**mycamera.py**

from picamera import PiCamera

from gpiozero import Button

from gpiozero import LED

from datetime import datetime

from signal import pause

from time import sleep

import RPi.GPIO as GPIO

import time

import sys

import math

from PIL import Image

import os

# SPI screen GPIO

cs = 23 # 片选

rs = 17 # 数据 / 命令 切换

sda = 13 # 数据

scl = 19 # 时钟

reset = 27 # 复位

camera = PiCamera()

capture\_button = Button(5)

recording\_button = Button(21)

recordinglight = LED(16)

left\_button = Button(24)

right\_button = Button(25)

index = -1

flag1 = True

flag2 = True

### take photos

def capture():

timestamp = datetime.now().isoformat()

camera.capture('/home/yes/images/%s.jpg' % timestamp)

### take videos

def func1():

timestamp = datetime.now().isoformat()

recording\_button.when\_pressed = camera.start\_recording('/home/yes/Videos/%s.h264' % timestamp)

recordinglight.on()

global flag1, flag2

flag1 = False

flag2 = True

def func2():

recording\_button.when\_pressed = camera.stop\_recording()

recordinglight.off()

global flag2, flag1

flag2 = False

flag1 = True

### display module

# 传输byte

def setByteData(data):

# print ""

# print "S-----------setByte---------------:", hex(data)

for bit in range(0, 8):

# 传入的数字从高位到低位依次判断是否为1，若为1则设置高电平，否则设置低电平

# 判断的方法是先向左移位，把要判断的位移动到最高位然后跟0x80（1000 0000）相与，

# 如果结果仍然是0x80（1000 0000）就表示最高位是1，否则最高位就是0

if ((data << bit) & 0x80 == 0x80):

setBitData(True)

# print "1",

else:

setBitData(False)

# print "0",

# print ""

# print "E-----------setByte---------------"

def setBitData(data):

GPIO.output(scl, False)

GPIO.output(sda, data)

GPIO.output(scl, True)

def write\_command(cmd):

GPIO.output(cs, False)

GPIO.output(rs, False)

setByteData(cmd)

GPIO.output(cs, True)

def write\_data(data):

GPIO.output(cs, False)

GPIO.output(rs, True)

setByteData(data)

GPIO.output(cs, True)

def write\_data\_16bit(dataH, dataL):

write\_data(dataH)

write\_data(dataL)

def lcd\_reset():

GPIO.output(reset, False)

time.sleep(0.1)

GPIO.output(reset, True)

time.sleep(0.1)

def lcd\_init(width, heigh):

lcd\_reset()

write\_command(0x11) # Exit Sleep

time.sleep(0.02)

write\_command(0x26) # Set Default Gamma

write\_data(0x04)

write\_command(0xB1) # Set Frame Rate

write\_data(0x0e)

write\_data(0x10)

write\_command(0xC0) # Set VRH1[4:0] & VC[2:0] for VCI1 & GVDD

write\_data(0x08)

write\_data(0x00)

write\_command(0xC1) # Set BT[2:0] for AVDD & VCL & VGH & VGL

write\_data(0x05)

write\_command(0xC5) # Set VMH[6:0] & VML[6:0] for VOMH & VCOML

write\_data(0x38)

write\_data(0x40)

write\_command(0x3a) # Set Color Format

write\_data(0x05)

write\_command(0x36) # RGB

write\_data(0xc8)

write\_command(0x2A) # Set Column Address

write\_data(0x00)

write\_data(0x00)

write\_data(0x00)

write\_data(width - 1)

write\_command(0x2B) # Set Page Address

write\_data(0x00)

write\_data(0x00)

write\_data(0x00)

write\_data(heigh - 1)

write\_command(0xB4)

write\_data(0x00)

write\_command(0xf2) # Enable Gamma bit

write\_data(0x01)

write\_command(0xE0)

write\_data(0x3f) # p1

write\_data(0x22) # p2

write\_data(0x20) # p3

write\_data(0x30) # p4

write\_data(0x29) # p5

write\_data(0x0c) # p6

write\_data(0x4e) # p7

write\_data(0xb7) # p8

write\_data(0x3c) # p9

write\_data(0x19) # p10

write\_data(0x22) # p11

write\_data(0x1e) # p12

write\_data(0x02) # p13

write\_data(0x01) # p14

write\_data(0x00) # p15

write\_command(0xE1)

write\_data(0x00) # p1

write\_data(0x1b) # p2

write\_data(0x1f) # p3

write\_data(0x0f) # p4

write\_data(0x16) # p5

write\_data(0x13) # p6

write\_data(0x31) # p7

write\_data(0x84) # p8

write\_data(0x43) # p9

write\_data(0x06) # p10

write\_data(0x1d) # p11

write\_data(0x21) # p12

write\_data(0x3d) # p13

write\_data(0x3e) # p14

write\_data(0x3f) # p15

write\_command(0x29) # Display On

write\_command(0x2C)

def show\_single\_color(DH, DL, width, heigh):

for i in range(0, heigh):

for j in range(0, width):

write\_data\_16bit(DH, DL)

def rgb2rgb565(image):

'''

将RGB（255，255，255）转换为（31，63，31）

'''

rgb565 = []

w, h = image.size

for y in range(h):

for x in range(w):

r, g, b = image.getpixel((x, y))[:3]

r = math.ceil(r / 0xFF \* 0x1F)

g = math.ceil(g / 0xFF \* 0x3f)

b = math.ceil(b / 0xFF \* 0x1F)

rgb565.append((r, g, b))

return rgb565

def show4rgb565(rgb565):

'''

将rgb合并后取高低8位用于传输显示

原文中的RGB565 从高位到低位依次是红、绿、蓝

但是我所用的屏幕需要蓝、绿、红才能正常显示

'''

DH = 0

DL = 0

for r, g, b in rgb565:

h16 = ((b << 6) + g << 5) + r # 蓝、绿、红

# h16 = ((r << 6) + g << 5) + b # 红、绿、蓝

DH = (h16 & 0xFF00) >> 8

DL = h16 & 0x00FF

write\_data\_16bit(DH, DL)

# print(hex(h16),hex(DH), hex(DL))

def find\_new\_file(dir, idx):

print(idx)

'''查找目录下最新的文件'''

file\_lists = os.listdir(dir)

file\_lists.sort(key=lambda fn: os.path.getmtime(dir + "/" + fn)

if not os.path.isdir(dir + "/" + fn) else 0)

idx = idx % len(file\_lists)

#print('最新的文件为： ' + file\_lists[index])

file = os.path.join(dir, file\_lists[idx])

#print('完整路径：', file)

return file

#if \_\_name\_\_ == '\_\_main\_\_':

def display(index):

#try:

GPIO.setmode(GPIO.BCM)

GPIO.setup(cs, GPIO.OUT)

GPIO.setup(rs, GPIO.OUT)

GPIO.setup(sda, GPIO.OUT)

GPIO.setup(scl, GPIO.OUT)

GPIO.setup(reset, GPIO.OUT)

lcd\_init(128, 160) # 屏幕宽、高

write\_command(0x2C)

filename = find\_new\_file("/home/yes/images", index)

#print(index)

image = Image.open(filename)

image = image.resize((128, 160), Image.ANTIALIAS) #修改图片大小 屏幕宽、高

rgb565 = rgb2rgb565(image)

show4rgb565(rgb565)

#while True:

# pass

#except KeyboardInterrupt:

# pass

# 清理GPIO口

#GPIO.cleanup()

def func\_left():

global index

#print(index)

display(index)

index = index - 1

def func\_right():

global index

display(index)

#print(index)

index = index + 1

left\_button.when\_pressed = func\_left

right\_button.when\_pressed = func\_right

capture\_button.when\_pressed = capture

while True:

while flag1:

recording\_button.when\_pressed = func1

while flag2:

recording\_button.when\_pressed = func2

pause()

**camera\_switch.py**

import subprocess

from gpiozero import Button

from gpiozero import LED

import signal

import os

control\_button = Button(12)

led = LED(4)

spid = 0

def startfunc():

global spid

proc = subprocess.Popen(["python3", "/home/yes/embeded\_proj/mycamera.py"])

spid = proc.pid

def killfunc():

global spid

os.kill(spid, signal.SIGKILL)

flag1 = True

flag2 = True

def func1():

startfunc()

led.on()

global flag1, flag2

flag1 = False

flag2 = True

def func2():

killfunc()

led.off()

global flag2, flag1

flag2 = False

flag1 = True

while True:

while flag1:

control\_button.when\_pressed = func1

while flag2:

control\_button.when\_pressed = func2

#subprocess.Popen(["python3", "mycamera.py"])

pause()