

OPL1000

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

TCP 客户端基于蓝牙配网说明文档



OPULINKS

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

Copyright © 2017-2020, OpuLinks. All Rights Reserved.

OPL1000-TCP 客户端基于蓝牙配网说明文档 | Version V1.0

版本纪录

日期	版本	更新内容
2018-05-31	0.1	<ul style="list-style-type: none">初版
2018-07-27	0.2	<ul style="list-style-type: none">新增处理: TCP 服务器向客户端响应 ACK 信息
2018-12-14	0.3	<ul style="list-style-type: none">新增 skip DTIM, timeout, high power, low power 定义
2019-07-19	0.4	<ul style="list-style-type: none">更新 DTIM 说明与示例同步
2020-01-12	0.5	<ul style="list-style-type: none">更新示例。之前的版本，AP SSID 和密码需在在项目中直接定义；在此版本，由于 BLE 配置 WIFI 功能已加入，DUT 可经由 BLE 连接取得 AP 信息加入新章节来介绍如何 BLE 配置 WIFI
2020-01-14	1.0	<ul style="list-style-type: none">将章节 2、3 修改为基于 BLEWIFI 的内容

目录

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. TCP_Client_BLEWIFI 示例工作原理	2
2.1. 工程组织	2
2.2. 工作原理和过程	3
2.3. 宏定义参数说明	4
2.4. API 调用	6
2.4.1. 初始化和任务处理句柄	6
2.4.2. 蓝牙配网	9
2.4.3. TCP 建立和数据传输	10
3. 使用 TCP Client BLEWIFI 功能	11
3.1. 编译 TCP_client Example	11
3.2. 下载固件	13
3.3. tcp client blewifi 功能验证	14
3.3.1. 开启蓝牙广播并完成蓝牙配网	15
3.3.2. PC 端执行网络调试助手协助验证	16
3.3.3. 睡眠模式设置和验证	17

图目录

Figure 1: 工程文件 2

Figure 2: TCP_CLIENT_BLEWIFI 示例网络连接图 3

Figure 3: 网络调试助手中 TCP Server 参数配置 4

Figure 4: 需要更新的宏定义 5

Figure 5: 定义 TCP server 和 port 6

Figure 6: 定义 UART0 的 PIN 6

Figure 7: BLE 消息处理函数映射表 7

Figure 8: WIFI 消息处理函数映射表 7

Figure 9: BLEWIFI controller 消息处理函数映射表 8

Figure 10: UART0 消息处理函数映射表 8

Figure 11: IOT 数据发送相关的消息处理函数映射表 8

Figure 12: 蓝牙配网相关参考文档 9

Figure 13 : TCP Server IP 的获取和更新 11

Figure 14: Keil C 编译工程 12

Figure 15: gcc 编译工程 13

Figure 16: PACK OPL1000 固件 13

Figure 17: 下载固件 14

Figure 18: 完成蓝牙配网 15

Figure 19: OPL1000 和 TCP Server 间 RX path 通信示例 16

Figure 20: OPL1000 和 TCP Server 间 TX path 通信示例 16

Figure 21: 切换睡眠模式 17

表目录

Table 1: 项目文件夹和内容 3

Table 2: PIN 参数说明 5

Table 3: 蓝牙配网相关的 API 9

Table 4: 调用 TCP API 说明 10

1. 介绍

1.1. 文档应用范围

本文档介绍 TCP-Client Blewifi 示例工程的工作原理和使用方法。该工程中完成如下功能：使用蓝牙配网使 OPL1000 设备和指定 AP 连接，然后将 OPL1000 配置为 TCP 客户端（Client），其后 OPL1000 设备就可以与同一网段的 TCP 服务器（Server）进行连接和数据传输。

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
DTIM	Delivery Traffic Indication Message 传输指示消息
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

访问链接：<https://github.com/Opulinks-Tech/OPL1000A2-SDK/tree/master/Doc/OPL1000A2-patch-download-tool-user-guide.pdf>

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

访问连接：<https://github.com/Opulinks-Tech/OPL1000A2-SDK/blob/master/Doc/OPL1000-SDK-Development-guide.pdf>

2. TCP_CLIENT_BLEWIFI 示例工作原理

2.1. 工程组织

本示例程序所在目录为 SDK\ APS_PATCH\examples\system\tcp_client_blewifi，主要包含蓝牙配网，数据传输，TCP 数据收发，串口处理和工程文件等。

Figure 1: 工程文件

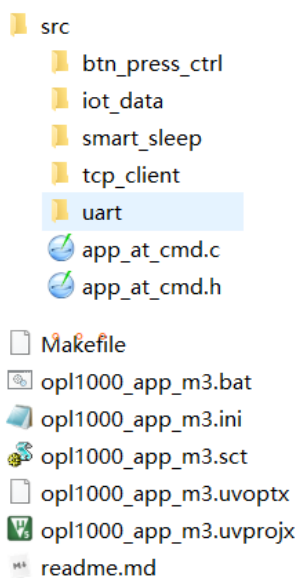


Table 1: 项目文件夹和内容

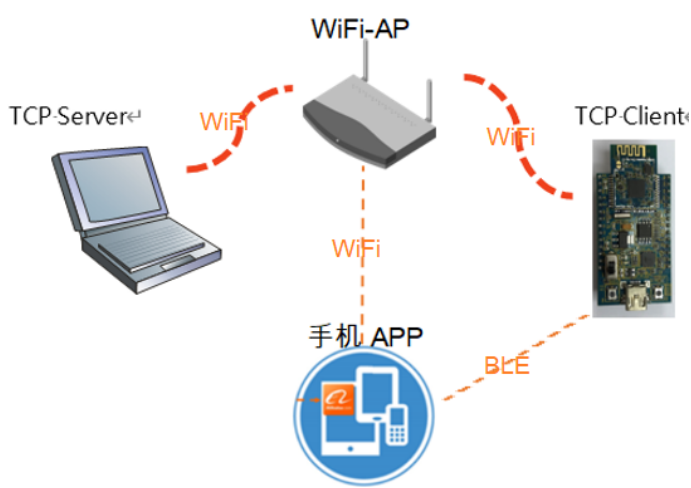
文件夹和文件	内容说明
src	存放蓝牙配网，各功能模块相关.c 和.h 头文件，以及 main 文件
src/iot_data	存放收发数据相关的代码
src/smart_sleep	存放跟睡眠模式相关的代码
src/tcp_client	存放跟收发 TCP 数据相关的代码
src/uart	存放跟 UART0 收发数据相关的代码
Makefile	用于以 gcc 方式编译的 makefile 文件
opl1000_app_m3.bat opl1000_app_m3.ini opl1000_app_m3.sct opl1000_app_m3.uvoptx opl1000_app_m3.uvprojx	编译工程文件。

2.2. 工作原理和过程

TCP server 和 OPL1000 建立 TCP 数据传输的网络拓扑如下图所示。

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。

Figure 2: TCP_CLIENT_BLEWIFI 示例网络连接图

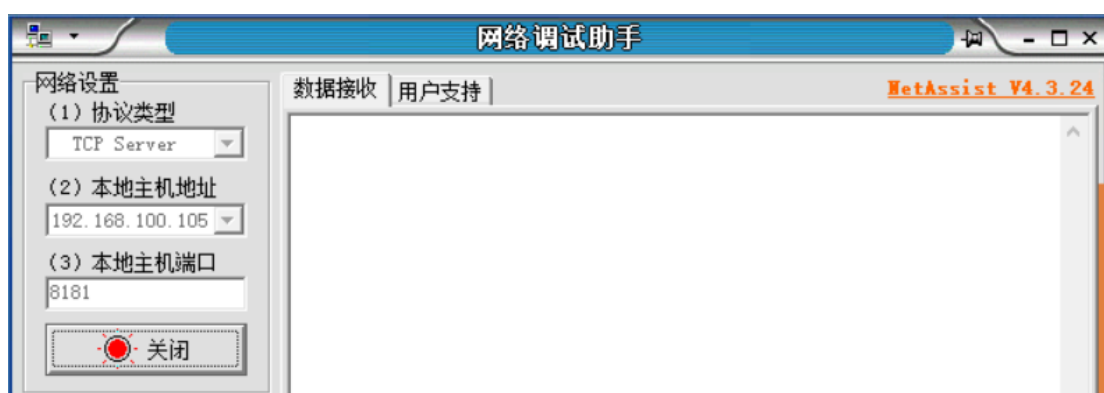


PC 端执行网络调试助手程序 NetAssist.exe。在网络设置对话框，选择 TCP Server 作为协议类型，填入 WIFI AP 分配给 PC 的 IP 地址，在本例中为 192.168.100.105。本地端口号可以任意取一个数，但最好不要使用已知、通用的端口号，例如 8080。在本例中端口号定义为 8181。

TCP server 的 IP 地址和端口参数在 在本例程中赋值给宏定义 TCP_SERVER_ADDR 和 TCP_PORT。

【注意：关于 TCP server 的 IP 地址的获取，请参考本文的第 3.1 节】

Figure 3: 网络调试助手中 TCP Server 参数配置



它的工作过程为：

- 1 OPL1000 模块上电后，拉低 IO5 管脚并保持 5 秒钟以上时间，以开启蓝牙广播；
- 2 在手机上打开 OPL1000 APP 进行蓝牙配网，选择合适的 WIFI AP，输入密码后连接 AP 成功；
- 3 OPL1000 模块获得 IP 后，会自动尝试连接 TCP server（由 Netassist.exe 实现）；
- 4 在成功和 TCP Server 在监听的端口号建立连接后，用户可以开始 TX 或 RX 这两个 path 的验证；
- 5 对于 RX path，用户可以在 Netassist 上发送一个字符串（比如，200 字节），OPL1000 模块在收到这串数据后会把它发送到串口 UART0【关于 UART0 的定义，请参考本文的 2.3 节】；
- 6 对于 TX path，用户在打开串口 UART0 后，在串口上输入一些测试数据（注意：在此输入的字符不会回显），OPL1000 模块在收到这串数据后发送数据到 TCP server 端，也就是说，在 Netassist 上会接收到这些数据。

2.3. 宏定义参数说明

本示例程序需要进行设定的宏定义主要存放在 src/blewifi_configuration.h 文件中，其中宏定义 MW_FIM_VER11_PROJECT 是一个必须要更新的宏，只需更新为跟原来的值不同即可。

Figure 4: 需要更新的宏定义

```

/*
FIM version
*/
#define MW_FIM_VER11_PROJECT          0x07    // 0x00 ~ 0xFF

```

而跟本示例程序相关的引脚定义如下：

Table 2: PIN 参数说明

宏定义	内容说明	默认值
BUTTON_IO_PORT	用于定义触发蓝牙配网的 IO 引脚，低电平有效且需保持 5S 及以上	GPIO_IDX_05
WAKEUP_IO_PORT	用于定义切换睡眠模式的 IO 引脚。高电平时进入睡眠模式；低电平时则唤醒睡眠模式	GPIO_IDX_17
NOTIFY_IO_PORT	定义用于通知主控 MCU 当前 OPL1000 的睡眠模式的 IO 引脚。主控 MCU 可根据该引脚的电平判断 OPL1000 是否在睡眠模式。高电平时表明 OPL1000 处于睡眠模式	GPIO_IDX_22
TCP_IO_PORT_0	用于定义指示当前 TCP 连接状态的引脚。高电平时表示连接正常；低电平时表示连接断开；	GPIO_IDX_06
TCP_IO_PORT_1	用于定义指示当前 TCP 连接状态的引脚。低电平时表示连接正常；高电平时表示连接断开；	GPIO_IDX_07

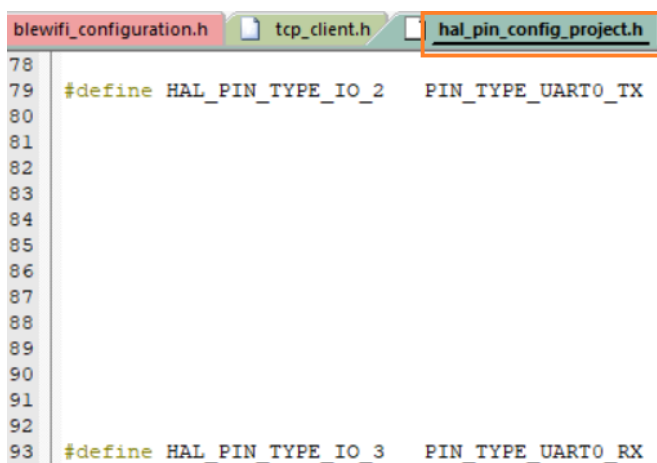
而指定连接的 TCP Server，端口号在 tcp_client.h 文件中定义。如下所示：

Figure 5: 定义 TCP server 和 port

```
#define TCP_SERVER_ADDR    "192.168.100.105"  
#define TCP_PORT          8181
```

在本示例程序中，OPL1000 为主控 MCU 提供了 UART0 做为收发透传数据的接口。在 hal_pin_config_project.h 文件中定义 IO2, IO3 用作 UART0 的 TX, RX 如下：

Figure 6: 定义 UART0 的 PIN



```
blewifi_configuration.h  tcp_client.h  hal_pin_config_project.h  
78  
79 #define HAL_PIN_TYPE_IO_2    PIN_TYPE_UART0_TX  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93 #define HAL_PIN_TYPE_IO_3    PIN_TYPE_UART0_RX
```

2.4. API 调用

2.4.1. 初始化和任务处理句柄

和所有基于 BLEWIFI 的工程一样，本示例程序的初始化入口函数为 src/main_patch.c 文件中的 BleWifiAppInit()函数。

该例程主要启用了五个任务处理句柄如下

1. BLE Handler

BLE Handler 功能是等待手机端蓝牙与 OPL1000 的连接，此时 OPL1000 会持续发送 BLE 广播，直到蓝牙建立连接，在 blewifi_data.c 文件中定义了一组消息处理函数如下：

Figure 7: BLE 消息处理函数映射表

```
static T_BleWifi_Ble_ProtocolHandlerTbl g_tBleProtocolHandlerTbl[] =
{
    {BLEWIFI_REQ_SCAN,          BleWifi_Ble_ProtocolHandler_Scan},
    {BLEWIFI_REQ_CONNECT,       BleWifi_Ble_ProtocolHandler_Connect},
    {BLEWIFI_REQ_DISCONNECT,    BleWifi_Ble_ProtocolHandler_Disconnect},
    {BLEWIFI_REQ_RECONNECT,     BleWifi_Ble_ProtocolHandler_Reconnect},
    {BLEWIFI_REQ_READ_DEVICE_INFO, BleWifi_Ble_ProtocolHandler_ReadDeviceInfo},
    {BLEWIFI_REQ_WRITE_DEVICE_INFO, BleWifi_Ble_ProtocolHandler_WriteDeviceInfo},
    {BLEWIFI_REQ_WIFI_STATUS,   BleWifi_Ble_ProtocolHandler_WifiStatus},
    {BLEWIFI_REQ_RESET,         BleWifi_Ble_ProtocolHandler_Reset},

    #if (BLE_OTA_FUNCTION_EN == 1)
    {BLEWIFI_REQ_OTA_VERSION,    BleWifi_Ble_ProtocolHandler_OtaVersion},
    {BLEWIFI_REQ_OTA_UPGRADE,    BleWifi_Ble_ProtocolHandler_OtaUpgrade},
    }
```

2. WIFI Handler

WIFI Handler 是 OPL1000 与 AP 建立连接后，连线及断线检查，断线后重连功能。在 blewifi_wifi_api.c 文件中定义了一组消息处理函数如下，每条记录对应一个消息和消息处理函数的映射：

Figure 8: WIFI 消息处理函数映射表

```
static T_BleWifi_Wifi_EventHandlerTbl g_tWifiEventHandlerTbl[] =
{
    {WIFI_EVENT_STA_START,      BleWifi_Wifi_EventHandler_Start},
    {WIFI_EVENT_STA_CONNECTED,  BleWifi_Wifi_EventHandler_Connected},
    {WIFI_EVENT_STA_DISCONNECTED, BleWifi_Wifi_EventHandler_Disconnected},
    {WIFI_EVENT_SCAN_COMPLETE,  BleWifi_Wifi_EventHandler_ScanComplete},
    {WIFI_EVENT_STA_GOT_IP,     BleWifi_Wifi_EventHandler_GotIp},
    {WIFI_EVENT_STA_CONNECTION_FAILED, BleWifi_Wifi_EventHandler_ConnectionFailed},

    {0xFFFFFFFF,               NULL}
};
```

3. BLEWIFI controller Handler

BLEWIFI controller Handler 是跟 OPL1000 BLEWIFI controller 相关的功能。blewifi_ctrl.c 文件中定义了一组消息处理函数如下，每条记录对应一个消息和消息处理函数的映射：

Figure 9: BLEWIFI controller 消息处理函数映射表

```
static T_BleWifi_Ctrl_EvtHandlerTbl g_tCtrlEvtHandlerTbl[] = 每个消息对应一个消息处理函数
{
    {BLEWIFI_CTRL_MSG_BLE_INIT_COMPLETE,      BleWifi_Ctrl_TaskEvtHandler_BleInitComplete},
    {BLEWIFI_CTRL_MSG_BLE_ADVERTISING_CFM,     BleWifi_Ctrl_TaskEvtHandler_BleAdvertisingCfm},
    {BLEWIFI_CTRL_MSG_BLE_ADVERTISING_EXIT_CFM, BleWifi_Ctrl_TaskEvtHandler_BleAdvertisingExitCfm},
    {BLEWIFI_CTRL_MSG_BLE_ADVERTISING_TIME_CHANGE_CFM, BleWifi_Ctrl_TaskEvtHandler_BleAdvertisingTimeChangeCfm},
    {BLEWIFI_CTRL_MSG_BLE_CONNECTION_COMPLETE, BleWifi_Ctrl_TaskEvtHandler_BleConnectionComplete},
    {BLEWIFI_CTRL_MSG_BLE_CONNECTION_FAIL,     BleWifi_Ctrl_TaskEvtHandler_BleConnectionFail},
    {BLEWIFI_CTRL_MSG_BLE_DISCONNECT,          BleWifi_Ctrl_TaskEvtHandler_BleDisconnect},
    {BLEWIFI_CTRL_MSG_BLE_DATA_IND,            BleWifi_Ctrl_TaskEvtHandler_BleDataInd},

    {BLEWIFI_CTRL_MSG_WIFI_INIT_COMPLETE,      BleWifi_Ctrl_TaskEvtHandler_WifiInitComplete},
    {BLEWIFI_CTRL_MSG_WIFI_SCAN_DONE_IND,      BleWifi_Ctrl_TaskEvtHandler_WifiScanDoneInd},
    {BLEWIFI_CTRL_MSG_WIFI_CONNECTION_IND,      BleWifi_Ctrl_TaskEvtHandler_WifiConnectionInd},
    {BLEWIFI_CTRL_MSG_WIFI_DISCONNECTION_IND,   BleWifi_Ctrl_TaskEvtHandler_WifiDisconnectionInd},
    {BLEWIFI_CTRL_MSG_WIFI_GOT_IP_IND,          BleWifi_Ctrl_TaskEvtHandler_WifiGotIpInd},
    {BLEWIFI_CTRL_MSG_WIFI_AUTO_CONNECT_IND,    BleWifi_Ctrl_TaskEvtHandler_WifiAutoConnectInd},
}
```

4. 与 UART0 消息处理相关的句柄

该句柄在 src/uart/uart.c 文件中实现，用于处理 UART0 上收到的数据并转发。

Figure 10: UART0 消息处理函数映射表

```
static T_IoT_Uart0_EvtHandlerTbl g_tAppIotUart0EvtHandlerTbl[] =
{
    {IOT_UART0_MSG_DATA_IN,      Iot_Uart0_EvtHandler_Data_In},
    {0xFFFFFFFF,                 NULL}
};
```

5. 与 IOT 数据发送相关的句柄

该句柄在 src/iot_data/iot_data.c 文件中实现，主要用于建立 TCP 连接，发送 TCP 数据和 heartbeat。

Figure 11: IOT 数据发送相关的消息处理函数映射表

```
static T_IoT_Data_EvtHandlerTbl g_tAppIotDataTxTaskEvtHandlerTbl[] =
{
    {IOT_DATA_TX_MSG_DATA_POST,      Iot_Data_TxTaskEvtHandler_DataPost},
    {IOT_DATA_TX_MSG_EST_TCP_CONNECTION, Iot_Data_TxTaskEvtHandler_Est_Tcp_Connection},
    {IOT_DATA_TX_MSG_SEND_HEARTBEAT,   Iot_Data_TxTaskEvtHandler_Send_Heartbeat},
    {0xFFFFFFFF,                       NULL}
};
```

2.4.2. 蓝牙配网

本例程中使用到的蓝牙配网相关的 API 说明如下所示

Table 3: 蓝牙配网相关的 API

API 接口	API 说明
BleWifi_Wifi_Init	WiFi 相关的初始化函数
wifi_event_loop_init	初始化事件处理回调函数，本例中定义为 BleWifi_Wifi_EventHandlerCb
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_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_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
BleWifi_Ble_Init	直接调用 BleWifi_Ble_ServerAppInit
BleWifi_Ble_ServerAppInit	创建 BLE task，BLE 相关的初始化函数
BleWifi_Ble_TaskHandler	BLE 相关的消息处理函数
BleWifi_Ctrl_Init	BLEWIFI controller 相关的初始化函数。主要包括创建 task，message queue，timer，Event Group 等。
BleWifi_Ctrl_Task	接收消息并交由消息处理句柄处理
BleWifi_Ctrl_TaskEvtHandler	BLEWIFI controller 相关的消息处理句柄

更多信息，请参考 <https://github.com/Opulinks-Tech/OpulinksTech-WIKI/wiki/Documents>

Figure 12: 蓝牙配网相关参考文档

其他应用说明文档

- [蓝牙配网应用设计说明](#)
- [基于OPL1000的物联网应用框架说明](#)
- [OPL1000系统初始化说明](#)

2.4.3. TCP 建立和数据传输

本例程中使用到的 TCP 相关的 API 说明如下所示

Table 4: 调用 TCP API 说明

API 接口	API 说明
connect_tcp	注册内部 WIFI 事件句柄
disconnect_tcp	初始化事件处理回调函数 · 本例中定义为 wifi_event_handler_cb
write_tcp	初始化 WIFI 任务所用堆栈以及 wifi 初始化完成事件句柄
read_tcp	设置 OPL1000 WIFI 工作模式

3. 使用 TCP CLIENT BLEWIFI 功能

3.1. 编译 TCP_client Example

编译 tcp client blewifi 示例工程包括三个步骤：

Step1: 将 PC 连接 WIFI AP，可以使用 ipconfig 命令查看 WIFI AP 分配给 PC 的 IP 地址。将 PC 的 IP 信息填入到宏定义 TCP_SERVER_ADDR 和 TCP_PORT 中。

```
#define TCP_SERVER_ADDR      "192.168.100.105"
#define TCP_PORT             8181
```

Figure 13：TCP Server IP 的获取和更新

无线局域网适配器 WLAN:

连接特定的 DNS 后缀 :
本地连接 IPv6 地址. : fe80::6dd5:dc8d:d8c:e787%19
IPv4 地址 : 192.168.100.105
子网掩码 : 255.255.255.0
默认网关. : 192.168.100.1

#include "lwip/dns.h"

#define TCP_SERVER_ADDR "192.168.100.105"
#define TCP_PORT 8181

uintptr_t connect_tcp(const char *host, uint16_t port);
int disconnect_tcp(uintptr_t fd);
int write_tcp(uintptr_t fd, const char *buf, uint32_t len, uint32_t timeout_ms);
int read_tcp(uintptr_t fd, char *buf, uint32_t len, uint32_t timeout_ms);

无线局域网适配器 WLAN:

连接特定的 DNS 后缀 :
本地连接 IPv6 地址. : fe80::6dd5:dc8d:d8c:e787%19
IPv4 地址 : 192.168.100.105
子网掩码 : 255.255.255.0
默认网关. : 192.168.100.1

确认这两个是一样的

Step2: 在 blewifi_configuration.h 文件中更新 MW_FIM_VER11_PROJECT 宏

【注：更新为跟原来不一样的值】：

```
#define MW_FIM_VER11_PROJECT      0x07    // 0x00 ~ 0xFF
```

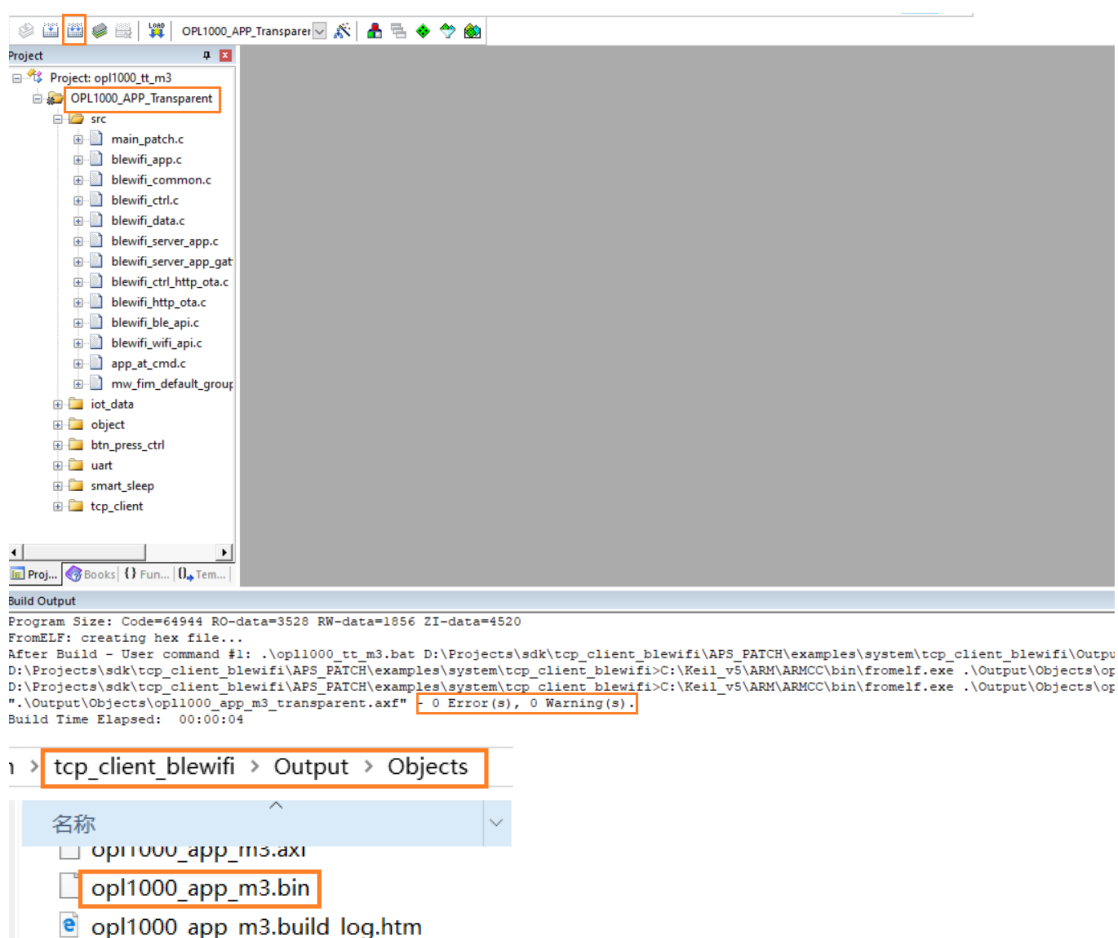
为了方便演示工程代码，把 src\iot_data\iot_data.c 文件里的“#if 0”修改为“#if 1”

```
iot_data.c  
368  
369 #if 1 默认是#if 0  
370     for(int i=0;i<ret;i++){  
371         Hal_Uart_DataSend(UART_IDX_0, szBuf[i]);  
372     }  
373 #endif  
374
```


Step3: tcp client blewifi 示例工程可通过 Keil C (打开 opl1000_tt_m3.uvprojx 工程文件) 和 Gcc (makefile 文件) 两种方式编译 , 任取其中一种方式即可。使用 Keil C 编译 tcp_client_blewifi 工程。工程文件路径 :

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

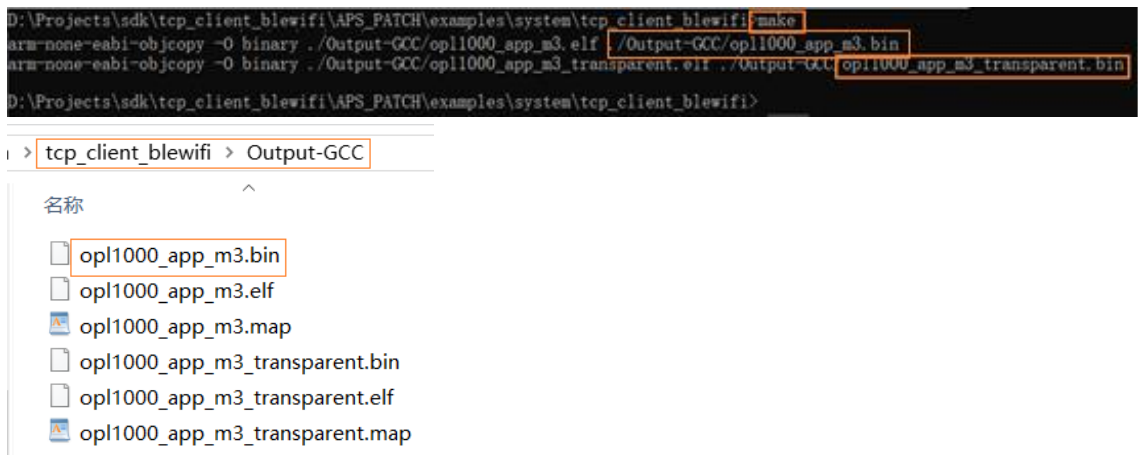
Figure 14: Keil C 编译工程



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

而通过 gcc 编译的结果如下：

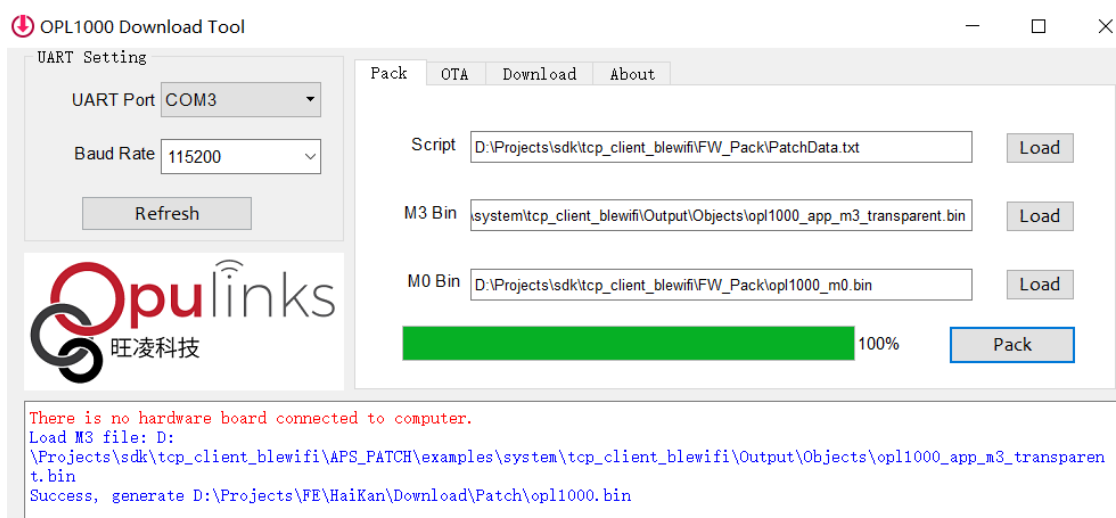
Figure 15: gcc 编译工程



3.2. 下载固件

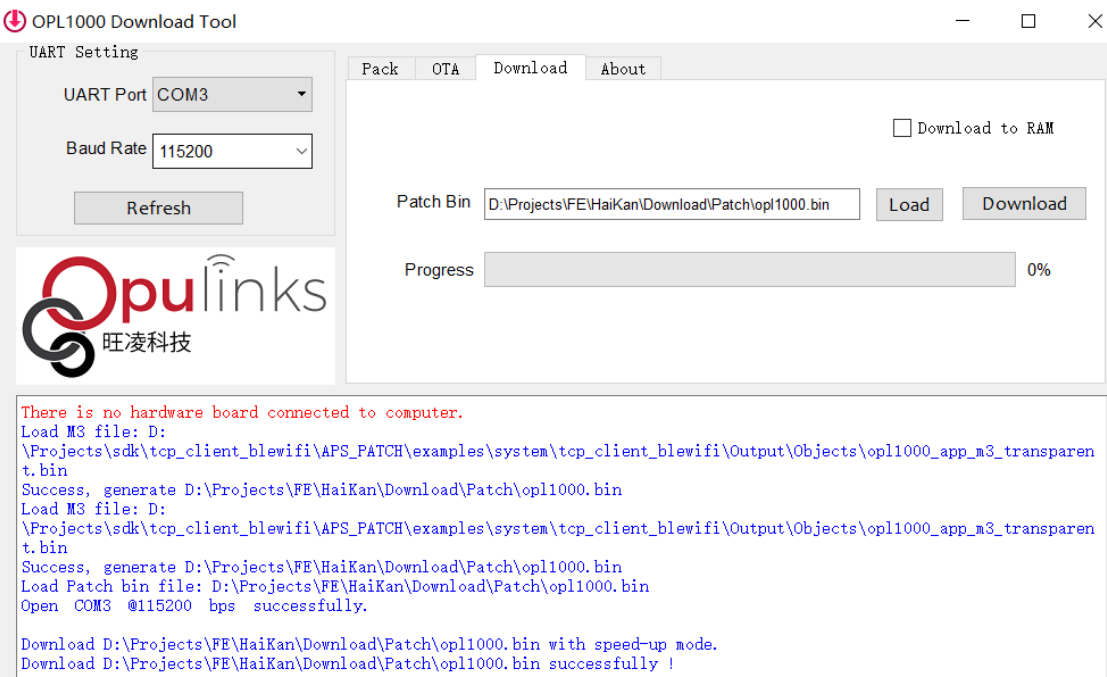
编译成功后, 在 download tool 的 pack 界面上，选择相应的 opl1000_app_m3.bin 文件，将它 pack 成 OPL1000.bin 如下。

Figure 16: PACK OPL1000 固件



在 pack 好 OPL1000 固件后，用 download tool 下载固件到 OPL1000 模块如下：

Figure 17: 下载固件



关于 download tool 的更多信息，请参考 Download 工具使用指南 [OPL1000-patch-download-tool-user-guide.pdf](#)。

3.3. tcp client blewifi 功能验证

下载固件到 OPL1000 模块后，用户可开始验证本例程如下：

3.3.1. 开启蓝牙广播并完成蓝牙配网

1. 上电后拉低 IO5 引脚并保持 5 秒及以上的时间，以开启蓝牙配网；
2. 打开 OPL1000 手机 APP，完成蓝牙配网后，OPL1000 在获得 IP 后会自动尝试和下节描述的网络调试助手（用作 TCP server）建立连接；

Figure 18: 完成蓝牙配网

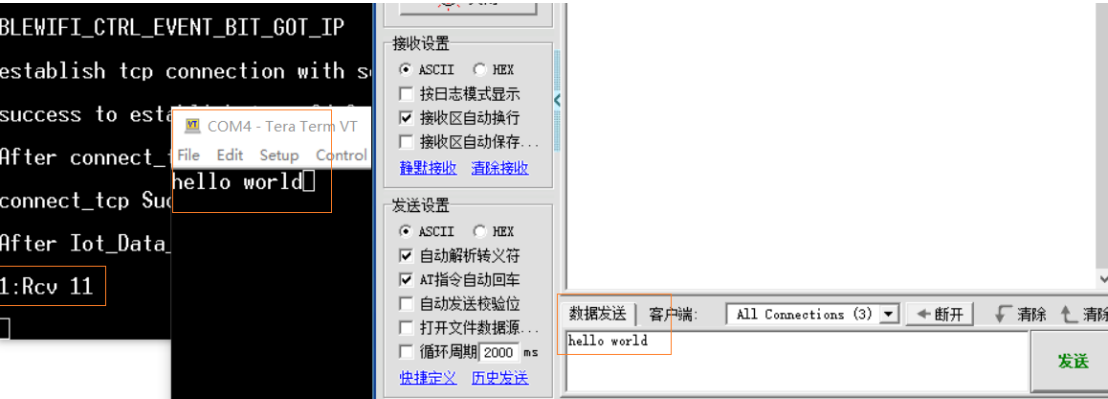


3.3.2. PC 端执行网络调试助手协助验证

PC 端启动网络调试助手程序。选择 TCP_server 协议，填写 IP 地址和端口号。注意一定要和 tcp_client.h 文件中定义的 TCP_SERVER_ADDR、TCP_SERVER_PORT 一致。点击“连接”按钮，启动 TCP Server 服务。

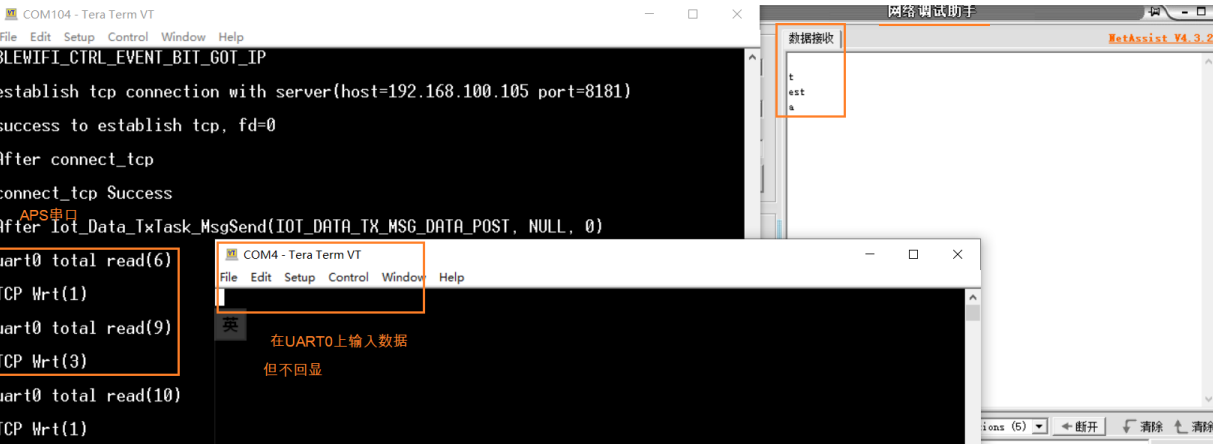
下图展示了 RX path 通信成功的执行结果。在网络调试助手上发送数据（hello world）后，UART0（COM4）上会接收到这串数据，而在 APS 口上则会显示收到这个字符串的提示。

Figure 19: OPL1000 和 TCP Server 间 RX path 通信示例



下图展示了 TX path 通信成功的执行结果。在 PC 上打开 UART0（COM4），输入一些字符（不回显）；OPL1000 接收到这些字符后，在 APS 串口上输入提示信息，并将字符串（数据）发送到 TCP server,即网络调试助手上。

Figure 20: OPL1000 和 TCP Server 间 TX path 通信示例



3.3.3. 睡眠模式设置和验证

在连上 TCP server 后，用 `at+sysmode=2` 命令切换为用户模式，再用 `at+sleep=1,17` 命令进入睡眠模式。在把 IO17 拉高后又可唤醒 OPL1000。

Figure 21: 切换睡眠模式



CONTACT

sales@Opulinks.com