# Getting started with the STM32Cube function pack for high speed datalogging and ultrasound processing

## Introduction

The ST High Speed Datalog (FP-SNS-DATALOG2) is a comprehensive multisensor data capture and visualization toolkit, engineered to facilitate the development of embedded data science applications.

The tool has been designed as an open and modular instrument, tailored for data scientists and embedded designers, that streamlines the capture of wideband and heterogeneous digital data streams from a variety of sensing and actuation platforms.

ST High Speed Datalog is compatible with STDATALOG-PYSDK, a data-centric design and user-friendly Python SDK, and can run with hardware boards that supply real-time data streams empowering users with full control of the data acquisition process.

The included firmware is compatible with the STBLESensor app, which also lets you manage the board and sensor configurations, start/stop data acquisition on SD card, and control data labeling. Sensor data can also be streamed using a C++-based companion host software or can be stored onto a microSD™ card.

Via the host PC and Bluetooth® Low Energy app, users can configure intelligent sensor processing unit (ISPU) and machine learning core unit (MLC). See Release Notes for the complete list of sensors supported.

The FP-SNS-DATALOG2 firmware can run on STEVAL-STWINBX1, STEVAL-STWINKT1B, STEVAL-MKBOXPRO, STEVAL-AFCI1, B-U585I-IOT02A, X-NUCLEO-IKS02A1 and X-NUCLEO-IKS4A1 with NUCLEO-U575ZI-Q or with NUCLEO-H7A3ZI-Q.

ST High Speed Datalog also natively supports STEVAL-PDETECT1, STEVAL-C34KAT1, STEVAL-C34KAT2, STEVAL-C34KPM1, STEVAL-C34DIL24, STEVAL-MKI230KA, STEVAL-MKI245KA, STEVAL-MKI246KA and SENSEVAL-SCB4XV1 add-ons for the STEVAL-STWINBX1.

It also supports STEVAL-MKI153V1, STEVAL-MKI223V1K, STEVAL-MKI229A, STEVAL-MKI234KA, STEVAL-MKI240KA, STEVAL-MKI247A and STEVAL-MKI251A add-ons for STEVAL-MKBOXPRO.

The ST High Speed Datalog is part of the ST Edge AI suite, which is an integrated collection of software tools, designed to facilitate the development and deployment of embedded AI applications.

This comprehensive suite supports both optimization and deployment of machine learning algorithms and neural network models, starting from data collection to final deployment on hardware, streamlining the workflow for professionals across various disciplines.

The ST Edge AI suite supports various ST products: STM32 microcontrollers and microprocessors, Stellar microcontrollers and MEMS smart sensors.

The ST Edge AI suite represents a strategic move to democratize edge AI technology, making it a pivotal resource for developers looking to harness the power of AI in embedded systems efficiently and effectively.

The software is also available 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-DATALOG2 software expansion for STM32Cube

## 1.1 Overview

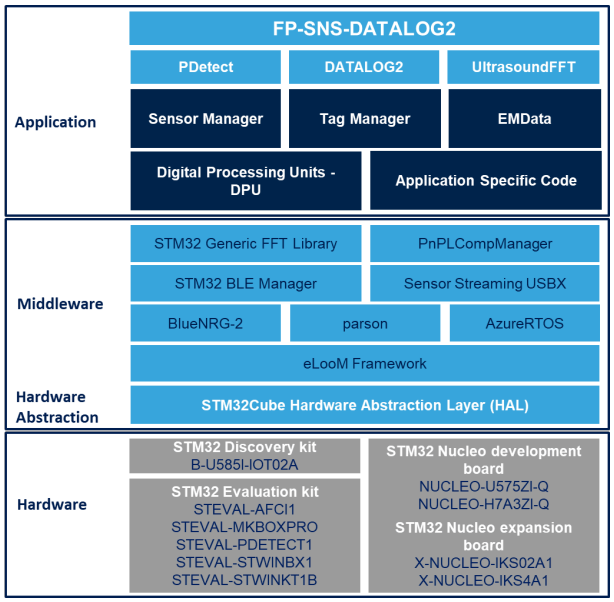
The FP-SNS-DATALOG2 features:

- High-rate (up to 6 Mbit/s) data capture software suite:
  - Compatible with STDATALOG-PYSDK, ready-to-use for integration into any data science design flow
  - Compatible with STBLESensor app for system setup and real-time control
  - Able to configure and enable ISM330DHCX and LSM6DSV16X machine learning core unit (MLC) and ISM330IS intelligent sensor processing unit (ISPU)
  - Synchronized timestamping and labeling mechanisms common to all sensors
- Generic FFT library middleware to enable frequency domain analysis for any kind of sensor through fast Fourier transform (with programmable size, overlapping, and windowing)
- AzureRTOS: ThreadX, FileX, USBX
- Easy portability across different MCU families, thanks to STM32Cube
- Firmware modular examples based on eLooM (embedded light object oriented framework for STM32) to enable code reusability at application level
- Free, user-friendly license terms

## 1.2 Architecture

The FP-SNS-DATALOG2 software has been developed for the B-U585I-IOT02A, STEVAL-AFCI1, STEVAL-STWINBX1, STEVAL-STWINKT1B and STEVAL-MKBOXPRO development kit, X-NUCLEO-IKS02A1 and X-NUCLEO-IKS4A1 with NUCLEO-U575ZI-Q or with NUCLEO-H7A3ZI-Q.

**Figure 1. FP-SNS-DATALOG2 software architecture**




The FP-SNS-DATALOG2, compliant with STM32Cube architecture, is structured into a set of layers of increasing abstraction.

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 (I²C, 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, which are inherited from STM32Cube, FP-SNS-DATALOG2 adds the code reusability at application level.

FP-SNS-DATALOG2 is an eLooM-based application-level firmware. It is based on several firmware modules that interface and offer their data to other application modules according to well defined design patterns and specific APIs.

Each firmware module is packed into a folder. They are totally self-contained and independent form each other. They also can be added to your custom firmware application by just dragging and dropping the needed folder.

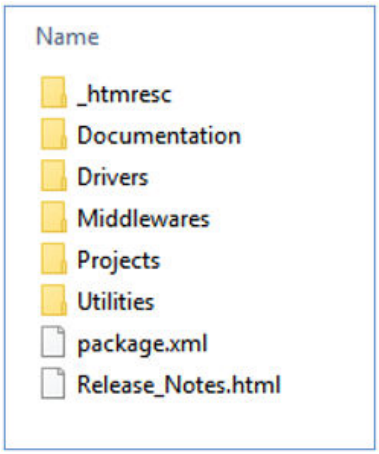
Each firmware module implements concretely or extends services, classes and objects made available by the eLooM framework. More specifically, here you can find:

- drivers, which are objects that implements the base interface for any low-level subsystem that can be used into the firmware module (I2C, DFSDM)
- events, which are objects that handle information about something that happened in the system at a given moment. These files implement the event and source/listener design patterns
- services, which are other utilities for the firmware module

## 1.3 Folder structure

The FP-SNS-DATALOG2 firmware package folder structure follows the layer-based approach of the STM32Cube architecture.

**Figure 2. FP-SNS-DATALOG2 package folder structure**

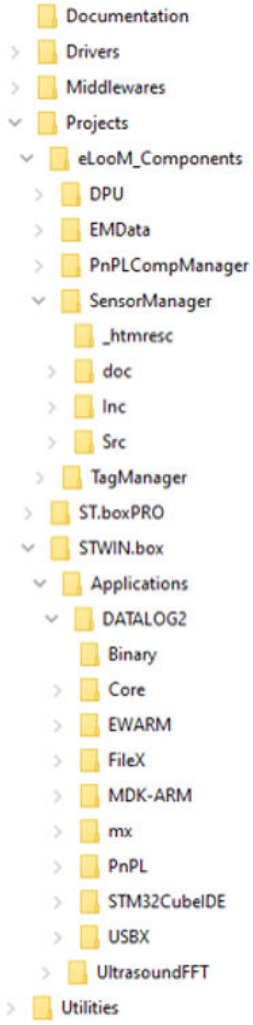



The folders included in the software package are:

- **Documentation**: contains a compiled HTML file generated from the source code, which details the software components andAPIs.
- **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 the eLooM (embedded light object-oriented framework) for STM32 and a set of libraries built around Microsoft® Azure® RTOS middleware (ThreadX real-time operating system (RTOS), FileX advanced Flash file system, USBX USB Device stacks and classes) and other in-house (BlueNRG-2, BLEManager, PnPLManager, GenericFFT library) and open source (Parson library). All are integrated and customized for STM32 MCU devices and enriched with corresponding application examples based on STM32 evaluation boards.

- **Projects**: contains eLooM application components (Data Processing Unit, EMData, PnPL Manager, Sensor Manager, Tag Manager) and samples application for High Speed Datalogging and Ultrasound analysis. This application is provided for the B-U585I-IOT02A, STEVAL-AFCI1, STEVAL-STWINBX1, STEVAL-STWINKT1B and STEVAL-MKBOXPRO evaluation kit, X-NUCLEO-IKS02A1 and X-NUCLEO-IKS4A1 with NUCLEO-U575ZI-Q or with NUCLEO-H7A3ZI-Q, with three development environments (IAR Embedded Workbench for ARM, RealView Microcontroller Development Kit (MDK-ARM), and STM32CubeIDE).
- **Utilities**: contains some complementary project files (C++ based cli_example, UCF and JSON configuration examples).

**Figure 3. FP-SNS-DATALOG2 subfolders**




## 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 How to program STWIN, STWIN.box and SensorTile.box PRO

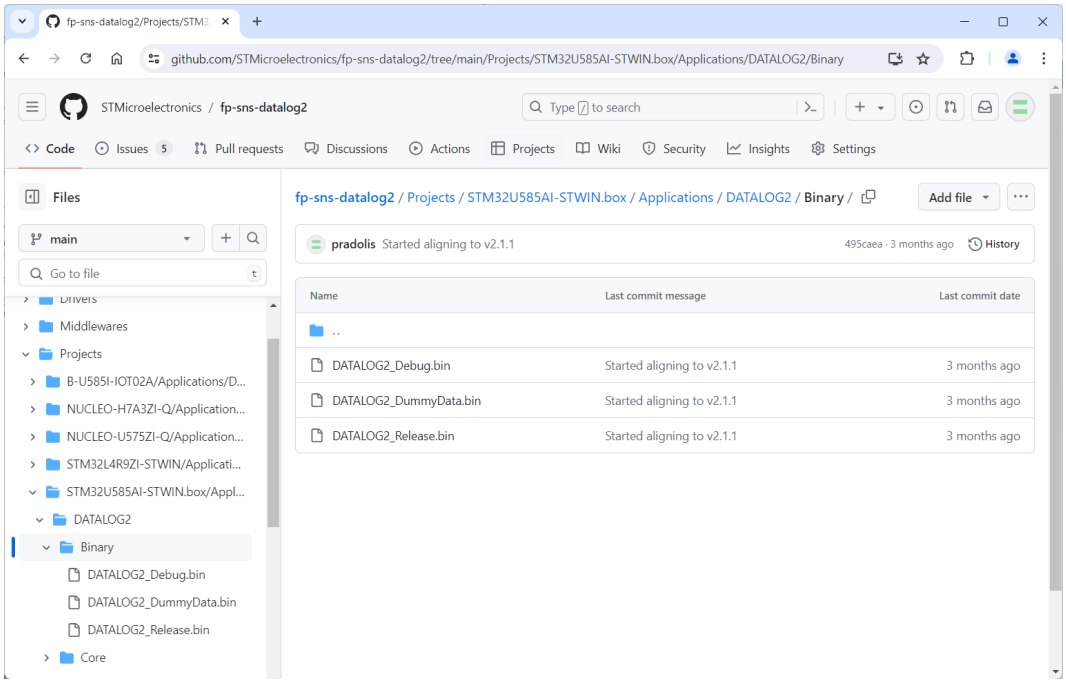
DATALOG2, PDetect and UltrasoundFFT are not the default firmware.

Next chapters explain different ways to update the STWIN, STWIN.box and the SensorTile.box PRO with the desired firmware.

The easiest way to update the firmware on the STEVAL-STWINKT1B, STEVAL-STWINBX1 and STEVAL-MKBOXPRO core system is to use the FOTA procedure through STBLESensor app.

FP-SNS-DATALOG2 also offers pre-compiled binaries (i.e.: in the folders Projects\STM32U585AI-STWIN.box\Applications\DATALOG2\Binary).

**Figure 4. FP-SNS-DATALOG2 application binary folders**



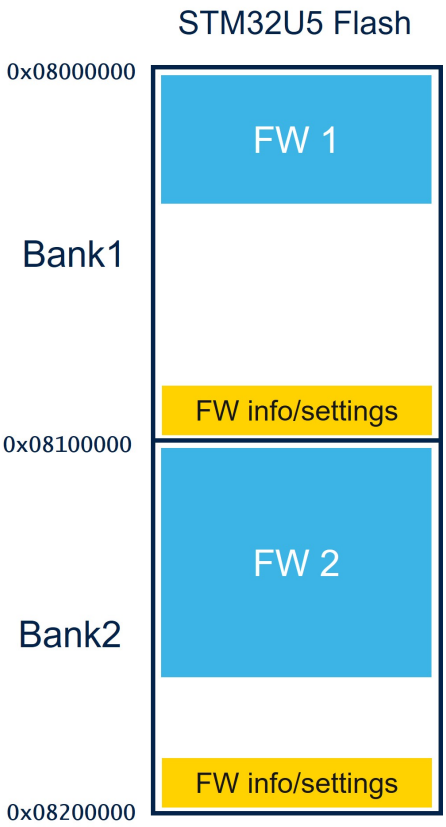

You are also free to recompile and modify the needed application with your preferred toolchain.

### 2.1.1 FFOTA - Fast Firmware update Over The Air

The STEVAL-STWINBX1 is released with a default firmware that enables the Bluetooth pairing via NFC and Fast Firmware On-The-Air upgrade through the STBLESensor app.

Both STM32L4+ and STM32U5 microcontroller families support a double-bank flash memory.

Figure 5. **Double-bank flash memory**

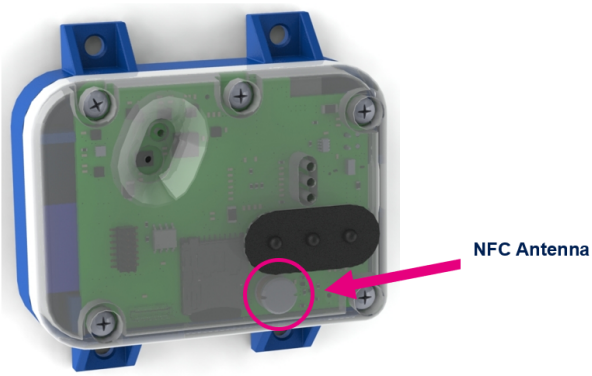



This feature enables Fast FOTA procedure without the need of designing custom bootloader firmware.

Moreover, two different types of firmware can be downloaded separately (one for each bank) and the boot bank is controlled through STM32 options bytes.

By just turning on Bluetooth and NFC on the smartphone and placing your smartphone on top of the NFC antenna of the STWIN.box, the smartphone reads the Bluetooth pairing information and automatically loads the STBLESensor.

Figure 6. **STWIN.box antenna**




The above procedure is available only for STWIN.box. STWIN and SensorTile.box PRO default firmware doesn't enable BLE pairing via NFC.

In alternative, STWIN, STWIN.box and SensorTile.box PRO can be connected to the STBLESensor app by just open manually the application.

The board presents itself as BLEDfFw. During the BLE pairing, if requested, you must insert the following PIN: 123456.

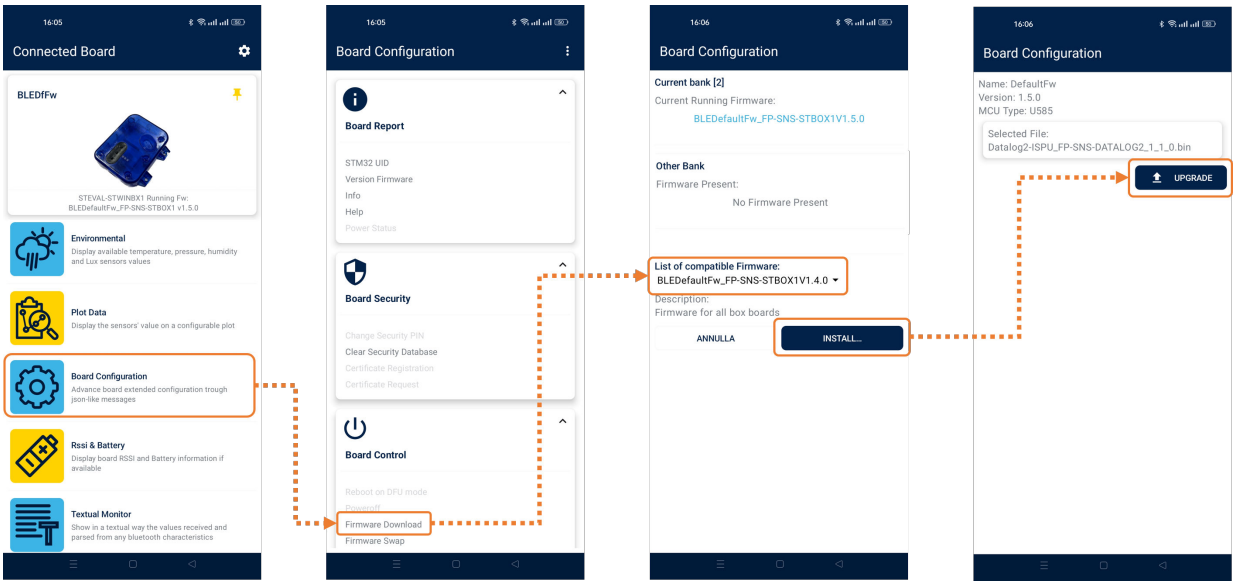
By selecting the specific tab in the ST BLESensor app, the application can show and plot data coming from the board.

At this point, you can choose to upgrade the firmware on the board directly by using the mobile app, by selecting one of the available firmware.

The firmware will be automatically loaded in the other flash bank.

Upgrade the firmware by following the steps shown in the following figure.

**Figure 7. Firmware upgrade procedure**




Once the download is finished on the other flash memory bank, the bank is automatically switched and the new firmware restarts automatically. To reconnect to the STBLESensor app (if needed), restart the app.

In Board Configuration tab you can also manually swap between 2 firmware already loaded into STWIN, STWIN.box and SensorTile.box PRO flash, download a new firmware or upgrade the current firmware with the latest version available from st.com.

### 2.1.2 How to program in “USB mode”

This is the easiest mode if you just want to download a binary into the board via USB, without the need of any debugging capabilities.

The advantage is that no additional debugger is needed, just a USB cable and STM32CubeProgrammer installed on your PC.

To enter the "Firmware upgrade" mode, follow the procedure below.

**Step 1.** Plug the board.

**Step 2.** Press the RESET button and, while keeping the button pressed, press USR button in STWIN and STWIN.box, DFU button in SensorTile.box PRO.

**Step 3.** Release the RESET button and then release USR button in STWIN and STWIN.box, DFU button in SensorTile.box PRO.

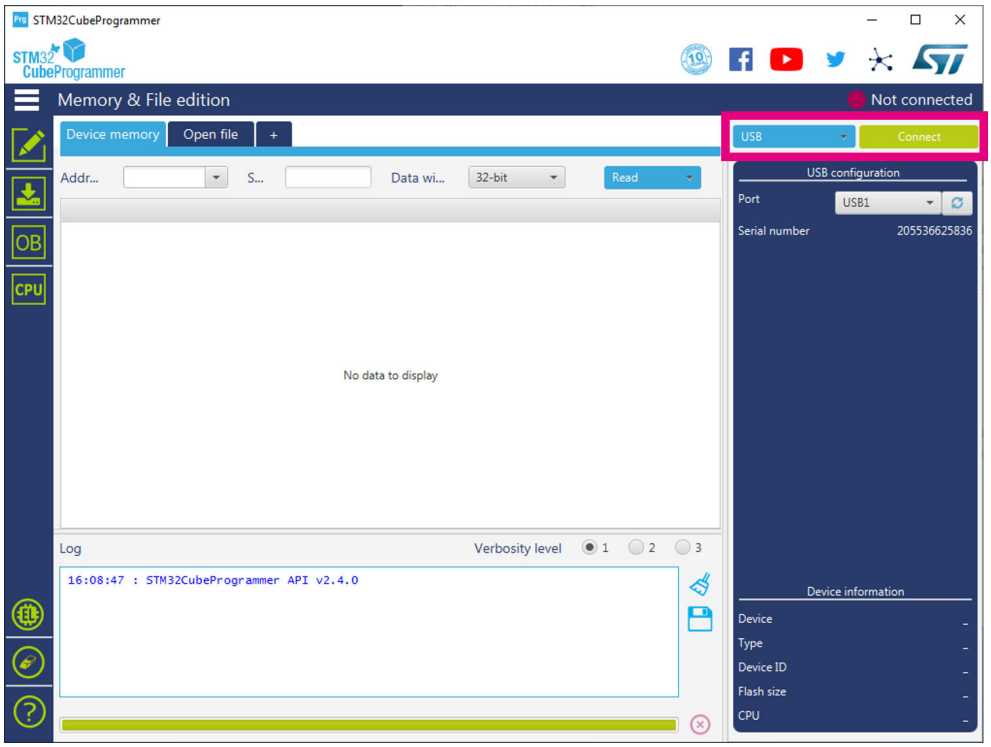
The board starts in the DFU mode and should be automatically recognized by the PC.

**Step 4.** You can upgrade the firmware by following the steps below:

**Step 4a.** Open STM32CubeProgrammer.

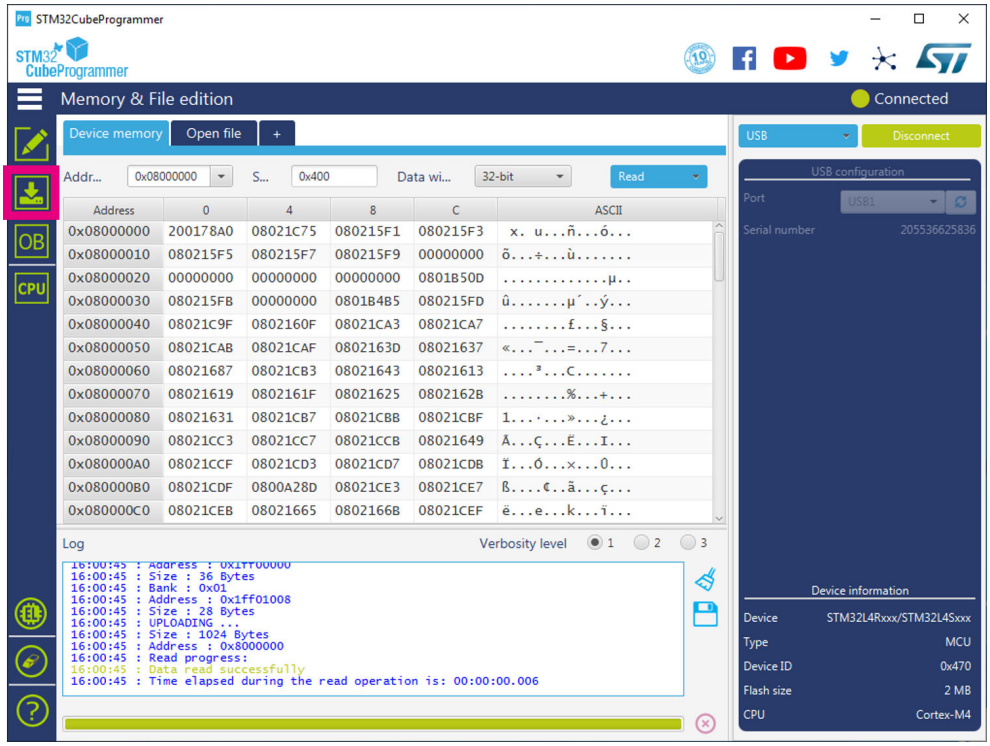
**Step 4b.** Select [**USB**] on the top-right corner.

**Figure 8. STM32CubeProgrammer - USB mode selection**




**Step 4c.** Click on [**Connect**].

**Figure 9. STM32CubeProgrammer - connection**

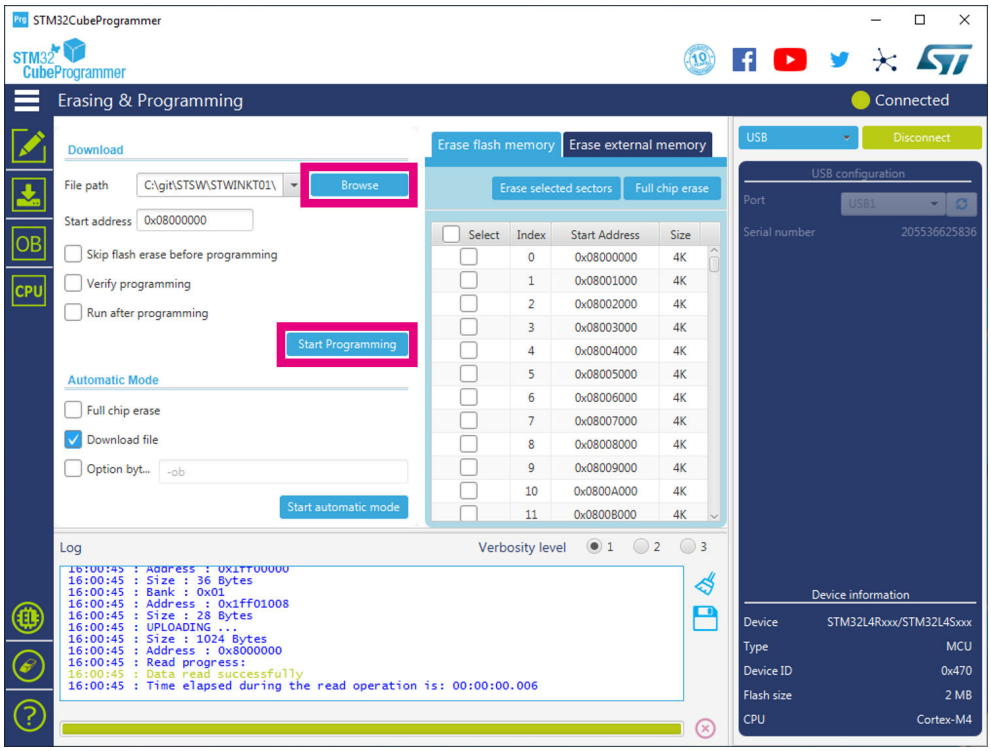



**Step 4d.** Go to the [**Erasing & Programming**] tab.

**Step 4e.** Search for the new .bin or .hex binary file to be flashed into the board.

**Step 4f.** Click on [**Start Programming**].

**Figure 10. STM32CubeProgrammer - programming**




### 2.1.3 How to program with an external debugger

STWIN, STWIN.box and SensorTile.box PRO programming connector is natively compatible with the STLINK-V3 debugger family (STLINK-V3SET or STLINK-V3MINIE).

*Note: STLINK-V3 programmer is included in STWIN kit only. STWIN.box and SensorTile.box PRO kits don't include any STLINK programmer.*

**Figure 11. STLINK-V3MINI connected to STWIN core system board**

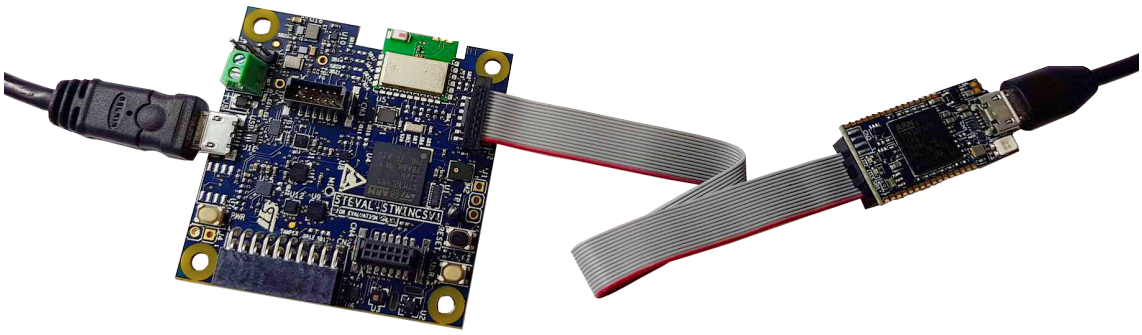
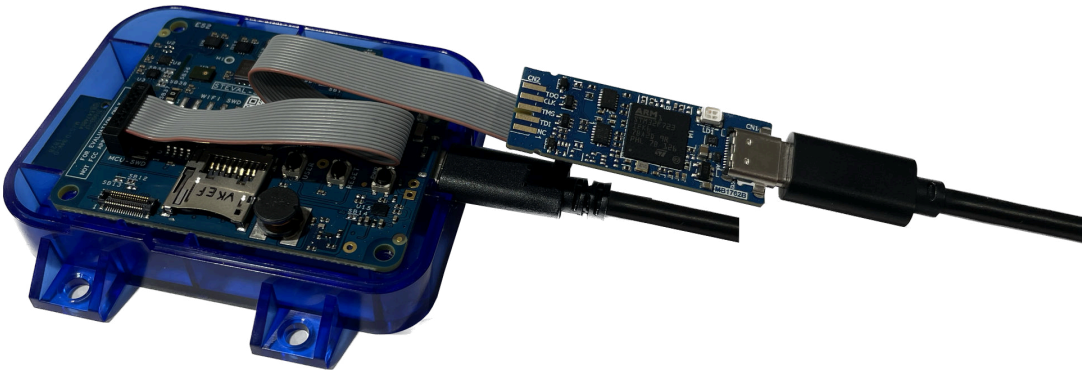


Figure 12. **STWIN.box and STLINK-V3MINIE programmer**



Alternatively, to offer more alternatives in STWIN.box and SensorTile.box PRO kits only. An adapter to ST-LINK V2-1 (STM32 Nucleo development board) or standard JTAG connector is included in the kit.

When using an STM32 Nucleo development board as an external debugger, you need to disconnect the on-board STM32 by removing the two jumpers on CN2 (see the picture below).

Figure 13. **STWIN.box, adapter, and STM32 Nucleo development board**

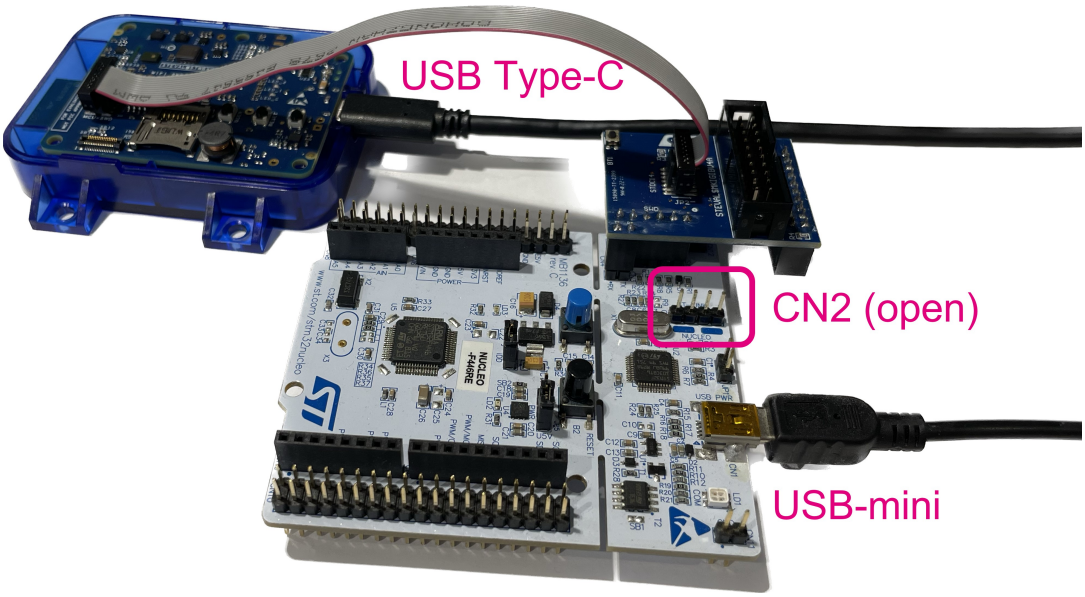
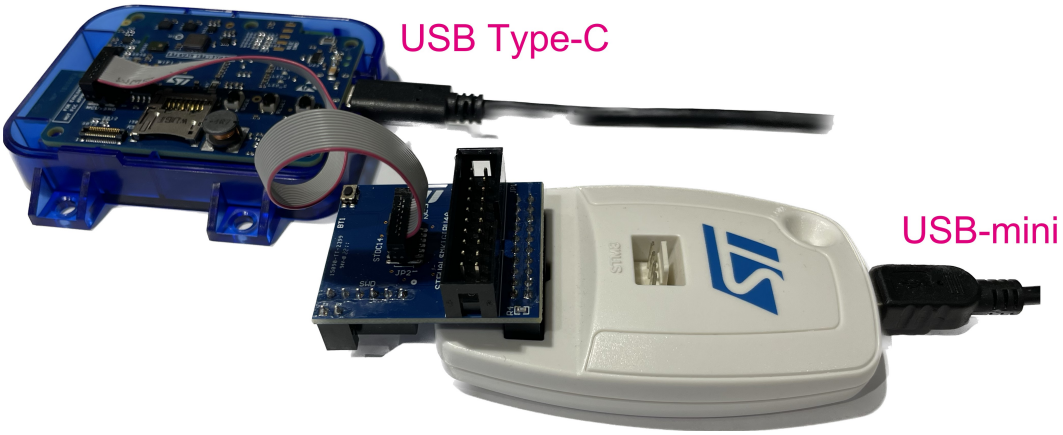



Figure 14. **STWIN.box and ST-LINK/V2 debugger (JTAG 20-pin 2.54 mm pitch connector)**




Once the hardware connections are in place, you can either:

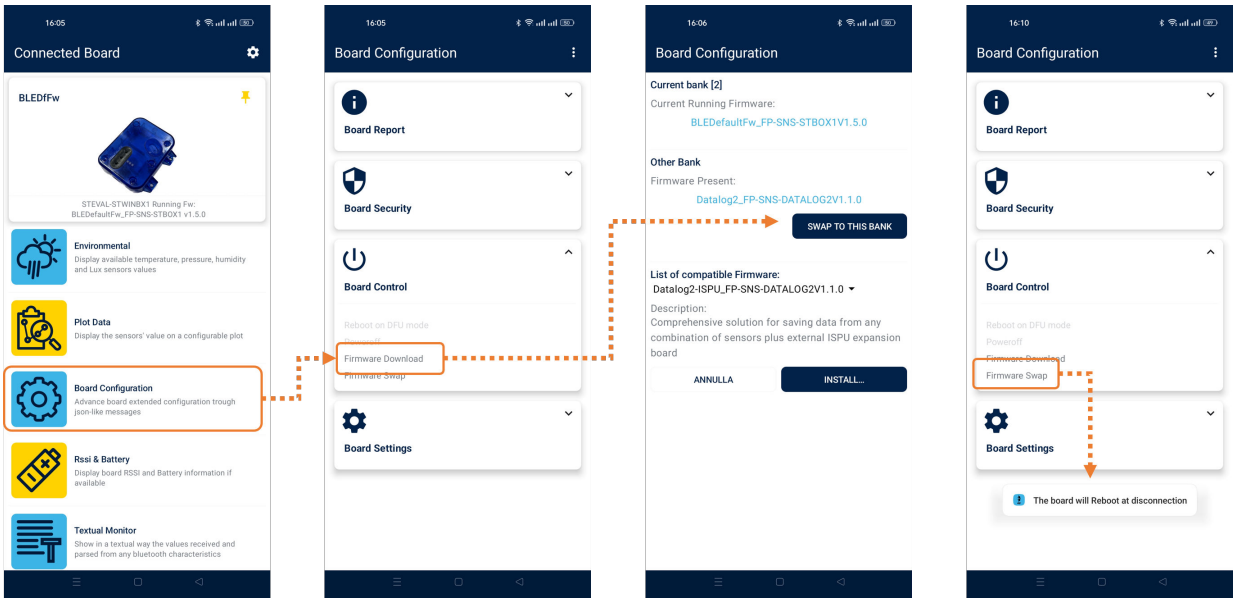
- download one of the sample application binaries provided using STM32CubeProgrammer
- recompile and flash memory one of the provided projects with your preferred IDE (STM32CubeIDE, EWARM, or Keil®)

### 2.1.4 How to handle double bank flash memory

To debug the code using one of the compatible IDE and debuggers, you need to make sure that the active flash bank is the first one. You can do this using the STBLESensor Mobile App (if the current FW supports the BLE) or using the STM32CubeProgrammer.

- STBLESensor app
  - Check the active bank as shown in the picture below and swap it if necessary.

**Figure 15. Swap flash bank with mobile app**




- STM32CubeProgrammer
  - Check the option byte configuration and make sure that the SWAP_BANK field (for STWIN.box and SensorTile.box) or BFB2 field (for STWIN) is unchecked.

Figure 16. **Swap flash bank with STM32CubeProgrammer 1/4**

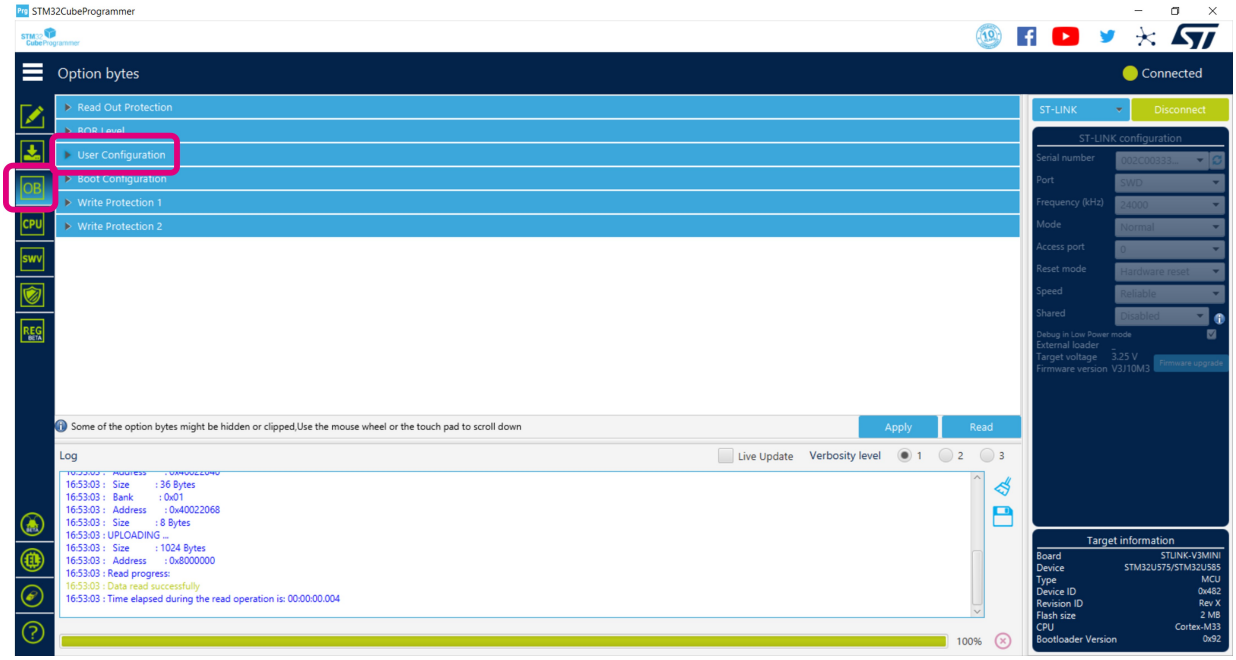



Figure 17. **Swap flash bank with STM32CubeProgrammer 2/4**

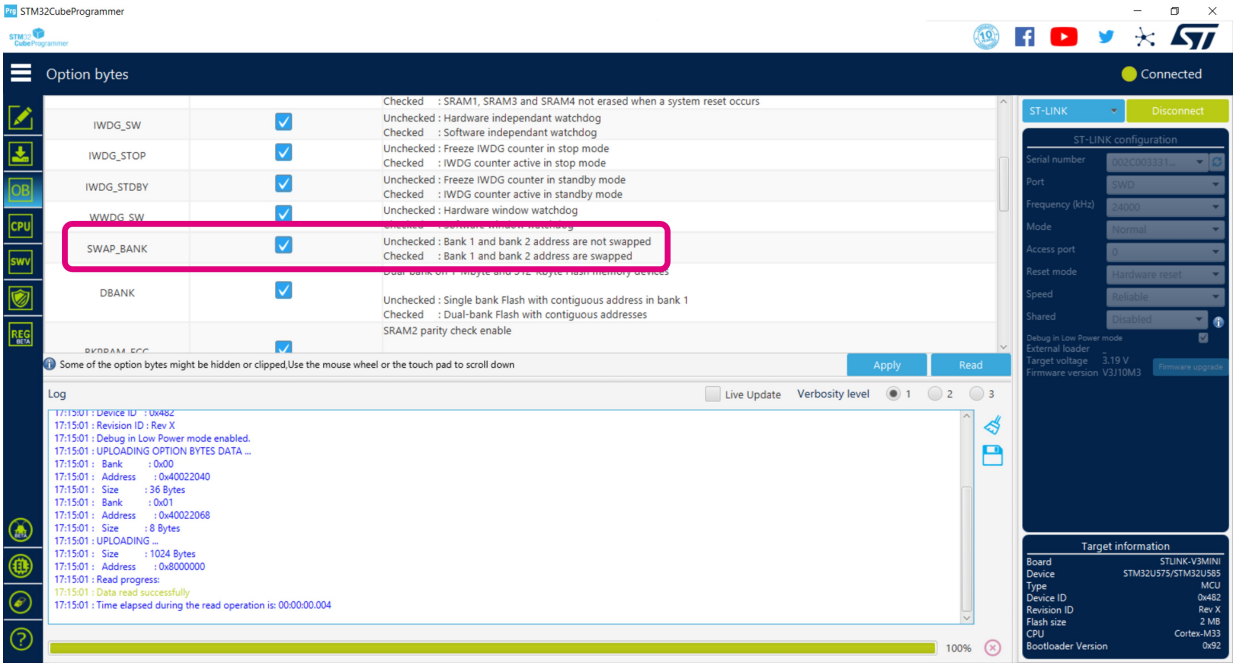


Figure 18. **Swap flash bank with STM32CubeProgrammer 3/4**

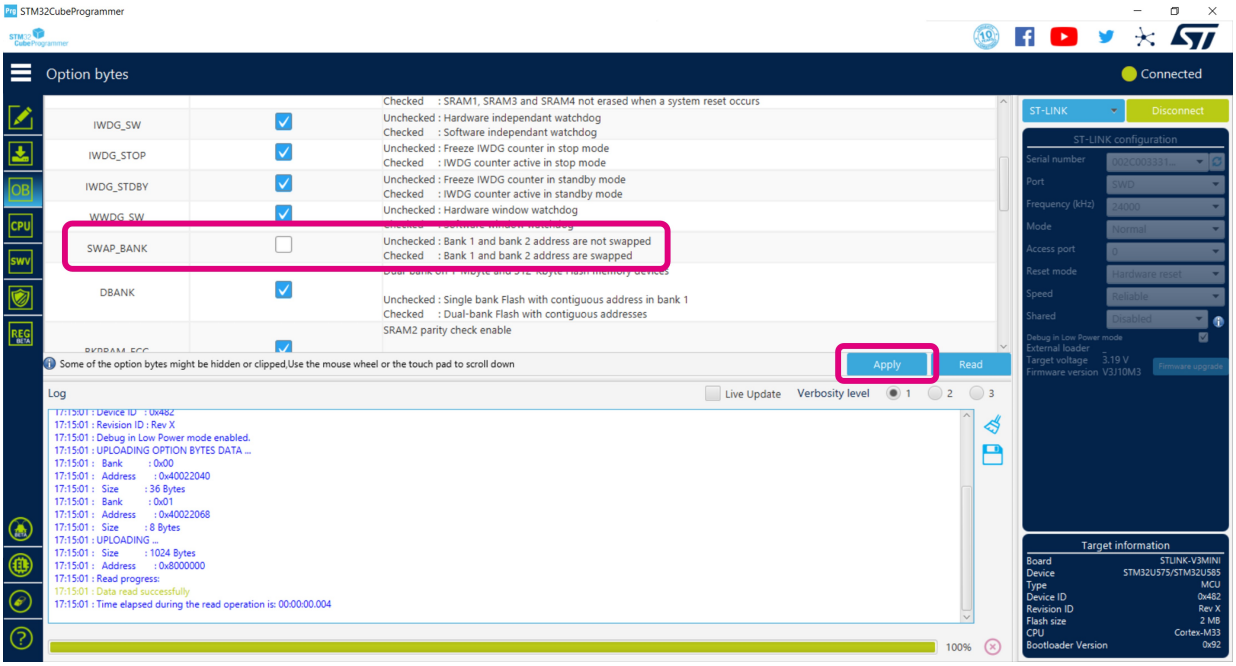
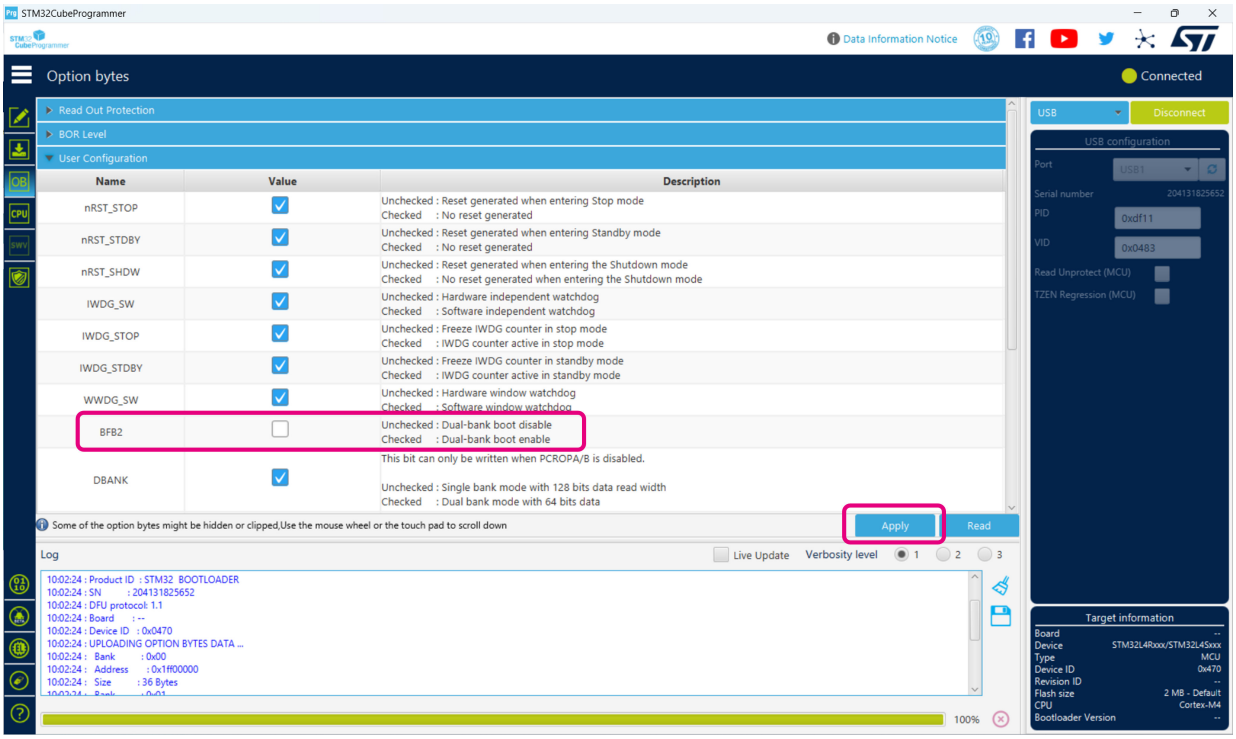



Figure 19. **Swap flash bank with STM32CubeProgrammer 4/4**

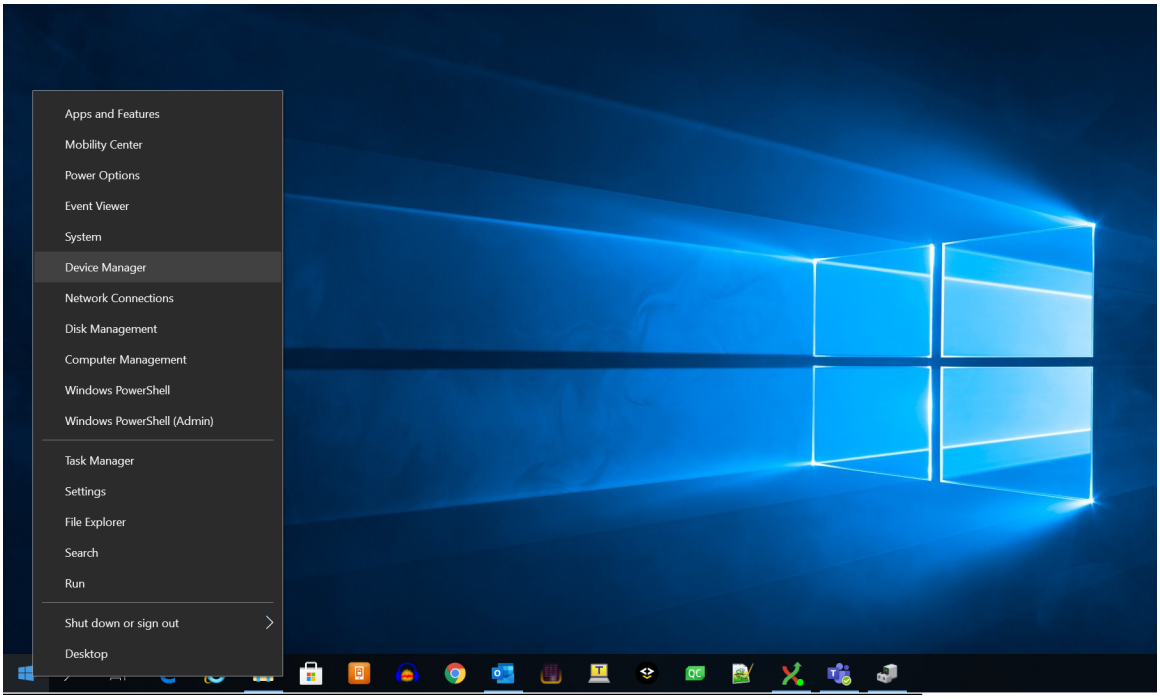



## 2.2 DATALOG2 and PDetect sample application

### 2.2.1 USB mode

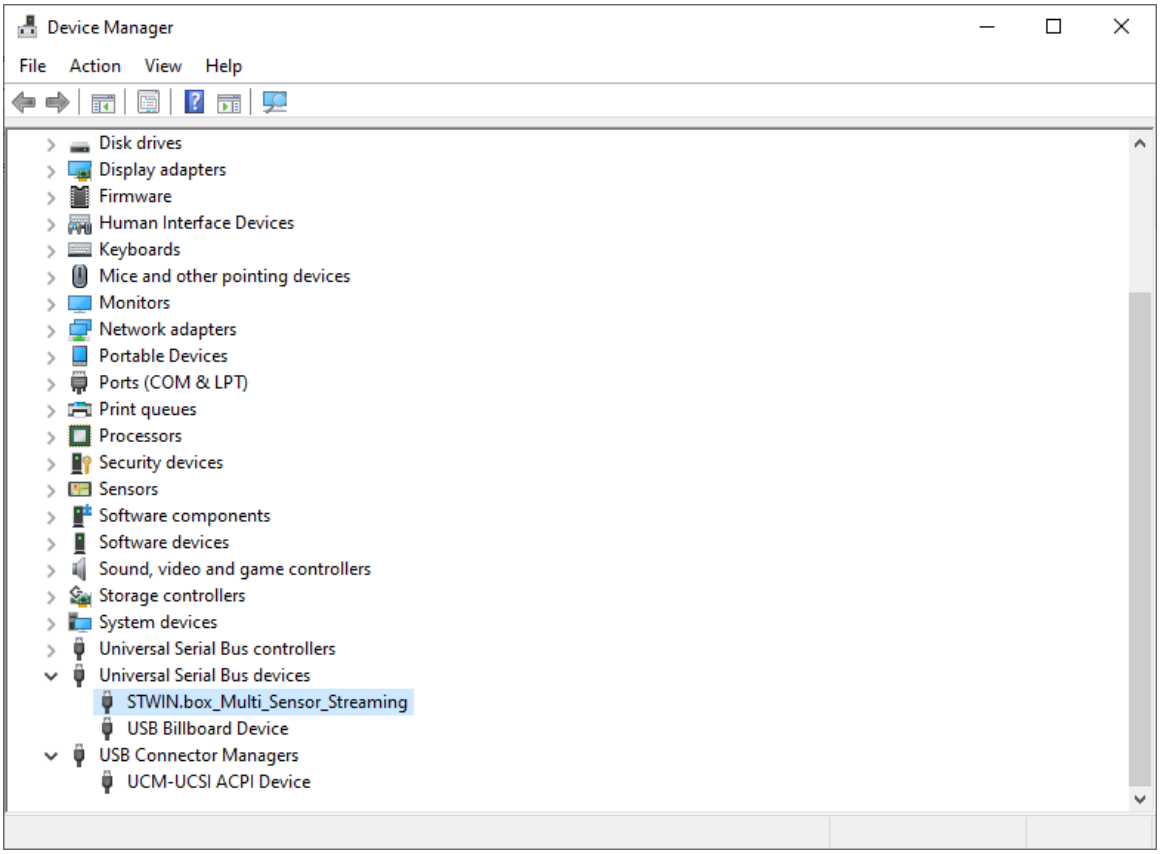
Once you plug the board to a PC via USB with the DATALOG2 and PDetect firmware already installed, the Operating System should recognize the board as a new USB device and automatically install the required drivers.

Figure 20. **Device Manager Windows**




To verify it, check whether you can see a Multi-Sensor Streaming device in the Device Manager Windows settings.

Figure 21. **Device Manager settings**

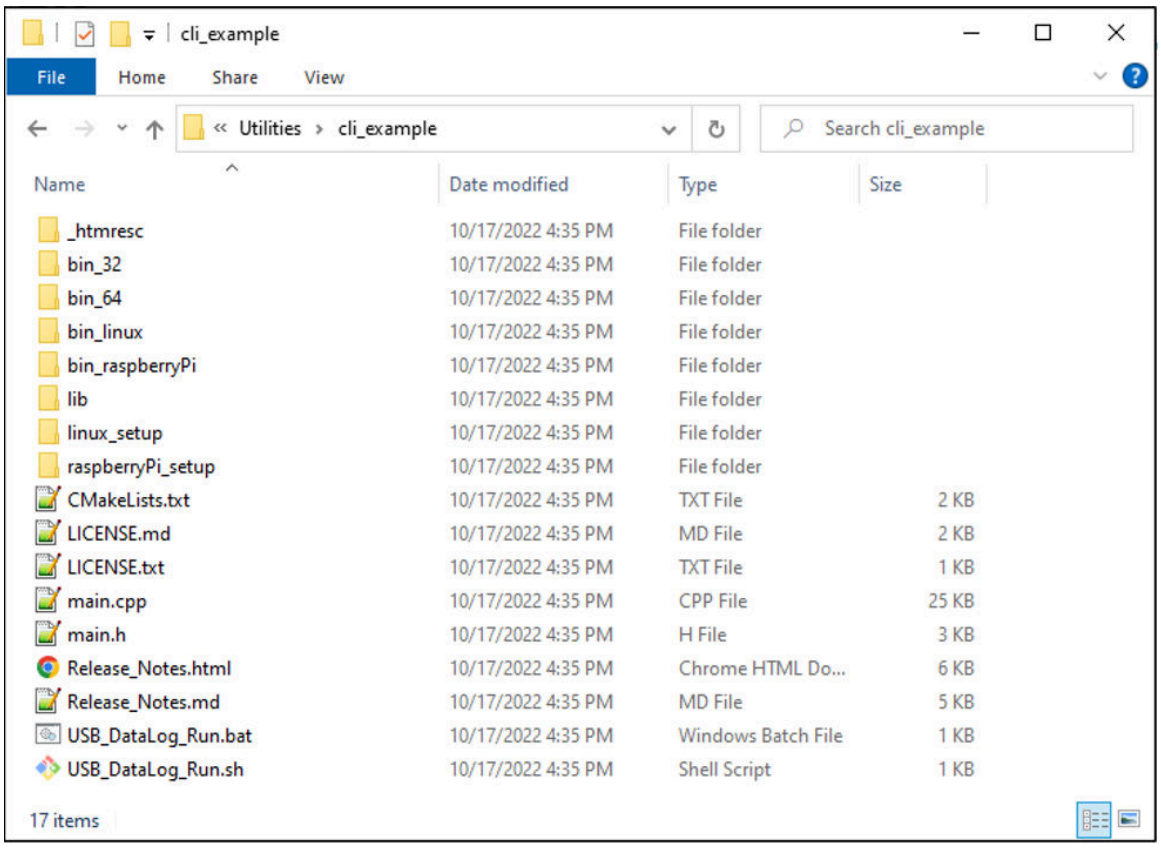


### 2.2.2 Command Line Interface

The Utilities folder contains a command line example.

The bin folder contains a pre-compiled version of the program available for Windows 32 and 64 bit, Linux and RaspberryPi platforms. A CMake project is also provided to make recompiling the application easy.

**Figure 22. Command line interface example**




If needed, the application can receive a configuration file in .json format, a configuration file to set up the machine learning core (MLC) or the intelligent sensor processing unit (ISPU) in .ucf format and a timeout as parameters.

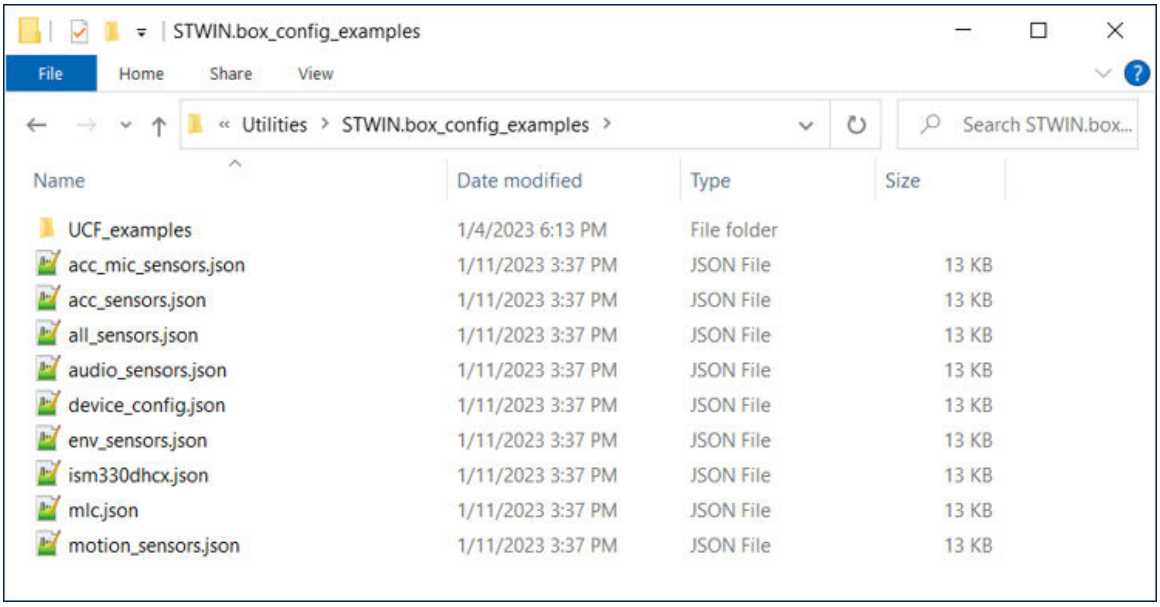
USB_DataLog_Run.bat for Windows and USB_DataLog_Run.sh for Linux scripts provide a ready-to-use example. You are free to customize the scripts to run the desired configurations.

**Figure 23. Customization of the scripts**

```
16 REM
17 REM Welcome to HS_DataLog Command Line Interface example
18 REM Usage: cli_example.exe [-COMMAND [ARGS]]
19 REM Commands:
20 REM -h Show this help
21 REM -f <filename>: Device Configuration file (JSON)
22 REM -u <filename>: UCF Configuration file for MLC
23 REM -t <seconds>: Duration of the current acquisition (seconds)
24
25
26 set PATH=%PATH%;.\bin_64\
27
28 cli_example.exe -f ..\STWIN.box_config_examples\device_config.json -t 100
29
30 pause
```

To configure the selected sensors, you can use one of the available device configuration examples in Utilities/STWIN.box_config_examples.

**Figure 24. Device configuration examples**




For retro-compatibility, it is still supported the possibility to define a custom sensors configuration by editing one of the available examples. Be aware that it is an error-prone procedure we would not suggest.

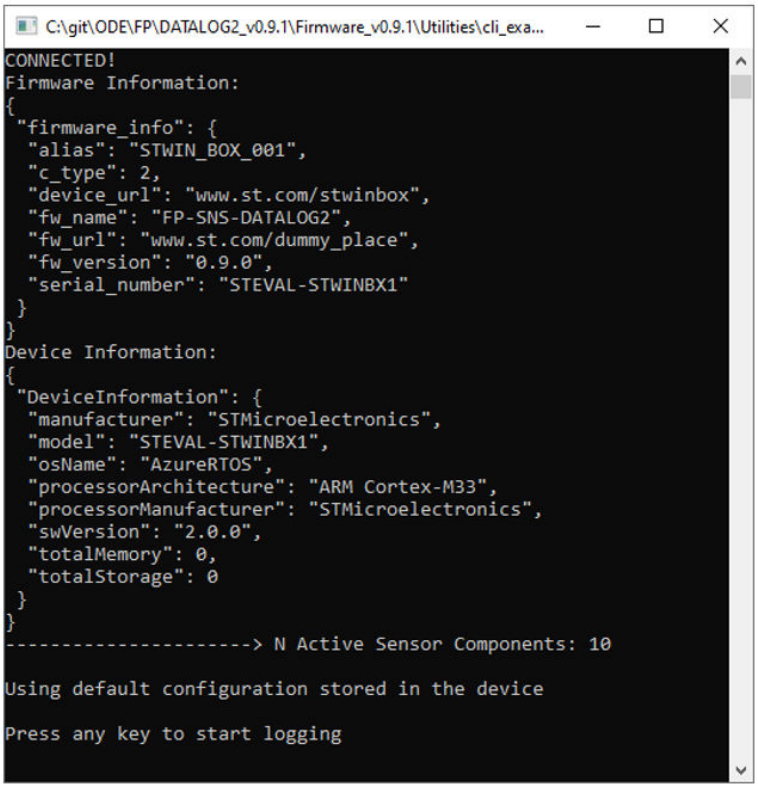
*Note: odr and fs fields are enumerative values defined into the Device Template Model. Please refer to Section 2.6.5: Device Template Model and DATALOG2 DTM for the mapping between the enum values and the real ODR and Full Scale.*

Figure 25. **Sensor configuration example**


Other UCF examples are freely available on Github.

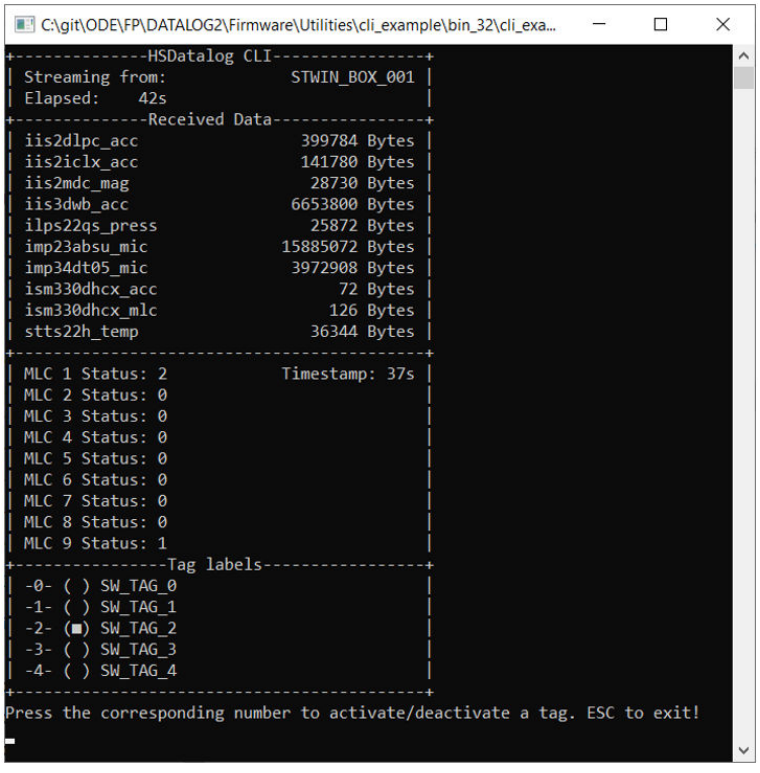
By double clicking on the USB_DataLog_Run batch script or launching directly the cli_example excutable, the application starts and the following command line appears, showing information about the connected board.

Figure 26. **Information about the connected board**




Press any key to start the datalogging.

Figure 27. **Datalogging**



The cli_example shows the amount of data received from any sensor and the MLC status (if available and enabled). It allows tagging the acquisitions by pressing 0-4 keyboard buttons.

The application stops automatically if a timeout has been set. In any case, you can stop the data acquisition by pressing the ESC button.

The application creates a YYYYMMDD_HH_MM_SS (for example, 20200128_16_33_00) folder, which contains the raw data, the JSON configuration file, and the UCF configuration file, if loaded.

For further details on the acquisition folder, see Section 2.6: Acquisition folders.

### 2.2.3 Real Time Plot: stdatalog_GUI

From FP-SNS-DATALOG2 v3.0.0, former HSDPython_SDK has been expanded and moved to a separate software product: STDATALOG-PYSDK.

The STDATALOG-PYSDK is a comprehensive Python framework designed to facilitate the capture, processing, and visualization of data from a wide range of sources, including sensors, algorithms, simulated signals, and telemetry from actuators. It is compatible with all firmware examples available in FP-SNS-DATALOG2, FP-IND-DATALOGMC, and FP-SNS-DATALOG1.

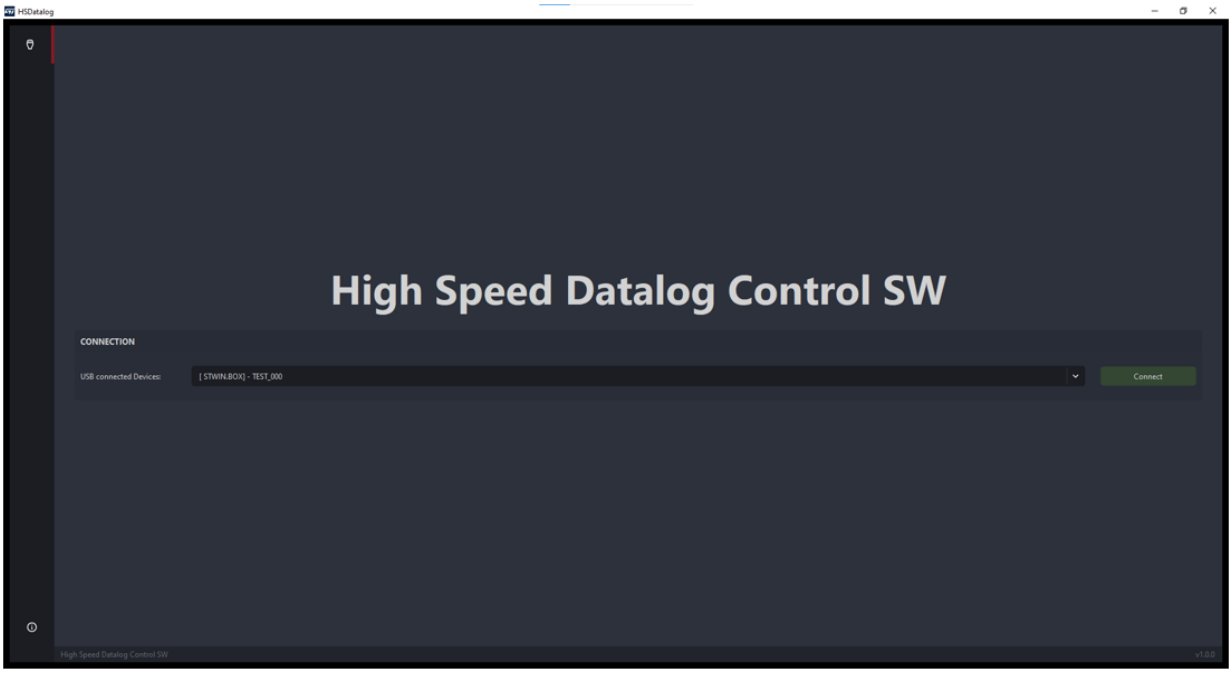
The python software development kit (SDK) for data logging has been developed using Python 3.12. To properly use it, Python 3.10, 3.11, or 3.12 must be already installed on the user's machine.

The STDATALOG-PYSDK requires different Python modules. The package is distributed with installers that solve all the required dependencies

*Important:* *This chapter show how to use the stdatalog_GUI.py script available in STDATALOG-PYSDK. Please see the full documentation available on the SDK landing page.*

Once the board is connected via USB and the Python environment has been properly updated, you can launch the real-time plot by just executing *stdatalog_GUI.py*.

**Figure 28. High Speed Datalog Control SW**




Click on the *Connect* button to allow the connection between the board and the PC.

Once the connection is established, you can:

- Enable/disable the needed sensors
- Set up data rate, full scale, timestamps
- Retrieve sensor status
- Load UCF to set up a MLC or an ISPU
- Save and load a configuration via a JSON file

- Start/stop logging data on the PC

Once you click on the *Start Log* button, data are live plotted and the application creates a YYYYMMDD_HH_MM_SS (for example, 20200128_16_33_00) folder that contains the raw data and the JSON configuration file.

**Figure 29. Real Time Plot GUI - MLC configuration**

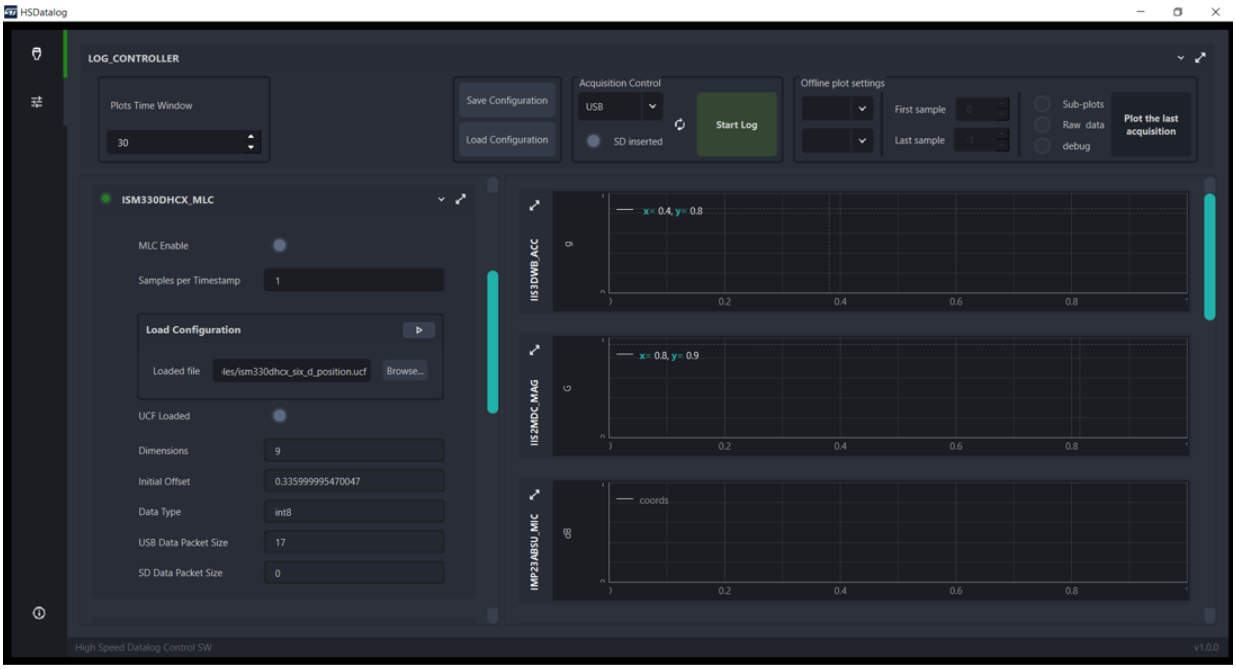
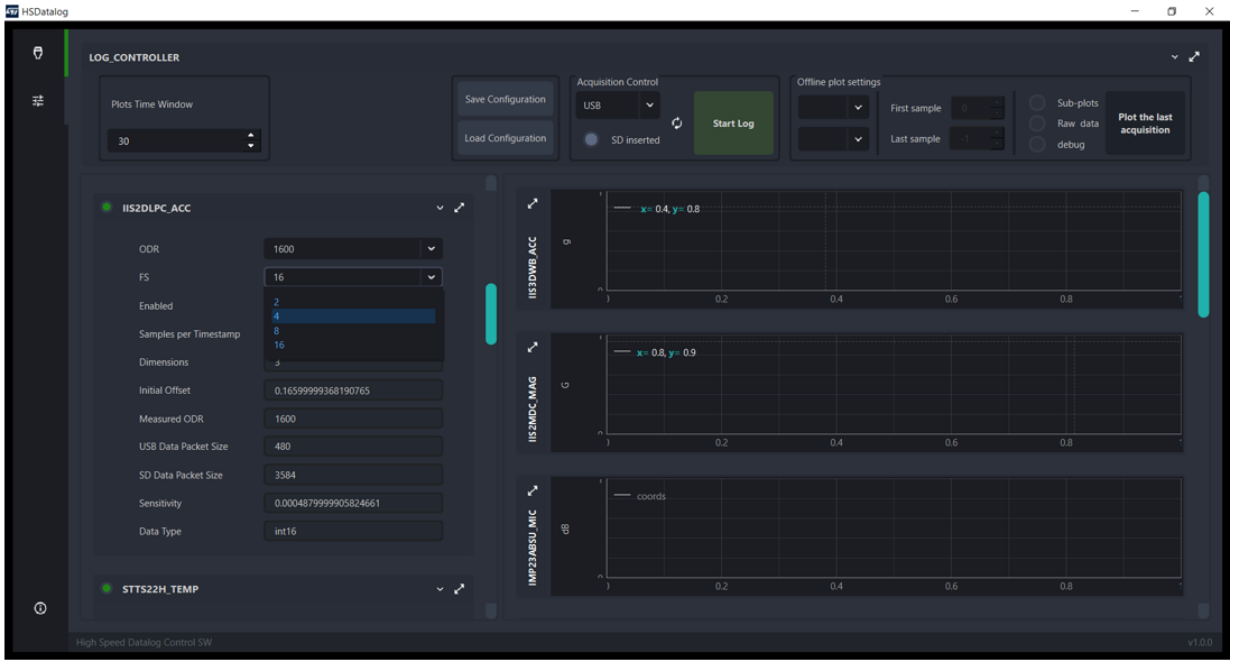



**Figure 30. Real Time Plot GUI - Sensor parameters configuration**




The Real Time Plot GUI allows also to:

- Send a UCF configuration file and visualize outputs
- Set up tag classes and handle data tagging and labelling of an ongoing acquisition
- Enable FFT calculation for accelerometers and microphones

- Set up the acquisition name and description

**Figure 31. Real Time Plot GUI - Data streaming and labelling**

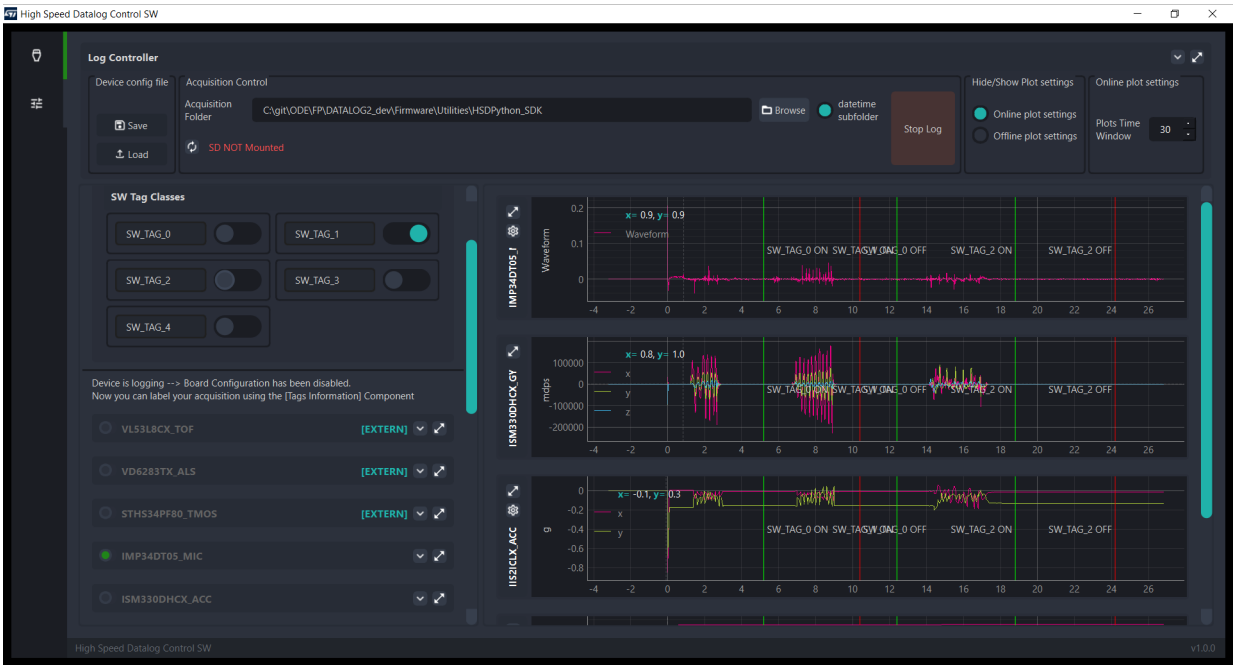
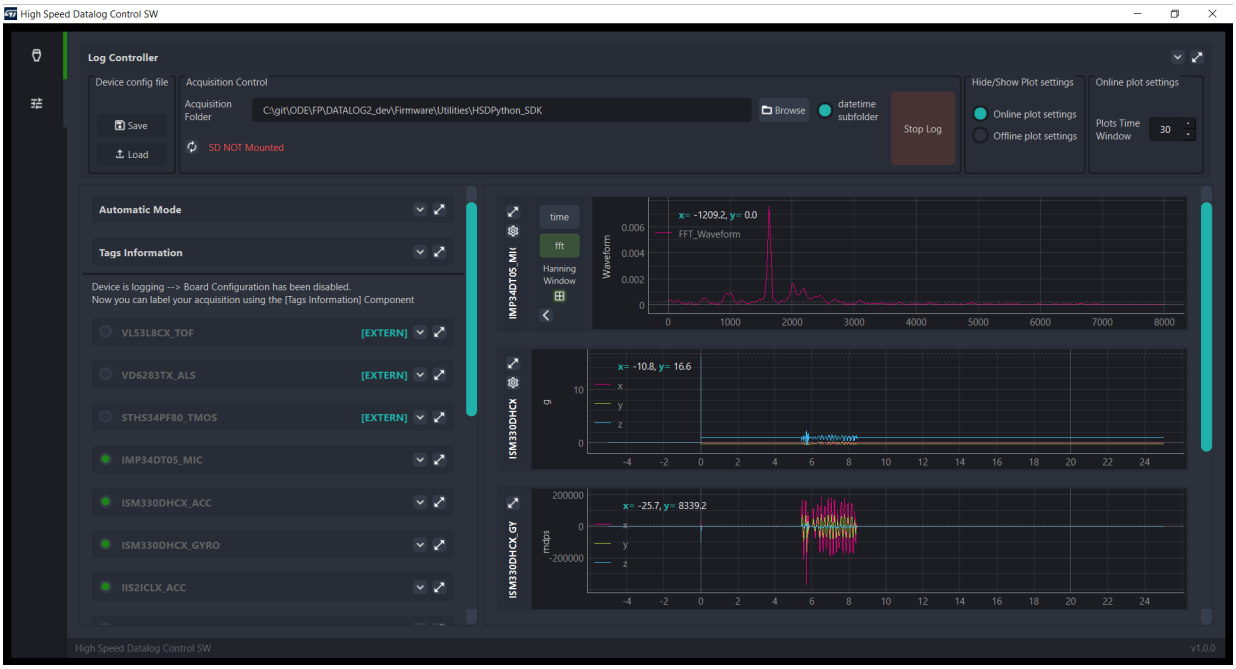



**Figure 32. Real Time Plot GUI - Data streaming and FFT calculation**




The GUI is designed using independent widgets that you are free to pop out, zoom, open and close as you wish.

To pop out the widget, click on the arrow symbol on the upper-left corner.

To zoom a graph, use the mouse wheel while pressing CTRL key. You can zoom only horizontally or vertically (by pointing the cursor on the related axes) or zoom the entire 2D figure (by pointing the cursor on the graph).

To reset the widget to the default setup, move the cursor on the lower-left corner and click on the A button that appears.

**Figure 33. Real Time Plot GUI - Floating widgets**


FP-SNS-DATALOG2 provides also PDetect, a dedicated example for human presence and motion detection. A new set of enriched widgets and sensor parameters has been added to fully support the sensors available.

**Figure 34. Real Time Plot GUI - PDetect widgets (1 of 2)**

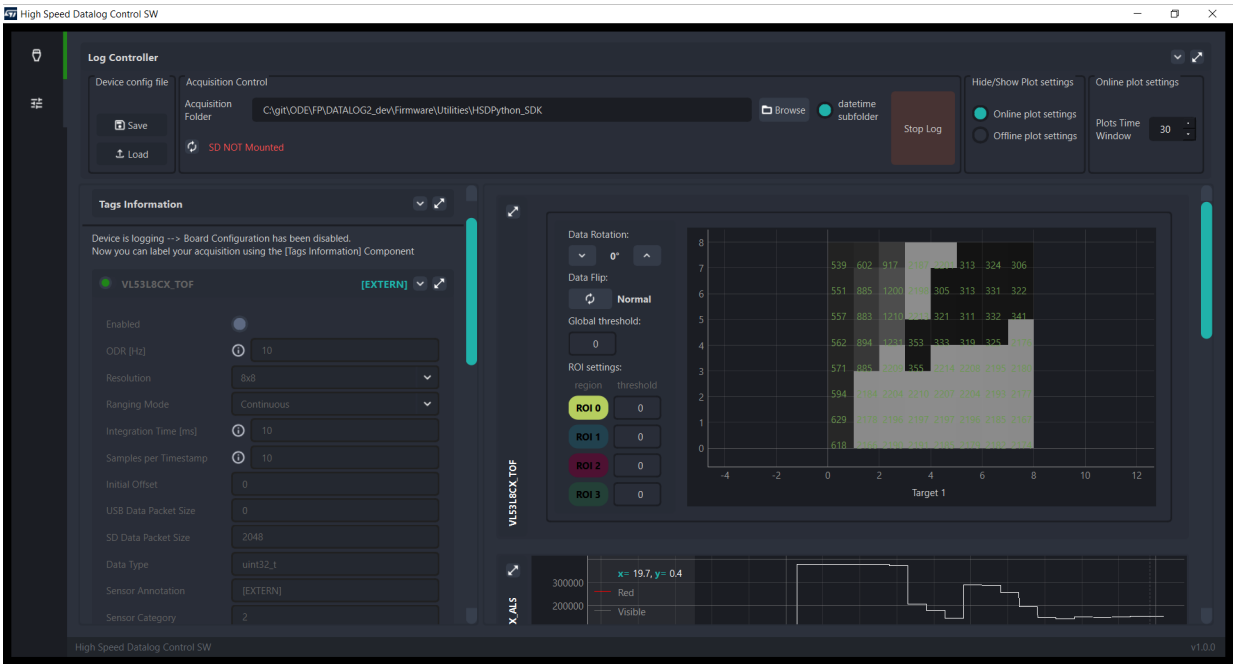
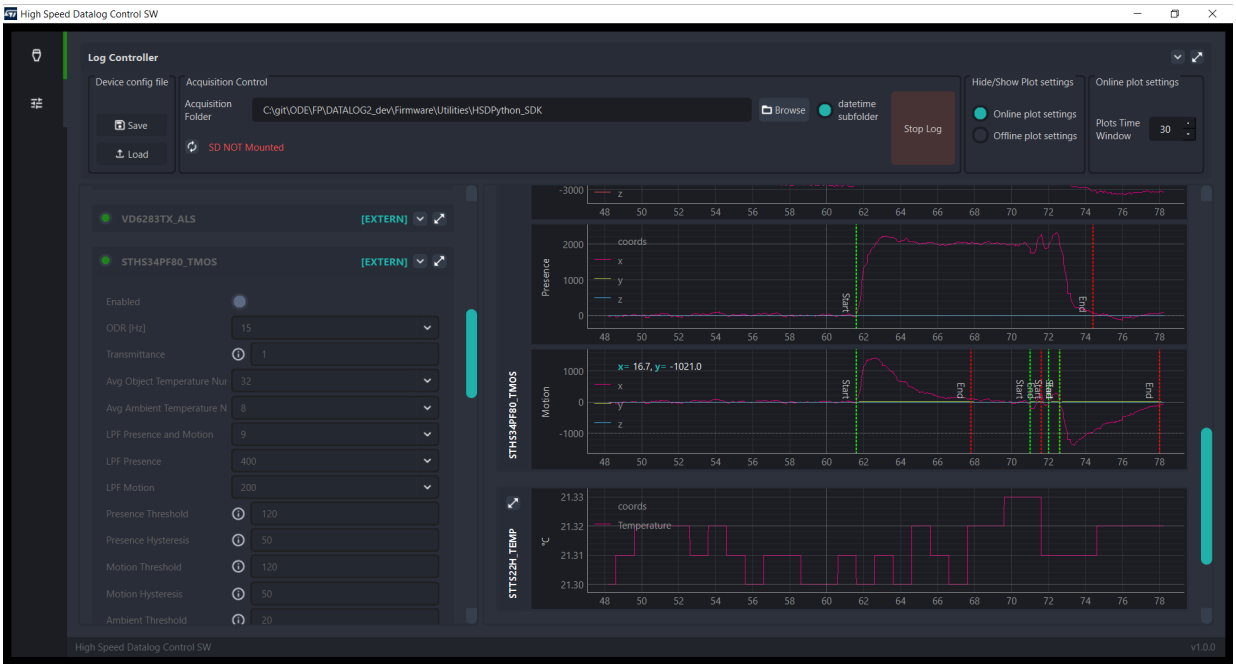


Figure 35. **Real Time Plot GUI - PDetect widgets (2 of 2)**




Once the test has been finished, you can also check and plot the desired dataset by clicking on the *Plot the last acquisition* button.

You can plot all the available sensors or select the one you are interested in, highlight a tag, create a sub-plot for each axis or visualize the sensor raw data.

Figure 36. **Real Time Plot GUI - Offline plot (1 of 2)**

Figure 37. **Real Time Plot GUI - Offline plot (2 of 2)**


From FP-SNS-DATALOG2 v3.0.0, HSDPython_SDK has been expanded and moved to a separate software product: STDATALOG-PYSDK. For full details, please see the full documentation available on the STDATALOG-PYSDK landing page.

### 2.2.4 STBLESensor app

DATALOG2 and PDetect applications can be controlled via Bluetooth® Low Energy using the STBLESensor app (both Android and iOS from v5.2 and above), which lets you manage the board and sensor configurations, start/stop data acquisition on SD card and control data labelling.

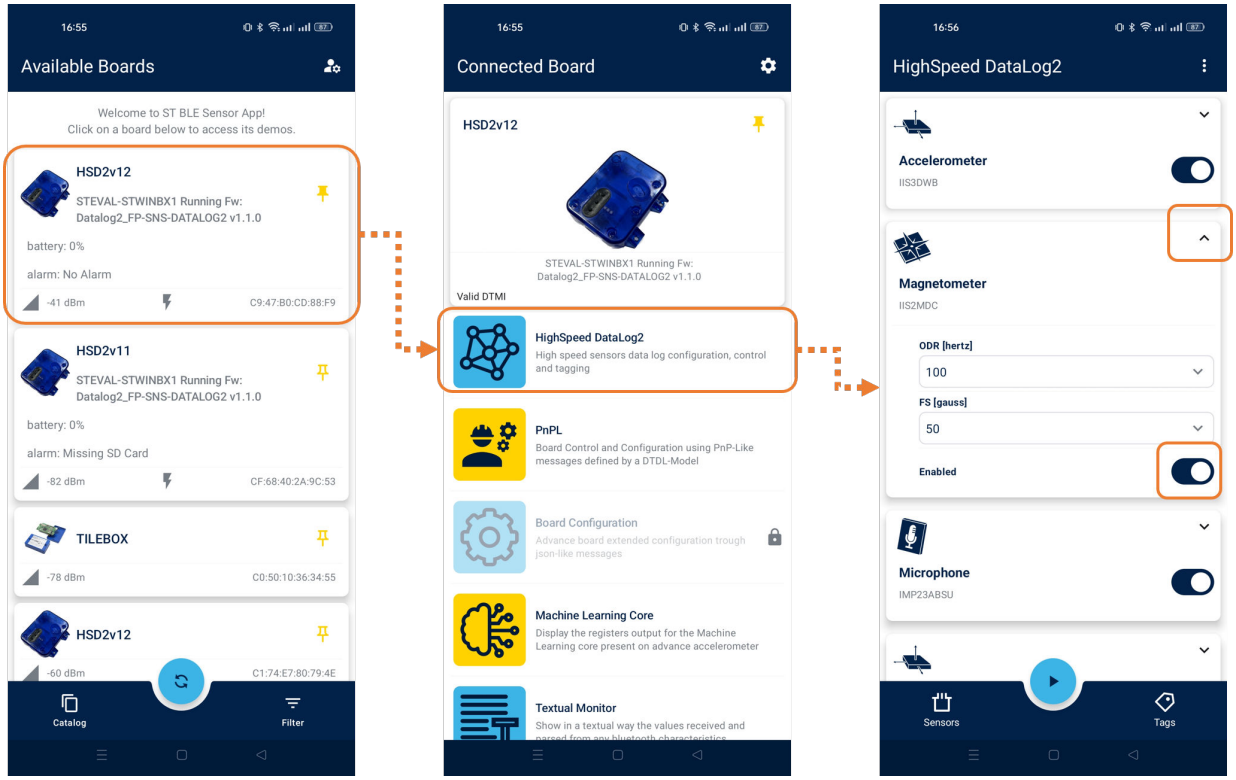
Through the STBLESensor app you can also set up an MLC or an ISPU and visualize MLC outputs.

The demo page contains two tabs (Configuration and Run), accessible through the bottom navigation bar.

Once connected, you can configure the device by:

- enabling/disabling a specific sensor
- changing sensor parameters, such as ODR, FS, etc.
- updating the "Board Alias"

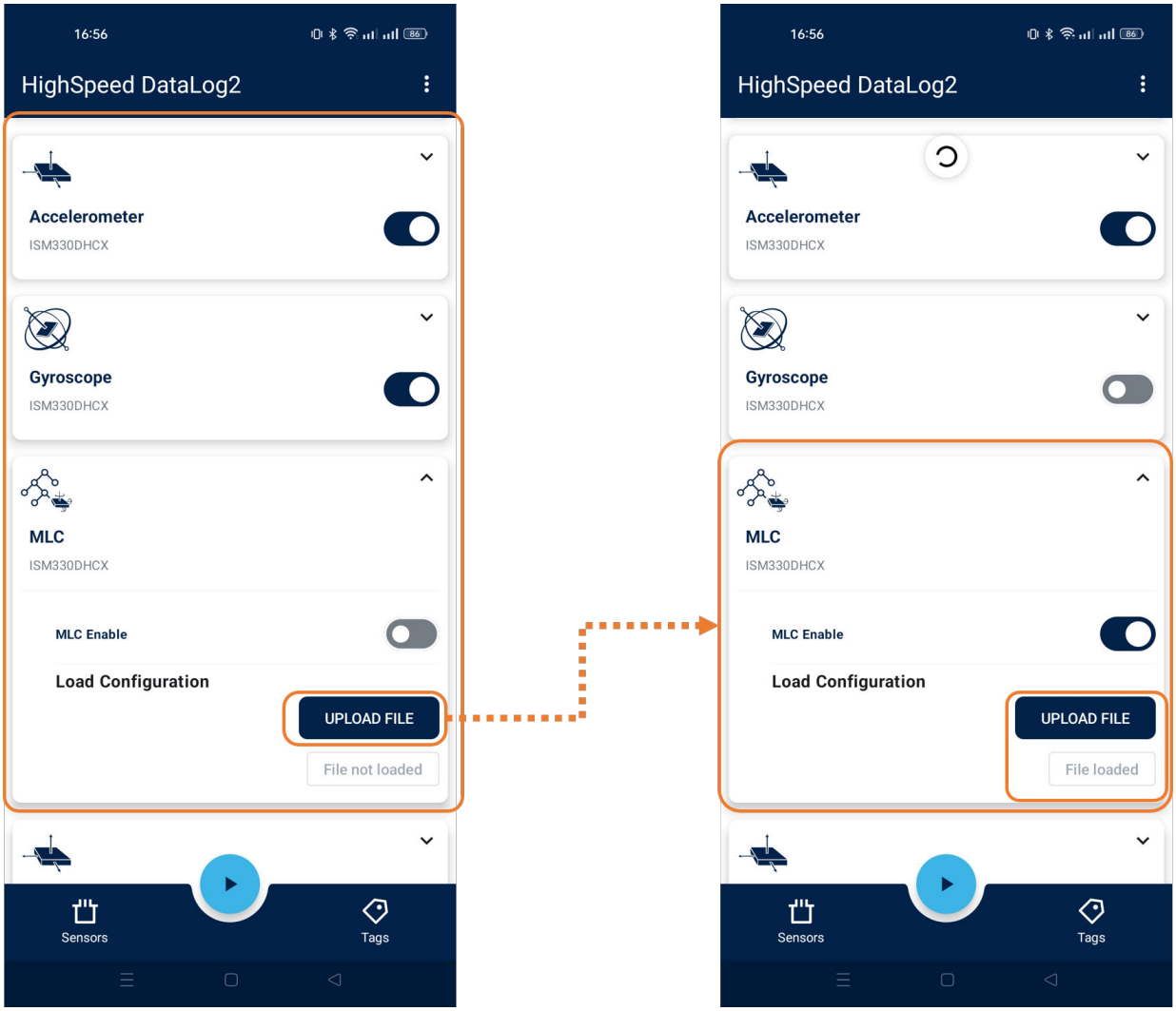
Figure 38. **STBLESensor configuration page**




The Configuration tab also allows:

- sending a UCF configuration file to set up an MLC or an ISPU. The UCF file can be retrieved either from the smartphone memory or from the cloud storage (Google Drive, Microsoft OneDrive, etc.)
- saving the current device configuration on the smartphone (JSON file)
- overwriting the default device configuration so that the new one is loaded automatically at power-on (an SD card is needed to use this feature)
- loading a specific device configuration (JSON file) from the smartphone

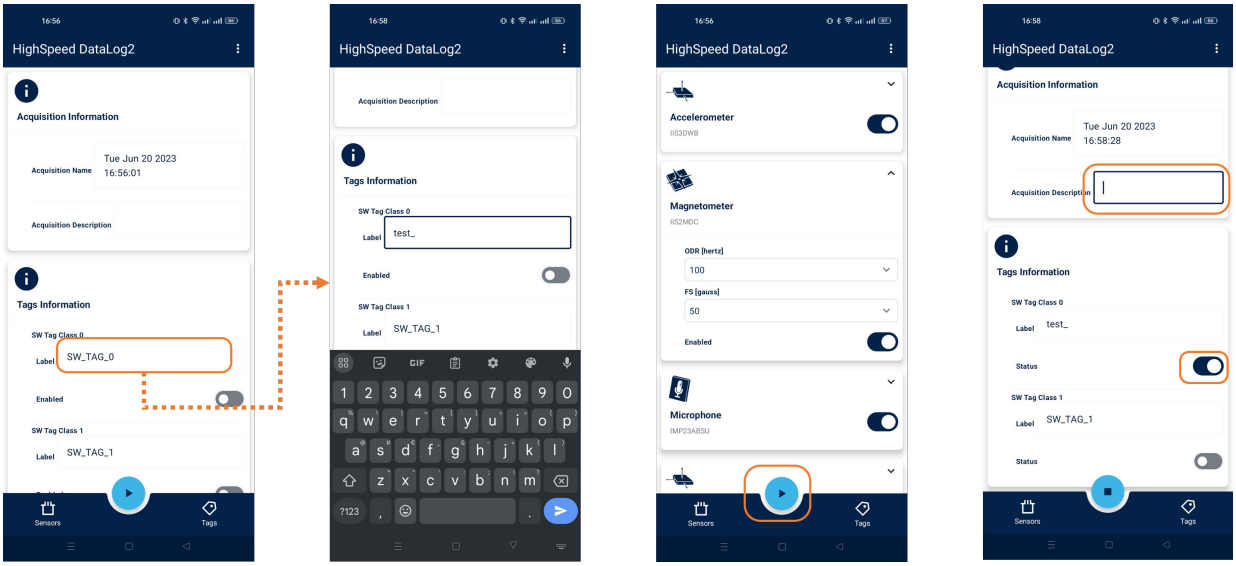
Figure 39. **STBLESensor configuration settings (1 of 2)**




By clicking on the Tags button, you can switch to the acquisition settings and control tab to:

- start and stop an acquisition (to an SD card)
- choose which tag class is used for the next acquisition
- handle data tagging and labelling of an ongoing acquisition
- set up the acquisition name and description

Figure 40. **STBLESensor configuration settings (2 of 2)**



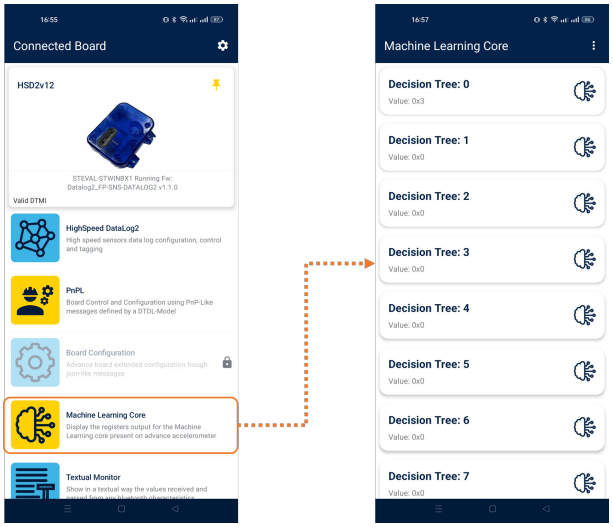

By tapping the start button, the data collection starts on the SD card. A YYYYMMDD_HH_MM_SS (for example, 20200128_16_33_00) folder that contains the raw data and the JSON configuration file (and eventually a UCF file) is created into the SD card.

For further details on the acquisition folder, see Section 2.6: Acquisition folders.

*Important: Do not unplug the SD card or turn the board off before stopping the acquisition to prevent data corruption on the SD card.*

If you have enabled the machine learning core, you can also visualize its output values in the machine learning core page. You just have to open the demo list of the app or by swiping from the left, and then selecting *Machine Learning Core*.

Figure 41. **STBLESensor Machine Learning Core**




### 2.2.5 Standalone

DATALOG2 and PDetect can also work standalone, saving sensor data at the desired rate into the SD card.

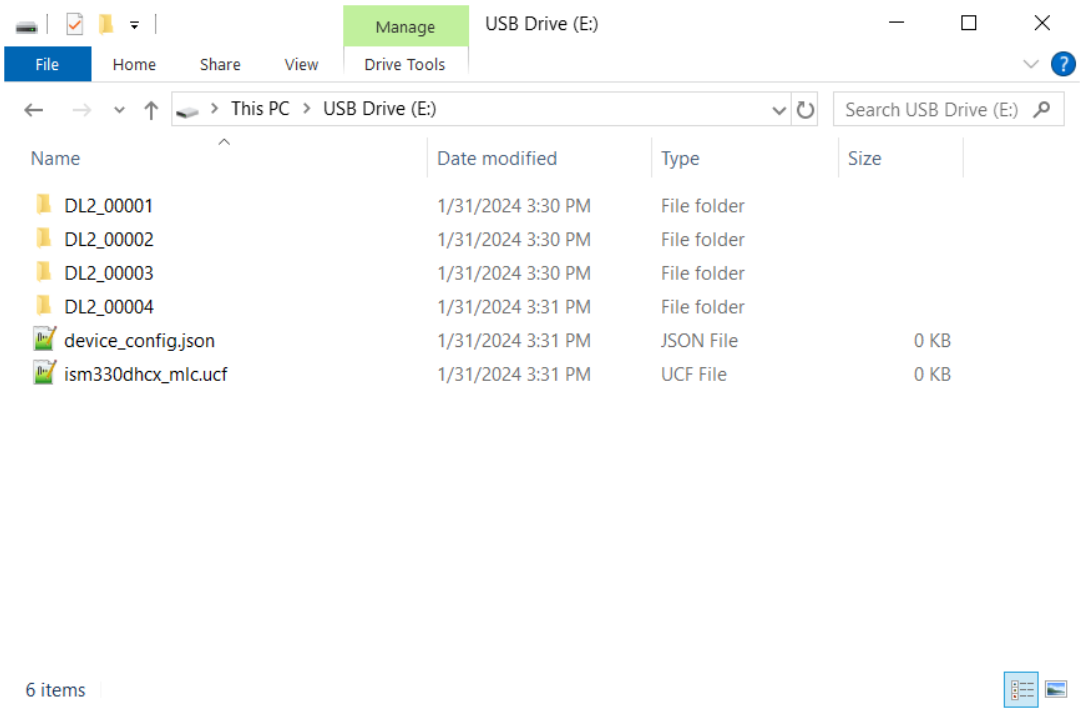
The firmware can also read custom sensor configuration from the SD card root folder. To do so, you can simply save a JSON configuration file in the root folder of the SD card.

In the same way, you can configure the MLC or the ISPU by saving the proper UCF in the root folder.

Once the firmware is already flashed on the board:

- Insert an appropriate SD card (see Section 2.2.7: SD card considerations)
- If the board is battery-powered and switched off, press the PWR button to switch on the board.
- Press the RESET button
  - If the SD card is not inserted properly, the orange LED is switched off. Check the SD card and restart the procedure.
    If instead the SD is inserted properly, the orange LED is switched on.
    When in idle state, the green LED is blinking slowly.
  - If a JSON configuration file is present in the root folder of the SD card, the device configuration is loaded from the file itself, setting up where available the sensors configuration, the device information, the tag properties and the automode.
  - If a UCF configuration file is present in the root folder of the SD card, the related MLC or the ISPU configuration is loaded onto the proper component.
- Press the USR button to start saving data. You can see the green LED blinking fast.
- To stop the data acquisition, press the USR button again.
- Remove the SD card and insert it into an appropriate SD card slot on your PC. The log files are stored in DL2_#####, where ##### is a sequential number determined by the application to ensure log file names are unique.

**Figure 42. Log file folders**




Each folder contains a .dat file for each enabled sensor, the device_config.json, the acquisition_info.json and the .ucf (if available).

For further details on the acquisition folder, see Section 2.6: Acquisition folders.

*Important: Do not unplug the SD card or turn the board off before stopping the acquisition to prevent data corruption on the SD card.*

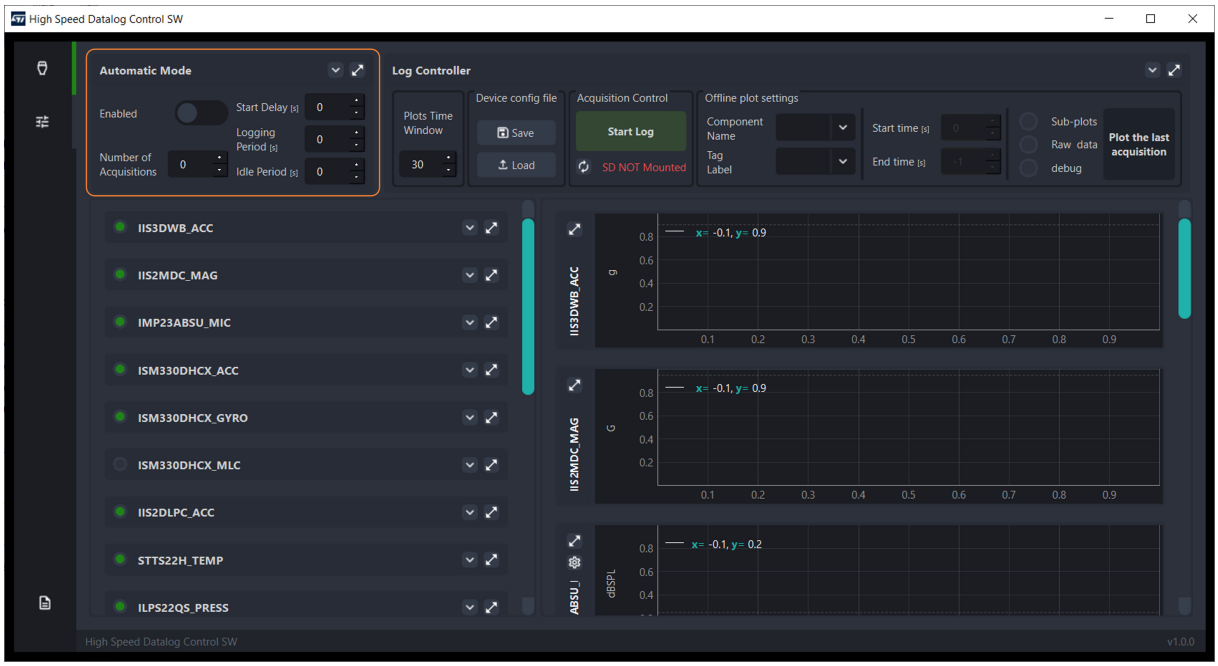
### 2.2.6 Automatic Mode (Automode)

The automode feature allows to automatically start and stop a sequence of acquisitions according to some parameters that can be specified in the device_config.json file or by using the stdatalog GUI. This mode can be useful if there is a need to log the data with a duty cycle or if there is a need to split a long acquisition into many smaller ones (this also helps avoid losing all data in case of an issue).

Automode allows automatically saving data on the SD card or to the PC via USB, generating different acquisitions folders.

The easiest way to set up the Automode is by using the dedicated section of the PC GUI that takes care of writing the correct parameters into the device_config.json file.

**Figure 43. Automode component**




The different parameters that can be configured in this component are:

- *enabled*: if true, the automode starts after the reset and node initialization. By default, enabled is false, so the automode is not enabled and all the other fields are ignored
- *nof_acquisitions*: gives the number of times the automode is executed; zero indicates an infinite loop and it is the default value
- *start_delay_s*: indicates the initial delay in seconds applied after reset and before the first execution phase starts when the automode is enabled
- *logging_period_s*: specifies the duration in seconds of the datalog phase
- *idle_period_s*: specifies the duration in seconds of the idle phase

To avoid issues while opening SD card or handling files, a minimum delay of 3 seconds is automatically setup in the firmware

Further details on the remaining fields and on device_config.json are available in Section 2.6.1: device_config.json.

As for the standalone mode, to enable the automode you must place the proper device_config.json in the root folder of the SD card before switching on the board.

### 2.2.7 SD card considerations

DATALOG2 also allows you to benchmark your setup and verify whether the communication channel is working properly.

To test if the sensor data can be streamed correctly, before starting any acquisition you can first recompile the firmware setting the *#define HSD_USE_DUMMY_DATA* to 1 (in SensorManager_conf.h file).

When *HSD_USE_DUMMY_DATA* is enabled, a predefined test signal (a sawtooth signal generated with a loop counter) is streamed instead of the real sensor data.

Once this configuration has been enabled, start your benchmark acquisition as you wish (standalone mode, via STBLESensor app, etc.).

Once your testing datalog is ready, launch the Python script *stdatalog_check_dummy_data.py*, available in the STDATALOG-PYSDK.

The script checks if there are any issues on the testing signals acquired for each active sensor and prints the result on the screen.

Using large buffers is far more efficient than using small ones when writing data to the SD card.

As the data logging application may involve large volumes of sensor data, the selected micro-SD card must be capable of handling the data rates without issues.

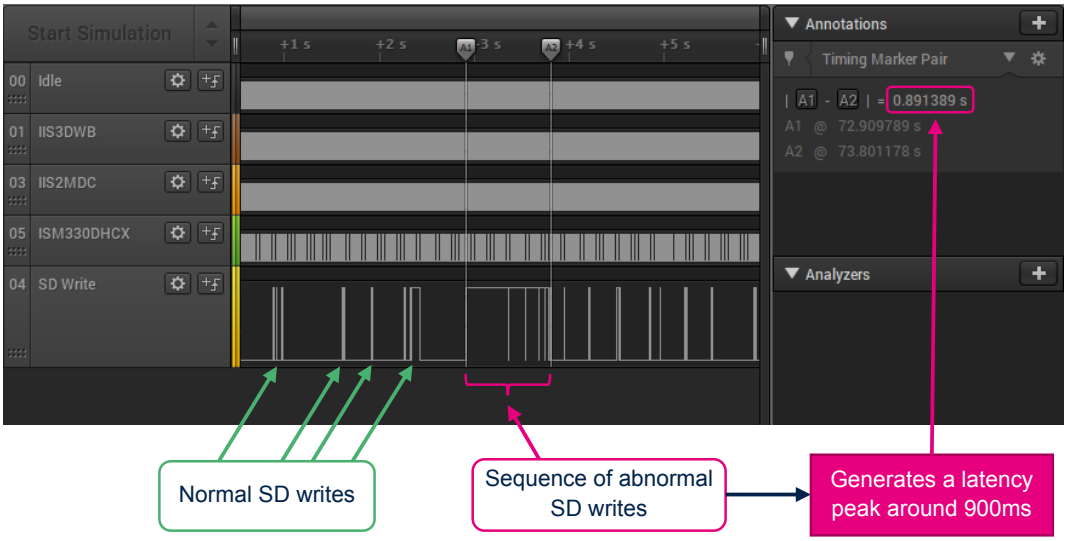
By default, DATALOG2 switches on and streams all the sensors at the highest sampling rate available, generating a big amount of data (about 6Mbit/s), but memory access time (and, consequently, the effective writing time) can drastically change depending on the SD card model used, thus impacting the reachable acquisition rate.

SD cards are designed to support an average writing throughput that may be even far above 6Mbit/s, but they also might present a time-varying latency with hundreds of milliseconds peaks.

Taken into consideration that the STM32 RAM memory is limited, the system might not be able to buffer enough data to compensate for the latency generated by the SD card writing process.

It is recommended to switch off the sensors you do not need so that the sizes of the RAM buffers are optimized and the overall available space is filled up with the relevant data from the selected sensors; in this way, the system can handle higher latency peaks caused by the SD card.

**Figure 44. SD "write buffer" instructions duration measured with a logic analyzer**




The application has been tested with the following SD cards, formatted FAT32 with 32 KB allocation table:

- SanDisk 32 GB Ultra HC C10 U1 A1 (p/n SDSQUA4-032G-GN6MA)
- Verbatim 16 GB Class 10 U1 (p/n 44082)
- Transcend Premium 16 GB U1 C10 (TS16GUSDCU1)
- Kingston 8 GB HC C4 (SDC4/8 GB)

*Note: Smaller allocation tables may impact performance.*

From FP-SNS-DATALOG2 v3.0.0, HSDPython_SDK has been expanded and moved to a separate software product: STDATALOG-PYSDK. For full details, please see the full documentation available on the STDATALOG-PYSDK landing page.

### 2.2.8 Data labelling

Labelled data are a group of samples that have been tagged with one or more labels. Labelled data are specifically useful in certain types of data driven algorithms such as supervised machine learning (for example, to create a labelled dataset for a classifier).

FP-SNS-DATALOG2 examples allow setting up labels to tag data live, during an acquisition.

The example supports: software tags, saved in a separate file called acquisition_info.json, available in the acquisition folder.

Software tags are enabled/disabled manually through the STBLESensor app, the cli_example application on the PC, or the stdatalog GUI.

Figure 45. **Tags configuration using the STBLESensor app or the CLI**

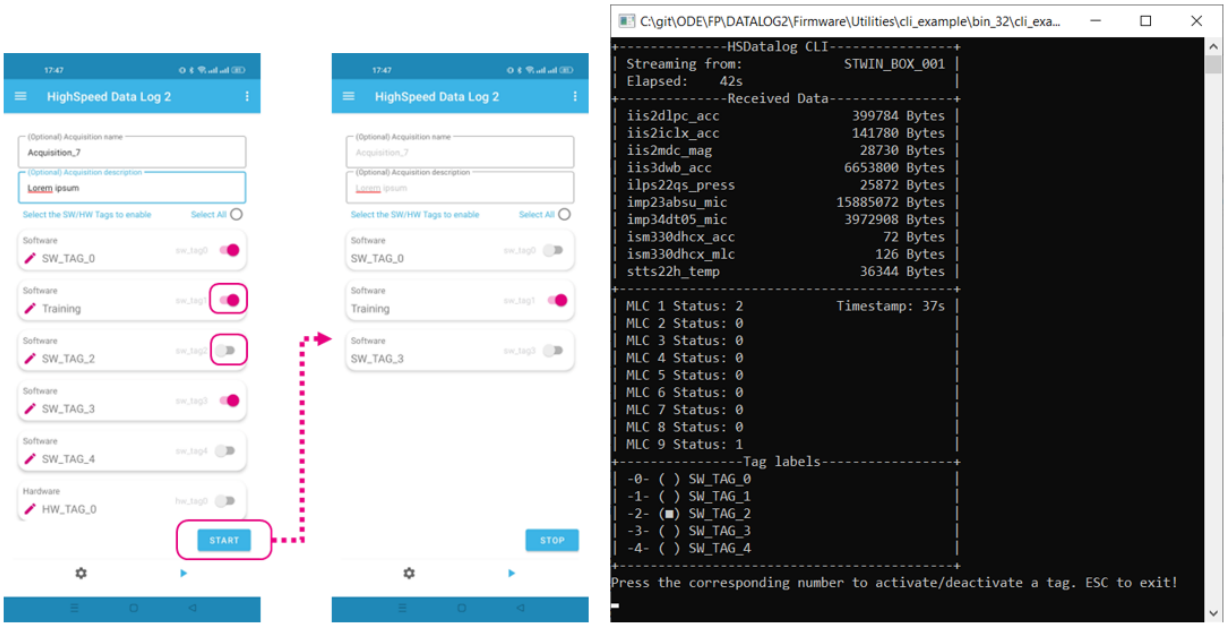



Figure 46. **Tags configuration using Python GUI**


Tags can also be customized, by changing the label name or switching them on/off.

This feature is available in the STBLESensor app and stdatalog GUI, but also handled in the SD card standalone mode, by manually acting on the tags_info component inside the device_config.json file.

**Figure 47. Enabling tags in standalone mode through the device_config.json file**


```
{
    "tags_info": {
        "max_tags_num": 100,
        "sw_tag0": {
            "label": "SW_TAG_0",
            "enabled": true,
            "status": false
        },
        "sw_tag1": {
            "label": "SW_TAG_1",
            "enabled": true,
            "status": false
        },
        "sw_tag2": {
            "label": "SW_TAG_2",
            "enabled": true,
            "status": true
        },
        "sw_tag3": {
            "label": "SW_TAG_3",
```


The following figure shows a portion of an acquisition_info.json containing some tag references.

**Figure 48. acquisition_info.json with tag references**


```
"tags": [
    {
        "e": true,
        "l": "SW_TAG_0",
        "ta": "2022-10-17T13:03:24.479Z"
    },
    {
        "e": true,
        "l": "SW_TAG_1",
        "ta": "2022-10-17T13:03:26.772Z"
    },
    {
        "e": false,
        "l": "SW_TAG_0",
        "ta": "2022-10-17T13:03:30.217Z"
    },
    {
        "e": true,
        "l": "SW_TAG_3",
        "ta": "2022-10-17T13:03:31.336Z"
    },
    {
        "e": false,
        "l": "SW_TAG_3",
```


For a complete description of device_config.json and acquisition_info.json, refer to Section 2.6: Acquisition folders.

## 2.3 FTP Data Retrieving over Wi-Fi

For STEVAL-STWINBX1 only, DATALOG2 example enables data retrieving from SD card via FTP protocol over Wi-Fi. To activate the FTP server, you just need to setup the Wi-Fi connection using the BLESensor app.

**Figure 49. Wi-Fi setup (1 of 2)**

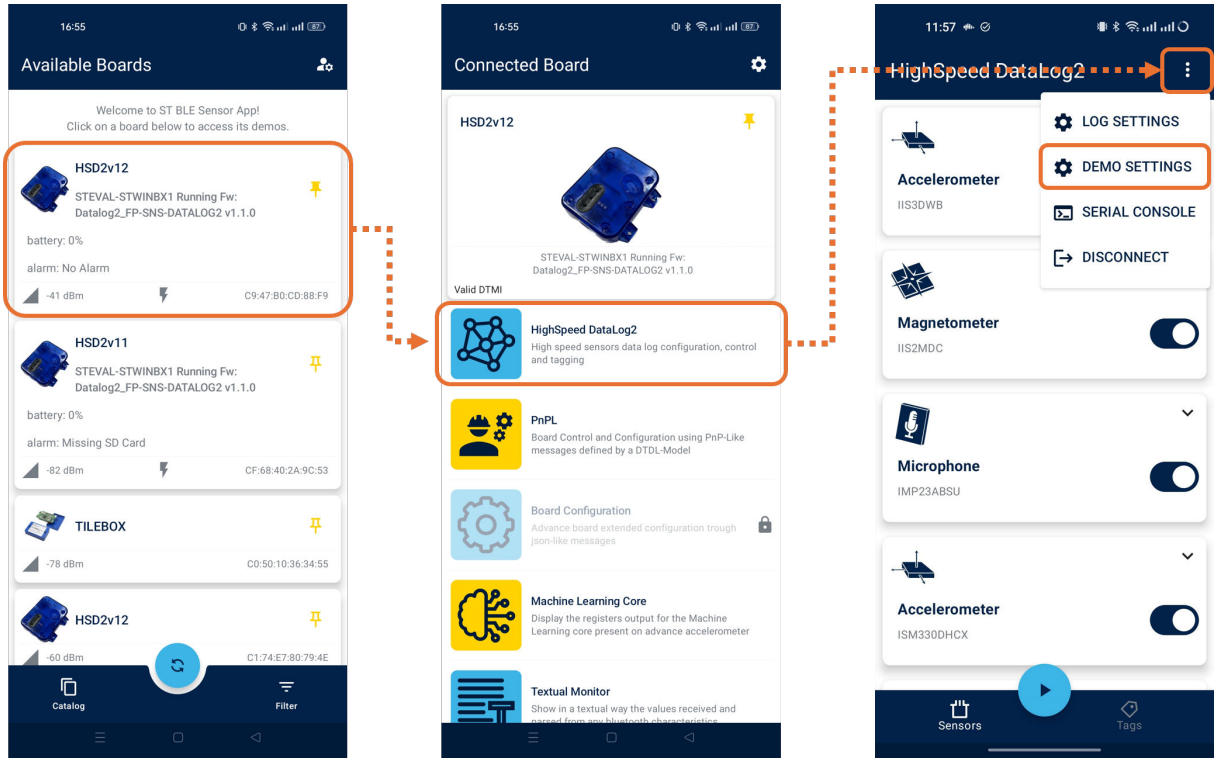
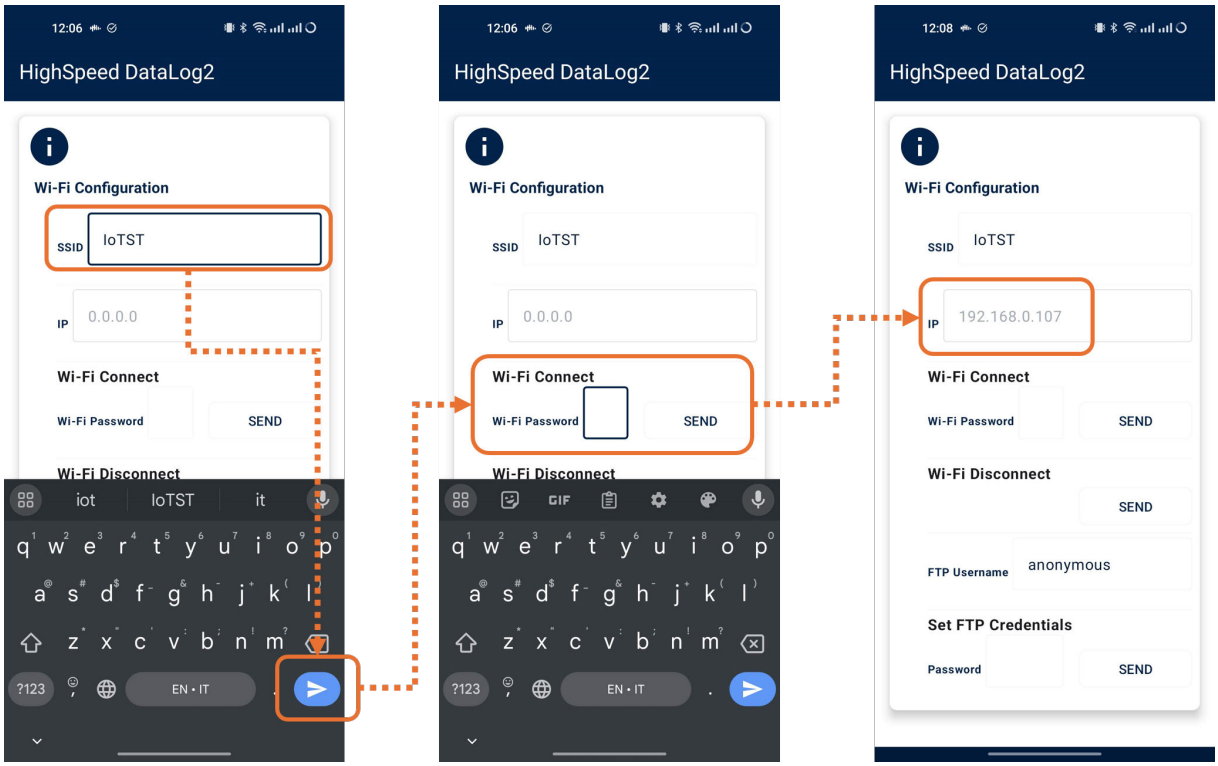



**Figure 50. Wi-Fi setup (2 of 2)**

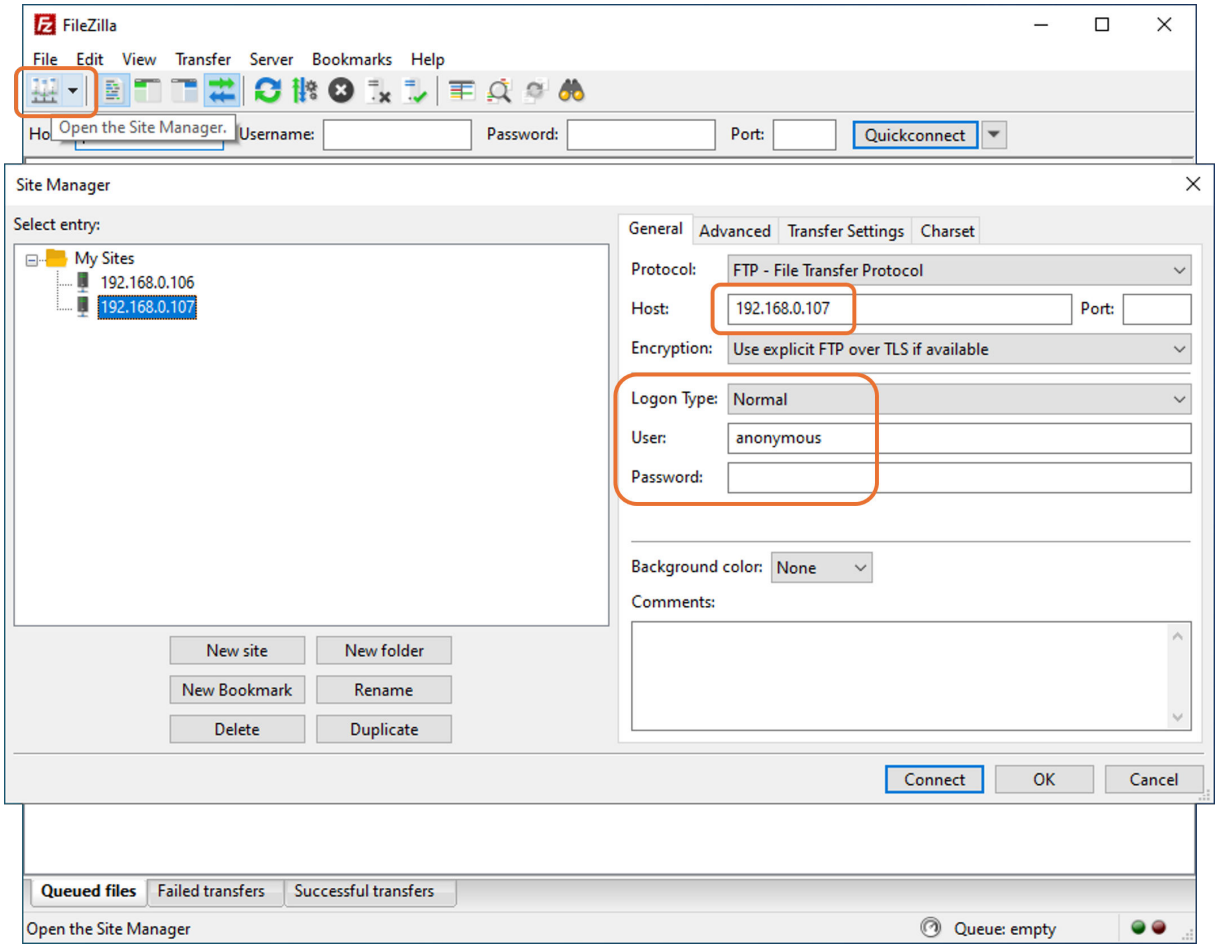


Once the Wi-Fi connection is established, the FTP Server instance is activated on the STWIN.box and it is possible to retrieve the data stored on the SDCard. You can use any FTP Client application, on PC or mobile, as long as the client device is connected to the same network as the board.

In the following example we use a Windows 10 PC with FileZilla FTP Client (https://filezilla-project.org/):

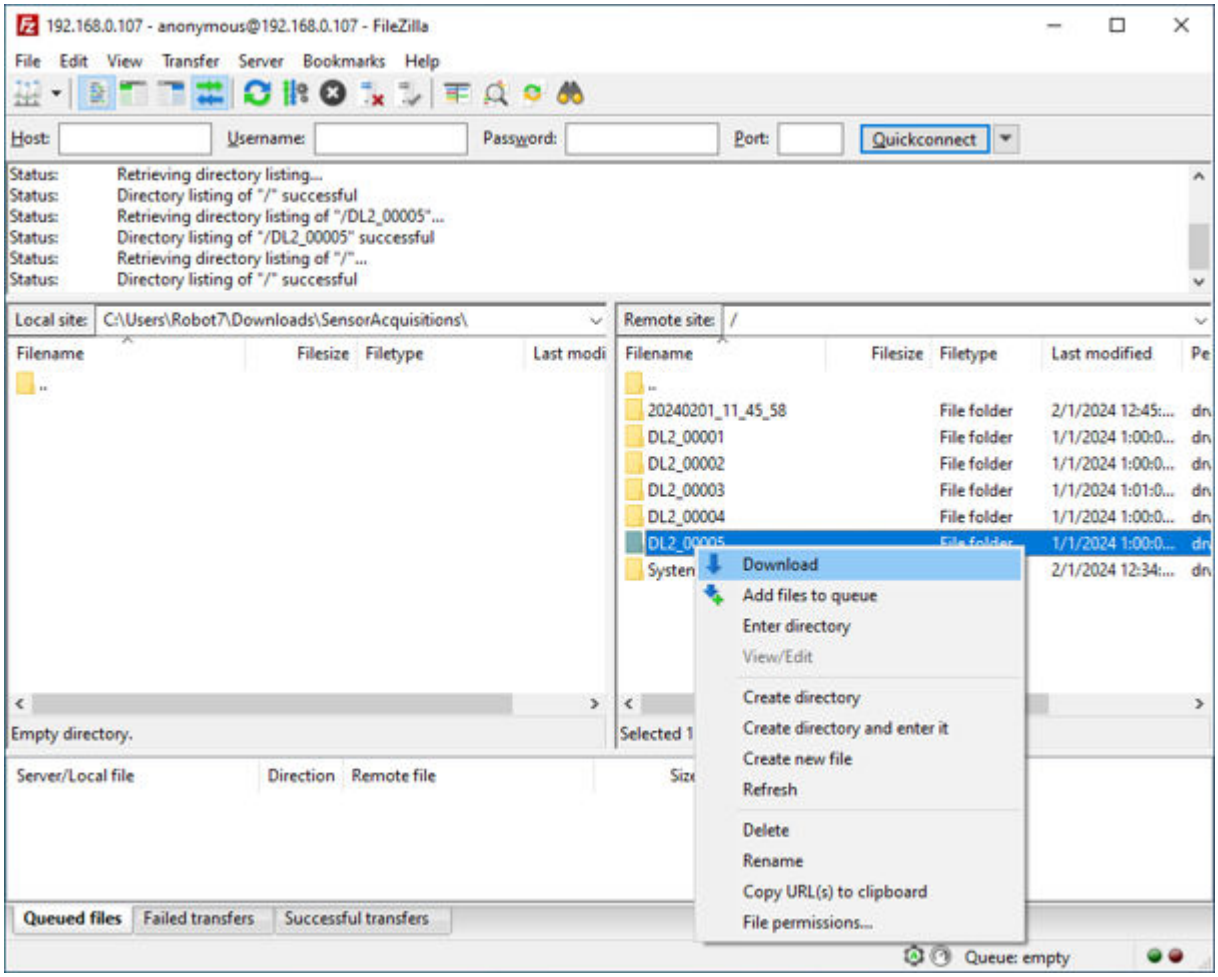
- Go to “Site Manager”
- For the first connection select “New Site”
  - Host: IP address (you can read it from the mobile app)
  - Logon Type: Normal
  - User: “anonymous”
  - Password: leave empty
- Click OK

**Figure 51. FTP client setup (1 of 2)**




The list of available folders and files will appear on the right. Right click on one acquisition folder and select “Download” to copy to the selected folder to the PC (left).

Figure 52. **FTP client setup (2 of 2)**




It is recommended to transfer the entire folder and not just one single file.

*Note: See Section 2.6: Acquisition folders for the full details on DATALOG2 folders content.*

## 2.4 Ultrasound FFT for STEVAL-STWINBX1

UltrasoundFFT example represents the evolution of the well-known example available for STEVAL-STWINKT1B in the X-CUBE-MEMSMIC1 package.

The example has been redesigned to support the STEVAL-STWINBX1 based on the eLooM framework.

In particular, SensorManager, EMData, and DPU eLooM components have been used to obtain the required processing chain.

Ultrasound_FFT streams to the PC the real-time FFT of the analog microphone signal, sampled at 192 kHz. The application uses a fixed FFT length of 512.

The GUI has also been redesigned, based on the same Python framework shown in the Real Time Plot GUI. An additional channel for the raw microphone data streaming has been added.

**Figure 53. Additional channel**


Following the same procedure described in 2.3.1.2, once launched the acquisition, you can:

- Enable/disable FFT or microphone data
- Click on the [**Connect**] button to allow the connection between the board and the PC
- Start/stop logging data on the PC

Once clicked on the [**Start Log**] button, data are live plotted. The application creates a YYYYMMDD_HH_MM_SS (for example, 20200128_16_33_00) folder containing the data and the JSON configuration file.

For further details on the acquisition folder, see Section 2.6: Acquisition folders

## 2.5 From HSDPython_SDK to STDATALOG-PYSDK

From FP-SNS-DATALOG2 v3.0.0, HSDPython_SDK has been expanded and moved to a separate software product: STDATALOG-PYSDK.

The STDATALOG-PYSDK is a comprehensive Python framework designed to facilitate the capture, processing, and visualization of data from a wide range of sources, including sensors, algorithms, simulated signals, and telemetry from actuators.

It is compatible with all firmware examples available in FP-SNS-DATALOG2, FP-IND-DATALOGMC, and FP-SNS-DATALOG1.

The python software development kit (SDK) for data logging has been developed using Python 3.12. To properly use it, Python 3.10, 3.11, or 3.12 must be already installed on the user's machine.

The STDATALOG-PYSDK requires different Python modules. The package is distributed with installers that solve all the required dependencies

For full details, please see the full documentation available on the STDATALOG-PYSDK landing page

## 2.6 Acquisition folders

When an acquisition is performed, both in SD and USB modes, DATALOGMC generates a folder in which you can find the following files:

- device_config.json
- acquisition_info.json
- raw data, saved into .dat files, whose name is based on the sensor name and type (for example, stts22h_temp.dat or ism330dhcx_gyro.dat)

### 2.6.1 device_config.json

The "device_config.json" file contains the information related to the devices that have contributed to the acquisition.

It is developed following a virtualization paradigm based on the Device Template Model concept.

*Note: For further details about Device Template Model, see Section 2.6.5: Device Template Model.*

It consists of the following fields:

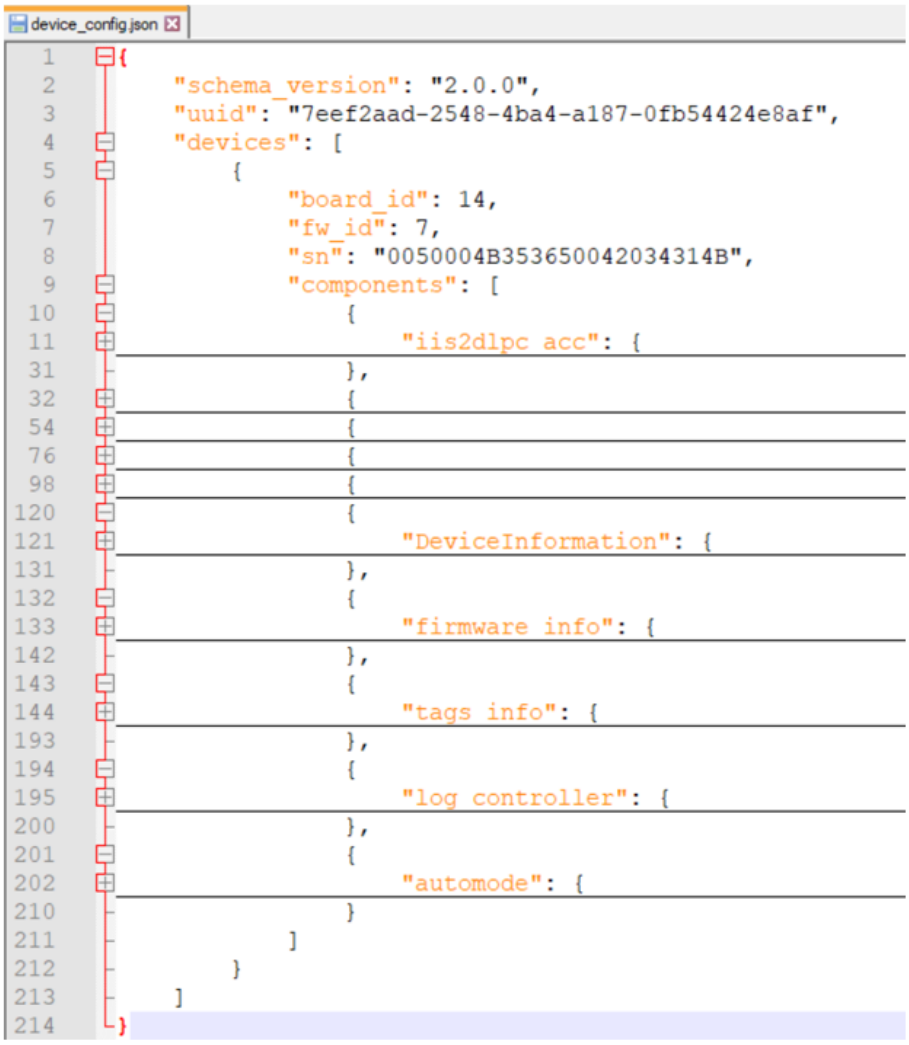
- *schema_version*, a reference to the schema version adopted
- *uuid*, the unique identifier of an acquisition, which is the same as the one within the "acquisition_info.json"
- *devices*, an array of devices that contribute to the same acquisition

In the DATALOG2 use case, *devices* consists of only one element, whose fields are:

- *board_id*, the unique identifier of a board type, according to ST catalogs
- *fw_id*, the unique identifier of a firmware type, according to ST catalogs
- *sn*, the unique serial number of the device that contributes to the acquisition
- *components*, an array of components of the device

An element of the *components* array can be a sensor, a descriptor, etc.

Figure 54. **device_config.json fields**




A sensor component is described by a struct, whose key is a unique name composed by the sensor part number and the sensor type (partNumber_sensorType; for example, iis2iclx_acc).

The value of the field is another struct, composed by different key-value couples describing the actual configuration of the single sensor (for example, whether the sensor is active or not, enumerative value for actual ODR and FS, time offset, data transmitted per unit of time, etc.).

The image below shows the *iis2iclx_acc* component.

Figure 55. **iis2iclx_acc component**

```
},
{
    "iis2iclx_acc": {
        "odr": 6,
        "fs": 3,
        "enable": false,
        "samples_per_ts": 800,
        "dim": 2,
        "ioffset": 0,
        "measodr": 0,
        "usb_dps": 166,
        "sd_dps": 1536,
        "sensitivity": 0.00012179999612271786,
        "data_type": "int16",
        "sensor_annotation": "",
        "sensor_category": 0,
        "st_ble_stream": {
            "id": 9,
            "acc": {
                "enable": false,
                "unit": "g",
                "format": "int16_t",
                "elements": 54,
                "channels": 2,
                "multiply_factor": 0.00012179999612271786,
                "odr": 833
            }
        },
        "c_type": 0,
        "stream_id": -1,
        "ep_id": -1
    }
},
```


*Note:* *odr and fs fields are enumerative values defined into the Device Template Model. Please refer to Section 2.6.5: Device Template Model and DATALOG2 DTM for the mapping between the enum values and the real ODR and Full Scale. A table providing the mapping for quick reference is also available here Section 2.6.6: ODR and FS mapping table .*

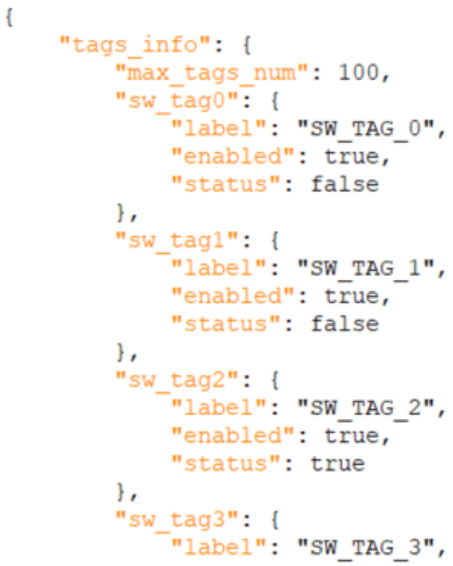
device_config.json contains also informative components, such as *log_controller, DeviceInformation* or *firmware_info* that describe the board and the firmware used for the related acquisition.

Figure 56. **Informative components**


```
{
    "DeviceInformation": {
        "manufacturer": "STMicroelectronics",
        "model": "STEVAL-STWINBX1",
        "swVersion": "2.0.0",
        "osName": "AzureRTOS",
        "processorArchitecture": "ARM Cortex-M33",
        "processorManufacturer": "STMicroelectronics",
        "totalStorage": 0,
        "totalMemory": 0
    }
},
{
    "firmware_info": {
        "alias": "STWIN_BOX_001",
        "fw_name": "FP-SNS-DATALOG2",
        "fw_version": "0.9.0",
        "serial_number": "STEVAL-STWINBX1",
        "device_url": "www.st.com/stwinbox",
        "fw_url": "www.st.com/dummy_place",
        "c_type": 2
    }
},
```


The *tags_info* component describes the labels activated by the user.

Figure 57. **tags_info component**

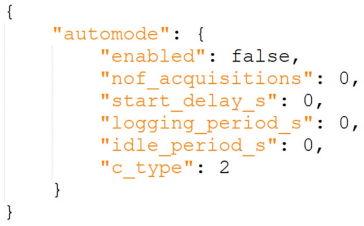



The available fields are:

- *label*: the tag name and can be customized by the user
- *enabled*: if true, the tag is available and can be used during the acquisition; if false, it is disabled. It can be set up by the user
- *status*: is the last available status

The *automode* component can be used to set up and enable the automode for the SD card acquisition (see Section 2.2.6: Automatic Mode (Automode)).

Figure 58. **Automode component**




The different parameters that can be configured in this component are:

- *enabled*: if true, the automode starts after the reset and node initialization. By default, enabled is false, so the automode is not enabled and all the other fields are ignored
- *nof_acquisitions*: gives the number of times the automode is executed; zero indicates an infinite loop and it is the default value
- *start_delay_s*: indicates the initial delay in seconds applied after reset and before the first execution phase starts when the automode is enabled
- *logging_period_s*: specifies the duration in seconds of the datalog phase
- *idle_period_s*: specifies the duration in seconds of the idle phase

### 2.6.2 acquisition_info.json

The “acquisition_info.json” file contains the information related to the acquisition and the tagged data (if any).

It consists of the following fields:

- *schema_version*: reference to the schema version adopted
- *dat_ext*: file extension for the data acquired
- *uuid*: unique identifier of an acquisition, same as the one within the “device_config.json”
- *name*: optional field for a name of an acquisition
- *description*: optional field for a description of an acquisition (blank by default)
- *start_time*: start acquisition time, in absolute time - ISO 8601 format
- *end_time*: end acquisition time, in absolute time - ISO 8601 format
- *interface*: enum identifying the interface used: "sd"|"usb"|"serial"|"ble"
- *tags*: array of tag elements, where
  - *e* is the status true or false
  - *l* is the label of the tag used
  - *ta* is when the tag has been released, in absolute time - ISO 8601 format

**Figure 59. acquisition_info.json**


### 2.6.3 ucf configuration file: MLC and ISPU

To set up the Machine Learning Core or the Finite State Machine, it is required a list of register configuration (register + data), saved in a text file with .ucf extension. You can build a ucf configuration file using the Unico-GUI tool or you can download a ready-to-use example from the official ST github (https://github.com/STMicroelectronics/STMems_Machine_Learning_Core).

Once the .ucf is available, you can pass this configuration file to the STWIN.box , STWIN and SensorTile.box PRO via Command Line Interface, via SD card, or via ST BLE Sensor app.

Figure 60. **ucf configuration file**

ism330dhcx_six_d_position.ucf

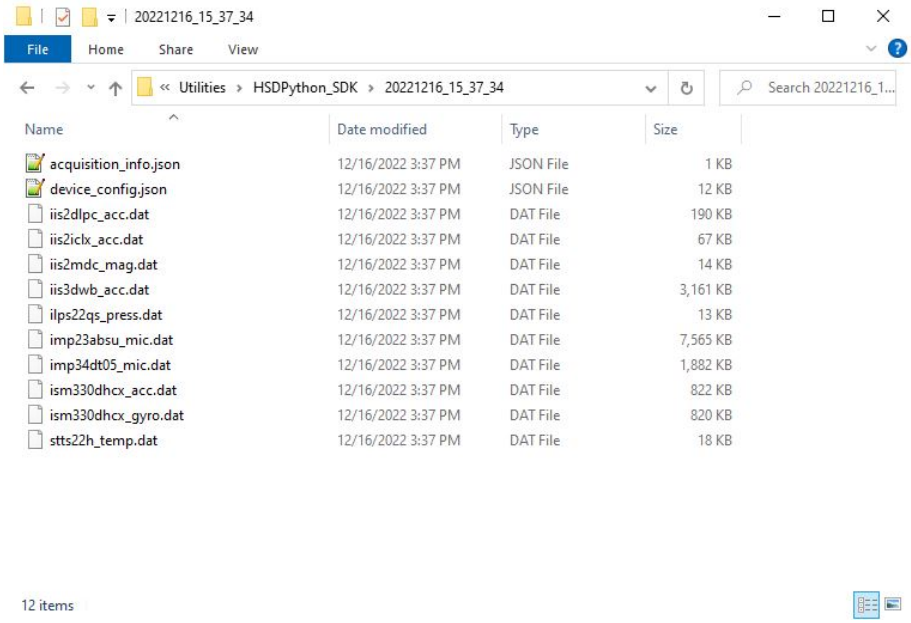
```
-- Machine Learning Core Tool v1.0.3.0 Beta, ISM330DHCX

Ac 10 00
Ac 11 00
Ac 01 80
Ac 05 00
Ac 17 40
Ac 02 11
Ac 08 EA
Ac 09 58
Ac 02 11
Ac 08 EB
Ac 09 03
Ac 02 11
Ac 08 EC
Ac 09 62
Ac 02 11
Ac 08 ED
Ac 09 03
Ac 02 11
Ac 08 EE
Ac 09 00
Ac 02 11
Ac 08 EF
Ac 09 00
Ac 02 11
Ac 08 F0
Ac 09 0A
Ac 02 11
Ac 08 F2
Ac 09 10
Ac 02 11
Ac 08 FA
Ac 09 3C
Ac 02 11
Ac 08 FB
```

### 2.6.4 Raw data files (.dat)

Sensor raw data are saved in files with .dat extension. The name of the file describes the sensor part number and the sensor type, as follows:

- Name: <sensor_name>_<subsensor_type>.dat
  - <sensor_name>: component part number
  - <subsensor_type>: acc, gyro, mag, temp, press, mic, mlc, ispu

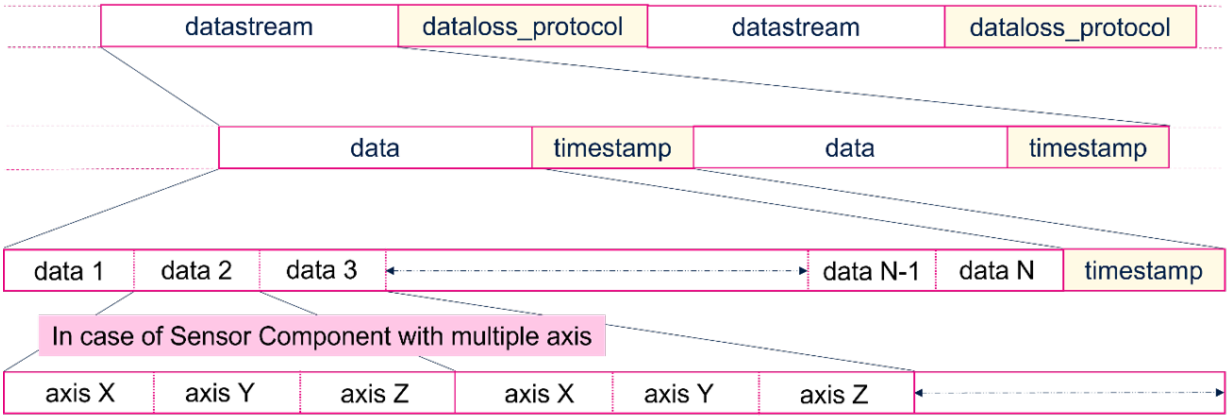
Figure 61. **Sensor raw data folder**




One file is generated for each sub-sensor. Composite sensors such as ISM330DHCX or ILPS22QS may thus generate multiple files. For example, ilps22qs_press.dat contains pressure raw data from ILPS22QS sensor or ism330dhcx_gyro.dat contains gyroscope raw data from the ISM330DHCX sensor.

A .dat file contains raw data and their timestamps. Related sensor configuration information is available in the device_config.json file. The data stream has the following structure:

Figure 62. **.dat file - data stream structure**




where

- dataloss_protocol is a 4-byte incremental counter that represents the total number of bytes streamed up to that point. It can be used to detect data packet loss
- “data k” (k = 1.. N) represents a sample generated by a sensor Component e.g., from an iis2dlpc_acc, an ism330dhcx_acc, and so on
- In case of sensor Components with multiple axis, such as motion and magnetic sensors (i.e., ism330dhcx_acc, iis2mdc_mag, iis2iclx_acc, iis3dwb_acc, ...) each “data k” packet is one sample for each axis, as in the following schema:| axis X | axis Y | axis Z |
- It is possible to know the number of axes of a sensor by looking at its status in the device_config.json searching for the **dim** field
- **Data_type** field is also available in the sensor attribute inside the device_config.json

- N corresponds to the **val** field of the **samples_per_ts** attribute available for each sensor Component within the device_config.json
- timestamp is a double value (8 bytes) calculated in seconds

### 2.6.5 Device Template Model

A Device Template Model (DTM) represents the model of a specific application.

It is defined by means of the “Digital Twins Definition Language v2” from Microsoft (full language specifications here: DTDLv2).

Each Device Model:

- Is composed of a list of Components
- Each Component is defined by an Interface
- Each Interface can define a collection of device Properties, Commands or Telemetry types

As an example, it is possible to imagine a Device Template Model as a list of Sensor Component each of which defines its Properties (e.g., ODR, FS, …).

In the same way, it is possible to describe software components, which, e.g., contain higher-level properties or describe complex algorithms.

Each Device Model must have a unique Model Identifier, which must respect the following format:

dtmi:namespace:name;version_number

Where:

- dtmi: is a reserved keyword (Digital Twin Model Identifier)
- namespace: a hierarchical class of elements in which each element has a name unique to that class. this allows to reuse the same names in different contexts (namespaces)
- name: Device Template name
- version_number: to keep track of all the versions.

Device Template Models can be created using the Azure IoT Central service.

This service allows the users to:

- Design Device Template Models using an intuitive GUI
- Generate the DTM complete DTDLv2 description
- Create Views (sort of Dashboards) able to show property values, send/receive commands and/or telemetries.

Each ST DTDL-based firmware has a device template model associated, which completely describes its capabilities, properties and behaviors.

All the available DTDL are organized in an online catalog, used both by STDATALOG-PYSDK and ST BLESensor app to retrieve information about the firmware.

The online catalog is available here: DTDL online catalog.

FP-SNS-DATALOG2 is an example of ST DTDL-based firmware, whose Device Template Model has been created respecting the above rules and using the Azure IoT Central service.

DATALOG2 DTM can be retreived here: DATALOG2 DTM.

### 2.6.6 ODR and FS mapping table

For quick reference, here the mapping for STWIN.box sensors in DATALOG2 application.

**Figure 63. Mapping table 1**

| IMP34ABSU | Enum | AOP [dBSPL] | Enum | ODR [Hz] |
|---|---|---|---|---|
| | 0 | 130 | 0 | 16000 |
| | | | 1 | 32000 |
| | | | 2 | 48000 |
| | | | 3 | 96000 |
| | | | 4 | 192000 |

Figure 64. **Mapping table 2**

| IIS3DWB | Enum | FS [g] | Enum | ODR [Hz] |
|---|---|---|---|---|
| | 0 | 2 | 0 | 26667 |
| | 1 | 4 | | |
| | 2 | 8 | | |
| | 3 | 16 | | |
| **STTS22H** | **Enum** | **FS [C]** | **Enum** | **ODR [Hz]** |
| | 0 | 100 | 0 | 1 |
| | | | 1 | 25 |
| | | | 2 | 50 |
| | | | 3 | 100 |
| | | | 4 | 200 |

Figure 65. **Mapping table 3**

| IIS2ICLX | Enum | FS [g] | Enum | ODR [Hz] |
|---|---|---|---|---|
| | 0 | 0.5 | 0 | 12.5 |
| | 1 | 1 | 1 | 26 |
| | 2 | 2 | 2 | 52 |
| | 3 | 3 | 3 | 104 |
| | | | 4 | 208 |
| | | | 5 | 416 |
| | | | 6 | 833 |
| **IIS2DLPC** | **Enum** | **FS [g]** | **Enum** | **ODR [Hz]** |
| | 0 | 2 | 0 | 12.5 |
| | 1 | 4 | 1 | 25 |
| | 2 | 8 | 2 | 50 |
| | 3 | 16 | 3 | 100 |
| | | | 4 | 200 |
| | | | 5 | 400 |
| | | | 6 | 800 |
| | | | 7 | 1600 |

Figure 66. **Mapping table 4**

| ILPS22QS | Enum | FS [hPa] | Enum | ODR [Hz] |
|---|---|---|---|---|
| | 0 | 1260 | 0 | 1 |
| | 1 | 4060 | 1 | 4 |
| | | | 2 | 10 |
| | | | 3 | 25 |
| | | | 4 | 50 |
| | | | 5 | 75 |
| | | | 6 | 100 |
| | | | 7 | 200 |

Figure 67. **Mapping table 5**

| ISM330DHCX | Enum | FS [g] | Enum | FS[mdps] | Enum | ODR [Hz] |
|---|---|---|---|---|---|---|
| | 0 | 2 | 0 | 125 | 0 | 12.5 |
| | 1 | 4 | 1 | 250 | 1 | 26 |
| | 2 | 8 | 2 | 500 | 2 | 52 |
| | 3 | 16 | 3 | 1000 | 3 | 104 |
| | | | 4 | 2000 | 4 | 208 |
| | | | 5 | 4000 | 5 | 416 |
| | | | | | 6 | 833 |
| | | | | | 7 | 1666 |
| | | | | | 8 | 3332 |
| | | | | | 9 | 6667 |

Full details for all the boards supported by FP-SNS-DATALOG2 are available in the DTDL online catalog.

# 3 System setup guide

The FP-SNS-DATALOG2, together with the suggested combination of STM32 and ST devices, can be used to develop specific AI and datalogging applications for early detection of warning signs of potential failure.

## 3.1 Hardware description

### 3.1.1 STEVAL-STWINBX1 evaluation kit

The STWIN.box (STEVAL-STWINBX1) is a development kit and reference design that simplifies prototyping and testing of advanced industrial sensing applications in IoT contexts such as condition monitoring and predictive maintenance.

It is an evolution of the original STWIN kit (STEVAL-STWINKT1B) and features a higher mechanical accuracy in the measurement of vibrations, an improved robustness, an updated bill of materials (BOM) to reflect the latest and best-in-class microcontroller unit (MCU) and industrial sensors, and an easy-to-use interface for external add-ons.

The STWIN.box kit consists of an STWIN.box core system, a 480mAh LiPo battery, an adapter for the ST-LINK debugger, a plastic case, an adapter board for DIL 24 sensors and a flexible cable.

The many on-board industrial-grade sensors and the ultra-low-power MCU enable applications that feature: ultra-low power, nine DoF motion sensing, wide-bandwidth vibration analysis, audio and ultrasound acoustic inspection, very precise local temperature, and environmental monitoring.

A rich set of software packages is available in source code. Optimized firmware libraries and a complete companion cloud application help to speed up the design cycle to develop end-to-end solutions.

The kit supports a broad range of connectivity options. For wired connectivity, it includes a USB Type-C® port that can be used for power supply, data transfer and STM32 programming via DFU, and an RS-485 transceiver. For wireless connectivity, the kit offers Bluetooth® Low Energy, Wi-Fi, and NFC options.

The STWIN.box also includes a 34-pin expansion connector for small form-factor daughter boards associated with the STM32 family, such as the STEVAL-C34KAT1, STEVAL-C34KAT2 and STEVAL-PDETECT1 expansion boards.

The STWIN.box is suitable for field trials, demonstrations, and proof of concept for industrial IoT applications that use ST software and third-party software.

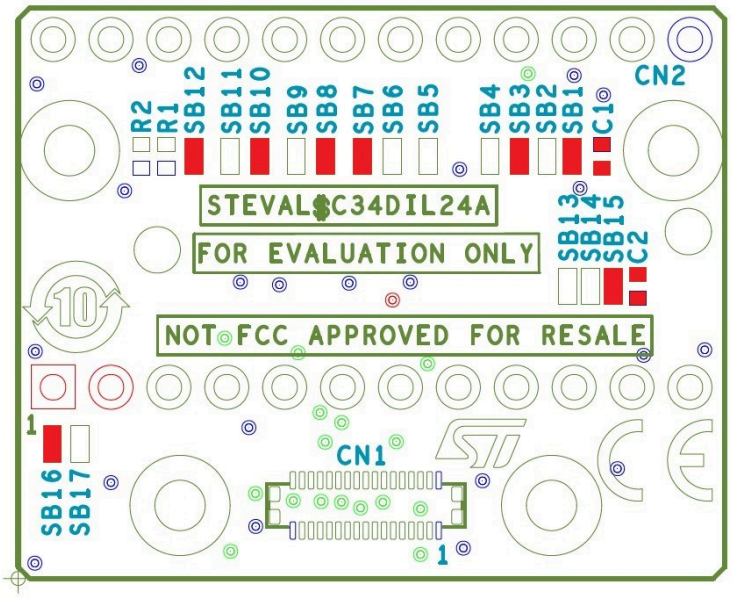
**Figure 68. STWIN.box mounted with the plastic case**

The included DIL24 adapter (STEVAL-C34DIL24) allows to expand the capabilities of the STWIN.box by giving the possibility to connect additional sensors. The DIL24 form factor is compatible with several STEVAL boards, like STEVAL-MKI230KA, STEVAL-MKI246KA or SENSEVAL-SCB4XV1.

Figure 69. **STWIN.box expanded with DIL24 adapter**

By default the DIL24 adapter is configured in SPI mode with the following solder bridges configuration:

Figure 70. **DIL24 adapter configured in SPI mode**



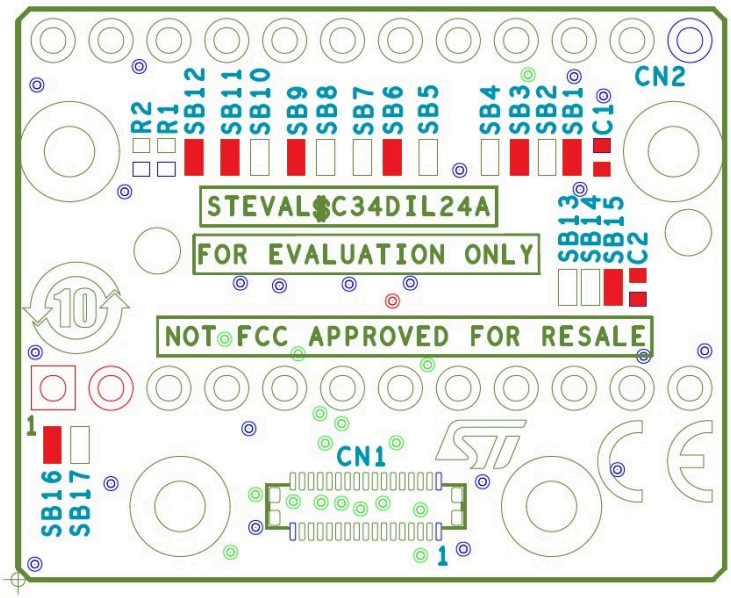

The default configuration is valid with STEVAL-MKI230KA, STEVAL-MKI245KA and STEVAL-MKI246KA.

If you need to use $I^2C$ sensors (for example with STEVAL-MKI223V1K or SENSEVAL-SCB4XV1), you need to modify the solder bridges as the following:

– Remove SB7, SB8, SB10

– Solder SB6, SB9, SB11

Figure 71. DIL24 I²C mode



### 3.1.2 STEVAL-STWINKT1B evaluation kit

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

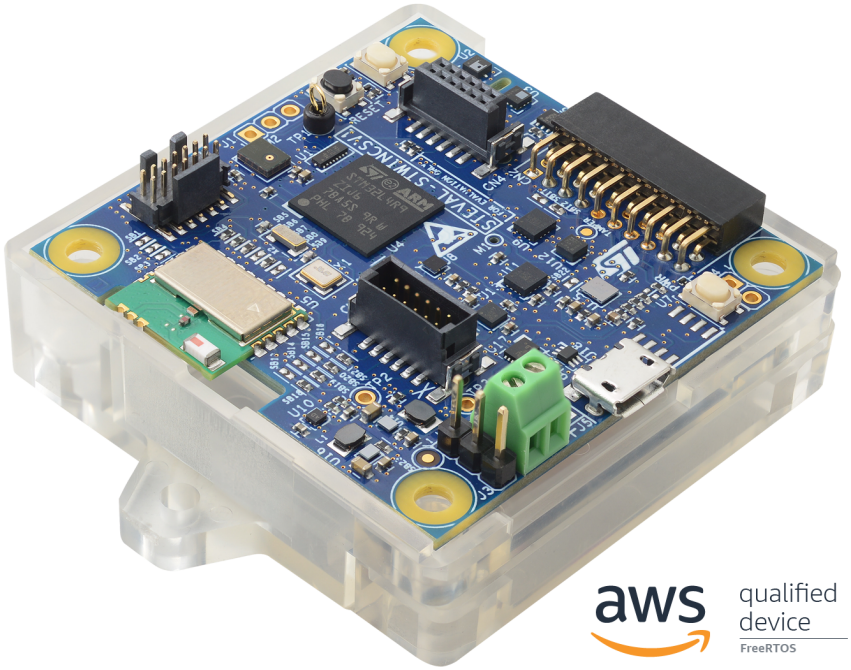
The kit features a core system board with a range of embedded industrial-grade sensors and an ultra-low-power microcontroller for vibration analysis of 9-DoF motion sensing data across a wide range of vibration frequencies, including very high frequency audio and ultrasound spectra, and high precision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

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

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

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

**Figure 72. STEVAL-STWINKT1B SensorTile Wireless Industrial Node**



### 3.1.3 STEVAL-MKBOXPRO evaluation kit

The STEVAL-MKBOXPRO (SensorTile.box PRO) is the new ready-to-use programmable wireless box kit for developing any IoT application based on remote data gathering and evaluation. It exploits the full kit potential by leveraging both motion and environmental data sensing, along with a digital microphone, and enhances connectivity and smartness in all environments.

The hardware node is a board that fits into the palm of your hand (40x63 mm) with a long-life 3.7 V 480 mAh rechargeable battery. The user can connect to the board via Bluetooth® by using the STBLESensor app (available both on Google Play and the Apple Store) on their smartphone and immediately build their own apps through a special interface. Apps can be developed quickly regardless of the level of expertise:

- **Entry mode**: play around with the default apps and see what STMicroelectronics sensors can achieve
- **Expert mode**: create your own app in a simple graphic environment
- **Pro mode**: develop code in an intuitive way using the STM32 open development environment (ODE) and STMicroelectronics function pack libraries

The kit board includes an ultra-low power programmable Bluetooth® Low Energy wireless SoC solution BlueNRG-LP 355AC, that is compliant with Bluetooth(R) Low Energy v5.2. This transmitter module is FCC (FCC ID: S9N-MKBOXPRO) certified and IC (IC: 8976C-MKBOXPRO; PMN: STEVAL-MKBOXPRO; HVIN: STEVAL-MKBOXPRO; FVIN: STSW-MKBOXPRO-BL) certified. The board also includes a wireless charger and a programmable NFC tag.

In addition, a DIL24 socket for easy MEMS adapter connection and a microSD™ card for storing data are available.

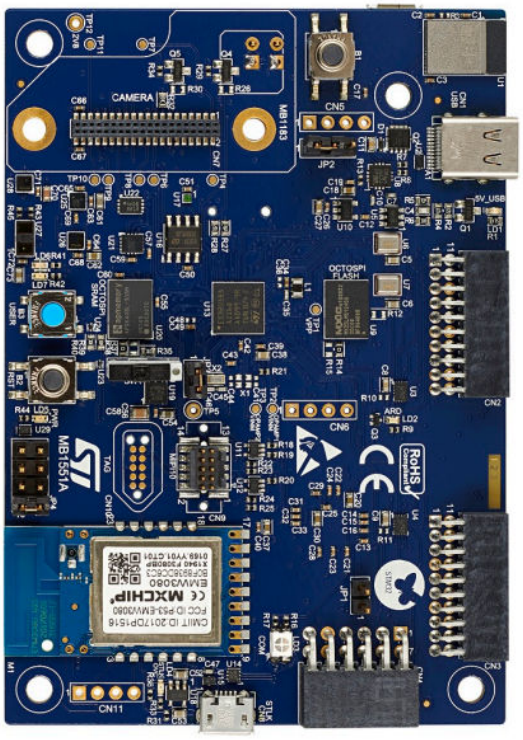
**Figure 73. STEVAL-MKBOXPRO (SensorTile.box PRO) multi-sensor and wireless connectivity development kit**

*Notice: For dedicated assistance, submit a request through our online support portal at **www.st.com/support**.*

### 3.1.4 B-U585I-IOT02A Discovery kit for IoT node

The B-U585I-IOT02A Discovery kit provides a complete demonstration and development platform for the STM32U585AI microcontroller, featuring an Arm® Cortex®-M33 core with Arm® TrustZone® and Armv8-M mainline security extension, 2 Mbytes of Flash memory and 786 Kbytes of SRAM, as well as smart peripheral resources. This Discovery kit enables a wide diversity of applications by exploiting low-power communication, multiway sensing, and direct connection to cloud servers. It includes Wi-Fi® and Bluetooth® modules, as well as microphones, temperature and humidity, magnetometer, accelerometer and gyroscope, pressure, time-of-flight, and gesture-detection sensors. The support for ARDUINO® Uno V3, STMod+, and Pmod™ connectivity provides unlimited expansion capabilities with a large choice of specialized add-on boards. For even more user-friendliness, the on-board STLINK-V3E debugger provides out-of-the-box loading and debugging capabilities, as well as a USB Virtual COM port bridge. The B-U585I-IOT02A Discovery kit leverages the STM32U5 Series key assets to enable prototyping for a variety of wearable or sensor applications in fitness, metering, industrial, or medical, with state-of-the-art energy efficiency and higher security

**Figure 74. B-U585I-IOT02A Discovery kit for IoT node**



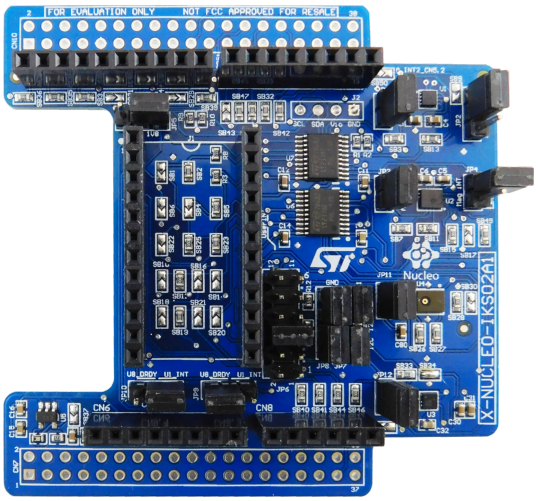
### 3.1.5 X-NUCLEO-IKS02A1 motion and microphone expansion board

The X-NUCLEO-IKS02A1 industrial motion MEMS sensor expansion board is compatible with the Arduino UNO R3 connector layout.

It embeds the ISM330DHCX 3-axis accelerometer and 3-axis gyroscope, the IIS2MDC 3-axis magnetometer, the IIS2DLPC 3-axis accelerometer, the IMP34DT05 digital microphone.

The X-NUCLEO-IKS02A1 interfaces with the STM32 microcontroller via I²C pin, with the possibility of changing the default I²C port.

**Figure 75. X-NUCLEO-IKS02A1 motion and microphone expansion board**



### 3.1.6 NUCLEO-U575ZI-Q nucleo board

he STM32U5 Nucleo-144 board based on the MB1549 reference board (order codes NUCLEO-U575ZI-Q and NUCLEO-U5A5ZJ-Q) provides an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power-consumption features, provided by the STM32U5 microcontroller. The ST Zio connector, which extends the ARDUINO® Uno V3 connectivity, and the ST morpho headers provide easy expansion of the functionality of the STM32 Nucleo open development platform with a wide choice of specialized shields. The STM32U5 Nucleo-144 board does not require any separate probe as it integrates the STLINK-V3E debugger/programmer. The STM32U5 Nucleo-144 board comes with the STM32 comprehensive free software libraries and examples available with the STM32CubeU5 MCU Package

**Figure 76. NUCLEO-U575ZI-Q nucleo board**


### 3.1.7 X-NUCLEO-IKS4A1 Motion MEMS and environmental sensor expansion board for STM32 Nucleo

The X-NUCLEO-IKS4A1 is a motion MEMS and environmental sensor evaluation board kit consisting of the main board X-NUCLEO-IQS4A1, which hosts the motion MEMS and environmental sensors, and the detachable add-on board STEVAL-MKE001A, which hosts the Qvar swipe electrodes.

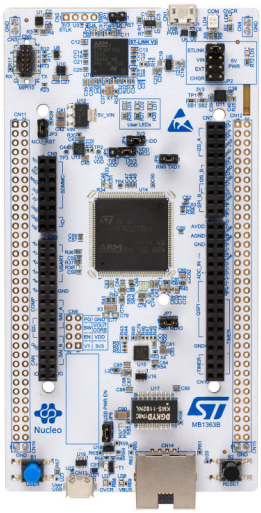
This expansion board allows application development with features like sensor HUB (LSM6DSO16IS and LSM6DSV16X), electronic image stabilization (EIS) and optical image stabilization (OIS) for camera applications through LSM6DSV16X SPI interface and Qvar touch/swipe gestures (thanks to the equipped electrode).

There is also the possibility to integrate presence and motion detection with an IR sensor as well as combining the features of multiple sensors through the DIL24 adapter.

### 3.1.8 NUCLEO-H7A3ZI-Q nucleo board

The STM32H7 Nucleo-144 boards based on the MB1363 reference board (NUCLEO-2H745ZI-Q, NUCLEO-H755ZI-Q, NUCLEO-H7A3ZI-Q) provide an affordable and flexible way for users to try out new concepts and build prototypes, by choosing from the various combinations of performance and power-consumption features provided by the STM32H7 microcontroller. The ST Zio connector, which extends the ARDUINO® Uno V3 connectivity, and the ST morpho headers provide an easy means of expanding the functionality of the Nucleo open development platform with a wide choice of specialized shields. The STM32H7 Nucleo-144 boards do not require any separate probe as they integrate the STLINK-V3E debugger/programmer. The STM32H7 Nucleo-144 boards come with comprehensive free software libraries and examples available with the STM32Cube MCU Package.

**Figure 77. NUCLEO-U575ZI-Q nucleo board**



## 3.2 Hardware setup

### 3.2.1 STEVAL-STWINBX1 , STEVAL-STWINKT1B and STEVAL-MKBOXPRO

The following hardware components are needed:

- a STEVAL-STWINBX1, STEVAL-STWINKT1B or STEVAL-MKBOXPRO development kit
- Laptop/PC with Windows 7, 8, or 10
- a microUSB + a USB Type-C® cables for STEVAL-STWINKT1B and STEVAL-MKBOXPRO
- 2 microUSB cables for STEVAL-STWINKT1B
- an SD card
- a smartphone with the STBLESensor app installed

*Note: DATALOG2 also supports STEVAL-PDETECT1, STEVAL-C34KAT1, STEVAL-C34KAT2 and STEVAL-C34DIL24 with STEVAL-MKI230KA add-ons for STEVAL-STWINBX1 through flex cable. Notice that hot plug is not supported. Add-ons must be attached to the STEVAL-STWINBX1 before powering it. See full remarks in chapter 2 and 3 from UM3021 and chapter 2 from UM2965.*

### 3.2.2 B-U585I-IOT02A, X-NUCLEO-IKS02A1 and X-NUCLEO-IKS4A1

In alternative, you can use:

- B-U585I-IOT02A, X-NUCLEO-IKS02A1 and X-NUCLEO-IKS4A1 with NUCLEO-U575ZI-Q or with NUCLEO-H7A3ZI-Q
- Laptop/PC with Windows 7, 8 or 10
- 2 USB cables

*Note: On these platforms, sensors are connected to STM32 via I2C. Due to I2C bandwidth limitations with respect to SPI, high ODR may result in corrupted data. See also Section 4: Datalog troubleshooting.*

## 3.3 Software setup

The following software components are required for the setup of a suitable development environment:

- FP-SNS-DATALOG2 firmware, available on www.st.com
- A standard user terminal as Putty or Tera Term, only for debug purposes (v. 4.97 or higher)
- STBLESensor app (only for STEVAL-STWINBX1, STEVAL-STWINKT1B and STEVAL-MKBOXPRO)
- To save, plot and elaborate data, Python utility scripts are available
- Development tool-chain and Compiler. The STM32Cube expansion software supports the three following environments to select from:
  - IAR Embedded Workbench for Arm® toolchain + ST-LINK
  - RealView Microcontroller Development Kit toolchain + ST-LINK
  - STM32CubeIDE + ST-LINK

# 4 Datalog troubleshooting

This section describes the most common issues that may arise while using FP-SNS-DATALOG2 and the possible workarounds:

- Cannot debug the code from an IDE:
  - As described in Section 2.1.4: How to handle double bank flash memory, STWIN.box and SensorTile.box PRO have two memory banks. Debug with the provided firmware only works while using flash bank 1, so the microcontroller configuration must match this setting. In order to reset the board configuration to the default flash bank you can follow the instructions in Section 2.1.4: How to handle double bank flash memory or run the reset script provided under the “Utilities/BoardReset” folder.
- Losing data during an acquisition:
  - Verify that you are using the latest available release of the Firmware
  - The amount of data generated by the DATALOG2 it is quite big: more than 6Mbps when all sensors are active on STWIN.box. To avoid losing data, especially during long acquisition sessions, enable only the needed sensors with the proper data rate. Avoid enabling all the sensors at the highest rate if it is not required, if needed, launch a debug session in DummyData mode to verify your setup before starting an acquisition campaign
  - See Section 2.2.7: SD card considerations for best suggestions on SD card
  - USB streaming depends on PC performances. If you are losing data check that the PC is not busy in other operations and avoid using USB hubs. Furthermore, check the standby options of the OS: normally, when a PC goes in standby or sleep, the USB port also goes in a low power state thus impacting on USB streaming performances
  - Consider enabling automode to avoid losing all the data in case of an issue (see chapter Section 2.2.6: Automatic Mode (Automode))
- Common issues with ST BLESensor app:
  - BLE app is stuck (command not working properly, demo page not updating, broken files, or acquisition folders): if the green led is not blinking anymore, you must disconnect the BLE app and perform an hardware reset of the board through the RESET button or by switching off and on the board
  - START button not enabled: check if the SD card is properly inserted. If not, insert it and refresh the demo page. If the issue persists, disconnect and reconnect to the BLE app
- Common issues using stdatalog_GUI
  - No device available (error message: Empty device list…):
    - Check if the board is properly connected to the PC via USB
    - Check if the FW is correctly loaded on the board. Remember that DATALOG2 is not the default firmware. See update instructions in Section 2.1: How to program STWIN, STWIN.box and SensorTile.box PRO
    - Refresh the application through the refresh button in the middle of the GUI
    - Update the STDATALOG-PYSDK to the latest version available
- Broken sensors: reset the board before starting a new acquisition session. If the issue persists, power down the board so that all the sensors perform an hard reset.

# Revision history

Table 1. **Document revision history**

| Date | Revision | Changes |
|---|---|---|
| 24-Jan-2023 | 1 | Initial release. |
| 10-Mar-2023 | 2 | Updated *Figure 22. Device configuration examples*, *Section 2.5.1 device_config.json* and *Section 2.5.4 Raw data files (.dat)*.<br>Added *Section 2.5.5 Device Template Model.*<br>Minor text changes. |
| 05-Apr-2023 | 3 | Modified Title in cover page.<br>Updated *Figure 1. FP-SNS-DATALOG2 software architecture*<br>Added references to STEVAL-MKBOXPRO, LSM6DSV16X.<br>Updated *Figure 1. FP-SNS-DATALOG2 software architecture* and *Section 2.1 How to program STWIN.box and Sensortile.box PRO.*<br>Added *Section 2.1.4 How to program the SensorTile.box PRO with an external debugger,*<br>Section 2.1.5 *How to handle double bank flash memory* and *Section 3.1.2 STEVAL-MKBOXPRO evaluation kit.*<br>Updated *Section 2.4 HSDPython_SDK*<br>Minor text changes. |
| 19-Jul-2023 | 4 | Updated introduction in cover page.<br>Updated *Section 1.2 Architecture, Section 1.3 Folder structure, Section 2.2.6 Automode and Section 2.5.6 ODR and FS mapping table.*<br>Added *Section 2.4.1 How to install Python, Section 2.4.2 Hot to install HSDPython_SDK, Section 2.5.6 ODR and FS mapping table , Section 3.1.4 B-U585I-IOT02A Discovery kit for IoT node, Section 3.1.5 X-NUCLEO-IKS02A1 motion and microphone expansion board, Section 3.1.6 NUCLEO-U575ZI-Q nucleo board, Section 3.1.7 NUCLEO-H7A3ZI-Q nucleo board* and *Section 4 Datalog troubleshooting.* |
| 06-Oct-2023 | 5 | Updated *Section Introduction, Section 1.2 Architecture, Section 1.3 Folder structure, Section 2.3.5 Standalone, Section 2.6.3 ucf configuration file: MLC and ISPU, Section 3.2.1 STEVAL-STWINBX1 , STEVAL-STWINKT1B and STEVAL-MKBOXPRO and Section 3.3 Software setup.*<br>Added *Section 2.2.1 How to program the STWIN with the STLINK-V3MINI, Section 2.2.2 How to program STWIN without STLINK-V3MINI using STM32CubeProgrammer "USB mode"* and *Section 3.1.2 STEVAL-STWINKT1B evaluation kit.* |

| Date | Revision | Changes |
|---|---|---|
| 06-Mar-2024 | 6 | Updated *Section Introduction, Figure 1. FP-SNS-DATALOG2 software architecture, Section 1.3 Folder structure, Section 2.3.3 Real Time Plot: hsdatalog_GUI, Section 2.3.5 Standalone, Section 2.6.2 Hot to install HSDPython_SDK, and Section 3.2.1 STEVAL-STWINBX1 , STEVALSTWINKT1B and STEVAL-MKBOXPRO.*<br>Added *Section 2.4 FTP Data Retrieving over Wi-Fi.*<br>Minor text changes. |
| 08-Jul-2024 | 7 | Updated Section Introduction, Section 2.2.8: Data labelling, Section 3: System setup guide.<br>Removed Section 2.1.4 How to program the SensorTile.box PRO with an external debugger and Section 2.2 How to program STWIN. |
| 15-May-2025 | 8 | Updated Section Introduction, Section 1.1: Overview, Section 1.2: Architecture, Section 1.3: Folder structure, Section 2.2.2: Command Line Interface, Section 2.2.3: Real Time Plot: stdatalog_GUI, Section 2.2.6: Automatic Mode (Automode), Section 2.2.7: SD card considerations, Section 2.2.8: Data labelling, Section 2.5: From HSDPython_SDK to STDATALOG-PYSDK , Section 2.6.1: device_config.json, Section 2.6.5: Device Template Model, Section 3.2.2: B-U585I-IOT02A, X-NUCLEO-IKS02A1 and X-NUCLEO-IKS4A1 and Section 4: Datalog troubleshooting. Removed 2.5.1 How to install Python and 2.5.2 How to install HSDPython_SDK. Inserted Section 3.1.7: X-NUCLEO-IKS4A1 Motion MEMS and environmental sensor expansion board for STM32 Nucleo. |

# Contents

# List of tables

# List of figures