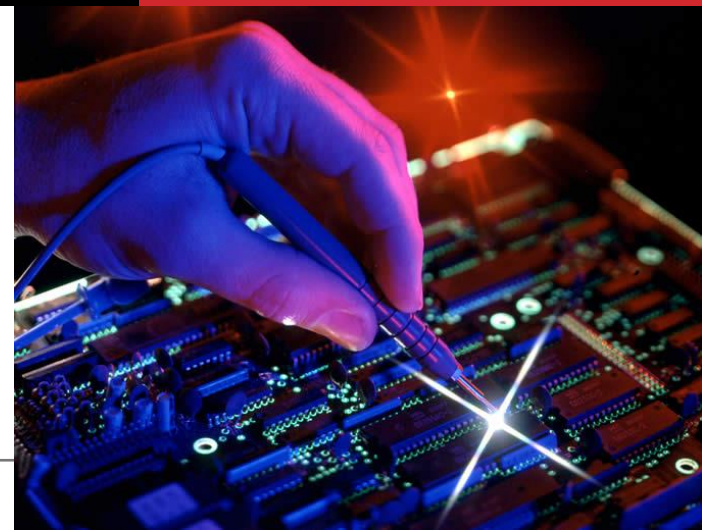
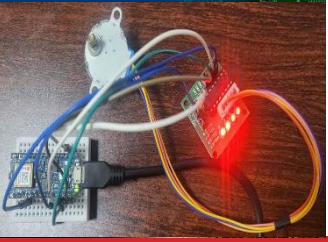


Robotics

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

Dennis A. N. Gookyi





CONTENTS

❖ Project Overview

❖ Project Hardware

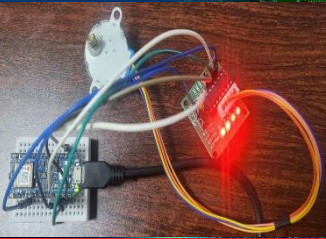
- ❑ Arduino Nano 33 BLE Sense
- ❑ 28BYJ-48 Stepper Motor
- ❑ ULN2003 Driver Board

❖ Arduino Code – Using Built-in Stepper Library

❖ Speech Commands Dataset

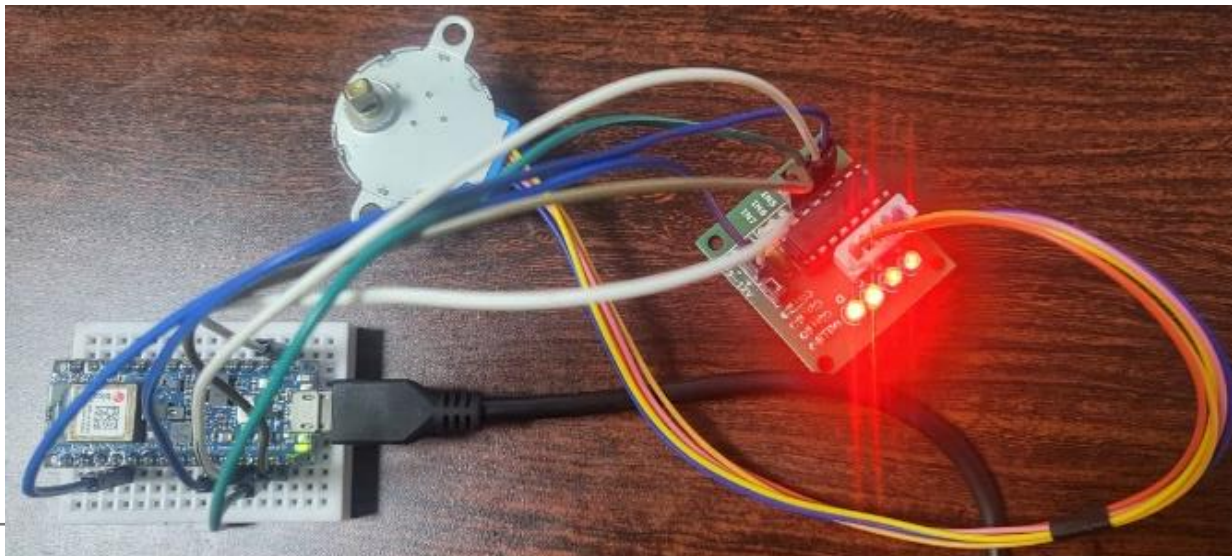
❖ Machine Learning Model Training With Edge Impulse

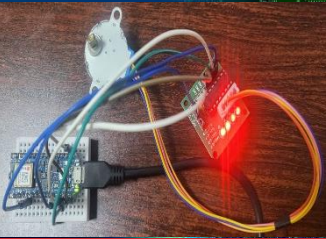




PROJECT OVERVIEW

- ❖ 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

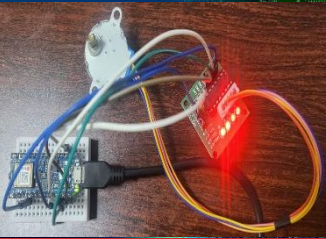




PROJECT OVERVIEW

- ❖ 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

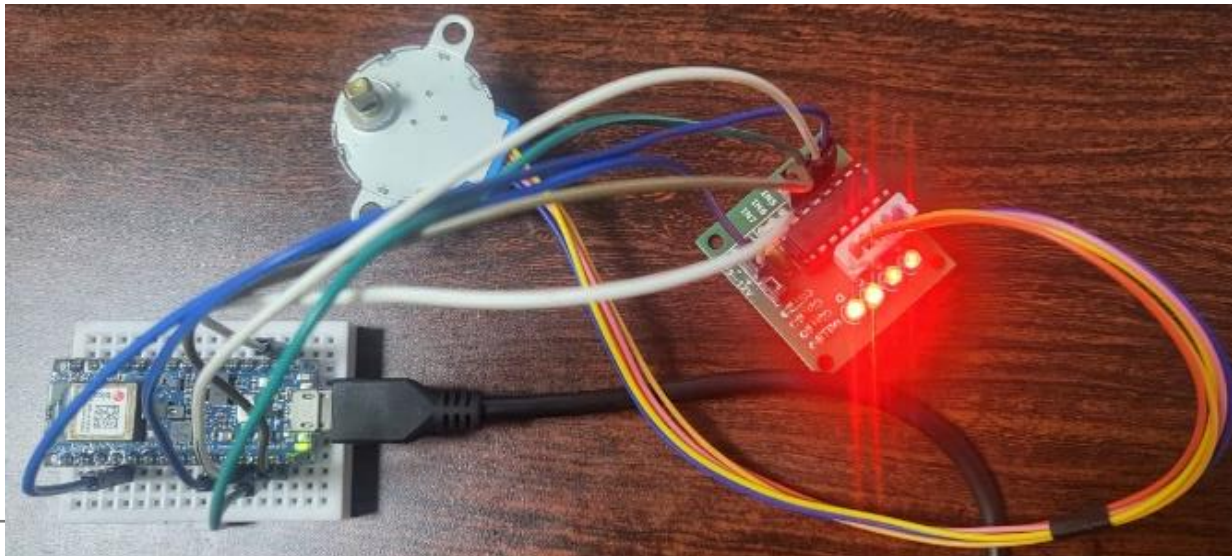


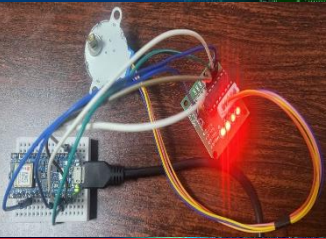


PROJECT HARDWARE

❖ The hardware components for the project include:

- Arduino Nano 33 BLE Sense
- 28BYJ-48 Stepper Motor
- ULN2003 Driver

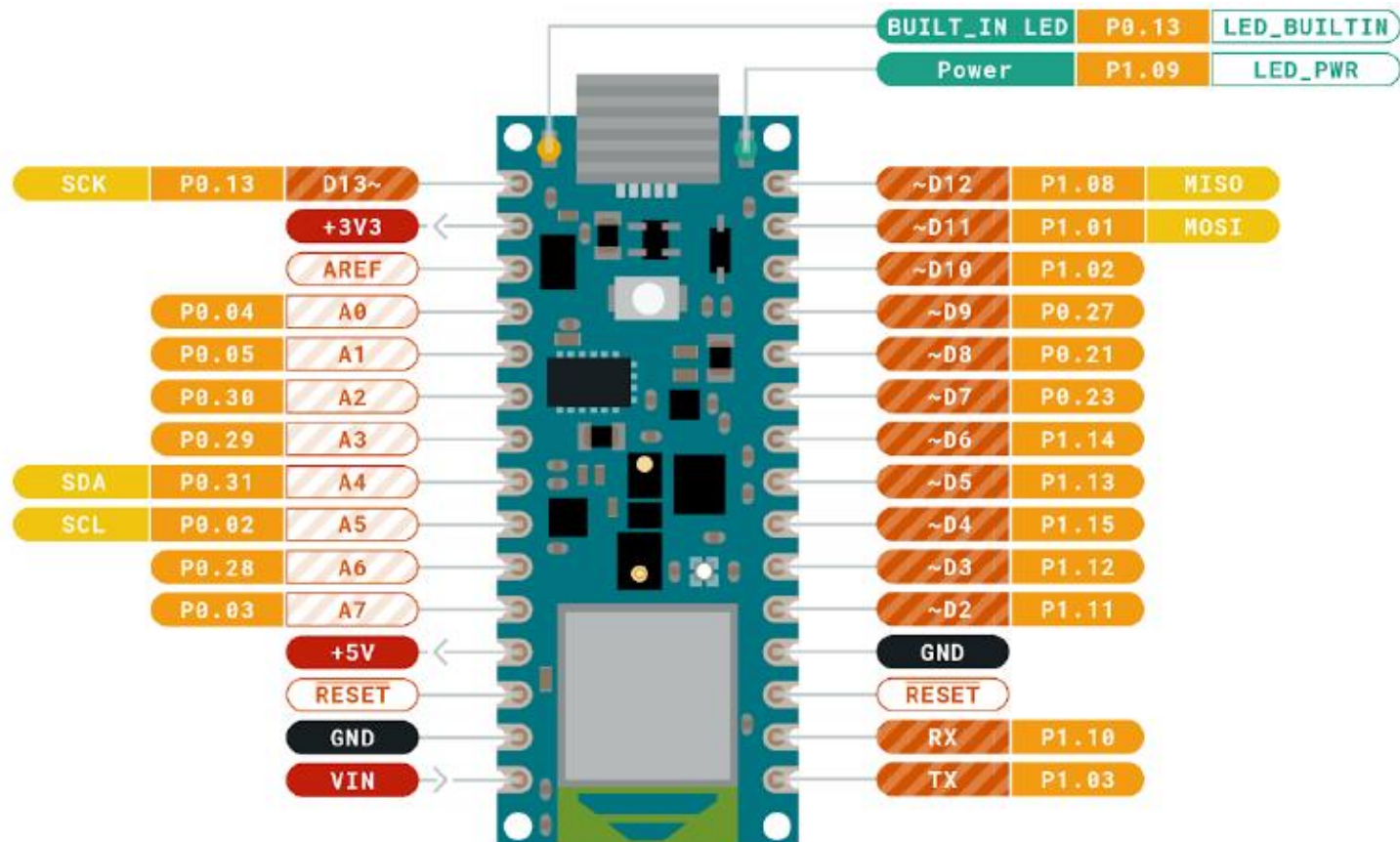


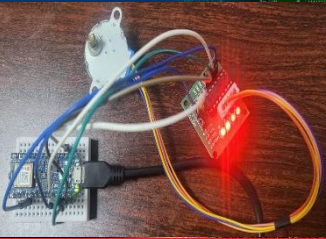


PROJECT HARDWARE – NANO 33 BLE SENSE

❖ Arduino Nano 33 BLE Sense

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

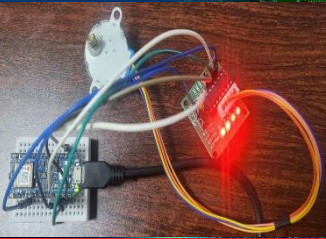




PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

- ❖ 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

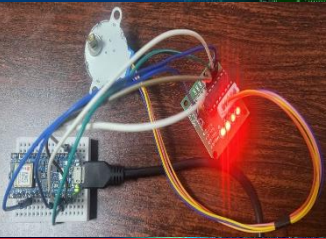




PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

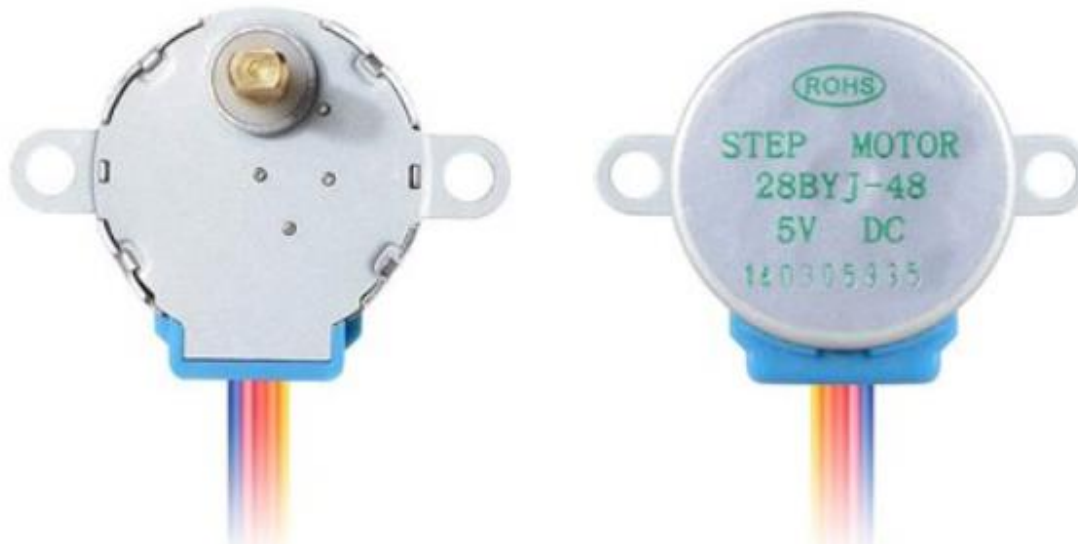
- ❖ The 28BYJ-48 is a 5-wire unipolar stepper motor that runs on 5V
- ❖ It is perfect for projects requiring precise positioning, like vent opening and closing

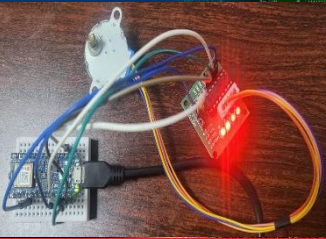




PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

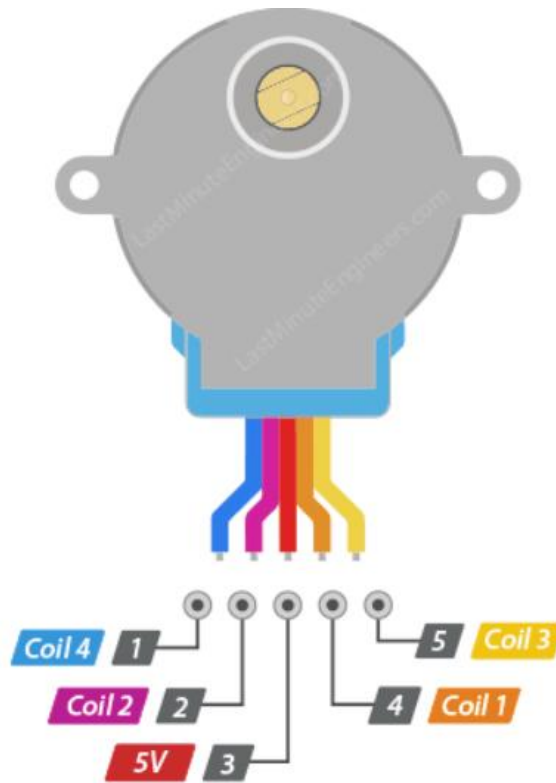
- ❖ Despite its small size, the motor delivers a decent torque of 34.3 mN.m (millinewton-meters) at a speed of around 15 RPM
- ❖ It provides good torque even at a standstill and maintains it as long as the motor receives power
- ❖ The only drawback is that it is somewhat power-hungry and consumes energy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

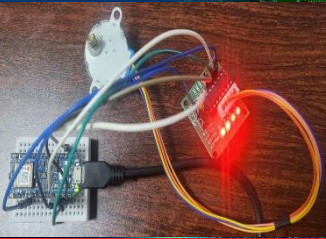




PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

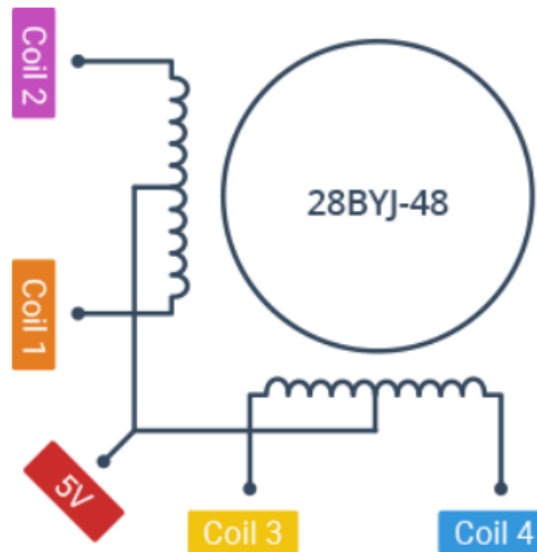
- ❖ The motor has five wires
- ❖ The 28BYJ-48 has two coils, each of which has a center tap
- ❖ These two center taps are connected internally and brought out as the 5th wire (red wire)

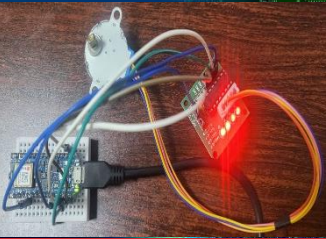




PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

- ❖ 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

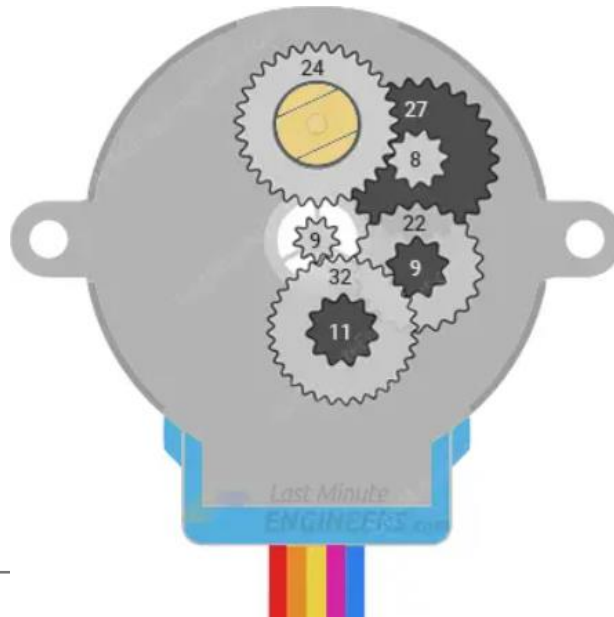




PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

❖ 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 ($360^\circ / 11.25^\circ = 32$)
- In addition, the gearbox inside the motor has a 64:1 gear reduction
 - This results in 2048 (32×64) steps per revolution



Gear Ratios:

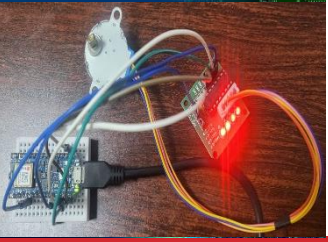
- $32 / 9$
- $22 / 11$
- $27 / 9$
- $24 / 8$

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



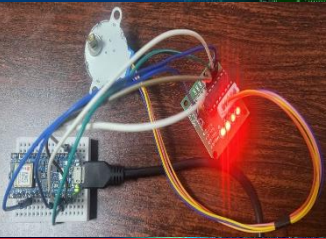


PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

❖ 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



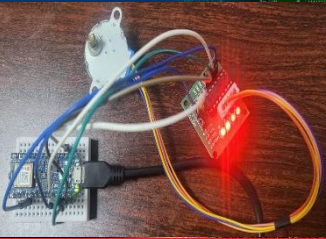


PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

❖ Technical Specifications

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

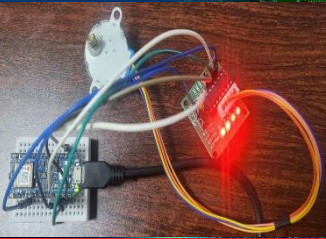




PROJECT HARDWARE – ULN2003 D RIVER BOARD

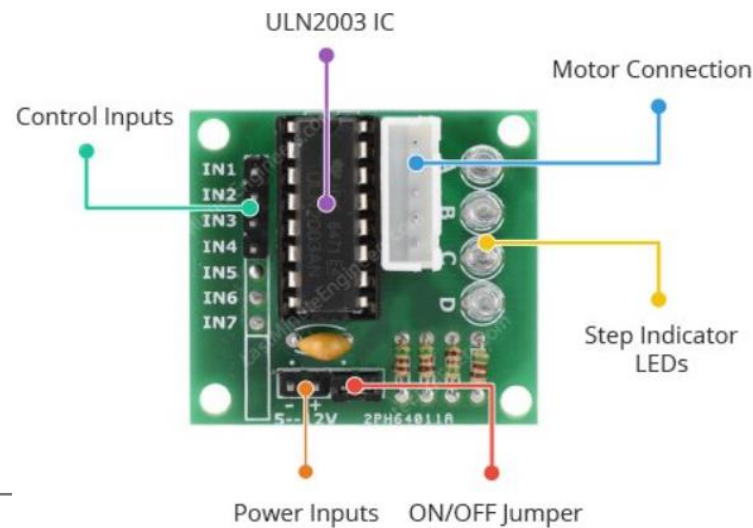
- ❖ Because the 28BYJ-48 stepper motor consumes a significant amount of power, it cannot be controlled directly by a microcontroller such as Arduino
- ❖ To control the motor, a driver IC such as the ULN2003 is required
 - Therefore, this motor typically comes with a ULN2003-based driver board

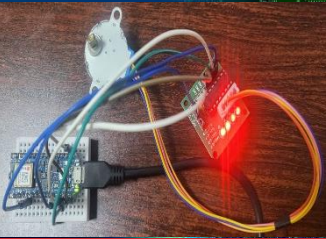




PROJECT HARDWARE – ULN2003 D RIVER BOARD

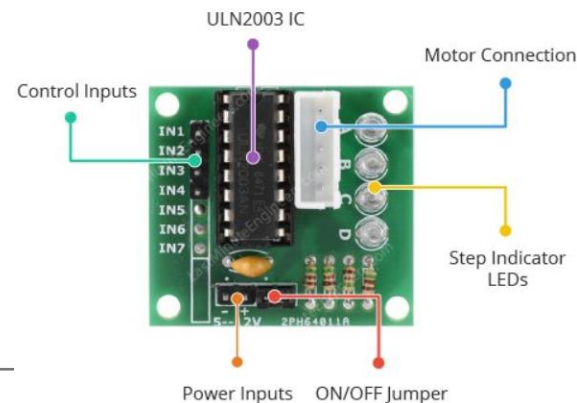
- ❖ The ULN2003, known for its high current and high voltage capability, provides a higher current gain than a single transistor and allows a microcontroller's low voltage low current output to drive a high current stepper motor
- ❖ The ULN2003 consists of an array of seven Darlington transistor pairs,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

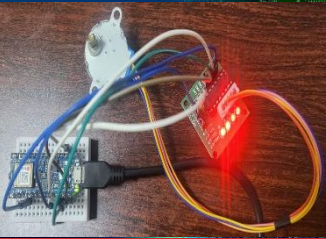




PROJECT HARDWARE – ULN2003 D RIVER BOARD

- ❖ 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

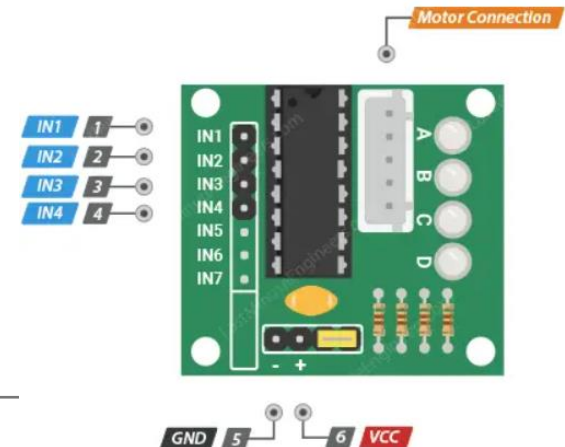


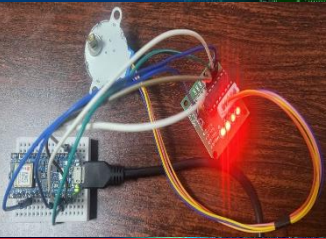


PROJECT HARDWARE – ULN2003 D RIVER BOARD

❖ 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

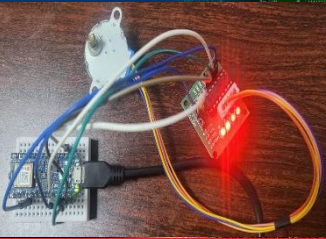




PROJECT HARDWARE






- ❖ 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

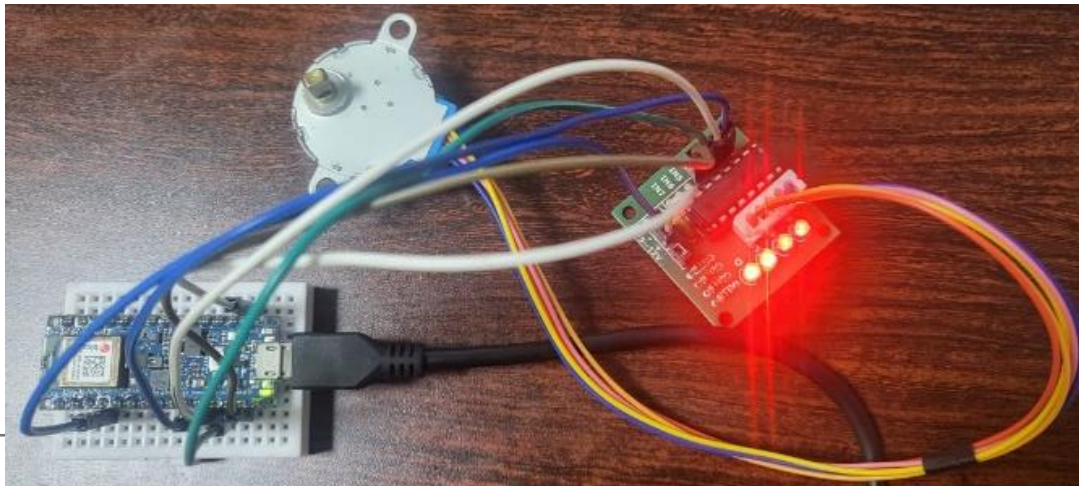


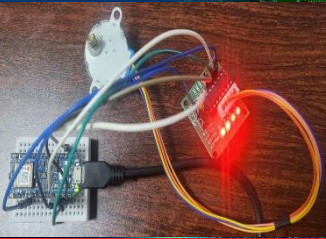


PROJECT HARDWARE

❖ The following table lists the pin connections:

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

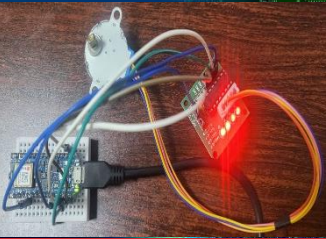




ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

- ❖ 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



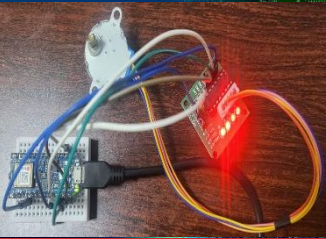


ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

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

```
1 //Includes the Arduino Stepper Library
2 #include <Stepper.h>
3
4 // Defines the number of steps per rotation
5 const int stepsPerRevolution = 2048;
6
7 // Creates an instance of stepper class
8 // Pins entered in sequence IN1-IN3-IN2-IN4 for proper step sequence
9 Stepper myStepper = Stepper(stepsPerRevolution, 8, 10, 9, 11);
10
11 void setup() {
12     // Nothing to do (Stepper Library sets pins as outputs)
13 }
14
15 void loop() {
16     // Rotate CW slowly at 5 RPM
17     myStepper.setSpeed(5);
18     myStepper.step(stepsPerRevolution);
19     delay(1000);
20
21     // Rotate CCW quickly at 10 RPM
22     myStepper.setSpeed(10);
23     myStepper.step(-stepsPerRevolution);
24     delay(1000);
25 }
```





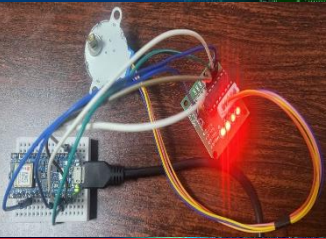
ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

❖ Code Explanation:

- The sketch starts by including the built-in stepper library

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





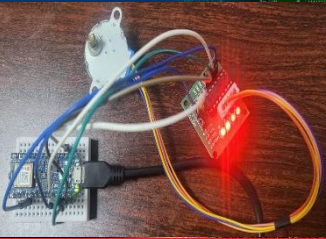
ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

❖ 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

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





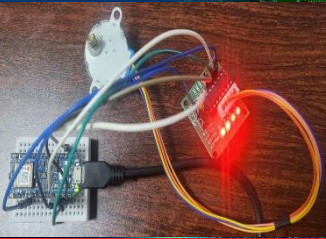
ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

❖ 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

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





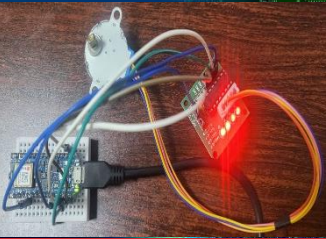
ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

❖ 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

```
10
11 void setup() {
12     // Nothing to do (Stepper Library sets pins as outputs)
13 }
```





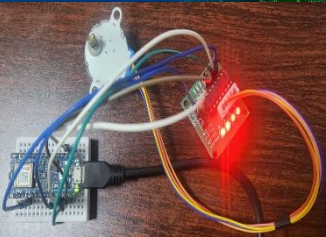
ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

❖ 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

```
15 void loop() {  
16     // Rotate CW slowly at 5 RPM  
17     myStepper.setSpeed(5);  
18     myStepper.step(stepsPerRevolution);  
19     delay(1000);  
20  
21     // Rotate CCW quickly at 10 RPM  
22     myStepper.setSpeed(10);  
23     myStepper.step(-stepsPerRevolution);  
24     delay(1000);  
25 }
```





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>



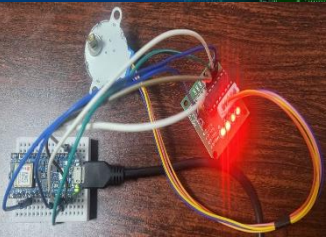


SPEECH COMMANDS DATASET

❖ 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





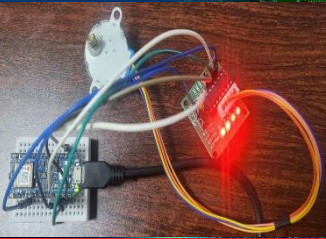
SPEECH COMMANDS DATASET

❖ 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





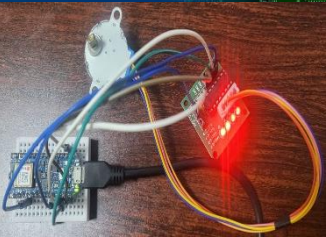
SPEECH COMMANDS DATASET

❖ 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**"



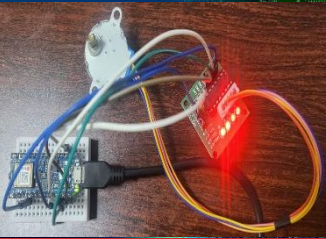


SPEECH COMMANDS DATASET

❖ 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





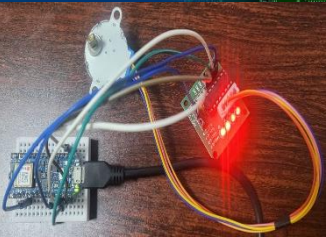
SPEECH COMMANDS DATASET

❖ 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





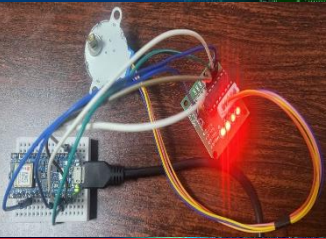
SPEECH COMMANDS DATASET

❖ 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





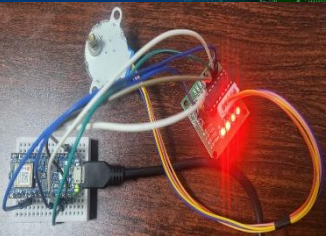
SPEECH COMMANDS DATASET

❖ 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





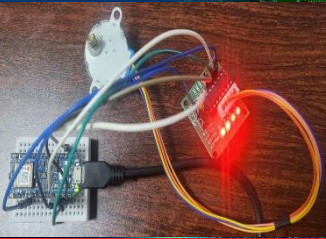
SPEECH COMMANDS DATASET

❖ Speech Commands Data Set v0.02

□ Background Noise

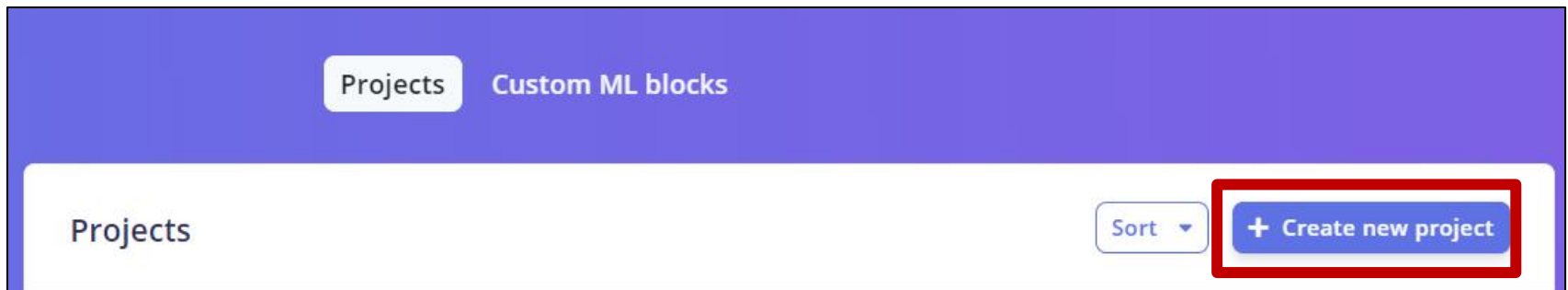
- To help train networks to cope with noisy environments, it can be helpful to mix in realistic background audio
- The ‘_background_noise_’ folder contains a set of longer audio clips that are either recordings or mathematical simulations of noise
- For more details, see the ‘_background_noise_/README.md’

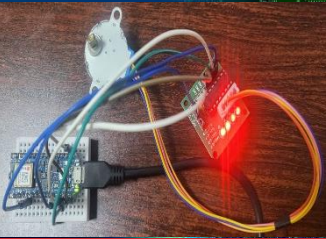




MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Create project





MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Create project

Enter the name for your new project:


Robotic_SubSystem


Choose your project type:


☒ **Personal**
20 min job limit, 4GB or 4 hours of data, limited collaboration.

☐ **Enterprise**
No job or data size limits, higher performance, custom blocks.

Choose your project setting:

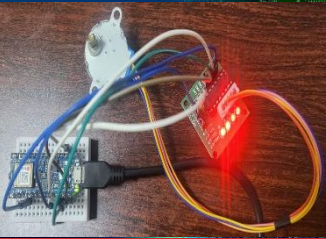
☒  **Public**
Anyone on the internet can view and clone this project under the licence: [Apache 2.0](#). Only invited users will be able to edit.

☐  **Private** (0 of 2 remaining)
Only invited users can edit and view your project.

 Want full-feature access and unlimited projects? [Try Enterprise free.](#)

Create new project





MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Data acquisition

- ☐ Forward
- ☐ Backward
- ☐ Noise/Silence



Dashboard



Devices



Data acquisition



Impulse design



Create impulse

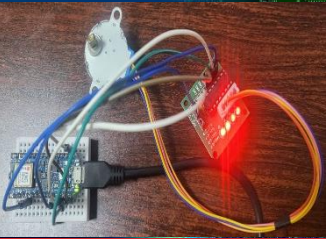


MFCC



Classifier





MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Data acquisition

- ☐ Forward
- ☐ Backward
- ☐ Noise/Silence

Upload data

You can upload CBOR, JSON, CSV, WAV, JPG, PNG, AVI or MP4 files. You can also upload an annotation file named "info.labels" with your data to assign bounding boxes, labels, and/or metadata. View [Uploader docs](#) to learn more. Alternatively, you can use our [Python SDK](#) to programmatically ingest data in various formats, such as pandas or numpy.

For CSV files, [configure the CSV Wizard](#) to define how your files should be processed before uploading files.

Upload mode

☐ Select individual files ?

☒ Select a folder ?

Select files

Choose Files

No file chosen

Upload into category

☒ Automatically split between training and testing ?

☐ Training

☐ Testing

Label

☒ Infer from filename ?

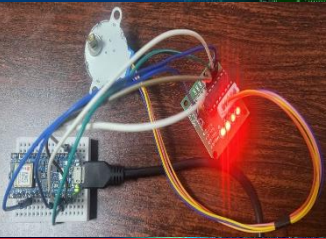
☐ Leave data unlabeled ?

☐ Enter label:

Enter a label

< Back

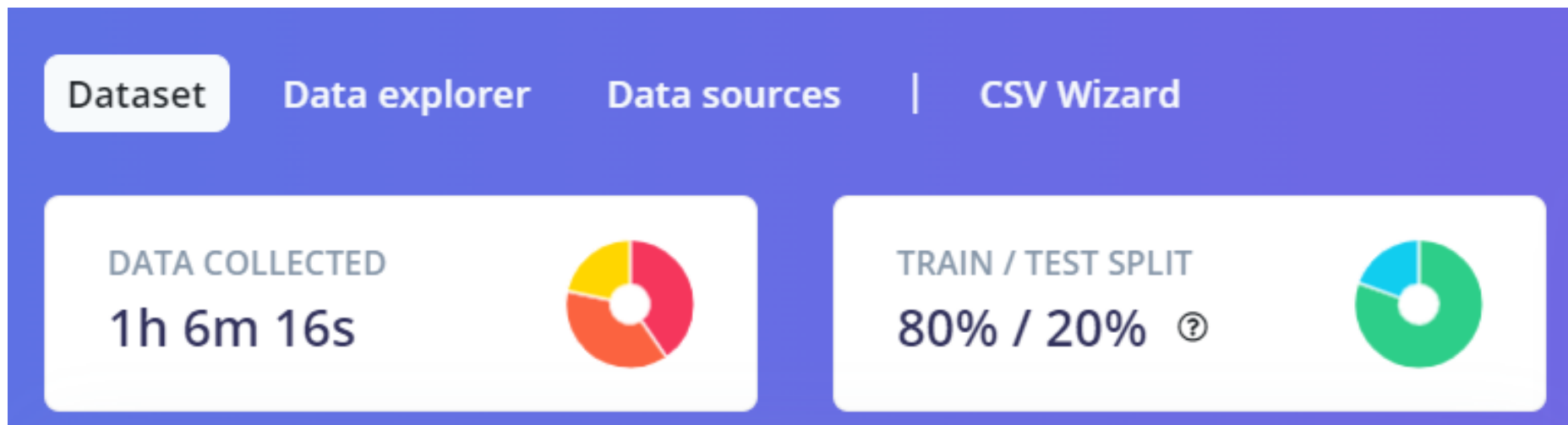
Upload data

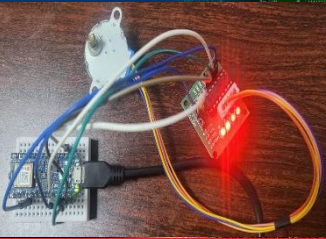


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Data acquisition

- ☐ Forward
- ☐ Backward
- ☐ Noise/Silence

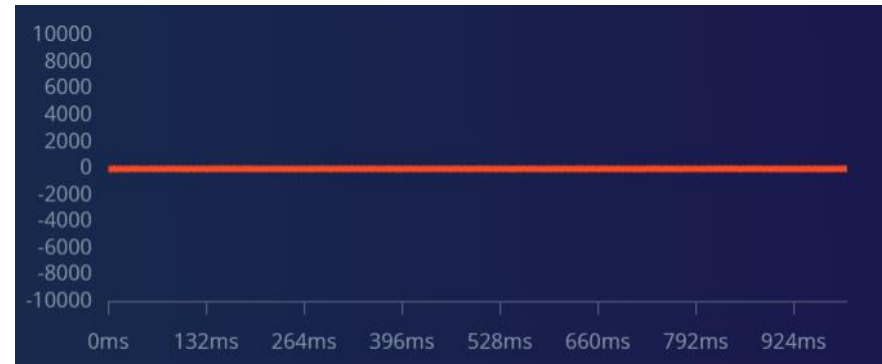




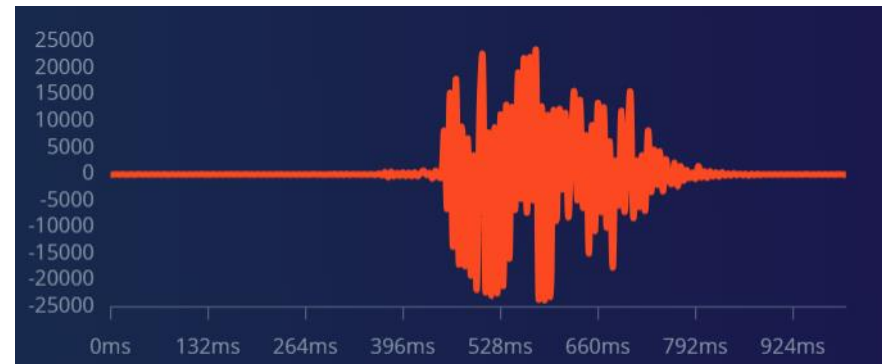
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Data acquisition

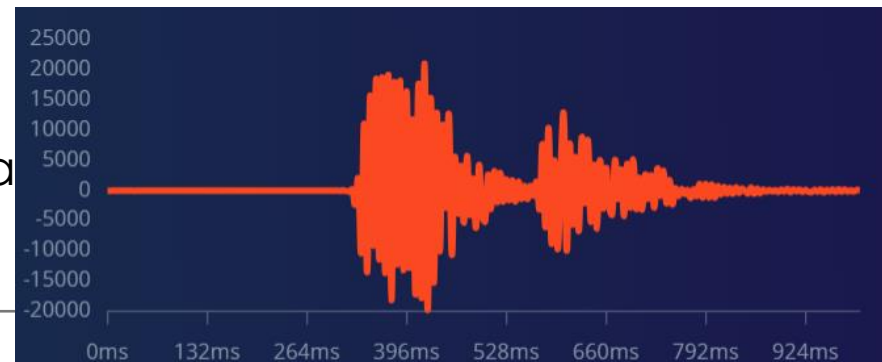
- ☐ Forward
 - ☐ Backward
 - ☐ Noise/Silence
- Silence raw data

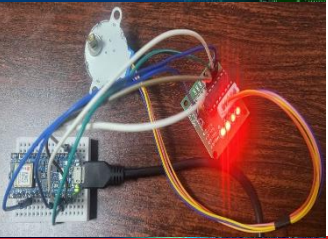


Forward raw data



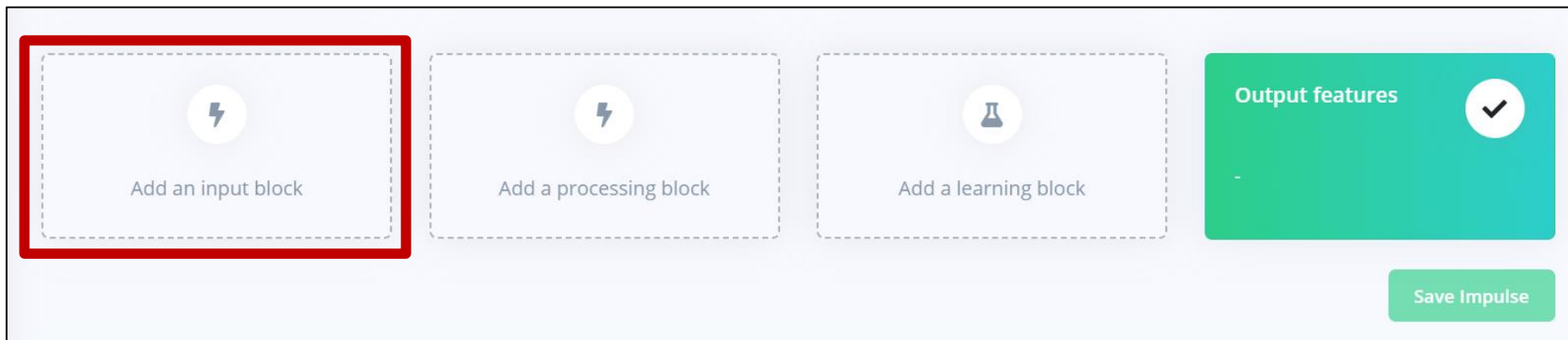
Backward raw data

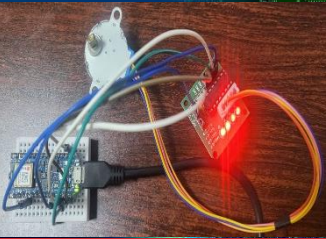




MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE


- ❖ Impulse design
 - Create impulse






MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

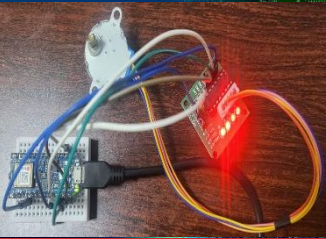
- ❖ Impulse design
 - Create impulse

 Add an input block ×

DESCRIPTION	RECOMMENDED
Time series data <small>OFFICIALLY SUPPORTED</small> Operates on time series sensor data like vibration or audio data. Lets you slice up data into windows.	 Add
Images <small>OFFICIALLY SUPPORTED</small> Processes discrete images for object detection or classification.	Add

Cancel






MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE


- ❖ Impulse design
 - Create impulse

Time series data




Input axes


audio


Window size 


1,000 ms.

Window increase 



500 ms.

Frequency (Hz) 

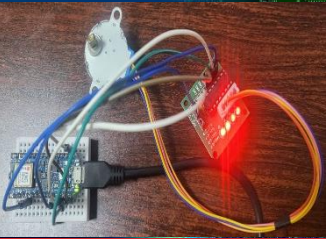


Zero-pad data 

☒



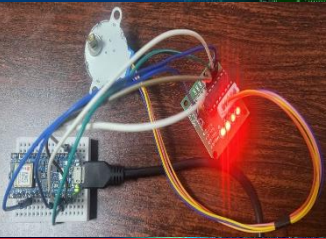


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- ❖ Impulse design
 - Create impulse


The screenshot displays the Edge Impulse web interface for creating a machine learning model. On the left, the 'Time series data' panel is visible, containing settings for 'Input axes' (audio), 'Window size' (1,000 ms), 'Window increase' (500 ms), 'Frequency (Hz)' (16000), and 'Zero-pad data' (checked). The central workspace has two dashed boxes for adding blocks: 'Add a processing block' (highlighted with a red border) and 'Add a learning block'. On the right, the 'Output features' panel shows a checkmark icon, and a 'Save Impulse' button is located below it.





MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

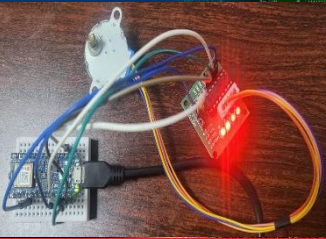
- ❖ Impulse design
 - Create impulse

 Add a processing block ×

Did you know? You can [bring your own DSP code](#).


DESCRIPTION	AUTHOR	RECOMMENDED
Audio (MFCC) OFFICIALLY SUPPORTED Extracts features from audio signals using Mel Frequency Cepstral Coefficients, great for human voice.	Edge Impulse	★ <button>Add</button>
Audio (MFE) OFFICIALLY SUPPORTED Extracts a spectrogram from audio signals using Mel-filterbank energy features, great for non-voice audio.	Edge Impulse	★ <button>Add</button>
Spectrogram OFFICIALLY SUPPORTED Extracts a spectrogram from audio or sensor data, great for non-voice audio or data with continuous frequencies.	Edge Impulse	<button>Add</button>
Audio (Syntiant) OFFICIALLY SUPPORTED Syntiant only. Compute log Mel-filterbank energy features from an audio signal.	Syntiant	<button>Add</button>
Raw Data OFFICIALLY SUPPORTED Use data without pre-processing. Useful if you want to use deep learning to learn features.	Edge Impulse	<button>Add</button>





MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- ❖ Impulse design
 - Create impulse

Audio (MFCC) 


Name

MFCC

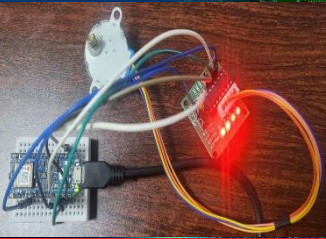
Input axes (1)

Signal ?

audio







MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- ❖ Impulse design
 - Create impulse

Time series data

Input axes

audio

Window size

1,000 ms.

Window increase

500 ms.

Frequency (Hz)

16000

Zero-pad data

☒

Audio (MFCC)

Name

MFCC

Input axes (1)

Signal

audio

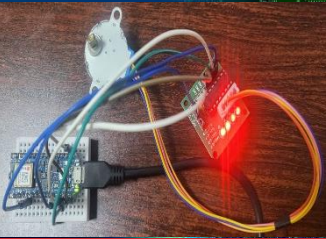
Add a learning block

Output features

-

Save Impulse







MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Impulse design

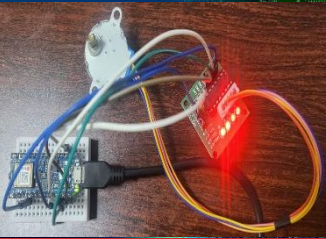
□ Create impulse

 Add a learning block ×

Did you know? You can [bring your own model](#) in PyTorch, Keras or scikit-learn.

DESCRIPTION	AUTHOR	RECOMMENDED
Classification <small>OFFICIALLY SUPPORTED</small> Learns patterns from data, and can apply these to new data. Great for categorizing movement or recognizing audio.	Edge Impulse	 <button>Add</button>
Regression <small>OFFICIALLY SUPPORTED</small> Learns patterns from data, and can apply these to new data. Great for predicting numeric continuous values.	Edge Impulse	<button>Add</button>
Transfer Learning (Keyword Spotting) <small>OFFICIALLY SUPPORTED</small> Fine tune a pre-trained keyword spotting model on your data. Good performance even with relatively small keyword datasets.	Edge Impulse	<button>Add</button>
Anomaly Detection (GMM) <small>PROFESSIONAL</small> <small>ENTERPRISE</small> Find outliers in new data. A Gaussian mixture model (GMM) models the shape of data using a probability distribution. New data that is unlikely according to this model can be considered anomalous.	Edge Impulse	






MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- ❖ Impulse design
 - Create impulse

Classification




Name

Input features

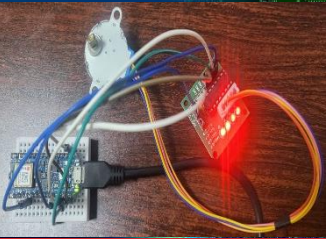
☒ MFCC

Output features

3 (Backward, Forward, Silence)







MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- ❖ Impulse design
 - Create impulse

Time series data

Input axes

audio

Window size

1,000 ms.

Window increase

500 ms.

Frequency (Hz)

16000

Zero-pad data

☒

Audio (MFCC)

Name

MFCC

Input axes (1)

audio

Signal

audio

Classification

Name

Classifier

Input features

☒ MFCC

Output features

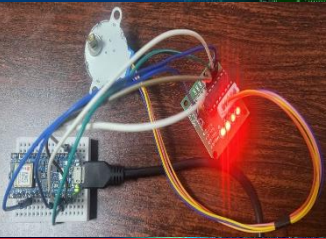
3 (Backward, Forward, Silence)

Output features

3 (Backward, Forward, Silence)

Save Impulse

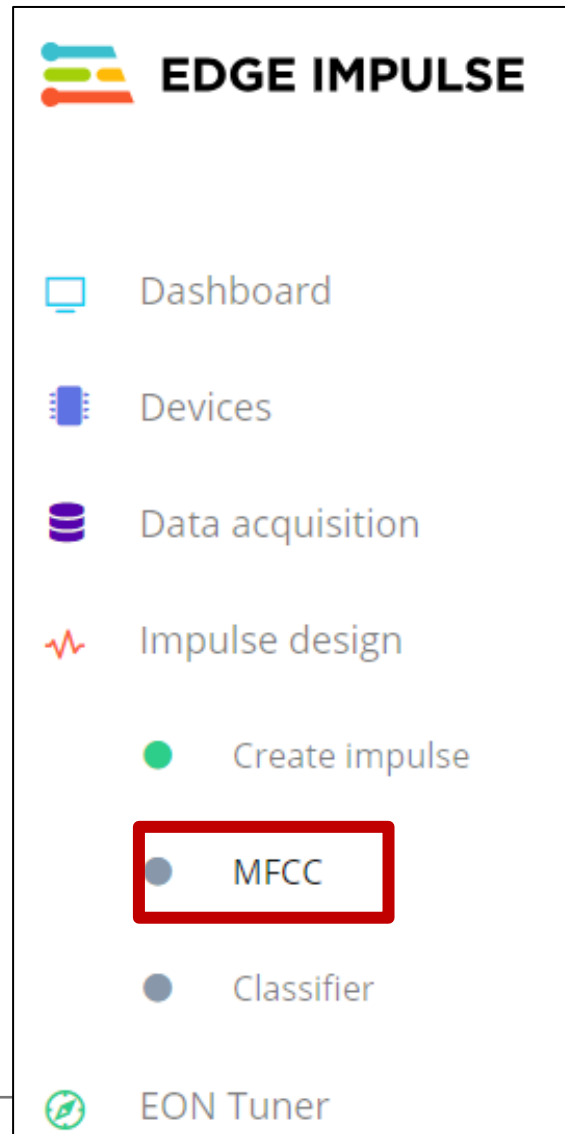
52

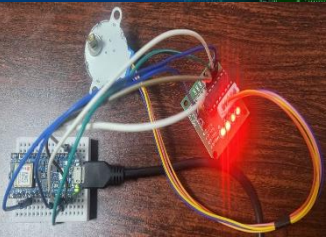


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Impulse design

□ MFCC



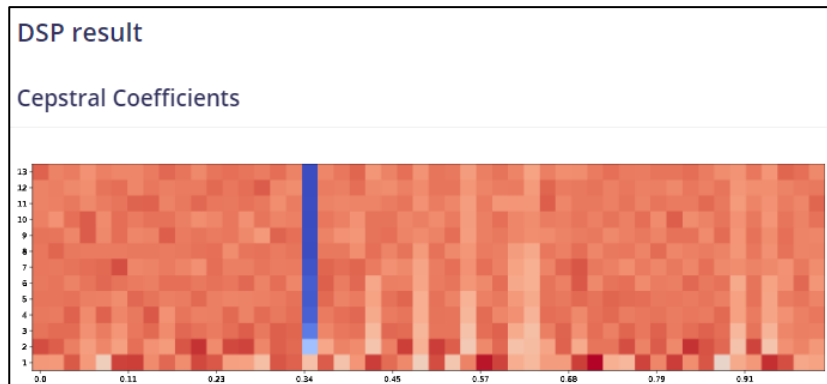


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

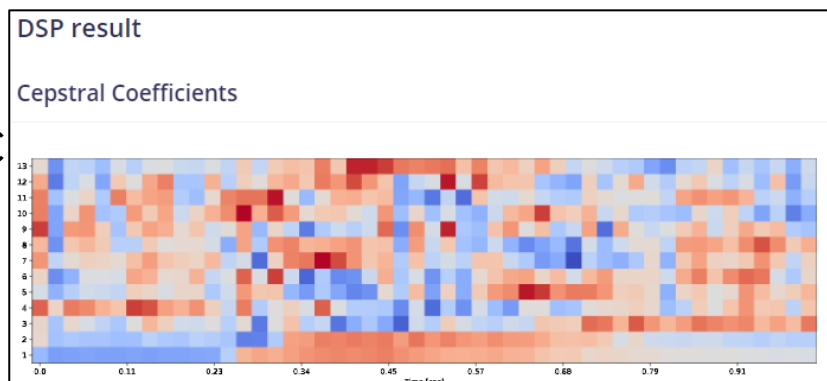
❖ Impulse design

□ MFCC

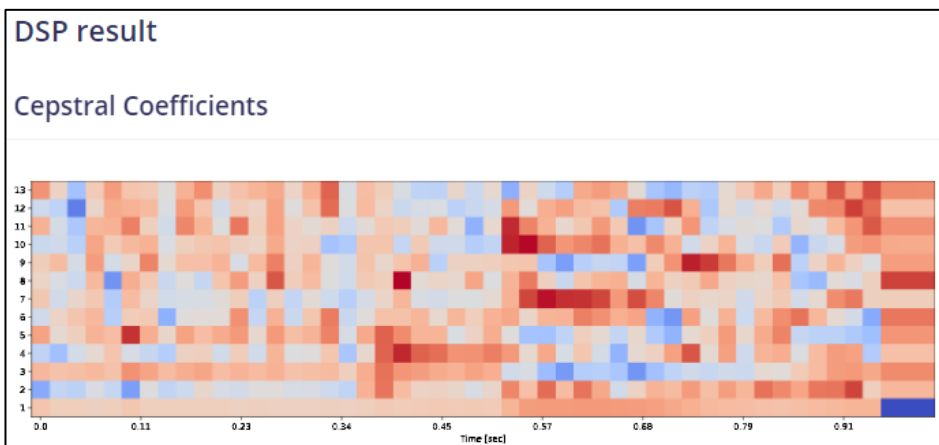
Silence MFCC

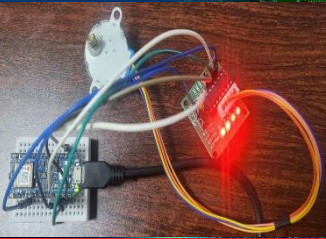


Forward MFCC



Backward MFCC





MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Impulse design

□ MFCC

Parameters

Autotune parameters

Mel Frequency Cepstral Coefficients

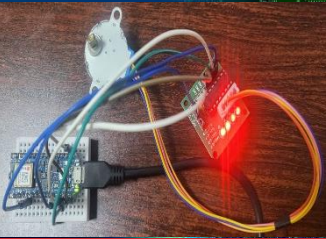
Number of coefficients ?	13
Frame length ?	0.02
Frame stride ?	0.02
Filter number ?	32
FFT length ?	256
Normalization window size ?	101
Low frequency ?	0
High frequency ?	Click to set

Pre-emphasis

Coefficient ?	0.98
---------------	------

Save parameters





MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Impulse design

□ MFCC

#1 ▾ Click to set a description for this version

Parameters **Generate features**

Training set

Data in training set	53m 18s
Classes	3 (Backward, Forward, Silence)
Training windows	3,235

Generate features

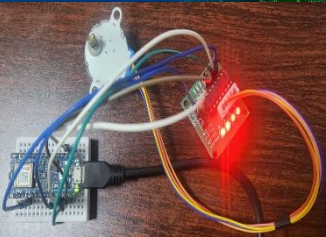
Feature explorer

No features generated yet.

Feature generation output

🔊 (0) ▾





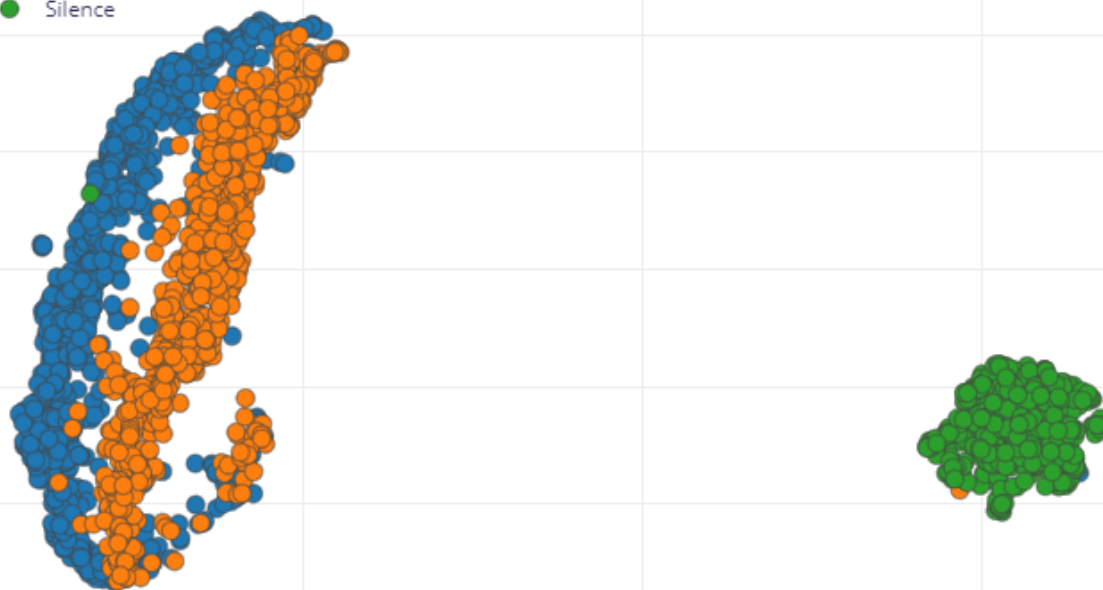
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Impulse design

□ MFCC

Feature explorer ⓘ

● Backward
● Forward
● Silence



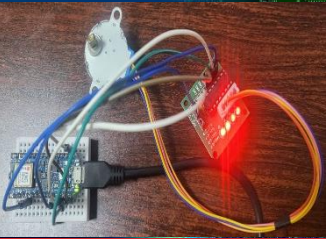
On-device performance ⓘ



PROCESSING TIME
176 ms.



PEAK RAM USAGE
13 KB



MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Impulse design

□ Classifier

Neural Network settings

Training settings

Number of training cycles ?

100

Use learned optimizer ?



Learning rate ?

0.005

Training processor ?

CPU



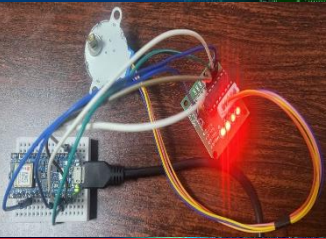
Advanced training settings



Audio training options

Data augmentation ?





MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Impulse design

□ Classifier

Neural network architecture

Architecture presets ⓘ [1D Convolutional \(Default\)](#) [2D Convolutional](#)

Input layer (650 features)

Reshape layer (13 columns)

1D conv / pool layer (8 neurons, 3 kernel size, 1 layer)

Dropout (rate 0.25)

1D conv / pool layer (16 neurons, 3 kernel size, 1 layer)

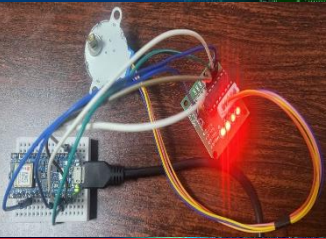
Dropout (rate 0.25)

Flatten layer

Add an extra layer

Output layer (3 classes)

Start training



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)

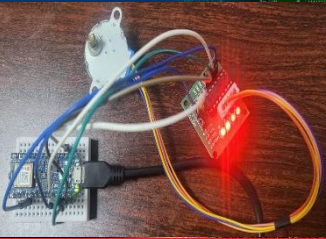
	BACKWARD	FORWARD	SILENCE
BACKWARD	98.5%	1.2%	0.4%
FORWARD	2.9%	97.1%	0%
SILENCE	0.7%	0%	99.3%
F1 SCORE	0.98	0.98	0.99

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

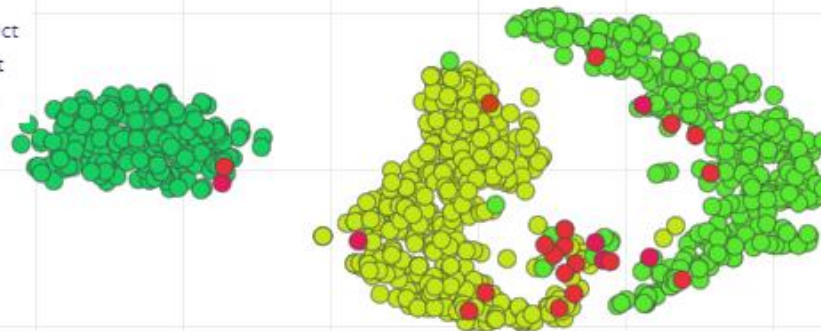
Model

Model version: ?

Unoptimized (float32) ▼

Data explorer (full training set) ?

- Backward - correct
- Forward - correct
- Silence - correct
- Backward - incorrect
- Forward - incorrect
- Silence - incorrect



On-device performance ?

Engine: ?

EON™ Compiler ▼



INFERRING TIME
42 ms.

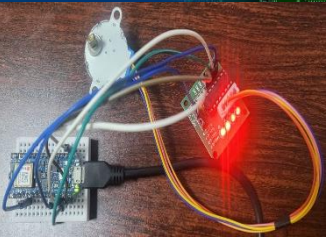


PEAK RAM USAGE
7.0K



FLASH USAGE
28.0K

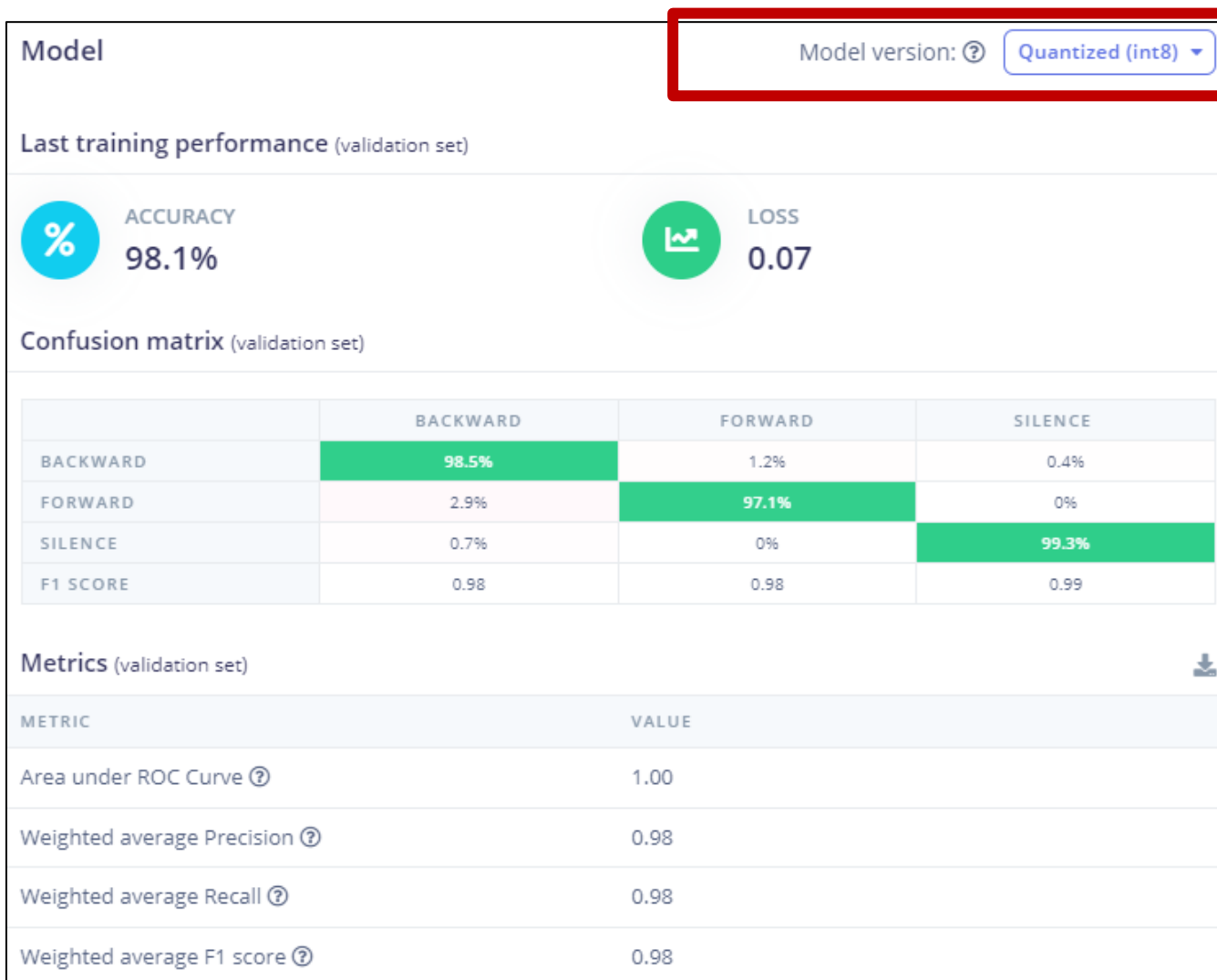


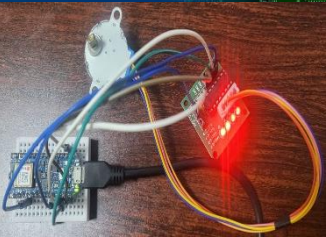


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Impulse design

□ Classifier

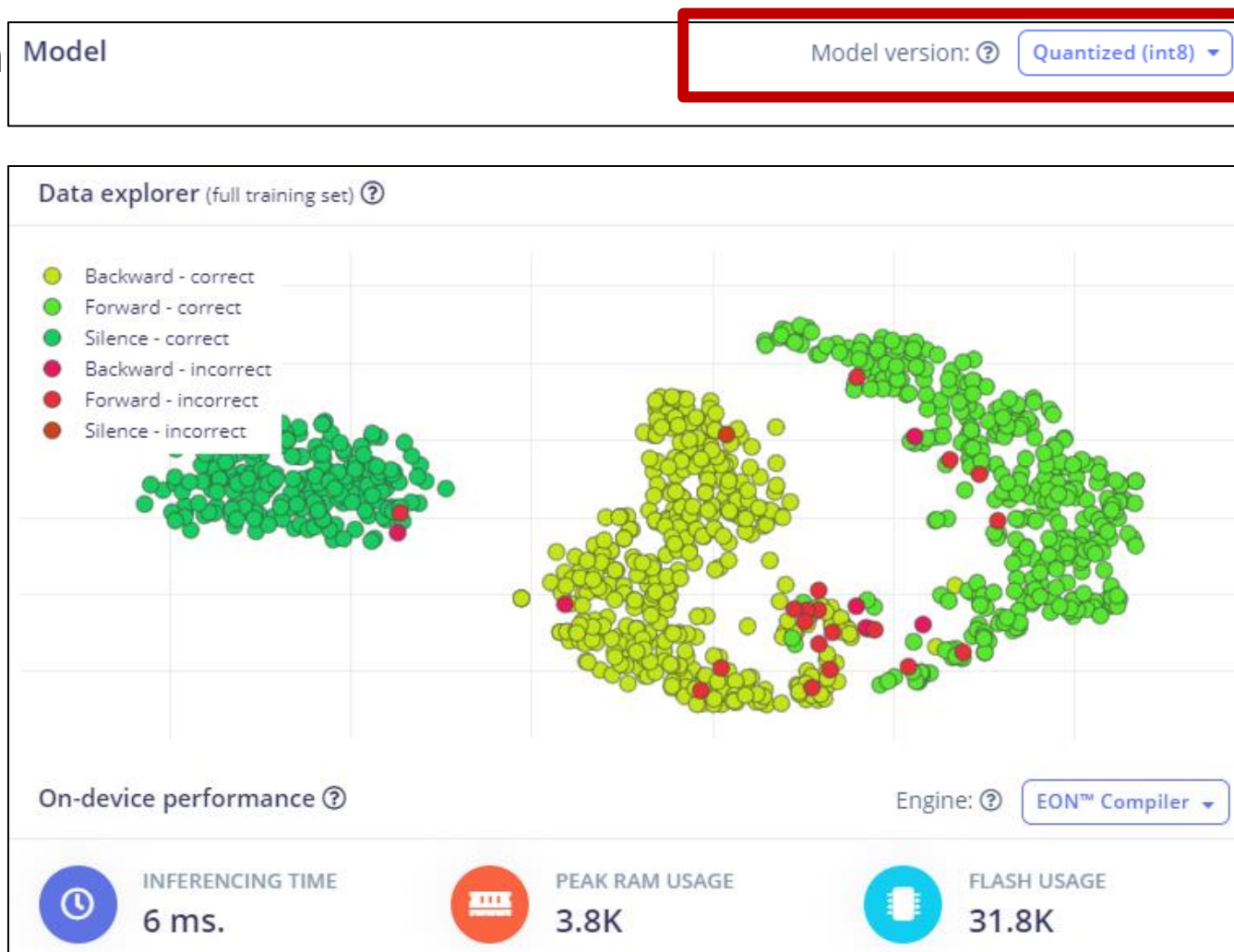


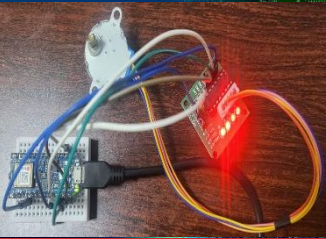


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Impulse design

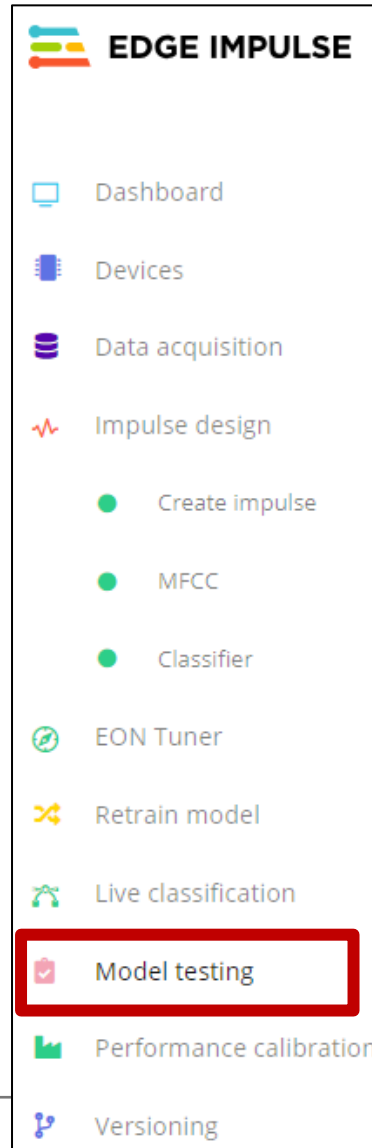
□ Classifier

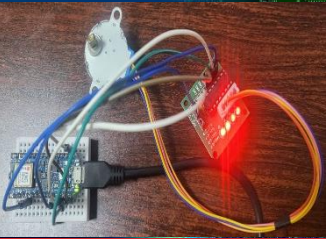




MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

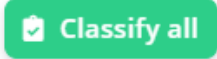

❖ Model testing



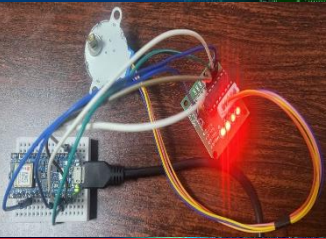


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Model testing

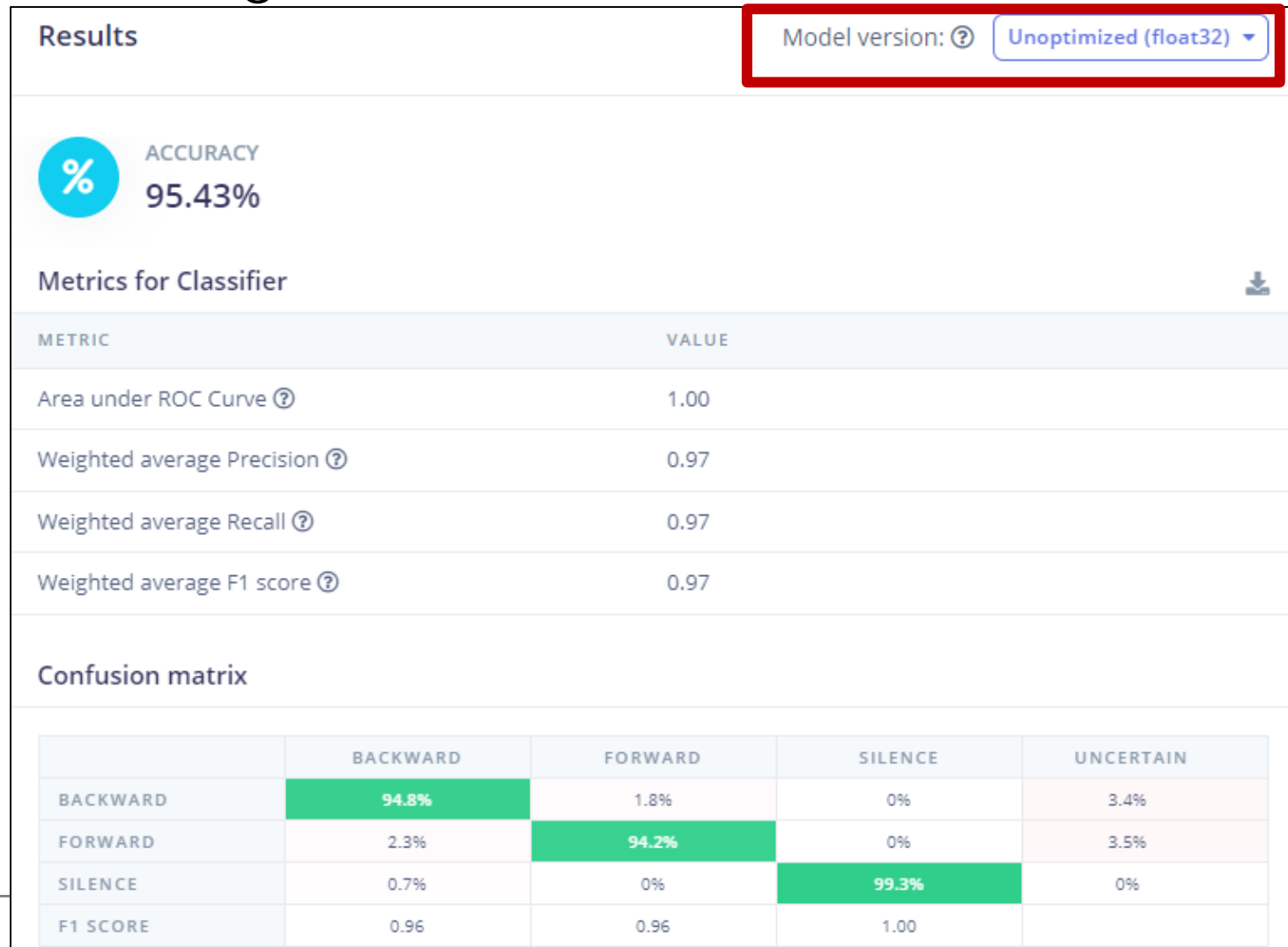
Test data					 
Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse.					
SAMPLE NA...	EXPECTED OUT...	LENG...	ACCURACY	RESULT	
93Label	Silence	1s			⋮
95Label	Silence	1s			⋮
834Label	Silence	1s			⋮
82Label	Silence	1s			⋮
827Label	Silence	1s			⋮
817Label	Silence	1s			⋮

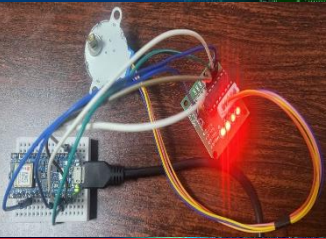




MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

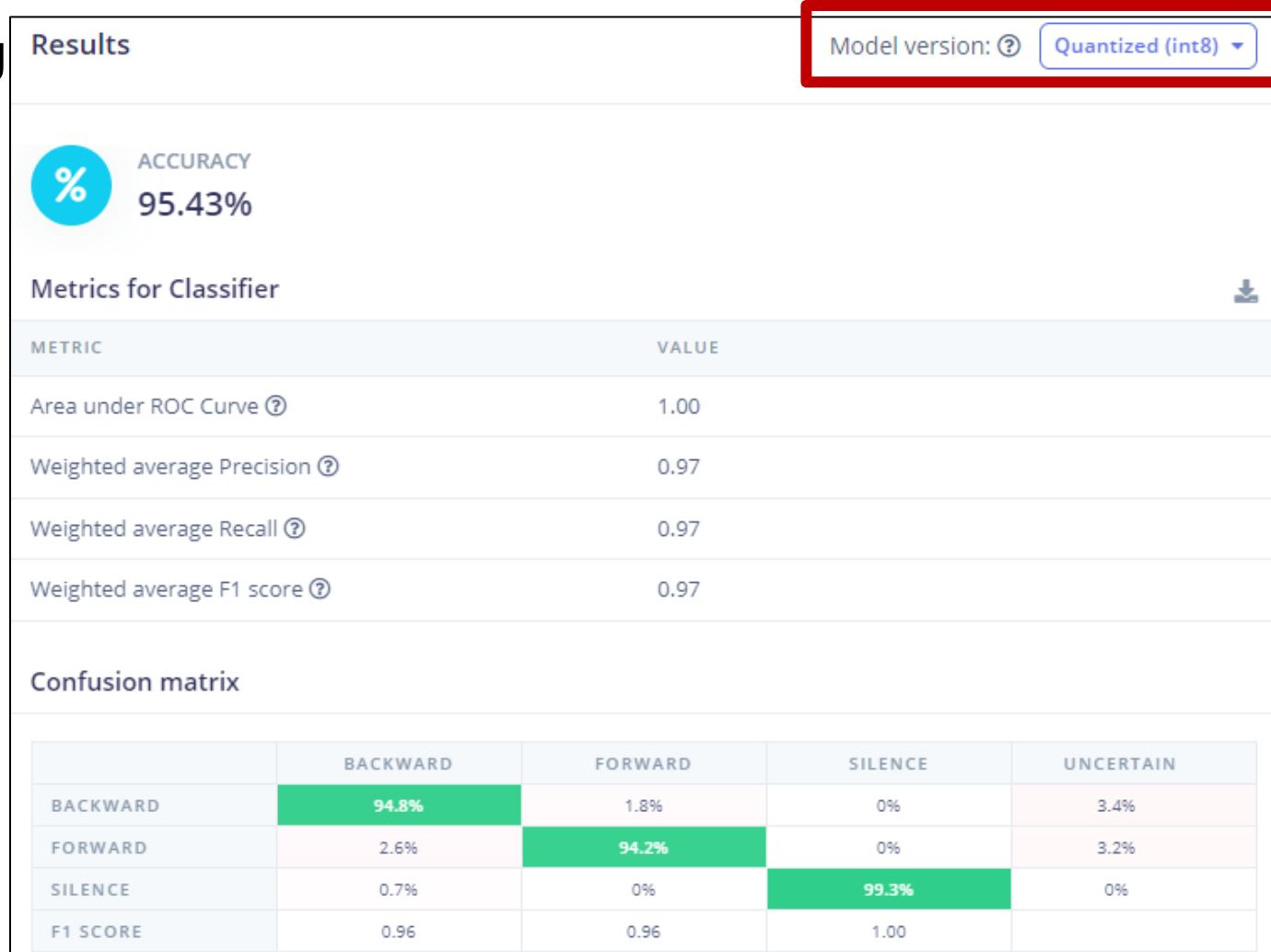
❖ Model testing

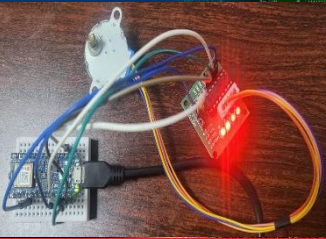




MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE













❖ Model testing



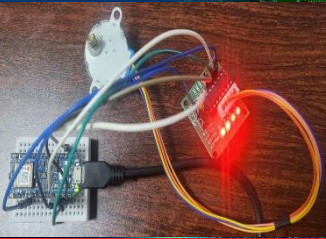


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Live Classification

-  Data acquisition
-  Impulse design
 -  Create impulse
 -  MFCC
 -  Classifier
-  EON Tuner
-  Retrain model
-  Live classification
-  Model testing
-  Performance calibration
-  Versioning
-  Deployment







MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Live Classification

Classify new data



Device ?

No devices connected

Sensor

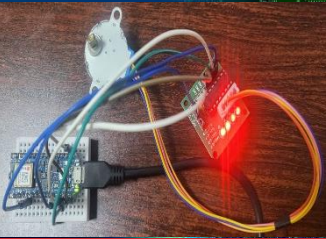
Sample length (ms.)

5000

Frequency

Start sampling






MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE


❖ Live Classification

Collect new data


Collect data directly from your phone, computer, device, or development board.



Scan QR code to connect to your phone

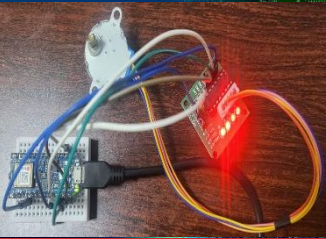


Connect to your computer



Connect your device or development board





MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ 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:

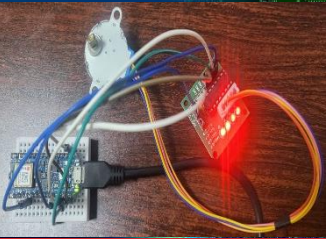


Arduino Nano 33 BLE Sense



View installation docs

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



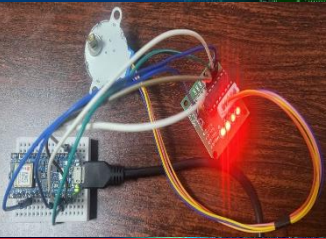
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ 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`





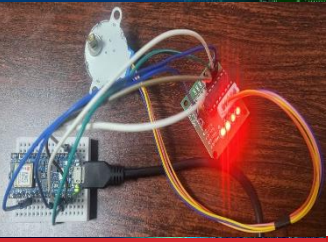
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ 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=wOkMZUdPLUM>



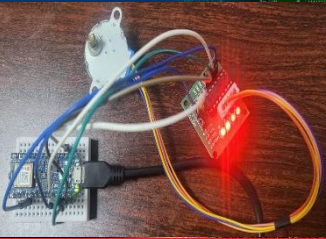


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ 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





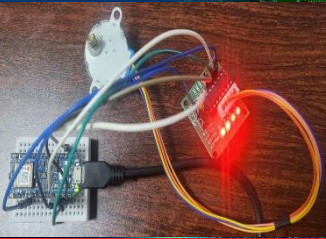
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ 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.



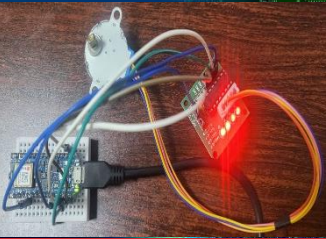


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ 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/>





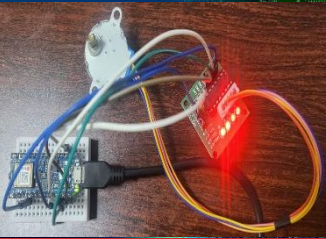
MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ 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)?
```





MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

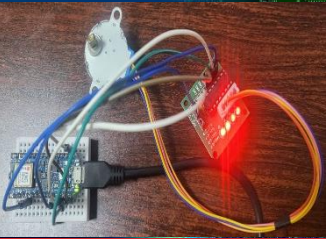
❖ Live Classification

 Select C:\WINDOWS\system32\cmd.exe - "node" "C:\Users\user\AppData\Roaming\npm\node_modules\edge-impulse-cli\build\cli\cli.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
? What is your user name or e-mail address (edgeimpulse.com)? dennisgookyi@gmail.com
? What is your password? [input is hidden]
```





MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

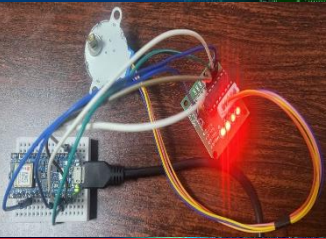
❖ Live Classification

```
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-arduinoonano-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
```

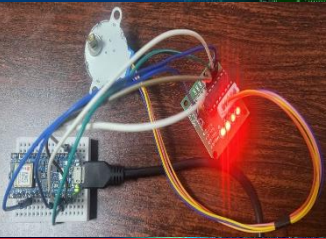
```
[WS ] Connecting to wss://remote-mgmt.edgeimpulse.com
```

```
[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

Classify new data

Device ?

7B:81:E9:42:AE:BC

Sensor

Built-in microphone

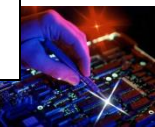
Sample length (ms.)

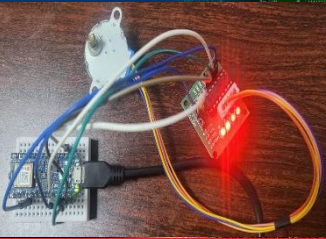
5000

Frequency

16000Hz

Start sampling





MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Live Classification

Classification result

Summary

Model version: Quantized (int8)

Name

Label

CATEGORY	COUNT
Backward	4
Forward	0
Silence	2
uncertain	3

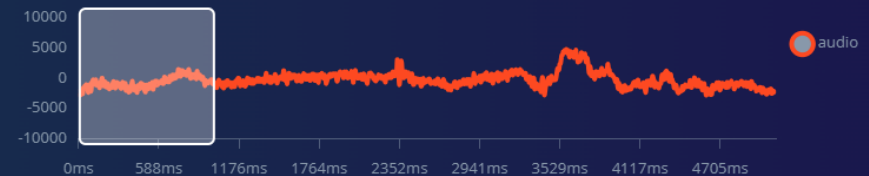
Detailed result

☐ 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

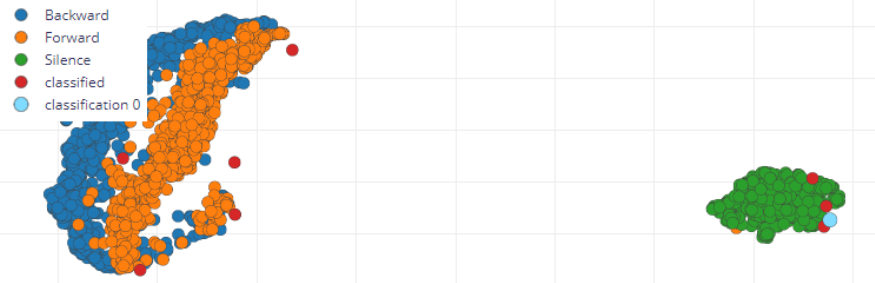
testing.5140e2er



Raw features

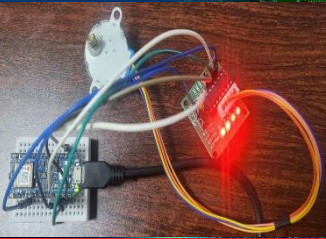
-2170, -2173, -2205, -2199, -2187, -2153, -2146, -2121, -2136, -2140, -2120, -2088, -2041, -2051, -203...

MFCC



Processed features

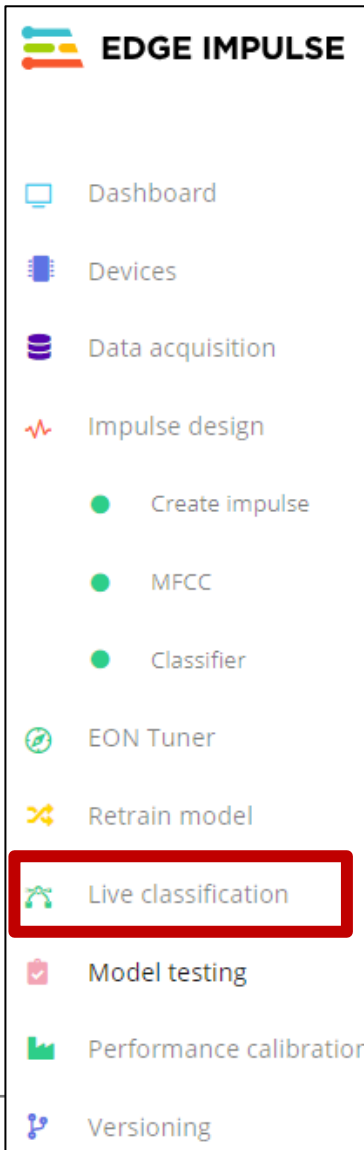
4.2478, -1.0004, 0.4061, 0.9148, 0.1024, -0.1381, 0.7006, 0.4282, 0.4814, 0.1135, 0.5369, 1.2589, 0.7001, 1.6...

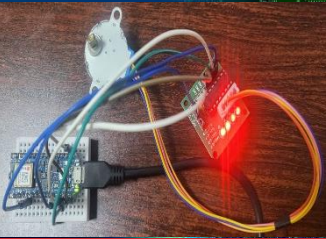


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Live classification

□ Using WebUSB







MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Live classification

- Using WebUSB

Classify new data



Device ?

No devices connected

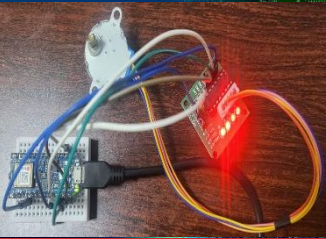
Sensor

Sample length (ms.)

5000

Frequency

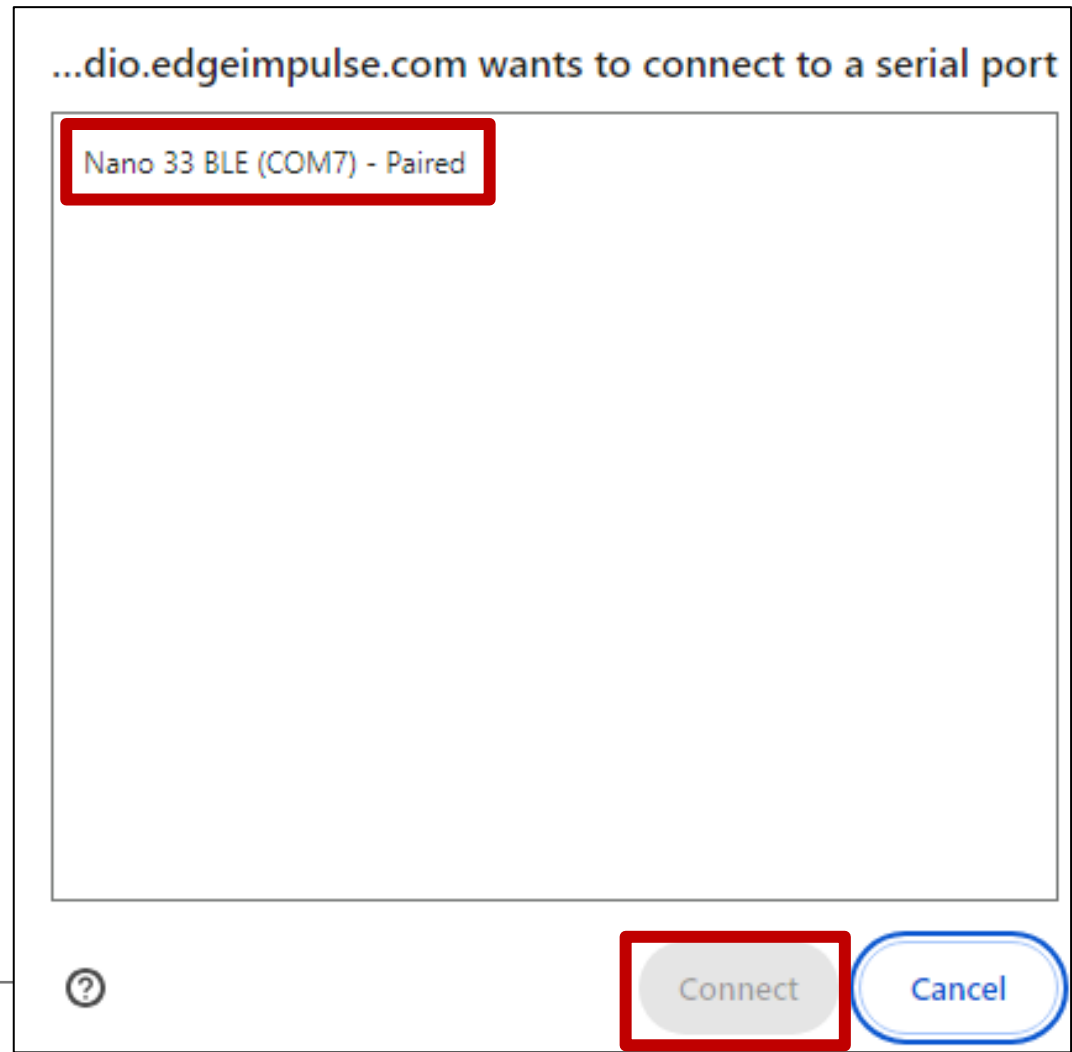
Start sampling

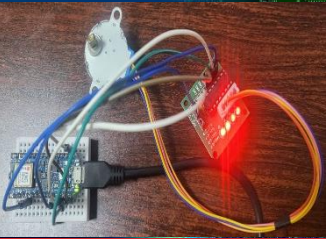


MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Live classification

- Using WebUSB






MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Live classification

□ Using WebUSB

Classify new data



Device ?

NANO1

▼

Sensor

Built-in microphone

▼

Sample length (ms.)

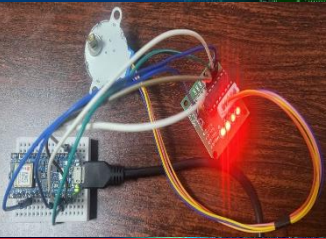
5000

Frequency

16000Hz

▼

Start sampling



MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Live classification

□ Using WebUSB

Classification result

Summary

Model version: ? Quantized (int8) ⋮

Name testing.51ej9u3k

Label testing

CATEGORY	COUNT
Backward	7
Forward	1
Silence	0
uncertain	1

Detailed result

☐ Show only unknowns

TIMESTAMP	BACKWARD	FORWARD	SILENCE
0	0.82	0.15	0.04
500	0.63	0.30	0.07
1,000	0.47	0.53	0

RAW DATA

testing.51ej9u3k



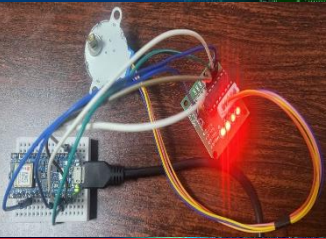
0:00 / 0:01

Raw features

MFCC

Error while loading features: {"response":{"readyState":0,"status":0,"statusText":"error"},"errorThrown":""}

Processed features



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

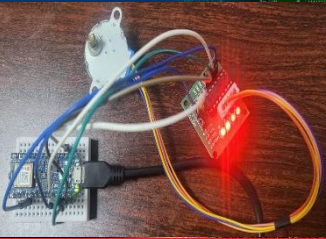
❖ Arduino IDE

- ❑ 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

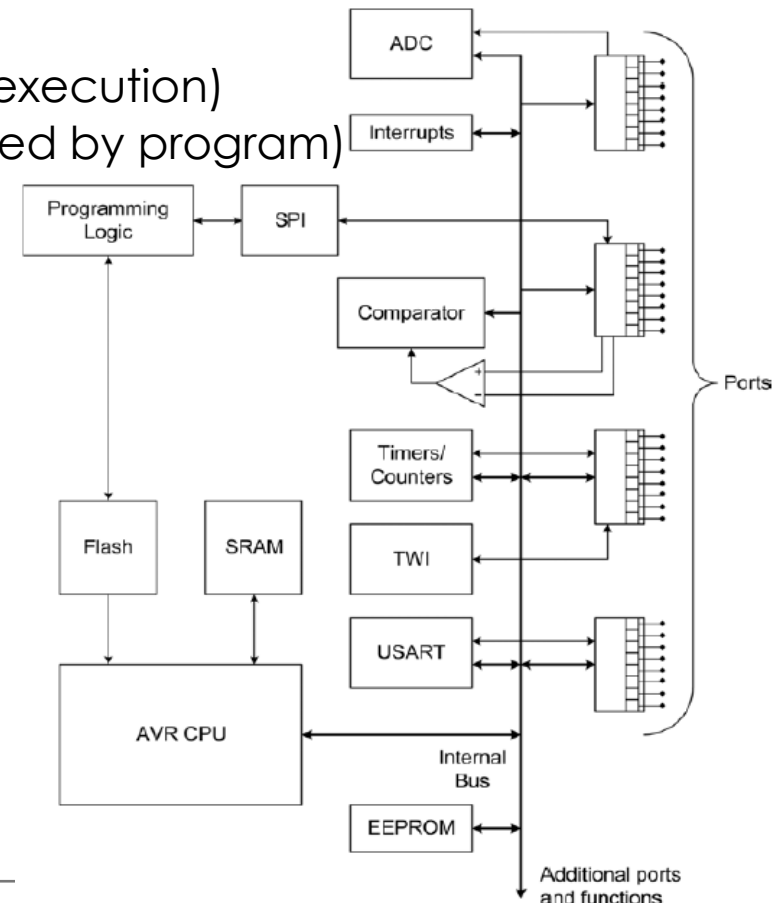


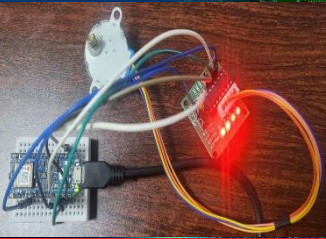


MODEL DEPLOYMENT HARDWARE - ARDUINO

❖ 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

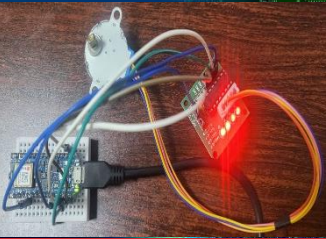
❖ Interfacing with Arduino

- ☐ 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>



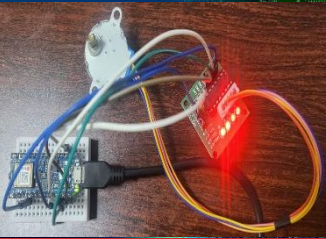


MODEL DEPLOYMENT HARDWARE - ARDUINO

❖ 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

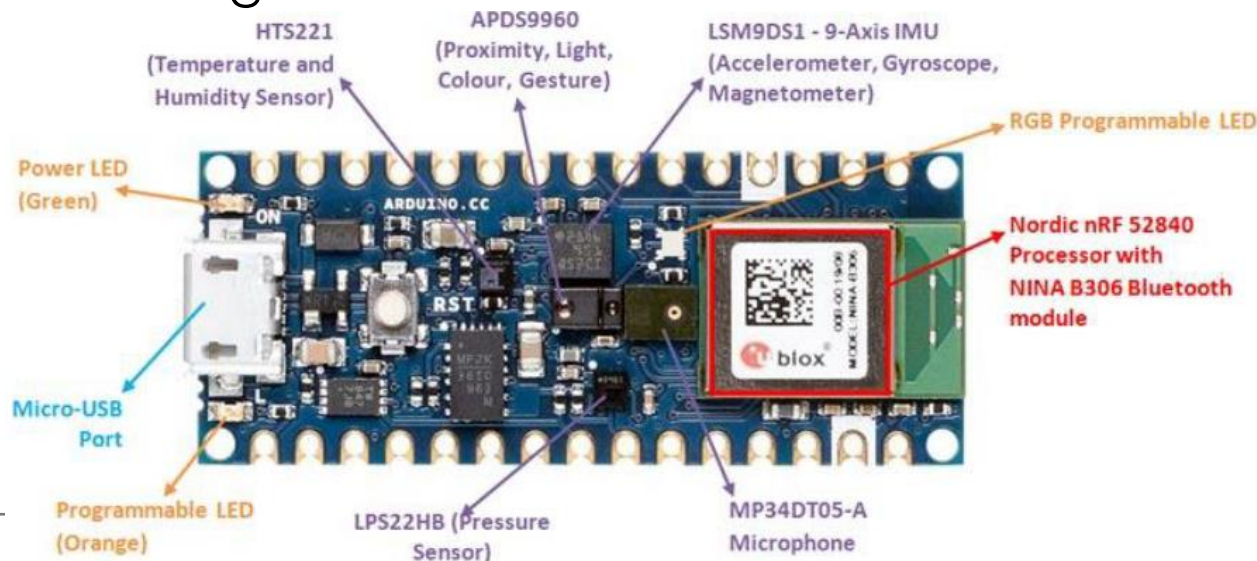


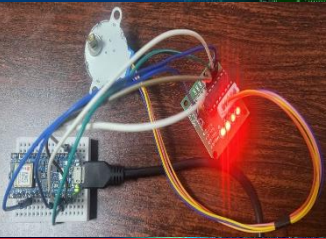


MODEL DEPLOYMENT HARDWARE - ARDUINO

❖ Features of Nano 33 Sense

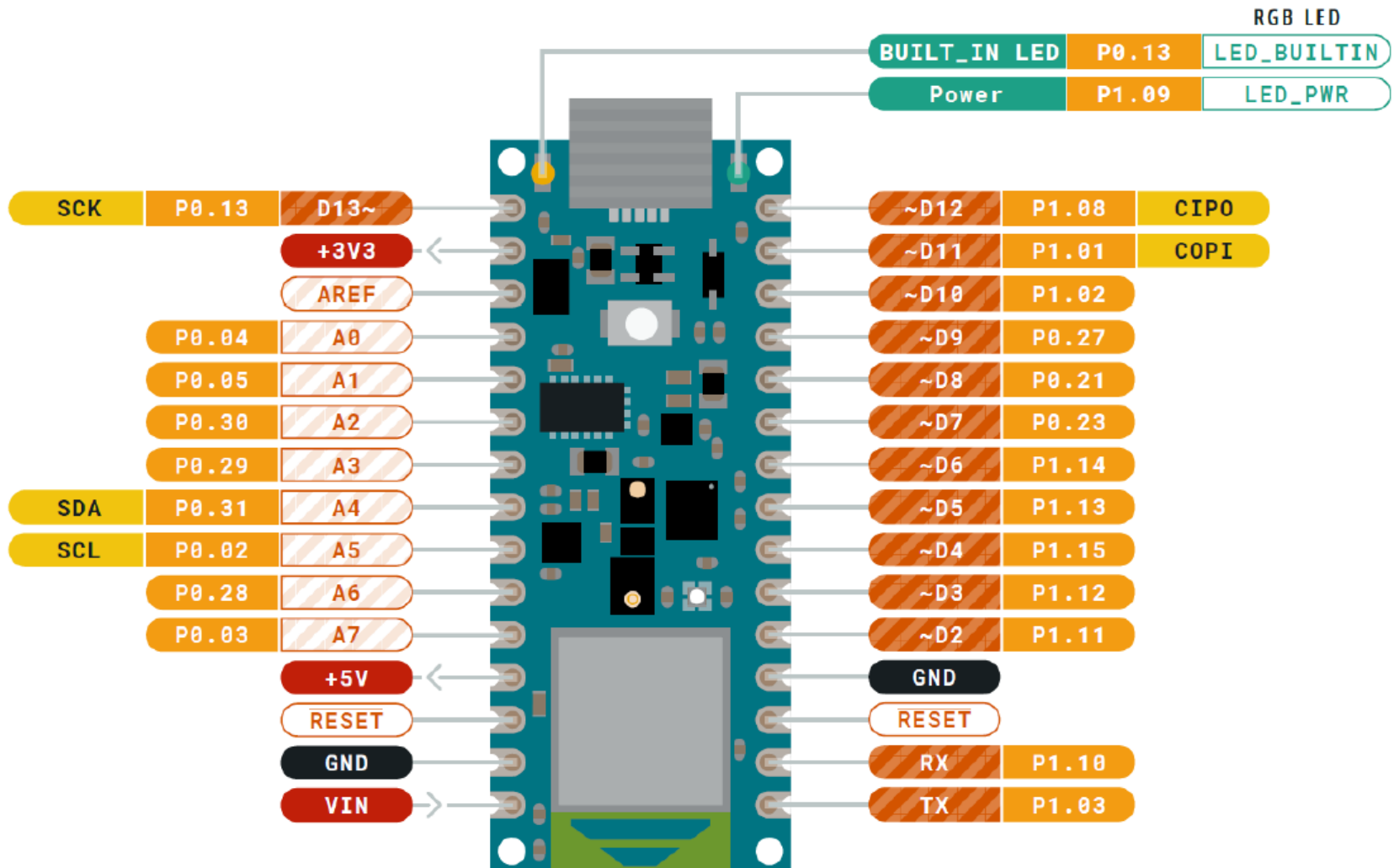
- 8 Analog Input Pins can provide 12-bit ADC at about 30 kHz
- Integrated sensors (IMU, Mic, Light, Pressure, Temperature, Humidity)
- All digital pins can trigger interrupts
- Only supports 3.3V I/Os and is NOT 5V tolerant so please make sure you are not directly connecting 5V signals to this board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

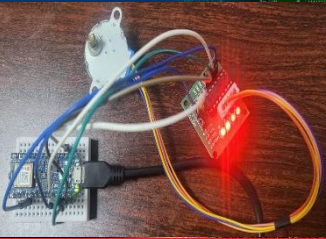




MODEL DEPLOYMENT HARDWARE - ARDUINO

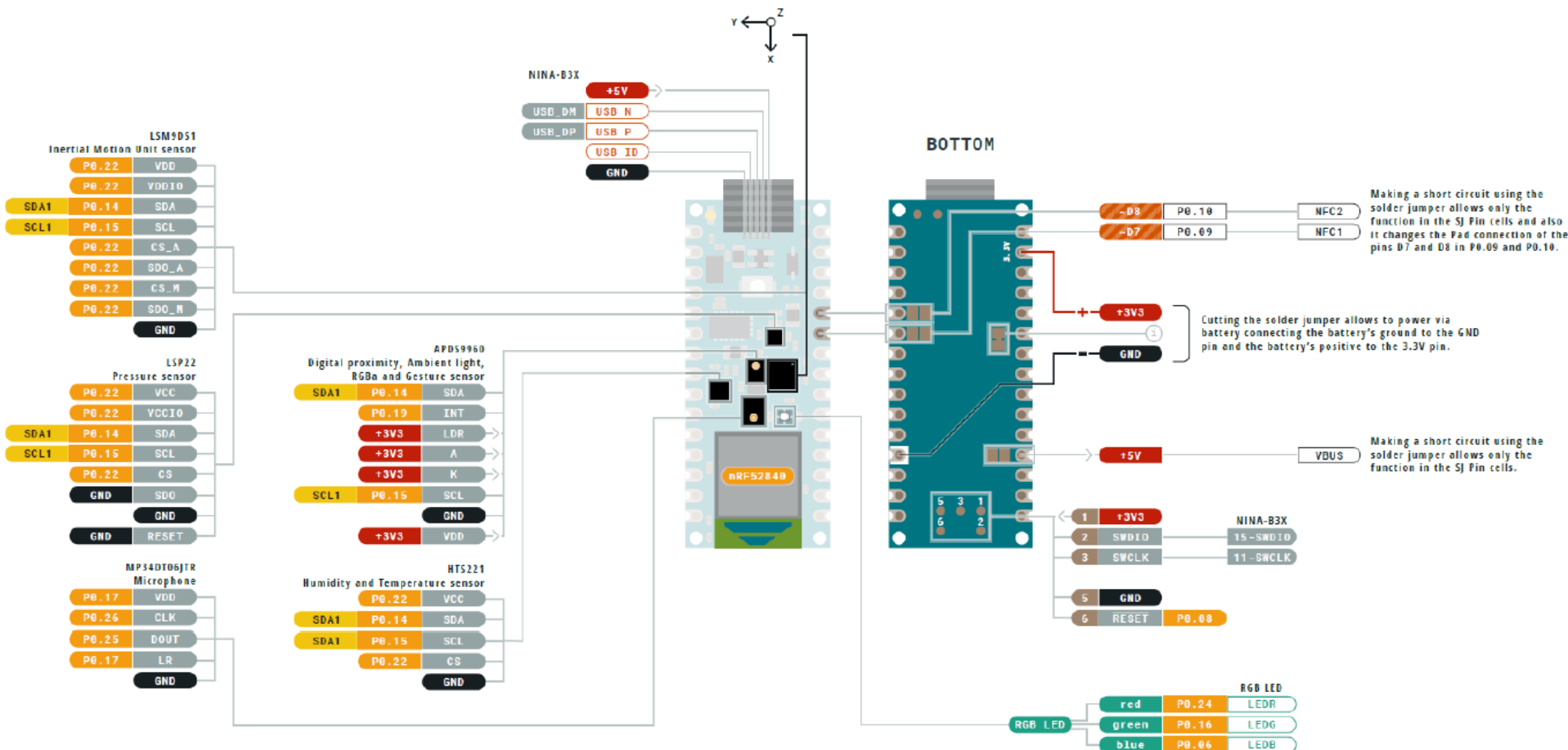
❖ Nano 33 Sense

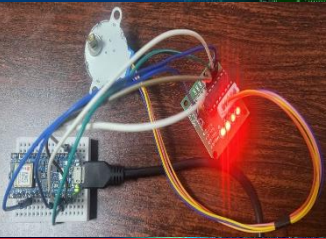




MODEL DEPLOYMENT HARDWARE - ARDUINO

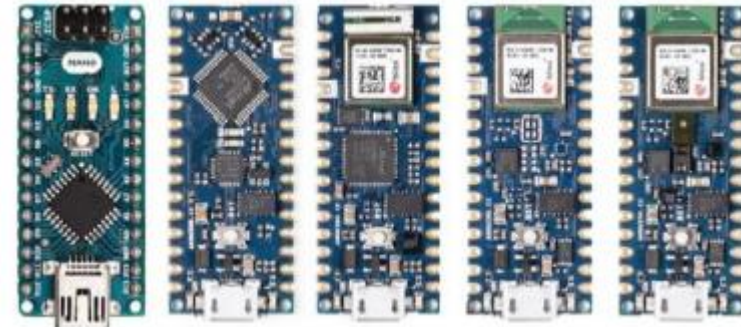
❖ Nano 33 Sense





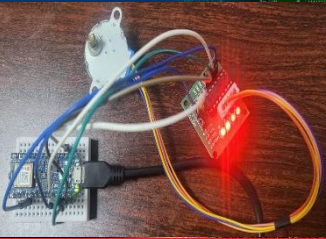
MODEL DEPLOYMENT HARDWARE - ARDUINO

❖ 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



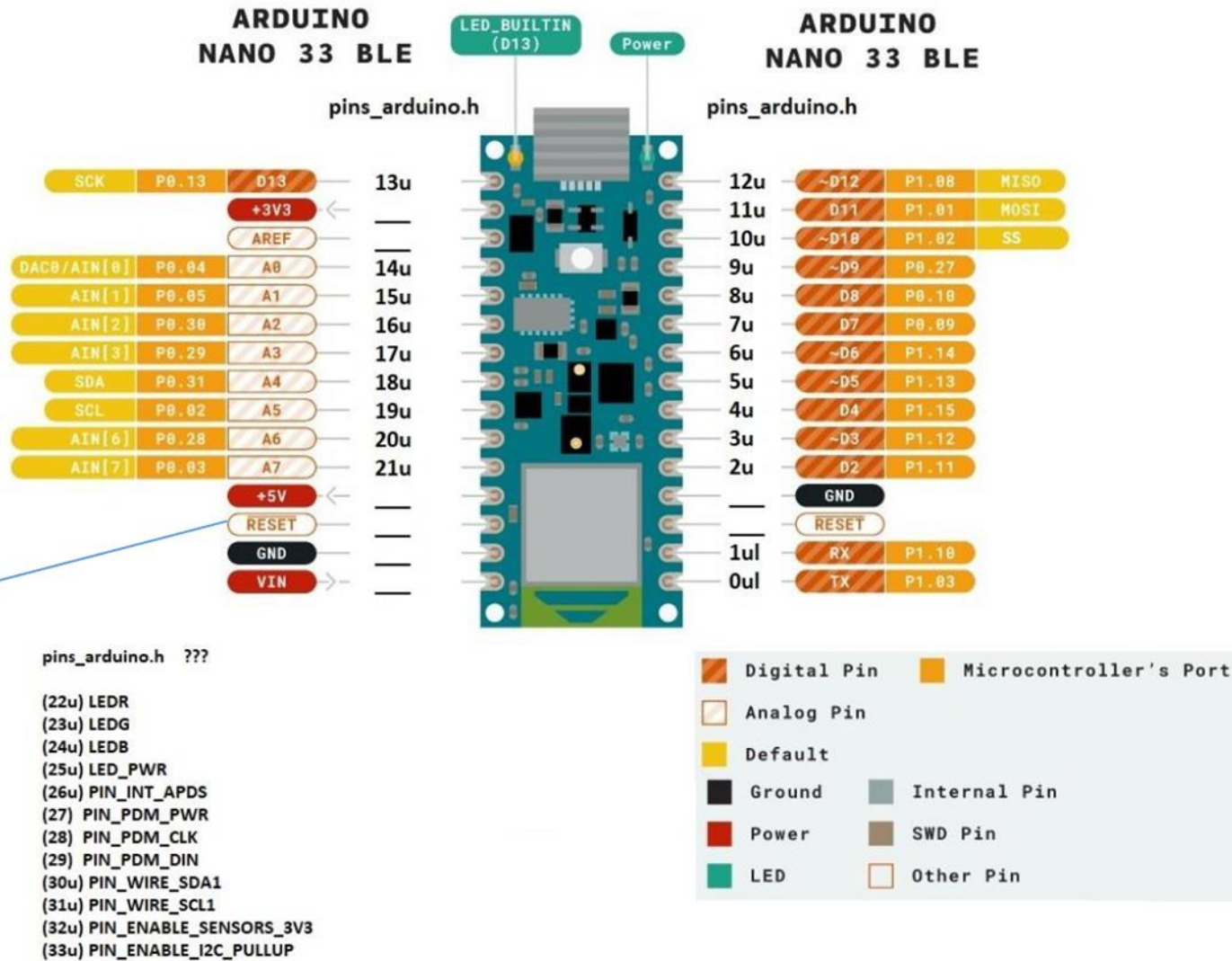


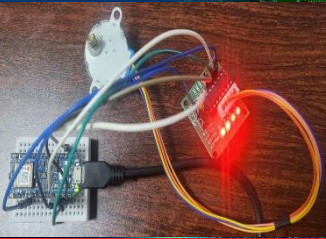
MODEL DEPLOYMENT HARDWARE - ARDUINO

❖ 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.

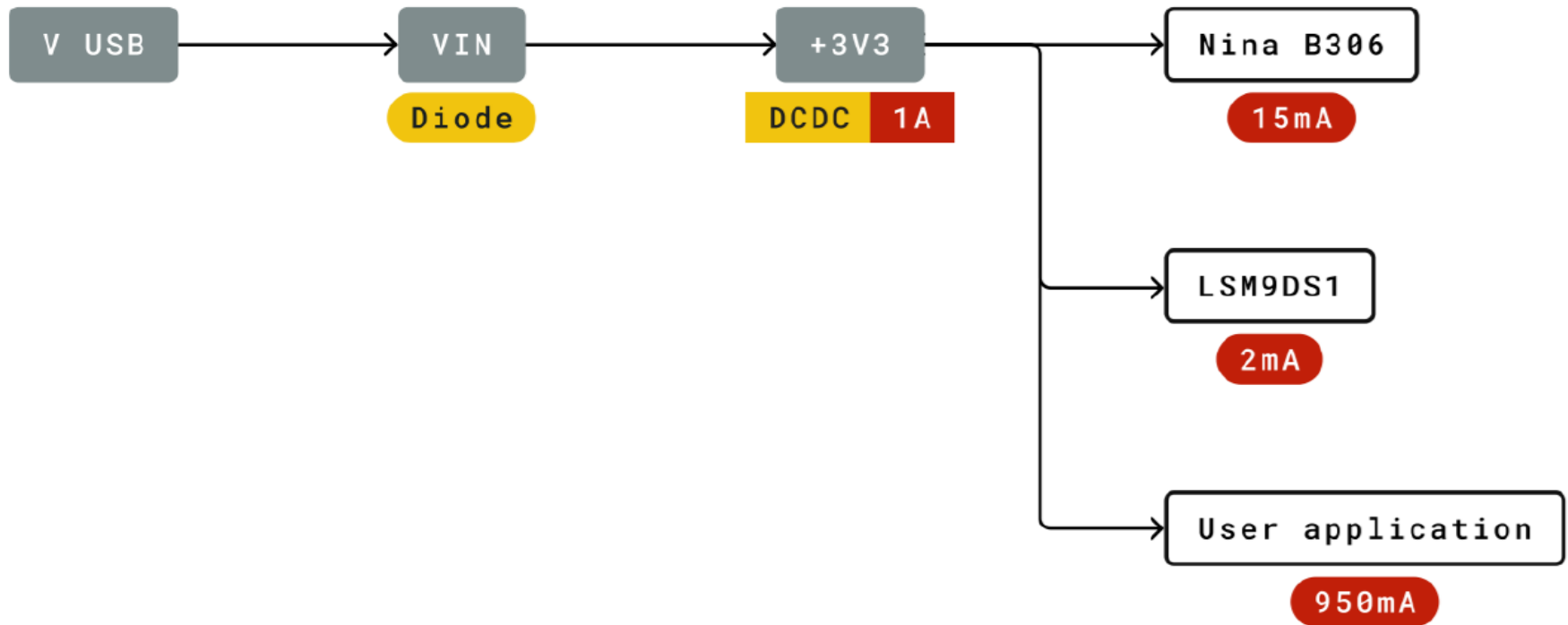
Ground the RESET pin to reset





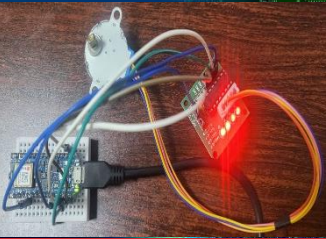
MODEL DEPLOYMENT HARDWARE - ARDUINO

❖ 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.





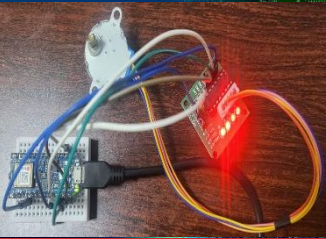
MODEL DEPLOYMENT HARDWARE

- 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



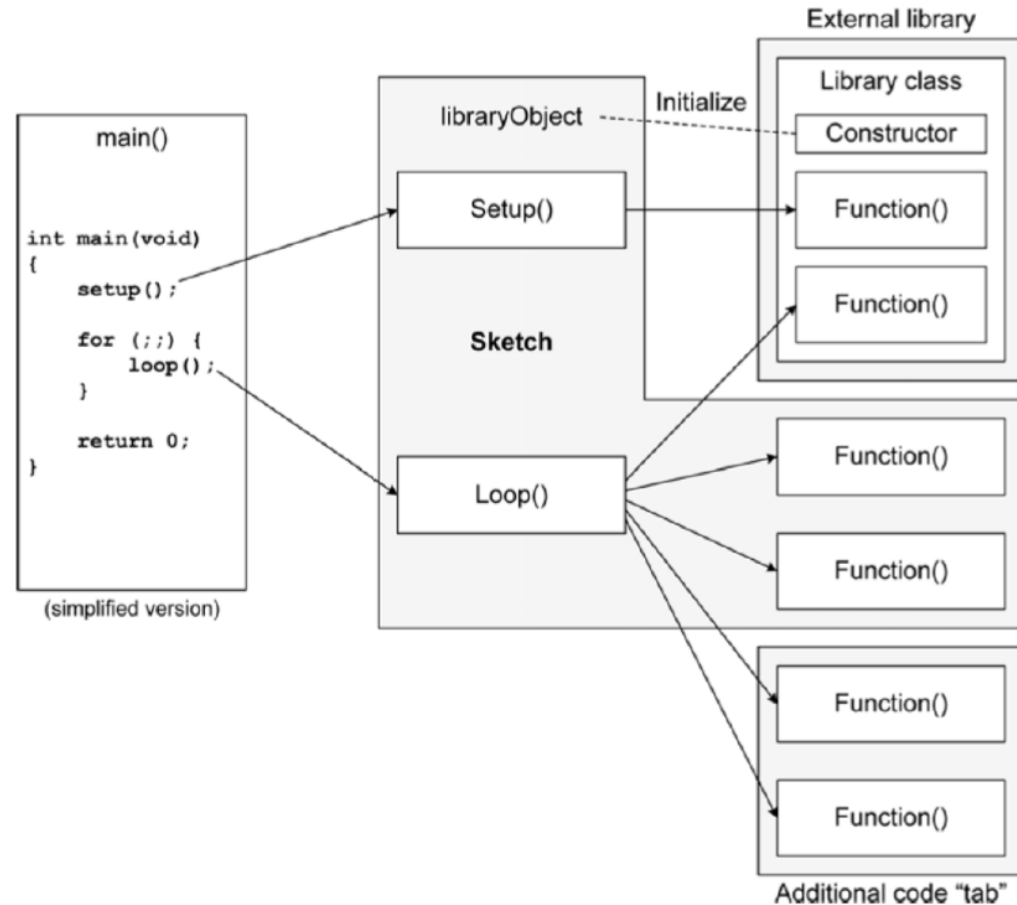


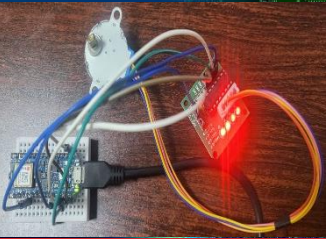
MODEL DEPLOYMENT HARDWARE - ARDUINO

❖ 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();  
  
#ifdef(USBCON)  
  USBDevice.attach();  
#endif  
  
  setup();  
  for (;;) {  
    loop();  
    if (serialEventRun) serialEventRun();  
  }  
  return 0;  
}
```



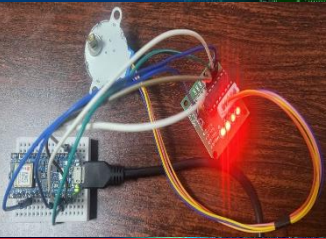


MODEL DEPLOYMENT HARDWARE

❖ Model Deployment

- Impulse design
 - Create impulse
 - MFCC
 - Classifier
- EON Tuner
- Retrain model
- Live classification
- Model testing
- Performance calibration
- Versioning
- Deployment**





MODEL DEPLOYMENT HARDWARE

❖ 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 x

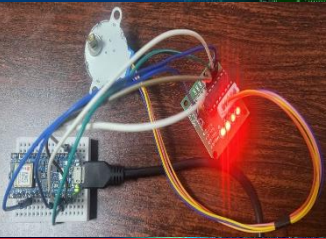


SELECTED DEPLOYMENT

Arduino library

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





MODEL DEPLOYMENT HARDWARE

❖ Model Deployment

MODEL OPTIMIZATIONS

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



EON™ Compiler

Same accuracy, 19% less RAM, 33% less ROM.

Quantized
(int8)

Selected ✓

	MFE	CLASSIFIER	TOTAL
LATENCY	1,521 ms.	91 ms.	1,612 ms.
RAM	83.7K	41.8K	83.7K
FLASH	-	41.6K	-
ACCURACY			92.18%

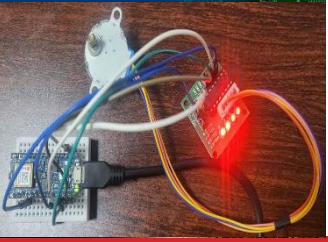
Unoptimized
(float32)

Select

	MFE	CLASSIFIER	TOTAL
LATENCY	1,521 ms.	1,485 ms.	3,006 ms.
RAM	83.7K	157.9K	157.9K
FLASH	-	67.1K	-
ACCURACY			92.18%

Estimate for Arduino Nano 33 BLE Sense (Cortex-M4F 64MHz) - [Change target](#)

Build



MODEL DEPLOYMENT HARDWARE

❖ Model Deployment



Built Arduino library

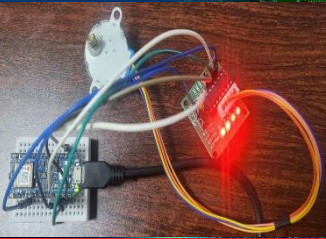
Add this library through the Arduino IDE via:

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

Examples can then be found under:

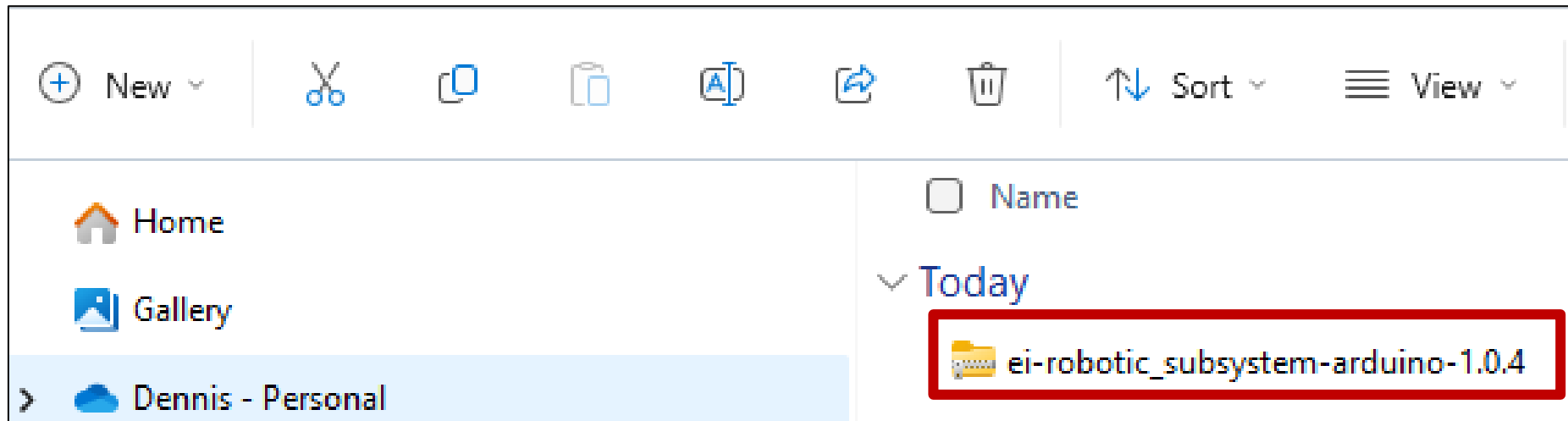
`File > Examples > Robotic_SubSystem_inferencing`

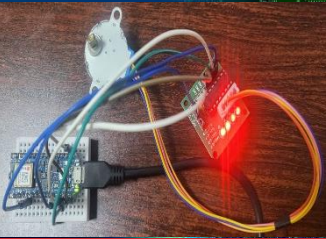




MODEL DEPLOYMENT HARDWARE

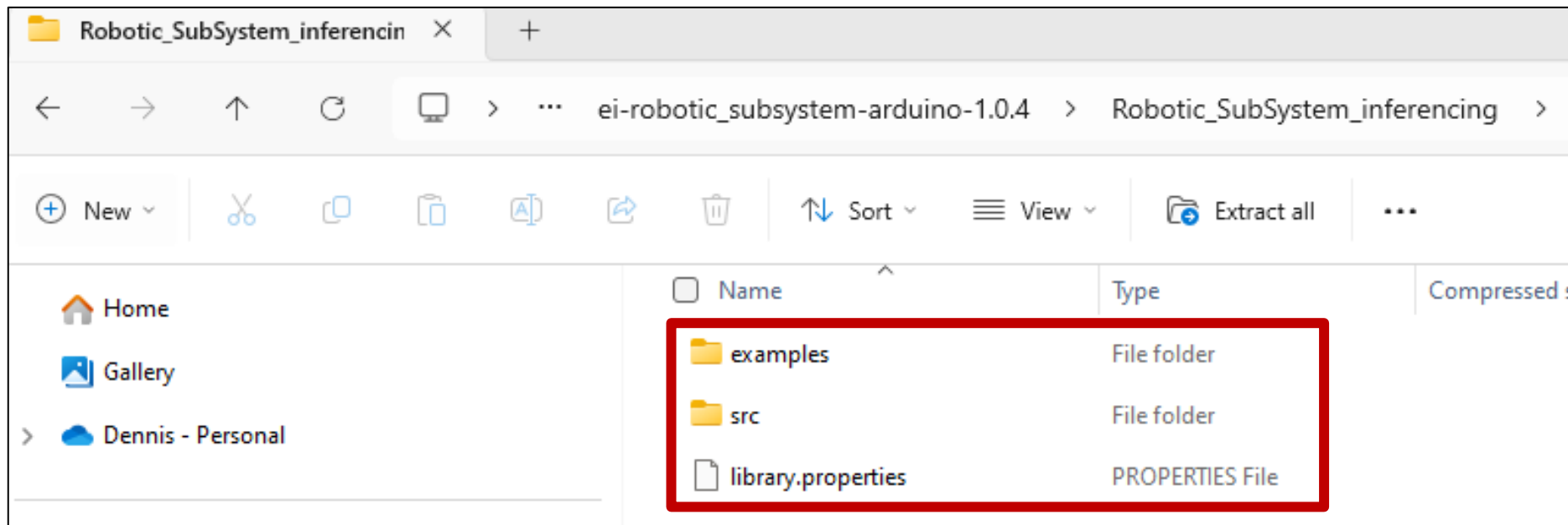
❖ Generated files

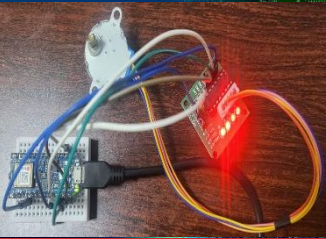




MODEL DEPLOYMENT HARDWARE

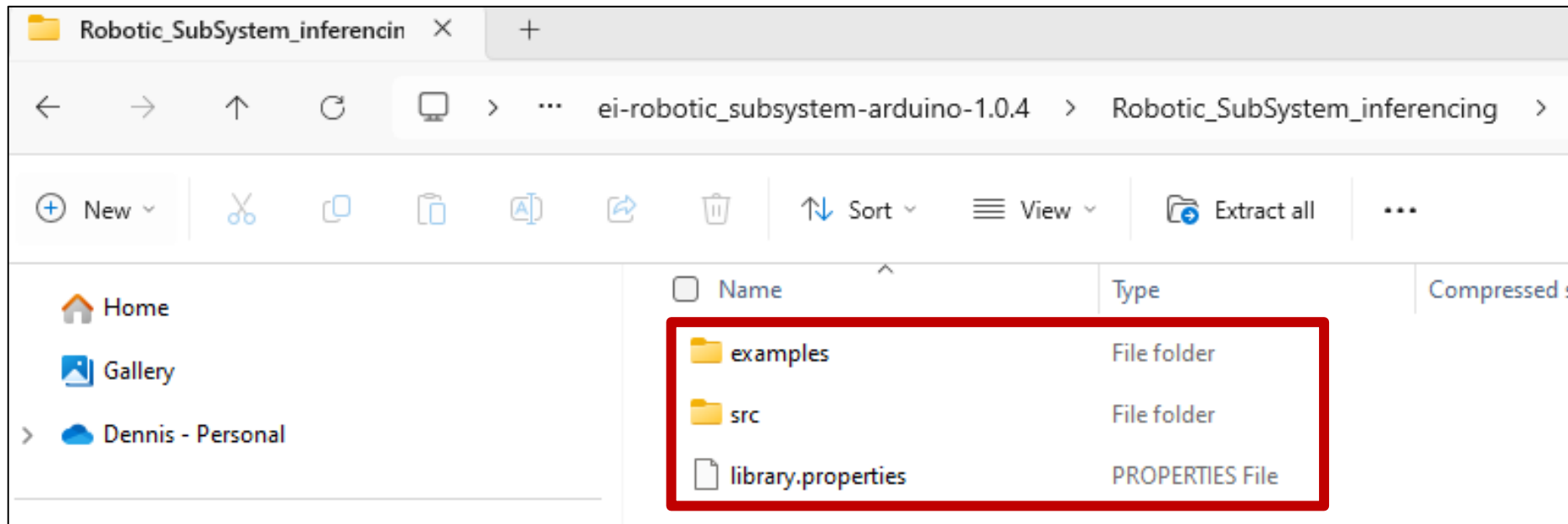
❖ Generated files

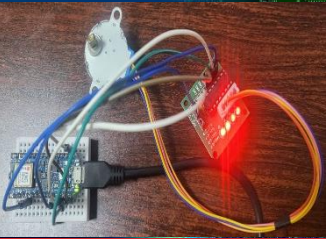




MODEL DEPLOYMENT HARDWARE

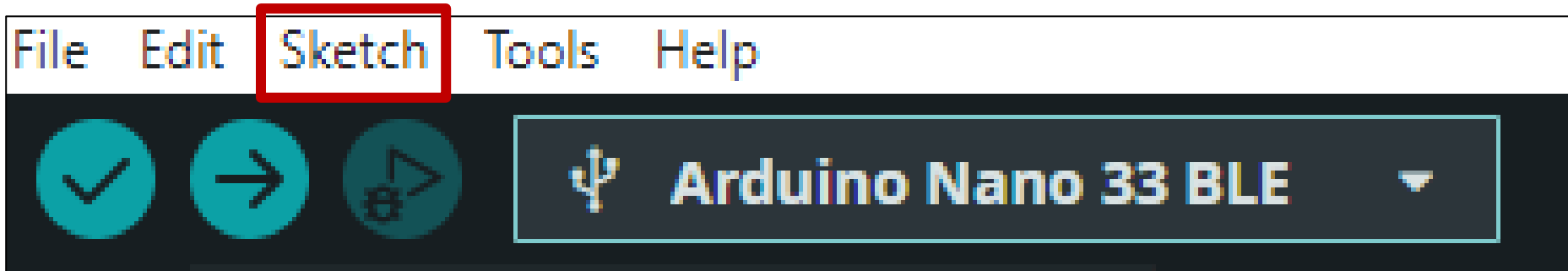
❖ Generated files

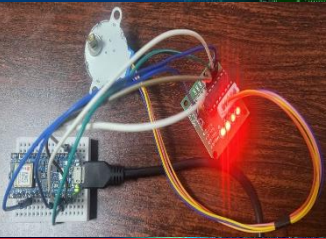




MODEL DEPLOYMENT HARDWARE

❖ Arduino





MODEL DEPLOYMENT HARDWARE

❖ Arduino

le33_sense_microphone | Arduino IDE 2.2.1

Sketch Tools Help

Verify/Compile Ctrl+R

Upload Ctrl+U

Configure and Upload

Upload Using Programmer Ctrl+Shift+U

Export Compiled Binary Alt+Ctrl+S

Optimize for Debugging

Show Sketch Folder Alt+Ctrl+K

Include Library

Add File...

Manage Libraries... Ctrl+Shift+I

Add .ZIP Library...

Contributed libraries

Adafruit ADT7410 Library

Adafruit APDS9960 Library

Adafruit Arcada Library

Adafruit BMP280 Library

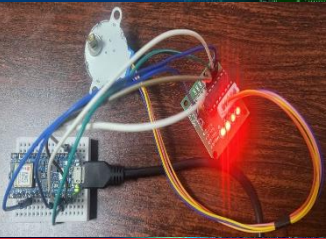
Adafruit BusIO

Adafruit Circuit Playground

Adafruit EPD

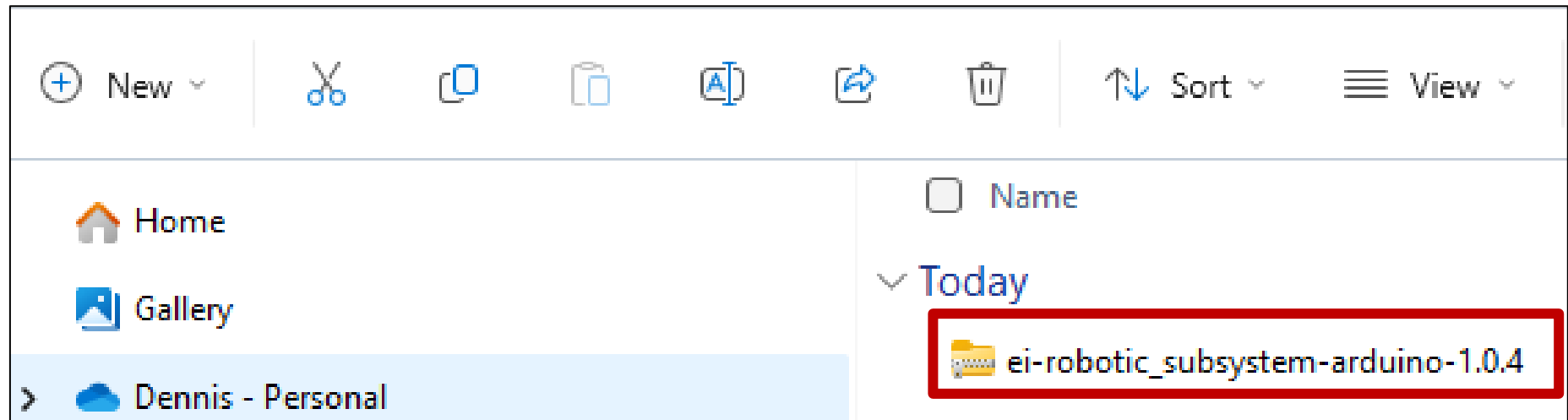
Adafruit FreeTouch Library

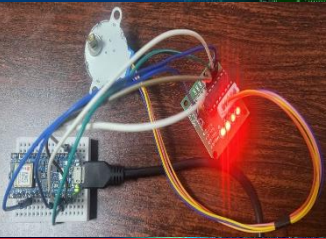




MODEL DEPLOYMENT HARDWARE

❖ Arduino





MODEL DEPLOYMENT HARDWARE

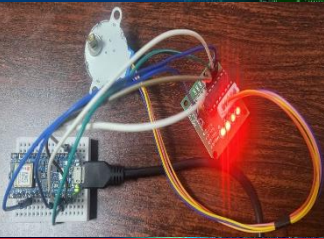
❖ Arduino

File Edit Sketch Tools Help



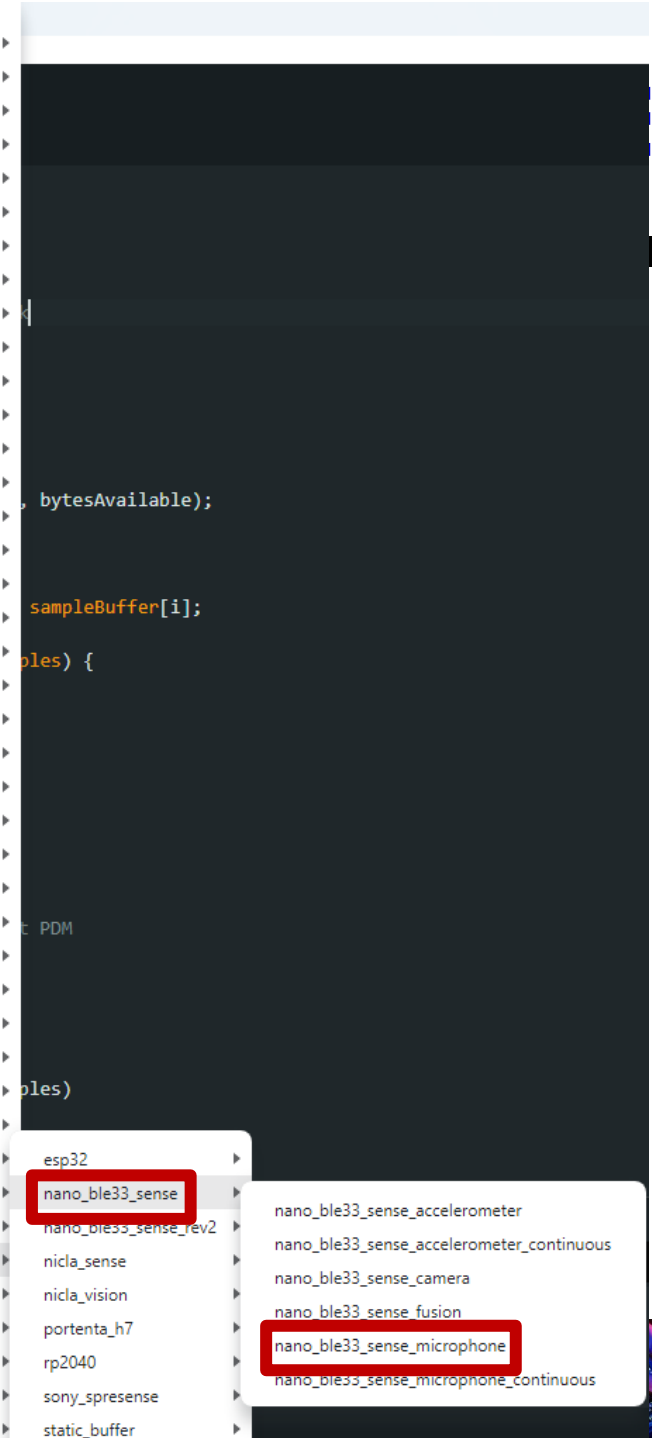
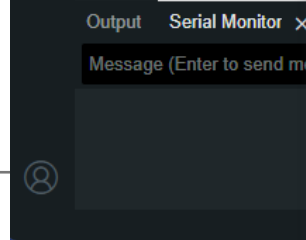
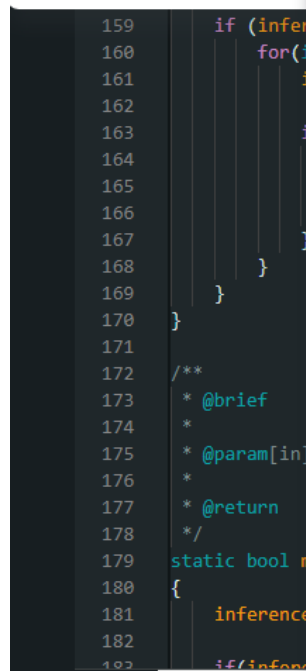
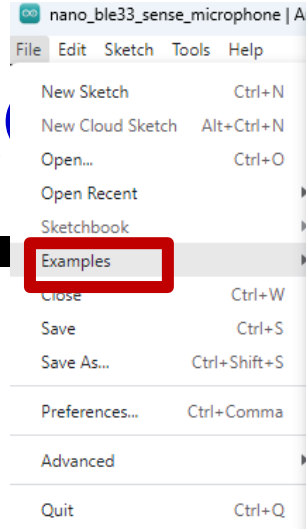
Arduino Nano 33 BLE

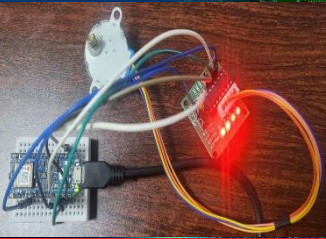




M

❖ Arduino





MODEL DEPLOYMENT HARDWARE

❖ Arduino code for spotting keywords (1)

```

17 // If your target is limited in memory remove this macro to save 10K RAM
18 #define EIDSP_QUANTIZE_FILTERBANK 0
19
20 /*
21 ** NOTE: If you run into TFLite arena allocation issue.
22 **
23 ** This may be due to may dynamic memory fragmentation.
24 ** Try defining "-DEI_CLASSIFIER_ALLOCATION_STATIC" in boards.local.txt (create
25 ** if it doesn't exist) and copy this file to
26 ** `<ARDUINO_CORE_INSTALL_PATH>/arduino/hardware/<mbd_core>/<core_version>`.
27 **
28 ** See
29 ** (https://support.arduino.cc/hc/en-us/articles/360012076960-Where-are-the-installed-cores-located-)
30 ** to find where Arduino installs cores on your machine.
31 **
32 ** If the problem persists then there's not enough memory for this model and application.
33 */
34
35 /* Includes ----- */
36 #include <PDM.h>
37 #include <Robotic_SubSystem_inferencing.h>
38
39 /** Audio buffers, pointers and selectors */
40 typedef struct {
41     int16_t *buffer;
42     uint8_t buf_ready;
43     uint32_t buf_count;
44     uint32_t n_samples;
45 } inference_t;
46
47 static inference_t inference;
48 static signed short sampleBuffer[2048];
49 static bool debug_nn = false; // Set this to true to see e.g. features generated from the raw signal
50

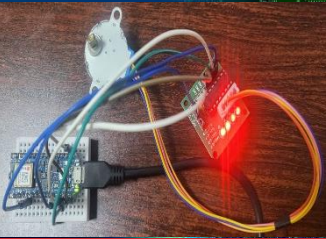
```

```

51 /**
52  * @brief Arduino setup function
53  */
54 void setup()
55 {
56     // put your setup code here, to run once:
57     Serial.begin(115200);
58     // comment out the below line to cancel the wait for USB connection (needed for native USB)
59     while (!Serial);
60     Serial.println("Edge Impulse Inferencing Demo");
61
62     // summary of inferencing settings (from model_metadata.h)
63     ei_printf("Inferencing settings:\n");
64     ei_printf("\tInterval: %.2f ms.\n", (float)EI_CLASSIFIER_INTERVAL_MS);
65     ei_printf("\tFrame size: %d\n", EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE);
66     ei_printf("\tSample length: %d ms.\n", EI_CLASSIFIER_RAW_SAMPLE_COUNT / 10);
67     ei_printf("\tNo. of classes: %d\n", sizeof(ei_classifier_inferencing_categories) / sizeof(ei_classifier_inferencing_categories[0]));
68
69     if (microphone_inference_start(EI_CLASSIFIER_RAW_SAMPLE_COUNT) == false) {
70         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);
71         return;
72     }
73
74

```





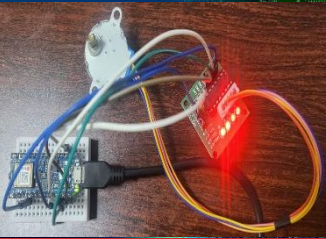
MODEL DEPLOYMENT HARDWARE

❖ Arduino code for spotting keywords (2)

```
nano_ble33_sense_microphone.ino
//
78 void loop()
79 {
80     ei_printf("Starting inferencing in 2 seconds...\n");
81
82     delay(2000);
83
84     ei_printf("Recording...\n");
85
86     bool m = microphone_inference_record();
87     if (!m) {
88         ei_printf("ERR: Failed to record audio...\n");
89         return;
90     }
91
92     ei_printf("Recording done\n");
93
94     signal_t signal;
95     signal.total_length = EI_CLASSIFIER_RAW_SAMPLE_COUNT;
96     signal.get_data = &microphone_audio_signal_get_data;
97     ei_impulse_result_t result = { 0 };
98
99     EI_IMPULSE_ERROR r = run_classifier(&signal, &result, debug_nn);
100     if (r != EI_IMPULSE_OK) {
101         ei_printf("ERR: Failed to run classifier (%d)\n", r);
102         return;
103     }
104
105     // print the predictions
106     ei_printf("Predictions ");
107     ei_printf("(DSP: %d ms., Classification: %d ms., Anomaly: %d ms.)",
108             result.timing.dsp, result.timing.classification, result.timing.anomaly);
109     ei_printf("\n");
110     for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
111         ei_printf("    %s: %.5f\n", result.classification[ix].label, result.classification[ix].value);
112     }
113     #if EI_CLASSIFIER_HAS_ANOMALY == 1
114         ei_printf("    anomaly score: %.3f\n", result.anomaly);
115     #endif
116 }
```

```
142 /**
143  * @brief      Init inferencing struct and setup/start PDM
144  *
145  * @param[in]  n_samples  The n samples
146  *
147  * @return     { description_of_the_return_value }
148  */
149 static bool microphone_inference_start(uint32_t n_samples)
150 {
151     inference.buffer = (int16_t *)malloc(n_samples * sizeof(int16_t));
152
153     if(inference.buffer == NULL) {
154         return false;
155     }
156
157     inference.buf_count = 0;
158     inference.n_samples = n_samples;
159     inference.buf_ready = 0;
160
161     // configure the data receive callback
162     PDM.onReceive(&pdm_data_ready_inference_callback);
163
164     PDM.setBufferSize(4096);
165
166     // initialize PDM with:
167     // - one channel (mono mode)
168     // - a 16 kHz sample rate
169     if (!PDM.begin(1, EI_CLASSIFIER_FREQUENCY)) {
170         ei_printf("Failed to start PDM!");
171         microphone_inference_end();
172
173         return false;
174     }
175
176     // set the gain, defaults to 20
177     PDM.setGain(127);
178
179     return true;
180 }
```



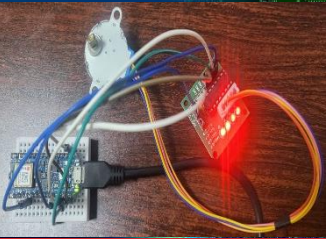


MODEL DEPLOYMENT HARDWARE

❖ Arduino code for spotting keywords (3)

```
182  /**
183   * @brief      Wait on new data
184   *
185   * @return     True when finished
186   */
187  static bool microphone_inference_record(void)
188  {
189      inference.buf_ready = 0;
190      inference.buf_count = 0;
191
192      while(inference.buf_ready == 0) {
193          delay(10);
194      }
195
196      return true;
197  }
198
199  /**
200   * Get raw audio signal data
201   */
202  static int microphone_audio_signal_get_data(size_t offset, size_t length, float *out_ptr)
203  {
204      numpy::int16_to_float(&inference.buffer[offset], out_ptr, length);
205
206      return 0;
207  }
208
209  /**
210   * @brief      Stop PDM and release buffers
211   */
212  static void microphone_inference_end(void)
213  {
214      PDM.end();
215      free(inference.buffer);
216  }
217
218  #if !defined(EI_CLASSIFIER_SENSOR) || EI_CLASSIFIER_SENSOR != EI_CLASSIFIER_SENSOR_MICROPHONE
219  #error "Invalid model for current sensor."
220  #endif
```

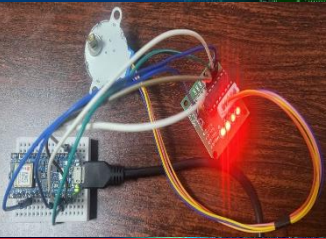




MODEL DEPLOYMENT HARDWARE

- ❖ 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

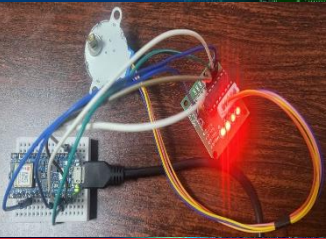
- ❖ Arduino code for spotting keywords (Explanation)
 - Preprocessor Directives and Includes

```
#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

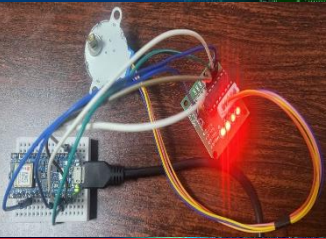
❖ Arduino code for spotting keywords (Explanation)

□ 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

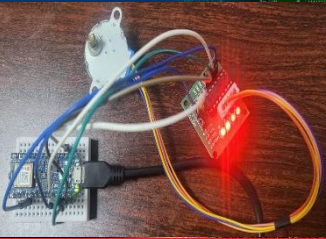
❖ Arduino code for spotting keywords (Explanation)

□ 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) / sizeof(ei_classifier_inferencing_category));  
  
    if (microphone_inference_start(EI_CLASSIFIER_RAW_SAMPLE_COUNT) == false) {  
        ei_printf("ERR: Could not allocate audio buffer (size %d)\n", EI_CLASSIFIER_RAW_SAMPLE_COUNT);  
        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

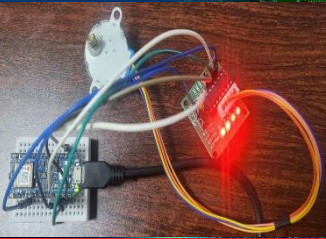
```
void loop() {
    ei_printf("Starting inferencing in 2 seconds...\n");
    delay(2000);
    ei_printf("Recording...\n");

    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 DEPLOYMENT

❖ 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; 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);

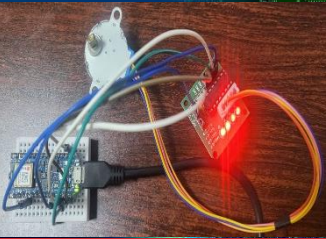
    if (!PDM.begin(1, EI_CLASSIFIER_FREQUENCY)) {
        ei_printf("Failed to start PDM!");
        microphone_inference_end();
        return false;
    }

    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);
}
```



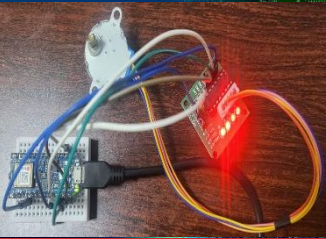
MODEL DEPLOYMENT HARDWARE

- ❖ 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

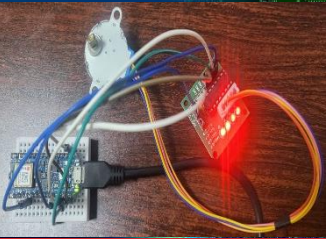




MODEL DEPLOYMENT HARDWARE

- ❖ 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





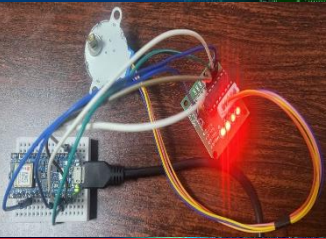
MODEL DEPLOYMENT HARDWARE

❖ 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...
```



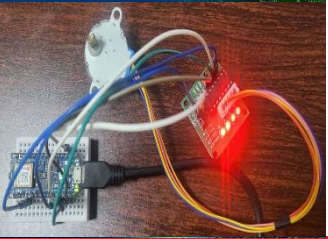


MODEL DEPLOYMENT HARDWARE

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

```
36  #include <PDM.h>
37  #include <Robotic_SubSystem_inferencing.h>
38
39  //ADDED CODE
40  //Includes the Arduino Stepper Library
41  #include <Stepper.h>
42  // Defines the number of steps per rotation
43  const int stepsPerRevolution = 2048;
44  // Creates an instance of stepper class
45  // Pins entered in sequence IN1-IN3-IN2-IN4 for proper step sequence
46  Stepper myStepper = Stepper(stepsPerRevolution, 8, 10, 9, 11);
47  //END ADDED CODE
```



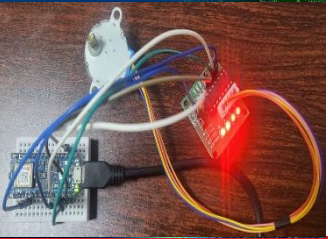


MODEL DEPLOYMENT HARDWARE

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

```
115 //ADDED CODE
116 if(result.classification[0].value > 0.9){
117     ei_printf("BACKWARD");
118     // Rotate CW at 10 RPM
119     myStepper.setSpeed(10);
120     myStepper.step(stepsPerRevolution);
121     delay(1000);
122 }
123 else if(result.classification[1].value > 0.9){
124     ei_printf("FORWARD");
125     // Rotate CCW at 10 RPM
126     myStepper.setSpeed(10);
127     myStepper.step(-stepsPerRevolution);
128     delay(1000);
129 }
```





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
```

