Lesson 1 Color Recognition

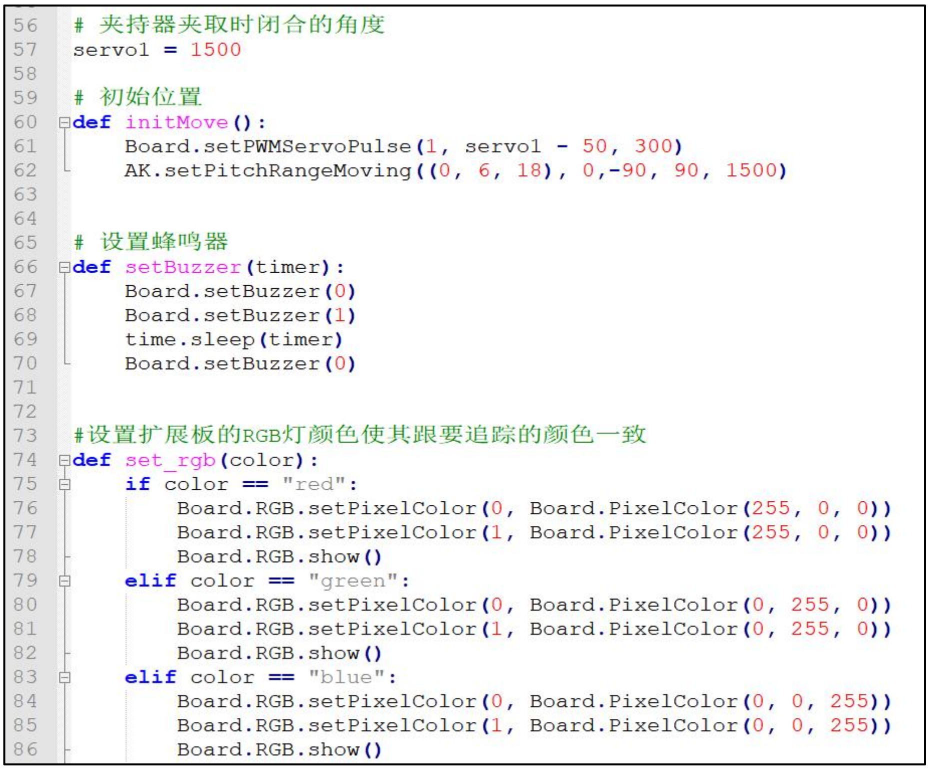
1. Working Principle

This lesson is divided into two parts which are color recognition and execution feedback after recognition.

For color recognition part, the color of the object is converted through LAB space, and then frame the outline of the target after processing the image.

After recognition, set servo, buzzer and RGB to make the robot perform corresponding feedback according to different colors.

The source code of the program is located in **/home/pi/MasterPi/Functions/ColorDetect.py**



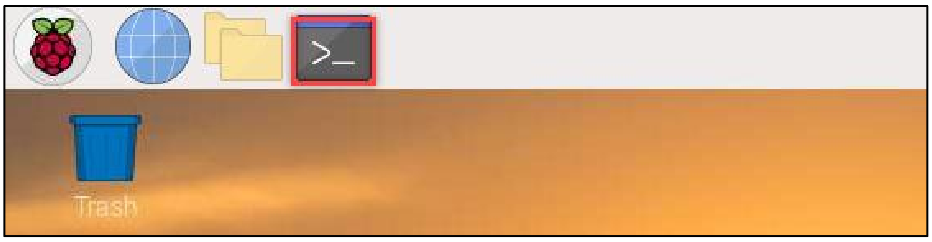
```python
# 夹持器夹取时闭合的角度
servo1 = 1500

# 初始位置
def initMove():
    Board.setPWMServoPulse(1, servo1 - 50, 300)
    AK.setPitchRangeMoving((0, 6, 18), 0,-90, 90, 1500)


# 设置蜂鸣器
def setBuzzer(timer):
    Board.setBuzzer(0)
    Board.setBuzzer(1)
    time.sleep(timer)
    Board.setBuzzer(0)


#设置扩展板的RGB灯颜色使其跟要追踪的颜色一致
def set_rgb(color):
    if color == "red":
        Board.RGB.setPixelColor(0, Board.PixelColor(255, 0, 0))
        Board.RGB.setPixelColor(1, Board.PixelColor(255, 0, 0))
        Board.RGB.show()
    elif color == "green":
        Board.RGB.setPixelColor(0, Board.PixelColor(0, 255, 0))
        Board.RGB.setPixelColor(1, Board.PixelColor(0, 255, 0))
        Board.RGB.show()
    elif color == "blue":
        Board.RGB.setPixelColor(0, Board.PixelColor(0, 0, 255))
        Board.RGB.setPixelColor(1, Board.PixelColor(0, 0, 255))
        Board.RGB.show()
```

2. Operation Steps

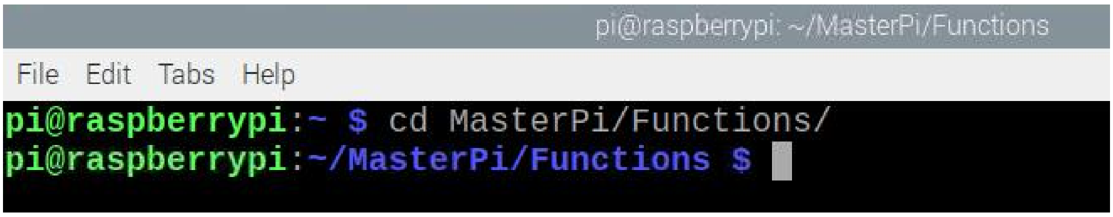
The entered command should be case sensitive.

Step 1: Turn on MaserPi, then connect to Raspberry Pi system desktop through VNC.

Step 2: Click >_ or press “Ctrl+Alt+T” to enter LX terminal.

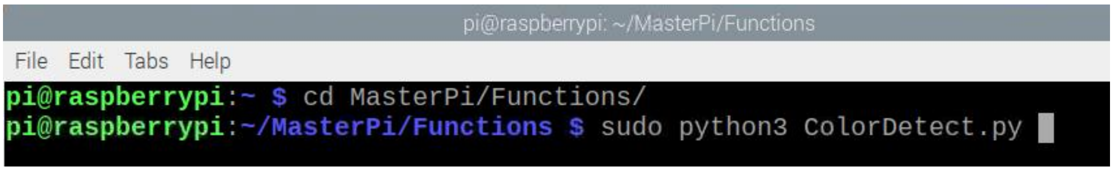


Step 3: Enter “cd MasterPi/Functions/” command, and then press “Enter” to come to the directory of games programmings.



```
pi@raspberrypi:~ $ cd MasterPi/Functions/
pi@raspberrypi:~/MasterPi/Functions $ 
```

Step 4: Enter “sudo python3 ColorDetect.py”, then press “Enter” to start the game.



```
pi@raspberrypi:~ $ cd MasterPi/Functions/
pi@raspberrypi:~/MasterPi/Functions $ sudo python3 ColorDetect.py 
```

Step 5: If you want to exit the game programming, press “Ctrl+C” in LX terminal interface. If the exit fails, please try it few more times.

3. Project Outcome

After the game starts, the robot will recognize colors and then perform corresponding feedback according to different colors as shown in the following table:

Object color	Buzzer	RGB light	Execution Action	The content printed by frame
Red	beep once	Red	“Nod”	red
Green	beep once	Green	“Shake head”	green
Blue	beep once	Blue	“Shake head”	blue

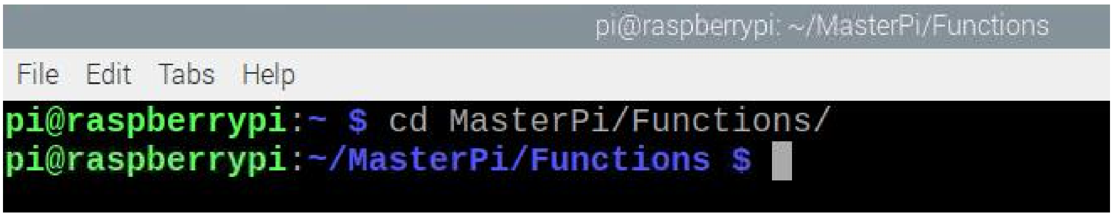
4. Function Extension

4.1 Modify Default Recognition Color

Red, green and blue are three built-in colors in the color recognition program and red is the default color. After the robot recognizes red object, it will execute nod action.

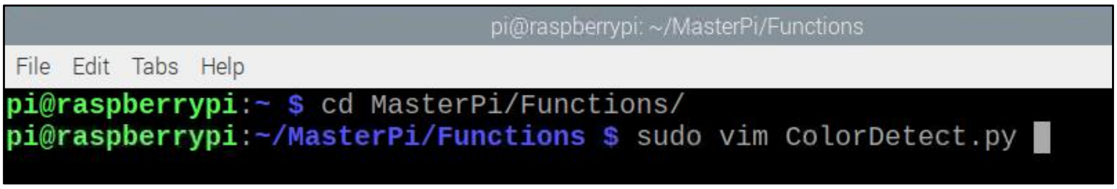
In the following steps, we’re going to modify the recognized color as green.

Step 1: Enter command “cd MasterPi/Functions/” and press “Enter ”to the directory of game programmings.



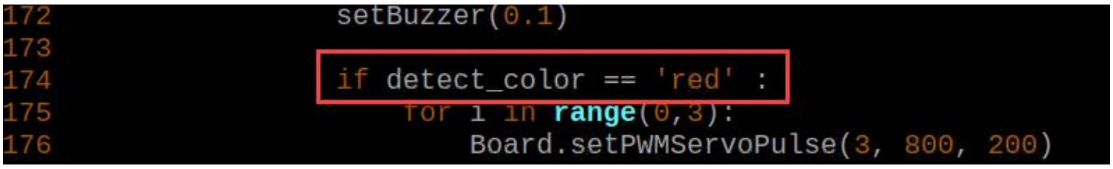
```
pi@raspberrypi:~ $ cd MasterPi/Functions/
pi@raspberrypi:~/MasterPi/Functions $ 
```

Step 2: Enter command “sudo vim ColorDetect.py”, and then press “Enter” to open program file.



```
pi@raspberrypi:~ $ cd MasterPi/Functions/
pi@raspberrypi:~/MasterPi/Functions $ sudo vim ColorDetect.py
```

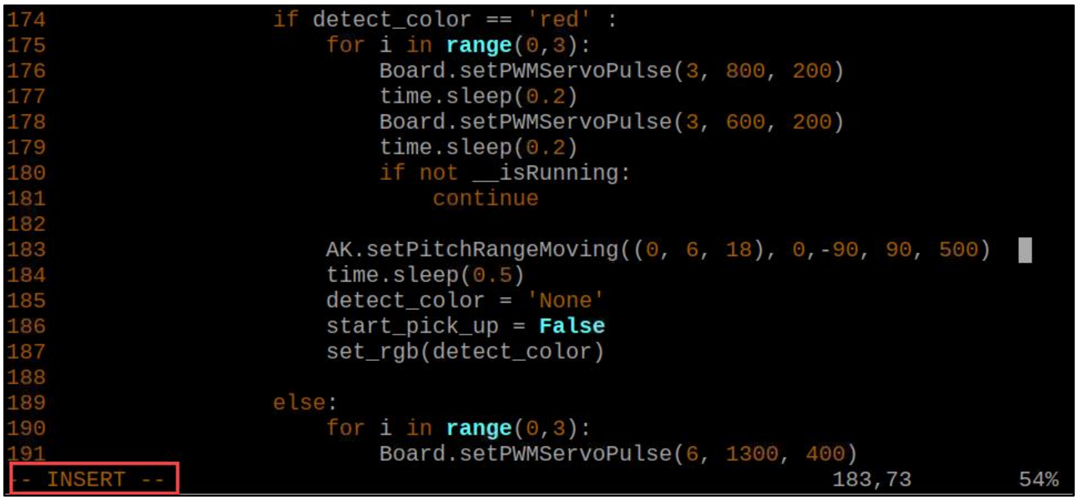
Step 3: Find the code shown in the following red box.



```
172                 setBuzzer(0.1)
173
174                 if detect_color == 'red' :
175                     for i in range(0,3):
176                         Board.setPWMServoPulse(3, 800, 200)
```

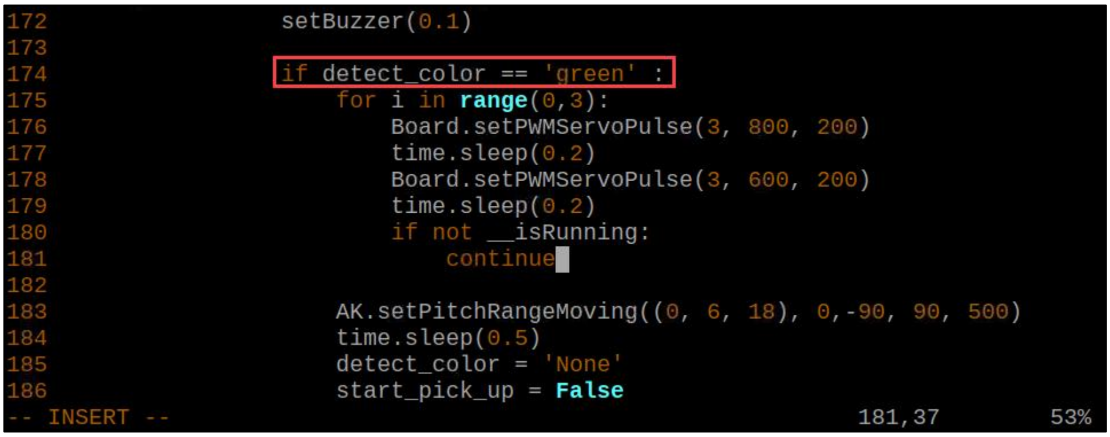
Note: After entering the position number of code, press “Shift+G” to jump to the corresponding position. (The position number of the code in figure is for reference only.)

Step 4: Press “i” to enter the editing mode.



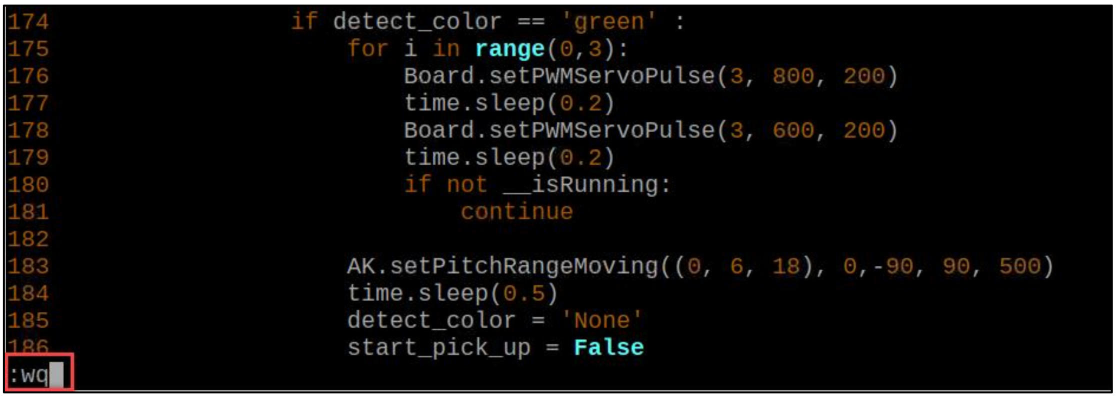
```
174                 if detect_color == 'red' :
175                     for i in range(0,3):
176                         Board.setPWMServoPulse(3, 800, 200)
177                         time.sleep(0.2)
178                         Board.setPWMServoPulse(3, 600, 200)
179                         time.sleep(0.2)
180                         if not __isRunning:
181                             continue
182
183                     AK.setPitchRangeMoving((0, 6, 18), 0,-90, 90, 500)
184                     time.sleep(0.5)
185                     detect_color = 'None'
186                     start_pick_up = False
187                     set_rgb(detect_color)
188
189                 else:
190                     for i in range(0,3):
191                         Board.setPWMServoPulse(6, 1300, 400)
-- INSERT --                                              183,73        54%
```

Step 5: Modify “red” in “detect_color == 'red'” to “green” as the figure shown below:



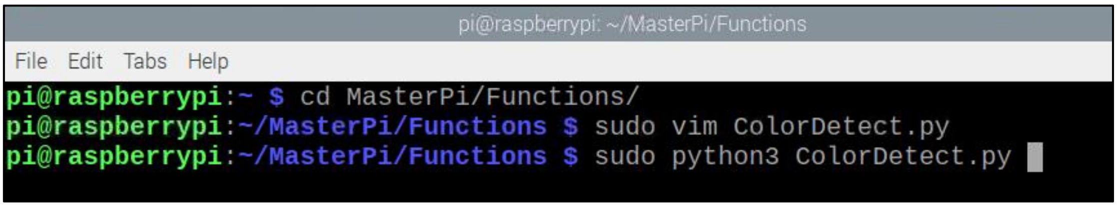
```
                setBuzzer(0.1)

                if detect_color == 'green' :
                    for i in range(0,3):
                        Board.setPWMServoPulse(3, 800, 200)
                        time.sleep(0.2)
                        Board.setPWMServoPulse(3, 600, 200)
                        time.sleep(0.2)
                        if not __isRunning:
                            continue

                    AK.setPitchRangeMoving((0, 6, 18), 0,-90, 90, 500)
                    time.sleep(0.5)
                    detect_color = 'None'
                    start_pick_up = False
-- INSERT --                                                   181,37        53%
```

Step 6: Then, save the modified content. Press "Esc", then enter ":wq" to save file and exit editor.



```
                if detect_color == 'green' :
                    for i in range(0,3):
                        Board.setPWMServoPulse(3, 800, 200)
                        time.sleep(0.2)
                        Board.setPWMServoPulse(3, 600, 200)
                        time.sleep(0.2)
                        if not __isRunning:
                            continue

                    AK.setPitchRangeMoving((0, 6, 18), 0,-90, 90, 500)
                    time.sleep(0.5)
                    detect_color = 'None'
                    start_pick_up = False
:wq
```

Step 7: Enter "sudo python3 ColorDetect.py" command again, and then press "Enter" to start color recognition.

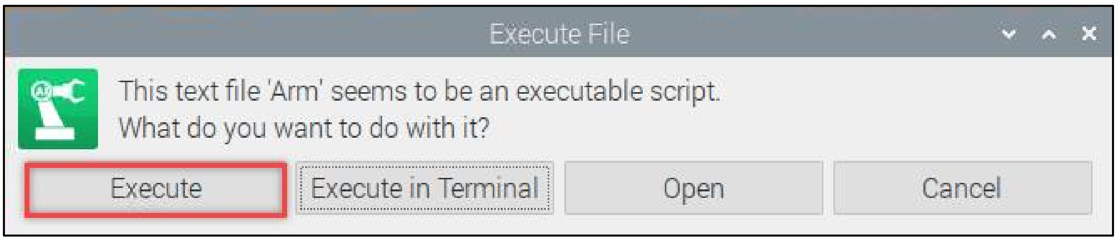


```
pi@raspberrypi:~ $ cd MasterPi/Functions/
pi@raspberrypi:~/MasterPi/Functions $ sudo vim ColorDetect.py
pi@raspberrypi:~/MasterPi/Functions $ sudo python3 ColorDetect.py
```

4.2 Add Recognized Color

In addition to the built-in recognized colors, you can set other recognized colors in the programming. Take purple as example:

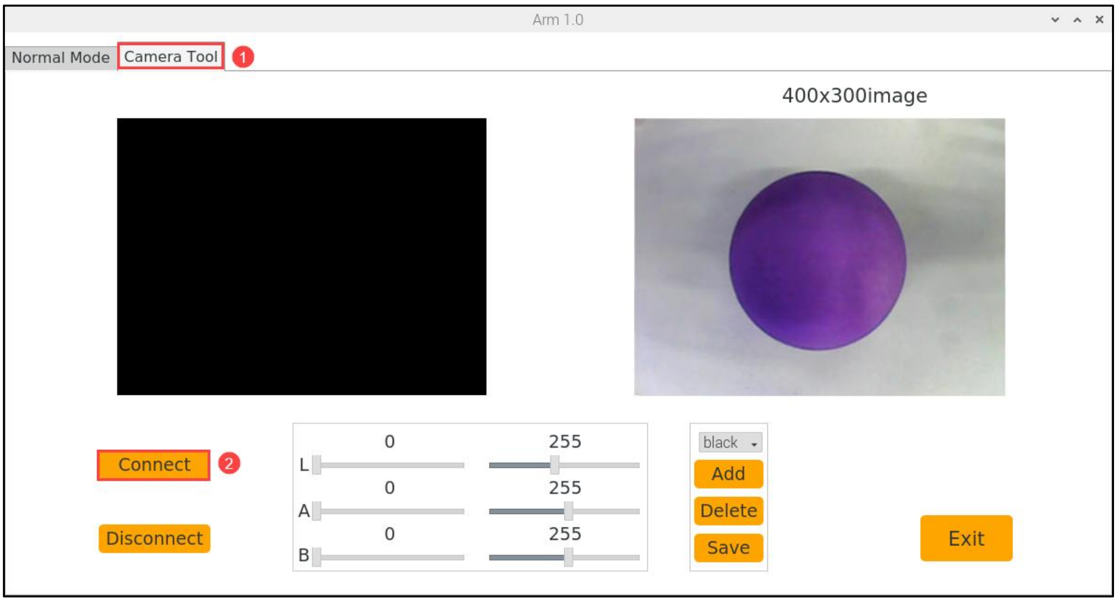
Step 1: Double-click ![Arm] on system desktop and then click "Execute" in the pops-up window.



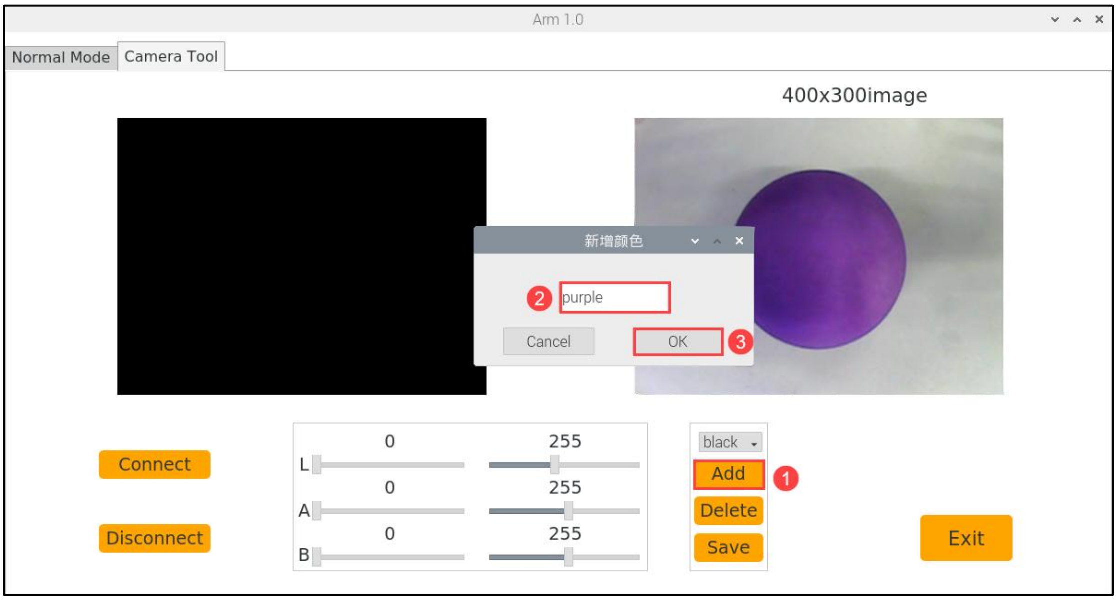
Execute File

This text file 'Arm' seems to be an executable script.
What do you want to do with it?

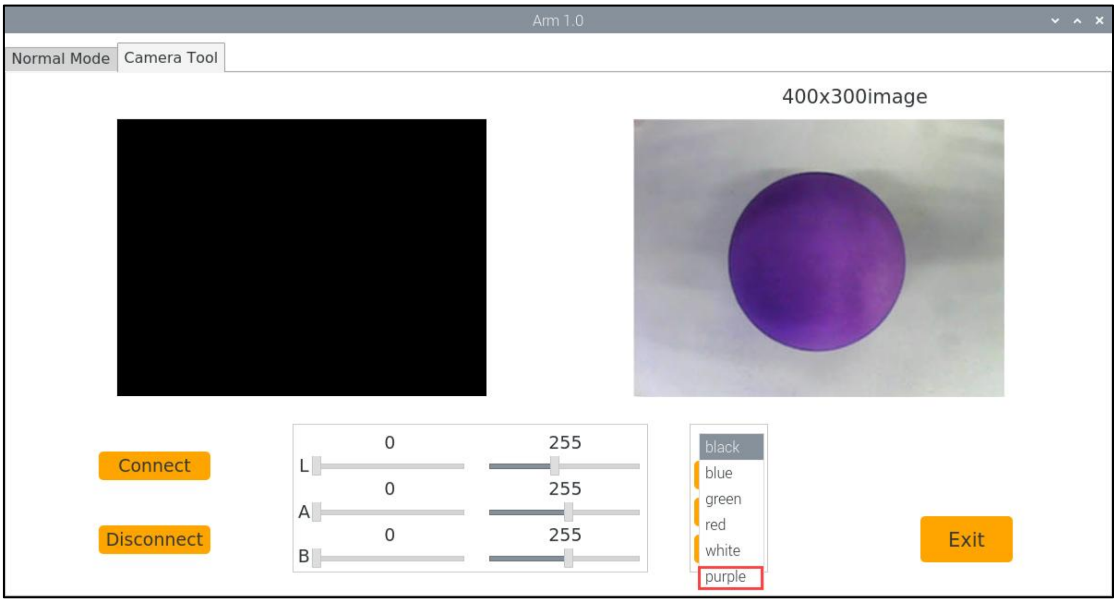
Step 2: In the pop-up interface, select "Camera Tool" and "Connect" in turn.



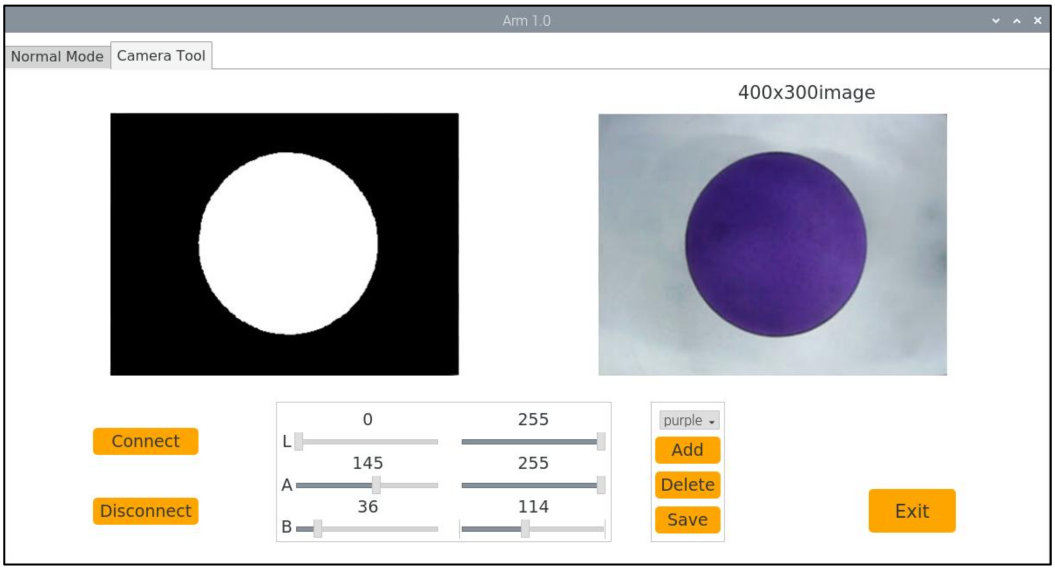
Step 3: Click "Add". Then name the added color (Take "purple" as an example) and click "OK".



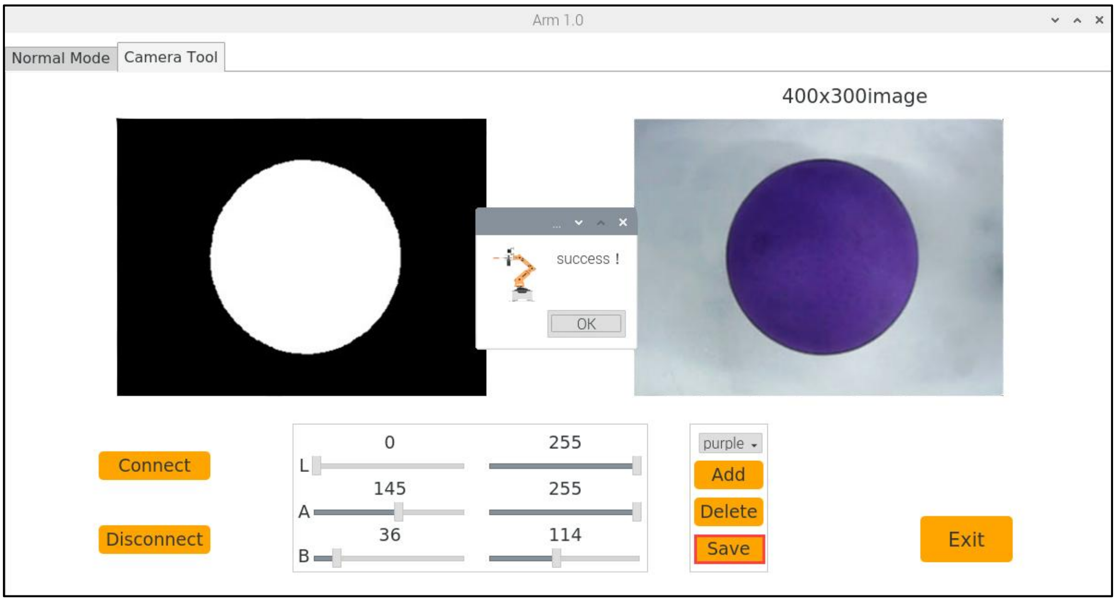
Step 4: Then select “purple” in the color potion bar.



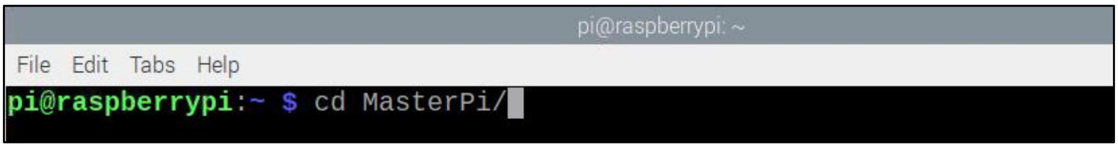
Step 5: Point the camera at the purple object. Drag the corresponding sliders of L, A, and B until the color area to be recognized becomes white and other areas become black.



Step 6: Click “Save” to save the adjusted color threshold.

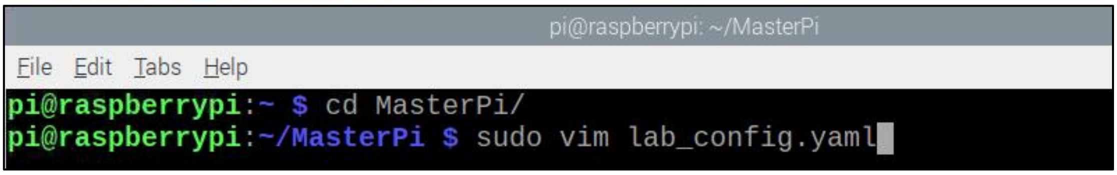


Step 7: After the modification is complete, check whether the modified data is written in successfully. Enter “cd MasterPi/” command and then press “Enter” to come to the directory where the program code is located.



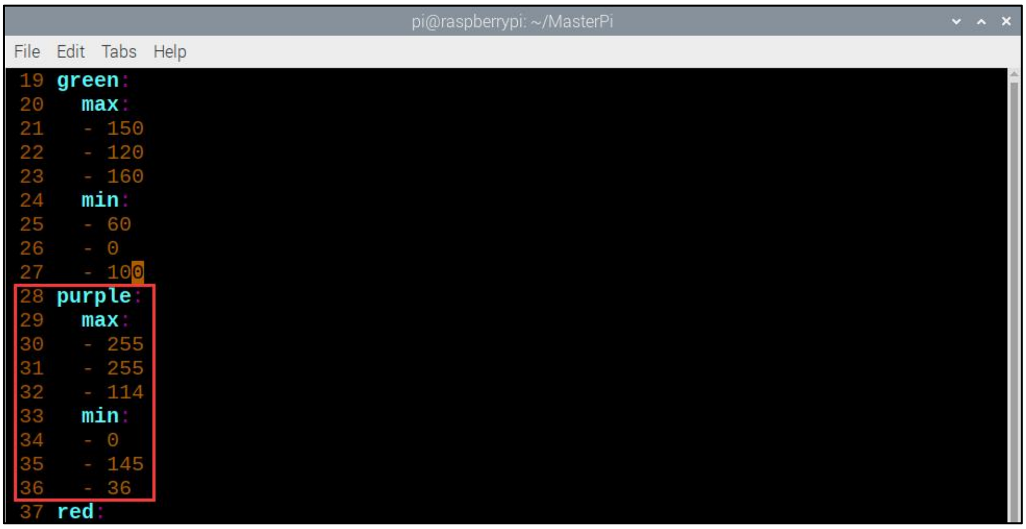
```
pi@raspberrypi:~ $ cd MasterPi/
```

Step 8: Enter “sudo vim lab_config.yaml” command, and then press “Enter” to open program file.



```
pi@raspberrypi:~ $ cd MasterPi/
pi@raspberrypi:~/MasterPi $ sudo vim lab_config.yaml
```

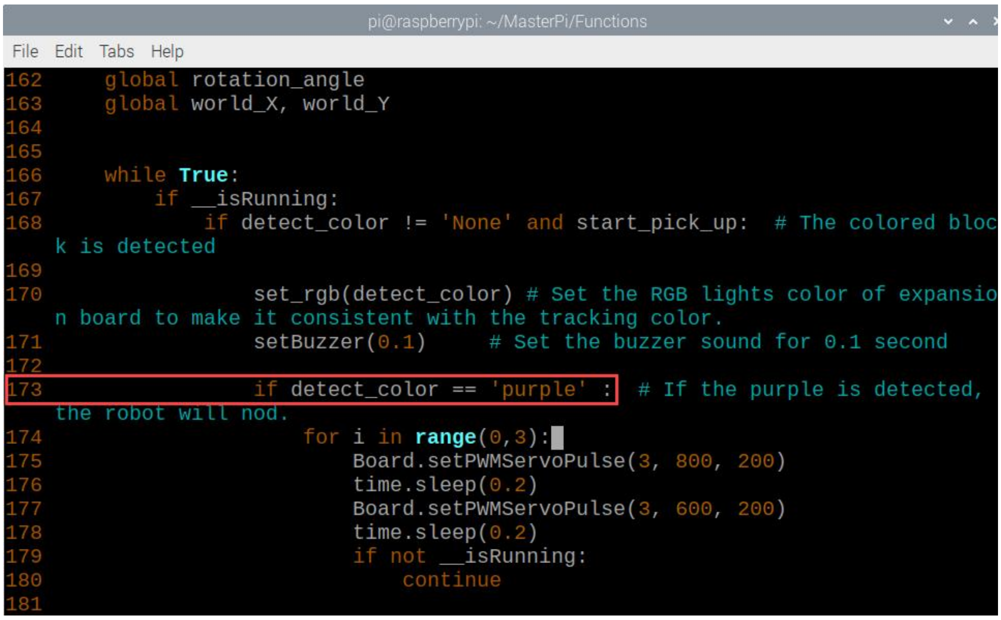
Step 9: After opening the color threshold program file, you can view the purple threshold parameters.



```
green:
  max:
  - 150
  - 120
  - 160
  min:
  - 60
  - 0
  - 100
purple:
  max:
  - 255
  - 255
  - 114
  min:
  - 0
  - 145
  - 36
red:
```

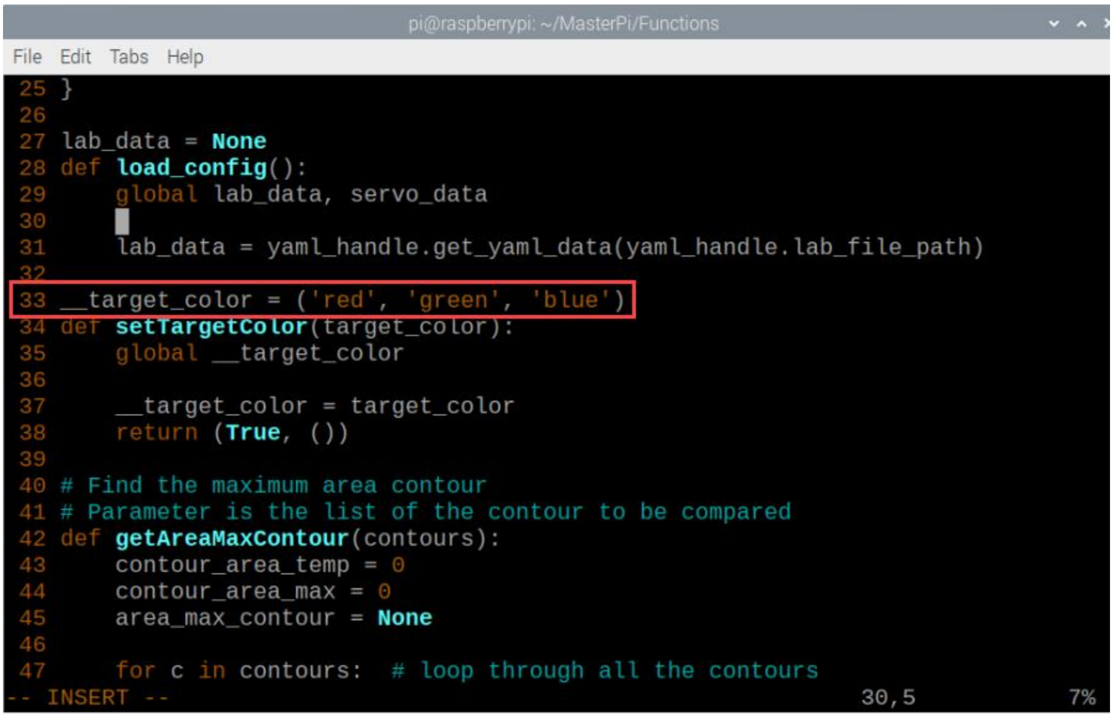
For game's performance, it's recommended to modify the value to the initial value by LAB_Tool after the modification is completed.

Step 10: According to the steps in "4.1 Modify recognized color", Modify "red" in "detect_color == 'red'" to "purple" as the figure shown below:



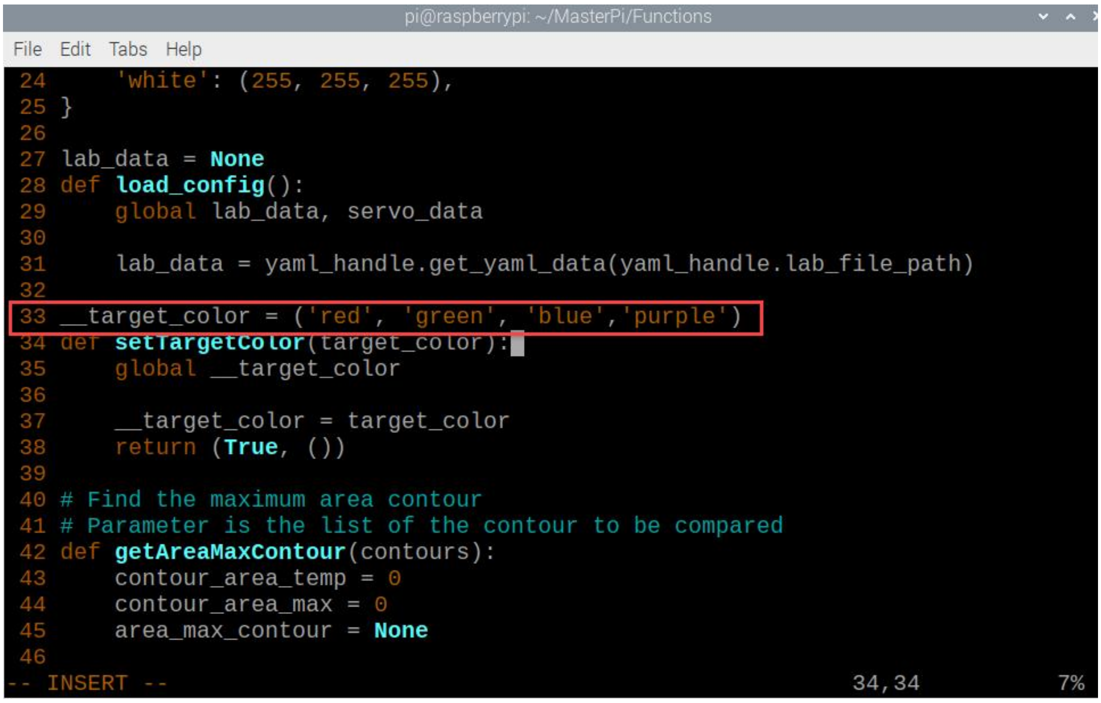
```
    global rotation_angle
    global world_X, world_Y


    while True:
        if __isRunning:
            if detect_color != 'None' and start_pick_up:  # The colored block is detected

                set_rgb(detect_color) # Set the RGB lights color of expansion board to make it consistent with the tracking color.
                setBuzzer(0.1)     # Set the buzzer sound for 0.1 second

                if detect_color == 'purple' :  # If the purple is detected, the robot will nod.
                    for i in range(0,3):
                        Board.setPWMServoPulse(3, 800, 200)
                        time.sleep(0.2)
                        Board.setPWMServoPulse(3, 600, 200)
                        time.sleep(0.2)
                        if not __isRunning:
                            continue

```

Step 11: Find the code shown in the following red box.



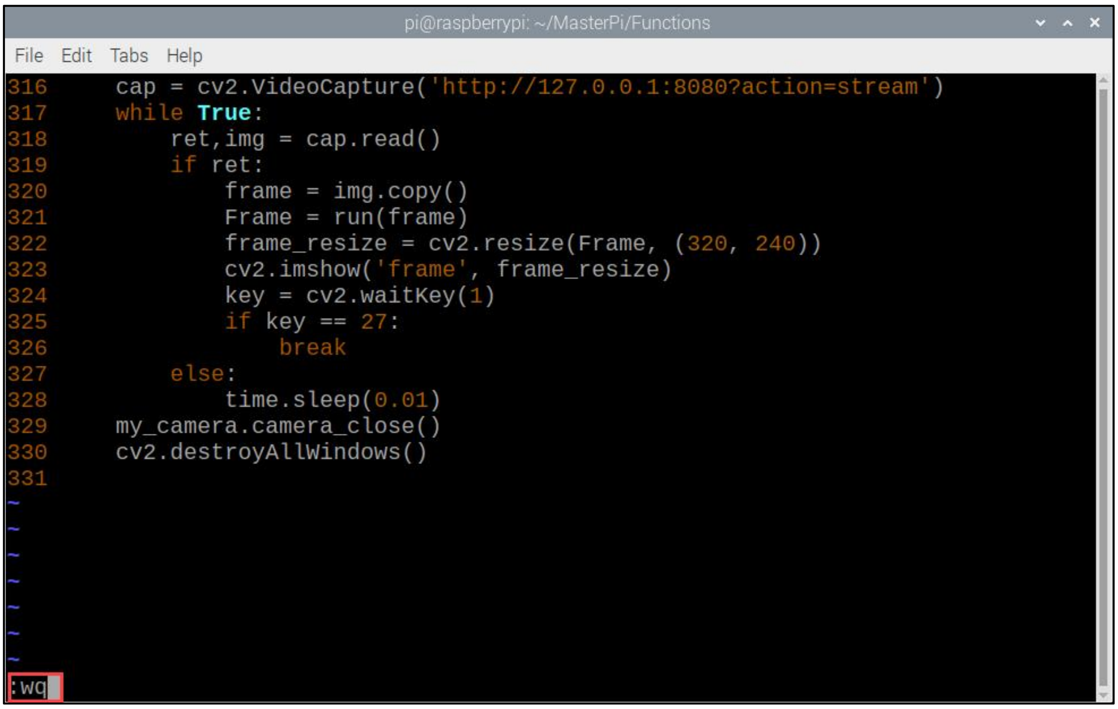
```
}

lab_data = None
def load_config():
    global lab_data, servo_data
    
    lab_data = yaml_handle.get_yaml_data(yaml_handle.lab_file_path)

__target_color = ('red', 'green', 'blue')
def setTargetColor(target_color):
    global __target_color

    __target_color = target_color
    return (True, ())

# Find the maximum area contour
# Parameter is the list of the contour to be compared
def getAreaMaxContour(contours):
    contour_area_temp = 0
    contour_area_max = 0
    area_max_contour = None

    for c in contours:  # loop through all the contours
-- INSERT --                                            30,5          7%
```

Step 12: Enter “purple” as the figure shown below:



```
    'white': (255, 255, 255),
}

lab_data = None
def load_config():
    global lab_data, servo_data

    lab_data = yaml_handle.get_yaml_data(yaml_handle.lab_file_path)

__target_color = ('red', 'green', 'blue','purple')
def setTargetColor(target_color):
    global __target_color

    __target_color = target_color
    return (True, ())

# Find the maximum area contour
# Parameter is the list of the contour to be compared
def getAreaMaxContour(contours):
    contour_area_temp = 0
    contour_area_max = 0
    area_max_contour = None

-- INSERT --                                            34,34         7%
```

Step 13: Save the modified content. Press “Esc”, and then enter “:wq”, and press “Enter” to save and exit.



```
    cap = cv2.VideoCapture('http://127.0.0.1:8080?action=stream')
    while True:
        ret,img = cap.read()
        if ret:
            frame = img.copy()
            Frame = run(frame)
            frame_resize = cv2.resize(Frame, (320, 240))
            cv2.imshow('frame', frame_resize)
            key = cv2.waitKey(1)
            if key == 27:
                break
        else:
            time.sleep(0.01)
    my_camera.camera_close()
    cv2.destroyAllWindows()

:wq
```

Step 14: Start the game again according to “2.Operation steps” and then place the purple object in front of the camera. You can find that the robot will perform “nod” action.

If you want to add other colors as recognizable color, please operate as the above steps.