Advanced Microprocessors

HANDS-ON PROJECT: CREATING A VOICE-CONTROLLED ROBOTIC SUBSYSTEM

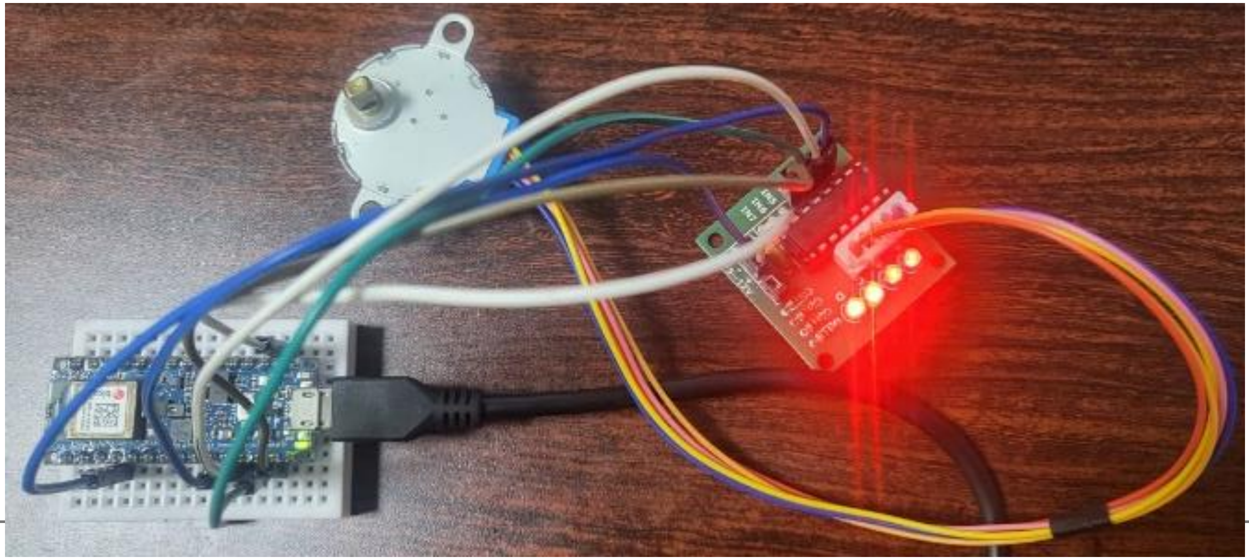
Dennis A. N. Gookyi

CONTENTS

PROJECT OVERVIEW

- In this project, we will be building a simple robotic subsystem that uses machine learning to respond to voice commands

- A microcontroller will collect inputs from a microphone, use ML to listen for the wake words like "**forward**" and “**backward**" and then drive a small DC motor in the commanded direction

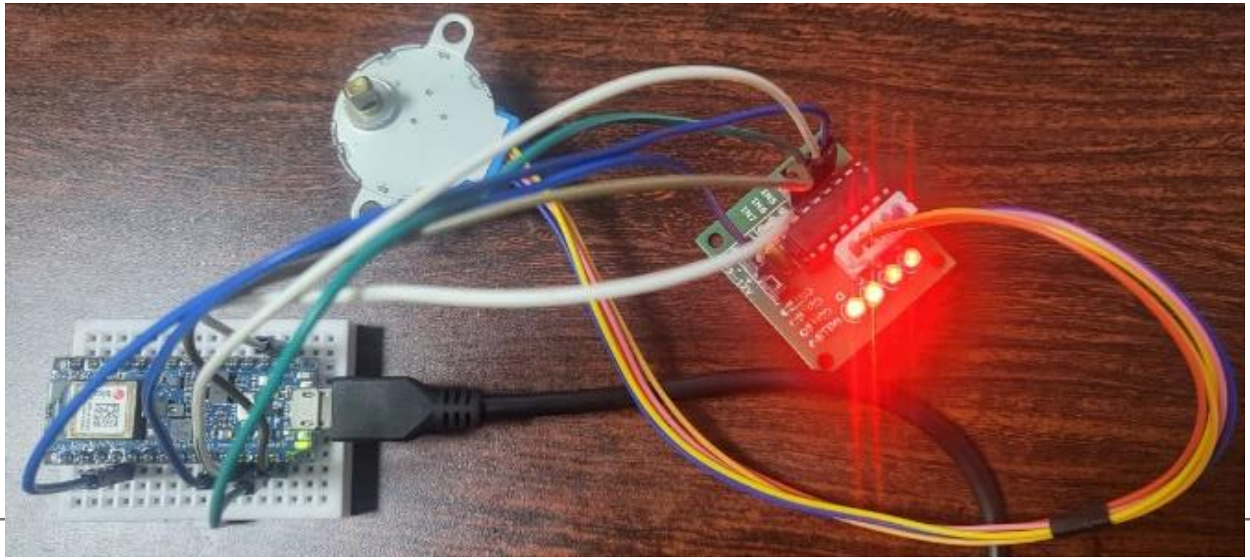


PROJECT OVERVIEW

- This project will primarily focus on demonstrating how to:
  - Use existing resources to train an ML model using the Edge Impulse platform
  - Quantize and deploy the model to an Arduino Nano 33 BLE Sense
  - Run local inference on the Arduino and have it control our motor

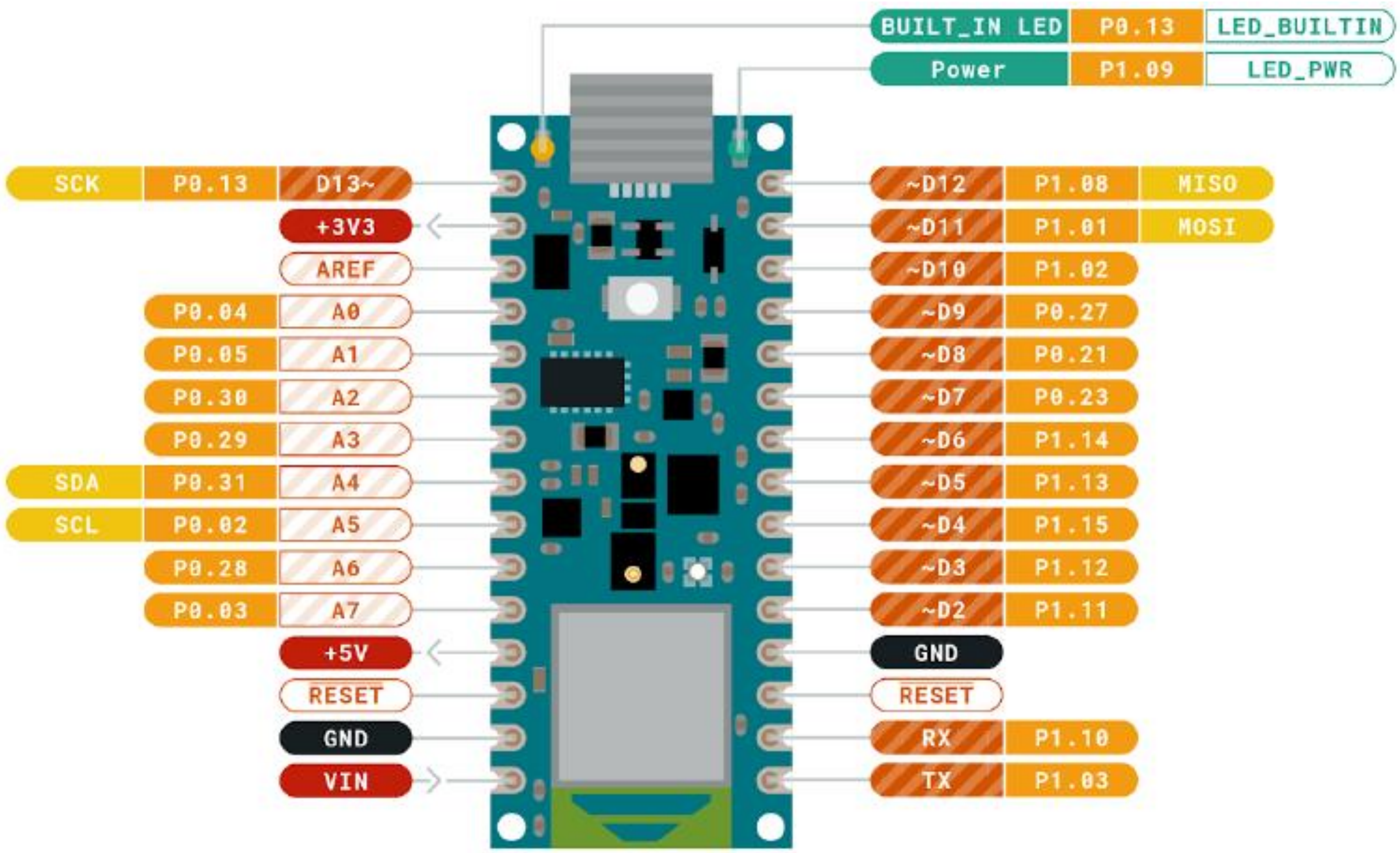
PROJECT HARDWARE

- The hardware components for the project include:
  - Arduino Nano 33 BLE Sense
  - 28BYJ-48 Stepper Motor
  - ULN2003 Driver



PROJECT HARDWARE – NANO 33 BLE SENSE

- Arduino Nano 33 BLE Sense
  - Full pinout (designation) for the Nano 33 BLE Sense development board



PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

- Stepper motors surround us without even realizing it
- They are used in so many everyday items including window blinds, 3D printers, DVD players, security cameras, etc

PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

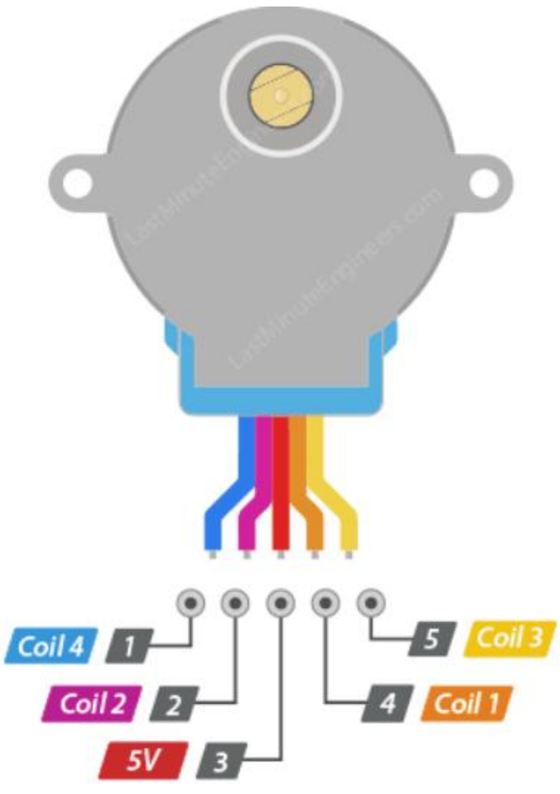
- The 28BYJ-48 is a 5-wire unipolar stepper motor that runs on 5V
- It is perfect for projects requiring precise positioning, like vent opening and closing

PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

- Despite its small size, the motor delivers a decent torque of 34.3 mN.m (millinewton-meters) at a speed of around 15 RPM
- It provides good torque even at a standstill and maintains it as long as the motor receives power
- The only drawback is that it is somewhat power-hungry and consumes energy even when it is stationary

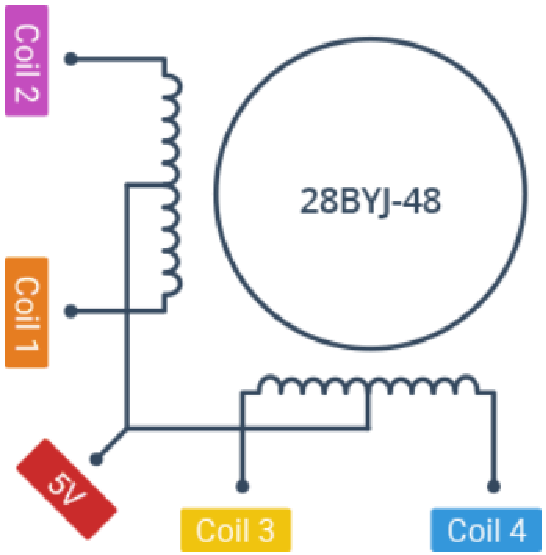
PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

- The motor has five wires
- The 28BYJ-48 has two coils, each of which has a center tap
- These two center taps are connected internally and brought out as the 5th wire (red wire)



PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

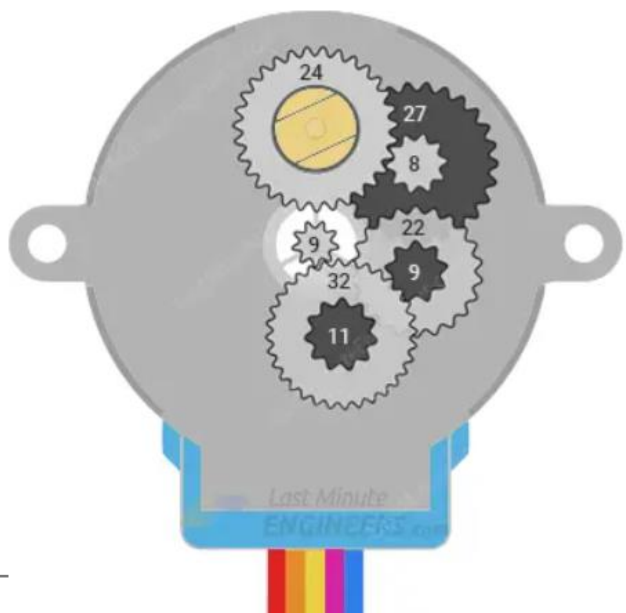
- Together, one end of the coil and the center tap form a Phase
- Thus, 28BYJ-48 has a total of four phases
- The red wire is always pulled HIGH, so when the other lead is pulled LOW, the phase is energized
- The stepper motor rotates only when the phases are energized in a logical sequence known as a step sequence



PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

❖ Gear Reduction Ratio

- When the 28BYJ-48 motor is operated in full-step mode, each step corresponds to a rotation of 11.25°
  - This means there are 32 steps per revolution (360° /11.25° = 32)
- In addition, the gearbox inside the motor has a 64:1 gear reduction
  - This results in 2048 (32*64) steps per revolution



Gear Ratios:

- 32 / 9
- 22 / 11
- 27 / 9
- 24 / 8

Multiplying the gear ratios:

$$\frac{32}{9} \times \frac{22}{11} \times \frac{27}{9} \times \frac{24}{8} = 64$$

This gives us a **64:1** gear ratio

PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

- Power Consumption

  - The 28BYJ-48 typically draws about 240 mA

  - Because the motor consumes a significant amount of power, it is preferable to power it directly from an external 5V power supply rather than from the Arduino

  - It is worth noting that the motor consumes power even when it is at rest to maintain its position

PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

❖ Technical Specifications

Operating Voltage	5VDC
Operating Current	240mA (typical)
Number of phases	4
Gear Reduction Ratio	64:1
Step Angle	5.625°/64
Frequency	100Hz
In-traction Torque	>34.3mN.m(120Hz)
Self-positioning Torque	>34.3mN.m
Friction torque	600-1200 gf.cm
Pull in torque	300 gf.cm

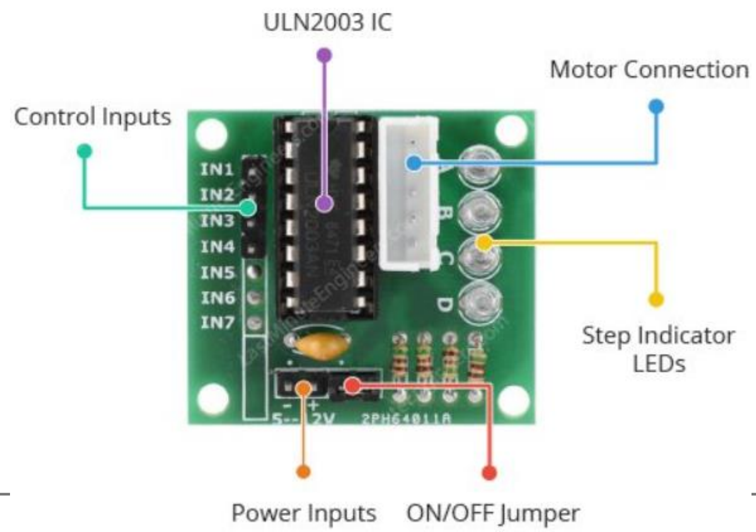
PROJECT HARDWARE – ULN2003 DRIVER BOARD

- Because the 28BYJ-48 stepper motor consumes a significant amount of power, it cannot be controlled directly by a microcontroller such as Arduino

- To control the motor, a driver IC such as the ULN2003 is required
  - Therefore, this motor typically comes with a ULN2003-based driver board

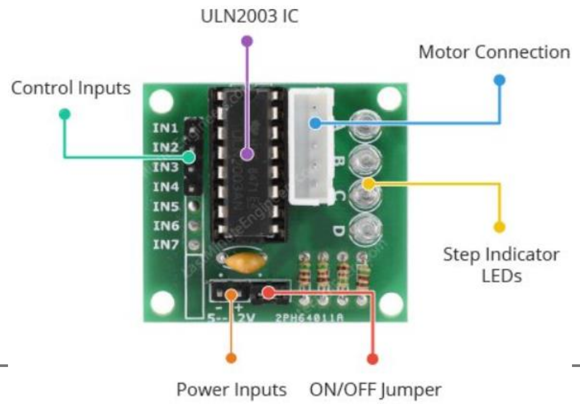
PROJECT HARDWARE – ULN2003 DRIVER BOARD

- The ULN2003, known for its high current and high voltage capability, provides a higher current gain than a single transistor and allows a microcontroller's low voltage low current output to drive a high current stepper motor
- The ULN2003 consists of an array of seven Darlington transistor pairs, each of which can drive a load of up to 500mA and 50V
  - This board utilizes four of the seven pairs



PROJECT HARDWARE – ULN2003 DRIVER BOARD

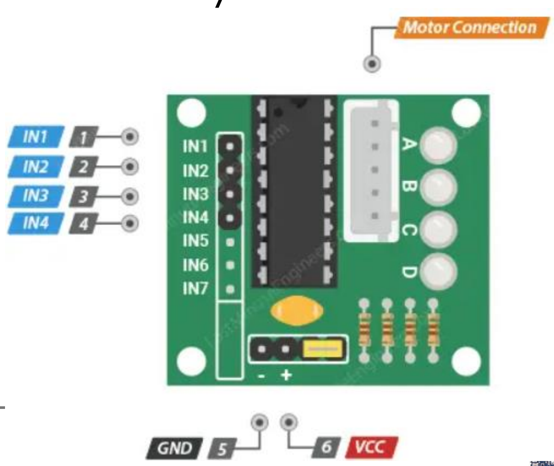
- The board has four control inputs and a power supply connection
- Additionally, there is a Molex connector that is compatible with the connector on the motor, allowing you to plug the motor directly into it
- The board includes four LEDs that indicate activity on the four control input lines
- They provide a good visual indication while stepping
- There is an ON/OFF jumper on the board for disabling the stepper motor if needed



PROJECT HARDWARE – ULN2003 DRIVER BOARD

❖ ULN2003 Stepper Driver Board Pinout

- IN1 – IN4 are motor control input pins
  - Connect them to the Arduino's digital output pin
- GND is the ground pin
- VCC pin powers the motor
  - Because the motor consumes a significant amount of power, it is preferable to use an external 5V power supply rather than the Arduino
- Motor Connector This is where the motor plugs in
  - The connector is keyed, so it will only go in one way



PROJECT HARDWARE

- Wiring 28BYJ-48 Stepper Motor and ULN2003 Driver to an Arduino

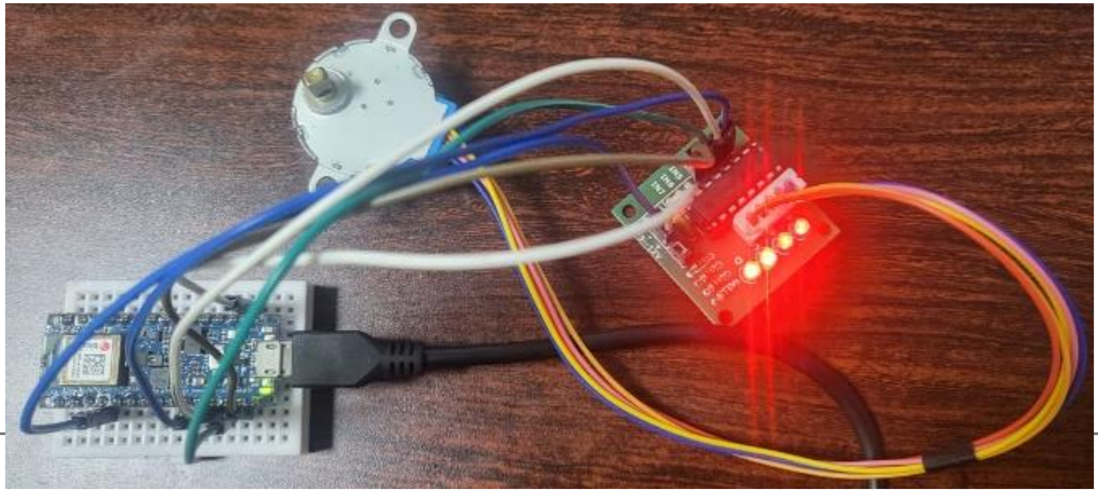
  - The connections are straightforward

    - Begin by connecting an external 5V power supply to the ULN2003 driver
    - Connect the driver board's IN1, IN2, IN3, and IN4 to Arduino digital pins 8, 9, 10, and 11, respectively
    - Then connect the stepper motor to the ULN2003 driver
    - Finally, make sure your circuit and Arduino share a common ground

PROJECT HARDWARE

- The following table lists the pin connections:

ULN2003 Driver	Arduino
IN1	8
IN2	9
IN3	10
IN4	11
GND	GND



ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

- We will use the Arduino Stepper Library, which comes with the Arduino IDE
  - https://www.arduino.cc/reference/en/libraries/stepper/

- The Arduino stepper library handles the stepping sequence and allows you to control a wide range of unipolar and bipolar stepper motors

ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

❖ Here is a simple sketch that turns the motor slowly in one direction, then rapidly in the opposite direction

```
//Includes the Arduino Stepper Library
#include <Stepper.h>

// Defines the number of steps per rotation
const int stepsPerRevolution = 2048;

// Creates an instance of stepper class
// Pins entered in sequence IN1-IN3-IN2-IN4 for proper step sequence
Stepper myStepper = Stepper(stepsPerRevolution, 8, 10, 9, 11);

void setup() {
    // Nothing to do (Stepper Library sets pins as outputs)
}

void loop() {
  // Rotate CW slowly at 5 RPM
  myStepper.setSpeed(5);
  myStepper.step(stepsPerRevolution);
  delay(1000);

  // Rotate CCW quickly at 10 RPM
  myStepper.setSpeed(10);
  myStepper.step(-stepsPerRevolution);
  delay(1000);
}
```

ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

- Code Explanation:
  - The sketch starts by including the built-in stepper library

```
//Includes the Arduino Stepper Library
#include <Stepper.h>
```

ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

- Code Explanation:
  - Next, a constant **stepsPerRevolution** is defined, which contains the number of ‘steps’ the motor takes to complete one revolution
  - In our case, it is 2048

```
// Defines the number of steps per rotation
const int stepsPerRevolution = 2048;
```

ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

- Code Explanation:
  - The step sequence of the 28BYJ-48 unipolar stepper motor is IN1-IN3-IN2-IN4
  - We will use this information to control the motor by creating an instance of the stepper library **myStepper** with the pin sequence 8, 10, 9, 11
  - Make sure you do it right; otherwise, the motor will not work properly

```
// Creates an instance of stepper class
// Pins entered in sequence IN1-IN3-IN2-IN4 for proper step sequence
Stepper myStepper = Stepper(stepsPerRevolution, 8, 10, 9, 11);
```

ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

- Code Explanation:
  - Since the stepper library internally configures the four control pins as outputs, there is nothing to configure in the setup function, so it is left empty

```

void setup() {
    // Nothing to do (Stepper Library sets pins as outputs)
}
```

ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

- Code Explanation:
  - In the loop function, we use the **setSpeed()** function to specify the speed at which the stepper motor should move and the **step()** function to specify how many steps to take
  - Passing a negative number to the **step()** function causes the motor to spin in the opposite direction
  - The first code snippet causes the motor to spin very slowly clockwise, while the second causes it to spin very quickly counter-clockwise

```
void loop() {
  // Rotate CW slowly at 5 RPM
  myStepper.setSpeed(5);
  myStepper.step(stepsPerRevolution);
  delay(1000);

  // Rotate CCW quickly at 10 RPM
  myStepper.setSpeed(10);
  myStepper.step(-stepsPerRevolution);
  delay(1000);
}
```

SPEECH COMMANDS DATASET

- Speech Commands Data Set v0.02
  - This is a set of one-second **.wav** audio files, each containing a single spoken English word
  - These words are from a small set of commands, and are spoken by a variety of different speakers
  - The audio files are organized into folders based on the word they contain, and this data set is designed to help train simple machine learning models
  - This dataset is covered in more detail at
    - https://arxiv.org/abs/1804.03209

SPEECH COMMANDS DATASET

- Speech Commands Data Set v0.02
  - It's licensed under the Creative Commons BY 4.0 license
    - https://creativecommons.org/licenses/by/4.0/
  - See the LICENSE file in this folder for full details
  - Its original location was at
    - http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz

SPEECH COMMANDS DATASET

- Speech Commands Data Set v0.02
  - History
    - Version 0.01 of the data set was released on August 3rd, 2017 and contained 64,727 audio files
    - This is version 0.02 of the data set containing 105,829 audio files, released on April 11th, 2018

SPEECH COMMANDS DATASET

- Speech Commands Data Set v0.02
  - Collection
    - The audio files were collected using crowdsourcing
      - aiyprojects.withgoogle.com/open_speech_recording
      - https://github.com/petewarden/extract_loudest_section
    - The goal was to gather examples of people speaking single-word commands, rather than conversational sentences, so they were prompted for individual words over the course of a five-minute session
    - Twenty core command words were recorded, with most speakers saying each five times
    - The core words are “**Yes**”, “**No**”, “**Up**”, “**Down**”, “**Left**”, "**Right**", "**On**", "**Off**", "**Stop**", "**Go**", "**Zero**", "**One**", "**Two**", "**Three**", "**Four**", "**Five**", "**Six**", "**Seven**", "**Eight**", and "**Nine**“
    - To help distinguish unrecognized words, there are also ten auxiliary words, which most speakers only said once
    - These include "**Bed**", "**Bird**", "**Cat**", "**Dog**", "**Happy**", "**House**", "**Marvin**", "**Sheila**", "**Tree**", and "**Wow**"

SPEECH COMMANDS DATASET

- Speech Commands Data Set v0.02
  - Organization
  - The files are organized into folders, with each directory name labelling the word that is spoken in all the contained audio files
  - No details were kept of any of the participants age, gender, or location, and random ids were assigned to each individual
  - These ids are stable though, and encoded in each file name as the first part before the underscore
  - If a participant contributed multiple utterances of the same word, these are distinguished by the number at the end of the file name
    - For example, the file path ‘**happy/3cfc6b3a_nohash_2.wav**’ indicates that the word spoken was "**happy**", the speaker's id was "**3cfc6b3a**", and this is the third utterance of that word by this speaker in the data set
    - The '**nohash**' section is to ensure that all the utterances by a single speaker are sorted into the same training partition, to keep very similar repetitions from giving unrealistically optimistic evaluation scores

SPEECH COMMANDS DATASET

- Speech Commands Data Set v0.02
  - Organization
    - The files are organized into folders, with each directory name labelling the word that is spoken in all the contained audio files
    - No details were kept of any of the participants age, gender, or location, and random ids were assigned to each individual
    - These ids are stable though, and encoded in each file name as the first part before the underscore
    - If a participant contributed multiple utterances of the same word, these are distinguished by the number at the end of the file name
      - For example, the file path '**happy/3cfc6b3a_nohash_2.wav**' indicates that the word spoken was "**happy**", the speaker's id was "**3cfc6b3a**", and this is the third utterance of that word by this speaker in the data set
      - The '**nohash**' section is to ensure that all the utterances by a single speaker are sorted into the same training partition, to keep very similar repetitions from giving unrealistically optimistic evaluation scores

SPEECH COMMANDS DATASET

- Speech Commands Data Set v0.02
  - Partitioning
    - The audio clips haven't been separated into training, test, and validation sets explicitly, but by convention, a hashing function is used to stably assign each file to a set

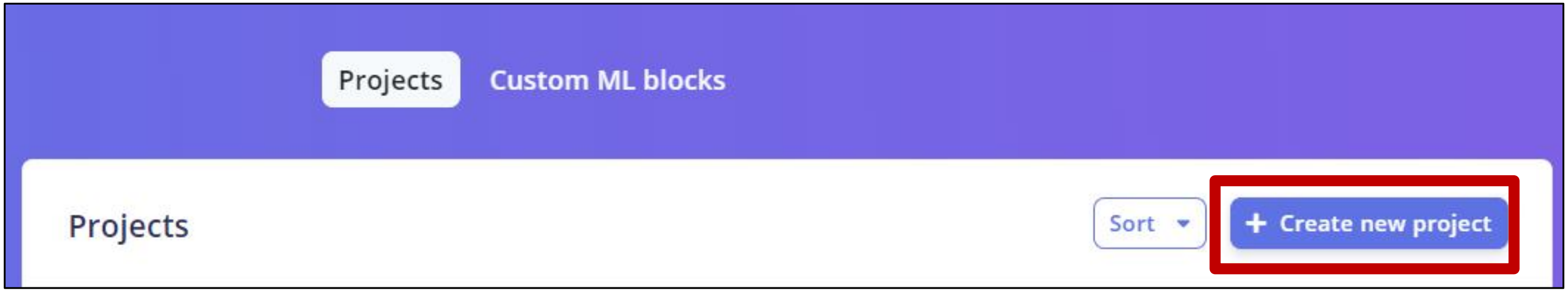
SPEECH COMMANDS DATASET

- Speech Commands Data Set v0.02
  - Processing
    - The original audio files were collected in uncontrolled locations by people around the world
    - The recording was done in a closed room for privacy reasons
    - This was by design since they wanted examples of the sort of speech data that are likely to be encountered in consumer and robotics applications, where we don't have much control over the recording equipment or environment
    - The data was captured in a variety of formats, for example, Ogg Vorbis encoding for the web app, and then converted to a 16-bit little-endian PCM-encoded WAVE file at a 16000 sample rate
    - The audio was then trimmed to a one-second length to align most utterances
      - https://github.com/petewarden/extract_loudest_section
    - The audio files were then screened for silence or incorrect words, and arranged into folders by label

SPEECH COMMANDS DATASET

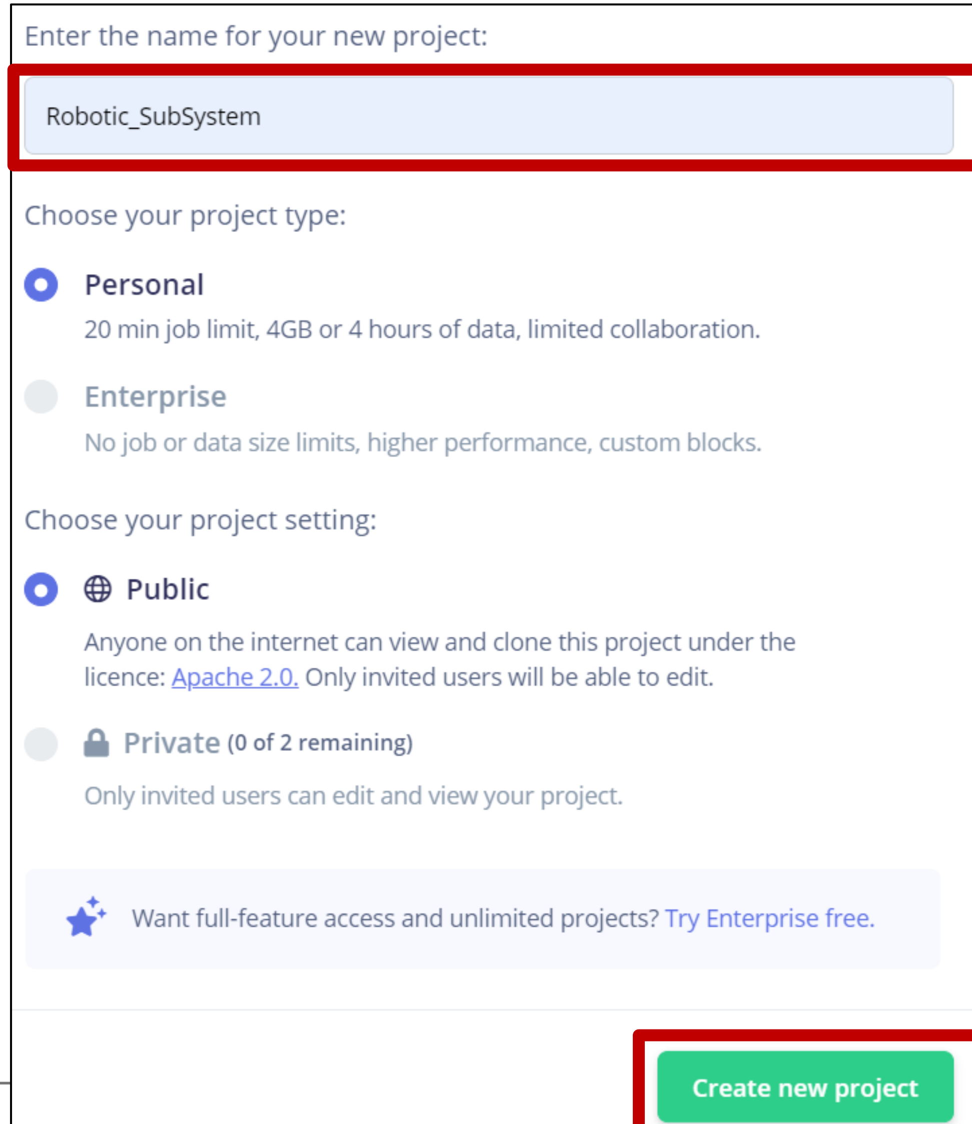
- Speech Commands Data Set v0.02
  - Background Noise
    - To help train networks to cope with noisy environments, it can be helpful to mix in realistic background audio
    - The '_background_noise_' folder contains a set of longer audio clips that are either recordings or mathematical simulations of noise
    - For more details, see the '_background_noise_/README.md'

- Create project



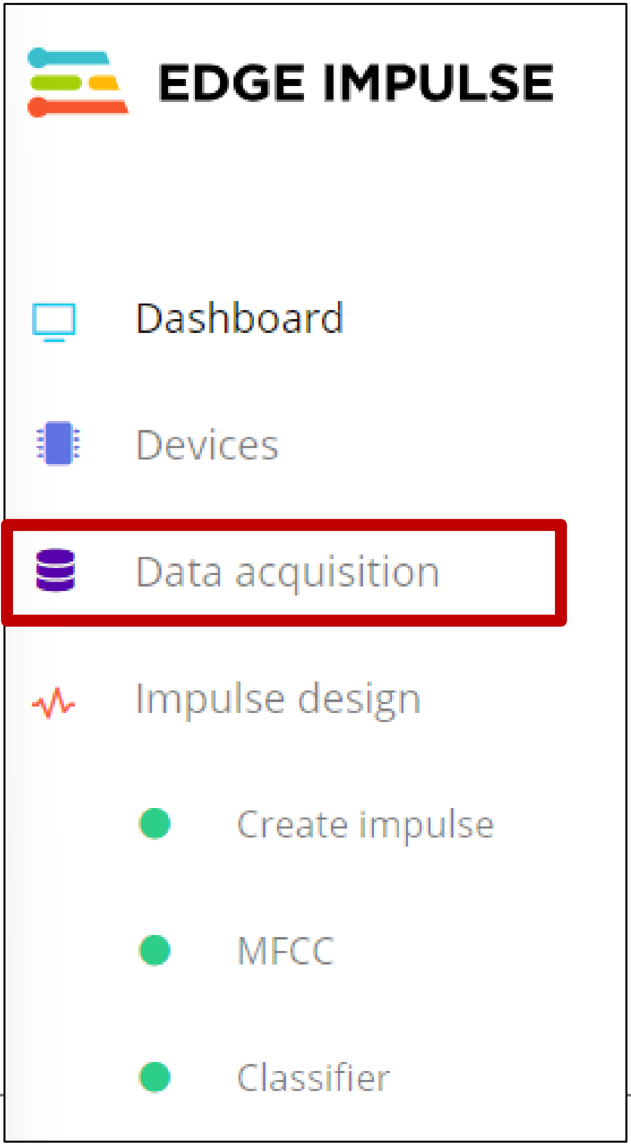
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Create project



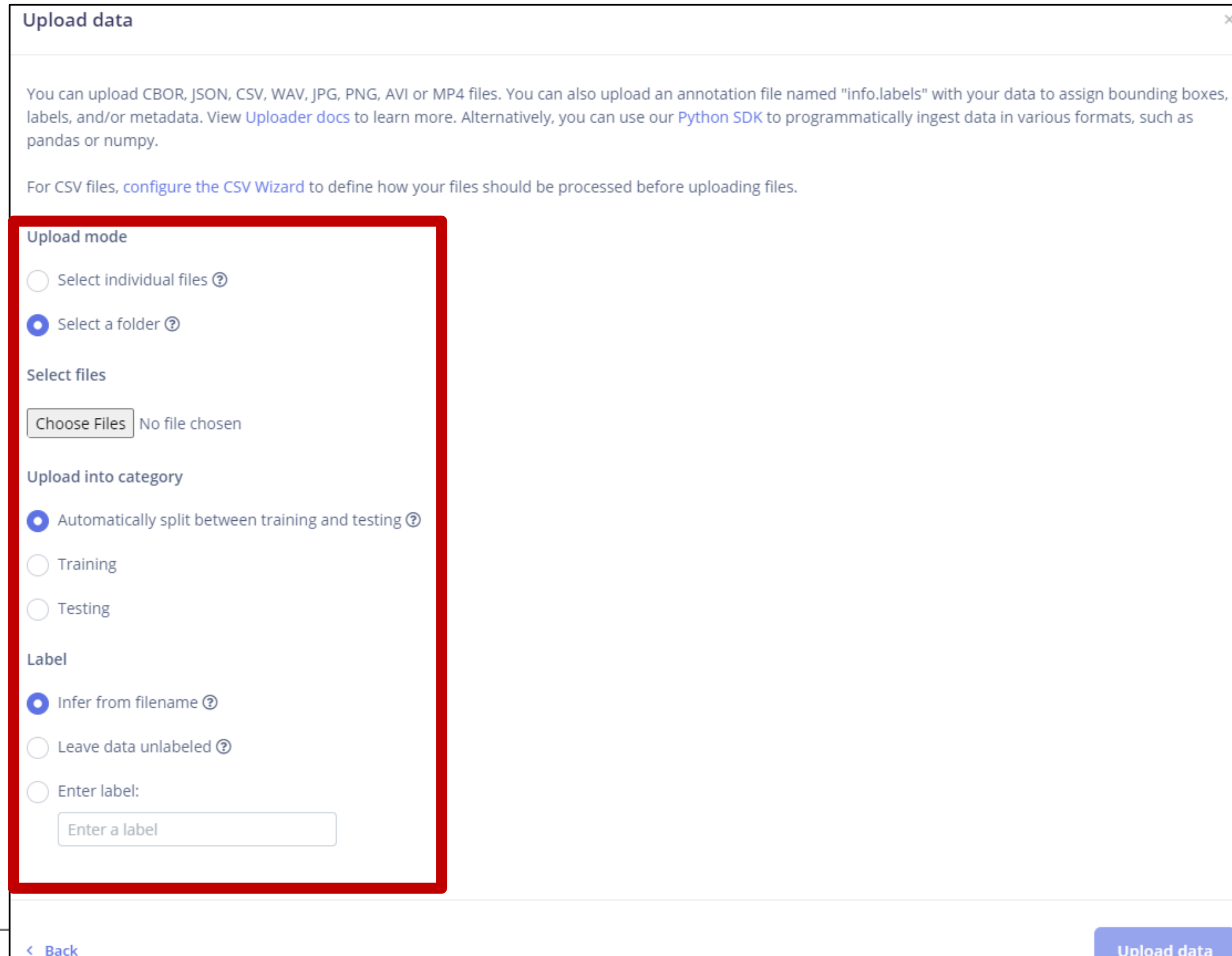
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Data acquisition
  - Forward
  - Backward
  - Noise/Silence

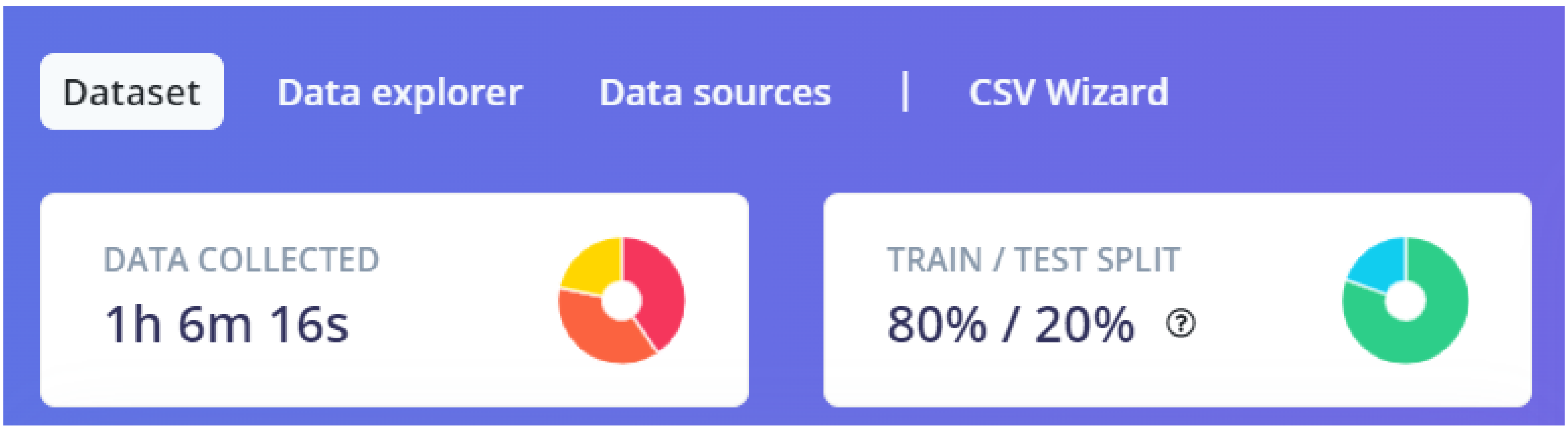


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Data acquisition
  - Forward
  - Backward
  - Noise/Silence



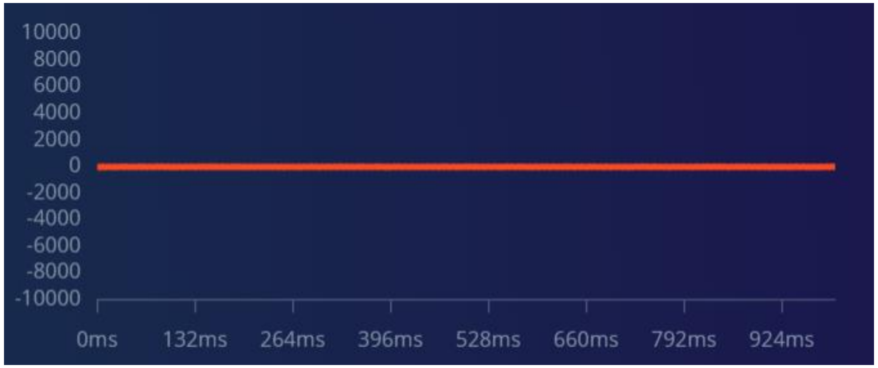
- Data acquisition
  - Forward
  - Backward
  - Noise/Silence



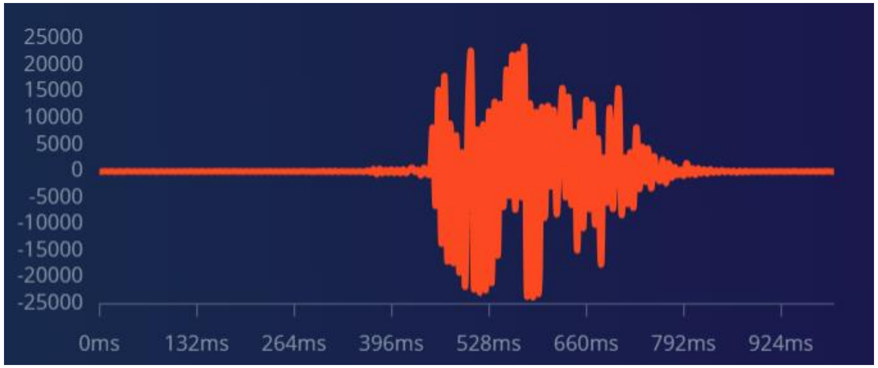
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Data acquisition
  - Forward
  - Backward
  - Noise/Silence

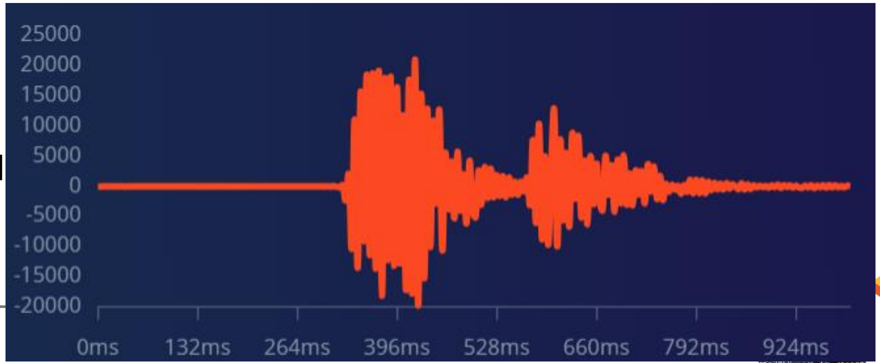
Silence raw data



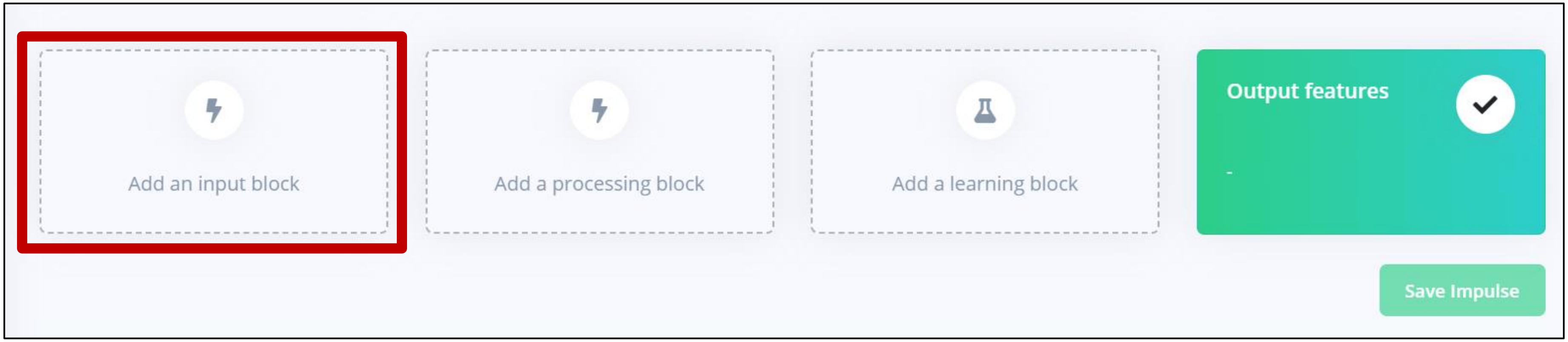
Forward raw data



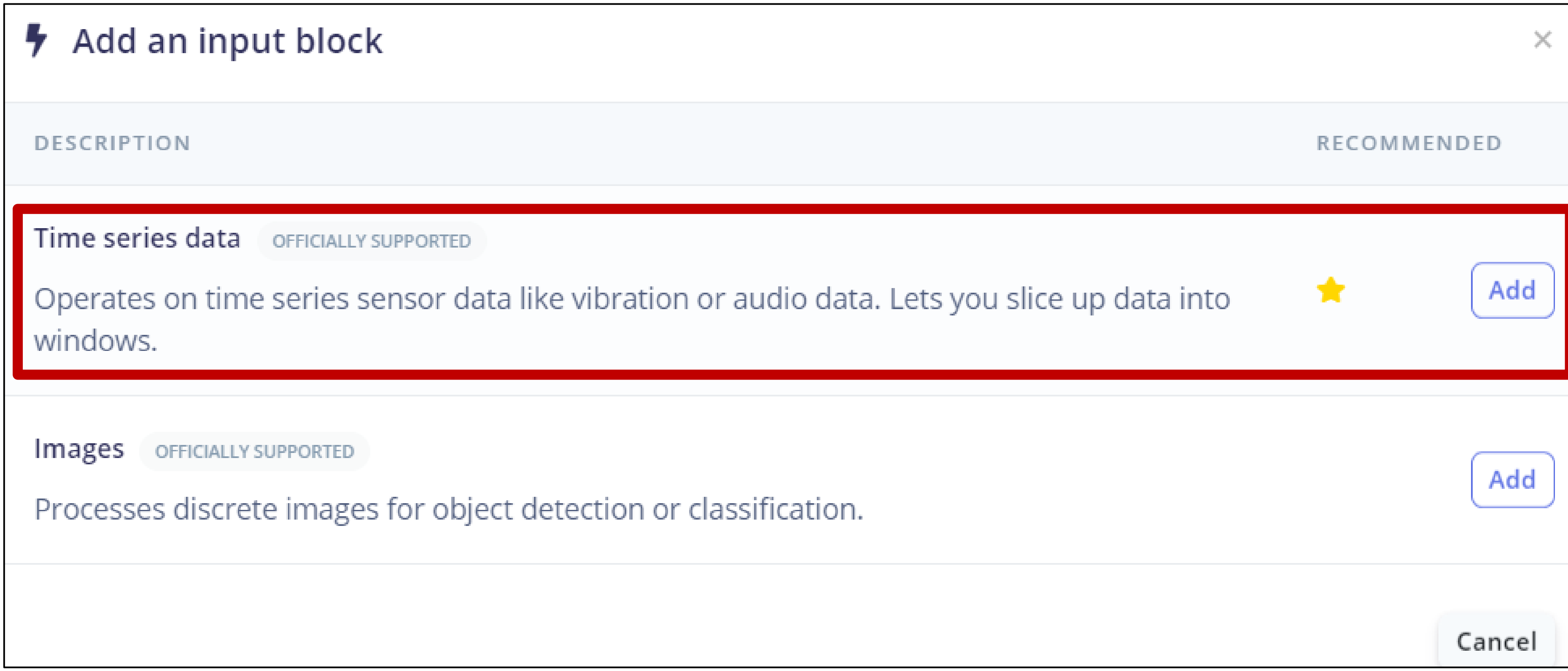
Backward raw data



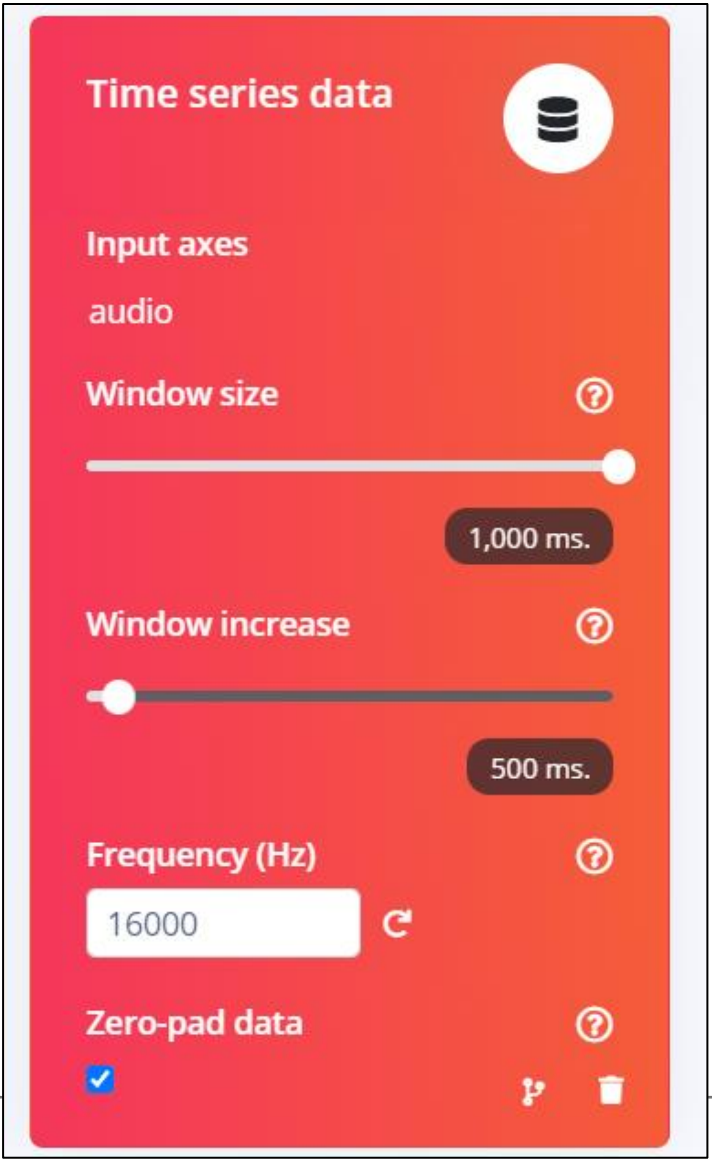
- Impulse design
  - Create impulse



- Impulse design
  - Create impulse

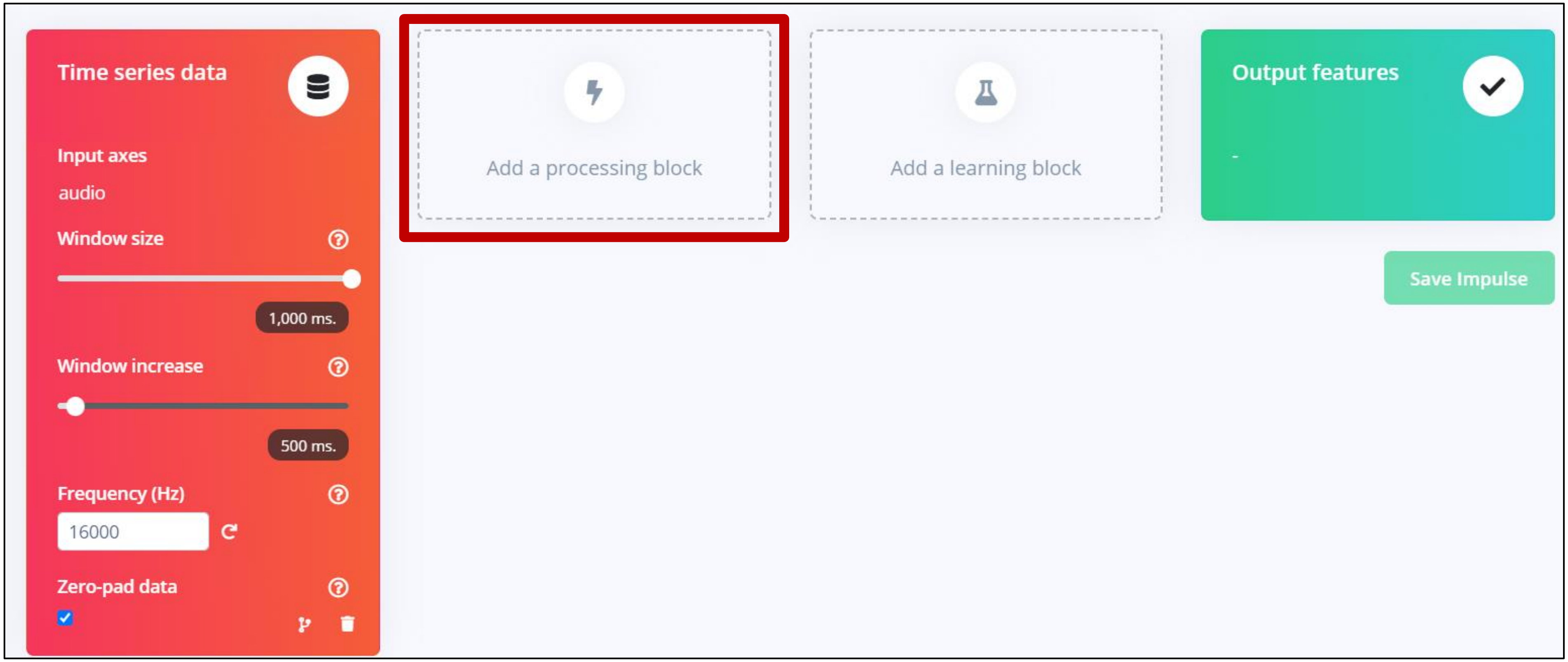


- Impulse design
  - Create impulse



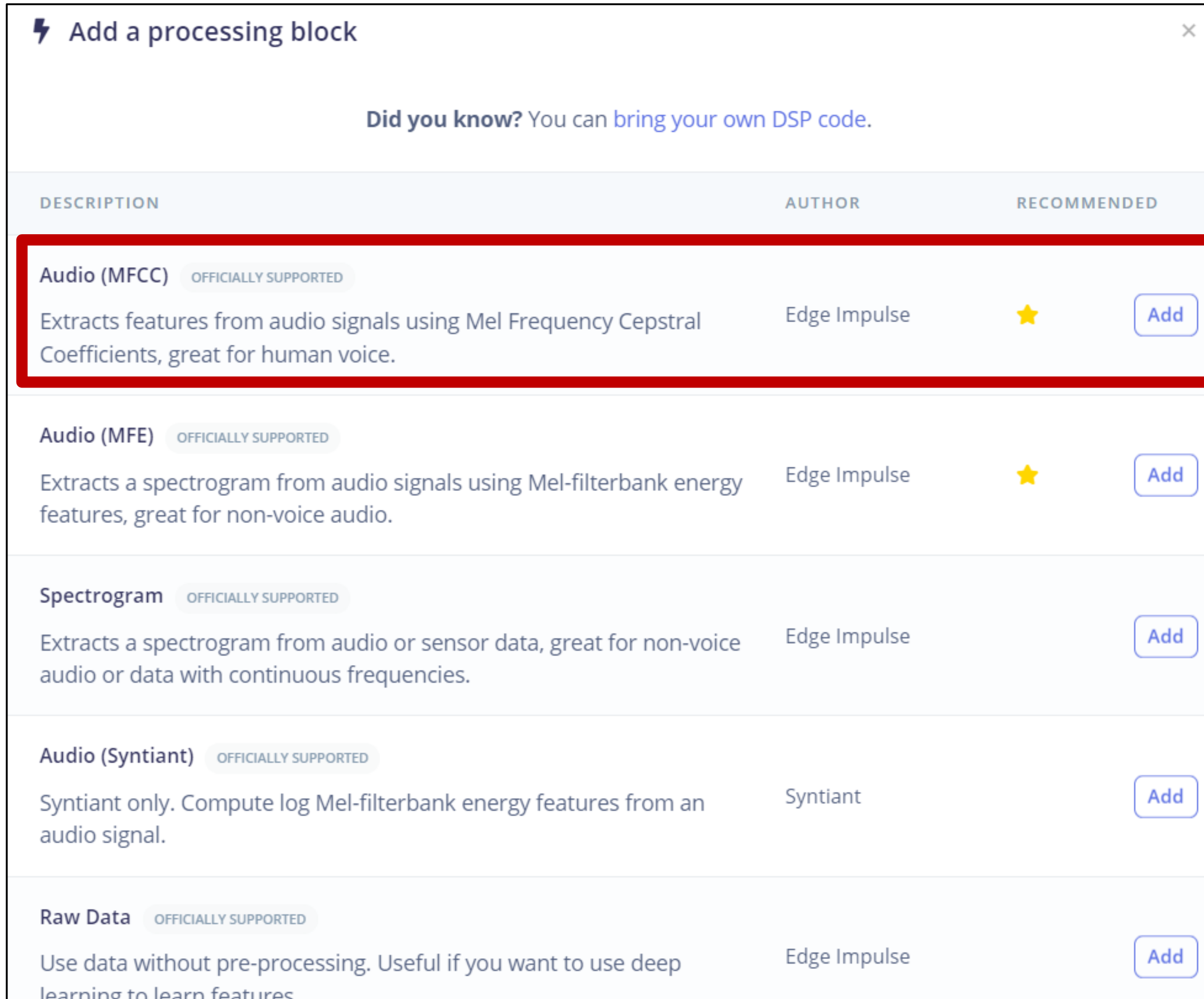
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Impulse design
  - Create impulse



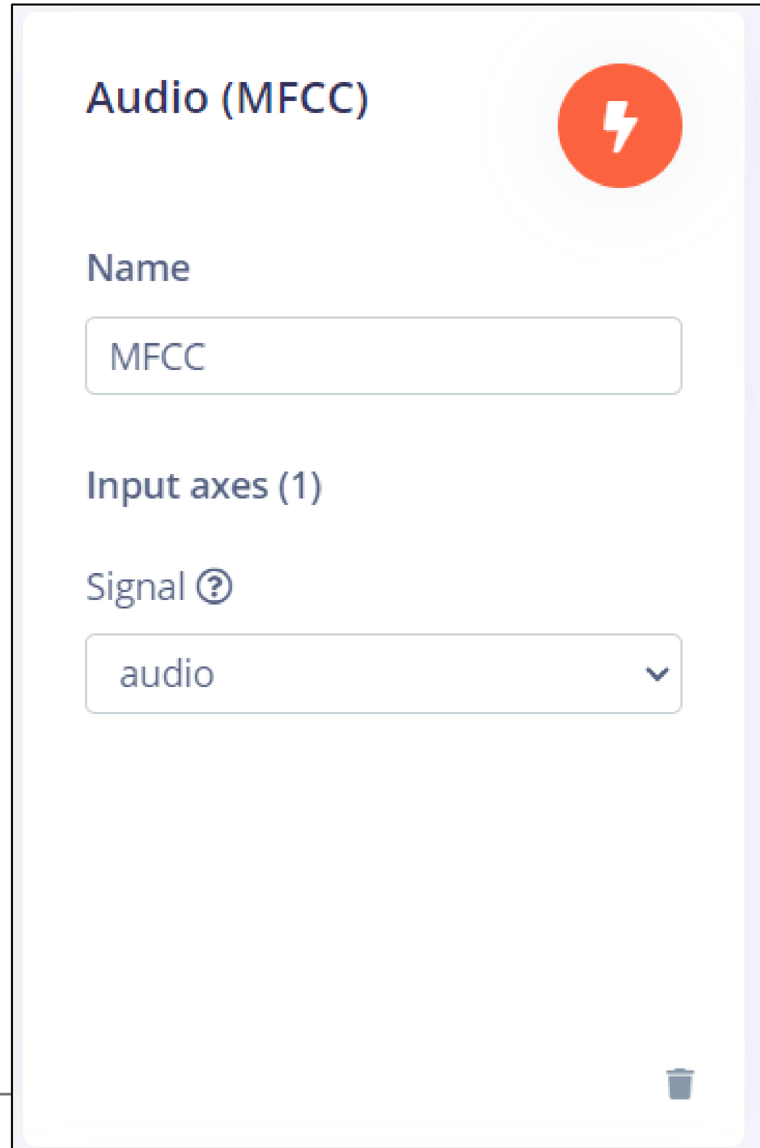
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Impulse design
  - Create impulse



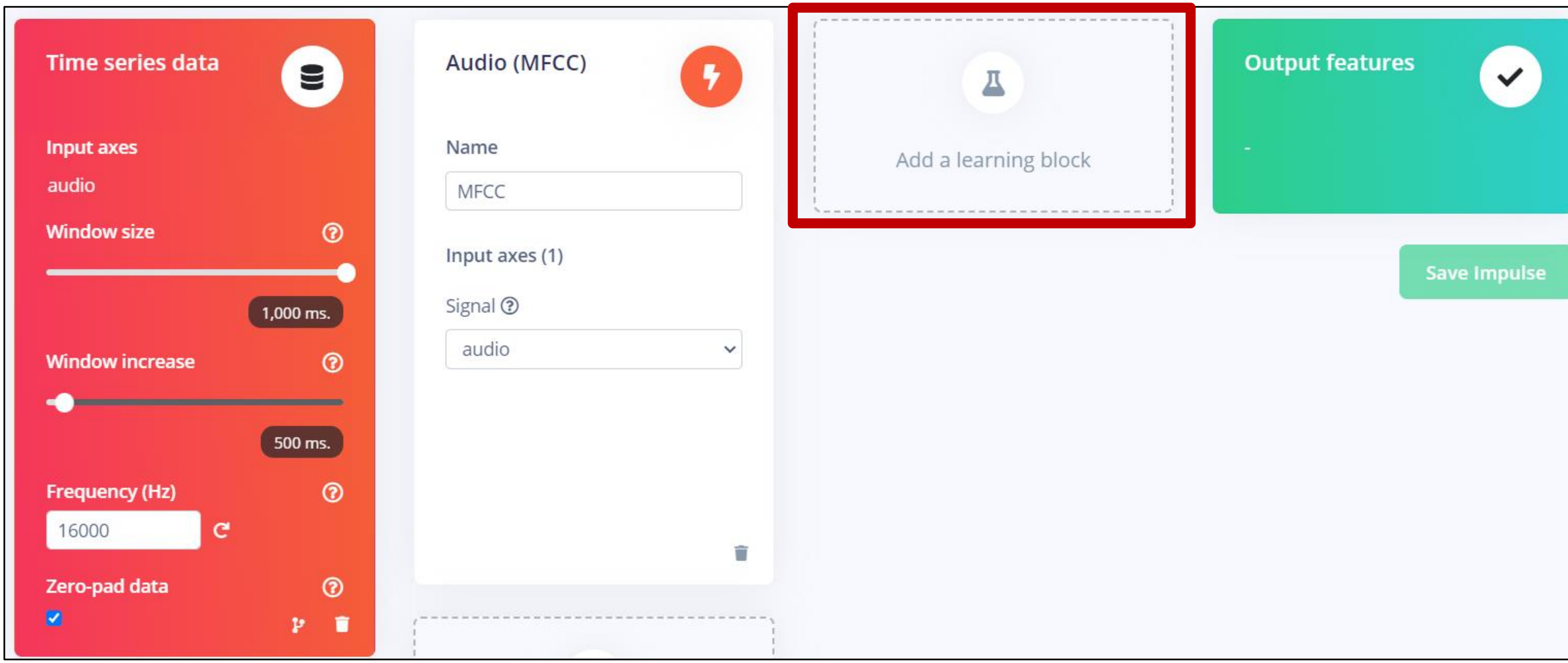
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Impulse design
  - Create impulse



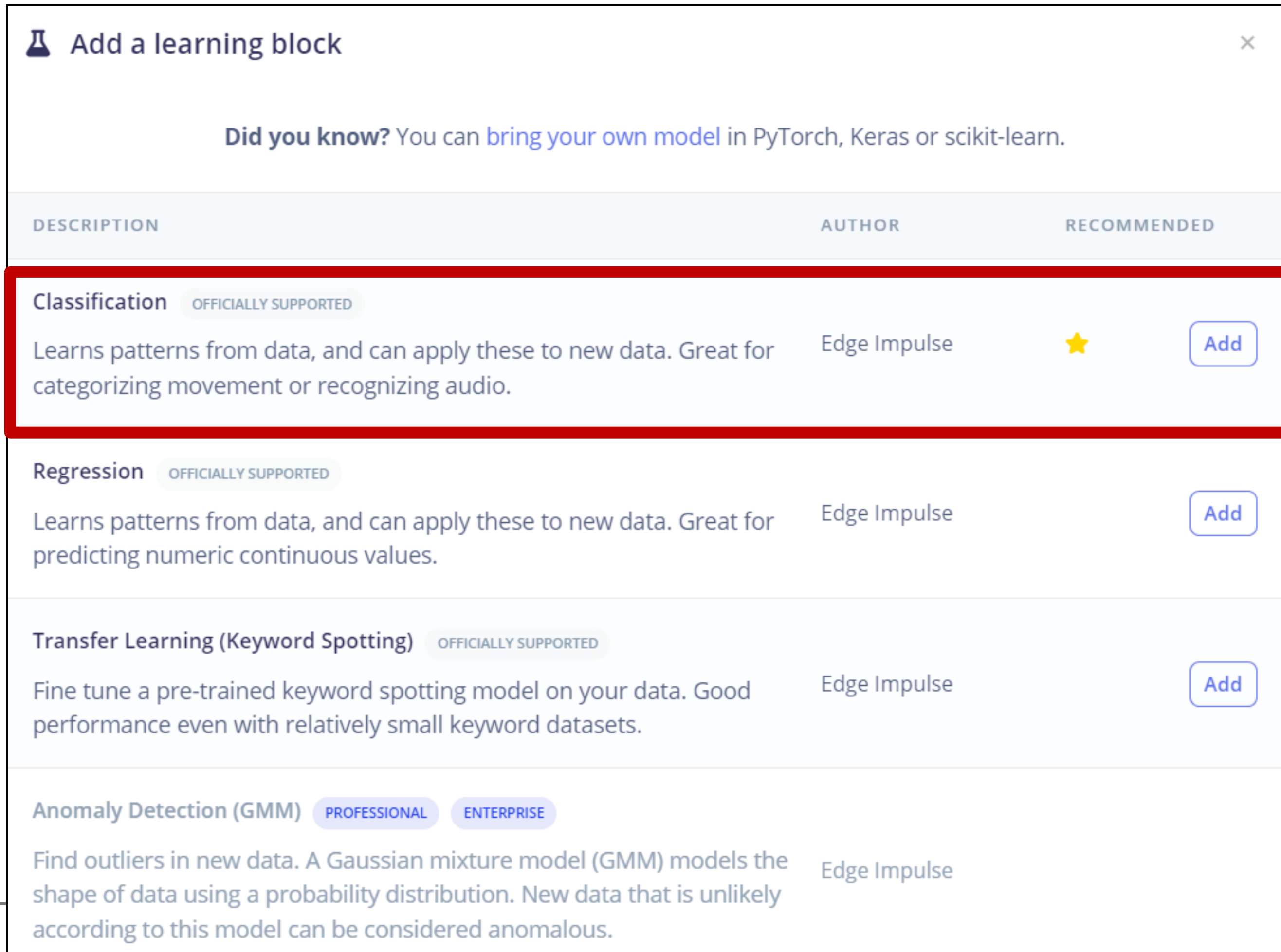
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Impulse design
  - Create impulse



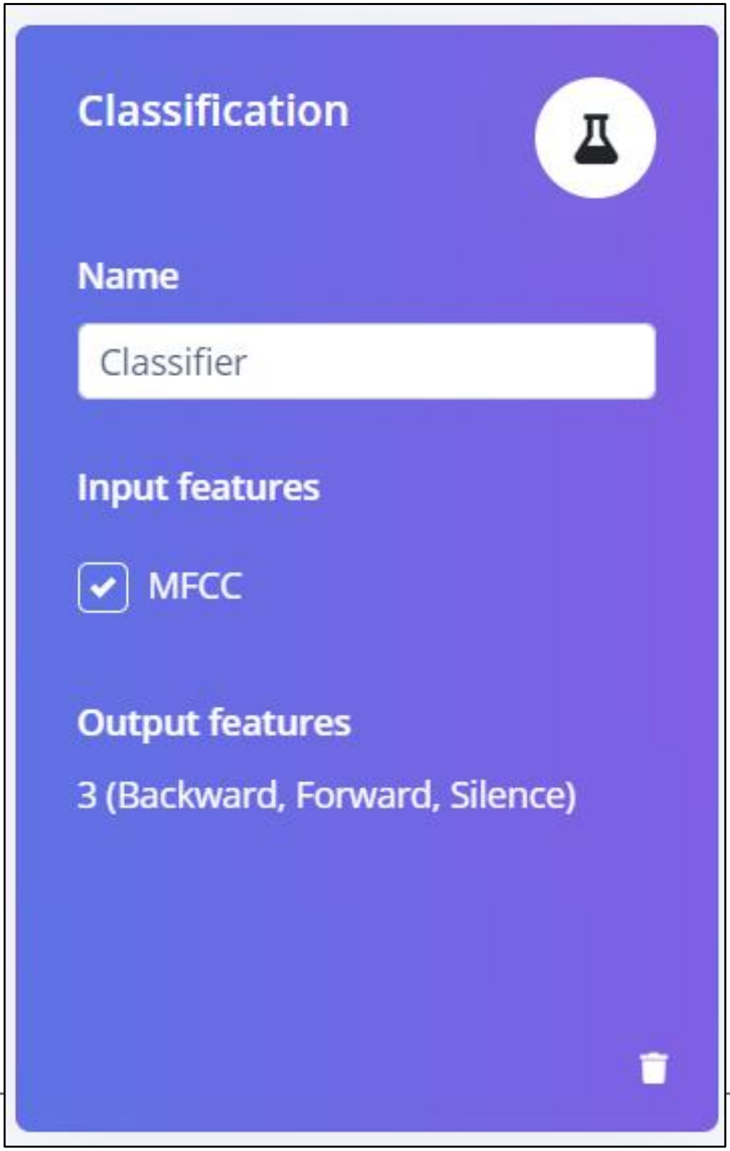
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Impulse design
  - Create impulse

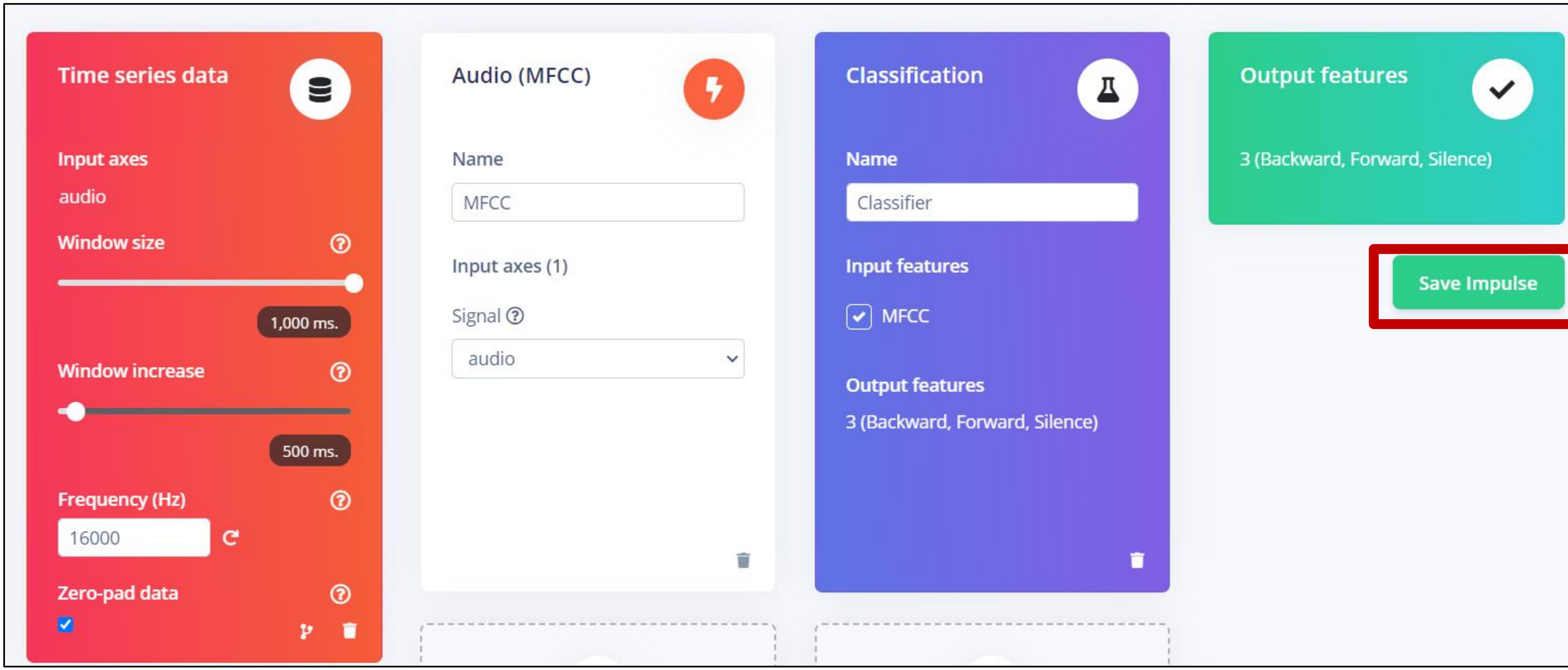


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Impulse design
  - Create impulse

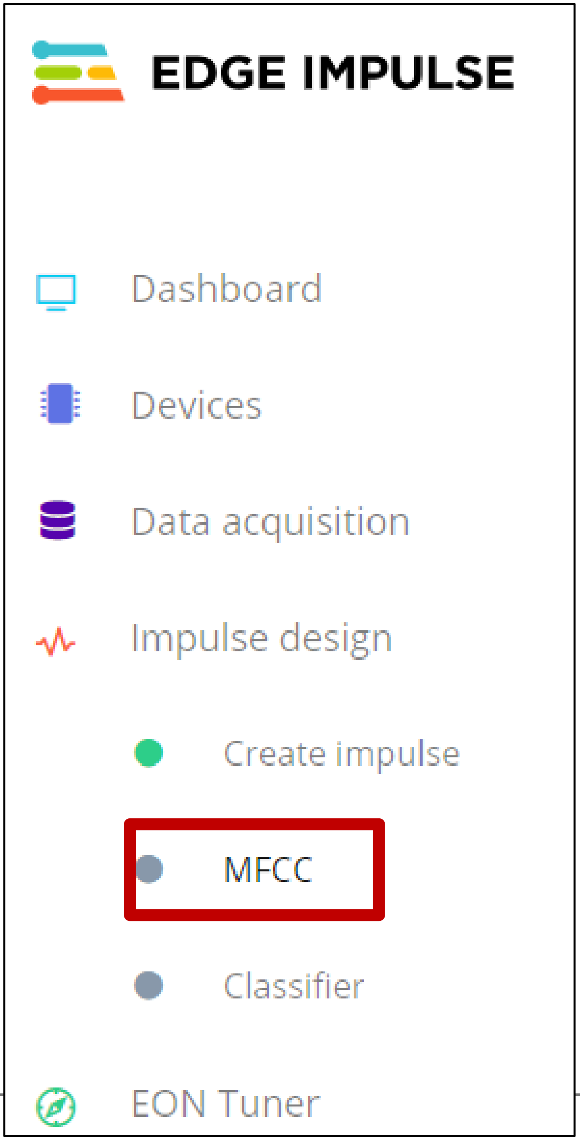


- Impulse design
  - Create impulse



MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

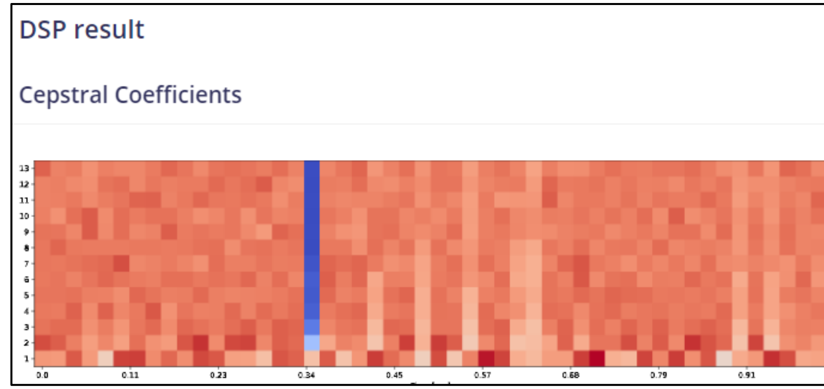
- Impulse design
  - MFCC



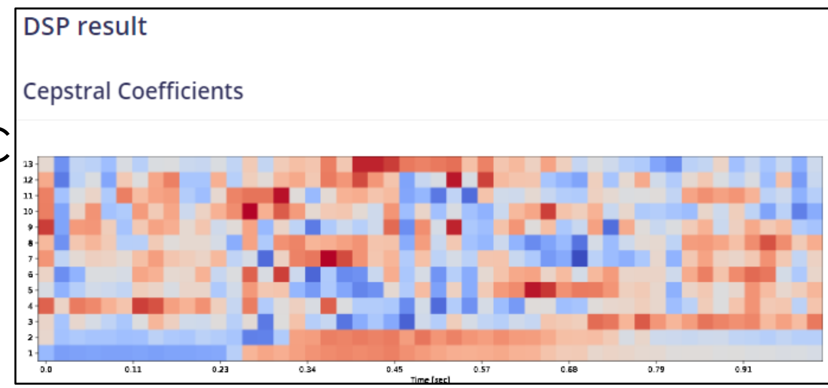
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

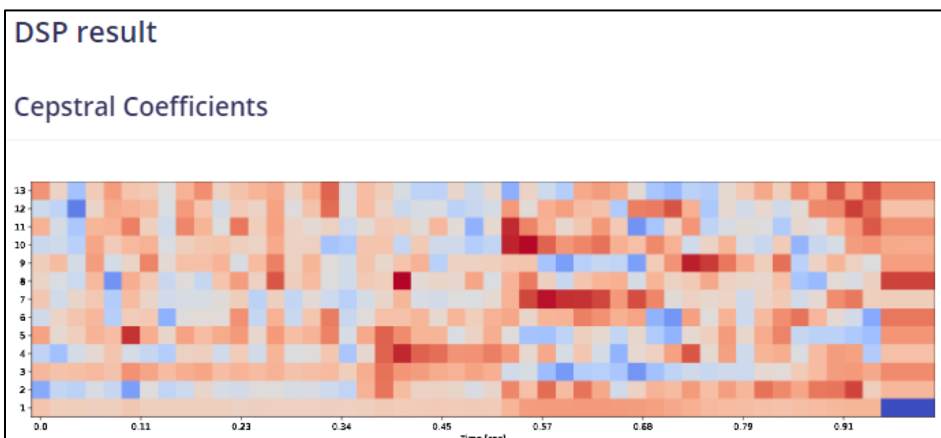
- Impulse design
  - MFCC

Silence MFCC



Forward MFCC





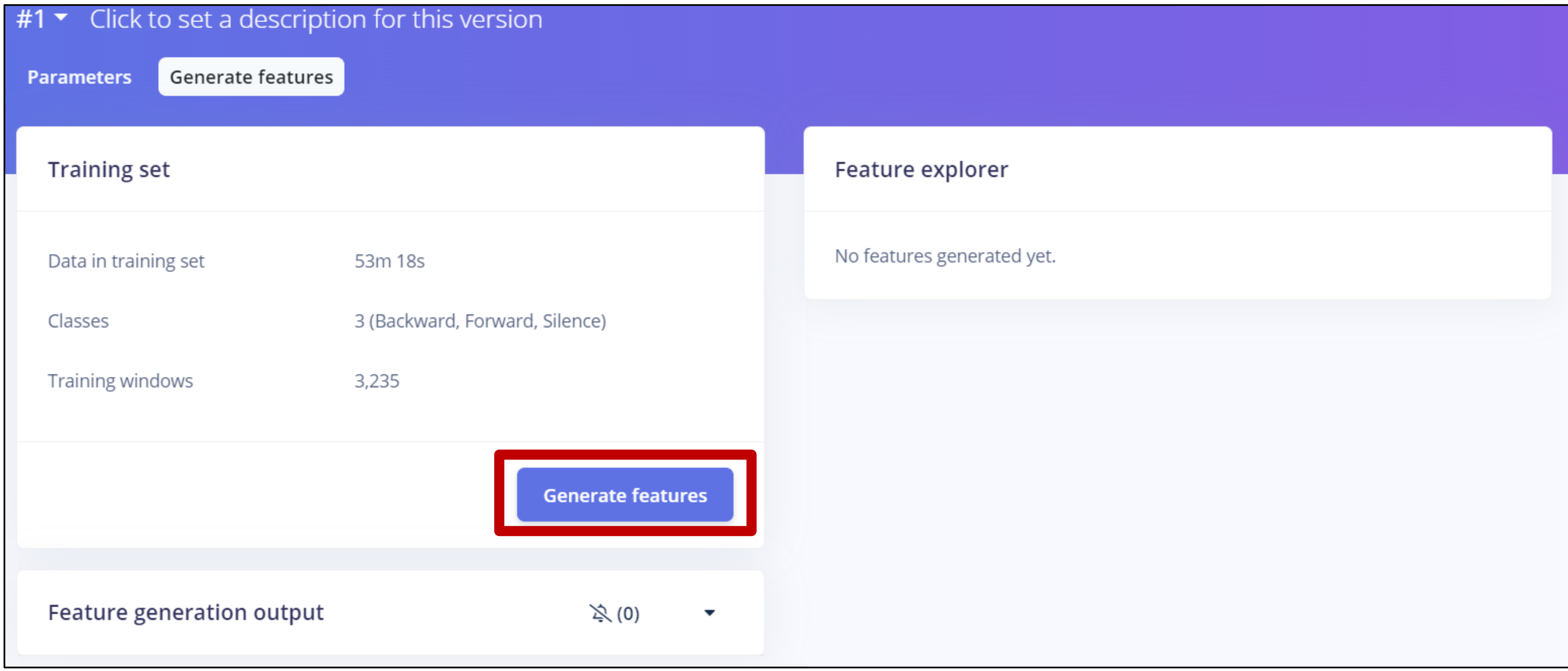
Backward MFCC

MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Impulse design
  - MFCC

MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Impulse design
  - MFCC

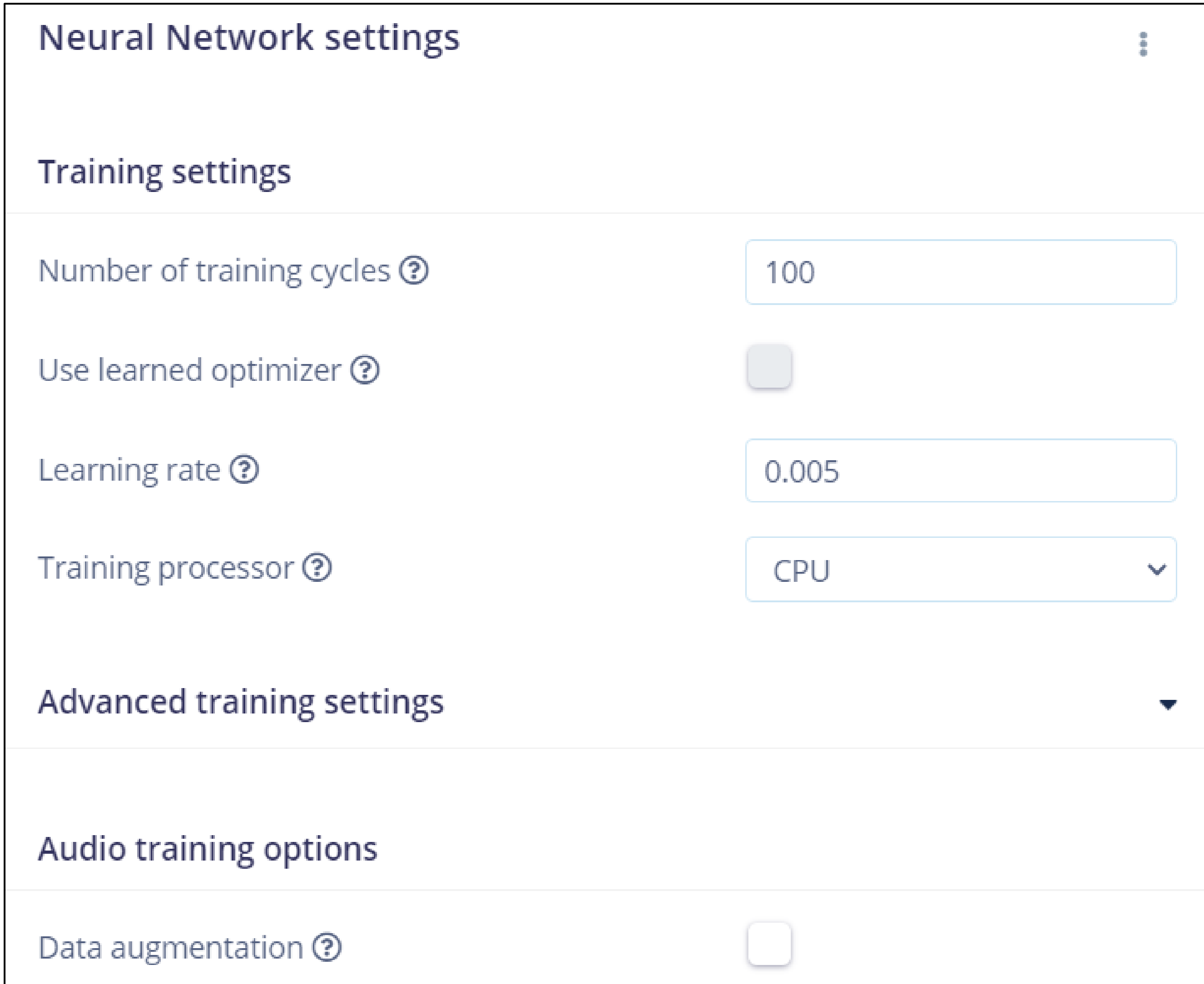


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Impulse design
  - MFCC

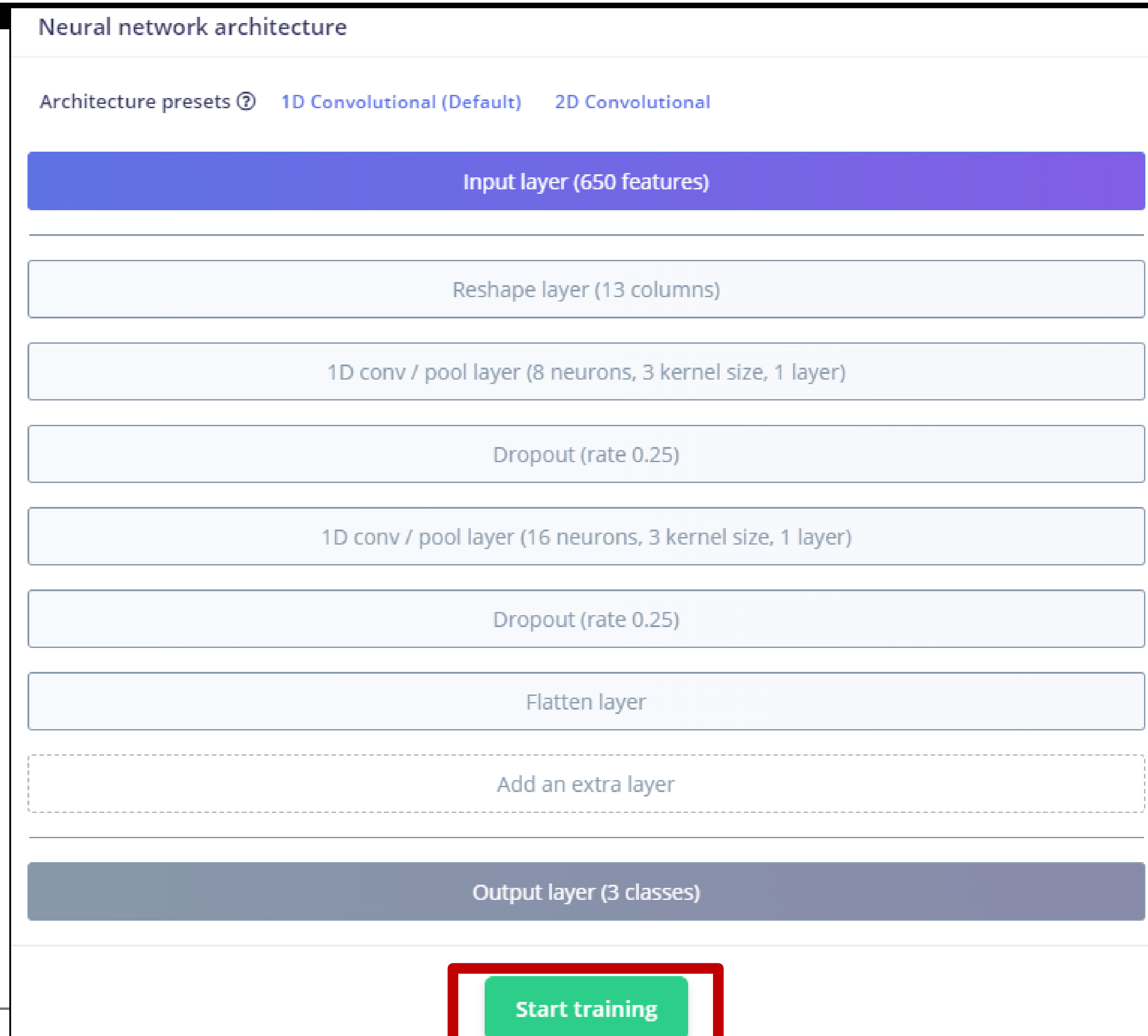
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Impulse design
  - Classifier



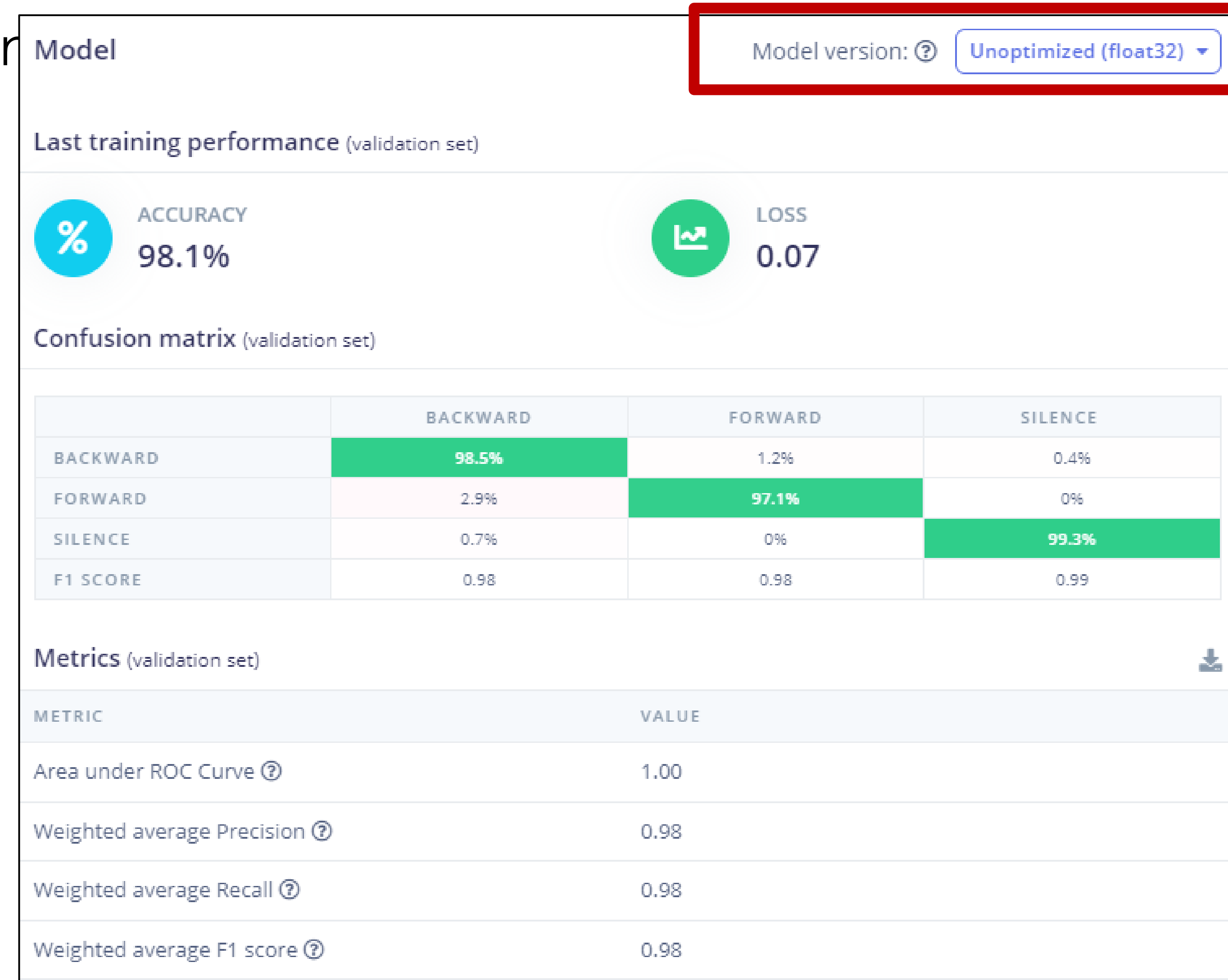
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Impulse design
  - Classifier



MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Impulse design
  - Classifier



Model

Model version: ⓘ Unoptimized (float32)

Last training performance (validation set)

ACCURACY 98.1%

LOSS 0.07

Confusion matrix (validation set)

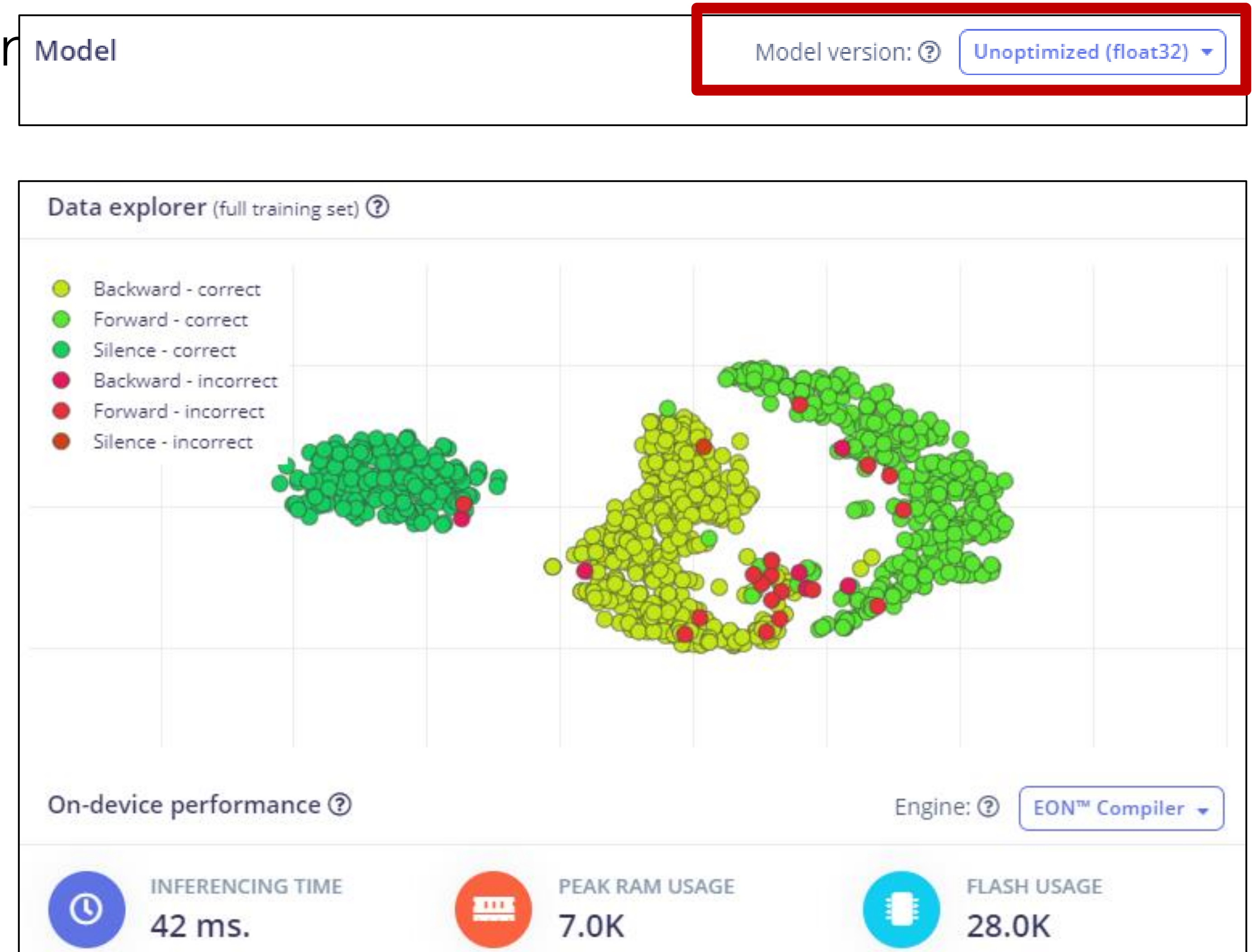
	BACKWARD	FORWARD	SILENCE
BACKWARD	98.5%	1.2%	0.4%
FORWARD	2.9%	97.1%	0%
SILENCE	0.7%	0%	99.3%
F1 SCORE	0.98	0.98	0.99

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve ⓘ	1.00
Weighted average Precision ⓘ	0.98
Weighted average Recall ⓘ	0.98
Weighted average F1 score ⓘ	0.98

MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Impulse design
  - Classifier



MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Impulse design
  - Classifier

Model

Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY
98.1%

LOSS
0.07

Confusion matrix (validation set)

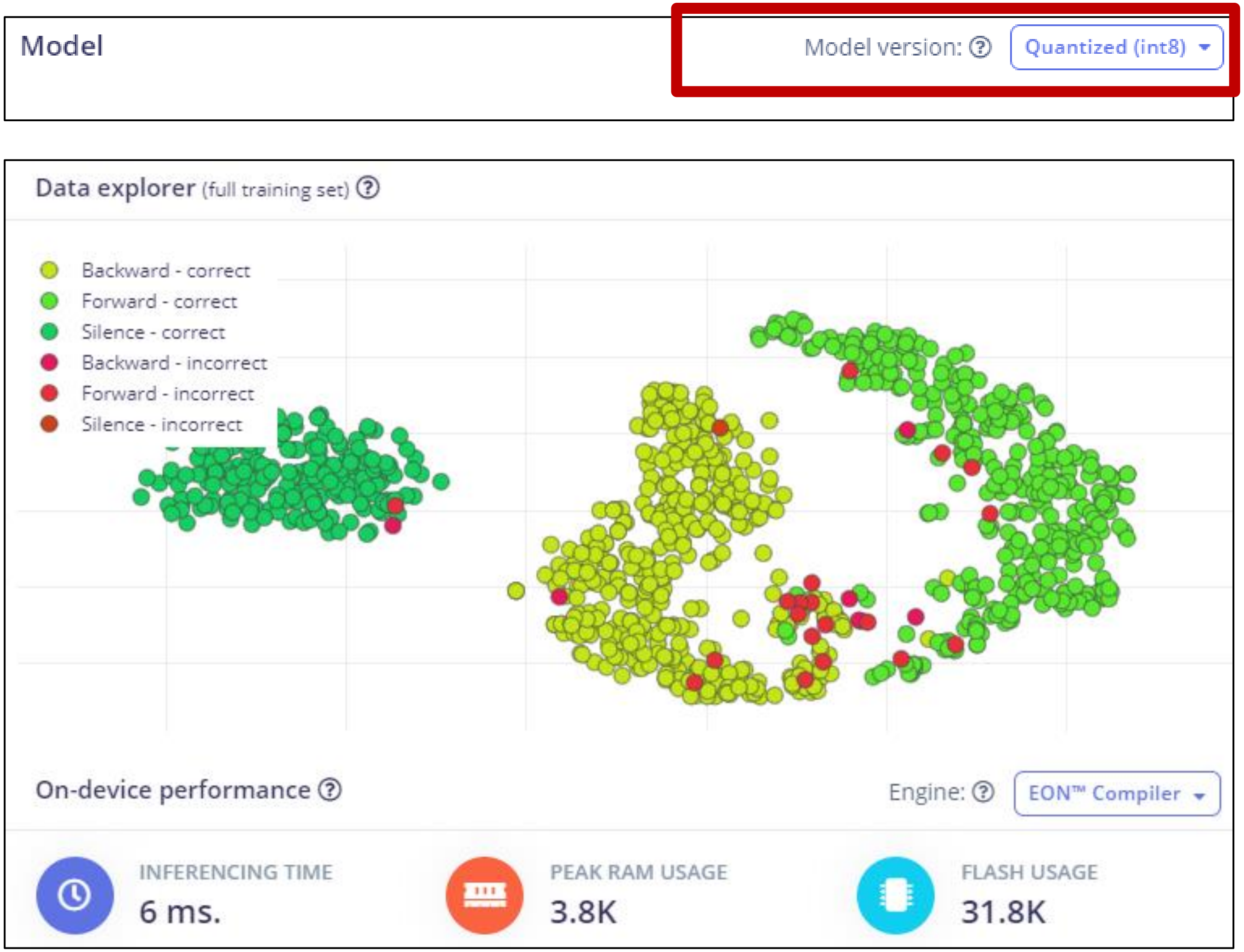
	BACKWARD	FORWARD	SILENCE
BACKWARD	98.5%	1.2%	0.4%
FORWARD	2.9%	97.1%	0%
SILENCE	0.7%	0%	99.3%
F1 SCORE	0.98	0.98	0.99

Metrics (validation set)

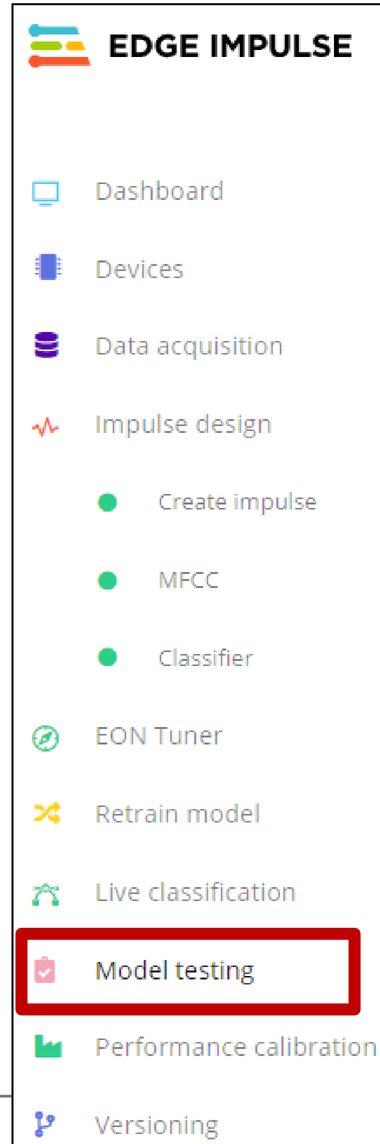
METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	0.98
Weighted average Recall	0.98
Weighted average F1 score	0.98

MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Impulse design
  - Classifier

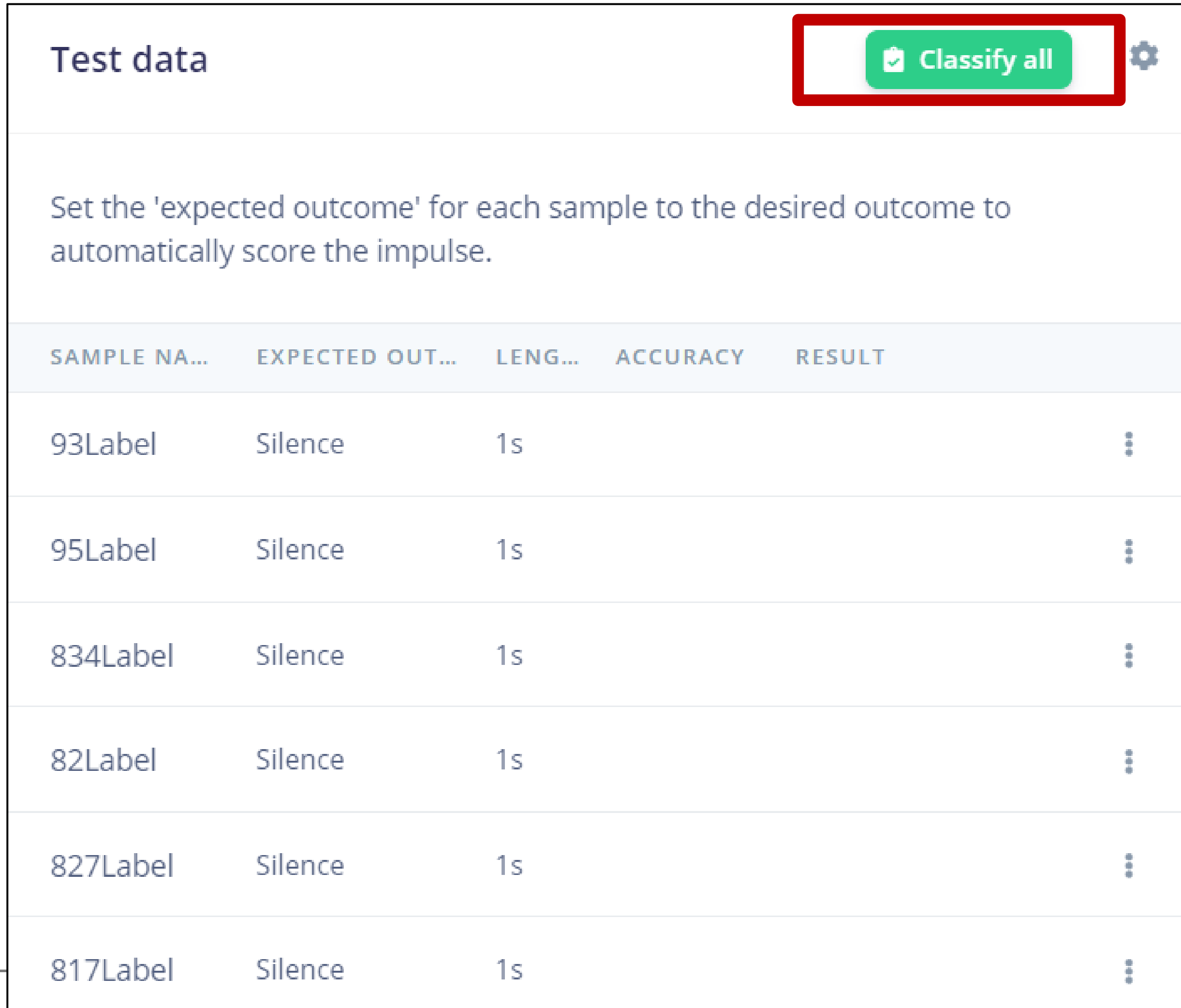


- Model testing



MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Model testing



MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Model testing

Results

Model version: ⓘ Unoptimized (float32)

ACCURACY

95.43%

Metrics for Classifier

METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	0.97
Weighted average Recall	0.97
Weighted average F1 score	0.97

Confusion matrix

	BACKWARD	FORWARD	SILENCE	UNCERTAIN
BACKWARD	94.8%	1.8%	0%	3.4%
FORWARD	2.3%	94.2%	0%	3.5%
SILENCE	0.7%	0%	99.3%	0%
F1 SCORE	0.96	0.96	1.00	

MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Model testing

Results

Model version: Quantized (int8)

ACCURACY

95.43%

Metrics for Classifier

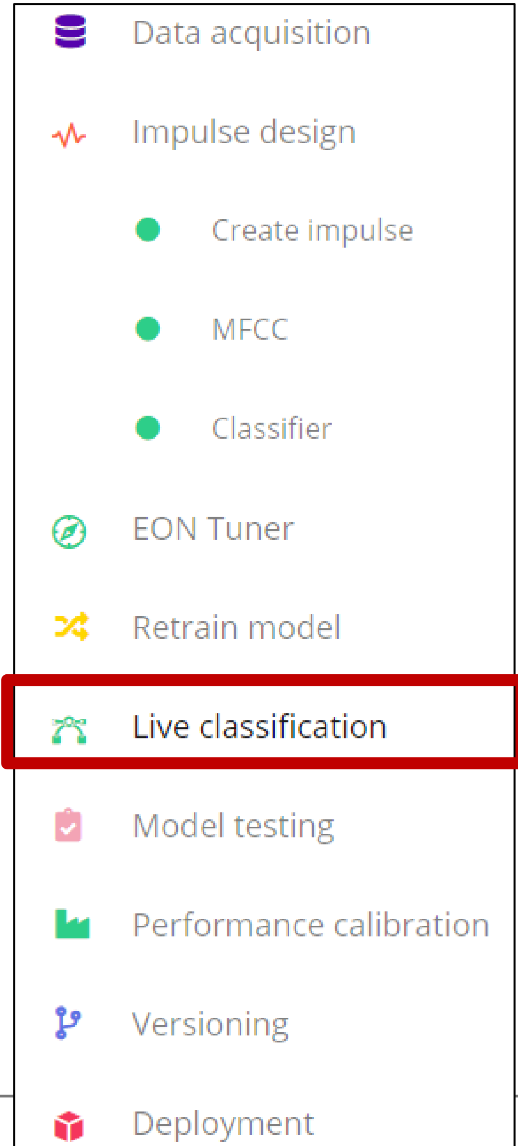
METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	0.97
Weighted average Recall	0.97
Weighted average F1 score	0.97

Confusion matrix

	BACKWARD	FORWARD	SILENCE	UNCERTAIN
BACKWARD	94.8%	1.8%	0%	3.4%
FORWARD	2.6%	94.2%	0%	3.2%
SILENCE	0.7%	0%	99.3%	0%
F1 SCORE	0.96	0.96	1.00	

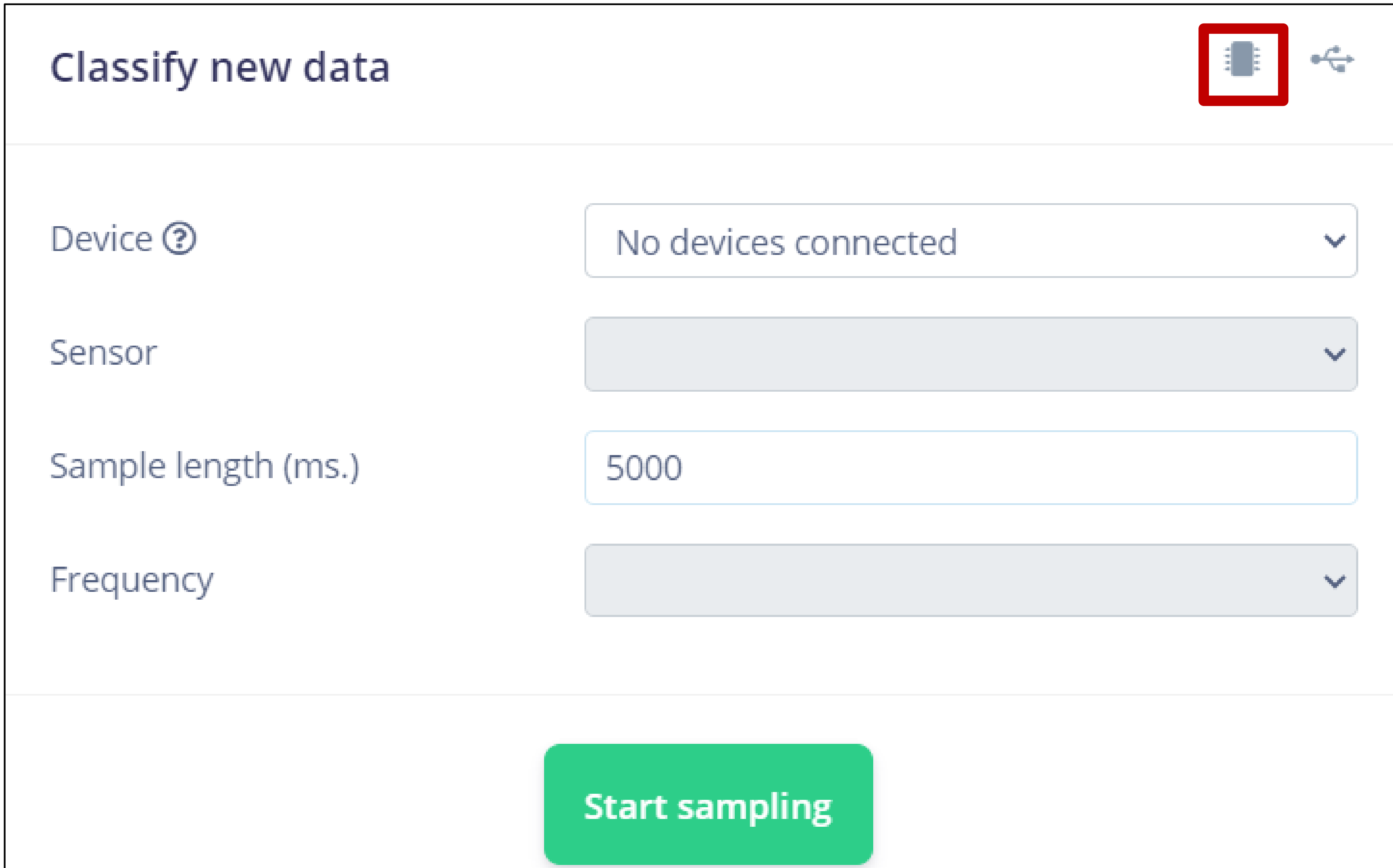
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Live Classification

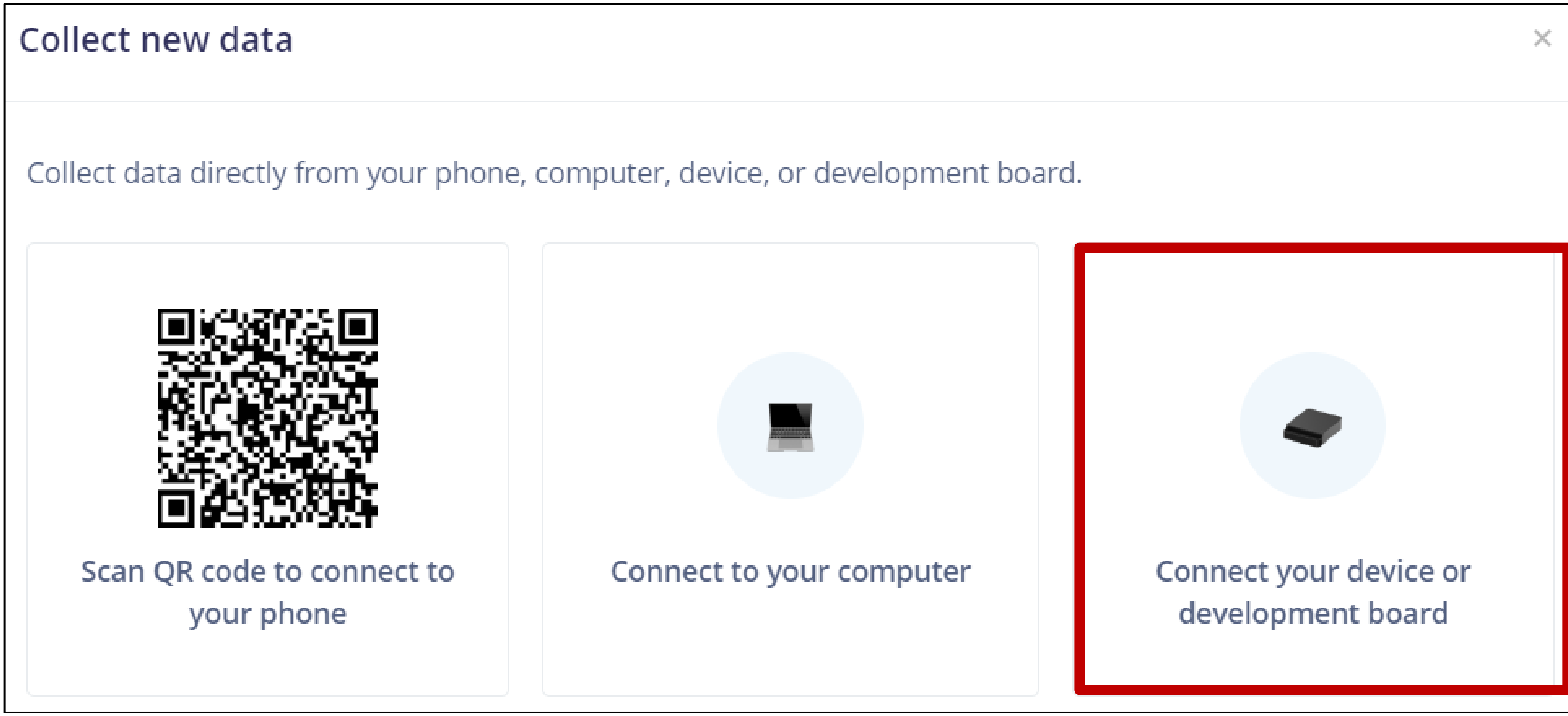


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Live Classification



❖ Live Classification



- Live Classification

Connect your device or development board ×

You can connect almost any device over serial using our data forwarder:

```
$ edge-impulse-data-forwarder
```

Or connect a supported device or development board:

Arduino Nano 33 BLE Sense ✖

To set up this device, download and install the software, and connect using the Edge Impulse CLI:

Arduino Nano 33 BLE Sense

View installation docs

To connect a device to a new project, run the CLI with `--clean`

MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Live Classification
  - Installing dependencies
    - To set this device up in Edge Impulse, you will need to install the following software:
      - Edge Impulse CLI (https://docs.edgeimpulse.com/docs/tools/edge-impulse-cli/cli-installation)
      - Arduino CLI (https://arduino.github.io/arduino-cli/1.0/)

        Here's an instruction video for Windows (https://www.youtube.com/watch?v=1jMWsFER-Bc)
      - The Arduino website (https://arduino.github.io/arduino-cli/1.0/installation/) has instructions for macOS and Linux
      - On Linux:

        GNU Screen: install for example via sudo apt install screen

- Live Classification
  - Connecting to Edge Impulse
    - With all the software in place it's time to connect the development board to Edge Impulse
      - https://www.youtube.com/watch?v=wOkMZUaPLUM

- Live Classification
  - Connect the development board to your computer
    - Use a micro-USB cable to connect the development board to your computer
    - Then press RESET twice to launch into the bootloader
    - The on-board LED should start pulsating to indicate this

MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Live Classification
  - Update the firmware
    - The development board does not come with the right firmware yet
    - To update the firmware:
    - Download the latest Edge Impulse firmware (https://cdn.edgeimpulse.com) and unzip the file
    - Open the flash script for your operating system (flash_windows.bat, flash_mac.command or flash_linux.sh) to flash the firmware
    - Wait until flashing is complete and press the RESET button once to launch the new firmware.

MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Live Classification
  - Setting keys
  - From a command prompt or terminal, run:
    - edge-impulse-daemon
  - This will start a wizard that will ask you to log in and choose an Edge Impulse project
  - If you want to switch projects run the command with --clean
  - Alternatively, recent versions of Google Chrome and Microsoft Edge can collect data directly from your development board without needing the Edge Impulse CLI
  - See this blog post for more information
    - https://edgeimpulse.com/blog/collect-sensor-data-straight-from-your-web-browser/

MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Live Classification

C:\WINDOWS\system32\cmd.exe - "node" "C:\Users\user\AppData\Roaming\npm\\node_modules'

```
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>edge-impulse-daemon
Edge Impulse serial daemon v1.19.3
? What is your user name or e-mail address (edgeimpulse.com)?
```

❖ Live Classification

Select C:\WINDOWS\system32\cmd.exe - "node" "C:\Users\user\AppData\Roaming\npm\\node_modules\edge-impulse-cli\build\cli\d

```
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>edge-impulse-daemon
Edge Impulse serial daemon v1.19.3
? What is your user name or e-mail address (edgeimpulse.com)? dennisgookyi@gmail.com
? What is your password? [input is hidden]
```

- Live Classification

```
Edge Impulse serial daemon v1.19.3
Endpoints:
    Websocket: wss://remote-mgmt.edgeimpulse.com
    API:       https://studio.edgeimpulse.com
    Ingestion: https://ingestion.edgeimpulse.com

? Which device do you want to connect to? COM5 (Microsoft)
[SER] Connecting to COM5
[SER] Serial is connected, trying to read config...
Failed to parse snapshot line [   ]
[SER] Retrieved configuration
[SER] Device is running AT command version 1.8.0

? To which project do you want to connect this device? (Use arrow keys)
> Dennis / my-smart-phone-motion-project
  Dennis / my-arduinonano-motion-project
  Dennis / meee
  Dennis / my-keyword-spotting-project
  Dennis / keyword-spotting-using-jupyter-notebook
  Dennis / maize_disease_detection
  Knust / enabling_deep_learning_on_edge_dev
  Dennis / for-python-sdk
  Dennis / Cifar_Dogs_vs_Cats
  Dennis / ai4drainproject
  Dennis / ai4dsubsample
  Dennis / try
  Dennis / CollectData
  Dennis / data
  Dennis / cvcdata
  Dennis / HIGHLOW
  Dennis / NextGenWeatherStation
  Dennis / Robotic_SubSystem
```

MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

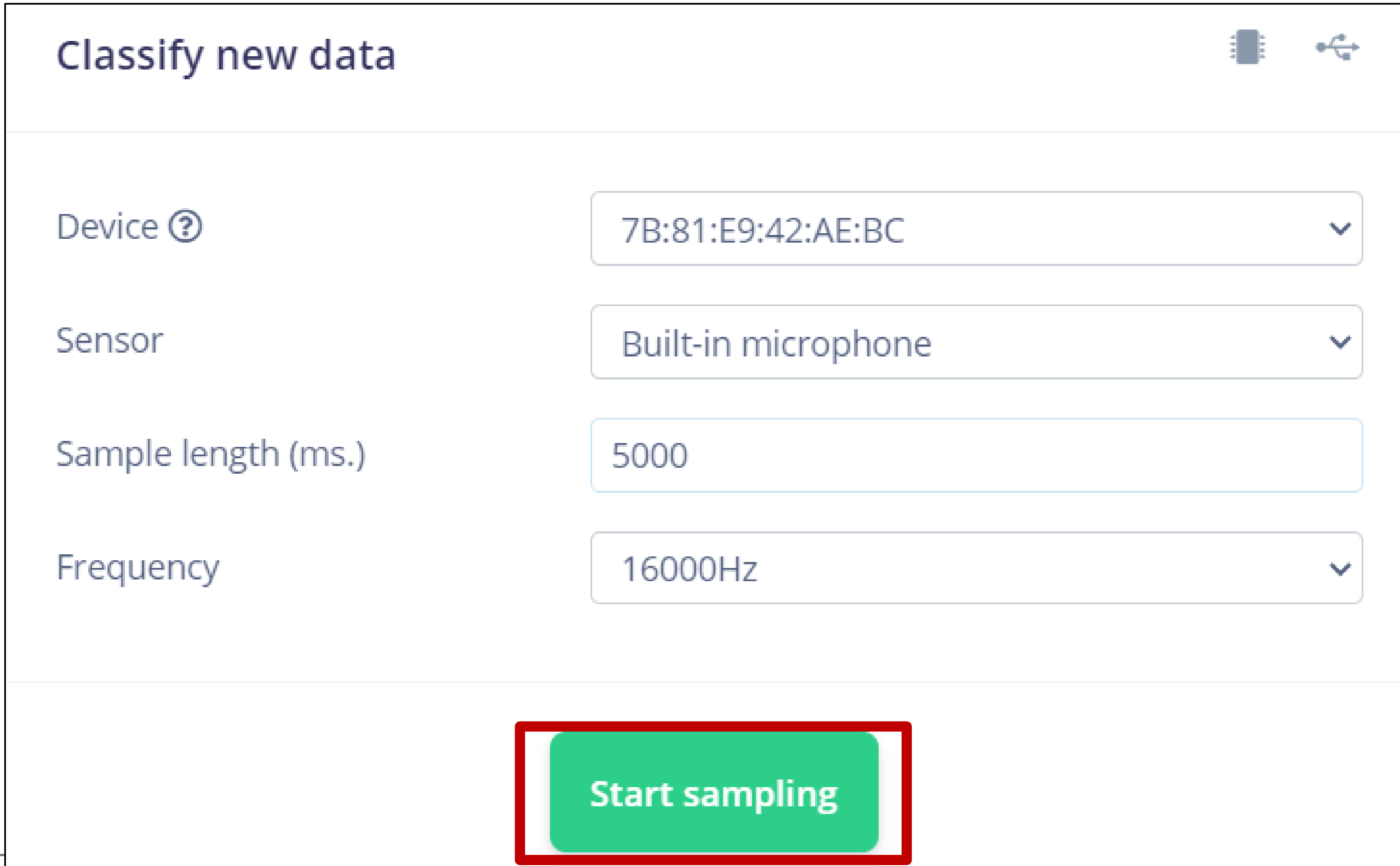
- Live Classification

C:\WINDOWS\system32\cmd.exe - "node" "C:\Users\user\AppData\Roaming\npm\\node_modules\edge-impulse-cli\build\cli\daemon.js"

```
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>edge-impulse-daemon
Edge Impulse serial daemon v1.19.3
Endpoints:
    Websocket: wss://remote-mgmt.edgeimpulse.com
    API:       https://studio.edgeimpulse.com
    Ingestion: https://ingestion.edgeimpulse.com

? Which device do you want to connect to? COM5 (Microsoft)
[SER] Connecting to COM5
[SER] Serial is connected, trying to read config...
Failed to parse snapshot line [  ]
[SER] Retrieved configuration
[SER] Device is running AT command version 1.8.0

? To which project do you want to connect this device? Dennis / Robotic_SubSystem
Setting upload host in device... OK
Configuring remote management settings... OK
Configuring API key in device... OK
Configuring HMAC key in device... OK
Failed to parse snapshot line [  ]
Failed to parse snapshot line [  ]
[SER] Device is not connected to remote management API, will use daemon
[WS ] Connecting to wss://remote-mgmt.edgeimpulse.com
[WS ] Connected to wss://remote-mgmt.edgeimpulse.com
? What name do you want to give this device? NANO1
[WS ] Device "NANO1" is now connected to project "Robotic_SubSystem". To connect to another project, run `edge-impulse
aemon --clean`.
[WS ] Go to https://studio.edgeimpulse.com/studio/423136/acquisition/training to build your machine learning model!
```

MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Live Classification



MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Live Classification

Classification result

Summary

Model version: Quantized (int8)

Name	testing.5140e2er
Label	testing

CATEGORY	COUNT
Backward	4
Forward	0
Silence	2
uncertain	3

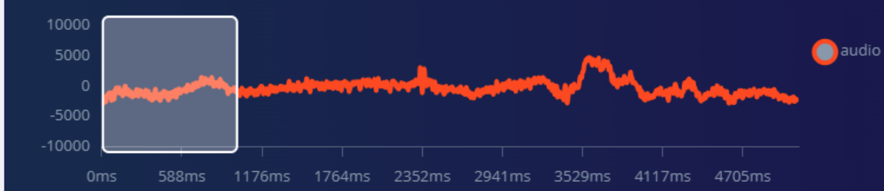
Detailed result

Show only unknowns

TIMESTAMP	BACKWARD	FORWARD	SILENCE
0	0.01	0.02	0.98
500	0.66	0.25	0.08
1,000	0.61	0.29	0.10
1,500	0.26	0.26	0.48

RAW DATA

testing.5140e2er



0:00 / 0:01

Raw features

-2170, -2173, -2205, -2199, -2187, -2153, -2146, -2121, -2136, -2140, -2120, -2088, -2041, -2051, -2051, -203…

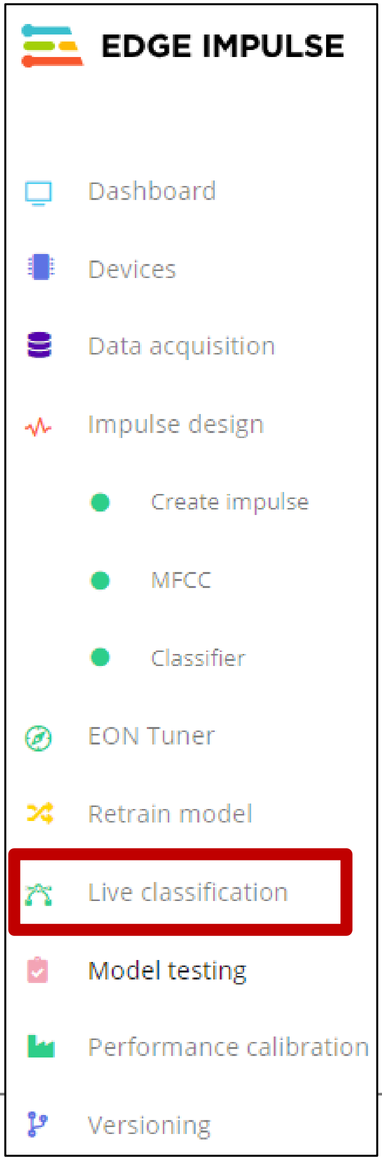
MFCC

Processed features

4.2478, -1.0004, 0.4061, 0.9148, 0.1024, -0.1381, 0.7006, 0.4282, 0.4814, 0.1135, 0.5369, 1.2589, 0.7001, 1.6…

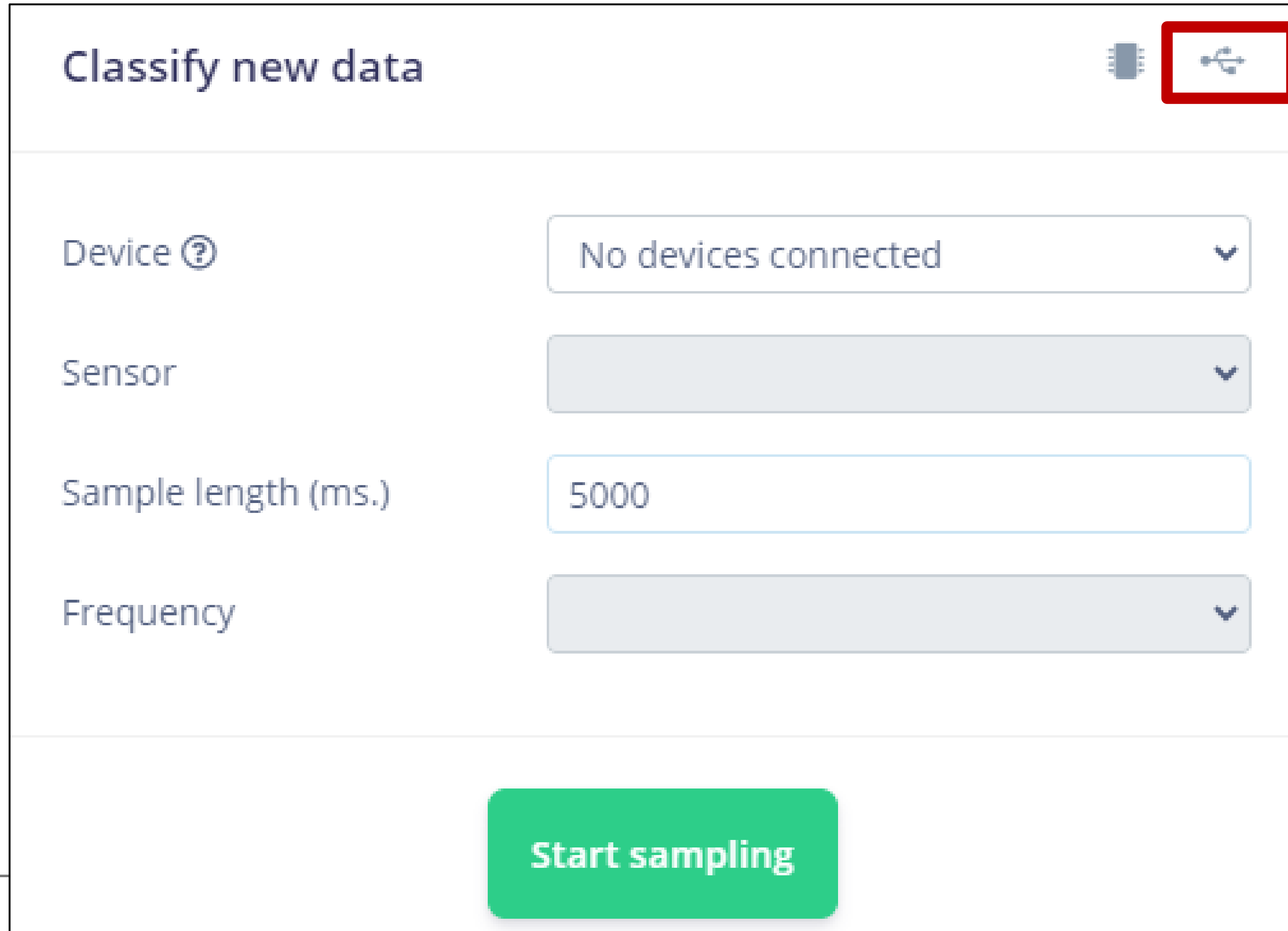
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Live classification
  - Using WebUSB

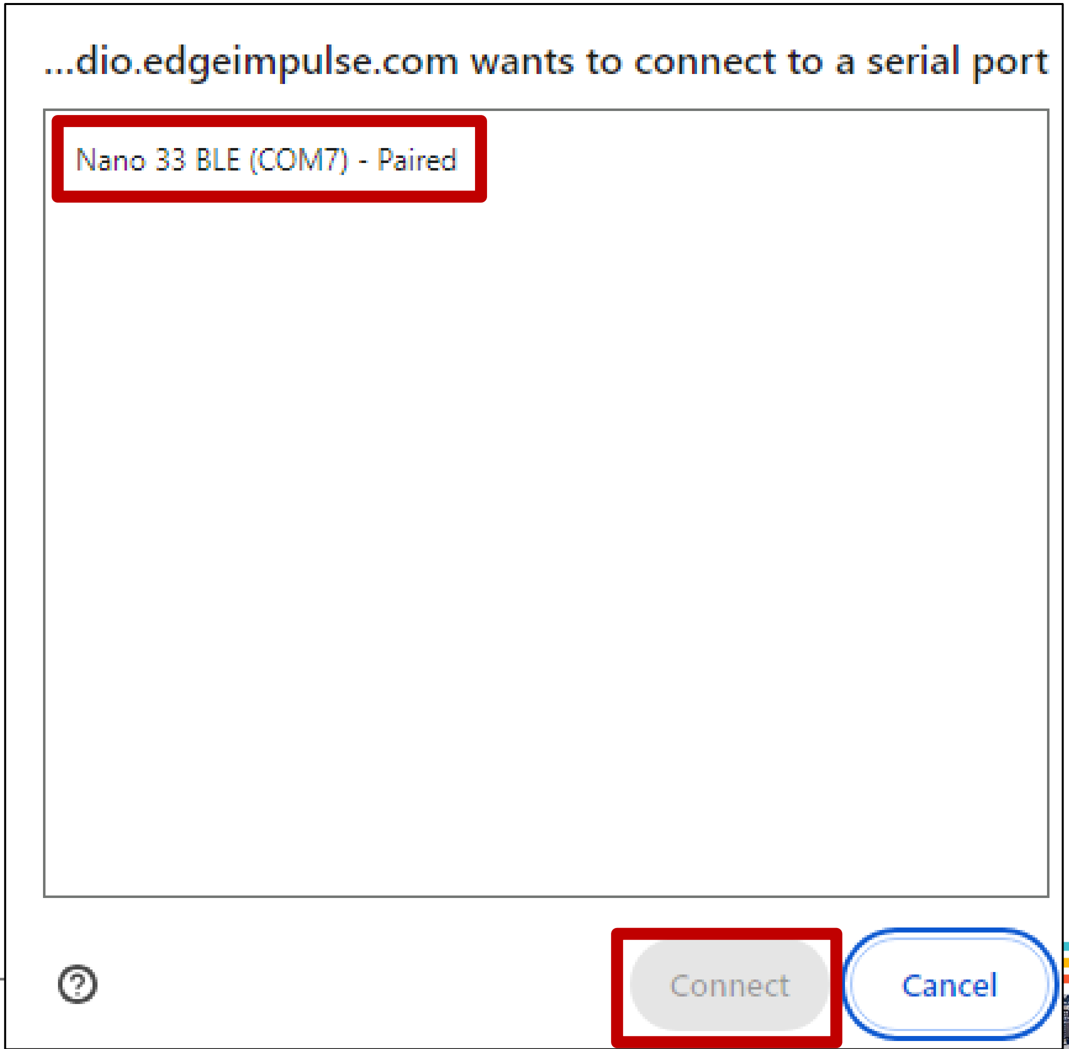


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

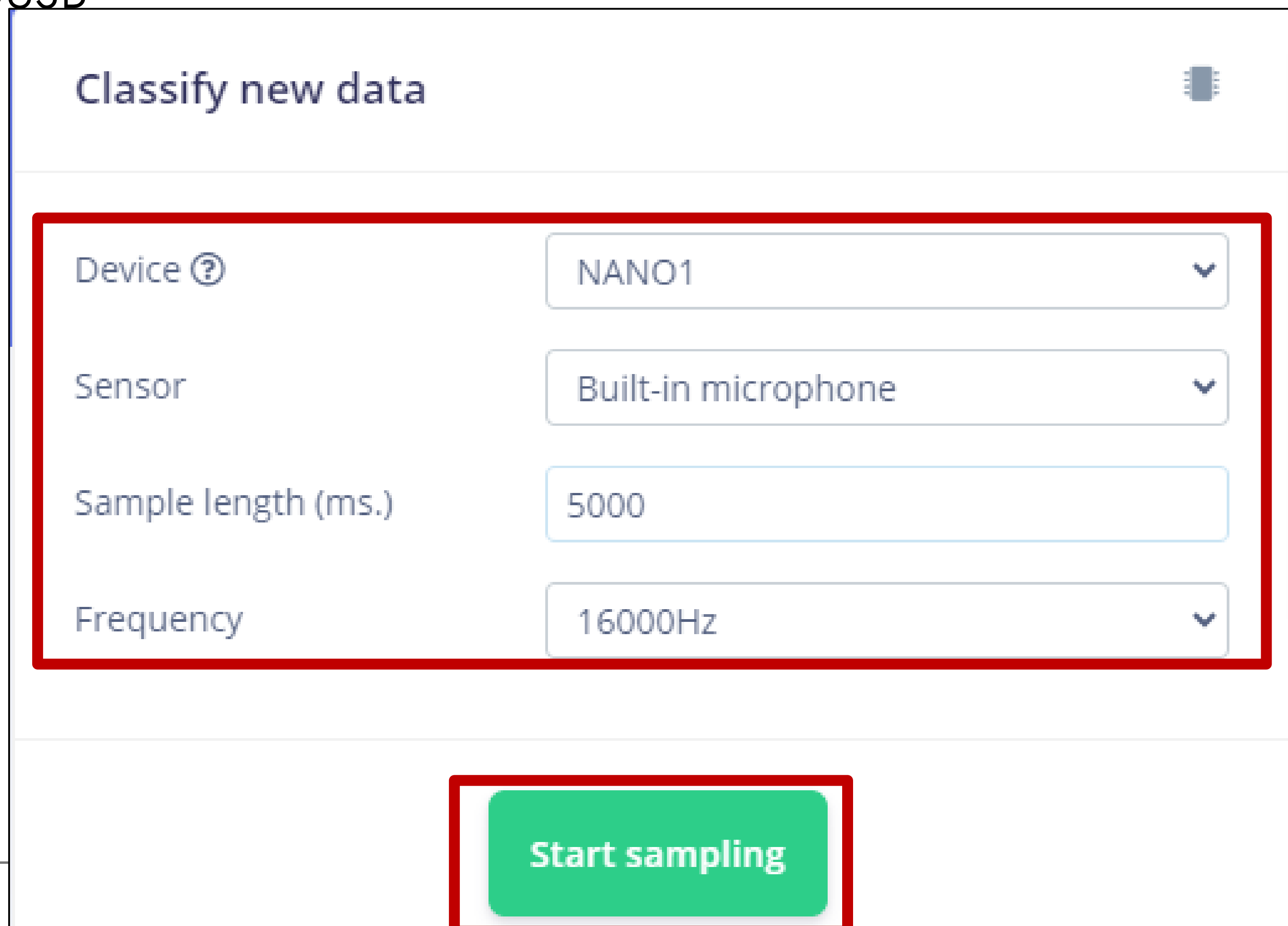
- Live classification
  - Using WebUSB



- Live classification
  - Using WebUSB

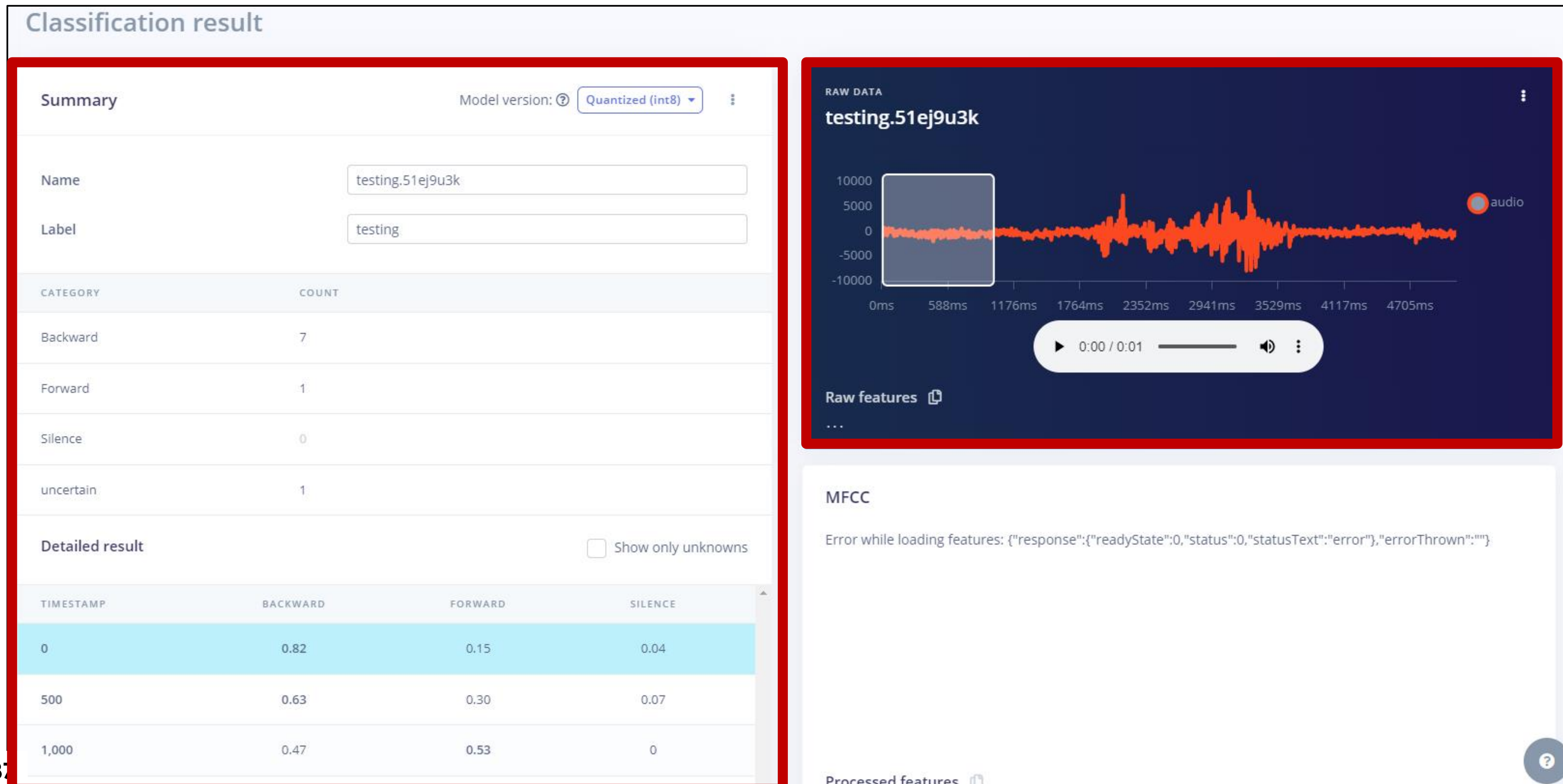


- Live classification
  - Using WebUSB



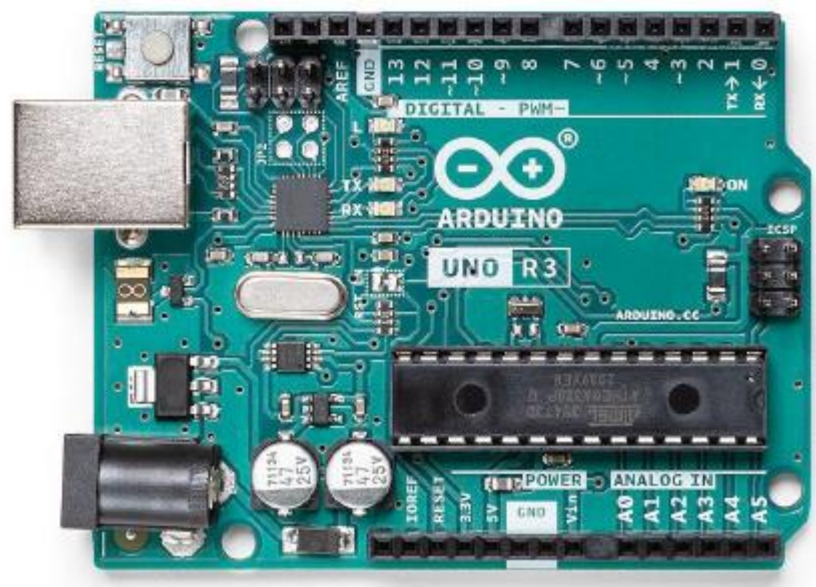
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- Live classification
  - Using WebUSB



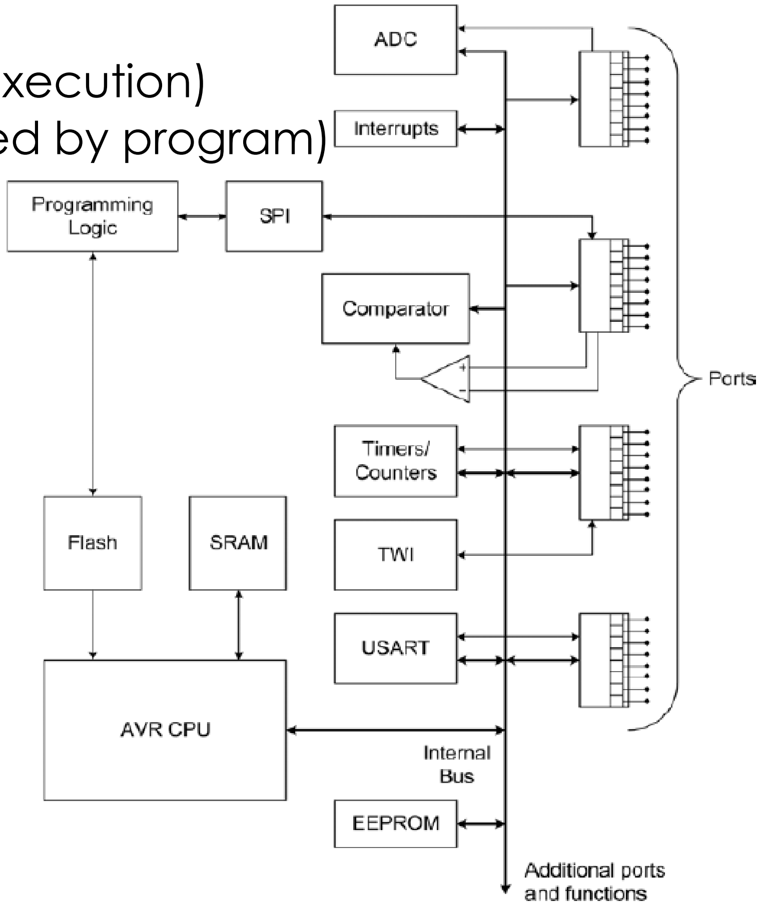
MODEL DEPLOYMENT HARDWARE - ARDUINO

- Arduino's purpose is to control things by interfacing with sensors and actuators
  - No keyboard, mouse and screen
  - Can be attached via "shields"
  - No operating system, limited memory
  - A single program enjoys 100% of CPU time
- Physical Arduino boards
  - Uno
  - Nano
  - Leonardo
  - Mega
  - Pro Mini
- Arduino IDE
  - Installed on a PC (Windows/Mac/Linux)
  - To develop, install and debug programs on Arduino boards
  - Communicates with Arduino board over USB
- Third party Arduino compatible boards
  - STM32
  - Nucleo / Discovery / Feather,
  - Adafruit
  - SparkFun



MODEL DEPLOYMENT HARDWARE - ARDUINO

- A generic AVR microcontroller block diagram
  - CPU
  - Internal Memory
    - Flash (stores program code)
    - SRAM (holds data and variable during execution)
    - EEPROM (holds persistent data generated by program)
  - Peripherals
    - A/D converter (ADC)
    - Timers
    - UART
    - SPI
    - DMA
    - GPIO
    - TWI
    - Comparator
    - RTC
    - WDT
    - RNG



MODEL DEPLOYMENT HARDWARE - ARDUINO

- Interfacing with Arduino
  - Temperature sensor
  - Pressure sensor
  - Switches
  - Variable resistor
  - Range finder
  - PIR (person
  - in room) sensor
  - Relay
  - Motor control
  - LED
  - 16x2 display
  - Graphic display
  - Bluetooth shield
  - WiFi
  - shield
  - Ethernet shield

Arduino Libraries:
https://www.arduinolibraries.info/libraries

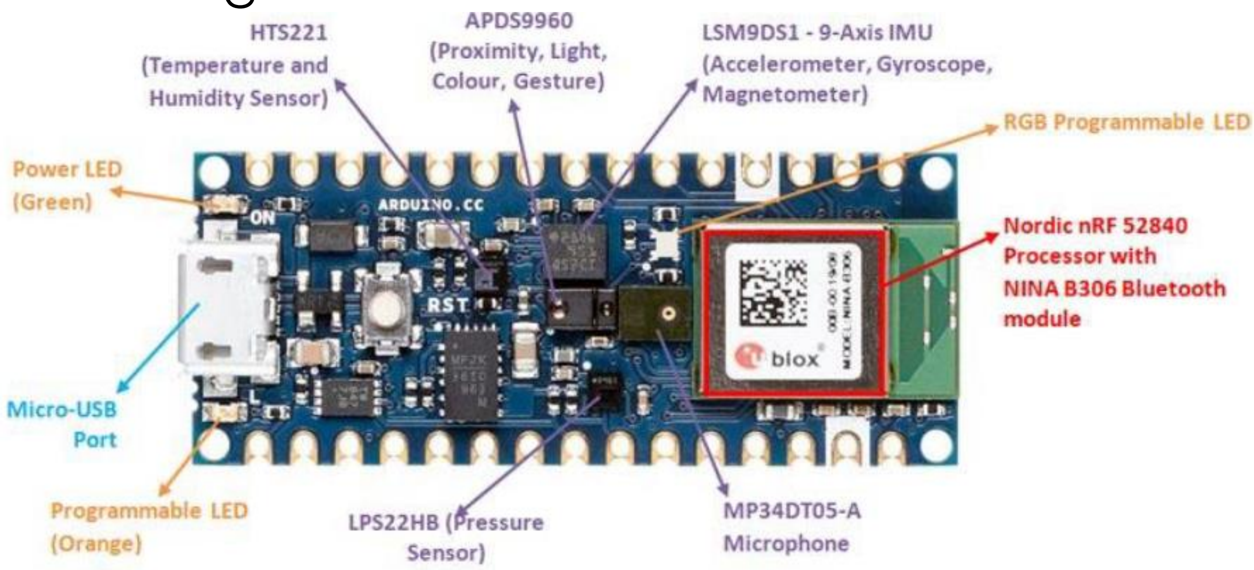
MODEL DEPLOYMENT HARDWARE - ARDUINO

❖ Uno vs Nano 33 BLE Sense

	Uno R3	Nano 33 BLE Sense
Chip	ATmega328P	nRF52840
Clock	16 MHz	64 MHz
Flash	32 KB	1 MB
SRAM	2 KB	256 KB
EEPROM	1 KB	none
Input Voltage	6 - 20 V	4.5 - 21 V
I/O Voltage	5 V	3.3 V
Pinout	14 digital, 6 PWM, 6 AnalogIn	14 digital (PWM), 8 AnalogIn
Interfaces	USB, SPI, I2C, UART	USB, SPI, I2C, I2S, UART
Connectivity	via shields	BLE 5.0
Weight	25 g	5 g

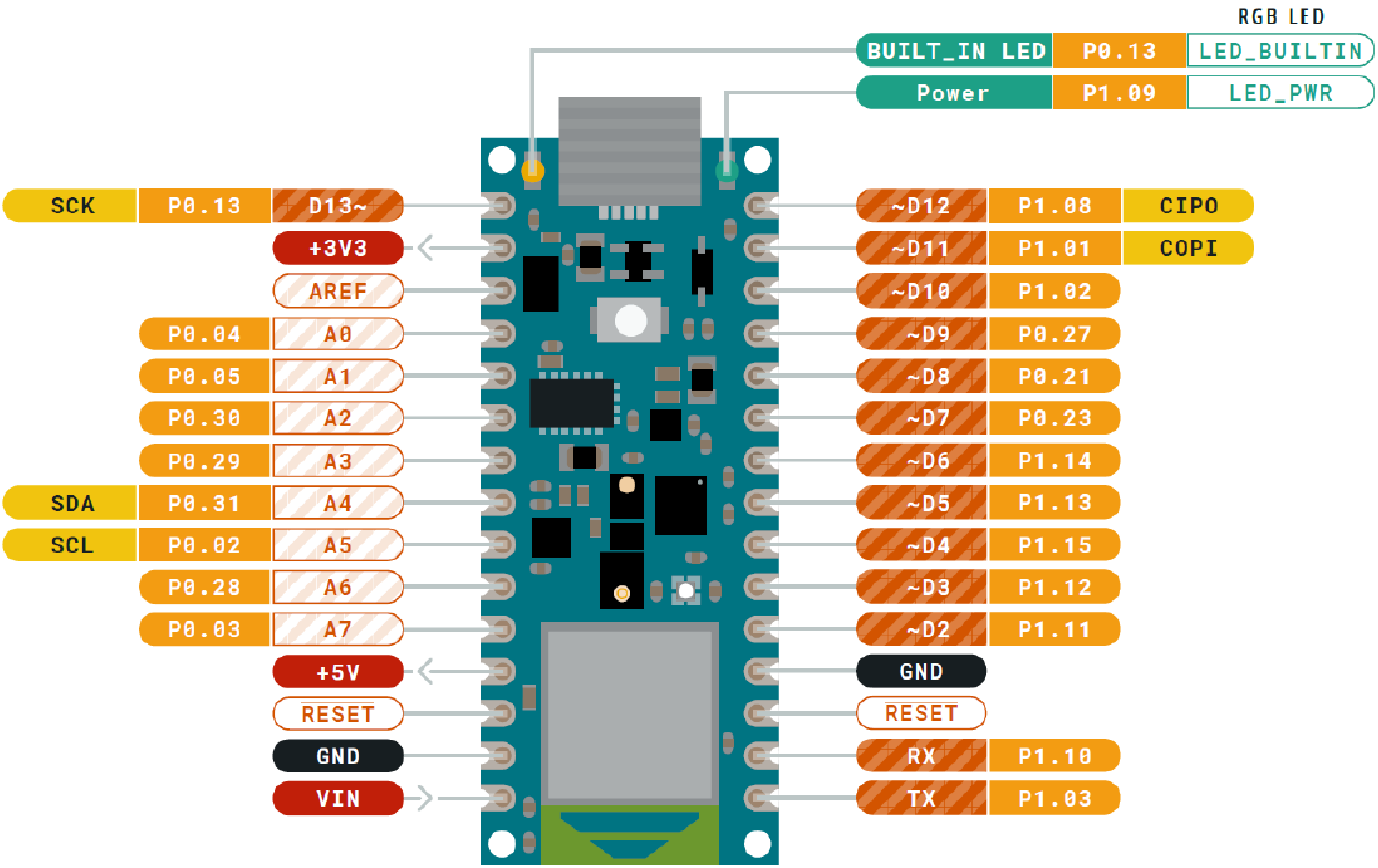
MODEL DEPLOYMENT HARDWARE - ARDUINO

- Features of Nano 33 Sense
  - 8 Analog Input Pins can provide 12-bit ADC at about 30 kHz
  - Integrated sensors (IMU, Mic, Light, Pressure, Temperature, Humidity)
  - All digital pins can trigger interrupts
  - Only supports 3.3V I/Os and is NOT 5V tolerant so please make sure you are not directly connecting 5V signals to this board or it will be damaged



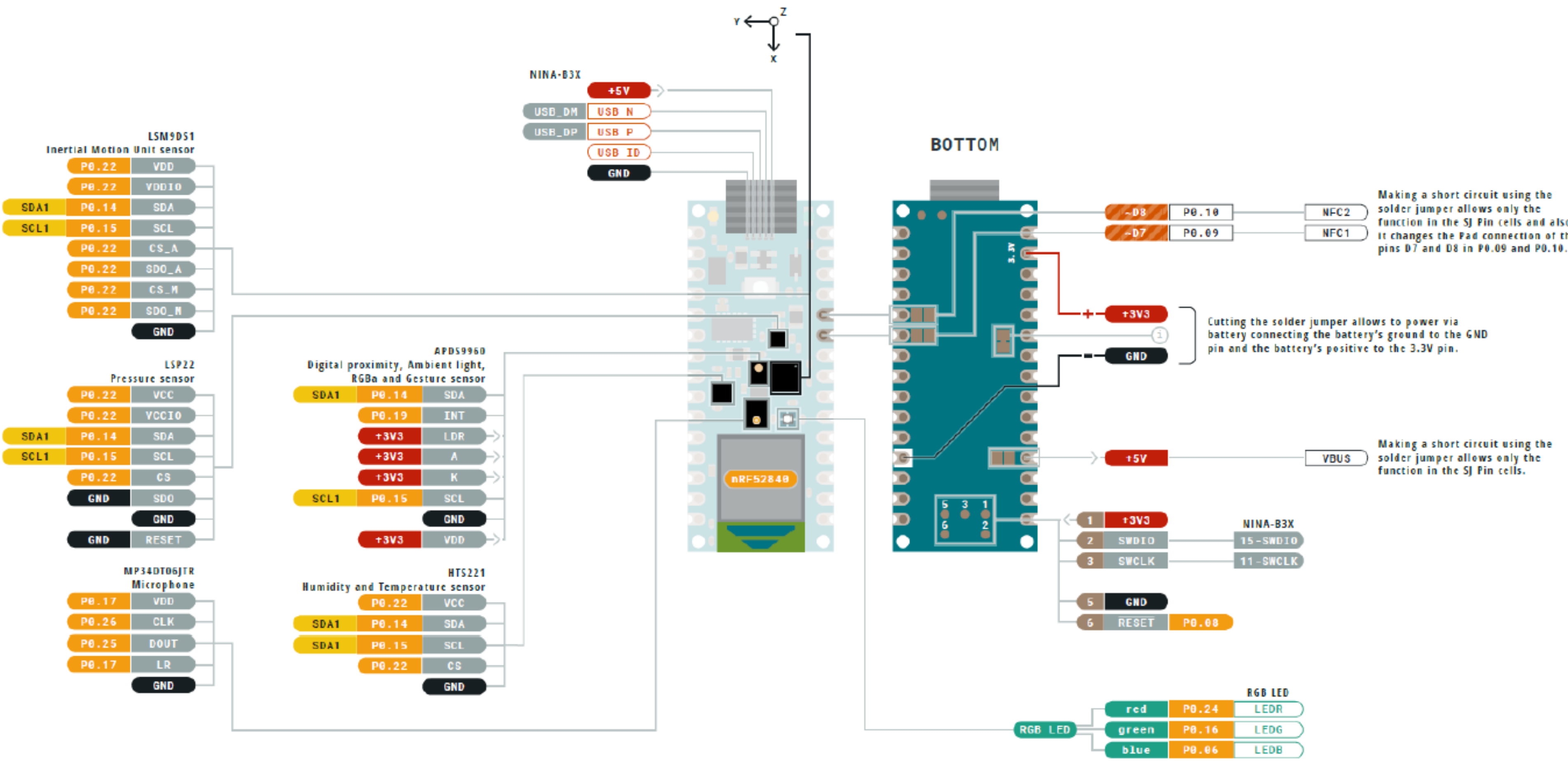
MODEL DEPLOYMENT HARDWARE - ARDUINO

❖ Nano 33 Sense



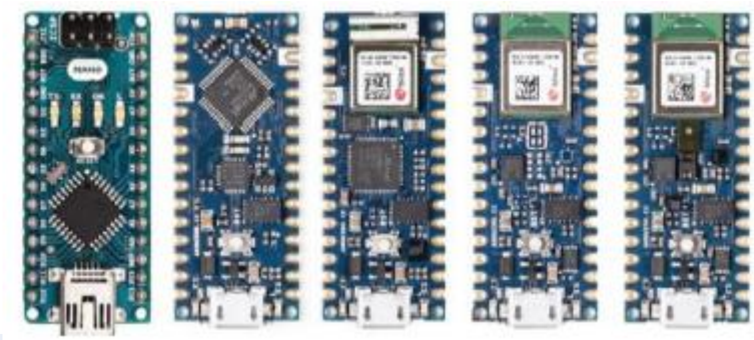
MODEL DEPLOYMENT HARDWARE - ARDUINO

❖ Nano 33 Sense



MODEL DEPLOYMENT HARDWARE - ARDUINO

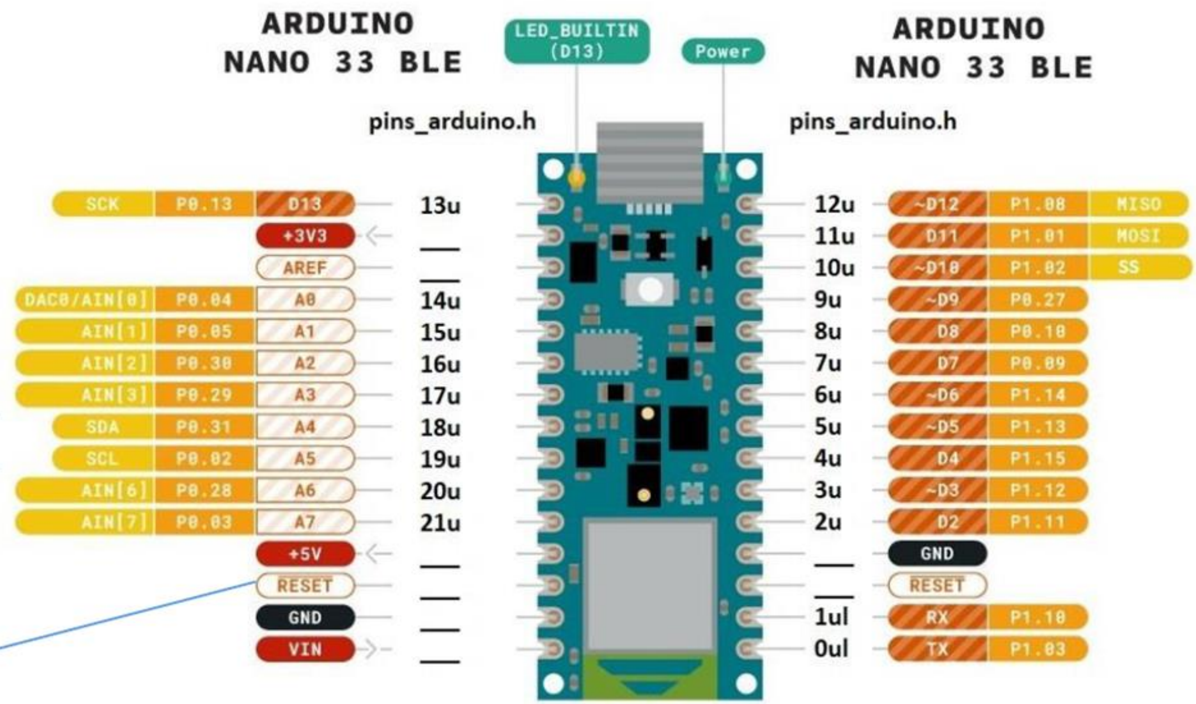
- Arduino Nano comparisons



Property	Arduino Nano	Arduino Nano Every	Arduino Nano 33 IoT	Arduino Nano 33 BLE	Arduino Nano 33 BLE Sense
Microcontroller	ATmega328	ATMega4809	SAMD21 Cortex®-M0+ 32bit low power ARM MCU	nRF52840 (ARM Cortex M4)	nRF52840 (ARM Cortex M4)
Operating voltage	5 V	5 V	3.3 V	3.3 V	3.3 V
Input voltage (VIN)	6-20 V	7-21 V	5-21 V	5-21 V	5-21 V
Clock speed	16 Mhz	20 MHz	48 MHz	64 MHz	64 MHz
Flash	32 KB	48 KB	256 KB	1 MB	1 MB
RAM	2 KB	6 KB	32 KB	256 KB	256 KB
Current per pin	40 mA	40 mA	7 mA	15 mA	15 mA
PWM pins	6	5	11	All	All
IMU	No	No	LSM6DS3 (6-axis)	LSM9DS1 (9-axis)	LSM9DS1 (9-axis)
Other sensors	No	No	No	No	Several
WiFi	No	No	Yes	No	No
Bluetooth	No	No	Yes	Yes	Yes
USB type	Mini	Micro	Micro	Micro	Micro

MODEL DEPLOYMENT HARDWARE - ARDUINO

❖ Pinout

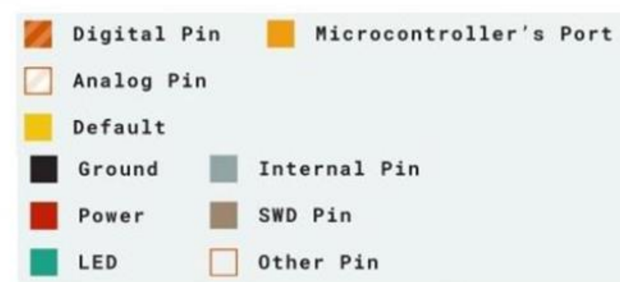


Pins A4 and A5 have an internal pull up and default to be used as an I2C Bus so usage as analog inputs is not recommended.

Ground the RESET pin to reset

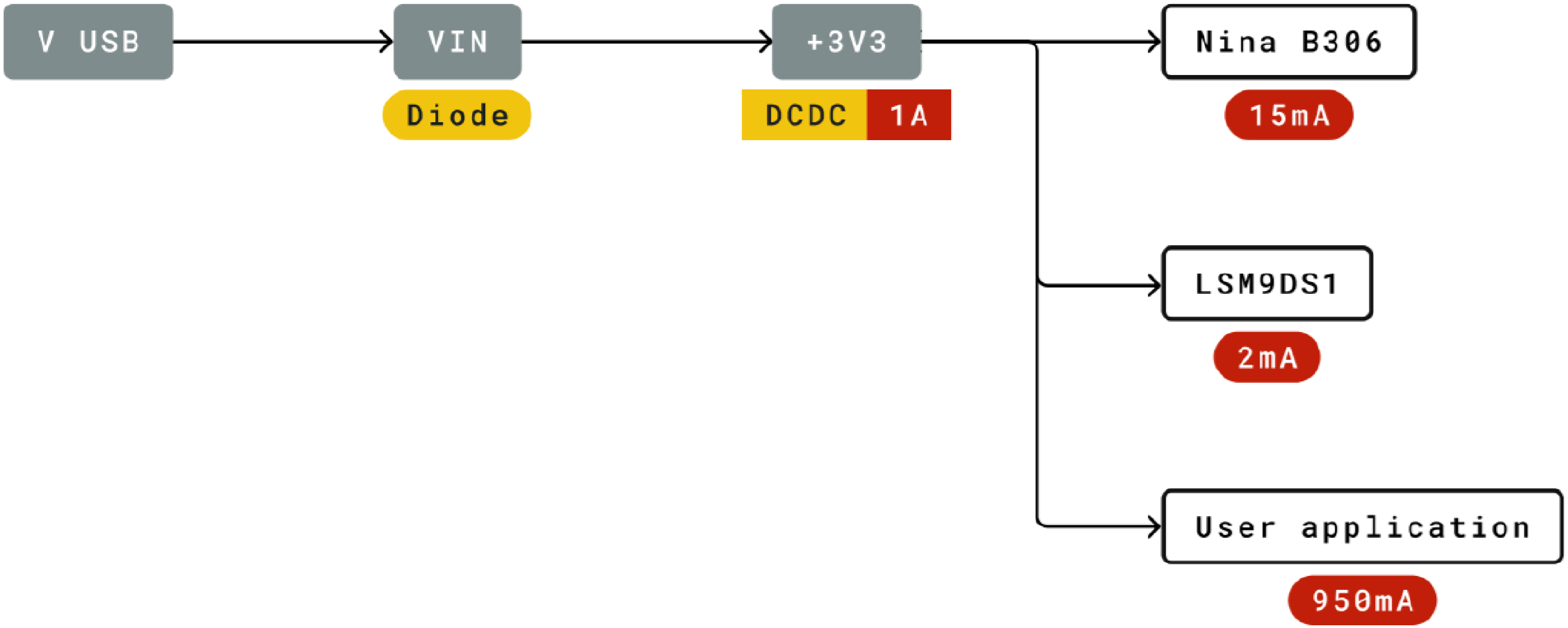
pins_arduino.h ???

(22u) LEDR
(23u) LEDG
(24u) LEDB
(25u) LED_PWR
(26u) PIN_INT_APDS
(27) PIN_PDM_PWR
(28) PIN_PDM_CLK
(29) PIN_PDM_DIN
(30u) PIN_WIRE_SDA1
(31u) PIN_WIRE_SCL1
(32u) PIN_ENABLE_SENSORS_3V3
(33u) PIN_ENABLE_I2C_PULLUP



MODEL DEPLOYMENT HARDWARE - ARDUINO

❖ Nano 33 BLE Power Tree



All Arduino boards have a built-in bootloader which allows flashing the board via USB. In case a sketch locks up the processor and the board is not reachable anymore via USB it is possible to enter bootloader mode by double-tapping the reset button right after power up.

MODEL DEPLOYMENT HARDWARE - ARDUINO

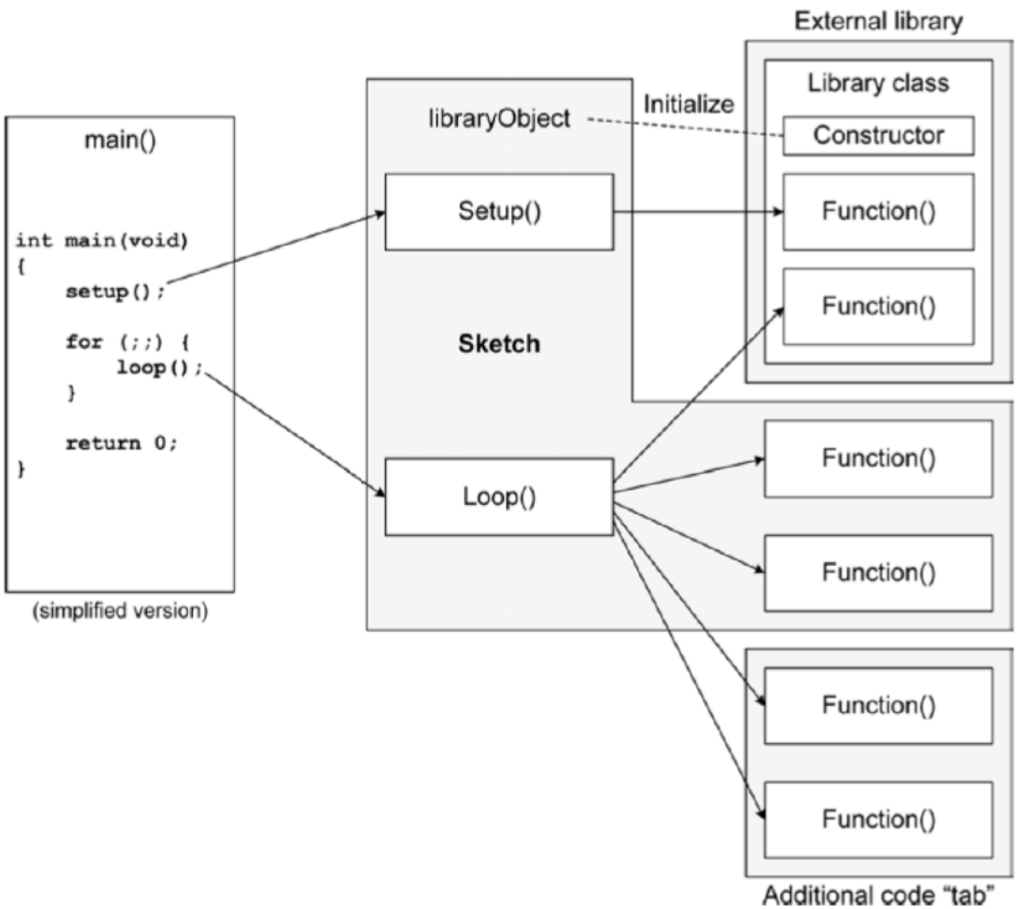
- nRF52840
  - Arduino Nano BLE (and BLE Sense) is based on the nRF52840 microprocessor made by Nordic
  - nRF52840 has the ARM Cortex M4 processor with single precision floating point unit (FPU)
  - The nRF52840 contains 1 MB of flash and 256 kB of RAM that can be used for code and data storage
  - The flash can be read an unlimited number of times by the CPU, but it has restrictions on the number of times it can be written and erased (minimum 10,000 times) and also on how it can be written
  - The flash is divided into 256 pages of 4 kB each that can be accessed by the CPU via both the ICODE and DCODE buses

MODEL DEPLOYMENT HARDWARE - ARDUINO

❖ Arduino Sketch Structure

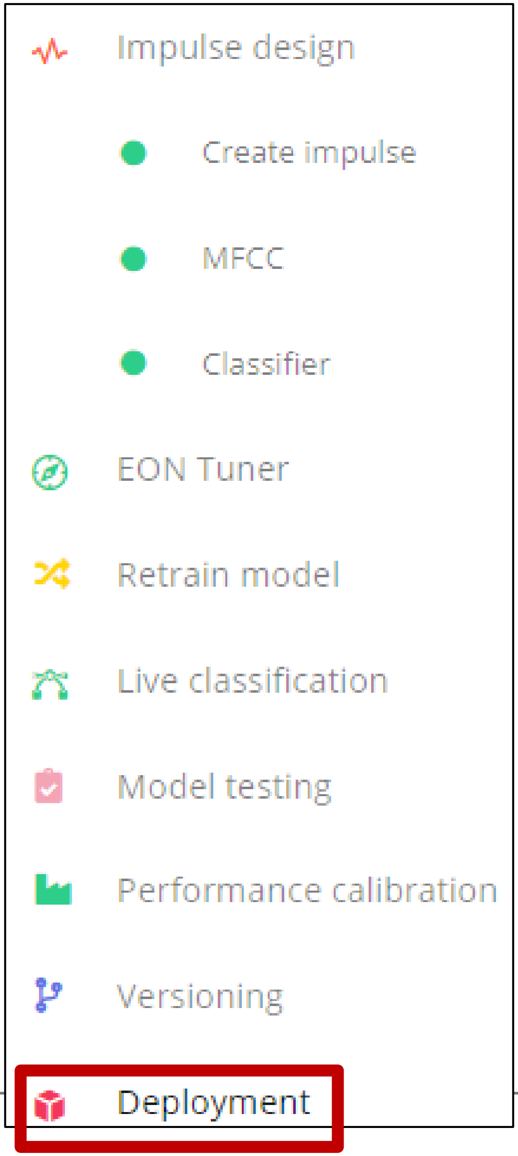
```
void setup() {
  // put your setup code here, to run once:

}
void loop() {
  // put your main code here, to run repeatedly:

}
```

```
int main(void)
{
    init();
    initVariant();

    #ifdefined(USBCON)
    USBDevice.attach();
    #endif

    setup();
    for (;;) {
        loop();
        if (serialEventRun) serialEventRun();
    }
    return 0;
}
```



MODEL DEPLOYMENT HARDWARE

- Model Deployment



❖ Model Deployment

Configure your deployment

You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. Read more.

Arduino library ×

SELECTED DEPLOYMENT

Arduino library

An Arduino library with examples that runs on most Arm-based Arduino development boards.

❖ Model Deployment

MODEL OPTIMIZATIONS

Model optimizations can increase on-device performance but may reduce accuracy.

EON™ Compiler

Same accuracy, 40% less RAM, 49% less ROM.

Quantized (int8)

Selected ✔

	MFCC	CLASSIFIER	TOTAL
LATENCY	294 ms.	8 ms.	**302 ms.**
RAM	13.0K	3.8K	**13.0K**
FLASH	-	31.8K	-
ACCURACY			**95.43%**

Unoptimized (float32)

Select

	MFCC	CLASSIFIER	TOTAL
LATENCY	294 ms.	133 ms.	**427 ms.**
RAM	13.0K	7.0K	**13.0K**
FLASH	-	28.0K	-
ACCURACY			**95.43%**

Estimate for Arduino Nano 33 BLE Sense (Cortex-M4F 64MHz) .

Build

MODEL DEPLOYMENT HARDWARE

- Model Deployment

Built Arduino library

Add this library through the Arduino IDE via:

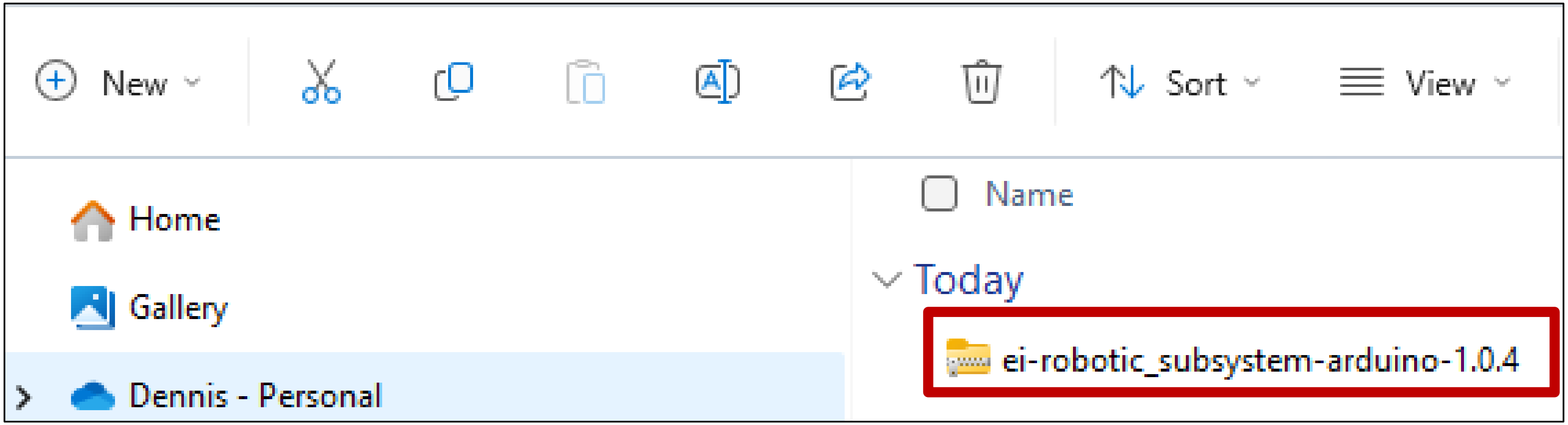
`Sketch > Include Library > Add .ZIP Library...`

Examples can then be found under:

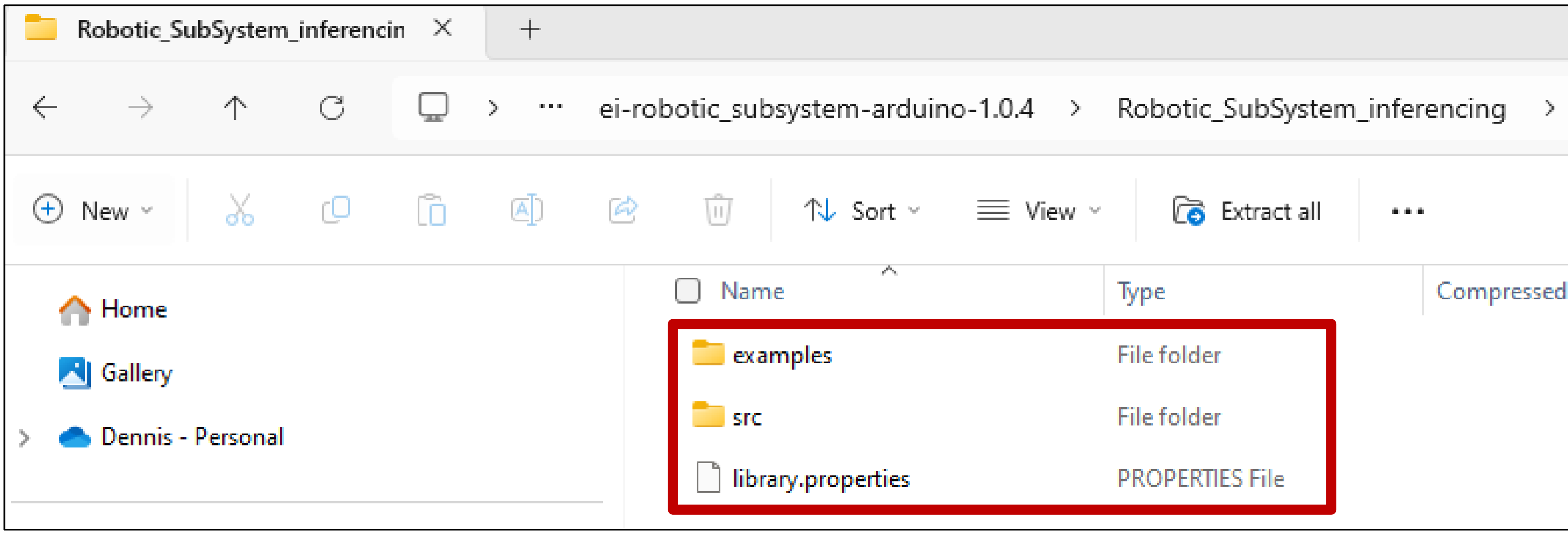
`File > Examples > Robotic_SubSystem_inferencing`

MODEL DEPLOYMENT HARDWARE

❖ Generated files

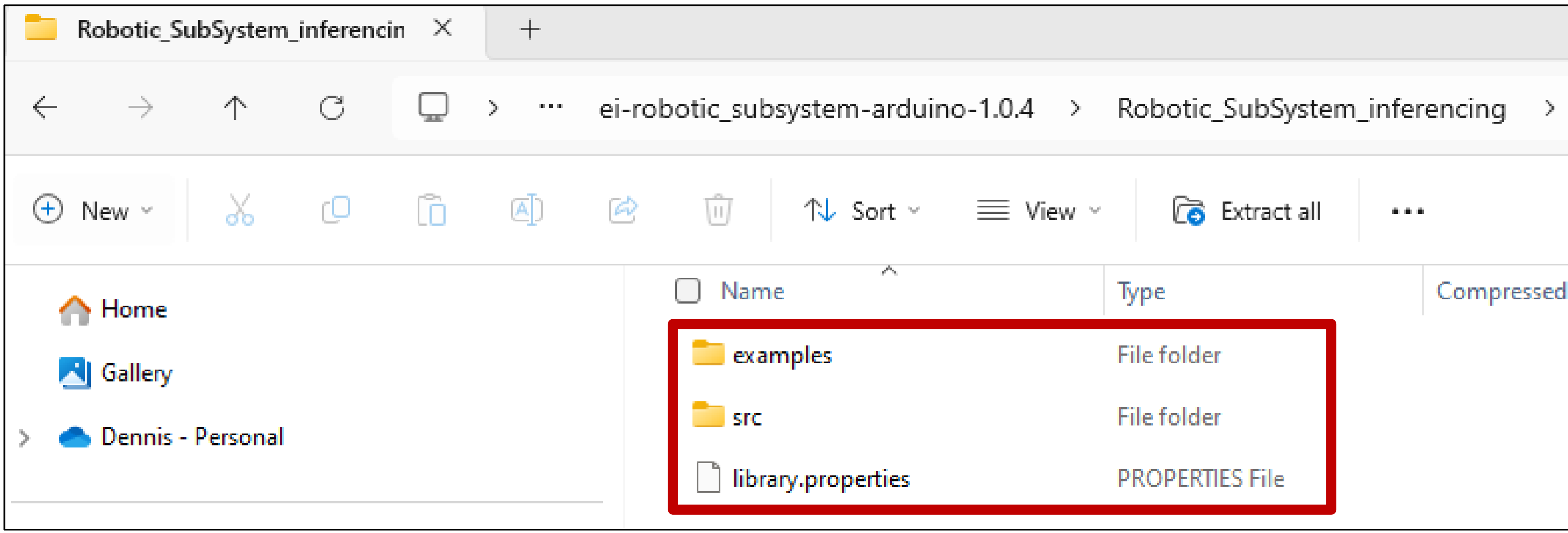


❖ Generated files



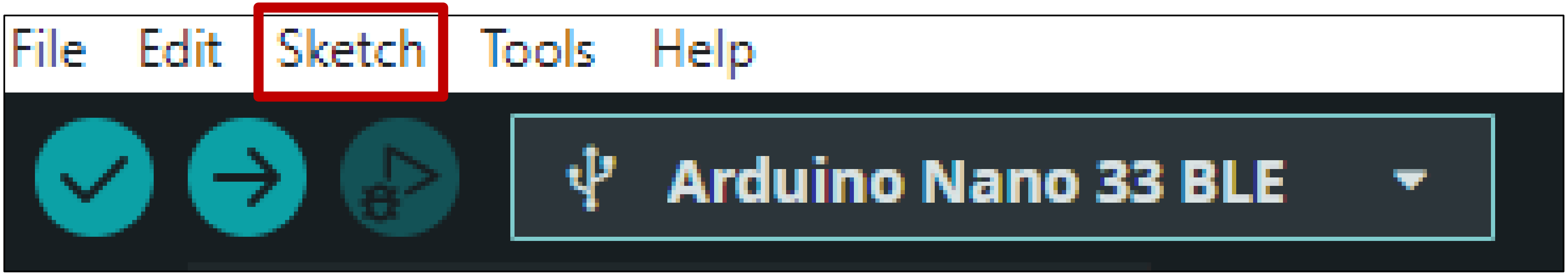
MODEL DEPLOYMENT HARDWARE

❖ Generated files



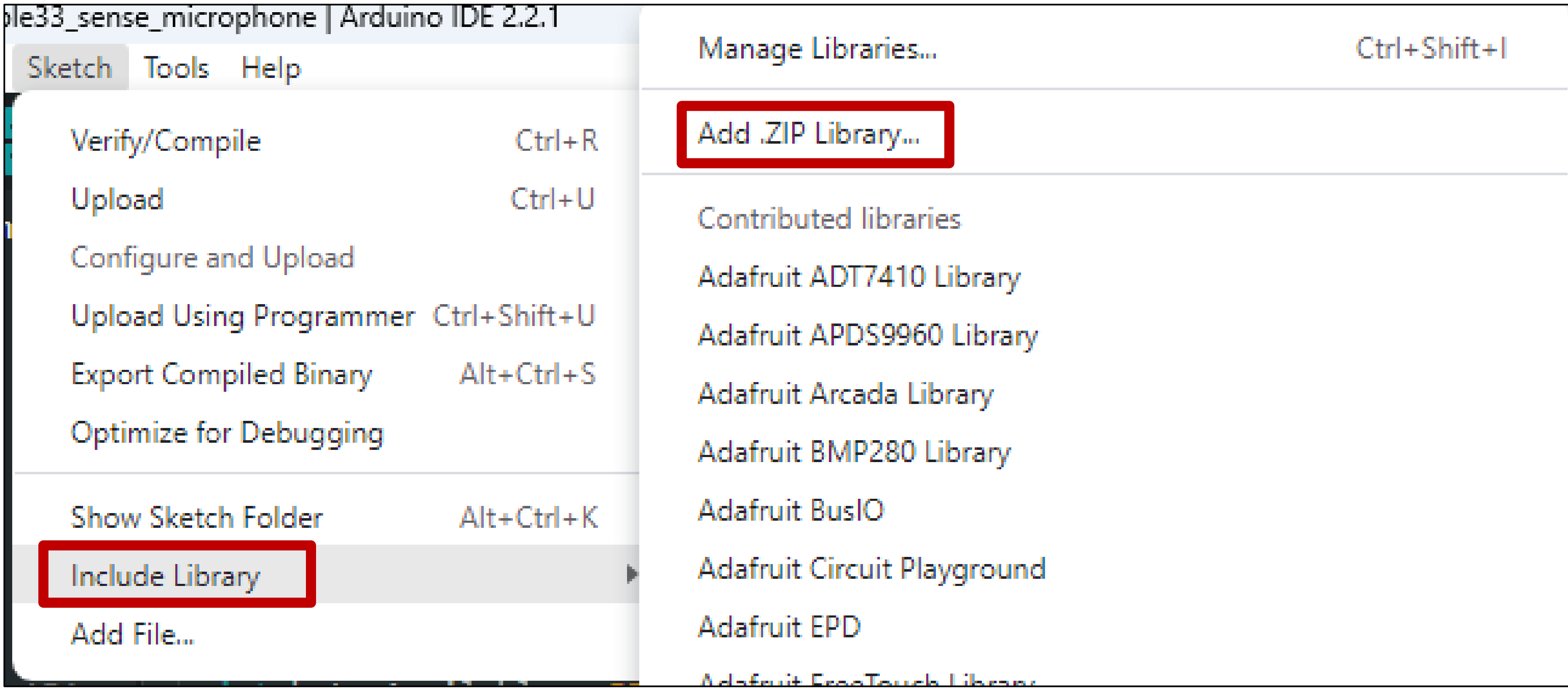
MODEL DEPLOYMENT HARDWARE

❖ Arduino

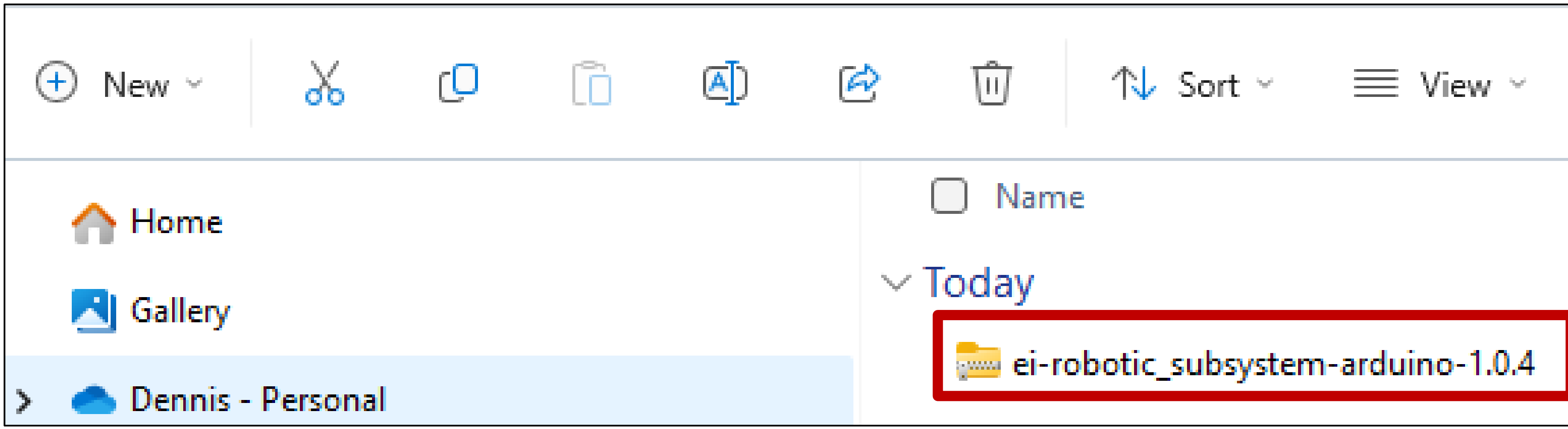


MODEL DEPLOYMENT HARDWARE

- Arduino

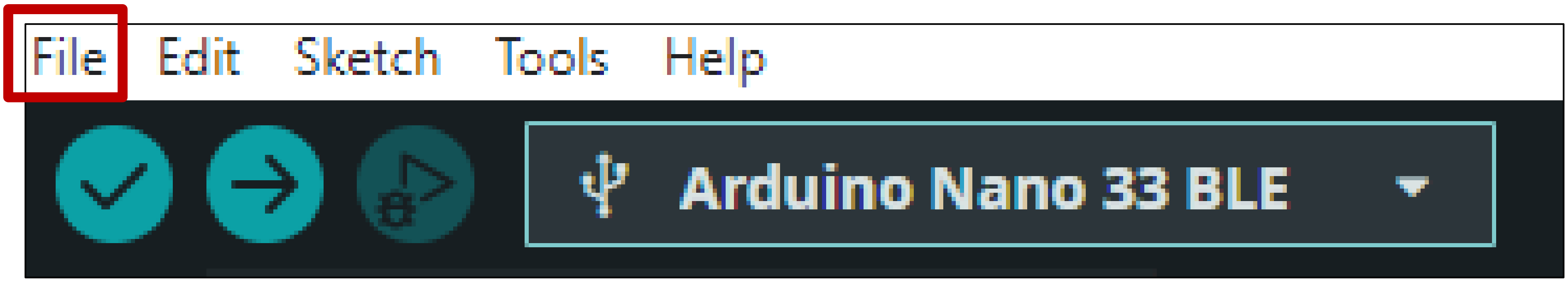


- Arduino

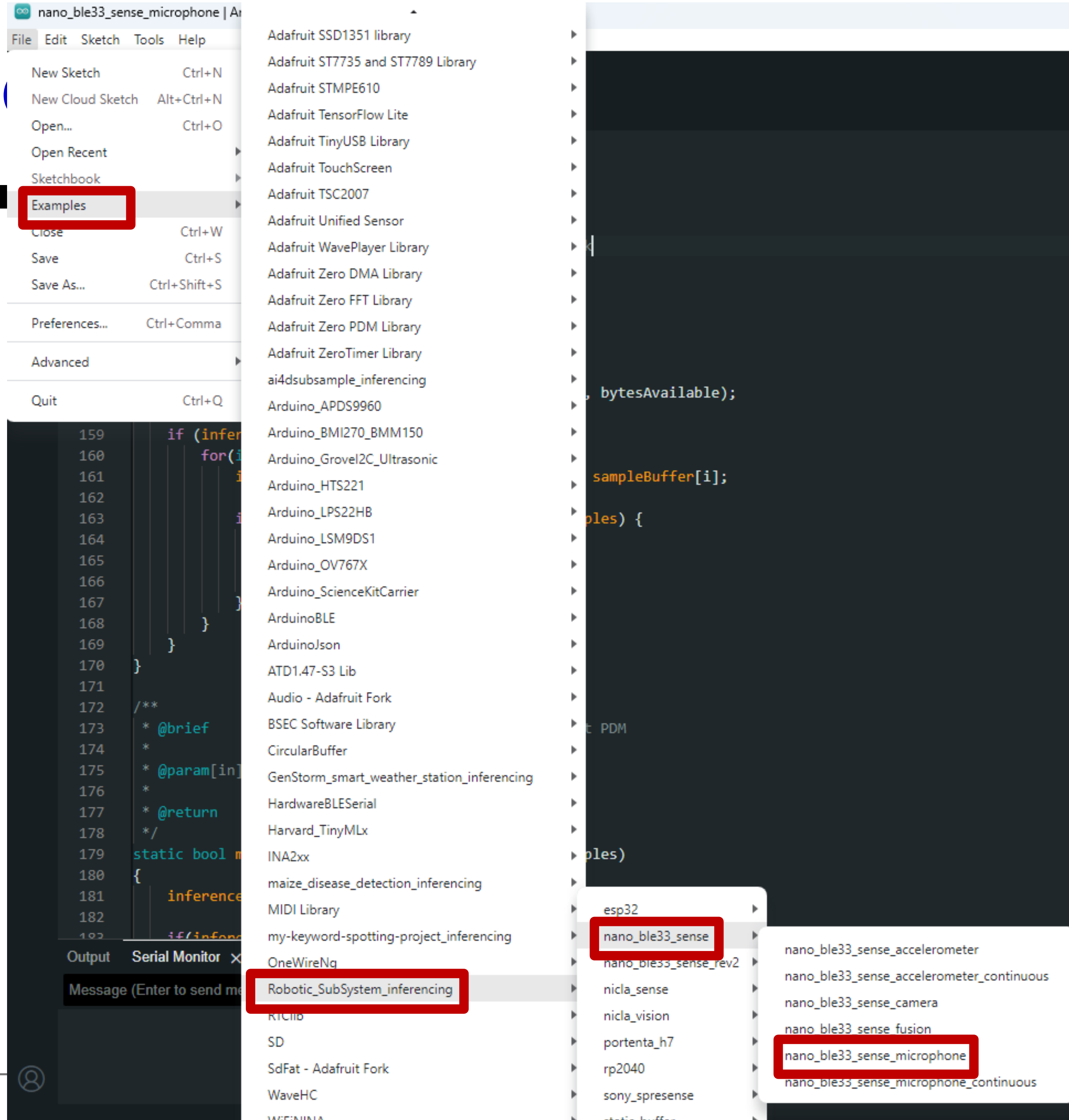

New
Sort
View
Name
Home
Gallery
Today
Dennis - Personal
ei-robotic_subsystem-arduino-1.0.4

MODEL DEPLOYMENT HARDWARE

❖ Arduino



Arduino


nano_ble33_sense_microphone | Ar
File
Edit
Sketch
Tools
Help
New Sketch Ctrl+N
New Cloud Sketch Alt+Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Preferences... Ctrl+Comma
Advanced
Quit Ctrl+Q
Adafruit SSD1351 library
Adafruit ST7735 and ST7789 Library
Adafruit STMPE610
Adafruit TensorFlow Lite
Adafruit TinyUSB Library
Adafruit TouchScreen
Adafruit TSC2007
Adafruit Unified Sensor
Adafruit WavePlayer Library
Adafruit Zero DMA Library
Adafruit Zero FFT Library
Adafruit Zero PDM Library
Adafruit ZeroTimer Library
ai4dsubsample_inferencing
Arduino_APDS9960
Arduino_BMI270_BMM150
Arduino_GroveI2C_Ultrasonic
Arduino_HTS221
Arduino_LPS22HB
Arduino_LSM9DS1
Arduino_OV767X
Arduino_ScienceKitCarrier
ArduinoBLE
ArduinoJson
ATD1.47-S3 Lib
Audio - Adafruit Fork
BSEC Software Library
CircularBuffer
GenStorm_smart_weather_station_inferencing
HardwareBLESerial
Harvard_TinyMLx
INA2xx
maize_disease_detection_inferencing
MIDI Library
my-keyword-spotting-project_inferencing
OneWireNg
Robotic_SubSystem_inferencing
SD
SdFat - Adafruit Fork
WaveHC
WiFiNINA
esp32
nano_ble33_sense
nicla_sense
nicla_vision
portenta_h7
rp2040
sony_spresense
static_buffer
nano_ble33_sense_accelerometer
nano_ble33_sense_accelerometer_continuous
nano_ble33_sense_camera
nano_ble33_sense_fusion
nano_ble33_sense_microphone
nano_ble33_sense_microphone_continuous
Output
Serial Monitor
Message (Enter to send me
, bytesAvailable);
sampleBuffer[i];
ples) {
ples)

❖ Arduino code for spotting keywords (1)

```c
// If your target is limited in memory remove this macro to save 10K RAM
#define EIDSP_QUANTIZE_FILTERBANK   0

/*
 ** NOTE: If you run into TFLite arena allocation issue.
 **
 ** This may be due to may dynamic memory fragmentation.
 ** Try defining "-DEI_CLASSIFIER_ALLOCATION_STATIC" in boards.local.txt (create
 ** if it doesn't exist) and copy this file to
 ** `<ARDUINO_CORE_INSTALL_PATH>/arduino/hardware/<mbed_core>/<core_version>/`.
 **
 ** See
 ** (https://support.arduino.cc/hc/en-us/articles/360012076960-Where-are-the-installed-cores-located-)
 ** to find where Arduino installs cores on your machine.
 **
 ** If the problem persists then there's not enough memory for this model and application.
 */

/* Includes ---------------------------------------------------------------- */
#include <PDM.h>
#include <Robotic_SubSystem_inferencing.h>

/** Audio buffers, pointers and selectors */
typedef struct {
    int16_t *buffer;
    uint8_t buf_ready;
    uint32_t buf_count;
    uint32_t n_samples;
} inference_t;

static inference_t inference;
static signed short sampleBuffer[2048];
static bool debug_nn = false; // Set this to true to see e.g. features generated from the raw signal

```

```c
/**
 * @brief      Arduino setup function
 */
void setup()
{
    // put your setup code here, to run once:
    Serial.begin(115200);
    // comment out the below line to cancel the wait for USB connection (needed for native USB)
    while (!Serial);
    Serial.println("Edge Impulse Inferencing Demo");

    // summary of inferencing settings (from model_metadata.h)
    ei_printf("Inferencing settings:\n");
    ei_printf("\tInterval: %.2f ms.\n", (float)EI_CLASSIFIER_INTERVAL_MS);
    ei_printf("\tFrame size: %d\n", EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE);
    ei_printf("\tSample length: %d ms.\n", EI_CLASSIFIER_RAW_SAMPLE_COUNT / 16);
    ei_printf("\tNo. of classes: %d\n", sizeof(ei_classifier_inferencing_categories) / sizeof(ei_classifier_inferencing_categories[0]));

    if (microphone_inference_start(EI_CLASSIFIER_RAW_SAMPLE_COUNT) == false) {
        ei_printf("ERR: Could not allocate audio buffer (size %d), this could be due to the window length of your model\r\n", EI_CLASSIFIER_RAW_SAMPLE_COUNT);
        return;
    }
}

```

MODEL DEPLOYMENT HARDWARE

❖ Arduino code for spotting keywords (2)

nano_ble33_sense_microphone.ino

```
 */
void loop()
{
    ei_printf("Starting inferencing in 2 seconds...\n");

    delay(2000);

    ei_printf("Recording...\n");

    bool m = microphone_inference_record();
    if (!m) {
        ei_printf("ERR: Failed to record audio...\n");
        return;
    }

    ei_printf("Recording done\n");

    signal_t signal;
    signal.total_length = EI_CLASSIFIER_RAW_SAMPLE_COUNT;
    signal.get_data = &microphone_audio_signal_get_data;
    ei_impulse_result_t result = { 0 };

    EI_IMPULSE_ERROR r = run_classifier(&signal, &result, debug_nn);
    if (r != EI_IMPULSE_OK) {
        ei_printf("ERR: Failed to run classifier (%d)\n", r);
        return;
    }

    // print the predictions
    ei_printf("Predictions ");
    ei_printf("(DSP: %d ms., Classification: %d ms., Anomaly: %d ms.)",
        result.timing.dsp, result.timing.classification, result.timing.anomaly);
    ei_printf(": \n");
    for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
        ei_printf("    %s: %.5f\n", result.classification[ix].label, result.classification[ix].value);
    }
#if EI_CLASSIFIER_HAS_ANOMALY == 1
    ei_printf("    anomaly score: %.3f\n", result.anomaly);
#endif
}
```

```
/**
 * @brief      Init inferencing struct and setup/start PDM
 *
 * @param[in]  n_samples  The n samples
 *
 * @return     { description_of_the_return_value }
 */
static bool microphone_inference_start(uint32_t n_samples)
{
    inference.buffer = (int16_t *)malloc(n_samples * sizeof(int16_t));

    if(inference.buffer == NULL) {
        return false;
    }

    inference.buf_count  = 0;
    inference.n_samples  = n_samples;
    inference.buf_ready  = 0;

    // configure the data receive callback
    PDM.onReceive(&pdm_data_ready_inference_callback);

    PDM.setBufferSize(4096);

    // initialize PDM with:
    // - one channel (mono mode)
    // - a 16 kHz sample rate
    if (!PDM.begin(1, EI_CLASSIFIER_FREQUENCY)) {
        ei_printf("Failed to start PDM!");
        microphone_inference_end();

        return false;
    }

    // set the gain, defaults to 20
    PDM.setGain(127);

    return true;
}
```

MODEL DEPLOYMENT HARDWARE

❖ Arduino code for spotting keywords (3)

```
/**
 * @brief      Wait on new data
 *
 * @return     True when finished
 */
static bool microphone_inference_record(void)
{
    inference.buf_ready = 0;
    inference.buf_count = 0;

    while(inference.buf_ready == 0) {
        delay(10);
    }

    return true;
}

/**
 * Get raw audio signal data
 */
static int microphone_audio_signal_get_data(size_t offset, size_t length, float *out_ptr)
{
    numpy::int16_to_float(&inference.buffer[offset], out_ptr, length);

    return 0;
}

/**
 * @brief      Stop PDM and release buffers
 */
static void microphone_inference_end(void)
{
    PDM.end();
    free(inference.buffer);
}

#if !defined(EI_CLASSIFIER_SENSOR) || EI_CLASSIFIER_SENSOR != EI_CLASSIFIER_SENSOR_MICROPHONE
#error "Invalid model for current sensor."
#endif
```

MODEL DEPLOYMENT HARDWARE

- Arduino code for spotting keywords (Explanation)

  - The code is for setting up and running an inferencing demo using the Edge Impulse SDK on an Arduino platform

  - It processes audio data, records it using a PDM (Pulse Density Modulation) microphone, and then classifies the data using a pre-trained machine learning model

MODEL DEPLOYMENT HARDWARE

- Arduino code for spotting keywords (Explanation)
  - Preprocessor Directives and Includes

```
#define EIDSP_QUANTIZE_FILTERBANK 0


#include <PDM.h>
#include <Robotic_SubSystem_inferencing.h>
```

- **EIDSP_QUANTIZE_FILTERBANK**: Defines a macro to control memory usage
- Includes necessary libraries for PDM microphone handling and inferencing

MODEL DEPLOYMENT HARDWARE

- Arduino code for spotting keywords (Explanation)
  - Data Structures and Global Variables

```
typedef struct {
    int16_t *buffer;
    uint8_t buf_ready;
    uint32_t buf_count;
    uint32_t n_samples;
} inference_t;

static inference_t inference;
static signed short sampleBuffer[2048];
static bool debug_nn = false;
```

- **inference_t**: Defines a structure to manage audio buffer data
- **inference**: An instance of the structure
- **sampleBuffer**: A buffer to store audio samples
- **debug_nn**: A flag to enable debugging information

MODEL DEPLOYMENT HARDWARE

❖ Arduino code for spotting keywords (Explanation)

☐ Setup Function

```
void setup() {
    Serial.begin(115200);
    while (!Serial);
    Serial.println("Edge Impulse Inferencing Demo");

    ei_printf("Inferencing settings:\n");
    ei_printf("\tInterval: %.2f ms.\n", (float)EI_CLASSIFIER_INTERVAL_MS);
    ei_printf("\tFrame size: %d\n", EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE);
    ei_printf("\tSample length: %d ms.\n", EI_CLASSIFIER_RAW_SAMPLE_COUNT / 16);
    ei_printf("\tNo. of classes: %d\n", sizeof(ei_classifier_inferencing_categories) / siz

    if (microphone_inference_start(EI_CLASSIFIER_RAW_SAMPLE_COUNT) == false) {
        ei_printf("ERR: Could not allocate audio buffer (size %d)\n", EI_CLASSIFIER_RAW_SA
        return;
    }
}
```

- Initializes the serial connection and waits for it to be ready
- Prints inferencing settings from the model metadata
- Starts the microphone inferencing with a specified number of samples

MODEL DEPLOYMENT HARDWARE

- Arduino code for spotting keywords (Explanation)
  - Loop Function
    - Waits for 2 seconds and starts recording audio
    - Runs the classifier on the recorded audio and prints the predictions

```
void loop() {
    ei_printf("Starting inferencing in 2 seconds...\n");
    delay(2000);
    ei_printf("Recording...\n");

    bool m = microphone_inference_record();
    if (!m) {
        ei_printf("ERR: Failed to record audio...\n");
        return;
    }
    ei_printf("Recording done\n");

    signal_t signal;
    signal.total_length = EI_CLASSIFIER_RAW_SAMPLE_COUNT;
    signal.get_data = &microphone_audio_signal_get_data;
    ei_impulse_result_t result = { 0 };

    EI_IMPULSE_ERROR r = run_classifier(&signal, &result, debug_nn);
    if (r != EI_IMPULSE_OK) {
        ei_printf("ERR: Failed to run classifier (%d)\n", r);
        return;
    }

    ei_printf("Predictions ");
    ei_printf("(DSP: %d ms., Classification: %d ms., Anomaly: %d ms.)",
        result.timing.dsp, result.timing.classification, result.timing.anomaly);
    ei_printf(": \n");
    for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
        ei_printf("    %s: %.5f\n", resu  classification[ix].label, result.classification
    }
```

MODEL DEPLOYMEN

❖ Arduino code for spotting keywords (Explanation)

- PDM Callback and Helper Functions
  - **pdm_data_ready_inference_callback**: Called when PDM data is ready, fills the inference buffer with audio samples
  - **microphone_inference_start**: Initializes the PDM microphone and sets up the buffer
  - **microphone_inference_record**: Waits for the buffer to be filled with recorded data
  - **microphone_audio_signal_get_data**: Converts audio data to a format suitable for inferencing
  - **microphone_inference_end**: Stops the PDM microphone and frees allocated memory

```
static void pdm_data_ready_inference_callback(void) {
    int bytesAvailable = PDM.available();
    int bytesRead = PDM.read((char *)&sampleBuffer[0], bytesAvailable);

    if (inference.buf_ready == 0) {
        for(int i = 0; i < bytesRead>>1; i++) {
            inference.buffer[inference.buf_count++] = sampleBuffer[i];
            if(inference.buf_count >= inference.n_samples) {
                inference.buf_count = 0;
                inference.buf_ready = 1;
                break;
            }
        }
    }
}
```

```
static bool microphone_inference_start(uint32_t n_samples) {
    inference.buffer = (int16_t *)malloc(n_samples * sizeof(int16_t));
    if(inference.buffer == NULL) {
        return false;
    }

    inference.buf_count  = 0;
    inference.n_samples  = n_samples;
    inference.buf_ready  = 0;

    PDM.onReceive(&pdm_data_ready_inference_callback);
    PDM.setBufferSize(4096);

    if (!PDM.begin(1, EI_CLASSIFIER_FREQUENCY)) {
        ei_printf("Failed to start PDM!");
        microphone_inference_end();
        return false;
    }

    PDM.setGain(127);
    return true;
}
```

```
static bool microphone_inference_record(void) {
    inference.buf_ready = 0;
    inference.buf_count = 0;
    while(inference.buf_ready == 0) {
        delay(10);
    }
    return true;
}

static int microphone_audio_signal_get_data(size_t offset, size_t length, float *out_ptr)
    numpy::int16_to_float(&inference.buffer[offset], out_ptr, length);
    return 0;
}

static void microphone_inference_end(void) {
    PDM.end();
    free(inference.buffer);
}
```

- Arduino code for spotting keywords (Explanation)
  - Error Handling

```
#if !defined(EI_CLASSIFIER_SENSOR) || EI_CLASSIFIER_SENSOR != EI_CLASSIFIER_SENSOR_MICROPH
#error "Invalid model for current sensor."
#endif
```

- Ensures that the correct sensor type is defined for the model being used

MODEL DEPLOYMENT HARDWARE

- Arduino code for spotting keywords (Explanation)
  - The code sets up a PDM microphone on an Arduino, records audio, and processes it using a machine learning model provided by Edge Impulse, printing out classification results

MODEL DEPLOYMENT HARDWARE

- Arduino code for spotting keywords
  - Inferencing results

```
    Silence: 0.24609
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 145 ms., Classification: 6 ms., Anomaly: 0 ms.):
    Backward: 0.94141
    Forward: 0.04297
    Silence: 0.01172
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 145 ms., Classification: 6 ms., Anomaly: 0 ms.):
    Backward: 0.99609
    Forward: 0.00000
    Silence: 0.00000
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 145 ms., Classification: 6 ms., Anomaly: 0 ms.):
    Backward: 0.82031
    Forward: 0.06250
    Silence: 0.11719
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 145 ms., Classification: 6 ms., Anomaly: 0 ms.):
    Backward: 0.80859
    Forward: 0.16797
    Silence: 0.02344
Starting inferencing in 2 seconds...
```

MODEL DEPLOYMENT HARDWARE

- Arduino code for controlling the stepper motor using keyword spotting
  - Added libraries

```
#include <PDM.h>
#include <Robotic_SubSystem_inferencing.h>

//ADDED CODE
//Includes the Arduino Stepper Library
#include <Stepper.h>
// Defines the number of steps per rotation
const int stepsPerRevolution = 2048;
// Creates an instance of stepper class
// Pins entered in sequence IN1-IN3-IN2-IN4 for proper step sequence
Stepper myStepper = Stepper(stepsPerRevolution, 8, 10, 9, 11);
//END ADDED CODE
```

MODEL DEPLOYMENT HARDWARE

- Arduino code for controlling the stepper motor using keyword spotting
  - Added codes

```
//ADDED CODE
if(result.classification[0].value > 0.9){
  ei_printf("BACKWARD");
  // Rotate CW at 10 RPM
  myStepper.setSpeed(10);
  myStepper.step(stepsPerRevolution);
  delay(1000);
}
else if(result.classification[1].value > 0.9){
  ei_printf("FORWARD");
  // Rotate CCW at 10 RPM
  myStepper.setSpeed(10);
  myStepper.step(-stepsPerRevolution);
  delay(1000);
}
```

MODEL DEPLOYMENT HARDWARE

- Arduino code for controlling the stepper motor using keyword spotting
  - Inferencing results

```
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 144 ms., Classification: 6 ms., Anomaly: 0 ms.):
    Backward: 0.19531
    Forward: 0.28125
    Silence: 0.51953
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 144 ms., Classification: 6 ms., Anomaly: 0 ms.):
    Backward: 0.10938
    Forward: 0.47266
    Silence: 0.41797
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 144 ms., Classification: 6 ms., Anomaly: 0 ms.):
    Backward: 0.63281
    Forward: 0.30469
    Silence: 0.06250
```