

ESP8266使用入门教程

本文目标：了解esp8266以及其开发流程

芯片介绍：8266片上集成wifi+MCU，使用的是一个M0的内核，而且成本很低，因为片上有wifi和MCU，所以作为网络终端非常的方便，当然，因为是wifi，所以低功耗方面就别想了，低功耗+联网，NB-IOT更加合适。

固件：下面先介绍一下芯片固件的概念，说白了，esp8266也是一个单片机，上电还是得从0地址开始跑，平时我们使用单片机，一般都是使用keil等软件编程，然后下载，软件很多事情已经帮我们做好了，我们的重心放在main函数之后就行了。所谓的固件，我们可以把它看做一个很大的程序，只不过人家帮我们写好了，上电就开始运行，然后一直等待我们给单片机发送指令，我们发送指令后就执行相应的操作。

esp8266的固件有两种

- AT固件，芯片出厂的时候里边刷的就是AT固件，AT固件，用户主要通过串口使用AT指令跟8266交互，要控制8266。所以使用这种固件的时候还需要一个主机通过串口跟8266连接，这种使用方法，就单纯将8266当做一个网络传输芯片，串口转wifi，本文不讨论AT固件。
- Node-mcu固件，重点来了，因为这个固件才能完全发挥8266的魅力，先说一下这个固件的魅力，官方介绍是，这套固件，可以让8266像Arduino一样操作硬件IO，而且让你能完全使用API接口进行开发，更要命的是，固件里边可操作的模块还很多，像gpio操作、json处理、file文件创建管理、网络连接等等。举个例子说明一个这个固件：这个固件就像是安卓手机的刷机包，刷机之后我们就可以通过图形界面进行各种操作，在安卓手机上运行各种应用程序，esp8266刷入nodemcu固件之后，也能在上边运行我们编写的应用程序。

下面放几段操作8266的代码

```
--操作GPIO
pin = 1
gpio.mode(pin,gpio.OUTPUT)
gpio.write(pin,gpio.HIGH)
gpio.mode(pin,gpio.INPUT)
print(gpio.read(pin))

--连接wifi
wifi.setmode(wifi.STATION)
wifi.sta.config("SSID","password")
print(wifi.sta.getip())
```

代码基本上不用注释，一看名称知道是做啥的

下面开始讲如何搭建开发环境

就像上文说的，芯片出厂的时候是AT固件，要刷如nodemcu固件才能使用这种开发方式，说以先要刷固件，当初我开始看的时候，网上一大堆各种各样的各种版本的固件，不是说只有两种固件吗？现在先不用管这个，按照步骤来，后边慢慢说。

先连接8266，建议大家开始研究的时候使用开发板，这样能省下很多时候时间，后期再上核心板

1.首先打开刷固件工具ESP8266Flasher.exe，选择要刷入的固件

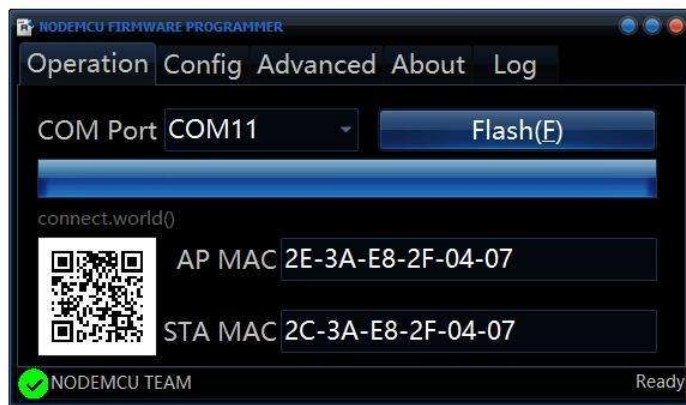


2.点击Flash开始烧写



<http://blog.csdn.net/CallMeSumo>

3. 等待一会烧写成功，如果不成功多试几次就行了



<http://blog.csdn.net/CallMeSumo>

接下来就可以开始写程序了，程序使用Lua语言编写的，至于为啥是Lua语言，因为这个固件里边包含一个Lua语言解释器，就好比安卓上使用java语言开发应用程序。

开始写第一个程序，最简单的就是串口输出了

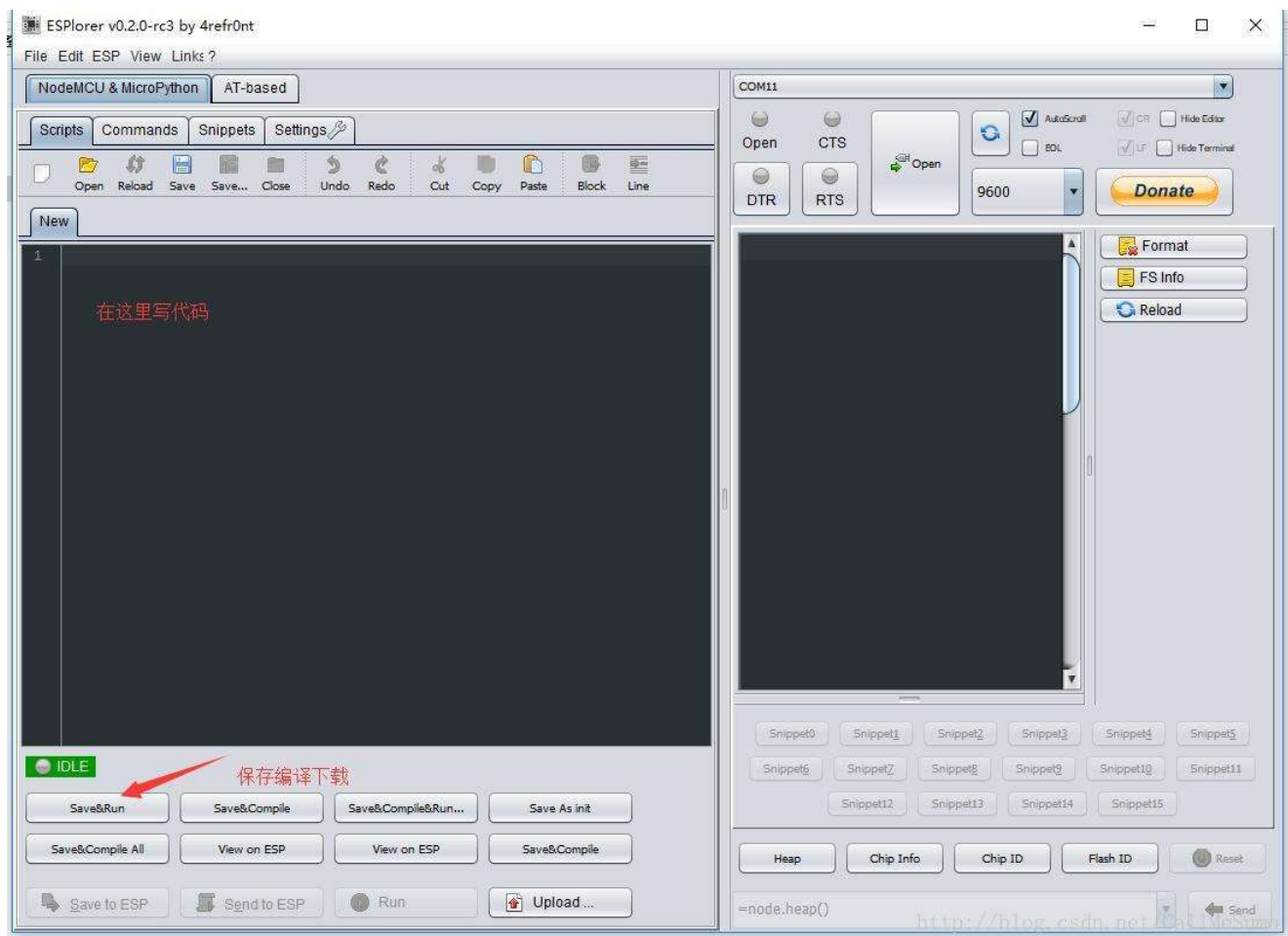
程序编辑以及烧写，使用另外一个软件ESPlorer

1. 解压ESPlorer.zip文件，得到以下东西

ESPlorer		
<div> <div> _lua</div> <div>2016/4/27 23:43</div> <div>文件夹</div> </div> <div> <div>_micropython</div> <div>2016/4/27 23:44</div> <div>文件夹</div> </div> <div> <div>lib</div> <div>2016/4/25 18:05</div> <div>文件夹</div> </div> <div> <div>ESPlorer.bat</div> <div>2016/4/21 23:04</div> <div>Windows 批处理...</div> <div>1 KB</div> </div> <div> <div>ESPlorer.jar</div> <div>2016/4/25 18:05</div> <div>Executable Jar File</div> <div>2,150 KB</div> </div> <div> <div>ESPlorer.Log</div> <div>2017/12/13 18:54</div> <div>文本文档</div> <div>4,212 KB</div> </div> <div> <div>ESPlorer.Log.1</div> <div>2017/12/8 17:38</div> <div>1 文件</div> <div>4 KB</div> </div> <div> <div>README.TXT</div> <div>2016/4/25 18:05</div> <div>TXT 文件</div> <div>2 KB</div> </div>		

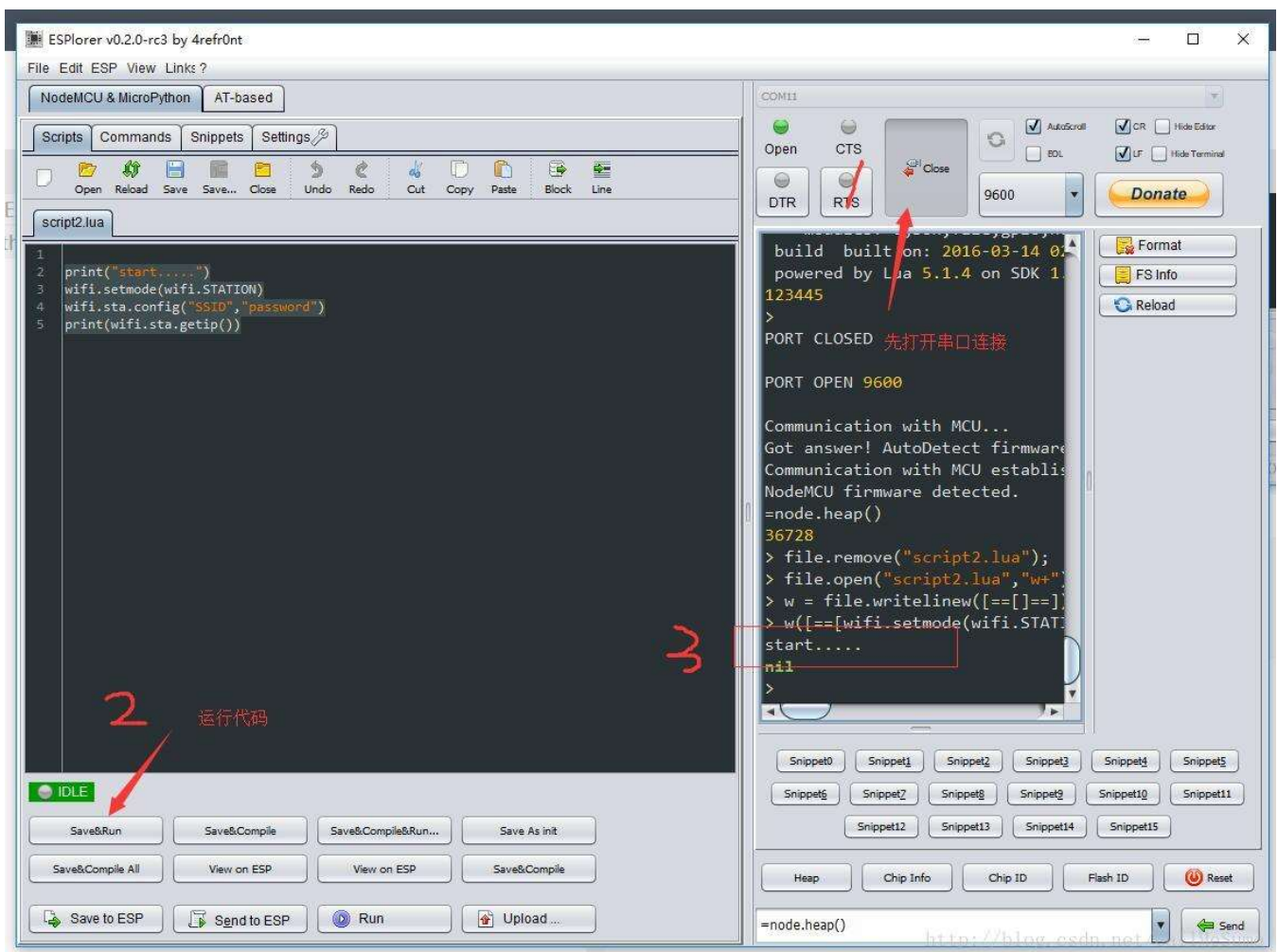
<http://blog.csdn.net/CallMeSumo>

2. 打开 ESPlorer.bat



3.开始写代码，我们让8266连接手机的wifi热点，当手机提示有新的终端接入的时候，就证明代码正确执行了

```
print("start.....")
wifi.setmode(wifi.STATION)
wifi.sta.config("SSID","password")
print(wifi.sta.getip())
```



看到串口这边有输出“start.....”,证明代码已经执行,等一会手机的热点应该会提示有新的设备接入了。

这里说明一下,8266复位的时候,默认是执行init.lua这个程序,所以我们要让程序一上电就开始运行,在保存文件的时候,就要将文件的名称改为init.lua,这样才能实现上电就运行

程序怎么写?

现在知道程序怎么写之后,就可以开始看一看这个固件的API文档了,里边有所有模块的API用法以及例子

网址: <https://nodemcu.readthedocs.io/en/master/en/modules/wifi/>

Overview

English

Home

Building the firmware

Flashing the firmware

Internal filesystem notes

Filesystem on SD card

Uploading code

FAQs

Lua Developer FAQ

Extension Developer FAQ

Hardware FAQ

Support

Modules

adc

ads1115

adxl345

am2320

apa102

bit

bme280

bmp085

cjson

coap

cron

crypto

Important

The WiFi subsystem is maintained by background tasks that must run periodically. Any function or task that takes longer than 15ms (milliseconds) may cause the WiFi subsystem to crash. To avoid these potential crashes, it is advised that the WiFi subsystem be suspended with `wifi.suspend()` prior to the execution of any tasks or functions that exceed this 15ms guideline.

The NodeMCU WiFi control is spread across several tables:

- `wifi` for overall WiFi configuration
- `wifi.sta` for station mode functions
- `wifi.ap` for wireless access point (WAP or simply AP) functions
- `wifi.ap.dhcp` for DHCP server control
- `wifi.eventmon` for wifi event monitor

方法名称，点击跳到详细说明和例子

<code>wifi.getchannel()</code>	Gets the current WiFi channel.
<code>wifi.getdefaultmode()</code>	Gets default WiFi operation mode.
<code>wifi.getmode()</code>	Gets WiFi operation mode.
<code>wifi.getphymode()</code>	Gets WiFi physical mode.
<code>wifi.nullmodesleep()</code>	Configures whether or not WiFi automatically goes to sleep in NULL_MODE.
<code>wifi.resume()</code>	Wake up WiFi from suspended state or cancel pending wifi suspension.
<code>wifi.setmode()</code>	Configures the WiFi mode to use.
<code>wifi.setphymode()</code>	Sets WiFi physical mode.
<code>wifi.startsmart()</code>	Starts to auto configuration, if success set up SSID and password automatically.
<code>wifi.stopsmart()</code>	Stops the smart configuring process.
<code>wifi.suspend()</code>	Suspend WiFi to reduce current consumption.

<http://blog.csdn.net/CallMeSumo>

固件的编译

可以看到nodemcu里边包含的模块很多，但是8266的资源是有限的，如果固件里边全部包含了这些模块，就很占用空间，这样我们可以写代码的地方就少了，而且有些模块并不是我们需要的，所以我们要能选择自己需要的模块，然后编译成自己定制的固件，然后再烧到芯片里边。

这个nodemcu是开源的，下载源码，设置好交叉编译链，选择需要的模块，在linux下可以编译出自己的固件，但是这样太麻烦，需要linux环境。官方还提供了一种方法，就是在线编译，选择自己需要的模块，填写自己的电子邮箱，一会之后就会将编译好的固件发送到你填写的邮箱

网址: <https://nodemcu-build.com/>

Select branch to build from

☒ master <> ☐ dev <> ☐ 1.5.4.1-final (frozen, for 512KB modules) <>

Click the <> to verify on GitHub that the selected branch actually contains what you expect it to.

Watch-out! Make sure you understand which SDK you get with a particular NodeMCU version. Double check the [release notes](#) and remember that NodeMCU master == latest release. When upgrading familiarize yourself with the [upgrade notes in the docs](#).

阿里云40+云产品6个月免费

限量领取8000元企业津贴 click.aliyun.com



广告 X

Select modules to include

在自己需要的模块前勾上, 不知道模块是干啥的就看看API文档

<input type="checkbox"/> ADC 📖	<input checked="" type="checkbox"/> file 📖	<input type="checkbox"/> PCM 📖	<input type="checkbox"/> struct 📖
<input type="checkbox"/> ADS1115 📖	<input type="checkbox"/> gdbstub 📖	<input type="checkbox"/> perf 📖	<input type="checkbox"/> Switec 📖
<input type="checkbox"/> ADXL345 📖	<input checked="" type="checkbox"/> GPIO 📖	<input type="checkbox"/> PWM 📖	<input type="checkbox"/> TCS34725 📖
<input type="checkbox"/> AM2320 📖	<input type="checkbox"/> HDC1080 📖	<input type="checkbox"/> RC (no docs)	<input type="checkbox"/> TM1829 📖
<input type="checkbox"/> APA102 📖	<input type="checkbox"/> HMC5883L 📖	<input type="checkbox"/> rfswitch 📖	<input checked="" type="checkbox"/> timer 📖
<input type="checkbox"/> bit 📖	<input type="checkbox"/> HTTP 📖	<input type="checkbox"/> rotary 📖	<input type="checkbox"/> TSL2561 📖
<input type="checkbox"/> BME280 📖	<input type="checkbox"/> HX711 📖	<input type="checkbox"/> RTC fifo 📖	<input type="checkbox"/> U8G 📖
<input type="checkbox"/> BMP085 📖	<input type="checkbox"/> I²C 📖	<input type="checkbox"/> RTC mem 📖	<input checked="" type="checkbox"/> UART 📖
<input type="checkbox"/> CoAP 📖	<input type="checkbox"/> L3G4200D 📖	<input type="checkbox"/> RTC time 📖	<input type="checkbox"/> UCG 📖
<input type="checkbox"/> Cron 📖	<input type="checkbox"/> MCP4725 📖	<input type="checkbox"/> SI7021 📖	<input type="checkbox"/> websocket 📖
<input type="checkbox"/> crypto 📖	<input type="checkbox"/> mDNS 📖	<input type="checkbox"/> Sigma-delta 📖	<input checked="" type="checkbox"/> WiFi 📖
<input type="checkbox"/> DHT 📖	<input type="checkbox"/> MQTT 📖	<input type="checkbox"/> SJSON 📖	<input type="checkbox"/> WPS 📖
<input type="checkbox"/> DS18B20 📖	<input checked="" type="checkbox"/> net 📖	<input type="checkbox"/> SNTP 📖	<input type="checkbox"/> WS2801 📖
<input type="checkbox"/> encoder 📖	<input checked="" type="checkbox"/> node 📖	<input type="checkbox"/> Somfy 📖	<input type="checkbox"/> WS2812 📖
<input type="checkbox"/> end user setup 📖	<input type="checkbox"/> 1-Wire 📖	<input type="checkbox"/> SPI 📖	<input type="checkbox"/> XPT2046 📖

Click the 📖 to go to the module documentation if you're uncertain whether you should include it or not.

按照步骤来就行了, 这就不细说了

工具下载: <https://pan.baidu.com/s/1nuDcgAX>

密码: i94p