Voice recognition wheelchair control system using STM32-based system [Group 3]

1. Hardware setup

   Integrate the NUCLEO-F446re with the MEM microphone INMP441 as the figure below. L/R of the microphone is connected to GND as mono-output I2S is used for our case.

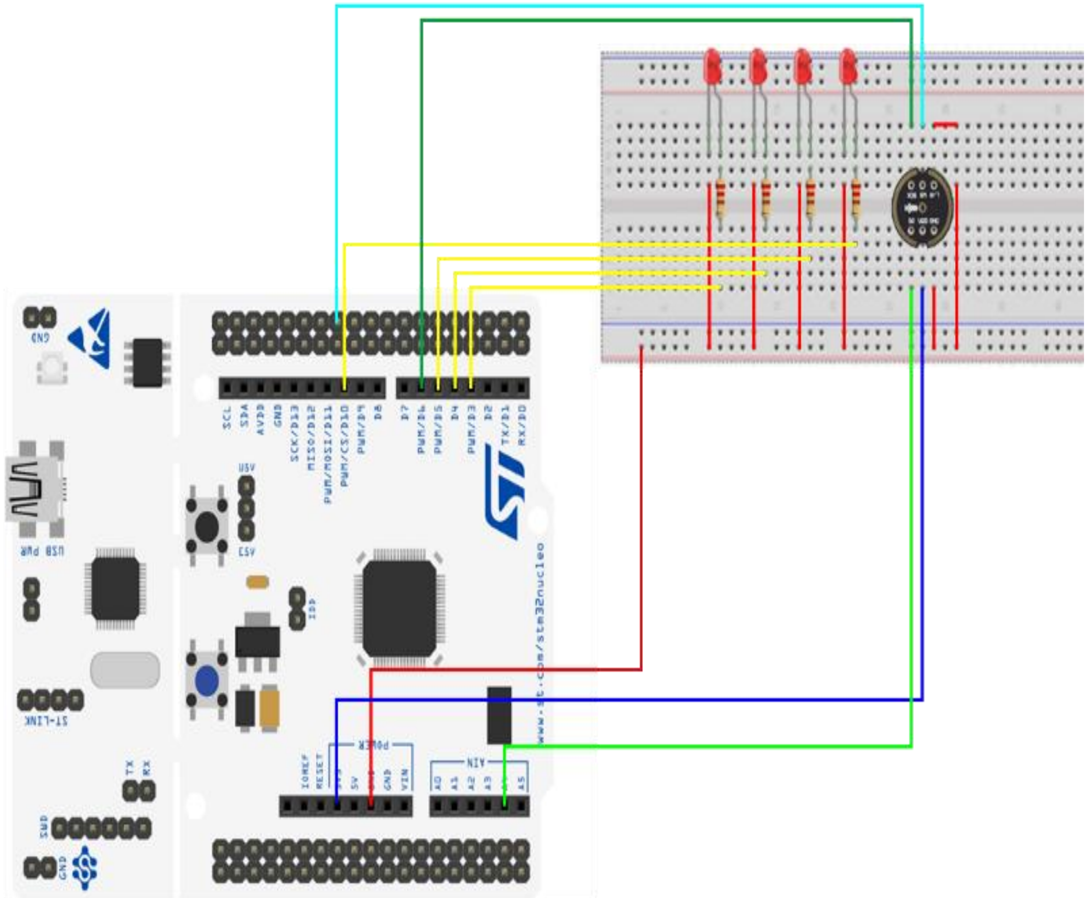

Figure 1: Setup of Hardware

Pin on microphone	Integration of microphone and MCU
GND	GND
VDD	3V3
L/R	GND
WS	PB12
SCK	PB10
SD	PC1

Table 1: Pin connection of microphone with MCU

2. Board configuration based on the hardware setup in the STM32CubeIDE. The board configuration includes the GPIO, DMA, USART and I2C setting.

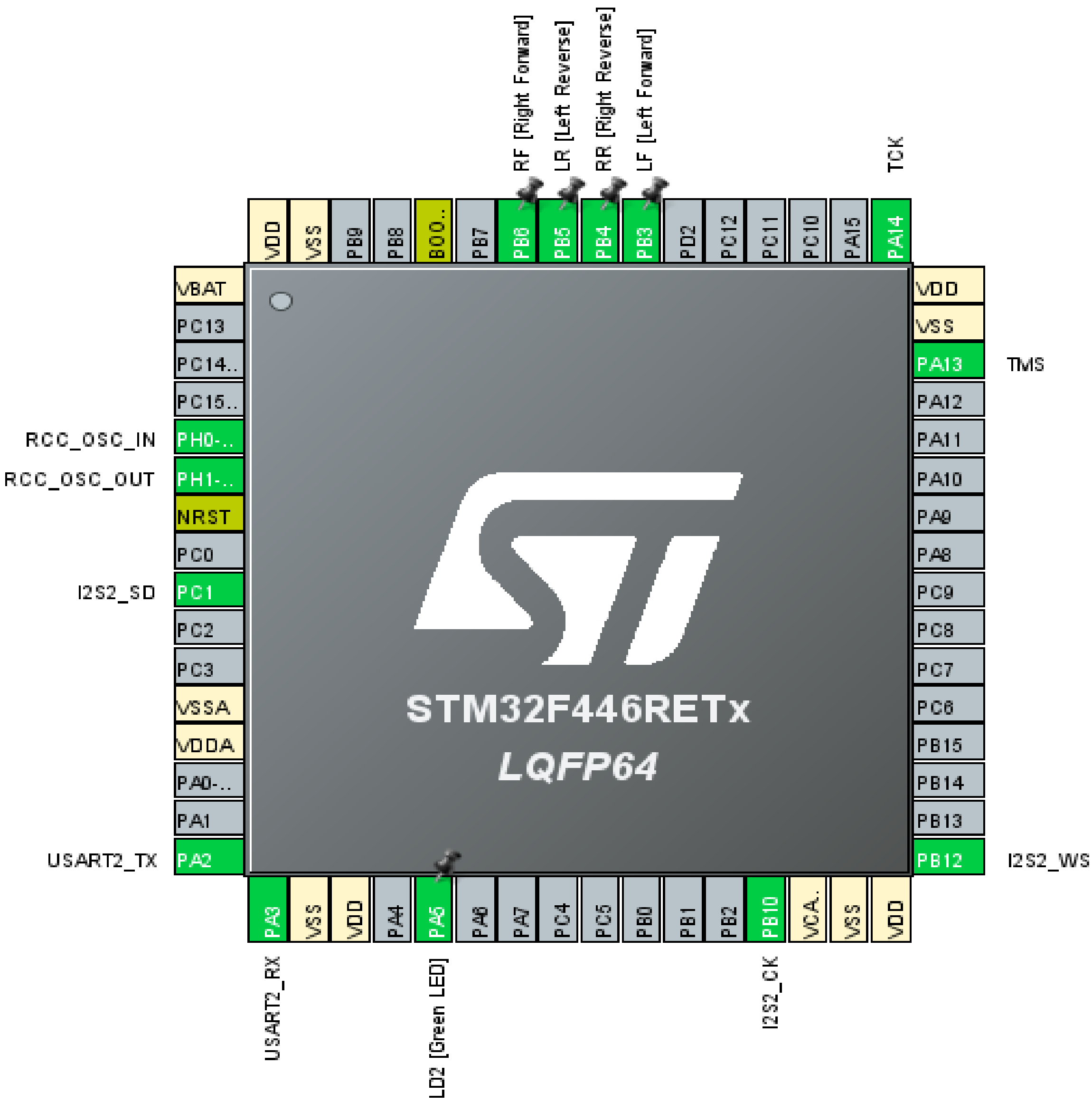

Figure 2: Overview of Board Configuration

- GPIO configuration

PA5, PB3, PB4 and PB5 is the output LED configuration.

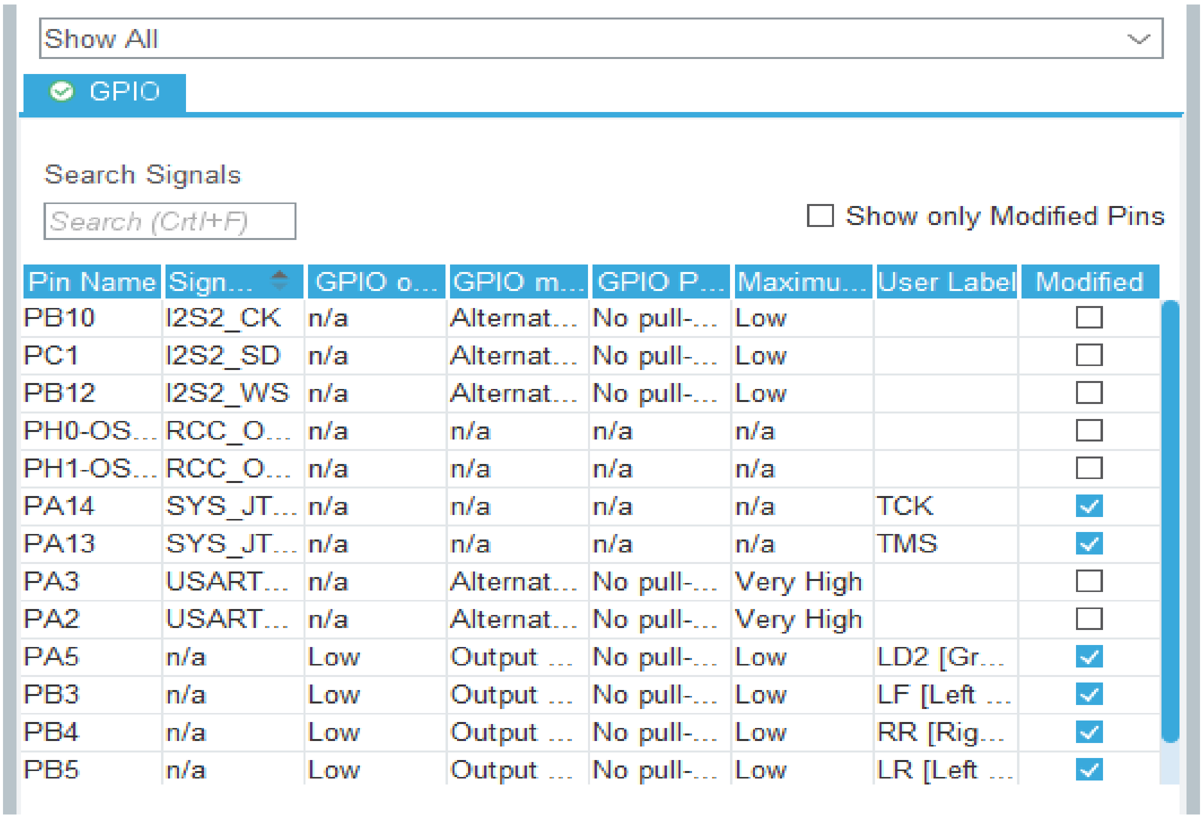

Figure 3: GPIO configuration

- DMA configuration

DMA is set to circular mode with data width of half word.

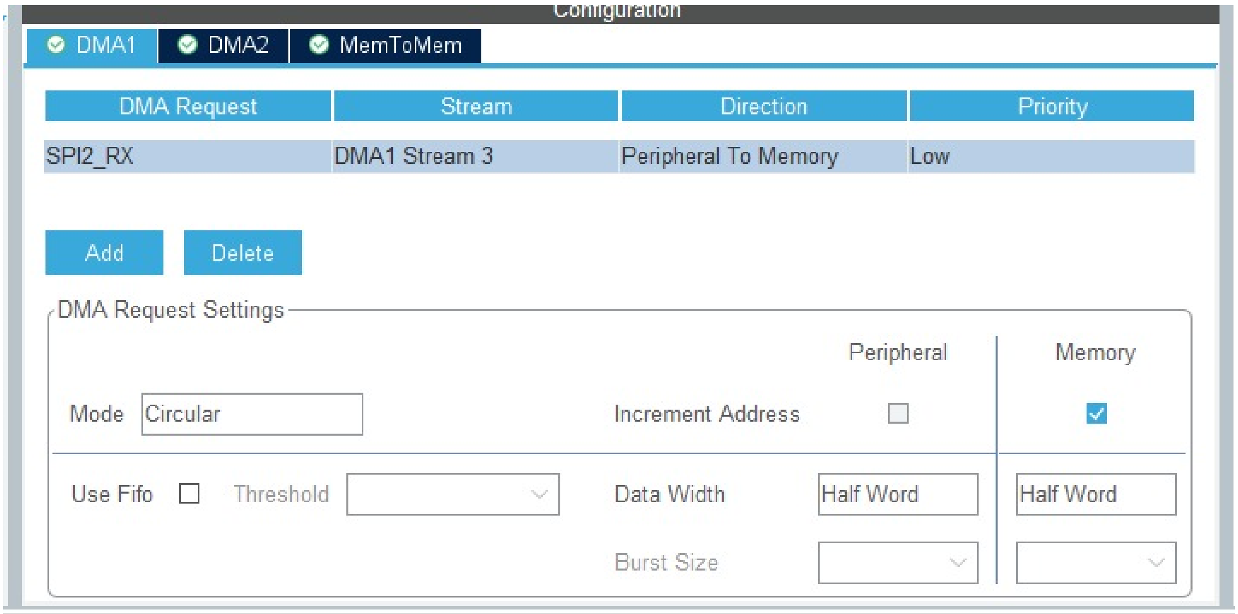

Figure 4: DMA configuration

- USART configuration
  The baud rate is set to 115200 Bits/s and the word length is set to 8 bits including the parity. The data direction is set to be receive and transmit with 16 samples.

Figure 5: USART configuration

- I2S configuration
  For I2S configuration, the transmission mode is set as Master Receive and the communication standard is I2S Philips. The data and frame format are set to 24 bits data on 32 bits frame with selected audio frequency of 32kHz.

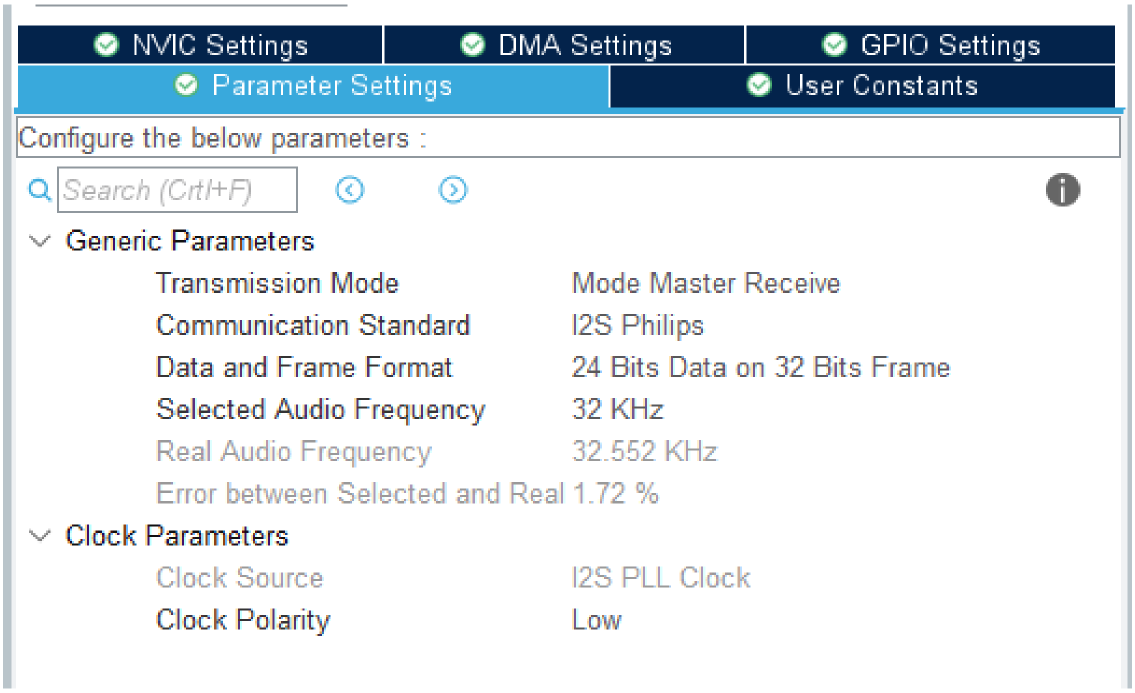

Figure 6: I2S configuration

3. CNN model training for keyword spotting algorithm

- Data Collection [Source : Colab]
  For keyword algorithm training, the data collection is obtained through google_speech_datasets instead of our custom keyword datasets. It is because the google_speech_datasets contains over 2000 samples which is enough for our project. The audio .wav samples of "left", "right", "forward" and "backward" is used where in the Colab target are set to the desired keywords.

```
### User Settings (do change these)

# Location of your custom keyword samples (e.g. "/content/custom_keywords")
# Leave blank ("") for no custom keywords. set to the CUSTOM_KEYWORDS_DIR
# variable to use samples from my custom-speech-commands-dataset repo.
CUSTOM_DATASET_PATH = ""

# Edge Impulse > your_project > Dashboard > Keys
EI_API_KEY = "ei_8745aae370404afe60c4f64f4f1111df3f7cb45e9b064bd525db126e2012001d"

# Comma separated words. Must match directory names (that contain samples).
# Recommended: use 2 keywords for microcontroller demo
TARGETS = "left, right, forward, backward"
```

Figure 7: Setting Desired Keywords

- Data Curation [Source : Colab]
  The data curation is done using dataset_curation.py where the samples .wav with background noise to produce a more robust dataset for training. Each keyword is curated to 1500 files as shown in the figure below. So, the curated data includes "left", "right", "forward", "backward", "_unknown" and "_noise".

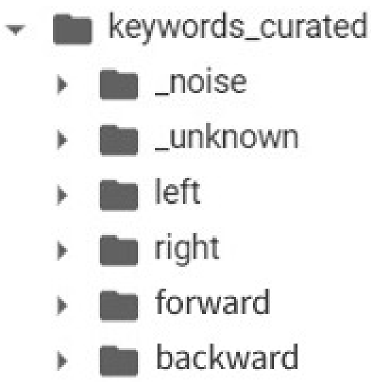

Figure 8: Curated Keywords

- Feature extraction [Source: Edge Impulse]
  CLI tool will send the completed curated dataset to Edge Impulse for training. Mel-frequency cepstral coefficients (MFCCs) extraction feature is used for sound processing purpose as a representation of short-term power spectrum of a sound. The step explanation for MFCCs extraction are:
  - Frame the signal into short frames.
  - Each frame calculates the periodogram estimate of the power spectrum.
  - Apply the mel filterbank to the power spectra, sum the energy in each filter.
  - Take the logarithm of all filterbank energies.
  - Take the DCT of the log filterbank energies.
  - Keep DCT coefficients 2-13, discard the rest.

  Figure 9 shows the diagram of the process of MFCC extraction while figure x shows the features generated for the keywords through MFCC.

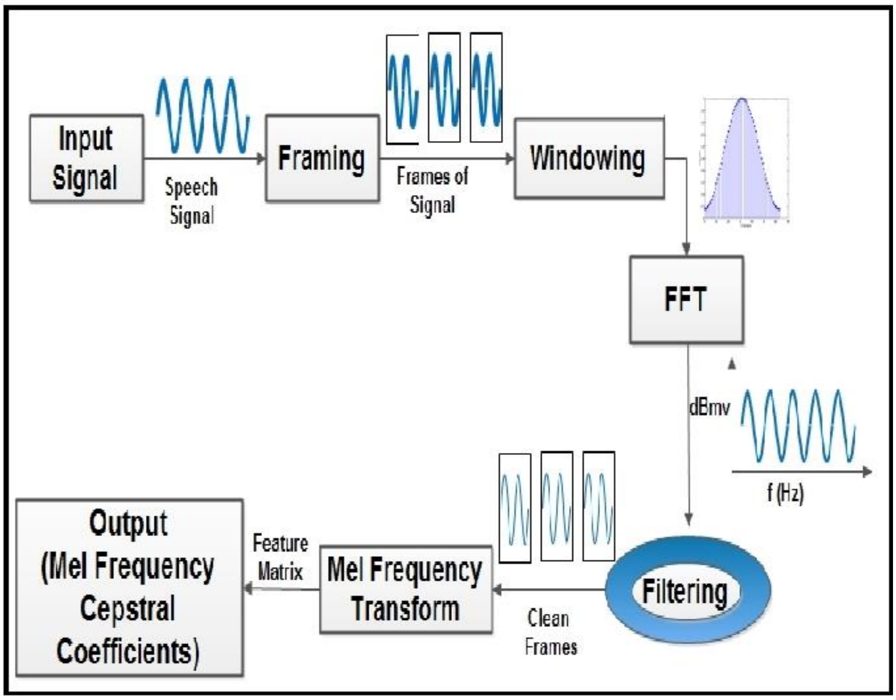

Figure 9: Flow of MFCC extraction

- Neural Network Training [Source: Edge Impulse]
  NN Classifier in the Edge Impulse is used to train the dataset for keyword spotting. Training settings such as number of training cycles, learning rate and minimum confidence rating need to be set accordingly. On the other hand, neural network architecture can be adjusted to meet the needs. Figure 10 is the result of training that obtained. The accuracy of 80.5% and loss of 0.64 is obtained as shown below.

Last training performance (validation set)

ACCURACY 80.5%

LOSS 0.64

Confusion matrix (validation set)

	_NOISE	_UNKNOWN	BACKWARD	FORWARD	LEFT	RIGHT
_NOISE	99.6%	0.4%	0%	0%	0%	0%
_UNKNOWN	5.5%	71.3%	1.2%	6.3%	13.0%	2.8%
BACKWARD	2.1%	16.4%	67.6%	2.9%	10.5%	0.4%
FORWARD	2.2%	12.6%	4.0%	78.9%	2.2%	0%
LEFT	1.2%	9.5%	0.4%	0%	88.9%	0%
RIGHT	2.5%	7.8%	0.8%	0%	11.9%	77.0%
F1 SCORE	0.93	0.66	0.78	0.83	0.78	0.85

Feature explorer (full training set)

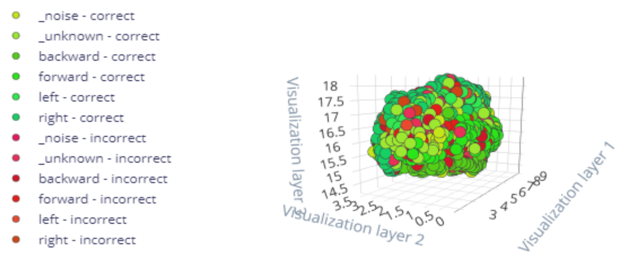

Figure 10: Result of Neural Network Training for Desired Keywords

4. On board testing
After the training is done on the Edge Impulse, the impulse or data can be deployed into any devices. This makes the model run without any internet connection, minimizes latency and runs with minimal power consumption. The trained model and model parameters are built from the deployment tab on Edge Impulse platform as c++ library file. The downloaded library files are then being feed into the STM32CubeIDE for on board testing purpose. All the needed header files are imported. The desired keywords "left", "right", "forward", "backward" and random words such as "testing" are used to test on the microcontroller to check if the microcontroller can recognise the keywords.

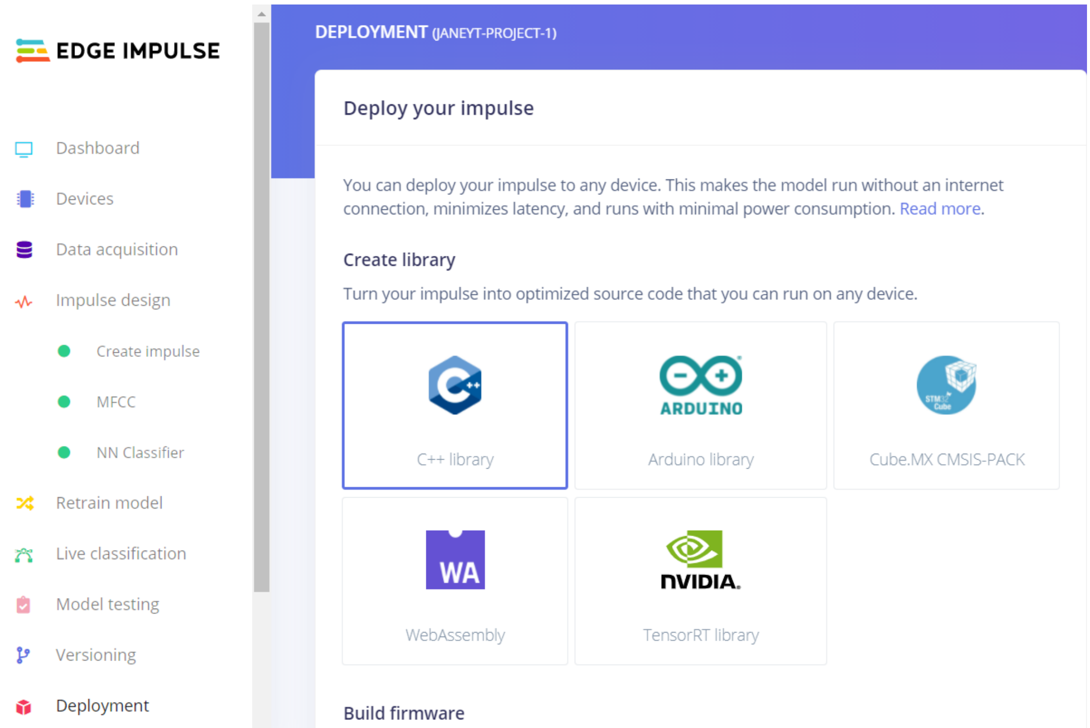

Figure 11: Data Deployment from Edge Impulse

Reference

- https://github.com/andysworkshop/usb-microphone
- https://github.com/Uberi/speech_recognition
- https://github.com/nodae/teensy_i2s_experimental
- https://github.com/wantt/stm32_speech_recognition/blob/master/README.md
- https://invensense.tdk.com/wp-content/uploads/2015/02/INMP441.pdf
- https://github.com/ShawnHymel/ei-keyword-spotting
- https://studio.edgeimpulse.com/studio/33717/deployment
- https://github.com/topics/mems
- https://andybrown.me.uk/2021/03/13/usb-microphone/
- https://github.com/Uberi/speech_recognition
- https://www.st.com/resource/en/application_note/cd00259245-audio-and-waveform-generation-using-the-dac-in-stm32-products-stmicroelectronics.pdf