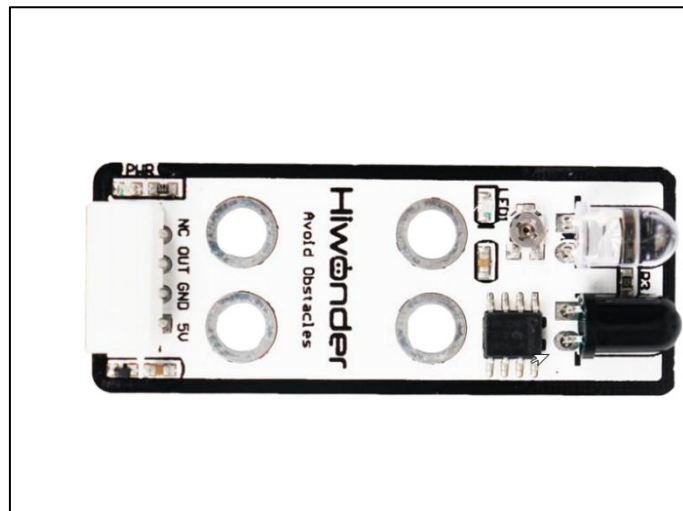


## Lesson 2 Dual Infrared Detection and Sorting

### 1. Working Principle

Infrared obstacle avoidance is a photoelectric sensor integrating IR transmitter and IR receiver. Featuring long detection distance and low interference from visible light , it is widely used in robot and assembly line piecework, etc.

This sensor detects obstacle by transmitting and receiving infrared. When the infrared transmitted by the sensor meets the obstacle ahead, the infrared will be reflected to the receiving terminal.



The path of the program file: "6.Secondary Development/Sensor-extension Game/Arduino Development/ Program Files/Dual Infrared Detection and Sorting/ InfraredSensor\_sorting/ InfraredSensor\_sorting.ino"

```
40 if (sensor_left == 0.0) {  
41     Serial.println("infrared left");  
42     setBuzzer(100);  
43     pos[0] = 70;  
44     pos[1] = -165;  
45     pos[2] = 120;  
46     set_position(pos, 1500);  
47     delay(1500);  
48     pos[0] = 70;  
49     pos[1] = -165;  
50     pos[2] = 86;  
51     set_position(pos, 800);  
52     Pump_on();  
53     delay(1000);  
54     pos[0] = 70;  
55     pos[1] = -165;  
56     pos[2] = 200;  
57     set_position(pos, 1000);  
58     delay(1000);  
59     pos[0] = 150;  
60     pos[1] = -35;  
61     pos[2] = 200;  
62     set_position(pos, 800);  
63     delay(800);  
64     SetPWMServo(1, 1800, 500);
```

When the signal is detected by infrared sensors, it will travel to microcontroller for processing. The closer the obstacle is, the stronger the reflection intensity; the farther the obstacle is, the weaker the reflection intensity. Different surface color has different reflection intensity. White is the strongest and black is the weakest.

Then, the object is detected by the infrared sensor and buzzer will make sound. MaxArm will perform the corresponding action.

Finally, execute the function for controlling action, buzzer, air pump to pick up and place the object.

---

**Note:** For better recognition effect, the sensitive of infrared sensor will be adjusted high, but also more susceptible to the impact of objective factors such as ambient light. If you need to adjust the sensitivity, you can refer to the content in "6.

**Sensitivity Adjustment"** in this lesson.

---

## 2. Preparation


### 2.1 Hardware

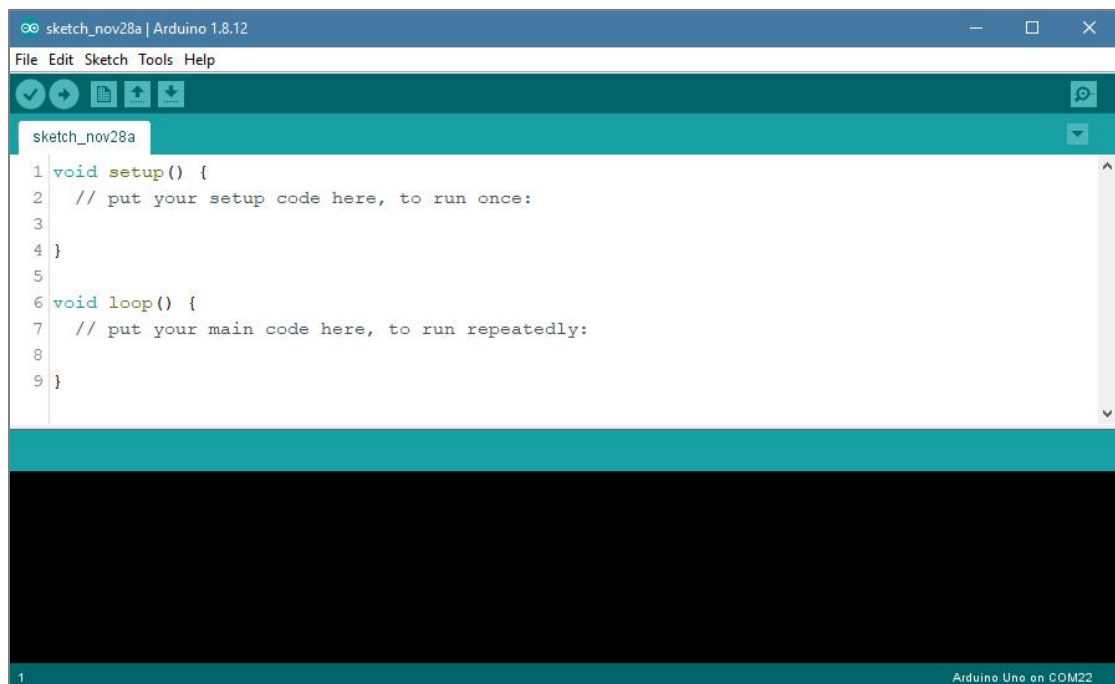
Please refer to “Lesson 1 Sensor Assembly ”to assemble infrared sensors to the corresponding position on MaxArm.

### 2.2 Software

Please connect MaxArm to Arduino editor according to the tutorial in folder “4. Underlying Program Learning/Arduino Development/Lesson 1 Set Development Environment”.

## 3. Program Download

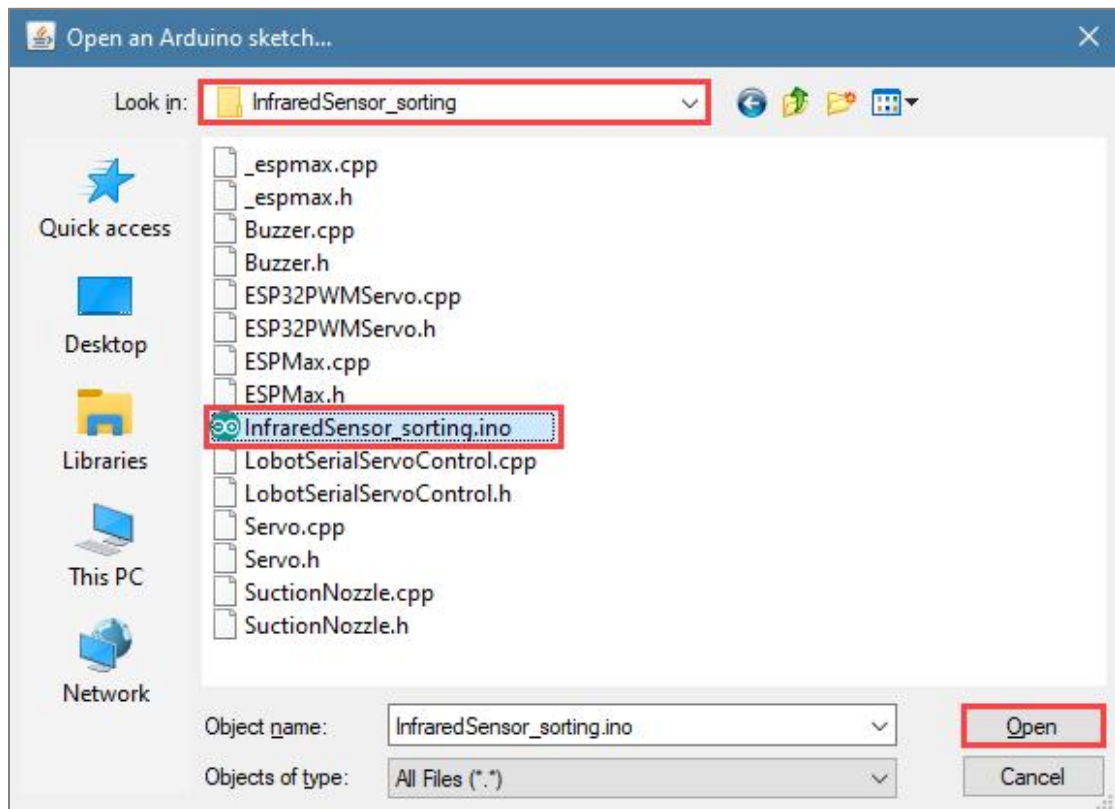
- 1) Click on  icon to open Arduino IDE.



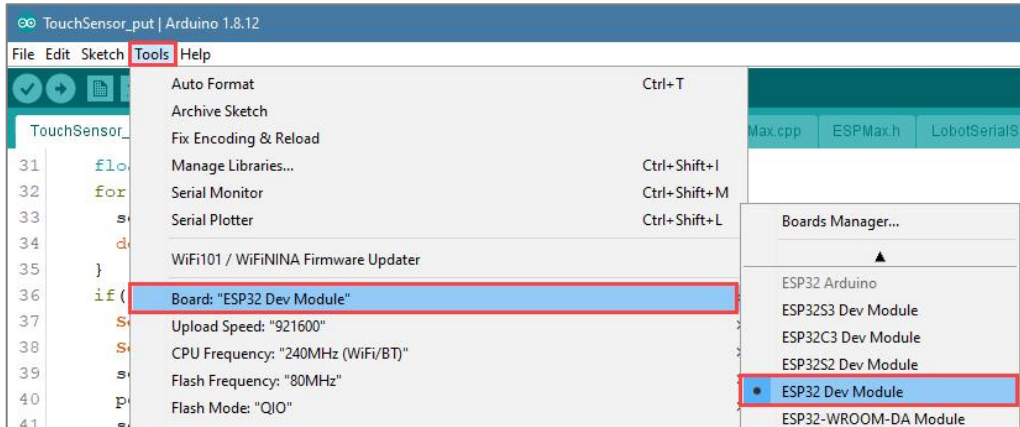
- 2) Click “File->Open” in turn.



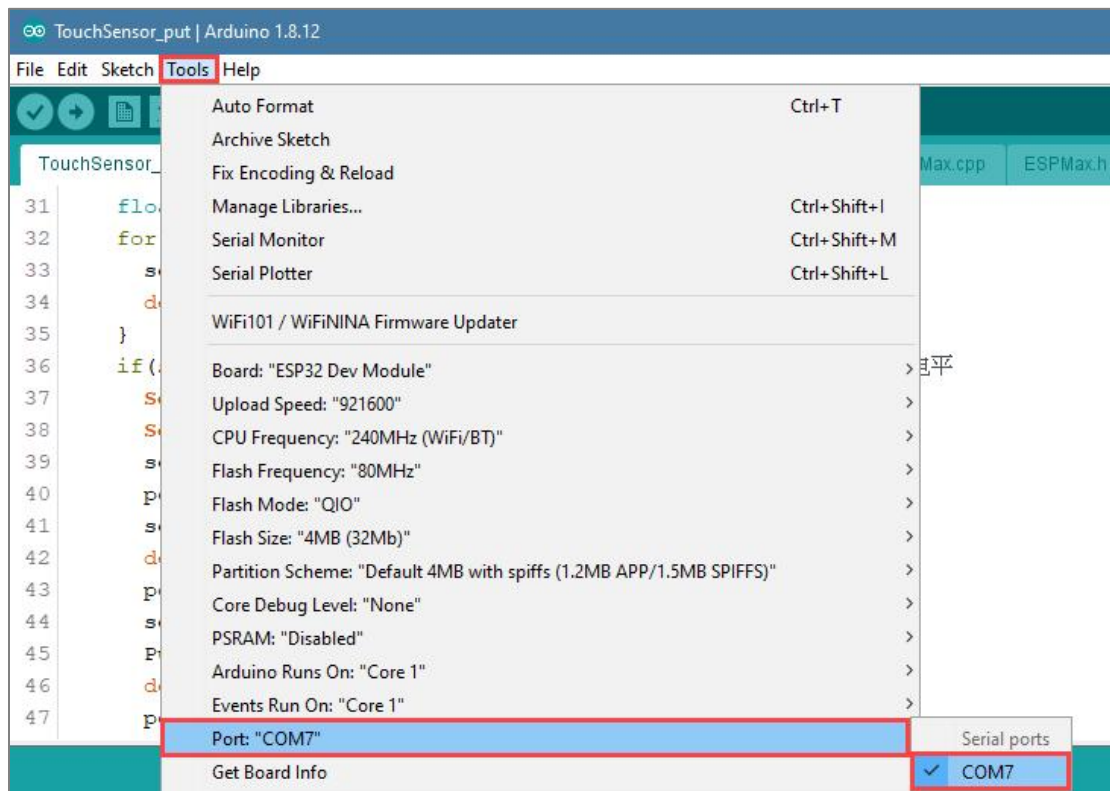
- 3) Open the program “InfraredSensor\_sorting.ino” in the folder “6.Secondary Development / Arduino Development/Sensor-extension Game/ Program Files/Dual Infrared Detection and Sorting/InfraredSensor\_sorting.”



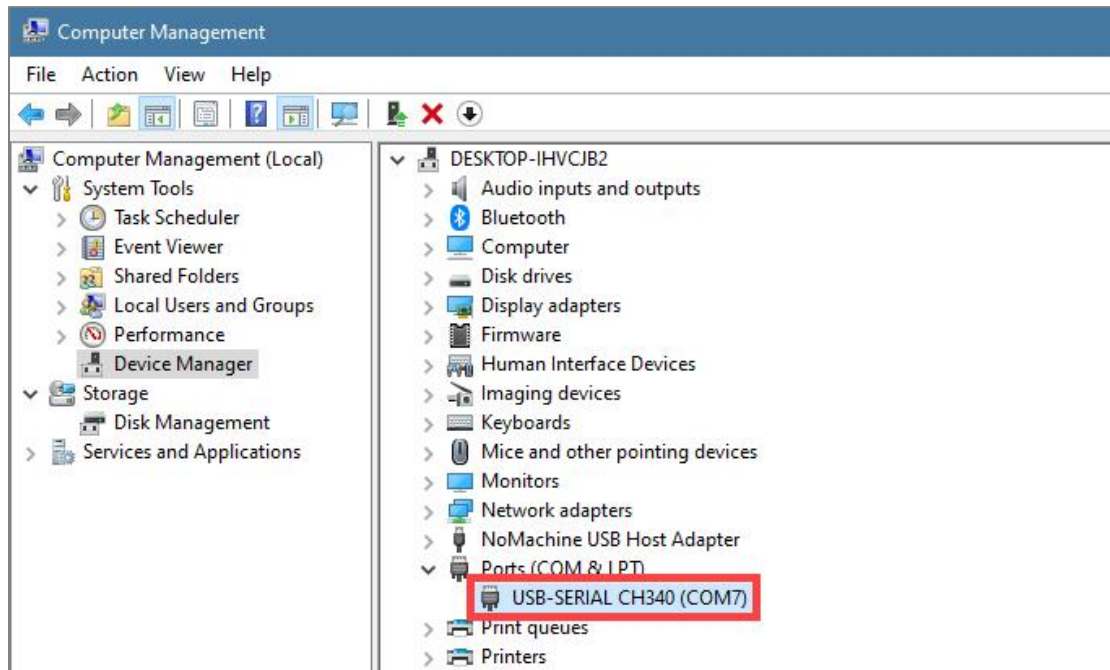
- 4) Select the model of the development board. Click “Tools-> Board” and select “ESP 32 Dev Module” (If the model of the development board has been configured when setting the development environment, you can skip this step).



- 5) Select the corresponding port of Arduino controller in “Tools->Port”. (Here take the port “COM5” as example. Please select the port based on your computer. If COM1 appears, please do not select because it is the system communication port but not the actual port of the development port.)




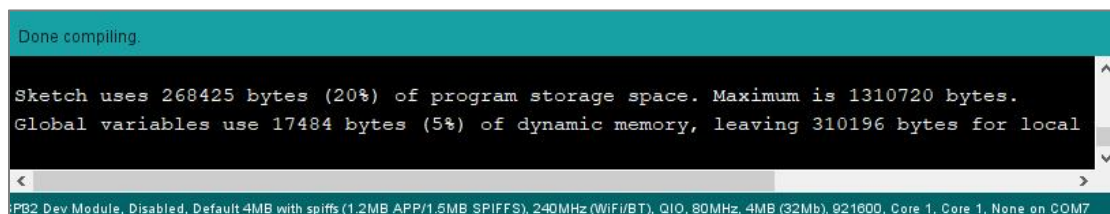
- 6) If you're not sure about the port number, please open the “This PC” and click “Properties->Device Manager” in turns to check the corresponding port number (the device is with CH340). Then select the correct port on Arduino editor.




- 7) After selecting, confirm the board “ESP32 Dev Module” in the lower right corner and the port number “COM5” (it is an example here, please refer to the actual situation).

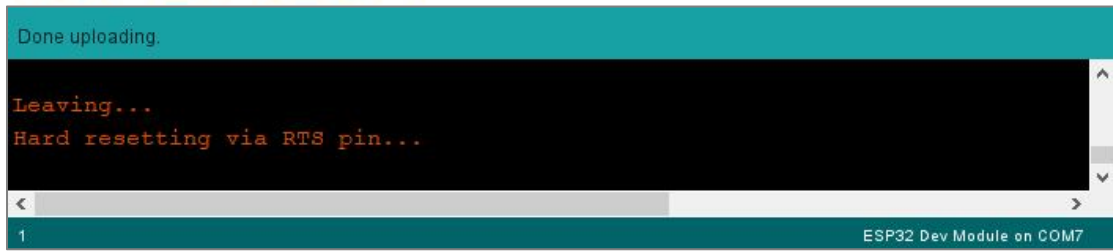


- 8) Then click on  icon to verify the program. If no error, the status area will display “Compiling->Compile complete” in turn. After compiling, the information such as the current used bytes, and occupied program storage space will be displayed.



- 9) After compiling, click on  icon to upload the program to the development board. The status area will display “Compiling->Uploading->Complete” in turn. After uploading, the status area will stop printing the uploading information.





## 4. Project Outcome

When the block is detected, MaxArm will rotate to the corresponding position and turn on air pump to pick up the block, and then place it to placement area in the same side. After the whole process is done, turn off the air pump.

---

Note: If there are blocks placed in front of the two infrared sensors, according to the program setting, MaxArm will first pick up the color block on the left side (Take robot arm as the first person view).

---

## 5. Program Instruction

### 5..1 Import function library and Initialize

The path of the program file: "6. Secondary Development/ Arduino Development/ Program Files/ Dual Infrared Detection and Sorting/ InfraredSensor\_sorting/InfraredSensor\_sorting.ino". If the program is modified, you can find a backup file in Appendix.

Before running the program, the buzzer, kinematics, kinematics encapsulation library, PWM servo, suction nozzle and other related library files need to be imported first.

```
1 #include "Buzzer.h"
2 #include "ESPMaX.h"
3 #include "_espmax.h"
4 #include "ESP32PWMServo.h"
5 #include "SuctionNozzle.h"
6 #include "LobotSerialServoControl.h"
```

Then, initialize the library file and the robotic arm.

```
10 #define infrared_left 23
11 #define infrared_right 32
12
13 void setup() {
14
15     Buzzer_init();
16     ESPMax_init();
17     Nozzle_init();
18     PWMServo_init();
19     pinMode(infrared_left, INPUT_PULLUP);
20     pinMode(infrared_right, INPUT_PULLUP);
21     Serial.begin(115200);
22     Serial.println("start...");
23     setBuzzer(100);
24     go_home(2000);
25     SetPWMServo(1, 1500, 2000);
26 }
```

## 5.2 Infrared Detection

Use the digitalRead() function to read the detected value and use the for() function to detect several times to get accurate result. Take the left infrared sensor as example. When object is detected by the left infrared sensor, infrared\_left is 0, otherwise, it is 1. Then the detected value plus sensor\_left is assigned to sensor\_left. If the sensor\_left is equal to 0.0 after five rounds, the object is detected, otherwise, no object is detected.



```

28 void loop() {
29     float pos[3];
30
31     float sensor_left = 0.0;
32     float sensor_right = 0.0;
33
34     for (int i = 0; i < 5; i++) {
35         sensor_left += digitalRead(infrared_left);
36         sensor_right += digitalRead(infrared_right);
37         delay(50);
38     }

```

### 5.3 Infrared Detection Feedback

When an object is detected by infrared sensor, MaxArm will execute the corresponding action. Here take the left infrared sensor for example.

```

40 if (sensor_left == 0.0) {
41     Serial.println("infrared_left");
42     setBuzzer(100);
43     pos[0] = 70;
44     pos[1] = -165;
45     pos[2] = 120;
46     set_position(pos, 1500);
47     delay(1500);
48     pos[0] = 70;
49     pos[1] = -165;
50     pos[2] = 86;
51     set_position(pos, 800);
52     Pump_on();
53     delay(1000);
54     pos[0] = 70;
55     pos[1] = -165;
56     pos[2] = 200;
57     set_position(pos, 1000);
58     delay(1000);
59     pos[0] = 150;
60     pos[1] = -20;
61     pos[2] = 200;

```

```

62  set_position(pos, 800);
63  delay(800);
64  SetPWMServo(1, 1800, 500);
65  delay(200);
66  pos[0] = 150;
67  pos[1] = -20;
68  pos[2] = 90;
69  set_position(pos, 800);
70  delay(800);
71  pos[0] = 150;
72  pos[1] = 10;
73  pos[2] = 88;
74  set_position(pos, 500);
75  delay(500);
76  Valve_on();
77  pos[0] = 150;
78  pos[1] = 10;
79  pos[2] = 200;
80  set_position(pos, 1000);
81  delay(1000);
82  Valve_off();
83  go_home(1500);
84  delay(200);
85  SetPWMServo(1, 1500, 500);
86  delay(1500);

```

Take setBuzzer() function, set\_position() function, Pump\_on() function and go\_home() for example.

The setBuzzer() is a function to control buzzer. Call "Buzzer.h" function in the same directory as "InfraredSensor\_sorting.ino" program. Fill in the parentheses with the duration time of buzzer, and the unit is ms. The code "setBuzzer(100)" means that the buzzer will respond for 100ms after the infrared sensor detects an object, and then the next set\_position() function will be executed.

```

42  setBuzzer(100);

```

The set\_position() is a function to call robotic arm. Call "ESPMax.h" function in the same folder as "InfraredSensor\_sorting.ino" program. Fill in the parentheses with the coordinate values and duration time. Take the code "set\_position(pos,1500)" for example, pos[0], pos[1], pos[2] is the representative of the robot arm corresponding to the position on the XYZ axis,

1500 is the running time, the unit is milliseconds (ms). After that, the Pump\_on() function is executed in the next step.

```
43 pos[0] = 70;  
44 pos[1] = -165;  
45 pos[2] = 120;  
46 set_position(pos, 1500);
```

The Pump\_on() is a function to control air pump. Pump\_on () function is to control the air pump on; Valve\_on () function is used to turn off air pump and open solenoid valve; Valve\_off () function to turn off the solenoid valve. Call "InfraredSensor\_sorting.ino" function in the same folder as the "SuctionNozzle.h" function, after the implementation of the next go\_home () function.

```
52 Pump_on();
```

```
76 Valve_on();
```

```
82 Valve_off();
```

The go\_home() is a function to reset robotic arm. Cal "ESPMax.h" function in the same folder as "InfraredSensor\_sorting.ino" function.Fill in the parentheses with the action duration time, and the unit is ms. Take the code "go\_home(1500)" for example. The parameter "1500" is the time it takes to complete the reset action. When this function is finished, the program will execute 1500ms delay function and then end this recognition and sorting actions and wait for the next recognition signal.

```
83 go_home(1500);
```

## 6. Adjust Sensitivity

There is an adjustable potentiometer knob on infrared obstacle avoidance sensor for adjusting the measuring distance. When performing the related games, if the measurement effect is not good enough, the measurement sensitivity of sensor can be adjusted by adjusting the knob.

It is recommended to use phillips screwdriver. Rotate the knob clockwise, as the figure shown below, to increase the infrared emission intensity and the measurement distance; rotate it counterclockwise to weaken the infrared emission intensity and decrease the measurement distance.



Please take notice of the following two tips:

- 1) Since the warm light (incandescent light, sunlight, etc) contains more infrared component, there will be a certain interference between infrared to affect the detection outcome. When using it, special attention should be paid to the surrounding environment. Therefore, you can not adjust the sensitivity alone.
- 2) The sensitivity adjustment has is set a threshold, which you can regard it as

a critical point. If the sensitivity exceeds the critical point, its value will return to the initial state.

3) Sensitivity adjustment needs to be based on the actual needs of the project.

It is better to be adjusted to the most suitable conditions.