# ESP8266
# FOTA Guide

Version 1.7

# About This Guide

This document explains how to upgrade ESP8266 firmware over Espressif Cloud through Wi-Fi.

The document is structured as follows.

| Chapter | Title | Content |
|---|---|---|
| Chapter 1 | Overview | Introduction to two firmware update modes: FOTA (Firmware Over the Air) and Non-FOTA. |
| Chapter 2 | Usage Guide | Introduction to firmware updates on Espressif Cloud, using ***ESP8266_NONOS_SDK\IoT_Demo*** as an example. |
| Chapter 3 | FOTA Mechanism | Details on FOTA implementation and demo applicaton. |
| Appendix I | Firmware Versioning Rules | Introduction to the naming rules of the firmware version when using ***ESP8266_NONOS_SDK\IoT_Demo***. |

## Release Notes

| Date | Version | Release Notes |
|---|---|---|
| 2016.04 | V1.6 | Initial release. |
| 2016.08 | V1.7 | Major revision. |

# Table of Contents

# 1. Overview

ESP8266 provides two main updating modes of the program firmware, specified as follows:

- Non-FOTA mode does not support downloading the latest version of the ESP8266 firmware from Espressif Cloud through Wi-Fi.
- FOTA (Firmware Over the Air) mode supports firmware download and upgrade to the latest version of the ESP8266 firmware from Espressif Cloud through Wi-Fi.

**Table 1-1. Comparison between FOTA and Non-FOTA**

| Firmware upgrade mode | Main program firmware | Compiling difference |
|---|---|---|
| Non-FOTA | ***eagle.flash.bin*** | Select Non-boot mode in STEP 1 in ***Section 2.1.2***, while compiling firmware. |
| | ***eagle.irom0text.bin*** | |
| FOTA | ***boot.bin*** | Select boot mode in STEP 1 and ***user1.bin*** in STEP 2, while compiling firmware. |
| | ***user1.bin*** | |

📖 ***Note:***

*For more information on compiling and burning, please refer to ESP8266-SDK Getting Started Guide.*

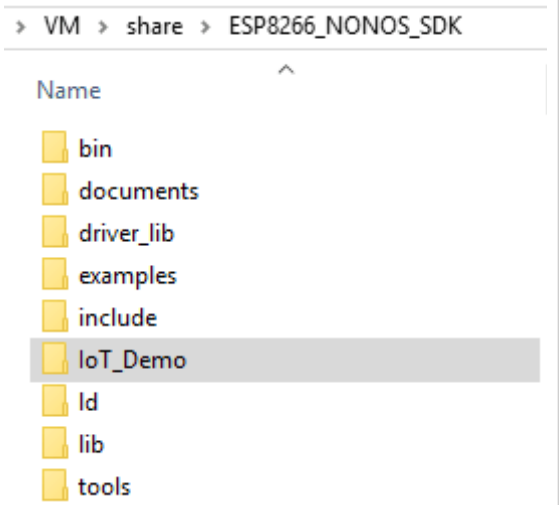
# 2. Usage Guide

This chapter uses ***IoT_Demo of ESP8266_NONOS_SDK*** as an example to illustrate the FOTA process.

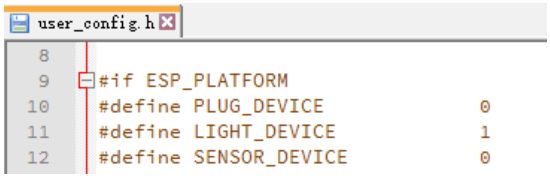
## 2.1. Compiling the Firmware

### 2.1.1. Modifying *IoT_Demo*

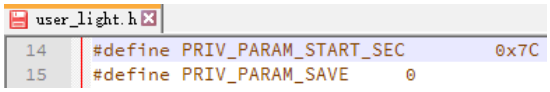
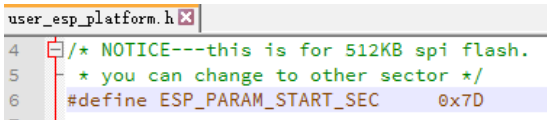
1. ESP8266 SDK download link:
   *http://www.espressif.com/support/download/sdks-demos*

| Step | Result |
|---|---|
| • Take ***ESP8266_NONOS_SDK_V2.0.0_16_07_19*** as an example. Users should download and unzip it.<br>• As shown in the screenshot 👉, copy ***ESP8266_NONOS_SDK\examples\IoT_Demo*** (to be compiled) to ***\ESP8266_NONOS_SDK***'s root directory. |  |

2. ***IoT_Demo*** provides demo application programs for three devices, Smart Light, Smart Plug and Sensor. From these three, Sensor does not support FOTA. The default device type is Smart Light.

| Step | Result |
|---|---|
| • Enable device type in ***ESP8266_NONOS_SDK \IoT_Demo\include\user_config.h.***<br>• As shown in the screenshot 👉, Smart Light is taken as an example.<br>⚠ ***Note:***<br>*Please enable only one device type to debug every time.* |  |

3. Modify the user parameter area location according to the actual flash size of the ESP8266 hardware module.

| Step | Result |
| --- | --- |
| As shown in the screenshot 👉, take 2048 KB flash and 512+512 map as an example, and modify the value of ***#define PRIV_PARAM_START_SEC*** in ***ESP8266_NONOS_SDK\IoT_Demo\include\user_light.h***.<br><br>💬 ***Note:***<br><br>If a *Smart Plug* device type is used, modify the value of ***#define PRIV_PARAM_START_SEC*** in ***user_plug.h***. |  |
| As shown in the screenshot 👉, take 2048 KB flash and 512+512 map as an example, and modify the value of ***#define ESP_PARAM_START_SEC*** in ***ESP8266_NONOS_SDK\IoT_Demo\include\user_esp_platform.h***.<br><br>💬 ***Note:***<br><br>Modify the same address if a *Smart Plug* device type is used. |  |

Different flash maps correspond to different modified locations in header files, as shown in Table 2-1.

**Table 2-1. Modifying the Field in include File (Unit: kB)**

| Default value (512) | Modified Value | | | | |
| --- | --- | --- | --- | --- | --- |
| | 1024 | 2048 (512+512) | 2048 (1024+1024) | 4096 (512+512) | 4096 (1024+1024) |
| 0x3C | 0x7C | 0x7C | 0xFC | 0x7C | 0xFC |
| 0x3D | 0x7D | 0x7D | 0xFD | 0x7D | 0xFD |

### 2.1.2. Compiling *IoT_Demo*

Figure 2-1 shows the process of compiling ***ESP8266_NONOS_SDK\IoT_Demo***. For more details, please refer to *ESP8266-SDK Getting Started Guide*.

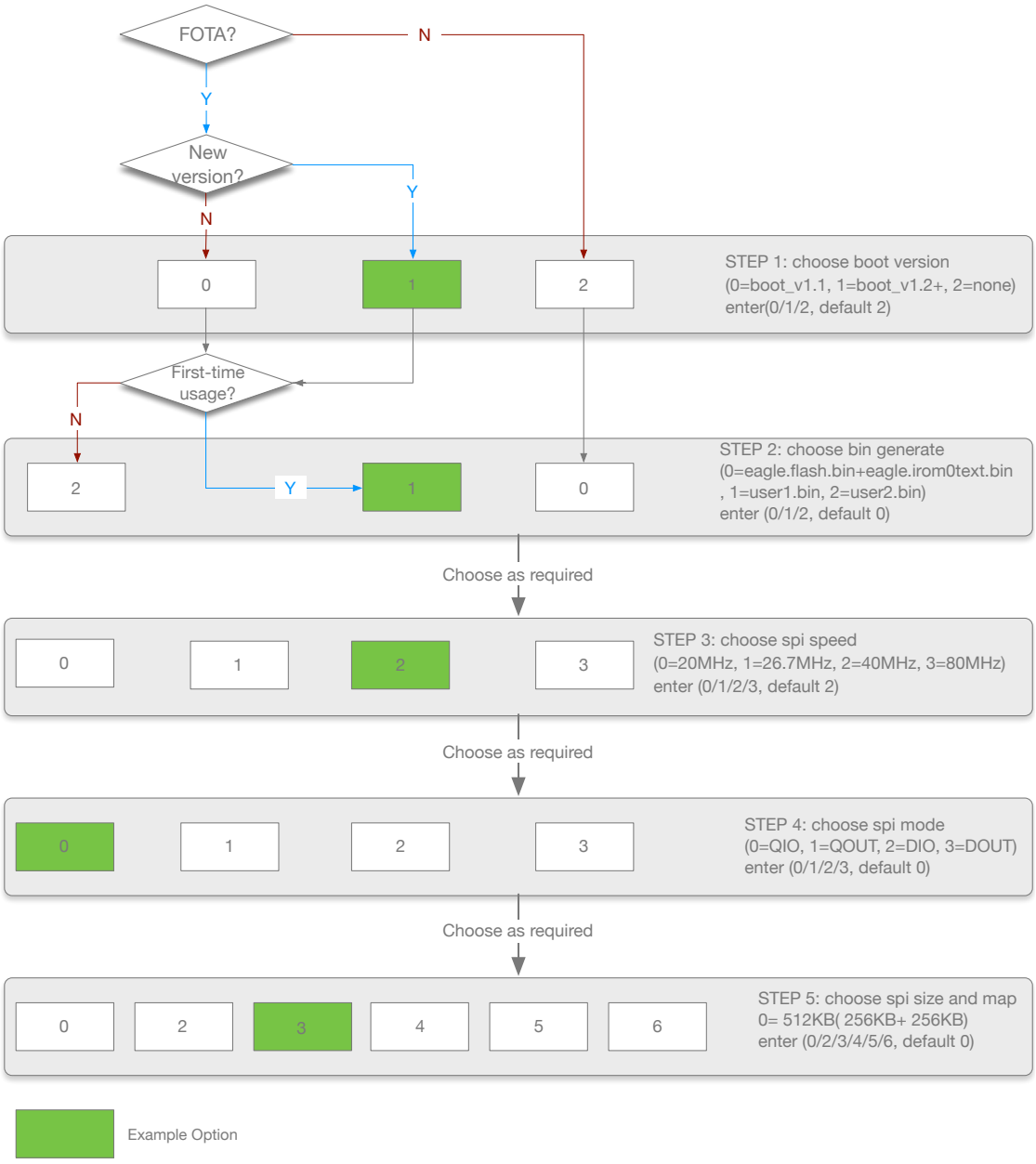



**Figure 2-1. Compiling Process**

> ⚠ ***Notes:***
>
> - ***user1.bin*** *and* ***user2.bin*** *are generated by compiling the same application code, but make sure you choose 2 in STEP 2 when generating* ***user2.bin****. Developers can see more details below*:
>
>  - *Compile the application to generate* ***user1.bin***
>  - *Call* `make clean` *to clear the temporary files generated from the last step*
>  - *Compile the application code to generate* ***user2.bin*** *(make exactly the same choices in the compiling process except for STEP 2).*
> - *The difference between* ***user1.bin*** *and* ***user2.bin*** *is that each is stored in a different location in flash.*
> - *By default, burn* ***user1.bin*** *to flash and upload* ***user1.bin*** *and* ***user2.bin*** *to cloud for repeating downloading and upgrading.*
> - *In Figure 2-1, the example options are marked in green. Users can select one option as required.*
> - *Only* ***sdk_v1.1.0 + boot 1.4 + flash download tool_v1.2*** *and later versions support compiling option 5 and option 6 in STEP 5.*

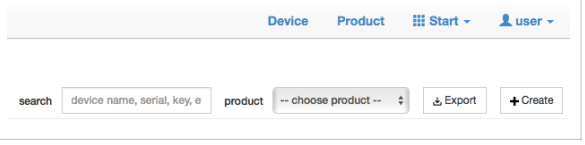
## 2.2. Downloading the Firmware

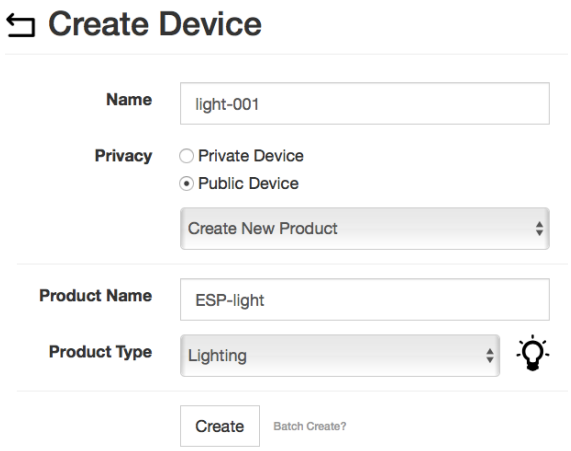
### 2.2.1. *master_device_key.bin*

***master_device_key*** is the device ID. Espressif Cloud Server assigns it automatically when the developer builds a smart device on it. Each ID is unique. The device can get Espressif Cloud services with this ID.

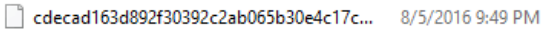
> 📖 ***Note:***
>
> *Please refer to http://iot.espressif.cn/#/help-en/ when using Espressif Cloud for the first time.*

1. Register an account and log in to Espressif Cloud (*http://iot.espressif.cn/#/*), create a smart light device.

| Step | Result |
|---|---|
| As shown in the screenshot 👉, log in to Espressif Cloud, click "***Device***" and then click "***+ Create***". |  |

| Step | Result |
| --- | --- |
| Create a smart light device, for example:<br>- Name: light-001<br>- Set privacy level to "***Public Device***" which supports sharing.<br>- Product Choice "***Create New Products***"<br>- Product Name: ESP-light<br>- Product Type: Lighting<br>Then click "***Create***" at the bottom.<br>💬 ***Note:***<br>*Developers can define "**Name**" and "**Product Name**" at their will.* |  |
| 1. As is shown in the screenshot 👉, a new device page will show up after the creation is finished.<br>2. Developers can see the Master Device Key value on the device page. |  |

2. Export ***master_device_key.bin*** from Espressif Cloud.

| Step | Result |
| --- | --- |
| As shown in the screenshot 👉, click "***Download Key BIN***", which is at the lower right corner of the "***light-001***" page. |  |
| As is shown in the screenshot 👉, the ***master_device_key.bin*** in "***light-001***" will be downloaded.<br>- The name of the bin file is the same as the Master Device Key value of "***light-001***". |  |

### 2.2.2. Download Addresses

Table 2-2 lists the download addresses for different flash sizes for FOTA firmware.

**Table 2-2. The Download Addresses for FOTA Firmware (Unit: KB)**

| Binaries | Download addresses for flash of different capacities | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| | 512 | 1024 | 2048 | | 4096 | |
| | | | 512+512 | 1024+1024 | 512+512 | 1024+1024 |
| *master_device_key.bin* | 0x3E000 | 0x7E000 | 0x7E000 | 0xFE000 | 0x7E000 | 0xFE000 |

| Binaries | Download addresses for flash of different capacities | | | | | |
|---|---|---|---|---|---|---|
| | 512 | 1024 | 2048 | | 4096 | |
| | | | 512+512 | 1024+1024 | 512+512 | 1024+1024 |
| *blank.bin* (Partition 1) | 0x7B000 | 0xFB000 | 0x1FB000 | | 0x3FB000 | |
| *esp_init_data_default.bin* | 0x7C000 | 0xFC000 | 0x1FC000 | | 0x3FC000 | |
| *blank.bin* (Partition 2) | 0x7E000 | 0xFE000 | 0x1FE000 | | 0x3FE000 | |
| *boot.bin* | 0x00000 | | | | | |
| *user1.bin* | 0x01000 | | | | | |

Table 2-3. FOTA Firmware Description

| Binaries | Description |
|---|---|
| *master_device_key.bin* | • Applied from Espressif Cloud for Espressif Cloud services.<br>• Stored in user parameter area, and user's application program define the storage address.<br>• The download address in Table 2-2 is an example set in ***IoT_Demo*** according to ***Section 2.1.1*** |
| *blank.bin* (Partition 1) | • Initializes the ***RF_CAL*** parameter area.<br>• ***user_rf_cal_sector_set*** defines the download address.<br>• The download address in Table 2-2 is an example set in ***IoT_Demo***.<br>• Provided by Espressif and is under the path ***ESP266_SDK\bin***. |
| *esp_init_data_default.bin* | • Initializes other RF parameters area, downloaded at least once.<br>• When ***RF_CAL*** parameter area is initialized, this binary needs to be burnt as well.<br>• Provided by Espressif and is under the path ***ESP266_SDK\bin***. |
| *blank.bin* (Partition 2) | • Initializes system parameter area.<br>• Provided by Espressif and is under the path ***ESP266_SDK\bin***. |
| *boot.bin* | Main program provided by Espressif, under the path ***ESP266_SDK\bin***. |
| *user1.bin* | Main program generated by compiling application program, under the path ***ESP266_SDK\bin\upgrade***. |

## 2.3. IOT Espressif Configuration

1. Refer to ***Section 2.1*** for modifying and compiling ***IoT_Demo***.
2. Refer to ***Section 2.2*** for burning firmware to ESP8266 module.
3. Power on ESP8266 (the default baud rate is 74880), and it will run smart light.
4. Connect ESP8266 module to the router with IOT Espressif (mobile App developed by Espressif).

> 📖 ***Note:***
>
> *To download IOT Espressif, please click http://www.espressif.com/support/download/apks.*

The IOT Espressif configuration process is as follows:

(1) Log in to IOT Espressif with the username and password for Espressif Cloud.

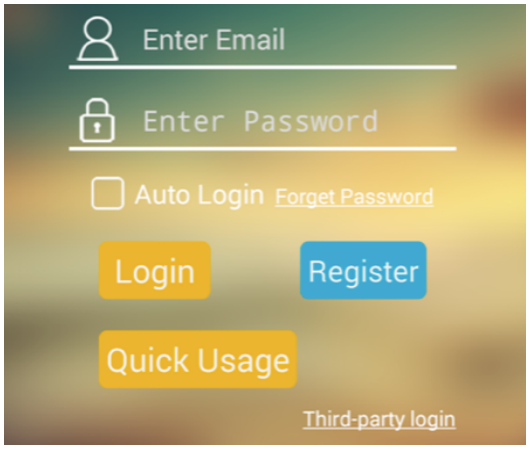



**Figure 2-2. IOT Espressif Login Page**

(2) Click the menu bar at the upper left corner and select "***Add Devices***".

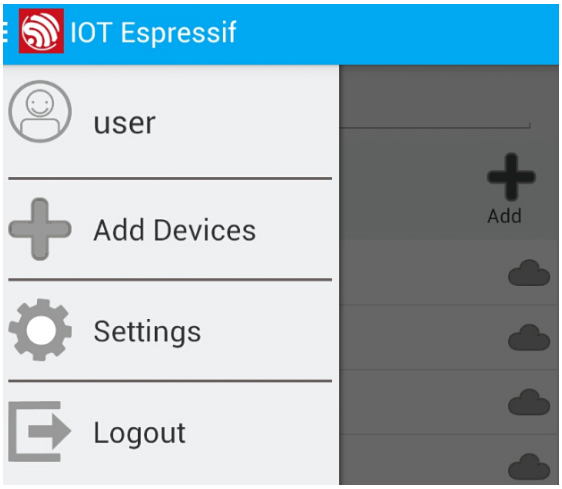



**Figure 2-3. IOT Espressif Menu Bar**

(3) Enter the "***Add Devices***" page and click "***SoftAP Configure***" at the upper right corner.

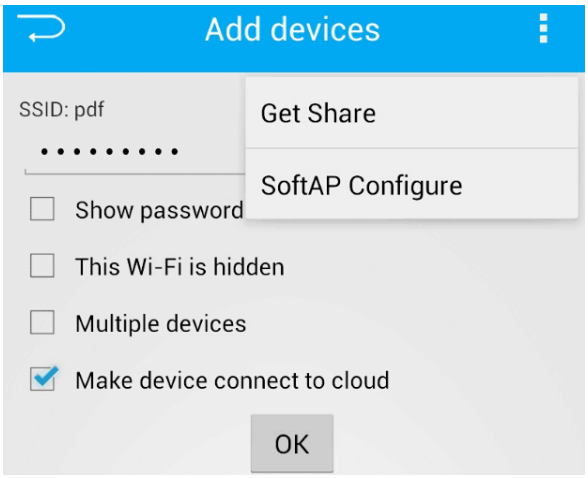



**Figure 2-4. Add devices Page**

(4) The App would list all the nearby APs whose SSIDs begin with "***ESP***". Click the target device and select "***Make device connect to cloud***".

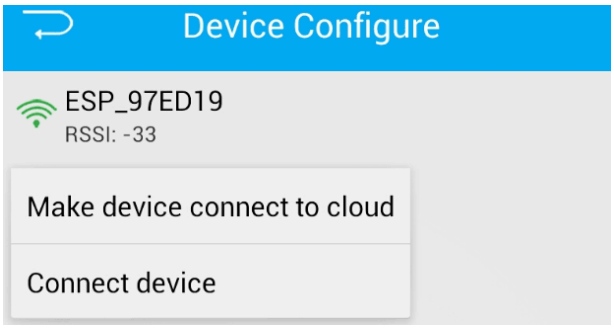



**Figure 2-5. Device Configure Page**

(5) After entering router SSID and password, the ESP8266 device will connect to the router, which will signal the end of the App configuration.

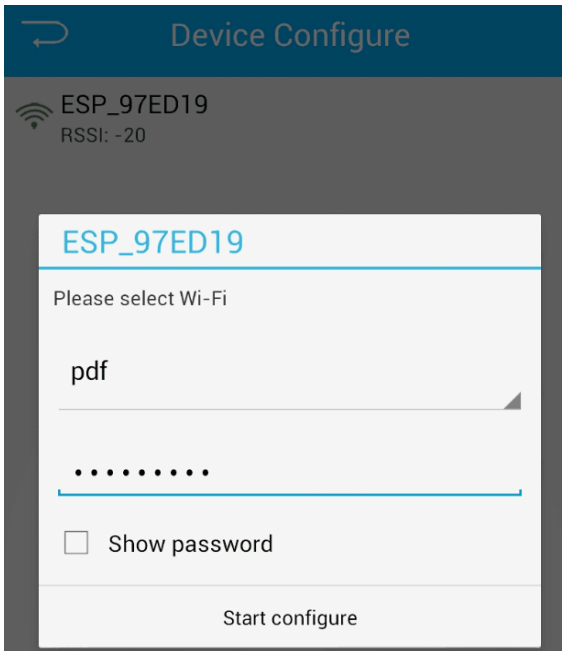



**Figure 2-6. Connecting ESP8266 Device to Router**

5. The ESP8266 device accesses to the internet via the router and activates itself on Espressif Cloud automatically. After verification, the device status will change to "***Activated***" on the right side of the device page after login.

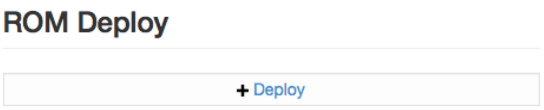
| Step | Result |
| --- | --- |
| • Log in to Espressif Cloud and click "***Device***" at the upper right corner.<br>• Click and enter device "***light-001***" page. The device status changes to "***Activated***" on the right side of the page 👉. | Activated 2016-08-08T12:15:03+08:00<br>Last Active a few seconds ago<br>Device Status developing |

6. ESP8266 Smart Light can receive Espressif Cloud services after verification.

## 2.4. FOTA Process

> 📖 ***Notes:***
>
> - *Espressif Cloud Help document http://iot.espressif.cn/#/help-en/.*
> - *Espressif Cloud API instructions http://iot.espressif.cn/#/api-en/.*

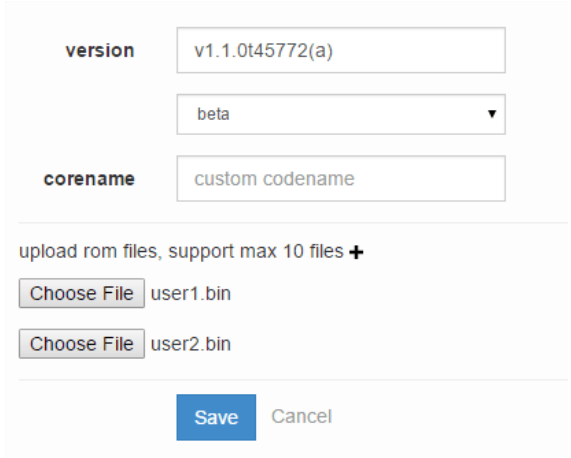
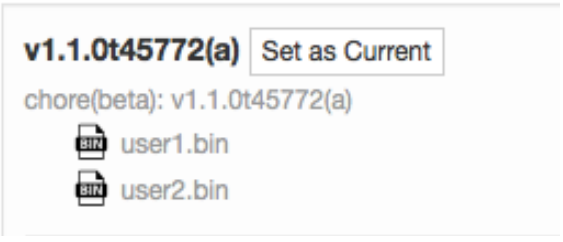
1. Log in to Espressif Cloud (*http://iot.espressif.cn/#/*), click "***Product***".

| Step | Result |
| --- | --- |
| Click the product to upgrade, for example, the product "***ESP-light***" that was created in ***Section 2.2.1.*** |  |
| As is shown in the figure on the right 👉, "***ROM Deploy***" is on the right side of this page. |  |

2. Upload the new version of of ***user1.bin*** and ***user2.bin*** to the Cloud Server.

📖 ***Notes:***

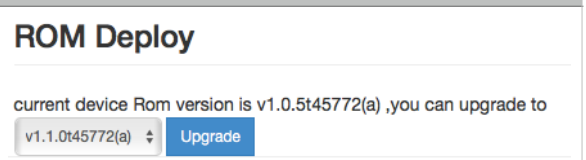
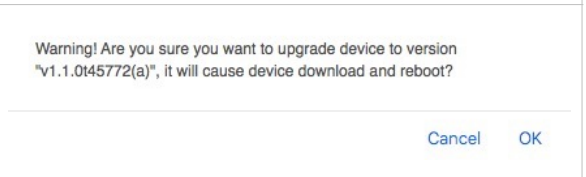
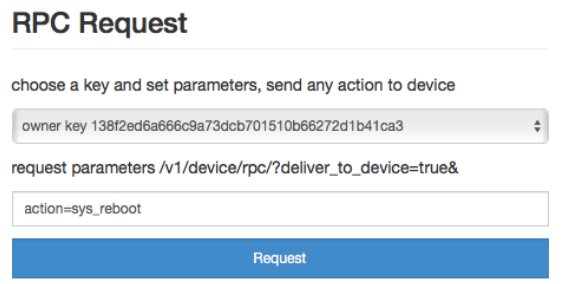
- If ***ESP8266_NONOS_SDK\examples\IoT_Demo*** is used, the firmware versioning should be compliant with the statement in the ***Appendix*** (such as *"v1.1.0t45772(a)"*); otherwise, upgrade will fail.
  - FOTA firmware versioning should be in accordance with the relevant definition in the ***IoT_Demo \include\user_iot_version.h*** *code*. For more details, please refer to the ***Appendix***.
- The firmware versioning rule is not required if the developers implement application by themselves, instead of using ***IoT_Demo***, or, if the application code is based on ***ESP8266_RTOS_SDK***.

| Step | Result |
| --- | --- |
| As is shown in the figure on the right 👉, click "***+ Deploy***" in ***ROM Deploy*** page and upload the new version of firmware.<br>- Version: v1.1.0t45772(a)<br>- Select ***user1.bin*** to upload<br>- Click "**+**" icon<br>- Select ***user2.bin*** to upload<br>- Click "***Save***"<br>⚠️ **Notes:**<br>• *Firmware name must be either **user1.bin** or **user2.bin**.*<br>• *Developers must upload both **user1.bin** and **user2.bin** to the server.* |  |
| As shown in the figure on the right 👉, click "***Set as Current***" after saving the new version of firmware. |  |

| Step | Result |
| --- | --- |
| As is shown in the figure on the right 👉, click "***OK***" in the pop-up tip box to confirm the version setting. |  |

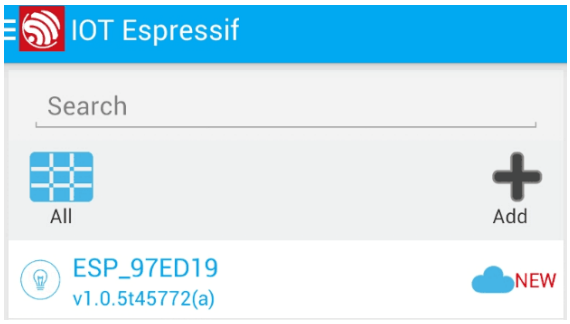
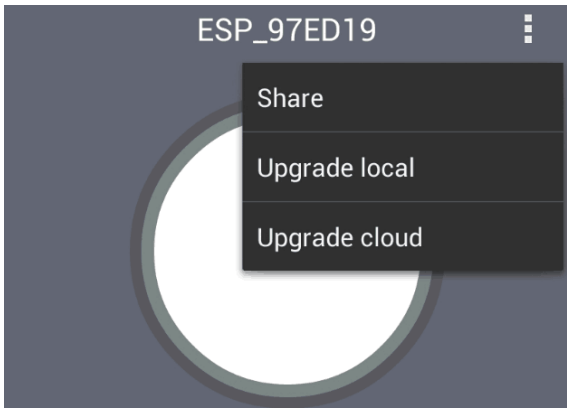
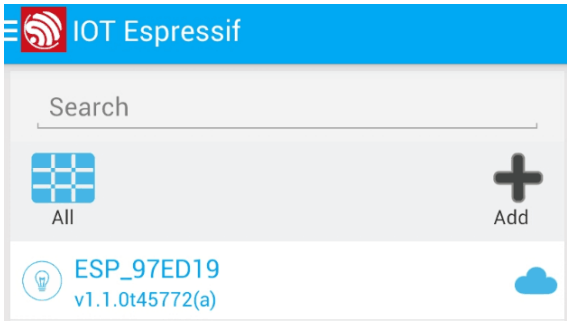
3. There are two ways to upgrade the new version of firmware after it has been uploaded to the Cloud Server.

   (1) Method 1: Upgrade through Espressif Cloud settings.

| Step | Result |
| --- | --- |
| • Enter the device page on Espressif Cloud, such as the "***light-001***" page that was created before.<br>• Select the new firmware version in "***ROM Deploy***" at the lower right corner.<br>• Click "***Upgrade***" as is shown in the figure 👉. |  |
| Click "***OK***" and start the upgrade. |  |
| • After upgrading the new version of firmware to ESP8266 module, insert action="`sys_reboot`" in the RPC Request page, and click "***Request***".<br>• Espressif Cloud will notify ESP8266 to reboot and run the new version of firmware. |  |
| As shown in the figure on the right 👉, after refreshing the device page, the new version name shows up. |  |

   (2) Method .2: Upgrade through the IOT Espressif configuration.

> ⚠ ***Notice:***
>
> *Firmware version cannot be later than v1.4.0 when using the mobile App to upgrade the firmware. Otherwise, upgrade will fail.*

| Step | Result |
| --- | --- |
| As shown in the figure on the right 👉, log in to IOT Espressif. The current version of the target device is v1.0.5. |  |
| Click the target device and select "***Upgrade cloud***" at the upper right corner of the device page. |  |
| After upgrading, the home page of the App shows that the firmware version of the ESP8266 smart light has been upgraded to the new version: v1.1.0t45772(a). |  |

📖 ***Notes:***

- *The device name "**ESP_97ED19**" is the name of **ESP8266 SoftAP** set in the application program (IoT_Demo). If it has not been set yet, then it is "**ESP_XXXXXX**" by default. And "**XXXXXX**" is the last three bytes of the actual MAC address of the device.*
- The ESP8266 device can download the latest firmware from the Cloud server after setting "***Upgrade cloud***".
  - After downloading, the *ESP8266 device* sends a notification of completion.
  - *APP* will notify the *ESP8266* device to restart.
  - Then *APP* will run the new version of firmware after it has received the noti*fication above*.
- If "***Upgrade local***" is selected, *the* App will:
  - download the new version of firmware from server to mobile phone,
  - then push it to the ESP8266 device via LAN (Local Area Network) to upgrade it.

Use this if your network is behind a proxy and internet cannot be accessed directly by the ESP device.

# 3. FOTA Mechanism

## 3.1. Flash Layout

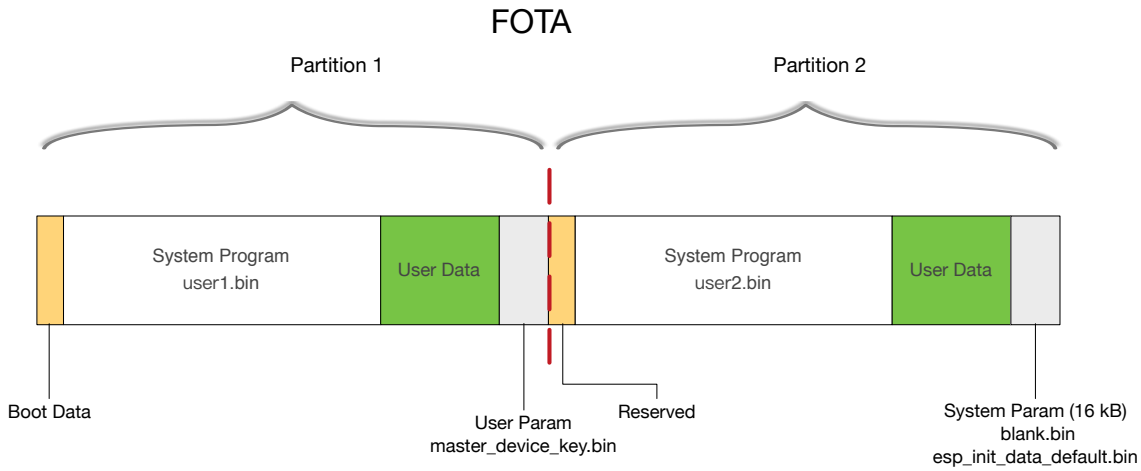



**Figure 3-1. Flash Layout**

**Partitioning Description**

- ***user1.bin*** and ***user2.bin*** are generated by the same application program using different compiling procedures. The binaries are stored in different locations in the SPI Flash. Both of them can run on the system.
- System parameter area stores a flag bit to indicate whether to run ***user1.bin*** or ***user2.bin***.
- ***Boot*** runs first upon initialization, reads the flag bit and decides whether to run ***user1.bin*** or ***user2.bin***. After that, it will read and run the binary from the corresponding address in SPI Flash.

**Example:**

1. Initial state: ***boot.bin*** + ***user1.bin of version 1.0.0*** + ***the flag bit indicates that user1 is running***.
2. Upload ***user1.bin*** and ***user2.bin*** of v1.1.0 to Cloud server.
3. When the ESP8266 smart device is upgrading, it reads the system flag bit first (while user1 is running), and downloads ***user2.bin*** of v1.1.0 from Cloud server to Partition 2 in Figure 3-1.
4. After downloading, users can restart the ESP8266 smart device and update firmware. The ESP8266 device will:
    - modify the flag bit to indicate that user2 is running, and
    - restart (***system_upgrade_reboot***) to run ***user2.bin*** of v1.1.0.

5. For the next upgrade, please repeat Step 2 and download ***user1.bin*** of v1.2.0 to Partition 1 and overwrite the old version. On completion, the device can then reboot and switch to ***user1.bin*** for the upgrade to take effect.

## 3.2. Software Implementation

> ⚠ ***Notice:***
>
> *Developers are not restricted to using Espressif Cloud services to upgrade firmware. They can use other Cloud servers as well through HTTP protocol by implementing the software interfaces provided by ESP8266 SDK.*

### 3.2.1. ESP8266_NONOS_SDK

Upgrade APIs:

- Reference document *ESP8266 Non-OS SDK API Reference*.

Demo application program:

- Refer to ***ESP8266 NONOS SDK\examples\IoT_Demo\user\user_esp_platform.c***. Upgrade is realized by calling ***user_esp_platform_upgrade_begin*** series functions.
- Download link: *http://www.espressif.com/support/download/sdks-demos*

### 3.2.2. ESP8266_RTOS_SDK

Upgrade APIs:

- Reference Document *ESP8266 RTOS SDK API Reference*.

Demo application program:

- Refer to ***ESP8266_IOT_PLATFORM\upgrade***.
- Download link: *https://github.com/espressif/ESP8266_IOT_PLATFORM*

> 📖 ***Note:***
>
> ***ESP8266_IOT_PLATFORM*** *is* a demo application program based on ***ESP8266_RTOS_SDK***.

### 3.2.3. Requirements for Custom Server

If developers use custom cloud server for upgrading instead of Espressif Cloud, the custom cloud server should support the following upgrading process:

1. The ESP8266 smart device sends the HEAD instruction to Cloud Server and quires the length of firmware to download. The HTTP packet header that server responds with is required to have the length information of firmware.
2. The ESP8266 smart device clears the flash sector (spi_flash_erase_sector) into which the new firmware will be written according to the length information obtained in Step 1.

3. ESP8266 smart device sends GET instruction to Cloud Server and downloads the new version firmware from Cloud Server to write into flash.

The custom server must support:

- the function of returning the firmware length information in Step 1, and
- the downloading function in Step 3.

Otherwise, FOTA upgrade cannot be supported.

# I. Appendix - Firmware Versioning Rules

***IoT_Demo*** of ***ESP8266_NONOS_SDK*** specifies the firmware versioning rules. When uploading the firmware to Espressif Cloud, please follow these rules. Otherwise, FOTA will fail.

The versioning rules are not applicable if:

- developers implement applications by themselves instead of using ***IoT_Demo***, or
- the application program code is based on ***ESP8266_RTOS_SDK***.

## I.I. Versioning Rules

- Template: [v|b]Num1.Num2.Num3.tPTYPE([o|l|a|n])
- Example: v1.0.2t45772(a)

**Table I. Firmware Versioning Rules Description**

| v | 1.0.2 | t | 45772 | (a) |
|---|---|---|---|---|
| version type | version value | type flag | device type | FOTA or Non-FOTA |
| variable | variable | fixed | variable | variable |

- Version Type: v or b
  - v: version, formal version
  - b: beta, beta version
  - Corresponds to ***#define VERSION_TYPE*** in ***IoT_Demo\include\user_iot_version.h***.

> ⚠ ***Notice:***
> Espressif Cloud does not support the co-existence of both formal and beta versions of firmware with the same version value. For example, *v1.0.5* and *b1.0.5* cannot co-exist on Espressif Cloud.

- Version value: Num1.Num2.Num3
  - Num: Integer, range of [0, 9].
  - Example: 1.0.5
  - Corresponds to the following definitions in ***IoT_Demo\include\user_iot_version.h***
  - Version value cannot be over 1.4.0. since IOT Espressif currently supports up to 1.4.0.

```
#define IOT_VERSION_MAJOR          1U
```

```
#define IOT_VERSION_MINOR          0U
#define IOT_VERSION_REVISION       5U
```

- Type flag: t
  - t: Type flag, followed by PTYPE, the device type value.
- Device type value: PTYPE
  - PTYPE: The device's ptype (product type) value on Espressif Cloud.
  - Inquire about ptype value on Espressif Cloud: *http://iot.espressif.cn/#/api*
  - Corresponds to ***#define device_type*** in ***IoT_Demo\include\user_iot_version.h***.

**Create Products**

`POST /v1/products/`

Parameters require userkey

| Method | Key | Value |
|---|---|---|
| POST | | |

```
{
  "products": [
    {
      "name": "name",
      "description": "description",
      "serial": "serial",
      "is_private": 1,
      "ptype": 27388,
      "status": 1
    }
  ]
}
```

All the parameters are optional, and if serial is provided, it must be unique. is_private: 1/0 (whether it is private or not), status: 1/2 (under development/deployed)

Multiple types of products are supported, whereas we suggest that product shall subject to one particular type. Below is detail information about the specific types of the products:

Common Sensor ptype: **27388** , img:

- FOTA or Non-FOTA
  - o: online, online upgrade support
  - l: local, local upgrade support
  - a: all supported , support for both online and local upgrade
  - n: none supported, lack of support for upgrade (Non-FOTA)
  - Corresponds to the following definitions of ***IoT_Demo\include\user_iot_version.h***

```
#define ONLINE_UPGRADE     0
#define LOCAL_UPGRADE      0
#define ALL_UPGRADE        1
#define NONE_UPGRADE       0
```

## I.II. Version Value-related Rules

- Espressif Cloud does not support two or more firmware versions that support FOTA (o, l, a) with the same ptype and version value.
  - For example, v1.0.3t45772(o), v1.0.3t45772(l), v1.0.3t45772(a), b1.0.3t45772(l) and b1.0.3t45772(a) are not allowed to co-exist, if b1.0.3t45772(o) also exists.
- Espressif Cloud does not support two or more firmware versions that do not support FOTA (o, l, a) with the same ptype and version value.
  - For example, v1.0.3t45772(n) is not allowed to exist if b1.0.3t45772(n) exists at the same time.
- When Non-FOTA firmware exits, Espressif Cloud does not support two or more firmware versions that support FOTA (o, l, a) with the same version value.
  - For example, if b1.0.3t45772(n) exists, only one version of firmware among b1.0.3t45772(o), b1.0.3t45772(l), b1.0.3t45772(a), v1.0.3t45772(o), v1.0.3t45772(l) and v1.0.3t45772(a) can exist.
- Firmware versions with different ptype can have the same version value.
  - For example, b1.0.3t12335(n) can co-exist with b1.0.3t45772(n).

Espressif IOT Team

*www.espressif.com*