Lesson 3 Color Tracking and Display

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.

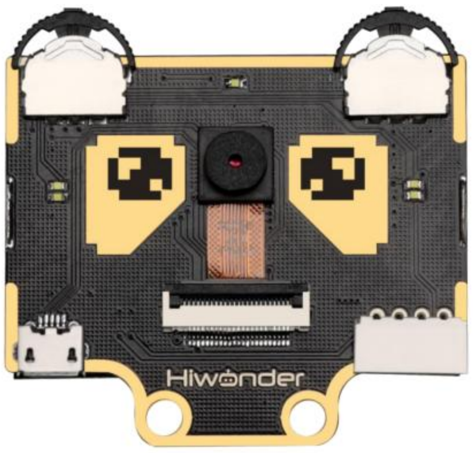
If you skip the step above, WonderCam module can not recognize the colors and complete this game.

1. Project Principle

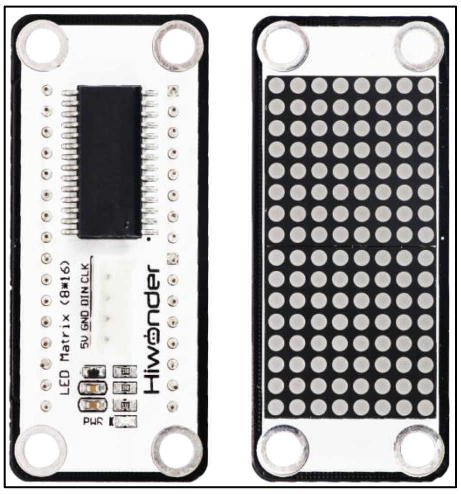
This game uses WonderCam to identify the object color. After the color is recognized, the initial letter of corresponding color will display on dot matrix module, as the figure shown below:

Color	Red	Green	Blue
Display content	R	G	B

The following picture shows WonderCam AI vision module:



The following picture shows the dot matrix module:

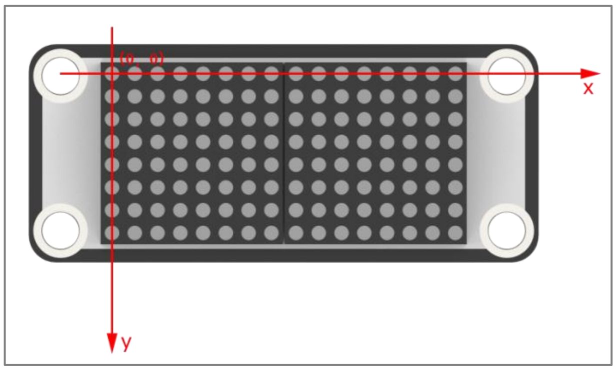


The module is composed of two red 8x8 LED dot-matrix screens, which can be controlled by driving the control chip.

The display content of the dot matrix is realized by setting the high and low levels of the two pins of the diodes. When the corresponding line is set to 1 level and a column is set to 0 level, the corresponding diodes light up, that is, 0 represents on and 1 represents off.

The program controls the display pattern with a set of hexadecimal data, a total of 16 data. Each data controls a column of LED lights, and the pattern is displayed by setting individual point.

To explain more clearer, the rows and columns of dot matrix are marked with corresponding numbers, as the figure shown below:



x: The specific origin position of the bitmap x-axis coordinate and the value is from 0 to 15. The set in the program is 0.

y: The specific origin position of the bitmap y-axis coordinate and the value is from 0 to -7. The set in the program is 0.

If want the LED at position (0, 0) to light up first, we need to set the binary (from top to bottom) of the first column to 01111111, which translates to 0x7f in hexadecimal.

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

2. Preparation

2.1 Hardware

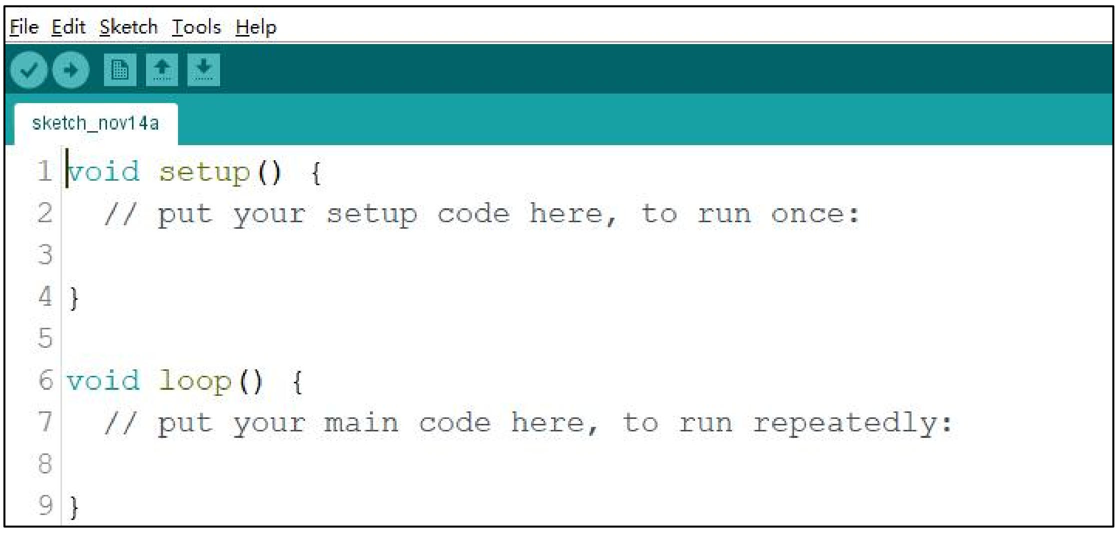
Please refer to “Lesson 1 Module Assembly” to assemble module to the corresponding position on MaxArm (WonderCam AI vision module has been assembled to robotic arm. The specific assembly tutorial for WonderCam module refers to “7. AI Vision Games/ WonderCam AI Vision Module Learning(Must read!)/ Lesson 3 Assembly”).

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

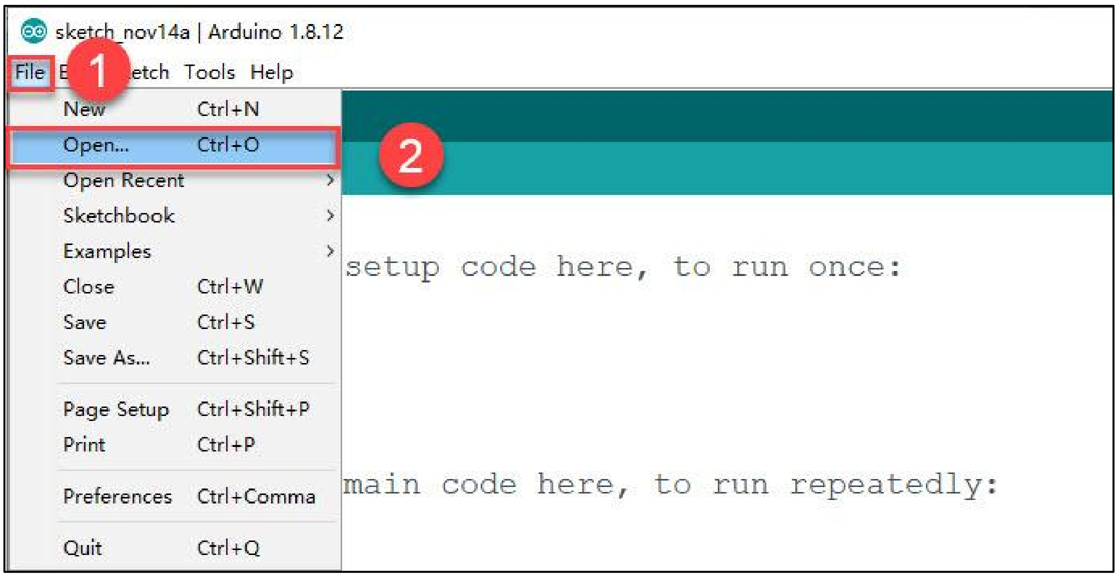
1) Double click to open Arduino IDE .


File Edit Sketch Tools Help

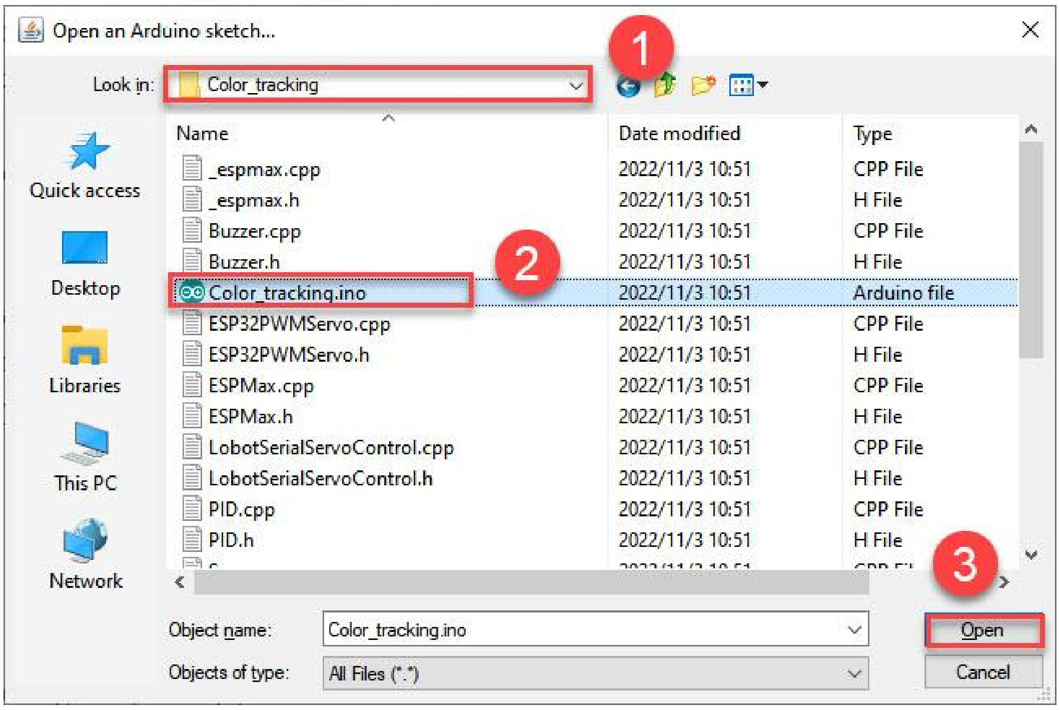
sketch_nov14a

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

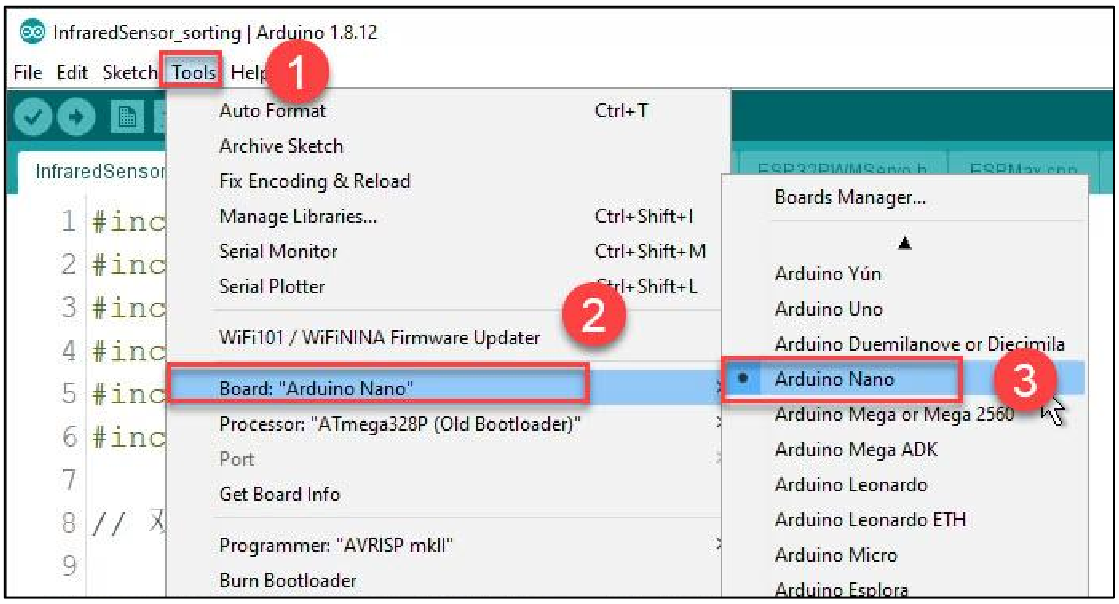
2) Click “File -- Open”.



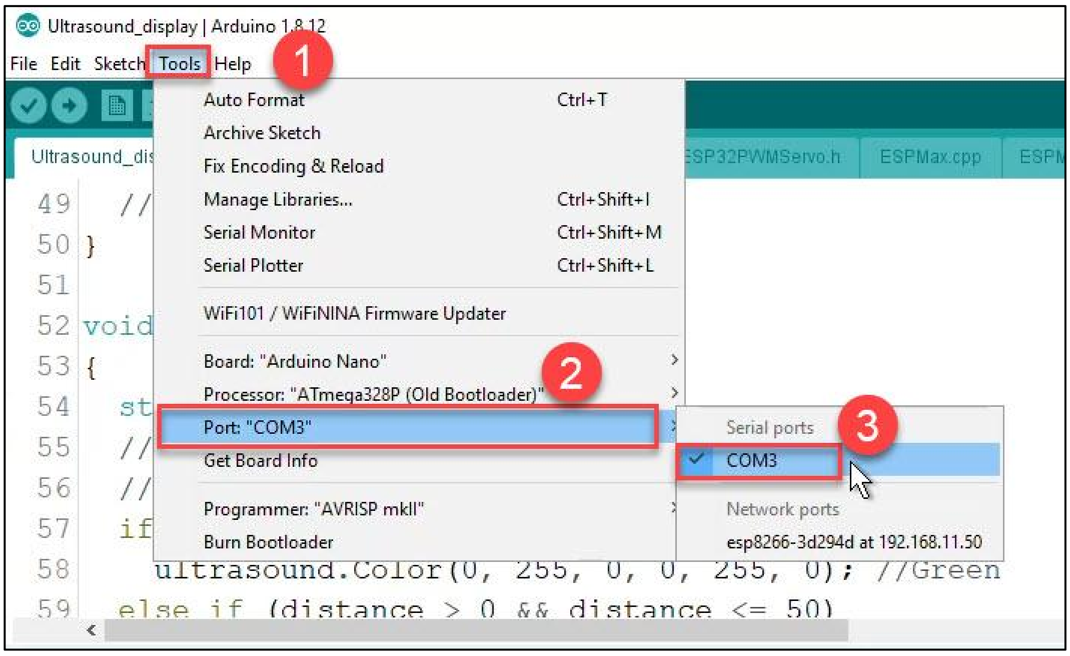
3) Open the program “Color_tracking.ino” in the folder “7. AI Vision Game/ Arduino Development/ Program Files/Color Tracking and Display/ Color_tracking”.


Open an Arduino sketch...
Look in: Color_tracking
1
Quick access
Desktop
Libraries
This PC
Network
Name
Date modified
Type
_espmax.cpp 2022/11/3 10:51 CPP File
_espmax.h 2022/11/3 10:51 H File
Buzzer.cpp 2022/11/3 10:51 CPP File
Buzzer.h 2022/11/3 10:51 H File
2
Color_tracking.ino 2022/11/3 10:51 Arduino file
ESP32PWMServo.cpp 2022/11/3 10:51 CPP File
ESP32PWMServo.h 2022/11/3 10:51 H File
ESPMax.cpp 2022/11/3 10:51 CPP File
ESPMax.h 2022/11/3 10:51 H File
LobotSerialServoControl.cpp 2022/11/3 10:51 CPP File
LobotSerialServoControl.h 2022/11/3 10:51 H File
PID.cpp 2022/11/3 10:51 CPP File
PID.h 2022/11/3 10:51 H File
3
Object name: Color_tracking.ino
Objects of type: All Files (*.*)
Open
Cancel

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.)

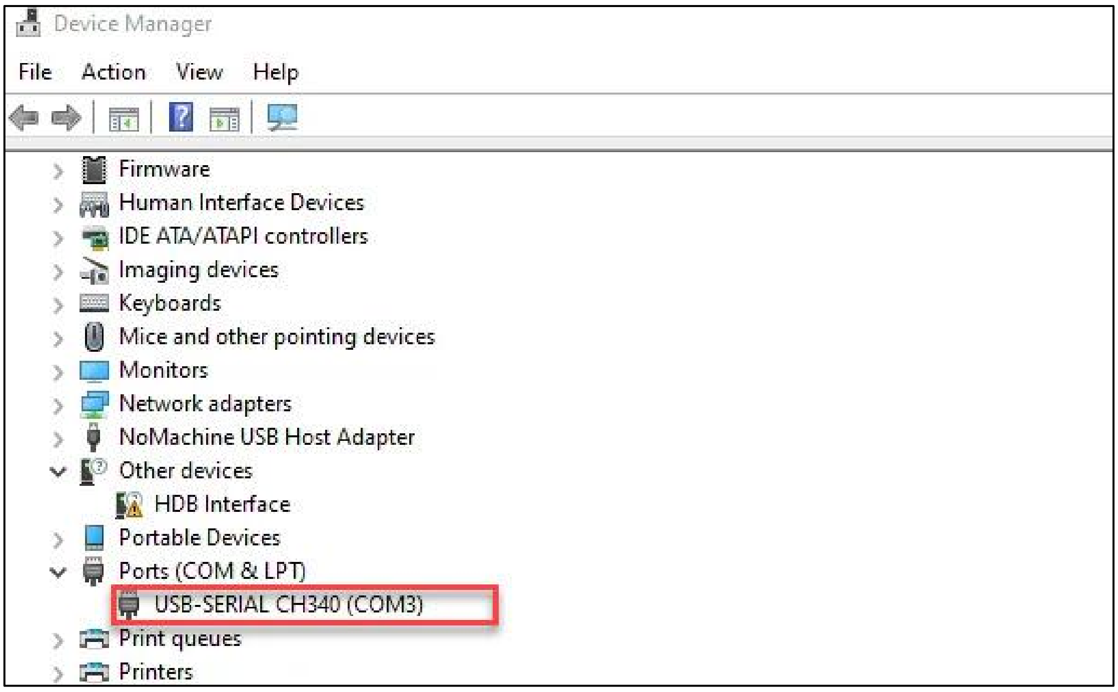

InfraredSensor_sorting | Arduino 1.8.12
File Edit Sketch Tools Help
1
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
2
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Nano"
Processor: "ATmega328P (Old Bootloader)"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino Yún
Arduino Uno
Arduino Duemilanove or Diecimila
● Arduino Nano
3
Arduino Mega or Mega 2560
Arduino Mega ADK
Arduino Leonardo
Arduino Leonardo ETH
Arduino Micro
Arduino Esplora
InfraredSensor
#inc
#inc
#inc
#inc
#inc
#inc
//

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.)

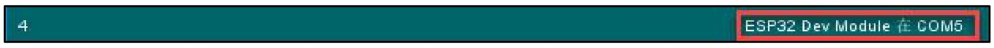

Ultrasound_display | Arduino 1.8.12
File Edit Sketch Tools Help
1
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Nano"
2
Processor: "ATmega328P (Old Bootloader)"
Port: "COM3"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
3
COM3
Network ports
esp8266-3d294d at 192.168.11.50
Ultrasound_dis
ESP32PWMServo.h
ESPMax.cpp
ESPM

```
  //
}

void
{
  st
  //
  //
  if
    ultrasound.Color(0, 255, 0, 0, 255, 0); //Green
  else if (distance > 0 && distance <= 50)
```

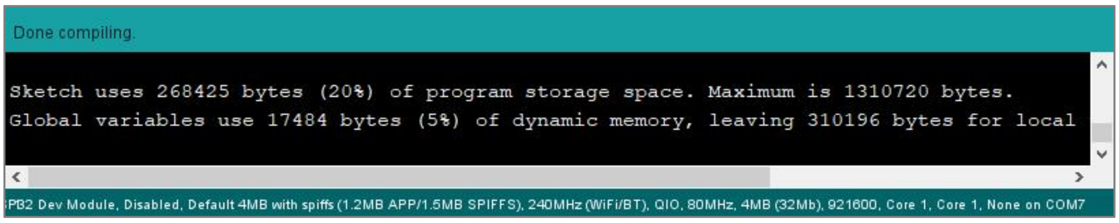
6) If you're not sure about the port number, please open the "This PC" and click "Properties->Device Manger" in turns to check the corresponding port number (the device is with CH340).



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).


4 ESP32 Dev Module 在 COM5

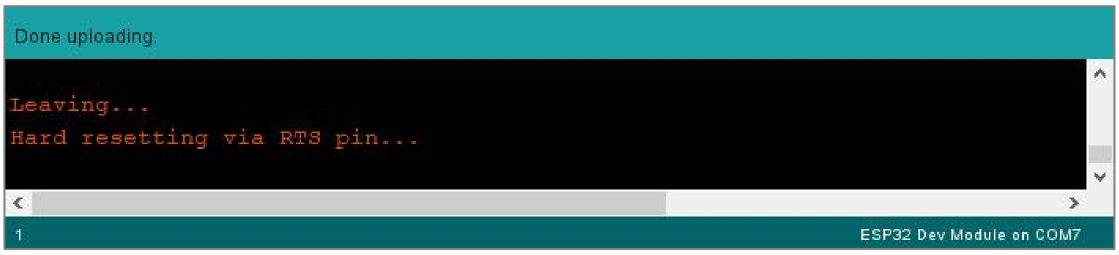
8) Then click on icon to verify the program. If no error, the status area will display “Compiling->Compile complete” in turn. After compiling, the information such as the current used bytes, and occupied program storage space will be displayed.


Done compiling.

```
Sketch uses 268425 bytes (20%) of program storage space. Maximum is 1310720 bytes.
Global variables use 17484 bytes (5%) of dynamic memory, leaving 310196 bytes for local
```

P32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, Core 1, Core 1, None on COM7

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.


Done uploading.

```
Leaving...
Hard resetting via RTS pin...
```

1 ESP32 Dev Module on COM7

4. Project Outcome

Hold red, green or blue block within module recognition area. After color is recognized, MaxArm will move with the block and the initial letter of corresponding color will be displayed on dot matrix module.

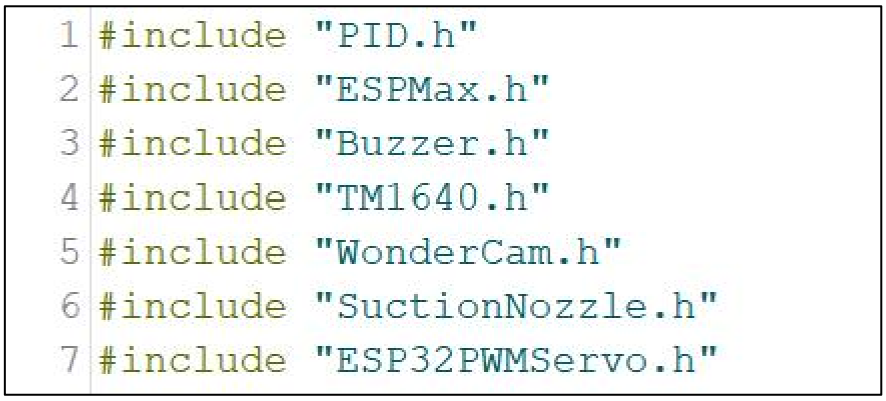
Color	Red	Green	Blue
Display content	R	G	B

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 Display/ Color_tracking/Color_tracking.ino”.

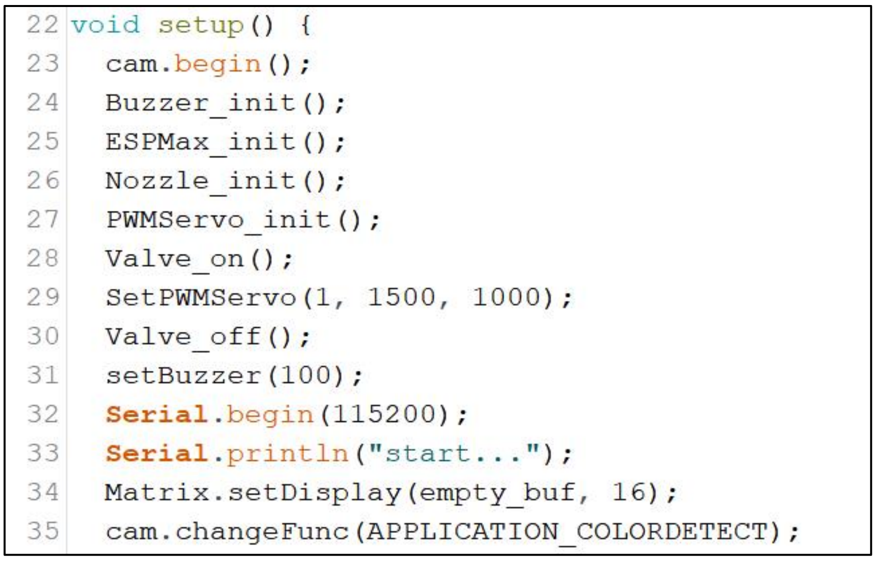
Before the program is executed, the PID, vision module, buzzer, dot matrix module, PWM servo, bus servo, air pump and other related function libraries are required to be import.



```
#include "PID.h"
#include "ESPMax.h"
#include "Buzzer.h"
#include "TM1640.h"
#include "WonderCam.h"
#include "SuctionNozzle.h"
#include "ESP32PWMServo.h"
```

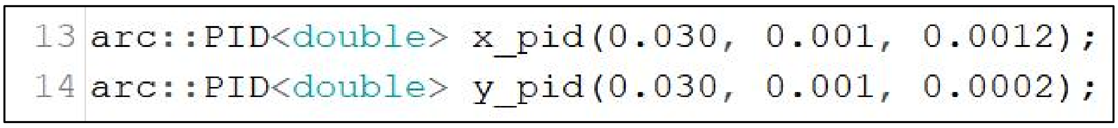
5.2 Initialization

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



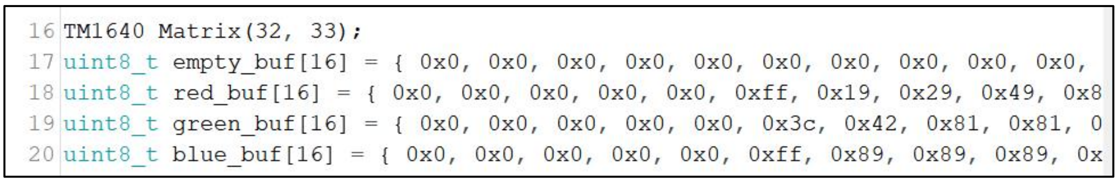
```
void setup() {
  cam.begin();
  Buzzer_init();
  ESPMax_init();
  Nozzle_init();
  PWMServo_init();
  Valve_on();
  SetPWMServo(1, 1500, 1000);
  Valve_off();
  setBuzzer(100);
  Serial.begin(115200);
  Serial.println("start...");
  Matrix.setDisplay(empty_buf, 16);
  cam.changeFunc(APPLICATION_COLORDETECT);
```

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.



```
arc::PID<double> x_pid(0.030, 0.001, 0.0012);
arc::PID<double> y_pid(0.030, 0.001, 0.0002);
```

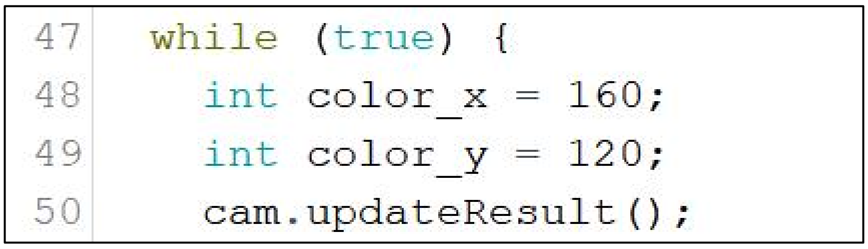
The set the hexadecimal address character of LED dot matrix module enable the module display the corresponding letter according to the recognized color, as the figure shown below:



```
TM1640 Matrix(32, 33);
uint8_t empty_buf[16] = { 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
uint8_t red_buf[16] = { 0x0, 0x0, 0x0, 0x0, 0x0, 0xff, 0x19, 0x29, 0x49, 0x8
uint8_t green_buf[16] = { 0x0, 0x0, 0x0, 0x0, 0x0, 0x3c, 0x42, 0x81, 0x81, 0
uint8_t blue_buf[16] = { 0x0, 0x0, 0x0, 0x0, 0x0, 0xff, 0x89, 0x89, 0x89, 0x
```

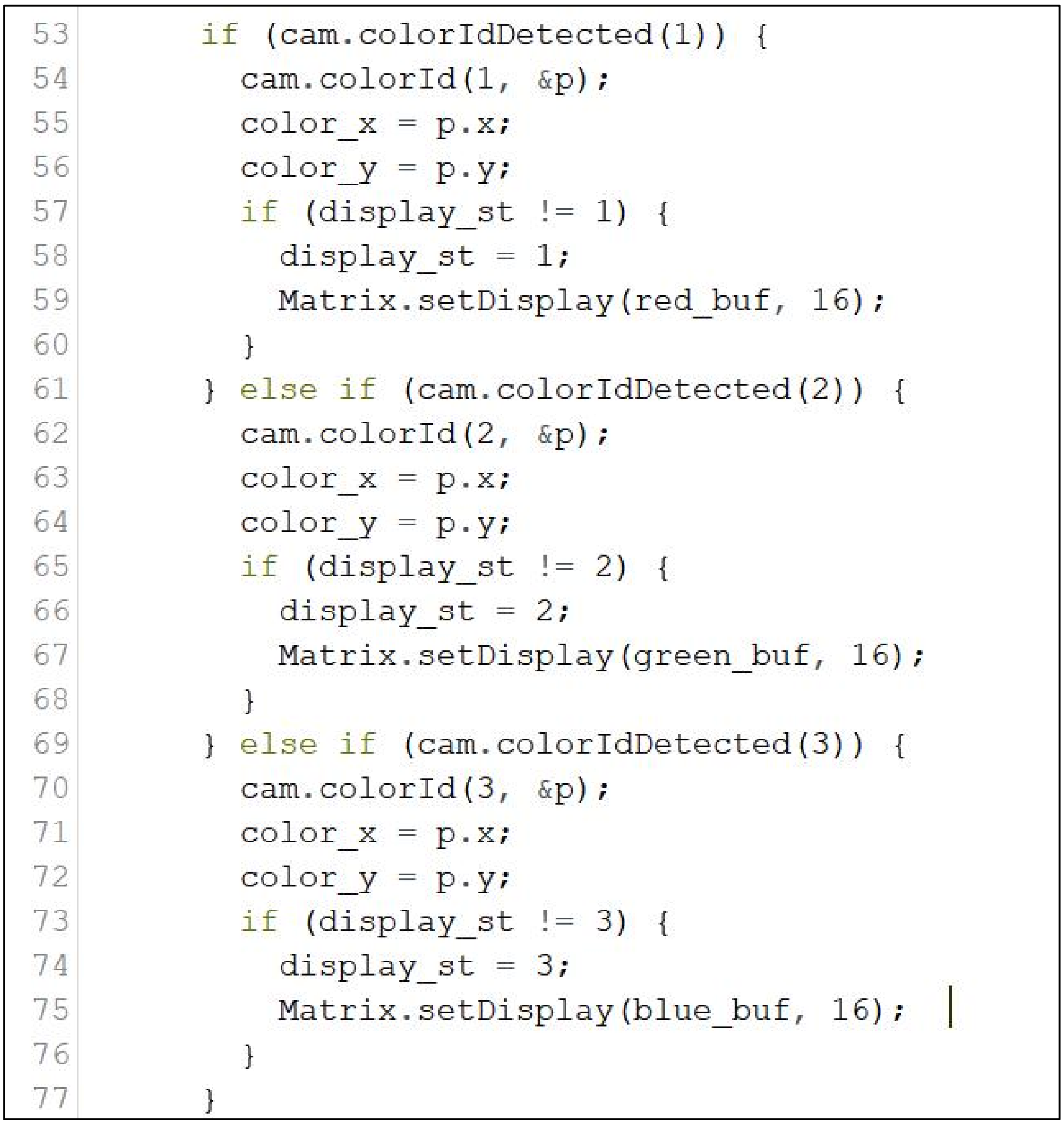
5.3 Recognize and Display color

Use While statement to constantly update the color data 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:

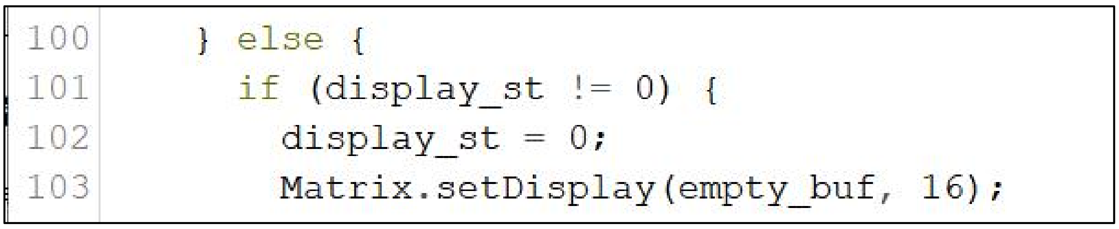


```
  while (true) {
    int color_x = 160;
    int color_y = 120;
    cam.updateResult();
```

Then use if condition statement to display the corresponding letter according to the recognized color. If it is not red, green or blue, the dot matrix module displays no content, as the figure shown below:



```
      if (cam.colorIdDetected(1)) {
        cam.colorId(1, &p);
        color_x = p.x;
        color_y = p.y;
        if (display_st != 1) {
          display_st = 1;
          Matrix.setDisplay(red_buf, 16);
        }
      } else if (cam.colorIdDetected(2)) {
        cam.colorId(2, &p);
        color_x = p.x;
        color_y = p.y;
        if (display_st != 2) {
          display_st = 2;
          Matrix.setDisplay(green_buf, 16);
        }
      } else if (cam.colorIdDetected(3)) {
        cam.colorId(3, &p);
        color_x = p.x;
        color_y = p.y;
        if (display_st != 3) {
          display_st = 3;
          Matrix.setDisplay(blue_buf, 16);
        }
      }
```



```
    } else {
      if (display_st != 0) {
        display_st = 0;
        Matrix.setDisplay(empty_buf, 16);
```

5.4 Robotic arm tracking

Color tracking uses PID algorithm to set x, y, and z values of end effector to achieve the tracking effect, as the figure shown below:

```
    if (abs(color_x - 160) < 15) {
      color_x = 160;
    }
    x_pid.setTarget(160);
    x_pid.setInput(color_x);
    pos[0] -= x_pid.getOutput();

    if (abs(color_y - 120) < 10) {
      color_y = 120;
    }
    y_pid.setTarget(120);
    y_pid.setInput(color_y);
    pos[2] += y_pid.getOutput();

    if (pos[0] > 100) pos[0] = 100;
    if (pos[0] < -100) pos[0] = -100;
    if (pos[2] > 180) pos[2] = 180;
    if (pos[2] < 100) pos[2] = 100;
```

Firstly, obtain the coordinate data of the target color on screen (take red for example), as the figure shown below:

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

Calculate the coordinates of the target color and center point obtained from vision module. If the calculated x-axis value satisfies the following conditions, PID algorithm is used to calculate the distance between the target color and the robot arm coordinate.

```
    if (abs(color_x - 160) < 15) {
      color_x = 160;
    }
    x_pid.setTarget(160);
    x_pid.setInput(color_x);
    pos[0] -= x_pid.getOutput();

```

The specific calculation method is as follow:

```
        x_pid.setTarget(160);
        x_pid.setInput(color_x);
        pos[0] -= x_pid.getOutput();
```

Because the movement of the robot arm on space coordinate has a certain range, it is necessary to set a movement range for x.

```
        if (pos[0] > 100) pos[0] = 100;  // 机械臂X轴范围限幅
        if (pos[0] < -100) pos[0] = -100;
```

The same calculation method for y and z axes.

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,

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

The first parameter “pos” represents the coordinate values of x, y and z axes of 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.