Tracking ultrasonic

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.

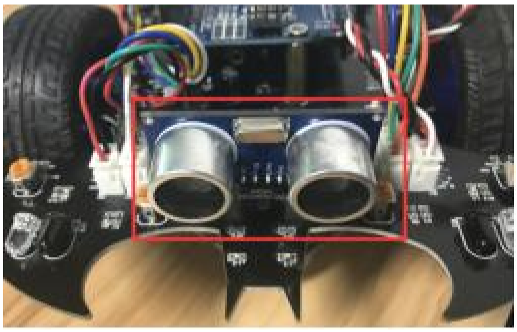
List of components required for the experiment:

BatCar*1

USB data cable*1

Patrol tarck*1

Ultrasonic sensor*1



Experimental code analysis:

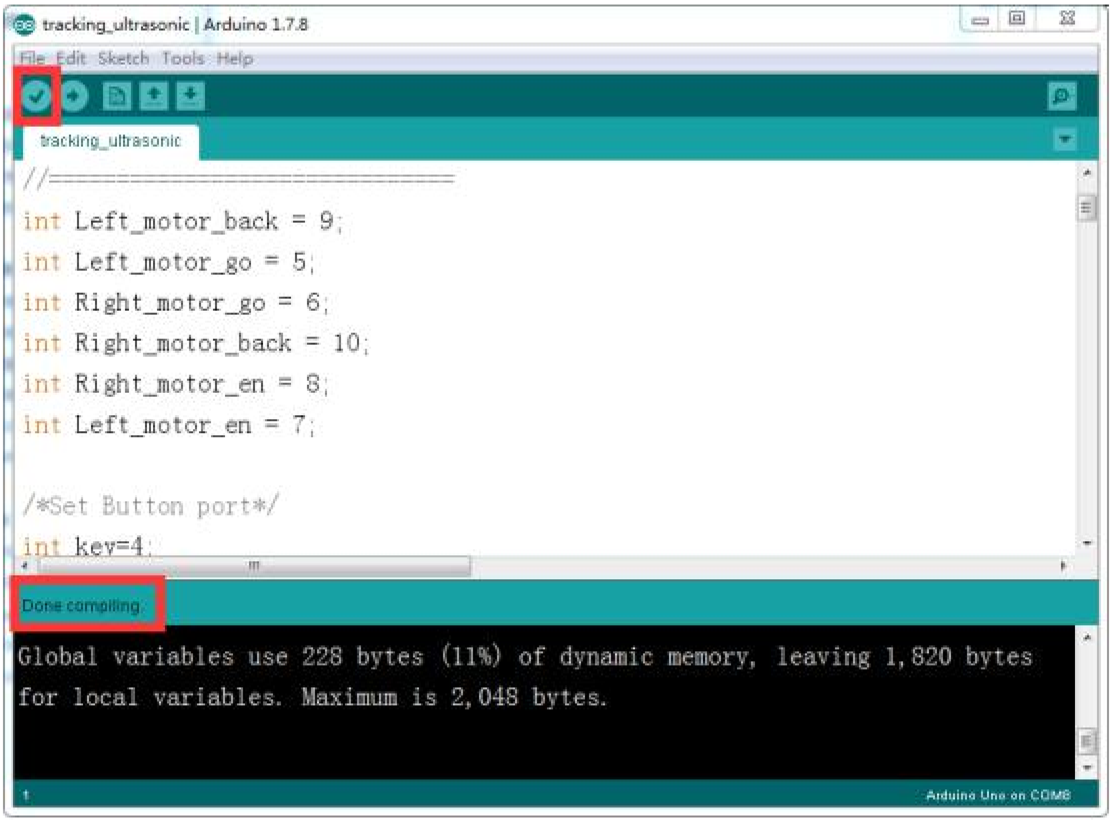
```
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
/*Set Button port*/
int key=4;
/*Set BUZZER port*/
```

```
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.

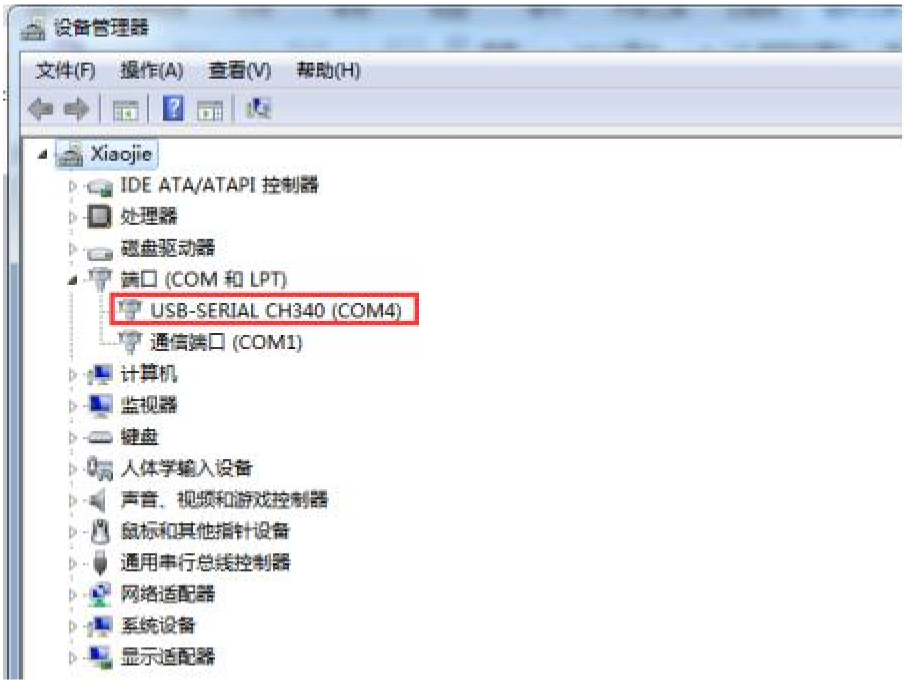

tracking_ultrasonic | Arduino 1.7.8
File Edit Sketch Tools Help
tracking_ultrasonic
Arduino Uno on COM8

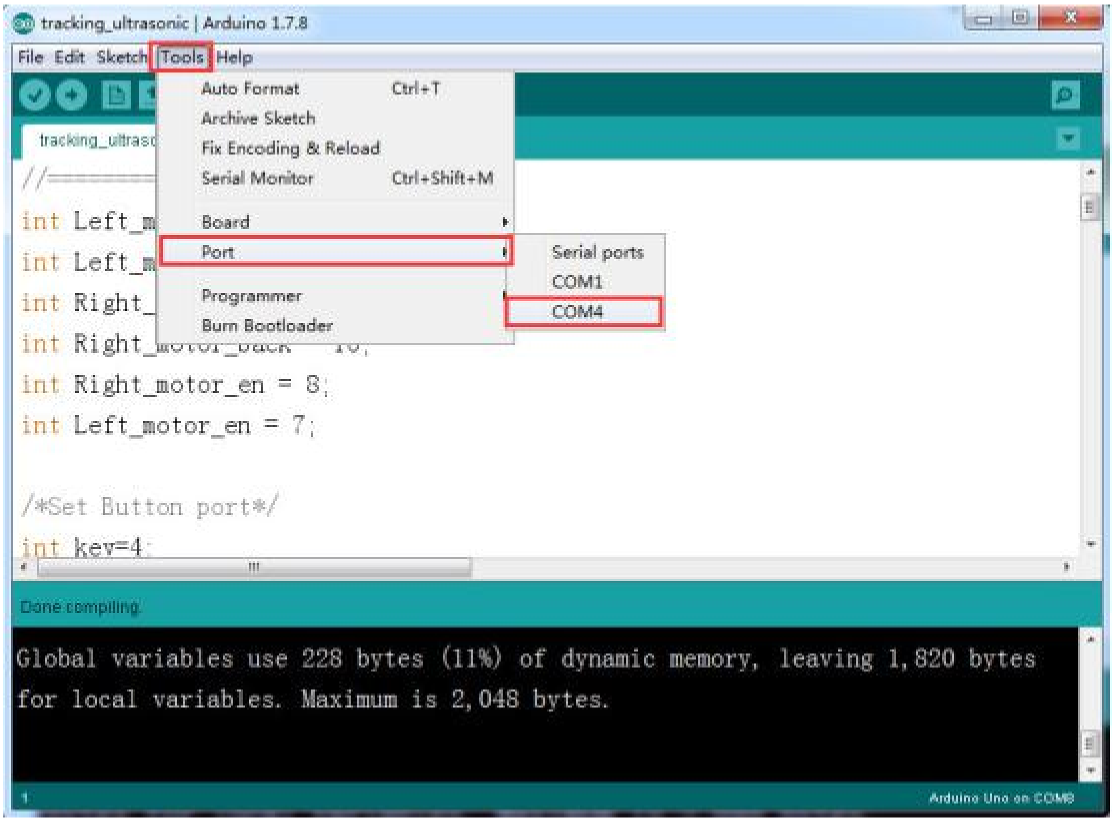
```
//============================
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;

/*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.





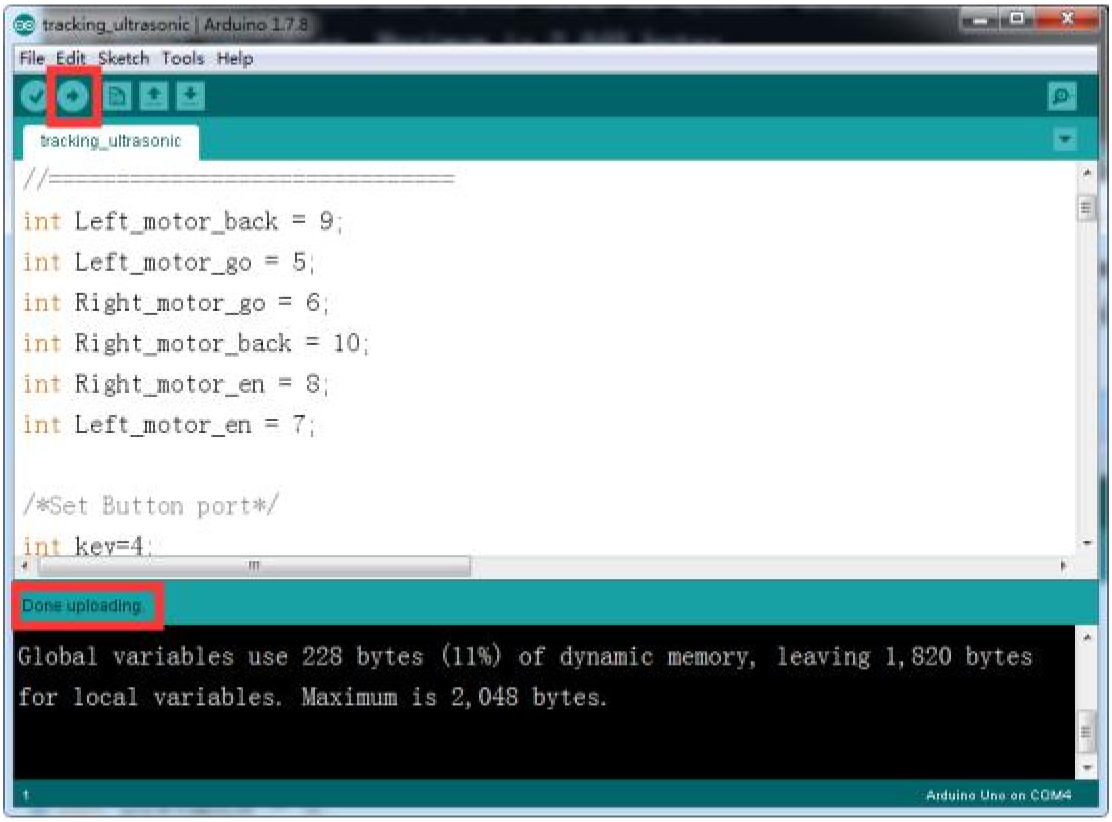
```
//==========
int Left_m
int Left_m
int Right_
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;

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

Done compiling.

```
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.



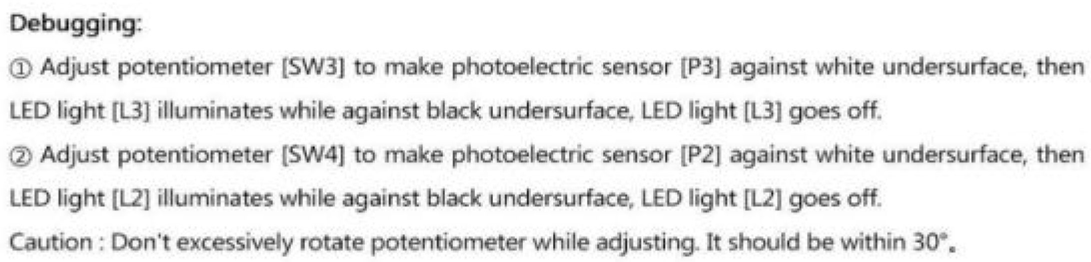
tracking_ultrasonic | Arduino 1.7.8
File Edit Sketch Tools Help
tracking_ultrasonic

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

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 uploading.

```
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.

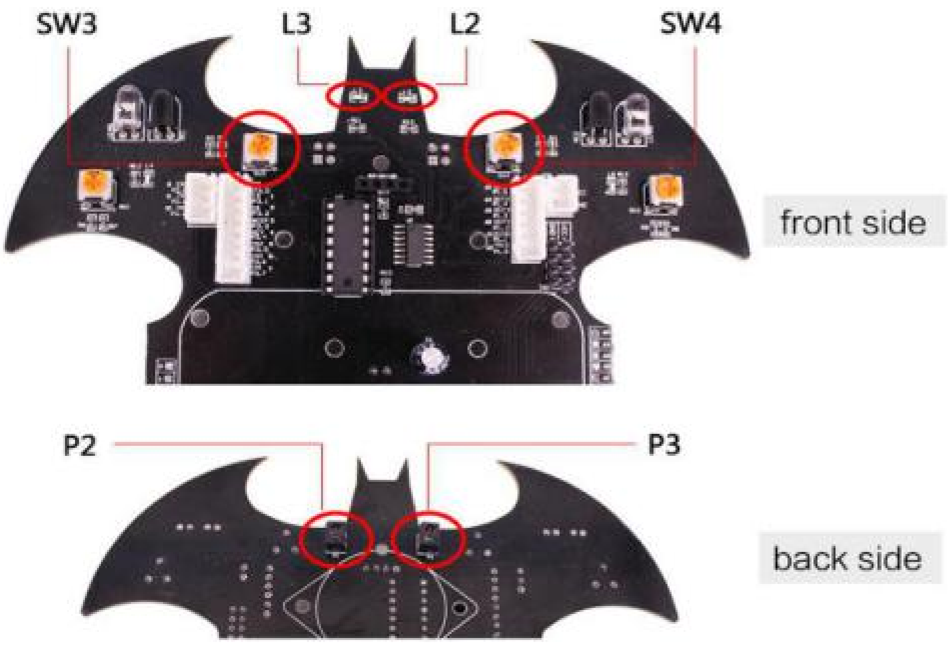


Debugging:

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.