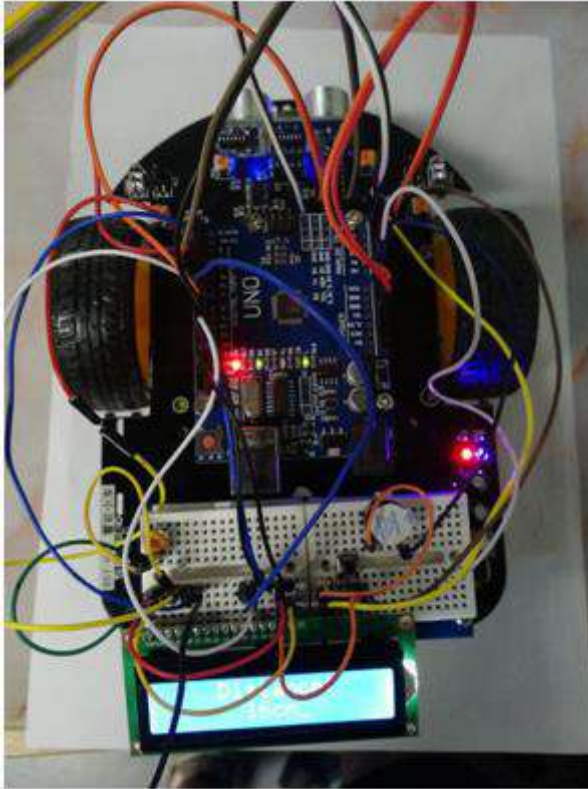
```

4. Please wire the Smart Car as shown below.

Ultrasonic obstacle avoidance(no servo)



5. Put the smart car that has uploaded the program **avoid_ultrasonic.ino** on the ground, press the start button of the tail of the car, the LCD screen of the car shows the distance measured by the ultrasonic sensor and starts to avoid the obstacle ahead.



11- 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.

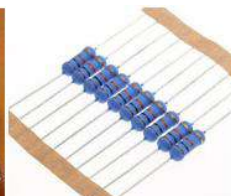
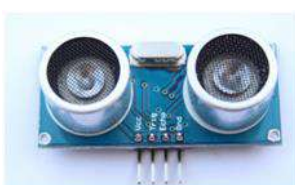
Precautions:

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



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 Continuously
    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;
    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);

```

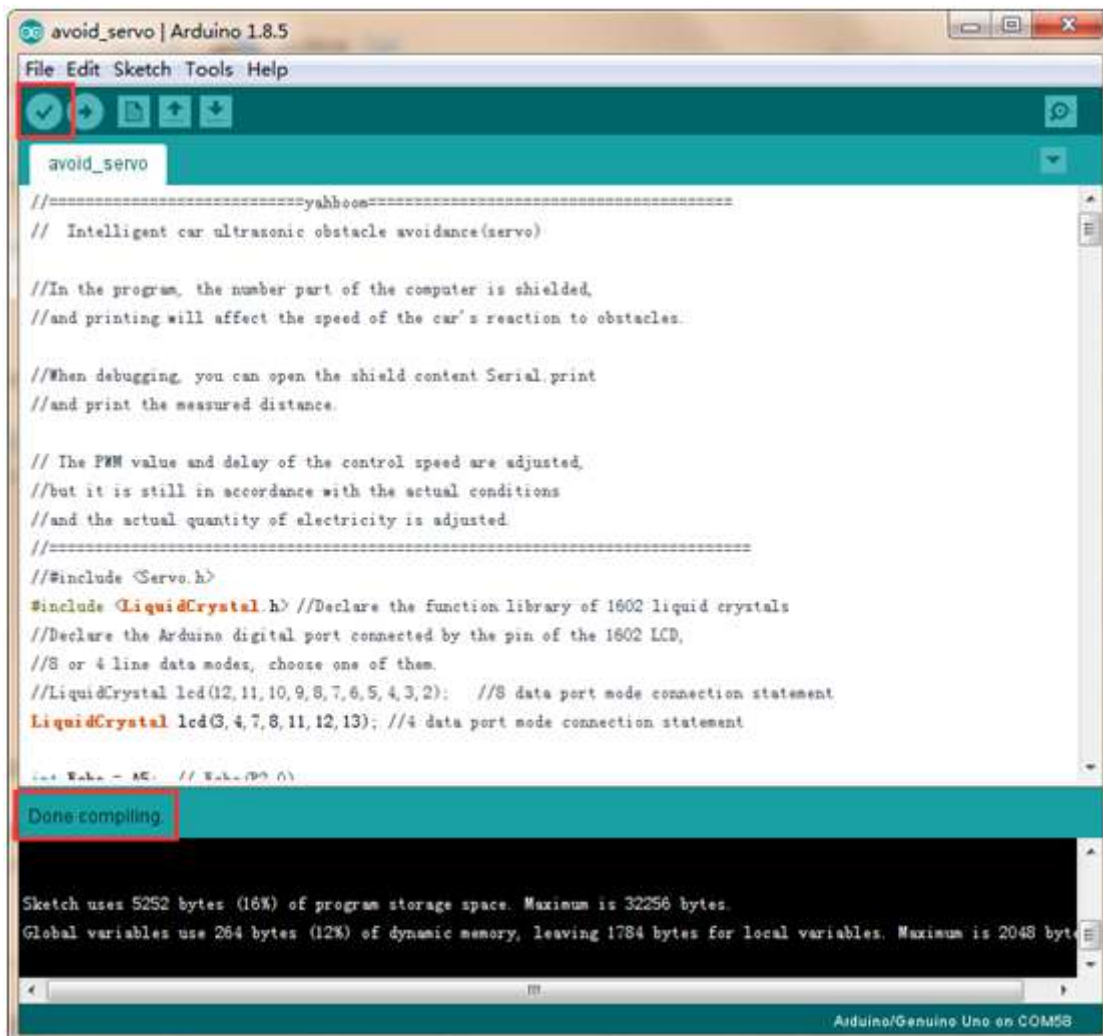
```

    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
}
}
}
}

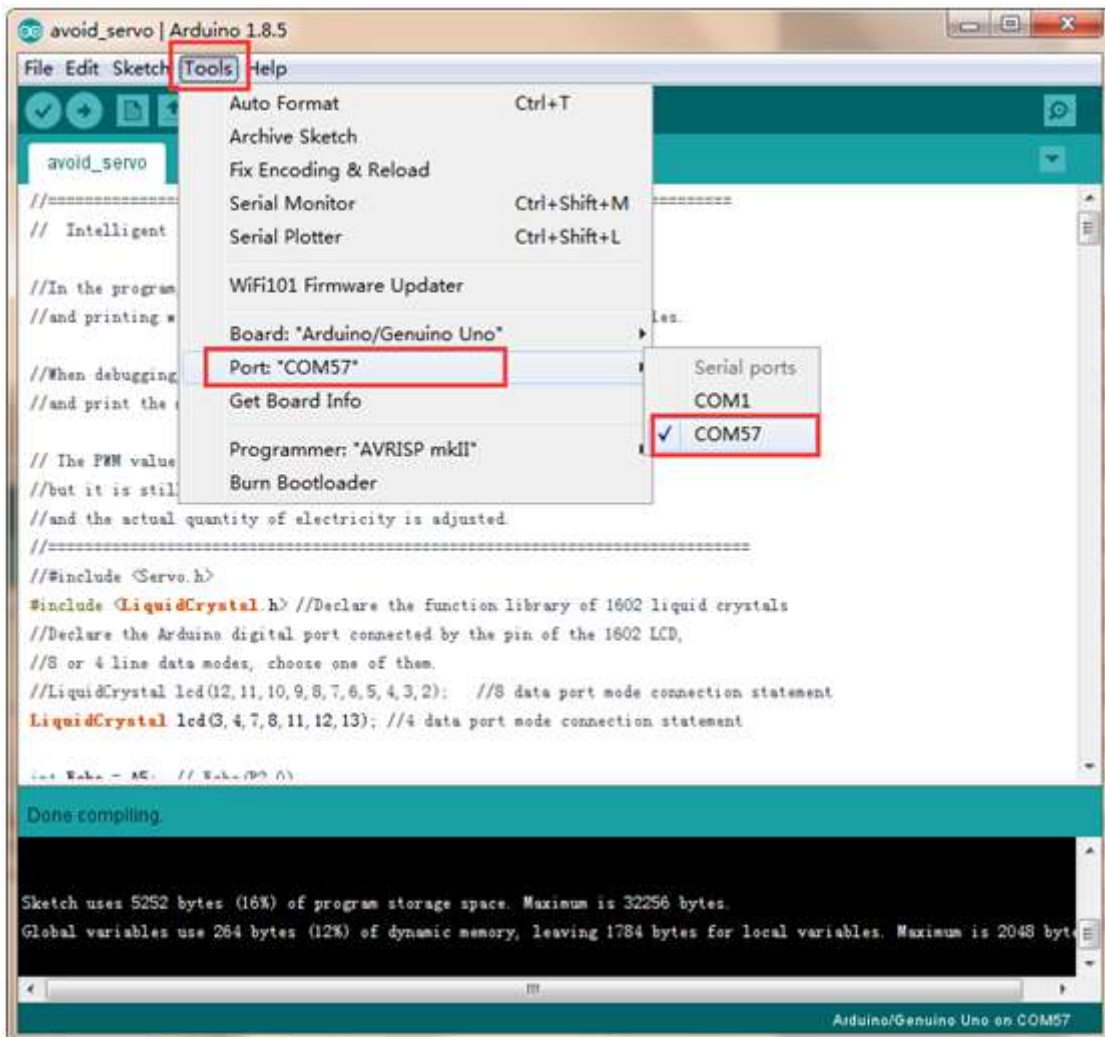
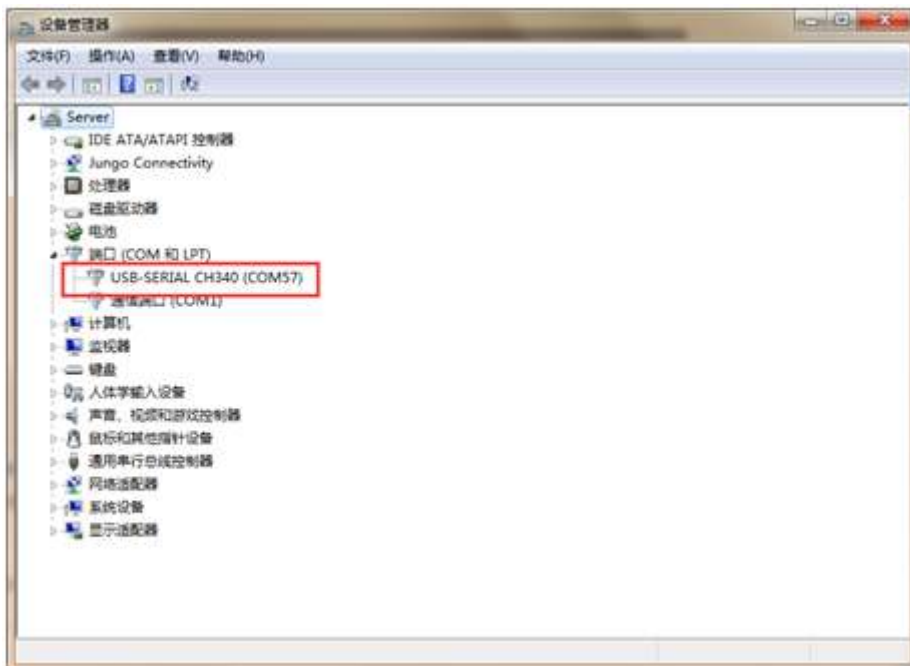
```

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.



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.

```

avoid_servo | Arduino 1.8.5
File Edit Sketch Tools Help
avoid_servo
=====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> //Declares 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

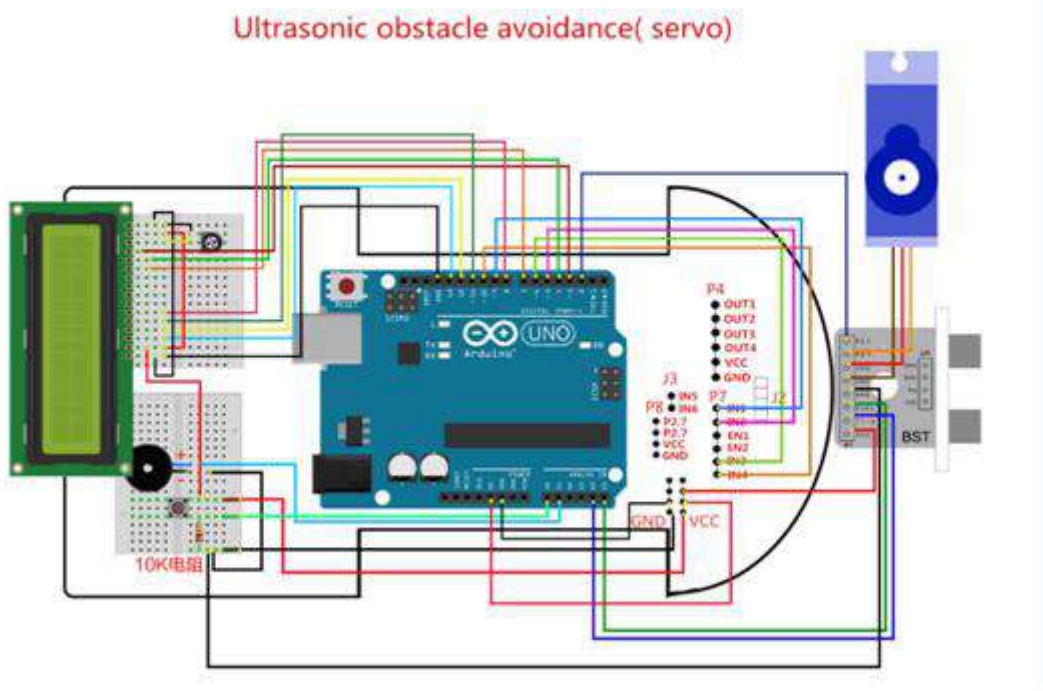
... Yaha - AC; // Yaha-(P2 0)

Done uploading
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 bytes.

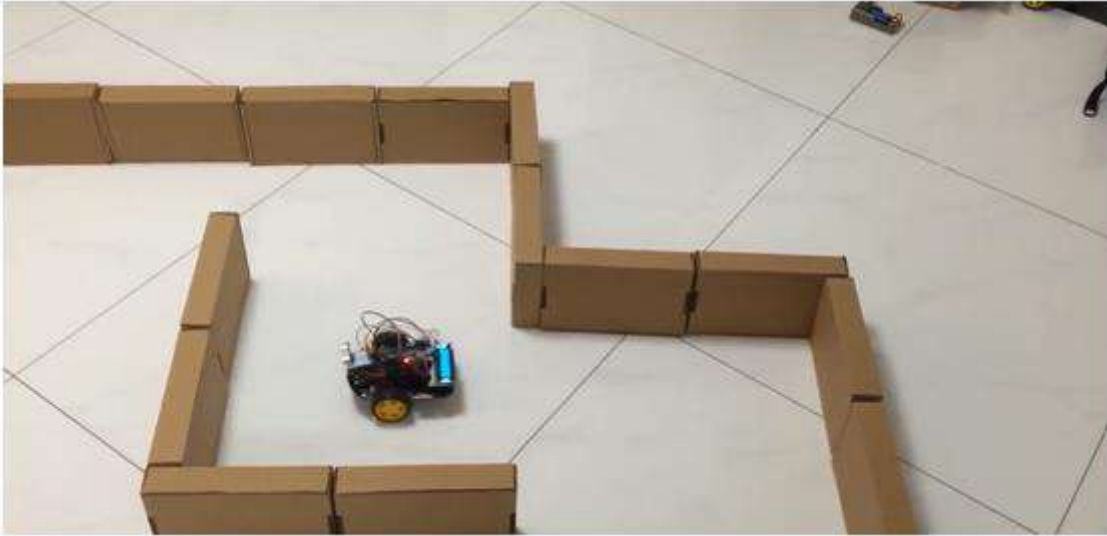
Arduino/Genuine Uno on COM5B

```

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.



12- Infrared remote control

The purpose of the experiment:

After uploading the Remote_contorl.ino program, place the car indoors and pull the curtains to block the outdoor lights. Align the infrared emitter of the infrared remote control with the infrared receiver at the rear of the Smart Car, then press the numeric keypad of the infrared remote control to control the Smart Car to complete the corresponding action.

Precautions:

1. Incorrect connection of the infrared receiving head will cause the receiving head to burn out, so this test must be strictly wired according to the wiring diagram.
2. This experiment requires the use of an infrared remote control. Remove the insulating plastic sheet from the bottom of the remote control before use.

List of components required for the experiment:

- Arduino Smart Car* 1
- USB cable* 1
- DuPont Line* 9
- Infrared receiver * 1
- Infrared remote control* 1



Experimental code analysis:

```
//=====yahboom=====
//
// Intelligent car IRemote contorl
//In the experiment, the received infrared signal is used as the signal for the distribution
remote control,
//and the signal value can be printed out with other infrared signals control.
//The speed of the motor can not be adjusted in this experiment.
//The adjustment of the PWM value will affect the signal reception of the infrared
```

```
//=====
=====
#include <IRremote.h>
int RECV_PIN = A4;//declare port
IRrecv irrecv(RECV_PIN);
decode_results results;//declare structure
int on = 0;//Marker bit
unsigned long last = millis();
long run_car = 0x00FF18E7; //key 2
long back_car = 0x00FF4AB5; //key 8
long left_car = 0x00FF10EF; //key 4
long right_car = 0x00FF5AA5; //key 6
long stop_car = 0x00FF38C7; //key 5
long left_turn = 0x00ff30CF; //key 1
long right_turn = 0x00FF7A85;//key 3
//=====
int Left_motor_back=9; // (IN1)
int Left_motor_go=5; // (IN2)
int Right_motor_go=6; // (IN3)
int Right_motor_back=10; // (IN4)
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(13, OUTPUT); //Define the key interface for the input interface
  Serial.begin(9600); //baud rate 9600
  irrecv.enableIRIn(); // Start the receiver
}
void run()
{
  digitalWrite(Right_motor_go,HIGH); //right motor go
  digitalWrite(Right_motor_back,LOW);
  //analogWrite(Right_motor_go,200);//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,200);//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()
{
  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()      //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() //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()     //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() //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,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()
{

```

```

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 dump(decode_results *results)
{
  int count = results->rawlen;
  if (results->decode_type == UNKNOWN)
  {
    //Serial.println("Could not decode message");
    brake();
  }
  /*Serial port printing, debugging can be opened,
  //the actual operation will affect the speed of reaction, it is recommended to shield
  */
  else
  {
    if (results->decode_type == NEC)
    {
      Serial.print("Decoded NEC: ");
    }
    else if (results->decode_type == SONY)
    {
      Serial.print("Decoded SONY: ");
    }
    else if (results->decode_type == RC5)
    {
      Serial.print("Decoded RC5: ");
    }
    else if (results->decode_type == RC6)
    {
      Serial.print("Decoded RC6: ");
    }
    Serial.print(results->value, HEX);
    Serial.print(" ");
    Serial.print(results->bits, DEC);
    Serial.println(" bits");
  }
  Serial.print("Raw (");
  Serial.print(count, DEC);
  Serial.print("): ");
  for (int i = 0; i < count; i++)
  {
    if ((i % 2) == 1)
    {

```

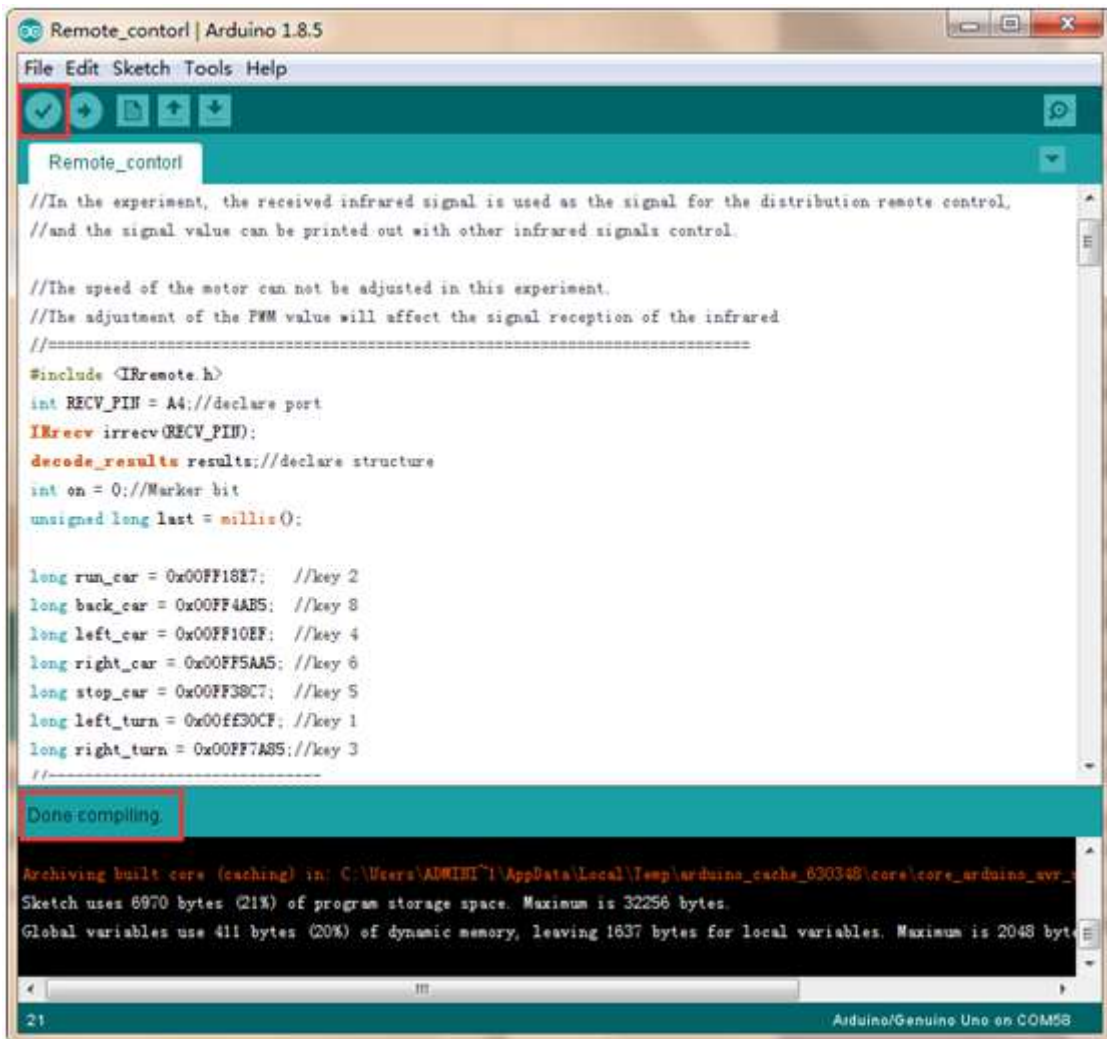
```

    Serial.print(results->rawbuf[i]*USECPERTICK, DEC);
  }
  else
  {
    Serial.print(-(int)results->rawbuf[i]*USECPERTICK, DEC);
  }
  Serial.print(" ");
}
Serial.println("");
*/
}
void loop()
{
  if (irrecv.decode(&results)) //Call library function: decode
  {
    // If it's been at least 1/4 second since the last
    // IR received, toggle the relay
    if (millis() - last > 250) //Determine the received signal
    {
      on = !on;
      digitalWrite(13, on ? HIGH : LOW);
      //The signal is received on the board, led_twinkle
      dump(&results); //Decoded infrared signal
    }
    if (results.value == run_car) //key 2
      run();
    if (results.value == back_car) //key 8
      back();
    if (results.value == left_car) //key 4
      left(); //turn left
    if (results.value == right_car) //key 6
      right(); //turn right
    if (results.value == stop_car) //key 5
      brake();
    if (results.value == left_turn) //key 1
      spin_left(); //left rotation
    if (results.value == right_turn) //key 3
      spin_right(); //right rotation
    last = millis();
    irrecv.resume(); // Receive the next value
  }
}

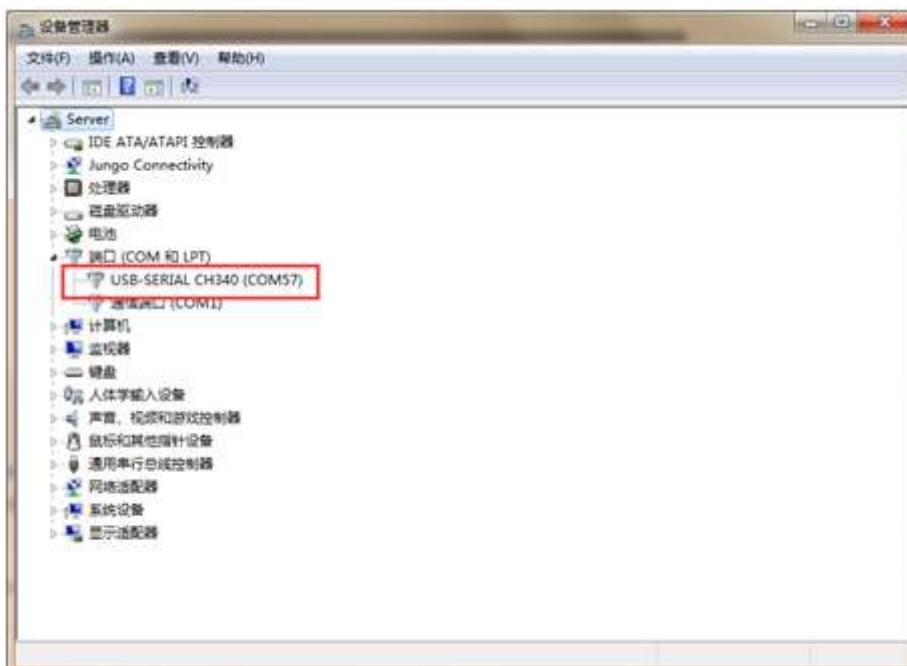
```

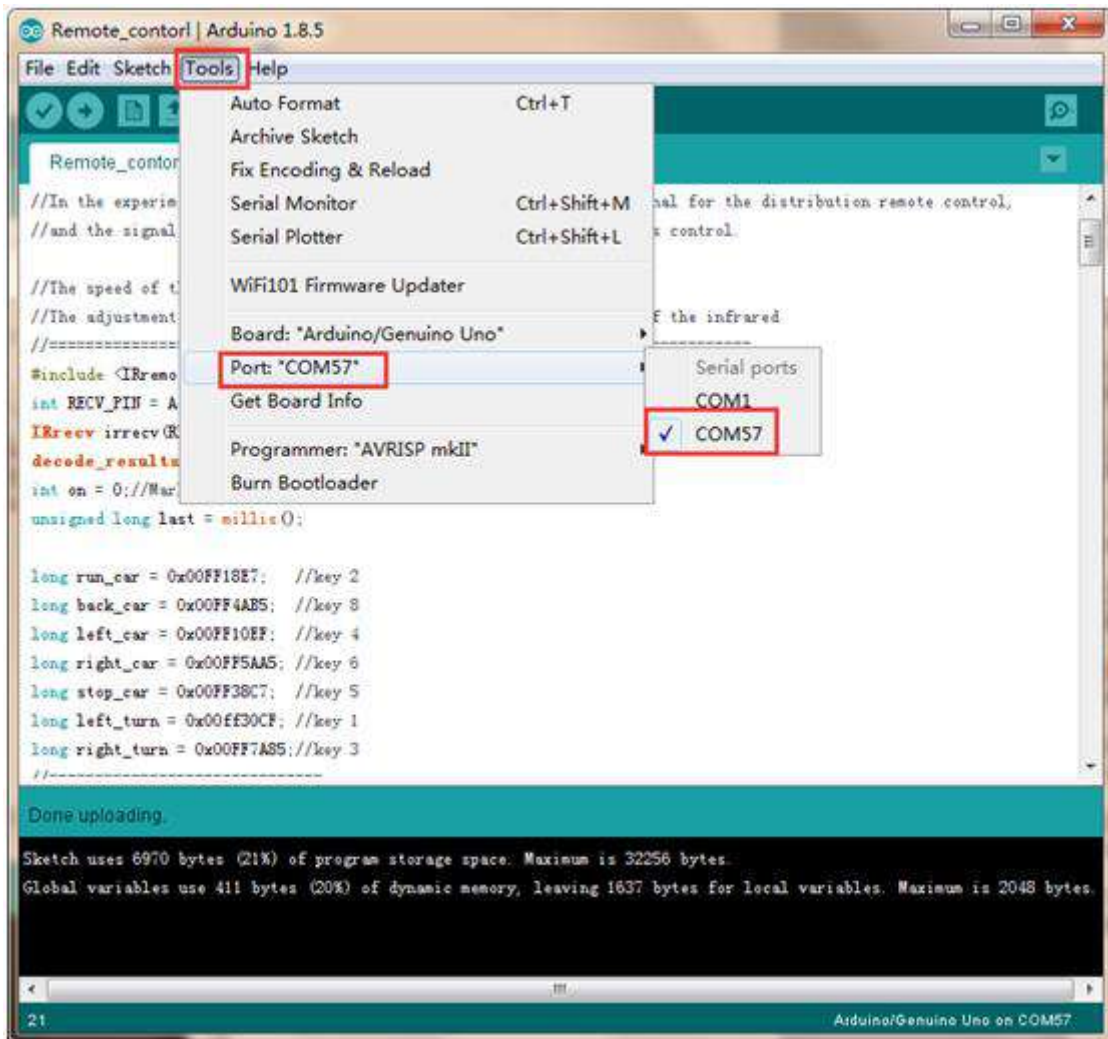
Experimental steps:

1. We need to open the code of this experiment: **Remote_contorl.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.

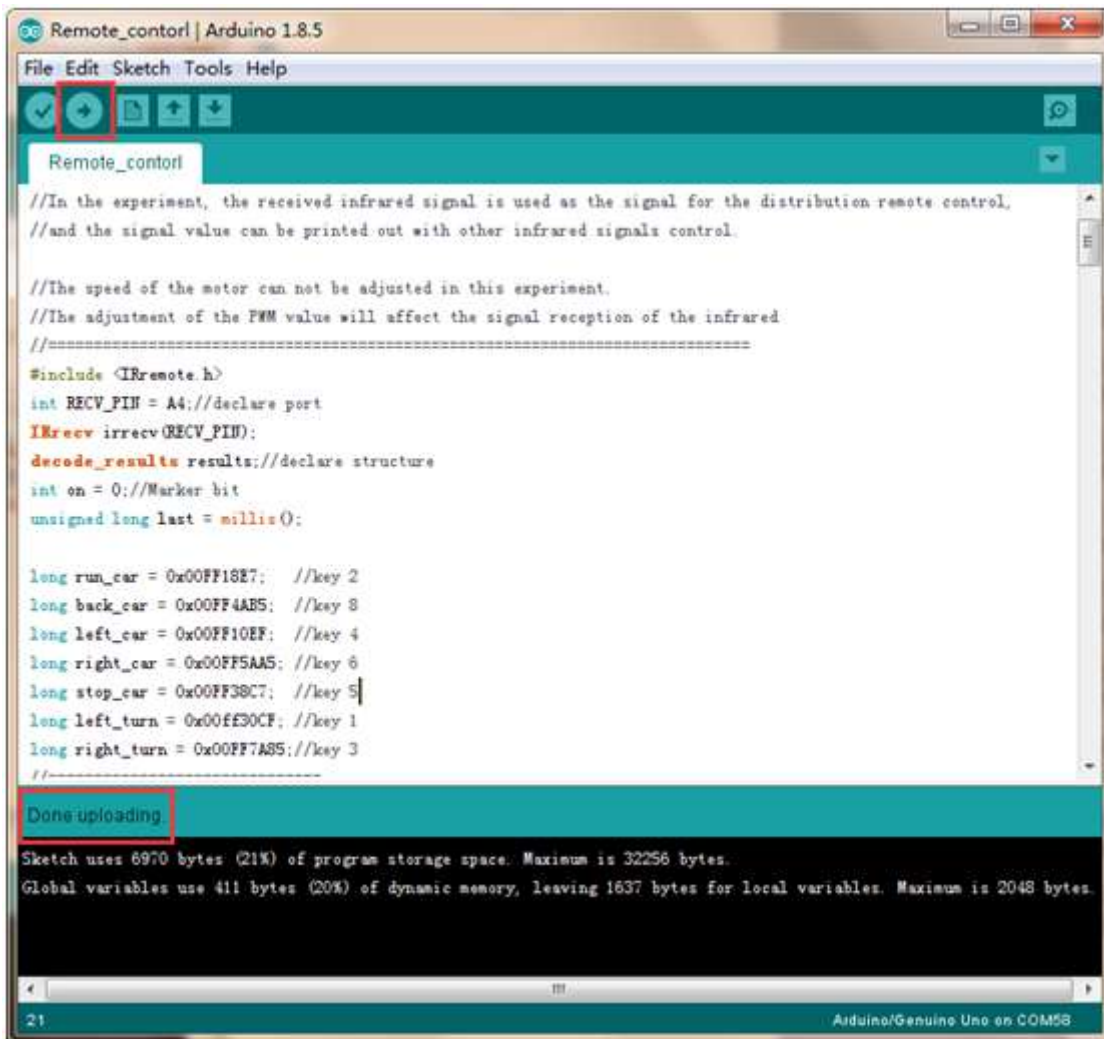


2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below.

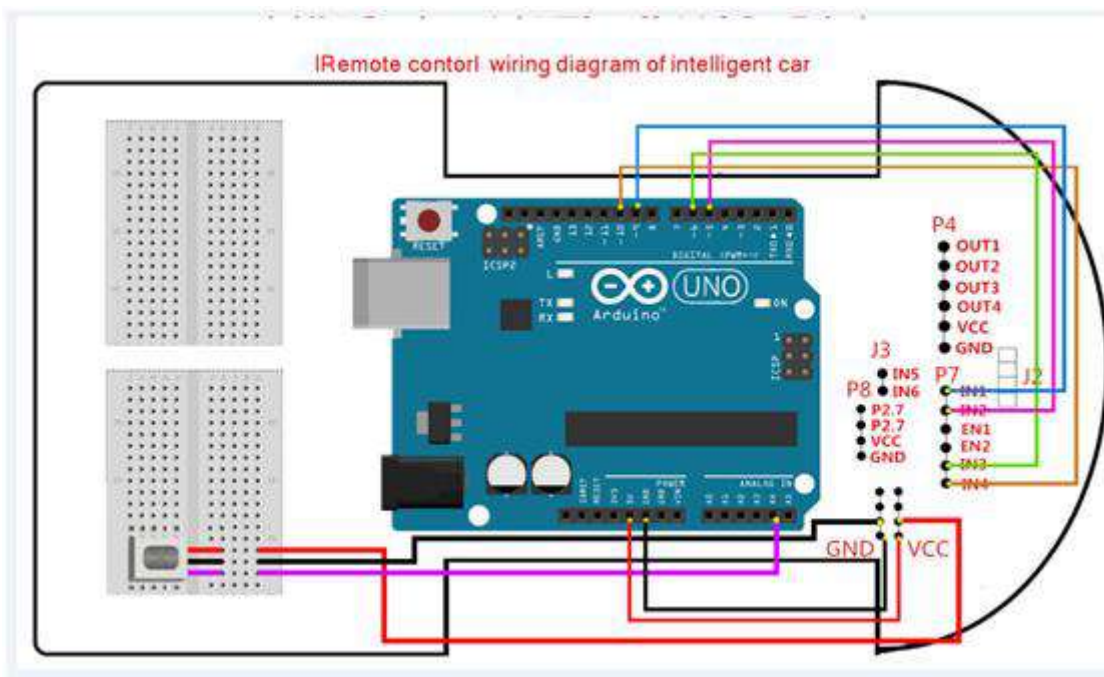




3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “**Done uploading**” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



4. Please wire the Smart Car as shown below.



5. This experiment requires the use of an infrared remote control. Before use, please remove the insulated plastic sheet at the bottom of the remote control. The numbers

2,8,4,6 on the remote control correspond to advance, return back, turn left and turn right; 1,3 corresponds to left and right rotation; 5 is the stop button.

6. The following is the user code value corresponding to the infrared remote control.

Corresponding user code value	The program controls the action of the BatCar	Remote control button
0x00FF9867	No control action	—
0x00FFB04F	No control action	C
0x00ff30CF	Left rotation	1
0x00FF18E7	Forward	2
0x00FF7A85	Right rotation	3
0x00FF10EF	Turn left	4
0x00FF38C7	Brake	5
0x00FF5AA5	Turn right	6
0x00FF42BD	No control action	7
0x00FF4AB5	Backward	8
0x00FF52AD	No control action	9

User code : 00FF



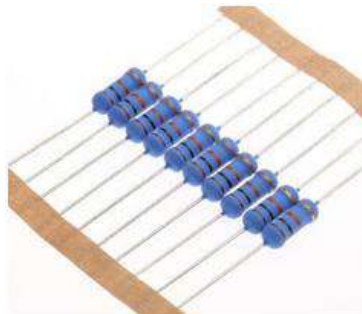
13- Tracking ultrasonic

The purpose of the experiment:

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.

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 is released
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 Continuously
float Fdistance = pulseIn(Echo, HIGH); //Reading high level time(unit: us)
Fdistance= Fdistance/58; // Y meter = (X second *344) /2
}

```

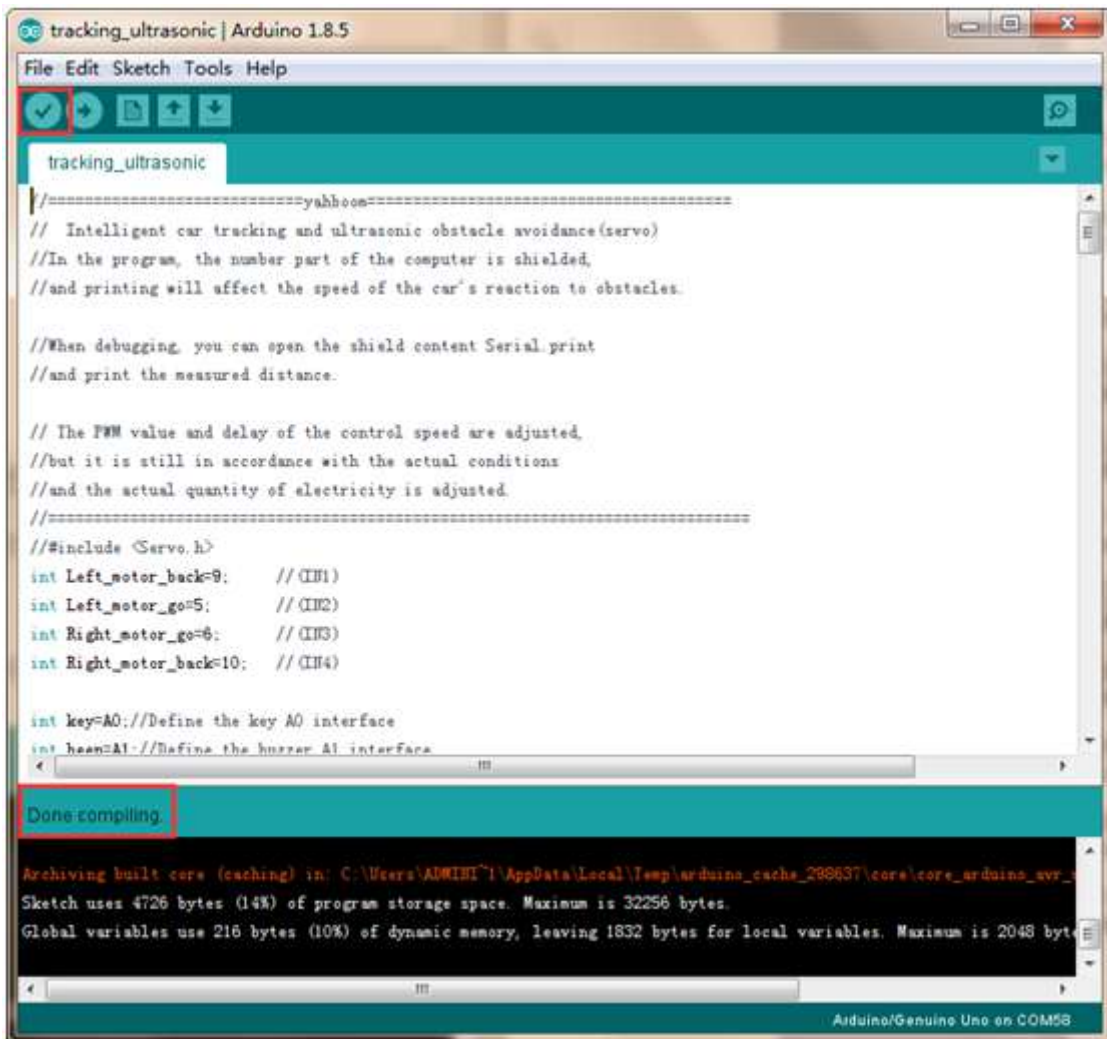
```

// 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();
    }
  }
}

```

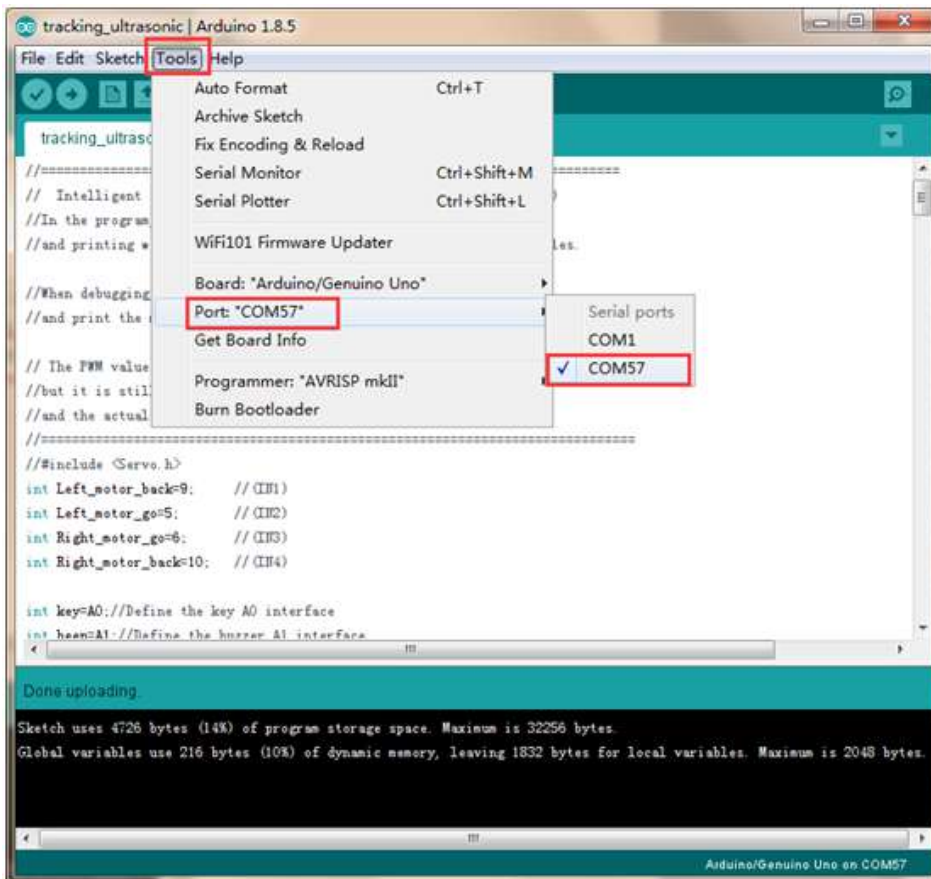
Experimental steps:

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.

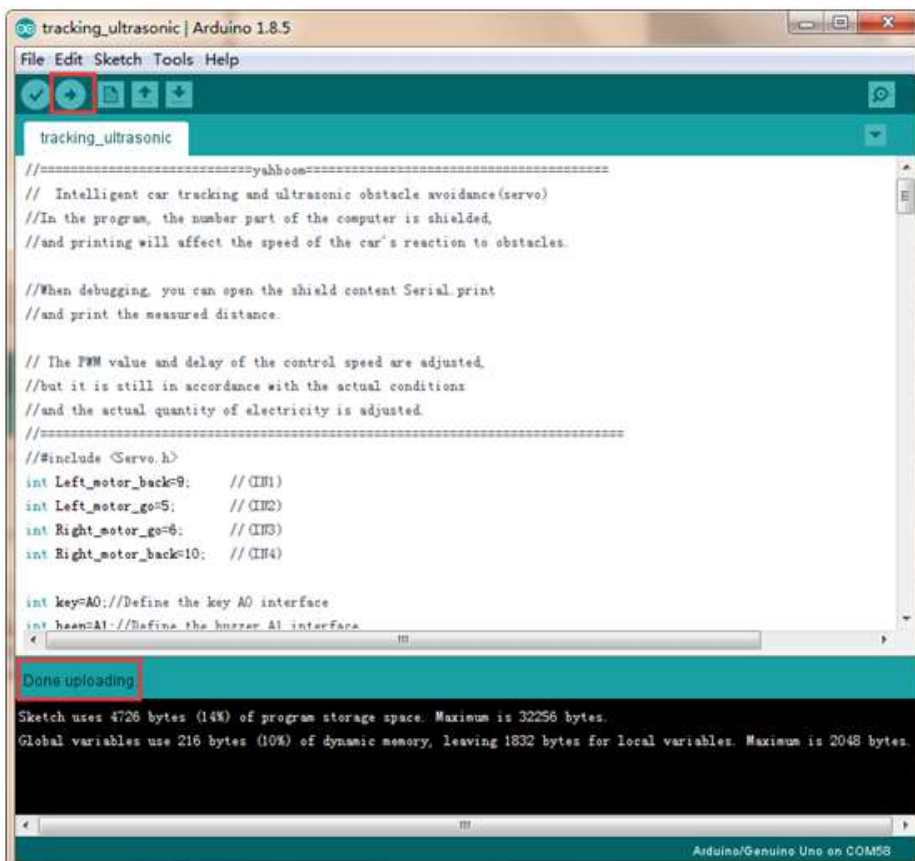


2. In the menu bar of Arduino IDE, we need to select **【Tools】** --- **【Port】** --- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below.

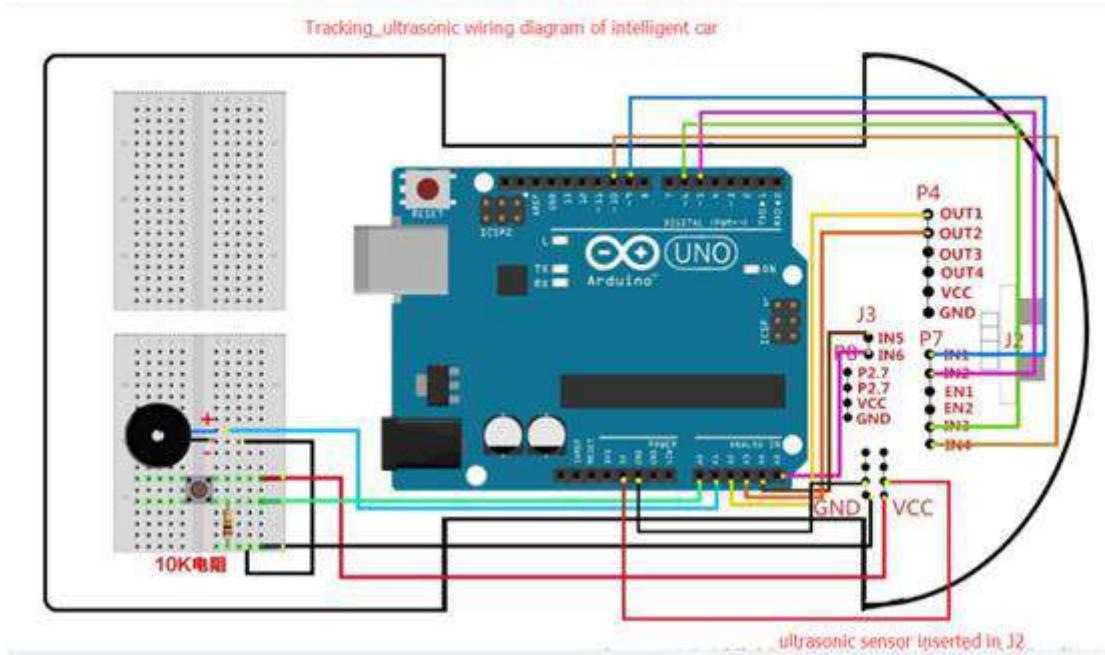




3. After the selection is completed, you need to click “→” under the menu bar to upload the code to the Arduino UNO board. When the word “**Done uploading**” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



4. Please wire the Smart Car as shown below.



5. 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.

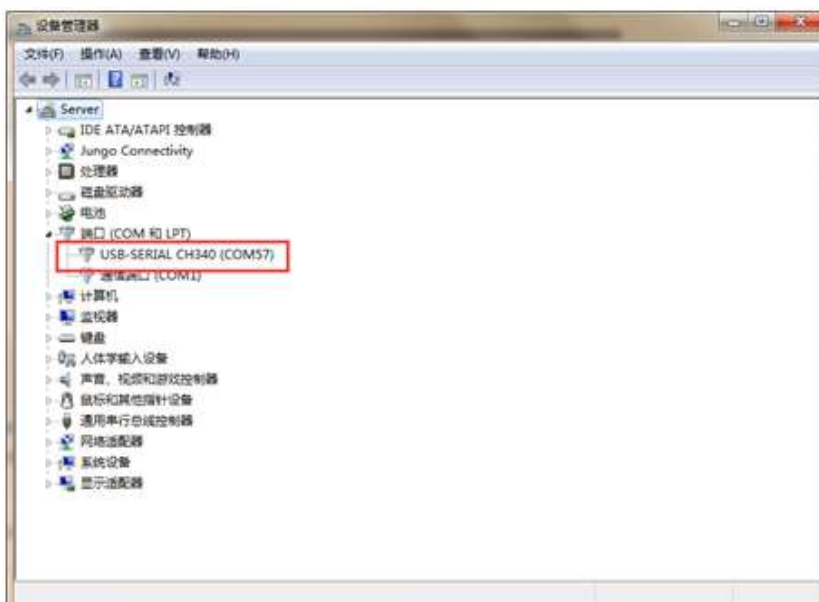
14- Bluetooth control

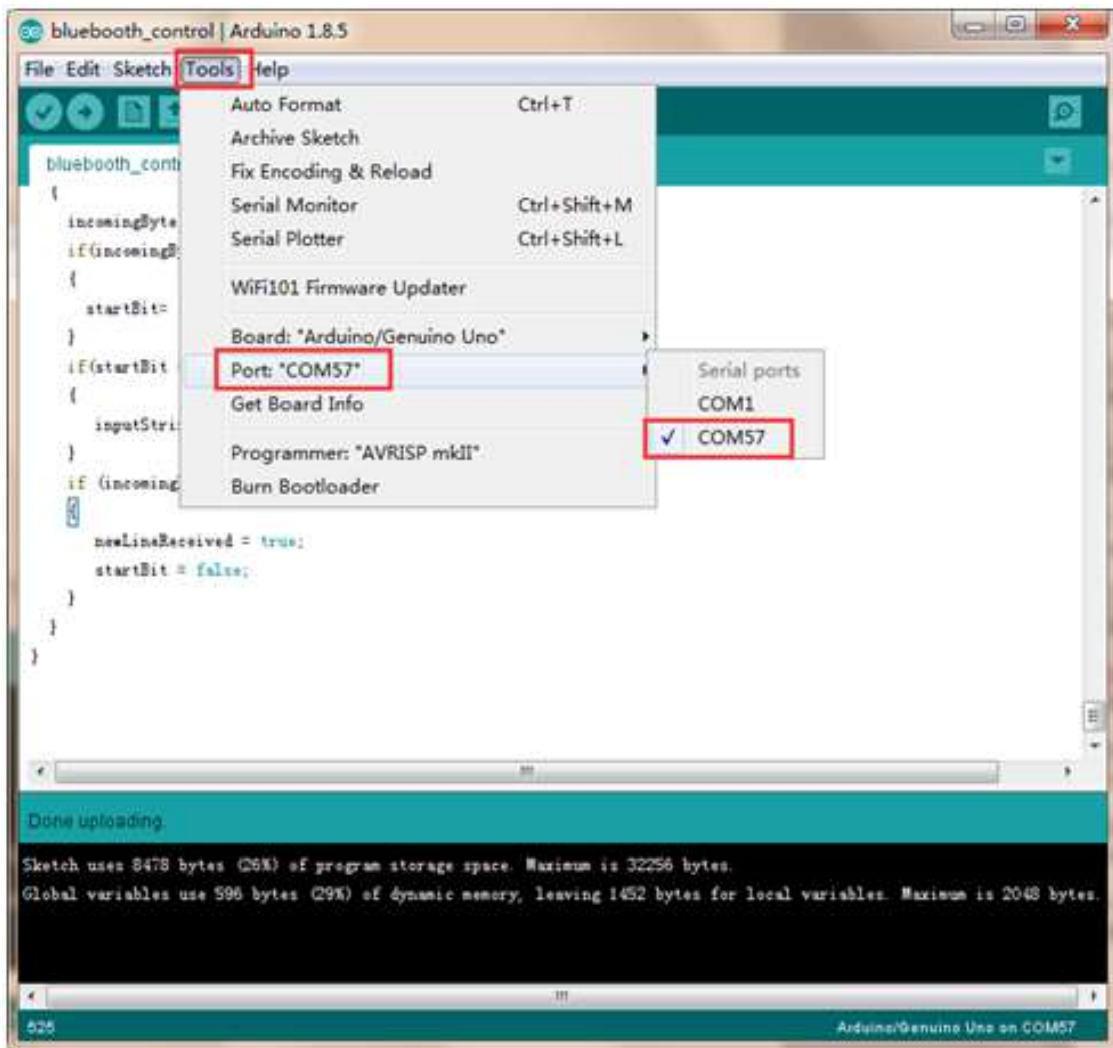
Experimental steps:

1. We need to open the code of this experiment: **bluebooth_control.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.





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.

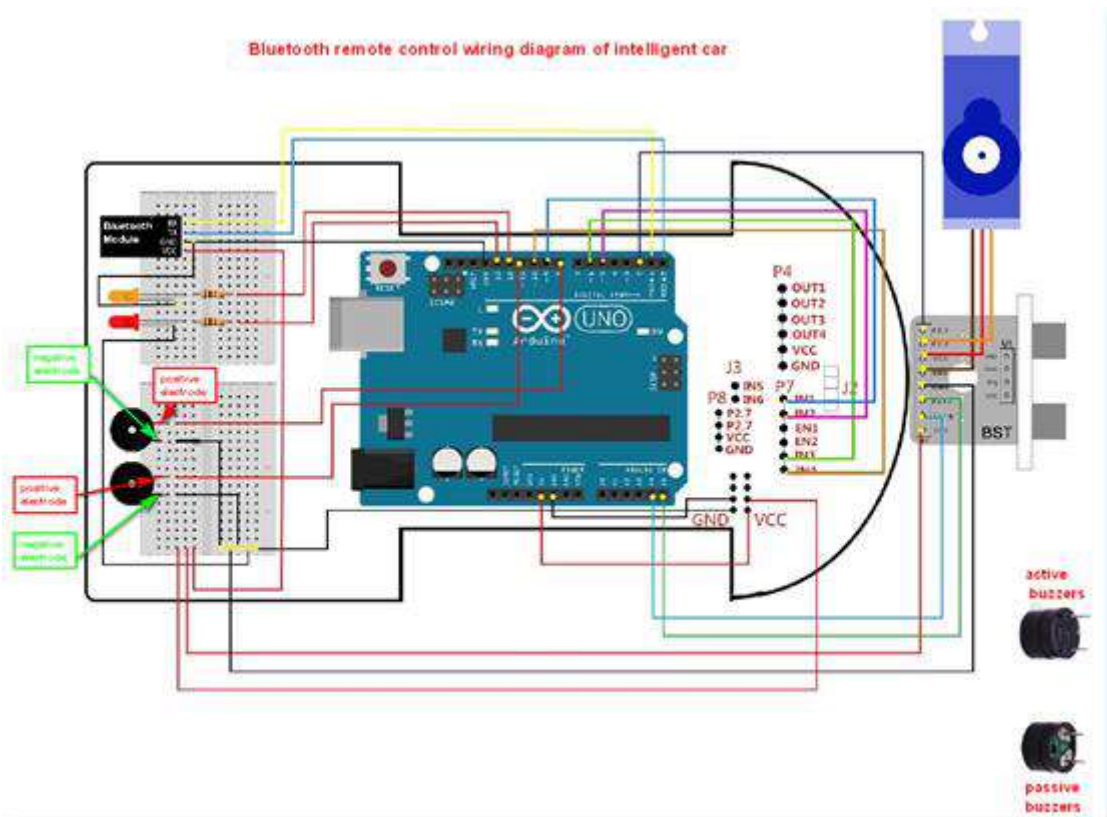
```

blueooth_control | Arduino 1.8.5
File Edit Sketch Tools Help
blueooth_control
{
  incomingByte = Serial.read();
  if(incomingByte == '$')
  {
    startBit= true;
  }
  if(startBit == true)
  {
    inputString += (char) incomingByte;
  }
  if (incomingByte == '#')
  {
    newLineReceived = true;
    startBit = false;
  }
}
}
}

Done uploading
Sketch uses 8478 bytes (26%) of program storage space. Maximum is 32256 bytes.
Global variables use 596 bytes (29%) of dynamic memory, leaving 1452 bytes for local variables. Maximum is 2048 bytes.
525 Arduino/Genuino Uno on COM5B

```

4. Please wire the Smart Car as shown below.



5. As the Bluetooth module and the serial port of the writer share the IO ports 0 and 1, it will cause failure in burning. Please remove the VCC of the Bluetooth module power supply before burning, and then supply power to the Bluetooth module after finishing.

6.

Arduino smart car(with bluetooth)

1.Package list

Arduino smart car(standard) + Bluetooth module

2.Introduction

The Arduino Smart Car (Bluetooth Version) has a Bluetooth module added to the standard configuration. Users can use the APP we provide to perform Bluetooth remote control. The Bluetooth remote control includes controlling the car's front, rear, left and right travel, adjusting the speed, and rotating. Smooth operation, good experience, suitable for users who need to play with remote control cars. The current APP program only supports Android phones.

3.Bluetooth remote control instructions




- ① First, connect the car according to the Bluetooth experiment wiring diagram P15.
- ② Turn on the smart car to ensure the power of the Bluetooth module is normal (the light of the Bluetooth module is blinking)
- ③ Open the Bluetooth of the mobile phone and find the Bluetooth module device in the Bluetooth settings. Click to enter the password: 1234.

④ Open Bluetooth App, remote control interface is as follows. If the Bluetooth function is not enabled on the phone, the display button next to the "Bluetooth switch" is dark. Click to open, as shown in the photo



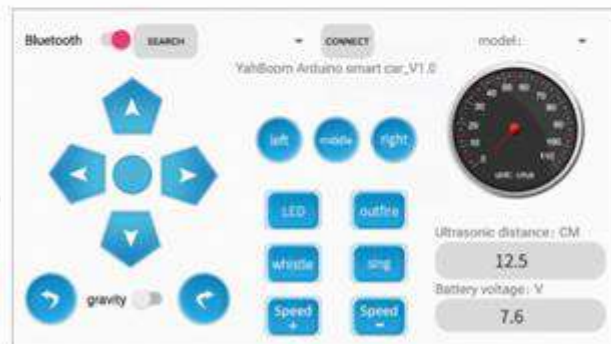
(Bluetooth can also be set in the mobile system)

⑤ Click "Search"

⑥ Click the drop-down button  to select the physical address of the connected Bluetooth module

⑦ Click "connect" . You will be prompted after successful connection. If the connection fails, you return to the first

step to check and try to reconnect.



ReadMe

!!!Note:If you want to upload the code, you need to remove the Bluetooth module.