Getting started with the STM32Cube function pack for STEVAL-SMARTAG2 evaluation board with dynamic NFC tag, environmental, motion, and ambient light sensors

Introduction

FP-SNS-SMARTAG2 is an STM32Cube function pack that allows you to read the ambient light, the motion, and the environmental sensor data on the STEVAL-SMARTAG2 evaluation board. These functions are performed via an NFC-enabled reader, such as a mobile phone or a tablet.

The package supports energy harvesting (enabled by NFC and only for version B of the evaluation board STEVAL$SMARTAG2B) and battery operated use cases.

This software, together with the suggested combination of STM32 and ST devices, can be used to develop tracking, cold chain, medical, smart sensing, smart home, city, and building applications.

The package contains also a simple example that shows how to update the firmware using the NFC and ST25 NFC tag application for Android/iOS.

The software runs on an STM32L4 ultralow power microcontroller. It includes drivers for the dynamic NFC tag and for the ambient light, motion, and environmental sensors.

You can register the NFC sensor tag node on the DSH-ASSETRACKING web application for asset tracking. This app stores and monitors on-board sensor data, as well as the geo-localization of the smartphone used to read the IoT node data.

The software is available also on GitHub, where the users can signal bugs and propose new ideas through [Issues] and [Pull Requests] tabs.

Related links

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

1 FP-SNS-SMARTAG2 software description

1.1 Overview

The key features of the FP-SNS-SMARTAG2 package are:

- Complete firmware to access data from an IoT node with a dynamic NFC tag, environmental, motion, and ambient light sensors
  - Ultra-low power operations, with the support of energy harvesting (only for version B of the evaluation board STEVAL$SMARTAG2B) and battery operated use cases
  - Compatible with the STAssetTracking application for Android/iOS. This allows data logs reading from the NFC tag and data logs sending to the DSH-ASSETRACKING cloud-based dashboard
  - Compatible with the STNFCSensor application for Android/iOS for reading and setting the data logs
- The package contains also one example that shows how to update the firmware using the fast transfer mode protocol (ST25FTM)
  - Compatible with the ST25 NFC tag application to download the firmware on the board via NFC
- Sample implementation available for the STEVAL-SMARTAG2 evaluation board
- Easy portability across different MCU families, thanks to STM32Cube

The package contains software to:

1. track the temperature, pressure, luminosity, and vibration values in a fixed time range, sending them via NFC. Using an Android or iOS device, you can monitor and display the logged data
   The software gathers:
   - the temperature, pressure, luminosity, and vibration sensor data for the STS22H, LPS22DF, VD6283, LSM6DSO32X, LIS2DUXS12 and ST25DV64K devices for the STEVAL-SMARTAG2 evaluation board running the STM32

   This package is compatible with the STAssetTracking Android (version 3.1.0 or higher)/iOS (version 3.1.0 or higher) application and with the STNFCSensor Android/iOS application (version 1.3.0 or higher) available on GooglePlay/App stores, to read the information sent via the NFC/RFID tag IC protocol
2. reads in energy harvesting mode the ambient light and environmental sensor data on your IoT node by the means of an NFC enabled reader, such as a mobile phone or tablet, throught a suitable AndroidST or iOST application (for only version B of the evaluation board - STEVAL$SMARTAG2B).
   The software gathers:
   - the temperature, pressure, luminosity, and vibration sensor data for the STS22H, LPS22DF, VD6283 and ST25DV64K devices for the STEVAL-SMARTAG2 evaluation board running the STM32
3. update the firmware, using the ST25 NFC tag application (version 3.7.0 or higher) available on GooglePlay/App stores

1.2 Architecture

The STM32Cube function packs leverage the modularity and interoperability of the STM32 Nucleo and the expansion boards, as well as of the STM32Cube and the expansion software, in order to create function examples, embodying some of the most common use cases for each application area.

These software function packs are designed to exploit as much as possible the underlying STM32 ODE hardware and software components to fit best the requirements of final users' applications.

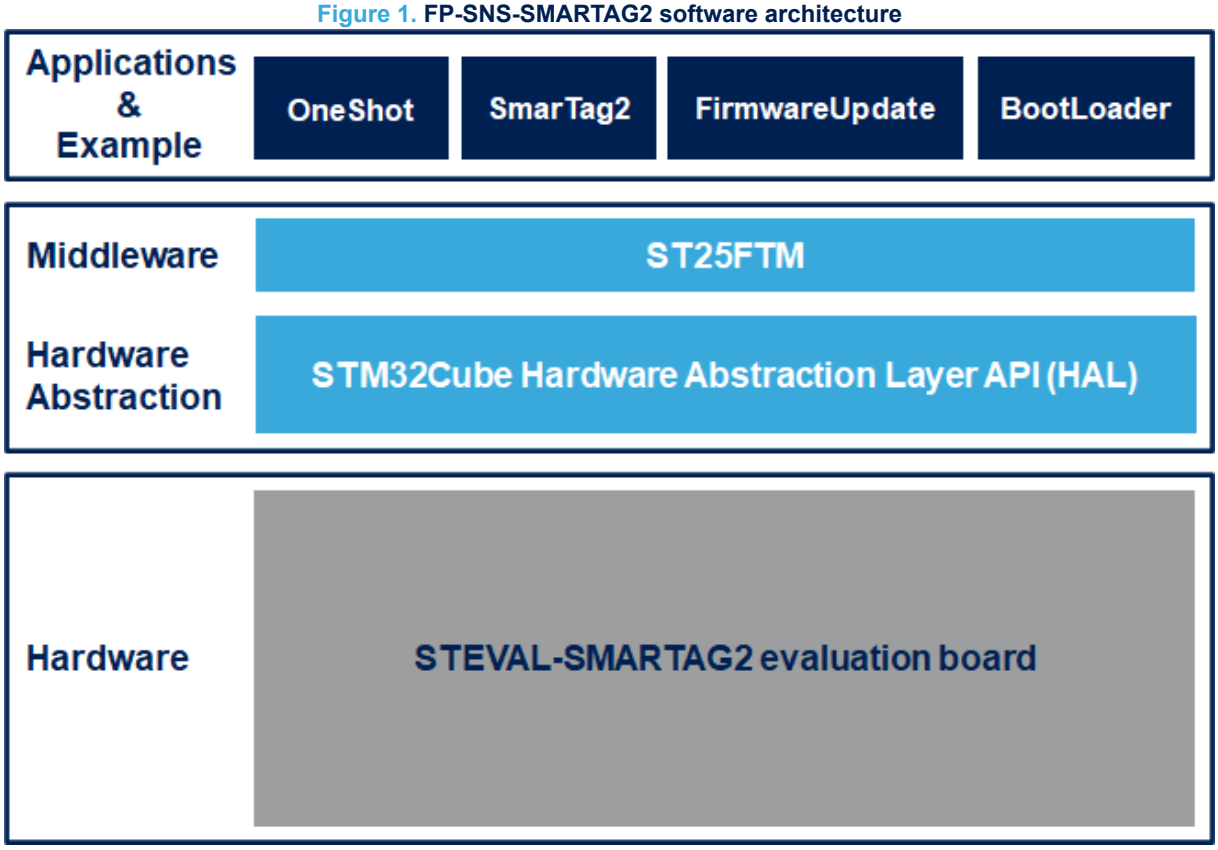
Moreover, function packs may include additional libraries and frameworks, which do not present the original expansion software packages, thus enabling new functionalities and creating a real and usable system for developers.

To access and use the sensor expansion board, the application software uses:

- **STM32Cube HAL layer**: provides a simple, generic, and multiinstance set of generic and extension APIs to interact with the upper layer application, libraries, and stacks. It is directly based on a generic architecture and allows the layers that are built on it, such as the middleware layer, to implement their functions without requiring the specific hardware configuration for a given microcontroller unit (MCU). This structure improves the library code reusability and guarantees an easy portability across other devices.

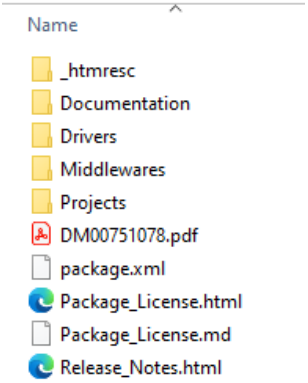
- **Board support package (BSP) layer:** supports the peripherals on the STM32 Nucleo development board (except the MCU) with a limited set of APIs, providing a programming interface for certain board-specific peripherals like the LED, the user button, etc. It helps determine the specific board version. For the sensor expansion board, it provides the programming interface for various inertial and environmental sensors and supports the sensor data initialization and reading.

Figure 1. FP-SNS-SMARTAG2 software architecture



1.3 Folder structure

Figure 2. FP-SNS-SMARTAG2 folders



The following folders are included in the software package:

- **Documentation**: contains a compiled HTML file generated from the source code, detailing the software components and APIs.
- **Drivers**: contains the HAL drivers, the board specific drivers for each supported board or hardware platform, including the on-board components, and the CMSIS vendor-independent hardware abstraction layer for the Cortex-M processor series.
- **Middlewares**: contains libraries and protocols for the ST25 fast transfer mode (ST25FTM).
- **Projects**: contains
  - four different projects for the STEVAL-SMARTAG2, which are compatible with the IAR Embedded Workbench for ARM, Real View Microcontroller Development Kit (MDK-ARM) and the integrated development environment for STM32 (STM32CubeIDE)
    1. an example (SmarTag2) used to track the motion, luminosity, and environmental sensor data on your IoT node via an NFC-enabled reader, such as a mobile phone or a tablet
    2. an example (OneShot) used to read in energy harvesting mode the ambient light and environmental sensor data on your IoT node by the via an NFC enabled reader, such as a mobile phone or tablet.This example is only for version B of the evaluation board STEVAL$SMARTAG2B
    3. a sample application (FirmwareUpdate) used to update the firmware via NFC using the ST fast transfer protocol
    4. a bootloader example (SimpleBootLoader) that must be used with the firmware update application to enable the capability of updating the current firmware running on the STEVAL-SMARTAG2

1.4 APIs

Detailed user-API technical information with full function and parameter descriptions is available in a compiled HTML file in the package “Documentation” folder.

1.5 Sample application description

The sample applications available for the STEVAL-SMARTAG2 evaluation board are:

- OneShot (only for version B of the evaluation board - STEVAL$SMARTAG2B)
- SmarTag2
- SimpleBootLoader
- FirmwareUpdate

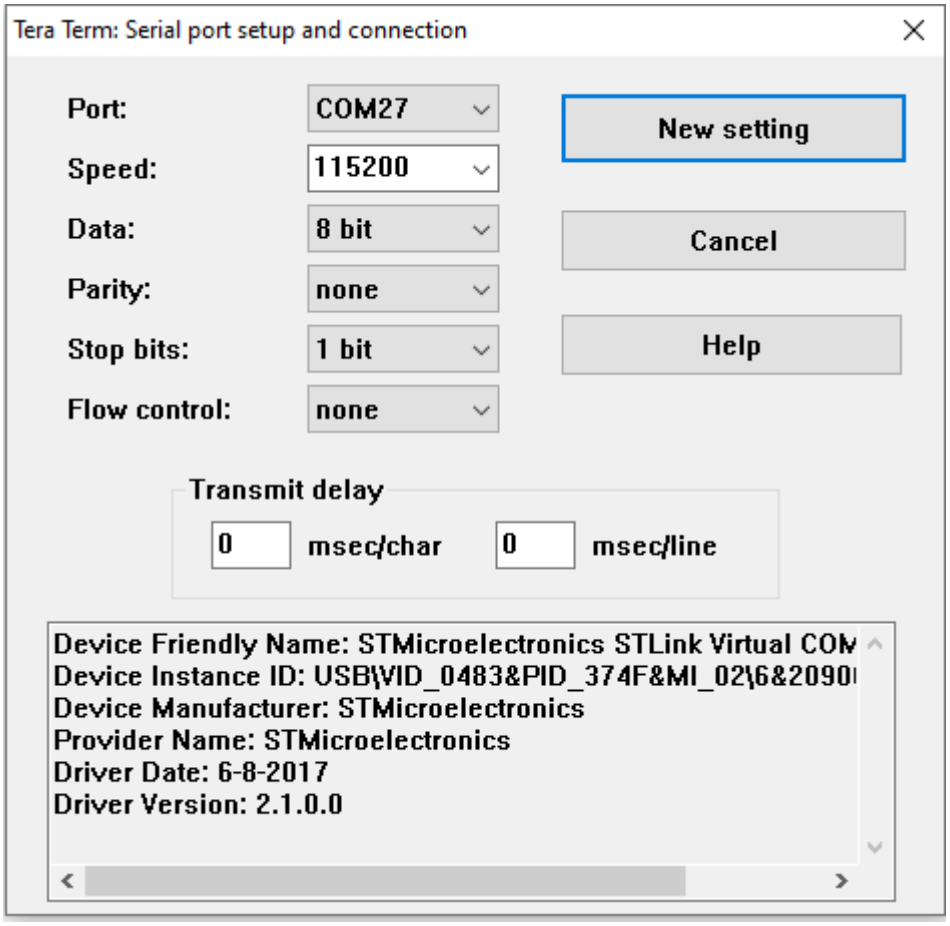
Ready-to-build projects are available for multiple IDEs.

1.5.1 OneShot application description

Note: This example can only be used for the version B of the evaluation board (STEVAL$SMARTAG2B).

You can set up a terminal window for the appropriate UART communication port in order to control the firmware phase.

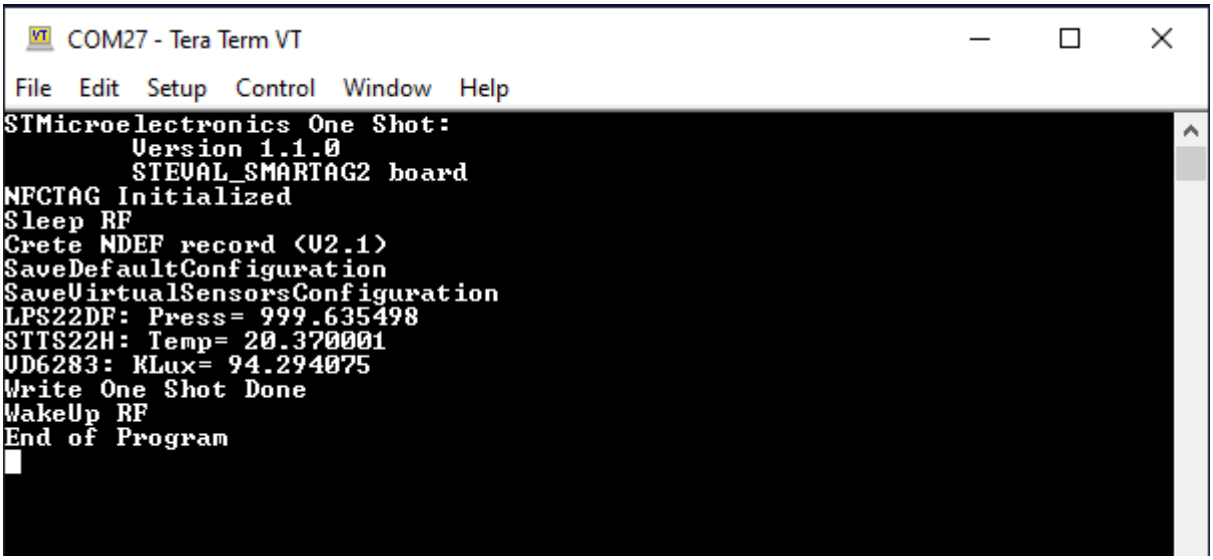
Figure 3. Terminal setting



If the STEVAL-SMARTAG2 is supplied using an NFC enabled reader, such as a mobile phone or tablet, the OneShot firmware starts.

After the initialization phase, the values from the environmental sensors (temperature, pressure, and ambient light) are read and wrote onto NDEF EEPROM.

Figure 4. OneShot terminal

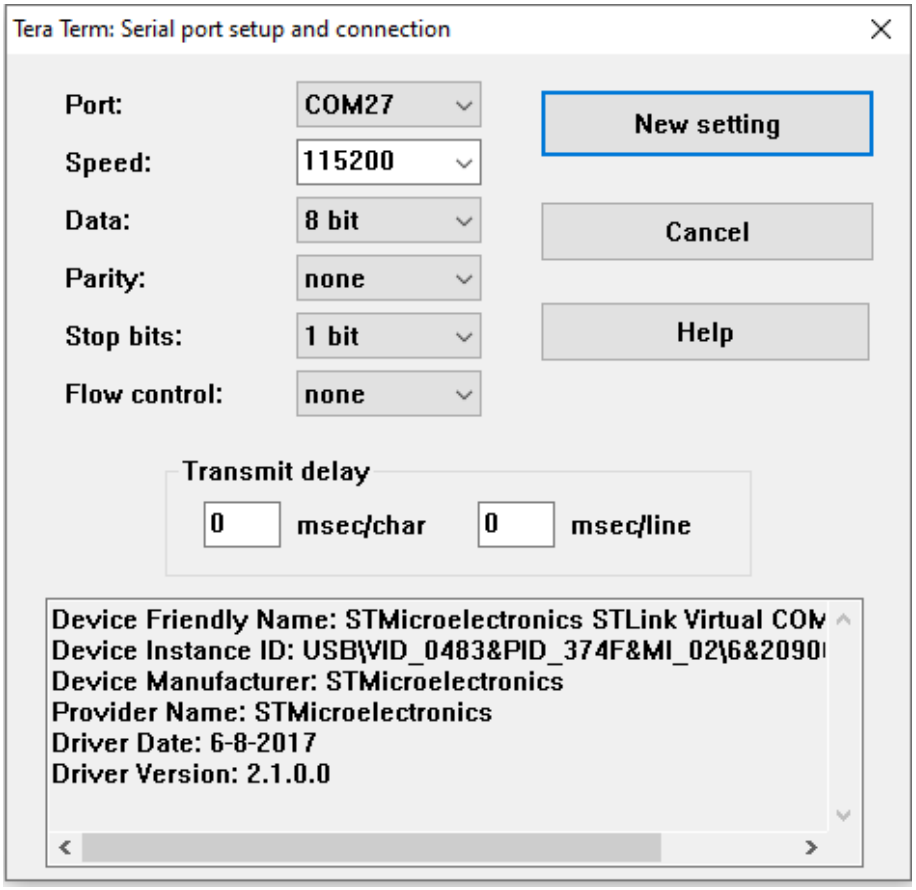


Through a suitable AndroidST or iOST application, such as STAssetTracking or STNFCSensor, these environmental sensors data can be read and show onto a mobile phone or tablet.

1.5.2 SmarTag2 application description

You can set up a terminal window for the appropriate UART communication port in order to control the initialization phase, as shown in the figure below.

Figure 5. **FP-SNS-SMARTAG2 terminal setting**



Note: To enable/disable this UART functionality on the STEVAL-SMARTAG2 board, you must recompile the code by uncommenting/commenting the line:

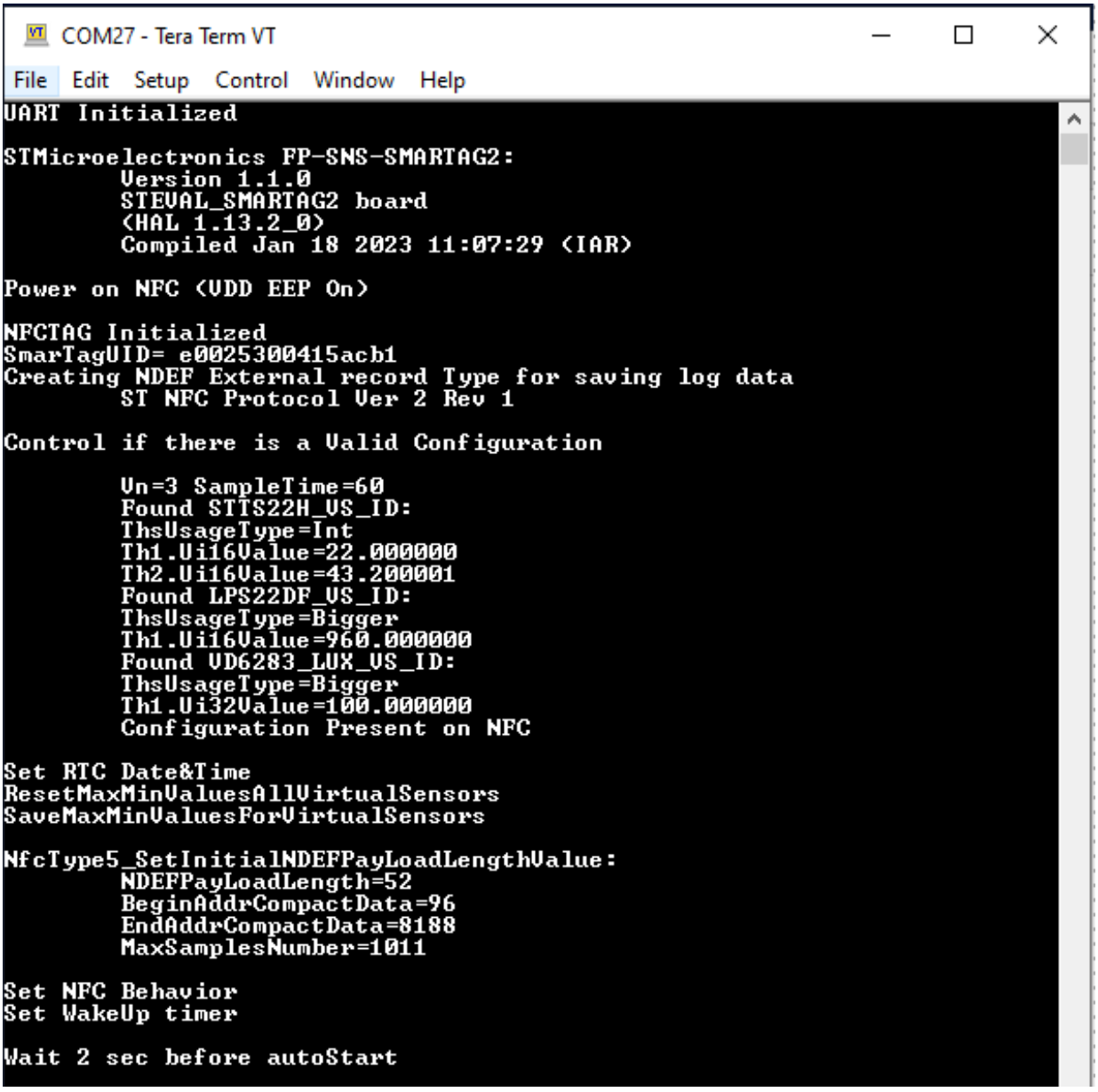
`#define SMARTAG2_ENABLE_PRINTF`

in the Projects\ STM32L4P5CE-SmarTag2\Examples\SmarTag2\Inc\SMARTAG2_config.h file.

When you first press the reset button, the application:

- starts initializing the UART and I²C interfaces
- shows the SmarTag UID
- reads the last configuration written on to NFC tag (if available)
- sets the NFC behavior
- sets the wakeup timer

Figure 6. **FP-SNS-SMARTAG2 UART output initialization**

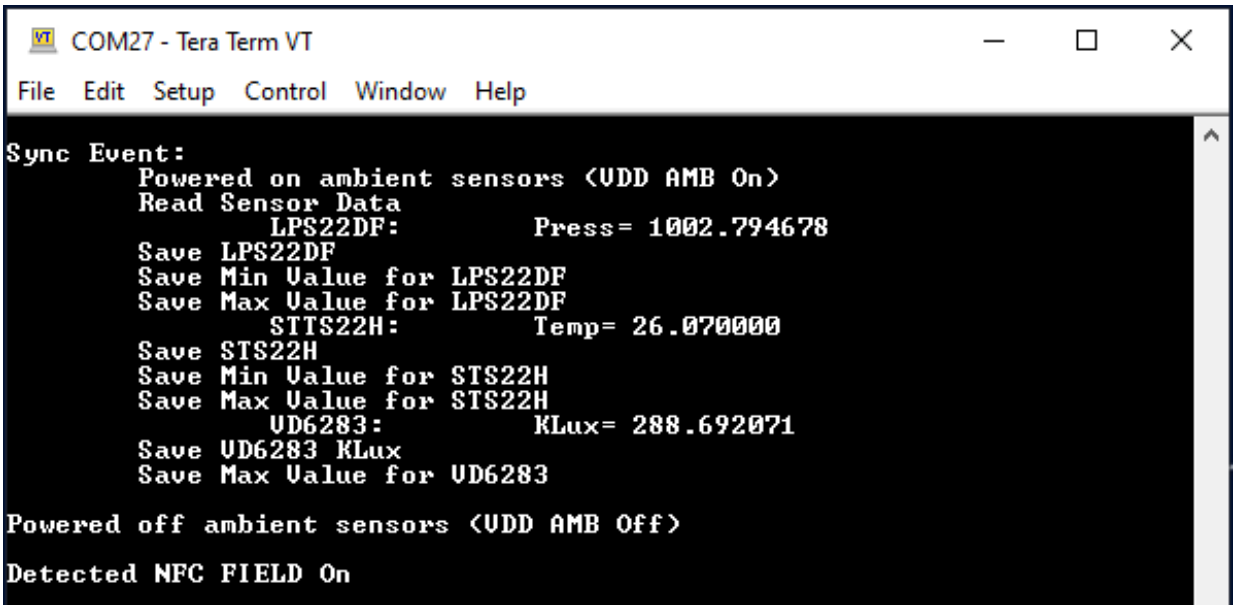


After the auto-start range time, the samples are logged using the written configuration on the NFC tag (or the default one, if not available).

Figure 7. **FP-SNS-SMARTAG2 UART output auto-start**

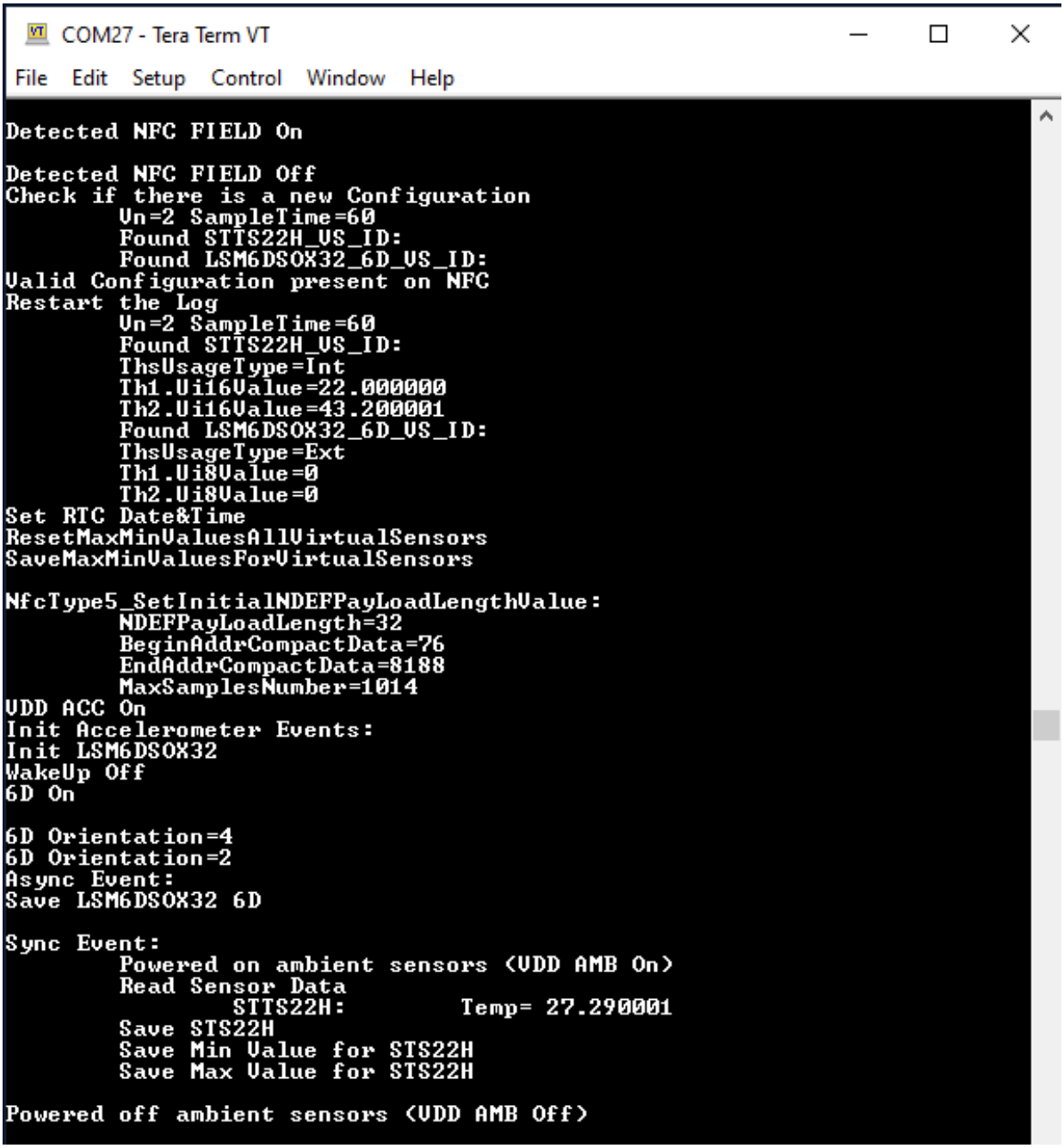
When the smartphone is close to the NFC tag, the message "Detected NFC FIELD On" appears.

Figure 8. FP-SNS-SMARTAG2 UART output NFC on



When the smartphone is kept distant from the NFC tag, the message "Detected NFC FIELD Off" appears together with the new configuration if a new one is detected.

Figure 9. **FP-SNS-SMARTAG2 UART output NFC off**



1.5.2.1 *SmarTag2: Android and iOS sample client application*

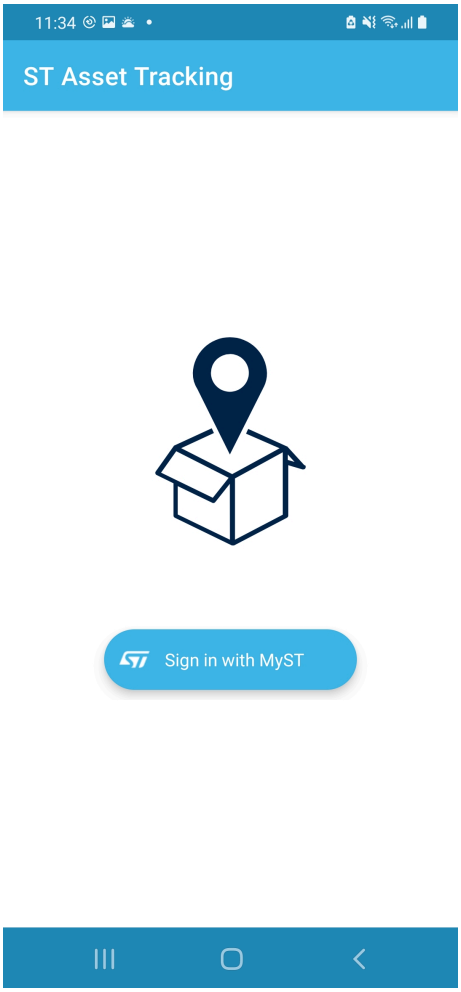
The SmarTag2 application is compatible with the STAssetTracking Android (version 3.1.0 or higher)/iOS (version 3.1.0 or higher) application and with the STNFCSensor Android/iOS application (version 1.4.0 or higher) available at their respective GooglePlay/App store.

The next sections show some use cases enabled by the Android application.

1.5.2.1.1 Sign-in with MyST

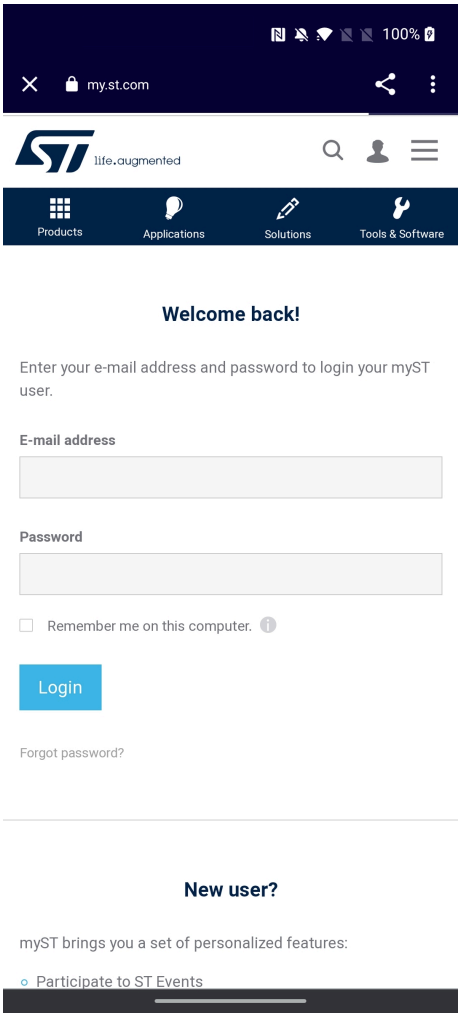
If you are not signed-in or you are not registered on MyST, when you open the STAssetTracking application, the following page appears.

Figure 10. **Sign-in with MyST (1 of 2)**



Click on "Sign-in with MyST". Then, insert your e-mail address and password to execute the login if you are already registered. Otherwise, click "New User" and follow the instructions.

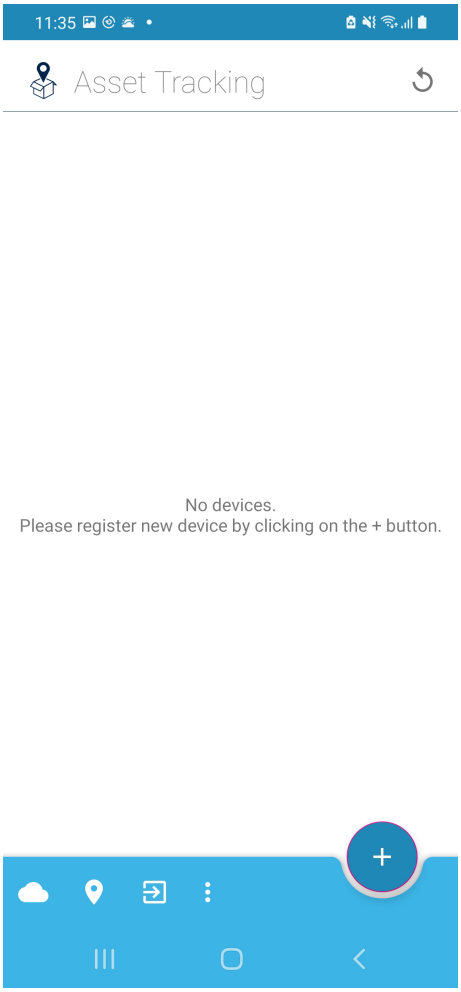
Figure 11. **Sign-in with MyST (2 of 2)**



1.5.2.1.2 How to register a new device

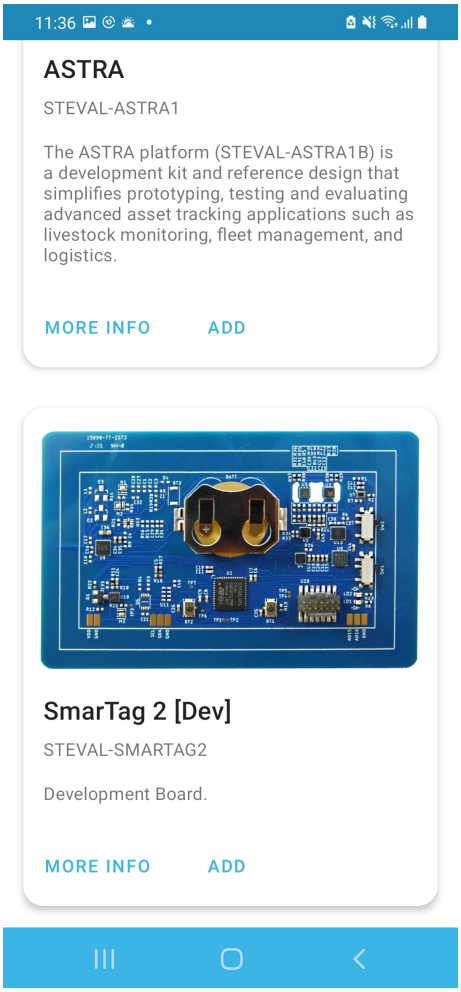
After signing-in to MyST, when reopening, the STAssetTracking application shows the following page.

Figure 12. STAssetTracking after signing-in to MyST



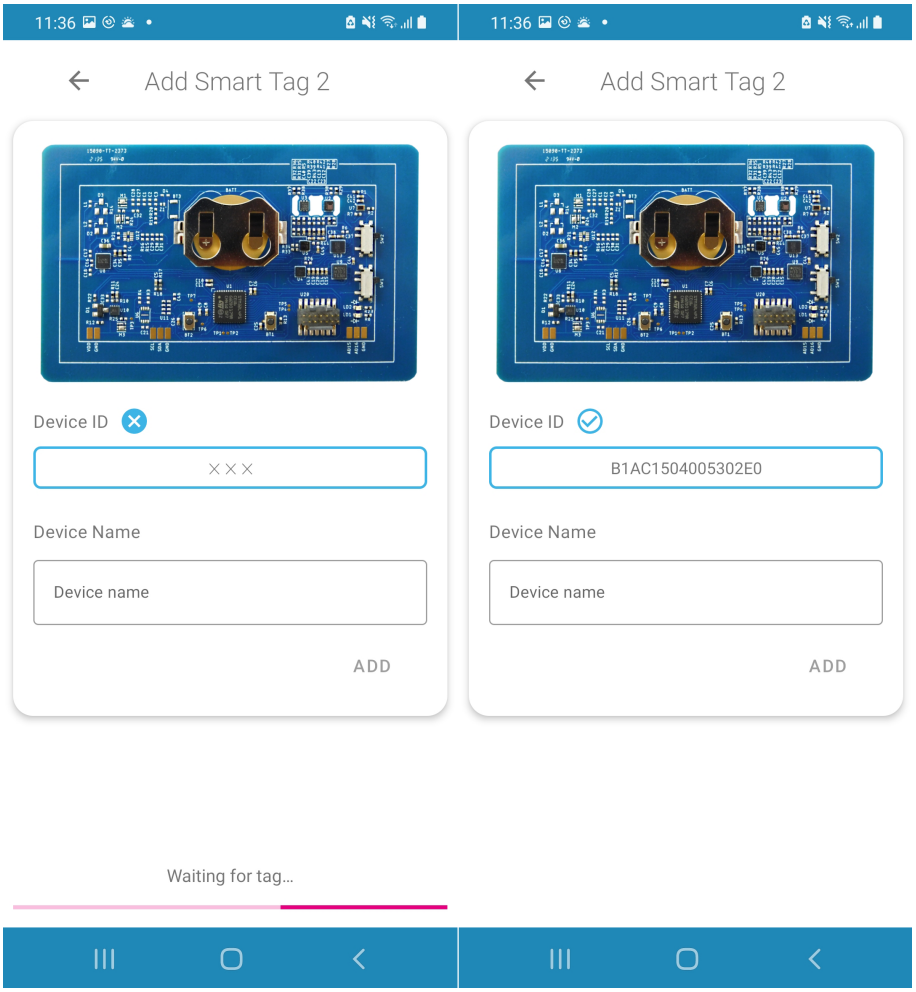
Click on the "+" button to register a new device and scroll down to find the Smartag2 device.

Figure 13. **STAssetTracking - registering a new device (1 of 3)**



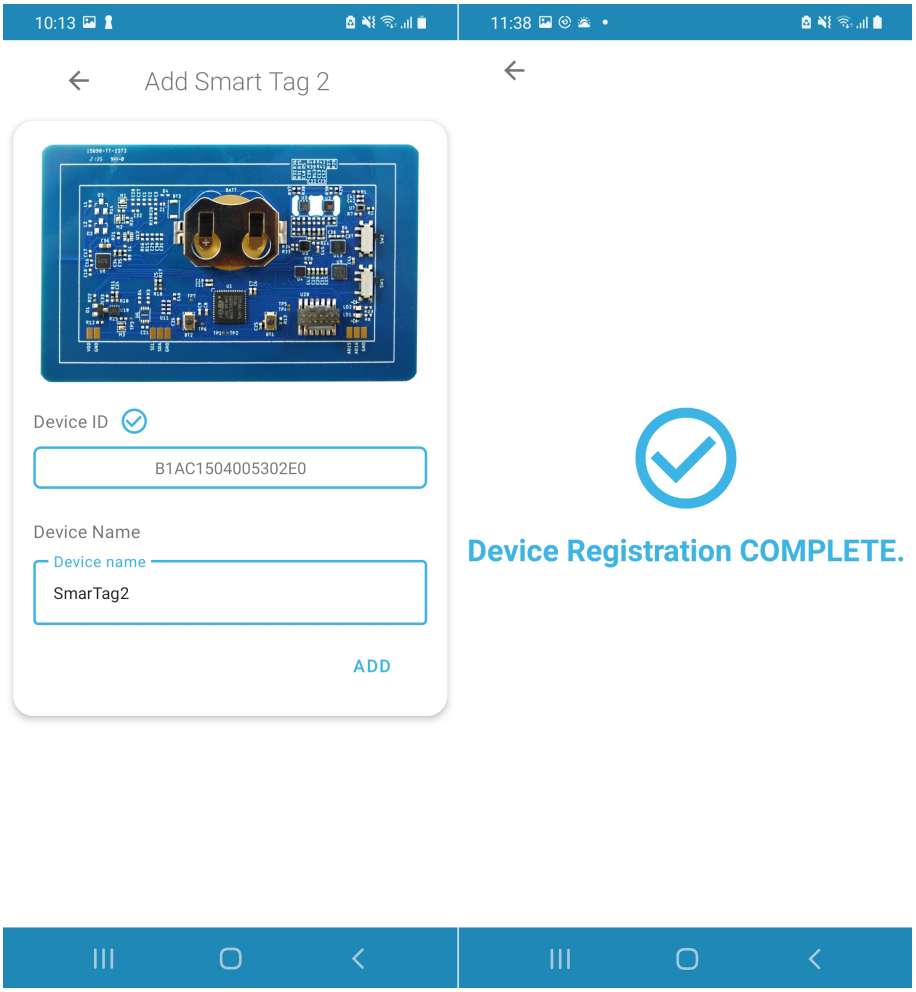
Click on the add button and put the mobile phone on top the device to detect the tag by NFC. When the tag is detected, the device ID is shown:

Figure 14. **STAssetTracking - registering a new device (2 of 3)**



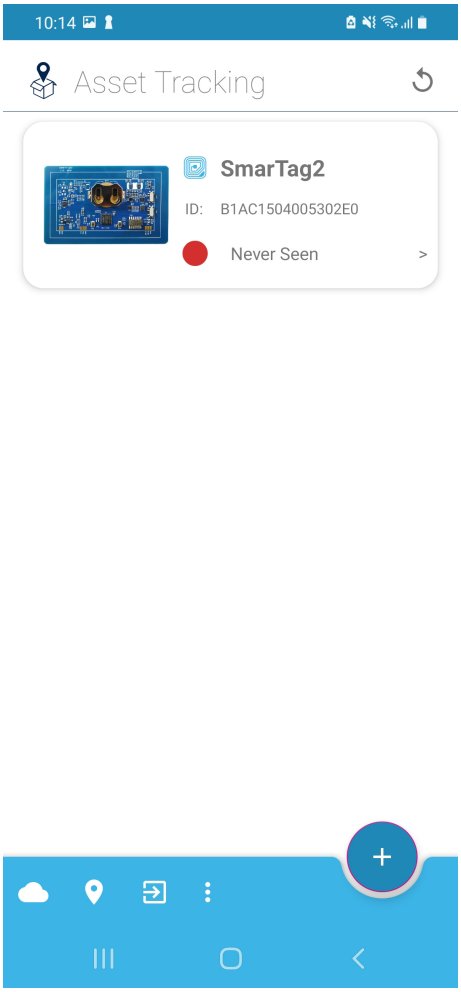
Insert the device name and click on the add button to complete the process.

Figure 15. **STAssetTracking - registering a new device (3 of 3)**



After the device is registered, the STAssetTracking shows it in the device list.

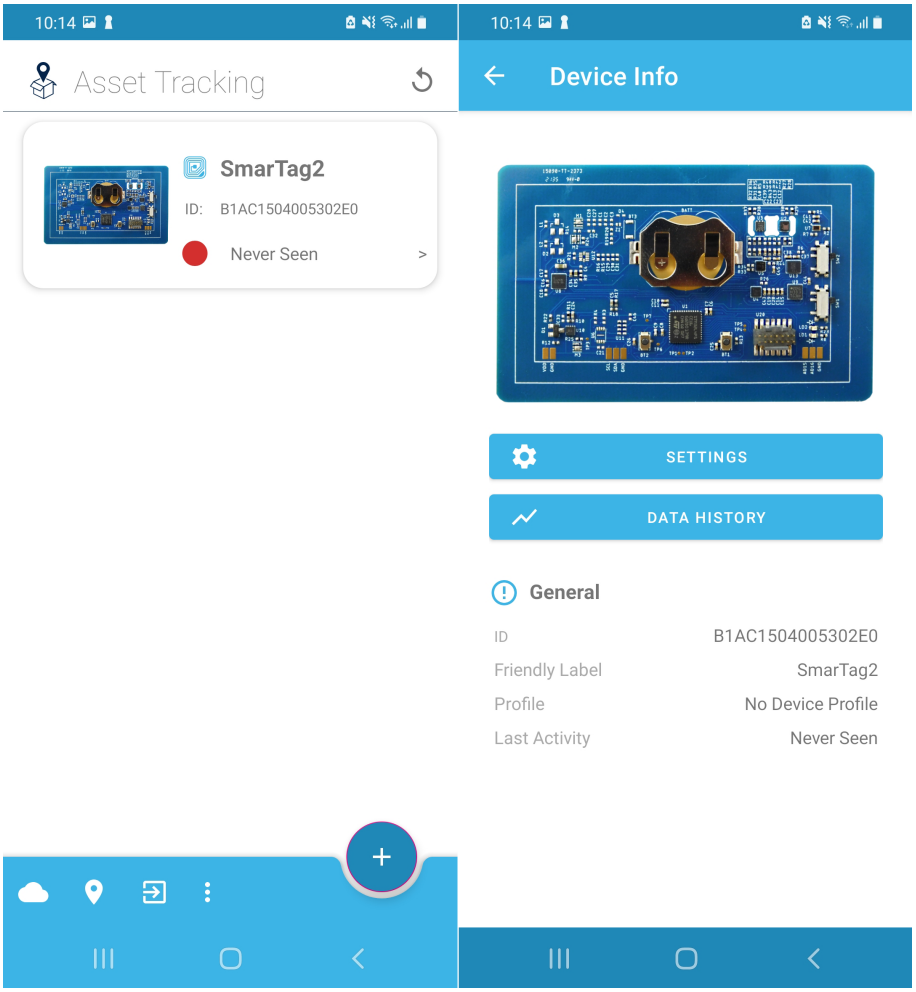
Figure 16. **STAssetTracking - device list**



1.5.2.1.3 Settings

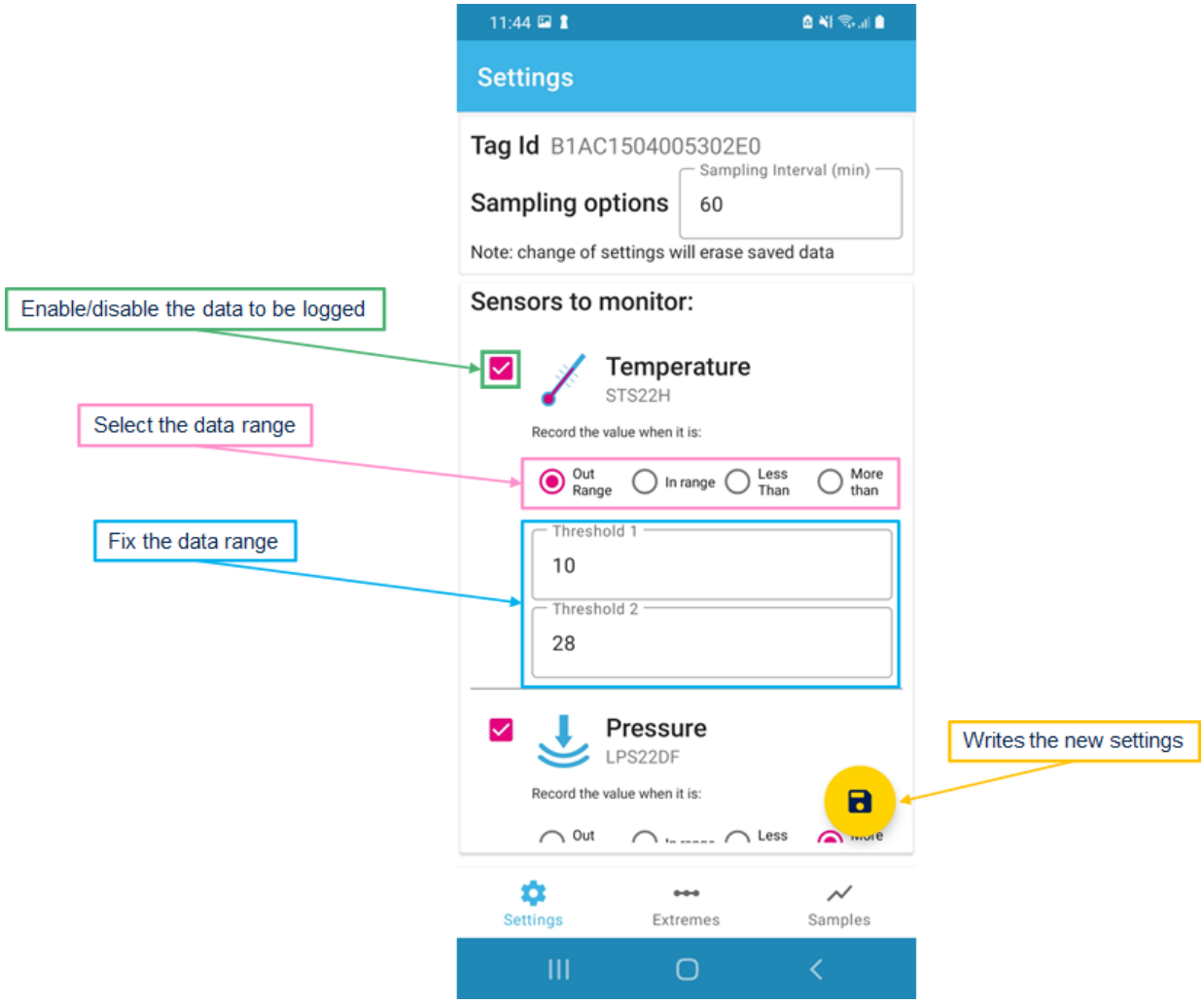
Start the STAssetTracking application and select your device from the list.

Figure 17. **STAssetTracking - device info**



Select settings and put the mobile phone on top the device to detect the tag by NFC. When the tag is detected, the application shows the following page.

Figure 18. **STAssetTracking - settings**



From this page, you can select the data to be logged (temperature, pressure, luminosity, IMU accelerometer, 6-axis orientation, and tilt) with the related data range as well as the time interval.

1.5.2.1.4 Extremes

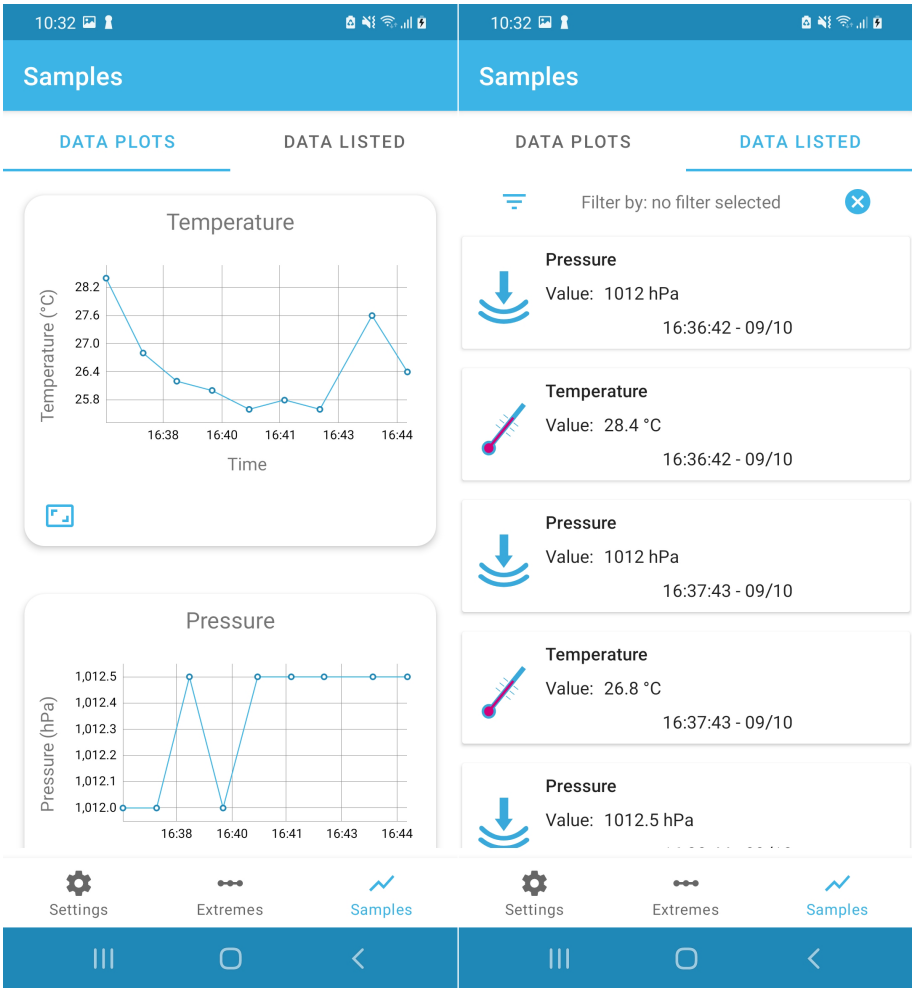
This page shows the maximum and minimum values obtained during the logging of the selected data.

Figure 19. **STAssetTracking - Extremes**

1.5.2.1.5 Samples

This page collects all the data logging for the selected data with the related data range.

Figure 20. **STAssetTracking - samples**



1.5.3 FirmwareUpdate and SimpleBootLoader applications description

The FirmwareUpdate application must work with the SimpleBootLoader example to enable the firmware update capability.

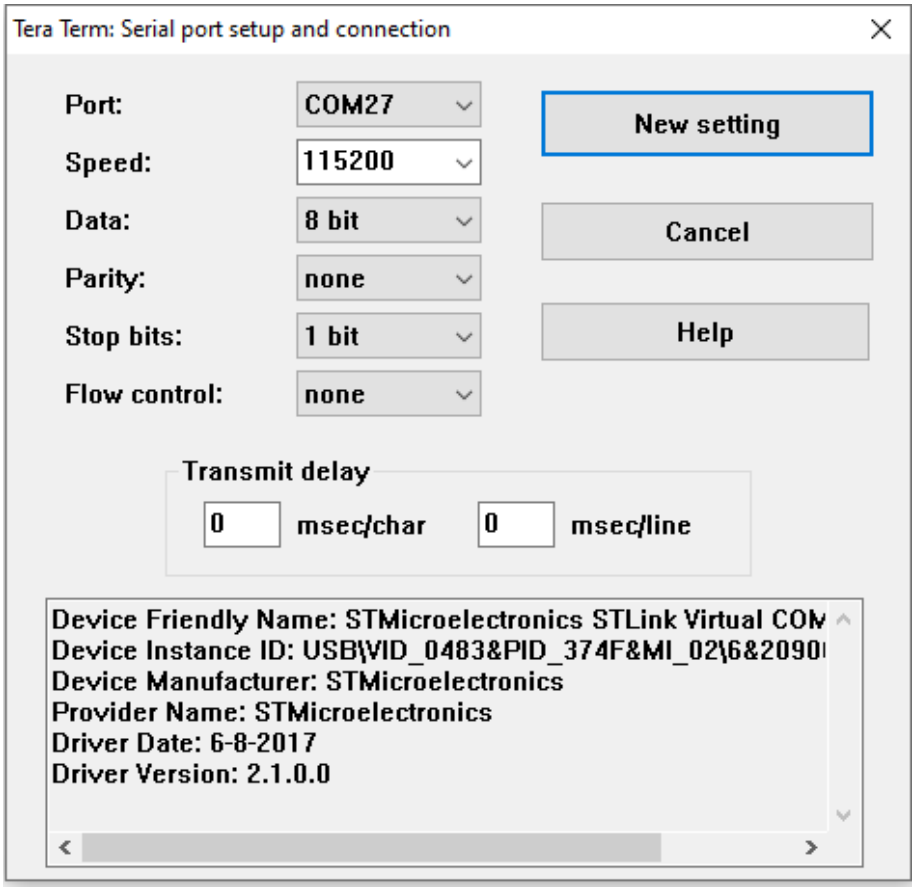
The FirmwareUpdate receives the new firmware via NFC using the ST FTM protocol and saves it on the right flash memory region.

The SimpleBootLoader replaces the FirmwareUpdate firmware currently running with the new version received via NFC.

1.5.3.1 Firmware update

You can set up a terminal window for the appropriate UART communication port in order to control the initialization phase, as shown in the figure below.

Figure 21. **FP-SNS-SMARTAG2 terminal setting**

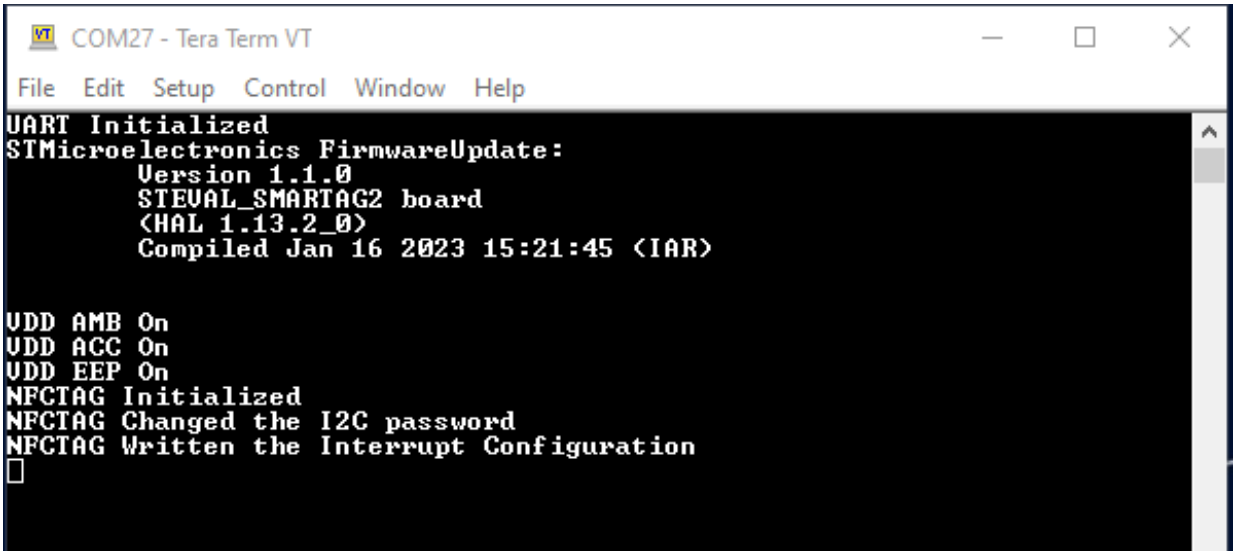


Note: To enable/disable this UART functionality on the STEVAL-SMARTAG2 board, you must recompile the code by uncommenting/commenting the line: `#define SMARTAG2_ENABLE_PRINTF` in the Projects\ STM32L4P5CE-SmarTag2\Applications\ FirmwareUpdate\Inc\Firmware_conf.h file.

When you first press the reset button, the application:

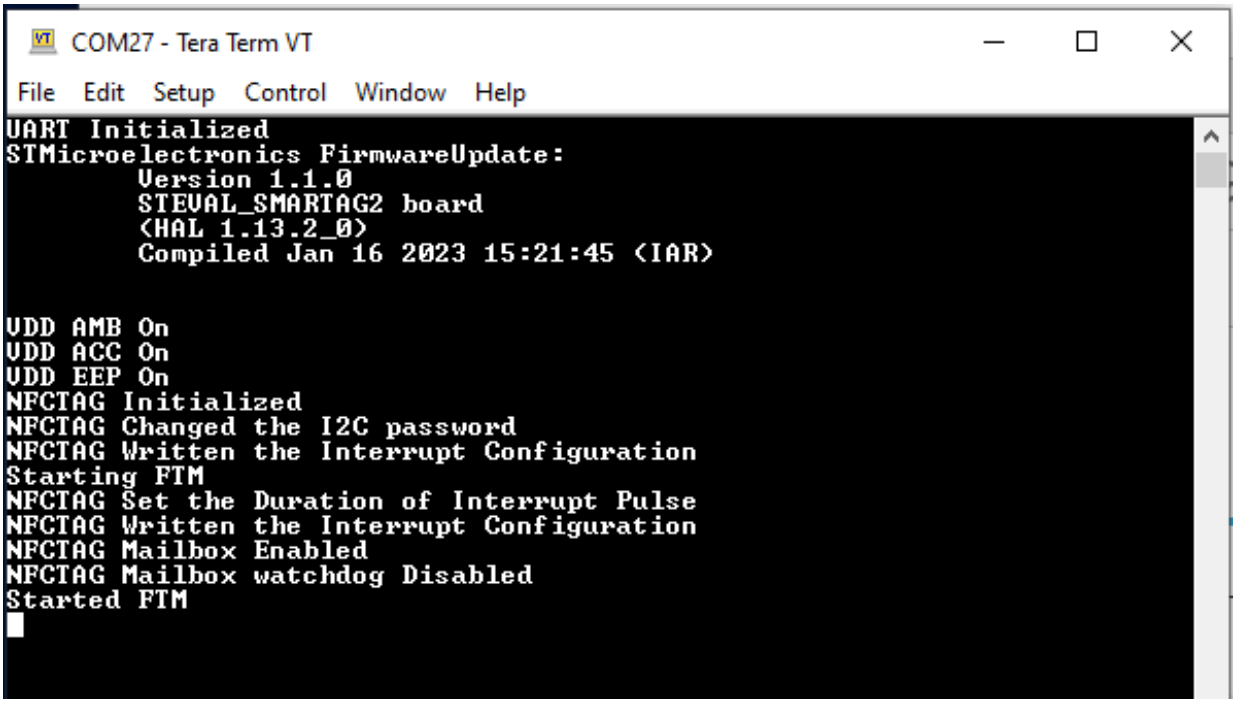
- initializes the UART
- initializes the NFC tag
- changes the I²C password on the NFC tag
- writes the interrupt configuration

Figure 22. **FP-SNS-SMARTAG2 UART output initialization**



After pressing the user button, the fast transfer mode (FTM) starts.

Figure 23. **FP-SNS-SMARTAG2 - FTM**



After the firmware update, the board restarts. For further details on how to execute the firmware update, see Section 1.5.3.5 FirmwareUpdate: Android and iOS sample client application.

Figure 24. **FP-SNS-SMARTAG2 - board restart (1 of 3)**

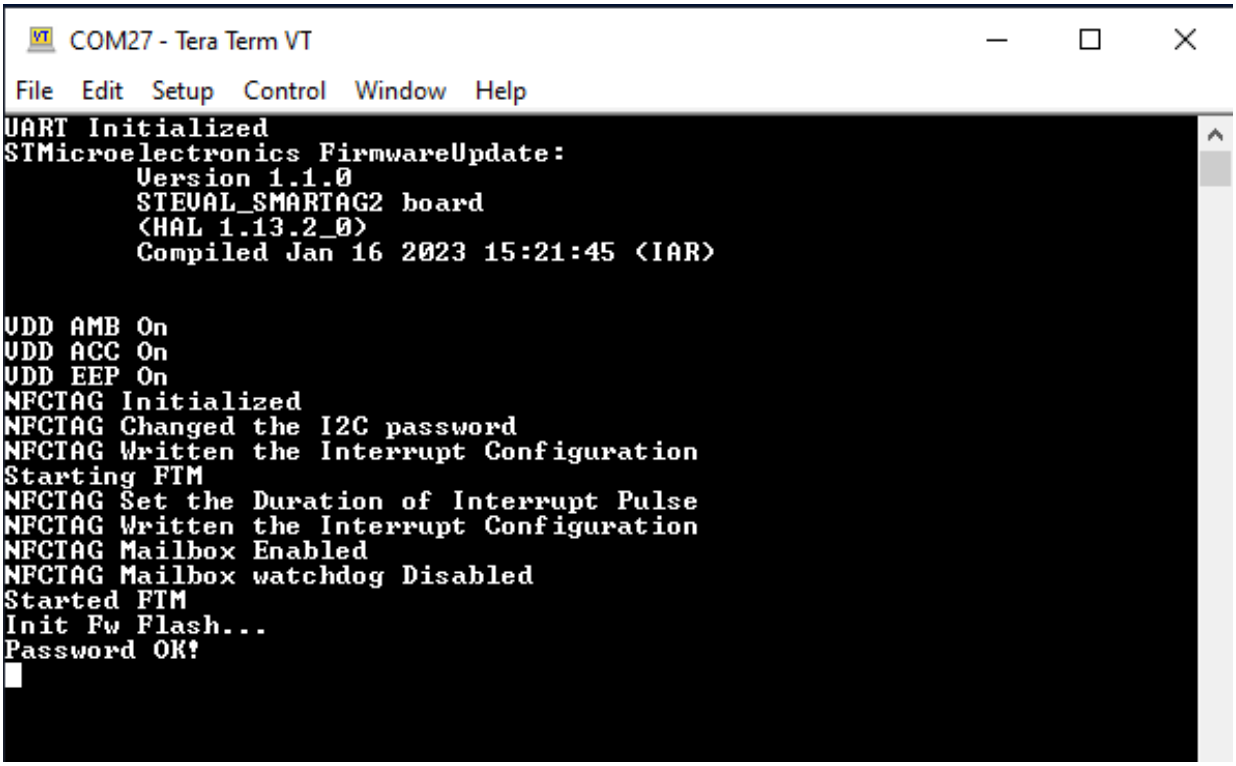


Figure 25. **FP-SNS-SMARTAG2 - board restart (2 of 3)**

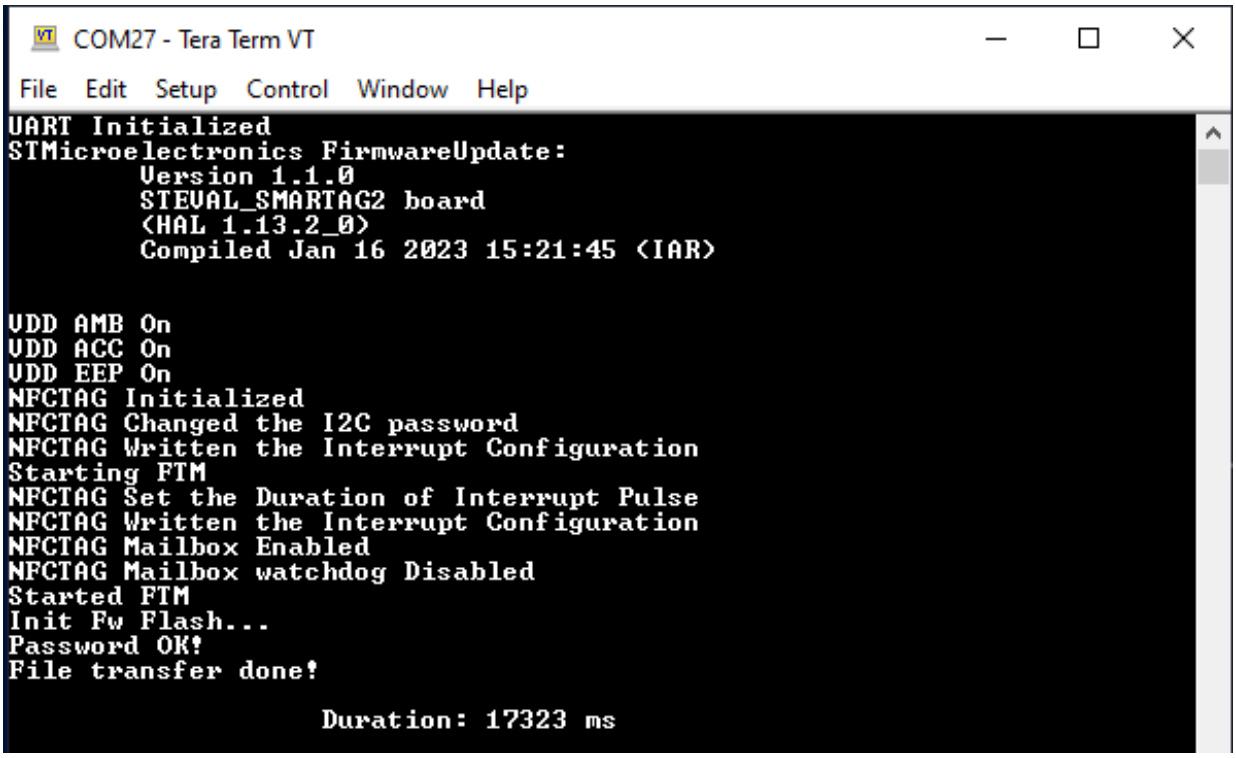
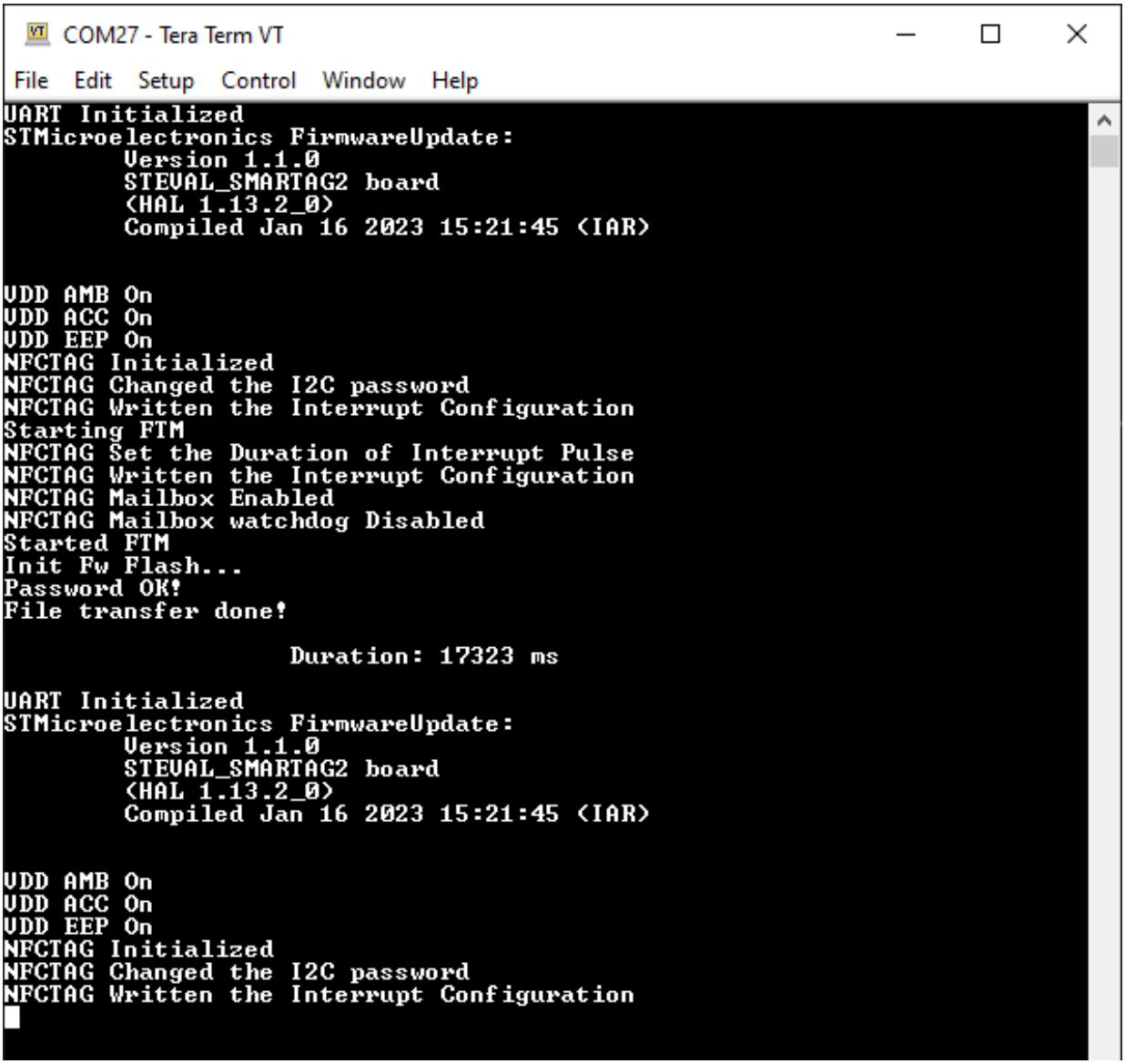


Figure 26. **FP-SNS-SMARTAG2 - board restart (3 of 3)**



1.5.3.2 *The installation process*

The package binary directory contains:

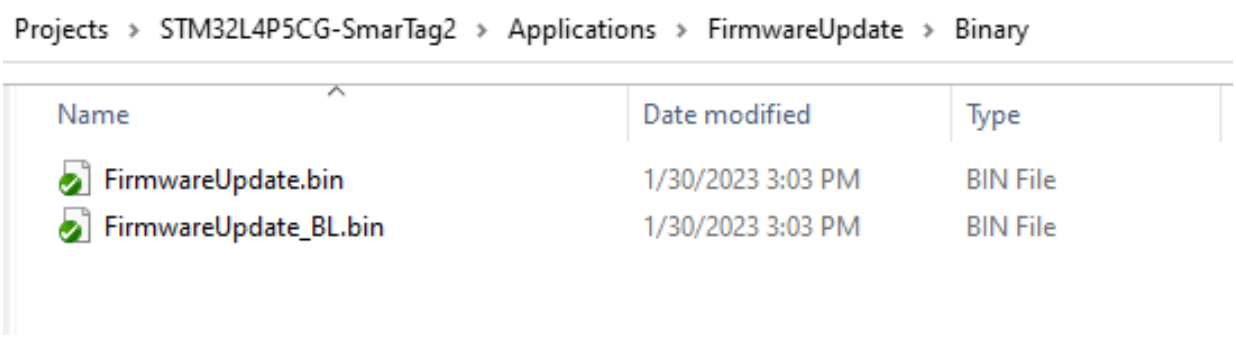
- precompiled firmware to be flashed with the STM32CubeProgrammer to the correct memory address (0x08002000)

Note: This precompiled binary is compatible with the firmware update procedure.

- precompiled firmware plus the bootloader to be directly flashed with the STM32CubeProgrammer

Note: This precompiled binary is not compatible with the firmware update procedure.

Figure 27. **Binary folder**



A batch script is provided to simplify the operation described before by saving the firmware and the bootloader to the right position. It is available for each IDE (IAR/RealView/STM32CubeIDE).

This script:

1. performs a full flash memory erase to start from a clean system
2. flashes the bootloader to the correct position (0x08000000)
3. flashes the firmware to the correct position (0x08002000)

The script also dumps an image that contains the bootloader and the firmware. This image file can be directly flashed to the beginning of the flash memory as the image provided in the binary folder.

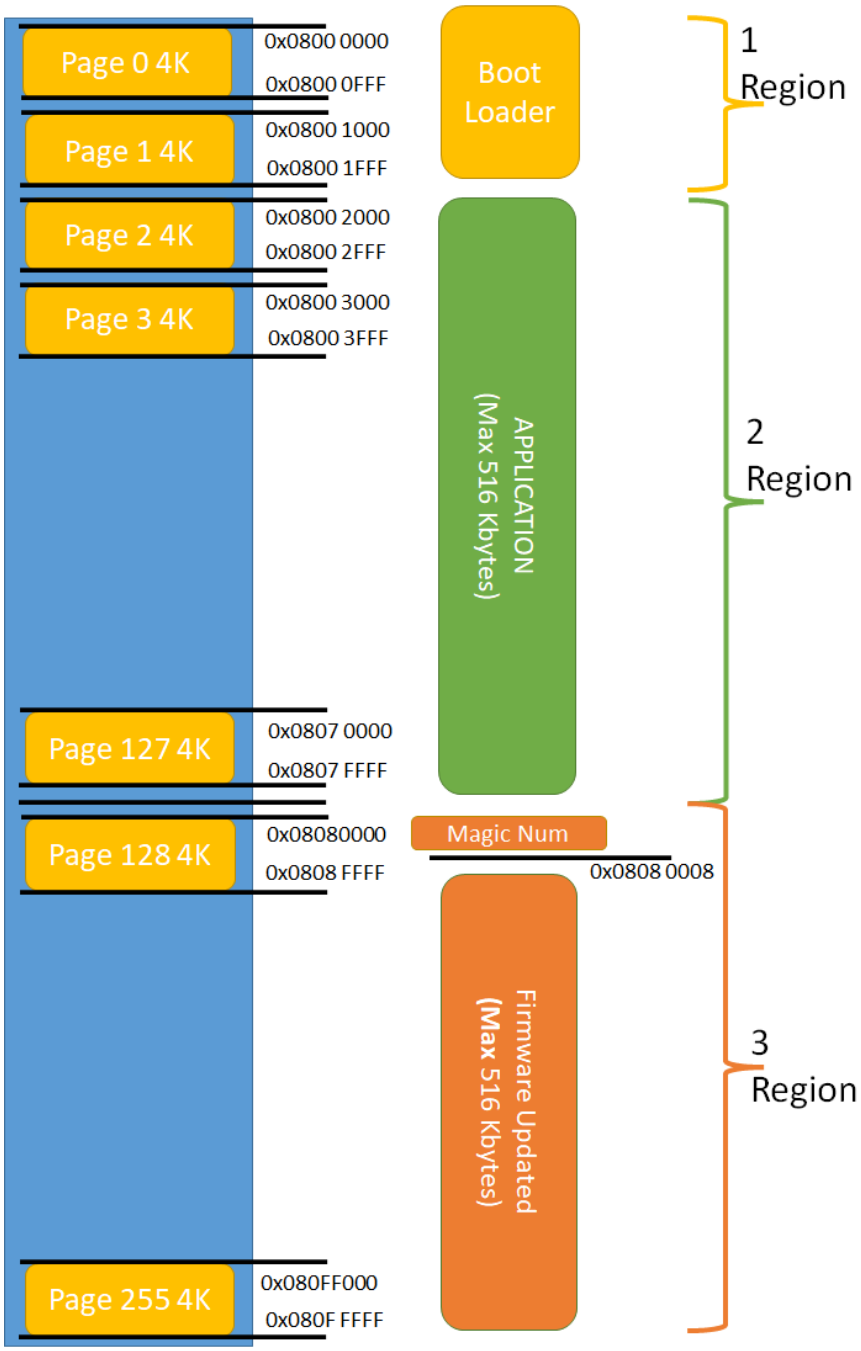
1.5.3.3 *Flash memory management*

To enable the firmware update, the whole flash management is split into:

1. a region that contains a custom bootloader
2. a region that contains the FirmwareUpdate application
3. a region used to store the firmware before the update

For further info on the flash memory configuration, see RM0432: "STM32L4+ Series advanced Arm®-based 32-bit MCUs".

Figure 28. **Flash memory management**



1.5.3.4 *The boot process*

The FirmwareUpdate cannot be flashed to the beginning of the flash memory (address 0x08000000). Therefore, it is compiled to run from the beginning of the second flash memory region (address0x08002000).

To enable this functionality, we have set the vector table offset with respect to the default value by modifying the Src/system_stm32l4xx.c files using the `#defineVECT_TAB_OFFSET 0x2000` define.

We also have changed the linker script.

For example, the linker script used for running and compiled using IAR Embedded Workbench for ARM is:

```
/*###ICF### Section handled by ICF editor, don't touch! ****/
/*-Editor annotation file-*/
/* IcfEditorFile="$TOOLKIT_DIR$\config\ide\IcfEditor\cortex_v1_0.xml" */
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__ = 0x08002000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = 0x08002000;
define symbol __ICFEDIT_region_ROM_end__   = 0x0807FFFF;
define symbol __ICFEDIT_region_RAM_start__ = 0x20000000;
define symbol __ICFEDIT_region_RAM_end__   = 0x2004FFFF;

/*-Sizes-*/
define symbol __ICFEDIT_size_cstack__ = 0x8000;
define symbol __ICFEDIT_size_heap__   = 0x8000;
/**** End of ICF editor section. ###ICF###*/
```

Using the above linker script, the maximum usable code size is fixed at 516 Kbytes.

The bootloader firmware is available in the package. It is the SimpleBootLoader application.

At any board reset, the board starts executing the bootloader.

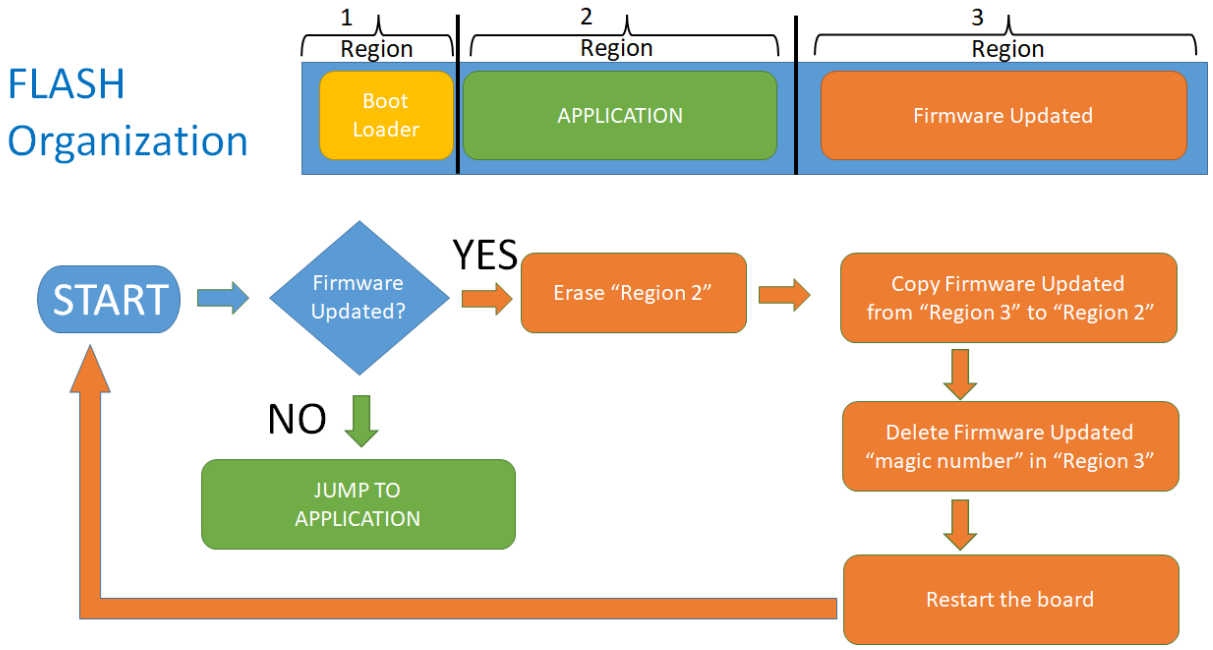
The boot loader checks whether a firmware update is available. The check is based on the presence of a "magic number" at the beginning of the third flash memory region (see Figure 28. Flash memory management).

If there is a firmware update available, the bootloader:

1. erases the second flash memory region (containing the FirmwareUpdate application)
2. replaces its content with the firmware update
3. erases the “magic number” used to check the firmware update presence
4. restarts the board

If there is no firmware update available, the bootloader jumps directly to the FirmwareUpdate firmware.

Figure 29. FirmwareUpdate application

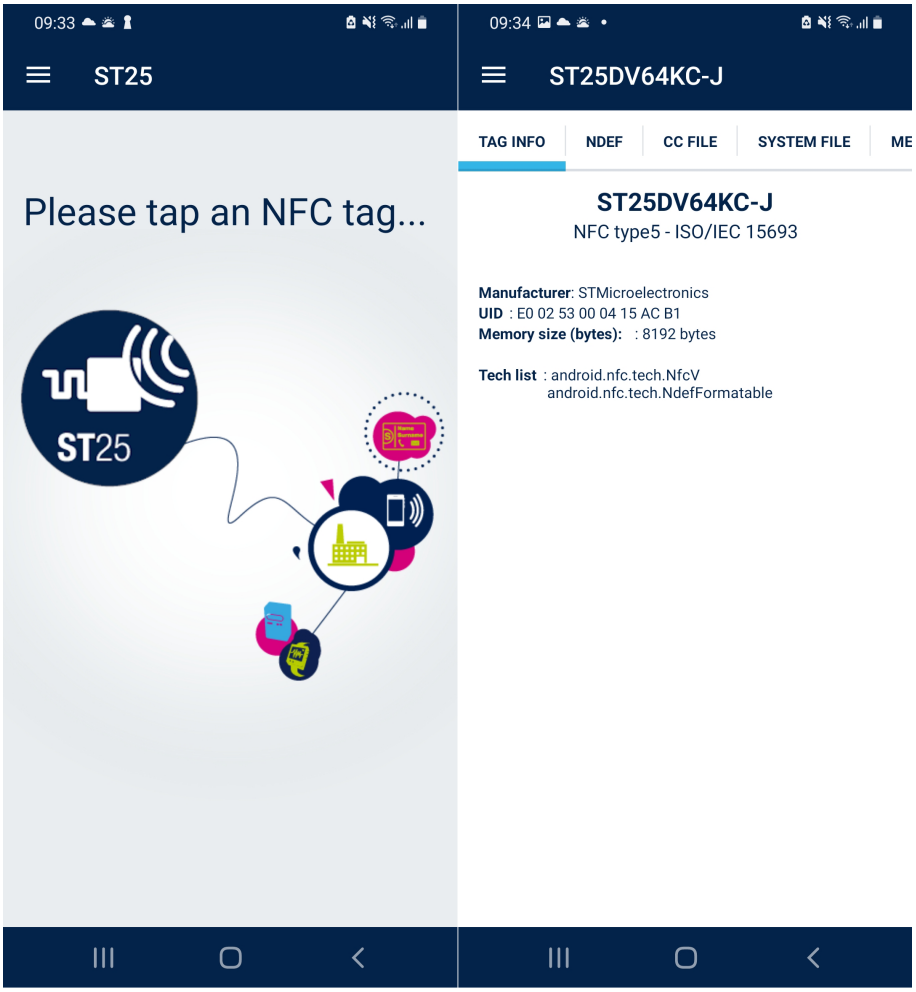


1.5.3.5 *FirmwareUpdate: Android and iOS sample client application*

Start the ST25 NFC tag mobile application and put your mobile phone on the board.

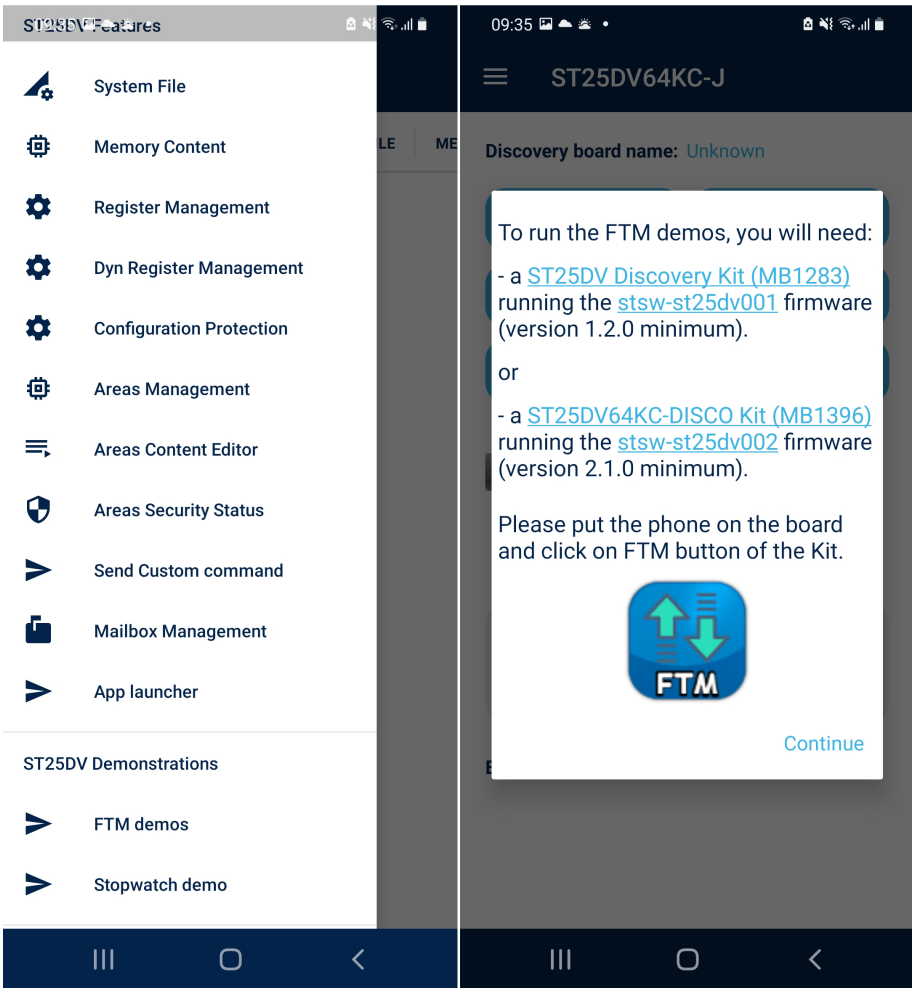
Select the menu option.

Figure 30. **ST25 NFC tag app - tag info**



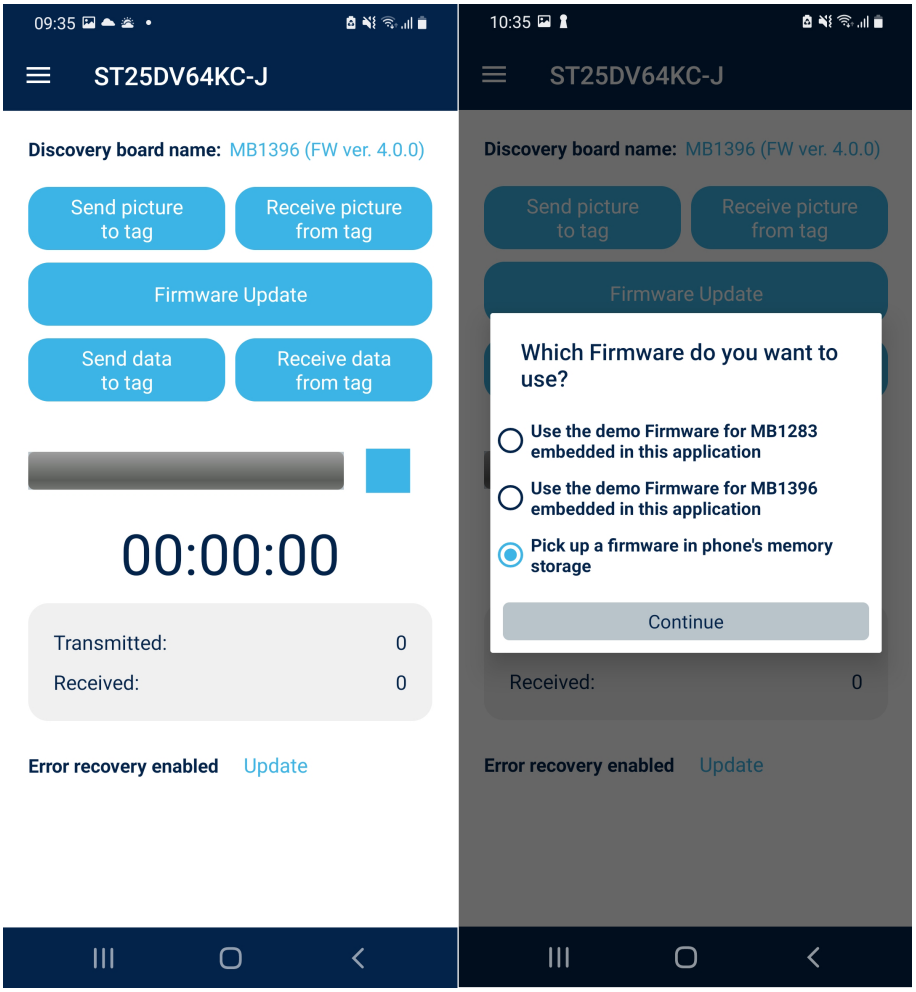
Select "FTM Demos" and click "continue".

Figure 31. **ST25 NFC tag app - FTM demos**



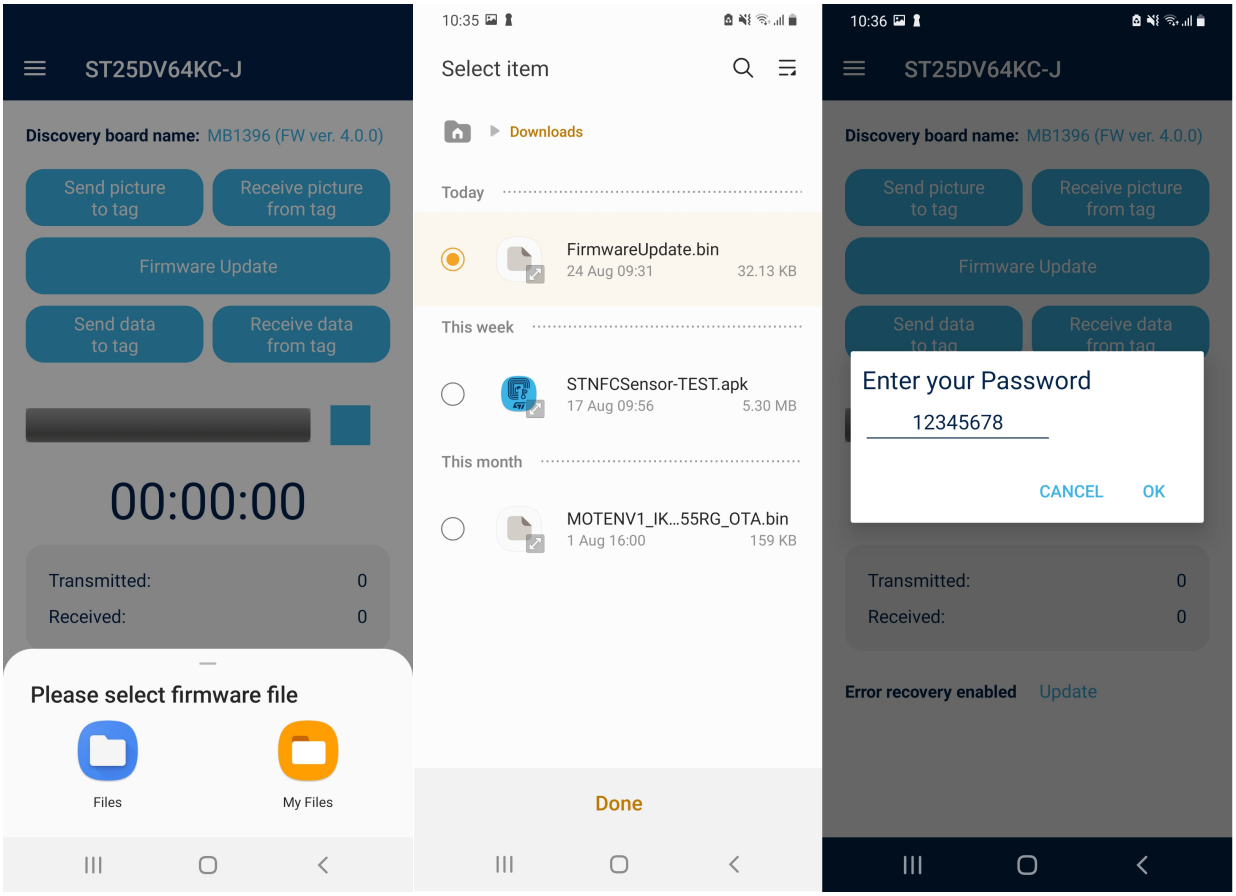
Select "Firmware Update", choose “Pick up a firmware in phone’s memory storage" and click "continue".

Figure 32. **ST25 NFC tag app - firmware update**



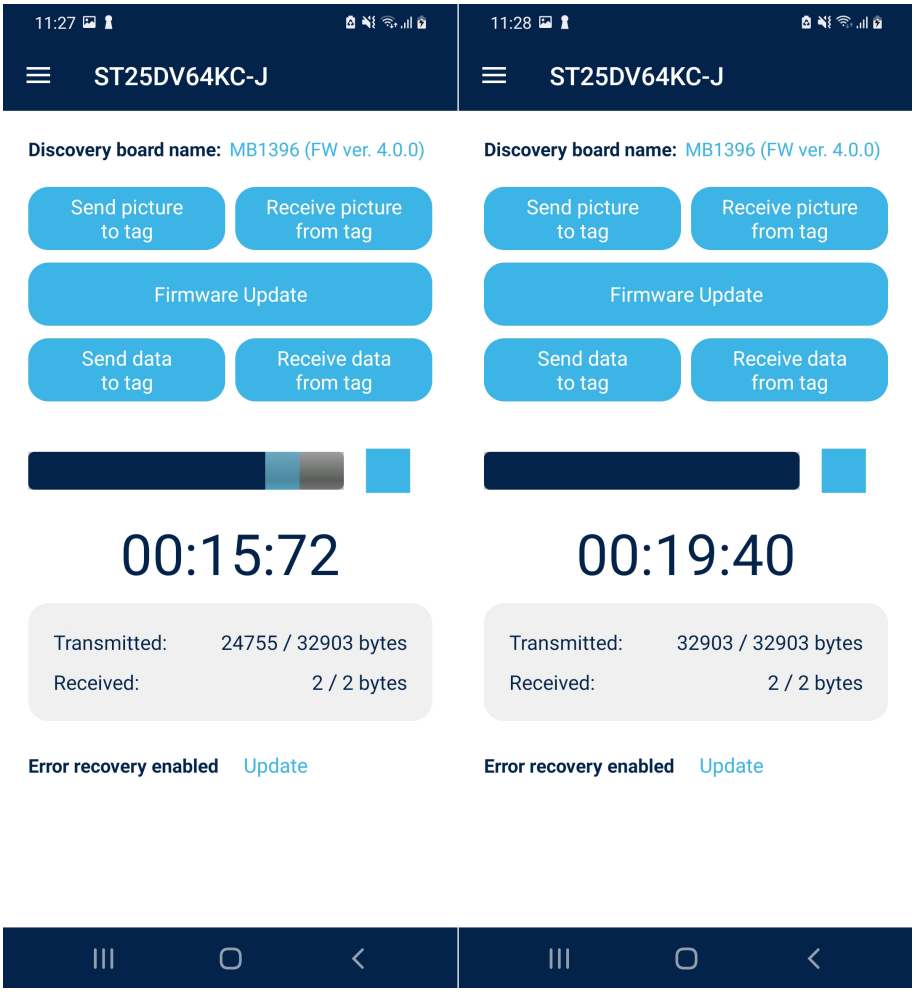
Select the firmware file and enter the password (12345678 is the default one).

Figure 33. **ST25 NFC tag app - password entering**



Wait for the firmware update to finish.

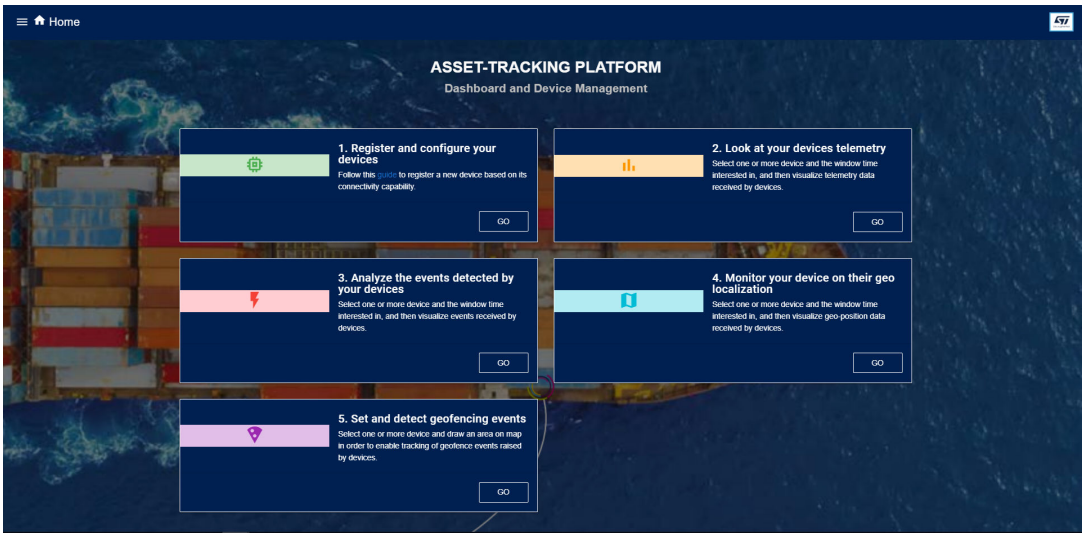
Figure 34. **ST25 NFC tag app - firmware update complete**



2 Using the Asset Tracking web dashboard

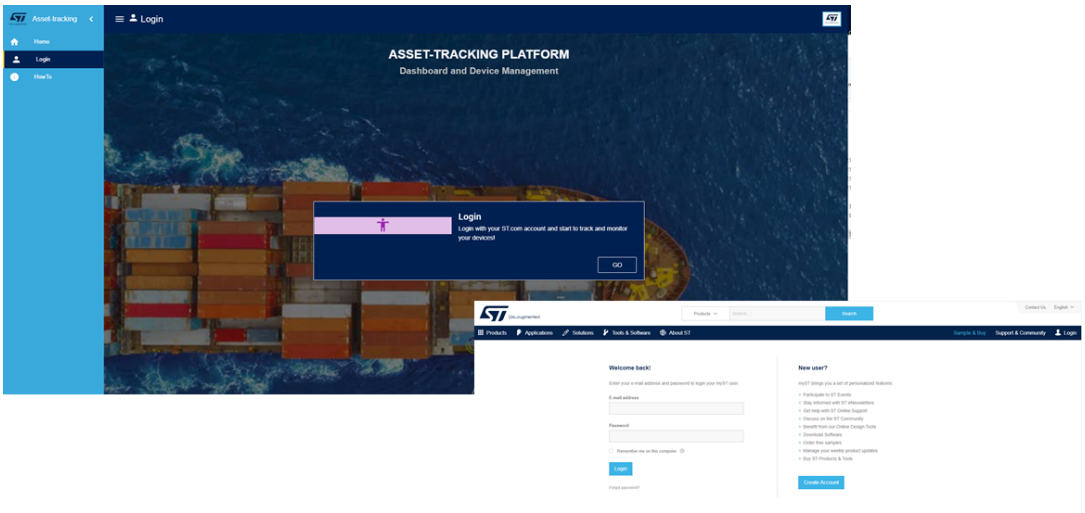
Step 1. Go to DSH asset tracking web dashoboard.

Figure 35. DSH-ASSETRACKING dashboard homepage



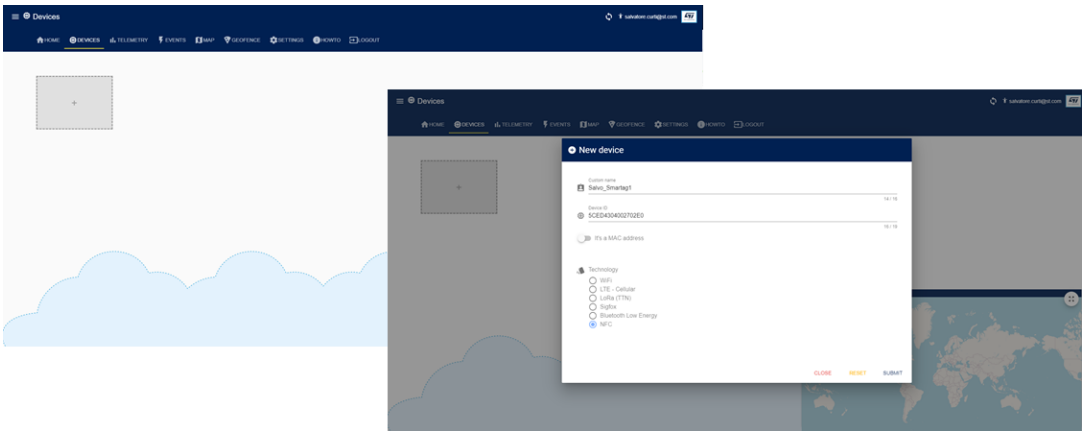
Step 2. Select login, click [**GO**] button and provide your username and password.

Figure 36. DSH-ASSETRACKING dashboard - login



Step 3. Add the new device.

Figure 37. DSH-ASSETRACKING dashboard - adding device



Step 4. Select the device you want to monitor.

Figure 38. DSH-ASSETRACKING dashboard - selecting the device

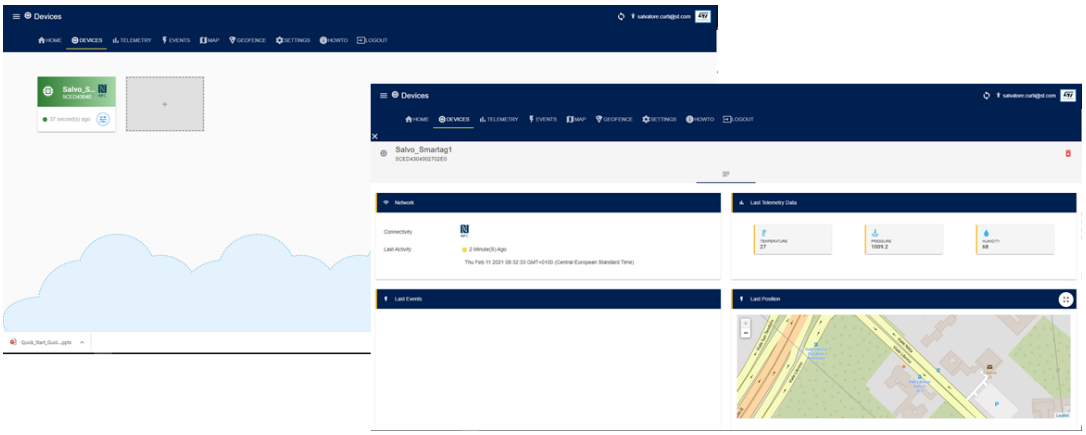


Figure 39. DSH-ASSETRACKING dashboard - selecting telemetry

To upload data on the cloud, you can also register the device from the ST Asset Tracking application.

Figure 40. Uploading data the cloud through ST Asset Tracking application

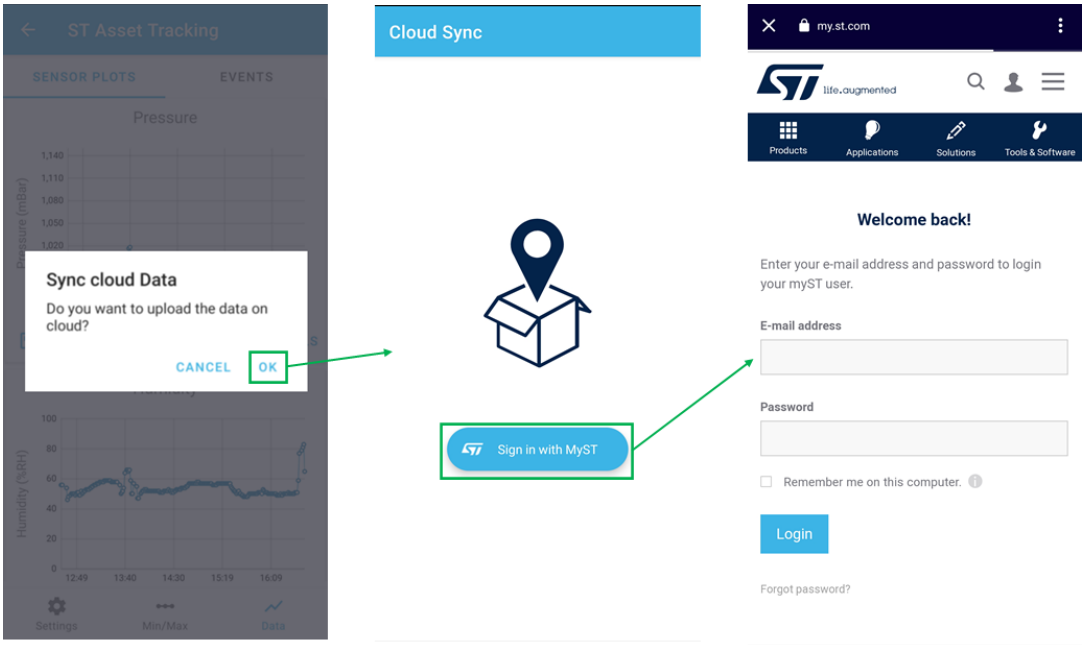
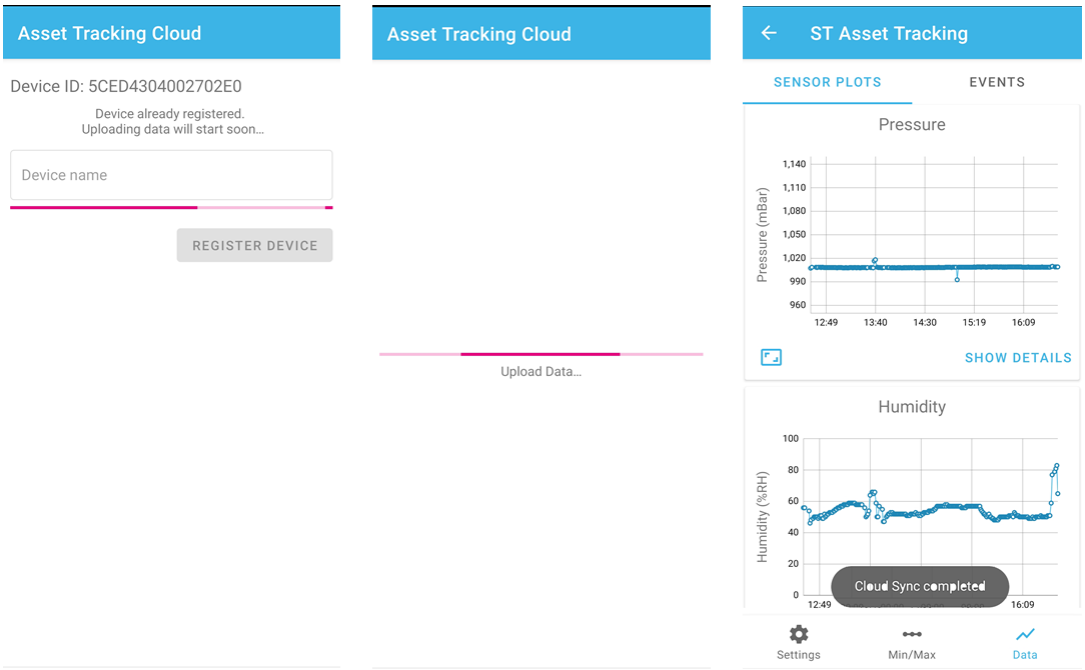


Figure 41. ST Asset Tracking application - data uploaded to the cloud and related plots



3 System setup guide

3.1 Hardware description

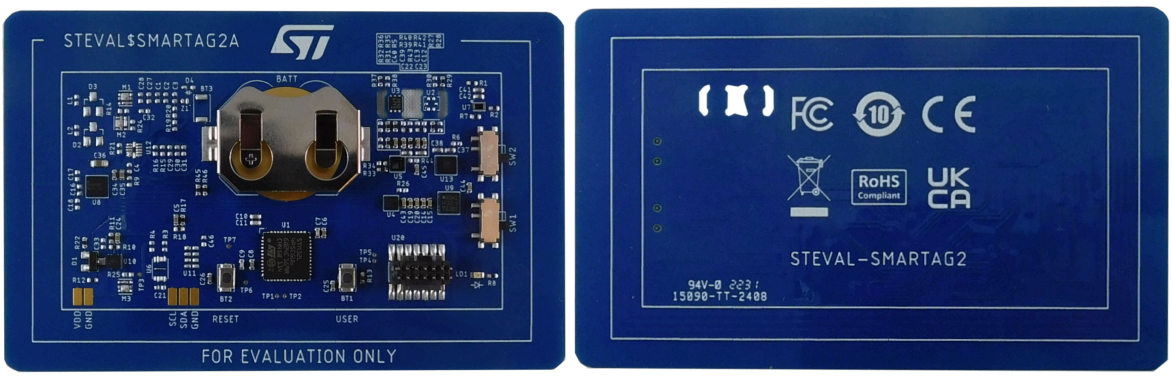
3.1.1 STEVAL-SMARTAG2 evaluation board

The STEVAL-SMARTAG2 is an NFC-enabled sensor node, which embeds inertial MEMS sensors and environmental sensors, an STM32 microcontroller, and a dynamic NFC tag for communication with NFC readers, such as tablets and smartphones.

The STEVAL-SMARTAG2 can optionally be equipped with a battery charger fed through a full-wave rectifier for NFC energy harvesting (to be put on top of the energy harvester already embedded in the dynamic NFC tag), a secure element to support authentication and state-of-the-art cryptographic security, and a real-time clock (RTC) with an embedded crystal oscillator to enable an accurate timekeeping and time stamping.

The board has a small and thin form factor, comparable to the size of a credit card. This makes it particularly fit for deployment in the field and data collection.

Figure 42. STEVAL-SMARTAG2 evaluation board (top and bottom views)



3.2 Hardware setup

The following hardware components are required:

- One STEVAL-SMARTAG2 evaluation board
- One STLINK-V3SET (or STLINK-V3MINI) debugger/programmer
- One USB type A to Micro-B USB cable to connect the STLINK-V3SET (or STLINK-V3MINI) to the PC
- One CR 2032 battery

3.3 Software setup

The following software components are needed to set up a suitable development environment to create applications for the STEVAL-SMARTAG2 evaluation board:

- FP-SNS-SMARTAG2: complete firmware to access data from an IoT node with a dynamic NFC tag, environmental, ambient light, and motion sensors and to enable the firmware update by NFC. The package provides easy portability across different MCU families, thanks to STM32Cube. The FP-SNS-SMARTAG2 firmware and related documentation is available on www.st.com.
- Development tool-chain and compiler. The STM32Cube expansion software supports the following environments:
  - IAR Embedded Workbench for ARM® toolchain + ST-LINK
  - Real View Microcontroller Development Kit (MDK-ARM) toolchain + ST-LINK
  - Integrated Development Environment for STM32 (STM32CubeIDE) + ST-LINK

3.4 STEVAL-SMARTAG2 evaluation board setup

To program the STEVAL-SMARTAG2 evaluation board, use the STLINK-V3SET (or STLINK-V3MINI) debugger/programmer. Download the relevant version of the STLINK-V3SET (or STLINK-V3MINI) USB driver by clicking STSW-LINK007 or STSW-LINK009.

Step 1. Connect the STLINK-V3SET (or STLINK-V3MINI) to your PC.

Step 2. Connect the STEVAL-SMARTAG2 to the STLINK-V3SET (or STLINK-V3MINI) via the SWD connector to start programming.

Figure 43. Connecting the STEVAL-SMARTAG2 via the SWD connector to the STLINK-V3SET

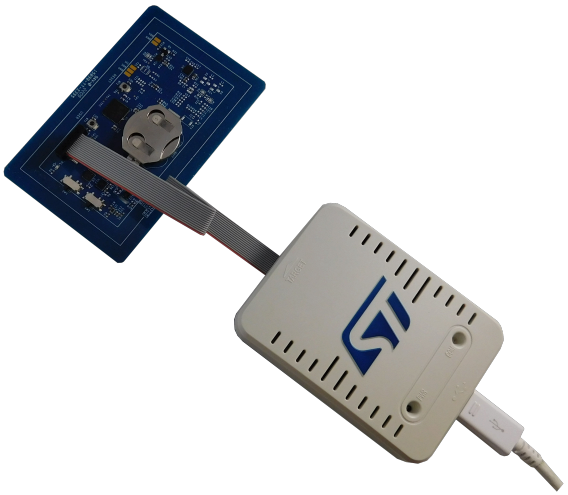
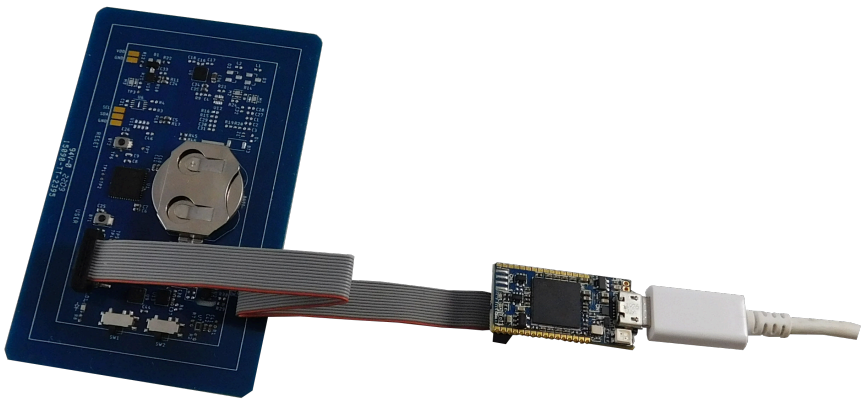


Figure 44. Connecting the STEVAL-SMARTAG2 via the SWD connector to the STLINK-V3MINI



Revision history

Table 1. **Document revision history**

Date	Revision	Changes
07-Oct-2022	1	Initial release.
03-Mar-2023	2	Updated Introduction, Section 1.1 Overview, Section 1.2 Architecture, Section 1.3 Folder structure, Section 1.5 Sample application description, Section 1.5.2 SmarTag2 application description, Section 1.5.3.1 Firmware update. Added Section 1.5.1 OneShot application description. Added references to STEVAL$SMARTAG2B, GitHub and STNFCSensor.

Contents

List of tables

List of figures