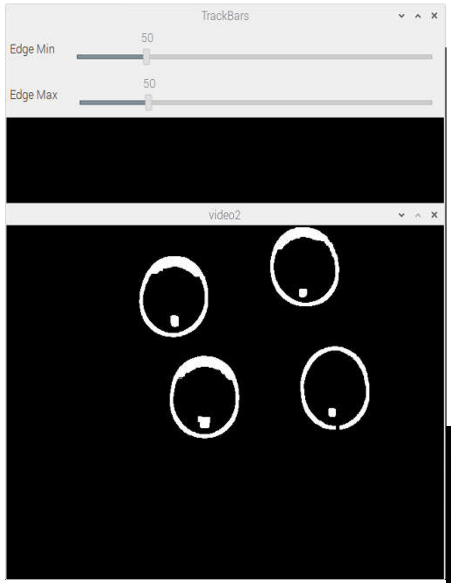
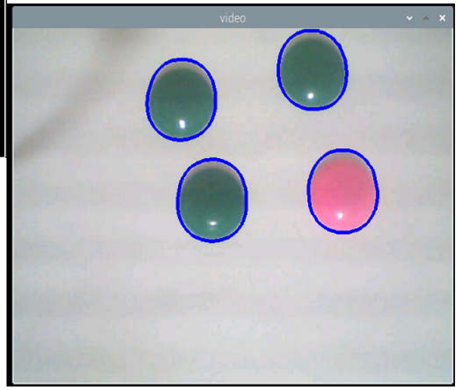
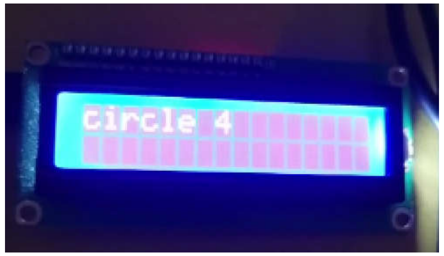


智能系统与控制

树莓派：OpenCV 形状检测 +GPIO（LCD显示）



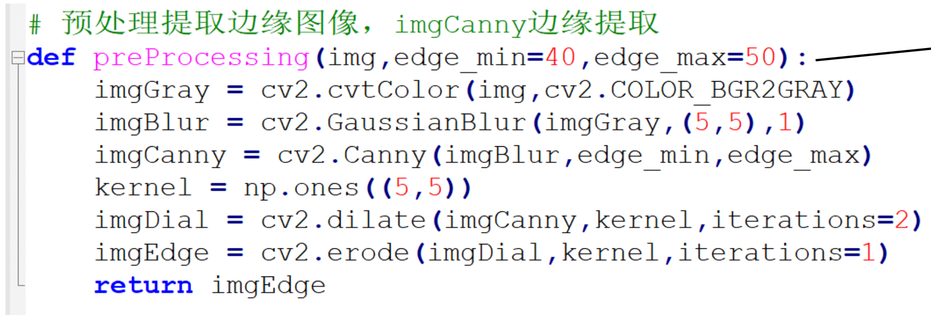


于泓
鲁东大学
信息与电气工程学院
2022.1.9

Opencv形状检测+GPIO硬件结合

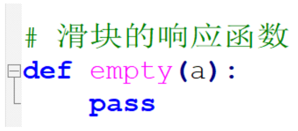
- 检测摄像头内有几个圆形
- 将结果在LCD上显示
- 圆形少于4个报警，并驱动蜂鸣器

视觉部分调试：决定需要的各个参数

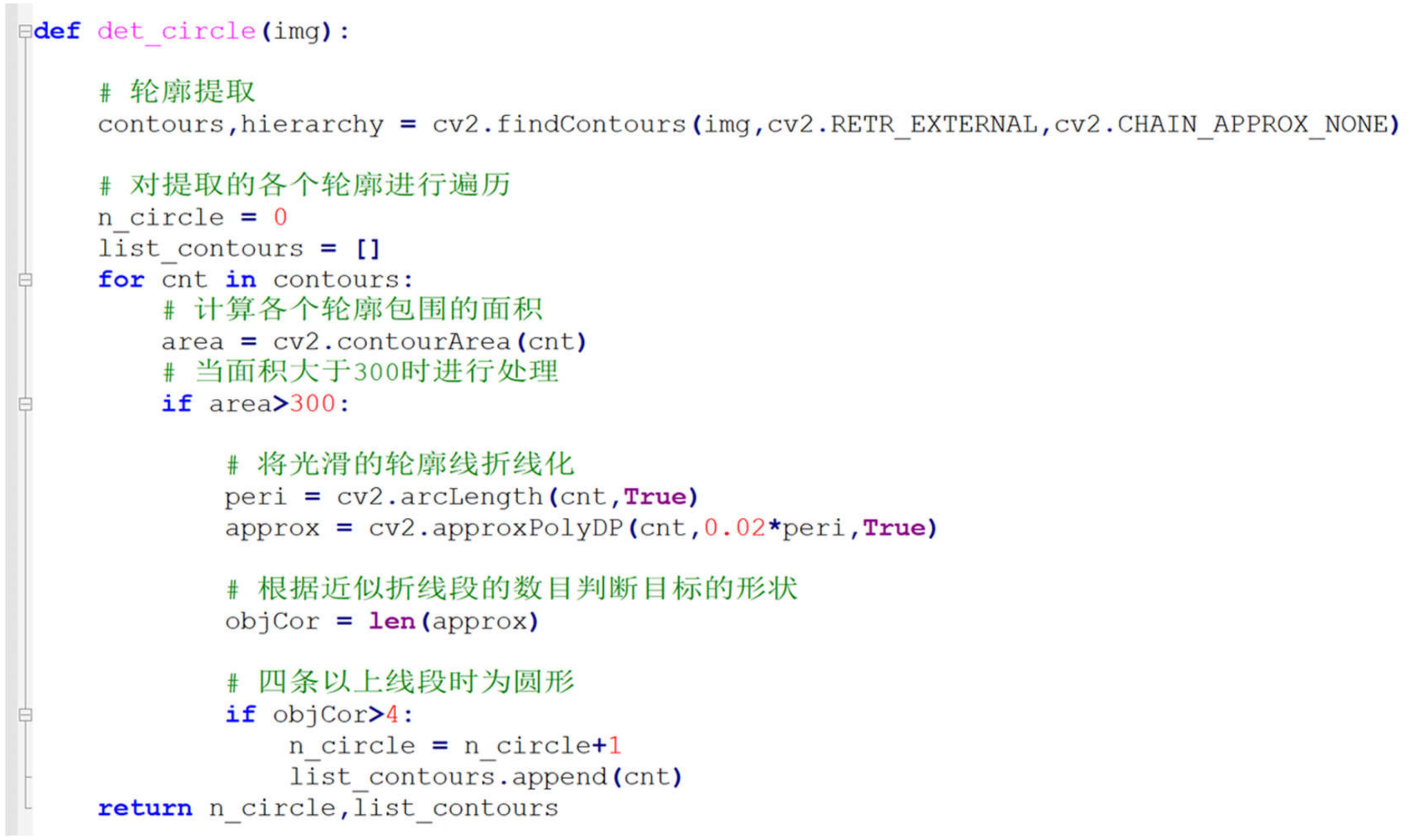


```python
# 预处理提取边缘图像， imgCanny边缘提取
def preProcessing(img,edge_min=40,edge_max=50):
    imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray,(5,5),1)
    imgCanny = cv2.Canny(imgBlur,edge_min,edge_max)
    kernel = np.ones((5,5))
    imgDial = cv2.dilate(imgCanny,kernel,iterations=2)
    imgEdge = cv2.erode(imgDial,kernel,iterations=1)
    return imgEdge
```

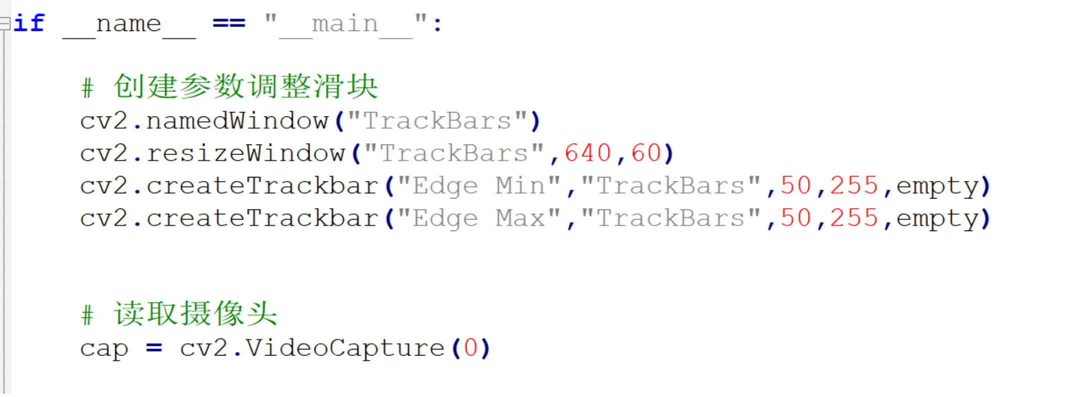
利用滑块来实现调节



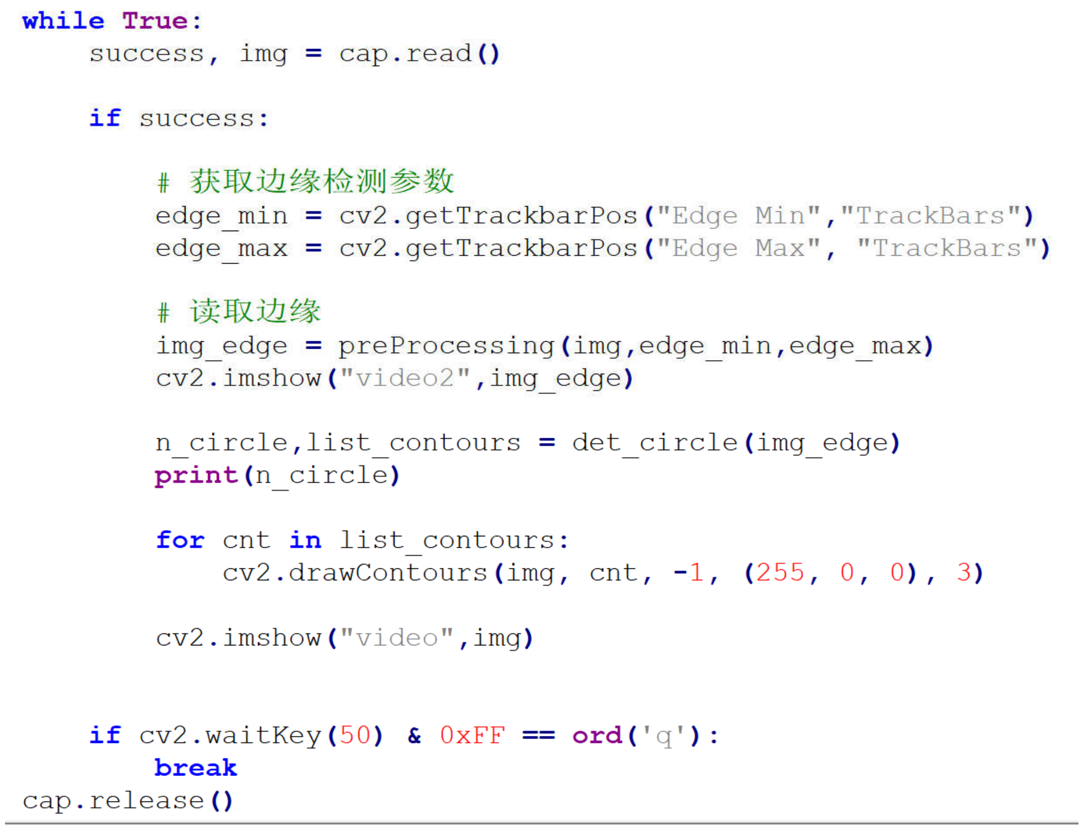
```python
# 滑块的响应函数
def empty(a):
    pass
```



```
def det_circle(img):

    # 轮廓提取
    contours,hierarchy = cv2.findContours(img,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)

    # 对提取的各个轮廓进行遍历
    n_circle = 0
    list_contours = []
    for cnt in contours:
        # 计算各个轮廓包围的面积
        area = cv2.contourArea(cnt)
        # 当面积大于300时进行处理
        if area>300:

            # 将光滑的轮廓线折线化
            peri = cv2.arcLength(cnt,True)
            approx = cv2.approxPolyDP(cnt,0.02*peri,True)

            # 根据近似折线段的数目判断目标的形状
            objCor = len(approx)

            # 四条以上线段时为圆形
            if objCor>4:
                n_circle = n_circle+1
                list_contours.append(cnt)
    return n_circle,list_contours
```

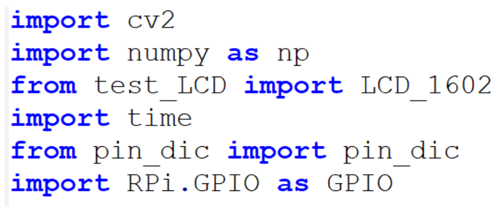


```python
if __name__ == "__main__":

    # 创建参数调整滑块
    cv2.namedWindow("TrackBars")
    cv2.resizeWindow("TrackBars",640,60)
    cv2.createTrackbar("Edge Min","TrackBars",50,255,empty)
    cv2.createTrackbar("Edge Max","TrackBars",50,255,empty)


    # 读取摄像头
    cap = cv2.VideoCapture(0)
```

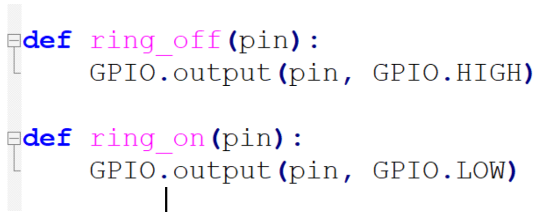


```python
while True:
    success, img = cap.read()

    if success:

        # 获取边缘检测参数
        edge_min = cv2.getTrackbarPos("Edge Min","TrackBars")
        edge_max = cv2.getTrackbarPos("Edge Max", "TrackBars")

        # 读取边缘
        img_edge = preProcessing(img,edge_min,edge_max)
        cv2.imshow("video2",img_edge)

        n_circle,list_contours = det_circle(img_edge)
        print(n_circle)

        for cnt in list_contours:
            cv2.drawContours(img, cnt, -1, (255, 0, 0), 3)

        cv2.imshow("video",img)


    if cv2.waitKey(50) & 0xFF == ord('q'):
        break
cap.release()
```

硬件结合，把代码中显示有关的部分去除用LCD显示来替代

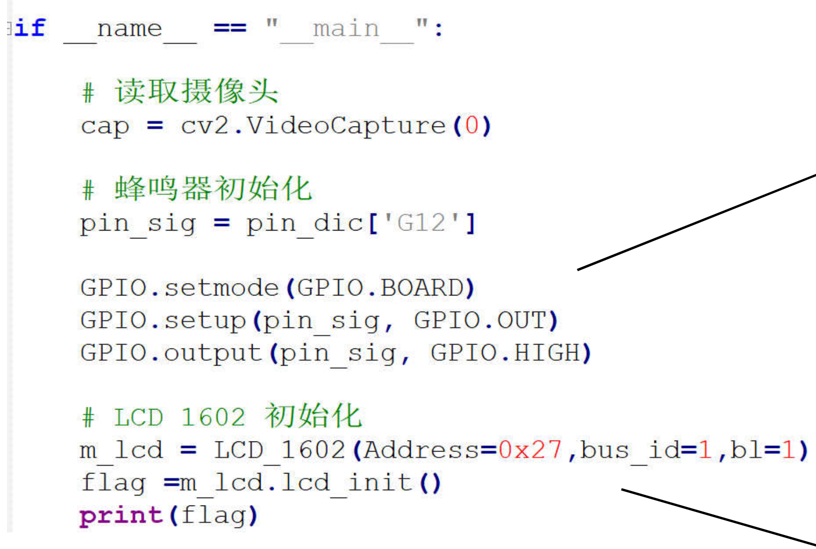


```python
import cv2
import numpy as np
from test_LCD import LCD_1602
import time
from pin_dic import pin_dic
import RPi.GPIO as GPIO
```



```python
def ring_off(pin):
    GPIO.output(pin, GPIO.HIGH)

def ring_on(pin):
    GPIO.output(pin, GPIO.LOW)
```

有源蜂鸣器、低电平启动



```python
if __name__ == "__main__":

    # 读取摄像头
    cap = cv2.VideoCapture(0)

    # 蜂鸣器初始化
    pin_sig = pin_dic['G12']

    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(pin_sig, GPIO.OUT)
    GPIO.output(pin_sig, GPIO.HIGH)

    # LCD 1602 初始化
    m_lcd = LCD_1602(Address=0x27,bus_id=1,bl=1)
    flag =m_lcd.lcd_init()
    print(flag)
```

蜂鸣器初始化

液晶初始化

```
try
    while True:
        success, img = cap.read()

        if success:
            # 读取边缘
            img_edge = preProcessing(img)

            # 圆形检测
            n_circle,list_contours = det_circle(img_edge)

            # 显示字符串
            str_led = 'circle %d     '%(n_circle)
            m_lcd.lcd_display_string(0,0,str_led)

            # 小于4个报警
            if n_circle<4:
                m_lcd.lcd_display_string(0,1,'Alarm')
                ring_on(pin_sig)
            else:
                m_lcd.lcd_display_string(0,1,'       ')
                ring_off(pin_sig)

            time.sleep(0.1)


        if cv2.waitKey(50) & 0xFF == ord('q'):
            break
    cap.release()
except KeyboardInterrupt:
    print('\n Ctrl + C QUIT')

finally:
    GPIO.cleanup()
```

在LCD上显示检测数目

器件小于4个则报警