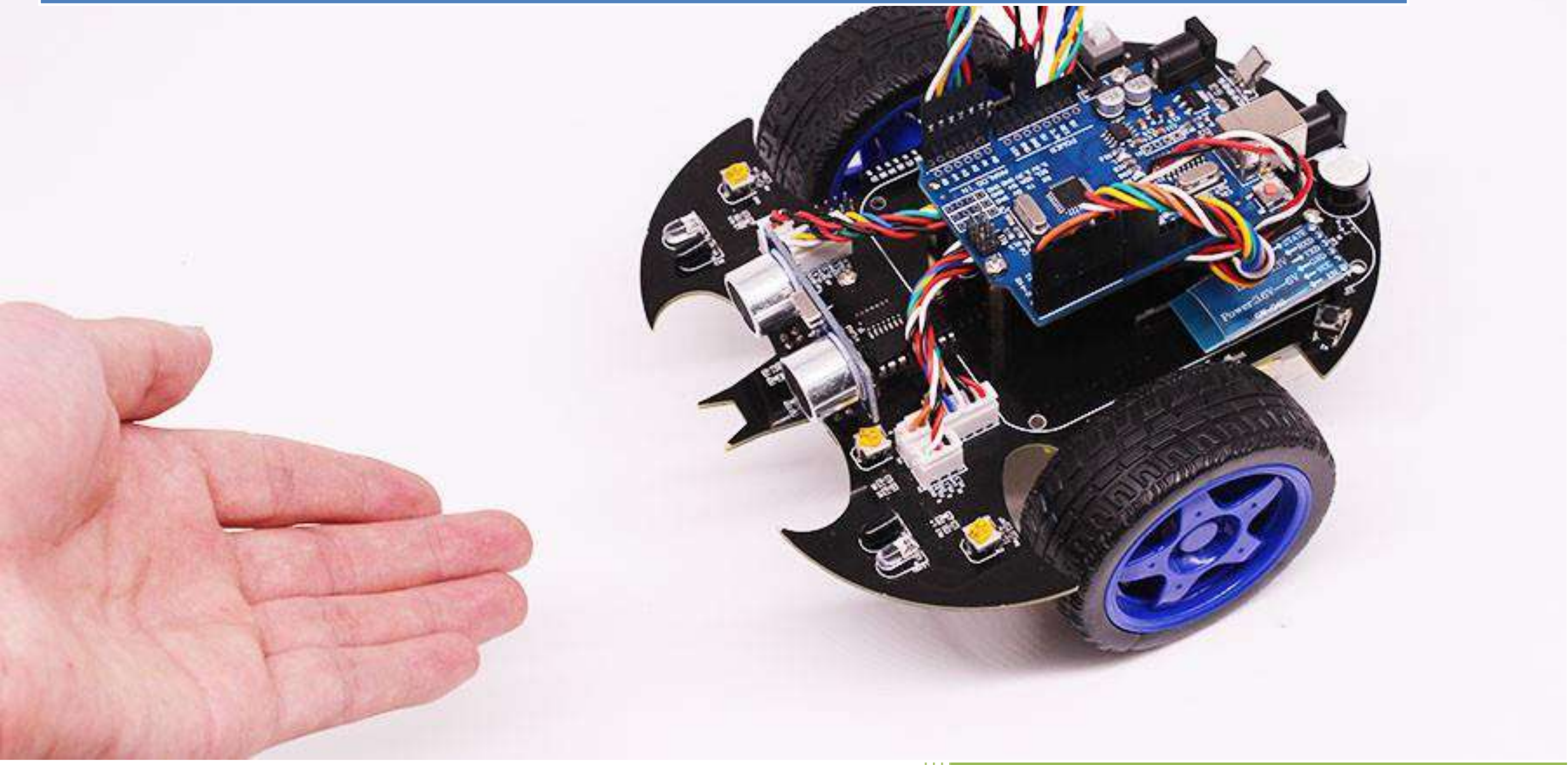
2019

Yahboom Arduino Batmobile

Arduino IDE Programming Tutorials



C programming

1- Go ahead

Advance

The purpose of the experiment:

The purpose of the experiment in this lesson is to open the power switch of the BatCar after the program is uploaded, and the BatCar starts to move forward after 2 seconds of stationary.

List of components required for the experiment:

BatCar*1

USB data cable*1

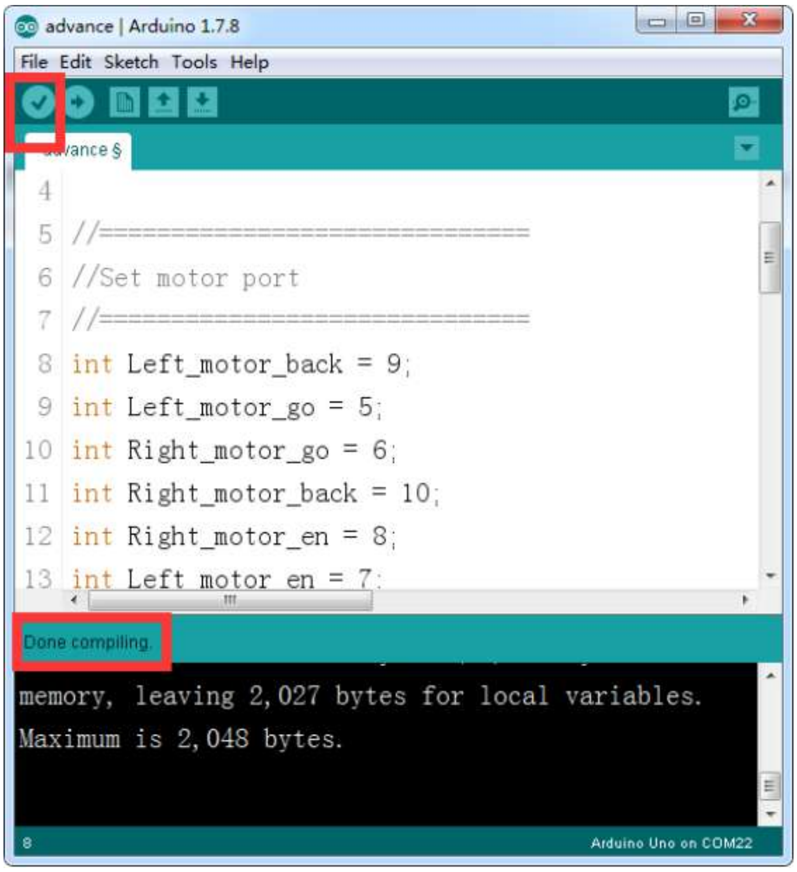
Experimental code analysis:

```
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
void setup()
{
  //Initialize motor drive for output mode
  pinMode(Left_motor_go,OUTPUT);
  pinMode(Left_motor_back,OUTPUT);
  pinMode(Right_motor_go,OUTPUT);
  pinMode(Right_motor_back,OUTPUT);
}
void run(int time) // go ahead
{
  digitalWrite(Left_motor_en,HIGH);  // set left motor enable
  digitalWrite(Right_motor_en,HIGH);  // set right motor enable
  digitalWrite(Right_motor_go,HIGH);  // right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,200);//PWM--Pulse Width Modulation(0~255). It can be
adjusted to control speed.
  digitalWrite(Left_motor_go,HIGH);  // set left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200);//PWM--Pulse Width Modulation(0~255).It can be
adjusted to control speed.
  delay(time * 100);   //Running time can be adjusted
}
```

```
void loop()
{
   delay(2000); //delay 2s
   run(10);   //Run
}
```

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.

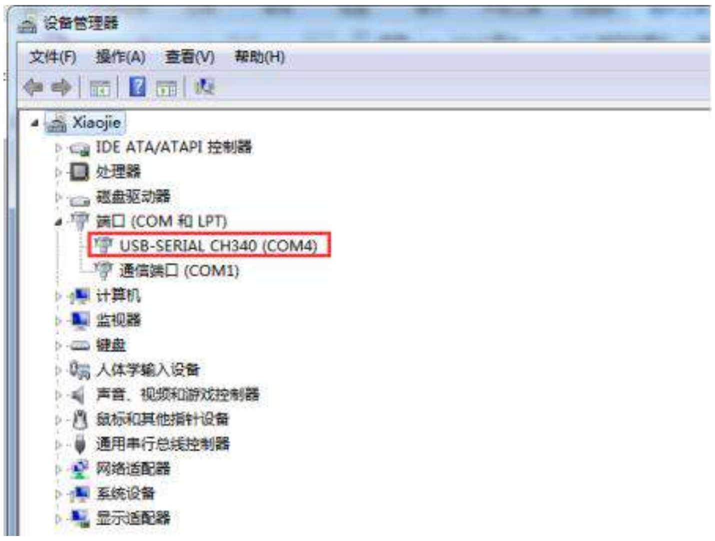


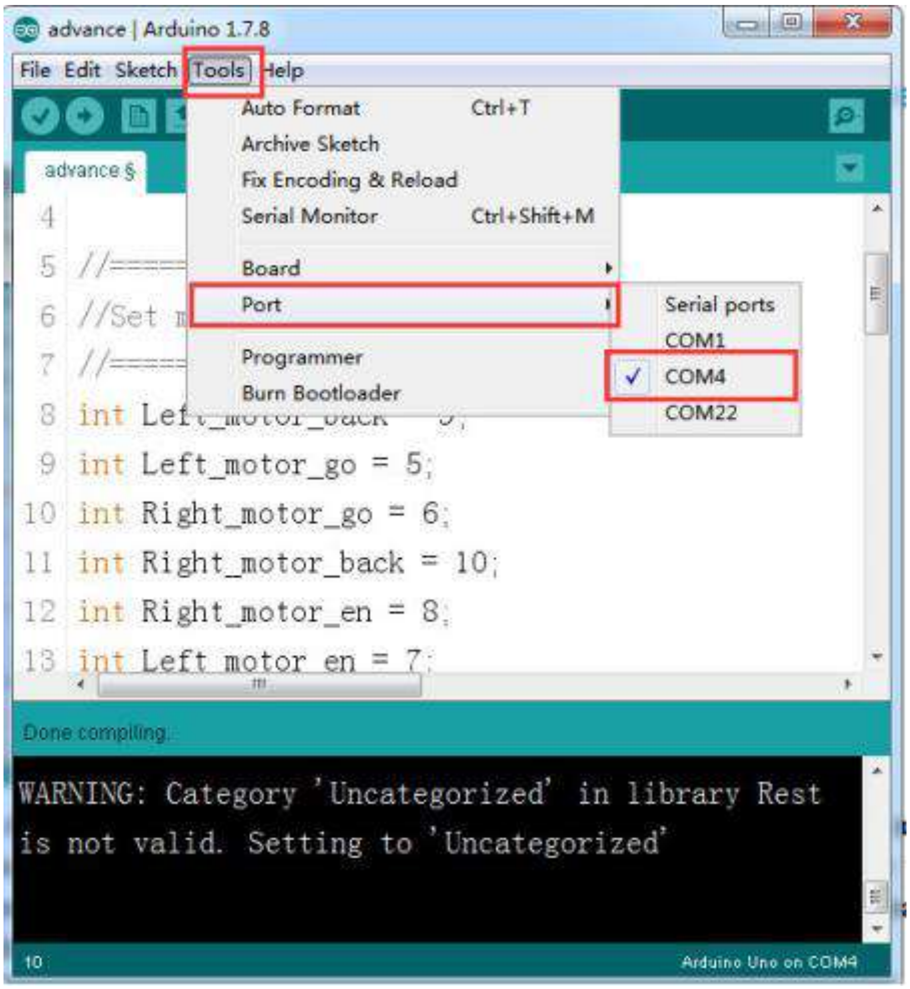
```
//==============================
//Set motor port
//==============================
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
```

Done compiling.

```
memory, leaving 2,027 bytes for local variables.
Maximum is 2,048 bytes.
```

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.

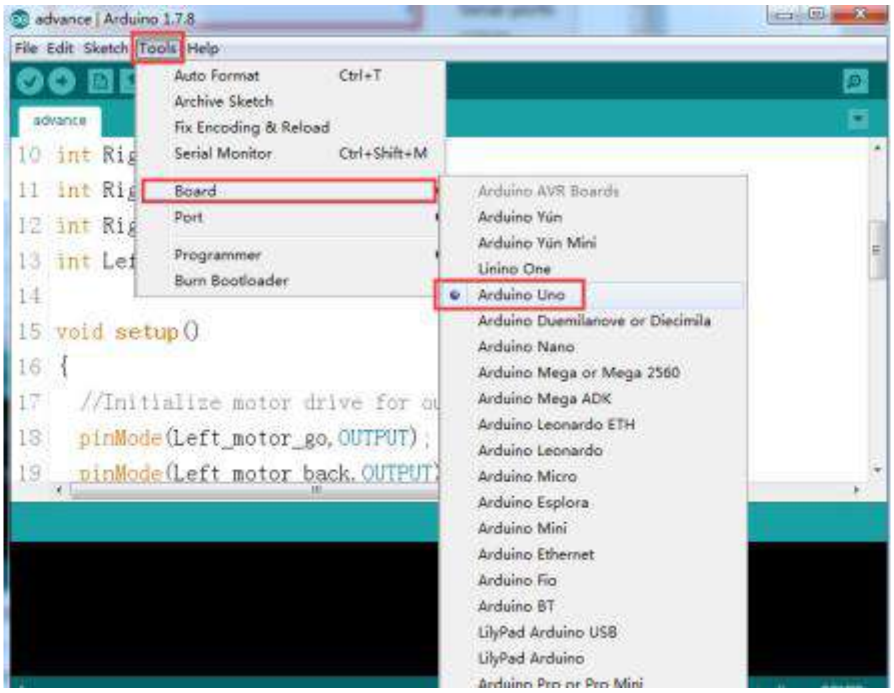




```
//====
//Set m
//====
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
```

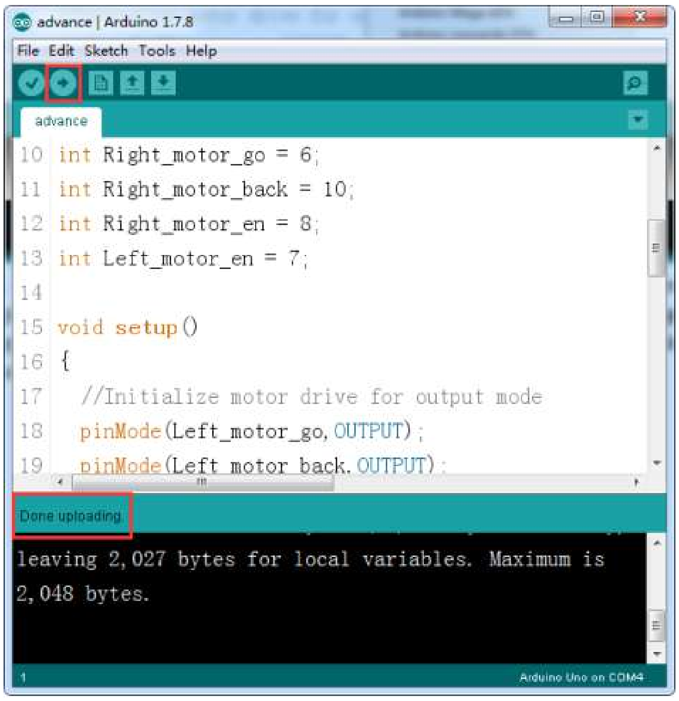
```
WARNING: Category 'Uncategorized' in library Rest
is not valid. Setting to 'Uncategorized'
```

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



```
int Rig
int Rig
int Rig
int Lef

void setup()
{
  //Initialize motor drive for ou
  pinMode(Left_motor_go,OUTPUT);
  pinMode(Left_motor_back,OUTPUT)
```

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.



```
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;

void setup()
{
  //Initialize motor drive for output mode
  pinMode(Left_motor_go,OUTPUT);
  pinMode(Left_motor_back,OUTPUT);
```

```
leaving 2,027 bytes for local variables. Maximum is
2,048 bytes.
```

4. After the program is successfully uploaded, as shown in Figure 1, press the power switch, you can see that the BatCar starts moving forward after 2 seconds .

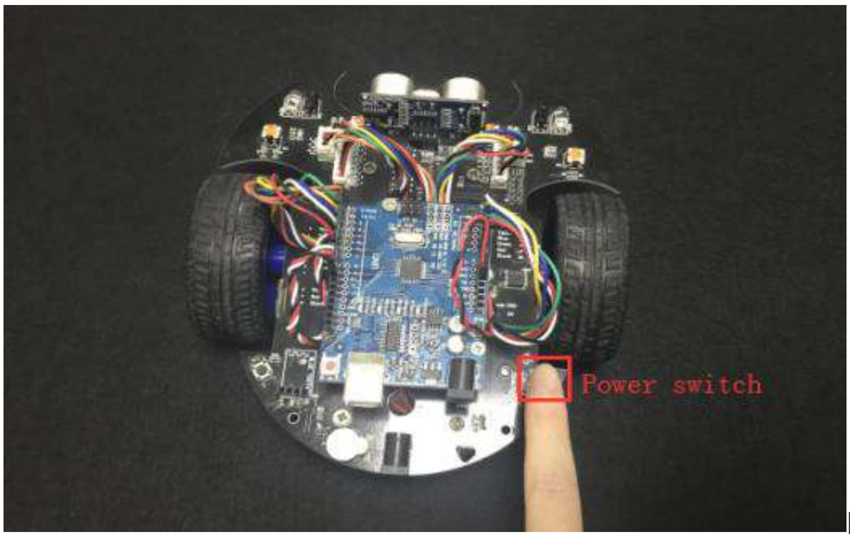


Figure 1

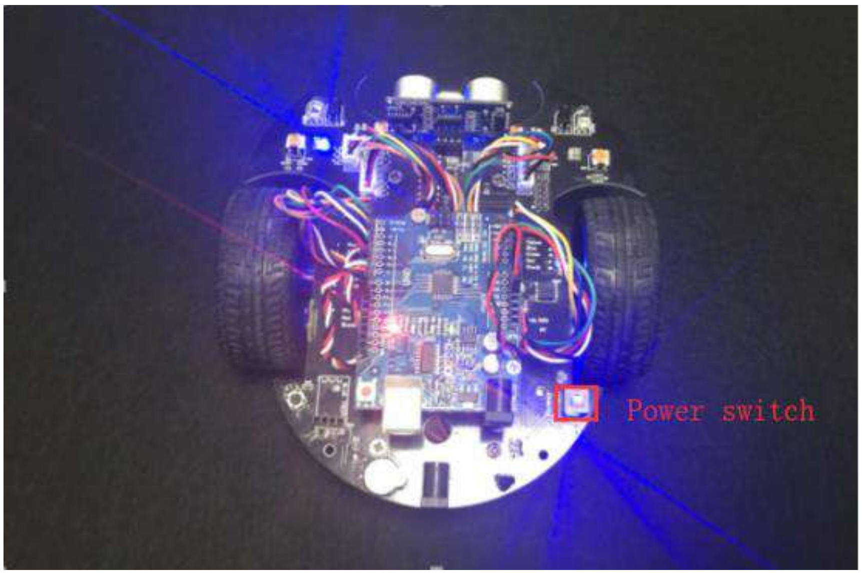


Figure 2

2- Button start

The purpose of the experiment:

After uploading the Button start program, turn on the power switch and press the start button K1. After a short whistle, the BatCar starts to move forward, backward, left turn, right turn, rotate left, rotate right.

List of components required for the experiment:

BatCar *1

USB cable *1

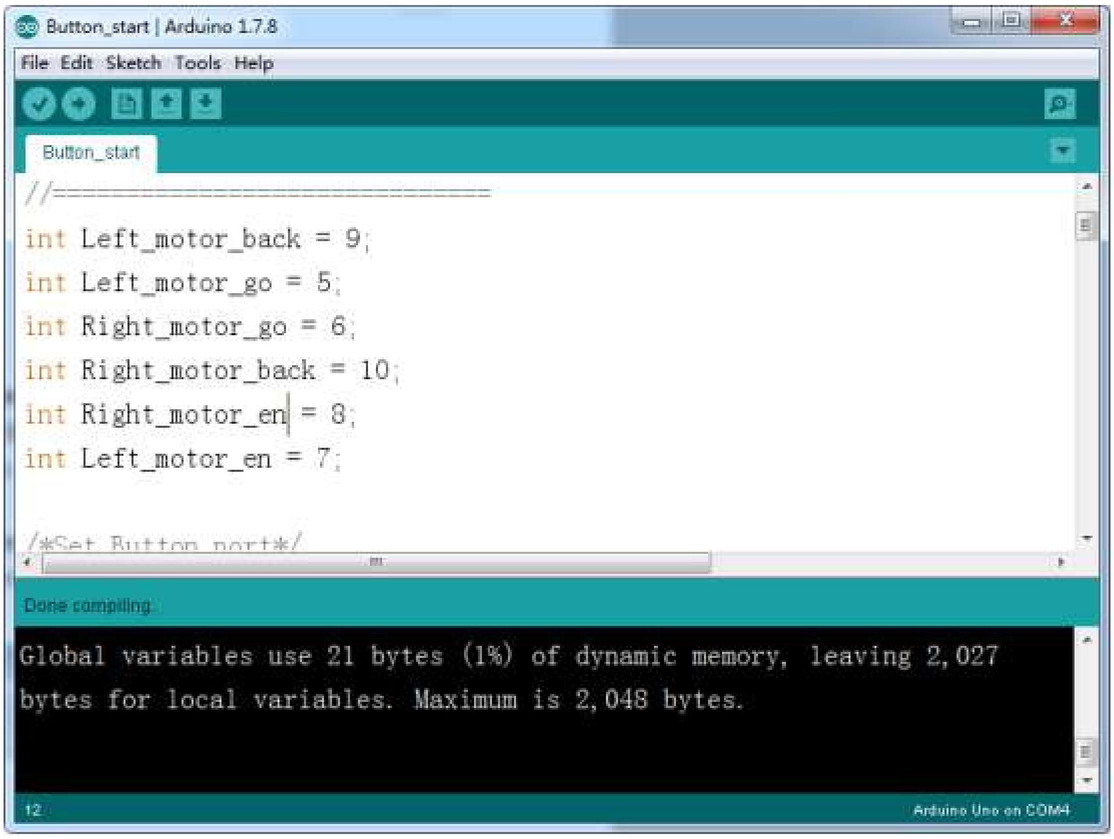
Experimental code analysis:

```
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
/*Set Button port*/
int key=4;
/*Set BUZZER port*/
int beep=3;
void setup()
{
  //Initialize motor drive for output mode
  pinMode(Left_motor_go,OUTPUT);
  pinMode(Left_motor_back,OUTPUT);
  pinMode(Right_motor_go,OUTPUT);
  pinMode(Right_motor_back,OUTPUT);
  pinMode(key,INPUT);// Set button as input
  pinMode(beep,OUTPUT); // Set buzzer as output
  digitalWrite(key,HIGH);//Initialize button
  digitalWrite(beep,HIGH);// set buzzer mute
}
void run(int time)     // run
{
  digitalWrite(Right_motor_go,HIGH); // right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,200);//PWM--Pulse Width Modulation(0~255). It can be
adjusted to control speed.
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH); // set left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200);//PWM--Pulse Width Modulation(0~255). It can be
adjusted to control speed.
  analogWrite(Left_motor_back,0);
  delay(time * 100);   //Running time can be adjusted
}
void brake(int time)          //stop
{
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,LOW);
  delay(time * 100);
}
void left(int time) //turn left
{
  digitalWrite(Right_motor_go,HIGH);// right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,200);// PWM--Pulse Width Modulation(0~255) control
speed
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,LOW);  // left motor stop
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,0);
  delay(time * 100);
}
void spin_left(int time)   //Left rotation
{
  digitalWrite(Right_motor_go,HIGH);// right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,200);// PWM--Pulse Width Modulation(0~255) control
speed
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,LOW);   // left motor back off
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,200);// PWM--Pulse Width Modulation(0~255) control
speed
  delay(time * 100);
}
void right(int time)      //turn right
{
  digitalWrite(Right_motor_go,LOW);   // right motor stop
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH);// left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200);// PWM--Pulse Width Modulation(0~255) control
speed
  analogWrite(Left_motor_back,0);
  delay(time * 100);
}
void spin_right(int time)   //Right rotation
{
  digitalWrite(Right_motor_go,LOW);  // right motor back off
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,200);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,HIGH);// left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200);// PWM--Pulse Width Modulation(0~255) control
speed
  analogWrite(Left_motor_back,0);
  delay(time * 100);
}
void back(int time)   //back off
{
  digitalWrite(Right_motor_go,LOW); //right motor back off
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,150);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,LOW);  //left motor back off
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,150);// PWM--Pulse Width Modulation(0~255) control
speed
  delay(time * 100);
}
void keysacn()
{
  int val;
  val=digitalRead(key);// Reads the button ,the level value assigns to val
  while(digitalRead(key))// When the button is not pressed
  {
    val=digitalRead(key);
  }
  while(!digitalRead(key))// When the button is pressed
  {
   delay(10); //delay 10ms
    val=digitalRead(key);// Reads the button ,the level value assigns to val
    if(val==LOW)  //Double check the button is pressed
    {
      digitalWrite(beep,LOW);//The buzzer sounds
      delay(100);//delay 100ms
      while(!digitalRead(key)) //Determine if the button is released or not
        digitalWrite(beep,HIGH);//mute
    }
    else
      digitalWrite(beep,HIGH);//mute
  }
```

```
}
void loop()
{
  delay(2000); //delay 2s
  keysacn();
  back(10); //back off for 1s
  brake(5); //stop for 0.5s
  run(10);//go ahead for 1s
  brake(5); //stop for 0.5s
  left(10);//turn left for 1s
  right(10);//turn right for 1s
  spin_left(20);//Left rotation for 2s
  spin_right(20);//Right rotation for 2s
 brake(5);  //stop for 0.5s
}
```

Experimental steps:

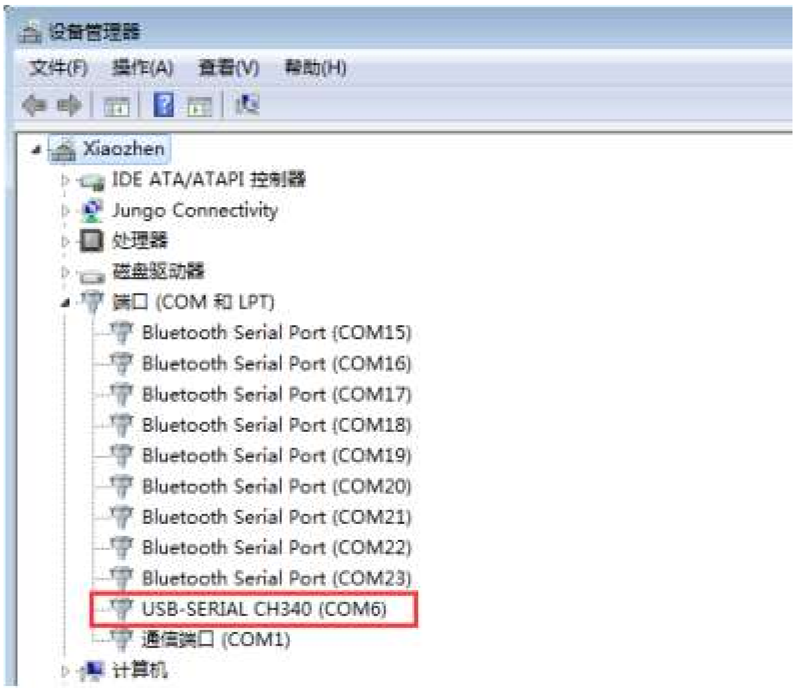
1. We need to open the code of this experiment: **Button_start.ino**, click"√" under the menu bar to compile the code, and wait for the word "**Done compiling** " in the lower right corner, as shown in the figure below.



```
//================================
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;

/*Set Button port*/
```

```
Global variables use 21 bytes (1%) of dynamic memory, leaving 2,027
bytes for local variables. Maximum is 2,048 bytes.
```

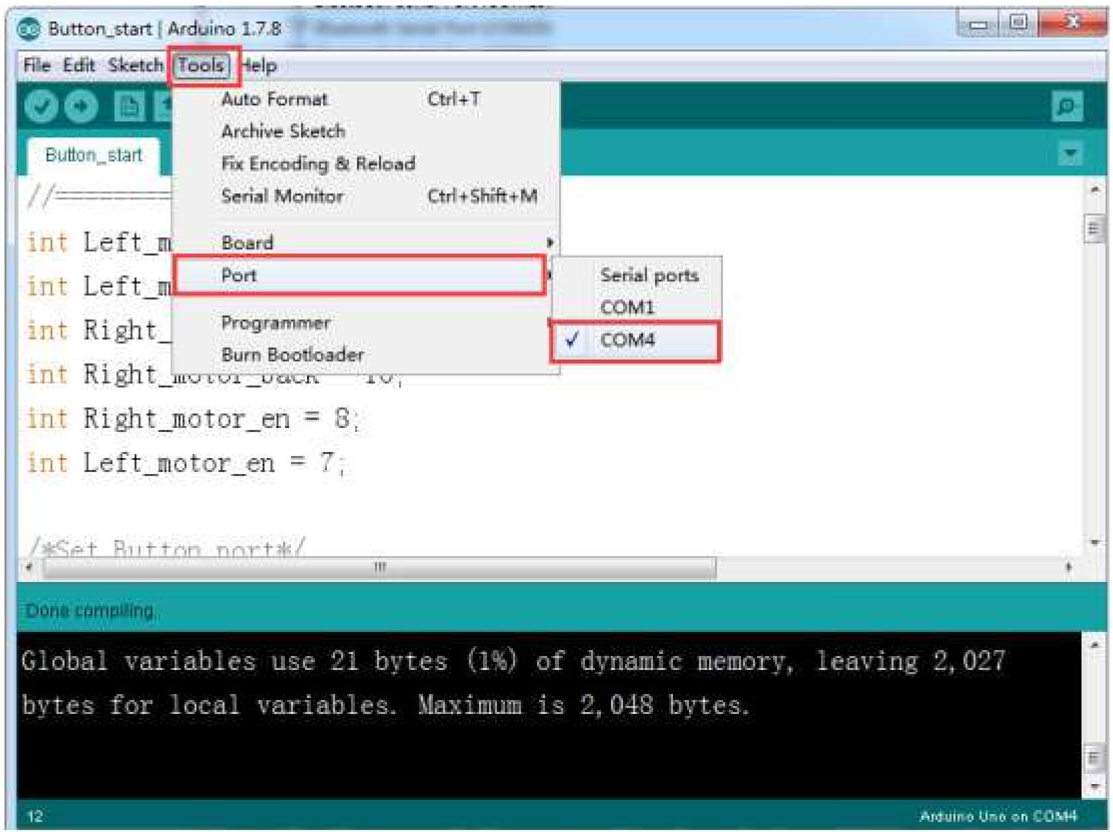
2. In the menu bar of Arduino IDE, we need to select 【Tools】---【Port】--- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below.



设备管理器

文件(F) 操作(A) 查看(V) 帮助(H)

```
Xiaozhen
  IDE ATA/ATAPI 控制器
  Jungo Connectivity
  处理器
  磁盘驱动器
  端口 (COM 和 LPT)
    Bluetooth Serial Port (COM15)
    Bluetooth Serial Port (COM16)
    Bluetooth Serial Port (COM17)
    Bluetooth Serial Port (COM18)
    Bluetooth Serial Port (COM19)
    Bluetooth Serial Port (COM20)
    Bluetooth Serial Port (COM21)
    Bluetooth Serial Port (COM22)
    Bluetooth Serial Port (COM23)
    USB-SERIAL CH340 (COM6)
    通信端口 (COM1)
  计算机
```



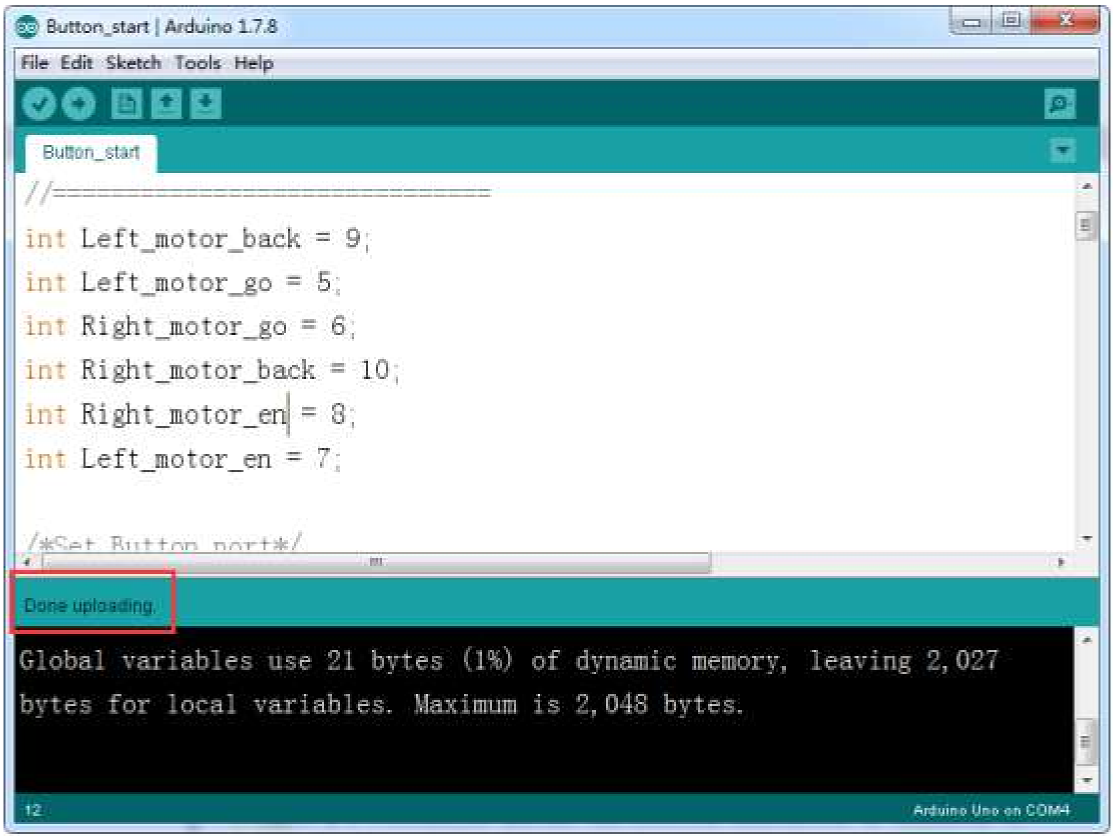
```
//========
int Left_m
int Left_m
int Right_
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;

/*Set Button port*/
```

Done compiling.

```
Global variables use 21 bytes (1%) of dynamic memory, leaving 2,027
bytes for local variables. Maximum is 2,048 bytes.
```

12 Arduino Uno on COM4

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.

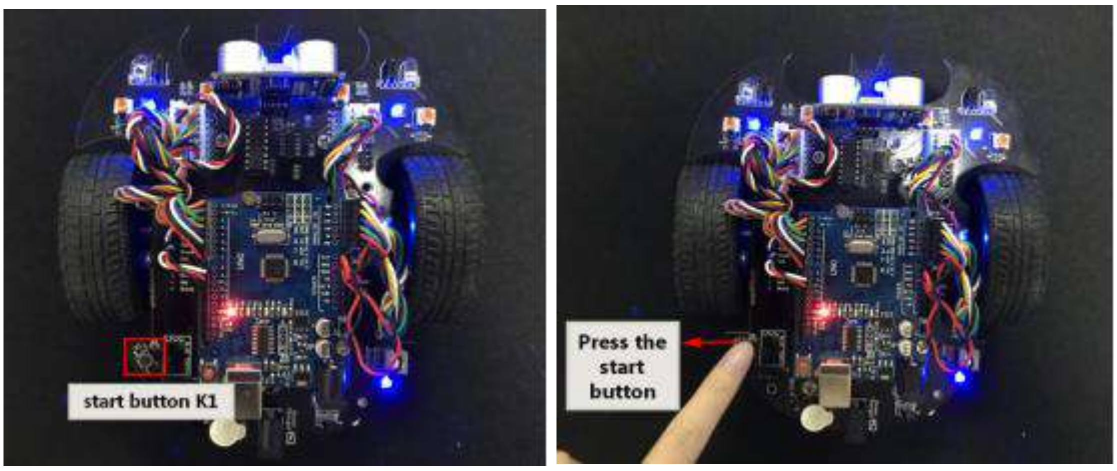


```
//==============================
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;

/*Set Button port*/
```

```
Done uploading.
```

```
Global variables use 21 bytes (1%) of dynamic memory, leaving 2,027
bytes for local variables. Maximum is 2,048 bytes.
```

4. Unplug the USB cable, put the BatCar in an open place, turn on the power switch, and the BatCar is still at rest until the start button K1 is pressed. After a short whistle, the BatCar starts to move forward and backward. Turn left, turn right, etc.



3- Line Walking

The purpose of the experiment:

After uploading the program, unplug the USB data cable, put the BatCar on the patrol tarck.Adjust the potentionmeterSW3 and SW4 so that the photoelectric sensor can recognize the black line.Press the start button K1, the BatCar starts running along the black line.

List of components required for the experiment:

BatCar*1

USB data cable*1

Patrol tarck*1

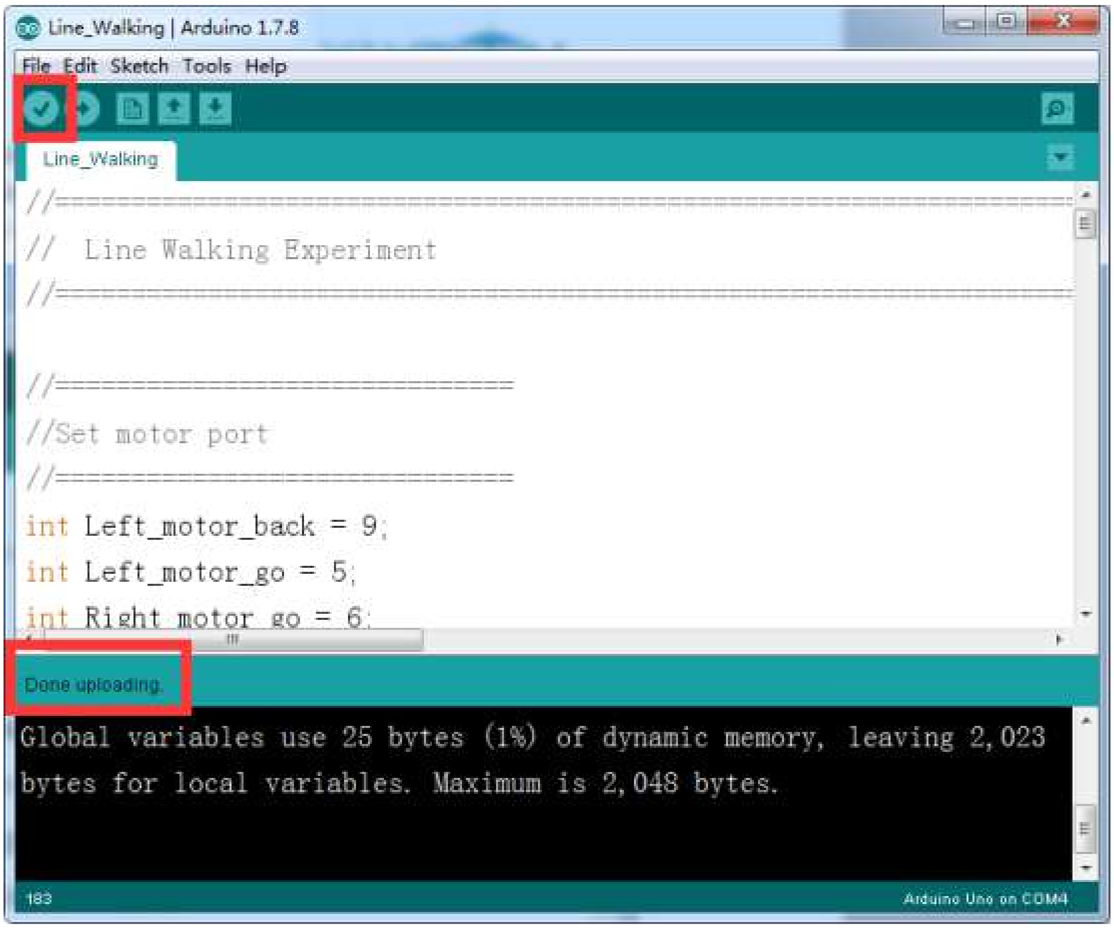
Experimental code analysis:

```
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
/*Set Button port*/
int key=4;
/*Set BUZZER port*/
int beep=3;
/*Line Walking*/
const int SensorRight = A3;    // Set Right Line Walking Infrared sensor port
const int SensorLeft = A2;     // Set Left Line Walking Infrared sensor port
int SL;    // State of Left Line Walking Infrared sensor
int SR;    // State of Right Line Walking Infrared sensor
void setup()
{
  //Initialize motor drive for output mode
  pinMode(Left_motor_go,OUTPUT);
  pinMode(Left_motor_back,OUTPUT);
  pinMode(Right_motor_go,OUTPUT);
  pinMode(Right_motor_back,OUTPUT);
  pinMode(key,INPUT);// Set button as input
  pinMode(beep,OUTPUT);// Set buzzer as output
  pinMode(SensorRight, INPUT); // Set Right Line Walking Infrared sensor as input
  pinMode(SensorLeft, INPUT); // Set left Line Walking Infrared sensor as input
  digitalWrite(key,HIGH);//Initialize button
  digitalWrite(beep,HIGH);// set buzzer mute
}
 //=====================Motor=======================
void run()
{
  digitalWrite(Right_motor_go,HIGH);// right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100);//PWM--Pulse Width Modulation(0~255). It can be
adjusted to control speed.
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH);// set left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100);//PWM--Pulse Width Modulation(0~255). It can be
adjusted to control speed.
  analogWrite(Left_motor_back,0);
}
void brake() //stop
{
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,LOW);
}
void left()//turn left
{
 digitalWrite(Right_motor_go,HIGH); // right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100);
  analogWrite(Right_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,LOW);    // left motor stop
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
}
void spin_left(int time)        //Left rotation
{
 digitalWrite(Right_motor_go,HIGH); // right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100); // PWM--Pulse Width Modulation(0~255) control
speed
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,LOW);    // left motor back off
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,100); // PWM--Pulse Width Modulation(0~255) control
speed
  delay(time * 100);
}
void right() //turn right
```

```
{
  digitalWrite(Right_motor_go,LOW);    // right motor stop
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH);// left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100);
  analogWrite(Left_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
}
void spin_right(int time)        //Right rotation
{
  digitalWrite(Right_motor_go,LOW);   // right motor back off
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,200);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,HIGH);// left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200);
  analogWrite(Left_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
  delay(time * 100);
}
void back(int time)  //back off
{
  digitalWrite(Right_motor_go,LOW);  //right motor back off
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,150);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,LOW);  //left motor back off
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,150);// PWM--Pulse Width Modulation(0~255) control
speed
  delay(time * 100);
}
//==========================================================
void keysacn()
{
  int val;
  val=digitalRead(key);// Reads the button ,the level value assigns to val
  while(digitalRead(key))// When the button is not pressed
  {
    val=digitalRead(key);
  }
  while(!digitalRead(key))// When the button is pressed
  {
 delay(10); //delay 10ms
    val=digitalRead(key);// Reads the button ,the level value assigns to val
    if(val==LOW)  //Double check the button is pressed
    {

      digitalWrite(beep,LOW);//The buzzer sounds
      delay(50);//delay 50ms
   while(!digitalRead(key)) //Determine if the button is released or not
        digitalWrite(beep,HIGH);//mute
    }
    else
      digitalWrite(beep,HIGH);//mute
  }
}
/*main loop*/
void loop()
{
  keysacn();//Press the button to start
  while(1)
  {
  /***************************************************************************************
  Infrared signal back means white undersurface ,returns low level and led lights up.
  Infrared signal gone means black undersurface ,returns high level and led lights off.
  ***************************************************************************************/
  SR = digitalRead(SensorRight);//Right Line Walking Infrared sensor against white undersurface,then LED[L2] light illuminates and while against black undersurface,LED[L2] goes off
  SL = digitalRead(SensorLeft);//Left Line Walking Infrared sensor against white undersurface,then LED[L3] light illuminates and while against black undersurface,LED[L3] goes off
  if (SL ==LOW&&SR== LOW)// Black lines were not detected at the same time
    run();   // go ahead
  else if (SL == LOW & SR == HIGH)// Left sensor against white undersurface and right against black undersurface , the car left off track and need to adjust to the right.
    right();
  else if (SR == LOW & SL ==  HIGH) // Rihgt sensor against white undersurface and left against black undersurface , the car right off track and need to adjust to the left.
    left();
  else // Black lines were detected at the same time , the car stop.
    brake();
  }
}
```

Experimental steps:

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

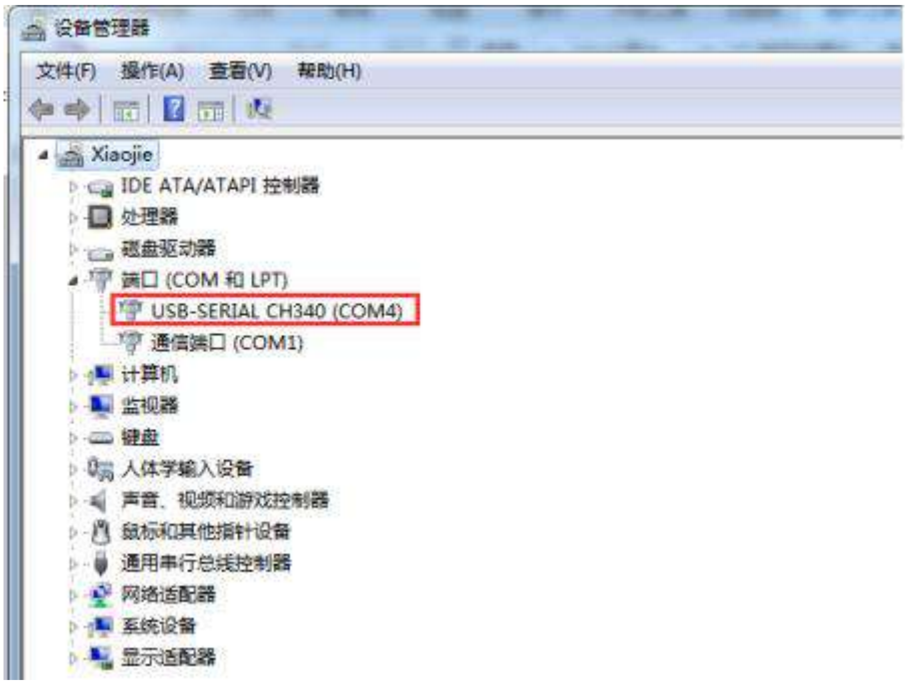


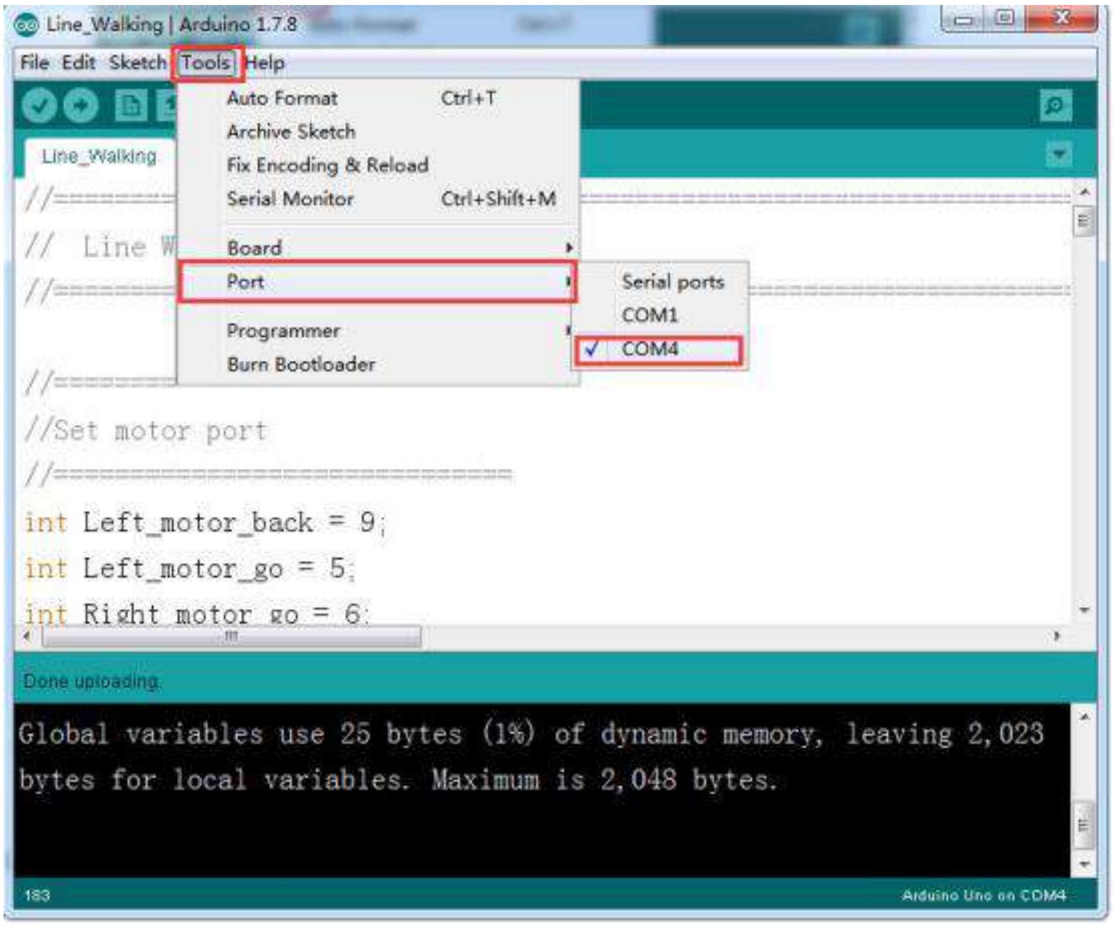
```
//==========================================================================
//  Line Walking Experiment
//==========================================================================


//==============================
//Set motor port
//==============================
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right motor go = 6;
```

Done uploading.

```
Global variables use 25 bytes (1%) of dynamic memory, leaving 2,023
bytes for local variables. Maximum is 2,048 bytes.
```

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.





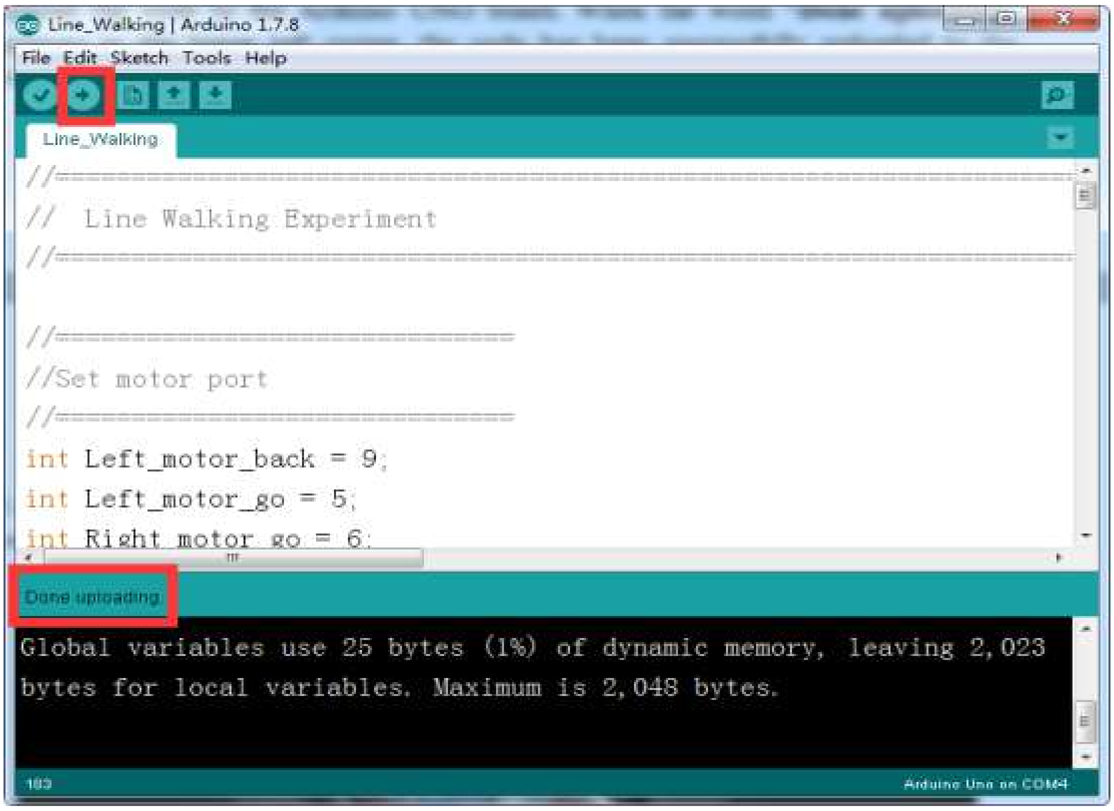
```
//=====
// Line W
//=====

//=====
//Set motor port
//==============================
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
```

Done uploading.

```
Global variables use 25 bytes (1%) of dynamic memory, leaving 2,023
bytes for local variables. Maximum is 2,048 bytes.
```

183 Arduino Uno on COM4

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.



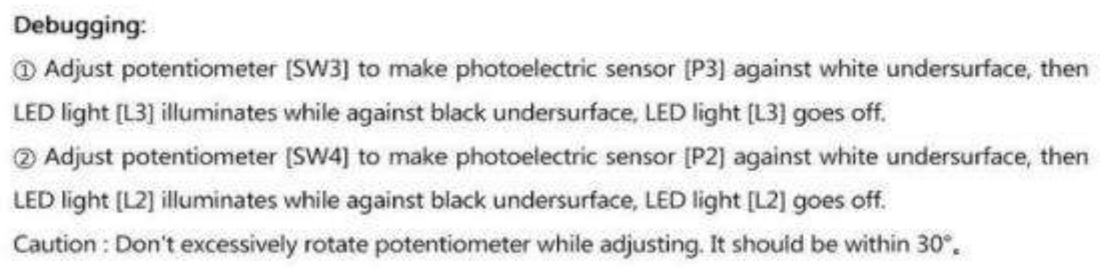
```
//=====================================================================
//  Line Walking Experiment
//=====================================================================

//==============================
//Set motor port
//==============================
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
```

Done uploading.

```
Global variables use 25 bytes (1%) of dynamic memory, leaving 2,023
bytes for local variables. Maximum is 2,048 bytes.
```

183 Arduino Uno on COM4

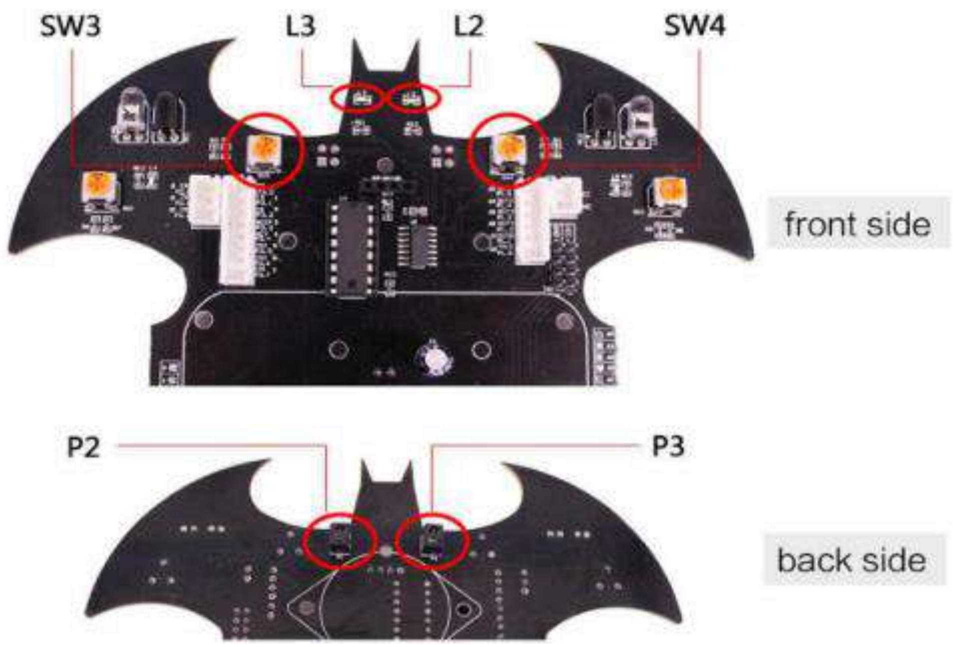
4. After the program is successfully downloaded, adjust potentiometers SW3 and SW4 to the correct angle.


Debugging:

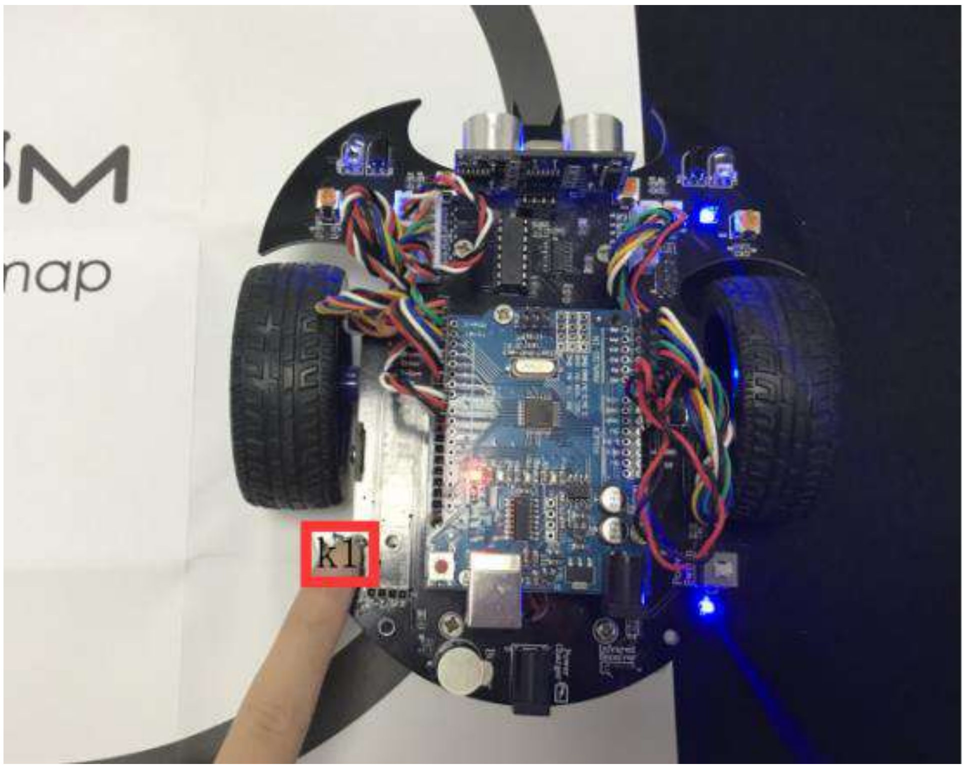
① Adjust potentiometer [SW3] to make photoelectric sensor [P3] against white undersurface, then LED light [L3] illuminates while against black undersurface, LED light [L3] goes off.

② Adjust potentiometer [SW4] to make photoelectric sensor [P2] against white undersurface, then LED light [L2] illuminates while against black undersurface, LED light [L2] goes off.

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



5.Put the BatCar on the patrol tarck and press the start button K1. With a short whistle, the BatCar starts running along the black line.



4- Infrared obstacle avoidance

The purpose of the experiment:

Debug potentiometers SW1 and SW2 according to the debug diagram at the end of the article.Turn on the power switch of the BatCar, press the start button K1, and when you hear the short whistle, the BatCar avoids the obstacle.

List of components required for the experiment:

BatCar*1

USB data cable*1

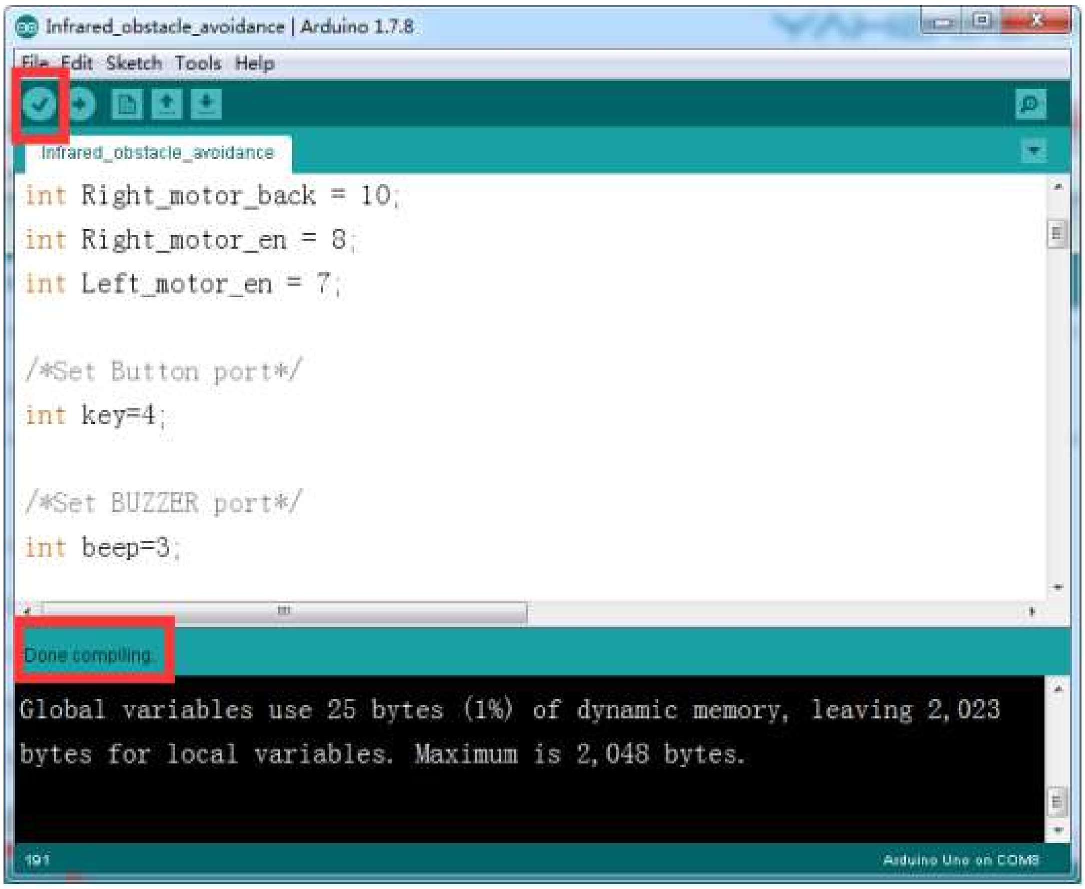
Experimental code analysis:

```
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
/*Set Button port*/
int key=4;
/*Set BUZZER port*/
int beep=3;
/*Line Walking*/
const int SensorRight = A3;     // Set Right Line Walking Infrared sensor port
const int SensorLeft = A2;      // Set Left Line Walking Infrared sensor port
int SL;     // State of Left Line Walking Infrared sensor
int SR;     // State of Right Line Walking Infrared sensor
/*Infrared obstacle avoidance*/
const int SensorRight_2 = A4;     // Right  Infrared sensor
const int SensorLeft_2 = A5;     // Left  Infrared sensor
int SL_2;    // State of Left  Infrared sensor
int SR_2;    // State of Right  Infrared sensor
void setup()
{
  //Initialize motor drive for output mode
  pinMode(Left_motor_go,OUTPUT);
  pinMode(Left_motor_back,OUTPUT);
  pinMode(Right_motor_go,OUTPUT);
  pinMode(Right_motor_back,OUTPUT);
  pinMode(key,INPUT);// Set button as input
  pinMode(beep,OUTPUT);// Set buzzer as output
  pinMode(SensorRight, INPUT); // Set Right Line Walking Infrared sensor as input
  pinMode(SensorLeft, INPUT); // Set left Line Walking Infrared sensor as input
  pinMode(SensorRight_2, INPUT); //Set Right  Infrared sensor as input
  pinMode(SensorLeft_2, INPUT); //Set left Infrared sensor as input
  digitalWrite(key,HIGH);//Initialize button
  digitalWrite(beep,HIGH);// set buzzer mute
}
//=====================Motor=======================
void run()
{
  digitalWrite(Right_motor_go,HIGH);// right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100);//PWM--Pulse Width Modulation(0~255). It can be
adjusted to control speed.
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH);// set left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100);//PWM--Pulse Width Modulation(0~255). It can be
adjusted to control speed.
  analogWrite(Left_motor_back,0);
}
void brake() //stop
{
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,LOW);
}
void left()//turn left
{
 digitalWrite(Right_motor_go,HIGH); // right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100);
  analogWrite(Right_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,LOW);    // left motor stop
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
}
void spin_left(int time)         //Left rotation
{
 digitalWrite(Right_motor_go,HIGH); // right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100); // PWM--Pulse Width Modulation(0~255) control
speed
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,LOW);    // left motor back off
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,100); // PWM--Pulse Width Modulation(0~255) control
speed
```

```
  delay(time * 100);
}
void right() //turn right
{
  digitalWrite(Right_motor_go,LOW);    // right motor stop
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH);// left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100);
  analogWrite(Left_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
}
void spin_right(int time)        //Right rotation
{
  digitalWrite(Right_motor_go,LOW);   // right motor back off
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,200);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,HIGH);// left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200);
  analogWrite(Left_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
  delay(time * 100);
}
void back(int time)  //back off
{
  digitalWrite(Right_motor_go,LOW);  //right motor back off
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,150);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,LOW);  //left motor back off
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,150);// PWM--Pulse Width Modulation(0~255) control
speed
  delay(time * 100);
}
//==========================================================
void keysacn()
{
  int val;
  val=digitalRead(key);// Reads the button ,the level value assigns to val
  while(digitalRead(key))// When the button is not pressed
  {
    val=digitalRead(key);
  }
  while(!digitalRead(key))// When the button is pressed
  {
    delay(10); //delay 10ms
    val=digitalRead(key);// Reads the button ,the level value assigns to val
    if(val==LOW)  //Double check the button is pressed
    {
        digitalWrite(beep,LOW);//The buzzer sounds
        delay(50);//delay 50ms
        while(!digitalRead(key)) //Determine if the button is released or not
        digitalWrite(beep,HIGH);//mute
    }
    else
      digitalWrite(beep,HIGH);//mute
  }
}
/*main loop*/
void loop()
{
  keysacn();//Press the button to start
  while(1)
  {
  /*******************************************************************************************
  Infrared signal back means there is something obstacled ,returns low level and led
lights up.
  Infrared signal gone means there is nothing obstacled ,returns high level and led lights
off.
  *******************************************************************************************/
    SR_2 = digitalRead(SensorRight_2);//Right infrared sensor detects the obstacle,then
LED[L5] light illuminates and otherwise it goes off.
    SL_2 = digitalRead(SensorLeft_2);//Left infrared sensor detects the obstacle,then
LED[L4] light illuminates and otherwise it goes off.
    if (SL_2 == HIGH&&SR_2==HIGH)//There is nothing obstacled ,goes ahead.
      run();
    else if (SL_2 == HIGH & SR_2 == LOW)// There is something obstacled on the right
then LED[L4] light illuminates,turns left.
        left();
    else if (SR_2 == HIGH & SL_2 == LOW)// There is something obstacled on the left
then LED[L4] light illuminates,turns right.
      right();
    else // //There is something obstacled, back off and adjust direction.
    {
      back(6);//back off for 600ms
      spin_right(3);//Right rotation for 300ms to adjust direction
    }
  }
}
```

Experimental steps:

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

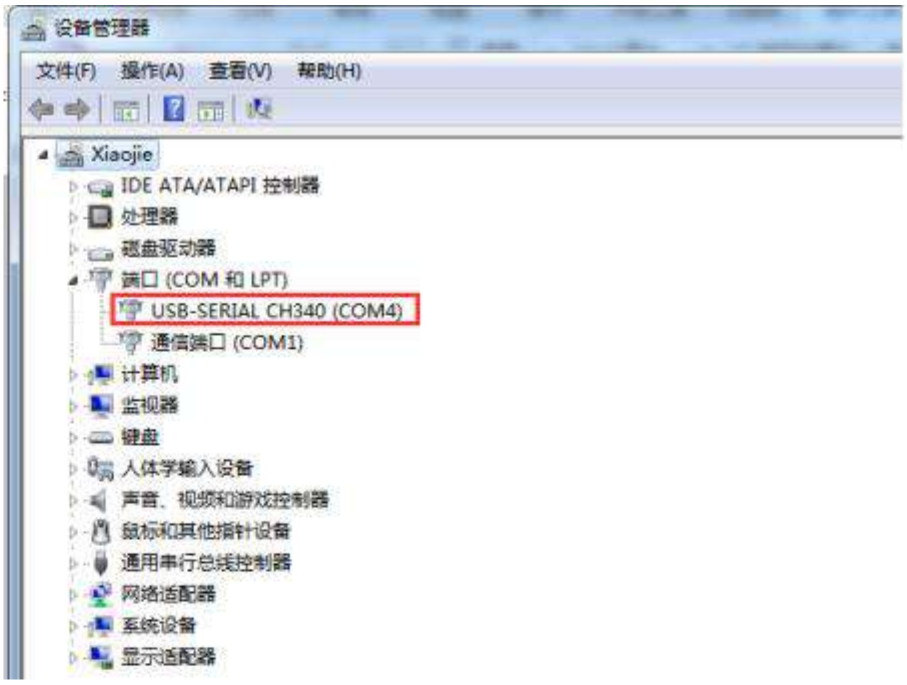

Infrared_obstacle_avoidance | Arduino 1.7.8
File Edit Sketch Tools Help
Infrared_obstacle_avoidance

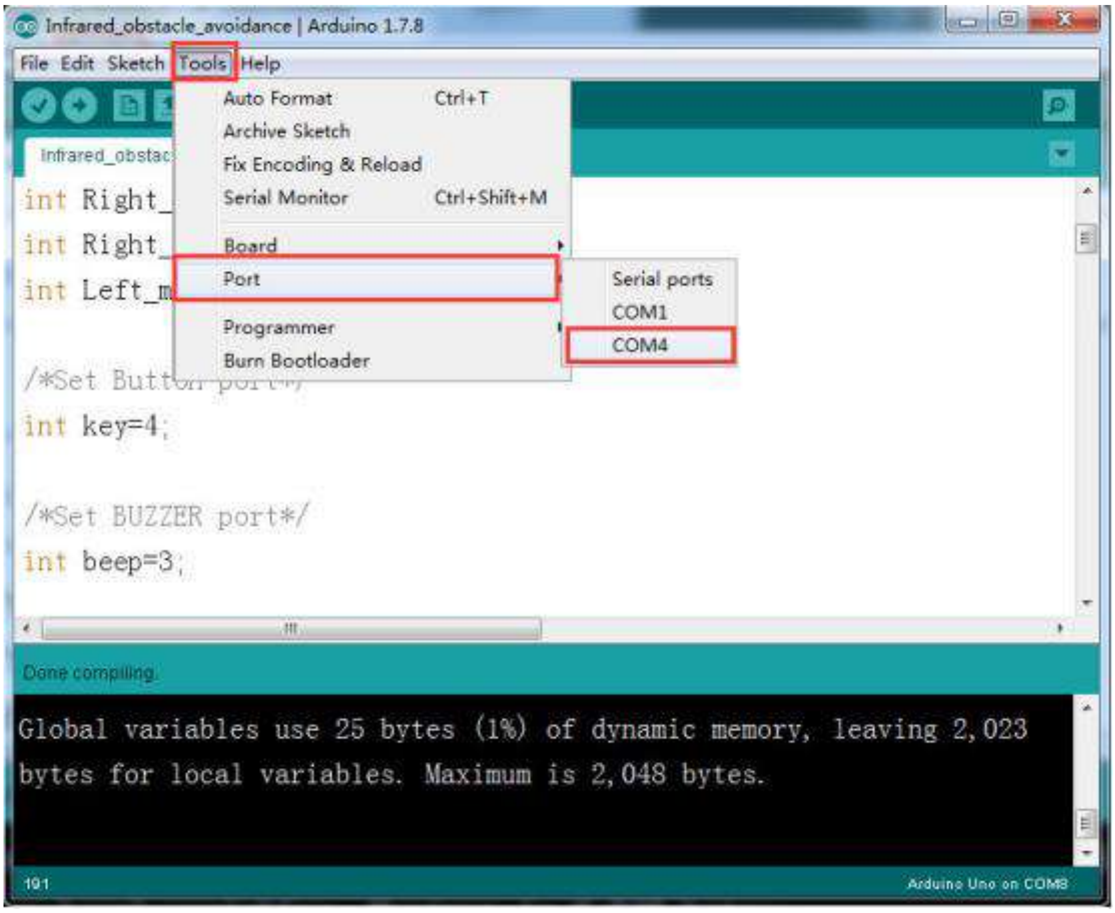
```
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;

/*Set Button port*/
int key=4;

/*Set BUZZER port*/
int beep=3;
```

Done compiling.

```
Global variables use 25 bytes (1%) of dynamic memory, leaving 2,023
bytes for local variables. Maximum is 2,048 bytes.
```

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.

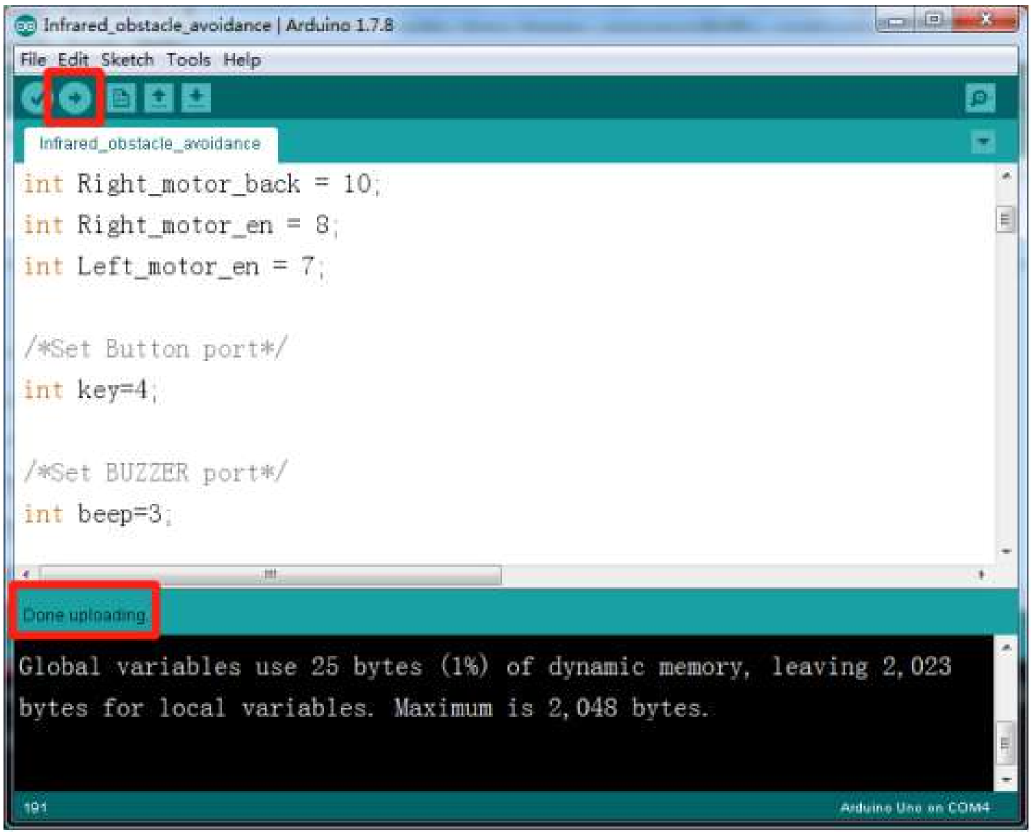




```
int Right_
int Right_
int Left_m

/*Set Butto

int key=4;

/*Set BUZZER port*/
int beep=3;
```

```
Global variables use 25 bytes (1%) of dynamic memory, leaving 2,023
bytes for local variables. Maximum is 2,048 bytes.
```

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

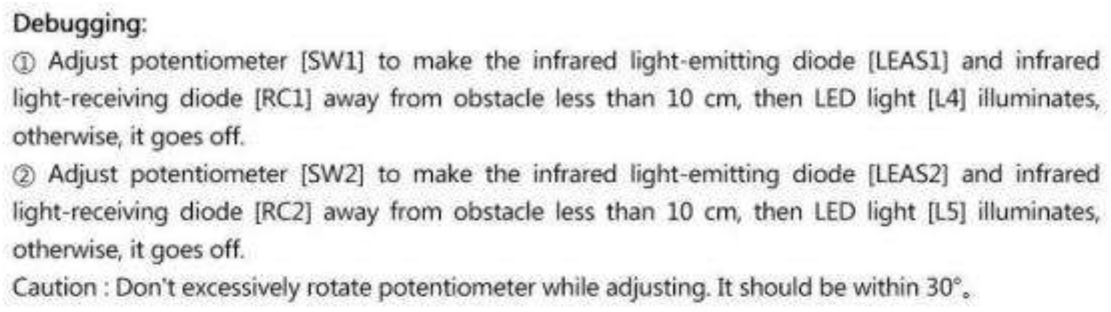


Infrared_obstacle_avoidance | Arduino 1.7.8
File Edit Sketch Tools Help
Infrared_obstacle_avoidance
Done uploading.
191
Arduino Uno on COM4

```
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;

/*Set Button port*/
int key=4;

/*Set BUZZER port*/
int beep=3;
```

```
Global variables use 25 bytes (1%) of dynamic memory, leaving 2,023
bytes for local variables. Maximum is 2,048 bytes.
```

4. After the program is successfully uploaded, debug the BatCar as shown below.

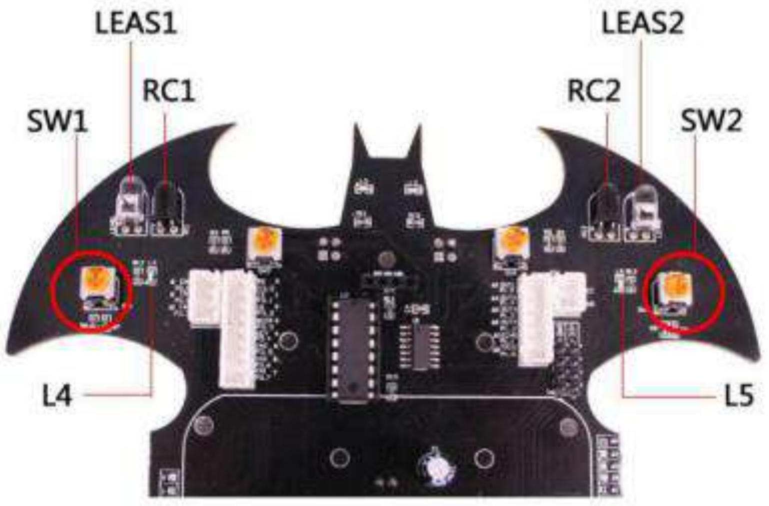


Debugging:

① Adjust potentiometer [SW1] to make the infrared light-emitting diode [LEAS1] and infrared light-receiving diode [RC1] away from obstacle less than 10 cm, 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 10 cm, then LED light [L5] illuminates, otherwise, it goes off.

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



5. Place the BatCar in an open space and place some cartons as obstacles. Turn on the BatCar's power switch, press the start button K1, and after hearing the short whistle, BatCar begins to avoid obstacles.

5- Tracking

The purpose of the experiment:

After the program is uploaded, debug BatCar's potentiometers SW1 and SW2 according to the debugging method at the end of the article, open the BatCar's power switch, press the start button K1, and after hearing the short whistle, BatCar starts to follow the previous object.

List of components required for the experiment:

BatCar*1

USB data cable*1

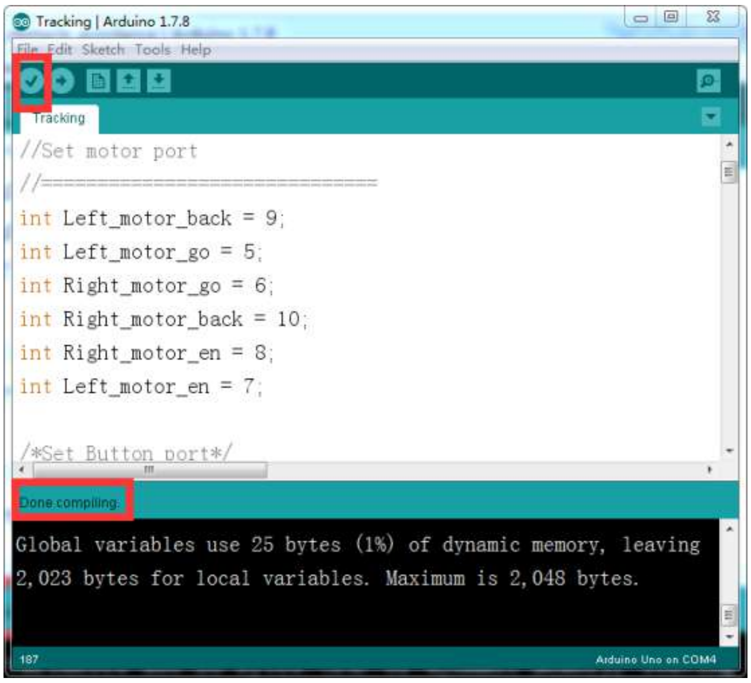
Experimental code analysis:

```
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
/*Set Button port*/
int key=4;
/*Set BUZZER port*/
int beep=3;
/*Line Walking*/
const int SensorRight = A3;    // Set Right Line Walking Infrared sensor port
const int SensorLeft = A2;     // Set Left Line Walking Infrared sensor port
int SL;    // State of Left Line Walking Infrared sensor
int SR;    // State of Right Line Walking Infrared sensor
/*Infrared obstacle avoidance*/
const int SensorRight_2 = A4;    // Right  Infrared sensor
const int SensorLeft_2 = A5;     // Left  Infrared sensor
int SL_2;    // State of Left  Infrared sensor
int SR_2;    // State of Right  Infrared sensor
void setup()
{
  //Initialize motor drive for output mode
  pinMode(Left_motor_go,OUTPUT);
  pinMode(Left_motor_back,OUTPUT);
  pinMode(Right_motor_go,OUTPUT);
  pinMode(Right_motor_back,OUTPUT);
  pinMode(key,INPUT);// Set button as input
  pinMode(beep,OUTPUT);// Set buzzer as output
  pinMode(SensorRight, INPUT); // Set Right Line Walking Infrared sensor as input
  pinMode(SensorLeft, INPUT); // Set left Line Walking Infrared sensor as input
  pinMode(SensorRight_2, INPUT); //Set Right  Infrared sensor as input
  pinMode(SensorLeft_2, INPUT); //Set left Infrared sensor as input
  digitalWrite(key,HIGH);//Initialize button
  digitalWrite(beep,HIGH);// set buzzer mute
}
//=======================Motor=========================
void run()
{
  digitalWrite(Right_motor_go,HIGH);// right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100);//PWM--Pulse Width Modulation(0~255). It can be
adjusted to control speed.
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH);// set left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100);//PWM--Pulse Width Modulation(0~255). It can be
adjusted to control speed.
  analogWrite(Left_motor_back,0);
}
void brake() //stop
{
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,LOW);
}
void left()//turn left
{
  digitalWrite(Right_motor_go,HIGH);  // right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100);
  analogWrite(Right_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,LOW);     // left motor stop
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
}
void spin_left(int time)         //Left rotation
{
  digitalWrite(Right_motor_go,HIGH); // right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100); // PWM--Pulse Width Modulation(0~255) control
speed
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,LOW);     // left motor back off
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,100); // PWM--Pulse Width Modulation(0~255) control
speed
```

```
  delay(time * 100);
}
void right() //turn right
{
  digitalWrite(Right_motor_go,LOW);    // right motor stop
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH);// left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100);
  analogWrite(Left_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
}
void spin_right(int time)        //Right rotation
{
  digitalWrite(Right_motor_go,LOW);   // right motor back off
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,200);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,HIGH);// left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200);
  analogWrite(Left_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
  delay(time * 100);
}
void back(int time)  //back off
{
  digitalWrite(Right_motor_go,LOW);  //right motor back off
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,150);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,LOW);  //left motor back off
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,150);// PWM--Pulse Width Modulation(0~255) control
speed
  delay(time * 100);
}
//==========================================================
void keysacn()
{
  int val;
  val=digitalRead(key);// Reads the button ,the level value assigns to val
  while(digitalRead(key))// When the button is not pressed
  {
    val=digitalRead(key);
  }
  while(!digitalRead(key))// When the button is pressed
  {
   delay(10); //delay 10ms
    val=digitalRead(key);// Reads the button ,the level value assigns to val
    if(val==LOW)  //Double check the button is pressed
    {
      digitalWrite(beep,LOW);//The buzzer sounds
      delay(50);//delay 50ms
    while(!digitalRead(key)) //Determine if the button is released or not
        digitalWrite(beep,HIGH);//mute
    }
    else
      digitalWrite(beep,HIGH);//mute
  }
}
/*main loop*/
void loop()
{
 keysacn();   //Press the button to start
  while(1)
  {
  /******************************************************************************************
  Infrared signal back means there is something obstacled ,returns low level and led
lights up.
  Infrared signal gone means there is nothing obstacled ,returns high level and led lights
off.
  ******************************************************************************************/
    SR_2 = digitalRead(SensorRight_2);//Right infrared sensor detects the obstacle,then
LED[L5] light illuminates and otherwise it goes off.
    SL_2 = digitalRead(SensorLeft_2);//Left infrared sensor detects the obstacle,then
LED[L4] light illuminates and otherwise it goes off.
    if (SL_2 == LOW&&SR_2==LOW)// Black lines were not detected at the same time
      run();   // go ahead
    else if (SL_2 == HIGH & SR_2 == LOW)// Left sensor against white undersurface and
right against black undersurface , the car left off track and need to adjust to the right.
      right();
    else if (SR_2 == HIGH & SL_2 == LOW) // Rihgt sensor against white undersurface
and left against black undersurface , the car right off track and need to adjust to the left.
      left();
    else  // Black lines were detected at the same time , the car stop.
    brake();
  }
}
```

Experimental steps:

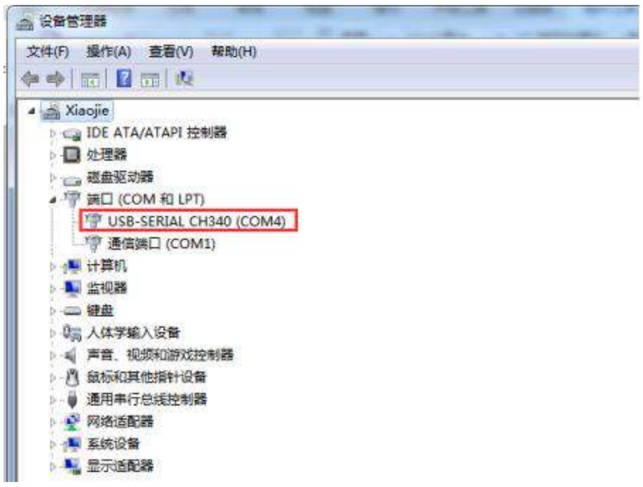
1. We need to open the code of this experiment: **Tracking.ino**, click"√" under the menu bar to compile the code, and wait for the word "**Done compiling** " in the lower right corner, as shown in the figure below.

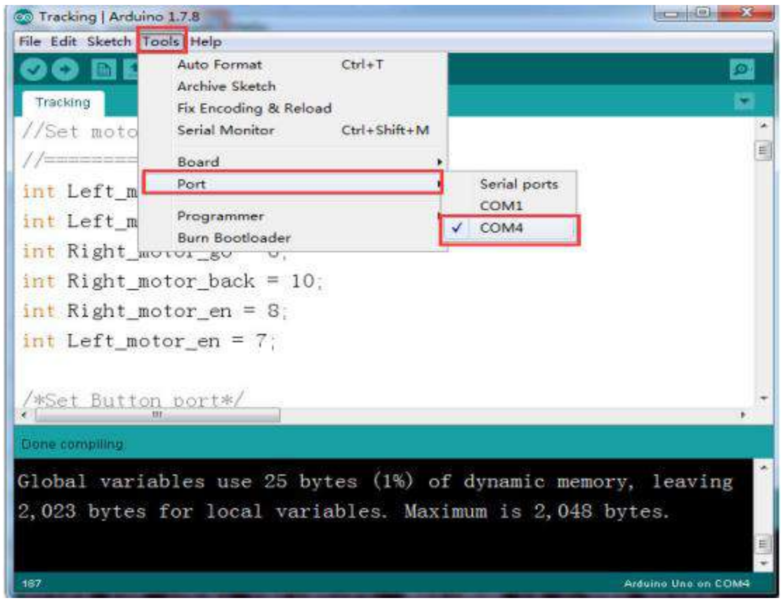


```
//Set motor port
//==============================
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;

/*Set Button port*/
```

```
Global variables use 25 bytes (1%) of dynamic memory, leaving
2,023 bytes for local variables. Maximum is 2,048 bytes.
```

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.

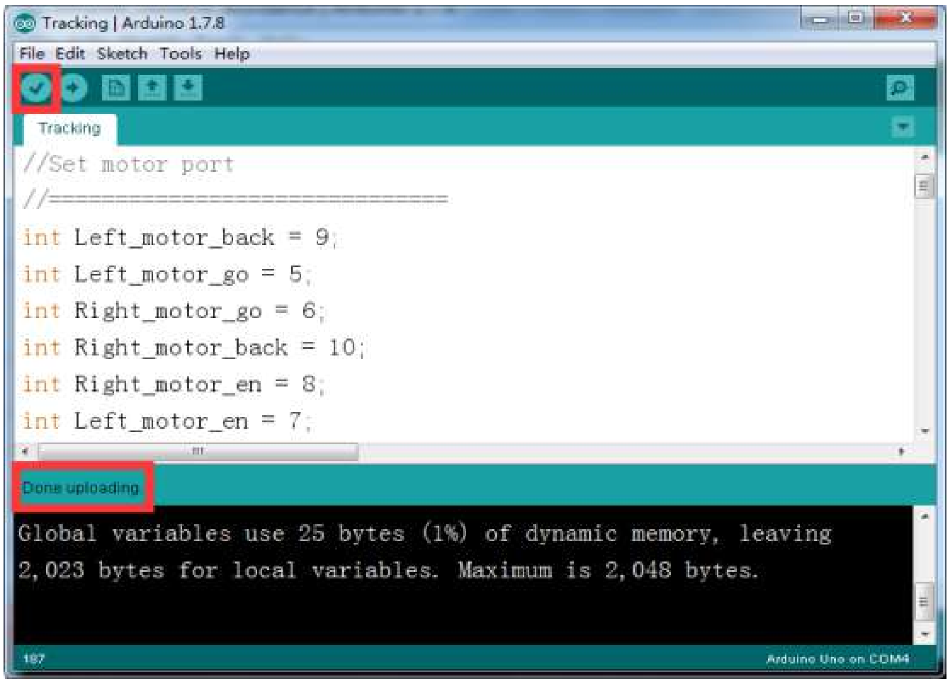




```
//Set moto
//=======
int Left_m
int Left_m
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;

/*Set Button port*/
```

```
Global variables use 25 bytes (1%) of dynamic memory, leaving
2,023 bytes for local variables. Maximum is 2,048 bytes.
```

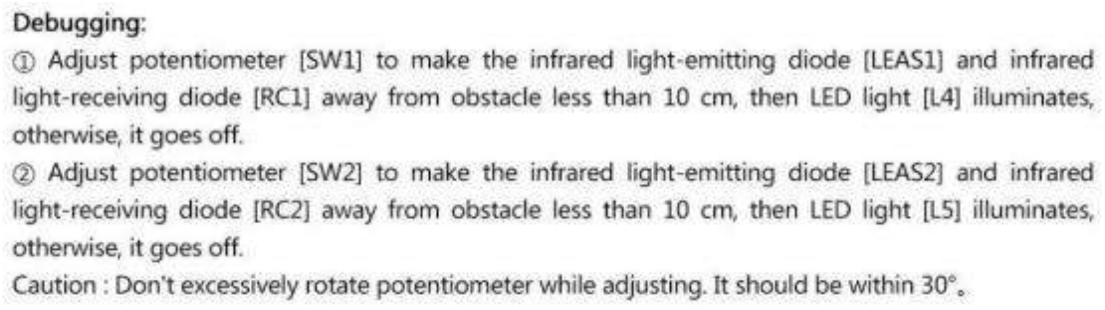
3. After the selection is completed, you need to click “→”under the menu bar to upload the code to the Arduino UNO board. When the word “**Done uploading**” appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.



```
//Set motor port
//==============================
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
```

```
Global variables use 25 bytes (1%) of dynamic memory, leaving
2,023 bytes for local variables. Maximum is 2,048 bytes.
```

4. After the program download is successful, debug potentiometers SW1 and SW2 as shown below.

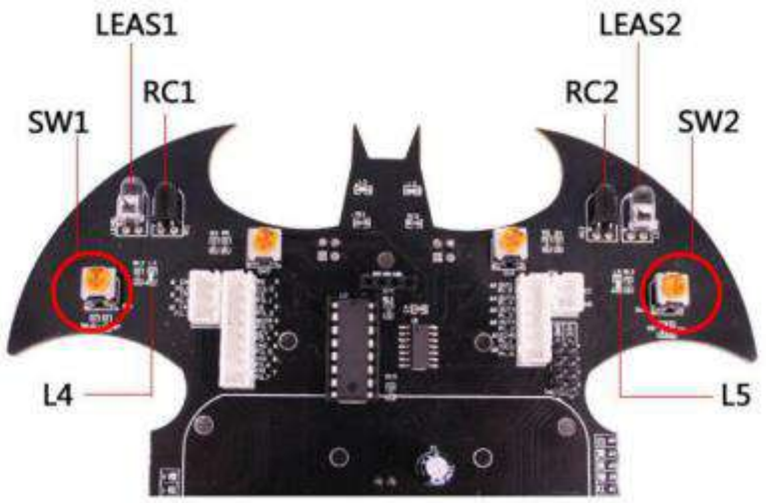


Debugging:

① Adjust potentiometer [SW1] to make the infrared light-emitting diode [LEAS1] and infrared light-receiving diode [RC1] away from obstacle less than 10 cm, 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 10 cm, then LED light [L5] illuminates, otherwise, it goes off.

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



5.Turn on the BatCar's power switch, press the start button K1, and after hearing the whistle, BatCar starts to follow the previous object.

6- Ultrasonic distance measurement

The purpose of the experiment:

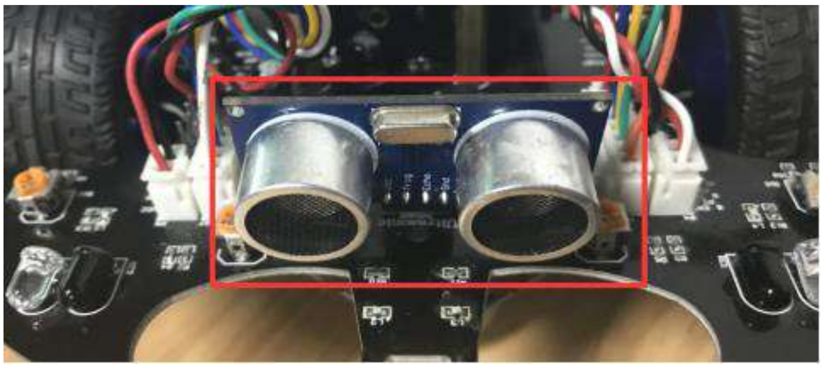
When the program upload is complete, open the serial monitor and you can see the distance measured by the ultrasonic sensor.

List of components required for the experiment:

BatCar*1

USB data cable*1

Ultrasonic sensor*1

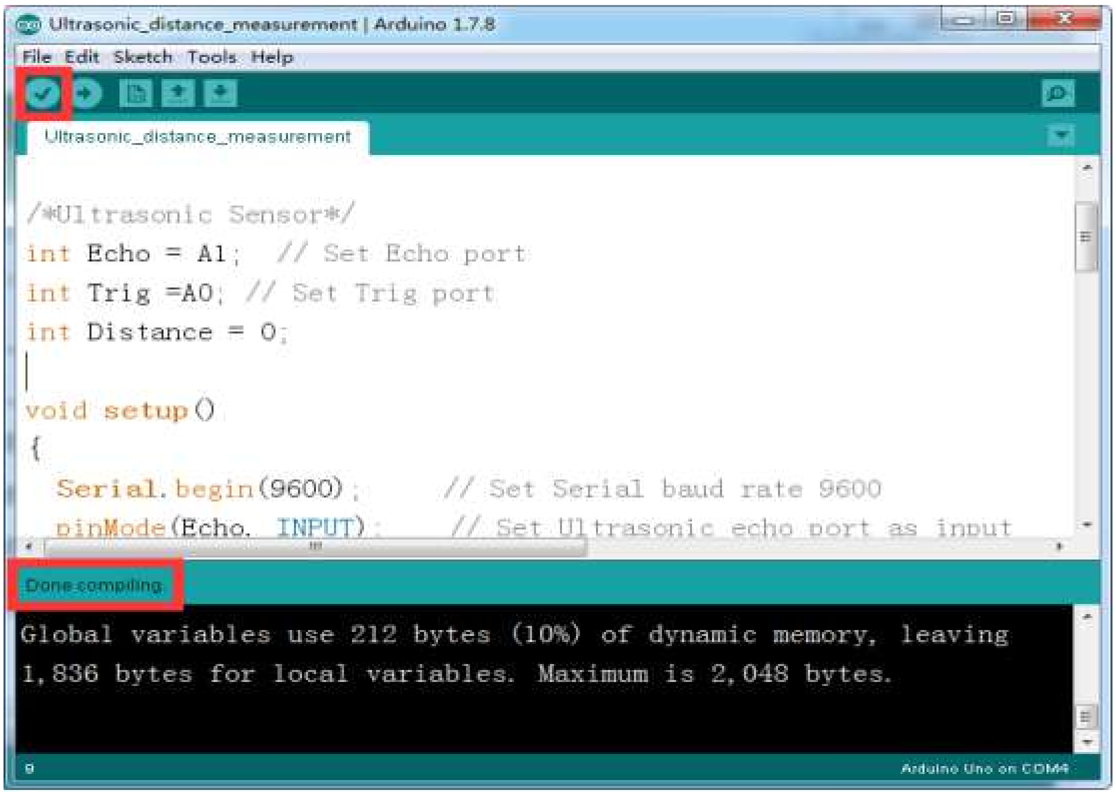


Experimental code analysis:

```
int Echo = A1;  // Set Echo port
int Trig =A0; // Set Trig port
int Distance = 0;
void setup()
{
  Serial.begin(9600);     // Set Serial baud rate 9600
  pinMode(Echo, INPUT);    // Set Ultrasonic echo port as input
  pinMode(Trig, OUTPUT);   // Set Ultrasonic trig port as input
}

void Distance_test()    // Measuring front distance
{
  digitalWrite(Trig, LOW);     // set trig port low level for 2μs
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH);  // set trig port high level for 10μs(at least 10μs)
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);     // set trig port low level
  float Fdistance = pulseIn(Echo, HIGH);  // Read echo port high level time(unit:μs)
  Fdistance= Fdistance/58;       // Distance(m) =(time(s) * 344(m/s)) / 2     /****** The
speed of sound is 344m/s.*******/
                                // ==> 2*Distance(cm) = time(µs) * 0.0344(cm/µs)
                                // ==> Distance(cm) = time(µs) * 0.0172 = time(µs) / 58
  Serial.print("Distance:");      //Output Distance(cm)
  Serial.println(Fdistance);         //display distance
  Distance = Fdistance;
}
/*main loop*/
void loop()
{
  Distance_test();//Measure and display distance
  delay(200);
}
```

Experimental steps:

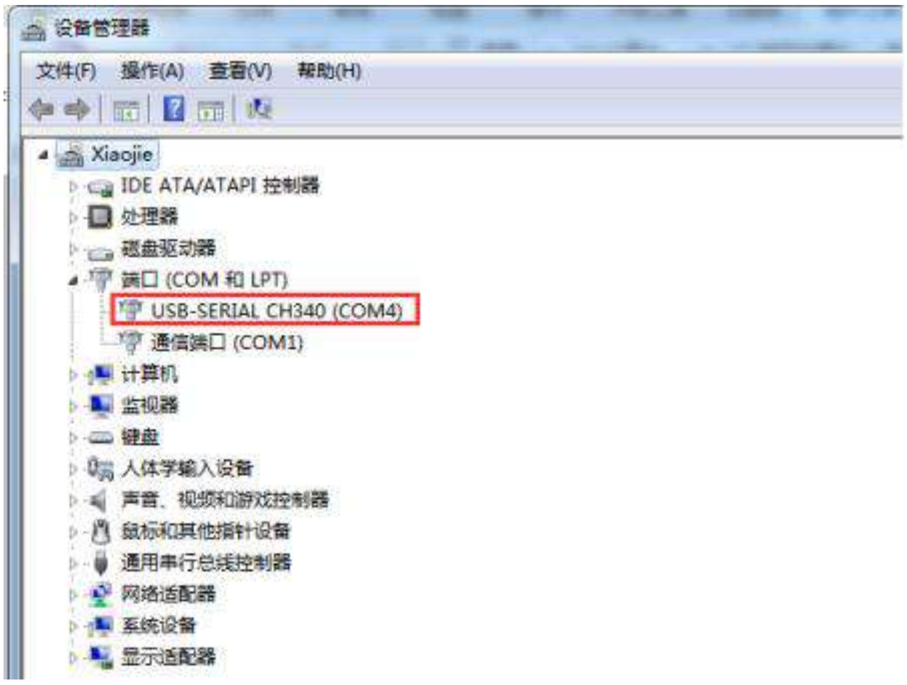
1. We need to open the code of this experiment:**Ultrasonic_distance_measurement.ino**, click"√" under the menu bar to compile the code, and wait for the word "**Done compiling** " in the lower right corner, as shown in the figure below.


Ultrasonic_distance_measurement | Arduino 1.7.8
File Edit Sketch Tools Help
Ultrasonic_distance_measurement
Done compiling
Arduino Uno on COM4

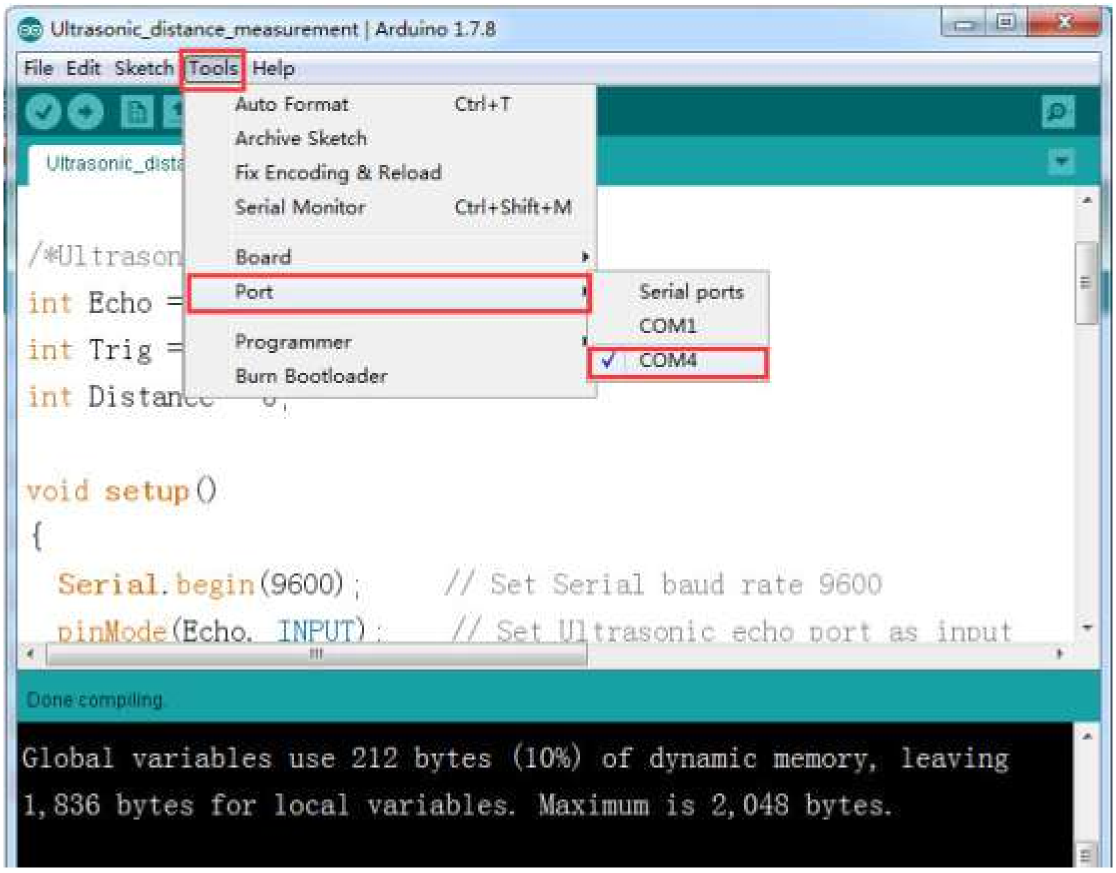
```
/*Ultrasonic Sensor*/
int Echo = A1;  // Set Echo port
int Trig =A0; // Set Trig port
int Distance = 0;

void setup()
{
  Serial.begin(9600);      // Set Serial baud rate 9600
  pinMode(Echo, INPUT);     // Set Ultrasonic echo port as input
```

```
Global variables use 212 bytes (10%) of dynamic memory, leaving
1,836 bytes for local variables. Maximum is 2,048 bytes.
```

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.



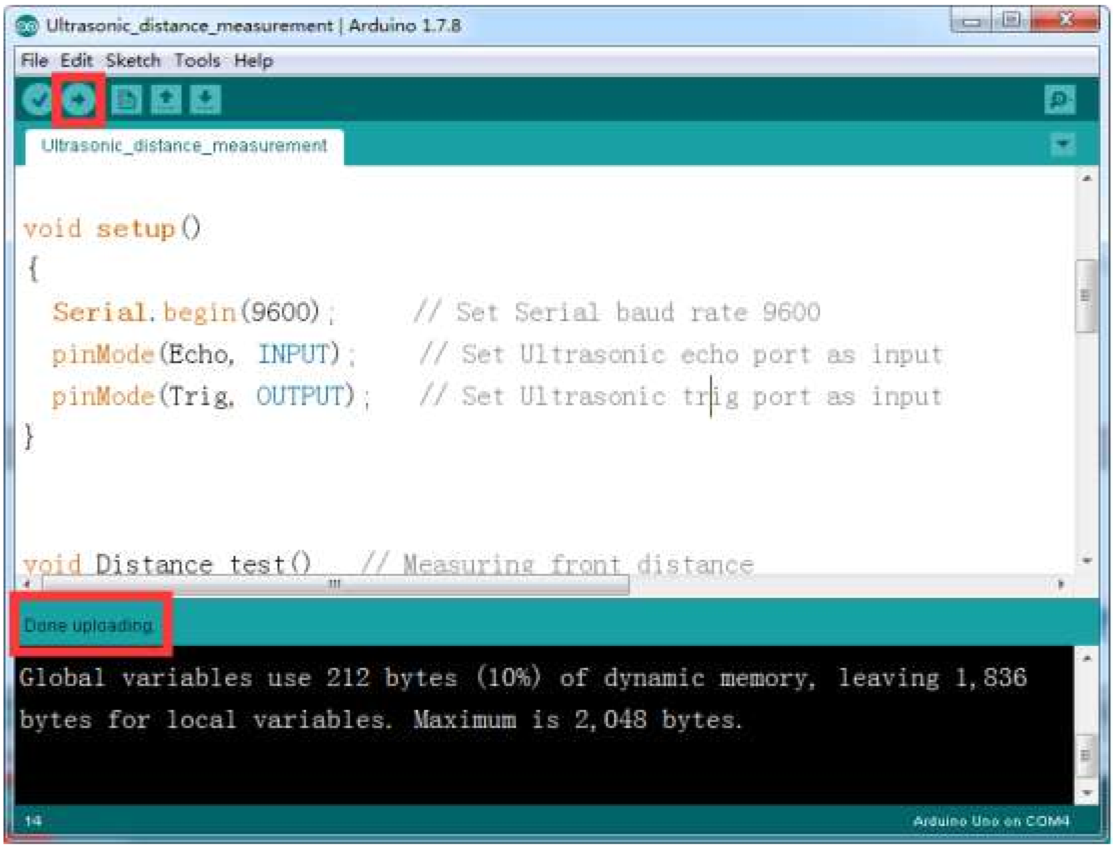
Xiaojie
- IDE ATA/ATAPI 控制器
- 处理器
- 磁盘驱动器
- 端口 (COM 和 LPT)
  - USB-SERIAL CH340 (COM4)
  - 通信端口 (COM1)
- 计算机
- 监视器
- 键盘
- 人体学输入设备
- 声音、视频和游戏控制器
- 鼠标和其他指针设备
- 通用串行总线控制器
- 网络适配器
- 系统设备
- 显示适配器



```
/*Ultrason
int Echo =
int Trig =
int Distance = 0;

void setup()
{
  Serial.begin(9600);       // Set Serial baud rate 9600
  pinMode(Echo, INPUT);     // Set Ultrasonic echo port as input
```

```
Global variables use 212 bytes (10%) of dynamic memory, leaving
1,836 bytes for local variables. Maximum is 2,048 bytes.
```

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



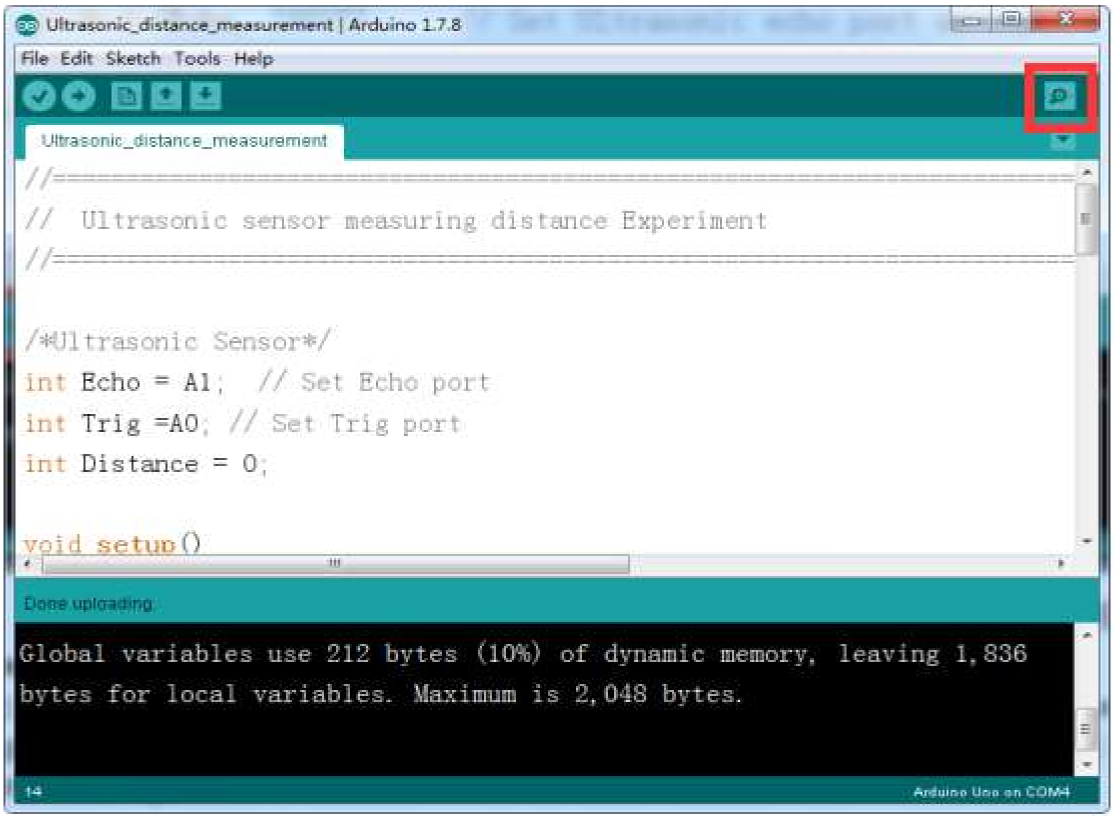
Ultrasonic_distance_measurement | Arduino 1.7.8
File Edit Sketch Tools Help
Ultrasonic_distance_measurement

```
void setup()
{
  Serial.begin(9600);      // Set Serial baud rate 9600
  pinMode(Echo, INPUT);    // Set Ultrasonic echo port as input
  pinMode(Trig, OUTPUT);   // Set Ultrasonic trig port as input
}


void Distance_test()   // Measuring front distance
```

Done uploading.

```
Global variables use 212 bytes (10%) of dynamic memory, leaving 1,836
bytes for local variables. Maximum is 2,048 bytes.
```

4. After the program is successfully uploaded, do not unplug the USB cable. Click on the serial port monitor as shown below.



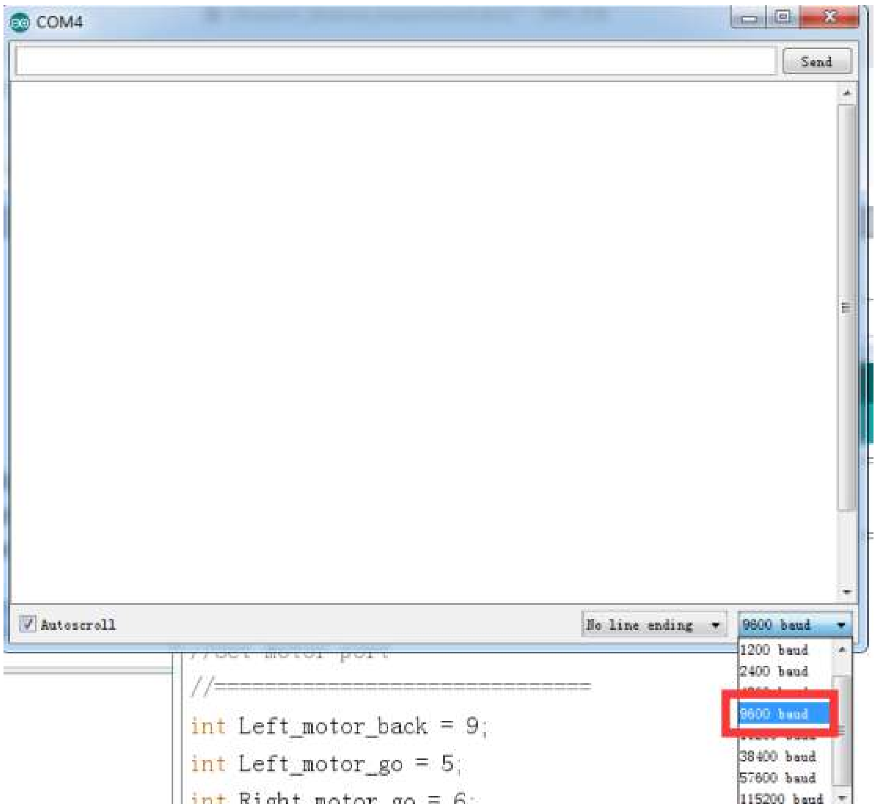
Ultrasonic_distance_measurement | Arduino 1.7.8
File Edit Sketch Tools Help
Ultrasonic_distance_measurement

```
//=============================================================================
//  Ultrasonic sensor measuring distance Experiment
//=============================================================================

/*Ultrasonic Sensor*/
int Echo = A1;  // Set Echo port
int Trig =A0; // Set Trig port
int Distance = 0;

void setup()
```

Done uploading.

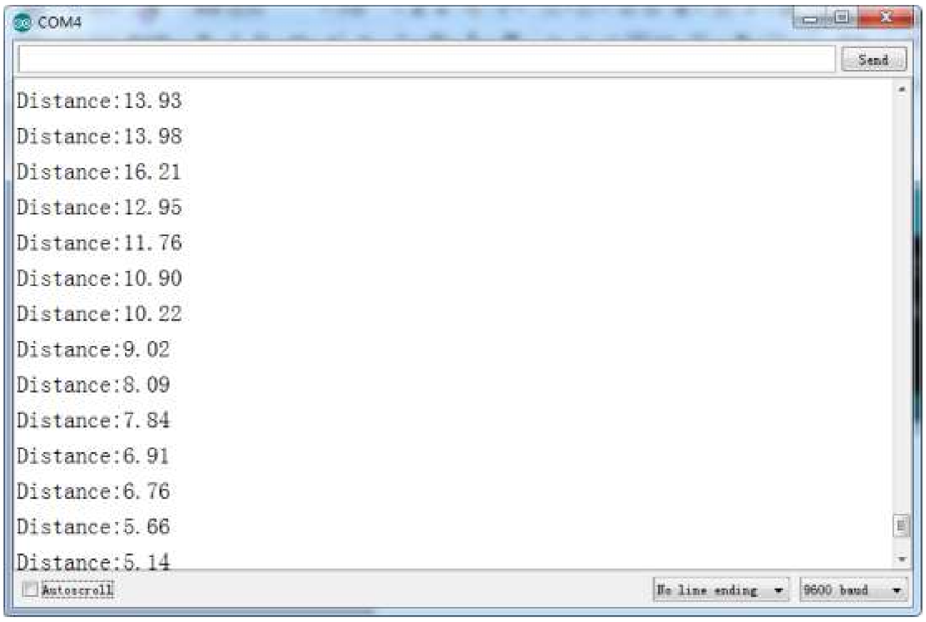
```
Global variables use 212 bytes (10%) of dynamic memory, leaving 1,836
bytes for local variables. Maximum is 2,048 bytes.
```

5. The serial port baud rate is 9600 as shown below.

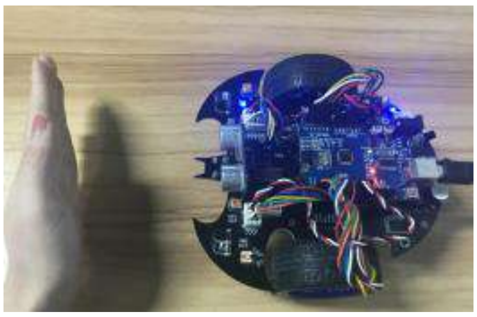
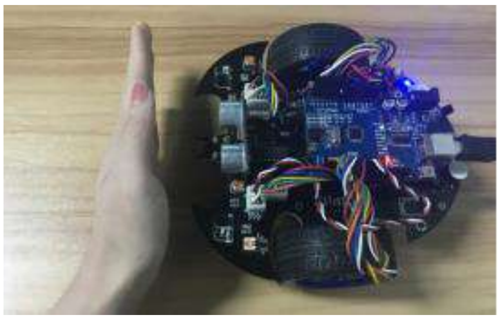


```
//=============================
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
```

6. It can be seen that the data measured by the ultrasonic sensor is refreshed every 200 milliseconds in the serial monitor, and the data is slowly moved forward by the hand in front of the ultrasonic sensor, and the data measured by the ultrasonic is gradually reduced.



```
Distance:13.93
Distance:13.98
Distance:16.21
Distance:12.95
Distance:11.76
Distance:10.90
Distance:10.22
Distance:9.02
Distance:8.09
Distance:7.84
Distance:6.91
Distance:6.76
Distance:5.66
Distance:5.14
```

7- Ultrasonic check obstacle and avoid

The purpose of the experiment:

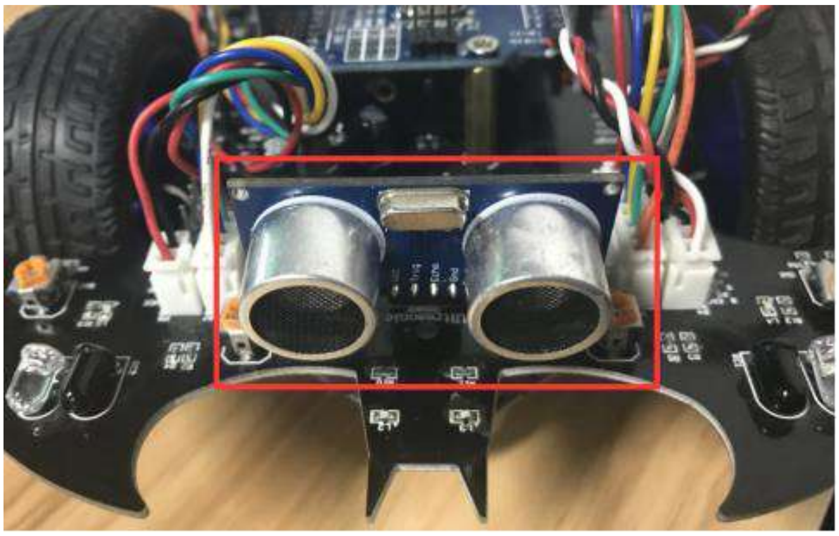
After the program is uploaded, place the BatCar in an open place and place some cartons as obstacles. Turn on the BatCar's power switch, press the start button K1, and after hearing the whistle, BatCar starts using the ultrasonic sensor to detect obstacles in front and avoid obstacles.

List of components required for the experiment:

BatCar*1

USB data cable*1

ultrasonic sensor*1



Experimental code analysis:

```
int Echo = A1;  // Set Echo port
int Trig =A0; // Set Trig port
int Distance = 0;
//==============================
//Set motor port
//==============================
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
/*Set Button port*/
```

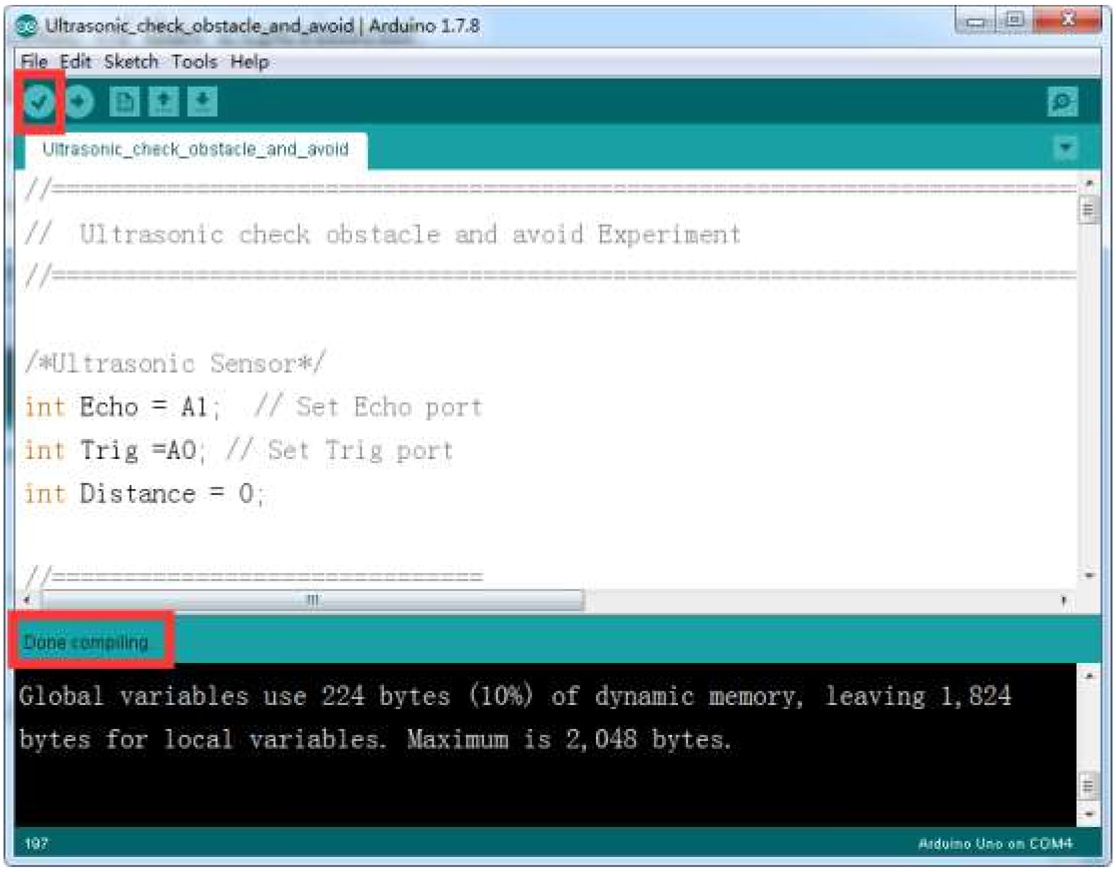
```
int key=4;
/*Set BUZZER port*/
int beep=3;
void setup()
{
  //Initialize motor drive for output mode
  pinMode(Left_motor_go,OUTPUT);
  pinMode(Left_motor_back,OUTPUT);
  pinMode(Right_motor_go,OUTPUT);
  pinMode(Right_motor_back,OUTPUT);
  pinMode(key,INPUT);// Set button as input
  pinMode(beep,OUTPUT);// Set buzzer as output
  pinMode(Echo, INPUT);    // Set Ultrasonic echo port as input
  pinMode(Trig, OUTPUT);   // Set Ultrasonic trig port as input
  digitalWrite(key,HIGH);//Initialize button
  digitalWrite(beep,HIGH);// set buzzer mute
}
//======================Motor=========================
void run()
{
  digitalWrite(Right_motor_go,HIGH);// right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100);//PWM--Pulse Width Modulation(0~255). It can be
adjusted to control speed.
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH);// set left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100);//PWM--Pulse Width Modulation(0~255). It can be
adjusted to control speed.
  analogWrite(Left_motor_back,0);
}
void brake() //stop
{
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,LOW);
}
void left()//turn left
{
  digitalWrite(Right_motor_go,HIGH); // right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100);
  analogWrite(Right_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,LOW);    // left motor stop
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
}
void spin_left(int time)         //Left rotation
```

```
{
  digitalWrite(Right_motor_go,HIGH); // right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100); // PWM--Pulse Width Modulation(0~255) control
speed
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,LOW);    // left motor back off
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,100); // PWM--Pulse Width Modulation(0~255) control
speed
  delay(time * 100);
}
void right() //turn right
{
  digitalWrite(Right_motor_go,LOW);    // right motor stop
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH);// left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100);
  analogWrite(Left_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
}
void spin_right(int time)        //Right rotation
{
  digitalWrite(Right_motor_go,LOW);   // right motor back off
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,100);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,HIGH);// left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100);
  analogWrite(Left_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
  delay(time * 100);
}
void back(int time)  //back off
{
  digitalWrite(Right_motor_go,LOW);  //right motor back off
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,100);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,LOW);  //left motor back off
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,100);// PWM--Pulse Width Modulation(0~255) control
speed
  delay(time * 100);
```

```
}
//============================================================
void keysacn()
{
  int val;
  val=digitalRead(key);// Reads the button ,the level value assigns to val
  while(digitalRead(key))// When the button is not pressed
  {
    val=digitalRead(key);
  }
  while(!digitalRead(key))// When the button is pressed
  {
   delay(10); //delay 10ms
    val=digitalRead(key);// Reads the button ,the level value assigns to val
    if(val==LOW)  //Double check the button is pressed
    {
        digitalWrite(beep,LOW);//The buzzer sounds
        delay(50);//delay 50ms
        while(!digitalRead(key)) //Determine if the button is released or not
        digitalWrite(beep,HIGH);//mute
    }
    else
      digitalWrite(beep,HIGH);//mute
  }
}
void Distance_test()   // Measuring front distance
{
  digitalWrite(Trig, LOW);    // set trig port low level for 2μs
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH);  // set trig port high level for 10μs(at least 10μs)
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);    // set trig port low level
  float Fdistance = pulseIn(Echo, HIGH);  // Read echo port high level time(unit:μs)
  Fdistance= Fdistance/58;       // Distance(m) =(time(s) * 344(m/s)) / 2     /****** The
speed of sound is 344m/s.*******/
                                //  ==> 2*Distance(cm) = time(μs) * 0.0344(cm/μs)
                                // ==> Distance(cm) = time(μs) * 0.0172 = time(μs) / 58
  Serial.print("Distance:");      //Output Distance(cm)
  Serial.println(Fdistance);         //display distance
  Distance = Fdistance;
}
/*main loop*/
void loop()
{
  keysacn();//Press the button to start
  while(1)
  {
    Distance_test();// Measuring front distance

    if(Distance < 25)//The value is the distance that meets the obstacle, and can be set
according to the actual situation
    {
      delay(10);
      Distance_test();//Measuring front distance
      while(Distance<25)//Determine whether there is an obstruction again.If there is obstacle , turn the direction and determine again.
      {
        spin_right(3);//Right rotation for 300ms
        brake();//stop
        delay(100);
        Distance_test();//Measuring front distance
      }
    }
    else
       run();//There is nothing obstacled. Go ahead.
  }
}
```

Experimental steps:

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

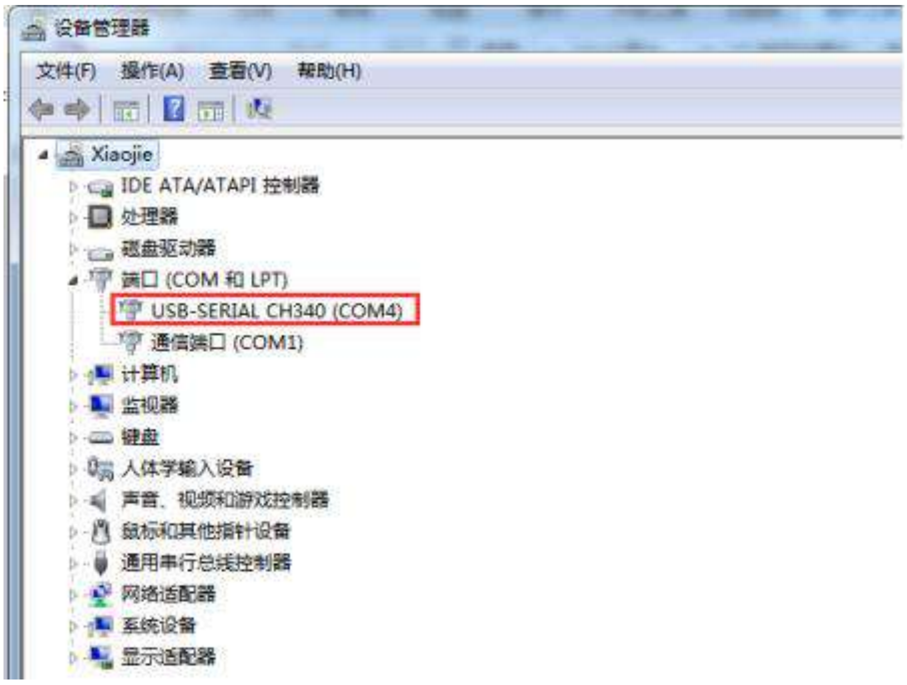

Ultrasonic_check_obstacle_and_avoid | Arduino 1.7.8
File Edit Sketch Tools Help
Ultrasonic_check_obstacle_and_avoid

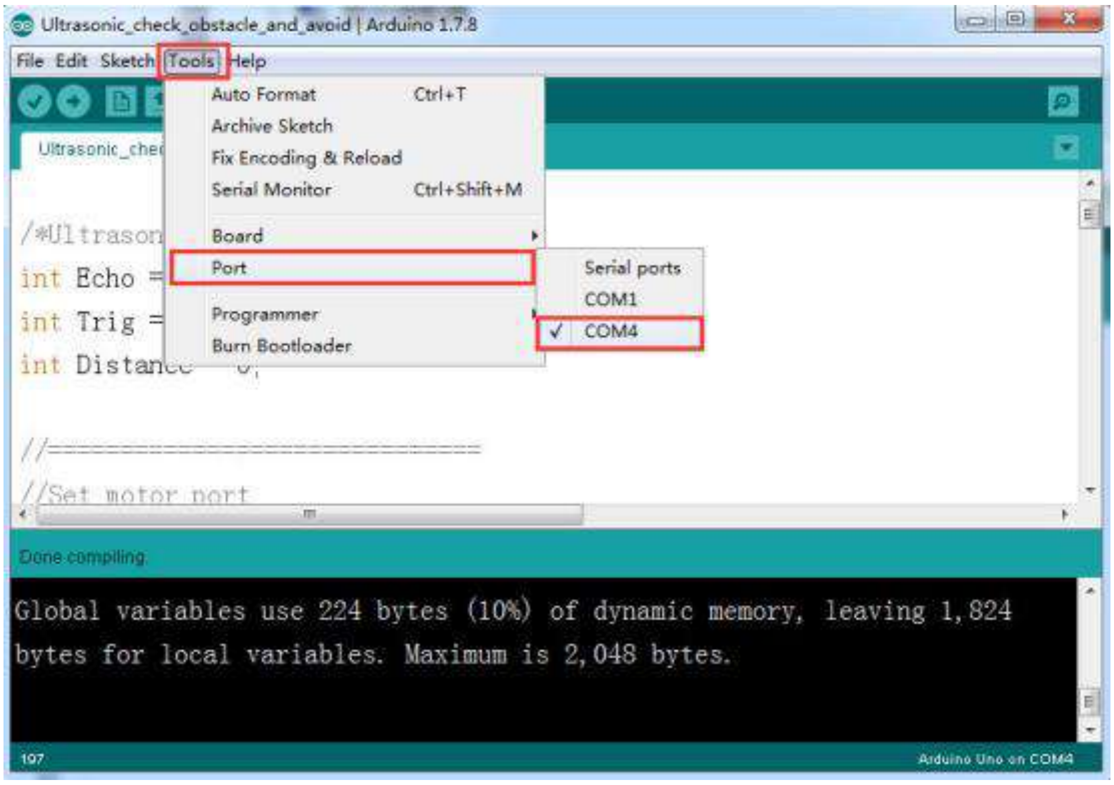
```
//=============================================================================
//  Ultrasonic check obstacle and avoid Experiment
//=============================================================================

/*Ultrasonic Sensor*/
int Echo = A1;  // Set Echo port
int Trig =A0;  // Set Trig port
int Distance = 0;

//==============================
```

Done compiling.

```
Global variables use 224 bytes (10%) of dynamic memory, leaving 1,824
bytes for local variables. Maximum is 2,048 bytes.
```

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.





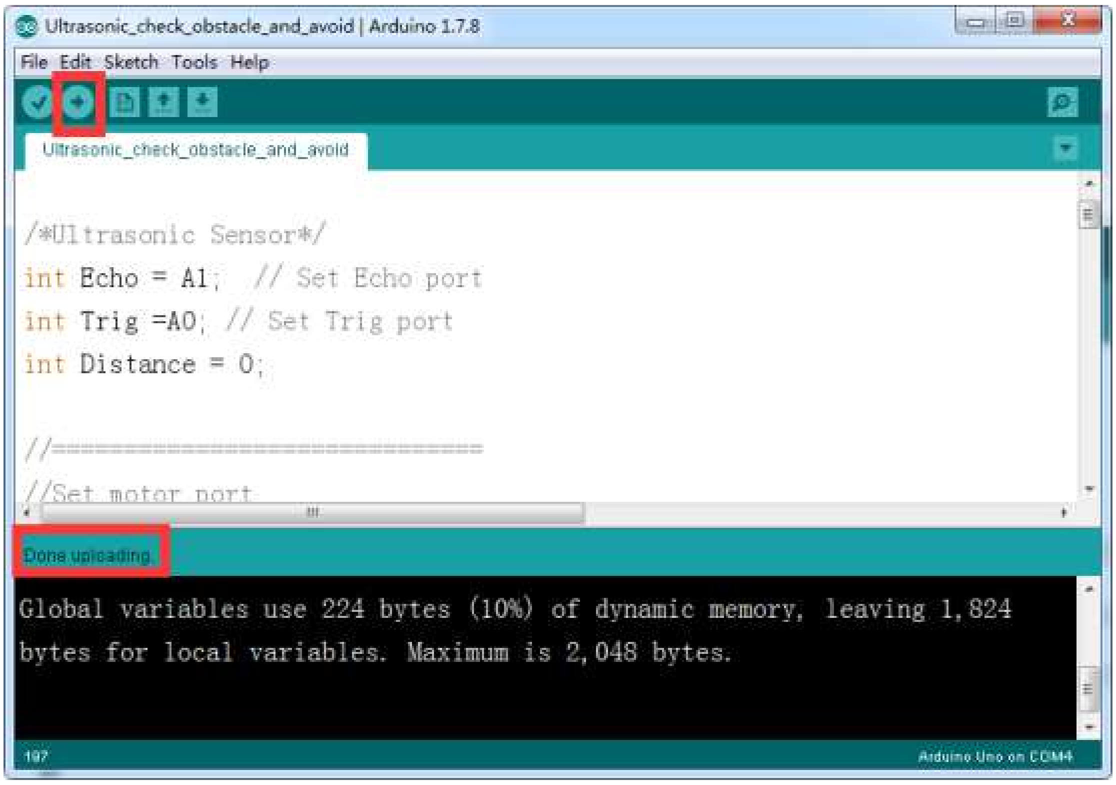
```
/*Ultrason
int Echo =
int Trig =
int Distance    0;

//==============================
//Set motor port
```

Done compiling.

```
Global variables use 224 bytes (10%) of dynamic memory, leaving 1,824
bytes for local variables. Maximum is 2,048 bytes.
```

197 Arduino Uno on COM4

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.


Ultrasonic_check_obstacle_and_avoid | Arduino 1.7.8
File Edit Sketch Tools Help
Ultrasonic_check_obstacle_and_avoid
Done uploading.
197
Arduino Uno on COM4

```
/*Ultrasonic Sensor*/
int Echo = A1;  // Set Echo port
int Trig =A0; // Set Trig port
int Distance = 0;

//================================
//Set motor port
```

```
Global variables use 224 bytes (10%) of dynamic memory, leaving 1,824
bytes for local variables. Maximum is 2,048 bytes.
```

4.After the program is uploaded, place the BatCar in an open place and place some cartons as obstacles. Turn on the BatCar's power switch, press the start button K1, and after hearing the whistle, BatCar starts using the ultrasonic sensor to detect obstacles in front and avoid obstacles. If the front distance is within 25 cm of the obstacle, the BatCar will circumvent the obstacles, otherwise the BatCar will advance.

8- Line Walking and check obstacle

The purpose of the experiment:

After the program Tracking ultrasonic is successfully uploaded, unplug the USB cable and place the BatCar on the patrol tarck. After turning on the power switch and pressing the start button K1, BatCar starts to patrol the black line. If the ultrasonic sensor detects an obstacle within 20 cm of its front side, BatCar stops, otherwise it continues to patrol the black line.

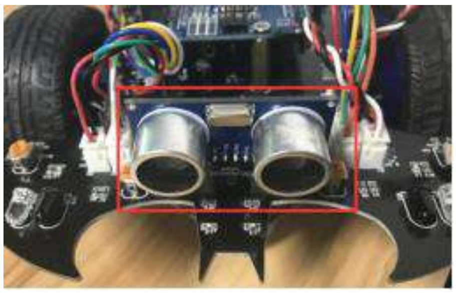
List of components required for the experiment:

BatCar*1

USB data cable*1

Patrol tarck*1

Ultrasonic sensor*1



Experimental code analysis:

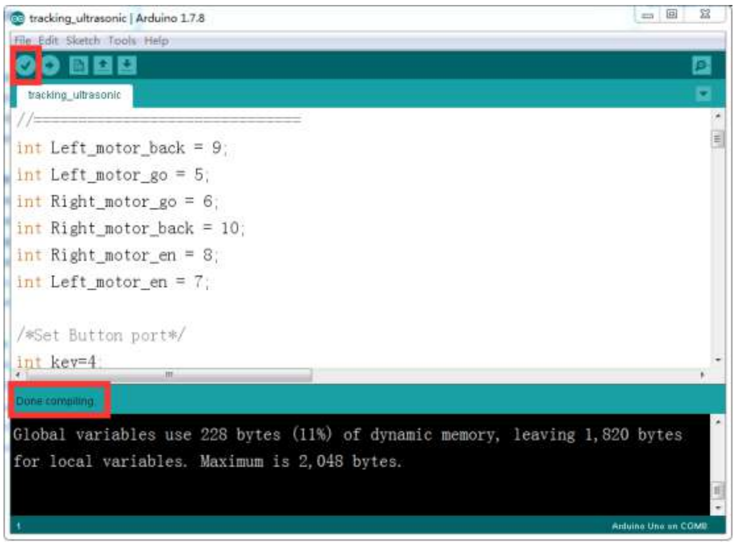
```
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
/*Set Button port*/
int key=4;
/*Set BUZZER port*/
int beep=3;
/*Line Walking*/
const int SensorRight = A3;    // Set Right Line Walking Infrared sensor port
const int SensorLeft = A2;     // Set Left Line Walking Infrared sensor port
int SL;    // State of Left Line Walking Infrared sensor
int SR;    // State of Right Line Walking Infrared sensor
/*Ultrasonic Sensor*/
int Echo = A1;  // Set Echo port
int Trig =A0; // Set Trig port
int Distance = 0;
void setup()
{
  //Initialize motor drive for output mode
  pinMode(Left_motor_go,OUTPUT);
  pinMode(Left_motor_back,OUTPUT);
  pinMode(Right_motor_go,OUTPUT);
  pinMode(Right_motor_back,OUTPUT);
  pinMode(key,INPUT);// Set button as input
  pinMode(beep,OUTPUT);// Set buzzer as output
  pinMode(SensorRight, INPUT); // Set Right Line Walking Infrared sensor as input
  pinMode(SensorLeft, INPUT); // Set left Line Walking Infrared sensor as input
  pinMode(Echo, INPUT);    // Set Ultrasonic echo port as input
  pinMode(Trig, OUTPUT);   // Set Ultrasonic trig port as input
  digitalWrite(key,HIGH);//Initialize button
  digitalWrite(beep,HIGH);// set buzzer mute
}
//=====================Motor========================
void run()
{
  digitalWrite(Right_motor_go,HIGH);// right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100);//PWM--Pulse Width Modulation(0~255). It can be
adjusted to control speed.
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH);// set left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100);//PWM--Pulse Width Modulation(0~255). It can be
adjusted to control speed.
  analogWrite(Left_motor_back,0);
}
void brake() //stop
{
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,LOW);
}
void left()//turn left
{
  digitalWrite(Right_motor_go,HIGH); // right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100);
  analogWrite(Right_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,LOW);    // left motor stop
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
}
```

```
void spin_left(int time)        //Left rotation
{
 digitalWrite(Right_motor_go,HIGH); // right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100); // PWM--Pulse Width Modulation(0~255) control
speed
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,LOW);    // left motor back off
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,100); // PWM--Pulse Width Modulation(0~255) control
speed
  delay(time * 100);
}
void right() //turn right
{
  digitalWrite(Right_motor_go,LOW);    // right motor stop
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH);// left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100);
  analogWrite(Left_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
}
void spin_right(int time)        //Right rotation
{
  digitalWrite(Right_motor_go,LOW);   // right motor back off
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,200);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,HIGH);// left motor go ahead
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200);
  analogWrite(Left_motor_back,0);// PWM--Pulse Width Modulation(0~255) control
speed
  delay(time * 100);
}
void back(int time)  //back off
{
  digitalWrite(Right_motor_go,LOW);  //right motor back off
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,150);// PWM--Pulse Width Modulation(0~255) control
speed
  digitalWrite(Left_motor_go,LOW);  //left motor back off
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,150);// PWM--Pulse Width Modulation(0~255) control
speed
```

```
  delay(time * 100);
}
//==========================================================
void keysacn()
{
  int val;
  val=digitalRead(key);// Reads the button ,the level value assigns to val
  while(digitalRead(key))// When the button is not pressed
  {
    val=digitalRead(key);
  }
  while(!digitalRead(key))// When the button is pressed
  {
   delay(10); //delay 10ms
    val=digitalRead(key);// Reads the button ,the level value assigns to val
    if(val==LOW)  //Double check the button is pressed
    {
        digitalWrite(beep,LOW);//The buzzer sounds
        delay(50);//delay 50ms
        while(!digitalRead(key)) //Determine if the button is released or not
        digitalWrite(beep,HIGH);//mute
    }
    else
      digitalWrite(beep,HIGH);//mute
  }
}
void Distance_test()   // Measuring front distance
{
  digitalWrite(Trig, LOW);    // set trig port low level for 2μs
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH);  // set trig port high level for 10μs(at least 10μs)
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);    // set trig port low level
  float Fdistance = pulseIn(Echo, HIGH);  // Read echo port high level time(unit:μs)
  Fdistance= Fdistance/58;       // Distance(m) =(time(s) * 344(m/s)) / 2     /****** The
speed of sound is 344m/s.*******/
                                // ==> 2*Distance(cm) = time(μs) * 0.0344(cm/μs)
                                // ==> Distance(cm) = time(μs) * 0.0172 = time(μs) / 58
  Serial.print("Distance:");      //Output Distance(cm)
  Serial.println(Fdistance);         //display distance
  Distance = Fdistance;
}
/*main loop*/
void loop()
{
  keysacn();//Press the button to start
  while(1)
  {
    /**************************************************************************************
    Infrared signal back means white undersurface ,returns low level and led lights up.
    Infrared signal gone means black undersurface ,returns high level and led lights off.
    *************************************************************************************/
    SR = digitalRead(SensorRight);//Right Line Walking Infrared sensor against white
undersurface,then LED[L2] light illuminates and while against black
undersurface,LED[L2] goes off
    SL = digitalRead(SensorLeft);//Left Line Walking Infrared sensor against white
undersurface,then LED[L3] light illuminates and while against black
undersurface,LED[L3] goes off
    Distance_test();// Measuring front distance
    if((Distance < 20))//The value is the distance that meets the obstacle, and can be set
according to the actual situation
      brake();//stop
    else
    {
      if (SL == LOW&&SR==LOW)// Black lines were not detected at the same time
        run();   // go ahead
      else if (SL == LOW & SR == HIGH)// Left sensor against white undersurface and
right against black undersurface , the car left off track and need to adjust to the right.
        right();
      else if (SR == LOW & SL ==  HIGH) // Rihgt sensor against white undersurface and
left against black undersurface , the car right off track and need to adjust to the left.
        left();
      else // Black lines were detected at the same time , the car stop.
        brake();
    }
  }
}
```

Experimental steps:

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



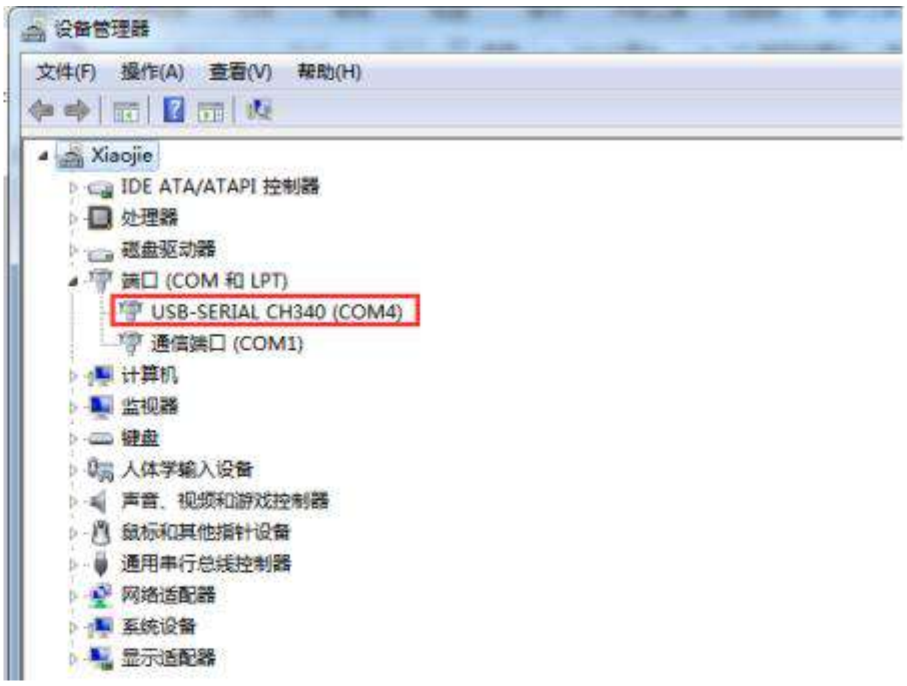
tracking_ultrasonic | Arduino 1.7.8

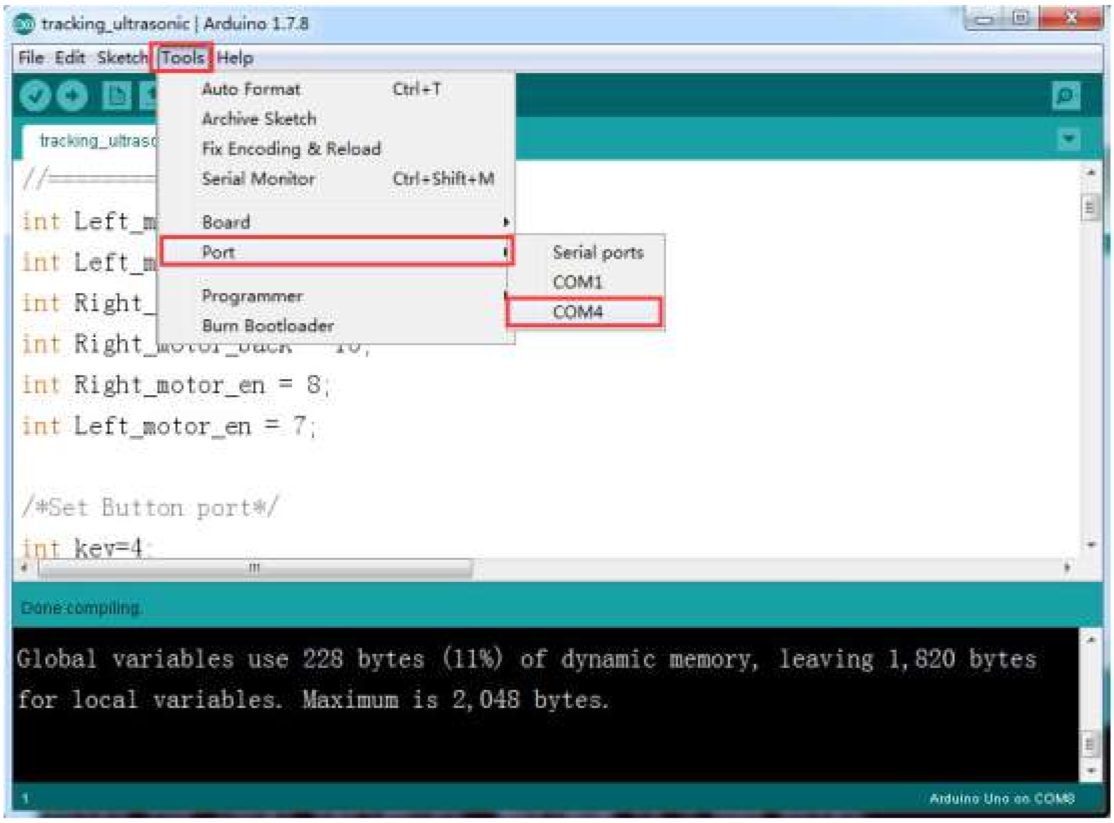
```
//================================
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;

/*Set Button port*/
int key=4;
```

Done compiling.

```
Global variables use 228 bytes (11%) of dynamic memory, leaving 1,820 bytes
for local variables. Maximum is 2,048 bytes.
```

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.

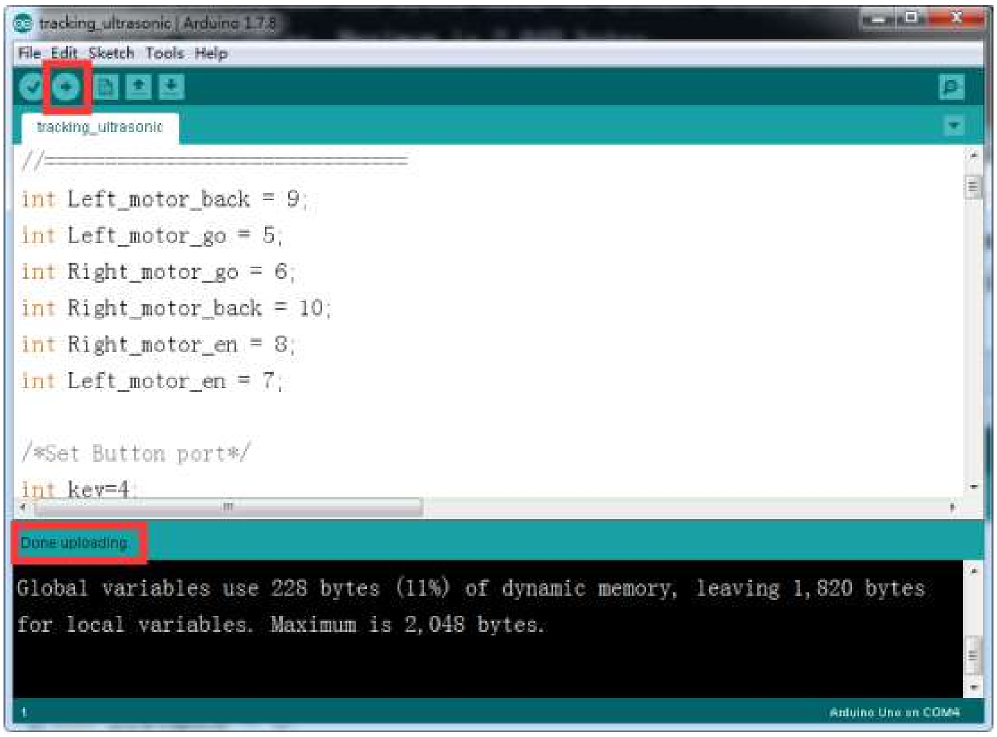




```
//=======
int Left_m
int Left_m
int Right_
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;

/*Set Button port*/
int key=4;
```

```
Global variables use 228 bytes (11%) of dynamic memory, leaving 1,820 bytes
for local variables. Maximum is 2,048 bytes.
```

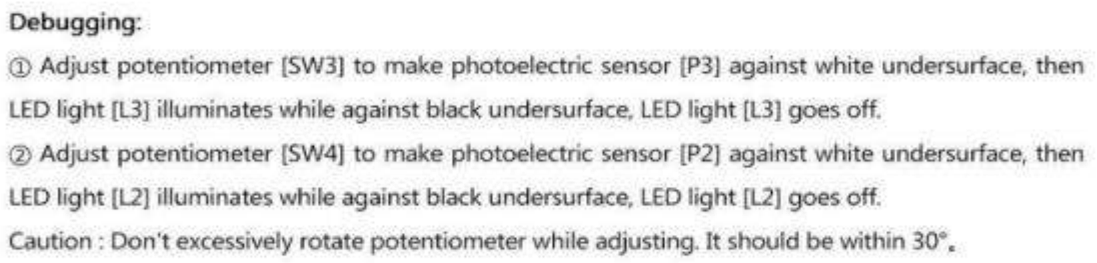
3. After the selection is completed, you need to click "→"under the menu bar to upload the code to the Arduino UNO board. When the word "**Done uploading**" appears in the lower left corner, the code has been successfully uploaded to the Arduino UNO board, as shown in the figure below.


tracking_ultrasonic | Arduino 1.7.8
File Edit Sketch Tools Help
tracking_ultrasonic
Done uploading.
Arduino Uno on COM4

```
//===============================
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;

/*Set Button port*/
int key=4;
```

```
Global variables use 228 bytes (11%) of dynamic memory, leaving 1,820 bytes
for local variables. Maximum is 2,048 bytes.
```

4. When the program upload is completed, unplug the USB data cable and place the BatCar on the tour linepatrol tarck after debugging according to the following figure.

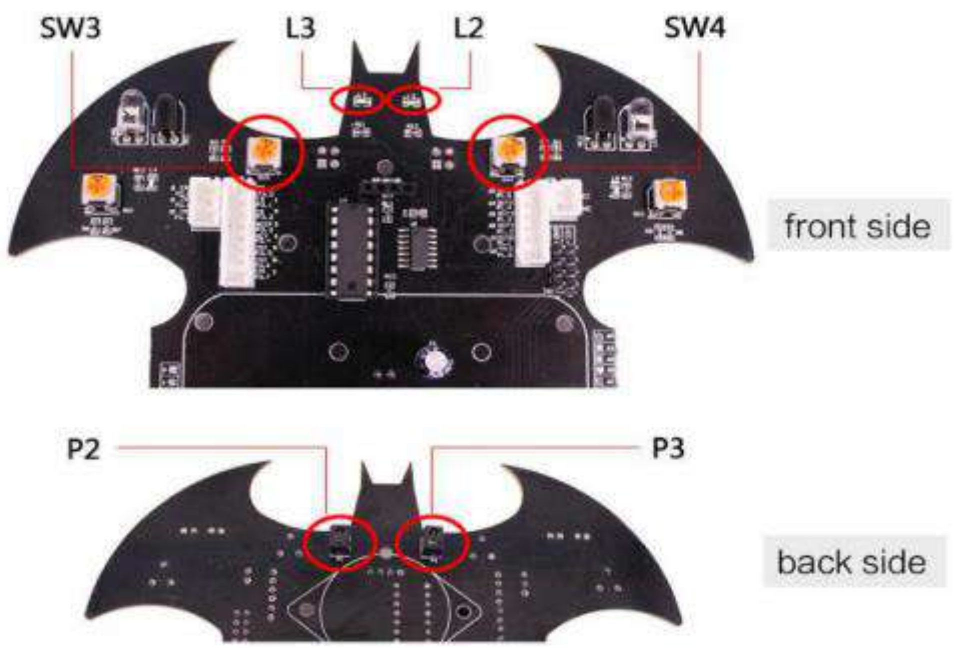


Debugging:

① Adjust potentiometer [SW3] to make photoelectric sensor [P3] against white undersurface, then LED light [L3] illuminates while against black undersurface, LED light [L3] goes off.

② Adjust potentiometer [SW4] to make photoelectric sensor [P2] against white undersurface, then LED light [L2] illuminates while against black undersurface, LED light [L2] goes off.

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



5.After turning on the power switch and pressing the start button K1, BatCar starts to patrol the black line. If the ultrasonic sensor detects an obstacle within 20 cm of its front side, BatCar stops, otherwise it continues to patrol the black line.

9- Remote control

The purpose of the experiment:

After uploading the **Remote control** 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 BatCar, then press the numeric keypad of the infrared remote control to control the BatCar to complete the corresponding action.

List of components required for the experiment:

BatCar*1

USB data cable*1

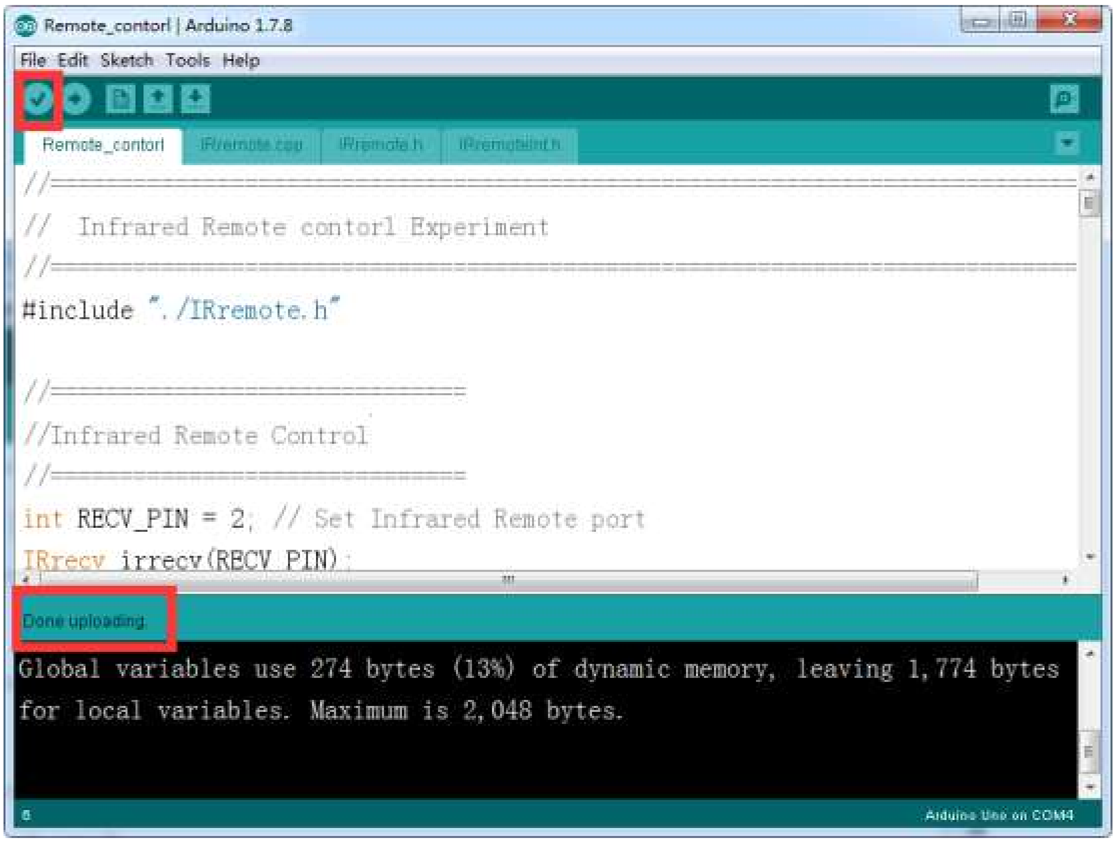
Infrared remote control*1

Experimental code analysis:

```
//=========================================================================
// Infrared Remote contorl Experiment
//=========================================================================
#include "./IRremote.h"
//==============================
//Infrared Remote Control
//==============================
int RECV_PIN = 2; // Set Infrared Remote port
IRrecv irrecv(RECV_PIN);
decode_results results; // Store infrared remote decode data
unsigned long last = millis();
int on = 0;//Infrared Received flag
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
//==============================
//Set motor port
//==============================
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
void setup()
{
  //Initialize motor drive for output mode
  pinMode(Left_motor_go,OUTPUT);
  pinMode(Left_motor_back,OUTPUT);
  pinMode(Right_motor_go,OUTPUT);
  pinMode(Right_motor_back,OUTPUT);
  pinMode(13, OUTPUT);//Show Infrared Remote Control Received Signal
  irrecv.enableIRIn(); // Start the receiver
}
void run()
{
  digitalWrite(Right_motor_go,HIGH);// right motor go ahead
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,HIGH);// set left motor go ahead
  digitalWrite(Left_motor_back,LOW);
}
void brake() //stop
{
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,LOW);
}
void left()//turn left
{
   digitalWrite(Right_motor_go,HIGH); // right motor go ahead
     digitalWrite(Right_motor_back,LOW);
     digitalWrite(Left_motor_go,LOW);    // left motor stop
     digitalWrite(Left_motor_back,LOW);
}
void spin_left()        //Left rotation
{
   digitalWrite(Right_motor_go,HIGH); // right motor go ahead
    digitalWrite(Right_motor_back,LOW);
    digitalWrite(Left_motor_go,LOW);    // left motor back off
   digitalWrite(Left_motor_back,HIGH);
}
void right() //turn right
{
  digitalWrite(Right_motor_go,LOW);    // right motor stop
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,HIGH);// left motor go ahead
  digitalWrite(Left_motor_back,LOW);
}
void spin_right()       //Right rotation
{
  digitalWrite(Right_motor_go,LOW);   // right motor back off
  digitalWrite(Right_motor_back,HIGH);
  digitalWrite(Left_motor_go,HIGH);// left motor go ahead
  digitalWrite(Left_motor_back,LOW);
}
void back()  //back off
{
  digitalWrite(Right_motor_go,LOW);  //right motor back off
  digitalWrite(Right_motor_back,HIGH);
  digitalWrite(Left_motor_go,LOW);  //left motor back off
  digitalWrite(Left_motor_back,HIGH);
}
//==========================================================
void dump(decode_results *results)//Decode Infrared Remote Control Received Signal
{
  int count = results->rawlen;
  if (results->decode_type == UNKNOWN)
  {
    brake();
  }
}
void loop()
{
  if (irrecv.decode(&results)) //receive infrared signal
  {
    if (millis() - last > 250) //make sure receive signal
    {
      on = !on;//Sign position reversal
      digitalWrite(13, on ? HIGH : LOW);//LED blink
      dump(&results);//decode
    }
    if (results.value == run_car )//key "2"
      run();//go ahead
    if (results.value == back_car )//key "8"
      back();//back off
    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();//stop
    if (results.value == left_turn )//key "1"
      spin_left();//Left rotation
    if (results.value == right_turn )//key "2"
      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.

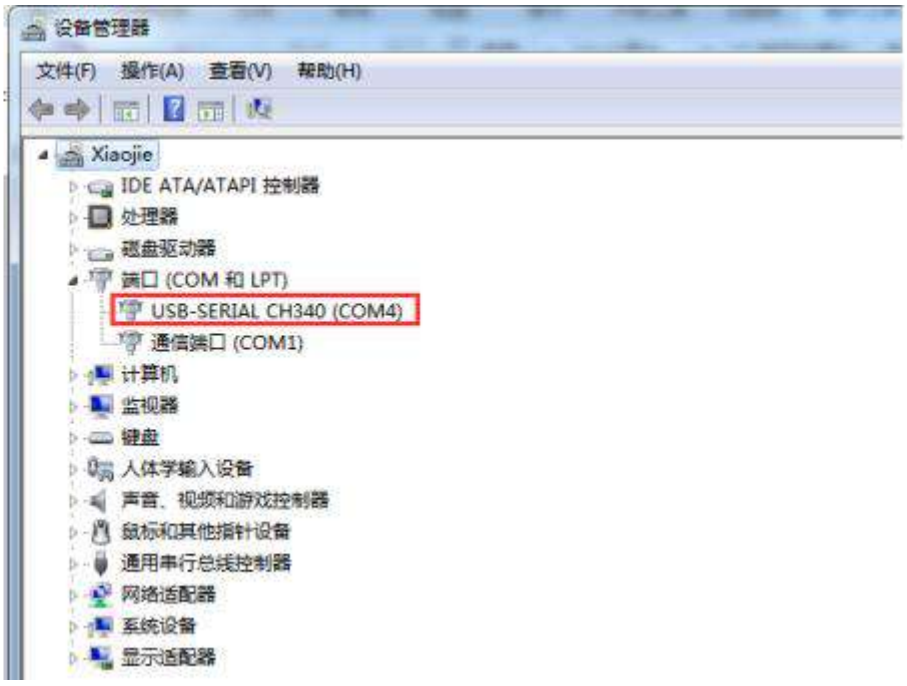


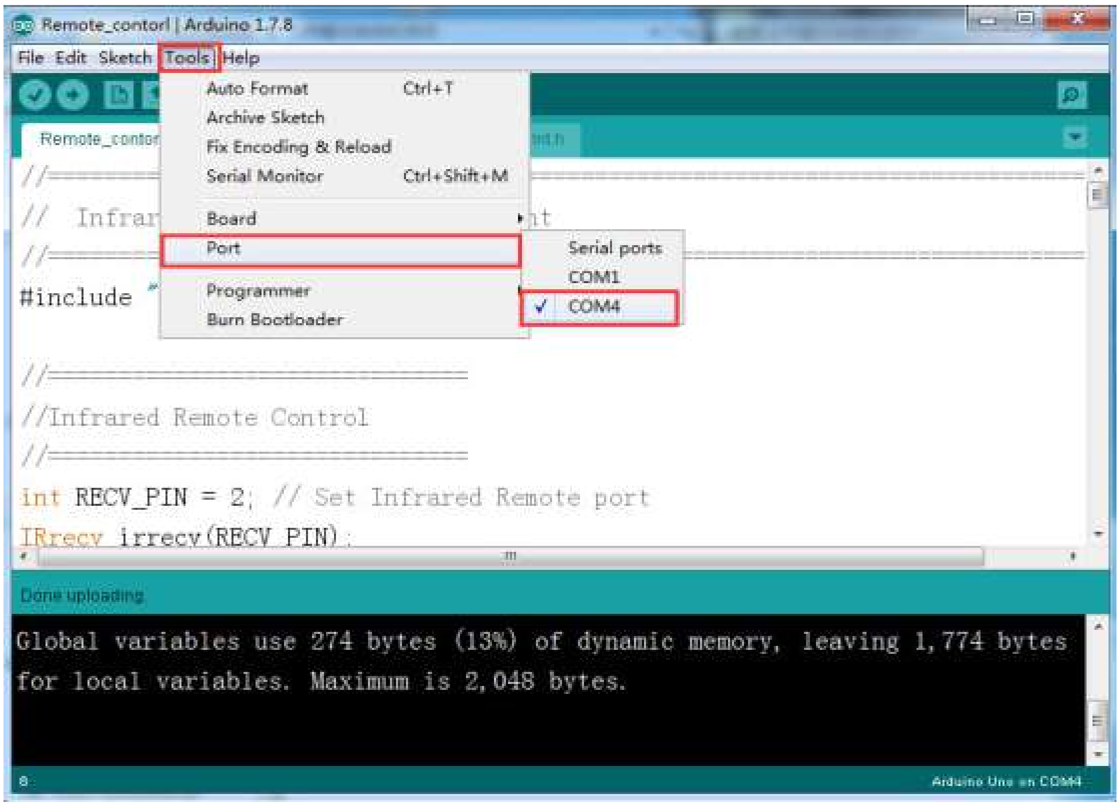
```
//==========================================================================
//  Infrared Remote contorl Experiment
//==========================================================================
#include "./IRremote.h"

//===============================
//Infrared Remote Control
//===============================
int RECV_PIN = 2; // Set Infrared Remote port
IRrecv irrecv(RECV_PIN);
```

```
Done uploading.
```

```
Global variables use 274 bytes (13%) of dynamic memory, leaving 1,774 bytes
for local variables. Maximum is 2,048 bytes.
```

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.

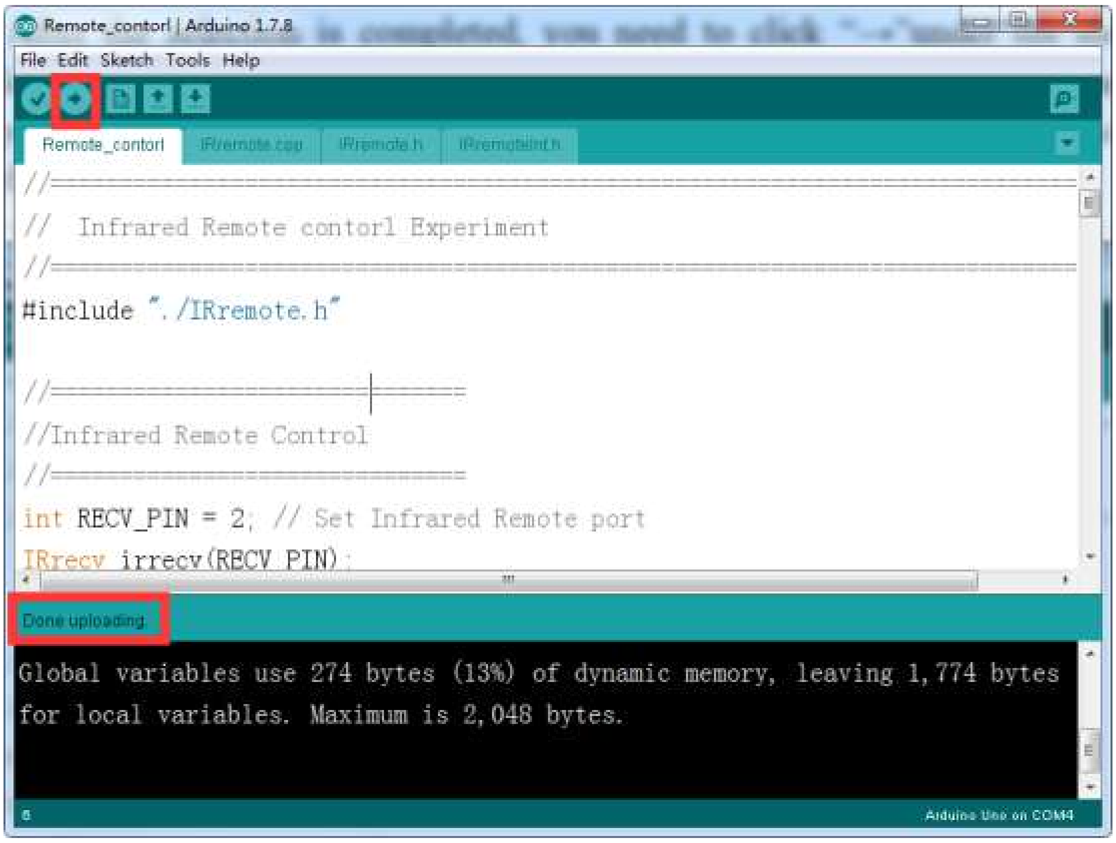




```
//=====
// Infrar
//=====
#include "
//===============================
//Infrared Remote Control
//===============================
int RECV_PIN = 2; // Set Infrared Remote port
IRrecv irrecv(RECV_PIN);
```

```
Global variables use 274 bytes (13%) of dynamic memory, leaving 1,774 bytes
for local variables. Maximum is 2,048 bytes.
```

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



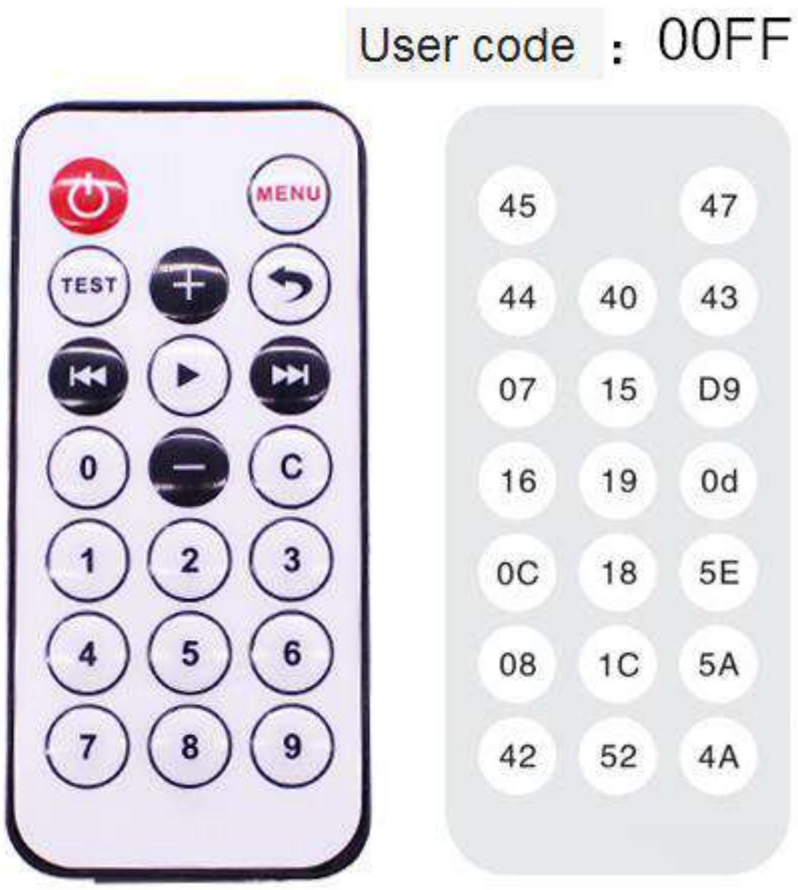
```
//=====================================================================
// Infrared Remote contorl Experiment
//=====================================================================
#include "./IRremote.h"

//===============================
//Infrared Remote Control
//===============================
int RECV_PIN = 2; // Set Infrared Remote port
IRrecv irrecv(RECV_PIN);
```

```
Global variables use 274 bytes (13%) of dynamic memory, leaving 1,774 bytes
for local variables. Maximum is 2,048 bytes.
```

4. After the program of **Remote contorl** is uploaded, put the BatCar indoors and pull the curtains to block the outdoor lights. Align the infrared remote control with the

infrared receiver at the rear of the BatCar. Press the digital button on the infrared remote control to control the BatCar to complete the corresponding action. The following is the user code value corresponding to the infrared remote control.

Remote control button	Corresponding user code value	The program controls the action of the BatCar
0	0x00FF6897	No control action
—	0x00FF9867	No control action
C	0x00FFB04F	No control action
1	0x00ff30CF	Left rotation
2	0x00FF18E7	Forward
3	0x00FF7A85	Right rotation
4	0x00FF10EF	Turn left
5	0x00FF38C7	Brake
6	0x00FF5AA5	Turn right
7	0x00FF42BD	No control action
8	0x00FF4AB5	Backward
9	0x00FF52AD	No control action


User code : 00FF
MENU
TEST
0
C
1
2
3
4
5
6
7
8
9
45
47
44
40
43
07
15
D9
16
19
0d
0C
18
5E
08
1C
5A
42
52
4A