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" under the same directory.

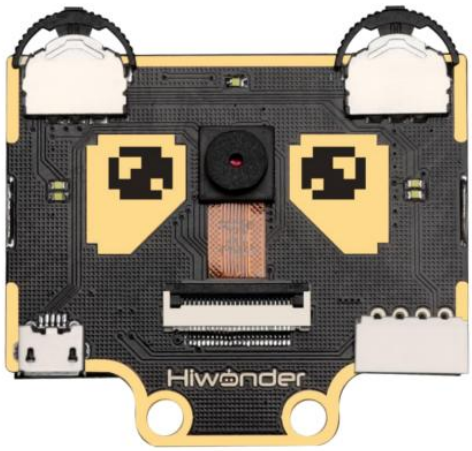
If you skip the step above, WonderCam can not recognize the colors and complete this game.

1. Project Principle

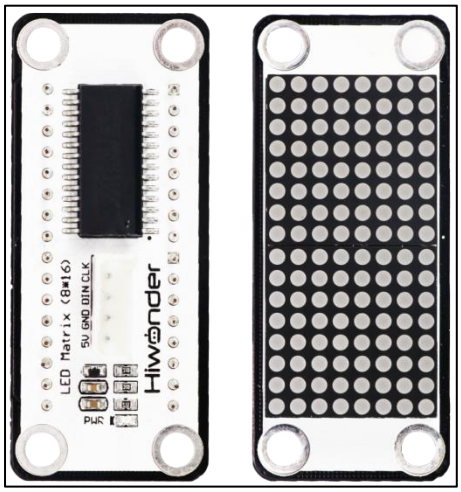
This game uses WonderCam to identify the object color. After the color is recognized, the initial letter of corresponding color will display on dot matrix module, as the figure shown below:

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:

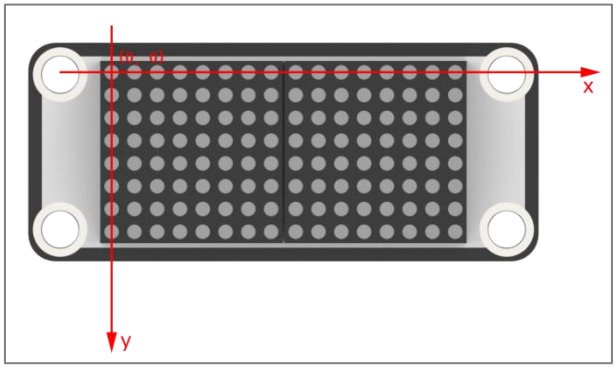


The module is composed of two red 8x8 LED dot-matrix screens, which can be controlled by driving the control chip.

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/ Python Development/ Program Files/ Color Tracking and Display/ main.py”.

```
while True:
    cam.update_result()
    if cam.get_color_blob(1):
        tm.write(red_buf)
        color_data = cam.get_color_blob(1)
    elif cam.get_color_blob(2):
        tm.write(green_buf)
        color_data = cam.get_color_blob(2)
    elif cam.get_color_blob(3):
        tm.write(blue_buf)
        color_data = cam.get_color_blob(3)
    else:
        tm.update_display
        color_data = None

    if color_data:
        center_x = color_data[0]
        center_y = color_data[1]
```

2. Preparation

2.1 Hardware

Please refer to “Lesson 2 Dot Matrix Module Assembly” in folder “7. AI Vision Game/ Python Development/ Lesson 2 Color Tracking and Display” to assemble module to the corresponding position on MaxArm (WonderCam AI vision module has been assembled to robotic arm. The specific assembly

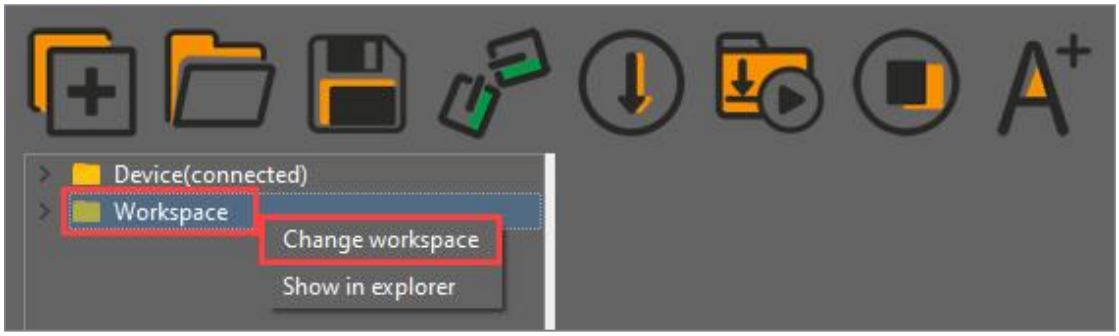
tutorial for WonderCam module refers to “7. AI Vision Games/ WonderCam AI Vision Module Learning(Must read!)/ Lesson 3 Assembly”).

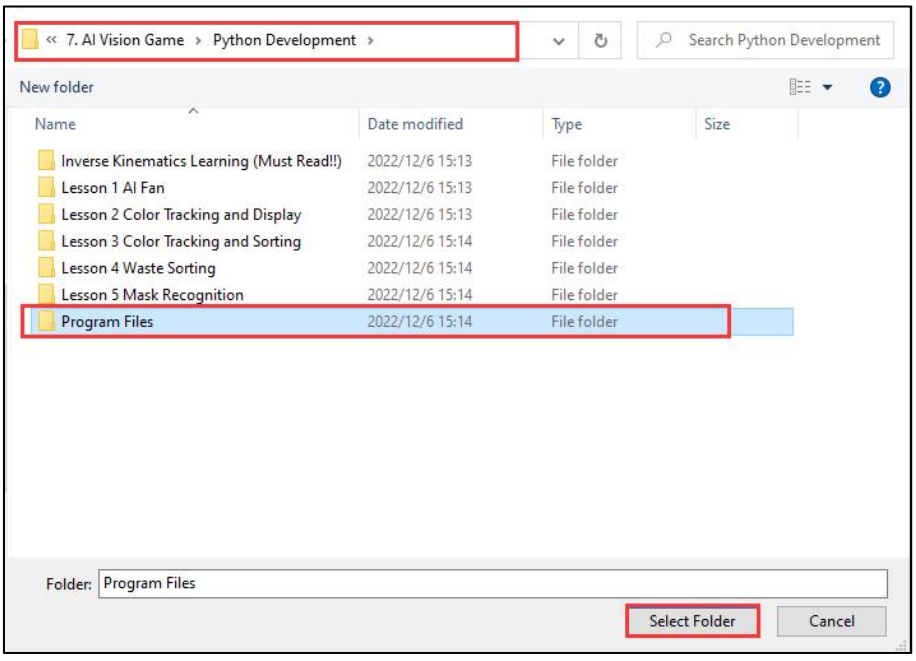
2.2 Software

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

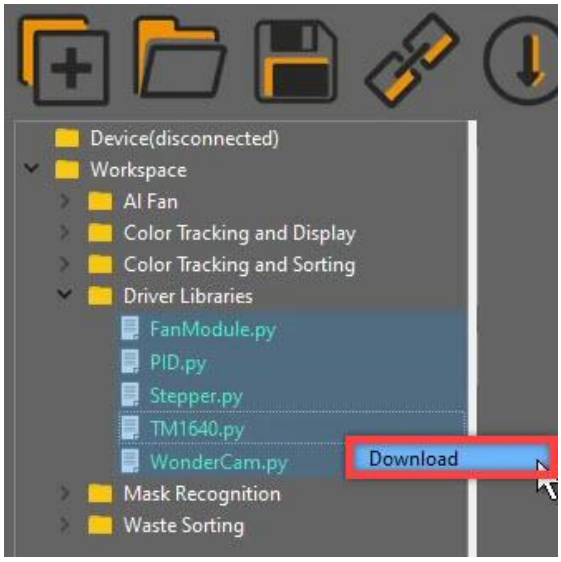
3. Program Download

1) After connecting, change the path of Workspace to “7. AI Vision Game/ Python Development”, and select “Program Files”.

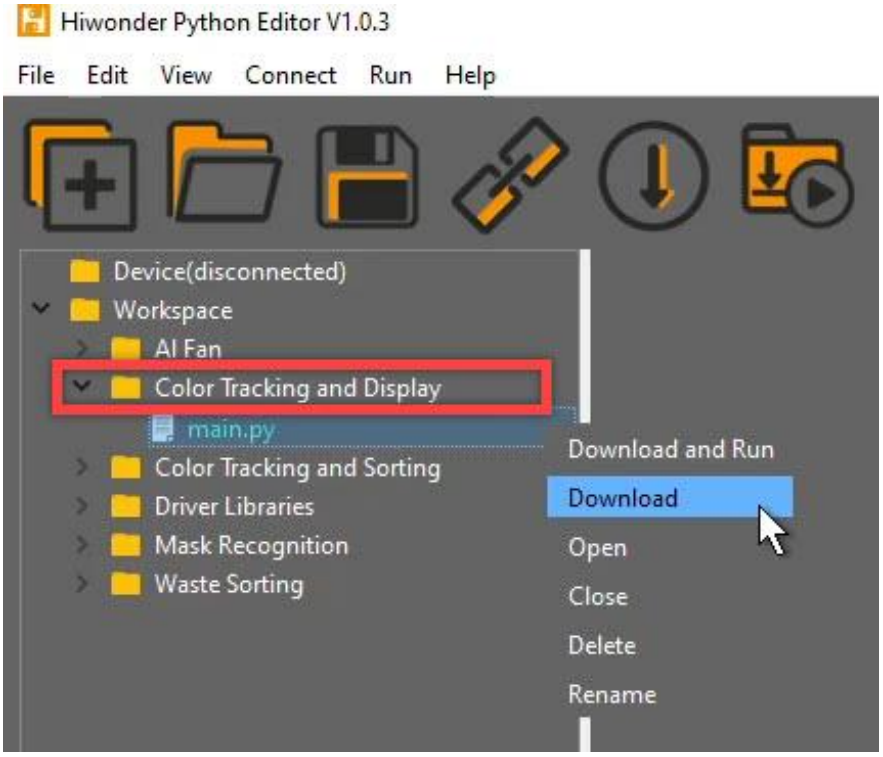




2) Then double click “Drive Library” folder, and select all library files, and right click and select “Download” to download the library files to the controller. (If you have downloaded this library file before, please skip this step.)



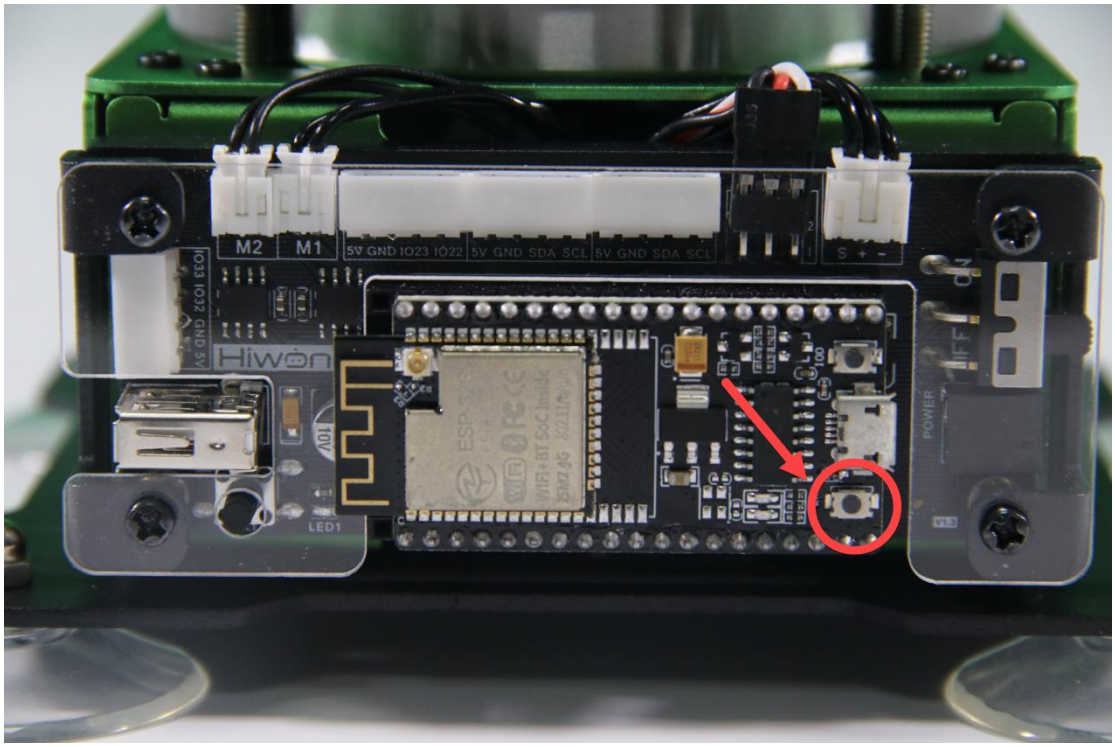
3) Then double click “Color Tracking and Display” folder and select “main.py” folder. Then right click and select “Download” to download all the program files to the controller.



When the terminal prints the prompt as shown in the image below, it means

download completed.

4) After downloading, click on the reset icon or press the reset button on ESP32 controller to run program.



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/ Python Development/ Program Files/ Color Tracking and Display/main.py”.

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

```
import time
import TM1640
from machine import Pin, I2C
from PID import PID
from WonderCam import *
from Buzzer import Buzzer
from espmax import ESPMax
from PWMServo import PWMServo
from BusServo import BusServo
from SuctionNozzle import SuctionNozzle
```

5.2 Initialization

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

```
pwm = PWMServo()
buzzer = Buzzer()
pwm.work_with_time()
bus_servo = BusServo()
arm = ESPMax(bus_servo)
nozzle = SuctionNozzle()
tm = TM1640.TM1640(clk=Pin(33), dio=Pin(32))
i2c = I2C(0, scl=Pin(16), sda=Pin(17), freq=400000)
cam = WonderCam(i2c)
cam.set_func(WONDERCAM_FUNC_COLOR_DETECT)
```

5.3 Main Program Analysis

Come to the part main program analysis. 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.

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:

```
if __name__ == '__main__':
    x, y, z = 0, -120, 150
    buzzer.setBuzzer(100)
    nozzle.set_angle(0,1000)
    arm.set_position((x, y, z), 2000)
    time.sleep_ms(2000)
    x_pid = PID(0.026, 0.001, 0.0008)
    z_pid = PID(0.030, 0.001, 0.0001)
    tm.update_display()

    red_buf = [0x0,0x0,0x0,0x0,0x0,0xff,0x19,0x29,0x49,0x86,0x0,0x0,0x0,0x0,0x0,0x0]
    green_buf = [0x0,0x0,0x0,0x0,0x0,0x3c,0x42,0x81,0x81,0xa1,0x62,0x0,0x0,0x0,0x0,0x0]
    blue_buf = [0x0,0x0,0x0,0x0,0x0,0xff,0x89,0x89,0x89,0x76,0x0,0x0,0x0,0x0,0x0,0x0]
```

5.4 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” under the same file directory), as the figure shown below:

```
while True:
    cam.update_result()
```

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.get_color_blob(1):
    tm.write(red_buf)
    color_data = cam.get_color_blob(1)
elif cam.get_color_blob(2):
    tm.write(green_buf)
    color_data = cam.get_color_blob(2)
elif cam.get_color_blob(3):
    tm.write(blue_buf)
    color_data = cam.get_color_blob(3)
else:
    tm.update_display
    color_data = None
```

5.5 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:

```
55  if color_data:
56      center_x = color_data[0]
57      center_y = color_data[1]
58
59      if abs(center_x - 160) < 15:
60          center_x = 160
61      x_pid.SetPoint = 160
62      x_pid.update(center_x)
63      x -= x_pid.output
64      x = 100 if x > 100 else x
65      x = -100 if x < -100 else x
66
67      if abs(center_y - 120) < 5:
68          center_y = 120
69      z_pid.SetPoint = 120
70      z_pid.update(center_y)
71      z += z_pid.output
72      z = 100 if z < 100 else z
73      z = 180 if z > 180 else z
74
75      arm.set_position((x,y,z),50)
76
77  time.sleep_ms(50)
```

Firstly, obtain the coordinate data of the target color on screen, as the figure shown below:

```
56      center_x = color_data[0]
57      center_y = color_data[1]
```

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.

```
59      if abs(center_x - 160) < 15:
60          center_x = 160
61      x_pid.SetPoint = 160
62      x_pid.update(center_x)
63      x -= x_pid.output
64      x = 100 if x > 100 else x
65      x = -100 if x < -100 else x
```

The specific calculation method is as follow:

```
61      x_pid.SetPoint = 160
62      x_pid.update(center_x)
63      x -= x_pid.output
```

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

```
64      x = 100 if x > 100 else x
65      x = -100 if x < -100 else x
```

The same calculation method for y and z axes.

```
67      if abs(center_y - 120) < 5:
68          center_y = 120
69      z_pid.SetPoint = 120
70      z_pid.update(center_y)
71      z += z_pid.output
72      z = 100 if z < 100 else z
73      z = 180 if z > 180 else z
```

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,

```
75      arm.set_position((x,y,z),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 movement time and its unit is millisecond.