//头文件

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <NewPing.h>

//wifiiiiiiiiiiiiiiii

#define SSID "替换为你的热点名" //改为你的热点名称, 不要有中文

#define PASSWORD "替换为你的热点密码"//改为你的WiFi密码Wi-Fi密码

#define DEVICEID "替换为你的OneNet平台上设备ID" //OneNet上的设备ID

String apiKey = "替换为你的APIKey";//与你的设备绑定的APIKey

/\*\*\*/

#define HOST\_NAME "api.heclouds.com"

#define HOST\_PORT (80)

#define INTERVAL\_SENSOR 5000 //定义传感器采样时间间隔 597000

#define INTERVAL\_NET 5000 //定义发送时间

/\*\*\*/

//传感器部分================================

#include <Wire.h> //调用库

#include <ESP8266.h>

#include <I2Cdev.h> //调用库

/\*\*\*\*\*\*\*温湿度\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*光照\*\*\*\*\*\*\*/

#define IDLE\_TIMEOUT\_MS 3000 // Amount of time to wait (in milliseconds) with no data

// received before closing the connection. If you know the server

// you're accessing is quick to respond, you can reduce this value.

//WEBSITE

char buf[10];

#define INTERVAL\_sensor 2000

unsigned long sensorlastTime = millis();

float tempOLED, humiOLED, lightnessOLED;

#define INTERVAL\_OLED 1000

String mCottenData;

String jsonToSend;

//3,传感器值的设置

float sensor\_tem, sensor\_hum, sensor\_lux; //传感器温度、湿度、光照

char sensor\_tem\_c[7], sensor\_hum\_c[7], sensor\_lux\_c[7] ; //换成char数组传输

#include <SoftwareSerial.h>

#define EspSerial mySerial

#define UARTSPEED 9600

SoftwareSerial mySerial(2, 3); /\* RX:D3, TX:D2 \*/

ESP8266 wifi(&EspSerial);

//ESP8266 wifi(Serial1); //定义一个ESP8266（wifi）的对象

unsigned long net\_time1 = millis(); //数据上传服务器时间

unsigned long sensor\_time = millis(); //传感器采样时间计时器

//int SensorData; //用于存储传感器数据

String postString; //用于存储发送数据的字符串

//String jsonToSend; //用于存储发送的json格式参数

////HX711压力传感器：

///函数

extern unsigned long HX711\_Read(void);

extern long Get\_Weight();

///变量定义

float Weight = 0;

int HX711\_SCK =2; /// 作为输出口

int HX711\_DT= 3; /// 作为输入口

long HX711\_Buffer = 0;

long Weight\_Maopi = 0, Weight\_Shiwu = 0;

#define GapValue 405 ///该值需校准 每个传感器都有所不同

//初始化HX711的两个io口

////HC-SR04超声波测距传感器

LiquidCrystal\_I2C lcd(0x27,16,2);//0x27 0x3F

#define TRIGGER\_PIN 4 // Arduino pin tied to trigger pin on the ultrasonic sensor.

#define ECHO\_PIN 5 // Arduino pin tied to echo pin on the ultrasonic sensor.

#define MAX\_DISTANCE 400 // Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-500cm.

NewPing sonar(TRIGGER\_PIN, ECHO\_PIN, MAX\_DISTANCE); // NewPing setup of pins and maximum distance.

////人体红外感应传感器

///变量定义

int a =600; // 阈值。根据实际情况进行调整。使得光线强时小于此值，弱时大于此值。

const int sensorPin = A0; //定义人体红外传感器的引脚

const int ledPin = 10;

//定义LED灯的引脚

int sensorValue = 0; //声明传感器数据变量

void setup() {

Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.

lcd.init();

lcd.backlight();

Weight\_Maopi = HX711\_Read(); //压力传感器

pinMode (HX711\_SCK, OUTPUT); ///SCK 为输出口 ---输出脉冲

pinMode (HX711\_DT, INPUT); /// DT为输入口 ---读取数据

pinMode(ledPin, OUTPUT); ///LED输出口

digitalWrite(ledPin,LOW); ///初始化LED

pinMode(sensorPin, INPUT); ///人体红外感应传感器 输入口

//后面是wifi加的

//初始化串口波特率

Wire.begin();

Serial.begin(115200);

while (!Serial); // wait for Leonardo enumeration, others continue immediately

Serial.print(F("setup begin\r\n"));

delay(100);

pinMode(sensorPin\_1, INPUT);

WifiInit(EspSerial, UARTSPEED);

Serial.print(F("FW Version:"));

Serial.println(wifi.getVersion().c\_str());

if (wifi.setOprToStationSoftAP()) {

Serial.print(F("to station + softap ok\r\n"));

} else {

Serial.print(F("to station + softap err\r\n"));

}

if (wifi.joinAP(SSID, PASSWORD)) {

Serial.print(F("Join AP success\r\n"));

Serial.print(F("IP:"));

Serial.println( wifi.getLocalIP().c\_str());

} else {

Serial.print(F("Join AP failure\r\n"));

}

if (wifi.disableMUX()) {

Serial.print(F("single ok\r\n"));

} else {

Serial.print(F("single err\r\n"));

}

Serial.print(F("setup end\r\n"));

}

void loop() /// 一直循环{}内容 ----- 同while（1）{xxx}

{

//wifi

if (sensor\_time > millis())

sensor\_time = millis();

if(millis() - sensor\_time > INTERVAL\_SENSOR) //传感器采样时间间隔

{

getSensorData(); //读串口中的传感器数据

sensor\_time = millis();

}

if (net\_time1 > millis()) net\_time1 = millis();

if (millis() - net\_time1 > INTERVAL\_NET) //发送数据时间间隔

{

updateSensorData(); //将数据上传到服务器的函数

net\_time1 = millis();

}

////HX711压力传感器

Weight = Get\_Weight(); //计算放在传感器上的重物重量

Serial.print(float(Weight/1000),3); //串口显示重量， 3意为保留三位小数

Serial.print(" kg\n"); //显示单位

Serial.print("\n"); //显示单位

delay(2000); //延时2s 两秒读取一次传感器所受压力

////HC-SR04超声波测距传感器

delay(100); // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.

unsigned int uS = sonar.ping(); // Send ping, get ping time in microseconds (uS).

Serial.print("Ping: ");

Serial.print(uS / US\_ROUNDTRIP\_CM); // Convert ping time to distance in cm and print result (0 = outside set distance range)

Serial.println("cm");

/\*lcd.setCursor(0, 0);

lcd.print("Distance:");

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.setCursor(9, 1);

lcd.print(uS / US\_ROUNDTRIP\_CM);

lcd.setCursor(12, 1);

lcd.print("cm");\*/

if(uS / US\_ROUNDTRIP\_CM<3)

lcd.print("Full");

////人体红外感应传感器

int n = analogRead(A0); //读取模拟口A0数值

Serial.println(n);

if (n>= a ) //对光线强度进行判断，小于预设值关闭LED,否则就点亮

{

sensorValue = analogRead(sensorPin); //读取传感器数据

if(sensorValue > 300){ //判断是否有人接近

digitalWrite(ledPin,HIGH);

delay(30000); //延时30秒

digitalWrite(ledPin,LOW); //关闭LED灯

}

else

{

digitalWrite(ledPin,LOW);

}

delay(100);

}

}

////压力传感器称重函数

long Get\_Weight()

{

HX711\_Buffer = HX711\_Read(); ///读取此时的传感器输出值

Weight\_Shiwu = HX711\_Buffer; ///将传感器的输出值储存

Weight\_Shiwu = Weight\_Shiwu - Weight\_Maopi; //获取实物的AD采样数值。

Weight\_Shiwu = (long)((float)Weight\_Shiwu/GapValue); //AD值转换为重量（g）

return Weight\_Shiwu;

}

unsigned long HX711\_Read(void) //选择芯片工作方式并进行数据读取

{

unsigned long count; ///储存输出值 plc四字节

unsigned char i; /// plc两字节

////high--高电平 1 low--低电平 0 plc通过断开与否控制？

digitalWrite(HX711\_DT, HIGH); //// digitalWrite作用： DT=1；

delayMicroseconds(1); ////延时 1微秒

digitalWrite(HX711\_SCK, LOW); //// digitalWrite作用： SCK=0；

delayMicroseconds(1); ////延时 1微秒

count=0;

while(digitalRead(HX711\_DT)); //当DT的值为1时，开始ad转换

for(i=0;i<24;i++) ///24个脉冲，对应读取24位数值

{

digitalWrite(HX711\_SCK, HIGH); //// digitalWrite作用： SCK=0；

/// 利用 SCK从0--1 ，发送一次脉冲，读取数值

delayMicroseconds(1); ////延时 1微秒

count=count<<1; ///用于移位存储24位二进制数值

digitalWrite(HX711\_SCK, LOW); //// digitalWrite作用： SCK=0；为下次脉冲做准备

delayMicroseconds(1);

if(digitalRead(HX711\_DT)) ///若DT值为1，对应count输出值也为1

count++;

}

digitalWrite(HX711\_SCK, HIGH); ///再来一次上升沿 选择工作方式 128增益

count ^= 0x800000; //按位异或 不同则为1 0^0=0; 1^0=1;

///对应二进制 1000 0000 0000 0000 0000 0000 作用为将最高位取反，其他位保留原值

delayMicroseconds(1);

digitalWrite(HX711\_SCK, LOW); /// SCK=0；

delayMicroseconds(1); ////延时 1微秒

return(count); ///返回传感器读取值

}

void getSensorData(){

sensor\_tem = Weight; //重量(原来是温度)

sensor\_hum = uS / US\_ROUNDTRIP\_CM; // 已满(原来是hum)

// sensor\_lux = analogRead(A0);

delay(1000);

dtostrf(sensor\_tem, 2, 1, sensor\_tem\_c);

dtostrf(sensor\_hum, 2, 1, sensor\_hum\_c);

//dtostrf(sensor\_lux, 3, 1, sensor\_lux\_c);

}

void updateSensorData() {

if (wifi.createTCP(HOST\_NAME, HOST\_PORT)) { //建立TCP连接，如果失败，不能发送该数据

Serial.print("create tcp ok\r\n");

jsonToSend="{\"Temperature\":";

dtostrf(sensor\_tem,1,2,buf);

jsonToSend+="\""+String(buf)+"\"";

jsonToSend+=",\"Humidity\":";

dtostrf(sensor\_hum,1,2,buf);

jsonToSend+="\""+String(buf)+"\"";

jsonToSend+=",\"Light\":";

dtostrf(sensor\_lux,1,2,buf);

jsonToSend+="\""+String(buf)+"\"";

jsonToSend+="}";

postString+="/datapoints?type=3 HTTP/1.1";

postString+="\r\n";

postString+="api-key:";

postString+=apiKey;

postString+="\r\n";

postString+="Host:api.heclouds.com\r\n";

postString+="Connection:close\r\n";

postString+="Content-Length:";

postString+=jsonToSend.length();

postString+="\r\n";

postString+="\r\n";

postString="POST /devices/";

postString+=DEVICEID;

postString+=jsonToSend;

postString+="\r\n";

postString+="\r\n";

postString+="\r\n";

const char \*postArray = postString.c\_str(); //将str转化为char数组

Serial.println(postArray);

wifi.send((const uint8\_t\*)postArray, strlen(postArray)); //send发送命令，参数必须是这两种格式，尤其是(const uint8\_t\*)

Serial.println("send success");

if (wifi.releaseTCP()) { //释放TCP连接

Serial.print("release tcp ok\r\n");

}

else {

Serial.print("release tcp err\r\n");

}

postArray = NULL; //清空数组，等待下次传输数据

} else {

Serial.print("create tcp err\r\n");

}

}