#!/usr/bin/env python

# encoding: utf-8

#树莓派GPIO控制灯泡

import RPi.GPIO

import time

from threading import Thread,Event

class GPIO(Thread):

def \_\_init\_\_(self):

Thread.\_\_init\_\_(self)

self.LED = 15

#设置GPIO的串口

self.SLEDR,self.SLEDG,self.SLEDB =13,19,26

#灯泡启动

def jox\_start(self,jox\_type):

print(jox\_type+":start")

#判断启动指令

RPi.GPIO.setmode(RPi.GPIO.BCM)

if jox\_type == "LED":

#启动灯泡

self.led\_exit = Event()

RPi.GPIO.setup(self.LED, RPi.GPIO.OUT)

self.pwm = RPi.GPIO.PWM(self.LED, 70)

self.pwm.start(0)

self.pwm.ChangeDutyCycle(100)

else:

#启动指示灯

RPi.GPIO.setup(self.SLEDR, RPi.GPIO.OUT)

RPi.GPIO.setup(self.SLEDG, RPi.GPIO.OUT)

RPi.GPIO.setup(self.SLEDB, RPi.GPIO.OUT)

self.pwmR = RPi.GPIO.PWM(self.SLEDR, 70)

self.pwmG = RPi.GPIO.PWM(self.SLEDG, 70)

self.pwmB = RPi.GPIO.PWM(self.SLEDB, 70)

self.pwmR.start(0)

self.pwmG.start(0)

self.pwmB.start(0)

#指示灯红色

if jox\_type == "RED":

self.pwmR.ChangeDutyCycle(100)

self.pwmG.ChangeDutyCycle(0)

self.pwmB.ChangeDutyCycle(0)

#指示灯绿色

elif jox\_type == "GREEN":

self.pwmR.ChangeDutyCycle(0)

self.pwmG.ChangeDutyCycle(100)

self.pwmB.ChangeDutyCycle(0)

#指示灯蓝色

elif jox\_type == "BULE":

self.pwmR.ChangeDutyCycle(0)

self.pwmG.ChangeDutyCycle(0)

self.pwmB.ChangeDutyCycle(100)

#关闭灯

def jox\_stop(self,jox\_type):

print(jox\_type+":stop")

if jox\_type == "LED":

self.pwm.ChangeDutyCycle(0)

elif jox\_type =="RGB":

self.pwmR.ChangeDutyCycle(0)

self.pwmG.ChangeDutyCycle(0)

self.pwmB.ChangeDutyCycle(0)

#退出且关闭串口

def jox\_exit(self):

RPi.GPIO.cleanup()

#!/usr/bin/env python

# encoding: utf-8

import jox\_GPIO

#灯泡操作api

gpio =jox\_GPIO.GPIO()

gpio.start()

oper = ""

while oper!="exit":

oper = input("qing shu ru caozuo:")

if oper == "start":

oper=""

gpio.jox\_start("LED")

elif oper == "stop":

oper=""

gpio.jox\_stop("LED")

elif oper == "startr":

oper=""

gpio.jox\_start("RED")

elif oper == "startg":

oper=""

gpio.jox\_start("GREEN")

elif oper == "startb":

oper=""

gpio.jox\_start("BULE")

elif oper == "stoprgb":

oper=""

gpio.jox\_stop("RGB")

elif oper == "exit":

gpio.jox\_exit()

break

else:

print("wu xiao zhi ling")

#账号权限token http接口

from flask\_restful import reqparse, Resource

from jox\_api import Token,Time

from jox\_config import httpState

class TokenRoute(Resource):

def post(self):

try:

self.tokenClass = Token.Token()

self.parser = reqparse.RequestParser()

self.parser.add\_argument('user', type=str, help='user: type is str')

self.parser.add\_argument('passwd', type=str, help='passwd: type is str')

self.parser.add\_argument('token', type=str, help='token: type is str')

self.args = self.parser.parse\_args()

self.user = self.args['user']

self.passwd = self.args['passwd']

self.datetime = Time.Time().get\_time()

if self.user==None or self.passwd == None:

return {"code":1002,"data":{"msg":httpState[1002]}},200

else:

self.state = self.tokenClass.obtain(self.user,self.passwd,self.datetime)

return self.state,200

except Exception as e:

return {"code": -1, "data": {"msg": httpState[-1]} + "=>" + str(e)}, 500

from jox\_api.Mysql import SelectMySQL

from jox\_config import httpState

import hashlib

from jox\_api import Time

import datetime

class Token():

def \_\_init\_\_(self):

self.Mysql = SelectMySQL()

self.time = Time.Time()

def permission(self,role\_id,identifier):

self.sql = "select \* from role\_perms where role\_id=%d" %int(role\_id)

self.resql = self.Mysql.select\_data(self.sql)

if self.resql['state'] == "T":

self.perms\_list = []

for data in self.resql['alldata']:

self.perms\_list.append(data['perms'])

if identifier in self.perms\_list:

return {"code": 4000, "msg": "账号权限验证通过"}

else:

return {"code": 4001, "msg": "账号权限不足"}

else:

return {"code": 4001, "msg": "账号权限不足"}

def validate(self,token,role\_id):

self.sql = "select \* from sys\_user\_token where token=\'%s\'" %token

self.resql = self.Mysql.select\_data(self.sql)

self.state = self.resql['state']

self.datetimes = self.time.get\_time()

if self.state == "T":

self.alldata = self.resql['alldata']

if self.datetimes >= datetime.datetime.strptime(self.alldata[0]['update\_time'], "%Y-%m-%d %H:%M:%S"):

return {"code":1003,"msg":"token已失效"}

else:

self.sql1 = "select \* from sys\_user\_role where user\_id=%d" % self.alldata[0]['user\_id']

self.resql1 = self.Mysql.select\_data(self.sql1)

if self.resql1['state'] == "T":

self.role\_list = []

for data in self.resql1['alldata']:

self.role\_list.append(data['role\_id'])

if int(role\_id) in self.role\_list:

return {"code": 1005, "msg": "token验证通过", "user\_id": self.alldata[0]['user\_id']}

else:

return {"code": 1004, "msg": "token验证不通过"}

else:

return {"code": 1004, "msg": "token验证不通过"}

elif self.state == "F":

return {"code": 1004, "msg": "token验证不通过"}

elif self.state == "E":

return {"code": 1004, "msg": "token验证不通过"}

def sele\_token(self,user\_id):

self.select\_token = "select \* from sys\_user\_token where user\_id=\'%s\'" %user\_id

self.resql\_token = self.Mysql.select\_data(self.select\_token)

self.state\_token = self.resql\_token['state']

if self.state\_token == 'T':

return 1006

elif self.state\_token == "F":

return 1007

elif self.state\_token == "E":

return 1008

def obtain(self,user,passwd,datetimes):

self.select\_user = "select \* from sys\_user where username=\'%s\' and password=\'%s\'" %(user,passwd)

self.resql\_user = self.Mysql.select\_data(self.select\_user)

self.state\_user = self.resql\_user['state']

if self.state\_user == "T":

self.alldata = self.resql\_user['alldata']

if self.alldata[0]['status'] != 1:

return {"code": 1002, "data": {"msg": httpState[1002]}}

else:

self.user\_id = self.alldata[0]['user\_id']

m2 = hashlib.md5()

m2.update((str(datetimes)+str(self.user\_id)).encode("utf8"))

self.token =m2.hexdigest()

self.user\_status = self.sele\_token(self.user\_id)

if self.user\_status == 1006:

self.update\_token = "UPDATE sys\_user\_token SET token=\'%s\',expire\_time=\'%s\',update\_time=\'%s\' WHERE user\_id=%d" % ( self.token, datetimes, self.time.more\_day(datetimes),self.user\_id)

self.resql\_token = self.Mysql.insert\_data(self.update\_token)

elif self.user\_status == 1007:

self.insert\_token = "INSERT INTO sys\_user\_token (user\_id,token,expire\_time,update\_time) VALUES(%d,\'%s\',\'%s\',\'%s\')" %(self.user\_id,self.token,datetimes, self.time.more\_day(datetimes))

self.resql\_token = self.Mysql.insert\_data(self.insert\_token)

elif self.user\_status == 1008:

return {"code":1008,"data":{"msg":httpState[1008]}}

self.state\_token = self.resql\_token['state']

if self.state\_token == "T":

self.sql1 = "select \* from get\_role where user\_id=%d" % self.alldata[0]['user\_id']

self.resql1 = self.Mysql.select\_data(self.sql1)

self.role\_list = []

if self.resql1['state'] == "T":

for data in self.resql1['alldata']:

self.role\_list.append({'role\_id':data['role\_id'],"role\_name":data['role\_name']})

return {"code":1000,"data":{"msg":httpState[1000],"token":self.token,"role\_list":self.role\_list}}

elif self.state\_token == "E":

return {"code":1001,"data":{"msg":httpState[1001]}}

elif self.state\_user == "F":

return {"code": 1002, "data": {"msg": httpState[1002]}}

elif self.state\_user == "E":

return {"code": 1001, "data": {"msg": httpState[1001]}}

#连接mysql数据库api

from jox\_api import Mysql,Token

from flask\_restful import request

mysqlClass = Mysql

def permission(identifier="sys:any:any"):

def wrapper(func):

def decorated\_view(\*args, \*\*kwargs):

if identifier == "sys:any:any":

return func(\*args, \*\*kwargs)

else:

tokenClass = Token.Token()

token = request.headers['token']

role\_id = request.headers['role-id']

validate\_token = tokenClass.validate(token,int(role\_id))

validate\_code = validate\_token["code"]

if validate\_code != 1005:

return validate\_token

else:

perms = tokenClass.permission(int(role\_id),identifier)

if perms['code'] == 4000:

return func(\*args, \*\*kwargs)

else:

return perms

return decorated\_view

return wrapper

import datetime

import time

import decimal

#utils类

#处理时间

class Time():

def get\_time(self,):

st = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())

sttotime = datetime.datetime.strptime(st, "%Y-%m-%d %H:%M:%S")

return sttotime

def more\_day(self,datetimes):

return datetimes + datetime.timedelta(days=1)

#处理元组

class Dict():

def data\_format(self,list):

for data in list:

for key, value in data.items():

if isinstance(value,datetime.datetime):

data[key] = value.strftime("%Y-%m-%d %H:%M:%S")

if isinstance(value,decimal.Decimal):

data[key] = float(value)

return list

#账号根据权限返回后台菜单menu接口

from jox\_api.Mysql import SelectMySQL

from jox\_api import Role

from jox\_config import httpState

from jox\_api import User

class Menu():

def \_\_init\_\_(self):

self.roleClass = Role.Role()

self.userClass = User.User()

def get\_menu(self,role\_id):

self.menu\_list = []

self.temp\_role\_menu = self.roleClass.role\_menu(role\_id)

if self.temp\_role\_menu['code'] == 1013:

self.temp = self.temp\_role\_menu["menu\_list"]

self.menu\_list.extend(self.temp)

# menu\_list去重

self.menu\_list = list(set(self.menu\_list))

self.menu\_sql = "("

for menu\_id in self.menu\_list:

self.menu\_sql += str(menu\_id)+","

self.menu\_sql = self.menu\_sql[:-1]

self.menu\_sql+=")"

self.Mysql = SelectMySQL()

self.select\_menu = "select \* from sys\_menu where menu\_id in %s" % self.menu\_sql

self.resql\_menu = self.Mysql.select\_data(self.select\_menu)

self.state\_menu = self.resql\_menu['state']

if self.state\_menu == "T":

self.format\_data,self.jur\_data,self.api\_data = self.format\_menu\_info(list(self.resql\_menu['alldata']))

return {"code":1014,"data":{"msg":"获取menu成功","menu\_list":self.format\_data,"jur\_list":self.jur\_data,"api\_list":self.api\_data}}

elif self.state\_menu == "F":

return {"code": 1009, "data": {"msg": httpState[1009]}}

elif self.state\_menu == "E":

return {"code": 1009, "data": {"msg": httpState[1009]}}

def format\_menu\_info(self,menu\_data):

self.clean\_data = []

self.jur\_data = []

self.data1 = []

self.data2 = []

self.data3 = []

self.api\_data = []

for j in menu\_data:

#url权限

if j['type'] == 1:

self.jur\_data.append(j['url'])

#接口权限

if j['type'] == 2:

self.api\_data.append(j['perms'])

for da in menu\_data:

# 一级菜单

if da['parent\_id']==0 and da['type'] == 1:

self.obj = {

"menu\_id":da['menu\_id'],

"parent\_id":da['parent\_id'],

"order\_num":da['order\_num'],

"icon":da['icon'],

"index":str(da['url'])+str(da['perms']),

"title":da['name']

}

self.clean\_data.append(self.obj)

# 一级菜单目录

elif da['parent\_id']==0 and da['type'] == 0:

self.data2.append(da)

# 二级菜单

elif da['parent\_id']!=0 and da['type'] == 1:

self.data3.append(da)

for x in self.data2:

self.temp\_obj1 = {

"menu\_id": x['menu\_id'],

"parent\_id": x['parent\_id'],

"order\_num": x['order\_num'],

"icon": x['icon'],

"index":str(x['url'])+str(x['perms']),

"title": x['name'],

"subs":[],

"perms":[]

}

for y in self.data3:

if y['parent\_id'] == x['menu\_id']:

self.temp\_obj2 = {

"menu\_id": y['menu\_id'],

"parent\_id": y['parent\_id'],

"order\_num": y['order\_num'],

"index":str(y['url'])+str(y['perms']),

"title": y['name']

}

self.temp\_obj1["subs"].append(self.temp\_obj2)

self.temp\_obj1["subs"] = self.clean\_data\_num(self.temp\_obj1["subs"])

self.clean\_data.append(self.temp\_obj1)

self.clean\_data = self.clean\_data\_num(self.clean\_data)

return self.clean\_data,self.jur\_data,self.api\_data

def clean\_data\_num(self,data):

for i in range(len(data) - 1):

for j in range(len(data) - 1 - i):

if data[j]['order\_num'] > data[j + 1]['order\_num']:

data[j], data[j + 1] = data[j + 1], data[j]

return data

from qcloud\_cos import CosConfig

from qcloud\_cos import CosS3Client

from jox\_config import ossConf

from jox\_api.Mysql import SelectMySQL

class Upload():

def \_\_init\_\_(self):

self.config = CosConfig(

Region=ossConf['region'],

SecretId=ossConf['secret\_id'],

SecretKey=ossConf['secret\_key'],

Token=None, Scheme=ossConf['scheme']

)

self.Mysql = SelectMySQL

self.client = CosS3Client(self.config)

def upload\_pic(self,file\_name,type):

try:

with open(file\_name, 'rb') as fp:

self.response1 = self.client.put\_object(

Bucket=ossConf['Bucket\_'+type],

Body=fp,

Key=file\_name,

StorageClass='STANDARD',

EnableMD5=False

)

if self.response1['ETag']!="" :

return {"code":2003,"data":{"path":ossConf['repath\_'+type]+file\_name+ossConf['renormal'],"path\_small":ossConf['repath\_'+type]+file\_name+ossConf['resmall']}}

else:

return {"code": 2002}

except Exception as e:

print(e)

return {"code":2002,"data":{"msg":str(e)}}

from flask\_restful import reqparse, Resource,request

from jox\_api import Upload,Token,Time,User

import hashlib

import os

class UploadRoute(Resource):

def post(self):

try:

self.tokenClass = Token.Token()

self.userClass = User.User()

self.timeClass = Time.Time()

self.upload = Upload.Upload()

self.parser = reqparse.RequestParser()

self.parser.add\_argument('type', type=str, help='type: type is str')

self.args = self.parser.parse\_args()

self.type = request.headers['type']

if self.type == 'shop':

self.token = request.headers['token']

self.role\_id = request.headers['role-id']

self.validate\_token = self.tokenClass.validate(self.token, self.role\_id)

self.user\_id = self.validate\_token['user\_id']

self.file = request.files['file']

self.file\_name = self.file.filename

self.list = self.file\_name.split('.')

self.m2 = hashlib.md5()

self.m2.update((str(self.timeClass.get\_time()) + self.file\_name).encode("utf8"))

self.file\_name =str(self.user\_id)+"-" +self.m2.hexdigest()+"."+self.list[len(self.list)-1]

self.file.save(self.file\_name)

self.restatus = self.upload.upload\_pic(self.file\_name,self.type)

try:

os.remove(self.file\_name)

except Exception as e:

print(e)

return self.restatus,200

else:

self.user\_id = request.headers['user-id']

self.file = request.files['file']

self.file\_name = self.file.filename

self.list = self.file\_name.split('.')

self.m2 = hashlib.md5()

self.m2.update((str(self.timeClass.get\_time()) + self.file\_name).encode("utf8"))

self.file\_name = self.user\_id+"-"+self.m2.hexdigest() + "." + self.list[len(self.list) - 1]

self.app\_user = self.userClass.app\_user(int(self.user\_id))

if self.app\_user['code'] == 4000:

self.file.save(self.file\_name)

self.restatus = self.upload.upload\_pic(self.file\_name, "user")

else:

return self.app\_user

try:

os.remove(self.file\_name)

except Exception as e:

print(e)

return self.restatus, 200

except Exception as e:

print(e)

return {"code":-1,"data":{"msg":str(e)}},500

#脑电波处理模块

# -\*- coding: utf-8 -\*-

import serial

import requests

import matplotlib.pyplot as plt

filename = 'jox1.txt'

t = serial.Serial('COM9', 57600) #打开脑电波蓝牙串口

b = t.read(3)

status = 0

temp = 0

vaul = []

i = 0

y = 0

p = 0

joxt = [0]

joxm = [0]

joxm2 = [0]

joxi = 0

is\_open = False

plt.ion() # interactive mode

plt.figure(figsize=(8, 6), dpi=80)

try:

#检测接收到的脑电波数据是否正确

while b[0] != 170 or b[1] != 170 \

or b[2] != 4:

b = t.read(3)

if b[0] == b[1] == 170 and b[2] == 4:

a = b + t.read(5)

if a[0] == 170 and a[1] == 170 and a[2] == 4 and a[3] == 128 and a[4] == 2:

while 1:

try:

i = i + 1

# print(i)

a = t.read(8)

# print(a)

sum = ((0x80 + 0x02 + a[5] + a[6]) ^ 0xffffffff) & 0xff

if a[0] == a[1] == 170 and a[2] == 32:

y = 1

else:

y = 0

if a[0] == 170 and a[1] == 170 and a[2] == 4 and a[3] == 128 and a[4] == 2:

p = 1

else:

p = 0

if sum != a[7] and y != 1 and p != 1:

# print("wrroy1")

b = t.read(3)

c = b[0]

d = b[1]

e = b[2]

while c != 170 or d != 170 or e != 4:

c = d

d = e

e = t.read()

if c == (b'\xaa' or 170) and d == (b'\xaa' or 170) and e == b'\x04':

g = t.read(5)

if c == b'\xaa' and d == b'\xaa' and e == b'\x04' and g[0] == 128 and g[1] == 2:

a = t.read(8)

break

# if a[0]==a[1]==170 and a[2]==4:

# print(type(a))

if a[0] == 170 and a[1] == 170 and a[2] == 4 and a[3] == 128 and a[4] == 2:

high = a[5]

low = a[6]

# print(a)

rawdata = (high << 8) | low

if rawdata > 32768:

rawdata = rawdata - 65536

# vaul.append(rawdata)

sum = ((0x80 + 0x02 + high + low) ^ 0xffffffff) & 0xff

if sum == a[7]:

vaul.append(rawdata)

if sum != a[7]:

# print("wrroy2")

b = t.read(3)

c = b[0]

d = b[1]

e = b[2]

# print(b)

while c != 170 or d != 170 or e != 4:

c = d

d = e

e = t.read()

if c == b'\xaa' and d == b'\xaa' and e == b'\x04':

g = t.read(5)

if c == b'\xaa' and d == b'\xaa' and e == b'\x04' and g[0] == 128 and g[1] == 2:

a = t.read(8)

break

if a[0] == a[1] == 170 and a[2] == 32:

c = a + t.read(28)

#创建脑电波专注度+冥想度图像表

if joxi > 20: # 20次数据后，图像向后推移

joxt = joxt[-20:]

joxm = joxm[-20:]

joxm2 = joxm2[-20:]

plt.cla()

plt.grid(True) # 添加网格

plt.xlabel('times')

plt.ylabel('data')

plt.title('EEG for jox')

joxi = joxi + 1

joxt.append(i)

joxm.append(c[32])

joxm2.append(c[34])

plt.plot(joxt, joxm, "bo-", linewidth=2.0, label="attention:" + str(c[32]))

plt.plot(joxt, joxm2, "g-.", linewidth=2.0, label="meditation:" + str(c[34]))

plt.legend(loc="upper left", shadow=True)

plt.show()

if c[32] >5 and not is\_open:

print("开启灯成功")

requests.get("http://192.168.0.111/gpio", params={"oper": "startg"})

requests.get("http://192.168.0.111/gpio", params={"oper": "start"})

is\_open = True

for v in vaul:

w = 0

if v <= 102:

w += v

q = w / len(vaul)

q = " v <= 102 "+ str(q)

with open(filename, 'a') as file\_object:

file\_object.write(q)

file\_object.write("\n")

if 102 < v <= 204:

w += v

q = w / len(vaul)

q = "102 < v <= 204 "+ str(q)

with open(filename, 'a') as file\_object:

file\_object.write(q)

file\_object.write("\n")

if 204 < v <= 306:

w += v

q = w / len(vaul)

q ="204 < v <= 306 " + str(q)

with open(filename, 'a') as file\_object:

file\_object.write(q)

file\_object.write("\n")

if 306 < v <= 408:

w += v

q = w / len(vaul)

q = "306 < v <= 408 " + str(q)

with open(filename, 'a') as file\_object:

file\_object.write(q)

file\_object.write("\n")

if 408 < v <= 510:

w += v

q = w / len(vaul)

q = "408 < v <= 510 "+ str(q)

with open(filename, 'a') as file\_object:

file\_object.write(q)

file\_object.write("\n")

# print(c)

vaul = []

except Exception as e:

# print(e)

sse=1

except Exception as e:

sse = 1