如何构建 STM32WBx5 微控制器的低功耗蓝牙® mesh 应用

引言

低功耗蓝牙®技术(BLE)-mesh 通过物联网（IoT）解决方案的 mesh 网络功能来连接多个低功耗技术器件。它通过嵌入式低功耗蓝牙通信技术将 STM35WBx5 产品集成到一个支持双向无线通信的强大且范围可扩展的 mesh 网络中。

该解决方案不仅包含用于安全通信的所有核心功能，还提供了构建应用所需的所有灵活性。它使用了支持 mesh 栈 APIs 和相关事件回调的 STM32WBx5 器件。软件开发套件（SDK）以库形式提供了 mesh 栈并以源代码的形式提供示例应用程序，以演示如何使用库。

BLE mesh 可用于需要通过低功耗蓝牙®技术在 mesh 网络中进行不频繁的数据传输的多种应用，以创建诸如以下分布式控制系统：

- 智能照明
- 家居和建筑自动化
- 工业自动化

本应用笔记中的示例将与带演示示例的 P-NUCLEO-WB55 pack 一起使用。演示示例用于更改应用接口，使用库实现所需的硬件和软件功能。演示应用可用于 P-NUCLEO-WB55 pack。

演示应用实现了智能照明控制方案，并可以修改，以满足特定要求。

1 概述

本文档适用于基于 STM35WBx5 Arm® 的微控制器。

提示 *Arm 是 Arm Limited（或其子公司）在美国和/或其他地区的注册商标。*

2 入门指南

本文档中的示例应用实现了智能照明控制方案。要修改应用，按顺序执行以下步骤：

1. 第 1 步.将板连接到 PC
2. 第 2 步.在 IDE 中编译固件
3. 第 3 步.将固件刷写到板中
4. 第 4 步.在 STM32WB-mesh 应用中配置板
5. 第 5 步.使用应用程序切换板载 LED。

2.1 板接口

下表详细列出了 P-NUCLEO-WB55 pack 电源管理应用和用户接口，例如 LED 和按钮。

表 1. 硬件细节

器件	硬件套件	电源	编程端口	LED	按钮
STM32WBx5	P-NUCLEO-WB55 pack	Micro USB 线或 CR 2032 电池	USB 端口	3 个用户 LED + 2 个电源指示	复位按钮 + 3 个用户按钮

2.2 P-NUCLEO-WB55 套件板设置

使用 USB 线将主 P-NUCLEO-WB55 pack 板连接到主机 PC。该线缆用于两个目的：

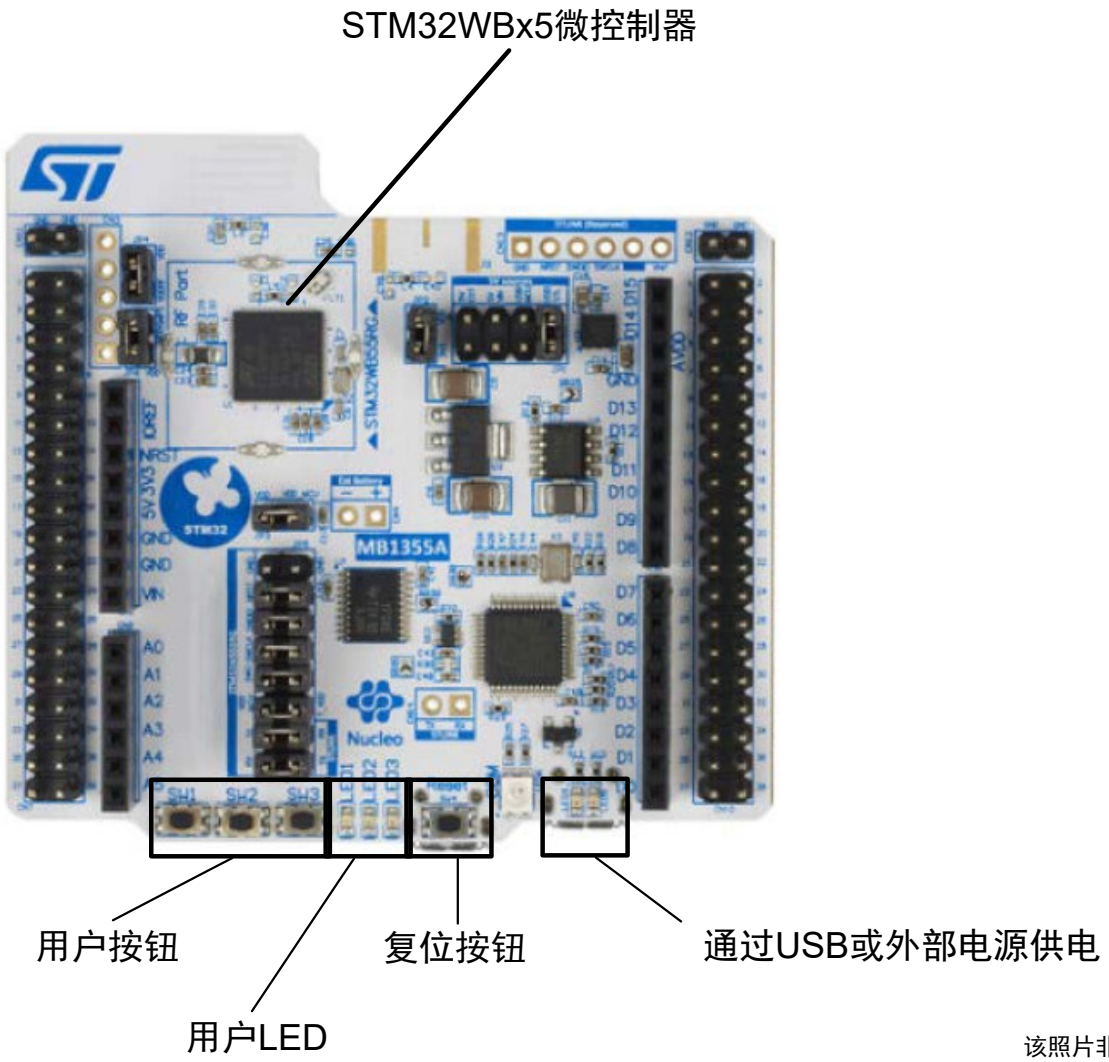
- 用于为 P-NUCLEO-WB55 pack 板供电；
- 在 PC 和 P-NUCLEO-WB55 pack 板之间建立一个串行数据链路。

P-NUCLEO-WB55 pack 板集成 ST-LINK/V2-1 调试工具/编程工具。可在 www.st.com 上获得 Microsoft® Windows®版的相关 STSW-LINK009 ST-LINK/V2-1 USB 驱动器版本。

通用板布局如图 1. STM35WBx5 板与 PC 的连接中所示。

提示 *确保将 STM35WBx5 升级为 BLE 栈版本 1.0.0 或更高版本。STM32 ST-LINK 实用程序 GUI 可用于管理更新。*

图 1. **STM35WBx5 板与 PC 的连接**



2.3 系统要求

设置和运行 BLE mesh 智能照明应用的最低系统要求为：

- 采用 Intel 或 AMD 处理器并运行以下 Microsoft 操作系统之一的 PC：
  - Windows®XP®
  - Windows®VISTA®
  - Windows 7®
  - Windows 10®
- 至少 128 MB RAM
- 2 个 USB 端口
- 有 40 MB 硬盘空间
- 开发工具链和编译器：
  - Keil® µVision® v5.23
  - IAR Embedded Workbench v8.20.2。

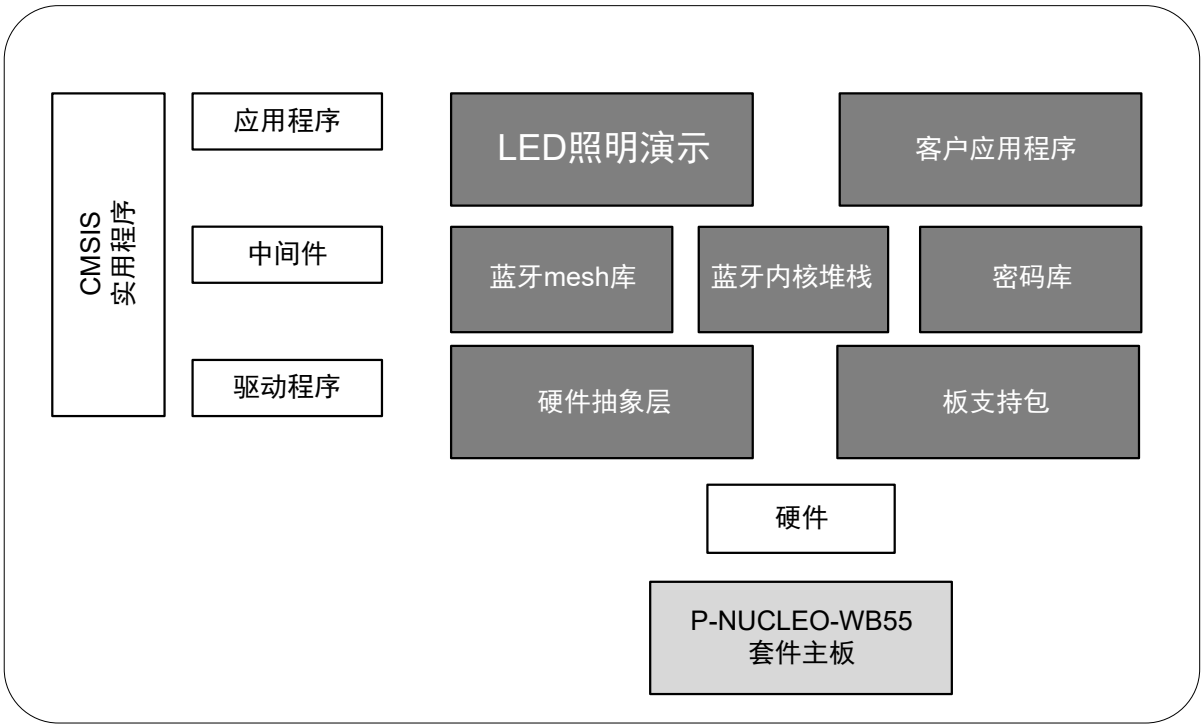
2.4 安装 BLE mesh

要安装 BLE mesh，请执行以下步骤：

1. 将数据包的内容解压到临时目录中。
2. 启动安装程序并按照屏幕上的说明进行操作
3. 安装在磁盘驱动器上的适当文件夹中。

3 固件结构

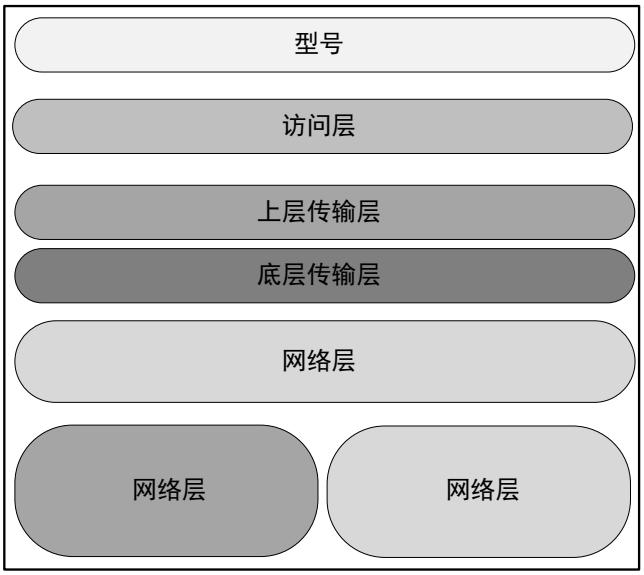
图 2. 固件架构



固件包用以下结构构建，如上图所示：

- 文档文件夹，其中包含通过源代码生成的一组已编译的 HTML 文件以及所有软件组件和 API 的详细描述；
- 驱动文件夹，其中包含硬件抽象层（HAL）、用于支持硬件平台和组件的板特定驱动，以及 Cortex®-M 处理器系列与供应商无关的 CMSIS 硬件抽象层。
- 中间件文件夹，其中包含 mesh 和 BLE 通信库：
  - 层组件之间的水平交互是通过调用特征 APIs 来直接执行的，而与底层驱动程序的垂直交互是通过库系统调用接口中实现的特定回调函数和静态宏来管理的；
  - BLE mesh 库 v1.09.000 根据蓝牙 SIG mesh 配置文件 v1.0 和 mesh 模型 v1.0 规范构建，如图 3. BLE mesh 库架构中所述。
- 项目文件夹，其中包含 P-NUCLEO-WB55 pack 板的 IAR Embedded Workbench 集成开发环境的工作区；
- EWARM 文件夹，其中包含 IAR Embedded 工作区。文件夹中的源文件通过绑定固件层来实现演示 BLE mesh 的功能。

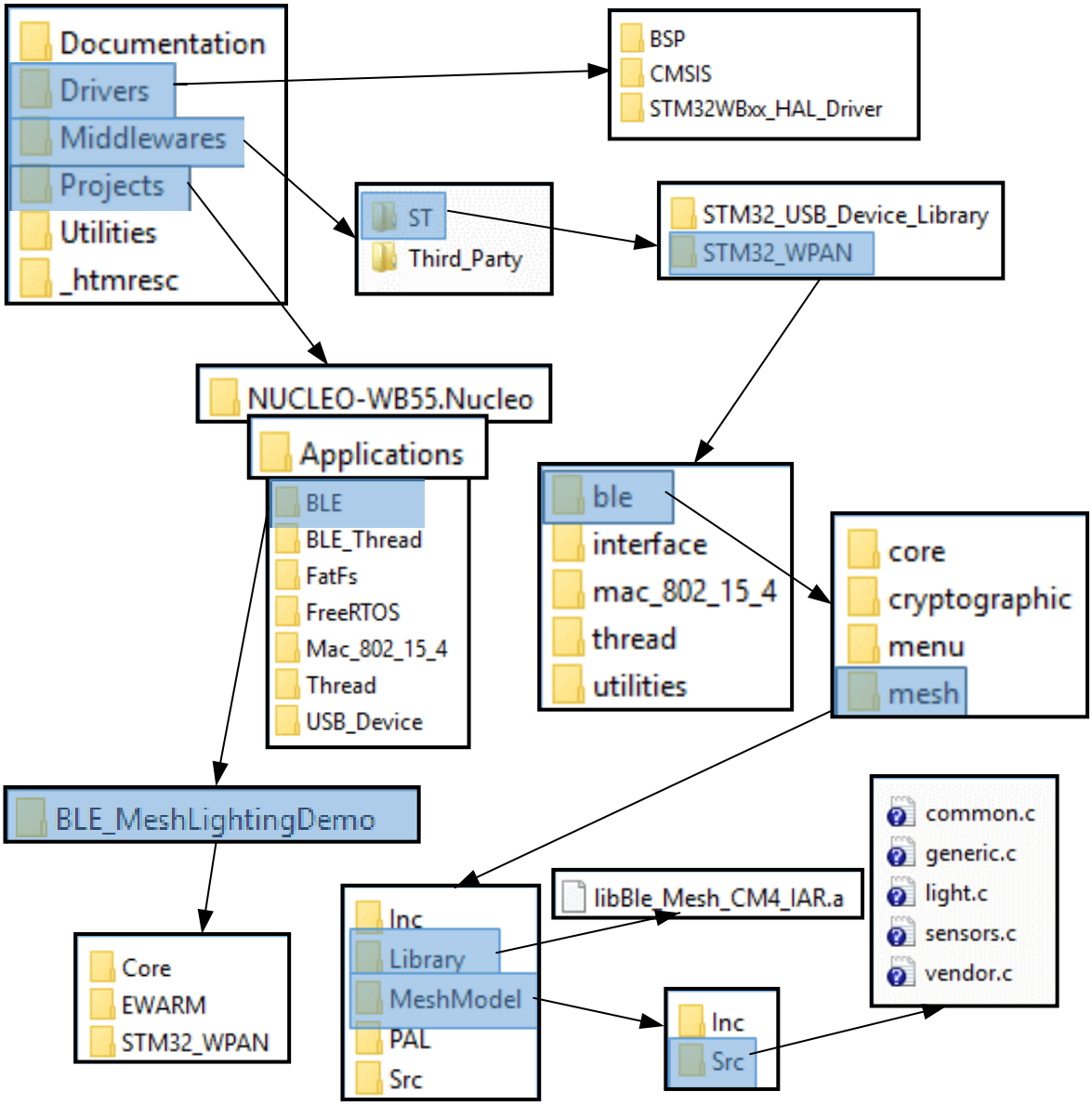
图 3. **BLE mesh 库架构**



提示 *GATT 代表通用属性*

文件夹结构如图 4. 文件夹、子文件夹和数据包内容中所述。

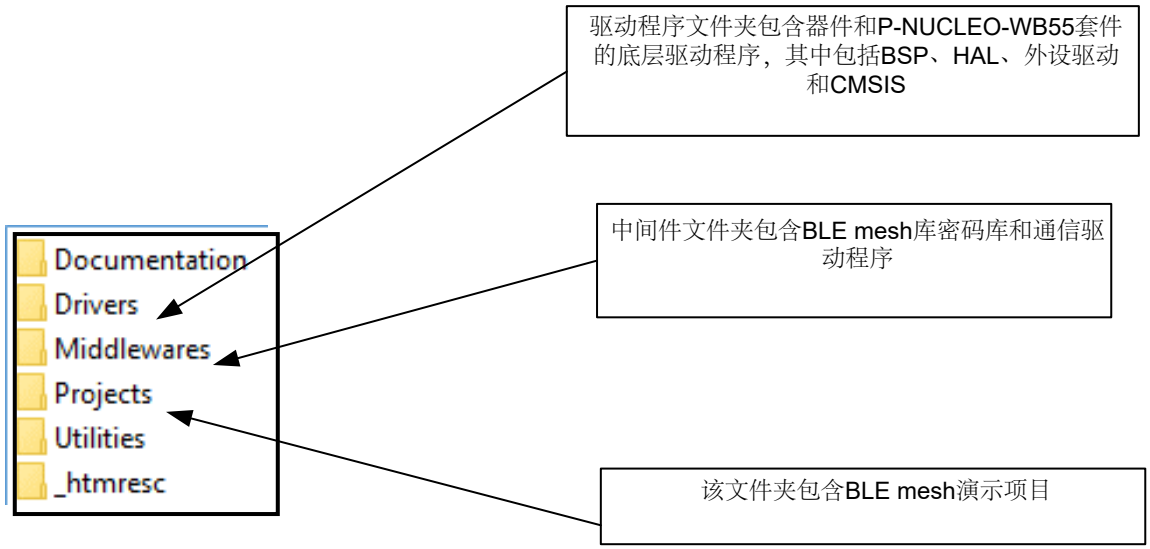
图 4. 文件夹、子文件夹和数据包内容



3.1 根文件夹

下图显示了固件包的根文件夹结构。

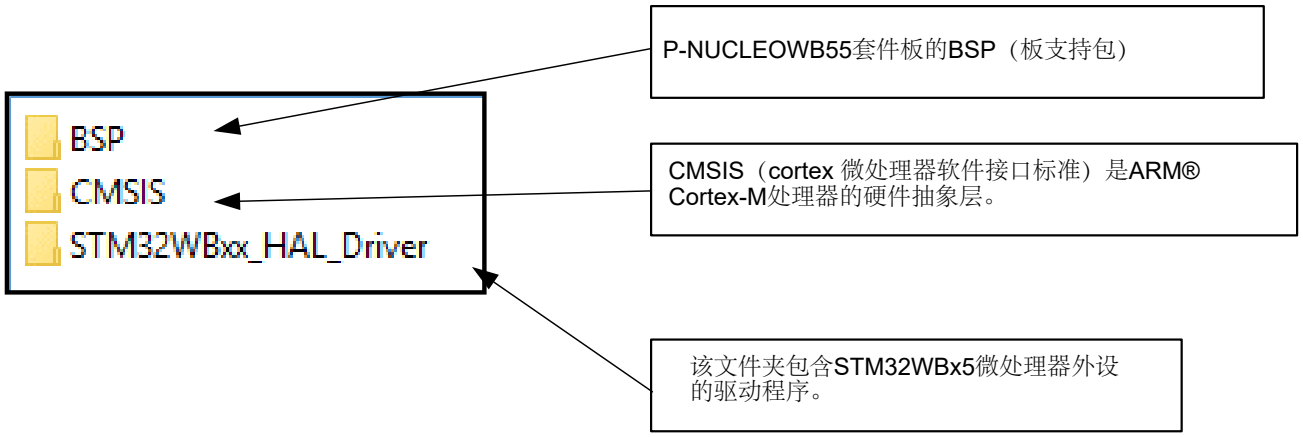
图 5. 根文件夹结构



3.2 驱动文件夹

驱动文件夹包含所有底层驱动，这包括外设驱动以及与硬件对应的 HAL 驱动。其结构详见下图。

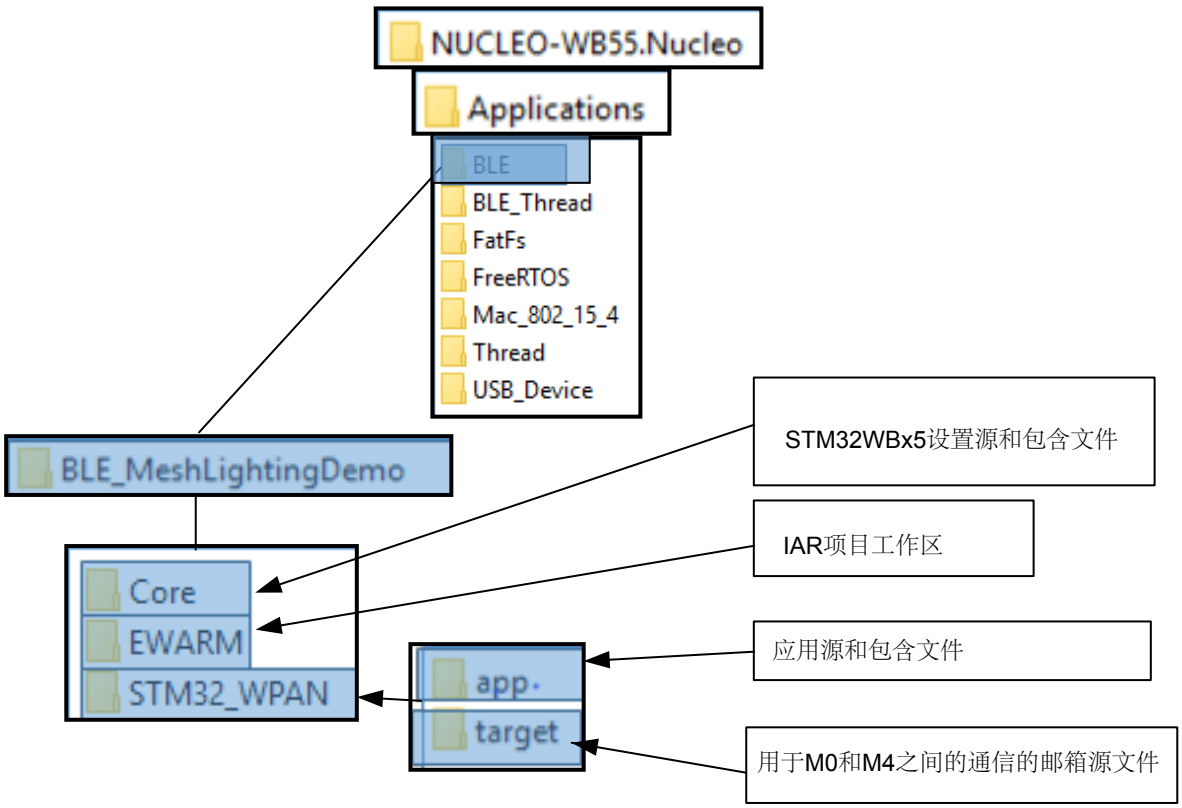
图 6. 驱动文件夹



3.3 项目文件夹

项目文件夹包含 IAR 项目，如下图所示。

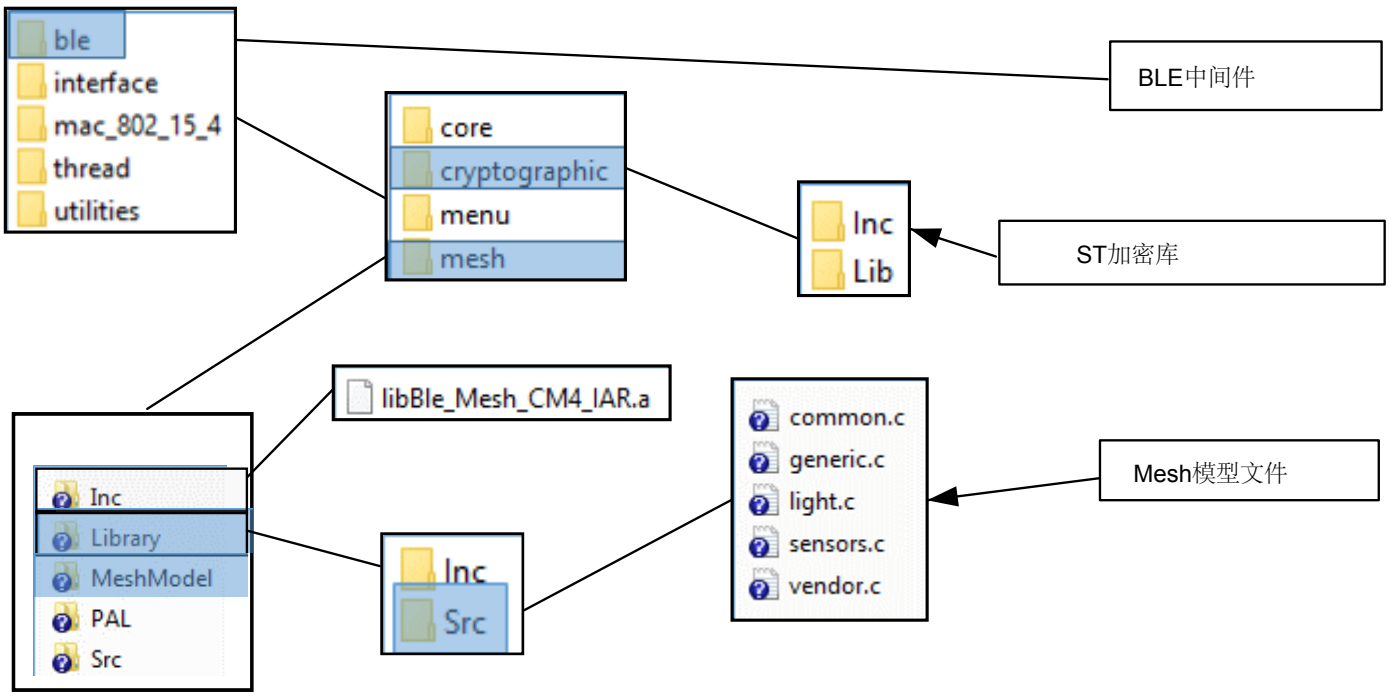
图 7. 内部项目文件夹



3.4 中间件文件夹

中间件文件夹（如下图所示）包含中间件项目，该项目包括用于 STM35WBx5 的预编译 mesh 库。

图 8. 中间件文件夹



4 使用 BLE mesh 演示

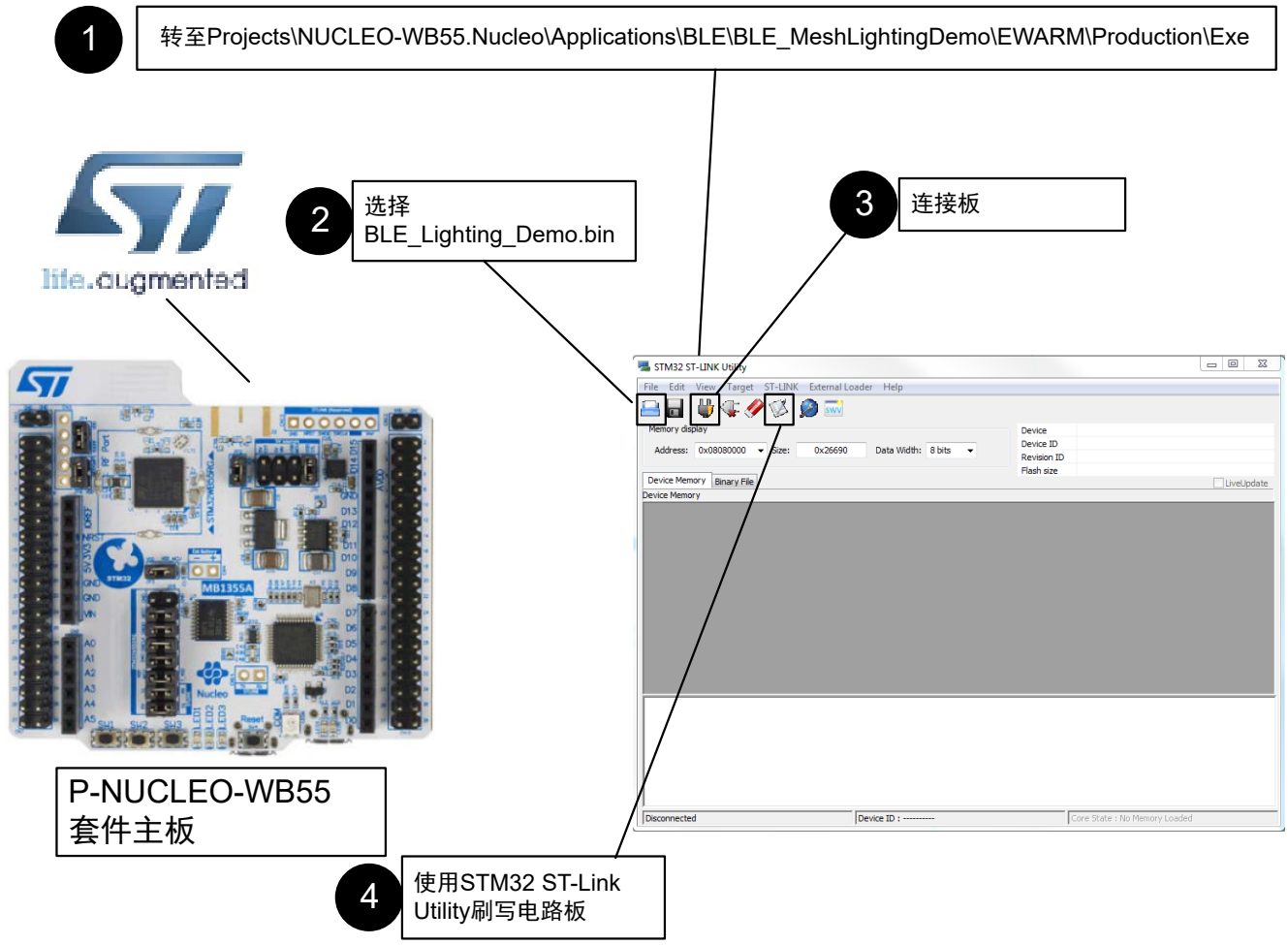
嵌入式 IAR 工作台用于调试固件和将固件烧写到 P-NUCLEO-WB55 pack 板 Flash 存储器中。

4.1 使用 STM32WBx5 二进制文件

1. 预编译的二进制文件位于 Project\<Platform>\Application\<Demo>\ EWARM\<Demo Name>\Exe 文件夹中。例如，对于 STM32WB55RG，路径为 `Projects\NUCLEO-WB55.Nucleo\Applications\BLE\ble_mesh_lighting_demo\EWARM\Lighting\Exe\BLE_Lighting_Demo.bin`；
2. 可使用不同的实用程序（如 ST LINK）或任何集成开发环境在器件中对这些二进制文件进行编程；
3. 使用“ST-LINK Utility”工具对通过 mini-USB 线连接的 P-NUCLEO-WB55 pack 板进行编程；
4. 或者，只需将.bin 文件拖放到 P-NUCLEO-WB55 套件板上的 ST-LINK 所创建的驱动程序中。在 P-NUCLEO-WB55 pack 板上的 STM35WBx5 器件中对二进制文件进行编程。

该过程如下图所示。

图 9. 有关如何对 **STM32WB55RG MCU** 中的二进制文件进行编程的概述



4.2 使用 IAR 项目

1. 选择文件/打开/工作区。从相应位置选择 `ble_mesh_lighting_demo.eww` 文件。例如，对于STM35WBx5 BLE mesh 演示，位置如下：`stm32wb_M4_Firmware\Firmware\Projects\NUCLEO-WB55.Nucleo \Applications\BLE\ble_mesh_lighting_demo\EWARM`。
2. 项目将在工作区中打开。该项目包含所有项目文件和文件夹（如驱动程序文件、应用程序文件、中间件、输出文件.....）。
3. 使用 Project / Make 选项来构建项目。
4. 按下载和调试按钮调试代码，并将其写入板中。
5. 按“运行”按钮来运行程序。

5 固件初始化和配置

本节介绍了基于低功耗蓝牙（BLE）器件 mesh 网络的应用开发的可用 API 函数。

BLE mesh 库管理以下功能：

- 在节点之间创建 mesh 网络
- 处理单播、广播寻址
- 管理中继功能：重新发送目标地址为其他节点的所有数据包
- 与设备通信，以实现配置和代理服务等高级功能
- 处理特定于平台的通信。

用户应用程序处理以下内容：

- mesh 堆栈初始化
- 应用程序所需的用户回调
- 应用程序处理。

以下章节介绍了固件初始化和配置的要求。

5.1 设置节点的发送功率

通过初始化 mesh 库回调来定义节点发送功率。其运行如下：

```
{
    Appli_BleStackInitCb,
    Appli_BleSetTxPowerCb,
    Appli_BleGattConnectionCompleteCb,
    Appli_BleGattDisconnectionCompleteCb,
    Appli_BleUnprovisionedIdentifyCb,
    Appli_BleSetUUIDCb,
    Appli_BleSetNumberOfElementsCb
};
```

`Appli_BleSetTxPowerCb()` 通过调用 aci 函数来设置功率 `aci_hal_set_tx_power_level(uint8_t En_High_Power, uint8_t PA_Level);`

默认情况下，在 STM35WBx5 的节点中采用+4 dbm 的配置，但用户可以更改该配置。

5.2 固件 UART 接口

板可以通过 USB 连接连接到 PC。任何软件终端（如使用 HyperTerminal、Hercules 或 Putty）都可用于打开 PC 上的串行通信端口，以检查板发送的消息。

板 UART 控制器通过虚拟通信（VCOM）端口连接到 PC。打开通信端口的设置为：

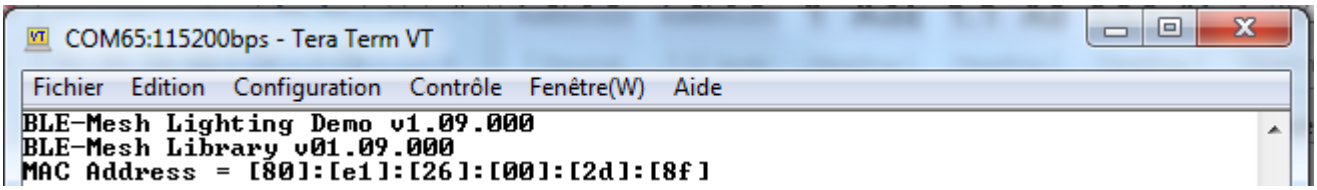
- 波特率：115200
- 数据大小：8
- 校验位：无
- 停止位：1
- 无硬件控制。

可使用以下代码通过 VCOM 打印固件 main.c 文件中的某些消息：

```
#if !defined(DISABLE_TRACES) /* 打印板的 MAC 地址 */
   printf("BLE-Mesh Lighting Demo v%s\n\r", BLE_MESH_APPLICATION_VERSION);
   printf("BLE-Mesh Library v%s\n\r", BLEMesh_GetLibraryVersion());
   printf("BD_MAC Address = [%02x]:[%02x]:[%02x]:[%02x]:[%02x] \n\r",
   bdaddr[5],bdaddr[4],bdaddr[3],bdaddr[2],bdaddr[1],bdaddr[0] );
#endif
```

在板已连接且终端窗口打开后，按复位按钮。如果固件成功启动，则会打开一个窗口，其中显示虚拟 COM 窗口的消息，如下图所示。

图 10. VCOM 窗口



5.3 MAC 地址管理

mesh 网络中的每个节点均需要唯一的 MAC 地址。下表列出了为节点配置 MAC 地址的可用选项

表 2. MAC 地址管理

号	MAC 地址管理	注释
1	使用外部 MAC 地址	用户可以使用所需的唯一 MAC 地址对节点进行编程。该地址存储在闪存中的特定位置。用户负责确保器件中的编程 MAC 地址符合蓝牙通信要求。
2	使用唯一器件序列号	可以使用每个器件中可用的唯一序列号配置器件的 MAC 地址。 这是默认设置。
3	使用静态随机 MAC 地址	可以使用静态随机 MAC 地址配置器件的 MAC 地址

5.4 按钮使用

下表列出了使用板载按钮可以执行的操作。

表 3. 按钮使用和 LED 管理

特性	按钮	LED
Mesh 库错误	-	LED2 持续闪烁
Flash 访问出错	-	LED3 点亮
Mesh 库成功启动	-	未闪烁
配置	-	未闪烁
取消配置	• 按下复位 + SW1 按钮 • 松开复位按钮 • 长按 SW1 按钮会使配置取消 • 使板复位	LED1 点亮，然后在取消配置期间保持闪烁
从板运行演示	按下任何板上的 SW1 按钮。该操作将在发布地址列表中的节点上发送指令	• 板上的 LED1 切换（如果订阅发布地址） • 其他板上的 LED1 也将切换（如果订阅发布地址）
代理连接	-	LED2 点亮

5.5 应用回调初始化

通过初始化不同事件和功能所需的回调来启动应用配置。这些回调用于 BLE mesh 库，以根据特定事件或库状态机来调用函数。如下面的示例代码所示：

```
{
   Vendor_WriteLocalDataCb,
   Vendor_ReadLocalDataCb,
   Vendor_OnResponseDataCb
}; /* Callbacks used by BLE-Mesh library */
BLEMesh_SetVendorCbMap(&vendor_cb);
```

MOBLE_VENDOR_CB_MAP 结构用于初始化应用实现的供应商模型。
BLEMesh_SetVendorCbMap(&vendor_cb);函数用于初始化库中的供应商回调。

5.6 初始化和应用程序主循环

此过程为 STM32WBx5 平台上的 BLE mesh 开发应用程序。

1. 第 1 步.调用 HAL_Init() API。
   该函数用于初始化 HAL 库；它必须是在主程序中执行的第一条指令（在调用任何其他 HAL 函数之前），它执行以下操作：
   – 配置 Flash 预取、指令缓存和数据缓存
   – 通过配置 SysTick 来产生一毫秒的中断，该中断由 MSI 计时（在该阶段，时钟尚未配置，因此系统通过内部 MSI 以 4 MHz 的频率运行）。
   – 将 NVIC 组优先级设为 4。
   – 调用用户文件“stm32wbxx_hal_msp.c”中定义的 HAL_MspInit() 回调函数，以执行全局低级硬件初始化
2. 第 2 步.调用 Reset_Device() API，以重置备份域和 IPCC。
3. 第 3 步.调用 SystemClock_Config() API。
   此 API 配置：
   – 系统时钟源
   – AHBCLK、APBCLK 分频器
   – 闪存延迟
   – PLL 设置（必要时）
4. 第 4 步.调用__HAL_RCC_CRC_CLK_ENABLE() API，以启用外设时钟。
5. 第 5 步.调用 SystemPower_Config() API，以配置系统低功耗模式。
6. 第 6 步.调用 Config_RTC() API，以初始化 RTC IP 和定时器服务器。
7. 第 7 步.调用 Init_Debug() API，以配置用于调试跟踪的 UART。

8. 第 8 步.调用 APPE_Init() API，以初始化 LED、按钮和所有传输层。
   所有传输层初始化均通过以下方式调用了 mesh 初始化：
   – 调用 Appli_CheckBdMacAddr() API，以检查 MAC 地址的有效性。
   – 如果 MAC 地址无效，则固件会卡在 while(1)循环中，LED 将持续闪烁。
   – 初始化 BLE 硬件的硬件回调函数。通过更新以下 API 完成：

```
MOBLE_USER_BLE_CB_MAP user_ble_cb =
{
  Appli_BleStackInitCb,Appli_BleSetTxPowerCb,Appli_BleGattConnectionCompleteCb,
  Appli_BleGattDisconnectionCompleteCb,
  Appli_BleUnprovisionedIdentifyCb,
  Appli_BleSetUUIDCb,
  Appli_BleSetNumberOfElementsCb
};
```

   这些 API 对于用于 BLE 无线电初始化和 TxPower 配置的应用接口很有用。
   – 为应用接口初始化 GATT 连接和断开连接回调。
   – 可通过调用 BLEMesh_BleHardwareInitCallBack(&user_ble_cb)来完成硬件回调的初始化。
   – 通过调用 BLEMesh_Init(&BLEMeshlib_Init_params)来初始化 BLE mesh 库。
   通过更新包含 mesh 库初始化信息数据的结构体来完成。

```
const Mesh_Initialization_t BLEMeshlib_Init_params =
{
  bdaddr,
  &FnParams,
  &LpnParams,
  MESH_FEATURES,
  &DynBufferParam
};
```

   如果发生错误，演示固件会在为板上 USB 连接创建的 VCOM 端口打开的终端窗口上打印一条消息，Appli_LedBlink() API 会使 LED 持续闪烁。
   – 可进行几项检查
     ◦ 检查器件是否已配置。已配置的器件具有在内部 Flash 存储器中配置的网络密钥和其他参数。
     ◦ 可通过 BLEMesh_IsUnprovisioned() API 进行检查。如果节点未配置，BLEMesh_InitUnprovisionedNode() API 会对其进行初始化。

   如果器件已配置，则 BLEMesh_InitprovisionedNode() API 有助于初始化器件。
   – 打印正在初始化的节点的终端窗口消息。消息还会打印分配给节点的 MAC 地址。
   – 初始化 BLEMesh_ModelsInit();事件触发的所有 3 个模型（供应商、通用和照明模型）。
   – 检查按钮状态。要将节点初始化为未配置状态，请按住用户 SW1 按钮。当检测到取消配置 SW1 按钮序列时，BLEMesh_Unprovision() API 会擦除在设备内部存储器中配置的所有网络参数。完成取消配置后，板需要复位。
9. 第 9 步.在 while(1)循环中处理 MoBLE 和 HCI 事件。
   应用程序必须在 while(1)循环中尽可能频繁地调用 SCH_Run()、BLEMesh_Process()、BLEMesh_ModelsProcess()、Appli_Process()。
   此函数会在内部调用 MobleStackProcess()，以处理 BLE 通信。
   任何应用程序实现都是在状态机中通过频繁调用 BLEMesh_Process()的非阻止函数来执行。
10. 第 10 步.为要执行的任何操作定期检查用户输入或按钮。

6 Mesh 网络信息

6.1 本地和远程概念

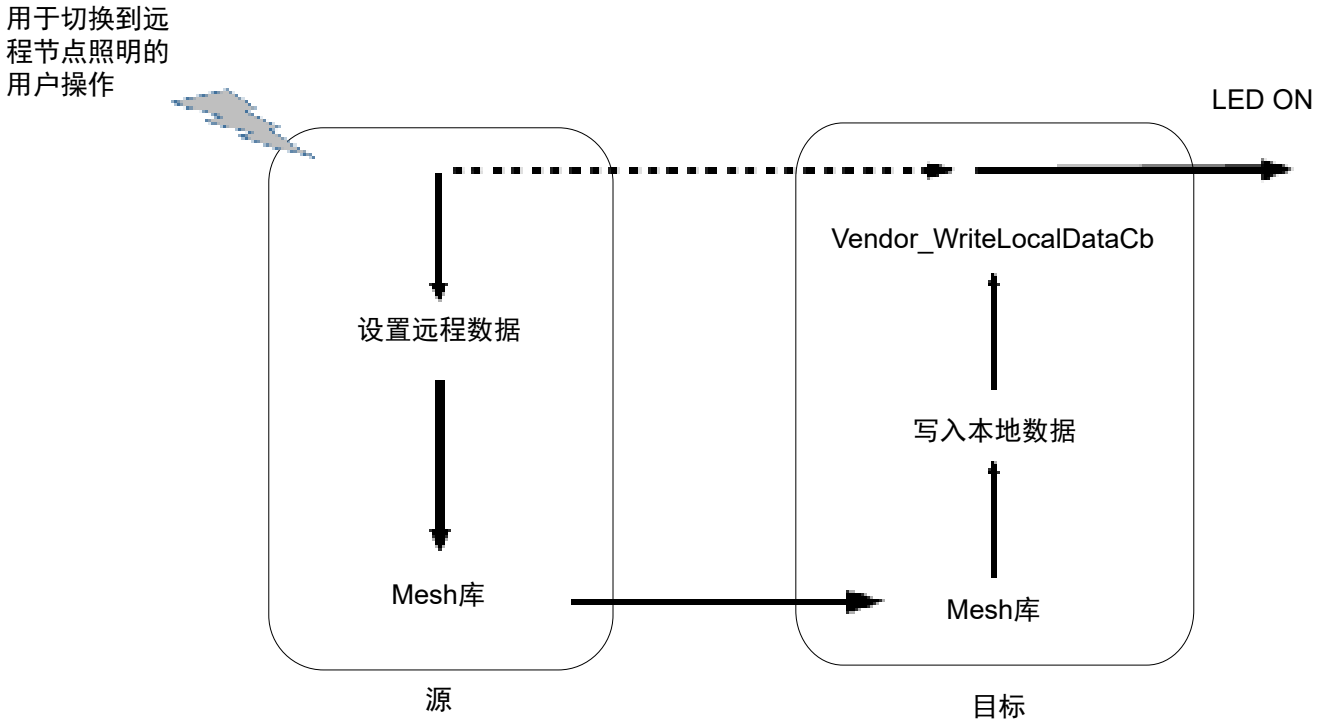
远程操作涉及其他网络节点上的操作，而本地操作则涉及同一网络节点上的资源。

例如，要检查 BLE mesh 应用节点上的 LEDs，请点击图标按钮，以从应用程序调用 `_SetRemoteData` 操作。这会导致节点上的 `Vendor_WriteLocalDataCb` 操作。

当需要控制电路板上的一组节点 LEDs 时，也是这种情况。按下电路板上的按钮时，寻址节点上的 LEDs 将切换。

要让节点收到相同的消息（对于任何 LEDs 操作），需要在该节点上执行 WriteLocal 操作。对于网络中不同节点上的 SetRemote/WriteLocal 操作，这如下图所示。

图 11. **SetRemote/WriteLocal 操作**



6.2 已确认和未确认的消息

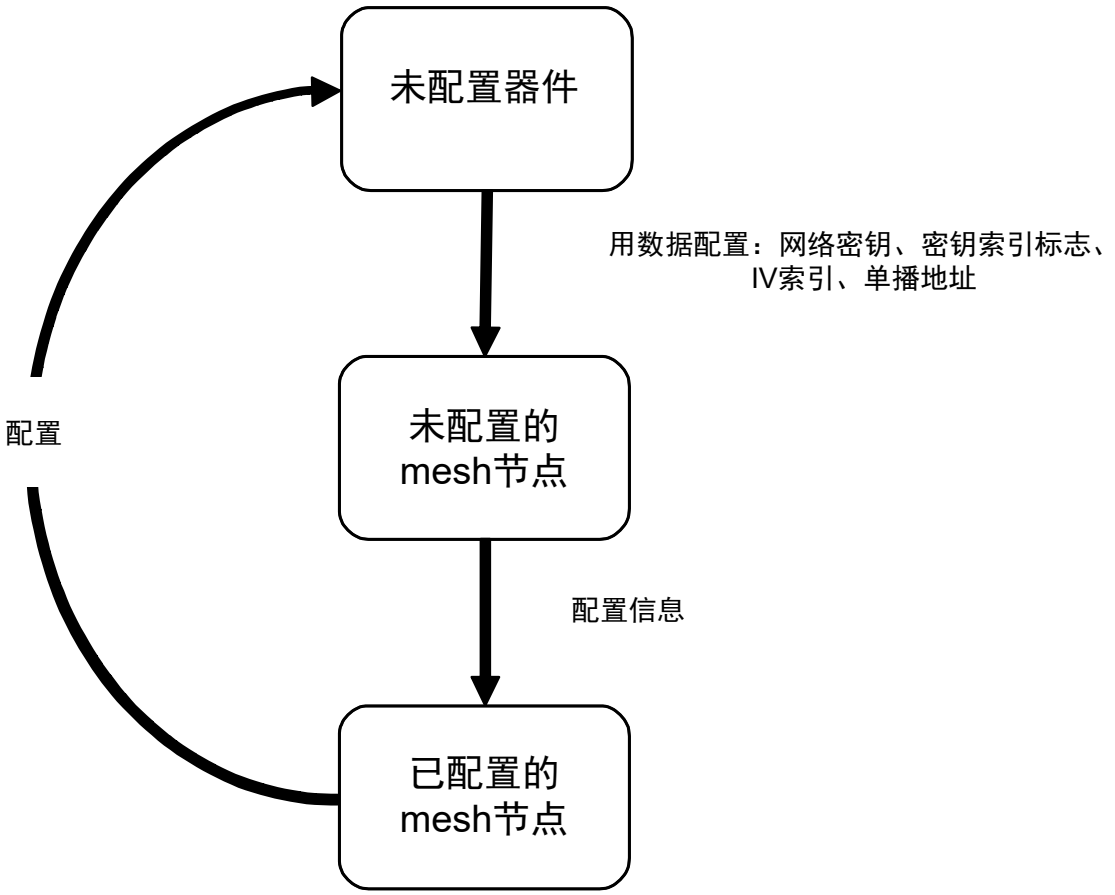
默认情况下，mesh 网络中的所有消息均被配置为未确认。已确认和未确认消息之间的区别在于对消息的响应。向节点写入消息可能获得已确认通信的响应。然而，在未经确认的通信中，可能不存在响应。

系统应配置为使用 mesh 网络中未确认的消息，以避免通过网络进行冗余消息交换，这可能会影响系统性能。

6.3 配置过程

在网络中配置设备的过程被称为配置。它由“配置程序”启动，这是在智能手机上运行的 BlueNRG-Mesh 应用程序。该循环如下图所示：

图 12. 配置过程


未配置器件
用数据配置：网络密钥、密钥索引标志、
IV索引、单播地址
未配置的
mesh节点
配置信息
已配置的
mesh节点
配置

BlueNRG-Mesh 移动应用程序可安装在支持 BLE 4.0 或更高版本的蓝牙的智能手机上。在安装过程中，应用程序会询问用户权限，有关更多信息，请参阅相关设备类型的 BlueNRG-Mesh 用户手册（Android™或 iOS™）。配置所需的步骤如下图所示。

图 13. 配置步骤


1
启动BlueNRG-Mesh应用程序
2
应用程序提醒开启蓝牙（如果未开启）
3
确定未配置的节点
4
用于识别节点的设备UUID
5
配置
6
配置过程启动
Add Device :
Provisioning . . .
40%
4/10
7
已配置的节点

配置通过智能手机和未配置设备之间的 GATT 连接来执行。

6.4 GATT 节点连接/断开

网络中的每个节点都可以通过 GATT 接口连接到智能手机。

在建立该连接后，节点将变成“代理”。代理负责桥接 mesh 网络和智能手机之间的指令和响应。通过以下回调来管理与智能手机的连接状态

```
Appli_BleGattConnectionCompleteCb;
Appli_BleGattDisconnectionCompleteCb;
```

这些回调在主循环期间执行初始化。当附近有很多节点时，可确定智能手机节点连接。

在配置过程中，与需要配置的节点建立 GATT 连接。如果智能手机移动到代理节点的范围之外，它将与下一个可用的节点建立新连接。LED 指示（LED2）用于显示代理连接。

6.5 从远程节点写入指令

从远程节点或智能手机到寻址节点的指令会调用 `WriteLocalData` 回调。

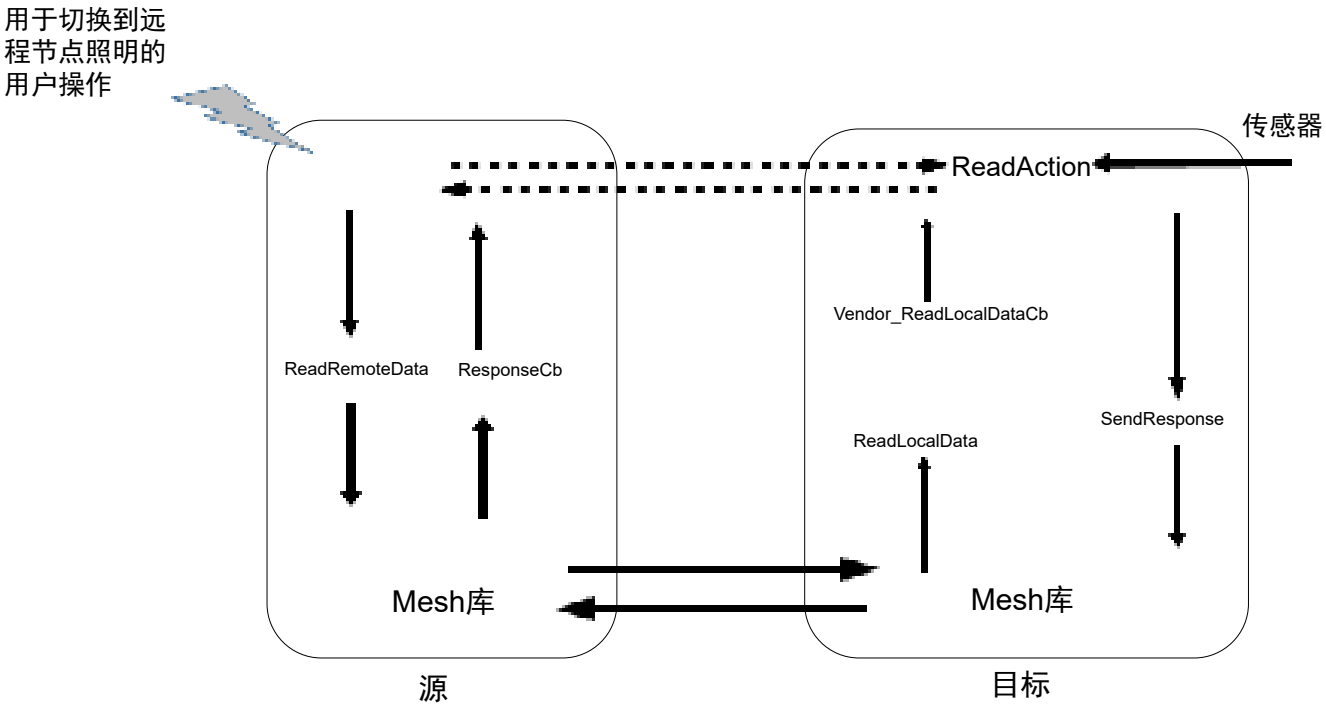
该回调可用于处理在网络中接收的指令或数据。在应用演示中，`Vendor_WriteLocalDataCb` 函数是处理数据或指令的回调函数。可在图 11. SetRemote/WriteLocal 操作中呈现指令/数据流。

通过 `SendResponse` 函数来发送节点响应数据。

6.6 从远程节点读取指令

从远程节点或从智能手机到寻址节点读取指令会使用 `Vendor_ReadLocalDataCb` 回调。此回调可用于提供远程节点正在请求的信息。在应用演示中，`Vendor_ReadLocalDataCb` 函数是处理读取指令的回调。可在下图中查看指令/数据流。

图 14. 从远程节点读取指令



通过 BLEMesh_SendResponse 函数来发送节点响应数据。

6.7 应用程序函数和回调

6.7.1 用户界面和说明

表 4. **Appli_LedCtrl**

功能	
原型	`void Appli_LedCtrl(void)`
函数说明	使板载 LED 闪烁。该功能在上电时使用，用于引起对错误状况的注意
输入参数	无
输出参数	Void

6.7.2 用户和按钮界面

表 5. **Appli_ShortButtonPress**

功能	说明
原型	`static void Appli_ShortButtonPress(void)`
函数说明	短按按钮时调用
输入参数	Void
输出参数	Void

表 6. **Appli_LongButtonPress**

功能	说明
原型	`static void Appli_LongButtonPress(void)`
函数说明	长按按钮时调用
输入参数	Void
输出参数	Void

表 7. **Appli_UpdateButtonState**

功能	说明
原型	`static void Appli_UpdateButtonState(int isPressed)`
函数说明	更新按钮状态
输入参数	Int isPressed
输出参数	Void

6.7.3 设备 BLE 配置类型接口

本节介绍了应用开发人员可在网络中使用的设备配置功能，如下表所示。

表 8. **Appli_BleStackInitCb**

功能	说明
原型	`static void Appli_BleStackInitCb()`
函数说明	此功能有助于硬件配置；主要是根据 `BlueNRG_Stack_Init_params` 中定义的结构化参数来初始化 BLE 栈。 用户可以修改在 CONFIG_TABLE 中定义的设备底层硬件配置数据，如 `LOW_SPEED_SOURCE` 和 `HS_STARTUP_TIME`，这可能取决于用户硬件设备。 这些参数用于初始化 BlueNRG 栈
输入参数	无
输出参数	`MOBLE_RESULT` 结果状态

表 9. **Appli_BleSetTxPowerCb**

功能	说明
原型	`MOBLE_RESULT Appli_BleSetTxPowerCb()`
函数说明	此回调设置 BLE 无线电的传输功率。此函数调用 `aci_hal_set_tx_power_level`。默认情况下，将功率级设为 +4 dbm
输入参数	无
输出参数	`MOBLE_RESULT` 结果状态

表 10. **Appli_BleGattConnectionCompleteCb**

功能	说明
原型	`void Appli_BleGattConnectionCompleteCb(void)`
函数说明	当节点检测到 GATT 连接时，将调用此函数。应用程序可使用此回调向用户指示该节点已连接到智能手机。
输入参数	Void
输出参数	Void

表 11. **Appli_BleGattDisconnectionCompleteCb**

功能	说明
原型	`void Appli_BleGattDisconnectionCompleteCb(void)`
函数说明	当节点检测到 GATT 断开时，将调用此函数。应用程序可使用此回调向用户指示该节点已不再连接到智能手机。
输入参数	Void
输出参数	Void

6.7.4 供应商模型网络数据通信功能

下面介绍的功能有助于开发人员管理网络数据通信并采取适当的措施。

表 12. Vendor_WriteLocalDataCb

功能	说明
原型	`MOBLE_RESULT Vendor_WriteLocalDataCb (MOBLE_ADDRESS peer_addr, MOBLE_ADDRESS dst_peer, MOBLEUINT8 command, MOBLEUINT8 const *data, MOBLEUINT32 length, MOBLEBOOL response)`
函数说明	在节点本身需要操作时回调函数
输入参数	MOBLE_ADDRESS peer_addr：对端地址。 MOBLE_ADDRESS dst_peer：命令的目标地址。这可能为订阅节点的一组地址，也可能是元素。“MOBLEUINT16 command：已收到操作”的单播地址。 MOBLEUINT8 const *data：指向数据的指针 MOBLEUINT32 length：数据长度 MOBLEBOOL response：如果为 MOBLE_TRUE，则发送方将收到确认信息
输出参数	MOBLE_RESULT 结果状态

表 13. Vendor_ReadLocalDataCb

功能	说明
原型	`MOBLE_RESULT Vendor_ReadLocalDataCb( MOBLE_ADDRESS peer_addr, MOBLE_ADDRESS dst_peer, MOBLEUINT8 command, MOBLEUINT8 const *data, MOBLEUINT32 length, MOBLEBOOL response)`
函数说明	节点需要数据时调用回调函数
输入参数	MOBLE_ADDRESS peer_addr：对端地址 MOBLE_ADDRESS dst_peer：命令的目标地址。这可能为订阅节点的一组地址，也可能是元素的单播地址。 MOBLEUINT16 command：收到的处理命令 MOBLEUINT8 const *data 指针 MOBLEUINT32 length：数据长度 MOBLEBOOL response：如果为 MOBLE_TRUE，则发送方期待收到确认信息。
输出参数	MOBLE_RESULT 结果状态

6.7.5 MAC 地址配置

下表列出了用于配置 MAC 地址的函数。

表 14. Appli_CheckBdMacAddr

功能	说明
原型	`int Appli_CheckBdMacAddr(void)`
函数说明	检查 MAC 地址的有效性
输入参数	Void
输出参数	结果中断状态

表 15. Appli_GetMACFromUniqueNumber

功能	说明

原型	`#ifdef INTERNAL_UNIQUE_NUMBER_MAC`
	`static void`
	`Appli_GetMACfromUniqueNumber(void)`

函数说明	读取设备的 OTP BD 地址，并通过该地址生成 MAC 地址。
输入参数	Void
输出参数	Void

6.7.6 BLE mesh 节点配置

下表列出了用于配置要在网络中使用的节点的函数。

表 16. BLEMesh_InitUnprovisionedNode

功能	说明
原型	`MOBLE_Result BLEMesh_InitUnprovisionedNode(void)`
函数说明	初始化未配置的节点
输入参数	Void
输出参数	MOBLE_RESULT 结果状态

表 17. BLEMesh_InitProvisionedNode

功能	说明
原型	`MOBLE_Result BLEMesh_InitProvisionedNode(void)`
函数说明	初始化已配置的节点。
输入参数	Void
输出参数	MOBLE_RESULT 结果状态。

表 18. BLEMesh_GetUnprovisioneState

功能	说明
原型	`MOBLEUINT8 BLEMesh_GetUnprovisionedState(void)`
函数说明	获取配置过程状态。
输入参数	Void
输出参数	MOBLEUINT8 结果状态

表 19. BLEMesh_GetAddress

功能	说明
原型	`MOBLE_ADDRESS BLEMesh_GetAddress (void)`
函数说明	获取节点 mesh 地址。
输入参数	Void
输出参数	节点 mesh 地址

表 20. BLEMesh_GetPublishAddress

功能	说明
原型	`MOBLE_ADDRESS BLEMesh_GetPublishAddress(MOBLEUINT8 elementNumber)`
函数说明	获取节点发布地址
输入参数	元件编号
输出参数	节点发布地址

表 21. BLEMesh_GetSubscriptionAddress

功能	说明
原型	`MOBLE_RESULT BLEMesh_GetSubscriptionAddress( MOBLE_ADDRESS *addressList, MOBLEUINT8 *sizeOfList, MOBLEUINT8 elementNumber)`
函数说明	获取节点订阅地址
输入参数	SizeofList, elementNumber
输出参数	MOBLE_RESULT 订阅地址结果列表状态

表 22. BLEMesh_SetTTL

功能	说明
原型	`MOBLE_RESULT BLEMesh_SetTTL (MOBLEUINT8 ttl)`
函数说明	设置默认 TTL 值
输入参数	MOBLEUINT8 ttl 值
输出参数	MOBLE_RESULT 结果状态。

表 23. **BLEMesh_GetTTL**

功能	说明
原型	`MOBLEUINT8 BLEMesh_GetTTL (void)`
函数说明	获取默认 TTL 值
输入参数	Void
输出参数	MOBLEUINT8 默认 TTL 值

表 24. **BLEMesh_SetNetworkTransmitCount**

功能	说明
原型	`MOBLE_RESULT BLEMesh_SetNetworkTransmitCount (MOBLEUINT8 count)`
函数说明	设置网络传输计数值
输入参数	MOBLEUINT8 计数、网络传输值。支持的值为 1-8
输出参数	MOBLE_RESULT 结果状态

表 25. **BLEMesh_GetNetworkTransmitCount**

功能	说明
原型	`MOBLE_RESULT BLEMesh_SetNetworkTransmitCount (MOBLEUINT8 count)`
函数说明	设置网络传输计数值
输入参数	MOBLEUINT8 计数、网络传输值。支持的值为 1-8
输出参数	MOBLE_RESULT 结果状态

表 26. **BLEMesh_SetRelayRetransmitCount**

功能	说明
原型	`MOBLE_RESULT BLEMesh_SetRelayRetransmitCount (MOBLEUINT8 count)`
函数说明	设置中继重传计数值
输入参数	MOBLEUINT8 count 中继重传值。支持的值为 1-8
输出参数	MOBLE_RESULT 结果状态

表 27. **BLEMesh_GetRelayRetransmitCount**

功能	说明
原型	`MOBLE_RESULT BLEMesh_GetRelayRetransmitCount (void)`
函数说明	获取中继重传计数值
输入参数	Void
输出参数	MOBLEUINT8 默认中继重传计数值

表 28. **BLEMesh_SetHeartbeatCallback**

功能	说明
原型	`MOBLE_RESULT BLEMesh_SetHeartbeatCallback (MOBLE_HEARTBEAT_CB cb)`
函数说明	设置用户处理心跳消息的回调
输入参数	MOBLE_HEARTBEAT_CB cb 回调
输出参数	MOBLE RESULT 结果状态

表 29. **BLEMesh_SetAttentionTimerCallback**

功能	说明
原型	`MOBLE_RESULT BLEMesh_SetAttentionTimerCallback (MOBLE_ATTENTION_TIMER_CB cb)`
函数说明	设置注意定时器的回调。用于配置期间的信息和健康模型
输入参数	MOBLE_ATTENTION_TIMER_CB cb 回调
输出参数	MOBLE RESULT 结果状态

6.7.7 BLE mesh 库配置

下表列出了用于配置和初始化 ST BLE mesh 库的功能。

表 30. **BLEMesh_Init**

功能	说明
原型	`MOBLE_RESULT BLEMesh_Init(Const Mesh_Initialization_t* pInit_params)`
函数说明	用于初始化 ST BLE mesh 库
输入参数	蓝牙设备地址、库支持的功能、低功耗 节点、好友和动态缓冲区参数
输出参数	MOBLE RESULT 结果状态

表 31. **BLEMesh_GetLibraryVersion**

功能	说明
原型	`Char * BLEMesh_GetLibraryVersiont(void)`
函数说明	用于获取最新的库版本
输入参数	Void
输出参数	字符串

表 32. BLEMesh_GetLibrarySubVersion

功能	说明
原型	`Char * BLEMesh_GetLibrarySubVersiont(void)`
函数说明	获取最新的库子版本
输入参数	Void
输出参数	字符串

表 33. BLEMesh_Process

功能	说明
原型	`MOBLE_RESULT BLEMesh_Process(void)`
函数说明	Mesh 任务处理功能
输入参数	Void
输出参数	MOBLE RESULT 结果状态

表 34. BLEMesh_SetRemoteData

功能	说明

原型	`MOBLE_RESULT BLEMesh_SetRemoteData(`
	`  MOBLE_ADDRESS peer,`
	`  MOBLEUINT8 command,`
	`  MOBLEUINT8 const * data,`
	`  MOBLEUINT32 length,`
	`  MOBLEUINT8 response)`

函数说明	设置给定对端上的远程数据
输入参数	对端目标地址， 供应商模型指令， 数据缓冲， 以字节为单位的数据长度， 响应（如果不是“0”，用于获得响应。如果为“0”，则无响应）
输出参数	MOBLE RESULT 结果状态

表 35. BLEMesh_ReadRemoteData

功能	说明

原型	`MOBLE_RESULT BLEMesh_ReadRemoteData(`
	`  MOBLE_ADDRESS peer,`
	`  MOBLEUINT16 command）`

函数说明	读取给定对端上的远程数据
输入参数	对端目标地址， 供应商模型指令
输出参数	MOBLE RESULT 结果状态

表 36. **BLEMesh_SendResponse**

功能	说明

原型	`MOBLE_RESULT BLEMesh_SendResponse(`
	`  MOBLE_ADDRESS peer,`
	`  MOBLEUINT8 status,`
	`  MOBLEUINT8 const * data,`
	`  MOBLEUINT32 length)`

函数说明	发送所收包的响应
输入参数	对端目标地址， 响应状态， 数据缓冲， 以字节为单位的数据长度
输出参数	MOBLE RESULT 结果状态

版本历史

表 37. 文档版本历史

日期	版本	变更
2019 年 3 月 22 日	1	初始版本。

目录

图一览

表一览