ESP8266 云端升级实现方案

Version 1.6

Espressif Systems IOT Team

Table of Contents

1. 前言

本文主要介绍 ESP8266 基于服务器实现云端升级软件的方法，供 ESP8266 软件开发，或者客户自行搭建服务器，作为参考。

主要背景知识（参考文档"Espressif IoT SDK 使用手册"）：

(1) 不支持云端升级的情况，烧录如下主程序 bin

- `eagle.flash.bin` 烧录到 flash `0x00000` 位置；
- `eagle.irom0text.bin` 烧录到 flash `0x40000` 位置

(2) 支持云端升级的情况，则烧录如下主程序 bin

- `boot.bin` 烧录到 flash `0x00000` 位置；
- `user1.bin` 烧录到 flash `0x01000` 位置；

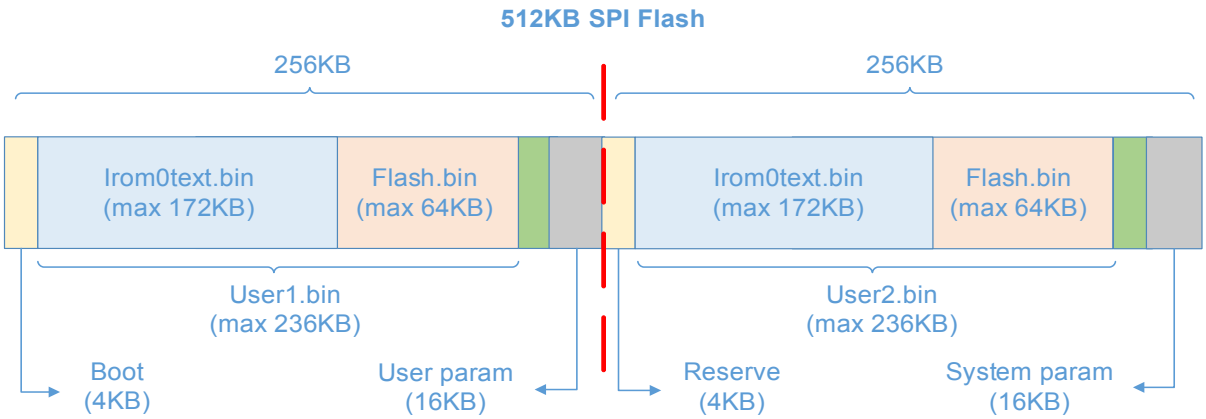
支持云端升级的软件中，`boot.bin` 用于选择运行 user1 还是 user2 （两者的关系由后文详述），而主程序由原本的 `eagle.flash.bin` 和 `eagle.irom0text.bin` 合并为 `user1.bin`（或 `user2.bin`）

2. Flash Layout

本文仅介绍支持云端升级的软件 load region layout 。后文图中绿色区域，均表示 User Data 区域；当程序区（`user1.bin` 或 `user2.bin`）未占满整个空间时，空闲区域均可用于存放用户数据。

2.1. 521KB Flash

非OS esp_iot_sdk 中的 IOT_Demo 以 512KB SPI Flash 为例，如下图：



512KB SPI Flash Layout			
区域	说明	地址	空间大小
Boot	存放 boot.bin	0~4KB	4KB
User APP1	存放 user1.bin (= flash.bin + irom0text.bin)	4KB~240KB	236KB
User param	用户参数区（4 x 4KB）	240KB~256KB	16KB
Reverse	保留空间，使 user2.bin 与 user1.bin 的偏移地址一致（0x01000）	256KB~260KB	4KB
User APP2	存放 user2.bin (= flash.bin + irom0text.bin)	260KB~496KB	236KB
System param	系统参数区（4 x 4KB）	496KB~512KB	16KB

User APP1 和 User APP2 是同一份软件的两个备份，后文分别简称为 user1 和 user2。

System param 区存了一个 flag，标识启动时应当运行 user1 还是 user2。

启动时先运行 Boot，Boot 读取 System param 区中的 flag，判断运行 user1 还是 user2，然后到 SPI Flash 的对应位置去取。

例如，

(1) 初始状态：boot.bin + 版本 v1.0.0 的 user1.bin + System param 区 flag 标志为使用 user1；

(2) 服务器上传软件更新版本 v1.0.1 的 user1.bin 和 user2.bin；

(3) 服务器推送通知，设备读取 flag 当前正使用 user1，则从服务器下载 v1.0.1 的 user2.bin 到 SPI Flash 260KB 之后的空间；

(4) 下载完成后，用户可以选择重启更新，则修改 System param 中的 flag 标志为使用 user2，设备重启，使用 v1.0.1 的 user2 软件。

(5) 再下次升级，则参考步骤2及之后流程。下载 v1.0.2 的 user1 到 SPI Flash 4KB 之后的空间，覆盖之前 v1.0.0 的 user1 软件。

注意:

- 上传时，将新版本的 user1.bin 和 user2.bin 均上传至服务器，由设备自行判断应该下载 user1.bin 还是 user2.bin
- user1.bin 和 user2.bin 是同样的可执行软件，差别仅在于 flash 的存放位置不同
- 依上述流程可知，一般无需烧录 user2.bin ，将 user1.bin 烧录到 flash 中，执行云端升级功能可下载 user2.bin 到 flash

2.2. 1024KB 及以上容量 Flash

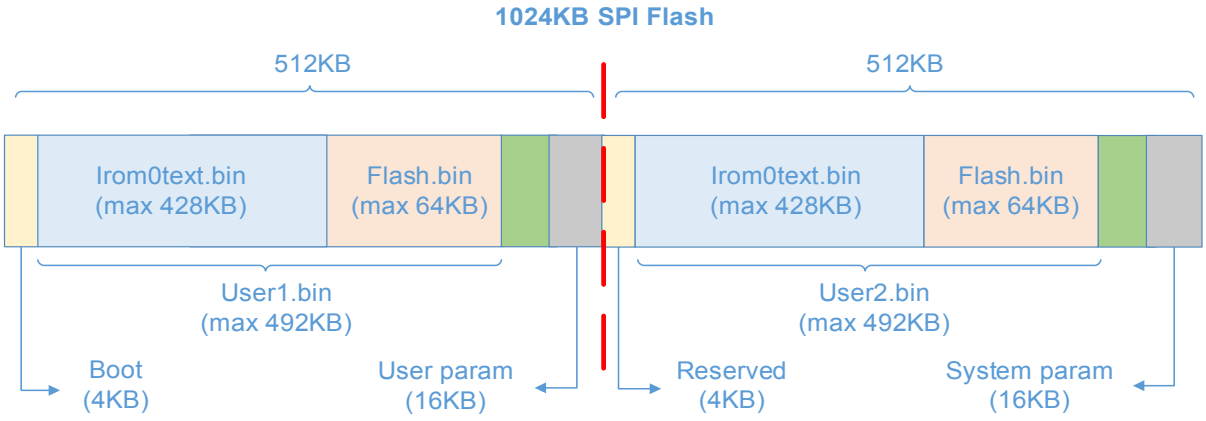
编译时 STEP 1 选择 1 boot_v1.2+，编译两次，分别生成 `user1.bin` 和 `user2.bin`，支持云端升级功能。则 STEP 5 时选择不同 flash size 对应的布局如下。各区域说明与 512KB Flash 一致。

注意：

- “System param” 系统参数区始终位于 flash 的最后 16KB。
- 后文图中绿色区域，均表示 User Data 区域；当程序区（`user1.bin` 或 `user2.bin`）未占满整个空间时，空闲区域均可用于存放用户数据。

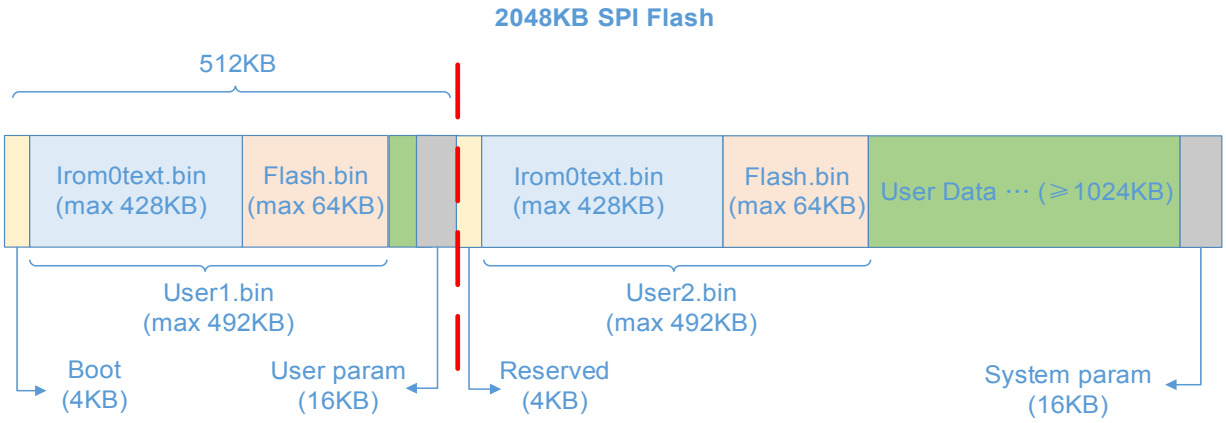
1. 1024KB flash

若编译时 STEP 1 选择 1 ，STEP 5 选择 2 ，则 flash map 如下

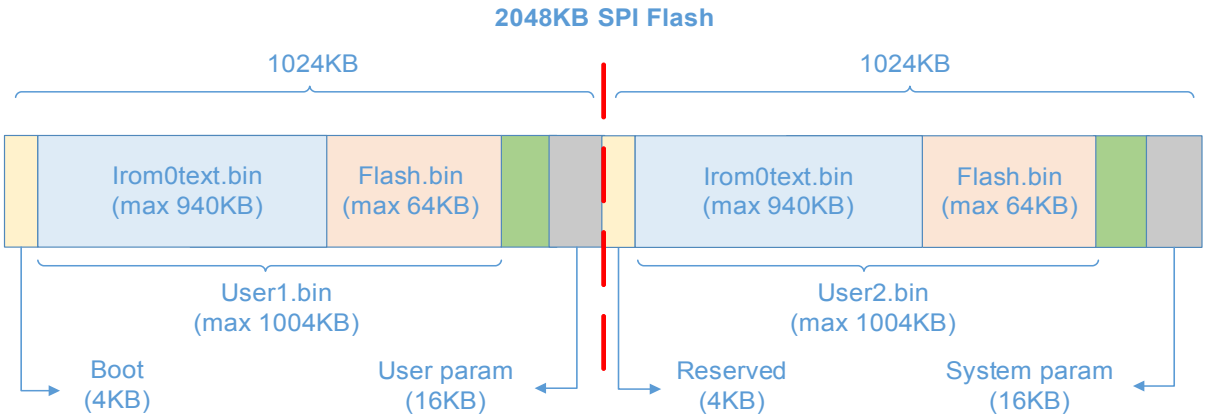


2. 2048KB flash

若编译时 STEP 1 选择 1 ，STEP 5 选择 3 ，仅前 1024KB 为代码区，对半划分为 512KB + 512KB，则 flash map 如下

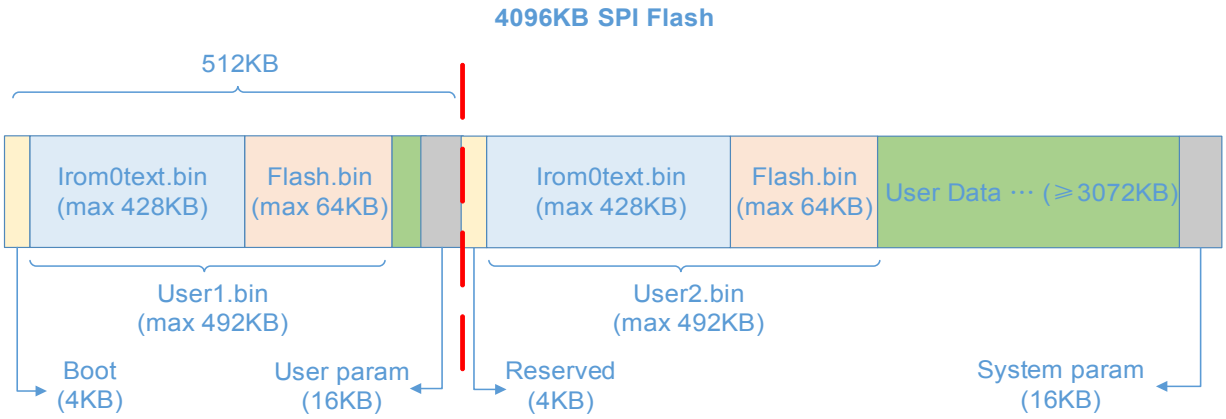


若编译时 STEP 1 选择 1，STEP 5 选择 5，代码区对半划分为 1024KB + 1024KB，sdk_v1.1.0 + boot 1.4 + flash download tool v1.2 及之后的版本支持， flash map 如下

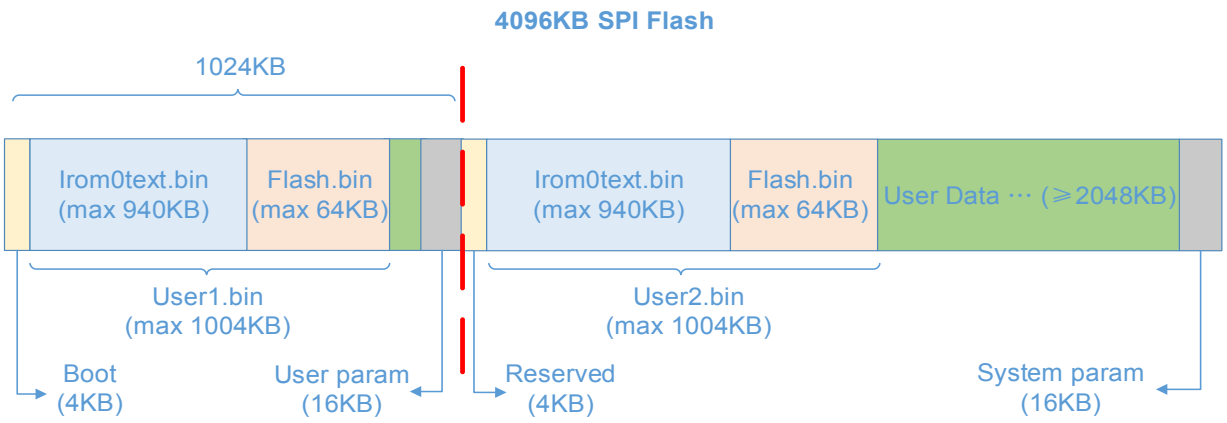


3. 4096KB flash

若编译时 STEP 1 选择 1，STEP 5 选择 4，仅前 1024KB 为代码区，对半划分为 512KB + 512KB，则 flash map 如下

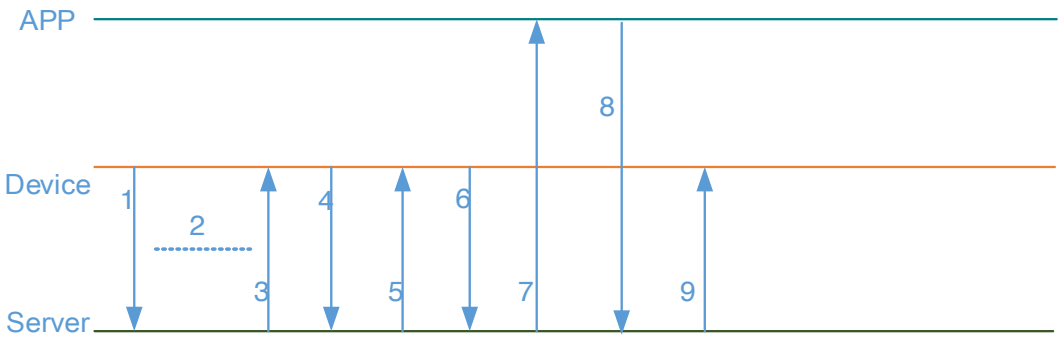


若编译时 STEP 1 选择 1，STEP 5 选择 6，前 2048KB 为代码区，对半划分为 1024KB + 1024KB，在 sdk_v1.1.0 + boot 1.4 + flash download tool v1.2 及之后的版本支持， flash map 如下


4096KB SPI Flash
1024KB
Irom0text.bin
(max 940KB)
Flash.bin
(max 64KB)
Irom0text.bin
(max 940KB)
Flash.bin
(max 64KB)
User Data ⋯ (⩾2048KB)
User1.bin
(max 1004KB)
User2.bin
(max 1004KB)
Boot
(4KB)
User param
(16KB)
Reserved
(4KB)
System param
(16KB)

3. FOTA 升级流程

将更新后的软件（user bin）上传到服务器后，服务器会推送消息升级的条件为服务器上软件的版本高于设备端的版本，升级过程如下图：



1. 设备激活过程中，上报设备的软件版本信息给服务器，服务器将对应设备的版本信息存于数据库。
2. 正常的使用过程。
3. 服务器端上传固件更新，推送升级消息到设备。
4. 设备根据 device key 以及升级路径请求对应固件（user bin）。
5. 设备下载升级对应的固件（user bin）到 SPI Flash。
6. 设备发送升级完成消息到服务器。
7. 服务器将升级完成消息推送给 APP。
8. APP 经由服务器发送反向控制消息，使设备重启运行新版本固件。
9. 服务器将重启消息转发至设备，设备重启运行新版本固件。

以上所有数据流均采用SSL加密，从第4步开始，需要结合 device key 才能获取到服务器的数据，从而保证云端升级过程的安全性。

注意：

由于温湿度计会进入休眠，升级流程将有所不同。IOT_Demo 仅实现了开关一类支持反向控制的设备的云端升级，对于温湿度计这类不支持反向控制的设备的云端升级，会在后续实现。

4. 使用指南

4.1. 如何生成 user1.bin 和 user2.bin

user1.bin 和 user2.bin 可由同一份代码采用不同的编译选项，编译生成；区别在于它们在 flash 中的存放位置不同；两者交替使用，交替更新。

- 在编译器中执行 "./gen_misc.sh"， STEP 1 选择 "1=boot_v1.2+" ，STEP 2 选择 "1=user1.bin" ，其余选项根据模块的实际配置选择；编译完成后，将在"\esp_iot_sdk\bin\upgrade" 路径下生成 user1.bin ；
- 执行 "make clean" 清除之前编译生成的文件；
- 执行 "./gen_misc.sh"， STEP 1 选择"1=boot_v1.2+" ， STEP 2 选择 "2=user2.bin"，其余选项根据模块的实际配置选择；编译完成后，将在"\esp_iot_sdk\bin\upgrade" 路径下生成 user2.bin

注意:

详细的编译烧录说明，请参考文档" 2A-ESP8266__IOT_SDK_User_Manual "。

4.2. 初次烧录

bin	烧录地址	说明
master_device_key.bin	0x3E000	用户在 Espressif Cloud 申请，依此享受 Espressif 云端服务，存放于用户参数区，IOT_Demo 中设置为 0x3E000，用户可自行更改。建议使用 1MB 及以上容量 flash 时，参考 BBS 修改，烧录到 0x7E000 http://bbs.espressif.com/viewtopic.php?f=10&t=305
blank.bin	0x7E000	初始化系统参数区，由 Espressif 在 SDK 中提供
boot.bin	0x00000	启动程序，由 Espressif 在 SDK 中提供，建议使用最新版本
user1.bin	0x01000	主程序，编译代码生成

user2.bin 无需烧录到 flash，可由后续云端升级下载。

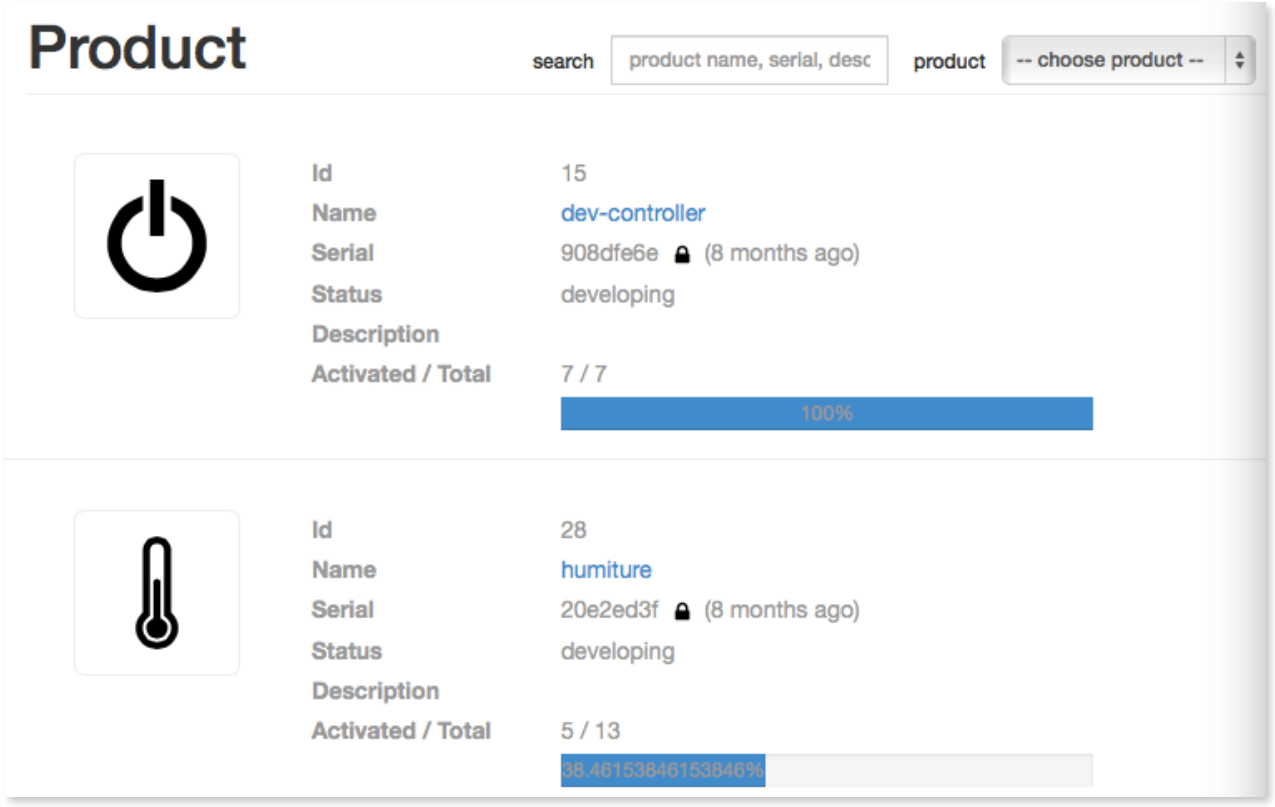
4.3. Website 操作说明

注意:

Espressif Cloud 帮助文档 http://iot.espressif.cn/#/help-zh-cn/

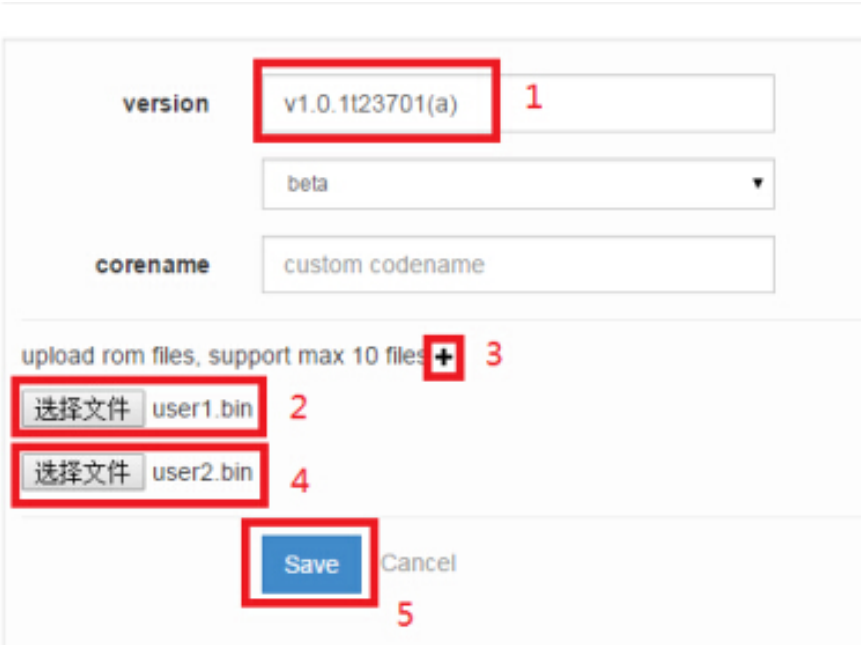
更多云端操作的介绍，请参考文档 "Espressif Cloud introduction".

(1) 用户名、密码登陆网站 http://iot.espressif.cn/#/ ，点击“产品管理，将列出您所有的产品



(2) 选择需要升级的产品，例如，对上述第一项产品 dev-controller 进行软件更新，则点击“dev-controller”，进入产品 dev-controller 后，在页面右侧看到“ROM发布”。

(3) 点击 "+ 发布" 上传固件更新（user1.bin 和 user2.bin），固件版本号命名参考本文附录，形如 "v1.0.1t23701(a)"



注意：请将 user1.bin 和 user2.bin 均上传至服务器。

(4) 保存后，会在"ROM发布"中看到新上传的v1.0.1t23701(a)版本软件，此时可以选择"设置为当前版本"，进行软件版本更新，设备将收到更新消息，进行升级。

(5) 在弹出的提示框中点击“确定”，更新 v1.0.1t23701(a) 成为“当前版本”。

(6) 用户将收到固件更新的提醒消息。

(7) 选择产品 “dev-controller” 下的任一设备，在“ROM 发布” 中选择固件版本，点击升级。

(8) 设备依据当前自己正在运行的软件，从服务器下载更新后的版本。例如，设备当前正在运行 `user1.bin`，则收到软件更新消息后，设备会从服务器下载新版本的 `user2.bin`。

(9) 新版本软件下载到设备后，用户可以重启设备，自动运行新版本软件。在设备界面，“RPC请求”中，action= 输入“`sys_reboot`”，点击“请求”。

(10) 设备重启后，自动运行新版本软件。

4.4. Curl 指令说明

说明:

PC 侧，指用户可通过 PC 发送 curl 指令控制；

设备侧，指设备收到的数据，或者设备自发向服务器发送的数据，无需用户操作。

1. 升级请求

◇ PC

请求升级的 curl 指令:

Linux/Cygwin curl:

```
curl -X GET -H "Content-Type:application/json" -H "Authorization: token HERE_IS_THE_OWNER_KEY" 'http://iot.espressif.cn/v1/device/rpc/?deliver_to_device=true&action=sys_upgrade&version=v1.0.1t23701(a)'
```

Windows curl:

```
curl -X GET -H "Content-Type:application/json" -H "Authorization: token HERE_IS_THE_OWNER_KEY" "http://iot.espressif.cn/v1/device/rpc/?deliver_to_device=true&action=sys_upgrade&version=v1.0.1t23701(a)"
```

注意：红色的版本号 v1.0.1t23701(a) 为举例，需填入实际将升级的新版本号

◇ 设备

设备收到的数据格式如下:

{"body": {}, "nonce": 855881582, "get": {"action": "sys_upgrade", "version": "v1.0.1t23701(a)", "deliver_to_device": "true"}, "token": "HERE_IS_THE_OWNER_KEY", "meta": {"Authorization": "token HERE_IS_THE_OWNER_KEY"}, "path": "/v1/device/rpc/", "post": {}, "method": "GET", "deliver_to_device": true}

2. 下载完成

◇ 设备

设备收到升级请求后，从服务器下载新版本bin文件，下载完成后，设备向服务器发送如下数据，通知已将新版本软件下载成功。

```
{"path": "/v1/messages/", "method": "POST", "meta": {"Authorization": "token HERE_IS_THE_MASTER_DEVICE_KEY"},"get":{"action":"device_upgrade_success"},"body": {"pre_rom_version":"v1.0.0t23701(a)","rom_version":"v1.0.1t23701(a)"}}
```

3. 重启升级

◇ PC

新版本软件下载完成后，设备重启运行新软件。设备重启升级的curl指令为：

Linux/Cygwin curl：

```
curl -X GET -H "Content-Type:application/json" -H "Authorization: token HERE_IS_THE_OWNER_KEY" 'http://iot.espressif.cn/v1/device/rpc/?deliver_to_device=true&action=sys_reboot'
```

Windows curl：

```
curl -X GET -H "Content-Type:application/json" -H "Authorization: token HERE_IS_THE_OWNER_KEY" "http://iot.espressif.cn/v1/device/rpc/?deliver_to_device=true&action=sys_reboot"
```

◇ 设备

设备收到的数据格式如下：

```
{"body": {}, "nonce": 856543282, "get": {"action": "sys_reboot", "deliver_to_device": "true"}, "token": "HERE_IS_THE_OWNER_KEY", "meta": {"Authorization": "token HERE_IS_THE_OWNER_KEY"}, "path": "/v1/device/rpc/", "post": {}, "method": "GET", "deliver_to_device": true}
```

设备收到 "sys_reboot" 后，会重启运行新版本软件。

5. 软件接口

5.1. struct upgrade_server_info

```
struct upgrade_server_info{
    uint8 ip[4];            // IP address
    uint16 port;            // port number
    uint8 upgrade_flag;     // true – upgrade succeed; false – upgrade fail
    uint32 check_times;     // time out (ms)
    uint8 *url;
    upgrade_states_check_callback check_cb;
```

```
        struct espconn *pespconn;
    };
```

5.2. 升级接口

从 esp_iot_sdk_v0.8 增加支持云端升级功能 (FOTA):

system_upgrade_userbin_check : 检查当前运行 user1.bin 还是 user2.bin

system_upgrade_start : 开始升级 (通过 HTTP 协议)

system_upgrade_reboot : 系统重启，运行新版本软件

system_upgrade_flag_set : 设置升级状态标志

system_upgrade_flag_check : 查询升级状态标志

用户如何调用上述接口，实现云端升级，可参考 IOT_Demo 以下函数：

user_esp_platform_upgrade_begin : 开始从 Espressif Cloud 下载固件

user_esp_platform_upgrade_rsp : 升级回调，进入回调可能是升级成功，可能是超时失败

1. system_upgrade_userbin_check

功能：	检查当前正在使用的固件是 user1 还是 user2 若当前正在运行 user1.bin，则下载新版本的 user2.bin; 若当前正在运行 user2.bin，则下载新版本的 user1.bin
函数定义：	uint8 system_upgrade_userbin_check()
返回：	0x00：UPGRADE_FW_BIN1，即 user1.bin 0x01：UPGRADE_FW_BIN2，即 user2.bin

2. system_upgrade_start

功能:	开始从服务器通过 HTTP 协议下载固件 (user bin)
函数定义:	`bool system_upgrade_start (struct upgrade_server_info *server)`
参数:	`struct upgrade_server_info *server` - 服务器结构体
返回:	true ： 开始升级； false： 已经在升级过程中，无法再次开始升级。

3. system_upgrade_reboot

功能:	重启系统，运行新版本软件。 用于新版固件从服务器下载成功后，调用本接口重启运行新固件。
函数定义:	`void system_upgrade_reboot (void)`
返回:	无

4. system_upgrade_flag_set

功能:	设置升级标志
注意:	如果调用 `system_upgrade_start` 接口升级，则无需调用本接口； 如果调用 `spi_flash_write` 自行写 flash 做升级，则需要调用本接口设置为 `UPGRADE_FLAG_FINISH`，再调用 `system_upgrade_reboot` 重启运行新固件。
函数定义:	`void system_upgrade_flag_set(uint8 flag)`

参数:	`uint8 flag:`
	`#define UPGRADE_FLAG_IDLE 0x00`
	`#define UPGRADE_FLAG_START 0x01`
	`#define UPGRADE_FLAG_FINISH 0x02`

返回:	无

5. system_upgrade_flag_check

功能:	查询升级标志
函数定义:	`uint8 system_upgrade_flag_check()`

返回:	`#define UPGRADE_FLAG_IDLE 0x00`
	`#define UPGRADE_FLAG_START 0x01`
	`#define UPGRADE_FLAG_FINISH 0x02`

6. user_esp_platform_upgrade_begin

功能:	配置结构体 upgrade_server_info，开始升级
函数定义:	`user_esp_platform_upgrade_begin (struct espconn *pespconn, struct upgrade_server_info *server, char *version)`
参数:	`struct espconn` - 服务器结构体，用于向 Espressif Cloud 回复消息。 `struct upgrade_server_info *server` - 升级相关参数 `char *version` - 新版本信息
返回:	无

7. user_esp_platform_upgrade_rsp

功能:	升级回调函数 在 `user_esp_platform_upgrade_begin` 中注册，进入回调，有可能是因为升级成功，有可能是因为超时失败。
函数定义:	`esp_platform_upgrade_rsp(void *arg)`
参数:	`void *arg` - 数据指针
返回:	无

注意:

ESP8266 不限制用户必须使用 Espressif Cloud 云端升级，用户可以自行搭建服务器，调用本章介绍的接口，通过 HTTP 协议下载固件更新即可。

6. 附录

设备固件版本的命名，与云端升级有较大的关系，如果命名未遵守规则，直接使用 IOT_Demo 升级，会判断名称非法，云端升级失败。

6.1. 命名规则

版本命名模板：`[v|b]Num1.Num2.Num3.tPTYPE([o|l|a|n])`

版本命名形如：v1.0.2t45772(a)

软件版本命名示例解析				
v	1.0.2	t	45772	(a)
版本类型	**版本值**	**类型标识**	**设备类型值**	**是否支持升级**
可变	可变	固定不变	可变	可变

1. 版本类型：**v 或 b**

   v: version，表示正式版本

   b: beta，表示测试版本

   注意：同一版本值的软件不能同时存在正式版和测试版，即 v1.0.2 与 b1.0.2 不能共存。

2. 版本值：**NUM1.NUM2.NUM3**

   Num: [0 - 999] 之间的数值

   示例：1.0.2

3. 类型标识：**t**

   t: 类型标识符，后面跟随设备类型 PTYPE 值

4. 设备类型值：**PTYPE**

   PTYPE: 设备在 Espressif Cloud 上的 ptype (product type) 值，即设备类型值。

   可在网站查询 http://iot.espressif.cn/#/api/#api-product-create

5. 升级支持 (FOTA):

o: online， 支持在线升级

l: local， 支持本地升级

a: all， 支持在线升级，也支持本地升级

n: not support , 不支持升级

6.2. 版本值规则

1. 固件版本值

上传服务器的固件版本命名形如：[v|b]Num1.Num2.Num3.tPTYPE([o||a|n])

版本值为：Num1*1000*1000 + Num2*1000 + Num3

例如：

在 Espressif Cloud 查询到 Light ptype = 45772，Switch ptype = 23701，那么，

软件 v1.0.2t45772(a) 为 Light 正式版 1.0.2，同时支持在线 Upgrade 和本地 Upgrade；

版本值 = 1*1000*1000+0*1000+2 = 1000002

软件 b1.0.3t23701(l) 为 Switch 测试版软件 1.0.3，只支持本地Upgrade；

版本值 = 1*1000*1000+0*1000+3 = 1000003

2. 版本值规则

(1) ptype 相同时，同一版本值，至多存在 1 个可升级 (o,l,a) 版本

- 例如，当 b1.0.3t45772(o) 已经存在，则 v1.0.3t45772(o)、v1.0.3t45772(l)、v1.0.3t45772(a)、b1.0.3t45772(l)、b1.0.3t45772(a) 不允许存在。

(2) ptype 相同时，同一版本值，至多存在 1 个不可升级 (n) 版本

- 例如，当 b1.0.3t45772(n) 已存在，则 v1.0.3t45772(n) 不允许存在。

(3) ptype 相同时，不可升级 (n) 版本存在时，至多存在 1 个可升级 (o,l或a) 版本

- 例如，b1.0.3t45772(n) 已存在，b1.0.3t45772(o)、b1.0.3t45772(l)、b1.0.3t45772(a)、v1.0.3t45772(o)、v1.0.3t45772(l)、v1.0.3t45772(a)至多允许存在其中1个

(4) 不同 ptype 可存在相同版本值

- 例如，当 b1.0.3t45772(n) 已存在，b1.0.3t12335(n) 也可以存在