**基本**

一、硬件基本概况

1.引脚：

（1）ESP8266芯片共有17个GPIO引脚（GPIO0~GPIO16）。这些引脚中的GPIO6~GPIO11被用于连接开发板的闪存（Flash Memory）。如果在实验电路中使用GPIO6~GPIO11，NodeMCU开发板无法正常工作。因此建议不适用这些引脚

（2）输入输出电压限制为3.3V，（小于arduino的5V，两者串联时需要注意），若外加3.6V的电压，则可能对芯片造成损害。引脚最大输入电流为：12mA

（3）只有一个模拟输入引脚，对于8266芯片，最大输入电压为1.0V，而对于整体的NodeMCU开发板，最大输入电压仍为3.3V

2.其他引脚及其特点在学习中记录

二、搭建开发环境

1.安装驱动

2.导入8266库文件

（1）下载esp8266的驱动，并且在选择数据线时要选择具有数据传输功能的线，我试了三根，前两根质量太差都只能充电

（2）在开发板管理器中添加esp8266的索引

<http://arduino.esp8266.com/stable/package_esp8266com_index.json>（下载速度惊人）

（3）我利用以上常规方法无法下载，下载其他安装包来辅助完成

（4）下载第三方库文件

三、接入点模式（Access Point，AP）

1.建立一个WiFi的代码

/\*

NodeMCU接入点模式

网络名: taichi-maker

密码：12345678

\*/

#include <ESP8266WiFi.h>        // 本程序使用ESP8266WiFi库

const char \*ssid = "taichi-maker"; // 这里定义将要建立的WiFi名称。此处以"taichi-maker"为示例

                                   // 您可以将自己想要建立的WiFi名称填写入此处的双引号中

const char \*password = "12345678";  // 这里定义将要建立的WiFi密码。此处以12345678为示例

                                    // 您可以将自己想要使用的WiFi密码放入引号内

                                    // 如果建立的WiFi不要密码，则在双引号内不要填入任何信息

void **setup**() {

**Serial**.begin(9600);              // 启动串口通讯

  WiFi.softAP(ssid, password);     // 此语句是重点。WiFi.softAP用于启动NodeMCU的AP模式。

                                   // 括号中有两个参数，ssid是WiFi名。password是WiFi密码。

                                   // 这两个参数具体内容在setup函数之前的位置进行定义。

**Serial**.print("Access Point: ");    // 通过串口监视器输出信息

**Serial**.println(ssid);              // 告知用户NodeMCU所建立的WiFi名

**Serial**.print("IP address: ");      // 以及NodeMCU的IP地址

**Serial**.println(WiFi.softAPIP());   // 通过调用WiFi.softAPIP()可以得到NodeMCU的IP地址

}

void **loop**() {

}

2.在点击串口监视器时，要按下“RST”才能看到串口的内容，这是不同于Arduino的，对于Arduino按下串口监视的同时setup程序便能自己运行一遍

3.验证是否连接成功，打开命令提示符ping一下IP地址就可以了。我第一次丢包率在40%，ping第二次就好了。

重要 四、无线终端模式（wireless Station）

1.手动连接WiFi

（1）代码

#include <ESP8266WiFi.h>        // 本程序使用ESP8266WiFi库

const char\* ssid     = "taichi-maker";      // 连接WiFi名（此处使用taichi-maker为示例）

                                            // 请将您需要连接的WiFi名填入引号中

const char\* password = "12345678";          // 连接WiFi密码（此处使用12345678为示例）

                                            // 请将您需要连接的WiFi密码填入引号中

void **setup**() {

**Serial**.begin(9600);         // 启动串口通讯

  WiFi.begin(ssid, password);                  // 启动网络连接

**Serial**.print("Connecting to ");              // 串口监视器输出网络连接信息

**Serial**.print(ssid); **Serial**.println(" ...");  // 告知用户NodeMCU正在尝试WiFi连接

  int i = 0;                                   // 这一段程序语句用于检查WiFi是否连接成功

**while** (WiFi.status() != WL\_CONNECTED) {      // WiFi.status()函数的返回值是由NodeMCU的WiFi连接状态所决定的。

**delay**(1000);                               // 如果WiFi连接成功则返回值为WL\_CONNECTED

**Serial**.print(i++); **Serial**.print(' ');      // 此处通过While循环让NodeMCU每隔一秒钟检查一次WiFi.status()函数返回值

  }                                            // 同时NodeMCU将通过串口监视器输出连接时长读秒。

                                               // 这个读秒是通过变量i每隔一秒自加1来实现的。

**Serial**.println("");                          // WiFi连接成功后

**Serial**.println("Connection established!");   // NodeMCU将通过串口监视器输出"连接成功"信息。

**Serial**.print("IP address:    ");             // 同时还将输出NodeMCU的IP地址。这一功能是通过调用

**Serial**.println(WiFi.localIP());              // WiFi.localIP()函数来实现的。该函数的返回值即NodeMCU的IP地址。

}

void **loop**() {

}

（2）串口监视器找IP地址

（3）在cmd中ping一下IP地址

2.自动连接信号最强的WiFi

（1）代码

#include <ESP8266WiFi.h>          // 本程序使用ESP8266WiFi库

#include <ESP8266WiFiMulti.h>   // 本程序使用ESP8266WiFiMulti库

ESP8266WiFiMulti wifiMulti;     // 建立ESP8266WiFiMulti对象,对象名称是'wifiMulti'

void **setup**() {

**Serial**.begin(9600);            // 启动串口通讯

//通过addAp函数存储  WiFi名称       WiFi密码

  wifiMulti.addAP("taichi-maker", "12345678");  // 这三条语句通过调用函数addAP来记录3个不同的WiFi网络信息。

  wifiMulti.addAP("taichi-maker2", "87654321"); // 这3个WiFi网络名称分别是taichi-maker, taichi-maker2, taichi-maker3。

  wifiMulti.addAP("taichi-maker3", "13572468"); // 这3个网络的密码分别是123456789，87654321，13572468。

                                                // 此处WiFi信息只是示例，请在使用时将需要连接的WiFi信息填入相应位置。

                                                // 另外这里只存储了3个WiFi信息，您可以存储更多的WiFi信息在此处。

**Serial**.println("Connecting ...");         // 通过串口监视器输出信息告知用户NodeMCU正在尝试连接WiFi

  int i = 0;

**while** (wifiMulti.run() != WL\_CONNECTED) {  // 此处的wifiMulti.run()是重点。通过wifiMulti.run()，NodeMCU将会在当前

**delay**(1000);                             // 环境中搜索addAP函数所存储的WiFi。如果搜到多个存储的WiFi那么NodeMCU

**Serial**.print('.');                       // 将会连接信号最强的那一个WiFi信号。

  }                                           // 一旦连接WiFI成功，wifiMulti.run()将会返回“WL\_CONNECTED”。这也是

                                              // 此处while循环判断是否跳出循环的条件。

**Serial**.println('\n');                     // WiFi连接成功后

**Serial**.print("Connected to ");            // NodeMCU将通过串口监视器输出。

**Serial**.println(WiFi.SSID());              // 连接的WiFI名称

**Serial**.print("IP address:\t");            // 以及

**Serial**.println(WiFi.localIP());           // NodeMCU的IP地址

}

void **loop**() {

}

（2）串口监视器中查看IP地址

（3）cmd中ping一下IP地址

5.这样ESP8266就成功配网了！完成进入物联网的第一步！

**网络服务器**

 一、建立基本的网络服务器

1.网络服务器端口：80

2.text/plain 纯文本信息

3.代码

#include <ESP8266WiFi.h>        // 本程序使用 ESP8266WiFi库

#include <ESP8266WiFiMulti.h>   //  ESP8266WiFiMulti库

#include <ESP8266WebServer.h>   //  ESP8266WebServer库

ESP8266WiFiMulti wifiMulti;     // 建立ESP8266WiFiMulti对象,对象名称是'wifiMulti'

ESP8266WebServer esp8266\_server(80);// 建立ESP8266WebServer对象，对象名称为esp8266\_server

                                    // 括号中的数字是网路服务器响应http请求的端口号

                                    // 网络服务器标准http端口号为80，因此这里使用80为端口号

void **setup**(void){

**Serial**.begin(9600);          // 启动串口通讯

  //通过addAp函数存储  WiFi名称       WiFi密码

  wifiMulti.addAP("taichi-maker", "12345678");  // 这三条语句通过调用函数addAP来记录3个不同的WiFi网络信息。

  wifiMulti.addAP("taichi-maker2", "87654321"); // 这3个WiFi网络名称分别是taichi-maker, taichi-maker2, taichi-maker3。

  wifiMulti.addAP("taichi-maker3", "13572468"); // 这3个网络的密码分别是123456789，87654321，13572468。

                                                // 此处WiFi信息只是示例，请在使用时将需要连接的WiFi信息填入相应位置。

                                                // 另外这里只存储了3个WiFi信息，您可以存储更多的WiFi信息在此处。

  int i = 0;

**while** (wifiMulti.run() != WL\_CONNECTED) {  // 此处的wifiMulti.run()是重点。通过wifiMulti.run()，NodeMCU将会在当前

**delay**(1000);                             // 环境中搜索addAP函数所存储的WiFi。如果搜到多个存储的WiFi那么NodeMCU

**Serial**.print(i++); **Serial**.print(' ');    // 将会连接信号最强的那一个WiFi信号。

  }                                          // 一旦连接WiFI成功，wifiMulti.run()将会返回“WL\_CONNECTED”。这也是

                                             // 此处while循环判断是否跳出循环的条件。

  // WiFi连接成功后将通过串口监视器输出连接成功信息

**Serial**.println('\n');                     // WiFi连接成功后

**Serial**.print("Connected to ");            // NodeMCU将通过串口监视器输出。

**Serial**.println(WiFi.SSID());              // 连接的WiFI名称

**Serial**.print("IP address:\t");            // 以及

**Serial**.println(WiFi.localIP());           // NodeMCU的IP地址

//--------"启动网络服务功能"程序部分开始-------- //  此部分为程序为本示例程序重点1

  esp8266\_server.begin();                   //  详细讲解请参见太极创客网站《零基础入门学用物联网》

  esp8266\_server.on("/", handleRoot);       //  第3章-第2节 ESP8266-NodeMCU网络服务器-1

  esp8266\_server.onNotFound(handleNotFound);

//--------"启动网络服务功能"程序部分结束--------

**Serial**.println("HTTP esp8266\_server started");//  告知用户ESP8266网络服务功能已经启动

}

/\* 以下函数语句为本示例程序重点3

详细讲解请参见太极创客网站《零基础入门学用物联网》

第3章-第2节 3\_2\_1\_First\_Web\_Server 的说明讲解\*/

void **loop**(void){

  esp8266\_server.handleClient();     // 处理http服务器访问

}

/\* 以下两个函数为本示例程序重点2

详细讲解请参见太极创客网站《零基础入门学用物联网》

第3章-第2节 3\_2\_1\_First\_Web\_Server 的说明讲解\*/

void handleRoot() {   //处理网站根目录“/”的访问请求

  esp8266\_server.send(200, "text/plain", "Hello from ESP8266");   // NodeMCU将调用此函数。

}

// 设置处理404情况的函数'handleNotFound'

void handleNotFound(){                                        // 当浏览器请求的网络资源无法在服务器找到时，

  esp8266\_server.send(404, "text/plain", "404: Not found");   // NodeMCU将调用此函数。

}

4.服务器响应头：404 text/plain，响应体：404：Not Found

5.建立网络服务器的步骤

* 1. 调用wifi库、wifimulti库、webserver库
  2. 建立WiFimulti和网络服务对象
  3. 启动串口通讯、添加WiFi、启动网络服务功能
  4. 不要忘记在串口监视器中显示IP地址

二、通过网络服务实现开发板的基本控制

1.代码

#include <ESP8266WiFi.h>        // 本程序使用 ESP8266WiFi库

#include <ESP8266WiFiMulti.h>   //  ESP8266WiFiMulti库

#include <ESP8266WebServer.h>   //  ESP8266WebServer库

ESP8266WiFiMulti wifiMulti;     // 建立ESP8266WiFiMulti对象,对象名称是 'wifiMulti'

ESP8266WebServer esp8266\_server(80);// 建立网络服务器对象，该对象用于响应HTTP请求。监听端口（80）

void **setup**(void){

**Serial**.begin(9600);   // 启动串口通讯

**pinMode**(LED\_BUILTIN, OUTPUT); //设置内置LED引脚为输出模式以便控制LED

  wifiMulti.addAP("ssid\_from\_AP\_1", "your\_password\_for\_AP\_1"); // 将需要连接的一系列WiFi ID和密码输入这里

  wifiMulti.addAP("ssid\_from\_AP\_2", "your\_password\_for\_AP\_2"); // ESP8266-NodeMCU再启动后会扫描当前网络

  wifiMulti.addAP("ssid\_from\_AP\_3", "your\_password\_for\_AP\_3"); // 环境查找是否有这里列出的WiFi ID。如果有

**Serial**.println("Connecting ...");                            // 则尝试使用此处存储的密码进行连接。

  int i = 0;

**while** (wifiMulti.run() != WL\_CONNECTED) {  // 此处的wifiMulti.run()是重点。通过wifiMulti.run()，NodeMCU将会在当前

**delay**(1000);                             // 环境中搜索addAP函数所存储的WiFi。如果搜到多个存储的WiFi那么NodeMCU

**Serial**.print(i++); **Serial**.print(' ');    // 将会连接信号最强的那一个WiFi信号。

  }                                          // 一旦连接WiFI成功，wifiMulti.run()将会返回“WL\_CONNECTED”。这也是

                                             // 此处while循环判断是否跳出循环的条件。

  // WiFi连接成功后将通过串口监视器输出连接成功信息

**Serial**.println('\n');

**Serial**.print("Connected to ");

**Serial**.println(WiFi.SSID());              // 通过串口监视器输出连接的WiFi名称

**Serial**.print("IP address:\t");

**Serial**.println(WiFi.localIP());           // 通过串口监视器输出ESP8266-NodeMCU的IP

  esp8266\_server.begin();                           // 启动网站服务

  esp8266\_server.on("/", HTTP\_GET, handleRoot);     // 设置服务器根目录即'/'的函数'handleRoot'

  esp8266\_server.on("/LED", HTTP\_POST, handleLED);  // 设置处理LED控制请求的函数'handleLED'

  esp8266\_server.onNotFound(handleNotFound);        // 设置处理404情况的函数'handleNotFound'

**Serial**.println("HTTP esp8266\_server started");//  告知用户ESP8266网络服务功能已经启动

}

void **loop**(void){

  esp8266\_server.handleClient();                     // 检查http服务器访问

}

/\*设置服务器根目录即'/'的函数'handleRoot'

  该函数的作用是每当有客户端访问NodeMCU服务器根目录时，

  NodeMCU都会向访问设备发送 HTTP 状态 200 (Ok) 这是send函数的第一个参数。

  同时NodeMCU还会向浏览器发送HTML代码，以下示例中send函数中第三个参数，

  也就是双引号中的内容就是NodeMCU发送的HTML代码。该代码可在网页中产生LED控制按钮。

  当用户按下按钮时，浏览器将会向NodeMCU的/LED页面发送HTTP请求，请求方式为POST。

  NodeMCU接收到此请求后将会执行handleLED函数内容\*/

void handleRoot() {

  esp8266\_server.send(200, "text/html", "<form action=\"/LED\" method=\"POST\"><input type=\"submit\" value=\"Toggle LED\"></form>");

}

//处理LED控制请求的函数'handleLED'

void handleLED() {

**digitalWrite**(LED\_BUILTIN,!**digitalRead**(LED\_BUILTIN));// 改变LED的点亮或者熄灭状态

  esp8266\_server.sendHeader("Location","/");          // 跳转回页面根目录

  esp8266\_server.send(303);                           // 发送Http相应代码303 跳转

}

// 设置处理404情况的函数'handleNotFound'

void handleNotFound(){

  esp8266\_server.send(404, "text/plain", "404: Not found"); // 发送 HTTP 状态 404 (未找到页面) 并向浏览器发送文字 "404: Not found"

}

*来自 <*[*http://www.taichi-maker.com/homepage/esp8266-nodemcu-iot/iot-c/esp8266-nodemcu-web-server/pin-control/*](http://www.taichi-maker.com/homepage/esp8266-nodemcu-iot/iot-c/esp8266-nodemcu-web-server/pin-control/)*>*

2.Timed out waiting for packet header 解决方法：

断开板子和数据线的连接，然后一直按住板子上的FLASH按钮（不要松开）重新插上USB线，下载程序，直到出现“Connecting....”时松开FLASH按钮

3.HTTP\_GET和HTTP\_POST两种请求服务器的方法

4.200和404的理解：两个状态信息，发送给服务器告诉信息的状态

5.基本步骤：

* 1. 建立WiFi和网络服务器
  2. 在网络服务器中设置handleroot和handleled两个程序（利用html）
  3. 理解两个函数的用法

三、通过网络服务将开发板的引脚状态显示在网页中

1.代码

#include <ESP8266WiFi.h>        // 本程序使用 ESP8266WiFi库

#include <ESP8266WiFiMulti.h>   //  ESP8266WiFiMulti库

#include <ESP8266WebServer.h>   //  ESP8266WebServer库

#define buttonPin D3            // 按钮引脚D3

ESP8266WiFiMulti wifiMulti;     // 建立ESP8266WiFiMulti对象,对象名称是'wifiMulti'

ESP8266WebServer esp8266\_server(80);// 建立网络服务器对象，该对象用于响应HTTP请求。监听端口（80）

bool pinState;  // 存储引脚状态用变量

void **setup**(){

**Serial**.begin(9600);   // 启动串口通讯

**pinMode**(buttonPin, INPUT\_PULLUP); // 将按键引脚设置为输入上拉模式

  wifiMulti.addAP("ssid\_from\_AP\_1", "your\_password\_for\_AP\_1"); // 将需要连接的一系列WiFi ID和密码输入这里

  wifiMulti.addAP("ssid\_from\_AP\_2", "your\_password\_for\_AP\_2"); // ESP8266-NodeMCU再启动后会扫描当前网络

  wifiMulti.addAP("ssid\_from\_AP\_3", "your\_password\_for\_AP\_3"); // 环境查找是否有这里列出的WiFi ID。如果有

**Serial**.println("Connecting ...");                            // 则尝试使用此处存储的密码进行连接。

  int i = 0;

**while** (wifiMulti.run() != WL\_CONNECTED) {  // 此处的wifiMulti.run()是重点。通过wifiMulti.run()，NodeMCU将会在当前

**delay**(1000);                             // 环境中搜索addAP函数所存储的WiFi。如果搜到多个存储的WiFi那么NodeMCU

**Serial**.print(i++); **Serial**.print(' ');    // 将会连接信号最强的那一个WiFi信号。

  }                                          // 一旦连接WiFI成功，wifiMulti.run()将会返回“WL\_CONNECTED”。这也是

                                             // 此处while循环判断是否跳出循环的条件。

  // WiFi连接成功后将通过串口监视器输出连接成功信息

**Serial**.println('\n');                     // WiFi连接成功后

**Serial**.print("Connected to ");            // NodeMCU将通过串口监视器输出。

**Serial**.println(WiFi.SSID());              // 连接的WiFI名称

**Serial**.print("IP address:\t");            // 以及

**Serial**.println(WiFi.localIP());           // NodeMCU的IP地址

  esp8266\_server.begin();                   // 启动网站服务

  esp8266\_server.on("/", handleRoot);       // 设置服务器根目录即'/'的函数'handleRoot'

  esp8266\_server.onNotFound(handleNotFound);// 设置处理404情况的函数'handleNotFound'

**Serial**.println("HTTP esp8266\_server started");//  告知用户ESP8266网络服务功能已经启动

}

void **loop**(){

  esp8266\_server.handleClient();     // 处理http服务器访问

  pinState = **digitalRead**(buttonPin); // 获取引脚状态

}

/\* 以下函数处理网站首页的访问请求。此函数为本示例程序重点1

详细讲解请参见太极创客网站《零基础入门学用物联网》

第3章-第2节“通过网络服务将开发板引脚状态显示在网页中”的说明讲解。\*/

void handleRoot() {

  String displayPinState;                   // 存储按键状态的字符串变量

**if**(pinState == HIGH){                     // 当按键引脚D3为高电平

    displayPinState = "Button State: HIGH"; // 字符串赋值高电平信息

  } **else** {                                  // 当按键引脚D3为低电平

    displayPinState = "Button State: LOW";  // 字符串赋值低电平信息

  }

  esp8266\_server.send(200, "text/plain", displayPinState);

                                            // 向浏览器发送按键状态信息

}

// 设置处理404情况的函数'handleNotFound'

void handleNotFound(){                                        // 当浏览器请求的网络资源无法在服务器找到时，

  esp8266\_server.send(404, "text/plain", "404: Not found");   // NodeMCU将调用此函数。

}

2.输入上拉和输入下拉的目的：

简单来说就是为了识别io口是低电平还是高电平

3.只有按下flash键不放才能让高电平变为低电平，而不是按一下

4.要查看是高电平还是低电平需要刷新界面，其原因是只有我们在，只有开发板向MCU请求首页的时候，才能触发“handroot”函数，调用后才会将首页信息变化

5.loop函数中，通过handClient函数不断访问服务器，以及宏定义的pinState获取引脚的状态

6.基本的handRooth和handNotFound函数的运用

**闪存服务系统**

 一、闪存文件系统的基本操作

1.4Mb空间的组成

* 1. 程序存储——1Mb
  2. 文件存储——<3Mb
  3. 系统文件——<1Mb

2.自动检测的闪存文件系统的大小

计算机生成了可选文字:
Conflqurunq亡上ashS上Ze。
Compressed298432bytesto217504．

3.程序文件大小以及占用空间也要观察

项 H 使 用 了 2 2 “ 字 节 ， 占 用 了 { 2 ， 程 序 存 睹 空 间 · 最 大 为 1 “ “ “ 字 节 · 
个 局 变 量 使 用 了 212Eg 字 节 ] 0 」 餉 动 态 内 有 ， 余 留 54E52 字 节 局 詡 变 量 最 大 为 0 字 

二、向SPIFFS中写入信息

1.代码

#include <FS.h>

String file\_name = "/taichi-maker/notes.txt"; //被读取的文件位置和名称

void **setup**() {

**Serial**.begin(9600);//启动串口通讯

**Serial**.println("");

**Serial**.println("SPIFFS format start");

  SPIFFS.format();    // 格式化SPIFFS（Serial Periphral Interface Flash File System）

**Serial**.println("SPIFFS format finish");

**if**(SPIFFS.begin()){ // 启动SPIFFS

**Serial**.println("SPIFFS Started.");

  } **else** {

**Serial**.println("SPIFFS Failed to Start.");

  }

  File dataFile = SPIFFS.open(file\_name, "w");// 建立File对象用于向SPIFFS中的file对象（即/notes.txt）写入信息

  dataFile.println("Hello IOT World.");       // 向dataFile写入字符串信息

  dataFile.close();                           // 完成文件写入后关闭文件

**Serial**.println("Finished Writing data to SPIFFS");

}

void **loop**() {

}

2.步骤

* 1. 建立被读取文件的位置与名称，启动串口通讯
  2. 格式化闪存文件系统
  3. 启动闪存文件系统
  4. 建立对象向SPIFFS中的在第一步建立的文件写入信息