**高二年级（下）信息技术第3课时（第5周）**

**《信息获取与控制（二）》学程拓展**

### 1. “远程遥控灯”参考程序

# 制作一个遥控灯，使用wifi通信和电脑端/移动终端完成远程控制

from simple import MQTTClient # 引入MQTT库

from machine import Pin # 引入基本引脚库

from machine import Timer # 引入定时器库

import ujson # 引入Json库

import network # 引入网络库

WiFI\_NAME = "<请填写wifi名称>" # 配置WiFi名，MQTT通信主题

WiFI\_PWD = "<请填写wifi密码>"

TOPIC = "/<请填写会话主题>"

CLIENT\_ID = "esp8266"

SERVER = "iotjx.51hhx.cn" # MQTT服务器(域名和端口)

PORT = 33075

led\_red = None # 声明全局变量：不同颜色LED

led\_yellow = None

led\_green = None

mMQTTClient = None

def conn\_wifi(name, pwd): # 连接wifi

 sta\_if = network.WLAN(network.STA\_IF)

 sta\_if.active(True)

 sta\_if.connect(name, pwd)

 sta\_if.ifconfig()

 while True:

 if(sta\_if.isconnected()):

 return

def sendMsg(\*self): # MQTT服务上报数据函数

 global mMQTTClient, led\_red, led\_yellow, led\_green

 status\_r = bool(led\_red.value()) # 获取状态

 status\_y = bool(led\_yellow.value())

 status\_g = bool(led\_green.value())

 obj = {'id' : CLIENT\_ID, 'r' : status\_r, 'y' : status\_y, 'g' : status\_g}

 s = ujson.dumps(obj)

 mMQTTClient.publish(TOPIC, s) # 上报给服务器

def callback(topic, msg): # MQTT服务回调函数

 global led\_red, led\_yellow, led\_green

 print((topic, msg))

 obj = ujson.loads(msg)

 id = obj["id"]

 if(id == "terminal"):

 status\_r = obj["r"]

 status\_y = obj["y"]

 status\_g = obj["g"]

 led\_red.value(status\_r)

 led\_yellow.value(status\_y)

 led\_green.value(status\_g)

 sendMsg()

def main(): # 主函数

 global mMQTTClient, led\_red, led\_yellow, led\_green

 conn\_wifi(WiFI\_NAME,WiFI\_PWD) # 连接WIFI

 led\_red = Pin(13, Pin.OUT, value = 0) # 初始化引脚

 led\_yellow = Pin(12, Pin.OUT, value = 0)

 led\_green = Pin(14, Pin.OUT, value = 0)

 mMQTTClient = MQTTClient("", SERVER, PORT) # 初始化MQTT服务

 mMQTTClient.set\_callback(callback) # 设置回调函数

 mMQTTClient.connect() # 连接MQTT服务器

 mMQTTClient.subscribe(TOPIC)

 print("Connected to %s, subscribed to %s topic" % (SERVER, TOPIC))

 timer = Timer(-1)

 timer.init(period = 2000, mode = Timer.PERIODIC, callback = sendMsg)

 try:

 while 1:

 mMQTTClient.wait\_msg() # 等待服务器发来数据

 finally:

 print("Disconnect")

 mMQTTClient.disconnect()

 timer.deinit()

main() # 调用主函数

### 2. “智能家居系统”参考程序

'''

于局域网的智能家居系统

使用局域网服务器，串口屏幕、温度/湿度传感器、彩色LED，USB-LED、蜂鸣器、按钮等

完成的一套模拟智能家居系统

由于ESP8266自身速度与容量原因，这里使用ESP32开发

'''

from microWebSrv import MicroWebSrv

import machine # 引入基本引脚库

from machine import Timer # 引入定时器库

from machine import Pin # 引入基本引脚库

import ujson # 引入Json库

import network # 引入网络库

import dht # 引入DHT11驱动

import \_thread # 引入线程

import time # 引入时间

from neopixel import NeoPixel # 导入彩灯控制

import tm1637 # 引入数码管TM1637驱动

import lcd # 导入lcd

WebSocket = None

PIN\_IN\_DHT11 = 16

PIN\_IN\_BTN = 17

PIN\_OUT\_USB = 25

PIN\_OUT\_BUZZER = 26

PIN\_OUT\_RGB = 27

PIN\_DIGITAL\_CLK = 5

PIN\_DIGITAL\_DIO = 4

PIN\_SCREEN\_TX = 22

PIN\_SCREEN\_RX = 23

def sendMsg(t, h, uv, bv): # 上报数据

 if (WebSocket == None):

 print("no ws")

 return

 obj = {'humidity': h, 'temperature': t, 'usb': uv, 'buzzer': bv} # 获取温度和湿度的值

 s = ujson.dumps(obj) # json对象转字符串

 WebSocket.SendText(s)

 print("send")

def \_acceptWebSocketCallback(webSocket, httpClient): # 注册WebSocket函数

 print("WS ACCEPT")

 webSocket.RecvTextCallback = \_recvTextCallback

 webSocket.RecvBinaryCallback = \_recvBinaryCallback

 webSocket.ClosedCallback = \_closedCallback

 global WebSocket

 WebSocket = webSocket

def \_recvTextCallback(webSocket, msg): # WebSocket收到文字回调

 print("WS RECV TEXT : %s" % msg)

 obj = ujson.loads(msg) # 将数据解析成Json对象

 usb.value(obj["usb"])

 buzzer.value(obj["buzzer"])

 temp = {'usb': usb.value(), 'buzzer': buzzer.value()}

 str = ujson.dumps(temp) # json对象转字符串

 webSocket.SendText(str)

def \_recvBinaryCallback(webSocket, data): # WebSocket收到字节码回调

 print("WS RECV DATA : %s" % data)

def \_closedCallback(webSocket): # WebSocket关闭回调函数

 print("WS CLOSED")

 global WebSocket

 if (WebSocket != None):

 WebSocket = None

d = dht.DHT11(machine.Pin(PIN\_IN\_DHT11)) # 初始化DHT11

btn = Pin(PIN\_IN\_BTN, Pin.IN) # 初始化按键

usb = Pin(PIN\_OUT\_USB, Pin.OUT, value=0) # 初始化灯

buzzer = Pin(PIN\_OUT\_BUZZER, Pin.OUT, value=0) # 初始化蜂鸣器

rgb = Pin(PIN\_OUT\_RGB, Pin.OUT) # 初始化彩灯

np = NeoPixel(rgb, 4)

display = tm1637.TM1637(clk=Pin(PIN\_DIGITAL\_CLK), dio=Pin(PIN\_DIGITAL\_DIO)) # 初始化TM1637

display.numbers(00, 00)

TRAFFIC\_LIGHT\_MODE = -1 # 交通灯

TRAFFIC\_LIGHT\_TIME = 0

TRAFFIC\_LIGHT\_TIME\_RED = 20

TRAFFIC\_LIGHT\_TIME\_YELLOW = 3

TRAFFIC\_LIGHT\_TIME\_GREEN = 20

def traffic\_light\_change\_mode(): # 改变模式

 global TRAFFIC\_LIGHT\_MODE, TRAFFIC\_LIGHT\_TIME

 if TRAFFIC\_LIGHT\_MODE == 2: # 如果模式等于2，标记一个循环结束，重新对模式赋值，进行下一次循环

 TRAFFIC\_LIGHT\_MODE = -1

 TRAFFIC\_LIGHT\_MODE = TRAFFIC\_LIGHT\_MODE + 1

 if TRAFFIC\_LIGHT\_MODE == 0:

 TRAFFIC\_LIGHT\_TIME = TRAFFIC\_LIGHT\_TIME\_RED

 if TRAFFIC\_LIGHT\_MODE == 1:

 TRAFFIC\_LIGHT\_TIME = TRAFFIC\_LIGHT\_TIME\_YELLOW

 if TRAFFIC\_LIGHT\_MODE == 2:

 TRAFFIC\_LIGHT\_TIME = TRAFFIC\_LIGHT\_TIME\_GREEN

def traffic\_light\_change():

 np[0] = (0, 0, 0) # 设置彩灯颜色RGB

 np[1] = (0, 0, 0)

 np[2] = (0, 0, 0)

 np[3] = (0, 0, 0)

 if TRAFFIC\_LIGHT\_MODE == 0:

 np[2] = (0, 255, 0)

 if TRAFFIC\_LIGHT\_MODE == 1:

 np[1] = (255, 128, 0)

 if TRAFFIC\_LIGHT\_MODE == 2:

 np[0] = (255, 0, 0)

 np.write() # 设置颜色

def task\_traffic\_light():

 global TRAFFIC\_LIGHT\_TIME

 if TRAFFIC\_LIGHT\_TIME == 0: # 如果倒计时为0，就转变模式和对应模式下的信号灯状态

 traffic\_light\_change\_mode()

 traffic\_light\_change()

 display.numbers(00, TRAFFIC\_LIGHT\_TIME) # 显示数字

 TRAFFIC\_LIGHT\_TIME = TRAFFIC\_LIGHT\_TIME - 1

screen = lcd.LCD(tx=PIN\_SCREEN\_TX, rx=PIN\_SCREEN\_RX) # 串口屏幕初始化

screen.init()

def task\_main(): # 主函数

 while True:

 d.measure()

 temp\_t = d.temperature()

 temp\_h = d.humidity()

 temp\_u = bool(usb.value())

 temp\_b = bool(buzzer.value())

 sendMsg(temp\_t, temp\_h, temp\_u, temp\_b) # 发送数据

 obj = {'temperature': temp\_t, 'humidity': temp\_h}

 screen.data(temp\_t, temp\_h) # 设置屏幕数据

 screen.state(temp\_u, temp\_b)

 print(obj)

 task\_traffic\_light()

 time.sleep(1)

def task\_button(): # 按键检测函数，单击和长按

 while True:

 if btn.value() == 0:

 begin = time.ticks\_ms()

 while True:

 if btn.value() == 1:

 if time.ticks\_ms() - begin > 20:

 print ('short')

 usb.value(not usb.value())

 break

 if time.ticks\_ms() - begin > 1000:

 print ('long')

 buzzer.value(not buzzer.value())

 while True: # 等着松手

 if btn.value() == 1:

 break

 break

\_thread.start\_new\_thread(task\_main, ()) # 启动线程

\_thread.start\_new\_thread(task\_button, ())

wlan = network.WLAN(network.STA\_IF) # 连接wifi

wlan.active(True)

'''

此处可以连接WiFi，但是没连

wlan.connect('<请填写wifi名称>', '<请填写wifi密码>')

展示wifi信息，主要用于显示连接后的ip地址

wlan.ifconfig()

获取Mac地址

'''

s = wlan.config('mac')

temp = ('%02x%02x') % (s[4], s[5])

ap\_name = ('ESP32-AP-' + temp).upper() # 设置热点名称

ap = network.WLAN(network.AP\_IF) # 开启ap模式

ap.active(True)

ap.config(essid=ap\_name)

srv = MicroWebSrv(webPath='www/') # 开启服务器

srv.MaxWebSocketRecvLen = 256

srv.WebSocketThreaded = False

srv.AcceptWebSocketCallback = \_acceptWebSocketCallback

srv.Start()