



**ISEL**  
INSTITUTO SUPERIOR DE  
ENGENHARIA DE LISBOA

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

# Intelligent Sports Weights



*TFM11 - Apresentação intermédia*  
*Dissertação de Mestrado 2023/2024*

Olga dos Santos Duarte, N° 27675

# Presentation Outline

Introduction & Motivation	03
Objectives	04
Related Works on Movement Recognition	05
Relevant Technologies on Movement Recognition	10
Low and No Code Platforms to train NN	11
Ultra Low Power Embedded System	12
Solution Outline	13
Preliminary Results	16
Work Plan	19
Conclusions	20

# Introduction & Motivation

- Build an embedded system in a microcontroller with a trained Neural Network, using TinyM, that can detect if an exercise is correctly done.
- Use of low code or no code platform.
- Program a neural network on a low power microcontroller.

01

## Safe Physical Exercise

Identify the correctness of fitness exercises to avoid injury.

02

## Embedded Systems

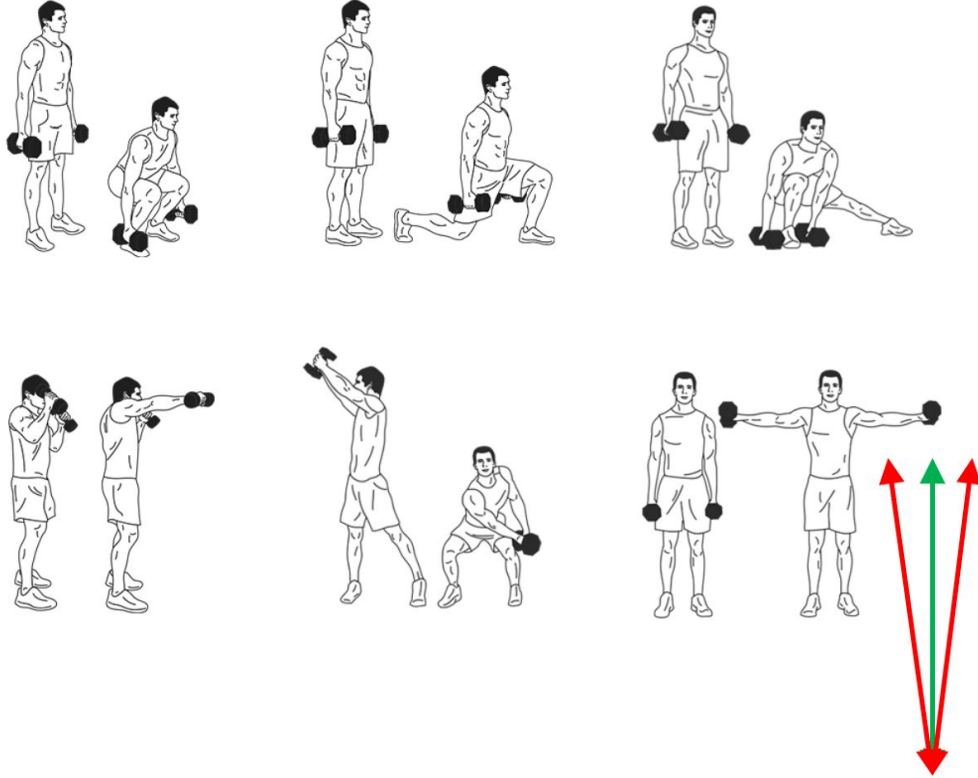
Autonomous, low-power, small size

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

## Human Activity Recognition

***S. Gupta, “A tinyml approach to human activity recognition,” 2022.***

It traces the evolution from traditional ML techniques to deep learning for HAR, highlighting the advantages of deep learning ability to learn from raw data.

Talks about the relevance of TinyML focusing on its power-saving features and potential to revolutionize HAR by enabling efficient, low-power and responsive systems.

***F. M. Rueda, R. Grzeszick, G. A. Fink, S. Feldhorst, and ten Michael Hompel, “Convolutional neural networks for human activity recognition using body-worn sensors,” 2018.***

This study presents an evaluation of a CNN architecture for HAR using multichannel time-series data from body-worn sensors, specifically IMUs, for recognizing human activities using data from those sensors.

This NN is designed to better capture how people move by considering information from different sensors.

# Related Works on Movement Recognition

02

## Neural Networks applied to Health

*M. S. Diab and E. Rodriguez-Villegas, “Embedded machine learning using microcontrollers in wearable and ambulatory systems for health and care applications: A review,” 2022.*

The article discusses the integration of ML into health and care applications, with a focus on wearable devices, using MCUs due to their advantages in terms of low power consumption, latency, size, flexibility, and cost.

Some challenges in designing healthcare wearables are indicated, such as user acceptability, compact design, ease of use, and minimal maintenance, data transmission, storage, security and power consumption.

The deployment of machine learning algorithms on resource-constrained embedded devices, offers advantages in latency, power efficiency and privacy compared to cloud computing.

The study explores the application of TinyML in wearable health and care systems, uncovering various applications, including medical condition detection, fitness tracking, elderly fall detection and rehabilitation with prosthetics.

# Related Works on Movement Recognition

03

## 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, skateboard, activity recognition in beach volleyball using a Deep Convolutional Neural Network - used also to avoid injuries, etc.

***J. M. Jarning, K.-M. Mok, B. H. Hansen, and R. Bahr, “Application of a tri-axial accelerometer to estimate jump frequency in volleyball,” 2015.***

Attempted to determine the jump frequency in volleyball to understand and prevent patellar tendinopathy, a disease also known as Jumper’s knee. Using an accelerometer data to determine the jump frequency.

***T. Kautz, B. H. Groh, J. Hannink, U. Jensen, H. Strubberg, and B. M. Eskofier, “Activity recognition in beach volleyball using a deep convolutional neural network,” 2017.***

Monitoring system for beach volleyball utilizing wrist-worn acceleration sensors to detect and classify 10 different player actions.

# Related Works on Movement Recognition

03

## Neural Networks applied to Sports

*B. H. Groh, T. Kautz, and D. Schuldhaus, “Imu based trick classification in skateboarding,” 2015.*

In this article they used IMUs, using wearable sensors to detect skateboard moves, using the IMU signal processing methods to classify specific activities.

IMUs have also been used for activity recognition in various other sports, e.g., in skiing, golf, etc.

*U. Jensen, M. Schmidt, M. Hennig, F. A. Dassler, T. Jaitner, and B. M. Eskofier, “An imu-based mobile system for golf putt analysis,” 2015.*



# Related Works on Movement Recognition

04

## Others

***Z. Wang, Y. Wu, Z. Jia, Y. Shi, and J. Hu, “Lightweight run-time working memory compression for deployment of deep neural networks on resource constrained,” 2021.***

In this article it's explained a compression algorithm to reduce the computational cost by removing certain filters on selected layers of DNNs, which could reduce the memory requirement of the corresponding layers at the same time. It was used a NN on resource constrained MCUs and the experimental results showed that without incurring heavy overhead on memory and run-time latency, the compressed NN could maintain the original accuracy or run with moderate accuracy loss.

***T. A. Gonçalves, “Convolutional Neural Network for Hand Gesture Identification on FPGAs,” Master's thesis, Nov. 2022.***

The goal was to have a Neural Network implemented in a FPGA. This project implements in an Arduino board a Convolution Neural Network (CNN) model that detects gestures of a person using a "wand".

In this master thesis, even if not related to sports, the goal remains the same - recognize gestures or movements, it was used an accelerometer placed in a wand to detect gestures from a person with the help of a NN model.

The and the model was trained using the TensorFlow Lite framework.

# 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

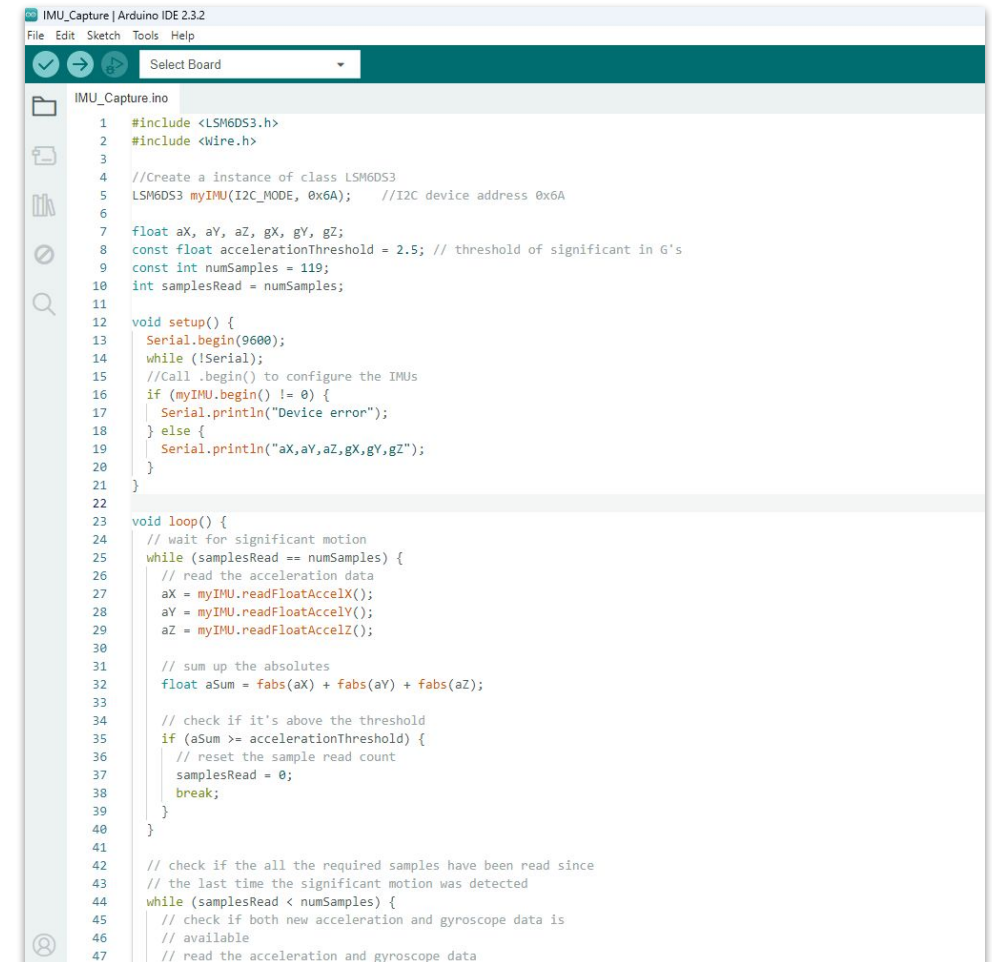
## Ultra Low Power Embedded System

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.



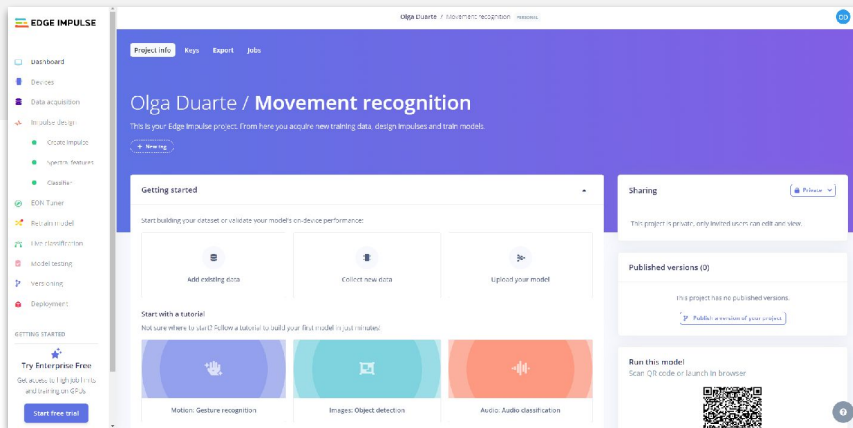
```
IMU_Capture | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Select Board

IMU_Capture.ino
1 #include <LSM6DS3.h>
2 #include <Wire.h>
3
4 //Create a instance of class LSM6DS3
5 LSM6DS3 myIMU(I2C_MODE, 0x6A); //I2C device address 0x6A
6
7 float aX, aY, aZ, gX, gY, gZ;
8 const float accelerationThreshold = 2.5; // threshold of significant in G's
9 const int numSamples = 119;
10 int samplesRead = numSamples;
11
12 void setup() {
13   Serial.begin(9600);
14   while (!Serial);
15   //Call .begin() to configure the IMUs
16   if (myIMU.begin() != 0) {
17     Serial.println("Device error");
18   } else {
19     Serial.println("aX,aY,aZ,gX,gY,gZ");
20   }
21 }
22
23 void loop() {
24   // wait for significant motion
25   while (samplesRead == numSamples) {
26     // read the acceleration data
27     aX = myIMU.readFloatAccelX();
28     aY = myIMU.readFloatAccelY();
29     aZ = myIMU.readFloatAccelZ();
30
31     // sum up the absolutes
32     float aSum = fabs(aX) + fabs(aY) + fabs(aZ);
33
34     // check if it's above the threshold
35     if (aSum >= accelerationThreshold) {
36       // reset the sample read count
37       samplesRead = 0;
38       break;
39     }
40   }
41
42   // check if the all the required samples have been read since
43   // the last time the significant motion was detected
44   while (samplesRead < numSamples) {
45     // check if both new acceleration and gyroscope data is
46     // available
47     // read the acceleration and gyroscope data
```

# Low and No Code Platforms to train NN

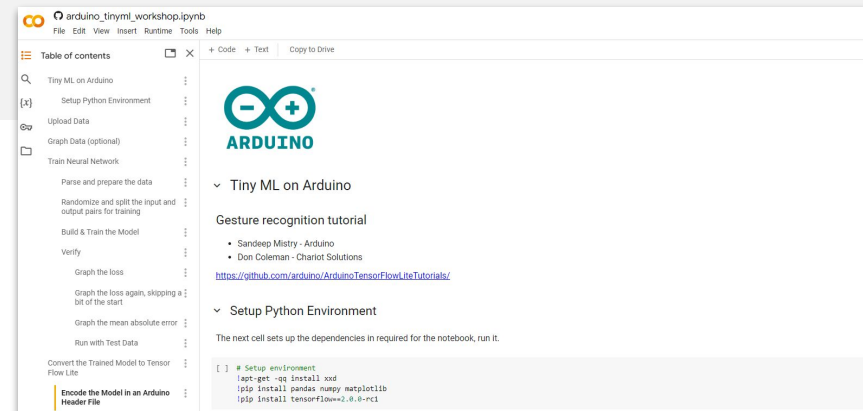
- No code platform.
- No cost.
- Integrates with small portable MCUs.
- Uses TensorFlow Lite for training, optimizing, and deploying deep learning models to embedded devices.

## Edge Impulse

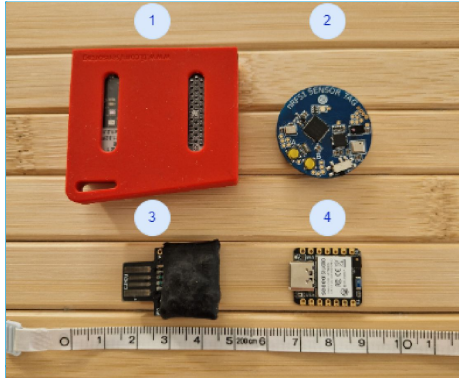


- Development platform.
- No cost.
- Designed to run ML models on MCUs and other devices with only few kilobytes of memory.
- TensorFlow Lite can be used.

## Google Colab



# Ultra Low Power Embedded System



## NRF51 Sensor Tag

## Seed 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

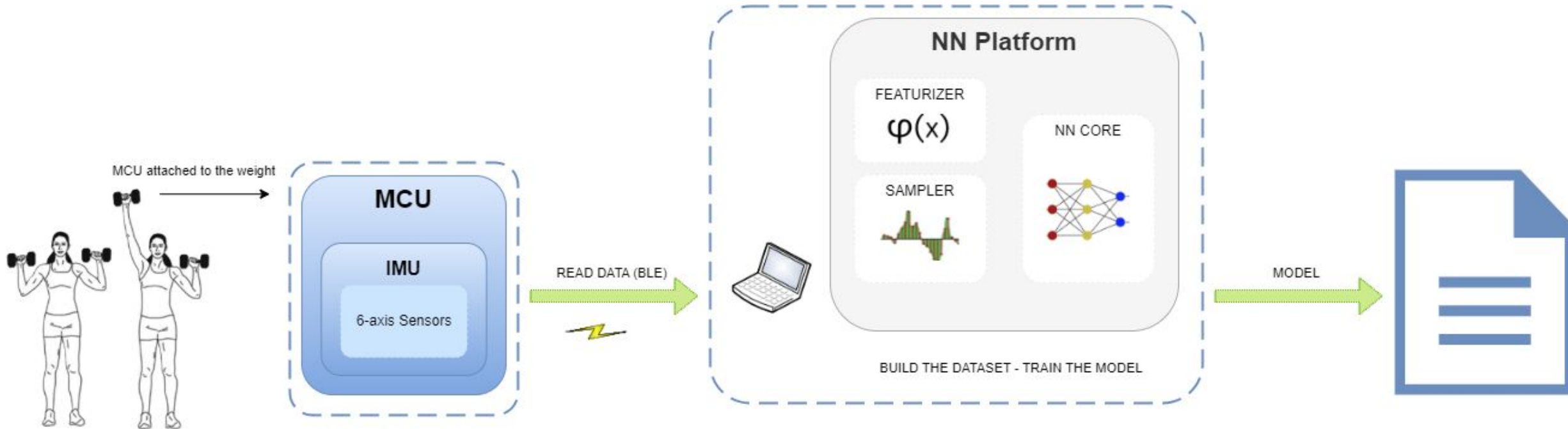
Doesn't have built-in Bluetooth capabilities.

Excluded due to the size, price and the learning curve of the software use to Develop.

# Solution Outline

01

## Data collection and Model creation



# Solution Outline

02

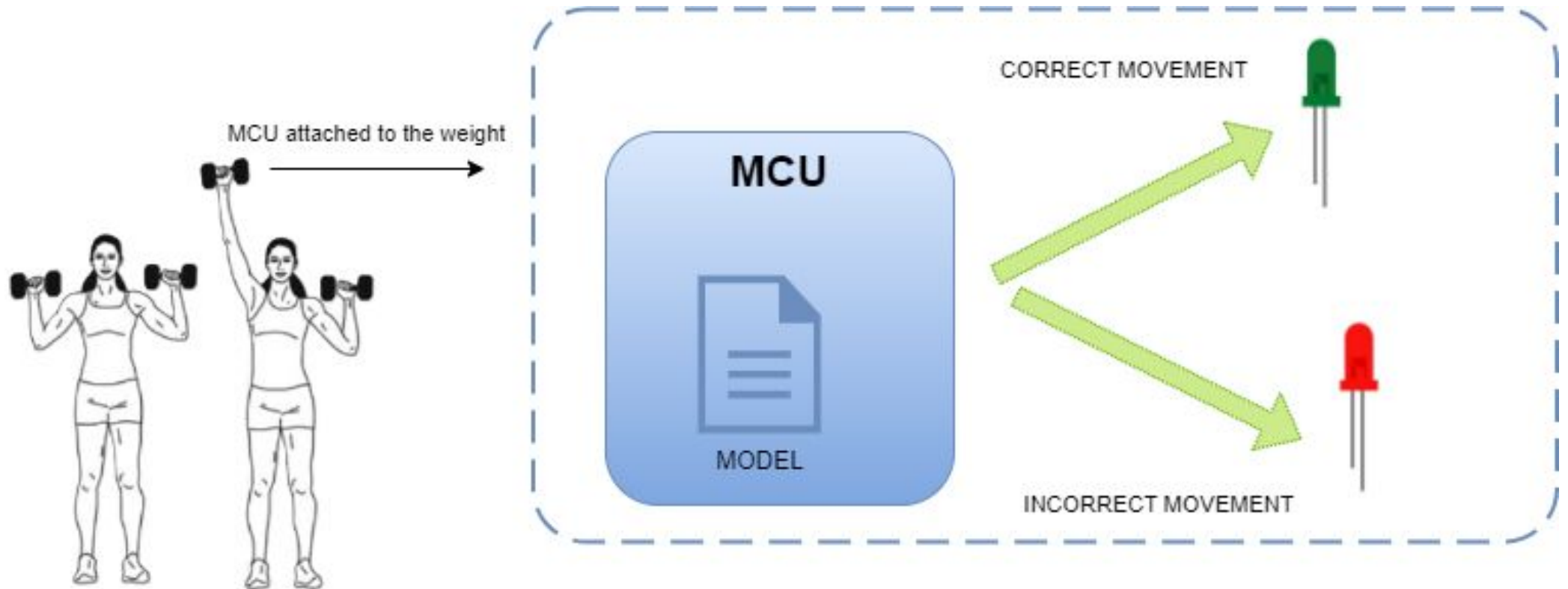
## MCU Code



# Solution Outline

03

## Solution Outline



# Preliminary Results

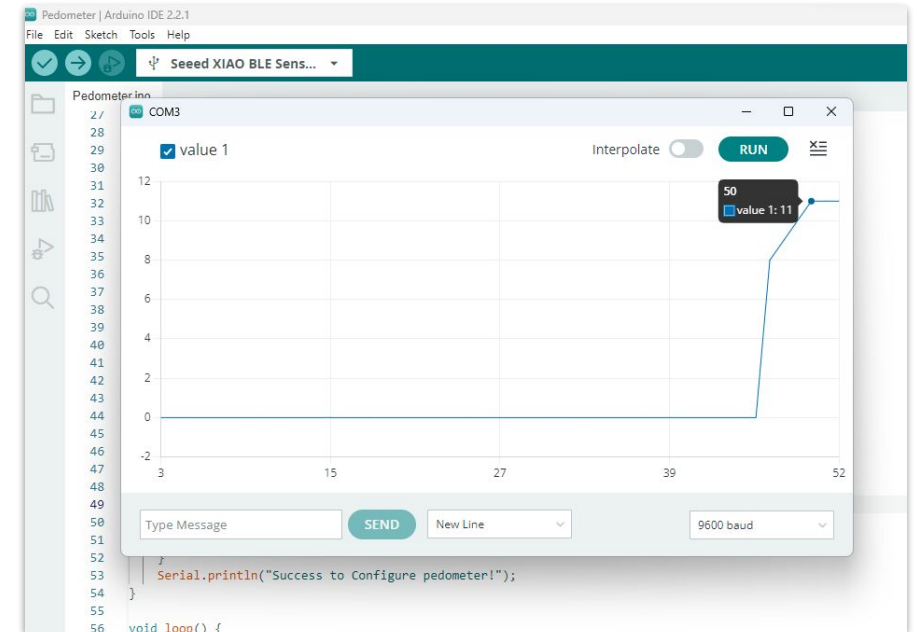
01

## Pedometer experience

Step counter, collecting the data from the sensor.

Confirm that the onboard sensors (such as the accelerometer and gyroscope) were working properly.

Check that the sensors provide accurate and reliable motion data that can be used for step detection.

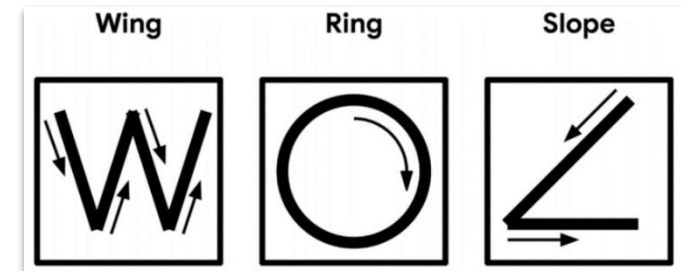


02

## Magic Wand – Tiny ML

The goal was to analyze the 3-axis (X, Y, and Z) returns in different movements.

Validate the values returned and classify the gestures based on the accelerometer data.





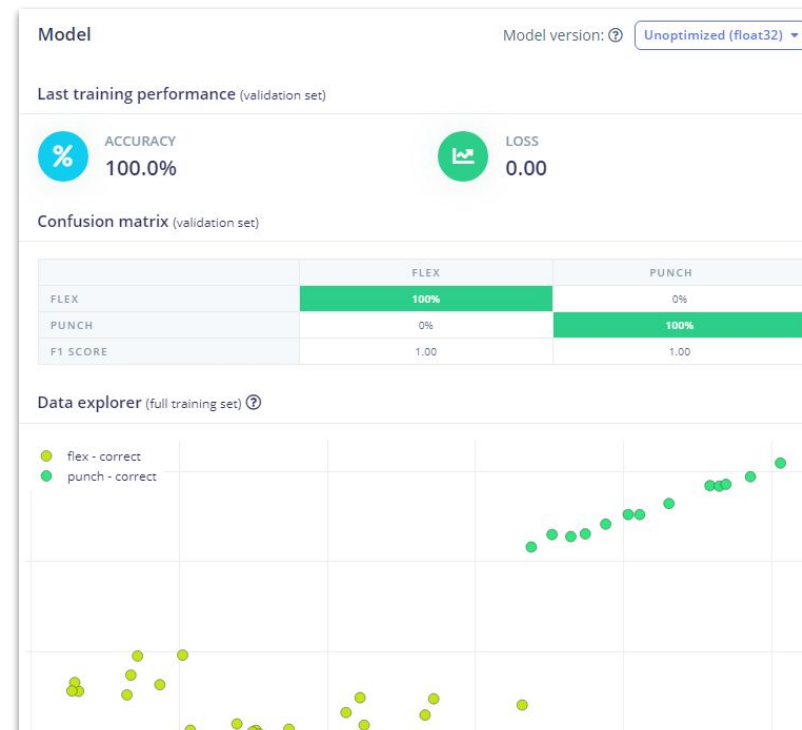
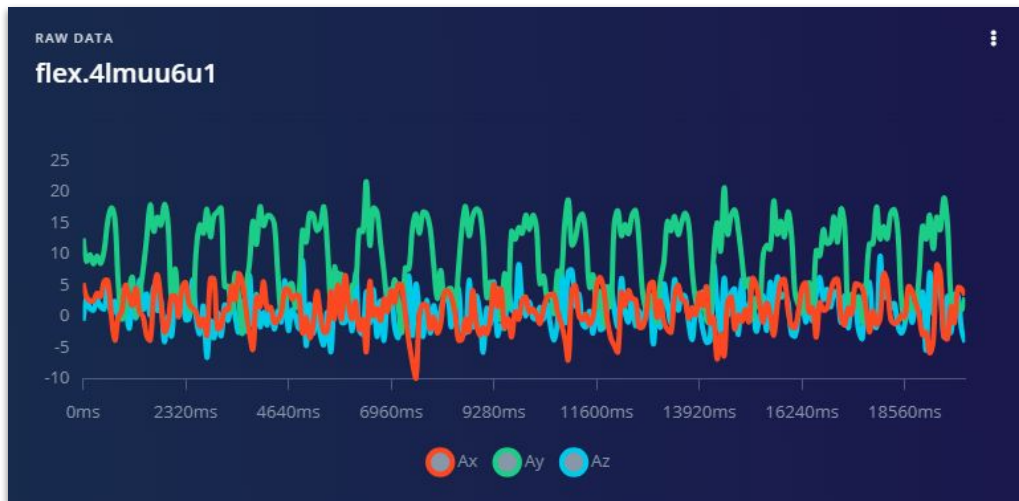
# Preliminary Results

03

## Evaluation of TensorFlow Lite using the Edge Impulse Platform

The data was capture using the IMU sensor to collect the accelerometer and gyroscope values, while performing a movement like flex or punch for 20 seconds each.

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



Output Serial Monitor X

Message (Enter to send message to)

```
punch: 0.020292  
flex: 0.979708  
  
punch: 0.035206  
flex: 0.964794  
  
punch: 0.005577  
flex: 0.994423  
  
punch: 0.000000  
flex: 1.000000
```

# Preliminary Results

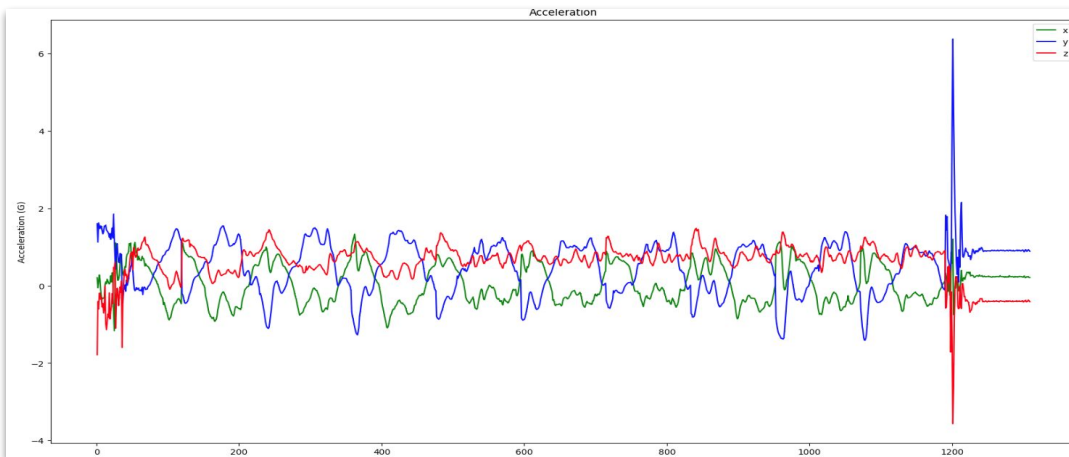
04

## Evaluation of TensorFlow Lite Using the Google Colab Platform

The data was captured using the IMU sensor to collect the accelerometer and gyroscope values, while performing a movement like flex or punch for 10 times each.

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

The downloaded TensorFlow Lite model file (model.h) was used to recognize the punch and flex actions from Seeed Studio XIAO nRF52840 Sense.



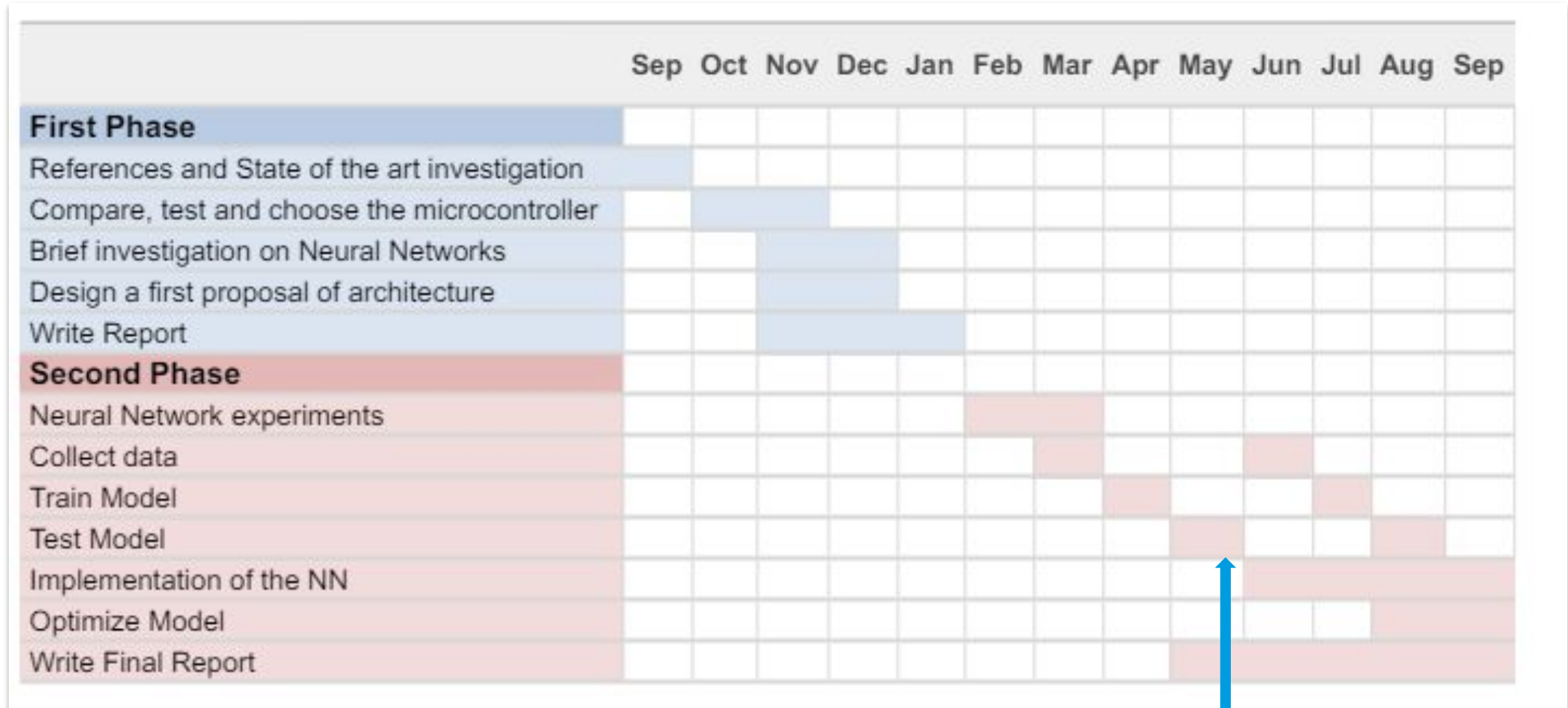
### Build & Train the Model

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

# Conclusions

Based on preliminary tests, I have validated that this solution can be effective for movements recognition, meeting the intended objectives.

I am currently studying neural networks and have some source code to use as a reference. This will guide my own implementation of an open-source neural network.

## **Next Steps:**

- Program a NN to run on the selected MCU.
- Optimize the Model due to limitations (such as limited storage and memory).
- Try to improve battery usage.

# Thank you!

---

## References:

1. Ang', T. A., & Gonç, A. (2022). *Convolutional Neural Network for Hand Gesture Identification on FPGAs Electrical and Computer Engineering*.
2. Banbury, C., Zhou, C., Fedorov, I., Navarro, R. M., Thakker, U., Gope, D., Reddi, V. J., Mattina, M., & Whatmough, P. N. (2021). *BAMBURY\_micronets-neural-network-architectures-for-deploying-tinyml-applications-on-commodity-microcontrollers-Paper\_2021*.
3. Brock, H., Ohgi, Y., & Lee, J. (2017). *Learning to Judge Like a Human: Convolutional Networks for Classification of Ski Jumping Errors*.
4. Buckley, C., O' Reilly, M. A., Farrel, A. V., Clark, L., & Longo, V. (2017). *Binary Classification of Running Fatigue using a Single Inertial Measurement Unit*.
5. Bulling, A., Blanke, U., & Schiele, B. (2014). *A Tutorial on Human Activity Recognition Using Body-Worn Inertial Sensors*.
6. Chowdhary, M., & Saha, S. S. (2023). *On-Sensor Online Learning and Classification Under 8 KB Memory*.
7. Cust Emily E, Sweeting Alice J, Ball Kevin, & Robertson Sam. (2018). *Machine and deep learning for sport-specific movement recognition: a systematic review of model development and performance*. <https://www.tandfonline.com/doi/full/10.1080/02640414.2018.1521769>
8. Diab, M. S., & Rodriguez-Villegas, E. (2022). *Embedded Machine Learning Using Microcontrollers in Wearable and Ambulatory Systems for Health and Care Applications: A Review*.
9. Groh, B. H., Kautz, T., & Schuldhaus, D. (2015). *IMU-based Trick Classification in Skateboarding*.
10. Gupta, S. (2022). *A TinyML Approach to Human Activity Recognition*.
11. Jarning, J. M., Mok, K.-M., Hansen, B. H., & Bahr, R. (2015). *Application of a tri-axial accelerometer to estimate jump frequency in volleyball*.
12. Jensen, U., Schmidt, M., Hennig, M., Dassler, F. A., Jaitner, T., & Eskofier, B. M. (2015). *An IMU-based Mobile System for Golf Putt Analysis*.
13. Kautz, T., Groh, B. H., Hannink, J., Jensen, U., Strubberg, H., & Eskofier, B. M. (2017). *Activity recognition in beach volleyball using a Deep Convolutional Neural Network*.
14. Lanraoui, N., & Touati, C. (2023). *Tiny ML for Gesture Recognition*.
15. Lattanzi, E., Donati, M., & Freschi, V. (2022). *Exploring Artificial Neural Networks Efficiency in Tiny Wearable Devices for Human Activity Recognition*. <https://www.mdpi.com/1424-8220/22/7/2637>
16. Munguia Tapia, E., Intille, S. S., Haskell, W., Larson, K., Wright, J., King, A., & Friedman, R. (2007). *Real-Time Recognition of Physical Activities and Their Intensities Using Wireless Accelerometers and a Heart Rate Monitor*.
17. Rindal, O. M. H., Seeberg, T. M., Tjønnås, J., Haugnes, P., & Sandbakk, Ø. (2017). *Automatic Classification of Sub-Techniques in Classical Cross-Country Skiing Using a Machine Learning Algorithm on Micro-Sensor Data*.
18. Rueda, F. M., Grzeszick, R., Fink, G. A., Feldhorst, S., & Hompel, ten M. (2018). *Convolutional Neural Networks for Human Activity Recognition Using Body-Worn Sensors*.
19. Sanchez-Iborra, R., & Skarmeta, A. (2021). *Who is wearing me? TinyDL-based user recognition in constrained personal devices*.
20. Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Published 2014 by Cambridge University Press.
21. Wang, Z., Wu, Y., Jia, Z., Shi, Y., & Hu, J. (2021). *Lightweight Run-Time Working Memory Compression for Deployment of Deep Neural Networks on Resource-Constrained*.
22. Warden Peter, & Situnayake Daniel. (2019). *TinyML - Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. [https://books.google.pt/books?hl=en&lr=&id=tn3EDwAAQBAJ&oi=fnd&pg=PP1&ots=qjllbr-7A4&sig=GAqcAA99f2EPyvOjOaddMu8gzss&redir\\_esc=y#v=onepage&q&f=false](https://books.google.pt/books?hl=en&lr=&id=tn3EDwAAQBAJ&oi=fnd&pg=PP1&ots=qjllbr-7A4&sig=GAqcAA99f2EPyvOjOaddMu8gzss&redir_esc=y#v=onepage&q&f=false)
23. Xia, S., de Godoy, D., Islam, B., Islam, M. T., Nirjon, S., Kinget, P. R., & Jiang, X. (2019). *Improving Pedestrian Safety in Cities using Intelligent Wearable Systems*.
24. Yauri, R., & Espino, R. (2022). *Edge device for movement pattern classification using neural network algorithms*.