UM3208

User manual

Getting started with the STM32Cube function pack for STEVAL-PROTEUS1 evaluation kit for predictive maintenance application based on artificial intelligence (AI)

Introduction

FP-AI-PDMWBSOC2 is an STM32Cube function pack for the STEVAL-PROTEUS1 evaluation kit, developed to get motion sensor data, process them for anomaly detection, faults classification, and fault severity estimation. The processing results can be sent to the STBLESensor app or a PC terminal console. The application is based vibrometer, accelerometer and gyroscope data, using them as inputs to execute machine learning algorithms to detect anomaly behaviors, to classify operating modes or faults, and to estimate fault severity in an industrial environment. The user can configure the industrial node by remote using wireless or wired connectivity.

The monitoring of equipment conditions for predictive maintenance applications, through the STEVAL-PROTEUS1, allows to reduce the loss of productivity due to machine downtime, and unplanned maintenance costs, improving the performance of the production chain.

Using a binary data logging file, the raw data can be extracted from the sensors within the STEVAL-PROTEUS1 board and provided into the NanoEdgeAIStudio software, which allows you to extract a machine learning library for anomaly detection, classification, and extrapolation.

These NanoEdge AI libraries, built on the labeled datasets, can be easily integrated into the proposed framework architecture, facilitating the early detection of an alert status, failure type recognition, and severity estimation. FP-AI-PDMWBSOC2 provides two different HMI communication tools at user level:

- A wireless mobile app (STBLESensor) with dedicated pages to configure the node and manage the processing.
- A wired interactive Command Line Interface (CLI) to provide the same functionalities.

To start/stop the learning, detection, classification, and extrapolation phases, an additional control, indicated by LEDs, can be performed by just pressing the user button. The NanoEdgeAI libraries' functionalities are based on a set of application-level modules (Sensor Manager, Digital Processing Unit, EMData), useful to reuse and easily extendable to build other customized applications.

These application modules implement innovative software design models based on application layers built on an embedded Light object-oriented fraMework (eLooM) middleware for embedded applications powered by STM32.

Related links

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

1 FP-AI-PDMWBSOC2 software expansion for STM32Cube

1.1 Overview

The FP-AI-PDMWBSOC2 features:

- Firmware to enable predictive maintenance applications based on ML algorithms; it is able to store raw data supporting maximum sensors data rate, it buffers data according to customizable size, and processes them with anomaly detection (binary classifier), n-class classification (multi-class classifier), and extrapolation (regression) models generated through the NanoEdgeAIStudio tool.
- Firmware to develop a WPAN sensor node for predictive maintenance applications; it sends processing results, receives commands, and exchanges setting parameters via Bluetooth® Low Energy.
- STM32 wireless personal area network middleware developed within the STM32WB framework used to support Bluetooth® Low Energy 5.
- Compatible with NanoEdgeAIStudio solution to enable AI-based applications.
- NanoEdgeAI library generated to run in the STM32WB module.
- Compatible with STBLESensor app (Android and iOS) to enable AI and sensors setting, and firmware update via fast FUOTA; STBLESensor can work as a bridge to an Azure IoT Central dashboard.
- Based on accelerometer and gyroscope data respectively up to 6 kHz and 3 kHz bandwidth.
- Firmware modular example based on the embedded Light object-oriented fraMework (eLooM) to enable code reusability at application level.
- AzureRTOS: ThreadX, a small but powerful real-time operating system for embedded systems, and USBX, USB Host, and USB Device embedded stack.
- Application for datalogging in binary format from FP-SNS-DATAPRO1 v1.1.0.
- Utilities: CLI real-time control of datalogging applications, Python SDK to manipulate data and make it compliant with NanoEdgeAIStudio formats.
- Free, user-friendly license terms.

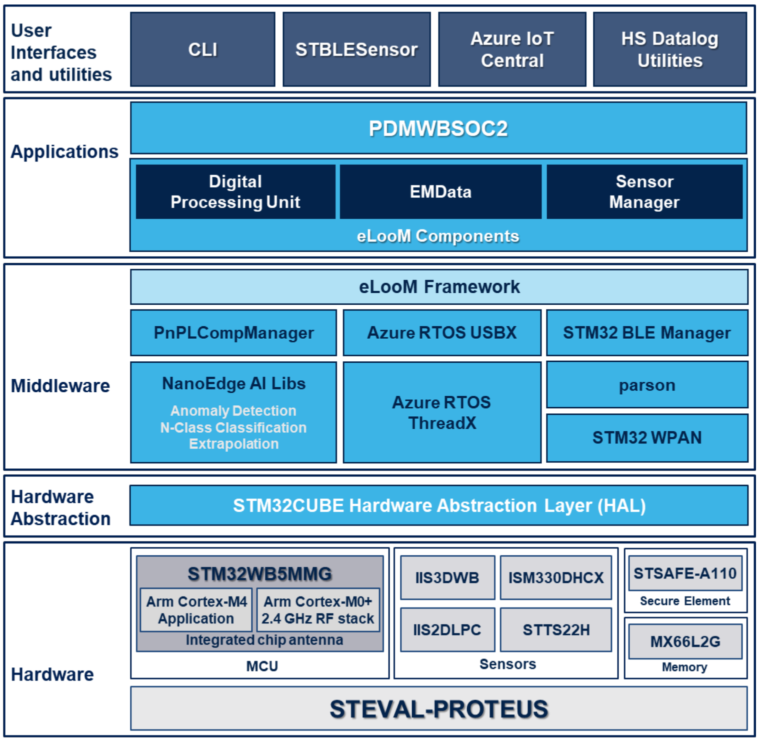
The software gathers:

- IIS3DWB vibrometer, ISM330DHCX inertial module that includes gyroscope and accelerometer, IIS2DLPC ultralow power accelerometer.
- The battery status from the STBC02.
- Green, red, and blue LEDs to indicate connection status and the running NEAI phase.

1.2 Architecture

The FP-AI-PDMWBSOC2 software is designed for the STEVAL-PROTEUS1 development kit, respecting the compliance with STM32Cube architecture, structured into a set of layers of increasing abstraction.

Figure 1. FP-AI-PDMWBSOC2 software architecture



The hardware abstraction layer (HAL) interfaces with the hardware and provides the low-level drivers and the hardware interfacing methods to interact with the upper layers (application, libraries, and stacks). It provides APIs for the communication peripherals (I²C, SPI, UART, etc.) for initialization and configuration, data transfer, and communication errors.

There are two types of HAL driver APIs:

- Generic APIs that provide common and generic functions to the entire STM32 series.
- Extension APIs that provide specific, customized functions for a particular family or a specific part number.

The package extends STM32Cube by providing a set of APIs that follows the hardware structure, including a component management layer as well as the specific layers of the boards used.

On top of such features, inherited from STM32Cube, the package FP-AI-PDMWBSOC2 is designed with a focus on code re-usability, in accordance with the eLooM-based application-level architecture.

It is based on several firmware modules that interface and offer data to other application modules according to well-defined design patterns and specific APIs.

Each firmware module (that is: Sensor Manager, Digital Processing Unit, EMData) is packed into a folder. They are independent from each other, and they can be added to your firmware application by just including the module folder inside your project.

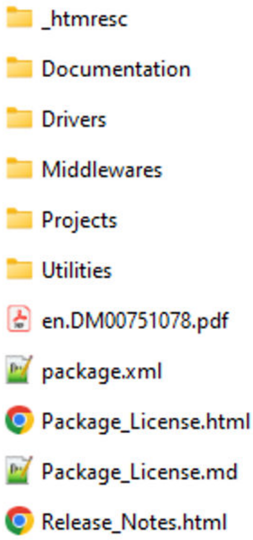
Every firmware module implements or extends drivers, events, and services made available by the eLooM framework. More specifically, here you can find:

- Drivers - objects that implement the base interface for any low-level subsystem that can be used in the firmware module (that is: SPI, I2C, UART).
- Events - 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 - additional utilities for the firmware module.

1.3 Folders

The FP-AI-PDMWBSOC2 firmware package folder structure follows the layer-based approach of the STM32Cube architecture.

Figure 2. FP-AI-PDMWBSOC2 package folder structure



The folders included in the software package are:

- **Documentation** - contains a compiled HTML file generated from the source code that details the software components and APIs.
- **Drivers** - contains the HAL drivers and the board-specific drivers for each supported board or hardware platform, including the onboard components and the CMSIS vendor-independent hardware abstraction layer for Arm Cortex®-M processor series.
- **Middleware** - contains all the libraries released by ST or third-party, as listed below:
  - **STM32_BLE_Manager**, it provides APIs to manage the BLE services
  - **Parson**, lightweight JSON library
  - **STM32_WPAN**, Wireless Personal Area Network Middleware developed within the STM32WB framework with protocols for Bluetooth connectivity
  - **PnPLCompManager**, interface used to handle PnP-like commands and properties
  - **NanoEdge_AI_Library**, default ("stub") NanoEdgeAI libraries
  - **eLooM**, embedded Light object-oriented fraMework for STM32 applications
  - **Azure RTOS USBX,** advanced industrial grade USB solution
  - **Azure RTOS ThreadX,** advanced RTOS designed for IoT applications
- **Projects** - contains one sample application for the STEVAL-PROTEUS1, in source code, for the following toolchains:
  - IAR Embedded Workbench for Arm (IAR-EWARM v9.20.1)
  - Keil® Microcontroller Development Kit for Arm (MDK-ARM v5.38.0)
  - ST Integrated Development Environment for STM32 (STM32CubeIDE v1.15.1)
  - Each IDE supports STLINK-V3MINIE (or other STLINK-V3 compact in-circuit debugger and programmer for STM32)

This folder also contains the STM32 Coprocessor Wireless BLE stack binary for the STM32WB5x.

- **Utilities** - that includes:
  - The **HSDatalog** folder that contains CLI software example to launch multi-sensors data acquisitions, PythonSDK to manipulate data acquisition folder, organize data, and make data conversions (Python script examples available).

1.4 APIs

Detailed technical information with full user API function and parameter description are in a compiled HTML file in the “Documentation” folder and in the "Doc" folder of each firmware module and eLooM middleware.

2 Application scenario

The condition monitoring by anomaly detection, classification, and extrapolation, based on STEVAL-PROTEUS1, enables the early detection of fault conditions, fault type recognition, and severity estimation directly on the equipment, facilitating the implementation of a predictive maintenance program, which reduces the loss of productivity in the production chain for unplanned downtime.

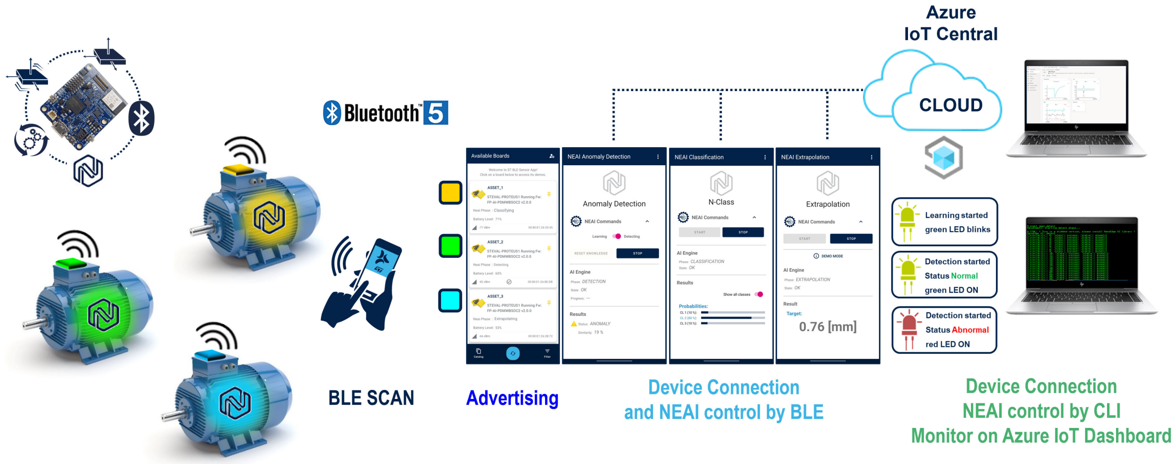
The STEVAL-PROTEUS1 evaluation board includes some sensors to acquire vibration data such as the IIS3DWB vibrometer (wide bandwidth up to 6 kHz), or ISM330DHCX accelerometer (bandwidth up to 3 kHz), useful to be processed by the embedded AD, NCC, and E library.

Bluetooth® Low Energy allows a direct connection with a smartphone and facilitates the WPAN node configuration, data monitoring, equipment status, and firmware upgrade over the air (FUOTA).

The STBLESensor app includes all the commands to start on-edge learning and detection by AD library, and also to start classification (NCC) and extrapolation (E) plus a complete set of setting commands to configure several library and sensors parameters; the same functionalities are supported by the CLI (USB) connectivity.

The following diagram shows a typical application overview for FP-AI-PDMWBSOC2.

Figure 3. FP-AI-PDMWBSOC2 application overview



The results of AD, NCC, and E processing are available in their respective dedicated pages of STBLESensor, or the CLI accessible by a standard USB terminal console.

The user can change the libraries and sensors' settings through the mobile app or CLI in the same application session.

The example is provided in the “Projects” directory, in source code, ready to be built for multiple IDEs.

The application mainly provides:

- A stable connection by USB with the CLI terminal console and, at the same time, runs a connectivity by BLE in advertising, where some status messages are also provided.
- Initialization for all the sensors data handling, and embedded NanoEdgeAI libraries including all the setting parameters.
- The battery status according to STBC02 measurements.

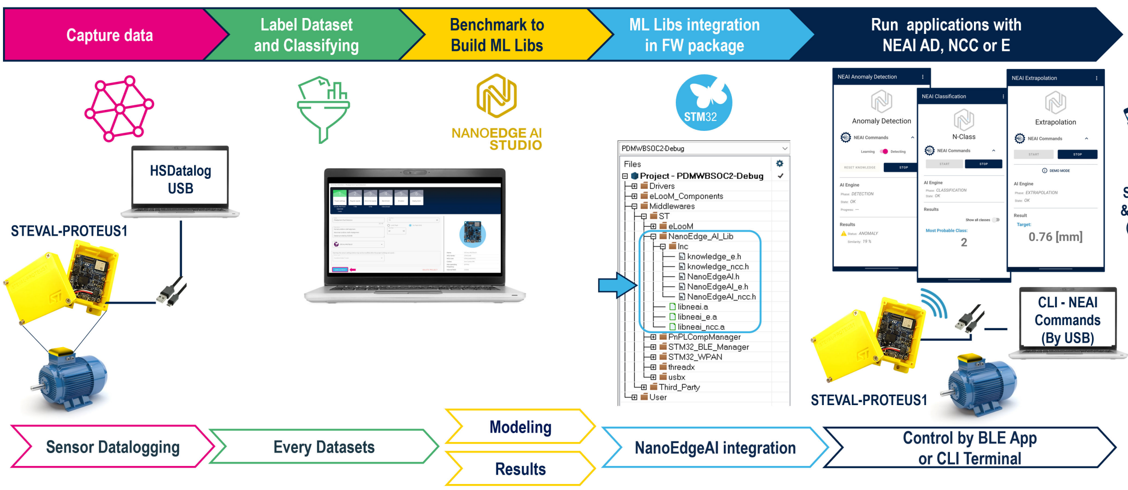
The package is mainly based on the following mandatory key elements:

- AzureRTOS: ThreadX, a small but powerful real-time operating system for embedded systems, and USBX, USB Host, and USB Device embedded stack
- The eLooM middleware to implement the object-oriented methodology for data management and processing with the AI libraries, easy to use at application level, thanks to the following reusable software modules:
  - The Sensor Manager: collects sensor data and sets the sensor parameters. It is implemented as an acquisition engine that:
    - Controls multiple task accesses to the sensor bus
    - Defines interfaces to avoid implementation dependencies
    - Dispatches events to notify when a certain amount of data is available
  - The Digital Processing Unit (DPU) component provides a set of processing blocks, which can be chained together, to create complex processing functions that use the data stream coming from the sensor manager. Available processing modules include:
    - Anomaly Detection (AD)
    - N-Class Classification (NCC)
    - Extrapolation (E)
- The EMData: defines a common data model among the eLooM framework aligned to the standards used in data science tools. Around this data format, the component also provides some useful classes and interfaces to address data-driven use-cases.

Further details are available in the "Documentation" folder.

The FP-AI-PDMWBSOC2 design and development follow a specific working flow based on different application firmware to use, and different software tools oriented to datalogging and library generation, such as explained in the next figure.

Figure 4. FP-AI-PDMWBSOC2 working flow



3 System setup guide

3.1 STEVAL-PROTEUS1 industrial sensor evaluation kit

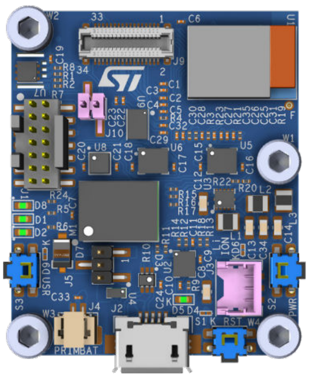
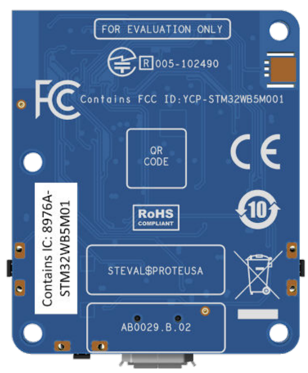
The FP-AI-PDMWBSOC2 software package is tailored for the STEVAL-PROTEUS evaluation board. This is the main board of the STEVAL-PROTEUS1 kit, which embeds an ultra-low-power and small form factor certified 2.4 GHz wireless module (STM32WB5MMG). This module supports Bluetooth® Low Energy 5.0 and others 802.15.4 proprietary protocols (for further details, refer to UM3000).

Figure 5. STEVAL-PROTEUS1 evaluation kit

The module is protocol stack royalty-free. It is featured by a dual Arm® core, a dedicated Arm® Cortex®-M0+ for radio and security tasks and a dedicated Arm® Cortex®-M4 CPU with FPU and ART (adaptive real-time accelerator) up to 64 MHz speed. It also comes with 1-Mbyte Flash memory and 256-Kbyte SRAM. These features allow running complex algorithms directly through the board and deliver ready-to-use results.

It includes ST MEMS sensors for battery-operated applications to target IoT as well as Industry 4.0.

Table 1. Hardware platform details

Features	STEVAL-PROTEUS	
Board		
Size (mm)	35 x 30 x 7	
Vibration analysis	Up to 6 kHz	
Connectivity	Bluetooth 5	
Datalog and GUI	Bluetooth® Low Energy	STBLESensor app (Android/iOS)
		Debug terminal via SWD
	CLI	Debug terminal via UART
Development tools	STLINK-V3MINIE	

The evaluation board comes with three inertial modules:
- the IIS3DWB ultra-wide bandwidth, low-noise, 3-axis digital vibration sensor
- the IIS2DLPC high-performance ultra-low-power 3-axis accelerometer
- the ISM330DHCX 3-axis accelerometer and 3-axis gyroscope with machine learning core (MLC)

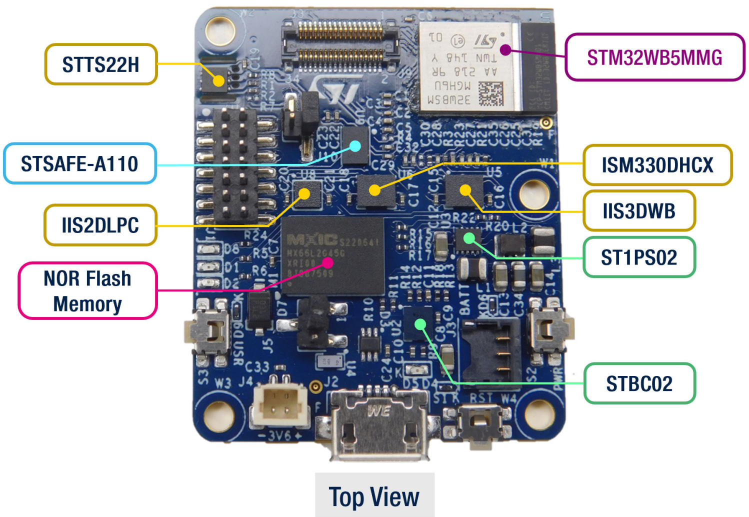
The STTS22H low-voltage, ultra-low-power, high accuracy temperature sensor has also been integrated in the board, far from the heat noise sources (the power management and the microcontroller) to provide a more precise temperature measurement. Its exposed pad and the PCB accurate design allow the temperature sensor to be in contact with the surface of the target equipment.

Moreover, the evaluation board integrates the STSAFE-A110 secure element that provides authentication and secure data management services to a local or remote host.

Furthermore, a NOR flash memory is soldered on the board. It is connected via QSPI to the STM32WB5MMG module for data buffering and event storage.

The ST1PS02 400 mA step down converter for low-power applications and the STBC02 Li-Ion linear battery charger feature the power management.

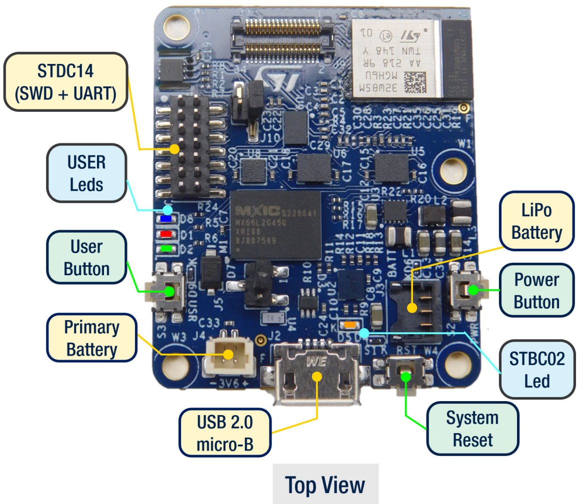
Figure 6. STEVAL-PROTEUS main devices



The STEVAL-PROTEUS can be powered via:
- a LiPo rechargeable battery
- USB (5 V at 500 mA)
- a primary battery (not included in the kit)

All components are mounted exclusively on the top side of the PCB to ensure easy mounting on other equipment, even when using the plastic case for the board and the LiPo battery (included in the kit).

Figure 7. STEVAL-PROTEUS power connector, SWD, user button, and LED



3.2 Hardware setup

The hardware prerequisites to prepare the kit are:

- two USB Type-A to Micro-B cables: one to monitor the debug console and program the board via the STDC14 connector, and one to use a CLI via USB
- an STLINK-V3MINIE (not included in the kit) for programming and debugging

The STEVAL-PROTEUS1 kit includes a plastic case with its related screws to host the board and the LiPo battery, allowing an easy fixing (for demo purposes only).

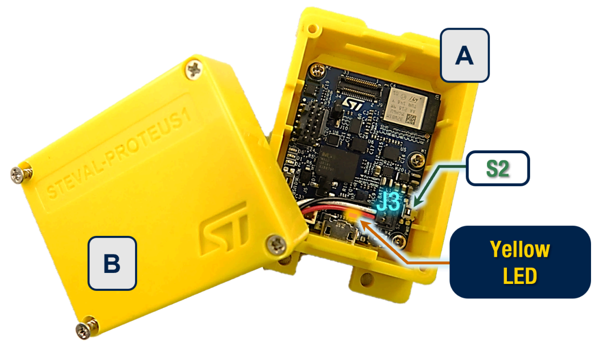
To assemble the kit, follow the procedure below.

1. Fix the main board to the case bottom (**A**) with the four screws included in the kit.

Note: Pay attention to the board orientation.

2. Put the LiPo battery in the top case (**B**). Then, insert the battery cable into the dedicated hole.
3. Put the cover on the battery and close it using two screws.
4. Plug the LiPo battery connector to the J3 connector.
5. Close the A and B case parts using four screws.
6. Use something thin (for example, a needle) to access the holes and press the push buttons.
7. Power on the board by pushing and hold the **S2** button for at least three seconds: the yellow LED turns on, then off. To power off, push and hold the **S2** button until the yellow LED turns on.

Figure 8. STEVAL-PROTEUS mounting and power-on



3.3 Software setup

To use the FP-AI-PDMWBSOC2, update, and run the application, you need:

- the STBLESensor app, available for Android and iOS, to handle the data exported by a Bluetooth® Low Energy device using the BlueST protocol. The app allows running the application and performing the firmware update over-the-air (FUOTA).

Figure 9. QR codes for STBLESensor

iOS

Android

- A Windows™ (v10 or higher) PC, with TeraTerm or another terminal emulator software
- NanoEdgeAIStudio (v4.5.0 or higher)
- STM32CubeProgrammer software
- A development toolchain and compiler. The STM32Cube expansion software supports the three following environments:
  - IAR
  - Keil®
  - STM32CubeIDE

3.3.1 STEVAL-PROTEUS flash memory mapping

The STM32WB5MMG comes with a 1 Mbyte flash memory, shared between the two Arm® cores as the user part to be run into the CPU1 (Cortex® M4) and the secure part to be run into the CPU2 (Cortex® M0+). The user part starts from the beginning of the flash memory, whereas the secure part is placed at the end.

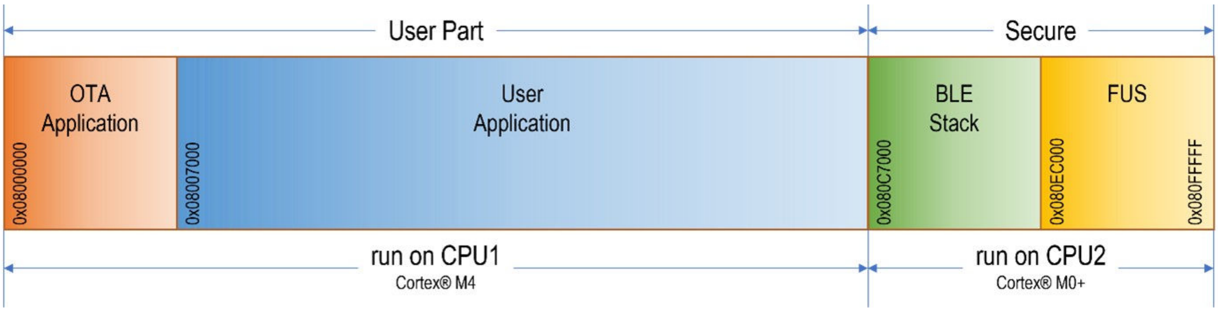
A specific memory mapping helps to develop an application easy to update.

To integrate the FUOTA features, design the firmware package as follows:

- the bootloader, placed in the first main memory address (0x08000000), must be an OTA application designed to handle the firmware update
- the user application, placed in the specific memory address (0x08007000), can be updated with the FUOTA and contains the main application
- the secure sectors include the application running in the CPU2 core; the secure part (FUS) is placed at the end and its size depends on the wireless Bluetooth® Low Energy stack to be used.

You can also update the Bluetooth® Low Energy stack using the same FUOTA functionality.

Figure 10. **STEVAL-PROTEUS memory map for Bluetooth® Low Energy OTA applications**



For further details on this topic, refer to AN5247, which describes the procedure for over-the-air (OTA) firmware update on ST32WB devices with Bluetooth® Low Energy connectivity.

3.3.2 Toolchains settings for STM32WB memory mapping

Usually, the binary code generated with default settings, for any toolchain, considers the firmware starting address as the first address of the main memory (0x08000000).

So, to generate the project with OTA capabilities correctly, follow the steps below.

1. Make sure that the OTA application firmware has already been downloaded into the STEVAL-PROTEUS from the first main memory address.
2. Open *$PROJ_DIR$\..\Src\system_stm32wbxx.c* and:
   – Verify that the `USER_VECT_TAB_ADDRESS` define is uncommented
   – Verify that `VECT_TAB_OFFSET` is defined as 0x00007000U
3. Apply the following changes for each IDE toolchain:
   – **IAR EWARM**
     Open [**Project**]>[**Options**]>[**Linker**] and use *$PROJ_DIR$\stm32wb5mxx_flash_cm4_ota.icf* as the linker configuration file
   – **Keil® MDK**
     Open [**Project**]>[**Options**]>[**Target**] and set IROM1 as 0x8007000 for start and 0x79000 for size.
     Open [**Project**]>[**Options**]>[**Linker**] and use *.\stm32wb5mxx_flash_cm4_ota.sct* as the scatter file.
     Open [**Project**]>[**Options**]>[**Linker**] and add two lines in the misc controls:
     ```
     --keep *.o(TAG_OTA_START)
     --keep *.o(TAG_OTA_END)
     ```
   – **STM32CubeIDE**
     Open [**Properties**]>[**C/C++ Build**]>[**Settings**]>[**Tool Settings**]>[**MCU GCC Linker-**]>[**General**] and use *${workspace_loc:/${ProjName}/STM32WB5MMGHX_FLASH_ota.ld}* as the linker script
4. Download the generated firmware into the STEVAL-PROTEUS that you are using from the address 0x08007000

3.3.3 STEVAL-PROTEUS flash programming via Bluetooth® Low Energy OTA

The STEVAL-PROTEUS comes with a preloaded application firmware, supporting the condition-based maintenance via Bluetooth® Low Energy connectivity (STSW-PROTEUS).

This preloaded firmware supports the Bluetooth® Low Energy OTA. So, to use directly the FP-AI-PDMWBSOC2 application, update the application coprocessor binary or the wireless coprocessor binary through the STBLESensor app.

3.3.3.1 Application firmware update

To update the main application via Bluetooth® Low Energy OTA, follow the steps below.

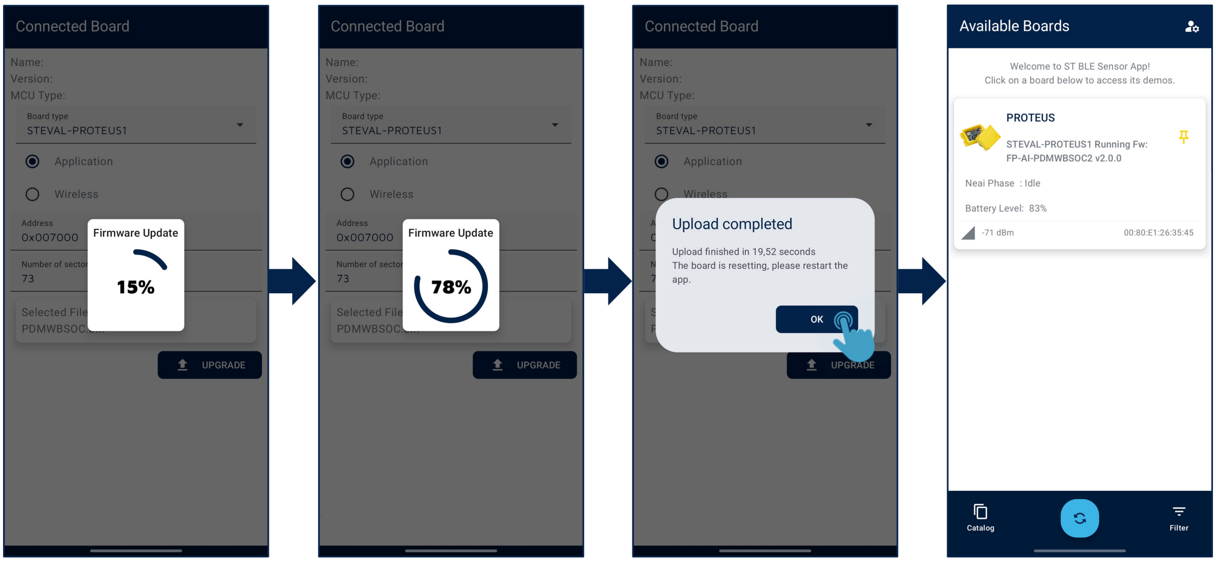
1. Copy the PDMWBSOC.bin binary file, available in the *$PROJ_DIR$\Projects\STM32WB5MMG-PROTEUS\Applications\PDMWBSOC2\Binary* folder, inside a mobile device folder.
2. Launch the mobile app and connect to the Proteus board.
3. Touching the gear icon, select *Firmware Update* option.
4. Tap on *Selected File* label, browse inside your smartphone memories and select PDMWBSOC2.bin binary file.

5. Tap on *Upgrade* button to launch OTA firmware upgrade.

Figure 11. **Start FUOTA and select FP-AI-PDMWBSOC binary application**

6. Wait a few dozen seconds until the completion message is displayed.

Figure 12. FUOTA progress and completion



As the upload has been completed, the board reboots as *STEVAL-PROTEUS* according to the uploaded firmware. It is visible in the list of the Bluetooth® Low Energy connectable devices.

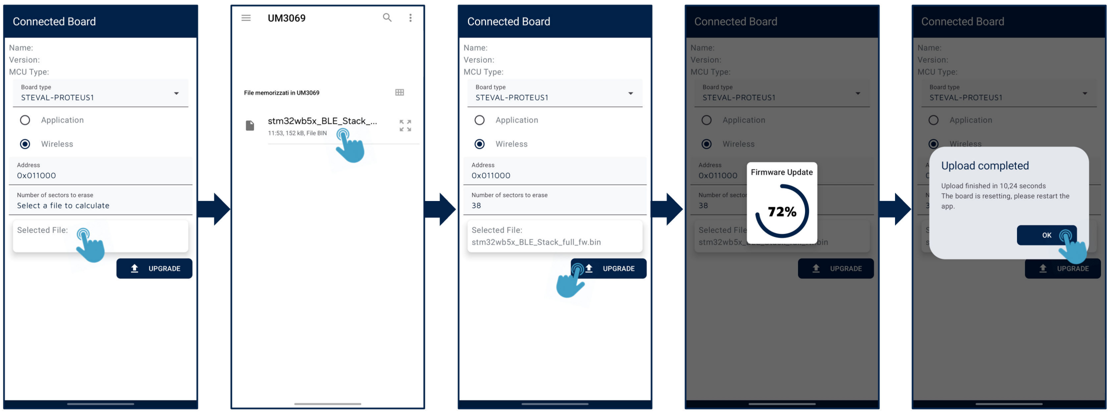
3.3.3.2 *Wireless stack firmware update*

To update the wireless stack via Bluetooth® Low Energy OTA, follow the steps below.

1. Copy the *stm32wb5x_BLE_Stack_full_fw.bin* binary file, available in the *$PROJ_DIR$ \Projects\STM32WB_Copro_Wireless_Binaries\STM32WB5x* folder, inside a mobile device folder.
2. Launch the mobile app and connect to the Proteus board.
3. Select Wireless Coprocessor Binary option, address is updated automatically.
4. Tap on *Selected File* label, browse inside your smartphone memories and select *stm32wb5x_BLE_Stack_full_fw.bin* binary file.
5. Tap on *Upgrade* button to launch OTA firmware upgrade.

6. Wait a few dozen seconds until the completion message is displayed.

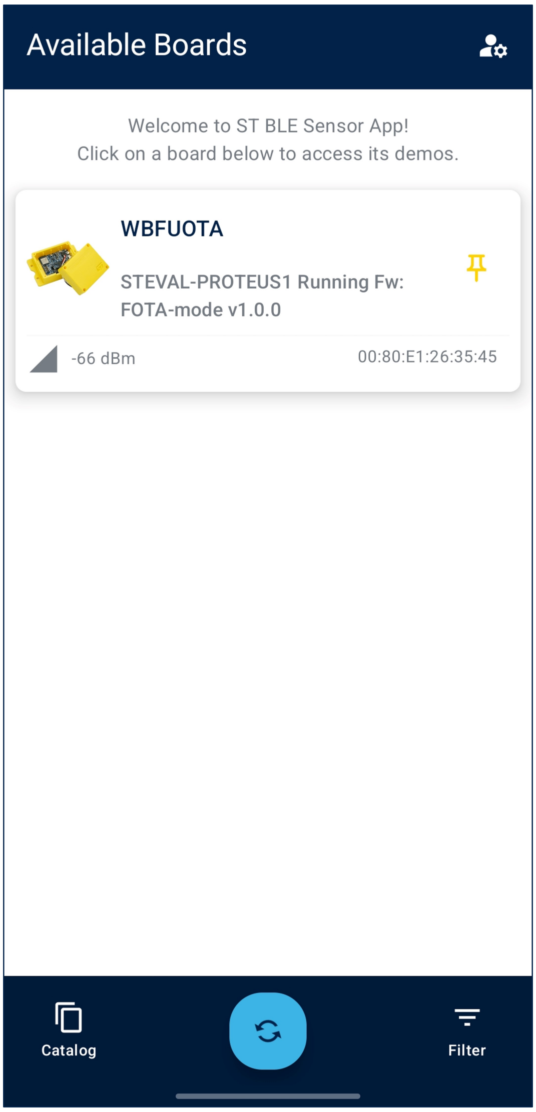
Figure 13. Start FUOTA and select wireless stack binary



Note: *The process is not yet complete. The firmware is stored in an area of the flash memory buffer. Then, the FUS copies the binary into the right flash memory area. Do not disconnect the power from the board and do not press any buttons on it. This process can take up to 25 seconds. During this time, the board is no longer visible in the device list of the app. Just tap on the Search button until you see the WBFUOTA board in the list of the Bluetooth® Low Energy connectable devices.*

Important: *The wireless stack update procedure erases the application firmware. So, you must update the related user application sector just after the stack update (see Application firmware update).*

Figure 14. Searching the WBFUOTA in the device list



After tapping on the *WBFUOTA* board, update the application firmware as shown in Application firmware update.

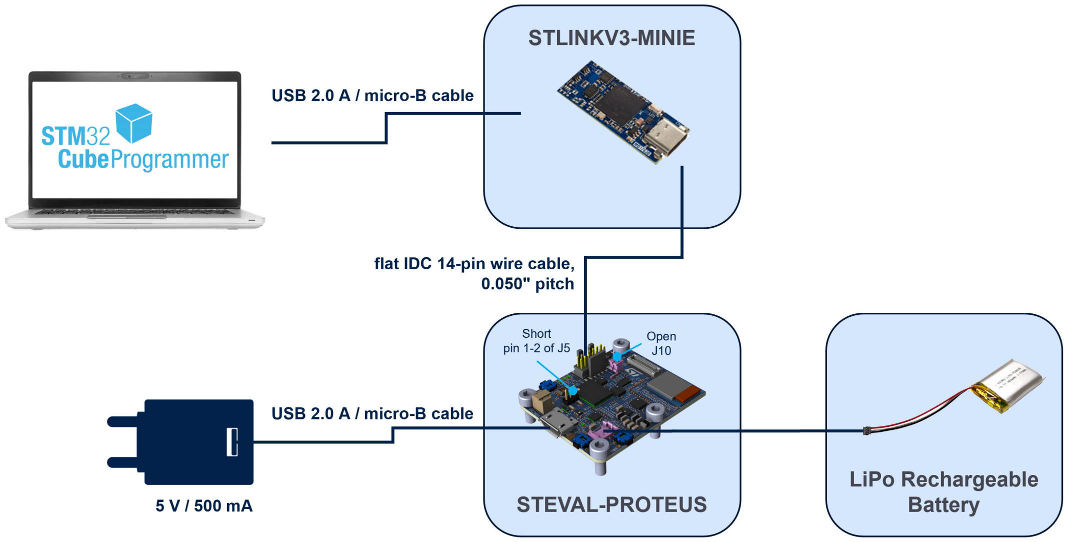
3.3.4 STEVAL-PROTEUS flash memory programming via ST-LINK

The STEVAL-PROTEUS flash memory programming methodology is stack independent.

There are different binaries files for the wireless stack and the application.

Install the STM32CubeProgrammer software on your PC and arrange the connection as shown below.

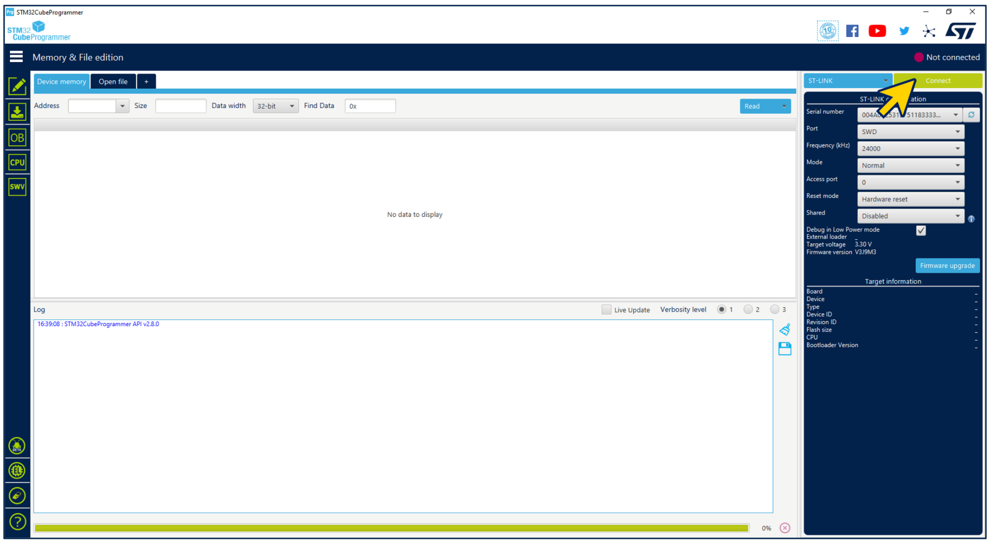
Figure 15. STEVAL-PROTEUS flash programming scenario



Note: You can choose to power the STEVAL-PROTEUS via USB or by a LiPo battery. You can also use both at the same time. When using only the battery, press the power button at least for three seconds to turn on the system.

Launch the STM32CubeProgrammer software and connect the attached board, as highlighted below.

Figure 16. Connecting the board

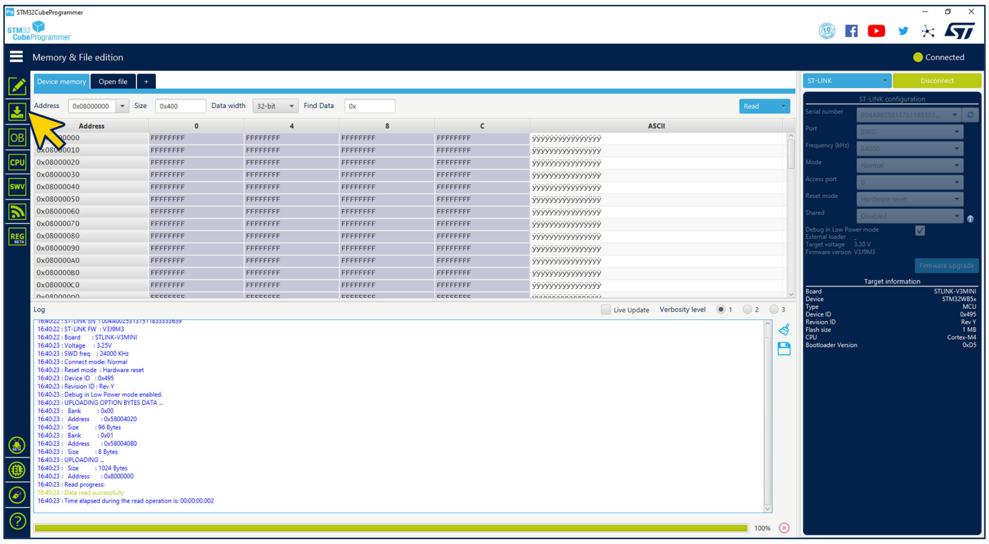


3.3.4.1 Application firmware update

Step 1. Launch STM32CubeProgrammer if not done already (see Figure 2).

Step 2. Select *Erasing & Programming*.

Figure 17. Erasing & Programming selection



Step 3. Browse to select the application binary file you are going to use (see Figure 18, arrow No. 1).

Step 4. Set the right start address to download the binary (see Figure 18, arrow No. 2).

Note: The table below shows the right start address to use with the chosen application binary.

Table 2. ***STM32WB Application Binaries to use***

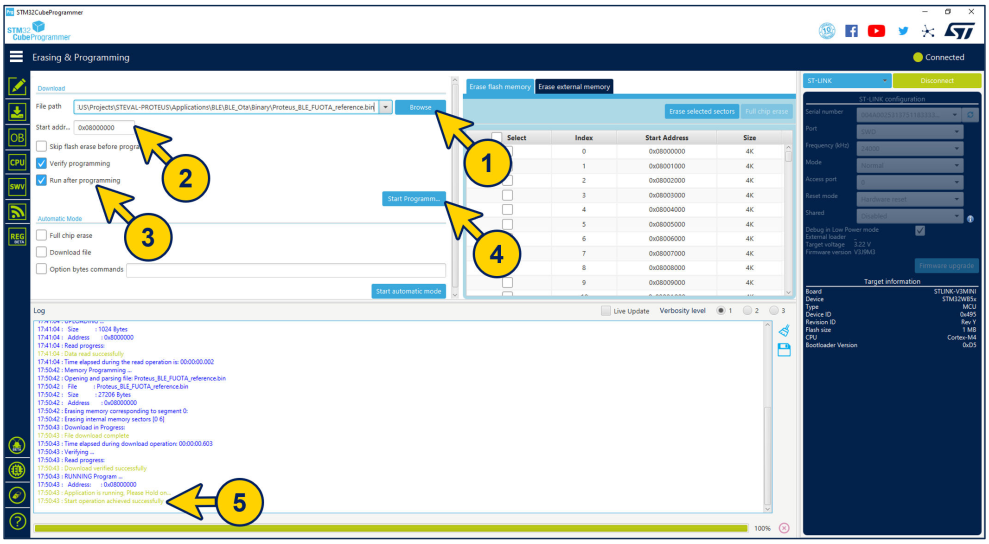
Hardware	*Connectivity*	*Application*	*Application binary*	*Install address*
STEVAL-PROTEUS	*Bluetooth® Low Energy*	*OTA*	*PROTEUS_BLE_FUOTA_RFWKPCLK_HSE_DIV1024.bin*	*0x08000000*
		FP-AI-PDMWBSOC2	*PDMWBSOC2.bin*	*0x08007000*
		FP-SNS-DATAPRO1	*FP-SNS-DATAPRO1_1_1_0.bin*	*0x08007000*

Step 5. Check both *Verify programming* and *Run after programming* (see Figure 18, arrow No. 3).

Step 6. Select *Start Programming* (see Figure 18, arrow No. 4).

Step 7. Wait for all operations to be successfully completed (see Figure 18, arrow No. 5).

Figure 18. Application firmware installation overview



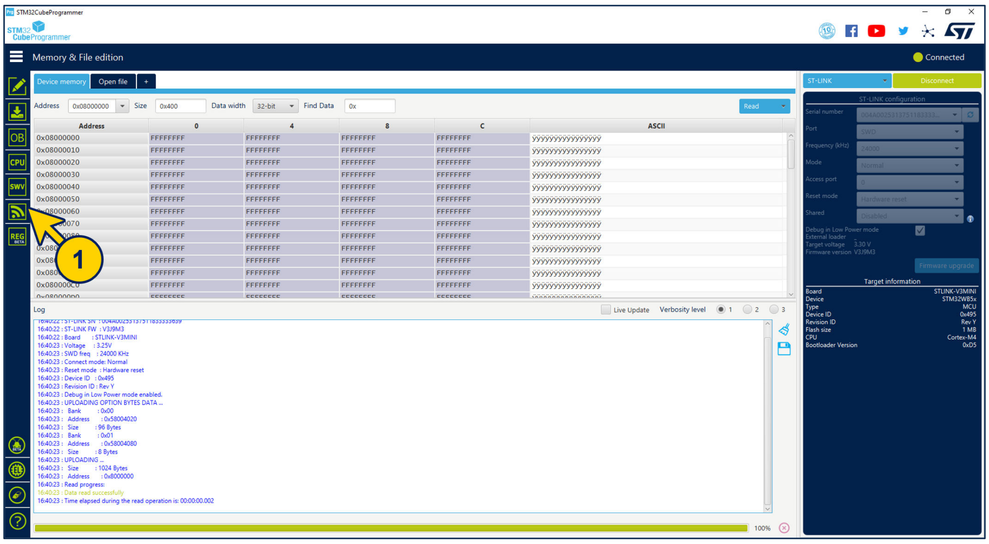
If no error occurs, the application firmware has been correctly downloaded into the STEVAL-PROTEUS flash memory.

3.3.4.2 *Wireless stack firmware update*

Step 1. Launch STM32CubeProgrammer if not done already (see Figure 2) and connect the board.

Step 2. Select *Firmware Upgrade Services* (see Figure 20. Firmware upgrade overview, arrow No. 1).

Figure 19. Firmware upgrade services



Step 3. Select *Start FUS* (see Figure 20, arrow No. 2).

Note: *If a failure message appears ("Start FUS failed!" or "Start FUS Operation Failure! Please, try again"), relaunch the command.*

Step 4. Browse to select the stack binary file you are going to use (see Figure 20, arrow No. 3), available in *$PROJ_DIR$ \Projects\STM32WB_Copro_Wireless_Binaries\STM32WB5x*.

Step 5. Set the right start address to download the binary (see Figure 20, arrow No. 4).

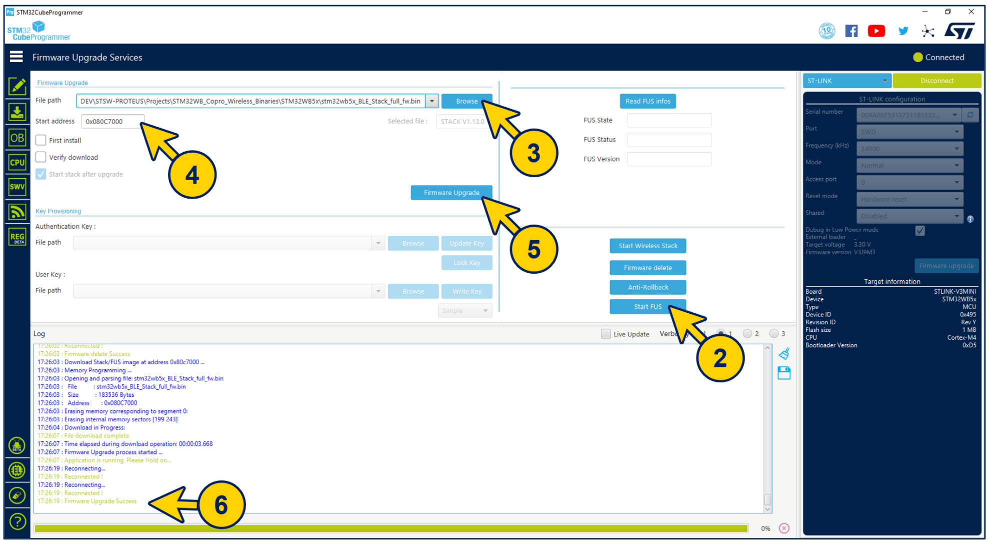
Table 3. STM32WB coprocessor wireless binaries to use

Hardware	Connectivity	Role	Wireless coprocessor binary	Install address
STEVAL-PROTEUS	BLE	Slave	stm32wb5x_BLE_Stack_full_fw.bin (V1.20.0)	0x080C7000

Step 6. Select *Firmware Upgrade* (see Figure 20, arrow No. 5).

Step 7. Wait for *Firmware Upgrade Success* (see Figure 20, arrow No. 6).

Figure 20. Firmware upgrade overview

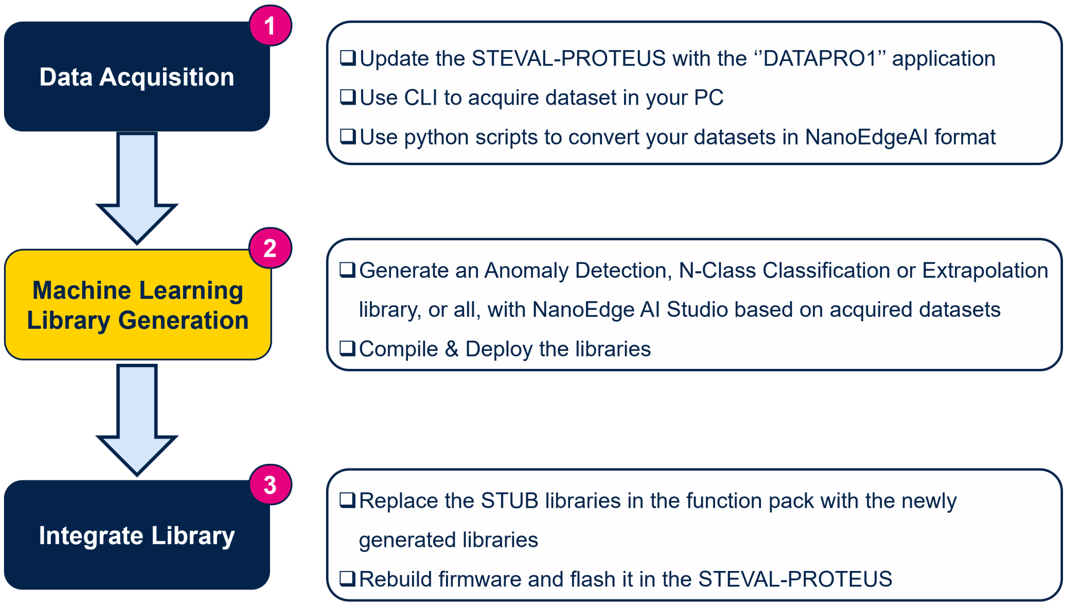


Note: *The wireless coprocessor stack firmware update procedure corrupts the data already stored in the first part of the flash memory (OTA application). So, after the stack update procedure, you must download the OTA application as explained in Application firmware update.*

4 Making and embedding NEAI libraries

The following section describes how to generate the Anomaly Detection, N-Class Classification, or Extrapolation library by using the NanoEdgeAIStudio, starting from the dataset acquisition, NEAI library creation, deployment, integration, and testing on the STEVAL-PROTEUS1, using the FP-AI-PDMWBSOC2.

Figure 21. Dataset acquisition, NEAI libraries creation, deployment, and integration



4.1 Data acquisition by DATAPRO1 application

The DATAPRO1 application firmware for the STEVAL-PROTEUS1 comes with Bluetooth® Low Energy and FUOTA features. The STEVAL-PROTEUS1 kit is already programmed with the BLE stack and OTA application. So, the DATAPRO1 application firmware can be downloaded into the MCU via OTA as explained in STEVAL-PROTEUS flash programming via Bluetooth® Low Energy OTA.

4.1.1 Acquiring sensors data using DATAPRO1

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

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

The data can be stored either in the embedded NOR flash memory or in a folder of the PC connected via USB.

This function pack also provides a solution to transfer the files stored in the NOR flash memory to a mobile device via Bluetooth.

A detailed description of DATAPRO1 operation is available in the UM3396.

4.1.2 Convert the dataset in NanoEdgeAI format

Since NanoEdgeAIStudio does not accept *.dat files as input datasets, users must convert their acquisition folders into *.csv files formatted as shown below:

Figure 22. **NanoEdgeAI dataset format for AD and NCC libraries**

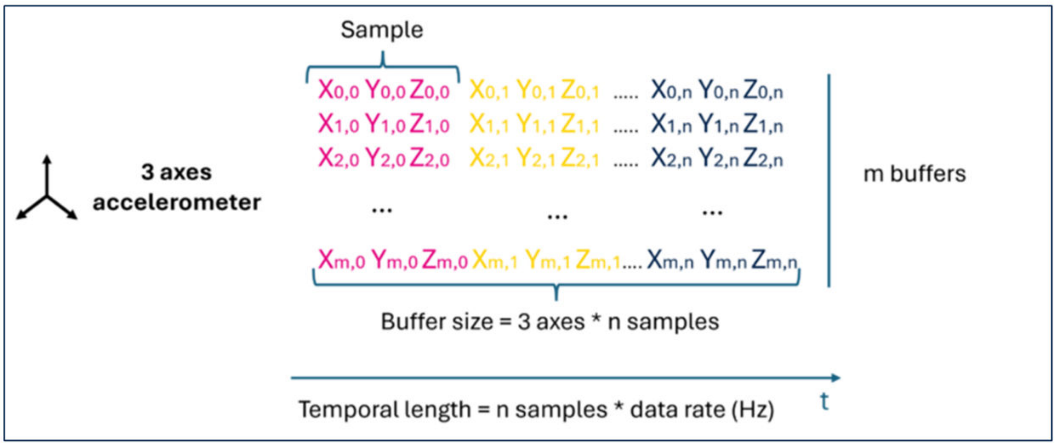
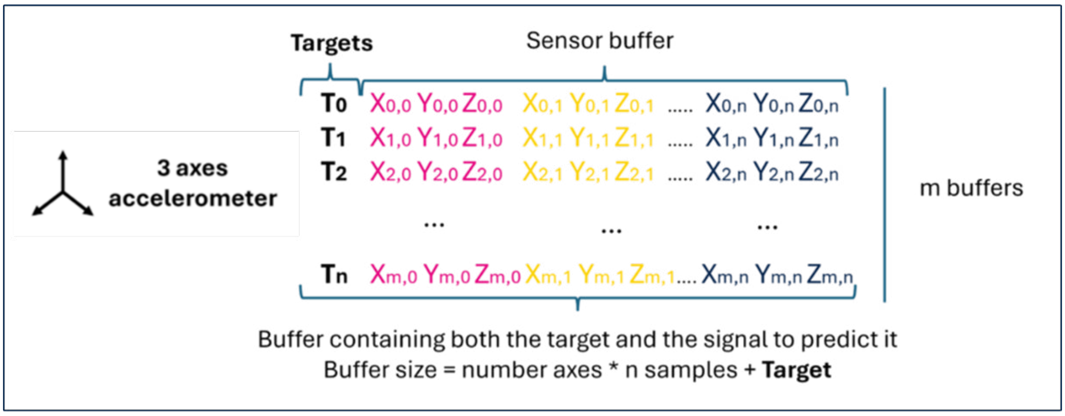


Figure 23. **NanoEdgeAI dataset format for E libraries**



The format required for extrapolation libraries is slightly different compared to the one required for Anomaly Detection and Classification. Extrapolation datasets must contain, the target column where the user can specify the value of the target that corresponds to each signal.

Most of time T_0 T_n assumes the same value and it varies from one dataset to another.

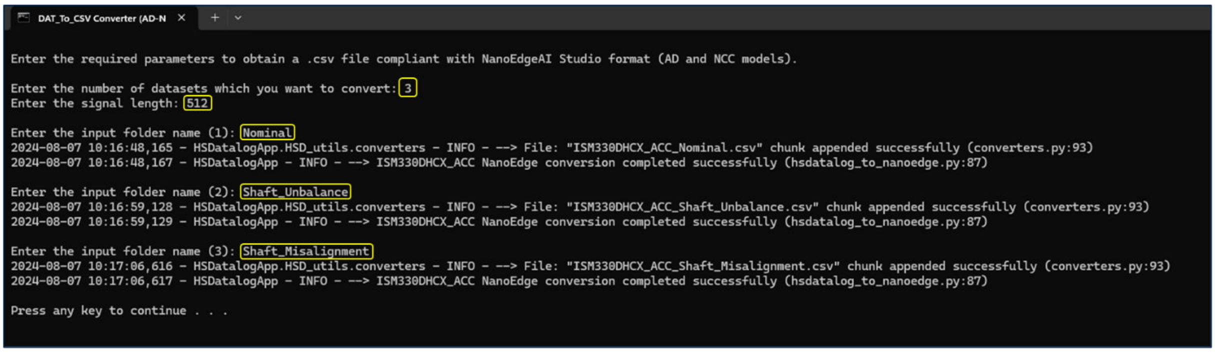
The easiest way to convert your datasets to make it compliant with the NanoEdgeAI format is to use the ready-to-use batch files located in the folder *$PROJ_DIR$ \Utilities\SwUtilities\HS_Datalog\PROTEUS_batch_file_examples* , here they are placed the *HS-DL_NanoEdge_Conversion_AD_NCC.bat* and *HS-DL_NanoEdge_Conversion_E.bat*.

The user of course uses one of them based on whether they want to generate an AD, NCC, or E library.

After launching the *HS-DL_NanoEdge_Conversion_AD_NCC.bat*, the user needs to enter the following information:

1. The quantity of acquisition folders that they want to convert in *.csv file.
2. The signal length, which is the buffer size.
3. The input folders' names (one at a time), for each name entered, the corresponding *.csv file is created.

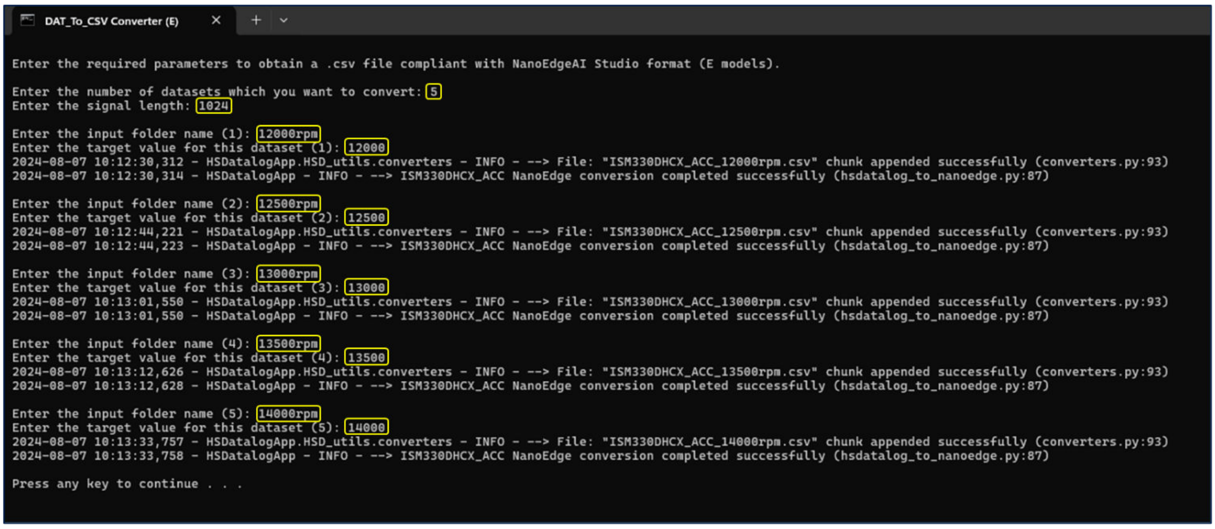
Figure 24. **Datasets converter batch example (AD/NCC)**



After launching the *HS-DL_NanoEdge_Conversion_E.bat*, the user needs to enter the following information:

1. The quantity of acquisition folders that they want to convert in *.csv file.
2. The signal length, which is the buffer size.
3. The input folders' names.
4. The target value that corresponds to the dataset specified under the previous point.

Figure 25. **Datasets converter batch example (E)**



When conversions are completed, the user finds *.csv file inside each folder acquisition; they can be directly imported into the NanoEdgeAIStudio.

Figure 26. **Converted dataset folder**

Note: *Consecutive samples collected by a sensor form buffer, and concatenating a certain number of buffers, you get a dataset (or signal). Inside .csv converted dataset you find a certain number of lines, which are the buffers, and its length is the signal length.*

Note: *Buffer size must be a power of 2 and the dataset should contain at least 300 lines.*

4.2 Library generation using NanoEdgeAI Studio

The NanoEdge AI Studio can generate libraries of four types:

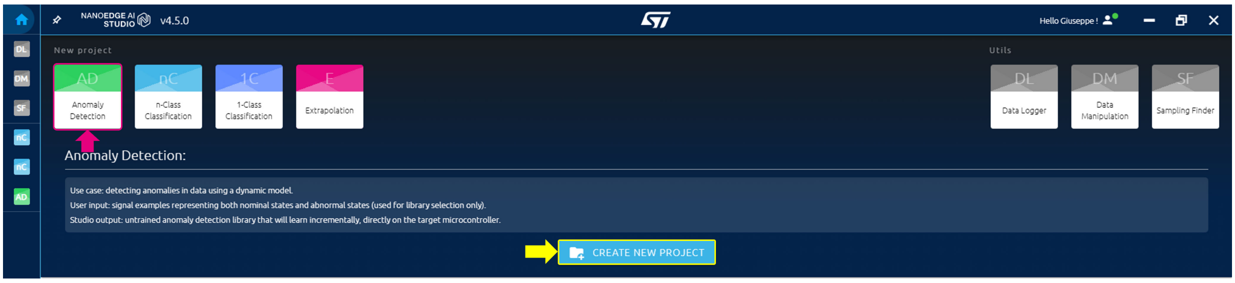
- Anomaly detection
- 1-Class-Classification
- N-Class Classification
- Extrapolation

The user must be careful to collect the dataset correctly, during the different working conditions to be monitored, and then use it to generate the desired ML model.

4.2.1 NEAI anomaly detection library

To generate the anomaly detection library using the NanoEdgeAI tool, you need to start by creating the new project using the AD button as shown in the following figure.

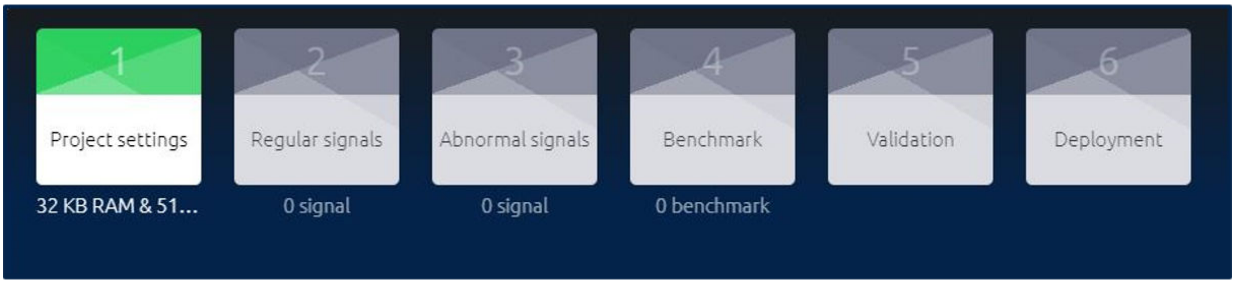
Figure 27. **Start a project (AD)**



The text shows the information on what the anomaly detection is good for. You can create the project by clicking on the **CREATE NEW PROJECT** button.

The process to generate the anomaly detection library consists of six steps as shown below.

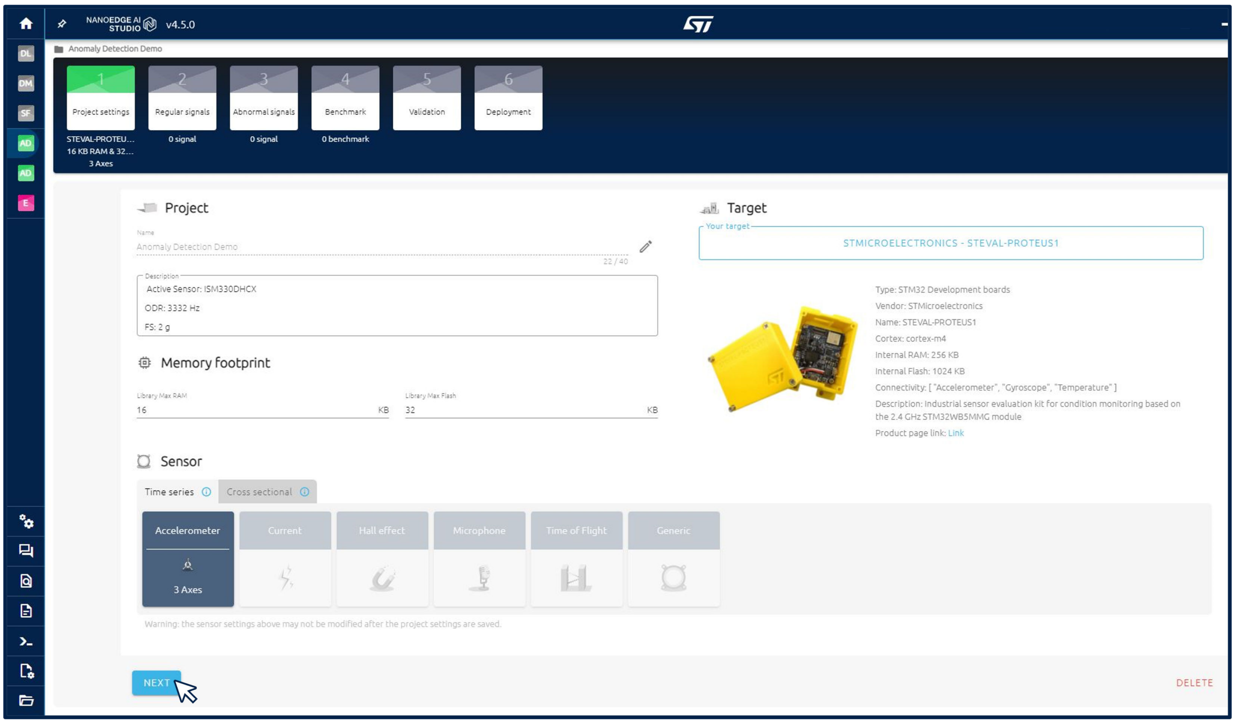
Figure 28. **Six-step workflow**



4.2.1.1 Project settings (AD)

1. Enter the project name and description.
2. Select the STEVAL-PROTEUS from the list provided in the drop-down menu under *Target*.
3. Enter the maximum amount of RAM to be allocated for the library. Usually, a few Kbytes is enough (but it depends on the data frame length used in the process of data preparation; 16 Kbytes is a good starting point).
4. Enter the maximum amount of FLASH to be allocated for the library. Usually, a few Kbytes is enough (but it depends on the data frame length used in the process of data preparation; 32 Kbytes is a good starting point).
5. Select the sensor type, that is the 3-axis accelerometer from the list in the drop-down menu.

Figure 29. **Project settings (AD)**



4.2.1.2 *Insert the labeled data: regular and abnormal signals*

To insert the labeled dataset, follow the procedure below.

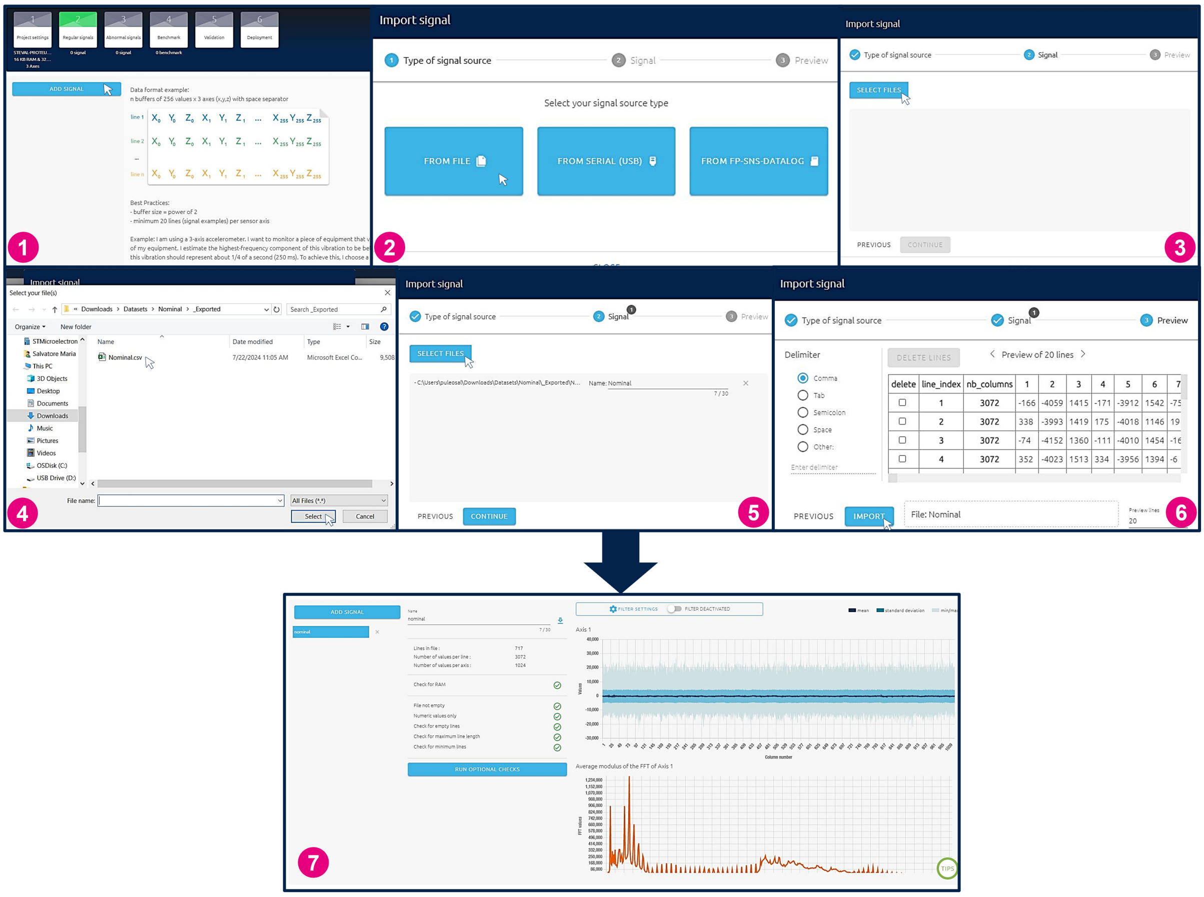
1. Start the data samples selection, for "Regular" class, pushing the "ADD SIGNAL" button.
2. Select the signal source.
   Even if three options are available, we strongly recommend to use the first one, by selecting "FROM FILE" option.
3. Open the dialog panel to select the directory including the files *.csv to use.
4. Select the folder generated in the Convert the dataset in NanoEdgeAI format, with dataset previously classified: *./My_Dataset/ filename.csv*.

Note: *Assign a class label to the imported data in the Name field once the data is imported. This label is used to differentiate the classes and is used in the emulation to show the predicted class.*

5. Verify that the "Comma" delimiter is asserted and then click on the IMPORT button.
6. Check the data import in the last panel, where are exposed the time domain and frequency domain plot, plus some filter settings.

Repeat all the steps to import the dataset for abnormal class.

Figure 30. Classified data import

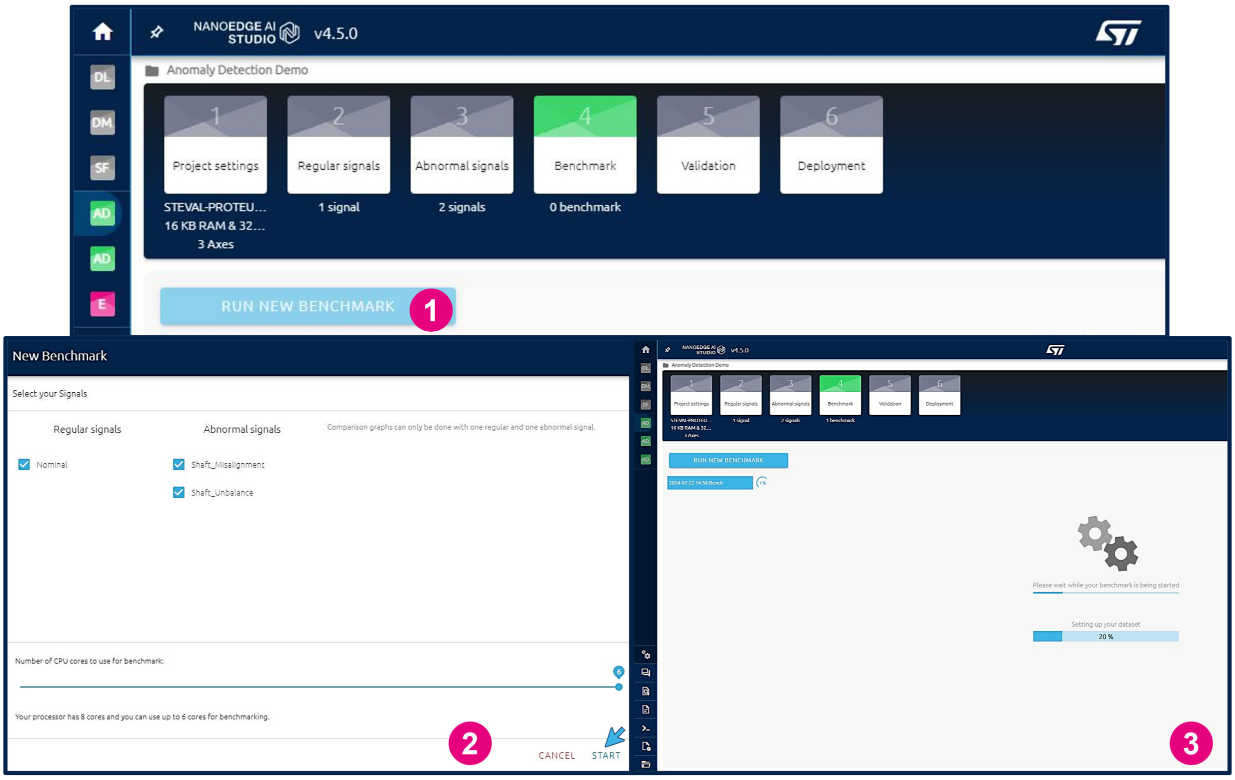


4.2.1.3 *Run the benchmark to create the NEAI AD libraries*

Benchmark the available models and choose the one that complies with the requirements and provides the best performance.

- Click on Run New Benchmark.
- A new window appears that allows you to select input files (signal examples) to use, and also to change the number of CPU cores to use.
- Start and wait until you see the perspective shown below.

Figure 31. **Start benchmark (AD)**



This perspective allows you to monitor some indicators like:

- Score
- Balanced accuracy
- RAM
- Flash memory

Keep in mind that:

- Benchmarks may take a long time (several hours) to complete and find a fully optimized library
- Benchmarks can be paused/resumed, or stopped at any time, without canceling the process (the best library found is not lost)
- Benchmark progress in percentage is displayed on the left side of the screen, next to the name/ID of the benchmark, in the benchmark list under the RUN NEW BENCHMARK button.

Figure 32. **Benchmark AD execution (1 of 2)**

After some hours, you should obtain a result similar to the one of the figures below.

Figure 33. Benchmark AD execution (2 of 2)

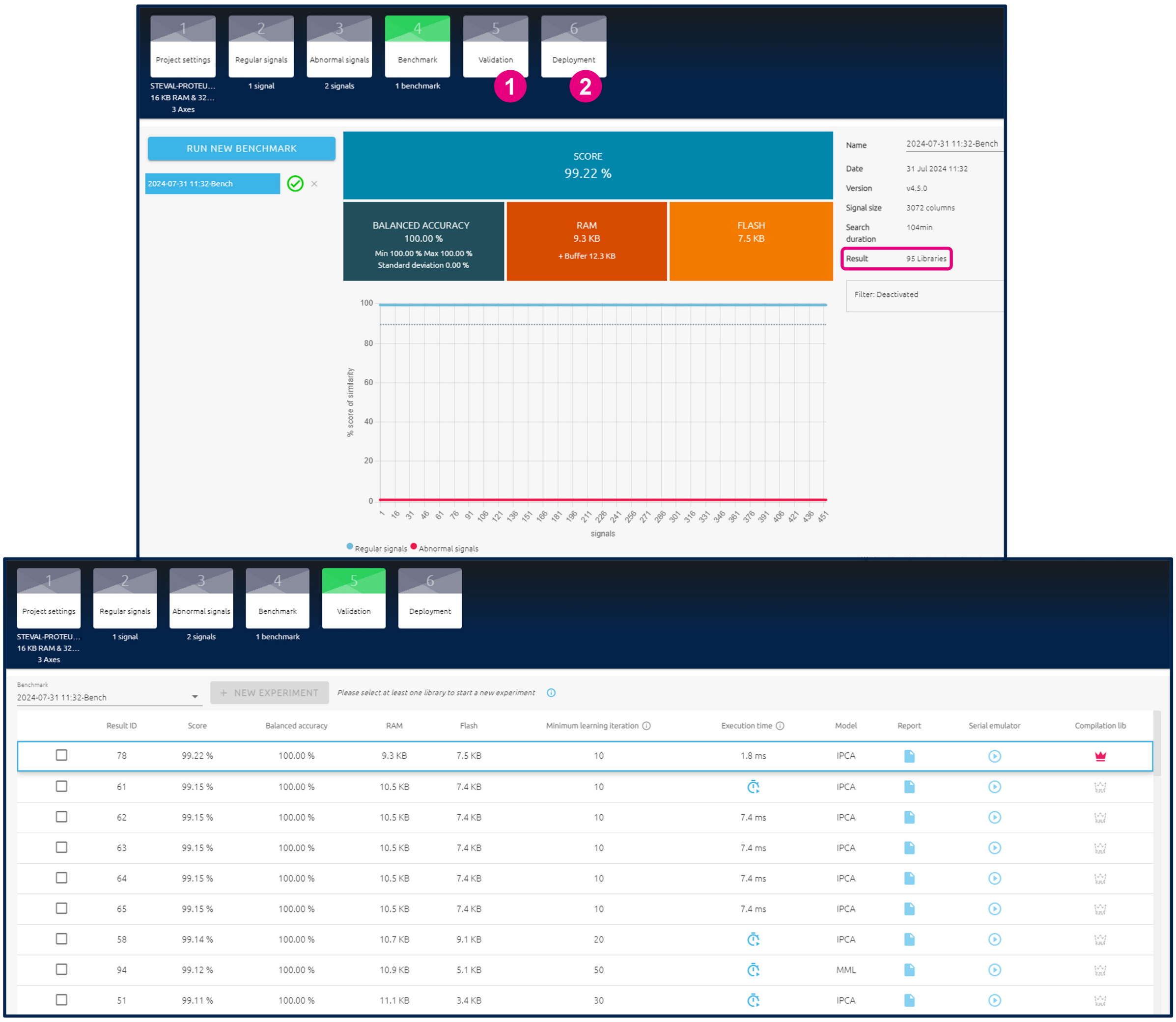
You can choose whether to continue searching an improved library or pause/stop the benchmark to see the results.

4.2.1.4 *NEAI AD libraries evaluation*

Now the user can proceed in two different ways:

- Start with a further models evaluation, pushing on **Validation** button. The tool shows a full list of the libraries evaluated, and allows to validate one or more libraries in the same time, selecting the libraries more suitable with your own final application.
- Accept the best library proposed by the tool, pushing on **Deployment** button, and follow the step described in the next paragraph.

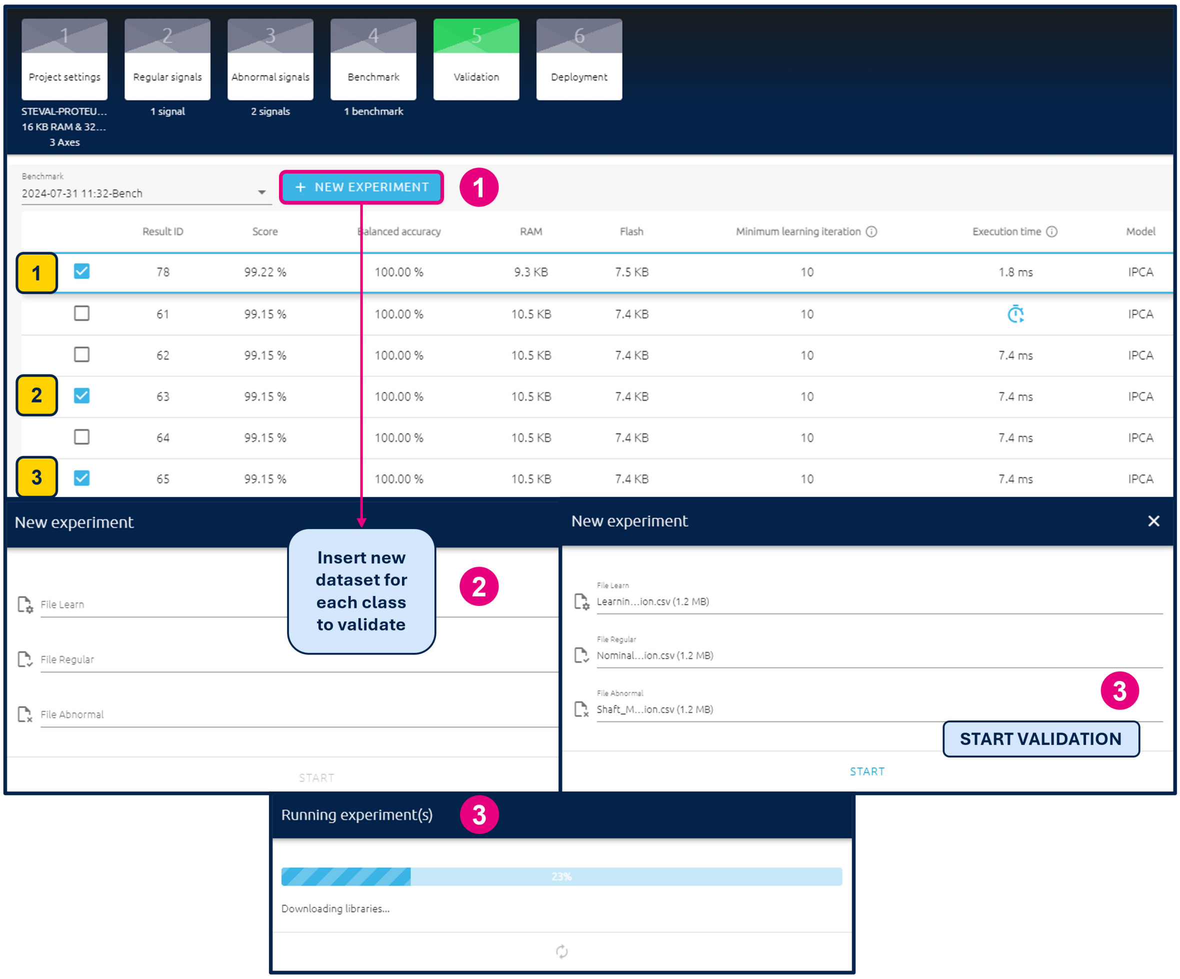
Figure 34. **Benchmark results and models summary (AD)**



In this page is possible to compare the several library models generated by benchmark, in terms of performance index, RAM and Flash occupancy, execution time and model. To validate and testing one or more libraries, the user must proceed as follow:

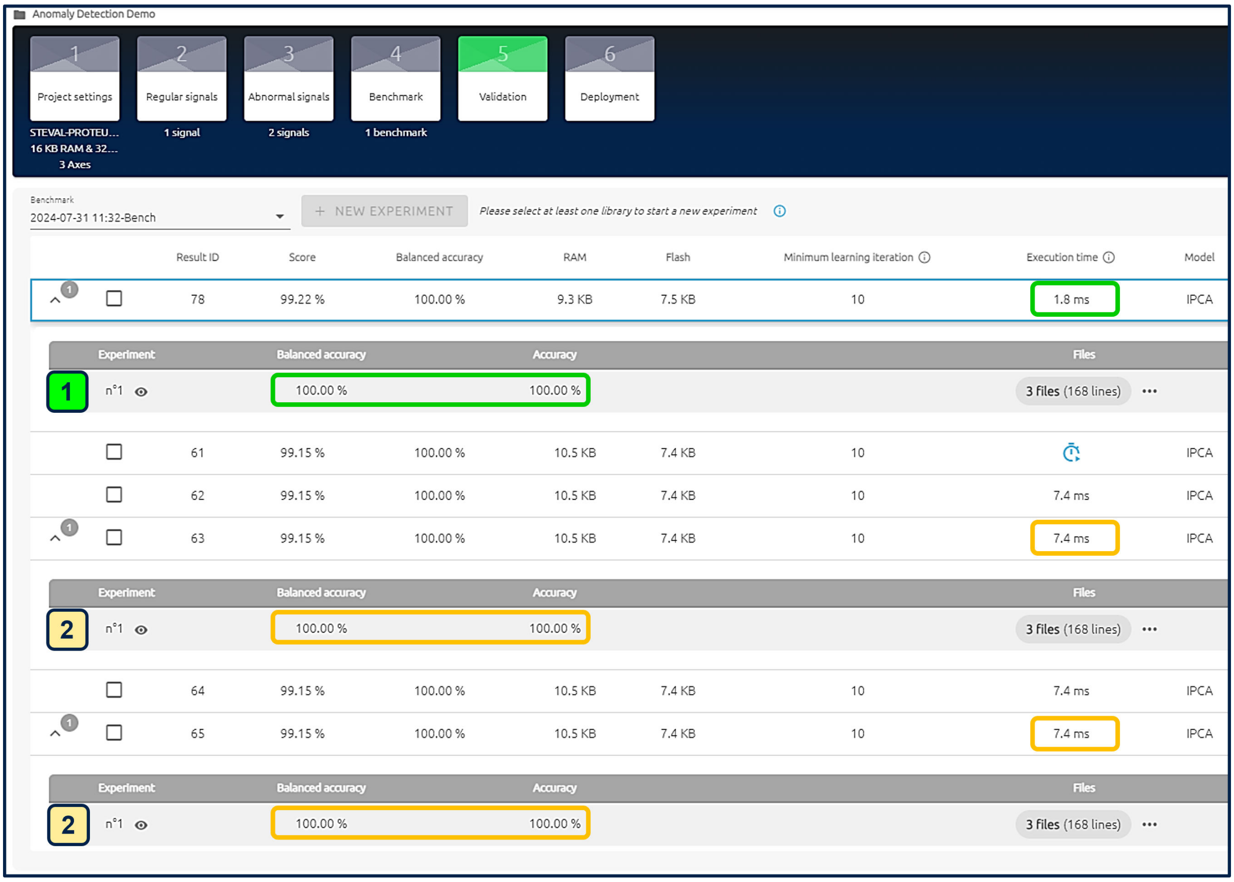
1. Select at least one library (after the first selection the button **NEW EXPERIMENT** is activated).
2. After that select the dataset acquired during the datalogging phase, for each classes condition to evaluate, and different from the data used for the benchmarking.
3. Run the evaluation experiment.

Figure 35. Models validation (AD)



4. Evaluate and compare the model performances. For each library model the **Balanced Accuracy** and **Accuracy** related to validation dataset used is provided, and the results could be different from the ranking outcomes from the benchmark.

Figure 36. Validation results (AD)



In the example shown above, the validation is performed between the models in the **1st**, **4th**and **6th**ranking position. After running the validation, the performance accuracy index has reached the maximum (100%) for all the libraries selected, but considering the execution time (*1.8 ms*) the best choice is the **1st**. Before proceeding with the final deployment the user must select this library pushing on the relates crown on the right side.

4.2.1.5 *NEAI AD library deployment*

The last step for the NanoEdgeAI tool is to compile and release the libraries in a unique ZIP file containing the library model in compiled format, plus other files to complete the integration.

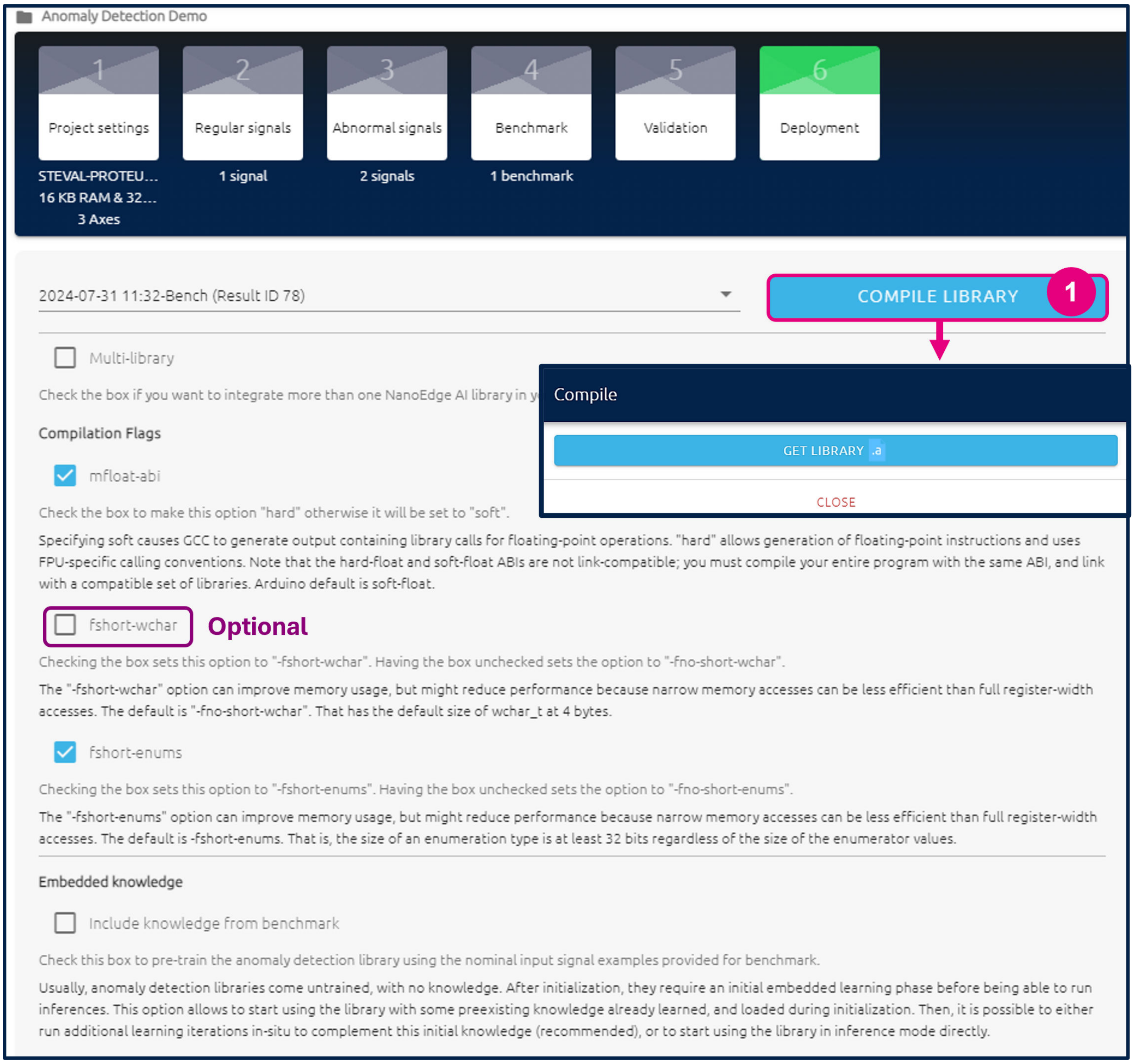
Pushing the last right button **Deployment,** to generate all the libraries files embedding the library models.

In case of classification models:

1. Take care to check the **fshort-wchar** option if the library is integrated in a KEIL® project and leave the other flags in their default state.
2. Push the **Compile Library** button.

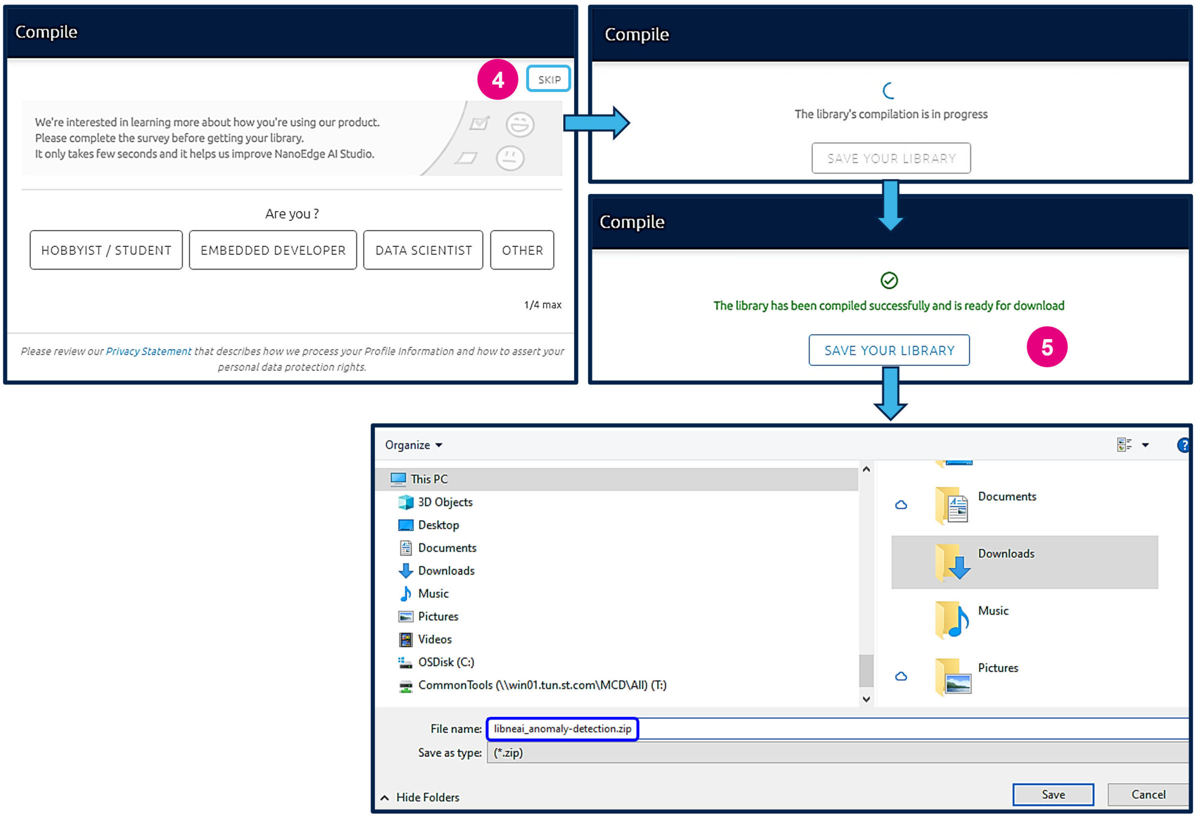
3. Push the **Get Library .a** button.

Figure 37. **Deployment setting (AD)**



4. Complete the survey or select **SKIP** option in the window to start the compilation and ZIP file creation.
5. Push on **SAVE YOUR LIBRARY** to store the ZIP file.

Figure 38. **Library compilation and ZIP file saving (AD)**

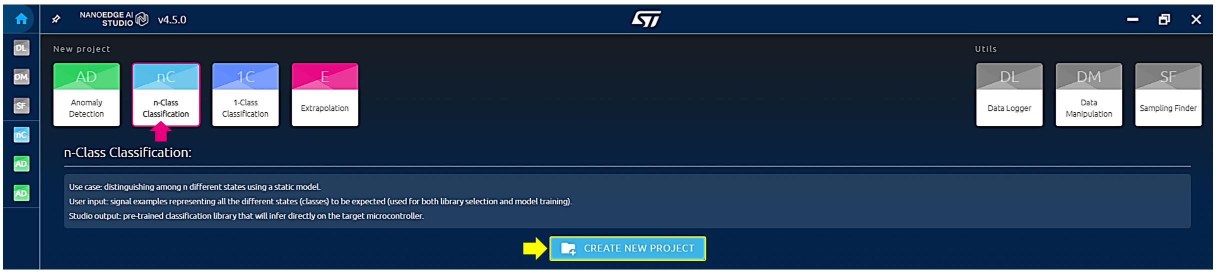


For further details about NanoEdgeAI Studio, see NanoEdge AI Studio Documentation.

4.2.2 NEAI N-Class Classification library

To generate the classification library using the NanoEdgeAI tool, you need to start by creating the new project using the **N-Class Classification** button as shown in the following figure.

Figure 39. **Start an N-Class Classification project**



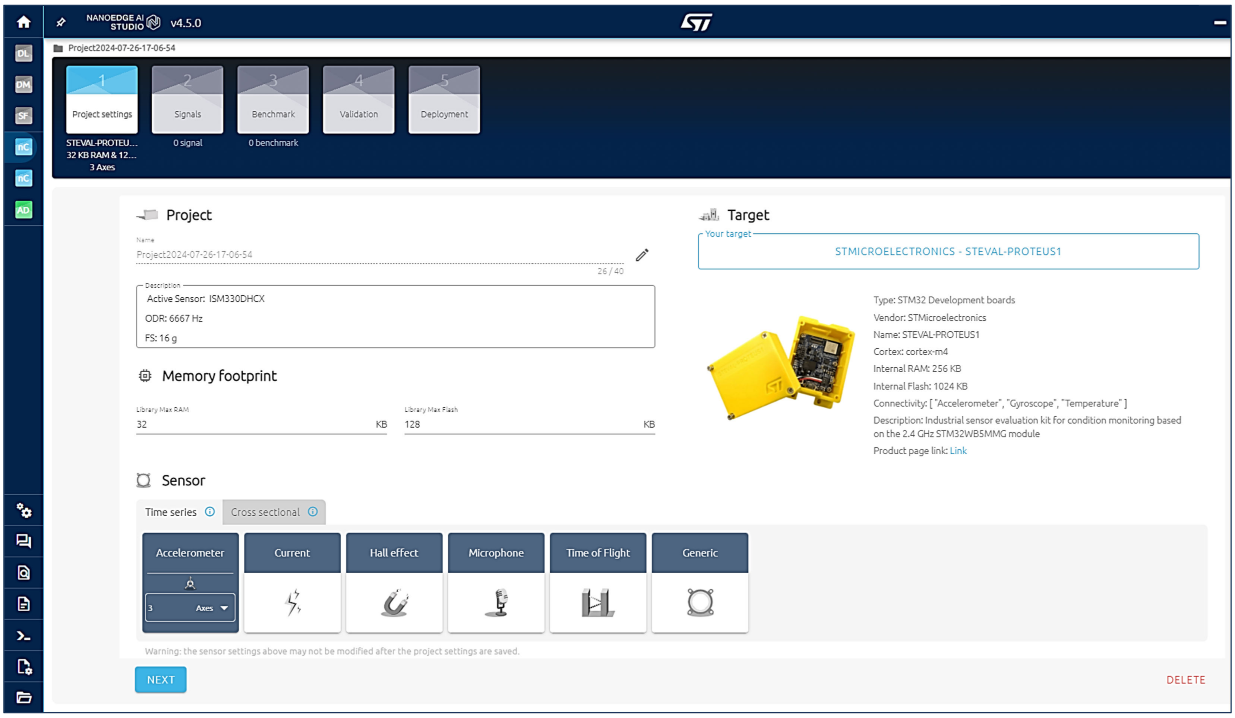
The text shows the information on what the classification is good for. You can create the project by clicking on the CREATE NEW PROJECT button. The process to generate the N-Class Classification library consists of five steps as shown below.

Figure 40. **Five-step workflow**

4.2.2.1 *Project settings (NCC)*

1. Enter the project name and description.
2. Select the STEVAL-PROTEUS from the list provided in the drop-down menu under Target.
3. Enter the maximum amount of RAM to be allocated for the library. Usually, a few Kbytes is enough (but it can depends on the data frame length used in the process of data preparation; 16 Kbytes is a good starting point).
4. Enter the maximum amount of FLASH to be allocated for the library. Usually, a hundred Kbytes is enough (but it depends on the data frame length used in the process of data preparation; 128 Kbytes is a good starting point).
5. Select the sensor type, that is the 3-axis accelerometer from the list in the drop-down menu.

Figure 41. **Project settings (NCC)**



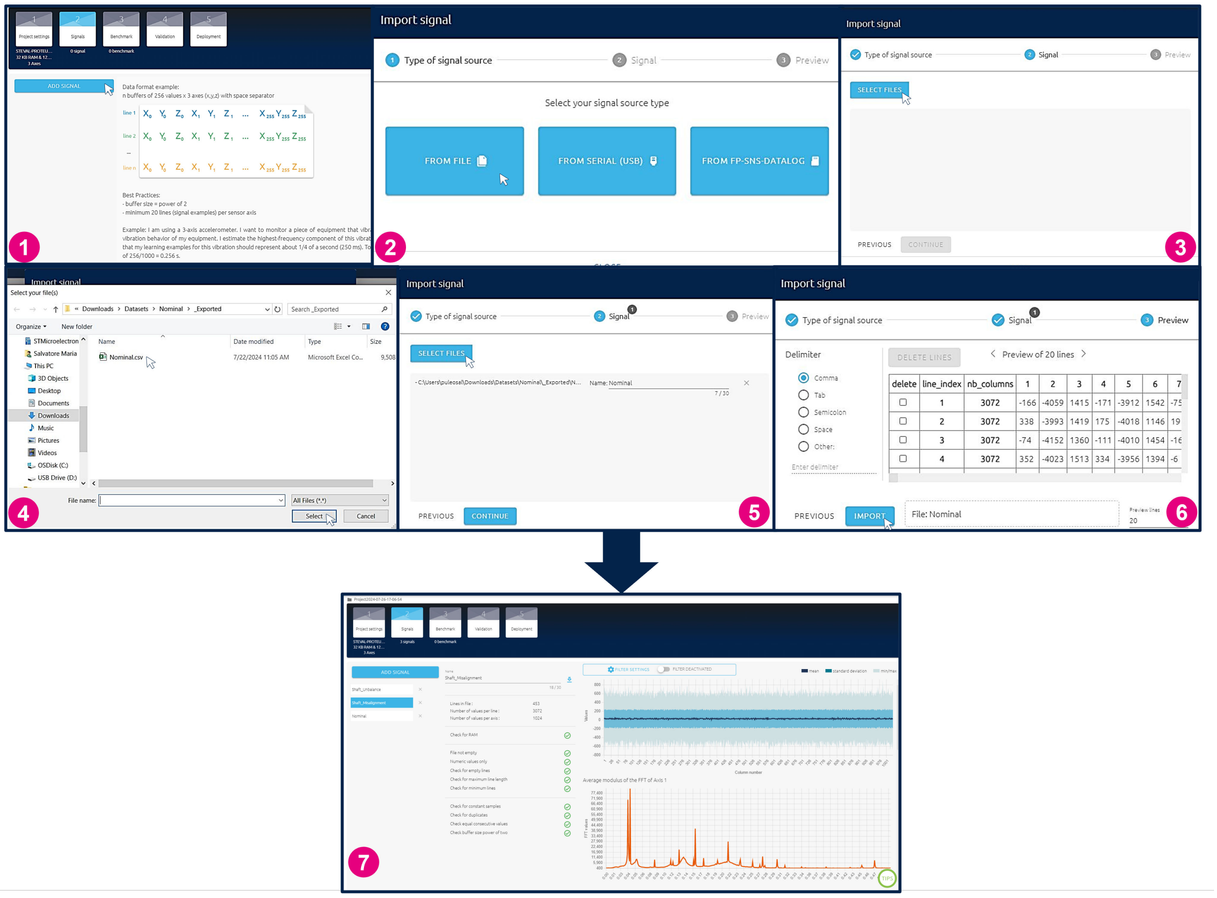
4.2.2.2 *Insert the classified data*

To insert the classified dataset, follow the procedure below.

1. Start the data samples selection pushing the “ADD SIGNAL” button.

2. Select the signal source. Even if three options are available, we strongly recommend using the first one, by selecting the "FROM FILE" option.
3. Open the dialog panel to select the directory including the files *.csv to use.
4. Select the folder generated in the Convert the dataset in NanoEdgeAI format, with dataset previously converted: *./My_Dataset/filename.csv*.
5. Verify that the "Comma" delimiter is asserted and then click on the IMPORT button.
6. Check the data import in the last panel, where are exposed the time domain and frequency domain plot, plus some filter settings.
7. Repeat all the steps to import the dataset for other classes.

Figure 42. NCC classified data import

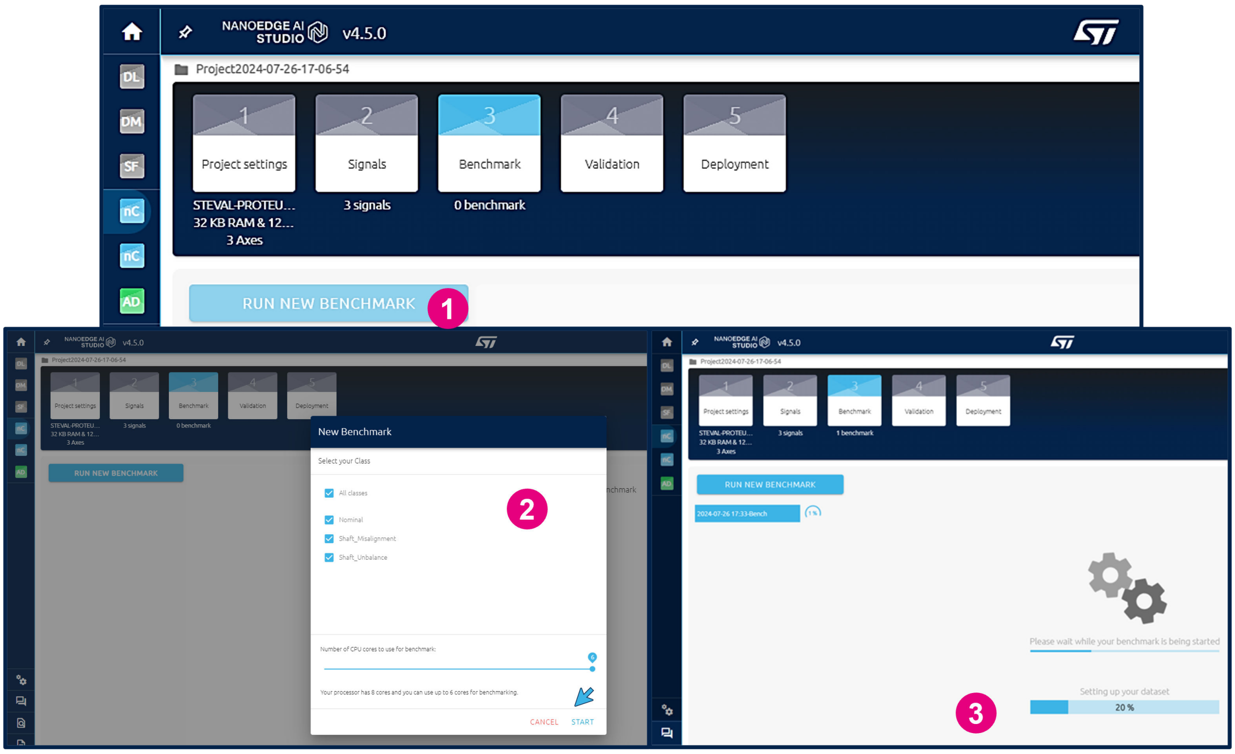


4.2.2.3 *Run the benchmark to create the NEAI N-Class Classification libraries*

Benchmark the available models and choose the one that complies with the requirements and provides the best performance.

- Click on Run New Benchmark.
- A new window appears that allows you to select input files (signal examples) to use, and also to change the number of CPU cores to use.
- Start and wait until you see the perspective shown below.

Figure 43. Start benchmark (NCC)



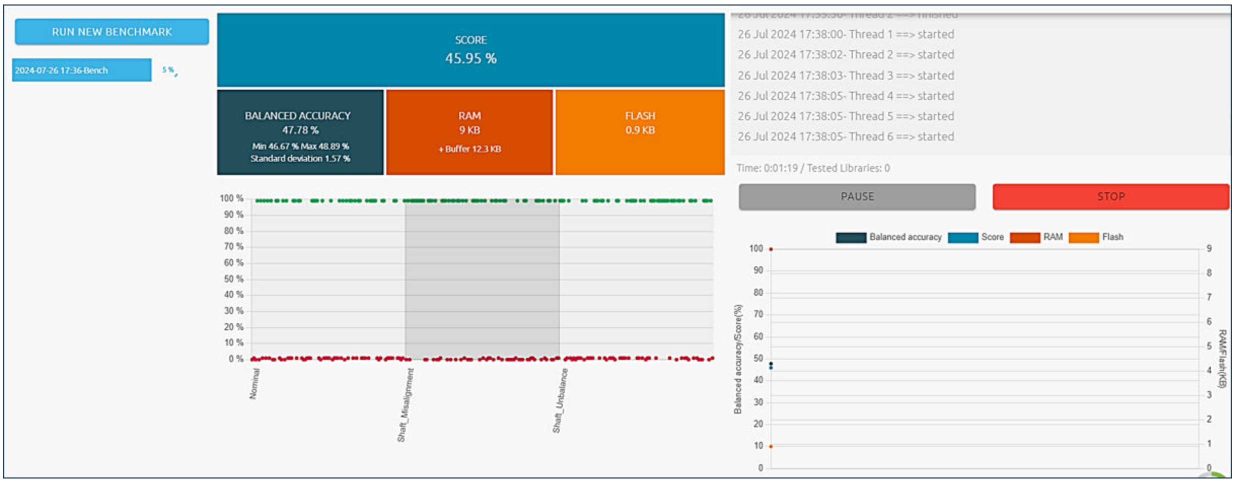
This perspective allows you to monitor some indicators like:

- Score
- Balanced accuracy
- RAM
- Flash memory

Keep in mind that:

- Benchmarks may take a long time (several hours) to complete and find a fully optimized library
- Benchmarks can be paused/resumed, or stopped at any time, without canceling the process (the best library found is not lost)
- Benchmark progress in percentage is displayed on the left side of the screen, next to the name/ID of the benchmark, in the benchmark list under the RUN NEW BENCHMARK button.

Figure 44. Benchmark NCC execution (1 of 2)



You can choose whether to continue searching an improved library or pause/stop the benchmark to see the results. After some hours (5 hours for this example), you should obtain a result similar to the one of the figures below.

.

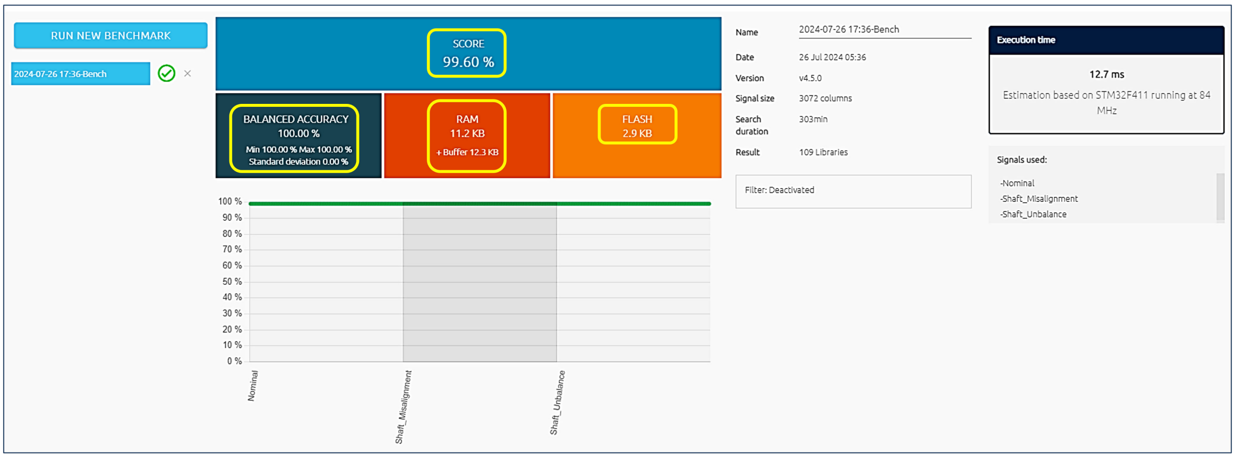


Figure 45. **Benchmark NCC execution (2 of 2)**

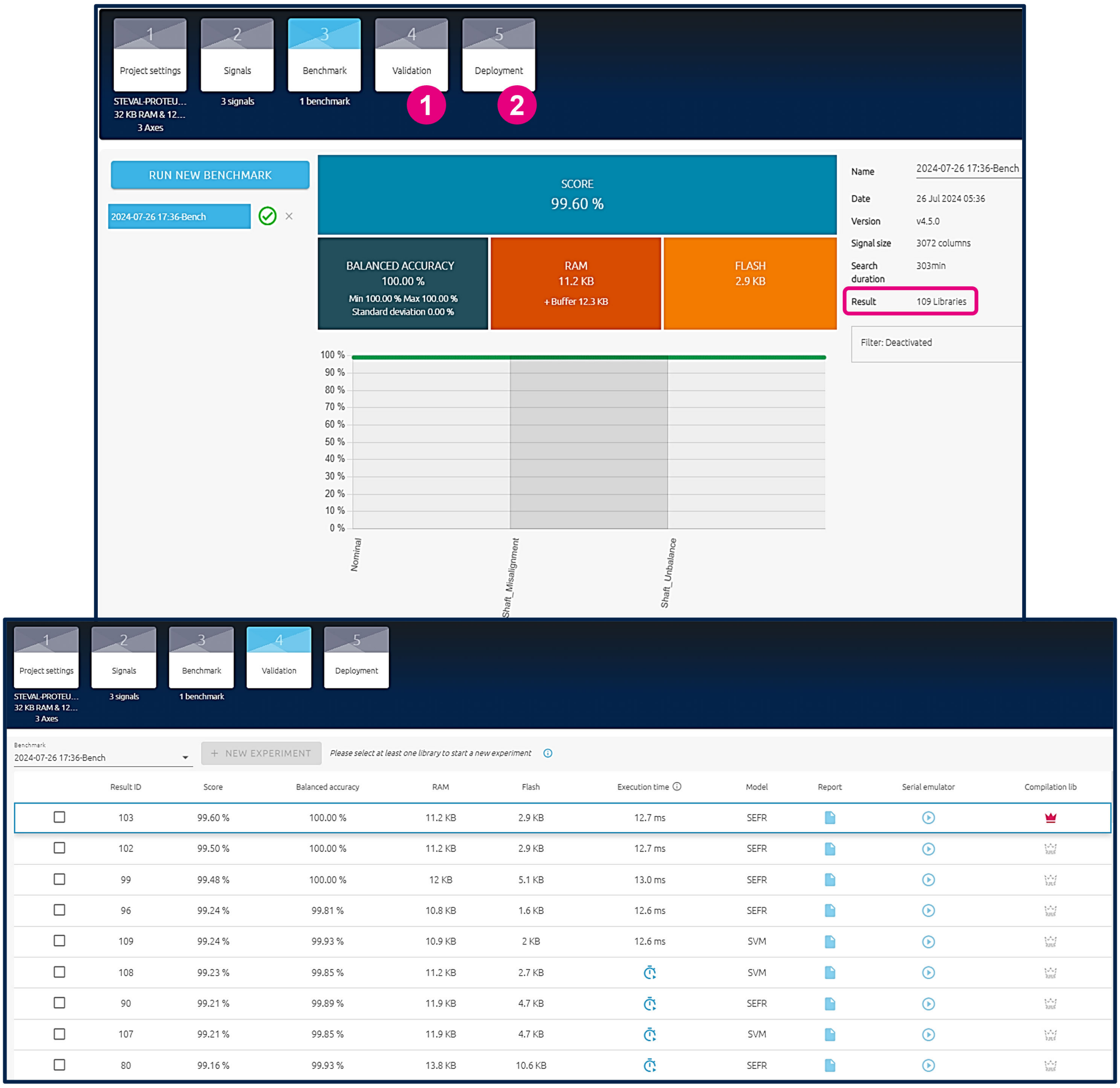
Where the library model more suitable for the tool is summarized, in terms of indicators detailed before.

4.2.2.4 *NEAI N-Class Classification libraries evaluation*

Now the user can proceed in two different ways:

- Start with a further models evaluation, pushing on **Validation** button. The tool shows a full list of the libraries evaluated, and allows to validate one or more libraries in the same time, selecting the libraries more suitable with your own final application.
- Accept the best library proposed by the tool, pushing on **Deployment** button, and follow the step described in the next paragraph.

Figure 46. **Benchmark results and models summary (NCC)**



Result ID	Score	Balanced accuracy	RAM	Flash	Execution time	Model
103	99.60 %	100.00 %	11.2 KB	2.9 KB	12.7 ms	SEFR
102	99.50 %	100.00 %	11.2 KB	2.9 KB	12.7 ms	SEFR
99	99.48 %	100.00 %	12 KB	5.1 KB	13.0 ms	SEFR
96	99.24 %	99.81 %	10.8 KB	1.6 KB	12.6 ms	SEFR
109	99.24 %	99.93 %	10.9 KB	2 KB	12.6 ms	SVM
108	99.23 %	99.85 %	11.2 KB	2.7 KB		SVM
90	99.21 %	99.89 %	11.9 KB	4.7 KB		SEFR
107	99.21 %	99.85 %	11.9 KB	4.7 KB		SVM
80	99.16 %	99.93 %	13.8 KB	10.6 KB		SEFR

In this page is possible to compare the several library models generated by benchmark, in terms of performance index, RAM and Flash occupancy, execution time and model. To validate and testing one or more libraries, the user must proceed as follow:

1. Select at least one or more libraries (after the first one selection the button **NEW EXPERIMENT** is activated).
2. After that select the dataset acquired during the datalogging phase, for each classes condition to evaluate, and different from the data used for the benchmarking.
3. Run the evaluation experiment.

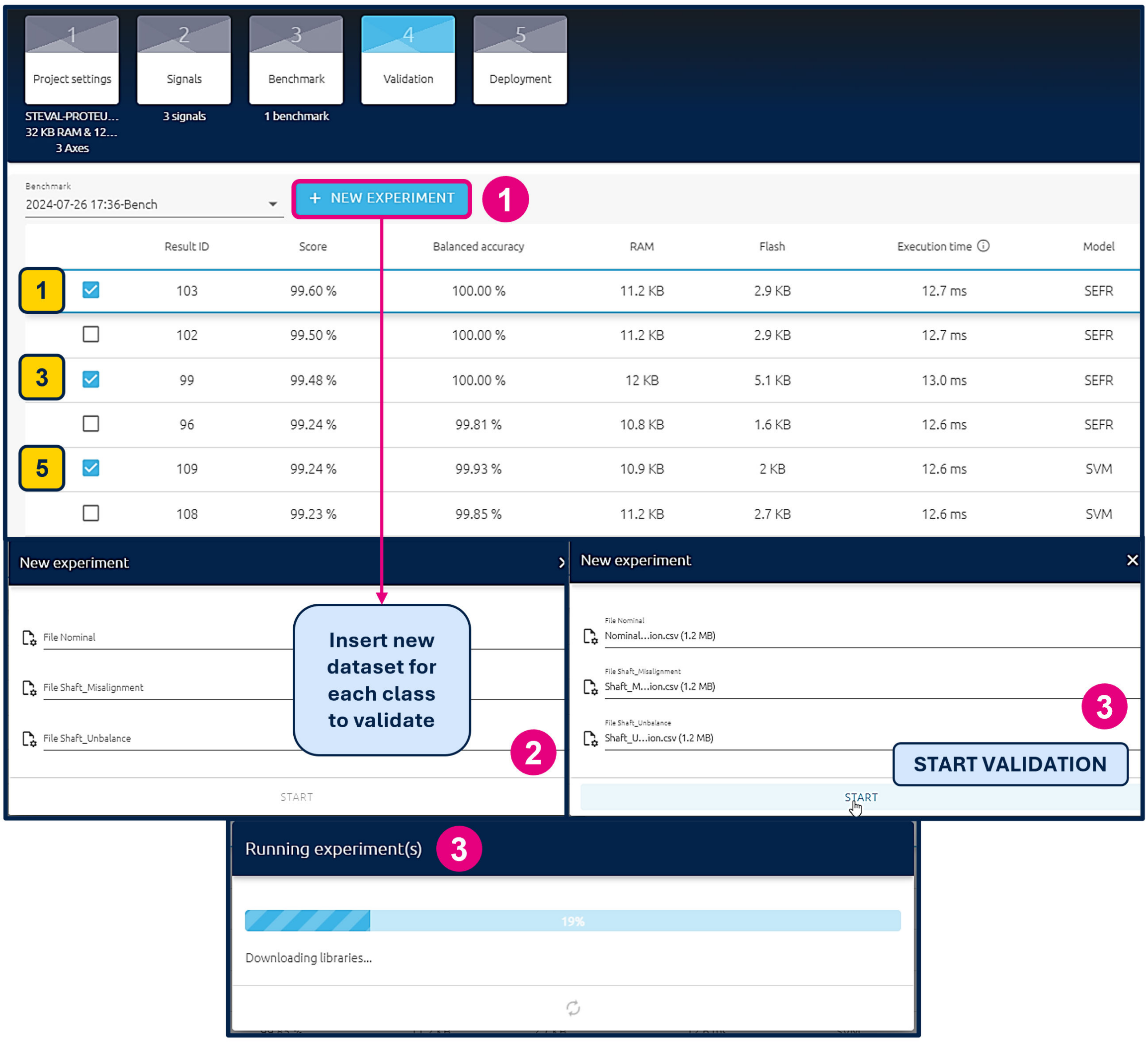
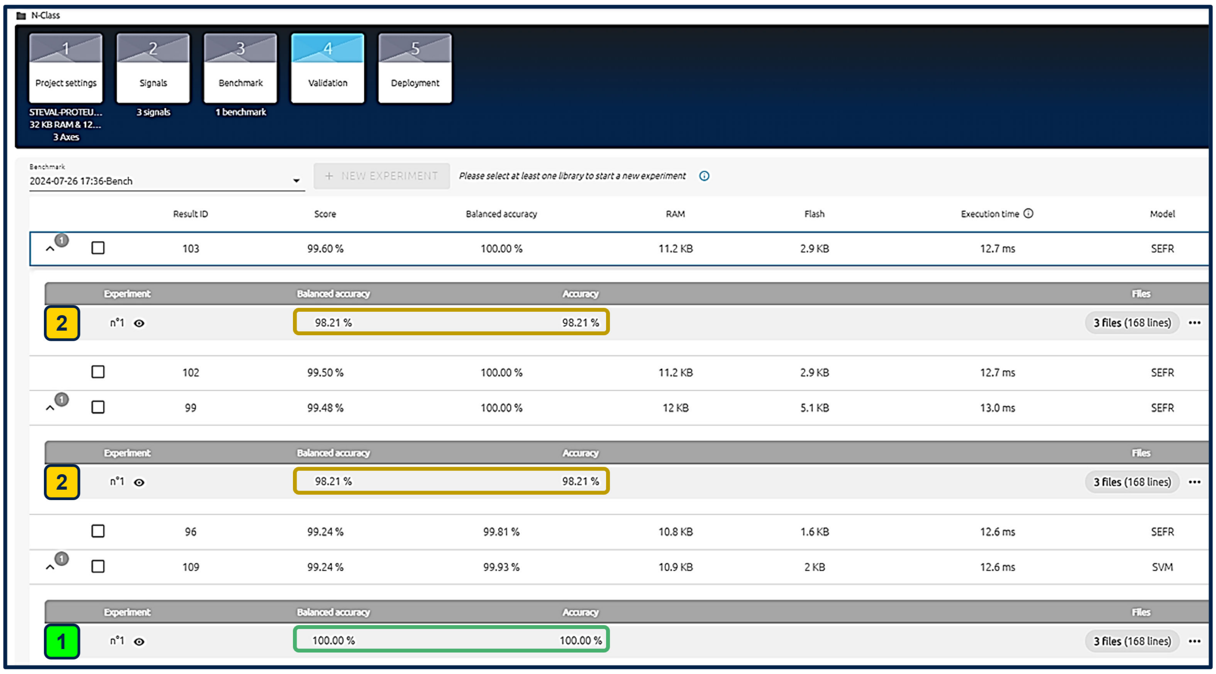


Figure 47. **Models validation (NCC)**

4. Evaluate and compare the model performances.
   For each library model the **Balanced Accuracy** and **Accuracy** related to validation dataset used is provided, and the results could be different from the ranking outcomes from the benchmark.

Figure 48. Validation results (NCC)



In the example shown above, the validation is performed between the models in the **1**st**, 3**rdand **5**th ranking position but, after running the validation, the performance accuracy index is changed and the best is the **5**th. So, using the validation procedure the user was able to find out that the 5th library is the most accurated one.

Before proceeding with the final deployment, the user must select this library pushing on the relates crown on the right side

4.2.2.5 *NEAI N-Class library deployment*

The last step for the NanoEdgeAI tool is to compile and release the libraries in a unique ZIP file containing the library model in compiled format, plus other files to complete the integration.

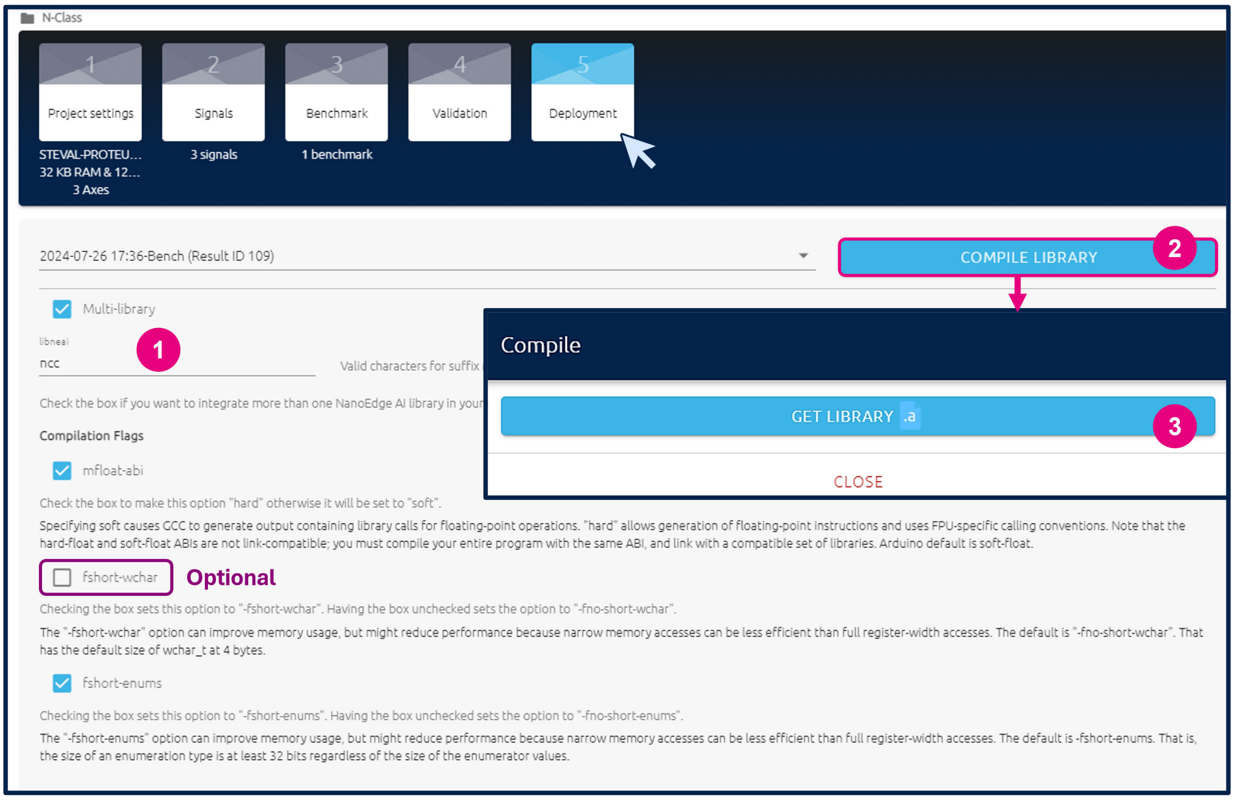
Pushing the last right button **Deployment,** to generate all the libraries files embedding the library models.

In case of classification models:

1. Set the **Multi-library** flag, to enable the suffix line insertion, and insert “**ncc**” just to identify these classification files from others to insert in the application. Take care to check the **fshort-wchar** option if the library is integrated in a KEIL® project and leave the other flags in their default state.
2. Push the **Compile Library** button.

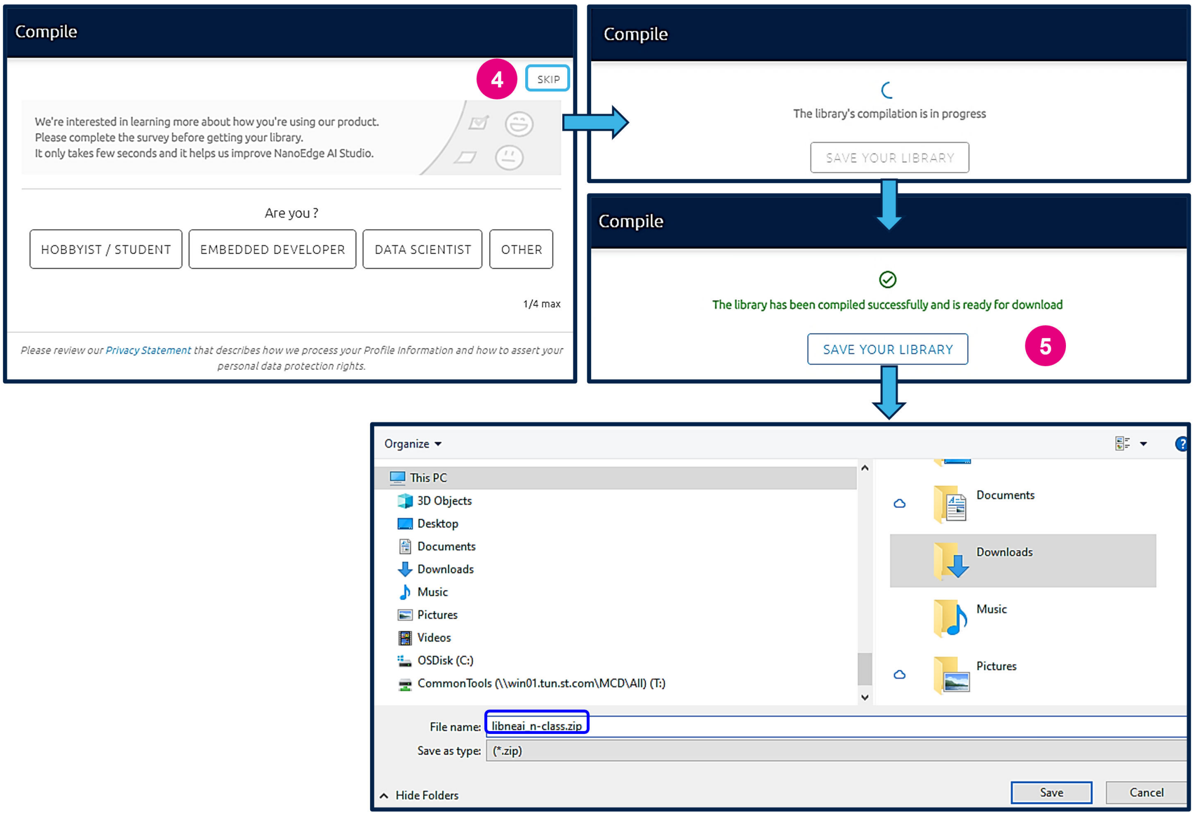
3. Push the **Get Library .a** button.

Figure 49. Deployment setting (NCC)



4. Complete the survey or select **SKIP** option in the window to start the compilation and ZIP file creation.
5. Push on **SAVE YOUR LIBRARY** to store the ZIP file.

Figure 50. **Library compilation and ZIP file saving (NCC)**

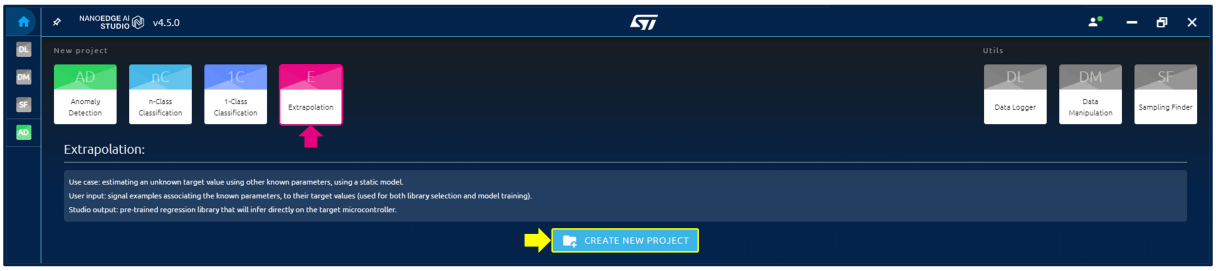


For further details about NanoEdgeAI Studio, see NanoEdge AI Studio Documentation.

4.2.3 NEAI E extrapolation library

To generate the extrapolation library using the NanoEdgeAI tool, you need to start by creating the new project using the E button as shown in the following figure.

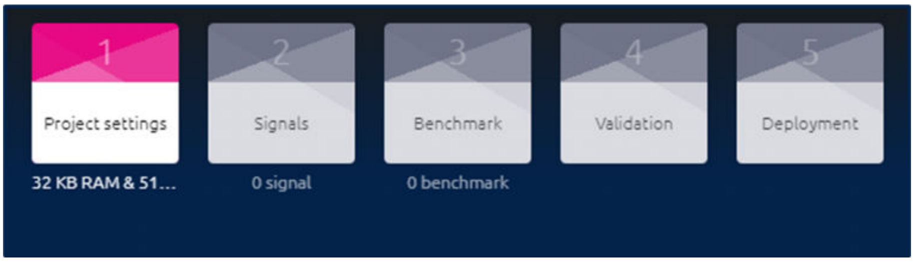
Figure 51. **Start an extrapolation project**



The text shows the information on what the extrapolation is good for. The project can be created by clicking on the **CREATE NEW PROJECT** button.

The process to generate the extrapolation library consists of five steps as shown below.

Figure 52. **Five-step workflow (E)**

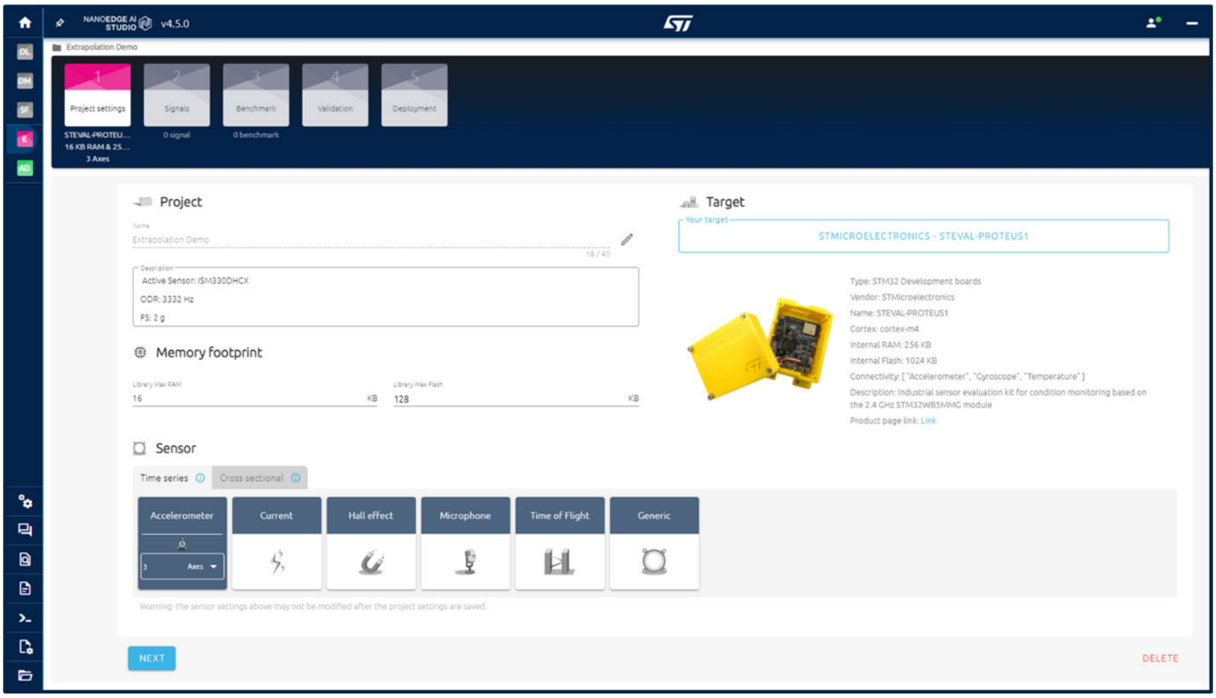


4.2.3.1 *Project settings (E)*

Project settings

1. Enter the project name and description.
2. Select the STEVAL-PROTEUS from the list provided in the drop-down menu under *Target*.
3. Enter the maximum amount of RAM to be allocated for the library. Usually, a few Kbytes is enough (but it depends on the data frame length used in the process of data preparation; 16 Kbytes is a good starting point).
4. Enter the maximum amount of FLASH to be allocated for the library. Usually, a hundred Kbytes is enough (but it depends on the data frame length used in the process of data preparation; 128 Kbytes is a good starting point).
5. Select the sensor type, that is the 3-axis accelerometer from the list in the drop-down menu.

Figure 53. **Project setting (E)**



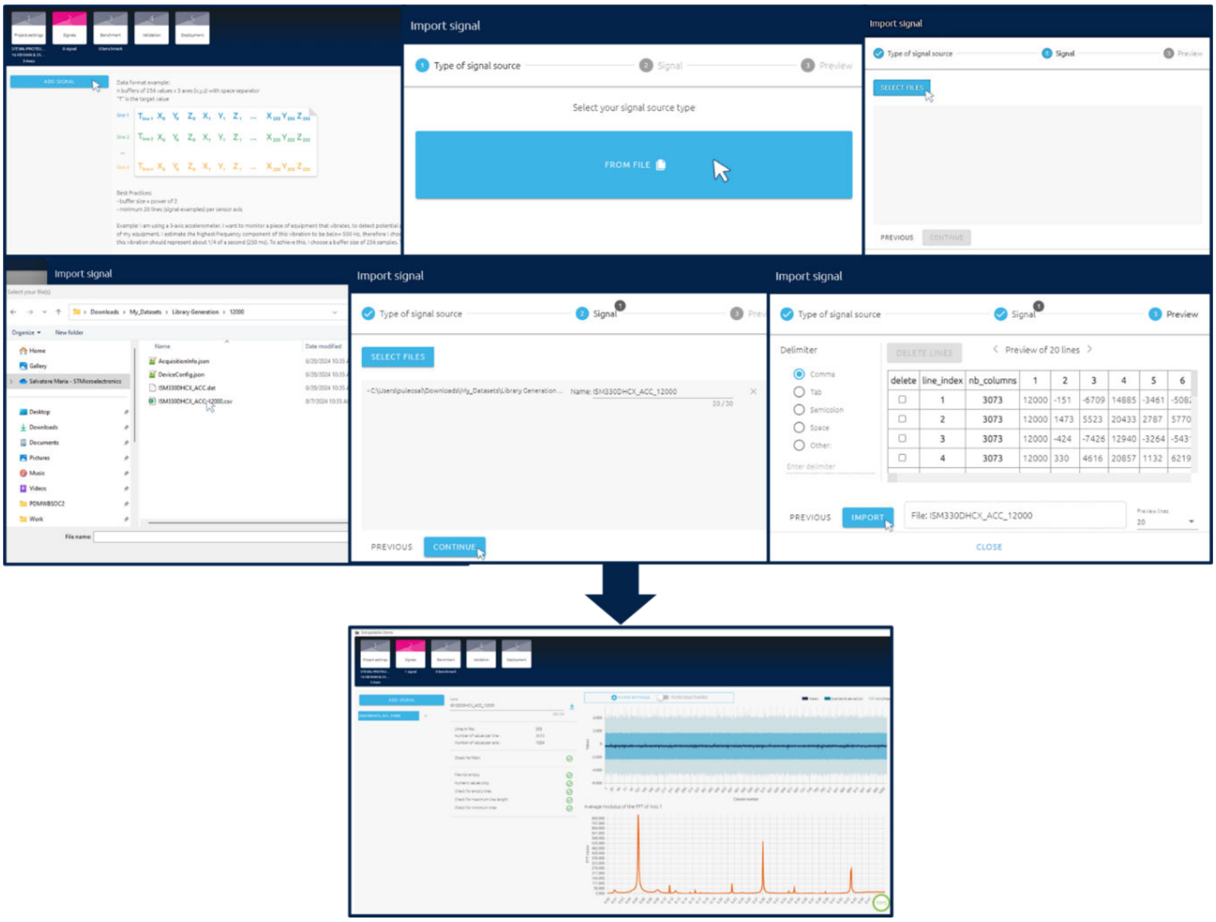
4.2.3.2 *Insert the labeled data (E)*

To insert the labeled dataset, follow the procedure below.

1. Start the dataset selection, for your first class, pushing the “ADD SIGNAL” button.
2. Select the signal source. Even if three options are available, it is strongly recommend to use the first one, by selecting the “FROM FILE” option.
3. Open the dialog panel to select the directory including the files *.csv to use.
4. Select the folder generated in Section 4.1.2, with the dataset previously converted: *./My_Dataset/filename.csv*.
5. Verify that the “Comma” delimiter is asserted and then click on the IMPORT button.
6. Check the data import in the last panel, where the time domain and frequency domain plot are exposed, plus some filter settings.

7. Repeat all the steps to import all your datasets.

Figure 54. **E labeled data import**

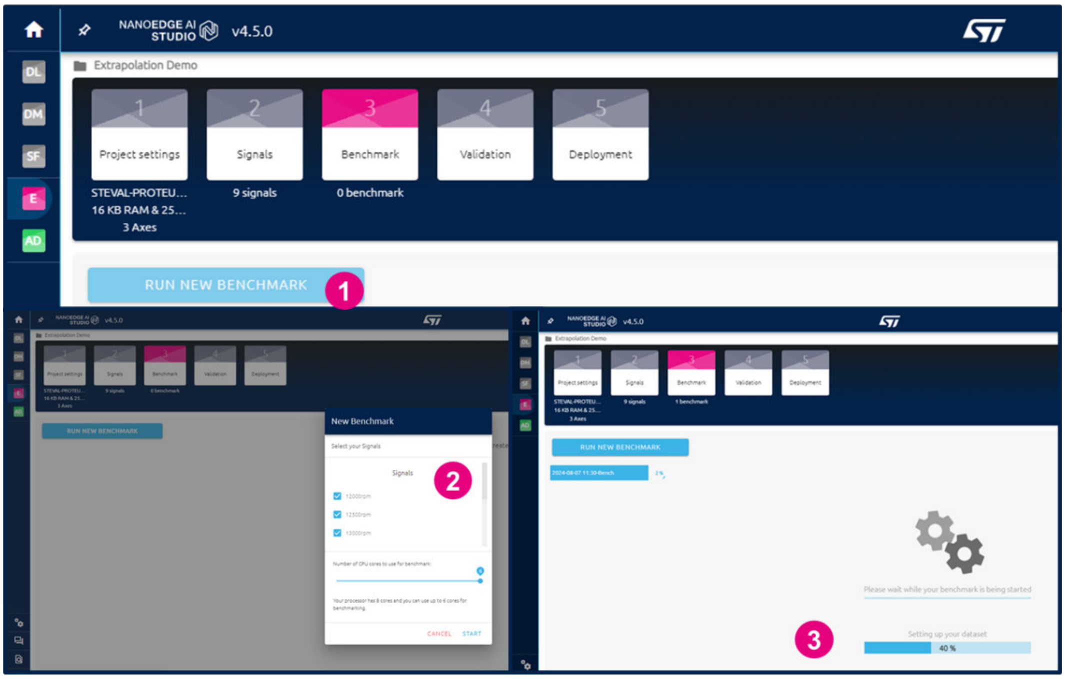


4.2.3.3 *Run the benchmark to create the NEAI E extrapolation library*

Benchmark the available models and choose the one that complies with the requirements and provides the best performance.

- Click on *Run New Benchmark*.
- A new window appears that allows you to select input files (signal examples) to use, and to change the number of CPU cores to use.
- Start and wait until you see the perspective shown below.

Figure 55. **Start benchmark (E)**



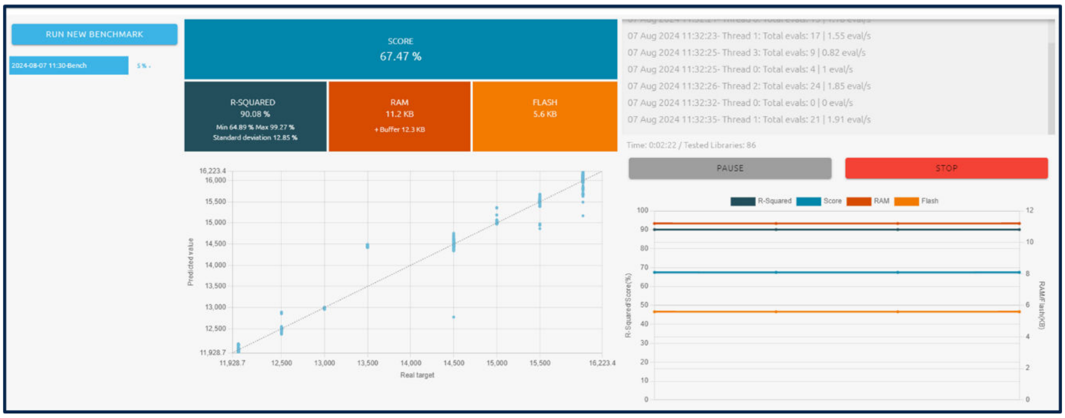
This perspective allows you to monitor some indicators like:

- Score
- R-Squared
- RAM
- Flash memory

Keep in mind that:

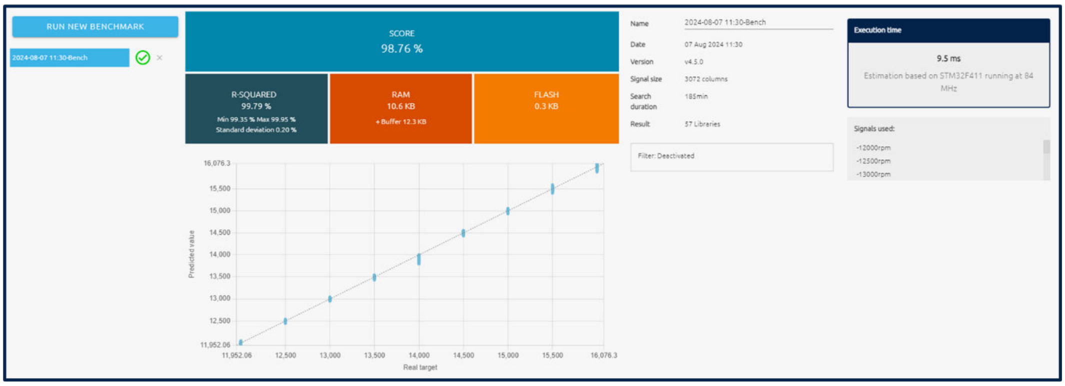
- Benchmarks may take a long time (several hours) to complete and find a fully optimized library.
- Benchmarks can be paused/resumed, or stopped at any time, without canceling the process (the best library found is not lost).
- Benchmark progress in percentage is displayed on the left side of the screen, next to the name/ID of the benchmark, in the benchmark list under the RUN NEW BENCHMARK button.

Figure 56. Benchmark E execution (1 of 2)



You can choose whether to continue searching for an improved library or pause/stop the benchmark to see the results. After some hours (3 hours for this example), you should obtain a result similar to the one in the figure below.

Figure 57. Benchmark E execution (2 of 2)



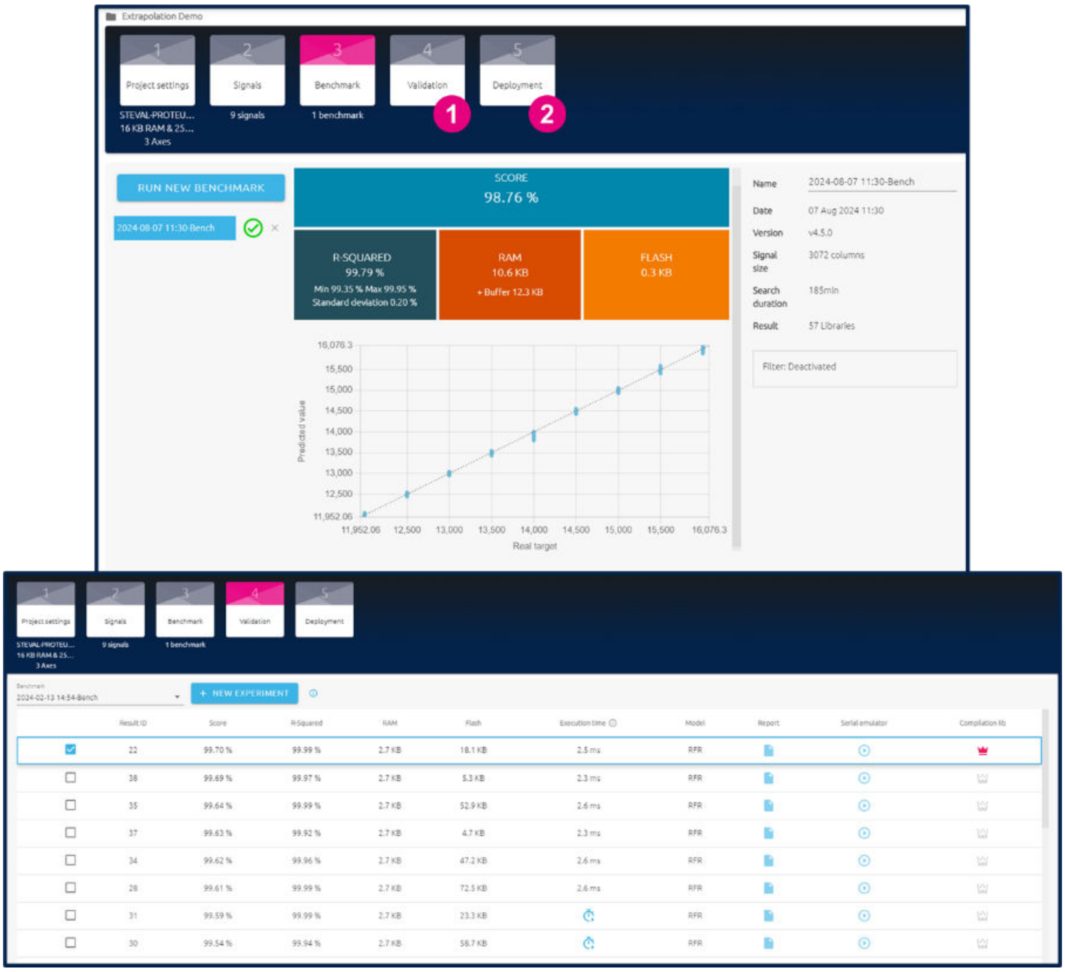
Where the library model more suitable for the tool is summarized, in terms of the indicators detailed before.

4.2.3.4 *NEAI E libraries evaluation*

The user can now proceed in two different ways:

1. Start with a further models' evaluation, pushing on the **Validation** button.
   The tool shows a full list of the libraries evaluated, and allows to validate one or more libraries at the same time, selecting the libraries more suitable to your own final application.
2. Accept the best library proposed by the tool, pushing on the **Deployment** button, and follow the steps described in the next paragraph.

Figure 58. **Benchmark results and models summary (E)**



In this page it is possible to compare the several library models generated by benchmark, in terms of performance index, RAM, and Flash occupancy, execution time and model. To validate and test one or more libraries, the user must proceed as follows:

1. Select at least one or more libraries (after the first one is selected the button NEW EXPERIMENT is activated).
2. After that select the dataset acquired during the datalogging phase, for each class condition to evaluate, and different from the data used for the benchmarking.
3. Run the evaluation experiment.

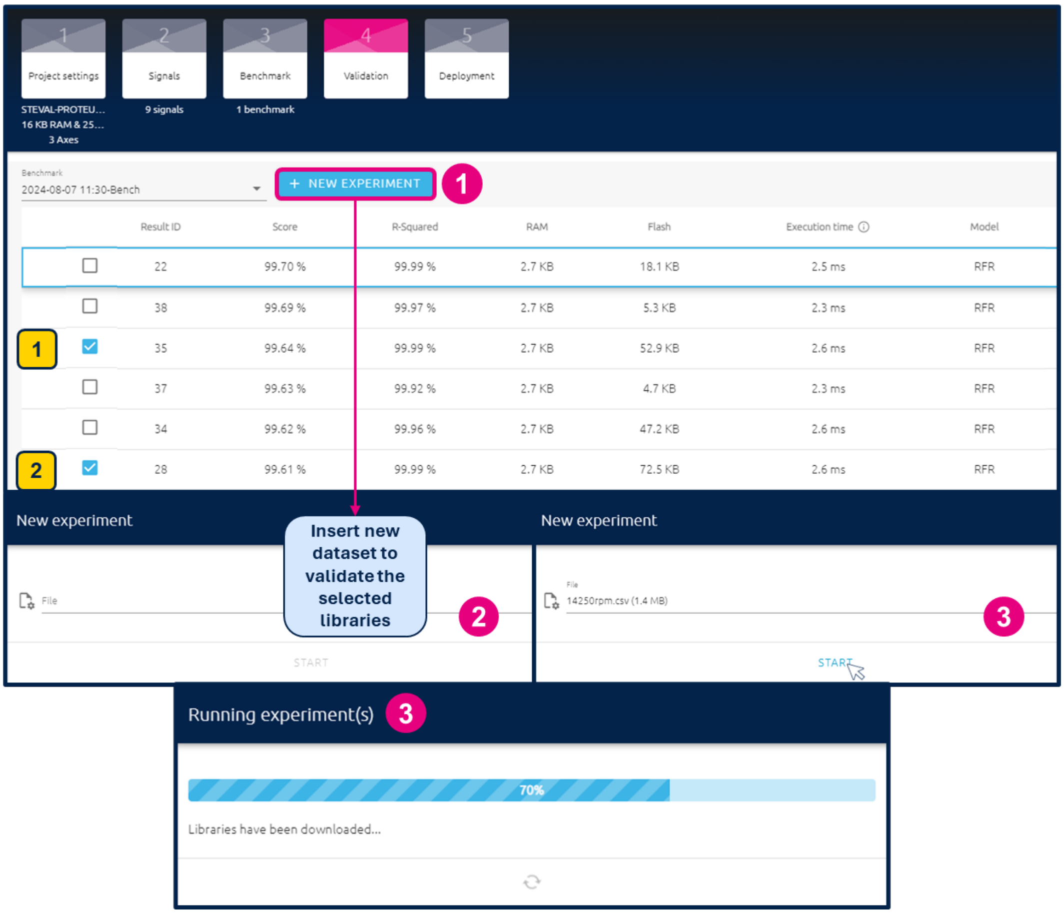
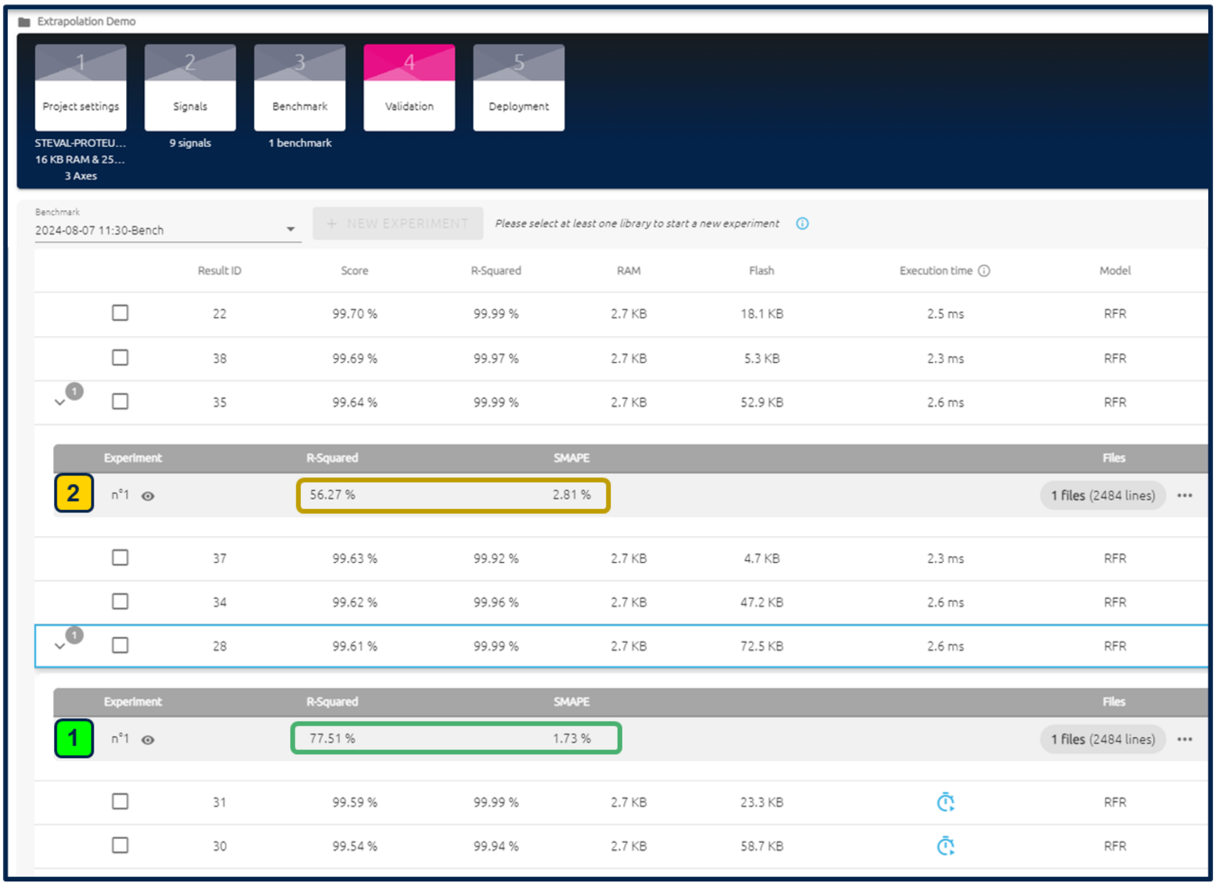


Figure 59. Models validation (E)

4. Evaluate and compare the model performances.
   Each library model is provided the R-Squared and R-Squared related to the validation dataset used, and the results could be different from the ranking outcomes from the benchmark.

Figure 60. **Validation results (E)**



In the example shown above, the validation is performed between the models in the 3rd and 6h ranking position but, after running the validation, the performance R-Squared index is changed, and the best is the 6th. So, using the validation procedure the user was able to find out that the 6th library is the most accurate and so the user can use it for the application.

Before proceeding with the final deployment, the user must select this library pushing on the relates crown on the right side.

4.2.3.5 *NEAI E library deployment*

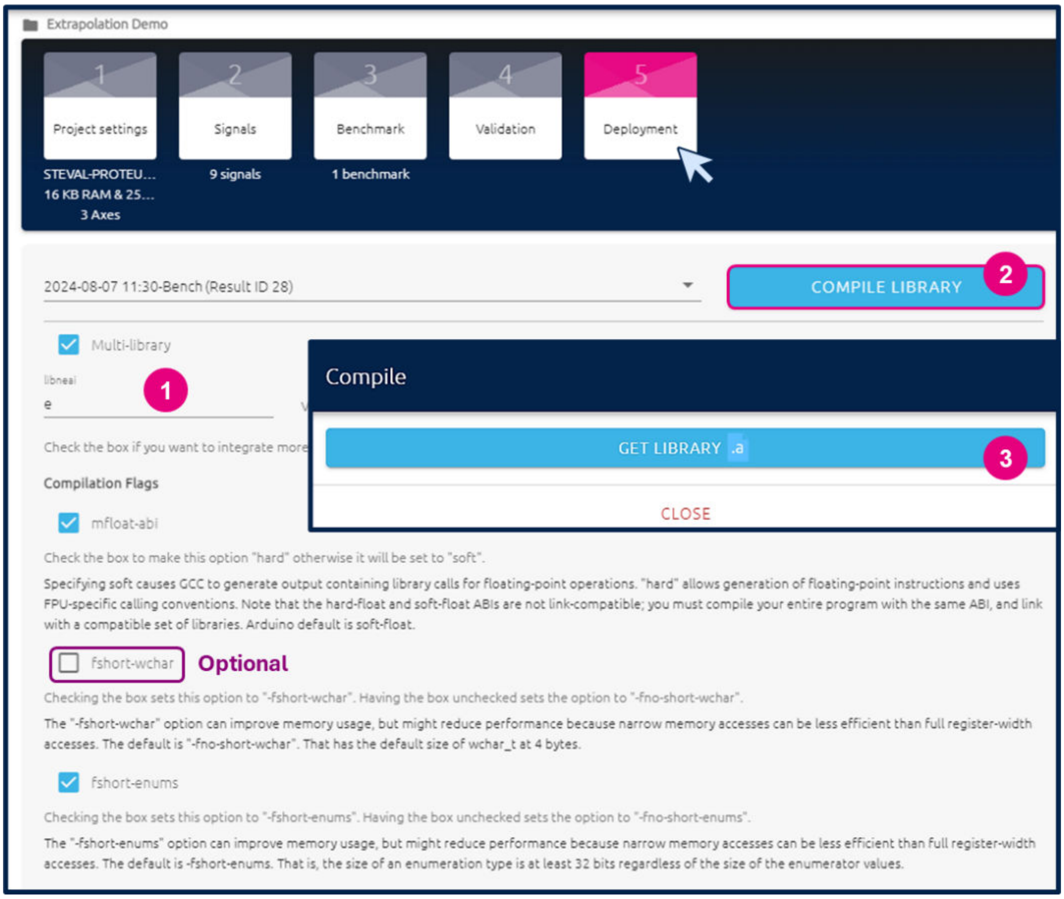
The last step for the NanoEdgeAI tool is to compile and release the libraries in a unique ZIP file containing the library model in compiled format, plus other files to complete the integration.

Pushing the **Deployment** button on the far right, to generate all the libraries files embedding the library models.

In case of extrapolation models:

1. Set the multi-library flag, to enable the suffix line insertion, and insert “e” just to identify these extrapolation files from others to insert in the application. Take care to check the fshort-wchar option if the library is integrated in a Keil® project and leave the other flags in their default state.
2. Push the **Compile Library** button.
3. Push the **Get Library .a** button.

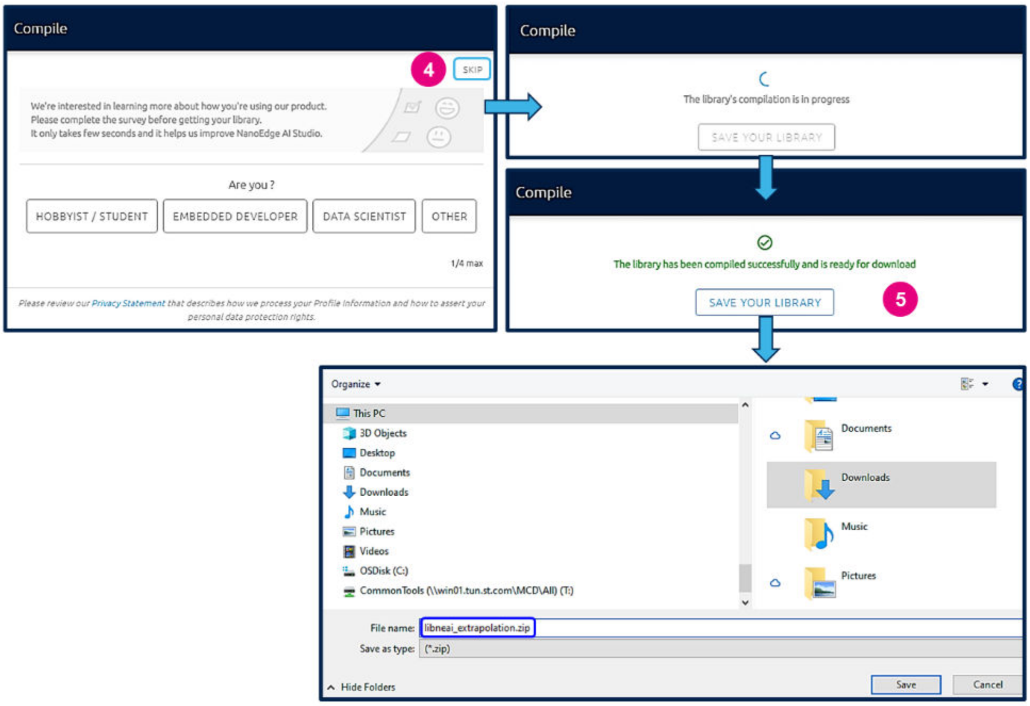
Figure 61. Deployment setting (E)



4. Replace with: Complete the survey or select the **SKIP** option in the window to start the compilation and ZIP file creation.

5. Push on **SAVE YOUR LIBRARY** to store the ZIP file.

Figure 62. Library compilation and ZIP file saving (E)



For further details about NanoEdge AI Studio, see **NanoEdge AI Studio Documentation**.

4.3 NEAI libraries integration

The FP-AI-PDMWBSOC2 firmware released contains three “stub” demonstration libraries useful just to see the functionality and behavior of, respectively, Anomaly Detection, N-Class Classification and Extrapolation.

These libraries must be replaced by the libraries generated from NanoEdgeAIStudio, in a proper way.

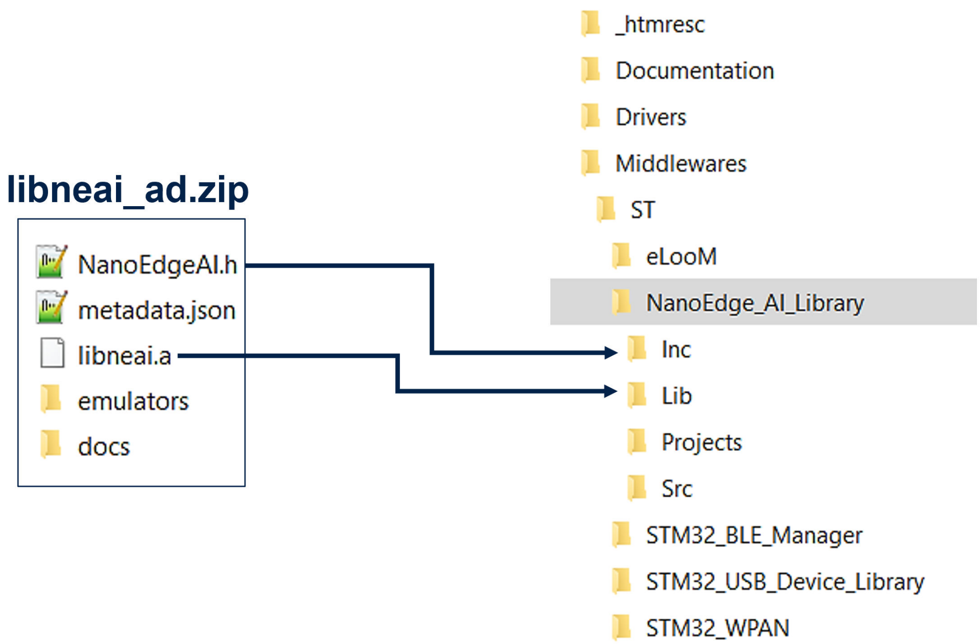
4.3.1 NEAI AD library integration

Open the ZIP file generated in NEAI AD library deployment and proceed as follow:

- Open the following path in the project folder: ***$PROJ_DIR$\Middlewares\ST\NanoEdge_AI_Library\Lib***
- Replace the existing stub libraries with the generated libraries:
  – ***libneai.a*** *(for IAR and STM32CubeIDE projects)*
  – ***libneai.lib*** *(for KEIL® projects)*
- Open the path: ***$PROJ_DIR$\Middlewares\ST\NanoEdge_AI_Library\Inc***

- Replace header files with new deployed:
  - **NanoEdgeAI.h**

Figure 63. **NEAI AD lib update in firmware architecture**



  - Compile again the FW application to update the application binary with the new library.
  - Update the STEVAL-PROTEUS directly as explained in the STEVAL-PROTEUS flash programming via Bluetooth® Low Energy OTA and STEVAL-PROTEUS flash memory programming via ST-LINK.

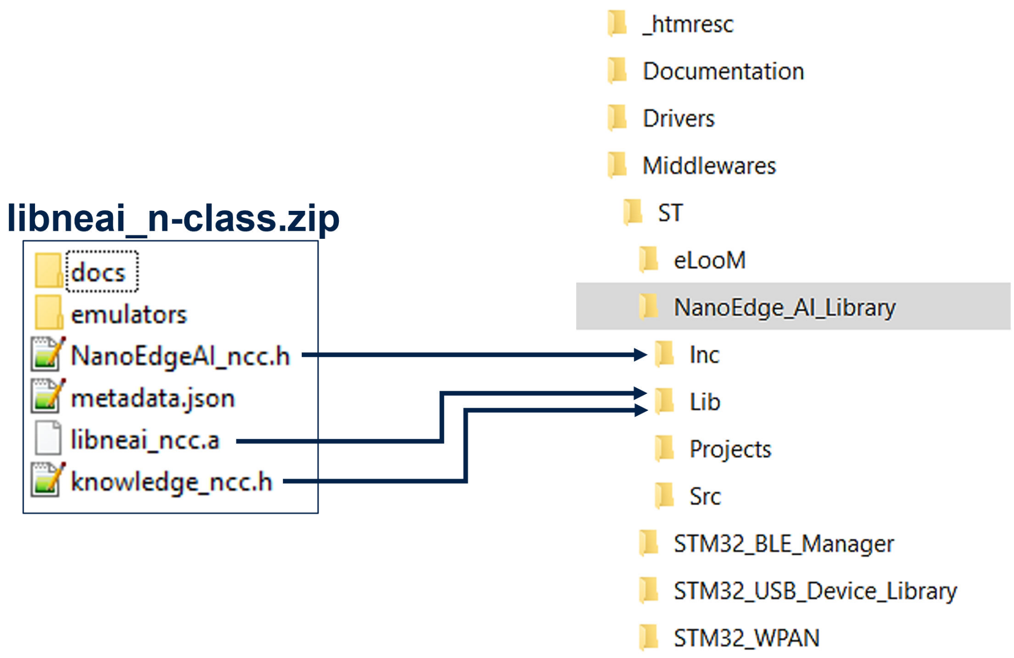
4.3.2 NEAI N-Class library integration

Open the ZIP file generated in NEAI N-Class library deployment and proceed as follow:

- Open the following path in the project folder: ***$PROJ_DIR$\Middlewares\ST\NanoEdge_AI_Library\Lib***
- Replace the existing stub library with the generated library:
  - ***libneai_ncc.a*** *(for IAR and STM32CubeIDE projects)*
  - ***libneai_ncc.lib*** *(for KEIL® projects)*
- Open the path: ***$PROJ_DIR$\Middlewares\ST\NanoEdge_AI_Library\Inc***

- Replace header files with new deployed:
  - **NanoEdgeAI_ncc.h**
  - **knowledge_ncc.h**

Figure 64. **NEAI NCC lib update in firmware architecture**



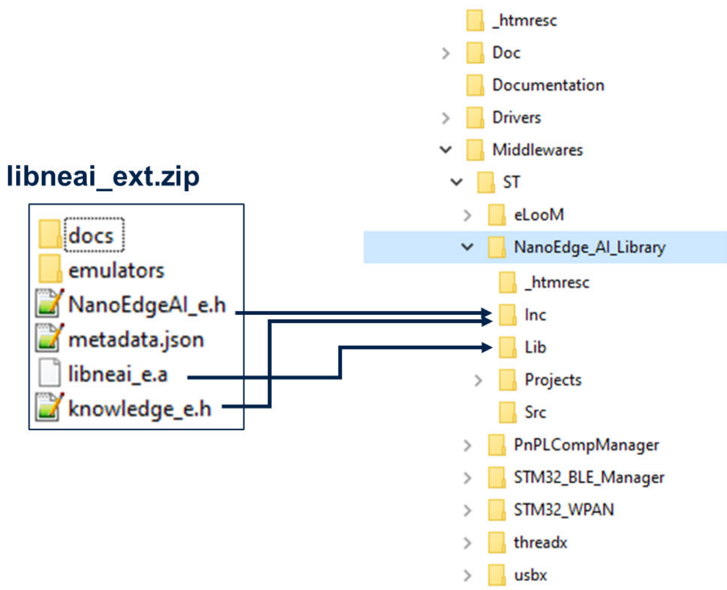
  - Compile again the FW application to update the application binary with the new library.
  - Update the STEVAL-PROTEUS directly as explained in the STEVAL-PROTEUS flash programming via Bluetooth® Low Energy OTA and STEVAL-PROTEUS flash memory programming via ST-LINK.

4.3.3 NEAI E library integration

Open the ZIP file generated in Section 4.2.3.5: NEAI E library deployment and proceed as follows:

- Open the following path in the project folder: ***$PROJ_DIR$\Middlewares\ST\NanoEdge_AI_Library\Lib***
- Replace the existing stub library with the generated library:
  - ***libneai_e.a*** *(for IAR and STM32CubeIDE projects)*
  - ***libneai_e.lib*** *(for Keil® projects)*
- Open the path: ***$PROJ_DIR$\Middlewares\ST\NanoEdge_AI_Library\Inc***
- Replace header files with new deployed:
  - **NanoEdgeAI_e.h**
  - **knowledge_e.h**

Figure 65. **NEAI E lib update in firmware architecture**



- Compile again the firmware application to update the application binary with the new library.
- Update the STEVAL-PROTEUS directly as explained in STEVAL-PROTEUS flash programming via Bluetooth® Low Energy OTA and STEVAL-PROTEUS flash memory programming via ST-LINK.

5 Running FP-AI-PDMWBSOC2

5.1 Using STEVAL-PROTEUS with ST BLE Sensor App

5.1.1 Bluetooth® Low Energy connectivity features

A typical Bluetooth® Low Energy application involves two parts: a Bluetooth® Low Energy master and a Bluetooth® Low Energy slave.

The STEVAL-PROTEUS plays the role of the slave (or peripheral), acting as the Bluetooth® Low Energy GATT server. It advertises and waits for connection.

A mobile device that runs the STBLESensor app is the master (or central), acting as the Bluetooth® Low Energy GATT client. It scans for devices and establishes the connection.

The STEVAL-PROTEUS sends data to the Bluetooth® Low Energy GATT client via Bluetooth® Low Energy and to a laptop via UART.

5.1.2 Bluetooth® Low Energy workflow overview

In the Bluetooth® Low Energy workflow, the STEVAL-PROTEUS performs the following actions:

1. initializes the hardware and software resources, making them ready for Bluetooth® Low Energy connection to a smartphone
2. sends the libraries status, battery level and, if a detection is running, also the asset status, inside the Bluetooth® Low Energy advertising message
3. accepts the Bluetooth® Low Energy connection request by a mobile device running the STBLESensor app
4. sends information about the AI libraries status and allows setting some parameters related to the libraries and sensors
5. sends information about the battery status

The mobile device that runs the STBLESensor can start the training and inference phases.

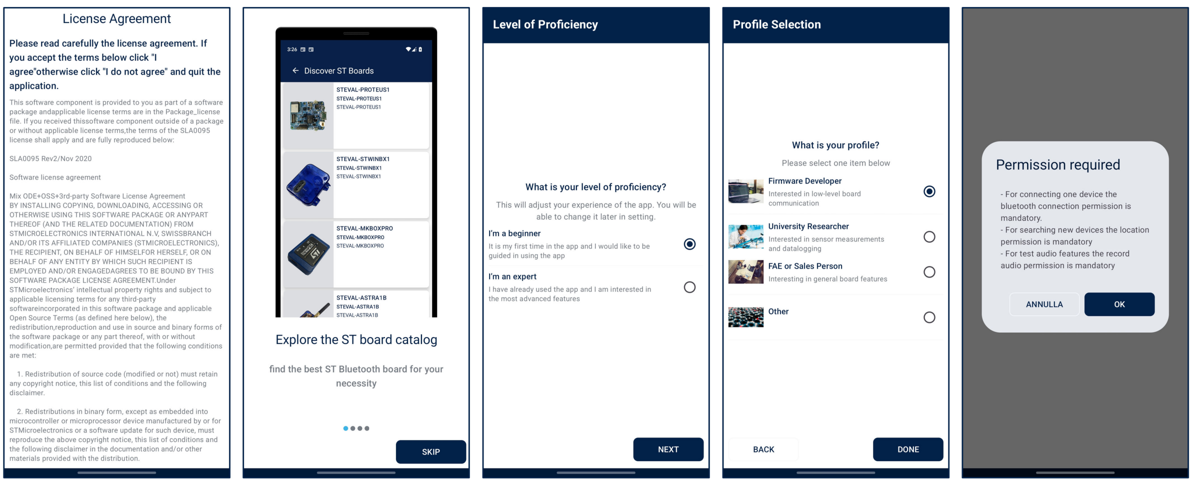
The application also sends information via UART to the PC, through the STDC14+ST-LINK+USB cable.

5.1.3 Bluetooth® Low Energy advertising

Since the evaluation board has been successfully programmed as described in STEVAL-PROTEUS flash programming via Bluetooth® Low Energy OTA and STEVAL-PROTEUS flash memory programming via ST-LINK has been switched on, the STEVAL-PROTEUS, as the Bluetooth® Low Energy peripheral device, sends Bluetooth® Low Energy advertising packets to establish a connection with a Bluetooth® Low Energy central device.

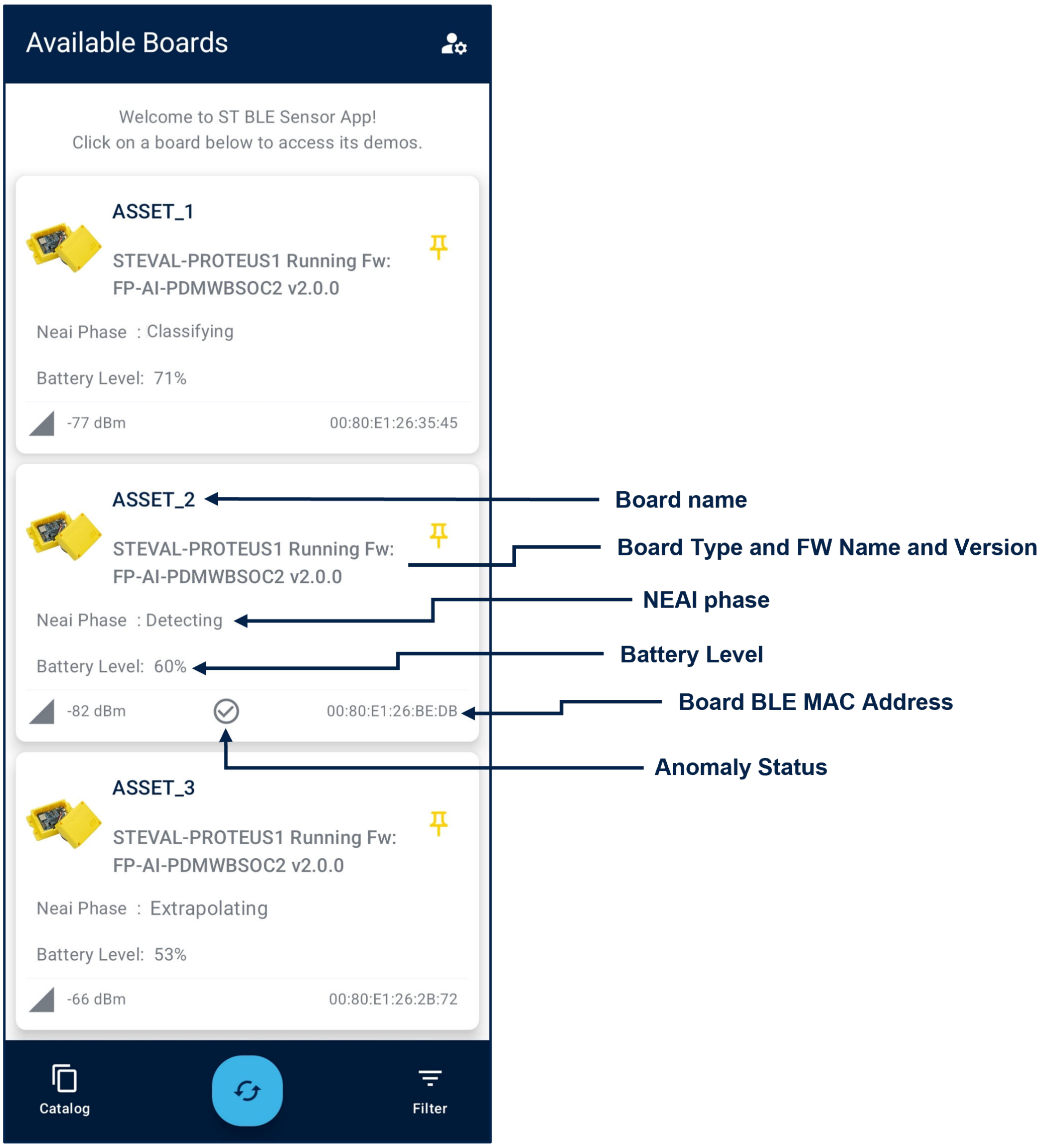
Launching the STBLESensor, you can connect a device.

Figure 66. Main page of the STBLESensor app for Android



When you open the STBLESensor app for the first time, you need to follow few steps before you can start using it. After granting all the required permissions, a new page displaying all available boards for connection is shown.

Figure 67. **Device list page (for Android STBLESensor)**



Active phase and battery level aside, if a detection is running, the STEVAL-PROTEUS sends the *Asset Status* (normal or anomaly behavior) to the Bluetooth® Low Energy advertising packet.

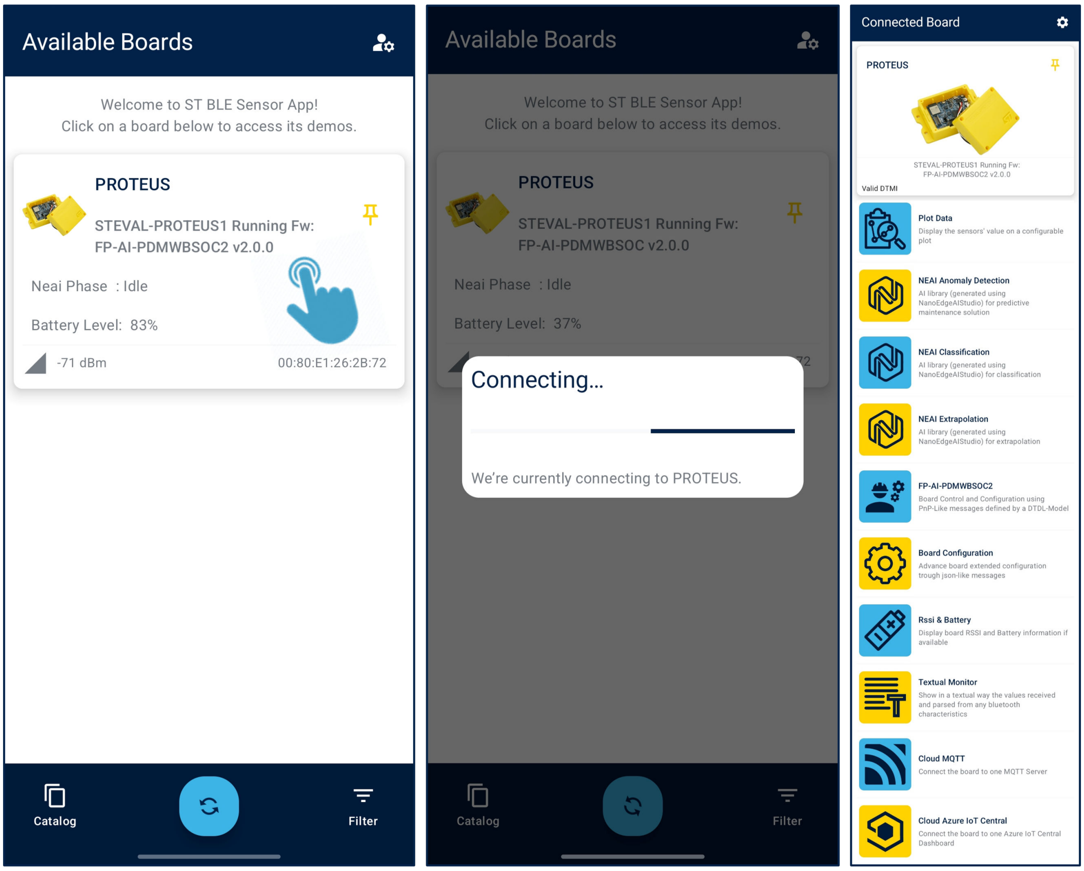
Table 4. **Asset status icons**

Normal behavior	Anomaly behavior
	

5.1.4 Connect a Bluetooth® Low Energy device and choose the application demo

To connect a device, just touch it. After that, you can retrieve all the information that the STBLESensor can manage for the selected board.

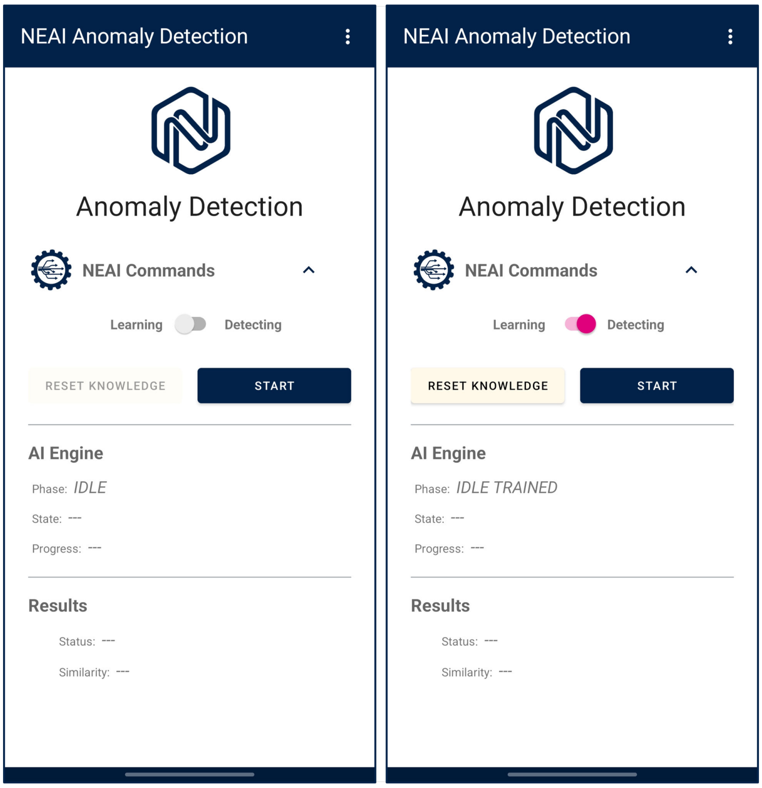
Figure 68. Device connection and demo summary



5.1.5 NEAI anomaly detection demo

As soon as the STEVAL-PROTEUS has been connected, user can open the NEAI Anomaly Detection page selecting it from the menu.

Figure 69. **NEAI anomaly detection demo (IDLE and IDLE TRAINED)**



The STEVAL-PROTEUS remains in the idle state until it gets a command to start a processing phase: learning or detecting.

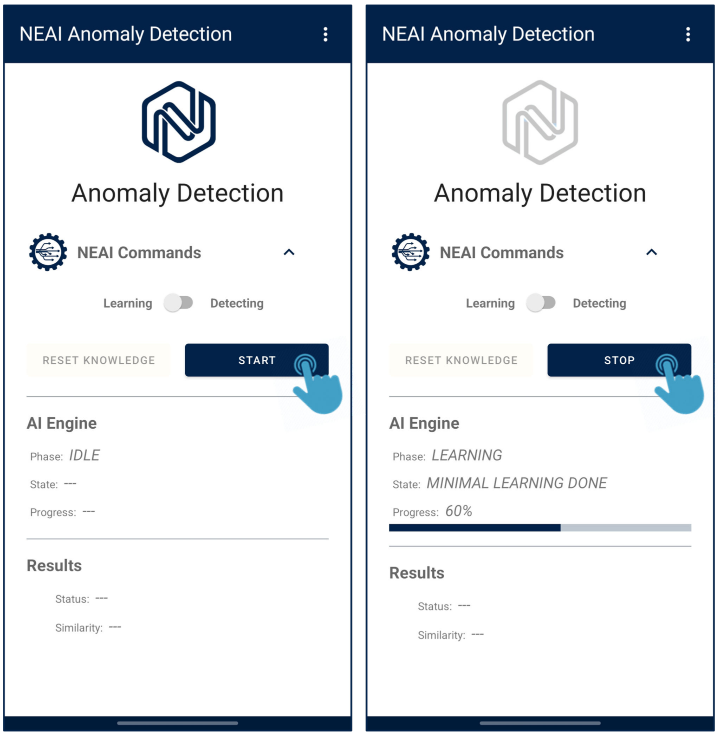
If you launch a learning phase, the library is trained with a number of signals that can be configured through the demo settings. The minimum amount of signals to ensure the proper operation is a predefined value in *$PROJ_DIR$\Middlewares\ST\NanoEdge_AI_Library\Inc\NanoEdgeAI.h*.

If a number of signals higher than the minimum value is used, the phase becomes IDLE TRAINED and the RESET KNOWLEDGE button is activated: by pressing it, the original hyperparameters of the library are restored, which restarts from the IDLE state.

Using the switch and start button, you can launch the learning or detecting phase:

- By moving the switch on learning side, the learning phase starts; during this phase, the vibration data are generated and sent to the NEAI library to train the ML model.

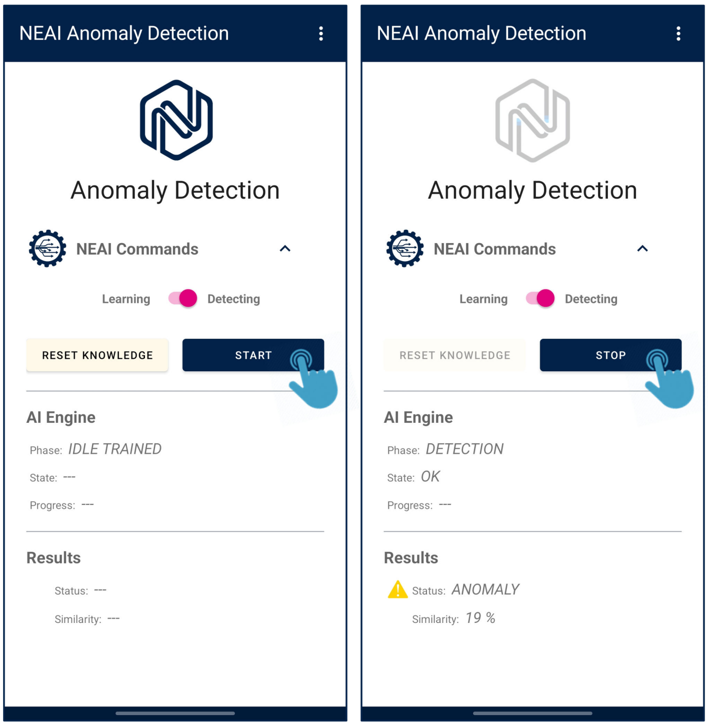
Figure 70. **Learning on the NEAI demo**



- By moving the switch on detecting side, the detecting phase starts; during this phase, the vibration data are generated and sent to the NEAI library to recognize if the asset behavior is normal or abnormal by an indicator that is named *similarity*, this last assumes (value from 0 to 100%).

If the similarity is greater than the set threshold value, an anomaly occurred. You can also set the threshold value by demo settings.

Figure 71. Detecting on the NEAI demo



In both the learning and detecting phase, the start button is replaced by the stop button to end the processing phase.

Note: The learning phase can be automatically stopped via timer or signal settings (see NEAI anomaly detection demo settings) but, by default, both are initialized to zero. Therefore you need to press the stop button to end the learning.

Note: You cannot start both phases at the same time.

The NEAI demo consists of two sections: *AI Engine* and *Signal*. These sections provide several info related to the NEAI library status and the detected signal.

Within *AI Engine* you can find:

- the active phase of the NanoEdgeAI library:
  - **IDLE**, if the library is not running.
  - **LEARNING**, if the library is learning from the raw data generated by a sensor to train the ML model.
  - **DETECTING**, if the library is monitoring the raw data generated by a sensor to detect anomaly behaviors.
  - **IDLE TRAINED**, if the library is not running but a learning phase has been done and the minimum signal quantity has been learned.
- The NEAI state, which describes the library operation:
  - **NOT ENOUGH LEARNING**, when you launch a learning or a detecting phase without having learnt the minimum quantity of signals.
  - **MINIMAL LEARNING DONE**, when you have launched a learning phase and you have achieved the minimum quantity of signals to learn.
  - **OK**, when you have launched a detecting phase and the library works properly.
- The progress bar, which is active only if you have set the learning time or signals to learn; in this case, the progress bar is filled according to the time or signal settings.

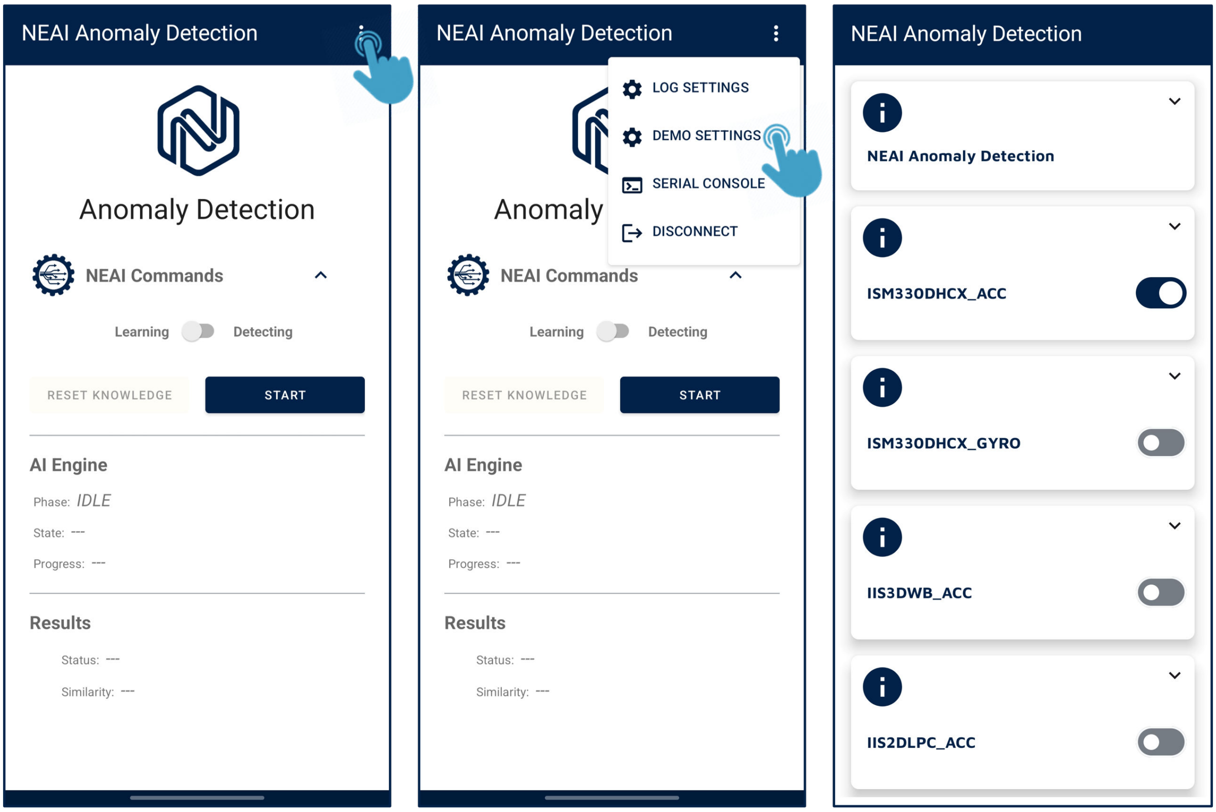
The *Signal* section is active during the detecting phase. Within *Signal* you can find:

- The status of your asset, which can be normal or anomaly according to the similarity indicator provided by the NEAI library.
- The similarity indicator, which can assume values in the range of 0-100%; if this indicator is less than the threshold value, the asset status is anomaly, otherwise the status is normal.

5.1.5.1 *NEAI anomaly detection demo settings*

Using demo setting button, user can access to several setting option where library parameters and sensors configuration can be changed according to user needs.

Figure 72. **NEAI anomaly detection demo settings**



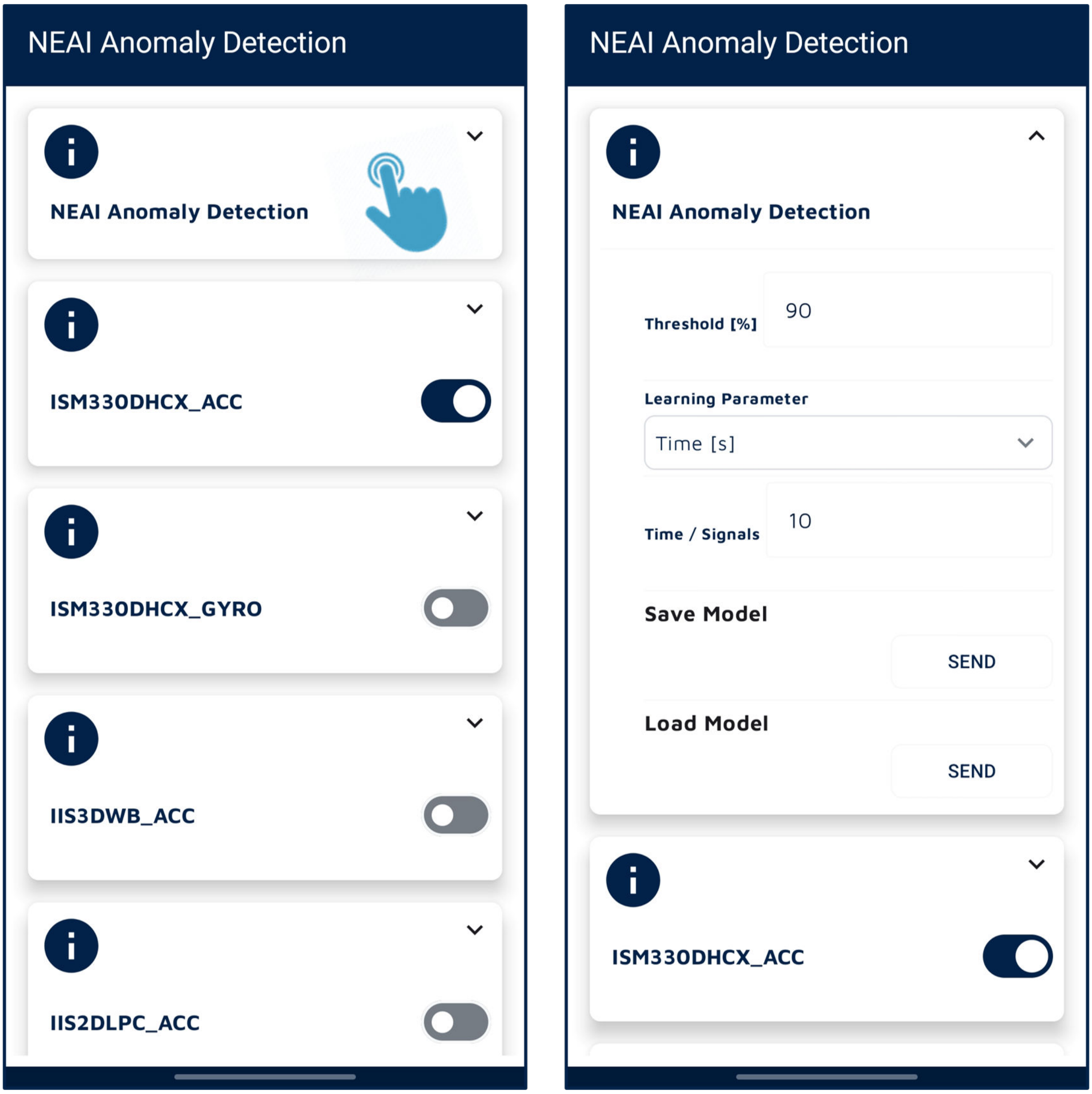
Anomaly detection section allow to view the current values and set the following parameters:

- The similarity threshold value under which a signal is considered normal or abnormal.
- The learning parameter, it can be equals to *Time* or *Signals*, if *Time* has been selected, after the specified number of seconds the phase is automatically stopped, else *Signals* has been selected, when the specified number of signals have been learned the phase is automatically stopped.
- Time/Signals field, where user can specify the leaning duration, if *Time* has been chosen, or number of signals, if *Signals* has been chosen.

Furthermore, there are two buttons which allow to save and load in or from the NOR external memory.

Note: *It is not allowed to set the Time and Signals as learning parameter for the same learning session.*

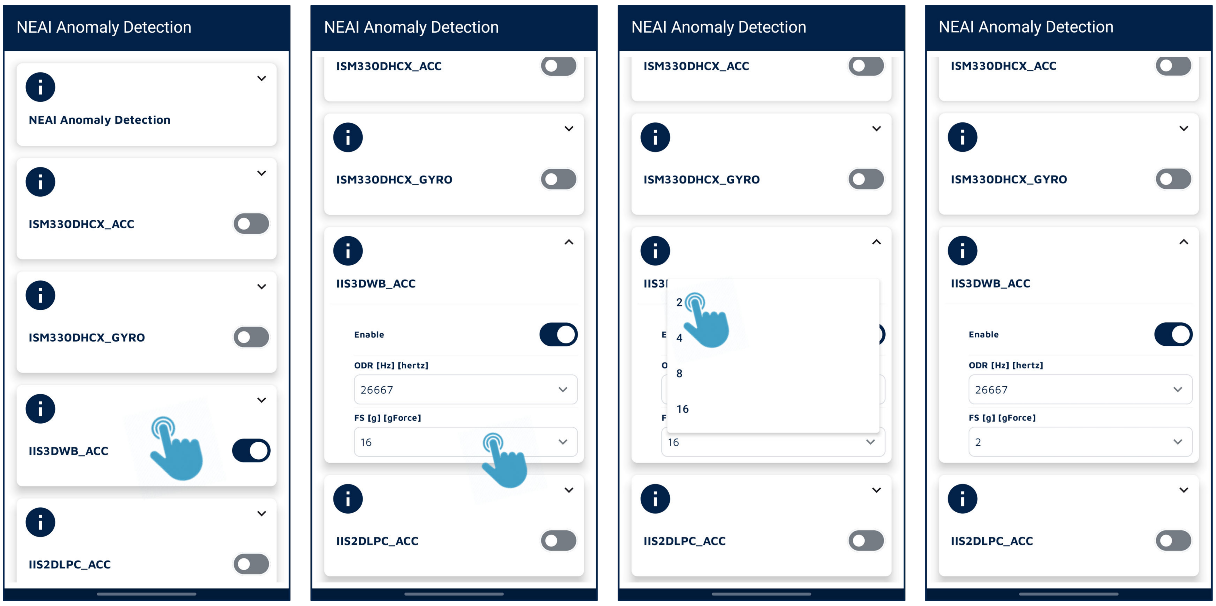
Figure 73. NEAI anomaly detection demo settings – Library parameters



Sensors sections allow to enable or disable, set the nominal output data rate and the full scale for every available sensor (IIS3DWB_ACC and ISM330DHCX_ACC).

Note: Only one sensor at a time can be enabled: the best sensor setup to use is the one chosen during the data acquisition phase.

Figure 74. **NEAI anomaly detection demo settings – Sensors parameters**

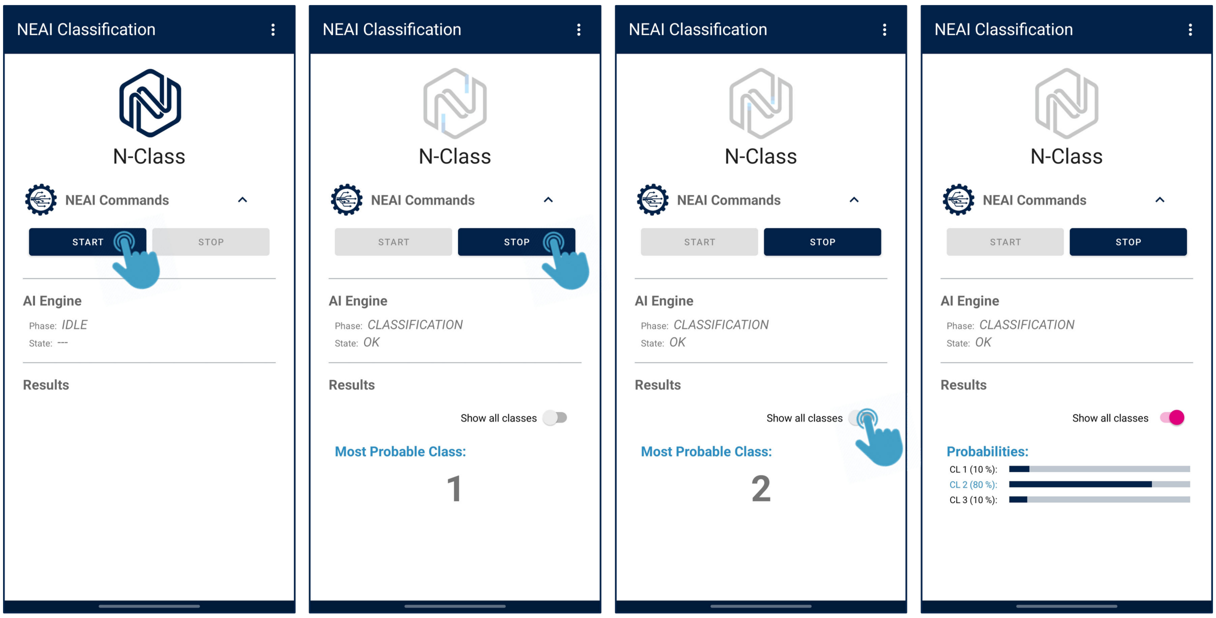


5.1.6 NEAI classification demo

As soon as the STEVAL-PROTEUS has been connected, user can open the *NEAI Classification* page selecting it from the menu.

The STEVAL-PROTEUS remains in the idle state until it gets a command to start the classification phase.

Figure 75. **Classifying on the NEAI demo**



Using the start button, you can launch the classification phase, the vibration data are generated and sent to the NEAI library to classify it.

The NEAI demo consists of two sections: AI Engine and Signal. These sections provide several info related to the NEAI library status and the classified signals.

Within AI Engine you can find:

- The active phase of the NanoEdgeAI library:
  - IDLE, if the library is not running.
  - CLASSIFICATION, if the library is classifying the raw data generated by a sensor.

- The NEAI state, which describes the library operation:
  - OK, when you have launched a classification phase and the library works properly.

The Signal section is active during the classification phase. Within Signal you can find:

- The *Most Probable Class*, namely the recognized class name, it is the class to which has been attributed the highest probability.
- Using the *Show all classes* switch, user can view the probabilities associated to every classes.

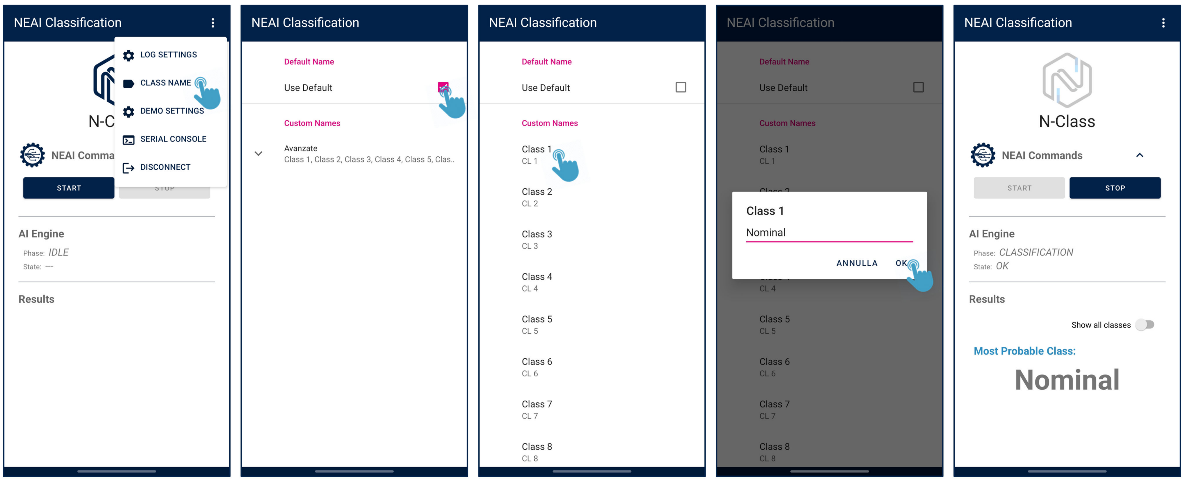
5.1.6.1 *Classes names customization*

Since classes names is strongly dependent from user use case, it is possible to change the default names through the *class name* setting.

To customize the classes names, user needs to follow the steps below:

- Open setting window and tap on *Class Name* option.
- Uncheck *Use Default* flag.
- Tap on the classes which you want to name and enter the new names.
- Return to the main page and start classification with your customized classes names

Figure 76. Classes names customization

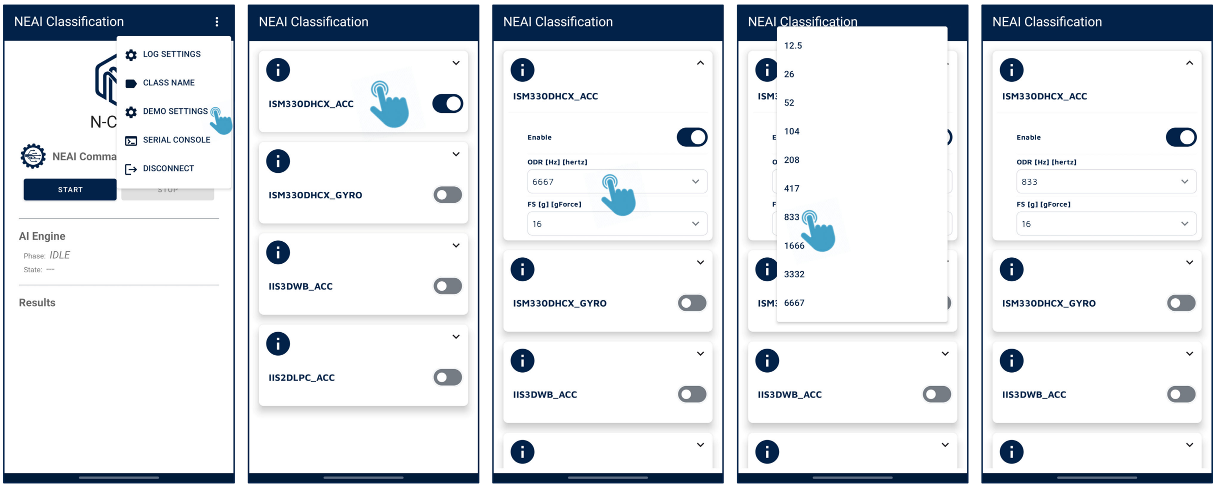


5.1.6.2 *NEAI classification demo settings*

Using demo setting button, user can access to several setting option where sensors configuration can be changed according to user needs. As explained in NEAI anomaly detection demo settings, it is easy to enable or disable sensors and set the nominal output data rate and full scale.

Note: *Only one sensor at a time can be enabled: the best sensor setup to use is the one chosen during the data acquisition phase.*

Figure 77. NEAI classification demo settings - Sensors Parameters

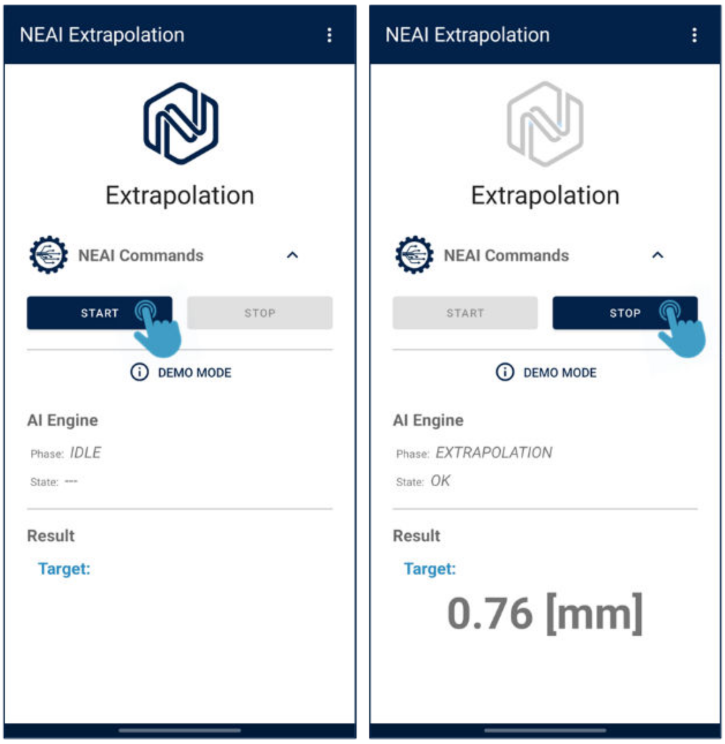


5.1.7 NEAI extrapolation demo

As soon as the STEVAL-PROTEUS has been connected, the user can open the *NEAI Extrapolation* page selecting it from the menu.

The STEVAL-PROTEUS remains in the idle state until it gets a command to start the extrapolation phase.

Figure 78. **Extrapolating on the NEAI demo**



Using the start button, you can launch the extrapolation phase, the vibration data are generated and sent to the NEAI library to extrapolate the target value.

The NEAI demo consists of two sections: AI Engine and Result. These sections provide various information related to the NEAI library status and the extrapolated result.

Within *AI Engine* you can find:

- the active phase of the NanoEdgeAI library:
  - IDLE, if the library is not running
  - EXTRAPOLATION, if the library is extrapolating from raw data generated by a sensor
- the NEAI state, which describes the library operation:
  - OK, when you have launched an extrapolation phase and the library works properly

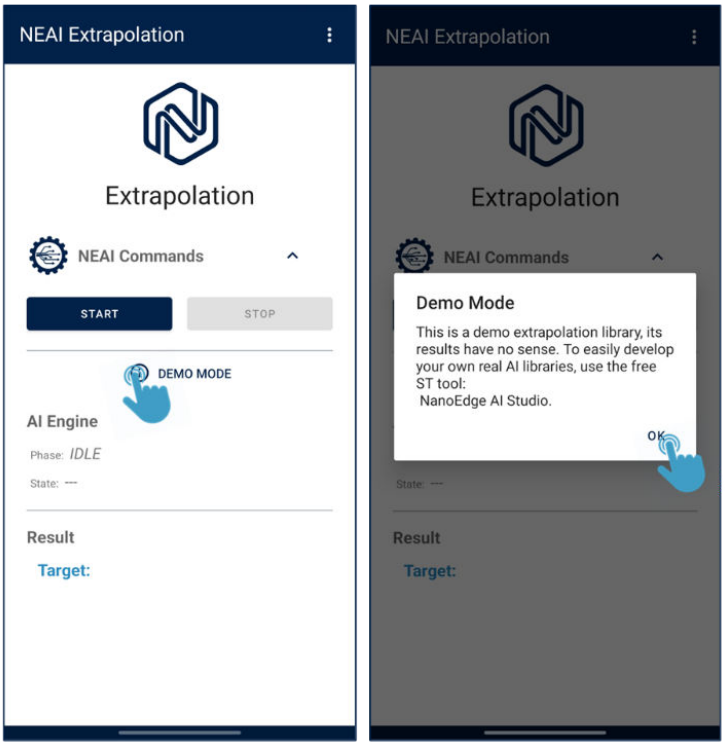
The Result section is active during the extrapolation phase. Within *Result* you can find:

- The *Target*, it is made up of a numerical value plus a unit of measurement.

Note: The displayed unit of measurement can be customized modifying the source code $PROJ_DIR$ \Projects\STM32WB5MMG-PROTEUS\Applications\PDMWBSOC2\Core\Src\AppController.c line 458. It is recommended not to exceed the length of 5 characters (for example, 'mm/s2').

When a stub (demo) extrapolation library was integrated into the PDMWBSOC2 application, an info label appears in the center of the demo.

Figure 79. Demo mode alert

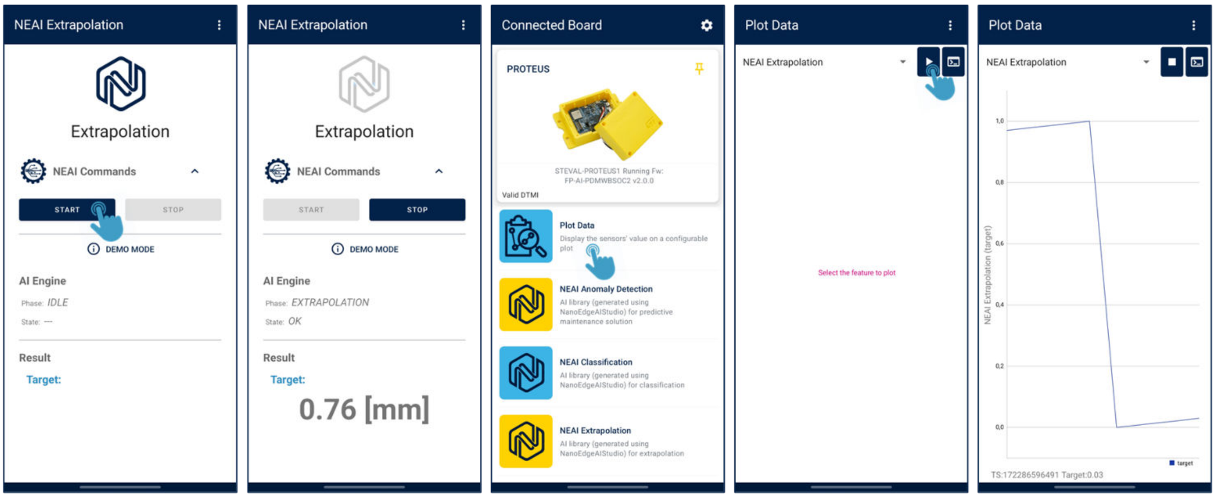


The *Demo Mode* label disappears when the user integrates its library.

5.1.7.1 *NEAI extrapolation result plot*

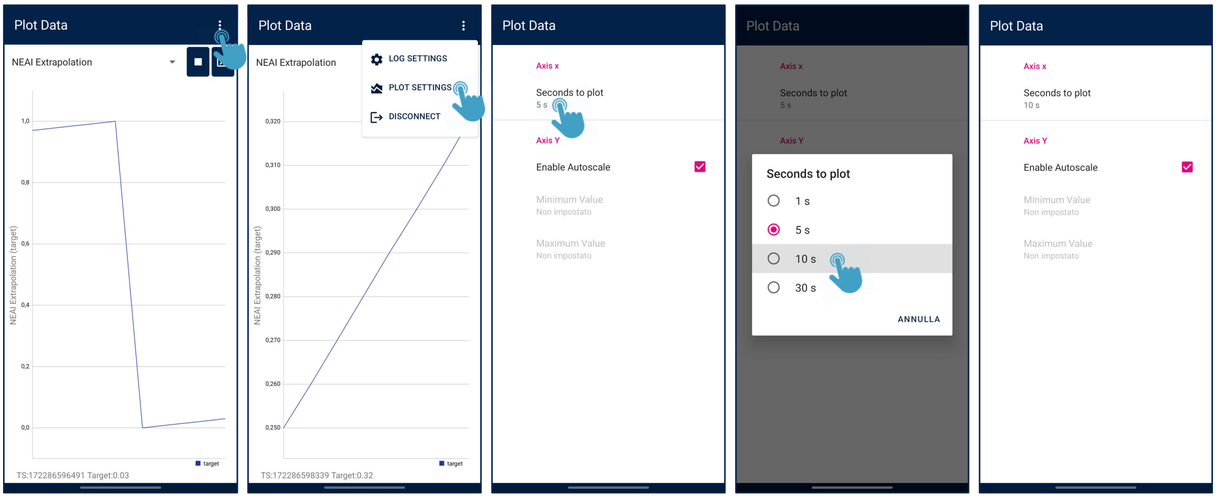
Since the target value is always a continuous variable, it is useful to monitor his trend over time. The user can view the extrapolating results plot, running extrapolation processing as explained in Section 5.1.7, and then opening the *Plot Data* demo.

Figure 80. NEAI extrapolation result plot



In addition, the user can modify the time interval on which the plot is generated by opening the *Plot Settings* menu and selecting the desired value for the x-axis.

Figure 81. NEAI extrapolation plot setting

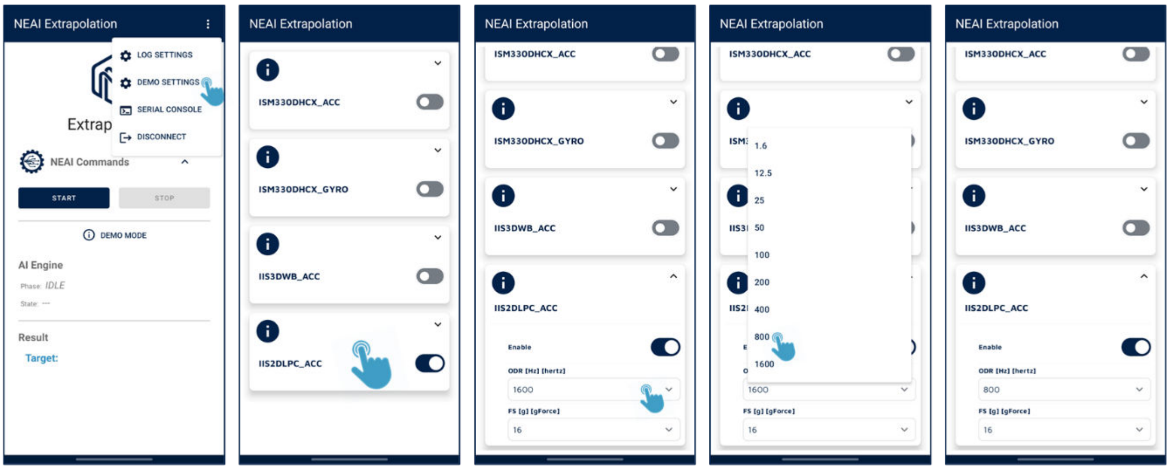


5.1.7.2 *NEAI extrapolation demo settings*

Using the demo setting button, the user can access several setting options where sensors configuration can be changed according to user needs. As explained in NEAI anomaly detection demo settings, it is easy to enable or disable sensors and set the nominal output data rate and full scale.

Note: *Only one sensor at a time can be enabled: the best sensor setup to use is the one chosen during the data acquisition phase.*

Figure 82. NEAI extrapolation demo settings - sensors parameters



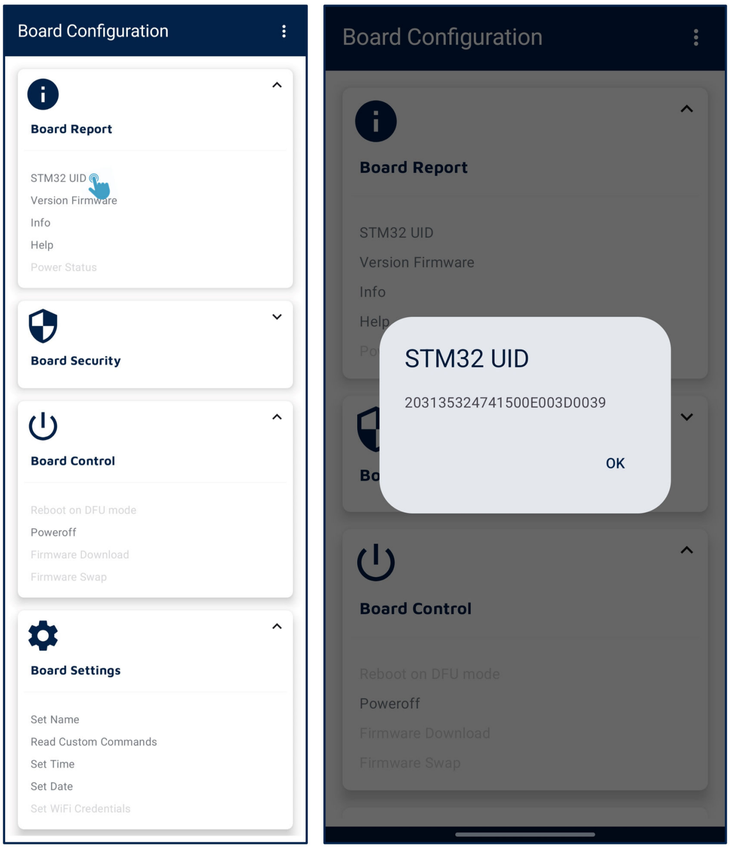
5.1.8 Board configuration

Touching *Board Configuration*, the mobile app shows information about the board.

In the *Board Report* section, you can retrieve the following information.

- **STM32 UID**: the unique 96bit ID of STM32WB mounted on the STEVAL-PROTEUS.

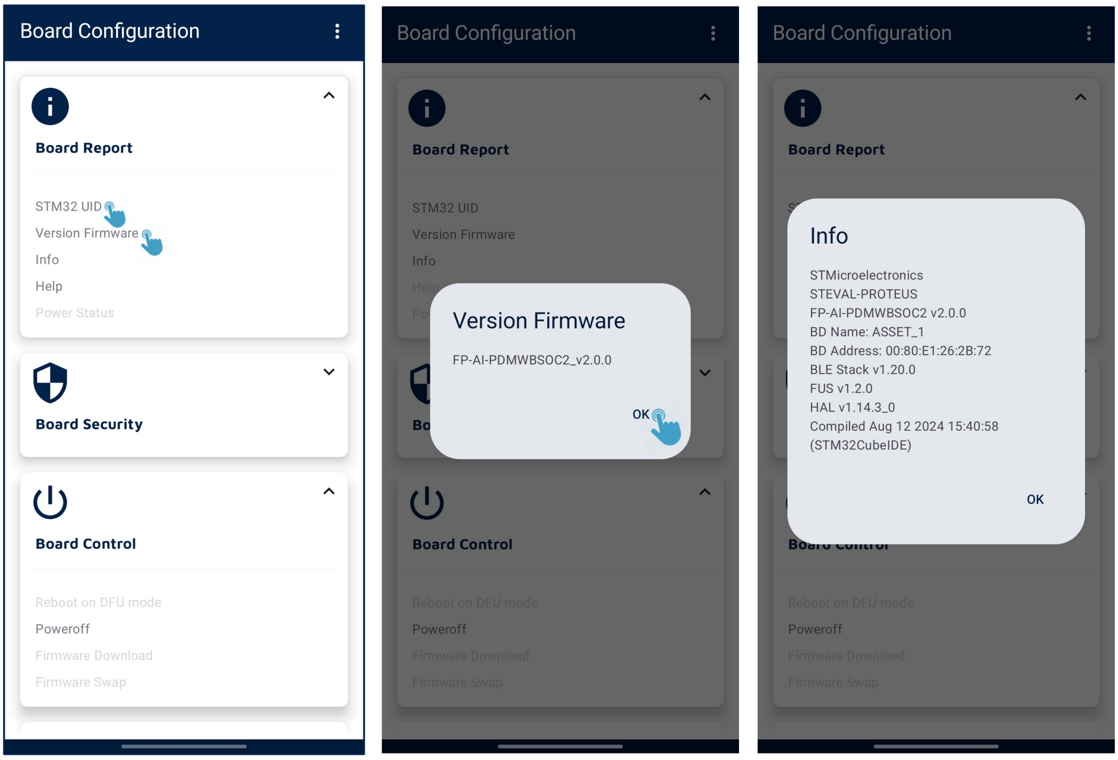
Figure 83. **Board configuration summary and STM32 unique ID**



- the **Version firmware**: the firmware release loaded on the STEVAL-PROTEUS.

- **Info**: a set of more detailed information about the firmware loaded into the STEVAL-PROTEUS.

Figure 84. **Firmware version and info**

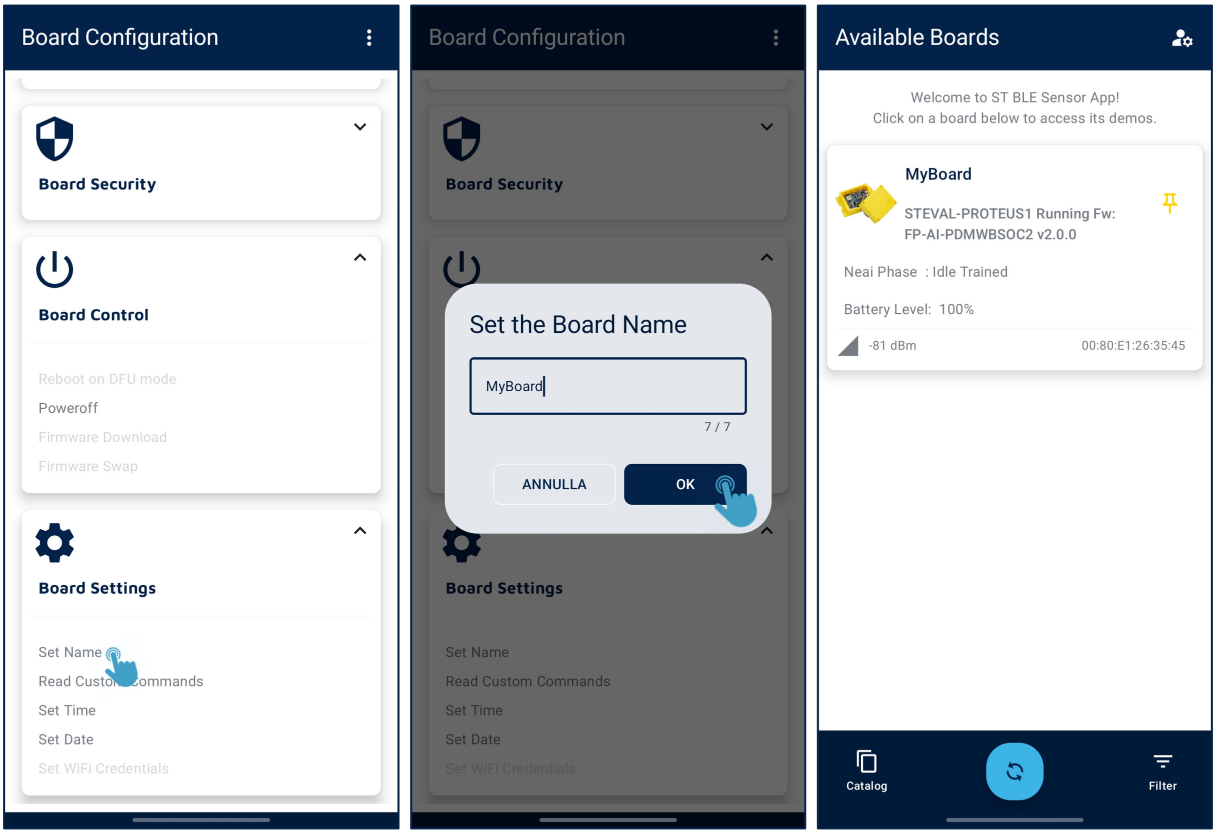


In the *Board Control* section, you can power off the board.

In the *Board Settings* section, you can handle the following information:

- **Set name**: to change the board connected name, keeping it internally, to show it during Bluetooth® Low Energy advertising.

Figure 85. Board name settings



- **Read custom commands**: to open a customized panel containing the commands setting directly built by the firmware, as described in the next section.

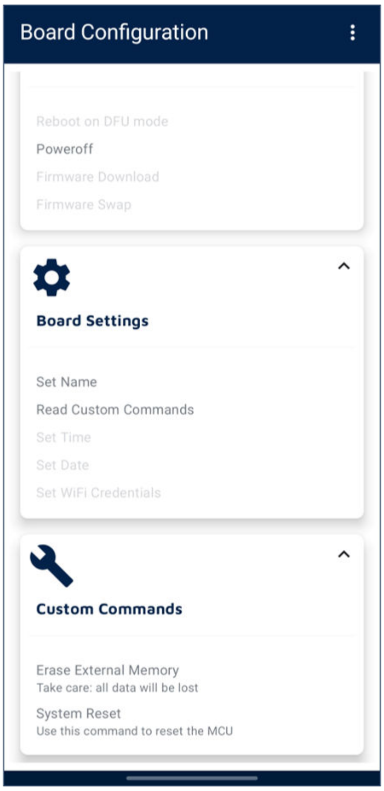
5.1.8.1 *Custom commands*

Tap on *Read Custom Commands* button to open the *Custom Commands* section.

This customized menu provides commands to:

- Erase external NOR flash memory content
- Reset the MCU

Figure 86. Read custom commands

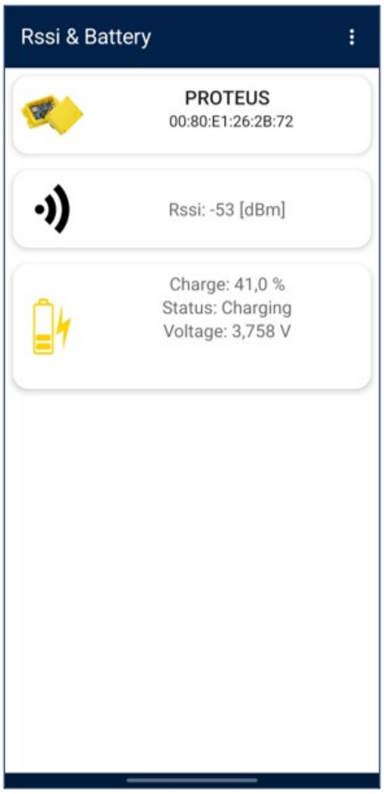


5.1.9 RSSI & Battery

Touching *RSSI & Battery* in the STBLESensor menu, the user gets information about:

- the board name and MAC address
- the RSSI
- the battery information such as the level and status of the charge and voltage

Figure 87. RSSI and battery status



5.1.10 Use FUOTA to test different libraries

You can easily switch between different application binaries that contain libraries with different performance in terms of score, accuracy, RAM, or flash memory, or generated for different use cases.

The best practice could be to:

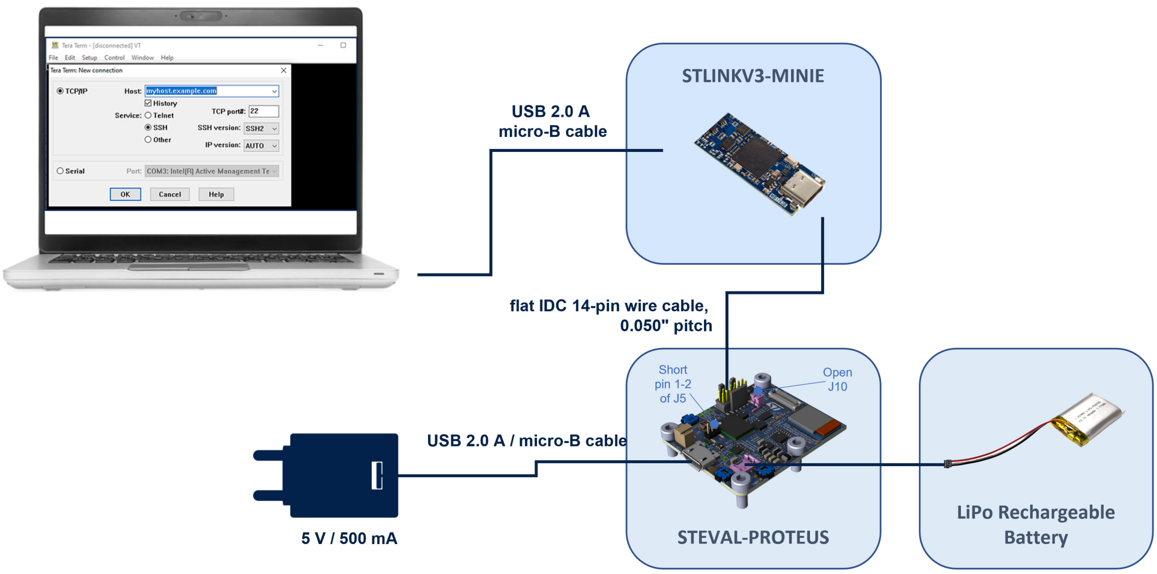
- Generate several NanoEdge AI libraries for different anomaly detection or/and classification use cases
- Embed these compiled binaries into the FP-AI-PDMWBSOC2, creating several application binaries
- Use the FUOTA to switch through the application, to cover and test different application use cases

5.1.11 Debug using a terminal emulator

The STLINK-V3MINIE also provides a virtual COM port interface, which allows the host PC to communicate with the target microcontroller through a UART. It also allows the STEVAL-PROTEUS that runs the FP-AI-PDMWBSOC2 to send information to a terminal console when connected to a PC.

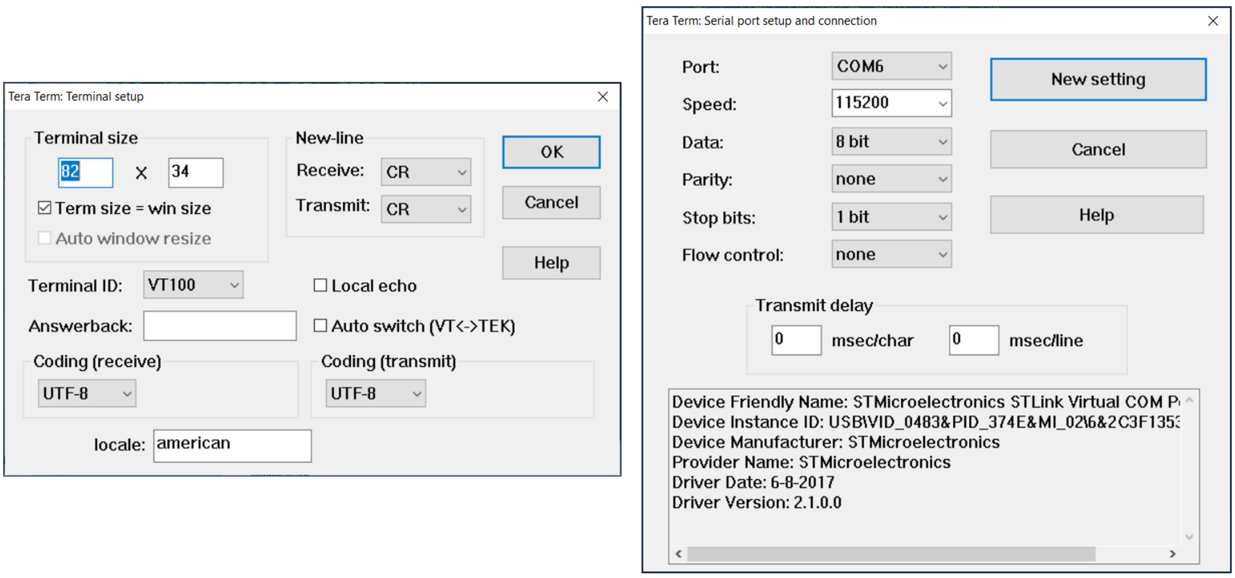
The figure below shows how to connect the STEVAL-PROTEUS to a PC able to run a terminal emulator such as Tera Term.

Figure 88. **STEVAL-PROTEUS connection with a terminal emulator**



Configure the terminal emulator as shown below.

Figure 89. **PC terminal setting for the debug console**



This debug terminal allows you to monitor:

- the system initialization: to check if all the application tasks and application services are properly initialized and started
- the Bluetooth® Low Energy initialization: to check if all the Bluetooth® Low Energy services and characteristics are properly initialized; it also shows the information about the Bluetooth® Low Energy stack and the FUS version, the Bluetooth® Low Energy address, etc.
- the application state machine: to check if state transactions occurred according to the state machine design

Note: *You can find more details about application state machine in $PROJ_DIR$\Documentation\FP-AI-PDMWBSOC2_Package.chm*

The following codeblock shows a possible output on the debug terminal.

```
System Initialization
INIT: reset flags: 0x4000000
SysTS: System timestamp service ready.
INIT: added 10 managed tasks.
INIT: task hardware initialization.
...
DONE.
CTRL: registered 3 STM32 generic commands
CTRL: registered 9 application commands
NMP: initialization done.
BCP: initialization done.
I2CBUS: connected device: 1
SPIBUS: connected device: 1
SPIBUS: connected device: 2
INIT: system initialized.
INIT: eLooM version v3.4.1
INIT: free heap = 44580.
INIT: SystemCoreClock = 64000000Hz.
I2C: start.
I2CBUS: start the driver.
SPI: start.
SPIBUS: start the driver.
ISM330DHCX: start.
IIS3DWB: start.
IIS2DLPC: start.
NEAI: start.
UTIL: start.
CDC: start.
CTRL: start.
DPT1:0 DPU_MESSAGE_ID_ADD_LISTENER
NEAI: NEAI_CMD_INIT
NEAI: NEAI_CMD_INIT
NMP: load NEAI AD data from NOR Flash memory ... (16354 bytes) done!
BLE: start.


SHCI_SUB_EVT_CODE_READY - WIRELESS_FW_RUNNING

***************************************************************************
STMicroelectronics
STM32STM32WB5MMG - Anomaly Detection,  Classification and Extrapolation by NEAI based on Thre
adX OS
FP-AI-PDMWBSOC2 V2.0.0 Compiled Sep 11 2024 11:10:22 (STM32CubeIDE)
MCU Unique device ID is 0x203135324741500E0022004B
MCU Flash Size is 1024 KB
PROTEUS
BD Address: 00:80:E1:26:2B:72
BLE Stack v1.20.0
BLE Stack Branch 0 Type 2
FUS v1.2.0
***************************************************************************

BlueST-SDK V2
Config  Service added successfully
Console Service added successfully
BLE PnPLike features ok
BLE NEAI Anomaly Detection char is ok
BLE NEAI Classification char is ok
BLE NEAI Extrapolation char is ok
BLE Battery features ok
BLE Ota features ok
Features Service added successfully (Status= 0x0)
Call to SetConnectableFunction
aci_gap_update_adv_data OK
```

5.2 Using the STEVAL-PROTEUS with the command line interface

The CLI allows you to control the application by sending the same commands, described in the STBLESensor app, via command-line input to be processed on the device.

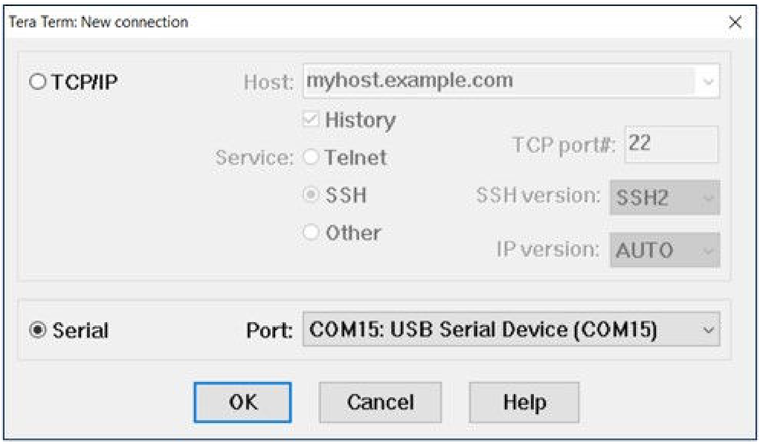
To use this interface, you need:

- a laptop with Tera Term software
- a micro-B to A USB cable

5.2.1 Set the serial terminal configuration

Start Tera Term and select the proper connection, that is the serial, which indicates the right port connected (USB serial device, COM15 in the example).

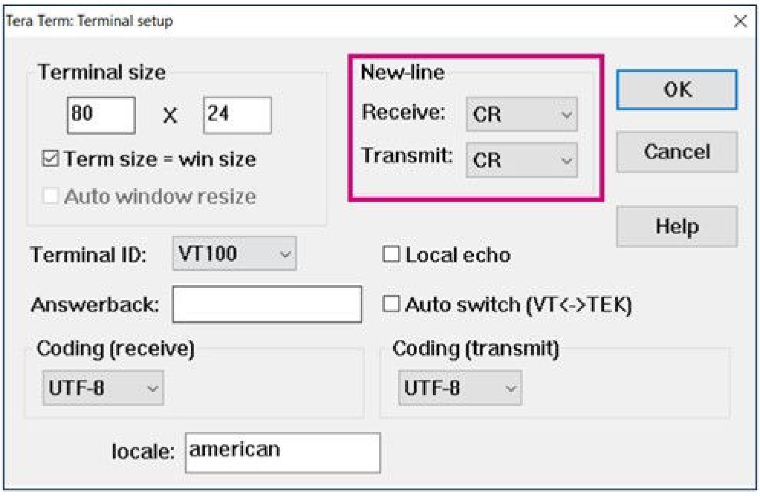
Figure 90. Open the CLI via PC terminal console



Use the following parameters for the terminal setup:

- New line
- Receive: CR
- Transmit: CR

Figure 91. Tera Term connection settings

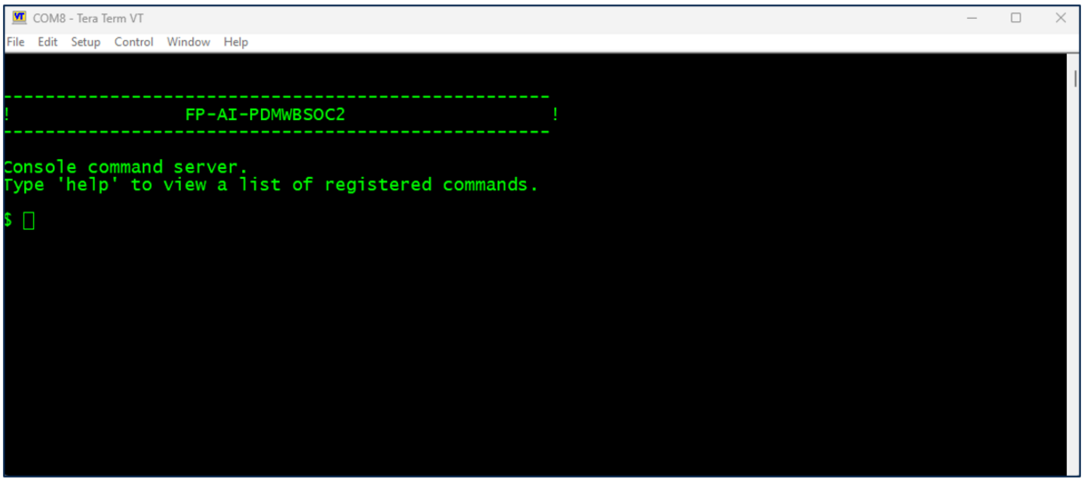


The interactive console can be used with the default values.

5.2.2 Start the FP-AI-PDMWBSOC firmware

Restart the board by pressing the reset button. The following welcome screen is displayed on the terminal.

Figure 92. Start the CLI program execution



From this point, start entering the commands directly or type *help* to get the list of the available commands along with their usage guidelines.

5.2.3 Execution phases and command summary

The two system execution phases are:

- **NanoEdge AI learning**: sensor data are transmitted to the NanoEdge AI library to train the model.
- **NanoEdge AI detecting**: sensor data are transmitted to the NanoEdge AI library to detect anomalies.
- **NanoEdge AI classifying**: sensor data are transmitted to the NanoEdge AI library to classify asset behaviours.
- **NanoEdge AI extrapolating**: sensor data are transmitted to the NanoEdge AI library to extrapolate the target value.

Each execution phase can be started and stopped through a user command, `$ start <execution phase>`, issued through the CLI, where the valid values for the `execution phase` are:

- **neai_learn**
- **neai_detect**
- **neai_class**
- **neai_extrapolate**

An execution context, which is a set of parameters that control the execution, is associated with each execution phase. One single parameter can belong to more than one execution context. The CLI provides commands to set and get the execution context parameters. The execution context cannot be changed while an execution phase is active.

If you try to set a parameter that belongs to any active execution context, the requested parameter is not modified.

Important: *Do not plug/unplug the USB connector during the execution phases.*

Table 5. CLI commands list

Command name	Command string	Note
CS1 - Generic commands		
help	help	Lists all registered commands with brief usage guidelines, including the list of the applicable parameters
info	info	Shows the firmware details and version
uid	uid	Shows the STM32 UID
reset	reset	Resets the MCU system
CS2 - AI specific commands		
start	start <phase>	Starts an execution phase according to its execution context. The available execution phases are: • **neai_learn** to learn the normal conditions using the NanoEdge AI library for anomaly detection • **neai_detect** to start the detection of the anomalies using the NanoEdge AI library for Anomaly Detection • **neai_class** to start the classification of asset behaviours using the NanoEdge AI library for N-Class Classification • **neai_extrapolate** to start the extrapolation of the target value using the NanoEdge AI library for Extrapolation
neai_init	neai_init	(Re)initializes the NanoEdge AI model and forgets any learning made on the device. Used at the beginning or to reinitialize the NanoEdge AI anomaly detection to start from scratch at any time
neai_set	neai_set <parameter> <value>	Sets a NanoEdge AI specific parameter. The valid parameters and values are: • **threshold**: integer [0, 100] • **signals**: integer [0, MAX_SIGNALS] • **timer**: integer [0, MAX_TIME_MS] • **sensor**: integer [0, 1]
neai_get	neai_get <parameter>	Displays the value of the NanoEdge AI parameter. The valid parameters are • **threshold** • **signals** • **timer** • **sensor** • **all**
CS3 - Sensor configuration commands		
sensor_info	sensor_info	Gets a list of all supported sensors and their ID
sensor_set	sensor_set <id> <parameter> <value>	Sets the value of a parameter for a sensor with the sensor ID provided in **id**. The settable parameters are: • **FS**: full scale • **ODR**: output data rate • **enable**: active [1 for true (active), 0 for false (disactive)]
sensor_get	sensor_get <id> <parameter>	Gets the value of a parameter for a sensor with the sensor ID provided in **id**. The valid parameters are: • **enable**: active [true (active), false (inactive)] • **ODR**: output data rate • **ODR_List**: list of supported ODRs • **FS**: full scale • **FS_List**: list of supported FSs • **all**: all sensor configurations

Note: The learning phase can be stopped automatically through a timer or signals setting. However, by default, both are initialized to zero. Therefore, you need to press the escape button to stop the learning.

5.2.4 Configuring the sensors

Through the CLI interface, you can configure the supported sensors for sensing and condition monitoring applications. The list of all the supported sensors can be displayed on the CLI console by entering the command `sensor_info`. This command prints the list of the supported sensors along with their IDs. The user can configure these sensors using these IDs. The configurable options for these sensors include:

- **enable**: to activate or deactivate the sensor
- **ODR**: to set the output data rate of the sensor from the list of available options
- **FS**: to set the full-scale range from the list of available options

Note: The best sensor setup is the one used during the data acquisition phase.

The current value of any of the parameters for a given sensor can be printed using the $ sensor_get <sensor_id> <param> command.

The information about the sensor can be printed using the $ sensor_get <sensor_id> all command.

The values for any of the available configurable parameters can be set through the $ sensor_set <sensor_id> <param> <val> command.

5.3 Using the STEVAL-PROTEUS with hardware buttons and LEDs

5.3.1 Button-operated mode

The button-operated mode for the FP-AI-PDMWBSOC2 allows running the application on the STEVAL-PROTEUS even in the absence of the CLI console and Bluetooth® Low Energy connection.

In this mode, you can control the sensor node through the user button instead of the interactive CLI console or STBLESensor. The default values for the node parameters and settings for the operations during the auto mode are provided in the firmware. Based on these configurations the learning, detecting and classifying phases can be started and stopped through the user button on the node.

Note: The auto-mode setup provides the IIS3DWB active with the last FS selected (by default, it is 16 G).

5.3.2 Buttons description

The button-operated mode can work with or without the CLI and STBLESensor. It is fully compatible and consistent with the others UI.

The supporting hardware for this function pack version has three buttons:

1. the user button, the only button usable by the software
2. the reset button, connected to the STM32 MCU reset pin
3. the power button, connected to the power management

There are also four LEDs:

1. the red LED, controlled by software
2. the green LED, controlled by software
3. the blue LED, controlled by software
4. the yellow LED, controlled by hardware, indicates the charging status when powered through a USB cable

Therefore, the basic user interactions for button-operated operations can be done through one button (user) and three LEDs (red, green, and blue).

5.3.3 User button operation and purpose

In the extended autonomous mode, you can trigger one of the following execution phases:

- *idle*: the system is waiting for a command (libraries are not running)
- *neai_learn*: sensor data are transmitted to the NanoEdge AI library to train the model
- *neai_detect*: sensor data are transmitted to the NanoEdge AI library to detect anomalies
- *neai_class*: sensor data are transmitted to the NanoEdge AI library to classify asset behaviours

To control the mentioned phases, there are three different button press modes for the user button.

Table 6. User button operation

Press type	Description	Action
Short press	The button is pressed for less than 200 ms and released	Stops the running execution phase if any, otherwise it starts the N-Class Classification library execution
Long press	The button is pressed for more than 200 ms and released	Starts the learning phase of Anomaly Detection library
Double press	A succession of two short presses	Starts the detecting phase of Anomaly Detection library

5.3.4 LEDs operation and purpose

The onboard LEDs indicate the status of the current execution phase, assuming a specific configuration.

Table 7. Red and green LEDs for NEAI phases

NEAI lib phase	Red LED	Green LED
Idle	Off	Off
Idle trained	Off	Off
Learning	Off	Blink
Detecting	On, if an anomaly occurs	On, if no anomaly occurs
Classifying	Off	Off
Extrapolating	Off	Off

Table 8. Blue LED for Bluetooth® Low Energy connectivity

Bluetooth® Low Energy connection to a smartphone	Blue LED
Connected	Slow blink
Not connected	Fast blink

The LED status allows you to know the state of the sensor node even when the USB is not plugged and/or the smartphone is not connected to the board.

Revision history

Table 9. **Document revision history**

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

Contents

List of tables

List of figures