

Lesson 3 Color Tracking and Sorting

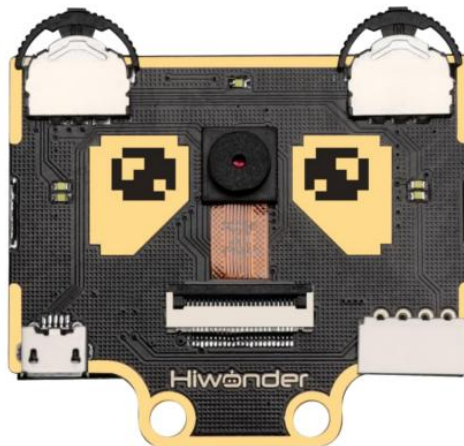
Note: Before starting this game, please make sure the colors including red, green, blue to be recognized are learned as ID1, ID2 and ID3 in sequence with WonderCam module. The specific content refer to “Lesson 1 Color Recognition Learning” under the same directory.

WonderCam vision module must face down when assembling on MaxArm. The specific assembly tutorial refers to “Lesson 2 WonderCam Module Assembly (Top View)” in the same directory.

1. Project Principle

This game uses WonderCam vision module to recognize color. After the color is recognized, robot arm is controlled to suck up object and sort it into the corresponding position by calling the functions in kinematics library.

The color tracking and sorting mainly use WonderCam module to recognize color, as the figure shown below:



The path of the program file: “7. AI Vision Game/ Arduino Development/ Program Files/ Color Tracking and Sorting/ Tracking_Sorting/ Tracking_Sorting.ino ”

2. Preparation

2.1 Hardware

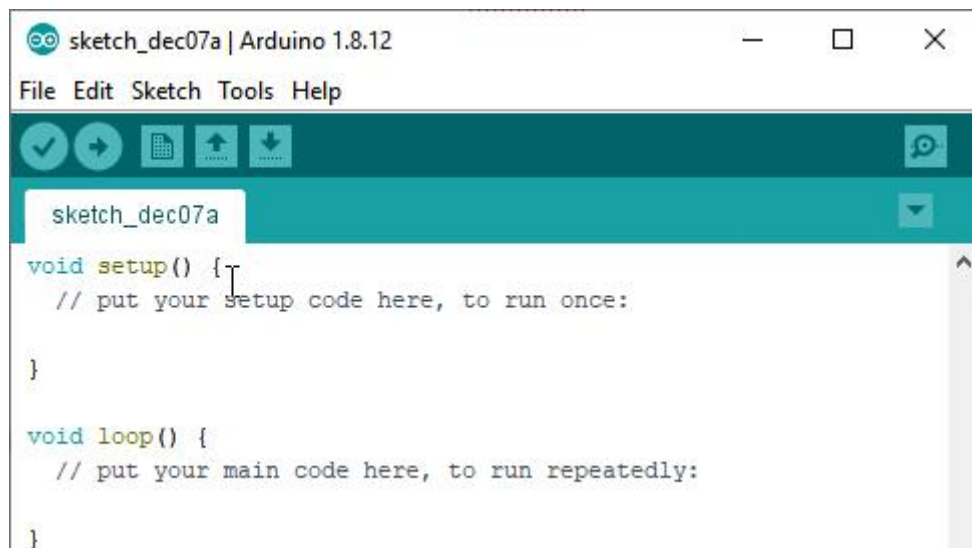
Please refer to “Lesson 2 WonderCam Module Assembly (Top view)” in folder “7. AI Vision Game/ Arduino Development/ Lesson 3 Color Tracking and Sorting” to assemble WonderCam to the corresponding position on MaxArm.

2.2 Software

Please refer to “4. Underlying Program Learning/ Arduino Development/ Lesson 1 Set Development Environment” to connect MaxArm to Arduino.

3. Program Download

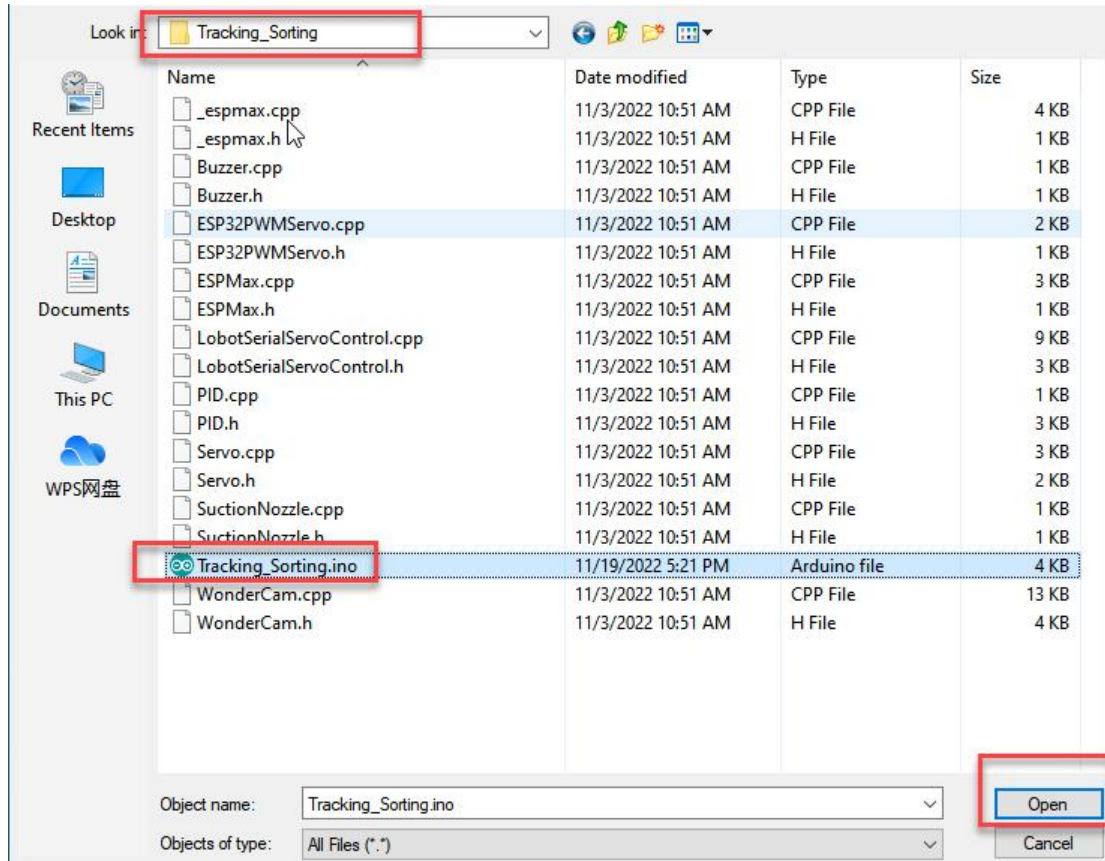
- 1) Double click to open Arduino IDE .



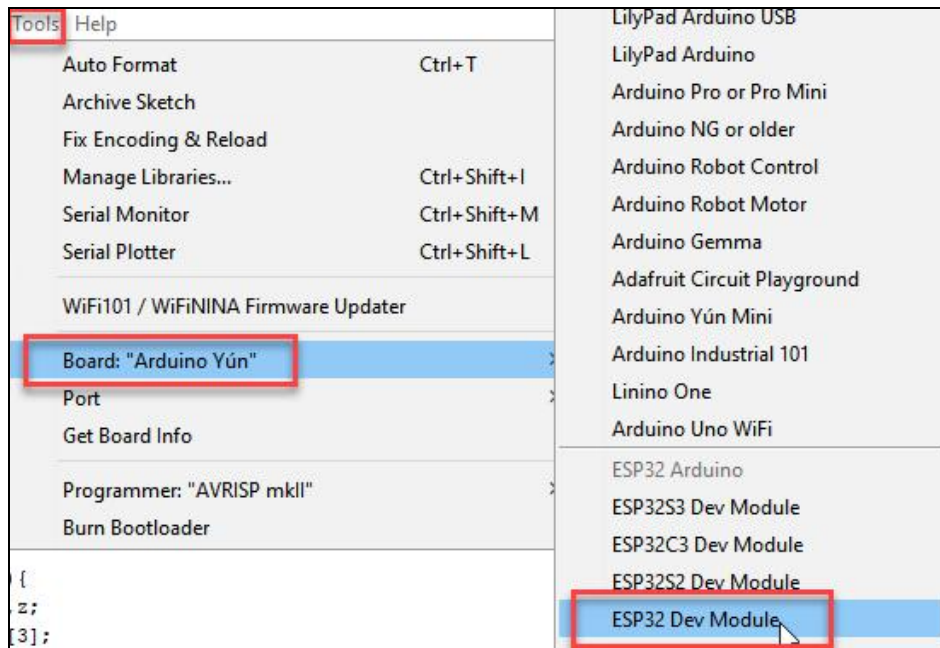
- 2) Click “File -- Open”.



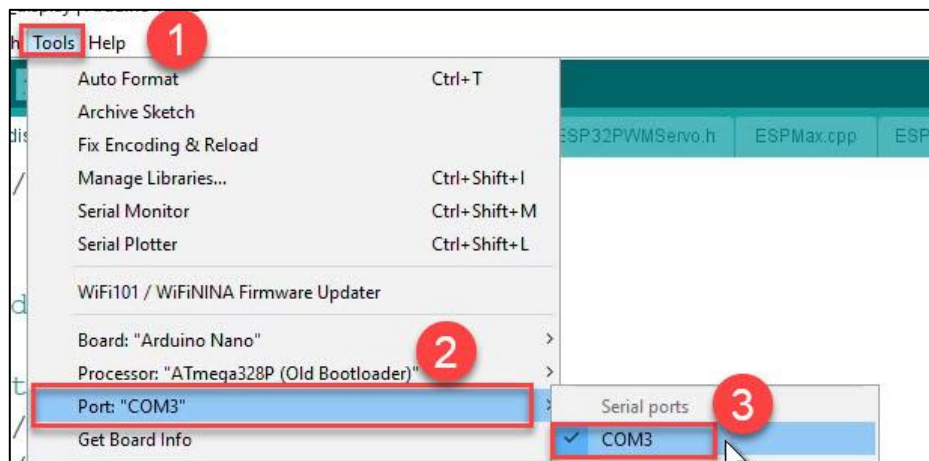
- 3) Open the program “Tracking_Sorting.ino” in the folder “6. Secondary Development/ Arduino Development/ Sensor-extension Game/Program Files/ Color Tracking and Sorting/ Tracking_Sorting”.



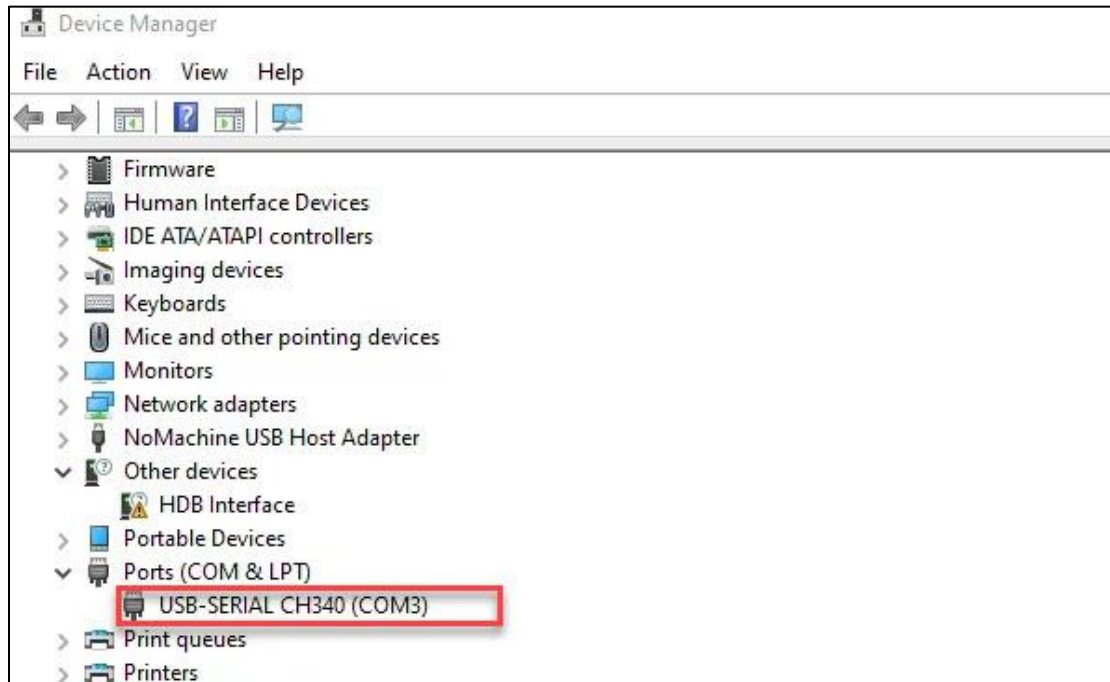
- 4) Select the corresponding board model. Click “Tool -- Board” and select “ESP 32 Dev Module” (If the board model has been configured when setting development environment, you can skip this step.)



- 5) Select the corresponding port 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.)




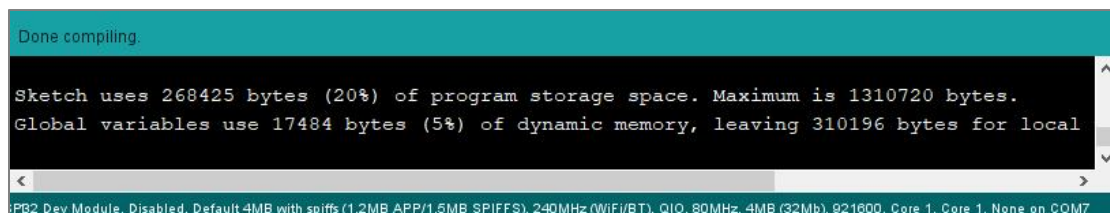
- 6) If you're not sure about the port number, please open the "This PC" and click "Properties->Device Manger" in turns to check the corresponding port number (the device is with CH340).




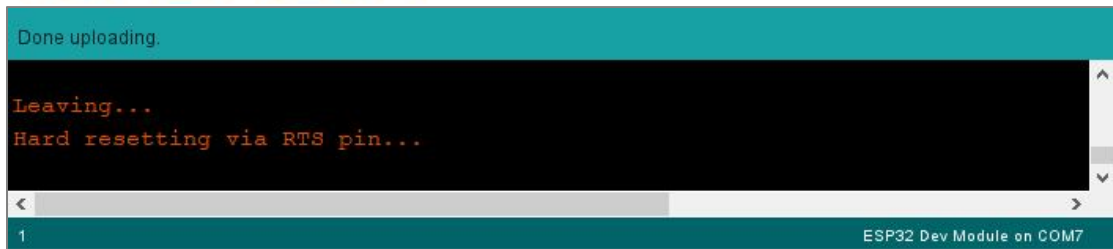
- 7) After selecting, confirm the board “ESP32 Dev Module” in the lower right corner and the port number “COM5” (it is an example here, please refer to the actual situation).



- 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

Hold red, green or blue block within module recognition area. After color is recognized, MaxArm will move with the block. When the block stop moving, robotic arm will suck and sort it into the corresponding area.

5. Program Analysis

5.1 Import function library

The path of the program file: "7. AI Vision Game/ Arduino Development/ Program Files/ Color Tracking and Sorting/ Tracking_Sorting/ Tracking_Sorting.ino".

Before the program is executed, the PID, vision module, buzzer, dot matrix module, PWM servo, bus servo, air pump and other related function libraries are required to import.

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


5.2 Initialization

Then, initialize the corresponding modules and set the vision module to color recognition mode.

Set PID parameters (Proportional, Integral and Derivative). Through PID algorithm, the difference between the center coordinate of image and the centre coordinate of screen can be calculated and thus robot arm can be controlled to track the target object, as the figure shown below:

```
10 WonderCam cam;  
11  
12 arc::PID<double> x_pid(0.045, 0.0001, 0.0001);  
13 arc::PID<double> y_pid(0.045, 0.0001, 0.0001);
```

```
15 void setup() {  
16     cam.begin();  
17     Buzzer_init();  
18     ESPMax_init();  
19     Nozzle_init();  
20     PWMServo_init();  
21     Valve_on();  
22     SetPWMServo(1, 1500, 1000);  
23     Valve_off();  
24     setBuzzer(100);  
25     Serial.begin(115200);  
26     Serial.println("start...");  
27     cam.changeFunc(APPLICATION_COLORDETECT);
```

5.3 Color Recognition

Use While statement to constantly update the color data and position coordinate detected by vision module. (WonderCam vision module must first learn the color before recognizing, please refer to “Lesson 1 Color Recognition Learning” under the same file directory), as the figure shown below:

```
41 while (true) {
42     int color_x = 160;
43     int color_y = 120;
44     cam.updateResult();
45     if (cam.anyColorDetected()) {
46         WonderCamColorDetectResult p;
47         if (cam.colorIdDetected(1)) {
48             cam.colorId(1, &p);
49             color_x = p.x; color_y = p.y;
50             angle_pwm = 2100;
51             place[0] = -120; place[1] = -140; place[2] = 85;
```

5.4 Color Tracking

The tracking function is realized by PID algorithm. Due to a certain movement range to robotic arm on space coordinate, so the coordinate limit of x, y and z axes needs to be set first, as the figure shown below:

```
82     if (pos[0] > 100) pos[0] = 100;
83     if (pos[0] < -100) pos[0] = -100;
84     if (pos[1] > -60) pos[1] = -60;
85     if (pos[1] < -240) pos[1] = -240;
86     set_position(pos, 50);
```

Then save the coordinate value of x and y axes obtained for vision module to these two variables "color_x" and "color_y", as the figure shown below:

```
49         color_x = p.x; color_y = p.y;
```

Next, the s-axis coordinate value of the color block is subtracted from the x-axis coordinate value of the center point of the vision module screen to obtain the distance between the color block and the center of the screen (take x-axis for example), which is calculated as follows.

```
66     if (abs(color_x - 160) < 15) {
67         color_x = 160;
68     }
69     x_pid.setTarget(160);
70     x_pid.setInput(color_x);
71     float dx = x_pid.getOutput();
72     pos[0] -= dx;
```


The same calculation method for y axis is shown below.

```
74     if (abs(color_y - 120) < 10) {  
75         color_y = 120;  
76     }  
77     y_pid.setTarget(120);  
78     y_pid.setInput(color_y);  
79     float dy = y_pid.getOutput();  
80     pos[1] -= dy;
```

After calculating the coordinate position of the target color on screen, substitute the value into the inverse kinematics function to achieve the tracking function, as the figure shown below,

```
86     set_position(pos, 50);
```

The first parameter “(x, y, z)” represents the coordinate of the end effector. (The difference between the coordinate of the center of the target color and the coordinates of the center of the screen).

The second parameter “50” represents the time it takes for the robot arm to move to the coordinate position.

5.5 Sorting

The sorting function is implemented by inverse kinematics function, as the figure shown below:

```

97         set_position(pos, 1000);
98         delay(1000);
99         pos[2] = 85;
100        set_position(pos, 600);
101        Pump_on();
102        delay(1000);
103        pos[2] = 150;
104        set_position(pos, 1000);
105        delay(1000);
106        pos[0] = place[0]; pos[1] = place[1];
107        set_position(pos, 1200);
108        delay(1200);
109        SetPWMServo(1, angle_pwm, 800);
110        delay(200);
111        set_position(place, 1000);
112        delay(1000);
113        Valve_on();
114        place[2] = 150;
115        set_position(place, 1000);
116        delay(1000);
117        Valve_off();
118        pos[0] = 0; pos[1] = -120; pos[2] = 150;
119        set_position(pos, 1500);
120        SetPWMServo(1, 1500, 800);

```

When module detects that the block stops moving, robot arm will execute sorting action, as the figure shown below:

```

88         if ((abs(dx) < 0.1) & (abs(dy) < 0.1)) {

```

To avoid wrong recognition, the program will detect ten times, and then start a new round of detection.

Then set coordinate parameter to sort the block into corresponding position. To make the block place in a right direction, a compensation angle will be set.

```

109        SetPWMServo(1, angle_pwm, 800); //设置角度补偿

```

The specific sorting process is as follow:

```

92      setBuzzer(100);
93      float d_x = pos[0] / 2.3;
94      float d_y = 68 - abs(d_x / 3);
95
96      pos[0] += d_x; pos[1] -= d_y;
97      set_position(pos, 1000);
98      delay(1000);
99      pos[2] = 85;
100     set_position(pos, 600);
101     Pump_on();
102     delay(1000);
103     pos[2] = 150;
104     set_position(pos, 1000);
105     delay(1000);
106     pos[0] = place[0]; pos[1] = place[1];
107     set_position(pos, 1200);
108     delay(1200);
109     SetPWMServo(1, angle_pwm, 800);
110     delay(200);
111     set_position(place, 1000);
112     delay(1000);
113     Valve_on();
114     place[2] = 150;
115     set_position(place, 1000);
116     delay(1000);
117     Valve_off();
118     pos[0] = 0; pos[1] = -120; pos[2] = 150;

```

The coordinates of the center of block and screen according to PID algorithm.

```

93      float d_x = pos[0] / 2.3;
94      float d_y = 68 - abs(d_x / 3);

```

Next, use kinematics function to control robot arm, as the figure shown below:

```

96      pos[0] += d_x; pos[1] -= d_y;
97      set_position(pos, 1000);

```

The first parameter "pos" represents the coordinates of the block, i.e. the coordinates of the block (dx, dy, dz) is calculated by pid algorithm.

The second parameter "1000" is the running time and the unit is ms.

Then reduce the value of the z-axis, i.e., Z = 86, which lower the block in

preparation for sucking the block, as shown below.

```
99      pos[2] = 85;  
100     set_position(pos, 600);
```

Then turn on air pump to suck the block, as the figure shown below:

```
101     Pump_on();
```

Increase z-axis value, e.i., z=150, to raise robot arm, as the figure shown below:

```
103     pos[2] = 150;  
104     set_position(pos, 1000);
```

Then place the block at the specific position, as the figure shown below:

```
106     pos[0] = place[0]; pos[1] = place[1];  
107     set_position(pos, 1200);
```

Finally, robot arm returns to the initial position and wait for the next round of game.

```
118     pos[0] = 0; pos[1] = -120; pos[2] = 150;  
119     set_position(pos, 1500);  
120     SetPWMServo(1, 1500, 800);
```