User manual

Getting started with the STM32Cube function pack for STEVAL-PROTEUS1 sensor data log and file transfer over BLE and USB

Introduction

FP-SNS-DATAPRO1 is an STM32Cube function pack that allows you to store data from any combination of the sensors, mounted on the STEVAL-PROTEUS1 evaluation kit, configured up to the maximum sampling rate.

Sensor data can be stored into the FatFs volume created inside the embedded 2Gb NOR Flash Memory or streamed to a PC via USB, since the evaluation board can act as a Windows Compatible ID (WCID) device, using the command line interface (cli) companion host software provided for Windows and Linux.

The FP-SNS-DATAPRO1 allows you to configure the board via a dedicated JSON file as well as starting and controlling data acquisition.

The STEVAL-PROTEUS that runs the FP-SNS-DATAPRO1 acts as a BLE device and it is supported by the ST BLE sensor (STBLESensor) application for Android and iOS. Through this application you can configure the sensors mounted on the STEVAL-PROTEUS, control the start/stop acquisition as well as transfer the data stored into its NOR Flash Memory to the mobile device that is running the app.

To read sensor data acquired using FP-SNS-DATAPRO1, easy-to-use scripts in Python is provided within the software package. The scripts have been successfully tested with Python 3.12.

Furthermore, the FP-SNS-DATAPRO1 can make the STEVAL-PROTEUS acting as a USB mass storage so, connecting it to a USB port of a PC and using a file manager you can manage folders and files inside the FatFs volume inside the embedded 2Gb NOR Flash Memory.

Related links

Visit the STM32Cube ecosystem web page for the most up to date resources and reference material

1 FP-SNS-DATAPRO1 software expansion for STM32Cube

1.1 Overview

The FP-SNS-DATAPRO1 is an STM32 ODE function pack and expands STM32Cube functionality.

The software package provides a comprehensive solution to save data from any combination of sensors configured up to the maximum sampling rate.

The data can be saved either into the embedded NOR Flash Memory or into a folder of the PC connected via USB.

This function pack also provide a solution to transfer the files stored into the NOR Flash Memory to a mobile device via Bluetooth.

Furthermore, it makes the board like a USB mass storage.

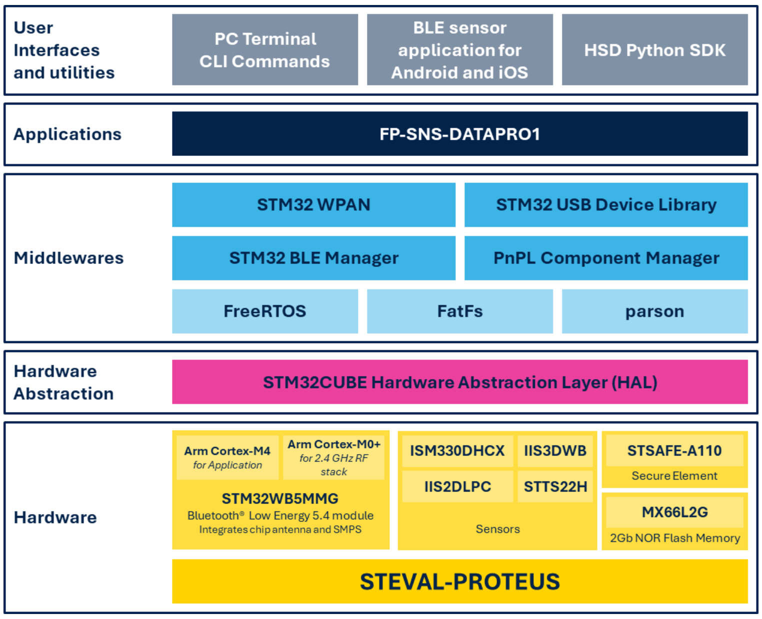
The key package features are:

- High-rate data capture software suite:
  - Two mutually exclusive modes to store data:
-
  -
    - Data storage into embedded 2Gb NOR Flash Memory.
    - Data sent to a PC via USB WCID and CLI companion host software.
-
  - Compatible with STBLESensor, Bluetooth® Low Energy app for system setup and real-time control.
  - Timestamping for sensor data synchronization.
  - Firmware updates over-the-air (FUOTA).
- Embedded software, middleware, and drivers:
  - STM32 Wireless Personal Area Network Middleware.
  - STM32 BLE Manager Library to manage the BLE service according to the STM32WB middleware APIs. (Parson library is needed).
  - PnPL Component Manager to handle PnP-like commands and properties generated through a Digital Twins Definition Language (DTDL).
  - FatFS third-party FAT file system module for small embedded systems.
  - FreeRTOS third-party RTOS kernel for embedded devices.
  - Parson third party lightweight json library written in C.
  - STM32Cube HAL drivers for easy portability across different MCU families.
  - Sensor BSP drivers.
  - STEVAL-PROTEUS BSP driver.
- Utilities:
  - Command-Line-Interface (CLI) companion host software for real-time control applications.
  - HSD Python SDK for sensor data analysis.
- Free, user-friendly license terms

1.2 Architecture

The FP-SNS-DATAPRO1 software is designed for the STEVAL-PROTEUS1 development kit, respecting the compliance with STM32Cube architecture, structured into a set of layers of increasing abstraction.

Figure 1. FP-SNS-DATAPRO1 Software Architecture



The hardware abstraction layer (HAL) interfaces with the hardware and provides the low-level drivers and the hardware interfacing methods to interact with the upper layers (application, libraries, and stacks). It provides APIs for the communication peripherals (I^2C, SPI, UART, etc.) for initialization and configuration, data transfer and communication errors.

There are two types of HAL driver APIs:

- generic APIs which provide common and generic functions to the entire STM32 series
- extension APIs which provide specific, customized functions for a particular family or a specific part number

The package extends STM32Cube by providing a board support package (BSP), which deals with the board specific peripherals and functions (LED, user button, etc.).

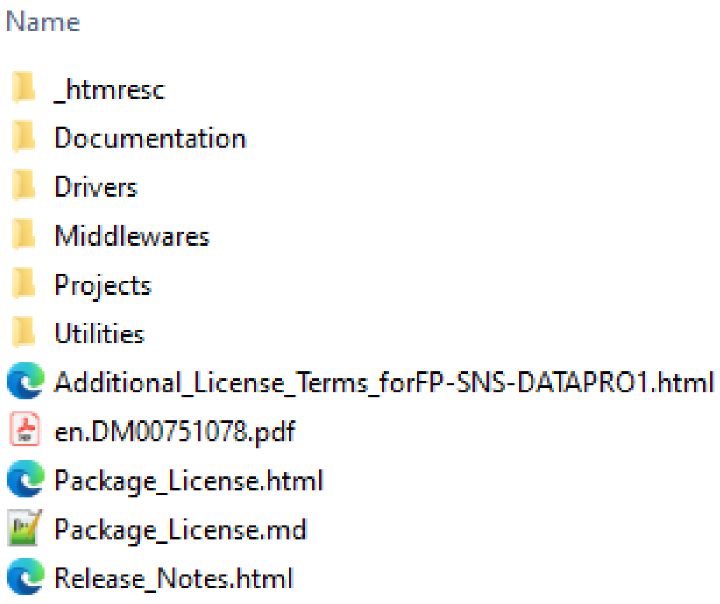
The BSP structure follows the hardware structure, including a component management layer as well as the specific layers of the boards used.

On top of such features, inherited from STM32Cube, FP-SNS-DATAPRO1 adds the code reusability at application level.

1.3 Folders

The FP-SNS-DATAPRO1 firmware package folder structure follows the layer-based approach of the STM32Cube architecture.

Figure 2. FP-SNS-DATAPRO1 package folder structure

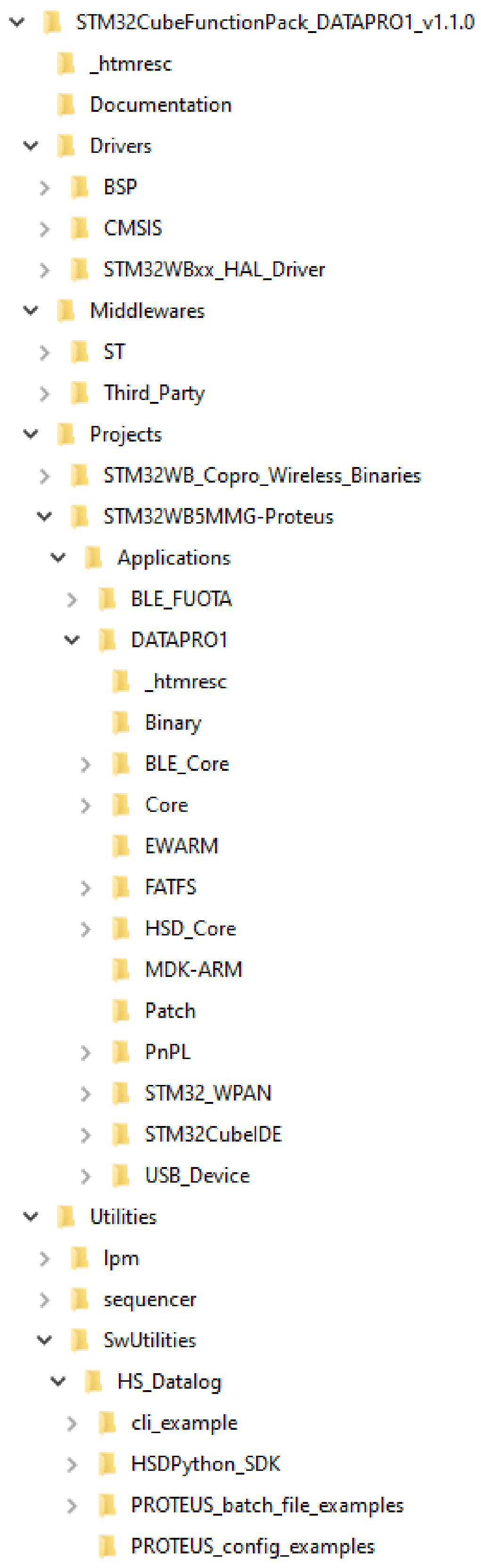


Name
_htmresc
Documentation
Drivers
Middlewares
Projects
Utilities
Additional_License_Terms_forFP-SNS-DATAPRO1.html
en.DM00751078.pdf
Package_License.html
Package_License.md
Release_Notes.html

The folders included in the software package are:

- **Documentation**: contains a compiled HTML file generated from the source code which details the software components and APIs.
- **Drivers**: contains the HAL drivers and 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 ARM Cortex-M processor series.
- **Middlewares**: contains all the libraries released by ST or Third Party, as listed below:
  - **STM32 WPAN**: STM32 Wireless Personal Area Network Middleware developed within the STM32WB framework that is used to support Bluetooth Low Energy 5 Certified Applications.
  - **STM32 BLE Manager**:STM32 middleware providing the APIs to manage the BLE service according to the STM32WB middleware APIs. Parson library must be included.
  - **STM32 USB Device Library**: STM32 middleware for USB connectivity.
  - **PnPL Component Manager**: implements the interface used to handle PnP-like commands and properties generated through a Digital Twins Definition Language (DTDL).
  - **FreeRTOS**: is a real-time operating system kernel for embedded devices.
  - **FatFs**: is a lightweight software library for microcontrollers and embedded systems that implements FAT file system support.
  - **Parson**: is a lightweight JSON (JavaScript Object Notation) library written in C.
- **Projects**: contains a sample application implementing the data logger. This application is provided for the STEVAL-PROTEUS1 development kit with three development environments:
  - IAR Embedded Workbench for ARM (IAR-EWARM v9.20.1).
  - Keil Microcontroller Development Kit for ARM (MDK-ARM v5.37.0).
  - ST Integrated Development Environment for STM32 (STM32CubeIDE v1.14.0).
  - Each IDE supports STLINK-V3MINIE (or other STLINK-V3 compact in-circuit debugger and programmer for STM32)

  This folder also contains the STM32 Coprocessor Wireless BLE stack binary for the STM32WB5x.
- **Utilities**: contains a few complementary project files as well as the application software (i.e., Python scripts, cli_example, and JSON configuration examples).



```
STM32CubeFunctionPack_DATAPRO1_v1.1.0
  _htmresc
  Documentation
  Drivers
    BSP
    CMSIS
    STM32WBxx_HAL_Driver
  Middlewares
    ST
    Third_Party
  Projects
    STM32WB_Copro_Wireless_Binaries
    STM32WB5MMG-Proteus
      Applications
        BLE_FUOTA
        DATAPRO1
          _htmresc
          Binary
          BLE_Core
          Core
          EWARM
          FATFS
          HSD_Core
          MDK-ARM
          Patch
          PnPL
          STM32_WPAN
          STM32CubeIDE
          USB_Device
  Utilities
    lpm
    sequencer
    SwUtilities
      HS_Datalog
        cli_example
        HSDPython_SDK
        PROTEUS_batch_file_examples
        PROTEUS_config_examples
```

Figure 3. FP-SNS-DATAPRO1 package sub-folder structure

1.4 APIs

Detailed technical information with full user API function and parameter description are in a compiled HTML file in the “Documentation” folder.

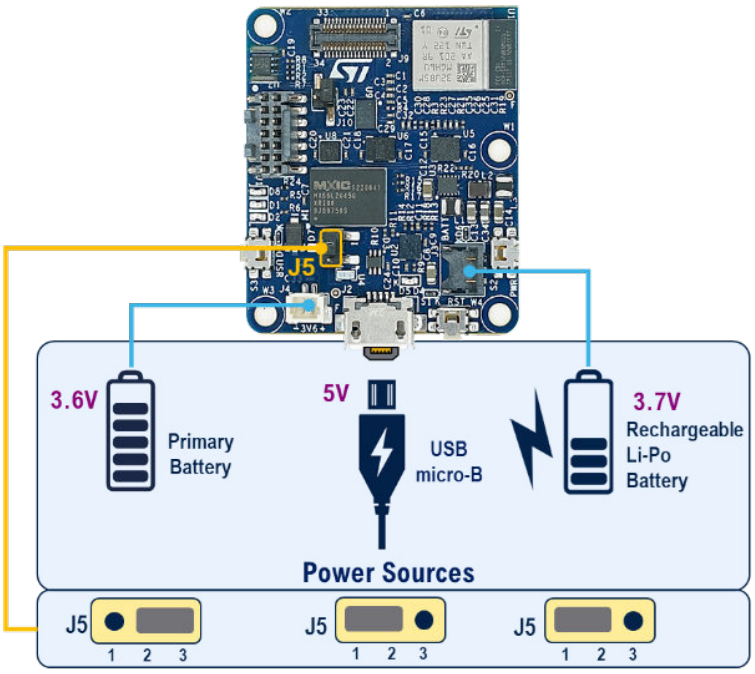
2 Getting started

2.1 STEVAL-PROTEUS power settings

Acting on J5 electrical jumper, two power path are possible to route the source to the DC/DC:

- Through the battery charger (the following sources can coexist):
  - Rechargeable Li-Po battery (3.7 VDC), included in the kit.
  - USB (5 VDC), also used to recharge the Li-Po battery.
- Bypassing the battery charger:
  - Primary battery (3.6 VDC), not included in the kit.

Figure 4. STEVAL-PROTEUS power settings



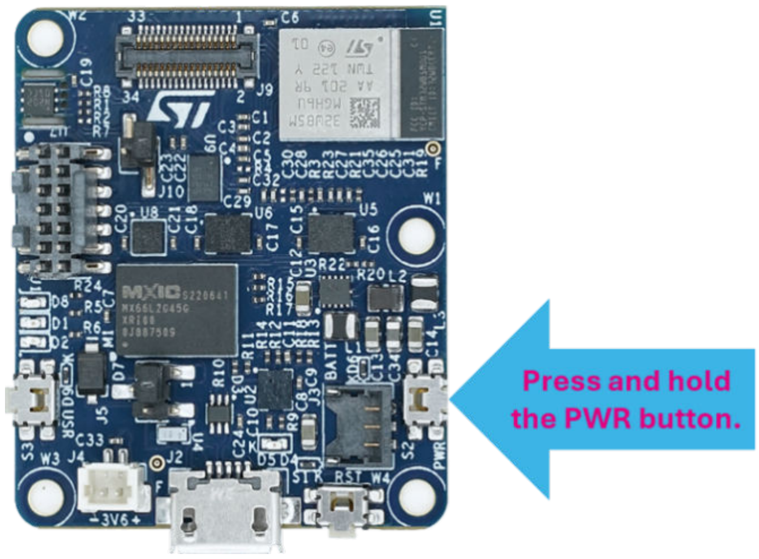
2.1.1 Turn the STEVAL-PROTEUS on and off

Regarding the type of source, you can turn the board on and off as shown in the following table.

Table 1. How turn on and off the STEVAL-PROTEUS

Power Source	Turn On	Turn Off
Li-Po battery only.	Press and hold the PWR button for more than 3 s. All LEDs blink.	Press and hold the PWR button for more than 3 s. Alternatively, via STBLESensor app (see Section 2.3.9: Board Configuration). All LEDs blink.
USB only.	By connecting the USB cable.	Disconnecting the USB cable.
Li-Po battery and USB.	Like USB only.	Like USB only then like Li-Po battery only.
Primary battery.	Connecting the battery.	Disconnecting the battery.

Figure 5. STEVAL-PROTEUS power button



2.2 Program STEVAL-PROTEUS with FP-SNS-DATAPRO1

As DATAPRO1 application included in the FP-SNS-DATAPRO1 function pack is not the default firmware on the STEVAL-PROTEUS1, you have to download the application firmware, *FP-SNS-DATAPRO1_1_1_0.bin*, that you can find in the folder:

"\Projects\STM32WB5MMG-Proteus\Applications\DATAPRO1\Binary".

Additionally, you may need to download the BLE stack, *stm32wb5x_BLE_Stack_full_fw.bin*, that you can find in the folder:

"Projects\STM32WB_Copro_Wireless_Binaries\STM32WB5x".

Both files can be download into the flash memory of STM32WB5MMG, mounted on the STEVAL-PROTEUS, following two ways:

- Wireless, via FUOTA using the STBLESensor app.

Figure 6. QR code to get STBLESensor - Android

Figure 7. QR code to get STBLESensor - iOS

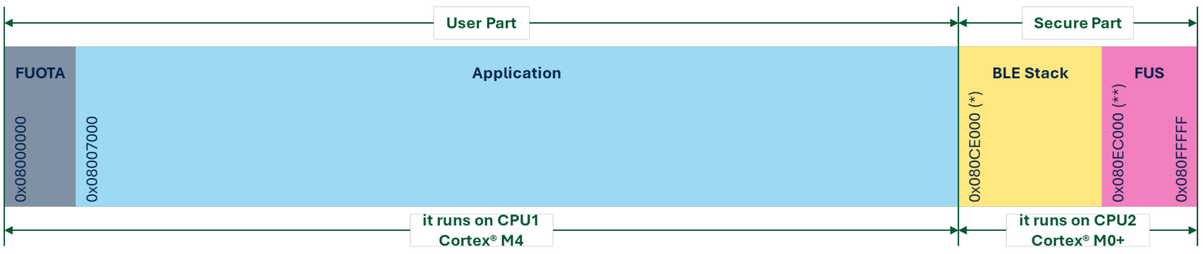
- Wired, using the STM32CubeProgrammer software for STM32 (STM32CubeProg) and the STLINK-V3 compact in-circuit debugger and programmer for STM32 (STLINK-V3MINIE).

In any case, the MCU flash memory must be properly loaded with the FUOTA application firmware, *PROTEUS_BLE_FUOTA_RFWKPCLK_HSE_DIV1024.bin*, that you can find in the folder:

"Projects\STM32WB5MMG-Proteus\Applications\BLE_FUOTA\Binary".

Consider that the binary file related to the FUOTA application firmware can only be downloaded in wired mode.

Figure 8. STM32WB5MMG Flash Memory partition



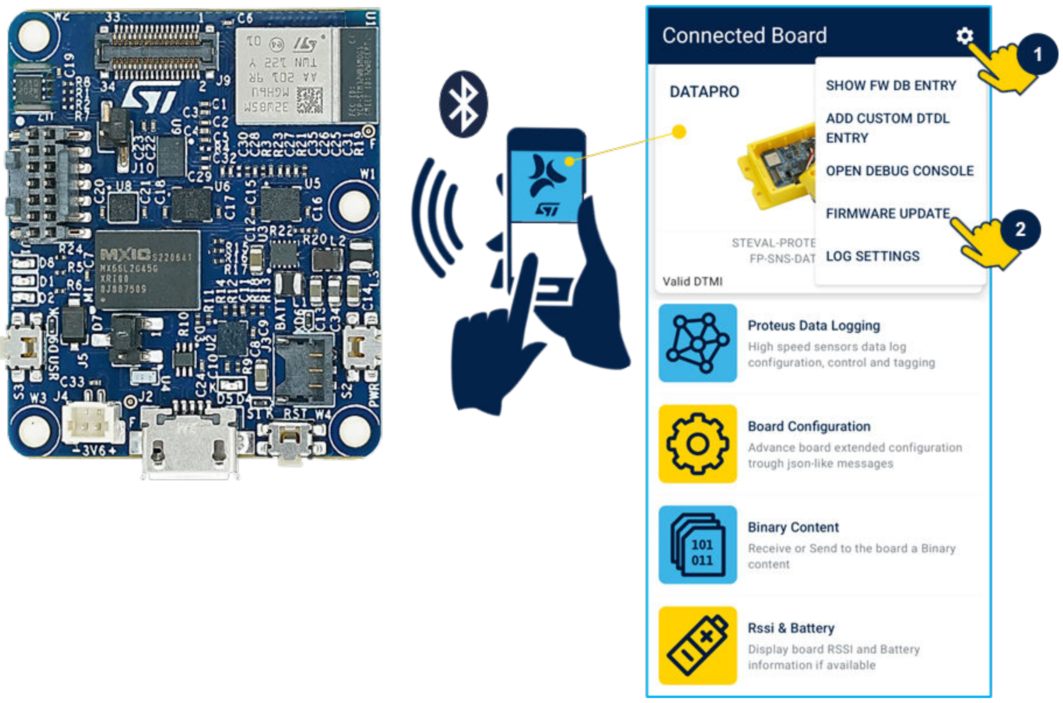
(*) start address for ***stm32wb5x_BLE_Stack_full_fw.bin V1.17.0***

(**) start address for ***stm32wb5x_FUS_fw.bin V1.2.0***

2.2.1 FUOTA - firmware update over-the-air

The STEVAL-PROTEUS1 comes out of the box with a default firmware that enables Bluetooth pairing and FUOTA through the STBLESensor app.

Figure 9. STBLESensor app to update STEVAL-PROTEUS firmware



Proteus Data Logging
High speed sensors data log configuration, control and tagging

Board Configuration
Advance board extended configuration trough json-like messages

Binary Content
Receive or Send to the board a Binary content

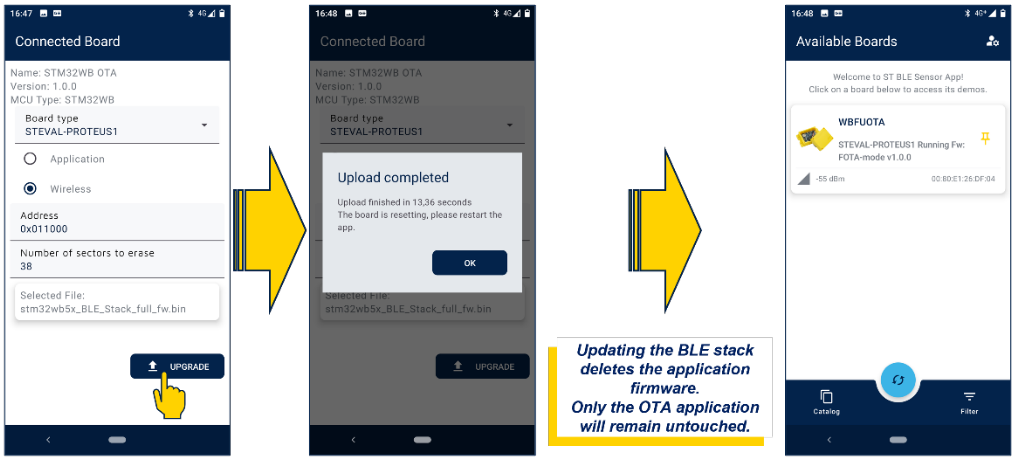
Rssi & Battery
Display board RSSI and Battery information if available

To update the firmware, please, act as following:

- Supply the STEVAL-PROTEUS.
- Copy *"FP-SNS-DATAPRO1_1_1_0.bin"* into the smart mobile device.
- Additionally, copy *"stm32wb5x_BLE_Stack_full_fw.bin"* as well.
- Launch the STBLESensor app from your smart mobile device.
- Discover the STEVAL-PROTEUS and connect to it.
- Tap the gear menu, in the top-right corner, and launch the firmware update.

If you need to update the BLE stack you must perform this action first.

Figure 10. STEVAL-PROTEUS BLE stack firmware update over-the-air



Name: STM32WB OTA
Version: 1.0.0
MCU Type: STM32WB

Address
0x011000

Number of sectors to erase
38

Selected File:
stm32wb5x_BLE_Stack_full_fw.bin

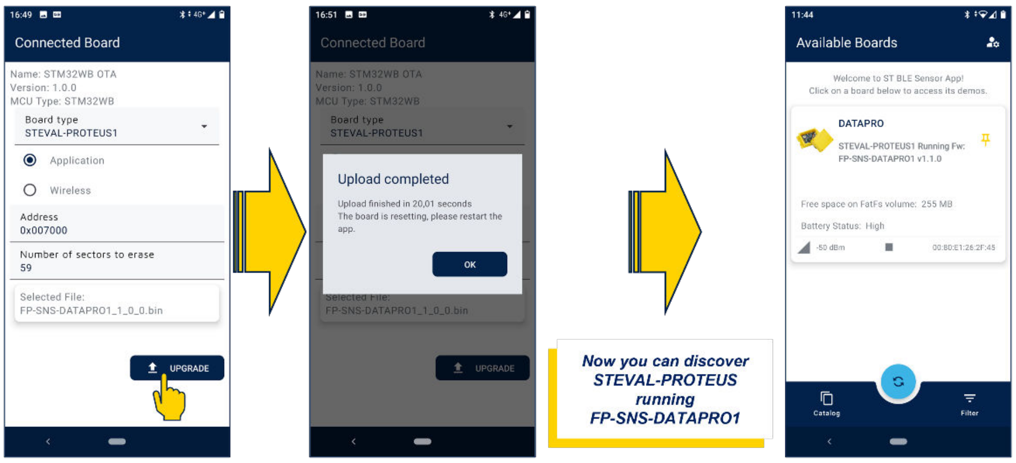
Upload completed
Upload finished in 13,36 seconds
The board is resetting, please restart the app.

Welcome to ST BLE Sensor App!
Click on a board below to access its demos.

WBFUOTA
STEVAL-PROTEUS1 Running Fw: FOTA-mode v1.0.0
-55 dBm 00:80:E1:26:DF:04

Updating the BLE stack deletes the application firmware.
Only the OTA application will remain untouched.

In the same way, you can update the application firmware to evaluate the FP-SNS-DATAPRO1.


Connected Board

Name: STM32WB OTA
Version: 1.0.0
MCU Type: STM32WB

Board type
STEVAL-PROTEUS1

Application
Wireless

Address
0x007000

Number of sectors to erase
59

Selected File:
FP-SNS-DATAPRO1_1_0_0.bin

Upload completed

Upload finished in 20,01 seconds
The board is resetting, please restart the app.

Now you can discover STEVAL-PROTEUS running FP-SNS-DATAPRO1

Available Boards

Welcome to ST BLE Sensor App!
Click on a board below to access its demos.

DATAPRO
STEVAL-PROTEUS1 Running Fw: FP-SNS-DATAPRO1 v1.1.0

Free space on FatFs volume: 255 MB
Battery Status: High
-50 dBm
00:80:E1:26:2F:45

Figure 11. STEVAL-PROTEUS application firmware update over-the-air

Note: In case STEVAL-PROTEUS is not displayed among the available boards in the STBLESensor app, you need to deactivate and activate bluetooth on your mobile device to discover the board. the STEVAL-PROTEUS so it will not be discovered by the STBLESensor app.

2.2.2 Firmware update with the programmer

If the FUOTA is not available, since the related application was corrupted or missing inside the MCU flash memory, or just because you prefer it, you can download the firmware using the wired mode.

You need:

- An STLINK-V3 compact in-circuit debugger and programmer for STM32 (STLINK-V3MINIE).
- A USB-A to USB-C cable.
- An STDC14 to STDC14 cable, provided with STLINK-V3MINIE.
- A PC able to run the STM32CubeProgrammer software for STM32 (STM32CubeProg).

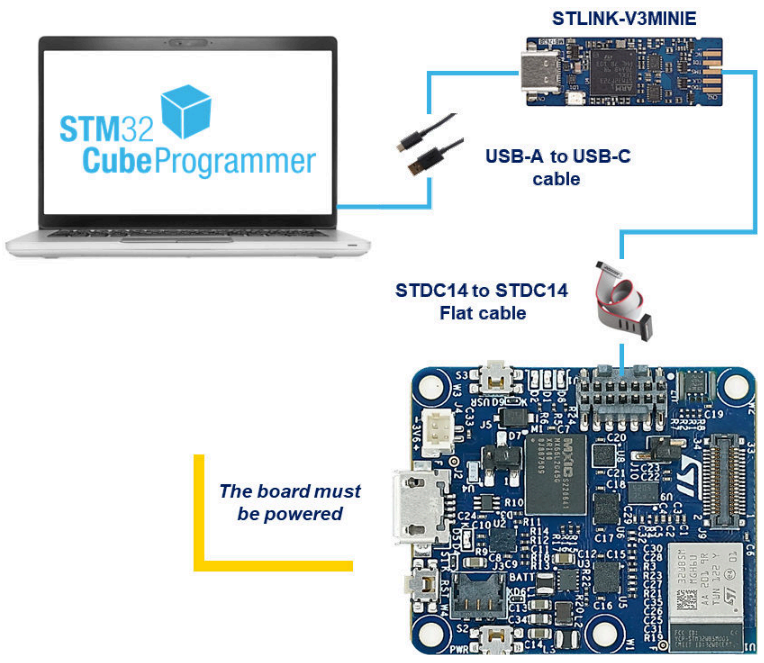


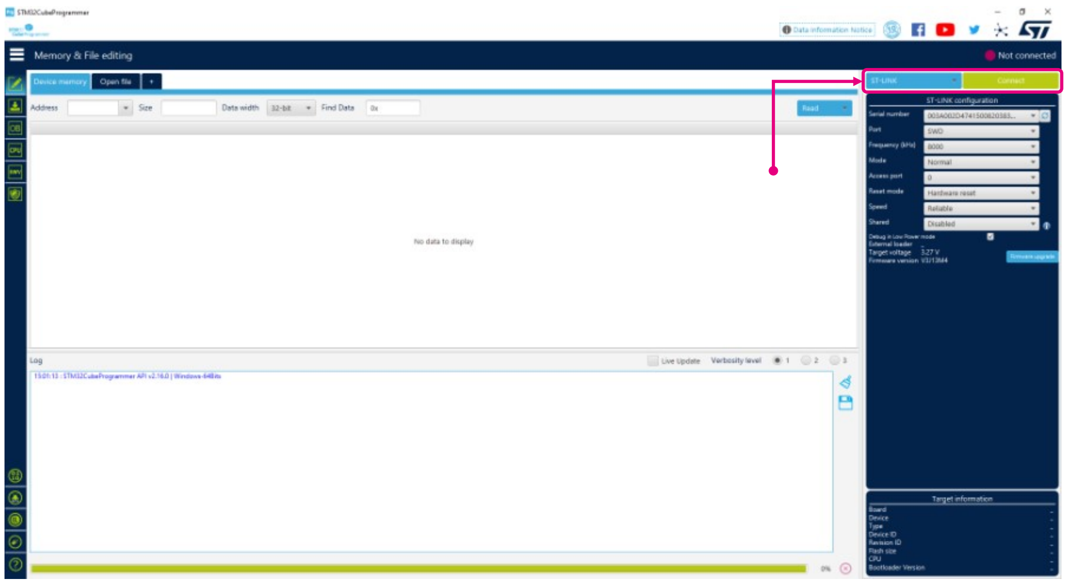
Figure 12. STEVAL-PROTEUS connected to a PC via STLINK-V3

The figure above shows how to connect the STEVAL-PROTEUS to a PC for programming it using STM32CubeProg through the STLINK-V3MINIE.

Now you are ready to proceed.

1. Launch the STM32CubeProg.
2. Select [ST-LINK] mode on the top-right corner.

Figure 13. **STM32CubeProgrammer - ST-LINK mode selection**



ST-LINK configuration

Serial number	003A002D4741500820383...
Port	SWD
Frequency (kHz)	8000
Mode	Normal
Access port	0
Reset mode	Hardware reset
Speed	Reliable
Shared	Disabled
Debug in Low Power mode	✓
External loader	-
Target voltage	3.27 V
Firmware version	V3J13M4

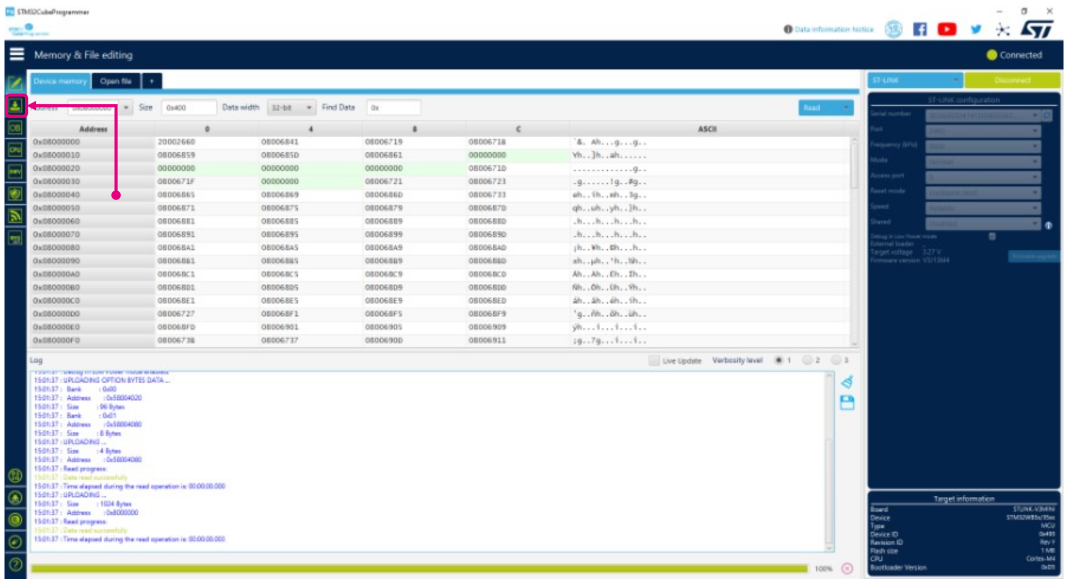
```
15:01:13 : STM32CubeProgrammer API v2.16.0 | Windows-64Bits
```

Target information

Board	-
Device	-
Type	-
Device ID	-
Revision ID	-
Flash size	-
CPU	-
Bootloader Version	-

1. *Click on [Connect].*

Figure 14. **STM32CubeProgrammer - Connect**



Address	0	4	8	C	ASCII
0x08000000	20002660	08006841	08006719	08006718	`&. Ah...g...g..
0x08000010	08006859	0800685D	08006861	00000000	Yh..]h..ah......
0x08000020	00000000	00000000	00000000	0800671D	.............g..
0x08000030	0800671F	00000000	08006721	08006723	.g......!g..#g..
0x08000040	08006865	08006869	0800686D	08006733	eh..ih..mh..3g..
0x08000050	08006871	08006875	08006879	0800687D	qh..uh..yh..}h..
0x08000060	08006881	08006885	08006889	0800688D	.h...h...h...h..
0x08000070	08006891	08006895	08006899	0800689D	.h...h...h...h..
0x08000080	080068A1	080068A5	080068A9	080068AD	¡h..¥h..©h...h..
0x08000090	080068B1	080068B5	080068B9	080068BD	±h..µh..¹h..½h..
0x080000A0	080068C1	080068C5	080068C9	080068CD	Áh..Åh..Éh..Íh..
0x080000B0	080068D1	080068D5	080068D9	080068DD	Ñh..Õh..Ùh..Ýh..
0x080000C0	080068E1	080068E5	080068E9	080068ED	áh..åh..éh..íh..
0x080000D0	08006727	080068F1	080068F5	080068F9	'g..ñh..õh..ùh..
0x080000E0	080068FD	08006901	08006905	08006909	ýh...i...i...i..
0x080000F0	08006738	08006737	0800690D	08006911	8g..7g...i...i..

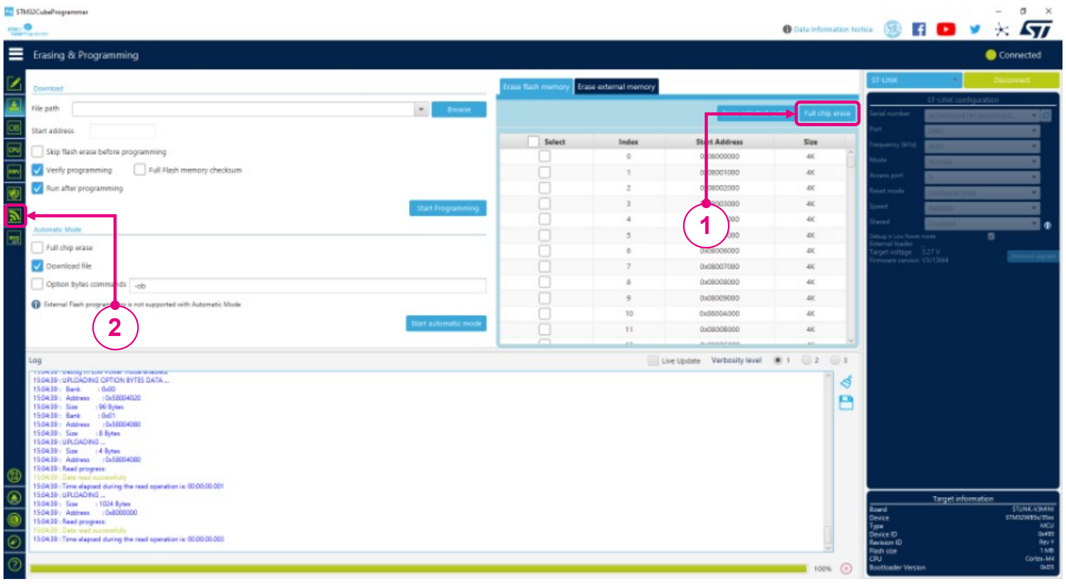
```
15:01:37 : Debug in Low Power mode enabled.
15:01:37 : UPLOADING OPTION BYTES DATA ...
15:01:37 :   Bank          : 0x00
15:01:37 :   Address       : 0x58004020
15:01:37 :   Size          : 96 Bytes
15:01:37 :   Bank          : 0x01
15:01:37 :   Address       : 0x58004080
15:01:37 :   Size          : 8 Bytes
15:01:37 : UPLOADING ...
15:01:37 :   Size          : 4 Bytes
15:01:37 :   Address       : 0x58004080
15:01:37 : Read progress:
15:01:37 : Data read successfully
15:01:37 : Time elapsed during the read operation is: 00:00:00.000
15:01:37 : UPLOADING ...
15:01:37 :   Size          : 1024 Bytes
15:01:37 :   Address       : 0x8000000
15:01:37 : Read progress:
15:01:37 : Data read successfully
15:01:37 : Time elapsed during the read operation is: 00:00:00.000
```

Target information

Board	STLINK-V3MINI
Device	STM32WB5x/35xx
Type	MCU
Device ID	0x495
Revision ID	Rev Y
Flash size	1 MB
CPU	Cortex-M4
Bootloader Version	0xD5

1. *Click on [Erasing & Programming].*

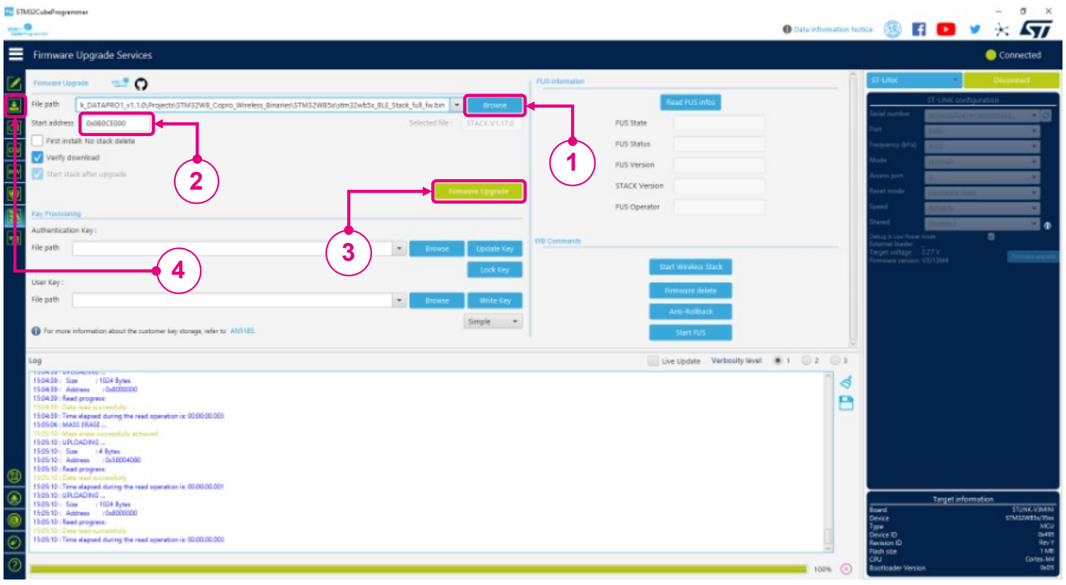
Figure 15. **STM32CubeProgrammer - Erasing**



```
15:04:39 : UPLOADING OPTION BYTES DATA ...
15:04:39 :   Bank         : 0x00
15:04:39 :   Address       : 0x58004020
15:04:39 :   Size         : 96 Bytes
15:04:39 :   Bank         : 0x01
15:04:39 :   Address       : 0x58004080
15:04:39 :   Size         : 8 Bytes
15:04:39 : UPLOADING ...
15:04:39 :   Size         : 4 Bytes
15:04:39 :   Address       : 0x58004080
15:04:39 : Read progress:
15:04:39 : Data read successfully
15:04:39 : Time elapsed during the read operation is: 00:00:00.001
15:04:39 : UPLOADING ...
15:04:39 :   Size         : 1024 Bytes
15:04:39 :   Address       : 0x8000000
15:04:39 : Read progress:
15:04:39 : Data read successfully
15:04:39 : Time elapsed during the read operation is: 00:00:00.000
```

1. *Click on [Full chip erase].*
2. *Wait for the operation to finish.*
3. *Click on [Firmware Upgrade Services].*

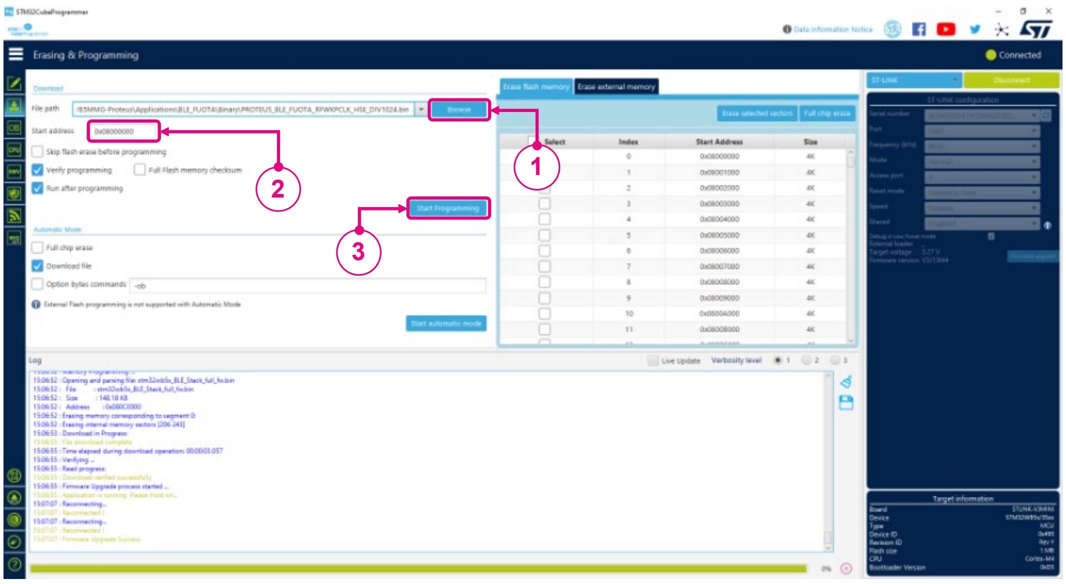
Figure 16. **STM32CubeProgrammer - FUS (BLE Stack)**



```
15:04:39 :   Size         : 1024 Bytes
15:04:39 :   Address       : 0x8000000
15:04:39 : Read progress:
15:04:39 : Data read successfully
15:04:39 : Time elapsed during the read operation is: 00:00:00.000
15:05:06 : MASS ERASE ...
15:05:10 : Mass erase successfully achieved
15:05:10 : UPLOADING ...
15:05:10 :   Size         : 4 Bytes
15:05:10 :   Address       : 0x58004080
15:05:10 : Read progress:
15:05:10 : Data read successfully
15:05:10 : Time elapsed during the read operation is: 00:00:00.001
15:05:10 : UPLOADING ...
15:05:10 :   Size         : 1024 Bytes
15:05:10 :   Address       : 0x8000000
15:05:10 : Read progress:
15:05:10 : Data read successfully
15:05:10 : Time elapsed during the read operation is: 00:00:00.000
```

1. *Select the file "stm32wb5x_BLE_Stack_full_fw.bin", as coprocessor wireless stack. You can find it in the folder "Projects\STM32WB_Copro_Wireless_Binaries\STM32WB5x".*
2. *Make sure the proposed start address matches the reported in the release notes related to the STM32WB Copro Wireless Binaries.*
3. *Click on [Firmware Upgrade] to perform the BLE stack upgrade.*
4. *Wait for the operation to finish.*
5. *Click on [Erasing & Programming].*

Figure 17. **STM32CubeProgrammer - Programming (FUOTA App)**

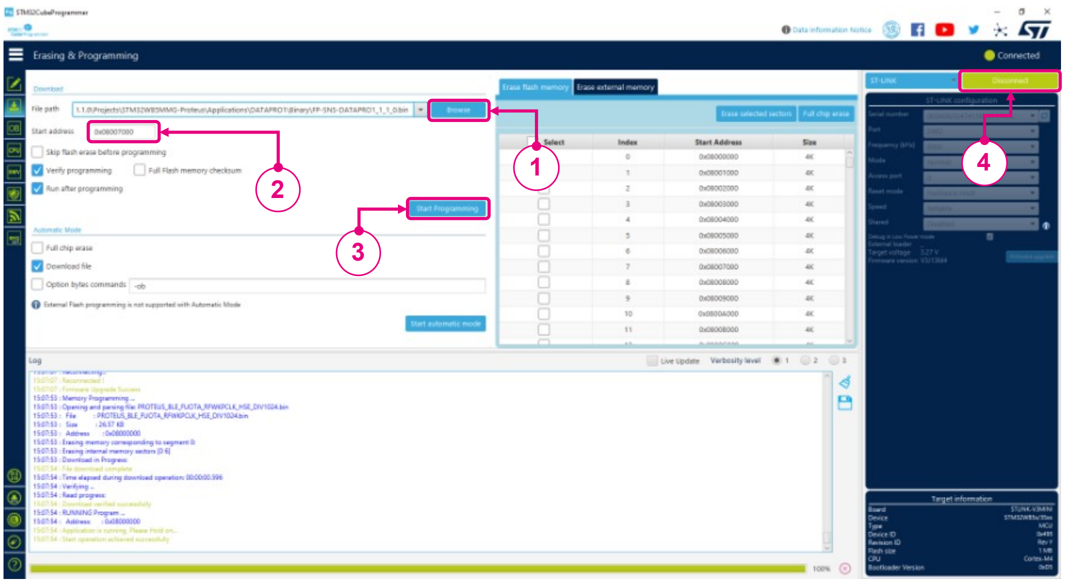


File path: /B5MMG-Proteus\Applications\BLE_FUOTA\Binary\PROTEUS_BLE_FUOTA_RFWKPCLK_HSE_DIV1024.bin
Start address: 0x08000000

```
15:06:52 : Opening and parsing file: stm32wb5x_BLE_Stack_full_fw.bin
15:06:52 :   File          : stm32wb5x_BLE_Stack_full_fw.bin
15:06:52 :   Size          : 148.18 KB
15:06:52 :   Address       : 0x080CE000
15:06:52 : Erasing memory corresponding to segment 0:
15:06:52 : Erasing internal memory sectors [206 243]
15:06:53 : Download in Progress:
15:06:55 : File download complete
15:06:55 : Time elapsed during download operation: 00:00:03.057
15:06:55 : Verifying ...
15:06:55 : Read progress:
15:06:55 : Download verified successfully
15:06:55 : Firmware Upgrade process started ...
15:06:55 : Application is running, Please Hold on...
15:07:07 : Reconnecting...
15:07:07 : Reconnected !
15:07:07 : Reconnecting...
15:07:07 : Reconnected !
15:07:07 : Firmware Upgrade Success
```

1. *Select the file "PROTEUS_BLE_FUOTA_RFWKPCLK_HSE_DIV1024.bin", as application firmware.*
2. *Set the start address at 0x08000000.*
3. *Click on [Start Programming] to download the FUOTA application firmware.*
4. *Wait for the operation to finish.*

Figure 18. **STM32CubeProgrammer - Programming (Demo App)**



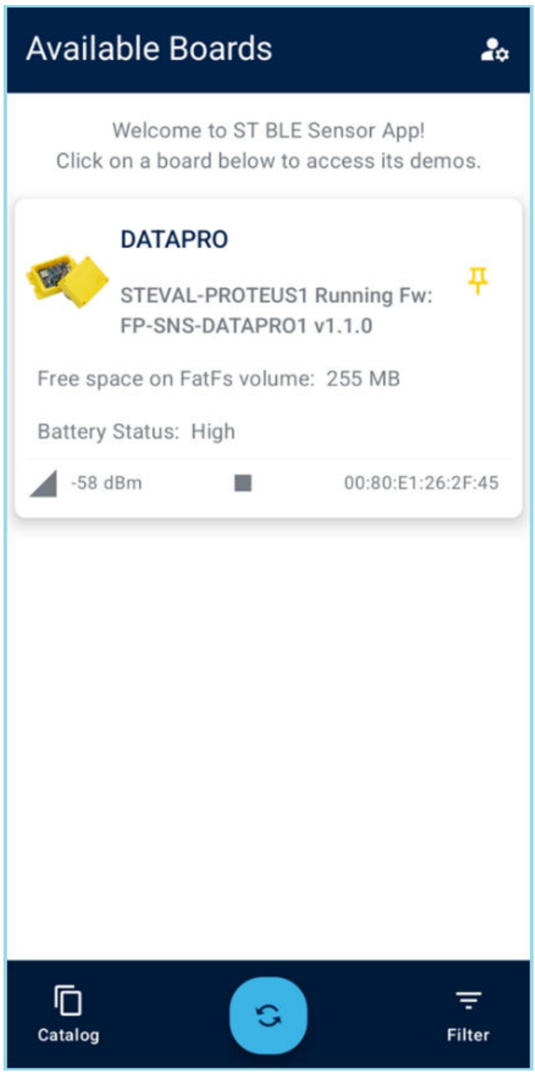
File path: 1.1.0\Projects\STM32WB5MMG-Proteus\Applications\DATAPRO1\Binary\FP-SNS-DATAPRO1_1_1_0.bin
Start address: 0x08007000

```
15:07:07 : Reconnected !
15:07:07 : Firmware Upgrade Success
15:07:53 : Memory Programming ...
15:07:53 : Opening and parsing file: PROTEUS_BLE_FUOTA_RFWKPCLK_HSE_DIV1024.bin
15:07:53 :   File          : PROTEUS_BLE_FUOTA_RFWKPCLK_HSE_DIV1024.bin
15:07:53 :   Size          : 26.57 KB
15:07:53 :   Address       : 0x08000000
15:07:53 : Erasing memory corresponding to segment 0:
15:07:53 : Erasing internal memory sectors [0 6]
15:07:53 : Download in Progress:
15:07:54 : File download complete
15:07:54 : Time elapsed during download operation: 00:00:00.596
15:07:54 : Verifying ...
15:07:54 : Read progress:
15:07:54 : Download verified successfully
15:07:54 : RUNNING Program ...
15:07:54 :   Address       : 0x08000000
15:07:54 : Application is running, Please Hold on...
15:07:54 : Start operation achieved successfully
```

1. *Select the file "FP-SNS-DATAPRO1_1_1_0.bin", as application firmware.*
2. *Set the start address at 0x08007000.*
3. *Click on [Start Programming] to download the DATAPRO1 application firmware.*
4. *Wait for the operation to finish.*
5. *The STEVAL-PROTEUS is fully programmed, you can click on [Disconnect].*

Now, the STEVAL-PROTEUS can be discovered by the STBLESensor app as shown in the following figure.

Figure 19. STEVAL-PROTEUS running FP-SNS-DATAPRO1 through the STBLESensor App



Available Boards

Welcome to ST BLE Sensor App!
Click on a board below to access its demos.

DATAPRO

STEVAL-PROTEUS1 Running Fw:
FP-SNS-DATAPRO1 v1.1.0

Free space on FatFs volume: 255 MB

Battery Status: High

-58 dBm

00:80:E1:26:2F:45

2.3 Evaluate STEVAL-PROTEUS with FP-SNS-DATAPRO1

With FP-SNS-DATAPRO1 you can capture data coming from the sensors mounted on the STEVAL-PROTEUS in two ways:

- stand-alone data logging without any cable connection, storing data into a FatFs volume created into the embedded 2Gb NOR flash memory.

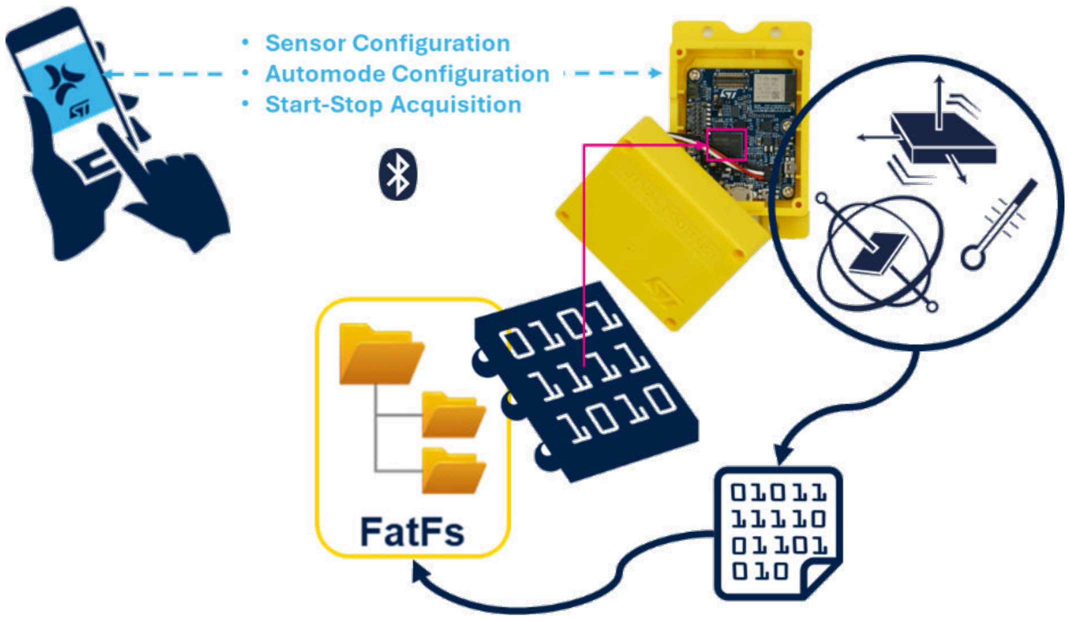
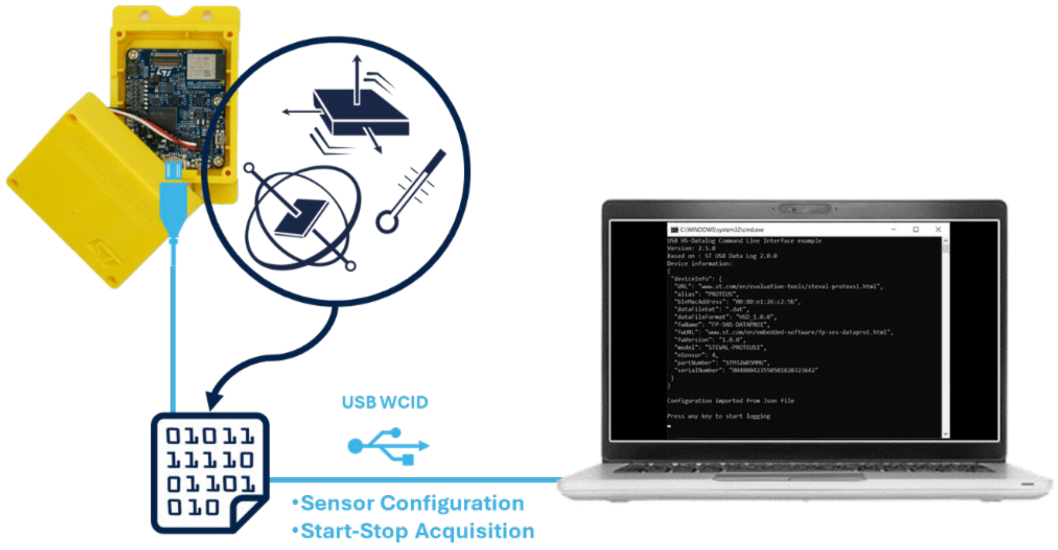


Figure 20. FP-SNS-DATAPRO1 stand-alone data logging

- USB WCID data logging, storing data into the connected PC.

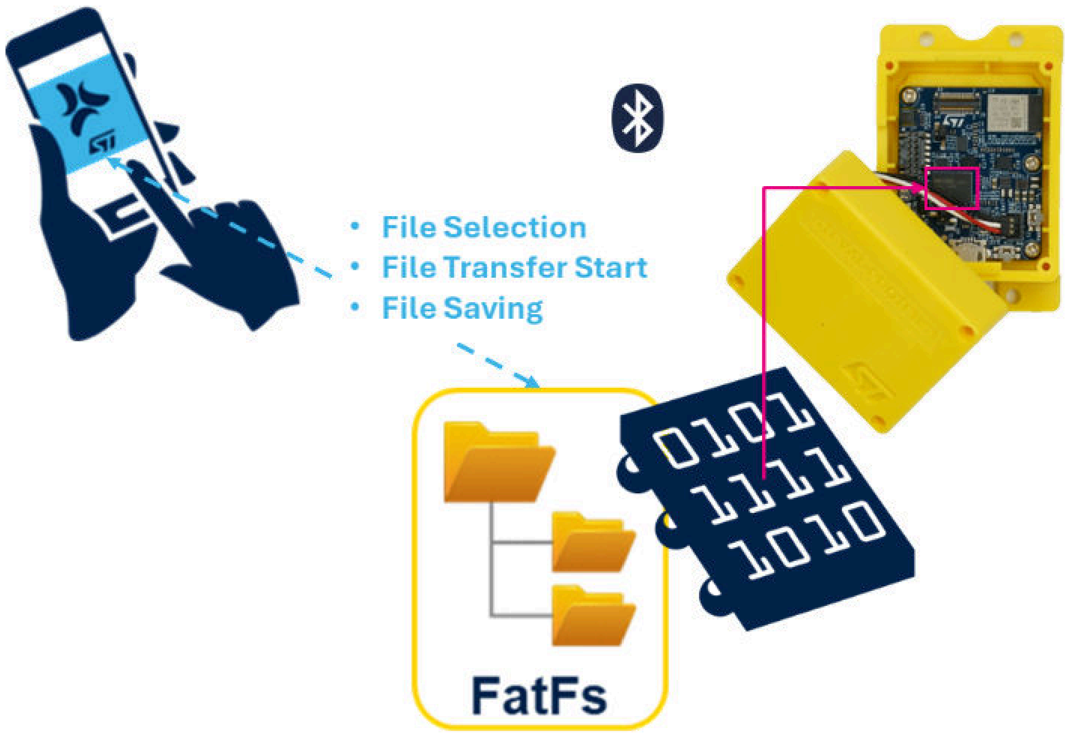


```
USB HS-Datalog Command Line Interface example
Version: 2.5.0
Based on : ST USB Data Log 2.0.0
Device information:
{
 "deviceInfo": {
  "URL": "www.st.com/en/evaluation-tools/steval-proteus1.html",
  "alias": "PROTEUS",
  "bleMacAddress": "00:80:e1:26:c2:5b",
  "dataFileExt": ".dat",
  "dataFileFormat": "HSD_1.0.0",
  "fwName": "FP-SNS-DATAPRO1",
  "fwURL": "www.st.com/en/embedded-software/fp-sns-datapro1.html",
  "fwVersion": "1.0.0",
  "model": "STEVAL-PROTEUS1",
  "nSensor": 4,
  "partNumber": "STM32WB5MMG",
  "serialNumber": "004800423550501820323642"
 }
}

Configuration imported from Json file

Press any key to start logging
```

Figure 21. FP-SNS-DATAPRO1 USB WCID data logging

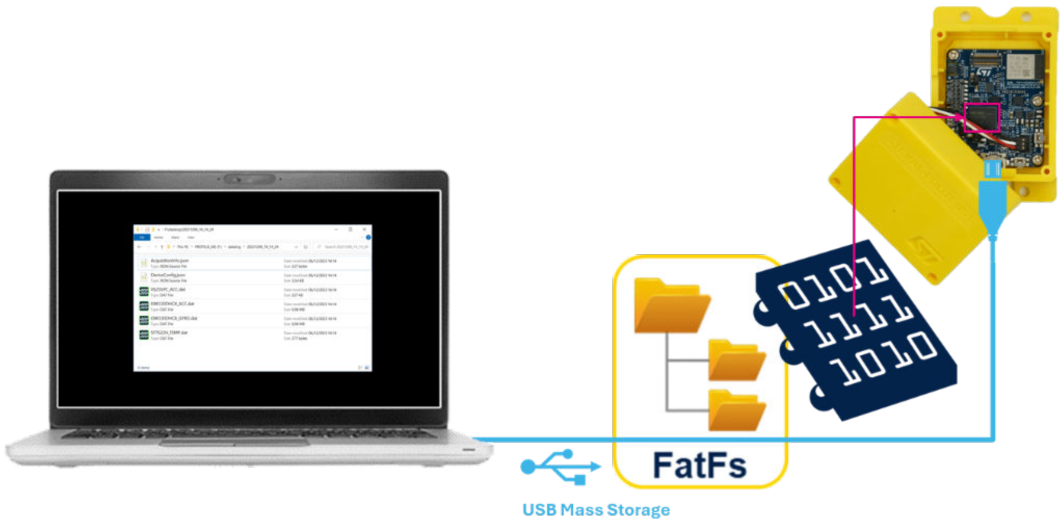
With FP-SNS-DATAPRO1 you can also transfer the captured sensor data from the FatFs volume, created into the embedded 2Gb NOR flash memory, to a mobile device that runs the STBLESensor app.

Figure 22. **FP-SNS-DATAPRO1 BLE file transfer**



With FP-SNS-DATAPRO1, the STEVAL-PROTEUS can be recognized by a computer as a USB mass storage so you can manage files and folders on the FatFs volume that is into the embedded 2Gb NOR flash memory.

Figure 23. **FP-SNS-DATAPRO1 USB mass storage**



AcquisitionInfo.json
DeviceConfig.json
IIS2DLPC_ACC.dat
ISM330DHCX_ACC.dat
ISM330DHCX_GYRO.dat
STTS22H_TEMP.dat

2.3.1 FP-SNS-DATAPRO1 stand-alone data logging

Sometimes the equipment to be monitored might be located in places where a wired connection to a laptop for data logging can be not easy.

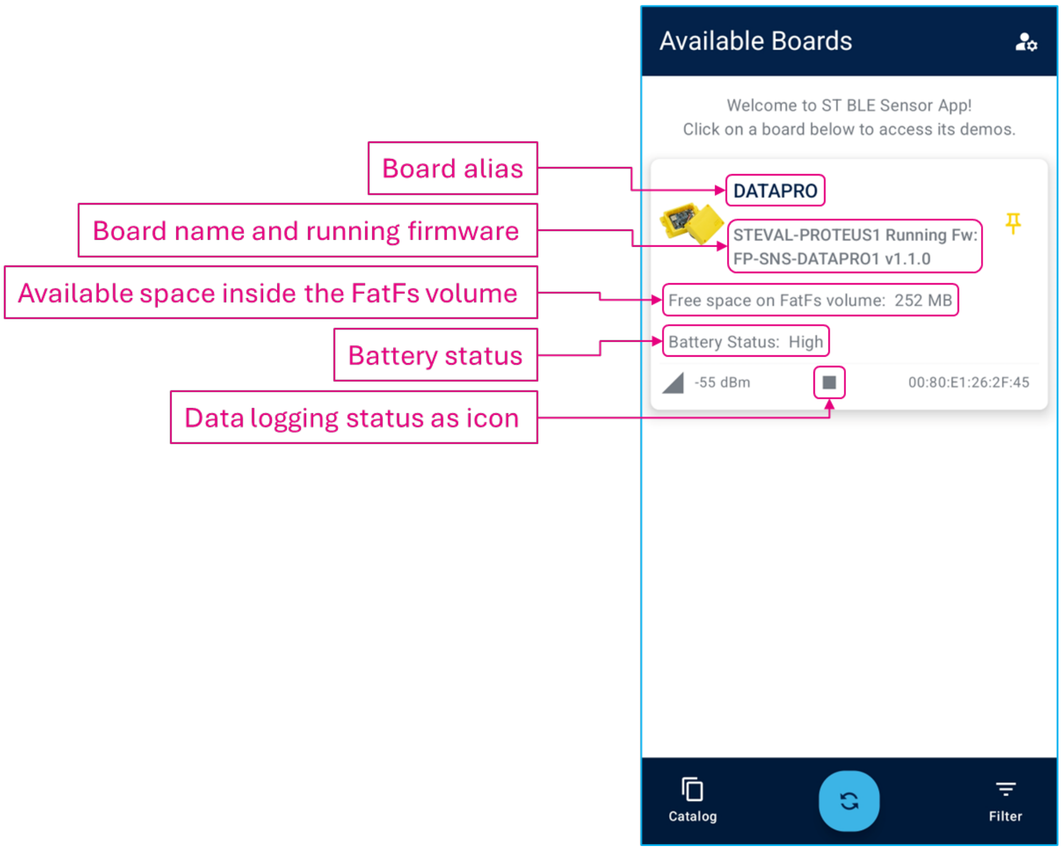
FP-SNS-DATAPRO1 gives you the possibility to use the STEVAL-PROTEUS to capture the data coming from its sensors and save it in a dedicated folder inside the FatFs volume located into the embedded 2Gb NOR flash memory. No cables or PC are needed, just a mobile device that runs the STBLESensor app. You can use it to connect, via BLE, to the STEVAL-PROTEUS, powered by the provided Li-Po battery, placed on the equipment to be monitored.

The paragraph Section 2.3.7: BLE file transfer (Binary Content) shows you how to retrieve these data via BLE. You can also use the board as USB mass storage (see Section 2.3.8: STEVAL-PROTEUS as Mass Storage Device).

To try this demo, please, follow the instructions listed below:

- Turn on the STEVAL-PROTEUS.
- Launch the STBLESensor app app on an Android or iOS-based mobile device.
- Wait until the app discovers the board.

Figure 24. **STBLESensor app discover STEVAL-PROTEUS running FP-SNS-DATAPRO1**



Available Boards

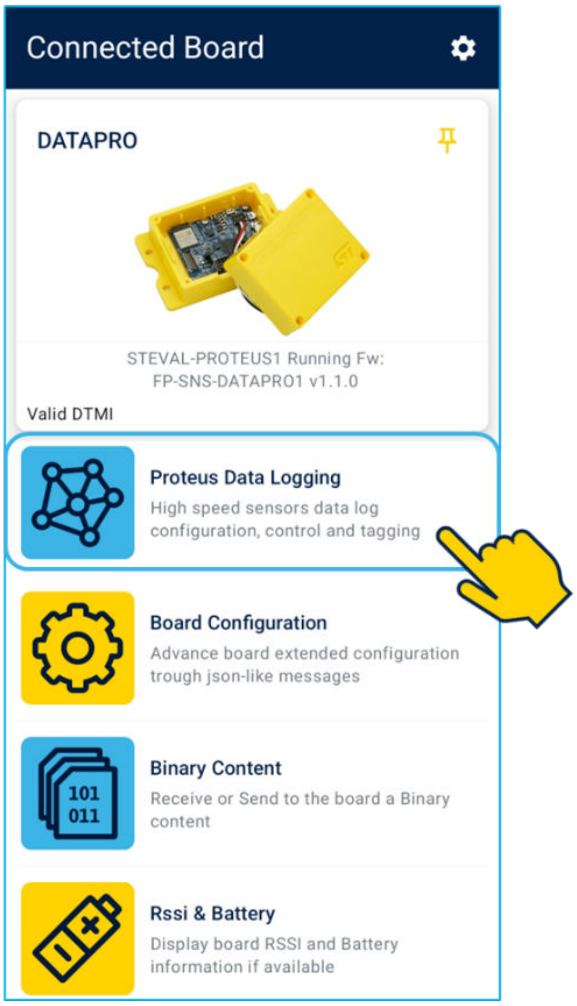
Welcome to ST BLE Sensor App!
Click on a board below to access its demos.

DATAPRO
STEVAL-PROTEUS1 Running Fw: FP-SNS-DATAPRO1 v1.1.0
Free space on FatFs volume: 252 MB
Battery Status: High
-55 dBm
00:80:E1:26:2F:45

Board alias
Board name and running firmware
Available space inside the FatFs volume
Battery status
Data logging status as icon
Catalog
Filter

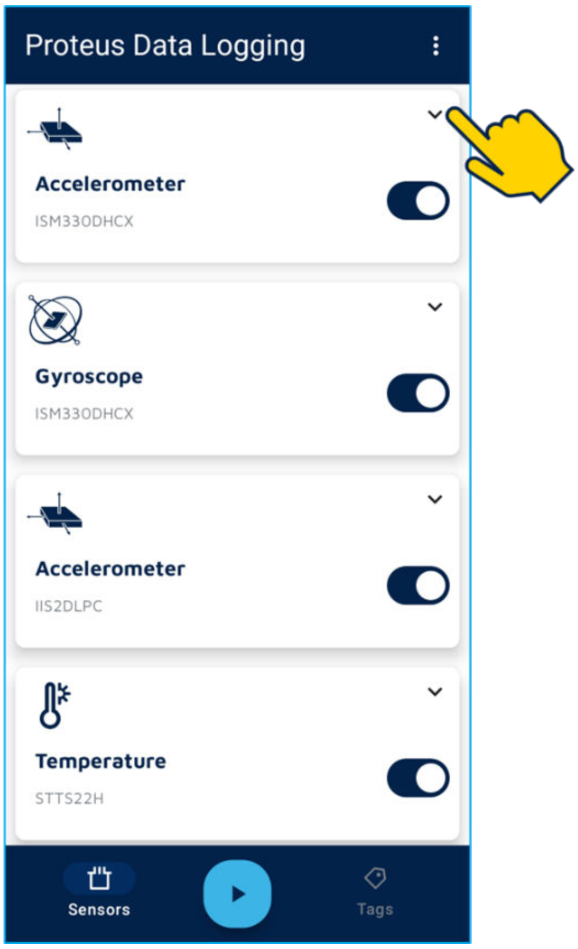
- Connect to the board.
- Tap *“Proteus Data Logging”*.

Figure 25. **FP-SNS-DATAPRO1 on STBLESensor app**


Connected Board
DATAPRO
STEVAL-PROTEUS1 Running Fw:
FP-SNS-DATAPRO1 v1.1.0
Valid DTMI
Proteus Data Logging
High speed sensors data log configuration, control and tagging
Board Configuration
Advance board extended configuration trough json-like messages
Binary Content
Receive or Send to the board a Binary content
Rssi & Battery
Display board RSSI and Battery information if available

The opened page shows the available sensors with a switch button nearby to use to disable/enable each one.

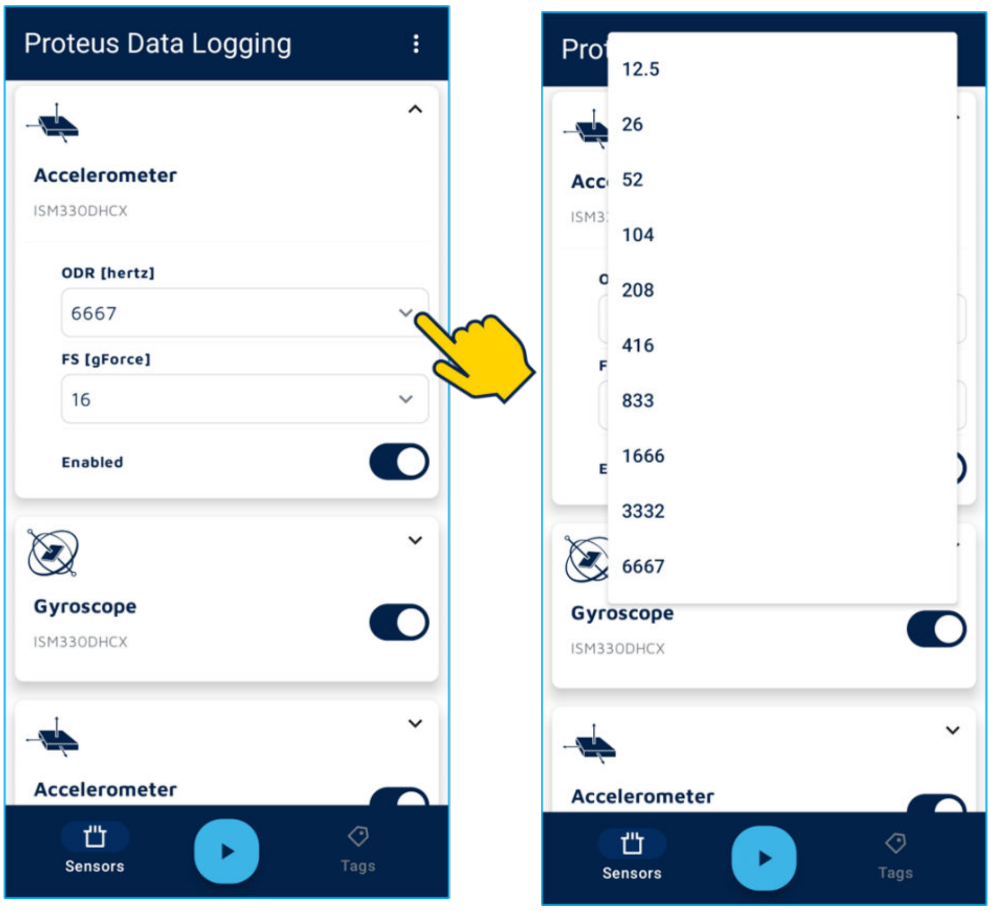
Figure 26. **Proteus Data Logging demo**



Tap the drop-down arrow, related to each sensor, to be able to configure the sensor in detail.

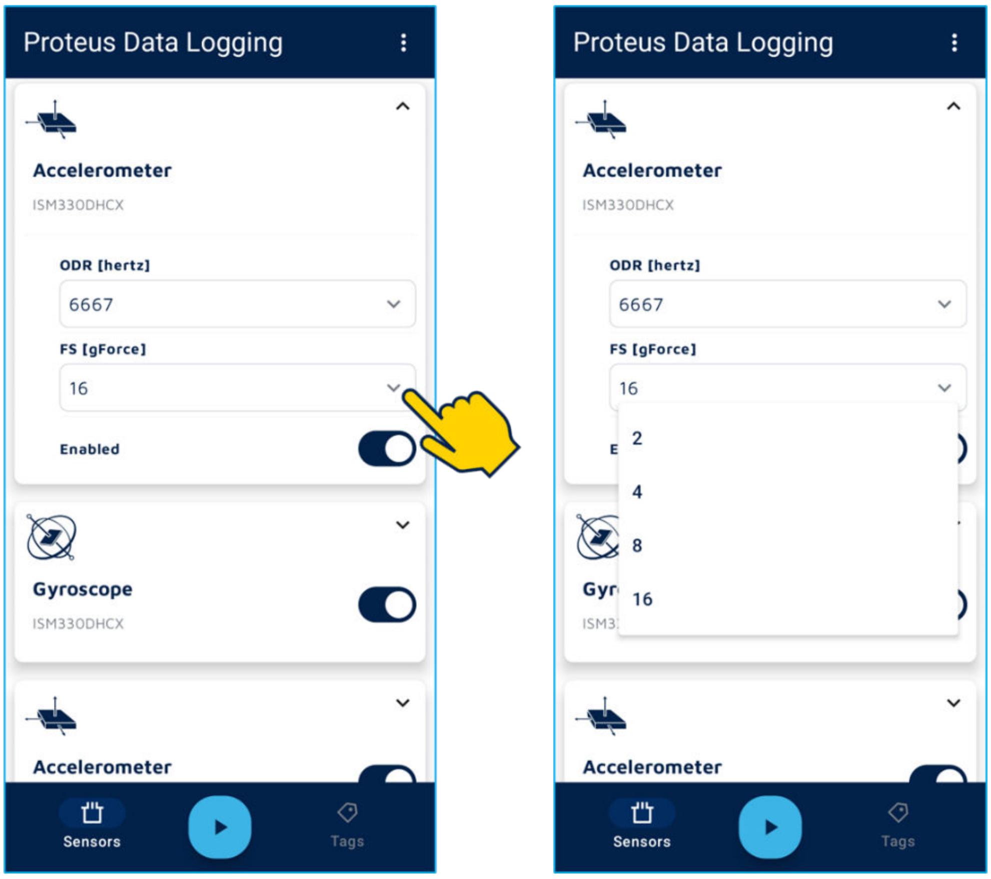
- You can configure the sensor output data rate (ODR):

Figure 27. Proteus Data Logging - ODR Sensor Settings



- You can configure the sensor full scale (FS):

Figure 28. **Proteus Data Logging - FS Sensor Settings**

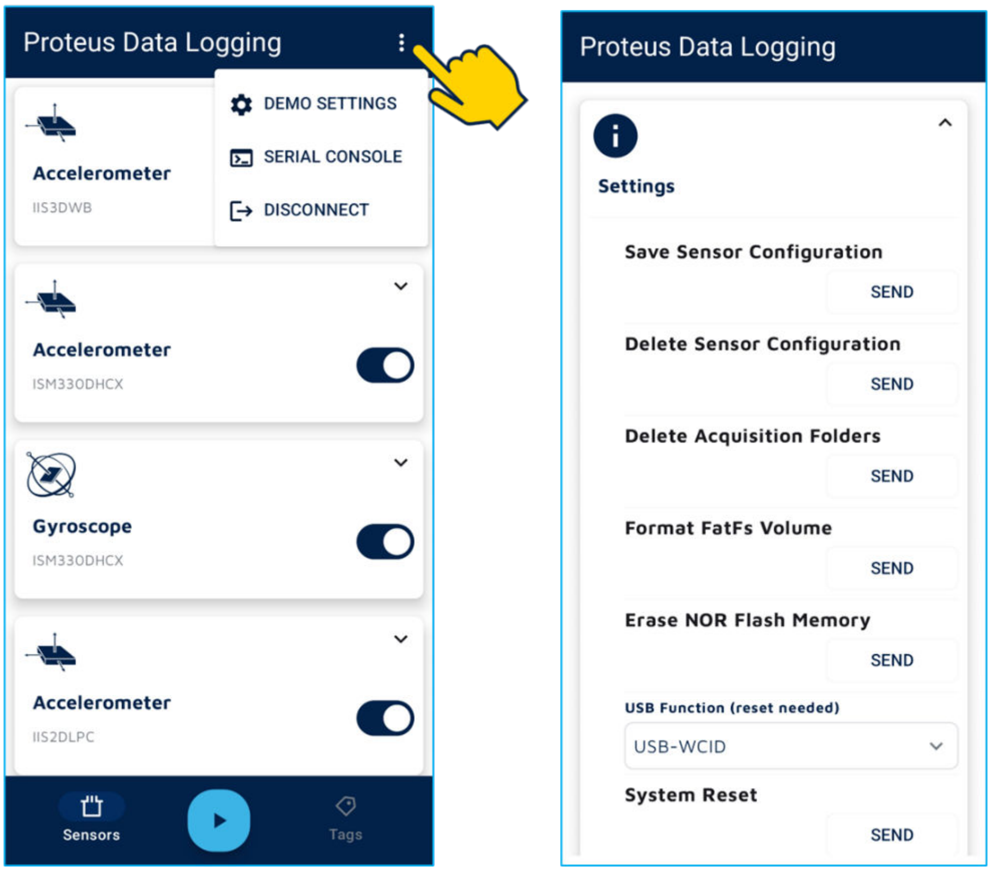


Tap the kebab menu, in top-right corner then tap *“Demo Settings”*.

Expand the *“Settings”* tab so you can:

- Save the current sensor configuration into a file located in the FatFs volume of the embedded NOR flash memory, *“[Drive]:\datalog\DeviceConfig.json”*. At each reboot, the sensors will be configured as described by this file. If the STEVAL-PROTEUS is connected to a computer as an USB as a mass storage, you can edit this file as described later.
- Delete the sensor configuration file *“[Drive]:\datalog\DeviceConfig.json”*.
- Delete all the acquisition folders already stored in the path *“[Drive]:\datalog\”*.
- Perform a quick format of the FatFs volume.
- Perform a complete NOR flash memory erasing and regenerate the FatFs volume.
- Set the USB function (i.e., WCID, Mass Storage, or none) to use after the next system reset.
- Perform a system reset.

Figure 29. Proteus Data Logging - Demo Settings



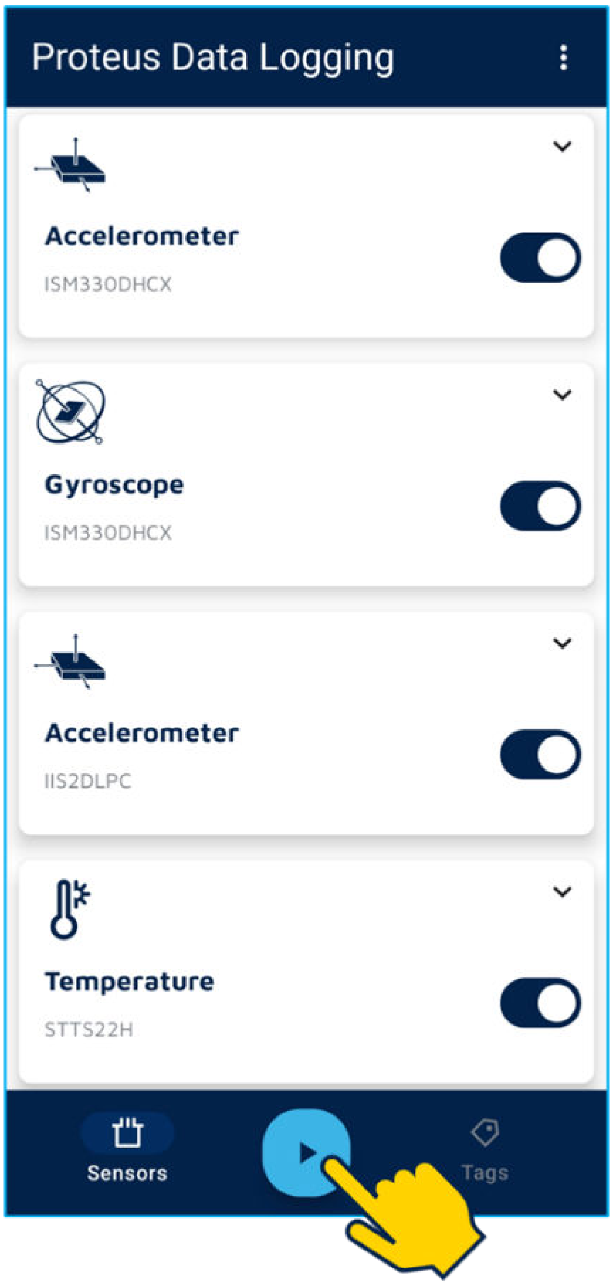
2.3.1.1 *User mode acquisition*

Once the sensors from which to acquire data have been configured, you can tap the “play” icon to begin the data logging.

Green LED will blink quickly.

From now on the RTC of the STEVAL-PROTEUS will be synchronized with the clock of the mobile device.

Figure 30. **Proteus Data Logging - Start**



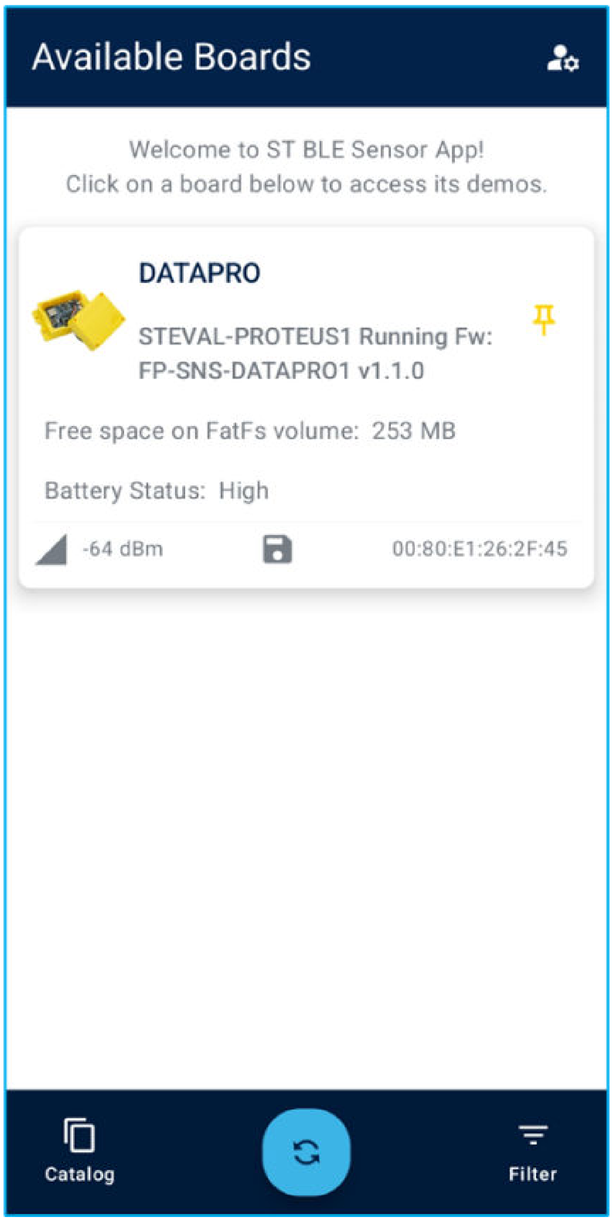
Data from the sensors are now being acquired. The data are stored in .dat files, one for each sensor and its sub-sensor (e.g., *“ISM330DHCX_ACC.dat” and “ISM330DHCX_GYRO.dat”*). It contains raw sensor data coupled with timestamps.

The files relating to this acquisition will be stored in the embedded NOR flash memory in the path "[Drive]:\datalog\" within a folder named in reference to the date and time of the acquisition (i.e., YYYYMMDD_HH_MM_SS).

During the data logging the BLE connection is not required, so you can disconnect your mobile device. The acquisition will stay on. In this case, when the STBLESensor app discover the STEVAL-PROTEUS a “disk” icon inform you that the data logging is on.

Please note that you will need to connect again to stop data logging.

Figure 31. **STBLESensor discovers the STEVAL-PROTEUS while data logging is on**



Available Boards

Welcome to ST BLE Sensor App!
Click on a board below to access its demos.

DATAPRO

STEVAL-PROTEUS1 Running Fw:
FP-SNS-DATAPRO1 v1.1.0

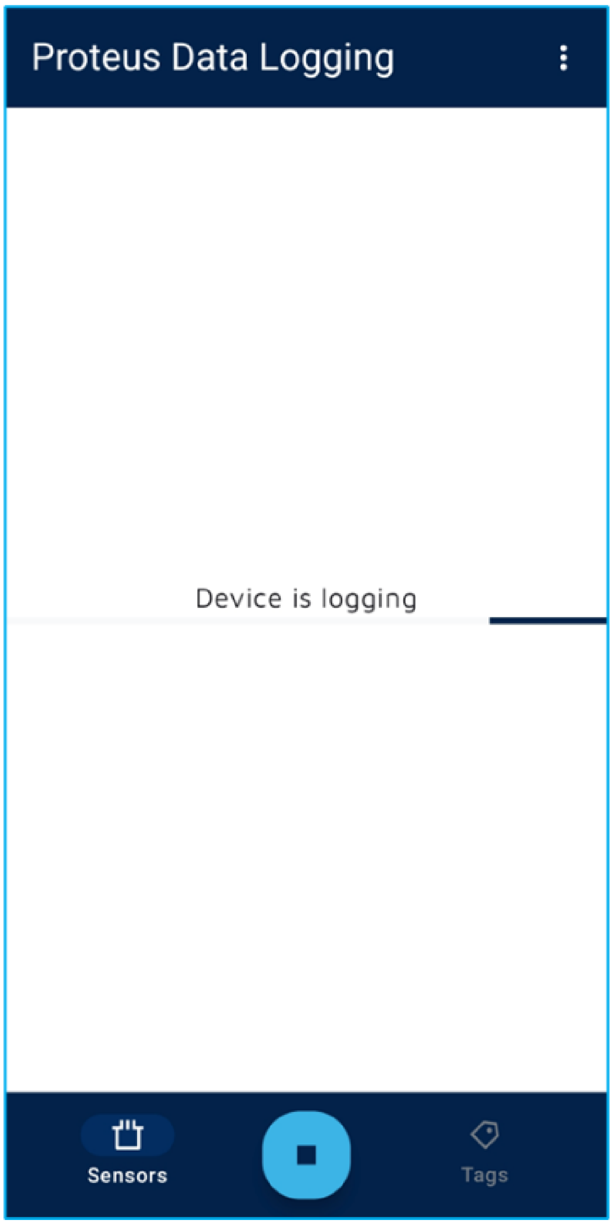
Free space on FatFs volume: 253 MB

Battery Status: High

-64 dBm 00:80:E1:26:2F:45

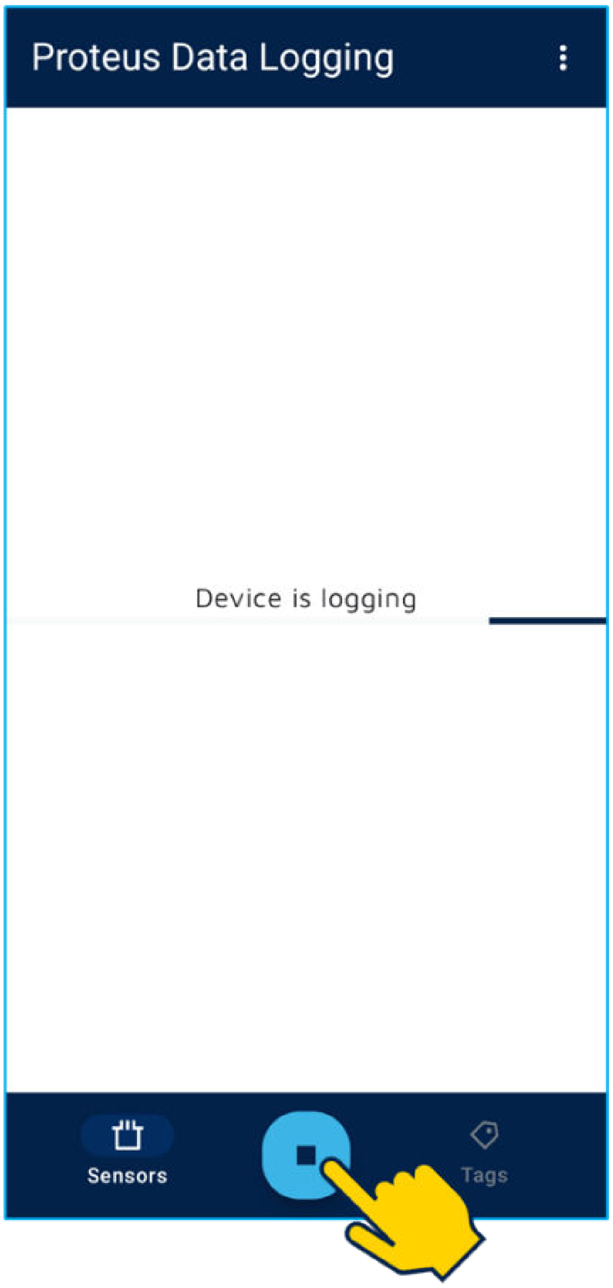
After the connection is established, the “Proteus Data Logging” demo shows the information “Device is logging”.

Figure 32. Proteus Data Logging - Device is logging



Tap the “stop” icon to end the data logging.
Green LED is turned off.

Figure 33. **Proteus Data Logging - Stop**



The .dat files are now being finalized.

After that, two important files will be added to the acquisition folder:

- *"AcquisitionInfo.json"*, that contains information about the acquisition.
- *"DeviceConfig.json"*, that contains specific information about the device configuration, necessary for correct data interpretation.

These files will be described in the paragraph Section 2.3.3: Datalog acquisition folder.

2.3.1.2 *Automode acquisition*

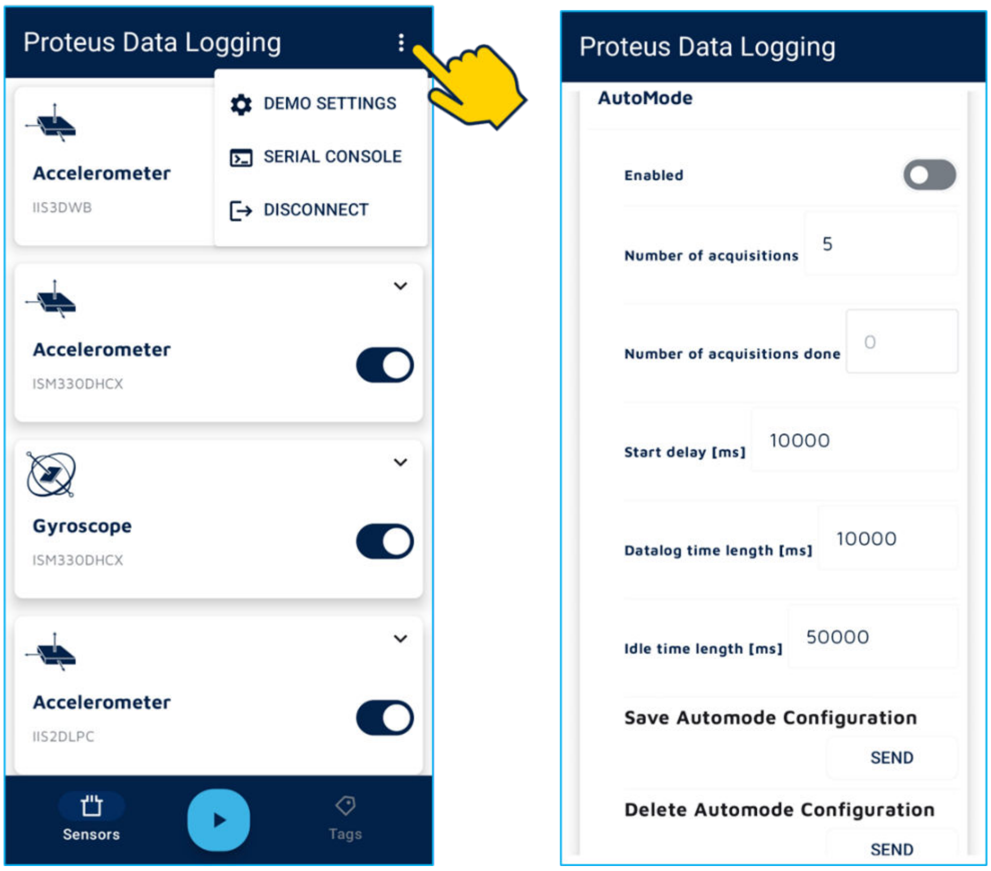
This mode gives the possibility to automatically perform a series of acquisitions. Two successive acquisitions are separated by idle time.

To use it you must act as explained at the beginning of the paragraph "Section 2.3.1: FP-SNS-DATAPRO1 stand-alone data logging" so the STEVAL-PROTEUS is already connected to the mobile device using the STBLESensor app and the "Proteus Data Logging" demo is active.

Tap the kebab menu, in top-right corner, to launch the *"Demo Settings"* menu than expand the *"AutoMode"* tab so you can:

- Enable or disable the auto mode.
- Set the number of acquisitions.
- Set the delay between starting auto mode and the first acquisition.
- Set the duration of each acquisition.
- Set the delay between two subsequent acquisitions.
- Save the current automode configuration into a file located in the FatFs volume of the embedded NOR flash memory, *"[Drive]:\datalog\AutomodeConfig.json"*. At each reboot, the automode will be configured as described by this file. If the STEVAL-PROTEUS is connected to a computer as an USB as a mass storage, you can edit this. Its structure is described in Figure 35. Proteus Data Logging - AutomodeConfig.json.
- Delete the automode configuration file *"[Drive]:\datalog\AutomodeConfig.json"*.

Figure 34. **Proteus Data Logging - Demo Settings**



Enabled
Number of acquisitions 5
Number of acquisitions done 0
Start delay [ms] 10000
Datalog time length [ms] 10000
Idle time length [ms] 50000

Figure 35. **Proteus Data Logging - AutomodeConfig.json**

```
{
    "AutomodeModel": [
        {
            "Enabled": true,
            "NrOfAcquisition": 5,
            "StartDelay": 10000,
            "LoggingPeriod": 10000,
            "IdlePeriod": 50000
        }
    ]
}
```

Since the automode has been configured and enabled via app or by editing the ‘AutomodeConfig.json’ into the embedded FatFs volume (take care that any changes will be loaded after a system reset), you can start the acquisition series as described for *“Section 2.3.1.1: User mode acquisition”*.

The demo will show the information “Device is logging” until the acquisition series is completed.

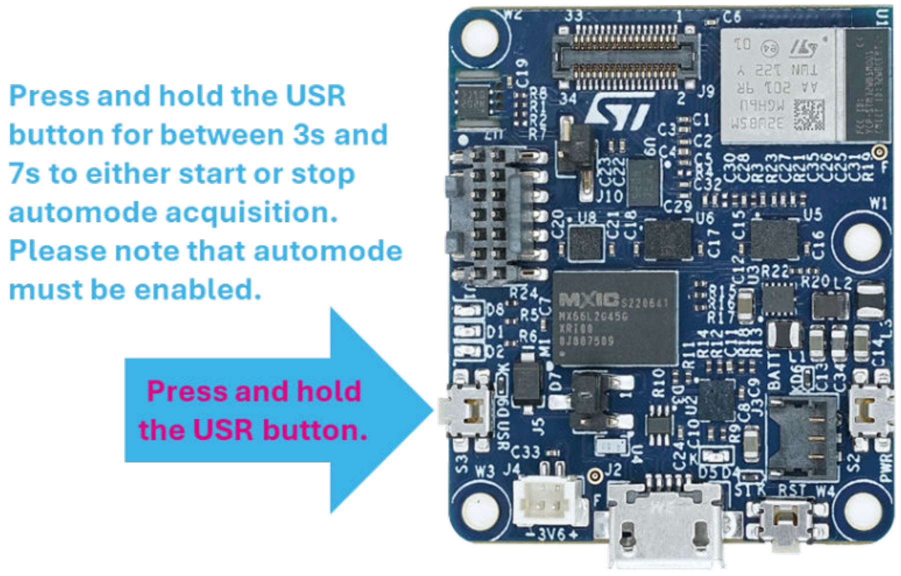
Like the user mode acquisition, BLE connection is not required, so you can disconnect your mobile device.

The acquisition series will continue as configured and will end as soon as the last acquisition has been performed.

If you need to stop the series of acquisitions before it finishes, while the board is connected to the mobile device with the app active on the ‘Proteus Data Logging” demo, tap “stop” icon.

If the automode is enabled, better via the file *“[Drive]:\datalog\AutomodeConfig.json”*, you can start the automode acquisition by holding down the user button for between 3s and 7s to . Now, if necessary, you can do the same to stop the acquisition early.

Figure 36. **Proteus Data Logging – Start-Stop automode by push button**

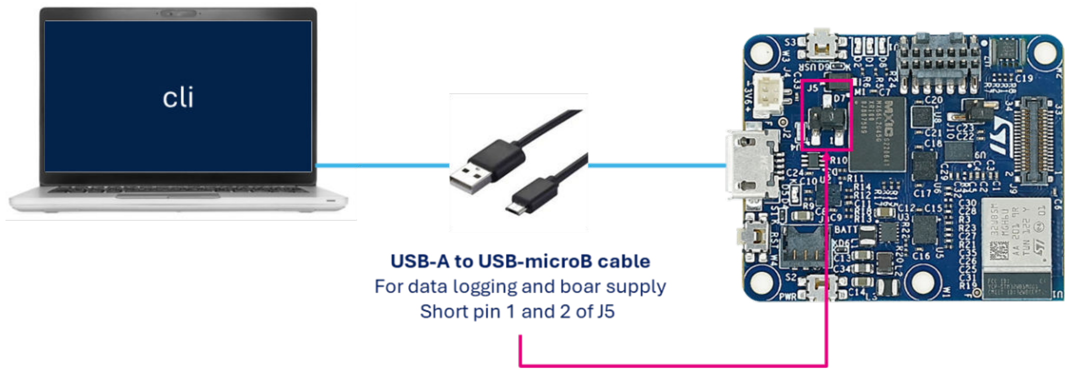

Press and hold the USR button for between 3s and 7s to either start or stop automode acquisition. Please note that automode must be enabled.
Press and hold the USR button.

2.3.2 FP-SNS-DATAPRO1 USB WCID data logging

When the STEVAL-PROTEUS runs the FP-SNS-DATAPRO1 and exposes USB as WCID functionality (it is by default), you can capture streaming data using different hosts based on Windows or Linux, both with 64-bit architecture. You can also use a single-board computer (SBC) like Raspberry Pi, with 32-bit architecture.

In any case, once you plug the board to your host via USB with the Operating System should recognize the STEVAL-PROTEUS as a new USB device named *“STEVAL-PROTEUS WCID-Streaming”* and automatically install the required drivers.

Figure 37. How to connect STEVAL-PROTEUS to a PC for USB WCID data logging



2.3.2.1 *cli_example*

Depending on the used operating system, it is necessary to prepare your host.

- Windows:
  - Nothing to do.
- Linux:
  - Launch a terminal.
  - Go to the folder: *“Utilities/SwUtilities/HS_Datalog/cli_example/linux_setup”*.
  - Execute the bash script: *“bash setup.sh”*.
  - Hereafter, you can execute the command *“bash udev_removal.sh”* to undo what you did with the setup script.
- Raspberry Pi (or other similar SBC):
  - Launch a terminal.
  - Go to the folder: *“Utilities/SwUtilities/HS_Datalog/cli_example/raspberry_setup”*.
  - Execute the bash script: *“bash setup_rpi3b.sh”*.
  - Hereafter, you can execute the command *“bash udev_removal_rpi3b.sh”* to undo what you did with the setup script.

Remember: To avoid any possible issues while executing the script in Linux-based environment, it is recommended to use dos2unix to properly reformat your files. So, install it on your system: “sudo apt-get install dos2unix”.

After that you can evaluate the data logging via USB WCID with an STEVAL-PROTEUS running the FP-SNS-DATAPRO1.

- Connected the STEVAL-PROTEUS to a PC through a USB-A to USB-microB cable (see Figure 37. How to connect STEVAL-PROTEUS to a PC for USB WCID data logging).
- Launch a terminal.
- Go to the folder: *“Utilities/SwUtilities/HS_Datalog/cli_example”*.
- Execute the script related to the used OS (for Linux or Raspberry, launch the scripts as bash):
  - USB_DataLog_Run.bat (for Windows)
  - USB_DataLog_Run.sh (for Linux)
  - USB_DataLog_Run_rpi3.sh (for Raspberry Pi)

Use text editor software to understand how the script works and edit it to fit your needs.

Figure 38. **CLI script for USB WCID data logging**

```
@echo off
REM -f <filename>: Device Configuration file (JSON)
REM -t <seconds>: Duration of the current acquisition (seconds)
.\bin\cli_example.exe -f ..\PROTEUS config examples\ALL SENSORS.json -t 20
pause
```

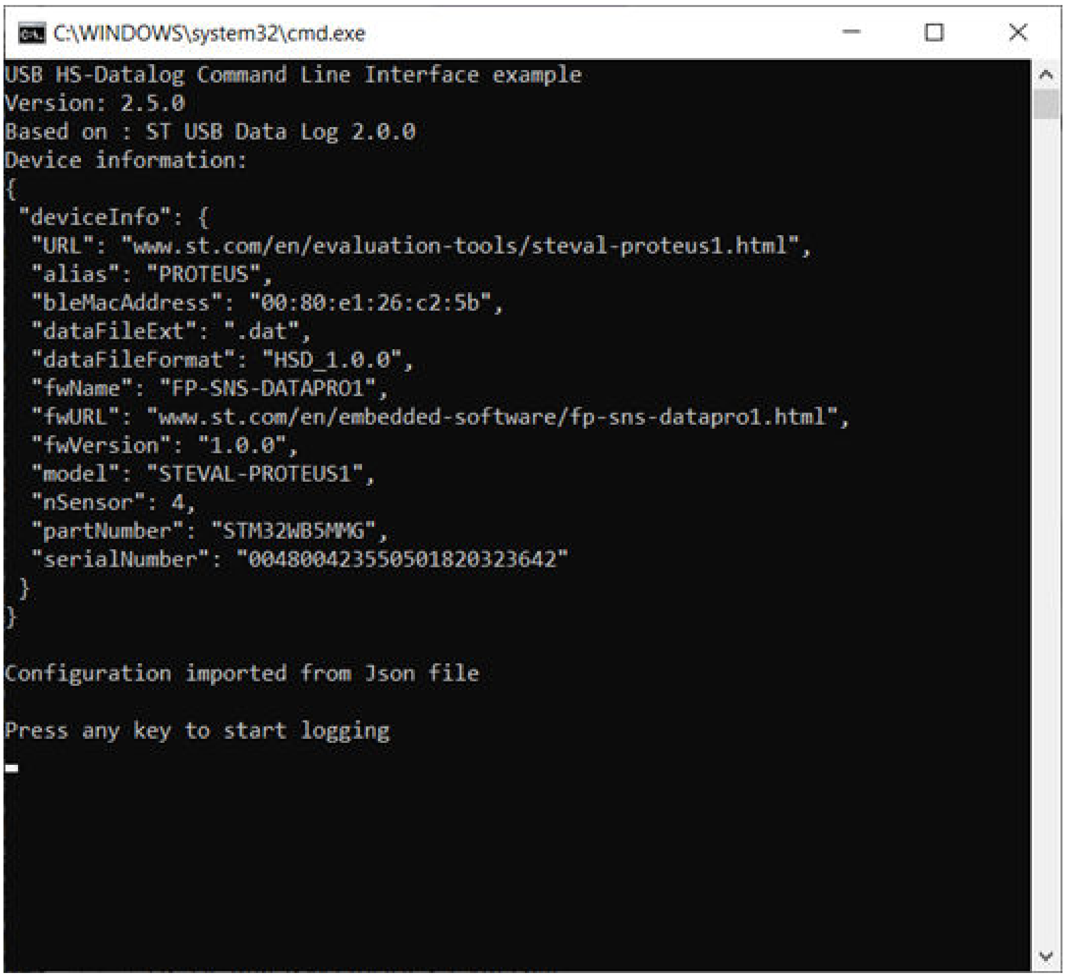
It calls the CLI program and sets two parameters, the device configuration file (-f) and the acquisition duration (-t).

The folder *"Utilities\SwUtilities\HS_Datalog\cli_example\bin"* contains the CLI software program.

The folder *"Utilities\SwUtilities\HS_Datalog\PROTEUS_config_examples"* contains a lot of device configuration files.

The folder *"Utilities\SwUtilities\HS_Datalog\PROTEUS_batch_file_examples"* contains a lot of batch files related to the corresponding configuration files (only for Windows OS).

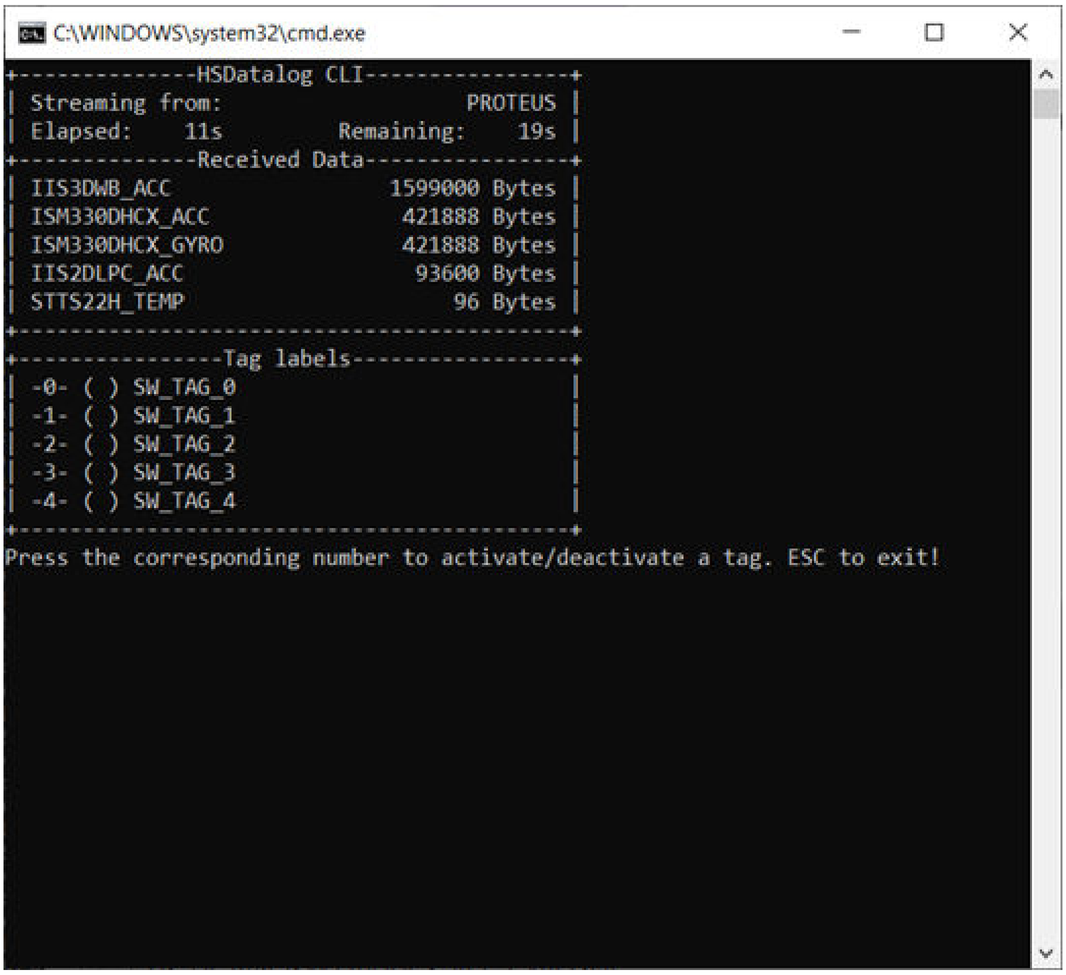
Figure 39. **USB WCID datalog CLI ready to start**



```
USB HS-Datalog Command Line Interface example
Version: 2.5.0
Based on : ST USB Data Log 2.0.0
Device information:
{
 "deviceInfo": {
  "URL": "www.st.com/en/evaluation-tools/steval-proteus1.html",
  "alias": "PROTEUS",
  "bleMacAddress": "00:80:e1:26:c2:5b",
  "dataFileExt": ".dat",
  "dataFileFormat": "HSD_1.0.0",
  "fwName": "FP-SNS-DATAPRO1",
  "fwURL": "www.st.com/en/embedded-software/fp-sns-datapro1.html",
  "fwVersion": "1.0.0",
  "model": "STEVAL-PROTEUS1",
  "nSensor": 4,
  "partNumber": "STM32WB5MMG",
  "serialNumber": "004800423550501820323642"
 }
}

Configuration imported from Json file

Press any key to start logging
```

Now press *"Return"* key.

Figure 40. **USB WCID datalog CLI acquisition running**



C:\WINDOWS\system32\cmd.exe

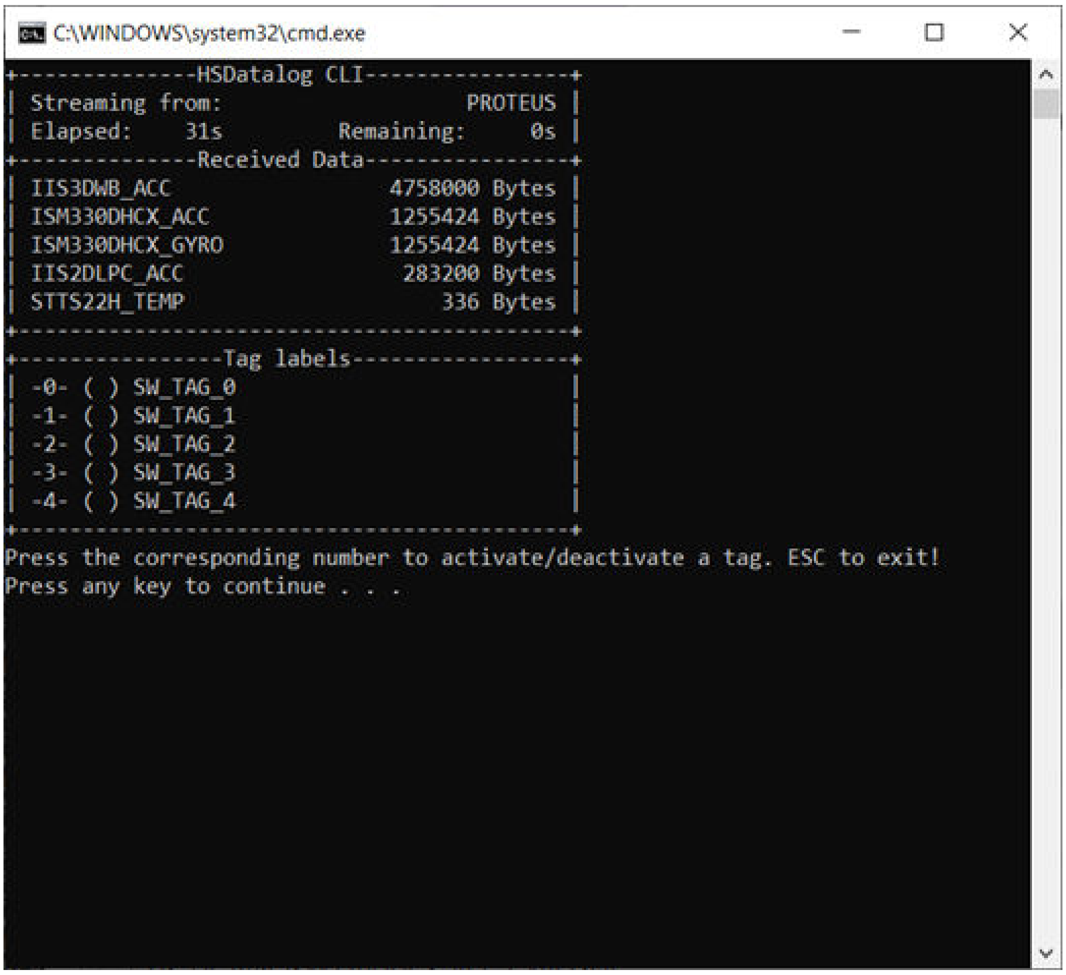
```
+--------------HSDatalog CLI----------------+
| Streaming from:                   PROTEUS |
| Elapsed:     11s         Remaining:    19s |
+--------------Received Data----------------+
| IIS3DWB_ACC                 1599000 Bytes |
| ISM330DHCX_ACC               421888 Bytes |
| ISM330DHCX_GYRO              421888 Bytes |
| IIS2DLPC_ACC                  93600 Bytes |
| STTS22H_TEMP                     96 Bytes |
+-------------------------------------------+
+----------------Tag labels-----------------+
| -0- ( ) SW_TAG_0                          |
| -1- ( ) SW_TAG_1                          |
| -2- ( ) SW_TAG_2                          |
| -3- ( ) SW_TAG_3                          |
| -4- ( ) SW_TAG_4                          |
+-------------------------------------------+
Press the corresponding number to activate/deactivate a tag. ESC to exit!
```

Data from the sensors are now being acquired. The data are stored in .dat files, one for each sensor and its sub-sensor (e.g., *"ISM330DHCX_ACC.dat"*). It contains raw sensor data coupled with timestamps.

The files relating to this acquisition will be stored in the same folder of the script but into a folder named in reference to the date and time of the acquisition (i.e., YYYYMMDD_HH_MM_SS).

The acquisition will finish after a configured duration time or by pressing *"Esc"* button.

Figure 41. **USB WCID datalog CLI acquisition completed**



C:\WINDOWS\system32\cmd.exe

```
+--------------HSDatalog CLI---------------+
| Streaming from:                  PROTEUS |
| Elapsed:     31s         Remaining:     0s |
+--------------Received Data---------------+
| IIS3DWB_ACC                 4758000 Bytes |
| ISM330DHCX_ACC              1255424 Bytes |
| ISM330DHCX_GYRO             1255424 Bytes |
| IIS2DLPC_ACC                 283200 Bytes |
| STTS22H_TEMP                    336 Bytes |
+------------------------------------------+
+----------------Tag labels----------------+
| -0-  ( ) SW_TAG_0                         |
| -1-  ( ) SW_TAG_1                         |
| -2-  ( ) SW_TAG_2                         |
| -3-  ( ) SW_TAG_3                         |
| -4-  ( ) SW_TAG_4                         |
+------------------------------------------+
Press the corresponding number to activate/deactivate a tag. ESC to exit!
Press any key to continue . . .
```

The .dat files are now being finalized.

After that, two important files will be added to the acquisition folder:

- *"AcquisitionInfo.json"*, that contains information about the acquisition.
- *"DeviceConfig.json"*, that contains specific information about the device configuration, necessary for correct data interpretation.

These files will be described later.

2.3.3 Datalog acquisition folder

As described above, each acquisition, both stand-alone and USB WCID, generates a folder containing the following files:

- Datalog files
- AcquisitionInfo.json
- DeviceConfig.json

2.3.3.1 *Datalog file*

It contains raw sensor data coupled with timestamps. A .dat file will be generated for each active sensor and its sub-sensor (e.g., "ISM330DHCX_ACC.dat").

The name of the file describes the sensor part number and the sensor type, as:

"<sensor_name>_<subsensor_type>.dat"

where:

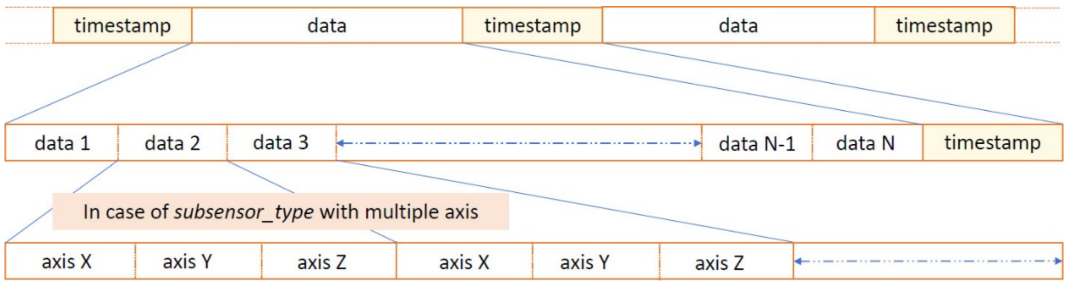
- <sensor_name>: component part number

- <subsensor_type>: ACC, GYRO, TEMP

The datalog file contains sensor raw data and timestamps. The file starts with data.

The following figure describes the structure of this file.

Figure 42. Datalog file structure



where:

- The k-th data (k = 1 ... N) represents the raw data generated by a sub-sensor of the sensor. If the sub-sensor has multiple axis, such as acceleration sensor (e.g., ISM330DHCX), each k-th data is a sequence of raw data for each axis (e.g., | axis X | axis Y | axis Z |).
- N is the number of samples per timestamp.
- Timestamp is expressed in second and stored as a double, 8-byte floating point number.

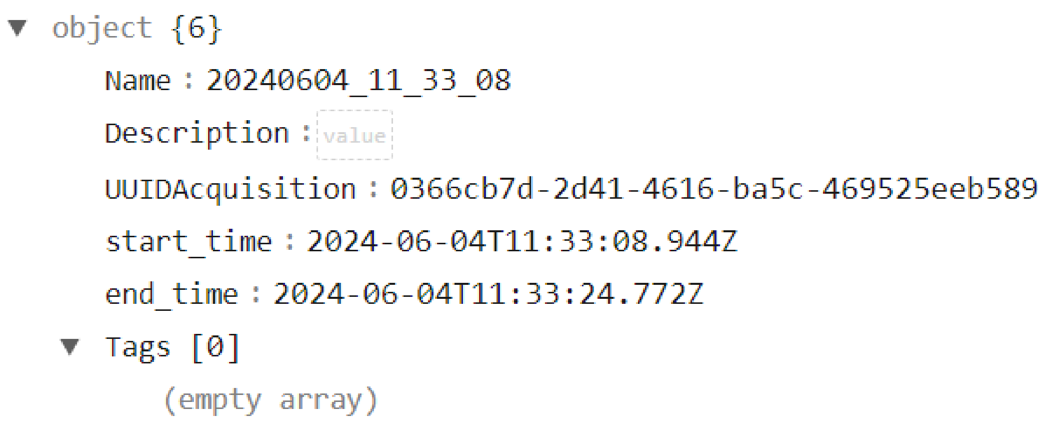
The datalog file can be decoded using the *"DeviceConfig.json"* file in particular referring to the section for the sub-sensor status related to the sensor and its sub-sensor.

2.3.3.2 *AcquisitionInfo.json*

It contains information about the acquisition.

Its content is a complex data structure written respecting the syntax of the JavaScript Object Notation (JSON).

Figure 43. AcquisitionInfo.json



```
▼ object {6}
      Name : 20240604_11_33_08
      Description : value
      UUIDAcquisition : 0366cb7d-2d41-4616-ba5c-469525eeb589
      start_time : 2024-06-04T11:33:08.944Z
      end_time : 2024-06-04T11:33:24.772Z
   ▼ Tags [0]
         (empty array)
```

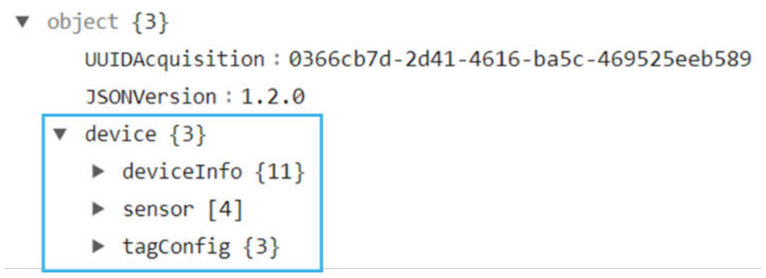
2.3.3.3 *DeviceConfig.json*

It contains specific information about the device configuration, necessary for correct data interpretation as shown below.

Its content is a complex data structure written respecting the syntax of the JavaScript Object Notation (JSON).

"device" is described as following:

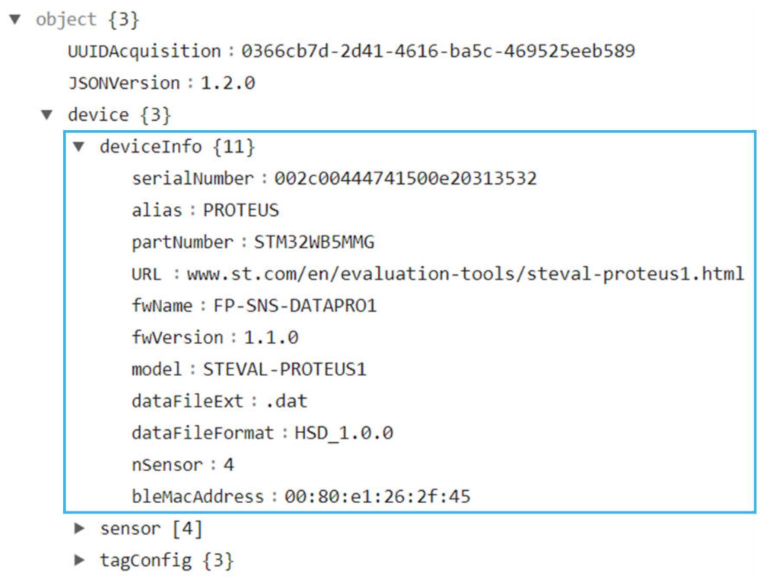
Figure 44. DeviceConfig.json - device



```
▼ object {3}
    UUIDAcquisition : 0366cb7d-2d41-4616-ba5c-469525eeb589
    JSONVersion : 1.2.0
  ▼ device {3}
    ▶ deviceInfo {11}
    ▶ sensor [4]
    ▶ tagConfig {3}
```

"deviceInfo" identifies the device.

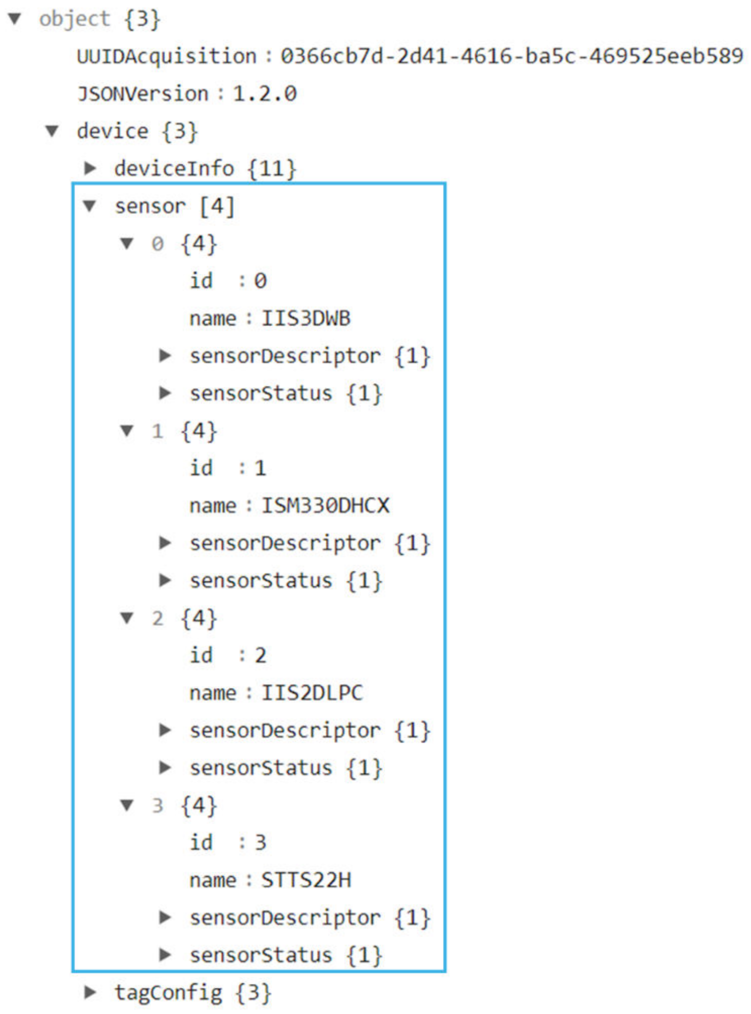
Figure 45. DeviceConfig.json - device info



```
▼ object {3}
    UUIDAcquisition : 0366cb7d-2d41-4616-ba5c-469525eeb589
    JSONVersion : 1.2.0
  ▼ device {3}
    ▼ deviceInfo {11}
        serialNumber : 002c00444741500e20313532
        alias : PROTEUS
        partNumber : STM32WB5MMG
        URL : www.st.com/en/evaluation-tools/steval-proteus1.html
        fwName : FP-SNS-DATAPRO1
        fwVersion : 1.1.0
        model : STEVAL-PROTEUS1
        dataFileExt : .dat
        dataFileFormat : HSD_1.0.0
        nSensor : 4
        bleMacAddress : 00:80:e1:26:2f:45
    ▶ sensor [4]
    ▶ tagConfig {3}
```

"sensor" is an array containing the description of each sensor available on the board.

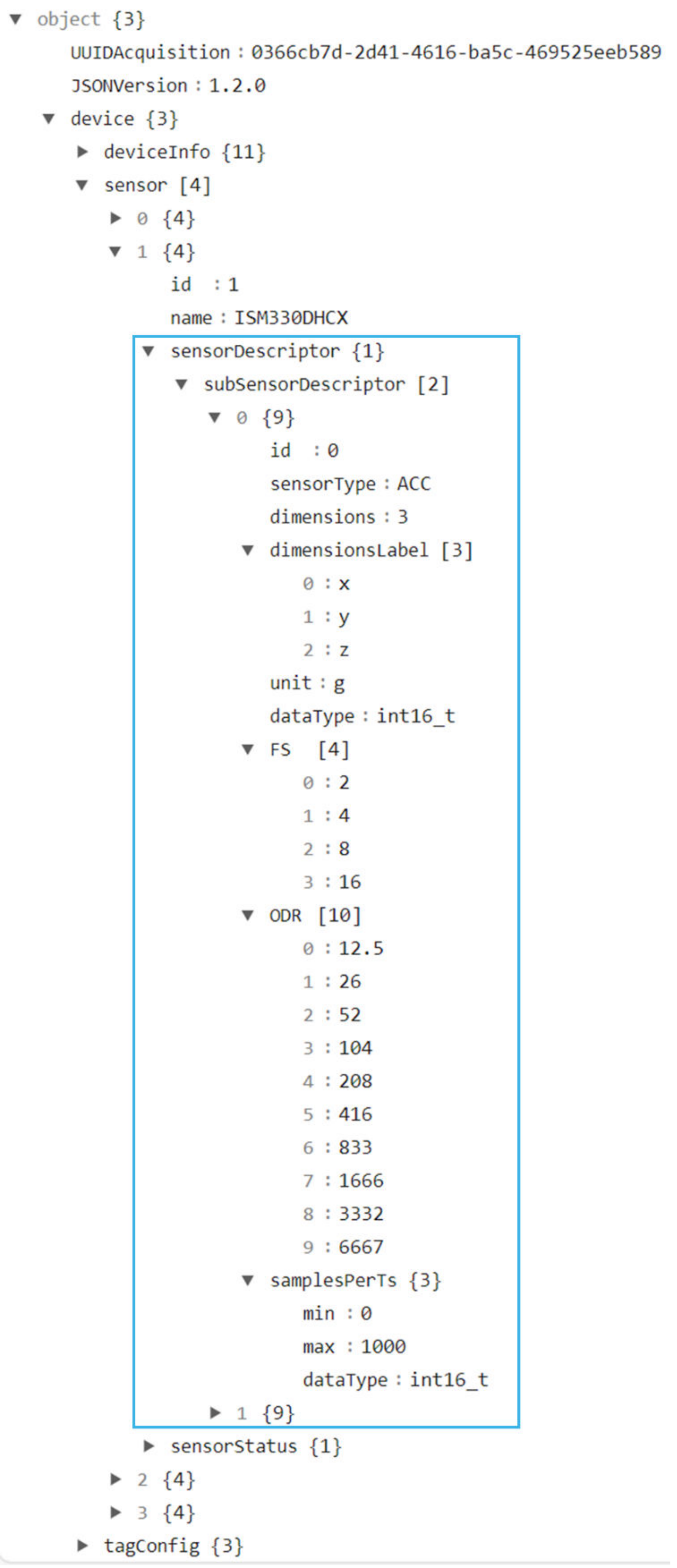
Figure 46. DeviceConfig.json - device\sensor



```
▼ object {3}
    UUIDAcquisition : 0366cb7d-2d41-4616-ba5c-469525eeb589
    JSONVersion : 1.2.0
  ▼ device {3}
    ▶ deviceInfo {11}
    ▼ sensor [4]
      ▼ 0 {4}
          id  : 0
          name : IIS3DWB
        ▶ sensorDescriptor {1}
        ▶ sensorStatus {1}
      ▼ 1 {4}
          id  : 1
          name : ISM330DHCX
        ▶ sensorDescriptor {1}
        ▶ sensorStatus {1}
      ▼ 2 {4}
          id  : 2
          name : IIS2DLPC
        ▶ sensorDescriptor {1}
        ▶ sensorStatus {1}
      ▼ 3 {4}
          id  : 3
          name : STTS22H
        ▶ sensorDescriptor {1}
        ▶ sensorStatus {1}
    ▶ tagConfig {3}
```

"sensorDescriptor" describes the main information about the sensor through the list of its "*subSensorDescriptor*". Each element of "*subSensorDescriptor"* describes the main information about each sub-sensor.

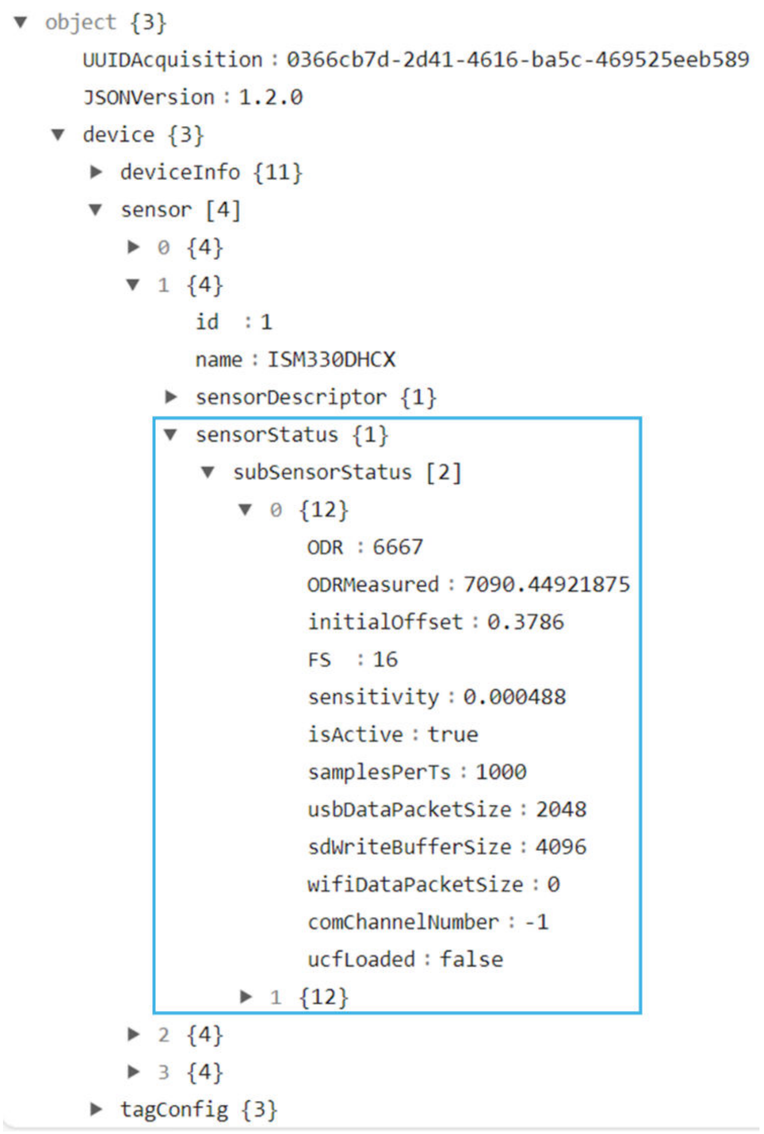
Figure 47. DeviceConfig.json - sensor\sensorDescriptor



```
▼ object {3}
    UUIDAcquisition : 0366cb7d-2d41-4616-ba5c-469525eeb589
    JSONVersion : 1.2.0
  ▼ device {3}
    ▶ deviceInfo {11}
    ▼ sensor [4]
      ▶ 0 {4}
      ▼ 1 {4}
          id  : 1
          name : ISM330DHCX
        ▼ sensorDescriptor {1}
          ▼ subSensorDescriptor [2]
            ▼ 0 {9}
                id  : 0
                sensorType : ACC
                dimensions : 3
              ▼ dimensionsLabel [3]
                  0 : x
                  1 : y
                  2 : z
                unit : g
                dataType : int16_t
              ▼ FS   [4]
                  0 : 2
                  1 : 4
                  2 : 8
                  3 : 16
              ▼ ODR  [10]
                  0 : 12.5
                  1 : 26
                  2 : 52
                  3 : 104
                  4 : 208
                  5 : 416
                  6 : 833
                  7 : 1666
                  8 : 3332
                  9 : 6667
              ▼ samplesPerTs {3}
                  min  : 0
                  max  : 1000
                  dataType : int16_t
            ▶ 1 {9}
        ▶ sensorStatus {1}
      ▶ 2 {4}
      ▶ 3 {4}
    ▶ tagConfig {3}
```

“sensorStatus” describes the actual configuration of the sensor through the list of its *“subSensorStatus”*. Each element of *“subSensorStatus”* describes the actual configuration of each sub-sensor.

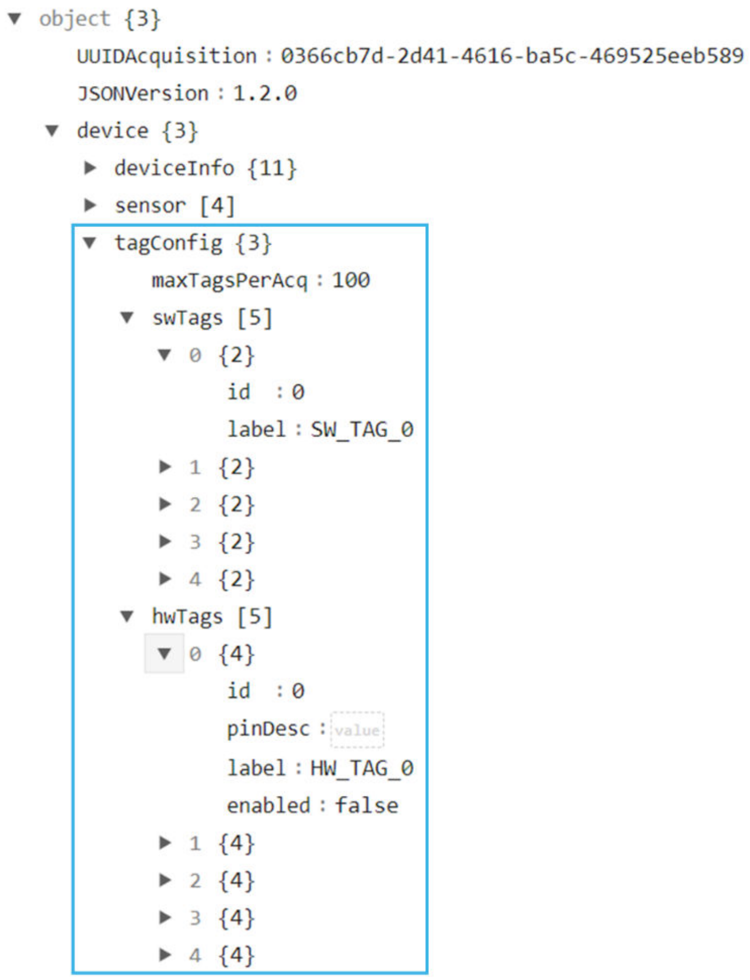
Figure 48. **DeviceConfig.json - sensor\sensorStatus**



```
▼ object {3}
    UUIDAcquisition : 0366cb7d-2d41-4616-ba5c-469525eeb589
    JSONVersion : 1.2.0
  ▼ device {3}
    ▶ deviceInfo {11}
    ▼ sensor [4]
      ▶ 0 {4}
      ▼ 1 {4}
          id  : 1
          name : ISM330DHCX
        ▶ sensorDescriptor {1}
        ▼ sensorStatus {1}
          ▼ subSensorStatus [2]
            ▼ 0 {12}
                ODR : 6667
                ODRMeasured : 7090.44921875
                initialOffset : 0.3786
                FS  : 16
                sensitivity : 0.000488
                isActive : true
                samplesPerTs : 1000
                usbDataPacketSize : 2048
                sdWriteBufferSize : 4096
                wifiDataPacketSize : 0
                comChannelNumber : -1
                ucfLoaded : false
            ▶ 1 {12}
      ▶ 2 {4}
      ▶ 3 {4}
    ▶ tagConfig {3}
```

“tagConfig” describes the labels activated by the user.

Figure 49. **DeviceConfig.json - device\tagConfig**



```
▼ object {3}
    UUIDAcquisition : 0366cb7d-2d41-4616-ba5c-469525eeb589
    JSONVersion : 1.2.0
  ▼ device {3}
    ▶ deviceInfo {11}
    ▶ sensor [4]
    ▼ tagConfig {3}
        maxTagsPerAcq : 100
      ▼ swTags [5]
        ▼ 0 {2}
            id  : 0
            label : SW_TAG_0
        ▶ 1 {2}
        ▶ 2 {2}
        ▶ 3 {2}
        ▶ 4 {2}
      ▼ hwTags [5]
        ▼ 0 {4}
            id  : 0
            pinDesc : value
            label : HW_TAG_0
            enabled : false
        ▶ 1 {4}
        ▶ 2 {4}
        ▶ 3 {4}
        ▶ 4 {4}
```

2.3.4 Analyze the acquired data log

You can use the *"HSDPython_SDK"* to analyze the data of a previous acquisition. It contains different Python scripts and classes that can be used to handle the datasets.

Depending on the used operating system, it is necessary to prepare your host.

- Windows:
  - Go to the folder: *"Utilities\SwUtilities\HS_Datalog\HSDPython_SDK"*.
  - Execute the batch file: *"HSDPython_SDK_install_noGUI.bat"*.
- Linux:
  - Launch a terminal.
  - Go to the folder: *"Utilities\SwUtilities\HS_Datalog\HSDPython_SDK"*.
  - Execute the bash script: *"bash HSDPython_SDK_install_noGUI.sh"*.
- Raspberry Pi (or other similar SBC):
  - No support

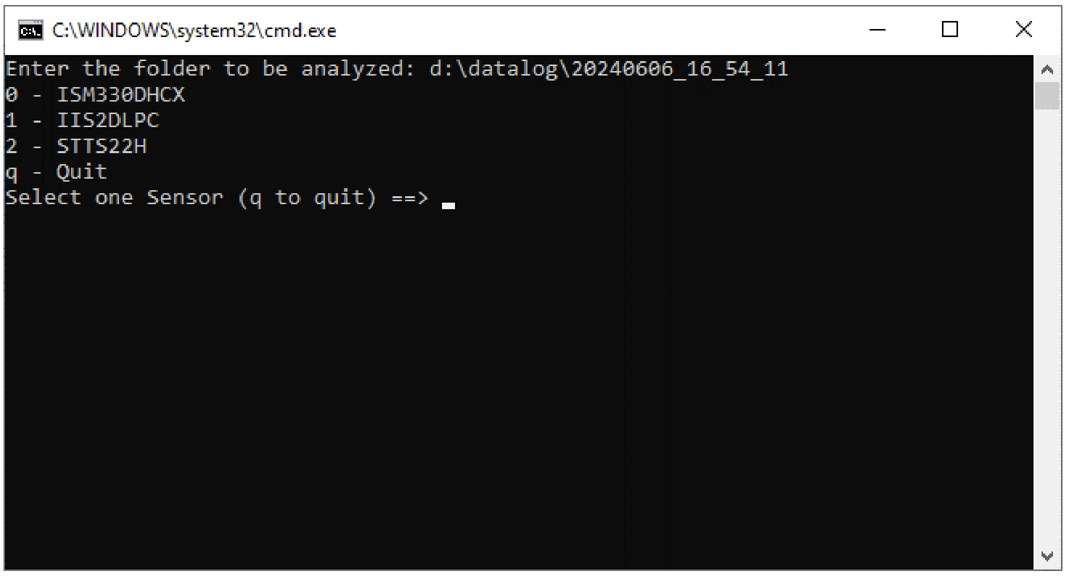
Remember: *To avoid any possible issues while executing the script in Linux-based environment, it is recommended to use dos2unix to properly reformat your files. So, install it on your system: "sudo apt-get install dos2unix".*

After that you can, for example, get a graph of each sensor data log in the dataset.

- Go to the folder: *"Utilities\SwUtilities\HS_Datalog\PROTEUS_batch_file_examples"*.

- Execute the script related to the used OS (for Linux, launch the scripts as bash):
  – HS-DL_Plot.bat (for Windows)
  – HS-DL_Plot.sh (for Linux)
- A command console will open, and you can enter the path to the acquisition folder that you will analyze.

Figure 50. **hsdatalog_plot.py shell**



```
Enter the folder to be analyzed: d:\datalog\20240606_16_54_11
0 - ISM330DHCX
1 - IIS2DLPC
2 - STTS22H
q - Quit
Select one Sensor (q to quit) ==> _
```

- Then, enter the number for the sensor (e.g., "0" for ISM330DHCX) whose data you want to graph.

Figure 51. **ISM330DHCX-ACC data plot example**

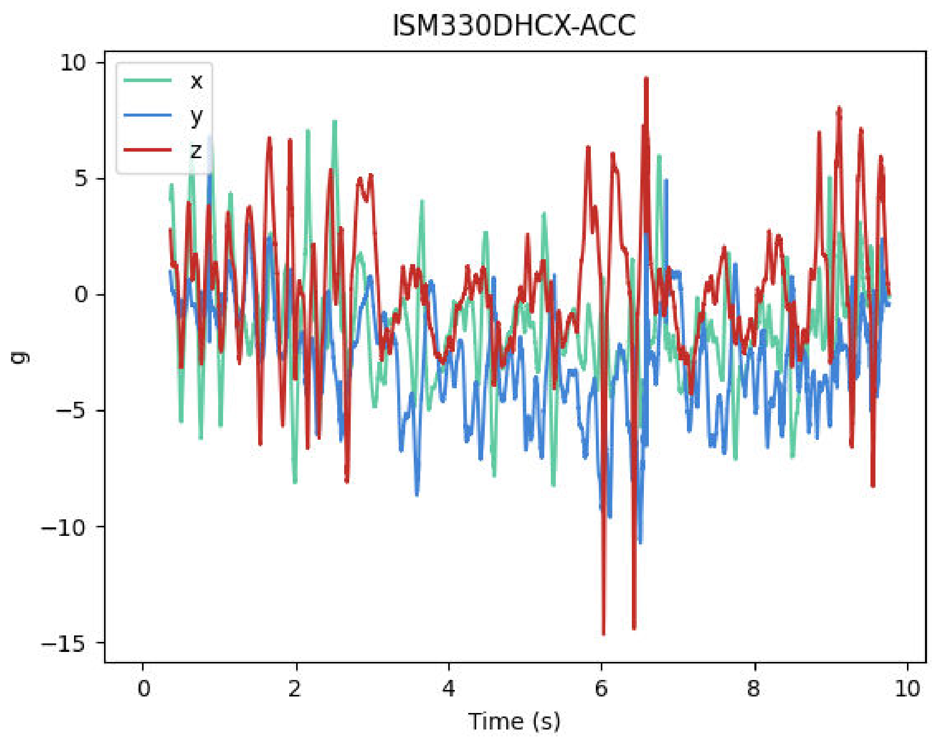


Figure 52. **ISM330DHCX-GYRO data plot example**

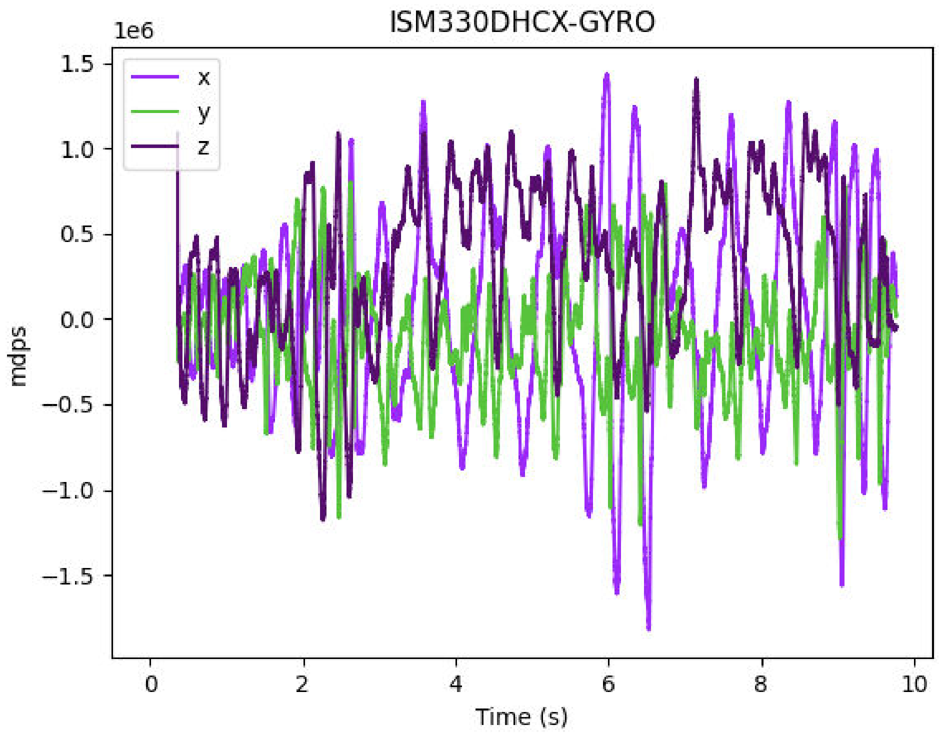
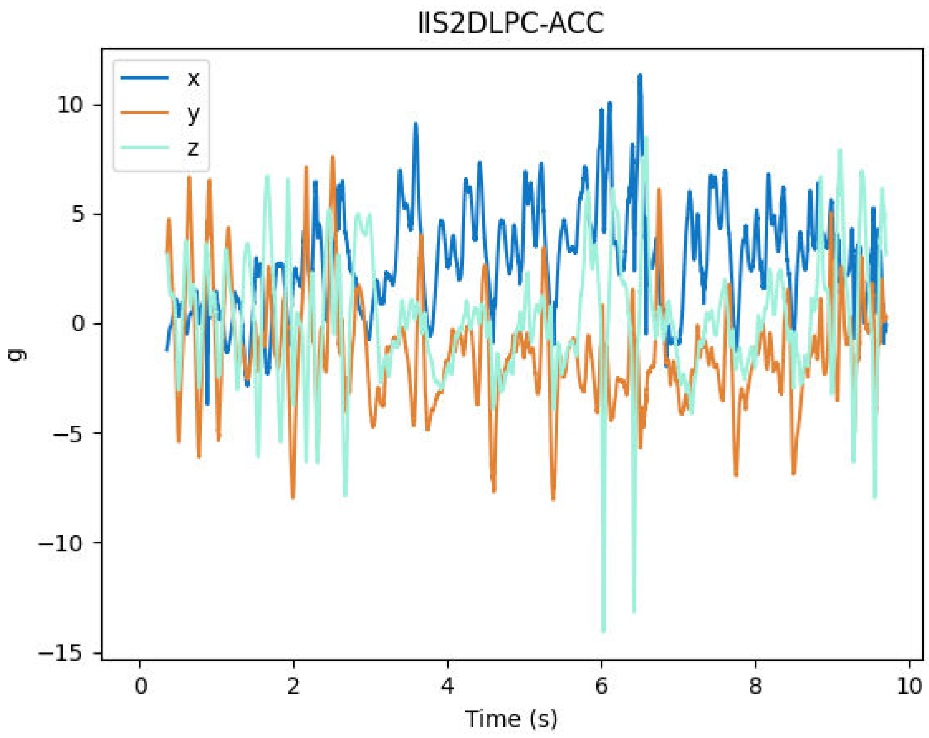


Figure 53. **IIS2DLPC-ACC data plot example**



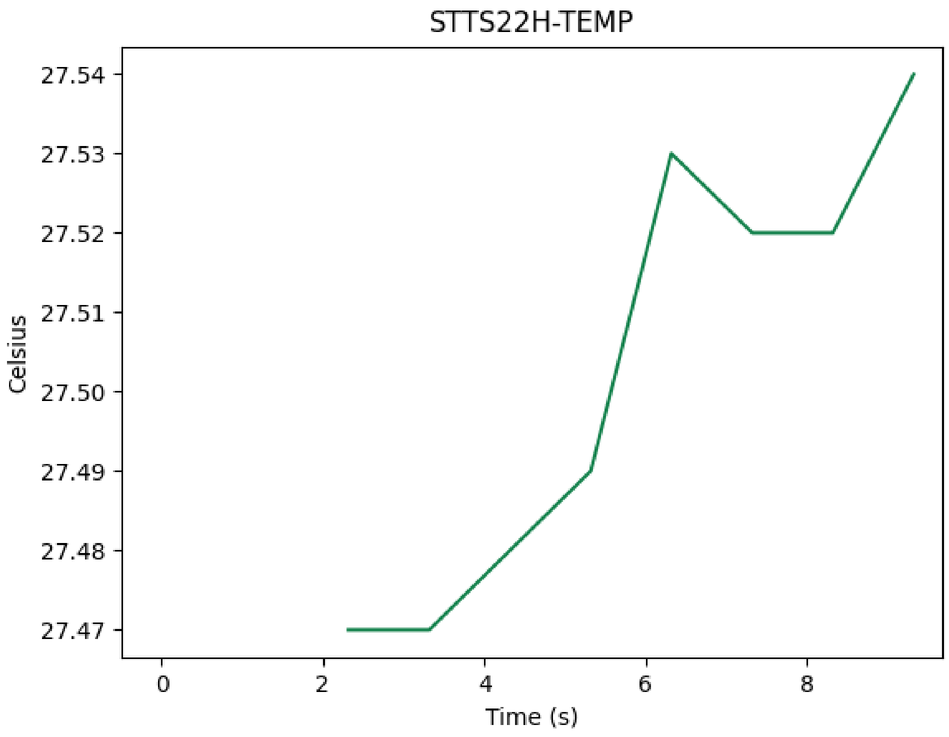


Figure 54. **STTS22H-TEMP data plot example**

2.3.5 Change default sensor configuration

You may be interested in changing the default sensor configuration.

For example, you could need to change the output data rate as well as the full scale or better yet to exclude a sensor from the data logging.

This goal can be reached by changing some parameters inside a *"DeviceConfig.json"* file to use either inside the FatFs volume, *"[Drive]:\datalog\DeviceConfig.json"* or as input for the USB WCID CLI command.

Open the *"DeviceConfig.json"* with a text editor software and look at the section *"subSensorStatus"* (see Figure 48. DeviceConfig.json - sensor\sensorStatus) related to the sub-sensor of the sensor you are going to modify.

You can edit the following fields:

- FS
- ODR
- isActive

So, for each sub-sensor, you can change full scale (i.e., FS), output data rate (i.e., ODR) as well as make it part of the data logging or not (i.e., isActive).

The accepted values for both FS and ODR are defined in the section *"subSensorDescription"* (see Figure 47. DeviceConfig.json - sensor\sensorDescriptor).

2.3.6 Change USB Functionality

The FP-SNS-DATAPRO1 allows to change the USB functionality of the STEVAL-PROTEUS among:

- USB-WCID: Windows Compatible ID used for wired data logging.
- USB-MSC: Mass Storage Class used to explore folders and files into the FatFs volume inside the embedded NOR flash memory as a mass storage device.
- USB not used: USB will only be used for charging.

You can do that in two ways:

- Using STBLESensor app.
- By pressing the user button.

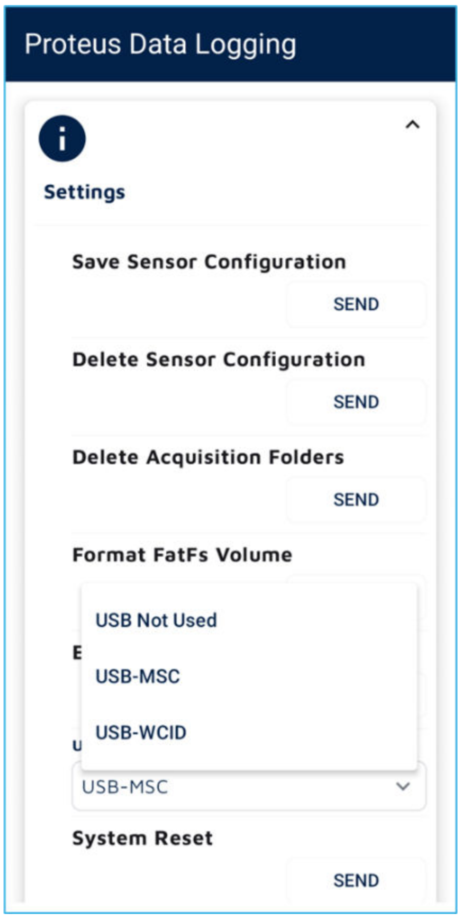
2.3.6.1 *Change USB functionality using STBLESensor app*

You can do it by performing the following steps:

- Turn on the STEVAL-PROTEUS.
- Launch the STBLESensor app app on an Android or iOS-based mobile device.
- Discover the board.
- Connect to the board.
- Tap *“Proteus Data Logging”*.
- Tap the kebab menu, in top-right corner.
- Tap *“Demo Settings”*.
- Expand the *“Settings”* tab.
- Tap the drop-down arrow, related to “USB Function” tab, and choose among:
  - USB Not Used
  - USB-MSC
  - USB-WCID
- Look at *“System Reset”* and tap *“Send”* button.

The STEVAL-PROTEUS will reboot and present the chosen USB functionality.

Figure 55. **Proteus Data Logging - Demo Settings, USB function**

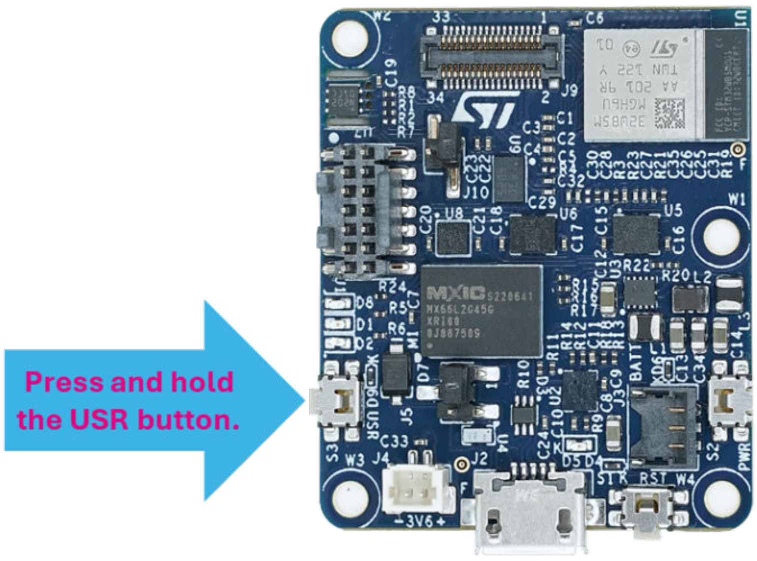


2.3.6.2 *Change USB functionality by pressing the user button*

You can do it by holding down the user button as follows:

- Between 10 s and 14 s: the STEVAL-PROTEUS will reboot and present the USB-WCID.
- Between 17 s and 21 s: the STEVAL-PROTEUS will reboot and present the USB-MSC.
- Between 24 s and 28 s: the STEVAL-PROTEUS will reboot and without any USB functionality.

Figure 56. **STEVAL-PROTEUS user button**



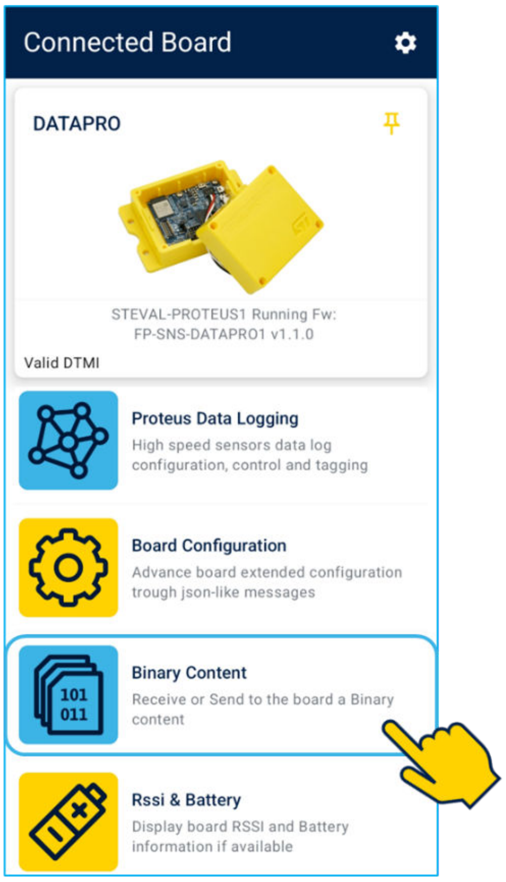
2.3.7 BLE file transfer (Binary content)

After FP-SNS-DATAPRO1 has been tried as stand-alone data logging (see Section 2.3.1) using the STBLESensor app the dataset has been saved in a folder of the FatFs volume into the embedded NOR flash memory.

The STBLESensor app allows you to transfer each file of the dataset from the STEVAL-PROTEUS to the mobile device running the app.

- Launch the STBLESensor app from your smart mobile device.
- Discover the STEVAL-PROTEUS and connect to it.
- Tap *"Binary Content"*.

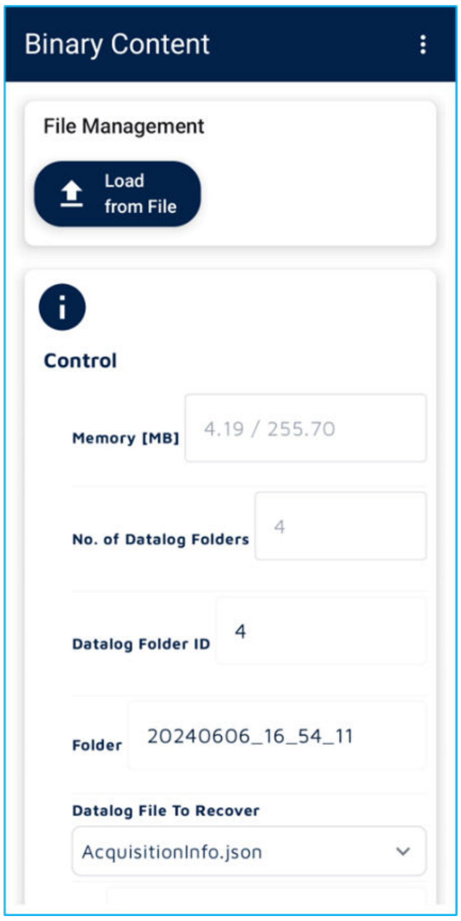
Figure 57. **STBLESensor app - Launch Binary Content**


Connected Board
DATAPRO
STEVAL-PROTEUS1 Running Fw:
FP-SNS-DATAPRO1 v1.1.0
Valid DTMI
Proteus Data Logging
High speed sensors data log configuration, control and tagging
Board Configuration
Advance board extended configuration trough json-like messages
Binary Content
Receive or Send to the board a Binary content
Rssi & Battery
Display board RSSI and Battery information if available

The demo shows:

- The FatFs volume occupancy.
- The total number of folders related to the data log.
- The active folder and its ID. You can enter a valid number for this ID to activate another folder.
- The active file you are going to transfer among the ones contained in the active folder.

Figure 58. **STBLESensor app - Binary Content Demo**



Memory [MB]: 4.19 / 255.70

No. of Datalog Folders: 4

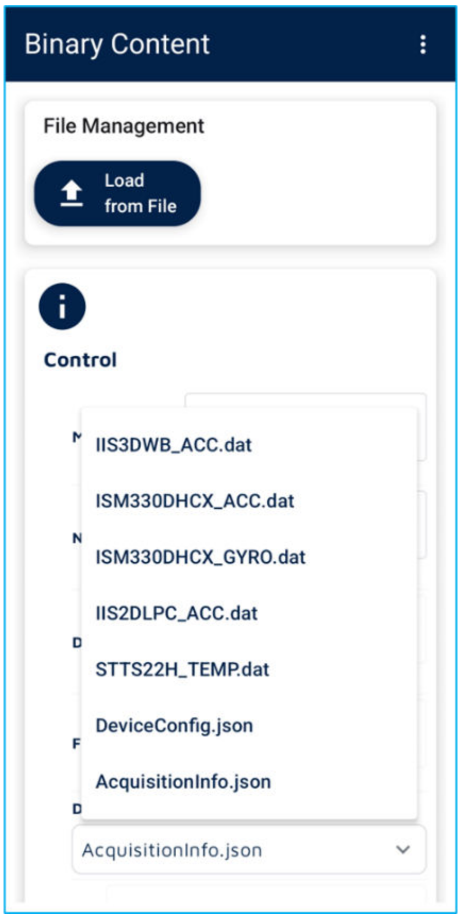
Datalog Folder ID: 4

Folder: 20240606_16_54_11

Datalog File To Recover: AcquisitionInfo.json

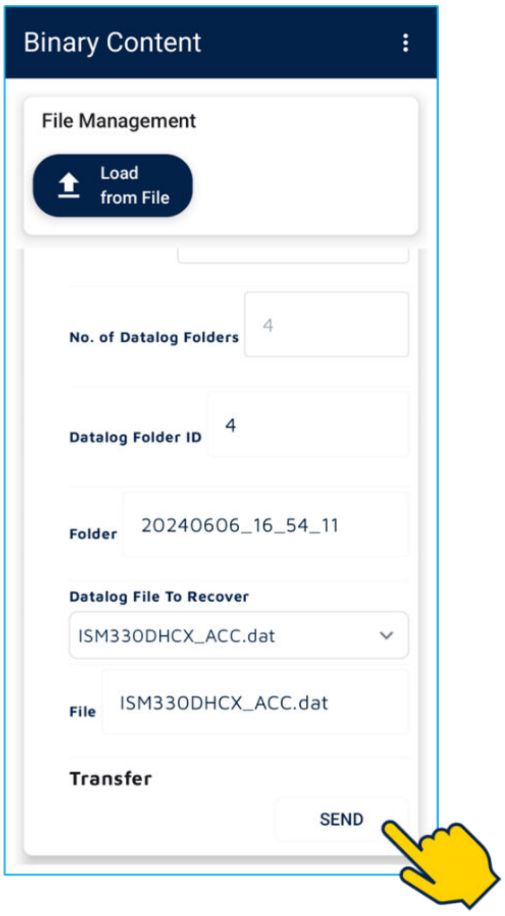
- Tap the drop-down arrow to change the active file.

Figure 59. STBLESensor app - Change active file for Binary Content



- Tap the *"SEND"* button to start the transfer.

Figure 60. STBLESensor app - Start file transfer with Binary Content



Binary Content

File Management

Load from File

No. of Datalog Folders: 4

Datalog Folder ID: 4

Folder: 20240606_16_54_11

Datalog File To Recover: ISM330DHCX_ACC.dat

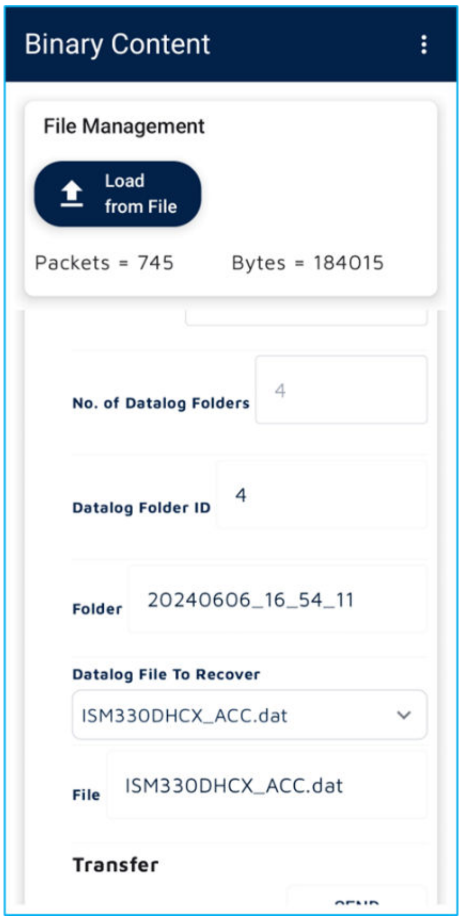
File: ISM330DHCX_ACC.dat

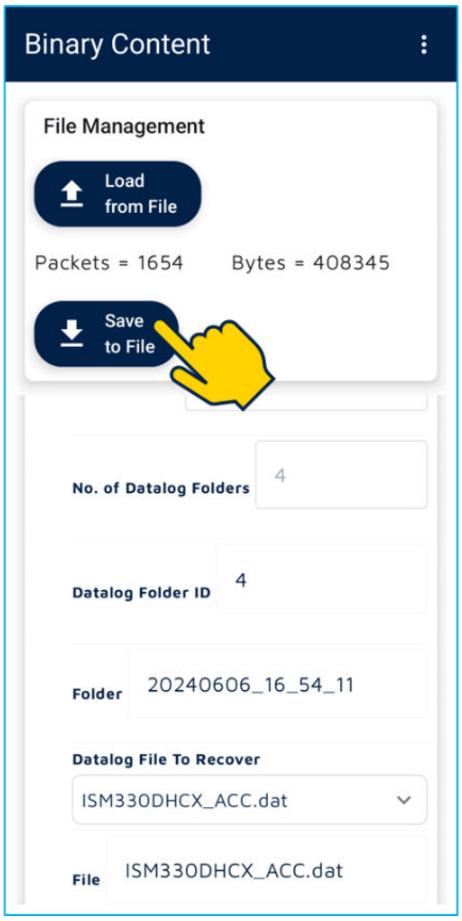
Transfer

SEND

- Look at the increase in the number of packets and bytes received. Each packet contains 1 byte of header and up to 247 bytes of data.
- As soon as all the bytes have been transferred, you can save them using the new button that has just appeared.

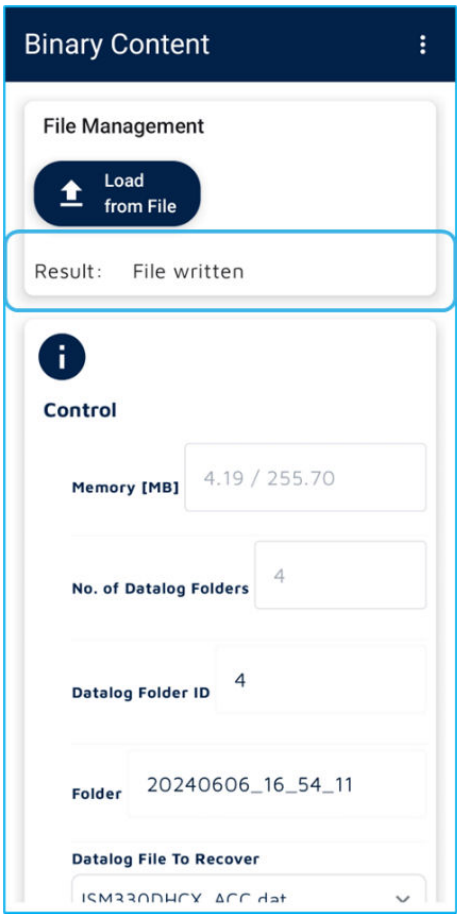
Figure 61. Proteus Data Logging - Transferring file with Binary Content





- Tap the “Save to file” button to save the file on your mobile device to a location you are asked to choose.

Figure 62. STBLESensor app - File has been correctly transferred with Binary Content



Result: File written

Memory [MB] 4.19 / 255.70

No. of Datalog Folders 4

Datalog Folder ID 4

Folder 20240606_16_54_11

Datalog File To Recover

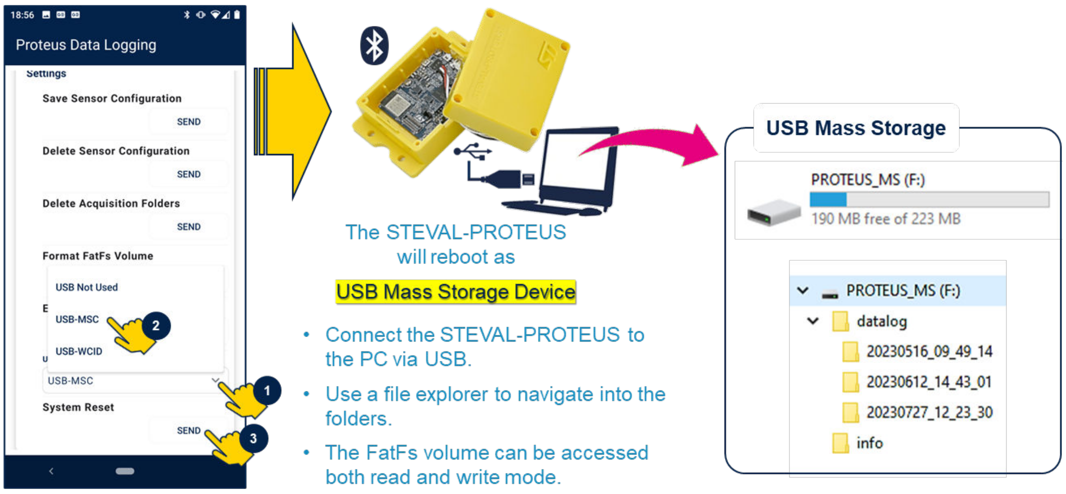
ISM330DHCX_ACC.dat

- At the end the result *"File written"* will be displayed.

2.3.8 STEVAL-PROTEUS as Mass Storage Device

The FP-SNS-DATAPRO1 allows to use the STEVAL-PROTEUS as a mass storage device (see Section 2.3.6: Change USB Functionality).

Figure 63. STBLESensor app – Board Configuration \ Custom Commands



The STEVAL-PROTEUS will reboot as USB Mass Storage Device

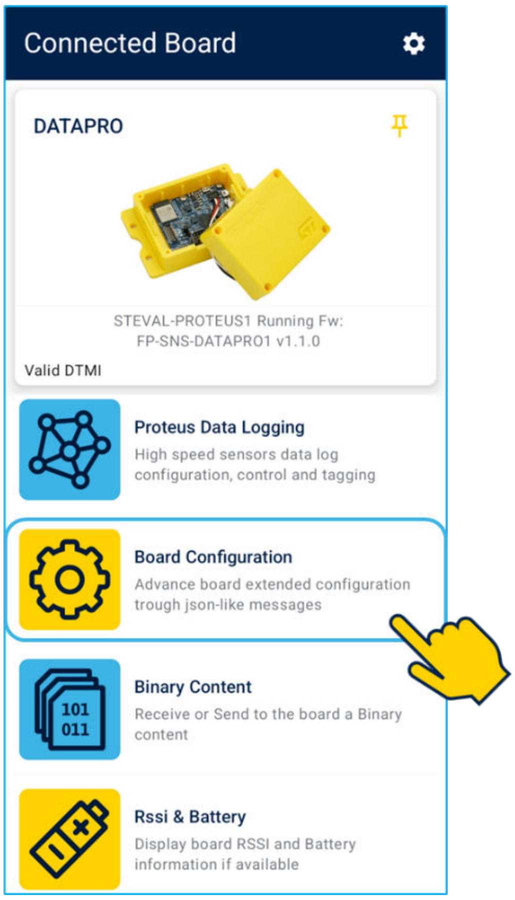
- Connect the STEVAL-PROTEUS to the PC via USB.
- Use a file explorer to navigate into the folders.
- The FatFs volume can be accessed both read and write mode.

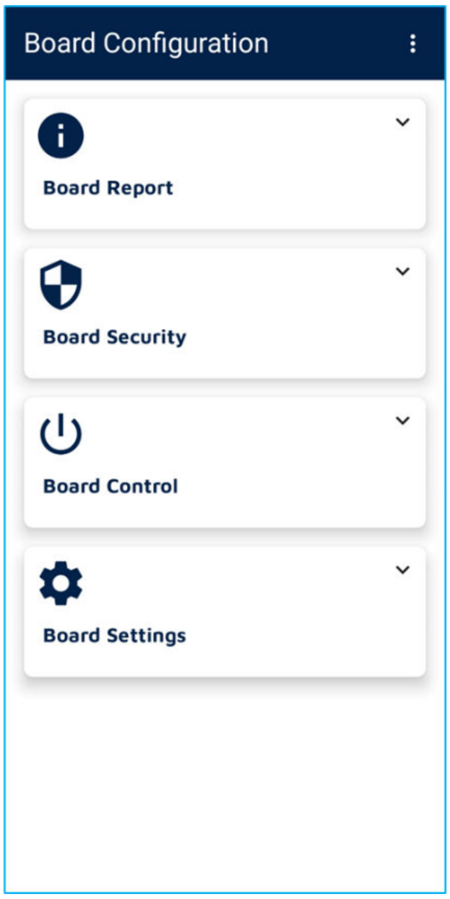
2.3.9 Board Configuration

The STBLESensor allows you to interact with the board so you can get some information and perform a few settings.

- Launch the STBLESensor app from your smart mobile device.
- Discover the STEVAL-PROTEUS and connect to it.
- Tap *"Board Configuration"*.

Figure 64. STBLESensor app - Board Configuration


Connected Board
DATAPRO
STEVAL-PROTEUS1 Running Fw:
FP-SNS-DATAPRO1 v1.1.0
Valid DTMI
Proteus Data Logging
High speed sensors data log configuration, control and tagging
Board Configuration
Advance board extended configuration trough json-like messages
Binary Content
Receive or Send to the board a Binary content
Rssi & Battery
Display board RSSI and Battery information if available



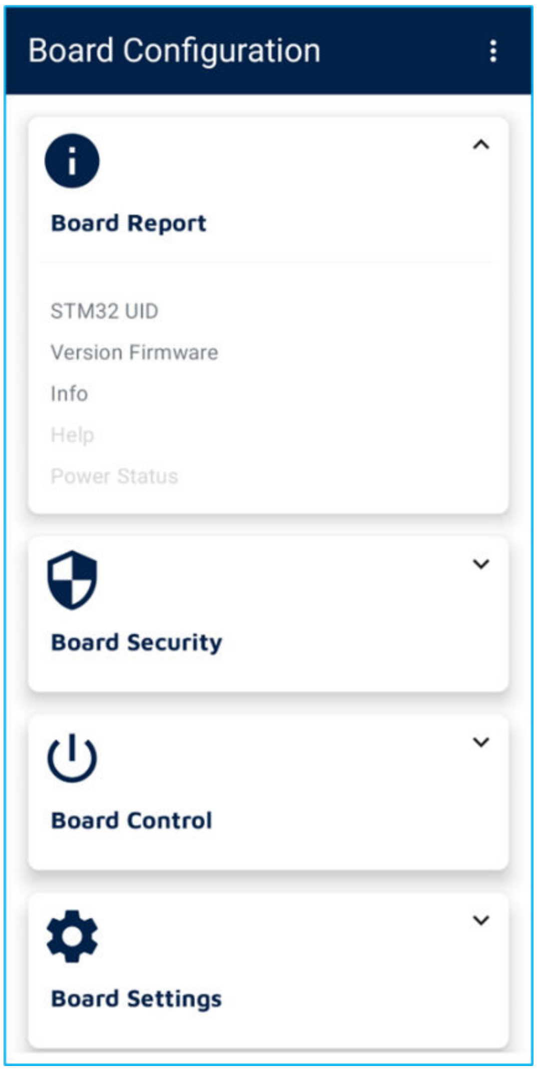
Here you can retrieve:

- Board report.
- Bard Security (not used by this application).
- Board Control.
- Board Settings.

Let us see them in detail.

- Tap *"Board Report"*.

Figure 65. **STBLESensor app - Board Configuration\Board Report**



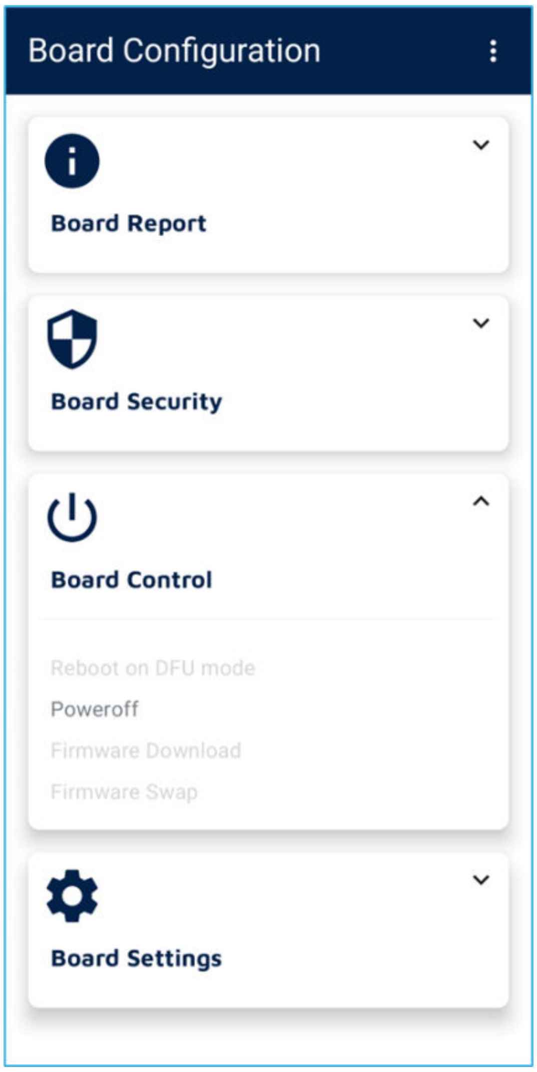
- STM32 UID: the Unique 96bit ID of the STM32WB mounted in the STEVAL-PROTEUS.
- Version Firmware: the FW release loaded on the STEVAL-PROTEUS.
- Info: a set of more detailed information about the FW loaded into the STEVAL-PROTEUS. It is already stored in a read-only json file in the FatFs volume inside the embedded NOR flash memory, *"[Drive]:\info\ BoardInfo.json"*. So, you can get information regarding the STEVAL-PROTEUS board and the FP-SNS-DATAPRO1 firmware just reading this file when the board exposes the USB as mass storage.

Figure 66. FP-SNS-DATAPRO1 - BoardInfo.json

```
{
    "Hardware": {
        "Manufacturer":
"STMicroelectronics",
        "Model": "STEVAL-PROTEUS1",
        "MCU Family": "STM32",
        "MCU Model": "WB5MMG",
        "Serial Number":
"002C00444741500E20313532"
    },
    "Software": {
        "FW Name": "FP-SNS-DATAPRO1",
        "FW Ver": "1.1.0",
        "FW Date": "Feb 16 2024",
        "CMSIS Ver": "5.3",
        "HAL Ver": "1.0,0",
        "RTOS": "FreeRTOS V10.3.1"
    }
}
```

- Tap *“Board Control”*, here you can perform:

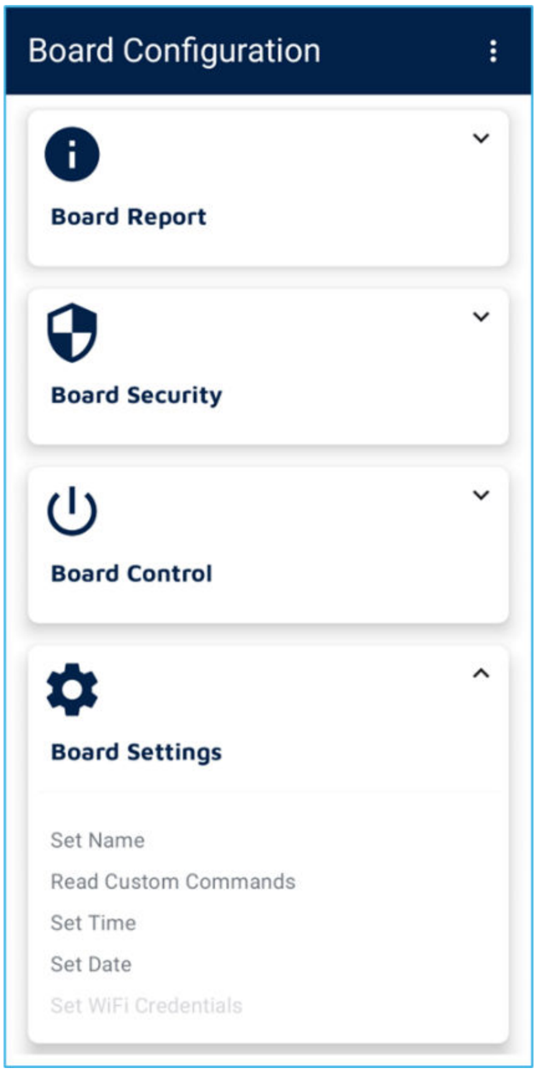
Figure 67. **STBLESensor app - Board Configuration \ Board Control**



  – Poweroff: tap it to turn off the STEVAL-PROTEUS.

- Tap *"Board Settings"*, here you can:

Figure 68. STBLESensor app - Board Configuration \ Board Settings



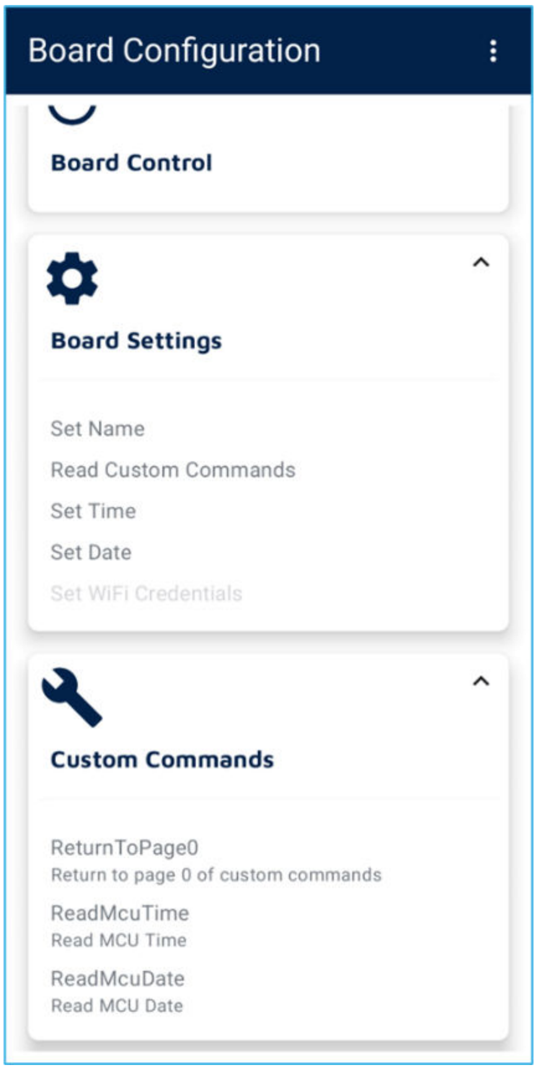
  - Set Name: tap it to change the BLE alias for the STEVAL-PROTEUS. This name will be also stored in a json file in the FatFs volume inside the embedded NOR flash memory, *"[Drive]:\info\NodeName.json"*. So, you can change the default BLE alias for the STEVAL-PROTEUS just modify the value of field *"NodeName"* in this file when the board exposes the USB as mass storage.

Figure 69. FP-SNS-DATAPRO1 - NodeName.json

```
{
    "NodeName": "DATAPRO"
}
```

  - Read Custom Commands: tap it to open a sub-menu that lets you execute some custom commands.
  - Set Time: tap it to change the RTC time for the STEVAL-PROTEUS.
  - Set Date: tap it to change the RTC date for the STEVAL-PROTEUS.

- Tap *“Read Custom Commands”*, here you can:
  - Read Time: tap it to read the RTC time for the STEVAL-PROTEUS.
  - Read Date: tap it to read the RTC date for the STEVAL-PROTEUS.


Board Configuration

Board Control

Board Settings

Set Name
Read Custom Commands
Set Time
Set Date
Set WiFi Credentials

Custom Commands

ReturnToPage0
Return to page 0 of custom commands
ReadMcuTime
Read MCU Time
ReadMcuDate
Read MCU Date

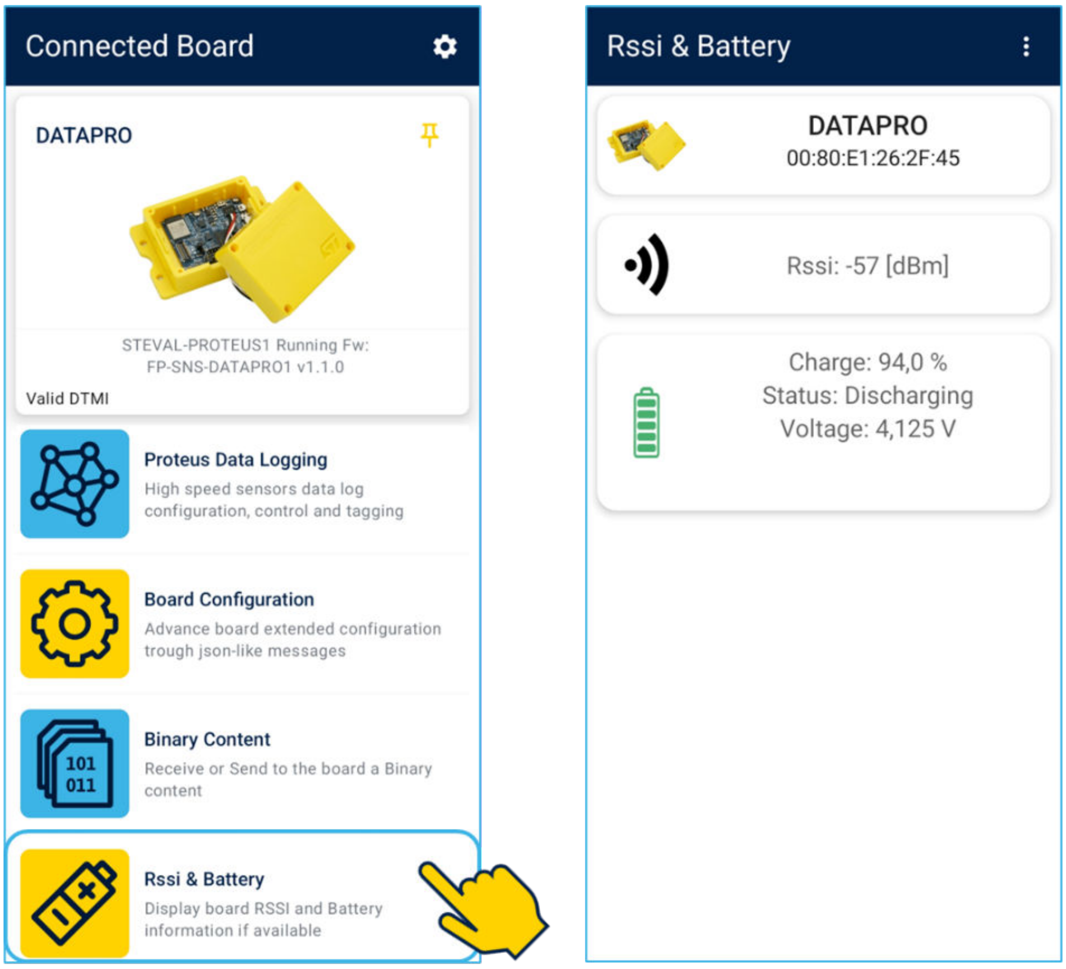
Figure 70. **STBLESensor app - Board Configuration \ Custom Commands**

2.3.10 RSSI & Battery

The STBLESensor allows you to get information about BLE and battery as BLE alias and MAC address, RSSI, and battery information such as level, status of the charge and voltage.

- Launch the STBLESensor app from your smart mobile device.
- Discover the STEVAL-PROTEUS and connect to it.
- Tap *“RSSI & Battery”*.

Figure 71. STBLESensor app - Board Configuration



Connected Board

DATAPRO

STEVAL-PROTEUS1 Running Fw:
FP-SNS-DATAPRO1 v1.1.0

Valid DTMI

Proteus Data Logging
High speed sensors data log configuration, control and tagging

Board Configuration
Advance board extended configuration trough json-like messages

Binary Content
Receive or Send to the board a Binary content

Rssi & Battery
Display board RSSI and Battery information if available

Rssi & Battery

DATAPRO
00:80:E1:26:2F:45

Rssi: -57 [dBm]

Charge: 94,0 %
Status: Discharging
Voltage: 4,125 V

Revision history

Table 2. **Document revision history**

Date	Revision	Changes
17-Sep-2024	1	Initial release.

Contents

List of tables

List of figures