UM2186

User manual

Getting started with the FP-CLD-AWS1 software package for IoT node with Wi-Fi, cellular modem module and sensors, connected to Amazon AWS IoT cloud

Introduction

FP-CLD-AWS1 is an STM32Cube function pack that lets you safely connect your IoT node to an Amazon AWS IoT service so you can transmit sensor data and receive commands from AWS-based cloud applications.

The package integrates the AWS FreeRTOS IoT device SDK middleware with APIs to simplify interaction between STM32-based devices and the Amazon AWS IoT services.

A companion AWS-based web dashboard (DSH-ASSETRACKING) is available to quickly evaluate the firmware package functions for comprehensive sensor data visualization and device control.

This software integrates the X-CUBE-SBSFU and the STSAFE-A110 to implement a Secure Boot (Root of Trust services) solution and a Secure Firmware Update solution allowing you to safely update the AWS program from the AWS console. It fully supports security and protocol requirements to interface an IoT node with AWS cloud.

This software together with our suggested combinations of STM32 and ST devices can be used to develop complete sensor-to-cloud applications for a broad range of smart home, smart industry, robotics, predictive maintenance, appliances, building automation, healthcare, defense or navigation.

RELATED LINKS

Visit the STM32Cube ecosystem web page on www.st.com for further information

1 Acronyms and abbreviations

Table 1. List of acronyms

Acronym	Description
AP	Access point
BSP	Base support package
FOTA	Firmware update over-the-air
GPIO	General purpose input/output
HAL	Hardware abstraction layer
HTML	Hypertext markup language
HTTP	Hypertext transfer protocol
IDE	Integrated development environment
IoT	Internet of things
I2C	Inter-integrated circuit
MCU	Microcontroller unit
MEMS	Micro electro-mechanical systems
ODE	Open development environment
REST API	Representational state transfer API
SDK	Software development kit
SMD	Surface mount device
SSID	Service set identifier
UART	Universal asynchronous receiver/transmitter
URL	Uniform resource locator
Wi-Fi	Wireless LAN based on IEEE 802.11
WLAN	Wireless local area network

2 FP-CLD-AWS1 software expansion for STM32Cube

2.1 Overview

The package features:

- Complete firmware to safely connect an IoT node with sensors and actuators to Amazon AWS IoT via Wi-Fi
- Middleware libraries featuring the Amazon AWS FreeRTOS SDK, Wi-Fi and transport-level security (mbedTLS)
- Ready-to-use binaries to connect the IoT node to a web dashboard running on Amazon AWS services for sensor data visualization
- Sample implementation available for the STEVAL-STWINKT1B wireless industrial node and the B-L4S5I-IOT01A STM32L4+ Discovery kit IoT node
- Easy portability across different MCU families, thanks to STM32Cube
- Free, user-friendly license terms

The package integrates the AWS FreeRTOS SDK middleware with APIs to simplify interaction between SensorTile or the Discovery Kit for IoT node and the Amazon AWS IoT services.

You can use it to prototype end-to-end sensors-to-cloud IoT applications, by registering your board to ST dashboard and begin exchanging real-time sensor data.

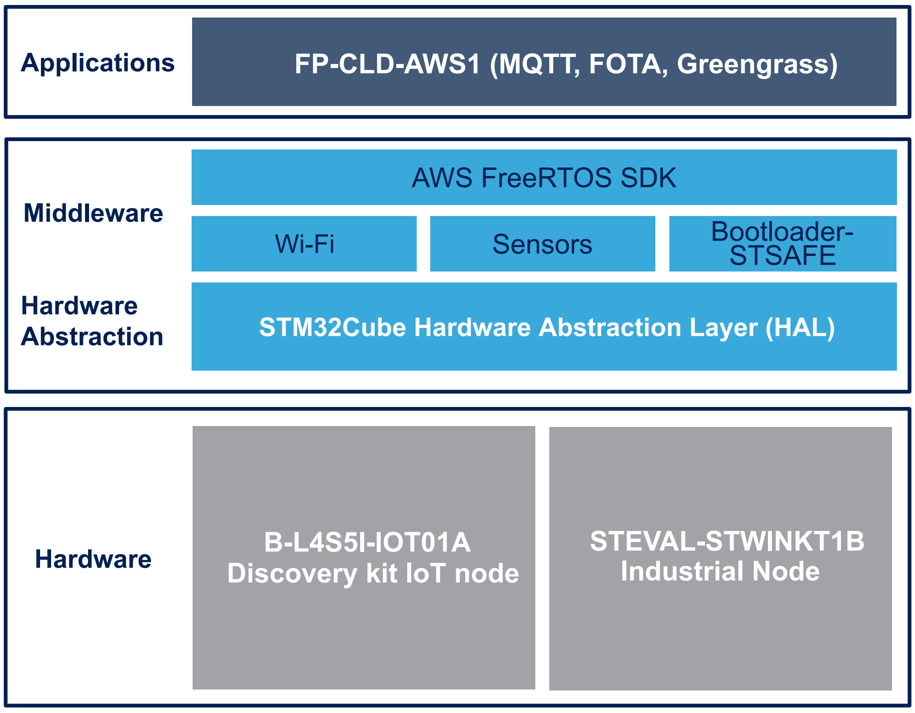
The web dashboard based on Amazon AWS is also provided free of charge to facilitate the evaluation of the function pack. For AWS license terms, visit https://aws.amazon.com.

2.2 Architecture

The software layers used by the application software to access and use the STM32 microcontroller, the Wi-Fi and the sensor expansion boards are:

- **STM32Cube HAL layer**: a simple, generic and multi-instance set of APIs (application programming interfaces) to interact with the upper layer applications, libraries and stack layers. The APIs are based on a common framework so that overlying software can implement functions and routines without specific microcontroller unit (MCU) hardware configurations. This structure improves library code reusability and guarantees easy portability across other devices.
- **Board support package (BSP) layer**: drives the STM32 Nucleo board peripherals such as LEDs, user button, etc. (except the MCU), with a specific set of APIs. This interface also helps in identifying the specific board version.
- **Middleware layers**: contains FreeRTOS SDK() to facilitate the connection of STM32 devices to AWS IoT services.

Figure 1. **FP-CLD-AWS1 software architecture**


Applications
FP-CLD-AWS1 (MQTT, FOTA, Greengrass)
Middleware
AWS FreeRTOS SDK
Wi-Fi
Sensors
Bootloader-STSAFE
Hardware Abstraction
STM32Cube Hardware Abstraction Layer (HAL)
Hardware
B-L4S5I-IOT01A Discovery kit IoT node
STEVAL-STWINKT1B Industrial Node

2.3 Folder structure

The folders in the software package are:

- **Documentation**: with a compiled HTML file generated from the source code detailing the software components and APIs (one for each project).
- **Drivers**: the HAL drivers and the board-specific drivers for each supported board or hardware platform, including those for the on-board components, and the CMSIS vendor-independent hardware abstraction layer for the ARM Cortex-M processor series.
- **Middlewares**: the middleware interface for Wi-Fi and mbedTLS, together with the porting of Amazon FreeRTOS SDK.
- **Utilities**: contains TeraTerm script parameters
- **Projects**: contains sample applications for STEVAL-STWINKT1B and B-L4S5I-IOT01A which can be used to register a device and transmit sensors data to Amazon AWS IoT; together with the companion web dashboard, these sample applications enable easy sensor data visualization and device control through AWS IoT. Sample applications can be compiled under the IAR Embedded Workbench for ARM, RealView Microcontroller Development Kit or STM32CubeIDE for STM32 development environments. For each sample application, the folder also contains the correspondent pre-compiled binary.

3 How to run the firmware application

Before you begin, you should set up the hardware configuration that is compatible with the FP-CLD-AWS1 function pack.

Below you will find the general steps that you should follow to set up your remote IoT application with cloud monitoring and control capability.

Step 1. Download firmware package from www.st.com.

Step 2. Recompile the provisioning tool application code or flash pre-compiled binaries to your board.

Step 3. Run the provisioning tool application to personalize the STSAFE-A110 secure element.

Step 4. Configure and launch a serial terminal application to check whether provisioning has been completed.

Step 5. Build Secure Bootloader: recompile the 2_Images_SECoreBin and 2_Images_SBSFU

Step 6. Recompile the *aws_demo tools* application code or flash pre-compiled binaries to your board.

Step 7. Run the *aws_demo tools* application.

Step 8. Configure and launch a serial terminal application to do input/output from device.

Step 9. Copy the Device Certificate displayed on the terminal.

Step 10. Open the web dashboard.

Step 11. Register your board in the web dashboard using the device certificate from the terminal display.

Step 12. Enter the device ID from the dashboard to the terminal.

Step 13. Complete Wi-Fi setup by entering SSID, security protocol and password in the terminal.

Step 14. Login to the dashboard and select the device for data telemetry.

Step 15. Get sensor data and display them on the dashboard.

RELATED LINKS

4 System setup guide on page 22

3.1 Board security credential provisioning

The STEVAL-STWINKT1B and the B-L4S5I-IOT01A come with an on-board STSAFE-A110 secure element support.

The secure element includes an inbuilt Device Certificate which is used in the Device Registration leveraging the AWS IoT Core Multi-Account Registration feature.

3.1.1 Blank board provisioning

The blank board requires device provisioning by programming and running the image from *Projects/<board_name>/Applications/BootLoader_STSAFE/STSAFE_Provisioning/Binary/Provisioning.bin* so that the STSAFE-A110 secure element and the MCU bootloader are paired and can communicate securely with each other.

Caution: Provision the STSAFE-A110 secure element only once, as provisioning it multiple times can lock the device and make it unusable.

The provisioning tool is used to customize STSAFE-A110 secure element for secure boot and secure firmware update (SBSFU) application.

The project has to be compiled to generate the binary file to be loaded into the STM32 which allows customizing STSAFE-A110 chip.

Provisioning tools also provide the values of pairing keys (Host_MAC_Key and Host_Cipher_Key) to SECoreBin project.

RELATED LINKS

UM2262 - Getting started with SBSFU software expansion for STM32Cube

3.2 Serial terminal setup

To receive the provisioning process output, you have to set up a serial terminal on a PC.

A serial terminal is also required to:

- read the device certificate from STSAFE-A110 (used as device certificate to register the device in the web dashboard)
- configure the board (device ID and Wi-Fi parameters)
- display the published and received AWS IoT messages locally

The example below refers to the STEVAL-STWINKT1B board and is based on Tera Term, but you can use any other similar tool.

Step 1. When using the STEVAL-STWINKT1B board for the first time

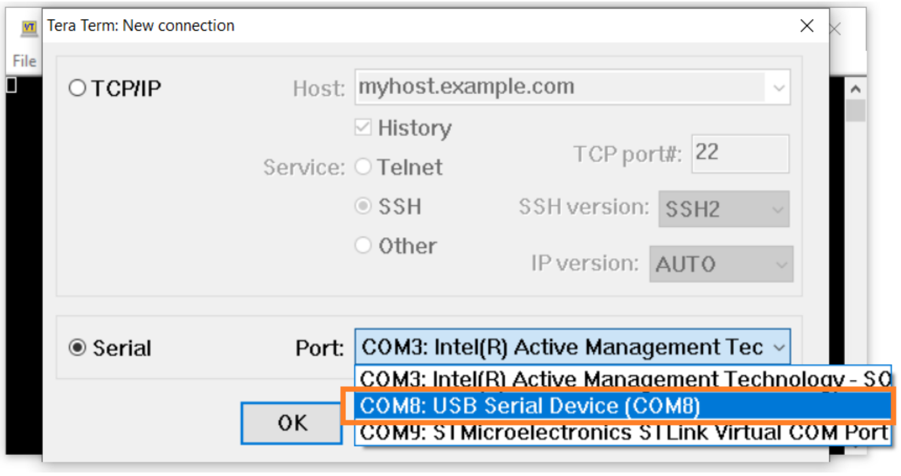
Step 1a. connect its ST-LINK port to an STLINK-V3MINI device

Step 1b. connect the STLINK-V3MINI to your PC USB port

Step 1c. connect the board USB micro device port to the PC USB port via USB micro cable

Step 2. Determine the USB device virtual COM port used for the STEVAL-STWINKT1B.

Figure 2. Opening the COM port for the STEVAL-STWINKT1B

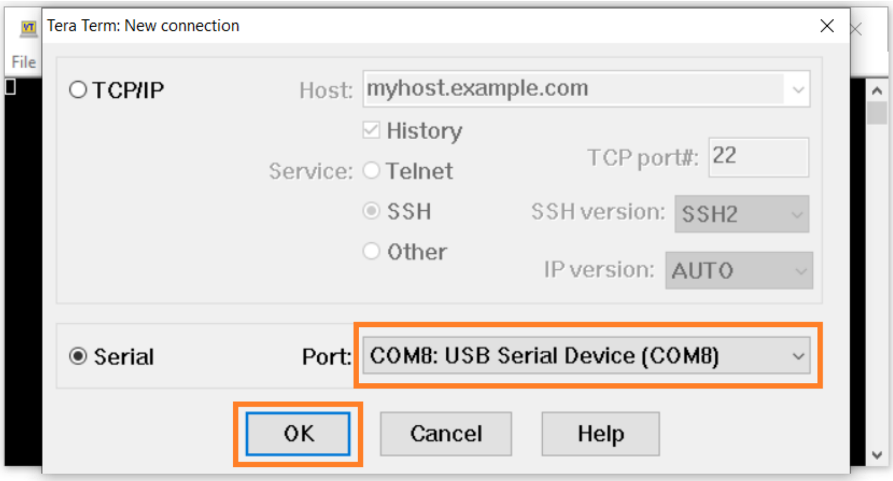

Tera Term: New connection

TCP/IP   Host: myhost.example.com
[✓] History
Service: ○ Telnet   TCP port#: 22
● SSH   SSH version: SSH2
○ Other   IP version: AUTO

● Serial   Port: COM3: Intel(R) Active Management Tec
COM3: Intel(R) Active Management Technology - SO
COM8: USB Serial Device (COM8)
COM9: STMicroelectronics STLink Virtual COM Port

Step 3. Configure the COM port number and the other parameters as shown below.

Figure 3. Serial COM port setup



Tera Term: New connection

File

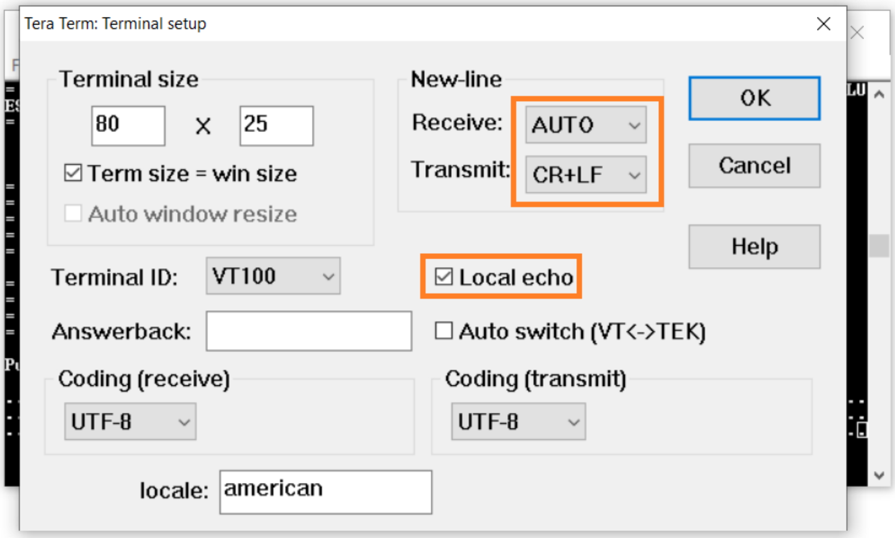
( ) TCP/IP   Host: myhost.example.com
[✓] History   TCP port#: 22
Service: ( ) Telnet
(●) SSH   SSH version: SSH2
( ) Other   IP version: AUTO

(●) Serial   Port: COM8: USB Serial Device (COM8)

OK   Cancel   Help

Step 4. Configure the terminal with the recommended parameters shown below.

Figure 4. Terminal setup



Tera Term: Terminal setup

Terminal size: 80 x 25
[✓] Term size = win size
[ ] Auto window resize

New-line
Receive: AUTO
Transmit: CR+LF

OK   Cancel   Help

Terminal ID: VT100
[✓] Local echo
Answerback:
[ ] Auto switch (VT<->TEK)
Coding (receive): UTF-8
Coding (transmit): UTF-8
locale: american

Note: *The virtual terminal new-line transmit configuration must be set to LineFeed (\n or LF) to allow copying and pasting from the UNIX type text files. The “Local echo” option makes copy and paste visible on the console.*

Step 5. Flash the board with the pre-compiled binaries provided in the FP-CLD-AWS1 package.

Step 6. Reset the board and follow the indications on the serial terminal to complete the provisioning process.

Important:
When using the B-L4S5I-IOT01A Discovery kit for the first time, you have to connect it to your PC USB port and determine the USB device virtual COM port used for the board.

3.3 Secured bootloader

The pre-integrated bootloader allows updating the user application (initially, one of the standard FreeRTOS user applications: *aws_demos*), adding new features and correcting potential issues. This is performed in a secure way, which prevents any unauthorized update.

The secure bootloader strengthens hardware security mechanisms on the STM32 by guaranteeing a unique entry point after reset, ensuring immutable boot code, reinforcing security check and performing firmware image verification (authenticity and integrity) before execution.

It exploits STM32 device hardware protection mechanisms:

- data protection reading (external access disablement, boot option protections)
- WRP/PCROP: data protection writing (protection of code and keys from Flash dump, as well as of trusted code)
- MPU: execution allowed in a chain of trust
- firewall: to protect the Flash memory and RAM at run-time (secure enclave for critical operations)

The secure bootloader is a standalone executable and is delivered in the package as a pre-built executable file and as source file, allowing advanced users to rebuild it with different settings.

Caution: The delivered pre-compiled bootloader does not enforce security, so that the user can still plug a debugger and debug the application. It must not be used in a production context.

Pre-compiled bootloader settings:

- dual-image mode
- enable local loader
- secure IP turned off
- X.509 ECDSA_SHA256 asymmetric authentication
- No firmware encryption

The dual-image mode enables safe image update with firmware image backup and rollback capabilities; the bootloader is totally integrated, so the user can still use the IDE when programming or debugging.

However, specific pre- and post-build phases are added to the sample projects:

- the post-build phase combines the bootloader with the application and overwrites the application executable so that it can be programmed and run as usual;
- the pre-build phase deletes the .elf file to force the link and post-build, even if the user project sources have not changed. It prevents post-build dependencies issues with STM32CubeIDE.

The source files are compiled and linked as usual, creating a binary file and an .elf format file. After the link, a post-build script is executed. A file with the .sfb extension is created, containing the standalone user application, optionally encrypted, and prefixed with meta data, ready for being downloaded and written at run-time to the firmware update slot.

The script outputs modified .elf and raw binary files, which combine the non-encrypted bootloader and the user application.

3.3.1 X-CUBE-SBSFU software expansion

The bootloader is based on X-CUBE-SBSFU software expansion technology. FP-CLD-AWS1 contains some X-CUBE-SBSFU sub-modules:

- Project/<board_name>/Applications/BootLoader_STSAFE/2_Images_SBSFU
- Project/<board_name>/Applications/BootLoader_STSAFE/2_Images_SECoreBin
- Project/<board_name>/Applications/BootLoader_STSAFE/Linker_Common

SECorebin contains the secure engine core sources which represent a protected environment where all critical data and operations can be managed in a secure way.

SBSFU implements the secure boot and secure firmware update with the secure engine core support.

Linker_Common contains the linker file to link the user application.

The original X-CUBE-SBSFU package is slightly modified to support the FreeRTOS user applications (OTA update image state management, offloading of PKCS#11 services to the STSAFE-A110 component) and to better integrate the IDEs. Post-script templates are also adapted to help with IDE integration.

Updating the bootloader implies rebuilding:

1. the Secure Engine library located in the 2_Images_SE_CoreBin folder
2. the Secure Boot/Secure Firmware Update executable (the result is called bootloader)

Depending on the IDE, the bootloader is located in:

- IAR Embedded Workbench
  Project/<board_name>/Applications/Bootloader_STSAFE/2_Images_SBSFU/EWARM/<boardname>Exe/Project.out
- STM32CubeIDE
  Project/<board_name>/Applications/Bootloader_STSAFE/2_Images_SBSFU/STM32CubeIDE/<board_name>_2_Images_SBSFU/Debug/SBSFU.elf

Rebuilding the bootloader is required if the bootloader configuration or some linker script files are changed.

The configuration is defined by several files and is based on C pre-processor statements:

- 2_Images_SBSFU/SBSFU/App/app_sfu.h for SBSFU settings
- 2_Images_SECoreBin/Inc/se_crypto_config.h for cryptography settings

The current bootloader is configured as follows:

- `SECBOOT_ECCDSA_WITH_AES128_CBC_SHA256`
- `SFU_DEBUG_MODE`
- `SECBOOT_DISABLE_SECURITY_IPS`

The secure firmware image is encrypted, but hardware protections are not enforced. For instance, the JTAG link is on, so that debugging is still possible.

RELATED LINKS

Refer to the release note in the X-CUBE-AWS software expansion package for information about the related X-CUBE-SBSFU version.

For further information, refer to X-CUBE-SBSFU sofware expansion package.

3.3.2 Programming user application

When security is enforced by the bootloader, it may be necessary to reprogram the option bytes to clear some protections and revert to RDPL0 before reprogramming the Flash memory.

Reprogramming is performed via STM32CubeProgrammer or STM32 ST-LINK Utility.

3.3.3 Debugging the application

The debugger retrieves the debug information from the user application .elf file (*aws_demos* or *aws_tests* in the present case). It has no access to the bootloader symbols. The debugger places a break-point on the user application `main()` function.

Upon reset:

- in IAR Embedded Workbench, the debugger starts the application at 0x0800 0000 to execute the bootloader. For this purpose, IAR Embedded Workbench is given an extra option: `[Debugger]>[extra options]>[command line option] --drv_vector_table_base=0x0800 0000`
- the STM32CubeIDE debugger requires a modification of the .elf file to start at 0x0800 0000.
  This modification is automatically performed during the project post-build stage (this is specific to STM32CubeIDE). The bigelftuner.exe changes the global .elf file entry point (combining the bootloader and the application) and sets this entry point to 0x0800 0000 to start the bootloader. The debugger places a break-point on the user application `main()` function.

3.3.4 Application boot

When launched, the bootloader enforces security and checks whether new firmware has be installed: if the case, it decrypts the new firmware if necessary and checks it before installing and running it; otherwise, it starts the already installed application.

3.4 Provisioned board configuration

Step 1. After provisioning the board, build, program, and run the aws demo application to complete board configuration.

Step 2. Build, program and run the image from:
Projects\<board_name>\Applications\Cloud\aws_demos\<toolchain>\PostBuild\SBSFU_<board_name>_aws_demos.bin (no need to configure it in the source code).

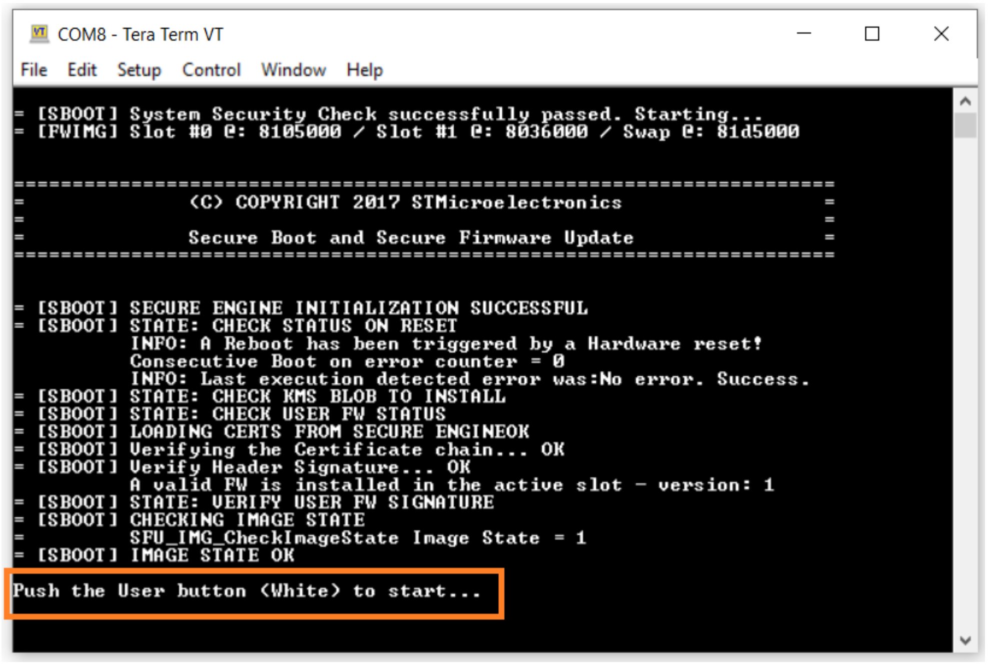
Step 3. Run the demo application:

Step 3a. Copy the SBSFU_<board_name>_aws_demos.bin and paste it to STLINK_V3S (Drive Letter) drive name visible under Windows System.
If outputs are not available in the terminal update, run STLINK-V3MINI firmware.

Step 3a. Flash the SBSFU_<board_name>_aws_demos.bin. fw to the device using STM32Cube Programmer

Step 3b. Flash the SBSFU_<board_name>_aws_demos.bin from the IDE.

Figure 5. First boot screenshot

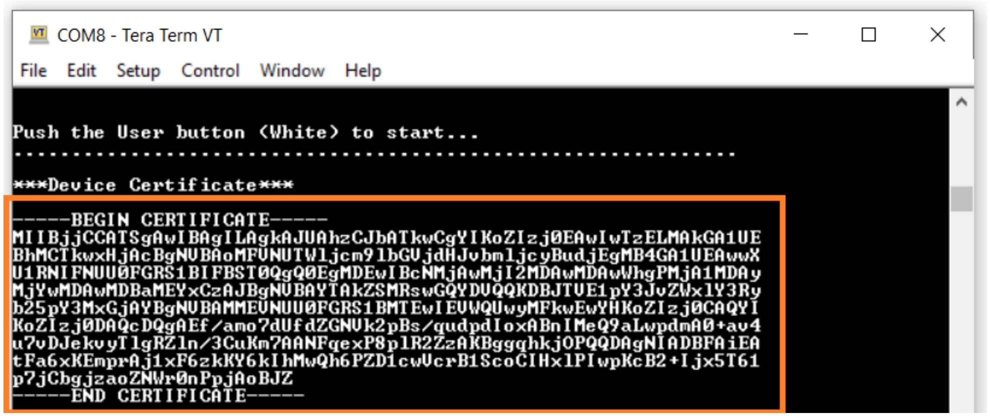


```
= [SBOOT] System Security Check successfully passed. Starting...
= [FWIMG] Slot #0 @: 8105000 / Slot #1 @: 8036000 / Swap @: 81d5000


======================================================================
=              (C) COPYRIGHT 2017 STMicroelectronics                 =
=                                                                    =
=              Secure Boot and Secure Firmware Update                =
======================================================================


= [SBOOT] SECURE ENGINE INITIALIZATION SUCCESSFUL
= [SBOOT] STATE: CHECK STATUS ON RESET
          INFO: A Reboot has been triggered by a Hardware reset!
          Consecutive Boot on error counter = 0
          INFO: Last execution detected error was:No error. Success.
= [SBOOT] STATE: CHECK KMS BLOB TO INSTALL
= [SBOOT] STATE: CHECK USER FW STATUS
= [SBOOT] LOADING CERTS FROM SECURE ENGINEOK
= [SBOOT] Verifying the Certificate chain... OK
= [SBOOT] Verify Header Signature... OK
          A valid FW is installed in the active slot - version: 1
= [SBOOT] STATE: VERIFY USER FW SIGNATURE
= [SBOOT] CHECKING IMAGE STATE
=         SFU_IMG_CheckImageState Image State = 1
= [SBOOT] IMAGE STATE OK

Push the User button (White) to start...
```

Step 4. Push the user button to move forward.

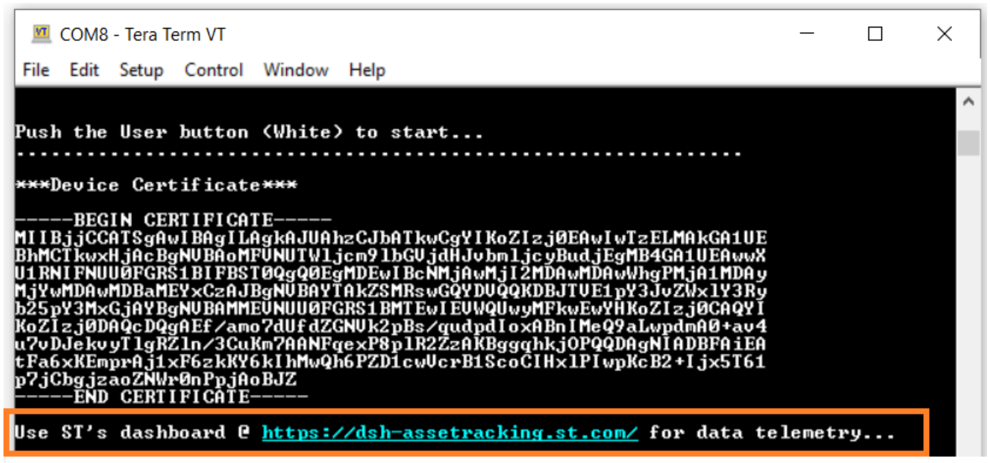
Step 5. Copy the PEM format certificate displayed on the virtual terminal to "my_extracted_cert.pem" text file.

Figure 6. Device certificate on terminal screen



```
Push the User button (White) to start...
--------------------------------------------------------------

***Device Certificate***
-----BEGIN CERTIFICATE-----
MIIBjjCCATSgAwIBAgILAgkAJUAhzCJbATkwCgYIKoZIzj0EAwIwTzELMAkGA1UE
BhMCTkwxHjAcBgNVBAoMFVNUTW1jcm9lbGVjdHJvbmljcyBudjEgMB4GA1UEAwwX
U1RNIFNUU0FGRS1BIFBST0QgQ0EgMDEwIBcNMjAwMjI2MDAwMDAwWhgPMjA1MDAy
MjYwMDAwMDBaMEYxCzAJBgNVBAYTAkZSMRswGQYDVQQKDBJTVE1pY3JvZWxlY3Ry
b25pY3MxGjAYBgNVBAMMEVNUU0FGRS1BMTEwIEVWQUwyMFkwEwYHKoZIzj0CAQYI
KoZIzj0DAQcDQgAEf/amo7dUfdZGNVk2pBs/qudpdIoxABnIMeQ9aLwpdmA0+av4
u7vDJekvyTlgRZln/3CuKm7AANFqexP8plR2ZzAKBggqhkjOPQQDAgNIADBFAiEA
tFa6xKEmprAj1xF6zkKY6kIhMwQh6PZD1cwVcrB1ScoCIHx1PIwpKcB2+Ijx5T61
p7jCbgjzaoZNWr0nPpjAoBJZ
-----END CERTIFICATE-----
```

Step 6. Go to the dashboard link and register a device using "my_extracted_cert.pem" text file device certificate.



```
Push the User button (White) to start...
------------------------------------------------------------

***Device Certificate***

-----BEGIN CERTIFICATE-----
MIIBjjCCATSgAwIBAgILAgkAJUAhzCJbATkwCgYIKoZIzj0EAwIwTzELMAkGA1UE
BhMCTkwxHjAcBgNVBAoMFVNUTW1jcm9lbGVjdHJvbmljcyBudjEgMB4GA1UEAwwX
U1RNIFNUU0FGRS1BIFBST0QgQ0EgMDEwIBcNMjAwMjI2MDAwMDAwWhgPMjA1MDAy
MjYwMDAwMDBaMEYxCzAJBgNVBAYTAkZSMRswGQYDVQQKDBJTVE1pY3JvZWxlY3Ry
b25pY3MxGjAYBgNVBAMMEVNUU0FGRS1BMTEwIEVWQUwyMFkwEwYHKoZIzj0CAQYI
KoZIzj0DAQcDQgAEf/amo7dUfdZGNVk2pBs/qudpdIoxABnIMeQ9aLwpdmA0+av4
u7vDJekvyTlgRZln/3CuKm7AANFqexP8plR2ZzAKBggqhkjOPQQDAgNIADBFAiEA
tFa6xKEmprAj1xF6zkKY6kIhMwQh6PZD1cwVcrB1ScoCIHxlPIwpKcB2+Ijx5T61
p7jCbgjzaoZNWr0nPpjAoBJZ
-----END CERTIFICATE-----
Use ST's dashboard @ https://dsh-assetracking.st.com/ for data telemetry...
```

Figure 7. **Dashboard link to register AWS Thing**

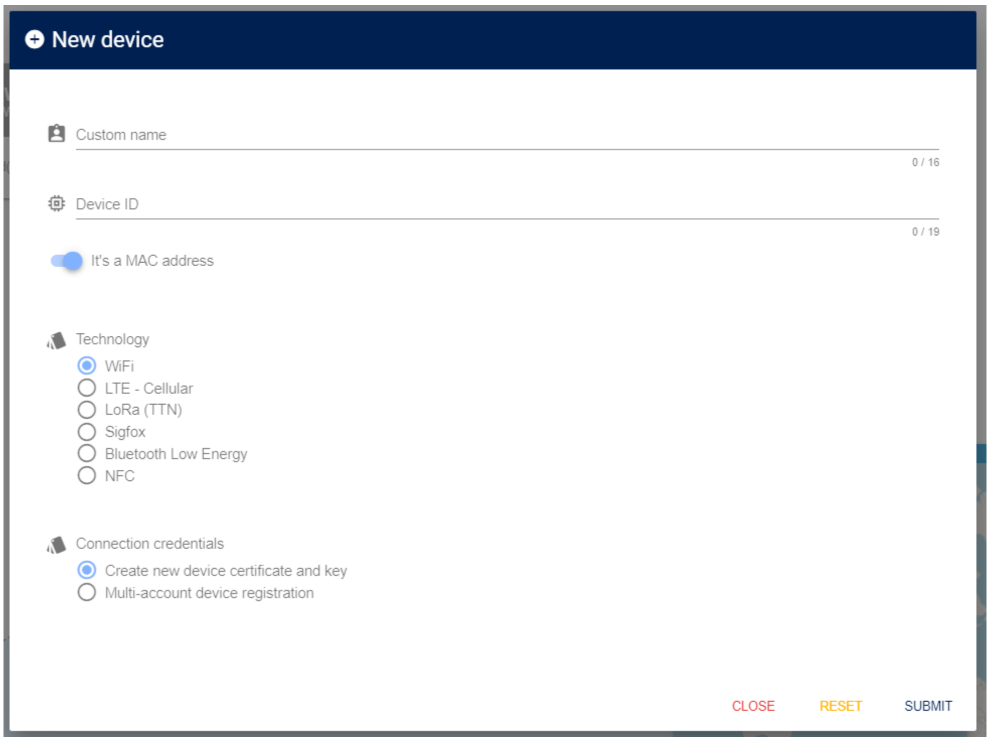
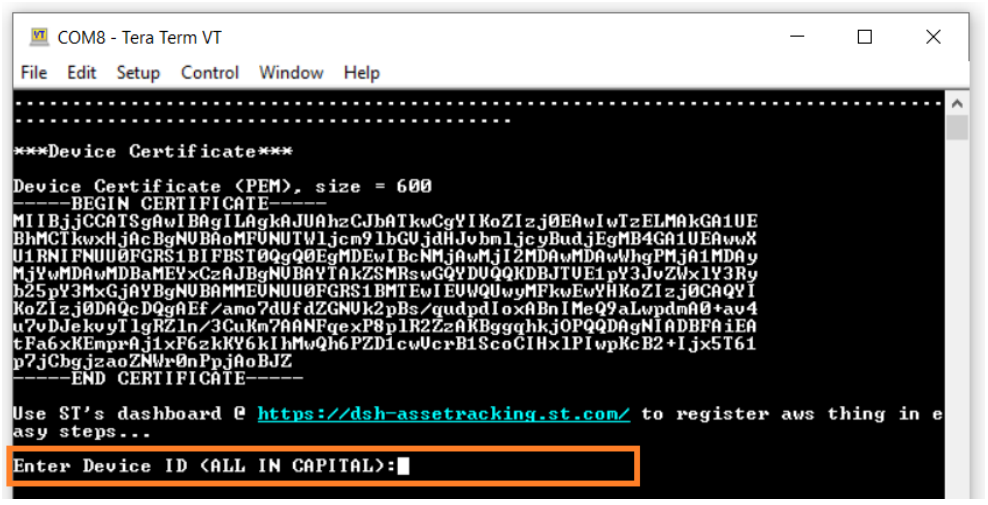


Figure 8. **Register a new device in the dashboard**

Step 7. Enter the device ID received in the dashboard back in the terminal

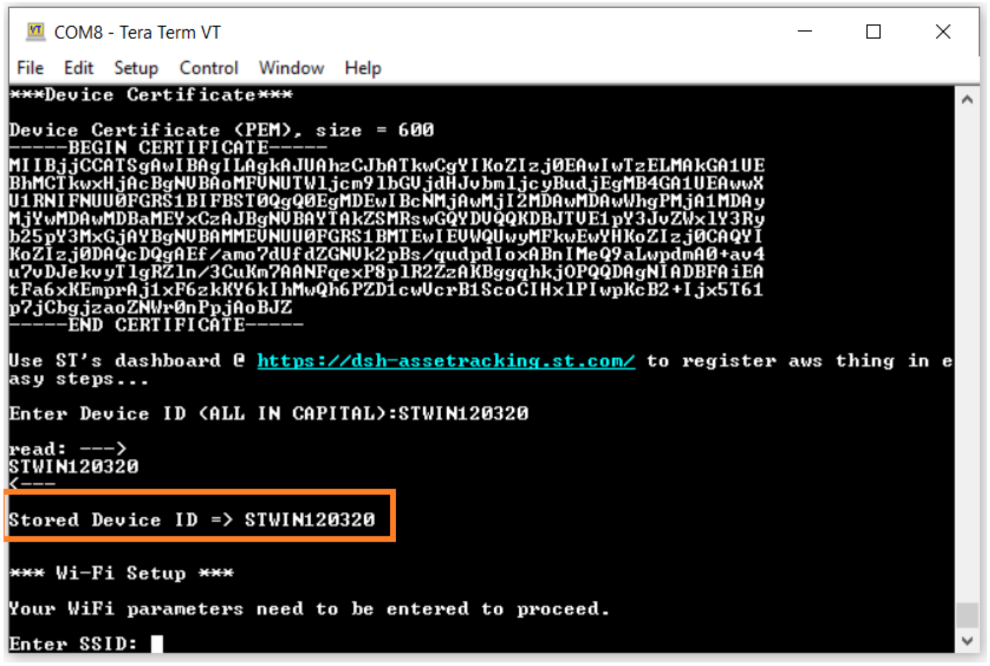


COM8 - Tera Term VT

File Edit Setup Control Window Help

```
--------------------------------------------------------------------------------
-------------------------------------------

***Device Certificate***

Device Certificate (PEM), size = 600
-----BEGIN CERTIFICATE-----
MIIBjjCCATSgAwIBAgILAgkAJUAhzCJbATkwCgYIKoZIzj0EAwIwTzELMAkGA1UE
BhMCTkwxHjAcBgNVBAoMFVNUTW1jcm9lbGVjdHJvbmljcyBudjEgMB4GA1UEAwwX
U1RNIFNUU0FGRS1BIFBST0QgQ0EgMDEwIBcNMjAwMjI2MDAwMDAwWhgPMjA1MDAy
MjYwMDAwMDBaMEYxCzAJBgNVBAYTAkZSMRswGQYDVQQKDBJTVE1pY3JvZWxlY3Ry
b25pY3MxGjAYBgNVBAMMEVNUU0FGRS1BMTEwIEVWQUwyMFkwEwYHKoZIzj0CAQYI
KoZIzj0DAQcDQgAEf/amo7dUfdZGNVk2pBs/qudpdIoxABnIMeQ9aLwpdmA0+av4
u7vDJekvyTlgRZln/3CuKm7AANFqexP8plR2ZzAKBggqhkjOPQQDAgNIADBFAiEA
tFa6xKEmprAj1xF6zkKY6kIhMwQh6PZD1cwVcrB1ScoCIHxlPIwpKcB2+Ijx5T61
p7jCbgjzaoZNWr0nPpjAoBJZ
-----END CERTIFICATE-----

Use ST's dashboard @ https://dsh-assetracking.st.com/ to register aws thing in e
asy steps...

Enter Device ID (ALL IN CAPITAL):
```

Figure 9. Entering device ID in the terminal



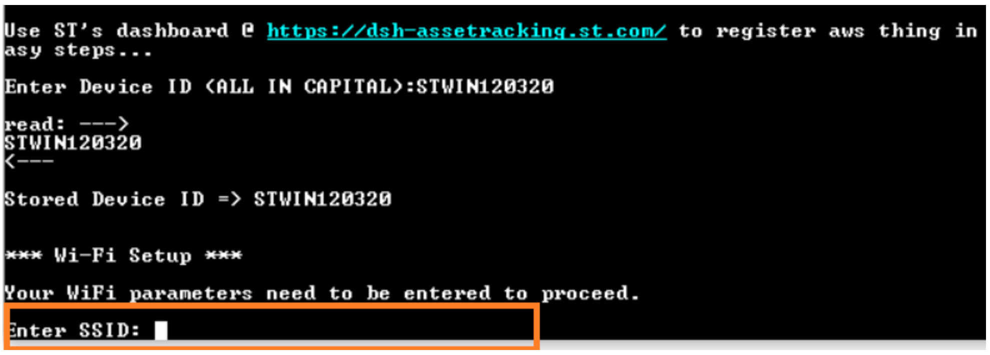
COM8 - Tera Term VT

File Edit Setup Control Window Help

```
***Device Certificate***

Device Certificate (PEM), size = 600
-----BEGIN CERTIFICATE-----
MIIBjjCCATSgAwIBAgILAgkAJUAhzCJbATkwCgYIKoZIzj0EAwIwTzELMAkGA1UE
BhMCTkwxHjAcBgNVBAoMFVNUTW1jcm9lbGVjdHJvbmljcyBudjEgMB4GA1UEAwwX
U1RNIFNUU0FGRS1BIFBST0QgQ0EgMDEwIBcNMjAwMjI2MDAwMDAwWhgPMjA1MDAy
MjYwMDAwMDBaMEYxCzAJBgNVBAYTAkZSMRswGQYDVQQKDBJTVE1pY3JvZWxlY3Ry
b25pY3MxGjAYBgNVBAMMEVNUU0FGRS1BMTEwIEVWQUwyMFkwEwYHKoZIzj0CAQYI
KoZIzj0DAQcDQgAEf/amo7dUfdZGNVk2pBs/qudpdIoxABnIMeQ9aLwpdmA0+av4
u7vDJekvyTlgRZln/3CuKm7AANFqexP8plR2ZzAKBggqhkjOPQQDAgNIADBFAiEA
tFa6xKEmprAj1xF6zkKY6kIhMwQh6PZD1cwVcrB1ScoCIHxlPIwpKcB2+Ijx5T61
p7jCbgjzaoZNWr0nPpjAoBJZ
-----END CERTIFICATE-----

Use ST's dashboard @ https://dsh-assetracking.st.com/ to register aws thing in e
asy steps...

Enter Device ID (ALL IN CAPITAL):STWIN120320

read: --->
STWIN120320
<---

Stored Device ID => STWIN120320

*** Wi-Fi Setup ***

Your WiFi parameters need to be entered to proceed.

Enter SSID:
```

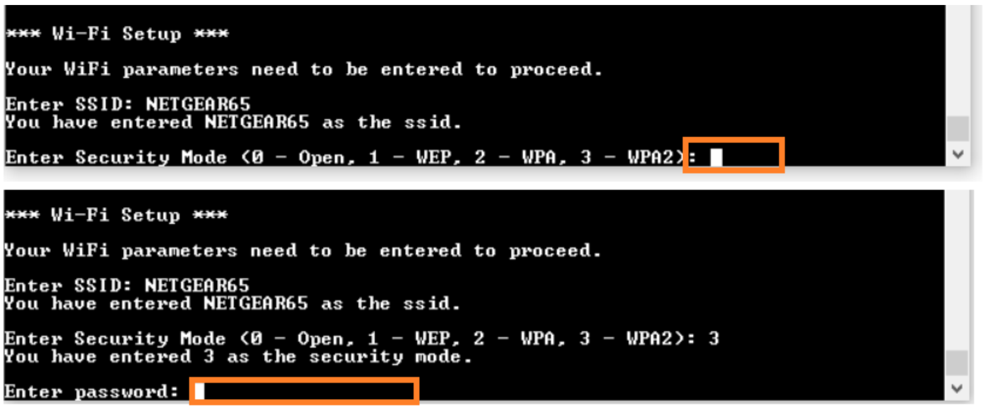
Figure 10. Device ID stored

Step 8. Enter Wi-Fi SSID, select security mode and enter password.



```
Use ST's dashboard @ https://dsh-assetracking.st.com/ to register aws thing in
asy steps...

Enter Device ID (ALL IN CAPITAL):STWIN120320

read: --->
STWIN120320
<---

Stored Device ID => STWIN120320


*** Wi-Fi Setup ***

Your WiFi parameters need to be entered to proceed.
Enter SSID:
```

Figure 11. Wi-Fi SSID

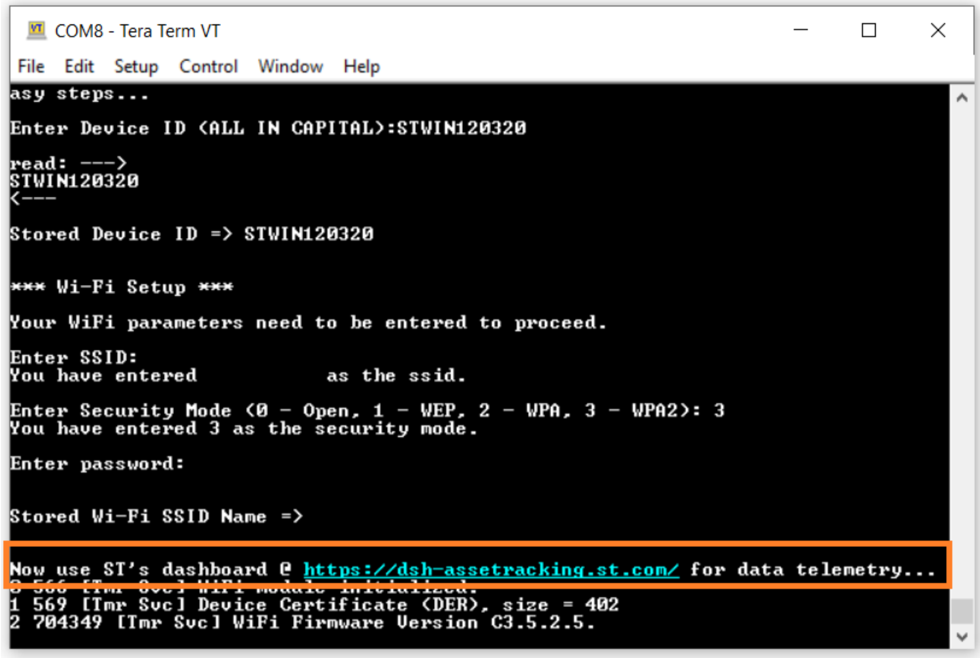


```
*** Wi-Fi Setup ***

Your WiFi parameters need to be entered to proceed.

Enter SSID: NETGEAR65
You have entered NETGEAR65 as the ssid.

Enter Security Mode (0 - Open, 1 - WEP, 2 - WPA, 3 - WPA2):
```

```
*** Wi-Fi Setup ***

Your WiFi parameters need to be entered to proceed.

Enter SSID: NETGEAR65
You have entered NETGEAR65 as the ssid.

Enter Security Mode (0 - Open, 1 - WEP, 2 - WPA, 3 - WPA2): 3
You have entered 3 as the security mode.

Enter password:
```

Figure 12. Security mode and password

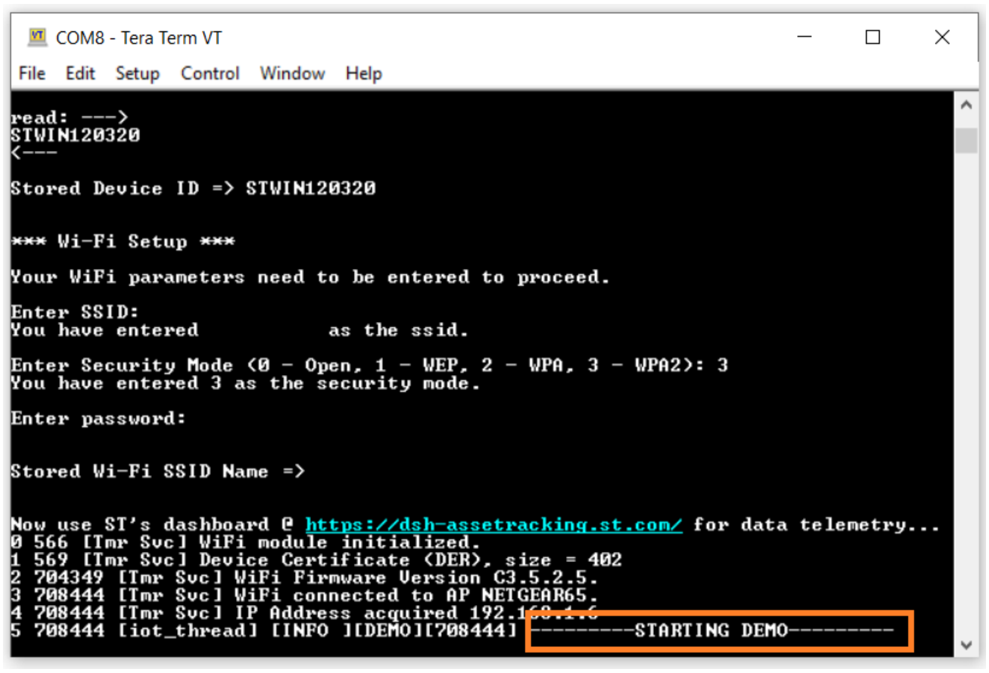
The device is now connected to the dashboard to which you can connect to in order to visualize data.



```
asy steps...

Enter Device ID (ALL IN CAPITAL):STWIN120320

read: --->
STWIN120320
<---

Stored Device ID => STWIN120320


*** Wi-Fi Setup ***

Your WiFi parameters need to be entered to proceed.

Enter SSID:
You have entered            as the ssid.

Enter Security Mode (0 - Open, 1 - WEP, 2 - WPA, 3 - WPA2): 3
You have entered 3 as the security mode.

Enter password:


Stored Wi-Fi SSID Name =>


Now use ST's dashboard @ https://dsh-assetracking.st.com/ for data telemetry...
0 566 [Tmr Svc] WiFi module initialized.
1 569 [Tmr Svc] Device Certificate (DER), size = 402
2 704349 [Tmr Svc] WiFi Firmware Version C3.5.2.5.
```

Figure 13. Device provisioning complete

Figure 14. Device ready to connect to the dashboard



```
read: --->
STWIN120320
<---

Stored Device ID => STWIN120320

*** Wi-Fi Setup ***

Your WiFi parameters need to be entered to proceed.

Enter SSID:
You have entered                as the ssid.

Enter Security Mode (0 - Open, 1 - WEP, 2 - WPA, 3 - WPA2): 3
You have entered 3 as the security mode.

Enter password:

Stored Wi-Fi SSID Name =>

Now use ST's dashboard @ https://dsh-assetracking.st.com/ for data telemetry...
0 566 [Tmr Svc] WiFi module initialized.
1 569 [Tmr Svc] Device Certificate (DER), size = 402
2 704349 [Tmr Svc] WiFi Firmware Version C3.5.2.5.
3 708444 [Tmr Svc] WiFi connected to AP NETGEAR65.
4 708444 [Tmr Svc] IP Address acquired 192.168.1.6
5 708444 [iot_thread] [INFO ][DEMO][708444] ---------STARTING DEMO---------
```

3.5 How to register the board to the ST IoT web dashboard

To run the test application included in the FP-CLD-AWS1 package and connect it to the web dashboard, it is necessary to register the board using the device certificate provided by STSAFE-A110 during boot process into Terminal.

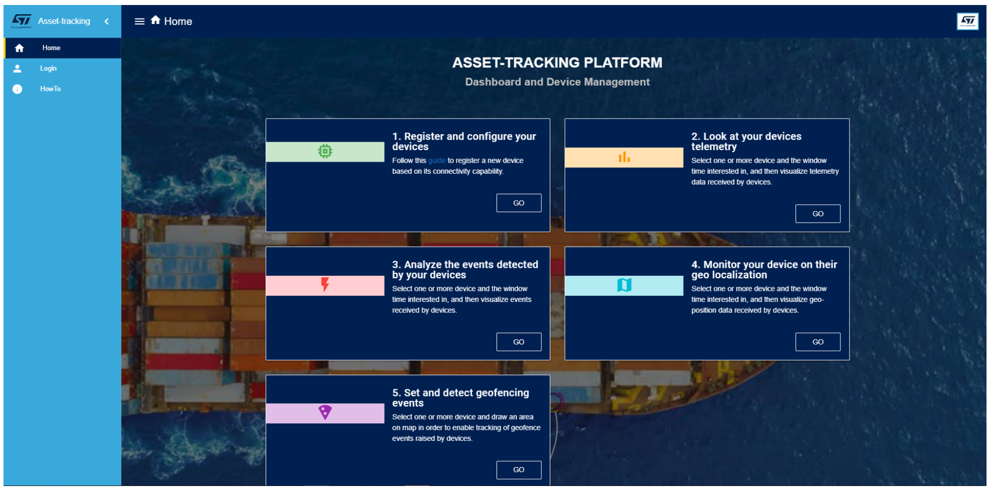
The communication between ST development boards and AWS IoT is protected by X.509 device certificate, a digital certificate that uses public key infrastructure (PKI) standard.

The dashboard automates the Thing Registration in the ST owned AWS account used to deploy and run the web application.

Step 1. To register a board, go to DSH-ASSETRACKING and click on [**Go To Site**].

MyST.COM page opens up: create a log-in in myST.com or re-use your log-in if you already have an account, sign the license agreement and then proceed with the steps outlined hereafter.

Figure 15. ST IoT web dashboard



ASSET-TRACKING PLATFORM
Dashboard and Device Management

1. Register and configure your devices
Follow this guide to register a new device based on its connectivity capability.

2. Look at your devices telemetry
Select one or more device and the window time interested in, and then visualize telemetry data received by devices.

3. Analyze the events detected by your devices
Select one or more device and the window time interested in, and then visualize events received by devices.

4. Monitor your device on their geo localization
Select one or more device and the window time interested in, and then visualize geo-position data received by devices.

5. Set and detect geofencing events
Select one or more device and draw an area on map in order to enable tracking of geofence events raised by devices.

Step 2. Under [**New Device**]:

Step 2a. in the [**Custom Name**] field, enter a user-friendly name for your device

Step 2b. in the [**Board ID**] field, enter a string to be used as AWS Thing name (slide the toggle button ([**It's a MAC address**]) off to enter a string instead of a MAC address)

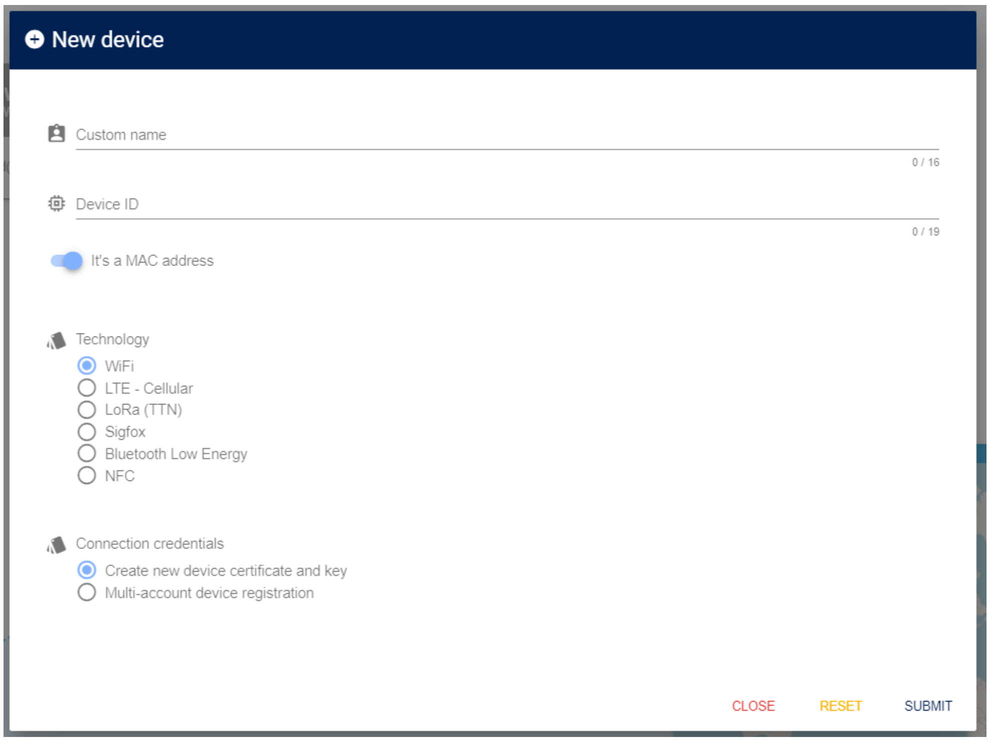
Step 2c. in the [**Technology**] field, select [**Wi-Fi**]

Step 2d. in the [**Connection Credentials**] field, select [**Multi-Account device registration**]

Step 2e. paste the device certificate copied from the terminal into [**Certificate PEM String**]

Step 2f. click on [**Submit**] to register your device

Figure 16. **New board registration in the ST AWS dashboard**



New device

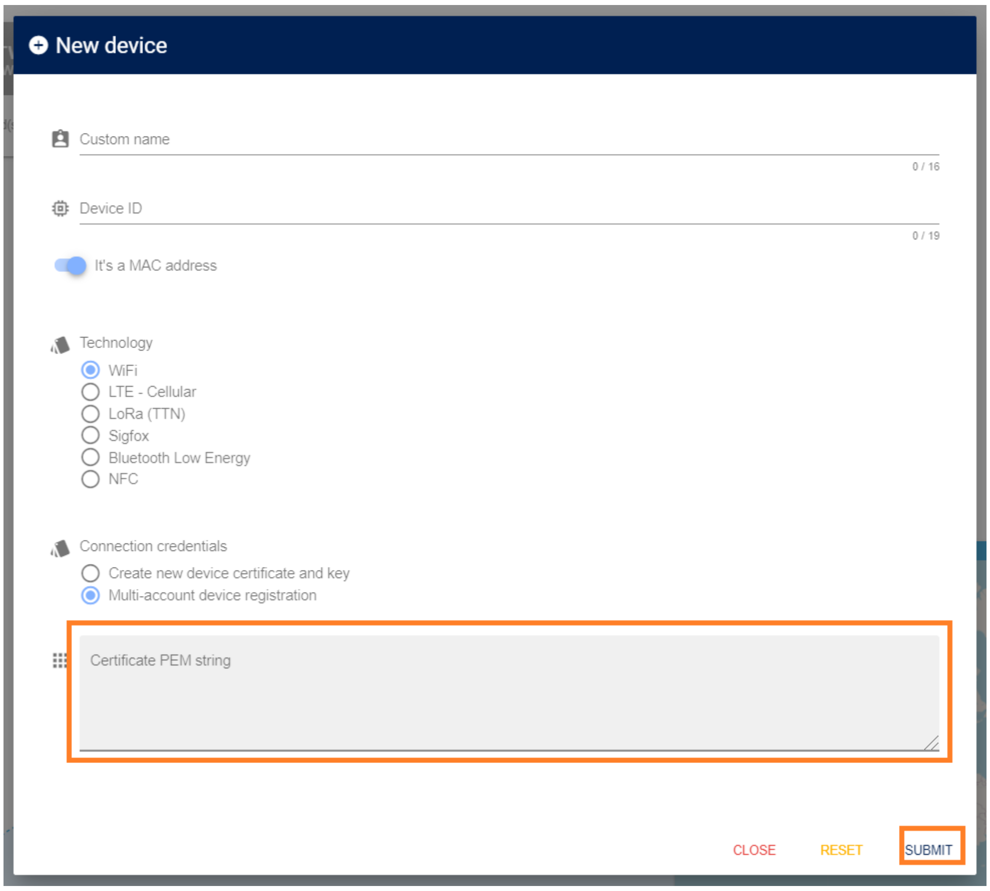
Custom name
0 / 16

Device ID
0 / 19

It's a MAC address

Technology
[✓ WiFi]
LTE - Cellular
LoRa (TTN)
Sigfox
Bluetooth Low Energy
NFC

Connection credentials
[✓ Create new device certificate and key]
Multi-account device registration


New device

Custom name 0 / 16

Device ID 0 / 19

It's a MAC address

Technology
[✓ WiFi]
LTE - Cellular
LoRa (TTN)
Sigfox
Bluetooth Low Energy
NFC

Connection credentials
Create new device certificate and key
[✓ Multi-account device registration]

Certificate PEM string

Figure 17. Certificate PEM string for new board registration

The web application automatically creates the AWS Thing in the AWS IoT core.
After successful creation, the following web icon appears.

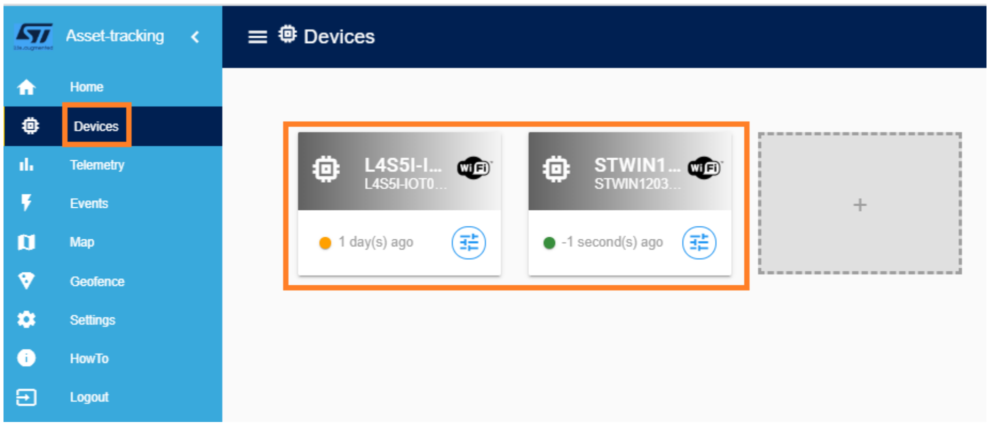


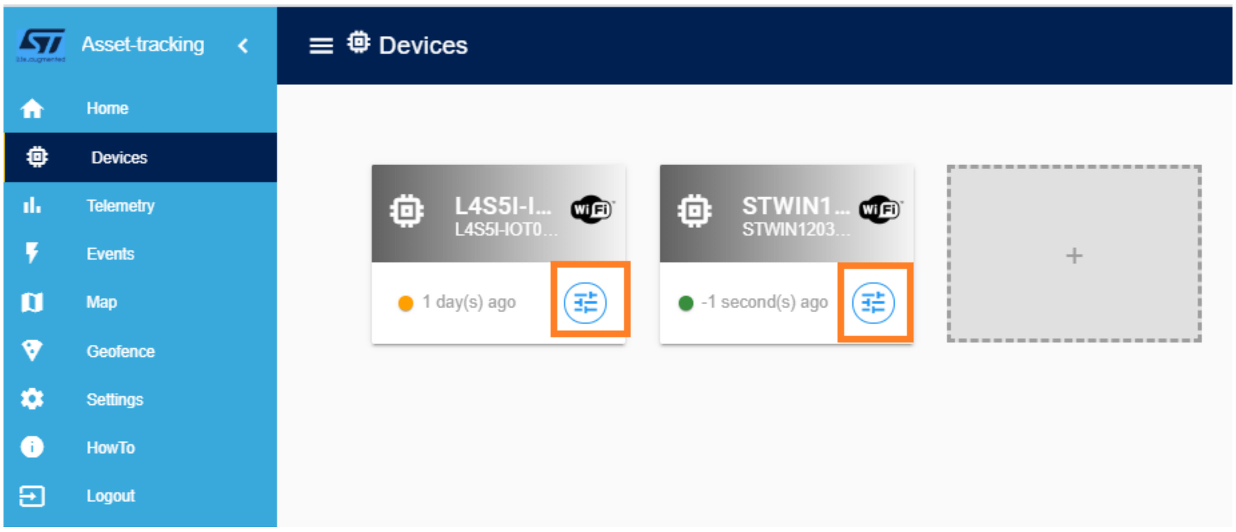
Figure 18. Registered device list

Note: The AWS Thing is created in a proprietary ST AWS account.

3.6 Delete a registered board

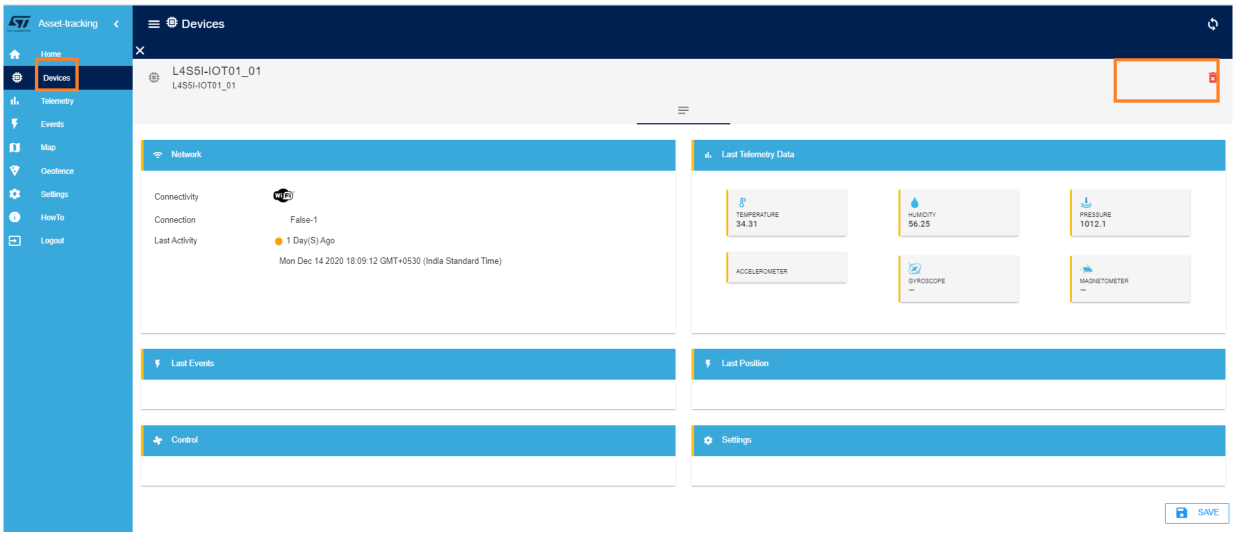
You can remove a registered board and the corresponding thing created in the AWS IoT Core directly from the web dashboard by selecting [**Devices**] in the left panel, clicking on the device icon to be deleted.

Figure 19. Dashboard: delete a registered board section



Then, you have to click on the red bin on the upper right angle to confirm device deletion.

Figure 20. Dashboard: delete a registered board section



L4S5I-IOT01_01
L4S5I-IOT01_01

Connectivity	
Connection	False-1
Last Activity	1 Day(S) Ago
	Mon Dec 14 2020 18:09:12 GMT+0530 (India Standard Time)

TEMPERATURE 34.31	HUMIDITY 56.25	PRESSURE 1012.1
ACCELEROMETER	GYROSCOPE –	MAGNETOMETER –

3.7 Telemetry data application

Once the device configuration is complete, the board sends sensor data to the AWS IoT cloud.

The telemetry data application is based on iot_demo_mqtt source code. The telemetry demo is the default demonstration of FP-CLD-AWS1 package and demonstrates the MQTT publish workflow.

The demo publishes bursts of data to a topic name to send data to dashboard. When a message arrives in the IoT core, the dashboard shows it to the user.

RELATED LINKS

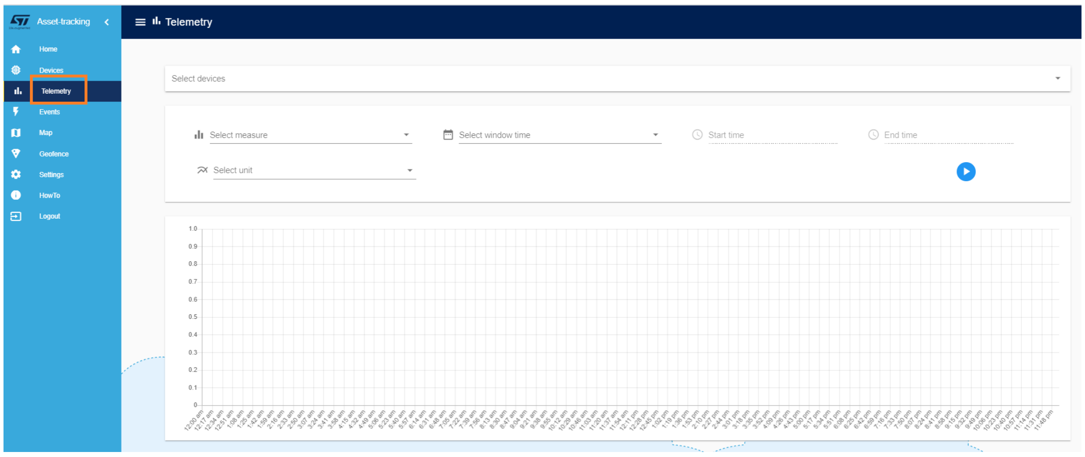
For further details on aws iot_demo_mqtt, refer to https://docs.aws.amazon.com/freertos/latest/lib-ref/c-sdk/mqtt/mqtt_demo.html

3.7.1 Telemetry data demonstration in the web dashboard

To visualize sensor data, follow the procedure below.

Step 1. Log in to the dashboard and select [**Telemetry**] from the left panel.

Figure 21. **Telemetry data graph**



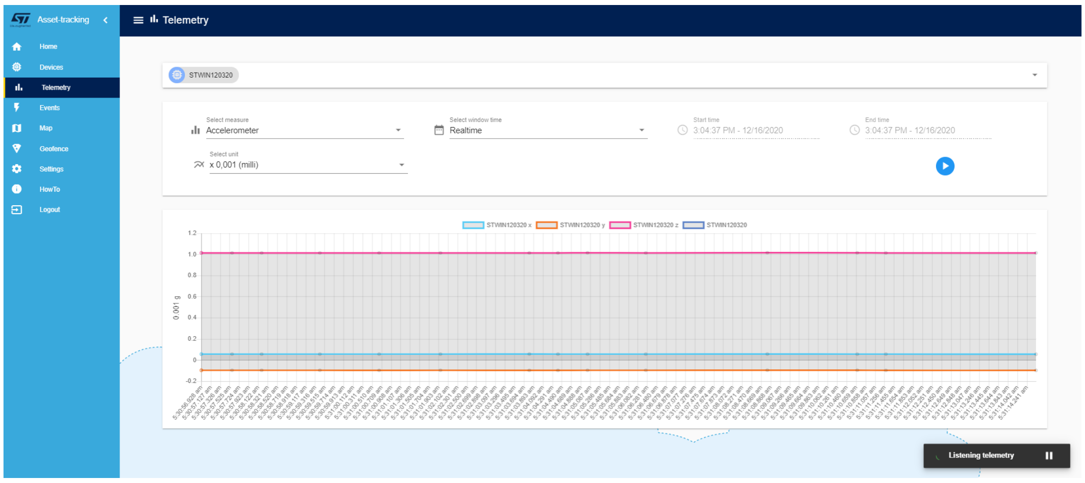
Sensor data transmitted by the device are shown in the visualization page under [**Select measure**].

Step 2. Select the sensor to use for data visualization.

Step 3. Under [**Select window time**] select [**Realtime**] for real-time streaming of data.

Step 4. Push [**Play**] to visualize data.

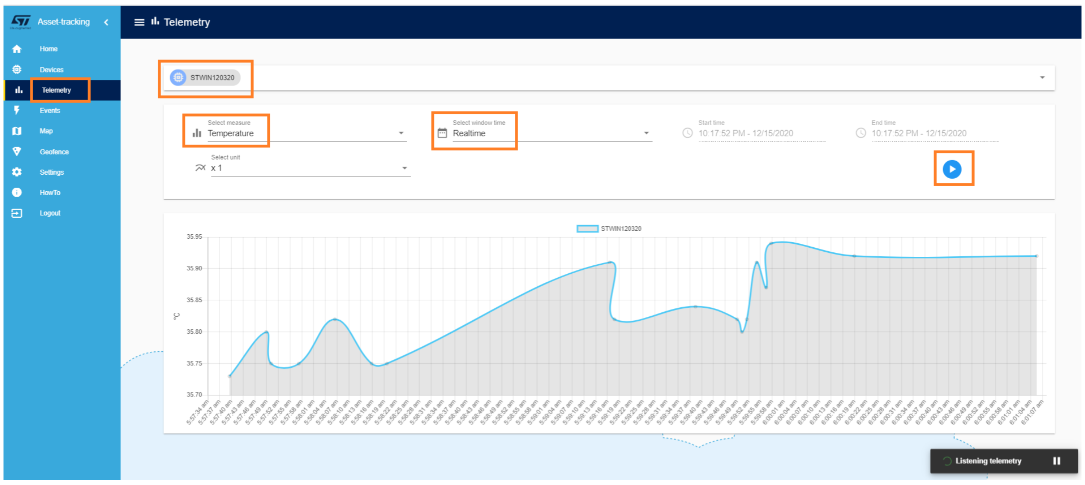
Figure 22. **Accelerometer data**



Step 5. Visualize temperature, humidity, pressure, accelerometer, gyroscope and magnetometer values in graphic charts from the telemetry window.

You can choose to display these values for a period of 1, 3, 6 or custom-hour time-frame from [**Select Window Time**]

Figure 23. Temperature data



3.8 MQTT OTA firmware update demonstration

The Amazon Web Services firmware over-the-air (AWS FOTA) library combines the AWS jobs and FOTA download libraries to create a user-friendly library that can perform an over-the-air firmware update using MQTT TLS. It connects to the specified broker using the existing certificates and uses TLS for the MQTT connection. Data sent in each MQTT message are encrypted.

This demonstration application connects to the configured AWS IoT MQTT broker and subscribes to several topics related to AWS IoT jobs. When an update job is created in the AWS IoT service, the application receives a notification through MQTT. Triggered by the notification, the application receives the update job over MQTT.

Step 1. To run the MQTT OTA firmware update application, log in to your account in the AWS console as the firmware update application cannot be run from ST dashboard account.

Step 2. Open Middlewares\Third_Party\amazon-freertos\demos\include\ aws_clientcredential.h and assign new values to the below defines according to your user account.

– `#define clientcredentialMQTT_BROKER_ENDPOINT`
– `#define clientcredentialIOT_THING_NAME`
– `#define clientcredentialWIFI_SSID`
– `#define clientcredentialWIFI_PASSWORD`
– `#define clientcredentialWIFI_SECURITY`

Step 3. Define a project level (`#define USE_PARAMETER_HARD_CODING`)

Step 4. Open Projects\ <board_name>\Applications\Cloud\aws_demos\config_files/aws_demo_config.h, comment out `#define CONFIG_MQTT_DEMO_ENABLED`, and define `CONFIG_OTA_UPDATE_DEMO_ENABLED`.

Step 5. To run MQTT OTA application, re-build the application with the new configuration, program and launch it.

aws_demos successively:

– connects to the endpoint over TCP/IP and Wi-Fi
– authenticates itself through TLS through the pre-provisioned device certificate
– enters the MQTT wait loop, pending on the reception of an OTA firmware update job

Step 6. Test OTA update:

Step 6a. create a code signing certificate and profile in the AWS console (refer to "Creating a code-signing certificate for the FreeRTOS Windows simulator page" of the FreeRTOS user guide to generate and register your own ECDSA code signing certificate)

Step 6b. copy the code signing certificate string to the pcClientCertificatePem[] static array in the OTA PAL (Projects/<board_name>/Applications/Cloud/aws_demos/Src/ports/aws_ota_pal.c)

Important:
This code signing certificate is not the device certificate (“my_device_cert_arn”) previously used for the device registration.

Step 7. Prepare the application update file and upload it to AWS.

Step 7a. Increment the application version in aws_application_version.h. This is mandatory to avoid that the update is rejected by the demo self-test.

Step 7b. Build the application, but do not program it to the board.

Step 7c. Upload the resulting <toolchain_name>/PostBuild/<board_name>_aws_demos.sfb file to an Amazon S3 bucket listed in the IAM role for OTA update job.

Step 8. Create a FreeRTOS OTA update job from the AWS console ([**IoT Core**]>[**Manage**]>[**Jobs**]>[**Create**]) and select:

– the thing to update: my_thing_name
– the MQTT protocol for image transfer (HTTP is not supported)
– [**Sign a new firmware for me**]
– the code signing profile
– the .sfb firmware file, referenced from the S3 bucket where it is uploaded
– any non-empty destination pathname (such as firmware.bin)
– the role for OTA update jobs, which gives access to the S3 buckets and to the IoT Core services

Finally, choose a unique job ID.

The application automatically:

– receives and handles the job request
– downloads the .sfb file and stores it into the Slot #0 region of the system Flash memory
– verifies the integrity and the authenticity of the .sfb file against the file signature attached to the job description, and of the AWS code signing certificate provisioned in the OTA PAL. If the file signature verification fails (this message is displayed on the device console: `[prvPAL_CloseFile] ERROR - Failed to pass sig-sha256-ecdsa signature verification`), the OTA update job is aborted and reported FAILED to the AWS OTA update service
– resets the MCU and lets the secure bootloader detect the new .sfb in Slot #0
– verify the integrity and the authenticity of the new user application against the secure bootloader certificate, install it into Slot #1 and reset the MCU again
– launches the new application version, which then performs the self-test and informs the OTA update service of the update success. In case of self-test error, the user application update is rejected and the previous application version is restored. If the new application version cannot complete the self-test within about 90 seconds, a timer expiration resets the MCU, the secure bootloader reverts to the previous application version and resets the MCU again, and the OTA update job is reported as FAILED upon reconnection.

3.9 AWS IoT Greengrass discovery demo application

Step 1. To run the Greengrass discovery demo application, log in to your own account in the AWS console as the firmware update application cannot be run from ST dashboard account.

Step 2. Open Middlewares\Third_Party\amazon-freertos\demos\include\ aws_clientcredential.h and assign new values to the below defines according to your user account.

- `#define clientcredentialMQTT_BROKER_ENDPOINT`
- `#define clientcredentialIOT_THING_NAME`
- `#define clientcredentialWIFI_SSID`
- `#define clientcredentialWIFI_PASSWORD`
- `#define clientcredentialWIFI_SECURITY`

Step 3. Define a project level (`#define USE_PARAMETER_HARD_CODING`)

Step 4. To run AWS IoT Greengrass Discovery demo for FreeRTOS, set up AWS, AWS IoT Greengrass, and AWS IoT.

For AWS setup, you have to follow the instructions in "Setting up your AWS account and permissions".

For AWS IoT Greengrass setup, you have to create a Greengrass group and then add a Greengrass core.

After setting up AWS and AWS IoT Greengrass, you need to configure some additional permissions for AWS IoT Greengrass (refer to https://docs.aws.amazon.com/freertos/latest/userguide/gg-demo.html for detailed steps and guide).

Step 5. Open Projects\ <board_name>\Applications\Cloud\aws_demos\config_files/aws_demo_config.h, comment out `#define CONFIG_MQTT_DEMO_ENABLED`, and define `CONFIG_GREENGRASS_DISCOVERY_DEMO_ENABLED`.

Step 6. Download and configure FreeRTOS

Step 7. Build, flash, and run the Greengrass demo on your device.

The Greengrass demo publishes a series of messages to the Greengrass core and to the AWS IoT MQTT client.

4 System setup guide

4.1 Hardware description

This section summarizes the hardware components needed for connecting the STM32 based platform to the Amazon AWS IoT console.

4.1.1 STEVAL-STWINKT1B wireless industrial node

The STWIN SensorTile wireless industrial node (STEVAL-STWINKT1B) is a development kit and reference design that simplifies prototyping and testing of advanced industrial IoT applications such as condition monitoring and predictive maintenance.

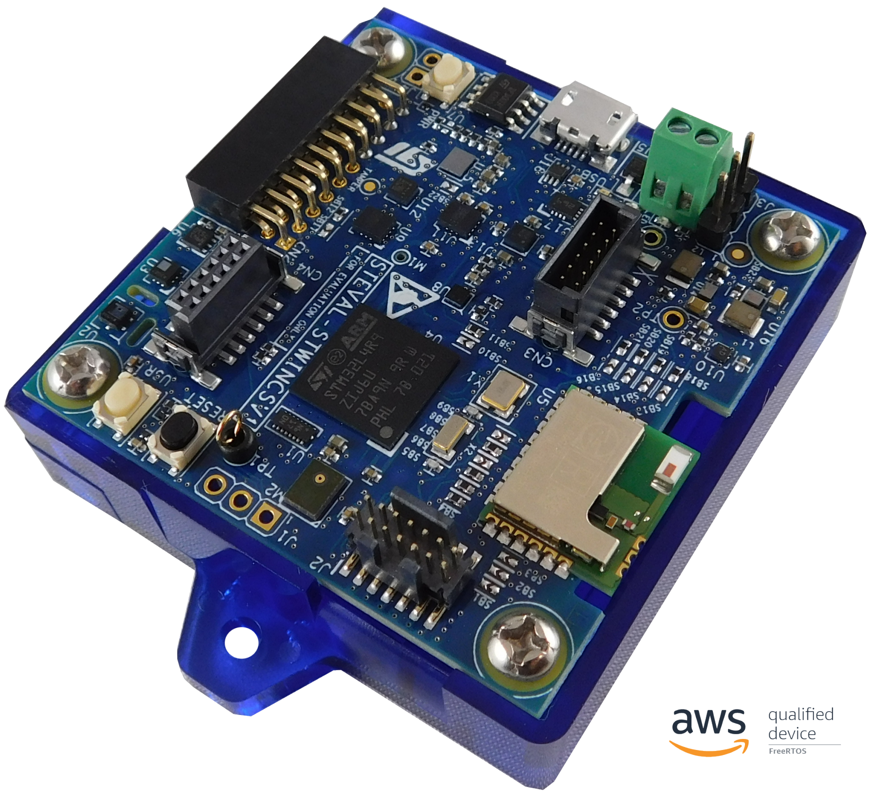
The kit features a core system board with a range of embedded industrial-grade sensors and an ultra-low-power microcontroller for vibration analysis of 9-DoF motion sensing data across a wide range of vibration frequencies, including very high frequency audio and ultrasound spectra, and high precision local temperature and environmental monitoring.

The development kit is complemented with a rich set of software packages and optimized firmware libraries, as well as a cloud dashboard application, all provided to help speed up design cycles for end-to-end solutions.

The kit supports BLE wireless connectivity through an on-board module, and Wi-Fi connectivity through a special plugin expansion board (STEVAL-STWINWFV1). Wired connectivity is also supported via an on-board RS485 transceiver. The core system board also includes an STMod+ connector for compatible, low cost, small form factor daughter boards associated with the STM32 family, such as the LTE Cell pack.

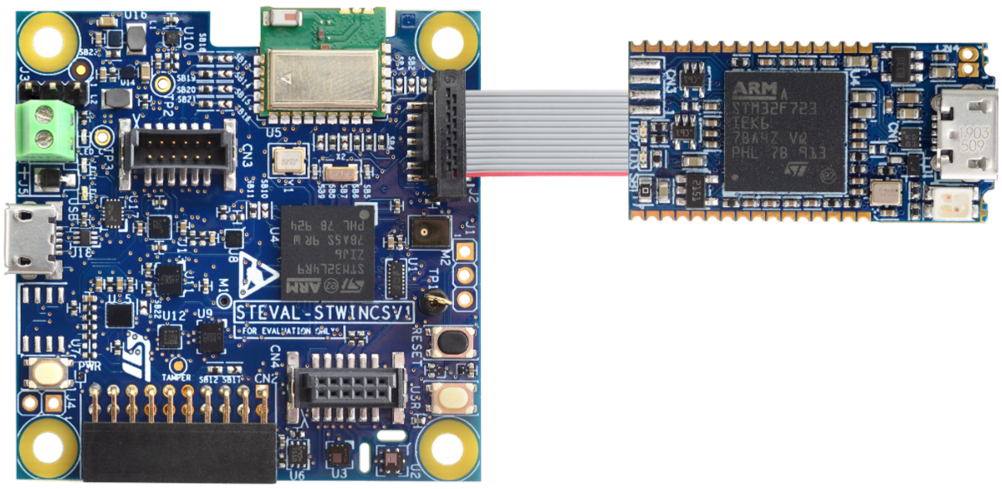
Apart from the core system board, the kit is provided complete with a 480 mAh Li-Po battery, an STLINK-V3MINI debugger and a plastic box.

Figure 24. STEVAL-STWINKT1B SensorTile Wireless Industrial Node



The STLINK-V3MINI is a standalone debugging and programming mini probe for STM32 microcontrollers, with JTAG/SWD interfaces for communication with any STM32 microcontroller located on an application board. It provides a virtual COM port interface for host PCs to communicate with target MCUs via UART. The STLINK-V3MINI is supplied with an STDC14 to STDC14 flat cable.

Figure 25. **STEVAL-STWINKT1B connected to STLINK-V3MINI**



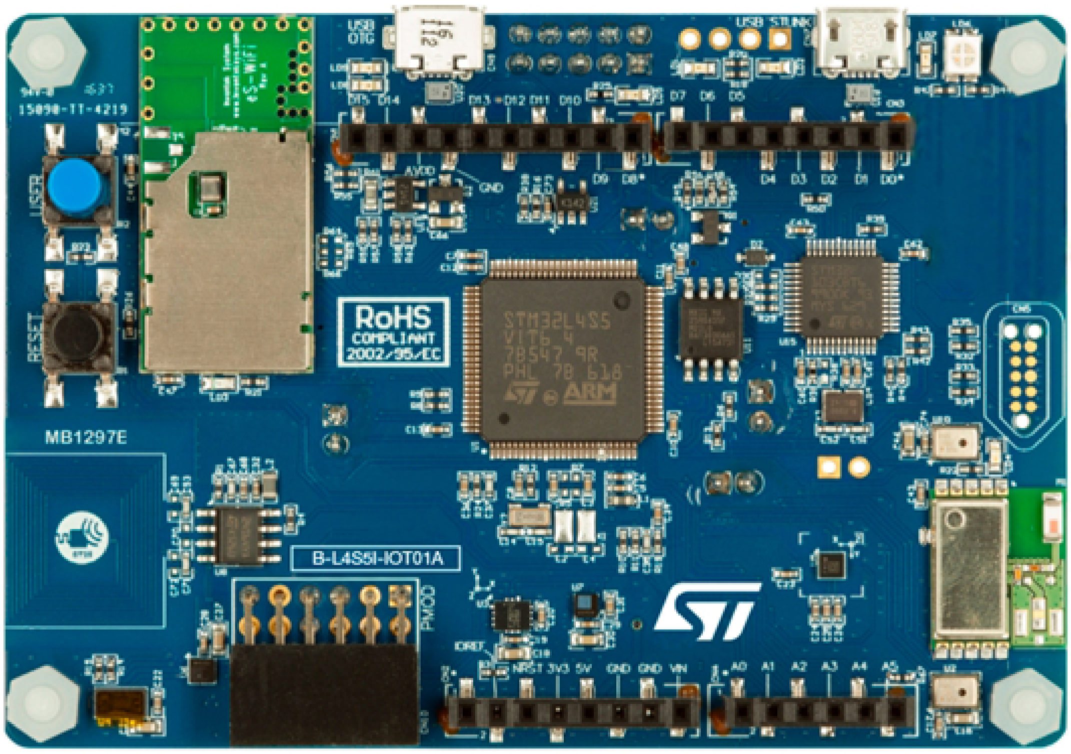
4.1.2 STM32L4+ Discovery kit IoT node

The B-L4S5I-IOT01A Discovery kit for IoT node allows you to develop applications to directly connect to cloud servers.

The Discovery kit enables a wide variety of applications by exploiting low-power communication, multi-way sensing and ARM® Cortex® -M4+ core-based STM32L4+ series features.

It supports Arduino Uno R3 and PMOD connectivity providing unlimited expansion capabilities with a large choice of dedicated add-on boards.

Figure 26. **B-L4S5I-IOT01A Discovery kit**



4.2 Hardware setup

The following hardware components are needed:

1. A wireless industrial node (order code: STEVAL-STWINKT1B)
2. A Wi-Fi expansion board (order code: STEVAL-STWINWFV1)
3. A debugging and programming mini probe (order code: STLINK-V3MINI)
4. As alternative to the board above, an STM32L4+ Discovery Kit for IoT node (order code: B-L4S5I-IOT01A)
5. Two micro USB cables to connect the platform to the PC

4.2.1 How to program STWIN with STLINK-V3MINI

Step 1. Connect the STWIN core system board to the STLINK-V3MINI programmer using the 14-pin flat cable.
The programmer and the cable are included in the STEVAL-STWINKT1B kit.

Step 2. Connect the board and the debugger to a PC via micro USB cables.

Step 3. Download the firmware onto the core system board in one of the following ways.
– Copy (or drag and drop) the .bin file to the USB mass storage created when the STEVAL-STWINKT1B is connected to the computer. This is typically STLINK_V3S on Windows PC and /media/STLINK_V3S on Linux.
– Program the STEVAL-STWINKT1B directly through ST-Link Utility or STM32CubeProgrammer.

4.2.2 How to program B-L4S5I-IOT01A

To program the B-L4S5I-IOT01A, you can choose one of the steps below.

Step 1. Copy (or drag and drop) the .bin file to the USB mass storage mount point created when the B-L4S5I-IOT01A discovery kit is connected to the computer.
This is typically DIS_L4IOT on Windows PC and /media/DIS_L4IOT on Linux.

Step 2. Program the B-L4S5I-IOT01A directly through ST-Link Utility or STM32CubeProgrammer.

If the application does not start automatically after programming, you have to reset the board through ST-Link Utility, STM32CubeProgrammer or pushing the black reset button.

4.3 Software setup

The following software components are needed in order to set up a suitable development environment for compiling and running the FP-CLD-AWS1 package:

- FP-CLD-AWS1 software
- One of the following development tool-chain and compilers:
  - IAR Embedded Workbench for ARM® (IAR-EWARM) toolchain + ST-LINK
  - STM32CubeIDE + ST-LINK

Note: *Project files for the supported IDEs can be found in FP-CLD-AWS-Release/Projects/STEVAL-STWINKT1/Applications/Cloud/AWS_demos for the STEVAL-STWINKT1B board or in FP-CLD-AWS-Release/Projects/B-L4S5I-IOT01A/Applications/Cloud/AWS for the B-L4S5I-IOT01A Discovery kit.*

- a serial line monitor (e.g. Tera Term, https://ttssh2.osdn.jp/index.html.en)

Revision history

Table 2. Document revision history

Date	Version	Changes
22-Mar-2017	1	Initial release
10-Oct-2018	2	Update all content to reflect FP-CLD-AWS1 package 2.0.0 release.
01-Aug-2019	3	Updated Introduction, Section 2.1 Overview, Section 2.2 Architecture, Section 2.3 Folder structure, Section 3.2.3 How to register the board to the ST IoT web dashboard, Section 3.4.1 Blank board, Figure 34. Device security parameter update, Section 3.5 Data visualization in the web dashboard, Section 3.6 Delete a registered board, Section 4.2 Hardware setup and Section 4.3 Software setup. Added Section 3.3 Configuration of cellular credentials, Section 3.3.1 How to register the board to the ST IoT web dashboard, Section 4.1.1 P-L496G-CELL02 discovery pack, Section 4.1.2 X-NUCLEO-IKS01A3 expansion board and Section 4.2.1 P-L496G-CELL02 Discovery Kit and X-NUCLEO-IKS01A3 expansion board setup
21-Dec-2020	4	Updated all content to reflect FP-CLD-AWS1 package 3.0.0 release.

Contents

List of figures

List of tables