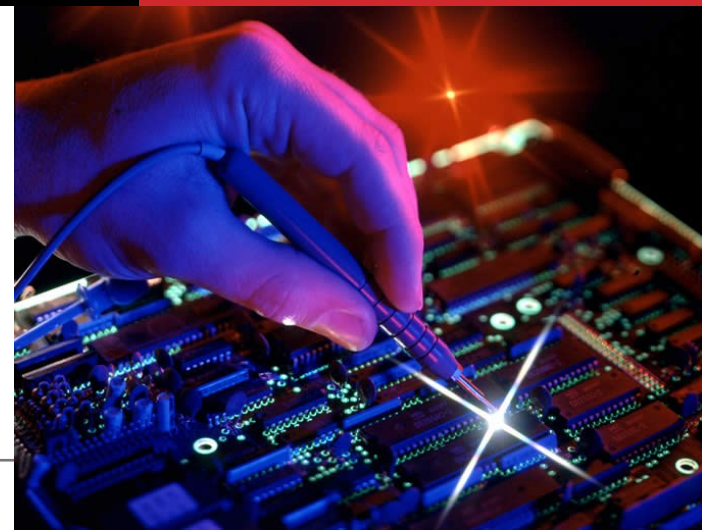
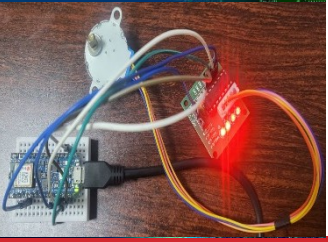


# Robotics

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

Dennis A. N. Gookyi

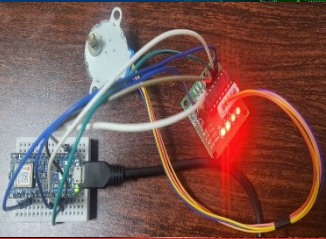




# CONTENTS

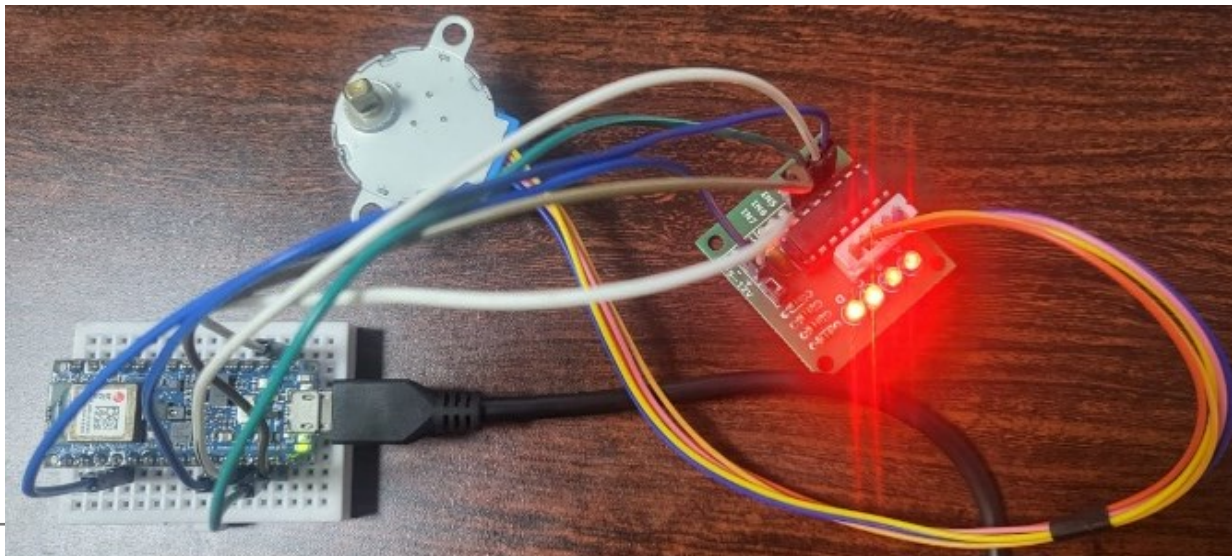
## ❖ Introduction to Edge Impulse

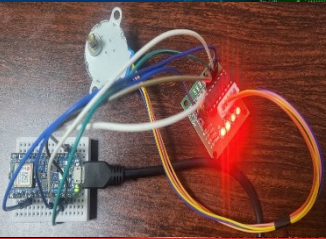




# PROJECT OVERVIEW

- ❖ 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

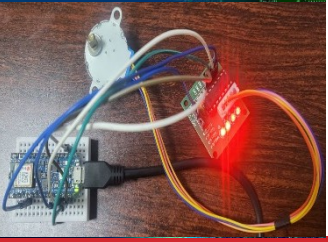




# PROJECT OVERVIEW

- ❖ 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

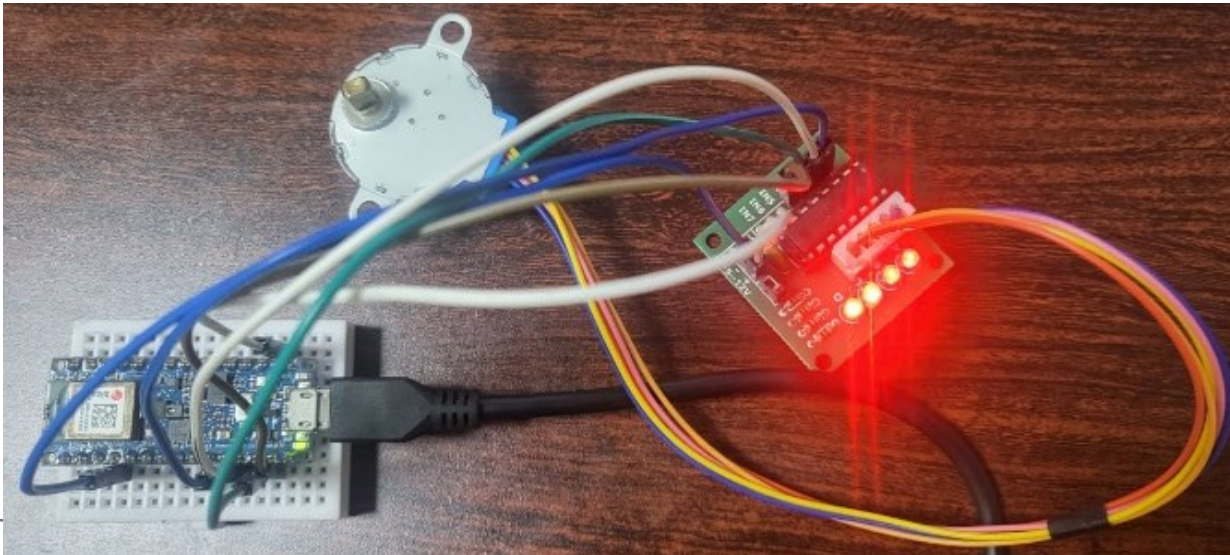




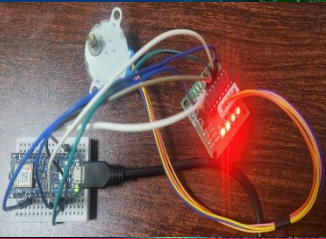
# PROJECT HARDWARE

❖ The hardware components for the project include:

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



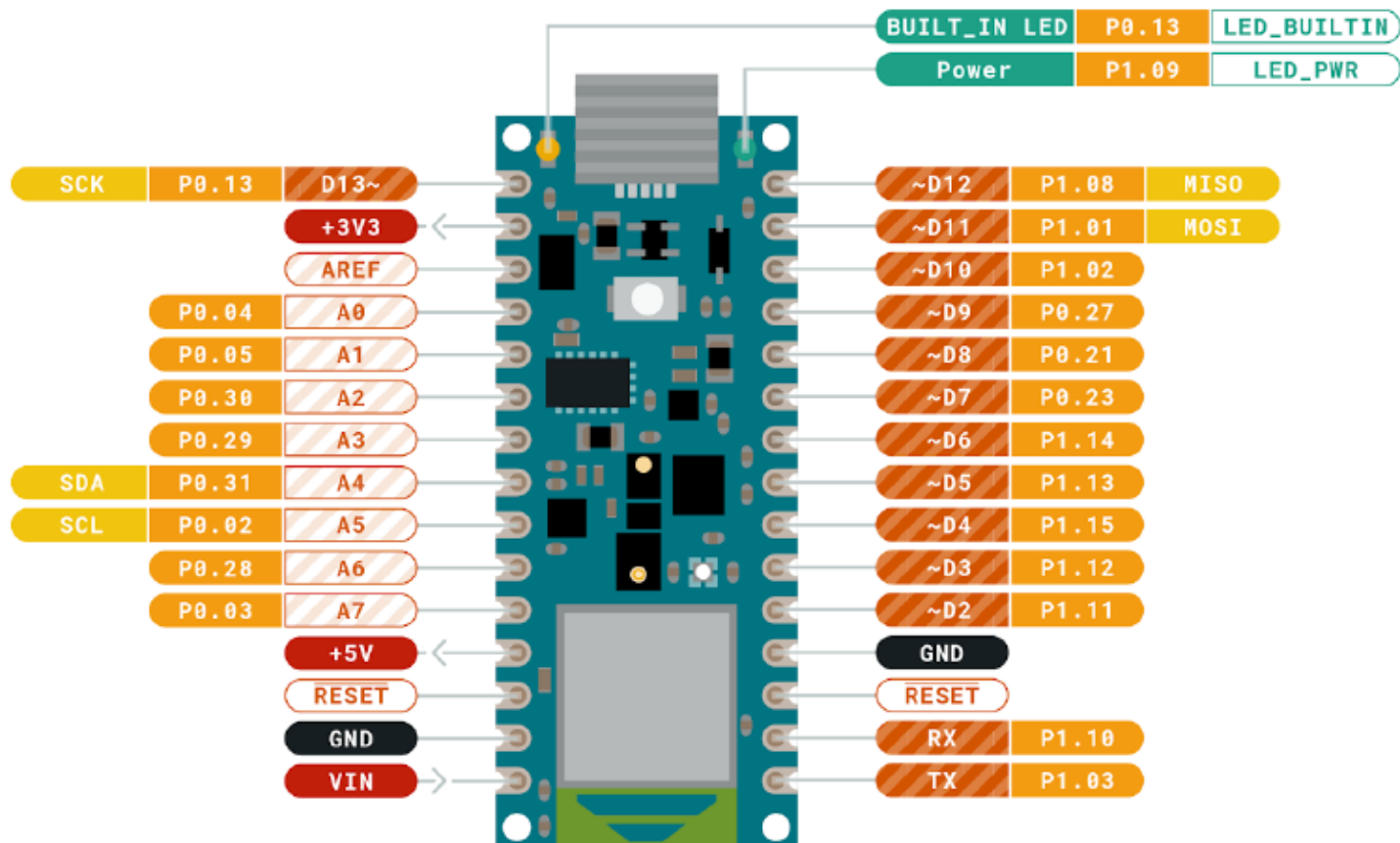


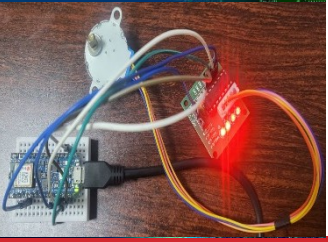


# PROJECT HARDWARE – NANO 33 BLE SENSE

## ❖ Arduino Nano 33 BLE Sense

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

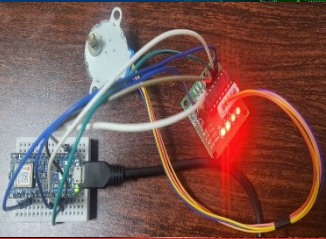




# PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

- ❖ 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



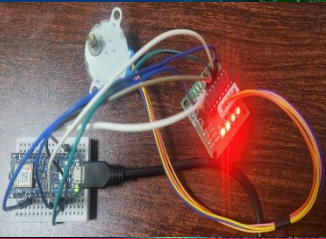


# PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

- ❖ 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

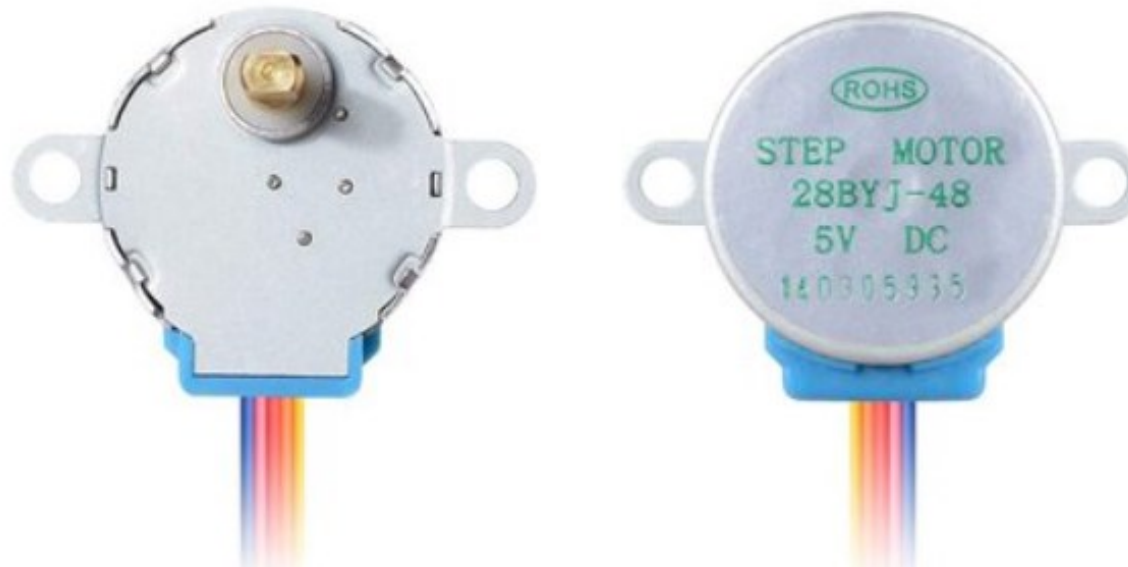


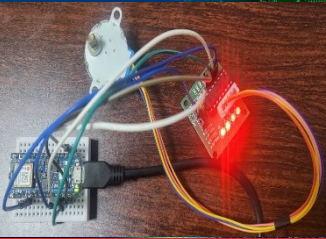




# PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

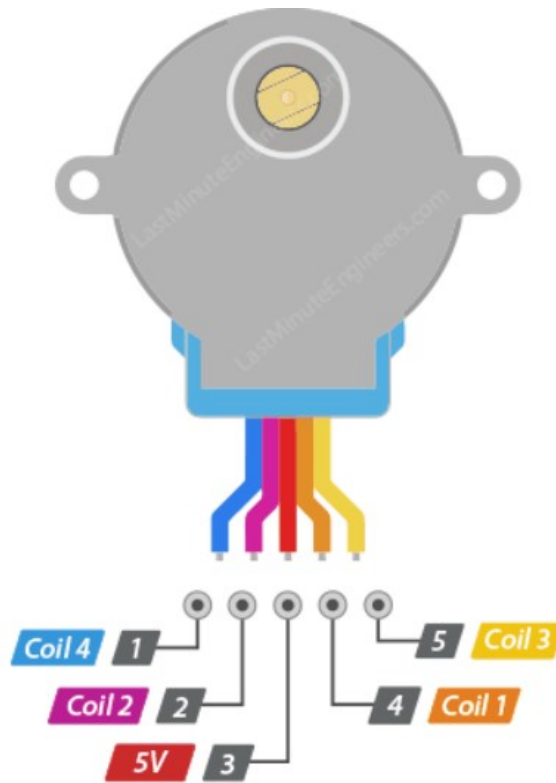
- ❖ Despite its small size, the motor delivers a decent torque of 34.3 mN.m at a speed of around 15 RPM
- ❖ It provides good torque even at a standstill and maintains it as long as the motor receives power
- ❖ The only drawback is that it is somewhat power-hungry and consumes energy even when it is stationary

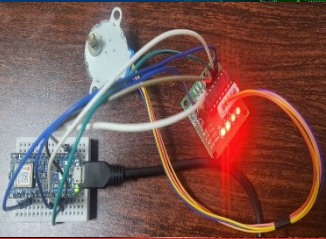




# PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

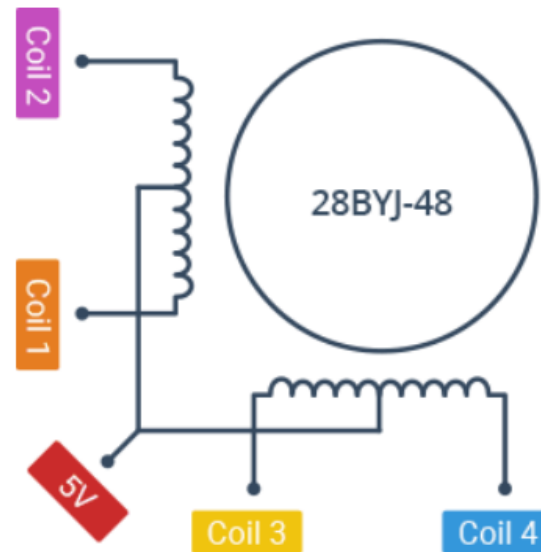
- ❖ The motor has five wires
- ❖ The 28BYJ-48 has two coils, each of which has a center tap
- ❖ These two center taps are connected internally and brought out as the 5th wire (red wire)

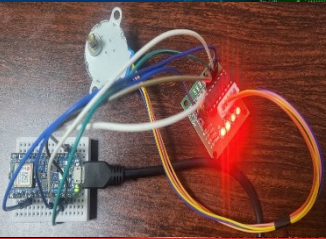




# PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

- ❖ 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

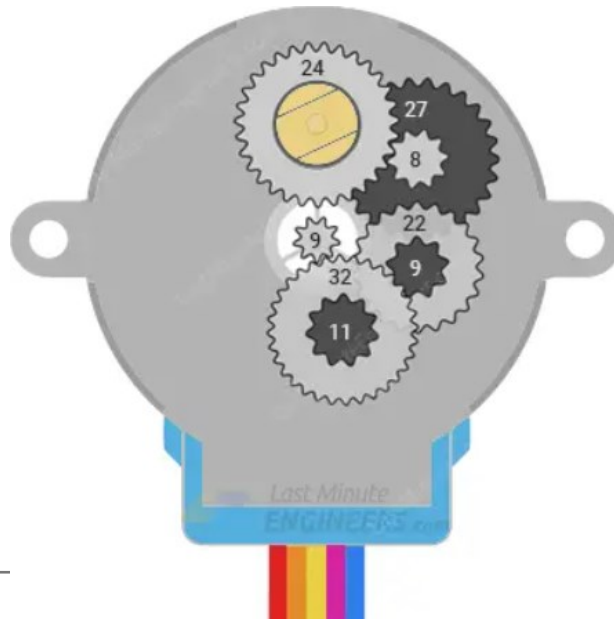




# PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

## ❖ Gear Reduction Ratio

- When the 28BYJ-48 motor is operated in full-step mode, each step corresponds to a rotation of  $11.25^\circ$ 
  - 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 ( $32 \times 64$ ) steps per revolution



Gear Ratios:

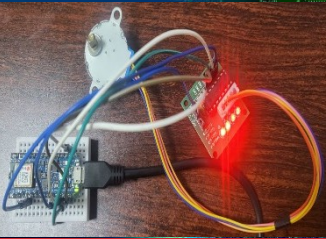
- $32 / 9$
- $22 / 11$
- $27 / 9$
- $24 / 8$

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





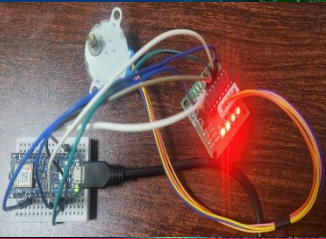
# PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

## ❖ 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





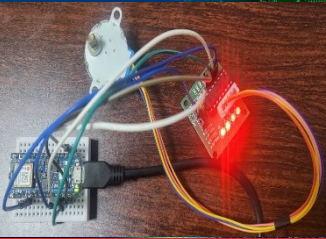


# PROJECT HARDWARE – 28BYJ-48 STEPPER MOTOR

## ❖ 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                        |

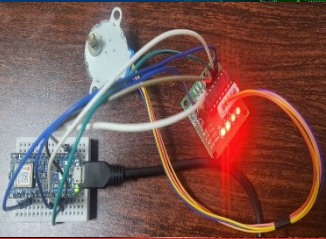




# PROJECT HARDWARE – ULN2003 DRIVER BOARD

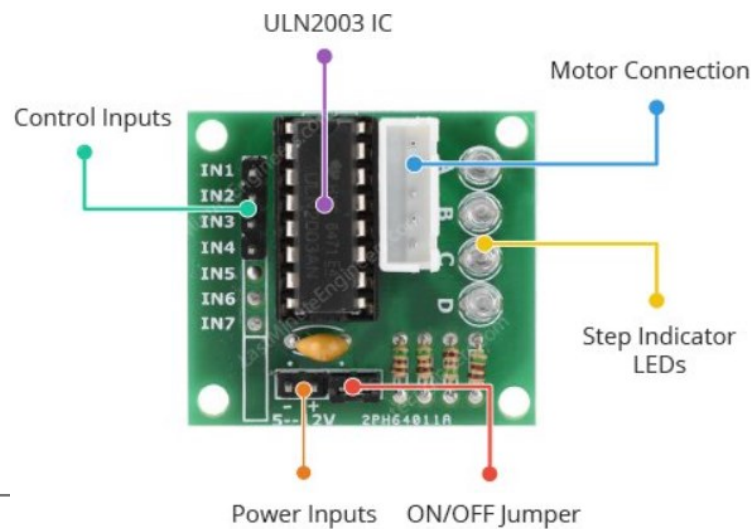
- ❖ Because the 28BYJ-48 stepper motor consumes a significant amount of power, it cannot be controlled directly by a microcontroller such as Arduino
- ❖ To control the motor, a driver IC such as the ULN2003 is required
  - Therefore, this motor typically comes with a ULN2003-based driver board

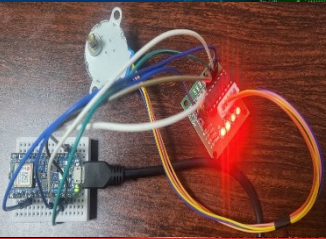




# PROJECT HARDWARE – ULN2003 D RIVER BOARD

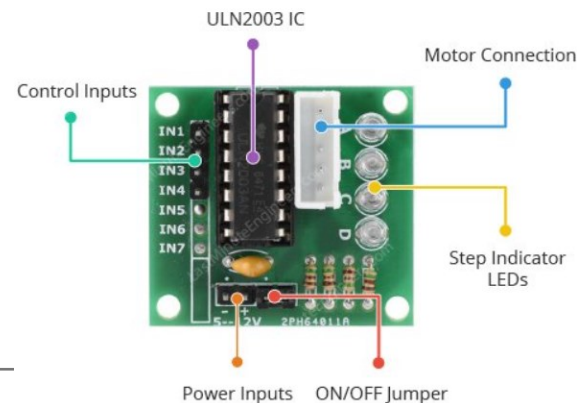
- ❖ The ULN2003, known for its high current and high voltage capability, provides a higher current gain than a single transistor and allows a microcontroller's low voltage low current output to drive a high current stepper motor
- ❖ The ULN2003 consists of an array of seven Darlington transistor pairs,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

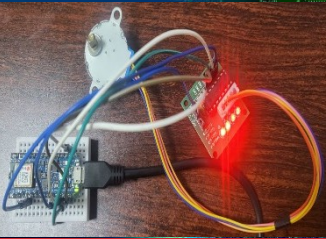




# PROJECT HARDWARE – ULN2003 D RIVER BOARD

- ❖ 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

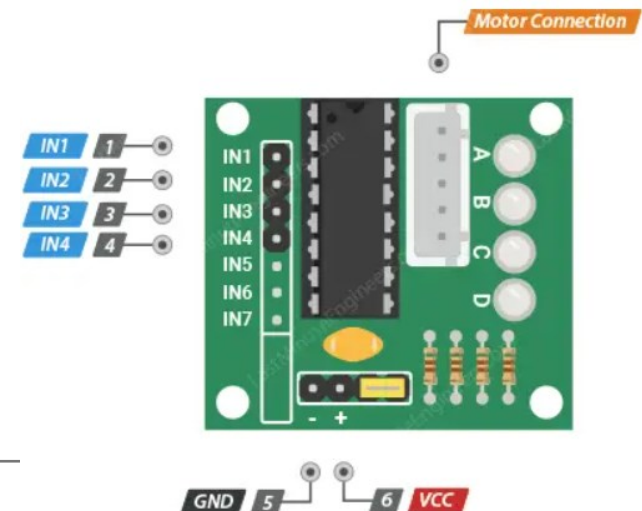




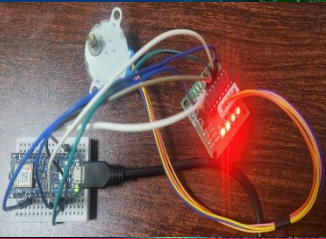
# PROJECT HARDWARE – ULN2003 D RIVER BOARD

## ❖ 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



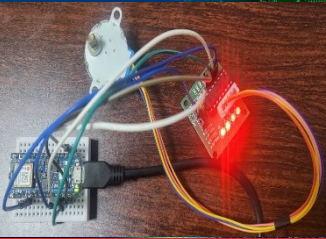




# PROJECT HARDWARE






- ❖ 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

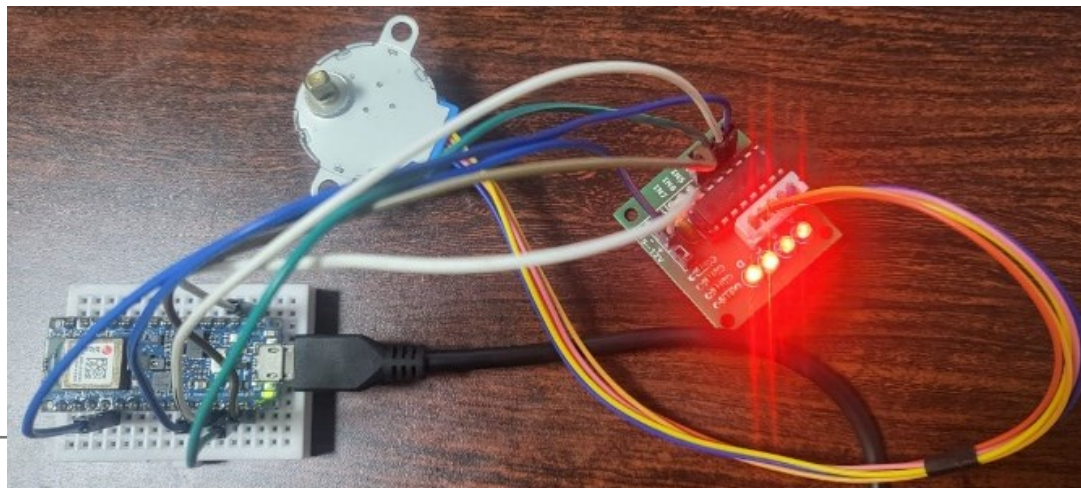


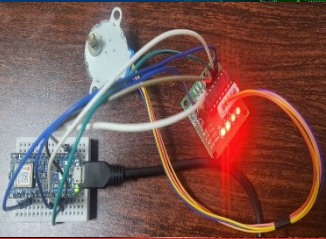


# PROJECT HARDWARE

❖ The following table lists the pin connections:

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

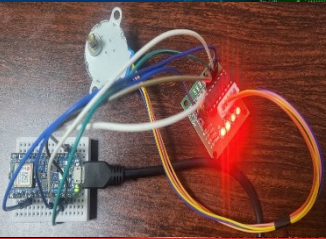




# ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

- ❖ 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



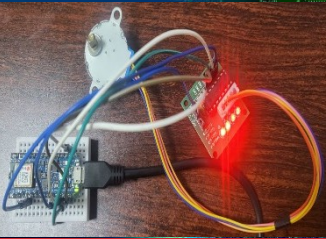


# ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

- ❖ 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

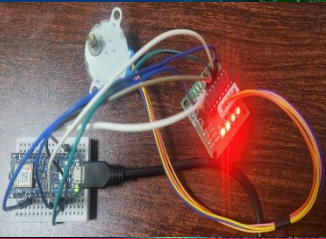
## ❖ Code Explanation:

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

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







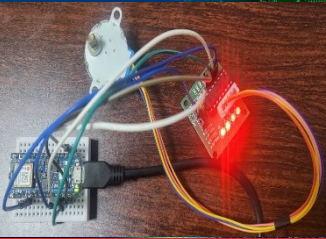
# ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

## ❖ 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;
```





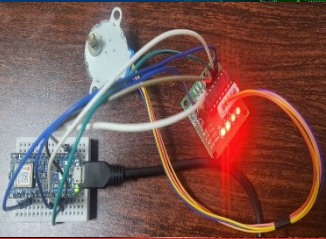
# ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

## ❖ 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);
```





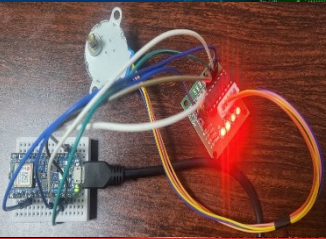
# ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

## ❖ 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 }
```





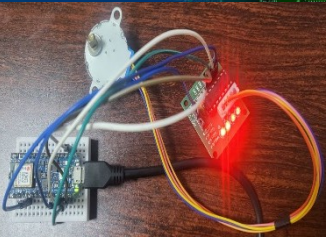
# ARDUINO CODE – USING BUILT-IN STEPPER LIBRARY

## ❖ 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 }
```





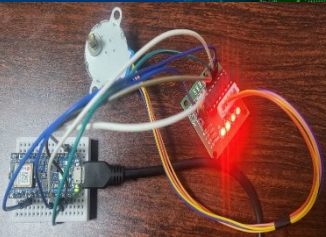
# SPEECH COMMANDS DATASET

## ❖ 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>





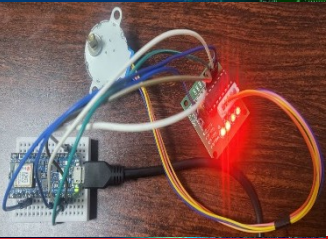


# SPEECH COMMANDS DATASET

## ❖ 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](http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz)





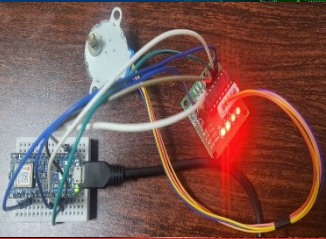
# SPEECH COMMANDS DATASET

## ❖ 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 11<sup>th</sup>, 2018





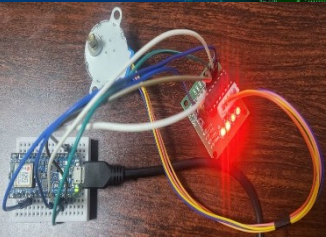
# SPEECH COMMANDS DATASET

## ❖ Speech Commands Data Set v0.02

### □ Collection

- The audio files were collected using crowdsourcing
  - ◇ [aiyprojects.withgoogle.com/open\\_speech\\_recording](https://aiyprojects.withgoogle.com/open_speech_recording)
  - ◇ [https://github.com/petewarden/extract\\_loudest\\_section](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"**



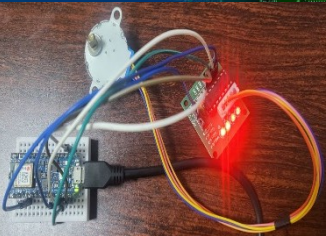


# SPEECH COMMANDS DATASET

## ❖ 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





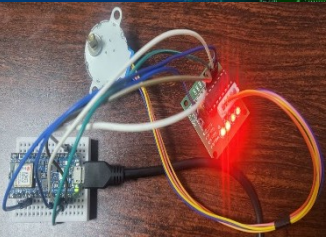
# SPEECH COMMANDS DATASET

## ❖ 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

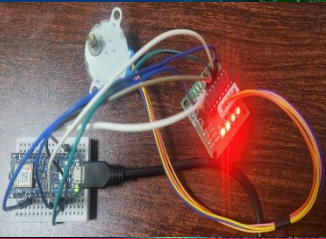
## ❖ Speech Commands Data Set v0.02

### □ 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







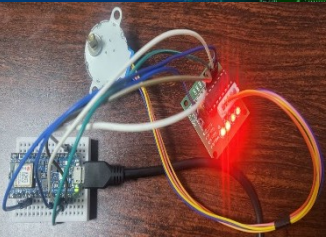
# SPEECH COMMANDS DATASET

## ❖ 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](https://github.com/petewarden/extract_loudest_section)
- The audio files were then screened for silence or incorrect words, and arranged into folders by label





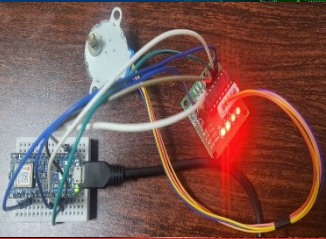
# SPEECH COMMANDS DATASET

## ❖ Speech Commands Data Set v0.02

### □ Background Noise

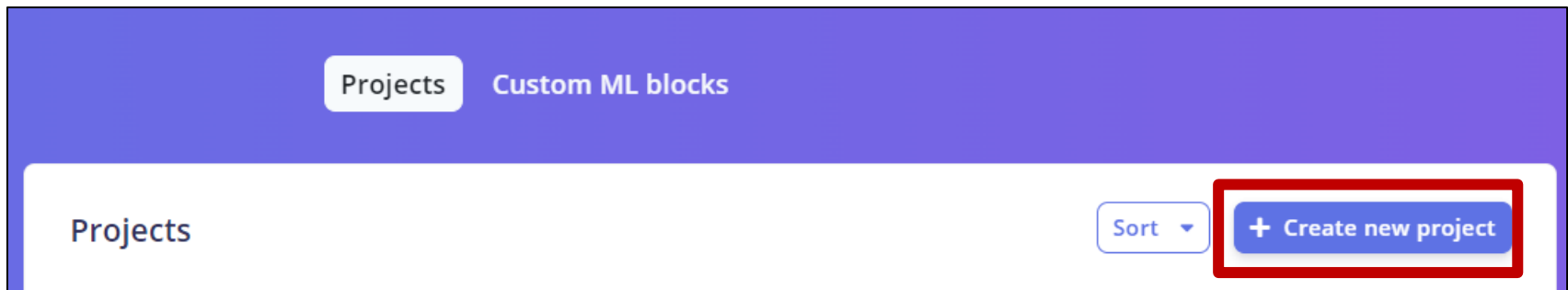
- To help train networks to cope with noisy environments, it can be helpful to mix in realistic background audio
- The ‘\_background\_noise\_’ folder contains a set of longer audio clips that are either recordings or mathematical simulations of noise
- For more details, see the ‘\_background\_noise\_/README.md’

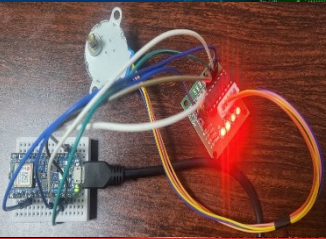




# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

## ❖ 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:

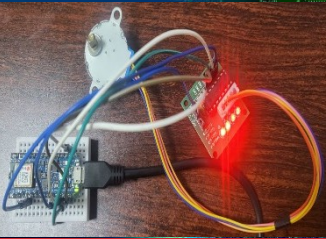
☒ **Public**  
Anyone on the internet can view and clone this project under the licence: [Apache 2.0](#). Only invited users will be able to edit.

☐ **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

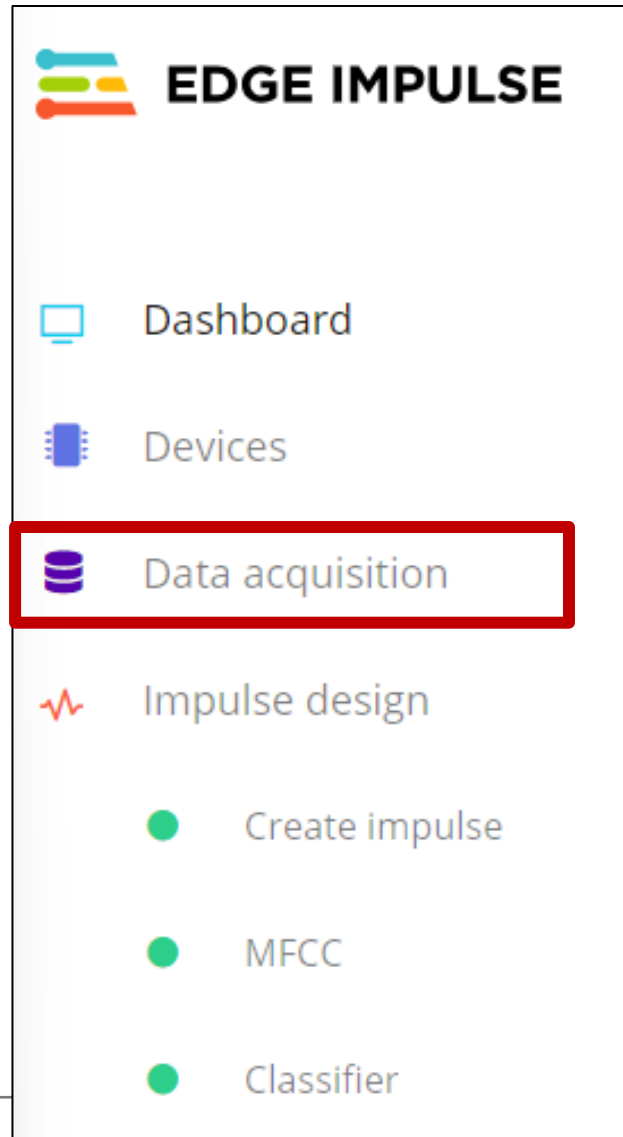


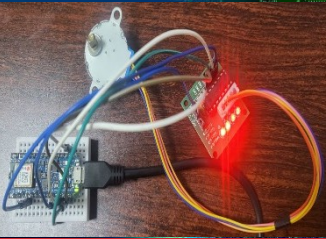


# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

## ❖ Data acquisition

- ☐ Forward
- ☐ Backward
- ☐ Noise/Silence





# 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 ?

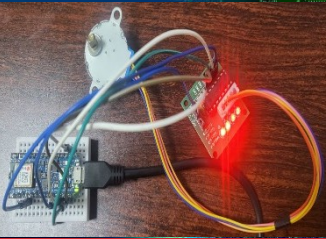
☐ Leave data unlabeled ?

☐ Enter label:

Enter a label

< Back

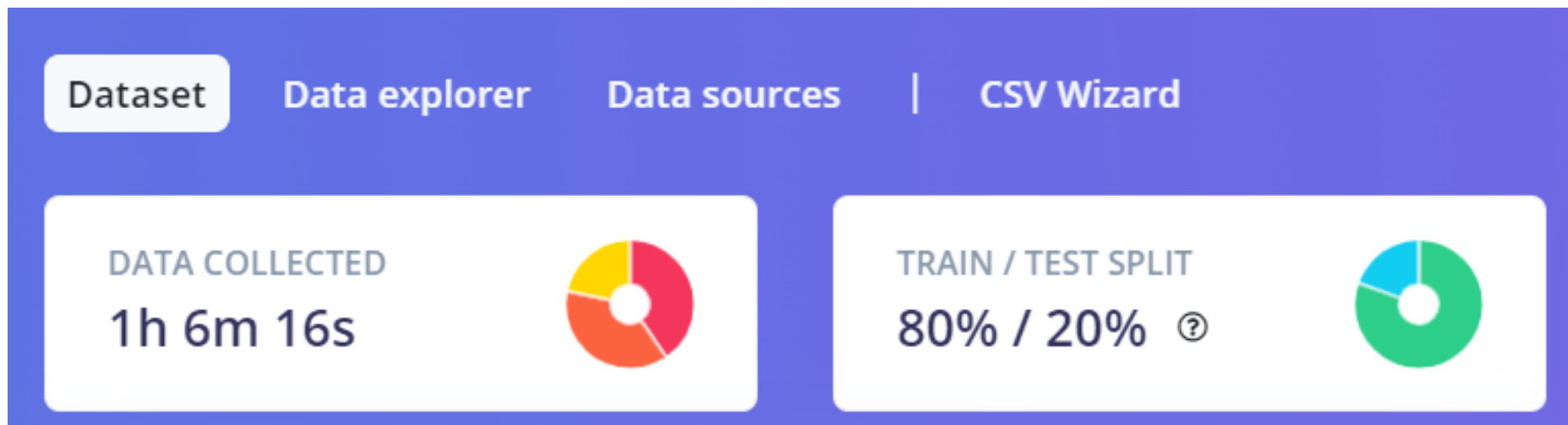
Upload data



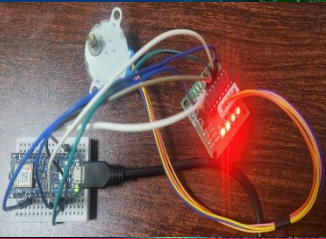
# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

## ❖ Data acquisition

- ☐ Forward
- ☐ Backward
- ☐ Noise/Silence



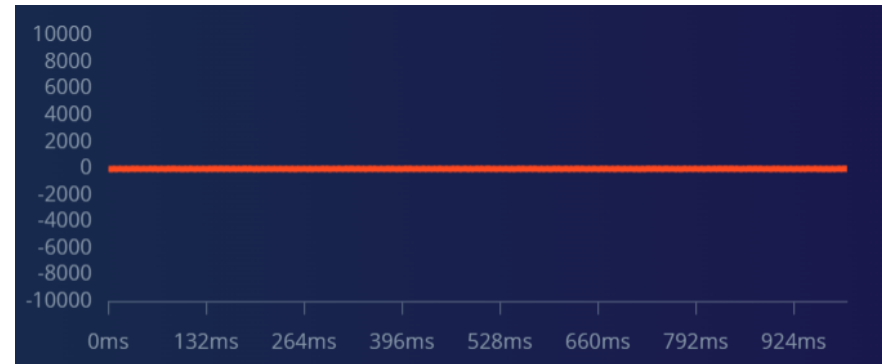




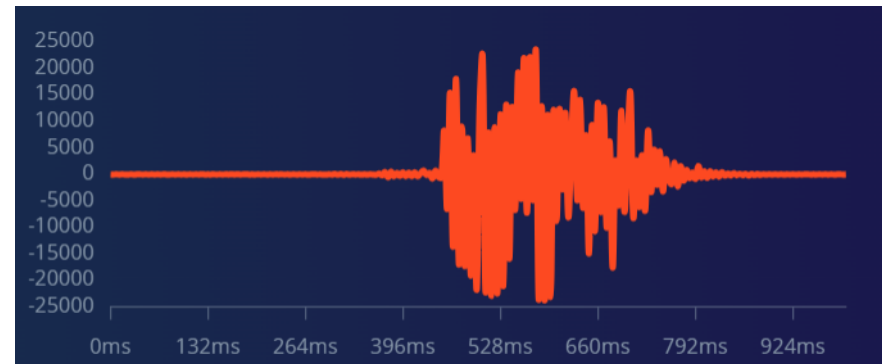
# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

## ❖ Data acquisition

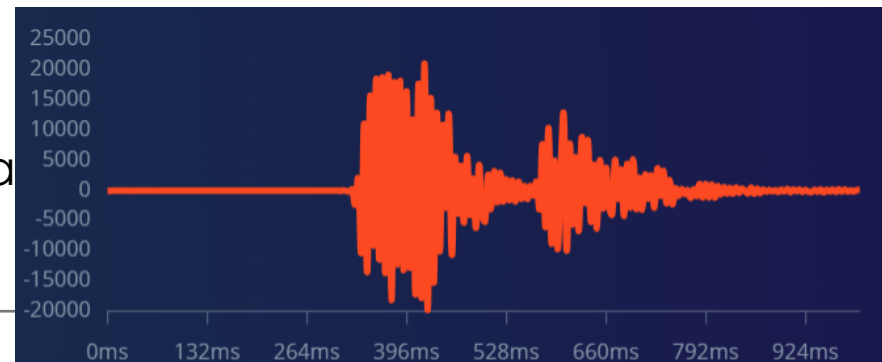
- ☐ Forward
  - ☐ Backward
  - ☐ Noise/Silence
- Silence raw data

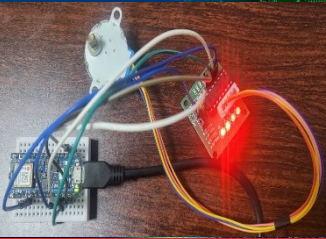


Forward raw data



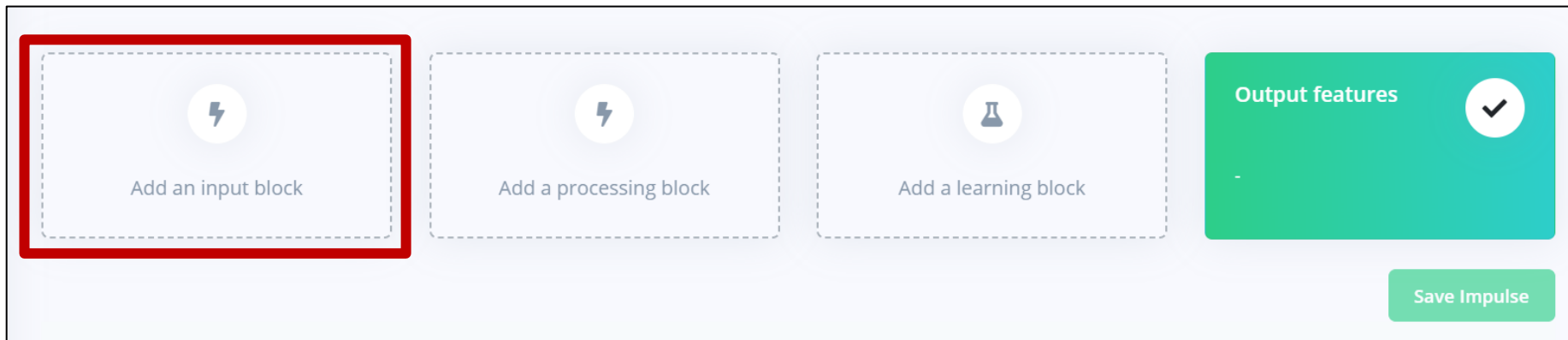
Backward raw data

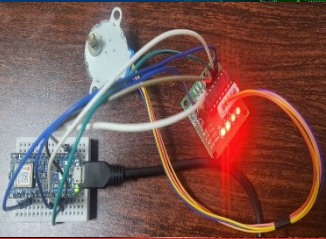




# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE


- ❖ Impulse design
  - Create impulse






# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

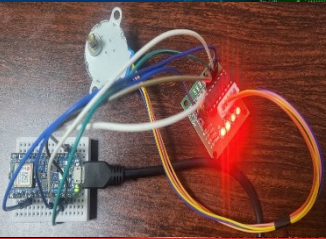
- ❖ Impulse design
  - Create impulse

 Add an input block ×

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

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) 

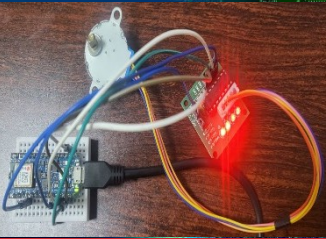


Zero-pad data 

☒



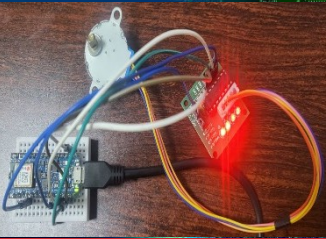


# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- ❖ Impulse design
  - Create impulse


The screenshot displays the Edge Impulse web interface. On the left, the 'Time series data' configuration panel is visible, featuring a database icon and settings for input axes (audio), window size (1,000 ms), window increase (500 ms), frequency (16000 Hz), and zero-pad data (checked). The main workspace contains two dashed boxes: '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 at the bottom right.





# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

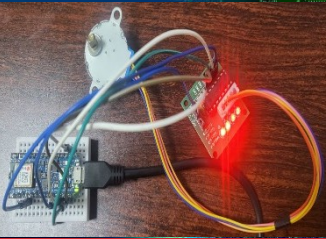
- ❖ Impulse design
  - Create impulse

 Add a processing block ×

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


| DESCRIPTION  | AUTHOR       | RECOMMENDED            |
|--|--------------|------------------------|
| <b>Audio (MFCC)</b> <small>OFFICIALLY SUPPORTED</small><br>Extracts features from audio signals using Mel Frequency Cepstral Coefficients, great for human voice.          | Edge Impulse | ★ <button>Add</button> |
| <b>Audio (MFE)</b> <small>OFFICIALLY SUPPORTED</small><br>Extracts a spectrogram from audio signals using Mel-filterbank energy features, great for non-voice audio.       | Edge Impulse | ★ <button>Add</button> |
| <b>Spectrogram</b> <small>OFFICIALLY SUPPORTED</small><br>Extracts a spectrogram from audio or sensor data, great for non-voice audio or data with continuous frequencies. | Edge Impulse | <button>Add</button>   |
| <b>Audio (Syntiant)</b> <small>OFFICIALLY SUPPORTED</small><br>Syntiant only. Compute log Mel-filterbank energy features from an audio signal.                             | Syntiant     | <button>Add</button>   |
| <b>Raw Data</b> <small>OFFICIALLY SUPPORTED</small><br>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)

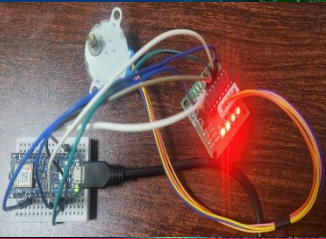
Signal ?

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

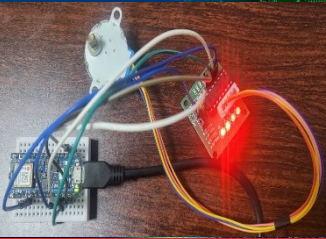
Add a learning block

Output features

-

Save Impulse







# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

## ❖ Impulse design

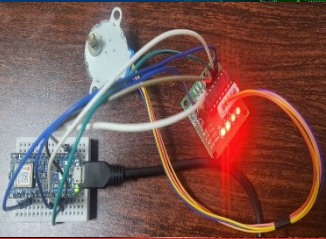
### □ Create impulse

 Add a learning block ×

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

| DESCRIPTION   | AUTHOR       | RECOMMENDED  |
|---|--------------|--|
| <b>Classification</b> <small>OFFICIALLY SUPPORTED</small><br>Learns patterns from data, and can apply these to new data. Great for categorizing movement or recognizing audio.  | Edge Impulse |  <button>Add</button> |
| <b>Regression</b> <small>OFFICIALLY SUPPORTED</small><br>Learns patterns from data, and can apply these to new data. Great for predicting numeric continuous values.  | Edge Impulse | <button>Add</button>   |
| <b>Transfer Learning (Keyword Spotting)</b> <small>OFFICIALLY SUPPORTED</small><br>Fine tune a pre-trained keyword spotting model on your data. Good performance even with relatively small keyword datasets.   | Edge Impulse | <button>Add</button>   |
| <b>Anomaly Detection (GMM)</b> <small>PROFESSIONAL</small> <small>ENTERPRISE</small><br>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

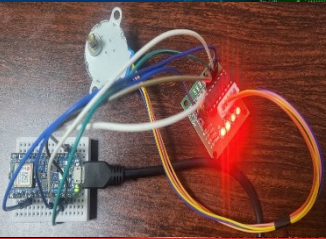
☒ MFCC

### 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

**Classification**

**Name**  
Classifier

**Input features**  
☒ MFCC

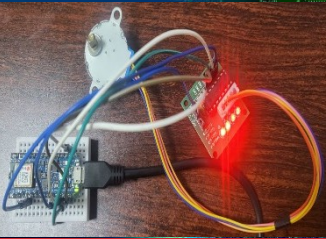
**Output features**  
3 (Backward, Forward, Silence)

**Output features**

3 (Backward, Forward, Silence)

Save Impulse

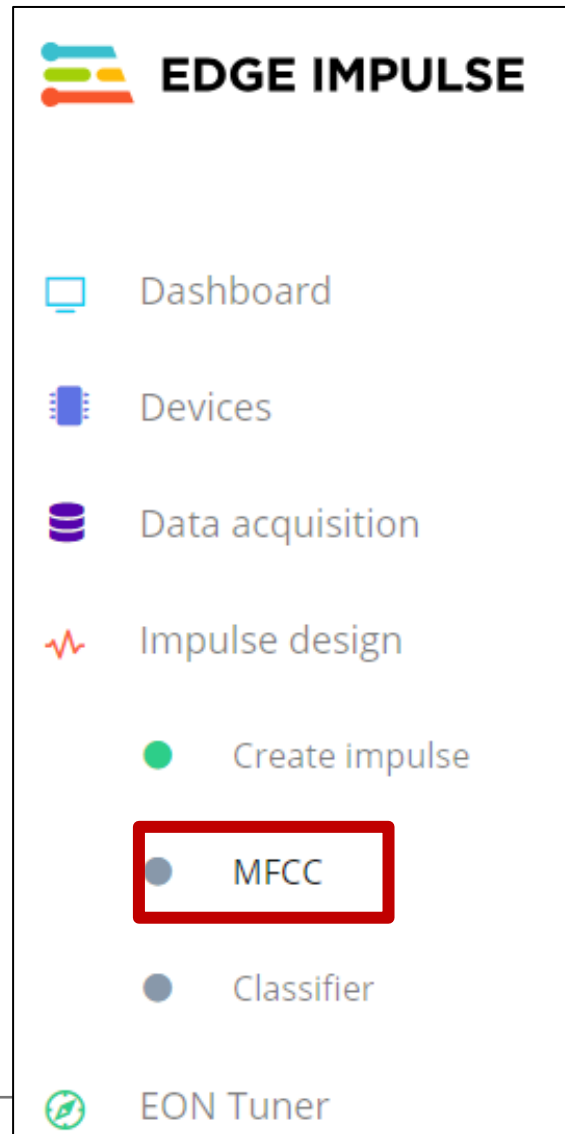
52

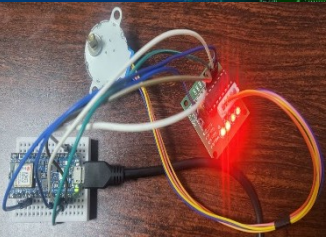


# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

## ❖ Impulse design

### □ MFCC

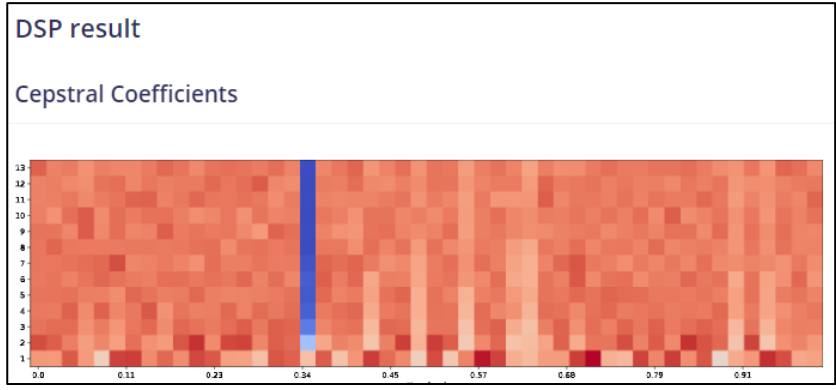




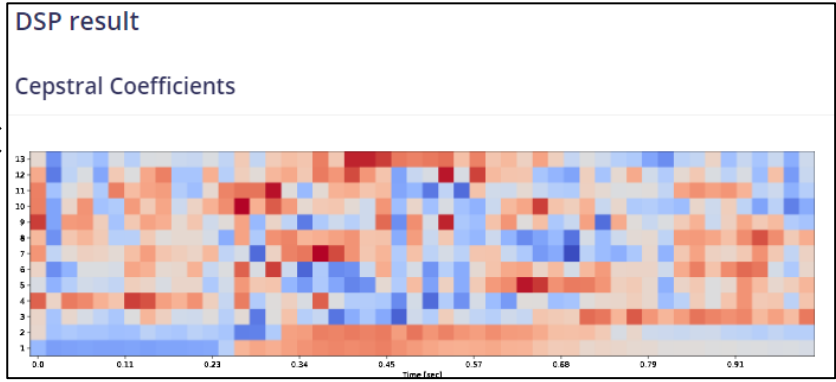
# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

- ❖ Impulse design
  - MFCC

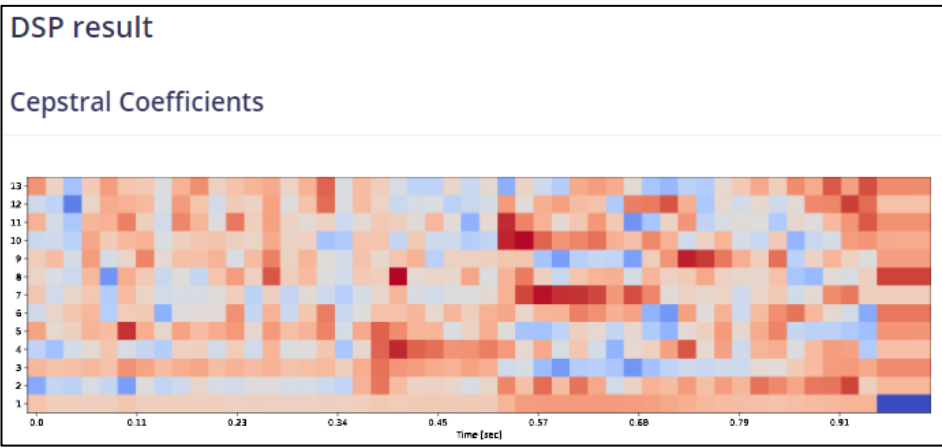
Silence MFCC

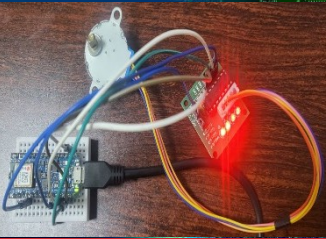


Forward MFCC



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 |

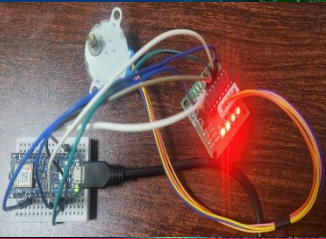
#### Pre-emphasis

|               |      |
|---------------|------|
| 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**

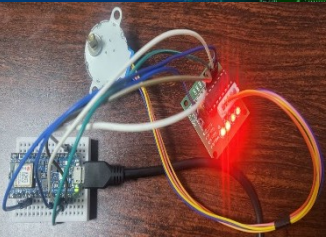
#### Feature explorer

No features generated yet.

#### Feature generation output

🔊 (0) ▾





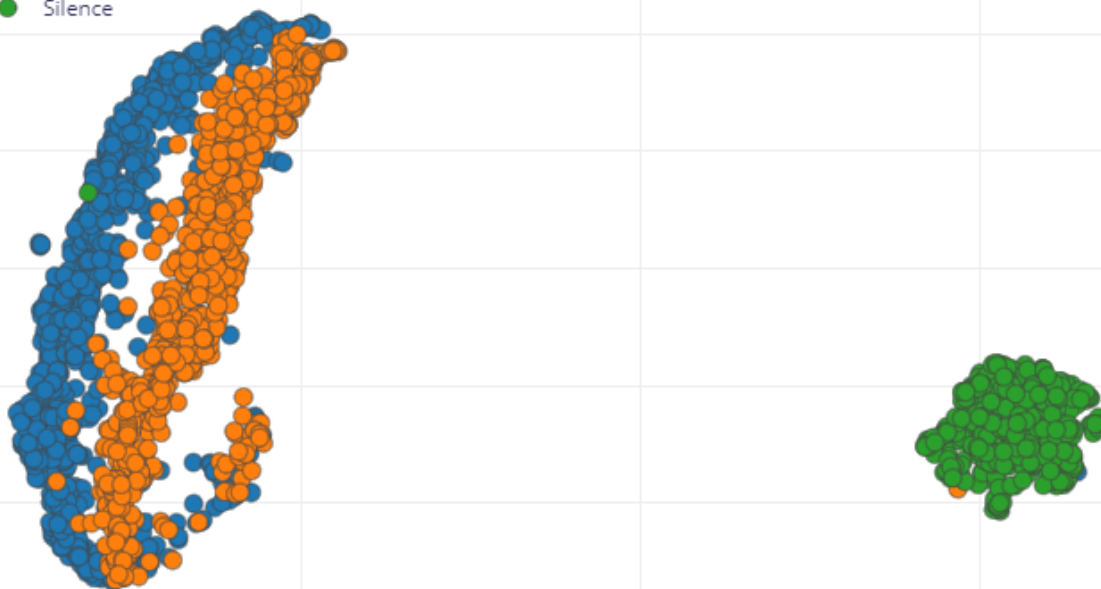
# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

❖ Impulse design

□ MFCC

Feature explorer ⓘ

● Backward  
● Forward  
● Silence



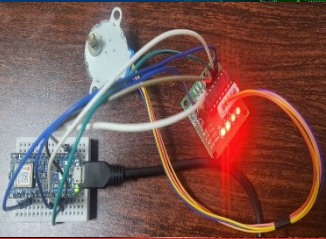
On-device performance ⓘ



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



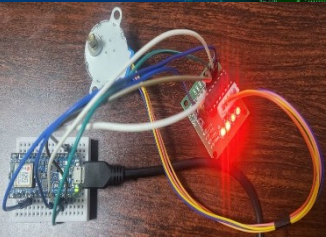
#### Advanced training settings



#### 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)

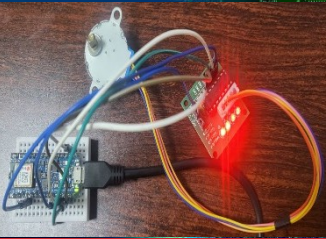
Dropout (rate 0.25)

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)

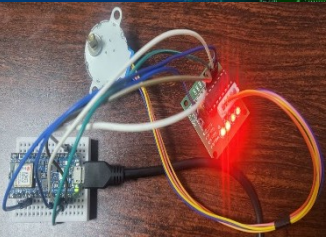
|          | BACKWARD | FORWARD | SILENCE |
|----------|----------|---------|---------|
| BACKWARD | 98.5%    | 1.2%    | 0.4%    |
| FORWARD  | 2.9%     | 97.1%   | 0%      |
| SILENCE  | 0.7%     | 0%      | 99.3%   |
| F1 SCORE | 0.98     | 0.98    | 0.99    |

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

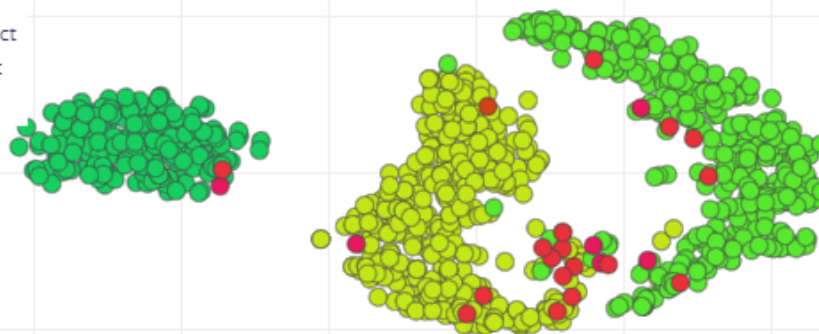
Model

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.

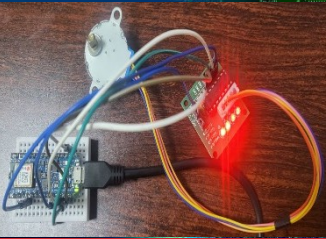


PEAK RAM USAGE  
7.0K



FLASH USAGE  
28.0K

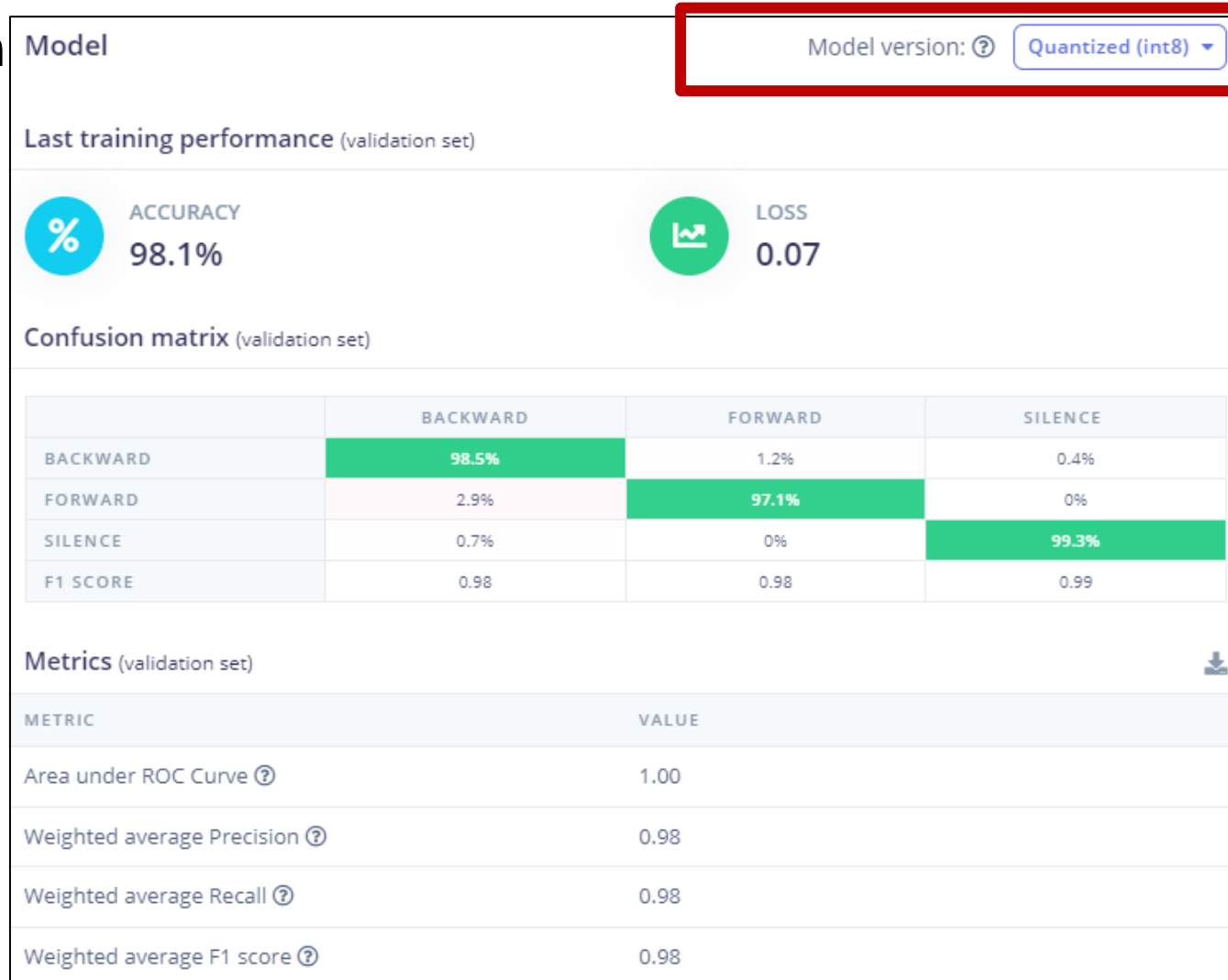




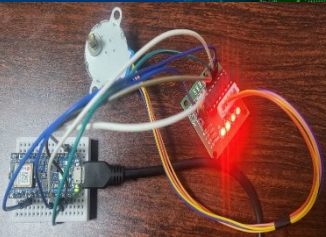
# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

## ❖ Impulse design

### □ Classifier



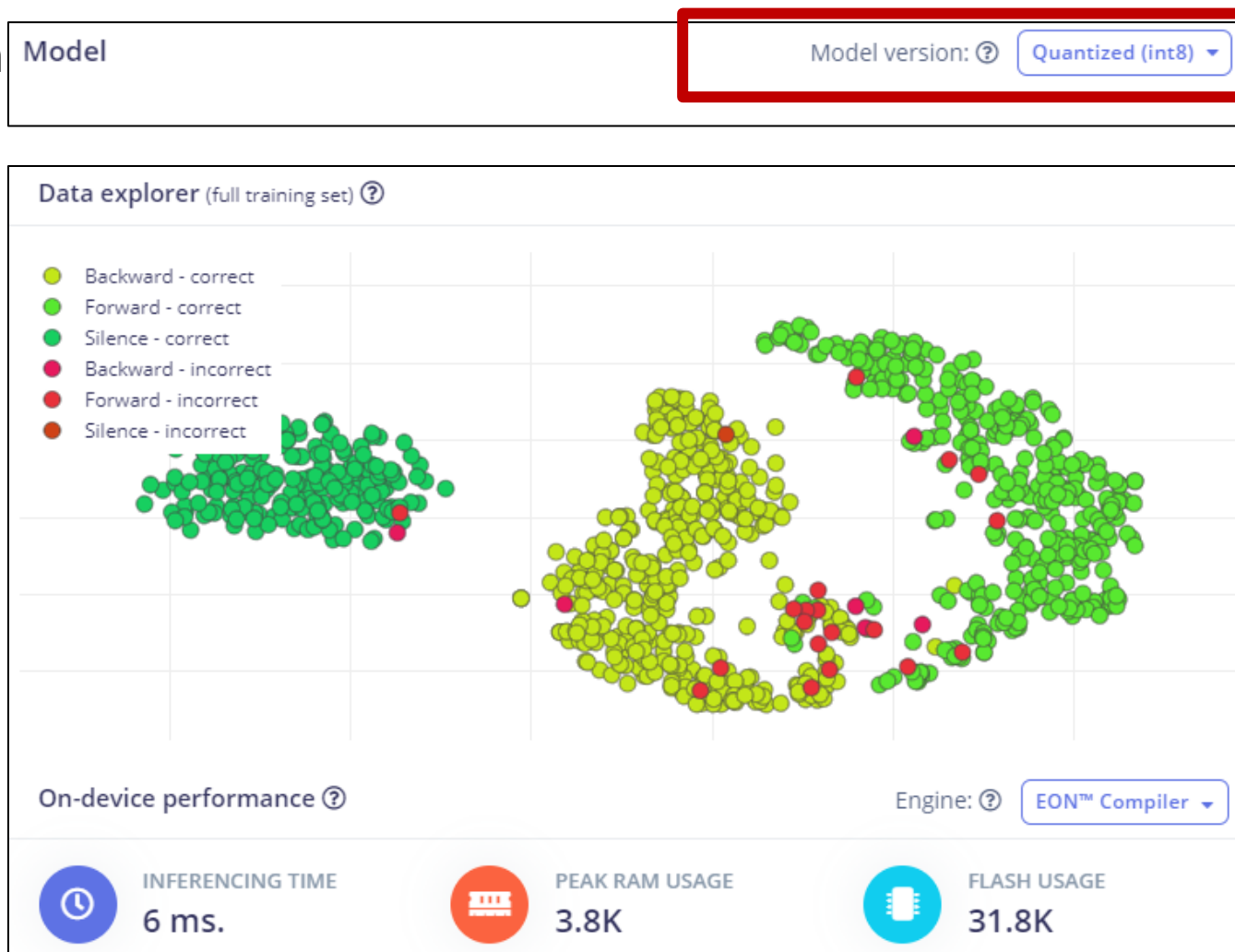


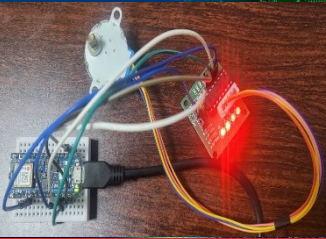


# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

## ❖ Impulse design

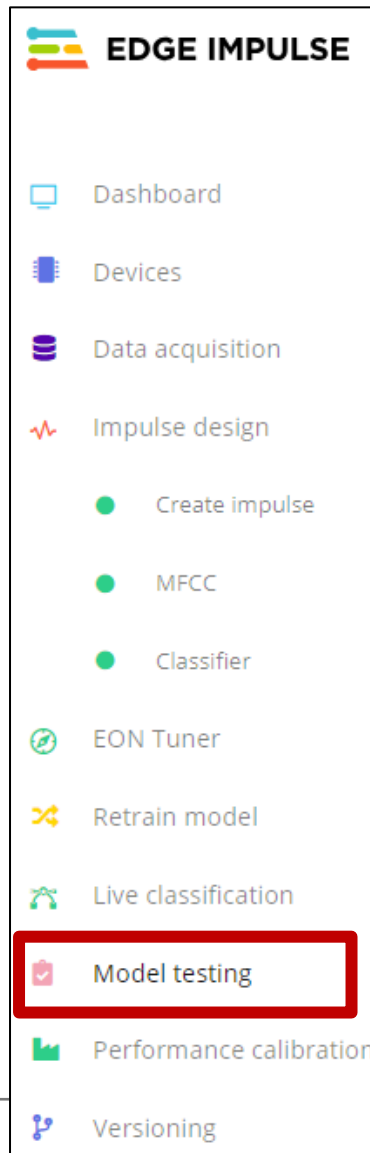
### □ Classifier

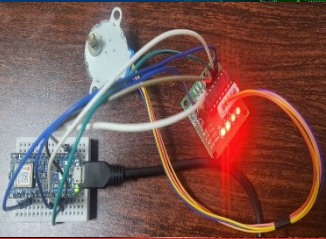




# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

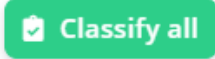

## ❖ Model testing



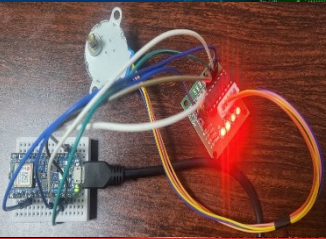


# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

## ❖ Model testing

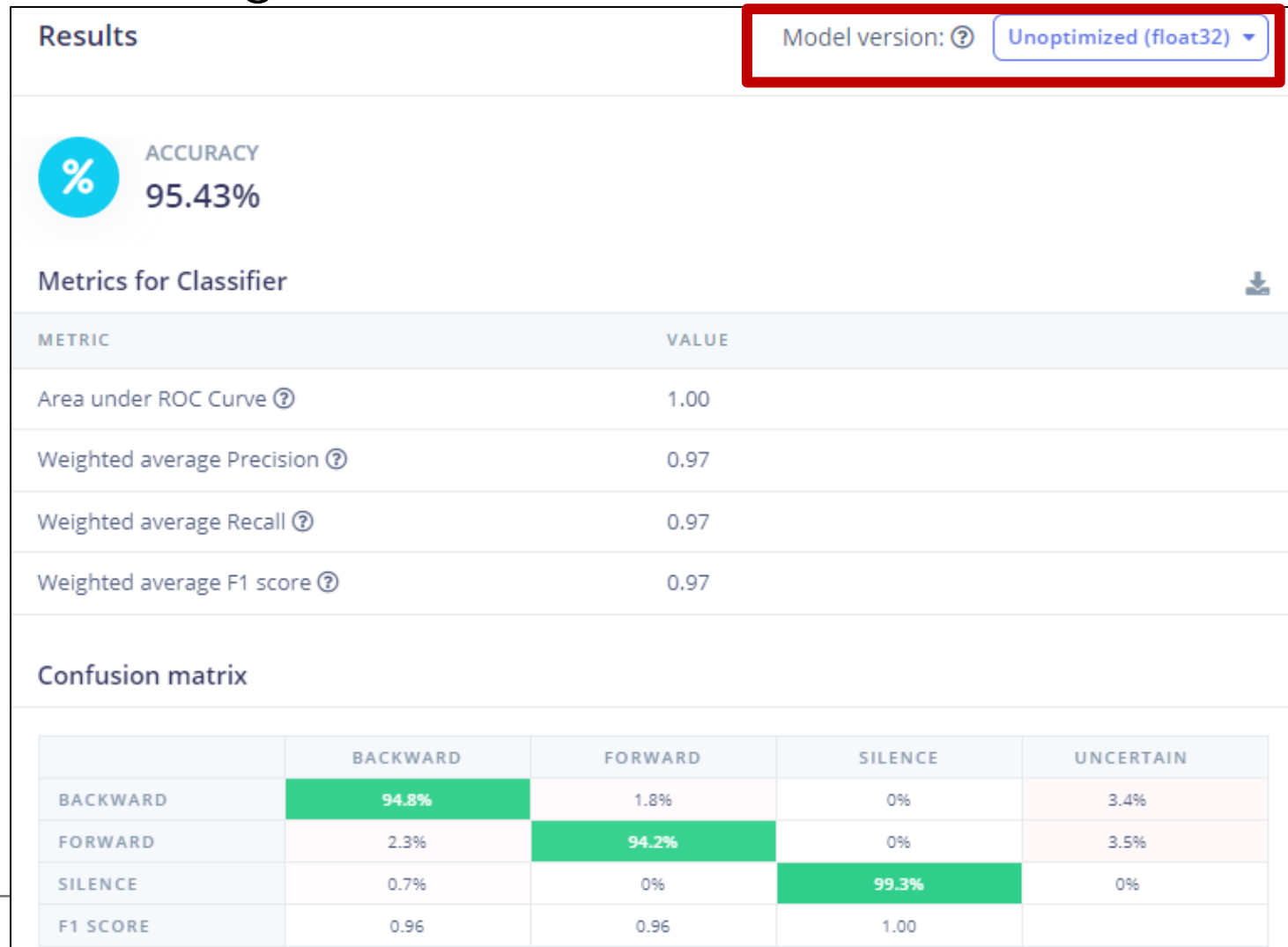
| Test data   |                 |         |          |        |   |
|---|-----------------|---------|----------|--------|---|
| Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse. |                 |         |          |        |   |
| SAMPLE NA...  |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

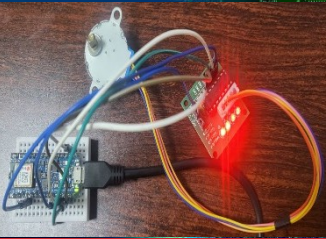




# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

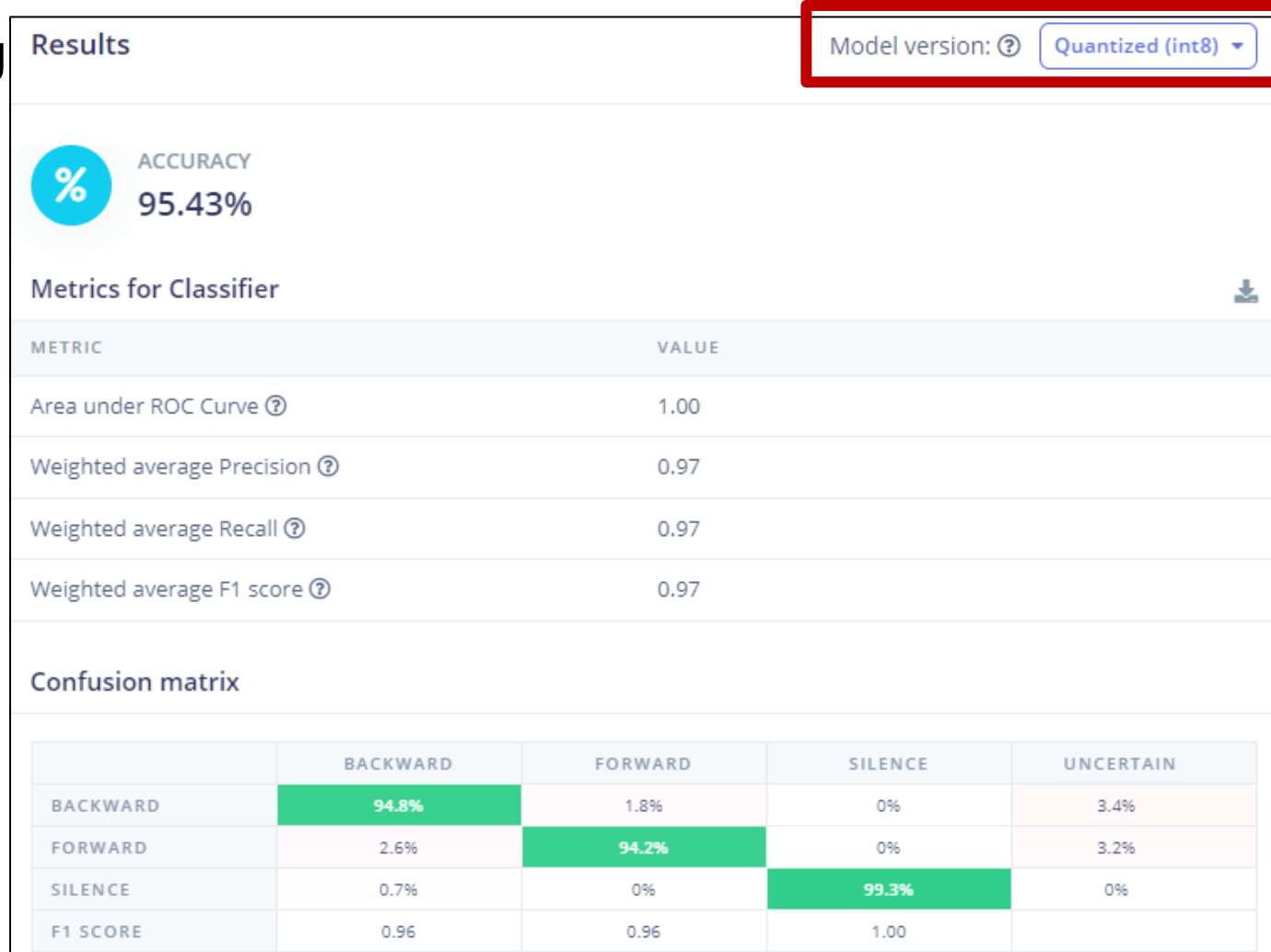
## ❖ Model testing

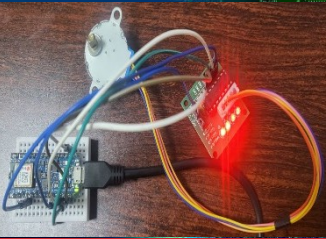




# MACHINE LEARNING MODEL TRAINING WITH EDGE IMPULSE

## ❖ Model testing



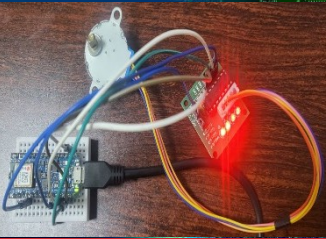


# MODEL DEPLOYMENT HARDWARE

## - 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

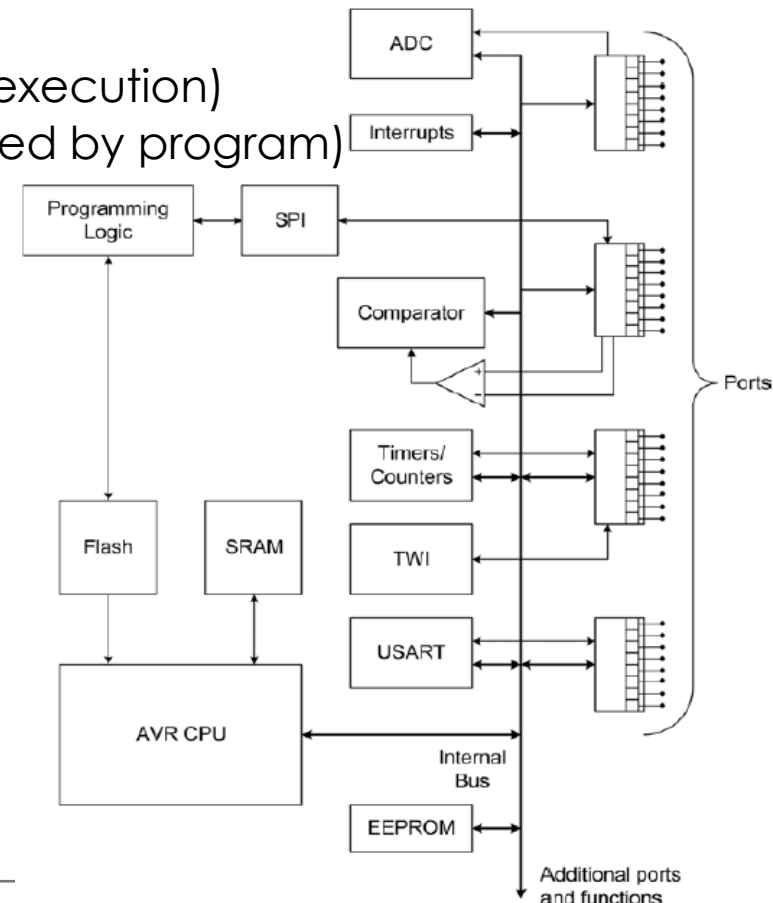




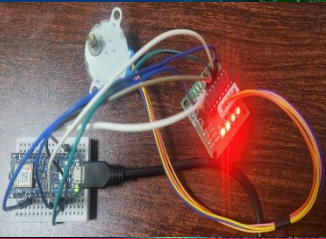
# MODEL DEPLOYMENT HARDWARE - ARDUINO

## ❖ A generic AVR microcontroller block diagram

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







# MODEL DEPLOYMENT HARDWARE - ARDUINO

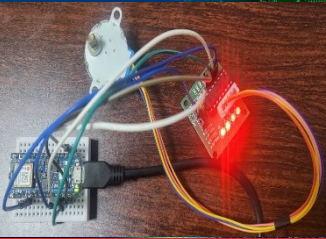
## ❖ Interfacing with Arduino

- ☐ 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>



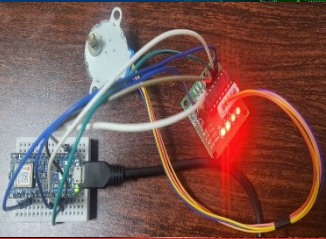


# MODEL DEPLOYMENT HARDWARE - ARDUINO

## ❖ 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                          |

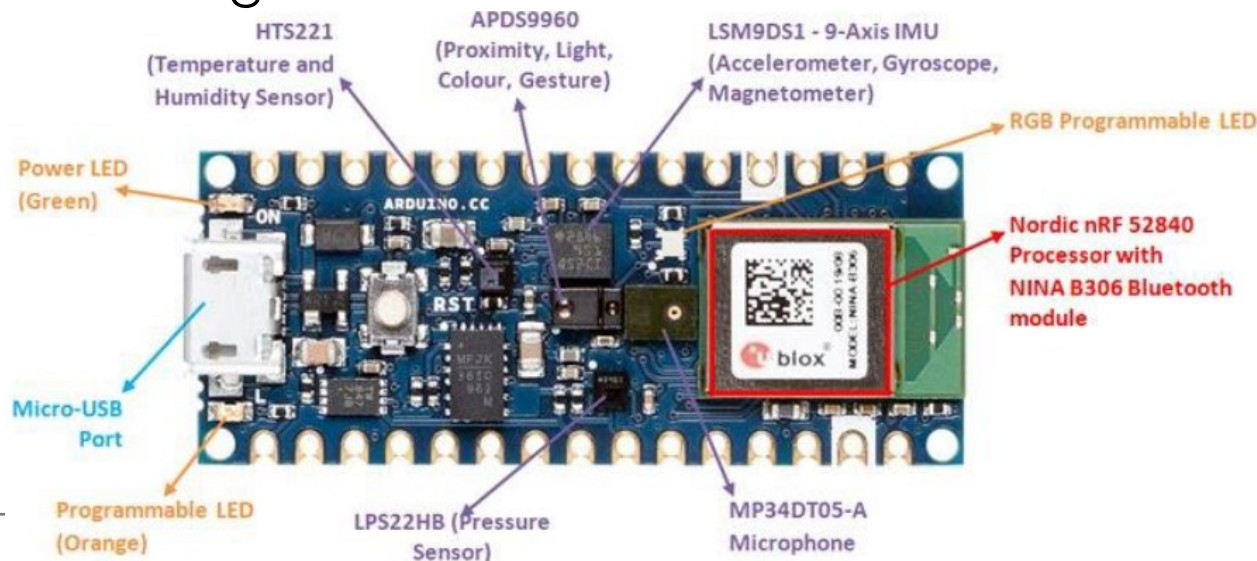


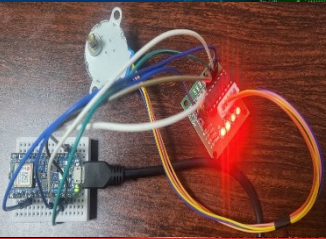


# MODEL DEPLOYMENT HARDWARE - ARDUINO

## ❖ Features of Nano 33 Sense

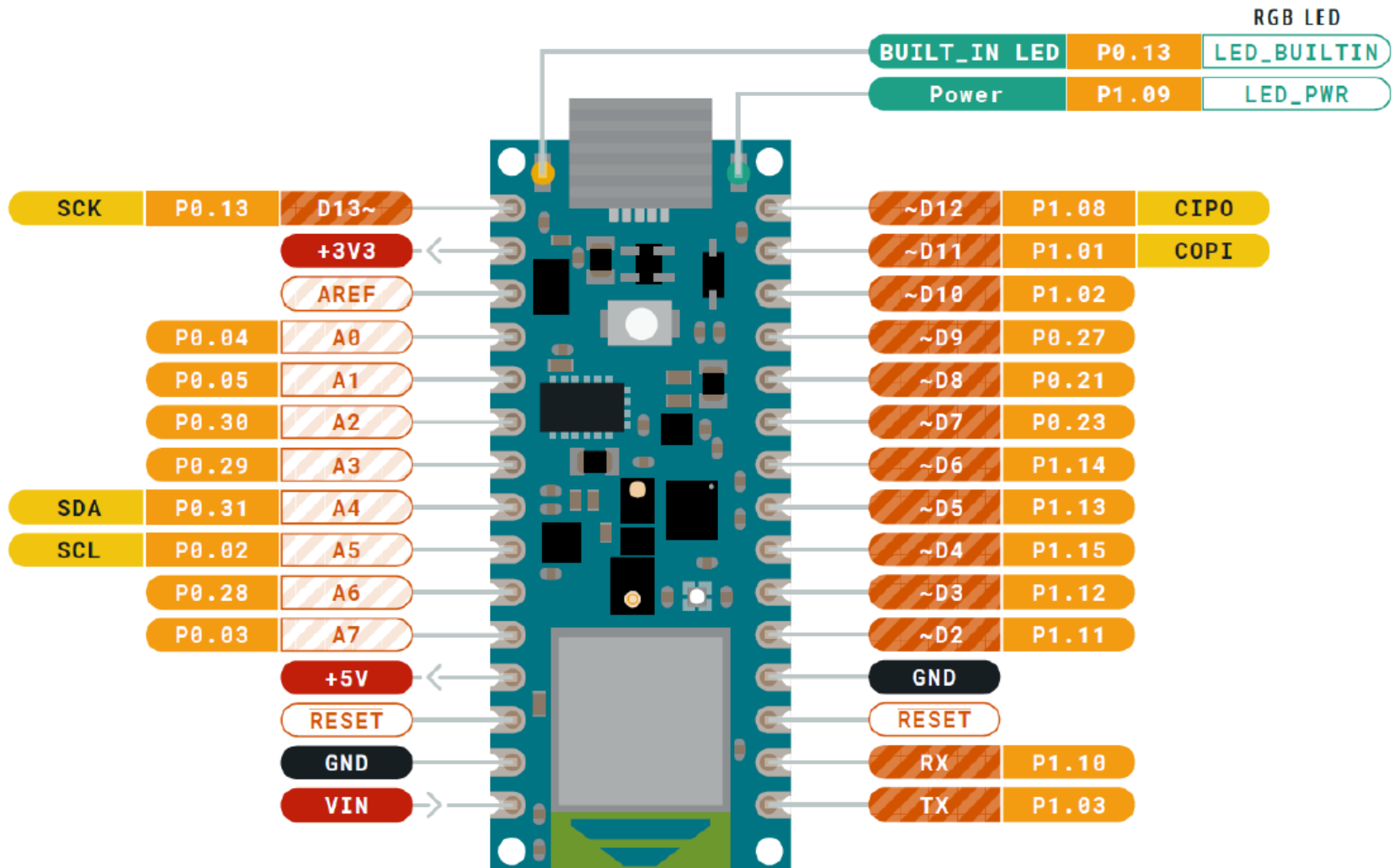
- 8 Analog Input Pins can provide 12-bit ADC at about 30 kHz
- Integrated sensors (IMU, Mic, Light, Pressure, Temperature, Humidity)
- All digital pins can trigger interrupts
- Only supports 3.3V I/Os and is NOT 5V tolerant so please make sure you are not directly connecting 5V signals to this board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

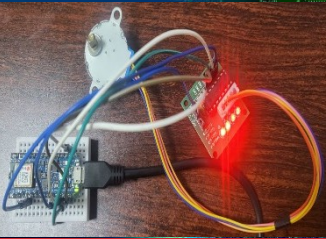




# MODEL DEPLOYMENT HARDWARE - ARDUINO

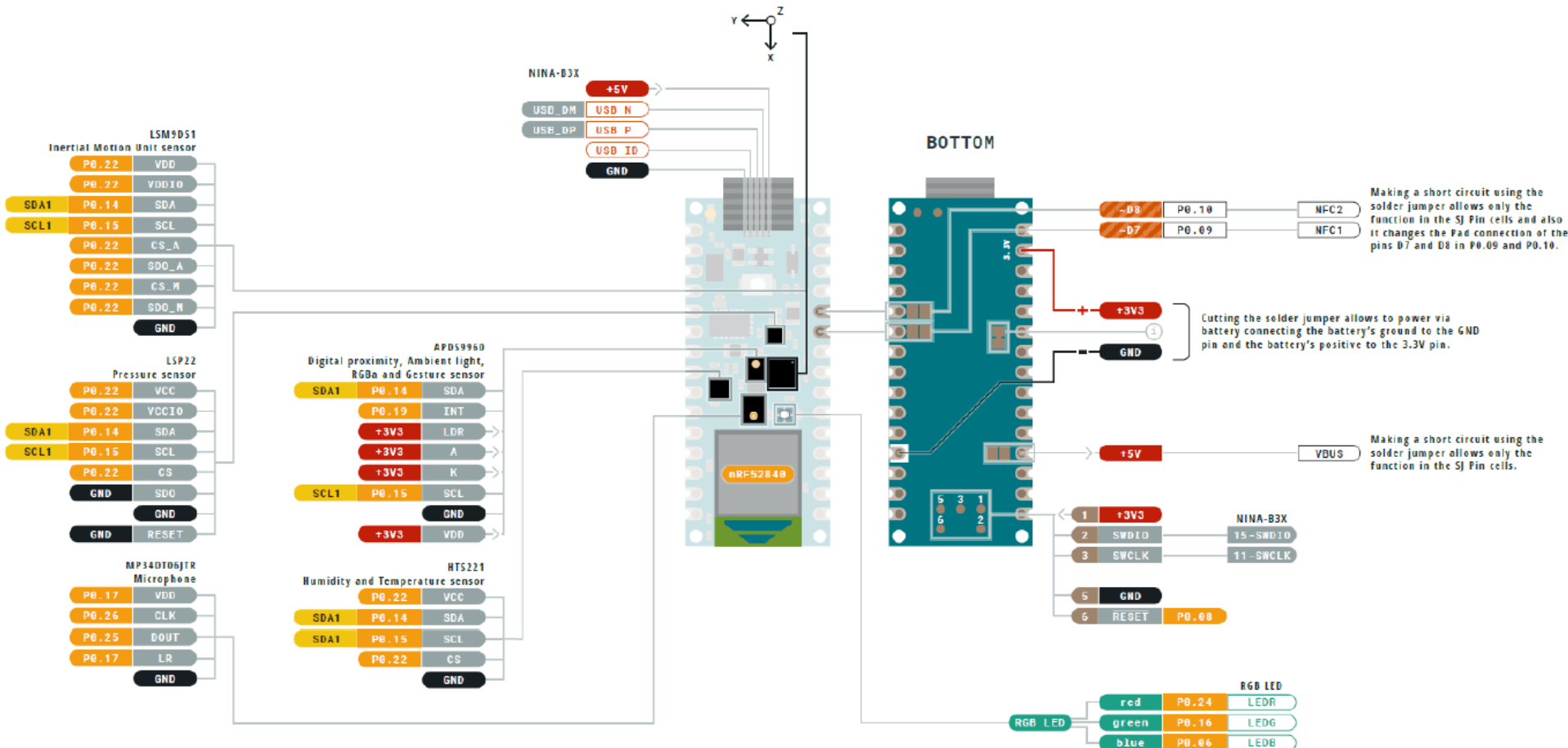
## ❖ Nano 33 Sense



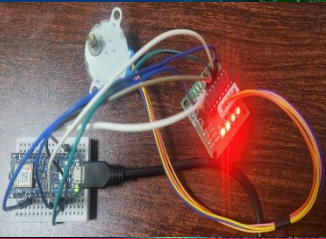


# MODEL DEPLOYMENT HARDWARE - ARDUINO

## ❖ Nano 33 Sense

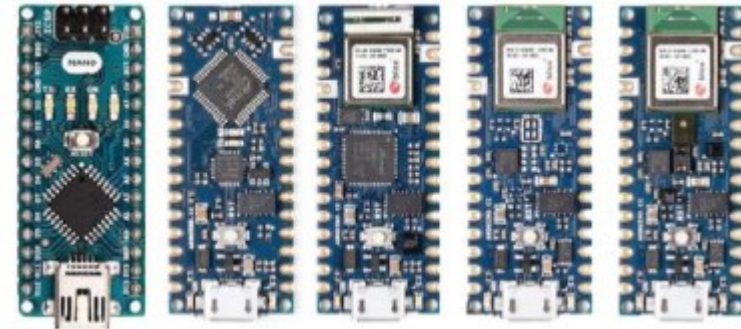






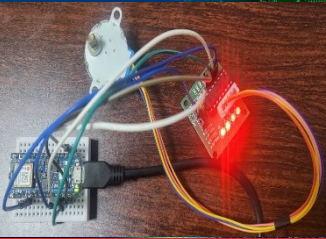
# MODEL DEPLOYMENT HARDWARE - ARDUINO

## ❖ 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                     |



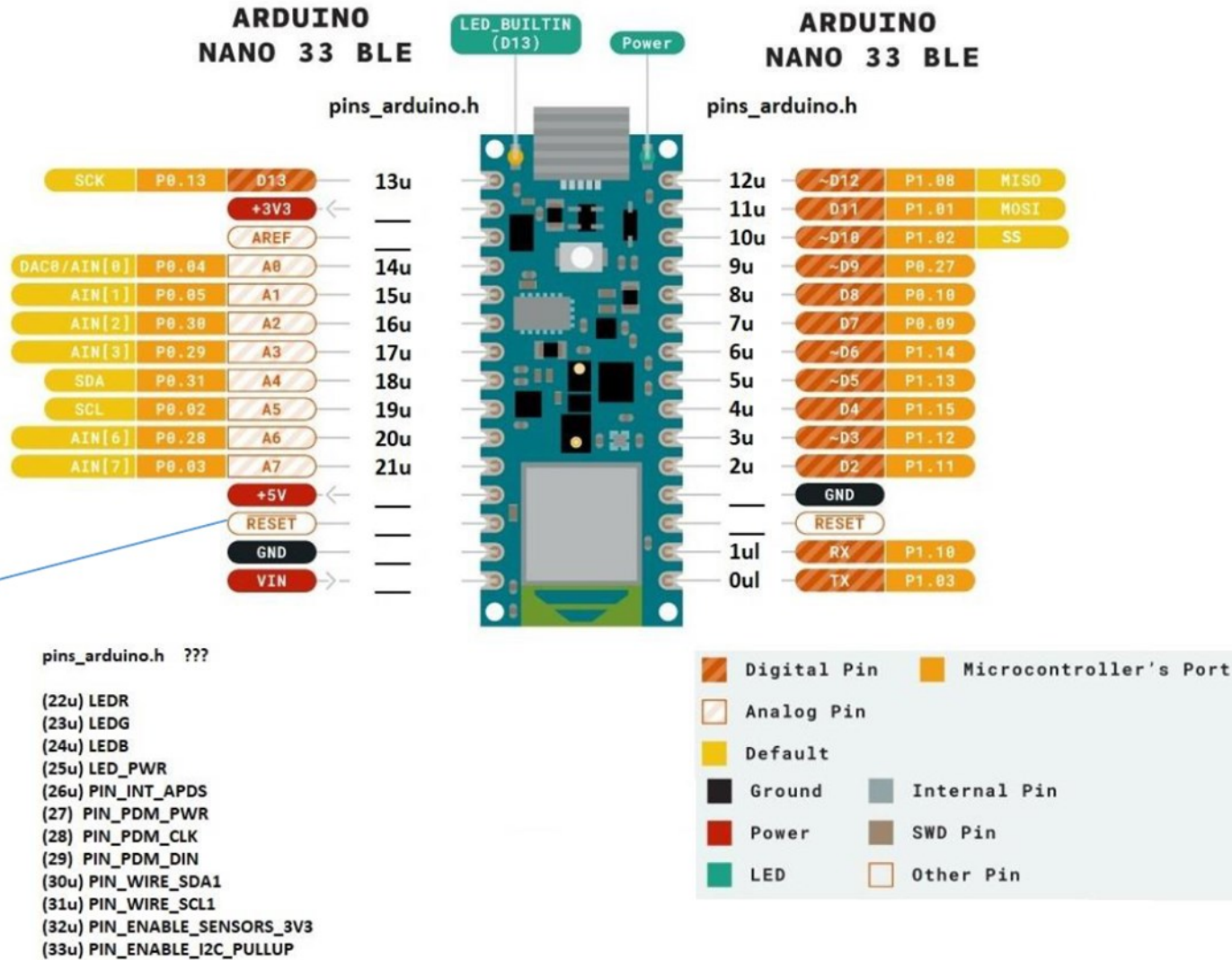


# MODEL DEPLOYMENT HARDWARE - ARDUINO

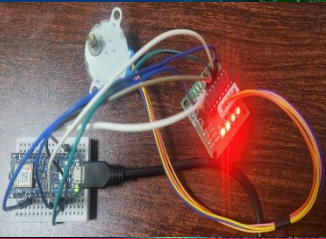
## ❖ Pinout

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

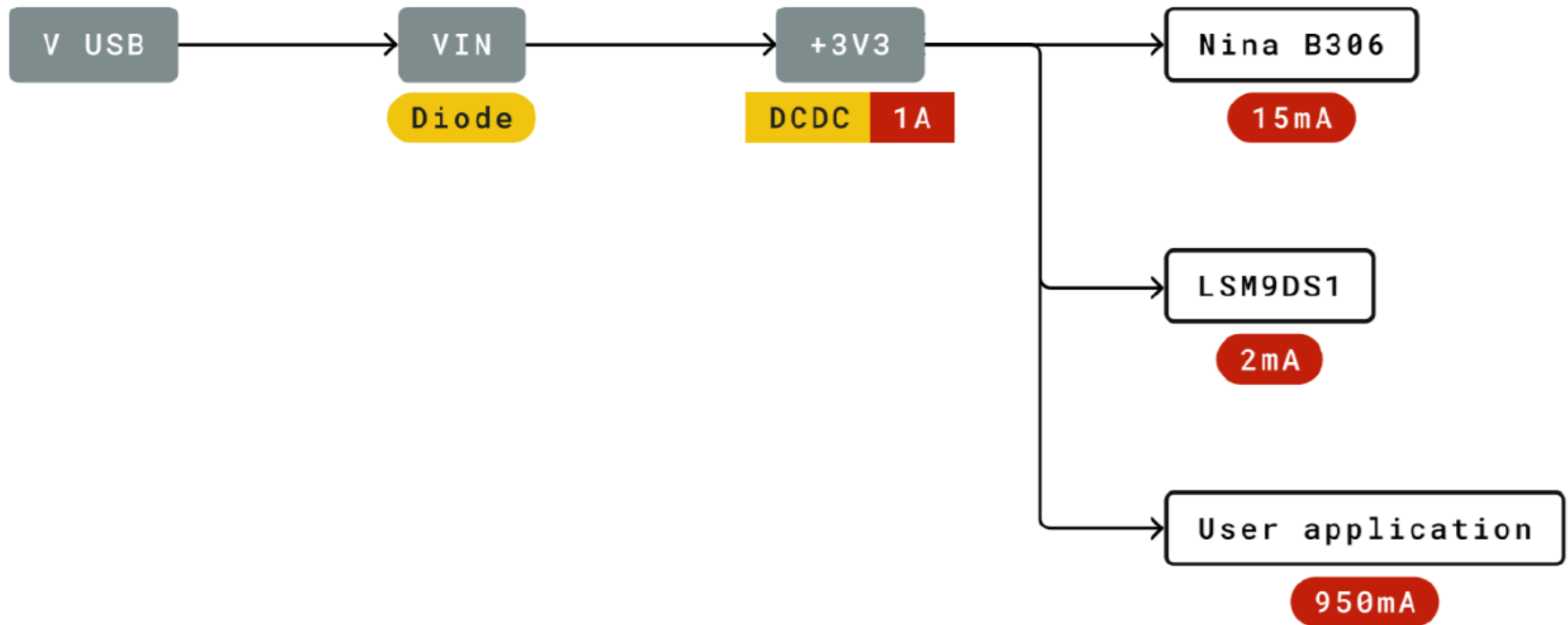






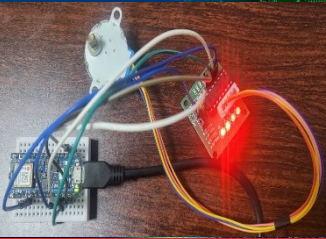
# MODEL DEPLOYMENT HARDWARE - ARDUINO

## ❖ 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.





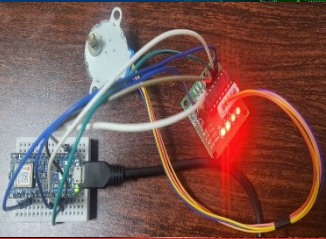
# MODEL DEPLOYMENT HARDWARE

## - 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;  
}
```

