

# Developer Guide

## Security Starter Kit with STM32WB55 and OPTIGA™ Trust M

Date: December 3, 2020 | Version 1.0

FINAL

---



The Solutions People



### Confidentiality Notice

Copyright (c) 2020 eInfochips. - All rights reserved

This document is authored by eInfochips and is eInfochips intellectual property, including the copyrights in all countries in the world. This document is provided under a license to use only with all other rights, including ownership rights, being retained by eInfochips. This file may not be distributed, copied, or reproduced in any manner, electronic or otherwise, without the express written consent of eInfochips.

## CONTENTS

<b>1</b>	<b>INTRODUCTION.....</b>	<b>4</b>
1.1	Purpose of the Document .....	4
1.2	Architecture .....	4
1.3	SSK Suit Package Contents.....	4
<b>2</b>	<b>SETTING UP THE DEVELOPMENT ENVIRONMENT .....</b>	<b>5</b>
2.1	Hardware setup .....	5
2.2	Software Setup .....	6
2.2.1	STM32CubeIDE .....	6
2.2.2	STM32CubeProgrammer.....	6
2.2.3	OpenSSL .....	6
2.2.4	AWS CLI.....	6
2.2.5	Terminal emulator.....	6
2.2.6	Android studio or Xcode.....	6
2.3	OTA configuration in AWS .....	7
2.4	Mobile App setup.....	7
<b>3</b>	<b>INTEGRATION OF AWS FREERTOS AND OPTIGA™ TRUST M.....</b>	<b>13</b>
3.1	AWS Sources .....	13
3.2	OPTIGA™ Trust M Source .....	13
3.3	Mutual Authentication Source .....	13
<b>4</b>	<b>CODE COMPILATION AND BINARY FLASHING .....</b>	<b>16</b>
4.1	Code Compilation Steps .....	16
4.2	Binary flashing.....	20
<b>5</b>	<b>SETUP AWS IOT THING ON AWS CONSOLE.....</b>	<b>22</b>
<b>6</b>	<b>MQTT DEMO .....</b>	<b>25</b>
6.1	Setup AWS config in the STM32WB55 Source Code .....	25
6.2	Run MQTT Demo .....	25
<b>7</b>	<b>OTA DEMO .....</b>	<b>28</b>
7.1	Setup AWS config in the STM32WB55 Source Code .....	28
7.1.1	OTA Firmware Version upgrade .....	28
7.1.2	AWS configuration for OTA Job.....	29
7.2	Run OTA Demo .....	34
<b>8</b>	<b>REFERENCES.....</b>	<b>37</b>

## DEFINITION, ACRONYMS AND ABBREVIATIONS

Definition/Acronym/Abbreviation	Description
AWS	Amazon Web Services
AWS CLI	AWS Command Line Interface
BLE	Bluetooth Low Energy
DFU	Device Firmware Update
FUS	Firmware Update Service
MQTT	Message Queuing Telemetry Transport
OTA	Over-the-Air
SSK	Security Starter Kit
STM	STMicroelectronics

# 1 INTRODUCTION

## 1.1 Purpose of the Document

This document describes how to integrate the AWS FreeRTOS, BLE, MQTT and OPTIGA™ Trust M stack on STM32WB55 board. The AWS basic demo is also enabled, which securely communicates with AWS Cloud using OPTIGA™ Trust M chip. This document will also guide the user through the AWS IoT console features to understand the Amazon IoT web services.

## 1.2 Architecture

The STM32WB55 board has Bluetooth Low Energy (BLE) as the wireless interface. A proxy or gateway will be required to access AWS services to gain internet connectivity via Wi-Fi or Ethernet. In this case, a mobile device (either Android or iOS) having BLE and 4G (or Wi-Fi) connectivity will facilitate as gateway. An Android or iOS Mobile application will act as a proxy between BLE and another network.



Figure 1: Hardware Setup

## 1.3 SSK Suit Package Contents

1. Download the SSK STM32WB55 release package (STM32WB55\_SSK\_Pkg\_Rel tar.gz) from the Arrow Electronics Github portal; <https://github.com/ArrowElectronics/Security-Starter-Kits>
2. Extract the STM32WB55\_SSK\_Pkg\_Rel tar.gz
  - Extracting the tar file, one will find the below contents:
    - Developer\_Guide\_STM32WB55\_SSK.pdf
    - Quick\_Start\_Guide\_STM32WB55\_SSK.pdf
    - Firmware\_Images
    - Source\_Code
    - SSK\_Cert\_And\_Config
    - RELEASE\_NOTES.txt, information about the release

## 2 SETTING UP THE DEVELOPMENT ENVIRONMENT

### 2.1 Hardware setup

The following hardware is needed for this project:

- STM32WB55 board from ST's P-NUCLEO-WB55 package
- Infineon's Shield2Go OPTIGA™ Trust M
- Custom cable to connect the OPTIGA™ Trust M to the STM32WB55 board
- Micro USB cables to Power on the Board. The same cable is used for debugging
- IOS or Android Mobile Device (As a Proxy or Gateway)
- A development PC or Mac for either Android or iOS application compilation.

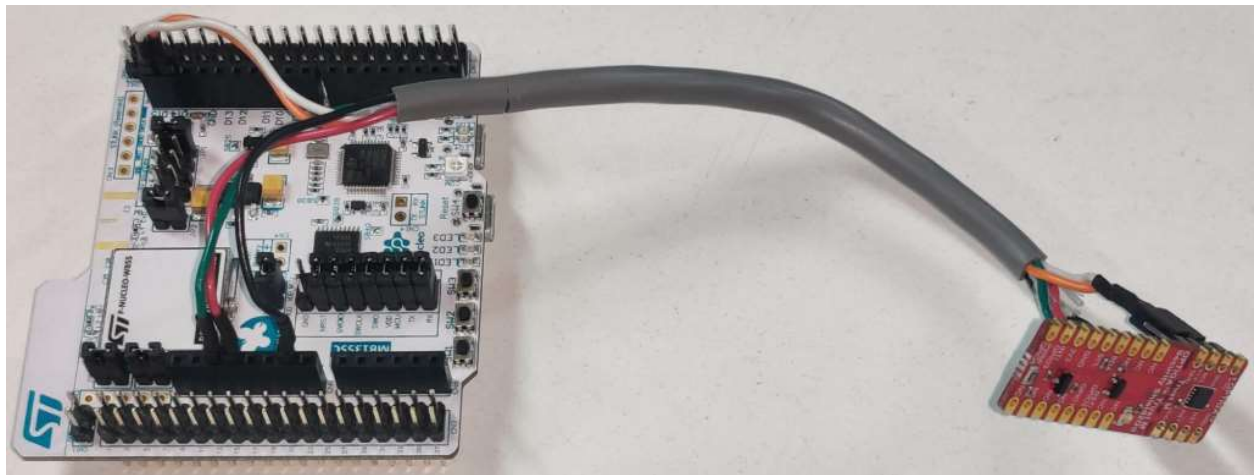
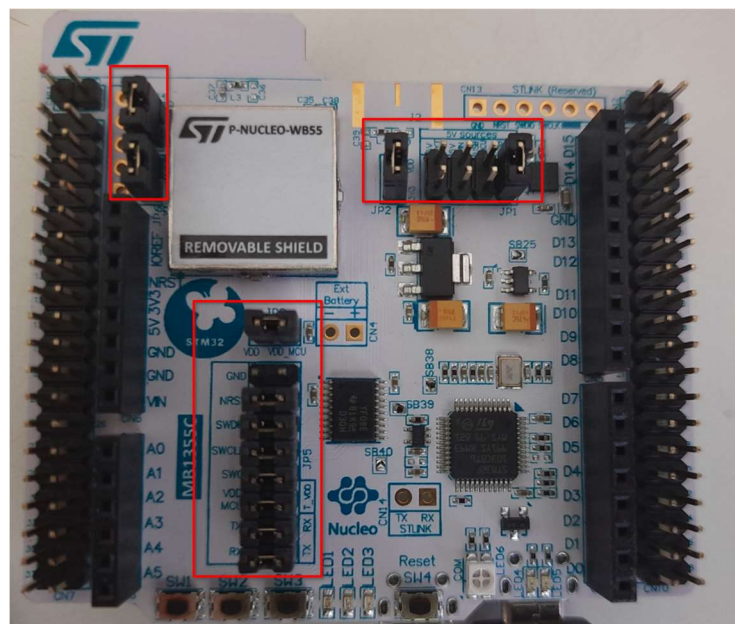


Figure 2: Hardware Setup

1. Please make sure that the below shown jumpers are connected on STM32WB55 board.



2. Connection between OPTIGA™ Trust M with STM32WB55 board using the ribbon cable as per below table.

Connections No	OPTIGA™ TrustM	STM32WB55
1(Red)	VCC	CN6.4 (+3v3)
2(Black)	GND	CN6.6 (GND)
3(Green)	RST	CN6.3 (NRST)
4(White)	SCL	CN5.10(D15)
5(Orange)	SDA	CN5.9(D14)

3. Connect Micro USB cable to power on the board and same for debugging purpose

## 2.2 Software Setup

### 2.2.1 STM32CubeIDE

STM32CubeIDE is software tool to compile the source code and generate the binary to flash on the STM32WB55.

- Download and Install STM32CubeIDE from ST site  
<https://www.st.com/en/development-tools/stm32cubeide.html>

### 2.2.2 STM32CubeProgrammer

STM32CubeProgrammer is required to program the Flash memory of STM32WB55 board.

- Install STM32CubeProgrammer from ST site:  
<https://www.st.com/en/development-tools/stm32cubeprog.html>

### 2.2.3 OpenSSL

OpenSSL is needed for certificate creation if you want to do OTA update.

### 2.2.4 AWS CLI

AWS CLI is needed to automate various tasks and import OTA certificate in AWS.

### 2.2.5 Terminal emulator

A serial terminal software like **Putty** or **Minicom** is needed for debugging the Amazon FreeRTOS application running on STM32WB55 (using USB virtual COM port).

### 2.2.6 Android studio or Xcode

The software development tool like Android Studio to manage Android app or Xcode for IOS Mobile is needed to compile the Amazon FreeRTOS BLE gateway Mobile application.

[**Note:** Source code are available under Source\_Code/Mobile\_App\_Source.zip]

## 2.3 OTA configuration in AWS

To perform OTA, user should have to configure the below listed items on [AWS console](#)

- Check the Prerequisites for OTA updates using MQTT.
- Create an Amazon S3 bucket to store your update.
- Create an OTA Update service role.
- Create an OTA user policy.
- Create a code-signing certificate.
- Grant access to code signing for AWS IoT

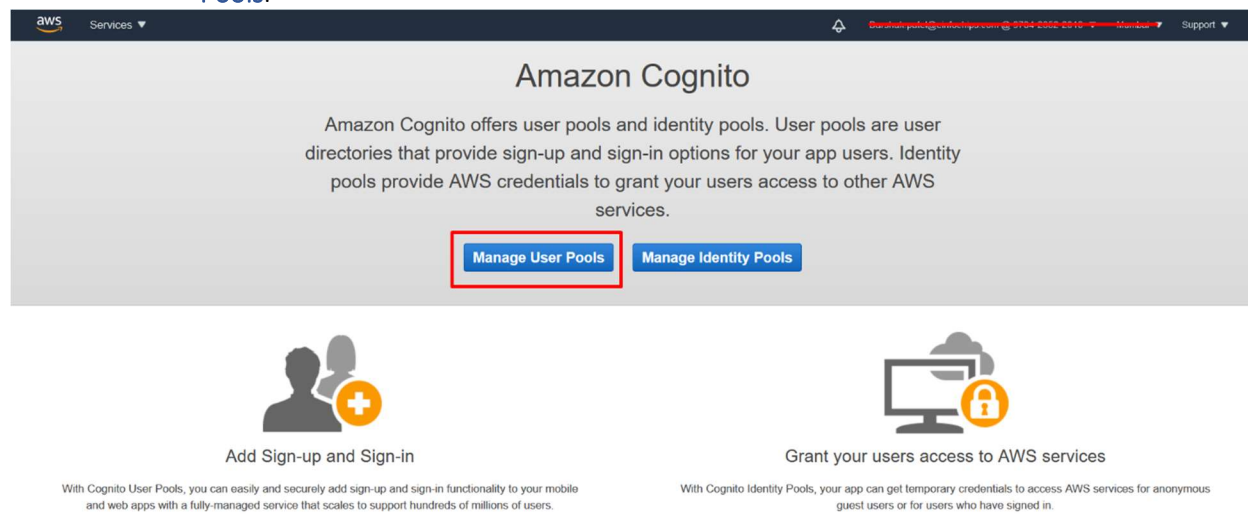
Detailed configuration and steps available on AWS document.

<https://docs.aws.amazon.com/freertos/latest/userguide/ota-prereqs.html>

## 2.4 Mobile App setup

Please follow below steps to create AWS Cognito user. This is needed to login into the Mobile APP. It is presumed that the User has AWS Account Login - <https://aws.amazon.com/console/>

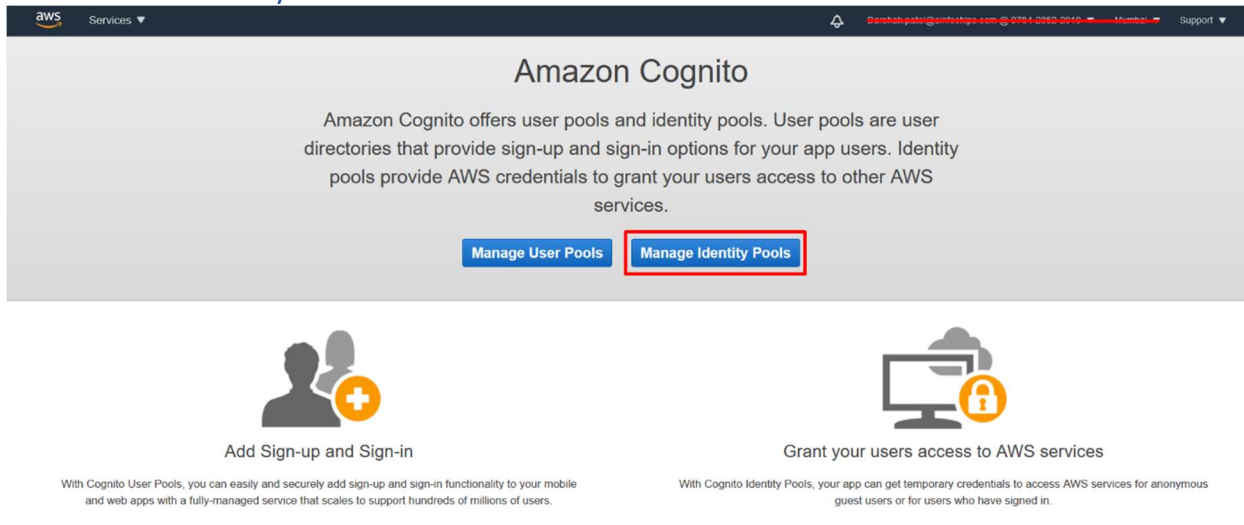
1. In case if you have created the Amazon Cognito user pool in AWS console then go to step 6, otherwise continue with step 2
2. To create an Amazon Cognito user pool in AWS Console
  - In AWS console, Open the Amazon Cognito console (Service), and choose **Manage User Pools**.



- Choose **Create a user pool**.
- Give the user pool a name, and then choose **Review defaults**.
- From the navigation pane, choose **App clients**, and then choose **Add an app client**.
- Enter a name for the app client, and then choose **Create app client**.
- From the navigation pane, choose **Review**, and then choose **Create pool**.
- **Make a note** of the pool ID that appears on the **General Settings** page of your user pool.
- From the navigation pane, choose **App clients**, and then choose **Show details**.
- **Make a note** of the app client ID and app client secret.

### 3. To create an Amazon Cognito identity pool in AWS Console

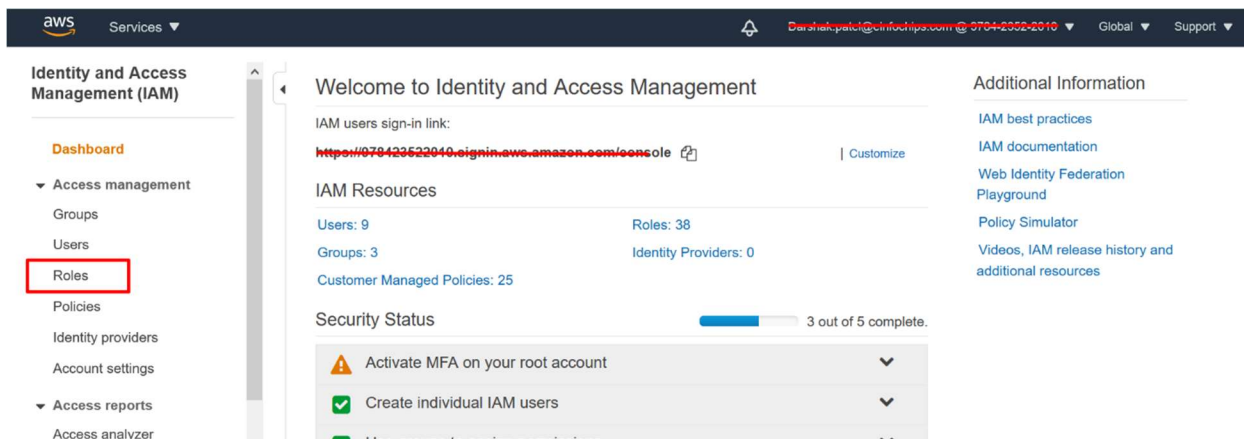
- In AWS Console, Open the Amazon Cognito console (Service), and choose **Manage Identity Pools**.



- Choose **Create new identity pool**
- Enter a name for your identity pool.
- Expand **Authentication providers**, choose the **Cognito** tab, and then enter your user pool ID and app client ID.
- Choose **Create Pool**.
- Expand **View Details**, and **make a note** of the two IAM role names. Choose **Allow** to create the IAM roles for authenticated and unauthenticated identities to access Amazon Cognito.
- Choose **Edit identity pool**. **Make a note** of the identity pool ID. It should be of the form `us-west-2:12345678-1234-1234-1234-123456789012`.

### 4. To create and attach an IAM policy to the authenticated identity

- In AWS console, Open the IAM console, and from the navigation pane, choose **Roles**.



- Find and choose your authenticated identity's role, choose **Add inline policy**.



- Choose the **JSON** tab, and paste the following JSON:

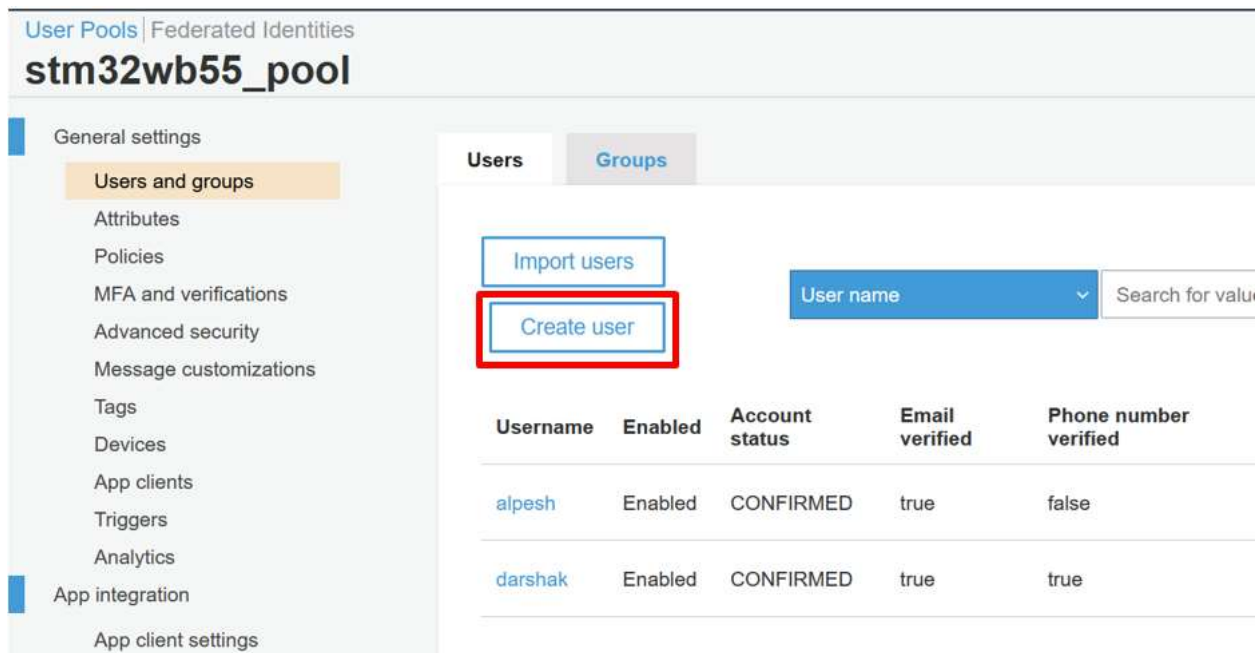
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:AttachPolicy",
        "iot:AttachPrincipalPolicy",
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot>DeleteThingShadow"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

- Choose **Review policy**, enter a name for the policy, and then choose **Create policy**.

## 5. Create New User in Cognito User pool

This can be done either from the Android application (login screen) or from the AWS console. In case, if a confirmation is needed this can be done from the AWS Cognito console (in Users and groups).

- Login AWS console - <https://aws.amazon.com/console/>
- AWS Console >> Cognito Service >> Manage User pool >> Select created user pool**
- General Settings >> Users and groups >> **Create user**
- Make a note** of created Username and Password, which is used for login into the Mobile APP.



#### 6. Create an AWS IoT policy

- Open the [AWS IoT console](#).
- In the navigation pane, choose **Secure**, choose **Policies**, and then choose **Create**. Enter a name (**STM32WB55-policy**) to identify your policy. In the **Add statements** section, choose **Advanced mode**. Copy and paste the following JSON into the policy editor window.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:*",
      "Resource": "*"
    }
  ]
}
```

- Choose **Create**.

**[Note:** Keep your AWS IoT and Amazon Cognito information on hand. You need the endpoint and IDs to authenticate your mobile application with the AWS Cloud]

#### 7. Create a new txt file with the following name **AWS\_Config.txt** and enter the above noted parameters in the file i.e.

```
AWS_REGION#<Enter-Region>
AWS_IOT_POLICY#STM32WB55-policy
AWS_COGNITO_IDENTITY_POOLID#<Enter-Identity-Pool-ID>
```

AWS\_COGNITO\_USER\_POOLID#<Enter-User-Pool-ID>  
 AWS\_COGNITO\_USER\_APPCLIENTID#<Enter-User-App-ID>  
 AWS\_COGNITO\_USER\_APPCLIENTSECRET#<Enter-User-App-SECRET>

8. Install provided Mobile APK from the Github site below and run.  
<https://github.com/ArrowElectronics/Security-Starter-Kits>
9. Follow for FIRST TIME SETUP ONLY otherwise press “SKIP”  
 Click on “Browse for the Cognito details” and upload the [AWS\\_Config.txt](#) file, verify the configurations, and then Press “Save” button on screen as shown - [First Time configuration screen](#)

[First Time configuration screen](#)

SSK APP

BROWSE FOR COGNITO DETAILS

AWS Region\*  
 ap-south-1

AWS IoT Policy Name\*  
 SecurityPolicy

AWS Cognito Identity PoolId\*  
 ap-south-1:a-988b-209b8c1

AWS Cognito User PoolId\*  
 ap-south-1:PR

AWS Cognito User AppClientId\*  
 30g7m21933tan

AWS Cognito User AppClientSecret\*  
 1hv8f1n5t6pvh59

SAVE

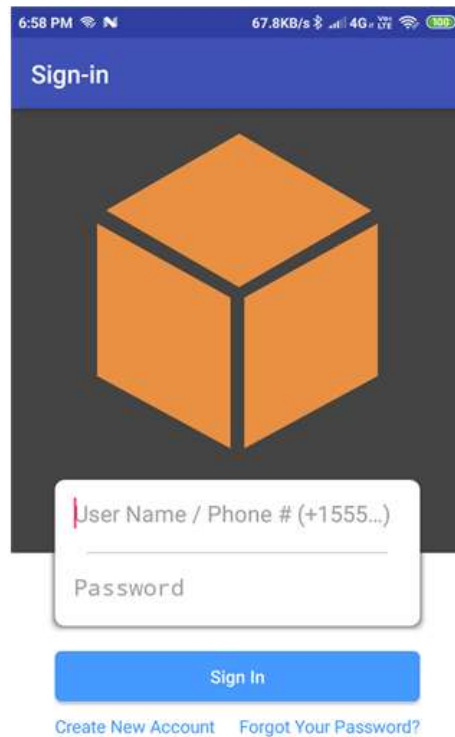
[Press Skip or reconfigure if required](#)

SSK APP

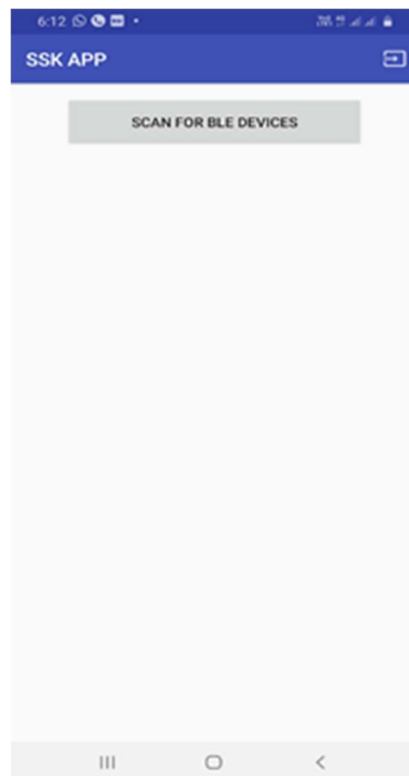
SKIP

BROWSE FOR COGNITO DETAILS

10. Login in the App using the Cognito user credentials



11. After Successful Login, It shows below screen.



### 3 INTEGRATION OF AWS FREERTOS AND OPTIGA™ TRUST M

#### 3.1 AWS Sources

The Amazon FreeRTOS for STM32WB55.Nucleo software package is an extension of Amazon FreeRTOS and is available at the Github repository;

- It is provided as a `Source_Code/STM32WB55_SSK_AWS_Rel` with mandatory SLA0078 license acceptance.
- `aws_demos`: Provides the MQTT and OTA demo. User can select demos using compilation flags, more details are described in [section 6.1](#) and [section 7.1](#)

#### 3.2 OPTIGA™ Trust M Source

The OPTIGA™ Trust M is a high-end security solution that provides an anchor of trust for connecting IoT devices to the cloud, giving every IoT device its own unique identity. This pre-personalized turnkey solution offers secured, zero-touch onboarding and the high performance needed for quick cloud access.

OPTIGA™ Trust M security chip Software Framework is ported In provided package, available at location: <https://github.com/Infineon/OPTIGA-trust-m>

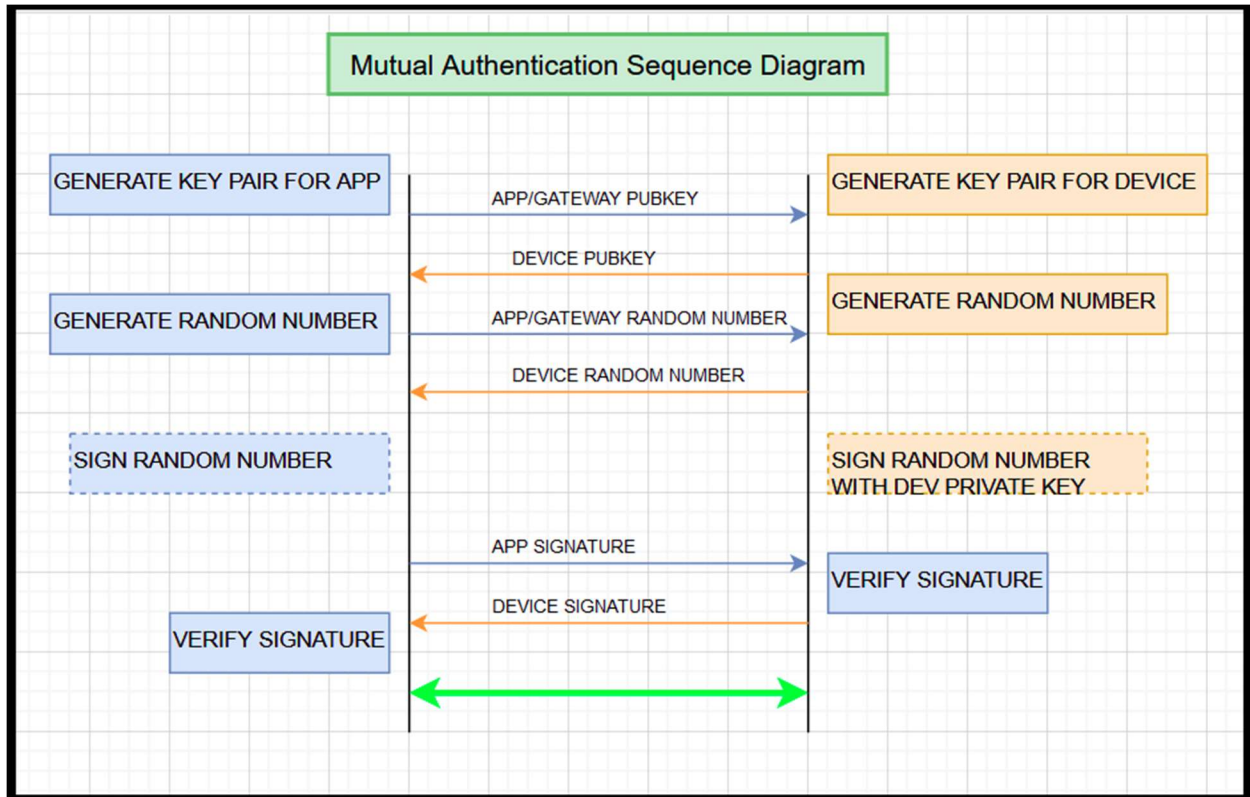
To port software framework template files shown below has to be updated based on your Hardware platform.

1. `pal_ifx_i2c_config.c`
2. `pal_i2c.c`
3. `pal_gpio.c`
4. `pal_os_timer.c`
5. `pal_os_event.c`

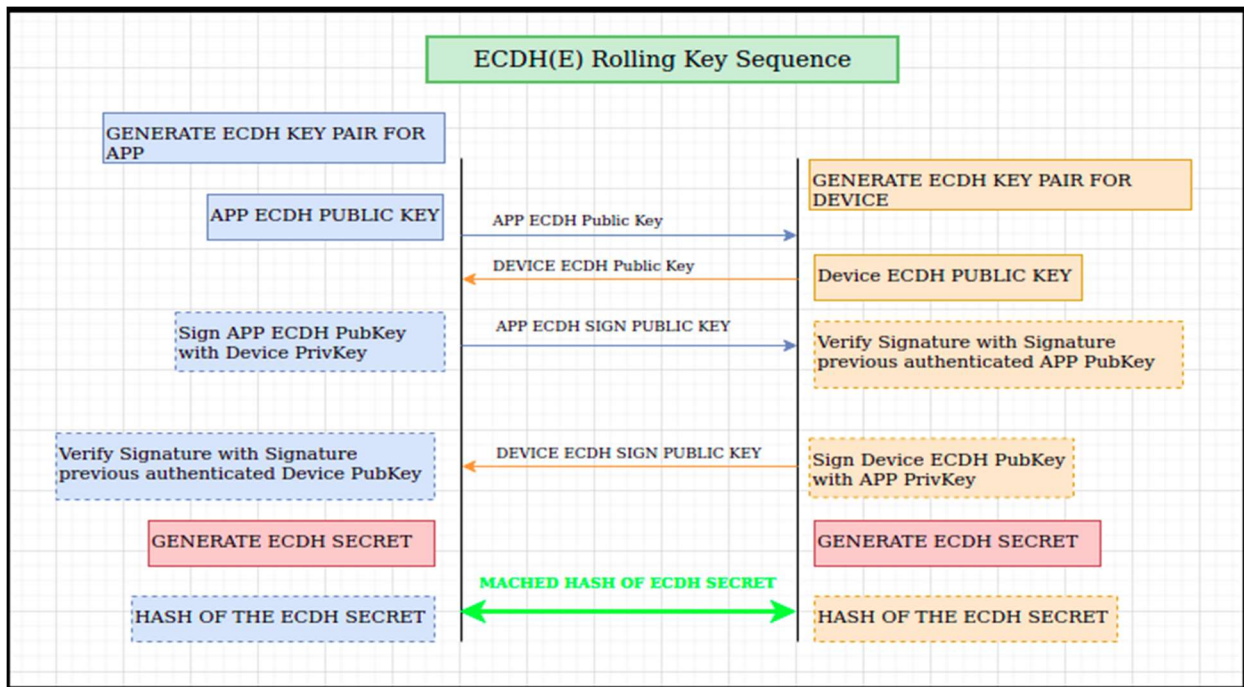
Refer link for reference: <https://github.com/Infineon/OPTIGA-trust-m/wiki/Porting-Guide>

#### 3.3 Mutual Authentication Source

Mutual Authentication is the device authentication process, which has to be executed before Application begins to communicate with AWS cloud. It fails if any unauthenticated devices try to communicate with the mobile app.



- After completing mutual authentication, the kit generates one secret key, which uses a AES 256 symmetric key to encrypt and decrypt for all the communications.

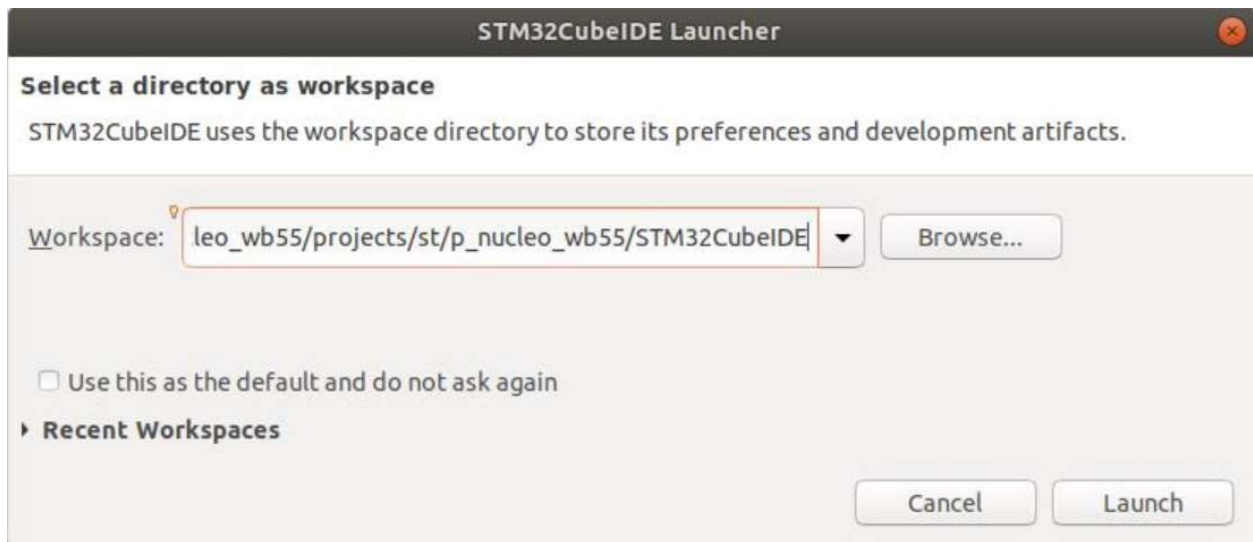


- The SSK uses OPTIGA™ Trust M security chip to store the secrets e.g. certificates, keys etc. Also, it uses hardware crypto. Library provided by OPTIGA™ Trust M software stack.
- To implement mutual authentication, add one custom BLE service in STM32WB55 AWS source code available at [SSK\\_SUIT\\_EVAL\\_STM32WB55\\_Rel\\_\[Release\]/Source\\_code / STM32\\_SSK\\_AWS\\_Rel\\_\[version\]/demos/ble/aws\\_ble\\_mutual\\_auth.c](#)

## 4 CODE COMPILATION AND BINARY FLASHING

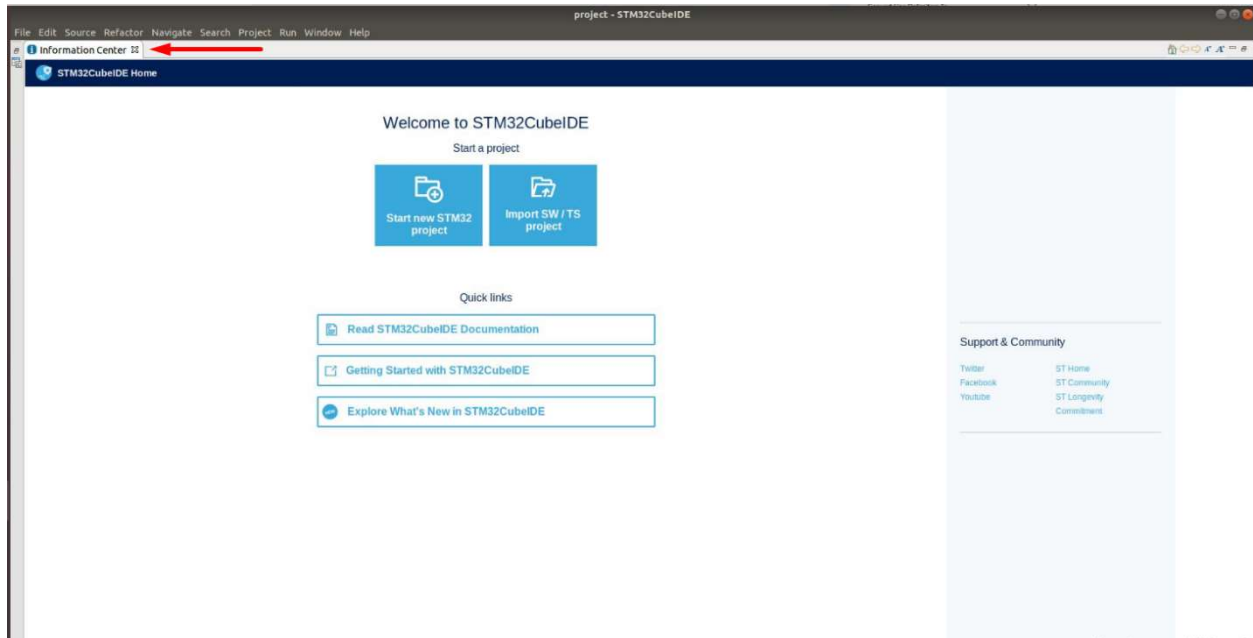
### 4.1 Code Compilation Steps

1. Download and extract the SSK STM32WB55 release package (STM32WB55\_SSK\_Pkg\_Rel.tar.gz) from the Arrow Electronics Github portal; <https://github.com/ArrowElectronics/Security-Starter-Kits>
2. In the extracted directory, project available under [Source\\_Code/STM32\\_SSK\\_AWS\\_Rel\\_\[version\]/projects/st/p\\_nucleo\\_wb55/STM32CubeIDE](#).
3. Launch STM32CubeIDE application.
4. Open a workspace under “projects\st\p\_nucleo\_wb55\STM32CubeIDE”.

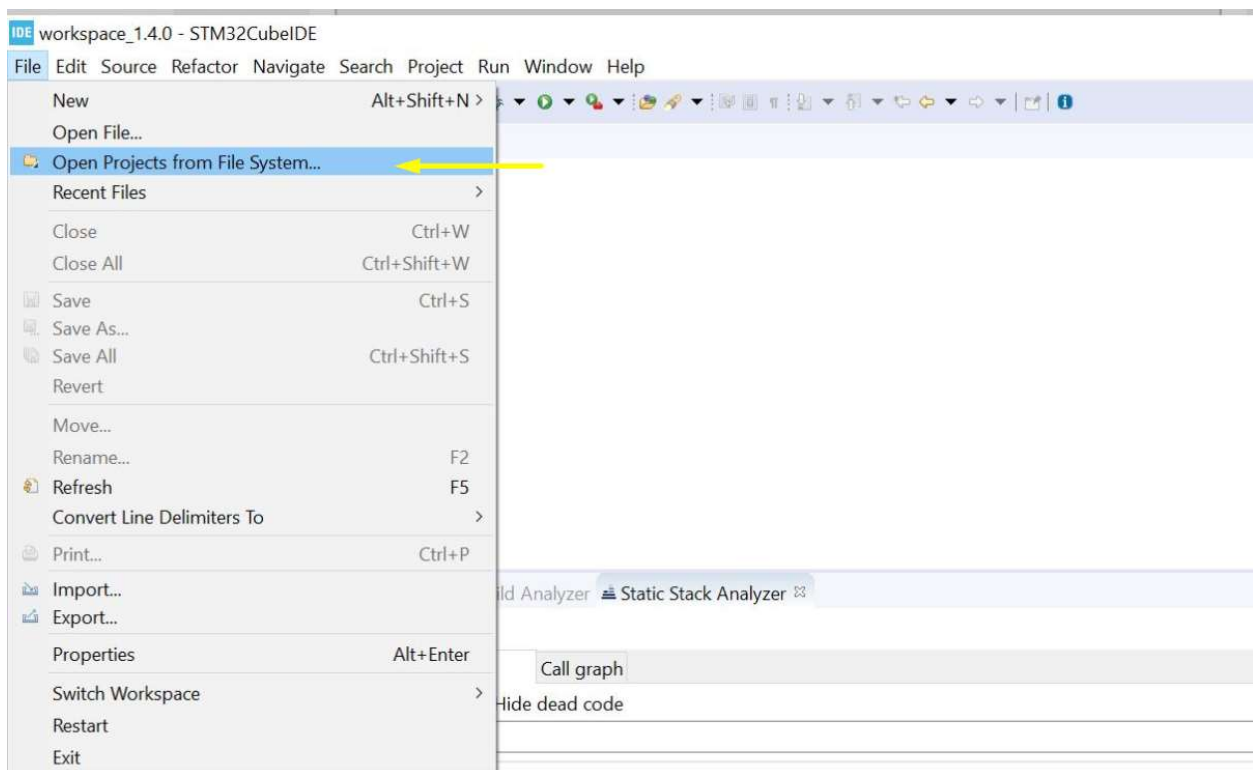


5. Close the Information Center to view the C/C++ perspective.





6. Select menu File >> Open project from File system... >> Finish.

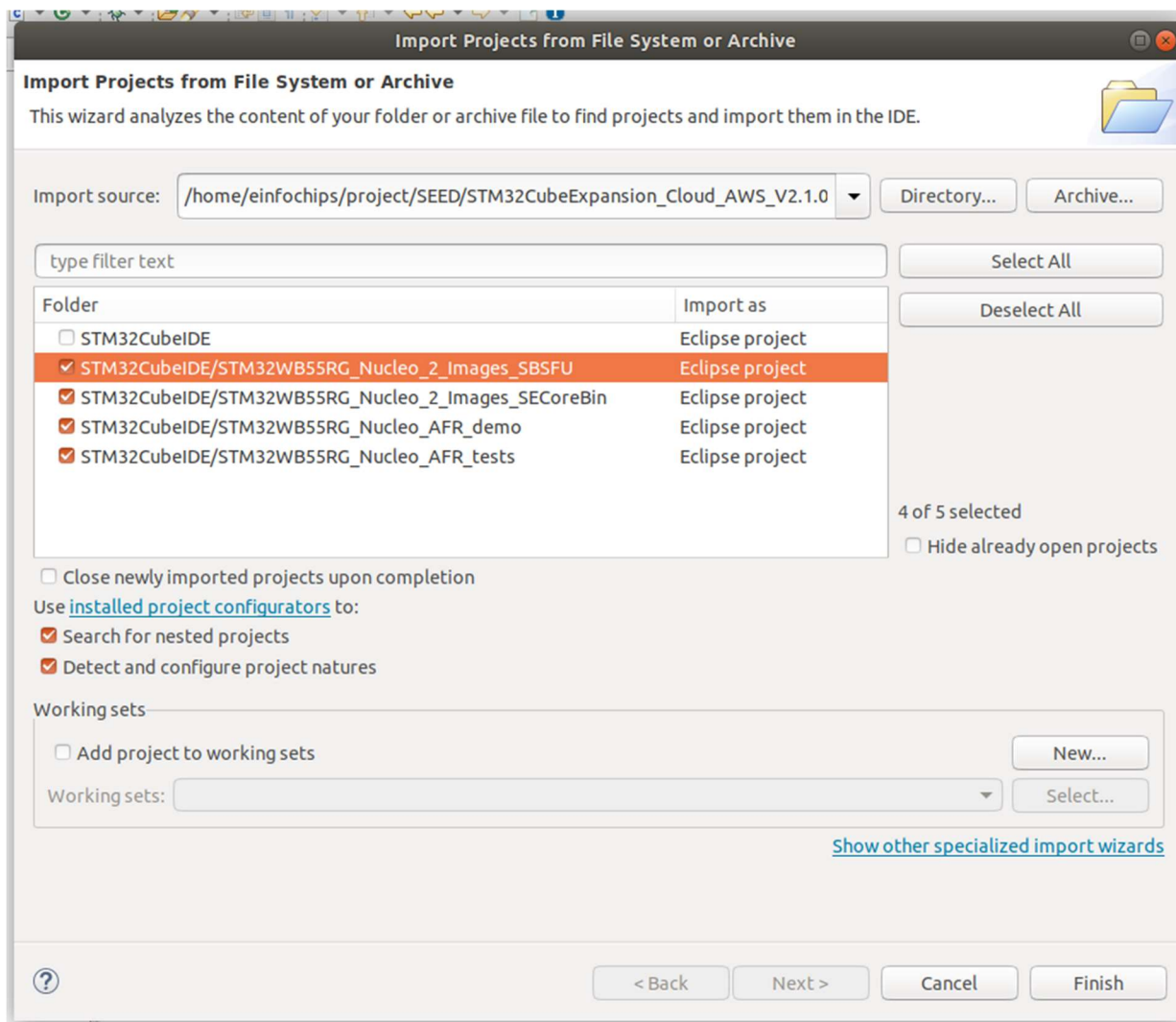


7. In “Select directory” browse to projects/st/ p\_nucleo\_wb55/STM32CubeIDE folder.

8. Import the four projects:

- STM32WB55RG\_Nucleo\_2\_Images\_SECoreBin
- STM32WB55RG\_Nucleo\_2\_Images\_SBSFU

- STM32WB55RG\_Nucleo\_AFR\_demo
- STM32WB55RG\_Nucleo\_AFR\_tests



- For Compilation, you need to follow below sequence.
- Compile **SECoreBin** first, then **SBSFU**. And then **AFR\_demo** projects to avoid the build conflict. Its open console and you can see build messages as shown below.

```

Problems Tasks Console Properties
CDT Build Console [STM32WB55RG_Nucleo_2_Images_SECoreBin]

arm-none-eabi-size SeCoreBin.elf
text data bss dec hex filename
8338 56 552 8946 22f2 SeCoreBin.elf
Finished building: default.size.stdout

arm-none-eabi-objdump -h -S SeCoreBin.elf > "SeCoreBin.list"
Finished building: SeCoreBin.list

arm-none-eabi-objcopy -O binary SeCoreBin.elf "SeCoreBin.bin"
Finished building: SeCoreBin.bin

"/home/einfochips/project/SEED/STM32CubeExpansion_Cloud_AWS_V2.1.0.ALPHA1,
Copying SeCoreBin.bin and SeCoreBin.elf to common build directory.

10:54:35 Build Finished. 0 errors, 0 warnings. (took 5s.531ms)

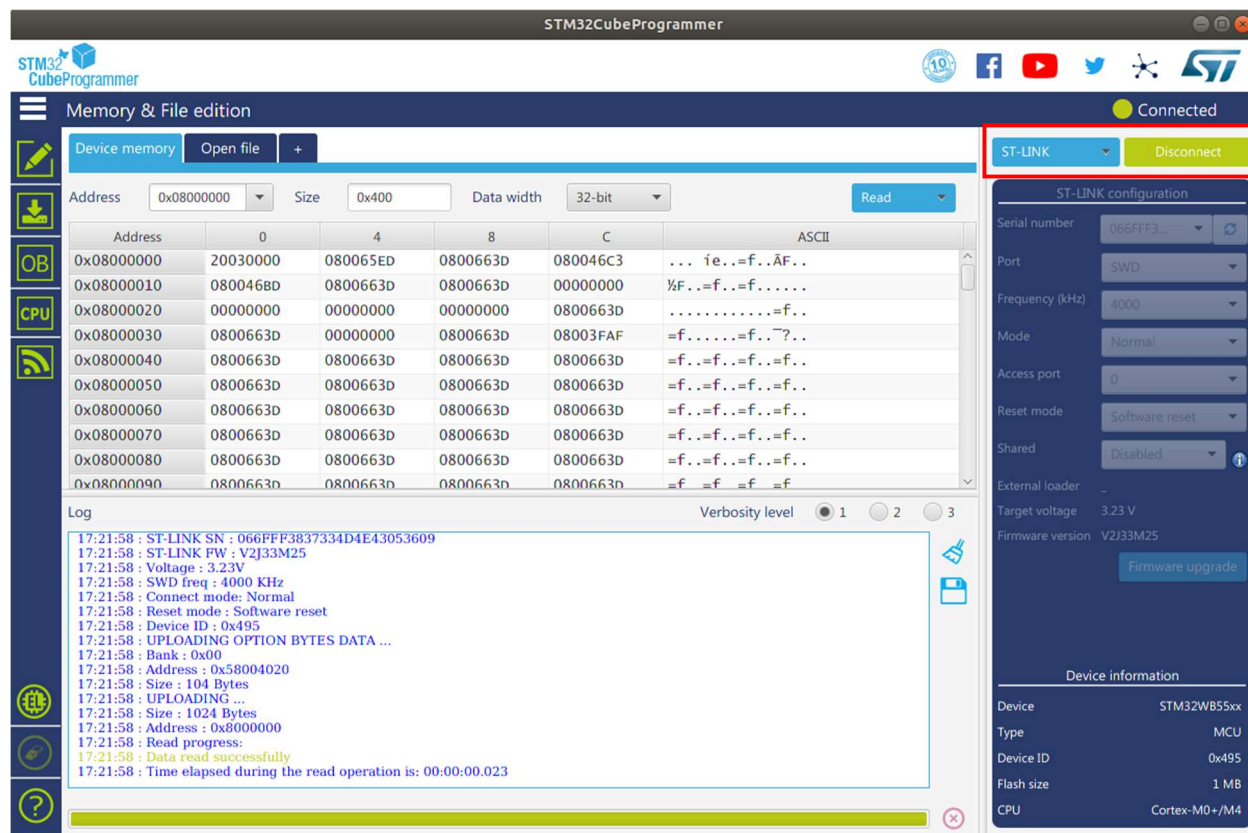
```



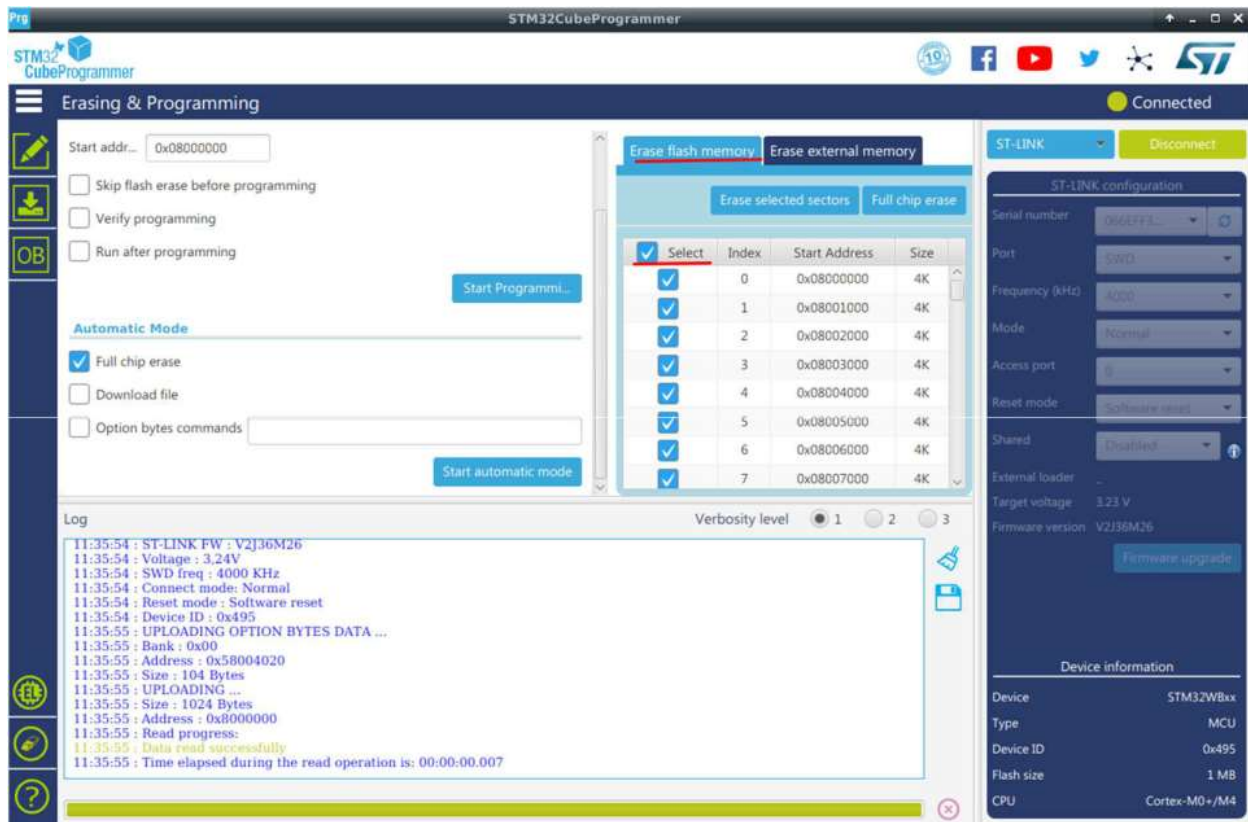
## 4.2 Binary flashing

Final binaries are located in "Source\_Code/STM32\_SSK\_AWS\_Rel\_[version]/build" directory.

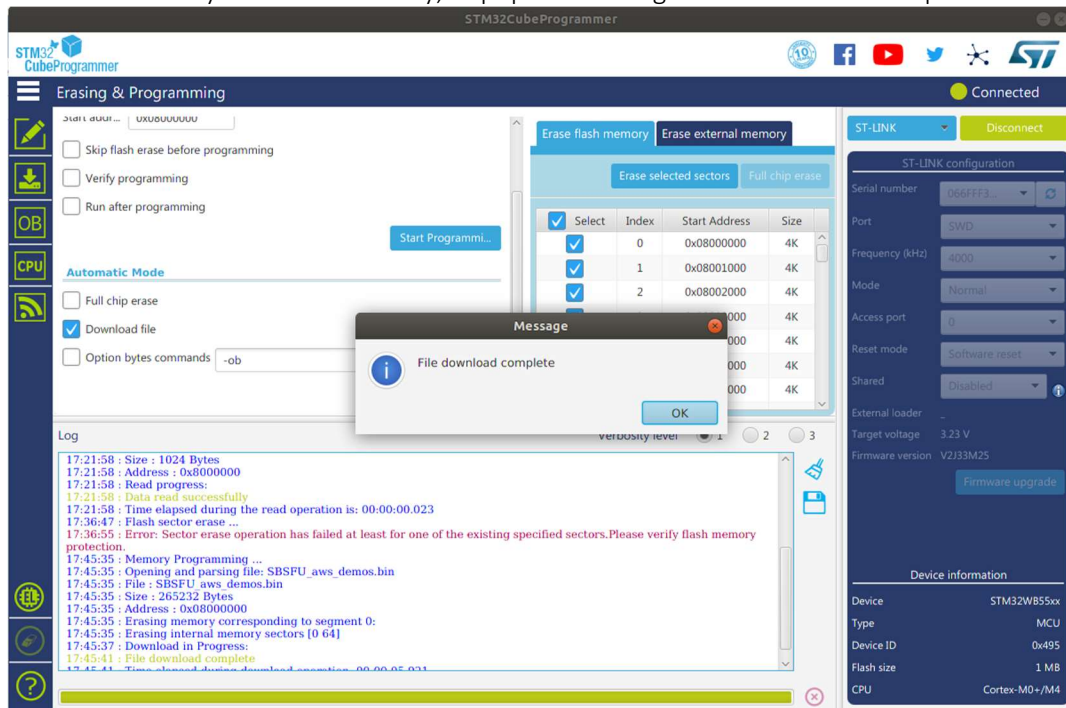
- SBSFU\_aws\_demos.bin - bootloader + demo application
  - aws\_demos.sfb - demo application + OTA update header
1. Plug a USB cable from PC to the STM32WB55 board (Board backside, named as "ST-LINK")
  2. Open STM32CubeProgrammer and click on connect button to connect the board as shown below.



3. First, the Flash memory must be entirely erased as depicted in the screenshot below.

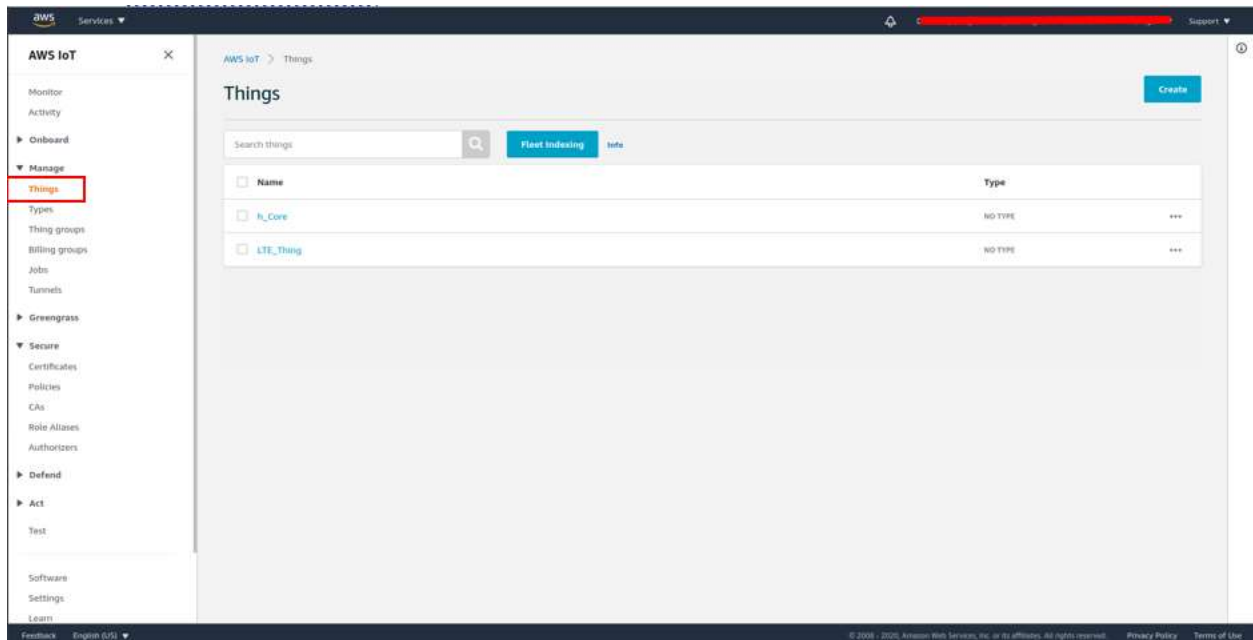


- Flash the binary (e.g. build\SBSFU\_aws\_demos.bin) using STM32CubeProgrammer (address 0x08000000).
- After successfully flashed the Binary, Popup the message "File Download complete". Press OK

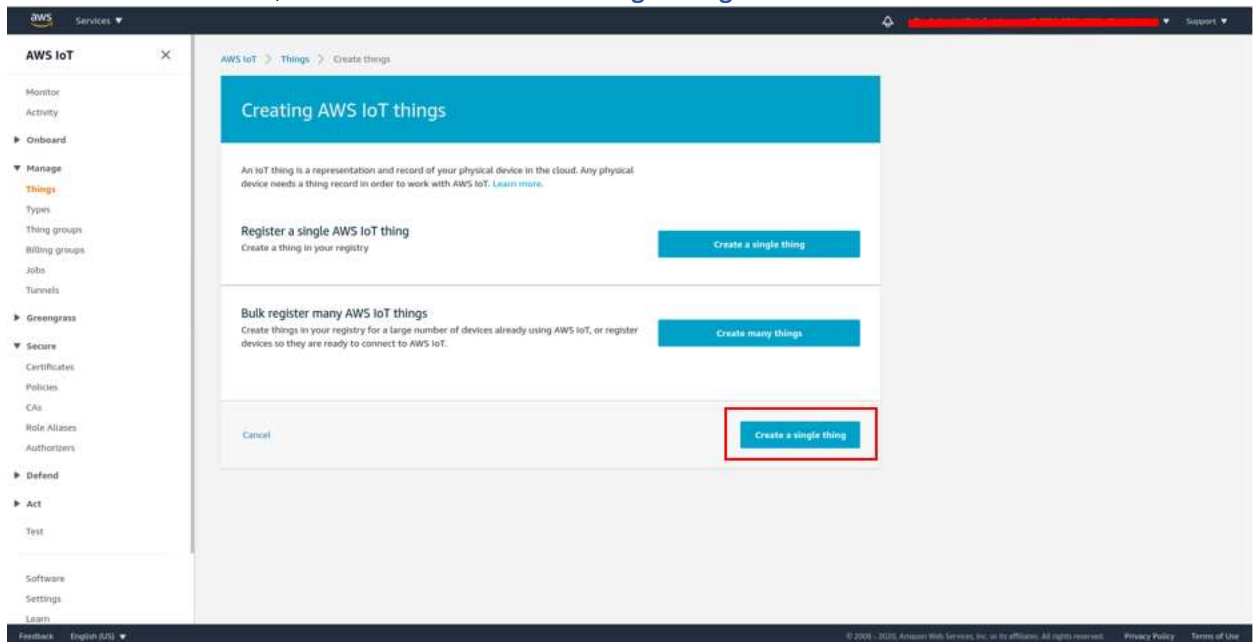


## 5 SETUP AWS IOT THING ON AWS CONSOLE

1. Open the [AWS IoT console](#), and from the navigation pane, choose **Manage**, and then choose **Things**.



2. Choose **Create**, and then choose **Create a single thing**.



3. Enter a name for your device, and then choose **Next**.

**Add your device to the thing registry**

This step creates an entry in the thing registry and a thing shadow for your device.

**Name**  
STM32WB55

**Apply a type to this thing**  
Using a thing type simplifies device management by providing consistent registry data for things that share a type. Types provide things with a common set of attributes, which describe the identity and capabilities of your device, and a description.

Thing Type  
No type selected [Create a type](#)

**Add this thing to a group**  
Adding your thing to a group allows you to manage devices remotely using jobs.

Thing Group  
Groups / [Create group](#) [Change](#)

**Set searchable thing attributes (optional)**  
Enter a value for one or more of these attributes so that you can search for your things in the registry.

Attribute key	Value
Provide an attribute key, e.g. Manufacturer	Provide an attribute value, e.g. Acme-Corporation

[Add another](#) [Clear](#)

**Show thing shadow** ▾

[Cancel](#) [Back](#) [Next](#)

4. If you are connecting your microcontroller to the cloud through a mobile device, choose **Create thing without certificate**. Because the Mobile SDKs use Amazon Cognito for device authentication, you do not need to create a device certificate for demos that use Bluetooth Low Energy.

**CREATE A THING** STEP 2/3

**Add a certificate for your thing**

A certificate is used to authenticate your device's connection to AWS IoT.

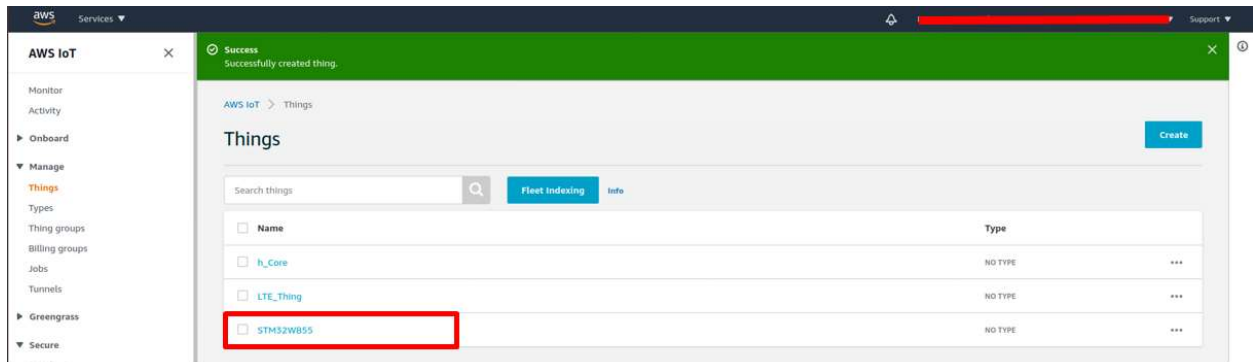
**One-click certificate creation (recommended)**  
This will generate a certificate, public key, and private key using AWS IoT's certificate authority. [Create certificate](#)

**Create with CSR**  
Upload your own certificate signing request (CSR) based on a private key you own. [Create with CSR](#)

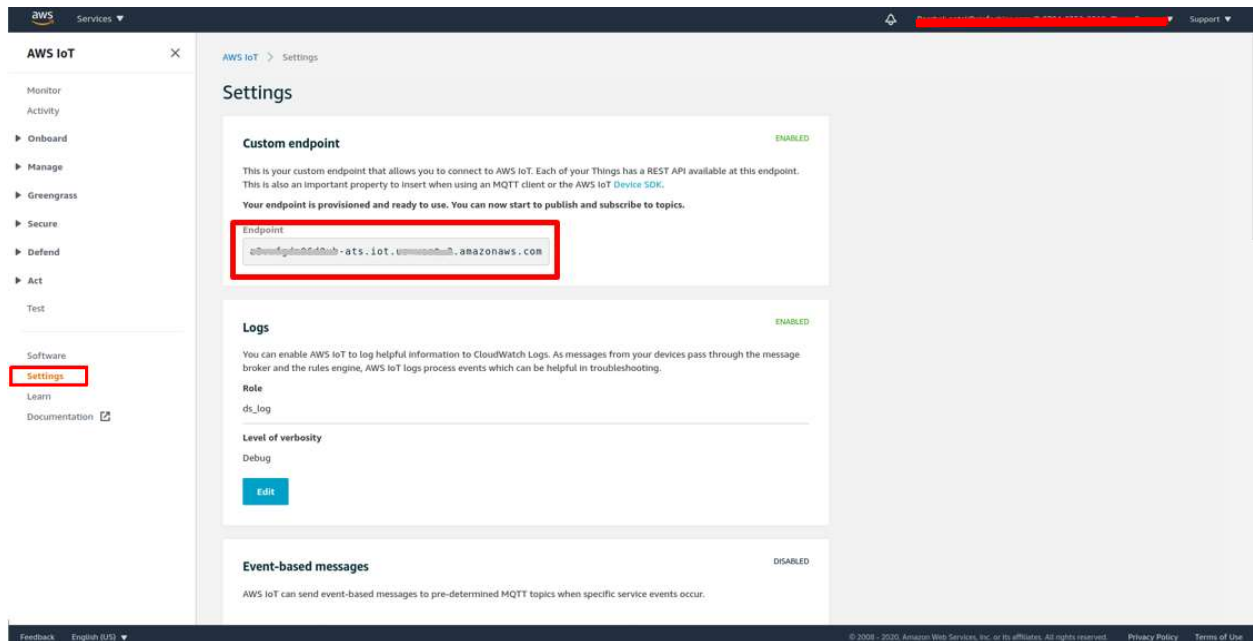
**Use my certificate**  
Register your CA certificate and use your own certificates for one or many devices. [Get started](#)

**Skip certificate and create thing**  
You will need to add a certificate to your thing later before your device can connect to AWS IoT. [Create thing without certificate](#)

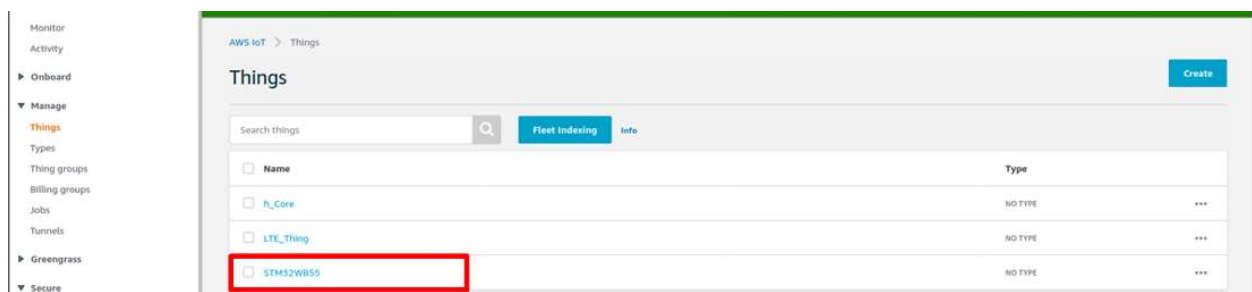




5. In the left navigation panel, choose Settings. Your AWS IoT endpoint is displayed in Endpoint. It should be like `*<accountID>*-ats.iot.*<region>*.amazonaws.com`. Record the endpoint it required when you first time start the board.



6. In the left navigation pane, choose **Manage >> Things**. Your device should have an AWS IoT thing name. **Record this name.**





## 6 MQTT DEMO

### 6.1 Setup AWS config in the STM32WB55 Source Code

1. Configure AWS IoT thing name into `demos/include/aws_clientcredential.h` file in IDE, and specify values for the following `#define clientcredentialIOT_THING_NAME ST32WB55` (Thing name, created in AWS console).
2. Make sure aws\_demos are configured for MQTT demo in `config_files\aws_demo_config.h` in CubeIDE (or vendors\st\boards\p\_nucleo\_wb55\aws\_demos\config\_files\aws\_demo\_config.h):

```
#define CONFIG_MQTT_DEMO_ENABLED
```

[Note: In addition, all the other CONFIG\_\*\_ENABLED flags are commented.]

3. Then compile the `STM32WB55RG_Nucleo_AFR_demo` project as mentioned in the [section 4.1](#) and Flash the binary on the STM32WB55 board as mentioned in the [section 4.2](#)

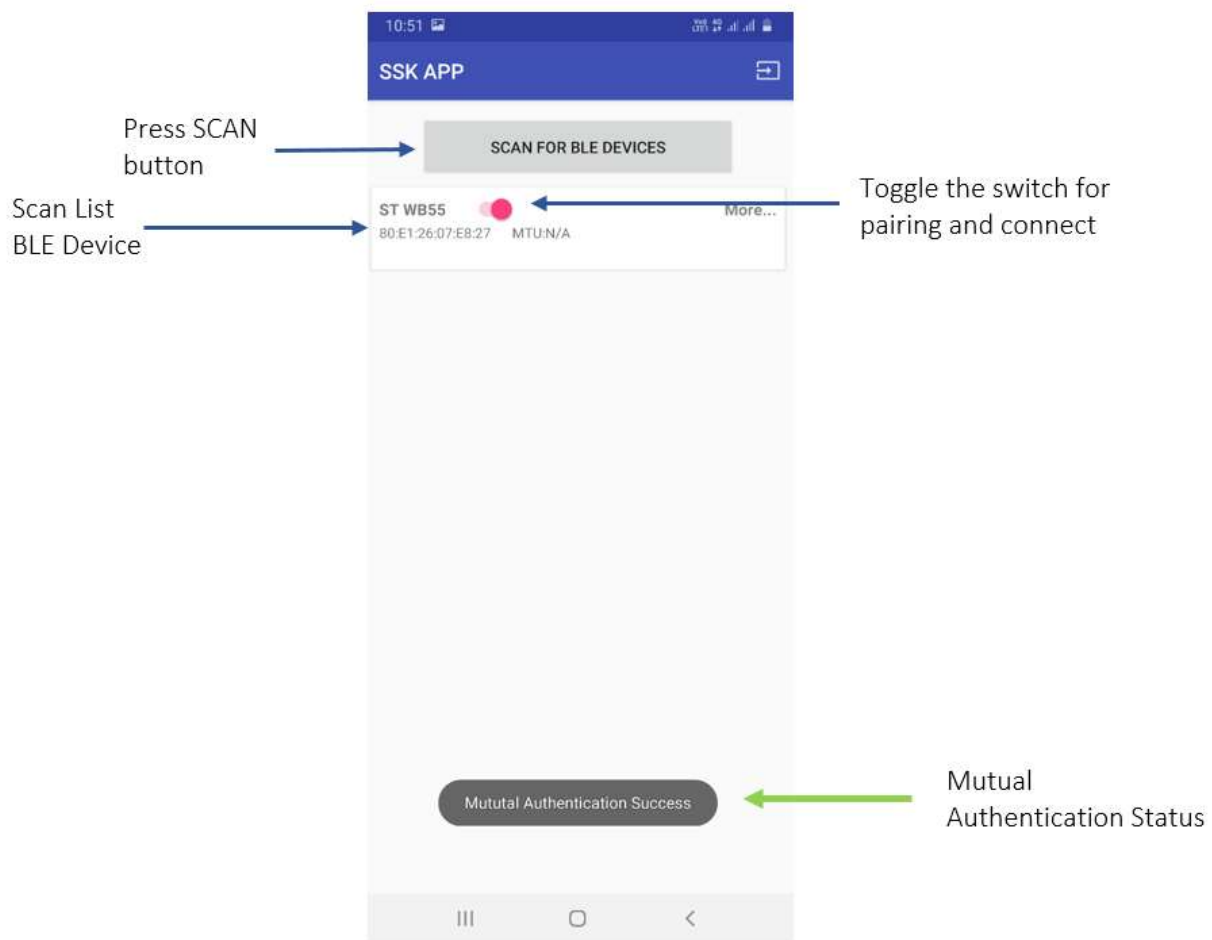
### 6.2 Run MQTT Demo

1. If you want to modify the Endpoint URL, then follow the [step 2](#) otherwise Reset the board and jump to [step 3](#).
2. Press the SW2 button and Reset the board (By Pressing “Reset” button as shown in figure 2). It will ask to enter endpoint URL on Serial console screen as below.  
Paste the Endpoint URL here and press the “Enter” key on keyboard. (Note: Default Echo is OFF, so cannot see your Endpoint URL value.)  
So, to verify that the correct endpoint URL was entered, on the serial console it will display the URL as shown above (with yellow marked).

```
===== Please Enter Your Endpoint URL =====
e.g- xxxxxxxxxxxx-ats.iot.xx-xxx-x.amazonaws.com

Entered Endpoint URL=2wufgdn06dwb-ats.iot.ap-south-1.amazonaws.com
=====
```

3. Click on the SCAN FOR BLE DEVICES button. Connect and Pair with STM32WB55 device. On the serial console Press “Y” if passkey matches with mobile pair key.



4. During this demo, **Mutual Authentication** is enabled between the Mobile Device and STM32WB55. Only **provisioning** device can start communication else Mutual Authentication fails and logs out from the AWS Cognito Session in Mobile APP.
5. To see the MQTT message on AWS console, Open AWS Console >> IoT Service >> Test page and Subscribe the Topic: **iotdemo/topic/#**

AWS IoT > Test

MQTT client Info Connected as iotconsole-1600938282582-0

**Subscriptions**

[Subscribe to a topic](#)

[Publish to a topic](#)

Subscribe

Devices publish MQTT messages on topics. You can use this client to subscribe to a topic and receive these messages.

Subscription topic

iotdemo/topic/#

Max message capture Info

100

Quality of Service Info

☒ 0 - This client will not acknowledge to the Device Gateway that messages are received

☐ 1 - This client will acknowledge to the Device Gateway that messages are received

MQTT payload display

☒ Auto-format JSON payloads (improves readability)

☐ Display payloads as strings (more accurate)

☐ Display raw payloads (in hexadecimal)

[Subscribe to topic](#)

## MQTT Messages on AWS console,

AWS IoT > Test

MQTT client Info Connected as iotconsole-1600938282582-0

**Subscriptions**

[Subscribe to a topic](#)

[Publish to a topic](#)

iotdemo/topic/#

[Publish to topic](#)

1 {  
2 "message": "Hello from AWS IoT console"  
3 }

iotdemo/topic/4 September 24, 2020, 14:39:06 (UTC+0530) [Export](#) [Hide](#)

We cannot display the message as JSON, and are instead displaying it as UTF-8 String.

BLE MQTT Message Count 19!

iotdemo/topic/3 September 24, 2020, 14:39:06 (UTC+0530) [Export](#) [Hide](#)

We cannot display the message as JSON, and are instead displaying it as UTF-8 String.

BLE MQTT Message Count 18!

6. MQTT Message will display on the screen i.e. BLE MQTT Message Count 2!

```

40 16521 [iot_thread] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/2
Publish topic name: iotdemo/topic/2
Publish retain flag: 0
Publ41 16536 [iot_thread] (MQTT connection 0x200116b8) MQTT PUBLISH operation queued.
42 16545 [iot_thread] Acknowledgment message for PUBLISH 1 will beYeeXegoe;/Ref++++s++++oc<#=#Eo[4]eUeeA[4]e+|e+sDv,eeN [4]eOIeeOt hae? [4]e(
43 16557 [iot_thread] [SECURITY][SEND] CIPHER TEXT = eVecJWHe:Q#e
44 16571 [iot_thread] 2 publishes received.
45 16575 [iot_thread] Publishing messages 2 to 3.
46 16580 [iot_thread] Plain Text=BLE MQTT Message Count 2!
47 16591 [iot_thread] [SECURITY][SEND] CIPHER TEXT = eVecJW_8ee-e|eUe'eePee\ee(eeGeeekensQ JTeMW
48 16601 [iot_thread] (MQTT connection 0x200116b8) MQTT PUBLISH operation queued.
49 16609 [iot_thread] Plain Text=BLE MQTT Message Count 3!
Zede++*Se]ee+?deee0![SECURITY][SEND] CIPHER TEXT = eVecJW_8ee-e|eUe'eePeeNeQee[4]e

```

## 7 OTA DEMO

### 7.1 Setup AWS config in the STM32WB55 Source Code

1. The aws\_demos application configured for OTA demonstrates an Over the Air update of the application firmware. The new firmware is downloaded via BLE to the STM32WB55 device and installed securely with the SBSFU bootloader.
2. Make sure aws\_demos are configured for OTA demo in config\_files\aws\_demo\_config.h in CubeIDE (or vendors\st\boards\p\_nucleo\_wb55\aws\_demos\config\_files\aws\_demo\_config.h):

```
#define CONFIG_OTA_UPDATE_DEMO_ENABLED
```

[Note: In addition, all the other CONFIG\_\*\_ENABLED flags are commented.]

3. Make sure you have configured the aws\_demos (especially the code signing certificate in demos\include\aws\_ota\_codesigner\_certificate.h).
4. To create the code-signing certificate refer below link.  
<https://docs.aws.amazon.com/freertos/latest/userguide/ota-code-sign-cert-win.html>
5. For OTA: Replace “signingcredentialSIGNING\_CERTIFICATE\_PEM” variable value in demos\include\aws\_ota\_codesigner\_certificate.h with created certificate content (beginning with “-----BEGIN CERTIFICATE-----”);
6. Then compile the STM32WB55RG\_Nucleo\_AFR\_demo project as mentioned in the section 4.1 and Flash the binary on the STM32WB55 board as mentioned in the section 4.2

#### 7.1.1 OTA Firmware Version upgrade

1. OTA update process, Firmware version needs to be upgrade in the STM32WB55 code and generate new \*.sfb file to upload it on AWS S3 bucket. Detailed steps are mentioned below:
2. Presuming that the current version of the application is flashed to the STM32WB55 board. Now upgrade the firmware version in demos/include/aws\_application\_version.h as shown below:

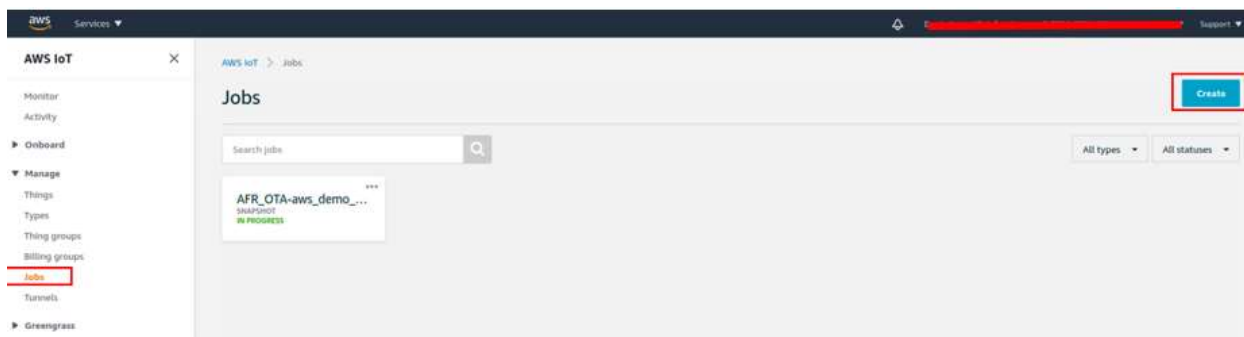
```
26 #ifndef _AWS_APPLICATION_VERSION_H_
27 #define _AWS_APPLICATION_VERSION_H_
28
29 #include "iot_appversion32.h"
30 extern const AppVersion32_t xAppFirmwareVersion;
31
32 #define APP_VERSION_MAJOR    0
33 #define APP_VERSION_MINOR    9
34 #define APP_VERSION_BUILD    2
35
36 #endif
```

3. Now compile it again the STM32WB55RG\_Nucleo\_AFR\_demo project as mentioned in the section 4.1 and Don't Flash the binary on the STM32WB55 board

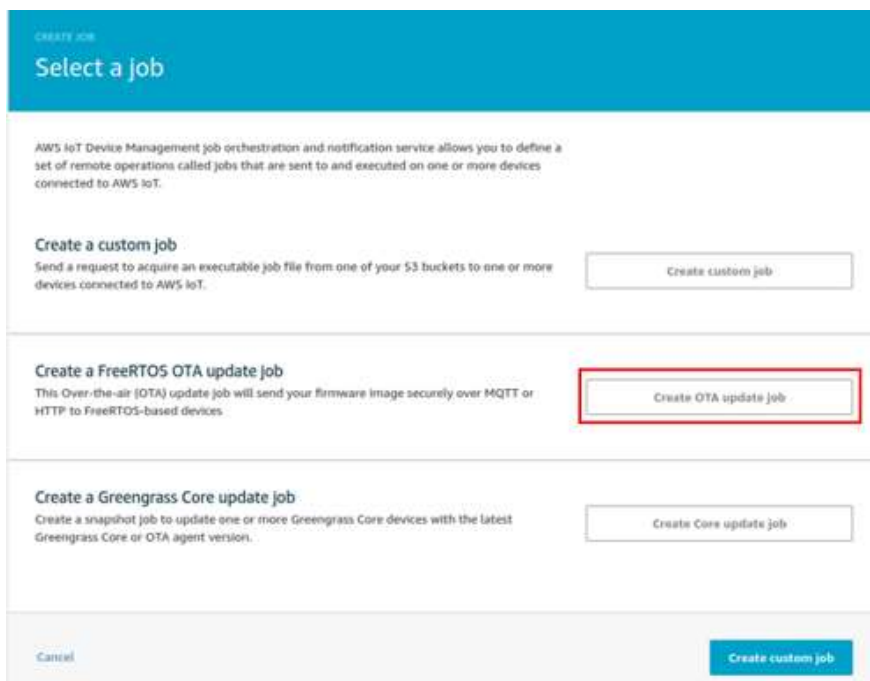
- Please collect firmware from under `build\aws_demos.sfb` file. Keep this firmware binary on the AWS S3 bucket (step mentioned in [section 2.3](#))
- The `aws_demos` project must be running on STM32WB55.Nucleo board and be connected to the mobile application.

### 7.1.2 AWS configuration for OTA Job

- On AWS console, create a FreeRTOS OTA update job (IoT Core / Manage / Jobs / Create):



- Select the **create OTA update job**



- Select your “thing name” `ST32WB55` and press **Next**.

**Create a FreeRTOS OTA update job**

This Over-the-air (OTA) update job will send your firmware image securely over MQTT or HTTP to FreeRTOS-based devices.

**Select devices to update**

Browse and select the devices you want to include in this job.

1 thing(s) and 0 thing group(s) selected. [Close](#)

**Things** | Thing groups | Summary

☒ STM32WB55

☐ h\_Core

☐ LTE\_Thing

[Cancel](#) [Back](#) [Next](#)

4. MQTT protocol for image transfer, (HTTP is not supported)
5. Select Sign a new firmware for me option

**Create a FreeRTOS OTA update job**

**Select the protocol for firmware image transfer**

HTTP and MQTT protocols are supported for firmware updates. [Learn more](#)

☐ HTTP [Info](#)

☒ MQTT

**Select and sign your firmware image**

Code signing ensures that devices only run code published by trusted authors and that the code has not been altered or corrupted since it was signed. You have three options for code signing. [Learn more](#)

☒ Sign a new firmware image for me

☐ Select a previously signed firmware image

☐ Use my custom signed firmware image

Code signing profile: [Learn more](#)

stk\_code\_sign | SHA256 | ECDSA | ecdsa.ctf [Clear](#) [Change](#)

6. Create code signing profile, Enter **Profile Name** (e.g. ssk\_code\_sign) , Device Hardware Platform select **Windows Simulator**, code signing certificate press “**Import**” and [browser the created code signing certificate and private key](#) , Enter the pathname of code signing certificate on device (e.g. ecdsa.crt).

**Create a code signing profile**

Profile name  
ssk\_code\_sign

Device hardware platform  
Windows Simulator SHA256 ECDSA [Change](#)

Code signing certificate  
AWS Certificate Manager (ACM) handles the complexity of creating and managing or importing SSL/TLS certificates. You use ACM to create an ACM Certificate or import a third-party certificate that you use for signing. You must have a certificate to sign code.

No certificate selected [Close](#)

Select Certificate  
[Browse...](#) ecdsasigner.crt

Select Certificate private key  
[Browse...](#) ecdsasigner.key

Select Certificate chain (optional)  
[Browse...](#) No file selected.

[Import](#)

Pathname of code signing certificate on device  
This is the platform-specific location and name of the certificate used by the FreeRTOS device firmware to perform OTA image signature verification.  
ecdsa.crt

[Cancel](#) [Create](#)

7. Select the aws\_demos.sfb firmware file which was uploaded on S3 bucket, and enter the destination pathname (e.g. firmware.bin)
8. Select the IAM role for OTA update Job which was created in [section 2.3](#) and click on **Next**

CREATE JOB

Create a FreeRTOS OTA update job

Select the protocol for firmware image transfer

HTTP and MQTT protocols are supported for firmware updates. [Learn more](#)

☐ HTTP [Info](#)

☒ MQTT

Select and sign your firmware image

Code signing ensures that devices only run code published by trusted authors and that the code has not been altered or corrupted since it was signed. You have three options for code signing. [Learn more](#)

☒ Sign a new firmware image for me

☐ Select a previously signed firmware image

☐ Use my custom signed firmware image

Code signing profile [Learn more](#)

ssk\_code\_sign

SHA256

ECDSA

mqtt.ota

Clear

Change


Select your firmware image in S3 or upload it

aws\_demos.sfb

Change

Pathname of firmware image on device [Learn more](#)

firmware.bin

IAM role for OTA update job 

Choose a role which grants AWS IoT access to the S3, AWS IoT Jobs and AWS Code signing resources to create an OTA update job. [Learn more](#)

Role (requires S3 access)

iamOTAUpdateRole

Select

Cancel

Back

Next



9. Enter Unique Job ID (e.g. Update\_ver3) and Click on Create Button, Successful message displayed on TOP.

CREATE JOB

Create a FreeRTOS OTA update job

ID

Update\_ver3

Description (optional)

Give your job a helpful description

Job type

A job can run on the devices and/or groups selected, or remain open, and apply to devices later added to a group.

☒ Your job will complete after deploying to the selected devices/groups (snapshot)

☐ Your job will continue deploying to any devices added to the selected groups (continuous)

Job executions rollout configuration - optional

Specify how quickly devices will be notified of a pending job execution. [Info](#)

☒ Constant rate

☐ Exponential rate

Maximum per minute (1-1000)

1000

Job abort configuration - optional

Specify criteria that will abort your job automatically once triggered. [Info](#)

Failure type	Threshold percentage	Abort action	Minimum executed things
Select		Select	

Clear

Add criteria

Job executions timeout configuration - optional

Specify a timeout duration for your in-progress job executions. [Info](#)

No timeout duration configured

Enable

Tags

Apply tags to your resources to help organize and identify them. A tag consists of a case-sensitive key-value pair.

Tag name	Value
Provide a tag name, e.g. Manufacturer	Provide a tag value, e.g. Acme-Corporation

Clear

Add another

Cancel

Back

Create

Success

Successfully created job.

View Job

## 7.2 Run OTA Demo

1. If you want to modify the Endpoint URL, then follow the [step 2](#) otherwise Reset the board and jump to [step 3](#).
2. Press the **SW2 button** and Reset the board (By Pressing “Reset” button as shown in [figure 2](#)). It will ask to enter endpoint URL on Serial console screen as below.  
Paste the Endpoint URL here and press the “Enter” key on keyboard. (Note: Default Echo is OFF, so cannot see your Endpoint URL value.)  
So, to verify that the correct endpoint URL was entered, on the serial console it will display the URL as shown above (with yellow marked).

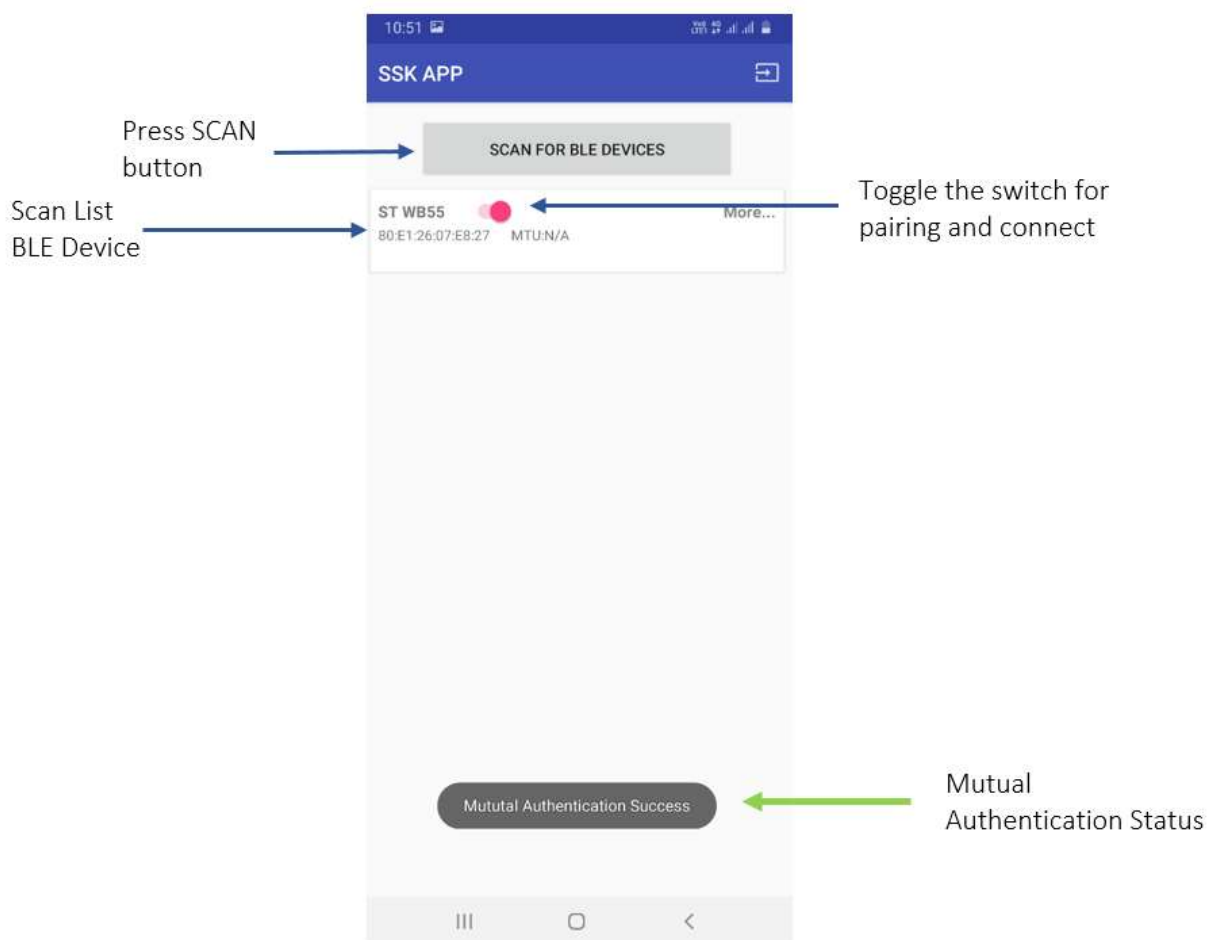
```

===== Please Enter Your Endpoint URL =====
e.g- xxxxxxxxxxx-ats.iot.xx-xxx-x.amazonaws.com

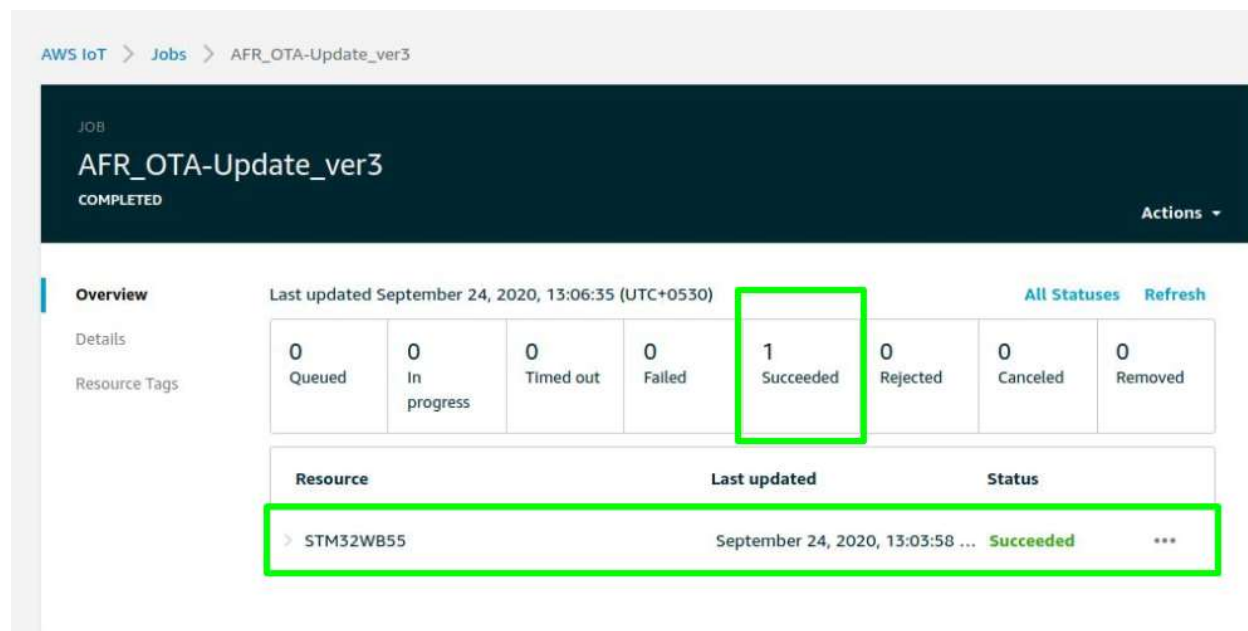
Entered Endpoint URL=2m5gdn06dwb-ats.iot.ap-south-1.amazonaws.com
=====

```

3. Click on the SCAN FOR BLE DEVICES button. Connect and Pair with STM32WB55 device. On the serial console to Press “Y” if passkey matches with mobile pair key.



- During this demo, **Mutual Authentication** is enabled between the Mobile Device and STM32WB55. Only **provisioning** device can start communication else Mutual Authentication fails and logs out from the AWS Cognito Session in Mobile APP.
- To see the OTA Job status on AWS console, Open **AWS Console >> IoT Service >> Manage >> Jobs >> Select created Job (i.e. AFR OTA-Update ver3)**



6. OTA Job and its packets shown in below screen

```

87 27516 [OTA Agent Task] (MQTT connection 0x20011b18, PUBLISH operation 0x200129e0) Wait complete with result SUCCESS.
88 27526 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [RequestingJob] Event [RequestJobDocument] New state [WaitingForJob]
89 27867 [RF_STACK][SECURITY][SEND] CIPHER TEXT = $o,$$X|||@
0 27874 [OTA Agent Task] [prvParseJobDoc] Size of OTA_FileContext_t [64]
1 27882 [OTA Agent Task] [prvParseJSONbyModel] Extracted parameter [ clientToken: 0:STM32WB55 ]
2 27891 [!ot_thread] State: RequestingJob Received: 1 Queued: 0 Processed: 0 Dropped: 0
3 27900 [OTA Agent Task] [prvParseJSONbyModel] Extracted parameter [ jobId: AFR_OTA-Update_ver3 ]
4 27910 [OTA Agent Task] [prvParseJSONbyModel] Extracted parameter [ protocols: ["MQTT"] ]
5 27918 [OTA Agent Task] [prvParseJSONbyModel] Extracted parameter [ streamname: AFR_OTA-Bdc48aaF-992b-4b17-91c9-4f5976324ffc ]
6 27930 [OTA Agent Task] [prvParseJSONbyModel] Extracted parameter [ filepath: firmware.bin ]
7 27938 [OTA Agent Task] [prvParseJSONbyModel] Extracted parameter [ filesize: 222048 ]
8 27947 [OTA Agent Task] [prvParseJSONbyModel] Extracted parameter [ fileId: 0 ]
9 27955 [OTA Agent Task] [prvParseJSONbyModel] Extracted parameter [ certfile: /home/certificate/ecdsasigner.crt ]
10 27966 [OTA Agent Task] [prvParseJSONbyModel] Extracted parameter [ sig_sha256_ecdsa: MEUCIQDNLtWNjRdG9mVWbmWS02rej... ]
11 27974 [OTA Agent Task] [prvParseJobDoc] Job was accepted successfully start transfer
12 27982 [!ot_thread] State: WaitingForJob Received: 1 Queued: 0 Processed: 0 Dropped: 0
13 29488 [OTA Agent Task] [prvSetDataInterface] Data Interface is set to MQTT.
14 29496 [OTA Agent Task] [prvProcessJobHandler] Setting OTA data Infaceer.
15 29503 [OTA Agent Task] [prvOTAAgentTask] called handler. Current State [WaitingForJob] Event [ReceivedJobDocument] New state [CreatingFile]
16 29524 [!ot_thread] [SECURITY][SEND] CIPHER TEXT = U||$ooXXooo ||$K||$||$ooooB{Yl+c ayoloJehEo$ ||$||$oooU{o6}o||$B|R|sooe(see)||$V&eeeePg
57 29538 [OTA Agent Task] (MQTT connection 0x20011b18) SUBSCRIBE operation scheduled.
58 29546 [OTA Agent Task] (MQTT connection 0x20011b18, SUBSCRIBE operation 0x20012fd8) Waiting for operation completion.
59 29908 [!ot_thread] State: WaitingForJob Received: 1 Queued: 0 Processed: 0 Dropped: 0
60 30140 [RF_STACK][SECURITY][RECEIVED] CIPHER TEXT = Rood$$$ *
60B
51 38149 [OTA Agent Task] (MQTT connection 0x20011b18) SUBSCRIBE operation 0x20012fd8) Wait complete with result SUCCESS.
52 38166 [OTA Agent Task] [prvInitTransferFrMqtg] OK: $aws/things/STM32WB55/stream/AFR_OT-A-Bdc48aaF-992b-4b17-91c9-4f5976324ffc/data/cbor
53 38173 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [CreatingFile] Event [CreateFile] New state [RequestingFileBlock]
54 38197 [!ot_thread] [SECURITY][SEND] CIPHER TEXT = U||$oot=eeEgw7H|$|.ooooibelo#||||$||$||$eq_sceooooooooHwzxBRCrJXeAeoZ?||$NRT0BN0d3 Pro-eS7e65 30211 [OTA Agent]
58) MQT Publish operation queued.
56 38219 [OTA Agent Task] [prvRequestFileBlock Mqgt] OK: $aws/things/STM32WB55/streams/AFR_OT-A-Bdc48aaF-992b-4b17-91c9-4f5976324ffc/get/cbor
57 38232 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [RequestingFileBlock] Event [RequestFileBlock] New state [WaitingForFileBlock]
58 38916 [!ot_thread] State: CreatingFile Received: 1 Queued: 0 Processed: 0 Dropped: 0
59 38189 [OTA Agent Task] [prvIngestDataBlock] Received file block 1, size 1024
60 31112 [OTA Agent Task] [prvPAL_WriteBlock] Write 1024 bytes at 1024 address
61 31119 [OTA Agent Task] [prvIngestDataBlock] Remaining: 216
62 31125 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [WaitingForFileBlock] Event [ReceivedFileBlock] New state [WaitingForFileBlock]
63 31674 [OTA Agent Task] [prvIngestDataBlock] Received file block 4, size 1024
64 31696 [OTA Agent Task] [prvPAL_WriteBlock] Write 1024 bytes at 4896 address
65 31704 [OTA Agent Task] [prvIngestDataBlock] Remaining: 215
66 31710 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [WaitingForFileBlock] Event [ReceivedFileBlock] New state [WaitingForFileBlock]
67 31924 [!ot_thread] State: WaitingForFileBlock Received: 3 Queued: 0 Processed: 0 Dropped: 0
68 32259 [OTA Agent Task] [prvIngestDataBlock] Received file block 2, size 1024
69 32281 [OTA Agent Task] [prvPAL_WriteBlock] Write 1024 bvtes at 2048 address.
```

7. Verify the received Firmware and its signature as shown in below screen

```

1065 151582 [OTA Agent Task] [prvPAL_WriteBlock] Write 864 bytes at 221184 address
1066 151590 [OTA Agent Task] [prvIngestDataBlock] Remaining: 1
1067 151596 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [WaitingForFileBlock] Event [ReceivedFileBlock] New state [WaitingForFileBlock]
1069 152104 [OTA Agent Task] [prvIngestDataBlock] Received file block 180, size 1024
1070 152127 [OTA Agent Task] [prvPAL_WriteBlock] Write 1024 bytes at 184320 address
1071 152135 [OTA Agent Task] [prvIngestDataBlock] Received final expected block of file.
1072 152143 [OTA Agent Task] [prvStopRequestTimer] Stopping request timer.
1073 153022 [!ot_thread] State: WaitingForFileBlock Received: 218 Queued: 0 Processed: 0 Dropped: 0
1074 153304 [OTA Agent Task] [prvIngestDataBlock] File receive complete and signature is valid.
1075 153312 [OTA Agent Task] [prvStopRequestTimer] Stopping request timer.
1076 153320 [OTA Agent Task] [prvUpdateJobStatus_Mqtt] Msg: {"status": "IN_PROGRESS", "statusDetails": {"self_test": "ready", "updatedBy": "0x90002"}}
1077 153345 [!ot_thread] SECURITYLEVEL_CHANGED TEXT: [0x00000000] [0x00000000] [0x00000000]

```

8. On Next reboot Board will power up with newly upgraded firmware

```
1346 [iot_thread] [INFO ][DEMO][lu] Successfully initialized the demo. Network type for the demo: 2
1355 [iot_thread] [INFO ][MQTT][lu] MQTT library successfully initialized.
1362 [iot_thread] OTA demo version 0.9.3
1366 [iot_thread] Creating MQTT Client...
```

## 8 REFERENCES

- [1] <https://aws.amazon.com/freertos/>
- [2] <https://github.com/Infineon/OPTIGA™-trust-m>
- [3] <https://www.infineon.com/cms/en/product/security-smart-card-solutions/OPTIGA™-embedded-security-solutions/OPTIGA™-trust/OPTIGA™-trust-m-sls32aia/>
- [4] <https://www.st.com/en/evaluation-tools/p-nucleo-wb55.html>