ESP8266 快速入门手册

2015 年 9 月 22 日

目录

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。

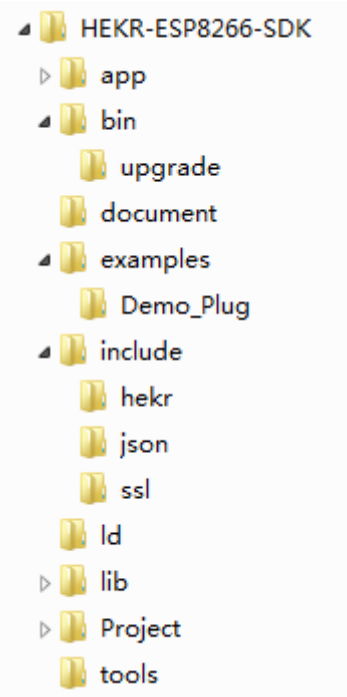
> 注：LUBUNTU 内已经安装好了交叉编译工具 xtensa-lx106-elf，无需另行安装。

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：编译工具以及烧录工具，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：

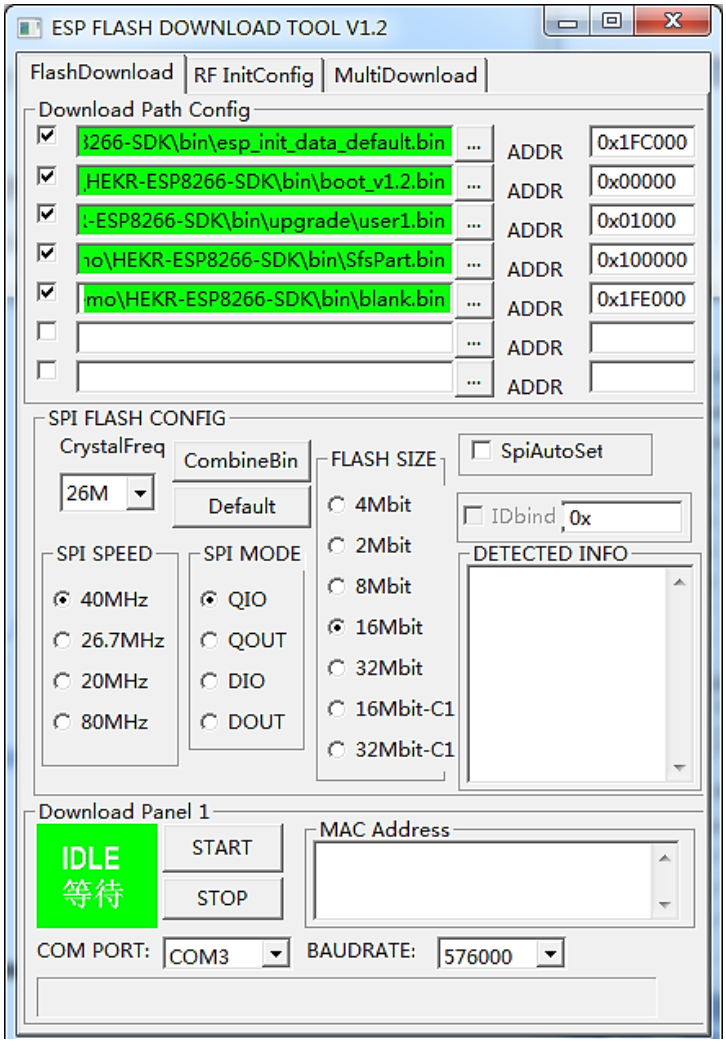
Bin 文件	烧录地址（byte）	备注
boot_v1.2.bin	0x00000	bootload
esp_init_data_default.bin	0x1FC000	存放 RF 的默认配置
user1.bin	0x01000	用户固件
SfsPart.bin	0x100000	文件系统
blank.bin	0x1FE000	用于擦除 WiFi 相关设置

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）官方链接备用链接



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）