ESP8266 Non-OS SDK IoT_Demo 指南



版本 1.4 版权 © 2016

关于本手册

本手册结构如下:

章	标题	内容
第1章	概述	简单介绍 <i>IoT_Demo</i> 。
第2章	IoT_Demo 应用程序	介绍 IoT_Demo 应用程序,及其编译烧录。
第3章	curl 工具	介绍 curl 工具及使用注意事项。
第4章	局域网功能	介绍 IoT_Demo 局域网内的功能示例。
第5章	广域网功能	介绍 IoT_Demo 与乐鑫云交互的功能示例。

发布说明

日期	版本	发布说明
2016.04	V1.3	首次发布
2016.08	V1.4	增加 curl 工具及使用注意事项

1.	概述.			1
2.	IoT_E	Demo 应	用程序	2
	2.1.	loT_Der	mo 程序简介	2
	2.2.	编译固件	‡	3
		2.2.1.	修改 IoT_Demo	3
		2.2.2.	编译 IoT_Demo	4
	2.3.	云端创始	建设备	6
		2.3.1.	获取 master_device_key.bin	6
		2.3.2.	创建数据模型	7
	2.4.	烧录固件	‡	9
3.	curl _	□具		11
4.	局域隔	网功能		12
	4.1.	通用功能	t	12
		4.1.1.	查询版本信息	12
		4.1.2.	设置 ESP8266 Station 连接 AP	12
		4.1.3.	设置 ESP8266 SoftAP 参数	14
	4.2.	重启或体	木眠功能	15
	4.3.	查找局域	或网内的 ESP8266 设备	15
	4.4.	智能插座	<u>×</u>	16
	4.5.	智能灯.		17
	4.6.	传感器记	ຽ备	18
5.	广域区	网功能		19
	5.1.	ESP826	6 设备激活	19

5.2.	ESP8266 设备认证	20	
5.3.	PING 服务器	21	
5.4.	智能插座	21	
5.5.	智能灯	23	
5.6.	温湿度传感器	26	
5.7.	用户自定义反向控制	27	



1.

ESP8266_NONOS_SDK 下载链接:

http://www.espressif.com/support/download/sdks-demos

ESP8266_NONOS_SDK\examples\loT_Demo 提供智能灯、智能插座和传感器三种设备的简单示例,通过连接到乐鑫云端服务器,实现对 ESP8266 智能设备的操作控制和数据采集等功能。

本文介绍基于 *IoT_Demo* 应用程序,通过 curl 指令控制 ESP8266 设备,以及与乐鑫云交互的示例说明。

说明:

- 乐鑫云端服务器 http://iot.espressif.cn/#/。
- 初次使用乐鑫云,请参考帮助文档 http://iot.espressif.cn/#/help-zh-cn/。



2.

IoT_Demo 应用程序

2.1. IoT_Demo 程序简介

示例应用 ESP8266_NONOS_SDK\examples\loT_Demo 的结构如下图。



图 2-1. IoT Demo 文件夹

1. user 文件夹

- user_main.c: 应用程序起始,实现初始化功能,程序主入口函数为 user_init。
- *user_esp_platform.c*: ESP8266 设备与乐鑫云通信的示例。
- user_esp_platform_timer.c: ESP8266 设备实现乐鑫云端定时器的示例。
- *user_webserver.c*: 创建 TCP server 的示例。
- user devicefind.c: 实现 UDP 传输的示例。
- user_sensor.c: ESP8266 传感器设备的示例。
- user_plug.c: ESP8266 智能插座的示例。
- user_light.c: ESP8266 智能灯设备的示例。
- *user_json.c*: json 包的处理示例。

2. include 文件夹

• IoT_Demo 应用程序的头文件。

3. driver 文件夹

- i2c_master.c: ESP8266 作为 I2C master 的示例。
- key.c: GPIO 的使用示例。

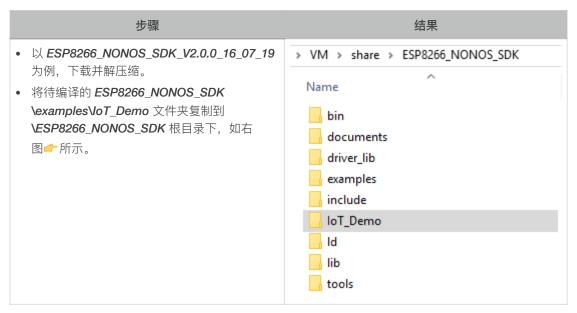


2.2. 编译固件

2.2.1. 修改 IoT_Demo

1. *ESP8266_NONOS_SDK* 下载链接:

http://www.espressif.com/support/download/sdks-demos



2. **IoT_Demo** 提供智能插座(PLUG_DEVICE)、智能灯(LIGHT_DEVICE)和智能传感器(SENSOR_DEVICE)三种设备的简单示例。默认设备类型为智能灯。



3. 根据实际使用的 ESP8266 硬件模块 Flash 大小,修改的用户参数区位置。





使用不同的 Flash map 对应头文件中的修改位置,如下表 2-1。

表 2-1. 修改 include 文件中的字段(单位: KB)

默认值 (512)	修改后的值						
	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		

1 注意:

IoT_Demo 的默认波特率为 74880。

2.2.2. 编译 IoT_Demo

编译 *ESP8266_NONOS_SDKVoT_Demo*,步骤如下图,详细的编译指南可参考文档《*ESP8266 SDK* 入门指南》。

Espressif 4/29 2016.08



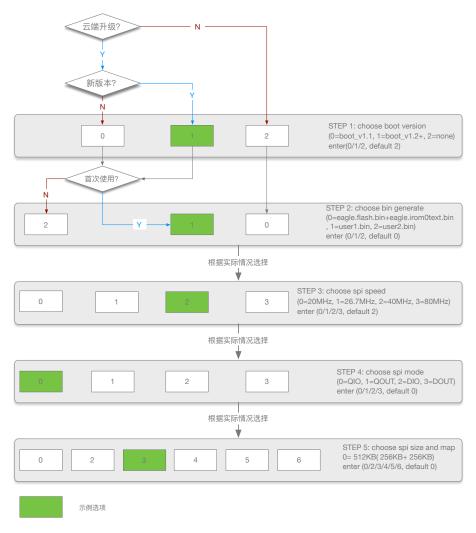


图 2-2. 编译说明

1 注意:

- 图 2-2 中, 颜色标示部分为示例选项, 您可以按照实际需求选择。
- 编译 STEP 5 的选项 5 和 6 仅 **sdk_v1.1.0 + boot 1.4 + flash download tool_v1.2** 及之后版本支持。
- user1.bin 和 user2.bin 由同一份应用代码在编译过程中 STEP 2 选择不同的选项,分别编译生成:
 - 编译生成 *user1.bin* 后,先运行 make clean 清除上次编译生成的临时文件后,再重新编译生成 *user2.bin* 。
 - **user2.bin** 用于实现云端升级功能,无需烧录到 *Flash*,详细说明见文档《*ESP8266* 云端升级指南》。

Espressif 5/29 2016.08



2.3. 云端创建设备

根据前文**章节 2.2.1** 中 *IoT_Demo* 使能的设备类型,在乐鑫云端服务器创建一个同样的设备类型。例如,假设 *ESP8266_NONOS_SDK\IoT_Demo\include\user_config.h* 中使能的是 LIGHT_DEVICE,则在乐鑫云创建一个智能灯设备。

2.3.1. 获取 master_device_key.bin

master_device_key 是开发者在乐鑫云服务器创建一个智能设备时,乐鑫云自动为该智能设备分配的 ID 值,具有唯一性。智能设备依此享受乐鑫云端服务。

说明:

初次使用乐鑫云,请参考帮助文档 http://iot.espressif.cn/#/help-zh-cn/。

1. 注册用户,并登录乐鑫云(<u>http://iot.espressif.cn/#/</u>),创建智能设备。例如,创建一个智能灯。





步骤	结果
 • 创建完成后,将自动跳转到新设备页面。 • 在设备页面,可以看到该设备的 Master Device Key 值。 • 如右图	□ 设备 { id: 8316, serial: 3d770f9d } □ ilight-001 □ Public Device Product Product Secret Device Secret Master Device Key Product Secret Master Device Key

2. 从乐鑫云导出 master_device_key.bin。

步骤	结果
 在设备 "light-001" 的页面右下角,点击 "下载 Key BIN"。 如右图→所示。 	下载 Key BIN
• 如上点击后,将下载设备 "light-001" 的 master_device_key.bin。	cdecad163d892f30392c2ab065b30e4c17c 8/5/2016 9:49 PM
- BIN 文件名称与设备 "light-001" 的 Master Device Key 值一致。	
• 如右图一所示。	

2.3.2. 创建数据模型

"**数据模型**"是开发者为智能设备定义的功能属性。开发者在应用程序代码中实现相关功能,在乐鑫云定义同名数据模型,从而实现 ESP8266 设备与乐鑫云之间对应功能的通信。

在 IoT_Demo 的示例中,实现了如下数据模型:

- 对于智能灯设备,实现了 light 数据模型,用于调节灯光色彩亮度。
- 对于智能插座设备,实现了 plug-status 数据模型,用于控制插座的开关状态。
- 对于温湿度传感器设备,实现了 tem hum 数据模型,用于上传温度、湿度信息。

🛄 说明:

- 应用程序中的对应代码为 IoT_Demo\user\user_esp_platform.c。
- 在乐鑫云创建"数据模型"时,在设备页面或者产品页面均可以创建,效果相同。
- 开发者可以参考 IoT_Demo 为智能设备自定义实现其他数据模型功能。



1. 如果是智能灯设备,在乐鑫云中的该设备页面,创建 light 数据模型。



2. 如果是智能插座设备,在乐鑫云中的设备页面,创建 plug-status 数据模型。



3. 如果是温湿度传感器,在乐鑫云中的设备页面,创建 tem_hum 数据模型。







2.4. 烧录固件

根据实际使用的 ESP8266 硬件模块 Flash 大小,对应烧录地址如下表。

表 2-2. FOTA 固件下载地址(单位: KB)

	各个 Flash 容量对应的下载地址					
BIN 文件	512	1024	2048		4096	
			512+512	1024+1024	512+512	1024+1024
master_device_key.bin	0x3E000	0x7E000	0x7E000	0xFE000	0x7E000	0xFE000
blank.bin(烧录位置一)	0x7B000	0xFB000	0x1FB000		0x3FB000	
esp_init_data_default.bin	0x7C000	0xFC000	0x1FC000		0x3FC000	
blank.bin(烧录位置二)	0x7E000	0xFE000	0x1I	FE000	0x3F	E000
boot.bin	0x00000					
user1.bin				0x01000		

表 2-3. FOTA 固件说明

BIN 文件	说明
master_device_key.bin	用户从乐鑫云申请,依此享受乐鑫云端服务。 存放于用户参数区,储存地址由用户应用程序自定义。 表 2-2 中的烧录位置为 <i>loT_Demo</i> 程序按照 章节 2.2.1 设定的示例位置。
blank.bin (烧录位置一)	初始化 RF_CAL 参数区。 烧录位置由应用程序中的 user_rf_cal_sector_set 设置决定。 表 2-2 中的烧录位置为 loT_Demo 程序中设定的示例位置。 由乐鑫官方提供,位于 ESP266_SDK\bin 路径下。



BIN 文件	说明
esp_init_data_default.bin	初始化其他射频参数区,至少烧录一次。 当 RF_CAL 参数区初始化烧录时,本区域也需烧录。 由乐鑫官方提供,位于 ESP266_SDK\bin 路径下。
blank.bin (烧录位置二)	初始化系统参数区。 由乐鑫官方提供,位于 <i>ESP266_SDK\bin</i> 路径下。
boot.bin	主程序,由乐鑫官方提供,位于 ESP266_SDK\bin 路径下。
user1.bin	主程序,编译应用程序生成,位于 <i>ESP266_NONOS_SDK\bin\upgrade</i> 路径下。



3.

curl 工具

curl 工具下载链接: http://curl.haxx.se/download.html

说明:

- 若用户使用 Windows curl,则后文中的 curl 指令请参照 "Windows curl" 的示例。
- 若用户使用 Linux curl 或者 Cygwin curl,则后文中的 curl 指令请参照 "Linux/Cygwin curl" 的示例。
- 后文中的 curl 指令若无上述区分列举,则表示可以通用。

curl 指令使用时的常见错误:

- 注意 curl 指令中的字符大小写。若字符大小写使用错误,则指令出错。
- curl 指令中均为英文标点符号。若 curl 指令夹杂了中文符号,则指令出错。
- 注意 curl 指令中的空格符。若缺少了空格符,或者多打成两个空格符,则指令出错。
- 根据 curl 指令工具(Linux/Cygwin 或 Windows)不同,注意选择正确的指令格式。
- 与乐鑫云通信的随机 token 具有——对应性,请勿多个设备共用同—随机 token。



4.

局域网功能

说明:

ESP8266 SoftAP 接口默认 IP 为192.168.4.1, Station 接口的 IP 由路由分配。以下 URL 中 "i p"信息指SoftAP 和 Station 模式下的 IP, 需输入实际的 IP 地址。

4.1. 通用功能

4.1.1. 查询版本信息

- PC 作为 Station 连接到 ESP8266 SoftAP, 并发送以下 curl 指令查询。
- curl 指令中的 ip 需使用 ESP8266 实际的 IP 地址。

```
curl -X GET http://ip/client?command=info
```

响应:

4.1.2. 设置 ESP8266 Station 连接 AP

ESP8266 初始状态为 SoftAP + Station 模式, PC 作为 Station 连接到 ESP8266 SoftAP, 发送以下 curl 指令设置 ESP8266 Station 连接目标 AP(路由器)。

Linux/Cygwin curl:

```
curl -X POST -H "Content-Type:application/json" -d '{"Request":
{"Station":{"Connect_Station":
```



```
{"ssid":"tenda","password":"1234567890","token":
"1234567890123456789012345678901234567890"}}}' http://192.168.4.1/
config?command=wifi
```

Windows curl:

1 注意:

- 上述红色 token 字段是个随机的长度为 40 的 16 进制数的字符串,具有一一对应性,请勿多个 ESP8266 设备共用同一随机 token 值。
 - ESP8266 设备后续使用此随机 token 向乐鑫云激活认证。
 - 用户将使用同一个随机 token 向乐鑫云申请该 ESP8266 设备的控制权限。
 - 因此,随机 token 与 ESP8266 设备是——对应的关系,不能与其他设备共用。

特殊 AP 配置

如果 AP 的加密方式为 WEP HEX,则密码需要转为 ASC 码 HEX 值。

示例:

假设路由 SSID 为 "wifi_1", 密码为 "tdr0123456789", 加密方式为 WEP, 则

Linux/Cygwin curl:

Windows curl:

查询连接状态

在配置 ESP8266 Station 接口连接 AP 的过程中,可通过如下 curl 指令,查询连接状态。

curl -X GET http://ip/client?command=status



连接状态的定义如下:

```
enum {
    Station_IDLE = 0,
    Station_CONNECTING,
    Station_WRONG_PASSWORD,
    Station_NO_AP_FOUND,
    Station_CONNECT_FAIL,
    Station_GOT_IP
};
enum {
    DEVICE_CONNECTING = 40,
    DEVICE_ACTIVE_DONE,
    DEVICE_ACTIVE_FAIL,
    DEVICE_CONNECT_SERVER_FAIL
};
```

4.1.3. 设置 ESP8266 SoftAP 参数

发送以下 curl 指令可以设置 ESP8266 SoftAP 的参数,例如 SSID,密码等。

Linux/Cygwin curl:

Windows curl:

```
curl -X POST -H "Content-Type:application/json" -d "{\"Request\":
{\"SoftAP\":{\"authmode\":\"0PEN\",\"channel\":
6,\"ssid\":\"ESP_IOT_SoftAP\",\"password\":\"\"}}}" http://
192.168.4.1/config?command=wifi
```

1 注意:

- authmode 支持: OPEN, WPAPSK, WPA2PSK, WPAPSK/WPA2PSK。
- password 长度至少为 8 字节。



4.2. 重启或休眠功能

• 对于智能插座或者智能灯设备,可以发送以下 curl 指令让 ESP8266 设备重启。

curl -X POST http://ip/config?command=reboot

• 对于传感器设备,可以发送以下 curl 指令让 ESP8266 设备休眠。

curl -X POST http://ip/config?command=sleep

传感器设备,休眠30秒后,将自动唤醒重新运行。

4.3. 查找局域网内的 ESP8266 设备

PC 可以通过向端口 1025 发送 UDP 广播包的方法,查找在同一局域网内的 ESP8266 智能设备,步骤如下:

- PC 发送向端口 1025 发送广播包 "Are You Espressif IOT Smart Device?"。
- ESP8266 智能设备监听 1025 端口,如果收到的 UDP 广播包,则判断是否为如上字符串;判断字符串匹配,则回复响应。

说明:

应用程序中的对应代码为 IoT Demo\user\user devicefind.c。

PC 可以使用网络调试工具发送 UDP 广播包,例如,使用网络调试助手



图 4-1. PC 网络调试助手



• 如果是 ESP8266 智能插座, 响应:

I'm Plug.xx:xx:xx:xx:xx yyy.yyy.yyy

• 如果是 ESP8266 智能灯, 响应:

I'm Light.xx:xx:xx:xx:xx:xx yyy.yyy.yyy

• 如果是 ESP8266 温湿度传感器,响应:

I'm Humiture.xx:xx:xx:xx:xx yyy.yyy.yyy

• 如果是 ESP8266 可燃气体检测传感器,响应:

I'm Flammable Gas.xx:xx:xx:xx:xx yyy.yyy.yyy

说明:

- "xx:xx:xx:xx:xx:xx 表示 ESP8266 设备的实际 MAC 地址。
- "yyy.yyy.yyy"表示 ESP8266 设备的实际 IP 地址。

4.4. 智能插座

查询插座状态

如果 ESP8266 的设备类型是智能插座,可以通过以下 curl 指令查询插座的状态。

curl -X GET http://ip/config?command=switch

响应:

```
{
    "Response": {
        "status": 0
    }
}
```

说明:

"status"为 0 表示插座处于关闭状态,为 1 表示插座处于开启状态。

设置插座状态

如果 ESP8266 的设备类型是智能插座,可以通过以下 curl 指令设置插座的状态。

Linux/Cygwin curl:



Windows curl:

```
curl -X POST -H "Content-Type:application/json" -d "{\"Response\":
{\"status\":1}}" http://ip/config?command=switch
```

4.5. 智能灯

查询灯的状态

如果 ESP8266 的设备类型是智能灯,可以通过以下 curl 指令查询灯的状态。

```
curl -X GET http://ip/config?command=light
```

响应:

说明:

- "period" 表示 PWM 周期, 单位: 毫秒。
- "status" 表示智能灯的状态:
 - 0: 关灯
 - 1: 开灯
 - 2: 关闭白灯, 但改变红灯、蓝灯、绿灯的颜色
 - 3: 关闭红灯、蓝灯、绿灯, 但改变白灯的亮度
- "rgb"的取值范围即 PWM 波形的占空比 (duty) 取值范围:
 - ESP8266_NONOS_SDK_V2.0 及之前版本, PWM duty 的取值范围为 [0, 22222]。
 - ESP8266_NONOS_SDK_V2.0 之后版本, PWM duty 的取值范围为 [0, 255]。
 - ESP8266_RTOS_SDK_V1.4.0 及之前版本, PWM duty 的取值范围为 [0, 1023]。
 - ESP8266_RTOS_SDK_V1.4.0 之后版本, PWM duty 的取值范围为 [0, 255]。



设置灯的状态

如果 ESP8266 的设备类型是智能灯,可以通过以下 curl 指令设置灯的状态。

• 如果使用 ESP8266 NONOS SDK V2.0 之后的版本:

Linux/Cygwin curl:

```
curl -X POST -H "Content-Type:application/json" -d '{"period":
1000, "status":3, "color":{"red":0, "green":0, "blue":0, "white":255}'
http://ip/config?command=light
```

Windows curl:

```
curl -X POST -H "Content-Type:application/json" -d "{\"period\":
1000,\"status\":3,\"color\":{\"red\":0,\"green\":0,\"blue\":0,\"white
\":255}' http://ip/config?command=light
```

• 如果使用 ESP8266 NONOS SDK V2.0 及之前版本:

Linux/Cygwin curl:

```
curl -X POST -H "Content-Type:application/json" -d '{"period":1000,
    "rgb":{"red":200, "green":0, "blue":0, "cwhite":0, "wwhite":0}}'
http://ip/config?command=light
```

Windows curl:

```
curl -X POST -H "Content-Type:application/json" -d "{\"period\":
1000,\"rgb\":{\"red\":200,\"green\":0,\"blue\":0}}" http://ip/config?
command=light
```

4.6. 传感器设备

如果 ESP8266 的设备类型是智能传感器,则不支持本地获取或设置状态,仅支持在广域 网下通过乐鑫云获取或设置设备状态。



5.

广域网功能

说明:

- 后述 "设备" 指 ESP8266 设备自动与乐鑫云进行的通信, 无需用户操作。
- 后述 "PC" 指用户通过 PC 发送 curl 指令,操作 ESP8266 设备。

5.1. ESP8266 设备激活

设备:

ESP8266 设备通过路由器连接外网(参考**章节 4.1.2**)之后,设备将自动连接乐鑫云。

如果当前 ESP8266 设备是第一次连接乐鑫云,将自动向乐鑫云(端口:8000)发送如下格式的 TCP 数据包进行激活。

```
{"path": "/v1/device/activate/", "method": "POST", "meta":
{"Authorization": "token HERE_IS_THE_MASTER_DEVICE_KEY"}, "body":
{"encrypt_method": "PLAIN", "bSSID": "18:fe:34:70:12:00", "token":
"1234567890123456789012345678901234567890"}}
```

说明:

- "HERE IS THE MASTER DEVICE KEY" 指 ESP8266 设备的实际 Master Device Key 值。
- 红色 "token" 值为前文章节 4.1.2 中 curl 指令设置的随机 token 值。

乐鑫云响应:

```
{"status": 200, "device": {device}, "key": {key}, "token":
{token}}
```

PC:

用户使用 PC 向乐鑫云发送如下 curl 指令,向乐鑫云申请对应 ESP8266 设备的控制权。

Linux/Cygwin curl:

```
curl -X POST -H "Authorization:token
c8922638bb6ec4c18fcf3e44ce9955f19fa3ba12" -d '{"token":
"1234567890123456789012345678901234567890"}' http://iot.espressif.cn/
v1/key/authorize/
```

Windows curl:

```
curl -X POST -H "Authorization:token
c8922638bb6ec4c18fcf3e44ce9955f19fa3ba12" -d "{\"token\":
\"1234567890123456789012345678901234567890\"}" http://
iot.espressif.cn/v1/key/authorize/
```



说明:

"c8922638bb6ec4c18fcf3e44ce9955f19fa3ba12"指开发者在乐鑫云注册用户时,自动获得的 *user key* (用户身份 *ID* 值) 的示例。查询步骤如下:

- 登录乐鑫云 (http://iot.espressif.cn/)
- 点击右上角的用户名称
- 点击进入"设置"
- 点击 "开发者",则可以看到 user key 值

响应:

```
{"status": 200, "key": {"updated": "2014-05-12 21:22:03",
"user_id": 1, "product_id": 0, "name": "device activate share
token", "created": "2014-05-12 21:22:03", "source_ip": "*",
"visibly": 1, "id": 149, "datastream_tmpl_id": 0, "token":
"e474bba4b8e11b97b91019e61b7a018cdbaa3246", "access_methods": "*",
"is_owner_key": 1, "scope": 3, "device_id": 29,
"activate_status": 1, "datastream_id": 0, "expired_at": "2288-02-22
20:31:47"}}
```

■ 说明:

"e474bba4b8e11b97b91019e61b7a018cdbaa3246" 指乐鑫云返回给用户的 *owner key* 值,表示乐鑫云认可用户为该 *ESP8266* 设备的拥有者,用户可使用 *owner key* 控制该 *ESP8266* 设备。

5.2. ESP8266 设备认证

设备:

激活成功后,ESP8266 设备每次连接乐鑫云时 (包括进行激活的当前次),向乐鑫云(端口:8000) 发送如下格式的 TCP 包,认证自己是合法的 ESP8266 设备。

```
{"nonce": 560192812, "path": "/v1/device/identify", "method":
"GET", "meta": {"Authorization": "token
HERE_IS_THE_MASTER_DEVICE_KEY"}}
```

说明:

- "nonce" 值为一组随机的整数。乐鑫云回复时,响应中将包含同一 "nonce" 值。
- "HERE_IS_THE_MASTER_DEVICE_KEY" 指 ESP8266 设备的实际 Master Device Key 值。

乐鑫云认证 ESP8266 设备提供的确实是合法的乐鑫 Master Device Key 值后,向 ESP8266 设备回复如下数据包,设备身份认证成功。

响应:

```
{"device": {"productbatch_id": 0, "last_active": "2014-06-19 10:06:58", "ptype": 12335, "activate_status": 1, "serial":
```



```
"334a8481", "id": 130, "bSSID": "18:fe:34:97:d5:33", "last_pull":
"2014-06-19 10:06:58", "last_push": "2014-06-19 10:06:58",
"location": "", "metadata": "18:fe:34:97:d5:33 temperature",
"status": 2, "updated": "2014-06-19 10:06:58", "description":
"device-description-79eba060", "activated_at": "2014-06-19
10:06:58", "visibly": 1, "is_private": 1, "product_id": 1,
"name": "device-name-79eba060", "created": "2014-05-28 17:43:29",
"is_frozen": 0, "key_id": 387}, "nonce": 560192812, "message":
"device identified", "status": 200}
```

说明:

上述认证过程,在智能灯和智能插座的应用中,需要运用。

5.3. PING 服务器

设备:

为了保持 ESP8266 设备与乐鑫云之间的 socket 连接,ESP8266 设备每 50 秒向乐鑫云(端口: 8000) 发送如下格式的 TCP 包。

```
{"path": "/v1/ping/", "method": "POST", "meta": {"Authorization": "token HERE_IS_THE_MASTER_DEVICE_KEY"}}
```

乐鑫云响应:

```
{"status": 200, "message": "ping success", "datetime": "2014-06-19 09:32:28", "nonce": 977346588}
```

说明:

PING 服务器的机制,在智能灯和智能插座的应用中,需要运用。

5.4. 智能插座

设备:

• 当 ESP8266 设备收到乐鑫云发来的 GET 命令时,ESP8266 设备需要将自身的状态上传至乐鑫云。

乐鑫云发来的 GET 命令格式如下:

```
{"body": {}, "nonce": 33377242, "is_query_device": true, "get":
{}, "token": "HERE_IS_THE_OWNER_KEY", "meta": {"Authorization":
    "token HERE_IS_THE_OWNER_KEY"}, "path": "/v1/datastreams/plug-
status/datapoint/", "post": {}, "method": "GET"}
```

ESP8266 设备回复乐鑫云的响应格式如下:



```
{"status": 200, "datapoint": {"x": 0}, "nonce": 33377242, "is_query_device": true}
```

• 当 ESP8266 设备收到乐鑫云发来的 POST 命令时,ESP8266 设备需要根据乐鑫云的命令、改变自身的状态。例如,开启智能插座开关的命令格式如下:

```
{"body": {"datapoint": {"x": 1}}, "nonce": 620580862,
"is_query_device": true, "get": {}, "token":
"HERE_IS_THE_OWNER_KEY", "meta": {"Authorization": "token
HERE_IS_THE_OWNER_KEY"}, "path": "/v1/datastreams/plug-status/
datapoint/", "post": {}, "method": "POST", "deliver_to_device":
true}
```

ESP8266 智能插座完成动作后,向乐鑫云回复状态更新成功的响应,格式如下:

```
{"status": 200, "datapoint": {"x": 1}, "nonce": 620580862, "deliver_to_device": true}
```

说明:

响应中的 nonce 值必须与乐鑫云发送的控制命令中的 nonce 值一致,以表示每次控制命令和响应相互对应。

PC:

查询插座状态

用户使用 PC 向乐鑫云发送如下 curl 指令,查询智能插座的状态。

```
curl -X GET -H "Content-Type:application/json" -H "Authorization:
token HERE_IS_THE_OWNER_KEY" http://iot.espressif.cn/v1/datastreams/
plug-status/datapoint/
```

响应:

```
{"status": 200, "nonce": 11432809, "datapoint": {"x": 1}, "deliver_to_device": true}
```

设置插座状态

用户使用 PC 向乐鑫云发送如下 curl 指令,设置智能插座的状态。

Linux/Cygwin curl:

```
curl -X POST -H "Content-Type:application/json" -H "Authorization:
token HERE_IS_THE_OWNER_KEY" -d '{"datapoint":{"x":1}}' http://
iot.espressif.cn/v1/datastreams/plug-status/datapoint/?
deliver to device=true
```



Windows curl:

```
curl -X POST -H "Content-Type:application/json" -H "Authorization:
token HERE_IS_THE_OWNER_KEY" -d "{\"datapoint\":{\"x\":1}}" http://
iot.espressif.cn/v1/datastreams/plug-status/datapoint/?
deliver_to_device=true
```

响应:

```
{"status": 200, "nonce": 11432809, "datapoint": {"x": 1}, "deliver_to_device": true}
```

5.5. 智能灯

设备:

当 ESP8266 设备收到乐鑫云发来的 GET 命令时,ESP8266 设备需要将自身的状态上传至乐鑫云。

乐鑫云发来的 GET 命令格式如下:

```
{"body": {}, "nonce": 8968711, "is_query_device": true, "get": {},
  "token": "HERE_IS_THE_OWNER_KEY", "meta": {"Authorization": "token
  HERE_IS_THE_OWNER_KEY"}, "path": "/v1/datastreams/light/datapoint/",
  "post": {}, "method": "GET"}
```

如果使用 *ESP8266_NONOS_SDK_V2.0* 之后的版本, ESP8266 设备回复乐鑫云的响应格式如下:

```
{"nonce": 5619936, "datapoint": {"x": 1, "y": 1000, "z":{"red": 0, 
"green": 0, "blue": 0, "white": 255}}, "deliver_to_device": true}
```

说明:

- "x"表示智能灯的状态:
 - 0: 关灯
 - 1: 开灯
 - 2: 关闭白灯, 但改变红灯、蓝灯、绿灯的颜色
 - 3: 关闭红灯、蓝灯、绿灯, 但改变白灯的亮度
- "y" 表示 PWM 周期, 单位: 毫秒。
- "z"表示红绿蓝的颜色信息。
- 如果使用 *ESP8266_NONOS_SDK_V2.0* 及之前的版本, ESP8266 设备回复乐鑫云的响应格式如下:

```
{"nonce": 5619936, "datapoint": {"x": 1, "y": 1000, "z":100, "k":1, "l":2}, "deliver_to_device": true}
```



当 ESP8266 设备收到乐鑫云发来的 POST 命令时, ESP8266 设备需要根据乐鑫云的命令, 改变自身的状态。例如, 设置智能灯光调色的命令格式如下:

```
{"body": {"datapoint": {"y": 2, "x": 1, "k": 4, "z": 3, "l": 5}},
"nonce": 65470541, "mdev_mac": "18fe34ed861c", "meta":
{"Authorization": "token HERE_IS_THE_OWNER_KEY", "Time-Zone": "Asia/
Kashgar"}, "path": "/v1/datastreams/light/datapoint/", "method":
"POST", "deliver_to_device": true}
```

ESP8266 智能灯完成动作后,向乐鑫云回复状态更新成功的响应,格式如下:

• 如果使用 *ESP8266_NONOS_SDK_V2.0* 之后的版本:

```
{"status": 200, "nonce": 65470541, "datapoint": {"x": 1, "y": 1000, "z": {"red": 0, "green": 0, "blue": 0, "white": 255}}, "deliver_to_device": true, "mdev_mac": "18FE34ED861C"}
```

说明:

- "x" 表示智能灯的状态:
 - 0: 关灯
 - 1: 开灯
 - 2: 关闭白灯, 但改变红灯、蓝灯、绿灯的颜色
 - 3: 关闭红灯、蓝灯、绿灯, 但改变白灯的亮度
- "v" 表示 PWM 周期, 单位: 毫秒。
- "z"表示红绿蓝的颜色信息。
- 如果使用 *ESP8266_NONOS_SDK V2.0* 及之前的版本:

```
{"status": 200, "nonce": 65470541, "datapoint": {"x": 1, "y": 1000, "z": 1, "k":2, "l":3}, "deliver_to_device":true, "mdev_mac": "18FE34ED861C"}
```

PC:

查询灯的状态

用户使用 PC 向乐鑫云发送如下 curl 指令,查询智能灯的状态。

```
curl -X GET -H "Content-Type:application/json" -H "Authorization:
token HERE_IS_THE_OWNER_KEY" http://iot.espressif.cn/v1/datastreams/
light/datapoint/
```

响应:

• 如果使用 *ESP8266_NONOS_SDK_V2.0* 之后的版本:

```
{"status": 200, "datapoint": {"updated": "2016-07-05 15:06:17", "created": "2016-07-05 15:06:17", "datatype": 0, "k": 0.0, "visibly": 1, "datastream_id": 7969, "at": "2016-07-05 15:06:17", "y": 1000,
```



```
"x": 2, "z": {"red": 0, "green": 255, "blue": 0, "white": 0} , "id":
4131591, "l": 0.0}}
```

■ 说明:

- "x"表示智能灯的状态:
 - 0: 关灯
 - 1: 开灯
 - 2: 关闭白灯, 但改变红灯、蓝灯、绿灯的颜色
 - 3: 关闭红灯、蓝灯、绿灯、但改变白灯的亮度
- "y" 表示 PWM 周期, 单位: 毫秒。
- "z"表示红绿蓝的颜色信息。
- 如果使用 *ESP8266_NONOS_SDK_V2.0* 及之前的版本:

```
{"status": 200, "datapoint": {"updated": "2016-07-05 15:06:17", "created": "2016-07-05 15:06:17", "datatype": 0, "k": 0.0, "visibly": 1, "datastream_id": 7969, "at": "2016-07-05 15:06:17", "y": 0.0, "x": 1000.0, "z": 0.0, "id": 4131591, "l": 0.0}}
```

设置灯的状态

用户使用 PC 向乐鑫云发送如下 curl 指令,设置智能灯的状态。

• 如果使用 *ESP8266_NONOS_SDK_V2.0* 之后的版本:

Linux/Cygwin curl:

```
curl -X POST -H "Content-Type:application/json" -H "Authorization:
token HERE_IS_THE_OWNER_KEY" -d '{"datapoint":{"x": 2, "y": 1000,
"z": {"red": 0, "green": 255, "blue": 0, "white": 0}}} ' http://
iot.espressif.cn/v1/datastreams/light/datapoint/?deliver_to_device
=true
```

Windows curl:

```
curl -X POST -H "Content-Type:application/json" -H "Authorization:
token HERE_IS_THE_OWNER_KEY" -d "{\"datapoint\":{\"x\": 100, \"y\":
200, \"z\": {\"red\": 0, \"green\": 255, \"blue\": 0, \"white\":
0}}}" http://iot.espressif.cn/v1/datastreams/light/datapoint/?
deliver_to_device=true
```

响应:

```
{"nonce": 28541403.0, "status": 200, "mdev_mac": "5CCF7F14C682",
"datapoint": {"updated": "2016-07-28 13:51:54", "created":
"2016-07-28 13:51:54", "datatype": 0, "k": 0, "visibly": 1, "id":
4807512, "at": "2016-07-28 13:51:54", "y": 1000, "x": 2, "z": {"red":
```



```
0, "green": 255, "blue": 0, "white": 0}, "datastream_id": 7969, "l":
50}}
```

• 如果使用 *ESP8266 NONOS SDK V2.0* 及之前的版本:

Linux/Cygwin curl:

```
curl -X POST -H "Content-Type:application/json" -H "Authorization:
token HERE_IS_THE_OWNER_KEY" -d '{"datapoint":{"x": 100, "y": 200,
"z": 0, "k": 0, "l": 50}} ' http://iot.espressif.cn/v1/datastreams/
light/datapoint/?deliver_to_device=true
```

Windows curl:

```
curl -X POST -H "Content-Type:application/json" -H "Authorization:
token HERE_IS_THE_OWNER_KEY" -d "{\"datapoint\":{\"x\": 100, \"y\":
200, \"z\": 0, \"k\": 0, \"l\": 50}}" http://iot.espressif.cn/v1/
datastreams/light/datapoint/?deliver_to_device=true
```

响应:

```
{"nonce": 28541403.0, "status": 200, "mdev_mac": "5CCF7F14C682", "datapoint": {"updated": "2016-07-28 13:51:54", "created": "2016-07-28 13:51:54", "datatype": 0, "k": 0, "visibly": 1, "id": 4807512, "at": "2016-07-28 13:51:54", "y": 200, "x": 100, "z": 0, "datastream_id": 7969, "l": 50}}
```

5.6. 温湿度传感器

设备:

ESP8266 设备将主动上传温湿度信息到乐鑫云、格式如下:

```
{"nonce": 153436234, "path": "/v1/datastreams/tem_hum/datapoint/",
"method": "POST", "body": {"datapoint": {"x": 35, "y": 32}},
"meta": {"Authorization": "token HERE_IS_THE_MASTER_DEVICE_KEY"}}
```

温湿度信息上传成功后,乐鑫云返回如下响应:

```
{"status": 200, "datapoint": {"updated": "2014-05-14 18:42:54",
"created": "2014-05-14 18:42:54", "visibly": 1, "datastream_id":
16, "at": "2014-05-14 18:42:54", "y": 32, "x": 35, "id": 882644}}
```

道 说明:

- "x" 表示温度值, "y" 表示湿度值。
- 乐鑫云返回的响应信息中,带有数据更新的最后时间戳。

PC:

用户使用 PC 向乐鑫云发送如下 curl 指令,查询最新的温湿度数据。



curl -X GET -H "Content-Type:application/json" -H "Authorization: token HERE_IS_THE_OWNER_KEY" http://iot.espressif.cn/v1/datastreams/ tem hum/datapoint

说明:

上述命令如果返回 "remote device is disconnect or busy",是正常情况,因为温湿度传感器不支持反向操作。

用户使用 PC 向乐鑫云发送如下 curl 指令,查询温湿度的历史数据。

curl -X GET -H "Content-Type:application/json" -H "Authorization:
token HERE_IS_THE_OWNER_KEY" http://iot.espressif.cn/v1/datastreams/
tem hum/datapoints

5.7. 用户自定义反向控制

乐鑫云支持开发者自定义反向控制的行为,即支持发送任意自定义的 action 到 ESP8266 设备,并且支持附带参数,实现灵活的反向控制。

指令格式如下:

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=your_custom_action&any_parameter=any_va
lue'

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=your_custom_action&any_parameter=any_va
lue"

ESP8266 设备收到的数据如下:

```
{"body": {}, "nonce": 872709859, "get": {"action":
    "your_custom_action", "any_parameter": "any_value",
    "deliver_to_device": "true"}, "token": "HERE_IS_THE_DEVICE_KEY",
    "meta": {"Authorization": "token HERE_IS_THE_DEVICE_KEY "}, "path":
    "/v1/device/rpc/", "post": {}, "method": "GET",
    "deliver_to_device": true}
```



说明:

- 上述红色标注的为可由开发者自定义的控制行为及参数。
- 在 ESP8266 设备应用程序 (IoT_Demo) 中新增解析 action 及 parameter, 并实现自定义功能即可。
- 如上的自定义反向控制行为,不支持保存历史记录。



乐鑫 IOT 团队 www.espressif.com

免责申明和版权公告

本文中的信息,包括供参考的 URL 地址,如有变更,恕不另行通知。

文档"按现状"提供,不负任何担保责任,包括对适销性、适用于特定用途或非侵权性的任何担保,和任何提案、规格或样品在他处提到的任何担保。本文档不负任何责任,包括使用本文档内信息产生的侵犯任何专利权行为的责任。本文档在此未以禁止反言或其他方式授予任何知识产权使用许可,不管是明示许可还是暗示许可。

Wi-Fi 联盟成员标志归 Wi-Fi 联盟所有。蓝牙标志是 Bluetooth SIG 的注册商标。 文中提到的所有商标名称、商标和注册商标均属其各自所有者的财产,特此声明。

版权归© 2016 乐鑫所有。保留所有权利。