# Espressif IOT Demo

## Smart Light/Plug/Sensor

**Version 1.0.1**

Espressif Systems IOT Team

# Table of Content

# 1. 前言

基于 ESP8266 物联网平台的 IoT SDK 为用户提供了一个简单、快速、高效开发物联网产品的软件平台。本文主要介绍基于 Espressif IoT SDK 的嵌入式应用开发，在 IoT Demo 中，实现了三类产品：智能插座、智能灯、传感器，并且通过外网服务器，实现了对设备的反向控制以及数据的采集。

通过对本文的熟悉，用户可以快速的开发类似应用产品。

# 2. 概述

SDK 为用户提供了一套数据接收、发送的函数接口，用户不必关心底层网络，如 Wi-Fi、TCP/IP 等的具体实现，只需要专注于物联网上层应用的开发，利用相应接口完成网络数据的收发即可。

ESP8266 物联网平台的所有网络功能均在库中实现，对用户不透明。用户应用的初始化功能可以在 `user_main.c` 中实现。

`void user_init(void)`是上层程序的入口函数，给用户提供一个初始化接口，用户可在该函数内增加硬件初始化、网络参数设置、定时器初始化等功能。

SDK中提供了对 json 包的处理 API，用户也可以采用自定义数据包格式，自行对数据进行处理。

## 2.1. 代码结构

### 1. "usr"目录

usr 目录下为 IoT Demo 应用示例的功能实现代码，具体如下：

`user_main.c` – 程序主入口；

`user_webserver.c` – 创建 TCP 服务器的示例，提供 REST 的轻量 webserver 功能；

`user_devicefind.c` – UDP 传输功能的示例，提供 ESP8266 设备查找功能；

`user_esp_platform.c` – 与 Espressif 云端服务器通信的示例；

`user_json.c` – json 包的处理示例；

`user_plug.c` – 智能插座的功能示例代码；

`user_light.c` – PWM 实现智能灯的功能示例代码；

`user_humiture.c` – 温湿度传感器的功能示例代码；

### 2. "include"目录

include目录下为应用程序相关头文件，需要注意的是"`user_config.h`" 文件，在该头文件中可以对采用平台，以及具体的应用示例进行选择。

具体支持如下例子：

PLUG_DEVICE（智能插座），LIGHT_DEVICE（灯），SENSOR_DEVICE（传感器）；

其中 SENSOR_DEVICE （传感器）又分为 HUMITURE_SUB_DEVICE（温湿度传感器）和 FLAMMABLE_GAS_SUB_DEVICE（可燃气体检测）。

#### 3. "driver"文件夹

目前外围驱动支持 I2C Master，SPI，外部按键， PWM，双 UART。

#### 4. 工作模式

IoT Demo 在 `user_esp_platform_init` 中设置初始工作模式为 sotfAP+station 共存的模式，用户连入 ESP8266 softAP 接口的局域网，发指令让 ESP8266 station 接口连接可入外网的路由（AP），用户可通过向 ESP8266 softAP 接口发指令查询 ESP8266 station 连接 WiFi 的状况。ESP8266 station 接口连入路由后，自动连接 Espressif 云端服务器，对应代码 `user_esp_platform_check_ip` 。连接服务器完成后，切换进入station模式。

ESP8266 softAP 的 SSID 默认为 **ESP_XXXXXX**，其中 **XXXXXX** 为设备 MAC 地址的后三个字节，默认加密模式为 WPA/WPA2。

在 station 模式下，长按按键5秒，设备即复位并重启恢复初始 softAP+station 共存模式，可重新进行配置。

## 2.2. 调试工具

IoT Demo 在 user_webserver.c 内建立的 TCP server 采用 REST 架构，用户通过 PC 端与 IoT Demo 设备进行通讯时，可采用 curl 命令。

可在链接（ http://curl.haxx.se/download.html）进行指定版本的下载，后文中的 curl 指令请参照 “Windows curl” 的示例。

若使用 Linux curl 或者 Cygwin curl，后文中的 curl 指令请参照 “Linux/Cygwin curl” 的示例。

如无特别说明，则表示可以通用。

#### 1. curl 指令使用的常见错误

- 注意 curl 指令中的字符大小写，若大小写出错，则指令出错。
- curl 指令中均为英文标点符号，若指令夹杂了中文符号，则指令出错。
- curl 指令中的空格，若未打空格，或者多打成两个空格，则指令出错。
- 随机 token 不能与其他设备共用。
- 根据发 curl 指令的工具（Linux/Cygwin or Windows）不同，注意选择正确的指令格式。

# 3. 局域网功能

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 “ip” 信息指 softAP 和 station 模式下的 IP，需输入实际的 IP 地址。

ESP8266 softAP需要密码进行连接，密码格式为：ESP8266设备softAPMAC_PASSWORD，开发者可自行修改 esp_iot_sdk\app\include\user_config.h 中宏定义 PASSWORD，配置密码。

例如:

宏定义 "PASSWORD" 定义为 v*%W>L<@i&Nxe!

ESP8266 设备 softAP MAC 地址为 1a:fe:86:90:d5:7b

则 ESP8266 softAP 的 WiFi 连接密码为 1a:fe:86:90:d5:7b_v*%W>L<@i&Nxe!

## 3.1. 通用功能

### 1. 查询版本信息

```
curl –X GET http://ip/client?command=info
响应:
{
        "Version": {
                "hardware":"0.1",
                "software":"0.8.0"
        },
        "Device": {
                "product":"Plug",
                "manufacture",  "Espressif Systems"
        }
}
```

### 2. 设置连接参数

ESP8266 设备初始状态 softAP+station 模式，将 PC 连入设备 softAP 提供的局域网（密码如前述），通过 PC 发送 curl 指令设置。

#### 设置 station 模式

PC 连入 ESP8266 softAP 发送下述 curl 指令，设置 ESP8266 station 连接 AP 。

**Linux/Cygwin curl:**

```
curl -X POST -H "Content-Type:application/json" -d '{"Request":{"Station":
{"Connect_Station":{"SSID":"tenda", "password":"1234567890", "token":
"12345678901234567890123456789012345678901234567890"}}}}' http://192.168.4.1/config?
command=wifi
```

**Windows curl:**

```
curl -X POST -H "Content-Type:application/json" -d "{/"Request/":{/"Station/":
{/"Connect_Station/":
{/"SSID/":/"tenda/", /"password/":/"1234567890/", /"token/": /"12345678901234567890123456789012345678901234567890/"}}}}" http://192.168.4.1/config?command=wifi
```

设置完成后，ESP8266 连接指令中的路由。

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。ESP8266 设备后续会用此随机 token 向 Espressif Cloud 激活、认证；用户使用同一个随机 token 向 Espressif Cloud 申请该设备的控制权限。因此，随机 token 与 ESP8266 设备是一一对应的关系，不能与其他设备共用。

另有特殊情况：

若 AP（路由）的加密方式为 WEP HEX，则密码需要转为 ASC 码 HEX 值。

举例如下：

假设路由 SSID “wifi_1”，密码为 “tdr0123456789”，加密方式为 WEP，则

**Linux/Cygwin curl**:

```
curl -X POST -H Content-Type:application/json -d '{"Request":{"Station":
{"Connect_Station":{"SSID":"wifi_1", "password":"74647230313233343536373839",
"token": "12345678901234567890123456789012345678901234567890"}}}}' http://192.168.4.1/config?
command=wifi
```

**Windows curl**:

```
curl -X POST -H "Content-Type:application/json" -d "{/"Request/":{/"Station/":
{/"Connect_Station/":
{/"SSID/":/"wifi_1/", /"password/":/"74647230313233343536373839/", /"token/": /"12345678901234567890123456789012345678901234567890/"}}}}" http://192.168.4.1/config?command=wifi
```

在配置 ESP8266 station接口连接路由的过程中，可通过如下 curl 指令，查询设备的连接状态。

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

返回 status 说明如下:
```

```
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
};
```

对于智能插座或者智能灯，这种支持反向控制的设备，可发如下指令让 ESP8266 设备重启：

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

对于温湿度传感器，这种不支持反向控制的设备，可发如下指令让 ESP8266 设备休眠：

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

温湿度传感器休眠 30 秒后，自动唤醒。

### 设置 softAP 参数

发送如下 curl 指令，设置 ESP8266 softAP 的参数，例如 SSID、password 等。

**Linux/Cygwin curl**:

```
curl -X POST -H "Content-Type:application/json" -d '{"Request":{"Softap":
{"Connect_Softap":{"authmode":"OPEN", "channel":6, "SSID":"ESP_IOT_SOFTAP",
"password":""}}}}' http://192.168.4.1/config?command=wifi
```

**Windows curl**:

```
curl -X POST -H "Content-Type:application/json" -d "{/"Request/":{/"Softap/":
{/"Connect_Softap/":{/"authmode/":/"OPEN/", /"channel/":
6, /"SSID/":/"ESP_IOT_SOFTAP/", /"password/":/"/"}}}}" http://192.168.4.1/config?
command=wifi
```

**注意**:

authmode 支持: OPEN, WPAPSK, WPA2PSK, WPAPSK/WPA2PSK.

password 必须多于 8 bytes.

3. WiFi 连接与工作模式的切换

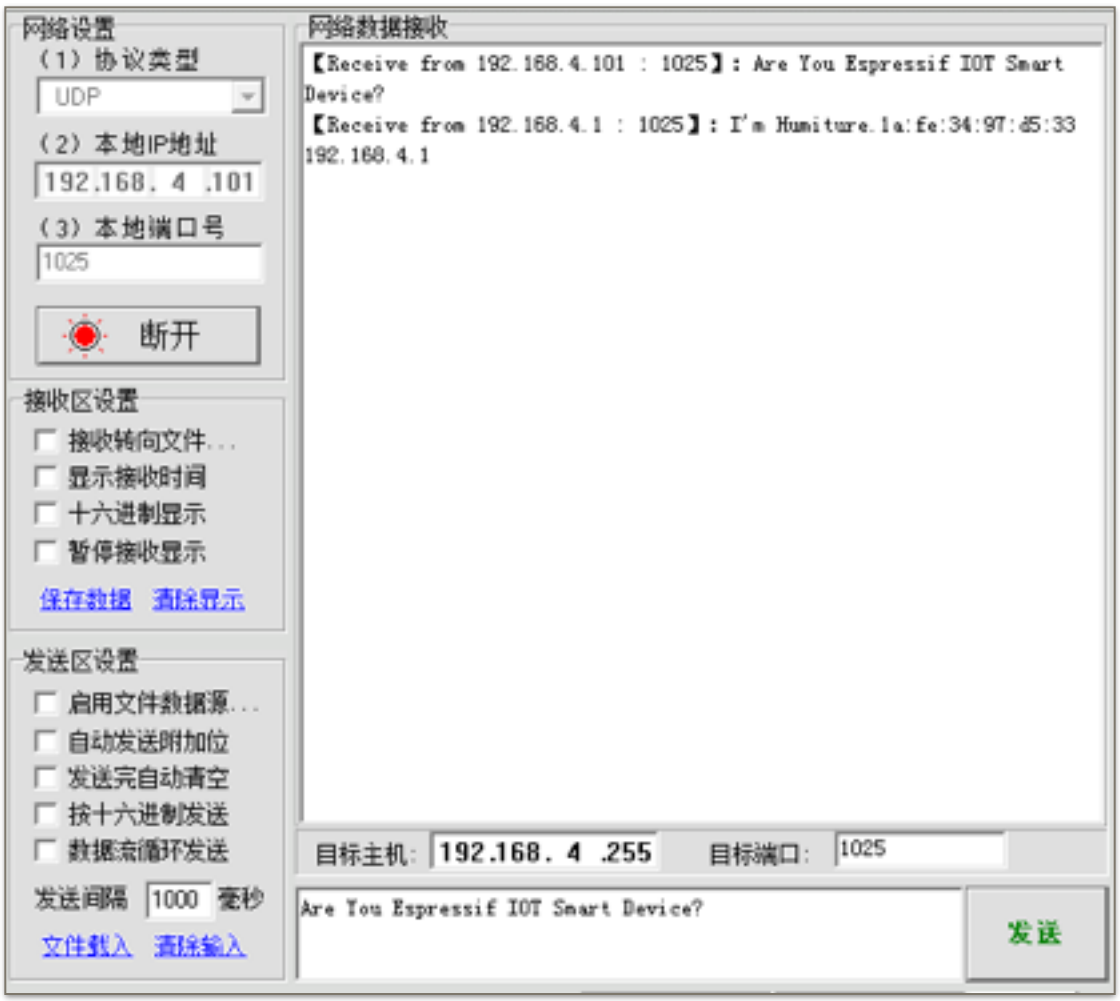
- 初始上电时，默认为 softAP + station 共存模式。
- 手机 APP (或 PC ) 连入 ESP8266 softAP 发 curl 指令，让 ESP8266 station 连接路由。过程中可以查询 station 的连接状态。
- ESP8266 station 连上路由后，向服务器认证，通过后，ESP8266 切换为单独 station 模式。
- 之后，ESP8266 保持 station 模式。当网络连接断开，尝试重连无效，ESP8266 切回 softAP + station 模式。此时可以重新从步骤2进行连接。

另，esp_iot_sdk_v0.9.2 及之后版本，支持网络连接失败时，自动切换已记录的 AP 进行连接，由宏定义 #define AP_CACHE 控制此功能开关。

## 3.1. 局域网内查找 ESP8266

PC 可以通过在局域网内向端口 1025 发送 UDP 广播包的方法进行 ESP8266 设备的查找，发送广播包信息 “Are You Espressif IOT Smart Device?”，ESP8266 设备对在 1025 端口收到的 UDP 广播包进行判断，如为该字符串，则回复响应。

可利用网络调试助手来测试此功能，例如：



响应：

- 智能插座

I’m Plug.xx:xx:xx:xx:xx:xxyyy.yyy.yyy.yyy

- 智能灯

I’m Light.xx:xx:xx:xx:xx:xxyyy.yyy.yyy.yyy

- 温湿度传感器

I’m Humiture.xx:xx:xx:xx:xx:xxyyy.yyy.yyy.yyy

其中 xx:xx:xx:xx:xx:xx 为设备 MAC 地址，yyy.yyy.yyy.yyy 为设备 IP 地址。

如不为该字符串，则不做响应。

## 3.2. 智能插座

### 1. 查询插座状态

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

响应

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

status 可以为 0 或者 1。

### 2. 设置插座状态

**Linux/Cygwin curl**:

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

**Windows curl**:

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

status 可以为 0 或者 1。

### 3.3. 智能灯

#### 1. 查询灯的状态

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

响应:

```
{
    "freq": 100,
    "rgb": {
        "red": 100,
        "green": 0,
        "blue": 0
    }
}
```

其中，freq 取值范围为1~500；red、green、blue 取值范围为0~255。

#### 2. 设置灯的状态

**Linux/Cygwin curl**:

```
curl –X POST –H "Content-Type:application/json" –d '{"freq":100, "rgb":{"red":200,
"green":0, "blue":0}}' http://ip/config?command=light
```

**Windows curl**:

```
curl –X POST –H "Content-Type:application/json" –d "{/"freq/":100, /"rgb/":{/"red/":
200, /"green/":0, /"blue/":0}}" http://ip/config?command=light
```

其中，freq 取值范围为1~500；red、green、blue 取值范围为0~255。

### 3.4. 温湿度传感器

温湿度传感器的状态需要在广域网下通过 Espressif Cloud 获取。

# 4. 广域网功能

## 4.1. Espressif Cloud

关于 Espressif 服务器平台的详细使用，会在 Espressif 服务器上提供详细的操作及API介绍。

注意:

- 后述"**设备**"指设备自行完成的动作，无需用户操作；
- 后述"**PC**"指用户可通过PC发指令，进行操作。

### master-device-key

ESP8266 设备凭借它作为身份认证，享受 Espressif Cloud 提供的云端服务，master-device-key.bin 需要向 Espressif Cloud 申请，烧录到 SPI flash。

### 激活

### 设备

ESP8266 设备根据 curl 命令设置的 ssid、password 及随机 token 连接路由后，会默认向 Espressif Cloud 激活认证。

激活需要往 Espressif Cloud (IP 地址: 115.29.202.58，端口: 8000) 发送如下格式的 TCP 包:

```
{"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"}}
```

HERE_IS_THE_MASTER_DEVICE_KEY 为烧录到 ESP8266 设备 SPI flash 的实际 master-device-key 值，1234567890123456789012345678901234567890 为之前 3.1.2 设置连接参数 中设置的随机 token

**响应**

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

### PC

PC 侧在配置 ESP8266 设备成功连接路由后，PC 也同样需要连接到一个可上外网的路由，向 Espressif Cloud 申请设备的控制权。

**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/
```

**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"}}
```

c8922638bb6ec4c18fcf3e44ce9955f19fa3ba12 为 user key（用户身份 ID）的举例，需填入用户实际的 user key 值，在 Espressif Cloud 注册用户时获得。 步骤如下:

- 注册并登陆 Espressif Cloud http://iot.espressif.cn/
- 点击右上角的用户名
- 点击进入设置
- 点击 “开发者”

e474bba4b8e11b97b91019e61b7a018cdbaa3246 为返回的设备 owner key，在 PC 侧使用 owner key 对设备进行控制。

### 1. 认证

激活后，设备向 Espressif Cloud (IP 地址: 115.29.202.58，端口: 8000) 发送如下格式的 TCP 包，进行认证:

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

这个 TCP 包的作用是，ESP8266 设备向 Espressif Cloud 认证自身的身份，每次 ESP8266 设备重新连接 Espressif Cloud 都需要向服务器发送这样一包数据。其中“nonce”是一组随机整数，token 后面是设备的 master-device-key。

Espressif Cloud 认证设备持有的确实是服务器发布的 master-device-key 后，会向设备回复一个身份确认成功的数据包。

**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}
```

认证过程在智能插座和灯的应用中需要。

### 2. PING 服务器

为了保持 ESP8266 设备与 Espressif Cloud 之间的 socket 连接，ESP8266 需要每 50 秒向 Espressif Cloud (IP 地址: 115.29.202.58，端口: 8000) 发送如下格式的的 TCP 包。

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

**响应**:

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

PING 服务器的机制，在智能插座及灯这种需要进行反向控制的设备中进行。

### 3. 智能插座

#### 设备

在进行对设备的反向控制时，存在如下两种情况：

- ESP8266 设备收到云端服务器发来的 GET 命令时，表示设备需要将自身的状态上传至服务器，服务器发给设备的 GET 命令格式如下所示：

```
{"body": {}, "nonce": 33377242, "is_query_device": true, "get": {}, "token":
"e474bba4b8e11b97b91019e61b7a018cdbaa3246", "meta": {"Authorization": "token
e474bba4b8e11b97b91019e61b7a018cdbaa3246"}, "path": "/v1/datastreams/plug-status/
datapoint/", "post": {}, "method": "GET"}
```

**响应**:

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

- ESP8266 设备收到云端服务器发来的 POST 命令时，表示设备需要改变自身状态。服务器相关的数据包实现对应的控制动作，例如，打开智能插座开关的命令：

```
{"body": {"datapoint": {"x": 1}}, "nonce": 620580862, "is_query_device": true,
"get": {}, "token": "e474bba4b8e11b97b91019e61b7a018cdbaa3246", "meta":
{"Authorization": "token e474bba4b8e11b97b91019e61b7a018cdbaa3246"}, "path": "/v1/
datastreams/plug-status/datapoint/", "post": {}, "method": "POST",
"deliver_to_device": true}
```

ESP8266 智能插座完成控制动作后，向服务器发送一个更新状态成功的响应，格式如下，响应回复的 nonce 值必须与云端服务器之前发送的控制命令中的 nonce 值一致，以表示每次控制和响应相互对应。

**响应**:

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

#### PC

**查询智能插座状态**

```
curl -X GET -H "Content-Type:application/json" -H "Authorization: token
e474bba4b8e11b97b91019e61b7a018cdbaa3246" http://iot.espressif.cn/v1/datastreams/
plug-status/datapoint/
```

**响应**:

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

设置智能插座状态

**Linux/Cygwin curl**:

```
curl -X POST -H "Content-Type:application/json" -H "Authorization: token
e474bba4b8e11b97b91019e61b7a018cdbaa3246" -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
e474bba4b8e11b97b91019e61b7a018cdbaa3246" -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}
```

### 4. 智能灯

设备

在进行对设备的反向控制时，存在如下两种情况：

- ESP8266 设备收到云端服务器发来的 GET 命令时，表示设备需要将自身的状态上传至服务器，服务器发给设备的 GET 命令格式如下所示：

```
{"body": {}, "nonce": 8968711, "is_query_device": true, "get": {}, "token":
"e474bba4b8e11b97b91019e61b7a018cdbaa3246", "meta": {"Authorization": "token
e474bba4b8e11b97b91019e61b7a018cdbaa3246"}, "path": "/v1/datastreams/light/
datapoint/", "post": {}, "method": "GET"}
```

响应:

```
{"nonce": 5619936, "datapoint": {"x": 100, "y": 200, "z": 0, "k": 0, "l": 50},
"deliver_to_device": true}
```

- ESP8266 设备收到云端服务器发来的 POST 命令时，表示设备需要改变自身状态。服务器相关的数据包实现对应的控制动作，例如，设置智能灯光调色的命令：

```
{"body": {"datapoint": {"y": 200, "x": 100, "k": 0, "z": 0, "l": 50}}, "nonce":
5619936, "is_query_device": true, "get": {}, "token":
"e474bba4b8e11b97b91019e61b7a018cdbaa3246", "meta": {"Authorization": "token
e474bba4b8e11b97b91019e61b7a018cdbaa3246"}, "path": "/v1/datastreams/light/
datapoint/", "post": {}, "method": "POST"
```

响应:

```
{"nonce": 5619936,  "datapoint": {"x": 100, "y": 200, "z": 0, "k": 0, "l": 50},
"deliver_to_device": true}
```

其中，X 表示频率，取值范围 1~500；Y (red)，Z (green)，K (blue) 调节智能灯的颜色，取值范围 0~255；L 为保留参数。

PC

查询灯的状态

```
curl -X GET -H "Content-Type:application/json" -H "Authorization: token
e474bba4b8e11b97b91019e61b7a018cdbaa3246" http://iot.espressif.cn/v1/datastreams/
light/datapoint
```

响应:

```
{"nonce": 5619936,  "datapoint": {"x": 100, "y": 200, "z": 0, "k": 0, "l": 50},
"deliver_to_device": true}
```

设置灯的状态

**Linux/Cygwin curl**:

```
curl -X POST -H "Content-Type:application/json" -H "Authorization: token
e474bba4b8e11b97b91019e61b7a018cdbaa3246" -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
e474bba4b8e11b97b91019e61b7a018cdbaa3246" -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": 5619936,  "datapoint": {"x": 100, "y": 200, "z": 0, "k": 0, "l": 50},
"deliver_to_device": true}
```

其中，X 表示频率，取值范围 1~500；Y (red)，Z (green)，K (blue) 调节智能灯的颜色，取值范围 0~255；L 为保留参数。

### 5. 温湿度传感器

设备

上传温湿度数据到 Espressif Cloud:

```
{"nonce": 1,  "path": "/v1/datastreams/tem_hum/datapoint/",  "method": "POST",
"body": {"datapoint": {"x": 35, "y": 32}},  "meta": {"Authorization": "token
HERE_IS_THE_MASTER_DEVICE_KEY"}}
```

X 表示温度值，Y 表示湿度值。

温湿度信息上传成功后，云端服务器返回如下响应:

**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}}
```

响应信息中携带数据更新的最后时间戳。

### PC

PC 侧通过如下两类 API 查询温湿度传感器的最新数据和历史数据，其中，红色字体是用户的 owner key。

- 查询最新数据:

```
curl -X GET -H "Content-Type:application/json" -H "Authorization: token
e474bba4b8e11b97b91019e61b7a018cdbaa3246" http://iot.espressif.cn/v1/datastreams/
tem_hum/datapoint
```

**注意**: 上述命令在 owner key 下会返回 “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。

- 查询历史数据:

```
curl -X GET -H "Content-Type:application/json" -H "Authorization: token
e474bba4b8e11b97b91019e61b7a018cdbaa3246" http://iot.espressif.cn/v1/datastreams/
tem_hum/datapoints
```

## 4.2. 用户自定义反向控制

Espressif 云端服务器支持用户自定义行为，可以发送任意的 action 到设备，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_value'
```

**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_value"
```

红色为可自定义部分。在 ESP8266 IoT Demo 中新增解析 action 及 parameter，并实现自定义的功能即可。

设备侧收到数据如下:

```
{"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}
```

**注意**：RPC 指令只实现灵活的反向控制，不保存历史记录。

例如，用户可以自定义 custom_action 控制智能风扇摇头，但云端服务器不会记录智能风扇摇了几次头，之前几点钟在摇头，几点钟停止摇头的历史信息。