**ESP8266学习记录**

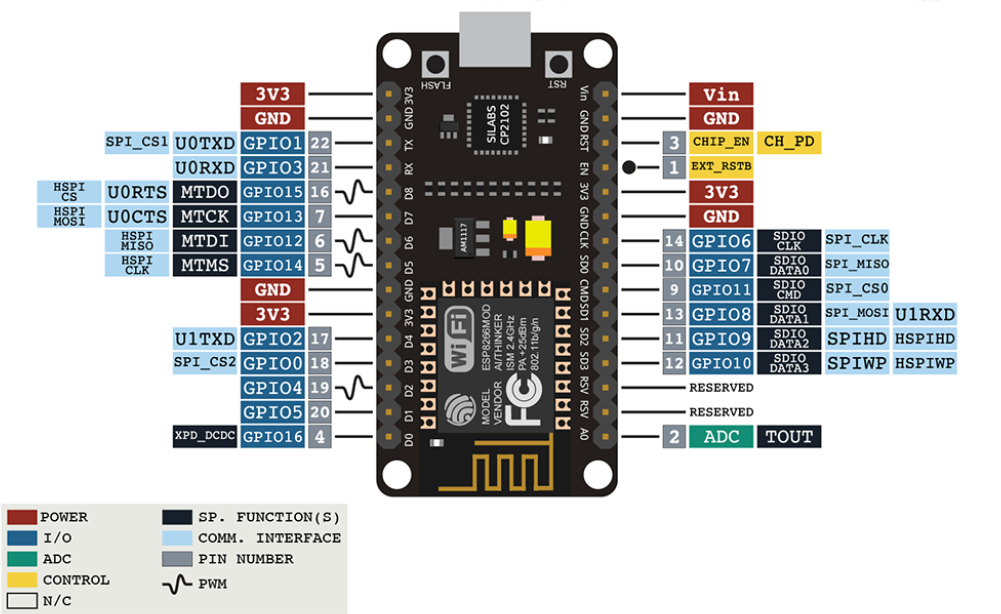
*——本学习记录借助Arduino进行开发*

**学习人：陈浩**

**参考资料：**[**www.taichi-maker.com**](http://www.taichi-maker.com)

**学习时间：2022-1-20——2022-1-29**

**一、基础知识篇**





工作电压：3.3V

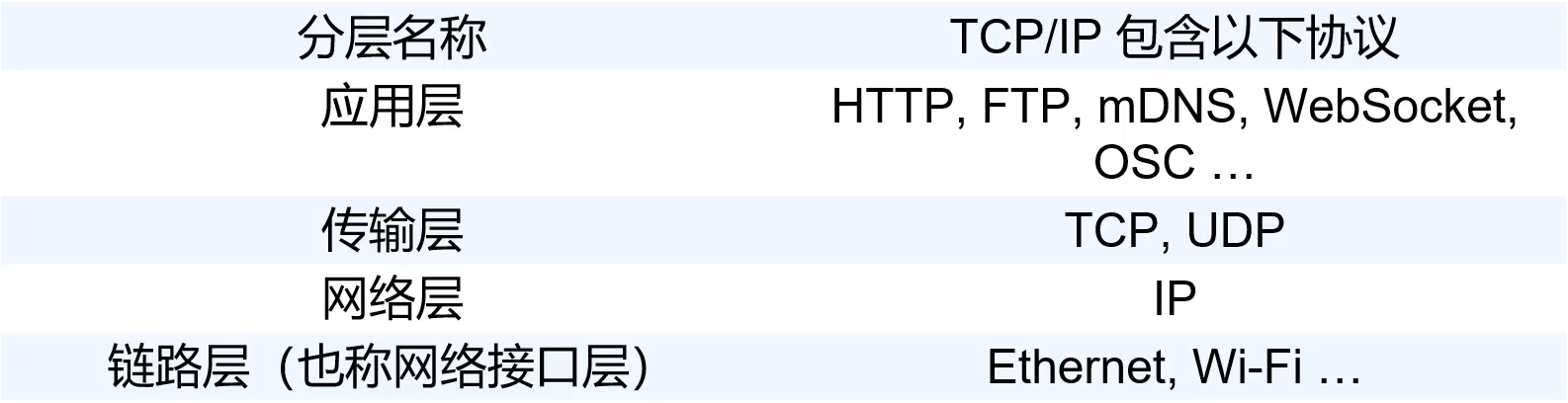
模拟引脚A0可读取电压范围：0~1V

digitalWrite(D0,HIGH)与digitalWrite(16,HIGH)效果一致

print函数与println函数类似，后者带换行前者不带

（一）、ESP8266物联网开发基础

**1、互联网协议（TCP/IP协议）**



* 链路层：实现设备之间物理连接，ESP8266利用WIFI联网有三种工作模式：

1）、接入点模式：ESP8266向外界发送WIFI信号，手机、电脑可以和

ESP8266所发射的WIFI信号进行连接

2）、无线终端模式：ESP8266作为无线终端连入WIFI

3）、混合模式：接入点模式与无线终端模式混合在一起，即可以用无

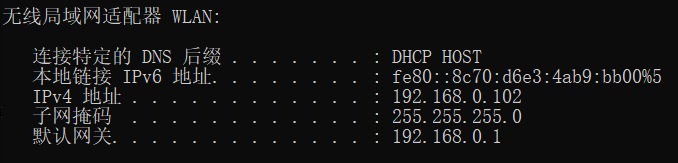
线终端模式连接WIFI，也可以自己也可以发射一个WIFI网络

* 网络层：为网络设备提供地址，IP协议版本：IPV4（四个十进制数组成：

192.168.0.3）和IPV6（八组十六进制数组成）

在windows命令提示符中输入ipconfig即可查看IPV4地址、子网掩码

和默认网关



IPv4地址：此电脑的IP地址

子网掩码：设备的IP地址相同部分对应子网掩码是255的部分，而设

备IP地址不同的部分对应子网掩码0。

默认网关：在此局域网中WIFI路由器的IP地址

* 传输层：主要有TCP、UDP协议组成

1）、TCP协议：特点稳；保证所有数据都能被接受端接受；数据传输

顺序不会被打乱；传输数据若有损坏则重发受损数据

2）、UDP协议：特点快；不能保证所有数据都能被接受端接受；数据

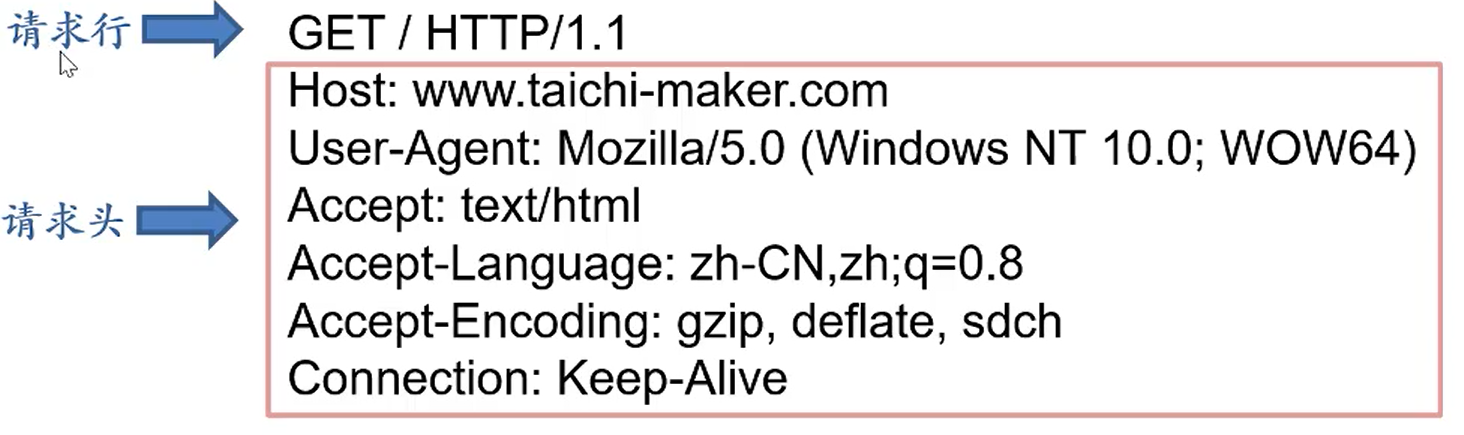
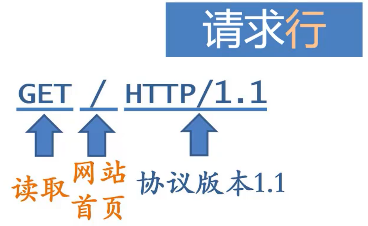
一旦损坏，UDP协议会抛弃受损数据；数据损坏不会重发

* 应用层：包括很多协议，主要学习HTTP协议

1)、HTTP协议：由请求与响应构成

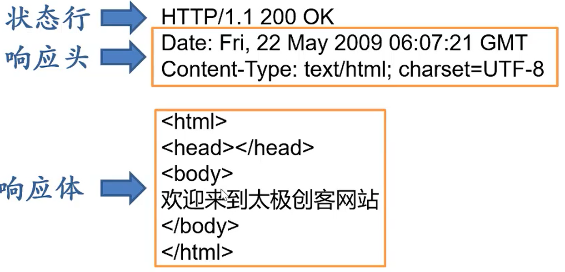
* 请求：

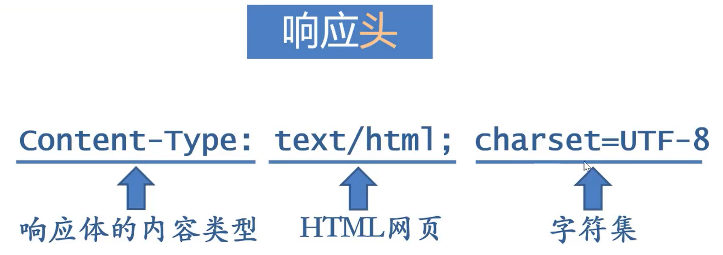




若想向网站服务器发送信息，应用POST而不是GET

* 响应：





**2、**esp8266以接入点模式建立的无线局域网，默认网关为192.168.4.1，可在

windows命令提示符中用ping 192.168.4.1进行请求

**3、esp8266三种工作模式：**

1）、接入点模式

1. #include <ESP8266WiFi.h> //引入esp8266WiFi库
2. **char** \*WiFi\_name = "Test\_chenhao"; //设置WiFi名为Test\_chenhao
3. **char** \*WiFi\_password = "12345678"; //设置WiFi密码为12345678，若不
4. **//** 设置密码""内不填任何信息即可
5. **void** setup()
6. {
7. WiFi.softAP(WiFi\_name, WiFi\_password); //启动无线接入点模式
8. Serial.begin(9600);             //配置串口波特率为9600
9. Serial.println(WiFi.softAPIP());//打印esp8266所建立WiFi局域网的
10. //默认网关IP地址
11. //程序下载完成后按下板子上RST复位按键即可在串口监视器中收到IP地址
12. }
14. **void** loop()
15. {
17. }

2）、无线终端模式

1. #include <ESP8266WiFi.h>            //引入ESP8266WiFi库
2. **char** \*WiFi\_name = "TP-LINK\_1CD4";   //设置连接WiFi名为TP-LINK\_1CD4
3. **char** \*WiFi\_password = "15150757563";//被连接WiFi密码为15150757563
4. **void** setup()
5. {
6. Serial.begin(9600);                  //配置串口比特率为9600
7. pinMode(LED\_BUILTIN,OUTPUT);         //设置LED\_BUILTIN引脚为输出模式
9. WiFi.begin(WiFi\_name, WiFi\_password);//开始连接WiFi
10. **while**(WiFi.status() != WL\_CONNECTED)
11. {
12. delay(100);
13. }
15. Serial.println(WiFi.localIP());      //打印ESP8266本地IP地址
16. }
17. **void** loop()
18. {
19. **if** (WiFi.status() == WL\_CONNECTED)   //若连接上WiFi，点亮蓝色LED
20. {
21. digitalWrite(LED\_BUILTIN,LOW);
22. }
23. }

（二）、ESP8266网络服务器

**1、建立基本网络服务器**

1. #include <ESP8266WiFi.h>
2. #include <ESP8266WebServer.h>
4. ESP8266WebServer esp8266\_server(80);//建立ESP8266WebServer对象，
5. //对象名称为esp8266\_server
6. //响应http请求端口号为80
7. //网络服务器标准http端口号为80
9. **char** \*WiFi\_name = "TP-LINK\_1CD4";   //被连接WiFi名称
10. **char** \*WiFi\_password = "15150757563";//被连接WiFi密码
11. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
12. **void** setup()
13. {
14. pinMode(LED\_BUILTIN,OUTPUT);//配置内置LED引脚为输出模式
15. Serial.begin(9600);         //配置串口波特率为9600
17. WiFi.begin(WiFi\_name, WiFi\_password);//开始连接WiFi
19. **while**(WiFi.status() != WL\_CONNECTED) //等待连接完成
20. {
21. delay(100);
22. }
23. digitalWrite(LED\_BUILTIN,LOW);
25. Serial.println(WiFi.localIP());      //串口打印esp8266本地IP地址
27. /\*------启动网络服务程序------\*/
28. esp8266\_server.begin();                //开始启动网络服务器
29. esp8266\_server.on("/", handle\_root);   //首页显示，调用
30. //handle\_root函数（/：为首页的意思）
31. esp8266\_server.onNotFound(handle\_not\_found);//其他页显示暂未
32. //开发，调用handle\_not\_found函数
33. }
34. /\*\*\*\*\*\*\*\*循环函数\*\*\*\*\*\*\*\*/
35. **void** loop()
36. {
37. esp8266\_server.handleClient();//处理http服务器访问
38. }
39. /\*\*\*\*\*\*\*\*处理首页访问函数\*\*\*\*\*\*\*\*/
40. **void** handle\_root()
41. {
42. //成功接受请求后，以纯文本方式发送字
43. //符 Hello From ESP8266 and ChenHao
44. esp8266\_server.send(200, "text/plain", "Hello From ESP8266 and ChenHao");
45. //200表示成功接受请求并已经完成整个处理过程
46. }
47. /\*\*\*\*\*\*\*\*其他页访问处理函数\*\*\*\*\*\*\*\*/
48. **void** handle\_not\_found()
49. {
50. //客服端网络服务请求错误，并以纯文本形式发送字符 404: Not Found
51. esp8266\_server.send(404, "text/plain", "404: Not Found");
52. //404表示客户端网页请求错误
53. }

**2、基本网络服务器开发板LED的控制**

1. #include <ESP8266WiFi.h>
2. #include <ESP8266WebServer.h>
4. ESP8266WebServer esp8266\_server(80);//建立ESP8266WebServer对象，对象名称为esp8266\_server
5. //响应http请求端口号为80
6. //网络服务器标准http端口号为80
8. **char** \*WiFi\_name = "TP-LINK\_1CD4";   //被连接WiFi名
9. **char** \*WiFi\_password = "15150757563";//被连接WiFi密码
10. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
11. **void** setup()
12. {
13. Serial.begin(9600);         //配置串口波特率为9600
14. pinMode(LED\_BUILTIN,OUTPUT);//设置内置LED端口为输出模式
16. WiFi.begin(WiFi\_name,WiFi\_password);//开始连接WiFi
17. **while**(WiFi.status() != WL\_CONNECTED)//等待连接完成
18. {
19. delay(100);
20. }
21. Serial.println(WiFi.localIP());     //串口打印esp8266本地IP地址
23. esp8266\_server.begin();                        //开始启动网页服务
24. //设置服务器根目录即"/"，HTTP\_GET表示其他终端HTTP协议请求，并调用handle\_root函数
25. esp8266\_server.on("/",HTTP\_GET,handle\_root);
26. //设置服务器目录LED"/LED"，HTTP\_POST表示其他终端HTTP协议的发送，并调用handle\_LED函数
27. esp8266\_server.on("/LED",HTTP\_POST,handle\_LED);
28. esp8266\_server.onNotFound(handle\_not\_found);   //若没有此网页，则调用handle\_not\_found函数
29. }
30. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
31. **void** loop()
32. {
33. esp8266\_server.handleClient();//时刻处理客户端访问
34. }
35. /\*\*\*\*\*\*\*\*处理网页根目录函数\*\*\*\*\*\*\*\*/
36. **void** handle\_root()
37. {
38. //以html格式发送一段代码
39. esp8266\_server.send(200,"text/html","<form action=\"/LED\" method=\"POST\"><input type=\"submit\" value=\"Toggle LED\"></form>");
40. //200表示成功接受请求并已经完成整个处理过程
41. }
42. /\*\*\*\*\*\*\*\*处理LED点亮与熄灭函数\*\*\*\*\*\*\*\*\*/
43. **void** handle\_LED()
44. {
45. digitalWrite(LED\_BUILTIN,!digitalRead(LED\_BUILTIN));//读取当前引脚电平并取反输出
46. esp8266\_server.sendHeader("Location","/");//跳转回页面根目录
47. esp8266\_server.send(303);                 //发送完成请求，303表示完成请求
48. }
49. /\*\*\*\*\*\*\*\*处理未发现该网页请求\*\*\*\*\*\*\*\*/
50. **void** handle\_not\_found()
51. {
52. esp8266\_server.send(404,"text/plain","404:Not Found");//以纯文本形式发送404:Not Found
53. //404表示客服端网页请求错误
54. }
55. <!--表头：建立/LED网页，方法为POST-->**>**
56. **<form** action="/LED" method="POST"**>**
57. <!--输入：类型为：submit，value为Toggle LED-->**>**
58. **<input** type="submit" value="Toggle LED"**>**
59. **</form>**  <!--表尾-->**>**

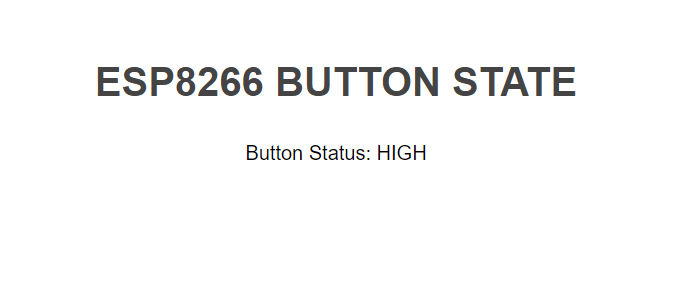
**3、基本网络服务器开发板引脚状态显示(手动刷新)**

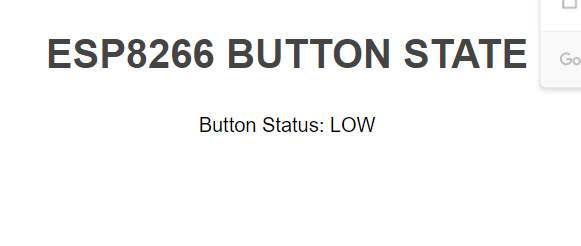
1. #include <ESP8266WiFi.h>
2. #include <ESP8266WebServer.h>
3. #define button\_pin D3  //宏定义button\_pin代替D3
5. ESP8266WebServer esp8266\_server(80);//建立ESP8266webServer对象
6. //响应HTTP请求端口为80
7. **char** \*WiFi\_name = "TP-LINK\_1CD4";   //被连接WiFi名
8. **char** \*WiFi\_password = "15150757563";//被连接WiFi密码
9. **bool** button\_status;                 //按键状态
10. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
11. **void** setup()
12. {
13. Serial.begin(9600);         //串口初始化
14. pinMode(LED\_BUILTIN,OUTPUT);//蓝色LED引脚初始化
15. pinMode(button\_pin, INPUT\_PULLUP);//设置button\_pin引脚为输入模式且打开上拉电阻
17. WiFi.begin(WiFi\_name,WiFi\_password);//连接WiFi
18. **while**(WiFi.status() != WL\_CONNECTED)//等待连接完成
19. {
20. delay(100);
21. }
23. digitalWrite(LED\_BUILTIN,LOW); //点亮LED,表示连接成功
24. Serial.println(WiFi.localIP());//串口输出esp8266本地IP地址
26. esp8266\_server.begin();                     //启动esp8266网络服务器
27. esp8266\_server.on("/",handle\_root);         //设置服务器根目录"/"，并调用handle\_root函数
28. esp8266\_server.onNotFound(handle\_not\_found);//若没有此网页则调用handle\_not\_found函数
29. }
30. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
31. **void** loop()
32. {
33. esp8266\_server.handleClient();          //时刻处理客户端访问
34. button\_status = digitalRead(button\_pin);//读取button\_pin引脚(D3)状态
35. }
36. /\*\*\*\*\*\*\*\*处理网页根目录函数\*\*\*\*\*\*\*\*/
37. **void** handle\_root()
38. {
39. String display\_button\_status;
40. **if**(button\_status == HIGH)//逻辑判断当前引脚状态
41. {
42. display\_button\_status = "Button Status : HIGH";
43. }
44. **else**
45. {
46. display\_button\_status = "Button Status : LOW";
47. }
48. //成功接受请求，则以纯文本形式发送字符变量display\_button\_status中内容
49. esp8266\_server.send(200,"text/plain",display\_button\_status);
50. //200表示成功接受请求并已经完成整个处理过程
51. }
52. /\*\*\*\*\*\*\*\*处理未发现网页函数\*\*\*\*\*\*\*\*/
53. **void** handle\_not\_found()
54. {
55. //客户端网页请求错误，则以纯文本形式发送字符404: Not Found
56. esp8266\_server.send(404,"text/plain","404: Not Found");
57. //404表示客户端网页请求错误
58. }

**4、基本网络服务器开发板引脚状态显示(自动刷新)**

1. #include <ESP8266WiFi.h>
2. #include <ESP8266WebServer.h>
3. #define button\_pin D3 //宏定义
5. ESP8266WebServer esp8266\_server(80);//建立ESP8266webServer对象
6. //响应HTTP请求端口为80
7. **char** \*WiFi\_name = "TP-LINK\_1CD4";   //被连接WiFi名
8. **char** \*WiFi\_password = "15150757563";//被连接WiFi密码
9. **bool** button\_status;                 //按键状态
10. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
11. **void** setup()
12. {
13. Serial.begin(9600);              //串口初始化
14. pinMode(LED\_BUILTIN,OUTPUT);     //蓝色LED引脚设置为输出模式
15. pinMode(button\_pin,INPUT\_PULLUP);//设置button\_pin引脚为输入模式且打开上拉电阻
17. WiFi.begin(WiFi\_name,WiFi\_password);//连接WiFi
18. **while**(WiFi.status() != WL\_CONNECTED)//等待连接完成
19. {
20. delay(100);
21. }
22. digitalWrite(LED\_BUILTIN,LOW); //点亮LED,表示连接成功
23. Serial.println(WiFi.localIP());//串口输出esp8266本地IP地址
25. esp8266\_server.begin();                     //启动esp8266网络服务器
26. esp8266\_server.on("/",handle\_root);         //设置服务器根目录"/"，并调用handle\_root函数
27. esp8266\_server.onNotFound(handle\_not\_found);//若没有此网页则调用handle\_not\_found函数
28. }
29. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
30. **void** loop()
31. {
32. esp8266\_server.handleClient();          //时刻处理客户端访问
33. button\_status = digitalRead(button\_pin);//读取button\_pin引脚(D3)状态
34. }
35. /\*\*\*\*\*\*\*\*处理网页根目录函数\*\*\*\*\*\*\*\*/
36. **void** handle\_root()
37. {
38. //成功接受请求，则以html形式发送函数send\_html返回字符串
39. esp8266\_server.send(200,"text/html",send\_html(button\_status));
40. //200表示成功接受请求并已经完成整个处理过程
41. }
42. /\*\*\*\*\*\*\*\*发送html代码函数\*\*\*\*\*\*\*\*/
43. String send\_html(**bool** button\_status\_temp)
44. {
45. String html\_code = "<!DOCTYPE html> <html>\n";
46. //content中可设置网页刷新率，当前1s刷新一次
47. html\_code +="<head><meta http-equiv='refresh' content='1'/>\n";
48. html\_code +="<title>ESP8266 Butoon State</title>\n";
49. html\_code +="<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}\n";
50. html\_code +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;} h3 {color: #444444;margin-bottom: 50px;}\n";
51. html\_code +="</style>\n";
52. html\_code +="</head>\n";
53. html\_code +="<body>\n";
54. html\_code +="<h1>ESP8266 BUTTON STATE</h1>\n";
55. //逻辑判断D3引脚当前电平状态
56. **if**(button\_status\_temp == HIGH)
57. {
58. html\_code +="<p>Button Status: HIGH</p>\n";
59. }
60. **else**
61. {
62. html\_code +="<p>Button Status: LOW</p>\n";
63. }
65. **return** html\_code; //返回html代码
66. }
67. /\*\*\*\*\*\*\*\*处理未发现网页函数\*\*\*\*\*\*\*\*/
68. **void** handle\_not\_found()
69. {
70. //客户端网页请求错误，则以纯文本形式发送字符404: Not Found
71. esp8266\_server.send(404,"text/plain","404: Not Found");
72. //404表示客户端网页请求错误
73. }

运行效果：

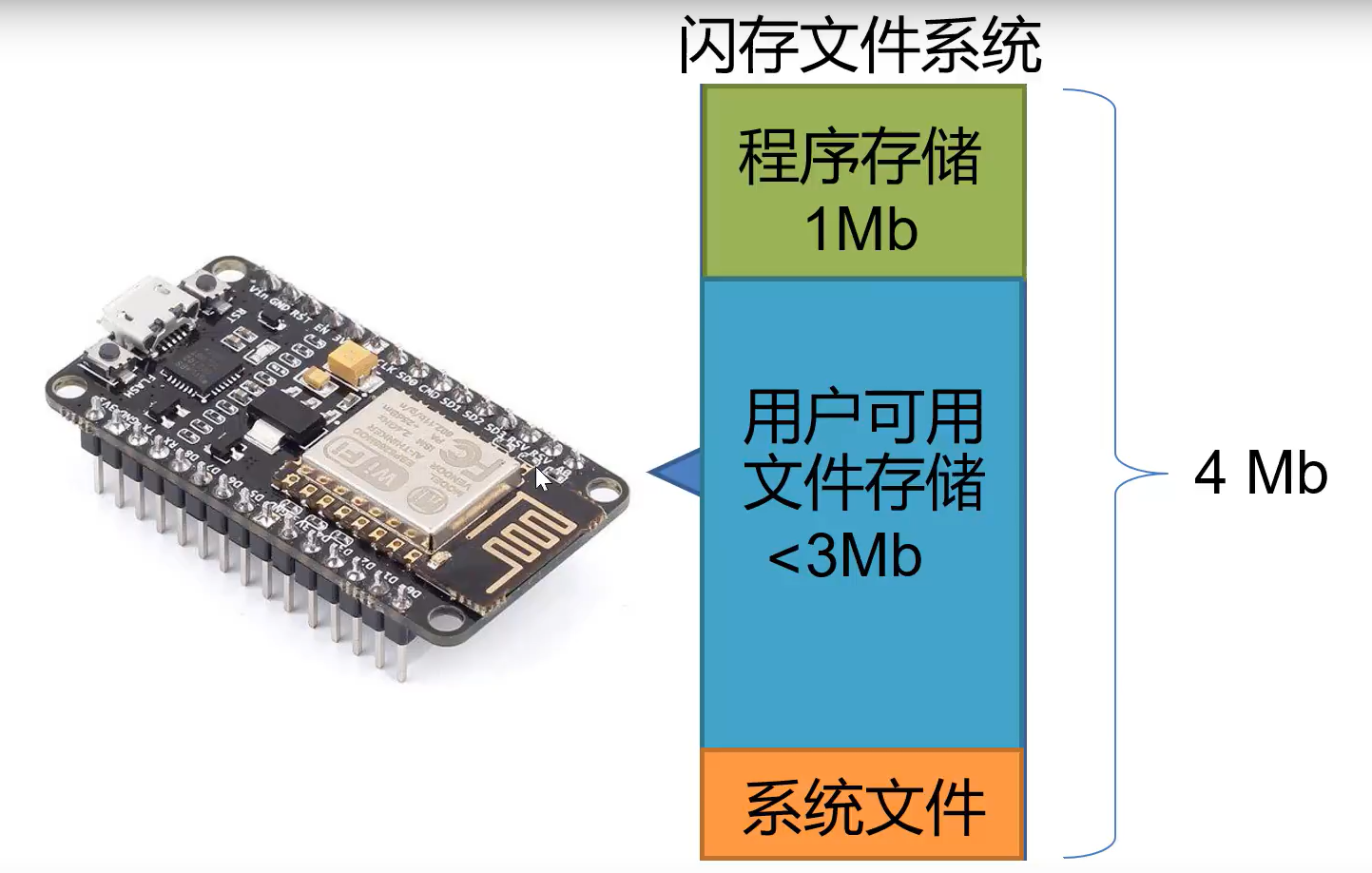




（三）、闪存文件系统

**1、闪存文件系统**

对于esp8266闪存文件系统共有4Mb，其中1Mb作为程序存储（编译后程序），3Mb作为文件存储（其中有一部分作为esp8266存储系统文件，即用户可用<3Mb）

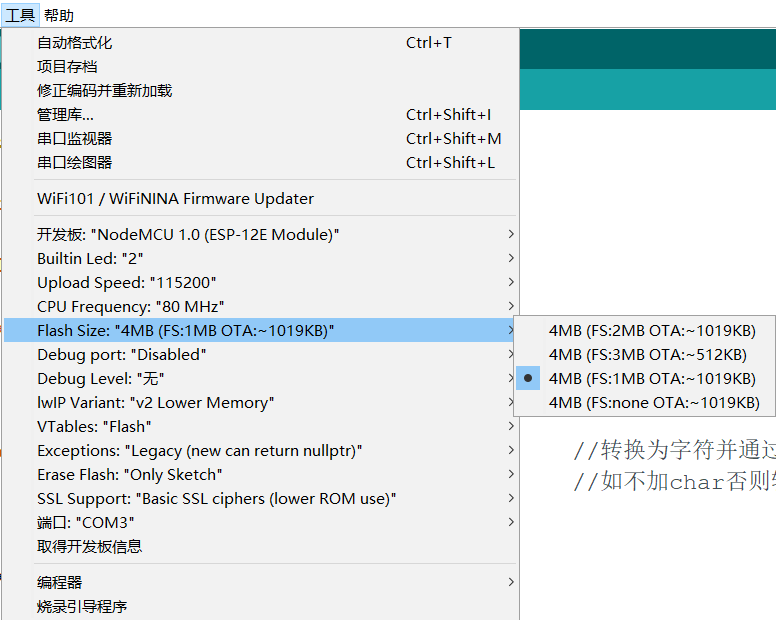


**2、闪存文件系统基本操作**

1）、闪存文件系统写入

1. #include <FS.h> //引入闪存文件操作依赖的库
3. String file\_name = "/Test/note\_1.txt"; //定义文件名字符串
4. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
5. **void** setup()
6. {
7. Serial.begin(9600);//以波特率为9600启动串口
9. SPIFFS.format();   //格式化闪存文件系统
10. //Serial Peripheral Interface Flash File System（SPIFFS）
11. //串行外设接口Flash文件系统即闪存文件系统
13. **if**(SPIFFS.begin()) //启动闪存文件系统
14. {
15. Serial.println("SPIFFS stated");
16. }
17. **else**
18. {
19. Serial.println("SPIFFS fail to start");
20. }
22. File date\_file = SPIFFS.open(file\_name,"w");//建立文件对象date\_file,闪存文件系统以写方式打开文件。
23. //如果文件系统没有/taichi-maker/notes.txt文件，此操作将会在文件系统中建立该文件；
24. //如果文件系统有该文件，则程序将会重新建立该文件，即原有文件信息将会被覆盖。
26. date\_file.println("Hello World 2022-1-24");//文件对象写入Hello World 2022-1-24
27. date\_file.close();                         //关闭文件对象，结束文件操作后应执行此操作
28. }
29. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
30. **void** loop()
31. {
33. }

注意：文件系统的写入需在Arduino IDE中额外设置写入文件大小，1MB、2MB、3MB根据实际写入文件大小进行选择



2）、闪存文件系统读取信息

1. #include <FS.h> //引入闪存文件操作依赖的库
3. String file\_name = "/Test/note\_1.txt"; //定义文件名字符串
4. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
5. **void** setup()
6. {
7. Serial.begin(9600);//配置串口波特率为9600
8. SPIFFS.begin();    //启动闪存文件系统
10. **if**(SPIFFS.exists(file\_name)) //判断该文件是否存在
11. {
12. File date\_file = SPIFFS.open(file\_name,"r");//定义文件对象，并以读方式打开file\_name
13. **for**(**int** i=0; i < date\_file.size(); i++)     //循环读取文件中信息
14. {
15. Serial.print((**char**)date\_file.read()); //转换为字符并通过串口输出
16. //如不加char否则输出字符对应ASCII码
17. }
18. date\_file.close(); //关闭操作文件
19. }
20. }
21. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
22. **void** loop()
23. {
25. }

3）、闪存文件系统追加信息

1. #include <FS.h> //导入文件操作系统依赖的库
3. String file\_name = "/Test/note\_1.txt"; //被操作文件名字符串
4. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
5. **void** setup()
6. {
7. SPIFFS.begin();    //启动文件操作系统
9. **if**(SPIFFS.exists(file\_name)) //逻辑判断是否存在被操作文件
10. {
11. File date\_file = SPIFFS.open(file\_name,"a");        //创建文件操作对象，并以追加(append)方式打开文件
12. date\_file.println("This is a appended information");//追加(println为换行)
13. date\_file.close();                                  //关闭操作文件对象
14. }
15. }
16. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
17. **void** loop()
18. {
20. }

4）、闪存文件系统读取文件夹目录

1. #include <FS.h> //引入闪存操作系统依赖的库
3. String folder\_name = "/Test"; //定义文件夹名的字符串
5. **void** setup()
6. {
7. Serial.begin(9600); //初始化串口并配置波特率为9600
8. SPIFFS.begin();     //初始化打开闪存文件操作系统
10. Dir dir = SPIFFS.openDir(folder\_name);//定义文件目录对象，并打开文件目录folder\_name
11. **while**(dir.next()) //检查目录中是否由下一个文件
12. {
13. Serial.println(dir.fileName()); //通过串口输出文件名
14. }
15. }
17. **void** loop()
18. {
20. }

5）、闪存文件操作系统删除文件

1. #include <FS.h> //引入闪存文件系统操作依赖的函数
3. String file\_name = "/Test/note\_1.txt";//定义文件名字符串
4. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
5. **void** setup()
6. {
7. Serial.begin(9600);//启动串口并配置波特率为9600
8. SPIFFS.begin();    //启动闪存文件操作系统
10. **if**(SPIFFS.remove(file\_name)) //删除文件名为file\_name的文件
11. {                            //并逻辑判断是否成功
12. Serial.println("remove sucessfully");//成功删除则输出remove sucessfully
13. }
14. **else**
15. {
16. Serial.println("remove fail");       //失败则输出remove fail
17. }
18. //注意：按下RST键会输出失败，这是由于程序已经上传后已经执行过一次，再按下RST程序会执行第二次
19. }
20. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
21. **void** loop()
22. {
24. }

**3、闪存文件系统建立功能丰富网络服务器**

1. #include <ESP8266WiFi.h>      //引入连接WiFi库
2. #include <ESP8266WebServer.h> //引入网络服务器库
3. #include <FS.h>               //引入闪存文件操作系统库
5. ESP8266WebServer esp8266\_server(80); //建立网络ESP8266WebServer对象，端口号默认为80
6. **char** \*WiFi\_name = "TP-LINK\_1CD4";    //被连接WiFi名称
7. **char** \*WiFi\_password = "15150757563"; //被连接WiFi密码
8. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
9. **void** setup()
10. {
11. //启动串口并配置波特率为9600
12. Serial.begin(9600);
13. //启动闪存文件操作系统
14. SPIFFS.begin();
15. //连接WiFi
16. WiFi.begin(WiFi\_name,WiFi\_password);
17. **while**(WiFi.status() != WL\_CONNECTED)//等待连接完成
18. {
19. delay(100);
20. }
21. Serial.println(WiFi.localIP());     //串口输出esp8266本地IP地址
22. //启动esp8266网络服务
23. esp8266\_server.begin();
24. esp8266\_server.onNotFound(handle\_user\_request);//处理用户请求
25. }
26. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
27. **void** loop()
28. {
29. esp8266\_server.handleClient();//时刻处理用户请求
30. }
31. /\*\*\*\*\*\*\*\*处理用户HTTP请求函数\*\*\*\*\*\*\*\*/
32. **void** handle\_user\_request()
33. {
34. //获取用户请求网址
35. String web\_address = esp8266\_server.uri();
36. //调用handle\_file\_read对文件进行处理
37. **bool** file\_read\_ok = handle\_file\_read(web\_address);
38. //逻辑判断是否由该文件，若无网页展示 404: Not Found
39. **if**(!file\_read\_ok)
40. {
41. esp8266\_server.send(404,"text/plain","404: Not Found");
42. }
43. }
44. /\*\*\*\*\*\*\*\*处理文件函数\*\*\*\*\*\*\*\*/
45. **bool** handle\_file\_read(String file\_path)
46. {
47. //若网址为/(即首页)，则修改网址为/index.html
48. **if**(file\_path.endsWith("/")) //endsWith函数即判断改字符串是否以 / 结尾
49. {
50. file\_path = "/index.html";
51. }
52. //调用get\_file\_type函数获取文件格式
53. String file\_type = get\_file\_type(file\_path);
54. //将文件发送给网页
55. **if**(SPIFFS.exists(file\_path)) //逻辑判断是否存在该文件路径
56. {
57. File file = SPIFFS.open(file\_path,"r");   //以读方式打开改路径文件
58. esp8266\_server.streamFile(file,file\_type);//发送文件以及文件类型给网页
59. file.close();                             //关闭文件
60. **return** **true**;                              //返回true
61. }
62. **return** **false**;                               //返回false
63. }
64. /\*\*\*\*\*\*\*\*获取文件类型函数\*\*\*\*\*\*\*\*/
65. String get\_file\_type(String file\_name)
66. {
67. **if** (file\_name.endsWith(".html")) **return** "text/html";
68. **else** **if**(file\_name.endsWith(".css")) **return** "text/css";
69. **else** **if**(file\_name.endsWith(".js")) **return** "application/javascript";
70. **else** **if**(file\_name.endsWith(".jpg")) **return** "image/jpeg";
71. **else** **if**(file\_name.endsWith(".png")) **return** "image/png";
72. **else** **if**(file\_name.endsWith(".gif")) **return** "image/gif";
73. **else** **if**(file\_name.endsWith(".ico")) **return** "image/x-icon";
74. **else** **return** "text/plain";
75. }

效果如下图：



注意：需将所保存到闪存文件中的内容放置于代码文件平行目录下并命名为data



然后在Arduino IDE 中选择工具中的ESP8266 Sketch Data Upload命令，将data文件中内容烧录进esp8266闪存文件系统中。

**4、网页控制ESP8266开发板引脚**

1. #include <ESP8266WiFi.h>      //引入连接WiFi库
2. #include <ESP8266WebServer.h> //引入网络服务器库
3. #include <FS.h>               //引入闪存文件操作系统库
5. ESP8266WebServer esp8266\_server(80); //建立网络ESP8266WebServer对象，端口号默认为80
6. **char** \*WiFi\_name = "TP-LINK\_1CD4";    //被连接WiFi名称
7. **char** \*WiFi\_password = "15150757563"; //被连接WiFi密码
8. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
9. **void** setup()
10. {
11. //启动串口并配置波特率为9600
12. Serial.begin(9600);
13. //启动闪存文件操作系统
14. SPIFFS.begin();
15. //设置引脚LED\_BUILTIN为输出模式
16. pinMode(LED\_BUILTIN,OUTPUT);
17. //连接WiFi
18. WiFi.begin(WiFi\_name,WiFi\_password);
19. **while**(WiFi.status() != WL\_CONNECTED)//等待连接完成
20. {
21. delay(100);
22. }
23. Serial.println(WiFi.localIP());     //串口输出esp8266本地IP地址
24. //启动esp8266网络服务
25. esp8266\_server.begin();
26. esp8266\_server.on("/LED-Control",handle\_LED\_control);//处理网页/LED请求
27. esp8266\_server.onNotFound(handle\_user\_request);      //处理其他请求
28. }
29. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
30. **void** loop()
31. {
32. esp8266\_server.handleClient();//时刻处理用户请求
33. }
34. /\*\*\*\*\*\*\*\*处理LED控制函数\*\*\*\*\*\*\*\*/
35. **void** handle\_LED\_control()
36. {
37. digitalWrite(LED\_BUILTIN,!digitalRead(LED\_BUILTIN));//控制LED\_BUILTIN引脚电平取反
38. esp8266\_server.sendHeader("Location","/LED.html");  //跳转位置，位置为/LED.html(即刷新当前页)
39. esp8266\_server.send(303);                           //发送完成跳转
40. }
41. /\*\*\*\*\*\*\*\*处理用户HTTP请求函数\*\*\*\*\*\*\*\*/
42. **void** handle\_user\_request()
43. {
44. //获取用户请求网址
45. String web\_address = esp8266\_server.uri();
46. //调用handle\_file\_read对文件进行处理
47. **bool** file\_read\_ok = handle\_file\_read(web\_address);
48. //逻辑判断是否由该文件，若无网页展示 404: Not Found
49. **if**(!file\_read\_ok)
50. {
51. esp8266\_server.send(404,"text/plain","404: Not Found");
52. }
53. }
54. /\*\*\*\*\*\*\*\*处理文件函数\*\*\*\*\*\*\*\*/
55. **bool** handle\_file\_read(String file\_path)
56. {
57. //若网址为/(即首页)，则修改网址为/index.html
58. **if**(file\_path.endsWith("/")) //endsWith函数即判断改字符串是否以 / 结尾
59. {
60. file\_path = "/index.html";
61. }
62. //调用get\_file\_type函数获取文件格式
63. String file\_type = get\_file\_type(file\_path);
64. //将文件发送给网页
65. **if**(SPIFFS.exists(file\_path)) //逻辑判断是否存在该文件路径
66. {
67. File file = SPIFFS.open(file\_path,"r");   //以读方式打开改路径文件
68. esp8266\_server.streamFile(file,file\_type);//发送文件以及文件类型给网页
69. file.close();                             //关闭文件
70. **return** **true**;                              //返回true
71. }
72. **return** **false**;                               //返回false
73. }
74. /\*\*\*\*\*\*\*\*获取文件类型函数\*\*\*\*\*\*\*\*/
75. String get\_file\_type(String file\_name)
76. {
77. **if** (file\_name.endsWith(".html")) **return** "text/html";
78. **else** **if**(file\_name.endsWith(".css")) **return** "text/css";
79. **else** **if**(file\_name.endsWith(".js")) **return** "application/javascript";
80. **else** **if**(file\_name.endsWith(".jpg")) **return** "image/jpeg";
81. **else** **if**(file\_name.endsWith(".png")) **return** "image/png";
82. **else** **if**(file\_name.endsWith(".gif")) **return** "image/gif";
83. **else** **if**(file\_name.endsWith(".ico")) **return** "image/x-icon";
84. **else** **return** "text/plain";
85. }

（四）、网络客户端

**1、ESP8266HTTPClient库实现网络通讯（不建议采用）**

1. #include <ESP8266WiFi.h>       //引入ESP8266库
2. #include <ESP8266HTTPClient.h> //引入esp网络客户端库
3. #define URL "http://www.example.com" //宏定义
5. **char** \*WiFi\_name = "TP-LINK\_1CD4";   //被连接wifi名
6. **char** \*WiFi\_password = "15150757563";//被连接wifi密码
7. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
8. **void** setup()
9. {
10. Serial.begin(9600); //以波特率为9600启动串口
11. WiFi.mode(WIFI\_STA);//设置esp8266工作模式为无线终端模式
13. WiFi.begin(WiFi\_name,WiFi\_password);//连接wifi
14. **while**(WiFi.status() != WL\_CONNECTED)//等待wifi连接完成
15. {
16. delay(100);
17. }
19. httpClientRequest();
20. }
21. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
22. **void** loop()
23. {
25. }
26. /\*\*\*\*\*\*\*\*处理客户端请求函数\*\*\*\*\*\*\*\*/
27. **void** httpClientRequest()
28. {
29. HTTPClient http\_client; //定义HTTPclient对象
31. http\_client.begin(URL); //esp8266发送请求目标网址
33. **int** http\_code = http\_client.GET(); //esp8266向网络服务器发送GET请求，
35. **if**(http\_code == HTTP\_CODE\_OK)
36. {
37. String response\_payload = http\_client.getString(); //获取服务器响应体的信息
38. Serial.println(response\_payload);                  //通过串口打印响应体信息
39. }
40. **else**
41. {
42. Serial.println(http\_code);
43. }
44. }

**2、WiFiClient库实现网络通讯（建议采用）**

1. #include <ESP8266WiFi.h>
2. String web = "www.example.com";      //定义发送请求信息的网站的字符指针
3. String wifi\_name = "TP-LINK\_1CD4";   //定义被连接wifi名字符指针
4. String wifi\_password = "15150757563";//定义被连接wifi密码字符指针
6. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
7. **void** setup()
8. {
9. Serial.begin(9600); //以波特率9600启动串口
11. WiFi.mode(WIFI\_STA);//设置esp8266为无线终端模式
13. WiFi.begin(wifi\_name,wifi\_password);//连接WiFi
14. **while**(WiFi.status() != WL\_CONNECTED)//等待连接完成
15. {
16. delay(100);
17. }
19. sendClientRequest(); //向目标网站发送客户端请求
20. }
21. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
22. **void** loop()
23. {
25. }
26. /\*\*\*\*\*\*\*\*发送客户端请求函数\*\*\*\*\*\*\*\*/
27. **void** sendClientRequest()
28. {
29. WiFiClient http\_client;//建立客户端操作对象
31. String http\_request = String("GET /") + " HTTP/1.1\r\n" + //建立http请求字符串
32. "Host:" + web + "\r\n" +
33. "Connection: close\r\n" +
34. "\r\n";
36. **if**(http\_client.connect(web,80))  //连接目标网站，host为被连接网站，80为默认http端口
37. {
38. http\_client.println(http\_request);//向目标网站发送http请求
40. **while**(http\_client.connected() || http\_client.available())
41. {
42. **if**(http\_client.available())
43. {
44. String payload = http\_client.readStringUntil('\n'); //读取服务器发来的信息直到遇到\n停止
45. Serial.println(payload);                            //串口打印接收到的信息
46. }
47. }
49. http\_client.stop();
50. }
51. }

（五）、stream型数据

**1、串口监视器演示stream数据发与收**

1. **void** setup()
2. {
3. Serial.begin(9600);
4. }
6. **void** loop()
7. {
8. **if**(Serial.available())
9. {
10. String data = Serial.readString();
11. Serial.print(data);
12. }
13. }

**2、WiFiClient库演示stream数据**

1. #include <ESP8266WiFi.h> //引入esp8266wifi库
3. String web = "www.example.com";      //连接的网站
4. String wifi\_name = "TP-LINK\_1CD4";   //被连接wifi名
5. String wifi\_password = "15150757563";//被连接wifi密码
6. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
7. **void** setup()
8. {
9. Serial.begin(9600); //启动串口并配置波特率为9600
11. WiFi.mode(WIFI\_STA);//设置wifi工作模式为无线终端模式
12. WiFi.begin(wifi\_name,wifi\_password);//连接wifi
13. **while**(WiFi.status() != WL\_CONNECTED)//等待连接完成
14. {
15. delay(100);
16. }
18. sendClientRequest(); //发送客户端请求
19. }
20. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
21. **void** loop()
22. {
24. }
25. /\*\*\*\*\*\*\*\*向目标网站发送客户端请求函数\*\*\*\*\*\*\*\*/
26. **void** sendClientRequest()
27. {
28. WiFiClient http\_client; //定义WiFiClient对象
30. String http\_request = String("GET /") + " HTTP/1.1\r\n" + //建立http请求字符串
31. "Host:" + web + "\r\n" +
32. "Connection: close\r\n" +
33. "\r\n";
35. **if**(http\_client.connect(web,80))   //连接目标网站，若成功则返回1否则返回0
36. {
37. http\_client.print(http\_request);//向目标网站发送http请求
38. //循环接受服务器发来的数据信息
39. **while**(http\_client.connected() || http\_client.available())
40. {
41. **if**(http\_client.available())
42. {
43. String payload = http\_client.readStringUntil('\n'); //接受目标网站发来的数据信息，知道遇到‘\n’
44. Serial.println(payload);                            //串口输出接收到的数据
45. }
46. }
47. }
49. http\_client.stop(); //停止esp8266连接服务器，即断开esp8266与服务器的连接
50. }

**3、FS库（闪存文件操作系统库）演示stream数据**

**略**

（六）、JSON

**1、单一对象JSON解析**

1. #include <ESP8266WiFi.h> //引入esp8266wifi库
3. String web = "www.example.com";      //连接的网站
4. String wifi\_name = "TP-LINK\_1CD4";   //被连接wifi名
5. String wifi\_password = "15150757563";//被连接wifi密码
6. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
7. **void** setup()
8. {
9. Serial.begin(9600); //启动串口并配置波特率为9600
11. WiFi.mode(WIFI\_STA);//设置wifi工作模式为无线终端模式
12. WiFi.begin(wifi\_name,wifi\_password);//连接wifi
13. **while**(WiFi.status() != WL\_CONNECTED)//等待连接完成
14. {
15. delay(100);
16. }
18. sendClientRequest(); //发送客户端请求
19. }
20. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
21. **void** loop()
22. {
24. }
25. /\*\*\*\*\*\*\*\*向目标网站发送客户端请求函数\*\*\*\*\*\*\*\*/
26. **void** sendClientRequest()
27. {
28. WiFiClient http\_client; //定义WiFiClient对象
30. String http\_request = String("GET /") + " HTTP/1.1\r\n" + //建立http请求字符串
31. "Host:" + web + "\r\n" +
32. "Connection: close\r\n" +
33. "\r\n";
35. **if**(http\_client.connect(web,80))   //连接目标网站，若成功则返回1否则返回0
36. {
37. http\_client.print(http\_request);//向目标网站发送http请求
38. //循环接受服务器发来的数据信息
39. **while**(http\_client.connected() || http\_client.available())
40. {
41. **if**(http\_client.available())
42. {
43. String payload = http\_client.readStringUntil('\n'); //接受目标网站发来的数据信息，知道遇到‘\n’
44. Serial.println(payload);                            //串口输出接收到的数据
45. }
46. }
47. }
49. http\_client.stop(); //停止esp8266连接服务器，即断开esp8266与服务器的连接
50. }

**2、JSON数组解析**

1. #include <ArduinoJson.h> //引入ArduinoJson解析库
3. **void** setup()
4. {
5. Serial.begin(9600);
7. //带解析json字符
8. String json = "[{\"name\":\"taichi-maker\"},{\"website\":\"www.taichi-maker.com\"}]";
9. //定义带解析json字符容量 和 定义名为doc的DynamicJsonDocument对象
10. **const** **size\_t** capacity = JSON\_ARRAY\_SIZE(2) + 2\*JSON\_OBJECT\_SIZE(1) + 60;
11. DynamicJsonDocument doc(capacity);
12. //利用deserialize函数（反序列化函数）解析json并将结果保存至doc对象中
13. deserializeJson(doc,json);
14. //将解析结果转换为字符串分别赋值给name\_str 和 website\_str
15. String name\_str = doc[0]["name"].as<String>();
16. String website\_str = doc[1]["website"].as<String>();
18. Serial.println(name\_str);
19. Serial.println(website\_str);
20. }
22. **void** loop()
23. {
25. }

**3、JSON解析代码生成**

1. #include <ArduinoJson.h>
2. **void** setup()
3. {
4. Serial.begin(9600);
5. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
6. //网站自动解析生成的代码（https://arduinojson.org/v5/assistant/）
7. **const** **size\_t** capacity = JSON\_ARRAY\_SIZE(1) + JSON\_OBJECT\_SIZE(1) + JSON\_OBJECT\_SIZE(2) + 2\*JSON\_OBJECT\_SIZE(3) + 130;
8. DynamicJsonBuffer jsonBuffer(capacity);
9. **const** **char**\* json = "{\"results\":[{\"location\":{\"name\":\"Beijing\",\"country\":\"CN\"},\"now\":{\"text\":\"Clear\",\"code\":\"1\",\"temperature\":\"3\"},\"last\_update\":\"2020-03-01T20:10:00+08:00\"}]}";
10. JsonObject& root = jsonBuffer.parseObject(json);
11. JsonObject& results\_0 = root["results"][0];
12. **const** **char**\* results\_0\_location\_name = results\_0["location"]["name"]; // "Beijing"
13. **const** **char**\* results\_0\_location\_country = results\_0["location"]["country"]; // "CN"
14. JsonObject& results\_0\_now = results\_0["now"];
15. **const** **char**\* results\_0\_now\_text = results\_0\_now["text"]; // "Clear"
16. **const** **char**\* results\_0\_now\_code = results\_0\_now["code"]; // "1"
17. **const** **char**\* results\_0\_now\_temperature = results\_0\_now["temperature"]; // "3"
18. **const** **char**\* results\_0\_last\_update = results\_0["last\_update"]; // "2020-03-01T20:10:00+08:00"
19. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
20. //输出json解析结果
21. Serial.println(results\_0\_location\_name);
22. Serial.println(results\_0\_location\_country);
23. Serial.println(results\_0\_now\_text);
24. Serial.println(results\_0\_now\_code);
25. Serial.println(results\_0\_now\_temperature);
26. Serial.println(results\_0\_last\_update);
27. }
29. **void** loop()
30. {
31. }

**4、读取ESP8266存储的JSON并解析**

1. #include <ArduinoJson.h> //引入ArduinoJson解析库
2. #include <ESP8266WiFi.h> //引入ESP8266WiFi库
3. #include <FS.h>          //引入闪存文件操作库
5. **void** setup()
6. {
7. //从闪存文件系统中config.json的文件读出wifi的ssid与password
8. SPIFFS.begin();
9. File config\_file = SPIFFS.open("config.json","r");
10. String json = config\_file.readString();
11. config\_file.close();
12. //解析读出的json
13. **const** **size\_t** capacity = JSON\_OBJECT\_SIZE(1) + JSON\_OBJECT\_SIZE(2) + 50;
14. DynamicJsonBuffer jsonBuffer(capacity);
15. JsonObject& root = jsonBuffer.parseObject(json);
16. **const** **char**\* wifi\_ssid = root["wifi"]["ssid"];         // "TP-LINK\_1CD4"
17. **const** **char**\* wifi\_password = root["wifi"]["password"]; // "15150757563"
18. //连接WiFi
19. WiFi.begin(wifi\_ssid,wifi\_password);
20. **while**(WiFi.status() != WL\_CONNECTED)
21. {
22. delay(10);
23. }
24. //设置LED引脚为输出模式
25. pinMode(LED\_BUILTIN,OUTPUT);
26. }
28. **void** loop()
29. {
30. //若esp8266已连接wifi则打开LED
31. **if**(WiFi.status() == WL\_CONNECTED)
32. {
33. digitalWrite(LED\_BUILTIN,LOW);
34. }
35. }

（七）、HTTP API应用、

**1、获取并解析心知天气实时天气信息（天气、温度）**

1. #include <ESP8266WiFi.h> //引入ESP8266WiFi库
2. #include <ArduinoJson.h> //引入ArduinoJson库
3. //wifi名与wifi密码
4. String wifi\_ssid = "TP-LINK\_1CD4";
5. String wifi\_password = "15150757563";
6. //心知天气api接口
7. String host = "api.seniverse.com";
8. **int** port = 80;
9. //心知天气获取天气必备
10. String user\_key = "S-9a3-Z4sX\_eWvo7H"; //私钥
11. String city = "Suqian"; //城市：宿迁
12. String language = "en"; //语言：英语
13. String unit = "c";     //温度单位：摄氏度
14. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
15. **void** setup()
16. {
17. Serial.begin(9600);         //初始化串口
18. pinMode(LED\_BUILTIN,OUTPUT);//设置LED引脚为输出模式
19. connect\_wifi();             //连接wifi
20. }
21. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
22. **void** loop()
23. {
24. wifi\_ok\_LED();//检测esp8266是否连接wifi，若连接点亮LED
25. get\_weather();//获取当前天气状况
26. delay(3000);  //延时3s，免费版心知天气，访问频率：20次/分
27. }
28. /\*\*\*\*\*\*\*\*连接wifi函数\*\*\*\*\*\*\*\*/
29. **void** connect\_wifi()
30. {
31. WiFi.begin(wifi\_ssid,wifi\_password);
32. **while**(WiFi.status() != WL\_CONNECTED)
33. {
34. delay(100);
35. }
36. }
37. /\*\*\*\*\*\*\*\*wifi连接，LED点亮函数\*\*\*\*\*\*\*\*/
38. **void** wifi\_ok\_LED()
39. {
40. **if**(WiFi.status() == WL\_CONNECTED)
41. {
42. digitalWrite(LED\_BUILTIN,LOW);
43. }
44. }
45. /\*\*\*\*\*\*\*\*获取天气函数\*\*\*\*\*\*\*\*/
46. **void** get\_weather()
47. {
48. String temp = "/v3/weather/now.json?key=" + user\_key + "&location=" + city + "&language=" +language + "&unit=" + unit;
49. //http请求
50. String http\_request = "GET " + temp + " HTTP/1.1\r\n" +
51. "Host: " + host + "\r\n" +
52. "Connection: close\r\n" +
53. "\r\n";
54. WiFiClient http\_client; //建立WiFiClient对象
55. **if**(http\_client.connect(host,port)) //连接心知天气api，成功连接返回1
56. {
57. http\_client.print(http\_request); //发送http请求
58. **if**(http\_client.find("\r\n\r\n")) //寻找"\r\n\r\n"
59. {
60. String json = http\_client.readString(); //读取http响应中的json数据
61. parse\_json(json);                       //调用json解析函数，并串口输出解析结果
62. }
63. //    while(http\_client.connected() || http\_client.available())
64. //    {
65. //      if(http\_client.available())
66. //      {
67. //        String data = http\_client.readStringUntil('\n');
68. //        Serial.println(data);
69. //      }
70. //    }
71. }
72. http\_client.stop(); //断开与服务器连接
73. }
74. /\*\*\*\*\*\*\*\*json解析函数\*\*\*\*\*\*\*\*/
75. **void** parse\_json(String json)
76. {
77. /\*以下代码为ArduinoJson库作者的在线网站（https://arduinojson.org/v5/assistant/）自动生成解析json代码，注意：这里实用json库为V5版本\*/
78. **const** **size\_t** capacity = JSON\_ARRAY\_SIZE(1) + JSON\_OBJECT\_SIZE(1) + 2\*JSON\_OBJECT\_SIZE(3) + JSON\_OBJECT\_SIZE(6) + 240;
79. DynamicJsonBuffer jsonBuffer(capacity);
80. JsonObject& root = jsonBuffer.parseObject(json);
81. JsonObject& results\_0 = root["results"][0];
82. JsonObject& results\_0\_location = results\_0["location"];
83. **const** **char**\* results\_0\_location\_id = results\_0\_location["id"]; // "WWH1780CEFBR"
84. **const** **char**\* results\_0\_location\_name = results\_0\_location["name"]; // "Suqian"
85. **const** **char**\* results\_0\_location\_country = results\_0\_location["country"]; // "CN"
86. **const** **char**\* results\_0\_location\_path = results\_0\_location["path"]; // "Suqian,Suqian,Jiangsu,China"
87. **const** **char**\* results\_0\_location\_timezone = results\_0\_location["timezone"]; // "Asia/Shanghai"
88. **const** **char**\* results\_0\_location\_timezone\_offset = results\_0\_location["timezone\_offset"]; // "+08:00"
89. JsonObject& results\_0\_now = results\_0["now"];
90. **const** **char**\* results\_0\_now\_text = results\_0\_now["text"]; // "Haze"
91. **const** **char**\* results\_0\_now\_code = results\_0\_now["code"]; // "31"
92. **const** **char**\* results\_0\_now\_temperature = results\_0\_now["temperature"]; // "4"
93. **const** **char**\* results\_0\_last\_update = results\_0["last\_update"]; // "2022-01-27T16:04:05+08:00"
94. /\*以上代码为ArduinoJson库作者的在线网站（https://arduinojson.org/v5/assistant/）自动生成解析json代码，注意：这里实用json库为V5版本\*/
96. //串口输出解析结果
97. Serial.println(results\_0\_location\_name);   //城市：Suqian
98. Serial.println(results\_0\_location\_country);//国家：CN
99. Serial.println(results\_0\_location\_path);   //路径：Suqian,Suqian,Jiangsu,China
100. Serial.println(results\_0\_now\_text);        //天气：Cloudy
101. Serial.println(results\_0\_now\_code);        //对应天气码：4
102. Serial.println(results\_0\_now\_temperature); //摄氏温度：4
103. Serial.println(results\_0\_last\_update);     //上一次更新时间：2022-01-27T16:43:16+08:00
104. }

获取实时天气信息为JSON格式，内容为：

1. {
2. "results":
3. [
4. {
5. "location":
6. {
7. "id":"WWH1780CEFBR",
8. "name":"Suqian",
9. "country":"CN",
10. "path":"Suqian,Suqian,Jiangsu,China",
11. "timezone":"Asia/Shanghai",
12. "timezone\_offset":"+08:00"
13. },
14. "now":
15. {
16. "text":"Cloudy",
17. "code":"31",
18. "temperature":"4"
19. },
20. "last\_update":"2022-01-27T16:04:05+08:00"
21. }
22. ]
23. }

**2、 获取心知天气未来三日预报信息（白天天气、夜晚天气、最高温度、最低温度、降水量、**

**降水概率、风向、风向角度、风速、风力等级、相对湿度）**

1. #include <ESP8266WiFi.h> //引入ESP8266WiFi库
2. #include <ArduinoJson.h> //引入ArduinoJso库
3. //wifi名称与密码
4. String wifi\_ssid = "TP-LINK\_1CD4";
5. String wifi\_password = "15150757563";
6. //心知天气api接口网址与服务器端口
7. String host = "api.seniverse.com";
8. **int** port = 80;
9. //心知天气获取未来三天天气预报必备信息
10. String user\_key = "S-9a3-Z4sX\_eWvo7H";
11. String location = "Suqian";
12. String language = "en";
13. String unit = "c";
14. String start = "0";
15. String days = "3";
16. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
17. **void** setup()
18. {
19. Serial.begin(9600);
20. pinMode(LED\_BUILTIN,OUTPUT);
21. connect\_wifi();
22. }
23. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
24. **void** loop()
25. {
26. wifi\_connect\_ok();
27. get\_weather\_forecast();
28. delay(3000);
29. }
30. /\*\*\*\*\*\*\*\*连接wifi函数\*\*\*\*\*\*\*\*/
31. **void** connect\_wifi()
32. {
33. WiFi.begin(wifi\_ssid,wifi\_password);
34. **while**(WiFi.status() != WL\_CONNECTED)
35. {
36. delay(10);
37. }
38. }
39. /\*\*\*\*\*\*\*\*检验wifi连接是否成功\*\*\*\*\*\*\*\*/
40. **void** wifi\_connect\_ok()
41. {
42. **if**(WiFi.status() == WL\_CONNECTED)
43. {
44. digitalWrite(LED\_BUILTIN,LOW);
45. }
46. }
47. **void** get\_weather\_forecast()
48. {
49. //https://api.seniverse.com/v3/weather/daily.json?key=S-9a3-Z4sX\_eWvo7H&location=beijing&language=zh-Hans&unit=c&start=0&days=5
50. String temp = "/v3/weather/daily.json?key=" + user\_key +
51. "&location=" + location + "&language=" + language +
52. "&unit=" +  unit + "&start=" + start + "&days=" + days;
53. String http\_request = "GET " + temp + " HTTP/1.1\r\n" +
54. "host: " + host + "\r\n" +
55. "Connection: close\r\n" +
56. "\r\n";
57. WiFiClient http\_client;
58. **if**(http\_client.connect(host,port))
59. {
60. http\_client.print(http\_request);
61. **if**(http\_client.find("\r\n\r\n"))
62. {
63. String json = http\_client.readString();
64. parse\_json(json);
65. }
66. //    while(http\_client.connected() || http\_client.available())
67. //    {
68. //      String data = http\_client.readStringUntil('\n');
69. //      Serial.println(data);
70. //    }
72. http\_client.stop();
73. }
74. }
75. /\*\*\*\*\*\*\*\*解析JSON函数\*\*\*\*\*\*\*\*/
76. **void** parse\_json(String json)
77. {
78. //ArduinoJson库作者网站自动生成代码，库版本V5
79. **const** **size\_t** capacity = JSON\_ARRAY\_SIZE(1) + JSON\_ARRAY\_SIZE(3) + JSON\_OBJECT\_SIZE(1) + JSON\_OBJECT\_SIZE(3) + JSON\_OBJECT\_SIZE(6) + 3\*JSON\_OBJECT\_SIZE(14) + 860;
80. DynamicJsonBuffer jsonBuffer(capacity);
81. JsonObject& root = jsonBuffer.parseObject(json);
82. JsonObject& results\_0 = root["results"][0];
83. JsonObject& results\_0\_location = results\_0["location"];
84. **const** **char**\* results\_0\_location\_id = results\_0\_location["id"]; // "WWH1780CEFBR"
85. **const** **char**\* results\_0\_location\_name = results\_0\_location["name"]; // "Suqian"
86. **const** **char**\* results\_0\_location\_country = results\_0\_location["country"]; // "CN"
87. **const** **char**\* results\_0\_location\_path = results\_0\_location["path"]; // "Suqian,Suqian,Jiangsu,China"
88. **const** **char**\* results\_0\_location\_timezone = results\_0\_location["timezone"]; // "Asia/Shanghai"
89. **const** **char**\* results\_0\_location\_timezone\_offset = results\_0\_location["timezone\_offset"]; // "+08:00"
90. JsonArray& results\_0\_daily = results\_0["daily"];
91. JsonObject& results\_0\_daily\_0 = results\_0\_daily[0];
92. **const** **char**\* results\_0\_daily\_0\_date = results\_0\_daily\_0["date"]; // "2022-01-28"
93. **const** **char**\* results\_0\_daily\_0\_text\_day = results\_0\_daily\_0["text\_day"]; // "Overcast"
94. **const** **char**\* results\_0\_daily\_0\_code\_day = results\_0\_daily\_0["code\_day"]; // "9"
95. **const** **char**\* results\_0\_daily\_0\_text\_night = results\_0\_daily\_0["text\_night"]; // "Overcast"
96. **const** **char**\* results\_0\_daily\_0\_code\_night = results\_0\_daily\_0["code\_night"]; // "9"
97. **const** **char**\* results\_0\_daily\_0\_high = results\_0\_daily\_0["high"]; // "3"
98. **const** **char**\* results\_0\_daily\_0\_low = results\_0\_daily\_0["low"]; // "-1"
99. **const** **char**\* results\_0\_daily\_0\_rainfall = results\_0\_daily\_0["rainfall"]; // "0.00"
100. **const** **char**\* results\_0\_daily\_0\_precip = results\_0\_daily\_0["precip"]; // "0.00"
101. **const** **char**\* results\_0\_daily\_0\_wind\_direction = results\_0\_daily\_0["wind\_direction"]; // "NE"
102. **const** **char**\* results\_0\_daily\_0\_wind\_direction\_degree = results\_0\_daily\_0["wind\_direction\_degree"]; // "45"
103. **const** **char**\* results\_0\_daily\_0\_wind\_speed = results\_0\_daily\_0["wind\_speed"]; // "8.4"
104. **const** **char**\* results\_0\_daily\_0\_wind\_scale = results\_0\_daily\_0["wind\_scale"]; // "2"
105. **const** **char**\* results\_0\_daily\_0\_humidity = results\_0\_daily\_0["humidity"]; // "54"
106. JsonObject& results\_0\_daily\_1 = results\_0\_daily[1];
107. **const** **char**\* results\_0\_daily\_1\_date = results\_0\_daily\_1["date"]; // "2022-01-29"
108. **const** **char**\* results\_0\_daily\_1\_text\_day = results\_0\_daily\_1["text\_day"]; // "Cloudy"
109. **const** **char**\* results\_0\_daily\_1\_code\_day = results\_0\_daily\_1["code\_day"]; // "4"
110. **const** **char**\* results\_0\_daily\_1\_text\_night = results\_0\_daily\_1["text\_night"]; // "Cloudy"
111. **const** **char**\* results\_0\_daily\_1\_code\_night = results\_0\_daily\_1["code\_night"]; // "4"
112. **const** **char**\* results\_0\_daily\_1\_high = results\_0\_daily\_1["high"]; // "3"
113. **const** **char**\* results\_0\_daily\_1\_low = results\_0\_daily\_1["low"]; // "-2"
114. **const** **char**\* results\_0\_daily\_1\_rainfall = results\_0\_daily\_1["rainfall"]; // "0.00"
115. **const** **char**\* results\_0\_daily\_1\_precip = results\_0\_daily\_1["precip"]; // "0.00"
116. **const** **char**\* results\_0\_daily\_1\_wind\_direction = results\_0\_daily\_1["wind\_direction"]; // "NE"
117. **const** **char**\* results\_0\_daily\_1\_wind\_direction\_degree = results\_0\_daily\_1["wind\_direction\_degree"]; // "45"
118. **const** **char**\* results\_0\_daily\_1\_wind\_speed = results\_0\_daily\_1["wind\_speed"]; // "8.4"
119. **const** **char**\* results\_0\_daily\_1\_wind\_scale = results\_0\_daily\_1["wind\_scale"]; // "2"
120. **const** **char**\* results\_0\_daily\_1\_humidity = results\_0\_daily\_1["humidity"]; // "52"
121. JsonObject& results\_0\_daily\_2 = results\_0\_daily[2];
122. **const** **char**\* results\_0\_daily\_2\_date = results\_0\_daily\_2["date"]; // "2022-01-30"
123. **const** **char**\* results\_0\_daily\_2\_text\_day = results\_0\_daily\_2["text\_day"]; // "Cloudy"
124. **const** **char**\* results\_0\_daily\_2\_code\_day = results\_0\_daily\_2["code\_day"]; // "4"
125. **const** **char**\* results\_0\_daily\_2\_text\_night = results\_0\_daily\_2["text\_night"]; // "Cloudy"
126. **const** **char**\* results\_0\_daily\_2\_code\_night = results\_0\_daily\_2["code\_night"]; // "4"
127. **const** **char**\* results\_0\_daily\_2\_high = results\_0\_daily\_2["high"]; // "6"
128. **const** **char**\* results\_0\_daily\_2\_low = results\_0\_daily\_2["low"]; // "-3"
129. **const** **char**\* results\_0\_daily\_2\_rainfall = results\_0\_daily\_2["rainfall"]; // "0.00"
130. **const** **char**\* results\_0\_daily\_2\_precip = results\_0\_daily\_2["precip"]; // "0.00"
131. **const** **char**\* results\_0\_daily\_2\_wind\_direction = results\_0\_daily\_2["wind\_direction"]; // "S"
132. **const** **char**\* results\_0\_daily\_2\_wind\_direction\_degree = results\_0\_daily\_2["wind\_direction\_degree"]; // "180"
133. **const** **char**\* results\_0\_daily\_2\_wind\_speed = results\_0\_daily\_2["wind\_speed"]; // "3.0"
134. **const** **char**\* results\_0\_daily\_2\_wind\_scale = results\_0\_daily\_2["wind\_scale"]; // "1"
135. **const** **char**\* results\_0\_daily\_2\_humidity = results\_0\_daily\_2["humidity"]; // "52"
136. **const** **char**\* results\_0\_last\_update = results\_0["last\_update"]; // "2022-01-28T08:00:00+08:00"
137. /\*----------串口输出今天及未来两天天气情况----------\*/
138. Serial.println(results\_0\_location\_name);   //位置：Suqian
139. Serial.println(results\_0\_location\_country);//国家：CN
140. Serial.println(results\_0\_location\_path);   //路径：Suqian,Suqian,Jiangsu,China
141. //2022.1.28天气情况
142. Serial.println("--------------------");
143. Serial.println(results\_0\_daily\_0\_date);                 //日期：2022-01-28
144. Serial.println(results\_0\_daily\_0\_text\_day);             //白天天气：Overcast
145. Serial.println(results\_0\_daily\_0\_code\_day);             //白天天气码：9
146. Serial.println(results\_0\_daily\_0\_text\_night);           //夜晚天气：Overcast
147. Serial.println(results\_0\_daily\_0\_code\_night);           //夜晚天气码：9
148. Serial.println(results\_0\_daily\_0\_high);                 //最高温度：3
149. Serial.println(results\_0\_daily\_0\_low);                  //最低温度：-1
150. Serial.println(results\_0\_daily\_0\_rainfall);             //降水量：0.00   （注意：单位为mm）
151. Serial.println(results\_0\_daily\_0\_precip);               //降水概率：0.00 （注意：单位为%）
152. Serial.println(results\_0\_daily\_0\_wind\_direction);       //风向：NE
153. Serial.println(results\_0\_daily\_0\_wind\_direction\_degree);//风向角度：45
154. Serial.println(results\_0\_daily\_0\_wind\_speed);           //风速：8.4   （注意：单位为km/h）
155. Serial.println(results\_0\_daily\_0\_wind\_scale);           //风力等级：2
156. Serial.println(results\_0\_daily\_0\_humidity);             //相对湿度：54 （注意：单位为%）
157. //2022.1.29天气情况
158. Serial.println("--------------------");
159. Serial.println(results\_0\_daily\_1\_date);
160. Serial.println(results\_0\_daily\_1\_text\_day);
161. Serial.println(results\_0\_daily\_1\_code\_day);
162. Serial.println(results\_0\_daily\_1\_text\_night);
163. Serial.println(results\_0\_daily\_1\_code\_night);
164. Serial.println(results\_0\_daily\_1\_high);
165. Serial.println(results\_0\_daily\_1\_low);
166. Serial.println(results\_0\_daily\_1\_rainfall);
167. Serial.println(results\_0\_daily\_1\_precip);
168. Serial.println(results\_0\_daily\_1\_wind\_direction);
169. Serial.println(results\_0\_daily\_1\_wind\_direction\_degree);
170. Serial.println(results\_0\_daily\_1\_wind\_speed);
171. Serial.println(results\_0\_daily\_1\_wind\_scale);
172. Serial.println(results\_0\_daily\_1\_humidity);
173. //2022.1.30天气情况
174. Serial.println("--------------------");
175. Serial.println(results\_0\_daily\_2\_date);
176. Serial.println(results\_0\_daily\_2\_text\_day);
177. Serial.println(results\_0\_daily\_2\_code\_day);
178. Serial.println(results\_0\_daily\_2\_text\_night);
179. Serial.println(results\_0\_daily\_2\_code\_night);
180. Serial.println(results\_0\_daily\_2\_high);
181. Serial.println(results\_0\_daily\_2\_low);
182. Serial.println(results\_0\_daily\_2\_rainfall);
183. Serial.println(results\_0\_daily\_2\_precip);
184. Serial.println(results\_0\_daily\_2\_wind\_direction);
185. Serial.println(results\_0\_daily\_2\_wind\_direction\_degree);
186. Serial.println(results\_0\_daily\_2\_wind\_speed);
187. Serial.println(results\_0\_daily\_2\_wind\_scale);
188. Serial.println(results\_0\_daily\_2\_humidity);
189. //数据更新时间
190. Serial.println("--------------------");
191. Serial.println(results\_0\_last\_update);
192. }

被解析json：

1. {
2. "results":
3. [
4. {
5. "location":
6. {
7. "id":"WWH1780CEFBR",
8. "name":"Suqian",
9. "country":"CN",
10. "path":"Suqian,Suqian,Jiangsu,China",
11. "timezone":"Asia/Shanghai",
12. "timezone\_offset":"+08:00"
14. },
15. "daily":
16. [
17. {
18. "date":"2022-01-28",
19. "text\_day":"Overcast",
20. "code\_day":"9",
21. "text\_night":"Overcast",
22. "code\_night":"9",
23. "high":"3",
24. "low":"-1",
25. "rainfall":"0.00",
26. "precip":"0.00",
27. "wind\_direction":"NE",
28. "wind\_direction\_degree":"45",
29. "wind\_speed":"8.4",
30. "wind\_scale":"2",
31. "humidity":"54"
32. },
33. {
34. "date":"2022-01-29",
35. "text\_day":"Cloudy",
36. "code\_day":"4",
37. "text\_night":"Cloudy",
38. "code\_night":"4",
39. "high":"3",
40. "low":"-2",
41. "rainfall":"0.00",
42. "precip":"0.00",
43. "wind\_direction":"NE",
44. "wind\_direction\_degree":"45",
45. "wind\_speed":"8.4",
46. "wind\_scale":"2",
47. "humidity":"52"
48. },
49. {
50. "date":"2022-01-30",
51. "text\_day":"Cloudy",
52. "code\_day":"4",
53. "text\_night":"Cloudy",
54. "code\_night":"4",
55. "high":"6",
56. "low":"-3",
57. "rainfall":"0.00",
58. "precip":"0.00",
59. "wind\_direction":"S",
60. "wind\_direction\_degree":"180",
61. "wind\_speed":"3.0",
62. "wind\_scale":"1",
63. "humidity":"52"
64. }
65. ],
66. "last\_update":"2022-01-28T08:00:00+08:00"
67. }
68. ]
69. }

**3、获取心知天气实时生活指数（洗车、穿衣、流感、运动、旅行、紫外线共六项建议）**

1. #include <ESP8266WiFi.h> //引入ESP8266WiFi库
2. #include <ArduinoJson.h> //引入ArduinoJson库
3. //所连接wifi名称与密码
4. String wifi\_ssid = "TP-LINK\_1CD4";
5. String wifi\_password = "15150757563";
6. //心知天气qpi接口与端口
7. String host = "api.seniverse.com";
8. **int** port = 80;
9. //获取生活指数必备信息
10. String user\_key = "S-9a3-Z4sX\_eWvo7H";
11. String location = "Suqian";
12. String language = "en";
13. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
14. **void** setup()
15. {
16. Serial.begin(9600);         //串口初始化波特率为9600
17. pinMode(LED\_BUILTIN,OUTPUT);//设置LED引脚为输出模式
18. wifi\_connect();             //连接wifi
19. }
20. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
21. **void** loop()
22. {
23. wifi\_connect\_ok();//检测WiFi是否连接成功，若连接成功打开蓝色LED
24. get\_life\_index(); //获取实时生活指数
25. delay(3000);      //延时3s，心知天气免费用户请求频率限制：20次/分
26. }
27. /\*\*\*\*\*\*\*\*连接WiFi\*\*\*\*\*\*\*\*/
28. **void** wifi\_connect()
29. {
30. WiFi.begin(wifi\_ssid,wifi\_password);//连接wifi
31. **while**(WiFi.status() != WL\_CONNECTED)//等待连接完成
32. {
33. delay(10);
34. }
35. }
36. /\*\*\*\*\*\*\*\*检验wifi是否连接成功\*\*\*\*\*\*\*\*/
37. **void** wifi\_connect\_ok()
38. {
39. //若连接成功。打开LED指示灯
40. **if**(WiFi.status() == WL\_CONNECTED)
41. {
42. digitalWrite(LED\_BUILTIN,LOW);
43. }
44. }
45. /\*\*\*\*\*\*\*\*获取实时生活指数\*\*\*\*\*\*\*\*/
46. **void** get\_life\_index()
47. {
48. //获取生活指数网址：https://api.seniverse.com/v3/life/suggestion.json?key=S-9a3-Z4sX\_eWvo7H&location=shanghai&language=zh-Hans
49. String temp = "/v3/life/suggestion.json?key=" + user\_key + "&location=" + location + "&language=" + language;
50. String http\_request = "GET " + temp + " HTTP/1.1\r\n" + //http请求
51. "host: " + host + "\r\n" +
52. "Connection: close\r\n" +
53. "\r\n";
54. WiFiClient http\_client; //建立WiFiClient对象
55. **if**(http\_client.connect(host,port)) //连接服务器，若成功连接返回真，否则返回假
56. {
57. http\_client.print(http\_request); //发送http请求
58. **if**(http\_client.find("\r\n\r\n")) //寻找响应体部分
59. {
60. String json = http\_client.readString(); //读取响应体json
61. parse\_json(json);                       //调用json解析函数
62. }
63. //    此部分用于接受所有相应内容，用于查看响应体中json，便于在线网站生成解析json代码
64. //    while(http\_client.connected() || http\_client.available())
65. //    {
66. //      String data = http\_client.readStringUntil('\n');
67. //      Serial.println(data);
68. //    }
69. http\_client.stop(); //断开与服务器连接
70. }
71. }
72. /\*\*\*\*\*\*\*\*解析json\*\*\*\*\*\*\*\*/
73. **void** parse\_json(String json)
74. {
75. /\*----------以下为Arduinoson库作者的网站生成的自动解析JSON代码，版本V5----------\*/
76. //https://arduinojson.org/v5/assistant/
77. **const** **size\_t** capacity = JSON\_ARRAY\_SIZE(1) + JSON\_OBJECT\_SIZE(1) + 6\*JSON\_OBJECT\_SIZE(2) + JSON\_OBJECT\_SIZE(3) + 2\*JSON\_OBJECT\_SIZE(6) + 400;
78. DynamicJsonBuffer jsonBuffer(capacity);
79. JsonObject& root = jsonBuffer.parseObject(json);
80. JsonObject& results\_0 = root["results"][0];
81. JsonObject& results\_0\_location = results\_0["location"];
82. **const** **char**\* results\_0\_location\_id = results\_0\_location["id"]; // "WWH1780CEFBR"
83. **const** **char**\* results\_0\_location\_name = results\_0\_location["name"]; // "Suqian"
84. **const** **char**\* results\_0\_location\_country = results\_0\_location["country"]; // "CN"
85. **const** **char**\* results\_0\_location\_path = results\_0\_location["path"]; // "Suqian,Suqian,Jiangsu,China"
86. **const** **char**\* results\_0\_location\_timezone = results\_0\_location["timezone"]; // "Asia/Shanghai"
87. **const** **char**\* results\_0\_location\_timezone\_offset = results\_0\_location["timezone\_offset"]; // "+08:00"
88. JsonObject& results\_0\_suggestion = results\_0["suggestion"];
89. **const** **char**\* results\_0\_suggestion\_car\_washing\_brief = results\_0\_suggestion["car\_washing"]["brief"]; // "Good"
90. **const** **char**\* results\_0\_suggestion\_dressing\_brief = results\_0\_suggestion["dressing"]["brief"]; // "Cold"
91. **const** **char**\* results\_0\_suggestion\_flu\_brief = results\_0\_suggestion["flu"]["brief"]; // "Probably"
92. **const** **char**\* results\_0\_suggestion\_sport\_brief = results\_0\_suggestion["sport"]["brief"]; // "Not Good"
93. **const** **char**\* results\_0\_suggestion\_travel\_brief = results\_0\_suggestion["travel"]["brief"]; // "Good"
94. **const** **char**\* results\_0\_suggestion\_uv\_brief = results\_0\_suggestion["uv"]["brief"]; // "Very Low"
95. **const** **char**\* results\_0\_last\_update = results\_0["last\_update"]; // "2022-01-29T08:23:04+08:00"
96. /\*----------以上为Arduinoson库作者的网站生成的自动解析JSON代码，版本V5----------\*/
97. //串口输出实时生活指数
98. Serial.println(results\_0\_location\_name);   //位置：Suqian
99. Serial.println(results\_0\_location\_country);//国家：CN
100. Serial.println(results\_0\_location\_path);   //路径：Suqian,Suqian,Jiangsu,China
101. Serial.println("--------------------");
102. Serial.println(results\_0\_suggestion\_car\_washing\_brief);//洗车概要：Good
103. Serial.println(results\_0\_suggestion\_dressing\_brief);   //穿衣概要：Cold
104. Serial.println(results\_0\_suggestion\_flu\_brief);        //流感概要：Probably
105. Serial.println(results\_0\_suggestion\_sport\_brief);      //运动概要：Not Good
106. Serial.println(results\_0\_suggestion\_travel\_brief);     //旅游概要：Good
107. Serial.println(results\_0\_suggestion\_uv\_brief);         //紫外线概要：Very Low
108. Serial.println(results\_0\_last\_update);                 //上次更新时间：2022-01-29T08:23:04+08:00
109. }

被解析json格式：

1. {
2. "results":
3. [
4. {
5. "location":
6. {
7. "id":"WWH1780CEFBR",
8. "name":"Suqian",
9. "country":"CN",
10. "path":"Suqian,Suqian,Jiangsu,China",
11. "timezone":"Asia/Shanghai",
12. "timezone\_offset":"+08:00"
13. },
14. "suggestion":
15. {
16. "car\_washing":
17. {
18. "brief":"Good",
19. "details":""
20. },
21. "dressing":
22. {
23. "brief":"Cold",
24. "details":""
25. },
26. "flu":
27. {
28. "brief":"Probably",
29. "details":""
30. },
31. "sport":
32. {
33. "brief":"Not Good",
34. "details":""
35. },
36. "travel":
37. {
38. "brief":"Good",
39. "details":""
40. },
41. "uv":
42. {
43. "brief":"Very Low",
44. "details":""
45. }
46. },
47. "last\_update":"2022-01-29T08:23:04+08:00"
48. }
49. ]
50. }

**4、利用ESP8266\_Seniverse库实现以上3中功能**

**1）、获取实时天气信息**

1. #include <ESP8266WiFi.h> //引入ESP8266WiFi库
2. #include <ArduinoJson.h> //引入ArduinoJson库
3. #include <ESP8266\_Seniverse.h> //引入ESP8266\_Seniverse库
4. //被连接WiFi名称与密码
5. String wifi\_ssid = "TP-LINK\_1CD4";
6. String wifi\_password = "15150757563";
7. //获取心知天气信息必备信息
8. String user\_key = "S-9a3-Z4sX\_eWvo7H";
9. String location = "Suqian";
10. String unit = "c";
11. //创建WeatherNow
12. WeatherNow weather\_now;
13. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
14. **void** setup()
15. {
16. Serial.begin(9600); //串口初始画波特率为9600
17. WiFi.begin(wifi\_ssid,wifi\_password); //连接WiFi
18. **while**(WiFi.status() != WL\_CONNECTED) //等待连接成功
19. {
20. delay(10);
21. }
23. weather\_now.config(user\_key,location,unit); //配置心知天气
24. }
25. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
26. **void** loop()
27. {
28. **if**(weather\_now.update()) //如果成功连接至心知天气服务器
29. {
30. Serial.println(weather\_now.getWeatherText()); //获取实时天气
31. Serial.println(weather\_now.getWeatherCode()); //获取实时天气对应代码
32. Serial.println(weather\_now.getDegree());      //获取实时温度
33. Serial.println(weather\_now.getLastUpdate());  //获取上次更新时间
34. }
35. delay(3000); //延时3s
36. }

**2）、获取近三天天气预报信息**

略，与上面程序类似（详情可参考ESP8266参考库文件中关于ESP8266\_Seniverse库描述）

**3）、获取实时生活指数**

略，与上面程序类似（详情可参考ESP8266参考库文件中关于ESP8266\_Seniverse库描述）

（八）、Ticker库多任务处理

**1、呼吸灯＋串口输出**

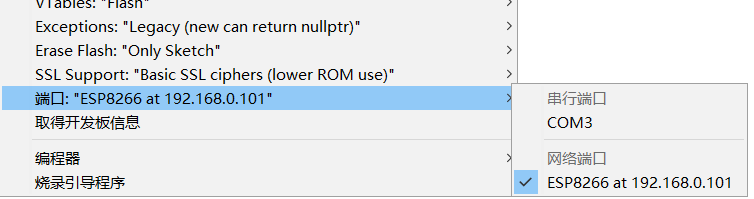
1. #include <Ticker.h> //引入Ticker库
2. Ticker ticker; //建立Ticker对象
4. **void** setup()
5. {
6. Serial.begin(9600);         //初始化串口
7. pinMode(LED\_BUILTIN,OUTPUT);//设置LED为输出模式
9. ticker.attach(1,say\_hi);    //每间隔1s执行say\_hi函数
10. //ticker.attach(1,say\_hi,10); //每间隔1s执行say\_hi函数，执行10次
11. }
12. **void** loop()
13. {
14. //呼吸灯
15. **for**(**int** i=0;i<1024;i++)
16. {  analogWrite(LED\_BUILTIN,i);
17. delay(1);  }
18. **for**(**int** i=1023;i>-1;i--)
19. {  analogWrite(LED\_BUILTIN,i);
20. delay(1);  }
21. }
22. **void** say\_hi()
23. {
24. **static** **int** count = 0;
25. Serial.print(count);
26. Serial.println("hi");
27. count++;
28. }
29. //void say\_hi(int a)
30. //{
31. //  static int count = 0;
32. //  Serial.print(count);
33. //  Serial.println("hi");
34. //  count++;
35. //  if(count == a)
36. //  {
37. //    ticker.detach(); //停止ticker对象调用say\_hi函数
38. //  }
39. //}

（九）、ArduinoOTA库

**1、OTA修改LED闪烁时间**

1. #include <ESP8266WiFi.h> //引入ESP8266WiFi库
2. #include <ArduinoOTA.h>  //引入ArduinoOTA库
3. #include <Ticker.h>      //引入Ticker库
4. //被连接wifi名与密码
5. String wifi\_ssid = "TP-LINK\_1CD4";
6. String wifi\_password = "15150757563";
7. //建立Ticker对象
8. Ticker ticker;
9. /\*\*\*\*\*\*\*\*配置函数\*\*\*\*\*\*\*\*/
10. **void** setup()
11. {
12. connect\_wifi(); //连接wifi
14. pinMode(LED\_BUILTIN,OUTPUT);     //设置LED引脚为输出模式
15. ticker.attach\_ms(1000,LED\_blink);//每1秒调用LED\_blink函数
17. ArduinoOTA.setHostname("ESP8266"); //设置端口名为ESP8266
18. ArduinoOTA.setPassword("12345678");//设置连接密码为12345678
19. ArduinoOTA.begin();                //启动OTA
20. }
21. /\*\*\*\*\*\*\*\*主循环函数\*\*\*\*\*\*\*\*/
22. **void** loop()
23. {
24. ArduinoOTA.handle(); //处理OTA下载程序请求
25. }
26. /\*\*\*\*\*\*\*\*连接wifi\*\*\*\*\*\*\*\*/
27. **void** connect\_wifi()
28. {
29. WiFi.begin(wifi\_ssid,wifi\_password);
30. **while**(WiFi.status() != WL\_CONNECTED)
31. {
32. delay(10);
33. }
34. }
35. /\*\*\*\*\*\*\*\*LED闪烁\*\*\*\*\*\*\*\*/
36. **void** LED\_blink()
37. {
38. digitalWrite(LED\_BUILTIN,!digitalRead(LED\_BUILTIN));
39. }

注：下载完程序，关闭重启Arduino IDE，即可在端口出选择OTA端口



网络端口中ESP8266 at 192.168.0.101中的ESP8266即为以下程序所建立。



可以通过修改attach\_ms中时间间隔，重新通过WiFi将程序下载进开发板。