EWN-8720CMN1AA Getting Started Guide

Version: A0

Catalog

1 EWN-8720CMN1AA Module

1.1 General Specifications

EWN-8720CMN1AA Module home page: https://eardatek.com/about/contact.html

EWN-8720CMN1AA is a highly integrated module with a low power 802.11n Wireless LAN compatible network controller, the IC it uses is RTL8720CM. It combine a Real-M300(KM4) CPU that is based on ARMv8-M architecture, and carry WLAN MAC, a 1T1R capable WLAN baseband, an RF circuit, and Bluetooth Low Energy(BLE) in a single chip.

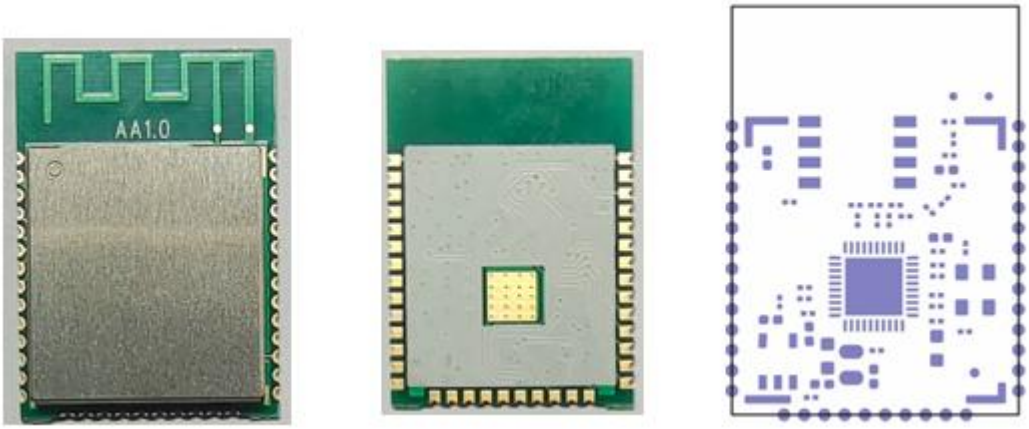


Figure 1-1 EWN-8720CMN1AA Module Appearance View

1.2 Features

MCU Features

- Real-M300(KM4) clock frequency up to 100MHz
- I-Cache 32KB/D-Cache 16KB
- Supports DMA
- EXecute In Place (XIP) on flash

Internal Memory

- Supports 384KB ROM
- Supports 256KB RAM
- Supports external flash interface
- Supports MCM embedded 4MB Psram
- Supports 2Mbit~32Mbit Flash

Wi-Fi Features

- 2.4GHz 802.11 b/g/n
- Support low power Tx/Rx for short-range application
- Support WPA/WPA2/ WMM/WPS
- Frame aggregation for increased MAC efficiency
- Low latency immediate High-Throughput Block Acknowledgement
- Long NAV for media reservation with CF-End for NAV release

Bluetooth Low Energy

- Bluetooth Low Energy(BLE) 4.2
- Supports LE secure connections/LE scatternet
- Supports 1 Master/1 Slave

Secure

- Supports secure boot
- Wi-Fi WEP, WPA, WPA2, WPS
- MD5/SHA-1/SHA2-224/SHA2-256/HMAC/AES Crypto engine

Interface

- SPI/SDIO/UART interface
- SPI/I2C master for sensor
- I2S, PWM and GPIO
- 1 GDMA with 2 channels

1.3 PCB Layout Overview

Module Size (Unit: mm): 18.0(±0.2) *25.50(±0.3)*2.9(±0.2)

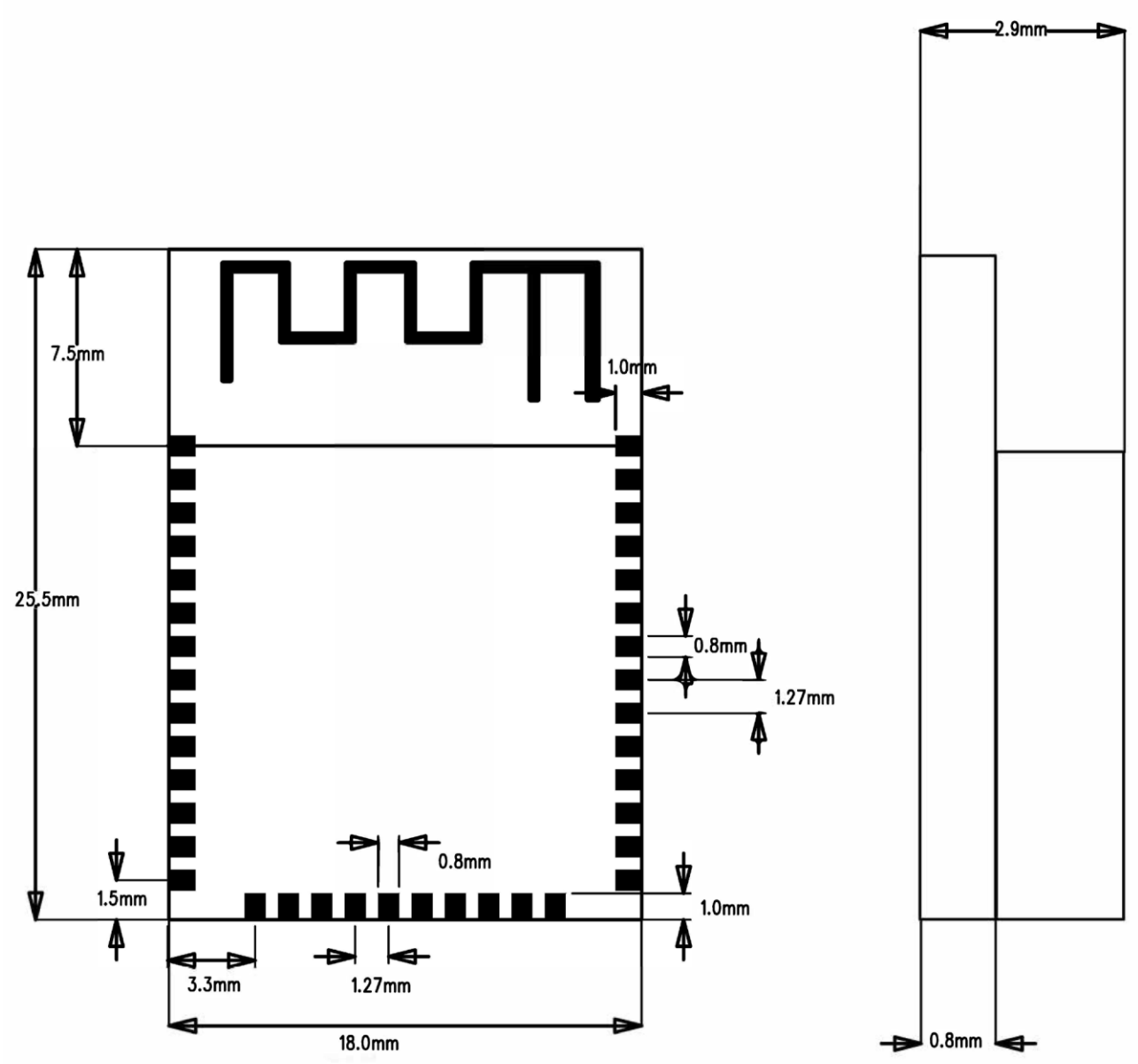


Figure 1-2 EWN-8720CMN1AA Module Size

1.4 Pin-Out Reference

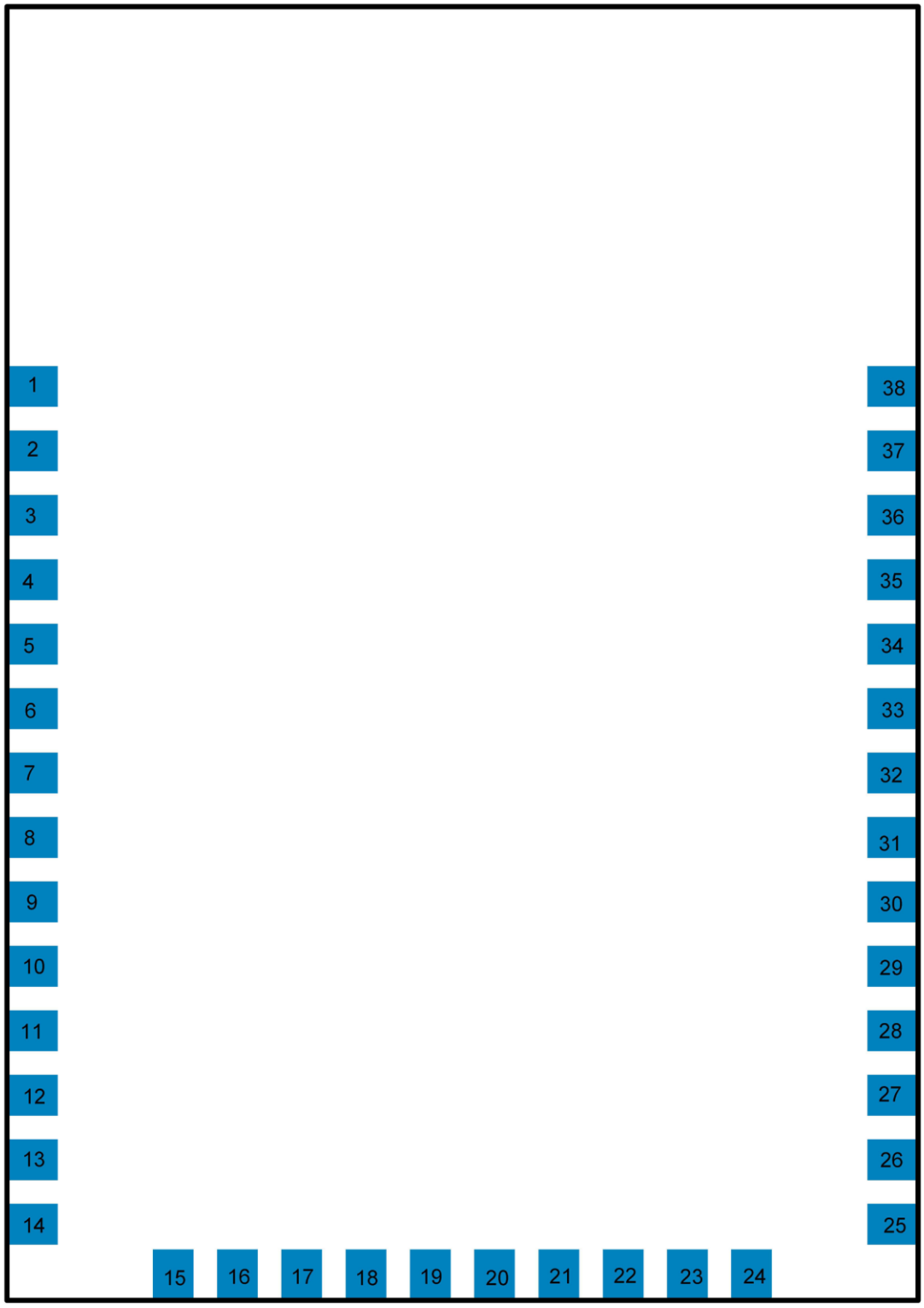


Figure 1-3 EWN-8720CMN1AA Module Pin Definition

PIN	Definition	Description
1	GND	Ground
2	VCC	Supply power 3.3V;
3	CHIP_EN	1:Enable chip 2:Shutdown chip
4	GPIOA_0 (TEST_MODE_SEL)	JTAG_CLK/UART1_IN/PWM[0] 1:Enter into test/debug mode 0:Normal operation mode
5	GPIOA_1 (Autoload_Fail)	JTAG_TMS/UART1_OUT/PWM[1] 1:eFUSE settings are not loaded 2:eFUSE settings are loaded
6	GPIOA_2	JTAG_TDO/UART1_IN/PWM[2]/SPI_CSn/I2C_SCL
7	GPIOA_3	JTAG_TDI/UART1_OUT/PWM[3]/SPI_SCL/I2C_SDA
8	GPIOA_4	JTAG_TRST/UART1_CTS/PWM[4]/SPI_MOSI
9	GPIOA_7	Flash_SPI_CS/SPI_M_CS/SPI_CSn
10	GPIOA_8	Flash _SPI_CLK/SPI_M_CLK/SPI_SCL
11	GPIOA_9	Flash _SPI_WP/SPI_M_DATA[2]/SPI_MOSI/UART0_RTS
12	GPIOA_10	Flash _SPI_SO/SPI_M_DATA[1]/SPI_MISO/UART0_CTS
13	GPIOA_11	Flash _SPI_SI/SPI_M_DATA[0]/I2C_SCL/UART0_OUT/PWM[0]
14	GPIOA_12	Flash _SPI_HOLD/SPI_M_DATA[3]/I2C_SDA/UART0_IN/PWM[1]
15	GND	Ground
16	GPIOA_13	UART0_IN/PWM[7]
17	GPIOA_14	SDIO_INT/UART0_OUT/PWM[2]
18	GPIOA_15	SDIO_D[2]/UART2_IN/PWM[3]/SPI_CSn/I2C_SCL
19	GPIOA_16	SDIO_D[3]/UART2_OUT/PWM[4]/SPI_SCL/I2C_SDA
20	GPIOA_17	SDIO_CMD/ PWM[5]
21	GPIOA_18	SDIO_CLK/ PWM[6]
22	GPIOA_19	SDIO_D[0]/UART2_CTS/PWM[7]/SPI_MOSI/I2C_SCL
23	GPIOA_20	SDIO_D[1]/UART2_RTS/PWM[0]/SPI_MISO/I2C_SDA

PIN	Definition	Description
27	GPIOA_23	LED_0/PWM[7]
38	GND	Ground

Table 1- EWN-8720CMN1AA Module Pin Definition Description

Notes:

1、UART download：Using A15&A16, and before power on.

2、Default states of all pins are High-impedance; Unused pins should be kept floating.

3、Some IO are used to connect to the flash of the module. It is recommended not to use these IO. As shown in the table.

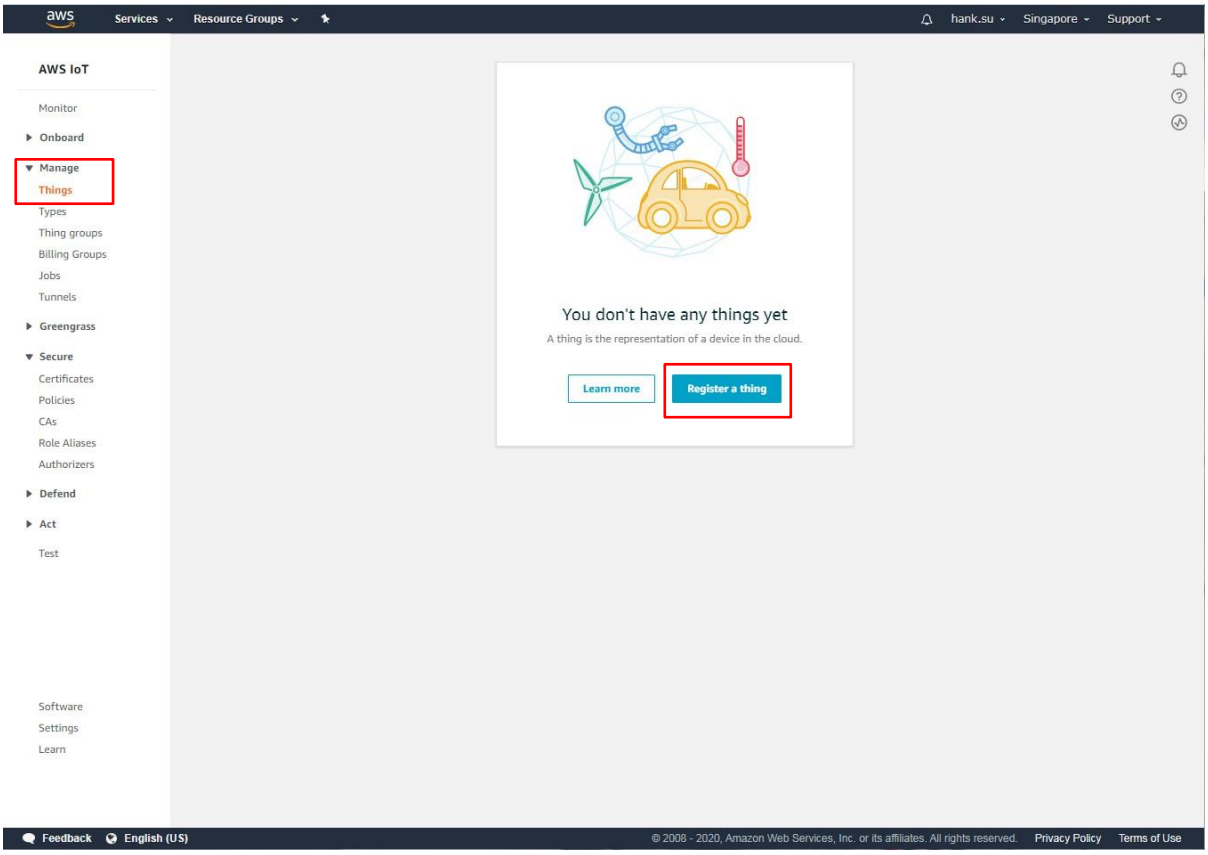
PIN	Definition	Description
9	GPIOA_7	Flash_SPI_CS
10	GPIOA_8	Flash _SPI_CLK
11	GPIOA_9	Flash _SPI_WP
12	GPIOA_10	Flash _SPI_SO
13	GPIOA_11	Flash _SPI_SI
14	GPIOA_12	Flash _SPI_HOLD

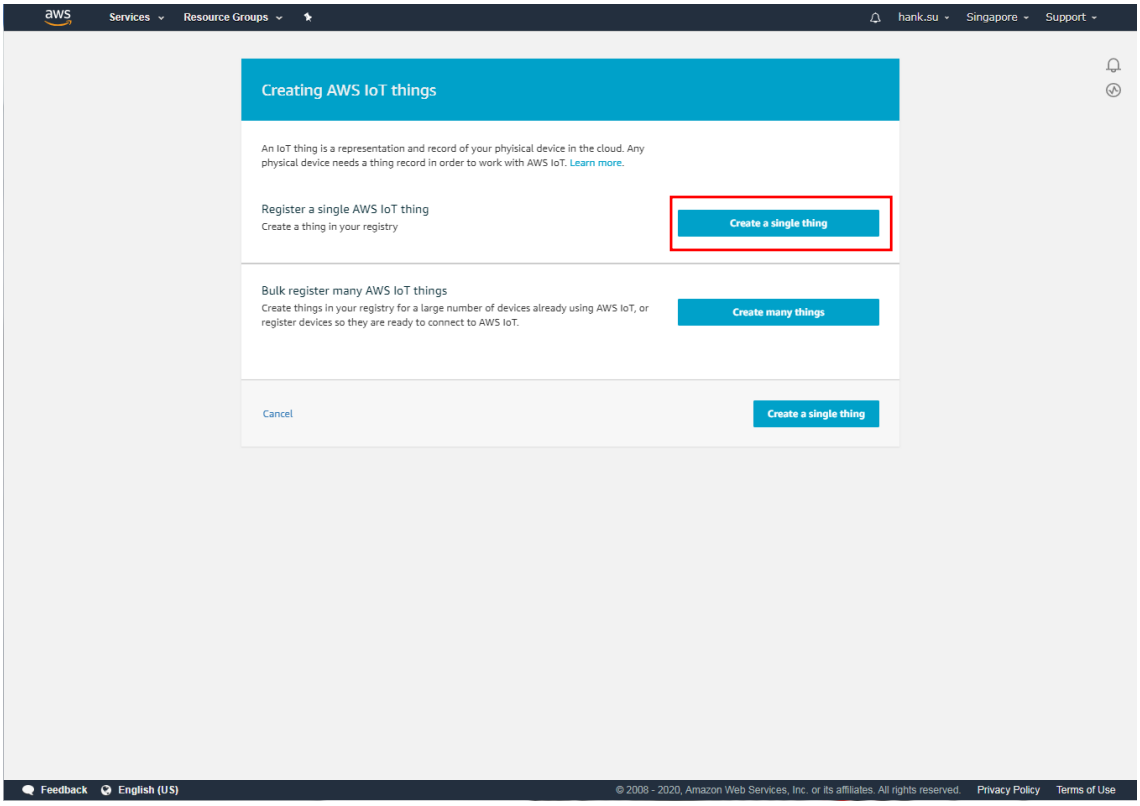
Table 2- EWN-8720CMN1AA Module Flash definition

2 Configure AWS IoT Core

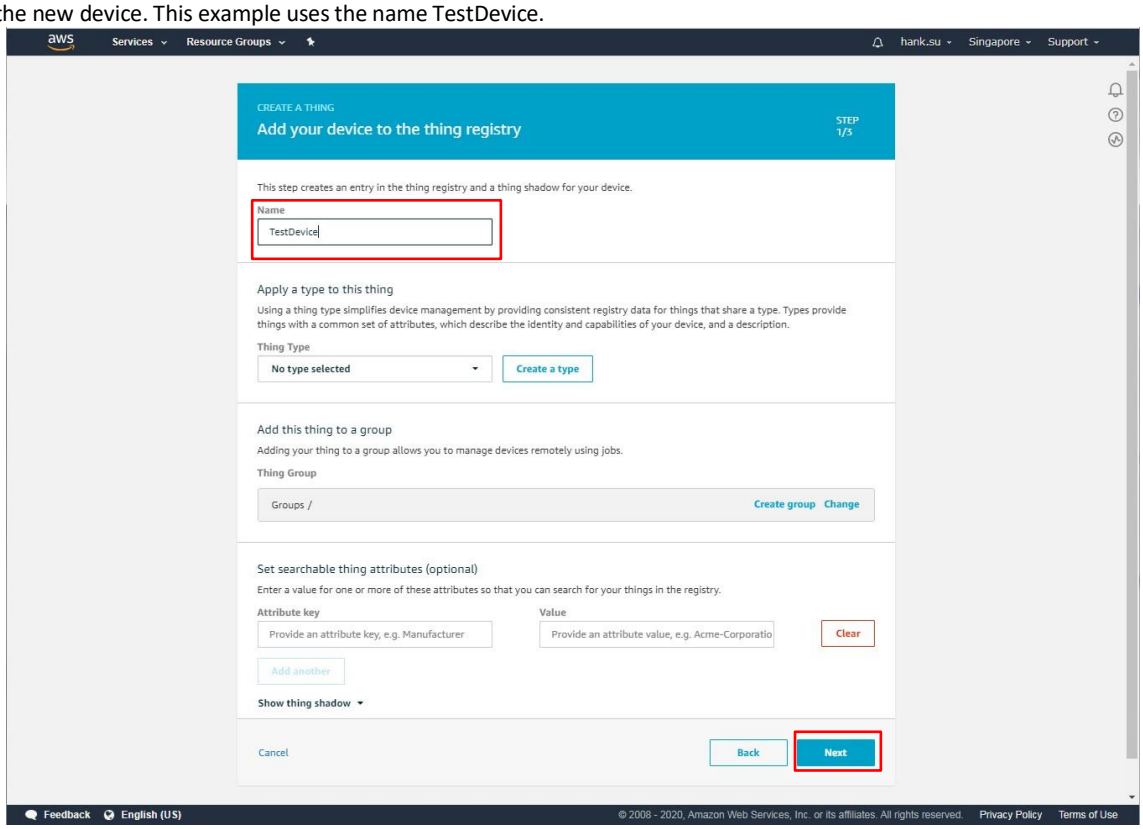
2.1 Create a New Device

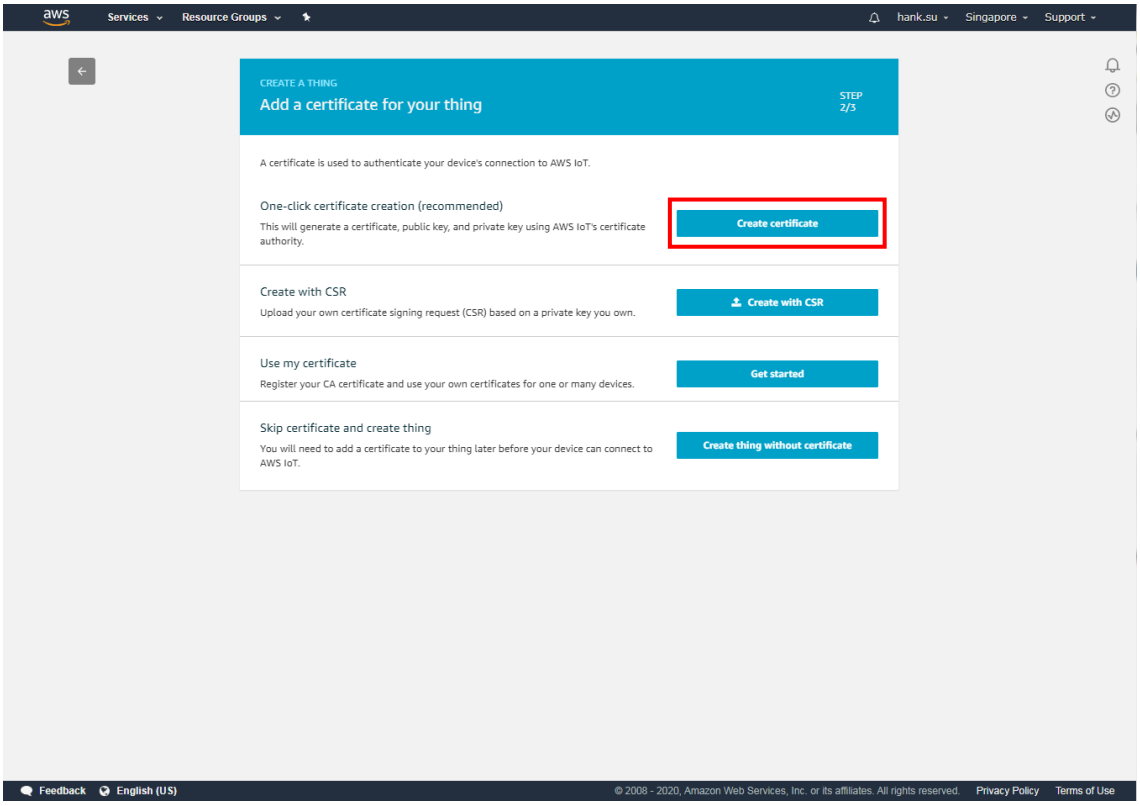
To create a new device, navigate to Manage -> Things in the left-hand navigation menu. Then click “Register a thing”.



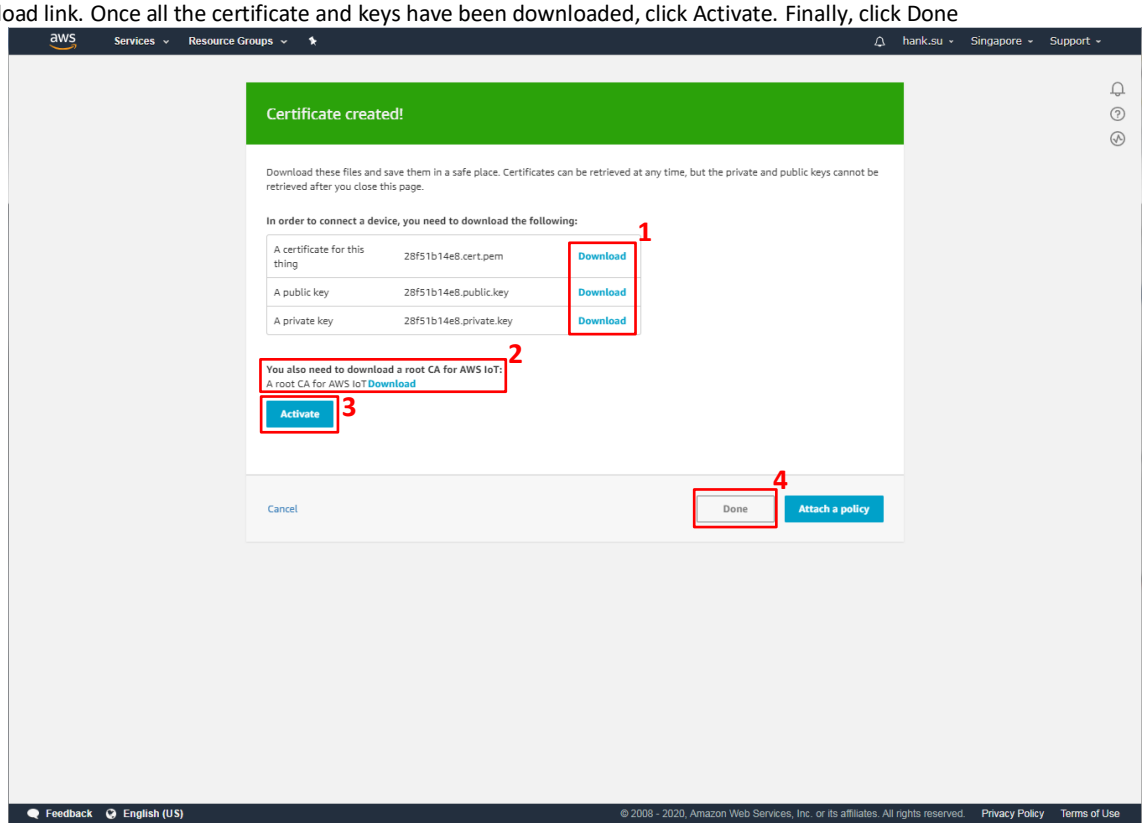


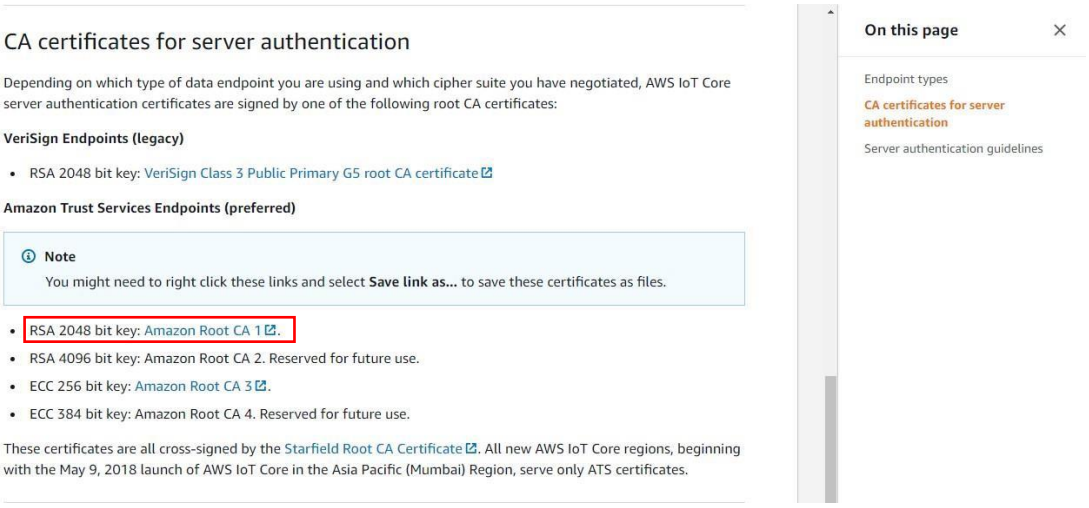
Then, name the new device. This example uses the name TestDevice.





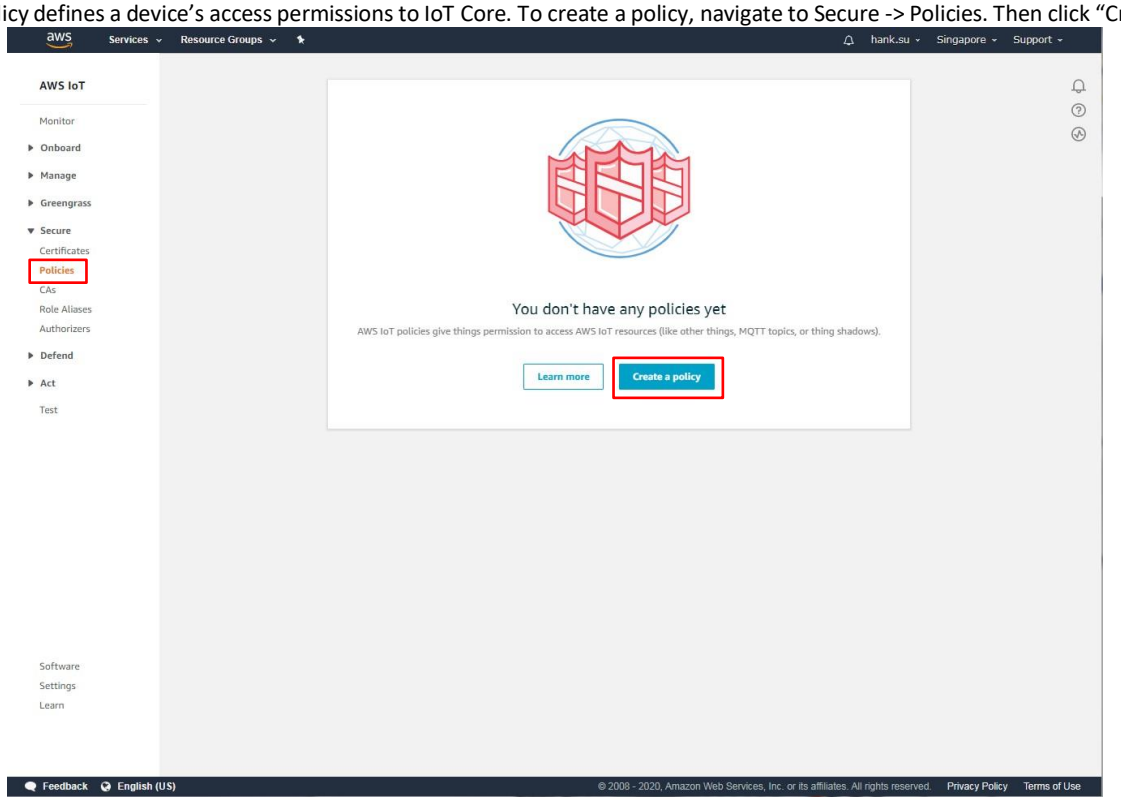
Download the certificate, public key, and private key for the device by clicking Download. Next, download the root CA for AWS IoT by clicking to the Download link. Once all the certificate and keys have been downloaded, click Activate. Finally, click Done



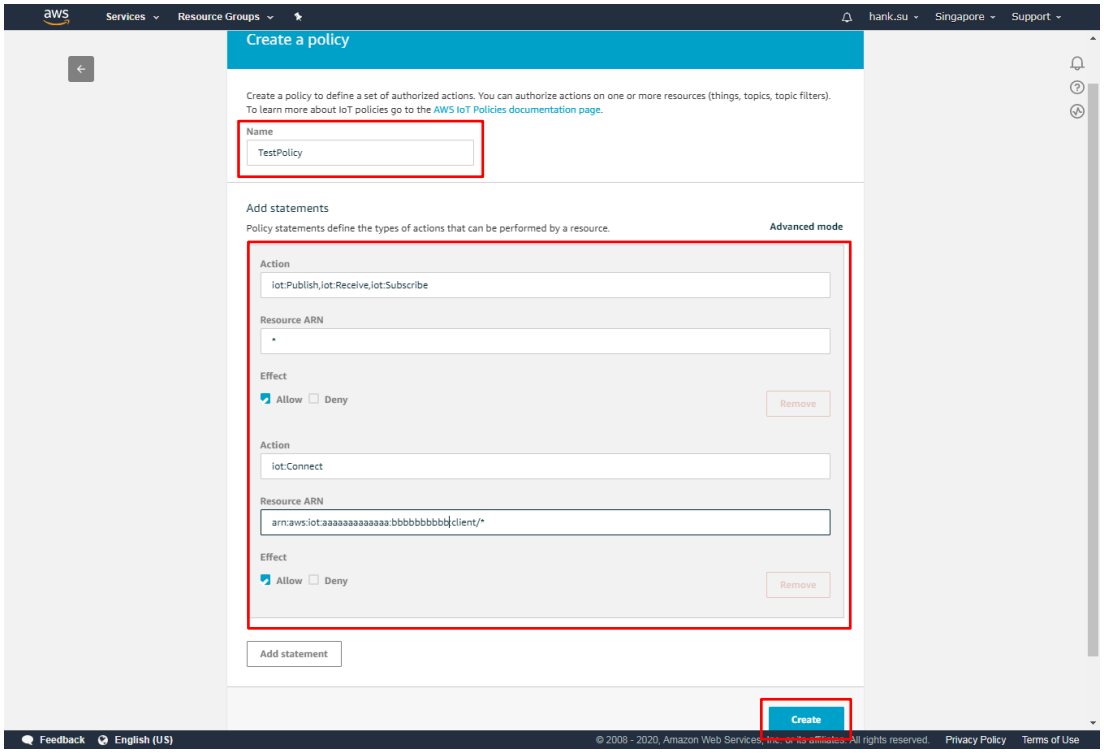


2.2 Create a policy

A policy defines a device's access permissions to IoT Core. To create a policy, navigate to Secure -> Policies. Then click "Create a policy"

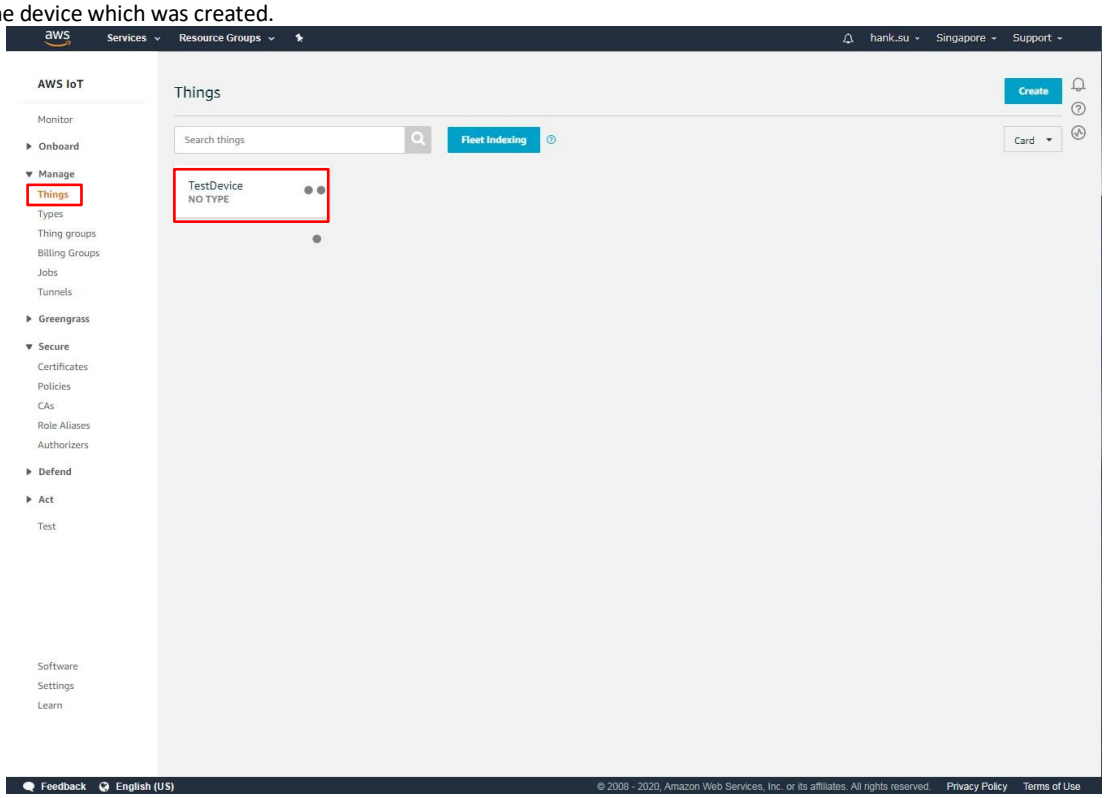


NOTE – this policy grants unrestricted access for all iot operations, and is to be used only in a development environment. For non-dev environments, all devices in your fleet must have credentials with privileges that authorize intended actions only, which include (but not limited to) AWS IoT MQTT actions such as publishing messages or subscribing to topics with specific scope and context. The specific permission policies can vary for your use cases. Identify the permission policies that best meet your business and security requirements.
For sample policies, refer to https://docs.aws.amazon.com/iot/latest/developerguide/example-iot-policies.html.
Also refer to https://docs.aws.amazon.com/iot/latest/developerguide/security-best-practices.html

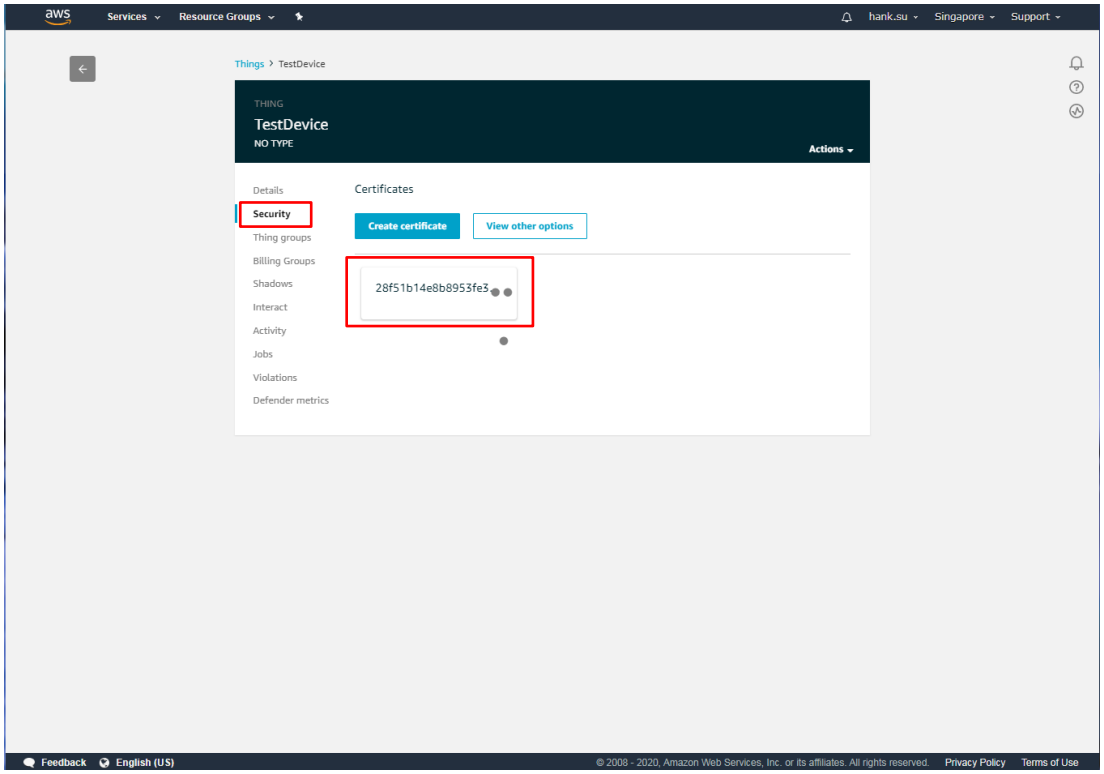


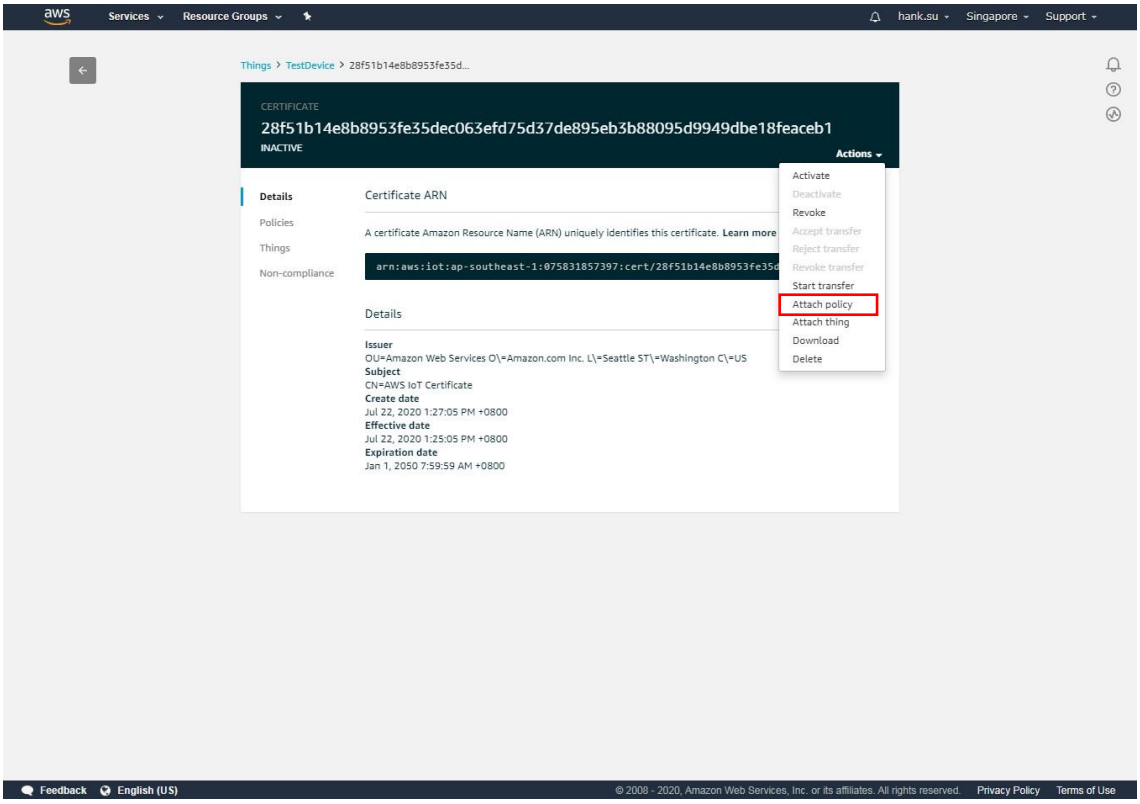
2.3 Attach Policy

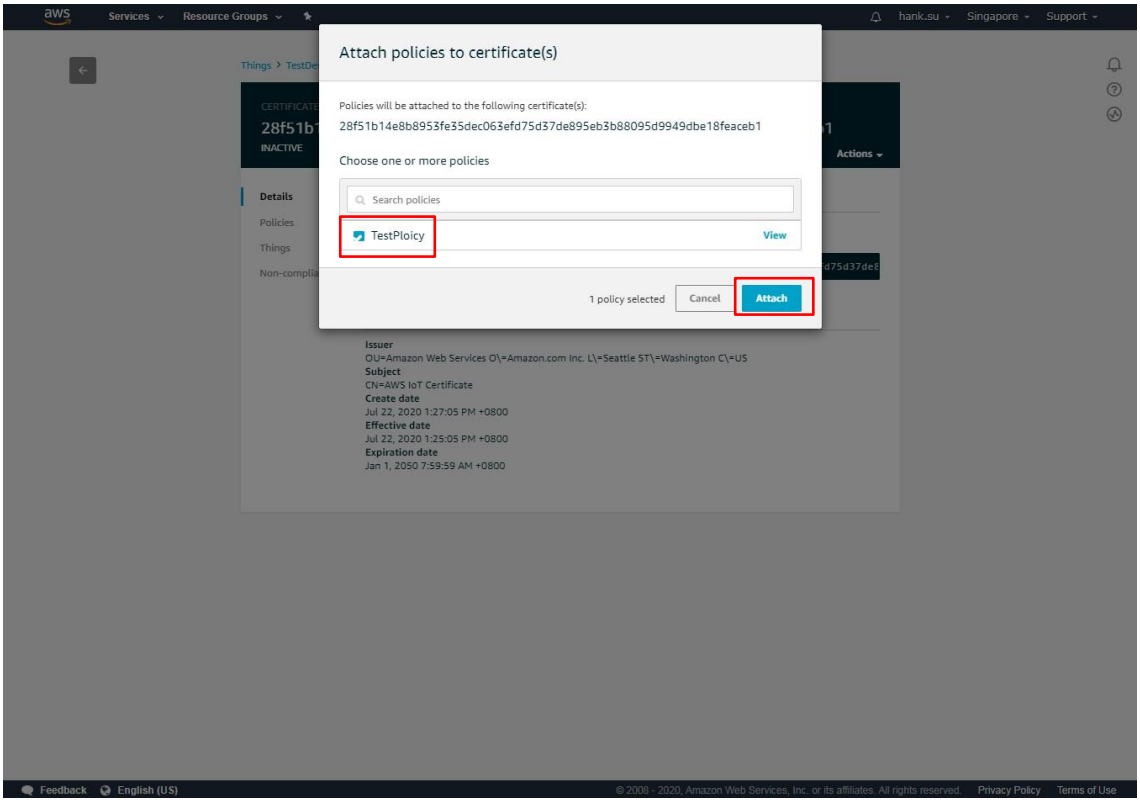
The last step to configuring the device is attaching a policy. To attach a policy to new device, navigate to Manage -> Things. Then click on the device which was created.



Click Security, then click the certificate create in previous step.


Services
Resource Groups
hank.su
Singapore
Support
Things > TestDevice
THING
TestDevice
NO TYPE
Actions
Details
Security
Thing groups
Billing Groups
Shadows
Interact
Activity
Jobs
Violations
Defender metrics
Certificates
Create certificate
View other options
28f51b14e8b8953fe3
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

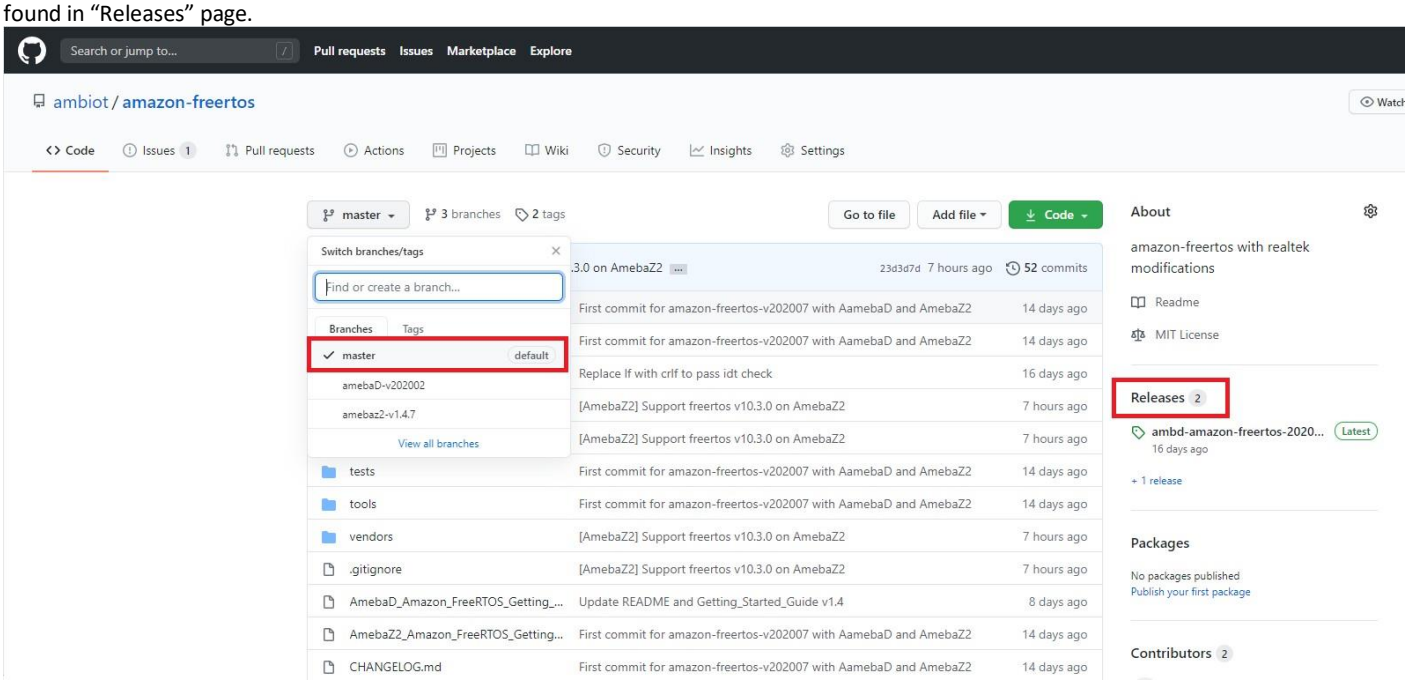

Services
Resource Groups
hank.su
Singapore
Support
Things > TestDevice > 28f51b14e8b8953fe35d...
CERTIFICATE
28f51b14e8b8953fe35dec063efd75d37de895eb3b88095d9949dbe18feaceb1
INACTIVE
Actions
Activate
Deactivate
Revoke
Accept transfer
Reject transfer
Revoke transfer
Start transfer
Attach policy
Attach thing
Download
Delete
Details
Policies
Things
Non-compliance
Certificate ARN
A certificate Amazon Resource Name (ARN) uniquely identifies this certificate. Learn more
arn:aws:iot:ap-southeast-1:075831857397:cert/28f51b14e8b8953fe35d
Details
Issuer
OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US
Subject
CN=AWS IoT Certificate
Create date
Jul 22, 2020 1:27:05 PM +0800
Effective date
Jul 22, 2020 1:25:05 PM +0800
Expiration date
Jan 1, 2050 7:59:59 AM +0800
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


Attach policies to certificate(s)
Policies will be attached to the following certificate(s):
28f51b14e8b8953fe35dec063efd75d37de895eb3b88095d9949dbe18feaceb1
Choose one or more policies
Search policies
TestPloicy
View
1 policy selected
Cancel
Attach
Issuer
OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US
Subject
CN=AWS IoT Certificate
Create date
Jul 22, 2020 1:27:05 PM +0800
Effective date
Jul 22, 2020 1:25:05 PM +0800
Expiration date
Jan 1, 2050 7:59:59 AM +0800

3 Configure AmebaZ2 Amazon FreeRTOS

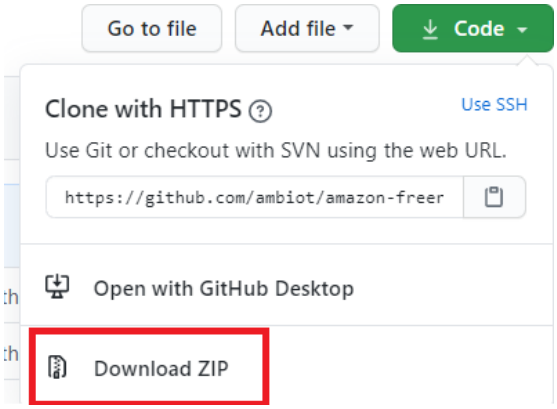
3.1 Download Source Code from github

Open source link: https://github.com/ambiot/amazon-freertos and select master for get newest source code. The stable version could be found in “Releases” page.



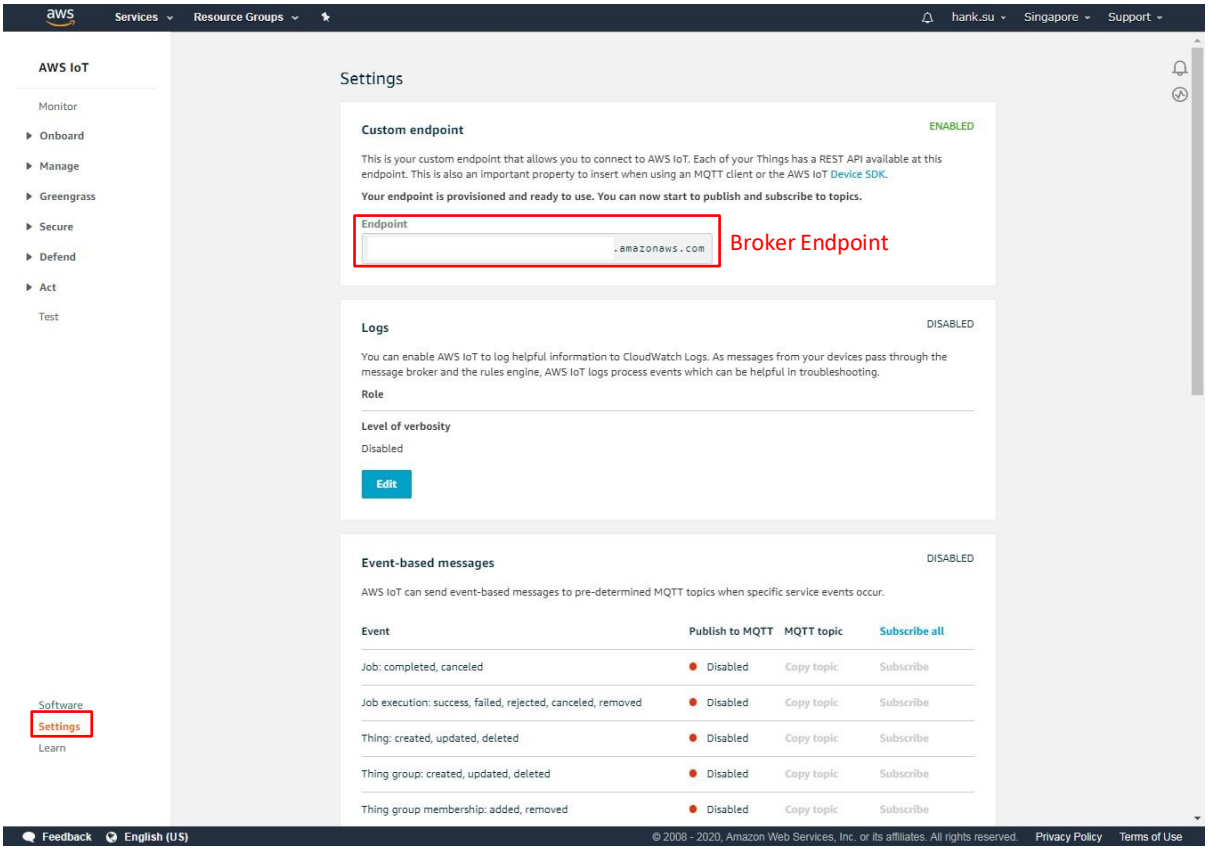
3.1.1 Cloning a repository by Download ZIP

1. On GitHub, navigate to the main page of the repository.
2. Above the list of files, click **Code**.
3. Click **Donwload ZIP** to get source code.

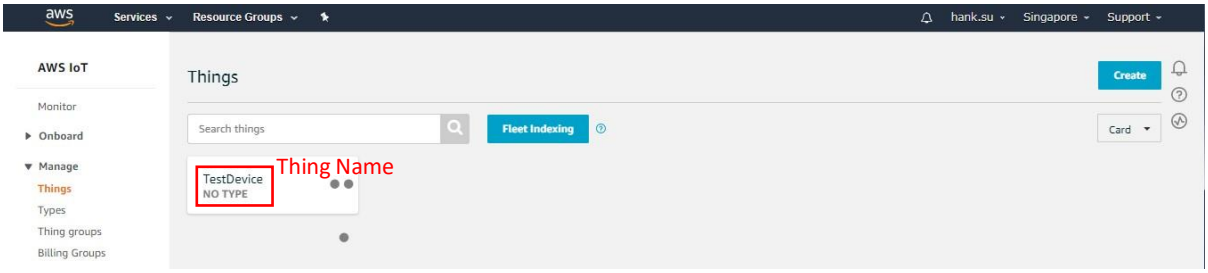


For more information, please refer "Cloning a repository from GitHub to GitHub Desktop."

3.2 Get Broker Endpoint by AWS IoT Core



3.3 Get Thing Name



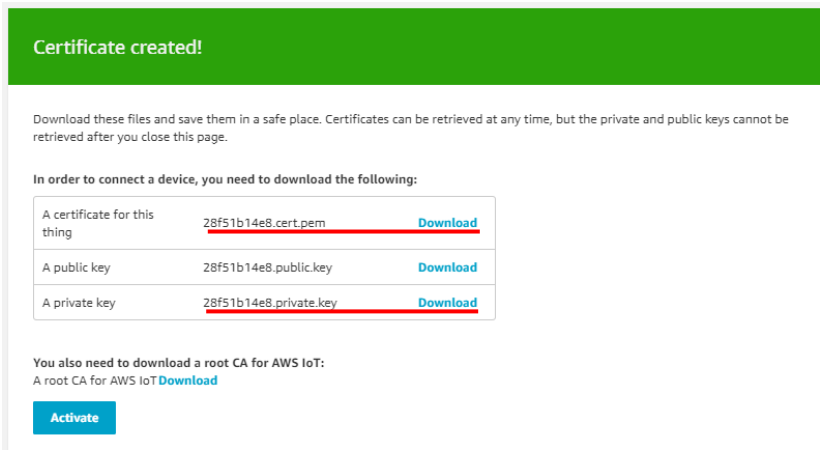
3.4 Setup IoT Core Information with AmebaZ2 Amazon FreeRTOS

Setup BROKER_ENDPOINT, THING_NAME, WIFI_SSID, PASSWORD in "amazon-freertos/blob/master/demos/include/aws_clientcredential.h"

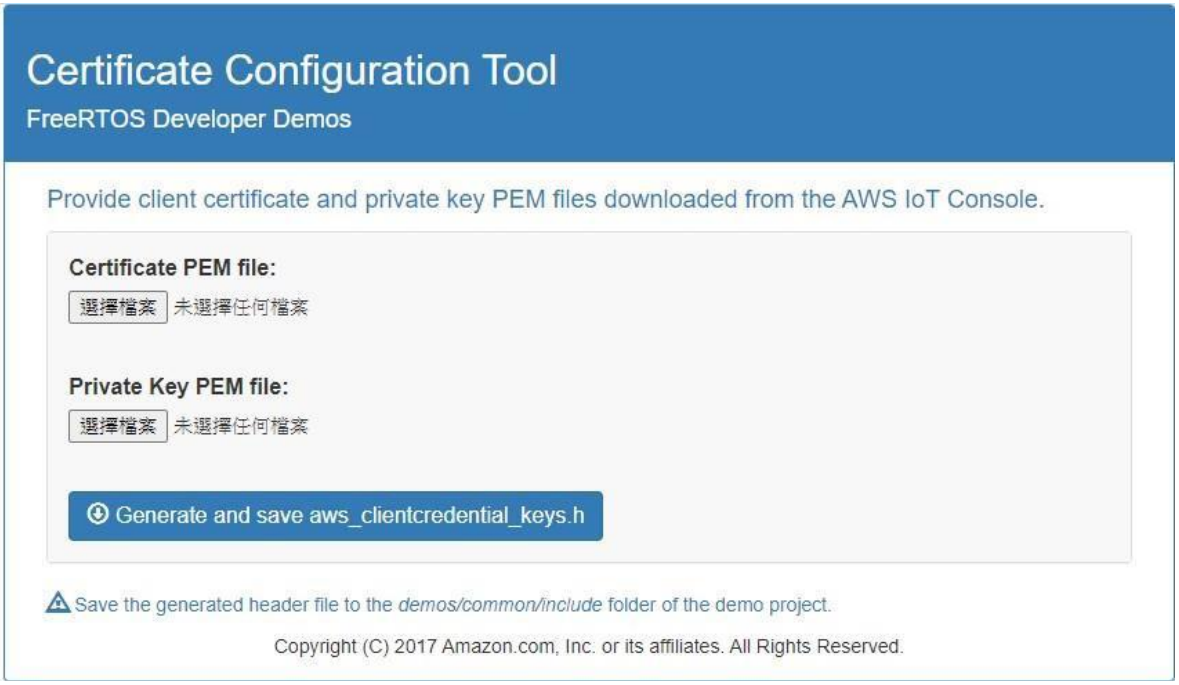
```
#define clientcredentialMQTT_BROKER_ENDPOINT          "xxxxxxxxxxxxxx.amazonaws.com"

/*
 * @brief Host name.
 *
 * @todo Set this to the unique name of your IoT Thing.
 */
#define clientcredentialIOT_THING_NAME                "TestDevice"

/*
 * @brief Port number the MQTT broker is using.
 */
#define clientcredentialMQTT_BROKER_PORT              8883

/*
 * @brief Port number the Green Grass Discovery use for JSON retrieval from cloud is using.
 */
#define clientcredentialGREENGRASS_DISCOVERY_PORT     8443

/*
 * @brief Wi-Fi network to join.
 *
 * @todo If you are using Wi-Fi, set this to your network name.
 */
#define clientcredentialWIFI_SSID                     "TestAP"

/*
 * @brief Password needed to join Wi-Fi network.
 * @todo If you are using WPA, set this to your network password.
 */
#define clientcredentialWIFI_PASSWORD                 "password"

/*
 * @brief Wi-Fi network security type.
 *
 * @see WIFISecurity_t.
 *
 * @note Possible values are eWiFiSecurityOpen, eWiFiSecurityWEP, eWiFiSecurityWPA,
 * eWiFiSecurityWPA2 (depending on the support of your device Wi-Fi radio).
 */
#define clientcredentialWIFI_SECURITY                 eWiFiSecurityWPA2

#endif /* ifndef __AWS_CLIENTCREDENTIAL__H__ */
```

3.4.1 Setup Thing's Private Key and Certificate

Filled keyCLIENT_CERTIFICATE_PEM and keyCLIENT_PRIVATE_KEY_PEM in "amazon-freertos/blob/master/demos/include/aws_clientcredential_keys.h" by xxxxxxxx-certifiacte.pem and xxxxxxxx-private.pem.key.



It can done by amazon-freertos/tools/certificate_configuration/CertificateConfigurator.html



Final aws_clientcredential_keys.h overview.

```
/*
 * @brief PEM-encoded client certificate.
 *
 * @todo If you are running one of the FreeRTOS demo projects, set this
 * to the certificate that will be used for TLS client authentication.
 *
 * @note Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----\n"\
 * "...base64 data...\n"\
 * "-----END CERTIFICATE-----\n"
 */
#define keyCLIENT_CERTIFICATE_PEM \
"-----BEGIN CERTIFICATE-----\n"\
"MIIDWjCCAkKgAwIBAgIVAIDLSSoG+EARSbBprT4Im1uu8j2vMA0GCSqGSIb3DQEB\n"\
                                                                    \n"\
"                                                                   \n"\
"                                                                   \n"\
"                                                                   \n"\
"                                                                   \n"\
"                                                                   \n"\
"                                                                   \n"\
"                                                                   \n"\
"                                                                   \n"\
"                                                                   \n"\
"                                                                   \n"\
"                                                                   \n"\
"                                                                   \n"\
"k5+NsBroU/YdvOUmzKn6XfI4nX4hLQJ2TbhAT8aq1ounGk6ZGqCbxt4mg5bB0w==\n"\
"-----END CERTIFICATE-----"
```

```
/*
 * @brief PEM-encoded client private key.
 *
 * @todo If you are running one of the FreeRTOS demo projects, set this
 * to the private key that will be used for TLS client authentication.
 *
 * @note Must include the PEM header and footer:
 * "-----BEGIN RSA PRIVATE KEY-----\n"\
 * "...base64 data...\n"\
 * "-----END RSA PRIVATE KEY-----\n"
 */
#define keyCLIENT_PRIVATE_KEY_PEM \
"-----BEGIN RSA PRIVATE KEY-----\n"\
"MIIEpAIBAAKCAQEAwop96WNucGebARFjD8O+CLsqcBNn/AHyhEcozLZC8qoECUOn\n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"                                                                  \n"\
"pOWEuLUuz2FAv1noAbN/6OQ8H/PT0AFJT/ghA04GnIUF0kjSzY60ehS2mVp6neP+\n"\
"AZjzZ6QJYlb5/PFz9oES448kpyaAoS2ke86+R4r4YOMBK+I5RVbfSQ==\n"\
"-----END RSA PRIVATE KEY-----\n"
```

3.4.2 Enable FreeRTOS demo on AmebaZ2

Find platform_opts.h in amazon-freertos\vendors\realtek\boards\amebaZ2\aws_demos\config_files and enable **CONFIG_EXAMPLE_AMAZON_FREERTOS**

```
/* For Amazon FreeRTOS SDK example */
#define CONFIG_EXAMPLE_AMAZON_FREERTOS   1
```

Fine aws_demo_config.h in amazon-freertos\vendors\realtek\boards\amebaZ2\aws_demos\config_files and add **CONFIG_MQTT_DEMO_ENABLED**

```
/* To run a particular demo you need to define one of these.
 * Only one demo can be configured at a time
 *
 *          CONFIG_MQTT_DEMO_ENABLED
 *          CONFIG_SHADOW_DEMO_ENABLED
 *          CONFIG_OTA_UPDATE_DEMO_ENABLED
 *
 *  These defines are used in iot_demo_runner.h for demo selection */

#define CONFIG_MQTT_DEMO_ENABLED
```

Now you can start to compile AmebaZ2 Amazon FreeRTOS

4 Compile AmebaZ2 Amazon FreeRTOS

4.1 Pre-Requisite

- Required source code. (https://github.com/ambiot/amazon-freertos)
- AmebaZ2 Demo board
- Realtek Image Tool
- IAR Embedded Workbench ver.8.30.1

4.2 IAR Build Environment Setup

The IAR IDE (integrated development environment) only supports Windows OS, this section is applicable for **Windows OS only**.

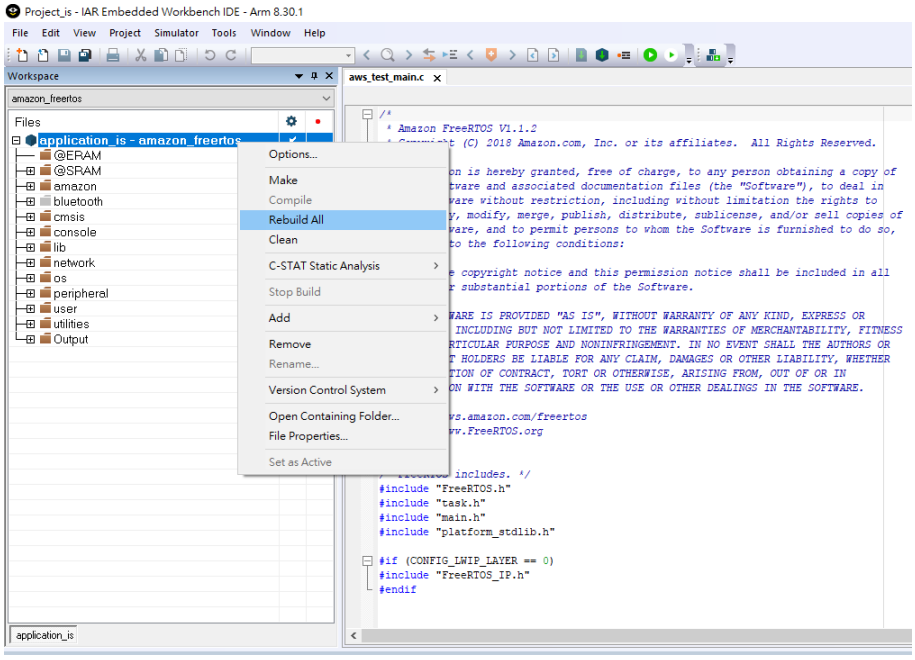
4.3 Install IAR IDE

IAR IDE provides the toolchain for Ameba-ZII. It allows users to write programs, compile and upload them to your board. Also, it supports step-by-step debug function.
User can visit the official website of **IAR Embedded Workbench** and install the IDE by following its instructions.

Note: Please use IAR version **8.30** or above.

4.4 Compilation

1) Open amazon-freertos/projects/realtek/amebaZ2/IAR/aws_demos/Project_is.eww.
2) Confirm 'application_is' in Work Space, right click 'application_is' and choose "**Rebuild All**" to compile.
3) Make sure there is no error after compile.



4.5 Generate Image Binary

After compile, the images **partition.bin**, **bootloader.bin**, **firmware_is.bin** and **flash_is.bin** can be seen in the amazon-freertos/projects/realtek/amebaZ2/IAR/aws_demos/Debug/Exe.

1) **partition.bin** stores partition table, recording the address of Boot image and firmware image;
2) **bootloader.bin** is bootloader image;
3) **firmware_is.bin** is application image;
4) flash_is.bin links partition.bin, bootloader.bin and firmware_is.bin. Users need to choose flash_is.bin when downloading the image to board by Image Tool

5 ImageTool

The tool can be find in amazon-freertos/vendors/realtek/tools/AmebaZ2_PGTool_v1.2.8

5.1 Introduction

This chapter introduces how to use Image Tool to generate and download images. As show in picture below, Image Tool has two menu pages:

- Download: used as image download server to transmit images to Ameba through UART.

Note: If you need to download code via external uart, must use FT232 USB to connect UART dongle.

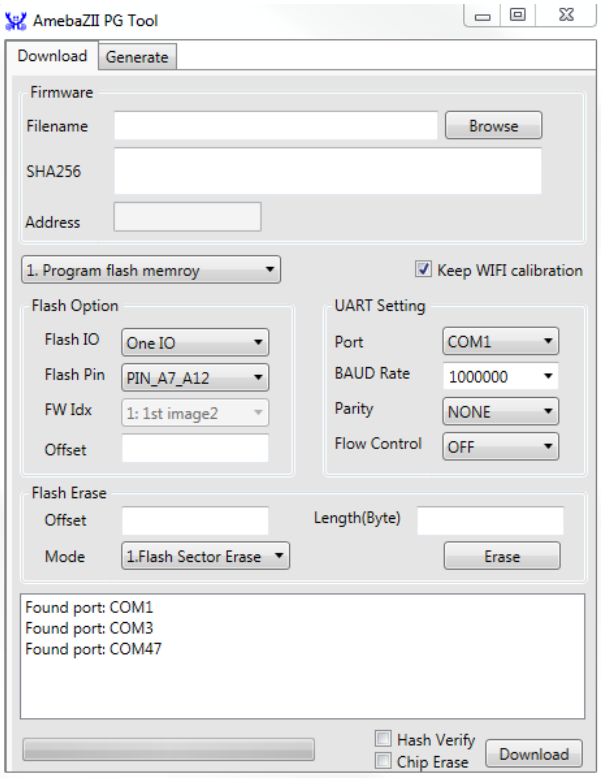


Fig 5-1 AmebaZII ImageTool UI

5.2 Environment Setup

5.2.1 Hardware Setup

The hardware connection diagram is shown below.

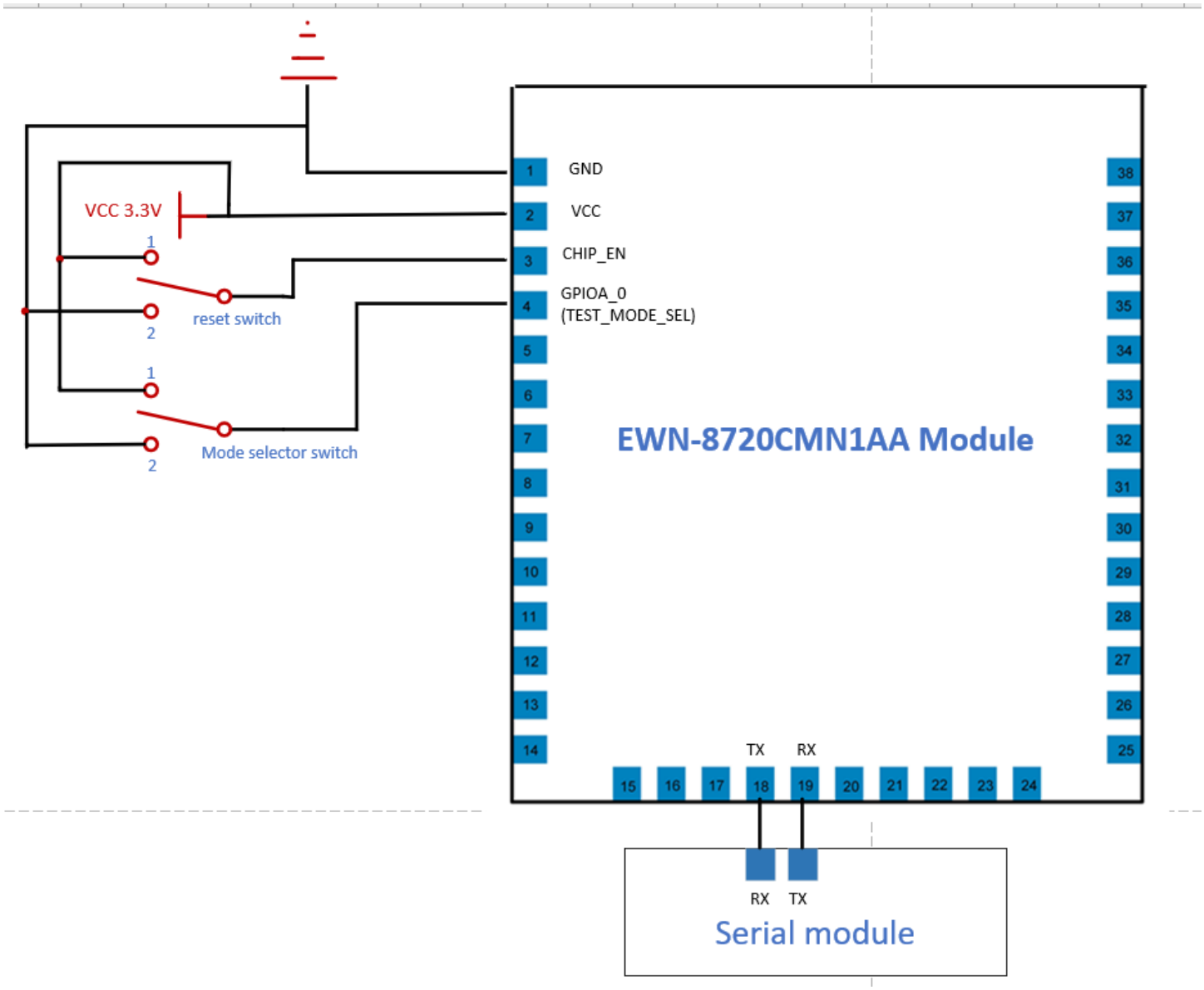


Fig 5-2 hardware connection diag

5.2.2 Software Setup

- Environment Requirements: EX. WinXP, Win 7 Above, Microsoft .NET Framework 3.5
- AmebaZII_PGTool_v1.2.8.exe

5.3 Image Download

User can download the image to demo board by following steps:

1) Trigger Ameba-ZII chip enter **UART download mode** by:

a. First, ensure that the log UART is properly connected.As shown in figure 5-2, flip the mode selector switch to position 1,then flip the reset switch to position 2, and finally flip the reset switch to position 1.

b. If the chip enters **download mode**, the below log should be shown on log UART console.

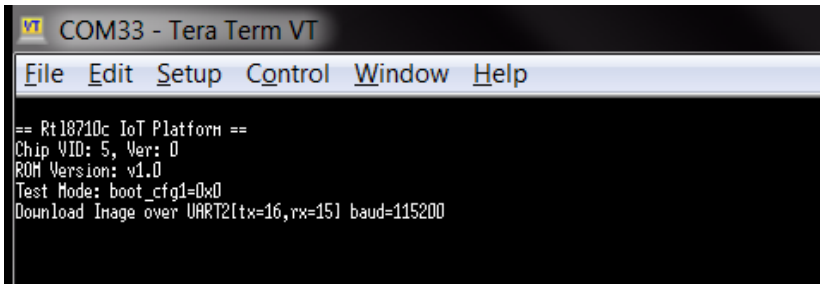


Fig 5-3 Ameba-ZII UART download mode

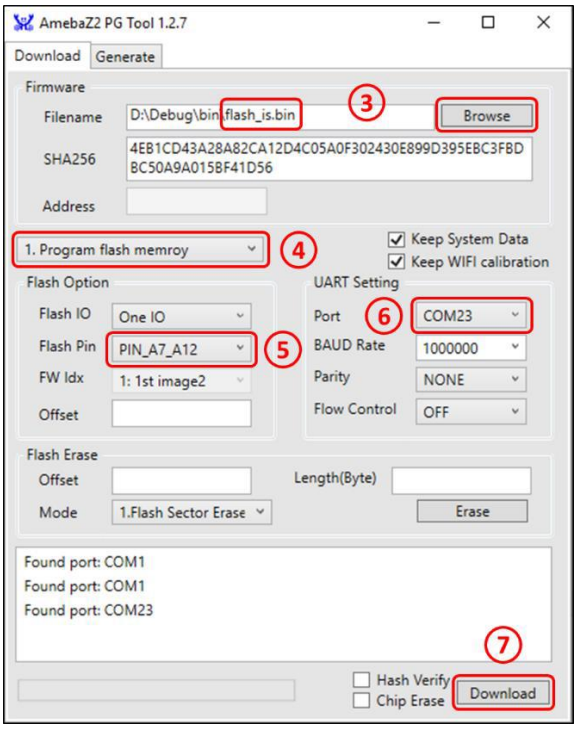
c. After confirming it is in download mode, **remember to disconnect the log UART console before using Image Tool to download**, because the tool will also need to connect to this log UART port.

Notes:

After downloading, if you want to switch to normal operation mode, follow the instructions below.

As shown in figure 5-2, flip the mode selector switch to position 2,then flip the reset switch to position 2, and finally flip the reset switch to position 1.

2) Open **AmebaZ2 PG Tool**



3) "**Browse**" to choose the image to be downloaded (amazon-freertos/projects/realtek/amebaZ2/IAR/aws_demos/Debug/Exe/flash_is.bin)

4) Choose "**1. Program flash memory**"

5) Choose correct "**Flash Pin**" according to the IC part number

Flash Pin	**IC part number**
PIN_A7_A12	RTL8710CX/RTL8720CM
PIN_B6_B12	RTL8720CF

6) Choose the correct **UART port** (use **rescan** to update the port list)

7) Click "**Download**" to start downloading image. While downloading, the status will be shown on the left bar.

Note: It's recommended to use the default settings unless user is familiar with them.

6 MQTT Demo

6.1 Get Device Log

Install Tera Term to get device log

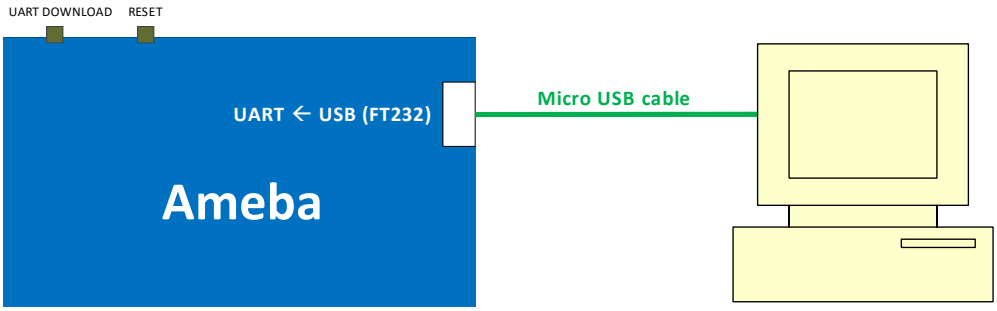
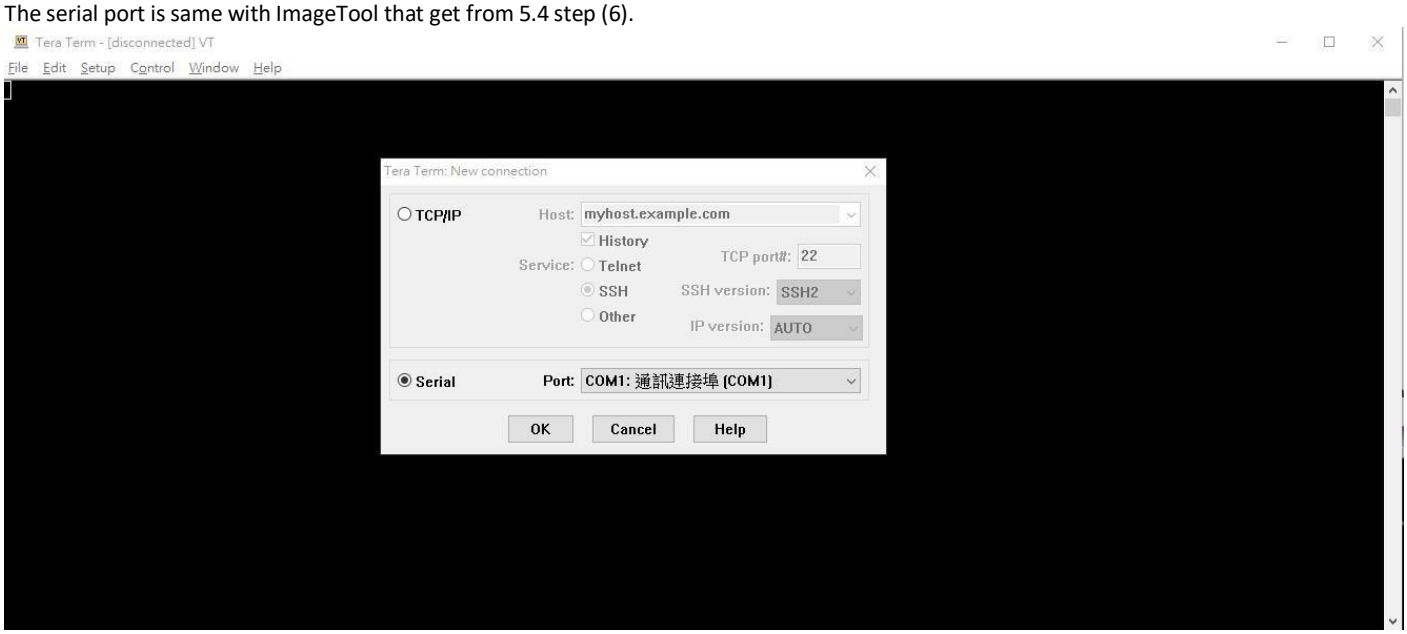


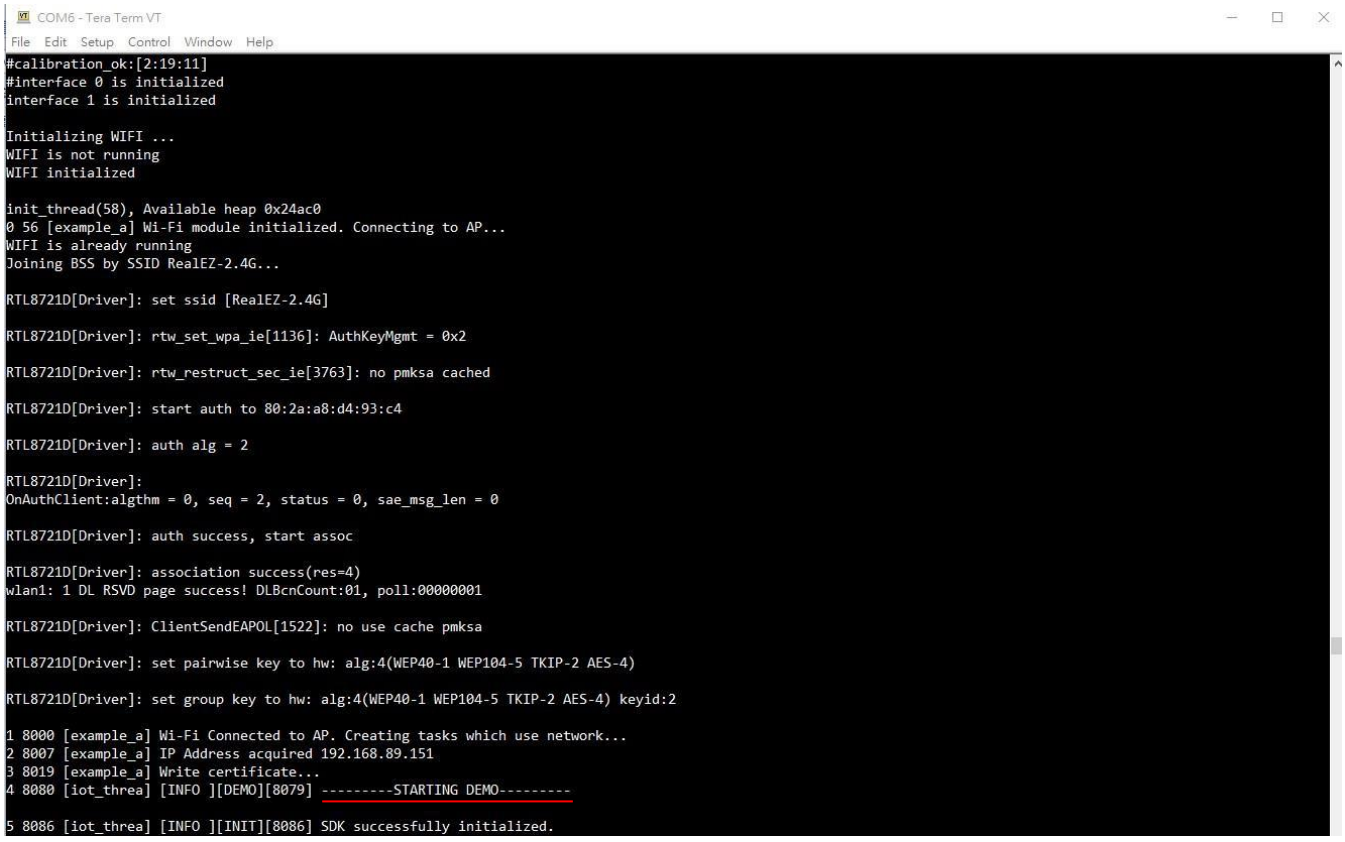
Fig 6-1 Hardware setup

The serial port is same with ImageTool that get from 5.4 step (6).

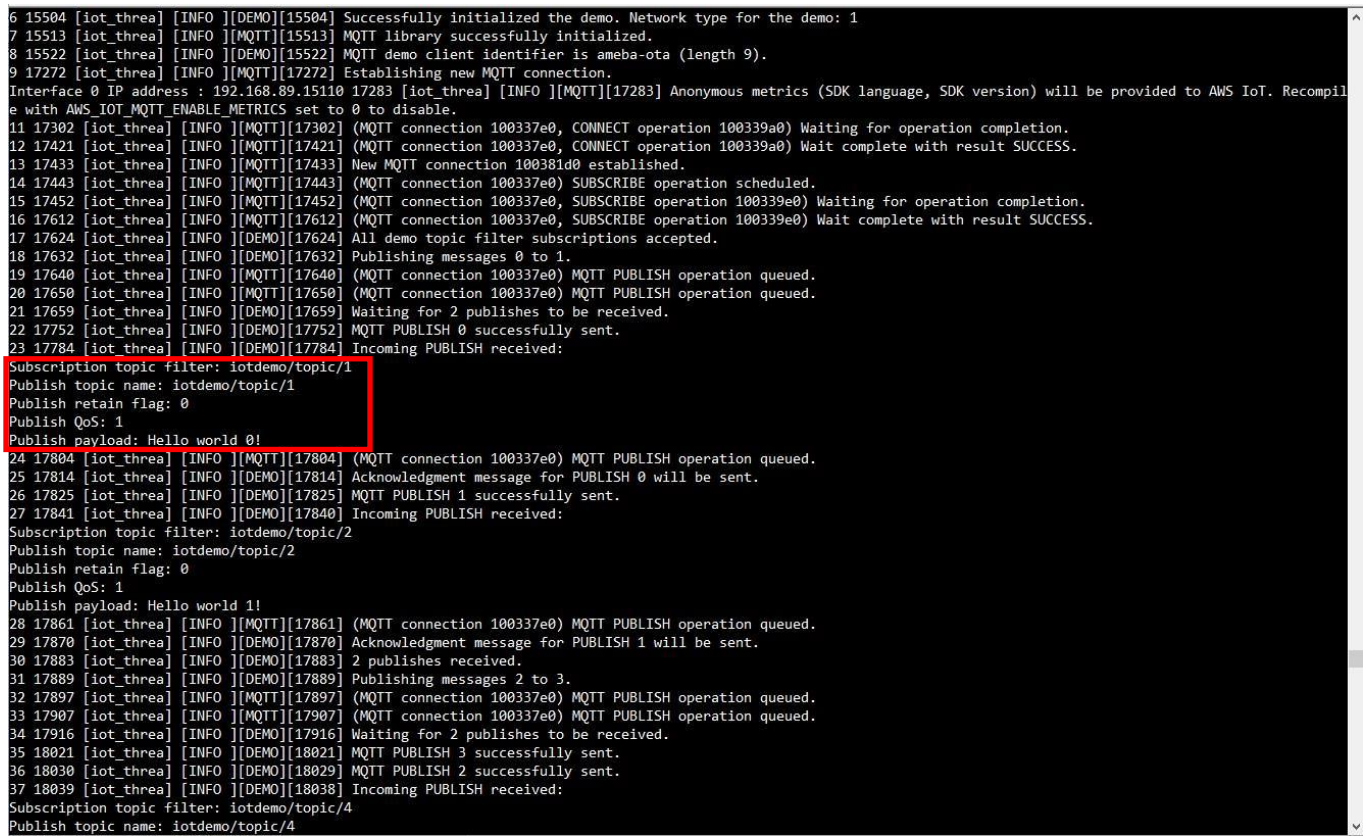


6.2 Run MQTT Demo

Default setting of SDK are enable MQTT demo. Once the AmebaD EVB has rebooted, the application will automatically start run MQTT demo and communicate to IoT Core.


COM6 - Tera Term VT
File Edit Setup Control Window Help
#calibration_ok:[2:19:11]
#interface 0 is initialized
interface 1 is initialized
Initializing WIFI ...
WIFI is not running
WIFI initialized
init_thread(58), Available heap 0x24ac0
0 56 [example_a] Wi-Fi module initialized. Connecting to AP...
WIFI is already running
Joining BSS by SSID RealEZ-2.4G...
RTL8721D[Driver]: set ssid [RealEZ-2.4G]
RTL8721D[Driver]: rtw_set_wpa_ie[1136]: AuthKeyMgmt = 0x2
RTL8721D[Driver]: rtw_restruct_sec_ie[3763]: no pmksa cached
RTL8721D[Driver]: start auth to 80:2a:a8:d4:93:c4
RTL8721D[Driver]: auth alg = 2
RTL8721D[Driver]:
OnAuthClient:algthm = 0, seq = 2, status = 0, sae_msg_len = 0
RTL8721D[Driver]: auth success, start assoc
RTL8721D[Driver]: association success(res=4)
wlan1: 1 DL RSVD page success! DLBcnCount:01, poll:00000001
RTL8721D[Driver]: ClientSendEAPOL[1522]: no use cache pmksa
RTL8721D[Driver]: set pairwise key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4)
RTL8721D[Driver]: set group key to hw: alg:4(WEP40-1 WEP104-5 TKIP-2 AES-4) keyid:2
1 8000 [example_a] Wi-Fi Connected to AP. Creating tasks which use network...
2 8007 [example_a] IP Address acquired 192.168.89.151
3 8019 [example_a] Write certificate...
4 8080 [iot_threa] [INFO][DEMO][8079] ---------STARTING DEMO---------
5 8086 [iot_threa] [INFO][INIT][8086] SDK successfully initialized.

...


6 15504 [iot_threa] [INFO][DEMO][15504] Successfully initialized the demo. Network type for the demo: 1
7 15513 [iot_threa] [INFO][MQTT][15513] MQTT library successfully initialized.
8 15522 [iot_threa] [INFO][DEMO][15522] MQTT demo client identifier is ameba-ota (length 9).
9 17272 [iot_threa] [INFO][MQTT][17272] Establishing new MQTT connection.
Interface 0 IP address : 192.168.89.15110 17283 [iot_threa] [INFO][MQTT][17283] Anonymous metrics (SDK language, SDK version) will be provided to AWS IoT. Recompil
e with AWS_IOT_MQTT_ENABLE_METRICS set to 0 to disable.
11 17302 [iot_threa] [INFO][MQTT][17302] (MQTT connection 100337e0, CONNECT operation 100339a0) Waiting for operation completion.
12 17421 [iot_threa] [INFO][MQTT][17421] (MQTT connection 100337e0, CONNECT operation 100339a0) Wait complete with result SUCCESS.
13 17433 [iot_threa] [INFO][MQTT][17433] New MQTT connection 100381d0 established.
14 17443 [iot_threa] [INFO][MQTT][17443] (MQTT connection 100337e0) SUBSCRIBE operation scheduled.
15 17452 [iot_threa] [INFO][MQTT][17452] (MQTT connection 100337e0, SUBSCRIBE operation 100339e0) Waiting for operation completion.
16 17612 [iot_threa] [INFO][MQTT][17612] (MQTT connection 100337e0, SUBSCRIBE operation 100339e0) Wait complete with result SUCCESS.
17 17624 [iot_threa] [INFO][DEMO][17624] All demo topic filter subscriptions accepted.
18 17632 [iot_threa] [INFO][DEMO][17632] Publishing messages 0 to 1.
19 17640 [iot_threa] [INFO][MQTT][17640] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
20 17650 [iot_threa] [INFO][MQTT][17650] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
21 17659 [iot_threa] [INFO][DEMO][17659] Waiting for 2 publishes to be received.
22 17752 [iot_threa] [INFO][DEMO][17752] MQTT PUBLISH 0 successfully sent.
23 17784 [iot_threa] [INFO][DEMO][17784] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/1
Publish topic name: iotdemo/topic/1
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 0!
24 17804 [iot_threa] [INFO][MQTT][17804] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
25 17814 [iot_threa] [INFO][DEMO][17814] Acknowledgment message for PUBLISH 0 will be sent.
26 17825 [iot_threa] [INFO][DEMO][17825] MQTT PUBLISH 1 successfully sent.
27 17841 [iot_threa] [INFO][DEMO][17840] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/2
Publish topic name: iotdemo/topic/2
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 1!
28 17861 [iot_threa] [INFO][MQTT][17861] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
29 17870 [iot_threa] [INFO][DEMO][17870] Acknowledgment message for PUBLISH 1 will be sent.
30 17883 [iot_threa] [INFO][DEMO][17883] 2 publishes received.
31 17889 [iot_threa] [INFO][DEMO][17889] Publishing messages 2 to 3.
32 17897 [iot_threa] [INFO][MQTT][17897] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
33 17907 [iot_threa] [INFO][MQTT][17907] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
34 17916 [iot_threa] [INFO][DEMO][17916] Waiting for 2 publishes to be received.
35 18021 [iot_threa] [INFO][DEMO][18021] MQTT PUBLISH 3 successfully sent.
36 18030 [iot_threa] [INFO][DEMO][18029] MQTT PUBLISH 2 successfully sent.
37 18039 [iot_threa] [INFO][DEMO][18038] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/4
Publish topic name: iotdemo/topic/4

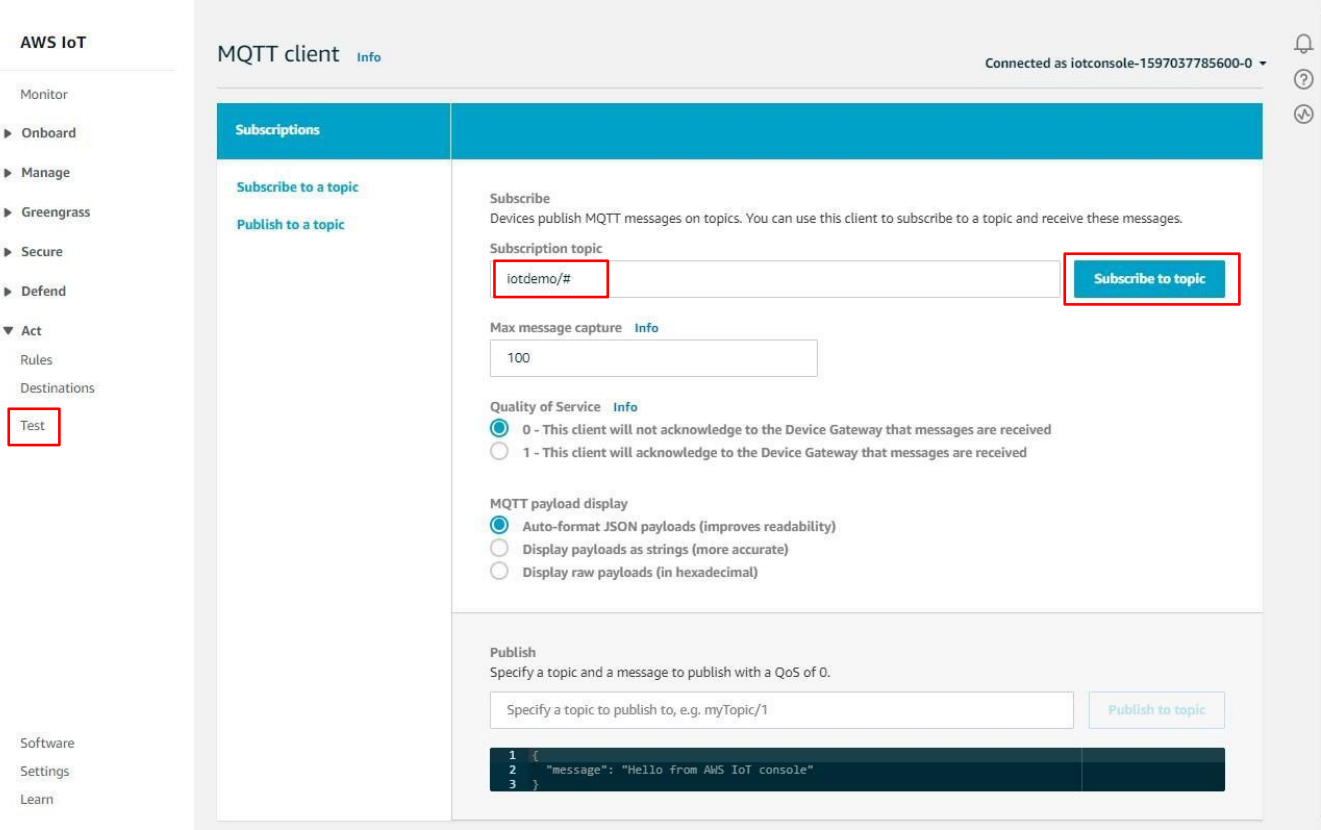
...

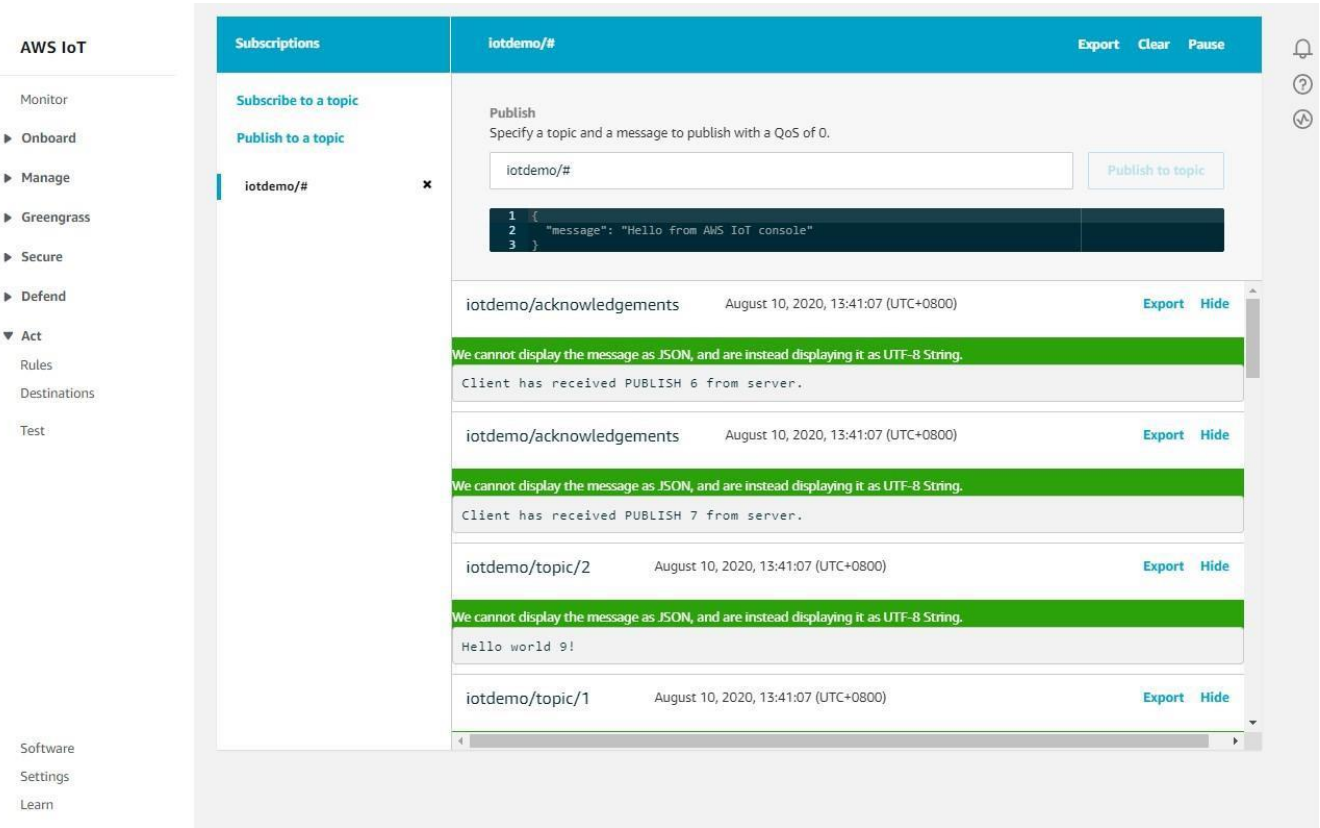
```
Publish payload: Hello world 16!
132 19827 [iot_threa] [INFO ][MQTT][19827] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
133 19837 [iot_threa] [INFO ][DEMO][19837] Acknowledgment message for PUBLISH 16 will be sent.
134 19851 [iot_threa] [INFO ][DEMO][19851] 2 publishes received.
135 19857 [iot_threa] [INFO ][DEMO][19857] Publishing messages 18 to 19.
136 19865 [iot_threa] [INFO ][MQTT][19865] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
137 19876 [iot_threa] [INFO ][MQTT][19876] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
138 19885 [iot_threa] [INFO ][DEMO][19885] Waiting for 2 publishes to be received.
139 19953 [iot_threa] [INFO ][DEMO][19953] MQTT PUBLISH 18 successfully sent.
140 19980 [iot_threa] [INFO ][DEMO][19980] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/3
Publish topic name: iotdemo/topic/3
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 18!
141 20001 [iot_threa] [INFO ][MQTT][20001] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
142 20011 [iot_threa] [INFO ][DEMO][20011] Acknowledgment message for PUBLISH 18 will be sent.
143 20053 [iot_threa] [INFO ][DEMO][20053] MQTT PUBLISH 19 successfully sent.
144 20069 [iot_threa] [INFO ][DEMO][20068] Incoming PUBLISH received:
Subscription topic filter: iotdemo/topic/4
Publish topic name: iotdemo/topic/4
Publish retain flag: 0
Publish QoS: 1
Publish payload: Hello world 19!
145 20089 [iot_threa] [INFO ][MQTT][20089] (MQTT connection 100337e0) MQTT PUBLISH operation queued.
146 20099 [iot_threa] [INFO ][DEMO][20099] Acknowledgment message for PUBLISH 19 will be sent.
147 20108 [iot_threa] [INFO ][DEMO][20108] 2 publishes received.
148 20116 [iot_threa] [INFO ][MQTT][20116] (MQTT connection 100337e0) UNSUBSCRIBE operation scheduled.
149 20129 [iot_threa] [INFO ][MQTT][20128] (MQTT connection 100337e0, UNSUBSCRIBE operation 100339e0) Waiting for operation completion.
150 20322 [iot_threa] [INFO ][MQTT][20321] (MQTT connection 100337e0, UNSUBSCRIBE operation 100339e0) Wait complete with result SUCCESS.
151 20335 [iot_threa] [INFO ][MQTT][20335] (MQTT connection 100337e0) Disconnecting connection.
152 20347 [iot_threa] [INFO ][MQTT][20347] (MQTT connection 100337e0, DISCONNECT operation 100339e0) Waiting for operation completion.
153 20359 [iot_threa] [INFO ][MQTT][20359] (MQTT connection 100337e0, DISCONNECT operation 100339e0) Wait complete with result SUCCESS.
154 20371 [iot_threa] [INFO ][MQTT][20371] (MQTT connection 100337e0) Connection disconnected.
155 20380 [iot_threa] [INFO ][MQTT][20380] (MQTT connection 100337e0) Network connection closed.
156 21622 [iot_threa] [INFO ][MQTT][21622] (MQTT connection 100337e0) Network connection destroyed.
157 21631 [iot_threa] [INFO ][MQTT][21631] MQTT library cleanup done.
158 21637 [iot_threa] [INFO ][DEMO][21637] Demo completed successfully.

LwIP_DHCP: dhcp stop.
Deinitializing WIFI ...
159 21772 [iot_threa] [INFO ][INIT][21772] SDK cleanup done.
160 21777 [iot_threa] [INFO ][DEMO][21777] -------DEMO FINISHED-------
```

6.3 Monitoring MQTT messages on the cloud

To subscribe to the MQTT topic with the AWS IoT MQTT client

1. Sign in to the AWS IoT console.
2. In the navigation pane, choose Test to open the MQTT client.
3. In Subscription topic, enter iotdemo/#, and then choose Subscribe to topic.


AWS IoT
Monitor
Onboard
Manage
Greengrass
Secure
Defend
Act
Rules
Destinations
Test
Software
Settings
Learn
MQTT client Info
Connected as iotconsole-1597037785600-0
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/#
Subscribe to 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)
Publish
Specify a topic and a message to publish with a QoS of 0.
Specify a topic to publish to, e.g. myTopic/1
Publish to topic
{
"message": "Hello from AWS IoT console"
}


AWS IoT
Monitor
Onboard
Manage
Greengrass
Secure
Defend
Act
Rules
Destinations
Test
Software
Settings
Learn
Subscriptions
iotdemo/#
Export Clear Pause
Subscribe to a topic
Publish to a topic
iotdemo/#
Publish
Specify a topic and a message to publish with a QoS of 0.
iotdemo/#
Publish to topic
{
"message": "Hello from AWS IoT console"
}
iotdemo/acknowledgements August 10, 2020, 13:41:07 (UTC+0800) Export Hide
We cannot display the message as JSON, and are instead displaying it as UTF-8 String.
Client has received PUBLISH 6 from server.
iotdemo/acknowledgements August 10, 2020, 13:41:07 (UTC+0800) Export Hide
We cannot display the message as JSON, and are instead displaying it as UTF-8 String.
Client has received PUBLISH 7 from server.
iotdemo/topic/2 August 10, 2020, 13:41:07 (UTC+0800) Export Hide
We cannot display the message as JSON, and are instead displaying it as UTF-8 String.
Hello world 9!
iotdemo/topic/1 August 10, 2020, 13:41:07 (UTC+0800) Export Hide

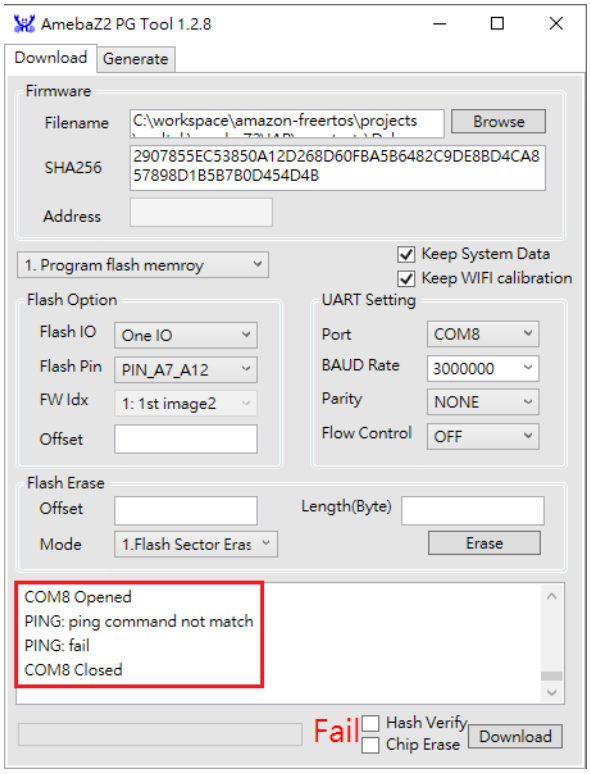
7 Troubleshooting

If these steps don't work, look at the device log in the serial terminal. You should see some text that indicates the source of the problem.

For general troubleshooting information about Getting Started with FreeRTOS, see Troubleshooting getting started.

7.1 Image Tool Download Fail

Please check device in UART_DOWNLOAD mode or not. Refer 5.3 for more detail.



7.2 ERROR: Invalid Key

Please check **WIFI_SSID** and **WIFI_PASSWORD** in in ambd_amazon-freertos/blob/master/demos/include/aws_clientcredential.h

```
Enter SSID for Soft AP started
3 1098 [example_a] Wi-Fi configuration successful.
4 1108 [iot_threa] [INFO ][DEMO][1108] ---------STARTING DEMO---------

5 1115 [iot_threa] [INFO ][INIT][1115] SDK successfully initialized.

LwIP_DHCP: dhcp stop.
Deinitializing WIFI ...
WIFI deinitialized
Initializing WIFI ...
WIFI initialized

Joining BSS by SSID ...

ERROR:Invalid Key
ERROR: Can't connect to AP
Joining BSS by SSID ...

ERROR:Invalid Key
ERROR: Can't connect to AP
Joining BSS by SSID ...
```

7.3 Failed to establish new MQTT connection

Please check **clientcredentialMQTT_BROKER_ENDPOINT** in ambd_amazon-freertos/blob/master/demos/include/aws_clientcredential.h

```
6 12508 [iot_threa] [INFO ][DEMO][12508] Successfully initialized the demo. Network type for the demo: 1
7 12517 [iot_threa] [INFO ][MQTT][12517] MQTT library successfully initialized.
8 12524 [iot_threa] [INFO ][DEMO][12524] MQTT demo client identifier is ameba-ota (length 9).
9 12624 [iot_threa] [ERROR][NET][12624] Failed to resolve [redacted].amazonaws.com.
10 12934 [iot_threa] [ERROR][MQTT][12934] Failed to establish new MQTT connection, error NETWORK ERROR.
11 12943 [iot_threa] [ERROR][DEMO][12943] MQTT CONNECT returned error NETWORK ERROR.
12 12951 [iot_threa] [INFO ][MQTT][12950] MQTT library cleanup done.
13 12957 [iot_threa] [ERROR][DEMO][12957] Error running demo.
Interface 0 IP address : 192.168.90.185
LwIP_DHCP: dhcp stop.
Deinitializing WIFI ...
14 13094 [iot_threa] [INFO ][INIT][13094] SDK cleanup done.
15 13099 [iot_threa] [INFO ][DEMO][13099] -------DEMO FINISHED-------
```

7.4 TLS_Connect fail

Please check **keyCLIENT_CERTIFICATE_PEM** and **keyCLIENT_PRIVATE_KEY_PEM** in ambd_amazon-freertos/blob/master/demos/include/aws_clientcredential_keys.h

```
8 13501 [iot_threa] [INFO ][DEMO][13501] Successfully initialized the demo. Network type for the demo: 1
9 13511 [iot_threa] [INFO ][MQTT][13511] MQTT library successfully initialized.
10 13518 [iot_threa] [INFO ][DEMO][13518] MQTT demo client identifier is ameba-ota (length 9).
11 20102 [iot_threa] ERROR: Private key not found. 12 20107 [iot_threa] TLS Connect fail (0x7d4, [redacted].amazonaws.com)
13 20115 [iot_threa] [ERROR][NET][20115] Failed to establish new connection. Socket status: -1.
14 20424 [iot_threa] [ERROR][MQTT][20424] Failed to establish new MQTT connection, error NETWORK ERROR.
15 20433 [iot_threa] [ERROR][DEMO][20433] MQTT CONNECT returned error NETWORK ERROR.
16 20441 [iot_threa] [INFO ][MQTT][20441] MQTT library cleanup done.
17 20447 [iot_threa] [ERROR][DEMO][20447] Error running demo.
Interface 0 IP address : 192.168.90.185
LwIP_DHCP: dhcp stop.
Deinitializing WIFI ...
18 20586 [iot_threa] [INFO ][INIT][20586] SDK cleanup done.
19 20591 [iot_threa] [INFO ][DEMO][20591] -------DEMO FINISHED-------
```