

Developer Guide

Security Starter Kit with STM32WB55 and OPTIGA™ Trust M

Date: March 03, 2021 | Version 1.1



The Solutions People



CONTENTS

1	INTRODUCTION.....	4
1.1	Purpose of the Document	4
1.2	Architecture	4
1.3	SSK Suit Package Contents.....	4
2	SETTING UP THE DEVELOPMENT ENVIRONMENT	5
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
3	INTEGRATION OF AWS FREERTOS AND OPTIGA™ TRUST M.....	14
3.1	AWS Sources	14
3.2	OPTIGA™ Trust M Source	14
3.3	Mutual Authentication Source	14
4	CODE COMPILATION AND BINARY FLASHING	17
4.1	Code Compilation Steps.....	17
4.2	Binary flashing.....	20
5	SETUP AWS IOT THING ON AWS CONSOLE.....	23
6	MQTT DEMO	27
6.1	Setup AWS config in the STM32WB55 Source Code	27
6.2	Run MQTT Demo	28
7	OTA DEMO	31
7.1	Setup AWS config in the STM32WB55 Source Code	31
7.1.1	OTA Firmware Version upgrade	32
7.1.2	AWS configuration for OTA Job.....	33
7.2	Run OTA Demo	38
8	REFERENCES.....	41

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 covered, 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 - [Security-Starter-Kits-STM32WB55-SSK.zip](#) from the Arrow Electronics GitHub portal
 - Extract the [Security-Starter-Kits-STM32WB55-SSK.zip](#)
 - Extracting the zip file, one will find the below contents:
 - [STM32WB55-SSK Developers Guide.pdf](#)
 - [STM32WB55-SSK Quick Start Guide.pdf](#)
 - Firmware_Images
 - Source_Code
 - SSK_Cert_And_Config
 - Andriod-APP.zip
 - IOS_APP.url

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.

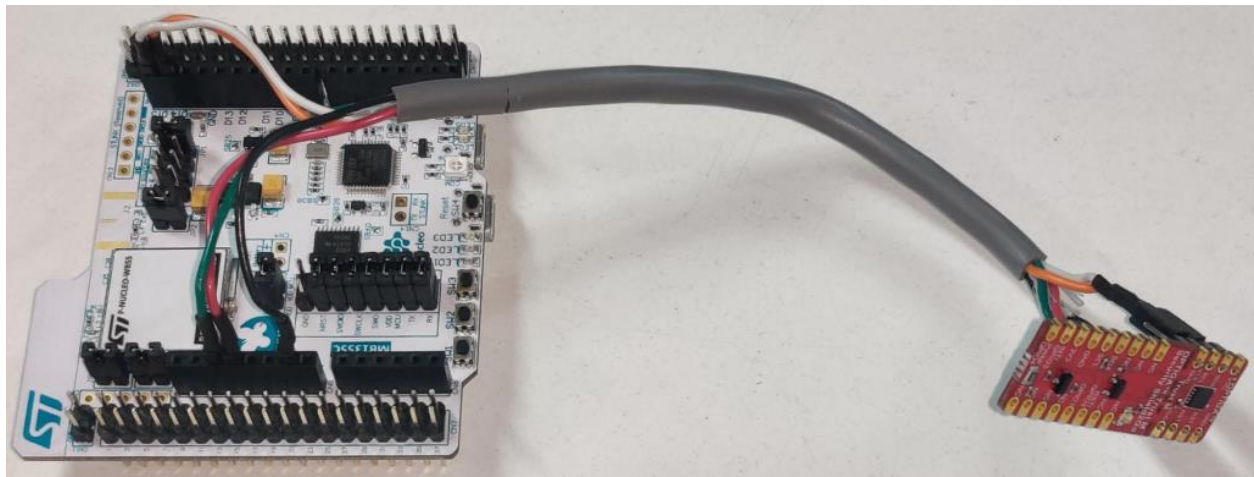
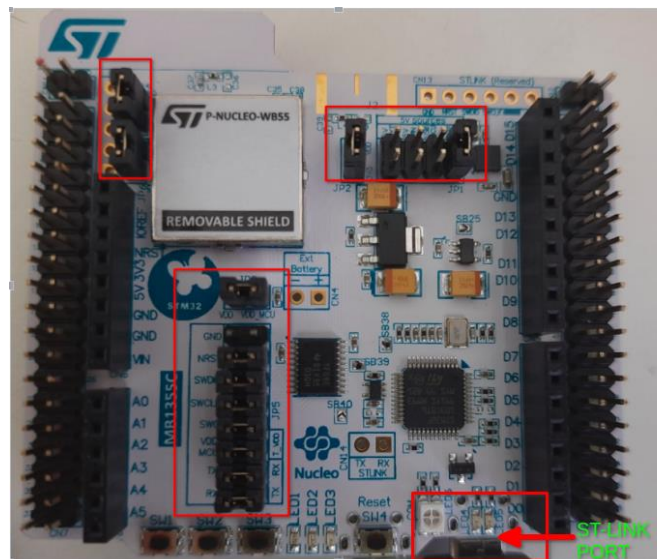


Figure 2: Hardware Setup

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 ST-LINK (port) as show in figure to power on the board and for debugging purposes.

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 certificates 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

A 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 codes are available under Source_Code/Mobile_App_Source.zip]

2.3 OTA configuration in AWS

To perform OTA, user will have to configure the below listed items in the [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.

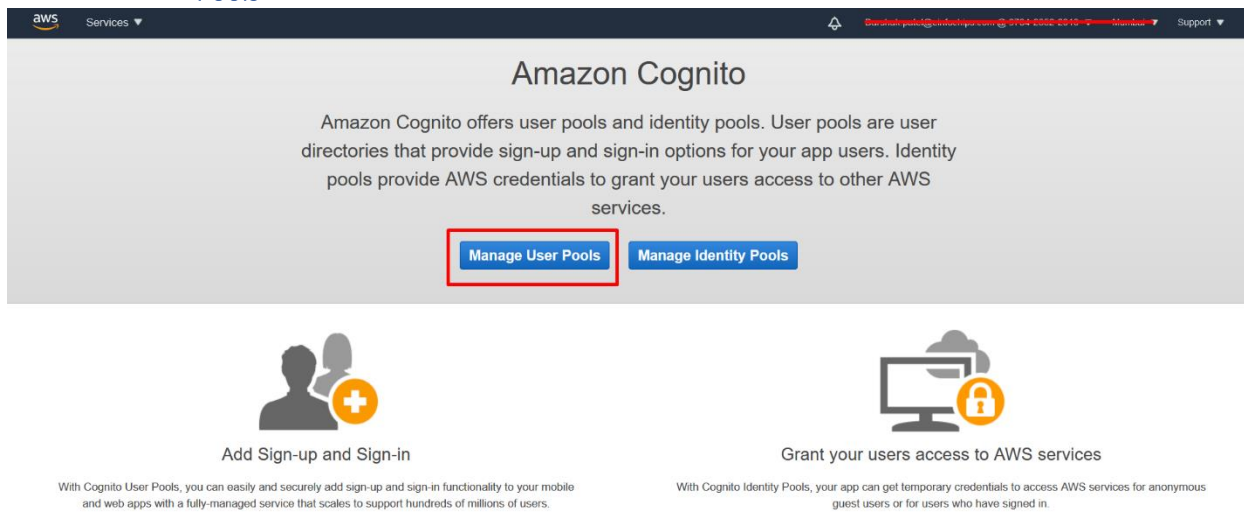
Detailed configuration and steps available on an AWS document here:

<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/>

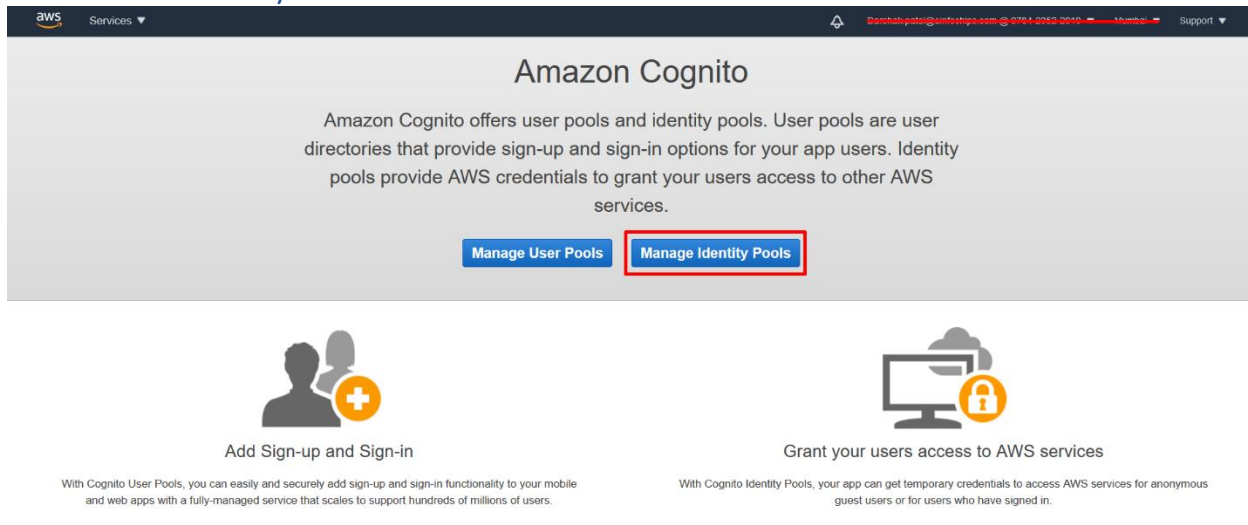
1. If you have already created the Amazon Cognito user pool in the AWS console then go to step 6, otherwise continue with step 2.
2. To create an Amazon Cognito user pool in AWS Console:
 - In AWS console, open the **Amazon Cognito** console (Service), and choose **Manage User Pools**.



- 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

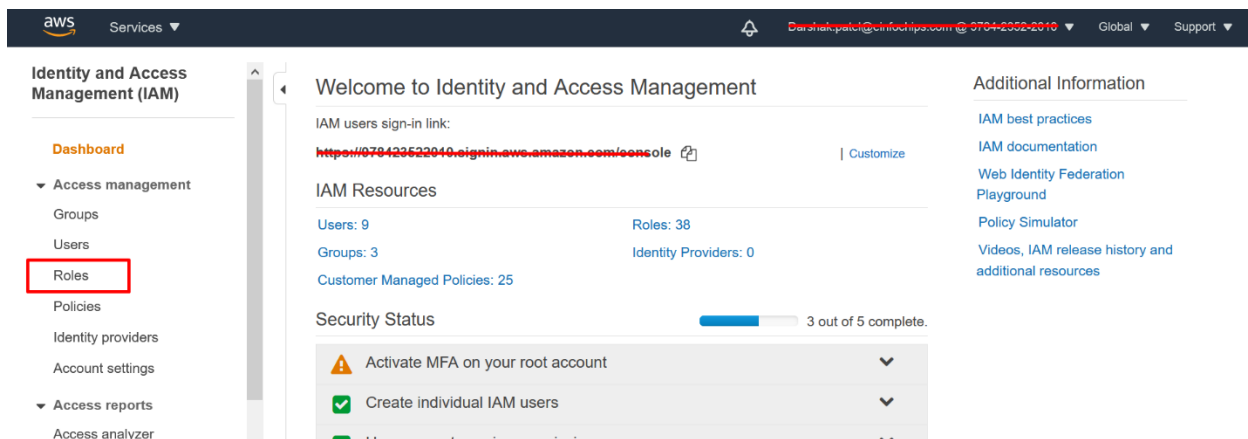
- In AWS Console, Open the Amazon Cognito console (Service), and choose **Manage Identity Pools**.



- 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 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 the created Username and Password**, which is used to login into the Mobile app.

User Pools | Federated Identities

stm32wb55_pool

General settings

Users and groups

Attributes

Policies

MFA and verifications

Advanced security

Message customizations

Tags

Devices

App clients

Triggers

Analytics

App integration

App client settings

Users

Groups

Import users

Create user

User name

Search for value

Username	Enabled	Account status	Email verified	Phone number verified
alpesh	Enabled	CONFIRMED	true	false
darshak	Enabled	CONFIRMED	true	true

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 text file with the name **AWS_Config.txt** and enter the parameters:

```
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 it:
<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*
SecurWSJ_policy

AWS Cognito Identity PoolId*
ap-south-1:a2786f1b-fc21-4a-988b-209b8c1

AWS Cognito User PoolId*
ap-south-1:31a9a9a1-PR

AWS Cognito User AppClientId*
30g7m31a9a9a1-21933tan

AWS Cognito User AppClientSecret*
1hv8f1n5t31a9a9a1-6pvh59

SAVE

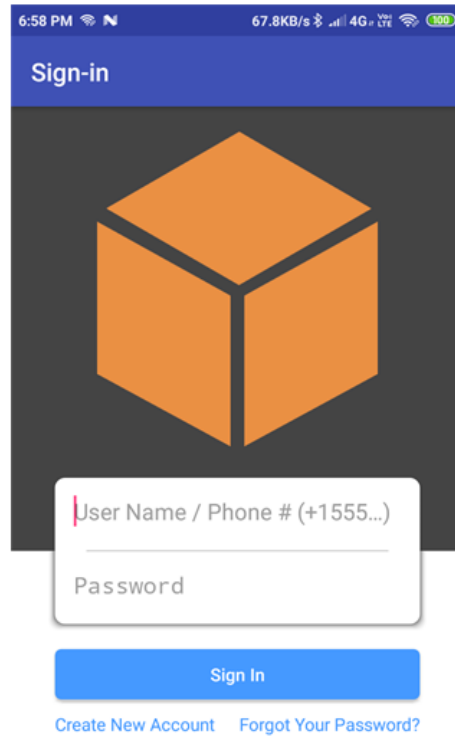
Press Skip or reconfigure if required

SSK APP

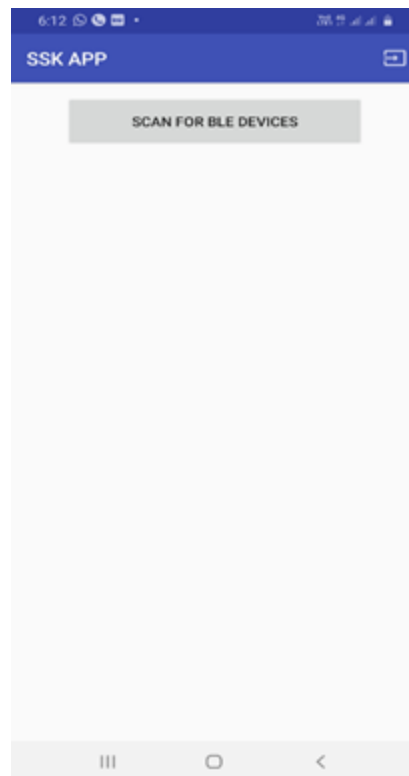
SKIP

BROWSE FOR COGNITO DETAILS

10. Login in the App using the Cognito user credentials:



11. After Successful Login, you should see the 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. Users 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 the provided package, available at: <https://github.com/Infineon/OPTIGA-trust-m>

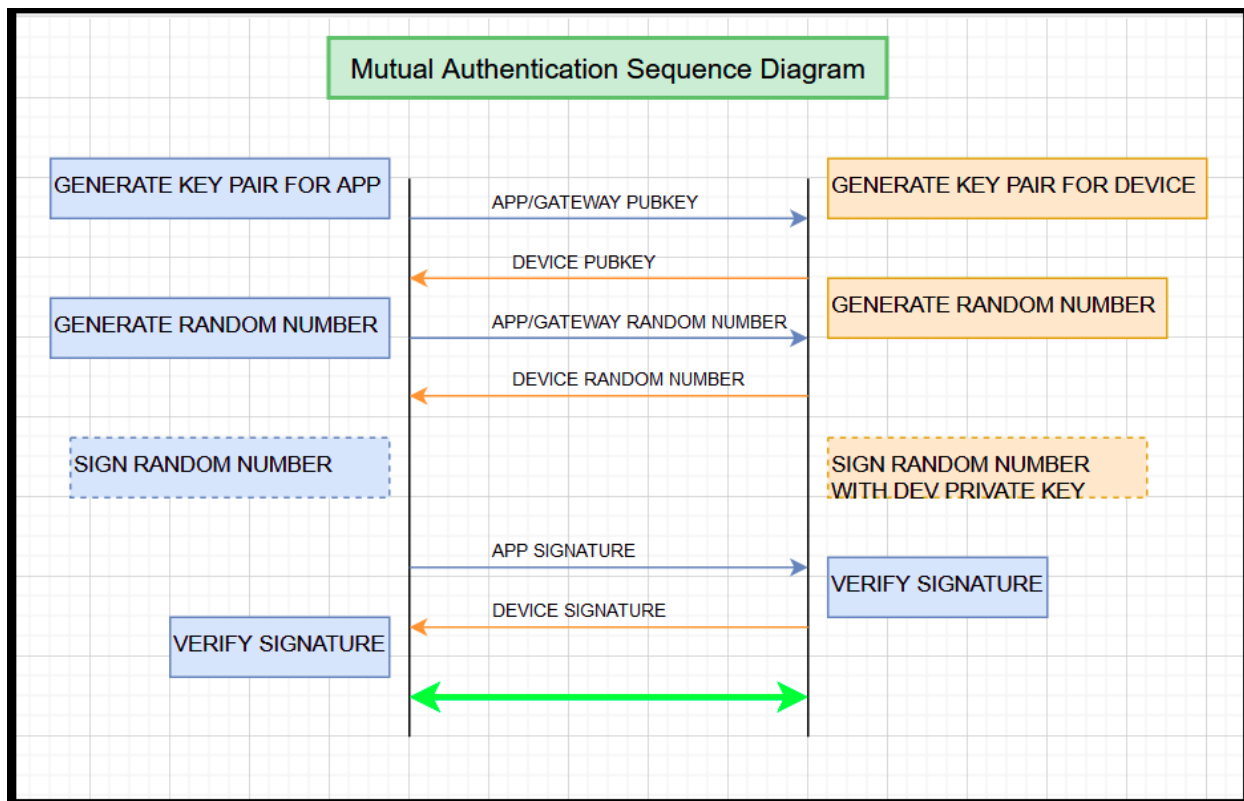
To port software framework templates, the files shown below must be updated based on your Hardware platform.

1. `pal_ifx_i2c_config.c`
2. `pal_i2c.c`
3. `pal_gpio.c`
4. `pal_os_timer.c`
5. `pal_os_event.c`
- 6.

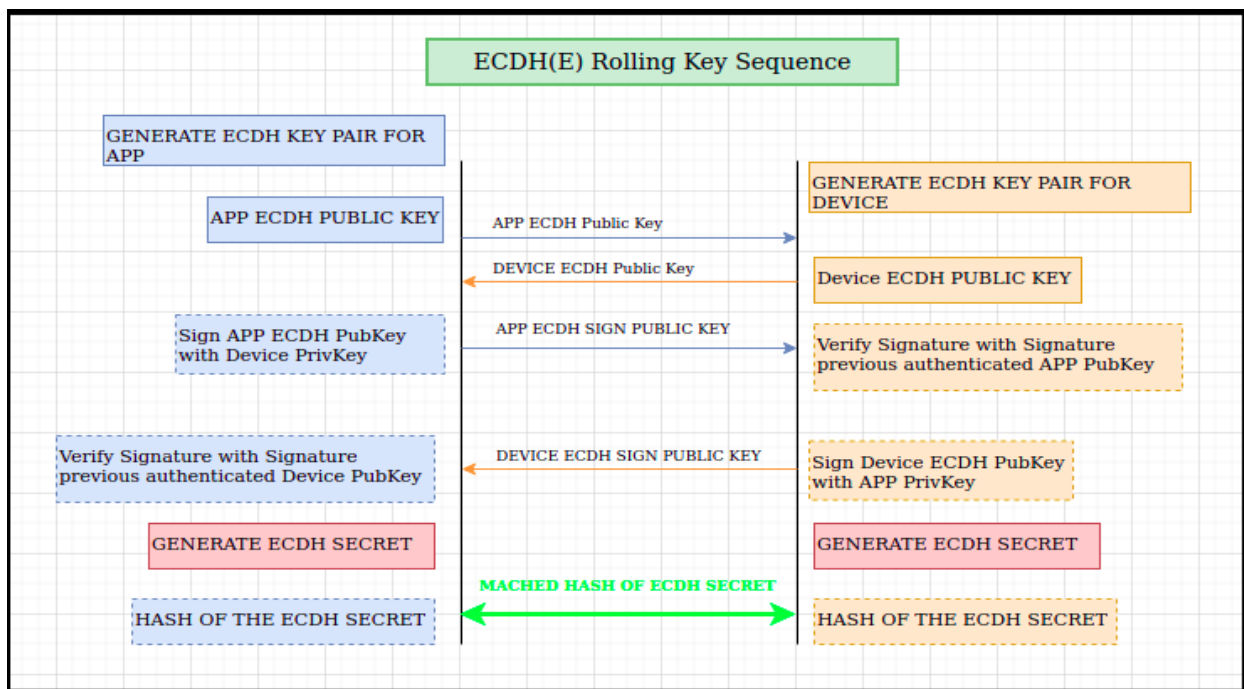
For reference, please refer to: <https://github.com/Infineon/OPTIGA-trust-m/wiki/Porting-Guide>

3.3 Mutual Authentication Source

Mutual Authentication is the device authentication process, which must be executed before an Application begins to communicate with the 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 an AES 256 symmetric key to encrypt and decrypt all communications.

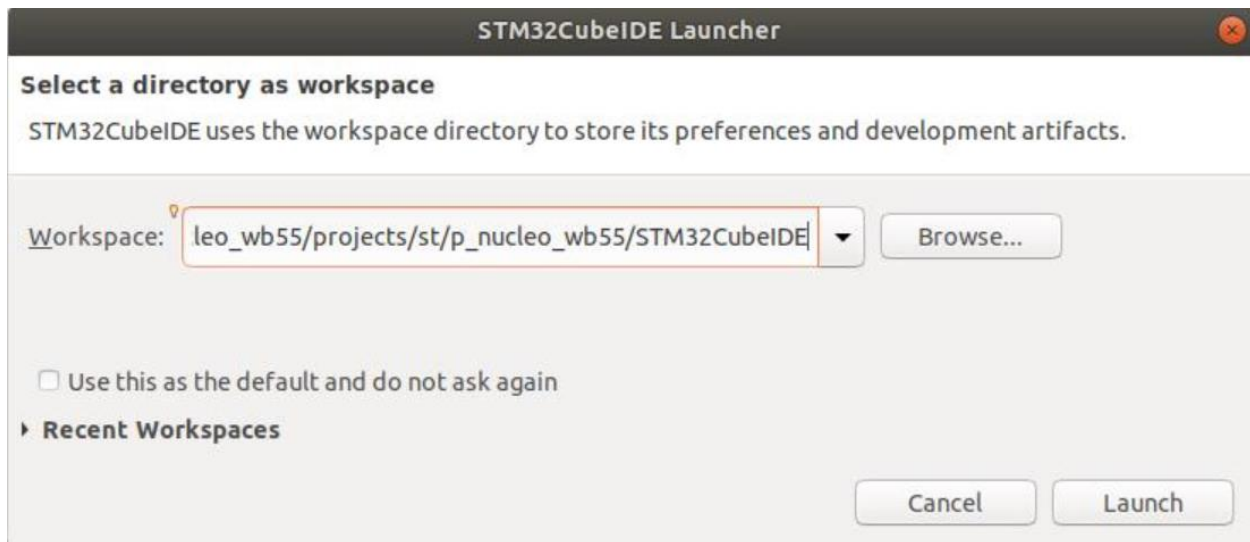


- The SSK uses OPTIGA™ Trust M security chip to store the secrets e.g.: certificates, keys, etc. Also, it uses the hardware crypto library provided by OPTIGA™ Trust M software stack.
- To implement mutual authentication, add one custom BLE service in STM32WB55 AWS source code available at:
[SSK_SUIT_EVAL_STM32WB55_Rel_\[Release\]/Source_code/STM32_SSK_AWS_Rel_\[version\]/demos/ble/aws_ble_mutual_auth.c](#)

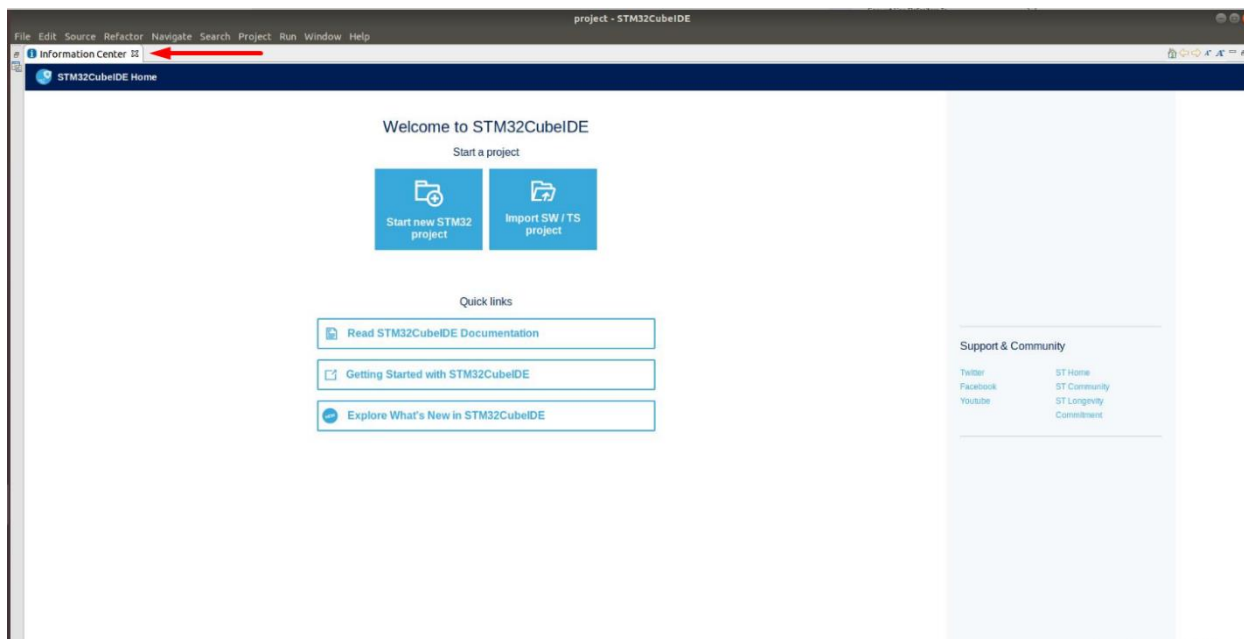
4 CODE COMPILATION AND BINARY FLASHING

4.1 Code Compilation Steps

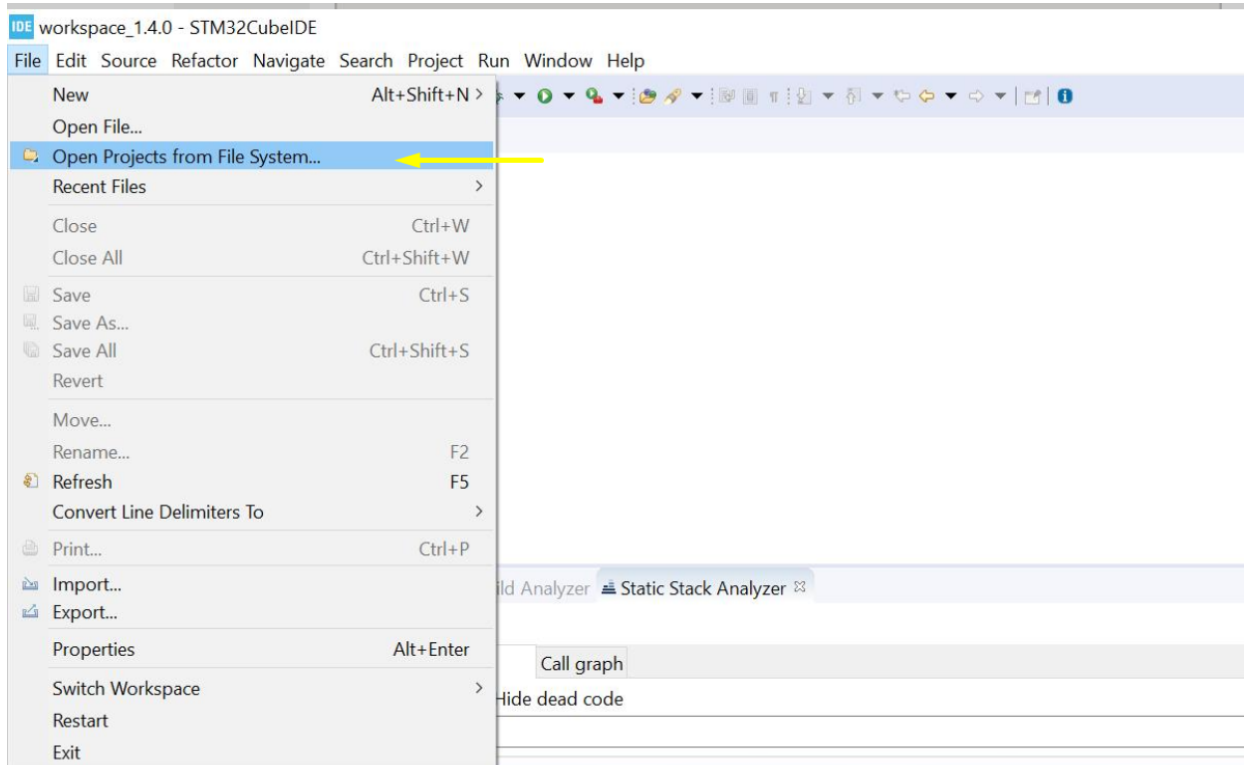
1. Download and extract the SSK STM32WB55 release package (STM32WB55_SSK_Pkg_Rel.zip) from the Arrow Electronics Github portal:
<https://github.com/ArrowElectronics/Security-Starter-Kits>
2. In the extracted directory, the project is available under:
[Source_Code/STM32_SSK_AWS_Rel_\[version\]/projects/st/p_nucleo_wb55/STM32CubeIDE](#)
3. Launch the STM32CubeIDE application.
4. Open a workspace under [projects\st\p_nucleo_wb55\STM32CubeIDE](#)



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

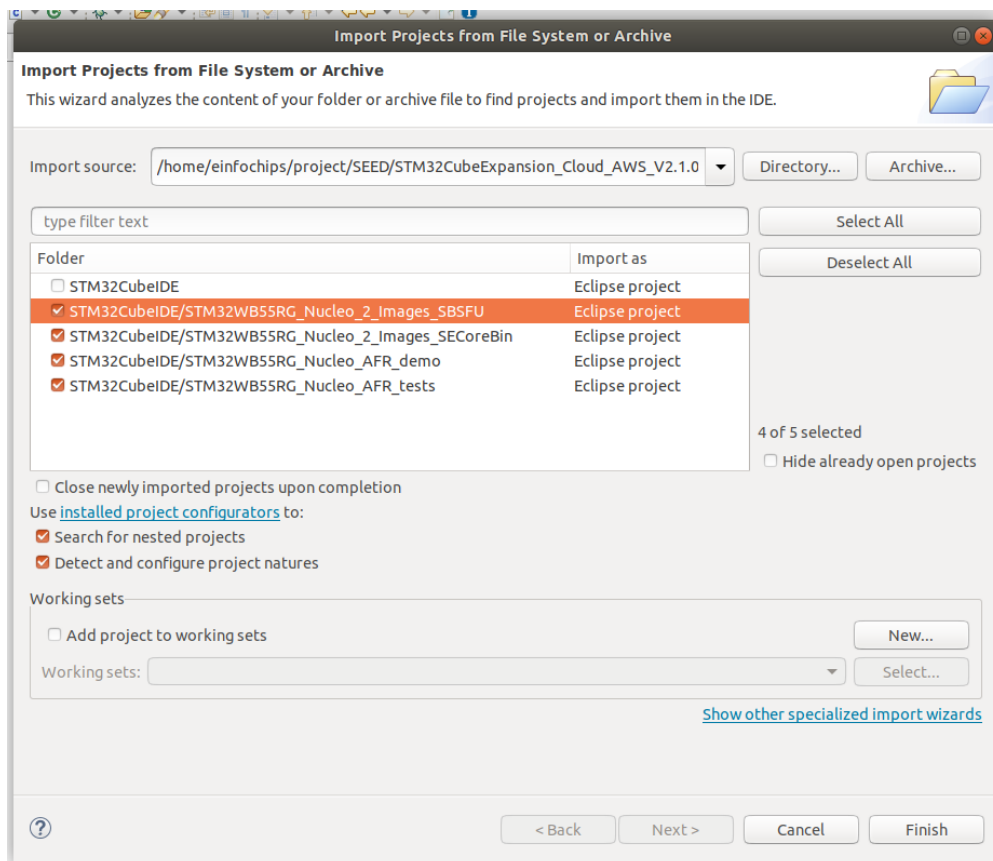


6. Select menu **File >> Open project from File system... >> Finish**



7. In "Select directory" browse to **projects/st/ p_nucleo_wb55/STM32CubeIDE**

8. Import the four projects:
 - STM32WB55RG_Nucleo_2_Images_SECoreBin
 - STM32WB55RG_Nucleo_2_Images_SBSFU
 - STM32WB55RG_Nucleo_AFR_demo
 - STM32WB55RG_Nucleo_AFR_tests



9. For Compilation, you need to follow this sequence:
10. Compile **SECoreBin** first, then **SBSFU**, and then **AFR_demo** to avoid build conflicts. With the console open 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.

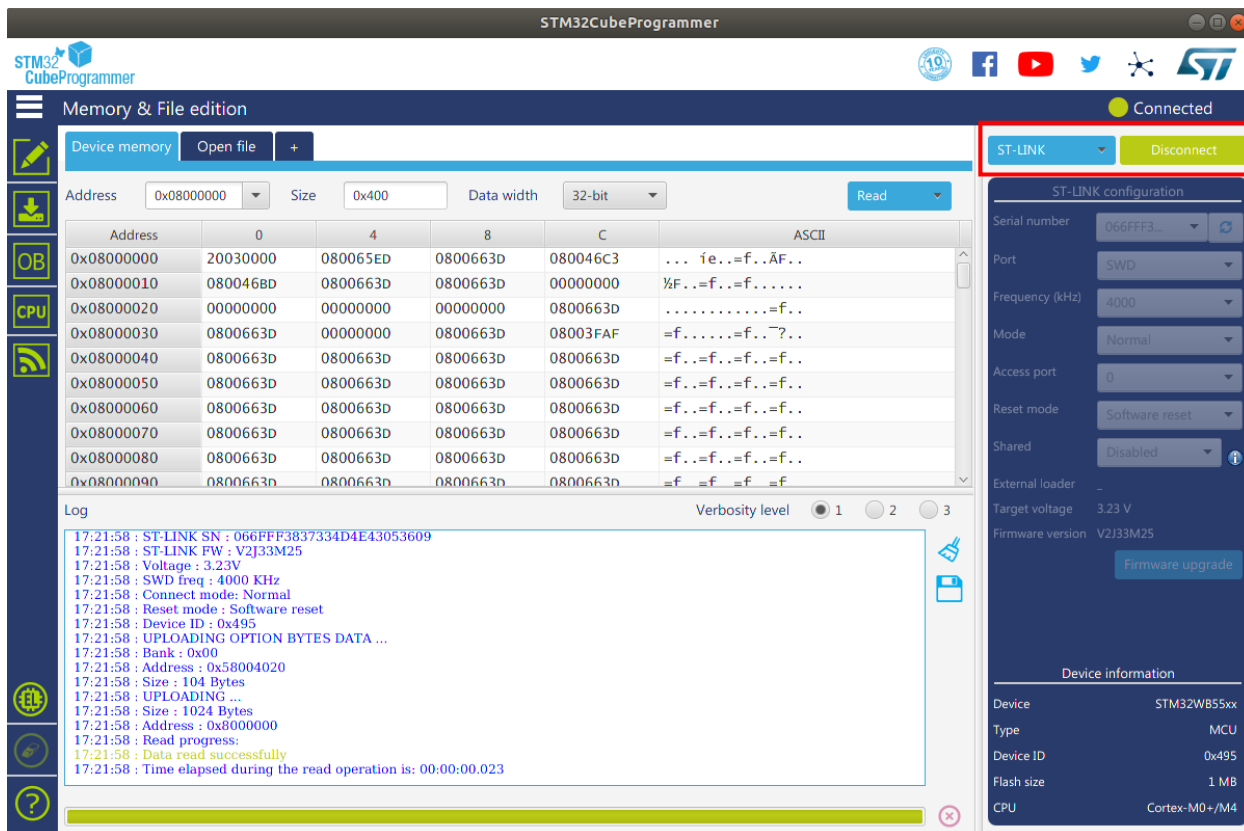
10:54:35 Build Finished. 0 errors, 0 warnings. (took 5s.531ms)

```

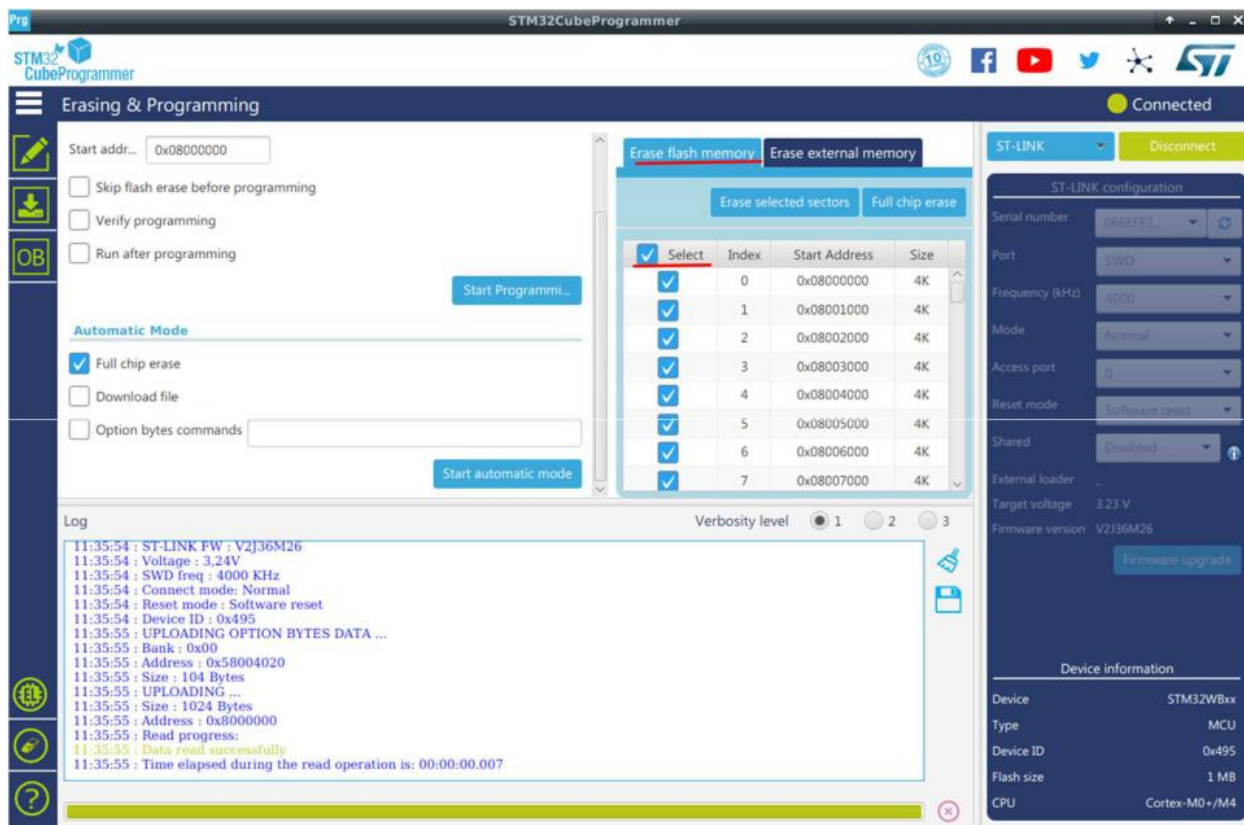
4.2 Binary flashing

Final binaries are located in [Source_Code/STM32_SSK_AWS_Rel_\[version\]/build](#) directory.

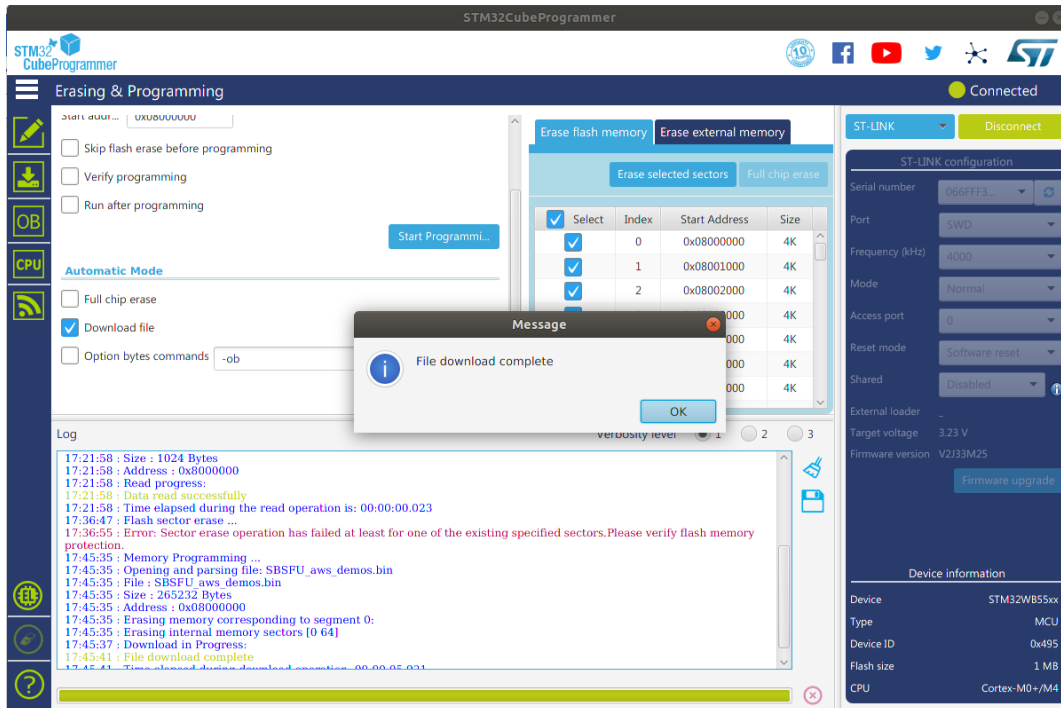
- SBSFU_aws_demos.bin - bootloader + demo application
 - aws_demos.sfb - demo application + OTA update header
1. Plug a USB cable from PC to the STM32WB55 board (Board backside port named "ST-LINK")
 2. Open STM32CubeProgrammer and click on connect button to connect the board as shown below:



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

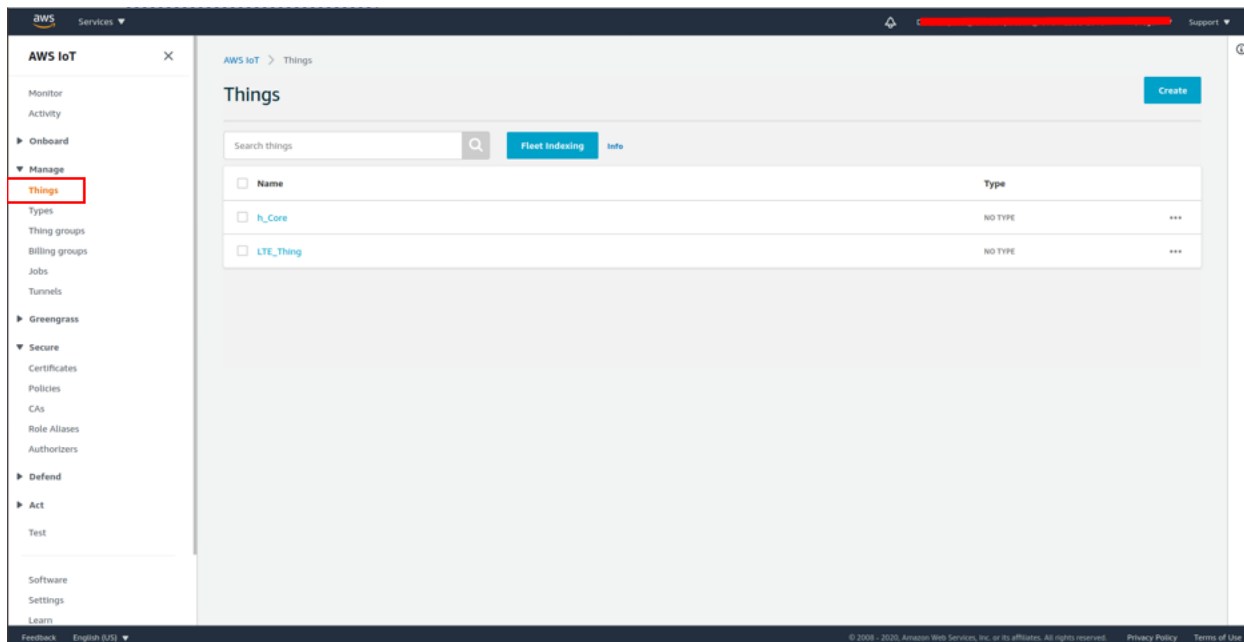


4. Flash the binary (e.g. `build\SBSFU_aws_demos.bin`) using STM32CubeProgrammer (address `0x08000000`).
5. After successfully flashing the Binary, you will see a Popup message- “File Download complete”. Press OK

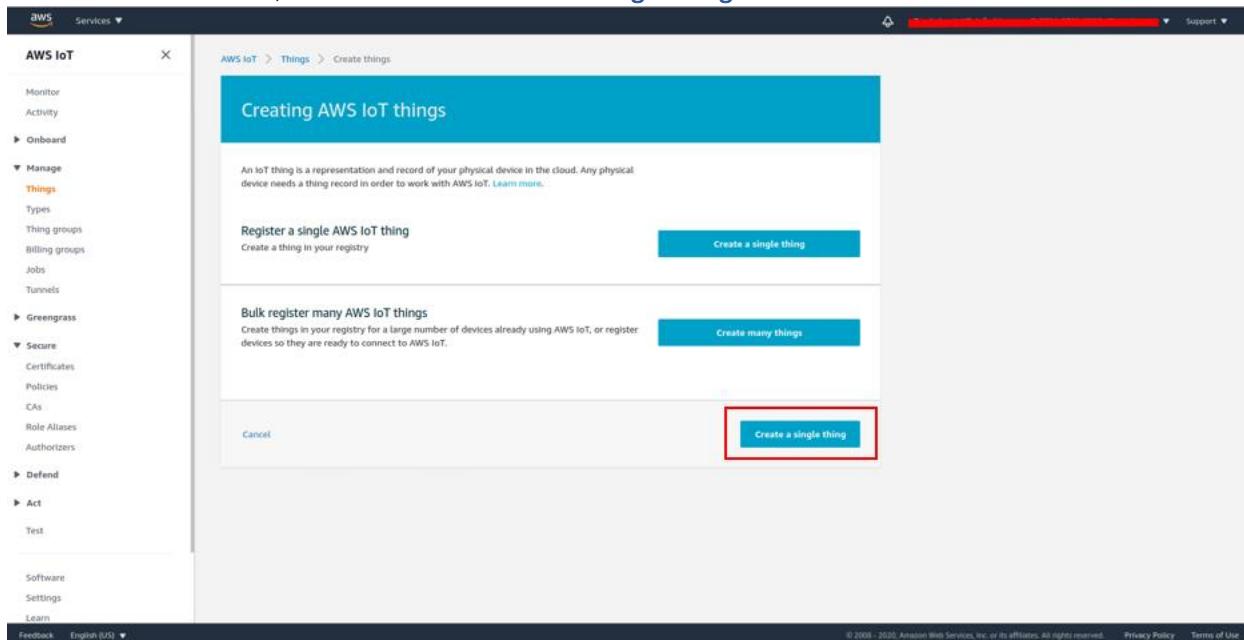


5 SETUP AWS IOT THING ON AWS CONSOLE

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**.

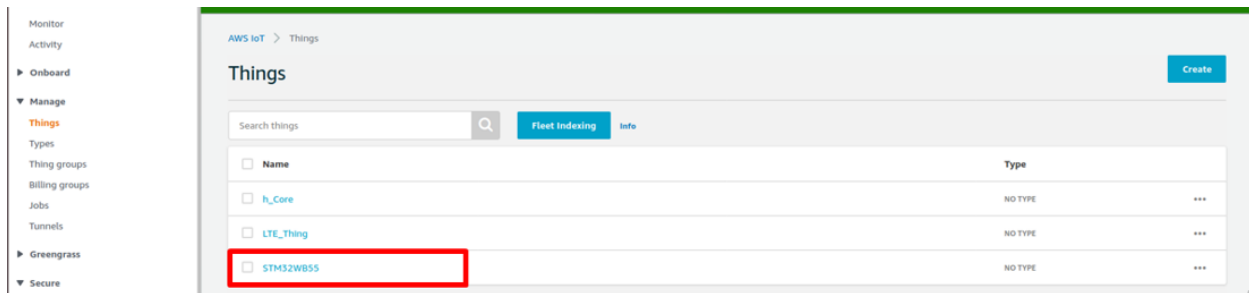
The screenshot shows the AWS IoT console interface for adding a device. The left sidebar contains navigation options like Monitor, Activity, Onboard, Manage, Things, Types, Thing groups, Billing groups, Jobs, Tunnels, Greengrass, Secure, Certificates, Policies, CAs, Role Aliases, Authorizers, Defend, Act, Test, Software, Settings, Learn, and Documentation. The main content area is titled 'Add your device to the thing registry' and includes sections for: 'This step creates an entry in the thing registry and a thing shadow for your device.' with a 'Name' field containing 'STM32WB55'; 'Apply a type to this thing' with a 'Thing Type' dropdown set to 'No type selected'; 'Add this thing to a group' with a 'Thing Group' dropdown; and 'Set searchable thing attributes (optional)' with fields for 'Attribute key' and 'Value'. At the bottom, there are 'Cancel', 'Back', and 'Next' buttons, with the 'Next' button highlighted by a red box.

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.

The screenshot shows the AWS IoT console interface for adding a certificate. The left sidebar is the same as the previous screenshot. The main content area is titled 'Add a certificate for your thing' and includes a description: 'A certificate is used to authenticate your device's connection to AWS IoT.' Below this are four options: 'One-click certificate creation (recommended)' with a 'Create certificate' button; 'Create with CSR' with a 'Create with CSR' button; 'Use my certificate' with a 'Get started' button; and 'Skip certificate and create thing' with a 'Create thing without certificate' button, which is highlighted by a red box.

The screenshot shows the AWS IoT console's 'Settings' page. On the left is a navigation sidebar with options like Monitor, Activity, Onboard, Manage, Greengrass, Secure, Defend, Act, Test, Software, Settings (highlighted), Learn, and Documentation. The main area has three settings cards. The first card, 'Custom endpoint', is 'ENABLED'. It explains that this allows connecting to AWS IoT via a REST API and provides the endpoint URL: 'ats.iot.us-east-1.amazonaws.com', which is highlighted by a red rectangle. The second card, 'Logs', is also 'ENABLED'. It states that enabling AWS IoT logs sends messages to CloudWatch Logs and lists the log name as 'ds_log'. Below this, it shows the 'Level of verbosity' set to 'Debug' with an 'Edit' button. The third card, 'Event-based messages', is 'DISABLED'. At the bottom of the page are links for Feedback, English (US), and copyright information from 2008-2020 Amazon Web Services, Inc.

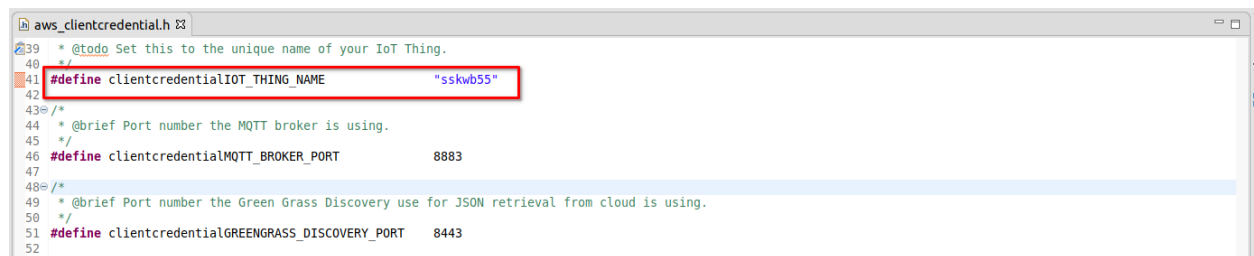
6. In the left navigation pane, choose **Manage >> Things**. Your device should have an AWS IoT thing name. **Record this name.**



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).

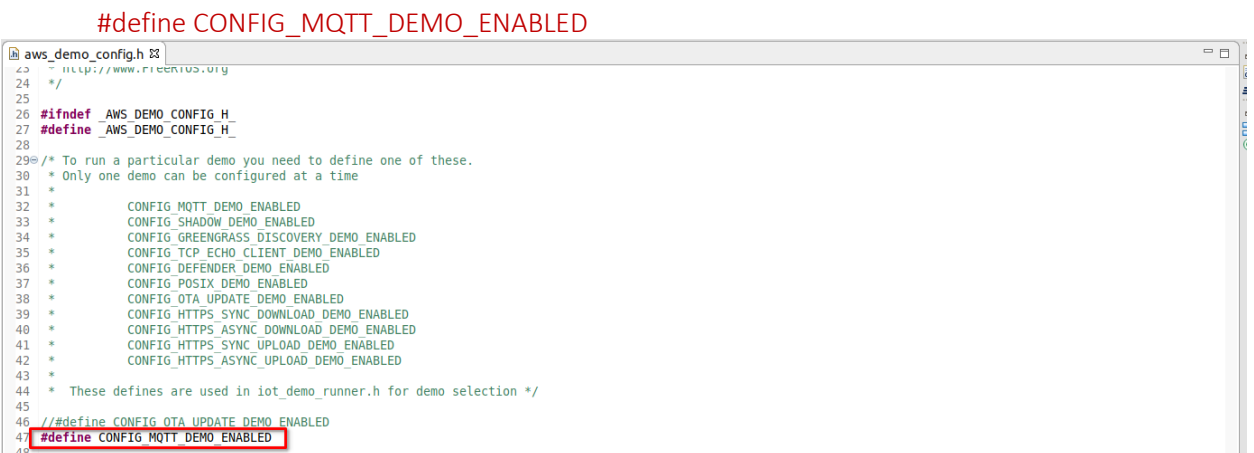


```

39  * @todo Set this to the unique name of your IoT Thing.
40  */
41  #define clientcredentialIOT_THING_NAME "sskwb55"
42
43  /*
44  * @brief Port number the MQTT broker is using.
45  */
46  #define clientcredentialMQTT_BROKER_PORT 8883
47
48  /*
49  * @brief Port number the Green Grass Discovery use for JSON retrieval from cloud is using.
50  */
51  #define clientcredentialGREENGRASS_DISCOVERY_PORT 8443
52

```

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

24  /*
25  */
26  #ifndef AWS_DEMO_CONFIG_H
27  #define AWS_DEMO_CONFIG_H
28
29  /* To run a particular demo you need to define one of these.
30  * Only one demo can be configured at a time
31  */
32  *
33  * CONFIG MQTT DEMO ENABLED
34  * CONFIG SHADOW DEMO ENABLED
35  * CONFIG GREENGRASS_DISCOVERY DEMO ENABLED
36  * CONFIG TCP ECHO CLIENT DEMO ENABLED
37  * CONFIG DEFENDER DEMO ENABLED
38  * CONFIG POSIX DEMO ENABLED
39  * CONFIG OTA UPDATE DEMO ENABLED
40  * CONFIG HTTPS_SYNC_DOWNLOAD DEMO ENABLED
41  * CONFIG HTTPS_ASYNC_DOWNLOAD DEMO ENABLED
42  * CONFIG HTTPS_SYNC_UPLOAD DEMO ENABLED
43  * CONFIG HTTPS_ASYNC_UPLOAD DEMO ENABLED
44  * These defines are used in iot_demo_runner.h for demo selection */
45
46  // #define CONFIG OTA UPDATE DEMO ENABLED
47  #define CONFIG MQTT DEMO ENABLED
48

```

[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” [SW4] button as shown in figure 2). It will ask the user to enter an endpoint URL in the 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 below (with yellow redactions):

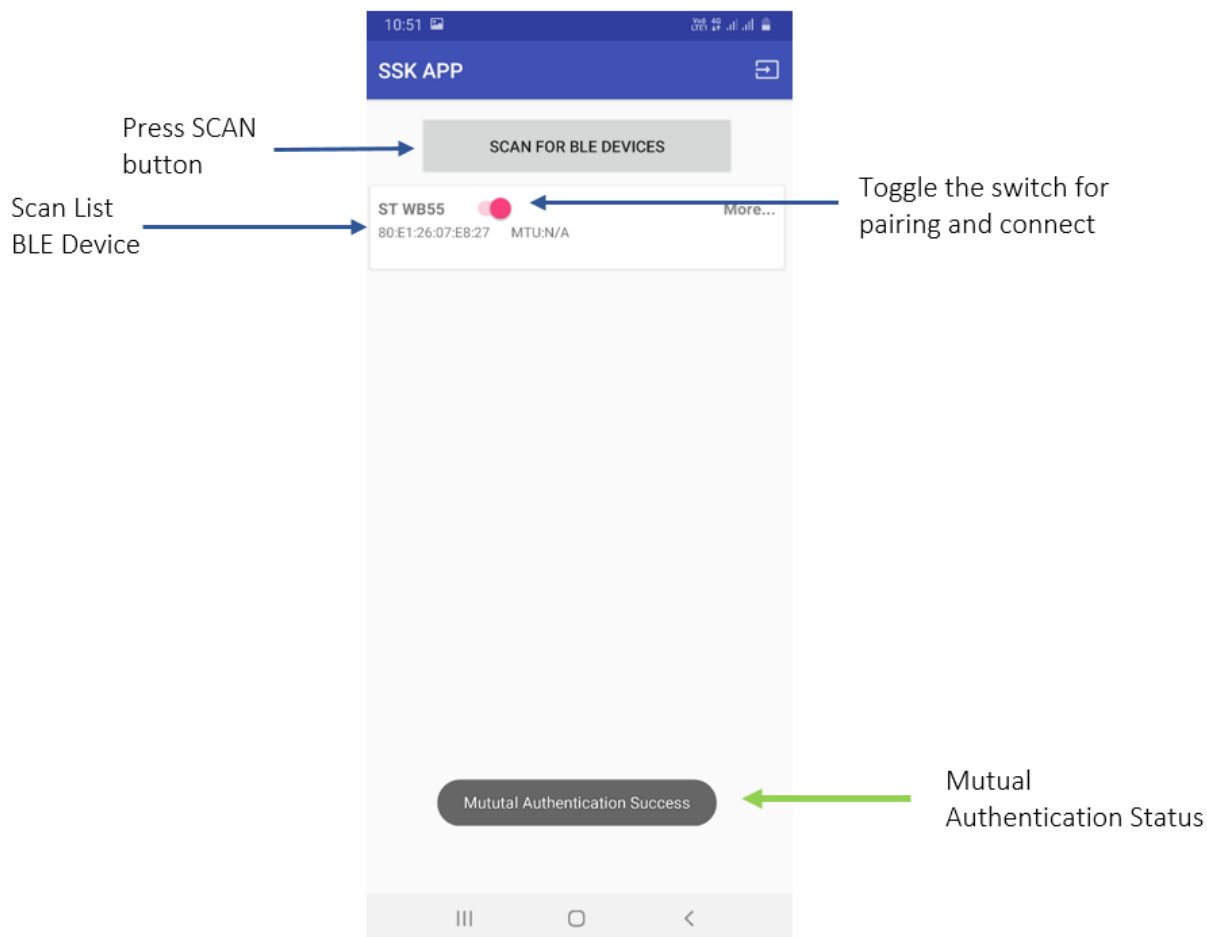
```

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

Entered Endpoint URL=a2u6gdn06dwb-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.



- During this demo, **Mutual Authentication** is enabled between the Mobile Device and STM32WB55. Only an Authenticated device can start communication, otherwise Mutual Authentication fails and logs out from the AWS Cognito Session in the Mobile app.
- 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/# ✕

iotdemo/topic/# Export Clear Pause

Publish

Specify a topic and a message to publish with a QoS of 0.

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 beYeeXeg;;/Ref*****s*****ooc#=#3EoUeeAec|eesDv,eeN 0Iee0t ha?((
43 16557 [iot_thread] [SECURITY][SEND] CIPHER TEXT = eVecJWH:Q#
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(eeGeeekenS+Q J+T+MM
48 16601 [iot_thread] (MQTT connection 0x200116b8) MQTT PUBLISH operation queued.
49 16609 [iot_thread] Plain Text=BLE MQTT Message Count 3!
Zedee*ee}ee?edee! [SECURITY][SEND] CIPHER TEXT = eVecJW_8ee-e|eUe'eePeeNeQee

```

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

aws_demo_config.h
23 // http://www.freertos.org
24 */
25
26 #ifndef AWS_DEMO_CONFIG_H
27 #define AWS_DEMO_CONFIG_H
28
29 /* To run a particular demo you need to define one of these.
30 * Only one demo can be configured at a time
31 *
32 * CONFIG_MQTT_DEMO_ENABLED
33 * CONFIG_SHADOW_DEMO_ENABLED
34 * CONFIG_GREENGRASS_DISCOVERY_DEMO_ENABLED
35 * CONFIG_TCP_ECHO_CLIENT_DEMO_ENABLED
36 * CONFIG_DEFENDER_DEMO_ENABLED
37 * CONFIG_POSIX_DEMO_ENABLED
38 * CONFIG_OTA_UPDATE_DEMO_ENABLED
39 * CONFIG_HTTPS_SYNC_DOWNLOAD_DEMO_ENABLED
40 * CONFIG_HTTPS_ASYNC_DOWNLOAD_DEMO_ENABLED
41 * CONFIG_HTTPS_SYNC_UPLOAD_DEMO_ENABLED
42 * CONFIG_HTTPS_ASYNC_UPLOAD_DEMO_ENABLED
43 *
44 * These defines are used in iot demo runner.h for demo selection */
45
46 #define CONFIG_OTA_UPDATE_DEMO_ENABLED
47 // #define CONFIG_MQTT_DEMO_ENABLED
48
```

[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 to the below link:
<https://docs.aws.amazon.com/freertos/latest/userguide/ota-code-sign-cert-win.html>

- For OTA: Replace “`signingcredentialSIGNING_CERTIFICATE_PEM`” variable value in `demos\include\aws_ota_codesigner_certificate.h` with created certificate content (beginning with “-----BEGIN CERTIFICATE-----”):

```

11 *
12 * The above copyright notice and this permission notice shall be included in all
13 * copies or substantial portions of the Software.
14 *
15 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
16 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
17 * FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
18 * COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
19 * IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
20 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
21 *
22 * http://aws.amazon.com/freertos
23 * http://www.FreeRTOS.org
24 */
25
26 #ifndef AWS_CODESIGN_KEYS_H
27 #define AWS_CODESIGN_KEYS_H
28
29 /*
30 * PEM-encoded code signer certificate
31 *
32 * Must include the PEM header and footer:
33 * "-----BEGIN CERTIFICATE-----\n"
34 * "...base64 data...\n"
35 * "-----END CERTIFICATE-----\n";
36 */
37 static const char signingcredentialSIGNING_CERTIFICATE_PEM[] =
38 "-----BEGIN CERTIFICATE-----\n"
39 "MIICWwYDQYDQYDQDASzZWYKOGFycm93LmNvbTAeFw0yMDA5MDYwNjQ5NDhBaFw0OTEx\n"
40 "MDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5\n"
41 "MDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5\n"
42 "MDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5\n"
43 "MDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5\n"
44 "MDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5\n"
45 "MDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5\n"
46 "MDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5\n"
47 "MDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5\n"
48 "MDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5\n"
49 "MDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5NDhBaFw0OTExMDYwNjQ5\n"

```

- Then compile the `STM32WB55RG_Nucleo_AFR_demo` project as mentioned in [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

- This section covers the OTA update process, Firmware version needs to be upgraded in the STM32WB55 code, and generating new *.sfb files to upload to an AWS S3 bucket. Detailed steps are mentioned below:
- 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

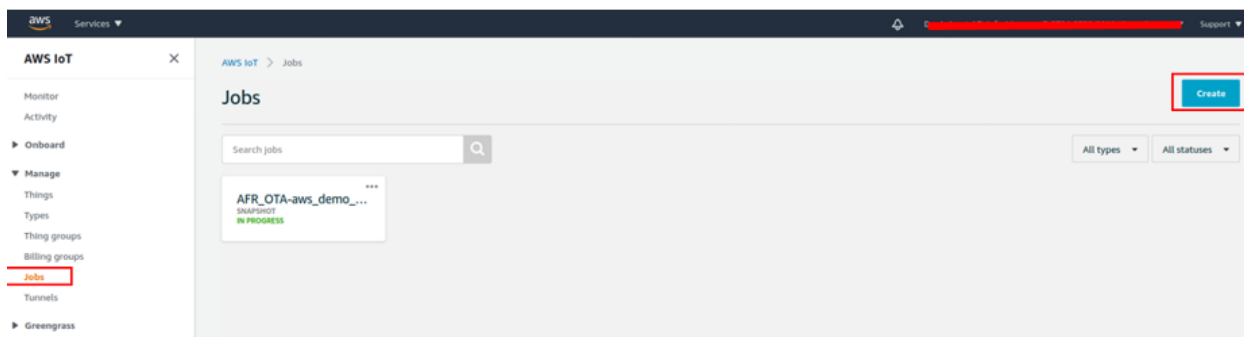
```

- Now compile the `STM32WB55RG_Nucleo_AFR_demo` project as mentioned in [section 4.1](#) but **Do not Flash** the binary on the STM32WB55 board.

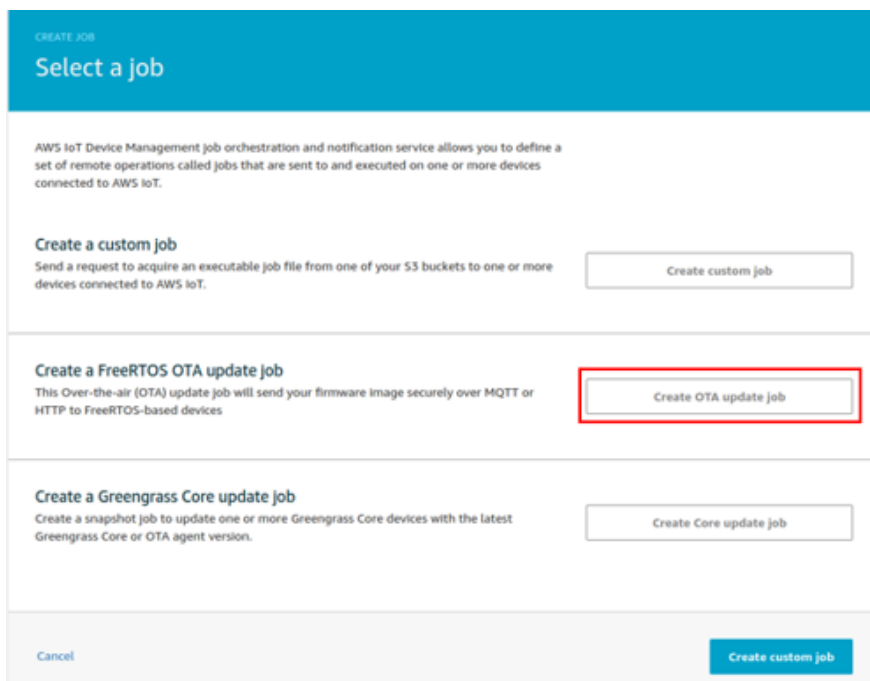
- Please collect firmware from the [build\aws_demos.sfb](#). Keep this firmware binary on the AWS S3 bucket (step mentioned in [section 2.3](#))
- The aws_demos project must be running on STM32WB55 Nucleo board and be connected to the mobile application.

7.1.2 AWS configuration for OTA Job

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



- Select '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. Select 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](#)

ssk_code_sign SHA256 ECDSA ecdsa.crt [Clear](#) [Change](#)

6. Create a code signing profile, enter a **Profile Name** (e.g. ssk_code_sign), for Device Hardware Platform select **Windows Simulator**, code signing certificate press “**Import**” and [browse to the created code signing certificate and private key](#). Enter the path name of the 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 ecdsa.crt [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. User should see a 'Success' message displayed on the top of the page.

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

aws Services

AWS IoT

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” [SW4] 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 below (with yellow redactions).

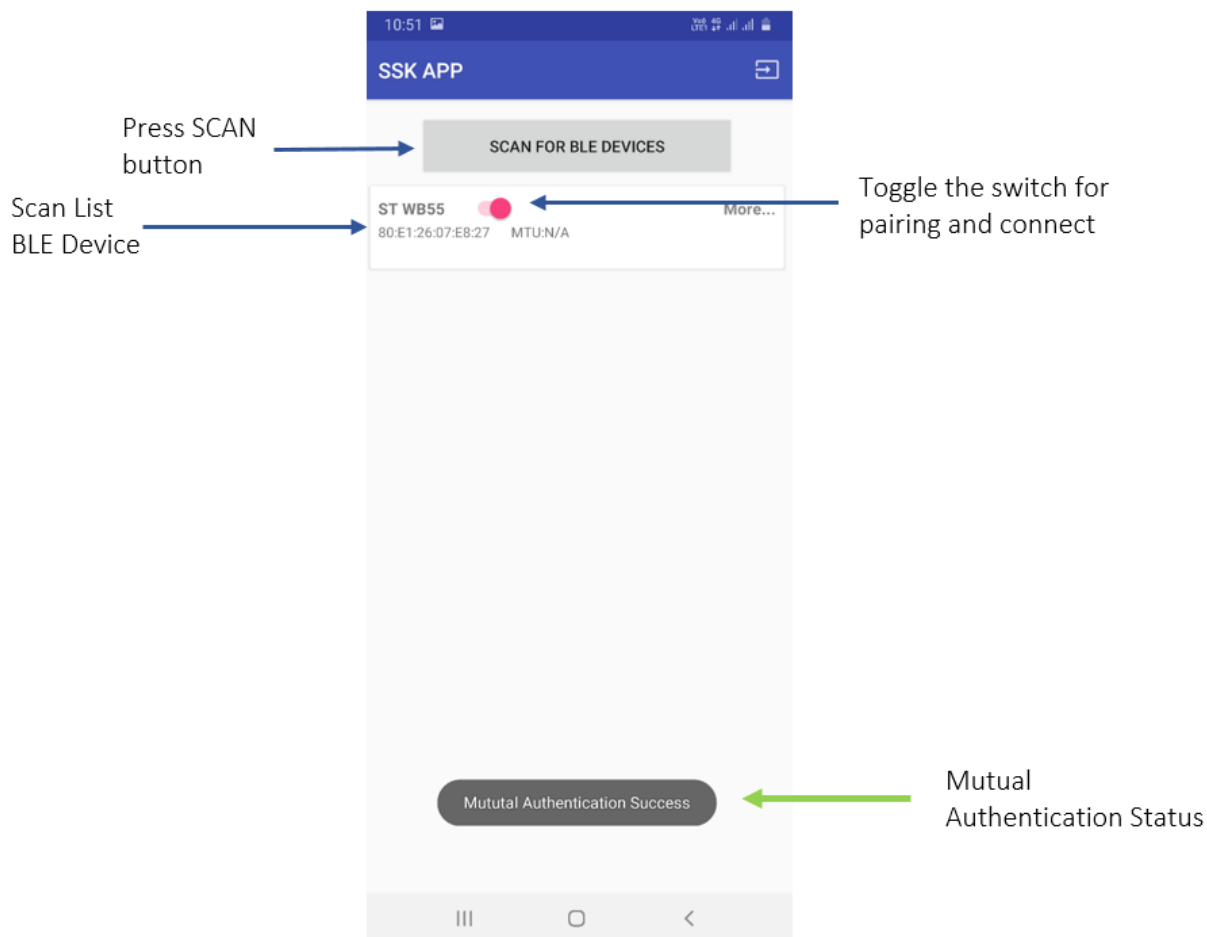
```

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

Entered Endpoint URL=a2u6gdn06dwb-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.



4. During this demo, **Mutual Authentication** is enabled between the Mobile Device and STM32WB55. Only Authenticated devices can start communication, otherwise Mutual Authentication fails and logs out from the AWS Cognito Session in Mobile app.
5. 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)**

AWS IoT > Jobs > AFR_OTA-Update_ver3

JOB
AFR_OTA-Update_ver3
COMPLETED Actions ▾

Overview
Details
Resource Tags

Last updated September 24, 2020, 13:06:35 (UTC+0530) All Statuses Refresh

0 Queued	0 In progress	0 Timed out	0 Failed	1 Succeeded	0 Rejected	0 Canceled	0 Removed
-------------	------------------	----------------	-------------	----------------	---------------	---------------	--------------

Resource	Last updated	Status
> STM32WB55	September 24, 2020, 13:03:58 ...	Succeeded ***

6. OTA Job and its packets shown in below screen capture:

```

57 27516 [OTA Agent Task] [MQTT connection 0x20011b18, PUBLISH operation 0x200129e0] Wait complete with result SUCCESS.
58 27526 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [RequestingJob] Event [RequestJobDocument] New state [WaitingForJob]
59 27867 [RF_STACK] [SECURITY][SEND] CIPHER TEXT = 50,0030 x[redacted]@

0 27874 [OTA Agent Task] [prvParseJobDoc] Size of OTA_FileContext_t [64]
1 27882 [OTA Agent Task] [prvParseJSONbyModel] Extracted parameter [ clientToken: 0:STM32MB55 ]
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 27918 [OTA Agent Task] [prvParseJSONbyModel] Extracted parameter [ protocols: ['MQTT'] ]
5 27926 [OTA Agent Task] [prvParseJSONbyModel] Extracted parameter [ streamname: AFR_OTA-8bc48aaf-992b-4b17-91c9-4f5976324ffc ]
6 27935 [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 [ filetype: 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: MEUCIQLNTVNrD7g9mwYQmBwS02rej... ]
11 27977 [OTA Agent Task] [prvParseJobDoc] Job was accepted. Attempting to start transfer.

52 28908 [!ot_thread] State: WaitingForJob Received: 1 Queued: 0 Processed: 0 Dropped: 0
53 28918 [OTA Agent Task] [prvOTADataInterface] Data interface is set to MQTT.
54 29496 [OTA Agent Task] [prvProcessJobHandler] Setting OTA data interface.
55 29503 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [WaitingForJob] Event [ReceivedJobDocument] New state [CreatingFile]
56 29524 [!ot_thread] [SECURITY][SEND] CIPHER TEXT = U[redacted]00t=00Eg7H[redacted].00001b00[redacted]#0[redacted]0110000_0ce00000[redacted]Hw=008K0R[XA0000Z[redacted]0eNRT0B0B3 PAr=05765 30211 [OTA Agent Task]

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 = Re00500 0
60B

18) MQTT PUBLISH operation queued.
56 30149 [OTA Agent Task] [MQTT connection 0x20011b18, SUBSCRIBE operation 0x20012fd8] Wait complete with result SUCCESS.
56 30160 [OTA Agent Task] [prvInitFileTransfer_Mqtt] OK: $aws/things/STM32MB55/streams/AFR_OTA-8bc48aaf-992b-4b17-91c9-4f5976324ffc/data/cbor
56 30173 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [CreatingFile] Event [CreateFile] New state [RequestingFileBlock]
56 30197 [!ot_thread] [SECURITY][SEND] CIPHER TEXT = U[redacted]00t=00Eg7H[redacted].00001b00[redacted]#0[redacted]0110000_0ce00000[redacted]Hw=008K0R[XA0000Z[redacted]0eNRT0B0B3 PAr=05765 30211 [OTA Agent Task]

18) MQTT PUBLISH operation queued.
56 30219 [OTA Agent Task] [prvRequestFileBlock_Mqtt] OK: $aws/things/STM32MB55/streams/AFR_OTA-8bc48aaf-992b-4b17-91c9-4f5976324ffc/get/cbor
56 30232 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [RequestingFileBlock] Event [RequestingFileBlock] New state [WaitingForFileBlock]
56 30249 [!ot_thread] State: [WaitingForFileBlock] Received: 1 Queued: 0 Processed: 0 Dropped: 0
59 31089 [OTA Agent Task] [prvIngestDataBlock] Received file block 1, size 1024
70 31112 [OTA Agent Task] [prvPAL_WriteBlock] Write 1024 bytes at 1024 address
71 31119 [OTA Agent Task] [prvIngestDataBlock] Remaining: 216
72 31125 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [WaitingForFileBlock] Event [ReceivedFileBlock] New state [WaitingForFileBlock]
73 31674 [OTA Agent Task] [prvIngestDataBlock] Received file block 4, size 1024
74 31690 [OTA Agent Task] [prvPAL_WriteBlock] Write 1024 bytes at 4096 address
75 31704 [OTA Agent Task] [prvIngestDataBlock] Remaining: 215
75 31704 [OTA Agent Task] [prvOTAAgentTask] Called handler. Current State [WaitingForFileBlock] Event [ReceivedFileBlock] New state [WaitingForFileBlock]
75 31924 [!ot_thread] State: [WaitingForFileBlock] Received: 3 Queued: 0 Processed: 0 Dropped: 0
78 32259 [OTA Agent Task] [prvIngestDataBlock] Received file block 2, size 1024
78 32281 [OTA Agent Task] [prvPAL_WriteBlock] Write 1024 bytes at 2048 address.

```

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

```

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 153313 [OTA Agent Task] [prvStopRequestTimer] Stopping request timer.
1076 153316 [OTA Agent Task] [prvSendMsg] Msg: {"status": "PROGRESS", "statusDetails": {"self_test": "ready", "updatedBy": "0x90002"}}
1077 153345 [!ot_thread] [prvSendMsg] MSG: {"status": "PROGRESS", "statusDetails": {"self_test": "ready", "updatedBy": "0x90002"}}

```

- On Next reboot the 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>