

OPL1000

ULTRA-LOW POWER 2.4GHZ WI-FI + BLUETOOTH SMART SOC

OTA WIFI Demo User Guide



OPULINKS

<http://www.opulinks.com/>

Copyright © 2017-2018, OpuLinks. All Rights Reserved.

OPL1000-Demo-ota-wifi-guide-R01 | Version V01

Date	Version	Contents Updated
2018-08-23	0.1	<ul style="list-style-type: none">Initial Release

TABLE OF CONTENTS

1. 介绍 1

1.1. 文档应用范围 1

1.2. 缩略语 1

1.3. 参考文献 1

2. OTA WIFI 程序实现 2

2.1. 工作原理 2

2.2. API 调用 3

3. 验证 OTA WIFI 功能 5

3.1. 编译 OTA WIFI Example 5

3.2. 生成和下载 OTA 固件 5

3.3. PC 端 HTTP 服务器的执行 10

3.4. OPL1000 http 客户端执行 12

LIST OF FIGURES

Figure 1: AP SSID 和 Password 设置.....2

Figure 2: HTTP server IP 和 Port 设置2

Figure 3: OPL1000 http client 示例网络连接图.....3

Figure 4: HTTP Server 执行界面3

Figure 5: 将 OTA WIFI M3 文件和 M0 文件合并为 OPL1000.bin.....7

Figure 6: 产生 OTA Image 文件9

Figure 7: 下载 opl1000_ota.bin 文件9

Figure 8: OTA Image 执行的输出 log 信息 10

Figure 9: PC HTTP server 服务 12

Figure 10: OPL1000 成功连接 WIFI AP 12

Figure 11: OPL1000 成功连接 http server 13

Figure 12: OPL1000 成功下载 OTA image 15

Figure 13: 升级后的 OTA image 执行输出信息 15

LIST OF TABLES

Table 1: 调用 API 说明3

1. 介绍

1.1. 文档应用范围

本文档介绍（1）如何通过使用调用 SDK API 实现 通过 WIFI 空中下载并升级 OPL1000 固件。（2）如何使用提供的 http_server 工具在本机搭建一个简易的 http 服务器，实现空中下载过程。

1.2. 缩略语

Abbr.	Explanation
AP	Wireless Access Point 无线访问接入点
APP	APPLication 应用程序
APS	Application Sub-system 应用子系统，在本文中亦指 M3 MCU
Blewifi	BLE config WIFI 蓝牙配网应用
DevKit	Development Kit 开发工具板
OTA	Over-the-Air Technology 空间下载技术
TCP	Transmission Control Protocol 传输控制协议

1.3. 参考文献

[1] DEVKIT 快速使用指南 OPL1000-DEVKIT-getting-start-guide.pdf

[2] Download 工具使用指南 OPL1000-patch-download-tool-user-guide.pdf

[3] SDK 应用程序开发指南 OPL1000-SDK-Development-guide.pdf

2. OTA WIFI 程序实现

2.1. 工作原理

OTA wifi 示例程序目录为 SDK\APS_PATCH\examples\system\ota_wifi

它的工作过程为：

- 1 启动 WIFI 任务，将 OPL1000 配置为 Station 模式。
- 2 扫描可用的 AP。
- 3 如果指定连接的 AP SSID 在扫描到的 AP 列表中，则尝试去连接。
- 4 AP 连接成功后，OPL1000 作为 http 客户端和 http 服务器建立联系。http 服务器的地址和端口通过宏定义 HTP_SERVER_IP 和 HTP_SERVER_PORT 定义。
- 5 和 http 服务器建立联系后，使用 http get 命令从指定的 http 服务器获取 OTA Image 文件。
- 6 调用 ota 相关 API 将获取的 OTA Image 数据按照特定格式存放在 Flash 中。

指定连接的 AP SSID 和 Password 在 http_ota_example.h 文件中定义。如 Figure 1 所示：

Figure 1: AP SSID 和 Password 设置

```
#define WIFI_SSID "Opulinks-TEST-AP"  
#define WIFI_PASSWORD "1234abcd"
```

指定连接的 HTTP Server IP 和端口号在 http_ota_example.c 文件中定义。如 Figure 2 所示：

Figure 2: HTTP server IP 和 Port 设置

```
#define HTP_SERVER_IP "192.168.1.102"  
#define HTP_SERVER_PORT "8000"
```

用户可以将使用的 AP 名称和 Password 设置为 http_ota_example.h 定义的 AP SSID 和 Password。这样就不用修改 http_ota_example.h 头文件。

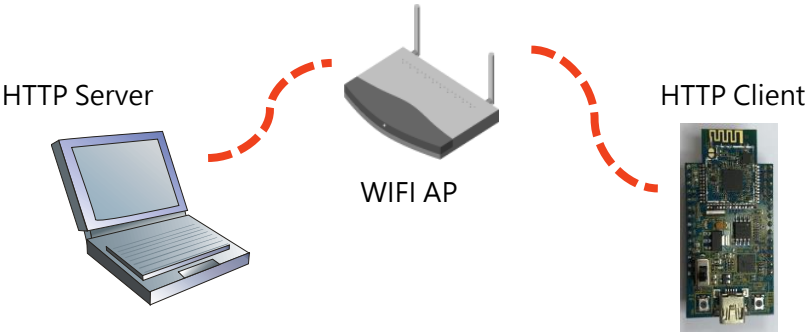
在本演示工程中，使用 PC 作为 Http Server。PC 的 IP 是在 PC 连接 AP 之后分配的动态 IP 地址。可以用 ipconfig 命令查找 PC 的 IP 地址，填充到 Figure 2 的 HTP_SERVER_IP 宏定义中。

HTTP server 和 OPL1000 建立数据传输的网络拓扑如 Figure 3 所示。

OPL1000 以 Station 的角色连接到 WIFI AP，PC 端也作为 Station 连接到 WIFI AP，这样 PC 和 OPL1000 接入到同一个 AP，处于一个局域网网段。PC 上运行 http_server.exe 程序，即充当了 http 服务器角色。注意 http_server.exe 程序默认去连接 Opulinks-TEST-AP 无线接入点。因此最好将试验用的 AP 名称和 Password 按照 Figure 1 来定义。

http_server.exe 程序存放在 Demo\OTA_WIFI 目录下

Figure 3: OPL1000 http client 示例网络连接图



PC 端首先执行 http_server.exe 程序（路径 Demo\ OTA_WIFI），执行界面如 Figure 4 所示。会弹出信息给出 WIFI AP 分配给 PC 的 IP 地址。用户将此 IP 地址填入到 HTTP_SERVER_IP 宏定义中，然后编译，生成 OTA image 固件下载到 OPL1000 中。OPL1000 连接 WIFI AP，成功后连接 HTTP Server，从 HTTP server 上取升级用的 OTA Image 固件，然后保存在 Flash 中。下次上电的时候新下载的固件将被载入运行。

Figure 4: HTTP Server 执行界面

```
Try to connect AP: Opulinks-TEST-AP
PC ping AP @ 192.168.1.1
Ping 192.168.1.1 successfully, Server IP = 192.168.1.102
-----
- HTTP Server Running @ 192.168.1.102 -
-----
serving at port 8000
```

2.2. API 调用

OTA wifi 例程中使用到的 API 说明如表 Table 1 所示。

Table 1: 调用 API 说明

API 接口	API 说明
wifi_register_event_handler	注册内部 WIFI 事件句柄
wifi_event_loop_init	初始化事件处理回调函数，本例中定义为 wifi_event_handler_cb
wifi_init	初始化 WIFI 任务所用堆栈以及 wifi 初始化完成事件句柄
wifi_set_config	设置 OPL1000 WIFI 工作模式
wifi_start	启动 WIFI 任务
osThreadCreate	创建用户应用进程，进程入口为 user_wifi_app_entry
wifi_event_handler_cb	定义收到 WIFI 相关事件消息 ID 时对应的处理操作
wifi_do_scan	扫描可用的无线接入点
wifi_connection	如果指定的 AP 在扫描到的 AP 列表中，则连接它
lwip_net_start	启动 lwip 网路协议栈
lwip_network_init	Tcpip 协议栈和网络接口初始化操作
lwip_net_ready	等待连接并从 AP 获取动态分配 IP 地址
ota_download_by_http	通过 http 下载 OTA 固件
httpclient_connect	http 客户端连接 http 服务器,被 ota_download_by_http 调用
ota_http_retrieve_get	OPL1000 使用 http get 请求从服务器取 OTA 固件数据，被 ota_download_by_http 调用
httpclient_close	关闭 http 客户端连接
httpclient_send_request	http 客户端发出 get 请求,从服务器取特定数据。被 ota_http_retrieve_get 调用。
ota_http_retrieve_offset	将数据保存在缓存区指定偏移位置，被 ota_http_retrieve_get 调用
ota_data_write	将缓存区数据写入到 Flash 中，被 ota_http_retrieve_get 调用

3. 验证 OTA WIFI 功能

3.1. 编译 OTA WIFI Example

编译 ota wifi 示例工程包括三个步骤：

Step1: 首先确定用户需要连接的 WIFI AP，将其 SSID 和访问密码填写到 http_ota_example.h 文件的宏定义 WIFI_SSID，WIFI_PASSWORD 中。如果 AP 是开放接入没有密码，则将 WIFI_PASSWORD 定义为空字符。

```
#define WIFI_SSID      "Opulinks-TEST-AP"  
#define WIFI_PASSWORD "1234abcd"
```

Step2: 执行 http_server.exe 程序将 PC 连接 WIFI AP，并启动 http server 服务。在 Figure 4 中的执行界面可以知道 http server 的 IP 为 192.168.1.102。

```
#define HTTP_SERVER_IP "192.168.1.102"  
#define HTTP_SERVER_PORT "8000"
```

Step3: 使用 Keil C 编译 ota wifi 工程。工程文件路径：

SDK\APS_PATCH\examples\system\ota_wifi\opl1000_app_m3.uvprojx

Keil C 工具设置以及编译过程可以参考文献[3] [SDK 应用程序开发指南](#)。

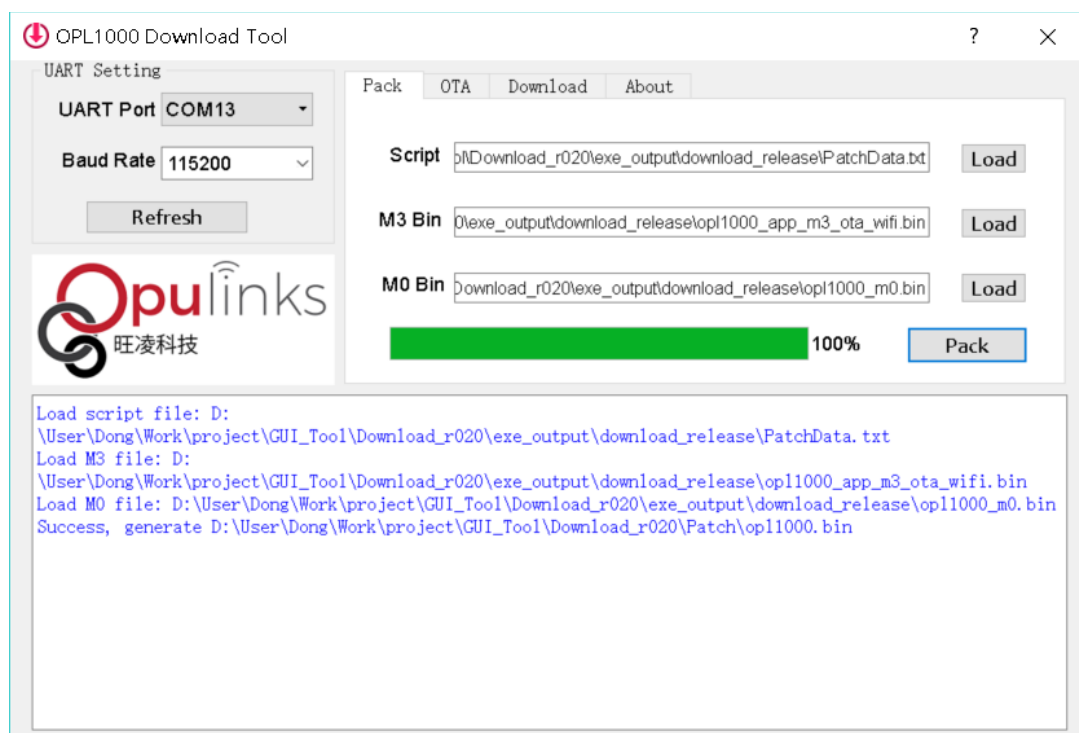
3.2. 生成和下载 OTA 固件

编译成功后在 SDK\APS_PATCH\examples\system\ota_wifi\Output\Objects 目录下产生 opl1000_app_m3.bin 文件。为使 opl1000_app_m3.bin 转换为支持 OTA 功能的固件，需要经过三个步骤：

步骤 1：将 opl1000_app_m3.bin 拷贝到 FW_binary 目录下，然后使用 download 工具将其和 m0 bin 文件合并，产生 opl1000.bin 文件。如图

Figure 5 所示。

Figure 5: 将 OTA WIFI M3 文件和 M0 文件合并为 OPL1000.bin



步骤 2：步骤 1 生成的 opl1000.bin 文件是不带 OTA loader 的固件。我们需要根据它产生 Header 信息，然后和 OTA loader 合并，产生 OTA Image 文件。如

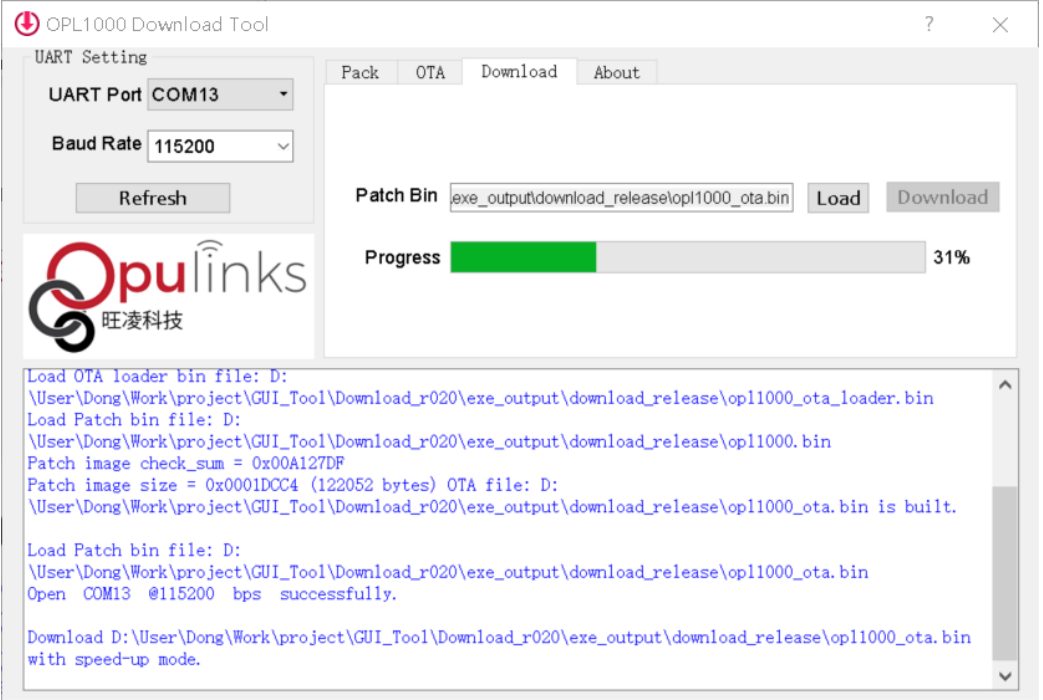
Figure 6 所示，在 OTA tab 页面载入 OTA loader 文件（FW_binary 目录下）和刚才产生的 opl1000.bin 文件。点击 Build OTA Image 按钮，产生 opl1000_ota.bin 文件。

Figure 6: 产生 OTA Image 文件



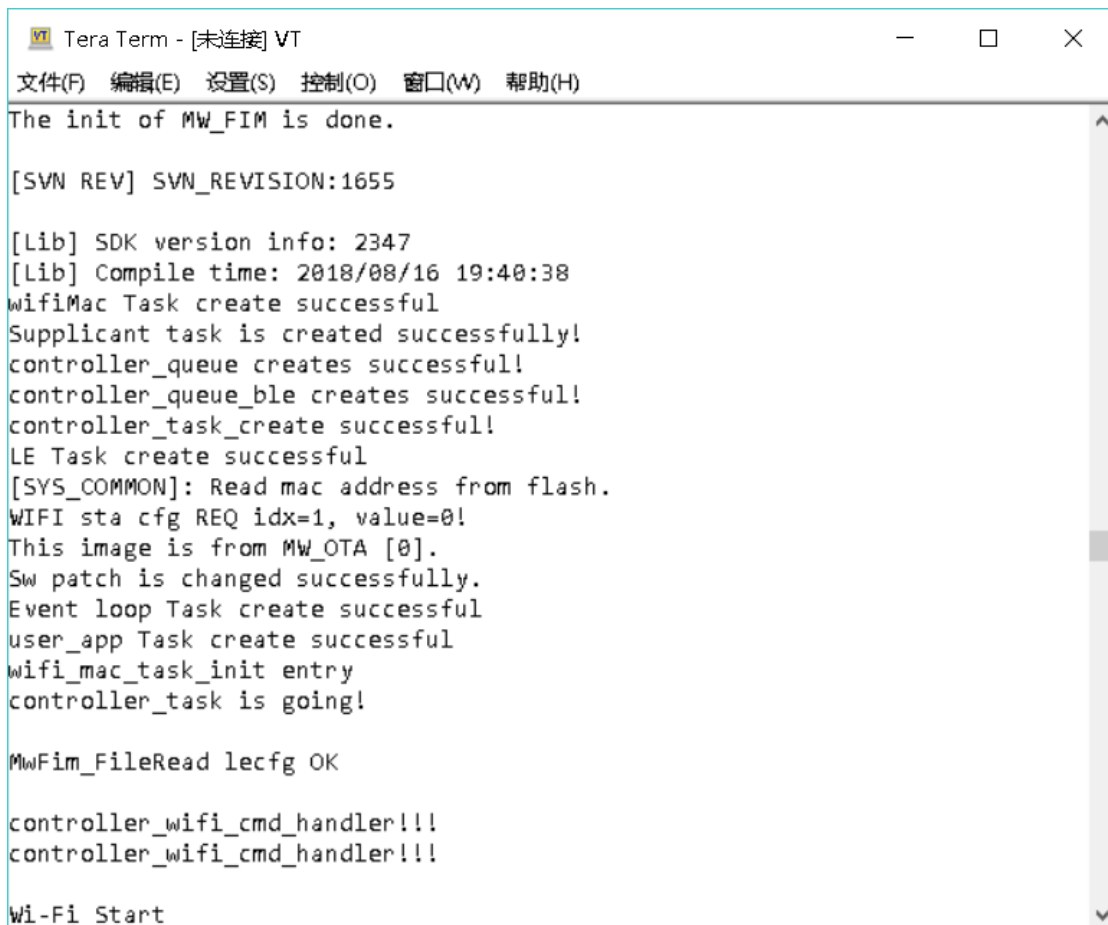
步骤 3：在 download tab 页面载入步骤 2 生成的 opl1000_ota.bin 文件。注意点击 download 按钮后 5 秒钟内需要按 DEVKIT 板上的 reset 按钮复位 OPL1000。如 Figure 7 所示。

Figure 7: 下载 opl1000_ota.bin 文件



下载完成后在 Debug UART 上可以看到 “This image is from MW_OTA[0]” 表示 OTA 固件生成和下载成功。如 Figure 8 所示。

Figure 8: OTA Image 执行的输出 log 信息



```
Tera Term - [未连接] VT
文件(F) 编辑(E) 设置(S) 控制(O) 窗口(W) 帮助(H)
The init of MW_FIM is done.

[SVN REV] SVN_REVISION:1655

[Lib] SDK version info: 2347
[Lib] Compile time: 2018/08/16 19:40:38
wifiMac Task create successful
Supplicant task is created successfully!
controller_queue creates successful!
controller_queue_ble creates successful!
controller_task_create successful!
LE Task create successful
[SYS_COMMON]: Read mac address from flash.
WIFI sta cfg REQ idx=1, value=0!
This image is from MW_OTA [0].
Sw patch is changed successfully.
Event loop Task create successful
user_app Task create successful
wifi_mac_task_init entry
controller_task is going!

MwFim_FileRead lecfg OK

controller_wifi_cmd_handler!!!
controller_wifi_cmd_handler!!!

Wi-Fi Start
```

下载工具的使用可参考文献[2] [Download 工具使用指南](#)。DEVKIT 板的使用可参考文献[1] [DEVKIT 快速使用指南](#)。

3.3. PC 端 HTTP 服务器的执行

PC 端启动 http server 服务后，等待 OPL1000 的 Get 请求。收到请求后给予响应，在 Server 端会打印 Client 端的反馈信息。如

Figure 9 所示

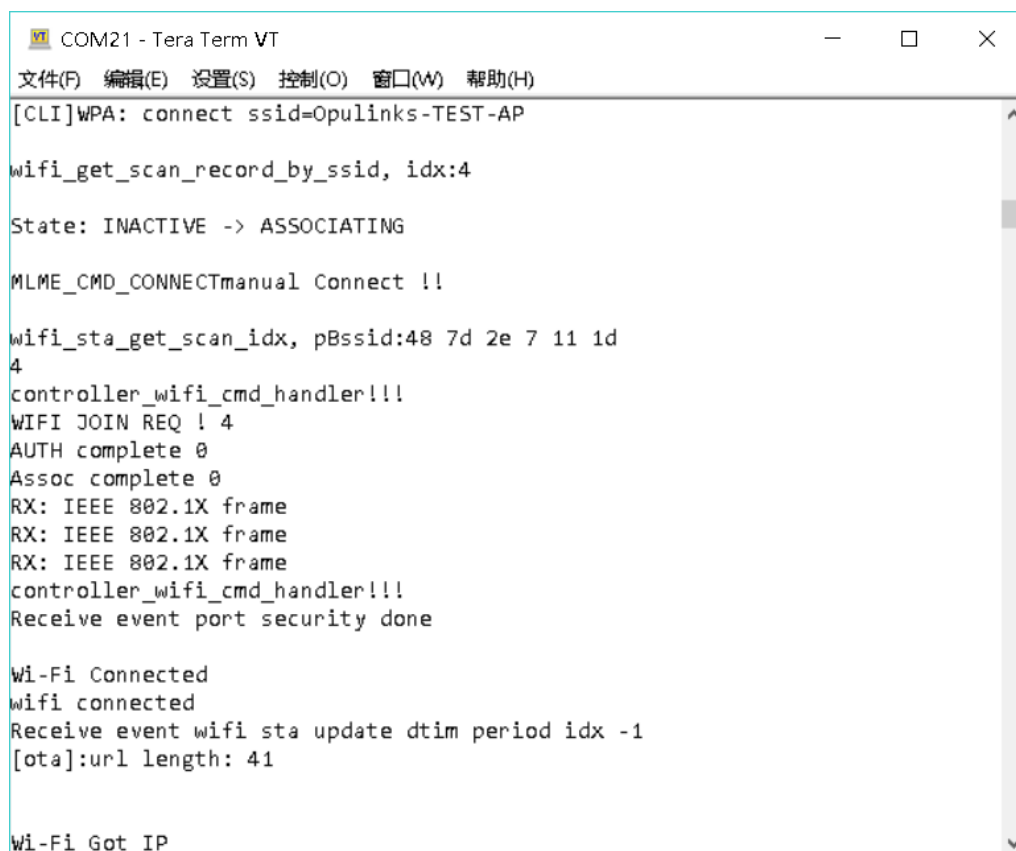
Figure 9: PC HTTP server 服务

```
Try to connect AP: Opulinks-TEST-AP
PC ping AP @ 192.168.1.1
Ping 192.168.1.1 successfully, Server IP = 192.168.1.102
-----
- HTTP Server Running @ 192.168.1.102 -
-----
serving at port 8000
192.168.1.103 - - [23/Aug/2018 14:02:59] "GET /opl1000_ota.bin HTTP/1.1" 200 -
```

3.4. OPL1000 http 客户端执行

OPL1000 http client 端的执行情况可以从 Debug UART 打印信息观察到。图 Figure 10 显示 OPL1000 成功连接 WIFI AP。

Figure 10: OPL1000 成功连接 WIFI AP



```
COM21 - Tera Term VT
文件(F) 编辑(E) 设置(S) 控制(O) 窗口(W) 帮助(H)
[CLI]WPA: connect ssid=Opulinks-TEST-AP

wifi_get_scan_record_by_ssid, idx:4

State: INACTIVE -> ASSOCIATING

MLME_CMD_CONNECTmanual Connect !!

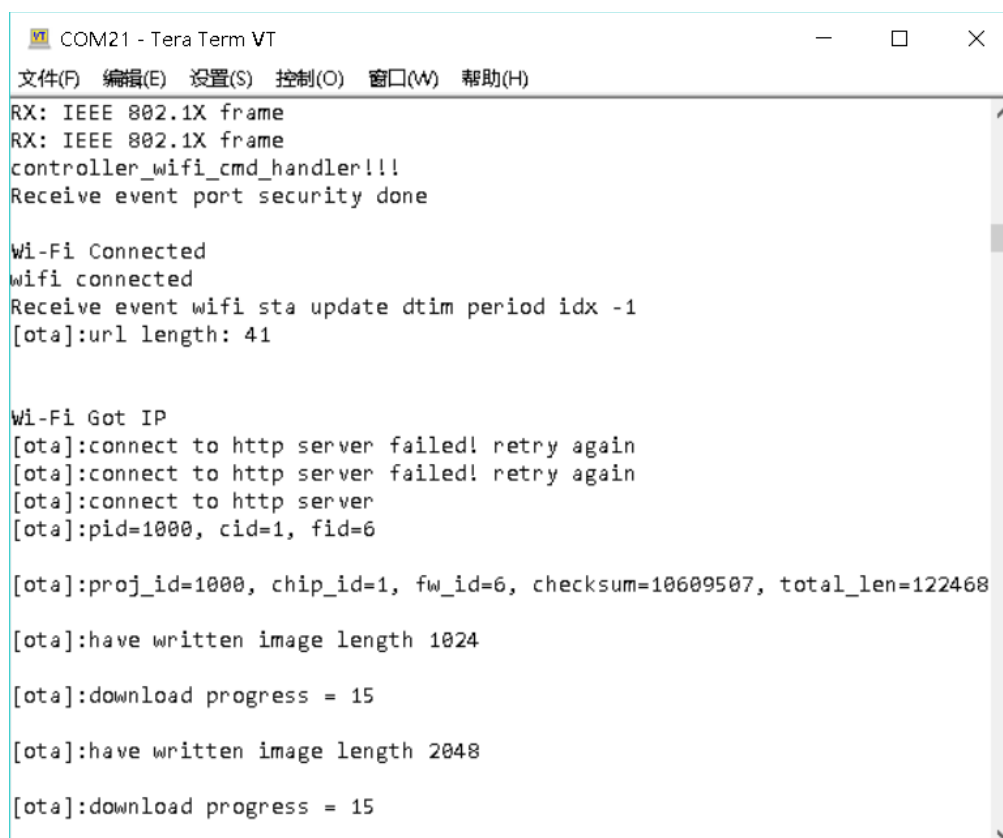
wifi_sta_get_scan_idx, pBssid:48 7d 2e 7 11 1d
4
controller_wifi_cmd_handler!!!
WIFI JOIN REQ ! 4
AUTH complete 0
Assoc complete 0
RX: IEEE 802.1X frame
RX: IEEE 802.1X frame
RX: IEEE 802.1X frame
controller_wifi_cmd_handler!!!
Receive event port security done

Wi-Fi Connected
wifi connected
Receive event wifi sta update dtim period idx -1
[ota]:url length: 41

Wi-Fi Got IP
```

OPL1000 连接 AP 后，接着尝试连接 Http server。Figure 11 显示 OPL1000 成功连接 http server, 并获取到了存放在 Server 上的 OTA image 头信息。

Figure 11: OPL1000 成功连接 http server



```
COM21 - Tera Term VT
文件(F) 编辑(E) 设置(S) 控制(O) 窗口(W) 帮助(H)
RX: IEEE 802.1X frame
RX: IEEE 802.1X frame
controller_wifi_cmd_handler!!!
Receive event port security done

Wi-Fi Connected
wifi connected
Receive event wifi sta update dtim period idx -1
[ota]:url length: 41

Wi-Fi Got IP
[ota]:connect to http server failed! retry again
[ota]:connect to http server failed! retry again
[ota]:connect to http server
[ota]:pid=1000, cid=1, fid=6

[ota]:proj_id=1000, chip_id=1, fw_id=6, checksum=10609507, total_len=122468

[ota]:have written image length 1024

[ota]:download progress = 15

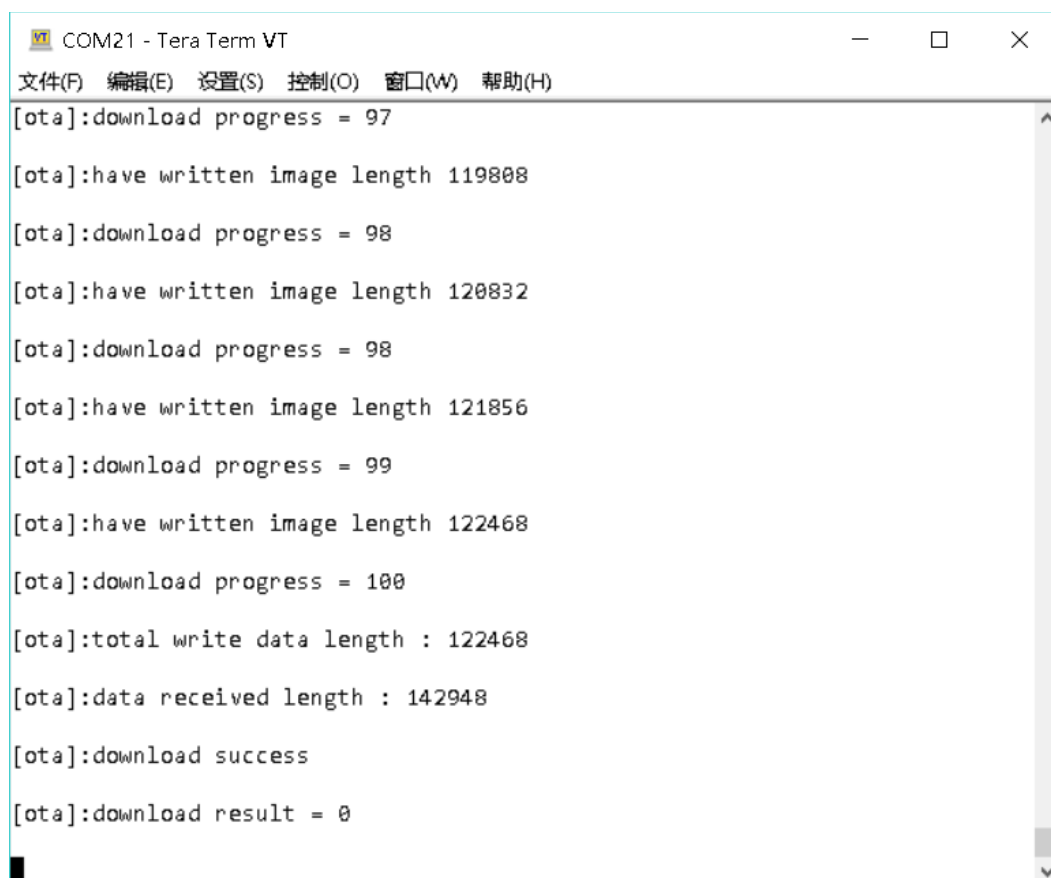
[ota]:have written image length 2048

[ota]:download progress = 15
```

接下来 Opl1000 分段从 http server 获取 OTA image 文件。如果整个过程正常，在最终将打印输出 download success 信息。

Figure 12 显示 OTA image 被成功下载。

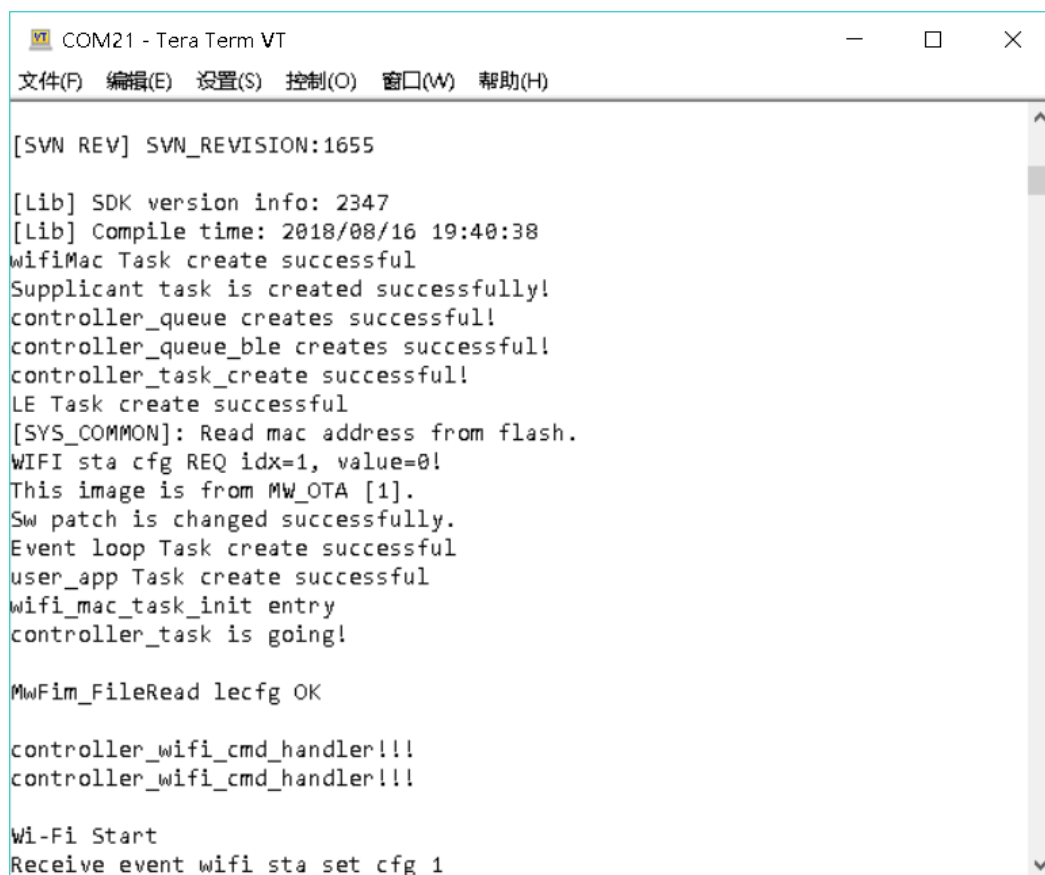
Figure 12: OPL1000 成功下载 OTA image



```
COM21 - Tera Term VT
文件(F) 编辑(E) 设置(S) 控制(O) 窗口(W) 帮助(H)
[ota]:download progress = 97
[ota]:have written image length 119808
[ota]:download progress = 98
[ota]:have written image length 120832
[ota]:download progress = 98
[ota]:have written image length 121856
[ota]:download progress = 99
[ota]:have written image length 122468
[ota]:download progress = 100
[ota]:total write data length : 122468
[ota]:data received length : 142948
[ota]:download success
[ota]:download result = 0
```

重新启动 OPL1000, 下载保存的 OTA Image 文件将被载入并执行。 “Figure 8: OTA Image 执行的输出 log 信息” 中原有 OTA 固件保存在 MW_OTA[0]区域，新的固件保存在 MW_OTA[1]区域。在 Figure 13 中可以观察到 This image is from MW_OTA[1] 信息。这表明新下载的固件被执行。

Figure 13: 升级后的 OTA image 执行输出信息



```
COM21 - Tera Term VT
文件(F) 编辑(E) 设置(S) 控制(O) 窗口(W) 帮助(H)

[SVN REV] SVN_REVISION:1655

[Lib] SDK version info: 2347
[Lib] Compile time: 2018/08/16 19:40:38
wifiMac Task create successful
Supplicant task is created successfully!
controller_queue creates successful!
controller_queue_ble creates successful!
controller_task_create successful!
LE Task create successful
[SYS_COMMON]: Read mac address from flash.
WIFI sta cfg REQ idx=1, value=0!
This image is from MW_OTA [1].
Sw patch is changed successfully.
Event loop Task create successful
user_app Task create successful
wifi_mac_task_init entry
controller_task is going!

MwFim_FileRead lecfig OK

controller_wifi_cmd_handler!!!
controller_wifi_cmd_handler!!!

Wi-Fi Start
Receive event wifi sta set cfg 1
```

CONTACT

sales@Opulinks.com