

Lesson 3 AI Fan

NOTE: The face models have been burnt to WonderCam in advance, you can start this game directly.

If fail to recognize face, please re-burn the firmware according to “WonderCam Module Learning/3. Appendix/4. Firmware (Installation method included)”.

1. Project Principle

In this game, when face is detected by WonderCam, fan module will turn on and dot matrix module will display a smile expression, and then robotic arm will keep tracking the face,

The tracking function is implemented by PID algorithm which controls robotic arm to move within a certain area.

PID control algorithm combines Proportional (P), Integral (I), and Derivative (D). The controller attempts to correct the error between a measured process variable and desired setpoint by calculating the difference.

Based on the position information obtained in face recognition, and the coordinates of the center point of the vision module are compared to calculate the distance between robot arm and target position.

The path of program file: “7. AI Vision Game/ Python Development/ Program Files/ AI Fan/main.py”.

```

49     tm.write(smiling_buf) # 表情显示笑脸
50
51     center_x = face_data[0]
52     center_y = face_data[1]
53
54     if abs(center_x - 160) < 15: # X轴PID算法追踪
55         center_x = 160
56         x_pid.SetPoint = 160
57         x_pid.update(center_x)
58         x -= x_pid.output
59         x = 100 if x > 100 else x # 机械臂x轴范围限幅
60         x = -100 if x < -100 else x
61
62     if abs(center_y - 120) < 5: # Y轴PID算法追踪
63         center_y = 120
64         z_pid.SetPoint = 120
65         z_pid.update(center_y)
66         z += z_pid.output
67         z = 100 if z < 100 else z # 机械臂z轴范围限幅
68         z = 180 if z > 180 else z
69
70     arm.set_position((x,y,z),50) # 驱动机械臂
71
72     else: # 未识别到人脸
73         if fan_st:
74             fan.off() # 关闭风扇电机
75             fan_st = False
76             tm.update_display() # 点阵屏显示
77
78     time.sleep_ms(50) # 延时50ms

```

2. Preparation

2.1 Hardware

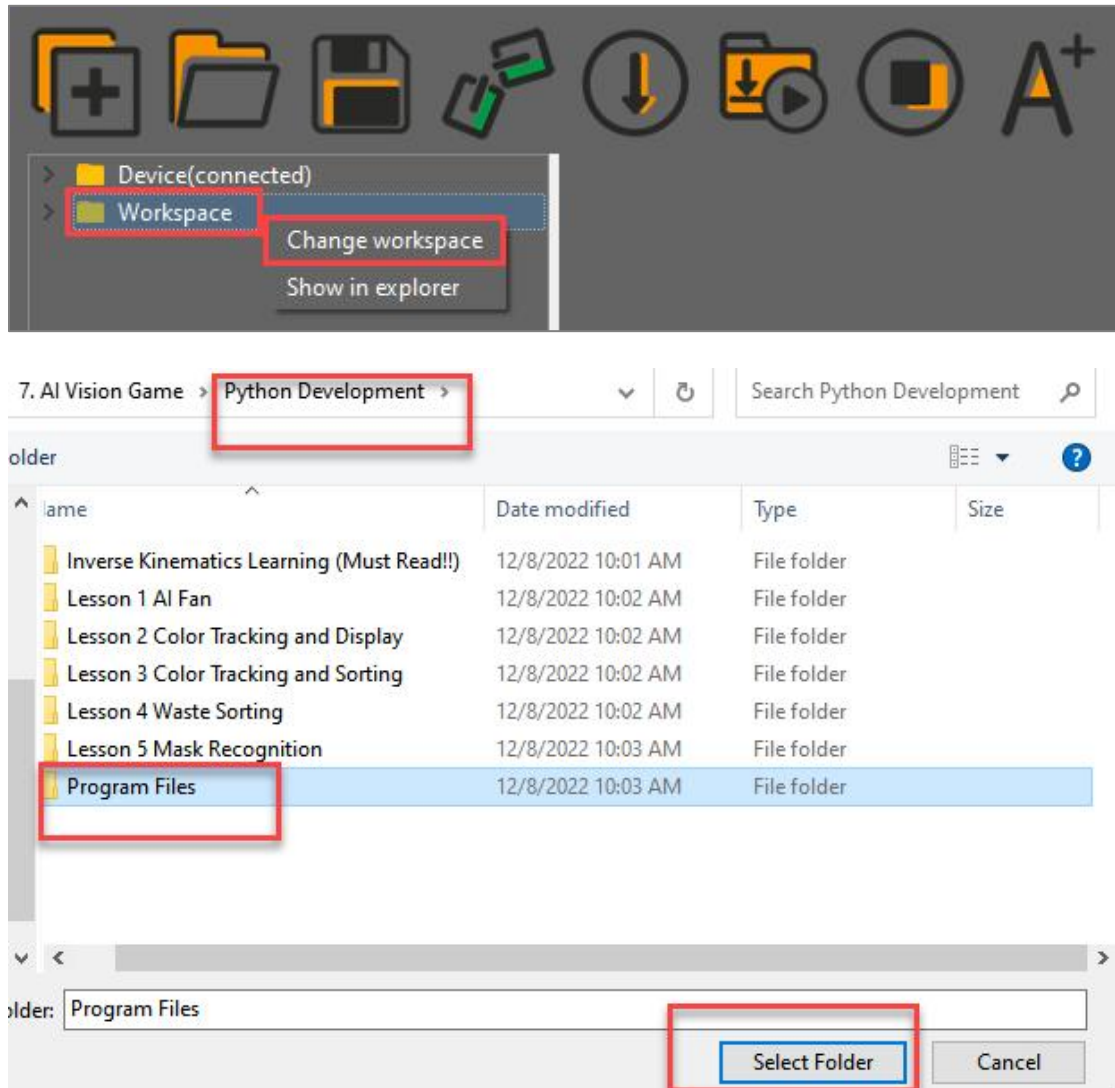
Please refer to “Lesson 2 Module Assembly” under the same directory to assemble fan module to the corresponding position on MaxArm (WonderCam AI vision module has been assembled by default).

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

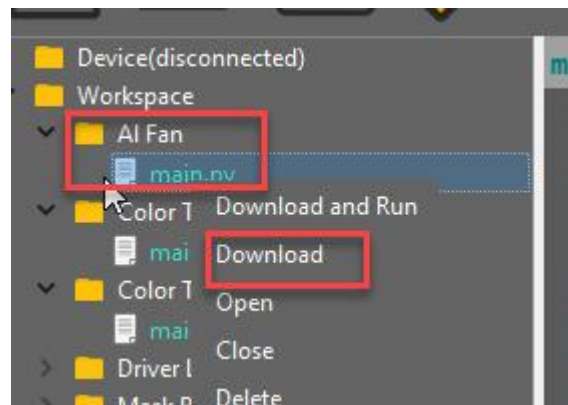
1) After connecting, change the path of Workspace to “7. AI Vision Game/Python Development”, and select “Program Files”.



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.)



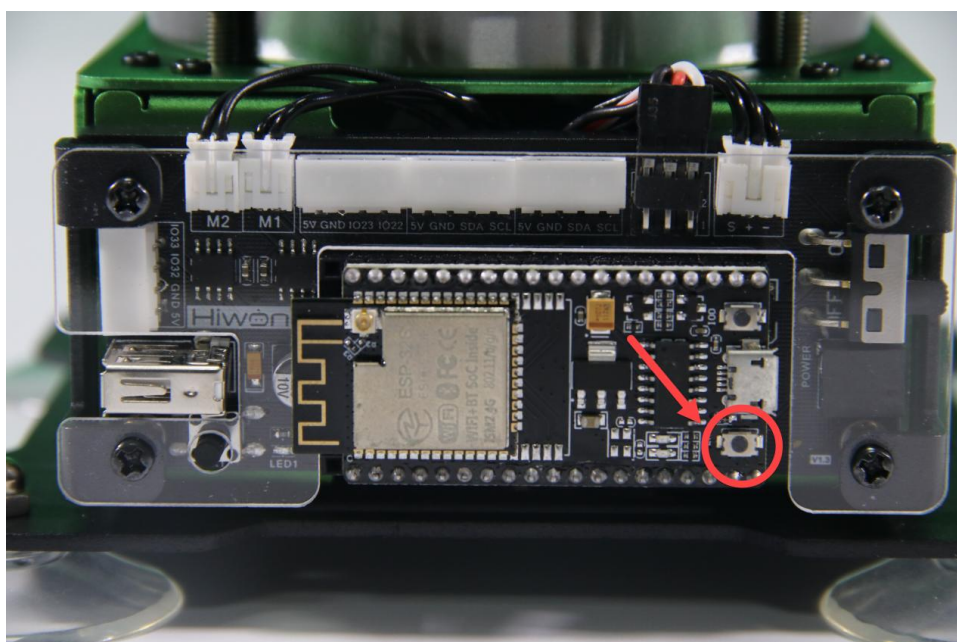
3) Then double click “AI Fan” folder and select “main.py” folder. Then right click and select “Download” to download all the program files to the controller.



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

```
>>>
Downloading.....
main.py Download ok !
>>>
```

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



4. Project Outcome

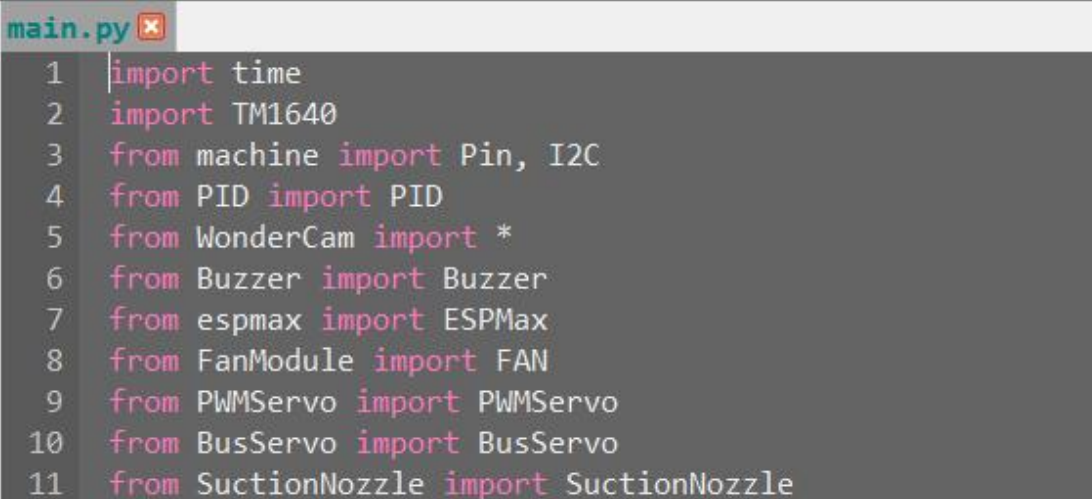
After the game is started, MaxArm keeps searching human face within a certain range. When a human face is detected, fan will turn on and a smile expression will display on dot matrix module. If the human face disappears, the fan will stop and dot matrix module will go black.

5. Program Analysis

5.1 Import function library

The path of program file: "7. AI Vision Game/ Python Development/ Program Files/ AI Fan/main.py".

Before the program is executed, I2C protocol, recognition module, buzzer, inverse kinematics, PWM servo, bus servo, air pump and other related Python libraries are required to be imported first.



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

5.2 Initialization

Initialize the required module and set WonderCam vision module as facial recognition mode.


```

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

```

```

28 if __name__ == '__main__':
29     fan.off()
30     fan_st = False
31     x, y, z = 0, -120, 150
32     buzzer.setBuzzer(100)
33     nozzle.set_angle(0,1000)
34     arm.set_position((x, y, z), 2000)
35     time.sleep_ms(2000)
36     x_pid = PID(0.026, 0.001, 0.0008)
37     z_pid = PID(0.030, 0.001, 0.0001)
38     tm.update_display()
39
40     smiling_buf = [0x0,0xc,0x2,0x19,0x21,0x42,0x80,0x80,0x80,0x80,0x42,0x21,0x19,0x2,0xc,0x0]

```

The program uses the `fan.off()` function to turn off fan and set a status for the fan which is true when turned on and false when turned off. The values of x, y, and z axes of the initial position are 0, -120, 150 respectively, as the figure shown below:

```

29 fan.off()
30 fan_st = False
31 x, y, z = 0, -120, 150

```

Next, use `setBuzzer()` function to control buzzer. The parameter 100 represents the sound duration is 100ms, as the figure shown below.

```

32 buzzer.setBuzzer(100)

```

The use `nozzle.set_angle()` function to control the rotation angel of the suction nozzle, as the figure shown below:

```

33 nozzle.set_angle(0,1000)

```

Among them, the first parameter "0" represents the rotation angle of the

suction nozzle is 0.

The second parameter “1000” represents the duration of rotation.

Then use the inverse kinematics function `arm.set_position ()` to control robot arm to move, as the figure shown below:

```
34 arm.set_position((x, y, z), 2000)
```

The first parameter “(x, y, z)” represents the coordinate of end effector, which are calculated by PID algorithm. The specific calculation method refers to “5.5 Search Human Face”.

The second parameter “2000” represents the duration of rotation.

Then use `x_pid = PID ()` and `z_pid = PID()` to calculate the offset value of end effector so that robot arm can constantly adjust its position and makes the recognized face coincide with the center of screen, thus calculate the coordinate of the human face, as the figure shown below:

```
36 x_pid = PID(0.026, 0.001, 0.0008)
37 z_pid = PID(0.030, 0.001, 0.0001)
```

The use `tm.update_display()` function to clear the screen of the dot matrix module, as the figure shown below:

```
38 tm.update_display()
```

A smiley face is displayed on dot matrix module by setting address character.

```
40 smiling_buf = [0x0,0xc,0x2,0x19,0x21,0x42,0x80,0x80,0x80,0x80,0x42,0x21,0x19,0x2,0xc,0x0]
```

5.3 Recognize Human Face

Robot arm keeps rotating to recognize a human face through vision module, and updating the module data.

The judge the returned data, as the figure shown below:

```
44 face_data = cam.get_face(1, 2)
```

Among them, the first parameter “1” represents that the human face recognized has been learned by vision module.

Among them, the first parameter “1” represents that the human face recognized has not learned by vision module.

If a human face is recognized, then determine whether the fan is turned off. If it is off, use the fan.on() function to turn on fan and display smile expression on the dot matrix module.

```
42 = while True:
43     cam.update_result()
44     face_data = cam.get_face(1, 2)
45     if face_data:
46         if not fan_st:
47             fan.on()
48             fan_st = True
49             tm.write(smiling_buf)
```

Compared with the data returned by vision module, if no human face is detected, the fan will turn off and the screen of dot matrix will go black through the tm.update_display() function, as the figure shown below:

```
72 = else:
73     if fan_st:
74         fan.off()
75         fan_st = False
76         tm.update_display()
77
78     time.sleep_ms(50)
```

5.4 Search Human Face

Use PID algorithm to set the x, y and z values of end-effector to search human face to realize the tracking function.


```

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

```

Firstly, obtain the coordinate of the human face on vision module, as the figure shown below:

```

51     center_x = face_data[0]
52     center_y = face_data[1]

```

Calculate the coordinates of the human face 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 robot arm and the face coordinate.

```

54     if abs(center_x - 160) < 15:
55         center_x = 160

```

The specific calculation method is as follow:

```
56 x_pid.SetPoint = 160
57 x_pid.update(center_x)
58 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 the x and z.

```
59 x = 100 if x > 100 else x
60 x = -100 if x < -100 else x
```

The calculation method of y and z axes is the same.

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

```
34 arm.set_position((x, y, z), 2000)
```

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

The second parameter “2000” represents the movement time.