（一）：基本使用

ESP8266 测试板到了，在此记录一下使用过程。

先上图：

白色的板子。

上电后测试手机 APK，先安装 APK 程序（见附件“ESP8266 安卓客户端.rar”）。

再打开手机 wifi，会发现有一个 ssid 名为 ESP8266 的 AP,选中进行连接，连接密码为“0123456789”：

连接成功后，打开安装的 APK 程序，先点“连接”按钮，与测试板建立 TCP 连接，然后就可以控制灯/继电器/蜂鸣器：

经测试一切正常。

注意板上有两个拨码开关，如果设置不正确可能无法通过手机来控制测试板。

拨码开关设置含义如下（UP 表示上方的拨码开关，DOWN 表示下方的拨码开关,'1'表示 ON 位置）：

手机控制测试板状态（板载 MCU 和 ESP8266 模块串口通信），如第一张图所示:

UP :011010

DOWN:110000

计算机通过 MIni-USB 烧写/连接 MCU 串口：

DOWN:000101

计算机通过 MIni-USB 烧写/连接 ESP8266：

UP :011110(烧写)、011010（连接运行）

DOWN:001010

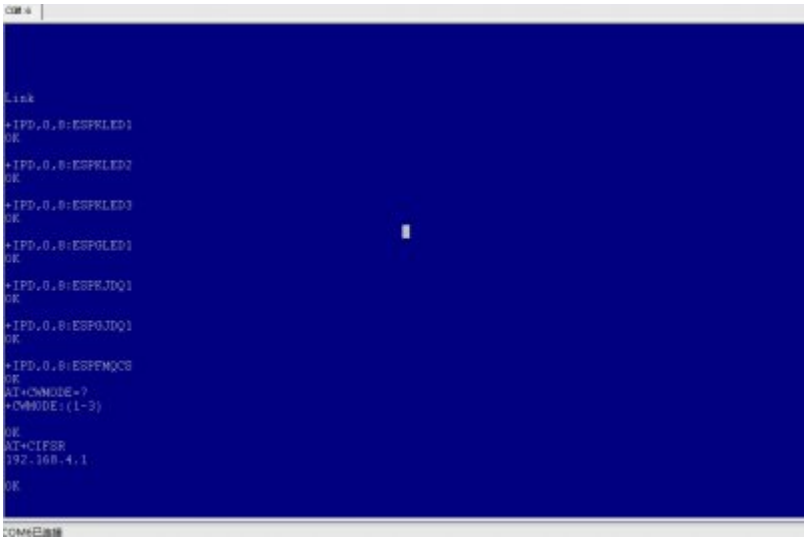
用 USB 转 DC 口的转接线，可以通过移动电源给测试板供电：

通过白尾巴测试板子的功耗，大概 1W 左右（5.01V*0.19A)，也可以通过 Mini-USB 口直接供电。

通过笔记本连接测试板的 Mini-usb，会提示找到串口设备，如需安装驱动请见附件“STC 下载器驱动 CH341SER.zip”。

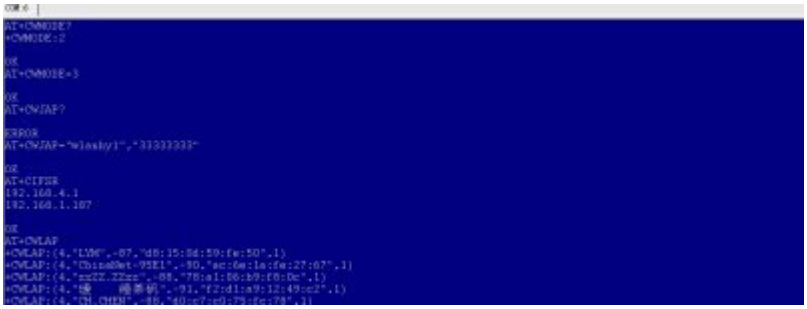
正常安装后，在设备管理器会发现新的串口设备：

打开串口工具，连接 com6，然后将下方的拨码开关由 110000 变成 001010，就可以在计算机上发送 AT 命令给 ESP8266 或接收 ESP8266 收到的信息：



注意 AT 命令需以回车、换行（0x0D、0X0A）结尾，有的终端工具只是发送回车键，因此不能触发 AT 命令的响应。

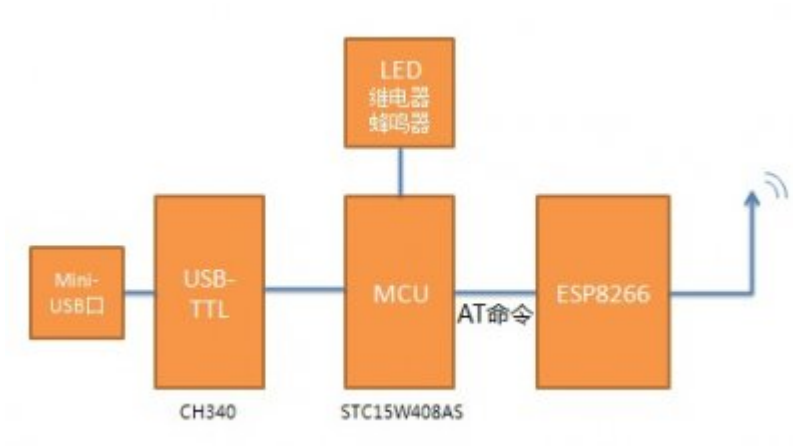
测试板上的 MCU 缺省设置 ESP8266 为 AP 模式（模式 2），我们可以通过命令设置为模式 3（AP+STA 模式），并使测试板连上家中的 AP：



基本操作就是这些，比较简单。

（二）：烧写固件

测试板的整体框图如下：



有三个主要芯片：USB 转串口芯片、STC 单片机、ESP8266

USB 转串口芯片主要用于调试和烧写芯片，输出的串口信号可以通过拨码开关连接至 MCU 或 ESP8266

MCU 起主控作用，通过 AT 命令设置 ESP8266 的工作模式、开启 ESP8266 的 TCP 服务器，并根据 8266 接收到的 TCP 信息来控制外围设备（如 LED）的动作。

ESP8266 加载 AT 固件，工作在从模式。与 AT 模式相对应的是 IOT 主模式。

IOT 和 AT 模式的区别：
IOT：物联网(Internet Of Things)
AT：调制解调器命令语言

在 SDK 源码 project 目录中有 AT 和 IOT 源码，区别如下：
1、IOT_Demo 位于软件包中 "examples" 文件夹，给出三种物联网设备“智能开关”，“灯”，“传感器”的简单 demo，三种设备在 user_config.h 中定义，请每次只使能一种设备调试~

2、AT 是另一个应用 demo，示范 ESP8266 作为 slave 外接一个 Host，Host 通过 AT 指令控制 ESP8266 联网传数据等操作。
3、AT 是与 IOT_Demo 同一级别的应用 demo，请勿同时拷贝到文件夹“app” 编译。未改动代码的情况下，要么作为独立运行的 IOT_Demo，要么作为附属 wifi 功能的 AT。

简而言之，如果使用 MCU 作为主控，ESP8266 一般为 AT 模式，如果不使用 MCU，ESP8266 作为主控，即为 IOT 模式。

下面介绍如何烧写 MCU 和 ESP8266：

一、烧写 MCU

1、准备需加载的 MCU 文件

源码及 HEX 文件如附件“MCU.rar”,官方发布的不太对 ,修改了一些 ,如将波特率从 115200 改为 9600，增加发送字符间的延时。

源码采用 Keil uVision4 编译

2、准备烧写程序

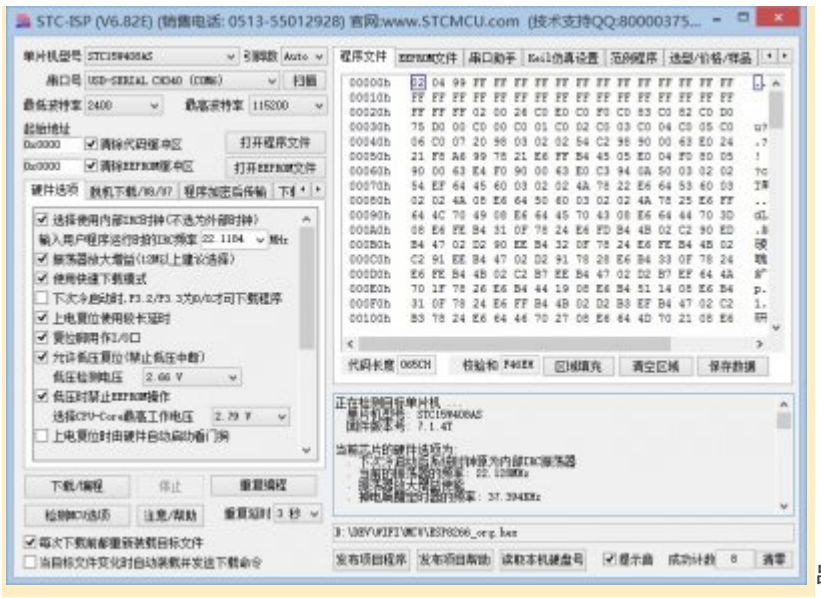
见附件 “ISP.rar “

3、设置测试板上的拨码开关

参考《玩转 ESP8266 测试板（一 》》

下方拨码开关设置为:000101

4、启动烧写程序，按如下设置，注意时钟频率的设置(22.11M)



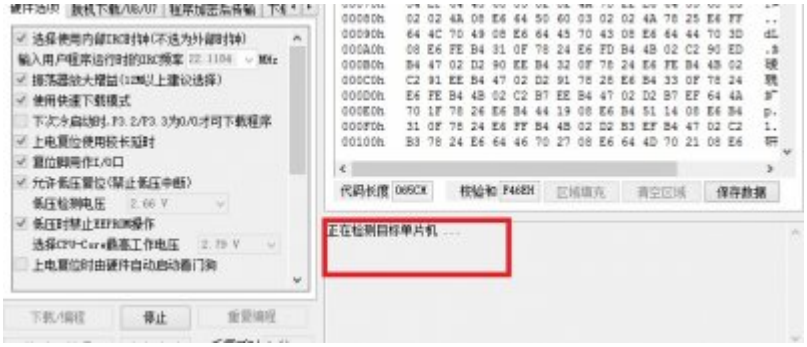
串口设置为自己的串口，

点“打开程序文件”，选中需加载的 MCU 文件

5、点“下载/编程”按钮，然后测试板下电再上电

因为 STC 单片机内置的 ISP 烧写程序在启动时才生效，因此单片机需复位才能烧写，测试板没有 MCU 的复位键，只能下电再上电

没重启时，程序提示“正在检测目标单片机”：



重启后会正常烧写。

给测试板的建议：

1）将 Mini-USB 改为 Micro-USB

2）增加 MCU 的复位键，目前只有 ESP8266 的复位键

二、烧写 ESP8266

1、准备 ESP8266 的烧写固件

可以加载官方的固件或 IDE 编译后的固件，官方 AT 固件见附件“v0.9.5.0 AT Firmware.rar”

2、准备烧写程序

见附件“ESP8266Flasher-x86-v0.9.2.3.rar”

3、设置测试板上的拨码开关

参考《玩转 ESP8266 测试板（一）》

上方拨码开关设置为:011110

下方拨码开关设置为:001010

4、启动烧写程序，按如下设置

先选择对应的串口：

再在配置页面选择固件：

然后返回操作页面，点击“一键烧写”，等待烧写完成，失败的话多复位几次：

烧写完成后记得把拨码开关恢复为正常状态（上方拨码开关设置为:011010）。

（三）：远程控制测试板

目前我们只能手机本地连接测试板，控制 LED 的开关，下面来试试怎么在远程任何地方控制测试板。

首先来看看软件的控制流程：

上电启动后，MCU 对 ESP8266 进行配置：

AT+CWMODE=2 设置成路由模式

AT+CWSAP="ESP8266","0123456789",11,0 设置路由

AT+RST 重启

AT+CIPMUX=1 设置成多连接

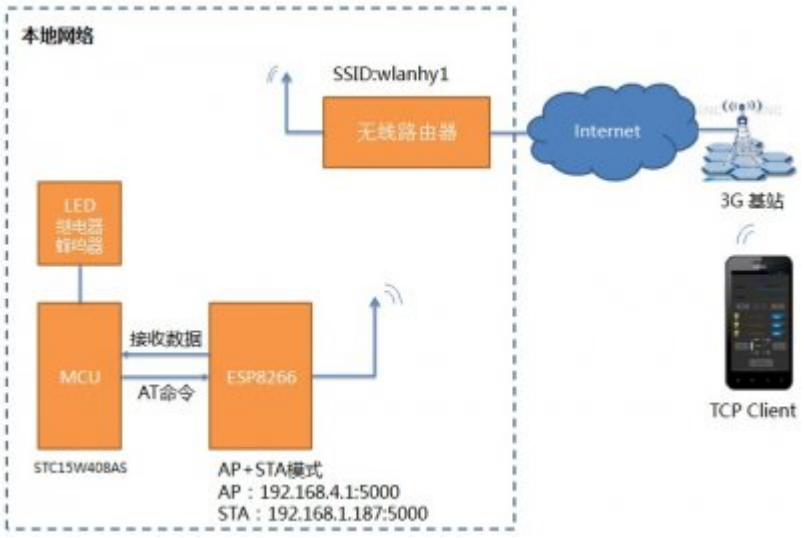
AT+CIPSERVER=1,5000 开启 TCP 服务端口

ESP8266 被配置成了 AP 模式，并开启了 TCP 服务器。

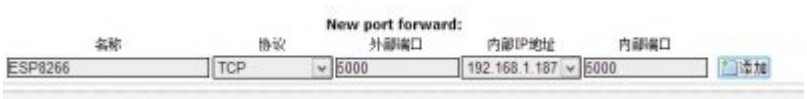
在手机侧的 APK 其实就是一个 TCP 客户端，当用户在手机上点开灯按钮时，会发送相应的 TCP 数据（如 ESPKLED1）给 ESP8266 上的服务器，ESP8266 收到 TCP 数据后，会在串口进行转发（如+IPD,0,10:ESPKLED1 ），MCU 的串口收到 ESP8266 串口的信息后，分析

其内容并控制相应的 LED 点亮。

如果想远程控制测试板，我们需要将测试板连接至家里的无线路由器，手机通过无线路由器远程来控制测试板：



如上图所示，首先需将 ESP8266 设置成 AP+STA 模式(mode=3)或 STA 模式(mode=1)，ESP8266 的 STA 连上无线路由器，获取到 IP 地址，比如是 192.168.1.187；再在 AP 和 STA 的 5000 端口都开启 TCP 服务器，接收 TCP 连接。然后需要在无线路由器上设置端口映射，将无线路由器收到的 TCP 信息转发到 ESP8266 的 STA 地址：

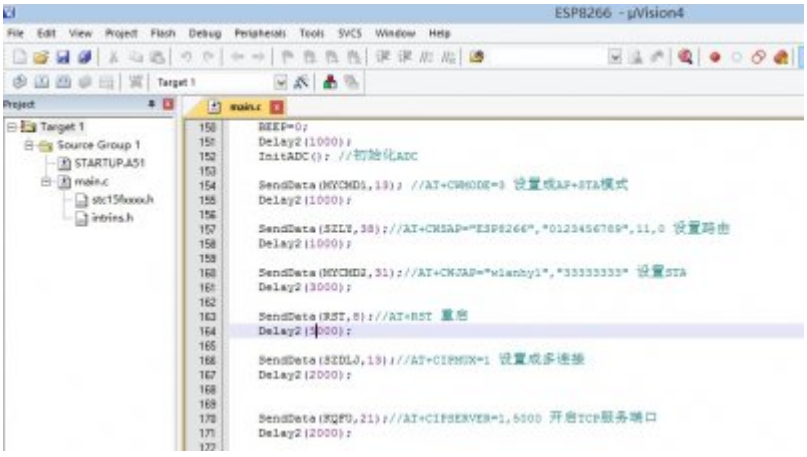


上面是我的路由器配置，其他路由器请参见相应的端口映射说明。

最后手机开启 3G 网络，打开 APK 控制程序，将 TCP 服务器地址填写成**无线路由器的公网 IP**，即可在全球任何可以上网的地方控制你家中的测试板了：

下面我们 Step by Step 来实现它：

1、修改 MCU 程序，用编译器编译，并烧写至 MCU：



源码见附件“MCU1.rar”，我还加了一些 ADC 的验证代码，可无视，里面的无线路由器的 SSID 和密码需改成你自己的。

2、验证 MCU 的正确性，将手机连接至无线路由器（注意不是测试板），然后手机上启动控

制 APK，里面的 TCP 服务器地址填 ESP8266 STA 的 IP 地址（如 192.168.1.187，可通过 AT+CIFSR 查询或在无线路由器上查询），看是否能正常控制测试板

3、在无线路由器上增加到 STA IP 地址的端口映射

4、手机通过 3G 上网，手机上启动控制 APK，里面的 TCP 服务器地址填无线路由器的 IP 地址，看是否能正常控制测试板

如一切正常，你就实现了第一个物联网设备了。

（四）：SDK 编程

下面通过实例来验证 ESP8266 的 SDK 编程，我们通过控制 ESP8266 的 GPIO0 口，使测试板上与 GPIO0 相连的 LED 1 秒钟闪烁一次。

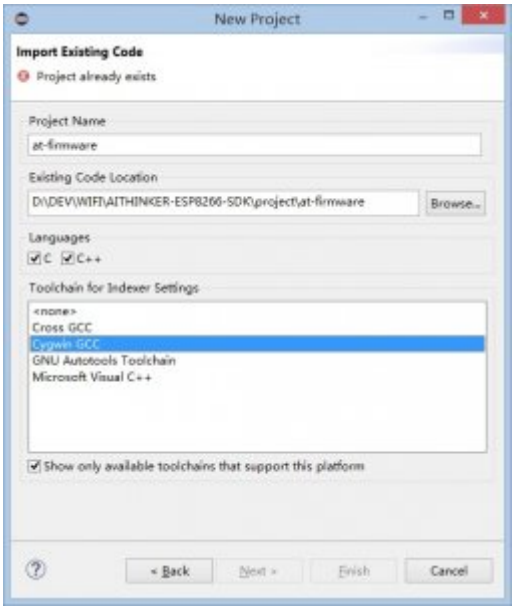
首先需要下载 IDE 2.0 编译环境：

http://www.ai-thinker.com/forum.php?mod=viewthread&tid=650&extra=page%3D1

因为测试板上使用的是 AT 固件，因此我们在 IDE 中导入下面路径的工程：

\AITHINKER-ESP8266-SDK\project\at-firmware

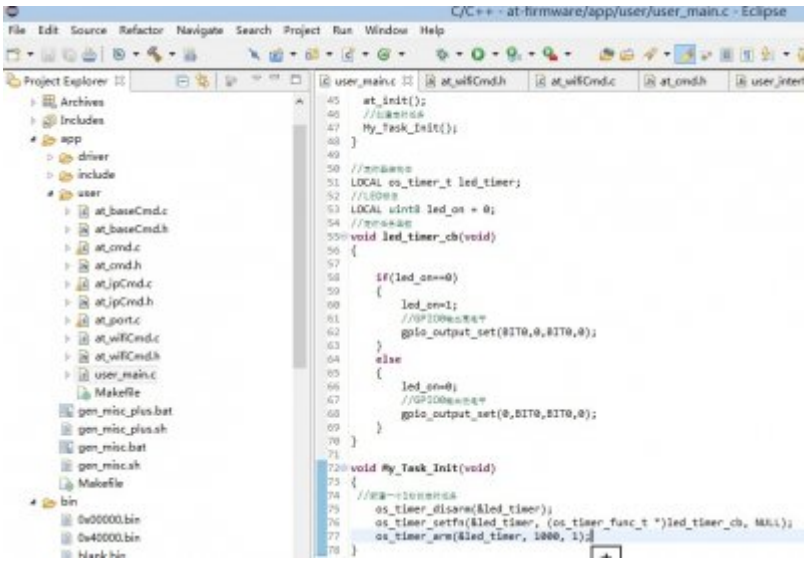
在 eclipse 中选“File”-“Import”-"Existing Code as Makefile Project"导入



导入后，打开“at-firmware\app\user\user_main.c"文件，先修改波特率，因为测试板缺省是 9600 波特率，修改为如下语句：

```
if(tempUart.saved == 1)
{
    uart_init(tempUart.baud, BIT_RATE_9600);
}
else
{
    uart_init(BIT_RATE_9600, BIT_RATE_9600);
}
```

再创建 LED 闪烁的定时任务：



关于 SDK 编程的文档参见附件“ESP 8266 SDK 文档.rar ”

修改后的 user_main.c 文件如下 “user_main.rar “

修改完成后保存并编译，会在” \AITHINKER-ESP8266-SDK\project\at-firmware\bin “目录生成两个文件 0x00000.bin、0x40000.bin。

烧写这两个文件到 ESP8266，设置如下：

大功告成!