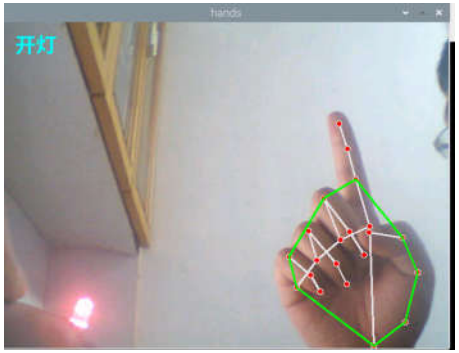
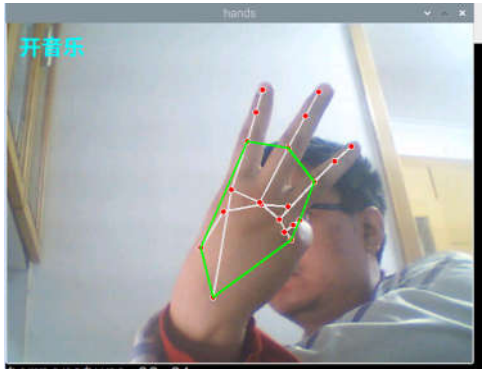
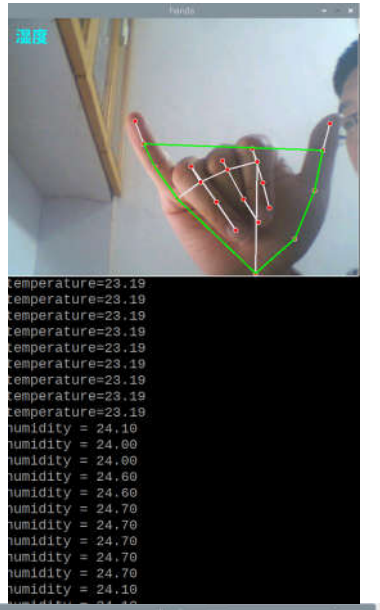
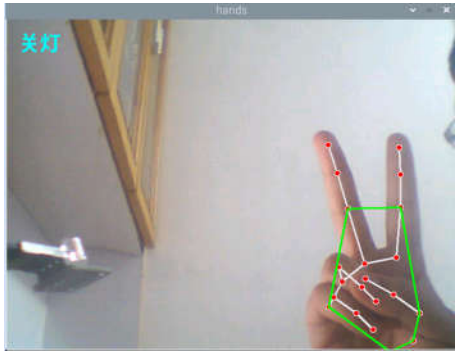
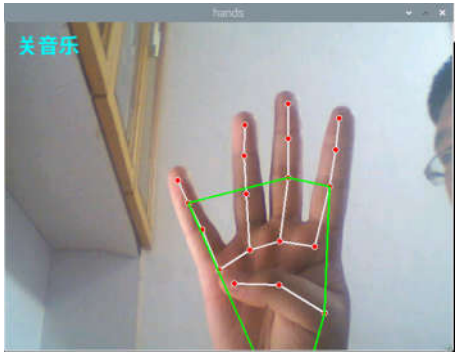
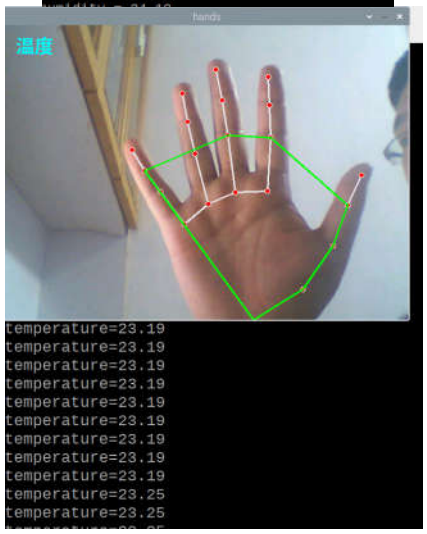










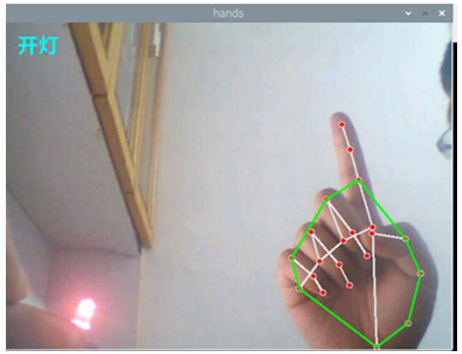


智能系统与控制

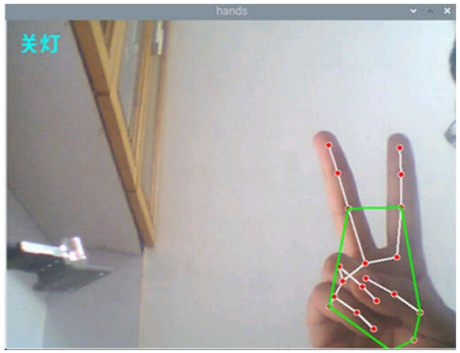
树莓派
Mediapipe-手势+硬件控制

于泓
鲁东大学
信息与电气工程学院
2022.3.20

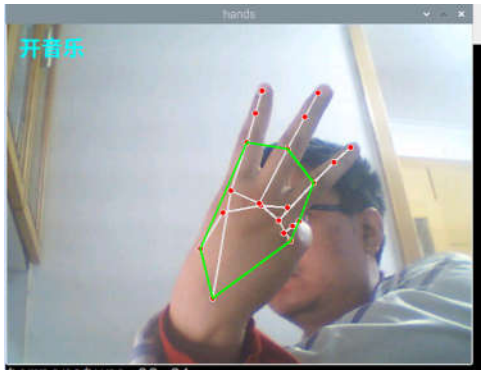
利用mediapipe的手势识别实现硬件控制



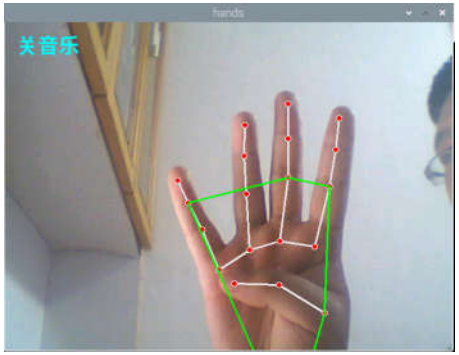
开灯



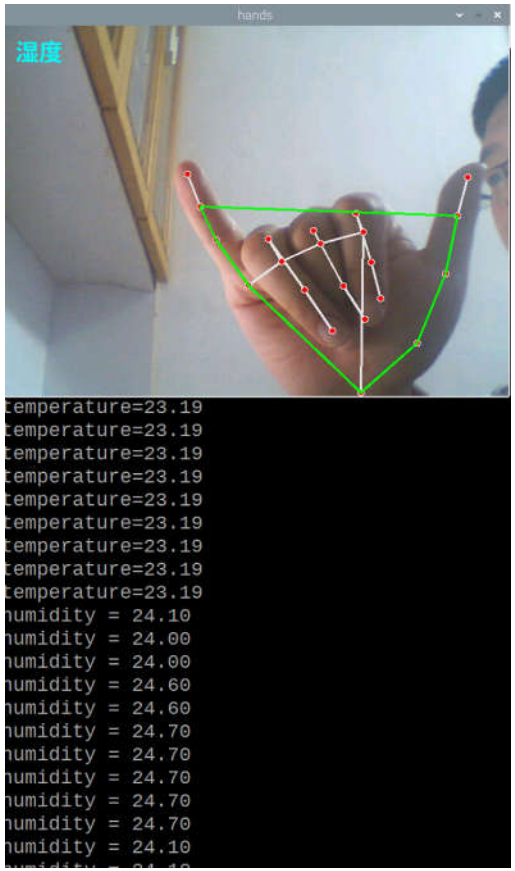
关灯



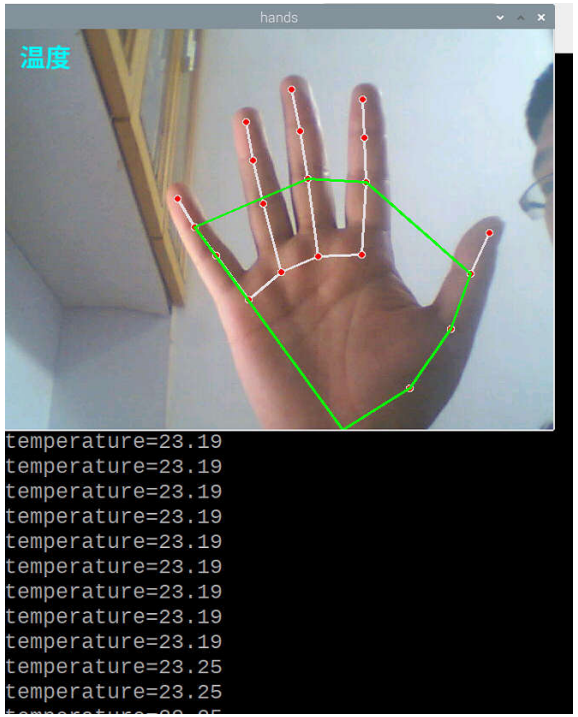
开音乐



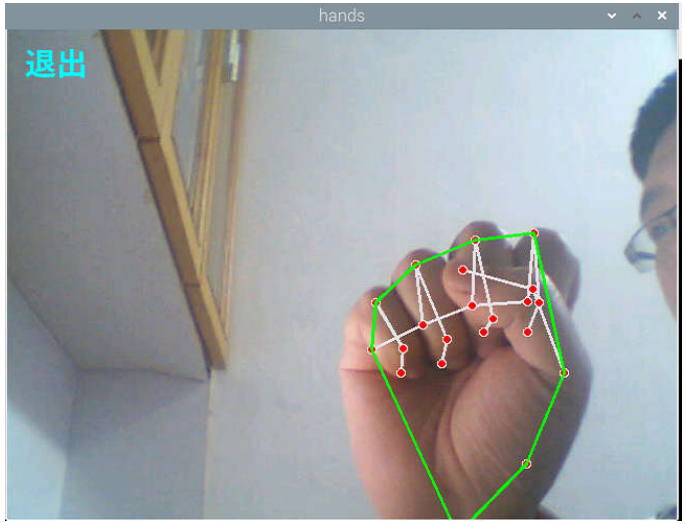
关音乐



DHT11 湿度测量

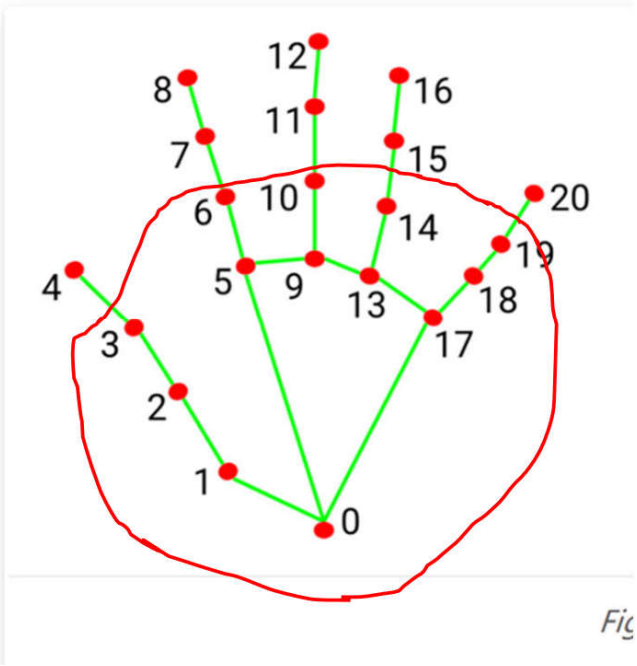


Ds18b20 温度



退出

基于规则的手势识别方法

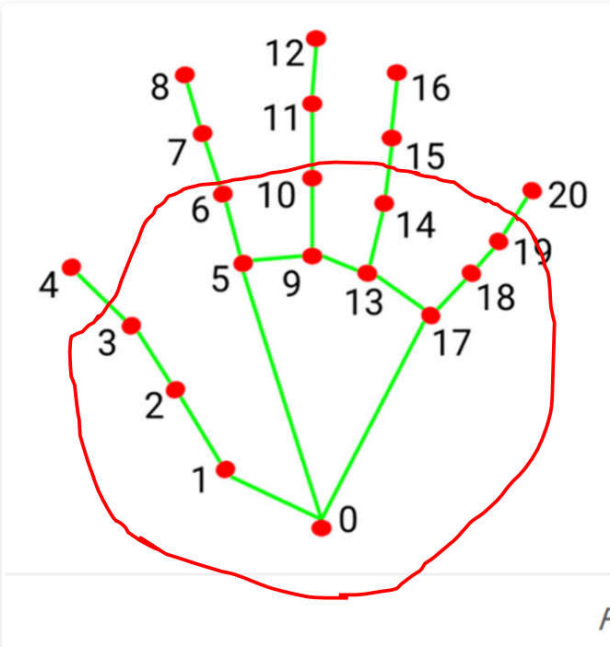


关键点 **[0,1,2,3,6,10,14,19,18,17,10]**

构造一个**凸包**，通过计算

[4,8,12,16,20] 谁在凸包外

来进行简单的手势识别



“1” 8 在外
“2” 8,12 在外
“3” 8,12,16 在外
“4” 8,12,16,20 在外
“5” 4,8,12,16,20在外
“6” 4,20 在外
“10” 都在内

硬件部分： LED 小灯 循环显示

基础模块

```
class Color_LED(object):
    def __init__(self,pins):

        self.pins = pins

        for pin in self.pins:
            GPIO.setup(pin, GPIO.OUT)

        self.pwm_R = GPIO.PWM(self.pins[0], 2000)
        self.pwm_G = GPIO.PWM(self.pins[1], 2000)
        self.pwm_B = GPIO.PWM(self.pins[2], 2000)

        # 初始占空比为0
        self.pwm_R.start(0)
        self.pwm_G.start(0)
        self.pwm_B.start(0)

    def color2ratio(self,x,min_color,max_color,min_ratio,max_ratio):
        return (x - min_color) * (max_ratio - min_ratio) / (max_color - min_color) + min_ratio

    def setColor(self,col):
        R_val,G_val,B_val = col

        R =self.color2ratio(R_val, 0, 255, 0, 100)
        G =self.color2ratio(G_val, 0, 255, 0, 100)
        B =self.color2ratio(B_val, 0, 255, 0, 100)

        # 改变占空比
        self.pwm_R.ChangeDutyCycle(R)
        self.pwm_G.ChangeDutyCycle(G)
        self.pwm_B.ChangeDutyCycle(B)
```

线程控制部分：

引脚 G12 G6 G5

```
class Run_LED(threading.Thread):

    def __init__(self,pins,colors):
        super(Run_LED, self).__init__()

        self.m_led = Color_LED(pins)
        self.flag_run = 0
        self.colors = colors
        self.flag_beak = 0

    def dobreak(self):
        self.flag_beak = 1

    def dostart(self):
        self.flag_run = 1

    def dostop(self):
        self.flag_run = 0

    def getState(self):
        return self.flag_run
```

```
def run(self):

    while True:
        for col in self.colors:

            if self.flag_beak:
                break

            if not self.flag_run:
                col_show=(0,0,0)
            else:
                col_show= col

            # 设置颜色
            self.m_led.setColor(col_show)
            # 延时
            time.sleep(1)
```

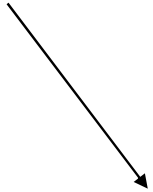
蜂鸣器 基础部分

引脚 G27

```
class Buzzer(object):

    def __init__(self,pin):

        self.pin_buzzer = pin
        GPIO.setup(self.pin_buzzer,GPIO.OUT)

        self.Buzzer = GPIO.PWM(self.pin_buzzer , 440)
        self.Buzzer.start(0)

        self.note2freq = {"cl1":131,"cl2":147 ,'cl3':165 ,"cl4":175 ,"cl5":196 ,"cl6":211 ,"cl7":248,
                          "cm1":262,"cm2":294 ,'cm3':330 ,"cm4":350 ,"cm5":393 ,"cm6":441 ,"cm7":495,
                          "ch1":525,"ch2":589 ,'ch3':661 ,"ch4":700 ,"ch5":786 ,"ch6":882 ,"ch7":990
                          }
        self.delay_beat = 0.3


    def play_one_note(self,note,beat):
        self.Buzzer.ChangeDutyCycle(50)
        self.Buzzer.ChangeFrequency(self.note2freq[note])
        time.sleep(self.delay_beat*beat)
    # def delay_one_beat(self,beat)
        # time.sleep(self.delay_beat*beat)

    def play_silce(self):
        self.Buzzer.ChangeDutyCycle(0)
```

```
class Play_Buzzer(threading.Thread):
    def __init__(self,pin,notes,beats):
        super(Play_Buzzer, self).__init__()


        self.m_buzzer =  Buzzer(pin)
        self.notes = notes
        self.beats = beats

        self.flag_run = 0
        self.flag_beak = 0

    def dobreak(self):
        self.flag_beak = 1


    def dostart(self):
        self.flag_run = 1

    def dostop(self):
        self.flag_run = 0


    def getState(self):
        return self.flag_run
```

```
def run(self):

    while True:
        for note,beat in zip(self.notes,self.beats):

            if self.flag_beak:
                break

            if not self.flag_run:
                self.m_buzzer.play_silce()
            else:
                self.m_buzzer.play_one_note(note,beat)
```



蜂鸣器线程部分

```python
class Ds18b20(object):

    def __init__(self,str_id):
        self.str_id = str_id

    def read(self):
        # 读取温度传感器的数值
        location = os.path.join( "/sys/bus/w1/devices",self.str_id,"w1_slave")

        if os.path.exists(location):
            with open(location) as tf:
                lines = tf.read().splitlines()

            text = lines[-1]
            temperaturedata = text.split(" ")[-1]

            temperature = float(temperaturedata[2:])

            return temperature/1000.0
        else:
            return False
```

温度读取部分

引脚 G4 固定

```
class Read_Ds18b20(threading.Thread):

    def __init__(self,str_id):
        super(Read_Ds18b20, self).__init__()
        self.str_id = str_id

        self.m_ds18b20 =  Ds18b20(self.str_id)
        self.temperature = None
        self.flag_run = 0
        self.flag_break =0

    def dostart(self):
        self.flag_run = 1

    def dostop(self):
        self.flag_run = 0

    def dobreak(self):
        self.flag_break = 1

    def getState(self):
        return self.flag_run
```

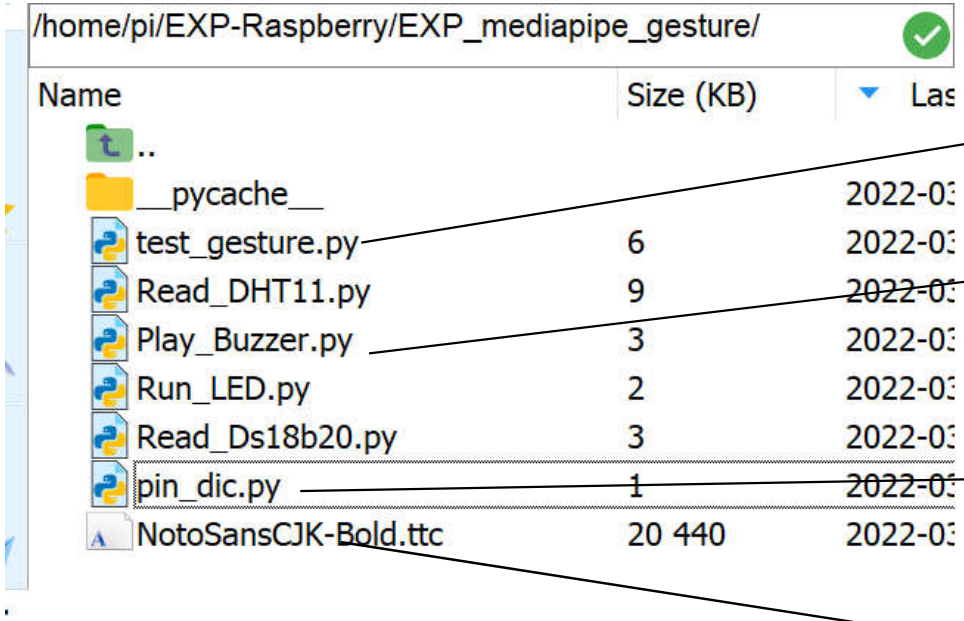
```
def run(self):

    while True:

        if  self.flag_break:
            break

        if self.flag_run:

            t = self.m_ds18b20.read()

            if t:
                self.temperature = t
            else:
                self.temperature = None
        else:
            self.temperature = None

        time.sleep(0.5)

def get_temperature(self):
    return self.temperature
```

温度读取线程部分

```
class Read_DHT11(threading.Thread):

    def __init__(self,pin_D):
        super(Read_DHT11, self).__init__()
        self._pin = pin_D

        self.m_DHT11 =  DHT11(self._pin)
        self.humidity = None
        self.flag_run = 0
        self.flag_break =0

    def dostart(self):
        self.flag_run = 1

    def dostop(self):
        self.flag_run = 0

    def dobreak(self):
        self.flag_break = 1

    def getState(self):
        return self.flag_run
```

```
def run(self):

    while True:

        if self.flag_break:
            break

        if self.flag_run:
            flag, result = self.m_DHT11.read_DHT()

            if flag:
                self.humidity = result[0]
        else:

            self.humidity = None

        time.sleep(0.5)


def get_humidity(self):

    return self.humidity
```

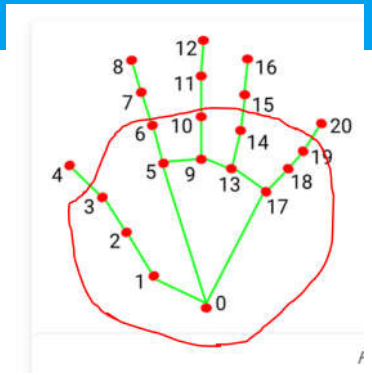
湿度读取线程部分　　引脚G18


/home/pi/EXP-Raspberry/EXP_mediapipe_gesture/
Name
Size (KB)
Las
..
__pycache__
2022-0
test_gesture.py
6
2022-0
Read_DHT11.py
9
2022-0
Play_Buzzer.py
3
2022-0
Run_LED.py
2
2022-0
Read_Ds18b20.py
3
2022-0
pin_dic.py
1
2022-0
NotoSansCJK-Bold.ttc
20 440
2022-0
主程序
四个硬件线程控制文件
引脚说明
中文字库

```
import mediapipe as mp
import cv2
import numpy as np
from PIL import Image,ImageFont,ImageDraw

import threading
import RPi.GPIO as GPIO
from pin_dic import pin_dic
import time

from Run_LED import Run_LED
from Read_Ds18b20 import Read_Ds18b20
from Read_DHT11 import Read_DHT11
from Play_Buzzer import Play_Buzzer
```

绘制中文

```
def paint_chinese_opencv(im,chinese,pos,color,font_size = 20):
    img_PIL = Image.fromarray(cv2.cvtColor(im,cv2.COLOR_BGR2RGB))
    font = ImageFont.truetype('NotoSansCJK-Bold.ttc',font_size,encoding="utf-8")
    fillColor = color
    position = pos

    draw = ImageDraw.Draw(img_PIL)
    draw.text(position,chinese,fillColor,font)
    img = cv2.cvtColor(np.asarray(img_PIL),cv2.COLOR_RGB2BGR)
    return img
```




```
def get_guester(img,list_lms):

    hull_index = [0,1,2,3,6,10,14,19,18,17,10]
    hull = cv2.convexHull(list_lms[hull_index,:])
    # 绘制凸包
    cv2.polylines(img,[hull], True, (0, 255, 0), 2)

    # 查找外部的点数
    n_fig = -1
    ll = [4,8,12,16,20]
    up_fingers = []

    for i in ll:
        pt = (int(list_lms[i][0]),int(list_lms[i][1]))
        dist= cv2.pointPolygonTest(hull,pt,True)
        if dist <0:
            up_fingers.append(i)

guester = None

if len(up_fingers)==1 and up_fingers[0]==8:
    guester = 1
elif len(up_fingers)==2 and up_fingers[0]==8 and up_fingers[1]==12:
    guester = 2

elif len(up_fingers)==3 and up_fingers[0]==8 and up_fingers[1]==12 and up_fingers[2]==16:
    guester = 3

elif len(up_fingers)==4 and up_fingers[0]==8 and up_fingers[1]==12 and up_fingers[2]==16 and up_fingers[3]==20:
    guester = 4

elif len(up_fingers)==5:
    guester = 5

elif len(up_fingers)==2 and up_fingers[0]==4 and up_fingers[1]==20:
    guester = 6
elif len(up_fingers)==2 and up_fingers[0]==4 and up_fingers[1]==8:
    guester = 8

elif len(up_fingers)==0:
    guester = 10

return  guester
```

```
if __name__ == "__main__":

    # 硬件部分初始化
    GPIO.setmode(GPIO.BOARD)

    # LED 小灯 初始化
    pin_R = pin_dic['G12']
    pin_G = pin_dic['G6']
    pin_B = pin_dic['G5']
    colors = [(255,0,0),(0,255,0),(0,0,255),(255,255,0),(0,197,204),(192,255,62),(148,0,211),(118,238,200)];
    pins_LED = [pin_R,pin_G,pin_B]

    m_runing_LED = Run_LED(pins_LED,colors)
    m_runing_LED.setDaemon(True)

    # 启动小灯线程
    m_runing_LED.dostop()
    m_runing_LED.start()


    # 湿度传感器初始化
    pin_dht=  pin_dic['G18']
    m_read_DHT11 =  Read_DHT11(pin_dht)
    m_read_DHT11.setDaemon(True)

    # 启动湿度读取线程
    m_read_DHT11.dostart()
    m_read_DHT11.start()
```

```
# 蜂鸣器初始化
pin_buzzer = pin_dic['G27']

notes = ['cm1' ,'cm1' , 'cm1' , 'cl5' , 'cm3' , 'cm3' , 'cm3' , 'cm1' ,
         'cm1' , 'cm3' , 'cm5' , 'cm5' , 'cm4' , 'cm3' , 'cm2' , 'cm2' ,
         'cm3' , 'cm4' , 'cm4' , 'cm3' , 'cm2' , 'cm3' , 'cm1' , 'cm1' ,
         'cm3' , 'cm2' , 'cl5' , 'cl7', 'cm2' , 'cm1']
beats = [1 , 1 , 2 , 2 , 1 , 1 , 2 , 2 ,
        1 , 1 , 2 , 2 , 1 , 1 , 3 , 1 ,
        1 , 2 , 2 , 1 , 1 , 2 , 2 , 1 ,
        1 , 2 , 2 , 1 , 1 , 3]

m_play_buzzer = Play_Buzzer(pin_buzzer,notes,beats)
m_play_buzzer.setDaemon(True)

# 启动蜂鸣器线程
m_play_buzzer.dostop()
m_play_buzzer.start()


# 温度传感器初始化
str_id = "28-0300a2794829"
m_read_Ds18b20 =  Read_Ds18b20(str_id)
m_read_Ds18b20.setDaemon(True)

# 启动温度读取线程
m_read_Ds18b20.dostart()
m_read_Ds18b20.start()
```

```
dic_guester= {1:"开灯",2:"关灯",3:"开音乐",4:"关音乐",5:"温度",6:"湿度",10:"退出",8:" "}

cap = cv2.VideoCapture(0)
# 定义手 检测对象
mpHands = mp.solutions.hands
hands = mpHands.Hands()
mpDraw = mp.solutions.drawing_utils

c_guester = None
p_guester = None
str_show = ""
while True:

    # 读取一帧图像
    success, img = cap.read()
    if not success:
        continue
    image_height, image_width, _ = np.shape(img)

    # 转换为RGB
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # 得到检测结果
    results = hands.process(imgRGB)
```

```
if results.multi_hand_landmarks:
    hand = results.multi_hand_landmarks[0]

    mpDraw.draw_landmarks(img,hand,mpHands.HAND_CONNECTIONS)

    # 采集所有关键点的坐标
    list_lms = []
    for i in range(21):
        pos_x = hand.landmark[i].x*image_width
        pos_y = hand.landmark[i].y*image_height
        list_lms.append([int(pos_x),int(pos_y)])

    # 获取手势
    list_lms = np.array(list_lms,dtype=np.int32)
    c_guester = get_guester(img,list_lms)


    if not c_guester is None and c_guester != p_guester:
        p_guester = c_guester
```

获取当前手势

只有手势变化才更新

```
# 根据动作进行处理
if p_guester == 1:# 开灯
    m_runing_LED.dostart()

elif p_guester == 2:  # 关灯
    m_runing_LED.dostop()

elif p_guester == 3: # 开音乐
    m_play_buzzer.dostart()

elif p_guester == 4: # 关音乐
    m_play_buzzer.dostop()

elif p_guester == 5: # 温度

    temperature = m_read_Ds18b20.get_temperature()
    if not temperature is None:
        print("temperature=%.2f"%(temperature))
elif p_guester == 6: # 湿度
    humidity = m_read_DHT11.get_humidity()
    if not humidity is None:
        print("humidity = %.2f"%(humidity))



if  not p_guester is None:
    str_show = ' %s'%(dic_guester[p_guester])


img = paint_chinese_opencv(img,str_show,(10,10),(0,255,255),font_size=30)

cv2.imshow("hands",img)
```

```
    key =  cv2.waitKey(1) & 0xFF

    # 按键 "q" 退出
    if key ==  ord('q') or  p_guester == 10:
        m_play_buzzer.dobreak()
        m_read_DHT11.dobreak()
        m_read_Ds18b20.dobreak()
        m_runing_LED.dobreak()
        time.sleep(2)
        GPIO.cleanup()

        break
cap.release()
```

手势10 退出