



## OTA 升级说明

硬件产品开发 > 嵌入式软件开发 > MCU 开发接入 > Wi-Fi 通用方案

文档版本: 20201212

[查看在线版本](#)

## 目录

<b>1</b>	<b>SDK 开发</b>	<b>2</b>
1.1	准备工作 . . . . .	2
1.2	协议说明 . . . . .	2
1.3	功能配置 . . . . .	2
1.4	相关命令字 . . . . .	5
1.5	升级回调函数 . . . . .	7
<b>2</b>	<b>平台配置</b>	<b>8</b>
<b>3</b>	<b>功能调试</b>	<b>10</b>
<b>4</b>	<b>常见问题</b>	<b>11</b>

您可以通过涂鸦 IoT 平台，先将需要更新的固件文件上传至涂鸦服务器，然后 Wi-Fi 模组通过涂鸦协议对文件进行分包传输，最后 MCU 接收升级包并写入本地闪存，最终实现固件的升级。

**说明：**OTA（Over-the-Air）即空中下载技术，通过网络远程为设备更新和升级软件程序。

## 1 SDK 开发

### 1.1 准备工作

- 在 IoT 平台的产品 > 硬件开发页面的底部，点击下载 **MCU SDK**、**产品串口通讯协议**、**涂鸦串口调试助手**和**功能点调试文件**。
- 移植涂鸦协议 SDK 的代码。详情请参考 [MCU SDK 移植](#)。
- 自行完成 BootLoader 开发。

### 1.2 协议说明

MCU 升级协议在涂鸦 Wi-Fi 模组串口协议中有相关命令字定义，具体协议格式参考 [MCU 升级服务](#)。

注意：

- Wi-Fi 模组发送完所有的升级包后，模组会重启。重新发送 01 命令字查询产品信息。
- MCU 需要在一分钟内回复产品信息中的软件版本号，带上升级后的 MCU 版本号。版本号需要和在涂鸦平台配置升级的版本号保持一致。

### 1.3 功能配置

了解协议的交互，有助于理解 SDK 的相关代码逻辑。本小节介绍 MCU SDK 相关功能配置。

- 在 SDK 所在路径打开 `protocol.h`，将 `MCU_VER` 数值改为升级后的参数值。

注意：需要升级的固件中，老固件 `MCU_VER` 为当前固件，例如 1.0.0。升级后的固件中 `MCU_VER` 固件版本要改为升级后的目标版本，例如 1.0.1（目标版本号）。

```

1  ~~~
2                                     1: 修改产品信息
3  ****
4  *****/
5  #define PRODUCT_KEY "xghwyjvd3ofo****" // 开发平台创建产品后生成的
    16 位字符产品唯一
6  标识
7  #define MCU_VER "1.0.0" // 用户的软件版本，用于 MCU 固
    件升级。MCU
8  升级版本需修改
9  /*****
10 ****
11  ~~~

```

- 打开 `protocol.h` 中找到升级部分。

```

1  /*
    ****

2  ****
3                                     2: MCU是否需要支持固件升级
4  如需要支持MCU固件升级，请开启该宏
5  MCU可调用mcu_api.c文件内的mcu_firm_update_query()函数获取当前MCU固
    件更新情况
6                                     *****WARNING!!!*****
7  当前接收缓冲区为关闭固件更新功能的大小，固件升级包可选择，默认256
    字节大小
8  如需要开启该功能，串口接收缓冲区会变大
9  ****

10 *****/
11 // #define SUPPORT_MCU_FIRM_UPDATE // 开启
    MCU固件
12 升级功能(默认关闭)
13 // 固件包大小选择
14 #ifdef SUPPORT_MCU_FIRM_UPDATE
15 #define PACKAGE_SIZE 0 // 包大小为256字节
16 // #define PACKAGE_SIZE 1 // 包大小为512字节
17 // #define PACKAGE_SIZE 2 // 包大小为1024字节
18 #endif
19 /*
    ****

20 ****

```

- 打开宏定义 `SUPPORT_MCU_FIRM_UPDATE`。

```

1  /*
    *****

2  *****
3      3: 定义收发缓存:
4      如当前使用MCU的RAM不够, 可修改为24
5  *****

6  *****/
7  #ifndef SUPPORT_MCU_FIRM_UPDATE
8  #define WIFI_UART_RECV_BUF_LMT          16          // 串口数
    据接收缓存区
9  大小, 如MCU的RAM不够, 可缩小
10 #define WIFI_DATA_PROCESS_LMT          24          // 串口数据
    处理缓存区大
11 小, 根据用户DP数据大小量定, 必须大于24
12 #else
13 #define WIFI_UART_RECV_BUF_LMT          128         // 串口数据
    接收缓存区大小
14 , 如MCU的RAM不够, 可缩小
15 //请在此处选择合适的 MCU 升级缓存大小 (根据上面固件包选择大小来选
    择开启多大的 MCU 升级缓存)
16 #define WIFI_DATA_PROCESS_LMT          300         // 固件升
    级缓冲区, 需
17 大缓存, 如单包大小选择256, 则缓存必须大于260
18 // #define WIFI_DATA_PROCESS_LMT          600         // 固件升级
    缓冲区, 需大
19 缓存, 如单包大小选择512, 则缓存必须大于520
20 // #define WIFI_DATA_PROCESS_LMT          1200        // 固件升级
    缓冲区, 需大缓
21 存, 如单包大小选择1024, 则缓存必须大于1030
22 #endif
23 #define WIFIR_UART_SEND_BUF_LMT          48         // 根据用户
    DP数据大小
24 量定, 必须大于48
25 /*
    *****

26  *****

```

**说明:** 云端下发数据长度有 3 种, 请根据 `PACKAGE_SIZE` 对应 `WIFI_DATA_PROCESS_LMT` 的值。例如: `#define PACKAGE_SIZE 1` 对应 `#define WIFI_DATA_PROCESS_LMT 600`。

## 1.4 相关命令字

```
1 #define          UPDATE_START_CMD          0x0a
2                //升级开始
3 #define          UPDATE_TRANS_CMD         0x0b
4                //升级传输
```

### 升级开始 (0x0a)

```
1 #ifdef SUPPORT_MCU_FIRM_UPDATE
2     case UPDATE_START_CMD:                // 升级开始
3         // 获取升级包大小全局变量
4         firm_flag = PACKAGE_SIZE;
5         if(firm_flag == 0) {
6             firm_size = 256;
7         }else if(firm_flag == 1) {
8             firm_size = 512;
9         }else if(firm_flag == 2) {
10            firm_size = 1024;
11        }
12        firm_length = wifi_data_process_buf[offset + DATA_START];
13        firm_length <= 8;
14        firm_length |= wifi_data_process_buf[offset + DATA_START + 1];
15        firm_length <= 8;
16        firm_length |= wifi_data_process_buf[offset + DATA_START + 2];
17        firm_length <= 8;
18        firm_length |= wifi_data_process_buf[offset + DATA_START + 3];
19
20        upgrade_package_choose(PACKAGE_SIZE);
21        firm_update_flag = UPDATE_START_CMD;
22        break;
```

**说明：**以上代码是处理接收函数。根据上文标志位的长度，选择数据包的规格，并回复云端。云端接收到数据包规格后下发数据。

### 升级传输 (0x0b)

```
1  case UPDATE_TRANS_CMD: // 升级传输
2      if(firm_update_flag == UPDATE_START_CMD)
3      {
4          //停止一切数据上报
5          stop_update_flag = ENABLE;
6
7          total_len = wifi_data_process_buf[offset + LENGTH_HIGH] * 0x10
8  0;
9          total_len += wifi_data_process_buf[offset + LENGTH_LOW];
10
11         dp_len = wifi_data_process_buf[offset + DATA_START];
12         dp_len <= 8;
13         dp_len |= wifi_data_process_buf[offset + DATA_START + 1];
14         dp_len <= 8;
15         dp_len |= wifi_data_process_buf[offset + DATA_START + 2];
16         dp_len <= 8;
17         dp_len |= wifi_data_process_buf[offset + DATA_START + 3];
18
19         firmware_addr = (unsigned char *)wifi_data_process_buf;
20         firmware_addr += (offset + DATA_START + 4);
21
22         if((total_len == 4) && (dp_len == firm_length))
23         {
24             // 最后一包
25             ret = mcu_firm_update_handle(firmware_addr,dp_len,0);
26
27             firm_update_flag = 0;
28         }
29         else if((total_len - 4) <= firm_size)
30         {
31             ret = mcu_firm_update_handle(firmware_addr,dp_len,total_len
32 - 4);
33         }
34         else
35         {
36             firm_update_flag = 0;
37             ret = ERROR;
38         }
39
40         if(ret == SUCCESS)
41         {
42             wifi_uart_write_frame(UPDATE_TRANS_CMD,0);
43         }
44         // 恢复一切数据上报
45         stop_update_flag = DISABLE;
46     }
47     break;
48 #endif
```



## 1.5 升级回调函数

```
1  /*****
2  *****/
3  函数名称: mcu_firm_update_handle
4  功能描述: MCU进入固件升级模式
5  输入参数: value:固件缓冲区
6             position: 数据包的位置
7             length: 当前固件包长度(固件包长度为0时, 表示固件包发送完成)
8  返回参数: 无
9  使用说明: MCU 需要自行实现该功能
10 *****/
11 *****/
12 unsigned char mcu_firm_update_handle(const unsigned char value[],uns
13 igned long position,unsigned short length)
14 {
15     #error "请自行完成 MCU 固件升级代码, 完成后请删除该行"
16     if(length == 0)
17     {
18         // 固件数据发送完成
19     }
20     }
21     else
22     {
23         // 固件数据处理
24     }
25 }
26 return SUCCESS;
27 }
28 #endif
```

## 2 平台配置

说明：已在 **产品开发 > 硬件开发 > 已选固件 > 新增自定义固件** 中，新增固件。

本小节以取暖器为例，介绍 OTA 升级的配置步骤。

1. 登录 [IoT 控制台产品列表](#)。
2. 鼠标悬浮至一款**开发中**的产品，单击**进入开发**。
3. 单击**产品配置**。
4. 在**固件升级**栏，单击**设置**。
5. 在**固件版本管理**页面左上角，选择固件。



6. 单击页面右上角**新增固件版本**，创建新固件版本。

- **固件上传**：固件升级包为 `.bin` 格式。
- **固件版本**：版本号格式为 `xx.xx.xx`，例如 1.0.6。
- **升级方式**：
  - **提醒升级**：App 中出现升级弹窗，可选择升级或不升级。

1 **\*\*静默升级\*\***：App 中不出现升级弹窗，固件通电后自动检测固件版本并升级。

1 **\*\*强制升级\*\***：App 中出现升级弹窗，用户必须升级后才能继续使用。

1. **\*\*检测升级\*\***：App 中不出现升级弹窗，点击相关固件版本检测，并主动更新。

## 7. 添加测试设备。

1. 在**固件版本管理**页面，单击**常用白名单管理**。
2. 在**设备白名单**页面，选择白名单区域，单击**新增白名单设备**。
3. 在**验证码验证**页面，输入**涂鸦账号**和**验证码**，单击**确定**，添加测试设备。

## 8. 固件推送并验证。

1. 在**固件版本管理**页面，单击**固件升级栏的验证**。

rtlibn\_tls\_comr...n5sjxrqcryptn)

固件升级文档

常用白名单管理

新增固件版本

固件名称/固件Key	固件类型	创建/更新时间	固件版本	固件包来源	升级方式	升级文案	固件升级	操作
rtlibn_9600keym	模组固件	2020-09-21 16:30:12 2020-09-21 16:33:00	7.7.7	涂鸦推送	App 提醒升级	中文：2.1.3静默 英文：2.1.3 <a href="#">更多语言 &gt;</a>	<div><div>1 验证 ①</div><div>2 发布 ①</div></div>	<a href="#">编辑</a> <a href="#">删除</a>
rtlibn_9600keym	模组固件	2020-09-21 11:40:24 2020-09-21 11:40:39	2.1.3	涂鸦推送	App 提醒升级	中文：新版本2.1.3 英文：new2.1.3 <a href="#">更多语言 &gt;</a>	<div><div>1 验证</div><div>2 发布</div></div>	<a href="#">编辑</a> <a href="#">删除</a>

2. 在页面右上角选择验证地域。> **说明**：当前支持六区（中国区、美国区、欧洲区、微软区、印度区和西欧区）进行设备验证。
3. 单击**验证是否完成升级**进行测试设备验证。

**说明**：支持以添加设备 ID 或从白名单管理中选择设备 ID 的方式选择测试设备。

常用白名单管理

中国区

测试设备验证列表

测试结果仅代表OTA升级是否成功，即设备当前固件版本与目标版本一致，不包含固件功能可用性测试。[请参考文档](#)

从白名单选择添加

通过设备号直接添加

验证是否完成升级

设备ID	版本	用户ID	测试结果	操作
vdevo160446167453448	1.0.0	86-18913191123	未验证	<a href="#">删除</a>

共 1 条记录 < 1 >

## 3 功能调试

功能调试方法，参见[使用模组调试助手](#)。

## 4 常见问题

Q: 如果升级包存在错误数据升级失败后, 重新进行升级, Wi-Fi 会重复发送当前数据吗? A: 会。当前数据发送三次, 三次之后判断升级失败。失败后, 需要下次重新启动升级, Wi-Fi 将重新发送所有数据。

Q: 如何获取产品 ID? A: 您可以在 App 端产品页面的**编辑** (铅笔图标) > **设备信息** > **虚拟 ID**, 复制设备 ID。

