Ultrasonic check obstacle and avoid

The purpose of the experiment:

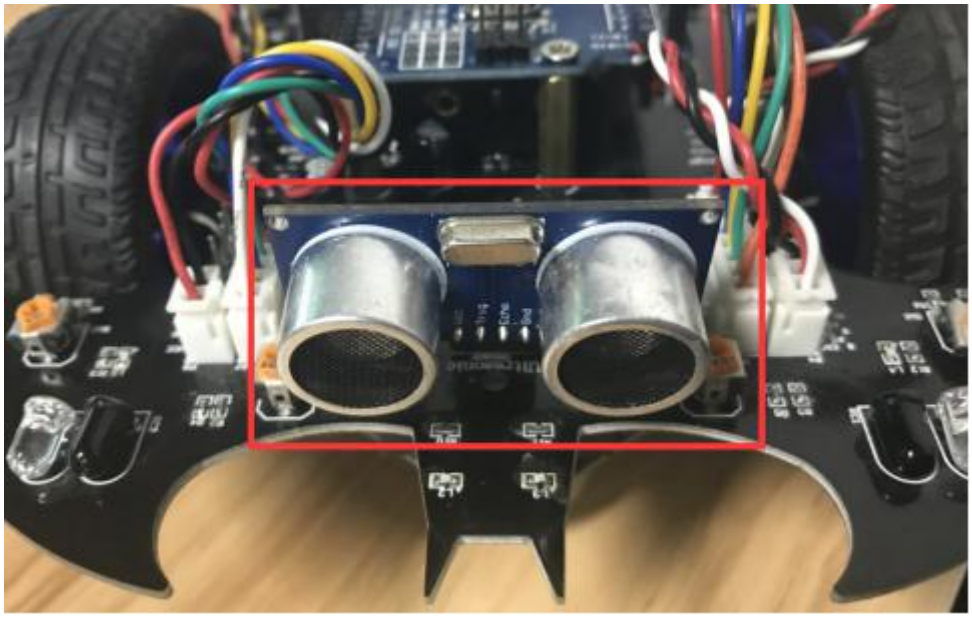
After the program is uploaded, place the BatCar in an open place and place some cartons as obstacles. Turn on the BatCar's power switch, press the start button K1, and after hearing the whistle, BatCar starts using the ultrasonic sensor to detect obstacles in front and avoid obstacles.

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)
```

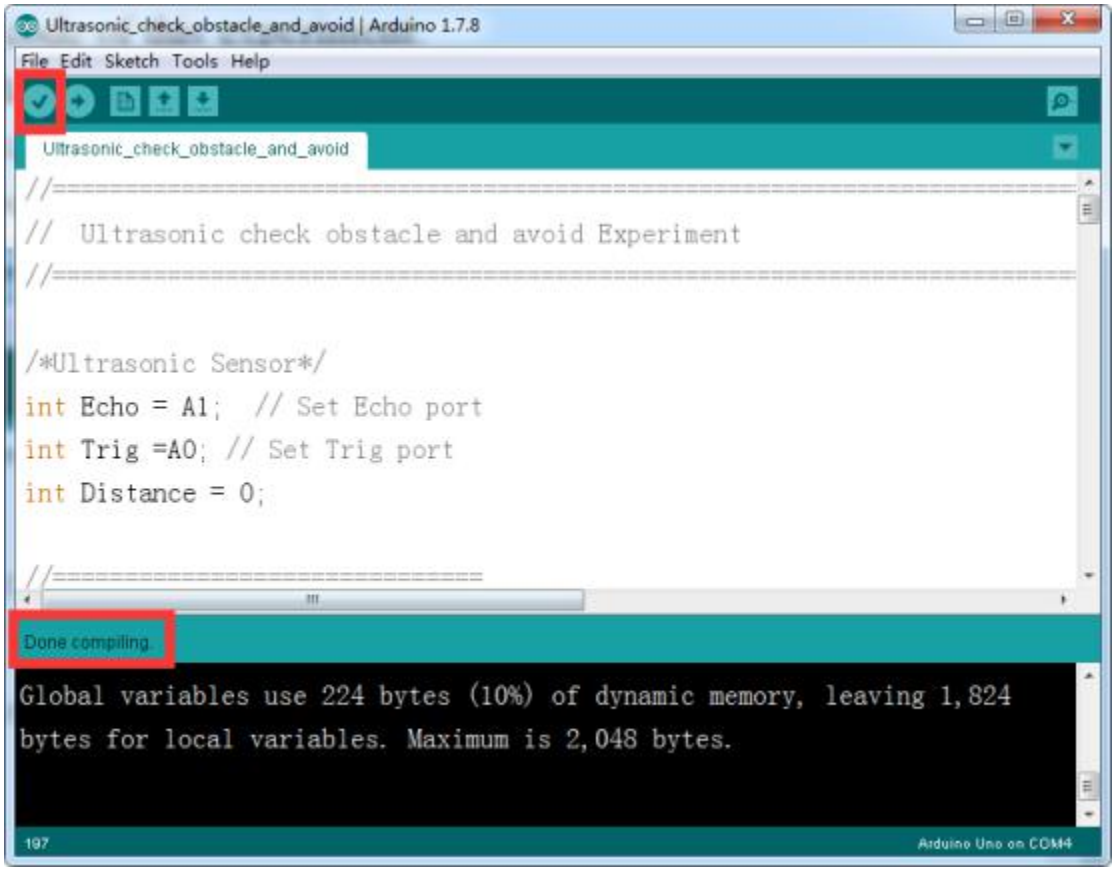
```
  Fdistance= Fdistance/58;       // Distance(m) =(time(s) * 344(m/s)) /
2     /****** The speed of sound is 344m/s.*******/
                              //   ==> 2*Distance(cm) = time(μs) * 0.0344(cm/μs)
                              // ==> Distance(cm) = time(μs) * 0.0172 = time(μs) / 58
  Serial.print("Distance:");       //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.
  }
}
```

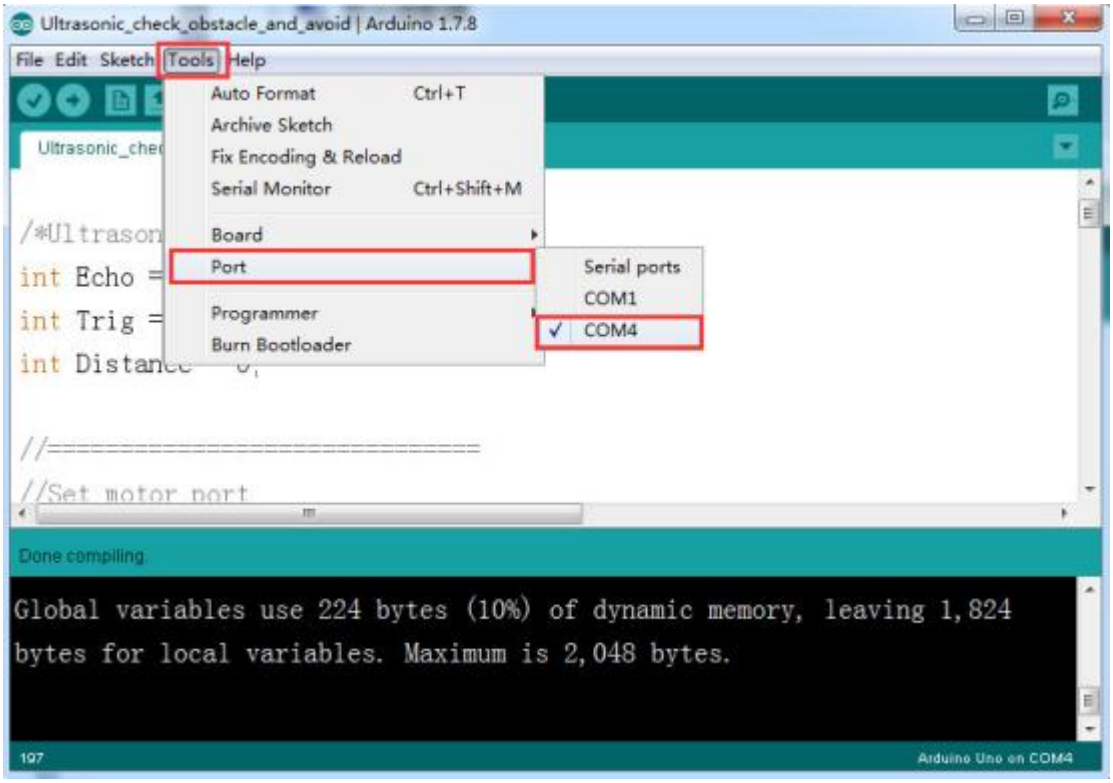
Experimental steps:

1. We need to open the code of this experiment: **Ultrasonic_check_obstacle_and_avoid.ino**, click“√” under the

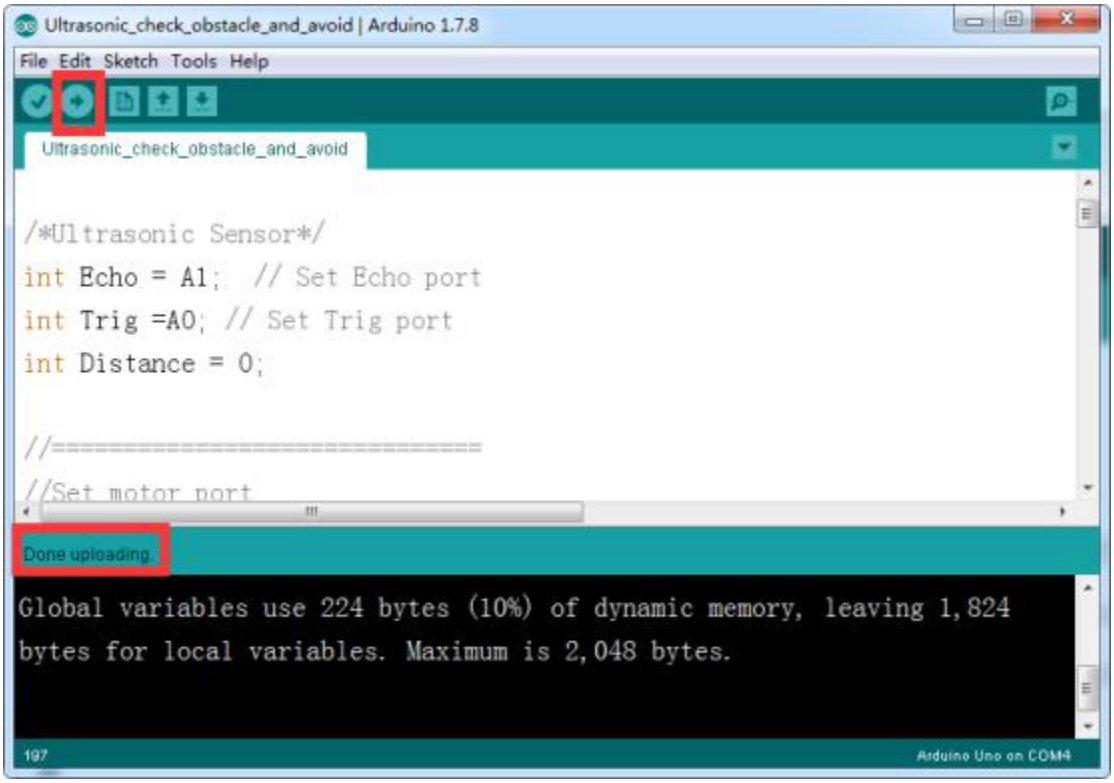
menu bar to compile the code, and wait for the word "**Done compiling** " in the lower right corner, as shown in the figure below.



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.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.