

Lesson 2 Make a Color Recognition Alarm

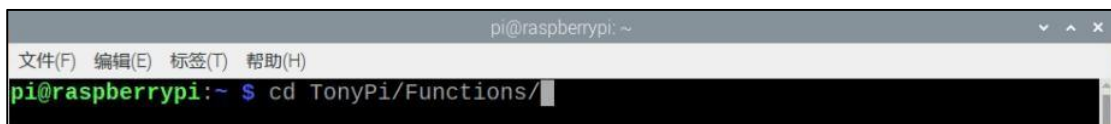
This section will program a project with OpenCV. When camera recognizes the set target color, the buzzer will alarm.

1. Operation Steps

1) Turn on TonyPi Pro and connect to VNC.

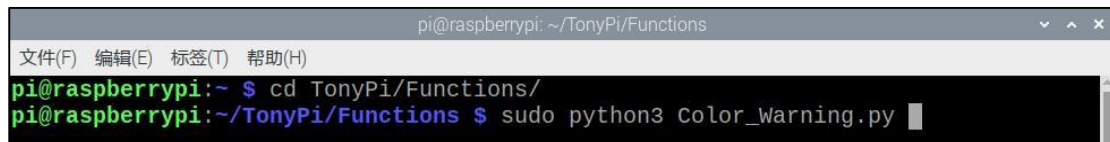
2) Press “Ctrl+Alt+T” or click  icon in the upper left corner to open LX terminal.

3) Enter “cd TonyPi/Functions/” command, and then press “Enter” to come to the category of games programmings.



```
pi@raspberrypi: ~  
文件(F) 编辑(E) 标签(T) 帮助(H)  
pi@raspberrypi:~ $ cd TonyPi/Functions/
```

4) Enter “sudo python3 Color_Warning.py” command, and then press “Enter” to start game.



```
pi@raspberrypi: ~/TonyPi/Functions  
文件(F) 编辑(E) 标签(T) 帮助(H)  
pi@raspberrypi:~ $ cd TonyPi/Functions/  
pi@raspberrypi:~/TonyPi/Functions $ sudo python3 Color_Warning.py
```

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

2. Project Outcome

After starting the game, the robot recognizes the red, and then the buzzer will make a “Beep” sound. If the green or blue is recognized, the buzzer will keep silent.

3. Project Analysis

Firstly, recognize the object color. Lab color space is used to process the image, and convert RGB color space into Lab color space. Next, obtain the contour only containing the target color after binarization, dilation, erosion, and etc. Then, frame the color contour with a circle to realize the color recognition.

◆ Recognize Multiple Colors

Previously, we have learned how to use OpenCV to recognize red and circle the object in the returned image, and then print the corresponding color information as a feedback. We will add two recognizable colors which are blue and green in the following.

Firstly, add the RGB value of these two colors.

```
range_rgb = {  
    'red': (0, 0, 255),  
    'blue': (255, 0, 0),  
    'green': (0, 255, 0),  
    'black': (0, 0, 0),  
    'white': (255, 255, 255),  
}
```

Then, through the color recognition method learned from the previous lesson, the image is processed to obtain the contour with maximum area. Then determine the circle color and the printed content according to the different target color.

```
else:  
    detect_color = 'None'  
    draw_color = range_rgb["black"]  
    cv2.putText(img, "Color: " + detect_color, (10, img.shape[0] - 10),  
cv2.FONT_HERSHEY_SIMPLEX, 0.65, draw_color, 2)
```

◆ Buzzer Alarm

In the custom “run” function, “setBuzzer” function can be called to control the buzzer. When the parameter of this function is 0, it means the buzzer is turned off. When it is 1, it means the buzzer is turned on. To prevent noise interference, we turn off the buzzer first and then turn it on. After a delay of 0.1 second, turn it off to achieve the effect of a short buzzer alarm

```
Board.setBuzzer(0) # Close  
  
Board.setBuzzer(1) # Open  
  
time.sleep(0.1) # Delay  
  
Board.setBuzzer(0) #Close
```

◆ Set Flag

Since we acquire the real-time image, Raspberry Pi will repeatedly detect and process. In this project, we only need to control the robot to make a sound when recognizing the target. Therefore, set a flag “beer” to determine whether the buzzer makes sound according to the flag status.

Firstly, define the global variable “beer” and assign it to “true”.

```
beer = True
```

In addition, the flag “beer” can be used after function is declared internally.

Otherwise, the program will report an error, as the figure shown below:

```
def run(img):  
    global beer
```

Then, judge whether the detected maximum area contour is the red part in “run” function and add a statement to control the buzzer. The program is as follows:

```
if color_area_max == 'red': #Red has the maximum area.
```

```
    detect_color = 'red'
```

```
    draw_color = range_rgb["red"]
```

```
    if beer == True:
```

```
        Board.setBuzzer(0) #Close
```

```
        Board.setBuzzer(1) # Open
```

```
        time.sleep(0.1) # Delay
```

```
        Board.setBuzzer(0) #Close
```

```
    beer = False
```

Note: After buzzer makes sound, do not forget to set the flag as “False”. Otherwise, the buzzer will keep alarming after recognizing the target color.

Finally, judge whether the detected maximum area contour is green or blue, and then turn on the flag switch of buzzer. At this time, even in an environment where three colors of red, green and blue exist at the same time, the robot can also make alarm when the color of the maximum area recognized is red.

```
if color_area_max == 'red': #Red has the maximum contour area
```

```
    detect_color = 'red'
```

```
    draw_color = range_rgb["red"] if
```

```
    beer == True:
```

```
        Board.setBuzzer(0) # Close
```

```
        Board.setBuzzer(1) # Open
```

```
        time.sleep(0.1) # Delay
```

```
        Board.setBuzzer(0) #Close
```

```
    beer = False
```

```
elif color_area_max == 'green': #Green has the maximum contour area
```

```
    beer = True
```

```
detect_color = 'green'
```

```
draw_color = range_rgb["green"]  
  
elif color_area_max == 'blue': #Blue has the maximum contour area.  
  
    beer = True  
  
    detect_color = 'blue'  
  
    draw_color = range_rgb["blue"]
```