



life.augmented



# Quick Start Guide

## STDATALOG-PYSDK

Version 1.0 – Jan '25

# Agenda

1 Software overview

2 Installation guide

3 Demo Examples

4 Documents & Related Resources

# 1 – Software overview

# STDATALOG-PYSDK

## Software Overview

### Software Description

The STDATALOG-PYSDK (formerly known as HSDPython\_SDK, previously distributed in FP-SNS-DATALOG1, FP-SNS-DATALOG2, and FP-IND-DATALOGMC function packs) 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.

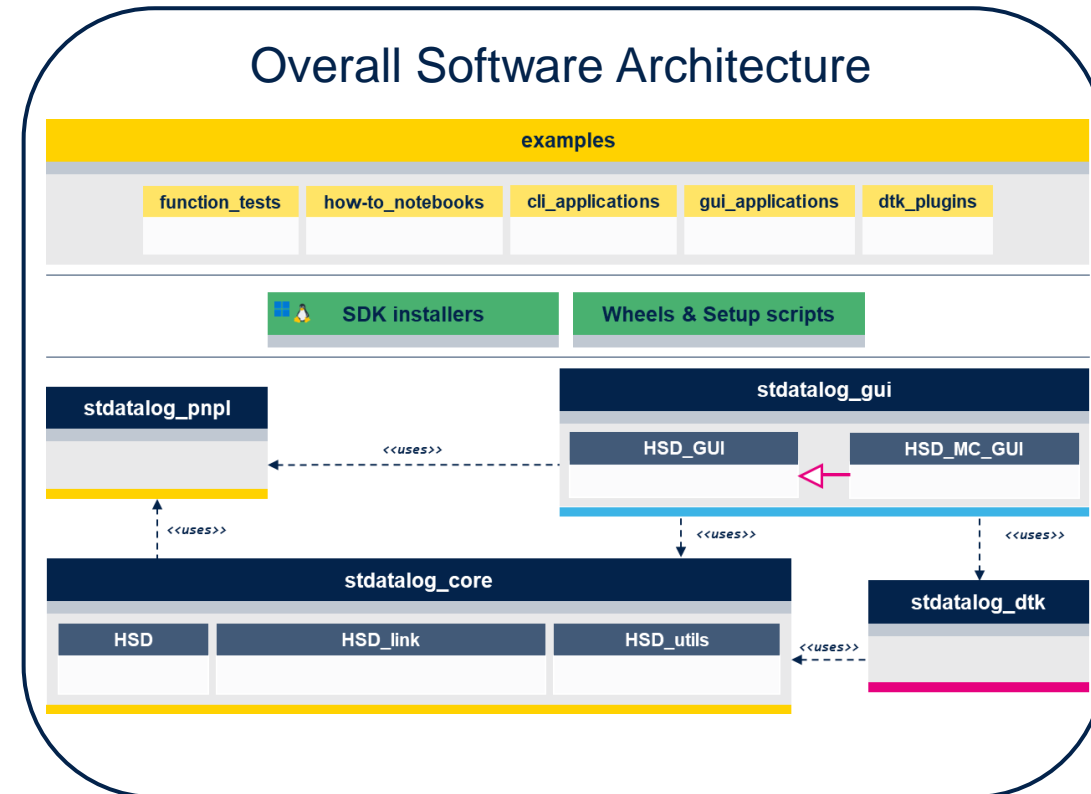
This software development kit is designed with an open and modular architecture, making it an excellent resource for data scientists and embedded designers.

It provides a range of tools and utilities designed to simplify the development of applications that use data from ST system solutions.

It includes Python scripts to create, elaborate, and organize data into structured datasets. These datasets are compatible with mainstream data science toolchains, promoting reusability across multiple projects. Additionally, the scripts can be easily integrated into any data science design workflow.

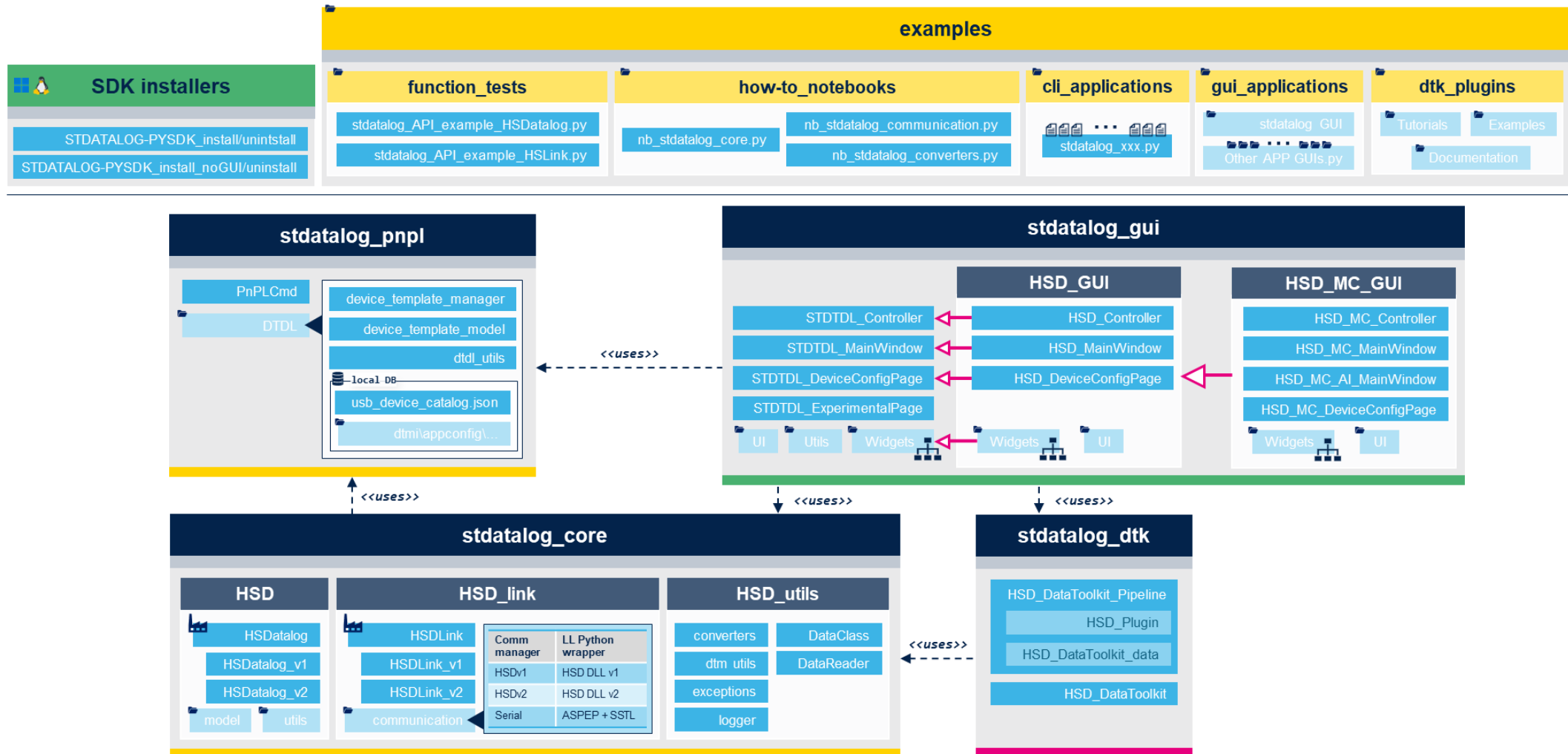
The STDATALOG-PYSDK is structured into four distinct Python packages, each serving a specialized purpose: the stdatalog\_core package, stdatalog\_dtk package, stdatalog\_gui package, and the stdatalog\_pnpl package.

It complements and is natively compatible with FP-SNS-DATALOG2, FP-IND-DATALOGMC, and FP-SNS-DATALOG1



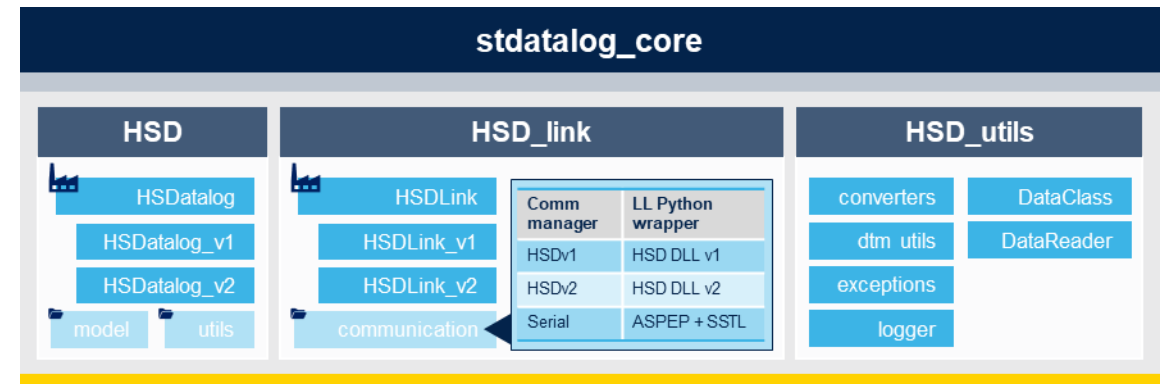
# STDATALOG-PYSDK

## Software architecture Overview



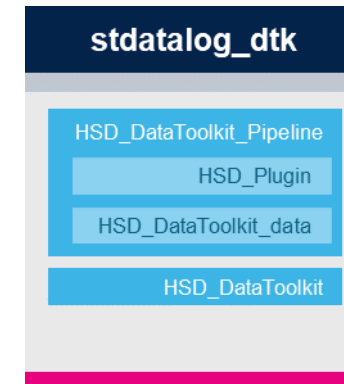
# stdatalog\_core

- The stdatalog\_core package is designed for high-speed data logging and communication with STMicroelectronics hardware devices.
- It provides a comprehensive set of tools for data acquisition, processing, conversion, and visualization.
- It manages USB communication to retrieve connected board information and data, set target properties, and control the data acquisition process.
- Additionally, it oversees error management and application log messages, ensuring smooth and reliable operation.
- This package is composed of three sub-packages (the acronym HSD stands for High-Speed Data Logger):
  - HSD: This package contains the core classes and functions to manage HSD acquisition folders.
  - HSD\_link: This package contains the classes and functions to manage the communication with a connected board.
  - HSD\_utils: This package contains utility functions to manage HSD data and folders.



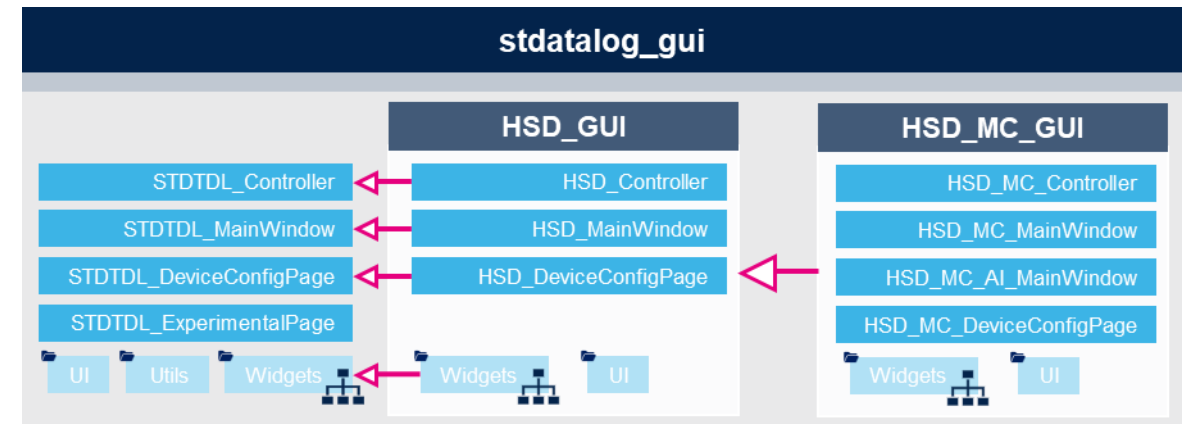
# stdatalog\_dtk

- The stdatalog\_dtk (the acronym stands for Data ToolKit) provides functionalities to realize a data processing pipeline that can handle data from various sources and process them in many different and customizable ways.
- A data processing pipeline is a series of data processing elements connected in series, where the output of one element is the input of the next. These processing elements are called Plugins.
- The package provides an abstract Plugin class that must be inherited to create a new plugin that can be added to the pipeline.
- It is designed to simplify the development of applications using data from ST sensors, providing complete hardware abstraction, making it easier to handle real-time data from connected ST system solutions or stored datasets.



# stdatalog\_gui

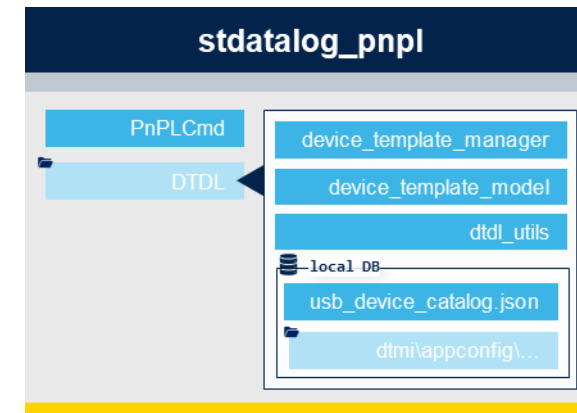
- The stdatalog\_gui package is a UI toolkit developed by STMicroelectronics, based on DTDLv2 (Digital Twin Definition Language) and PySide6.
- It provides a graphical user interface for high-speed data logging and communication with STMicroelectronics hardware devices.
- The package provides a set of graphical widgets useful to display live data streams, configure, and show connected device parameters and manage data collection. These widgets are the basic building blocks for creating interactive graphical user interfaces (GUIs) to manage datalogging applications and device configuration.
- In addition, the package offers two packages that specialize the base classes for specific use cases:
  - HSD\_GUI: a package that provides a set of widgets to create GUI applications for high-speed data logging. It includes widgets to display live data streams, configure and show connected device parameters, and manage data collection.
  - HSD\_MC\_GUI: a package that provides a set of widgets to create GUI applications for high-speed data logging in the context of motor control applications. It includes widgets to display motor control telemetries, and to configure and control connected motors parameters.





# stdatalog\_pnpl

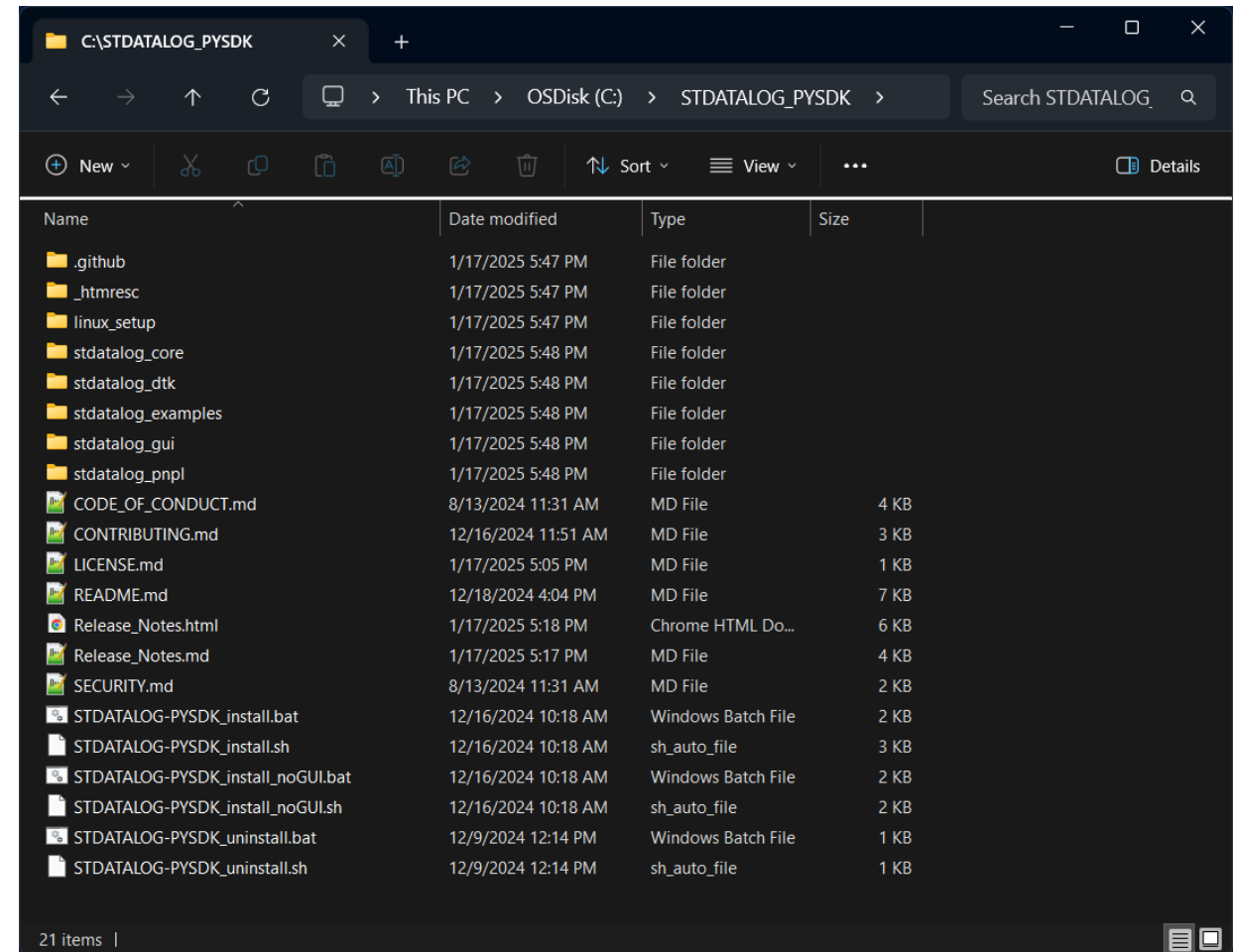
- The stdatalog\_pnpl package is used to manage device template models, which are high-level descriptors of the (board + firmware) system.
- Device Template Models are designed following the DTDLv2 standard, which is a JSON-based language that describes the capabilities of a system and its components.
- The package facilitates the creation and dynamic management of the commands-set that can be exchanged between target devices and the Python SDK.
- To achieve this, the package provides the PnPLCmd class, which provides a set of methods to create and manage various types of PnPL commands. The PnPL acronym stands for "Plug and Play Like" and is inspired by the Plug and Play standard from the Azure IoT ecosystem.
- This feature is particularly useful for developers who need to customize and integrate various devices into their projects, ensuring seamless communication between the devices and the SDK.



## 2 – Installation guide

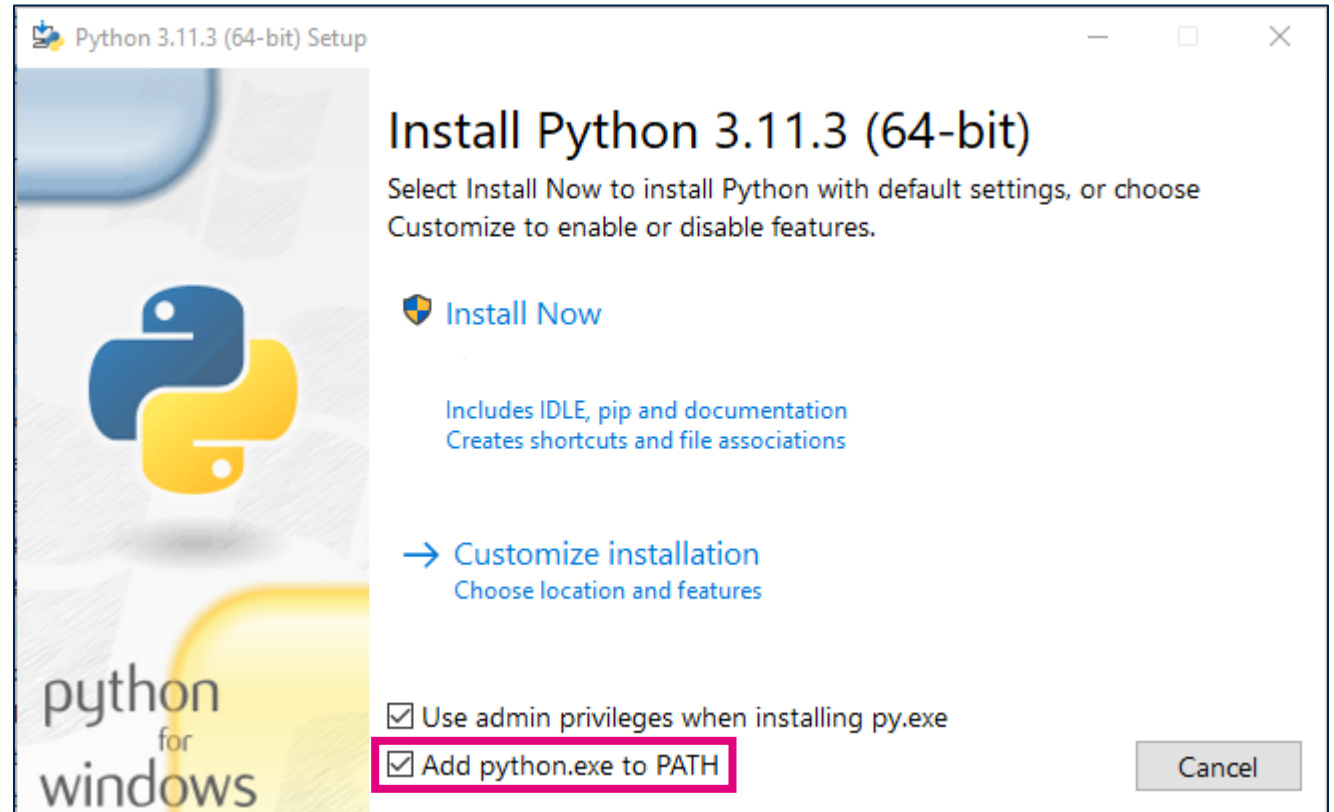
# STDATALOG-PYSDK

- The STDATALOG-PYSDK (formerly known as HSDPython\_SDK) was previously distributed in FP-SNS-DATALOG1, FP-SNS-DATALOG2, and FP-IND-DATALOGMC function packs.
- This python software development kit (SDK) for data logging is a comprehensive Python framework designed to facilitate the capture, processing, and visualization of data from a wide range of sources.
- STDATALOG-PYSDK has been developed in Python 3.12, but it is compatible also with Python 3.11 and 3.10
- The example scripts provided in the stdatalog\_examples folder take advantage for the API provided by the stdatalog\_core, stdatalog\_dtk, stdatalog\_gui, and the stdatalog\_pnpl Python modules.



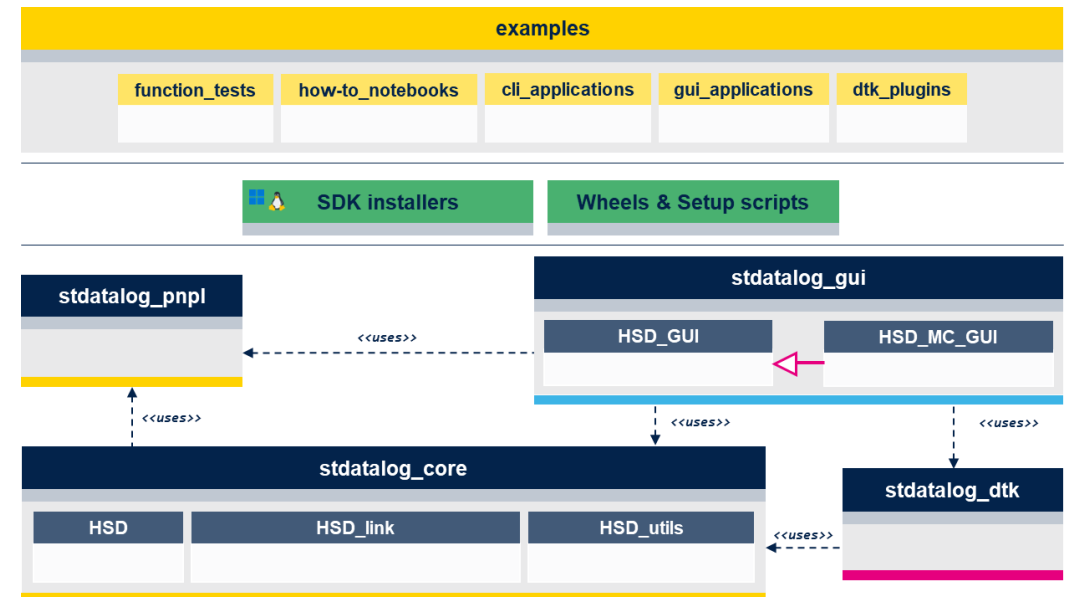
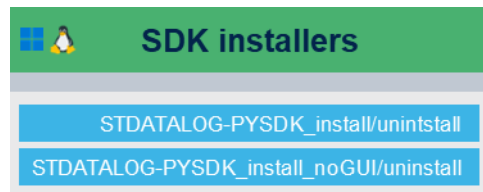
# STDATALOG-PYSDK

- Before using STDATALOG-PYSDK, Python 3.10, 3.11 or 3.12 must be properly installed on your machine.
- The following steps are valid for a Windows machine. Similar approach can be followed on other OS as well.
  - Download the installer from [python.org](https://python.org) and launch it
  - Select **Add python.exe to PATH flag** and click **Install Now**. Administrator privileges are needed.
  - Once the setup is complete, you can use Python on your machine



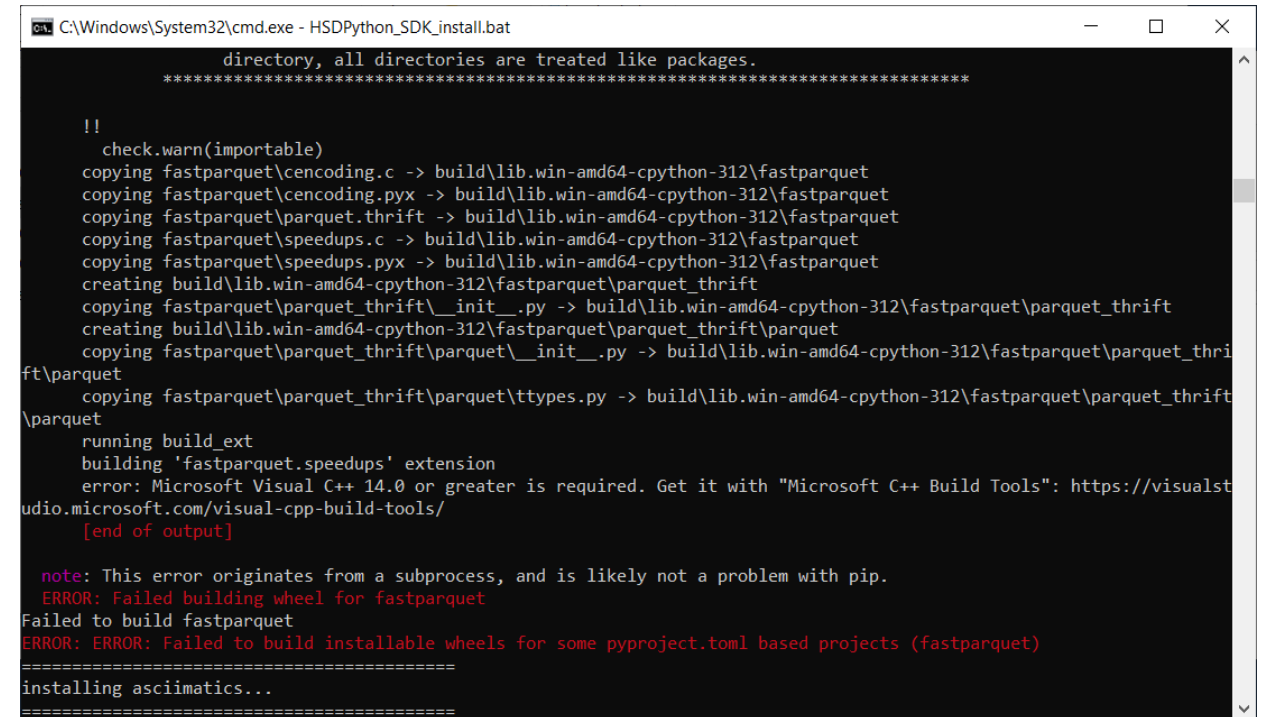
# STDATALOG-PYSDK installation

- The four modules are distributed as Python wheels
- Launch *STDATALOG-PYSDK\_install.bat* (Windows) or *STDATALOG-PYSDK\_install.sh* (Linux)
- The SDK modules and their dependencies will be installed in your Python environment
- For Linux users, further steps are needed.
  - A step-by-step procedure is described in detail in the README.



# STDATALOG-PYSDK installation troubleshooting

- When using Python 3.12, the installer may fail due to a broken internal package dependency. To solve the issue:
  - Manually install fastparquet version 2024.5.0 by running  
`python -m pip install fastparquet==2024.5.0`
  - Relaunch the installer
- For Windows users, the installer can also fail due to some missing Microsoft Visual C++ packages on the user's PC.
- The log describes which packages are missing and where you can download them.
- Next slide describes the full procedure to upgrade your Windows environment if needed



```
C:\Windows\System32\cmd.exe - HSDPython_SDK_install.bat

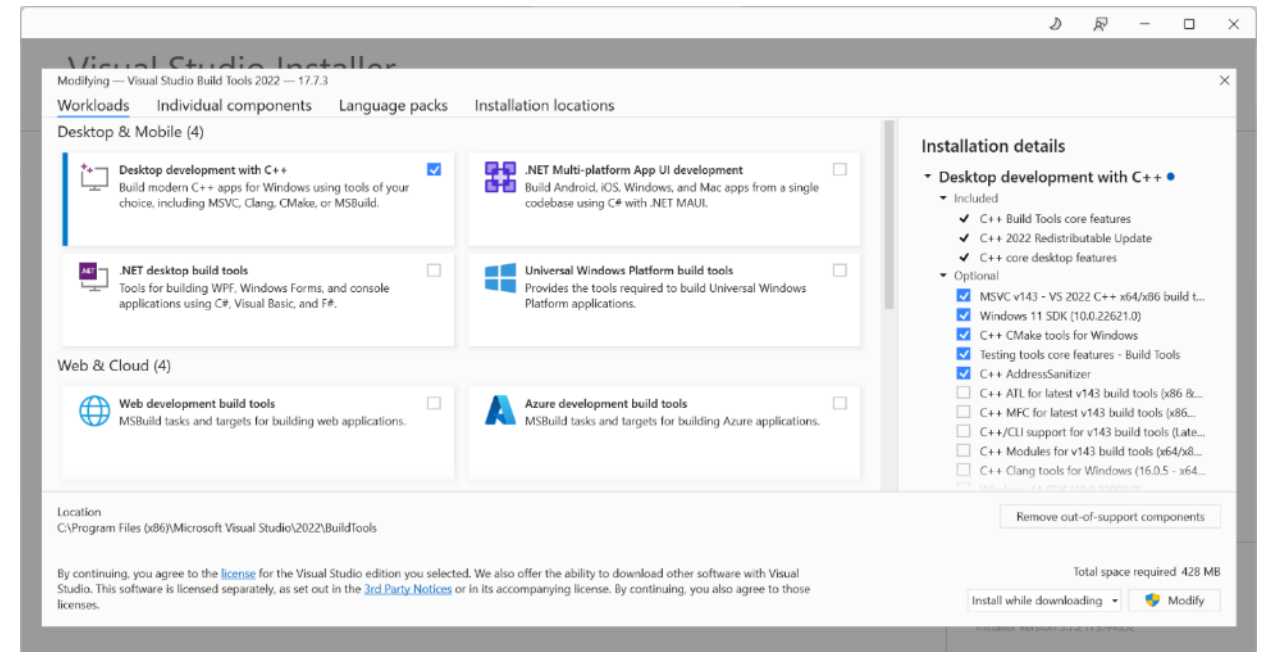
directory, all directories are treated like packages.
*****

!!
  check.warn(importable)
  copying fastparquet\cencoding.c -> build\lib.win-amd64-cpython-312\fastparquet
  copying fastparquet\cencoding.pyx -> build\lib.win-amd64-cpython-312\fastparquet
  copying fastparquet\parquet.thrift -> build\lib.win-amd64-cpython-312\fastparquet
  copying fastparquet\speedups.c -> build\lib.win-amd64-cpython-312\fastparquet
  copying fastparquet\speedups.pyx -> build\lib.win-amd64-cpython-312\fastparquet
  creating build\lib.win-amd64-cpython-312\fastparquet\parquet_thrift
  copying fastparquet\parquet_thrift\__init__.py -> build\lib.win-amd64-cpython-312\fastparquet\parquet_thrift
  creating build\lib.win-amd64-cpython-312\fastparquet\parquet_thrift\parquet
  copying fastparquet\parquet_thrift\parquet\__init__.py -> build\lib.win-amd64-cpython-312\fastparquet\parquet_thrift\parquet
  copying fastparquet\parquet_thrift\parquet\ttypes.py -> build\lib.win-amd64-cpython-312\fastparquet\parquet_thrift\parquet
  running build_ext
  building 'fastparquet.speedups' extension
  error: Microsoft Visual C++ 14.0 or greater is required. Get it with "Microsoft C++ Build Tools": https://visualstudio.microsoft.com/visual-cpp-build-tools/
  [end of output]

note: This error originates from a subprocess, and is likely not a problem with pip.
ERROR: Failed building wheel for fastparquet
Failed to build fastparquet
ERROR: ERROR: Failed to build installable wheels for some pyproject.toml based projects (fastparquet)
=====
installing asciimatics...
=====
```

# STDATALOG-PYSDK installation troubleshooting

- Download and install Microsoft C++ Build Tools from [this](#) page and wait for the installation to complete.
- Install the needed components by checking the "Desktop development with C++" checkbox on the left side and installing the modules that are checked by default on the right side.
- If some further Microsoft packages are still missing, you can select "Windows SDK" package from Microsoft C++ Build Tools. Please, select the version that matches your operating system (e.g., Windows 10 OS --> Windows 10 SDK)



# STDATALOG-PYSDK scripts

- The SDK can be used to develop a custom application either by importing the provided modules in a new project or by modifying one of the available scripts
- STDATALOG-PYSDK contains a series of ready-to-use demos and examples, distributed in the `stdatalog_examples` folder and organized in the following folders:
  - *cli\_applications*
  - *dtk\_plugins*
  - *function\_tests*
  - *gui\_applications*
  - *how-to\_notebooks*

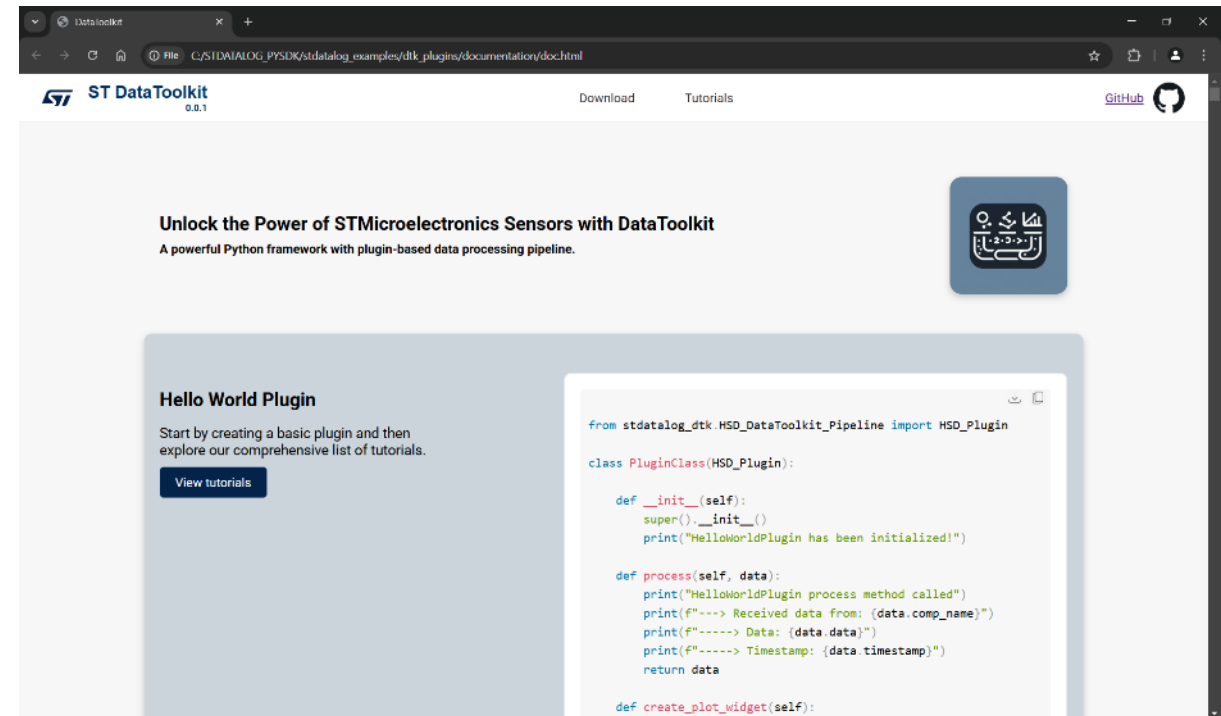




# cli\_applications

- cli\_application contains a set of ready-to-use command-line applications that demonstrate the functionalities of the stdatalog\_core:
  - *stdatalog\_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* into *SensorManager\_conf.h*).
  - *stdatalog\_data\_export.py* can convert data into TXT, CSV, TSV, PARQUET.
  - *stdatalog\_data\_export\_by\_tags.py* can be used for tagged acquisition to convert data into different files, one for each tag used.
  - *stdatalog\_dataframes.py* can save data as pandas dataframe for further processing needs.
  - *stdatalog\_hdf5\_viewer.py* is designed to display the contents of an HDF5 file using the HDF5 Viewer (additional Python packages are required).
  - *stdatalog\_plot.py* can plot the desired data.
  - *stdatalog\_plot\_large.py* designed to plot large dataset (additional Python packages are required).
  - *stdatalog\_to\_nanoedge.py* can prepare data to be imported into NanoEdge AI Studio solution.
  - *stdatalog\_to\_unico.py* can prepare data to be imported into Unico-GUI.
  - *stdatalog\_to\_wav.py* can convert audio data into a wave file.

- dtk\_plugins contains a set of plugins that can be used to create a data processing pipeline leveraging the stdatalog\_dtk (Data Toolkit framework) to extend stdatalog\_gui functionalities. In addition this folder contains an extensive html documentation that describes the stdatalog\_dtk package and all its functionalities. Available plugins are organized as follows in the "tutorial" folder:
  - simple:
    - *HelloWorldPlugin.py*: A simple plugin used to describe the basic structure of a plugin and how it works.
    - *FilterPlugin.py*: A plugin that filters received accelerometer data computing the norm of its three axis values.
    - *ProcessPlugin.py*: A plugin that implements a simple control algorithm to detect if the input filtered data is above a certain threshold.
    - *PluginWithGUI.py*: This plugin displays the same output as the *FilterPlugin.py* in a dedicated graphical widget.
  - advanced:
    - *CSVDataSavePlugin.py*: A plugin that saves the received data in a CSV file.
    - *InclinationGamePlugin.py*: A plugin that implements a simple game where the user must keep a ball within a rectangle by tilting the device.



# function\_tests

- `function_tests` contains two test scripts that demonstrate the functionalities of the `stdatalog_core` package:
  - `stdatalog_API_examples_HSDatalog.py`: Includes various function calls to the `HSDatalog` class, showcasing the full range of its functionalities.
  - `stdatalog_API_examples_HSDLink.py`: Includes various function calls to the `HSDLink` class, showcasing the full range of its functionalities. (NOTE: This script needs a compatible device (board flashed with FP-SNS-DATALOG2) connected to the PC).

# gui\_applications

- **gui\_applications** folder contains a set of ready-to-use graphical applications that demonstrate the functionalities of the STDATALOG-PYSDK leveraging the stdatalog\_gui in synergy with all the other SDK packages:
  - **stdatalog**: Contains a set of applications that realizes a complete data logging and data monitoring system.
    - *stdatalog\_GUI.py* A GUI application that allows to configure a connected device and control the data logging process storing, visualizing and labeling live data streams.
    - *stdatalog\_TUI.py* A TUI (Text-based User Interface) application that allows to configure a connected device and control the data logging process storing and labeling live data streams.
  - **stdatalog\_mc** Contains a set of applications that extends the functionalities of the stdatalog\_GUI with motor control features. This applications are designed for and work with the FP-IND-DATALOGMC. For a more detailed description of the functionalities, dependencies and usage of these applications please refer to the FP-IND-DATALOGMC documentation.
    - *stdatalog\_MC\_GUI.py* A GUI application based on the stdatalog\_GUI.py that adds the capability to retrieve and display motor control telemetries and to set motor control parameters.
    - *stdatalog\_MC\_AI\_GUI.py* A GUI application based on the stdatalog\_GUI.py that adds the capability to display AI classification results on different motor fault conditions.
  - **stdatalog\_ultrasound\_fft**: This example application is designed to work with the UltrasoundFFT application FW contained in the FP-SNS-DATALOG2 Function pack.
    - *ultrasound\_fft\_app.py*: A GUI application that allows to display analog microphone live data and its FFT. These signals are both streamed from the connected device (The FFT is performed directly on the board).
  - *assisted\_segmentation.py*: A GUI application that allows to perform assisted segmentation of acquired data files.

# how-to\_notebooks

- how-to\_notebooks contains Jupyter notebooks that guide user and shows how to use the subpackages distributed in stdatalog\_core:
  - *nb\_stdatalog\_communication.ipynb* shows how to use HSD\_link package
  - *nb\_stdatalog\_converters.ipynb* focuses on data format conversion features
  - *nb\_stdatalog\_core.ipynb* shows how to use stdatalog\_core package, focusing on core features

## 3 – Demo examples

# STDATALOG-PYSDK scripts

- You can execute the scripts in your preferred Python environment
  - i.e.: use the command `python stdatalog_plot.py`
- Discover the complete list of parameters for each script by executing with the `-h` option
  - i.e.: `python stdatalog_plot.py -h`
- As an example, next slides will show you how to use one of the most complete Python example available in the SDK: `stdatalog_GUI.py`

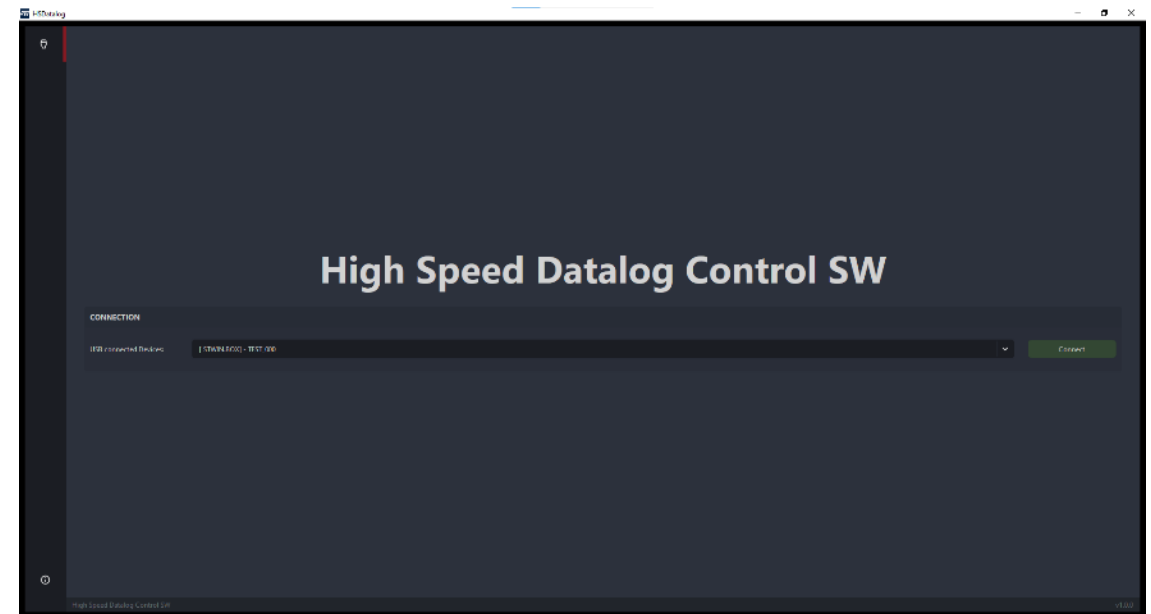
```
Windows PowerShell
(.venv) PS C:\STDATALOG_PYSDK\examples\cli_applications> python stdatalog_plot.py -h
Usage: stdatalog_plot.py [OPTIONS] ACQ_FOLDER

Options:
  -s, --sensor_name TEXT          Component name (Sensor or Algorithm) - use
                                  "all" to extract all active Component data,
                                  otherwise select a specific Component by
                                  name
  -st, --start_time INTEGER       Start Time - Data plot will start from this
                                  time (seconds)
  -et, --end_time INTEGER         End Time - Data plot will end up in this
                                  time (seconds)
  -r, --raw_data                  Uses Raw data (not multiplied by
                                  sensitivity)
  -l, --labeled                   Plot data including information about
                                  annotations taken during acquisition (if
                                  any)
  -tl, --tag_labels TEXT          A list of tag labels strings to filter and
                                  include only the corresponding entries in
                                  the converted output
  -p, --subplots                  Multiple subplot for multi-dimensional
                                  sensors
  -fp, --fft_plots                Display frequency plots for inertial sensors
                                  and microphones
  -cdm, --custom_device_model <INTEGER INTEGER TEXT>... Upload a custom Device Template Model (DTD)
                                  stdatalog_plot tool version number
  -v, --version                  [DEBUG] Check for corrupted data and
                                  timestamps
  -d, --debug                     Show this message and exit.
  -h, --help                     Show this message and exit.
  --help                         Show this message and exit.

-> Script execution examples:
python stdatalog_plot.py Acquisition_Folder_Path
python stdatalog_plot.py Acquisition_Folder_Path -s Sensor_Name
python stdatalog_plot.py Acquisition_Folder_Path -st 3 -et 6
python stdatalog_plot.py Acquisition_Folder_Path -r
python stdatalog_plot.py Acquisition_Folder_Path -l
python stdatalog_plot.py Acquisition_Folder_Path -tl tag1 -tl tag2
python stdatalog_plot.py Acquisition_Folder_Path -p
python stdatalog_plot.py Acquisition_Folder_Path -fp
python stdatalog_plot.py Acquisition_Folder_Path -cdm 1 2 custom_model.json
python stdatalog_plot.py Acquisition_Folder_Path -d
(.venv) PS C:\STDATALOG_PYSDK\examples\cli_applications>
```

# Execute *stdatalog\_GUI.py*

- *stdatalog\_GUI.py* works within the STDATALOG-PYSDK, developed in Python 3.12 on Windows and Linux environments.
- It is compatible with all firmware examples available in FP-SNS-DATALOG2 and FP-IND-DATALOGMC
  - *STDATALOG-PYSDK* requires different Python modules. The package is distributed with *installers* that solve all the required dependencies
  - Please refer to the [installation procedure](#) to install the SDK properly on your machine
- Once the board is connected via USB and the Python environment has been properly updated, you can launch the real-time plot by executing *stdatalog\_GUI.py* available in *stdatalog\_examples\gui\_applications\stdatalog\GUI*.
  - Depending on your local setup, to execute the script, you can open a command shell there and run *python stdatalog\_GUI.py*.
- Click on the Connect button to allow the connection between the board and the PC.





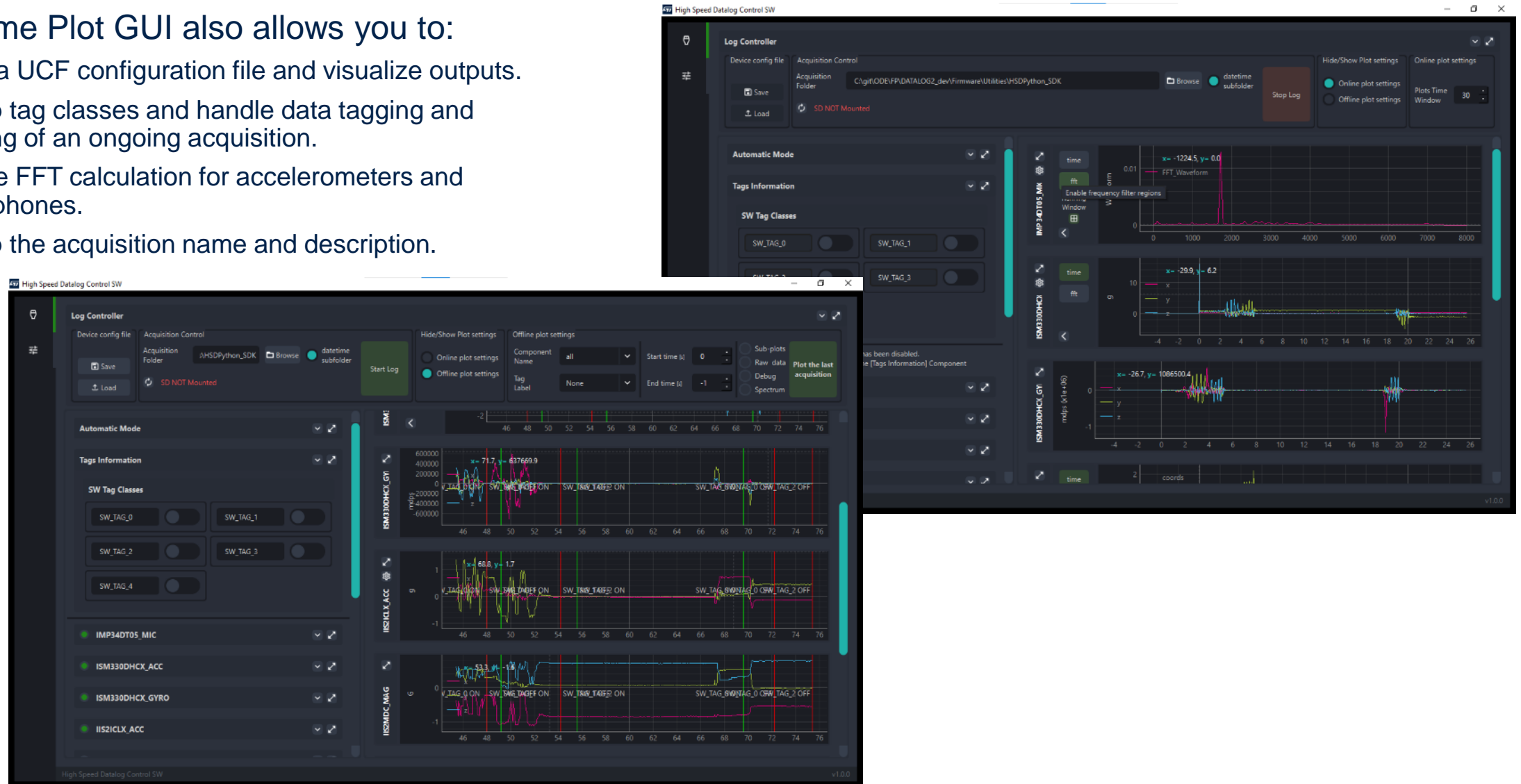
# stdatalog\_GUI.py

- Once the connection is established, you can:
  - Enable or disable the needed sensors.
  - Set up data rate, full scale, and timestamps.
  - Retrieve sensor status.
  - Load UCF to set up an MLC (machine learning core) or an ISPU (intelligent sensor processing unit).
  - Save and load a configuration via a JSON file.
  - Start or stop logging data on the PC.
- Once you click the “*Start Log*” button, data are live plotted, and the application creates a YYYYMMDD\_HH\_MM\_SS (e.g., 20200128\_16\_33\_00) folder containing the raw data and the JSON configuration file.



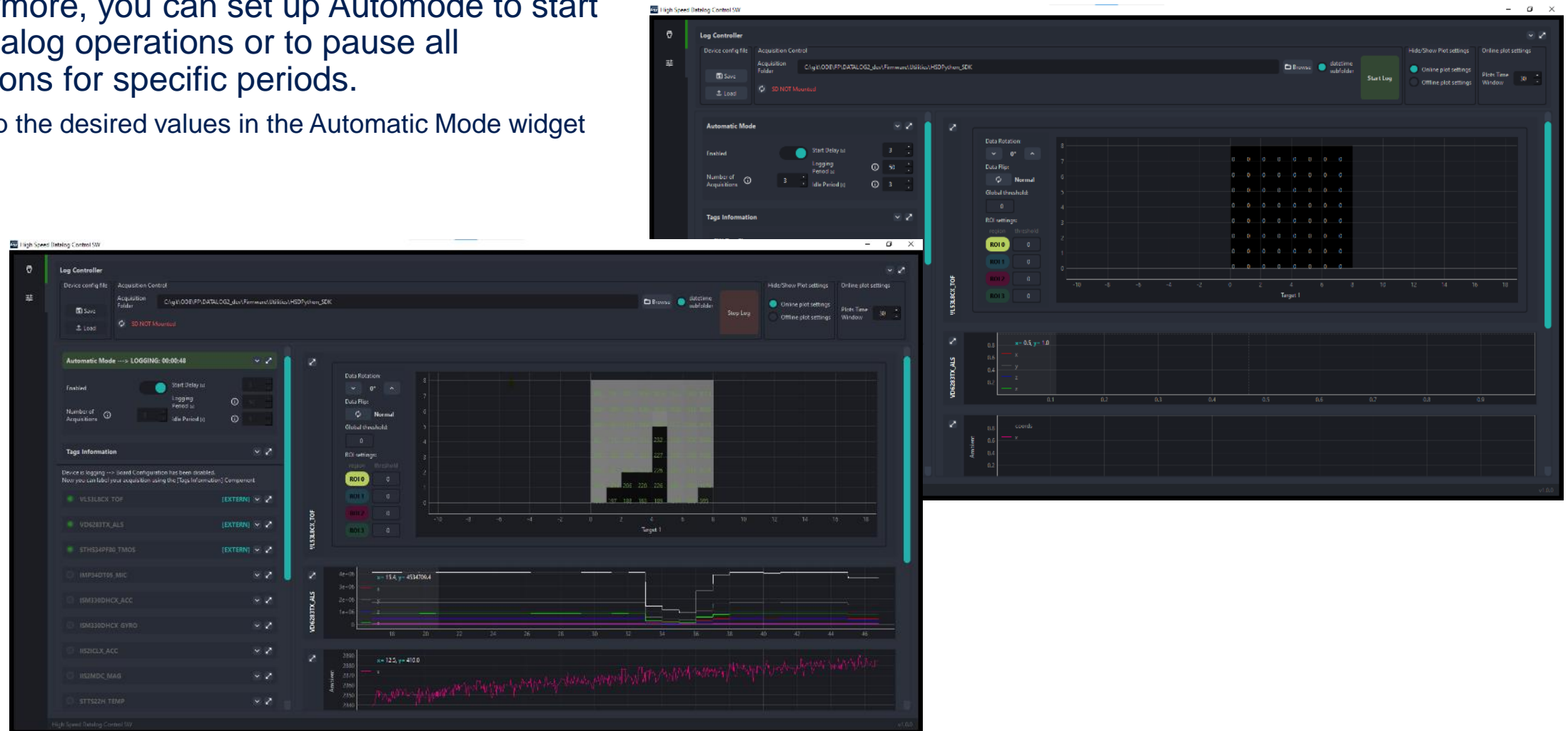
# stdatalog\_GUI.py

- Real Time Plot GUI also allows you to:
  - Send a UCF configuration file and visualize outputs.
  - Set up tag classes and handle data tagging and labeling of an ongoing acquisition.
  - Enable FFT calculation for accelerometers and microphones.
  - Set up the acquisition name and description.



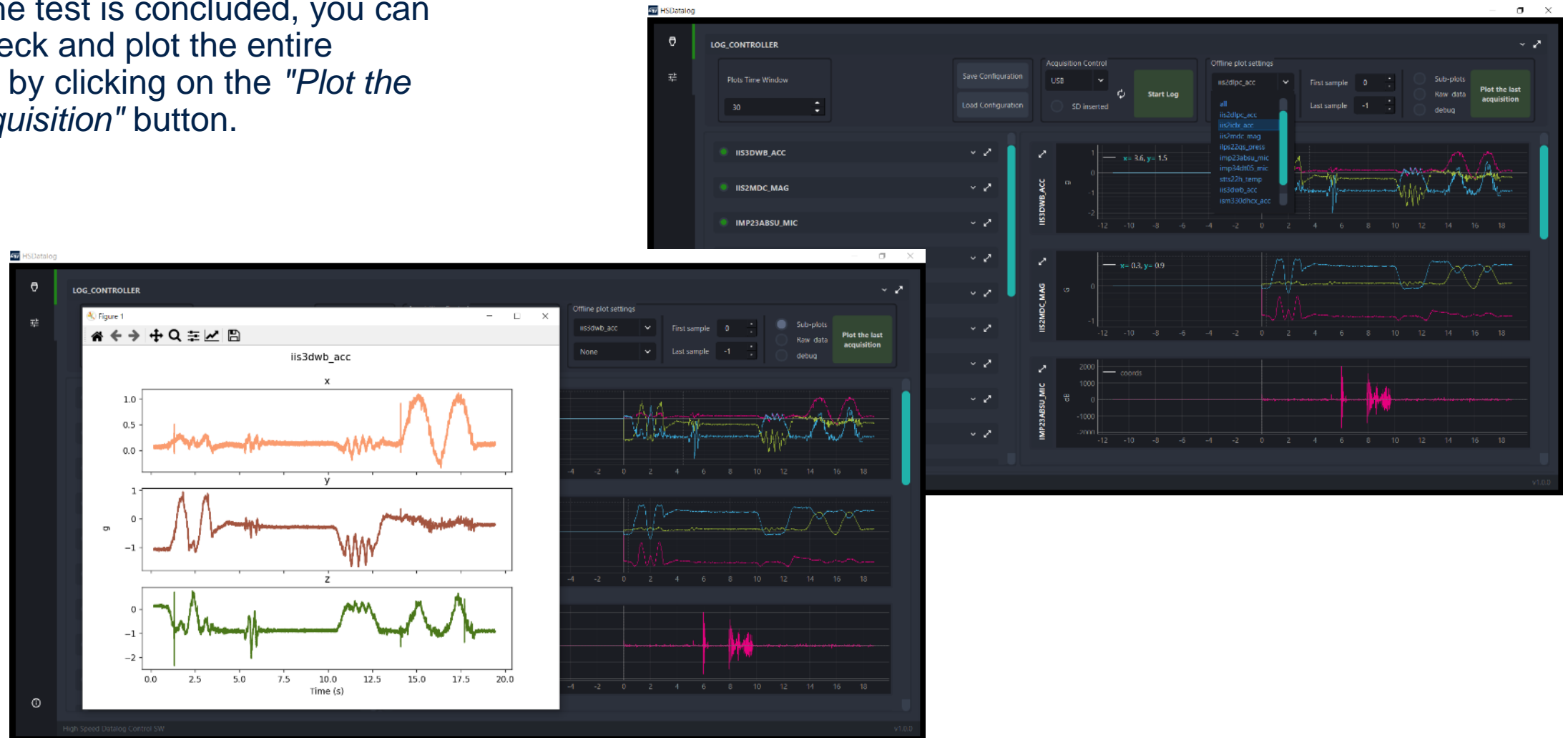
# stdatalog\_GUI.py

- Furthermore, you can set up Automode to start the Datalog operations or to pause all executions for specific periods.
  - Set up the desired values in the Automatic Mode widget



*stdatalog\_GUI.py*

- Once the test is concluded, you can also check and plot the entire dataset by clicking on the *"Plot the last acquisition"* button.



## **4 – Documents & Related Resources**

# Documents & Related Resources

## STDATALOG-PYSDK:

- **DB5446:** Python software development kit (SDK) for data logging: complete toolkit with extensive examples for developers – [data brief](#)

## FP-SNS-DATALOG2:

- **DB4865:** STM32Cube function pack for high speed datalogging and ultrasound processing – [data brief](#)
- **UM3106:** Getting started with the STM32Cube function pack for high speed datalogging and ultrasound processing – [user manual](#)

# Thank you

© STMicroelectronics - All rights reserved.

The STMicroelectronics corporate logo is a registered trademark of the STMicroelectronics group of companies. All other names are the property of their respective owners.



life.augmented