Embedded AI Application

Referencing: [Installation Setuping the Environment for Testing](https://docs.edgeimpulse.com/docs/edge-ai-hardware/mcu-+-ai-accelerators/syntiant-tinyml-board)

SETUP ENVIRONMENT FOR TESTING

1. Installing Dependencies:

• Arduino CLI

• Edge Impulse CLI

* 1. Arduino CLI

‘’’bash’’’

brew update

brew install Arduino-cli

* 1. Edge Impulse CLI
     1. Create an Edge Impulse Account
     2. Install python3
     3. Install node.js v20.x+ or

‘’’bash’’’

curl -sL <https://deb.nodesource.com/setup_20.x> | sudo -E bash –

sudo apt-get install -y nodejs

node -v

* + 1. Verify installation with: npm config get prefix
       1. If returns /usr/local/ run this to adjust the path:

‘’’bash’’’

mkdir ~/.npm-global

npm config set prefix ‘~/.npm-global’

echo ‘export PATH=~/.npm-global/bin:$PATH’ >> ~/.profile

* + 1. Installation CLI tools via: npm install -g edge-impulse-cli

1. There are two firmware (audio and IMU), for this application will be important to play with Audio firmware, because is a sensing AI application. Important to note that SD Card is needed for IMU data acquisition. To initiate select the firmware and download the file and then execute the system corresponding to the owned OS. If on MACOS it may output that Apple is not able to verify the trustiness, so is required to:
   1. Apple > Impostazioni di Sistema > Privacy & Sicurezza > Sicurezza
   2. Click on Open and then Open Anyway
   3. Insert login password and authorize that

To execute this the program should be in boost mode so double click on reset button

1. Check that TinyML is in Sound as an Input
2. Installation Arduino IDE, on which I will develop

Immagine che contiene elettronica, tastiera, Attrezzatura per ufficio, schermata

Descrizione generata automaticamente

Keyword Spotting: [Reference example](https://docs.edgeimpulse.com/docs/run-inference/hardware-specific-tutorials/responding-to-your-voice-syntiant-rc-commands-go-stop)

To familiarize with the system I changed the colors of the led saying a specific word (red, green, yellow, blue, pink, cyan, white)

1. Training custom model [source](https://docs.edgeimpulse.com/docs/run-inference/hardware-specific-tutorials/responding-to-your-voice-syntiant-rc-commands-go-stop)
   1. Data Acquisition: First record the label words and split them creating various sample with various sounds in order to get a various amount of raw data. Word recorded is blue, with the objective to activate a corresponding color

• The noise elements where picked from: [source](https://docs.edgeimpulse.com/datasets/audio/audio-classification/audio_classification_keyword_spotting) @misc{edgeimpulse\_dataset\_499022,

title = {Audio Classification - Keyword Spotting},

author = {Edge Impulse},

year = {2024},

url = {https://studio.edgeimpulse.com/public/499022/latest},

note = {Apache 2.0}

}

* 1. Classification: Starting from the raw data obtained we would like to create features according to some parameters. A recording will generate a spectrogram and a clearer one is more reliable allowing it to be distinguishable.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteImmagine che contiene testo, schermata, diagramma

Descrizione generata automaticamente

The features generated are the following:

Immagine che contiene testo, schermata, mappa

Descrizione generata automaticamente

* 1. Classifier: Basing on the features generated from classification to build the neural network we start from the input layer containing those features to an output one containing various classes. The levels of this network are:

• Input layer (1600 features)

• Dense layer (256 neurons)

• Dropout (rate 0.2)

• Dense layer (256 neurons)

• Dropout (rate 0.2)

• Dense layer (256 neurons)

• Dropout (rate 0.2)

• Output layer (4 classes)

This classifier will output a confusion matrix and a graphic:

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteImmagine che contiene testo, mappa, diagramma

Descrizione generata automaticamente

* 1. Training: To model the testing to score the various impulse based on an expected outcome and this will technically be shakier. This phase will generate a trained model that would function as reference. As classification is generated a confusion matrix and then a graph:

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteImmagine che contiene schermata, testo

Descrizione generata automaticamente

* 1. Deployment: After developed the model, we can test it directly on the hardware, so it is important to download the generated model to upload that on the NDP101, but to custom the interaction with hardware of host processor. First it is important to create that firmware to program directly the NDP101.

1. Coding hardware part of the Arduino MKRZero

After building the firmware with that model, starting from that is essential to code in my case to toggle the RGB Led to Blue, when the NDP101 is triggered and recognizes the BLUE label. To implement this behavior I used Arduino IDE to implement the logic, with a custom Enum containing the various possible labels and a service routine called via loop () function, so continuously, to toggle the led with a precise address.

This can be seen in the following code:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

To interact with the Arduino must be uploaded after the firmware one. After this the test successfully toggled the led after the word recognition, so the process works correctly.

The problems found along the way were primarily two:

1. Flash Memory address counting problem, after some testing and uploads the device didn’t allow any upload because it looked like reading in binary file 0xFF. To resolve this I had to totally reset with a command send to host processor (command :F) this like set the status at the original status
2. At some point the device was stuck in a blue light, different from the voice recognition one, this raised due to a model that wasn’t found, so when having a custom model is necessary to provide in Arduino code the path to the desired model, corresponding to the composed firmware