INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA
Departamento de Engenharia Eletrónica e Telecomunicações e Computadores

Intelligent Sports Weights

TFM11 - Dissertação de Mestrado 2023/2024

Olga dos Santos Duarte, Nº 27675

Presentation Outline

Introduction & Motivation

- Build an embedded system in a microcontroller with a trained Neural Network, using *TinyML*.
- Use of low code or no code platform.
- Program a neural network on low power microcontroller.

01

Safe Physical Exercise

Identify the correctness of fitness exercises to avoid injury.

02

Embedded Systems

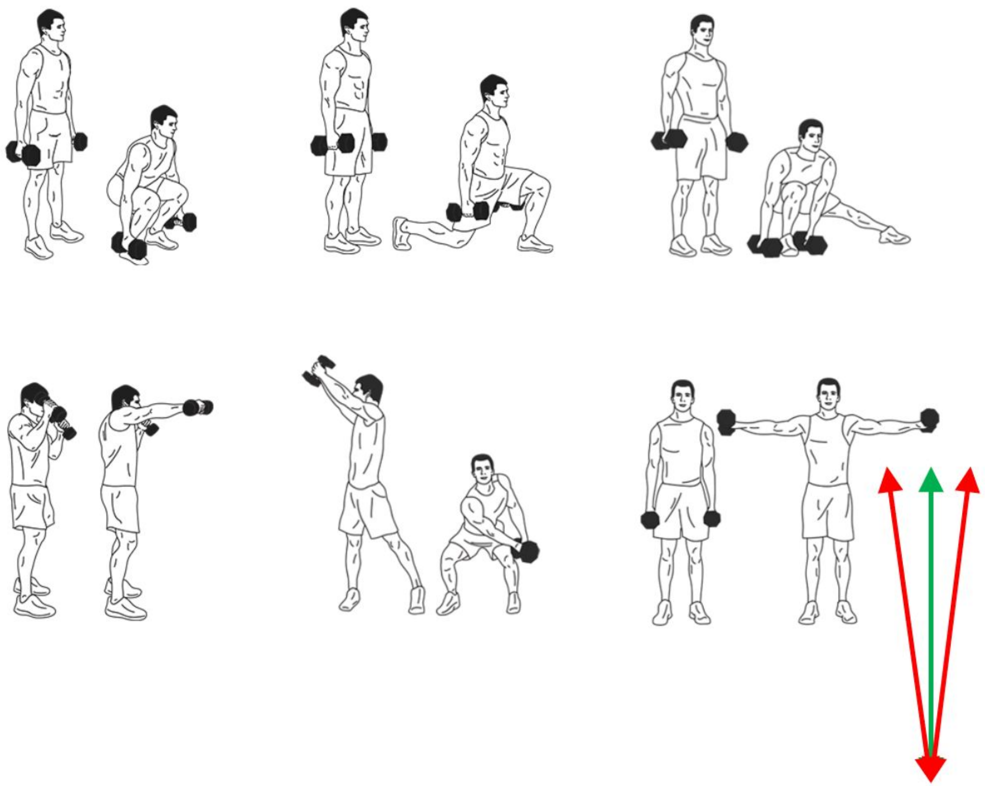
Autonomous, low-power, small size

03

Real time feedback

Sensors in gym equipment to provide feedback.

Objectives



01 **Autonomy** - Operate autonomously with a battery for more than 8h.

02 **Compact Design** – Compact and lightweight facilitating ease of use during exercise and to be attached to gym equipment.

03 **Real-time Data Acquisition** - Capability to collect motion data using accelerometer data in real-time and communicate to a host device using BLE.

04 **Training NN** - Train a NN model with a custom dataset for different exercise movements and with correct and incorrect labels.

05 **Classification** - validate the movement via a trained NN.

06 **Feedback** - provide real-time feedback to the user.

Related Works on Movement Recognition

Neural Networks for classification and movement recognition. Works were investigated in multiple areas such as:

01

Neural Networks applied to Sports

Wearable sensors to detect moves, using the IMU signal processing methods to classify specific activities. Examples like: jump frequency in volleyball, putt in golf, activity recognition in beach volleyball using a Deep Convolutional Neural Network - used also to avoid injuries, etc

02

Neural Networks applied to Health

The use of MCUs as edge inference devices for healthcare wearables, emphasize the need of TinyML solutions.

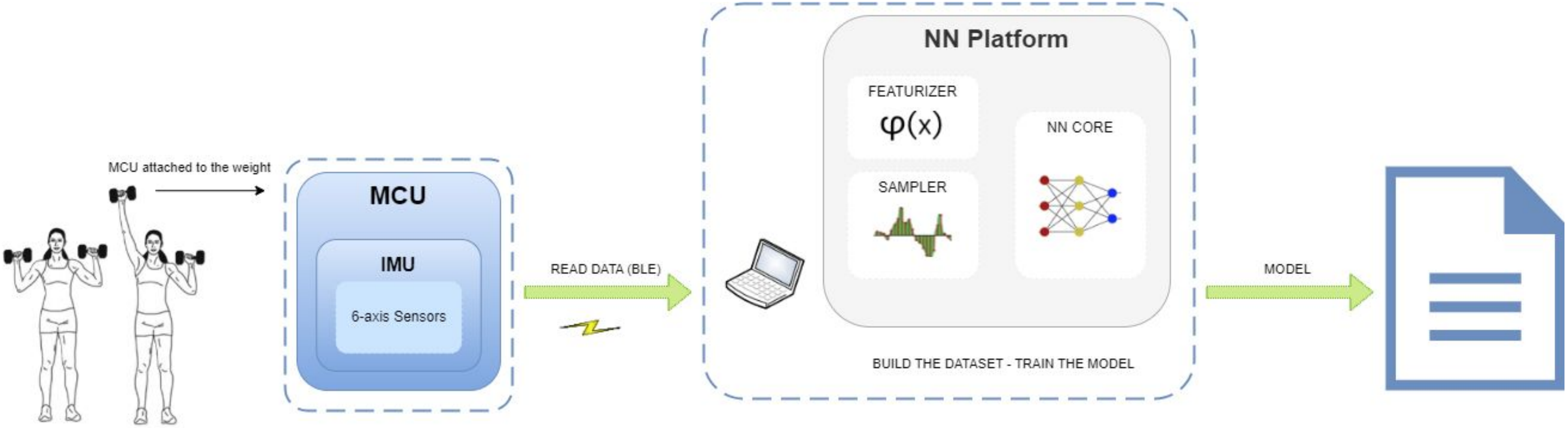
03

Generic Features

Use classification based on a set of generic features that were calculated from the sensor data to determine movements recognition.

Solution Outline

01 **Data collection and Model creation**

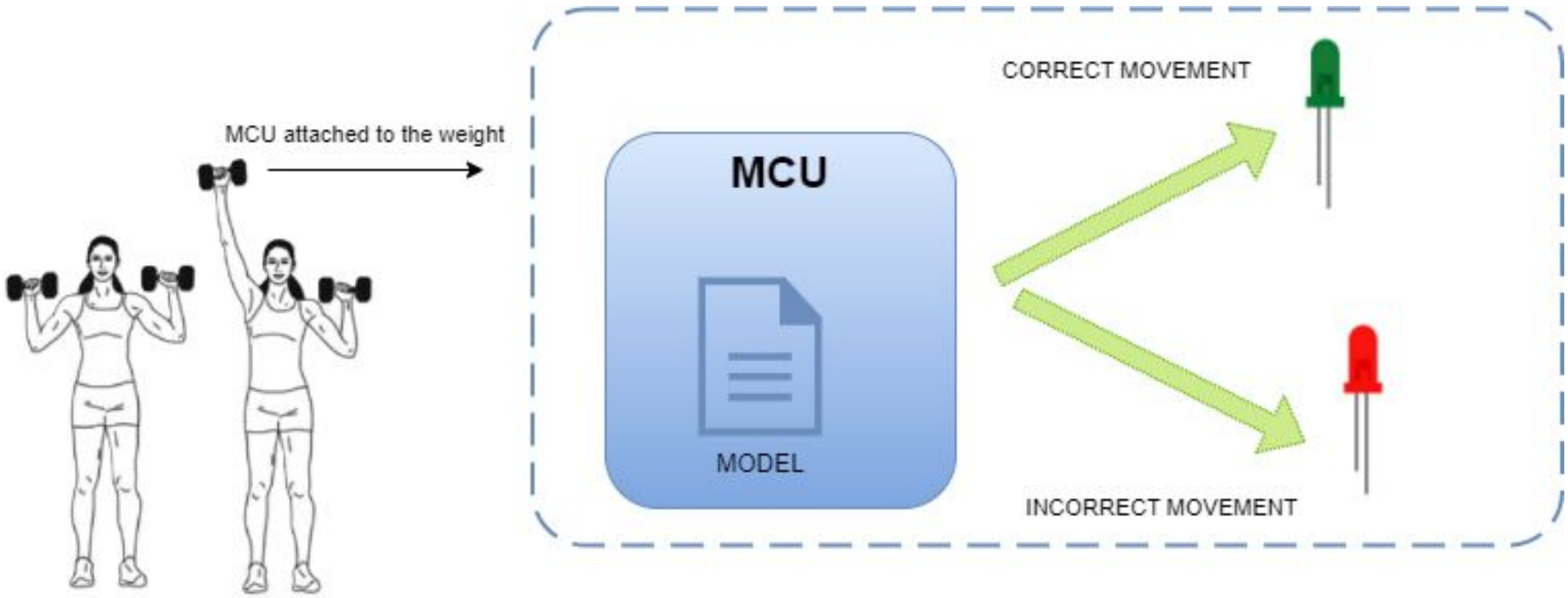


Solution Outline

02 MCU Code

Solution Outline

03 Solution Outline



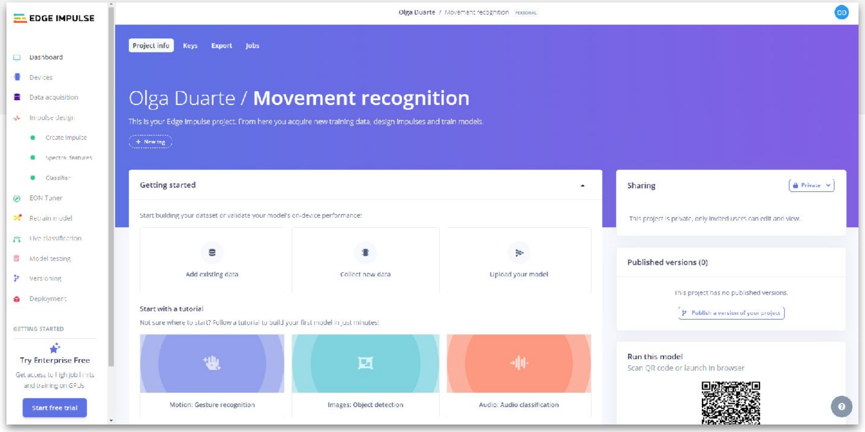
Low and No Code Platforms to train NN

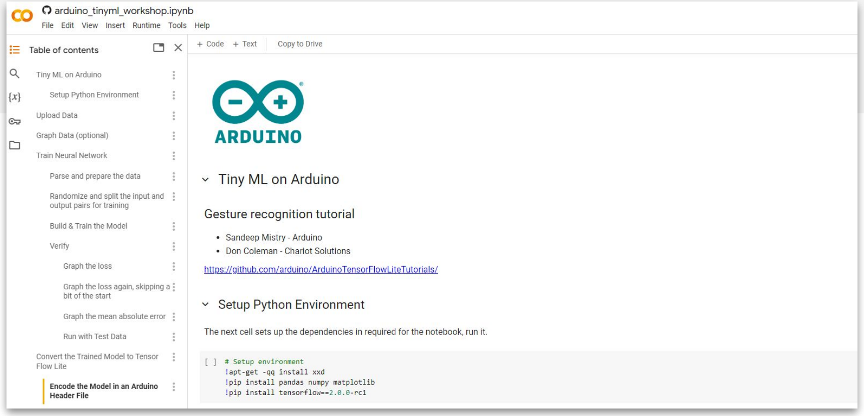
Edge Impulse

- No code platform.
- No cost.
- Integrates with small portable MCUs.
- Uses TensorFlow Lite for training, optimizing, and deploying deep learning models to embedded devices.

Google Colab

- Development platform.
- No cost.
- Designed to run ML models on MCUs and other devices with only few kilobytes of memory.
- TensorFlow Lite can be use.





Relevant Technologies on Movement Recognition

01

Accelerometer & Gyroscope

Acc detects linear acceleration of devices, that is, the acceleration along an axis. While gyro detects the angular velocity, i.e, how fast the body is turning.

02

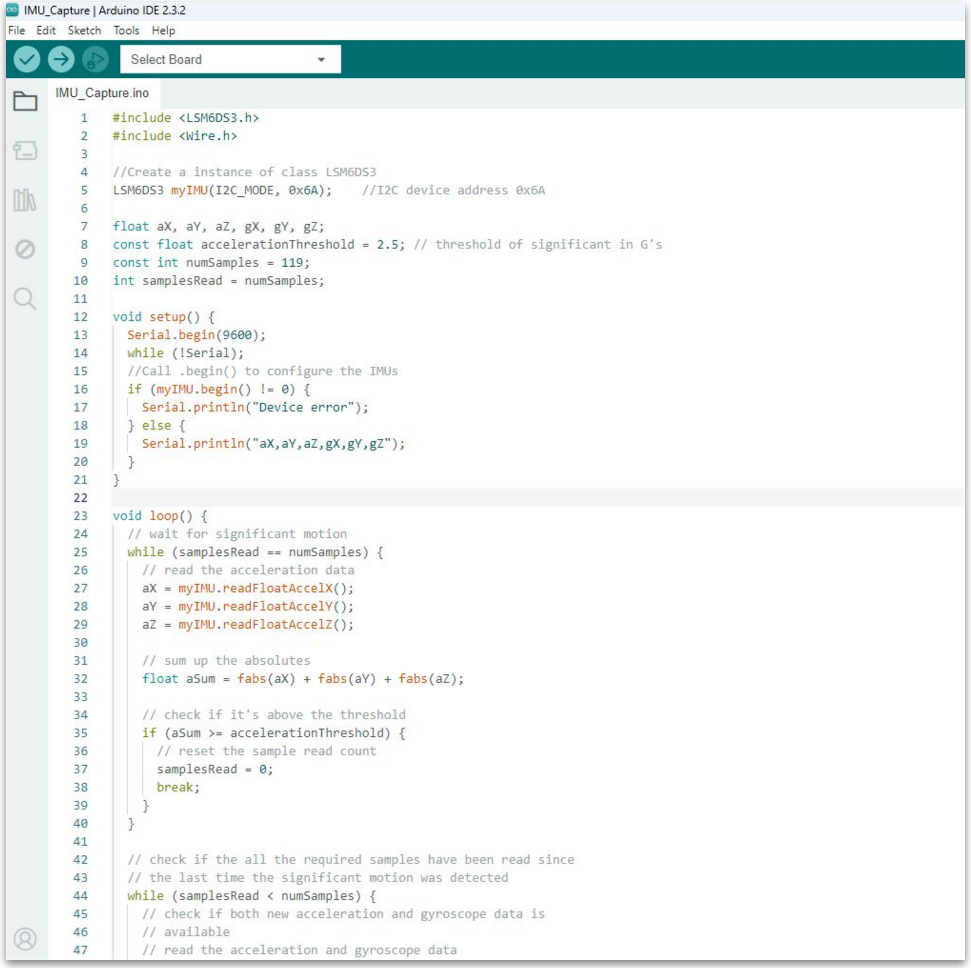
Ultra Low Power Embedded System

Capture, measure and report acceleration, orientation and other gravitational forces.

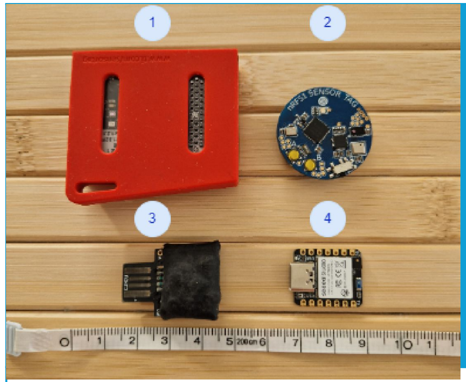
03

Bluetooth Low Energy

Wireless, low-power personal area network. Its goal is to connect devices over a relatively short range.



Ultra Low Power Embedded System

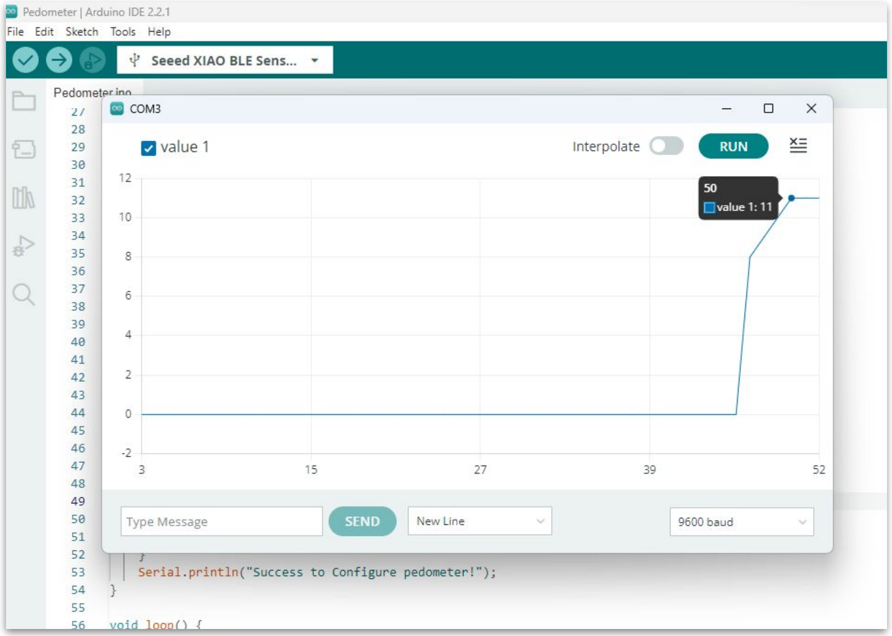
	NRF51 Sensor Tag	Seeed Studio XIAO nRF52840 Sense	CJMCU Beetle	Texas Instruments TIDC-CC2650ST K-SENSORTAG
Advantages	Low cost, 3 axis Accelerometer Sensor, BLE 5.0, Low Power Consumption Bluetooth.	Low cost, Low Power, BLE 5, Great documentation, not expensive, easy to start use and program, IMU with extra capabilities - like pedometer.	Low-power and extended-range Capabilities.	Advanced debugging and profiling Tools.
Disadvantages	Excluded due to the lack of examples and public documentation.	Pay more for the connectivity	Doesn’t have built-in Bluetooth capabilities.	Excluded due to the size, price and the learning curve of the software use to Develop.

Preliminary Results

01

Pedometer experience

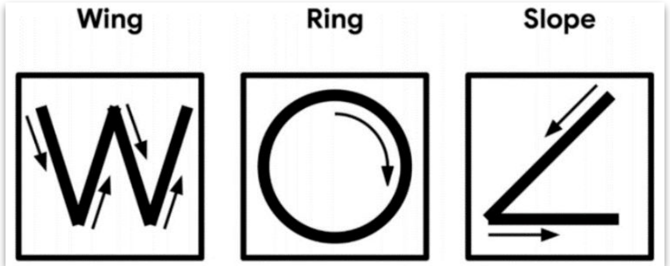
Step counter, collecting the data from the sensor.



02

Magic Wand – Tiny ML

The goal was to analyze the 3-axis returns in different movements

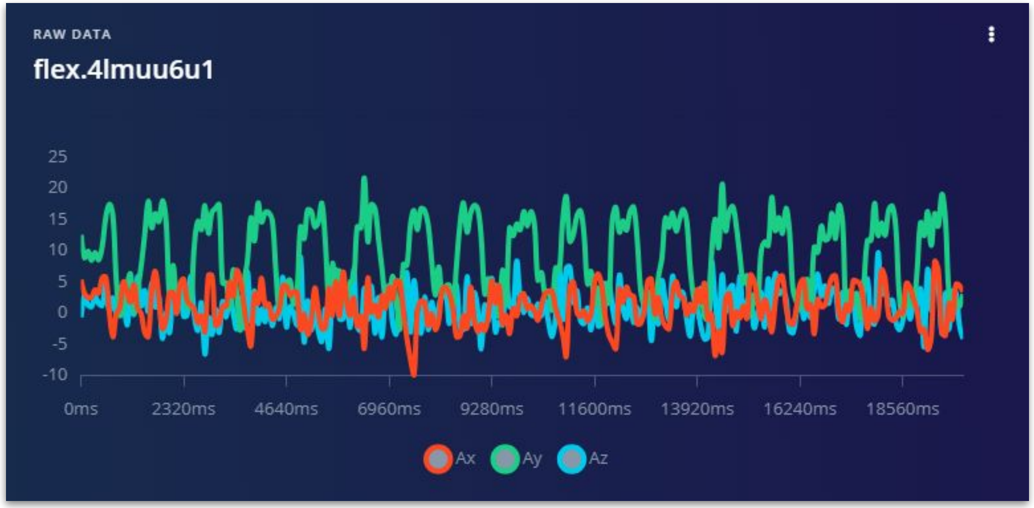


Preliminary Results

03

Evaluation of TensorFlow Lite using the Edge Impulse Platform

The NN was trained using Edge impulse and the model was uploaded to the MCU using TensorFlow Lite (in Arduino).

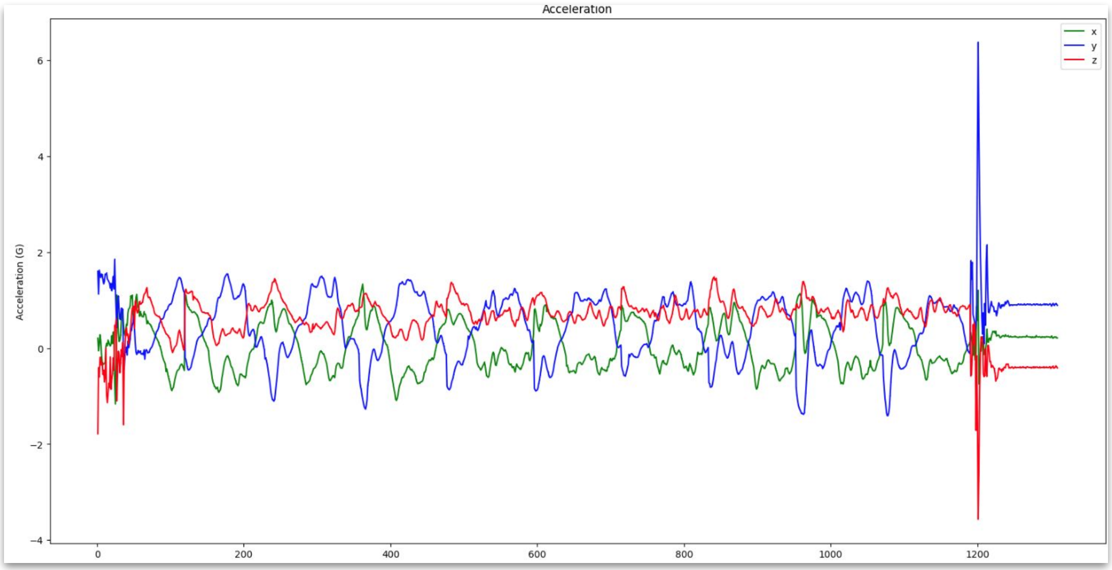


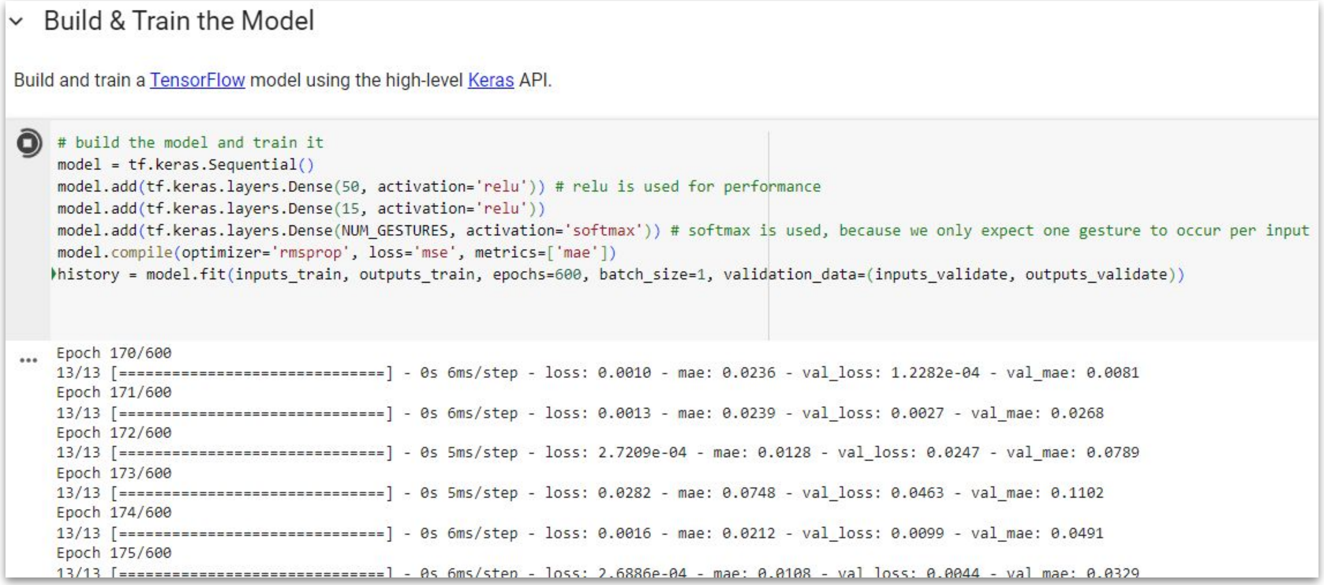
Preliminary Results

04

Evaluation of TensorFlow Lite Using the Google Colab Platform

The NN was trained using a notebook (python code) in Google Colab and the model was included in the MCU using TensorFlow Lite.





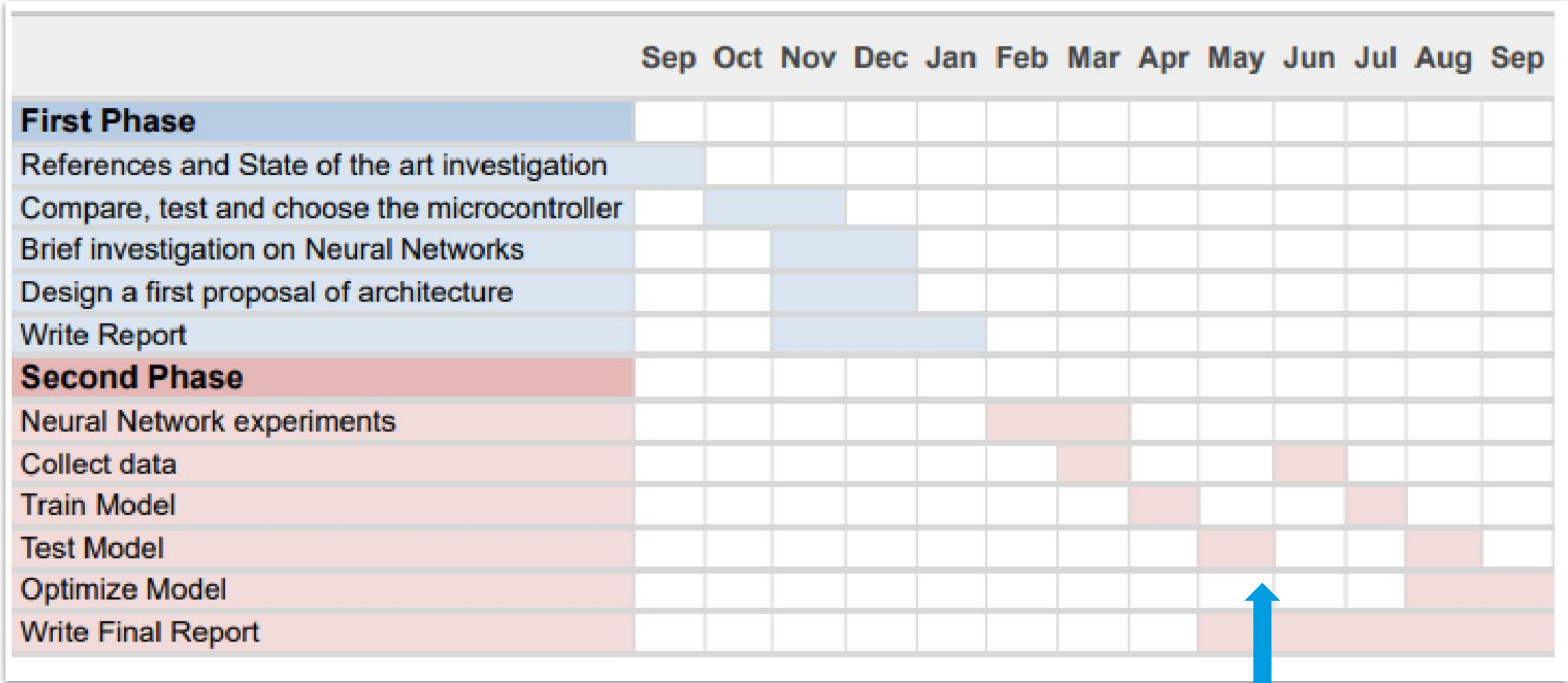
Build & Train the Model

Build and train a TensorFlow model using the high-level Keras API.

```
# build the model and train it
model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(50, activation='relu')) # relu is used for performance
model.add(tf.keras.layers.Dense(15, activation='relu'))
model.add(tf.keras.layers.Dense(NUM_GESTURES, activation='softmax')) # softmax is used, because we only expect one gesture to occur per input
model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
history = model.fit(inputs_train, outputs_train, epochs=600, batch_size=1, validation_data=(inputs_validate, outputs_validate))
```

```
Epoch 170/600
13/13 [==============================] - 0s 6ms/step - loss: 0.0010 - mae: 0.0236 - val_loss: 1.2282e-04 - val_mae: 0.0081
Epoch 171/600
13/13 [==============================] - 0s 6ms/step - loss: 0.0013 - mae: 0.0239 - val_loss: 0.0027 - val_mae: 0.0268
Epoch 172/600
13/13 [==============================] - 0s 5ms/step - loss: 2.7209e-04 - mae: 0.0128 - val_loss: 0.0247 - val_mae: 0.0789
Epoch 173/600
13/13 [==============================] - 0s 5ms/step - loss: 0.0282 - mae: 0.0748 - val_loss: 0.0463 - val_mae: 0.1102
Epoch 174/600
13/13 [==============================] - 0s 6ms/step - loss: 0.0016 - mae: 0.0212 - val_loss: 0.0099 - val_mae: 0.0491
Epoch 175/600
13/13 [==============================] - 0s 6ms/step - loss: 2.6886e-04 - mae: 0.0108 - val_loss: 0.0044 - val_mae: 0.0329
```

Work Plan



Here

Challenges and Limitations

Challenges

Program a NN to run on the selected MCU

Optimize the construction of the Model due to the limitations

Improve battery usage

Limitations

Ability of optimize the results of the NN Platforms used

Limited Storage

Limited RAM

Conclusions

When I accepted this challenge the first step was to analyze the state of the art and existing solutions and how I was going to compose my own. A preliminary evaluation of the solution was found and explored.

Next Steps:

- Improve the collect of the data.
- Improve the Training of the NN.
- Optimization of the network using *TinyML* (*TensorFlow*).

THANK YOU!

Q&A