APPLICATION NOTE

OTA Firmware Update Example

For Ambiq Apollo3 and Apollo4 SoC Family

A-SOCMSC-ANGA01EN v1.0

Legal Information and Disclaimers

AMBIQ MICRO INTENDS FOR THE CONTENT CONTAINED IN THE DOCUMENT TO BE ACCURATE AND RELIABLE. THIS CONTENT MAY, HOWEVER, CONTAIN TECHNICAL INACCURACIES, TYPOGRAPHICAL ERRORS OR OTHER MISTAKES. AMBIQ MICRO MAY MAKE CORRECTIONS OR OTHER CHANGES TO THIS CONTENT AT ANY TIME. AMBIQ MICRO AND ITS SUPPLIERS RESERVE THE RIGHT TO MAKE CORRECTIONS, MODIFICATIONS, ENHANCEMENTS, IMPROVEMENTS AND OTHER CHANGES TO ITS PRODUCTS, PROGRAMS AND SERVICES AT ANY TIME OR TO DISCONTINUE ANY PRODUCTS, PROGRAMS, OR SERVICES WITHOUT NOTICE.

THE CONTENT IN THIS DOCUMENT IS PROVIDED "AS IS". AMBIQ MICRO AND ITS RESPECTIVE SUPPLIERS MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY OF THIS CONTENT FOR ANY PURPOSE AND DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS CONTENT, INCLUDING BUT NOT LIMITED TO, ALL IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHT.

AMBIQ MICRO DOES NOT WARRANT OR REPRESENT THAT ANY LICENSE, EITHER EXPRESS OR IMPLIED, IS GRANTED UNDER ANY PATENT RIGHT, COPYRIGHT, MASK WORK RIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT OF AMBIQ MICRO COVERING OR RELATING TO THIS CONTENT OR ANY COMBINATION, MACHINE, OR PROCESS TO WHICH THIS CONTENT RELATE OR WITH WHICH THIS CONTENT MAY BE USED.

USE OF THE INFORMATION IN THIS DOCUMENT MAY REQUIRE A LICENSE FROM A THIRD PARTY UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF THAT THIRD PARTY, OR A LICENSE FROM AMBIQ MICRO UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF AMBIQ MICRO.

INFORMATION IN THIS DOCUMENT IS PROVIDED SOLELY TO ENABLE SYSTEM AND SOFTWARE IMPLEMENTERS TO USE AMBIQ MICRO PRODUCTS. THERE ARE NO EXPRESS OR IMPLIED COPYRIGHT LICENSES GRANTED HEREUNDER TO DESIGN OR FABRICATE ANY INTEGRATED CIRCUITS OR INTEGRATED CIRCUITS BASED ON THE INFORMATION IN THIS DOCUMENT. AMBIQ MICRO RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN. AMBIQ MICRO MAKES NO WARRANTY, REPRESENTATION OR GUARANTEE REGARDING THE SUITABILITY OF ITS PRODUCTS FOR ANY PARTICULAR PURPOSE, NOR DOES AMBIQ MICRO ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT, AND SPECIFICALLY DISCLAIMS ANY AND ALL LIABILITY, INCLUDING WITHOUT LIMITATION CONSEQUENTIAL OR INCIDENTAL DAMAGES. “TYPICAL” PARAMETERS WHICH MAY BE PROVIDED IN AMBIQ MICRO DATA SHEETS AND/OR SPECIFICATIONS CAN AND DO VARY IN DIFFERENT APPLICATIONS AND ACTUAL PERFORMANCE MAY VARY OVER TIME. ALL OPERATING PARAMETERS, INCLUDING “TYPICALS” MUST BE VALIDATED FOR EACH CUSTOMER APPLICATION BY CUSTOMER’S TECHNICAL EXPERTS. AMBIQ MICRO DOES NOT CONVEY ANY LICENSE UNDER NEITHER ITS PATENT RIGHTS NOR THE RIGHTS OF OTHERS. AMBIQ MICRO PRODUCTS ARE NOT DESIGNED, INTENDED, OR AUTHORIZED FOR USE AS COMPONENTS IN SYSTEMS INTENDED FOR SURGICAL IMPLANT INTO THE BODY, OR OTHER APPLICATIONS INTENDED TO SUPPORT OR SUSTAIN LIFE, OR FOR ANY OTHER APPLICATION IN WHICH THE FAILURE OF THE AMBIQ MICRO PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR. SHOULD BUYER PURCHASE OR USE AMBIQ MICRO PRODUCTS FOR ANY SUCH UNINTENDED OR UNAUTHORIZED APPLICATION, BUYER SHALL INDEMNIFY AND HOLD AMBIQ MICRO AND ITS OFFICERS, EMPLOYEES, SUBSIDIARIES, AFFILIATES, AND DISTRIBUTORS HARMLESS AGAINST ALL CLAIMS, COSTS, DAMAGES, AND EXPENSES, AND REASONABLE ATTORNEY FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PERSONAL INJURY OR DEATH ASSOCIATED WITH SUCH UNINTENDED OR UNAUTHORIZED USE, EVEN IF SUCH CLAIM ALLEGES THAT AMBIQ MICRO WAS NEGLIGENT REGARDING THE DESIGN OR MANUFACTURE OF THE PART.

Revision History

Revision	Date	Description
1.0	May 8, 2023	Initial release

Reference Documents

Document ID	Description

Table of Contents

SECTION

1

Introduction

This document describes the Over-the-Air (OTA) firmware update example using Bluetooth® Low Energy 5.0 connectivity for Apollo3 and Apollo4 families of system on chips (SoCs). The example project consists of the following components:

- An application running on the Apollo3/Apollo4 series SoCs which includes the following components:
  - AMOTA application (ble_freertos_amota)
  - Bootloader
  - OTA Bluetooth Low Energy service (AMOTA)
  - Bluetooth Low Energy stack (Arm® Cordio BLE stack)
- Firmware running on the Bluetooth Low Energy HCI controller module
- IOS or Android smartphone application (OTA Demo)
- Makefile to generate binary files for the smartphone app to load

The purpose of the example is to provide a reference for firmware update of the SoC and its HCI controller over Bluetooth Low Energy communication while the application is still running. Data transfer is a background operation of the application and can be paused and resumed during the update process. The data being transferred is verified with each communication package as well as a whole image when the transfer completes. The received data can be stored either inside the empty area of the internal flash (if there is enough space left in the internal flash of the SoC) or in external flash. After the entire image is received and stored correctly, the system keeps operating from the existing firmware until a system reset is triggered. The new image gets loaded into internal flash by the bootloader and, after a system reset, starts execution automatically if found to be available.

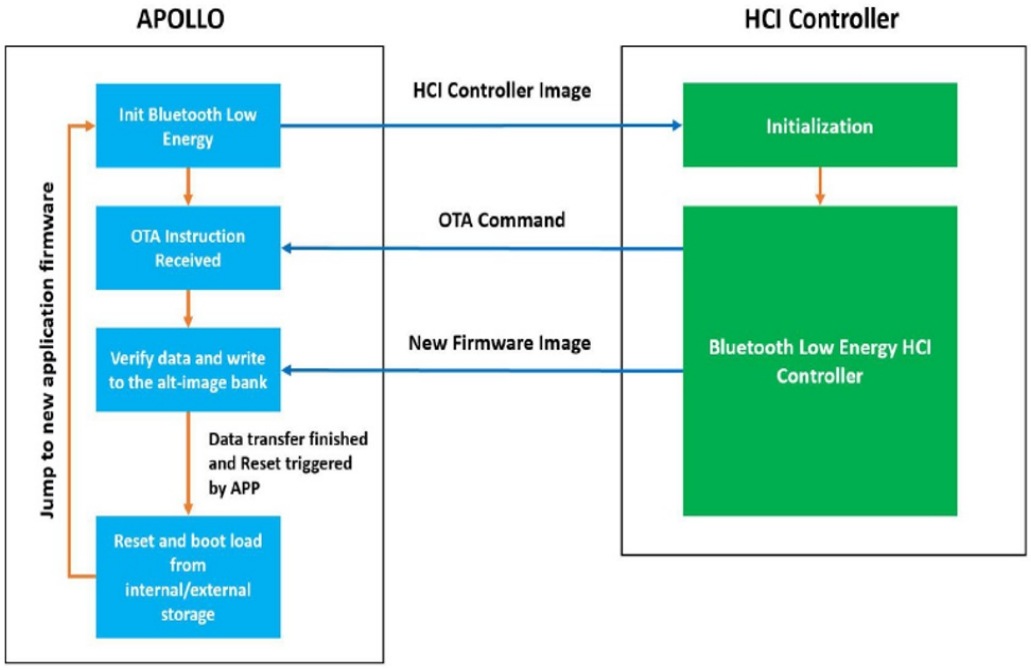
SECTION

2

OTA System Architecture and Operation Flow

A high-level OTA update flow diagram is shown below.

Figure 2-1: OTA Operation Flow



The Bluetooth Low Energy stack used in this example is the Arm Cordio Bluetooth Low Energy stack. The upper layers of the stack, including the HCI host layer, are running on an Apollo3 or Apollo4 SoC as part of the application firmware where the Bluetooth Low Energy module operates as a standard HCI controller. This architecture utilizes the ultra-low operating power features of the Apollo3/Apollo4 SoC. These features enable the power required for the Bluetooth Low Energy communication to be as low as possible, further reducing overall system power consumption.

At application boot up, the lower-level software driver sends the standard HCI controller image to the Bluetooth Low Energy Module. Once the Bluetooth Low Energy controller and the stack are initialized, Bluetooth Low Energy communication begins.

2.1 AMOTA Service

The following section describes the AMOTA service that is implemented in the **AmbiqSuite ble_freertos_amota** example project to perform the key function of the OTA process.

2.1.1 Service Declaration

The service UUID of Ambiq Micro OTA (AMOTA) service is defined as below:

00002760-08C2-11E1-9073-0E8AC72E1001

NOTE: The Base UUID of Bluetooth SIG is 00000000-0000-1000-8000-00805F9B34FB. All customized 128-bit UUID should be outside of the Bluetooth UUID Base range.

2.1.2 Service Characteristic Definitions

Rx: 00002760-08C2-11E1-9073-0E8AC72E0001
Tx: 00002760-08C2-11E1-9073-0E8AC72E0002

Table 2-1: Service Characteristic Definitions

Characteristic	Requirements	Mandatory Properties	Security Permissions	Description
Characteristic Rx	M	Write	None	Data from Client
Characteristic Rx User Description	N	Read	None	Value read by client
Characteristic Tx	M	Notify	None	Value notification to client
Characteristic Tx Client Characteristic Configuration descriptor	M	Read	None	Value notification configuration

2.1.3 Characteristics

The following characteristics are defined in the AM OTA Service. Only one instance of each characteristic is permitted within this service.

Table 2-2: Characteristic Definition

Characteristic Name	Mandatory Properties	Security Permission
Received Data Characteristic	Write Command	None
Send Data Characteristic	Notify	None

Characteristic Descriptors:

Characteristic User Description:

This characteristic descriptor defines the AM OTA version with read permission property.

Client Characteristic Configuration Descriptor:

The notification characteristic will start to notify if the value of the CCCD (Client Characteristic Configuration Descriptor) is set to 0x0001 by client. The send data characteristic will stop notifying if the value of the CCCD is set to 0x0000 by client.

2.1.4 Service Behaviors

The following are service behaviors:

1. OTA client sends firmware header and meta information to the server with the AMOTA packet format.
2. Server replies with received byte counters.
3. OTA client starts to send firmware data with AMOTA packet format.
4. Server replies with received byte counters.
5. OTA client sends the verify command to ask the server to calculate the whole firmware checksum.
6. Server replies with the checksum result to the client.
7. Client sends the reset command to the server (app behavior)
8. Server sends the reset command response to the server before reset (app behavior)

2.2 AMOTA Packet Format

Length: two bytes (data + checksum)

Cmd: 1 byte
Data: 0 ~ 512 bytes
Checksum: 4 bytes

Table 2-3: AMOTA Packet Format

Length	Command	Data	Checksum
Two bytes	1 byte	0-512 bytes	4 bytes

Commands:

```
/* amota commands */
typedef enum
{
    AMOTA_CMD_UNKNOWN,
    AMOTA_CMD_FW_HEADER,
    AMOTA_CMD_FW_DATA,
    AMOTA_CMD_FW_VERIFY,
    AMOTA_CMD_FW_RESET,
    AMOTA_CMD_MAX
}eAmotaCommand;
```

Firmware Header Info:

```
Amota packet header (two bytes length + 1 byte cmd)
amotaHeaderInfo_t
encrypted: 4 bytes
fwStartAddr: 4 bytes
fwLength: 4 bytes
fwCrc: 4 bytes
secInfoLen: 4 bytes
resvd1: 4 bytes
resvd2: 4 bytes
resvd3: 4 bytes
version: 4 bytes
fwDataType: 4 bytes
storageType: 4 bytes
resvd4: 4 bytes
Amota packet checksum (4 bytes)
```

Firmware Data Packet:

```
Amota packet header (two bytes length + 1 byte cmd)
Data: 0 ~ 512 bytes
Amota packet checksum (4 bytes)
```

Firmware Verify Command:

```
Amota packet header (two bytes length + 1 byte cmd)
Amota packet checksum (4 bytes)
```

Target Reset Command:

```
Amota packet header (two bytes length + 1 byte cmd)
Amota packet checksum (4 bytes)
```

Command Response Format:

```
Length: 2 bytes (data + status)
Cmd: 1 byte
Status: 1 byte
Data: 0 ~ 16 bytes
/* amota status */
typedef enum
{
    AMOTA_STATUS_SUCCESS,
    AMOTA_STATUS_CRC_ERROR,
    AMOTA_STATUS_INVALID_HEADER_INFO,
    AMOTA_STATUS_INVALID_PKT_LENGTH,
    AMOTA_STATUS_INSUFFICIENT_BUFFER,
    AMOTA_STATUS_INSUFFICIENT_FLASH,
    AMOTA_STATUS_UNKNOWN_ERROR,
    AMOTA_STATUS_FLASH_WRITE_ERROR,
    AMOTA_STATUS_MAX
 }eAmotaStatus;
```

Firmware Header Info Response and Firmware Data Response:

```
Amota packet header (two bytes length + 1 byte cmd)
Status: 1 byte
Received packet counter: 4 bytes
```

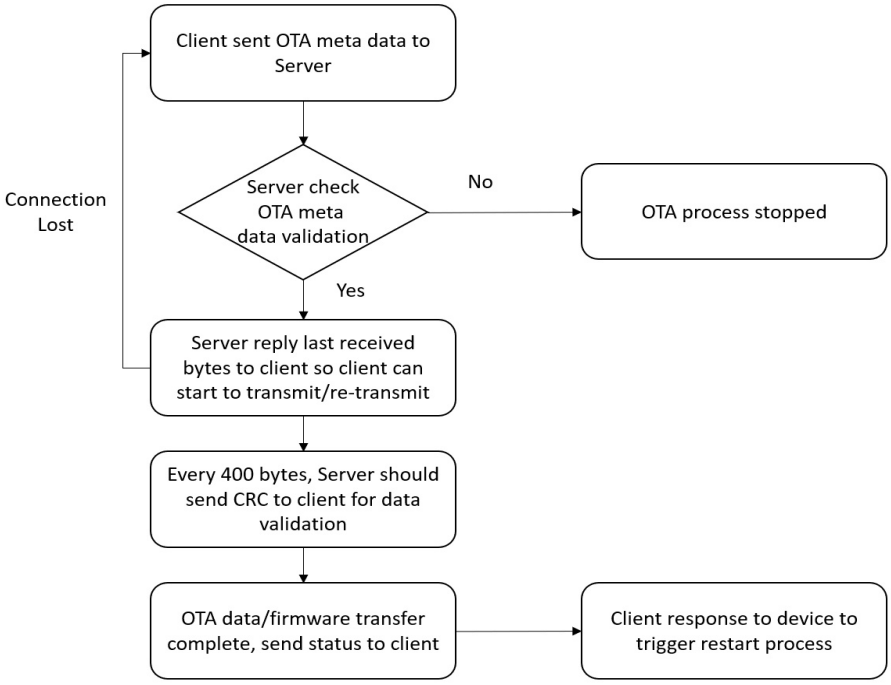
Firmware Verify and Target Reset Response:

```
Amota packet header (two bytes length + 1 byte cmd)
Status: 1 byte
```

Firmware Verify and Target Reset Response:

```
Amota packet header (two bytes length + 1 byte cmd)
Status: 1 byte
```

Figure 2-2: OTA Service Flow



- Create services
  vc_amotas.c
  svc_amotas.h
- Profile and OTA logic implementation
  amotas_main.c
  amotas_api.h
- Initialize in application

  1. Set **`AmotasCfg_t`**.
  2. Add **`AMOTAS_TX_CH_CCC_HDL`** in **`attsCccSet_t`**.
  3. Register callback in **`AmotaStart()`**:
     `SvcAmotasCbackRegister(NULL, amotas_write_cback);`
  4. Add service in **`AmotaStart ()`**:
     `SvcAmotasAddGroup();`
  5. Call **`amotas_proc_msg()`** in **`amotaProcMsg()`** for event **`DM_CONN_OPEN_IND`**
  6. Add **`amotas_start()`** and **`amotas_stop()`** in function **`amotaProcCccState()`**.

SECTION

3

Running OTA Update Example on Apollo3/Apollo4 Family SoC

The ble_freertos_amota example implements an over-the-air (OTA) firmware update for an Ambiq slave. This example is designed to allow loading of a binary software update from either an iOS or Android phone running the Ambiq OTA application.

3.1 Development Environment:

Hardware:

This example runs on Apollo3 and Apollo4 Blue series EVBs, make sure to have one available to run the example.

For details of the EVBs, visit www.ambiq.com.

Software:

- Install the latest AmbiqSuite.
- Install Python 3.x to run the helper scripts for OTA binary file generation and combination.
- Install Keil MDK-ARM Plus Version 5.20 or later/IAR/GCC tools based on the support in the SDK for code generation and debug.

iOS:

- Install iTunes PC tool for iOS device APP installation and file sharing.
- Visit our APP page on Apple AppStore at:
  https://itunes.apple.com/us/app/ambiq-ota/id1190453962?mt=8
  Or simply search for **Ambiq OTA** in the AppStore to install.

Android:

- Install our Ambiq OTA app directly from the APK located at:
  ..\AmbiqSuite\tools\amota

3.2 Run the Example

1. Modify the makefile under the AmbiqSuite 4.x.x or AmbiqSuite 3.x.x from the following directory:

   AmbiqSuite_R4.x.x\tools\apollo4_amota\scripts\Makefile

   OR

   AmbiqSuite_R3.x.x\tools\apollo3_amota\scripts\Makefile

2. Select the required board in Makefile.
   Example: Apollo4_BOARD = apollo4b_blue_evb

> **NOTE:** Can be other board for Apollo3/4 series. If AMOTA is used to load an application which does not itself support AMOTA, then the target MCU will stop supporting AMOTA once the new application is loaded. Therefore, normally AMOTA is only used to load binaries which include AMOTA support. However, for simple test purposes you could use AMOTA to load a simple project such as hello_world.

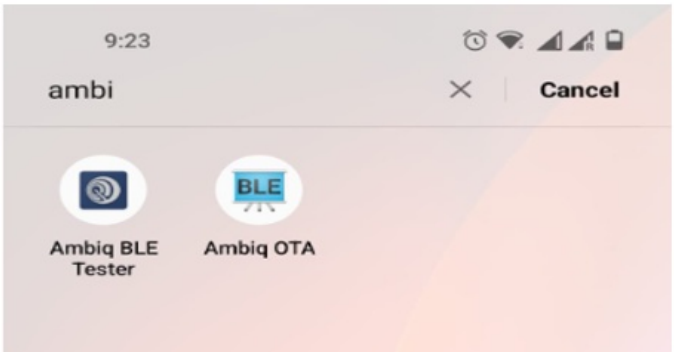
3. Open **msys/cygwin**, and run **make** from the folder below:
   AmbiqSuite_R4.x.x/tools/apollo4_amota/scripts

```
$ make
make -C ../../../boards/apollo4b_blue_evb/examples/ble/ble_freertos_amota/gcc/
make[1]: Entering directory '/cygdrive/c/New_Hire/projects/AmbiqSuite_R4.3.0/boards/apollo4b_blue_evb/e
mples/ble/ble_freertos_amota/gcc'
 Compiling gcc ../../../../../../third_party/cordio/ble-host/sources/sec/common/sec_aes.c
 Compiling gcc ../../../../../../third_party/cordio/ble-host/sources/sec/common/sec_aes_rev.c
 Compiling gcc ../../../../../../third_party/cordio/ble-host/sources/sec/common/sec_ccm_hci.c
 Compiling gcc ../../../../../../third_party/cordio/ble-host/sources/sec/common/sec_cmac_hci.c
```

4. After running successfully, the following is displayed and generated in the **AmbiqSuite_R4.x.x\tools\amota\scripts** directory **starter_binary_apollo4_blue.bin** and **update_binary_apollo4_blue.bin** files.

```
# Apollo4 Cordio
cp ../../../boards/apollo4b_blue_evb/examples/ble/ble_freertos_amota/gcc/bin/ble_freertos_amota.bin start
er_binary_apollo4_blue.bin
cp -r ../../apollo4b_scripts/oem_tools_pkg* oem_tools_pkg
cp -r ../../apollo4b_scripts/arm_utils* arm_utils
cp ../../apollo4b_scripts/am_defines.py am_defines.py
cp ../../apollo4b_scripts/apollo4b_keys.py apollo4b_keys.py
cp ../../apollo4b_scripts/create_cust_image_blob.py create_cust_image_blob.py
cp ../../apollo4b_scripts/key_table.py key_table.py
cp ../../apollo4b_scripts/sample/keys.ini keys.ini
python create_cust_image_blob.py -c firmware.ini
Using settings from firmware.ini.
python ota_binary_converter.py --appbin temp_binary_apollo4_blue.bin -o update_binary_apollo4_blue
hdr_length  48
load_address  0x4000 ( 0x4000 )
app_size  0x1a1f4 ( 106996 )
app_crc =   0xc9cb5366
Security Info Length 0x0
blob crc =   0xc9cb5366
app_ver 0 ( 0x0 )
bin_type 0 ( 0x0 )
str_type 0 ( 0x0 )
```

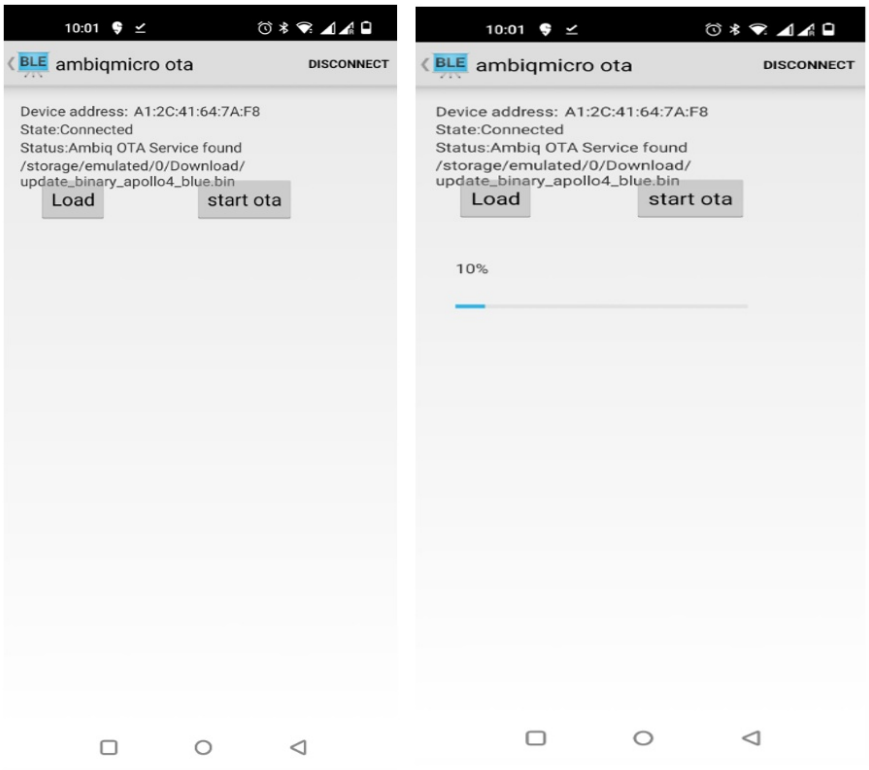
5. Install **ambiq_ota** app on the Android/iOS mobile device from the following path of SDK: **AmbiqSuite_R4.x.x\tools\apollo3_amota** or **Ambiq-Suite_R3.x.x\tools\apollo4_amota**



6. Copy the **update_binary_apollo4.bin** to a directory on the mobile device. Power up the evaluation board with the **ble_freertos_amota** example project and open **Ambiq OTA**. It should recognize the **ambiqmicro ota** device.

7. Select **Load** and find the **update_binary_apollo4.bin** file.

8. Click on the file **update_binary_apollo4.bin**, click on **start ota**, the system will reboot automatically after the upgrade is successful.



6500 River Place Boulevard, Building 7, Suite 200, Austin, TX 78730
www.ambiq.com
sales@ambiq.com
+1 (512) 879-2850

A-SOCMSC-ANGA01EN v1.0
May 2023