Getting started with the STMicroelectronics STM32L4+ Discovery Kit / BG96 IoT Node

Document Information

This tutorial provides instructions for getting started with the STMicroelectronics STM32L4+ Discovery Kit / BG96 IoT Node. Instructions are only for the Windows platform (Linux and MacOS are not supported).

- Set up your development environment, installing software on the host machine for developing and debugging embedded applications for your microcontroller board.
- Cross compiling a FreeRTOS demo application to a binary image.
- Loading the application binary image to your board, and then running the application.

Revision History (Version, Date, Description of change)

1.0	23-Nov-2020	Initial version

Overview

This document will provide you instructions to run FreeRTOS with Cellular connection on the STM32L4S5 MCU

Hardware Description

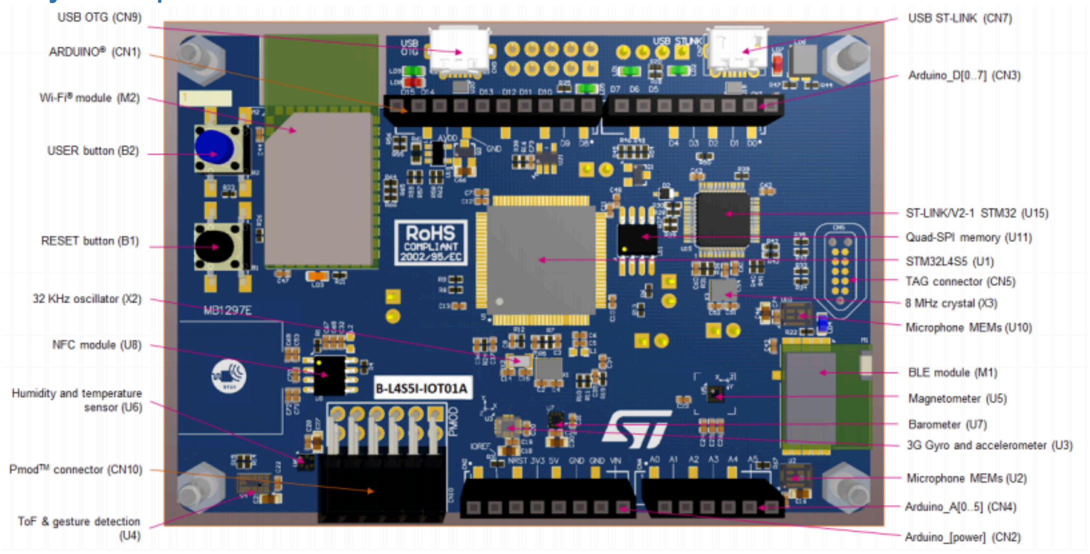
DataSheet

The package you are currently working with has been developed on STM32L4S5 family. The STM32L4S5x devices are an ultra-low-power microcontroller family (STM32L4+ Series) based on the high-performance Arm® Cortex®-M4 32-bit RISC core. They operate at a frequency of up to 120 MHz.
Datasheet of the device can be found under the following link on www.st.com :
https://www.st.com/resource/en/datasheet/stm32l4s5vi.pdf

Schematic

For PCB schematics, see MB1297-L4S5VI-Schematic-E02 Board Schematic on www.st.com

Key Components



For the key components description, jumper settings, Arduino connectors, power requirements, battery and LEDs descriptions, refer to the User Manual in **UM2708** Discovery kit for IoT node, multi-channel communication with STM32L4+ Series on www.st.com

Hardware requirements to run FreeRTOS demo

User will need to purchase the following 3 elements and to assemble them in order to reproduce the demo:

1. B-L4S5I-IOT01A Discovery Kit

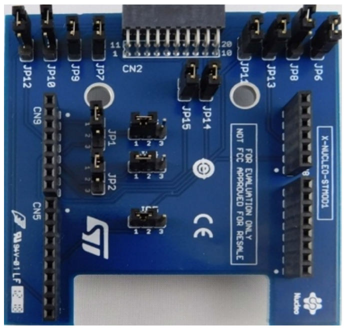
With the B-L4S5I-IOT01A Discovery kit for IoT node, users develop applications with direct connection to cloud servers. The Discovery kit enables a wide diversity of applications by exploiting low-power communication, multiway sensing and Arm® Cortex®-M4 core-based STM32L4+ Series features. The support for ARDUINO® Uno V3 and Pmod™ connectivity provides unlimited expansion capabilities with a large choice of specialized add-on boards.

More information can be found under the following link:
https://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-mpu-eval-tools/stm32-mcu-mpu-eval-tools/stm32-discovery-kits/b-l4s5i-iot01a.html

Likewise documentation like schematic or Bill of Material can be found under the following link:
https://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-mpu-eval-tools/stm32-mcu-mpu-eval-tools/stm32-discovery-kits/b-l4s5i-iot01a.html#documentation

Please obtain a USB cable (micro-USB to USB-A) in order to communicate and power the board (from a laptop).

2. X-NUCLEO-STMODA1 STMod+ connector expansion board for STM32 Nucleo



The X-NUCLEO-STMODA1 provides an easy way to expand your STM32 Nucleo board with the STMod+ connector, which allows interaction with the new set of STM32 Nucleo development boards using this connector. It provides an easy way to evaluate the STMod+ board solution together with other STM32 Nucleo boards.

More information can be found under the following link:
https://www.st.com/content/st_com/en/products/ecosystems/stm32-open-development-environment/stm32-nucleo-expansion-boards/stm32-ode-translate-hw/x-nucleo-stmoda1.html

Likewise documentation like schematic or Bill of Material can be found under the following link:
https://www.st.com/content/st_com/en/products/ecosystems/stm32-open-development-environment/stm32-nucleo-expansion-boards/stm32-ode-translate-hw/x-nucleo-stmoda1.html#documentation

No additional cable is required.

3. STEVAL-STMODLTE LTE connectivity expansion board for STMod+ connector compatible evaluation boards

The STEVAL-STMODLTE expansion board adds LTE connectivity to evaluation boards hosting the STMod+ expansion connector. It embeds an LTE cellular modem and an eSIM with a worldwide coverage.

More information can be found under the following link:
https://www.st.com/content/st_com/en/products/evaluation-tools/solution-evaluation-tools/communication-and-connectivity-solution-eval-boards/steval-stmodlte.html

Likewise documentation like schematic or Bill of Material can be found under the following link:
https://www.st.com/content/st_com/en/products/evaluation-tools/solution-evaluation-tools/communication-and-connectivity-solution-eval-boards/steval-stmodlte.html#documentation

A SIM card will be required in order to enable communication onto the cellular network. Please make sure that the network supported by your operator is the one supported by the modem embedded on the board.

In most cases, you have a compatible SIM card and an active data plan. If you do not have a compatible SIM card, you can obtain SIMs from the AWS marketplace.

3rd Party purchasable items

If you do not already have the STMicroelectronics boards mentioned above please visit the following to get access to our distributors:

1. B-L4S5I-IOT01A
2. X-NUCLEO-STMODA1
3. STEVAL-STMODLTE

Set up your development environment

1. Install CMAKE
   - Setup the GNU Arm Embedded Toolchain.
     1. Download the toolchain from the Arm Embedded Toolchain download page for **Windows**.
     2. Install CMake configure tool. You must have CMake version 3.13 or higher installed. You can download the binary distribution of CMake from CMake.org.

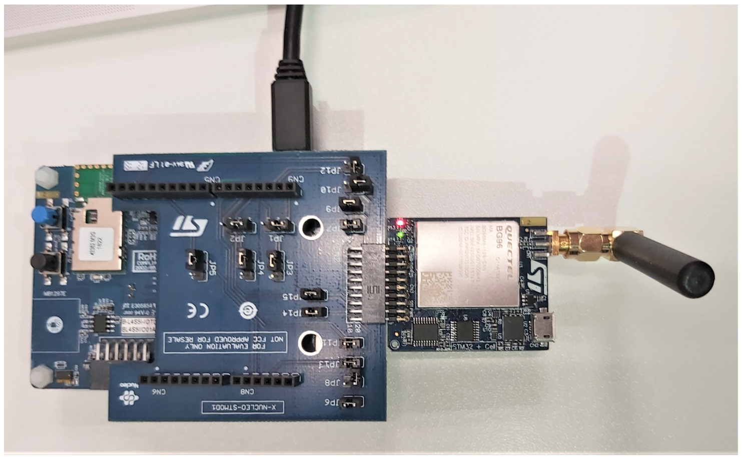
2. Install STM32CubeProgrammer firmware download tool.

Additional Software References

For more information on STM32CubeProgrammer, see the UM2237 STM32CubeProgrammer software description.

For more information about using CMake with other operating systems and options, see Using CMake with FreeRTOS.

Set up your hardware



To setup the B-L4S5I-IOT01A + X-NUCLEO-STMODA1 + STEVAL-STMODLTE:

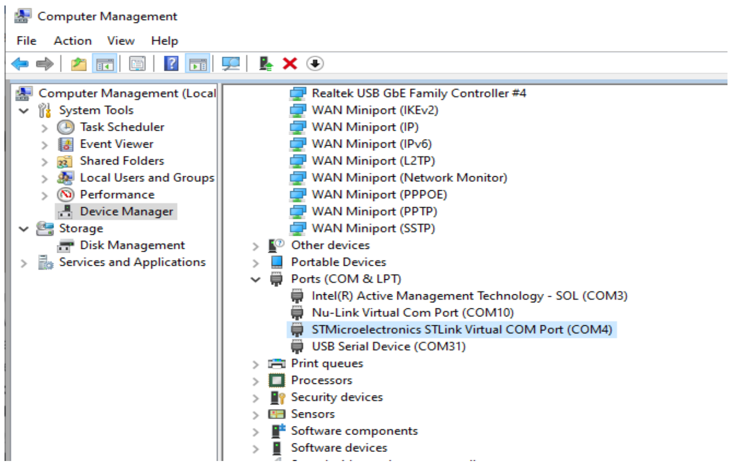
1. Plug the X-NUCLEO-STMODA1 onto the B-L4S5I-IOT01A through the Arduino connector.
2. Plug the STEVAL-STMODLTE add-on board onto the X-NUCLEO-STMODA1 through the STmod+ connector
3. Connect your laptop to the Micro USB port on your B-L4S5I-IOT01A. The On-board USB debug probe is used to program the board and provides Virtual COM Port support.

Prerequisites

- Make sure you have installed the latest BG96 firmware. You can download the latest BG96 firmware from here. (You will need to login or sign on with a Quectel account.)

Establishing a serial connection

1. Check the list of identified COM ports in the Windows Device Manager.



```
Computer Management (Local
  System Tools
    Task Scheduler
    Event Viewer
    Shared Folders
    Local Users and Groups
    Performance
    Device Manager
  Storage
    Disk Management
  Services and Applications
```

```
    Realtek USB GbE Family Controller #4
    WAN Miniport (IKEv2)
    WAN Miniport (IP)
    WAN Miniport (IPv6)
    WAN Miniport (L2TP)
    WAN Miniport (Network Monitor)
    WAN Miniport (PPPOE)
    WAN Miniport (PPTP)
    WAN Miniport (SSTP)
Other devices
Portable Devices
Ports (COM & LPT)
    Intel(R) Active Management Technology - SOL (COM3)
    Nu-Link Virtual Com Port (COM10)
    STMicroelectronics STLink Virtual COM Port (COM4)
    USB Serial Device (COM31)
Print queues
Processors
Security devices
Sensors
Software components
Software devices
```

2. Start a serial terminal and open a connection with the following settings:

- Baud rate: 115200
- Data: 8 bit
- Parity: None
- Stop bits: 1
- Flow control: None

You will need to install a terminal program for connecting computers to the device.

Setup your AWS account and Permissions

To create an AWS account, see Create and Activate an AWS Account.

To create an IAM user, follow the instructions at How to create an IAM user. For a deeper understanding of IAM, refer to the IAM User Guide.

- AmazonFreeRTOSFullAccess
- AWSIoTFullAccess

To attach the AmazonFreeRTOSFullAccess policy to your IAM user

1. Browse to the IAM console, and from the navigation pane, choose ** Users**.
2. Enter your user name in the search text box, and then choose it from the list.
3. Choose **Add permissions**.
4. Choose **Attach existing policies directly**.
5. In the search box, enter **AmazonFreeRTOSFullAccess**, choose it from the list, and then choose **Next: Review**.
6. Choose **Add permissions**.

To attach the AWSIoTFullAccess policy to your IAM user

1. Browse to the IAM console, and from the navigation pane, choose ** Users**.
2. Enter your user name in the search text box, and then choose it from the list.
3. Choose **Add permissions**.
4. Choose **Attach existing policies directly**.
5. In the search box, enter **AWSIoTFullAccess**, choose it from the list, and then choose **Next: Review**.
6. Choose **Add permissions**.

For more information about IAM and user accounts, see IAM User Guide.

For more information about policies, see IAM Permissions and Policies.

Download and Build the FreeRTOS demo project

You can download a release from GitHub with the following command:

To clone using HTTPS:

```
git clone --branch 202007.00.stm32l4s5 https://github.com/iotlabtpe/amazon-freertos-stm32l4plus.git --recurse-submodules
```

Using SSH:

```
git clone --branch 202007.00.stm32l4s5 git@github.com:iotlabtpe/amazon-freertos-stm32l4plus.git --recurse-submodules
```

If you have downloaded the repo without using the --recurse-submodules argument, you need to run:

```
git submodule update --init --recursive
```

To checkout the branch to 202007.00.stm32l4s5, you need to run:

```
git checkout 202007.00.stm32l4s5
```

Use CMake to generate the build files, and then use Make to build the application.

To generate the demo application's build files with CMake

1. Change directories to the root of your FreeRTOS download directory.
2. Use the following command to generate the build files:

   ```
   cmake -DVENDOR=st -DBOARD=stm32l4plus_discovery -DCOMPILER=arm-gcc -S $freertos -B $build-folder -GNinja
   ```

   If `arm-none-eabi-gcc` is not in your shell path, you also need to set the `AFR_TOOLCHAIN_PATH` CMake variable. For example:

   ```
   -D AFR_TOOLCHAIN_PATH=/home/user/opt/gcc-arm-none-eabi/bin
   ```

   For more information about using CMake with FreeRTOS, see Using CMake with FreeRTOS.

Provision the STM32L4+ with AWS IoT

Your board must be registered with AWS IoT to communicate with the AWS Cloud. To register your board with AWS IoT, you need the following:

An AWS IoT policy

The AWS IoT policy grants your device permissions to access AWS IoT resources. It is stored on the AWS Cloud.

An AWS IoT thing

An AWS IoT thing allows you to manage your devices in AWS IoT. It is stored on the AWS Cloud.

A private key and X.509 certificate

The private key and certificate allow your device to authenticate with AWS IoT.

To create an AWS IoT policy

1. To create an IAM policy, you need to know your AWS Region and AWS account number.

   To find your AWS account number, open the AWS Management Console, locate and expand the menu beneath your account name in the upper-right corner, and choose **My Account**. Your account ID is displayed under **Account Settings**.

   To find the AWS region for your AWS account, use the AWS Command Line Interface. To install the AWS CLI, follow the instructions in the AWS Command Line Interface User Guide. After you install the AWS CLI, open a command prompt window and enter the following command:

```
aws iot describe-endpoint
```

   The output should look like this:

```
{
   "endpointAddress": "xxxxxxxxxxxxxx.iot.us-west-2.amazonaws.com"
}
```

   In this example, the region is us-west-2.

2. Browse to the AWS IoT console.

3. In the navigation pane, choose **Secure**, choose **Policies**, and then choose **Create**.

4. Enter a name to identify your policy.

5. In the **Add statements** section, choose **Advanced mode**. Copy and paste the following JSON into the policy editor window. Replace *aws-region* and *aws-account* with your AWS Region and account ID.

```
{
   "Version": "2012-10-17",
   "Statement": [
   {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource":"arn:aws:iot:aws-region:aws-account-id:*"
   },
   {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "arn:aws:iot:aws-region:aws-account-id:*"
   },
   {
       "Effect": "Allow",
       "Action": "iot:Subscribe",
       "Resource": "arn:aws:iot:aws-region:aws-account-id:*"
   },
   {
       "Effect": "Allow",
       "Action": "iot:Receive",
       "Resource": "arn:aws:iot:aws-region:aws-account-id:*"
   }
   ]
}
```

This policy grants the following permissions:

`iot:Connect`
Grants your device the permission to connect to the AWS IoT message broker with any client ID.

`iot:Publish`
Grants your device the permission to publish an MQTT message on any MQTT topic.

`iot:Subscribe`
Grants your device the permission to subscribe to any MQTT topic filter.

`iot:Receive`
Grants your device the permission to receive messages from the AWS IoT message broker on any MQTT topic.

NOTE – 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

6. Choose **Create**.

To create an IoT thing

1. Browse to the AWS IoT console.
2. In the navigation pane, choose **Manage**, and then choose **Things**.
3. If you do not have any IoT things registered in your account, the **You don't have any things yet** page is displayed. If you see this page, choose **Register a thing**. Otherwise, choose **Create**.
4. On the **Creating AWS IoT things** page, choose **Create a single thing**.
5. On the **Add your device to the thing registry** page, enter a name for your thing, and then choose **Next**. The thing name should be unique within your account.
6. On the **Add a certificate for your thing** page, under **One-click certificate creation**, choose **Create certificate**.
7. Download your private key and certificate by choosing the **Download** links for each.
8. Choose **Activate** to activate your certificate. Certificates must be activated prior to use.
9. Choose **Attach a policy** to attach a policy to your certificate that grants your device access to AWS IoT operations.
10. Choose the policy you just created and choose **Register thing**.

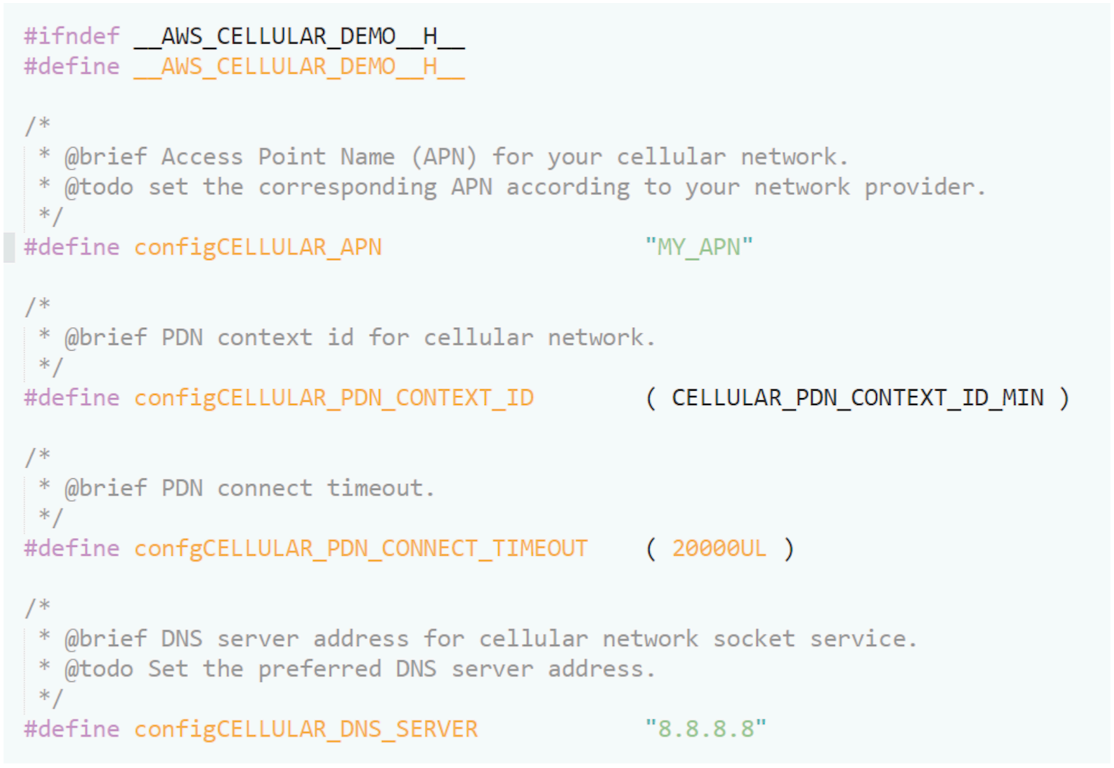
After your board is registered with AWS IoT, you can continue to configure FreeRTOS to send those credentials during the TLS connection.

FreeRTOS is a C language project, and the certificate and private key must be specially formatted to be added to the project.

Include the certificates in your FreeRTOS project

1. In a browser window, open <your FreeRTOS directory>/tools/certificate_configuration/CertificateConfigurator.html.

2. Under **Certificate PEM file**, choose the ID-certificate.pem.crt that you downloaded from the AWS IoT console.
3. Under **Private Key PEM file**, choose the ID-private.pem.key that you downloaded from the AWS IoT console.
4. Choose **Generate and save aws_clientcredential_keys.h**, and then save the file in <your FreeRTOS directory>/demos/include. This overwrites the existing file in the directory.
5. An APN or Access Point Name is the combination of values that indicates to your device how to connect to the mobile internet servers of your network provider. You can find out your specific APN settings on your network operator website. configCELLULAR_APN in aws_demos_cellular.h with the Cellular network provider.
6. PDN context identifier specifies a particular packet data protocol context definition. The range of permitted values is 1 to 16. The default value is minimum value 1. configCELLULAR_PDN_CONTEXT_ID in aws_demos_cellular.h with the PDN context id you desire to use.
7. configCELLULAR_DNS_SERVER in aws_demos_cellular.h with the DNS server that will be used.



```
#ifndef __AWS_CELLULAR_DEMO__H__
#define __AWS_CELLULAR_DEMO__H__

/*
 * @brief Access Point Name (APN) for your cellular network.
 * @todo set the corresponding APN according to your network provider.
 */
#define configCELLULAR_APN                   "MY_APN"

/*
 * @brief PDN context id for cellular network.
 */
#define configCELLULAR_PDN_CONTEXT_ID        ( CELLULAR_PDN_CONTEXT_ID_MIN )

/*
 * @brief PDN connect timeout.
 */
#define confgCELLULAR_PDN_CONNECT_TIMEOUT    ( 20000UL )

/*
 * @brief DNS server address for cellular network socket service.
 * @todo Set the preferred DNS server address.
 */
#define configCELLULAR_DNS_SERVER            "8.8.8.8"
```

Note
The certificate and private key are hard-coded for demonstration purposes only. Production-level applications should store these files in a secure location.

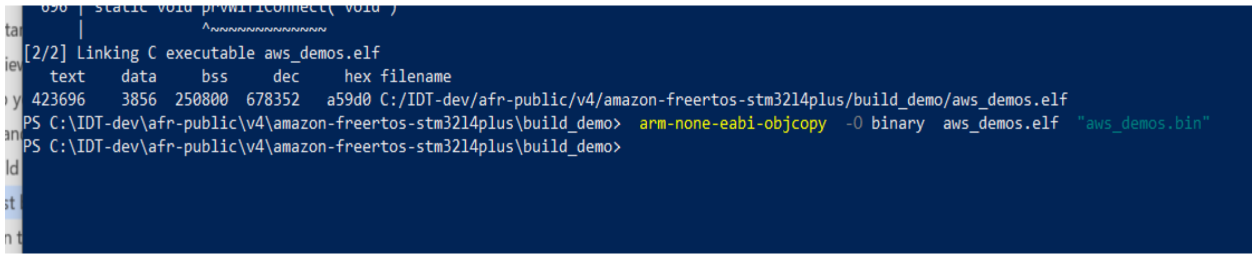
To build the application

1. Change directories to the build directory.
2. Invoke Ninja to build the application:

   ```
   ninja
   ```

   Or, use the generic CMake interface to build the application:

   ```
   cmake --build build-directory
   ```



```
 696 | static void prvWifiConnect( void )
     |             ^~~~~~~~~~~~~~
[2/2] Linking C executable aws_demos.elf
   text    data     bss     dec     hex filename
 423696    3856  250800  678352   a59d0 C:/IDT-dev/afr-public/v4/amazon-freertos-stm32l4plus/build_demo/aws_demos.elf
PS C:\IDT-dev\afr-public\v4\amazon-freertos-stm32l4plus\build_demo> arm-none-eabi-objcopy -O binary aws_demos.elf "aws_demos.bin"
PS C:\IDT-dev\afr-public\v4\amazon-freertos-stm32l4plus\build_demo>
```

If the **aws_demos.elf** successfully generated, you can go to next section “Post build scripts to sign the firmware”.

Post build scripts to sign the firmware.

STM32L4+ has secure boot, therefore we need to sign the image.

1. Navigate to the build directory, and generate **aws_demos.bin** file with following command:

   ```
   arm-none-eabi-objcopy  -O binary  aws_demos.elf  "aws_demos.bin"
   ```

2. Get STM32L4+ post scripts from x-cube-aws.html.
   a. After accepting the license agreement, sign and download X-CUBE-AWS software, unzip it into your target working directory, for example: “**C:\STM32CubeExpansion_Cloud_AWS_V2**”
3. Copy the **aws_demos.elf** and **aws_demos.bin** files to the working directory as below:

   Copy elf file and rename it to **B-L4S5I-IOT01_aws_demos.elf.**

   ```
   cp c:/<your_afr_dir>/amazon-freertos-stm32l4plus/build/aws_demos.elf
   c:/STM32CubeExpansion_Cloud_AWS_v2.0.0/Projects/B-L4S5I-
   IOT01A/Applications/BootLoader_STSAFE/2_Images_SECoreBin/STM32CubeIDE/B-
   L4S5I-IOT01_aws_demos.elf
   ```

   Copy bin file and rename it to **B-L4S5I-IOT01_aws_demos.bin**.

   ```
   cp c:/<your_afr_dir>/amazon-freertos-stm32l4plus/build/aws_demos.bin
   c:/STM32CubeExpansion_Cloud_AWS_v2.0.0/Projects/B-L4S5I-
   ```

```
IOT01A/Applications/BootLoader_STSAFE/2_Images_SECoreBin/STM32CubeIDE/B-
L4S5I-IOT01_aws_demos.bin
```

4. Run the post build script to sign the firmware:
   a. Navigate to your postbuild.sh script folder, for example with the above example working directory it would look like this:
   c:/STM32CubeExpansion_Cloud_AWS_v2.0.0/Projects/B-L4S5I-IOT01A/Applications/Cloud/aws_demos/STM32CubeIDE/Debug

   b. Run the **postbuild.sh** script, it will generate **SBSFU_B-L4S5I-IOT01_aws_demos.bin** to be flashed.

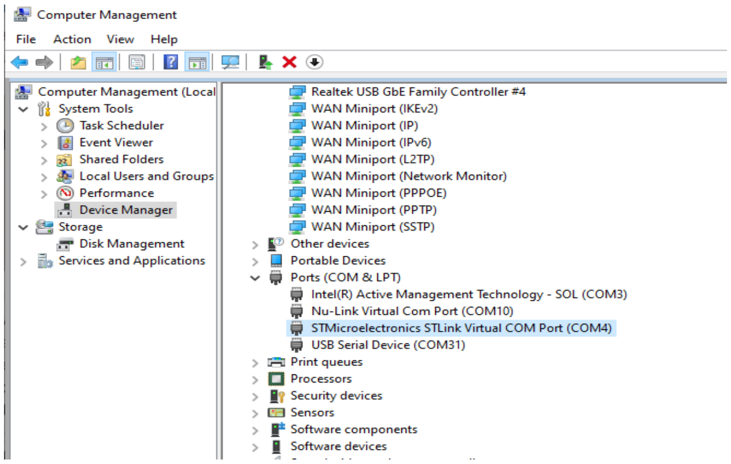
```
sh c:/STM32CubeExpansion_Cloud_AWS_v2.0.0/Projects/B-L4S5I-
IOT01A/Applications/BootLoader_STSAFE/2_Images_SECoreBin/STM32CubeIDE/
postbuild.sh .. c:/STM32CubeExpansion_Cloud_AWS_v2.0.0/Projects/B-L4S5I-
IOT01A/Applications/BootLoader_STSAFE/2_Images_SECoreBin/STM32CubeIDE/
B-L4S5I-IOT01_aws_tests.elf
c:/STM32CubeExpansion_Cloud_AWS_v2.0.0/Projects/B-L4S5I-
IOT01A/Applications/BootLoader_STSAFE/2_Images_SECoreBin/STM32CubeIDE/
B-L4S5I-IOT01_aws_tests.bin 1 bigelf
```

Run the FreeRTOS demo project

- Use a USB cable to connect your STMicroelectronics STM32L4+ Discovery Kit IoT Node to your computer.
- Download the **STM32CubeProgrammer** from here.
- Navigate to the signed firmware directory. In the example working directory:
  C: /amazon-freertos-stm32l4plus/vendors/st/stm32l4plus_discovery/PostBuildScripts
- Flash the firmware to STM32L4+ with following command

```
STM32_Programmer_CLI.exe -c port=SWD -d SBSFU_B-L4S5I-IOT01_aws_demos.bin
0x08000000 -v
```

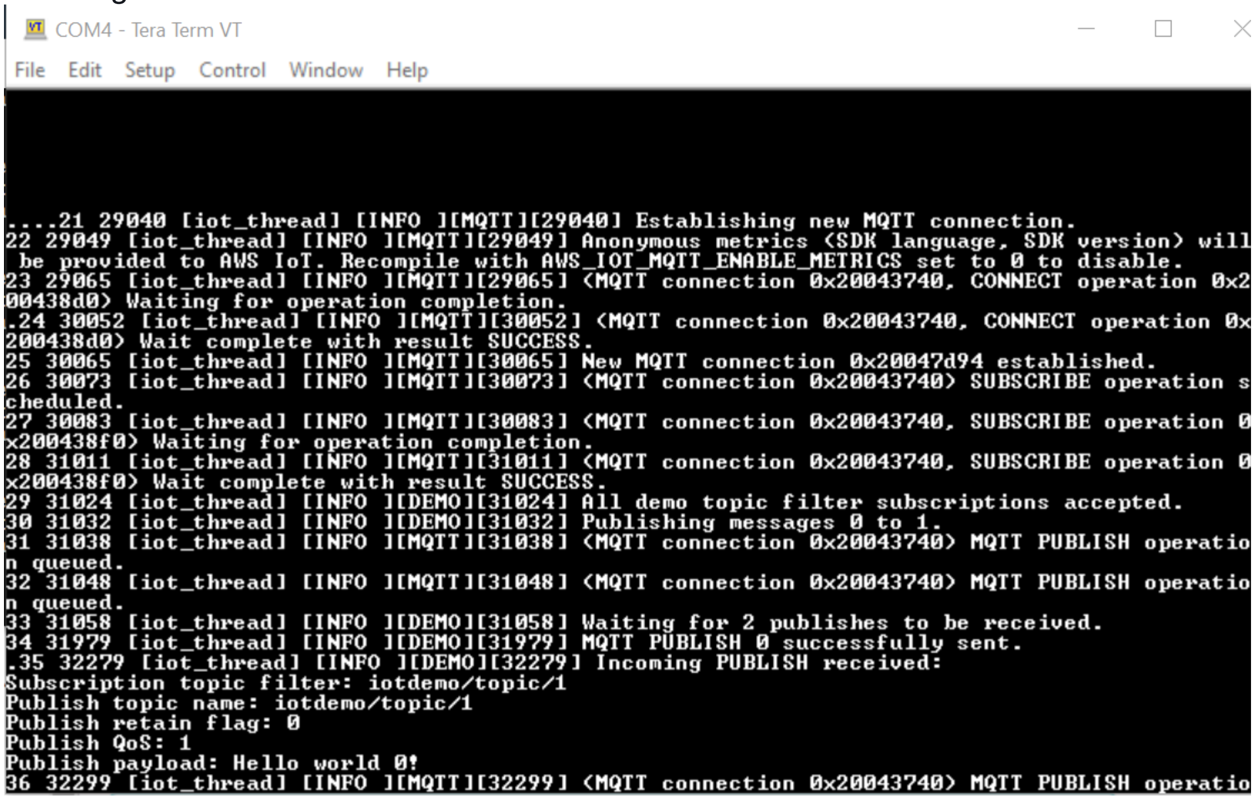
- Check the list of identified COM ports in the Windows Device Manager.



```
Computer Management (Local
  System Tools
    Task Scheduler
    Event Viewer
    Shared Folders
    Local Users and Groups
    Performance
    Device Manager
  Storage
    Disk Management
  Services and Applications

    Realtek USB GbE Family Controller #4
    WAN Miniport (IKEv2)
    WAN Miniport (IP)
    WAN Miniport (IPv6)
    WAN Miniport (L2TP)
    WAN Miniport (Network Monitor)
    WAN Miniport (PPPOE)
    WAN Miniport (PPTP)
    WAN Miniport (SSTP)
  Other devices
  Portable Devices
  Ports (COM & LPT)
    Intel(R) Active Management Technology - SOL (COM3)
    Nu-Link Virtual Com Port (COM10)
    STMicroelectronics STLink Virtual COM Port (COM4)
    USB Serial Device (COM31)
  Print queues
  Processors
  Security devices
  Sensors
  Software components
  Software devices
```

A successful run of the FreeRTOS MQTT Demo should have an output similar to the following in the serial console.



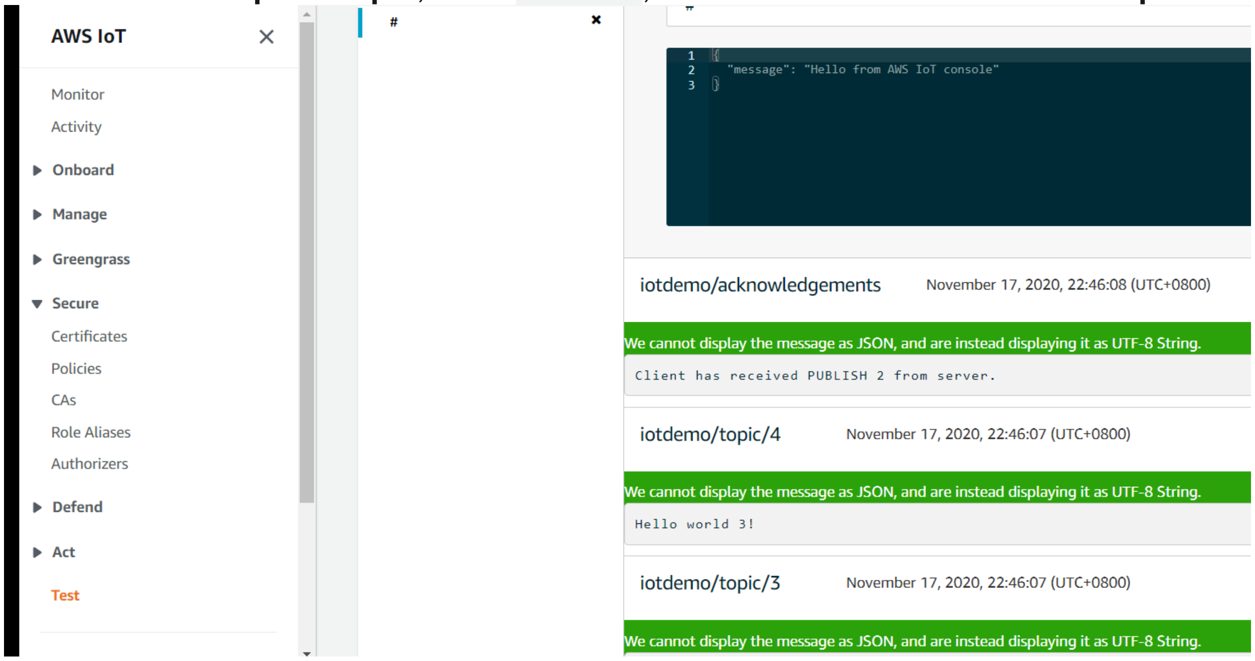
```
....21 29040 [iot_thread] [INFO ][MQTT][29040] Establishing new MQTT connection.
22 29049 [iot_thread] [INFO ][MQTT][29049] Anonymous metrics (SDK language, SDK version) will
 be provided to AWS IoT. Recompile with AWS_IOT_MQTT_ENABLE_METRICS set to 0 to disable.
23 29065 [iot_thread] [INFO ][MQTT][29065] (MQTT connection 0x20043740, CONNECT operation 0x2
00438d0) Waiting for operation completion.
.24 30052 [iot_thread] [INFO ][MQTT][30052] (MQTT connection 0x20043740, CONNECT operation 0x
200438d0) Wait complete with result SUCCESS.
25 30065 [iot_thread] [INFO ][MQTT][30065] New MQTT connection 0x20047d94 established.
26 30073 [iot_thread] [INFO ][MQTT][30073] (MQTT connection 0x20043740) SUBSCRIBE operation s
cheduled.
27 30083 [iot_thread] [INFO ][MQTT][30083] (MQTT connection 0x20043740, SUBSCRIBE operation 0
x200438f0) Waiting for operation completion.
28 31011 [iot_thread] [INFO ][MQTT][31011] (MQTT connection 0x20043740, SUBSCRIBE operation 0
x200438f0) Wait complete with result SUCCESS.
29 31024 [iot_thread] [INFO ][DEMO][31024] All demo topic filter subscriptions accepted.
30 31032 [iot_thread] [INFO ][DEMO][31032] Publishing messages 0 to 1.
31 31038 [iot_thread] [INFO ][MQTT][31038] (MQTT connection 0x20043740) MQTT PUBLISH operatio
n queued.
32 31048 [iot_thread] [INFO ][MQTT][31048] (MQTT connection 0x20043740) MQTT PUBLISH operatio
n queued.
33 31058 [iot_thread] [INFO ][DEMO][31058] Waiting for 2 publishes to be received.
34 31979 [iot_thread] [INFO ][DEMO][31979] MQTT PUBLISH 0 successfully sent.
.35 32279 [iot_thread] [INFO ][DEMO][32279] 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!
36 32299 [iot_thread] [INFO ][MQTT][32299] (MQTT connection 0x20043740) MQTT PUBLISH operatio
```

Monitoring MQTT messages on the cloud

You can use the MQTT test client in the AWS IoT console to monitor the messages that your device sends to the AWS Cloud.

- Sign in to the AWS IoT console.

- In the navigation pane, choose **Test** to open the MQTT test client.
- In **Subscription topic**, enter iotdemo/#, and then choose **Subscribe to topic**.



```
{
  "message": "Hello from AWS IoT console"
}
```

iotdemo/acknowledgements November 17, 2020, 22:46:08 (UTC+0800)

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

```
Client has received PUBLISH 2 from server.
```

iotdemo/topic/4 November 17, 2020, 22:46:07 (UTC+0800)

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

```
Hello world 3!
```

iotdemo/topic/3 November 17, 2020, 22:46:07 (UTC+0800)

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

Troubleshooting

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