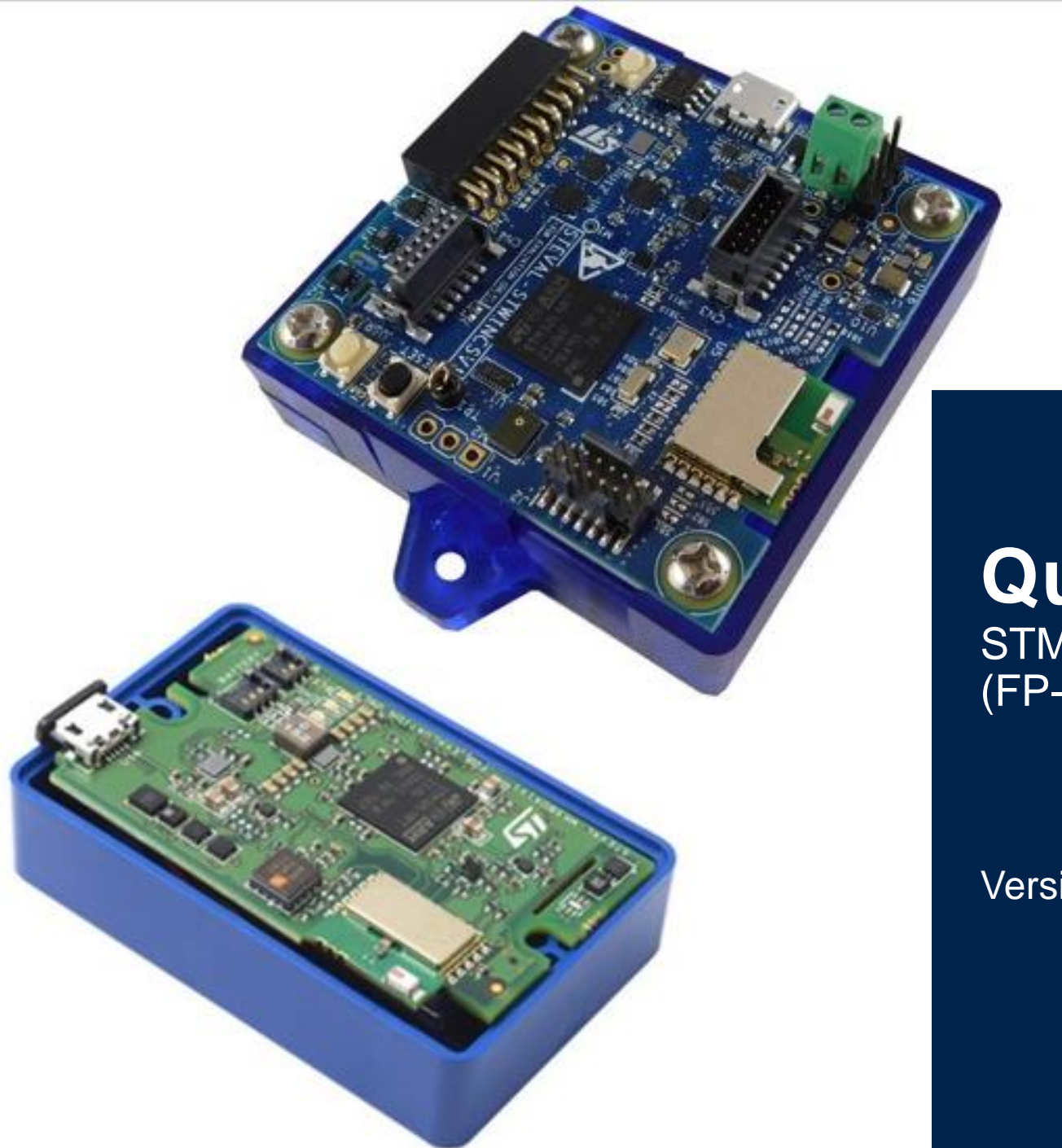




life.augmented



Quick Start Guide

STM32Cube High Speed Datalog function pack
(FP-SNS-DATALOG1)

Version 1.2 (June, 2023)

Agenda

1 Hardware and Software overview

2 Setup & Demo Examples

3 Documents & Related Resources

1- Hardware and Software overview

STEVAL-MKSBOX1V1 evaluation board

Hardware Overview

Multi sensor kit with portable sensor box and smart sensor app Hardware Description

The STEVAL-MKSBOX1V1 (SensorTile.box) is a ready-to-use box kit with wireless IoT and wearable sensor platform to help you use and develop apps based on remote motion and environmental sensor data, regardless of your level of expertise.

The SensorTile.box board fits into a small plastic shroud with a long-life rechargeable battery, and the ST BLE Sensor app on your smartphone connects via Bluetooth to the board and allows you to immediately begin using the wide range of default IoT and wearable sensor applications.

SensorTile.box includes a firmware programming and debugging interface that allows professional developers to engage in more complex firmware code development using the STM32 Open Development Environment (STM32 ODE), which includes a sensing AI function pack with neural network libraries.

Key Product on board

- Ultra-low-power STM32L4 Series MCUs based on ARM® Cortex® -M4 MCU 120 MHz with 2048 kbytes Flash (STM32L4R9ZI)
- Accurate temperature : STTS751
- Low power precise 6x IMU: LSM6DSOX
- Stand-alone XLs: LIS3DHH, LIS2DW12
- Magnetometer: LIS2MDL
- Altimeter / pressure sensor : LPS22HH
- Wide-band microphone: MP23ABS1
- Humidity sensor: HTS221



Latest info available at
www.st.com/sensortilebox

STWIN development kit - STEVAL-STWINKT1B

Hardware Overview

STWIN - SensorTile Wireless Industrial Node

The STWIN (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 supports BLE wireless connectivity through an on-board module and Wi-Fi connectivity through a special plugin expansion board (STEVAL-STWINWV1), wired RS485 and USB OTG connectivity.

Key Features

- Multi-sensing wireless platform implementing vibration monitoring and ultrasound detection
- Updated version of STEVAL-STWINKT1, now including STSAFE-A110 populated, BlueNRG-M2S module and IMP23ABSU MEMS microphone
- Ultra-low-power ARM Cortex-M4 MCU at 120 MHz with FPU, 2048 kbytes Flash memory (STM32L4R9)
- Micro SD Card slot for standalone data logging applications
- Option to implement Authentication and Brand protection secure solution with STSAFE-A110
- Wide range of industrial IoT sensors: ultra-wide bandwidth (up to 6 kHz), low-noise, 3-axis digital vibration sensor (IIS3DWB), 3D accelerometer + 3D Gyro iNEMO inertial measurement unit (ISM330DHCX) with machine learning core, ultra-low-power high performance MEMS motion sensor (IIS2DH), ultra-low-power 3-axis magnetometer (IIS2MDC), digital absolute pressure sensor (LPS22HH), relative humidity and temperature sensor (HTS221), low-voltage digital local temperature sensor (STTS751), industrial grade digital MEMS microphone (IMP34DT05), analog MEMS microphone with frequency response up to 80 kHz (IMP23ABSU)
- Modular architecture, expandable via on-board connectors: STMOD+ and 40-pin flex general purpose expansions, 12-pin male plug for connectivity expansions, 12-pin female plug for sensing expansions
- Other kit components: Li-Po battery 480 mAh, STLINK-V3MINI debugger with programming cable, Plastic box



Latest info available at
www.st.com/stwin

FP-SNS-DATALOG1

Software Overview

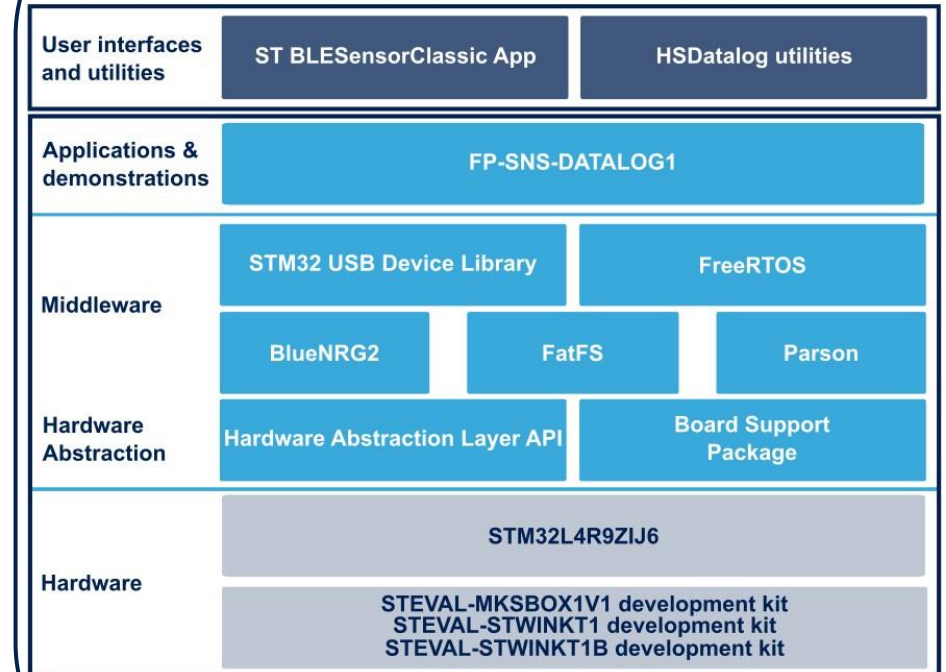
Software Description

The FP-SNS-DATALOG1 function pack implements High Speed Datalog application for STEVAL-MKSBOX1V1, STEVAL-STWINKT1 and STEVAL-STWINKT1B. It provides a comprehensive solution to save data from any combination of sensors and microphones configured up to the maximum sampling rate.

Key features

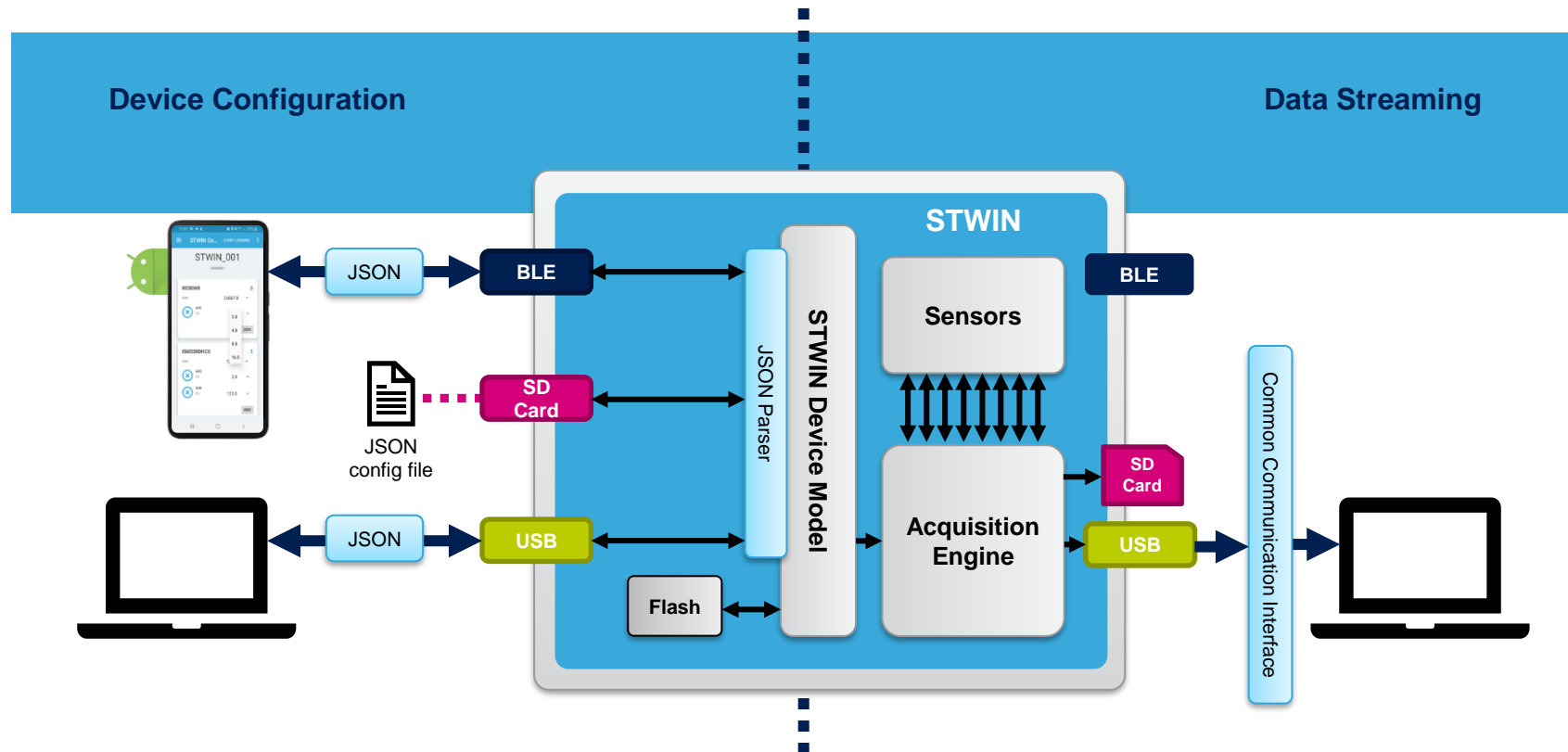
- High-rate (up to 6 Mbit/s) data capture software suite:
 - BLE app for system setup and real-time control
 - Python and C++ real-time control applications
 - Dedicated HSDPython_SDK for sensor data analysis, in common with FP-SNS-DATALOG2
 - Host developer's API enables integration into any data science design flow
 - Compatible with Unico-GUI which enables configuration of LSM6DSOX and ISM330DHCX Machine Learning Core unit
 - Timestamping for sensor data synchronization
- Embedded software, middleware and drivers:
 - FatFS third-party FAT file system module for small embedded systems
 - FreeRTOS third-party RTOS kernel for embedded devices
 - STWIN low-level BSP drivers
- Based on STM32Cube software development environment for STM32 microcontrollers

Overall Software Architecture



Intelligent Full Speed Data Logging

High Speed data log is part of FP-SNS-DATALOG1



- Optimized STM32 SW Supports streaming of all Sensors at Full data rate
 - USB Virtual Com Port and PC DLL
 - SD Card
 - Full control of acquisition via BLE app

2- Setup & Demo Examples

HW prerequisites for STEVAL-MKSBOX1V1

- 1x STEVAL-MKSBOX1V1 development board
- Laptop/PC with Windows 7, 8 or 10
- 1 x microUSB cables and 1 x USB type A to Mini-B USB cable
- ST-Link/V2 in-circuit debugger/programmer for STM8 and STM32
- 1 SD card
- 1 Android smartphone with ST BLESensorClassic App



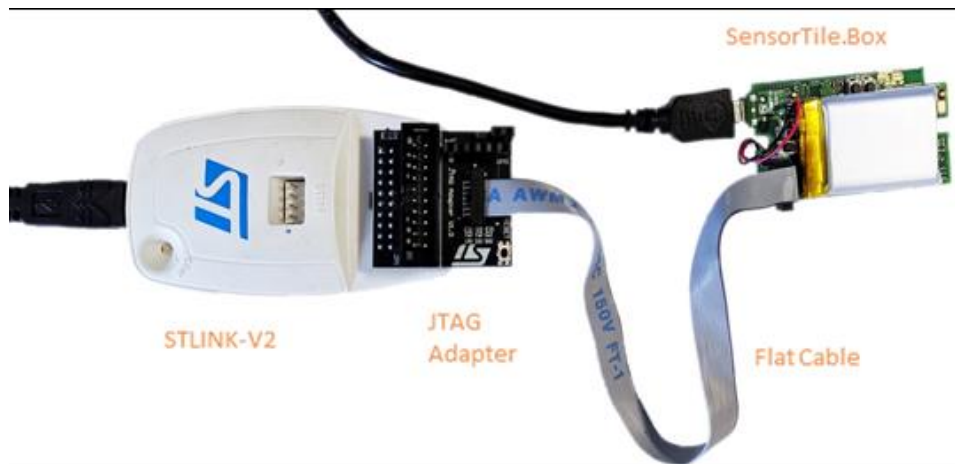
MicroUSB Cable



Mini USB



ST-Link/V2



STEVAL-MKSBOX1V1

HW prerequisites for STEVAL-STWINKT1B

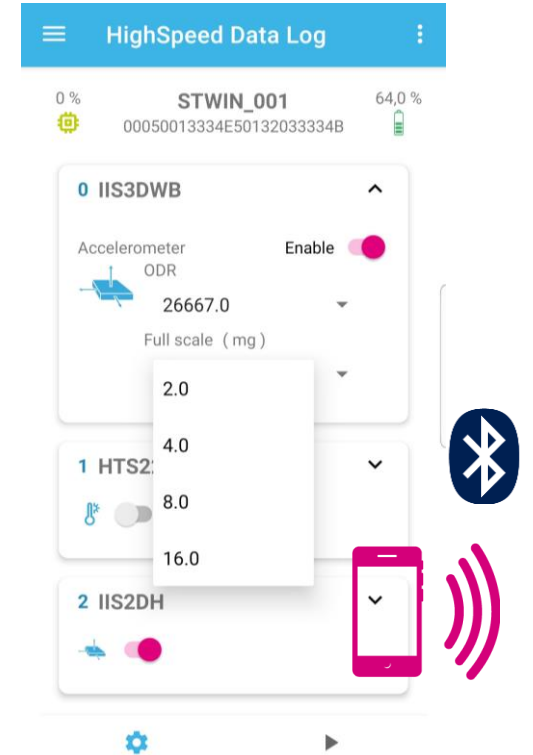
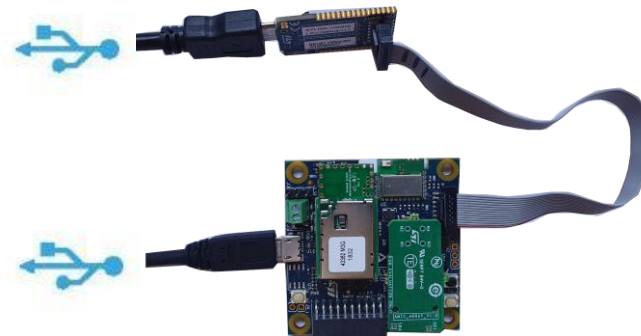
- 1x STEVAL-STWINKT1B development board
- Laptop/PC with Windows 7, 8 or 10
- 2x microUSB cables
- 1 SD card
- 1 Android smartphone with ST BLESensorClassic App



STEVAL-STWINKT1B



MicroUSB Cable



Software and other prerequisites

- **STM32CubeProgrammer Software**

- Download and install [STM32CubeProgrammer](#)

- **HSDatalog**

- Download the [FP-SNS-DATALOG1](#) package from www.st.com, copy the .zip file contents into a folder on your PC. The package contains binaries and source code with project files ([Keil](#), [IAR](#), [STM32CubeIDE](#))

- **ST BLESensorClassic App**

- Download and install ST BLESensorClassic App (for both Android and iOS - v4.17 and above)

- **Matlab or Python**

- To save and plot data, Matlab and Python utility scripts are available

HSDatalog **is not** the default firmware on STEVAL-MKSBOX1V1 nor STEVAL-STWINKT1.

To update the firmware, please follow the instructions available in [slide 13](#)

HSDatalog Sample applications



1 www.st.com/stwin

2

Select:
FP-SNS-DATALOG1

3

Download & unpack

Package structure

| Name |
|----------------------|
| └ _htmresc |
| └ Documentation |
| └ Drivers |
| └ Middlewares |
| └ Projects |
| └ Utilities |
| └ package.xml |
| └ Release_Notes.html |

Docs

BSP, HAL
drivers

Sample
applications

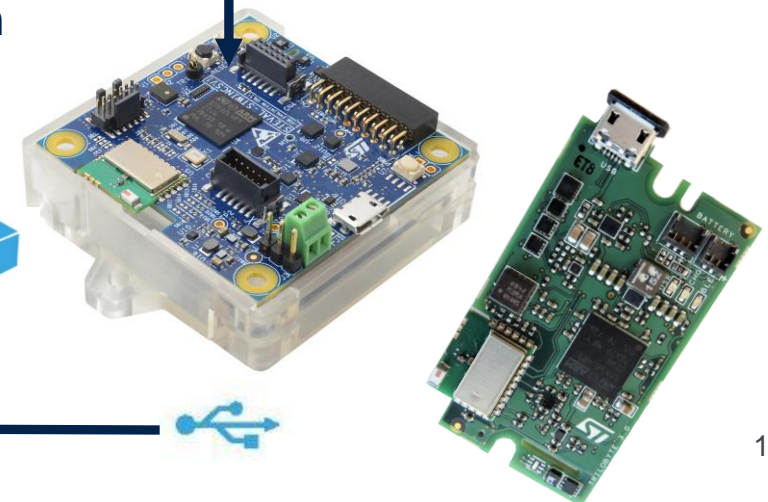
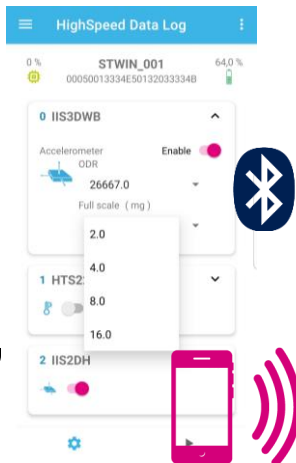
4

Use the pre-compiled binaries or re-compile the code
customizing your device configuration



5

6
Visualize log of sensors data and
control the device



HSDataLog Firmware Update

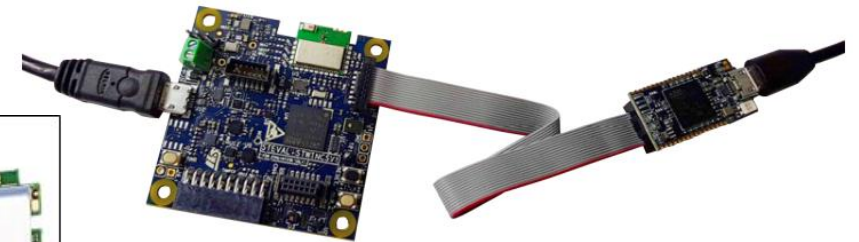
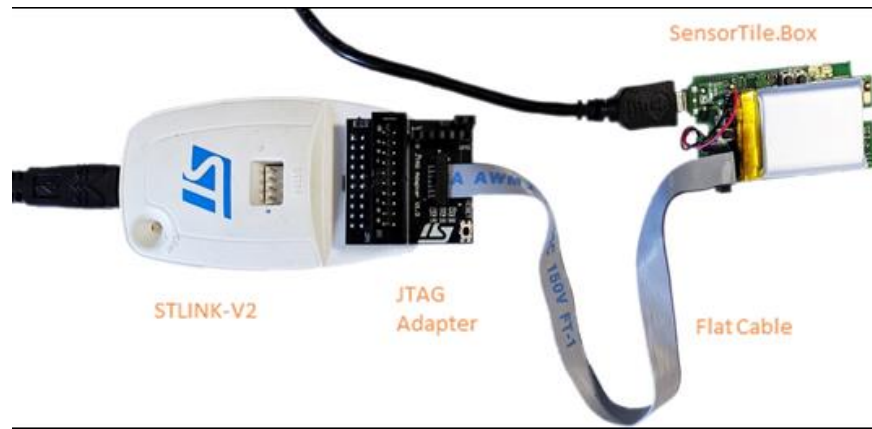
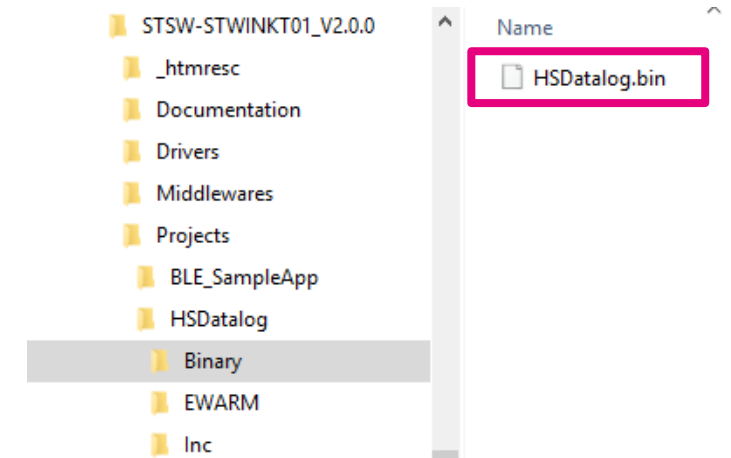
HSDataLog **is not** the default firmware on the STEVAL-MKSBOX1V1 nor on the STEVAL-STWINKT1, so it needs to be downloaded on the board by the user.

The easiest way is to use the **pre-compiled binary** provided in the package in the folder Projects/HSDataLog/Binary.

To update the firmware:

- Connect the board to the STLINK-V3MINI programmer.
- Connect both the boards to a PC using micro USB cables.
- Open [STM32CubeProgrammer](#), select the proper binary file and download the firmware.

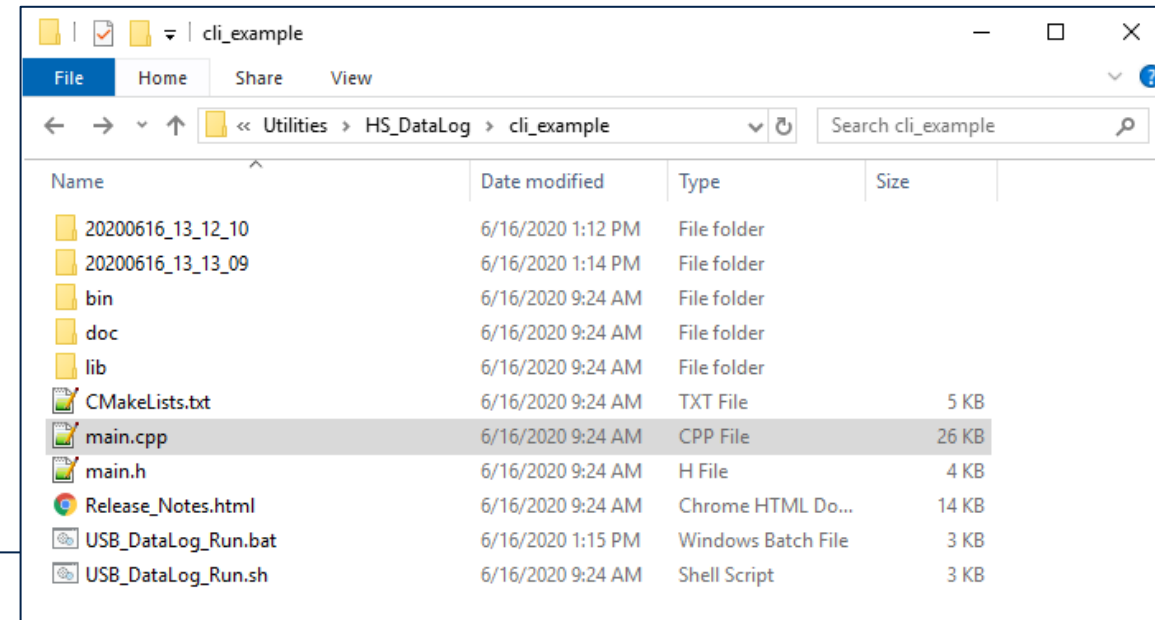
For further details, see [UM2622](#)



2.1- Command Line Interface example: USB sensor data streaming

cli_example overview

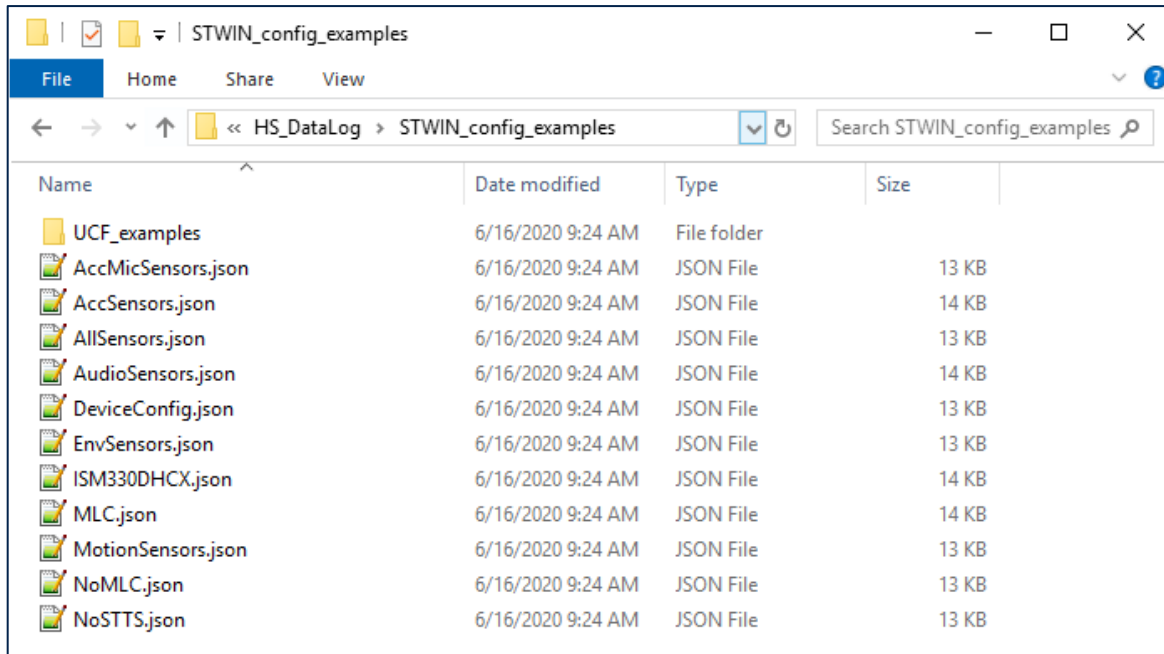
- The command line example is in the Utilities folder
- You can run the example after connecting the SensorTile.box or the STWIN to a PC via Micro-USB cable.
- USB_DataLog_Run.bat and USB_DataLog_Run.sh scripts provide a ready-to-use example.
- If needed, the application can receive a few parameters: timeout (**-t**), device configuration file (**-f**) and UCF file for the Machine Learning Core (**-u**).



```
4 REM Welcome to HS_DataLog Command Line Interface example
5 REM Usage: cli_example.exe [-COMMAND [ARGS]]
6 REM Commands:
7 REM -h Show this help
8 REM -f <filename>: Device Configuration file (JSON)
9 REM -t <seconds>: Duration of the current acquisition (seconds)
10
11
12 set PATH=%PATH%;.\bin\
13
14 cli_example.exe -u ..\STWIN_config_examples\UCF_examples\ism330dhcx_six_d_position.ucf -f ..\STWIN_config_examples\NoSTTS.json -t 100
15
16 pause
17
```


Modify device and sensors configuration

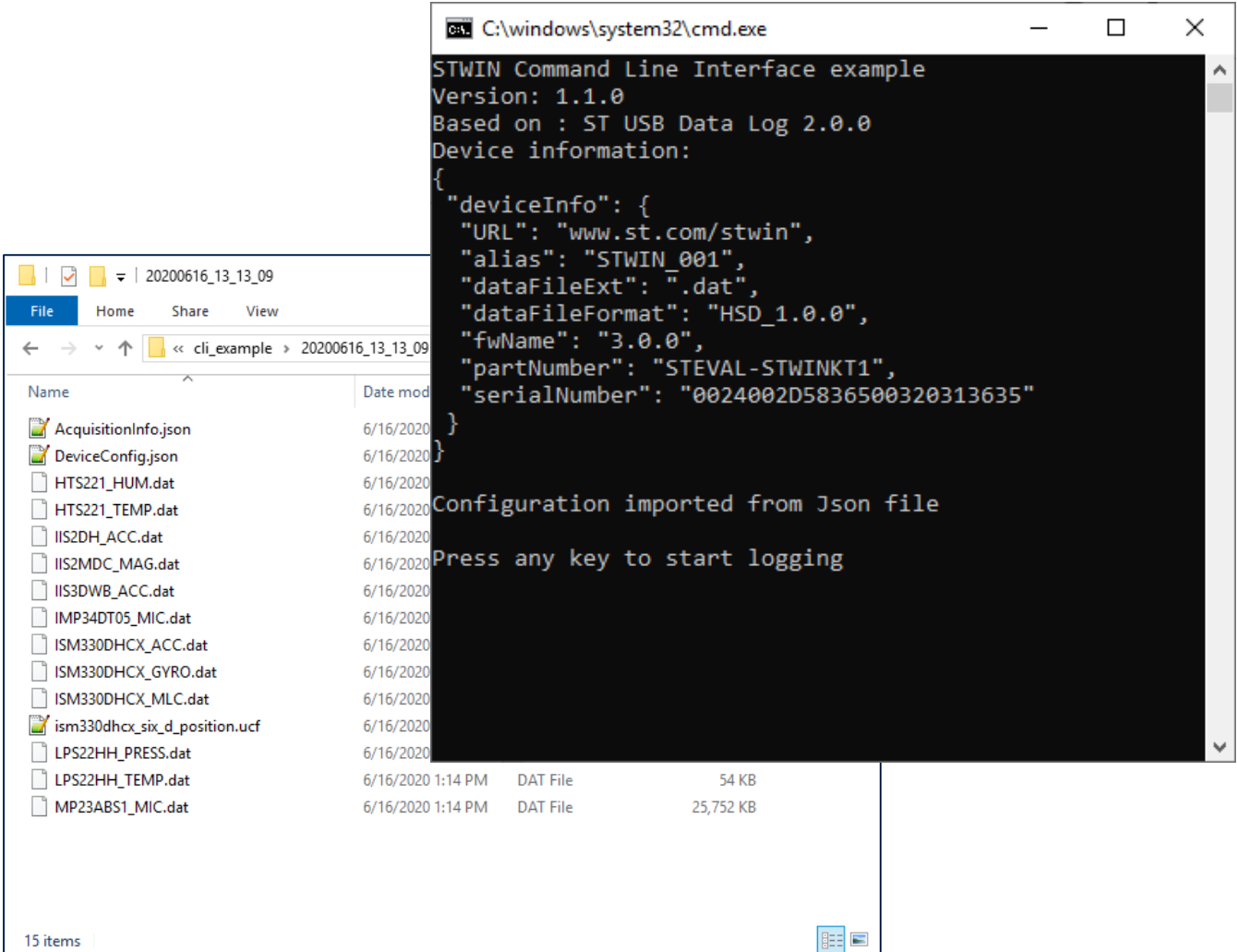
- To configure selected sensors, you can simply
 - Use one of the available device configuration examples in Utilities/STWIN_config_examples
 - Define your own sensors configuration by editing one of those examples (e.g. enable only a selected subset of sensors)



```
590 {
591   "id": 8,
592   "name": "STTS751",
593   "sensorDescriptor": {
594     "subSensorDescriptor": [
595       {
596         "FS": [
597           100
598         ],
599         "ODR": [
600           1,
601           2,
602           4
603         ],
604         "dataType": "float",
605         "dimensions": 1,
606         "dimensionsLabel": [
607           "tem"
608         ],
609         "id": 0,
610         "samplesPerTs": {
611           "dataType": "int16_t",
612           "max": 1000,
613           "min": 0
614         },
615         "sensorType": "TEMP",
616         "unit": "Celsius"
617       }
618     ],
619     "sensorStatus": {
620       "subSensorStatus": [
621         {
622           "FS": 100,
623           "ODR": 4,
624           "ODRMeasured": 0,
625           "comChannelNumber": -1,
626           "initialOffset": 0,
627           "isActive": true,
628           "samplesPerTs": 20,
629           "sdWriteBufferSize": 100,
630           "sensitivity": 1,
631           "usbDataPacketSize": 16,
632           "wifiDataPacketSize": 0
633         }
634       ]
635     }
636   }
637 }
```

Run the application

- Double click on the USB_DataLog_Run batch script
- The application starts, and the command line appears, showing information about the connected board.
- Press any key to start the datalogging
- Application will stop automatically if a timeout was set
- In any case, you can stop the data acquisition by pressing the ESC button
- The application will create a YYYYMMDD_HH_MM_SS (i.e., 20200128_16_33_00) folder containing the raw data and the JSON configuration file.



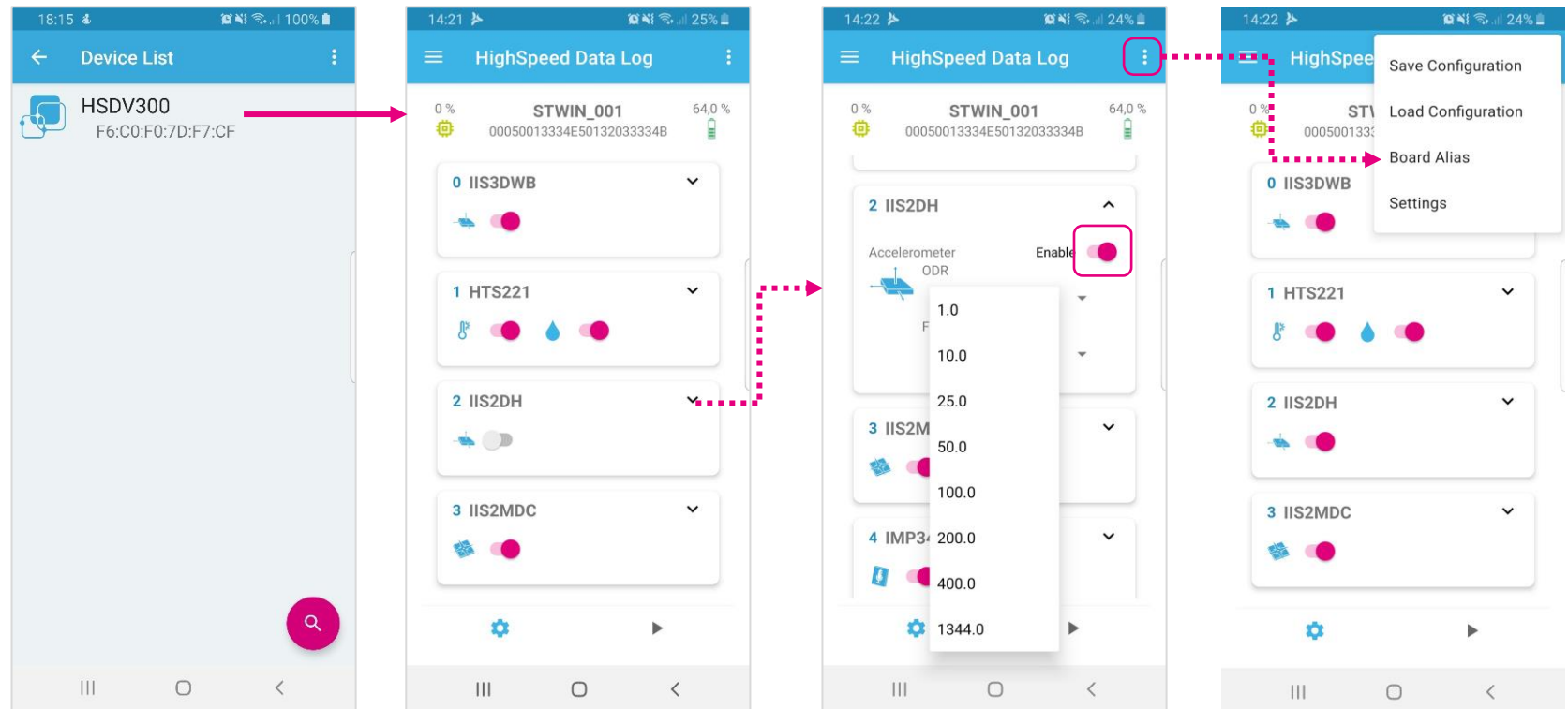
The screenshot displays two windows side-by-side. On the left is a Windows File Explorer window showing the contents of a folder named '20200616_13_13_09'. The folder contains 15 items, including JSON configuration files (AcquisitionInfo.json, DeviceConfig.json) and various data files (.dat and .ucf). On the right is a Windows Command Prompt window titled 'C:\windows\system32\cmd.exe'. It shows the output of the application, which includes the version (1.1.0), the ST USB Data Log version (2.0.0), and device information for a STEVAL-STWINKT1 board. The output also shows the configuration imported from a JSON file and a prompt to press any key to start logging.

```
C:\windows\system32\cmd.exe
STWIN Command Line Interface example
Version: 1.1.0
Based on : ST USB Data Log 2.0.0
Device information:
{
  "deviceInfo": {
    "URL": "www.st.com/stwin",
    "alias": "STWIN_001",
    "dataFileExt": ".dat",
    "dataFileFormat": "HSD_1.0.0",
    "fwName": "3.0.0",
    "partNumber": "STEVAL-STWINKT1",
    "serialNumber": "0024002D5836500320313635"
  }
}
Configuration imported from Json file
Press any key to start logging
```

2.2- HSDatalog and BLESensorClassic App: save data on SD card

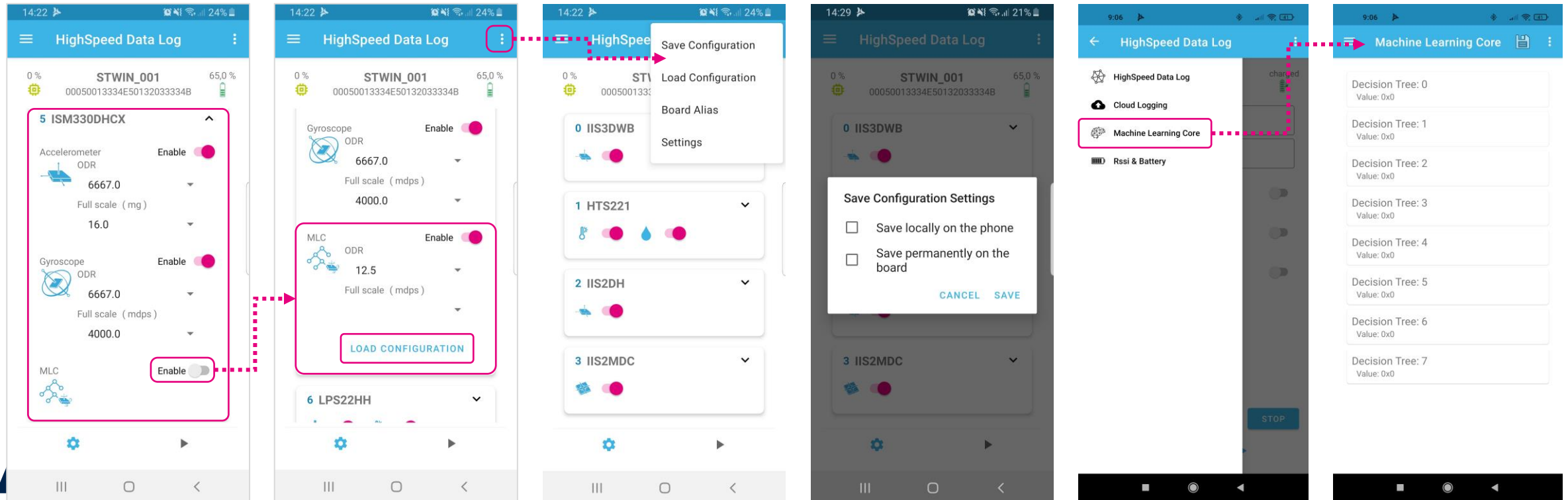
ST BLESensorClassic App: HSDatalog tab

- HSDatalog application can be controlled via Bluetooth using the [ST BLESensorClassic](#) app (available for Android and iOS) which lets you manage the board and sensor configurations, start/stop data acquisition on SD card and control data labelling.
- Once connected, you can configure the device by:
 - enabling/disabling a specific sensor
 - changing sensor parameters
 - updating the “Board Alias”



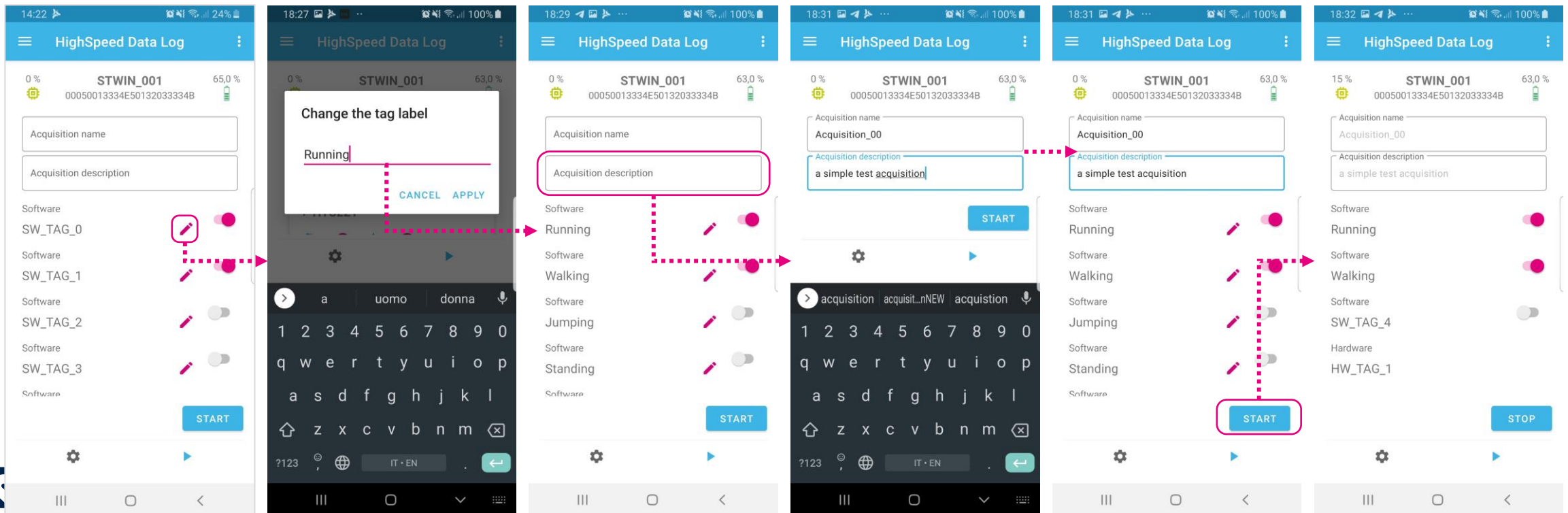
Configure MLC, save/load configuration

- You can also:
 - send a UCF configuration file to setup the [LSM6DSOX](#) (available on the SensorTile.box) or the [ISM330DHCX](#) (available on the STWIN) Machine Learning Core and visualize its outputs. The UCF file could be retrieved either from the smartphone memory or from a cloud storage (e.g. Google Drive, Microsoft OneDrive, etc.)
 - save the current device configuration on the smartphone (JSON file)
 - overwrite the default device configuration so that the new one is loaded automatically at power-on (an SD card is needed to use this feature)
 - load a specific device configuration (JSON file) from the smartphone



Acquisition settings and control

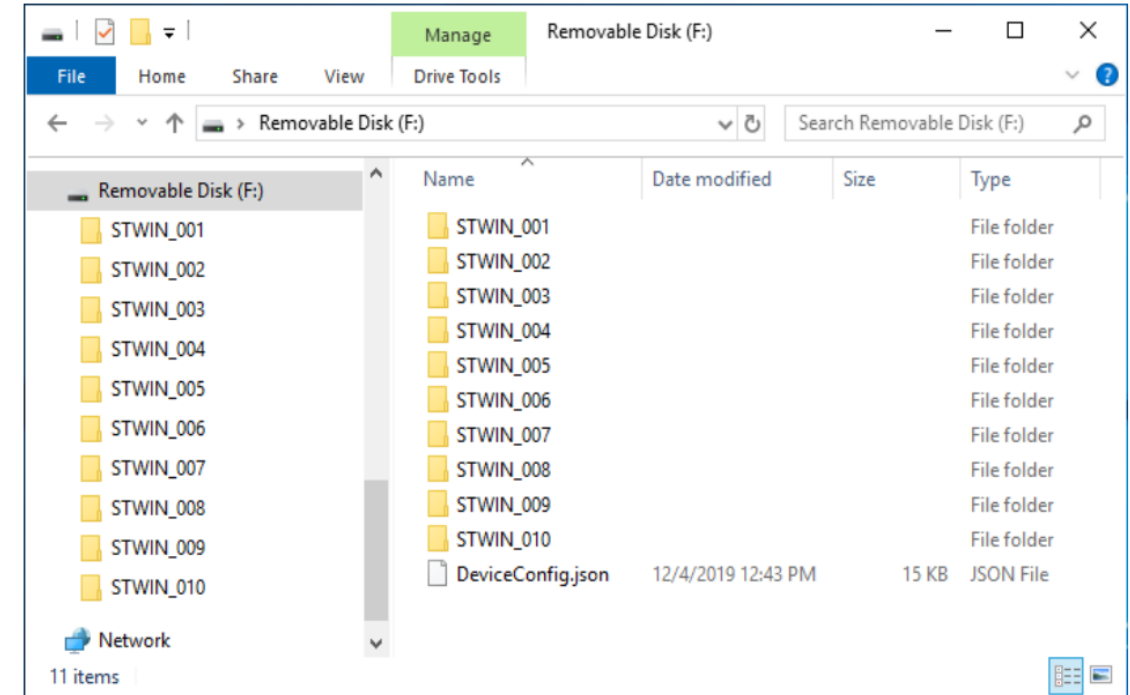
- By clicking to the play button ► you can switch to the acquisition settings and control tab to:
 - start and stop an acquisition (to an SD card)
 - choose which tag classes will be used for the next acquisition (both HW and SW tags)
 - handle hardware and software data tagging and labelling of an ongoing acquisition
 - set up the acquisition name and description



2.3- HSDatalog standalone: save data on SD card

Start an acquisition in standalone mode

- HSDatalog can work also standalone, saving all the sensor data at the highest possible rate into the SD card
- HSDatalog can also read a custom sensors 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 (same as described in 2.1).
- Once the firmware is already flashed on the board:
 - Insert the SD card
 - If the board is battery-powered and switched off, press PWR button to switch on the board
 - Press the RESET button
 - If the SD card is not inserted properly, the orange LED will blink very fast. Unless the orange LED will blink very slow.
 - If a JSON configuration file is present in the root folder of the SD card, the custom sensor configuration is loaded from the file itself
 - Press the USR button to start saving data. You will see the green LED blinking.
 - To stop the data acquisition, press again the USR button



Automode

- DATALOG1 also features the automode, which can be initiated automatically at the device power-up or reset
- This mode can be used to start the datalog operations or to pause all the executions for a specific period by putting the sensor node in the "idle" phase.
- Automode allows automatically saving data on the SD card, generating different acquisitions folders. It can be useful to automate long acquisition setups, avoid too big datasets and reduce SD card errors by avoiding data loss through autosaving.

```
{  
  "info": {  
    "version": "1",  
    "auto_mode": true,  
    "phases_iteration": 0,  
    "start_delay_ms": 3000,  
    "execution_plan": [  
      "datalog",  
      "idle"  
    ],  
  
    "datalog": {  
      "timer_ms": 7000  
    },  
    "idle": {  
      "timer_ms": 3000  
    }  
  }  
}
```

Automode

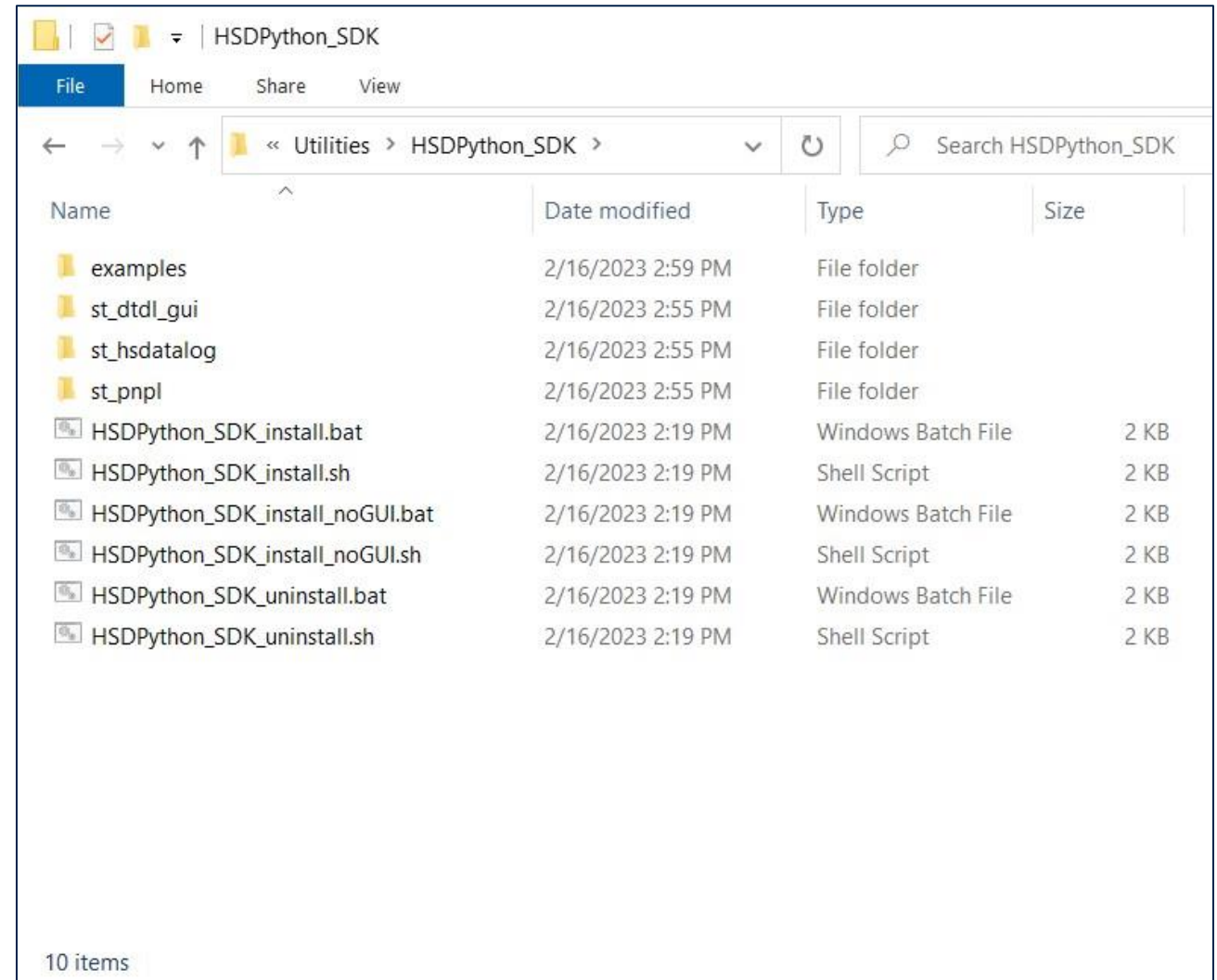
- As for the standalone mode, to enable the automode you must setup properly the `execution_config.json`
 - **auto_mode**: if true, auto-mode will start after reset and node initialization
 - **execution_plan**: is a sequence of maximum ten execution steps
 - **start_delay_ms**: indicates the initial delay in milliseconds applied after reset and before the first execution phase starts when auto-mode is selected
 - **phases_iteration**: gives the number of times the execution_plan is executed; zero indicates an infinite loop
 - phase step execution context settings:
 - `datalog`
 - `timer_ms`: specifies the duration in ms of the execution phase; zero indicates an infinite time
 - `idle`
 - `timer_ms`: specifies the duration in ms of the execution phase; zero indicates an infinite time
- Then place it in the root folder of the SD card
- If the board is battery-powered and switched off, press PWR button to switch on the board. Press the RESET button

```
{
  "info": {
    "version": "1",
    "auto_mode": true,
    "phases_iteration": 0,
    "start_delay_ms": 3000,
    "execution_plan": [
      "datalog",
      "idle"
    ],
    "datalog": {
      "timer_ms": 7000
    },
    "idle": {
      "timer_ms": 3000
    }
  }
}
```

2.4 – HSDPython_SDK

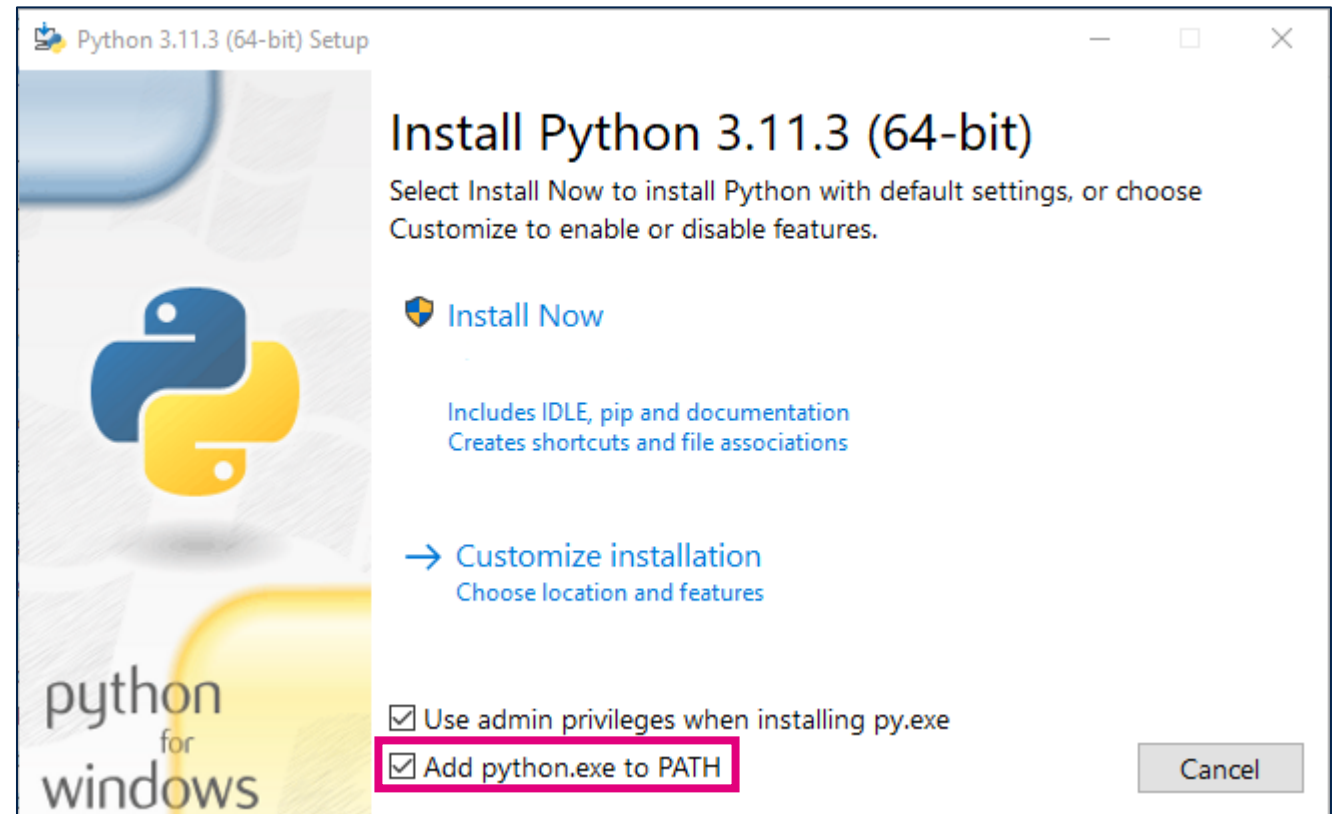
HSDPython_SDK

- Starting from v1.5.0, FP-SNS-DATALOG1 provides a dedicated Python SDK, ready-to-use for integration into any data science design flow.
- HSDPython_SDK is provided also in FP-SNS-DATALOG2 and can handle data acquired from both packages
- HSDPython_SDK has been developed in Python 3.10
- The SDK contains many Python scripts, examples and Jupiter notebook that can be used to log and elaborate data and shows up the main capabilities available into the st_dtdl_gui, st_hsdatalog and st_pnpl Python modules.



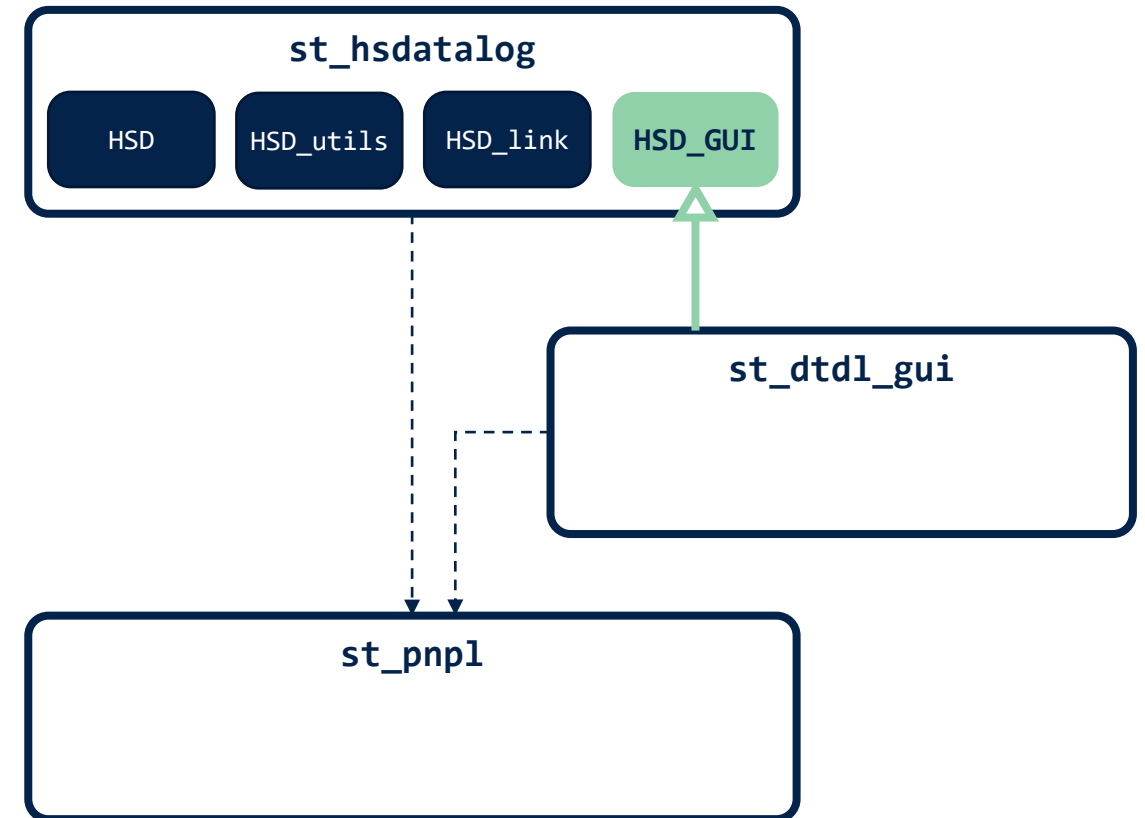
HSDPython_SDK

- Before using HSDPython_SDK, Python (from version 3.10 on) must be properly installed on your machine.
- The following steps are valid for a Windows machine. Similar approach can be used also on other OS.
 - Download the installer from python.org and launch it
 - Select **Add python.exe flag** and click **Install Now**. Administrator privileges are needed.
 - Once the setup is complete, you can use Python on your machine



HSDPython_SDK

- To fully exploit HSDPython_SDK capabilities and solve inter-dependencies, installer scripts are provided.
- By simply launching the needed *install.bat* (in Windows environment) or *.sh* (in Linux environment), the SDK is properly installed.
 - For Linux user, further steps are needed. Step-by-step procedure is described in detail in the *readme_linux* file.
- In this way, all the required dependencies are automatically solved. No other Python modules must be installed manually.
- The three modules are also distributed as Python wheels, that expert users are free to install manually in their environment by pip or apt-get command



HSDPython_SDK scripts

- User is free to develop its own projects by including modules or by modifying the already available scripts
- Here the list of the scripts available in examples folder:
 - *hsdatalog_check_dummy_data.py* can be used to debug the complete application and verify that data are stored or streamed correctly. You must recompile the firmware enabling *HSD_USE_DUMMY_DATA* define (set *#define HSD_USE_DUMMY_DATA 1*).
 - *hsdatalog_cli.py* is the Python version of the CLI described in Section 2.1
 - *hsdatalog_data_export.py* can convert data into CSV or TSV files.
 - *hsdatalog_data_export_by_tags.py* can be used for tagged acquisition to convert data into different files, one for each tag used.
 - *hsdatalog_dataframes.py* can save data as pandas dataframe for further processing needs.
 - *hsdatalog_plot.py* can plot the desired data.
 - *hsdatalog_to_nanoedge.py* can prepare data to be imported into NanoEdge AI Studio solution.
 - *hsdatalog_to_unico.py* can prepare data to be imported into Unico-GUI.
 - *hsdatalog_to_wav.py* can convert audio data into a wave file.

HSDPython_SDK scripts

- You can execute them in your preferred Python environment (i.e.: use the command *python hsdatalog_plot.py*)
- Each scripts can accept optional parameters. You can see them by executing the example with the *-h* option (i.e.: *python hsdatalog_ploy.py -h*)

```
C:\Windows\System32\cmd.exe
C:\git\ODE\FP\DATALOG2\Firmware\Utilities\HSDPython_SDK>python hsdatalog_plot.py -h
Usage: hsdatalog_plot.py [OPTIONS] ACQ_FOLDER

Options:
  -s, --sensor_name TEXT      Sensor Name - use "all" to plot all active
                              sensors data, otherwise select a specific
                              sensor by name
  -st, --sample_start INTEGER Sample Start - Data plot will start from this
                              sample
  -et, --sample_end INTEGER   Sample End - Data plot will end up in this
                              sample
  -r, --raw_data              Uses Raw data (not multiplied by sensitivity)
  -l, --labeled               Plot data including information about
                              annotations taken during acquisition (if any)
  -p, --subplots              Multiple subplot for multi-dimensional sensors
  -d, --debug                 [DEBUG] Check for corrupted data and timestamps
  -h, --help                  Show this message and exit.
  --help                      Show this message and exit.

-> Script execution examples:

-> HSDatalog1:
python hsdatalog_plot.py ..\STWIN_acquisition_examples\STWIN_00001
python hsdatalog_plot.py ..\STWIN_acquisition_examples\STWIN_00001 -s all
python hsdatalog_plot.py ..\STWIN_acquisition_examples\STWIN_00002 -s all -l
python hsdatalog_plot.py ..\STWIN_acquisition_examples\STWIN_00002 -l -p -r

-> HSDatalog2:
python hsdatalog_plot.py ..\STWIN.box_acquisition_examples\20221017_13_18_08
python hsdatalog_plot.py ..\STWIN.box_acquisition_examples\20221017_13_18_08 -s all
python hsdatalog_plot.py ..\STWIN.box_acquisition_examples\20221017_13_18_08 -s all -l
python hsdatalog_plot.py ..\STWIN.box_acquisition_examples\20221017_13_18_08 -l -p -r

C:\git\ODE\FP\DATALOG2\Firmware\Utilities\HSDPython_SDK>
```

3- Documents & Related Resources

Documents & Related Resources

FP-SNS-DATALOG1:

- **DB4322:** STM32Cube High Speed Datalog function pack – [databrief](#)
- **UM2688:** Getting started with the STM32Cube High Speed Datalog function pack – [user manual](#)

STEVAL-MKSBOX1V1:

- [Gerber files, BOM, Schematic](#)
- **DB3903:** SensorTile.box wireless multi sensor development kit with user friendly app for IoT and wearable sensor applications – [databrief](#)
- **UM2580:** How to use the wireless multi sensor development kit with customizable app for IoT and wearable sensor applications – [user manual](#)

STEVAL-STWINKT1B:

- [Gerber files, BOM, Schematic](#)
- **DB4345:** STWIN SensorTile Wireless Industrial Node development kit and reference design for industrial IoT applications – [databrief](#)
- **UM2777:** How to use the STEVAL-STWINKT1B SensorTile Wireless Industrial Node for condition monitoring and predictive maintenance applications – [user manual](#)

4- STM32 Open Development Environment: Overview

STM32 ODE Ecosystem

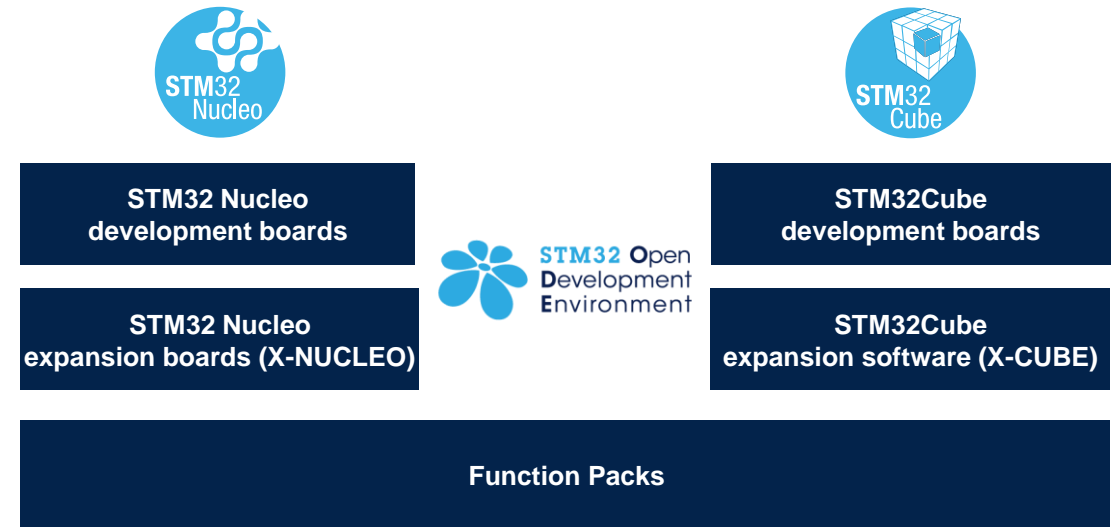
FAST, AFFORDABLE PROTOTYPING AND DEVELOPMENT

The STM32 Open Development Environment (ODE) is an **open, flexible, easy** and **affordable** way to develop innovative devices and applications based on the STM32 32-bit microcontroller family combined with other state-of-the-art ST components connected via expansion boards. It enables fast prototyping with leading-edge components that can quickly be transformed into final designs.

The STM32 ODE includes the following five elements:

- STM32 Nucleo development boards. A comprehensive range of affordable development boards for all STM32 microcontroller series, with unlimited unified expansion capability, and with integrated debugger/programmer
- STM32 Nucleo expansion boards. Boards with additional functionality to add sensing, control, connectivity, power, audio or other functions as needed. The expansion boards are plugged on top of the STM32 Nucleo development boards. More complex functionalities can be achieved by stacking additional expansion boards
- STM32Cube software. A set of free-of-charge tools and embedded software bricks to enable fast and easy development on the STM32, including a Hardware Abstraction Layer, middleware and the STM32CubeMX PC-based configurator and code generator
- STM32Cube expansion software. Expansion software provided free of charge for use with STM32 Nucleo expansion boards, and compatible with the STM32Cube software framework
- STM32Cube Function Packs. Set of function examples for some of the most common application cases built by leveraging the modularity and interoperability of STM32 Nucleo development boards and expansions, with STM32Cube software and expansions.

The STM32 Open Development Environment is compatible with a number of IDEs including IAR EWARM, Keil MDK, mbed and GCC-based environments.



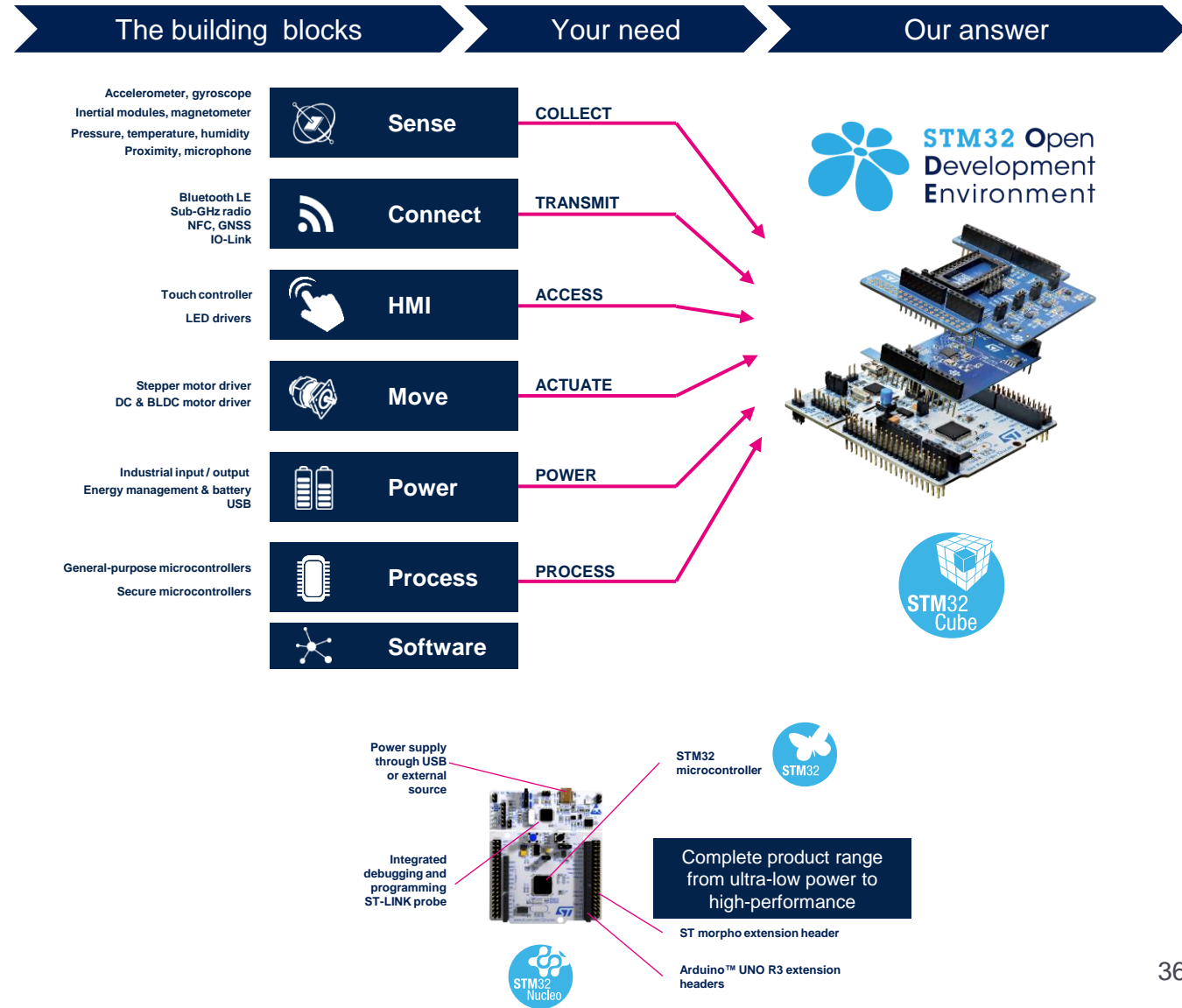
STM32 Open Development Environment: all that you need

The combination of a broad range of expandable boards based on leading-edge commercial products and modular software, from driver to application level, enables fast prototyping of ideas that can be smoothly transformed into final designs.

To start your design:

- Choose the appropriate STM32 Nucleo development board (MCU) and expansion (X-NUCLEO) boards (sensors, connectivity, audio, motor control etc.) for the functionality you need
- Select your development environment (IAR EWARM, Keil MDK, and GCC-based IDEs) and use the free STM32Cube tools and software.
- Download all the necessary software to run the functionality on the selected STM32 Nucleo expansion boards.
- Compile your design and upload it to the STM32 Nucleo development board.
- Then start developing and testing your application.

Software developed on the STM32 Open Development Environment prototyping hardware can be directly used in an advanced prototyping board or in an end product design using the same commercial ST components, or components from the same family as those found on the STM32 Nucleo boards.



Thank you