# ESP8266-WIFI模块开发手册

本章将向大家介绍ESP8266-WIFI模块使用，通过本章的学习，让大家能快速上手51单片机应用开发。本章分为如下几部分内容：  
1.1 ESP8266-WIFI模块简介  
1.2 硬件设计

1.3 软件设计  
1.4 实验现象

## 1.1 ESP8266-WIFI模块简介

### 1.1.1 ESP8266简介

ESP8266是一个完整且成体系的Wi-Fi网络解决方案，能够搭载软件应用，或通过另一个应处理器卸载所有Wi-Fi网络功能。我们使用的ESP8266是串口型 WIFI，速度比较低，不能用来传输图像或者视频这些大容量的数据，主要应用于数据量传输比较少的场合，比如温度信息，一些传感器的开关量等。当然传输的数据量虽说少，但也能一次传输几千字节的数据，而且通信非常稳定，可以满足大多数应用。

ESP8266-WIFI模块支持 STA/AP/STA+AP 三种工作模式。

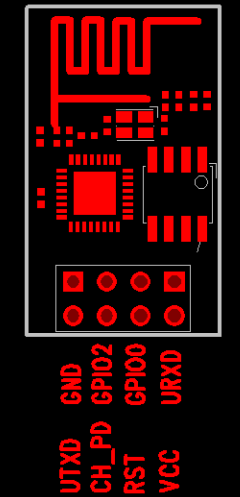
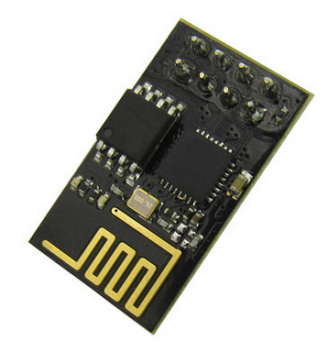
STA 模式：ESP8266 模块通过路由器连接互联网，手机或电脑通过互联网实现对设备的远程控制。

AP 模式：默认模式 PZ\_ESP8266 模块作为热点，实现手机或电脑直接与模块通信，实现局域网无线控制。

STA+AP 模式：两种模式的共存模式，即可以通过互联网控制可实现无缝切换，方便操作。

### 1.1.2 引脚说明

ESP8266模块非常小巧，模块通过 1个2\*4的间距为2.54的排针与外部连接，模块外观如图所示：



**管脚说明：**

**VCC(3V3)：**3.3V电源。

**RST**：ES8266复位管脚，可做外部硬件复位使用。

**CH\_PD(EN)**：使能管脚，高电平有效。

**UTXD(TX)**：串口发送管脚，与开发板上串口的RXD相连。

**URXD(RX)**：串口接收管脚，与开发板上串口的TXD相连。

**GPIO0(IO0)**：GPIO0 为高电平代表从 FLASH 启动， GPIO0 为低电平代表进入系统升级状态，此时可以经过串口升级内部固件。

**GPIO2(IO2)**：此管脚为ESP8266引出的一个IO口，这里我们不需要对此管脚操作。

**GND**：GND管脚，开发板上丝印已经标了。

其实我们不需要使用这么多管脚，只需要使用WIFI模块的串口UTXD、URXD管脚、电源VCC、GND管脚和CH\_PD(EN)管脚即可，其它的不用管。

## 1.2 ESP8266模块测试

### 1.2.1 硬件准备

首先我们需要准备以下设备：

①ESP8266模块一个

②电脑一台

③USB转TTL模块或普中51开发板（普中所有51开发板均含有USB转TTL模块）一块

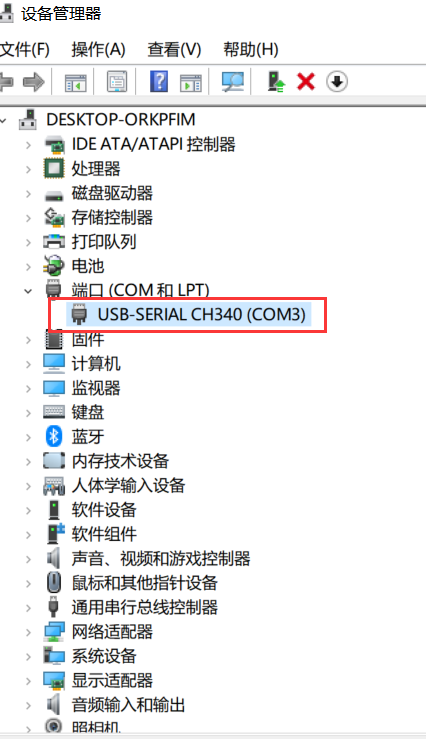
④USB数据线

### 1.2.2 使用USB转TTL串口测试

使用“\5--软件工具\串口调试助手\串口调试助手（丁丁）\sscom5.13.1.exe”可方便地测试ESP8266模块是否正常，测试步骤如下：

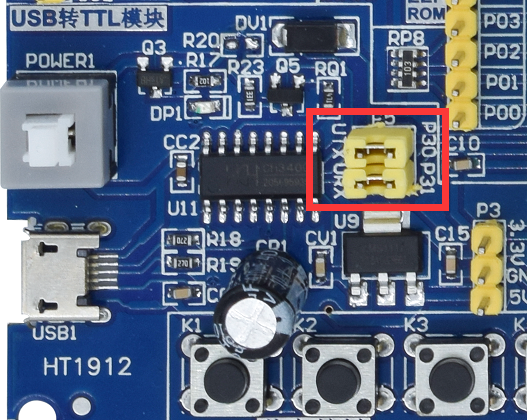
**（1）安装USB转TTL模块驱动**

使用一根USB数据线连接USB转TTL模块或普中开发板，如果使用普中开发板用户，可直接找到CH340驱动安装，这里默认已经安装好了CH340驱动，此时计算机就可识别开发板的CH340串口，可在设备管理器内查看到，如下图所示：



**（2）检测模块是否正常**

确保驱动安装成功后，将ESP8266模块与开发板上USB转TTL模块按照如下连接：（ESP8266模块-->普中开发板上USB转TTL模块（**此处以普中-2(A2)开发板为例，其它型号对应找到USB转TTL模块端子**））（**事先将开发板上USB转TTL模块端子的黄色跳线帽取下**）



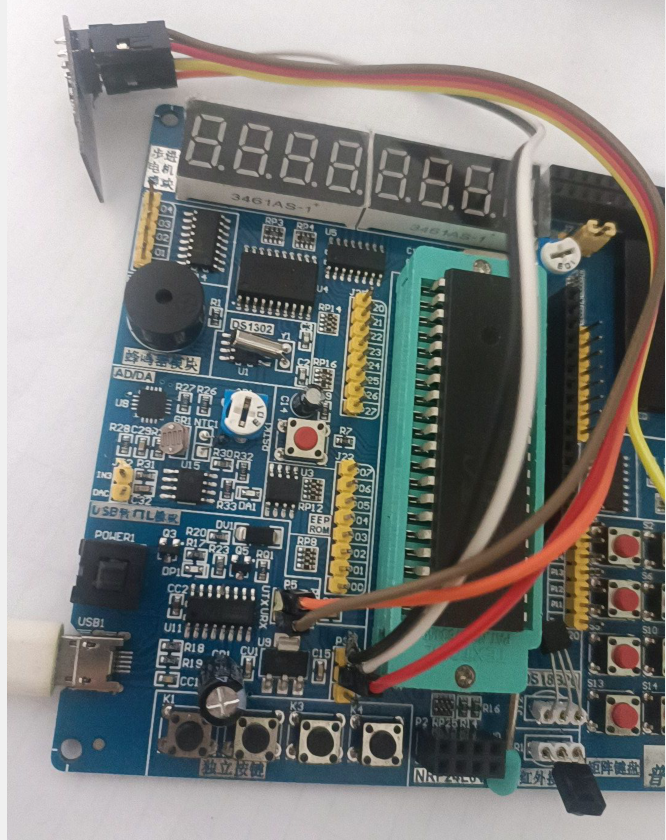
VCC(3V3)-->3.3V

CH\_PD(EN)-->3.3V

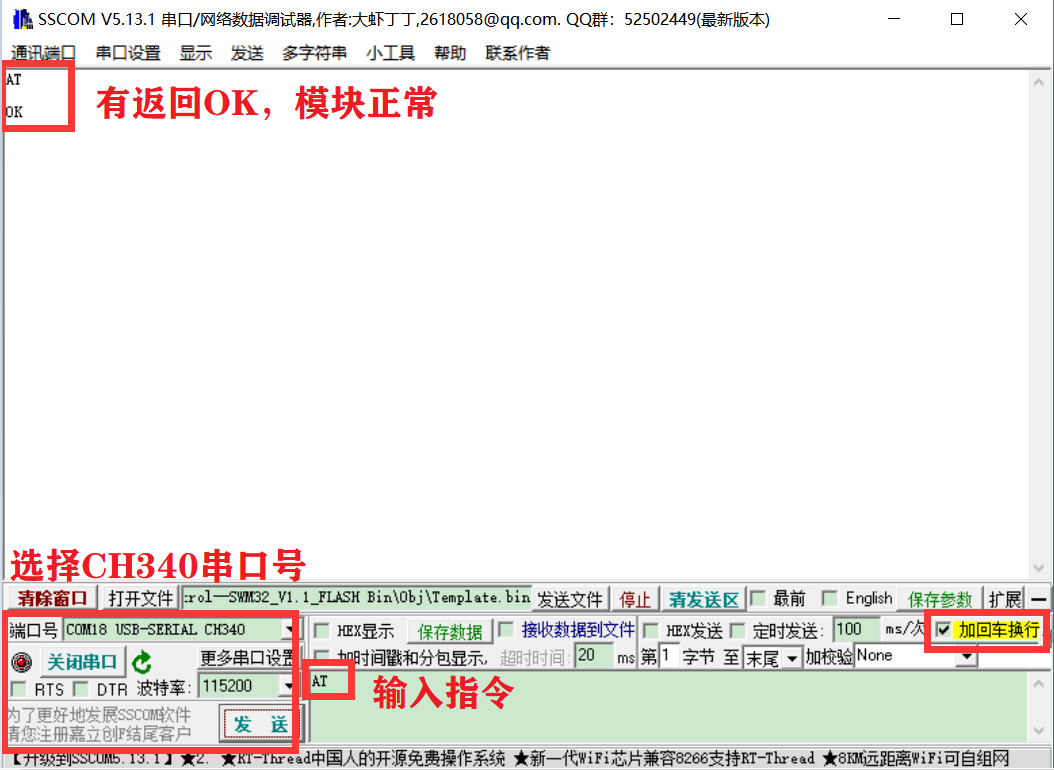
TXD-->URXD（开发板上USB转TTL模块）

RXD-->UTXD（开发板上USB转TTL模块）

GND-->GND



连接好模块和开发板之间的引脚，再用USB 数据线连接好电脑和开发板， 给开发板上电。打开串口调试助手 ，在端口菜单中选择步骤设备管理器中识别的CH340串口号，选择默认波特率“115200”，打开串口，勾选“发送新行”选项。在“字符串输入框”输入“AT”，点击“发送”，如果上方接收窗口有返回“OK”，那说明模块正常。如果没有返回“OK”，可以多次尝试，或检查前面步骤是否正常，**如若仍无该返回，请按照教程后面章节升级下固件**。升级后仍然无返回，模块可能存在异常。如下图所示：



**注意：如果模块没有返回OK，可以尝试9600波特率，还可以选择重新烧写固件测试，这个在后面小节会有介绍。测试完AT指令之后，可将前面取下来的黄色跳线帽还原，以方便后面单片机与WIFI模块通信和程序下载。**

### 1.2.3 通过检测模块WIFI信号

**（1）硬件准备**

该测试前需要准备以下一些设备：

①ESP8266模块一个

②能连接 WiFi 的手机、PAD 或笔记本等一部（我使用的手机）

③普中任意一款开发板（作为3.3V供电）

**（2）进入测试**

该测试方法非常简单，只是检测模块是否能发出 WiFi 信号。

①按照下列方式连接好电路（ESP8266模块-->普中51开发板）

VCC(3V3)-->3.3V

GND-->GND

CH\_PD(EN)-->3.3V

②使用手机扫描 WiFi 信号

给模块上电，打开手机的“WLAN”，扫描周围的热点，如果能扫描到模块发出的热点，如下图所示，那说明模块基本正常。该热点一般是“OPEN”型的，也就是无密码的，用户可连连看。模块断电后，该热点就消失。

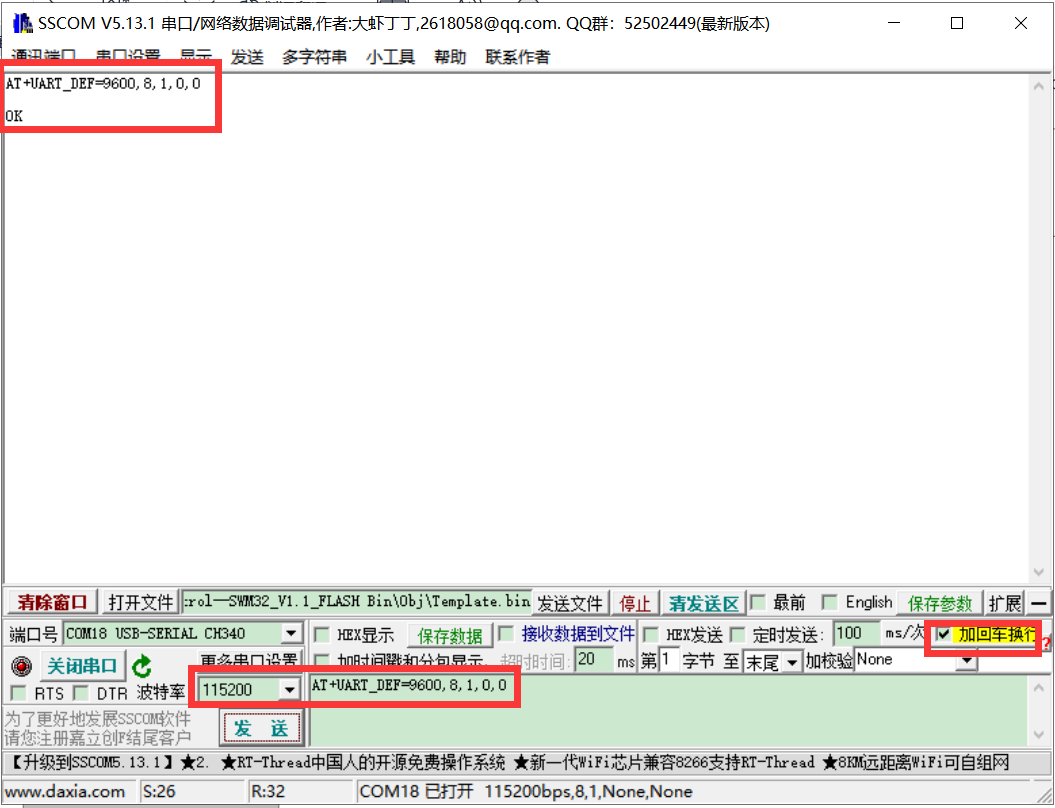
**注意：ESP8266模块产生的热点名称不一定如下图所示“B&T\_D23629”，用户可通过后面AT指令自定义模块WIFI热点名称**。



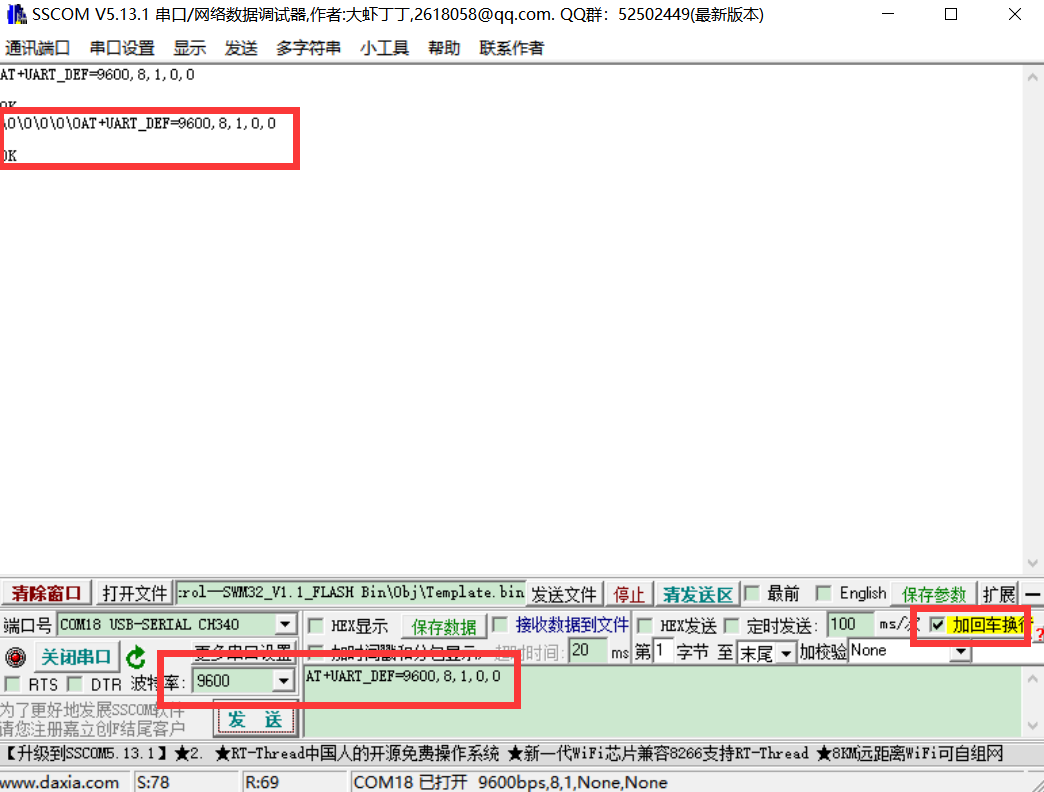
### 1.2.4 WIFI模块修改波特率值

前面将USB转TTL模块连接好WIFI模块后，使用串口调试助手发送AT指令后若没有OK返回，有可能ESP8266模块默认的通信波特率不为115200，比如9600，此时可在串口调试助手中选择相应的波特率，重新发送AT指令查看是否有返回。若波特率选择9600后有OK返回，说明默认通信波特率为9600，此时可不用修改波特率值。如果模块波特率是115200，则需要修改波特率，因为51单片机通信波特率无法设置到115200，因此就需要将WIFI模块的波特率改为9600。可通过发送命令：AT+UART\_DEF=9600,8,1,0,0

修改成功后会有OK返回。如下所示：



如再次发送该调命令，串口助手上则会显示\0\0等之类的乱码，这是因为此时WIFI模块波特率已设置为9600，而串口助手上依然还使用115200导致，所以可将串口助手波特率设置为9600，重新发送命令后输出如下：



至此，我们就将WIFI模块波特率修改成功，这样就可以使用51单片机串口与模块通信了，**注意：要将之前从USB转TTL模块上拔下来的两个黄色短接片重新插上去，方便后面下载程序。**

## 1.3 单片机控制ESP8266模块

### 1.3.1 硬件准备

本实验所需要的硬件资源如下：

①普中51开发板1个（**晶振需使用11.0592M**）

②ESP8266-WIFI模块 1 个

③USB线一条（用于供电和程序下载）

④带wifi功能的安卓手机一台

### 1.3.2 硬件连接

接线说明：DS18B20模块-->单片机IO

VCC-->5V

DATA-->P3.7

GND-->GND

ESP8266-WIFI模块-->单片机IO

TXD-->P3.0

RXD-->P3.1

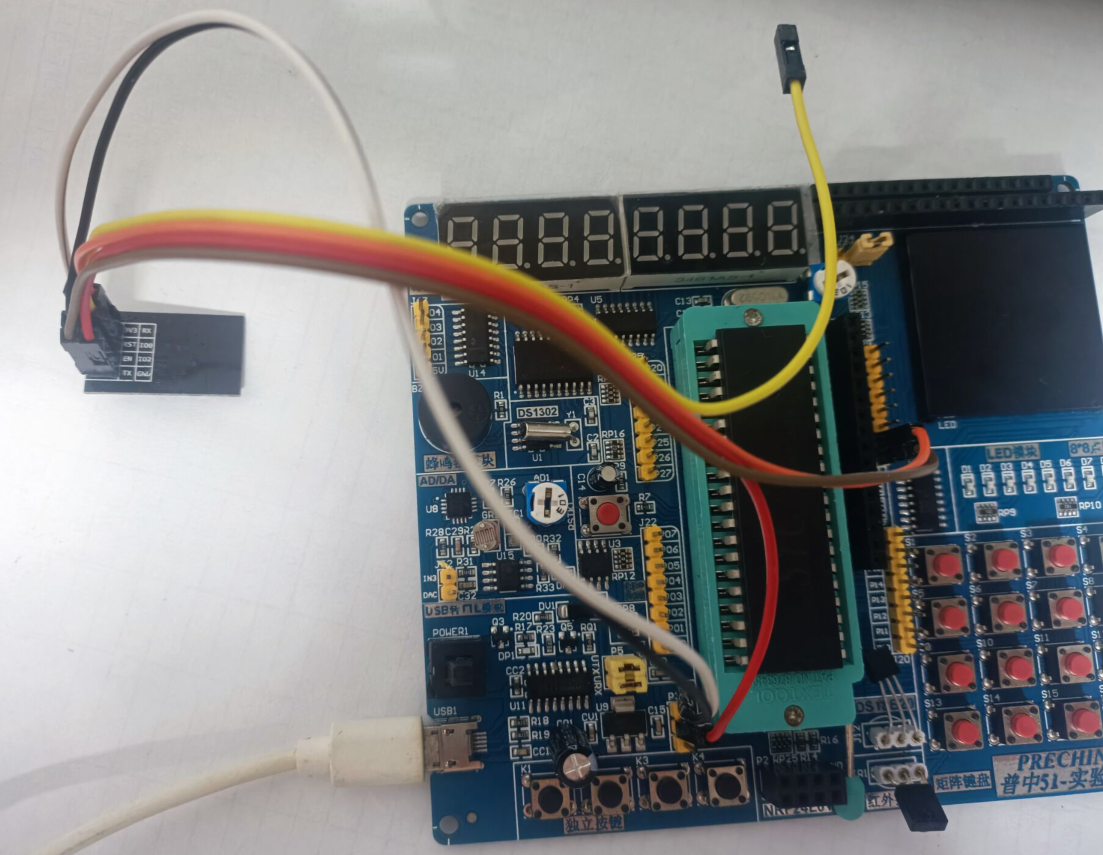
3V3-->3.3V

GND-->GND

EN-->3.3V

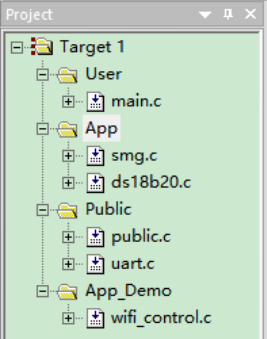
LED模块-->单片机IO

D1-->P2.0



### 1.3.3 软件设计

本实验程序是在开发板基础实验“LCD1602液晶显示实验”基础上修改，在App文件夹内新建了ds18b20、smg文件夹用于存放DS18B20温度传感器和数码管相关的驱动程序，在与App同级目录下新建了App\_Demo文件夹，在该文件夹内均新建了2个文件，一个.c源文件和一个.h头文件。将该文件添加到App工程组中，并添加对应头文件路径，这些操作如果不会请看基础实验视频教程，此处不再重复。添加好的工程如下所示：



smg.c是动态数码管实验相关代码，ds18b20.c是DS18B20温度传感器实验代码，uart.c是串口通信实验代码，wifi\_control.c是ESP8266驱动代码，我们仅对WIFI相关代码进行讲解，其它代码看基础实验教程。

#### 1.3.3.1 串口通信初始化函数

要让51单片机与WIFI模块进行通信，就需要对单片机的串口进行初始化配置，串口初始化设置在串口通信章节已经做过介绍，这里不多说，代码如下：

#define RELOAD\_COUNT 0xFA //宏定义波特率发生器的载入值 9600

void UART\_Init**(**void**)**

**{**

SCON**|=**0X50**;** //设置为工作方式1

TMOD**|=**0X20**;** //设置计数器工作方式2

PCON**=**0X80**;** //波特率加倍

TH1**=**RELOAD\_COUNT**;** //计数器初始值设置

TL1**=**TH1**;**

ES**=**0**;** //关闭接收中断

EA**=**1**;** //打开总中断

TR1**=**1**;** //打开计数器

// TI=1; //发送中断标记位，如果使用printf函数的必须设置

**}**

#### 1.3.3.2 WIFI模块初始化及数据命令发送函数

要使WIFI模块工作在AP模式下，则需要对模块进行AT指令设置，即通过51单片机的串口发送相关AT指令给模块，代码如下：

//ESP8266-WIFI模块工作模式初始化

void ESP8266\_ModeInit**(**void**)**

**{**

ESP8266\_SendCmd**(**"AT+CWMODE=2"**);**//设置路由器模式 1 staTIon模式 2 AP点 路由器模式 3 station+AP混合模式

ESP8266\_SendCmd**(**"AT+CWSAP=\"ESP8266\",\"123456\",11,0"**);** //设置WIFI热点名及密码

ESP8266\_SendCmd**(**"AT+CIPAP=\"192.168.4.1\""**);**

// ESP8266\_SendCmd("AT+RST");//重新启动wifi模块

// delay\_ms(2000);

ESP8266\_SendCmd**(**"AT+CIPMUX=1"**);** //开启多连接模式，允许多个各客户端接入

ESP8266\_SendCmd**(**"AT+CIPSERVER=1,8080"**);** //启动TCP/IP 端口为8080 实现基于网络控制

**}**

函数中涉及到的AT指令，大家可以通过“**\6--参考资料\ESP8266 AT指令集.pdf**”查询指令手册了解。函数调用了ESP8266\_SendCmd函数，该函数代码如下：

//ESP8266 WIFI发送AT指令

//pbuf：AT指令，字符串格式，如："AT"

void ESP8266\_SendCmd**(**u8 **\***pbuf**)**

**{**

**while(\***pbuf**!=**'\0'**)** //遇到空格跳出循环

**{**

UART\_SendData**(\***pbuf**);**

delay\_10us**(**5**);**

pbuf**++;**

**}**

delay\_10us**(**5**);**

UART\_SendData**(**'\r'**);**//回车

delay\_10us**(**5**);**

UART\_SendData**(**'\n'**);**//换行

delay\_ms**(**10**);**

**}**

函数内又调用了UART\_SendData函数，该函数即为串口发送字节函数，因为WIFI模块发送AT指令时需要发送一个换行结束符，所以函数最后会发送字符’\r’和’\n’。UART\_SendData函数代码如下：

void UART\_SendData**(**u8 dat**)**

**{**

ES**=**0**;** //关闭串口中断

TI**=**0**;** //清发送完毕中断请求标志位

SBUF**=**dat**;** //发送

**while(**TI**==**0**);** //等待发送完毕

TI**=**0**;** //清发送完毕中断请求标志位

ES**=**1**;** //允许串口中断

**}**

51单片机要发送温度数据到WIFI模块，还需要一个发送数据函数，代码如下：

//ESP8266 WIFI发送数据到APP

//pbuf：数据

void ESP8266\_SendData**(**u8 **\***pbuf**)**

**{**

ESP8266\_SendCmd**(**"AT+CIPSEND=0,7"**);**

**while(\***pbuf**!=**'\0'**)** //遇到空格跳出循环

**{**

UART\_SendData**(\***pbuf**);**

delay\_10us**(**5**);**

pbuf**++;**

**}**

UART\_SendData**(**'\n'**);**//换行

// delay\_ms(10);

**}**

函数内通过发送AT+CIPSEND=0,7指令，设定发送数据长度，然后发送数据到WIFI模块。

#### 1.3.3.3 温度数据获取及处理函数

该部分代码放在wifi控制函数内，代码如下：

//WIFI控制

void wifi\_control**(**void**)**

**{**

u16 i**=**0**;**

u8 temp\_buf**[**5**];**

int temp\_value**;**

u8 wifi\_send\_buf**[**7**];**

**while(**1**)**

**{**

i**++;**

**if(**i**%**50**==**0**)**//间隔一段时间读取温度值，间隔时间要大于温度传感器转换温度时间

temp\_value**=**ds18b20\_read\_temperture**()\***10**;**//保留温度值小数后一位

**if(**temp\_value**<**0**)**//负温度

**{**

temp\_value**=-**temp\_value**;**

temp\_buf**[**0**]=**0x40**;**//显示负号

wifi\_send\_buf**[**0**]=**'-'**;**

**}**

**else**

**{**

temp\_buf**[**0**]=**0x00**;**//不显示

wifi\_send\_buf**[**0**]=**'+'**;**

**}**

temp\_buf**[**1**]=**gsmg\_code**[**temp\_value**/**1000**];**//百位

temp\_buf**[**2**]=**gsmg\_code**[**temp\_value**%**1000**/**100**];**//十位

temp\_buf**[**3**]=**gsmg\_code**[**temp\_value**%**1000**%**100**/**10**]|**0x80**;**//个位+小数点

temp\_buf**[**4**]=**gsmg\_code**[**temp\_value**%**1000**%**100**%**10**];**//小数点后一位

smg\_display**(**temp\_buf**,**4**);**

**if(**i**%**200**==**0**)**//间隔一段时间将读取的温度发送到APP

**{**

wifi\_send\_buf**[**1**]=**temp\_value**/**1000**+**0x30**;**

wifi\_send\_buf**[**2**]=**temp\_value**%**1000**/**100**+**0x30**;**

wifi\_send\_buf**[**3**]=**temp\_value**%**1000**%**100**/**10**+**0x30**;**

wifi\_send\_buf**[**4**]=**'.'**;**

wifi\_send\_buf**[**5**]=**temp\_value**%**1000**%**100**%**10**+**0x30**;**

wifi\_send\_buf**[**6**]=**'\0'**;**

ESP8266\_SendData**(**wifi\_send\_buf**);**//通过串口发送温度数据到APP

**}**

**}**

**}**

#### 1.3.3.4 串口中断函数

前面我们已经初始化了串口，使能了串口接收中断，当手机端与WIFI模块建立起连接后，手机端发送数据，单片机串口即会进入中断函数接收数据，通过判断数据格式内容控制开发板上LED模块。代码如下：

//定义WIFI控制命令

#define LED1\_ON\_CMD '1'

#define LED1\_OFF\_CMD '2'

//串口中断服务函数

//接收手机APP发送的信号后控制板载资源

void UART\_IRQn**()** interrupt 4

**{**

static u8 i**=**0**;**

**if(**RI**)**

**{**

RI**=**0**;**

UART\_RX\_BUF**[**i**]=**SBUF**;**//读取接收到的数据

**if(**UART\_RX\_BUF**[**0**]==**'+'**)**i**++;**

**else** i**=**0**;**

**if(**i**==**10**)**

**{**

i**=**0**;**

//WIFI控制

**if(**UART\_RX\_BUF**[**9**]==**LED1\_ON\_CMD**)**

LED1**=**0**;**

**else** **if(**UART\_RX\_BUF**[**9**]==**LED1\_OFF\_CMD**)**

LED1**=**1**;**

**}**

**}**

**}**

一般APP发送出来的数据格式内容为：**+IPD,ID号,发送数据长度:数据**

**当手机连接热点成功后，其中+IPD,ID号就是固定的，用户可根据需要发送多少数据，然后长度就可以确定。这样在串口接收中断中就可以按照这个格式进行解析，我们主要关注的是“数据”那块。**

例如本章实验中控制LED1开的指令：+IPD,0,1:1

从中断函数中可以知道手机APP端发送的数据内容，如果大家想要自己设计APP，那么控制可是也可以参考我们这里的。

#### 1.3.3.5 主函数

最后我们看下主函数，看下如何将各个功能模块函数组合的，代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

实验名称：ESP8266-WIFI模块实验

接线说明：DS18B20模块-->单片机IO

VCC-->5V

DATA-->P3.7

GND-->GND

ESP8266-WIFI模块-->单片机IO

TXD-->P3.0

RXD-->P3.1

3V3-->3.3V

GND-->GND

EN-->3.3V

LED模块-->单片机IO

D1-->P2.0

实验现象：下载程序后，插上DS18B20温度传感器，数码管显示检测的温度值，连接

WIFI模块热点，打开手机APP，可控制开发板D1指示灯，同时采集的温度数据上传到手机

APP显示。

注意事项：注意模块默认波特率为115200，需将其修改为9600，修改波特率命令：AT+UART\_DEF=9600,8,1,0,0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "public.h"

#include "wifi\_control.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函 数 名 : main

\* 函数功能 : 主函数

\* 输 入 : 无

\* 输 出 : 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main**()**

**{**

wifi\_control\_init**();**

**while(**1**)**

**{**

wifi\_control**();**

**}**

**}**

#### 1.3.3.6 实验现象

首先将硬件连接好，程序下载成功后（**若出现下载失败，请先将ESP8266模块串口线拔掉，防止对开发板上CH340串口干扰**），实验现象如下：

首先需要给手机安装一个网络调试助手APP，在模块资料“\5--软件工具\手机端网络调试助手\TCP.apk”中我们已经提供，在安卓手机上安装该APP即可。

然后用手机连接前面给模块设置的WIFI热点：PRECHIN，无需密码。如下所示：



打开安装好的APP，如下所示：



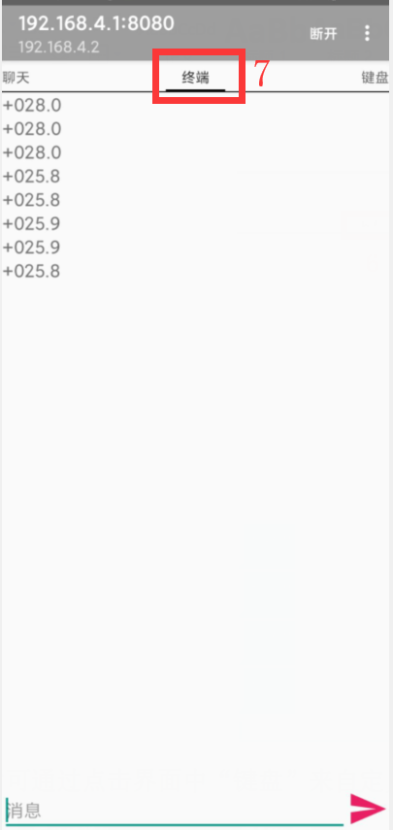
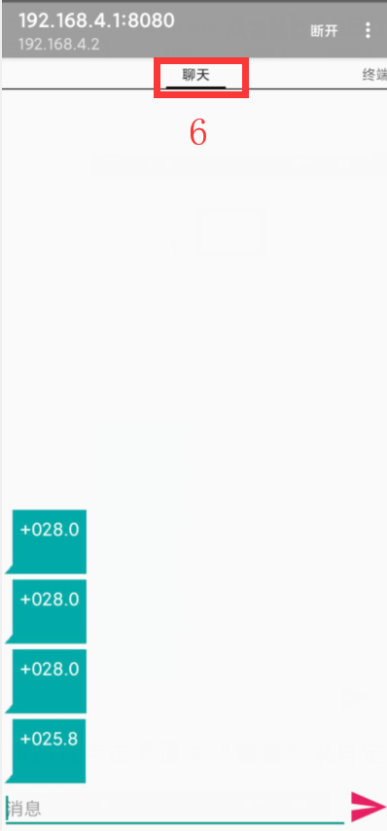
首先要新建一个主机，IP地址设置为：192.168.4.1，端口设置为：8080。操作如下：



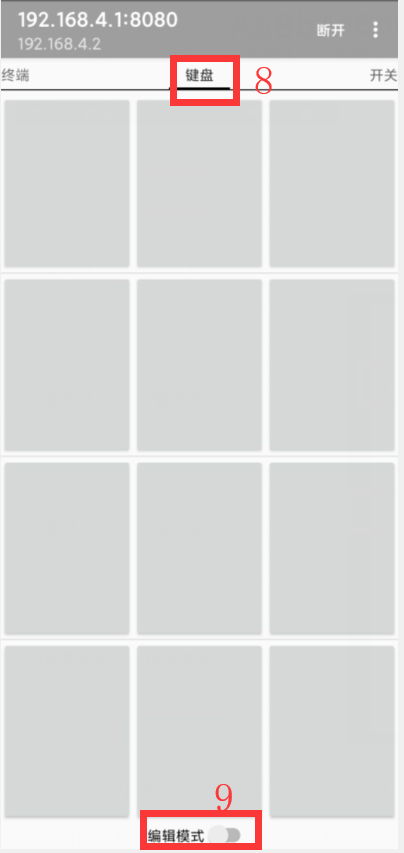




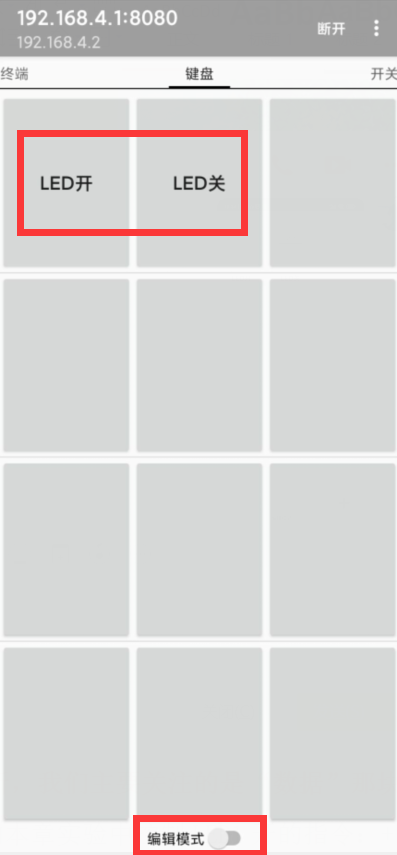
手机连接成功后，即可进入显示界面，此时可通过点击界面中“聊天”、“终端”查看单片机发送的温度数据，如下所示：



然后可通过点击界面中“键盘”来自定义APP发送的命令数据，如下所示：



定义完成后即可通过“键盘”中按钮发送相应命令，通过WIFI模块传输至单片机，单片机解析后即可控制相应板载外设。比如现在定义了“LED开”命令是字符1，“LED关”命令是字符2，因此可控制板载D1指示灯的亮和灭。



一般APP发送出来的数据格式内容为：**+IPD,ID号,发送数据长度:数据**

**当手机连接热点成功后，其中+IPD,ID号就是固定的，用户可根据需要发送多少数据，然后长度就可以确定。这样在串口接收中断中就可以按照这个格式进行解析，我们主要关注的是“数据”那块。**

例如本章实验中控制LED1开的指令：+IPD,0,1:1。

## 1.4 ESP8266模块固件升级

ESP8266 WiFi 模块升级固件的方法有两种：第一种是本地串口烧录，第二种是云端升级。笔者着重介绍第一种方法，第二种方法用户可以参考第三方资料的相关手册。

### 1.4.1 本地串口烧录

使用“本地串口烧录”方案刷新固件的步骤如下：

（1）断开模块电源，按照下列方式连接好电路。GPIO0（IO0）引脚拉低进入 FLASH 烧写模式。（事先将开发板上USB转TTL模块端子的黄色跳线帽取下）（ESP8266模块-->普中51开发板）

VCC-->3.3V

CH\_PD(EN)-->3.3V

TXD-->URXD（开发板上USB转TTL模块）

RXD-->UTXD（开发板上USB转TTL模块）

GND-->GND

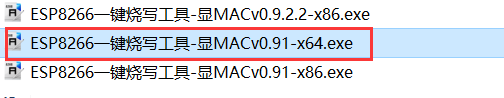
IO0-->GND

（2）上网下载固件。网上有很多技术论坛都有制作 ESP8266 芯片固件，用户可以自行上网搜索，根据自己的需求下载中意的固件。笔者要演示的固件是从“物联世界”论坛下载的安信可官方的 0.95 版本的固件，存放于“**\7--ESP8266固件及烧写工具\ESP8266固件\BAT\_AT\_V1.7.1.0\_1M.bin**”。

（3）选择符合电脑系统的“ESP8266 一键烧写工具”。在电脑上右键单击“计算机”图标选择“属性”查看电脑系统类型。如下：



打开“**\7--ESP8266固件及烧写工具\ESP8266一键烧写工具\ESP8266一键烧写工具**”，根据电脑系统类型选择“ESP8266一键烧写工具”，打开烧写工具，如下图。





（4）烧录固件。点击“ESP8266一键烧写工具”的“配置”选项卡，点击第二栏的图标，选择固件的路径（“**\7--ESP8266固件及烧写工具\ESP8266固件\BAT\_AT\_V1.7.1.0\_1M.bin**”），仅将第二栏打“x”，如下图所示：



按照前面接线方式，给模块上电。点击“操作”选项卡，选择连接到模块的串口，点击“一键烧写(F)”即可进行烧写固件。烧写过程中有进度条，而且还能读取模块的物理地址，如下图所示。



烧写完毕后，工具左下角会出现一个图标，如下图所示。



固件烧写完成后，可按照前面“单片机控制ESP8266”小节实验完成。

## 课后作业