Lesson 3 Mask Recognition

Note: Please refer to "Lesson 1 Image Classification Learning" to burn mask recognition model into WonderCam AI vision module.

If do not finish this step, please burn the models first according to "7. AI Vision Game/ Python Development/ Lesson 5 Mask Recognition/ Lesson 1 Image Classification" otherwise, you will fail to start this game.

1. Project Principle

When WonderCam recognizes a human wearing a mask, dot matrix module will display a smiley face, and when it recognizes a face without a mask, the dot matrix module displays a cross mark.

The function of image classification is used to identify the ID number with the highest confidence, and then judge whether it is a human face and whether the face is wearing a mask, and finally combine with the dot matrix module to set the prompt feedback.

The path of the program file: "7. AI Vision Game/ Python Development/ Sensor-extension Game/ Program Files/ Mask Recognition/ main.py".

```
25 if __name__ == '__main__':
26
27     arm.go_home()
28     buzzer.setBuzzer(100)
29     nozzle.set_angle(0,1000)
30     time.sleep_ms(2000)
31
32     tm.update_display()
33
34     cross_buf = [0x0,0x0,0x0,0x81,0xc3,0xe7,0x7e,0x3c,0x3c,0x7e,0xe7,0xc3,0x81,0x0,0x0
35     smiling_buf = [0x0,0xc,0x2,0x19,0x21,0x42,0x80,0x80,0x80,0x80,0x42,0x21,0x19,0x2,
36     result_data = []
37
38     while True:
39         cam.update_result()
40         result_data.append(cam.most_likely_id())
41         if len(result_data) == 5:
42             result = sum(result_data) / 5.0
43             result_data = []
44             if result == 2.0:
45                 tm.update_display(smiling_buf)
46             elif result == 3.0:
47                 tm.update_display(cross_buf)
48             else:
49                 tm.update_display()
50
51         time.sleep(0.05)
```

2. Preparation

2.1 Hardware

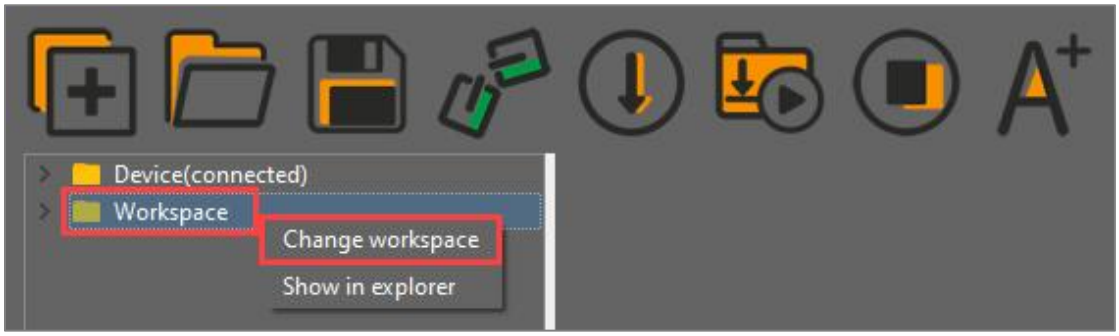
Please refer to the tutorial in “7. AI Vision Game/ WonderCam AI Vision Module Learning (Must Read!!)/ Lesson 3 WonderCam Assembly” and “Lesson 2 Module Assembly” in the same directory to assemble WonderCam and dot matrix modules to the corresponding position on MaxArm.

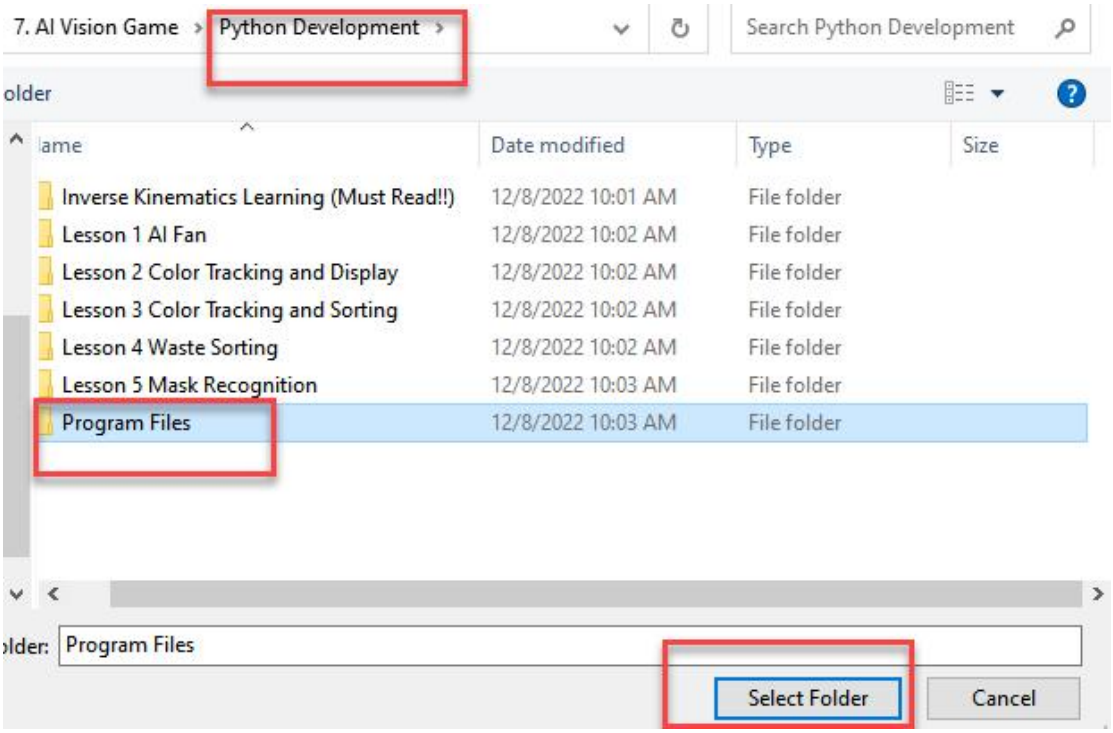
2.2 Software

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

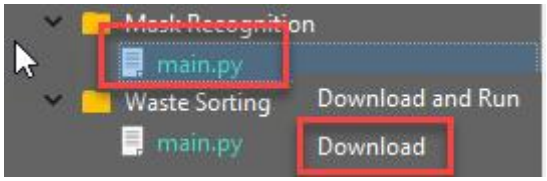
1) After connecting, change the path of Workspace to “7. AI Vision Game/ Python Development”, and select “Program Files”.



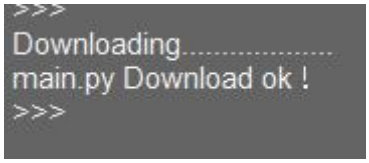


Then double click “Driver Library” folder, and select all library files, and right click and select “Download” to download the library files to the controller. (If you have downloaded this library file before, please skip this step.)

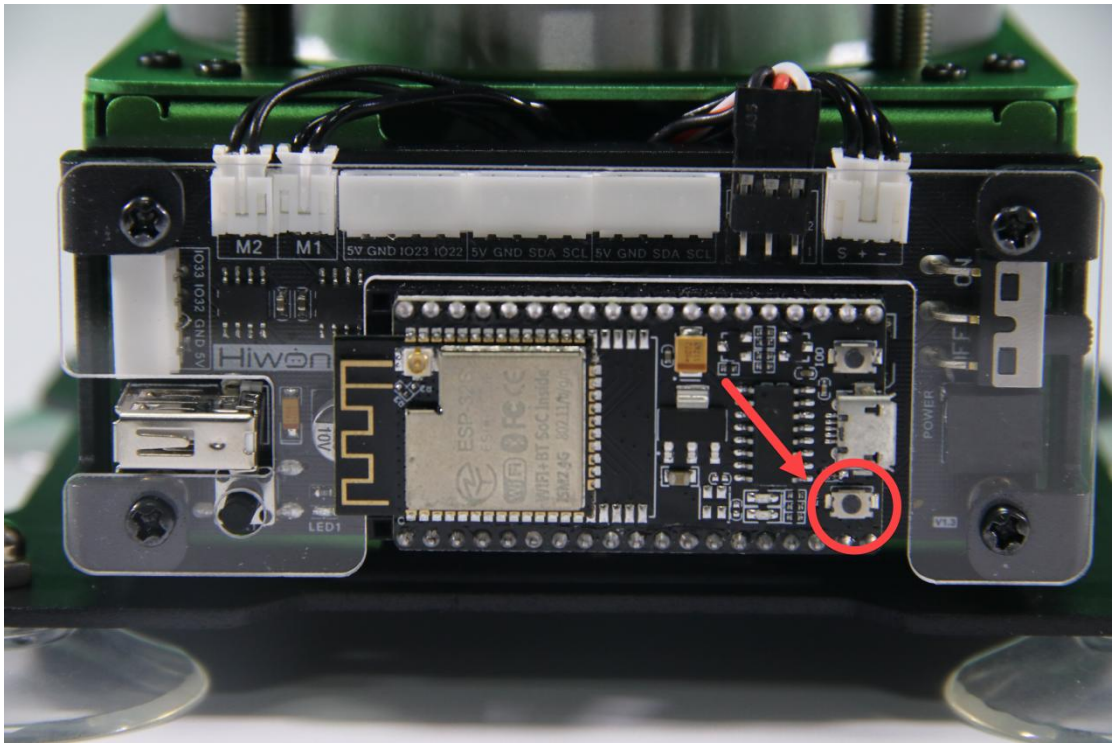
Then double click “Mask Recognition” folder and select “main.py” folder. Then right click and select “Download” to download all the program files to the controller.



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



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



4. Project Outcome

After program is downloaded, turn on robotic arm. At this time, LED dot matrix is off.

When detecting a face wearing no mask, the dot matrix module will display "✕". After wearing a mask, the matrix module will display a smiley face.

5. Program Instruction

The path of the program file: “7. AI Vision Game/ Python Development/ Sensor-extension Game/ Program Files/ Mask Recognition/main.py”

You can find the backup program file in Appendix if the original program file is modified.

5.1 Import Function Library and Initialization

Firstly, import the dot matrix library, WonderCam driver library, buzzer library, PWM servo library and other related function libraries for mask recognition.

```
main.py
1  import time
2  import TM1640
3  from machine import Pin, I2C
4  from WonderCam import *
5  from Buzzer import Buzzer
6  from espmax import ESPMax
7  from PWMServo import PWMServo
8  from BusServo import BusServo
9  from SuctionNozzle import SuctionNozzle
10
```

Next, instantiate library files.

```
14 pwm = PWMServo()
15 buzzer = Buzzer()
16 pwm.work_with_time()
17 bus_servo = BusServo()
18 arm = ESPMax(bus_servo)
19 nozzle = SuctionNozzle()
20 tm = TM1640.TM1640(clk=Pin(33), dio=Pin(32))
21 i2c = I2C(0, scl=Pin(16), sda=Pin(17), freq=400000)
22 cam = WonderCam(i2c)
23 cam.set_func(WONDERCAM_FUNC_CLASSIFICATION)
```

Then, initialize robotic arm.

```
if __name__ == '__main__':
    arm.go_home()
    buzzer.setBuzzer(100)
    nozzle.set_angle(0,1000)
    time.sleep_ms(2000)

    tm.update_display()

    cross_buf = [0x0,0x0,0x0,0x81,0xc3,0xe7,0x7e,0x3c,0x3c,0x7e,0xe7,0xc3,0x81,0x0,0x
    smiling_buf = [0x0,0xc,0x2,0x19,0x21,0x42,0x80,0x80,0x80,0x80,0x42,0x21,0x19,0x2,
    result_data = []
```

Among them,

cross_buf = [ ] arrya stores the data of the displayed cross mark

smiling_buf = [ ] array stores the data of the displayed smiley expression.

result_data = [ ] array stores the data of the detection result.

5.2 Image classification Recognition

Call cam.update_result() function to update the detection result, call cam.most_likely_id() function to get the maximum ID detected and stored in result_data (the ID for mask wearing is 2.0 and for no mask is 3.0).

```
while True:
    cam.update_result()
    result_data.append(cam.most_likely_id())
```

In order to prevent wrong detection, we take the average value after 5 times of detection and assign it to class_id to determine whether the person detected is wearing a mask or not, at this time the value of class_id is (2.0, 3.0, not equal to 2.0 or 3.0).

```
    if len(result_data) == 5:
        result = sum(result_data) / 5.0
        result_data = []
```

If result == 2.0, the face detected wears a mask. Then call the update_display() function in the TM1640.h library file to display a smiley face on dot matrix module.

```
44        if result == 2.0:
45            tm.update_display(smiling_buf)
```

If class_id == 3.0, it means the person detected does not put on a mask. At this time, dot matrix module display “✕”.

```
45            tm.update_display(smiling_buf)
46        elif result == 3.0:
```

Otherwise, it is assumed that the background is detected and the dot module is cleared.

```
48        else:
49            tm.update_display()
50
51    time.sleep(0.05)
```

6. Function Extension

In the program, three display states of the LED dot matrix module are set, which are off, smiley, and cross. If want to change the display pattern, you can refer to the following operations:

The LED dot matrix module has 8 rows and 16 columns, each column represents an 8-bit binary number, when it is 0, the LED is off, when it is 1, the LED is on. To facilitate writing, we will convert it into hexadecimal numbers to record "cross_buf[16] = { 0x0, 0x0, 0x0, 0x81, 0xc3, 0xe7, 0x7e, 0x3c, 0x3c, 0x7e, 0xe7, 0xe7, 0xc3, 0x81, 0x0, 0x0, 0x0, 0x0 };". Take displaying “✕” for example, as shown in the following table, coded from the bottom up.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0x0	0x0	0x0	0x81	0xc3	0xe7	0x7e	0x3c	0x3c	0x7e	0xe7	0xc3	0x81	0x0	0x0	0x0

0	0	0	1	1	1	0	0	0	0	1	1	1	0	0	0
0	0	0	0	1	1	1	0	0	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	1	1	1	0	0	1	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0	1	1	1	0	0	0

Here is an example of changing the ✕ displayed by cross_buf to a sad expression. The specific steps are shown below.

1) First calculate the hexadecimal numbers of the sad expressions, which are { 0x0, 0x0, 0x2, 0x2, 0x22, 0x12, 0x8, 0x8, 0x8, 0x8, 0x12, 0x22, 0x2, 0x2, 0x0, 0x0 } as shown in the table below, coded from the bottom up.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0x0	0x0	0x2	0x2	0x22	0x12	0x8	0x8	0x8	0x8	0x12	0x22	0x2	0x2	0x0	0x0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Find the following code.

```
34  cross_buf = [0x0,0x0,0x0,0x81,0xc3,0xe7,0x7e,0x3c,0x3c,0x7e,0xe7,0xc3,0x81,0x0,0x0,0x0]
35  smiling_buf = [0x0,0xc,0x2,0x19,0x21,0x42,0x80,0x80,0x80,0x80,0x42,0x21,0x19,0x2,0xc,0x0]
36  result_data = []
37
```

2) Use “#” to comment it out and input “cross_buf = [0x0, 0x0, 0x2, 0x2, 0x22, 0x12, 0x8, 0x8, 0x8, 0x8, 0x12, 0x22, 0x2, 0x2, 0x0, 0x0]” in the next line.

```
32  tm.update_display()
33
34
35  cross_buf = [0x0, 0x0, 0x2, 0x2, 0x22, 0x12, 0x8, 0x8, 0x8, 0x8, 0x12, 0x22, 0x2, 0x2, 0x0, 0x0]
36  smiling_buf = [0x0,0xc,0x2,0x19,0x21,0x42,0x80,0x80,0x80,0x80,0x42,0x21,0x19,0x2,0xc,0x0]
37
```

3) After inputting, press “Ctrl+S” to save program and refer to “3. Program Download” to upload. After the game starts, when a human face wearing no mask is detected, the module will display a sad expression.