Espressif IOT SDK:

USER MANUAL

Status	Released
Current version	V0.9
Author	Fei Yu
Completion Date	2014.11.07
Reviewer	JG Wu
Completion Date	2014.11.07

[√] CONFIDENTIAL
[   ] INTERNAL
[   ] PUBLIC

Version Info

Date	Version	Author	Comments / Changes
2013.12.24	0.1	JG Wu	Draft
2014.1.15 ~ 2014.6.19	0.2 ~ 0.6	JG Wu / Han Liu / Fei Yu	Internal modification
2014.7.10	0.7	Fei Yu	Support Cloud Upgrade（OTA）
2014.8.14	0.8	Fei Yu	Add Flash Download Tool
2014.11.07	0.9	Fei Yu	Revised compilation

Disclaimer and Copyright Notice

Information in this document, including URL references, is subject to change without notice.

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

The Wi-Fi Alliance Member Logo is a trademark of the Wi-Fi Alliance.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

Copyright © 2013 Espressif Systems Inc. All rights reserved.

Table of Contents

1. Foreword

This manual mainly introduces how to use ESP8266-based SDK for Internet of Things, including virtual machine installation, development tool usage, SDK software development kit etc.
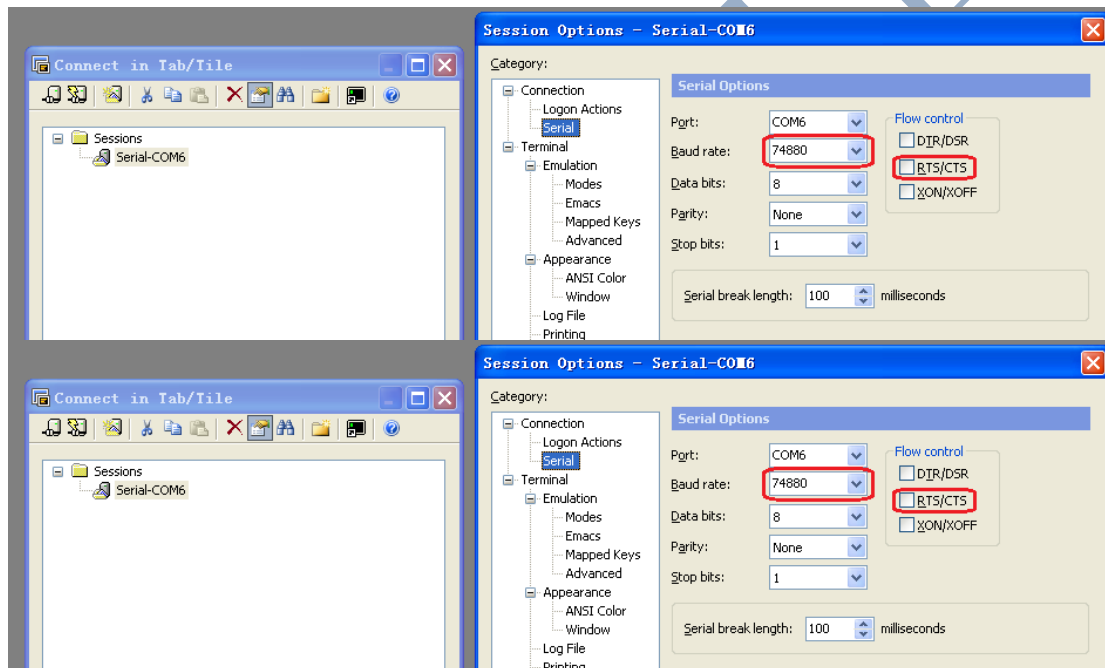
2. Development Tools

Use download tool to download the firmware to flash, use serial port tool to print logs to debug.
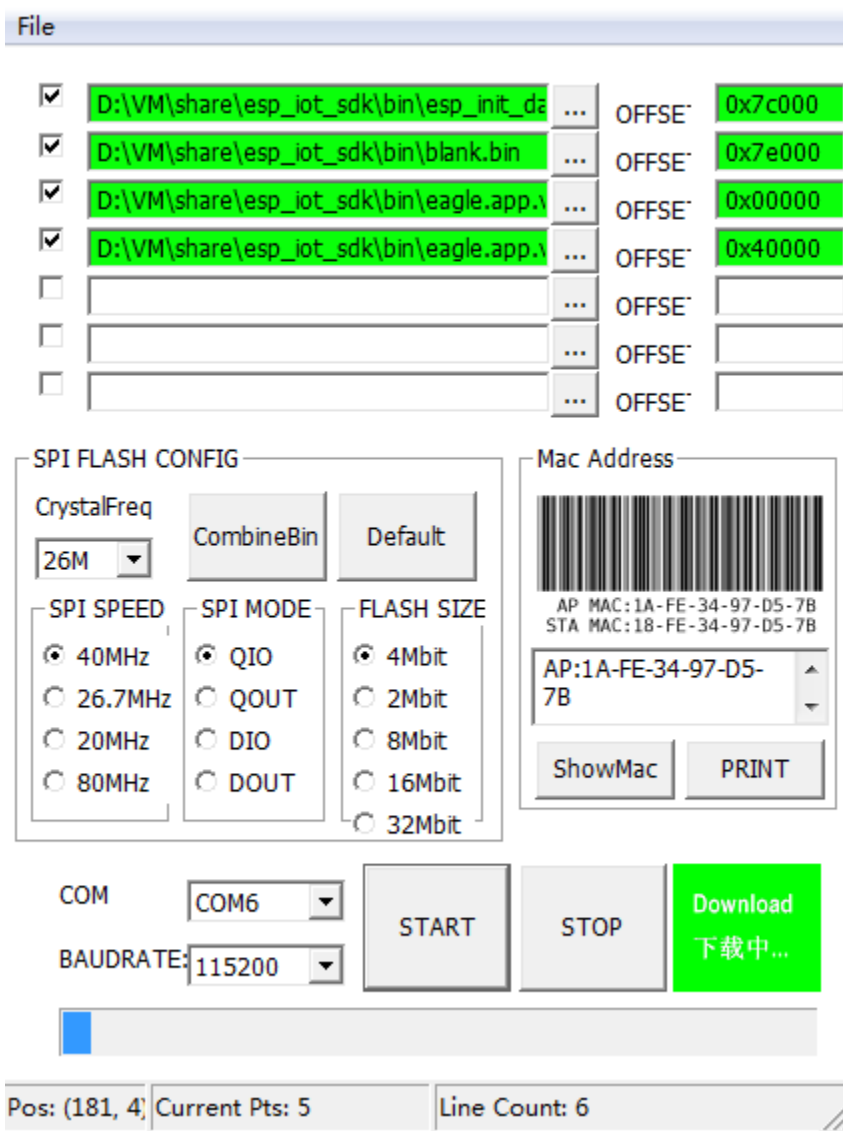
2.1. Serial Port Tool – SecureCRT

Here use SecureCRT as an example of serial port tool, in fact, you can use any other serial port tool to debug.

ESP8266 module adopts 74880 baud rate which can be set in SecureCRT.

2.2. Download Tools - FLASH_DOWNLOAD_TOOLS

Espressif provides the tool "ESP_FLASH_DOWNLOAD" for users to burn several bin files altogether at once, and download several complied *.bin files at a time into the SPI Flash on the ESP8266 motherboard.

"ESP_FLASH_DOWNLOAD" introduction：

（1） Bin-Select Area: Choose bins to burn，and burn them in corresponding address.

（2） SPI FLASH CONFIG: Set config of spi flash. "CombineBin" merges all bins selected above to one (target.bin). "Default" reset to the default config.

（3） Mac Address：Mac address of ESP8266.

Also set the jumper on the motherboard as MTDO：0，GPIO0：0，GPIO2：1, then it will enter download mode. Steps are as follows:

(1) See the red boxes in the picture above, select the bin file to be burned ->fill in the

path ->check burning options.

(2) Set COM port and baud rate.

(3) Click "START" to start downloading.

(4) After the downloading, disconnect the power for the motherboard, and change the jumper into operation mode. Re-connect the power for operation. Set the jumper on the motherboard as MTDO：0，GPIO0：1，GPIO2：1 for operating mode.

Note: please disconnect the power when setting the jumper.

2.3. NetAssist

NetAssist is used to test TCP and UDP.

2.4. Postman

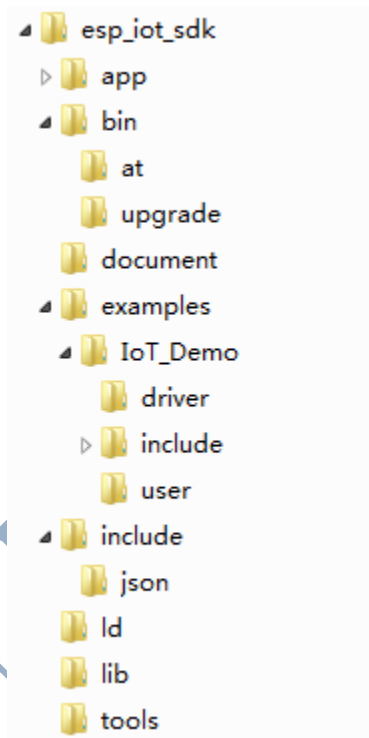Chrome plug-in is used to test REST-structured web service.

2.5. Tomcat

Web application server, used to store updates.

3. SDK Software Package

3.1. Directory Structure

All header files, library files and compilation files needed for secondary development are included in the SDK software package. See the picture below for directory structure:


Detailed description:

1. The "app" folder is the main working folder, we need to copy source codes to this folder to compile.

2. "bin" folder stores the bin files downloaded into the Flash, "at" folder stores the bin files that support AT+ instructions and "upgrade" folder stores the bin files that support cloud update.

3. "examples" folder stores SDK examples, we need to copy the source code here (all files in the IoT_Demo folder) to "app" folder;

4. "include" folder stores the header files pre-installed in the SDK, which may include relevant API functions and other definitions. Users can use them

directly and do not need to change anything;

5. "ld" folder stores the files needed for SDK software link. Users can use them directly and do not need to change anything;

6. "lib" folder stores the library files needed for SDK compilation;"tools" folder stores the tools needed for generating bin files. Users can use them directly and do not need to change anything.

3.2. Compiling and Burning into Flash

When compiling, please remember to copy the sub-folders in the esp_iot_sdk\examples\IOT_Demo to esp_iot_sdk\app.
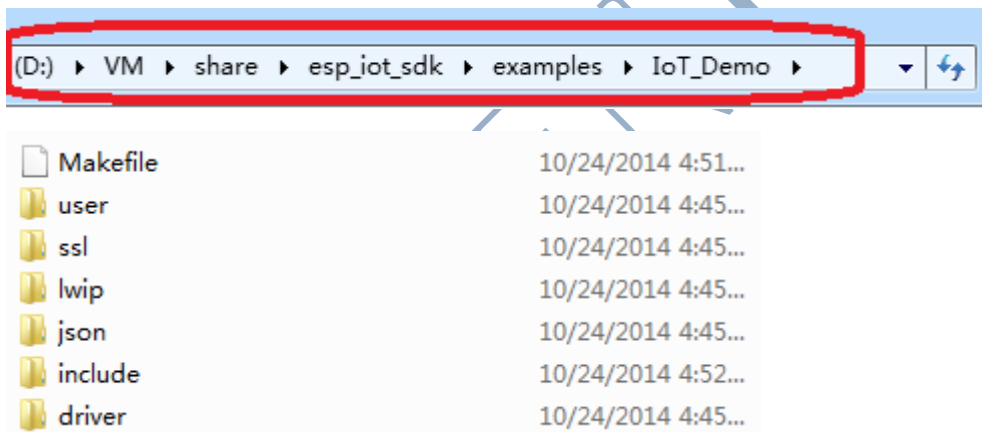
Copy all files in the picture above to esp_iot_sdk\app to compile.
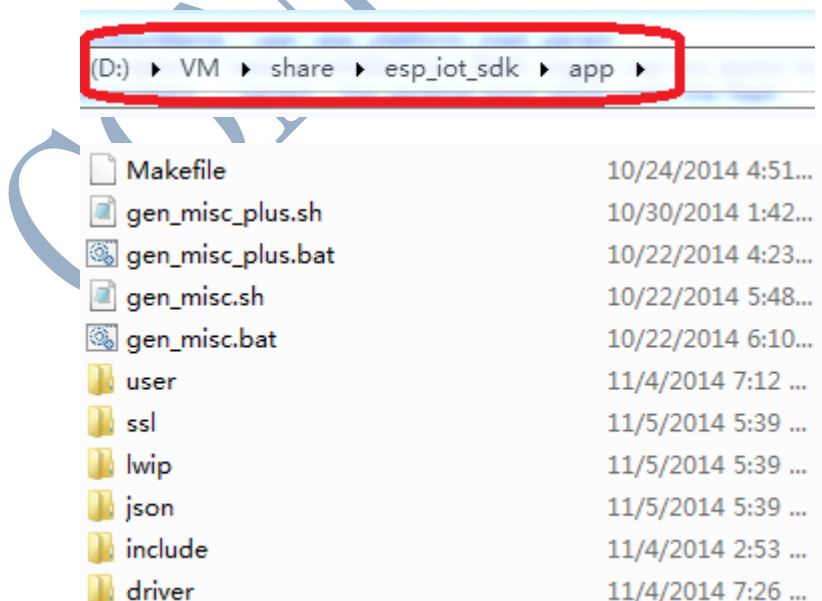
3.2.1. **Version that does not support Cloud Update（OTA）**

esp_iot_sdk_v0.7 and previous versions do not support cloud update.

Compile :   ./gen_misc.sh

3.2.1.1. **Bin file descriptions**

➢ blank.bin, provided in SDK; to be burned to 0x7E000

➢ eagle.app.v6.flash.bin, compiled by the steps said above; to be burned to 0x0000

➢ master_device_key.bin, obtained from Espressif Cloud Server; to be burned to 0x3E000

➢ eagle.app.v6.irom0text.bin, compiled by the steps said above; to be burned to 0x40000

➢ esp_init_data_default.bin, provided by Espressif; stores default parameter values and to be burned to 0x7c000.

Note:

1. It is not necessary to burn blank.bin every time and it is only necessary for SDK update or clearing of WIFI configuration

2. It is not necessary to burn master_device_key.bin every time and it is only necessary for initial write-in and revision of master_device_key

3. Normally, it is only necessary to burn these 2 bins
   eagle.app.v6.flash.bin->eagle.app.v6.irom0text.bin.

3.2.2. **Version that support Cloud Update（OTA）**

esp_iot_sdk_v0.8 and later versions support cloud update and are compatible

with previous compilation and burning methods.

Please refer to document ''Firmware update through cloud server" for details about cloud update.

Compilation Steps:

(1) Run "./gen_misc_plus.sh 1" to generate user1.bin at ''\esp_iot_sdk\bin\upgrade"

(2) Run "make clean" to clean up all previous compilation

(3) Run "./gen_misc_plus.sh 2" to generate user2.bin at "\esp_iot_sdk\bin\upgrade"

3.2.2.1. Bin file descriptions

- blank.bin; provided in SDK and to be burned to 0x7E000;

- esp_init_data_default.bin, provided by Espressif; stores default parameter values and to be burned to 0x7c000.

- boot.bin, provided in SDK and to be burned to 0x00000;

- user1.bin, compiled by the steps said above and to be burned to 0x01000;

- user2.bin, compiled by the steps said above and to be burned to 0x41000;

- master_device_key.bin, applied for through Espressif server and to be burned to 0x3E000;

Note:

For future updates, please upload user1.bin and user2.bin to the server and the server will send update information to users. If users choose to update, then the device will select and download user1.bin or user2.bin, whichever is necessary for cloud update.