# AmebaDplus Amazon FreeRTOS Getting Started Guide

**Realtek Semiconductor Corp.**
**No. 2, Innovation Road II, Hsinchu Science Park, Hsinchu 300, Taiwan**
**Tel.: +886-3-578-0211. Fax: +886-3-577-6047**
**www.realtek.com**

**COPYRIGHT**



**DISCLAIMER**

Please Read Carefully:

Realtek Semiconductor Corp., (Realtek) reserves the right to make corrections, enhancements, improvements and other changes to its products and services. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

Reproduction of significant portions in Realtek data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Realtek is not responsible or liable for such reproduced documentation. Information of third parties may be subject to additional restrictions.

Buyers and others who are developing systems that incorporate Realtek products (collectively, “Customers”) understand and agree that Customers remain responsible for using their independent analysis, evaluation and judgment in designing their applications and that Customers have full and exclusive responsibility to assure the safety of Customers' applications and compliance of their applications (and of all Realtek products used in or for Customers’ applications) with all applicable regulations, laws and other applicable requirements. Designer represents that, with respect to their applications, Customer has all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. Customer agrees that prior to using or distributing any applications that include Realtek products, Customer will thoroughly test such applications and the functionality of such Realtek products as used in such applications.

Realtek’s provision of technical, application or other design advice, quality characterization, reliability data or other services or information, including, but not limited to, reference designs and materials relating to evaluation kits, (collectively, “Resources”) are intended to assist designers who are developing applications that incorporate Realtek products; by downloading, accessing or using Realtek’s Resources in any way, Customer (individually or, if Customer is acting on behalf of a company, Customer’s company) agrees to use any particular Realtek Resources solely for this purpose and subject to the terms of this Notice.

Realtek’s provision of Realtek Resources does not expand or otherwise alter Realtek’s applicable published warranties or warranty disclaimers for Realtek’s products, and no additional obligations or liabilities arise from Realtek providing such Realtek Resources. Realtek reserves the right to make corrections, enhancements, improvements and other changes to its Realtek Resources. Realtek has not conducted any testing other than that specifically described in the published documentation for a particular Realtek Resource.

Customer is authorized to use, copy and modify any individual Realtek Resource only in connection with the development of applications that include the Realtek product(s) identified in such Realtek Resource. No other license, express or implied, by estoppel or otherwise to any other Realtek intellectual property right, and no license to any technology or intellectual property right of Realtek or any third party is granted herein, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which Realtek products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of Realtek Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from Realtek under the patents or other Realtek’s intellectual property.

Realtek’s Resources are provided “as is” and with all faults. Realtek disclaims all other warranties or representations, express or implied, regarding resources or use thereof, including but not limited to accuracy or completeness, title, any epidemic failure warranty and any implied warranties of merchantability, fitness for a particular purpose, and non-infringement of any third party intellectual property rights.

Realtek shall not be liable for and shall not defend or indemnify Customer against any claim, including but not limited to any infringement claim that related to or is based on any combination of products even if described in Realtek Resources or otherwise. In no event shall Realtek be liable for any actual, direct, special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of Realtek’s Resources or use thereof, and regardless of whether Realtek has been advised of the possibility of such damages. Realtek is not responsible for any failure to meet such industry standard requirements.

Where Realtek specifically promotes products as facilitating functional safety or as compliant with industry functional safety standards, such products are intended to help enable customers to design and create their own applications that meet applicable functional safety standards and requirements. Using products in an application does not by itself establish any safety features in the application. Customers must ensure compliance with safety-related requirements and standards applicable to their applications. Designer may not use any Realtek products in life-critical medical equipment unless authorized officers of the parties have executed a special contract specifically governing such use. Life-critical medical equipment is medical equipment where failure of such equipment would cause serious bodily injury or death. Such equipment includes, without limitation, all medical devices identified by the U.S.FDA as Class III devices and equivalent classifications outside the U.S.

Customers agree that it has the necessary expertise to select the product with the appropriate qualification designation for their applications and that proper product selection is at Customers' own risk. Customers are solely responsible for compliance with all legal and regulatory requirements in connection with such selection.

Customer will fully indemnify Realtek and its representatives against any damages, costs, losses, and/or liabilities arising out of Designer's non-compliance with the terms and provisions of this Notice.

**TRADEMARKS**

Realtek is a trademark of Realtek Semiconductor Corporation. Other names mentioned in this document are trademarks/registered trademarks of their respective owners.

**USING THIS DOCUMENT**

Though every effort has been made to ensure that this document is current and accurate, more information may have become available subsequent to the production of this guide.

# 1 Configure AWS IoT Core

## 1.1 Set up your AWS account and Permissions

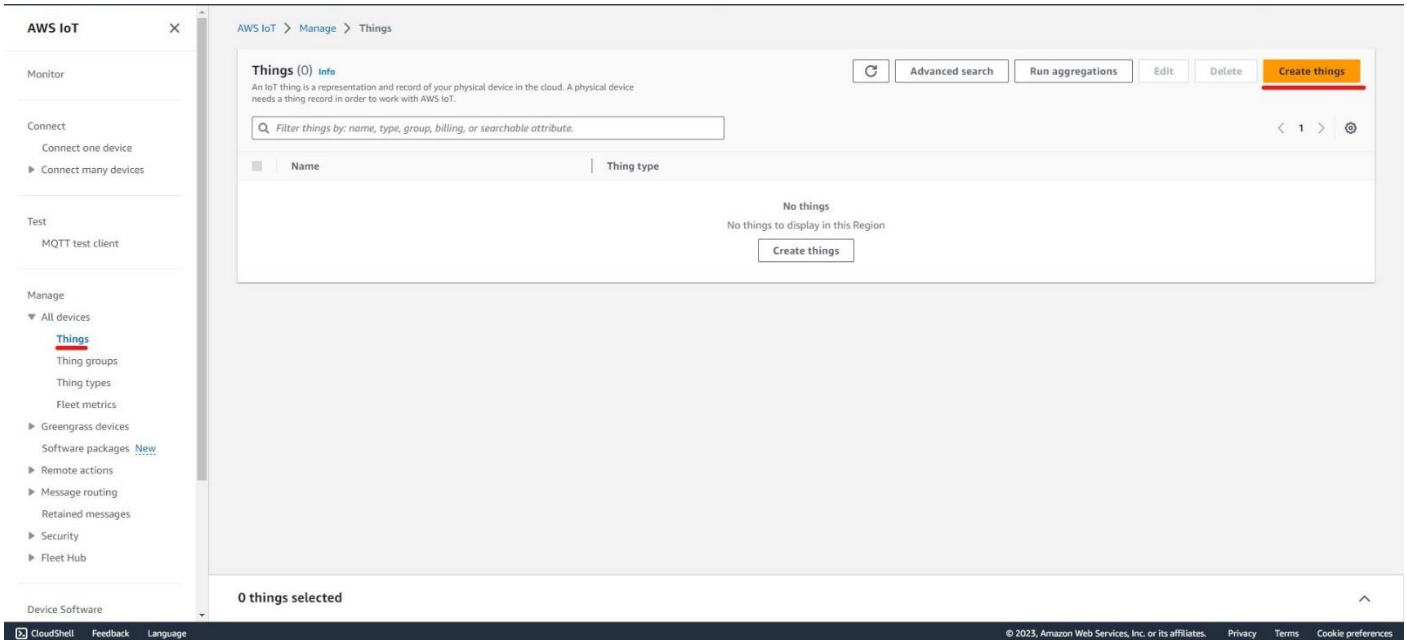
Refer to the instructions at Set up your AWS Account https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html. Follow the steps outlined in these sections to create your account and a user and get started:

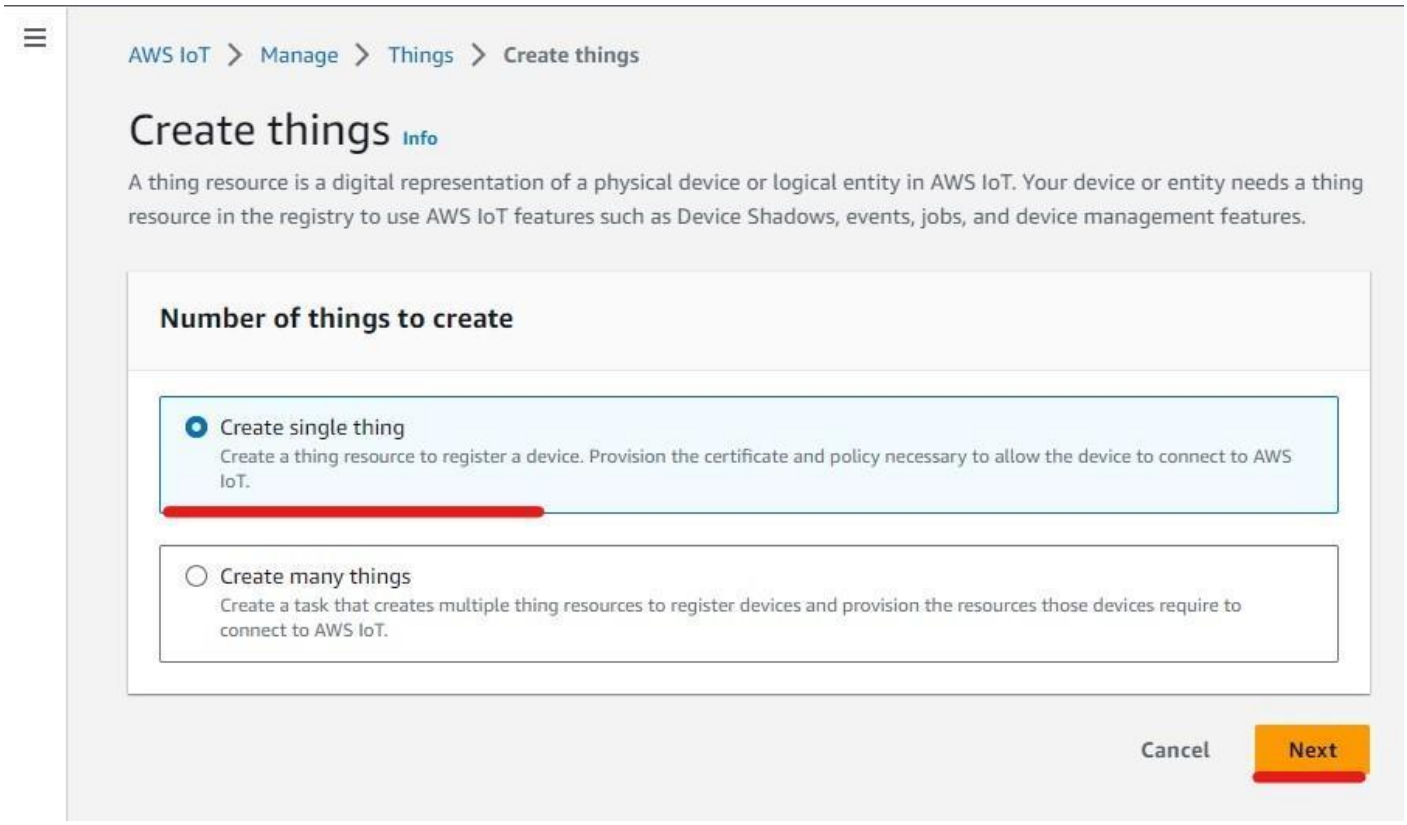
- Sign up for an AWS account (https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html#sign-up-for-aws)
- Create an administrative user (https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html#create-an-admin)
- Open the AWS IoT console(https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html#iot-console-signin)

Please pay special attention to the Notes in AWS webpage.

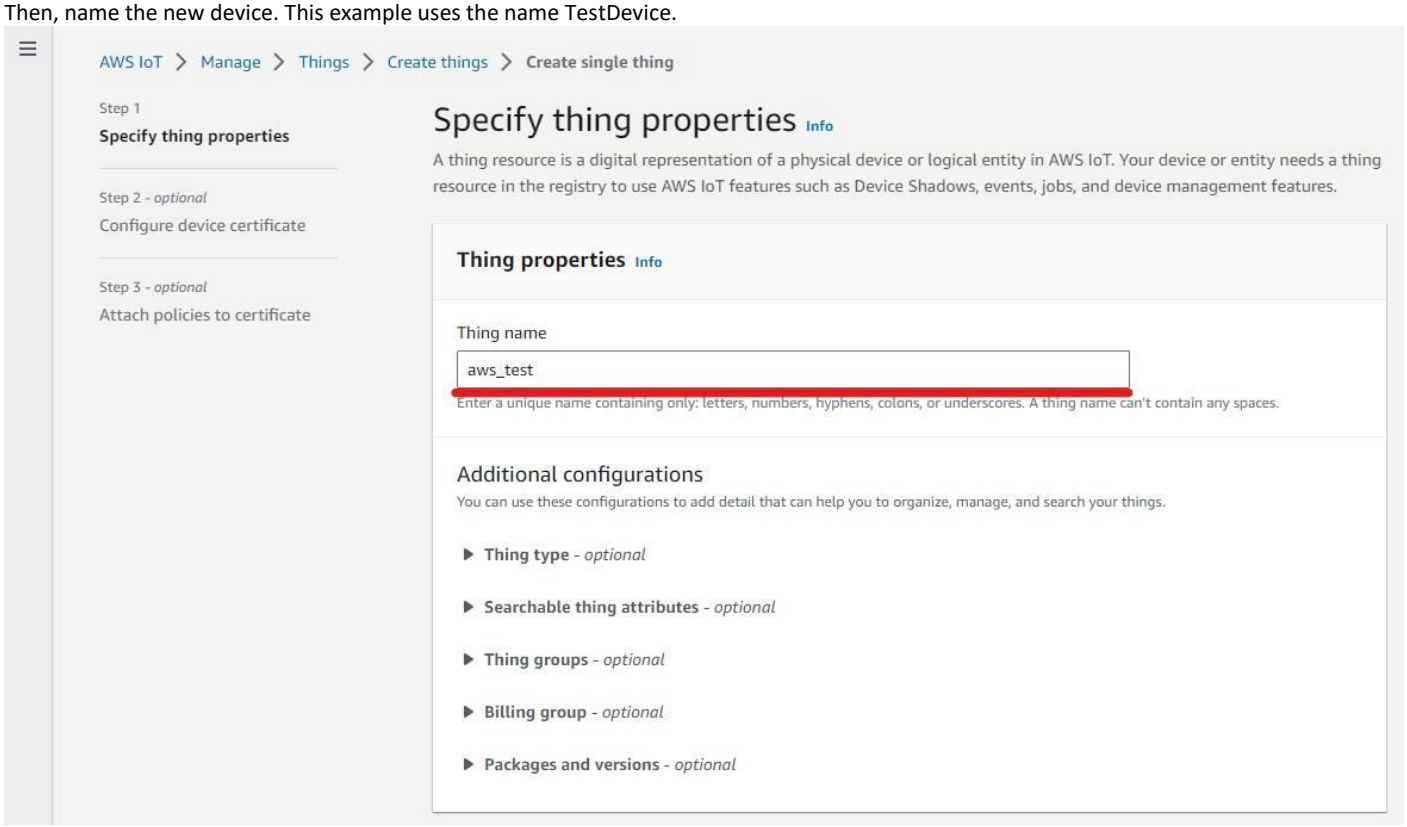
## 1.2 Create a New Device

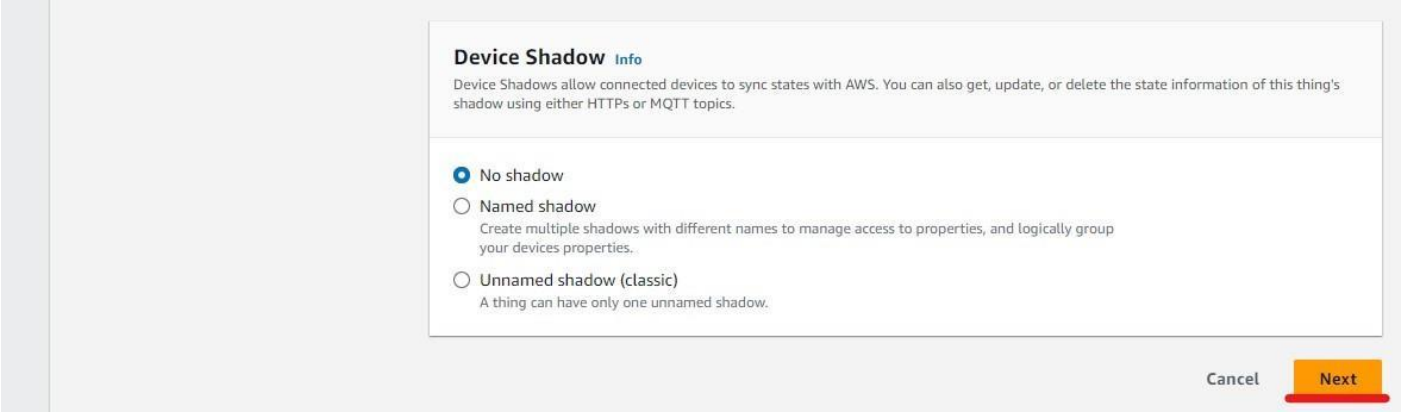
To create a new device, navigate to Manage -> Things in the left-hand navigation menu. Then click “Create things”.



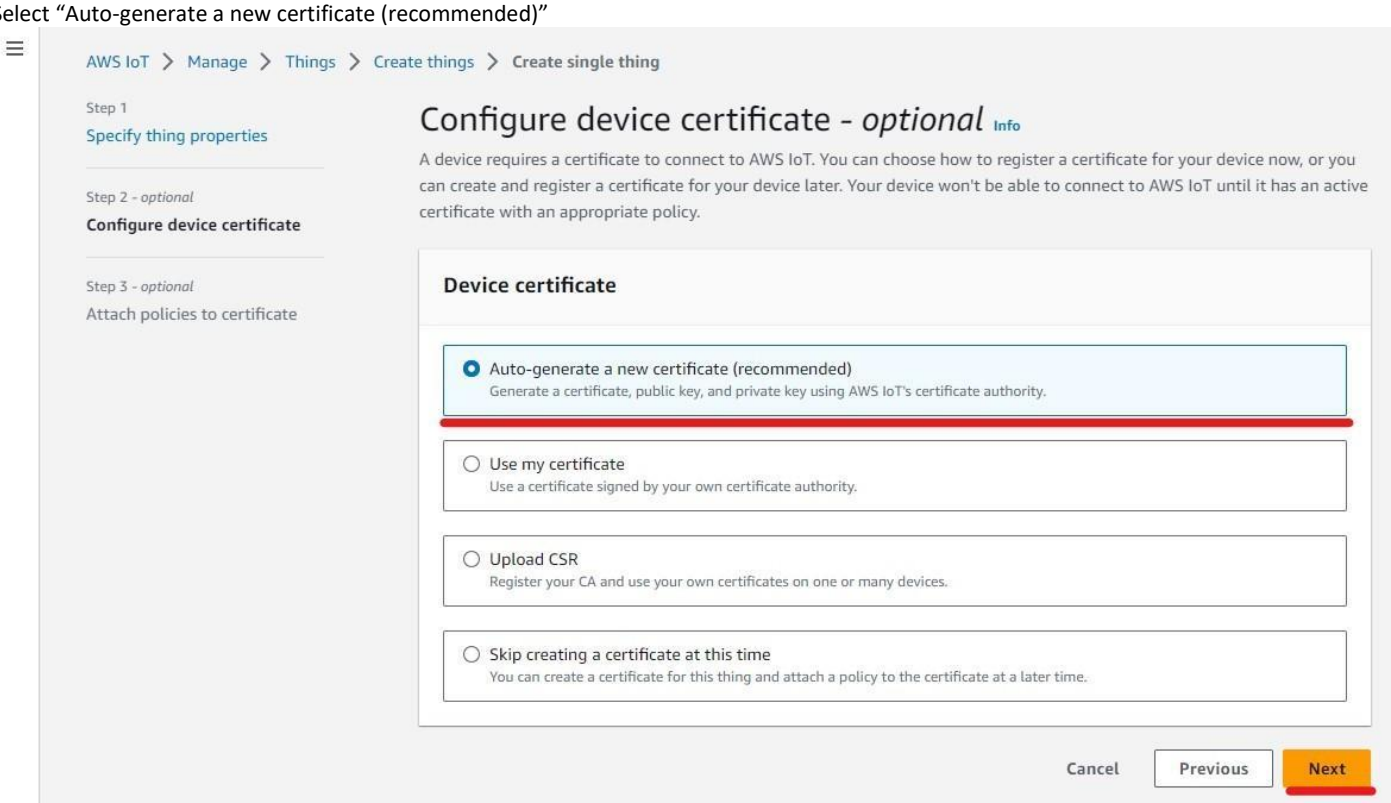



Then, name the new device. This example uses the name TestDevice.



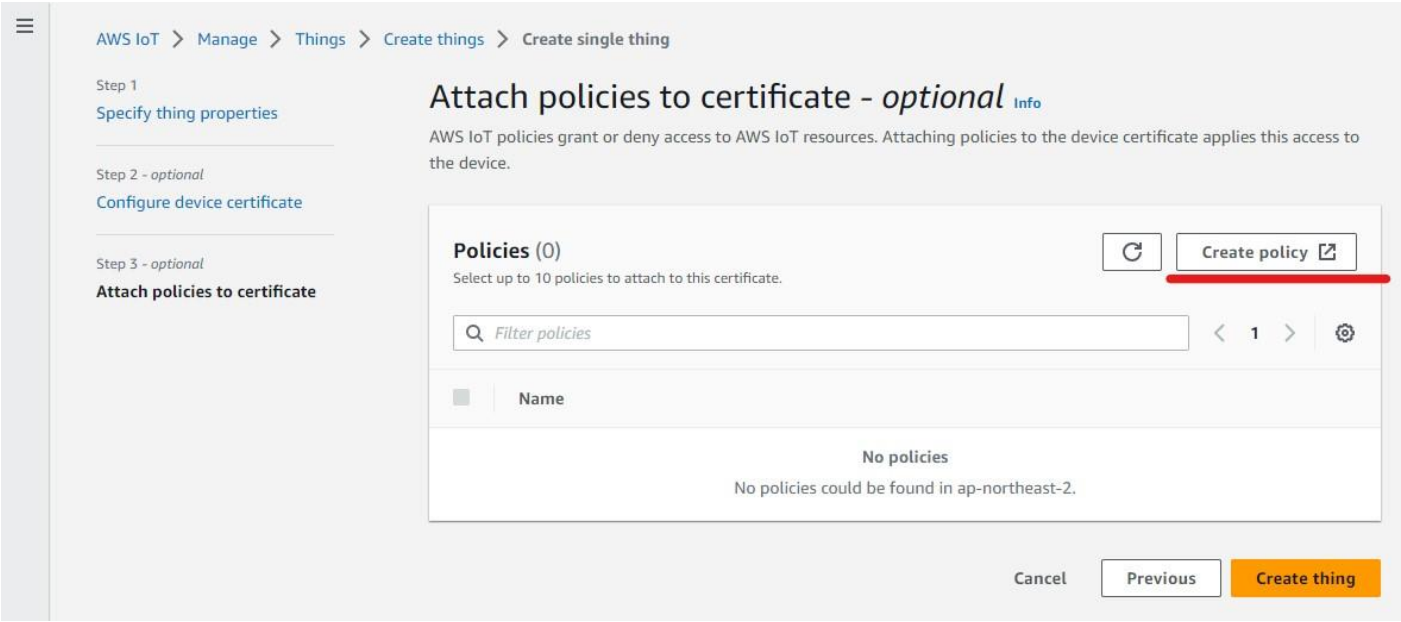



Select “Auto-generate a new certificate (recommended)”

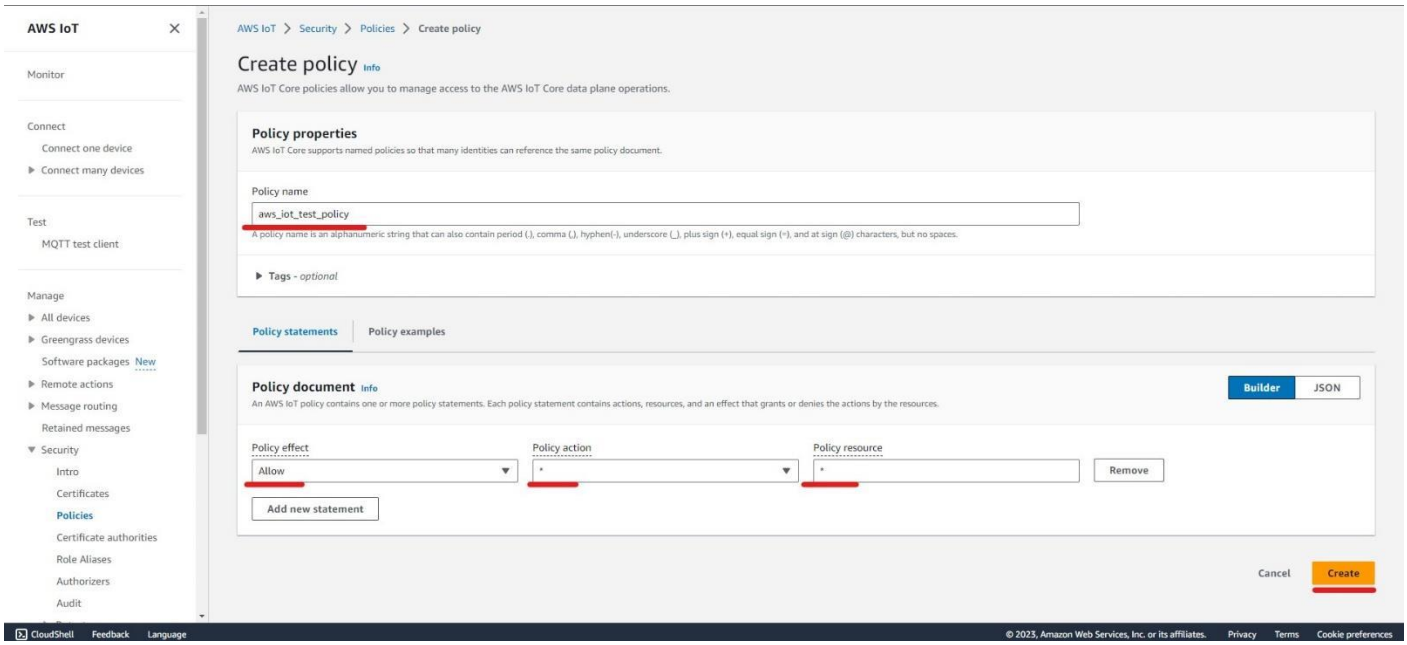


## 1.3 Create a policy

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



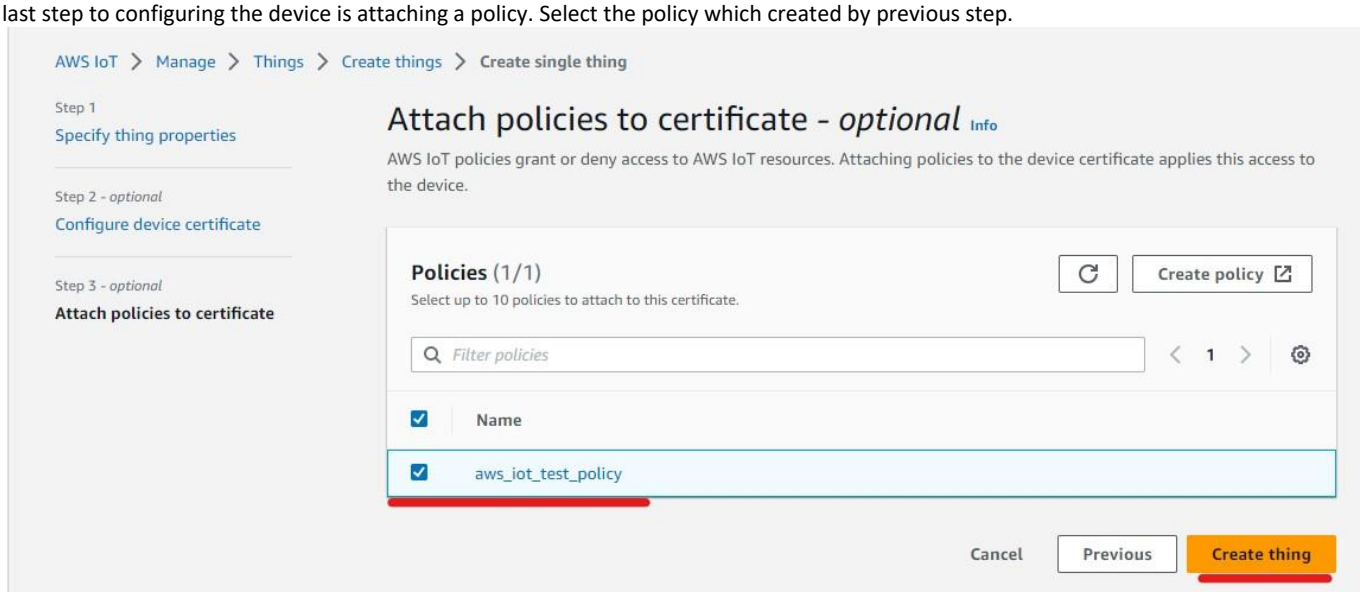

NOTE – this policy grants unrestricted access for all iot operations, and is to be used only in a development environment. For non-dev environments, all devices in your fleet must have credentials with privileges that authorize intended actions only, which include (but not limited to) AWS IoT MQTT actions 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




**Important: The policy is for reference. Please identify the permission policies that best meet your business and security requirements**

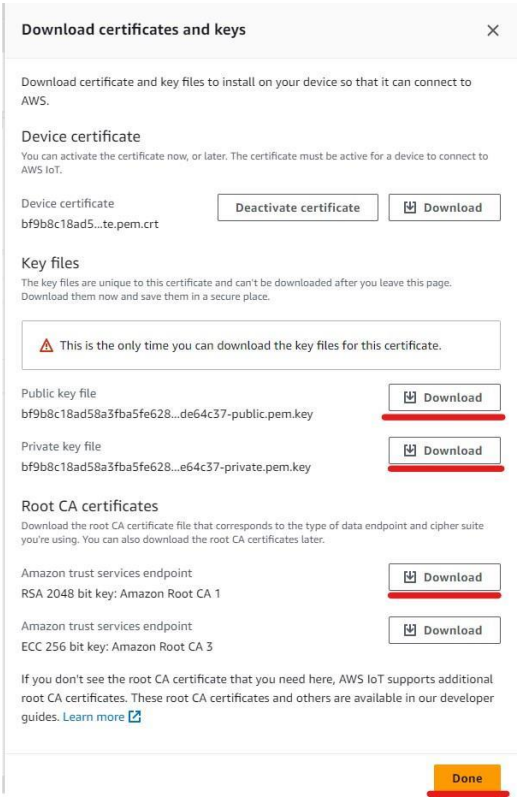
## 1.4 Attach Policy

The last step to configuring the device is attaching a policy. Select the policy which created by previous step.



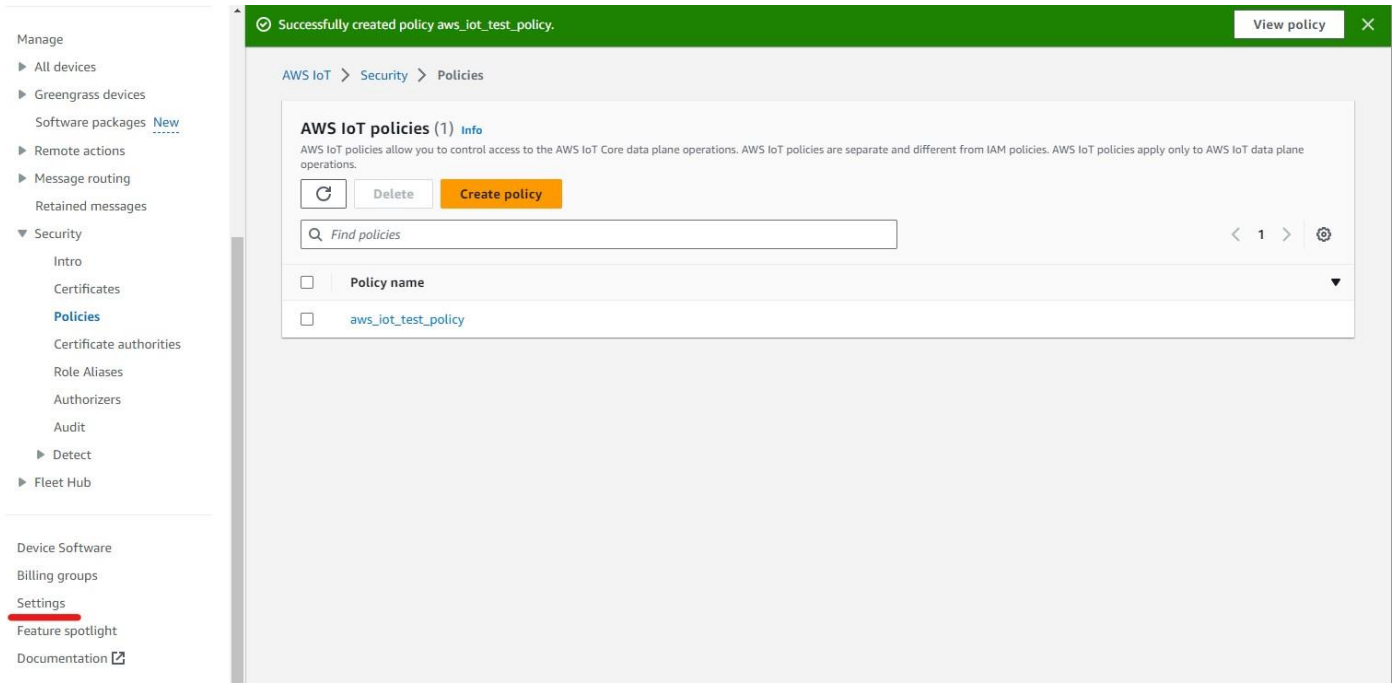

## 1.5 Download Certificates

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

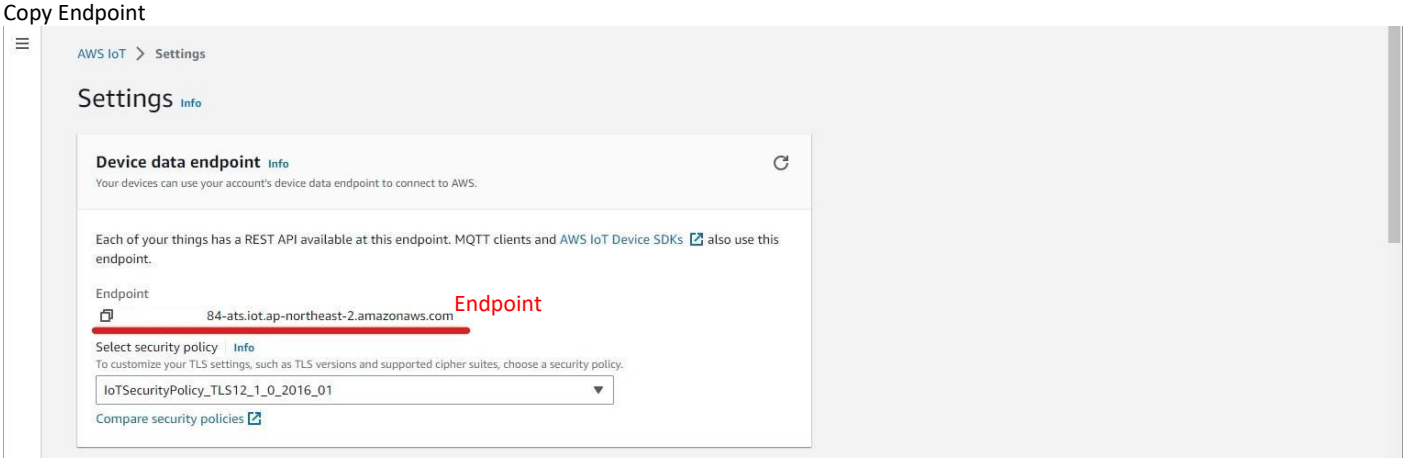


# 2 Configure AmebaDplus Amazon FreeRTOS

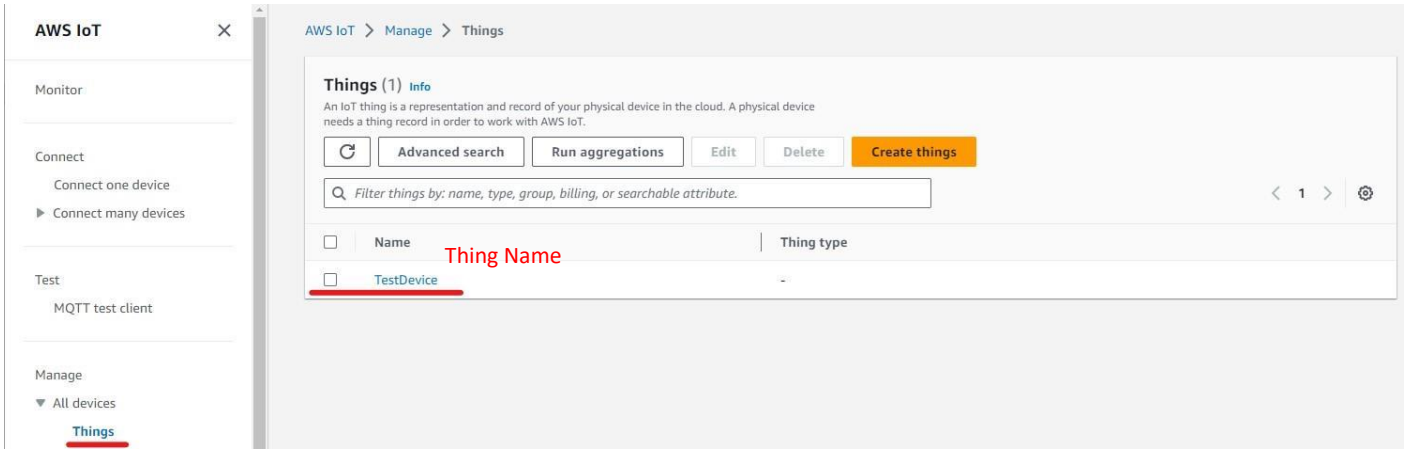
## 2.1 Get Endpoint by AWS IoT Core




Copy Endpoint



## 2.2 Get Thing Name



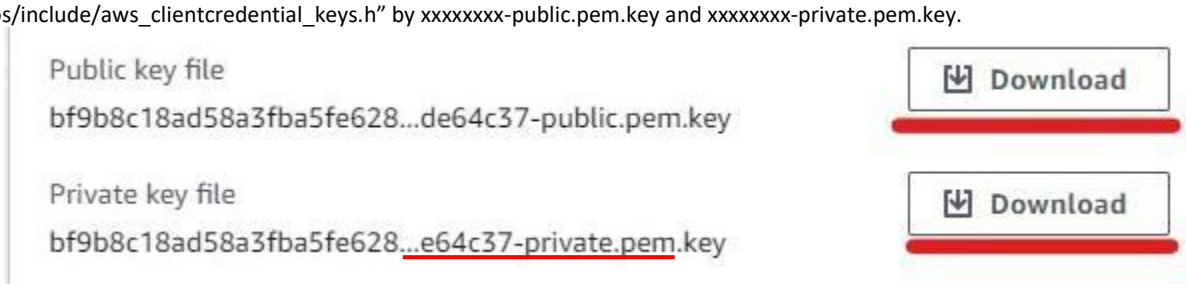

## 2.3 Setup IoT Core Information with AmebaDplus Amazon FreeRTOS

Setup BROKER_ENDPOINT, THING_NAME, WIFI_SSID, PASSWORD in “component/application/amazon/amazon-freertos/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__ */
```

### 2.3.1 Setup Thing's Private Key and Certificate

Fill keyCLIENT_CERTIFICATE_PEM and keyCLIENT_PRIVATE_KEY_PEM in "component/application/amazon/amazon-freertos/demos/include/aws_clientcredential_keys.h" by xxxxxxxx-public.pem.key and xxxxxxxx-private.pem.key.




It can done by "component/application/amazon/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"\
"                                                                \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"\
"pOWEuLUuz2FAv1noAbN/6OQ8H/PTOAFJT/ghA04GnIUF0kjSzY60ehS2mVp6neP+\n"\
"AZjzZ6QJYlb5/PFz9oES448kpyaAoS2ke86+R4r4YOMBK+I5RVbfSQ==\n"\
"-----END RSA PRIVATE KEY-----\n"
```

### 2.3.2 Enable FreeRTOS demo on AmebaDplus

Find aws_main.c in component/application/amazon/amazon-freertos/ports/amebaDplus/aws_main.c and enable **RunCoreMqttMutualAuthDemo**

```
int aws_main( void )
{
    /* Create tasks that are not dependent on the Wi-Fi being initialized. */
    xLoggingTaskInitialize( mainLOGGING_TASK_STACK_SIZE,
                            tskIDLE_PRIORITY+6,
                            mainLOGGING_MESSAGE_QUEUE_LENGTH );

    CRYPTO_ConfigureThreading();

    //mqtt mutual auto demo
    RunCoreMqttMutualAuthDemo(0, NULL, NULL, NULL, NULL);

    //http mutual auto demo
    //RunCoreHttpMutualAuthDemo(0, NULL, NULL, NULL, NULL);

    //device shadow demo
    //RunDeviceShadowDemo(0, NULL, NULL, NULL, NULL);

    //device defender demo
    //RunDeviceDefenderDemo(0, NULL, NULL, NULL, NULL);

    // ota over mqtt demo
    //RunOtaCoreMqttDemo(0, NULL, NULL, NULL, NULL);

    return 0;
}
```

Now you can start to compile AmebaDplus Amazon FreeRTOS

# 3 MQTT Demo

## 3.1 Get Device Log

Install Tera Term to get device log

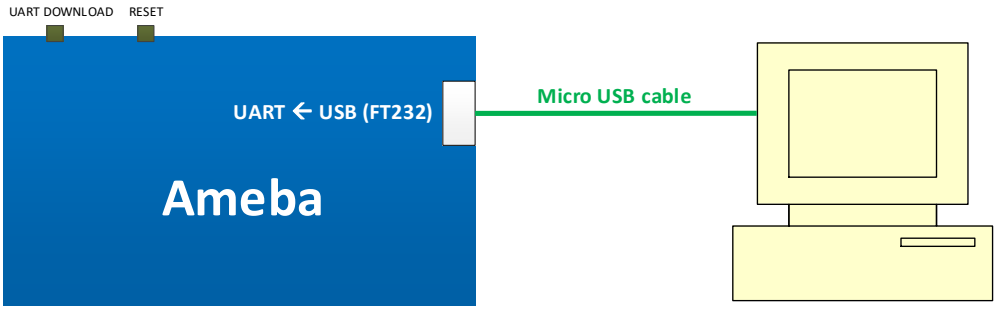
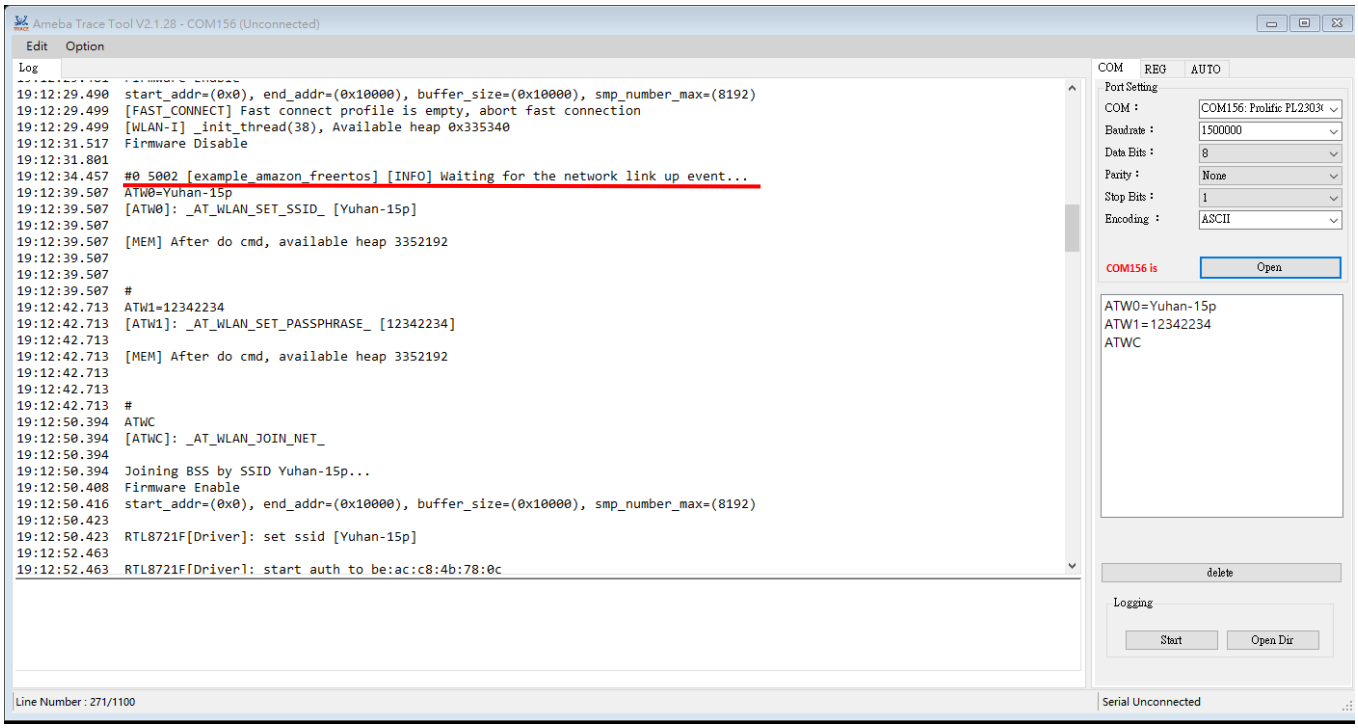


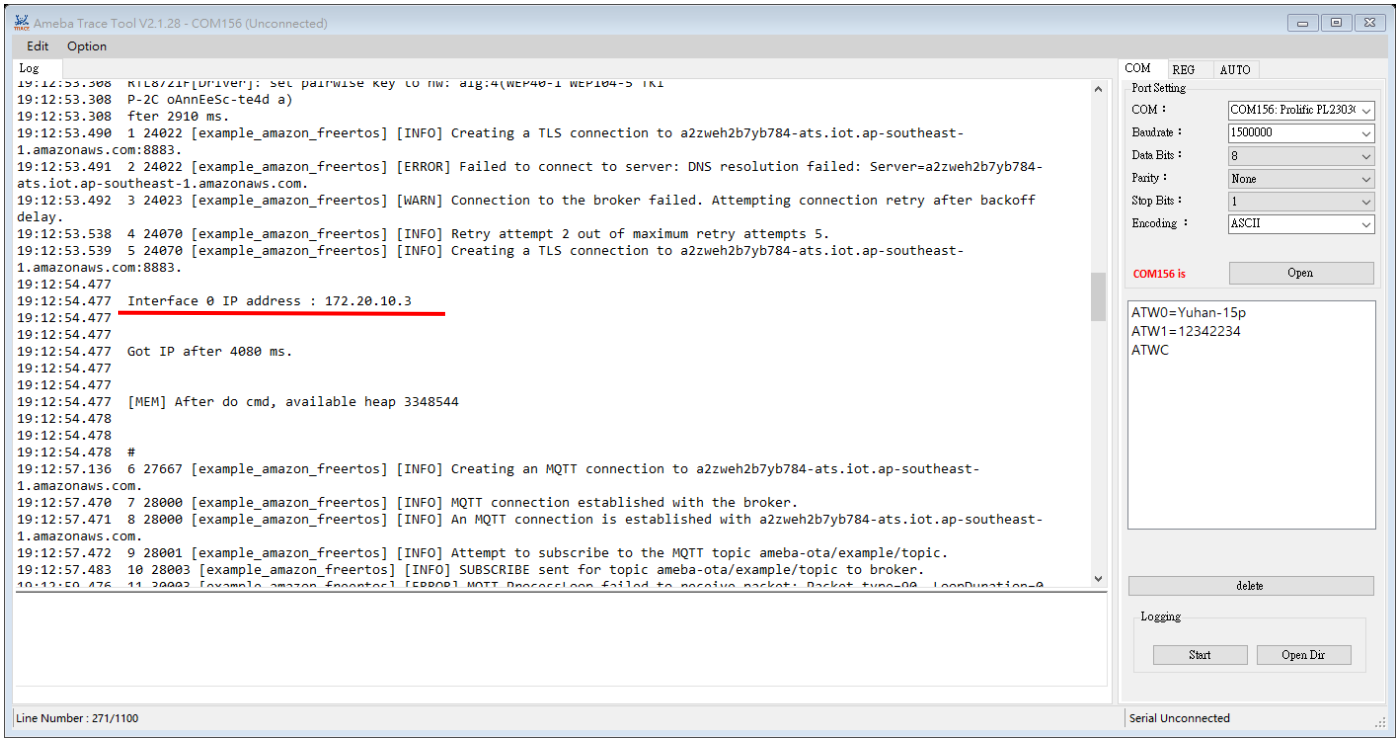

Fig 3-1 Hardware setup

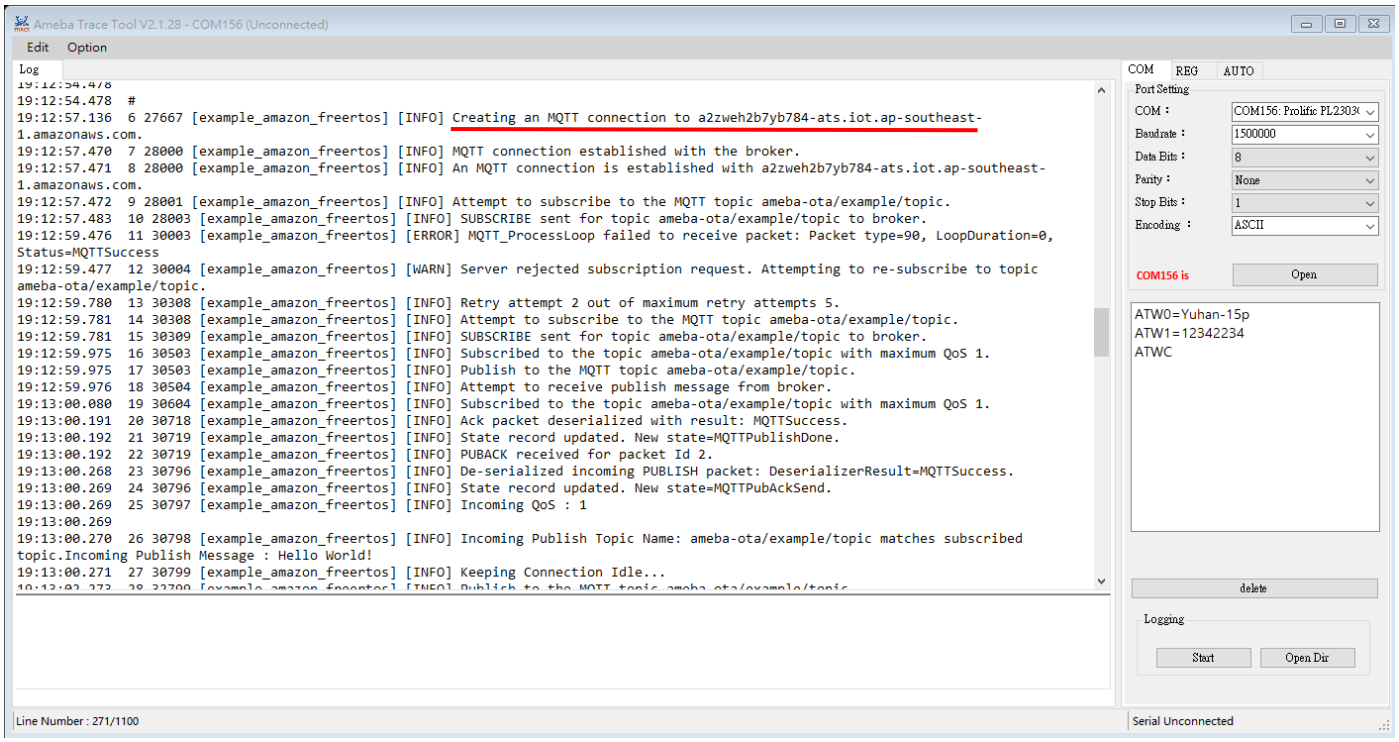
## 3.2 Run MQTT Demo

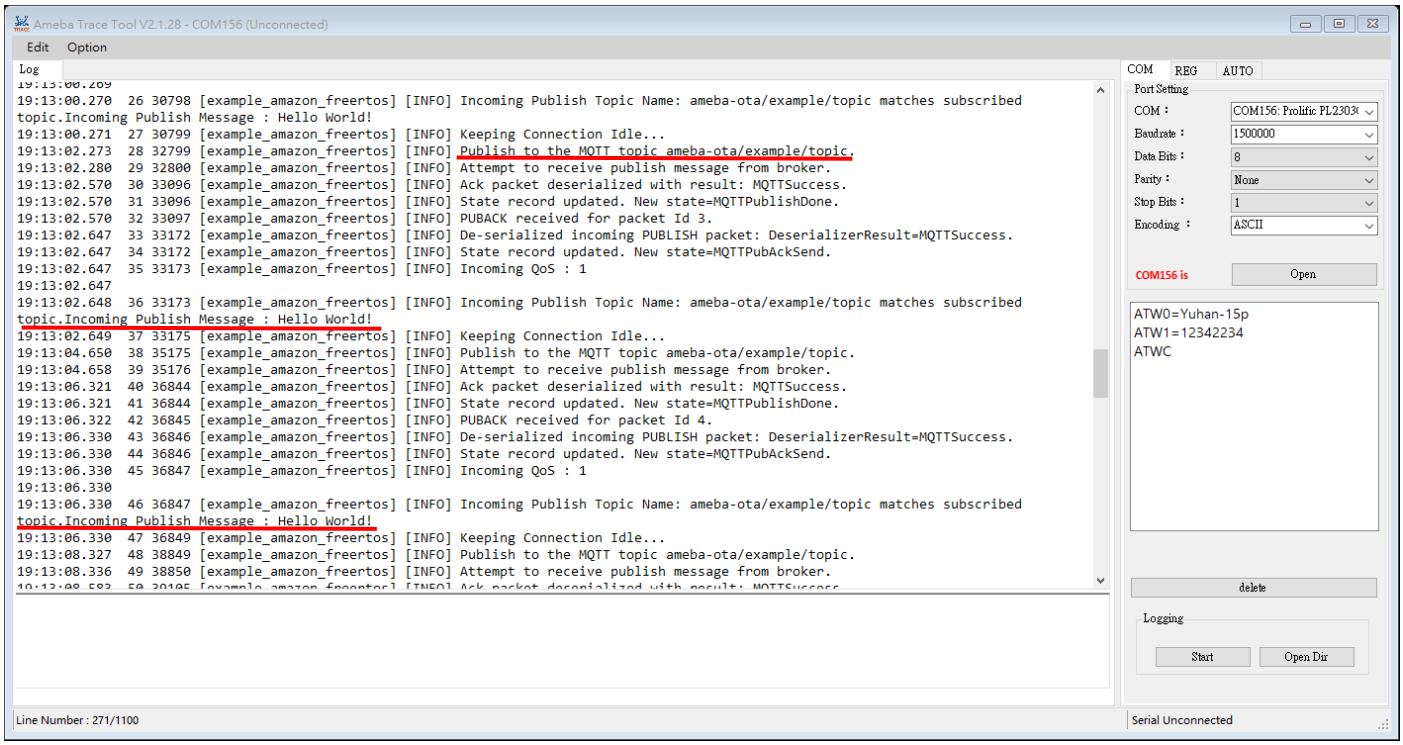
Default setting of SDK are enable MQTT demo. Once the AmebaDplus EVB has rebooted, the application will automatically start run MQTT demo and communicate to IoT Core.



Use ATcmd to connect wifi.



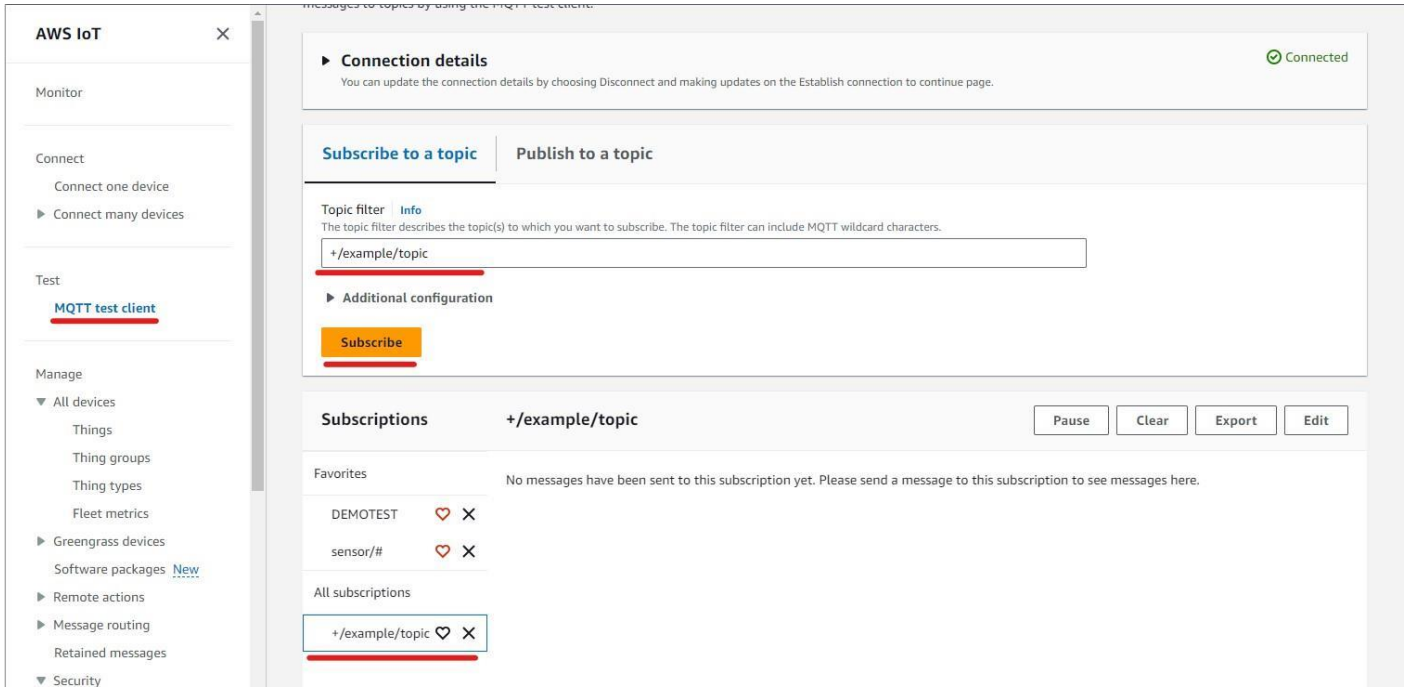



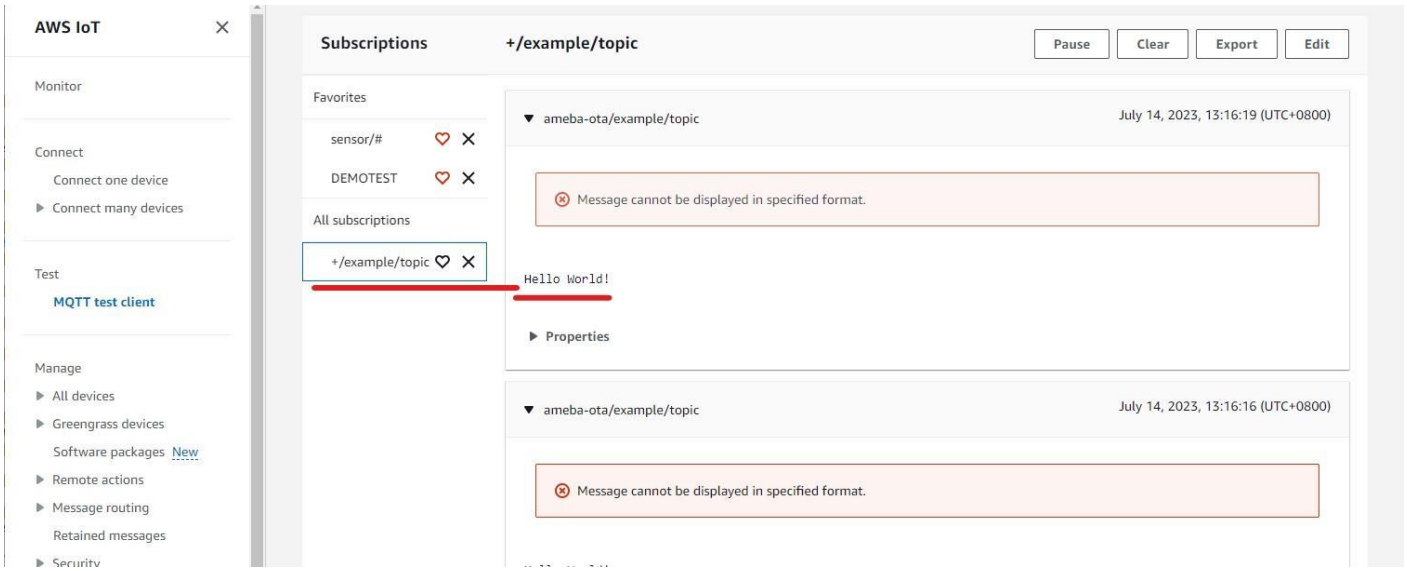



## 3.3 Monitoring MQTT messages on the cloud

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 “+/example/topic”, and then choose Subscribe to topic.





# 4 OTA Demo

## 4.1 OTA Update Prerequisites

Please refer to the AWS official guide (https://docs.aws.amazon.com/freertos/latest/userguide/ota-prereqs.html) and finish the following steps:

Step 1. Prerequisites for OTA updates using MQTT(https://docs.aws.amazon.com/freertos/latest/userguide/ota-mqtt-freertos.html)
Step 2. Create an Amazon S3 bucket to store your update(https://docs.aws.amazon.com/freertos/latest/userguide/dg-ota-bucket.html)
Step 3. Create an OTA Update service role(https://docs.aws.amazon.com/freertos/latest/userguide/create-service-role.html)
Step 4. Create an OTA user policy(https://docs.aws.amazon.com/freertos/latest/userguide/create-ota-user-policy.html)
…
**Step 5.** Create esdsasigner.key and ecdsasigner.crt by openSSL
you can create the key and certification by running:
**$ openssl ecparam -name prime256v1 -genkey -out ecdsa-sha256-signer-tmp.key.pem**
**$ openssl pkcs8 -topk8 -inform pem -in ecdsa-sha256-signer-tmp.key.pem -outform pem -nocrypt -out ecdsa-sha256-signer.key.pem**
**$ openssl req -new -x509 -days 3650 -key ecdsa-sha256-signer.key.pem -out ecdsa-sha256-signer.crt.pem**
…
Step 6. Add certificate pem(ecdsa-sha256-signer.crt.pem) into component\common\application\amazon\amazon-freertos\ports\amebaD\config_files\ota_demo_config.h

```
/**
 * @brief Certificate used for validating code signing signatures in the OTA PAL.
 */
#ifndef otapalconfigCODE_SIGNING_CERTIFICATE
#define otapalconfigCODE_SIGNING_CERTIFICATE \
"-----BEGIN CERTIFICATE-----\n" \
"                                                                \
"                                                                \
"                                                                \
"                                                                \
"                                                                \
"                                                                \
"                                                                \
"
"-----END CERTIFICATE-----"
#endif
```

## 4.2 Set the App Version to Image File

Step 7. The app number in “component\common\application\amazon\amazon-freertos\ports\amebaD\config_files\ota_demo_config.h ” decide the version of application code:

```
/**
 * @brief Major version of the firmware.
 *
 * This is used in the OTA demo to set the appFirmwareVersion variable
 * that is declared in the ota_appversion32.h file in the OTA library.
 */
#ifndef APP_VERSION_MAJOR
    #define APP_VERSION_MAJOR    0 /* TODO */
#endif

/**
 * @brief Minor version of the firmware.
 *
 * This is used in the OTA demo to set the appFirmwareVersion variable
 * that is declared in the ota_appversion32.h file in the OTA library.
 */
#ifndef APP_VERSION_MINOR
    #define APP_VERSION_MINOR    9 /* TODO */
#endif

/**
 * @brief Build version of the firmware.
 *
 * This is used in the OTA demo to set the appFirmwareVersion variable
 * that is declared in the ota_appversion32.h file in the OTA library.
 */
#ifndef APP_VERSION_BUILD
    #define APP_VERSION_BUILD    2 /* TODO */
#endif
```

Also, change manifest. Please find more detail about manifest in application note.

```
"//": "cert/app share IMG_ID/IMG_VER, rdp img is in app",
"app":
{
        "IMG_ID": "1",
        "IMG_VER_MAJOR": 1,
        "IMG_VER_MINOR": 1,
        "SEC_EPOCH": 1,

        "HASH_ALG": "sha256",

        "RSIP_IV": "2132536475 86a7b8"
},
```

Now, configure the **aws_demo_config.h** for OTA demo. Then, build the project and get the image file (.bin).

```
int aws_main( void )
{
    /* Create tasks that are not dependent on the Wi-Fi being initialized. */
    xLoggingTaskInitialize( mainLOGGING_TASK_STACK_SIZE,
                            tskIDLE_PRIORITY+6,
                            mainLOGGING_MESSAGE_QUEUE_LENGTH );

    CRYPTO_ConfigureThreading();

    //mqtt mutual auto demo
    //RunCoreMqttMutualAuthDemo(0, NULL, NULL, NULL, NULL);

    //http mutual auto demo
    //RunCoreHttpMutualAuthDemo(0, NULL, NULL, NULL, NULL);

    //device shadow demo
    //RunDeviceShadowDemo(0, NULL, NULL, NULL, NULL);

    //device defender demo
    //RunDeviceDefenderDemo(0, NULL, NULL, NULL, NULL);

    // ota over mqtt demo
    RunOtaCoreMqttDemo(0, NULL, NULL, NULL, NULL);

    return 0;
}
```

Please note that the newer image file must have the bigger app version number. So, now, you need two image file to perform this demo.

- One image with older version should be downloaded to your AmebaDplus, and wait the OTA job coming.
- Another image with newer version will be uploaded to S3 bucket. Then, create a new job for OTA.

**Note:** newer version image file should be signed by a private key before uploading. Next section will introduce how to sign the image.

## 4.3 How Custom Signed Image File is Created

The OTA binary (OTA_All.bin) will be generated when compile SDK and it locate at project/realtek_amebaDplus_va0_example/GCCRELEASE/auto_build/

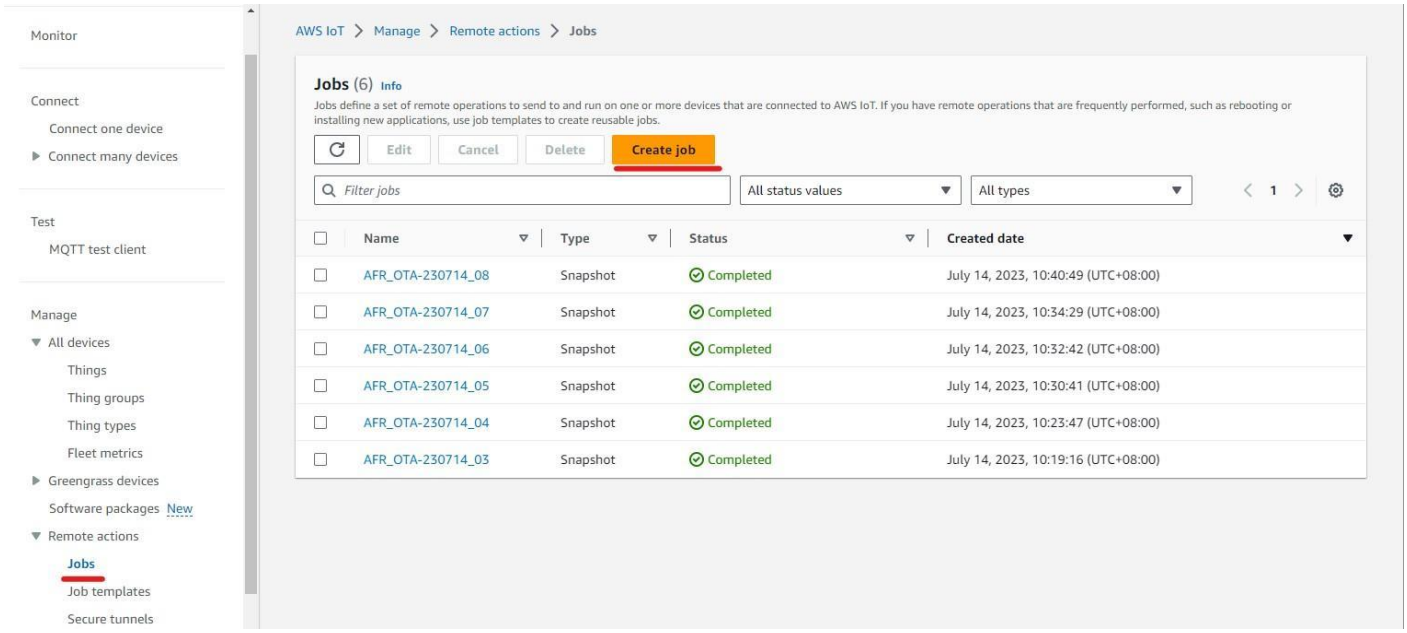
The signature of OTA_All.bin can be got by shell script (component/application/amazon/amazon_ota_tools/signer_gcc.sh. The signature will need for create AWS IoT OTA jobs.


After getting the custom signed of auto_build/**OTA_All.bin**, you can upload it to the S3 bucket.

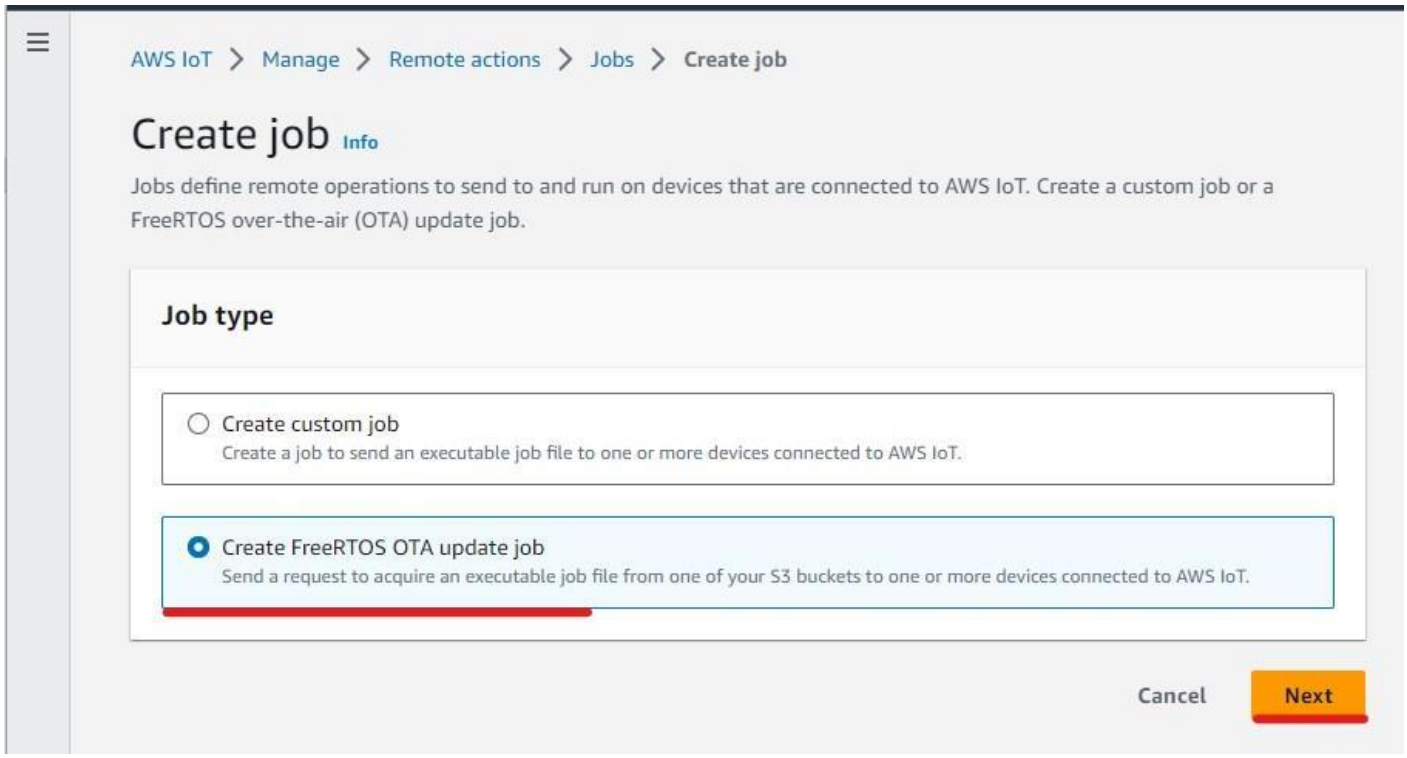
## 4.4 How to Trigger a Custom Signed OTA Job in Amazon AWS IOT Core

Go to AWS IoT Core https://console.aws.amazon.com/iot?p=icr&cp=bn&ad=c. Then, follow the following steps to create an AWS OTA task for AmebaDplus:

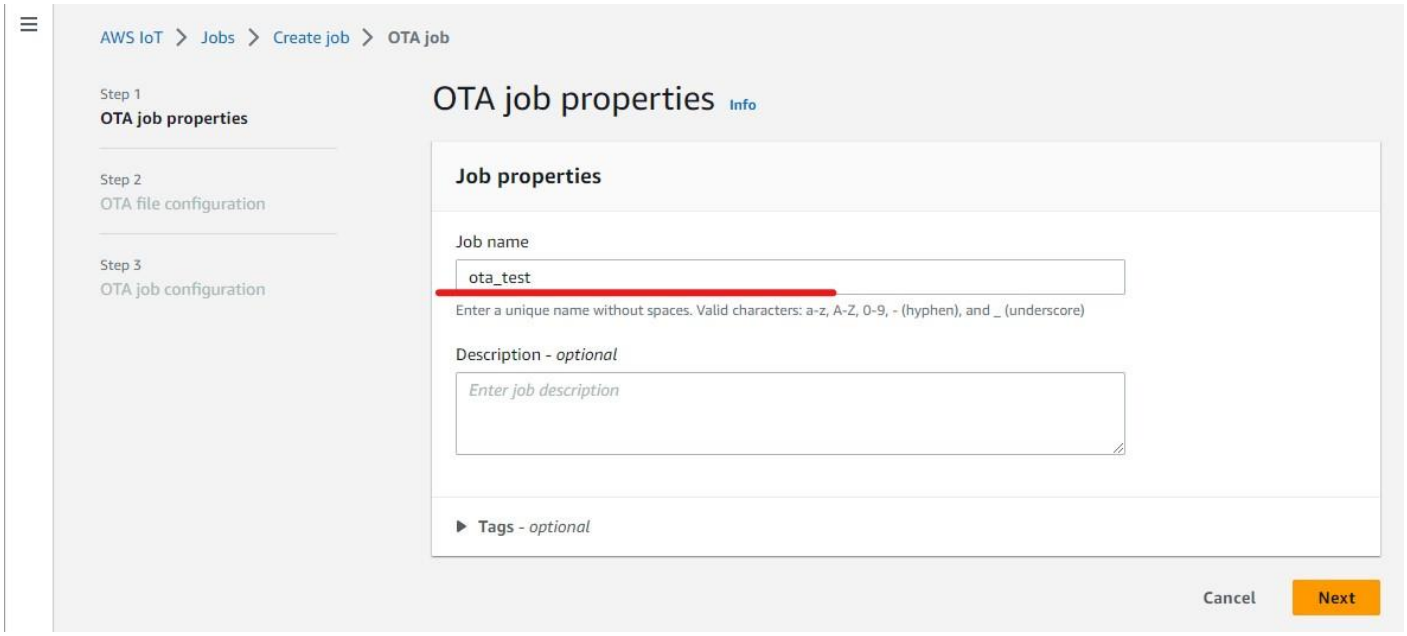
Step 1. Click on 'Create OTA update job', select your job type and then click next.



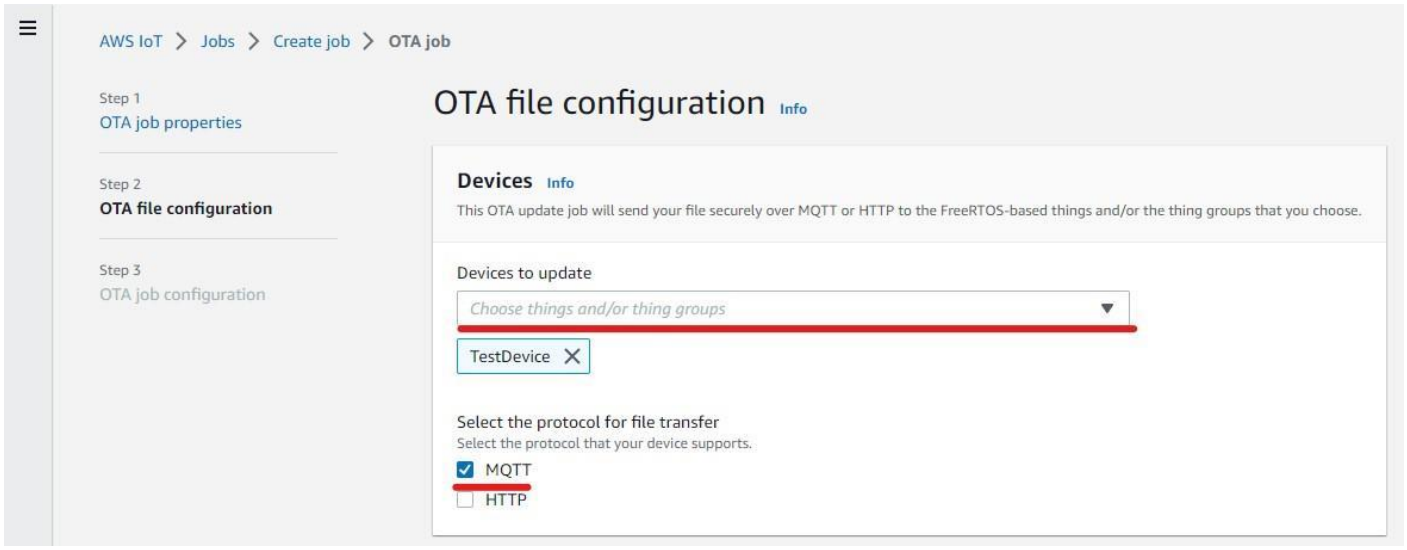

...



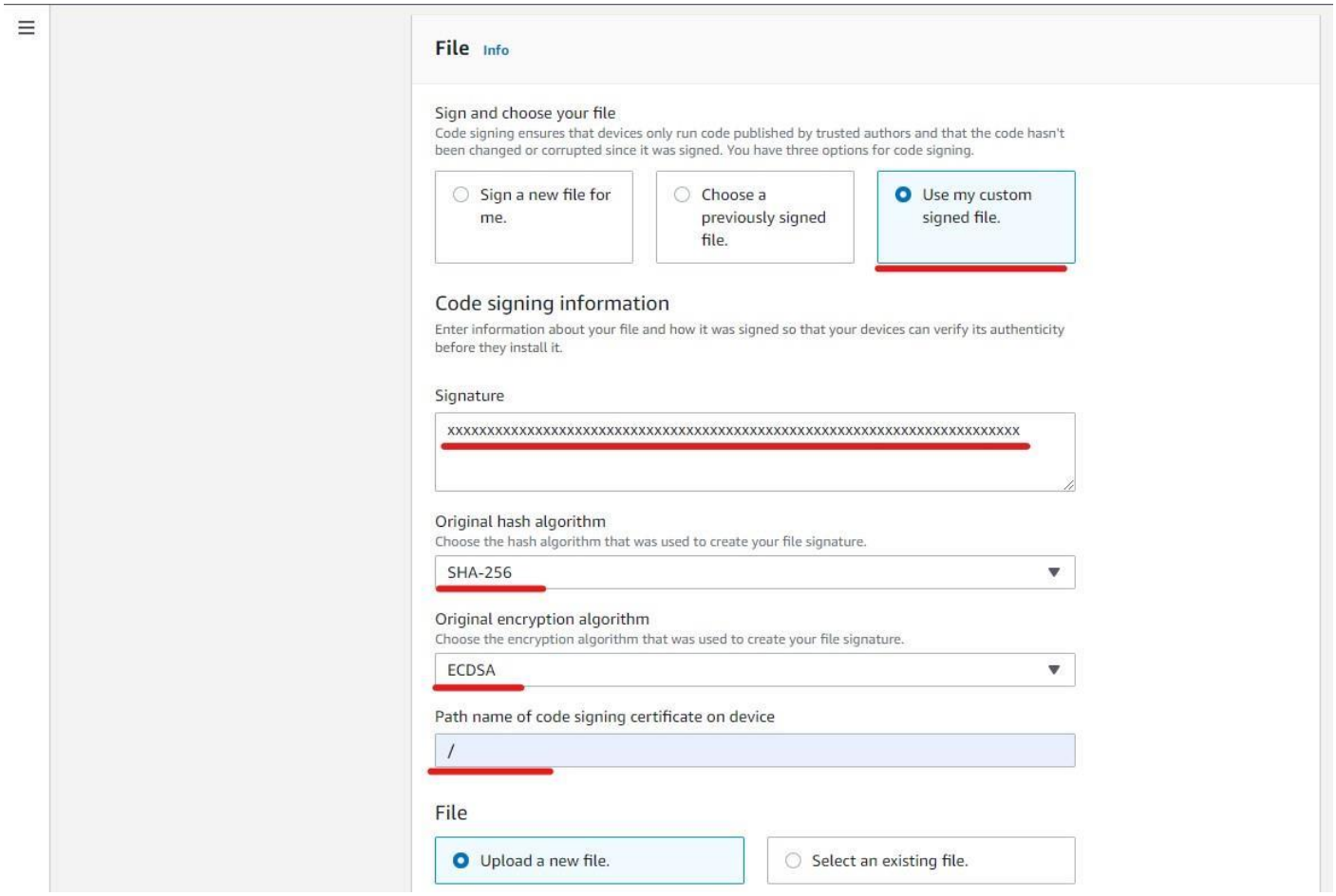

Step 2. For Job properties, give a unique name to your OTA job, then click next.



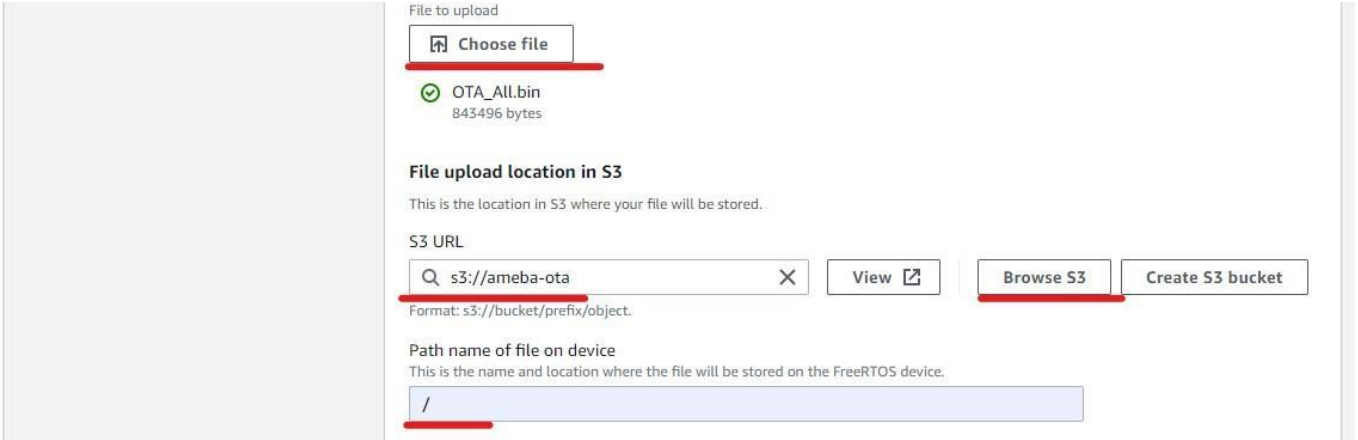

Step 3. In the following page, choose your device to update and select the protocol for file transfer



Step 4. In the following page, choose the option '**Use my custom signed file**'.
In the signature field enter content of '**IDT-OTA-Signature**' which generated by **signer_gcc.sh**.
Choose hash algorithm as '**SHA-256**'. Choose encryption algorithm as '**ECDSA**'
In "pathname of code signing certificate", enter '/'

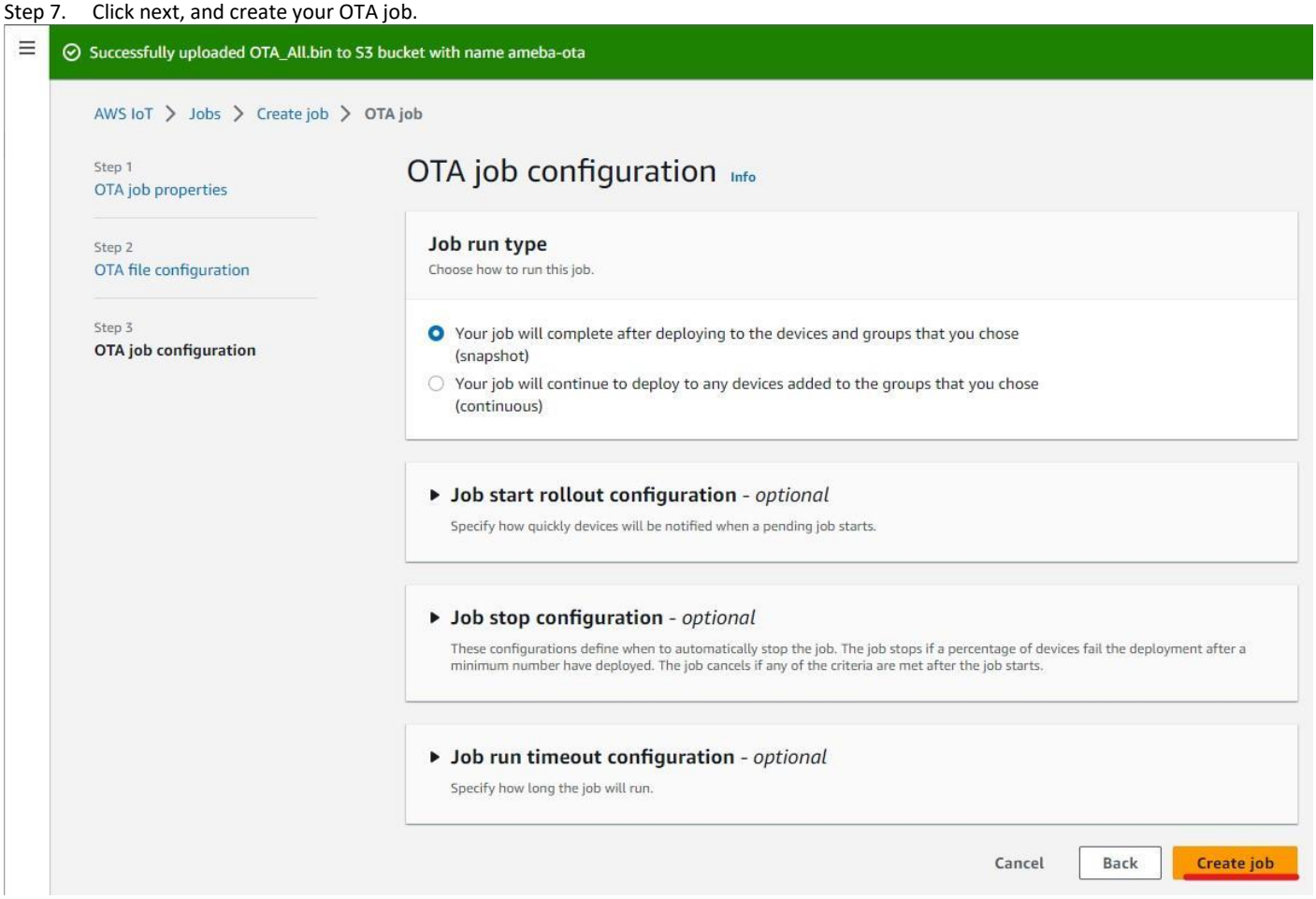


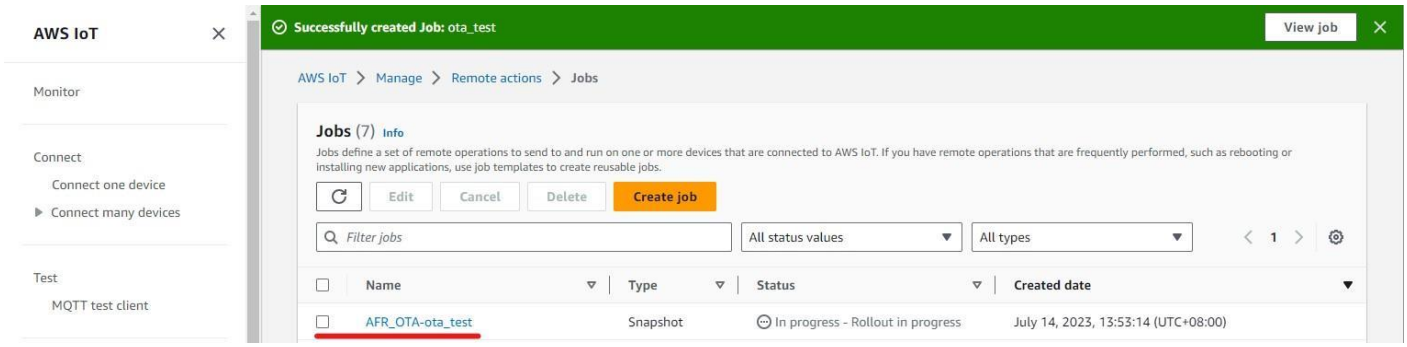

Step 5. Choose your custom signed firmware binary that was generated by the python script from S3 bucket.
In "Pathname of file on device", enter '/'



Step 6. Choose the IAM role for OTA update job. (This is the same IAM role as any OTA update job)


Step 7. Click next, and create your OTA job.



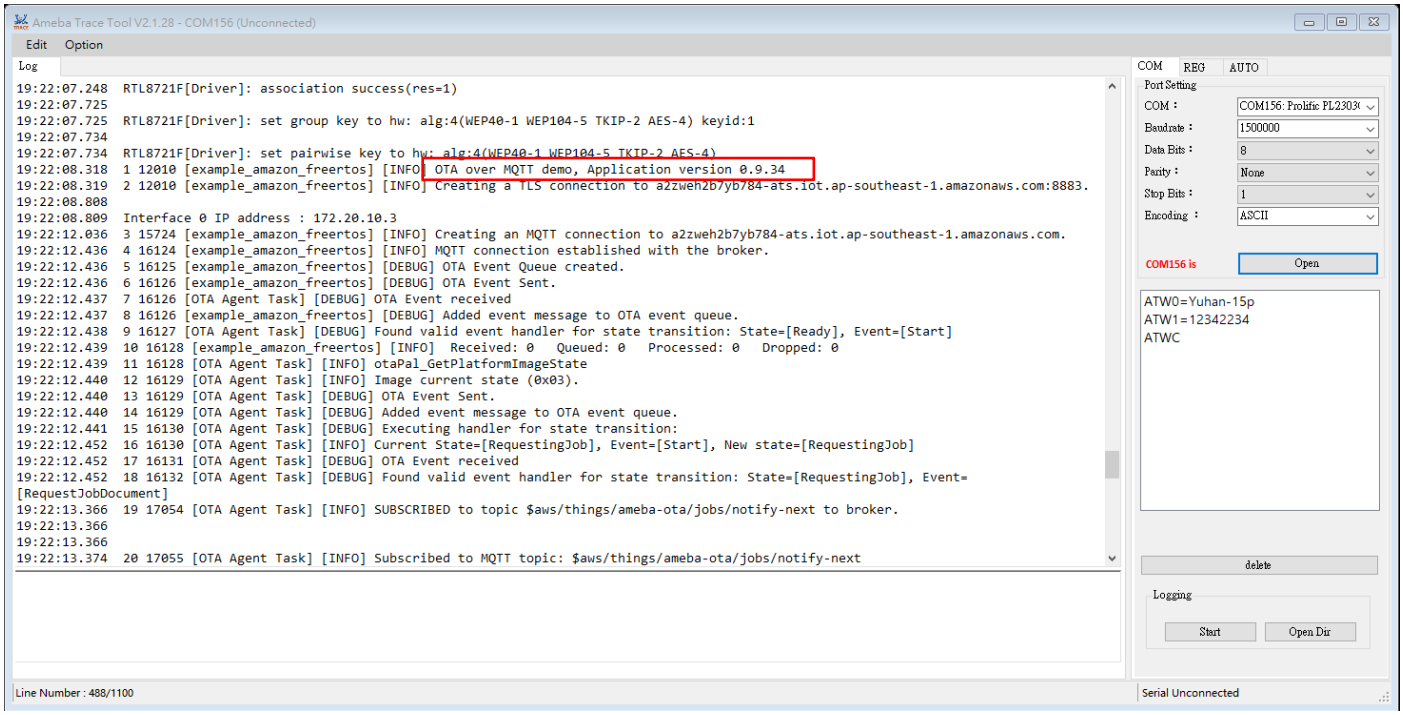



## 4.5 Run OTA Demo

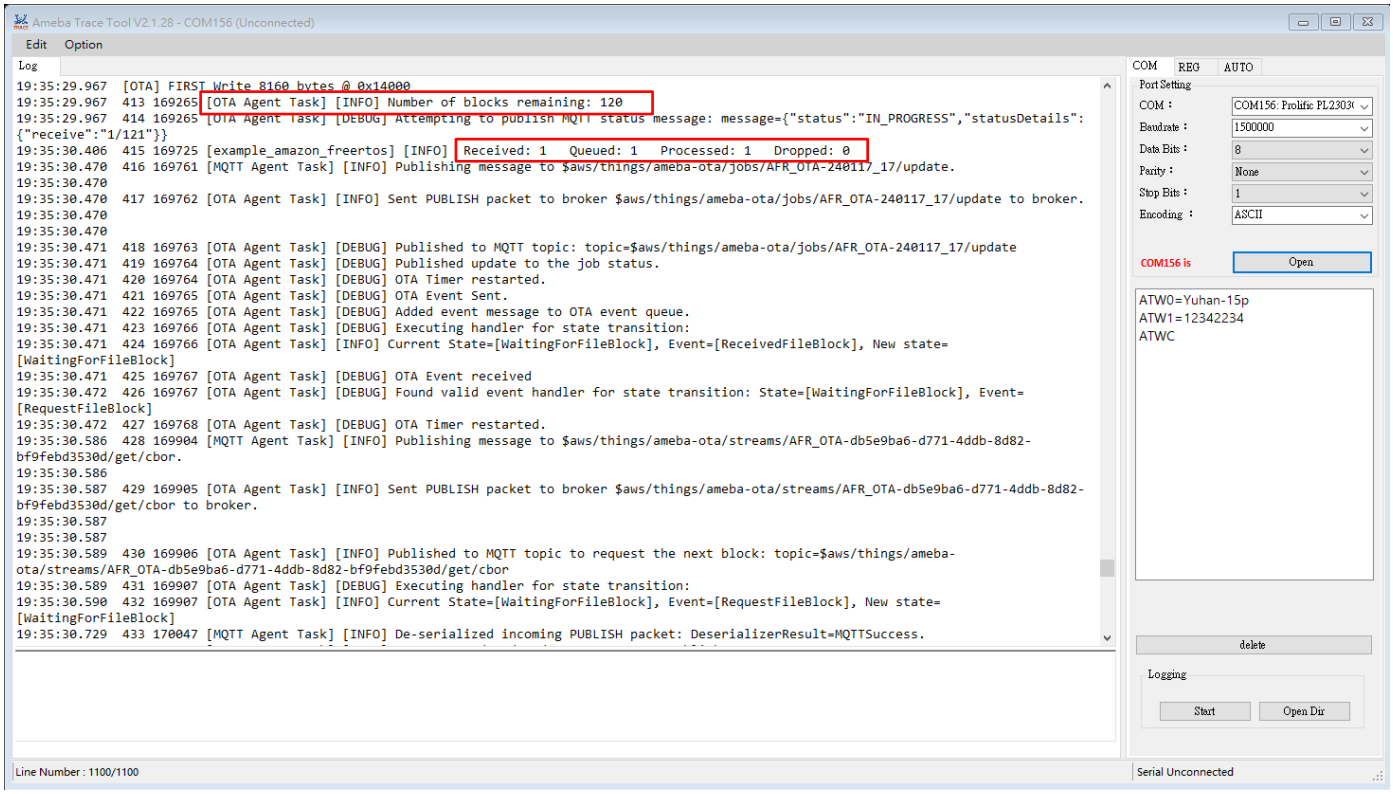
Now we can see that the status of OTA job on AWS IoT Core is “in progress”. It means that it is waiting AmebaDplus to request the update.

Next, download the image file with older version number to AmebaDplus and then reboot the device, the application will automatically start to run the OTA demo.

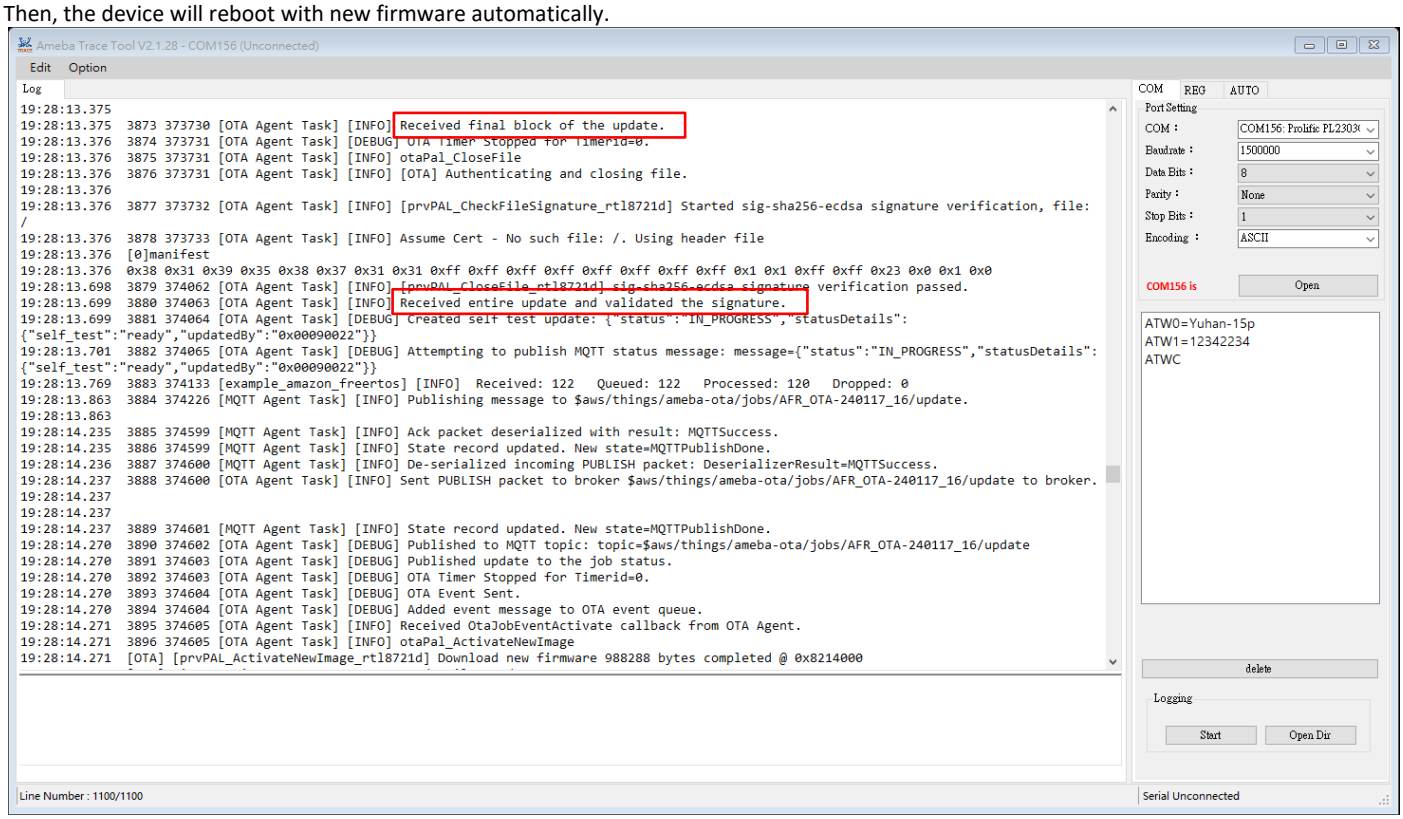
In the beginning, we can check the app version (0.9.34) of this running firmware, and the OTA process by the job ID:

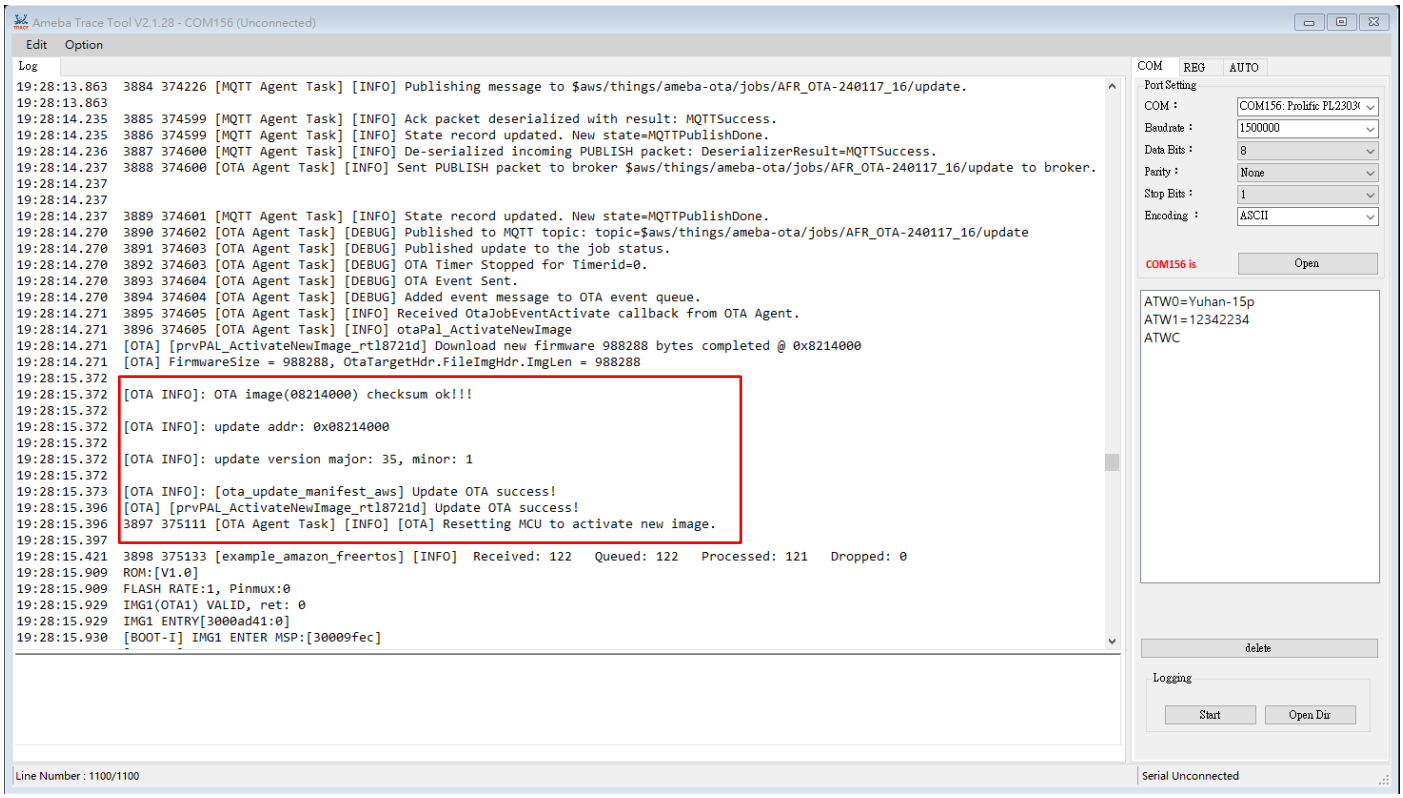



We can see that the OTA process start!

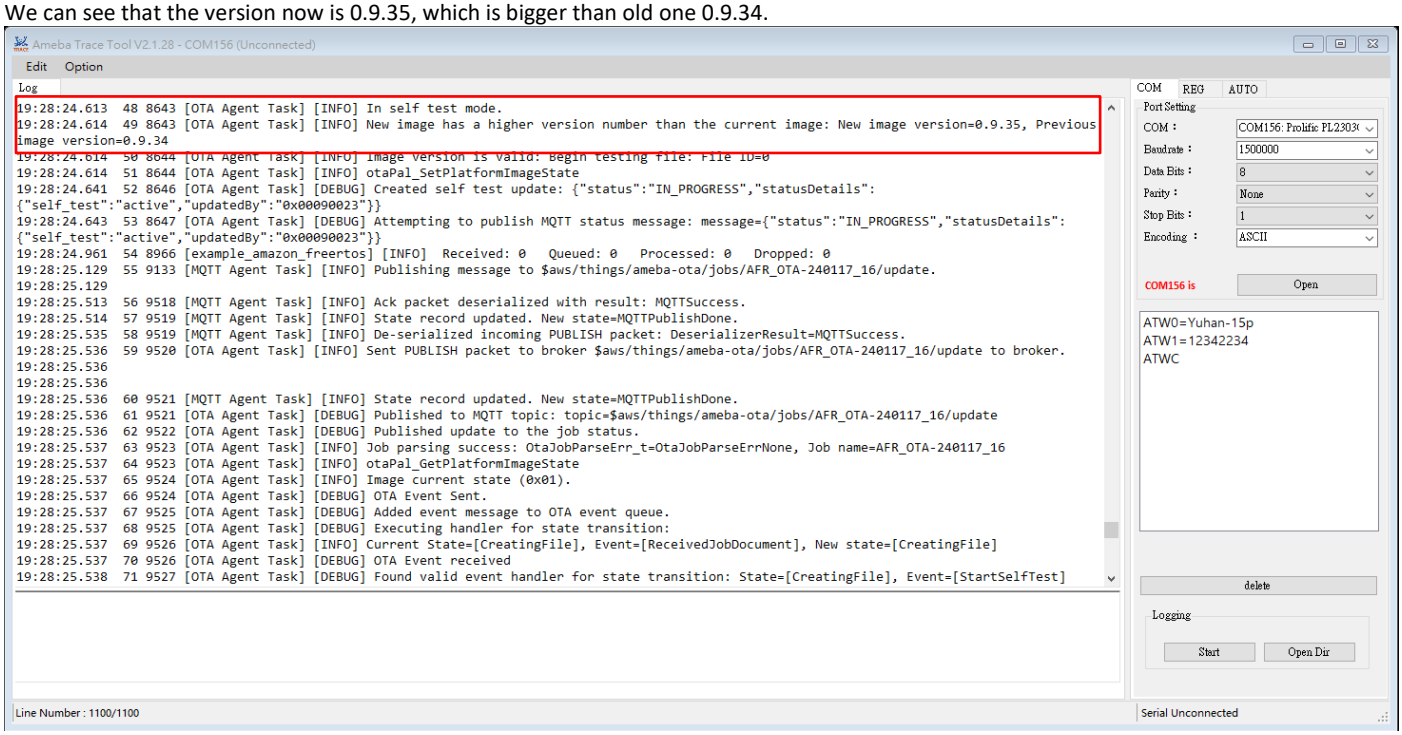


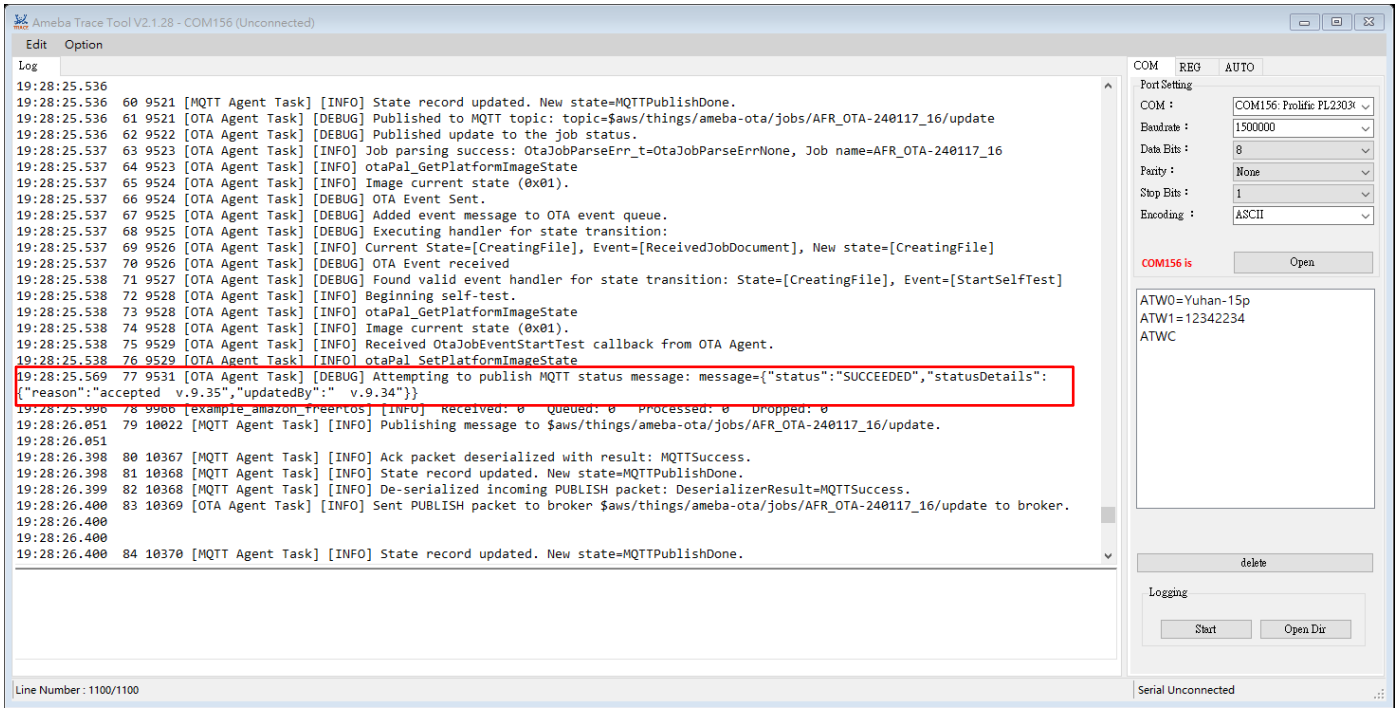

After receiving the final block, the signature will be checked if valid or not. If signature is valid, the OTA process is successful! Then, the device will reboot with new firmware automatically.






After booting with newer image, the device will start a self-test mode to check the app version is newer than before.
We can see that the version now is 0.9.35, which is bigger than old one 0.9.34.






In the final, the log imply that the OTA status is changed to “SUCCEEDED” !

# 5 Troubleshooting

## 5.1 ERROR: Invalid Key

Please check **WIFI_SSID** and **WIFI_PASSWORD** in in component/application/amazon/amazon-freertos-202012.00/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 ...
```

## 5.2 Failed to establish new MQTT connection

Please check **clientcredentialMQTT_BROKER_ENDPOINT** in component/application/amazon/amazon-freertos-202012.00/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-------
```

## 5.3 TLS_Connect fail

Please check **keyCLIENT_CERTIFICATE_PEM** and **keyCLIENT_PRIVATE_KEY_PEM** in component/application/amazon/amazon-freertos- 202012.00/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-------
```

# 6 Revision

| | |
|---|---|
| 1.0 | Initial version |
| 1.1 | Update picture |
| 1.1.1 | Fix typo |
| 2.1 | Update for FreeRTOS-LTS-202210.xx |