

PZ-ESP8266 WIFI 模块开发手册

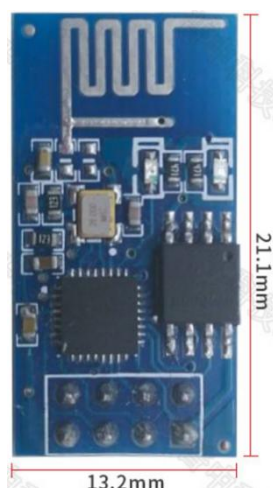
1 PZ-ESP8266 模块介绍与测试

1.1 ESP8266 简介

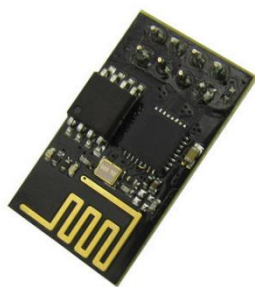
ESP8266 是一个完整且成体系的 Wi-Fi 网络解决方案，能够搭载软件应用，或通过另一个应处理器卸载所有 Wi-Fi 网络功能。我们使用的 ESP8266 是串口型 WIFI，速度比较低，不能用来传输图像或者视频这些大容量的数据，主要应用于数据量传输比较少的场合，比如温度信息，一些传感器的开关量等。当然传输的数据量虽说少，但也能一次传输几千字节的数据，而且通信非常稳定，可以满足大多数应用。

1.2 PZ-ESP8266 模块介绍

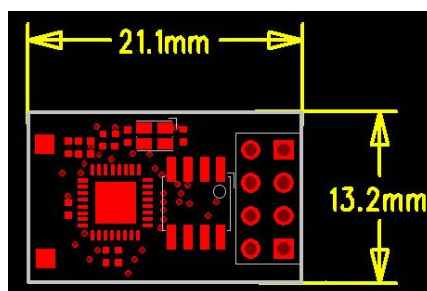
安信可推出的 ESP8266-WIFI 模块有很多，它们的使用方法都大同小异，我们普中推出了 1 款 WIFI 模块，如图：



与通用的 WIFI 模块一样使用，这里我们就以通用的 WIFI 模块介绍。如下图所示：

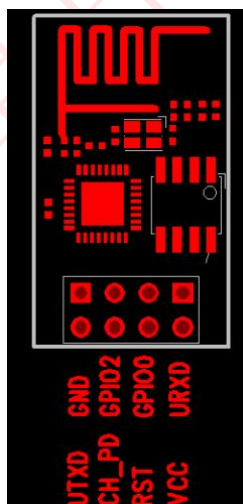


上图的 WIFI 模块尺寸图如下图所示：



如果要将此模块设计到自己产品内，可能需要参考这个尺寸值。

从 WIFI 模块实物图中可以看到，WIFI 模块提供了一个 2*4 的外接管脚，让我们连接到自己的电路中控制，这 8 个管脚两两间距是 2.54mm。管脚功能定义如下：



VCC: 3.3V 电源，开发板上丝印已经标了。

RST: ES8266 复位管脚，可做外部硬件复位使用。

CH_PD: 使能管脚，高电平有效。

UTXD: 串口发送管脚，与开发板上串口的 RXD 相连。

URXD: 串口接收管脚，与开发板上串口的 TXD 相连。

GPIO0: 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，这里我们不需要对此管脚操作。

GPI02: 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 管脚即可，其他的不用管。

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。

1.3 PZ-ESP8266 模块测试

下面开始对模块进行测试。

1.3.1 硬件准备

首先我们需要准备以下设备：

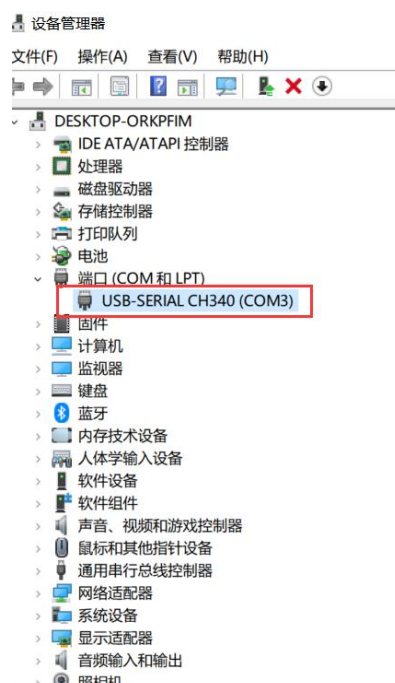
- ①PZ-ESP8266 模块一个
- ②电脑一台
- ③USB 转 TTL 模块或普中开发板（普中所有开发板均含有 USB 转 TTL 模块）一块
- ④USB 数据线

1.3.2 使用 USB 转 TTL 串口测试

使用“\3--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 PZ-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 PZ-ESP8266 模块与开发板上 USB 转 TTL 模块按照如下连接：（PZ-ESP8266 模块-->普中开发板上 USB 转 TTL 模块（**此处以普中-5 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-->3.3V

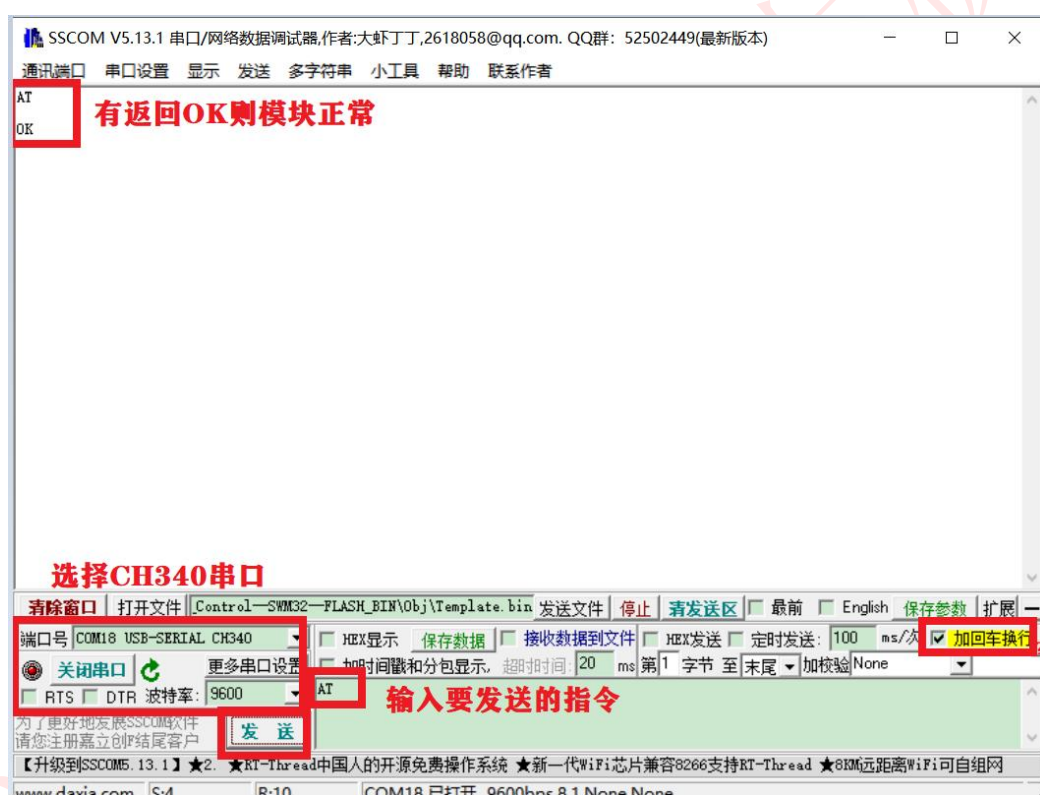
CH_PD-->3.3V

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“9600”，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。

1.3.3 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：

- ①PZ-ESP8266 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（PZ-ESP8266 模块-->普中开发板）

VCC-->3.3V

GND-->GND

CH_PD-->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：PZ-ESP8266 模块产生的热点名称不一定如下图所示

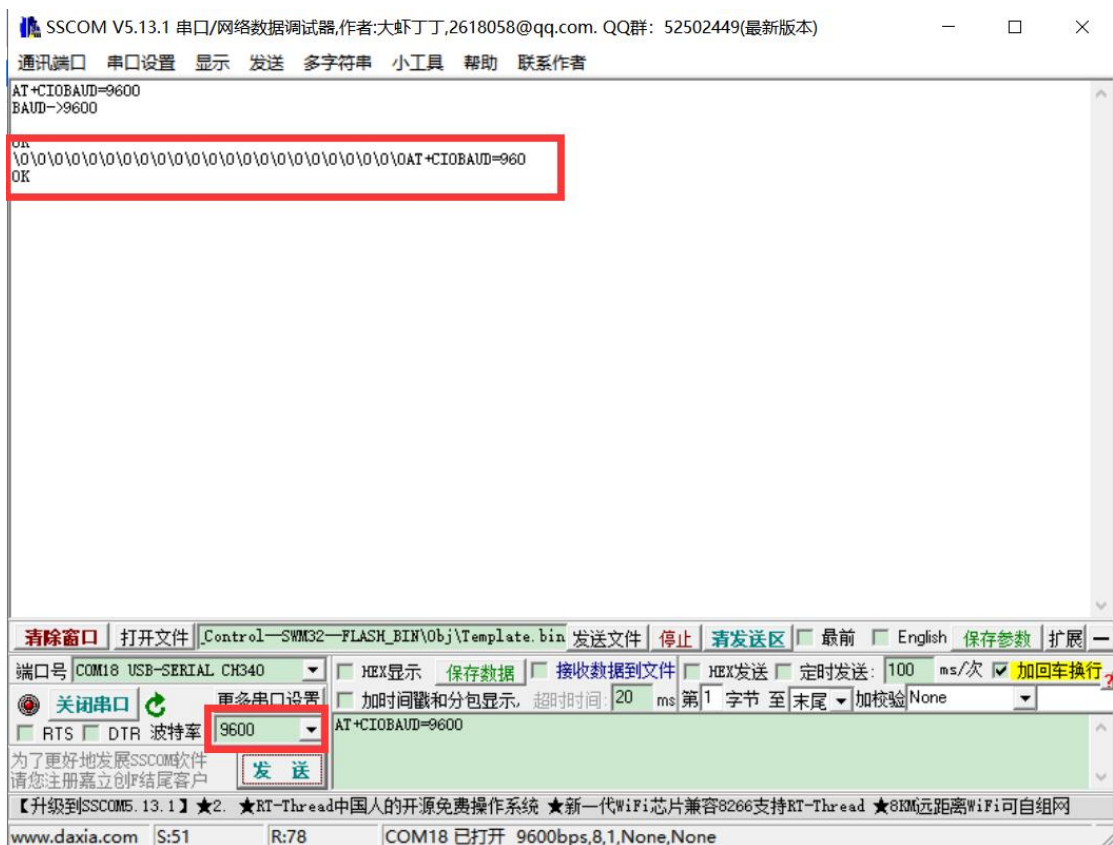
“AI-THINKER_28FCA3”，用户可通过后面 AT 指令自定义模块 WIFI 热点名称。



1.3.4 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 9600，比如 115200，此时可在串口调试助手选择相应的波特率，重新发送 AT 指令查看是否有返回。若波特率选择 115200 后有 OK 返回，说明默认通信波特率为 115200，而 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：



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。**

2 串口 AT 指令

2.1 概述

指令集主要分为：基础 AT 命令、Wifi 功能 AT 命令、TCP/IP 工具箱 AT 命令等。

2.2 指令说明

每条指令可以有细分四种命令形式。

命令形式	AT 指令格式	说明
测试命令	AT+<x>=?	该命令用于查询设置命令或内部程序设置的参数以及其取值范围。
查询命令	AT+<x>?	该命令用于返回参数的当前值。
设置命令	AT+<x>=<...>	该命令用于设置用户自定义的参数值。
执行命令	AT+<x>	该命令用于执行受模块内部程序控制的变参数不可变的功能。

- 注意：1. 不是每条 AT 指令都具备上述 4 类命令
2. []内数据为缺省值，不必填写或可能不显示
3. 使用双引号表示字符串数据
4. 默认波特率 960000

2.3 基础 AT 指令

2.3.1 基础 AT 指令一览表

命令	说明
AT	测试 AT 启动
AT+RST	重启模块
AT+GMR	查看版本信息

2.3.2 基础 AT 指令详解

1、AT：测试 AT 启动

AT	说明
执行指令 AT	响应 OK

2、AT+RST：重启模块

AT+RST	说明
执行指令 AT+RST	响应 OK

3、AT+GMR：查看版本信息

AT+GMR	说明
执行指令 AT+GMR	响应 <number> OK
	参数说明 < number >8 位版本号

2.4 Wifi 功能 AT 指令

2.4.1 Wifi 功能 AT 指令一览表

命令	说明
AT+CWMODE	选择 WIFI 应用模式
AT+CWJAP	加入 AP
AT+CWLAP	列出当前可用 AP
AT+CWQAP	退出与 AP 的连接
AT+CWSAP	设置 AP 模式下的参数
AT+CWLIF	查看已接入设备的 IP

2.4.2 Wifi 功能 AT 指令详解

1、AT+CWMODE：选择 WIFI 应用模式

AT+CWMODE	说明
测试指令 AT+CWMODE=?	响应 +CWMODE:(<mode>取值列表) OK 参数说明 见设置命令
查询命令 AT+CWMODE?	响应 返回当前模块的模式 +CWMODE:<mode> OK 参数说明 见设置指令
设置指令 AT+CWMODE=<mode>	响应 OK 参数说明 <mode>1 Station 模式 2 AP 模式 3 AP 兼 Station 模式

2、AT+CWJAP：加入 AP

AT+CWJAP	说明
查询命令 AT+CWJAP?	响应 返回当前选择的 AP + CWJAP:<ssid> OK 参数说明 见设置命令
设置指令 AT+CWJAP = <ssid>,<pwd>	响应 OK ERROR 参数说明 <ssid>字符串参数，接入点名称 <pwd>字符串参数，密码最长 64 字节 ASCII

3、AT+CWLAP：列出当前可用 AP

AT+CWLAP	说明
执行指令 AT+CWLAP	响应 终端返回 AP 列表 + CWLAP: <ecn>,<ssid>,<rsssi> OK ERROR 参数说明 <ecn> 0 OPEN 1 WEP 2 WPA_PSK 3 WPA2_PSK 4 WPA_WPA2_PSK <ssid>字符串参数，接入点名称 <rsssi>信号强度

4、AT+CWQAP：退出与 AP 的连接

AT+CWQAP	说明
测试指令 AT+CWQAP=?	响应 OK
执行指令 AT+CWQAP	响应 OK

5、AT+CWSAP：设置 AP 模式下的参数

AT+CWSAP	说明
查询命令 AT+CWSAP?	响应 返回当前 AP 参数 +CWSAP:<ssid>,<pwd>,<chl>,<ecn> 参数说明 见设置指令
设置指令 AT+CWSAP= <ssid>,<pwd>,<chl>,<ecn>	响应 OK ERROR 参数说明 指令只有在 AP 模式开启后有效 <ssid>字符串参数, 接入点名称 <pwd>字符串参数, 密码最长 64 字节 ASCII <chl>通道号 <ecn>0 OPEN 1 WEP 2 WPA_PSK 3 WPA2_PSK 4 WPA_WPA2_PSK

6、AT+CWLIF: 查看已接入设备的 IP

AT+CWLIF	说明
执行指令 AT+CWLIF	响应 <ip addr> OK 参数说明 <ip addr> 已接入设备的 IP 地址

2.5 TCP/IP 工具箱 AT 指令

2.5.1 TCP/IP 工具箱 AT 指令一览表

命令	说明
AT+CIPSTATUS	获得连接状态
AT+CIPSTART	建立 TCP 连接或注册 UDP 端口号
AT+CIPSEND	发送数据
AT+CIPCLOSE	关闭 TCP 或 UDP
AT+CIFSR	获取本地 IP 地址
AT+CIPMUX	启动多连接
AT+CIPSERVER	配置为服务器
AT+CIPMODE	设置模块传输模式
AT+CIPSTO	设置服务器超时时间

2.5.2 TCP/IP 工具箱 AT 指令详解

1、AT+CIPSTATUS：获得连接状态

AT+CIPSTATUS	说明
测试指令 AT+CIPSTATUS=?	响应 OK
执行指令 AT+CIPSTATUS	响应 返回当前模块的连接状态和连接参数 STATUS:<stat> + CIPSTATUS:<id>,<type>,<addr>,<port>,<tetype> OK
	参数说明 <id>连接的 id 号 0-4 <type>字符串参数，类型 TCP 或 UDP <addr>字符串参数，IP 地址 <port>端口号 <tetype> 0: 本模块做 client 的连接 1: 本模块做 server 的连接

2、AT+CIPSTART：建立 TCP 连接或注册 UDP 端口号

AT+CIPSTART	说明
测试指令 AT+CIPSTART=?	响应 1) 设置 AT+CIPMUX=0 +CIPSTART:(<type>取值列表),(<IP address>范围),(<port>范围) +CIPSTART:(<type>取值列表),(<domain name>范围),(<port>范围) OK 2) 设置 AT+CIPMUX=1 +CIPSTART:(id),(<type>取值列表),(<IP address>范围),(<port>范围) +CIPSTART: (id), (<type>取值列表),(<domain name>范围),(<port>范围) 参数说明 见设置命令
设置命令 1)单路连接 (+CIPMUX=0) AT+CIPSTART= <type>,<addr>,<port>	响应 如果格式正确且连接成功，返回 OK 否则返回 ERROR 如果连接已经存在，返回

1)多路连接 (+CIPMUX=1) AT+CIPSTART= <id><type>,<addr>, <port>	ALREAY CONNECT
	参数说明 <id>连接的 id 号 0-4 <type> 字符串参数, 表明连接类型 " TCP" 建立 tcp 连接 " UDP" 建立 UDP 连接 <addr> 字符串参数, 远程服务器 IP 地址 <port> 远程服务器端口号

3、AT+CIPSEND: 发送数据

AT+CIPSEND	说明
测试指令 AT+CIPSEND=?	响应 OK 参数说明 见设置命令
设置指令 1) 单路连接时 (+CIPMUX=0) AT+CIPSEND=<length> 2) 多路连接时 (+CIPMUX=1) AT+CIPSEND= <id>,<length>	响应 发送指定长度的数据。收到此命令后先换行返回">", 然后开始接收串口数据, 当数据长度满 length 时发送数据。 如果未建立连接或连接被断开, 返回 ERROR 如果数据发送成功, 返回 SEND OK 参数说明 <id>需要用于传输连接的 id 号 <length>数字参数, 表明发送数据的长度, 最大长度为 2048
执行指令 AT+CIPSEND	响应 收到此命令后先换行返回">" 然后就进入了透传模式, 每包数据以 20ms 间隔区分, 每包最大 2048 字节。 当输入单独一包"+++" 返回指令模式。 该指令必须在开启透传模式以及单连接模式下使用

4、AT+CIPCLOSE: 关闭 TCP 或 UDP

AT+CIPCLOSE	说明
测试指令 AT+CIPCLOSE=?	响应 OK
多路连接时 AT+CIPCLOSE=<id>	如果输入正确, 返回 OK 如果没有该连接则, 返回 Link is not 参数说明 <id>需要关闭的连接 id

	当 id=5 时关闭所有连接（开启 server 后 id=5 无效）
执行指令 单路连接时 AT+CIPCLOSE	响应 如果输入正确，返回 OK 如果没有连接则，返回 ERROR 当没有连接时返回状态打印 unlink

5、AT+CIFSR：获取本地 IP 地址

AT+CIFSR	说明
测试指令 AT+CIFSR=?	响应 OK
执行命令 AT+ CIFSR	响应 + CIFSR:<AP IP address> <STA IP address> OK ERROR
	参数说明 <AP IP address> 本机的 AP 模式的 IP 地址 <STA IP address> 本机的 STA 模式的 IP 地址
参考	说明 对应应用模式开通了才有其 IP 地址

6、AT+CIPMUX：启动多连接

AT+CIPMUX	说明
查询命令 AT+ CIPMUX?	响应 + CIPMUX:<mode> OK
	参数说明 见设置指令
设置指令 AT+ CIPMUX=<mode>	响应 OK 如果已经处于连接状态则，返回 Link is builded
	参数说明 <mode>0 单路连接模式 1 多路连接模式
参考	说明 只有当连接都断开后才能更改，如果开启过 server 需要重启模块

7、AT+CIPSERVER：配置为服务器

设置指令 AT+ CIPSERVER= <mode>[,<port>]	响应 OK 关闭 server 需要重启
	参数说明 <mode>0 关闭 server 模式 1 开启 server 模式 <port>端口号, 缺省值为 333
参考	说明 开启 server 后自动建立 server 监听 当有 client 接入会自动按顺序占用一个连接 AT+ CIPMUX=1 时才能开启服务器

8、AT+CIPMOD：设置模块传输模式

AT+CIPMOD	说明
查询命令 AT+ CIPMODE?	响应 + CIPMODE:<mode> OK
	参数说明 见设置指令
设置指令 AT+CIPMODE=<mode>	响应 OK 如果已经处于连接状态则, 返回 Link is builded
	参数说明 <mode>0 非透传模式 1 透传模式

9、AT+CIPSTO：设置服务器超时时间

AT+CIPSTO	说明
查询命令 AT+CIPSTO?	响应 + CIPSTO:<time> OK
	参数说明 见设置指令
设置指令 AT+CIPSTO=<time>	响应 OK
	参数说明 < time>0~28800 服务器超时时间, 单位为 s

2.6 其他指令

1、+IPD：接收到网络数据

+IPD	说明
参考 1) 单路连接时 (+CIPMUX=0) +IPD,<len>:<data> 2) 多路连接时 (+CIPMUX=1) +IPD,<id>,<len>:<data>	说明 此指令是模块发出指令，当模块接收到网络数据时向串口发送+IPD 和数据 <id>收到连接的 id 号 <len>数据长度 <data>收到的数据 此提示在指令状态下有效

2、AT+CIOBAUD：设置波特率

AT+CIOBAUD	说明
测试指令 AT+CIOBAUD=?	响应 +CIOBAUD:<baud_range> OK 参数说明 <baud_range>波特率取值范围
查询命令 AT+CIOBAUD?	响应 +CIOBAUD:<baud> OK 参数说明 <baud>当前的波特率
设置指令 AT+CIOBAUD=<baud>	响应 BAUD-><baud> OK 参数说明 <baud>要设置的波特率

2.7 附带说明

上述 AT 指令集是 PZ-ESP8266 模块的现有的常用 AT 指令集，模块的具有 AT 指令集因固件不同而不同，更新版本的固件一般包含更丰富的 AT 指令，并且一般新固件都兼容就固件。用户若想刷新版固件，可参考后面章节。

3 单片机控制 PZ-ESP8266 模块

PZ-ESP8266 模块支持 TTL 串口通讯标准，可直接与单片机的串口引脚连接通讯，非常方便使用单片机系统来控制。本章节以普中 51 开发板为例说明如何使用 STM32 来控制 PZ-ESP8266 模块。

3.1 硬件连接

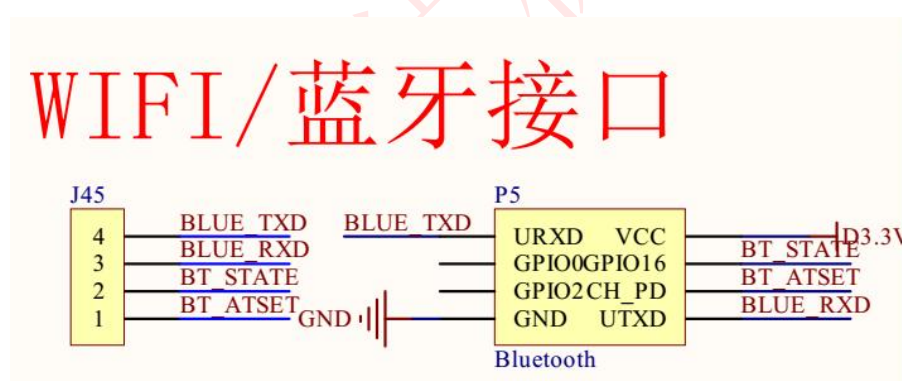
3.1.1 硬件准备

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**）
- ②PZ-ESP8266 WIFI 模块 1 个
- ③USB 线一条（用于供电和模块与电脑串口调试助手通信）
- ④安卓手机一台

3.1.2 模块与开发板连接

PZ-ESP8266 模块可通过杜邦线将模块管脚与单片机的 IO 口连接即可。对于普中 5-普中 7 型号产品板载了一个 WIFI/蓝牙模块接口，可将模块插入该接口后连接几根通信线即可。电路如下：



从上图可以看出，该电路是独立的，P5 接口是 WIFI/蓝牙模块的接口，可以将 PZ-ESP8266-WIFI 模块/PZ-HC05 蓝牙模块接在此处，默认已经将模块的电源管脚接好，而 WIFI/蓝牙模块芯片的控制管脚接至 J45 端子上，方便于我们自行将控制管脚与单片机连接。前面我们说了 ESP8266-WIFI 模块一般只需要 UTXD、URXD、CH_PD、RST 控制管脚即可，RST 对应原理图的 GPIO16，也就是 J45 端子上的 BT_STATE，CH_PD 对应 J45 端子的 BT_ATSET。

模块与单片机的 IO 口连接关系如下：（PZ-ESP8266 模块-->51 开发板）

SET-->3.3V

TXD-->P30

RXD-->P31

对于普中 2-普中 4 型号产品是没有专门的 WIFI/蓝牙接口，全部需要导线连接。

PZ-ESP8266 模块与单片机的 IO 口连接关系如下：（PZ-ESP8266 模块-->51 开发板）

VCC-->3.3V

CH_PD-->3.3V

TXD-->P30

RXD-->P31

GND-->GND

注意：使用普中 5-普中 7 开发板连接模块时，模块插入 WIFI 接口处请注意方向，当然也可直接通过导线连接，无需插入接口。

3.2 软件设计

我们打开对应的例程，该程序实现的功能是：通过手机 APP 控制开发板上的 LED 灯，同时开发板上的 DS18B20 温度传感器采集的温度上传到手机 APP 端显示。

下面开始分析下程序，这里我们主要讲解几个关键函数，详细的代码大家可以打开工程查看。

3.2.1 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，串口初始化设置在前面串口通信章节我们已经做过介绍，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
}

```

3.2.2 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=\"PRECHIN\", \"prechin168\", 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(1000);
}

```

```
}
```

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。

3.2.3 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;
int temp_value;
u8 temp_buf[5];
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_temperuture()*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%100==0)
    {
        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
    }
}
}

```

3.2.4 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。

3.2.5 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：

```
/******  
*****  
深圳市普中科技有限公司（PRECHIN 普中）  
技术支持：www.prechin.net  
  
实验名称：PZ-ESP8266 (模组) 模块实验  
接线说明：DS18B20 模块-->单片机 IO  
          VCC-->5V  
          DATA-->P3.7  
          GND-->GND  
  
          PZ-ESP8266 WIFI 模块-->单片机 IO  
          TXD-->P3.0  
          RXD-->P3.1  
          VCC-->3.3V  
          GND-->GND  
          CH_PD-->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：注意温度传感器的方向，在接口处我们已经用丝印画了一个凸起，  
          所以只需要将温度传感器对应插入即可  
  
*****  
*****/  
#include "public.h"  
#include "wifi_control.h"  
  
/******  
*****  
* 函 数 名      : main  
* 函数功能      : 主函数  
* 输    入      : 无  
* 输    出      : 无  
*****
```

```

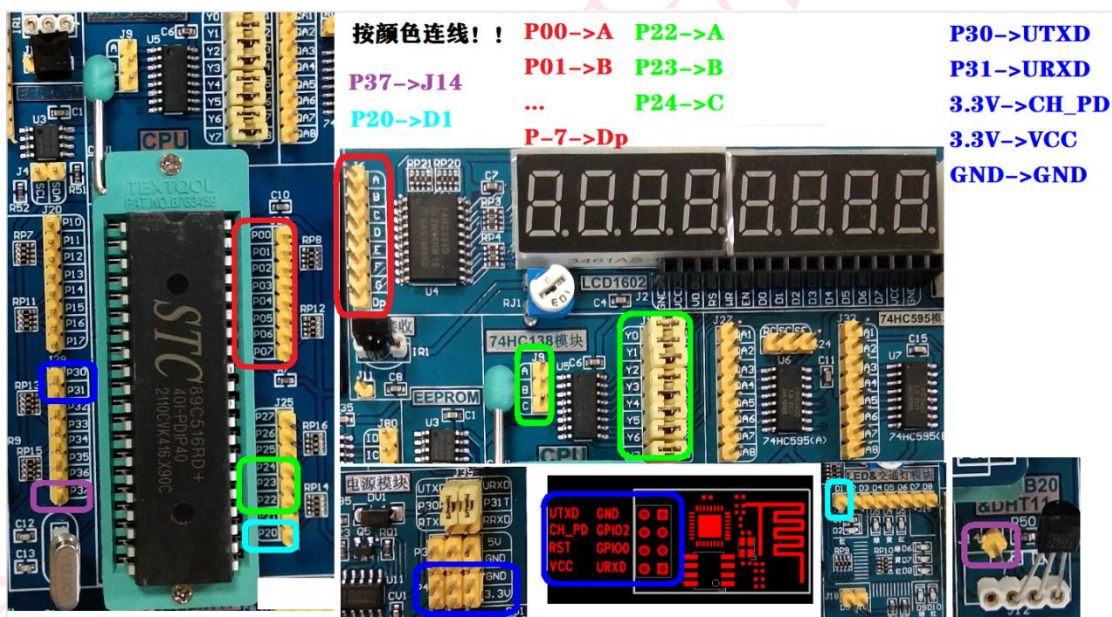
*****/
void main()
{
    wifi_control_init();

    while(1)
    {
        wifi_control();
    }
}

```

3.3 实验现象

使用 USB 线将开发板和电脑连接成功后（电脑能识别开发板上 CH340 串口），把编译后产生的 .hex 文件烧入到芯片内，然后将开发板上晶振电路的默认 12M 晶振替换成 11.0592M 晶振，按照如下接线方式：



下面需要我们用手机连接之前给模块创建的热点“PRECHIN”，并且前面我们说了，PZ-ESP8266 模块可同时多个手机连接热点，本实验只演示一个手机连接，并且其连接的 ID 是 ID0。

首先需要给手机安装一个网络调试助手 APP，在模块资料“..\调试工具\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：



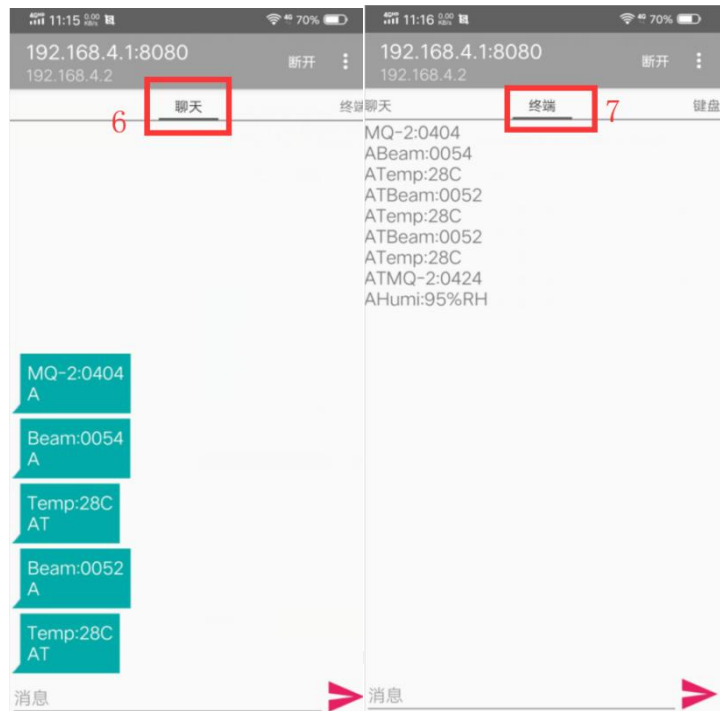
打开安装好的 APP，如下所示：



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“空调开”命令是字符 1，“空调关”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**

4 PZ-ESP8266 模块固件升级

PZ-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。

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 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)(PZ-ESP8266 模块-->普中 51 开发板)**

```
VCC-->3.3V
CH_PD-->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
```

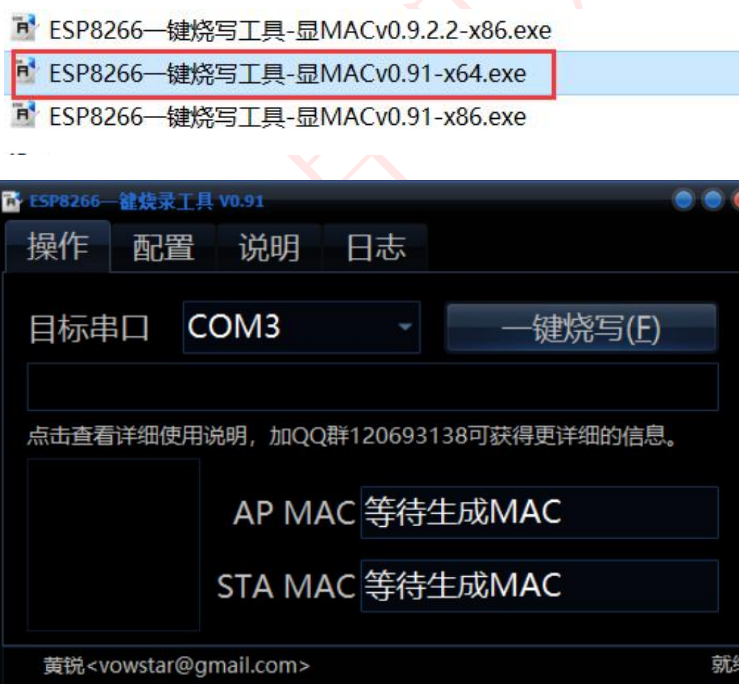
(2) 上网下载固件。网上有很多技术论坛都有制作 ESP8266 芯片固件，用户可以自行上网搜索，根据自己的需求下载中意的固件。笔者要演示的固件是从“物联世界”论坛下载的安信可官方的 0.95 版本的固件，存放于“**..\ESP8266 固件及烧写工具\ESP8266 固件\安信可官方 0.95\v0.9.5.0 AT Firmware.bin**”。


(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：

查看有关计算机的基本信息

Windows 版本	
Windows 10 家庭中文版	
© 2019 Microsoft Corporation。保留所有权利。	
系统	
制造商:	Hasee
处理器:	Intel(R) Core(TM) i5-9400 CPU @ 2.90GHz 2.90 GHz
已安装的内存(RAM):	8.00 GB (7.81 GB 可用)
系统类型:	64 位操作系统, 基于 x64 的处理器
笔和触控:	没有可用于此显示器的笔或触控输入

打开“..\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（“..\ESP8266 固件及烧写工具\ESP8266 固件\安信可官方 0.95\v0.9.5.0 AT Firmware.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。



5 其他

(1) 购买地址（普中授权店铺）

<http://www.prechin.net/forum.php?mod=viewthread&tid=38746&extra=>

(2) 资料下载

<http://prechin.net/forum.php?mod=viewthread&tid=35264&extra=page%3D1>

(3) 技术支持

普中官网: www.prechin.cn

普中论坛: www.prechin.net

技术电话: 0755-21509063（转技术）