life.augmented

NanoEdge AI Studio

Your fast track to smart products

Yijun DENG

APAC AI Competence Center

Aug 2022

NANOEDGE AI STUDIO

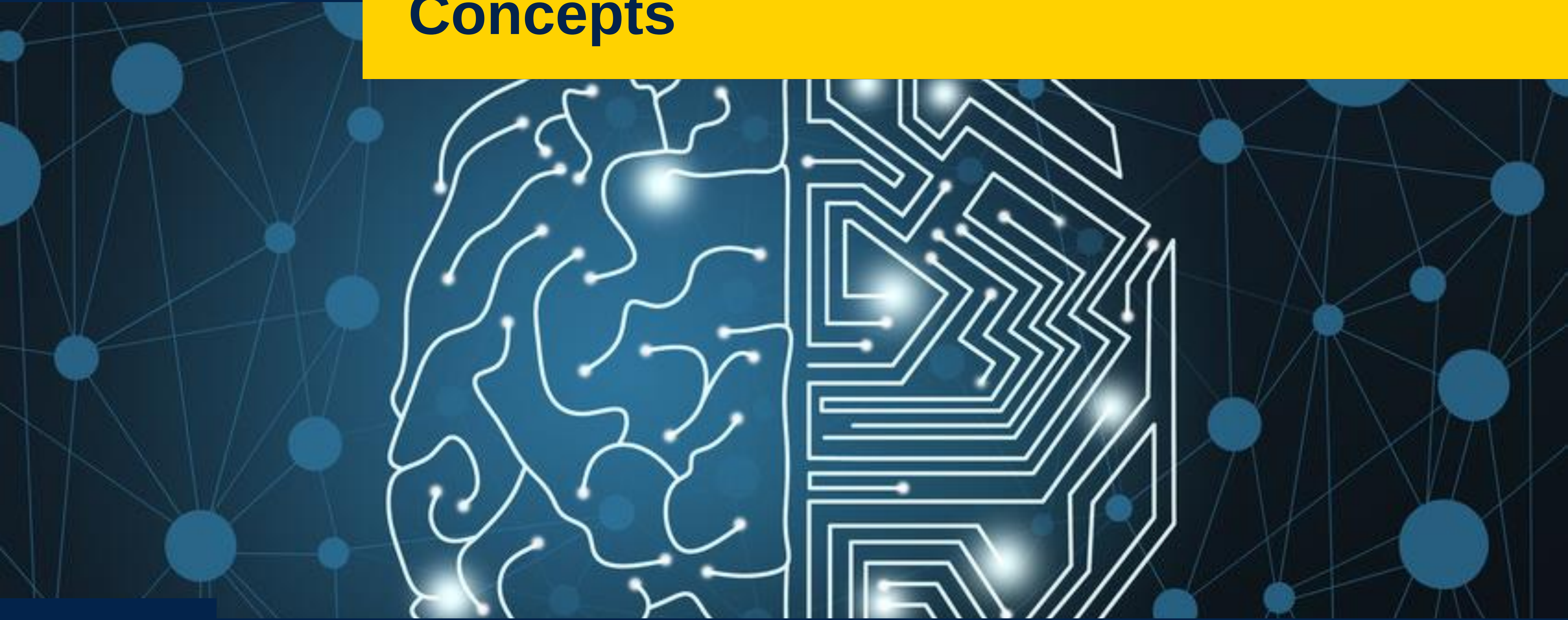
Agenda

1 Introduction to NanoEdge AI Studio

2 NanoEdge AI step by step

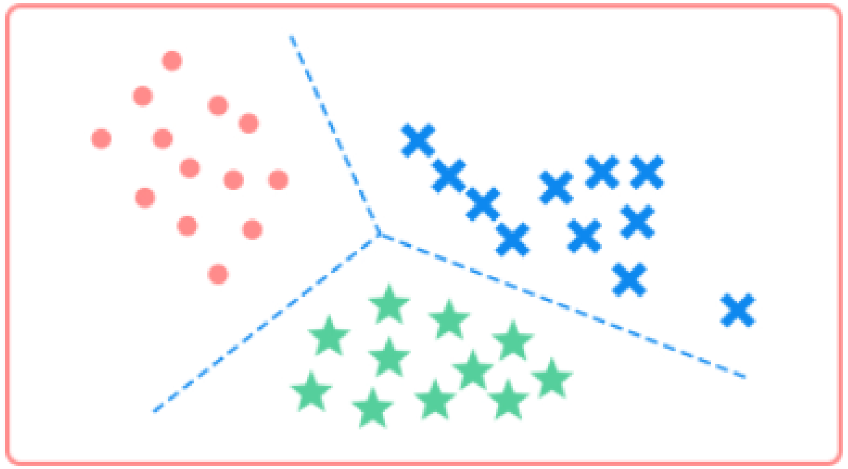
life.augmented

Quick Recap: Machine Learning Concepts

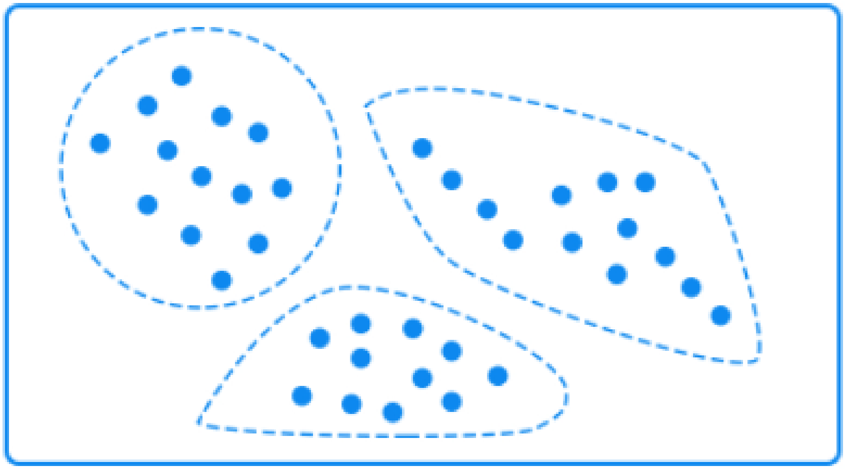


Machine Learning

- Can be divided into:
- Supervised learning: machine learning algorithm which learns a function that maps an input to an output based on example input-output pairs. E.g., Decision Tree, Support Vector Machine, Linear Regression, Deep Learning, …
- Unsupervised learning: machine learning algorithm which learns unknown patterns from un-labeled data. E.g., Clustering, K-Means, …



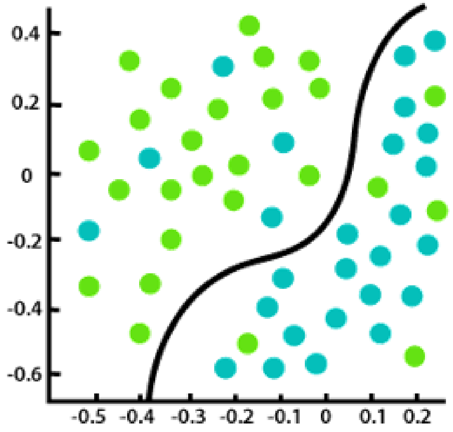
Supervised learning



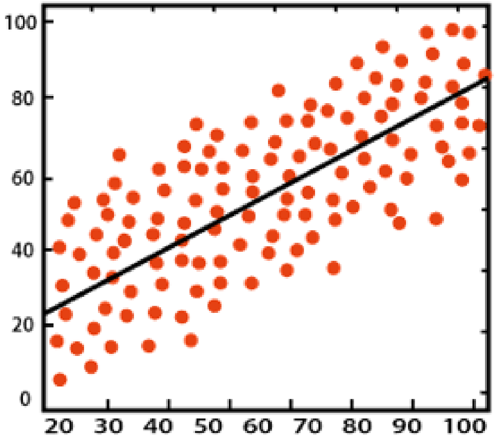
Unsupervised learning

Classification vs Regression

- Supervised learning contains mainly 2 categories:
- Classification: algorithm to predict a discrete class label
- Regression: algorithm to predict a continuous quantity

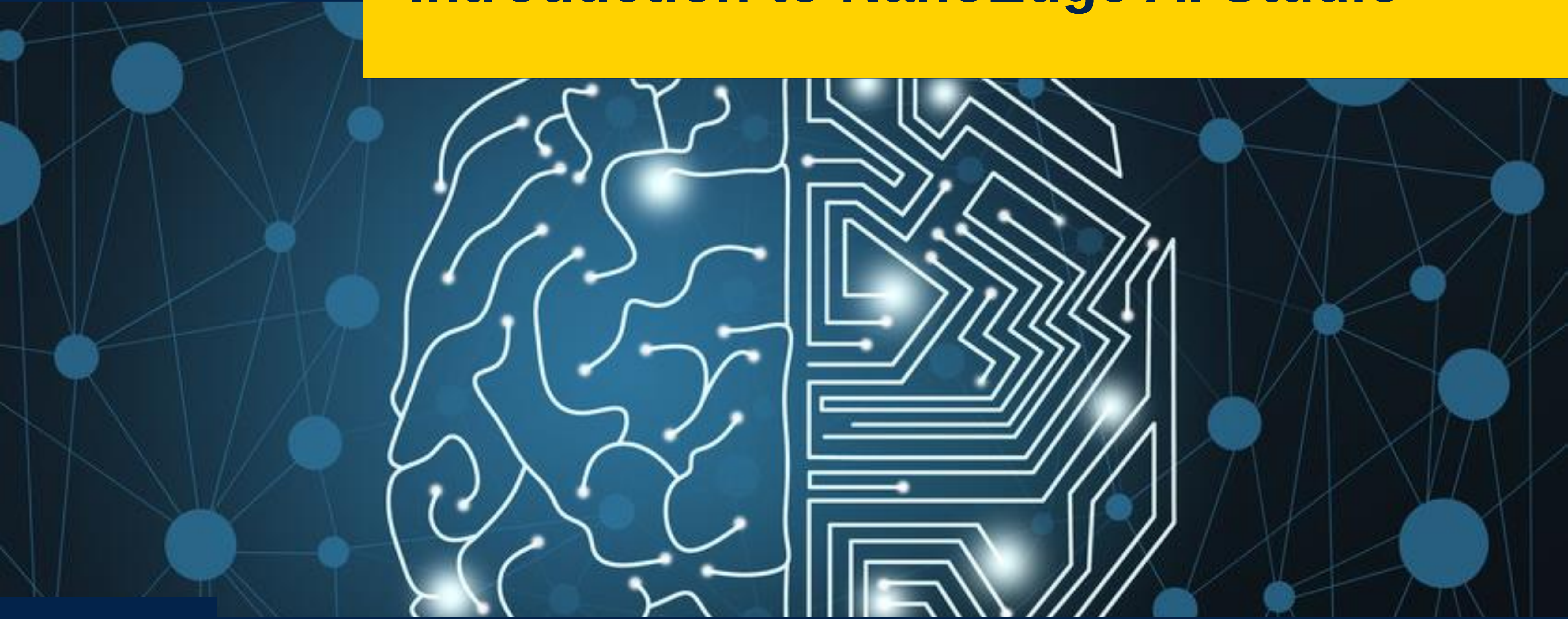


Classification



Regression

Introduction to NanoEdge AI Studio



NanoEdge AI Studio

NanoEdge AI Studio, an automated ML design solution

Generate ultra optimized ML library for any STM32

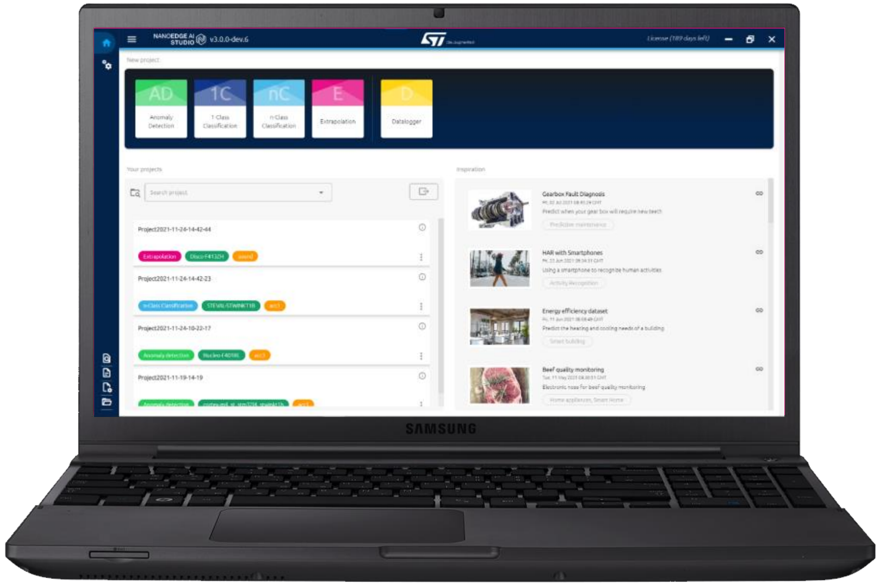
ML Model benchmark to speed up your development time

State of the art of ML implemented continuously: no specific AI skills needed

life.augmented

NanoEdgeAI Studio
For customers without AI expertise

1 Create the library, ONCE



NANOEDGE AI STUDIO v3.0.0-dev.6

License (189 days left)

New project

AD Anomaly Detection

1C 1 Class Classification

nC n-Class Classification

E Extrapolation

D Datalogger

Your projects

Search project

Project2021-11-24-14-42-44 — Extrapolation, Disco-F413ZH, sound

Project2021-11-24-14-42-23 — n-Class Classification, STEVAL-STWINKT1B, accl

Project2021-11-24-10-22-17 — Anomaly detection, Nucleo-F401RE, accl

Project2021-11-19-14-19

Inspiration

Gearbox Fault Diagnosis — Predict when your gear box will require new teeth

HAR with Smartphones — Using a smartphone to recognize human activities

Energy efficiency dataset — Predict the heating and cooling needs of a building

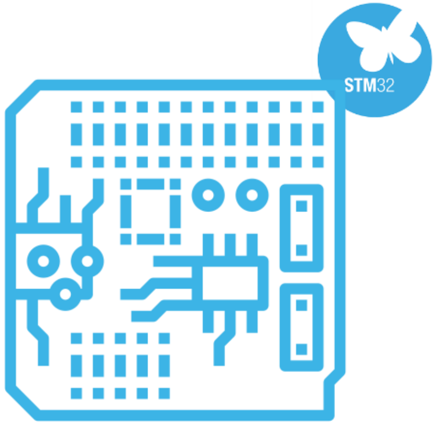
Beef quality monitoring — Electronic nose for beef quality monitoring

Standalone PC (Win/Linux) solution

NANOEDGE AI STUDIO

2 Use the library, MANY

Create and embed a self learning engine

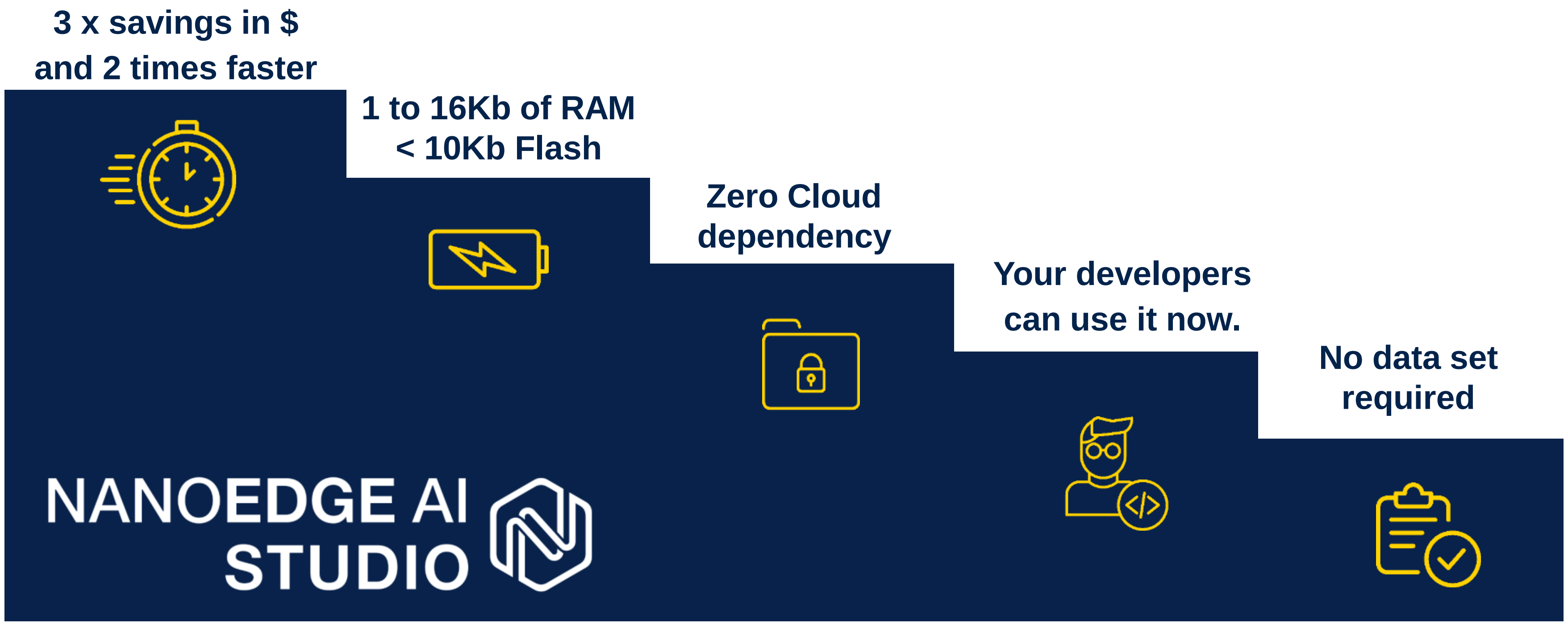


For anomaly detection, the model is self-trained at the Edge

life.augmented

Create a state-of-the-art AI solution in a simple, fast, and affordable way

The power to create Edge AI solution, simply, quickly and affordably.



3 x savings in $ and 2 times faster

1 to 16Kb of RAM < 10Kb Flash

Zero Cloud dependency

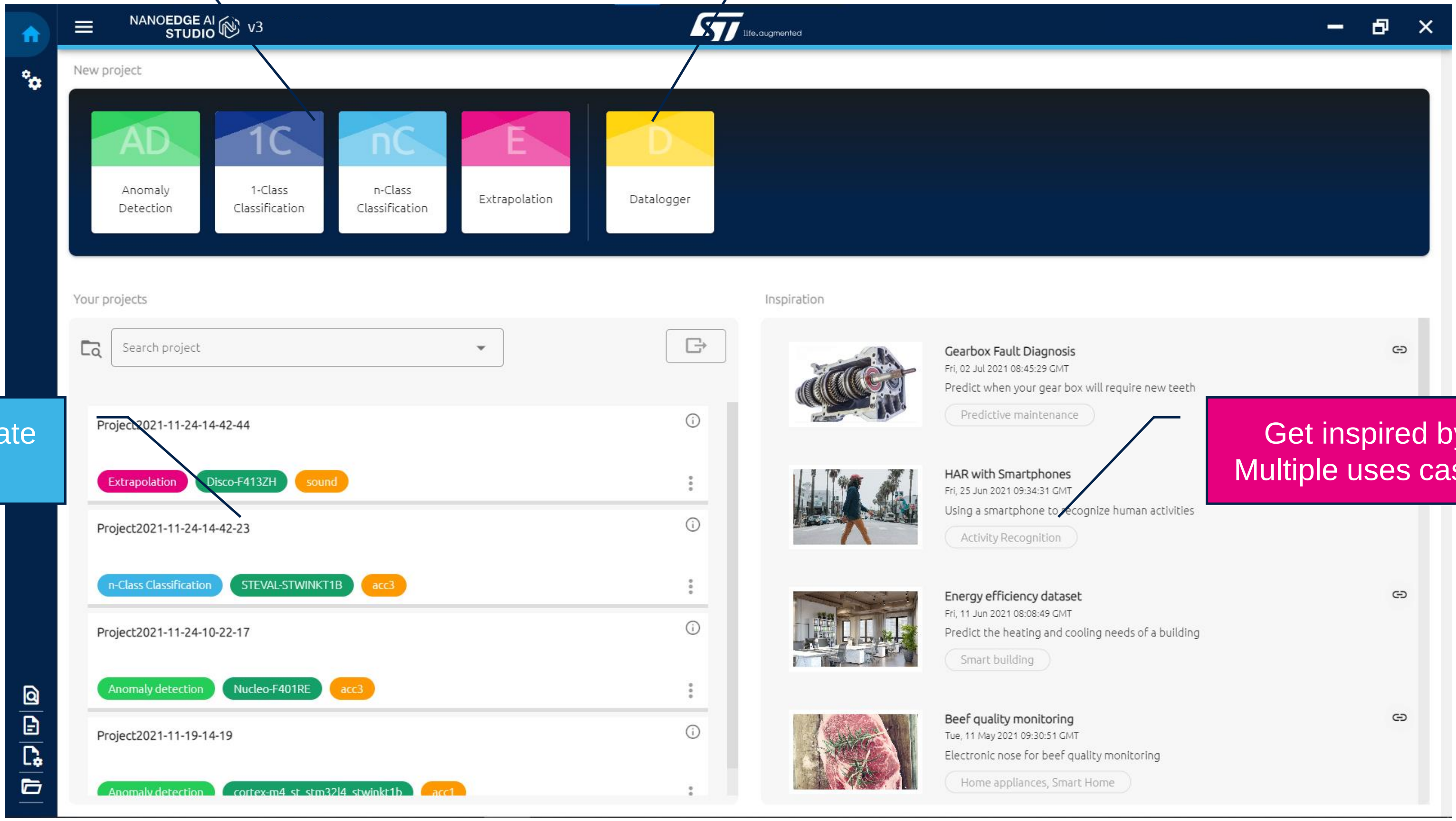
Your developers can use it now.

No data set required

NanoEdge AI Studio V3: New User Interface
More Functions, Better User Experience

New families of Machine Learning algorithms

New Datalogging experience



NANOEDGE AI STUDIO v3

life.augmented

New project

AD Anomaly Detection

1C 1-Class Classification

nC n-Class Classification

E Extrapolation

D Datalogger

Your projects

Search project

Project2021-11-24-14-42-44

Extrapolation | Disco-F413ZH | sound

Project2021-11-24-14-42-23

n-Class Classification | STEVAL-STWINKT1B | acc3

Project2021-11-24-10-22-17

Anomaly detection | Nucleo-F401RE | acc3

Project2021-11-19-14-19

Anomaly detection | cortex-m4_st_stm32l4_stwinkt1b | acc1

Inspiration

Gearbox Fault Diagnosis
Fri, 02 Jul 2021 08:45:29 GMT
Predict when your gear box will require new teeth
Predictive maintenance

HAR with Smartphones
Fri, 25 Jun 2021 09:34:31 GMT
Using a smartphone to recognize human activities
Activity Recognition

Energy efficiency dataset
Fri, 11 Jun 2021 08:08:49 GMT
Predict the heating and cooling needs of a building
Smart building

Beef quality monitoring
Tue, 11 May 2021 09:30:51 GMT
Electronic nose for beef quality monitoring
Home appliances, Smart Home

Easily retrieve or create projects

Get inspired by Multiple uses cases

life.augmented

NANOEDGE AI STUDIO

Our Customers Have Increasingly Ambitious Use Cases For Ever More Smart Products

AD
Anomaly Detection

"My pumps are installed in a variety of environments that I can't anticipate.
I want them to autonomously adapt to their target environment and detect anomalies by themselves."

1C
1-Class Classification

"I know exactly how my pumps behave.
I want to detect any outliers."

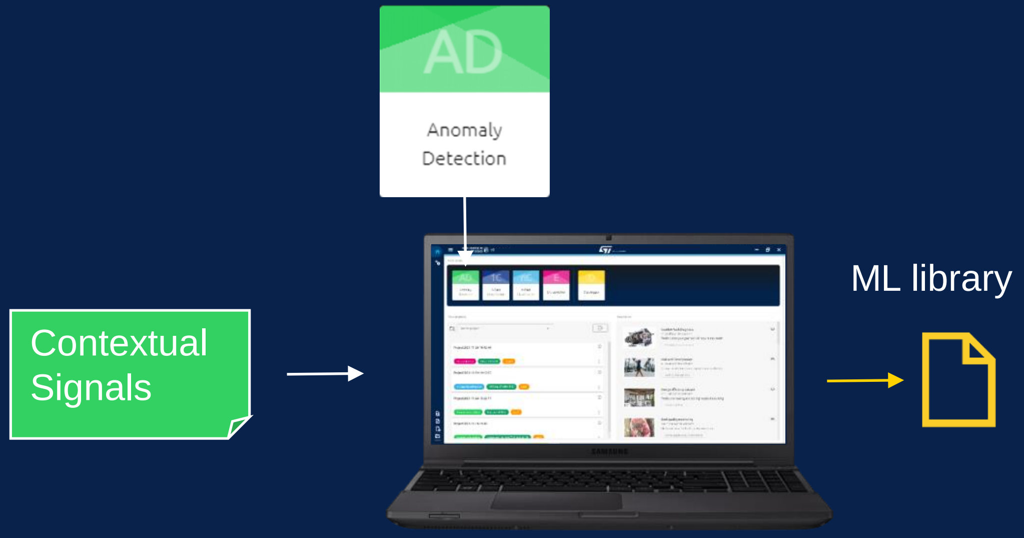
nC
n-Class Classification

"I know the signals when a pump is experiencing, for example, ball bearing or cavitation problems.
I want to know by name what problems are occurring."

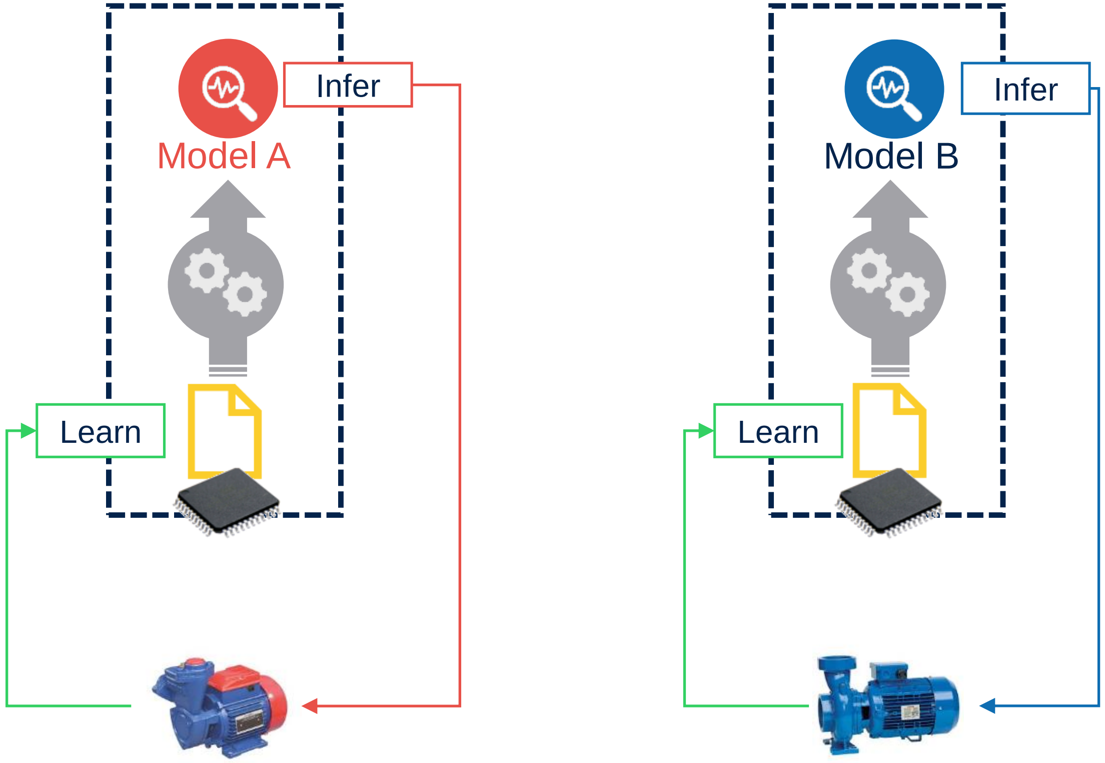
E
Extrapolation

"I know several vibration values of my machine.
I want to anticipate when a specific vibration level will be reached so that I have time to take corrective actions before reaching that limit."

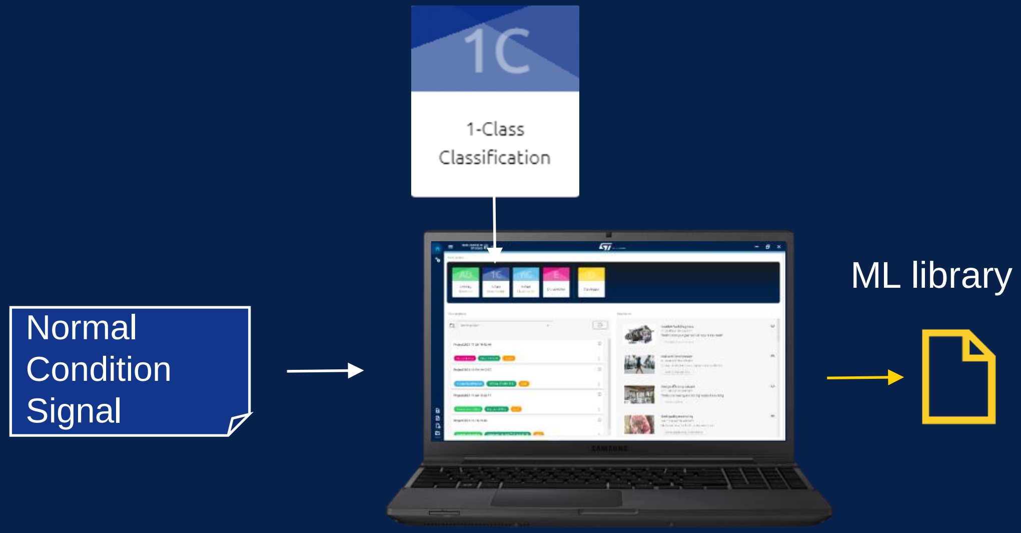
Step 1 **(PC Side)**
Creation of an
ANOMALY DETECTION
Machine Learning library



Step 2 **(MCU Side)**
Use of an
ANOMALY DETECTION
Machine Learning library



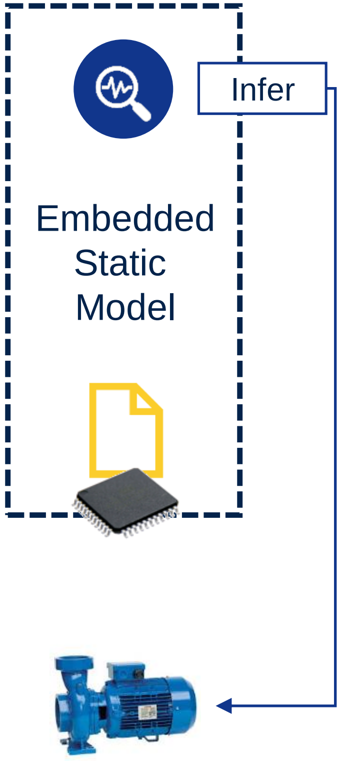
Step 1 **(PC Side)**
Creation of
ONE CLASS CLASSIFICATION (NEW)
Machine Learning library


Normal Condition Signal

1-Class Classification

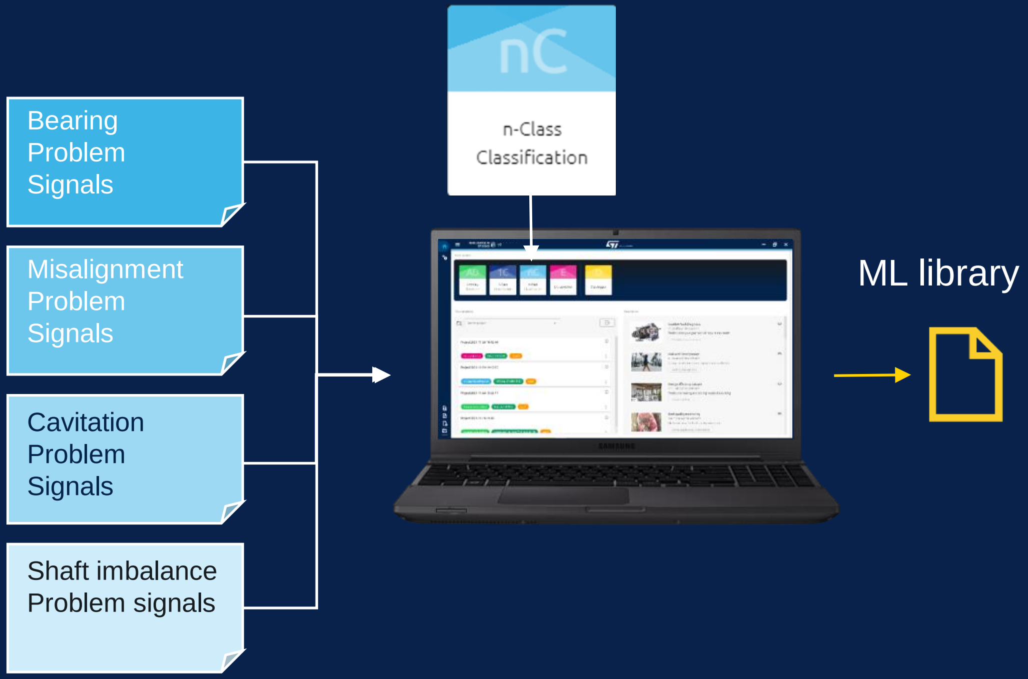
ML library

Step 2 **(MCU Side)**
Use of a
ONE CLASS CLASSIFICATION
Machine Learning library


Infer

Embedded Static Model

Step 1 **(PC Side)**
Creation of an
n CLASS CLASSIFICATION
Machine Learning library



Bearing Problem Signals

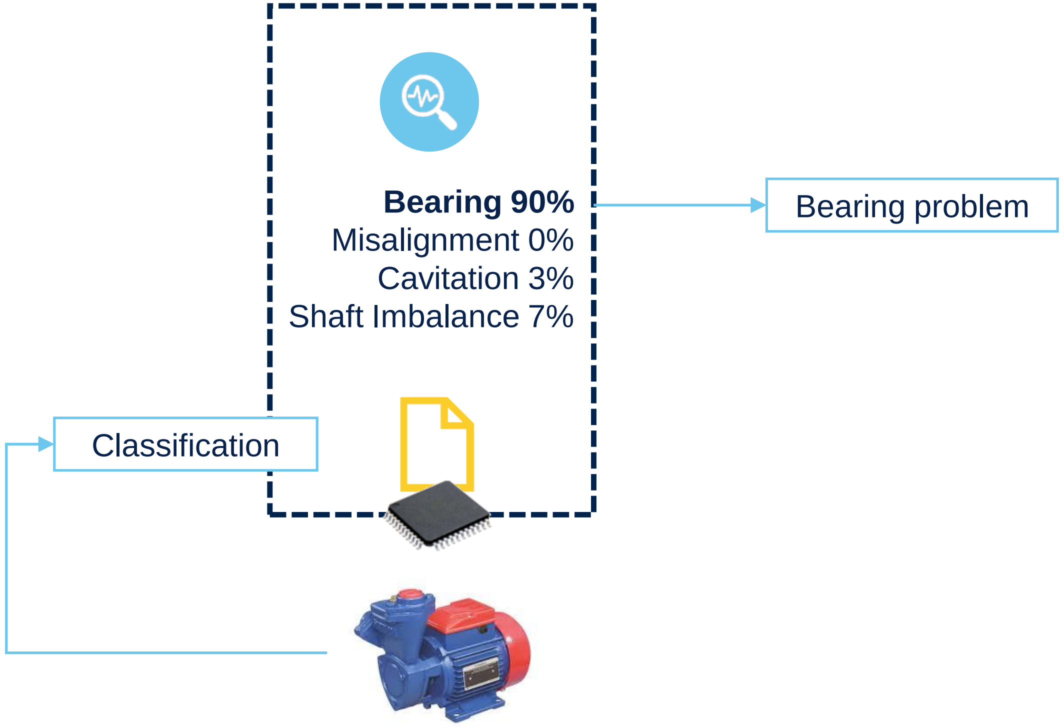
Misalignment Problem Signals

Cavitation Problem Signals

Shaft imbalance Problem signals

ML library

Step 2 **(MCU Side)**
Use of an
n CLASS CLASSIFICATION
Machine Learning library

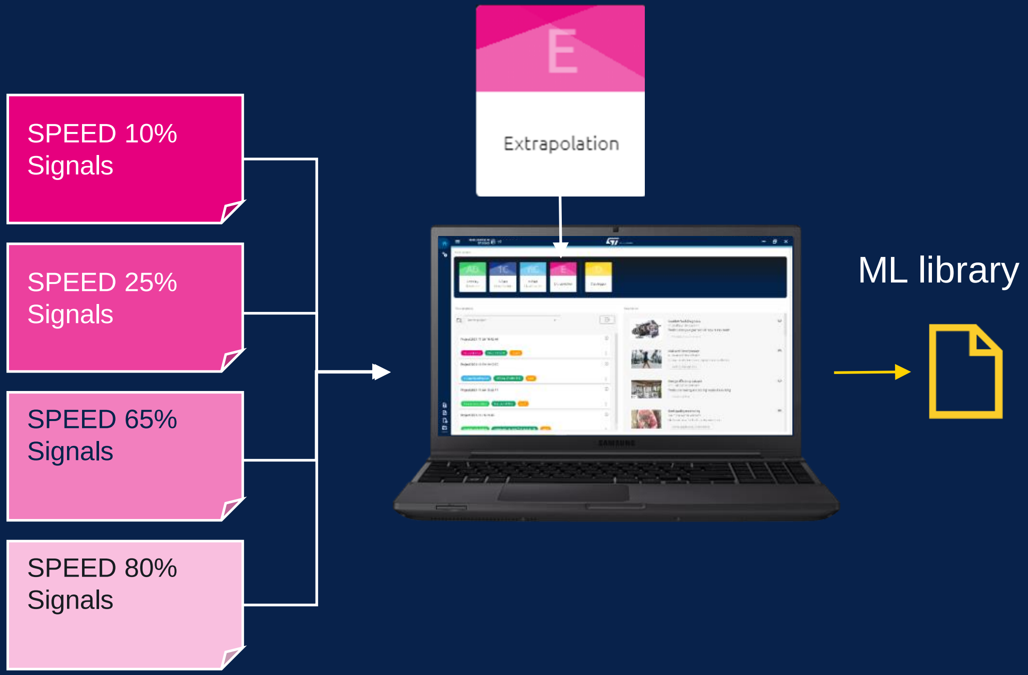


Bearing 90%
Misalignment 0%
Cavitation 3%
Shaft Imbalance 7%

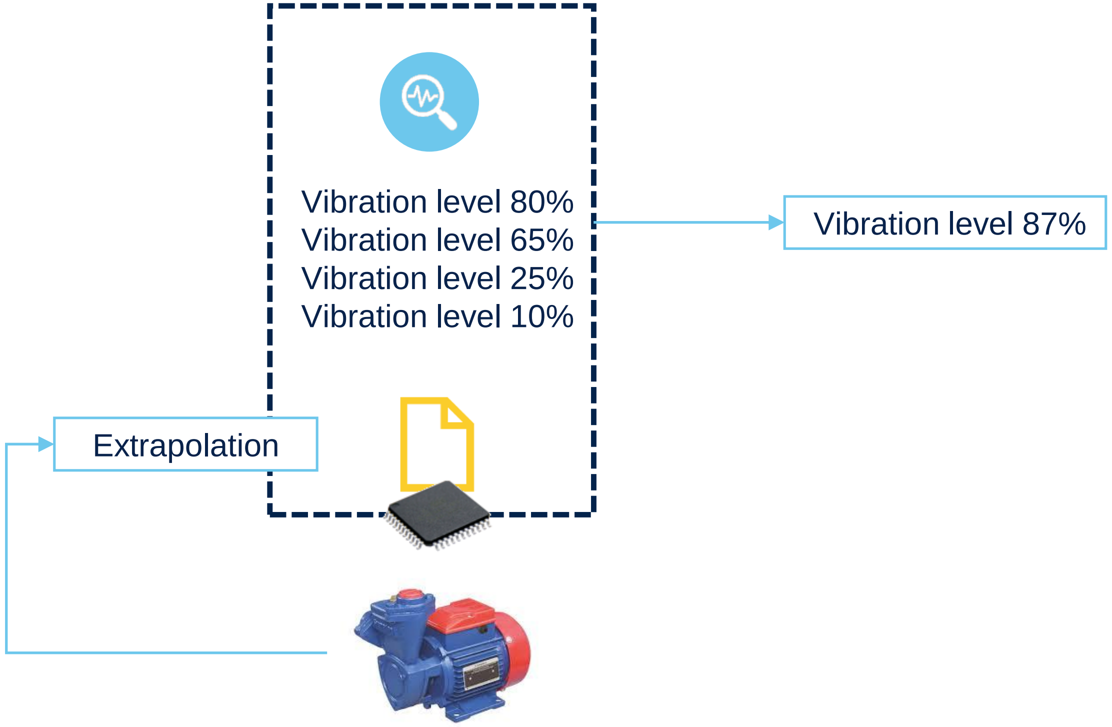
Bearing problem

Classification

Step 1 **(PC Side)**
Creation of an
EXTRAPOLATION (NEW)
Machine Learning library



Step 2 **(MCU Side)**
Use of an
EXTRAPOLATION
Machine Learning library



life.augmented

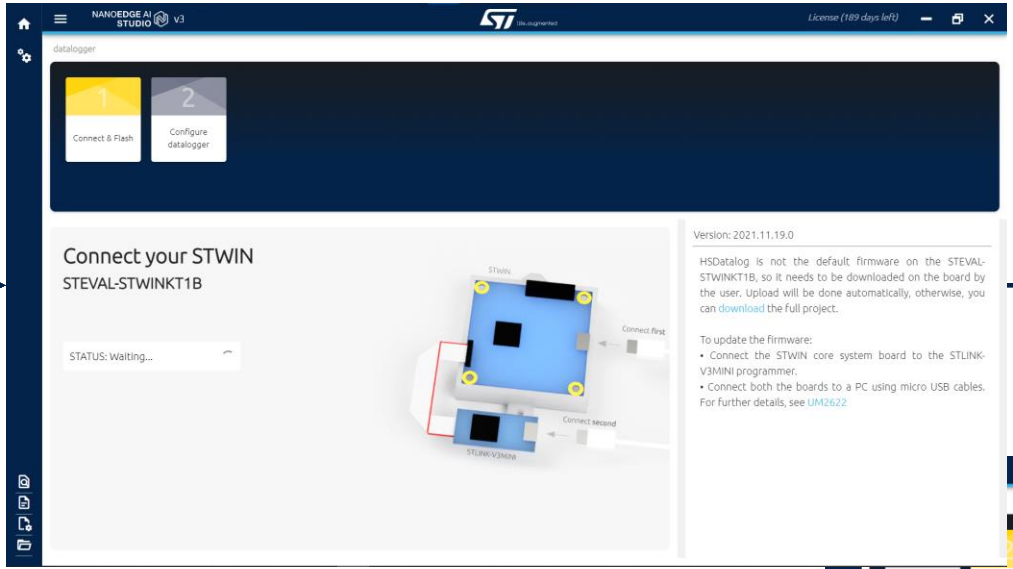
NANOEDGE AI STUDIO

From Idea To Datalogging In A Matter Of Minutes

(NEW)

Datalogger

- Streamlined data logging process
- No code
- All settings done using a graphic interface



Connect your STWIN

STEVAL-STWINKT1B

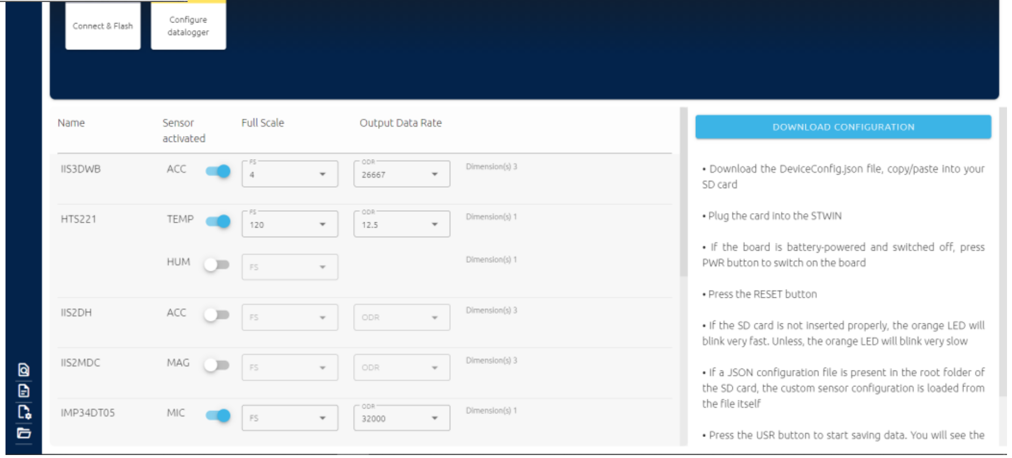
STATUS: Waiting...

Version: 2021.11.19.0

HSDatalog is not the default firmware on the STEVAL-STWINKT1B, so it needs to be downloaded on the board by the user. Upload will be done automatically, otherwise, you can download the full project.

To update the firmware:
- Connect the STWIN core system board to the STLINK-V3MINI programmer.
- Connect both the boards to a PC using micro USB cables.

For further details, see UM2622

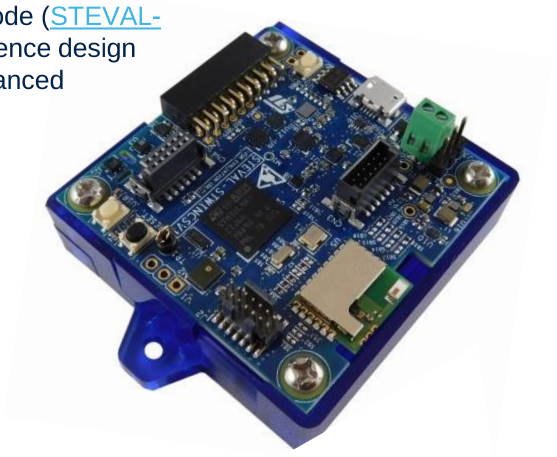


Name	Sensor activated		Full Scale	Output Data Rate	
IIS3DWB	ACC	●	FS 4	ODR 26667	Dimension(s) 3
HTS221	TEMP	●	FS 120	ODR 12.5	Dimension(s) 1
	HUM	○	FS		Dimension(s) 1
IIS2DH	ACC	○	FS	ODR	Dimension(s) 3
IIS2MDC	MAG	○	FS	ODR	Dimension(s) 3
IMP34DT05	MIC	●	FS	ODR 32000	Dimension(s) 1

- Download the DeviceConfig.json file, copy/paste into your SD card
- Plug the card into the STWIN
- If the board is battery-powered and switched off, press PWR button to switch on the board
- Press the RESET button
- If the SD card is not inserted properly, the orange LED will blink very fast. Unless, the orange LED will blink very slow
- If a JSON configuration file is present in the root folder of the SD card, the custom sensor configuration is loaded from the file itself
- Press the USR button to start saving data. You will see the

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

The kit features a core system board with a range of embedded industrial-grade sensors and an ultra-low-power microcontroller



life.augmented

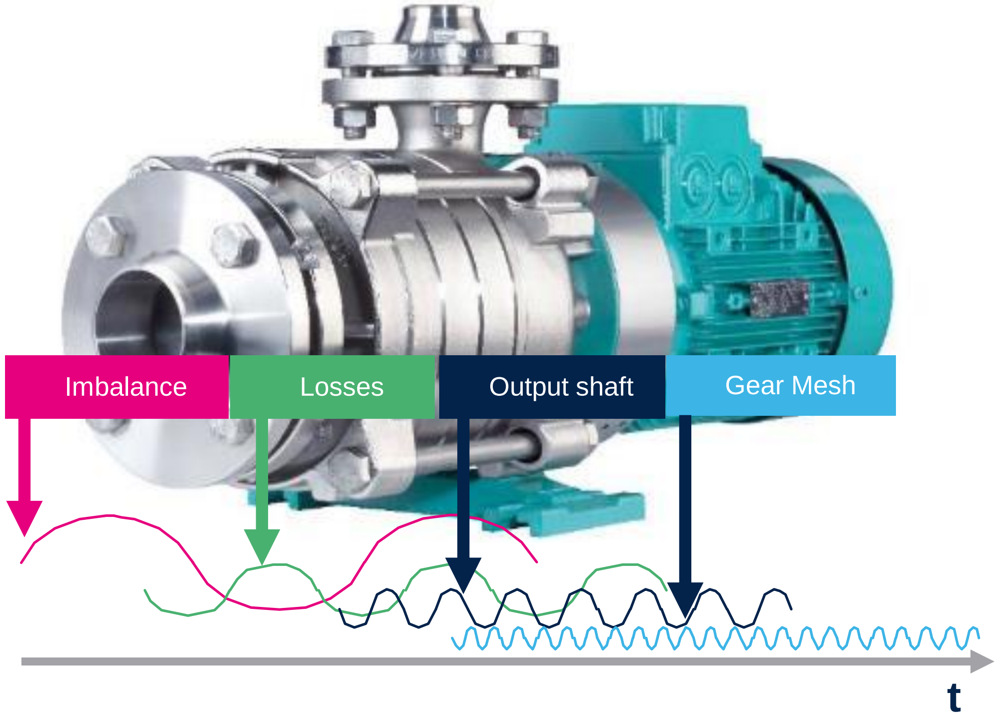
NANOEDGE AI STUDIO

Typical use case
industrial motor monitoring

Any parameter deviation is an indicator of potential failure

Mechanical vibration

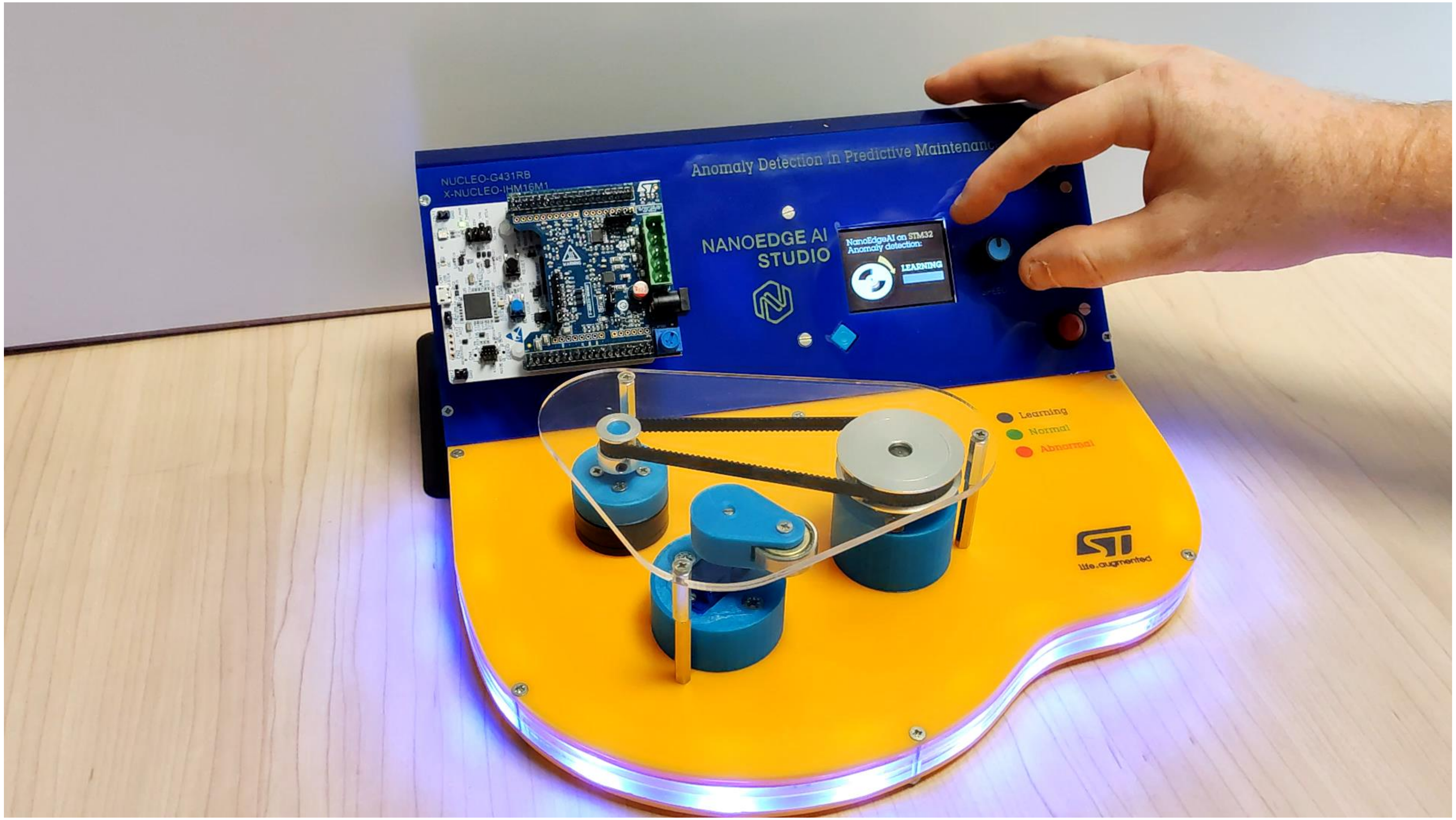
- Displacement
- Speed
- Acceleration
- Acoustic noise
- Angular speed
- Torque



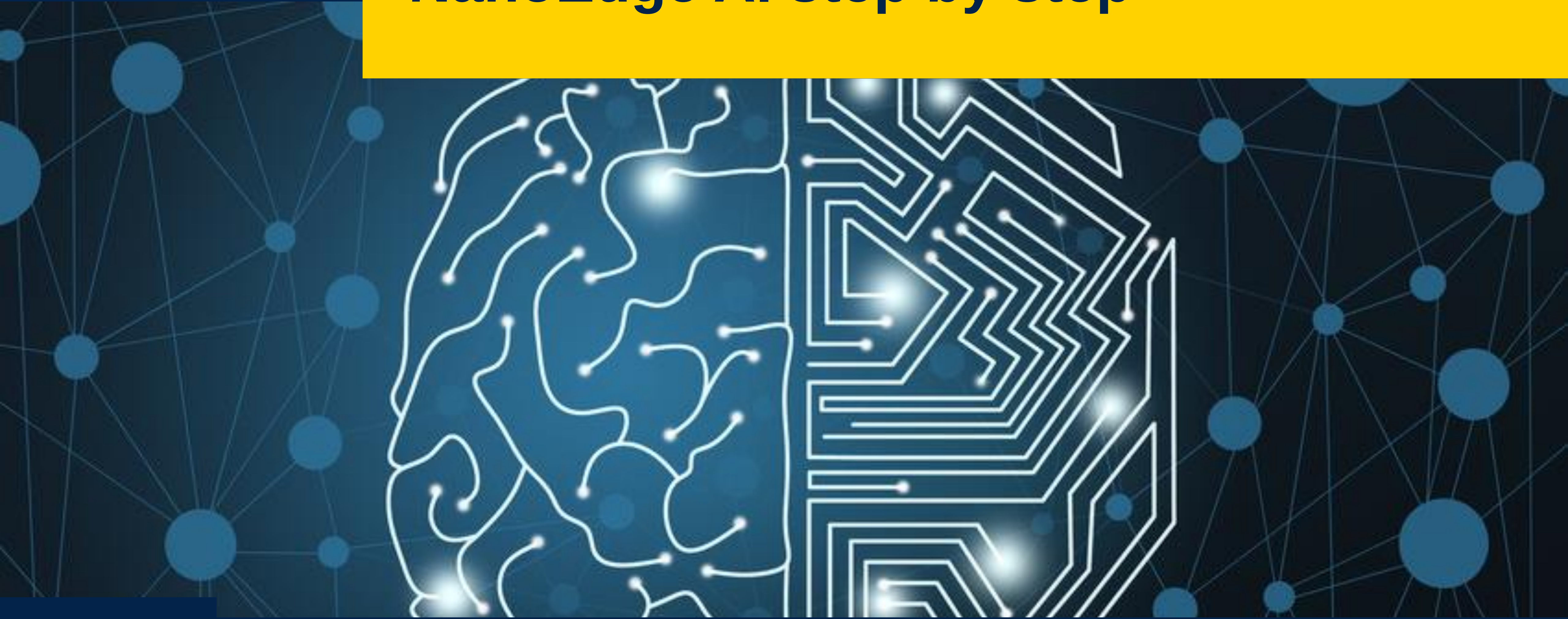
Functions to enable monitoring

- Vibration Capture
- Connectivity
- Processing
- Secure Connections

Demo
Motor Control with NanoEdge AI



NanoEdge AI step by step



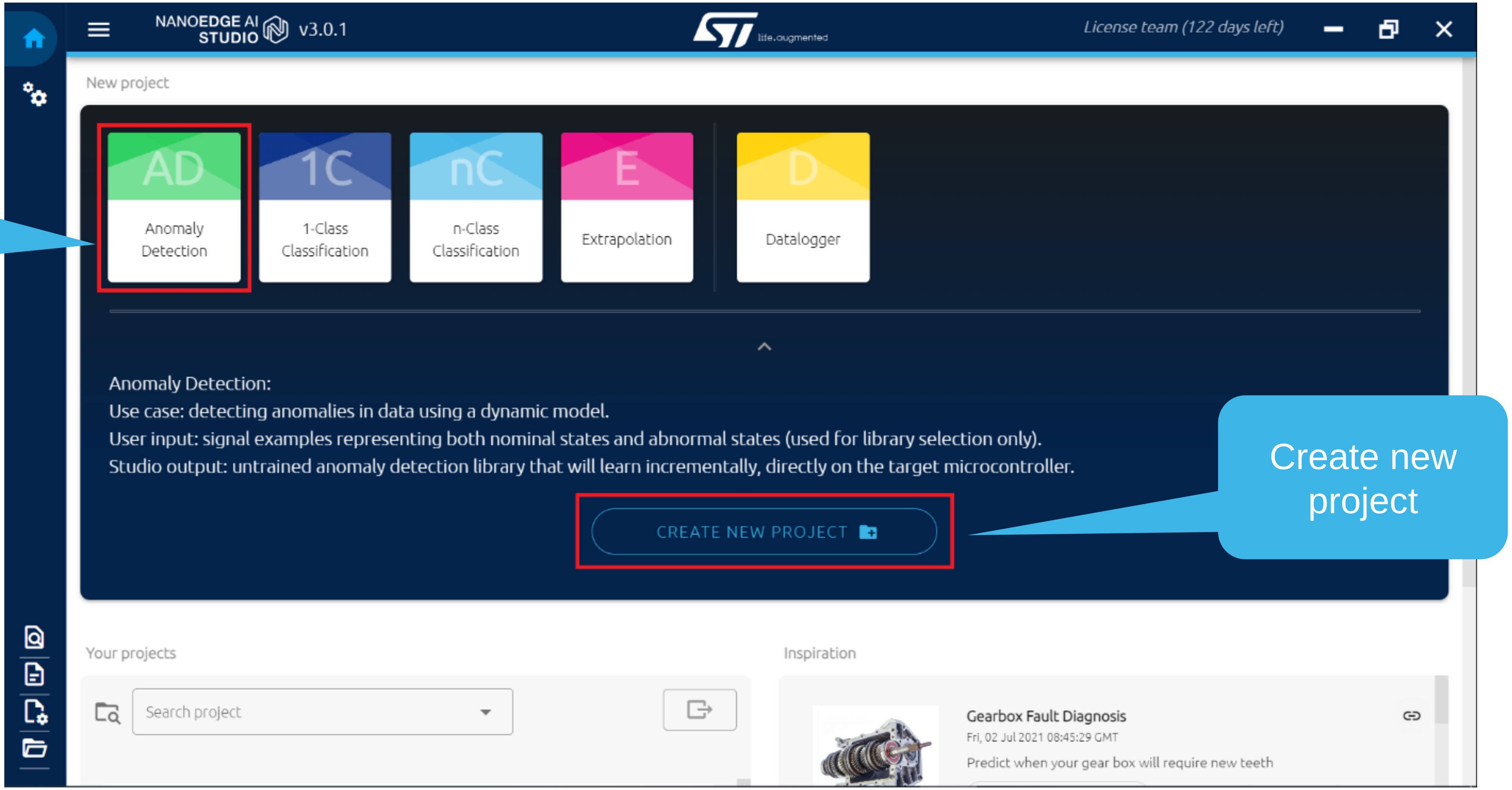
life.augmented

Creating new project

Suppose we want to detect anomaly of an electric motor based on current signal

1

Select Anomaly Detection



Anomaly Detection:
Use case: detecting anomalies in data using a dynamic model.
User input: signal examples representing both nominal states and abnormal states (used for library selection only).
Studio output: untrained anomaly detection library that will learn incrementally, directly on the target microcontroller.

Predict when your gear box will require new teeth

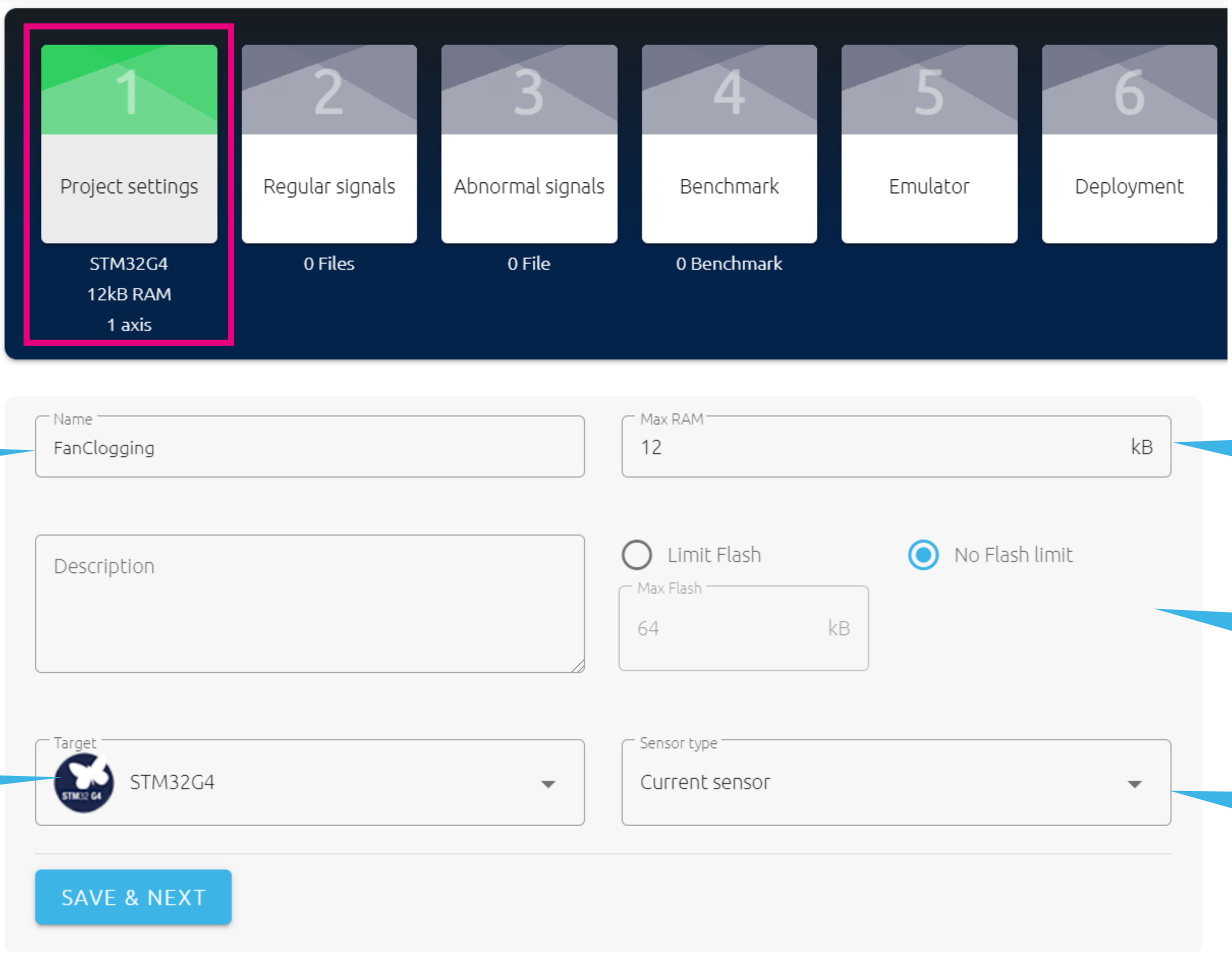
2

Create new project

life.augmented

Project settings

In “Project settings”, fill basic information of the project



STM32G4
12kB RAM
1 axis

0 Files

0 File

0 Benchmark

Name: FanClogging

Max RAM: 12 kB

Description

○ Limit Flash ● No Flash limit

Max Flash: 64 kB

Target: STM32G4

Sensor type: Current sensor

Name of project

Max RAM allocated for AI library

Max Flash allocated for AI library

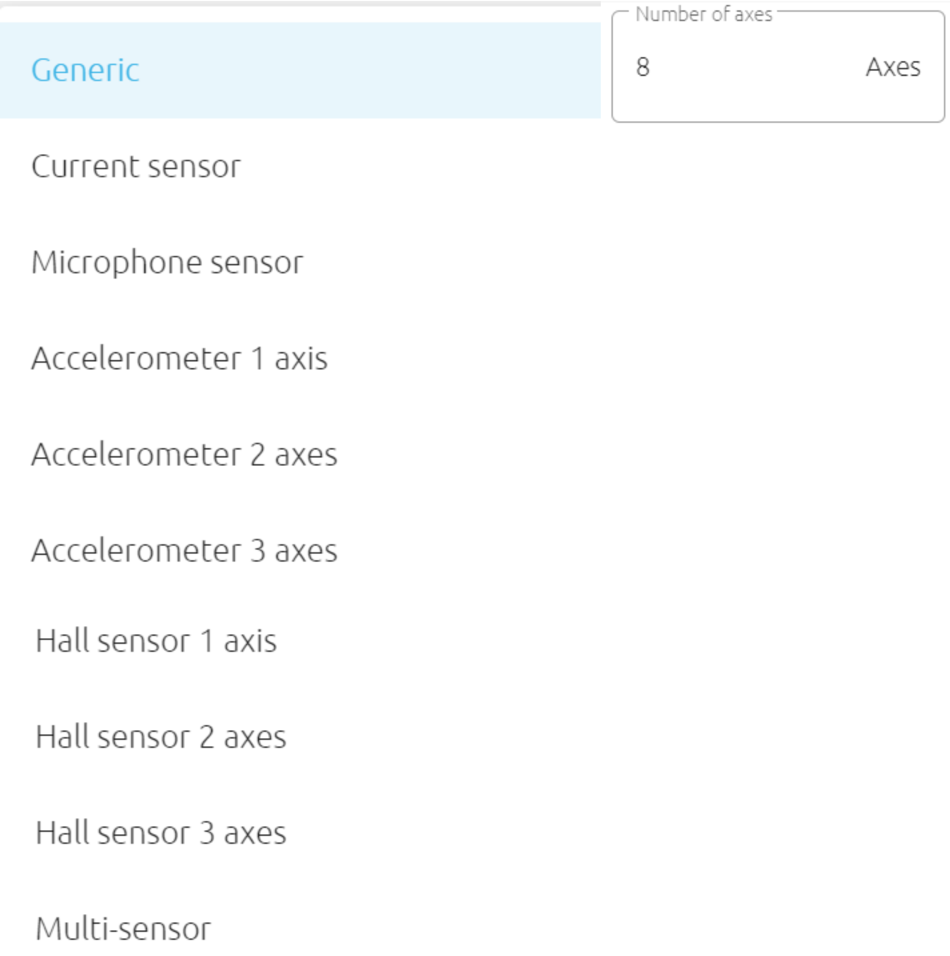
Target ISPU / MCU / board

Sensor type for input signal

Project settings

List of supported sensor types in NanoEdge AI

- Some typical sensor types are supported
- Also possible to combine different sensor types in the same input thanks to “**Generic**” type.



Generic

Number of axes
8 Axes

Current sensor

Microphone sensor

Accelerometer 1 axis

Accelerometer 2 axes

Accelerometer 3 axes

Hall sensor 1 axis

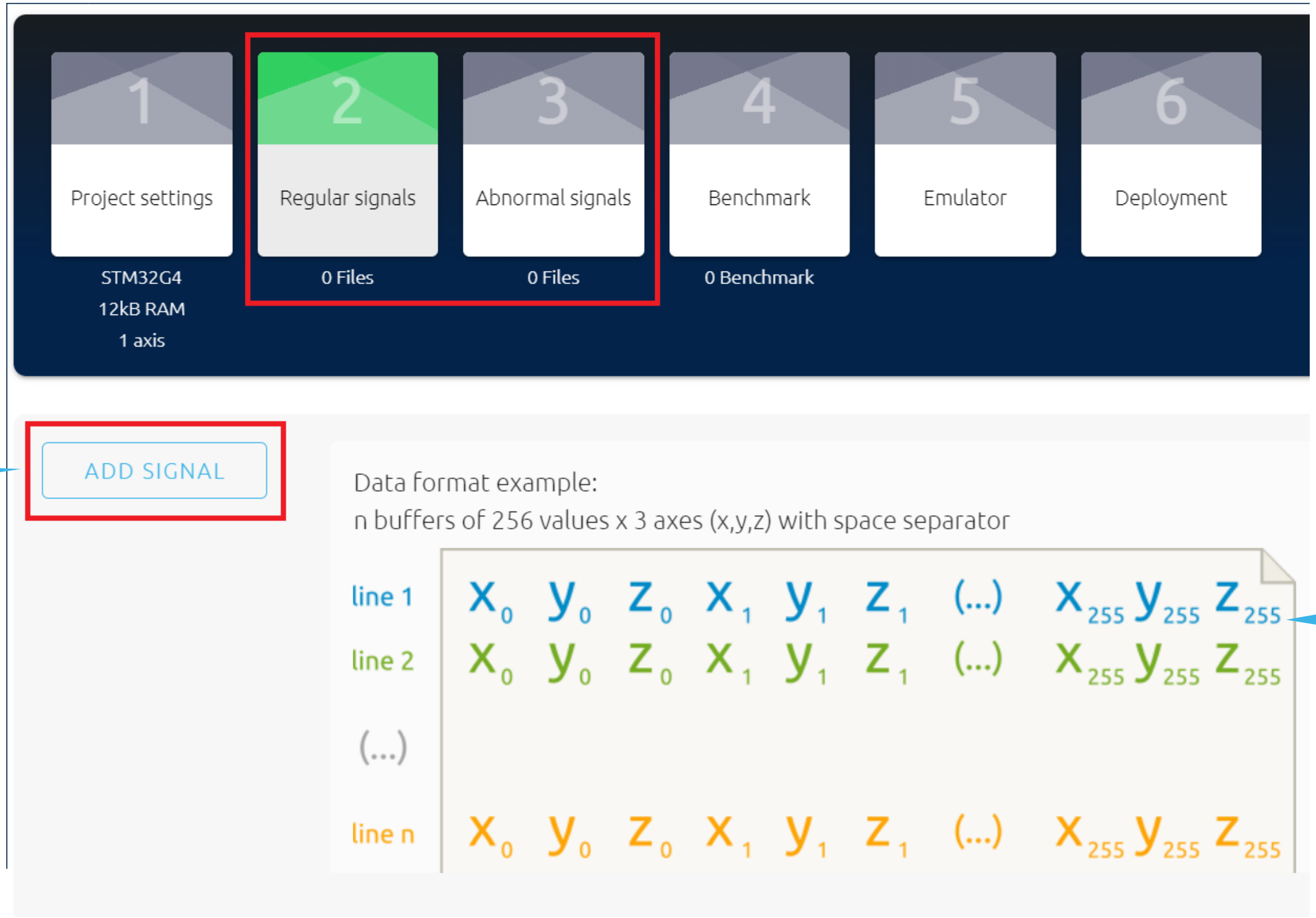
Hall sensor 2 axes

Hall sensor 3 axes

Multi-sensor

Adding signals

Adding signals for both “Regular” and “Abnormal” conditions



Data format example:
n buffers of 256 values x 3 axes (x,y,z) with space separator

line 1 x_0 y_0 z_0 x_1 y_1 z_1 (...) x_{255} y_{255} z_{255}

line 2 x_0 y_0 z_0 x_1 y_1 z_1 (...) x_{255} y_{255} z_{255}

(...)

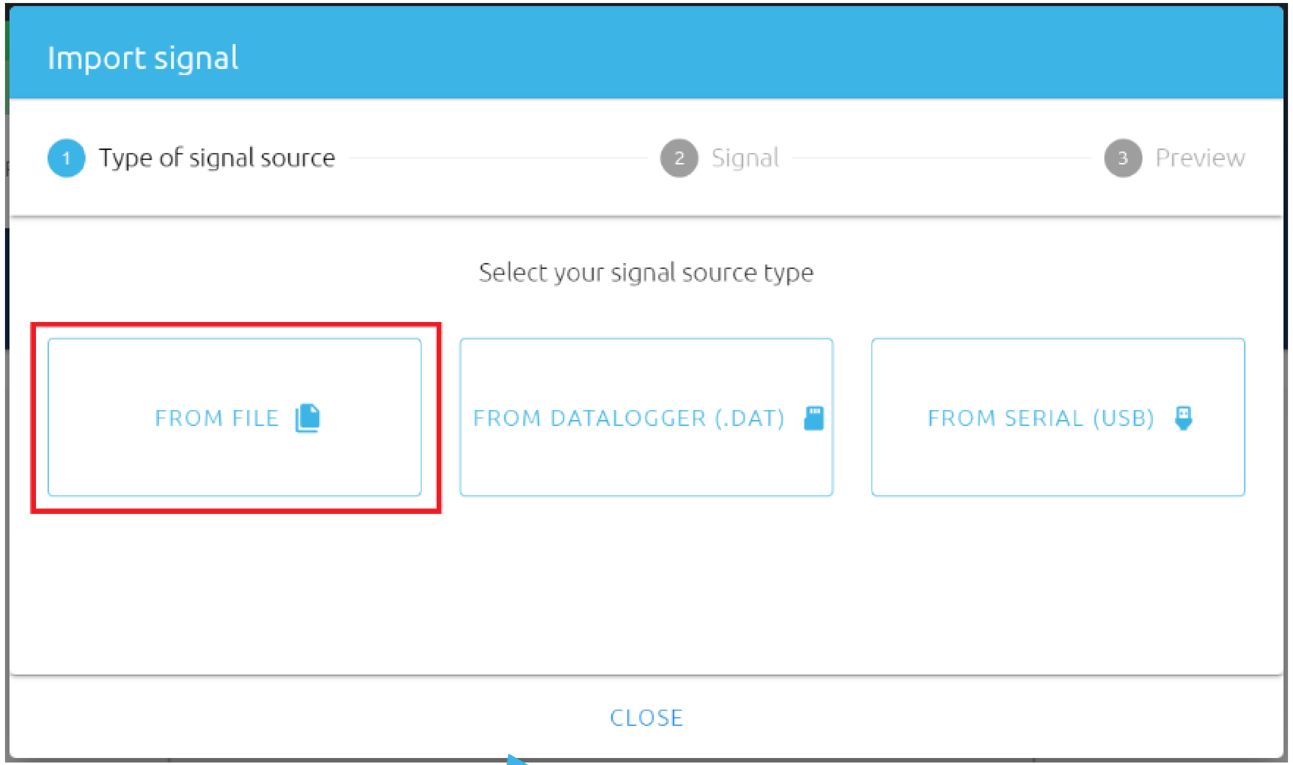
line n x_0 y_0 z_0 x_1 y_1 z_1 (...) x_{255} y_{255} z_{255}

Click to add signal

Signal format is explained here

Adding signals

3 possible types of signal sources: “From file”, “From Datalogger”, “From Serial (USB)”

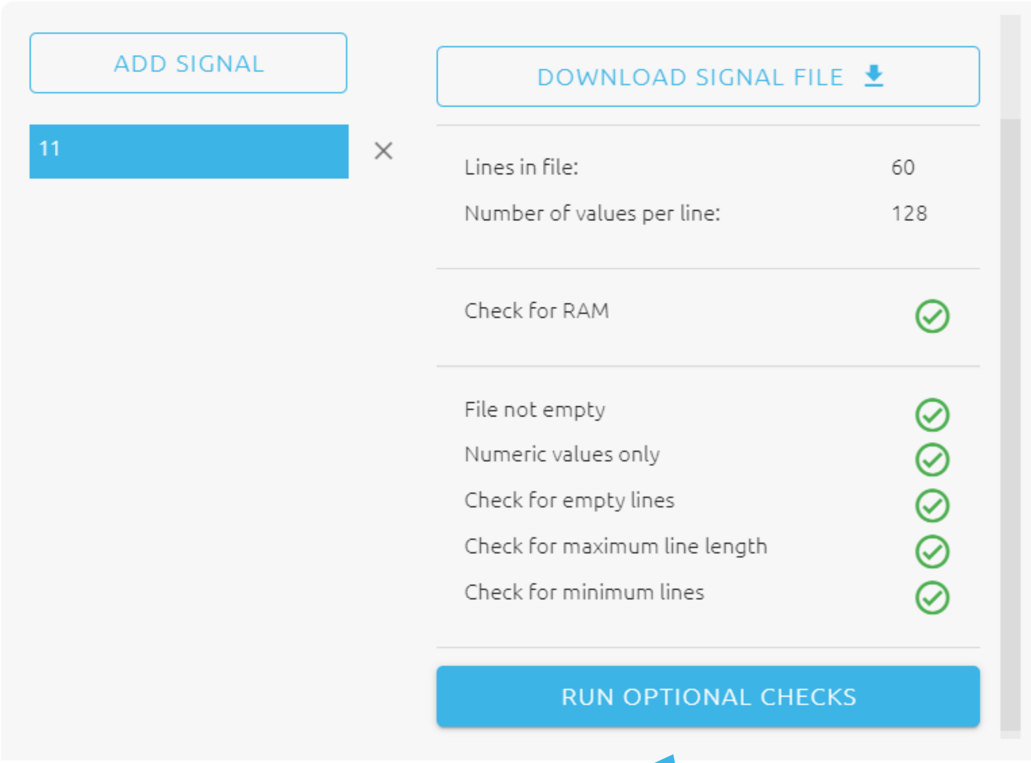


Import signal

1 Type of signal source — 2 Signal — 3 Preview

Select your signal source type

Select source type



Lines in file: 60

Number of values per line: 128

Check for RAM ✓

File not empty ✓

Numeric values only ✓

Check for empty lines ✓

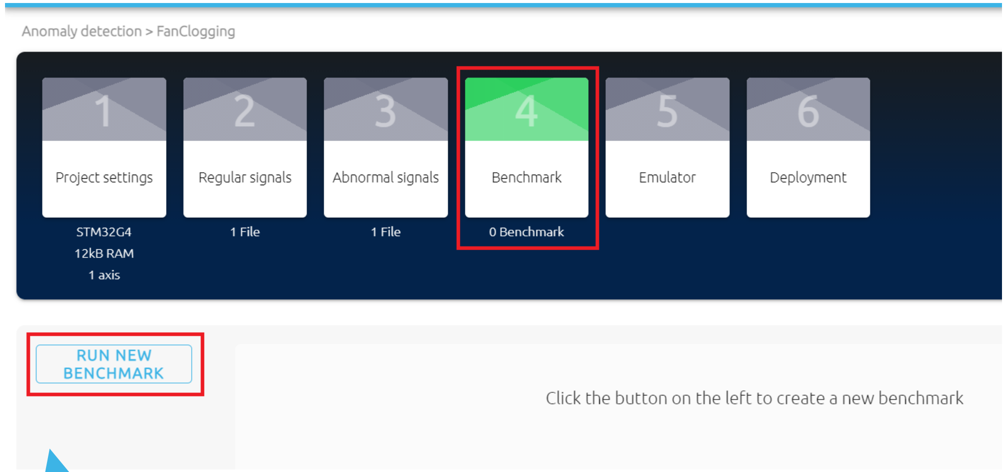
Check for maximum line length ✓

Check for minimum lines ✓

Check imported signals

Benchmarking of NanoEdge AI Library

Run benchmark to generate an AI model which best fit the provided signals

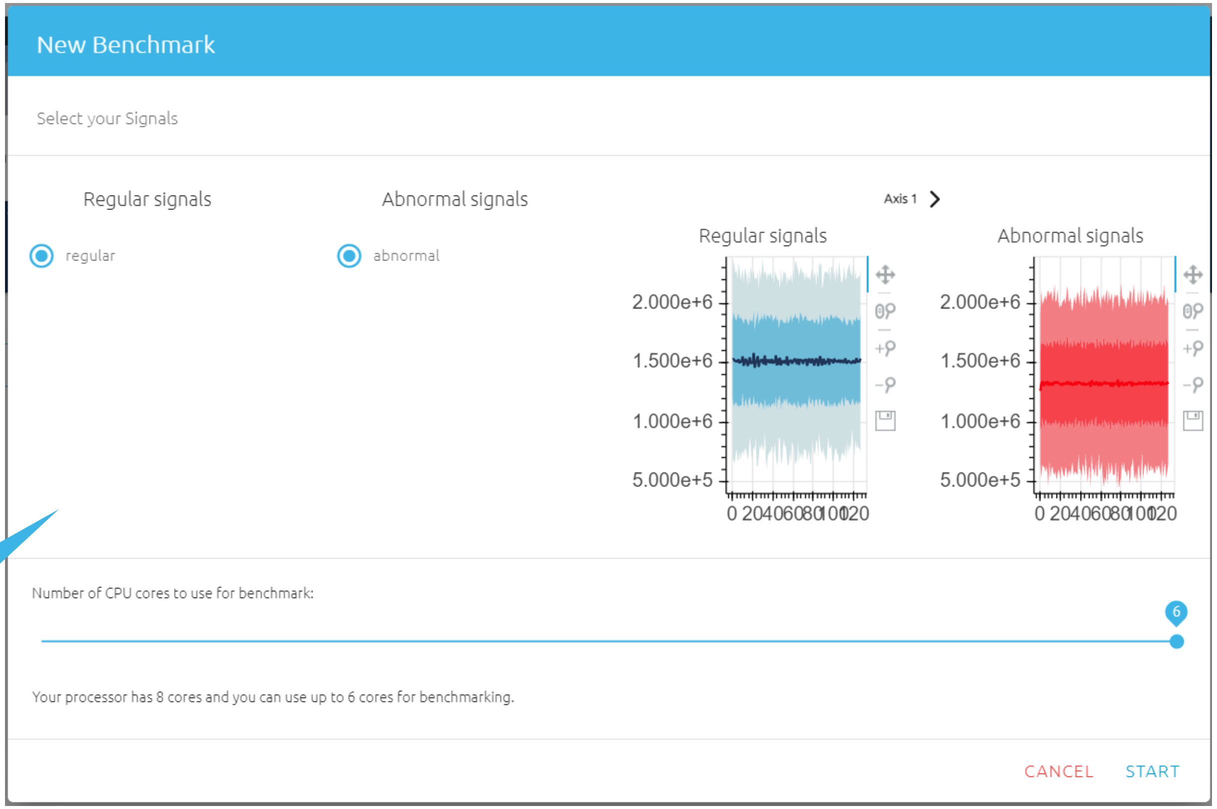


Anomaly detection > FanClogging

1 Project settings	2 Regular signals	3 Abnormal signals	4 Benchmark	5 Emulator	6 Deployment
STM32G4 12kB RAM 1 axis	1 File	1 File	0 Benchmark		

Click the button on the left to create a new benchmark

Run new benchmark



New Benchmark

Select your Signals

Regular signals: ● regular

Abnormal signals: ● abnormal

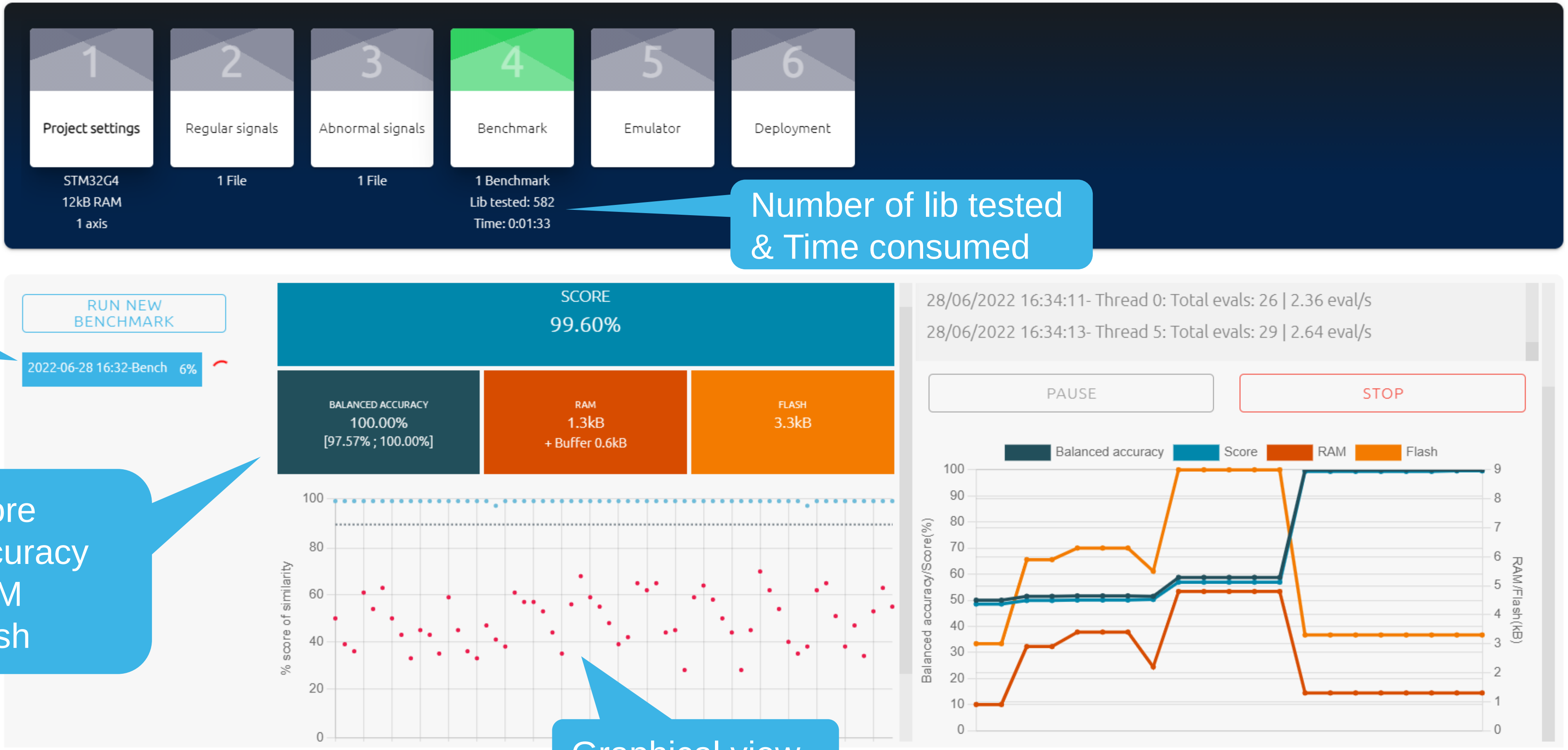
Number of CPU cores to use for benchmark: 6

Your processor has 8 cores and you can use up to 6 cores for benchmarking.

Choose signals and number of CPU core for benchmark

Benchmarking of NanoEdge AI Library

We can monitor different metrics of the current best model during the process of benchmark



1 Project settings
STM32G4
12kB RAM
1 axis

2 Regular signals
1 File

3 Abnormal signals
1 File

4 Benchmark
1 Benchmark
Lib tested: 582
Time: 0:01:33

5 Emulator

6 Deployment

RUN NEW BENCHMARK

2022-06-28 16:32-Bench 6%

SCORE
99.60%

BALANCED ACCURACY
100.00%
[97.57% ; 100.00%]

RAM
1.3kB
+ Buffer 0.6kB

FLASH
3.3kB

28/06/2022 16:34:11- Thread 0: Total evals: 26 | 2.36 eval/s
28/06/2022 16:34:13- Thread 5: Total evals: 29 | 2.64 eval/s

PAUSE

STOP

Number of lib tested & Time consumed

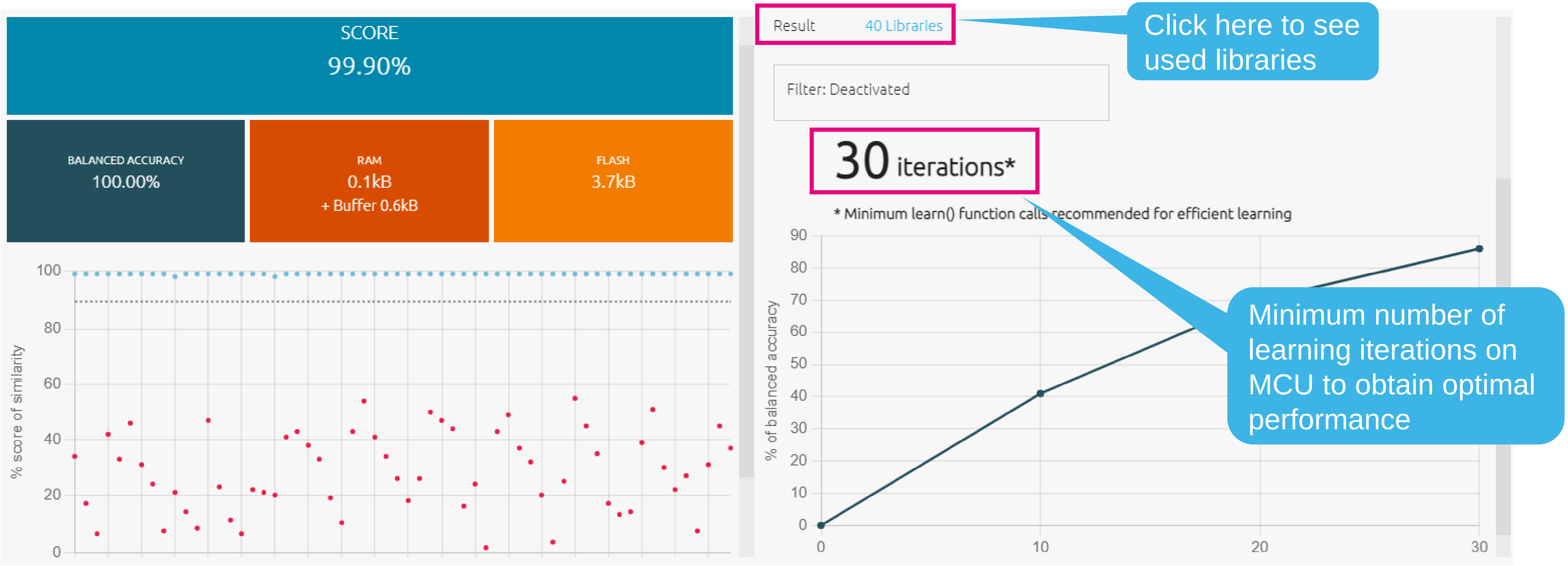
Progress

Score
Accuracy
RAM
Flash

Graphical view

Benchmarking of NanoEdge AI Library

After the benchmark, the recommended minimum learning iterations will appear



SCORE
99.90%

BALANCED ACCURACY
100.00%

RAM
0.1kB
+ Buffer 0.6kB

FLASH
3.7kB

Result 40 Libraries

Filter: Deactivated

30 iterations*

* Minimum learn() function calls recommended for efficient learning

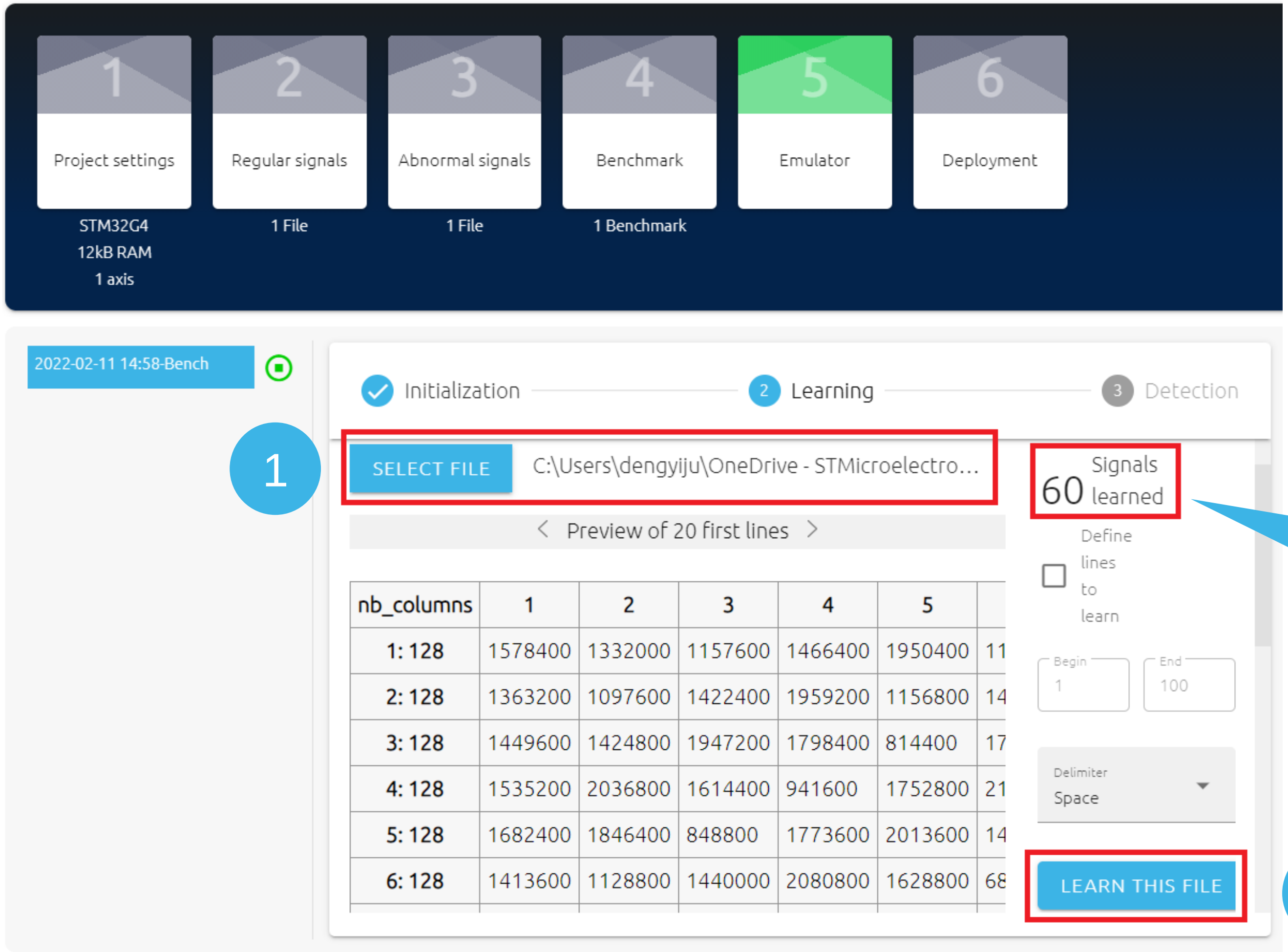
Click here to see used libraries

Minimum number of learning iterations on MCU to obtain optimal performance

life.augmented

Validating the library by emulator

NanoEdge AI provides emulator to test the AI library without creating any embedded software.



Project settings: STM32G4, 12kB RAM, 1 axis

Regular signals: 1 File

Abnormal signals: 1 File

Benchmark: 1 Benchmark

1. SELECT FILE C:\Users\dengyiju\OneDrive - STMicroelectro…

60 Signals learned

☐ Define lines to learn

Begin 1

End 100

Delimiter: Space

Preview of 20 first lines

nb_columns	1	2	3	4	5	
1: 128	1578400	1332000	1157600	1466400	1950400	11
2: 128	1363200	1097600	1422400	1959200	1156800	14
3: 128	1449600	1424800	1947200	1798400	814400	17
4: 128	1535200	2036800	1614400	941600	1752800	21
5: 128	1682400	1846400	848800	1773600	2013600	14
6: 128	1413600	1128800	1440000	2080800	1628800	68

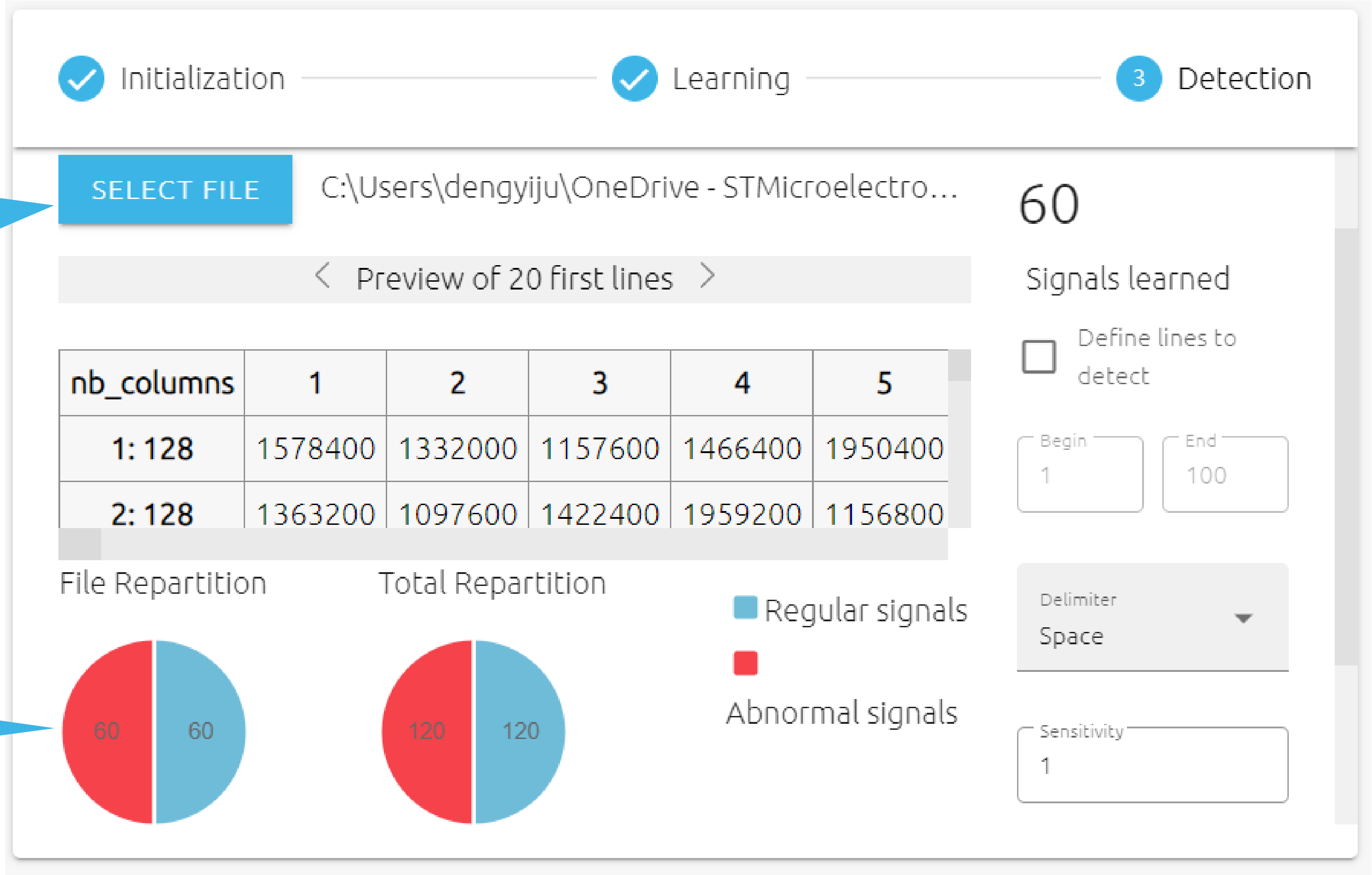
2. LEARN THIS FILE

Learn at least 30 normal signals from file or from serial (USB)

Validating the library by emulator

Go to detection after learning minimum number of normal signals

Select signal file for detection or detect in real-time from serial USB



C:\Users\dengyiju\OneDrive - STMicroelectro…

nb_columns	1	2	3	4	5
1: 128	1578400	1332000	1157600	1466400	1950400
2: 128	1363200	1097600	1422400	1959200	1156800

60

Signals learned

□ Define lines to detect

Begin: 1

End: 100

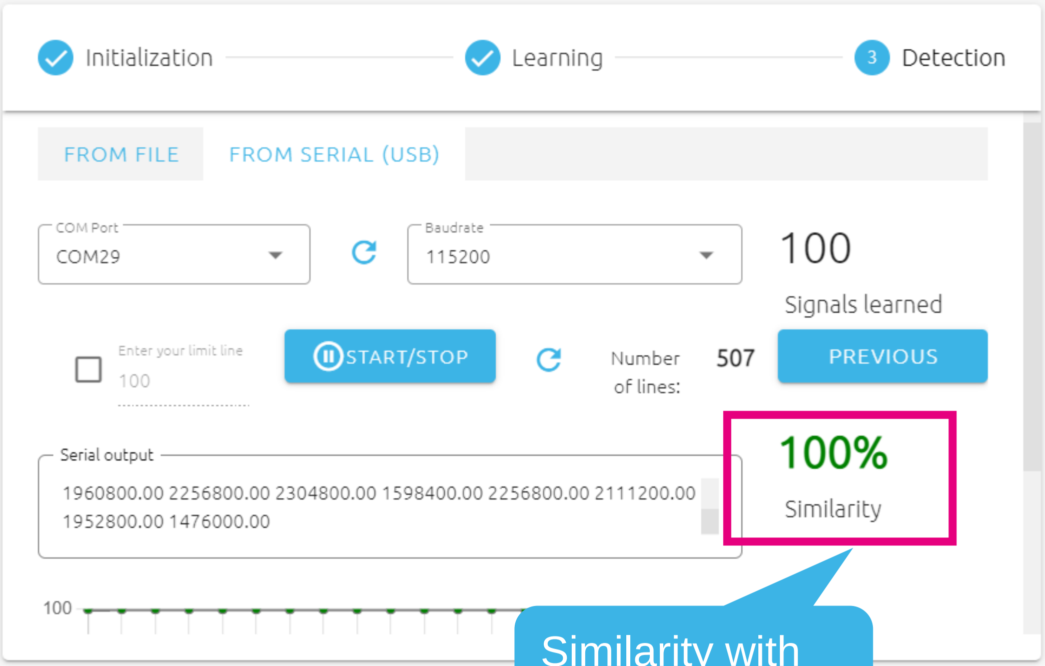
Delimiter: Space

Sensitivity: 1

Results of detection

Validating the library by emulator

We can also test in real-time the signals received from serial USB



✓ Initialization — ✓ Learning — 3 Detection

FROM FILE | FROM SERIAL (USB)

COM Port: COM29

Baudrate: 115200

100
Signals learned

□ Enter your limit line: 100

START/STOP

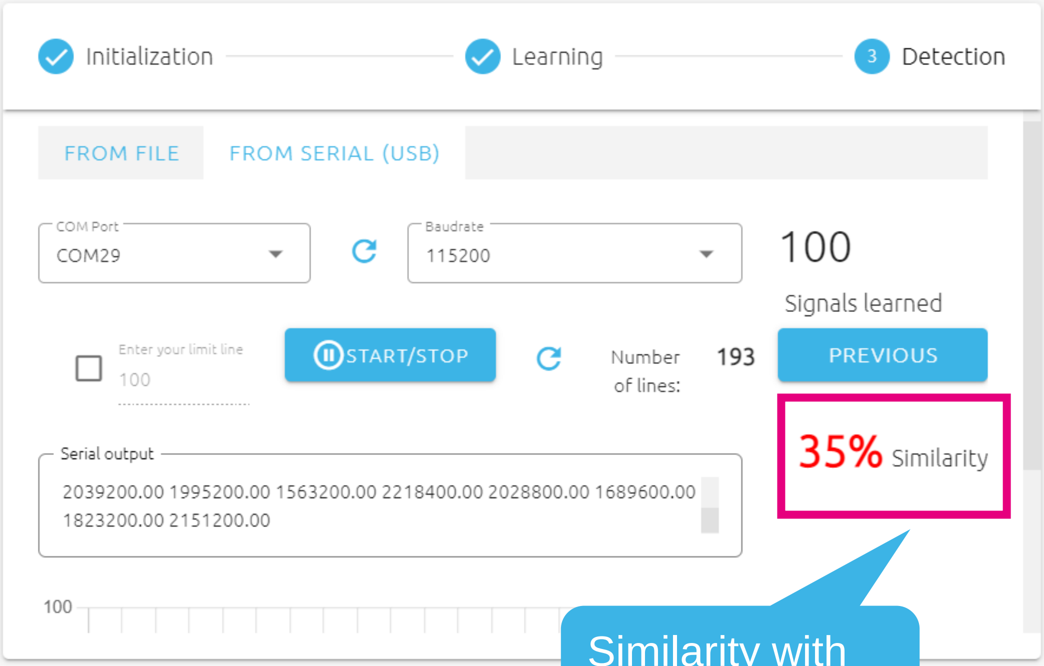
Number of lines: 507

PREVIOUS

100%
Similarity

Serial output:
```
1960800.00 2256800.00 2304800.00 1598400.00 2256800.00 2111200.00
1952800.00 1476000.00
```

Similarity with normal behavior



✓ Initialization — ✓ Learning — 3 Detection

FROM FILE | FROM SERIAL (USB)

COM Port: COM29

Baudrate: 115200

100
Signals learned

□ Enter your limit line: 100

START/STOP

Number of lines: 193

PREVIOUS

35% Similarity

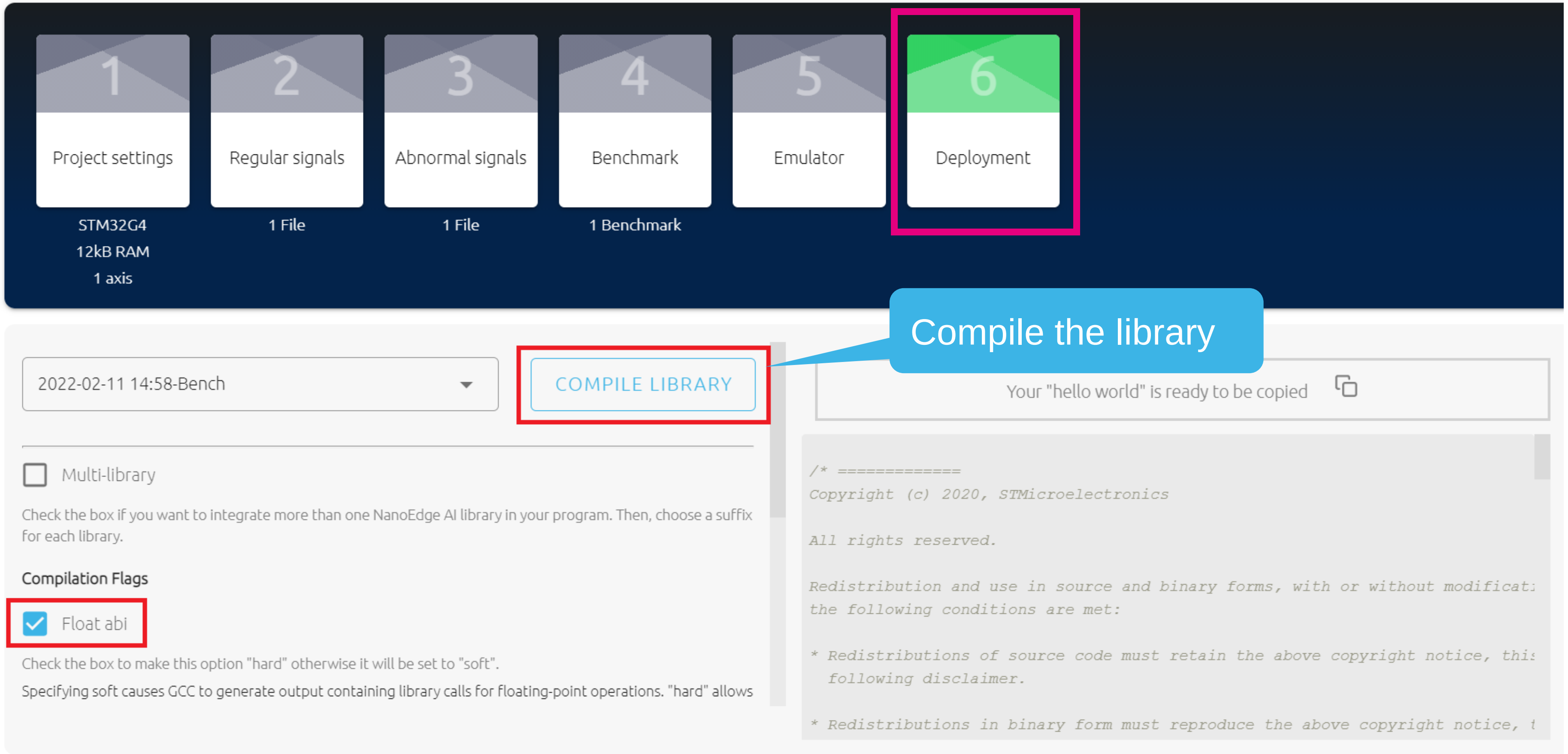
Serial output:
```
2039200.00 1995200.00 1563200.00 2218400.00 2028800.00 1689600.00
1823200.00 2151200.00
```

Similarity with normal behavior

Similarity > threshold (e.g. 90%) => Normal
Similarity < threshold (e.g. 90%) => Abnormal

Deployment of NanoEdge AI library

After the library being validated, we can go to “Deployment” and compile the model into C library



STM32G4
12kB RAM
1 axis

1 File

1 File

1 Benchmark

2022-02-11 14:58-Bench

□ Multi-library

Check the box if you want to integrate more than one NanoEdge AI library in your program. Then, choose a suffix for each library.

Compilation Flags

☑ Float abi

Check the box to make this option "hard" otherwise it will be set to "soft".

Specifying soft causes GCC to generate output containing library calls for floating-point operations. "hard" allows

Your "hello world" is ready to be copied

```
/* =============
Copyright (c) 2020, STMicroelectronics

All rights reserved.

Redistribution and use in source and binary forms, with or without modificat:
the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, t
```

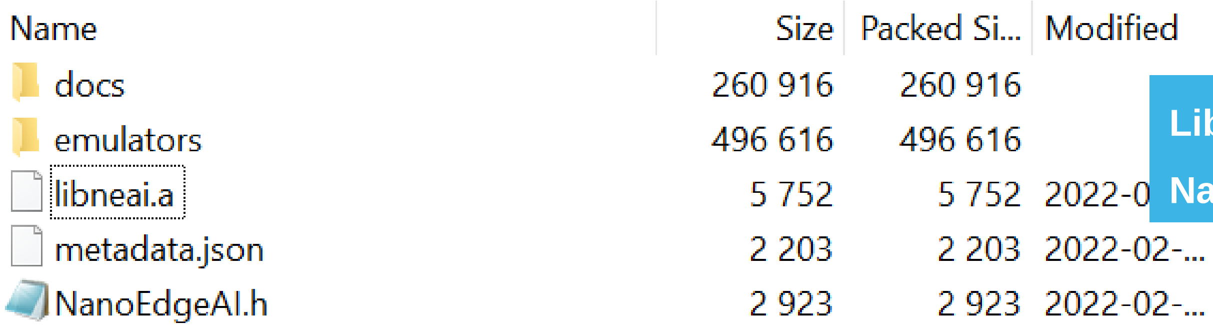
Compile the library

Check the compilation option

Integration of NanoEdge AI library

Finally, we can integrate the AI model into embedded software

Generated files after “Compile Library”

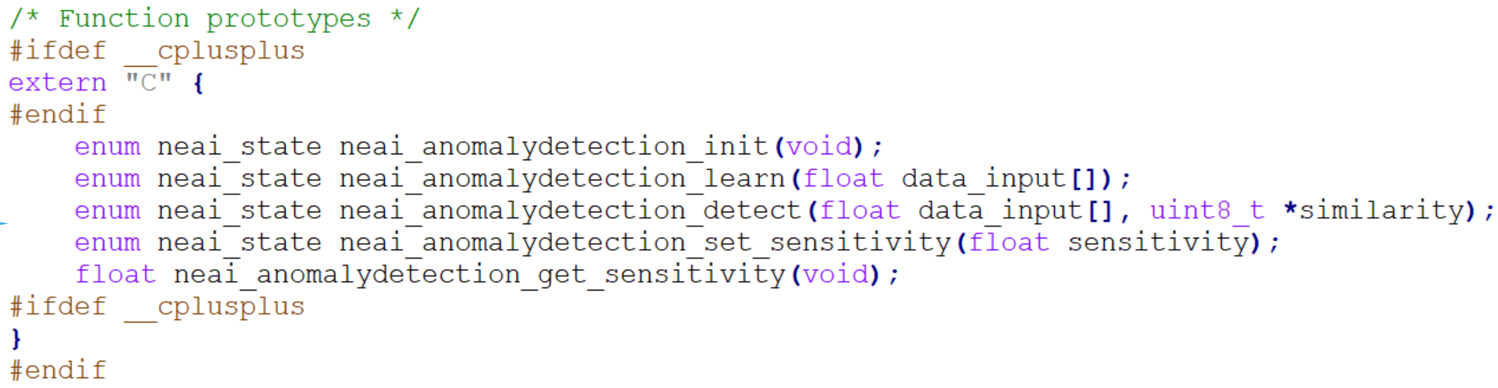


Name	Size	Packed Si...	Modified
docs	260 916	260 916	
emulators	496 616	496 616	
libneai.a	5 752	5 752	2022-0
metadata.json	2 203	2 203	2022-02-...
NanoEdgeAI.h	2 923	2 923	2022-02-...

Libneai.a: The static C library of AI model

NanoEdgeAI.h: The header file with all APIs

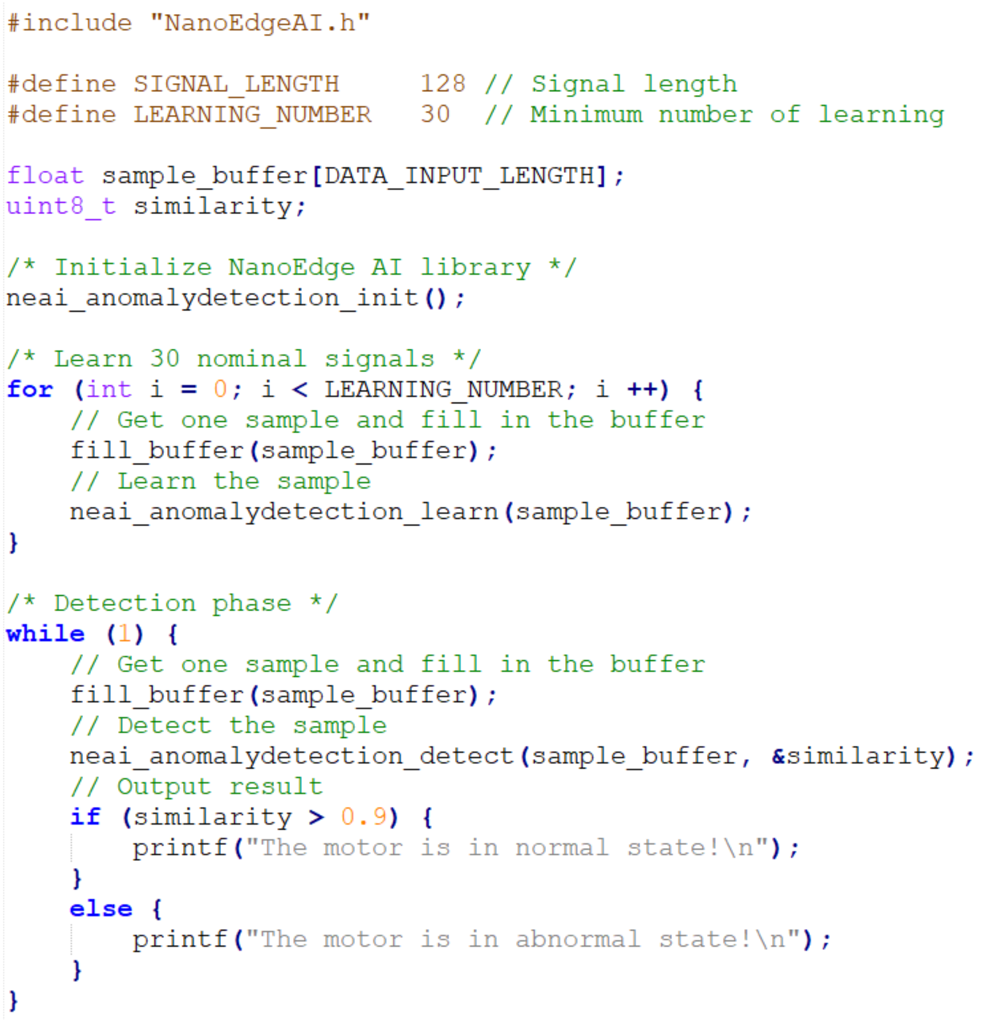
The APIs in NanoEdgeAI.h



```
/* Function prototypes */
#ifdef __cplusplus
extern "C" {
#endif
    enum neai_state neai_anomalydetection_init(void);
    enum neai_state neai_anomalydetection_learn(float data_input[]);
    enum neai_state neai_anomalydetection_detect(float data_input[], uint8_t *similarity);
    enum neai_state neai_anomalydetection_set_sensitivity(float sensitivity);
    float neai_anomalydetection_get_sensitivity(void);
#ifdef __cplusplus
}
#endif
```

Integration of NanoEdge AI library

Example codes to implement an anomaly detection library generated by NanoEdge AI Studio



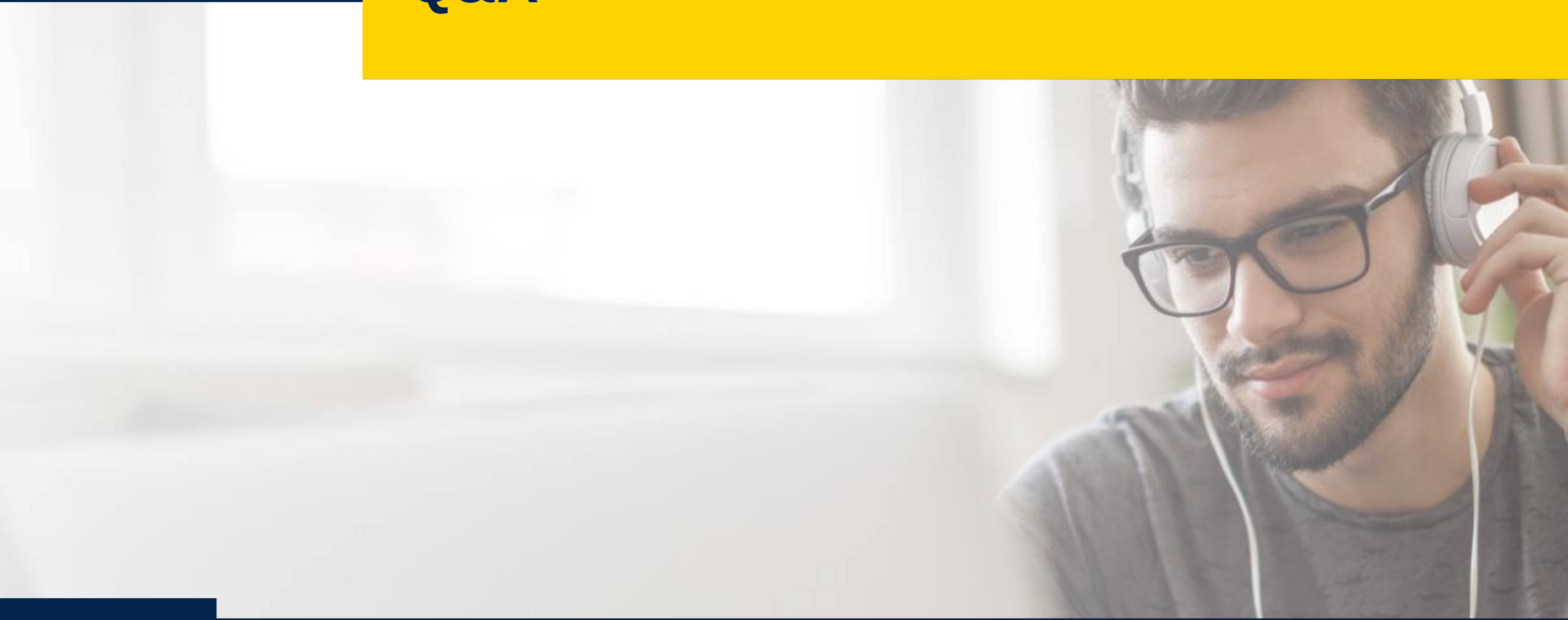
Initialization of the library

Learn minimum number of normal signals

Continuously detect anomaly

```c
#include "NanoEdgeAI.h"

#define SIGNAL_LENGTH     128 // Signal length
#define LEARNING_NUMBER   30  // Minimum number of learning

float sample_buffer[DATA_INPUT_LENGTH];
uint8_t similarity;

/* Initialize NanoEdge AI library */
neai_anomalydetection_init();

/* Learn 30 nominal signals */
for (int i = 0; i < LEARNING_NUMBER; i ++) {
    // Get one sample and fill in the buffer
    fill_buffer(sample_buffer);
    // Learn the sample
    neai_anomalydetection_learn(sample_buffer);
}

/* Detection phase */
while (1) {
    // Get one sample and fill in the buffer
    fill_buffer(sample_buffer);
    // Detect the sample
    neai_anomalydetection_detect(sample_buffer, &similarity);
    // Output result
    if (similarity > 0.9) {
        printf("The motor is in normal state!\n");
    }
    else {
        printf("The motor is in abnormal state!\n");
    }
}
```

Q&A



Our technology starts with You

life.augmented