目 录

**ESP8266快速入门手册**

**2015年9月22日**

[概述 1](#_Toc431398855)

[一、熟悉开发环境 1](#_Toc431398856)

[1-1开发环境搭建 1](#_Toc431398857)

[二、编译 1](#_Toc431398858)

[2-1 SDK结构说明 1](#_Toc431398859)

[2-2 SDK编译 2](#_Toc431398860)

[三、烧录 2](#_Toc431398861)

[3-1 烧录说明 2](#_Toc431398862)

[3-2 烧录工具使用 3](#_Toc431398863)

# 概述

本文档指导开发者快速上手HEKR-ESP8266-SDK进行应用开发。

HEKR-ESP8266-SDK源码托管在GitHub上维护，下载地址：<https://github.com/HEKR-Cloud/HEKR-ESP8266-SDK>

# 一、熟悉开发环境

## 1-1开发环境搭建

对于SDK编译建议在Linux环境下，可以采用ESP官方提供的开发环境工具包（下载地址<http://pan.baidu.com/s/1gd3T14n>），该工具包集成了VirtualBox-4.3.12 + LUBUNTU-14.04 + xtensa-lx106-elf（交叉编译工具），并附带环境搭建文档。

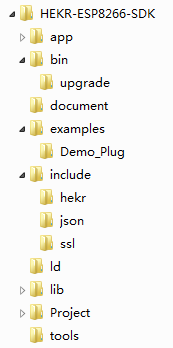
|  |
| --- |
| 注：LUBUNTU内已经安装好了交叉编译工具xtensa-lx106-elf，无需另行安装。 |

# 二、编译

## 2-1 SDK结构说明

HEKR-ESP8266-SDK源码中包含了开发者进行二次开发所需的头文件、API库文件以及相应功能的Demo示例等。

**目录结构如下：**



**目录结构说明：**

* app：用户工作区。用户在此目录下执行编译操作，用户级代码及头文件均放在此目录下。
* bin：二进制文件目录。该目录存放了编译生成的bin文件
* upgrade：该子目录存放编译生成的支持云端升级（FOTA）的固件（如user1.bin或user2.bin）。
* examples：示例demo目录。更多功能demo后续推出。
* Demo\_UART\_PASS：透传demo，使用时将该目录下的所有内容拷贝到app目录下编译。
* Demo\_Plug：智能插座demo，同上。
* include：该目录存放了自带头文件，包含了用户可使用的API函数以及相关宏定义，用户不需修改。
* ld：SDK编译链接时所需文件，用户不需修改
* lib：SDK编译所需库文件
* project：VS工程目录
* tools：编译生成bin文件所需工具，用户不需修改。

## 2-2 SDK编译

以串口透传示例代码为例。将examples/Demo\_UART\_PASS/目录下的所有文件拷贝至app目录下，进入app目录下，执行编译命令。命令原型如下：

make app=FIRMWARETYPE flash=FLASHSIZE

参数说明：

FIRMWARETYPE：选择编译生成固件的类型，决定是否支持云端升级

FLASHSIZE：Flash大小（单位为KiB）

执行命令：

make app=1 flash=2048

生成支持2MB大小Flash的固件user1.bin，在bin/upgrade目录下。

|  |
| --- |
| 1、SDK编译时默认生成支持FOTA的固件，对于不支持FOTA的固件编译不考虑。  2、烧录到Flash中的固件只需要user1.bin，而user2.bin在云端升级时才需要。 |

# 三、烧录

## 3-1 烧录说明

硬件平台：氦氪扩展板

**Flash Map**

根据Flash实际容量，将bin文件烧录到相应的地址。以下以2MB Flash为例：

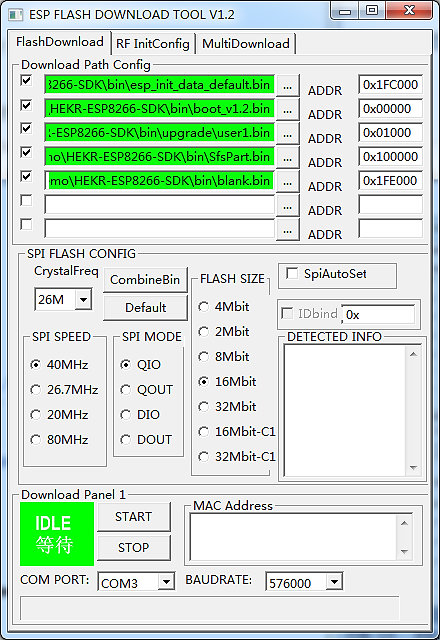
|  |  |  |
| --- | --- | --- |
| Bin文件 | 烧录地址（byte） | 备注 |
| boot\_v1.2.bin | 0x00000 | bootload |
| esp\_init\_data\_default.bin | 0x1FC000 | 存放RF的默认配置 |
| user1.bin | 0x01000 | 用户固件 |
| SfsPart.bin | 0x100000 | 文件系统 |
| blank.bin | 0x1FE000 | 用于擦除WiFi相关设置 |

|  |
| --- |
| 注：Flash大小必须使用2MB及其以上。Flash地址0x100000到0x1FA000被文件系统所占用，用户不要在里面写数据。 |

## 3-2 烧录工具使用

* **Windows环境**

烧录工具：FLASH\_DOWNLOAD\_TOOLS（以v1.2版本为例）[官方链接](http://bbs.espressif.com/viewtopic.php?f=57&t=433) [备用链接](http://yun.baidu.com/share/link?shareid=102192086&uk=4077187201)



1、Download Path Config区

选择要烧录的bin文件，填写相应的地址，勾选待烧录文件前的复选框。

2、SPI FLASH CONFIG

配置SPI Flash属性，分别设置SPI SPEED、SPI MODE、FLASH SIZE（16Mbit）。

3、将GPIO0拉低，上电使ESP进入**下载模式**，设置COM口和波特率，最后点击START按键进行烧录。

|  |
| --- |
| 下载模式：MTDO：0，GPIO0：0，GPIO2：1  运行模式：MTDO：0，GPIO0：1，GPIO2：1 |

4、下载完成后，将GPIO0拉高（悬空默认为高），重新上电使ESP进入**运行模式**，模块正常启动。

* **Linux环境**

烧录脚本：flashtool.py (需要先安装python)

1、将GPIO0拉低，上电使ESP进入**下载模式**。

2、Linux下识别模块设备节点（/dev/ttyUSB0）后，执行命令：

../tools/flashtool.py --port /dev/ttyUSB0 -b 921600 write\_flash --flash\_size 16m 0x01000 .. /bin/upgrade/user1.bin

参数说明：

/dev/ttyUSB0 ：模块设备节点，对应于windows下的COM口

921600下载波特率

16m FLASH大小，16m对应16Mbit Flash （16Mbit 以上的Flash也使用这个设置）

0x01000烧录地址

user1.bin 用户固件

|  |
| --- |
| 以上命令只实现用户固件的烧录，之前的配置信息并未擦除。如需实现完整烧录，请参考**debug.py脚本说明** |

3、下载完成后，将GPIO0拉高（悬空默认为高），重新上电使ESP进入**运行模式**，模块正常启动。

* **debug.py脚本说明**

为便于开发者进行开发，我们提供python脚本来快速实现SDK编译和固件烧写。

执行命令：

../tools/debug.py --onebin

参数：

--onebin：完成SDK编译并只烧录用户固件（配置不丢失）

--all：完成SDK编译并烧录所有固件（配置丢失）