life.augmented

Quick Start Guide

STM32Cube function pack for STEVAL-PROTEUS1 for AI anomaly detection, classification and extrapolation based on AzureRTOS

FP-AI-PDMWBSOC2

Version 2.0.0 (25, Sept 2024)

Agenda

1. Hardware and Software overview
2. Setup & Demo Examples
3. Documents & Related Resources
4. STM32 Open Development Environment: Overview

life.augmented

1- Hardware and Software overview

life.augmented

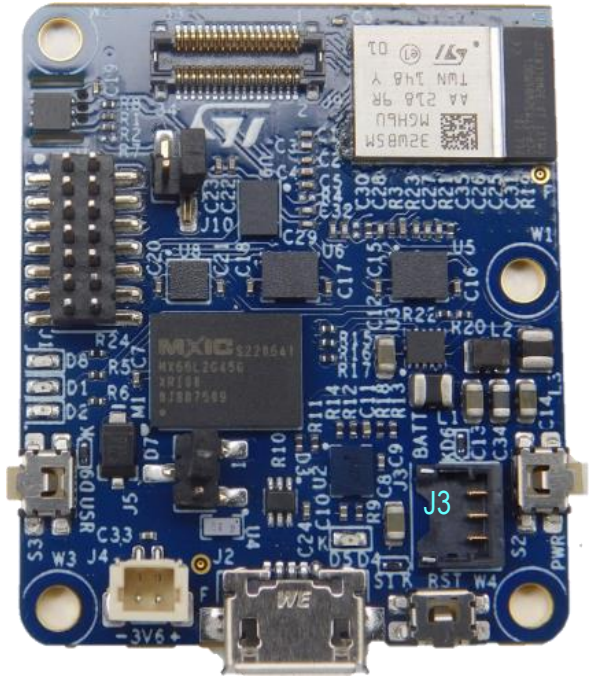
STEVAL – PROTEUS1
Hardware Overview

Industrial sensor evaluation kit for condition monitoring based on 2.4 GHz STM32WB5MMG module

The STEVAL-PROTEUS1 is an evaluation tool designed for temperature and vibration monitoring, based on a 2.4 GHz multiprotocol wireless SoC to address machine or facility condition monitoring for industrial applications. All components are mounted exclusively on the top side of the PCB to ensure an easy mounting on other equipment.

Key Features

- Kit content: the **STEVAL-PROTEUS** main board, LiPo battery 3.7 V, 480 mAh, plastic case and screws
- **STEVAL-PROTEUS**: STM32WB5MMG - ultra-low-power module, dual core 32-bit Arm Cortex-M4 MCU 64 MHz, Cortex-M0+ 32 MHz for real-time radio layer, with 1 Mbyte of flash memory, 256kbyte SRAM, and 2.4GHz RF supporting Bluetooth® Low Energy 5, 802.15.4, Zigbee 3.0, and Thread
- **IIS3DWB** - ultra-wide bandwidth up to 6 kHz, low noise, 3-axis digital accelerometer
- **ISM330DHCX** - iNEMO inertial module with machine learning core and finite state machine with digital output
- **IIS2DLPC** - high-performance ultra-low-power 3-axis digital accelerometer
- **STTS22H** - low-voltage, ultra-low-power, 0.5°C accuracy I²C/SMBus 3.0 temperature sensor
- Memory & Secure: 2Gb QSPI NOR flash memory for data storage, STSAFE-A110 - secure element
- Power: **STBC02** - Li-Ion linear battery charger with LDO, **ST1PS02** - step-down converter with digital voltage selection
- HMI: 3 push-buttons (Reset, User, Power-on with battery), 4 LEDs (three user LEDs, one STBC02 LED status)
- Flexible power supply options - LiPo battery, USB power, and primary battery
- Connectors: SWD connector for debugging and programming capability, 34-pin expansion connector compliant with STMOD+



Latest info available at:
https://www.st.com/en/evaluation-tools/steval-proteus1.html

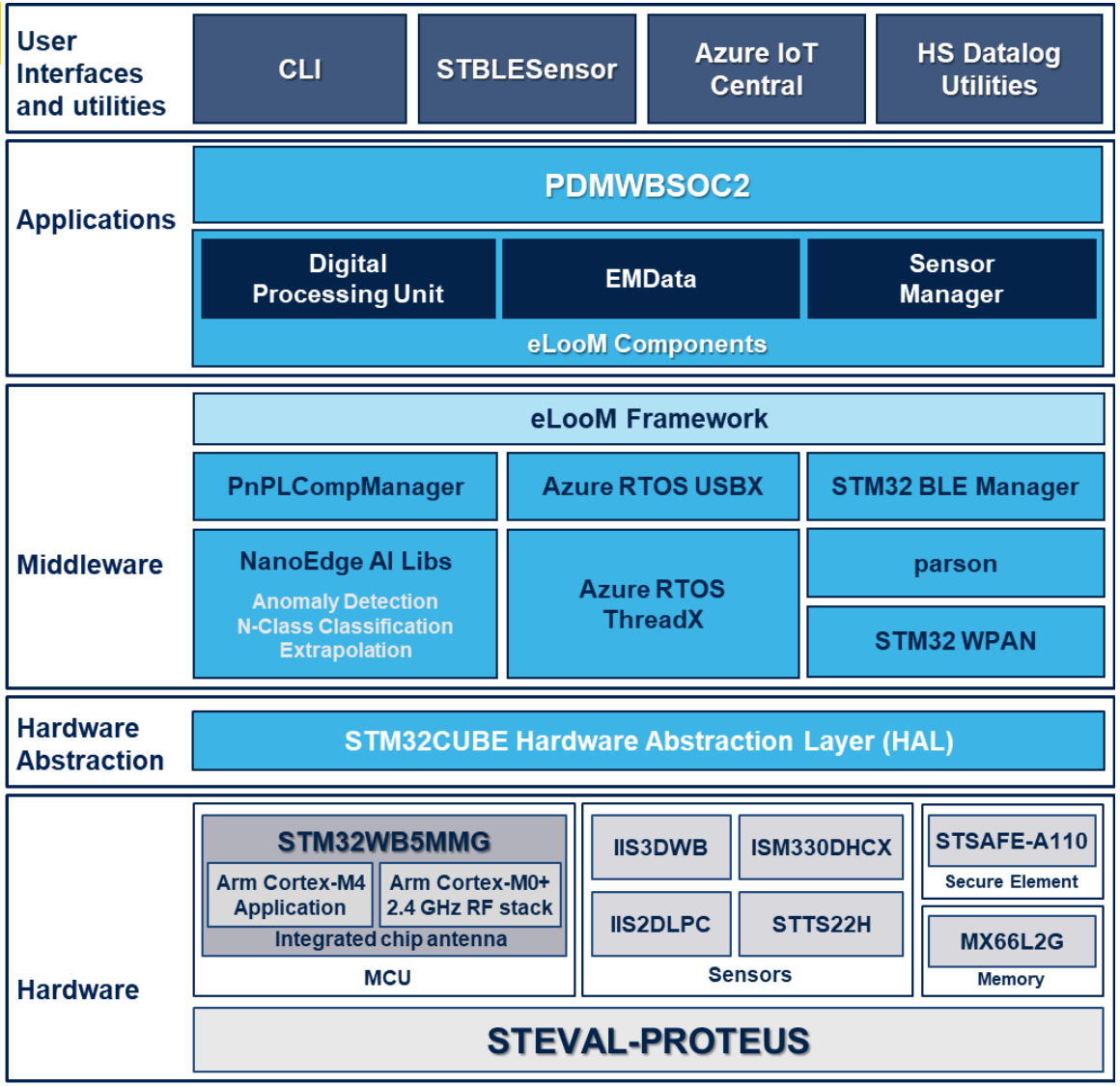
Contains:
FCC ID: YCP-STM32WB5M001
IC: 8976A-STM32WB5M01

life.augmented

FP-AI-PDMWBSOC2
FW Architecture

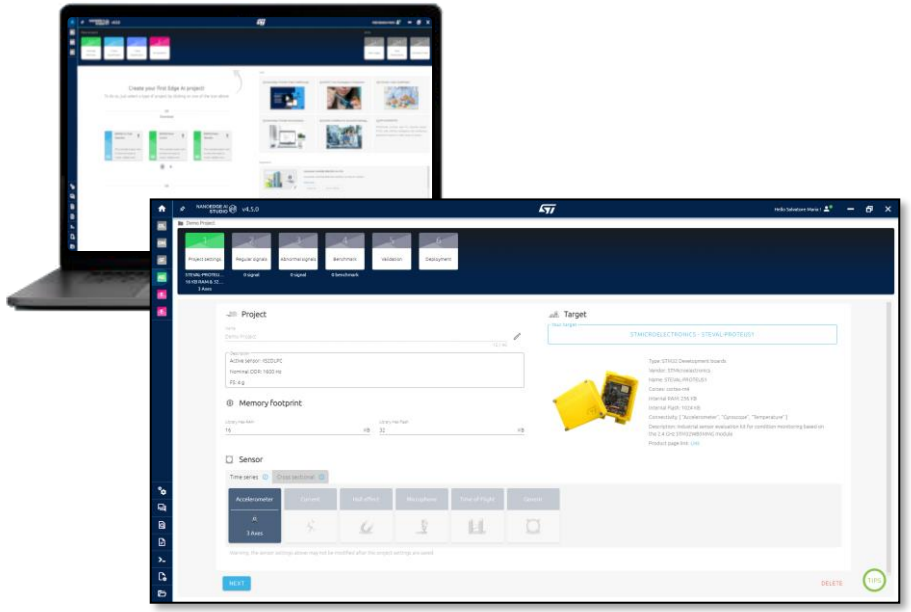
Key Features

- Firmware to enable predictive maintenance applications based on ML algorithms, it has the capability to store raw data supporting maximum sensors data rate, it buffers data according to customizable size and process them with anomaly detection (binary classifier), n-class classification (multi-class classifier) and extrapolation (regression) models generated through NanoEdgeAIStudio tool.
- Firmware to develop a WPAN sensor node for predictive maintenance applications, it sends processing results, receive commands and exchange setting parameters via BLE.
- Compatible with NanoEdgeAIStudio solution to enable AI-based applications.
- Compatible with STBLESensor app (Android and iOS) to enable AI and sensors setting, and firmware update via fast FUOTA, STBLESensor can works as a bridge to an Azure IoT Central dashboard.
- AzureRTOS: ThreadX, 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 NanoEdgeAI formats.



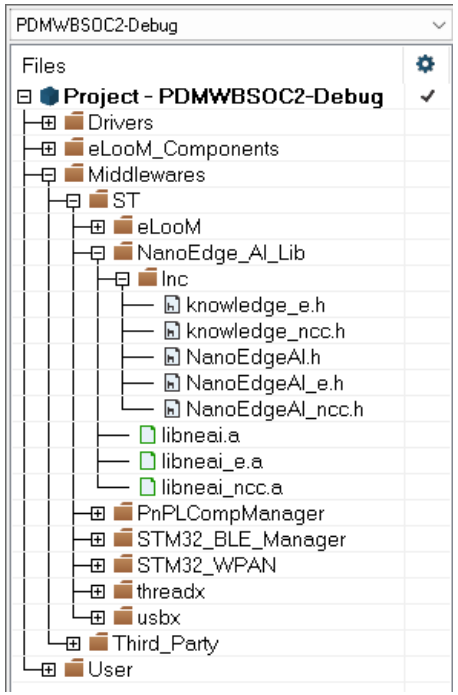
FP-AI-PDMWBSOC2
Ecosystem

Complete turnkey solution from datalog to anomaly detection, classification and extrapolation status on mobile app and up to cloud

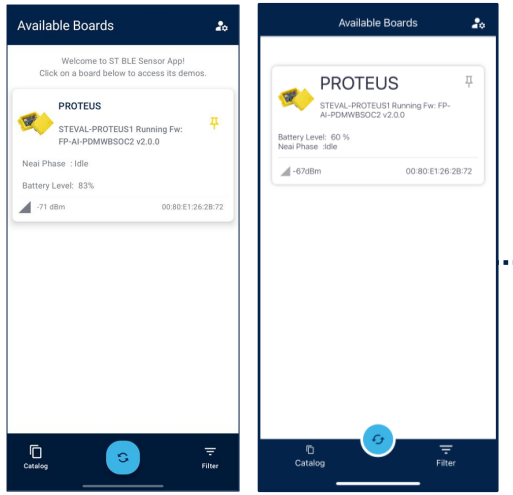


STEVAL-PROTEUS1 supported
in [NanoEdgeAIStudio](#) **[v4.5.0]**

NANOEDGE AI STUDIO



KEIL Tools by ARM · IAR SYSTEMS · STM32 CubeIDE



Azure
IoT Central

life.augmented

2- Setup & Demo Examples

life.augmented

Setup & Demo Examples
Software prerequisites

- **STM32CubeProgrammer Software**
  - Download and install STM32CubeProgrammer
- **FP-AI-PDMWBSOC2**
  - Copy the .zip file content into a folder on your PC. The package will contain source code example (Keil, IAR, STM32CubeIDE) based on **STEVAL-PROTEUS**
- **ST BLE Sensor**
  - Application for Android (from v5.2.4) / iOS (from v5.2.3) to download from Play Store / App Store

STEVAL-PROTEUS1 kit is not preprogrammed with **FP-AI-PDMWBSOC2**
To update the firmware, please follow the instructions available in slide 13-16

Setup & Demo Examples
Hardware prerequisites

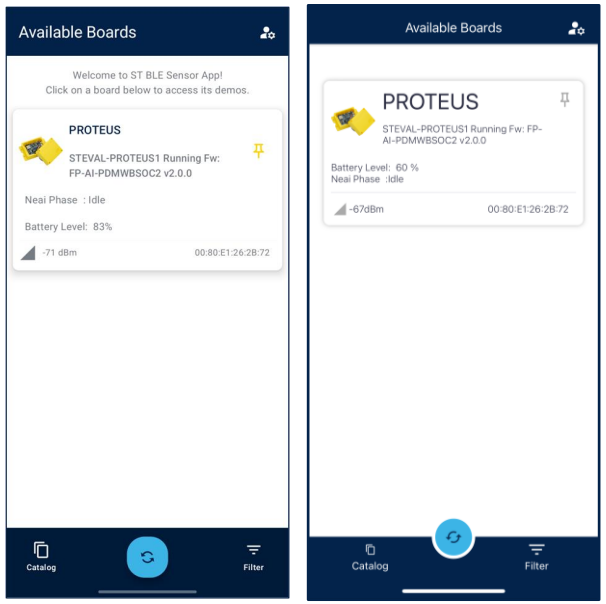
Recommended

- 1 STEVAL-PROTEUS1 evaluation kit
- 1 Laptop/PC with Windows 10 or 11
- 1 USB-A to USB-microB cable
- 1 smartphone with STBLESensor App (Android or IOS)

Optional (just for debugging and programming)

- 1 STLINK-V3MINIE
- 1 USB-A to USB-C cable to connect the STLINK-V3MINIE

STEVAL-PROTEUS



ST BLE Sensor App

USB micro-B

USB Type-C

STLINK-V3MINIE

2.1- Setup Overview

life.augmented

STEVAL-PROTEUS1
Unboxing

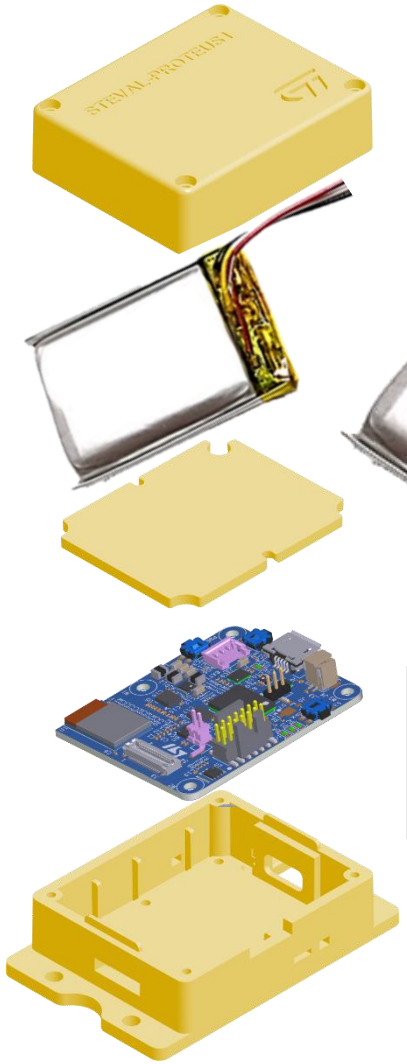
5
Lock the top case to the bottom one with the last four screws included in the kit.

4
Plug the battery connector on J3

3
Put the cover on the battery and close it using two screws.

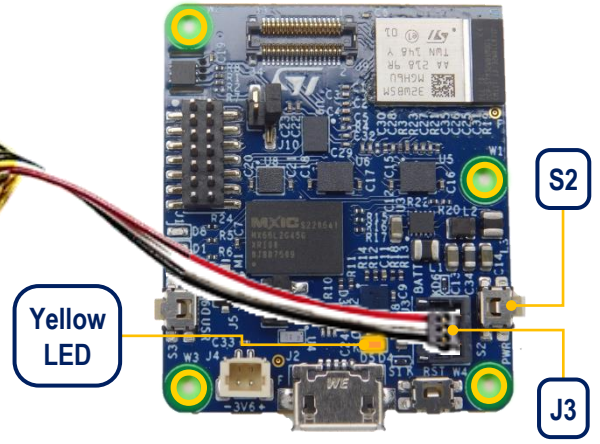
2
Put the Li-Po battery in the top case, insert the battery cable into the dedicated hole.

1
Fix the main board to the case bottom with the four screws included in the kit.



Use a needle to access the buttons behind the holes on the sides of the case.

Power ON
Push and hold S2 until the yellow LED turns on (HW Featrure)



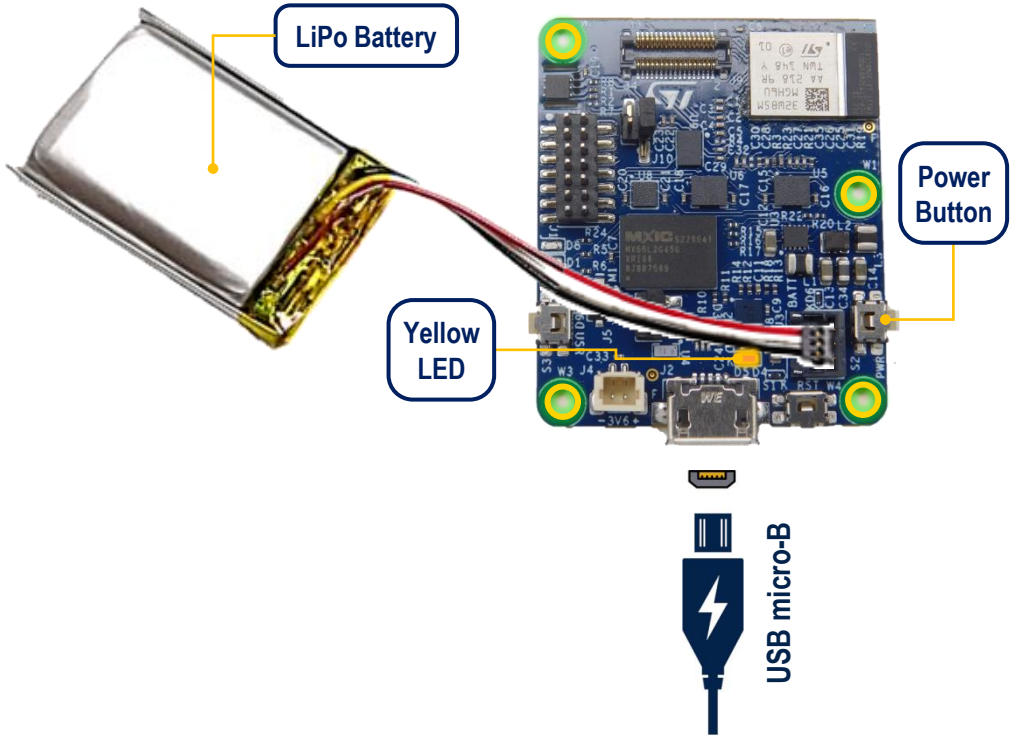
Power OFF
Push and hold S2 until the Yellow LED turns off (SW feature)

life.augmented

STEVAL-PROTEUS1
Power ON/OFF

- **Battery operated only** (no USB cable):
  - **Power ON**: push and hold the power button until the yellow LED turns on (~3 sec).
  - **Power OFF**: push and hold the power button until the yellow LED turns off (~3 sec).

- **Plugged mode** (USB cable)
  - **Power ON**: when USB is plugged-in, the STEVAL-PROTEUS is always on. It doesn't matter if the battery is present or not.
  - **Power OFF:** unplug the USB cable and, if the battery is connected, act as described above.



life.augmented

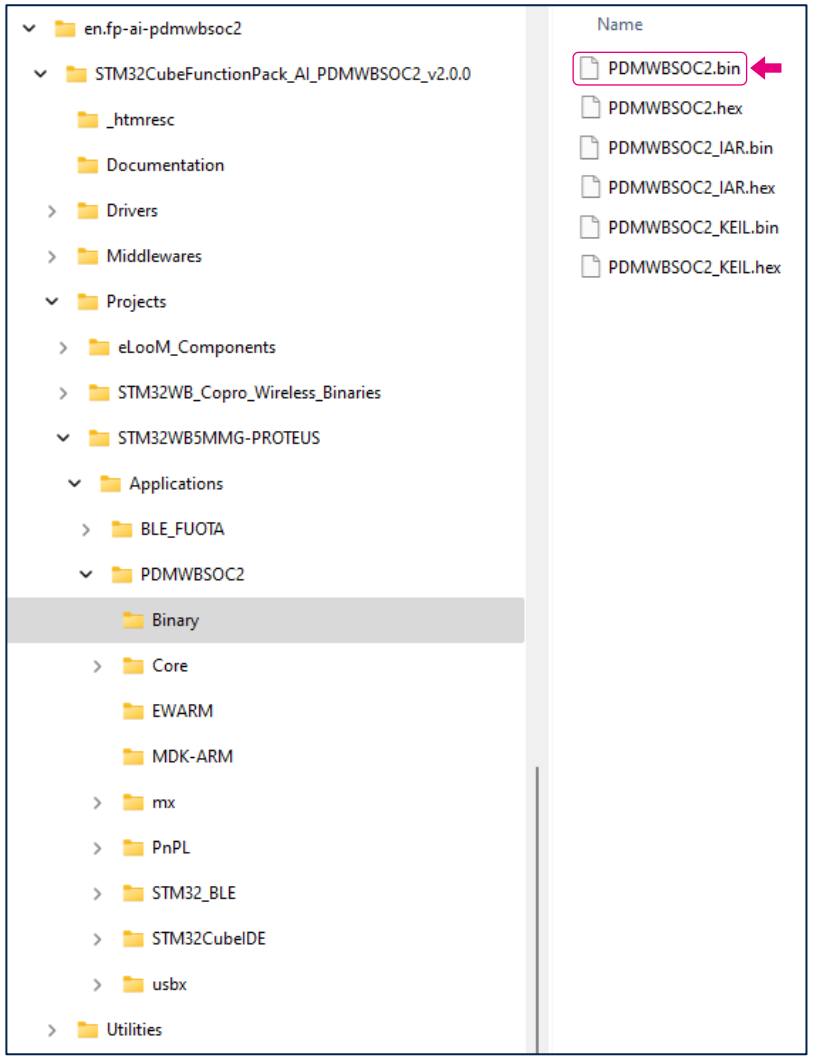
STEVAL-PROTEUS Setup
Firmware update by STLINK 1/2

STEVAL-PROTEUS evaluation board is pre-programmed with another default application so, it must be update downloading the **FP-AI-PDMWBSOC2** application.

The easiest way is to use the **pre-compiled binary** provided in the package in the following folder:
Projects\STM32WB5MMG-PROTEUS\Applications\PDMWBSOC2\Binary

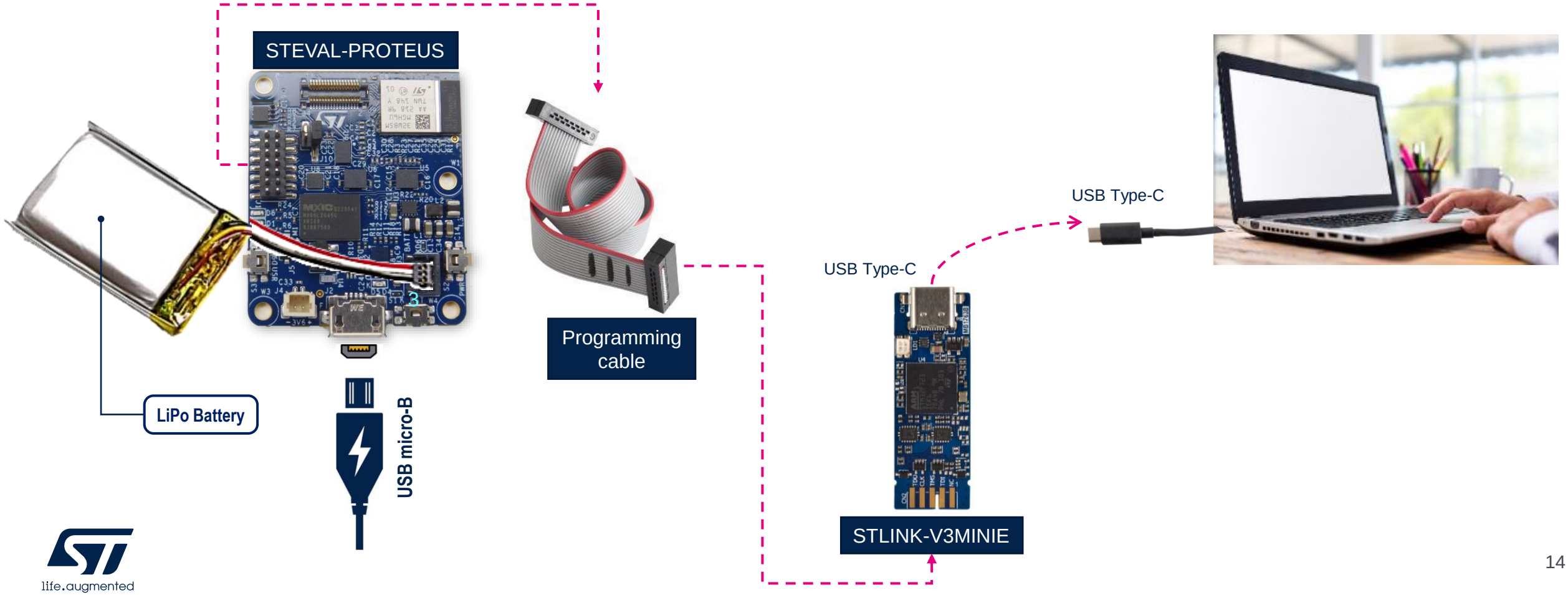
To update the firmware the user can choose one of the following procedure:

- Save the same binary file (in *.bin format) in your mobile device and upgrade firmware by FUOTA using STBLESensor Mobile App.
- Connect the STEVAL-PROTEUS board to the STLINK-V3MINIE programmer, and then use the STM32CubeProgrammer tool.



STEVAL-PROTEUS Setup
Firmware update by STLINK 2/2

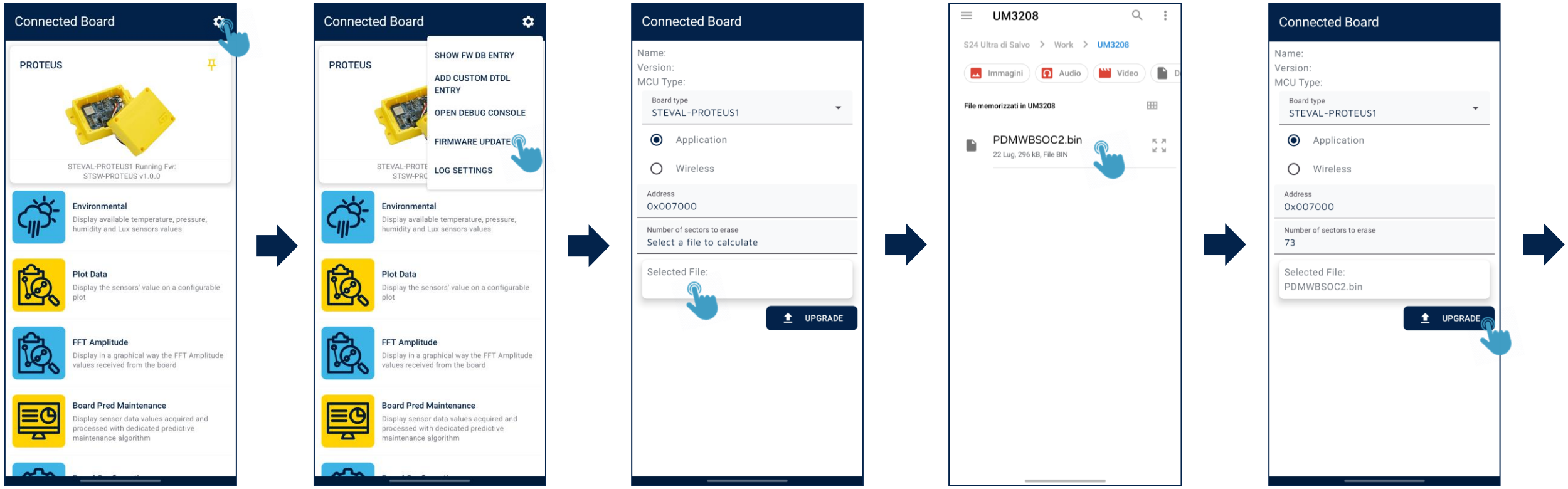
- ❑ Power the board using Battery or USB-microB connector
- ❑ Follow the connection by cables, as shown in the picture below



STEVAL-PROTEUS Setup
Firmware update by FUOTA 1/2

How to re-program the STEVAL-PROTEUS by FUOTA:

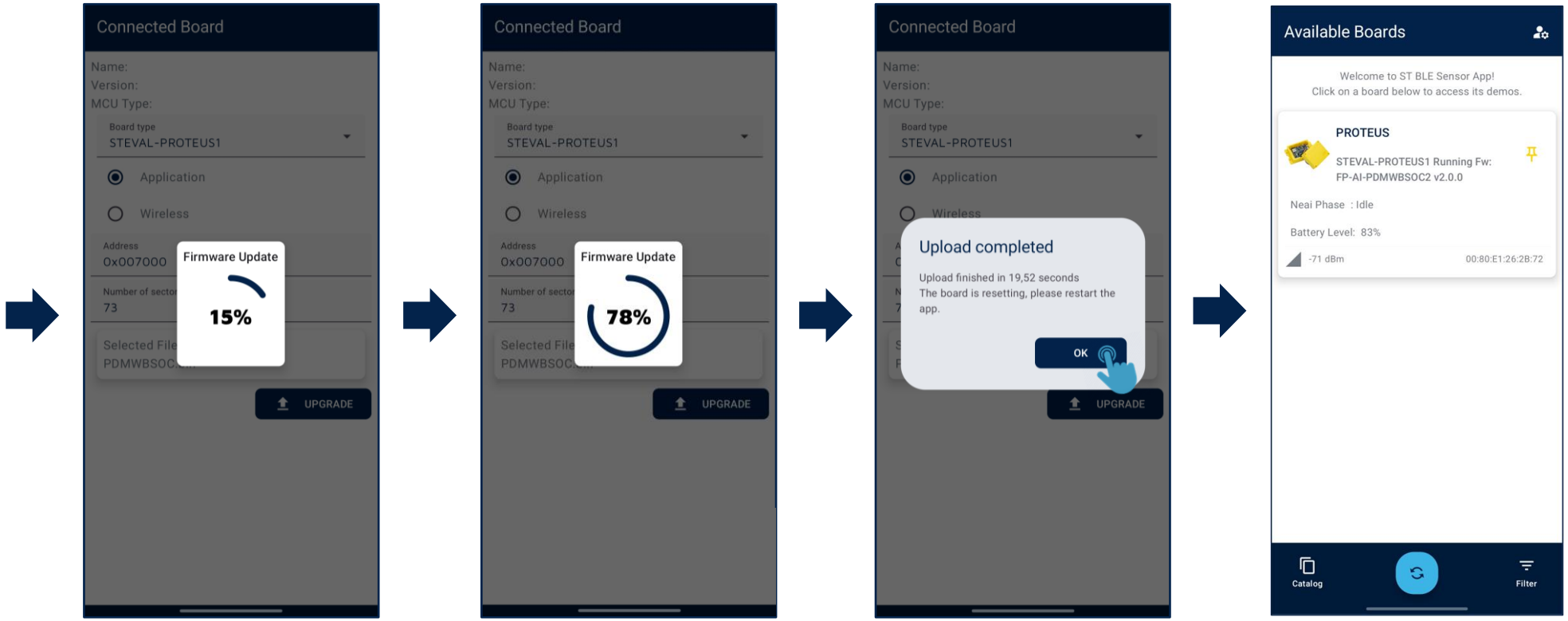
- Install and launch STBLESensor mobile App, connect board and follow below steps



(*) Download in your smartphone the binary file from FP package

STEVAL-PROTEUS Setup
Firmware update by FUOTA 2/2

- How to re-program the STEVAL-PROTEUS by FUOTA

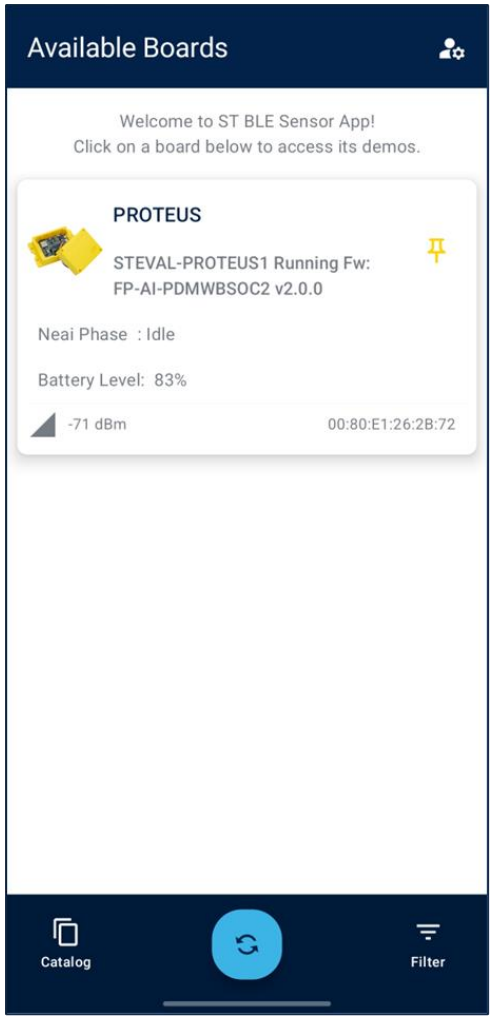


2.2- PDMWBSOC2 Application:

How to use the ST BLE Sensor App

life.augmented

Discovery View of ST STBLESensor App



After opening the App, you'll see the list of available boards to which can connect.

For each board are available:

- FW running name (**FP-AI-PDMWBSOC2 v2.0.0**)
- NEAI phase (idle, idle trained, learning, detecting, classifying, extrapolating)
- Battery level (0-100%)
- Status Icon, ✓ normal or ⚠ anomaly in case of anomaly detection phase

Until you aren't connected to the board, blue LED blinks slowly

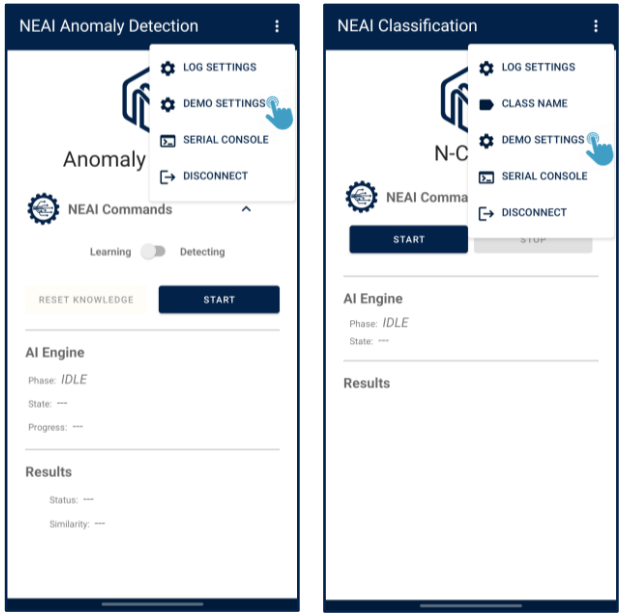
When you are connected to the board, blue LED blinks quickly

life.augmented

The proper workflow

Open ST BLE Sensor App & Connect Board

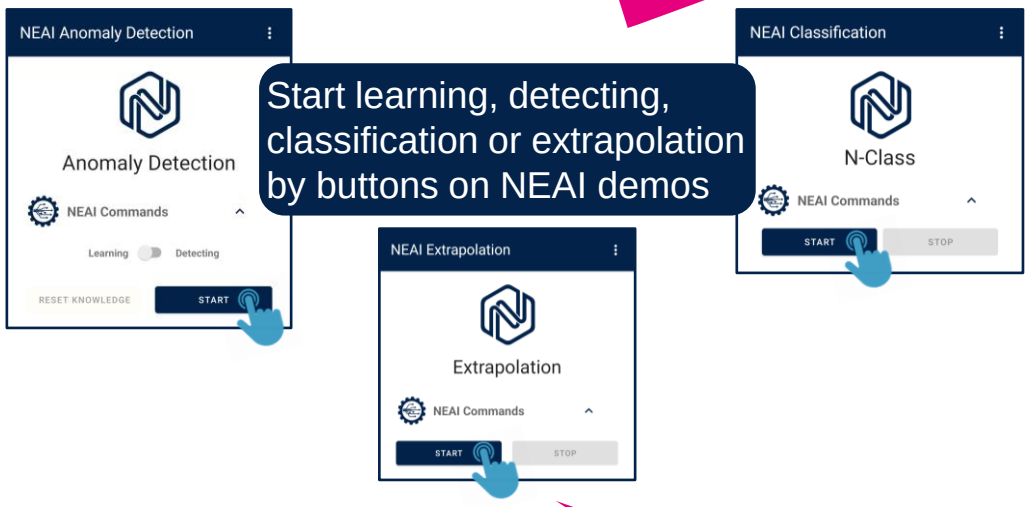
Tap on the gear icon to customize library parameters and sensor setup (*)



* This step is optional but remember that by default:

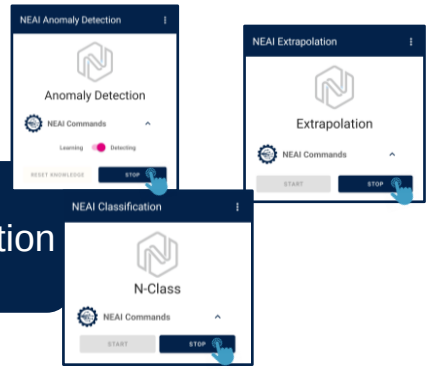
- ISM330DHCX is active with ODR = 6667 Hz and FS = 16 G
- Learning phase will end when you'll push stop button (*Time/Signals* parameter is initialized to zero)

It's strongly recommended to setup your sensor according to dataset used to generate NEAI library



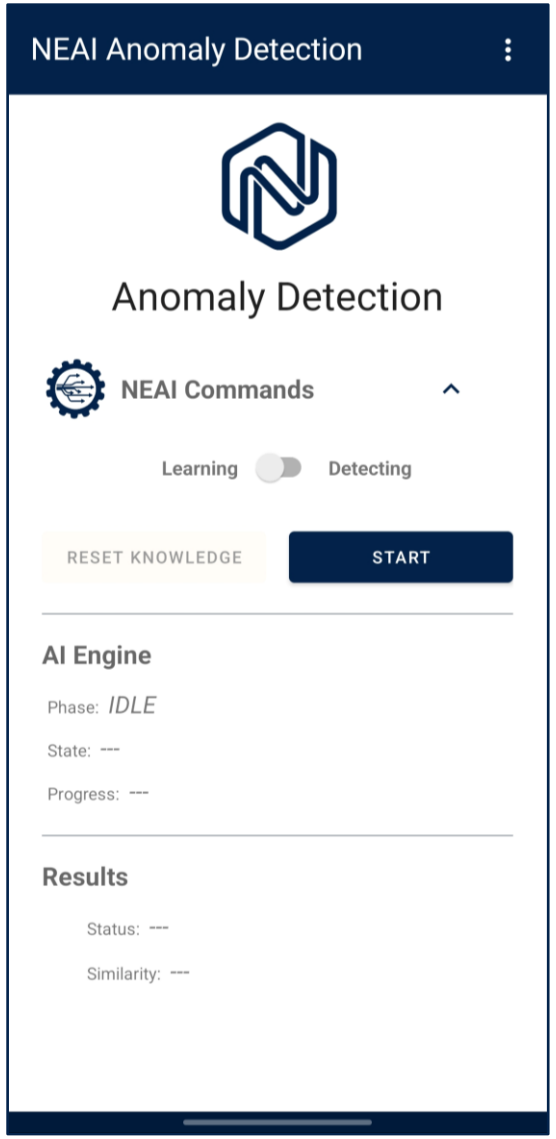
Start learning, detecting, classification or extrapolation by buttons on NEAI demos

Stop learning, detecting, classification and extrapolation by buttons on NEAI demos



life.augmented

Start learning phase
Three simple steps



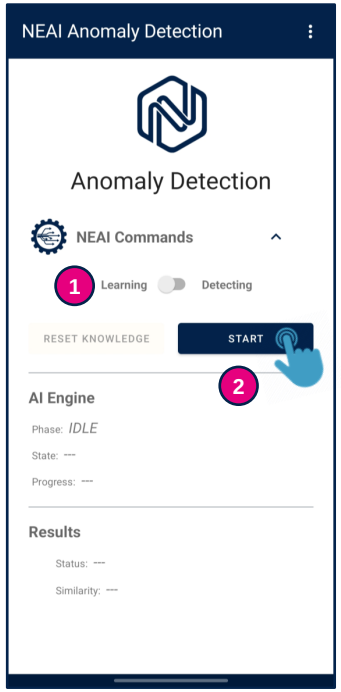
Through this demo you can monitor NanoEdgeAI AD library status and also start/stop learning and detecting phases. To start your first learning follow the steps below:

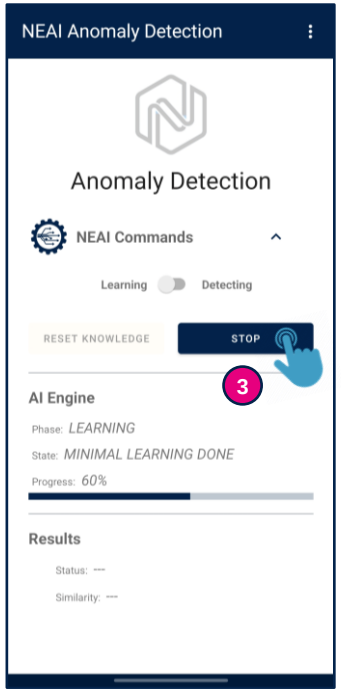
1. Move the commands switch on the left to enable **Learning**

2. Push start button

3. Push stop button when you want.

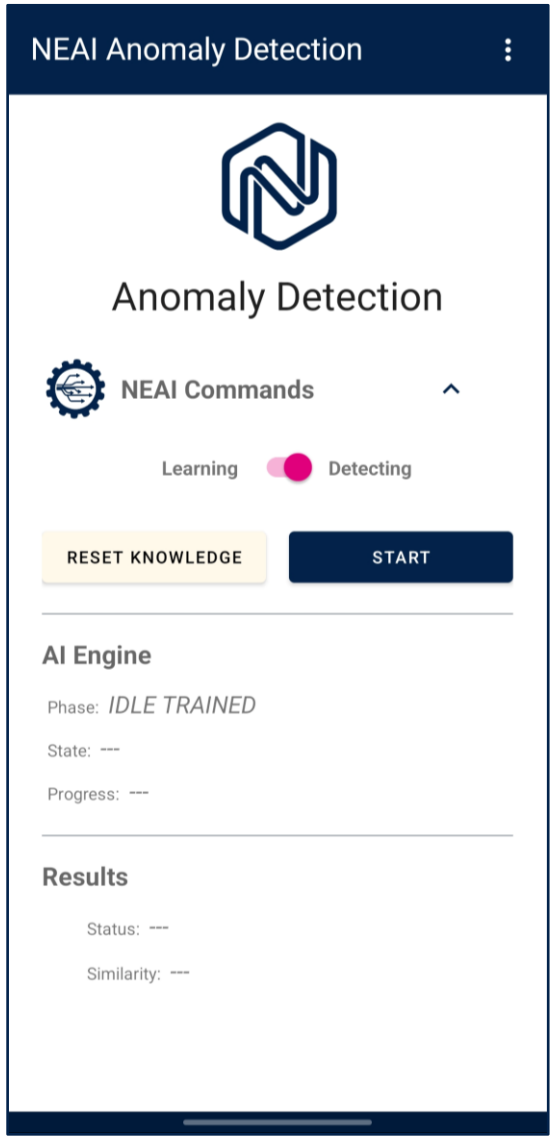
*(When the processed signals are more than required from the NEAI-AD library itself the state changes in "**MINIMAL LEARNING DONE**")*





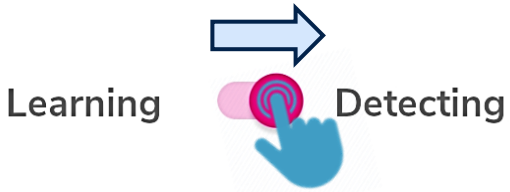
life.augmented

Start detecting phase
Three simple steps



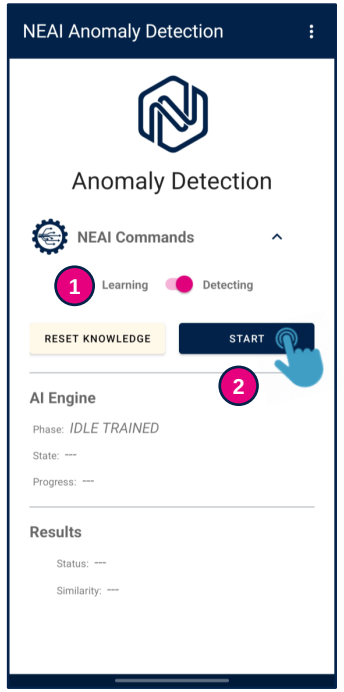
Through this demo you can monitor NanoEdgeAI AD library status and also start/stop learning and detecting phases. To start your first detection, follow the steps below:

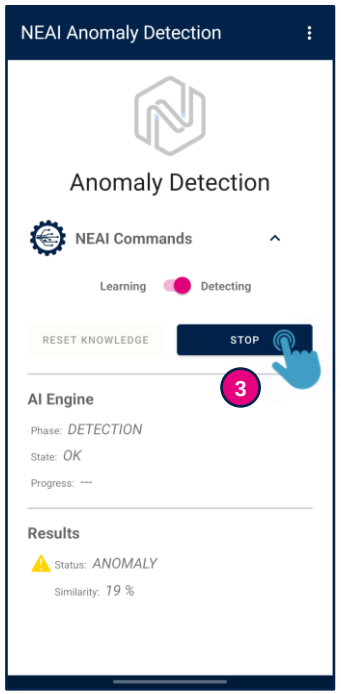
1. Move the commands switch on the right to enable **Detection**



2. Push start button

3. Push stop button when you want





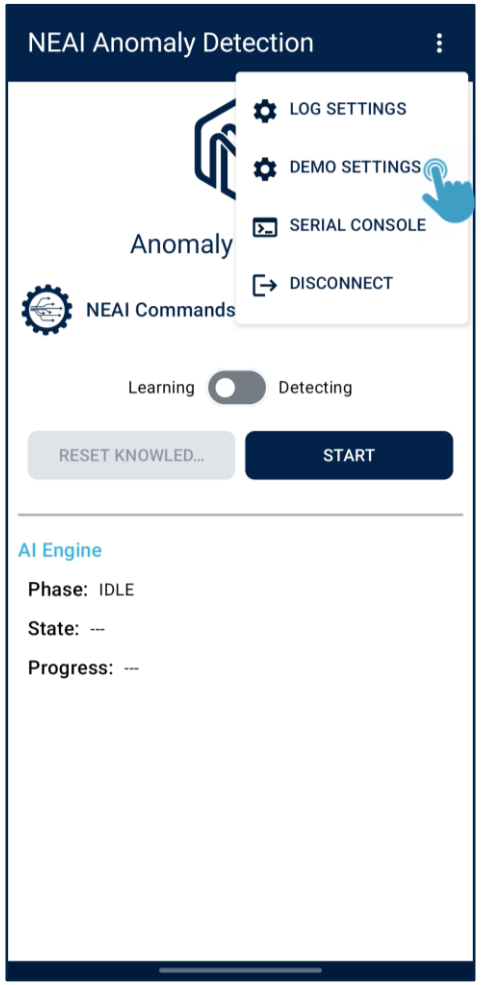
Detection started
Status Normal
green LED ON

Detection started
Status Abnormal
red LED ON

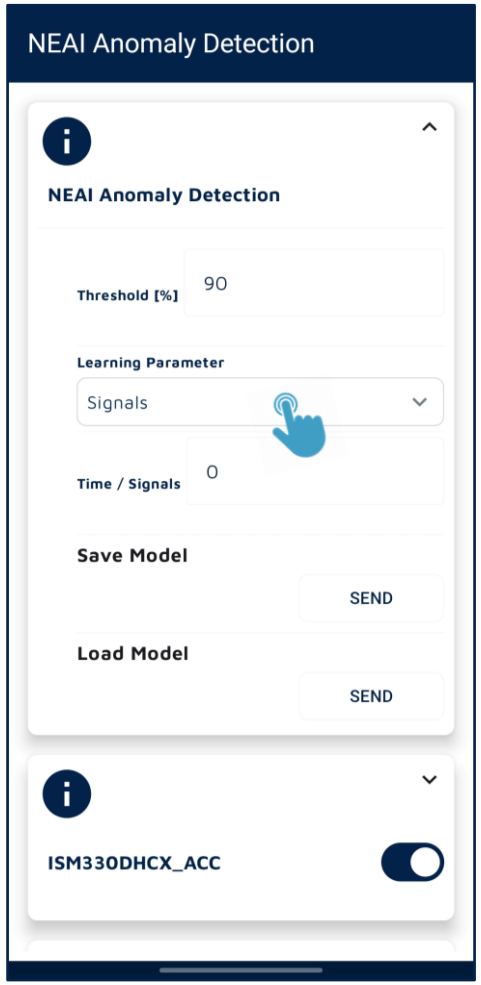
Remember that detecting phase will end only when you push stop button

life.augmented

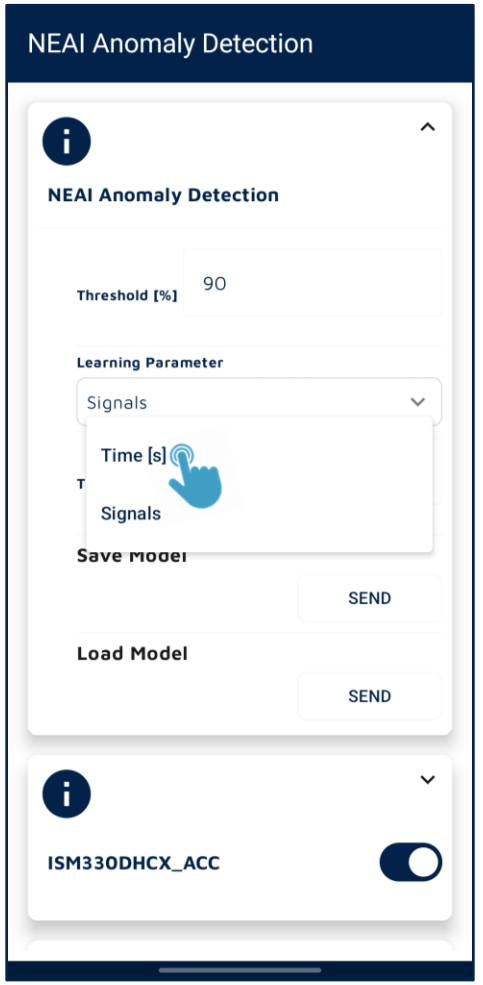
How to set learning phase time 1/2



First of all tap on gear icon to open the setting page



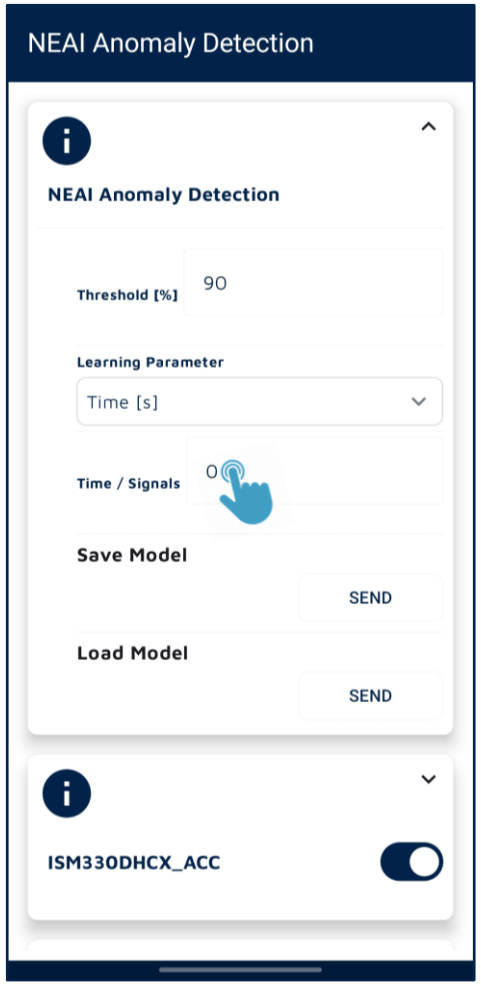
Tap on learning parameter and select *Time [s]* option



1 2 3

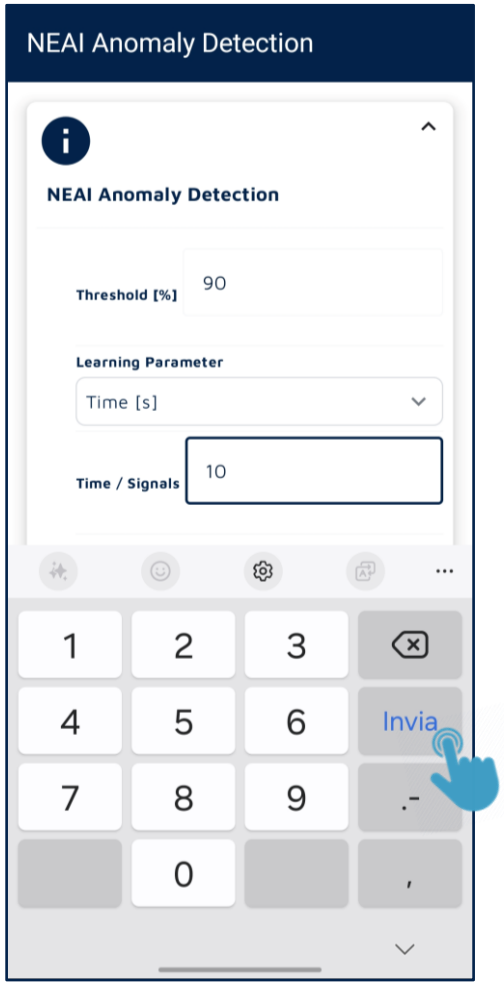
life.augmented

How to set learning phase time 2/2



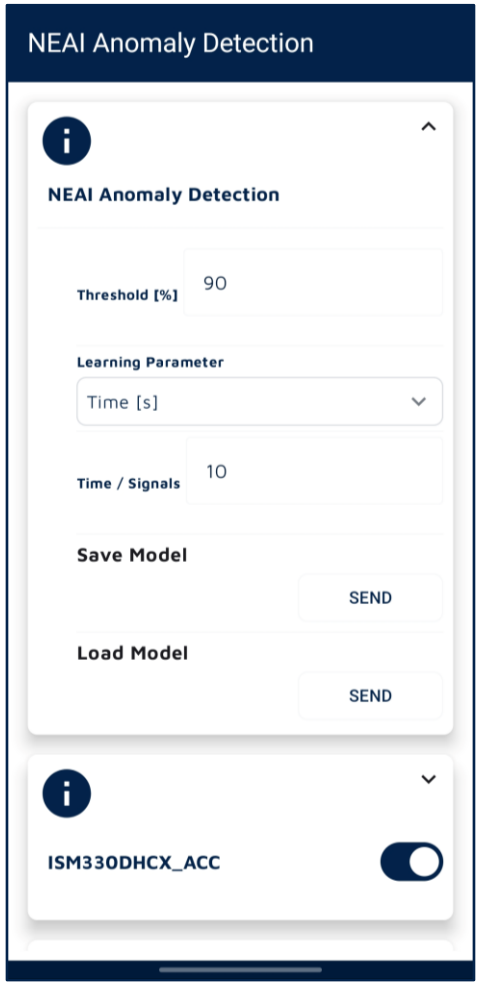
4

Tap on *Time/Signals* parameter



5

Enter the desired learning duration

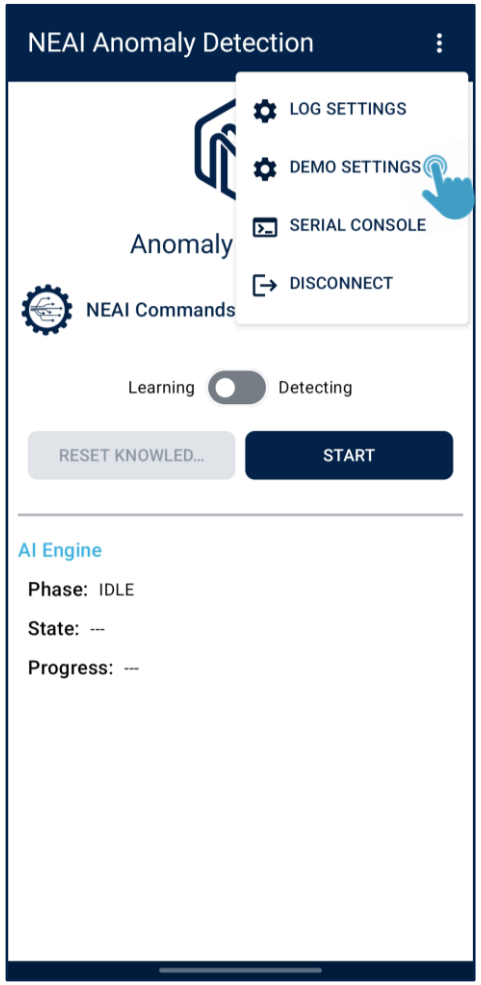


6

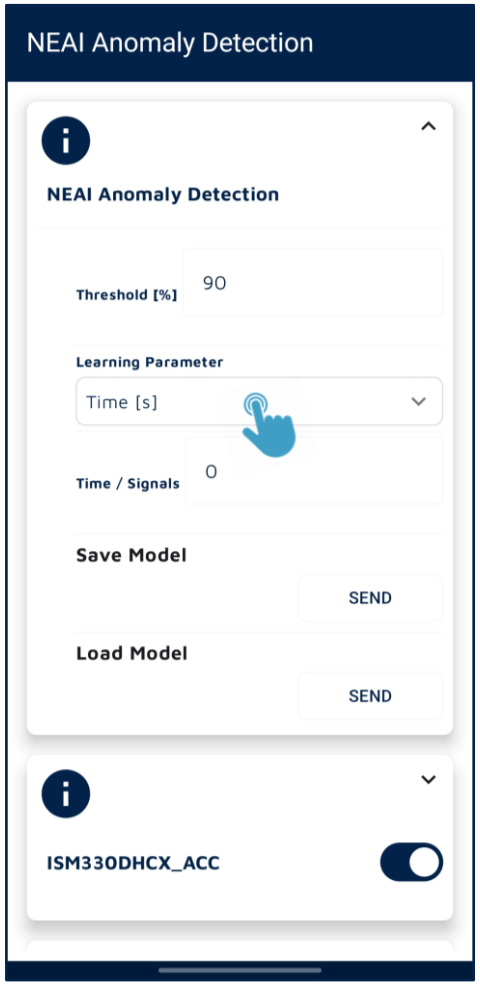
END

life.augmented

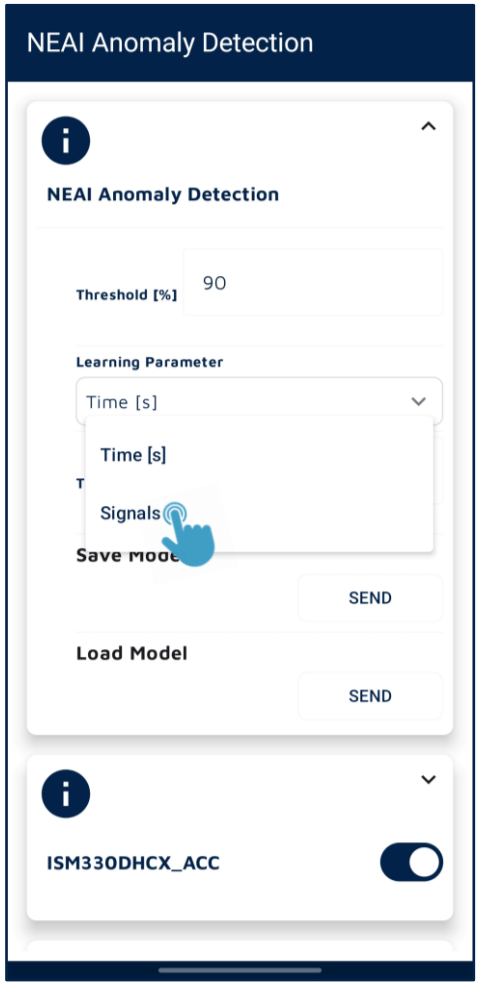
How to set signals to learn 1/2



First of all tap on gear icon to open the setting page



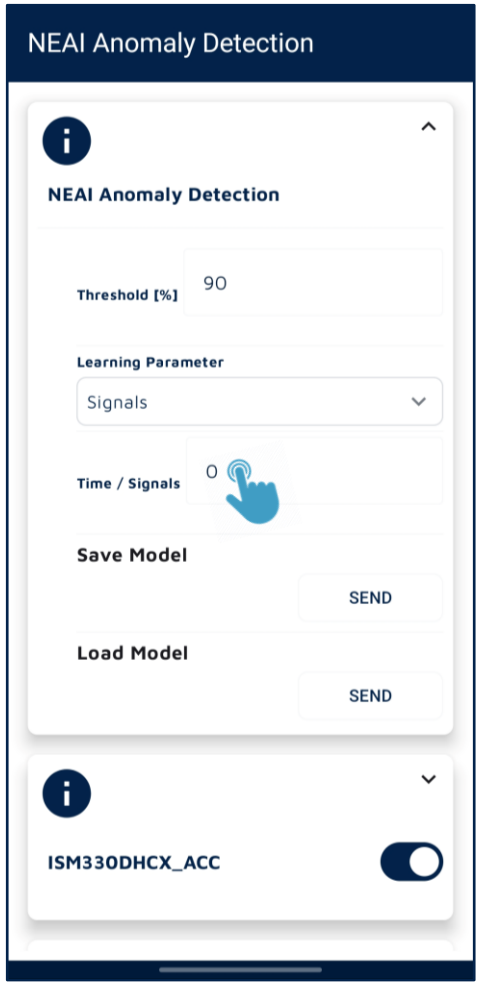
Tap on learning parameter and select *Signals* option



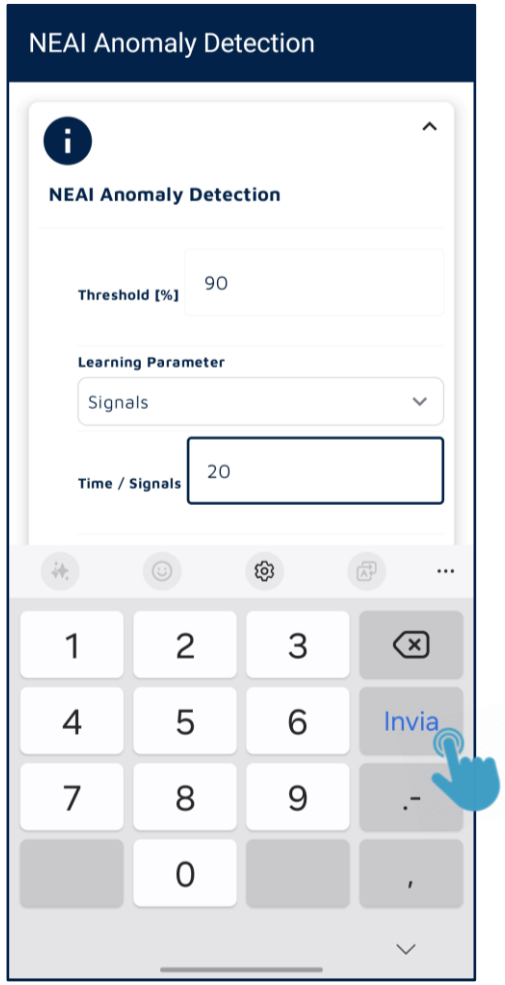
1 2 3

life.augmented

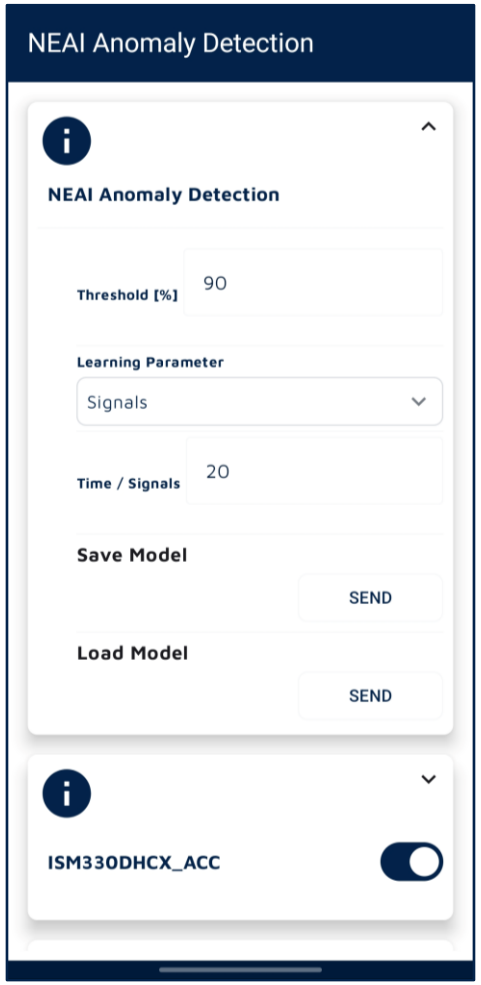
How to set signals to learn 2/2



Tap on *Time/Signals* parameter



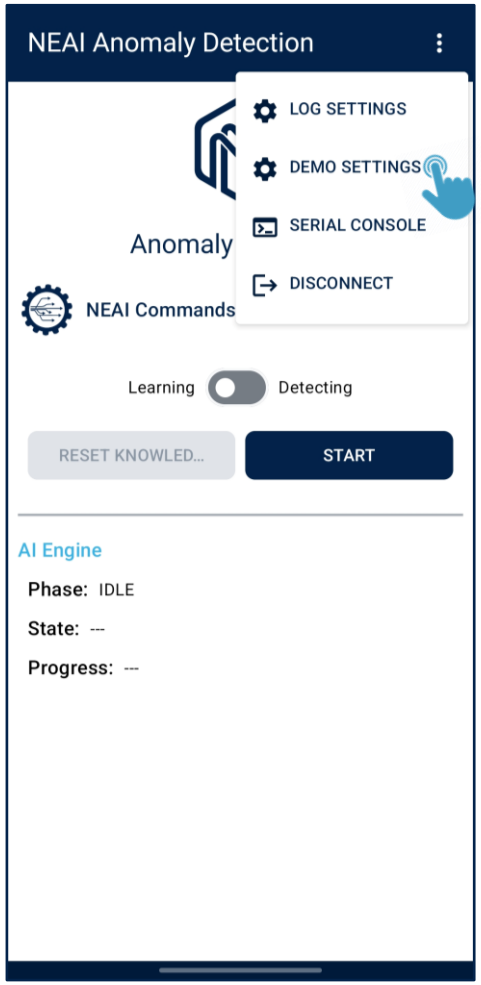
Enter the desired number of signals to learn



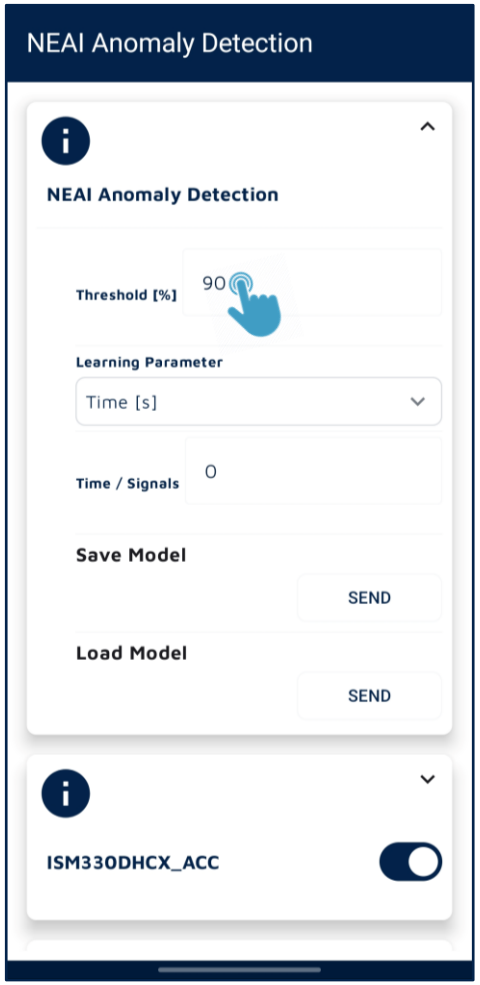
4 5 6 END

life.augmented

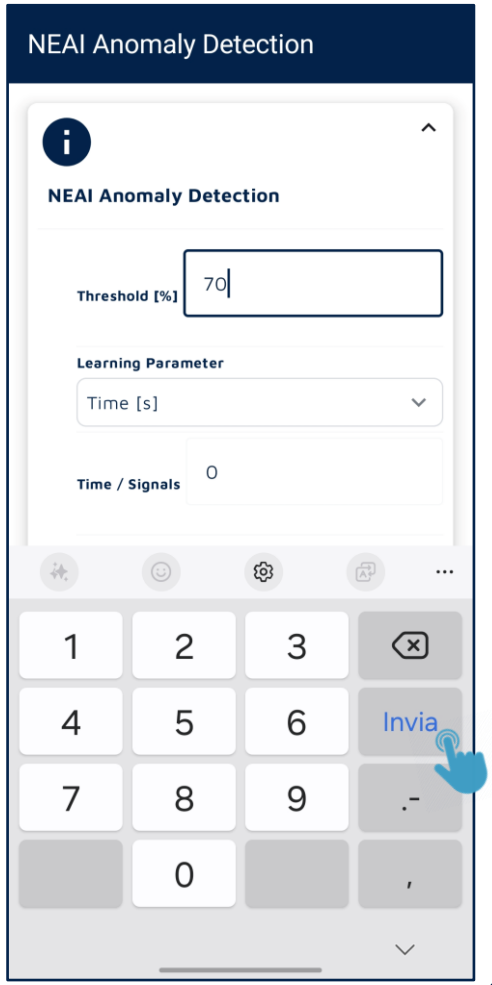
How to set AD library parameters



First of all tap on gear icon to open the setting page



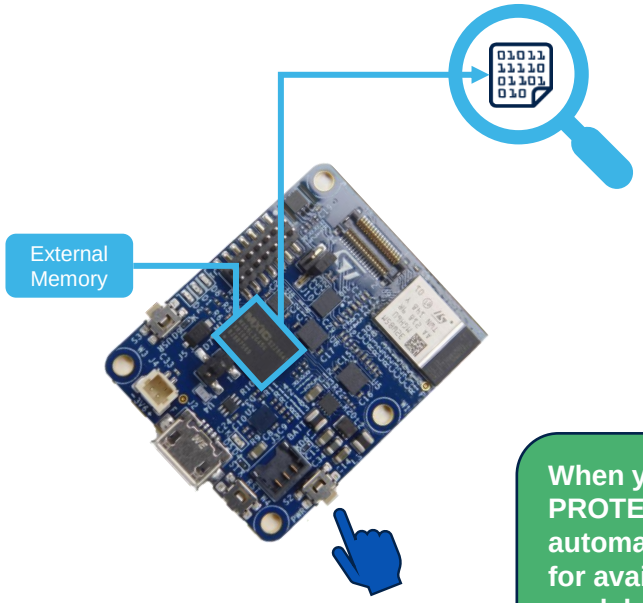
Tap on *Threshold* parameter to set it



1 2 3 END

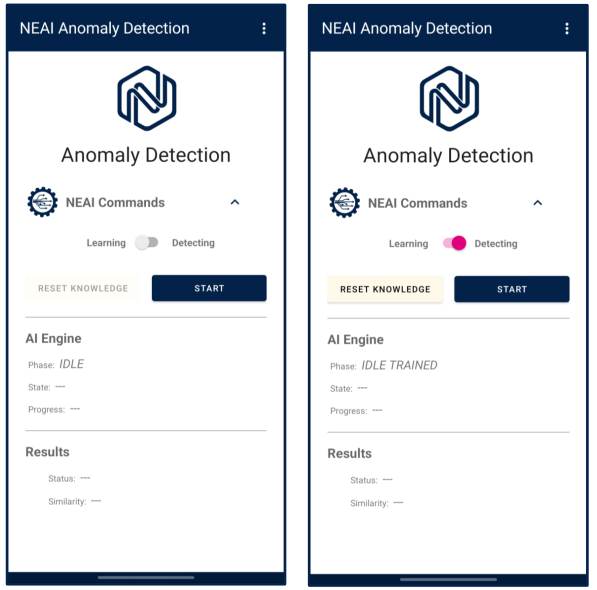
life.augmented

How to use save/load NEAI AD model 1/2



When you turn-on your PROTEUS, firmware automatically search for available NEAI AD model stored in the external flash memory

1



if no model was found, phase will be *IDLE*

before detect anomalies, you need to start a learning phase

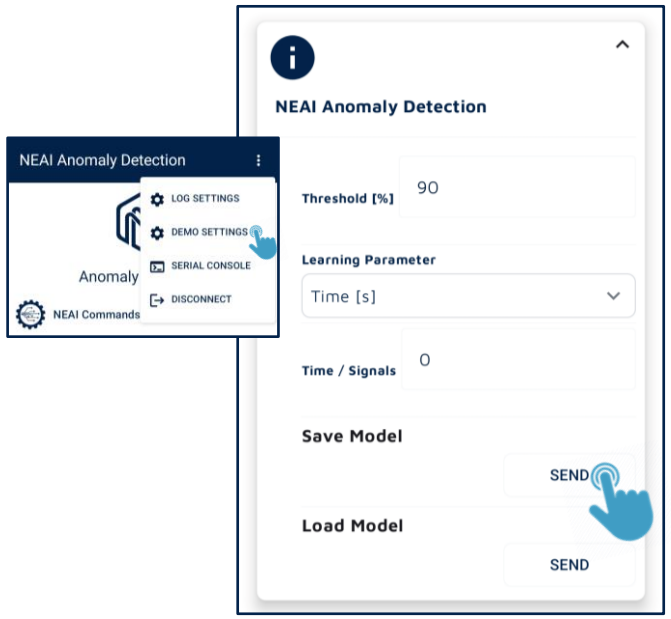
if a model was found, phase will be *IDLE TRAINED*

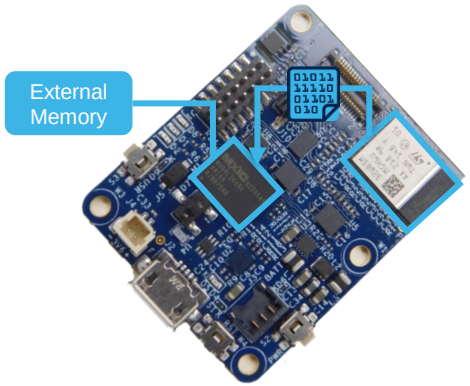
you are ready to start a detecting phase

2

life.augmented

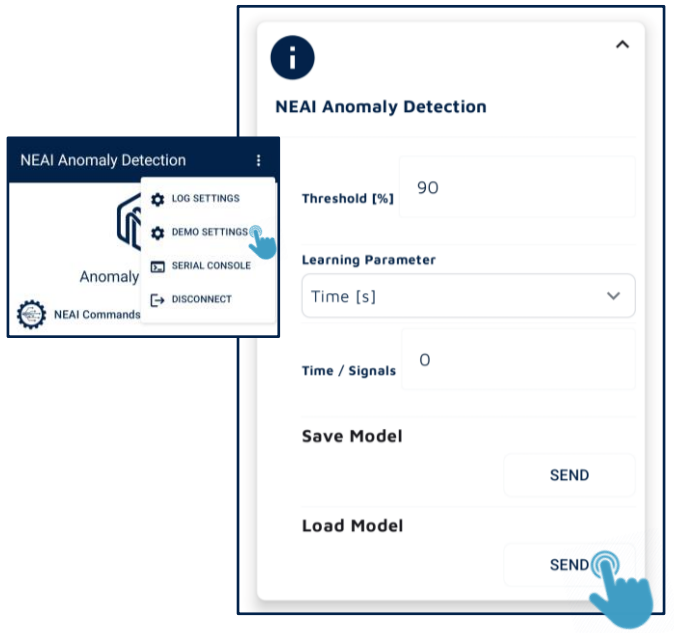
How to use save/load NEAI AD model 2/2

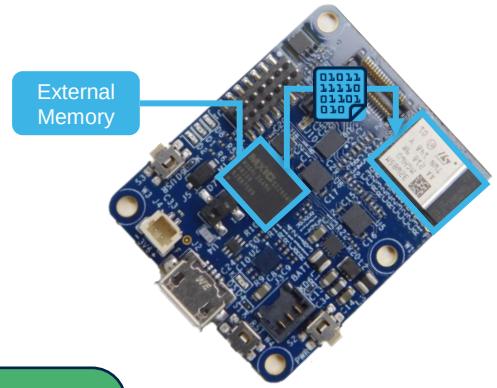




to avoid to lose your AD model after shutting down, you can open setting page and send the *Save Model* command

3





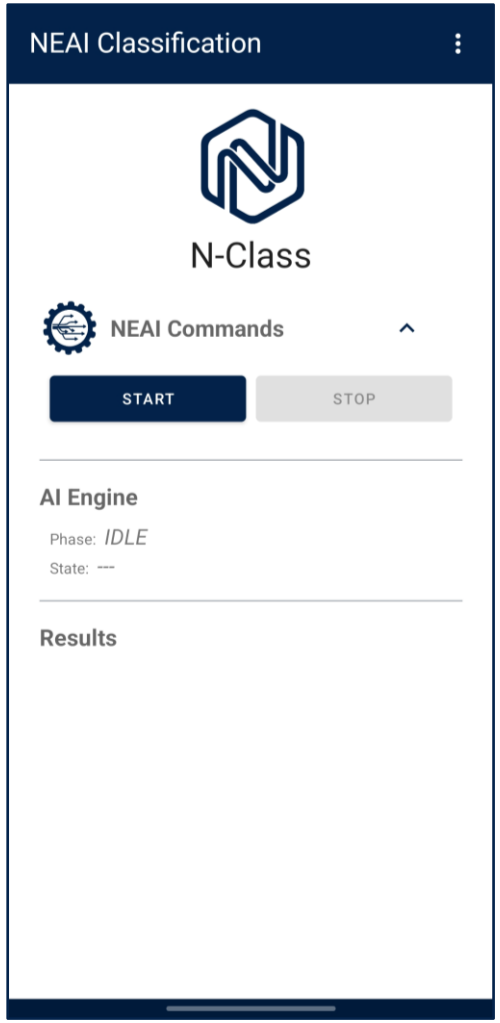
If you are dissatisfied with results coming from last learning, you can restore the model saved in the external flash memory by *Load Model* command

4

END

life.augmented

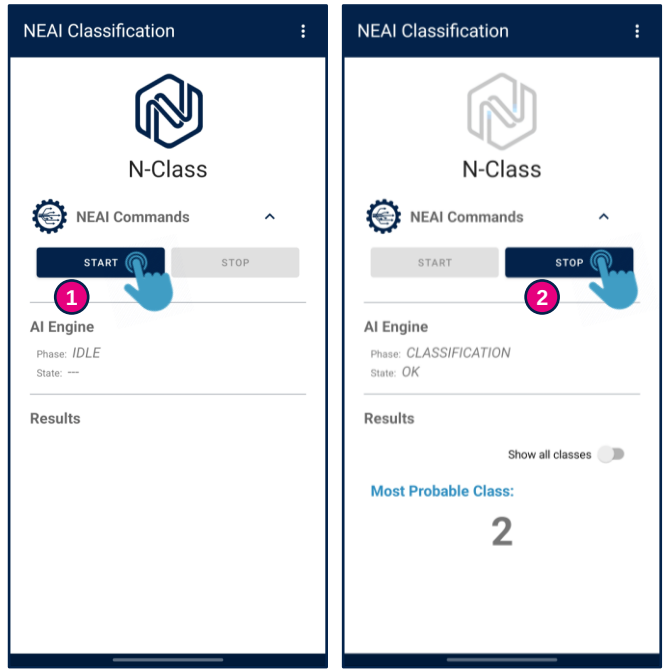
Start classification phase
Two simple steps



Through this demo you can monitor NanoEdgeAI NCC library status and also start/stop classifying phase. To start your first classification, follow the steps below:

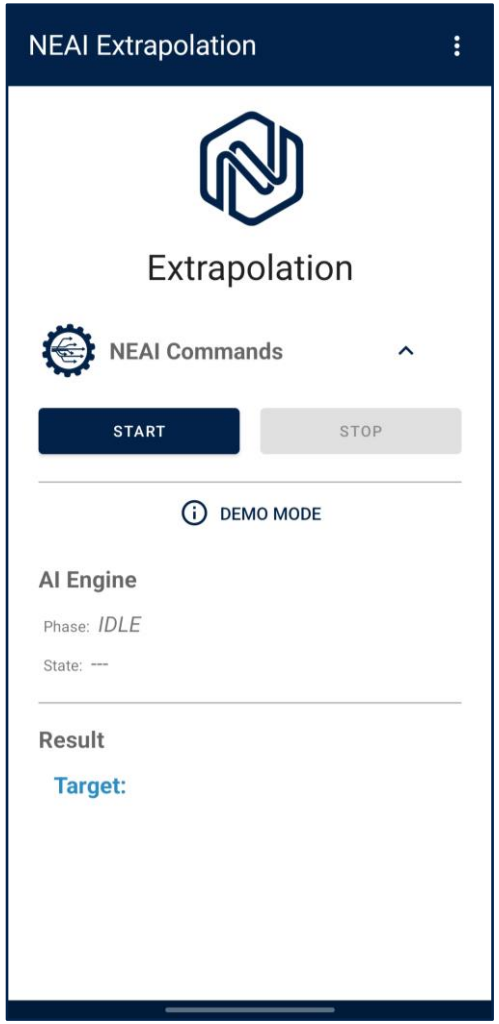
1. Push start button 

2. Push stop button when you want 



Remember that classifying phase will end only when you push stop button

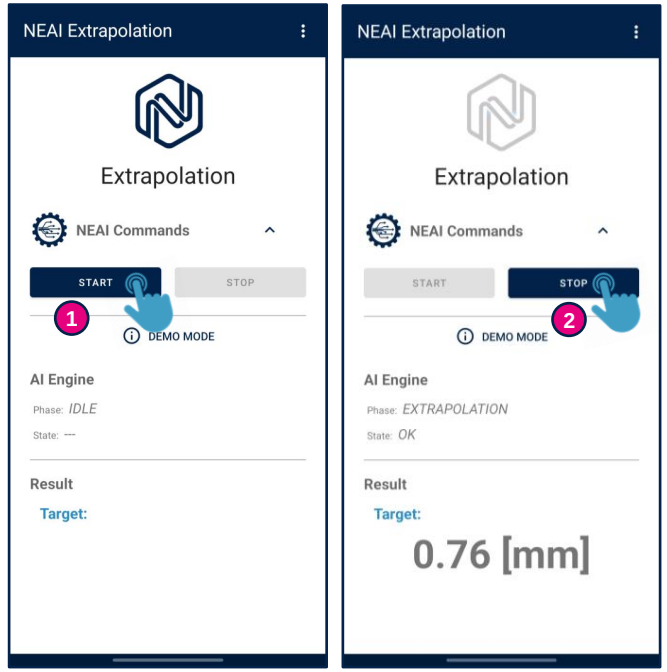
Start extrapolation phase
Two simple steps



Through this demo you can monitor NanoEdgeAI E library status and also start/stop extrapolation phase. To start your first extrapolation, follow the steps below:

1. Push start button 

2. Push stop button when you want 



Remember that extrapolation phase will end only when you push stop button

life.augmented

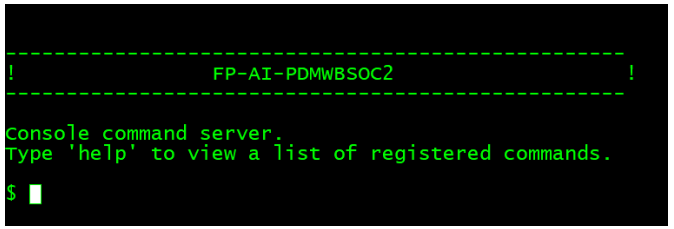
2.3- PDMWBSOC2 Application:

How to use CLI Terminal Console

life.augmented

The proper workflow

Plug USB connector and open Tera Term *



Enter *set_neai timer [t]* or *set_neai signals [n]* to customize your learning phase **

Start learning, detecting, classifying or extrapolating by specific commands

Stop learning, detecting, classifying or extrapolating automatically or pushing escape button

** This step is optional but remember that by default:

- ISM330DHCX is active with ODR = 6667 Hz and FS = 16 G
- Learning phase will end when you'll push stop button (*Time/Signals* parameter is initialized to zero)

It's strongly recommended to setup your sensor according to dataset used to generate NEAI library

*Be careful, it's strongly recommended plug/unplug USB connector only when the library is not running: NOT plug/unplug USB during learning/detecting/classifying phase.

life.augmented

Anomaly Detection:
Start learning phase

```
-------------------------------------------------
!                FP-AI-PDMWBSOC2                !
-------------------------------------------------

Console command server.
Type 'help' to view a list of registered commands.

$ start neai_learn
NanoEdgeAI: starting learn phase...

$ NanoEdge AI: learn
CTRL: ! This is a stubbed version, please install NanoEdge AI library !
{"signal": 1, "status": "need more signals"},
{"signal": 2, "status": "need more signals"},
{"signal": 3, "status": "need more signals"},
{"signal": 4, "status": "need more signals"},
{"signal": 5, "status": "need more signals"},
{"signal": 6, "status": "need more signals"},
{"signal": 7, "status": "need more signals"},
{"signal": 8, "status": "need more signals"},
{"signal": 9, "status": "need more signals"},
{"signal": 10, "status": "success"},
{"signal": 11, "status": "success"},
{"signal": 12, "status": "success"},
{"signal": 13, "status": "success"},
{"signal": 14, "status": "success"},
{"signal": 15, "status": "success"},
{"signal": 16, "status": "success"},
```

The user has to enter the proper command to start the learning phase

start neai_learn

Learning phase started, green LED blinks

life.augmented

Anomaly Detection:
Start detection phase

```
$ start neai_detect
NanoEdgeAI: starting detect phase...

$ NanoEdge AI: detect
CTRL: ! This is a stubbed version, please install NanoEdge AI library !
{"signal": 1, "similarity": 0 %, "state": 0, "status": anomaly}
{"signal": 2, "similarity": 1 %, "state": 0, "status": anomaly}
{"signal": 3, "similarity": 2 %, "state": 0, "status": anomaly}
{"signal": 4, "similarity": 3 %, "state": 0, "status": anomaly}
{"signal": 5, "similarity": 4 %, "state": 0, "status": anomaly}
{"signal": 6, "similarity": 5 %, "state": 0, "status": anomaly}
{"signal": 7, "similarity": 6 %, "state": 0, "status": anomaly}
{"signal": 8, "similarity": 7 %, "state": 0, "status": anomaly}
{"signal": 9, "similarity": 8 %, "state": 0, "status": anomaly}
{"signal": 10, "similarity": 9 %, "state": 0, "status": anomaly}
{"signal": 11, "similarity": 10 %, "state": 0, "status": anomaly}
{"signal": 12, "similarity": 11 %, "state": 0, "status": anomaly}
{"signal": 13, "similarity": 12 %, "state": 0, "status": anomaly}
{"signal": 14, "similarity": 13 %, "state": 0, "status": anomaly}
{"signal": 15, "similarity": 14 %, "state": 0, "status": anomaly}
{"signal": 16, "similarity": 15 %, "state": 0, "status": anomaly}
{"signal": 17, "similarity": 16 %, "state": 0, "status": anomaly}
{"signal": 18, "similarity": 17 %, "state": 0, "status": anomaly}
{"signal": 19, "similarity": 18 %, "state": 0, "status": anomaly}
{"signal": 20, "similarity": 19 %, "state": 0, "status": anomaly}
{"signal": 21, "similarity": 20 %, "state": 0, "status": anomaly}
{"signal": 22, "similarity": 21 %, "state": 0, "status": anomaly}
{"signal": 23, "similarity": 22 %, "state": 0, "status": anomaly}
```

The user has to enter the proper command to start detection phase

start neai_detect

Detection started
Status Normal
green LED ON

Detection started
Status Abnormal
red LED ON

Remember that detecting phase will end only when you push escape button

life.augmented

N-Class Classification:
Start classification phase

```
$ start neai_class
NanoEdgeAI: starting classification phase...

$ NanoEdge AI: classification
CTRL: ! This is a stubbed version, please install NanoEdge AI library !
{"signal": 1, "class": Class2},
{"signal": 2, "class": Class2},
{"signal": 3, "class": Class2},
{"signal": 4, "class": Class2},
{"signal": 5, "class": Class3},
{"signal": 6, "class": Class3},
{"signal": 7, "class": Class3},
{"signal": 8, "class": Class3},
{"signal": 9, "class": Class3},
{"signal": 10, "class": Class3},
{"signal": 11, "class": Class3},
{"signal": 12, "class": Class3},
{"signal": 13, "class": Class3},
{"signal": 14, "class": Class1},
{"signal": 15, "class": Class1},
{"signal": 16, "class": Class1},
{"signal": 17, "class": Class1},
{"signal": 18, "class": Class1},
{"signal": 19, "class": Class1},
{"signal": 20, "class": Class1},
{"signal": 21, "class": Class1},
```

The user has to enter the proper command to start classification phase

start neai_class

Remember that classifying phase will end only when you push escape button

Extrapolation:
Start extrapolation phase

```
$ start neai_extrapolate
NanoEdgeAI: starting extrapolation phase...

$ NanoEdge AI: extrapolation
CTRL: ! This is a stubbed version, please install NanoEdge AI library !
{"signal": 1, "extrapolated value": 0.00},
{"signal": 2, "extrapolated value": 0.01},
{"signal": 3, "extrapolated value": 0.02},
{"signal": 4, "extrapolated value": 0.03},
{"signal": 5, "extrapolated value": 0.04},
{"signal": 6, "extrapolated value": 0.05},
{"signal": 7, "extrapolated value": 0.06},
{"signal": 8, "extrapolated value": 0.07},
{"signal": 9, "extrapolated value": 0.08},
{"signal": 10, "extrapolated value": 0.09},
{"signal": 11, "extrapolated value": 0.10},
{"signal": 12, "extrapolated value": 0.11},
{"signal": 13, "extrapolated value": 0.12},
{"signal": 14, "extrapolated value": 0.13},
{"signal": 15, "extrapolated value": 0.14},
{"signal": 16, "extrapolated value": 0.15},
{"signal": 17, "extrapolated value": 0.16},
{"signal": 18, "extrapolated value": 0.17},
{"signal": 19, "extrapolated value": 0.18},
```

The user has to enter the proper command to start extrapolation phase

start neai_extrapolate

Remember that extrapolation phase will end only when you push escape button

life.augmented

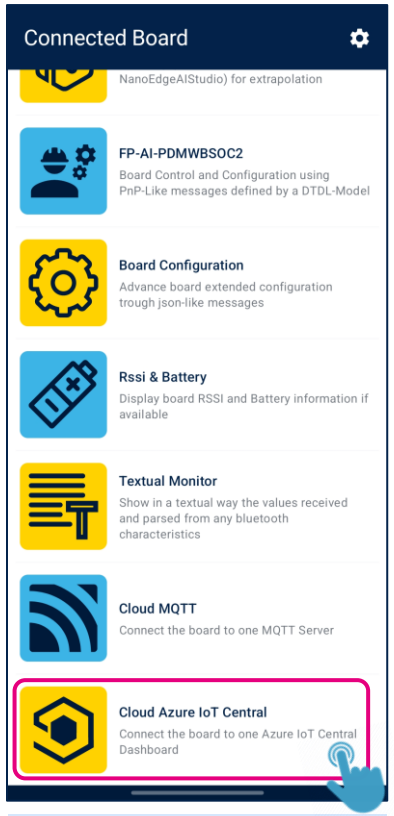
2.4- PDMWBSOC2 Application:

Azure IoT Central Cloud Service

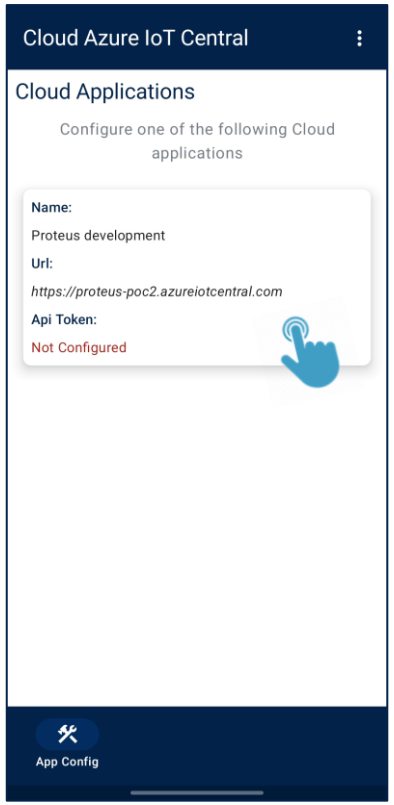
life.augmented

STBLESensor App

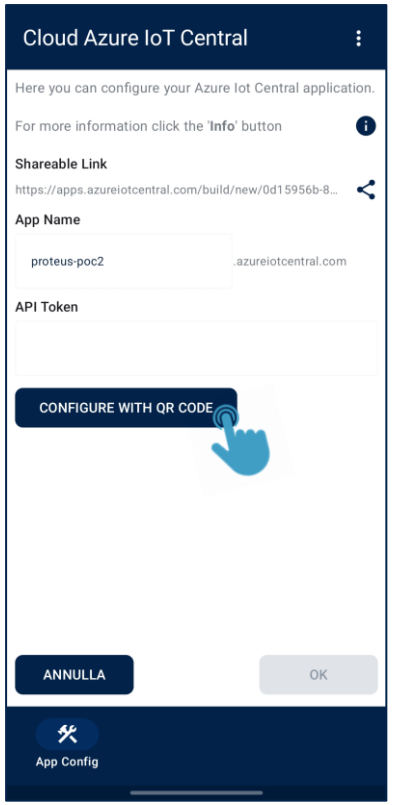
Azure IoT Central Cloud connection 1/2



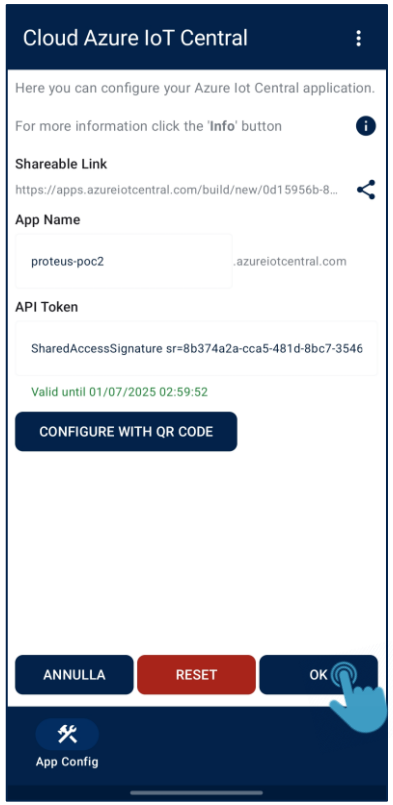
Open **Cloud Azure IoT Central**



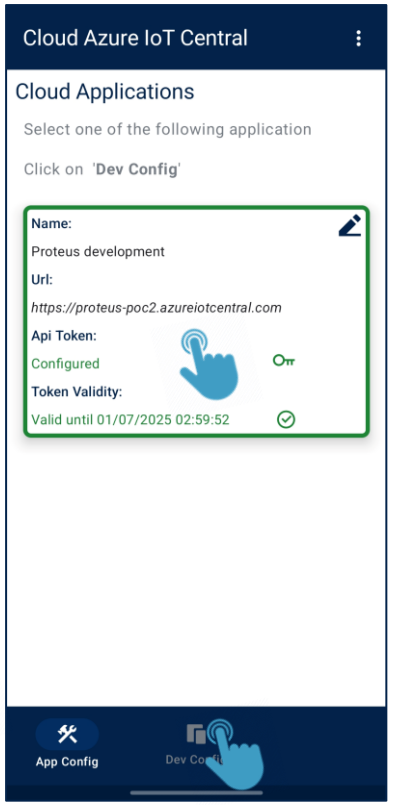
Select *Proteus development* cloud application



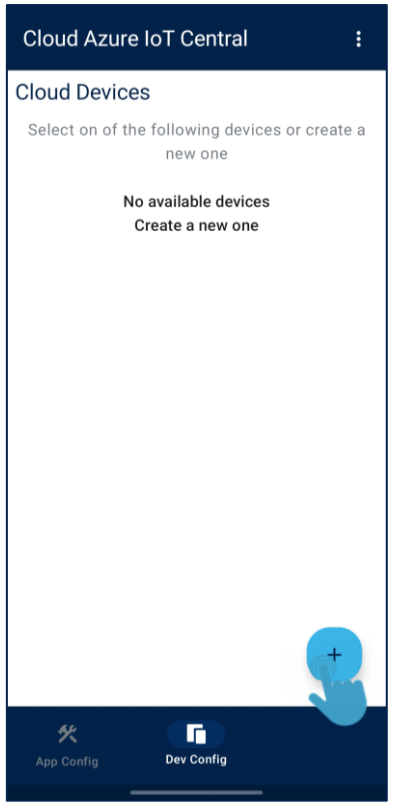
Insert the QR-Code retrieved during API token generation inside IoT Central UI



Confirm the configuration



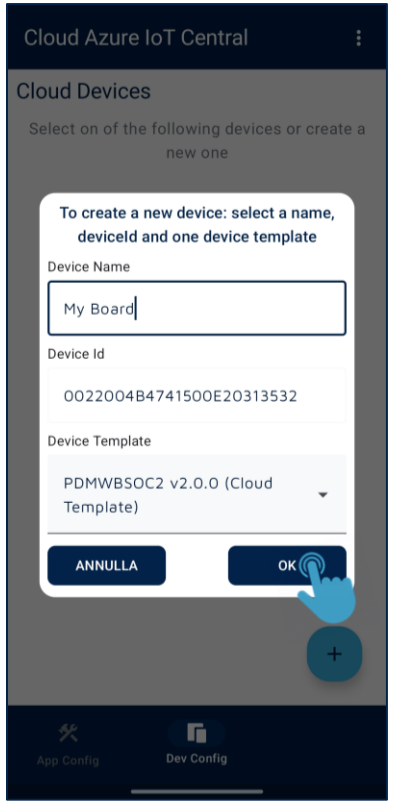
Select the configured cloud application and moves to *dev config* page



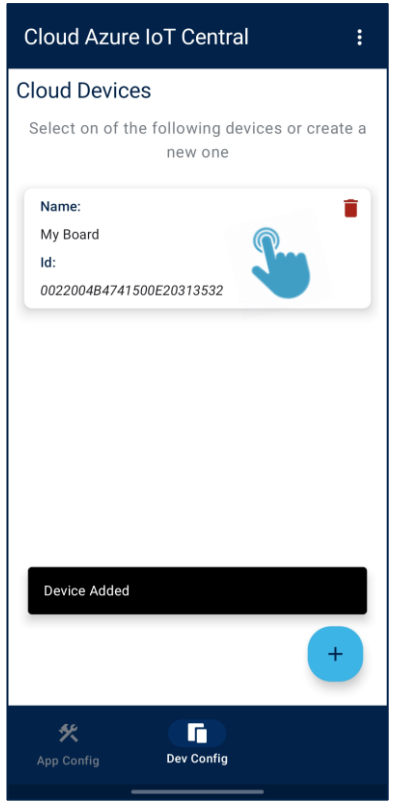
Tap on "**+**" icon to add your device

STBLESensor App

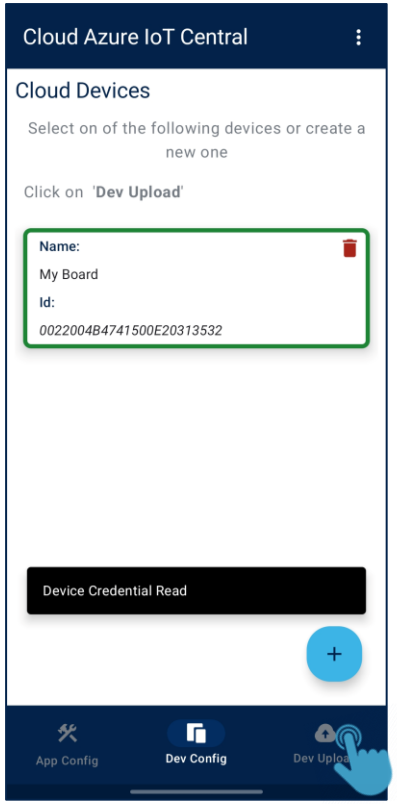
Azure IoT Central Cloud connection 2/2



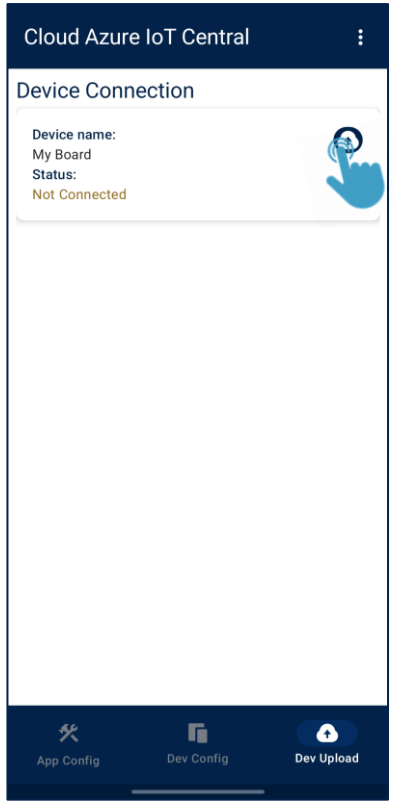
Enter the device name and select the right template



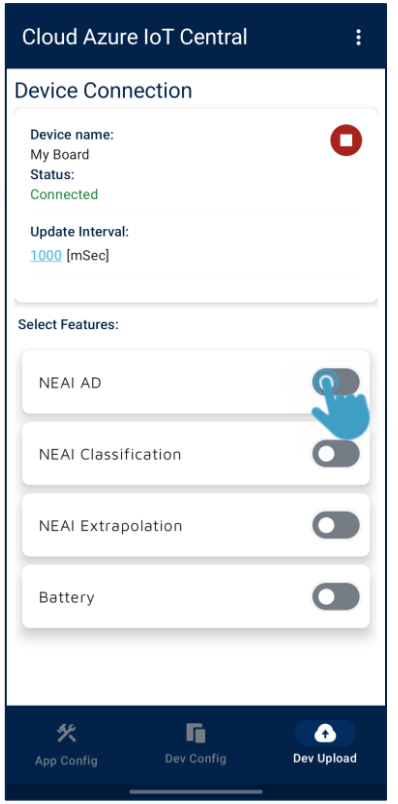
A pop-up will inform you that the device has been created



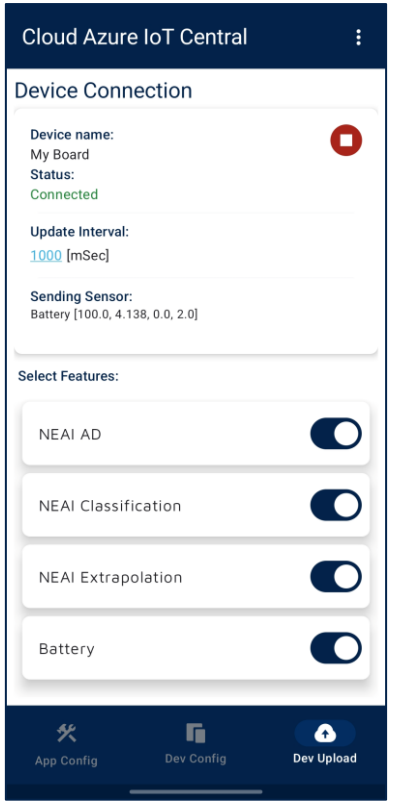
Select your device to complete the provisioning and move to *dev upload* page



Start data sending



Enable feature notifications using the corresponding switches



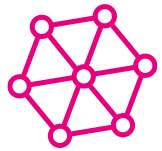
Open Azure IoT Central web page to remote monitor your industrial node device

life.augmented

2.5- Setup NanoEdgeAI library

FP-AI-PDMWBSOC2
Working flow

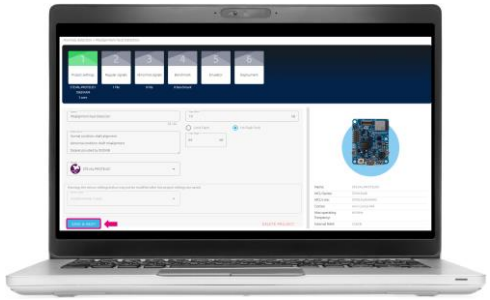
Capture data → **Label Dataset and Classifying** → **Benchmark to Build ML Libs** → **ML Libs integration in FW package** → **Run applications with NEAI AD, NCC or E**

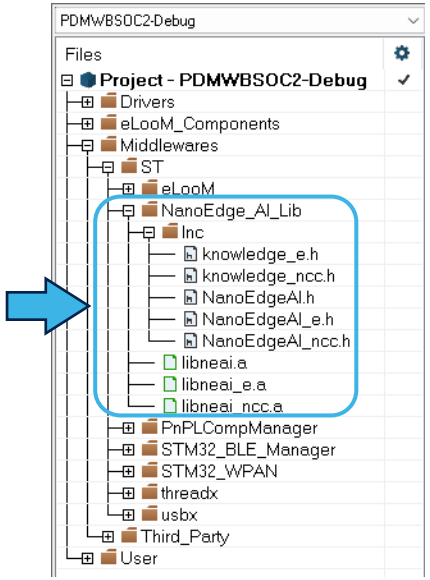


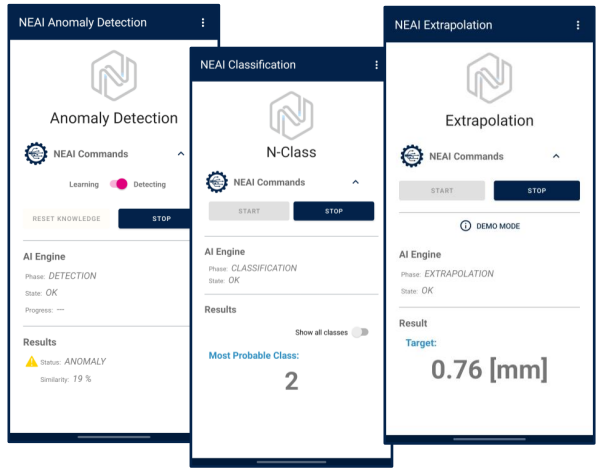
NANOEDGE AI STUDIO

STM32

STEVAL-PROTEUS1







AZURE IoT Central Cloud

NEAI Settings & FUOTA (By BLE)

STEVAL-PROTEUS1

Sensor Datalogging → **Every Datasets** → **Modeling** / **Results** → **NanoEdgeAI integration** → **Control by BLE App or CLI Terminal** → **Cloud**

life.augmented

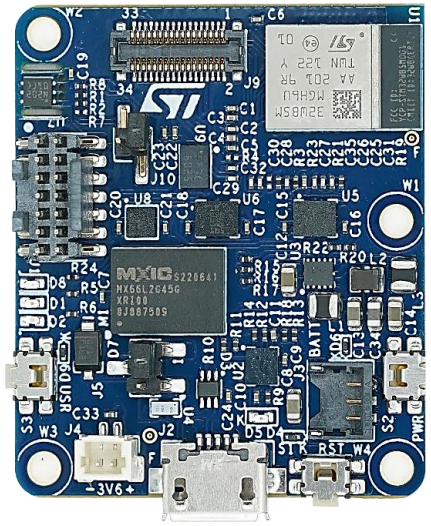
Capture data and create specific datasets

Generating contextual data using DATAPRO1 in Utilities folder

1.1 Generate datasets

- The equipment behaviors can be analyzed creating a lot of different datasets based on accelerometer, gyroscope or vibrometer data, they represent specific working condition to detect anomalies (AD) or to classify (NCC), or even they can be useful to extrapolate new information (E) from sensor raw data.
- To generate these datasets, you can use DATAPRO1 firmware available in:

  → ***Utilities/SwUtilities/HS_Datalog/FP-SNS-DATAPRO1_1_1_0.bin***



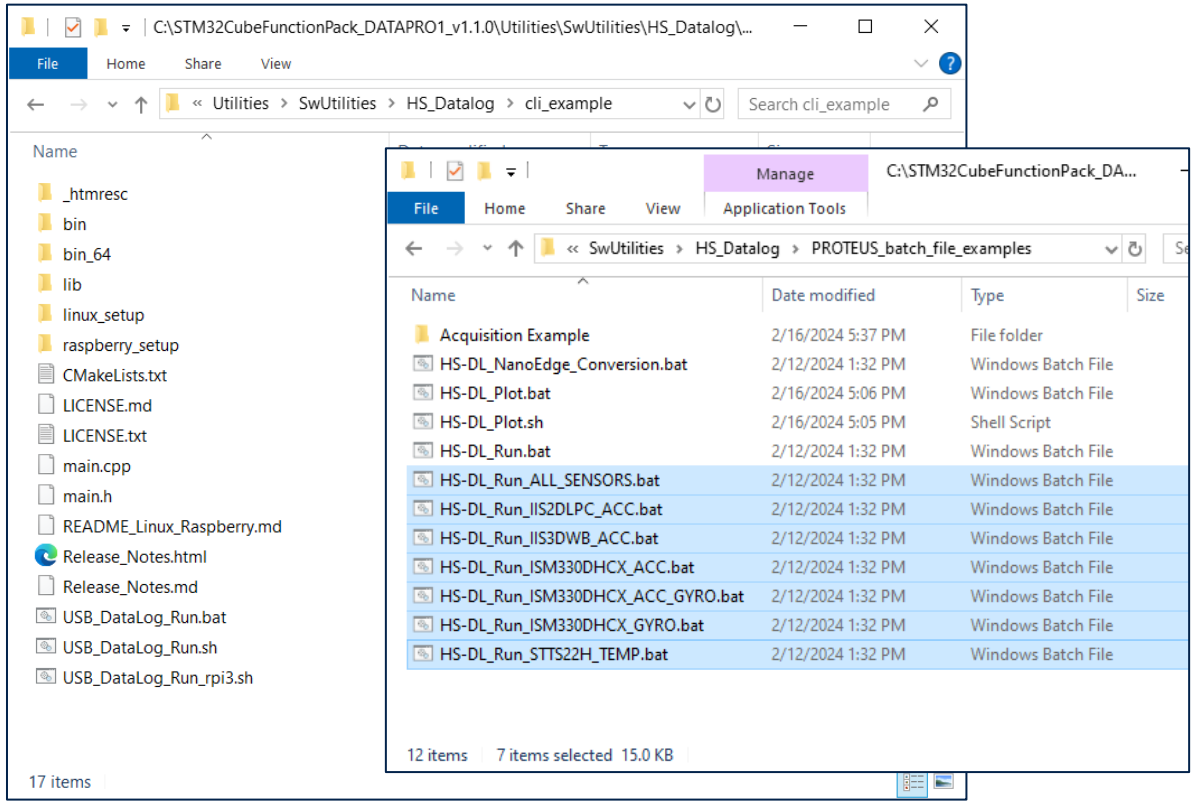
- As soon as you get the STEVAL-PROTEUS out of the box, it is already programmed with STSW-PROTEUS software package.
- In this case you can perform a firmware update over-the-air to update stack and application firmware.
- Supply the STEVAL-PROTEUS.
- Connect the smart mobile device running the STBLESensor app to the STEVAL-PROTEUS via BLE.
- Tap on the gear menu to launch the firmware update.
- Update the application firmware with DATAPRO1 binary file.

Capture data and create specific datasets

Generating contextual data using DATAPRO1 in Utilities folder

1.2 Generate datasets

- You can run the example after connecting the STEVAL-PROTEUS to a PC via USB-A to USB-microB cable.
- The command line example is in the ‘Utilities’ folder. It is available for Windows 32 and 64 bit, Linux 64 bit and Raspberry Pi 3 platforms.
- USB_DataLog_Run.bat, USB_DataLog_Run.sh and USB_DataLog_Run_rpi3.sh scripts provide a ready-to-use example. If needed, the application can receive as parameters: device configuration file (-f) and timeout (-t).
- For Linux or Raspberry, launch the scripts as bash.
- Utilities\SwUtilities\HS_Datalog\cli_example contains the CLI software program.
- Utilities\SwUtilities\HS_Datalog\PROTEUS_config_examples contains a lot of device configuration files.
- Utilities\SwUtilities\HS_Datalog\PROTEUS_batch_file_examples contains a lot of batch files related to the corresponding configuration files.

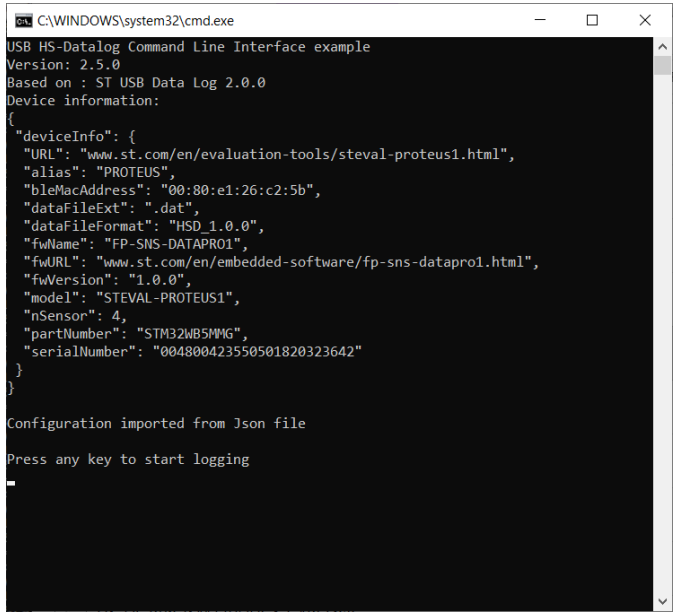


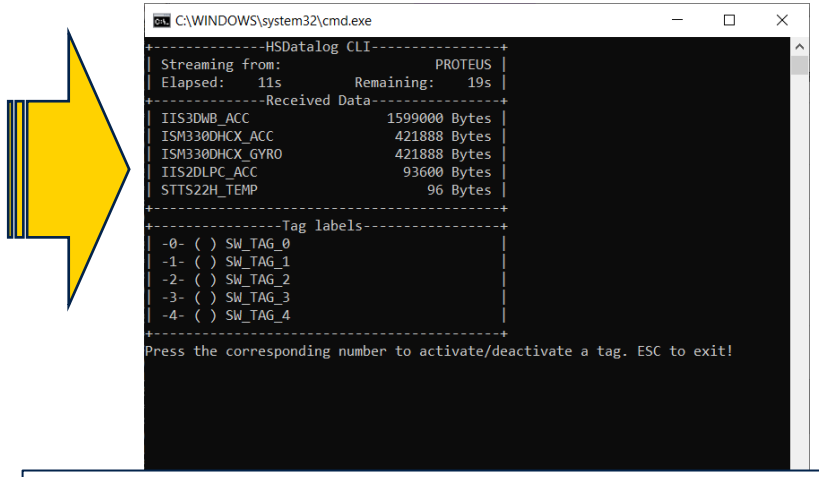
life.augmented

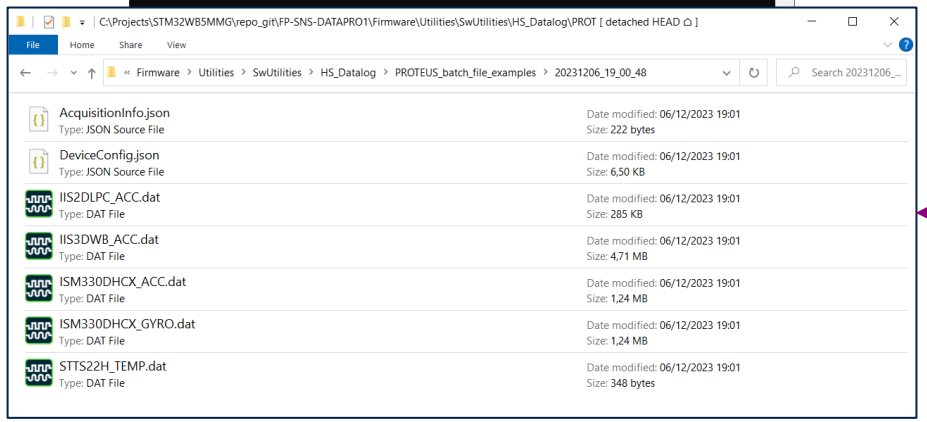
Capture data and create specific datasets

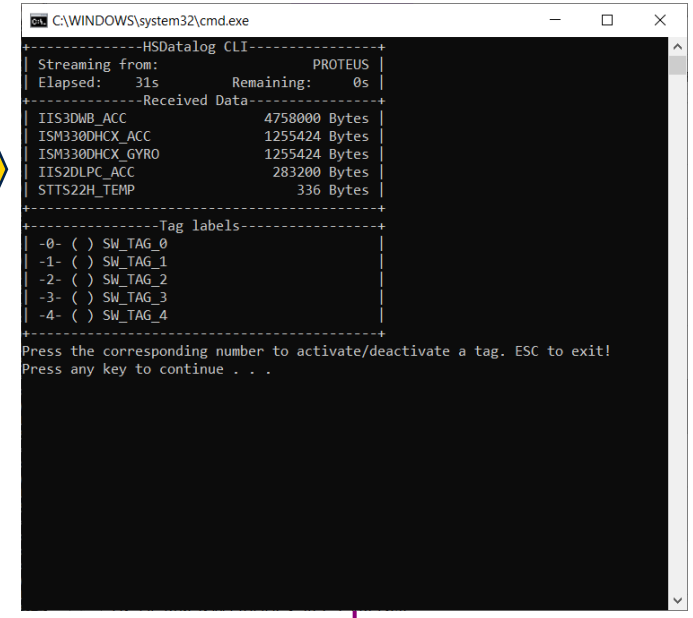
Generating contextual data using DATAPRO1 in Utilities folder

1.3 Generate datasets









As soon as the acquisition gets stop, the data are available into a dedicated folder in the PC.

life.augmented

Use NanoEdge AI Studio tool

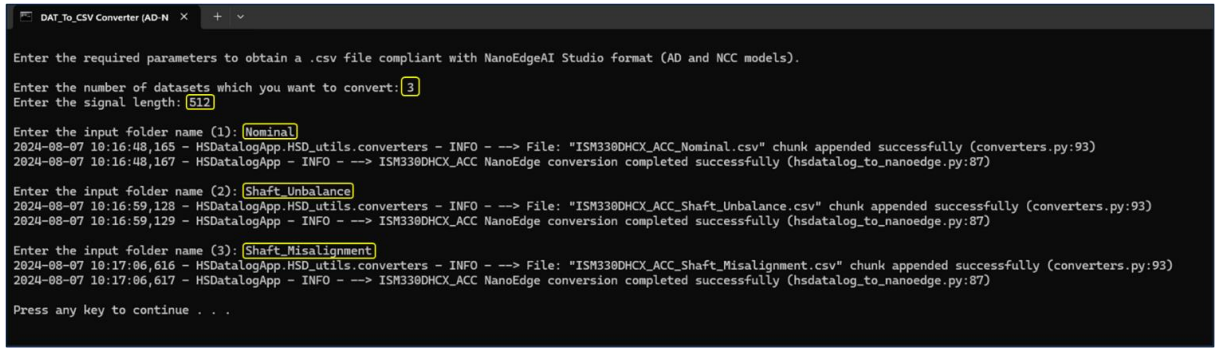
Label datasets to use them with NanoEdgeAI Studio

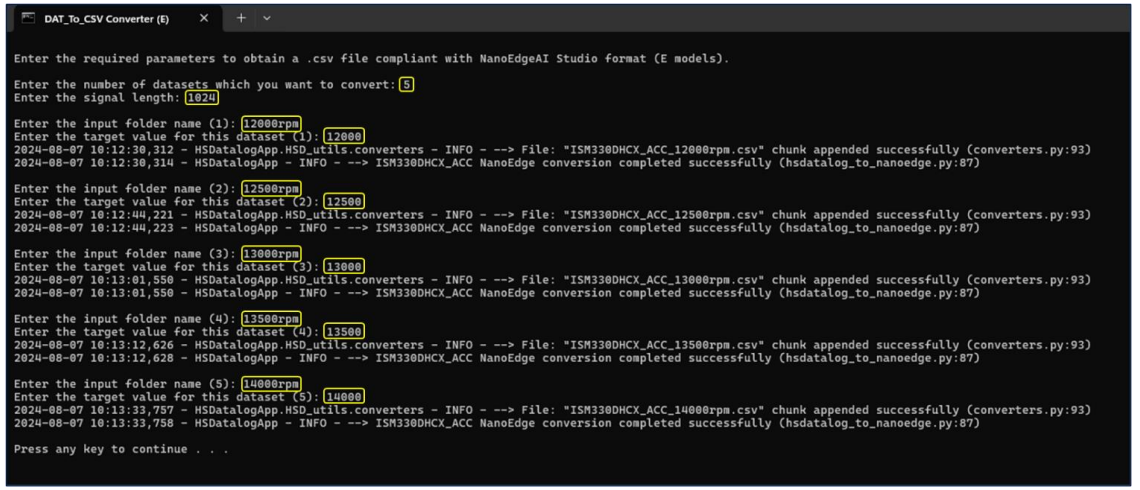
2.1 Convert each datasets in the right format (*.csv) accepted by NanoEdgeAI Studio tool

- The datasets previously generated using the datalogging firmware, must be used as input dataset to create the new libraries for Anomaly Detection, N-Class Classification or Extrapolation, to include inside the application FW.

- Since NanoEdgeAI Studio doesn't accept *.dat files as input dataset, you must convert your acquisition folders into *.csv using one of the following batch files:
  → ***"Utilities/SwUtilities/HS_Datalog/PROTEUS_batch_file_examples/HS-DL_NanoEdge_Conversion_AD_NCC.bat"***
  → ***"Utilities/SwUtilities/HS_Datalog/PROTEUS_batch_file_examples/HS-DL_NanoEdge_Conversion_E.bat"***

  acquisition folders must contain at least one ***.dat** file plus two json configuration files.
  You will use the AD_NCC version or the E version according to the kind of model which you want to generate through NanoEdgeAI Studio.

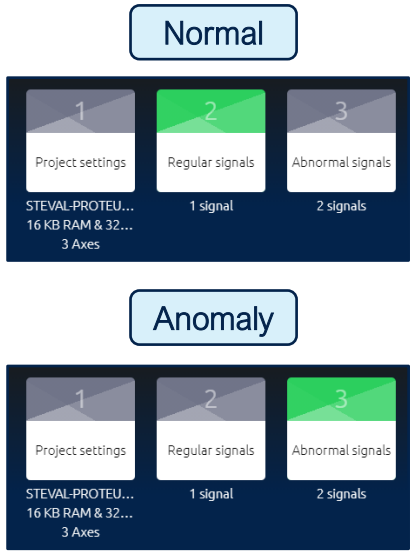


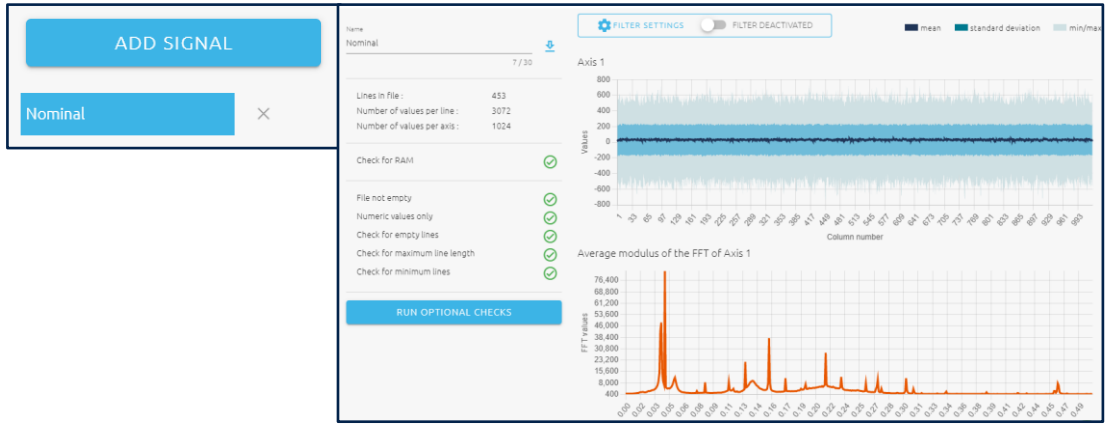


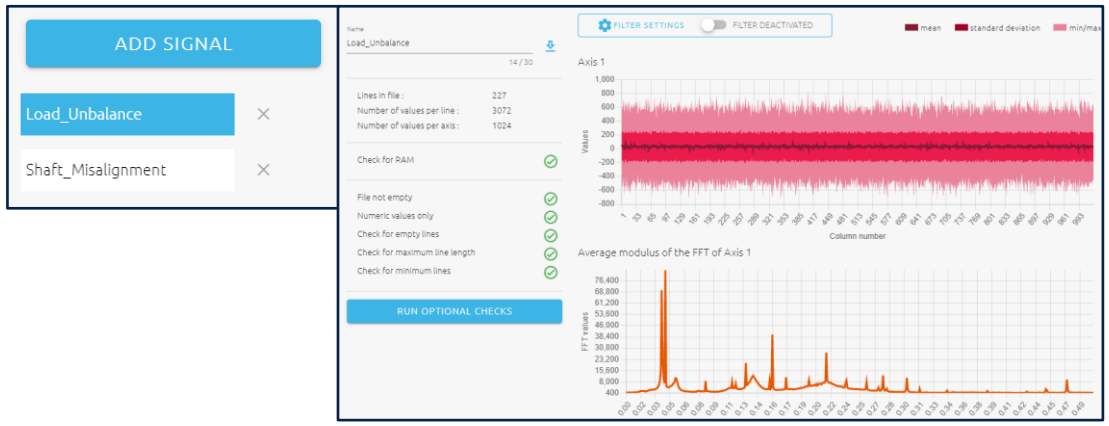
Use NanoEdge AI Studio tool

Label datasets to use them with NanoEdgeAI Studio

2.2 Create a new project for **Anomaly Detection** library starting from the datasets and build the library



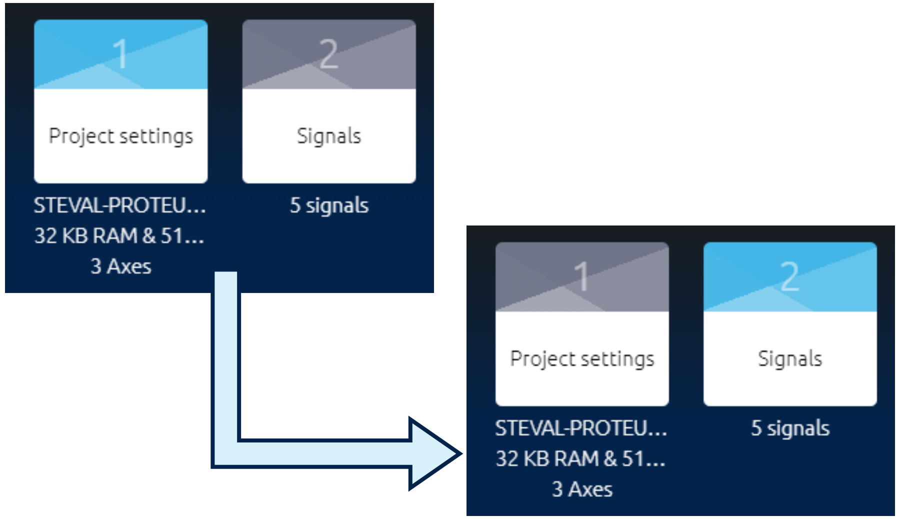


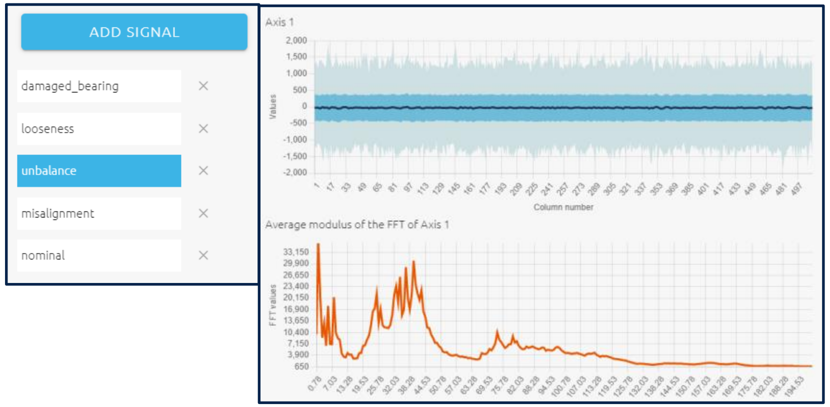


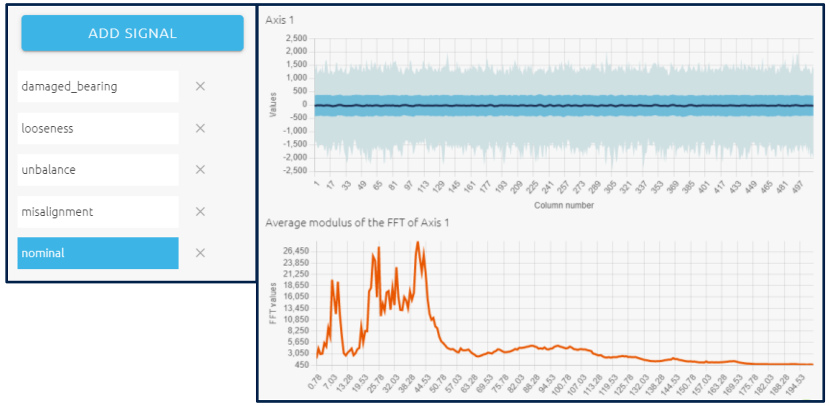
Use NanoEdge AI Studio tool

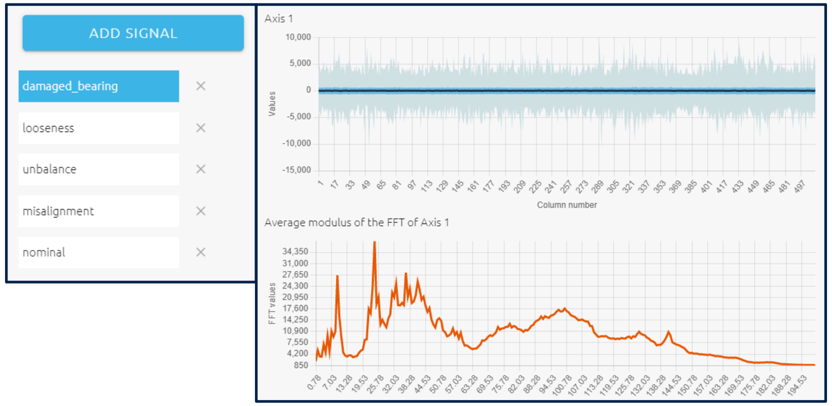
Label datasets to use them with NanoEdgeAI Studio

2.3 Create a new project for **N-Class Classification** library starting from the datasets and build the library







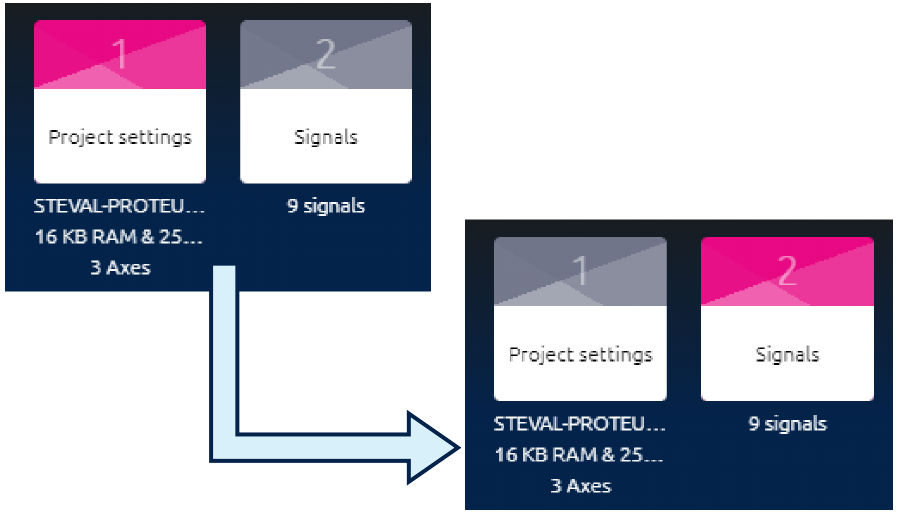


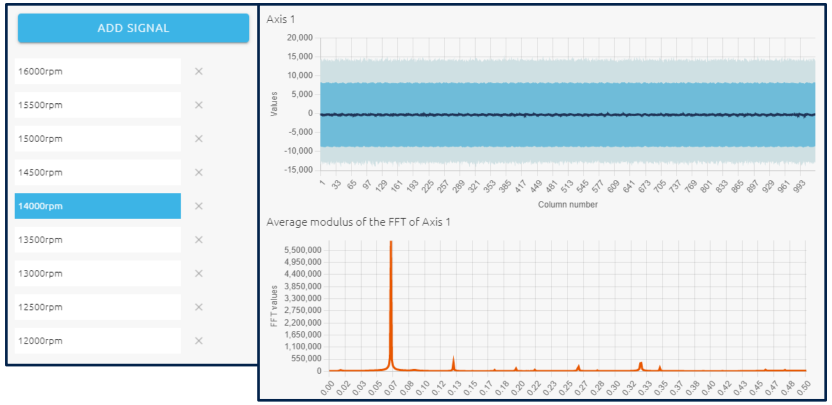
life.augmented

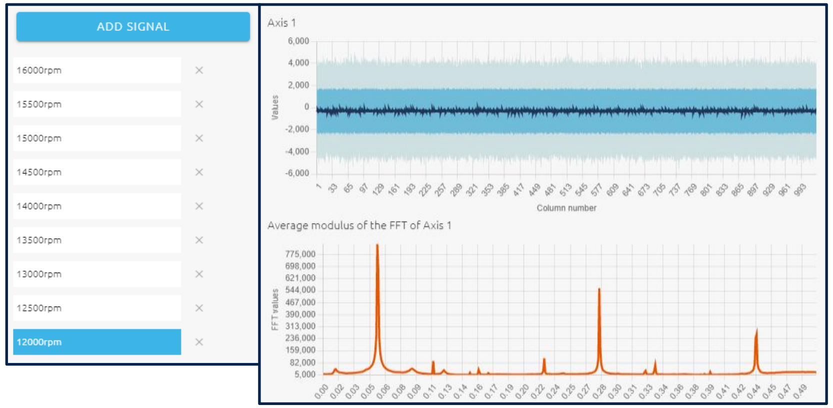
Use NanoEdge AI Studio tool

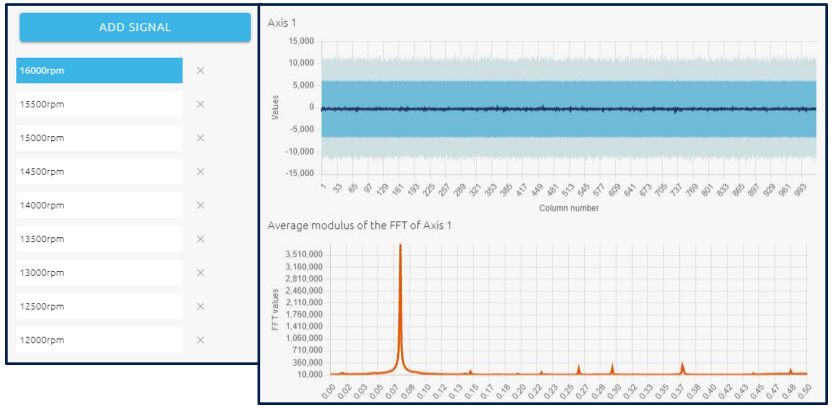
Label datasets to use them with NanoEdgeAI Studio

2.4 Create a new project for **Extrapolation** library starting from the datasets and build the library









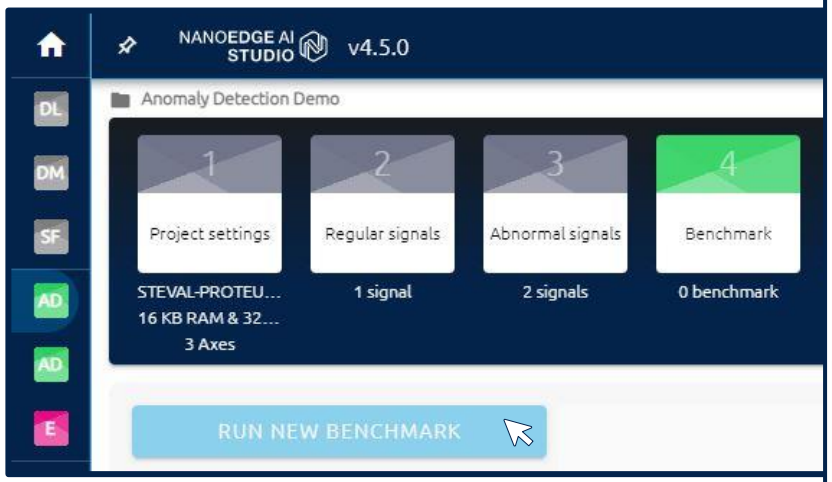
life.augmented

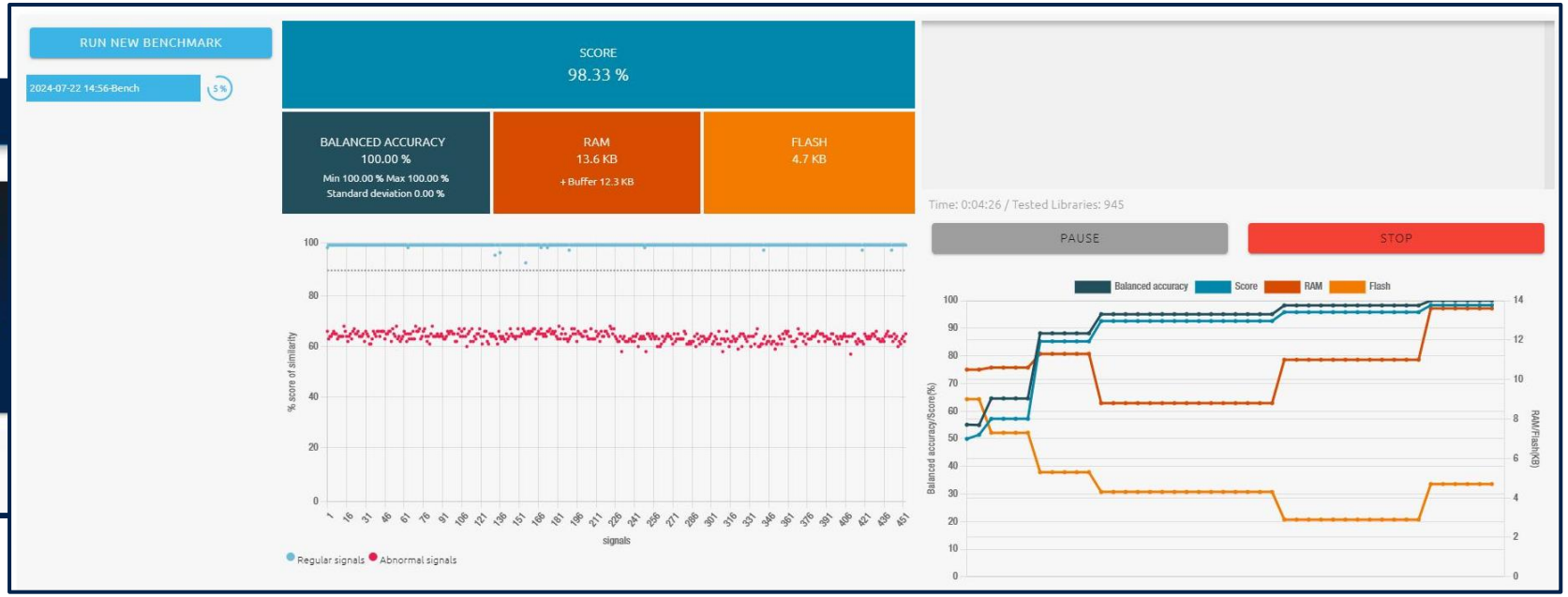
NANOEDGE AI STUDIO

Use NanoEdge AI Studio tool

Find the best model with NanoEdgeAI Studio

3.1 Run the benchmark to generate NEAI AD model





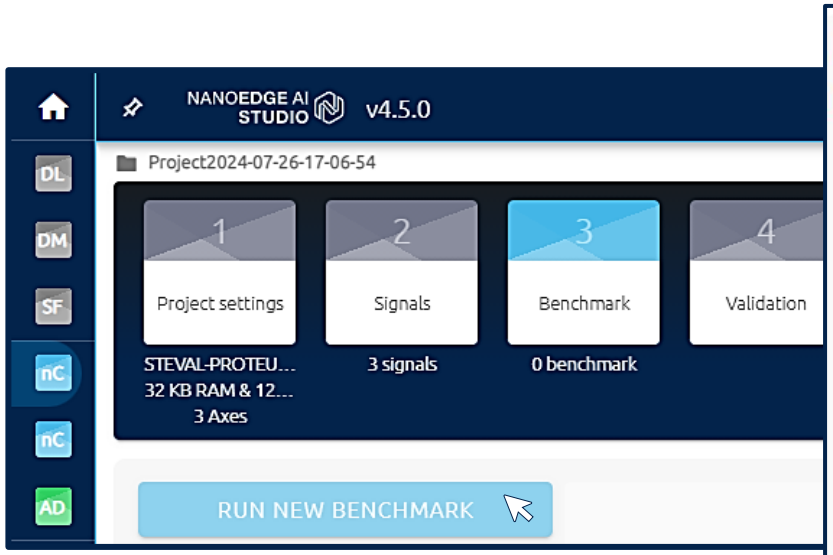
life.augmented

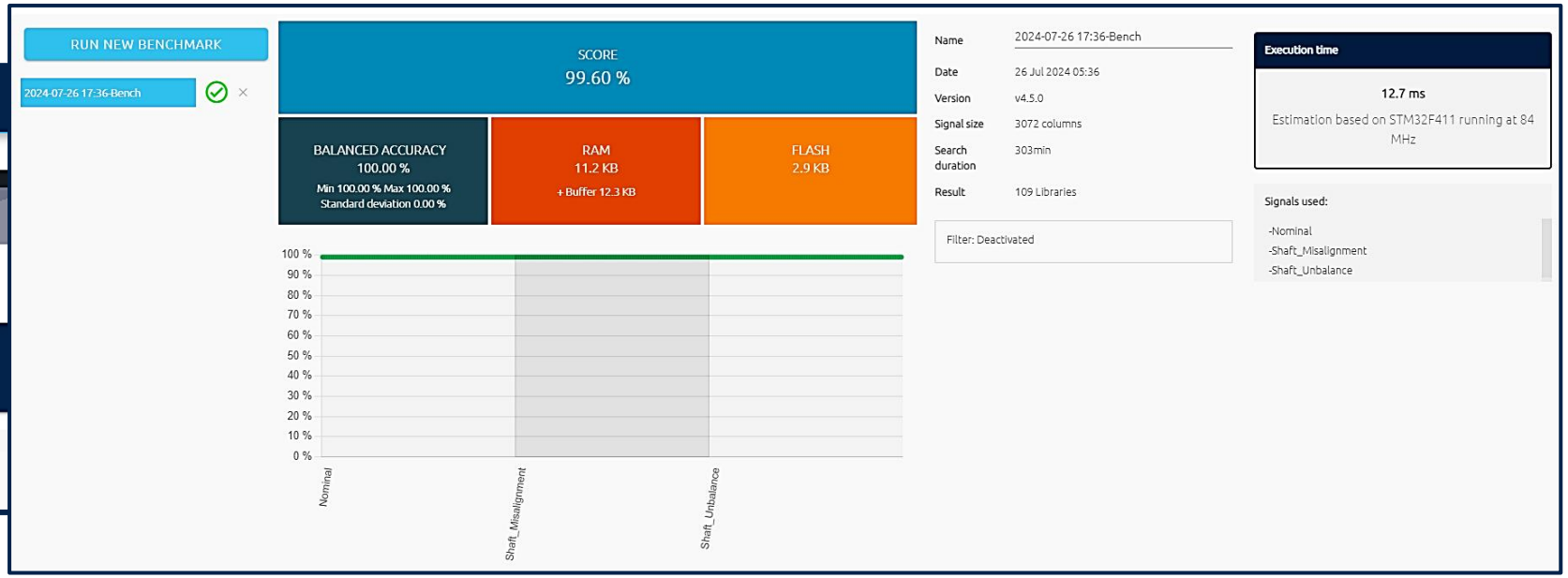
NANOEDGE AI STUDIO

Use NanoEdge AI Studio tool

Find the best model with NanoEdgeAI Studio

3.2 Run the benchmark to generate NEAI NCC model





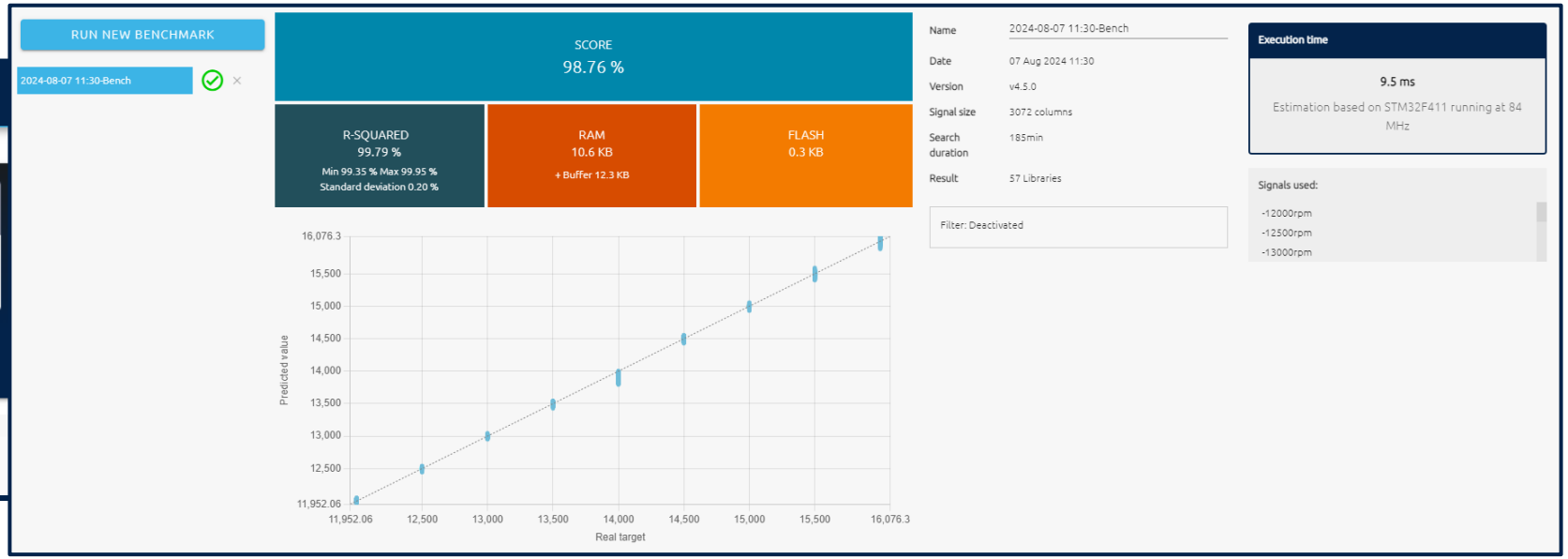
life.augmented

NANOEDGE AI STUDIO

Use NanoEdge AI Studio tool

Find the best model with NanoEdgeAI Studio

3.3 Run the benchmark to generate NEAI E model



life.augmented

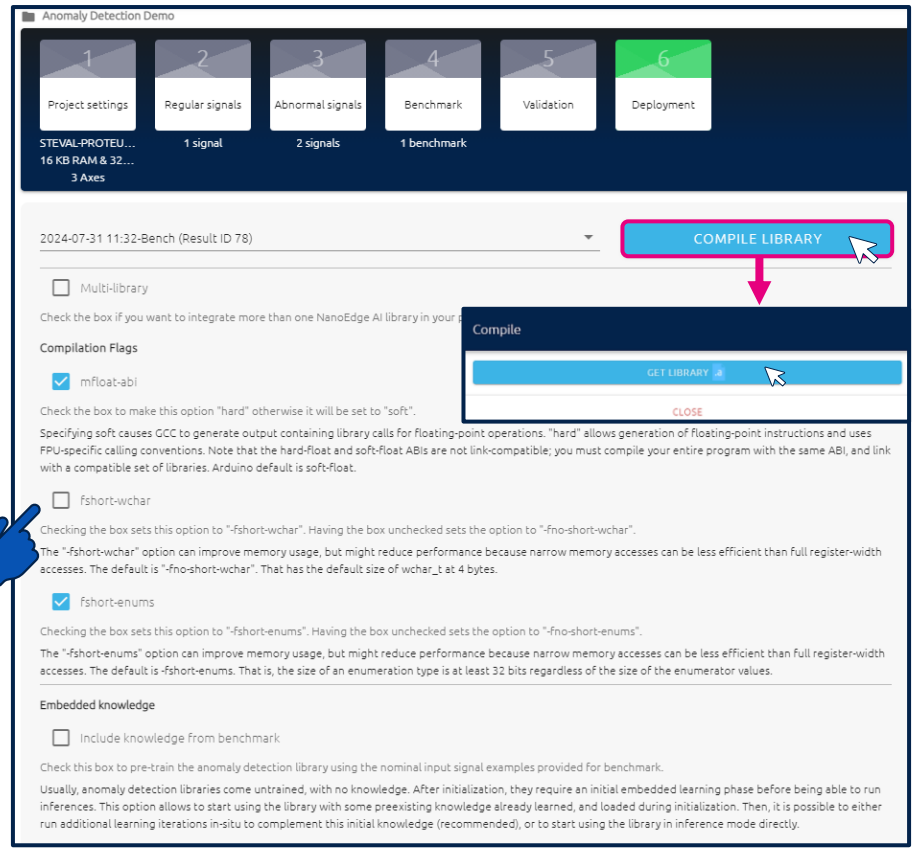
NANOEDGE AI STUDIO

Use NanoEdge AI Studio tool

Choose the best libraries and deploy in binary format

4.1 Press the “Compile Library” button to deploy a ZIP file including all the library files to include in your application

Anomaly Detection



Set "**fshort-wchar**" flag just to deploy the library for the **Keil®** toolchain

life.augmented

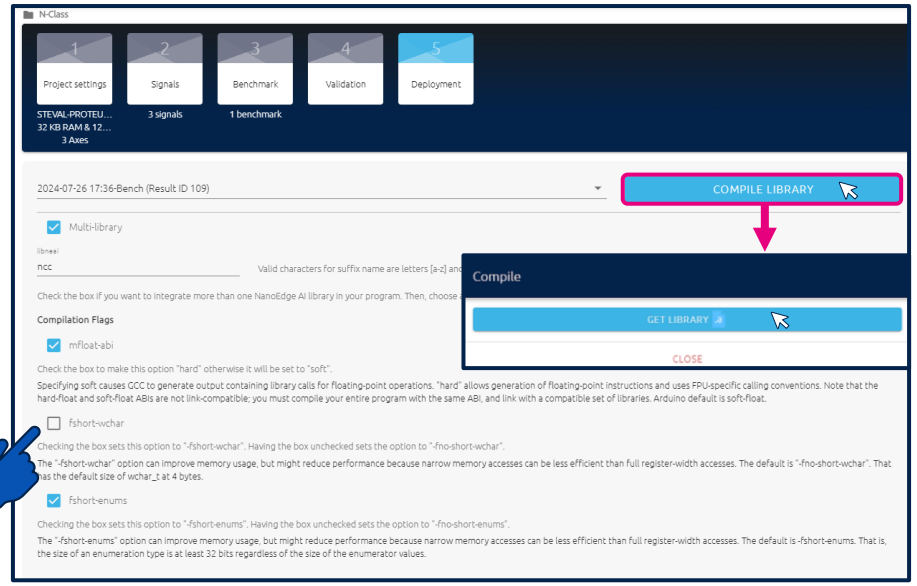
NANOEDGE AI STUDIO

Use NanoEdge AI Studio tool

Choose the best libraries and deploy in binary format

4.2 Press the “Compile Library” button to deploy a ZIP file including all the library files to include in your application

N-Class Classification



Set "fshort-wchar" flag just to deploy the library for the Keil® toolchain

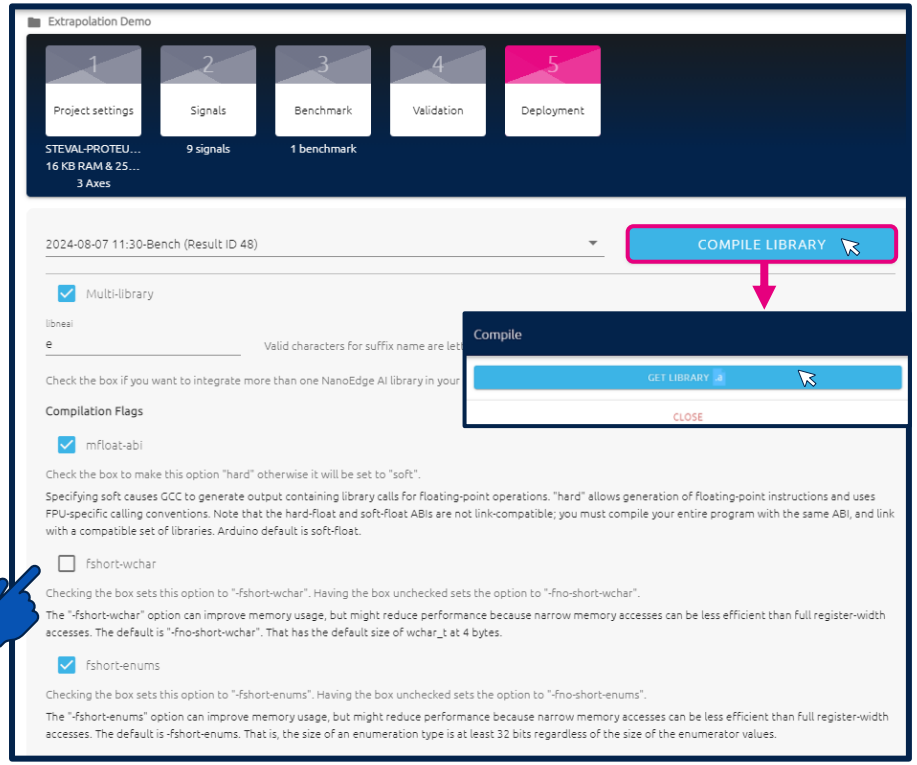
life.augmented

Use NanoEdge AI Studio tool

Choose the best libraries and deploy in binary format

4.3 Press the “Compile Library” button to deploy a ZIP file including all the library files to include in your application

Extrapolation



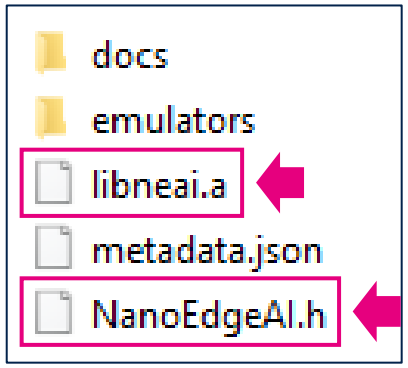
Set "**fshort-wchar**" flag just to deploy the library for the **Keil®** toolchain

NEAI Library Integration in the FP

Embed the machine learning library into FP-AI-PDMWBSOC2

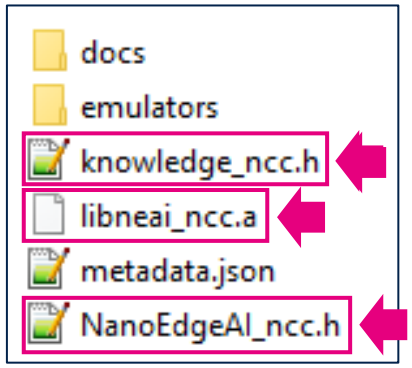
5.1 Replace just the following NEAI files deployed into the project folder

Anomaly Detection



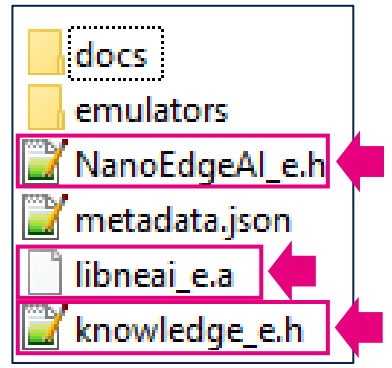
1. **libneai.a**
   (for IAR&STM32CubeIDE projects)
2. **libneai.lib**
   (for KEIL projects)
3. **NanoEdgeAI.h**

N-CLASS Classification



1. **libneai_ncc.a**
   (for IAR&STM32CubeIDE projects)
2. **libneai_ncc.lib**
   (for KEIL projects)
3. **NanoEdgeAI_ncc.h**
4. **knowledge_ncc.h**

Extrapolation



1. **libneai_e.a**
   (for IAR&STM32CubeIDE projects)
2. **libneai_e.lib**
   (for KEIL projects)
3. **NanoEdgeAI_e.h**
4. **knowledge_e.h**

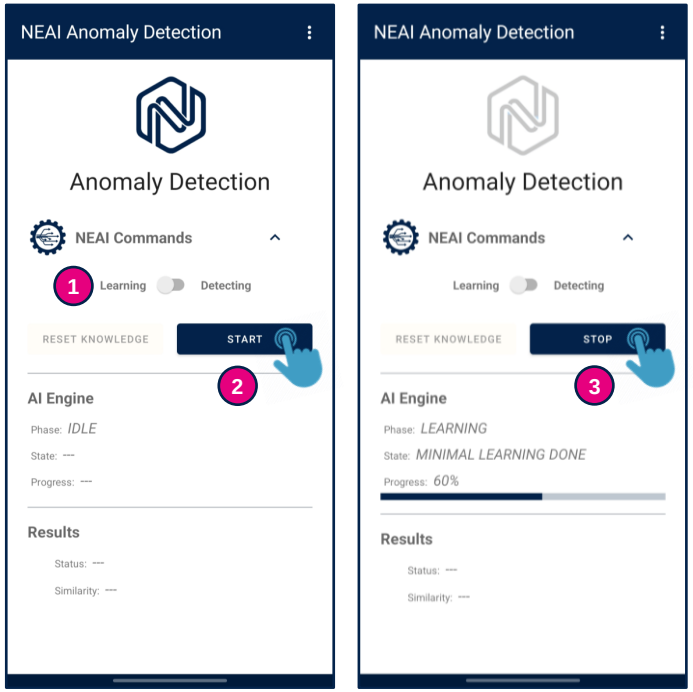
5.2 Compile again the FW application in order to update the NEAI libraries.
Once updated the application binaries, the BLE FUOTA functionalities allow to update directly the STEVAL-PROTEUS without using programmer tools.

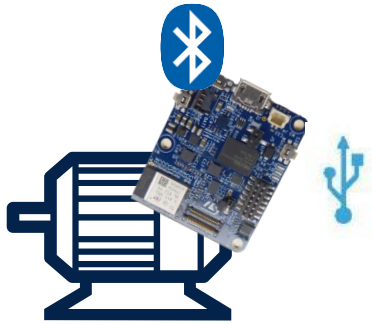
life.augmented

Run Learning and Detection by BLE or CLI

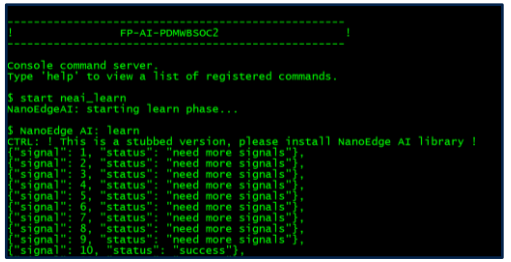
Run applications with NEAI Anomaly Detection

Run NEAI-AD Learning

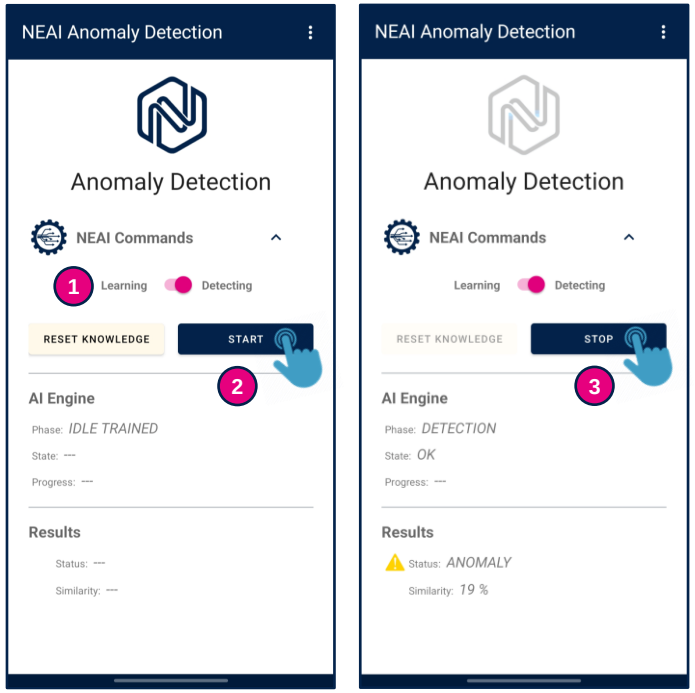


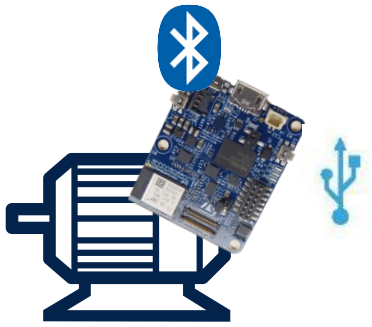


start neai_learn

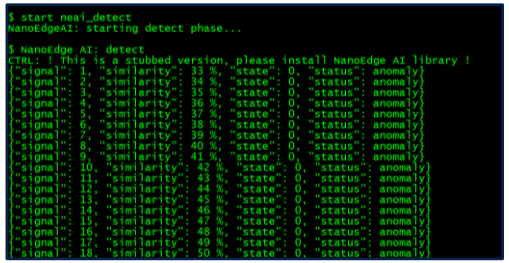


Run NEAI-AD Detection





start neai_detect



When the phase is IDLE, is mandatory to perform a new **LEARNING** phase before **DETECTING**, using the CLI or directly the STBLESensor App.

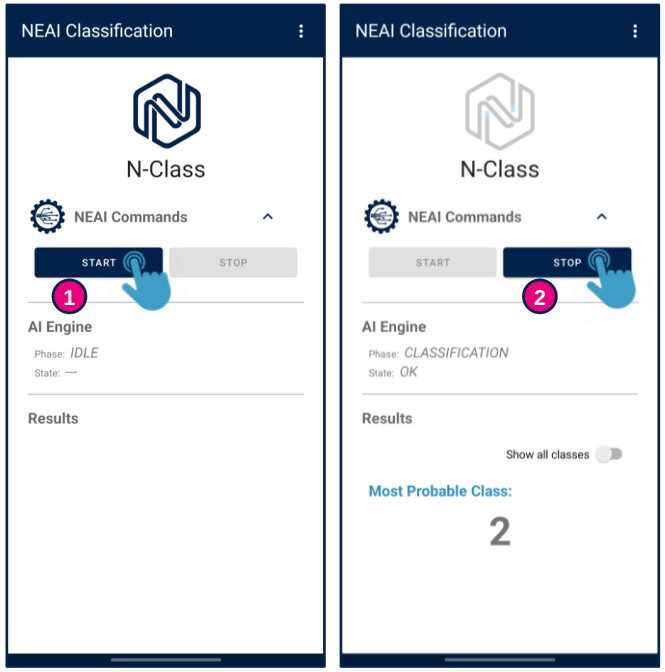
- ❑ **Learn** the normal modes on the edge
- ❑ **Detect** anomalies on your asset

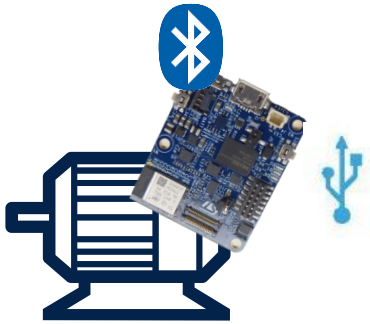
life.augmented

Run N-Class Classification by BLE or CLI

Run applications with NEAI N-Class Classification

Run NanoEdgeAI N-Class Classification





start neai_class

```
$ start neai_class
NanoEdgeAI: starting classification phase...

$ NanoEdge AI: classification
CTRL: ! This is a stubbed version, please install NanoEdge AI library !
{"signal": 1, "class": Class2},
{"signal": 2, "class": Class2},
{"signal": 3, "class": Class2},
{"signal": 4, "class": Class2},
{"signal": 5, "class": Class3},
{"signal": 6, "class": Class3},
{"signal": 7, "class": Class3},
{"signal": 8, "class": Class3},
{"signal": 9, "class": Class3},
{"signal": 10, "class": Class3},
{"signal": 11, "class": Class3},
{"signal": 12, "class": Class3},
{"signal": 13, "class": Class3},
{"signal": 14, "class": Class1},
{"signal": 15, "class": Class1},
{"signal": 16, "class": Class1},
{"signal": 17, "class": Class1},
```

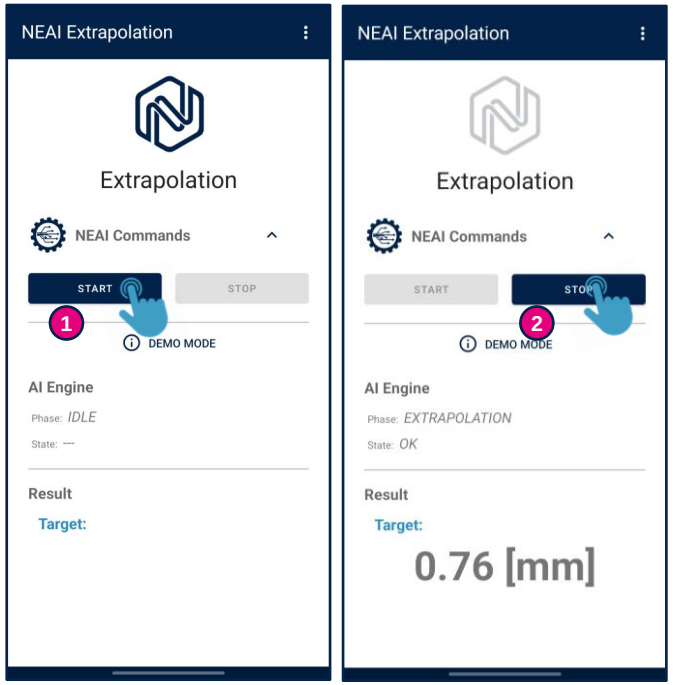
- **Class**ify asset behaviours

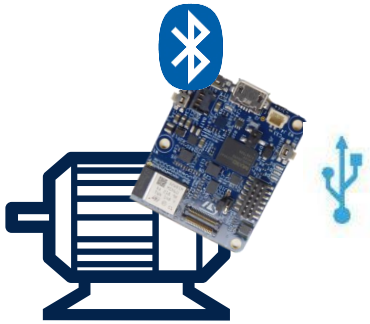
life.augmented

Run Extrapolation by BLE or CLI

Run applications with NEAI Extrapolation

Run NanoEdgeAI Extrapolation





start neai_extrapolate

```
$ start neai_extrapolate
NanoEdgeAI: starting extrapolation phase...

$ NanoEdge AI: extrapolation
CTRL: ! This is a stubbed version, please install NanoEdge AI library !
{"signal": 1, "extrapolated value": 0.00},
{"signal": 2, "extrapolated value": 0.01},
{"signal": 3, "extrapolated value": 0.02},
{"signal": 4, "extrapolated value": 0.03},
{"signal": 5, "extrapolated value": 0.04},
{"signal": 6, "extrapolated value": 0.05},
{"signal": 7, "extrapolated value": 0.06},
{"signal": 8, "extrapolated value": 0.07},
{"signal": 9, "extrapolated value": 0.08},
{"signal": 10, "extrapolated value": 0.09},
{"signal": 11, "extrapolated value": 0.10},
{"signal": 12, "extrapolated value": 0.11},
{"signal": 13, "extrapolated value": 0.12},
{"signal": 14, "extrapolated value": 0.13},
{"signal": 15, "extrapolated value": 0.14},
{"signal": 16, "extrapolated value": 0.15},
{"signal": 17, "extrapolated value": 0.16},
{"signal": 18, "extrapolated value": 0.17},
{"signal": 19, "extrapolated value": 0.18},
```

- **Extrapolate** info from sensor data

3- Documents & Related Resources

life.augmented

Documents & Related Resources

All documents are available in the DESIGN tab of the related products webpage

STEVAL-PROTEUS1

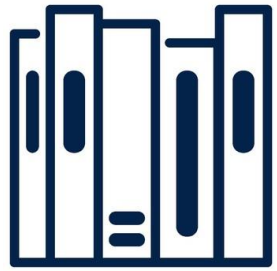
- **DB4641**: Industrial sensor evaluation kit for condition monitoring based on the 2.4 GHz STM32WB5MMG module – HW Data brief Hardware
- **UM3000**: Getting started with the STEVAL-PROTEUS1 evaluation kit for condition monitoring based on the 2.4 GHz STM32WB5MMG module – HW User Manual
- **Schematics**, **BOM**, **Gerber files**, Certifications

FP-AI-PDMWBSOC2

- **DB5055:** STM32Cube function pack for STEVAL-PROTEUS1 for AI anomaly detection, classification and extrapolation based on AzureRTOS
- **UM3208:** Getting started with the STM32Cube function pack for STEVAL-PROTEUS1 evaluation kit for predictive maintenance application based on artificial intelligence (AI)
- **QUICK START GUIDE**: STM32Cube function pack for STEVAL-PROTEUS1 for AI anomaly detection, classification and extrapolation based on AzureRTOS

SW TOOLS

- **STBLESensor :** ST BLE Sensor application for Android and iOS
- **NanoEdgeAI Studio**: Automated Machine Learning (ML) tool for STM32 developers



4 - STM32 Open Development Environment: Overview

life.augmented

STM32 ODE Ecosystem

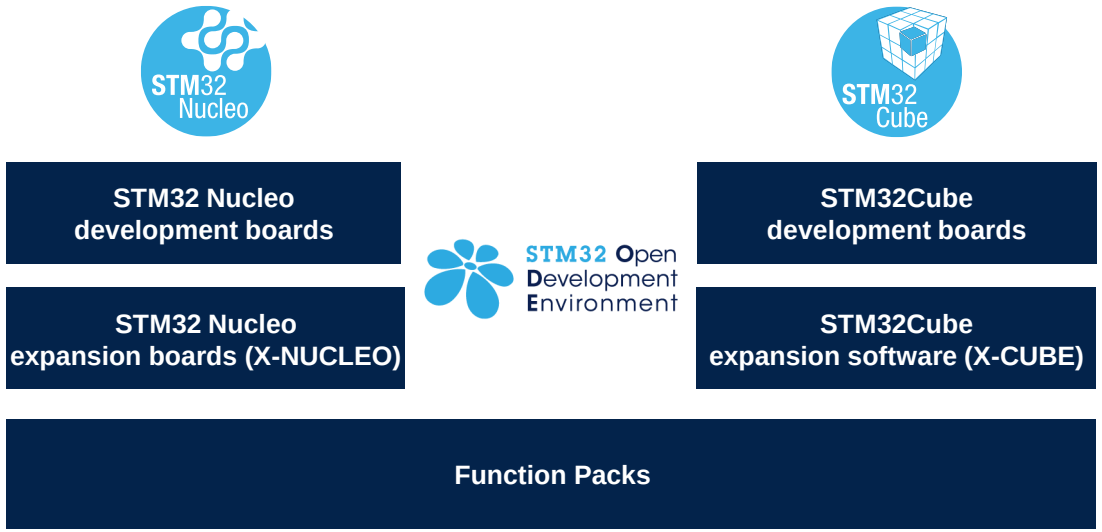
FAST, AFFORDABLE PROTOTYPING AND DEVELOPMENT

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

The STM32 ODE includes the following five elements:

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

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



life.augmented

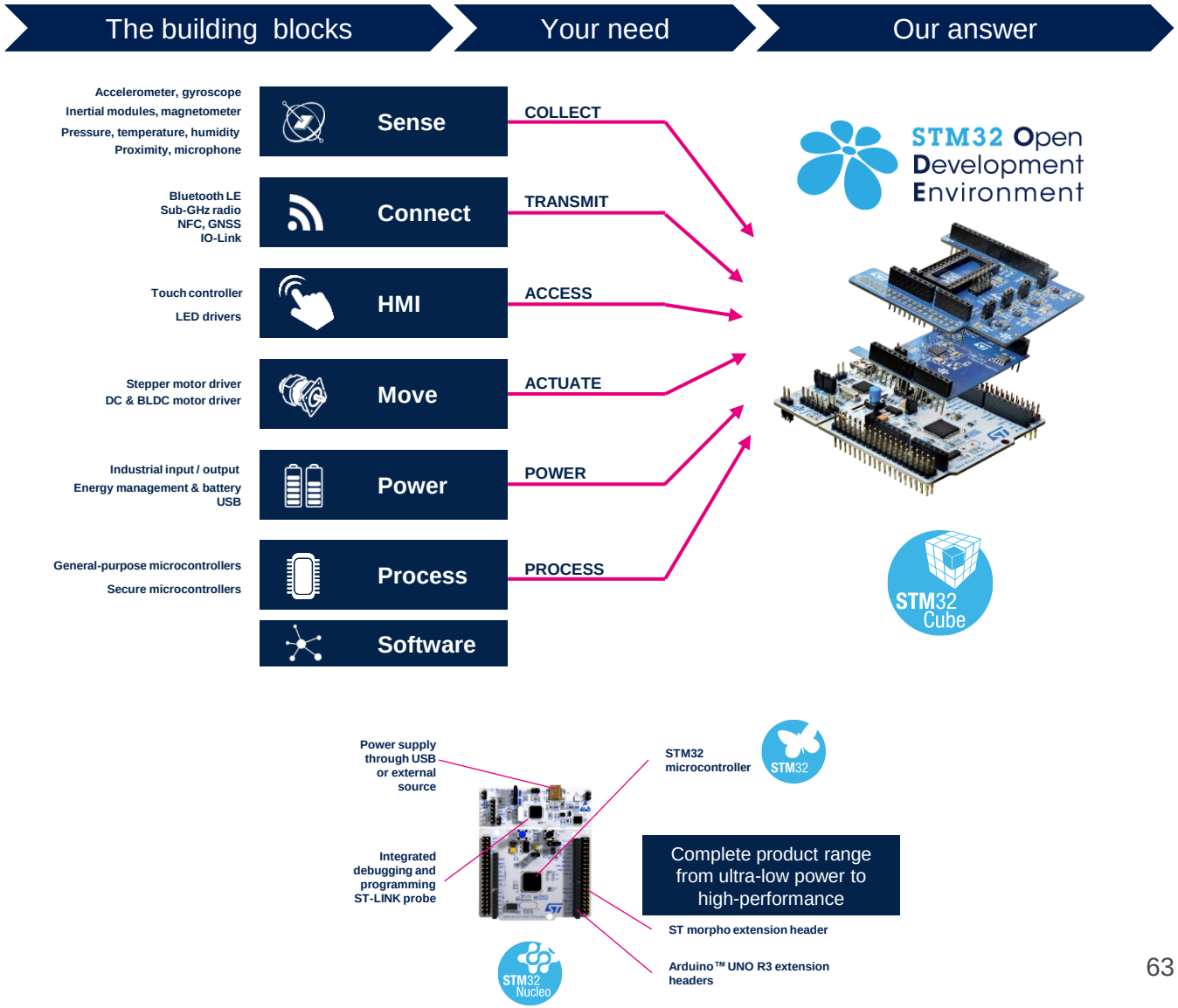
STM32 Open Development Environment
all that you need

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

To start your design:

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

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



life.augmented

Thank you

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

life.augmented