Lesson 3 Color Tracking and Sorting

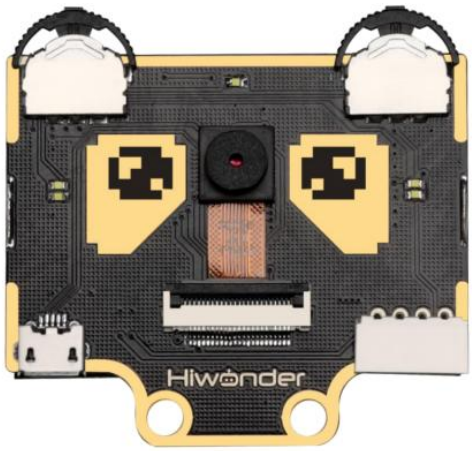
Note: Before starting this game, please make sure the colors including red, green, blue to be recognized are learned as ID1, ID2 and ID3 in sequence with WonderCam module. The specific content refer to "Lesson 1 Color Recognition Learning" under the same directory. If you skip the step above, WonderCam can not recognize the colors and complete this game.

Please refer to the tutorial in folder "7. AI Vision Game/ Python Development/ Lesson 3 Color Tracking and Sorting/ Lesson 2 WonderCam Assembly (Top view) " to assemble WonderCam module to MaxArm".

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/ Python Development/ Program Files/ Color Tracking and Sorting/main.py

```
while True:
    cam.update_result()
    if cam.get_color_blob(1):
        color_num = 1
        color_data = cam.get_color_blob(1)
    elif cam.get_color_blob(2):
        color_num = 2
        color_data = cam.get_color_blob(2)
    elif cam.get_color_blob(3):
        color_num = 3
        color_data = cam.get_color_blob(3)
    else:
        color_num = 0
        color_data = None

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

2. Preparation

2.1 Hardware

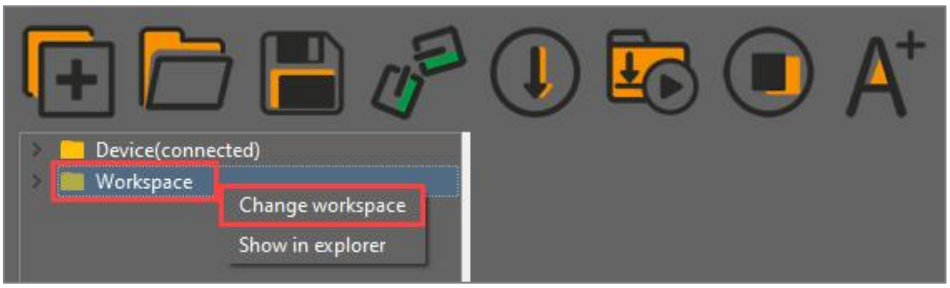
Please refer to “Lesson 2 WonderCam Assembly (Top view)” in folder “7. AI Vision Game/ Python Development/ Lesson 3 Color Tracking and Sorting” to assemble WonderCam to the corresponding position on MaxArm.

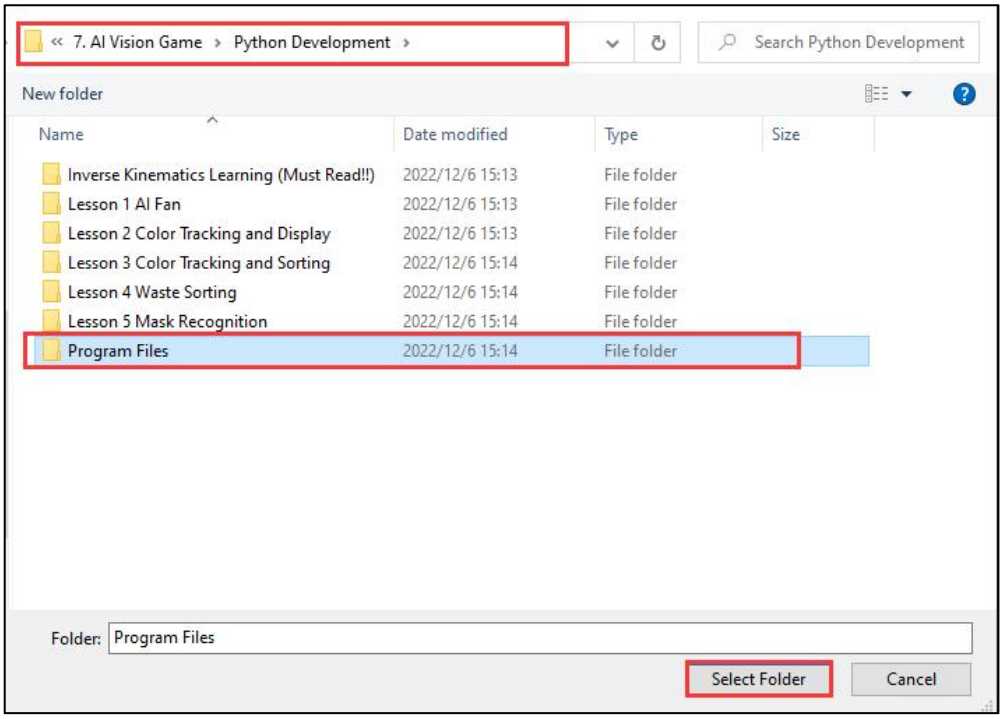
2.2 Software

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

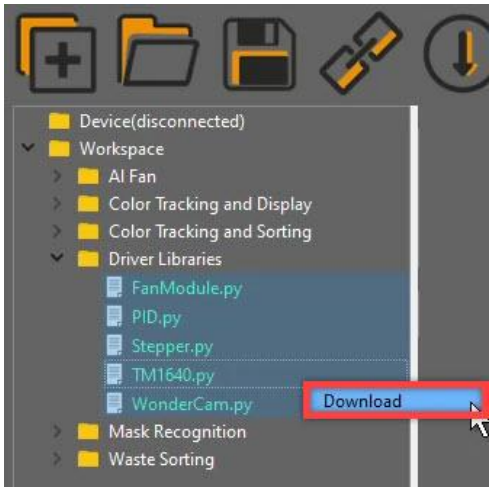
3. Program Download

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

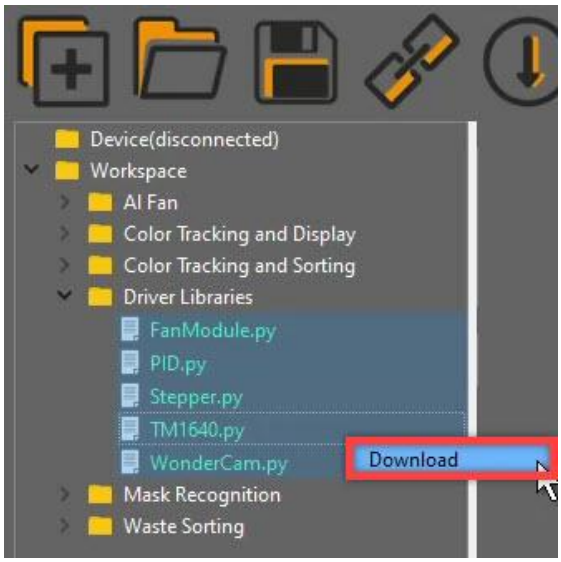




2) Then double click “Drive Library” folder, and select all library files, and right click and select “Download” to download the library files to the controller. (If you have downloaded this library file before, please skip this step.)

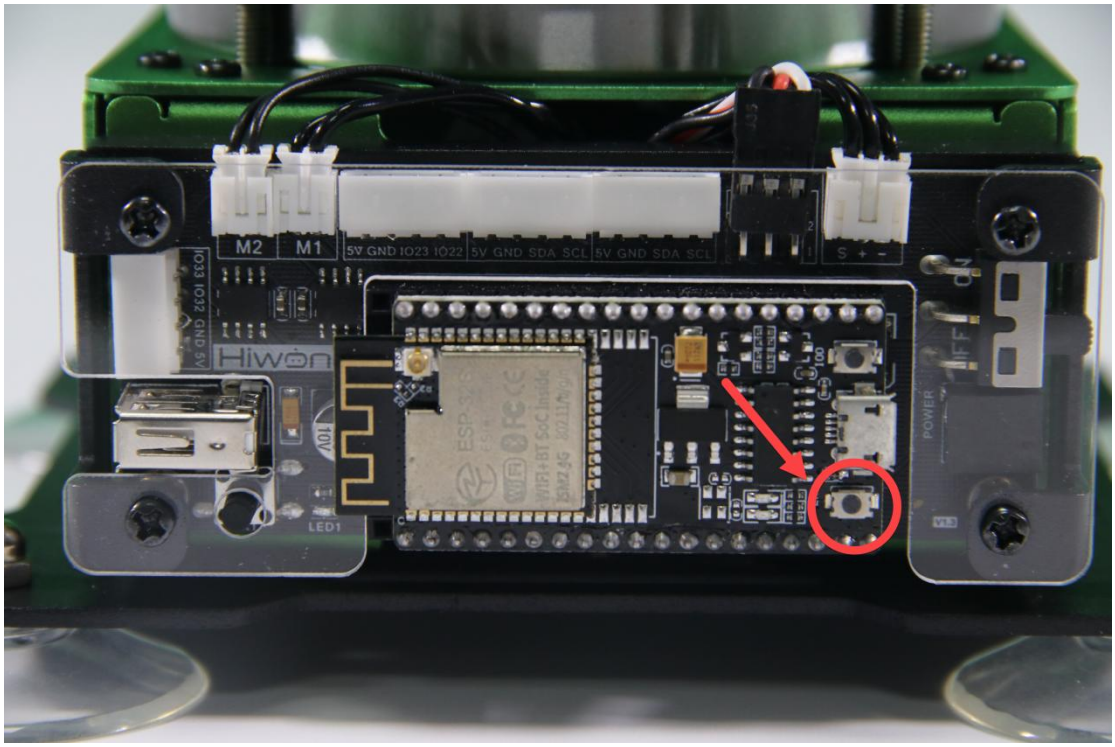


3) Then double click “Color Tracking and Sorting” folder and select “main.py” folder. Then right click and select “Download” to download all the program files to the controller.



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

5) 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. 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/ Python Development/ Program Files/ Color Tracking and Sorting/main.py”.

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

```
import time
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()
i2c = I2C(0, scl=Pin(16), sda=Pin(17), freq=400000)
cam = WonderCam(i2c)
cam.set_func(WONDERCAM_FUNC_COLOR_DETECT)
cam.set_led(False)
```

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：

```
    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(0.08, 0.003, 0.0003)
    y_pid = PID(0.08, 0.003, 0.0003)
```

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:

```
while True:
    cam.update_result()
    if cam.get_color_blob(1):
        color_num = 1
        color_data = cam.get_color_blob(1)
    elif cam.get_color_blob(2):
        color_num = 2
        color_data = cam.get_color_blob(2)
    elif cam.get_color_blob(3):
        color_num = 3
        color_data = cam.get_color_blob(3)
    else:
        color_num = 0
        color_data = None
```

After the color is recognized, the buzzer will sound and the recognized color will be displayed on terminal, as the figure shown below:

```
                buzzer.setBuzzer(100)
                if color_num == 1:
                    print('color: red')
                    angle = 45
                    (place_x, place_y, place_z) = (-120,-140,85)
                elif color_num == 2:
                    print('color: green')
                    angle = 62
                    (place_x, place_y, place_z) = (-120,-80,85)
                elif color_num == 3:
                    print('color: blue')
                    angle = 90
                    (place_x, place_y, place_z) = (-120,-20,85)
```

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:

```
54  if abs(center_x - 160) < 15:
55      center_x = 160
56  x_pid.SetPoint = 160
57  x_pid.update(center_x)
58  dx = x_pid.output
59  x -= dx
60  x = 100 if x > 100 else x
61  x = -100 if x < -100 else x
62
63  if abs(center_y - 120) < 5:
64      center_y = 120
65  y_pid.SetPoint = 120
66  y_pid.update(center_y)
67  dy = y_pid.output
68  y -= dy
69  y = -60 if y > -60 else y
70  y = -200 if y < -200 else y
71
72  arm.set_position((x,y,z),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:

```
50  if color_data:
51      center_x = color_data[0]
52      center_y = color_data[1]
```

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.

```
54      if abs(center_x - 160) < 15:
55          center_x = 160
56      x_pid.SetPoint = 160
57      x_pid.update(center_x)
58      dx = x_pid.output
59      x -= dx
60      x = 100 if x > 100 else x
61      x = -100 if x < -100 else x
```

The same calculation method for the value of z and y axes is shown below.

```
63      if abs(center_y - 120) < 5:
64          center_y = 120
65      y_pid.SetPoint = 120
66      y_pid.update(center_y)
67      dy = y_pid.output
68      y -= dy
69      y = -60 if y > -60 else y
70      y = -200 if y < -200 else y
```

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,

```
72      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 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:

```
95              d_y = (68-abs(d_x/3))
96              arm.set_position((x+d_x,y-d_y,100),1000)
97              time.sleep_ms(1000)
98              arm.set_position((x+d_x,y-d_y,86),600)
99              nozzle.on()
100             time.sleep_ms(1000)
101             arm.set_position((x+d_x,y-d_y,150),1000)
102             time.sleep_ms(1000)
103             arm.set_position((place_x,place_y,150),1500)
104             nozzle.set_angle(angle,1500)
105             time.sleep_ms(1500)
106             arm.set_position((place_x,place_y,place_z),1000)
107             time.sleep_ms(1200)
108             nozzle.off()
109             arm.set_position((place_x,place_y,150),1000)
110             time.sleep_ms(1000)
111
112             x, y, z = 0, -120, 150
113             arm.set_position((x, y, z), 2000)
114             nozzle.set_angle(0,1800)
115             time.sleep_ms(2000)
```

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

```
74          if abs(dx) < 0.1 and abs(dy) < 0.1:
```

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

```
75              i += 1
76              if i > 10:
77                  i = 0
```

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.

```
81              angle = 45
82              (place_x, place_y, place_z) = (-120,-140,85)
```

The specific sorting process is as follow:

```
95              d_y = (68-abs(d_x/3))
96              arm.set_position((x+d_x,y-d_y,100),1000)
97              time.sleep_ms(1000)
98              arm.set_position((x+d_x,y-d_y,86),600)
99              nozzle.on()
100             time.sleep_ms(1000)
101             arm.set_position((x+d_x,y-d_y,150),1000)
102             time.sleep_ms(1000)
103             arm.set_position((place_x,place_y,150),1500)
104             nozzle.set_angle(angle,1500)
105             time.sleep_ms(1500)
106             arm.set_position((place_x,place_y,place_z),1000)
107             time.sleep_ms(1200)
108             nozzle.off()
109             arm.set_position((place_x,place_y,150),1000)
110             time.sleep_ms(1000)
111
112             x, y, z = 0, -120, 150
113             arm.set_position((x, y, z), 2000)
114             nozzle.set_angle(0,1800)
115             time.sleep_ms(2000)
```

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

```
94              d_x = x/2.3
95              d_y = (68-abs(d_x/3))
```

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

```
96              arm.set_position((x+d_x,y-d_y,100),1000)
```

Among them:

The first parameter “x+d_x” represents x position.

The second parameter “y-d_y” represents y position.

The third parameter “100” represents z position.

The fourth parameter “1000” is the movement time and its unit is ms.

The decrease the value of z axis, e.i., z=86, to lower robotic arm.

```
98          arm.set_position((x+d_x,y-d_y,86),600)
```

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

```
99          nozzle.on()
```

Increase the value of z axis, e,i. z=150 to raise robotic up, as the figure shown below:

```
101         arm.set_position((x+d_x,y-d_y,150),1000)
```

Then place the block to the corresponding position, as the figure shown below:

```
106         arm.set_position((place_x,place_y,place_z),1000)
```

Then robot arm returns to the initial position for the next detection

```
112         x, y, z = 0, -120, 150
113         arm.set_position((x, y, z), 2000)
114         nozzle.set_angle(0,1800)
115         time.sleep_ms(2000)
```

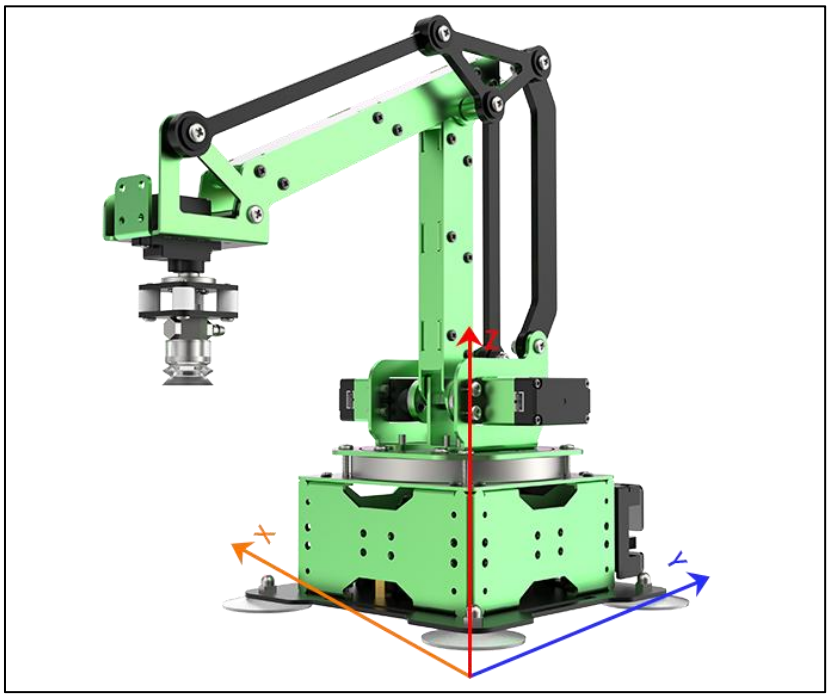
6. Function Extension

By default, the corresponding red, green and blue blocks are arranged closely in a row. Here we take the example of increasing the placement distance between the blocks and modify the position of the red, green and blue blocks after they are recognized and sucked.

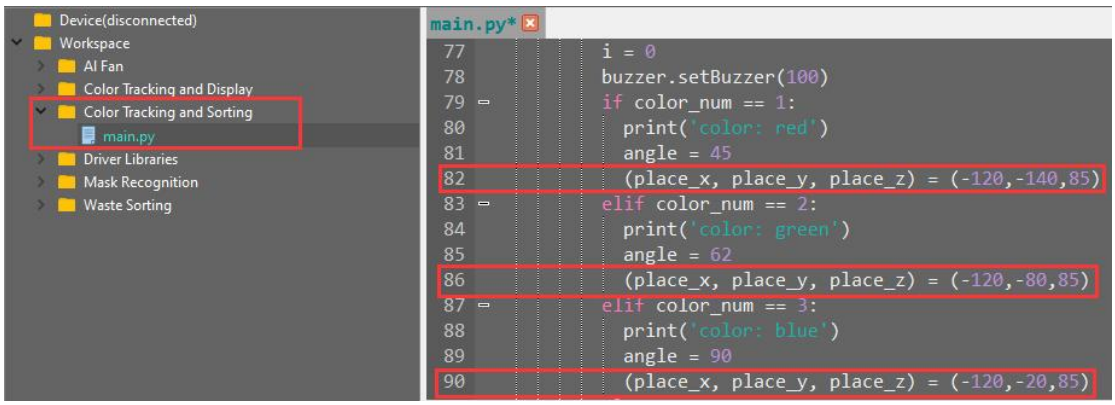
First of all, the xyz coordinate system described here and their corresponding coordinates all take your own as the first person view. The positive direction of the x-axis corresponds to the right side of the robot arm, the positive direction of the y-axis is the rear of the robot arm, and the positive direction of the z-axis is the top of the robot arm.

Taking x-axis as an example, the similar expressions about the robot arm moving to the right and so on mentioned below actually correspond to the robot arm moving to the negative direction of x-axis, and we will use this as the basis to adjust its position.

The specific coordinate system representation can be referred to the following figure.



Please connect the port according to steps 1-2 in “2.Preparation and 3. Program Download”. Then open the corresponding file and download library file, and find the code in “main.py”, as the figure shown below:



Take the codes in red box for example. place_x、place_y、place_z presents the coordinate of x, y and z axes of block placement position.

Modify the codes in red box to the following codes :

```
82              (place_x, place_y, place_z) = (-120,-140,85)
83          elif color_num == 2:
84              print('color: green')
85              angle = 62
86              (place_x, place_y, place_z) = (-120,-80,85)
87          elif color_num == 3:
88              print('color: blue')
89              angle = 90
90              (place_x, place_y, place_z) = (-120,-20,85)
```

Take the ID1 block in the first red box of the above figure as an example, its place_x is modified from -120 to -150, which corresponds to the placement position of the block changed from -120 to -150 on x-axis, i.e. the robot arm moves to the right; place_y is modified from -140 to -200 to the placement position of the block changed from -140 to -200 on y-axis, i.e. the robot arm moves to the front, which plays a role of increasing the distance of the color block.

Finally, the placement position of red block is adjusted from (-120，-140，85) to (-150，-200，85).

If want to adjust the placement position of other blocks, you can refer to the following list to modify the corresponding coordinate of the block.

	Parameter	Rotation Direction
X axis	Increase	Move to left
	Decrease	Move to right
Y axis	Increase	Move backwards

	Decrease	Move forwards
Z axis	Increase	Move up
	Decrease	Move down

Note: The rotation direction takes your own as the first person view.

After modification, please proceed to the next step according to the steps 3-5 in “3. Program Download” to check the effect. Robotic arm will move with the block. When the block stops moving, robotic arm will suck and transfer the block to the corresponding coordinate position and there is a gap between the color blocks.

Red block（ID1）：(place_x, place_y, place_z) = (-120,-140,85)

——》 (place_x, place_y, place_z) = (-150,-200,85)

Green block（ID2）：(place_x, place_y, place_z) = (-120,-80,85)

——》 (place_x, place_y, place_z) = (-150,-100,85)

Blue block（ID3）：(place_x, place_y, place_z) = (-120,-20,85)

——》 (place_x, place_y, place_z) = (-150,-0,85)