青莲云嵌入式 SDK 使用文档

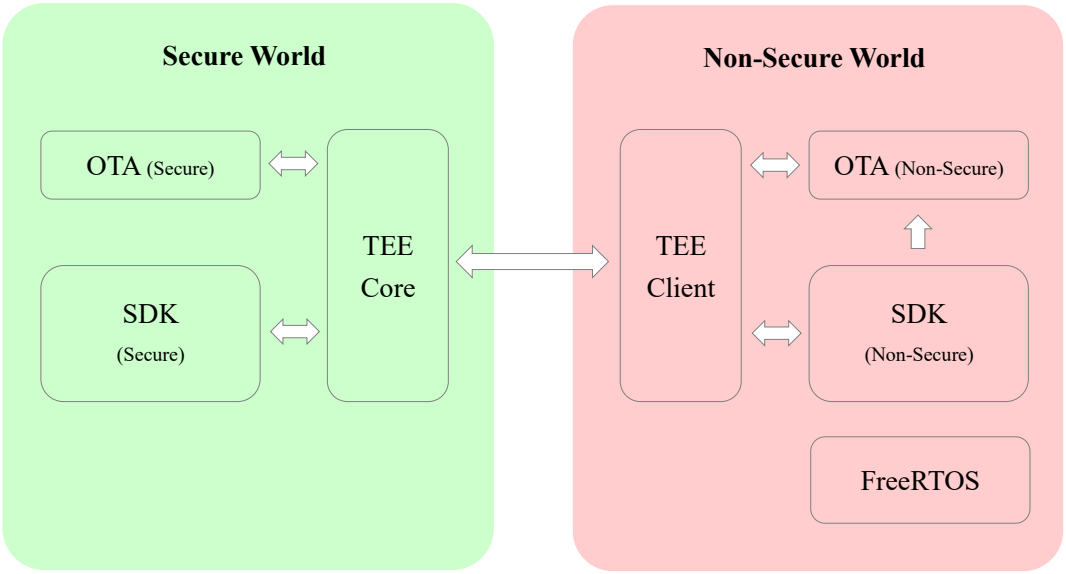
(M2351 平台)

版本	编写/修订说明	修订人	修订日期	备注
1.0.0	初始版本	刘晨	2019-08-19	无

1. 简介

本文档介绍了如何使用青莲云嵌入式 SDK 的 M2351 平台版本来构建基于 ARM TrustZone 技术的安全物联网终端设备。文档内容包括软硬件环境的准备、Demo 程序及 OTA 功能的使用说明。

青莲云嵌入式 SDK M2351 平台版本的总体应用框架如下：



大部分代码都在 Non-Secure (非安全) 区域执行，而涉及到密钥存储、数据加解密、Flash 写入等较为敏感的代码和数据则在 Secure (安全) 区域执行和存储。Secure 和 Non-Secure 两个区域之间的调用通过 TEE (可信执行环境) 接口完成。

在 Non-Secure 区域中，SDK 的使用和其他平台版本基本一致，具体请参考《青莲云嵌入式 SDK 开发使用文档》。使用上的差异主要有以下几点：

a) M2351 平台上产品密钥存储于 Secure 区域，数据的加密、解密也都在 Secure 区域进行，保证了密钥的安全性；
b) M2351 平台的 SDK 附带有 OTA 库，便于用户直接使用；
c) M2351 需通过外接 WiFi 模块接入网络，为此 SDK 增加了对 ESP8266 AT 固件模块的支持。

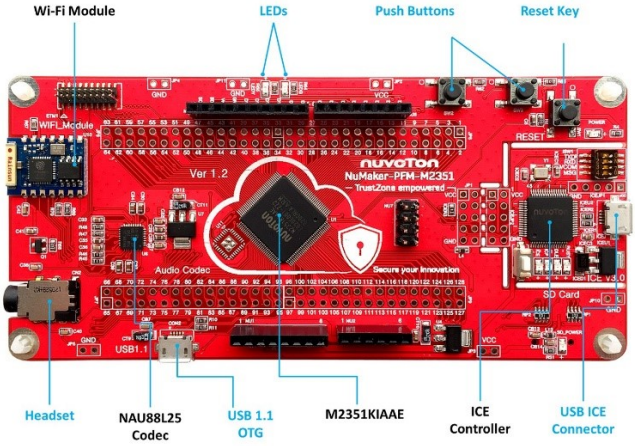
同时，当前版本的 SDK 在使用时有如下限制：

a) Flash 和 RAM 的 Secure 和 Non-Secure 区域划分不可更改；
b) CRPT 和 TRNG 外设必须保持配置为 Secure 的；
c) FreeRTOS 配置项中 configUSE_PREEMPTION 必须为 0，且 configTICK_RATE_HZ 必须为默认的 500。

2. 准备工作

2.1. 硬件

本 SDK 附带的 Demo 程序使用的硬件为新唐的 NuMaker-PFM-M2351 开发板，关于该开发板的更多信息请从新唐科技官网获取。



Demo 程序使用到了该开发板上的 ESP-03 WiFi 模块，及板载 Nu-Link-Me 的虚拟串口（调试信息从 UART0 输出，通过设置开发板上的拨码开关可以直接使用该虚拟串口与 UART0 通信）。

若使用其他硬件，则需要自行检查和修改 WiFi 模块、调试串口等配置。

2.2. 软件

2.2.1. M2351 系列 BSP

Demo 项目中含有复制自 M2351 系列 BSP 的必要 Library 文件，可独立编译，但其他准备工作仍需要 BSP 包。

请从新唐官网的“单片机 >Arm Cortex™-M23 单片机 >M2351 系列 > 软件”页面下载最新的 BSP 包（编写本文档时为 M2351_BSP_v3.00.003）。

2.2.2. Keil MDK 开发环境

软件开发环境当前仅支持 Keil MDK。软件的下载、安装和注册过程请参考 BSP 包 M2351_BSP_v3.00.003.zip 中的 NuMaker-PFM-M2351 Board Quick Start Guide.pdf 文档。

2.2.3. ESP8266 AT 固件

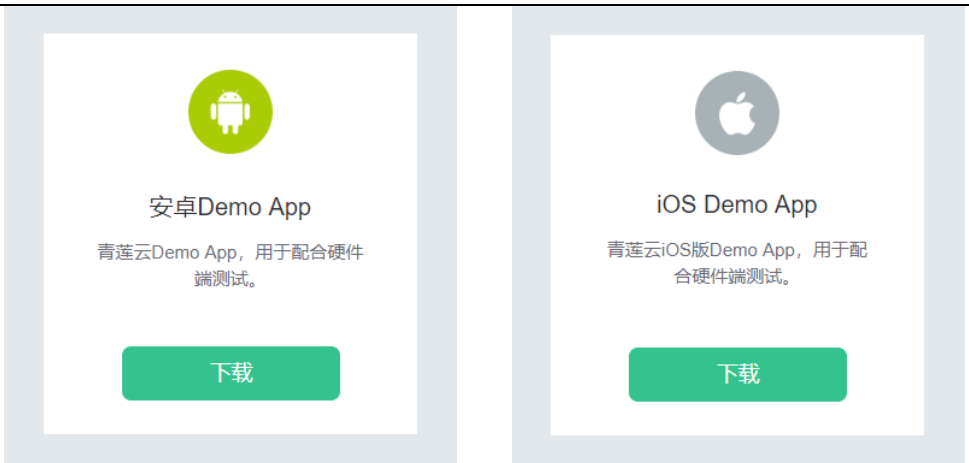
青莲云嵌入式 SDK 在 M2351 平台上通过与烧录有 AT 固件的 ESP8266 WiFi 模块进行 UART 通信来接入网络，所支持的最低 AT 固件版本为“ESP8266 AT Bin V1.6.2”。

Demo 程序默认使用 NuMaker-PFM-M2351 开发板上的 ESP-03 模块。关于如何为该模块下载和烧录 AT 固件的详细步骤，请参考“ESP8266 AT 固件烧录文档（M2351 平台）.pdf”文档。

2.2.4. 青莲云 Demo App

在成功下载 Demo 程序并运行后，需要通过青莲云 Demo App 来对设备进行网络配置和用户绑定。

该 App 可从 https://www.qinglianyun.com/Home/Doc/mobdownload 下载，提供 Android 和 iOS 两个版本。使用说明可参考《青莲云嵌入式 SDK Demo 使用文档》中的“功能调试”。



2.3. 青莲云账号

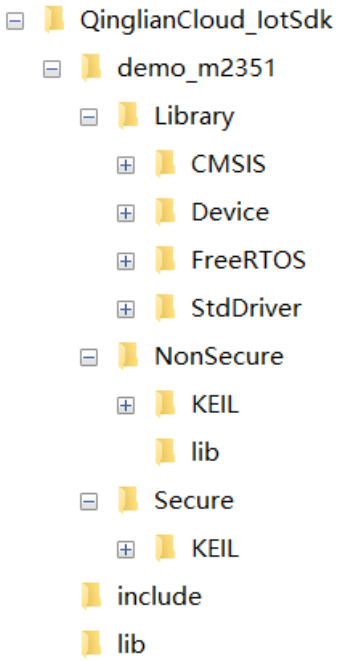
Demo 程序连接到云端时需要产品 ID 和产品 key。

需要在青莲云官网 https://www.qinglianyun.com/ 注册账号，并添加产品和功能点。具体步骤可参考《青莲云嵌入式 SDK Demo 使用文档》中的“产品开发”和“功能调试”。

3. Demo 程序内容

3.1. 目录结构

青莲云嵌入式 SDK M2351 平台版本的目录结构如下：



lib 和 include 目录下为 SDK 的静态库文件和对应的头文件，demo_m2351 下为示例工程。

lib 目录下的静态库文件如下表所示：

库文件	描述
iot_m2351_non_secure.lib	SDK (NonSecure, release 版)
iot_m2351_non_secure_debug.lib	SDK (NonSecure, debug 版)
iot_m2351_secure.lib	SDK (Secure)
ota_m2351_non_secure.lib	OTA (NonSecure)
ota_m2351_secure.lib	OTA (Secure)
tee_m2351_non_secure.lib	TEE (NonSecure)
tee_m2351_secure.lib	TEE (Secure)

include 目录下的头文件如下表所示：

头文件	库文件	描述
iot_base64.h	iot_m2351_non_secure.lib iot_m2351_non_secure_debug.lib	开放出来的 base64 接口
iot_cjson.h		开放出来的 cJSON 接口
iot_interface.h		NonSecure 区域回调函数和接口函数原型声明，具体请参看《青莲云嵌入式 SDK 开发使用文档》
iot_interface_secure.h	iot_m2351_secure.lib	Secure 区域结构体定义和接口函数原型声明
ota.h	ota_m2351_non_secure.lib ota_m2351_secure.lib	OTA 升级相关函数的原型声明
tee_client_api.h	tee_m2351_non_secure.lib	TEE Client API 头文件
tee_driver.h	tee_m2351_secure.lib	TEE Driver 头文件
tee_internal_api.h		TEE Internal Core API 头文件
wifi_interface.h	iot_m2351_non_secure.lib iot_m2351_non_secure_debug.lib	WiFi 相关回调函数和接口函数原型声明

demo_m2351 目录下的主要目录和文件如下表所示：

文件或目录	描述
Library	复制自 M2351 BSP 的 Demo 项目所需的 Library 文件
NonSecure\KEIL\NonSecure.uvprojx	NonSecure 项目文件
NonSecure\KEIL\nonsecure.sct	NonSecure 项目中链接器使用的分散文件
NonSecure\main.c	NonSecure 应用中 main()函数所在文件
Secure\KEIL\Secure.uvprojx	Secure 项目文件
Secure\KEIL\secure.sct	Secure 项目中链接器使用的分散文件
Secure\main.c	Secure 应用中 main()函数所在文件
Secure\partition_M2351.h	定义 Flash 和 RAM 空间划分和外设安全属性的头文件
Demo.uvmpw	Demo 程序的 Multi-Project Workspace 文件

3.2. Flash 和 RAM 空间划分

当前版本的 SDK 对 M2351 的 Flash 空间划分如下图所示，Secure 区域大小为 64KB, Non-Secure 区域大小为 512KB - 64KB = 448KB。

Non-Secure 区域中含有当前使用的非安全区固件（称为 active 固件）、新下载的非安全区固件或备份的升级前旧版本非安全区固件（称为 stored 固件）、固件信息以及本地存储这 4 个部分。

各部分的起始地址和大小如下表所示：

区域	起始地址 (Flash)	起始地址 (映射后)	大小
安全区固件	0x0000 0000	0x0000 0000	0x0001 0000 (64KB)
非安全区固件 (active)	0x0001 0000	0x1001 0000	0x0003 6800 (218KB)
非安全区固件 (stored)	0x0004 6800	0x1004 6800	0x0003 6800 (218KB)
固件信息	0x0007 D000	0x1007 D000	0x0000 2000 (8KB)
本地存储	0x0007 F000	0x1007 F000	0x0000 1000 (4KB)

RAM 空间划分如下图所示，Secure 区域大小为 24KB, Non-Secure 区域大小为 96KB - 24KB = 72KB。

安全区 RAM	非安全区 RAM

各部分的起始地址和大小如下表所示：

区域	起始地址 (RAM)	起始地址 (映射后)	大小
安全区 RAM	0x0000 0000	0x2000 0000	0x0000 6000 (24KB)
非安全区 RAM	0x0000 6000	0x3000 6000	0x0001 2000 (72KB)

在使用当前版本 SDK 时，这些区域的划分都应和 Demo 项目的默认配置一致。

4. Demo 程序使用

4.1. 修改配置

首先，在 Keil MDK 中打开 demo_m2351\Demo.uvmpw 多项目工作空间。

打开 demo_m2351\NonSecure\main.c 文件，将 iot_ctx 结构体中的产品 ID 修改为自己获取到的。

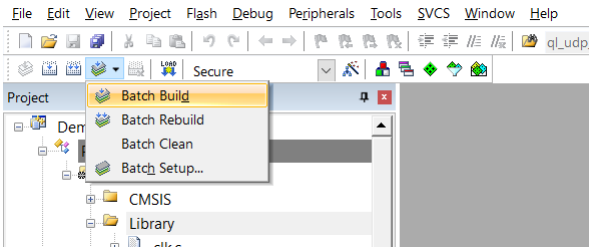
```
// 配置信息 (NonSecure)
static struct iot_context iot_ctx =
{
    // 平台生成，全网唯一产品 ID，请替换井号中间的 ID，并将 2 个井号删除
    // 注意：需与 Secure 项目中 iot_context_secure 结构体中的值一致
    #1002781074#,
    // 提示：M2351 平台上，产品密钥值在 Secure 项目中的 iot_context_secure 结构体中配置
    // MCU 固件版本，格式为 "xx.xx"，其中 x 为 0 - 9 的数字
    "01.01",
    // 数据接收缓冲区大小
    2048,
    // 数据发送缓冲区大小
    2048,
    // 云服务器地址
    "dispatch.qinglianyun.com",
    // 云服务器端口
    8990
};
```

再打开 demo_m2351\Secure\main.c 文件，将 iot_ctx_secure 结构体中的产品 ID 和产品 Key 都修改为自己获取到的。

```
// 配置信息 (Secure)
static struct iot_context_secure iot_ctx_secure =
{
    // 平台生成，全网唯一产品 ID，请替换井号中间的 ID，并将 2 个井号删除
    // 注意：需与 NonSecure 项目中 iot_context 结构体中的值一致
    #1002781074#,
    // 平台生成，全网唯一产品密钥，请替换井号中间的密钥，并将 2 个井号删除
    // 请将产品密钥字符串转换成相应的十六进制编码，即在每个字节前增加 0x 作为开头。如密钥是 5668，替换时应改为 0x56，0x68
    {#0x78, 0x79, 0x52, 0x72, 0x26, 0x4a, 0x35, 0x78, 0x23, 0x37, 0x24, 0xd1, 0x88, 0x68, 0x7d, 0x68#}
};
```

4.2. 编译

点击 Batch Build 按钮即可进行 Secure 和 NonSecure 项目的编译。



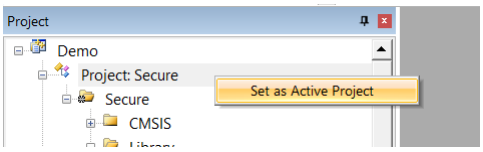
也可单独进行编译，但注意 NonSecure 项目要使用 Secure 项目编译产生的 nsclib_Secure.o 文件。若 Non-Secure callable 函数有变化，必须要先编译 Secure 项目。

4.3. 下载

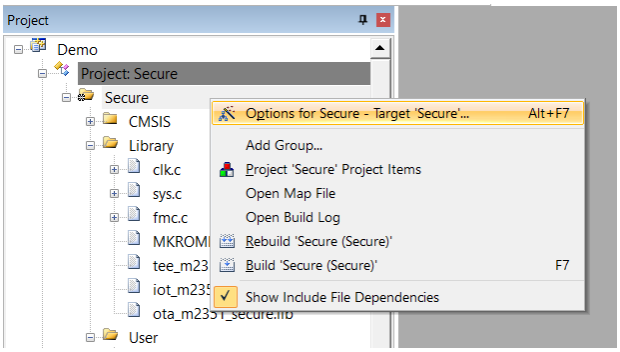
下载程序时，Secure 和 NonSecure 区域要单独下载。同时，如果 M2351 芯片的 Secure Setting 没有被设置过，还需要先设置 Secure Setting 中的 NonSecure 区 Base Address，否则默认值可能会与项目的 Secure, NonSecure 区域划分不同，导致引导到 NonSecure 区时失败。

4.3.1. 设置 Secure Setting

首先，右键点击“Project: Secure”，选择“Set as Active Project”，将 Secure 项目设为活动项目。

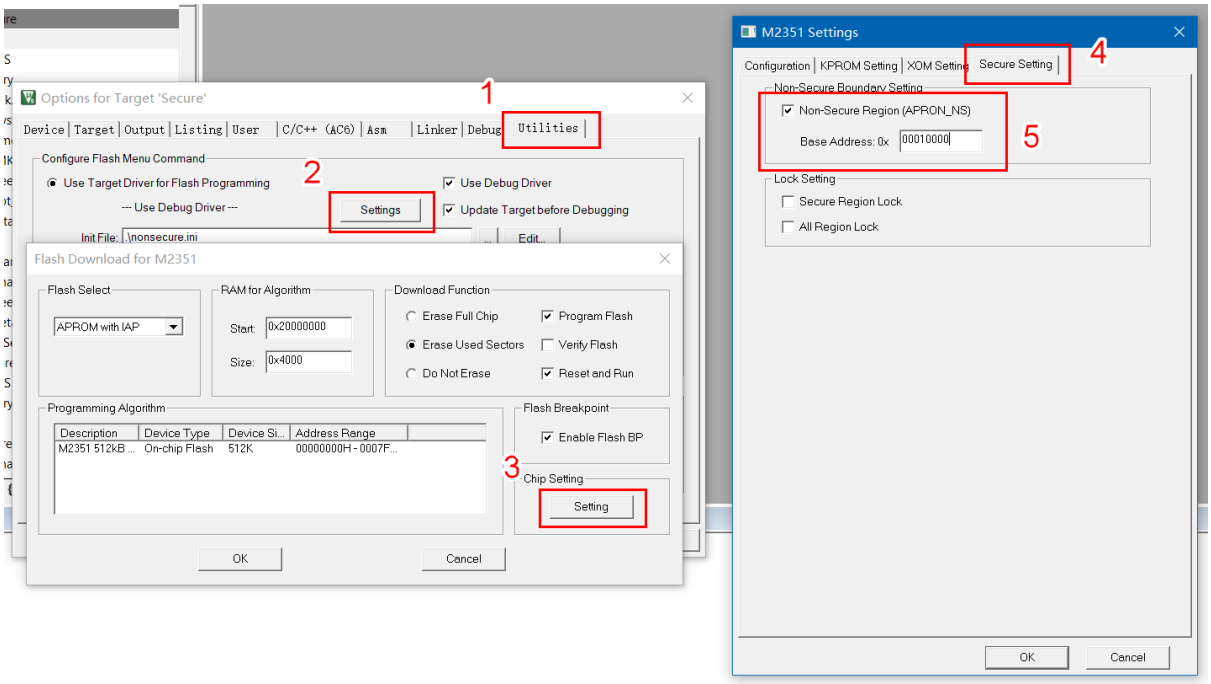


再右键点击“Project: Secure”下的“Secure”，选择“Options for Secure - Target 'Secure'...”。



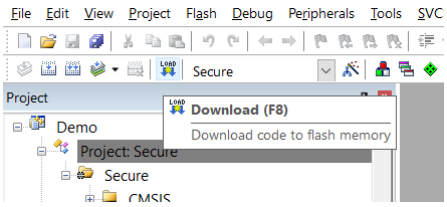
然后点击 Utilities 选项卡下的 Settings 按钮，进入 Flash 下载配置页面，点击 Chip Settings 内的 Settings 按钮，在弹出的 M2351 Settings 页面中选择 Secure Setting 选项卡，并勾选“Non-Secure Region (APROM_NS)”，将 Base Address 设为 0x00010000，点击“OK”。

此时可能会弹出“Are you sure you want to modify the chip settings?”和“It is necessary to erase whole chip before modifying Non-Secure Base Address. Continue?”的提示，均点击确定。此时 M2351 芯片的 Flash 被整体擦除，并且 Secure Setting 的值为刚才所配置的值。



4.3.2. 下载 Secure 区程序

将 Secure 项目设为活动项目后，点击工具栏的 Download 按钮下载 Secure 区程序。



在本 Demo 项目中，此步骤会将 Secure.bin 的 EXE_ROM (Secure 程序) 下载到 APROM 中偏移量 0x0，大小 0x9000 的区域；将 NSC_ROM (NonSecure callable 函数的入口函数) 下载到 APROM 中

偏移量 0x9000, 大小 0x1000 的区域。

4.3.3. 下载 NonSecure 区程序

同理，将 NonSecure 项目设为活动项目后，点击工具栏的 Download 按钮下载 NonSecure 区程序。

在本 Demo 项目中，此步骤会将 NonSecure.bin 的 EXE_ROM (NonSecure 程序) 下载到 APROM 中偏移量 0x10000, 大小 0x36800 的区域；将 FWINFO_ROM (全 0xFF) 下载到 APROM 中偏移量 0x7D000, 大小 0x02000 的区域。

FWINFO_ROM 的作用是清除 active firmware 的 OTA 固件信息页，以使复位后 Secure 的 OTA 程序为 active firmware 计算 checksum 并写入 OTA 固件信息页。

4.4. 运行

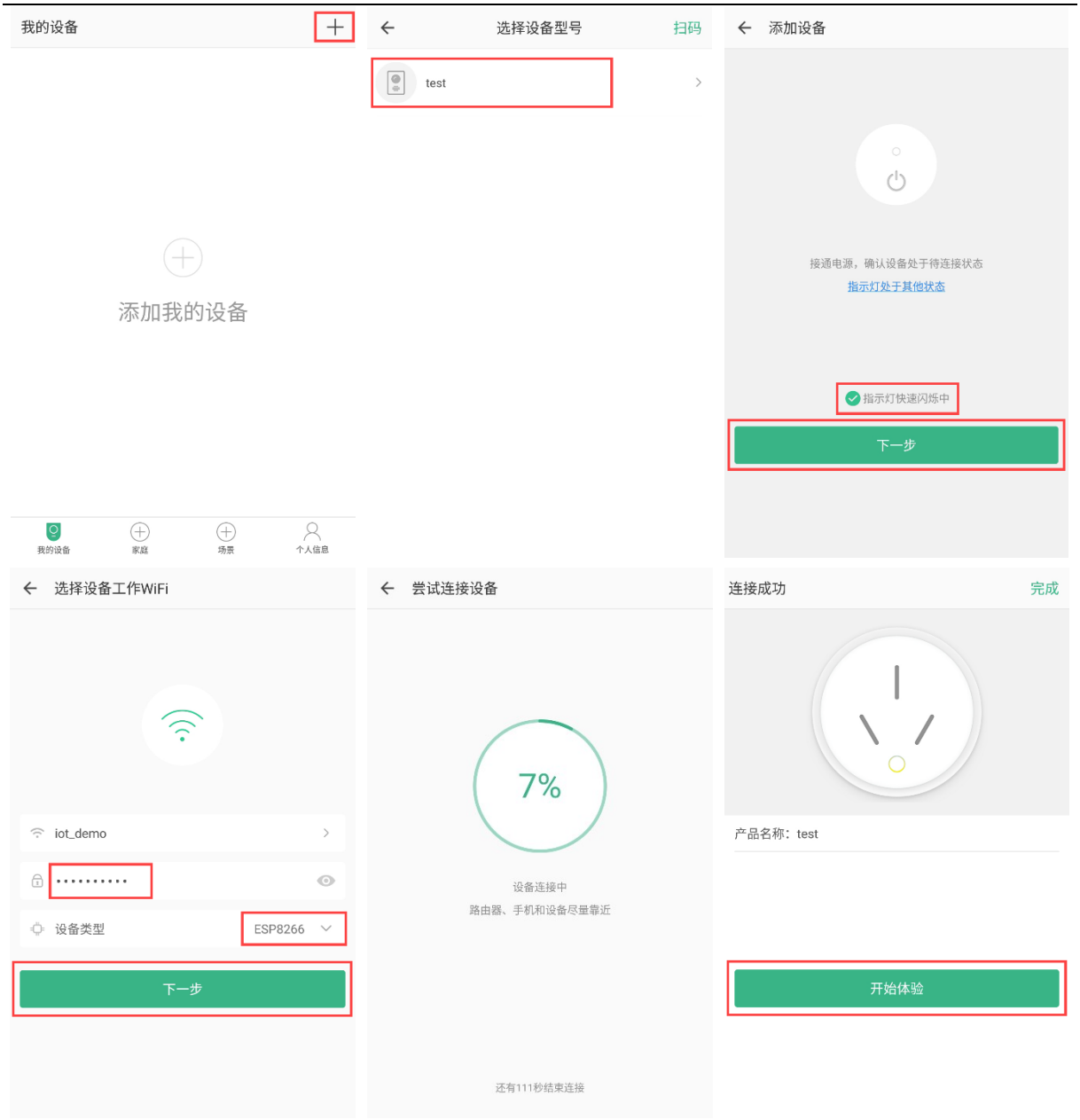
根据项目的默认配置，下载程序后会自动复位并运行。若未运行，则可手动按开发板上的 RESET 键进行复位。

设备上电复位后，调试 UART 口会输出类似如下的信息：

```
[OTA] [active fw] magicNumber: 0x57464c51, version: 0x00000001, checksum: 0x24795b70
[OTA] [stored fw] magicNumber: 0xffffffff, version: 0xffffffff, checksum: 0xffffffff
[OTA] [active fw] calculating the actual checksum of active fw...
[OTA] [active fw] calculated checksum: 0x24795b70
[OTA] [stored fw] calculating the actual checksum of stored fw...
[OTA] [stored fw] calculated checksum: 0x30f9a5ed
[OTA] no new firmware exist
[OTA]
[OTA] boot to active firmware @ 0x10010000 (OTA_ACTIVE_FIRMWARE_OFFSET = 0x10000)
Execute non-secure code ...
Entered NonSecure main()
WiFi module initializing...
WiFi module power on...
create FreeRTOS main task...
start FreeRTOS scheduler...
reset WiFi module...
```

4.5. 设备配网和用户绑定

设备开始运行后，在手机上打开“青联智能”Demo App (这里以 Android 版本为例)，点击右上角的“＋”号添加设备。按照如下截图步骤进行，即可通过 smartconfig 为设备配置好 WiFi 接入点的信息，并将设备绑定到登录用户上。



在配网过程中，调试 UART 口会输出类似如下的信息：

```
reset WiFi module...
wait for auto connecting to AP...
4 seconds to wait...
3 seconds to wait...
2 seconds to wait...
1 seconds to wait...
not connect to AP
start smartconfig...
[smartconfig] started
[smartconfig] in progress
[smartconfig] got wifi info, start to connect
[smartconfig] connected to AP
[smartconfig] stopped
smartconfig succeeded
connected to ap
iot_start()
```

```
... (省略部分输出)
not connected cloud!
device state:0, ts:1565594190
permit_bind
iot_data_cb :3.
iot_data_cb :4.
iot_data_cb :5.
loop begin
... (省略部分输出)
iot_data_cb :7.
iot_data_cb :8.
device state:5, ts:1565594191

bind user cb!
iot_data_cb :11.
iot_data_cb :12.
```

5. OTA 功能使用

5.1. 生成 OTA 升级固件

首先将 NonSecure 项目设为活动项目，打开 demo_m2351\NonSecure\main.c 文件，将 iot_ctx 结构体中的“MCU 固件版本”由默认的“01.01”修改为一个更高的版本号，如“01.02”。

再将 main()函数中 printf("It is new firmware!!!\r\n"); 一行取消注释，以便于检查 OTA 升级是否成功。

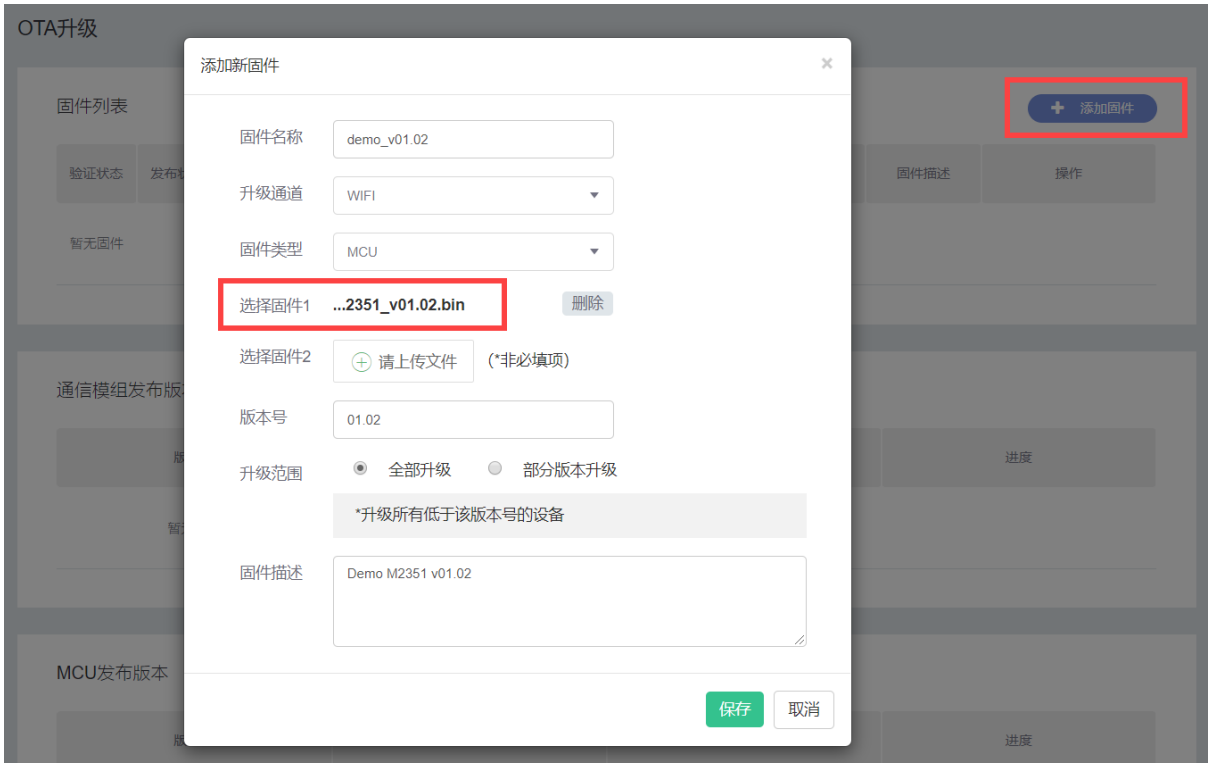
```
int main(void)
{
    printf("Entered NonSecure main()\r\n");

    printf("It is new firmware!!!\r\n");

    // 配置 GPIO 的模式
    GPIO_MODE_INIT();
```

此时点击 Build 构建 NonSecure 项目，完成后 demo_m2351\NonSecure\KEIL\bin\NonSecure.bin 目录中的 EXE_ROM 文件即为 OTA 升级所用固件。为便于识别将该文件复制一份并更名为 demo_m2351_v01.02.bin。

FWINFO_ROM 文件为使用 Keil 下载 NonSecure 程序时擦除 8KB 固件信息区域数据所用，OTA 升级不需要该文件。

5.2. 上传固件

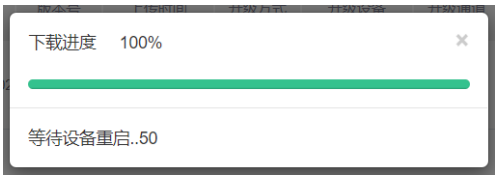
在后台管理的“云服务 - OTA 升级”页面，点击“添加固件”按钮，上传刚刚生成的 demo_m2351_v01.02.bin 文件，并填写固件版本号等其他必要信息，最后点击“保存”。

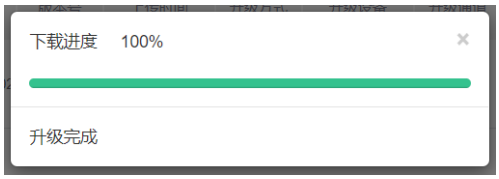


5.3. 测试 OTA 升级

在设备正常运行状态下，点击“验证固件”按钮，并输入设备的 MAC 地址，点击“立即验证”，即可开始验证所上传的固件是否能成功完成 OTA 升级。

验证过程中页面会显示实时的下载进度，固件全部下载完成后会等待设备重启。





整个 OTA 升级过程中，调试 UART 口输出的信息类似如下：

```
iot_data_cb :18.

info_type:3
owner:1
flen:118400
ver:01.02
get chunk offset = 0, size = 2048
[OTA] erase stored fw info page...
```

```
[OTA] erase stored fw, OTA_STORED_FIRMWARE_OFFSET = 0x46800
[OTA] write firmware data, offset = 0x0 -> 0x46800, length = 2048
get chunk offset = 2048, size = 2048
[OTA] write firmware data, offset = 0x800 -> 0x47000, length = 2048
iot_data_cb :21.
get chunk offset = 4096, size = 2048
[OTA] write firmware data, offset = 0x1000 -> 0x47800, length = 2048
... (省略部分输出)
get chunk offset = 112640, size = 2048
[OTA] write firmware data, offset = 0x1b800 -> 0x62000, length = 2048
get chunk offset = 114688, size = 2048
[OTA] write firmware data, offset = 0x1c000 -> 0x62800, length = 2048
get chunk offset = 116736, size = 1664
[OTA] write firmware data, offset = 0x1c800 -> 0x63000, length = 1664
get ota chunks over, wait to reboot...
[OTA] current active fw version: 0x00000001
[OTA] new fw version: 0x00000002
[OTA] new fw checksum: 0x8cbe7abf
[OTA] write new firmware page checksums...
get ota cmd success,reboot right now
[OTA] reboot from APROM_BASE @ 0x0
[OTA] [active fw] magicNumber: 0x57464c51, version: 0x00000001, checksum: 0x30f9a5ed
[OTA] [stored fw] magicNumber: 0x57464c51, version: 0x00000002, checksum: 0x8cbe7abf
[OTA] [active fw] calculating the actual checksum of active fw...
[OTA] [active fw] calculated checksum: 0x30f9a5ed
[OTA] [stored fw] calculating the actual checksum of stored fw...
[OTA] [stored fw] calculated checksum: 0x8cbe7abf
[OTA] stored fw is new, swap active and stored fw...
[OTA] swapping page 0 of 109...
[OTA] swapping page 1 of 109...
[OTA] the pages to swap are the same in contents, skipped swapping
[OTA] swapping page 2 of 109...
[OTA] the pages to swap are the same in contents, skipped swapping
[OTA] swapping page 3 of 109...
... (省略部分输出)
[OTA] swapping page 107 of 109...
[OTA] the pages to swap are the same in contents, skipped swapping
[OTA] swapping page 108 of 109...
[OTA] the pages to swap are the same in contents, skipped swapping
[OTA] erase page @ 0x0007e000
[OTA] swapping done
[OTA] reboot from APROM_BASE @ 0x0
[OTA] [active fw] magicNumber: 0x57464c51, version: 0x00000001, checksum: 0x30f9a5ed
[OTA] [stored fw] magicNumber: 0x57464c51, version: 0x00000002, checksum: 0x8cbe7abf
[OTA] [active fw] calculating the actual checksum of active fw...
[OTA] [active fw] calculated checksum: 0x8cbe7abf
[OTA] [stored fw] calculating the actual checksum of stored fw...
[OTA] [stored fw] calculated checksum: 0x30f9a5ed
[OTA] swapping active and stored firmware info pages...
[OTA] boot to active firmware @ 0x10010000 (OTA_ACTIVE_FIRMWARE_OFFSET = 0x10000)
Execute non-secure code ...
Entered NonSecure main()
It is new firmware!!!
WiFi module initializing...
```

可以看到最后一次引导后输出了“It is new firmware!!!”，即运行的是新固件。

6. SDK 使用

关于青莲云嵌入式 SDK 的数据传输、本地存储、子设备和系统函数调用等功能的使用，请参考各平台通用的《青莲云嵌入式 SDK 开发使用文档》。