Espressif IoT Demo 使用手册

Status	Released
Current version	V0.5
Author	Fei Yu
Completion Date	2014.9.24
Reviewer	Jiangang Wu
Completion Date	2014.9.24

CONFIDENTIAL

[ ] CONFIDENTIAL
[ ] INTERNAL
[√] PUBLIC

版本信息

日期	版本	撰写人	审核人	修改说明
2014.5.13	0.1	Jiangang Wu		草稿
2014.6.16	0.2	Fei Yu		更新文档指令
2014.7.10	0.3	Fei Yu		增加 RPC
2014.8.14	0.4	Fei Yu		更新文档指令
2014.9.24	0.5	Fei Yu		增加 WEP HEX curl 连接指令说明

CONFIDENTIAL

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。

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© 2014 乐鑫信息技术有限公司所有。保留所有权利。

目录

CONFIDENTIAL

1. 前言

本文主要介绍基于 Espressif IoT SDK 的嵌入式应用开发，在该 IoT Demo 中，实现了三类产品：插座、灯、传感器，并且基于外网服务器，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。

CONFIDENTIAL

2. 总体介绍

2.1. 代码结构

2.1.1. usr 目录

usr 目录下为 IoT Demo 功能实现代码，具体如下：
user_main.c——主入口文件；
user_webserver.c——提供 REST 的轻量 webserver 功能；
user_devicefind.c——提供设备查找功能；
user_esp_platform.c——基于 espressif 服务器的应用功能；
user_json.c——json 包处理功能；
user_plug.c——插座相关功能；
user_light.c——pwm 灯相关功能；
user_humiture.c——温湿度计相关功能；

2.1.2. include 目录

include 目录下为相关头文件，需要注意的是 user_config.h 文件，在该文件中可以对采用平台，以及具体 demo 进行选择，具体支持如下例子：

PLUG_DEVICE（开关/插座），LIGHT_DEVICE（灯），SENSOR_DEVICE（传感器）；其中 SENSOR_DEVICE 又分为 HUMITURE_SUB_DEVICE（温湿度计）和 FLAMMABLE_GAS_SUB_DEVICE（可燃气体检测）。

2.1.3. driver 目录

driver 当前支持 i2c master，外部按键，pwm，双 uart。

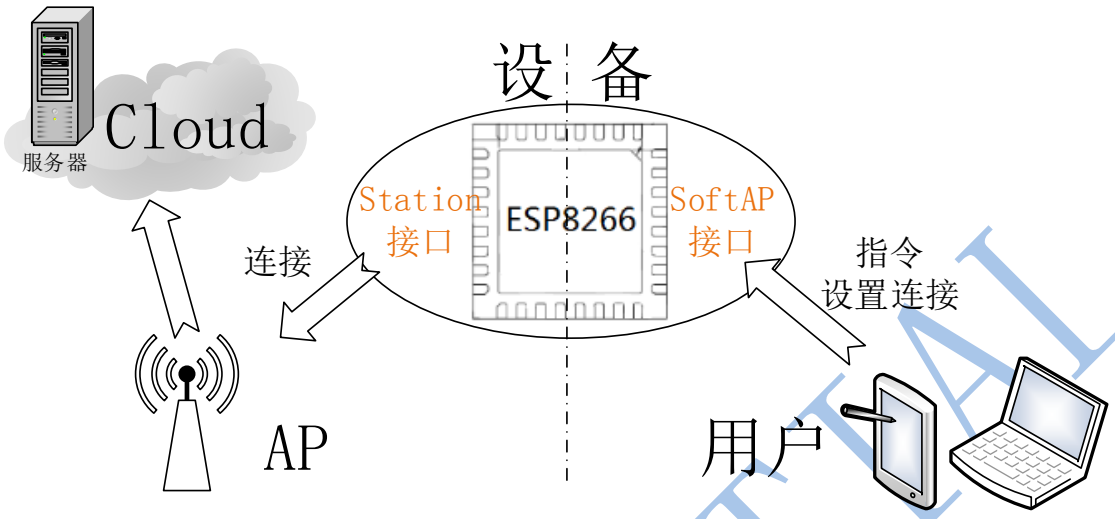
2.2. 工作模式

IoT Demo 中所有应用的 wifi 工作模式均为 softAP+station -> station 。出厂默认为 sotfAP+station 共存的模式，用户连入 softAP 接口的局域网，发指令让 station 接口连接可入外网的路由（AP），用户可通过向 softAP 接口发指令查询 station 的连接状况。Station 接口连接服务器完成后，用户可通过指令让设备 reboot（或 sleep）进入 station 模式（指令详见 3.1.2 设置连接参数）。

softAP 的 SSID 默认为 ESP_XXXXXX，其中 XXXXXX 为设备 MAC 地址的后面三个字节，默认加密模式为 WPA/WPA2。

CONFIDENTIAL

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。



2.3. 调试工具

IoT Demo 内的 server 采用 REST 架构，PC 端在与 IoT Demo 设备进行通讯时，可采用 curl 命令。

可在根据链接（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。

Curl 指令常见错误：

1） 需注意 curl 指令中的字符大小写，若大小写出错，则指令出错。

2） 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。

3） 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。

4） 随机 token 不能与其他设备共用。

5） 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。

CONFIDENTIAL

3. 局域网功能

softAP 模式默认 ip 为 192.168.4.1，station 模式下 ip 由路由分配。以下 URL 中 ip 信息代表了 softAP 和 station 模式下的 ip，需输入实际的 ip 地址。

Espressif softAP 需要密码进行连接，密码格式为：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，esp_iot_sdk_v.08 定义宏 PASSWORD 为“v*%W>L<@i&Nxe!”，某设备的 softAP MAC 地址为：1a:fe:86:90:d5:7b，则连接密码为：1a:fe:86:90:d5:7b_v*%W>L<@i&Nxe!

3.1. 通用功能

3.1.1. 获取版本信息

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

返回

```
{
    "Version": {
         "hardware":"0.1",
         "software":"0.8.0"
    },
    "Device": {
         "product":"Plug",
          "manufacture", "Espressif Systems"
    }
}
```

3.1.2. 设置连接参数

设备初始状态为 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 侧发送 curl 指令来控制设备。

- 设置 station 模式

PC 发送下述指令，将设备连入外网。

Linux/Cygwin curl：

```
curl -X POST -H "Content-Type:application/json" -d '{"Request":{"Station":{"
Connect_Station":{"ssid":"tenda","password":"1234567890","token": "12345678
90123456789012345678901234567890"}}}}' http://192.168.4.1/config?command=wi
fi
```

Windows curl：

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

设置完成后，设备去连接指令中的路由。

注：

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。设备后续会用此随机 token 向 Espressif Server 激活、认证；用户使用同一个随机 token 向 Espressif Server 申请该设备的控制权限。因此，随机 token 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 码的 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":{"Co
nnect_Station":{"ssid":"wifi_1","password":"7464723031323334353637383 9","to
ken": "12345678901234567890123456789012345678901234567890"}}}}' http://192.168.4.1/co
nfig?command=wifi
```

Windows curl：

```
curl -X POST -H "Content-Type:application/json" -d "{\"Request\":{\"Station\
":{\"Connect_Station\":{\"ssid\":\"wifi_1\",\"password\":\"7464723031323334
3536373839\",\"token\": \"12345678901234567890123456789012345678901234567890\"}}}}" h
ttp://192.168.4.1/config?command=wifi
```

在配置设备 station 接口连接路由的过程中，可通过 PC 侧发如下指令，查询设备的连接状态。

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

返回 status 的含义如下：

```
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
};
```

连接完成后，PC 侧发如下指令，将设备从 softAP+station 模式更改为 station 模式。

对于支持反向控制的设备，如开关、灯等，指令为：

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

对于不支持反向控制的设备，如温湿度计等，指令为：

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

温湿度计 sleep 30s 后会自动唤醒，模式更改为 station mode。

- 设置 softAP 参数

设备发送如下指令，设置 softAP 的参数，例如 ssid、password 等。

Linux/Cygwin curl：

```
curl -X POST -H "Content-Type:application/json" -d '{"Request":{"Softap":{"C
onnect_Softap":{"authmode":"OPEN","channel":6,"ssid":"ESP_IOT_SOFTAP","pass
word":""}}}}' 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
```

设置完后，设备需要重启才生效。

注意：

CONFIDENTIAL

authmode 支持如下模式：OPEN、WPAPSK、WPA2PSK、WPAPSK/WPA2PSK。password 长度需不小于 8 个字符。

3.1.3. 连接模式的切换

对于 esp_iot_sdk 的 IOT_Demo，连接过程中的模式切换如下：

（1） 初始上电时，默认为 softAP + station 共存模式。

（2） 手机 APP(或 PC)连入 ESP8266 softAP 下指令，让 ESP8266 station 连接路由。此时可以查询 station 的连接状态。

（3） 连上路由后，向服务器认证，通过后，ESP8266 切换为单独 station 模式。

（4） 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 之后，支持网络连接失败时，自动切换 AP 进行连接，由宏定义 #define AP_CACHE 控制此功能开关。

3.2. 局域网内设备查找

利用在局域网内向端口 1025 发送 UDP 广播包的方法进行设备的查找，发送信息为“Are You Espressif IOT Smart Device?”，设备对在 1025 端口收到的 UDP 广播包进行判断处理，如为该字符串，则做出正确响应。

可利用网络调试助手来测试此功能，例如：

CONFIDENTIAL

响应内容为：

- 插座

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

- 灯

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

- 温湿度

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

  其中 xx:xx:xx:xx:xx:xx 为开关 MAC 地址，yyy.yyy.yyy.yyy 为开关 ip 地址。

  如不为该字符串，则不做响应。

CONFIDENTIAL

3.3. 插座

3.3.1. 获取插座状态

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

返回

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

status 可以为 0 或 1。

3.3.2. 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 1。

3.4. 灯

3.4.1. 获取灯状态

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

返回

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

其中，freq 可以为 1~500，red、green、blue 可以为 0~255。

3.4.2. 设置灯状态

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 可以为 1~500，red、green、blue 可以为 0~255。

3.5. 温湿度

温湿度状态需要在广域网下通过 Espressif Server 获取。

CONFIDENTIAL

4. 广域网功能

4.1. espressif 服务器平台

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。

下述说明中，提到的“设备侧”指设备自行完成的动作，无需用户操作；“PC 侧”指用户可通过 PC 发指令，进行操作。

4.1.1. 关于设备的 master-device-key

根据 espressif 服务器的架构设计，每台设备均需要向 espressif 服务器申请一个 master device key，烧录到 spi flash 的 0x3e000 位置。

master device key 的获取方式，参见文档“如何创建 master-device-key 的 bin 文件”。

Bin 的烧写顺序：blank.bin -> eagle.app.v6.flash.bin -> master-device-key.bin -> eagle.app.v6.irom0text.bin，当不需要重新烧写 blank.bin 和 master-device-key.bin 时，可以直接 eagle.app.v6.flash.bin -> eagle.app.v6.irom0text.bin。

4.1.2. 激活

- 设备侧

在通过 softAP 接口设置了 ssid、password 及随机 token 后，设备 station 接口根据设置去连接路由，station 接口连上路由后，会默认向服务器激活设备。

激活需要往 espressif 的服务器，ip 地址为 114.215.177.97，端口为 8000，发送如下格式的的 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": "1234567890123456789012345678901234
567890"}}
```

其中的 HERE_IS_THE_MASTER_DEVICE_KEY 为存于 spi flash 的 device key，

CONFIDENTIAL

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 侧

PC 侧在配置完设备的 ssid、password 及 token 后需要连接到一个可上外网的路由，向服务器申请设备的控制权。

Linux/Cygwin curl：

```
curl -X POST -H "Authorization:token c8922638bb6ec4c18fcf3e44ce9955f19fa3ba
12" -d '{"token": "1234567890123456789012345678901234567890"}' http://114.2
15.177.97/v1/key/authorize/
```

Windows curl：

```
curl -X POST -H "Authorization:token c8922638bb6ec4c18fcf3e44ce9955f19fa3ba
12" -d "{\"token\": \"1234567890123456789012345678901234567890\"}" http://1
14.215.177.97/v1/key/authorize/
```

返回

```
{"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": "e474bba4b8e11
b97b91019e61b7a018cdbaa3246", "access_methods": "*", "is_owner_key": 1, "scope": 3,
 "device_id": 29, "activate_status": 1, "datastream_id": 0, "expired_at": "2288-02-2
2 20:31:47"}}
```

c8922638bb6ec4c18fcf3e44ce9955f19fa3ba12 为用户的 user key 举例，实际需要用户填入自己的 user key 值，获取方式如下：登陆 Espressif 服务器 http://iot.espressif.cn/ ，用户名密码登陆 -> 右上角用户名 -> 设置 -> 开发者。

e474bba4b8e11b97b91019e61b7a018cdbaa3246 为获取到的设备 owner key，后面在 PC 侧对设备进行控制时，需要采用这个 key。

4.1.3. 认证

设备激活完成后在连接 espressif 服务器时，需要往 espressif 的服务器，ip 地址为 114.215.177.97，端口为 8000，发送如下格式的的 tcp 包。

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

这个 tcp 的包的作用是确认设备自己的身份，每次设备重新连接服务器后的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 device key。

服务器收到设备发送的认证包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": 123
35, "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", "locati
on": "", "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-na
me-79eba060", "created": "2014-05-28 17:43:29", "is_frozen": 0, "key_id": 387}, "non
ce": 560192812, "message": "device identified", "status": 200}
```

认证过程在插座和灯的应用中需要。

4.1.4. PING 服务器

为了保持在长期不进行反向控制设备的情况下，保持 socket 的连接，需要在每 50s 往 espressif 的服务器，ip 地址为 114.215.177.97，端口为 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 服务器机制需要在插座及灯这样需要进行反向控制的设备中进行。

4.1.5. 插座

- 设备侧

在进行对设备的反向控制时，存在如下两种情况：

1.设备侧收到服务器发来的get命令时，表示设备需要将自身的状态发送至服务器上，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}
```

2.设备侧收到服务器发来的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来完成相应的控制动作，如打开开关命令：

```
{"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}
```

开关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 后面的值必须后服务器先前发送的控制命令中的 nonce 值一致，以表示每次控制和回应相互对应。

响应

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

- PC 侧

获取插座状态

```
curl -X GET -H "Content-Type:application/json" -H "Authorization: token e474
bba4b8e11b97b91019e61b7a018cdbaa3246" http://114.215.177.97/v1/datastreams/
plug-status/datapoint/
```

响应

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

设置插座状态

Linux/Cygwin curl：

```
curl -X POST -H "Content-Type:application/json" -H "Authorization: token e47
4bba4b8e11b97b91019e61b7a018cdbaa3246" -d '{"datapoint":{"x":1}}' http://11
4.215.177.97/v1/datastreams/plug-status/datapoint/?deliver_to_device=true
```

Windows curl：

```
curl -X POST -H "Content-Type:application/json" -H "Authorization: token e47
4bba4b8e11b97b91019e61b7a018cdbaa3246" -d "{\"datapoint\":{\"x\":1}}" http:
//114.215.177.97/v1/datastreams/plug-status/datapoint/?deliver_to_device=tr
ue
```

响应

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

4.1.6. 灯

✧ 设备侧

在进行对设备的反向控制时，存在如下两种情况：

1.设备侧收到服务器发来的get命令时，表示设备需要将自身的状态发送至服务器上，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_t
o_device": true}
```

2.设备侧收到服务器发来的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来完成相应的控制动作，如打开开关命令：

```
{"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_t
o_device": true}
```

其中 X 表示 freq 可以为 1~500，Y，Z，K 表示灯的不同颜色可以为 0~255 具体 Y 表示 red、Z 表示 green、K 表 blue。L 参数目前保留。

✧ PC 侧

获取灯状态

```
curl -X GET -H "Content-Type:application/json" -H "Authorization: token e474
bba4b8e11b97b91019e61b7a018cdbaa3246" http://114.215.177.97/v1/datastreams/
light/datapoint
```

响应

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

设置灯状态

Linux/Cygwin curl：

```
curl -X POST -H "Content-Type:application/json" -H "Authorization: token e47
4bba4b8e11b97b91019e61b7a018cdbaa3246" -d '{"datapoint":{"x": 100, "y": 200,
 "z": 0, "k": 0,"l": 50}} ' http://114.215.177.97/v1/datastreams/light/datap
oint/?deliver_to_device=true
```

Windows curl：

```
curl -X POST -H "Content-Type:application/json" -H "Authorization: token e47
4bba4b8e11b97b91019e61b7a018cdbaa3246" -d "{\"datapoint\":{\"x\": 100, \"y\
": 200, \"z\": 0, \"k\": 0, \"l\": 50}}" http://114.215.177.97/v1/datastream
s/light/datapoint/?deliver_to_device=true
```

返回

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

其中 X 表示 freq 可以为 1~500，Y，Z，K 表示灯的不同颜色可以为 0~255 具体 Y 表示 red、Z 表示 green、K 表 blue。L 参数目前保留。

4.1.7. 温湿度

✧ 设备侧

数据包上传，格式：

```
{"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"}}
```

温度值使用 X 来表示，湿度值使用 y 来表示，token 后面携带设备的 device key，数据上传成功后服务器会给设备上传成功的响应。

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": 3
2, "x": 35, "id": 882644}}
```

响应信息中携带数据更新的最后时间。

✧ PC 侧

PC 侧通过如下两类 API 来获得传感器的数据，红色字体是用户的 owner key。

1.获得最新的温湿度信息：

```
curl -X GET -H "Content-Type:application/json" -H "Authorization: token e474
bba4b8e11b97b91019e61b7a018cdbaa3246" http://114.215.177.97/v1/datastreams/
tem_hum/datapoint
```

注：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。

2.获得传感器采集的历史数据：

```
curl -X GET -H "Content-Type:application/json" -H "Authorization: token e474
bba4b8e11b97b91019e61b7a018cdbaa3246" http://114.215.177.97/v1/datastreams/
tem_hum/datapoints
```

4.1.8. 用户自定义反向控制

Espressif 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://114.215.177.97/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://114.215.177.97/v1/device/rpc/?deliver_to_device=
true&action=your_custom_action&any_parameter=any_value"
```

红色为可自定义部分。

在代码中解析 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}
```

需注意，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 action 控制风扇摇头，但服务器不会记录风扇摇了几次头，之前几点钟在摇头，几点钟停止摇头的历史信息。

CONFIDENTIAL