

## 第二章 ESP8266 SDK 开发环境

本章主要从开发环境搭建、工程模板建立、开发流程说明三个方面进行,在开发环境搭建中,采用了 Eclipse+cygwin 的开发环境,并在该开发环境下建立了基于 NONOS 版本的 2.0 版本 SDK 和基于 MESH 版本的 SDK 工程模板,同时建立了基于 RTOS 的 SDK 开发工程模板。

### 2.1 搭建 ESP8266 开发环境

ESP8266 的编程环境目前主要为在虚拟机 (Virtual Box) 中安装 Linux 系统进行编译或者使用安信可公司集成的 ESP8266 IDE 开发环境 (Eclipse+cygwin), 同时还有 Eclipse + MinGW 等开发方式。

本篇讲的是 eclipse+cygwin 的方式,通过在工程中配置相关的环境变量的方式来实现的,与安信可 ESP8266 IDE 方式有一些细微的区别。

#### 2.1.1 Eclipse 安装

(1) 下载 Eclipse IDE for C/C++ Developers, 选择对应的操作系统版本, 下载的 Eclipse 为 Zip 包格式, 免安装, 直接解压至相应文件夹即可。(在此本文示例解压至系统程序目录下 C:\Program Files\eclipse)。

下载地址: <https://www.eclipse.org/downloads/eclipse-packages/>



(2) 下载安装 JRE, 因为 Eclipse 运行需要 Java Runtime Environment (JRE) 支持, 在系统未安装 JRE 时, 运行 Eclipse 会出现以下错误。



直接百度搜索“JRE”或者于 Oracle 官网下载 JRE, 下载对应操作系统的 JRE 安装即可。

JRE

网页 新闻 贴吧 知道 音乐 图片 视频 地图 文库

百度为您找到相关结果约12,500,000个

Java Runtime Environment最新官方版下载 百度软件中心

电脑版 Mac版



版本: 8.0.1210.13  
大小: 53.8M  
更新: 2017-01-18  
环境: WinXP/Win2003/Win7/Win8  
使用百度下载助手进行安全高速下载。  
高速下载 普通下载

Java SE Runtime Environment 8 Downloads

Do you want to run Java™ programs, or do you want to develop Java programs? If you want to run Java programs, but not develop them, download the Java Runtime Environment, or JRE™.

If you want to develop applications for Java, download the Java Development Kit, or JDK™. The JDK includes the JRE, so you do not have to download both separately.

JRE 8u121 Checksum

Java SE Runtime Environment 8u121

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

☐ Accept License Agreement ☒ Decline License Agreement

Product / File Description	File Size	Download
Linux x86	56.92 MB	<a href="#">jre-8u121-linux-i586.rpm</a>
Linux x86	72.76 MB	<a href="#">jre-8u121-linux-i586.tar.gz</a>
Linux x64	54.39 MB	<a href="#">jre-8u121-linux-x64.rpm</a>
Linux x64	70.26 MB	<a href="#">jre-8u121-linux-x64.tar.gz</a>
Mac OS X	62.28 MB	<a href="#">jre-8u121-macosx-x64.dmg</a>
Mac OS X	53.91 MB	<a href="#">jre-8u121-macosx-x64.tar.gz</a>
Solaris SPARC 64-bit	52.05 MB	<a href="#">jre-8u121-solaris-sparcv9.tar.gz</a>
Solaris x64	49.9 MB	<a href="#">jre-8u121-solaris-x64.tar.gz</a>
Windows x86 Online	0.7 MB	<a href="#">jre-8u121-windows-i586-rtw.exe</a>
Windows x86 Offline	53.81 MB	<a href="#">jre-8u121-windows-i586.exe</a>
Windows x86	59.17 MB	<a href="#">jre-8u121-windows-i586.tar.gz</a>
Windows x64 Offline	61.18 MB	<a href="#">jre-8u121-windows-x64.exe</a>
Windows x64	62.66 MB	<a href="#">jre-8u121-windows-x64.tar.gz</a>

安装完毕后, 可以正常运行 Eclipse 说明本操作步骤完成。

## 2.1.2 cygwin 安装

(1) 从云盘链接下载 cygwin 压缩包, 直接解压至相应的文件夹即可 (在此本文示例解压至系统程序目录下 C:\Program Files(x86)\cygwin)。

参考: Win32 系统下, 拷备至 C:\Program Files

Win64 系统下, 拷备至 C:\Program Files (X86)


## 2.1.3 ESP8266 SDK 下载

关于 ESP8266 的 SDK 大版本上主要包括了 NONOS SDK 和 RTOS SDK, 其中 NONOS SDK 最新版本为 2.0, RTOS SDK 版本为 1.5, 同时还有一个特殊的版本为基于 NONOS SDK 1.5.3 版本的 MESH 版本, 当具有 MESH 自组网的应用需求时, 请选择 MESH 版本的 SDK。

所有的 SDK 都可以在乐鑫官方论坛 (<http://bbs.espressif.com/>) 下载, 进行 SDKs 页面, 选择对应的版本。

DOWNLOADS

TOPICS POSTS LAST POST




SDKs

Download Software Development Kits here.

27 34


[GO TO GITHUB for LATEST SDK ! 请...](#)  
by [ESP\\_Lishuo](#)  
Mon Nov 14, 2016 5:38 pm



APKs

Download Android Packages (APKs) here.

Total redirects: 28904




Demos

Check out our MP3 decoder and Mesh Light demos here.

Total redirects: 34355

No unread posts



Tools

Download in-house designed tools here.

4 5

[Re: Tool: Reduce more power c...](#)  
by [ESP\\_Alfred](#)  
Tue Feb 02, 2016 10:22 pm

### (1) ESP8266 NONOS SDK V2.0

下载地址: <http://bbs.espressif.com/viewtopic.php?f=46&t=2451>

### (2) ESP8266 RTOS SDK V1.5

下载地址: [https://github.com/espressif/ESP8266\\_RTOS\\_SDK](https://github.com/espressif/ESP8266_RTOS_SDK)

### (3) ESP8266 MESH V1.3.2 With SDK V1.5.3

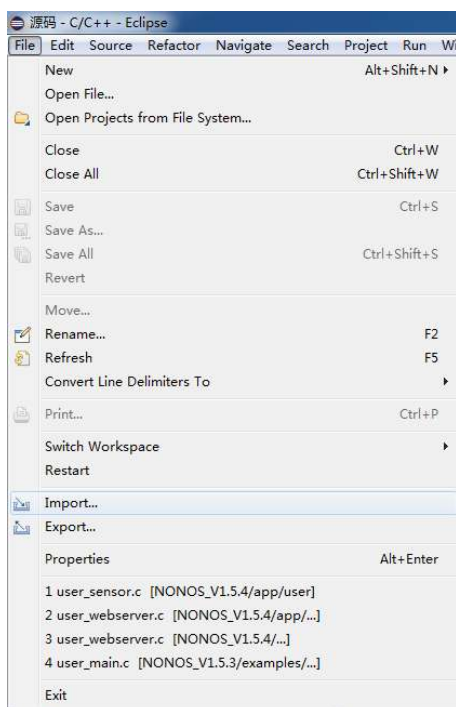
下载地址: [https://github.com/espressif/ESP8266\\_MESH\\_DEMO/releases/tag/v1.3](https://github.com/espressif/ESP8266_MESH_DEMO/releases/tag/v1.3)

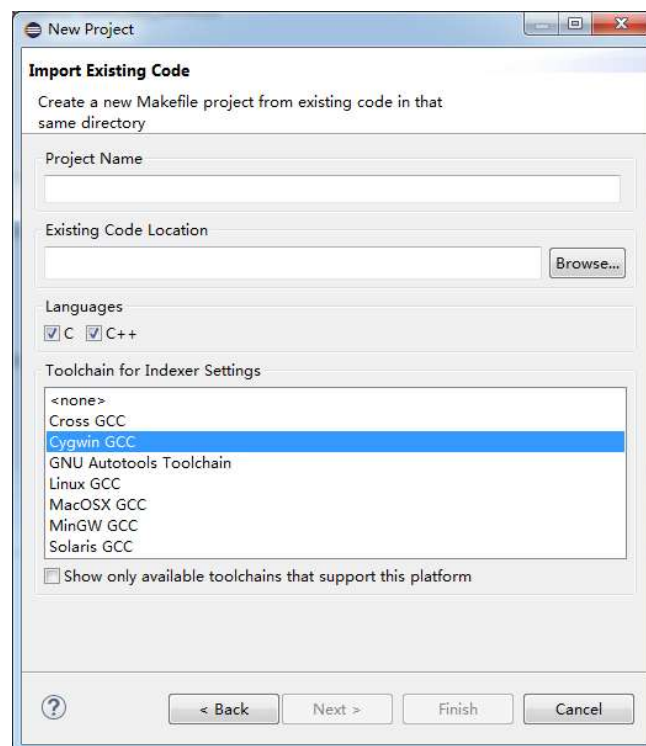
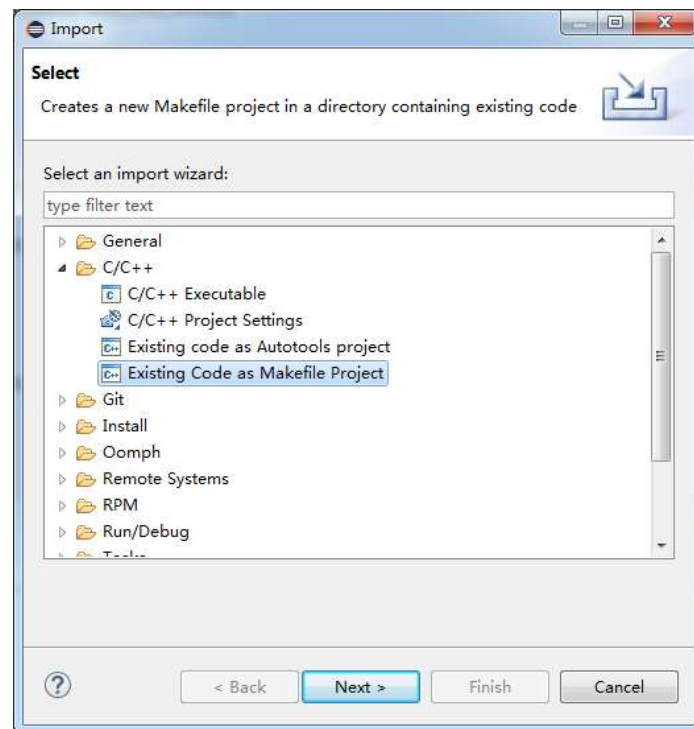
## 2.2 建立 ESP8266 开发工程

搭建 ESP8266 工程模板, 主要包括导入工程、修改路径、工程结构设置

### 2.2.1 ESP8266 SDK 工程导入建立

#### (1) 导入 Makefile 工程, 选择 Cygwin 工具链

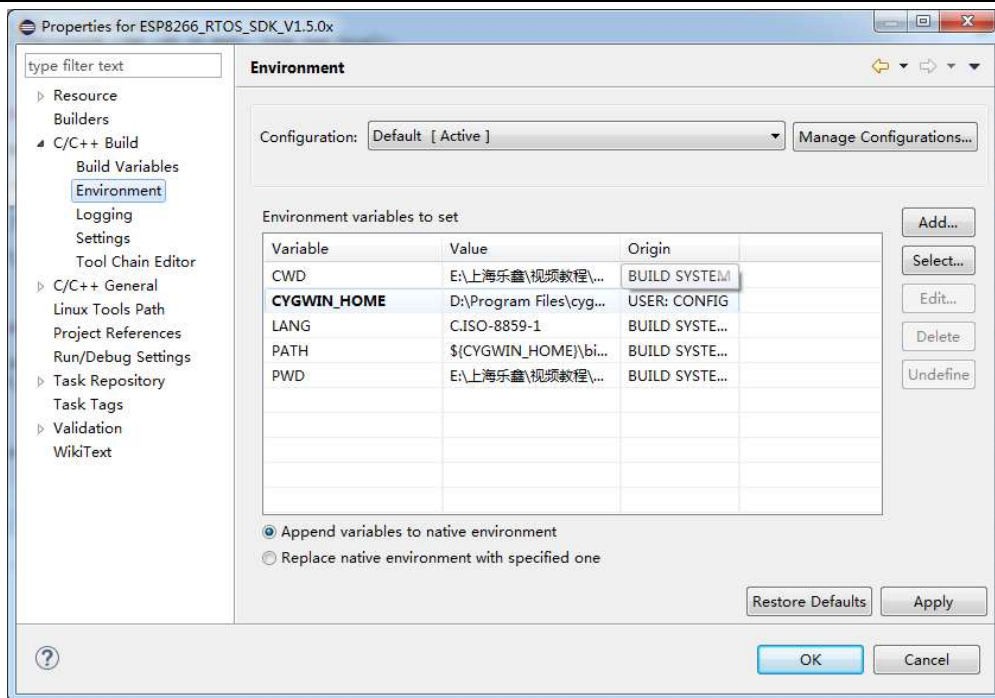




注: 如果没有 Cygwin GCC 选项, 请取消下方 show only available toolchains that support this platform 勾选框。

## 2.2.2 ESP8266 SDK 工程路径配置

点击相应的工程, 选择属性, 如下图, 选择 C/C++ Build 下的编译环境, 配置工程的相应编译路径:



需要配置的路径包括 CYGWIN\_HOME 和 PATH 两项:

(1) CYGWIN\_HOME 为 cygwin 的安装路径

(如: C:\Program Files (x86)\cygwin)

(2) 在 PATH 路径下添加 xtensa 处理器的编译路径

C:\Program Files (x86)\cygwin\opt\xtensa-lx106-elf\bin

(3) 与 NONOS 工程配置有一些区别在于, RTOS SDK 的路径配置完毕后, 需要修改工程用户文件夹下的 Makefile, 指定 SDK 路径 (SDK\_PATH) 和 BIN 路径 (BIN\_PATH), 修改后如下图所示:

```

13 #
14 TARGET = eagle
15 #FLAVOR = release
16 FLAVOR = debug
17
18 #EXTRA_CCFLAGS += -U
19 parent_dir=$(abspath $(shell pwd)/$(lastword $(MAKEFILE_LIST)))
20 parent_dir=$(shell dirname $(parent_dir))
21 parent_dir=$(shell dirname $(parent_dir))
22
23 SDK_PATH= $(parent_dir)
24 BIN_PATH=$(SDK_PATH)/bin
25
26 ifndef PDIR # {
27 GEN_IMAGES= eagle.app.v6.out
28 GEN_BINS= eagle.app.v6.bin
29 SPECIAL_MKTARGETS=$(APP_MKTARGETS)
30 SUBDIRS= \
31     user
32

```

即添加如下路径定义, 用于指明 SDK 与 BIN 文件路径配置:

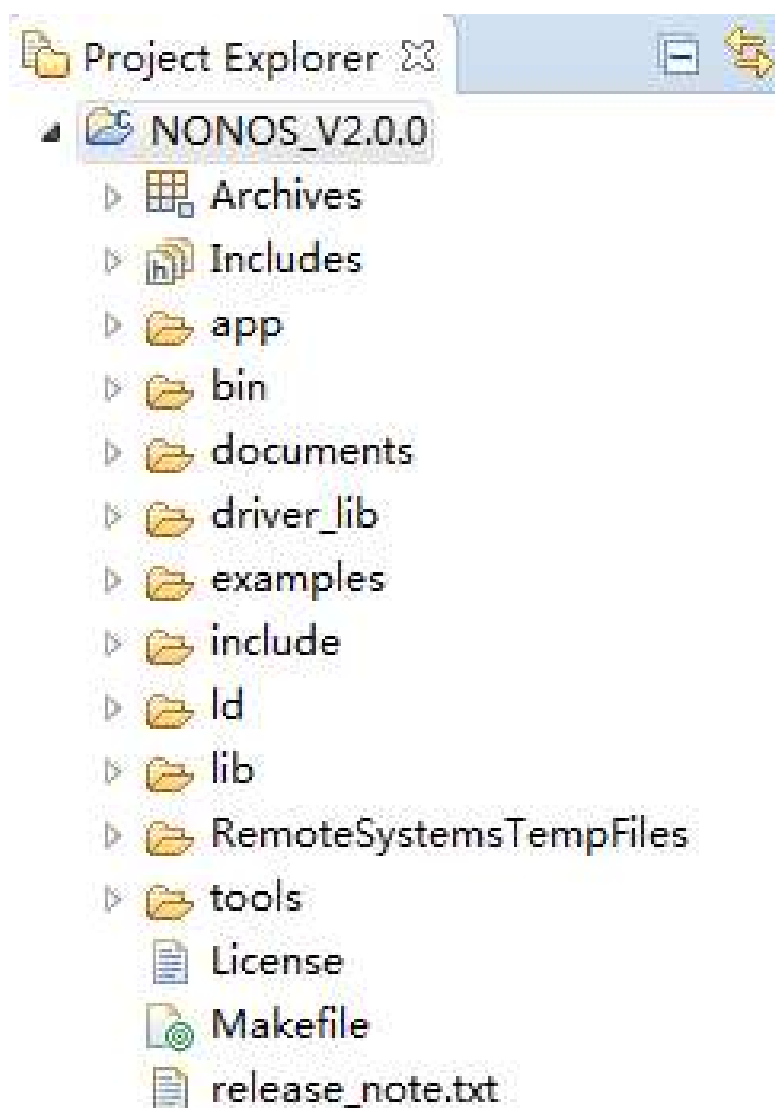
```
parent_dir=$(abspath $(shell pwd)/$(lastword $(MAKEFILE_LIST)))
parent_dir=$(shell dirname $(parent_dir))
parent_dir=$(shell dirname $(parent_dir))

SDK_PATH= $(parent_dir)
BIN_PATH=$(SDK_PATH)/bin
```

## 2.2.3 ESP8266 SDK 工程模板设计

### 1、工程文件结构说明

下图是在导入工程后, 在 Eclipse 中显示的一个对应工程模板, 其文件夹结构亦如 Eclipse 中显示完全一样, 该工程模板与乐鑫官方 SDK 具有较为一致的对应性, 建议按下工程模板开发用户程序。



结合以上工程文件结构,对结构中相应文件夹说明如下,在下面说明中,备注可删除的文件或文件夹可自行删除:

- (1) `app`: 用户程序代码文件夹;
- (2) `bin`: 用户编译完成后生成的待下载程序文件夹;
- (3) `documents`: 文档存放文件夹 (可删除);
- (4) `driver_lib`: 乐鑫官方的相关驱动程序,该目录中的文件,结合用户程序的使用可选择性拷贝至用户目录 (该文件夹可删除,如果不删除本文件夹,请通过改名或删除以使本文件夹下的 `Makefile` 文件无效);
- (5) `examples`: 乐鑫官方例程文件 (可删除);
- (6) `include`: 乐鑫官方 SDK 相关头文件;
- (7) `ld`: 程序生成链接文件;
- (8) `lib`: 乐鑫官方库文件;
- (9) `RemoteSystemTempFiles`: Eclipse 生成的文件;
- (10) `tools`: 部分工具文件;
- (11) `License`: 乐鑫官方版权说明文件。
- (12) `Makefile`: 整体工程编译及链接的文件;
- (13) `relese_note.txt`: SDK 版本说明文件

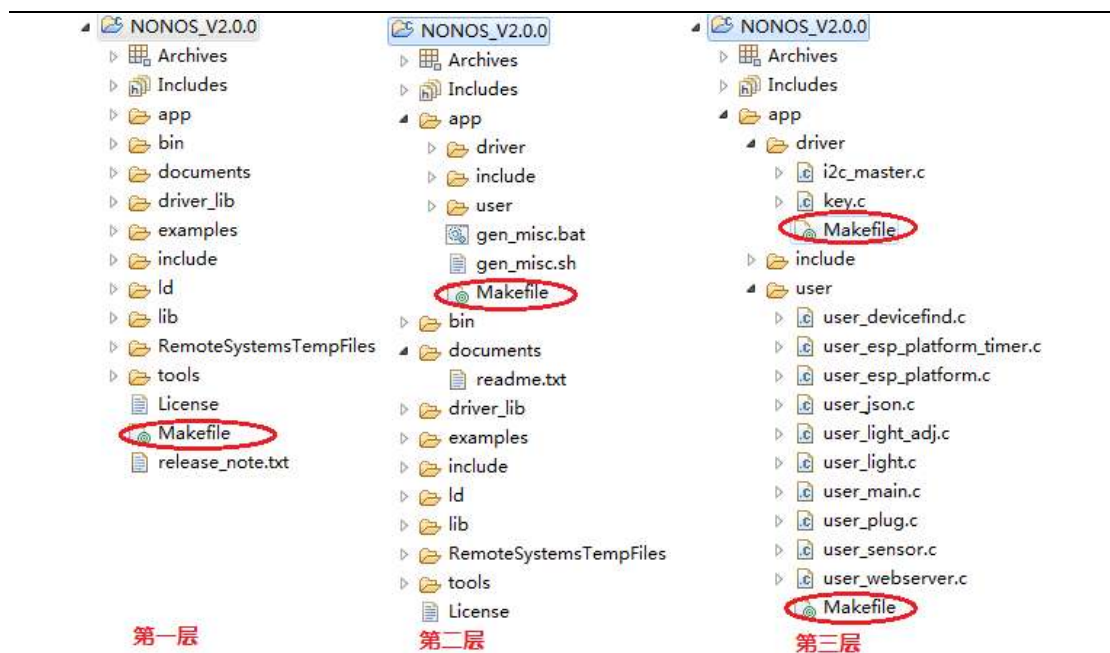
**注意:** 在需要更改文件夹名称时 (如 `IoT_Demo` 更改为 `app` 时,不要在 `eclipse` 下进行更改,因为在 `eclipse` 下进行更改时,会自动进行头文件引用的更改,导致出现引用错误),请在操作系统中进行更改后,在 `eclipse` 下进行刷新。

注: `NONOS` 版本的工程文件路径可以为中文路径, `RTOS` 版本的工程文件路径不可有中文和空格。

## 2、`Makefile` 文件说明

在整个工程中, `Makefile` 的结构分为 3 层,第一层为工程文件夹下的 `Makefile` 文件 (1 个),第二层为 `app` 文件夹下的 `Makefile` 文件 (1 个),第三层为 `app` 下的相关文件夹下的 `Makefile` 文件 (若干),其三层结构如下所示:





(1) 第一层 Makefile 文件在实际开发时, 主要用来修改与模块的对应参数, 主要包括是否支持云升级 boot、云升级模式下当前下载的 app、同时还有 SPI 下载相关的参数 SPI\_SPEED、SPI\_MODE、SPI\_SIZE\_MAP, 实际下载中, SPI 的相关参数可以在 FlashTool 下载时设置, 此处可不用更改, boot 及 app 参数说明如下:

**BOOT?=none**

说明: 用于选择Boot的版本, 通过Boot的版本对应了不同的Flash的组成结构, 在Flash容量小于1M时, 选择Boot版本为boot.bin; 在Flash容量为1M及以上时, 选择的Boot版本为boot\_v1.2+.bin。

选项值: none (无) old (512kFlash) new (1024k+Flash)

**APP?=0**

说明: 用于选择生成的为user1.bin或user2.bin。

选项值: 0 (无) 1 (user1.bin) 2 (user2.bin)

注: 参考《99c-esp8266\_ota\_upgrade\_cn\_v1.6.pdf》

(2) 第二层 Makefile 文件在实际开发时, 主要用于修改 app 文件夹下用户的源码编译关联性, 该文件建立了 app 文件夹下子文件夹的 Makefile 的联系, 在实际使用时, 主要修改的地方为需引入的子文件夹 (SUBDIRS =) 和用户程序的组成链接文件 (COMPONENTS\_eagle.app.v6 =), 具体对应位置如下图所示。



```

19
20 ifndef PDIR # {
21 GEN_IMAGES= eagle.app.v6.out
22 GEN_BINS= eagle.app.v6.bin
23 SPECTAI_MKTARGETS=$(APP_MKTARGETS)
24 SUBDIRS= \
25     user \
26     driver
27
28 endif # } PDIR
29
30 APPDIR = .
31 LDDIR = ../ld
32
33 CFLAGS += -Os
34
35 TARGET_LDFLAGS = \
40
41 ifeq ($(FLAVOR),debug)
42     TARGET_LDFLAGS += -g -O2
43 endif
44
45 ifeq ($(FLAVOR),release)
46     TARGET_LDFLAGS += -g -O0
47 endif
48
49 COMPONENTS_eagle.app.v6 = \
50     user/libuser.a \
51     driver/libdriver.a
52
53 LINKFLAGS_eagle.app.v6 = \
54     -L../lib \
55     -nostdlib \
56     -T$(LD_FILE) \

```

同时在部分特殊的情况下,可能会对引入官方库文件进行添加或删除,相应的修改位置为 LINKFLAGS,通过修改该参数以添加对应的链接库,如下图所示:

```

49 COMPONENTS_eagle.app.v6 = \
50     user/libuser.a \
51     driver/libdriver.a
52
53 LINKFLAGS_eagle.app.v6 = \
54     -L../lib \
55     -nostdlib \
56     -T$(LD_FILE) \
57     -Wl,--no-check-sections \
58     -Wl,--gc-sections \
59     -u call_user_start \
60     -Wl,-static \
61     -Wl,--start-group \
62     -lc \
63     -lgcc \
64     -lhal \
65     -lphy \
66     -lpp \
67     -lnet80211 \
68     -llwip \
69     -lwpa \
70     -lcrypto \
71     -lmain \
72     -ljson \
73     -lupgrade \
74     -lssl \
75     -lpwm \
76     -lsmartconfig \
77     $(DEP_LIBS_eagle.app.v6)
78     -Wl,--end-group
79

```

(3) 第二层 Makefile 文件非常简单,在实际开发时,只需要更改对应的文件夹源码生成文件 (GEN\_LIBS =), 具体修改位置如下图所示:

```
#####
3# Required variables for each makefile
4# Discard this section from all parent makefiles
5# Expected variables (with automatic defaults):
6# CSRCs (all "C" files in the dir)
7# SUBDIRS (all subdirs with a Makefile)
8# GEN_LIBS - list of libs to be generated ()
9# GEN_IMAGES - list of images to be generated ()
10# COMPONENTS_xxx - a list of libs/objs in the form
11#   subdir/lib to be extracted and rolled up into
12#   a generated lib/image xxx.a ()
13#
14ifndef PDIR
15GEN_LIBS = libuser.a
16endif
17
18
19#####
20# Configuration i.e. compile options etc.
21# Target specific stuff (defines etc.) goes in here!
22# Generally values applying to a tree are captured in the
23#   makefile at its root level - these are then overridden
24#   for a subtree within the makefile rooted therein
25#
26#DEFINES +=
```

user文件夹下Makefile

```
7# SUBDIRS (all subdirs with a Makefile)
8# GEN_LIBS - list of libs to be generated ()
9# GEN_IMAGES - list of images to be generated ()
10# COMPONENTS_xxx - a list of libs/objs in the form
11#   subdir/lib to be extracted and rolled up into
12#   a generated lib/image xxx.a ()
13#
14ifndef PDIR
15GEN_LIBS = libdriver.a
16endif
17
18
19#####
20# Configuration i.e. compile options etc.
21# Target specific stuff (defines etc.) goes in here!
22# Generally values applying to a tree are captured in the
23#   makefile at its root level - these are then overr
24#   for a subtree within the makefile rooted therein
25#
26#DEFINES +=
27
28#####
29# Recursion Magic - Don't touch this!!
```

driver文件夹下Makefile

(4) 三层 Makefile 文件在使用时，具体修改步骤如下：

①修改第一层 Makefile，结合实际的应用开发需求，确定是云端升级方式还是非云端升级方式；（后期也可随意更改）

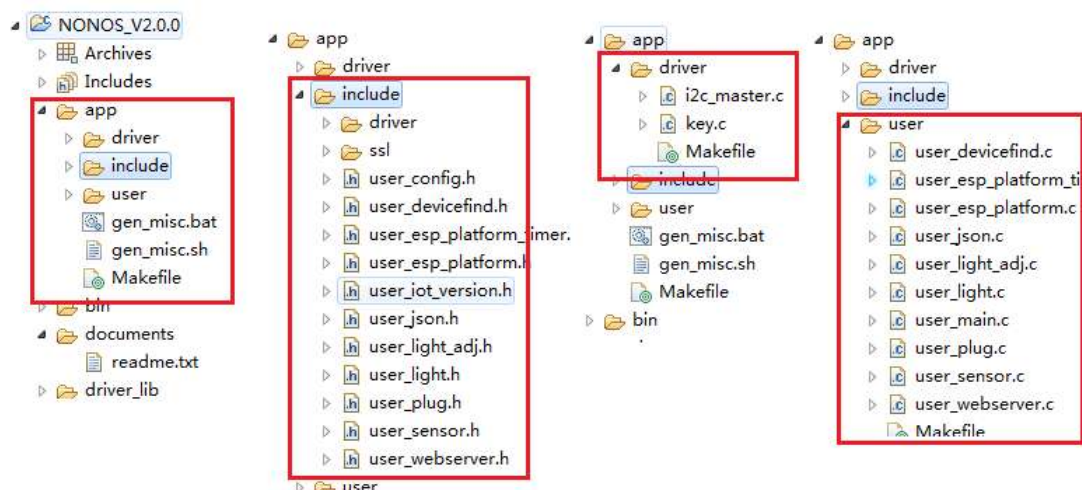
②建立 app 程序文件夹下的源程序分类子文件夹（如 driver、user）；（后期也可随需要增加）

③修改第三层源程序分类子文件夹下 Makefile，指定该文件夹下的源代码生成文件\*.a；

④结合 app 程序文件夹下的源程序分类子文件夹和相应的 Makefile 中生成文件，修改第二层 Makefile 中的 SUBDIRS 和 COMPONENTS\_eagle.app.v6。

### 3、app 程序文件夹结构说明

如上述工程模板，其中 app 文件夹下包括三个文件夹 driver、include、user，其中 driver 文件夹用于存放驱动相关的源程序 (\*.c) 文件（来源于工程目录下的 driver\_lib 文件夹），user 文件夹用于存放用户程序源文件 (\*.c) 文件，include 文件存放源程序需要引用的头文件 (\*.h)，文件夹目录结构如下：



在 `include` 文件夹下,其目录下首先存放的为为用户程序文件夹(`user`)中引用的头文件,在该文件夹下建立了子文件夹 `driver` 用于存放驱动程序文件夹(`driver`)中引用的头文件,此结构不建议更改,若需要增加新的文件夹时,在 `app` 目录下建立对应的文件夹用于存放源程序 (\*.c) 文件,同时在 `include` 下建立同样的文件夹用于存放相应的头文件 (\*.h)。特别是关于 `include` 文件夹下的 `user_config.h` 等部分特殊文件,不建议修改存放位置,因其在官方 SDK 驱动文件中已建立对应的引用,修改会导致较多的连锁修改,对后其对 SDK 进行版本更新时文件比较带来麻烦。

## 2.3 ESP8266 SDK 开发流程

### 2.3.1 工程编译

在此介绍开发流程时,使用对官方的 `example` 中 `IoT_Demo` 程序直接进行编译,通过在工程文件夹右键选择 `Build Project` 实现对工程文件的编译,编译成功后,在 `console` 框中会显示如下对应的信息,表明编译成功。

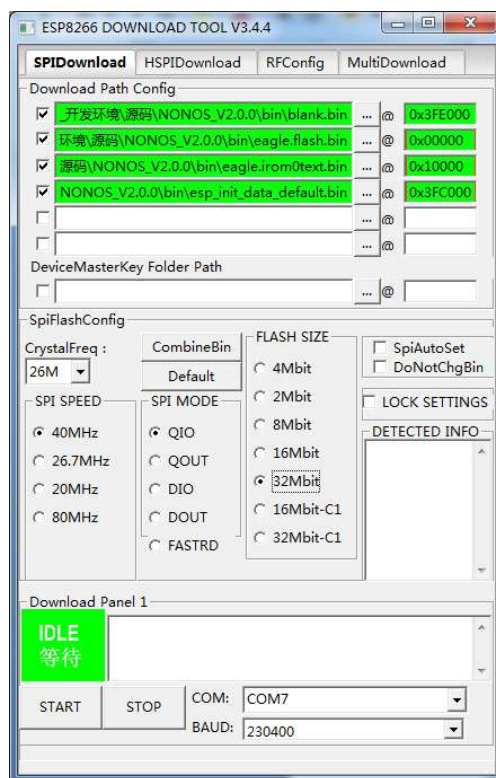
```
!!!
No boot needed.
Generate eagle.flash.bin and eagle.irom0text.bin successfully in folder bin.
eagle.flash.bin----->0x00000
eagle.irom0text.bin---->0x10000
!!!
make[1]: Leaving directory '/cygdrive/e/回清?回清?回清?回清?/回清?回清?回清?回清?/01_回清?回清?回清?回清?/NONOS_V2.0.0/app'
22:57:48 Build Finished (took 5s.121ms)
```

从以上编译结果显示中,可以得到对应程序文件的烧录地址, `eagle.flash.bin` 烧录地址为 `0x00000`, `eagle.irom0text.bin` 烧录地址为 `0x10000` (`eagle.irom0text.bin` 的烧录地址在 SDK 1.5 版本和 SDK 2. 版本不同, SDK 1.5 中烧录地址为 `0x40000`)

## 2.3.2 程序烧录

程序烧录的步骤包括设置参数、模块烧录 2 个步骤，烧录软件采用为乐鑫官方的 flash\_download\_tools。

(1) 设置参数：启动烧录软件，选择对应的程序参数



4M Flash（编码地址:0~400000）

### 3.1.4. 4096 KB Flash

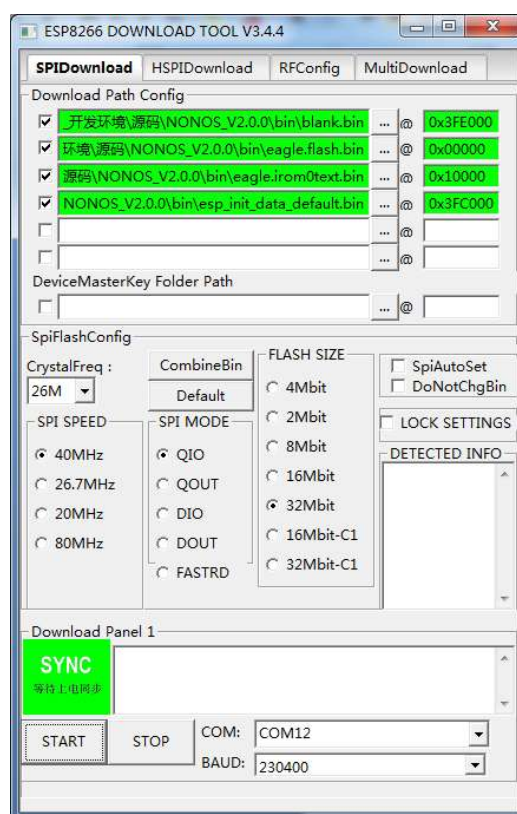
bin	Address	Description
master_device_key.bin	0x3E000	Obtained from Espressif Cloud by users to get Espressif Cloud service
esp_init_data_default.bin	0x3FC000	Contains default RF parameters provided in SDK
blank.bin	0x3FE000	Contains default system parameters provided in SDK
eagle.flash.bin	0x000000	Compiled from SDK
eagle.irom0text.bin	0x400000	Compiled from SDK

注：参考文档 3A\_ESP8266\_Flash\_tool\_user\_manual\_EN\_v1.0

## 3.2.4. 4096 KB Flash

bin	Address	Description
master_device_key.bin	0x3E000 (suggest to revise)	Obtained from Espressif Cloud by users to get Espressif Cloud service. Located in user parameter area. The default address in IOT_Demo is 0x3E000, which can be changed by the user. If selecting 4 in STEP 5 during the compilation, it is suggested to change the address to 0x7E000. If selecting 6 in STEP 5 during the compilation, it is suggested to change the address to 0xFE000 Refer to <a href="http://bbs.espressif.com/viewtopic.php?f=10&amp;t=305">http://bbs.espressif.com/viewtopic.php?f=10&amp;t=305</a> for details
esp_init_data_default.bin	0x3FC000	Contains default RF parameters provided in SDK
blank.bin	0x3FE000	Contains default system parameters provided in SDK
boot.bin	0x00000	Boot loader provided in SDK. Latest version is recommended.
user1.bin	0x01000	Compiled from SDK
user2.bin	0x81000	Compiled from SDK, does not need to be burned into Flash

(2) 程序烧录: 选对与模块连接对应的串口, 点击 **START** 按钮, 进行烧录等待同步信号状态, 如下图所示:



重启或复位模块, 使模块进入下载模式, 进入烧录过程, 烧录完毕后, 重启或复位模块即可运行模块查看模块运行状态。

注: 下载模式: 【MTDO (GPIO15): 低】 【GPIO0: 低】 【GPIO2: 高】

---

运行模式: 【MTDO (GPIO15): 低】 【GPIO0: 高】 【GPIO2: 高】

---

### 2.3.3 运行测试

由于 ESP8266 的开发无在线调试的功能, 因此在实际开发运行测试时, 主要需要通过串口或网络输出程序相应的运行信息以确认程序的运行状态。在此将建立一个基础应用的工程模板, 该工程模板未实现用户功能函数, 采用 UART 调试的方式, 原有固件会自动启动 WIFI 的相应功能。

在该工程模板中, 主要包括设置 UART 串口的波特率和设置模块的 WIFI 模式:

(1) UART0 波特率设置函数:

```
uart_div_modify(0, UART_CLK_FREQ / BIT_RATE_115200);
```

(2) WIFI 工作模式设置:

```
wifi_set_opmode(STATIONAP_MODE); //  
user_set_softap_config("ESP1234", "1234567890", 1);
```