ZHIYI智譯

Four-way tracking smart car

ZYC0043

V2.02.305.03

Introduction to the tutorial

This tutorial includes the file content shown in the figure below to help you learn more about the car kit and programming knowledge:

"1_Get_start": This folder stores the robot assembly guide and necessary software environment files, etc. In order to complete the assembly accurately and quickly, please be sure to review it in detail and assemble it according to the manual. At the same time, the premise of realizing the program function is to create the software environment correctly, please check the PDF file under this folder.

"2_Arduino_Code": This folder is used to store Arduino code files, and each code file is uploaded and used independently;

"3_References": This folder stores some reference materials of modules such as sensors;

After completing the assembly of the car and the creation of the environment, follow this tutorial to gradually complete the program burning of the smart car and realize different functions!

Table of contents

Tutorial Introduction 1

1. Let the car move freely 5

1.1 Description 5

1.2 Burn code 5

1.3 Code Analysis 8

1.4 How does the car move forward or turn? 10

2. Follow mode 13

2.1 Description 13

2.2 Upload code 13

2.3 Code Analysis 14

2.4 The car does not follow the movement? 20

3. Obstacle avoidance mode 23

3.1 Description 23

3.2 Upload code 23

3.3 Code Analysis 24

4. Tracking mode 28

4.1 Description 28

4.2 Upload code 28

4.3 Code Analysis 29

1. Let the car move freely

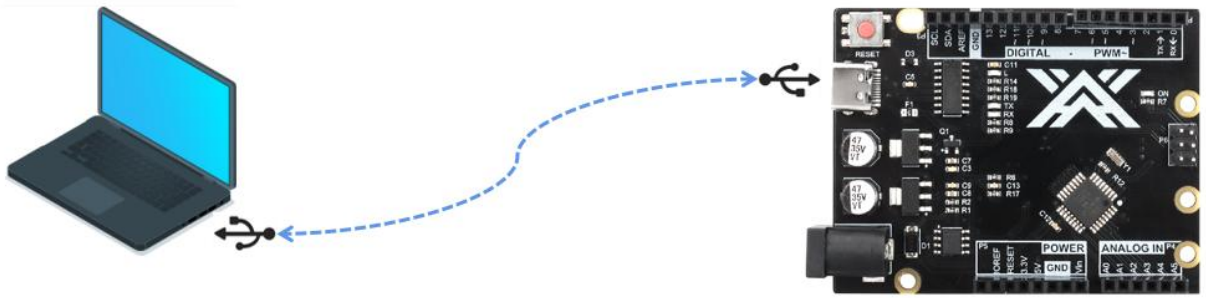
1.1 Description

This section mainly learns and understands how to make the smart car move to achieve forward, backward and turning actions.

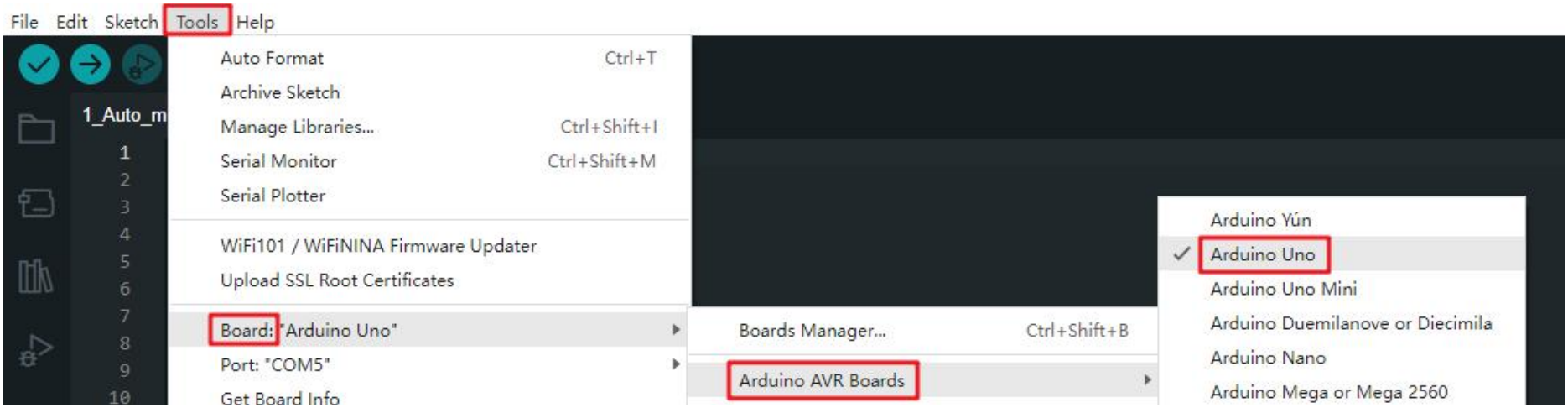
1.2 Burn code

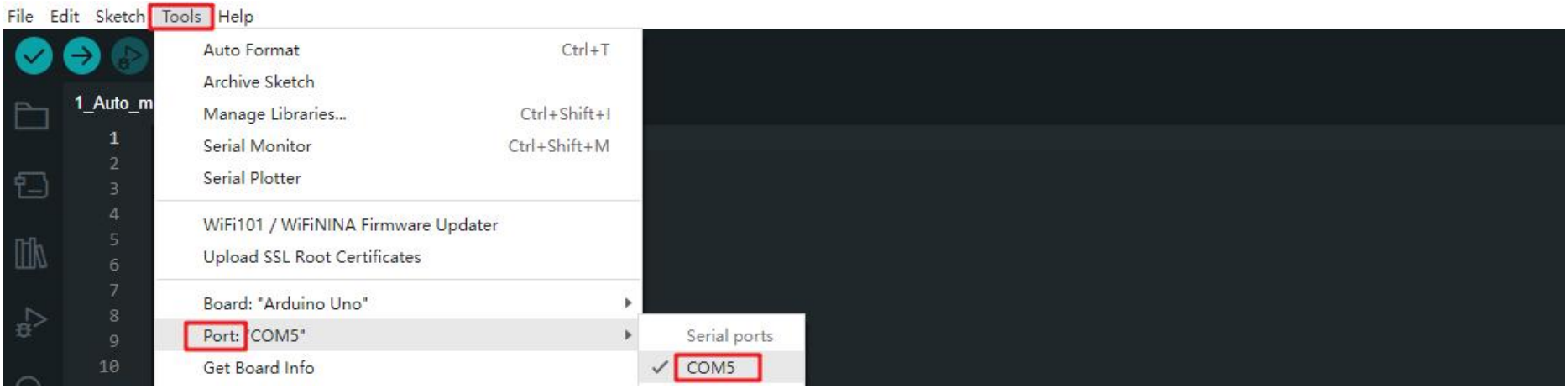
Open the code file (path: 2_Arduino_Code\1_Auto_move\1_Auto_move.ino)

Connect the main control board to the computer with a USB cable



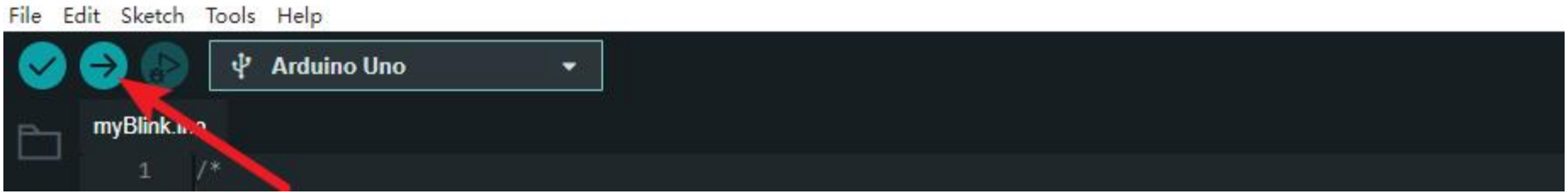
Select the board type as UNO and the serial port number as COM5



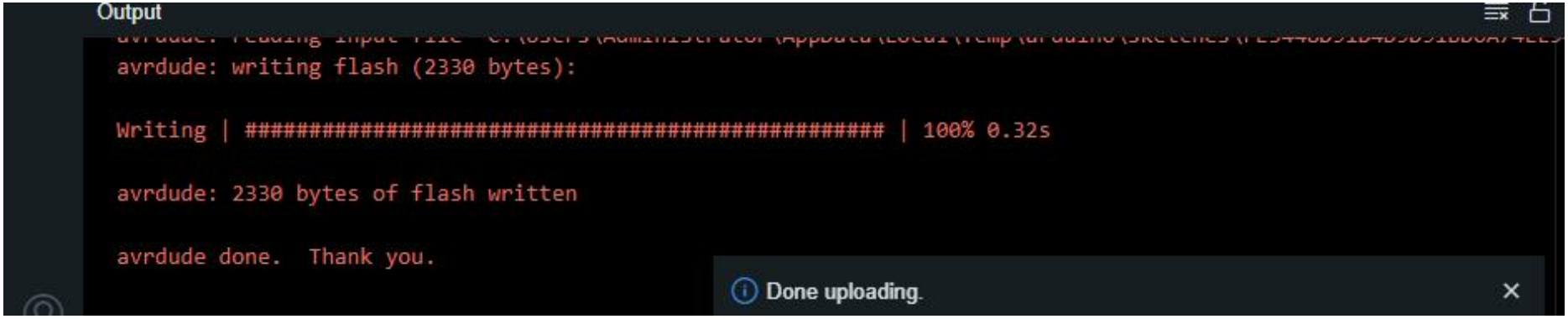


Note: The board type here is UNO and the serial port is COM5 . Actually everyone displays the serial port differently, although COM 5 is selected here , it could be COM3 or COM4 on your computer.

After clicking the "Upload" button, the program starts to upload.



After the upload is successful, it prompts "Done uploading".



Install 2 pieces of 18650 batteries in the battery box, turn on the switch, you can see the wheels of the car move forward for one second and backward for one second and repeat continuously.

1.3 Code Analysis

Define motor and pwm pins

```
1  #define ENA 5
2  #define ENB 6
3  #define IN1 3
4  #define IN2 4
5  #define IN3 2
6  #define IN4 7
```

Define the motor speed, you can modify the value by yourself (130~220 is more suitable) to get different moving speeds

```
8  #define carSpeed 130//Set the carSpeed to 130
```

For example, the forward() function will advance at the speed you set

```
10  void forward(){//forward function
11    analogWrite(ENA, carSpeed);//Set the speed of ENA
12    analogWrite(ENB, carSpeed);//Set the speed of ENB
13    digitalWrite(IN1, LOW);
14    digitalWrite(IN2, HIGH);
15    digitalWrite(IN3, HIGH);
16    digitalWrite(IN4, LOW);
17    Serial.println("Forward");
18  }
```

Of course, you can also enter the value directly

```
30  void left() {//left function
31    analogWrite(ENA, 200);//Set the speed of ENA
```

The setup function defines the baud rate as 9600 and the pin as output

```
57  void setup() {
58    Serial.begin(9600);
59    pinMode(IN1, OUTPUT);
60    pinMode(IN2, OUTPUT);
61    pinMode(IN3, OUTPUT);
62    pinMode(IN4, OUTPUT);
63    pinMode(ENA, OUTPUT);
64    pinMode(ENB, OUTPUT);
65    stop();
66  }
```

The loop function executes forward and backward for one second each. Of course, you can also change 1000 to 2000 to see

how it works.

```
68 void loop(){
69   forward();
70   delay(1000);
71   back();
72   delay(1000);
73   // left();
74   // delay(1000);
75   // right();
76   // delay(1000);
77 }
```

If you remove the "//" used to comment in front of the left() and right() functions, the car will not only move forward and backward, but also turn left and right.

```
68 void loop(){
69   forward();
70   delay(1000);
71   back();
72   delay(1000);
73   left();
74   delay(1000);
75   right();
76   delay(1000);
```

1.4 How does the car move forward or turn?

Each moving state of the car corresponds to a function, for example, moving forward corresponds to the forward() function,

and turning left corresponds to the left() function

```
10  void forward(){//forward function
11    analogWrite(ENA, carSpeed);//Set the speed of ENA
12    analogWrite(ENB, carSpeed);//Set the speed of ENB
13    digitalWrite(IN1, LOW);
14    digitalWrite(IN2, HIGH);
15    digitalWrite(IN3, HIGH);
16    digitalWrite(IN4, LOW);
17    Serial.println("Forward");
18  }
```

```
30  void left() {//left function
31    analogWrite(ENA, 200);//Set the speed of ENA
32    analogWrite(ENB, 200);//Set the speed of ENB
33    digitalWrite(IN1, LOW);
34    digitalWrite(IN2, HIGH);
35    digitalWrite(IN3, LOW);
36    digitalWrite(IN4, HIGH);
37    Serial.println("Left");
38  }
```

ENA and ENB control the pwm of the car, which is the moving speed;

The high and low levels of IN1/IN2/IN3/IN4 control the rotation direction of the car wheels.

When the car moves forward, the four wheels rotate forward, and IN1~IN4 are LOW/HIGH/HIGH/LOW respectively.

```
10  void forward(){//forward function
11    analogWrite(ENA, carSpeed);//Set the speed of ENA
12    analogWrite(ENB, carSpeed);//Set the speed of ENB
13    digitalWrite(IN1, LOW);
14    digitalWrite(IN2, HIGH);
15    digitalWrite(IN3, HIGH);
16    digitalWrite(IN4, LOW);
17    Serial.println("Forward");
18  }
```

When turning left, the left wheel rotates backward and the right wheel rotates forward. At this time, IN1~IN4 are

LOW/HIGH/LOW/HIGH respectively.

```
30  void left() {//left function
31    analogWrite(ENA, 200);//Set the speed of ENA
32    analogWrite(ENB, 200);//Set the speed of ENB
33    digitalWrite(IN1, LOW);
34    digitalWrite(IN2, HIGH);
35    digitalWrite(IN3, LOW);
36    digitalWrite(IN4, HIGH);
37    Serial.println("Left");
38  }
```

The entire control logic is implemented by the motor driver board L298N. For more detailed content, please refer to the relevant reference materials, folder: 3_Reference

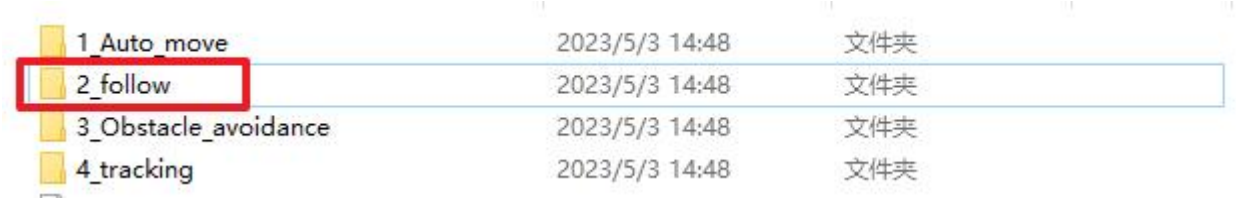
2. follow mode

2.1 Description

In the previous section, we learned how to make the car walk. In this section, we let the car follow the movement through the ultrasonic sensor and the LM393 infrared sensors on both sides .

2.2 Upload code

Open the code file (path: 2_Arduino_Code\2_follow\2_follow.ino)



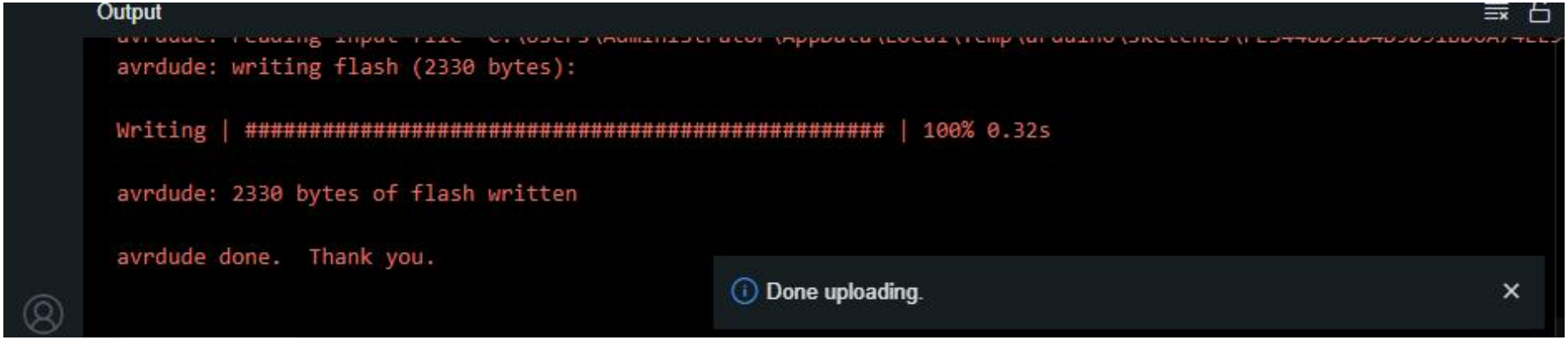
Similar to the operation method in the previous section, to successfully upload the program code, you need to perform several steps:

1) Connect the main control board to the computer with a USB cable;

2) In the IDE software, select the board type as UNO and correctly display the COM serial port number;

3) After clicking the "Upload" button, the program starts to upload;

4) After the upload is successful, it prompts "Done uploading".



5) Turn on the power switch, the car will move forward when there is an object at a certain distance in front, and the car will retreat when the distance is too close.

2.3 Code Analysis

In the previous section of the code, we have written the program code for the car to move. This section of the code is based on the previous section with the addition of ultrasonic, servo motor and infrared detection modules on both sides.

The servo library used by the servo motor has been added in the preparation work at the beginning of the tutorial (see the folder "1_Get_start"), just program an imported code here, and initialize the angle of the servo motor in the settings to be about the middle value 100 (the range is 0~180°).

```
7   #include <Servo.h>
```

```
30  void setup()
31  {
32      myservo.attach(A0);  // attaches the servo on pin 9 to the servo object
33      myservo.write(100);
```

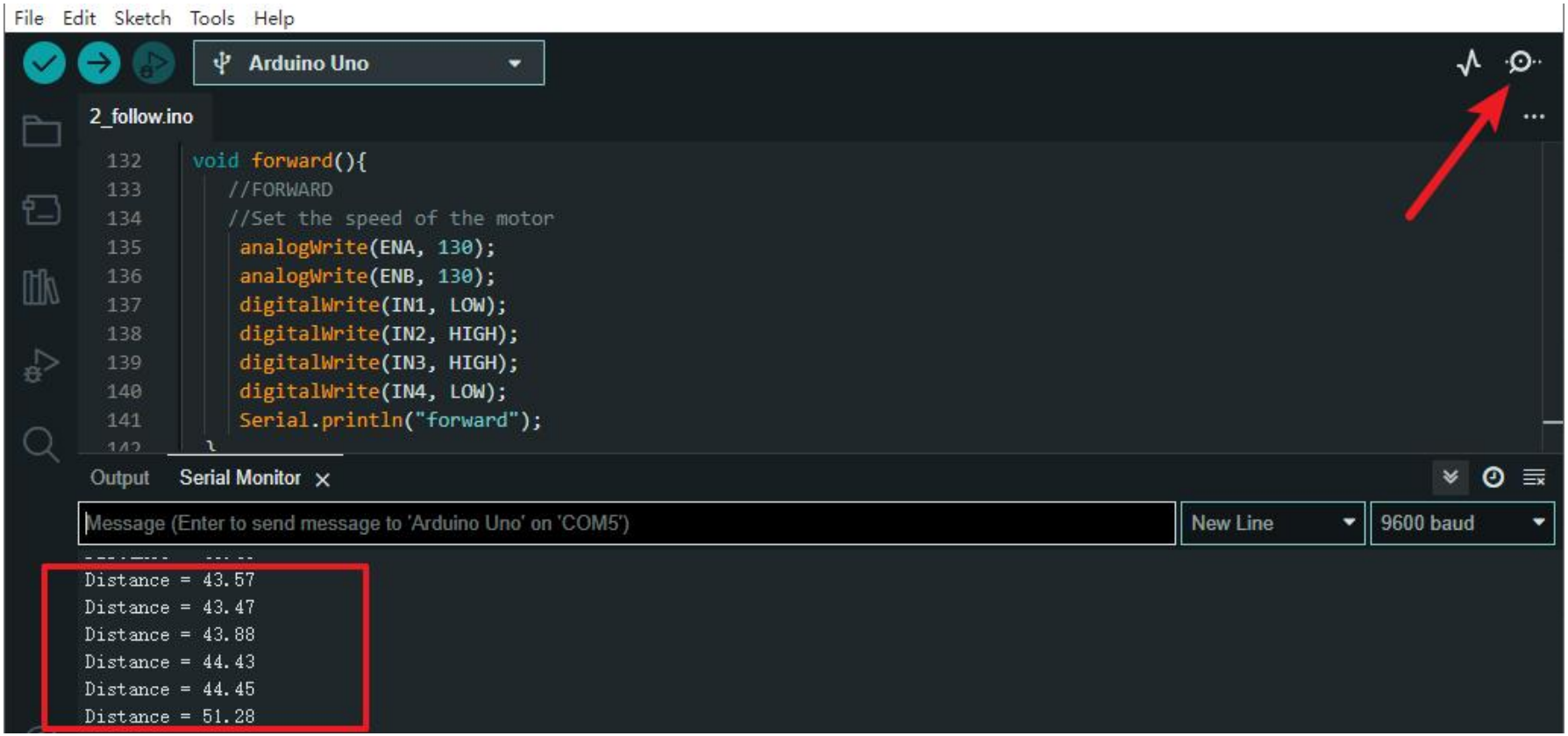
Next is to define the pin used by the ultrasonic wave and the detection distance variable cm

```
 8  #define Trig 12 //Pin Tring connects to D12
 9  #define Echo 13 //Pin Echo connects to D13
10  float cm; //Distance variable
```

Add a function to get the ultrasonic detection distance

```
94  float GetDistance()
95  {
96      float distance;
97      // Send a low short pulse to Trig to trigger the ranging
98      digitalWrite(Trig, LOW); //Send a low level to Trig
99      delayMicroseconds(2);
100     digitalWrite(Trig, HIGH);
101     delayMicroseconds(10);
102     digitalWrite(Trig, LOW);
103     distance = pulseIn(Echo, HIGH) / 58.00;
104     Serial.print("Distance = ");
105     Serial.println(distance);//The serial output distance is converted into cm
106     return distance;
107 }
```

The GetDistance() function includes a printout method Serial.print(). After uploading the code and opening the **serial monitor,** you can see the real-time printed distance between the ultrasonic wave and the object.



To achieve left and right following, define the pins and variables connected to the infrared detection modules on both sides

```
21 int Sensor1 = A2;//pin A2
22 int Sensor2 = A5;//pin A5
23 int SensorLeft;
24 int SensorRight;
```

Define two sensors as input

```
42      pinMode(Sensor1, INPUT);
43      pinMode(Sensor2, INPUT);
```

In the loop function loop(), save the values read by the ultrasonic and infrared detection modules on both sides into variables.

```
51      cm = GetDistance();
52      SensorLeft  =  digitalRead(A2);//The sensor on the left
53      SensorRight =  digitalRead(A5);//The sensor on the Right
```

The function of following or retreating is realized by comparing the ultrasonic distance with the values of the infrared detection modules on both sides.

```
54      if(cm >= 0 && cm <= 5)
55      {
56          back();
57      }else if((cm > 5 && cm <= 10) || cm > 30)
58      {
59          stop();
60      }else if(cm > 10 && cm <= 30)
61      {
62          forward();
63      }
64      if (SensorLeft == LOW)
65      {
66          left();
67      }
68      else if (SensorRight == LOW)
69      {
70          right();
```

If the distance of the obstacle detected by the ultrasonic wave is between 0 cm and 5 cm, it will move backwards; otherwise, if the distance detected by the ultrasonic wave is between 5 cm and 10 cm or greater than 30 cm, it will stop; otherwise, it will move forward if the distance is between 10 and 30 cm.

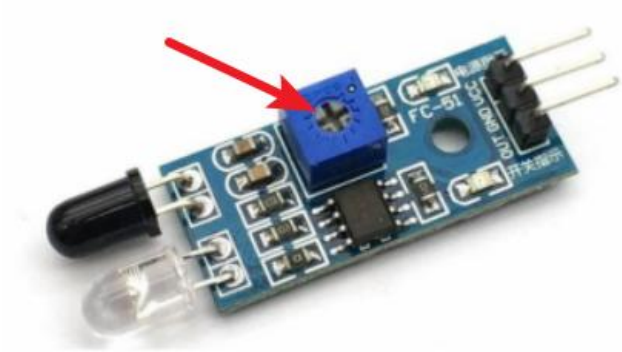
When the infrared sensors on both sides detect objects, the car will follow and turn

```
64      if (SensorLeft == LOW)
65      {
66          left();
67      }
68      else if (SensorRight == LOW)
69      {
70          right();
71      }
```

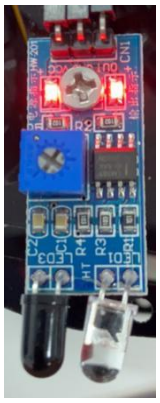
2.4 The car does not follow the movement?

Sometimes after uploading the program, it is found that the car does not follow the movement well. We first check whether the ultrasonic can detect the distance. Open the serial port manager and see that the printed distance is always Distance = 0 , which means that the ultrasonic wiring is wrong. If the distance can be printed normally , it is probably because **the sensitivity of the LM393 infrared detection module** is not well adjusted **.**

Adjustment via potentiometer knob :



We hope that only the power light is on when no object is detected, and the signal light is off (as shown in the lower left figure), and the signal light is on when an object is detected (as in the lower right figure).



If no object is detected, both lights are on. This is because the detection distance is too far, and it needs to be adjusted

counterclockwise to reduce the sensitivity , but it cannot be adjusted too small.

For a more detailed introduction to the ultrasonic and LM393 modules, please refer to the relevant reference materials, folder: 3_Reference

3. Obstacle avoidance mode

3.1 Description

Because the ultrasonic wave can detect the distance of objects and the LM393 module can detect whether there are objects on both sides, the car can not only follow the movement, but also realize the obstacle avoidance function. In this section, we will complete an automatic obstacle avoidance car.

3.2 Upload code

The code we chose to upload this time is: 3_Obstacle_avoidance

Open the code file (path: 2_Arduino_Code\3_Obstacle_avoidance\3_Obstacle_avoidance.ino)

The same as the content of the previous chapters, we complete the upload of the code in sequence

1） Connect the main control board to the computer with a USB cable;

2） In the IDE software, select the board type as UNO and correctly display the COM serial port number;

3） After clicking the "Upload" button, the program starts to upload;

4） After seeing the upload is successful, it prompts "Done uploading".

3.3 Code Analysis

In the previous two sections, we have learned about the use of ultrasonic and LM393 modules, and know how to obtain the distance of ultrasonic: GetDistance() function and the values SensorLeft and SensorRight read by LM393 module. In this section, we will pass these values Program to realize the function of obstacle avoidance.

In the loop function loop(), we save the values read by the LM393 modules on both sides of the car to the variables SensorLeft and SensorRight, and save the obtained ultrasonic distance to the variable middleDistance.

```
109  void loop()
110  {
111      SensorLeft  =  digitalRead(A2);//The sensor on the left
112      SensorRight =  digitalRead(A5);//The sensor on the Right
113
114      middleDistance = GetDistance();//getDistance();
```

Principle of obstacle avoidance

The distance between the ultrasonic detection of the car and the object in front is less than or equal to 30cm (that is, when an obstacle is encountered), the car stops, the servo motor drives the ultrasonic to turn right to obtain the distance of the obstacle on the right, and then turns left to obtain the distance of the obstacle on the left and save it to the variables "rightDistance" and "leftDistance". Then compare the distances, and turn to which side the distance is greater, so as to avoid obstacles.

In the code, first judge whether the distance is less than or equal to 30, the myservo.write() function controls the rotation of the servo motor, because the rotation range of the servo motor is 0~180°, the middle value is set to 100, 20 is to rotate to the right, and 160 is to rotate to the right Turn left, each rotation will perform an ultrasonic acquisition distance function GetDistance(), and save the corresponding distance value.

```
111     if(middleDistance <= 30)
112     {
113         back();
114         delay(50);
115         stop();
116         delay(300);
117         myservo.write(20);
118         delay(500);
119         rightDistance = GetDistance();//getDistance();
120         delay(200);
121         myservo.write(160);
122         delay(500);
123         leftDistance = GetDistance();//getDistance();
124         delay(200);
125         myservo.write(100);
```

Compare the values of the ultrasonic acquisition distances on the left and right sides, and the car turns to the side with the larger distance. The parameter 300 in delay() means that the turn execution time is 300 milliseconds. If you want to turn longer, you can modify a larger value to adjust the turn angle until it is suitable.

```
127         if(rightDistance > leftDistance){
128             right();
129             delay(300);
130         }else if(rightDistance < leftDistance) {
131             left();
132             delay(300)
133         }else if((rightDistance == leftDistance) && (rightDistance < 30)){
134             back();
135             delay(300);
136         }
```

If the sensors on the left and right sides detect obstacles (low level), they will also turn left and right to avoid them, otherwise if they both detect obstacles, they will back off.

```
141     if(!SensorLeft){
142       right();
143     }
144     else if(!SensorRight){
145       left();
146     }else if((!SensorLeft) && (!SensorRight)){
147       back();
148       delay(100);
149     }
150     else{
151         forward();
152     }
```

4. tracking mode

4.1 describe

In this section, we will make the car realize the line tracking function through the 4-way tracking module.

Tips: For the convenience of testing, you may need to stick a black line on the desktop with black tape in advance.with a width of about 1.5cm

4.2 upload code

The code we chose to upload this time is: 4_tracking

Open the code file (path: 2_Arduino_Code\4_tracking\4_tracking.ino)

The same as the content of the previous chapters, we complete the upload of the code in sequence

1) Connect the main control board to the computer with a USB cable ;

2) In the IDE software, select the board type as UNO and correctly display the COM serial port number;

3) After clicking the "Upload" button, the program starts to upload;

4) After seeing the upload is successful, it prompts "Done uploading".

4.3 code analysis

Here we use the four infrared detection sensors of the 4-way tracking module, and determine which sensor is on the black line by reading the high and low level values of each sensor.

Set four variables to store the read values

```
13  int Sensor1;
14  int Sensor2;
15  int Sensor3;
16  int Sensor4;
```

Four sensors correspond to four pin inputs

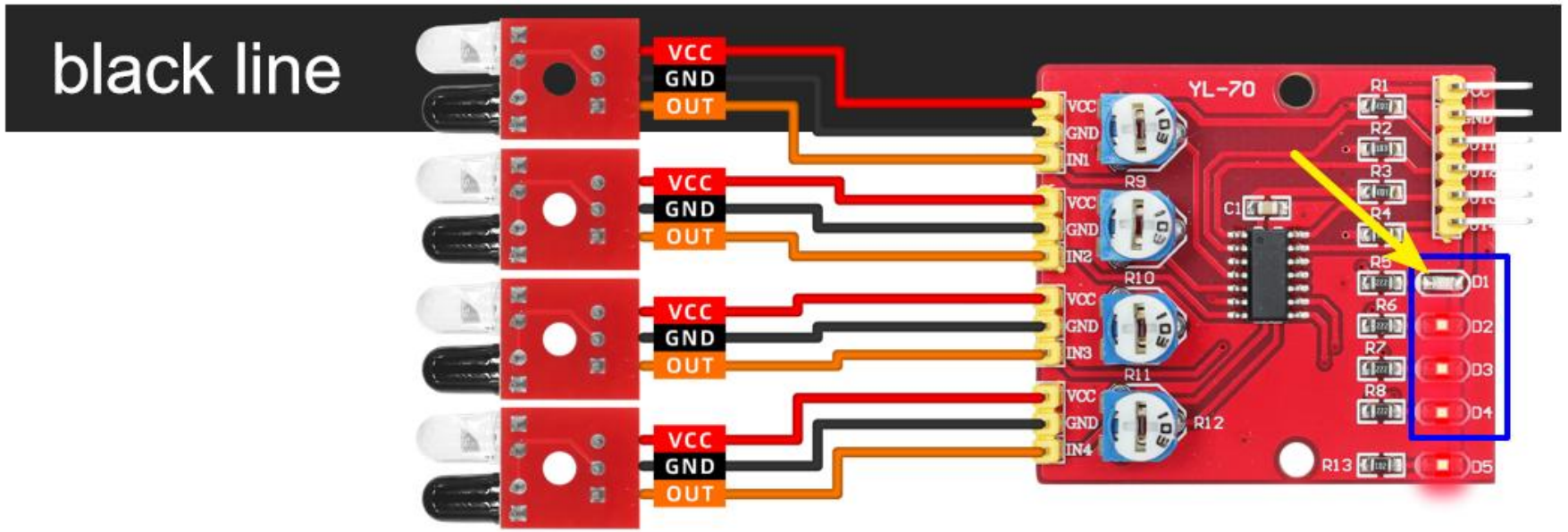
```
28    pinMode(8, INPUT);
29    pinMode(9, INPUT);
30    pinMode(10, INPUT);
31    pinMode(11, INPUT);
```

In the loop function loop(), save the read values of the four sensors into variables

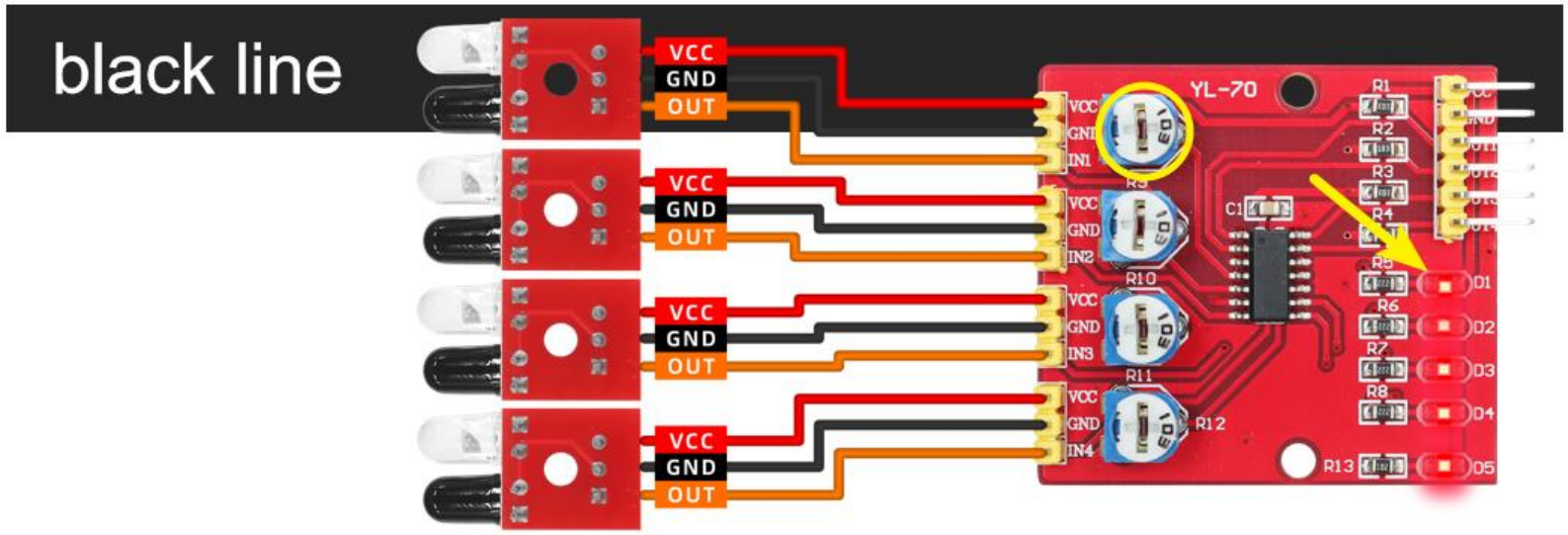
```
38      Sensor1 = digitalRead(8);//IN1
39      Sensor2 = digitalRead(9);//IN2
40      Sensor3 = digitalRead(10);//IN3
41      Sensor4 = digitalRead(11);//IN4
```

After the code is uploaded, we open the serial monitor to check the values read by the four infrared sensors. We can see that the four values are separated by "--" for easy viewing. Normally, the sensor that detects the black line reads a value of 1, and the other The value is 0. When we see the following values, it means that the IN1 (Sensor1) sensor detects the black line, and other sensors do not detect the black line.

As shown in the figure below, after the power is turned on, the power light D5 is on, and the four infrared sensors connected from top to bottom correspond to the four lights respectively D1/D2/D3/D4. When the IN1 (Sensor1) sensor detects a black line, the corresponding The light (D1 light) will be off, and the other lights will be on.



As shown in the figure below, when a black line is detected but the corresponding light is not off.



At this time, you need to use a screwdriver to turn the potentiometer (the circled position), and slowly adjust it to the critical value where the light just goes out. If other lights corresponding to the black line are not detected but off, you need to turn the potentiometer to turn the lights back on.

Use the same method to sequentially detect the black wires of the 4 infrared sensors to adjust the potentiometer to ensure that the lights can be turned off correctly every time the black wires are detected.

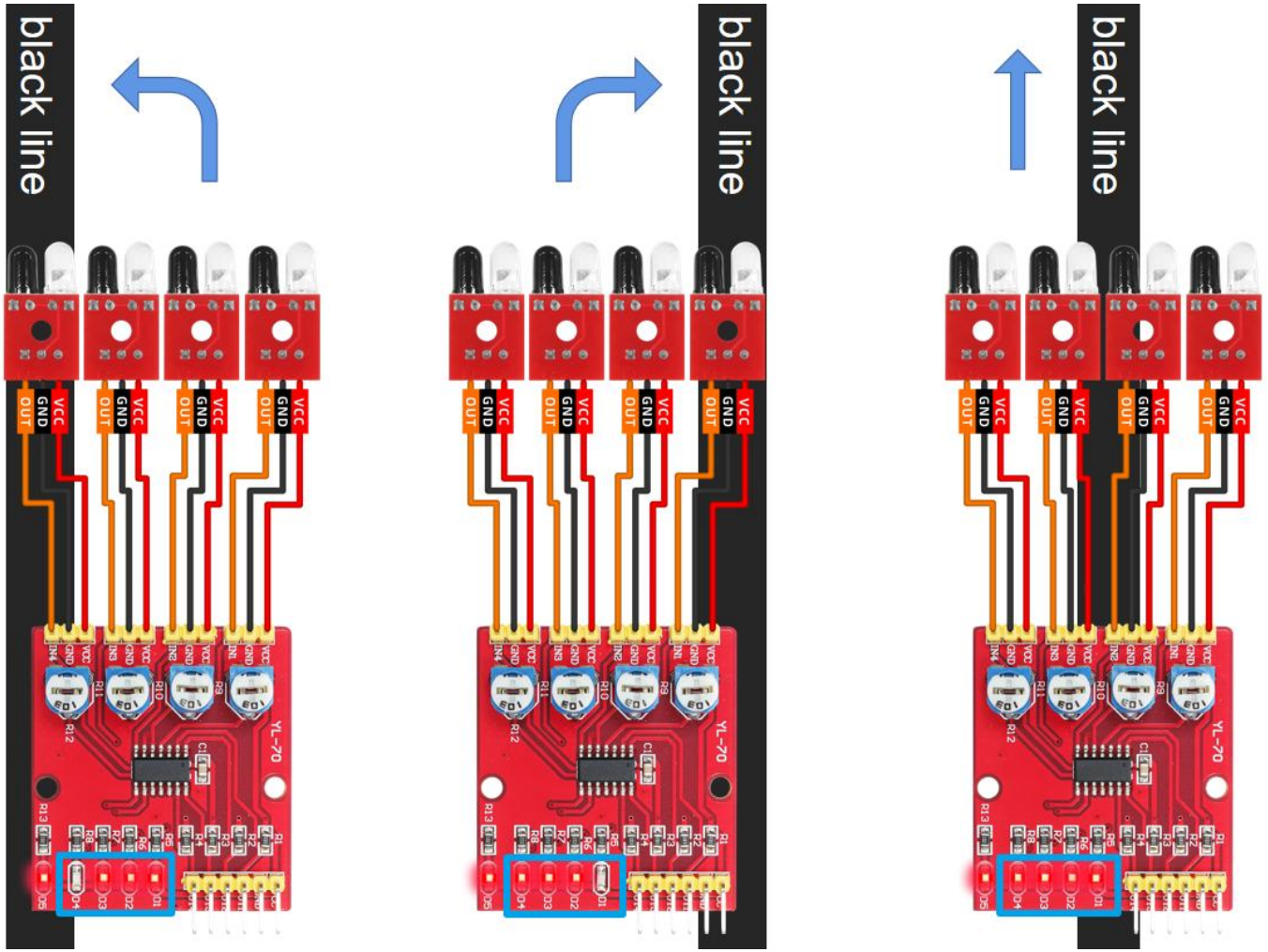
After completing all the sensor tests, you can put the car on the black line (the width of the black line is about 1.2cm and the

width of a black tape). The code to realize the tracking function is as follows:

```
51  if(Sensor4 == LOW && Sensor3 == LOW && Sensor2 == LOW && Sensor1 == LOW){
52    forward();
53  }else if(Sensor4 == HIGH && Sensor3 == HIGH && Sensor2 == HIGH && Sensor1 == HIGH){
54    stop();
55  }else if(Sensor4 == HIGH && Sensor3 == LOW && Sensor2 == LOW && Sensor1 == LOW){
56    left_M();
57  }else if(Sensor4 == LOW && Sensor3 == HIGH && Sensor2 == LOW && Sensor1 == LOW){
58    left();
59  }else if(Sensor4 == LOW && Sensor3 == LOW && Sensor2 == HIGH && Sensor1 == LOW){
60    right();
61  }else if(Sensor4 == LOW && Sensor3 == LOW && Sensor2 == LOW && Sensor1 == HIGH){
62    right_M();
63  }
```

In the code, the high level means that the black line is detected, and the low level means that the black line is not detected. By analyzing the black line detected by the four-way tracking sensor, the trajectory of the car can be determined. When the leftmost sensor Sensor4 detects the black line, it needs to turn to the left at a large angle to adjust, which is the left_M function. When the rightmost sensor Sensor1 detects the black line, it needs to turn to the right at a large angle to adjust, which is the right_M function. Similarly, when the two sensors in the middle, Sensor3/Sensor2, detect the black line, a small angle adjustment is required. The car goes straight when none of the sensors detect the black line. (When assembling, keep the distance between the two infrared sensors in the middle as far as possible, so that one is on the left of the black line and

the other is on the right of the black line.)



When it is found that the forward speed is too fast and it is easy to rush out of the route, it can be adjusted by modifying the speed parameter in the forward function forward().

```
67  void forward() {//forward function
68    analogWrite(ENA, 150) //Set the speed of ENA
69    analogWrite(ENB, 150) //Set the speed of ENB
70    digitalWrite(IN1, LOW);
71    digitalWrite(IN2, HIGH);
72    digitalWrite(IN3, HIGH);
73    digitalWrite(IN4, LOW);
74    // Serial.println("Forward");
```

Similarly, the size of the turn can also be adjusted by modifying the speed.

```
77  void left() {//left function
78    analogWrite(ENA, 150) //Set the speed of ENA
79    analogWrite(ENB, 150) //Set the speed of ENB
80    digitalWrite(IN1, LOW);
81    digitalWrite(IN2, HIGH);
82    digitalWrite(IN3, LOW);
83    digitalWrite(IN4, HIGH);
84    // Serial.println("Left");
85  }
86
87  void left_M() {//left function
88    analogWrite(ENA, 200);//Set the speed of ENA
89    analogWrite(ENB, 200);//Set the speed of ENB
90    digitalWrite(IN1, LOW);
91    digitalWrite(IN2, HIGH);
92    digitalWrite(IN3, LOW);
93    digitalWrite(IN4, HIGH);
94    // Serial.println("Left");
```

For a more detailed introduction to the 4-way tracking module, please refer to the relevant reference materials, folder:

3_Reference