浙大城市学院实验报告

课程名称 　物联网技术与应用　实验项目　实验五　ESP8266WIFI 基础实验

专业班级 　计算机 1803　　学号　31801150　　姓名

张帅

指导老师（签名 ）　蔡建平　日期　2020.11.2　　实验成绩 ______

注意：

- 务请保存好各自的源代码及实验报告文档，已备后用。
- 请把实验报告转为 PDF 文档上传到 BB 平台。
- 文件名格式：学号_姓名_日期_实验，如 30801001_姓名_20200305_实验 02

一、实验目的：

熟悉 OLED 显示模块 SSD1306 的使用，熟悉 128*64 点阵分辨率的 OLED 的显示控制，掌握显示各类对象的函数及参数的用法，掌握屏幕显示坐标计算。

二、实验内容：

1. IoT Client to PC server；
2. PC Client to IoT Server，OLED 屏显示收到的内容；
3. IoT Client to PC Server，将 ADXL345 数据发往 PC Server。

三、实验步骤：

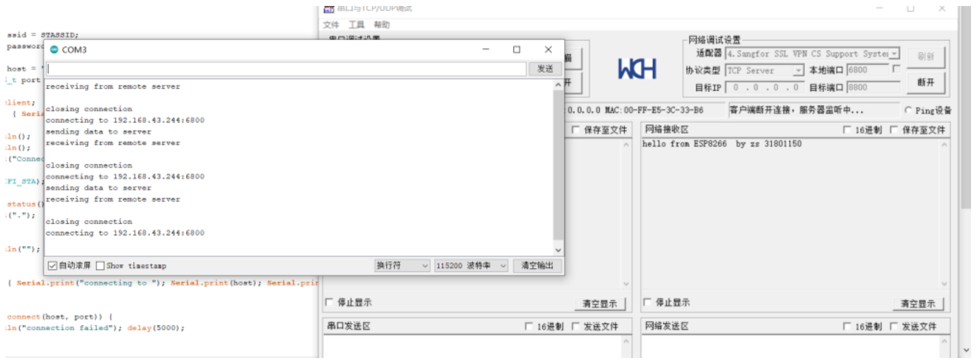
1. 将 ESP8266 设为 IoT Client，通过 PC 端工具（PC server）接收来自 ESP8266 的消息。

完整代码：

```
#include <ESP8266WiFi.h>
#ifndef STASSID
#define STASSID "Mi 10000 Ultra"
#define STAPSK    "88888888"
#endif

const char* ssid = STASSID;
const char* password = STAPSK;

const char* host = "192.168.43.244";
const uint16_t port = 6800;
```

```
WiFiClient client;
void setup() { Serial.begin(115200);

Serial.println();
Serial.println();
Serial.print("Connecting to "); Serial.println(ssid);

WiFi.mode(WIFI_STA); WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) { delay(500);
Serial.print(".");
}

Serial.println("");    Serial.println("WiFi    connected");    Serial.println("IP    address:    ");
Serial.println(WiFi.localIP());
}

void    loop()    {    Serial.print("connecting    to    ");    Serial.print(host);    Serial.print(':');
Serial.println(port);



if (!client.connect(host, port)) {
Serial.println("connection failed"); delay(5000);
return;
}



Serial.println("sending data to server");
if (client.connected()) {
client.println("hello from ESP8266   by zs 31801150");
}
delay(6000);
Serial.println("receiving from remote server");

while (client.available()) {
   char ch = static_cast<char>(client.read());
   Serial.print(ch);
}
// Close the connection
Serial.println();
Serial.println("closing connection");
client.stop();
delay(3000); // execute once every 5 minutes, don't flood remote service
}
```

截图:



```
receiving from remote server

closing connection
connecting to 192.168.43.244:6800
sending data to server
receiving from remote server

closing connection
connecting to 192.168.43.244:6800
sending data to server
receiving from remote server

closing connection
connecting to 192.168.43.244:6800
```

适配器: 4.Sangfor SSL VPN CS Support Syste
协议类型: TCP Server　本地端口: 6800
目标IP: 0 . 0 . 0 . 0　目标端口: 8800
0.0.0.0 MAC:00-FF-E5-3C-33-B6
客户端断开连接，服务器监听中...

网络接收区:

```
hello from ESP8266  by zs 31801150
```

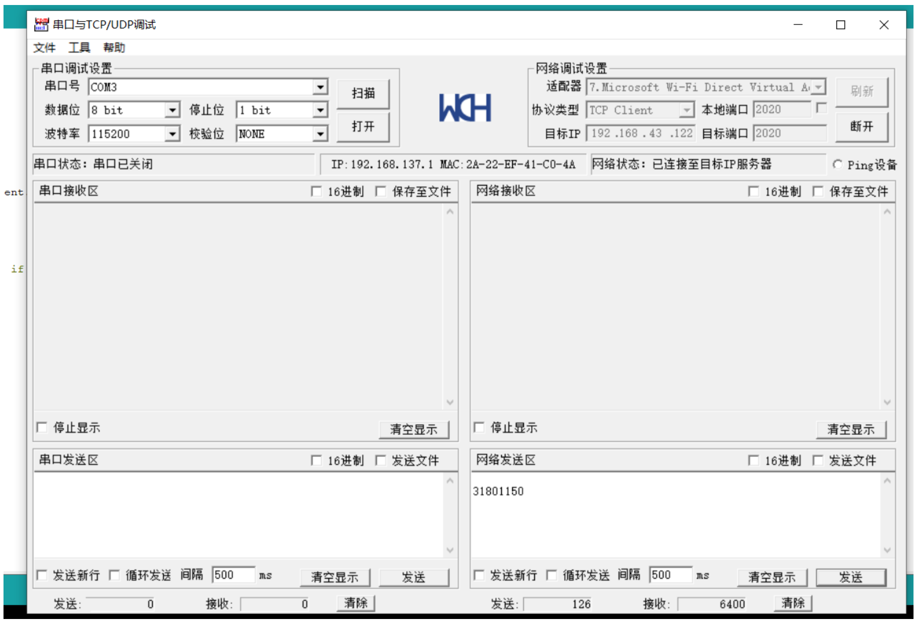
2. PC Client to IoT Server，将 ESP8266 设为 Server，从 PC 端 TCP Server 工具向 ESP8266 发送数据，在 OLED 屏上显示所接收到数据并支持内容持续刷新。

完整代码:

```
#include <ESP8266WiFi.h>
#include"SSD1306Wire.h"
SSD1306Wire display(0x3c,2,14);

#define MAX_SRV_CLIENTS 4
#define DebugBegin(baud_rate)    Serial.begin(baud_rate)
#define DebugPrintln(message) Serial.println(message)
#define DebugPrint(message) Serial.print(message)

const char* ssid = "Mi 10000 Ultra";
const char* password = "88888888";


WiFiServer server(2020);
WiFiClient   serverClients[MAX_SRV_CLIENTS];


void setup() {
    display.init();
  DebugBegin(115200);        WiFi.mode(WIFI_STA);        WiFi.begin(ssid,        password);
DebugPrint("\nConnecting to "); DebugPrintln(ssid);
  uint8_t i = 0;
  while (WiFi.status() != WL_CONNECTED && i++ < 20)
  { delay(500);
  }
```

```
    if (i == 21) {
    DebugPrint("Could not connect to"); DebugPrintln(ssid);
    while (1) { delay(500);
    }
}
     server.begin();
     server.setNoDelay(true);

    DebugPrint("Ready! Use 'telnet ");
    DebugPrint(WiFi.localIP());
    DebugPrintln(" 2020' to connect");
}

void loop(){
uint8_t i;

if (server.hasClient()) {
for (i = 0; i < MAX_SRV_CLIENTS; i++) {

if (!serverClients[i] || !serverClients[i].connected()) { if (serverClients[i]) {
serverClients[i].stop();
}

serverClients[i] = server.available();
DebugPrint("New client: ");
DebugPrint(i);
break;
}
}

if (i == MAX_SRV_CLIENTS) {
WiFiClient serverClient = server.available(); serverClient.stop(); DebugPrintln("Connection
rejected ");
}
}

for (i = 0; i < MAX_SRV_CLIENTS; i++) {
if (serverClients[i] && serverClients[i].connected()) { if (serverClients[i].available()) {
    display.flipScreenVertically();
    display.setFont(ArialMT_Plain_16);
    display.clear();
  String str="";
while (serverClients[i].available()) {
  str+=(char)serverClients[i].read();
//    Serial.write(serverClients[i].read());
}
    display.drawString(0,0,str);
    display.display();
    delay(2000);
}
}
}

if (Serial.available()) {

size_t len = Serial.available();
uint8_t sbuf[len]; Serial.readBytes(sbuf, len);

for (i = 0; i < MAX_SRV_CLIENTS; i++) {
if (serverClients[i] && serverClients[i].connected()) { serverClients[i].write(sbuf, len);
delay(1);
}
}
}
}
```

截图:



串口与TCP/UDP调试

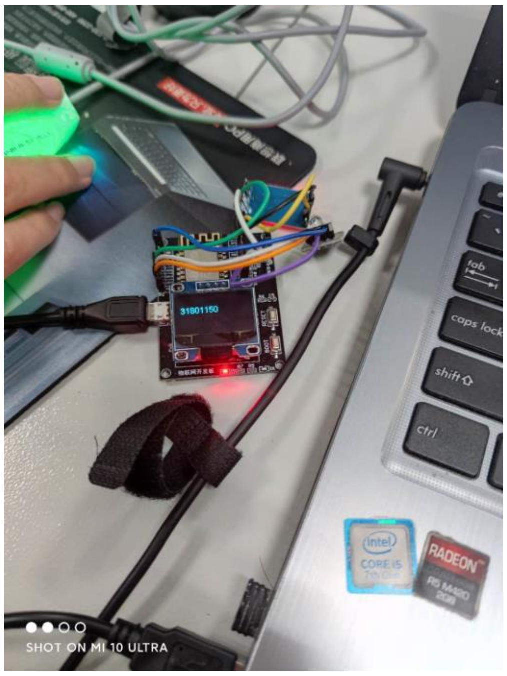
串口调试设置
串口号 COM3
数据位 8 bit　停止位 1 bit
波特率 115200　校验位 NONE

网络调试设置
适配器 7.Microsoft Wi-Fi Direct Virtual A
协议类型 TCP Client　本地端口 2020
目标IP 192 .168 . 43 .122　目标端口 2020

串口状态: 串口已关闭
IP: 192. 168. 137. 1 MAC: 2A-22-EF-41-C0-4A
网络状态: 已连接至目标IP服务器

网络发送区
31801150

间隔 500 ms　间隔 500 ms
发送: 0　接收: 0
发送: 126　接收: 6400



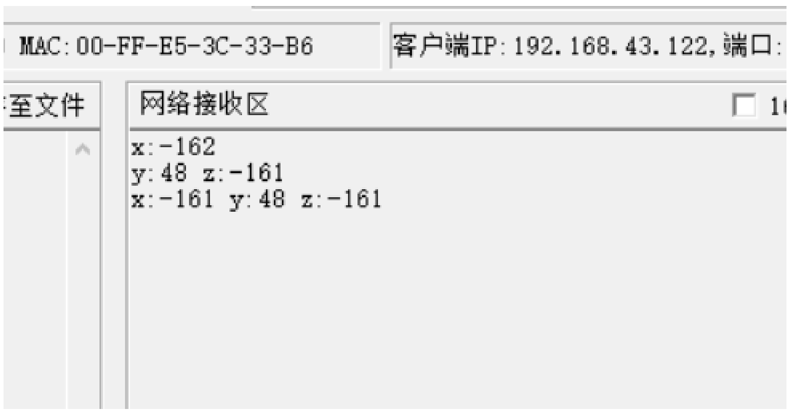
3. IoT Client to PC Server，将 ADXL345 数据发往 PC Server

完整代码：

```
#include <ESP8266WiFi.h>
#ifndef STASSID
#define STASSID "Mi 10000 Ultra"
#define STAPSK     "88888888"
#endif

const char* ssid = STASSID;
const char* password = STAPSK;

const char* host = "192.168.43.244";
const uint16_t port = 6800;

WiFiClient client;
#include<Wire.h>
#include"SSD1306Wire.h"
#define Addr 0x53
SSD1306Wire display(0x3c,2,14);
const uint8_t scl=2;
const uint8_t sda=14;

unsigned int data[6];
int xAccl;
int yAccl;
int zAccl;

void axdl(){
    Wire.beginTransmission(Addr);
    Wire.begin(sda,scl);
    Wire.write(0x2C);
    Wire.write(0x0A);
    Wire.endTransmission();

    Wire.beginTransmission(Addr);
    Wire.write(0x2D);
    Wire.write(0x08);
    Wire.endTransmission();



    Wire.beginTransmission(Addr);
    Wire.write(0x31);
    Wire.write(0x08);
    Wire.endTransmission();
    display.flipScreenVertically();
    display.setFont(ArialMT_Plain_16);

      for(int i=0;i<6;i++)
    {
        Wire.beginTransmission(Addr);
        Wire.write((50+i));
        Wire.endTransmission();
        Wire.requestFrom(Addr,1);

        if(Wire.available()==1)
        {
            data[i]=Wire.read();
        }
    }

    xAccl=(((data[1]&0x03)*256)+data[0]);
    if(xAccl>511){
        xAccl-=1024;
    }
      yAccl=(((data[3]&0x03)*256)+data[2]);
    if(yAccl>511){
        yAccl-=1024;
    }
      zAccl=(((data[5]&0x03)*256)+data[4]);
    if(zAccl>511){
        zAccl-=1024;
    }




}




void setup() { Serial.begin(115200);

Serial.println();
Serial.println();
Serial.print("Connecting to "); Serial.println(ssid);

WiFi.mode(WIFI_STA); WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) { delay(500);
Serial.print(".");
}

Serial.println("");    Serial.println("WiFi    connected");    Serial.println("IP    address:    ");
Serial.println(WiFi.localIP());
}

void    loop()    {    Serial.print("connecting    to    ");    Serial.print(host);    Serial.print(':');
Serial.println(port);

if (!client.connect(host, port)) {
Serial.println("connection failed"); delay(5000);
return;
}



Serial.println("sending data to server");
if (client.connected()) {
    axdl();
    client.printf("x:%d \n",xAccl);
    client.printf("y:%d \n",yAccl);
    client.printf("z:%d \n",zAccl);
}
delay(6000);
Serial.println("receiving from remote server");
```

```
while (client.available()) {
    char ch = static_cast<char>(client.read());
    Serial.print(ch);
}
// Close the connection
Serial.println();
Serial.println("closing connection");
client.stop();
delay(3000); // execute once every 5 minutes, don't flood remote service
}
```

截图:


MAC: 00-FF-E5-3C-33-B6　客户端IP: 192.168.43.122,端口:

至文件　网络接收区

```
x:-162
y:48 z:-161
x:-161 y:48 z:-161
```

4. 访问 API，获取 JSON 并提取所需的字段，申请个人的心知天气的 ID (https://www.seniverse.com/)，通过 TCP client 包装 HTTP 请求协议去调用天气接口获取天气信息。

本人心知天气的 host:

https://www.seniverse.com/products?iid=006c072b-1ca9-42f2-98f1-bb6b1946aef8

本人心知天气的 APIKEY:

SBD3OTFNJDBK1ibtX

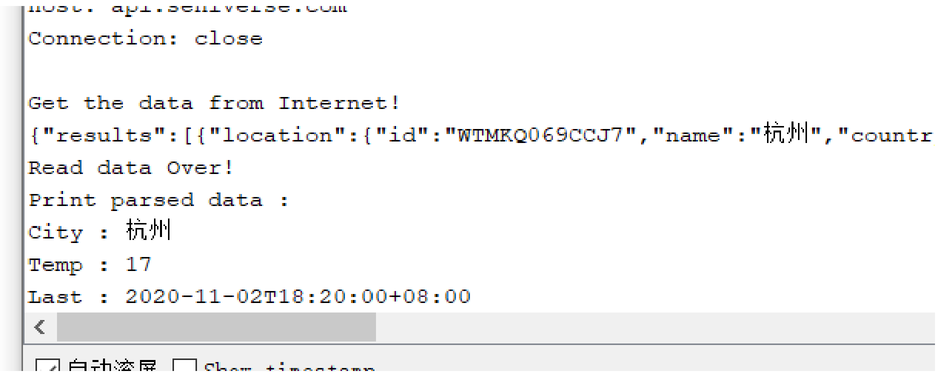
完整代码:

```
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>
```

```
//以下三个定义为调试定义
#define DebugBegin(baud_rate)    Serial.begin(baud_rate)
#define DebugPrintln(message) Serial.println(message)
#define DebugPrint(message) Serial.print(message)

const char* ssid = "Mi 10000 Ultra";
const char* password = "88888888";
const char* host = "api.seniverse.com";
const char* APIKEY = "SBD3OTFNJDBK1ibtX";    //API KEY
const char* city = "hangzhou";
const char* language = "zh-Hans";//zh-Hans   简体中文  会显示乱码

const unsigned long BAUD_RATE = 115200;
const unsigned long HTTP_TIMEOUT = 5000;
const size_t MAX_CONTENT_SIZE = 1000;


struct WeatherData {
  char city[16];//城市名称
char weather[32];//天气介绍（多云...）
char temp[16];//温度
char udate[32];//更新时间
};

WiFiClient client;
char response[MAX_CONTENT_SIZE];
char endOfHeaders[] = "\r\n\r\n";

void setup() {
// put your setup code here, to run once: WiFi.mode(WIFI_STA);   //设置 esp8266 工作模
式
DebugBegin(BAUD_RATE);
DebugPrint("Connecting to ");//提示
DebugPrintln(ssid);
WiFi.begin(ssid, password); //连接
  WiFi.setAutoConnect(true);
while (WiFi.status() != WL_CONNECTED) {
//这个函数是 wifi 连接状态，返回 wifi 链接状态
delay(500); DebugPrint(".");
}
DebugPrintln(""); DebugPrintln("WiFi connected"); delay(500);
DebugPrintln("IP address: "); DebugPrintln(WiFi.localIP());//WiFi.localIP()返回 8266 获得的
ip 地址
client.setTimeout(HTTP_TIMEOUT);
}

void loop() {
// put your main code here, to run repeatedly:
//判断 tcp client 是否处于连接状态，不是就建立连接
while (!client.connected()){
if (!client.connect(host, 80)){ DebugPrintln("connection...."); delay(500);
}
}
//发送 http 请求 并且跳过响应头 直接获取响应 body
if (sendRequest(host, city, APIKEY) && skipResponseHeaders()) {
//清除缓冲
clrEsp8266ResponseBuffer();
//读取响应数据
readReponseContent(response, sizeof(response)); WeatherData weatherData;
if (parseUserData(response, &weatherData)) { printUserData(&weatherData);
}

}
delay(5000);//每 5s 调用一次
}

/**
* @发送 http 请求指令
*/
bool sendRequest(const char* host, const char* cityid, const char* apiKey) {
// We now create a URI for the request
//心知天气　发送 http 请求
String GetUrl = "/v3/weather/now.json?key="; GetUrl += apiKey;
GetUrl += "&location="; GetUrl += city;
GetUrl += "&language="; GetUrl += language;
// This will send the request to the server
client.print(String("GET ") + GetUrl + " HTTP/1.1\r\n" +
"Host: " + host + "\r\n" + "Connection: close\r\n\r\n");
DebugPrintln("create a request:"); DebugPrintln(String("GET ") + GetUrl + " HTTP/1.1\r\n"
+
"Host: " + host + "\r\n" + "Connection: close\r\n");
delay(1000); return true;
}

/**
* @Desc　跳过 HTTP　头，使我们在响应正文的开头
*/
```

```
bool skipResponseHeaders() {
// HTTP headers end with an empty line
bool ok = client.find(endOfHeaders);
if (!ok) {
DebugPrintln("No response or invalid response!");
}
return ok;
}

/**
* @Desc  从 HTTP 服务器响应中读取正文
*/
void readReponseContent(char* content, size_t maxSize) { size_t length = client.readBytes(content, maxSize); delay(100);
DebugPrintln("Get the data from Internet!"); content[length] = 0;
DebugPrintln(content); DebugPrintln("Read data Over!"); client.flush();//清除一下缓冲
}

/**
  * @Desc  解析数据 Json 解析
  *  数据格式如下:
  * {
"results": [
*  {
* "location": {
* "id": "WX4FBXXFKE4F",
* "name": "北京",
* "country": "CN",
* "path": "北京,北京,中国",
* "timezone": "Asia/Shanghai",
* "timezone_offset": "+08:00"
* },
* "now": {
* "text": "多云",
* "code": "4",
* "temperature": "23"
* },
* "last_update":  "2017-09-13T09:51:00+08:00"
* }
  * ]
  *}
  */
bool parseUserData(char* content, struct WeatherData* weatherData) {
//  --  根据我们需要解析的数据来计算 JSON 缓冲区最佳大小
//  如果你使用 StaticJsonBuffer 时才需要
//  const size_t BUFFER_SIZE = 1024;
//  在堆栈上分配一个临时内存池
//  StaticJsonBuffer<BUFFER_SIZE>   jsonBuffer;
//  --  如果堆栈的内存池太大，使用 DynamicJsonBuffer jsonBuffer   代替
DynamicJsonBuffer jsonBuffer;

JsonObject& root = jsonBuffer.parseObject(content);

if (!root.success()) { DebugPrintln("JSON parsing failed!"); return false;
}

//复制感兴趣的字符串
strcpy(weatherData->city, root["results"][0]["location"]["name"]);
strcpy(weatherData->weather,     root["results"][0]["now"]["text"]);
strcpy(weatherData->temp,     root["results"][0]["now"]["temperature"]);
strcpy(weatherData->udate, root["results"][0]["last_update"]);

//获取温度、最近更新时间

//        --  这不是强制复制，你可以使用指针，因为他们是指向“内容”缓冲区内，所以你需要确保
//  当你读取字符串时它仍在内存中
return true;
}

// 打印从 JSON 中提取的数据
void printUserData(const struct WeatherData* weatherData) {
  DebugPrintln("Print parsed data :");
DebugPrint("City : "); DebugPrint(weatherData->city);
DebugPrint("\nTemp : "); DebugPrint(weatherData->temp);
DebugPrint("\nLast : "); DebugPrint(weatherData->udate);

//打印天气、气温、最近更新时间

}
// 关闭与 HTTP 服务器连接
void stopConnect() { DebugPrintln("Disconnect"); client.stop();
}

void clrEsp8266ResponseBuffer(void){ memset(response, 0, MAX_CONTENT_SIZE);}
```

截图：



```
Host: api.seniverse.com
Connection: close

Get the data from Internet!
{"results":[{"location":{"id":"WTMKQ069CCJ7","name":"杭州","countr
Read data Over!
Print parsed data :
City : 杭州
Temp : 17
Last : 2020-11-02T18:20:00+08:00
```