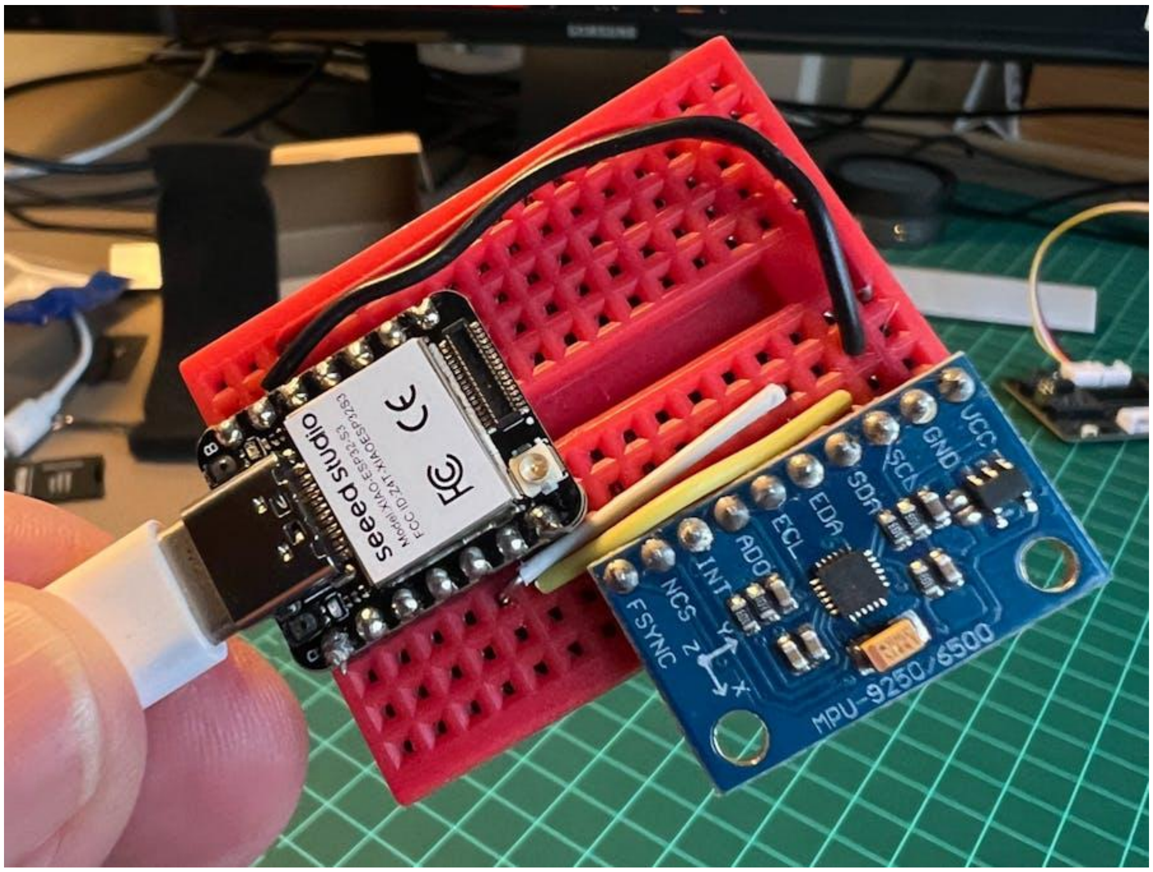
# TinyML Made Easy

# Motion Classification

Creating a TinyML Anomaly Detection & Motion Classification project with the Seeed XIAO ESP32S3 and IMU MPU6050



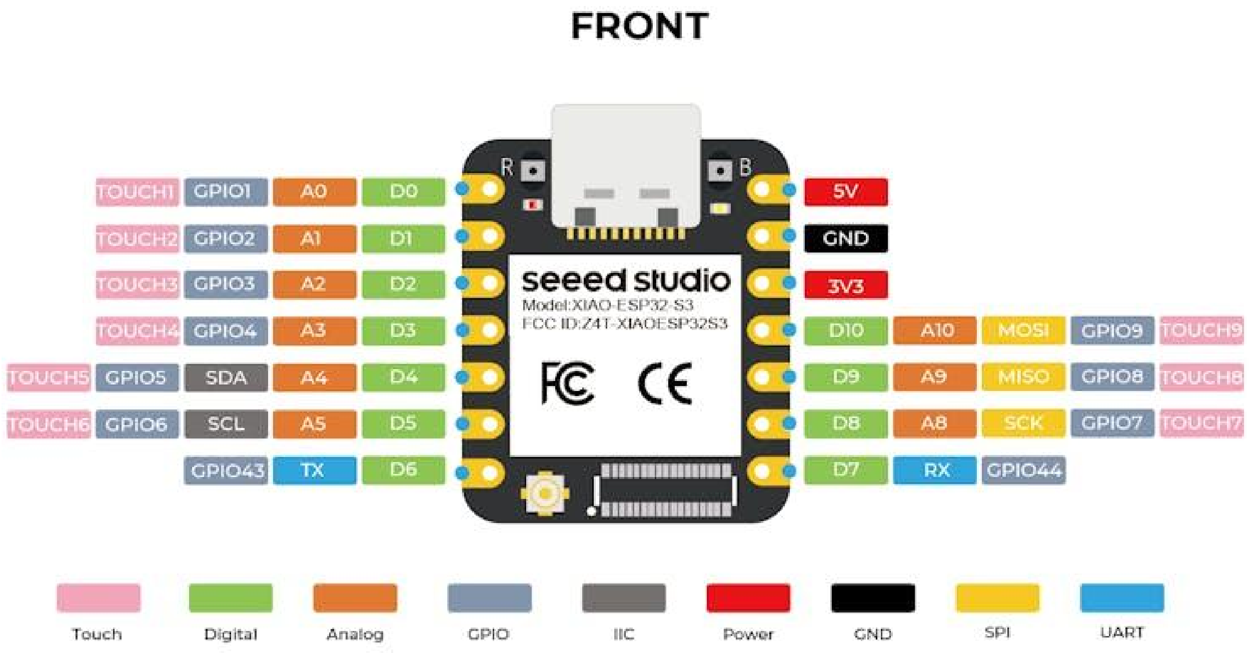
**MJRoBot (Marcelo Rovai)**

**Published May 16, 2023 © MIT**

https://www.hackster.io/mjrobot/exploring-machine-learning-with-the-new-xiao-esp32s3-6463e5

## Introduction

In my tutorial, TinyML Made Easy: Image Classification, we explored Image Classification on the new tiny device of the Seeed XIAO family, the ESP32S3 Sense. The Sense has a camera and a mic incorporated, but what happens if you want another type of sensor as an IMU? No problem! One great advantage of the XIAO ESP32S3 is its several pins available as an I2C bus (SDA/SCL pins).



Seeed Studio

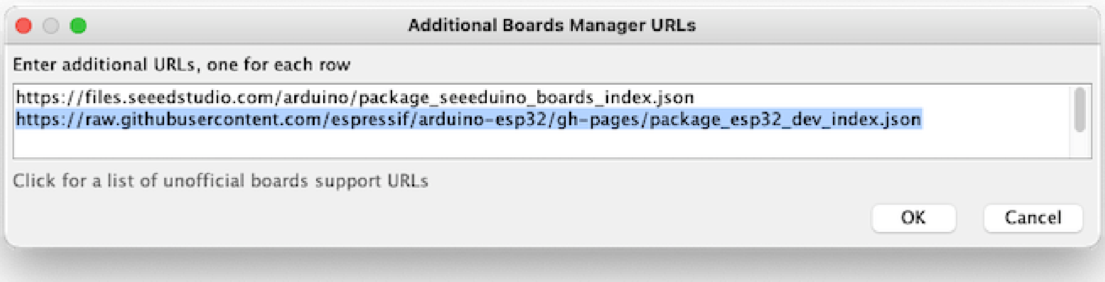
## Installing the XIAO ESP32S3 Sense on Arduino IDE

Following my last tutorial, you should have the device installed on the Arduino IDE. If not, let's do a quick review:

On Arduino IDE, navigate to **File > Preferences**, and fill in the URL:

`https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json`

on the field ==> **Additional Boards Manager URLs**

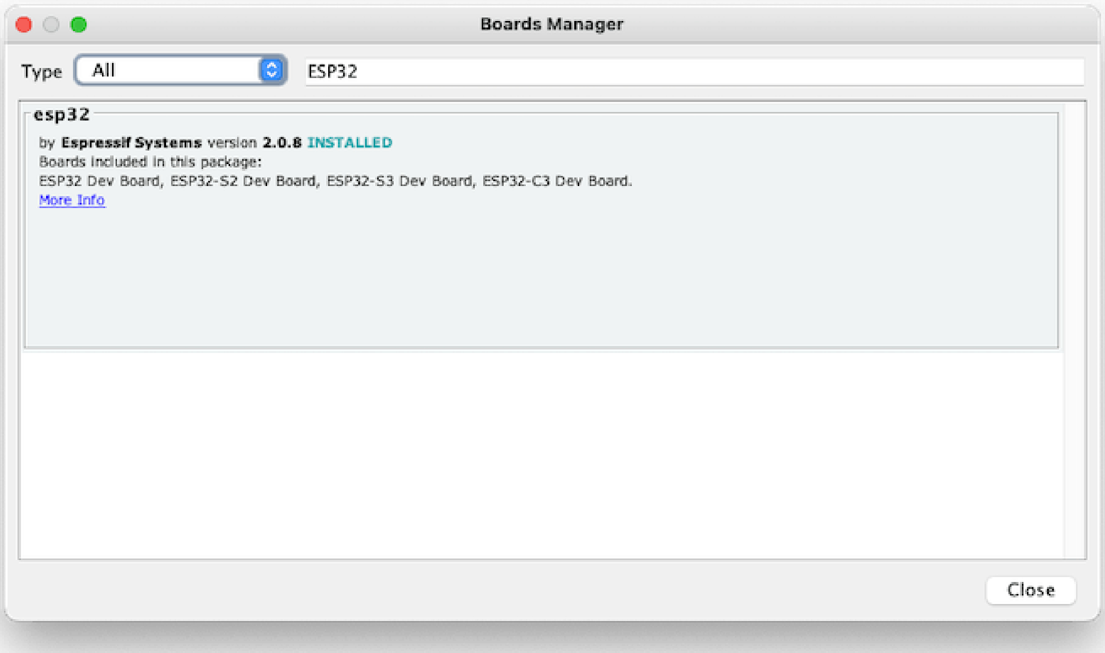

Additional Boards Manager URLs

Enter additional URLs, one for each row

```
https://files.seeedstudio.com/arduino/package_seeeduino_boards_index.json
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json
```

Click for a list of unofficial boards support URLs

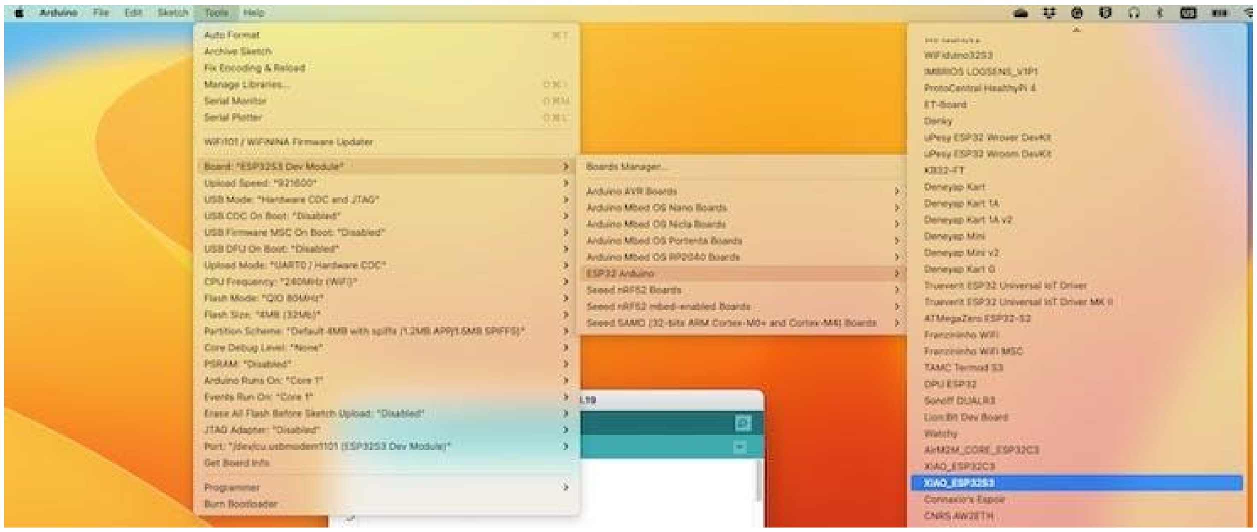
Next, open boards manager. Go to **Tools** > **Board** > **Boards Manager…** and enter with *esp32.* Select and install the most updated package:


Boards Manager

Type: All — ESP32

esp32
by **Espressif Systems** version **2.0.8** INSTALLED
Boards included in this package:
ESP32 Dev Board, ESP32-S2 Dev Board, ESP32-S3 Dev Board, ESP32-C3 Dev Board.
More Info

On **Tools**, select the Board (**XIAO ESP32S3**):


Board: "ESP32S3 Dev Module"
Upload Speed: "921600"
USB Mode: "Hardware CDC and JTAG"
USB CDC On Boot: "Disabled"
USB Firmware MSC On Boot: "Disabled"
USB DFU On Boot: "Disabled"
Upload Mode: "UART0 / Hardware CDC"
CPU Frequency: "240MHz (WiFi)"
Flash Mode: "QIO 80MHz"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Arduino Runs On: "Core 1"
Events Run On: "Core 1"
Erase All Flash Before Sketch Upload: "Disabled"
JTAG Adapter: "Disabled"
Port: "/dev/cu.usbmodem1101 (ESP32S3 Dev Module)"

Last, but not least, select the **Port** where the ESP32S3 is connected.

That is it! The device should be OK. To be sure, run the Blink sketch.

## Installing the IMU

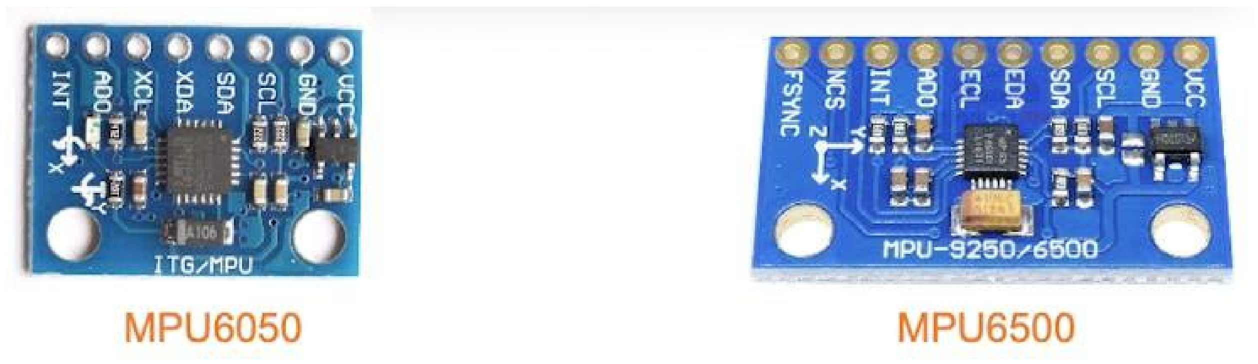
You could select your IMU from several devices found in the market, such as ADXL362 (3-axis), MAX21100 (6-axis), MPU6050 (6-axis), LIS3DHTR (3-axis), or the LCM20600 (6-axis), which is part of the Seeed Grove - IMU 9DOF (lcm20600+AK09918).

For this project, we will use an IMU, the MPU6050 (or 6500), and a low-cost (less than 2.00 USD) 6-Axis Accelerometer/Gyroscope unit.

In conclusion, I will also comment about using the LCM20600

The MPU-6500 is a 6-axis Motion Tracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion ProcessorTM (DMP) in a small 3x3x0.9mm package. It also features a 4096-byte FIFO that can lower the traffic on the serial bus interface and reduce power consumption by allowing the system processor to burst read sensor data and then go into a low-power mode.

With its dedicated I2C sensor bus, the MPU-6500 directly accepts inputs from external I2C devices. MPU-6500, with its 6-axis integration, on-chip DMP, and run-time calibration firmware, enables manufacturers to eliminate the costly and complex selection, qualification, and system-level integration of discrete devices, guaranteeing optimal motion performance for consumers. MPU-6500 is also designed to interface with multiple non-inertial digital sensors, such as pressure sensors, on its auxiliary I2C port.



I2C - Accelerometer & Gyroscope - I2C/SPI

Usually, the libraries available are for MPU6050, but they work for both devices.

**Connecting the HW**

Connect the IMU to the XIAO according to the below diagram:

- MPU6050 **SCL** --> XIAO **D5**
- MPU6050 **SDA** --> XIAO **D4**
- MPU6050 **VCC** --> XIAO **3.3V**
- MPU6050 **GND** --> XIAO **GND**

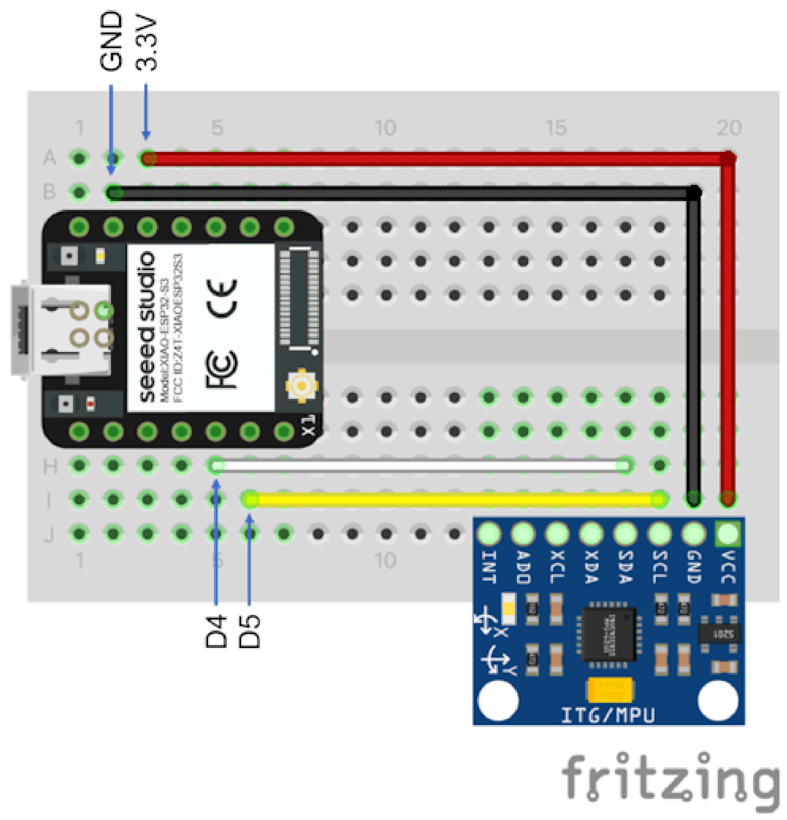
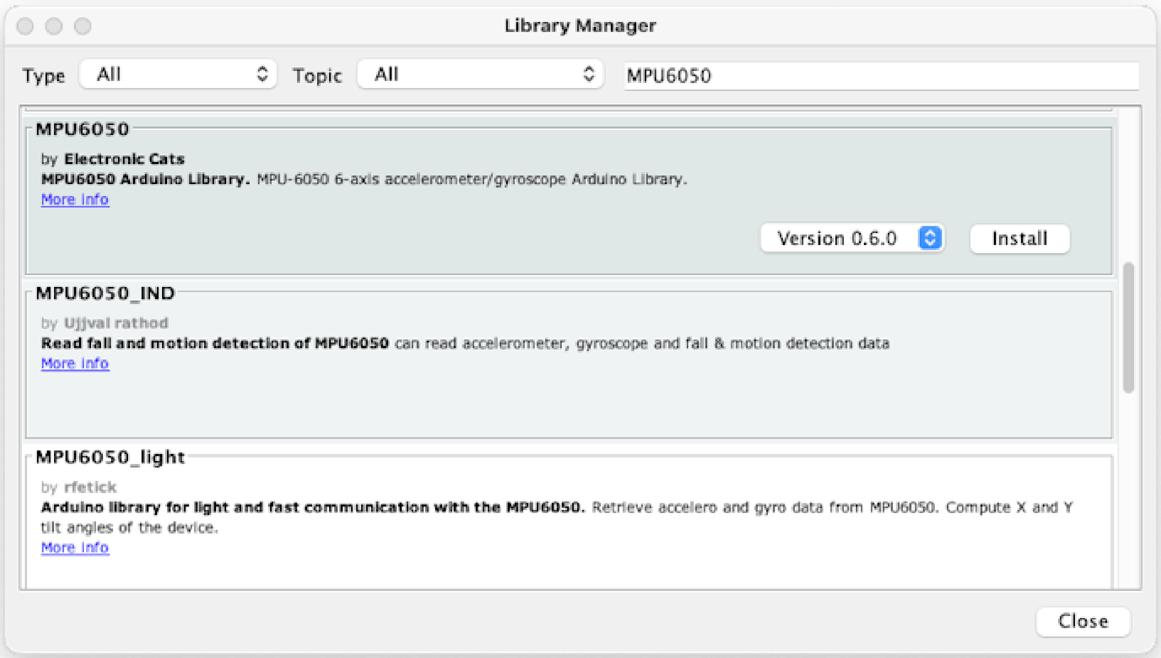


Image from author

## Install the Library

Go to Arduino Library Manager and type MPU6050. Install the latest version.


Library Manager

Type: All; Topic: All; MPU6050

MPU6050
by Electronic Cats
**MPU6050 Arduino Library.** MPU-6050 6-axis accelerometer/gyroscope Arduino Library.
More info

MPU6050_IND
by Ujjval rathod
**Read fall and motion detection of MPU6050** can read accelerometer, gyroscope and fall & motion detection data
More info

MPU6050_light
by rfetick
**Arduino library for light and fast communication with the MPU6050.** Retrieve accelero and gyro data from MPU6050. Compute X and Y tilt angles of the device.
More info

Download the sketch `MPU6050_Acc_Data_Acquisition.in:`

```
/*
* Based on I2C device class (I2Cdev) Arduino sketch for MPU6050 class by Jeff
Rowberg <jeff@rowberg.net>
* and Edge Impulse Data Forwarder Exampe (Arduino) -
https://docs.edgeimpulse.com/docs/cli-data-forwarder
*
* Developed by M.Rovai @11May23
*/

#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

#define FREQUENCY_HZ        50
#define INTERVAL_MS         (1000 / (FREQUENCY_HZ + 1))
#define ACC_RANGE           1 // 0: -/+2G; 1: +/-4G

// convert factor g to m/s2 ==> [-32768, +32767] ==> [-2g, +2g]
#define CONVERT_G_TO_MS2    (9.81/(16384.0/(1.+ACC_RANGE)))

static unsigned long last_interval_ms = 0;
```

```
MPU6050 imu;
int16_t ax, ay, az;

void setup() {

   Serial.begin(115200);


   // initialize device
   Serial.println("Initializing I2C devices...");
   Wire.begin();
   imu.initialize();
   delay(10);

//    // verify connection
//    if (imu.testConnection()) {
//      Serial.println("IMU connected");
//    }
//    else {
//      Serial.println("IMU Error");
//    }
   delay(300);

   //Set MCU 6050 OffSet Calibration
   imu.setXAccelOffset(-4732);
   imu.setYAccelOffset(4703);
   imu.setZAccelOffset(8867);
   imu.setXGyroOffset(61);
   imu.setYGyroOffset(-73);
   imu.setZGyroOffset(35);

   /* Set full-scale accelerometer range.
    * 0 = +/- 2g
    * 1 = +/- 4g
    * 2 = +/- 8g
    * 3 = +/- 16g
    */
   imu.setFullScaleAccelRange(ACC_RANGE);
}

void loop() {

     if (millis() > last_interval_ms + INTERVAL_MS) {
       last_interval_ms = millis();

       // read raw accel/gyro measurements from device
       imu.getAcceleration(&ax, &ay, &az);
```

```
        // converting to m/s2
        float ax_m_s2 = ax * CONVERT_G_TO_MS2;
        float ay_m_s2 = ay * CONVERT_G_TO_MS2;
        float az_m_s2 = az * CONVERT_G_TO_MS2;

        Serial.print(ax_m_s2);
        Serial.print("\t");
        Serial.print(ay_m_s2);
        Serial.print("\t");
        Serial.println(az_m_s2);
      }
}
```

**Some comments about the code:**

Note that the values generated by the accelerometer and gyroscope have a range: [-32768, +32767], so for example, if the default accelerometer range is used, the range in Gs should be: [-2g, +2g]. So, "1G" means 16384.

For conversion to m/s2, for example, you can define the following:

```
#define CONVERT_G_TO_MS2 (9.81/16384.0)
```

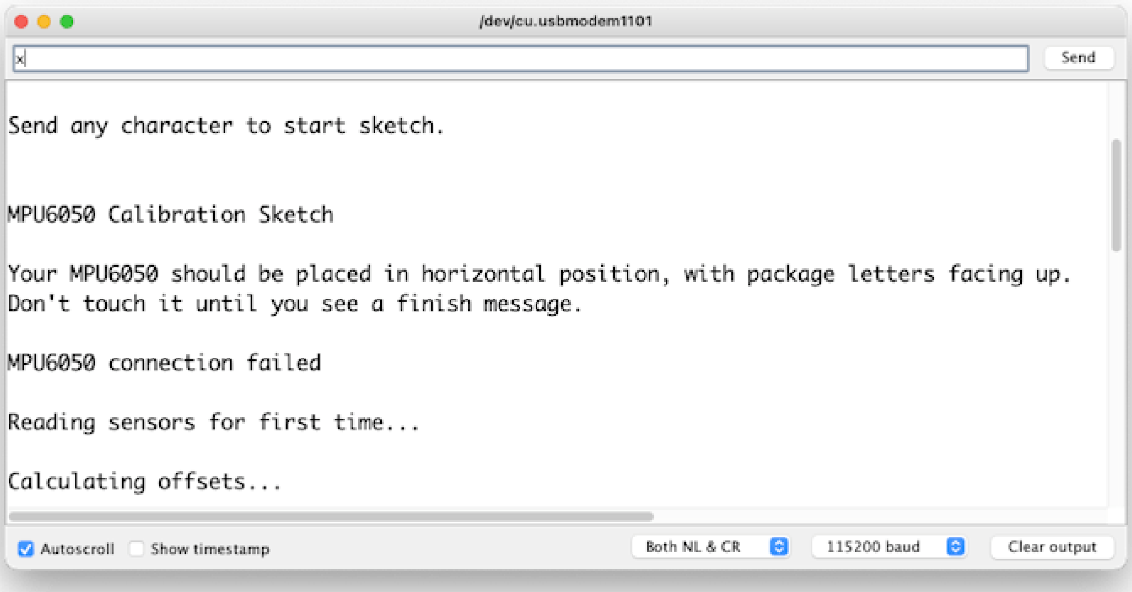
In the code, I left an option (`ACC_RANGE`) to be set to 0 (+/-2G) or 1 (+/- 4G). We will use +/-4G; that should be enough for us. In this case.

We will capture the accelerometer data on a frequency of 50Hz, and the acceleration data will be sent to the Serial Port as meters per squared second (m/s2).

When you ran the code with the IMU resting over your table, the accelerometer data shown on the Serial Monitor should be around: `0.00, 0.00, and 9.81`. If the values are a lot different, you should calibrate the IMU.

The MCU6050 can be calibrated using the sketch: mcu6050-calibration.ino.

Run the code. The following will be displayed on the Serial Monitor:

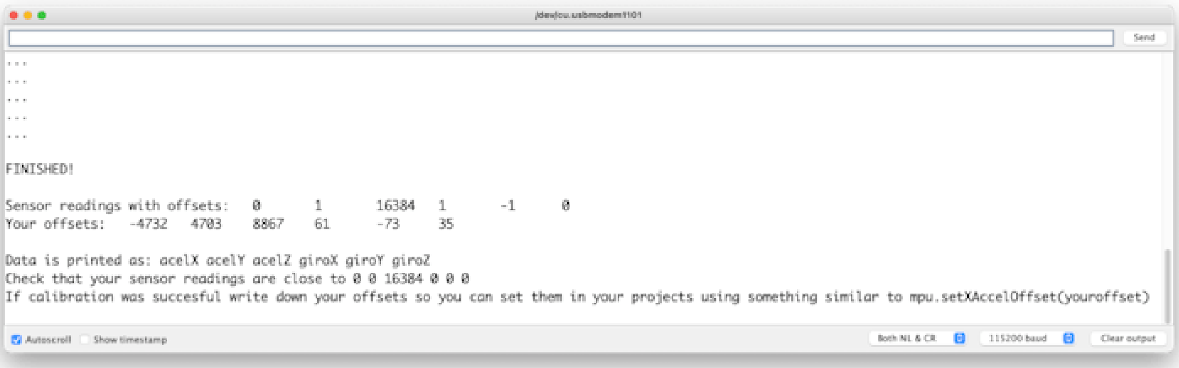


```
Send any character to start sketch.


MPU6050 Calibration Sketch

Your MPU6050 should be placed in horizontal position, with package letters facing up.
Don't touch it until you see a finish message.

MPU6050 connection failed

Reading sensors for first time...

Calculating offsets...
```

Send any character (in the above example, "x"), and the calibration should start.

Note that A message MPU6050 connection failed. Ignore this message. For some reason, `imu.testConnection()` is not returning a correct result.

In the end, you will receive the offset values to be used on all your sketches:



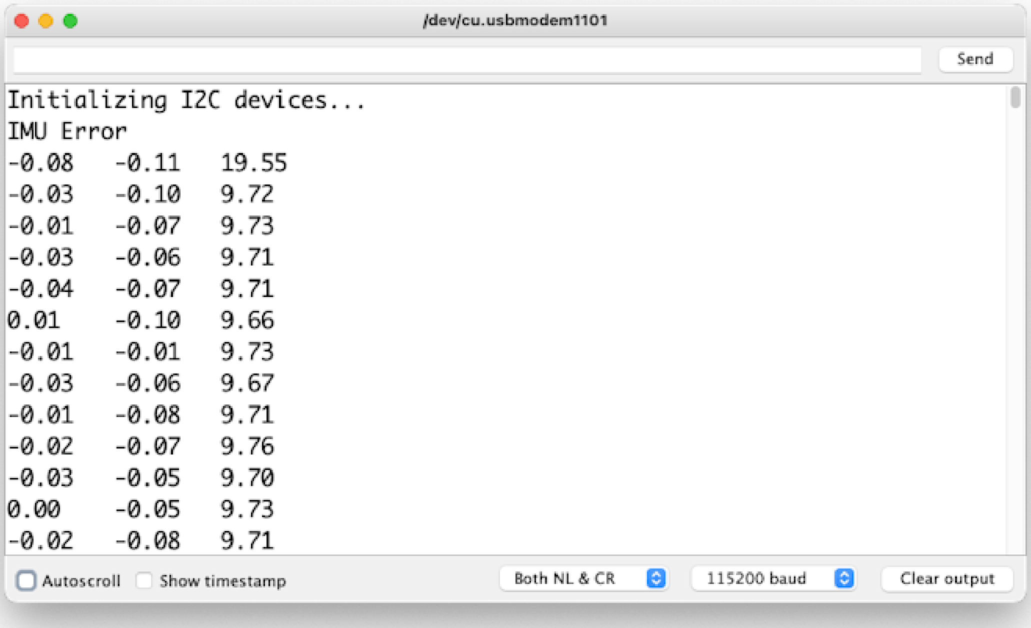
```
...
...
...
...
...

FINISHED!

Sensor readings with offsets:   0       1       16384   1       -1      0
Your offsets:   -4732   4703    8867    61      -73     35

Data is printed as: acelX acelY acelZ giroX giroY giroZ
Check that your sensor readings are close to 0 0 16384 0 0 0
If calibration was succesful write down your offsets so you can set them in your projects using something similar to mpu.setXAccelOffset(youroffset)
```

Take the values and use them on the setup:

```
//Set MCU 6050 OffSet Calibration
imu.setXAccelOffset(-4732);
imu.setYAccelOffset(4703);
imu.setZAccelOffset(8867);
imu.setXGyroOffset(61);
imu.setYGyroOffset(-73);
imu.setZGyroOffset(35);
```

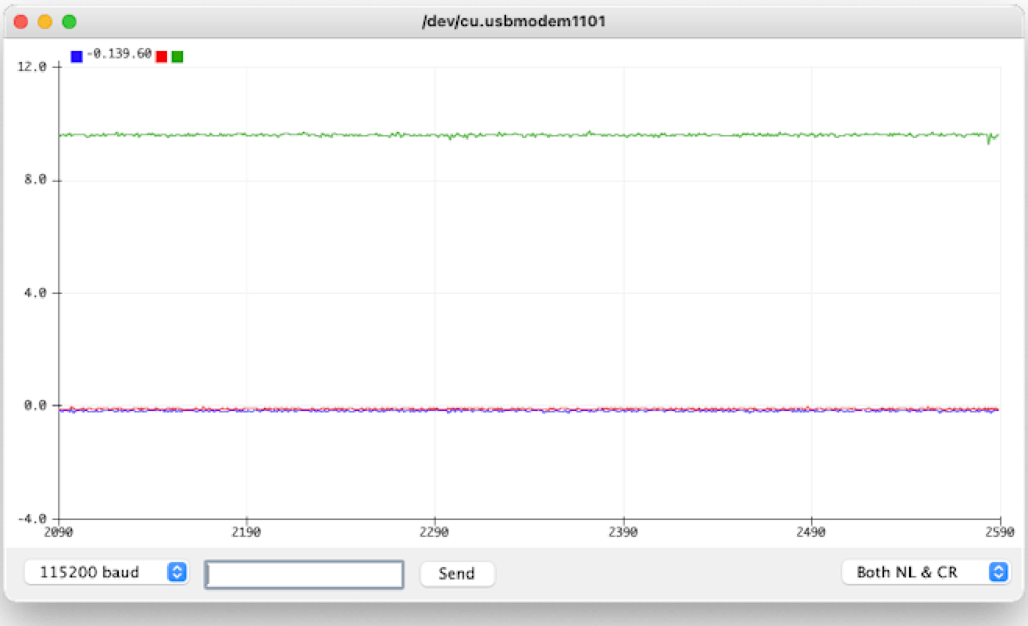
Now, run the sketch `MPU6050_Acc_Data_Acquisition.in:`

Once you run the above sketch, open the Serial Monitor:

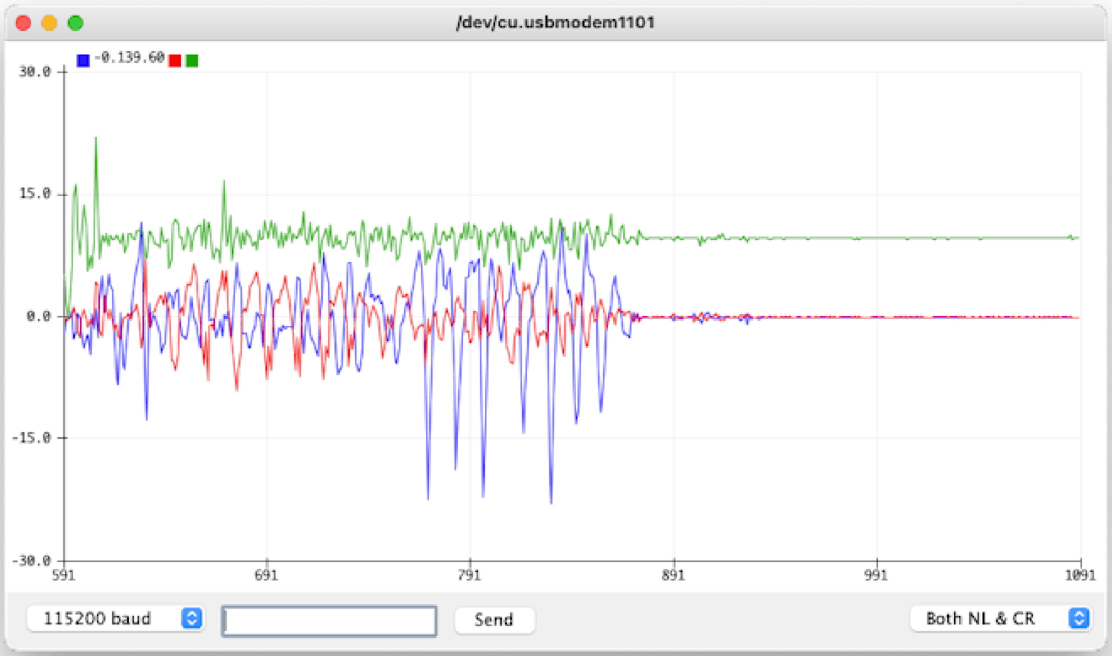


```
Initializing I2C devices...
IMU Error
-0.08   -0.11   19.55
-0.03   -0.10   9.72
-0.01   -0.07   9.73
-0.03   -0.06   9.71
-0.04   -0.07   9.71
0.01    -0.10   9.66
-0.01   -0.01   9.73
-0.03   -0.06   9.67
-0.01   -0.08   9.71
-0.02   -0.07   9.76
-0.03   -0.05   9.70
0.00    -0.05   9.73
-0.02   -0.08   9.71
```

Or check the Plotter:



Move your device in the three axes, and you should see the variation on Plotter:

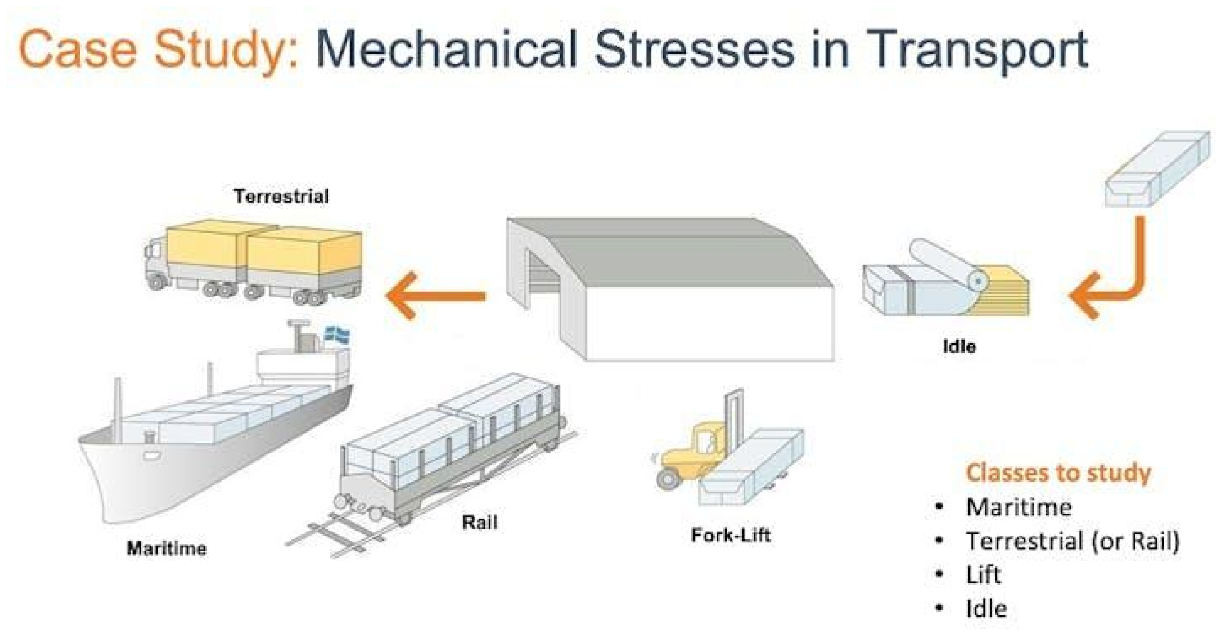


## The TinyML Motion Classification model

For our tutorial, we will simulate mechanical stresses in transport. Our problem will be to classify four classes of movement:

- **Maritime** (pallets in boats)
- **Terrestrial** (palettes in a Truck or Train)
- **Lift** (Palettes being handled by Fork-Lift)
- **Idle** (Palettes in Storage houses)

So, to start, we should collect data. Then, accelerometers will provide the data on the palette (or container).



From the above images, we can see that primarily horizontal movements should be associated with the "Terrestrial class, " Vertical movements with the "Lift Class, " no activity with the "Idle class, " and movent on all three axes to Maritime class.

## Connecting the device to Edge Impulse

For data collection, we should first connect our device to the Edge Impulse Studio, which will also be used for data pre-processing, model training, testing, and deployment.

Follow the instructions here to install the Node.js and Edge Impulse CLI on your computer.

Once the XIAO ESP32S3 is not a fully supported development board by Edge Impulse, we should, for example, use the CLI Data Forwarder to capture data from our sensor and send it to the Studio, as shown in this diagram:

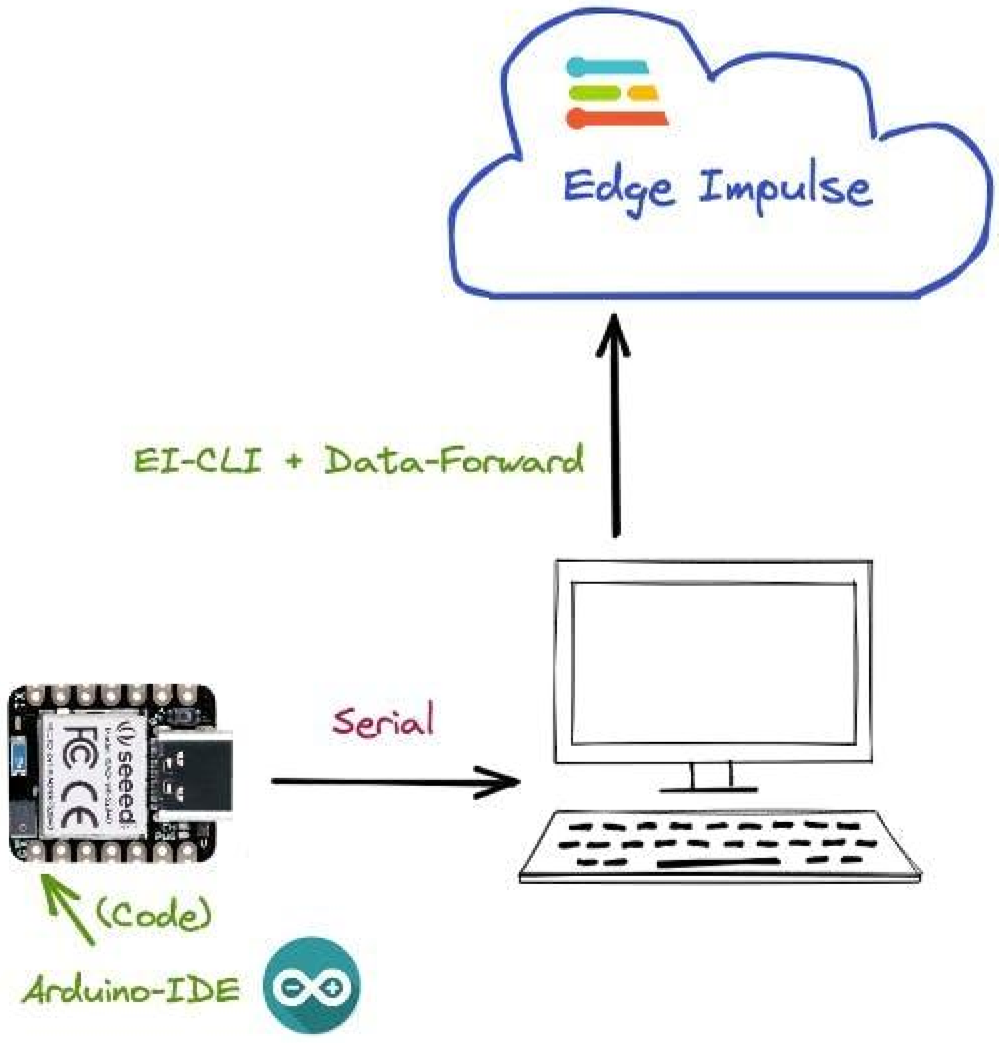
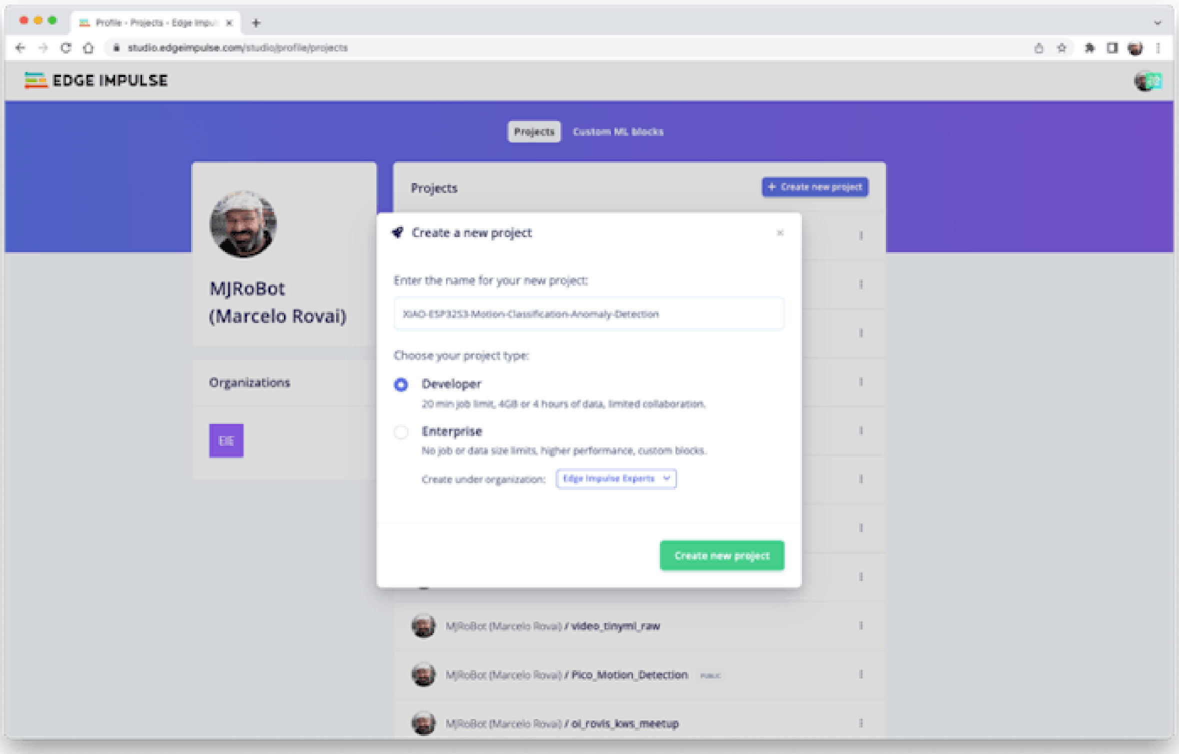


Image from author

You can also capture your data "offline", storing them on an SD card or send data via Bluetooth or Wifi for your computer. In this video, you can learn alternative ways to send data to the Edge Impulse Studio.

Connect your device to the serial port and run the previous code to capture IMU (Accelerometer) data, "printing them" on the serial. This will allow the Edge Impulse Studio to "capture" them.

Go to the Edge Impulse page and create a project.


Create a new project

Enter the name for your new project: XIAO-ESP32S3-Motion-Classification-Anomaly-Detection

Choose your project type:

[✓ Developer] 20 min job limit, 4GB or 4 hours of data, limited collaboration.

Enterprise No job or data size limits, higher performance, custom blocks.

Create under organization: Edge Impulse Experts

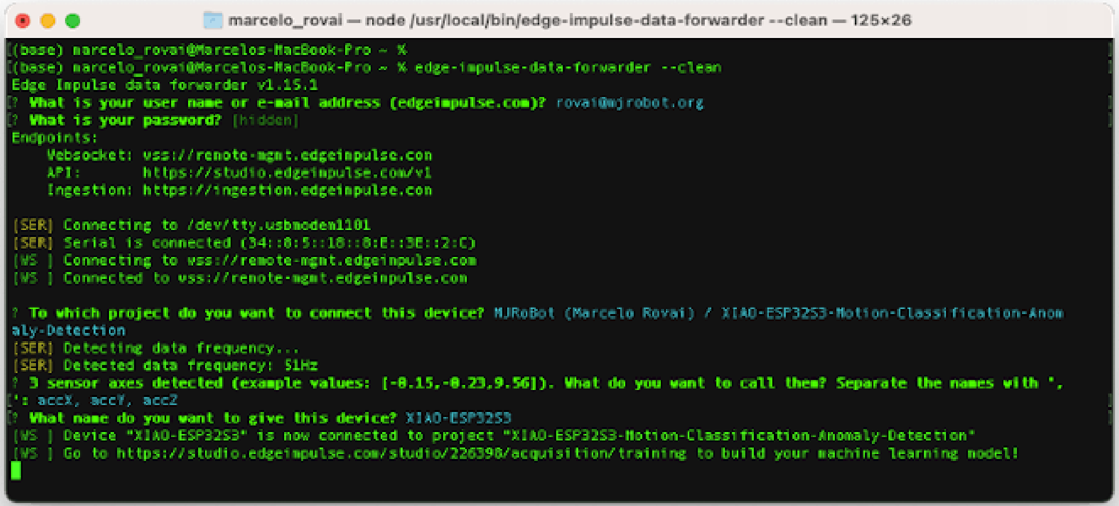
EDGE IMPULSE
Projects
Custom ML blocks
MJRoBot (Marcelo Rovai)
Organizations
Create new project
MJRoBot (Marcelo Rovai) / video_tinyml_raw
MJRoBot (Marcelo Rovai) / Pico_Motion_Detection
MJRoBot (Marcelo Rovai) / oi_rovis_kws_meetup

The maximum length for an Arduino library name is **63 characters**. Note that the Studio will name the final library using your project name and include "_inference" to it. In my case, the name that I choose at first will not work when I will try to deploy the Arduino library because it will result in 64 characters. So, I need to change it, taking out the "-anomaly-detection" part.

Next, start the CLI Data Forwarder on your terminal, entering (if it is the first time) the following command:

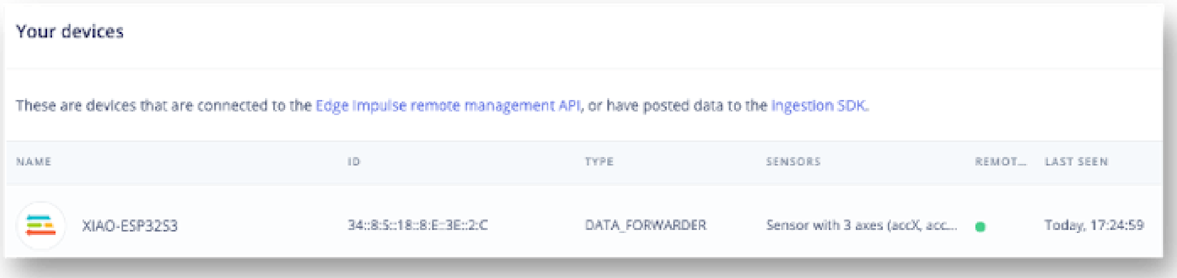
```
$ edge-impulse-data-forwarder --clean
```

Next, enter your EI credentials, and choose your project, variables, and device names:



```
marcelo_rovai — node /usr/local/bin/edge-impulse-data-forwarder --clean — 125×26
(base) marcelo_rovai@Marcelos-MacBook-Pro ~ %
(base) marcelo_rovai@Marcelos-MacBook-Pro ~ % edge-impulse-data-forwarder --clean
Edge Impulse data forwarder v1.15.1
? What is your user name or e-mail address (edgeimpulse.com)? rovai@mjrobot.org
? What is your password? [hidden]
Endpoints:
    Websocket: wss://remote-mgmt.edgeimpulse.com
    API:       https://studio.edgeimpulse.com/v1
    Ingestion: https://ingestion.edgeimpulse.com

[SER] Connecting to /dev/tty.usbmodem1101
[SER] Serial is connected (34::8:5::18::8:E::3E::2:C)
[WS ] Connecting to wss://remote-mgmt.edgeimpulse.com
[WS ] Connected to wss://remote-mgmt.edgeimpulse.com

? To which project do you want to connect this device? MJRoBot (Marcelo Rovai) / XIAO-ESP32S3-Motion-Classification-Anom
aly-Detection
[SER] Detecting data frequency...
[SER] Detected data frequency: 51Hz
? 3 sensor axes detected (example values: [-0.15,-0.23,9.56]). What do you want to call them? Separate the names with ',
': accX, accY, accZ
? What name do you want to give this device? XIAO-ESP32S3
[WS ] Device "XIAO-ESP32S3" is now connected to project "XIAO-ESP32S3-Motion-Classification-Anomaly-Detection"
[WS ] Go to https://studio.edgeimpulse.com/studio/226398/acquisition/training to build your machine learning model!
```

Go to your EI Project and verify if the device is connected (the dot should be green):



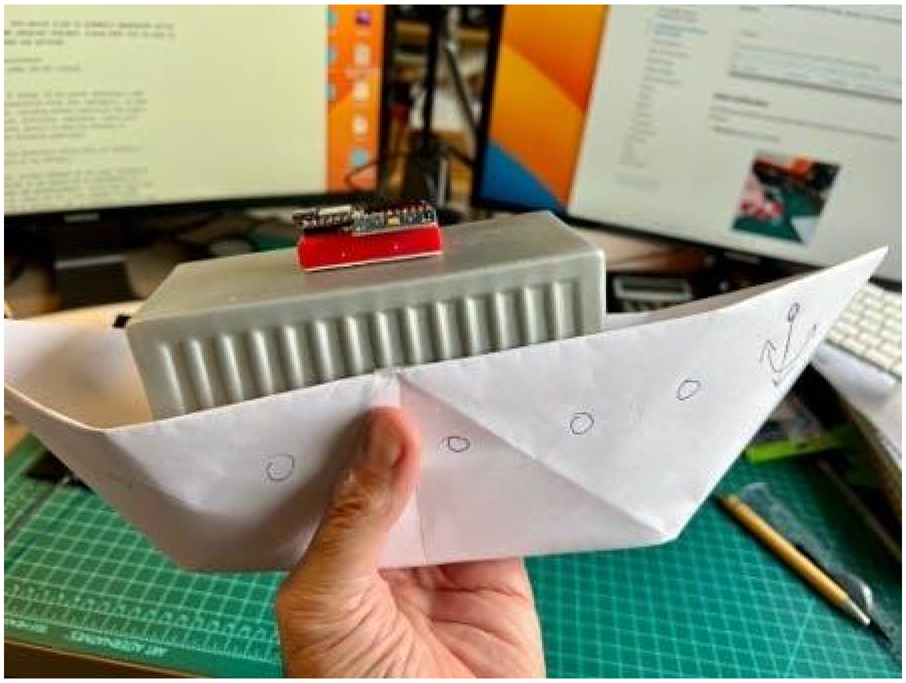
Your devices

These are devices that are connected to the Edge Impulse remote management API, or have posted data to the Ingestion SDK.

| NAME | ID | TYPE | SENSORS | REMOT... | LAST SEEN |
|---|---|---|---|---|---|
| XIAO-ESP32S3 | 34::8:5::18::8:E::3E::2:C | DATA_FORWARDER | Sensor with 3 axes (accX, acc... | ● | Today, 17:24:59 |

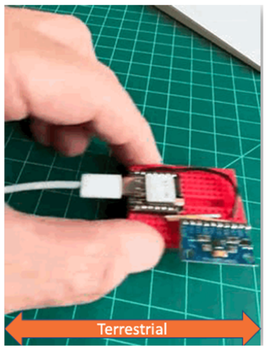
## Data Collection

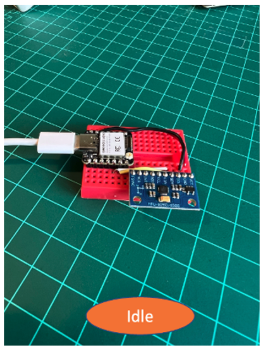
As discussed before, we should capture data from all four Transportation Classes. Imagine that you have a container with a built-in accelerometer:

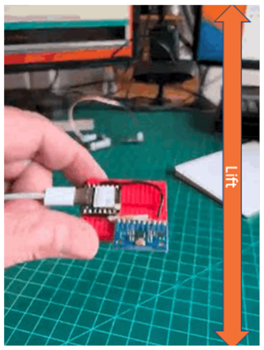


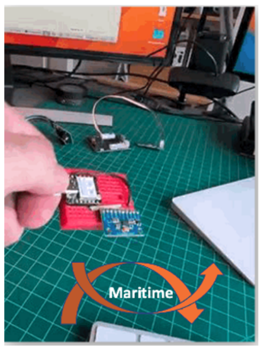
Now imagine your container is on a boat, facing an angry ocean, on a truck, etc.:

- **Maritime** (pallets in boats)
- **Terrestrial** (palettes in a Truck or Train)
- **Lift** (Palettes being handled by
- **Idle** (Palettes in Storage houses)

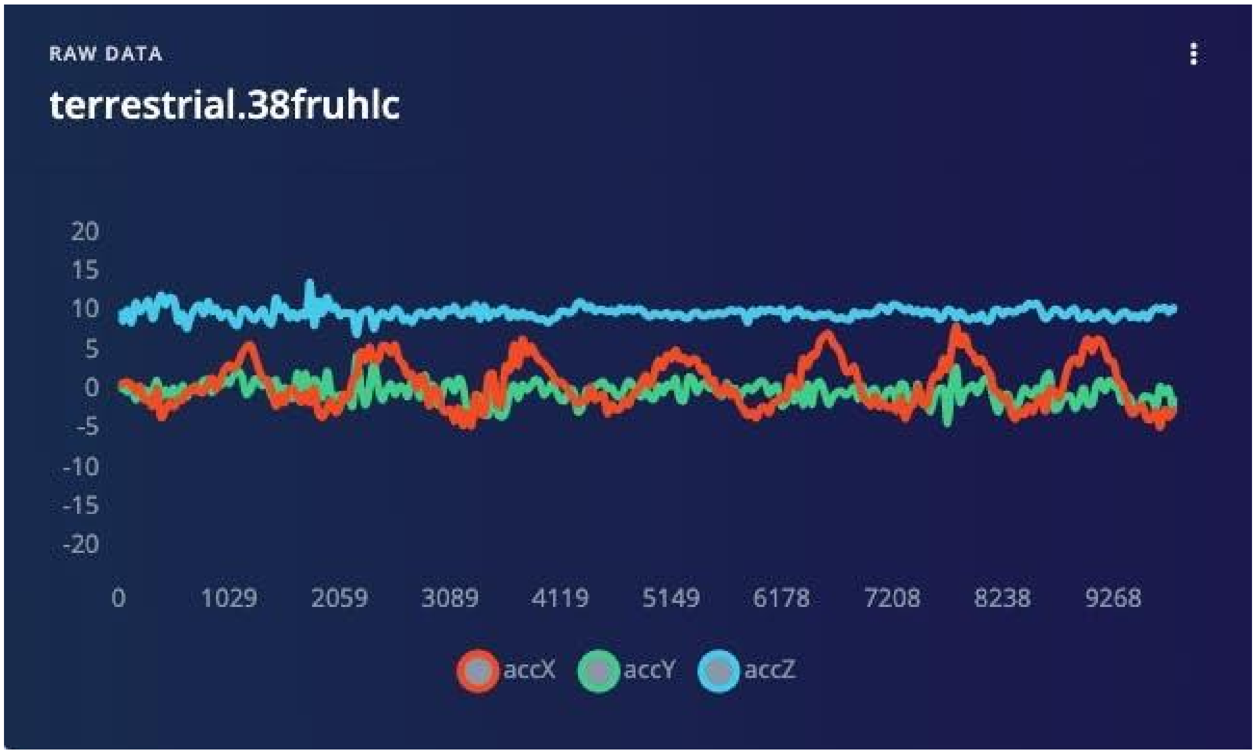




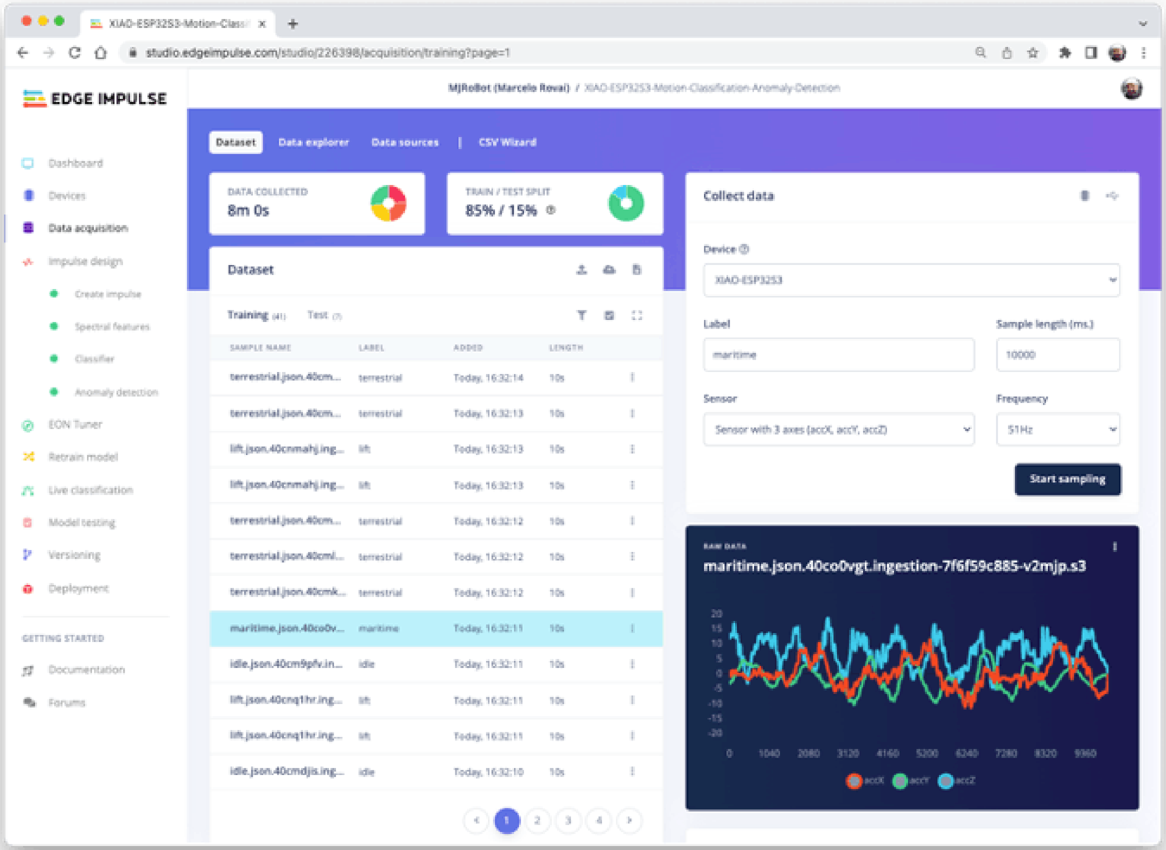




Below is one sample (raw data) of 10 seconds:



You can capture, for example, 2 minutes (twelve samples of 10 seconds each) for the four classes. Using the "3 dots" after each one of the samples, select 2, moving them for the Test set (or use the automatic Train/Test Split tool on the Danger Zone of Dashboard tab). Below are the result datasets:



DATA COLLECTED 8m 0s

TRAIN / TEST SPLIT 85% / 15%

Dataset

Training (41) Test (7)

| SAMPLE NAME | LABEL | ADDED | LENGTH |
|---|---|---|---|
| terrestrial.json.40cm... | terrestrial | Today, 16:32:14 | 10s |
| terrestrial.json.40cm... | terrestrial | Today, 16:32:13 | 10s |
| lift.json.40cnmahj.ing... | lift | Today, 16:32:13 | 10s |
| lift.json.40cnmahj.ing... | lift | Today, 16:32:13 | 10s |
| terrestrial.json.40cm... | terrestrial | Today, 16:32:12 | 10s |
| terrestrial.json.40cml... | terrestrial | Today, 16:32:12 | 10s |
| terrestrial.json.40cmk... | terrestrial | Today, 16:32:12 | 10s |
| maritime.json.40co0v... | maritime | Today, 16:32:11 | 10s |
| idle.json.40cm9pfv.in... | idle | Today, 16:32:11 | 10s |
| lift.json.40cnq1hr.ing... | lift | Today, 16:32:11 | 10s |
| lift.json.40cnq1hr.ing... | lift | Today, 16:32:11 | 10s |
| idle.json.40cmdjis.ing... | idle | Today, 16:32:10 | 10s |

Collect data

Device: XIAO-ESP32S3

Label: maritime

Sample length (ms.): 10000

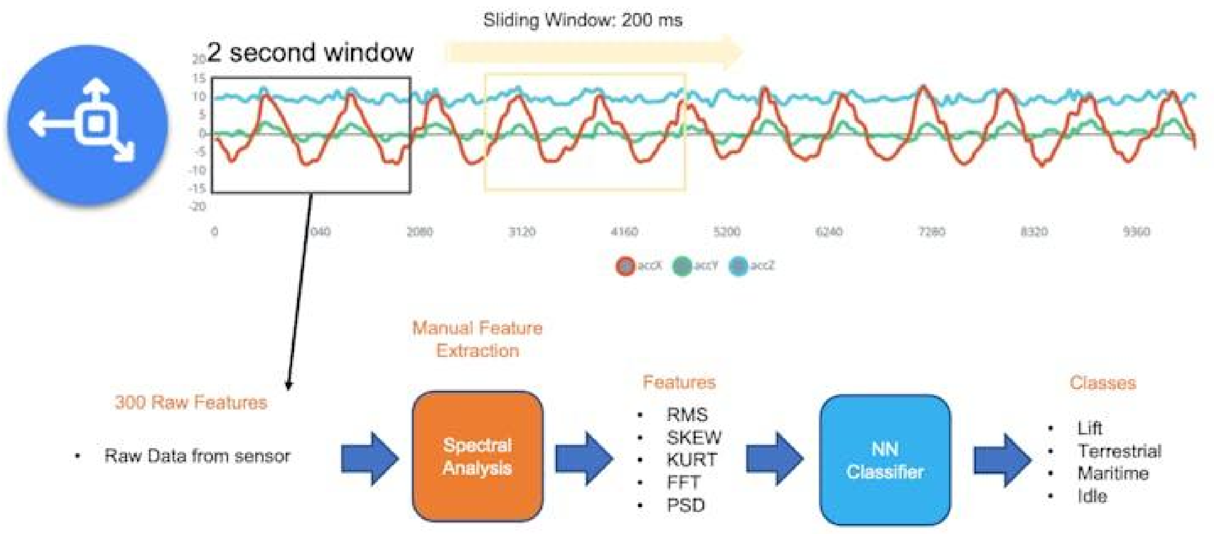
Sensor: Sensor with 3 axes (accX, accY, accZ)

Frequency: 51Hz

maritime.json.40co0vgt.ingestion-7f6f59c885-v2mjp.s3

## Data Pre-Processing

The raw data type captured by the accelerometer is a "time series" and should be converted to "tabular data". We can do this conversion using a sliding window over the sample data. For example, in the below figure,



we can see 10 seconds of accelerometer data captured with a sample rate (SR) of 50Hz. A 2 seconds window will capture 300 data points (3 axis x 2 seconds x 50 samples). We will slide this window each 200ms, creating a larger dataset where each instance has 300 raw features.

You should use the best SR for your case, taking into consideration, Nyquist's theorem, which states that a periodic signal must be sampled at more than twice the highest frequency component of the signal.

Data preprocessing is a challenging area for embedded machine learning. Still, Edge Impulse helps overcome this with its digital signal processing (DSP) preprocessing step and, more specifically, the Spectral Features.

On the Studio, this dataset will be the input of a Spectral Analysis block, which is excellent for analyzing repetitive motion, such as data from accelerometers. This block will perform a DSP (Digital Signal Processing), extracting features such as "FFT" or "Wavelets". In the most common case, FFT.

The **Time Domain Statistical features** per axis/channel are:

- RMS
- Skewness
- Kurtosis,

And the **Frequency Domain Spectral features** per axis/channel are:

- Spectral Power
- Skewness
- Kurtosis

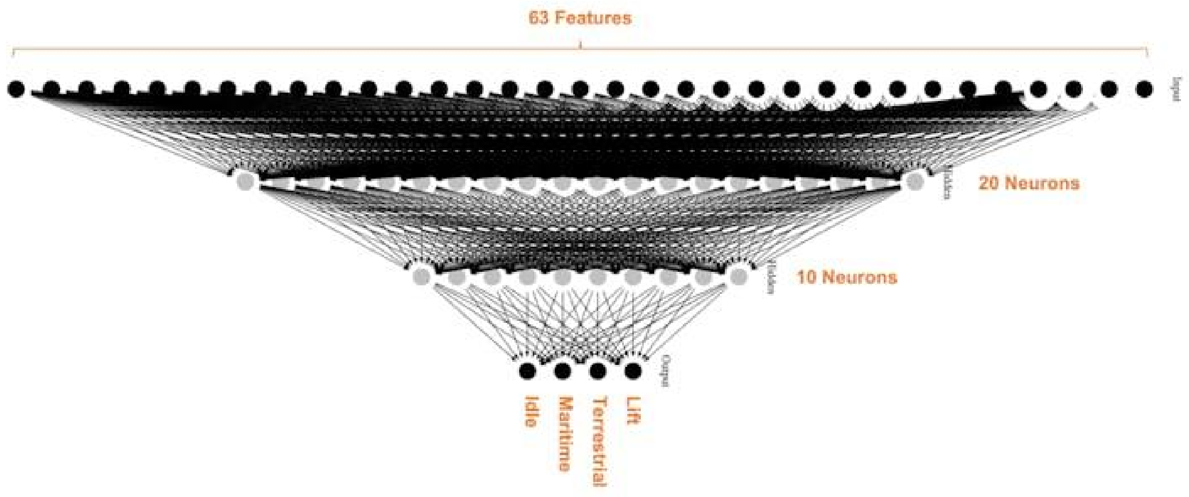
So, for example, for an FFT length of 32 points, the resulting output of the Spectral Analysis Block will be 21 features per axis (a total of 63 features).

Those 63 features will be the Input Tensor of a Neural Network Classifier and the Anomaly Detection model (K-Means).

You can learn more by digging into the tutorial TinyML under the hood: Spectral Analysis.

## Model Design

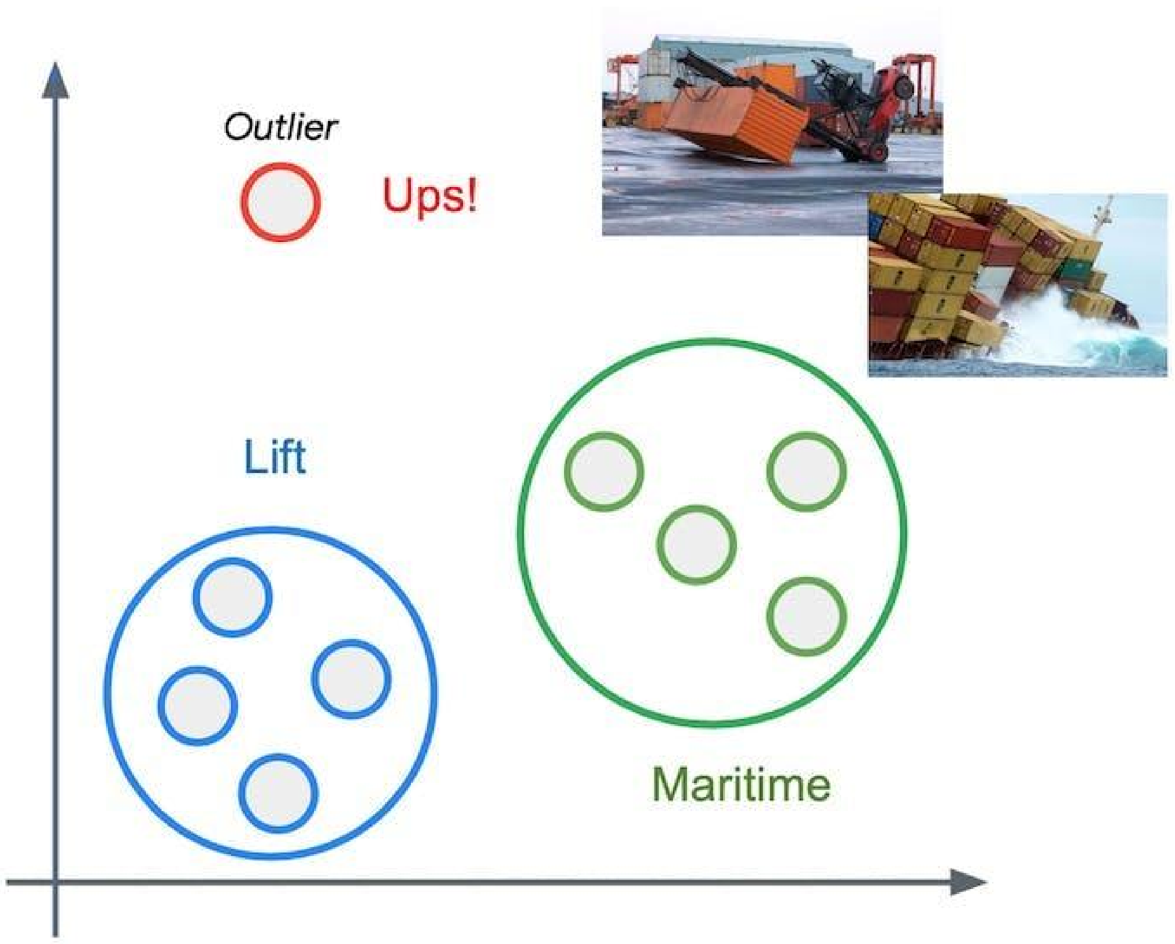
Our classifier will be a Dense Neural Network (DNN) that will have 63 neurons on its input layer, two hidden layers with 20 and 10 neurons, and an output layer with four neurons (one per each class), as shown here:



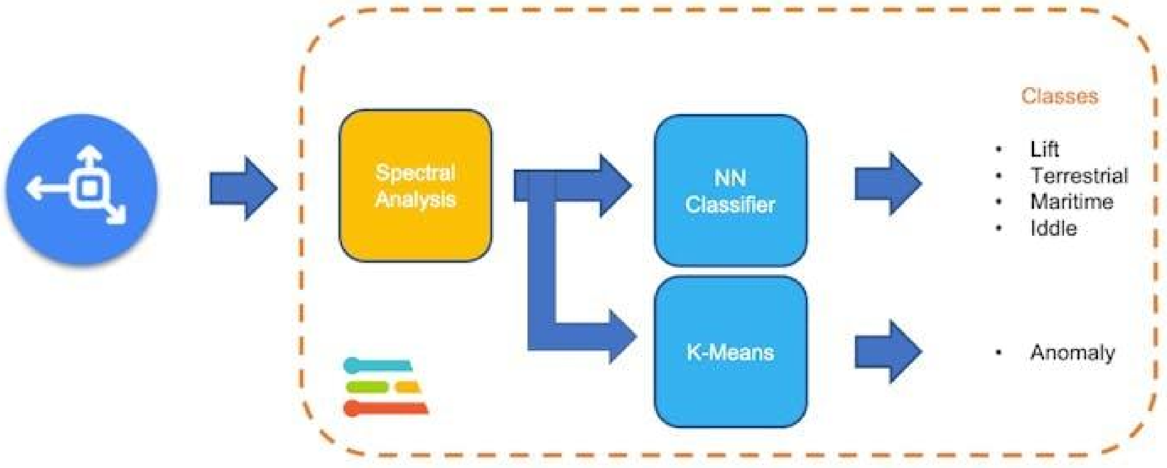
## Impulse Design

An impulse takes raw data uses signal processing to extract features and then uses a learning block to classify new data.

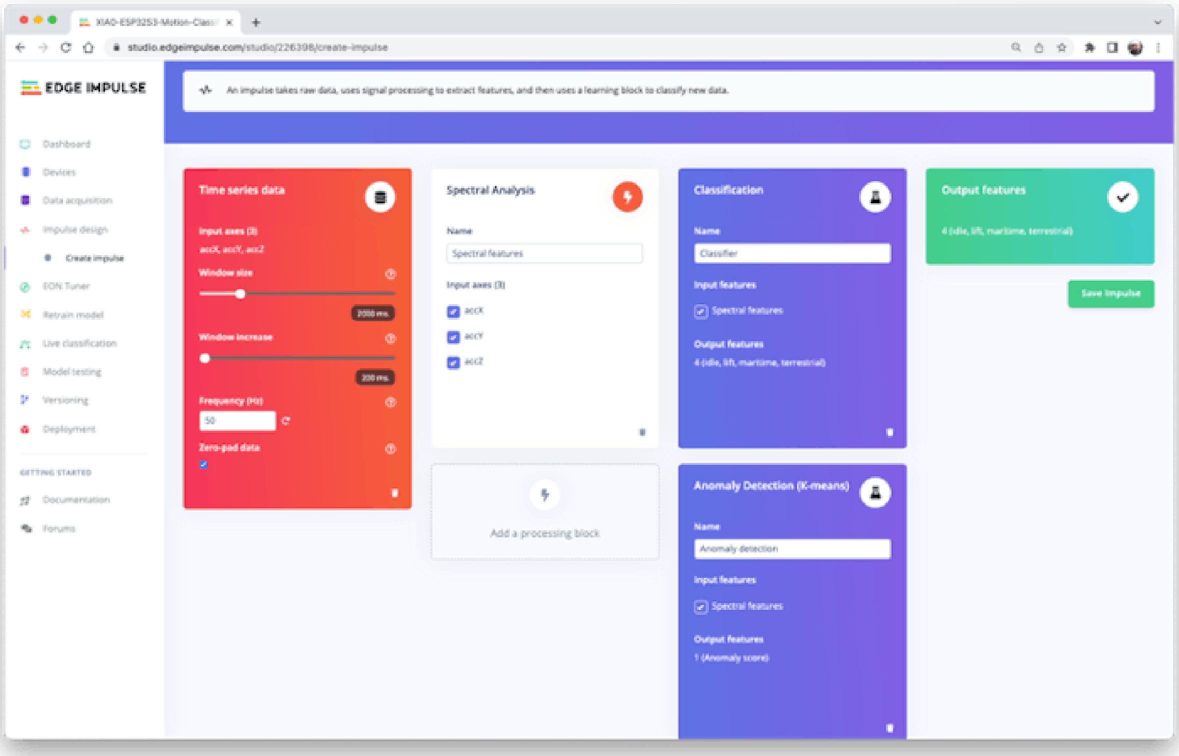
We also take advantage of a second model, the K-means, that can be used for Anomaly Detection. If we imagine that we could have our known classes as clusters, any sample that could not fit on that could be an outlier, an anomaly (for example, a container rolling out of a ship on the ocean).



For that, we can use the same input tensor that goes to the NN Classifier as the input of a K-means model:



Below is our final Impulse design:



An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.

**Time series data**
Input axes (3): accX, accY, accZ
Window size: 2000 ms.
Window increase: 200 ms.
Frequency (Hz): 50
Zero-pad data: ✓

**Spectral Analysis**
Name: Spectral features
Input axes (3): ✓ accX, ✓ accY, ✓ accZ

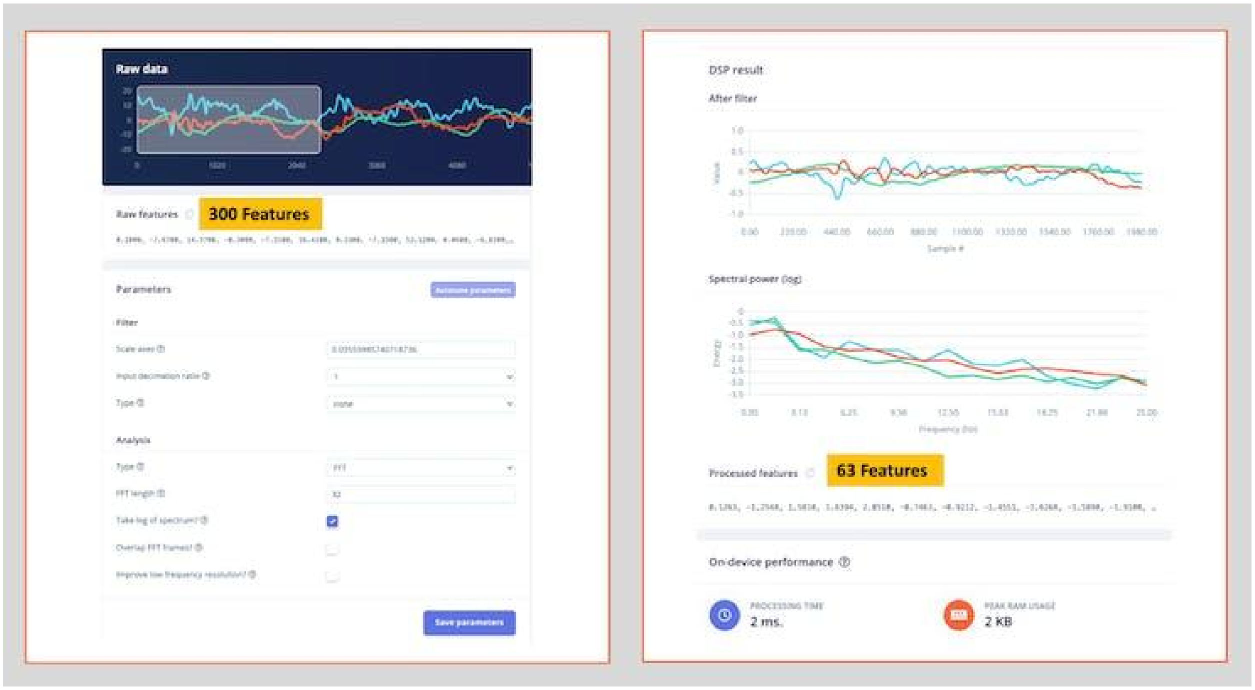
**Classification**
Name: Classifier
Input features: ✓ Spectral features
Output features: 4 (idle, lift, maritime, terrestrial)

**Anomaly Detection (K-means)**
Name: Anomaly detection
Input features: ✓ Spectral features
Output features: 1 (Anomaly score)

**Output features**
4 (idle, lift, maritime, terrestrial)

## Generating features

At this point in our project, we have defined the pre-processing method and the model designed. Now it is time to have the job done. First, let's take the raw data (time-series type) and convert it to tabular data. Go to the Spectral Features tab, and select Save Parameters:



Filter

Scale axes: [illegible]

Input decimation ratio: 1

Type: none

Analysis

Type: FFT

FFT length: 32

Take log of spectrum? [✓]

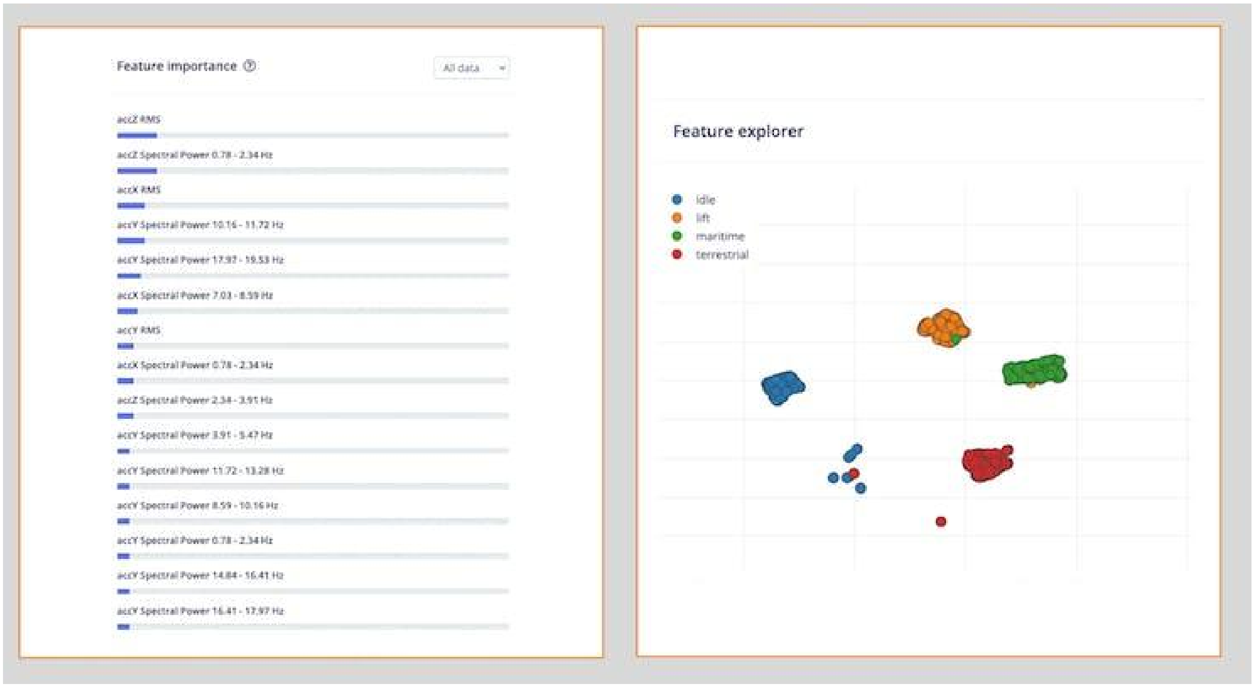
Overlap FFT frames? [ ]

Improve low frequency resolution? [ ]

At the top menu, select Generate Features option and Generate Features button. Each of our 2 seconds window data will be converted into one data point of 63 features each.

The Feature Explorer will show those data in 2D using UMAP. Uniform Manifold Approximation and Projection (UMAP) is a dimension reduction technique that can be used for visualization similarly to t-SNE but also for general non-linear dimension reduction.

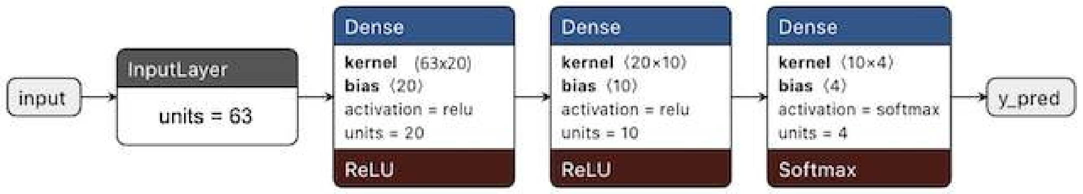
The visualization makes it possible to verify that the classes present an excellent separation, which indicates that the classifier should work well.

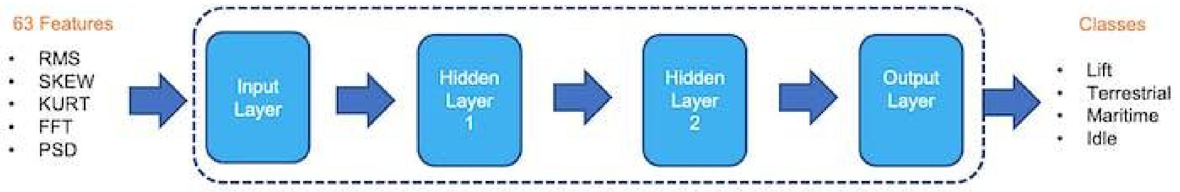

Feature importance
All data
accZ RMS
accZ Spectral Power 0.78 - 2.34 Hz
accX RMS
accY Spectral Power 10.16 - 11.72 Hz
accY Spectral Power 17.97 - 19.53 Hz
accX Spectral Power 7.03 - 8.59 Hz
accY RMS
accX Spectral Power 0.78 - 2.34 Hz
accZ Spectral Power 2.34 - 3.91 Hz
accY Spectral Power 3.91 - 5.47 Hz
accY Spectral Power 11.72 - 13.28 Hz
accY Spectral Power 8.59 - 10.16 Hz
accY Spectral Power 0.78 - 2.34 Hz
accY Spectral Power 14.84 - 16.41 Hz
accY Spectral Power 16.41 - 17.97 Hz
Feature explorer
idle
lift
maritime
terrestrial

Optionally you can analyze how important each one of the features is for one class compared with other classes.

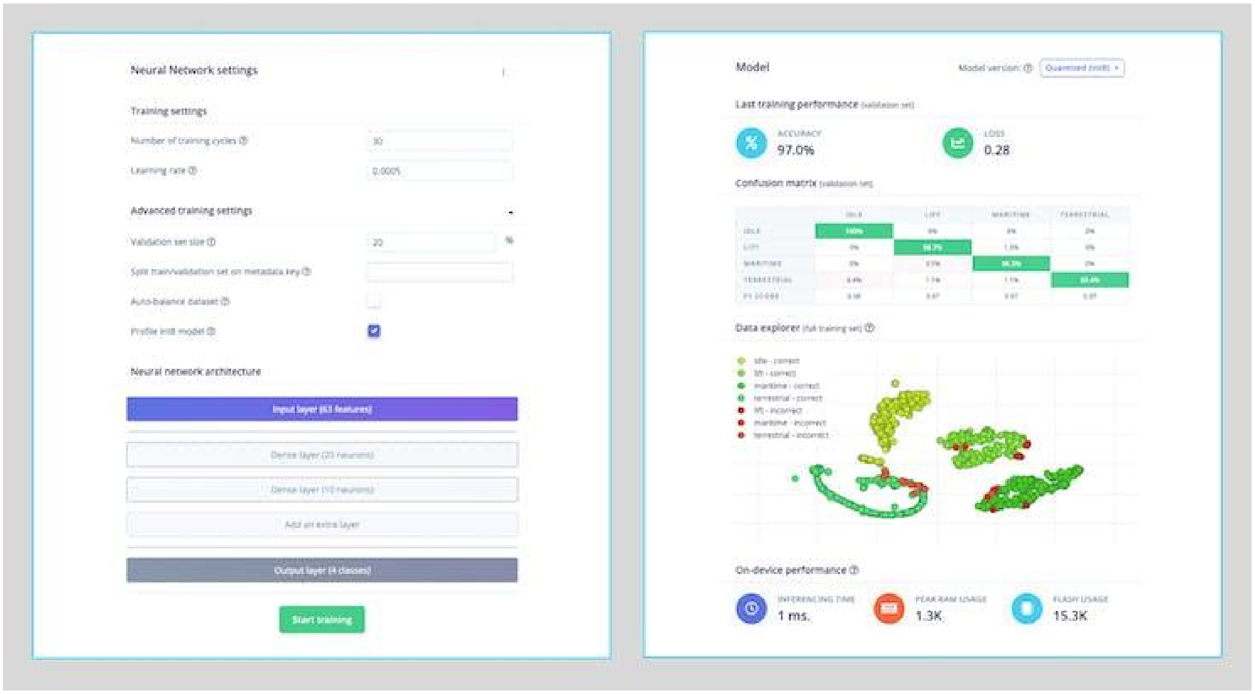
## Training

Our model has four layers, as shown below:





As hyperparameters, we will use a Learning Rate of 0.005 and 20% of data for validation for 30 epochs. After training, we can see that the accuracy is 97%.



Neural Network settings

Training settings

Number of training cycles: 30

Learning rate: 0.0005

Advanced training settings

Validation set size: 20 %

Split train/validation set on metadata key:

Auto-balance dataset: ☐

Profile int8 model: ✓

Neural network architecture

Model

Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 97.0%

LOSS 0.28

Confusion matrix (validation set)

| | IDLE | LIFT | MARITIME | TERRESTRIAL |
|---|---|---|---|---|
| IDLE | 100% | 0% | 0% | 0% |
| LIFT | 0% | 98.7% | 1.3% | 0% |
| MARITIME | 0% | 3.7% | 96.3% | 0% |
| TERRESTRIAL | 6.4% | 1.1% | 1.1% | 91.4% |
| F1 SCORE | 0.98 | 0.97 | 0.97 | 0.97 |

Data explorer (full training set)

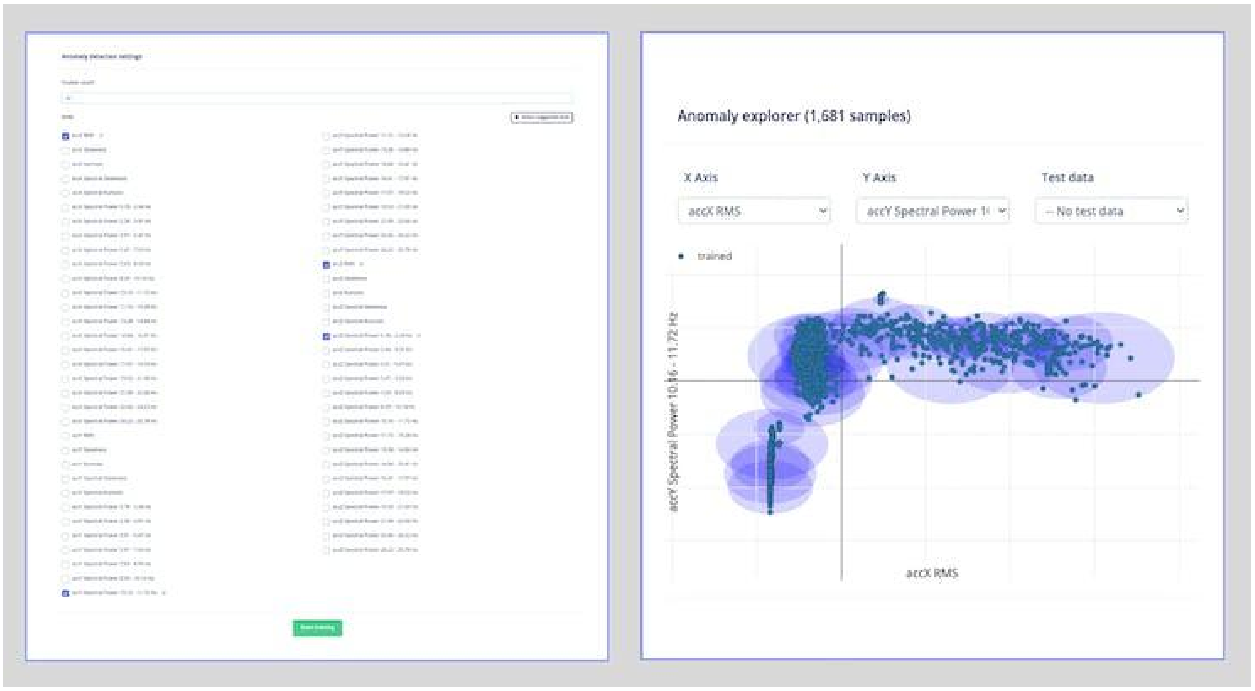
On-device performance

INFERENCING TIME 1 ms.

PEAK RAM USAGE 1.3K

FLASH USAGE 15.3K

And for Anomaly Detection, we should choose the suggested features that are precisely the most important ones found in the Feature Extraction. The number of clusters will be 32 as suggested by the Studio:


Anomaly explorer (1,681 samples)

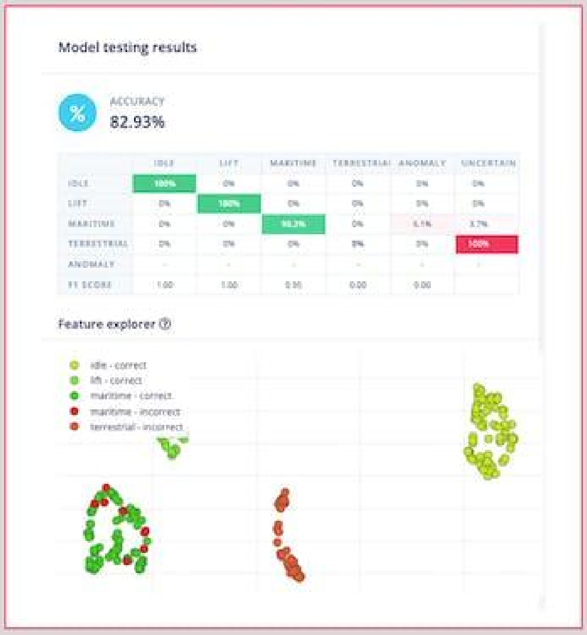
X Axis: accX RMS

Y Axis: accY Spectral Power 1(

Test data: -- No test data

## Testing

Using 20% of the data left behind during the data capture phase, we can verify how our model will behave with unknown data; if not 100% (what is expected), the result was not that good (8%), mainly due to the terrestrial class. Once we have four classes (which output should add 1.0), we can set up a lower threshold for a class to be considered valid (for example, 0.4):

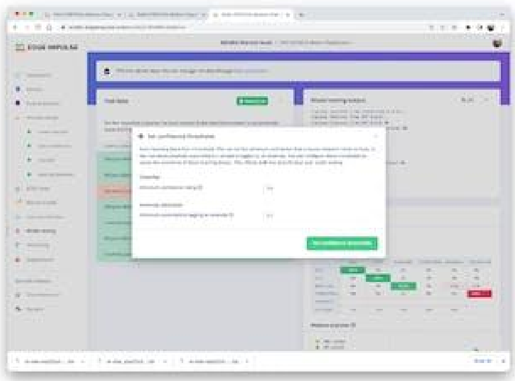


Model testing results

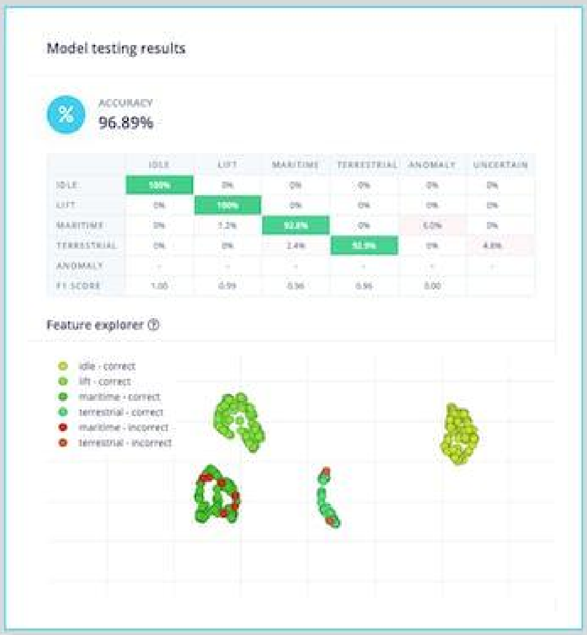
ACCURACY
82.93%

| | IDLE | LIFT | MARITIME | TERRESTRIAL | ANOMALY | UNCERTAIN |
|---|---|---|---|---|---|---|
| IDLE | 100% | 0% | 0% | 0% | 0% | 0% |
| LIFT | 0% | 100% | 0% | 0% | 0% | 0% |
| MARITIME | 0% | 0% | 93.2% | 0% | 3.1% | 3.7% |
| TERRESTRIAL | 0% | 0% | 0% | 0% | 0% | 100% |
| ANOMALY | - | - | - | - | - | - |
| F1 SCORE | 1.00 | 1.00 | 0.95 | 0.00 | 0.00 | |

Feature explorer ⓘ



Now, the Test accuracy will go up to 97%.

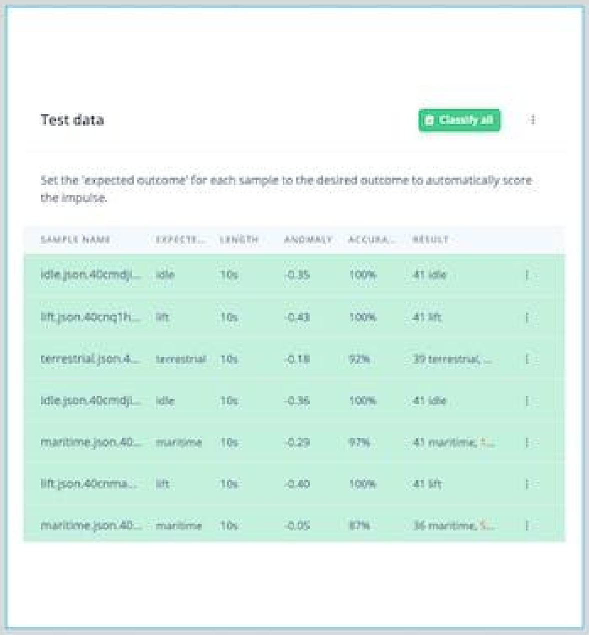


Model testing results

ACCURACY
96.89%

| | IDLE | LIFT | MARITIME | TERRESTRIAL | ANOMALY | UNCERTAIN |
|---|---|---|---|---|---|---|
| IDLE | 100% | 0% | 0% | 0% | 0% | 0% |
| LIFT | 0% | 100% | 0% | 0% | 0% | 0% |
| MARITIME | 0% | 1.2% | 92.8% | 0% | 6.0% | 0% |
| TERRESTRIAL | 0% | 0% | 2.4% | 92.9% | 0% | 4.8% |
| ANOMALY | - | - | - | - | - | - |
| F1 SCORE | 1.00 | 0.99 | 0.96 | 0.96 | 0.00 | |

Feature explorer ⓘ



Test data

Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse.

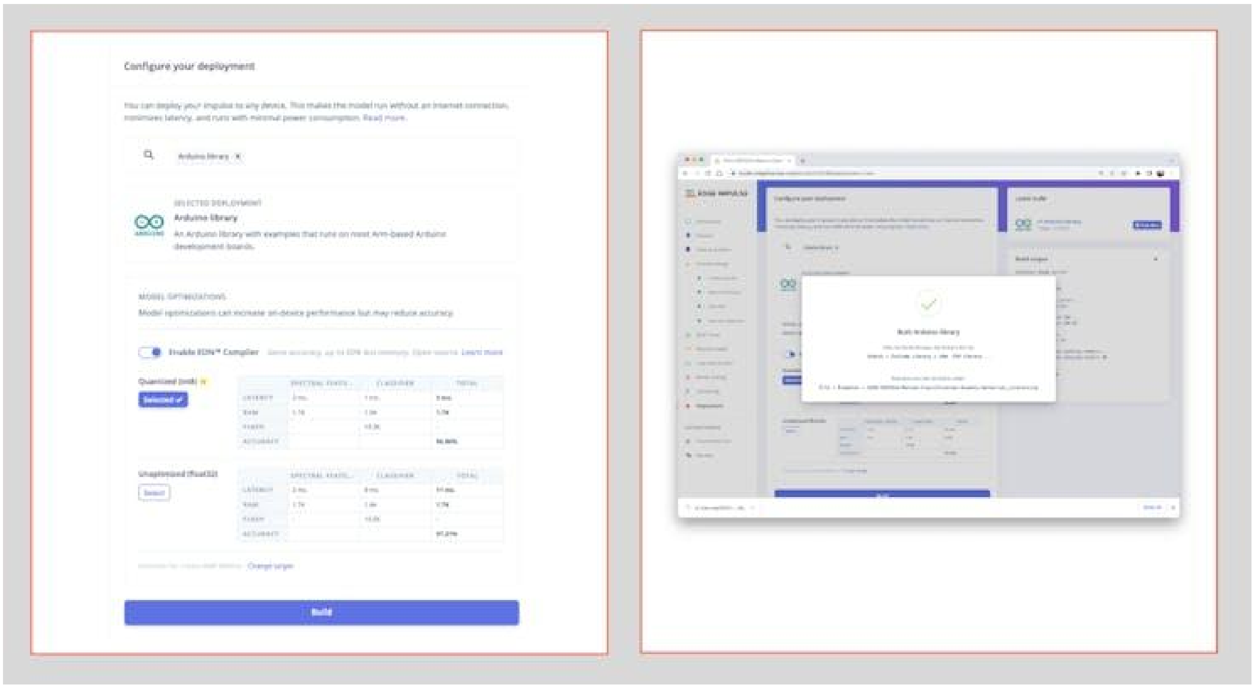
| SAMPLE NAME | EXPECTE... | LENGTH | ANOMALY | ACCURA... | RESULT |
|---|---|---|---|---|---|
| idle.json.40cmdji... | idle | 10s | -0.35 | 100% | 41 idle |
| lift.json.40cnq1h... | lift | 10s | -0.43 | 100% | 41 lift |
| terrestrial.json.4... | terrestrial | 10s | -0.18 | 92% | 39 terrestrial, ... |
| idle.json.40cmdji... | idle | 10s | -0.36 | 100% | 41 idle |
| maritime.json.40... | maritime | 10s | -0.29 | 97% | 41 maritime, 1... |
| lift.json.40cnma... | lift | 10s | -0.40 | 100% | 41 lift |
| maritime.json.40... | maritime | 10s | -0.05 | 87% | 36 maritime, 5... |

You should also use your device (which is still connected to the Studio) and perform some Live Classification.

Be aware that here you will capture real data with your device and upload it to the Studio, where an inference will be taken using the trained model (But the model is NOT in your device).

## Deploy

Now it is time for magic˜! The Studio will package all the needed libraries, preprocessing functions, and trained models, downloading them to your computer. You should select the option Arduino Library and at the bottom, select Quantized (Int8) and Build. A Zip file will be created and downloaded to your computer.



Configure your deployment

You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. Read more.

Arduino library ×

SELECTED DEPLOYMENT
Arduino library
An Arduino library with examples that runs on most Arm-based Arduino development boards.

MODEL OPTIMIZATIONS
Model optimizations can increase on-device performance but may reduce accuracy.

Enable EON™ Compiler [illegible] Learn more

Quantized (int8) — Selected ✓

| | SPECTRAL FEATS... | CLASSIFIER | TOTAL |
|---|---|---|---|
| LATENCY | 2 ms. | 1 ms. | 3 ms. |
| RAM | 1.7K | 1.5K | 1.7K |
| FLASH | | 15.3K | - |
| ACCURACY | | | [illegible] |

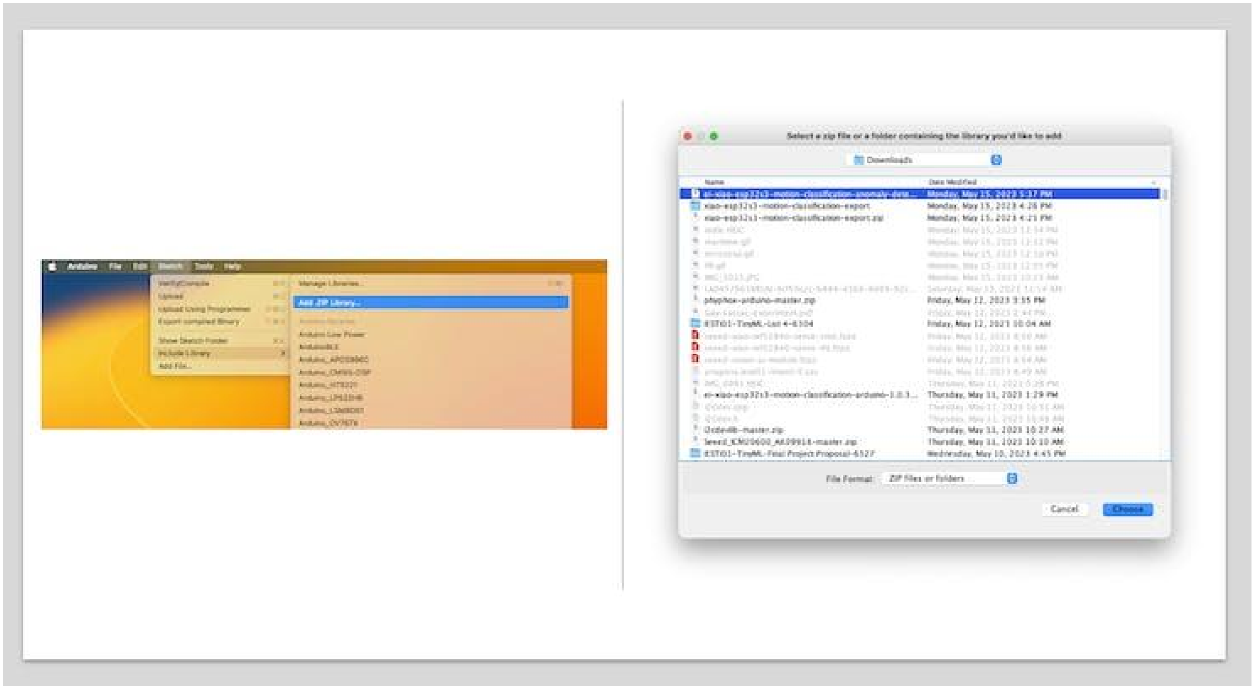
Unoptimized (float32) — Select

| | SPECTRAL FEATS... | CLASSIFIER | TOTAL |
|---|---|---|---|
| LATENCY | 2 ms. | 9 ms. | 11 ms. |
| RAM | 1.7K | 1.4K | 1.7K |
| FLASH | - | 15.0K | - |
| ACCURACY | | | [illegible] |

[illegible] Change target

Build

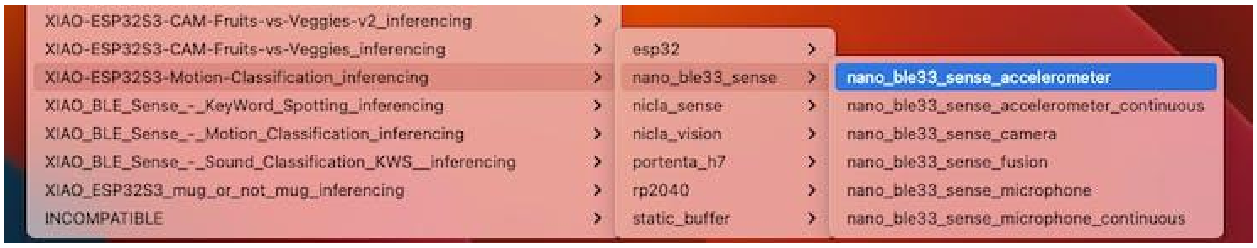
On your Arduino IDE, go to the Sketch tab and select the option Add.ZIP Library and Choose the.zip file downloaded by the Studio:



## Inference

Now it is time for a real test. We will make inferences wholly disconnected from the Studio. Let's change one of the code examples created when you deploy the Arduino Library.

In your Arduino IDE, go to File/Examples tab and look for your project, and on examples, select nano_ble_sense_accelerometer:


XIAO-ESP32S3-CAM-Fruits-vs-Veggies-v2_inferencing
XIAO-ESP32S3-CAM-Fruits-vs-Veggies_inferencing
XIAO-ESP32S3-Motion-Classification_inferencing
XIAO_BLE_Sense_-_KeyWord_Spotting_inferencing
XIAO_BLE_Sense_-_Motion_Classification_inferencing
XIAO_BLE_Sense_-_Sound_Classification_KWS__inferencing
XIAO_ESP32S3_mug_or_not_mug_inferencing
INCOMPATIBLE
esp32
nano_ble33_sense
nicla_sense
nicla_vision
portenta_h7
rp2040
static_buffer
nano_ble33_sense_accelerometer
nano_ble33_sense_accelerometer_continuous
nano_ble33_sense_camera
nano_ble33_sense_fusion
nano_ble33_sense_microphone
nano_ble33_sense_microphone_continuous

Of course, this is not your board, but we can have the code working with only a few changes.

For example, at the beginning of the code, you have the library related to Arduino Sense IMU:

```
/* Includes ---------------------------------------------------------------- */
#include <XIAO-ESP32S3-Motion-Classification_inferencing.h>
#include <Arduino_LSM9DS1.h>
```

Change the "includes" portion with the code related to the IMU:

```
#include <XIAO-ESP32S3-Motion-Classification_inferencing.h>
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
```

Change the Constant Defines

```
/* Constant defines -------------------------------------------------------- */
MPU6050 imu;
int16_t ax, ay, az;

#define ACC_RANGE           1 // 0: -/+2G; 1: +/-4G
#define CONVERT_G_TO_MS2    (9.81/(16384/(1.+ACC_RANGE)))
#define MAX_ACCEPTED_RANGE  (2*9.81)+(2*9.81)*ACC_RANGE
```

On the setup function, initiate the IMU, set the offset values and range:

```
// initialize device
Serial.println("Initializing I2C devices...");
Wire.begin();
imu.initialize();
delay(10);
//Set MCU 6050 OffSet Calibration
imu.setXAccelOffset(-4732);
imu.setYAccelOffset(4703);
imu.setZAccelOffset(8867);
imu.setXGyroOffset(61);
imu.setYGyroOffset(-73);
imu.setZGyroOffset(35);
imu.setFullScaleAccelRange(ACC_RANGE);
```

At the loop function, the buffers: buffer[ix], buffer[ix + 1], and buffer[ix + 2] will receive the 3-axis data captured by the accelerometer. On the original code, you have the line:

```
IMU.readAcceleration(buffer[ix], buffer[ix + 1], buffer[ix + 2]);
```

Change it with this block of code:

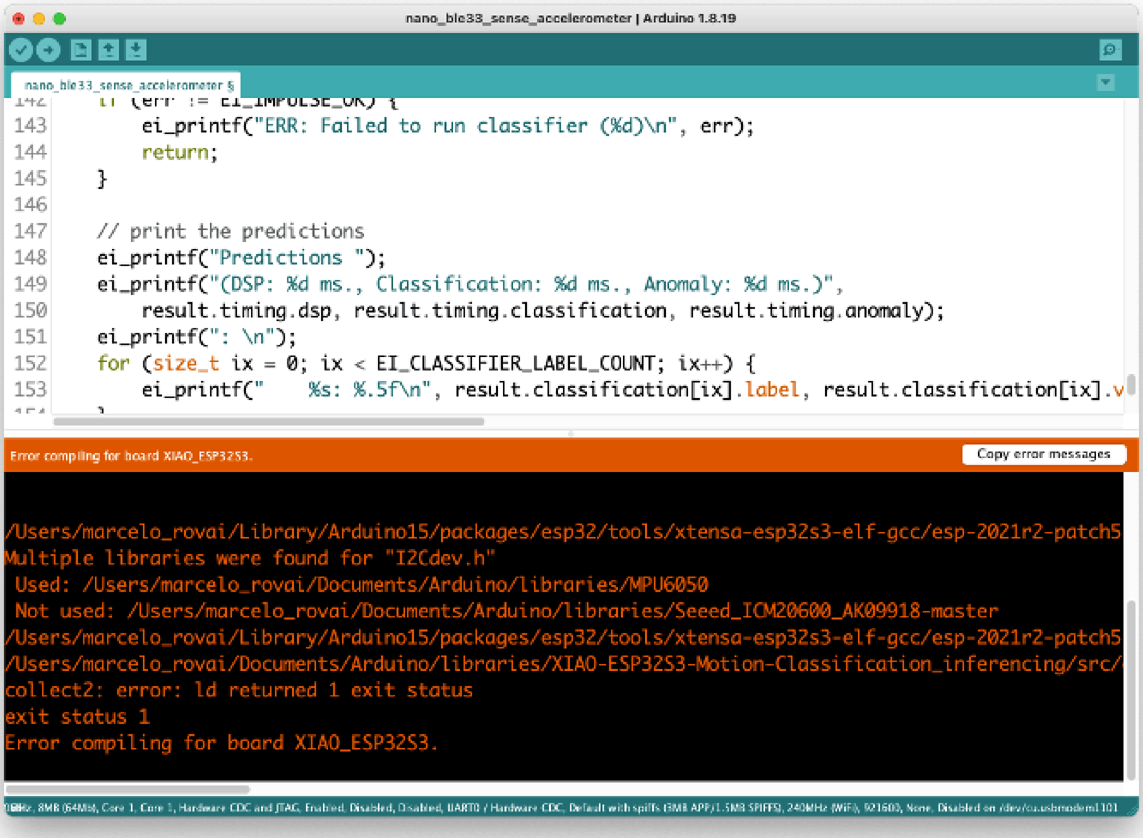
```
imu.getAcceleration(&ax, &ay, &az);
buffer[ix + 0] = ax;
buffer[ix + 1] = ay;
buffer[ix + 2] = az;
```

You should change the order of the following two blocks of code. First, you make the conversion to raw data to "Meters per squared second (ms2)", followed by the test regarding the maximum acceptance range (that here is in ms2, but on Arduino, was in Gs):

```
buffer[ix + 0] *= CONVERT_G_TO_MS2;
buffer[ix + 1] *= CONVERT_G_TO_MS2;
buffer[ix + 2] *= CONVERT_G_TO_MS2;

for (int i = 0; i < 3; i++) {
    if (fabs(buffer[ix + i]) > MAX_ACCEPTED_RANGE) {
       buffer[ix + i] = ei_get_sign(buffer[ix + i]) * MAX_ACCEPTED_RANGE;
    }
}
```

And that is it! You can now upload the code to your device and proceed with the inferences. The complete code is available on the project's GitHub.

If you get an error trying to upload the code to the XIAO ESP32S3 as below, you should switch off ESP NN acceleration.


nano_ble33_sense_accelerometer | Arduino 1.8.19
nano_ble33_sense_accelerometer §
Copy error messages

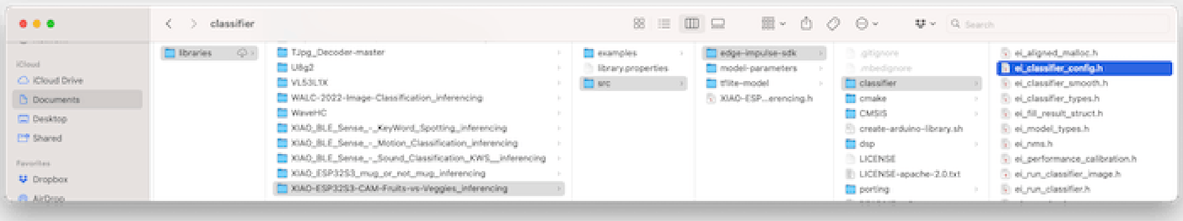
```
    if (err != EI_IMPULSE_OK) {
        ei_printf("ERR: Failed to run classifier (%d)\n", err);
        return;
    }

    // print the predictions
    ei_printf("Predictions ");
    ei_printf("(DSP: %d ms., Classification: %d ms., Anomaly: %d ms.)",
        result.timing.dsp, result.timing.classification, result.timing.anomaly);
    ei_printf(": \n");
    for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
        ei_printf("    %s: %.5f\n", result.classification[ix].label, result.classification[ix].v
```

```
Error compiling for board XIAO_ESP32S3.
```

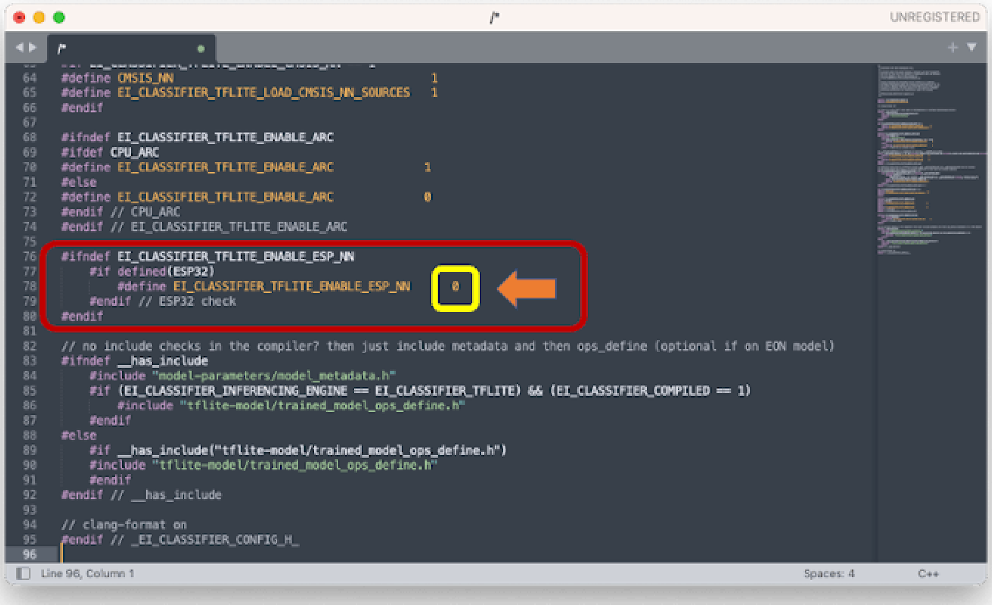
```
/Users/marcelo_rovai/Library/Arduino15/packages/esp32/tools/xtensa-esp32s3-elf-gcc/esp-2021r2-patch5
Multiple libraries were found for "I2Cdev.h"
 Used: /Users/marcelo_rovai/Documents/Arduino/libraries/MPU6050
 Not used: /Users/marcelo_rovai/Documents/Arduino/libraries/Seeed_ICM20600_AK09918-master
/Users/marcelo_rovai/Library/Arduino15/packages/esp32/tools/xtensa-esp32s3-elf-gcc/esp-2021r2-patch5
/Users/marcelo_rovai/Documents/Arduino/libraries/XIAO-ESP32S3-Motion-Classification_inferencing/src/
collect2: error: ld returned 1 exit status
exit status 1
Error compiling for board XIAO_ESP32S3.
```

```
Hz, 8MB (64Mb), Core 1, Core 1, Hardware CDC and JTAG, Enabled, Disabled, Disabled, UART0 / Hardware CDC, Default with spiffs (3MB APP/1.5MB SPIFFS), 240MHz (WiFi), 921600, None, Disabled on /dev/cu.usbmodem101
```

To do that, locate `ei_classifier_config.h` in exported Arduino library folder: `/scr/edge-impulse-sdk/classifier/:`

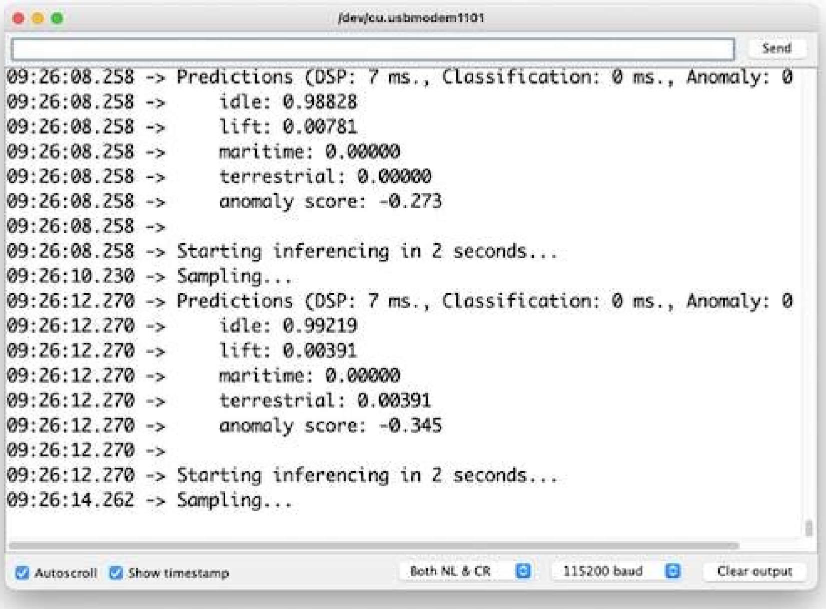


Locate the line with `#define EI_CLASSIFIER_TFLITE_ENABLE_ESP_NN 1`, and change it from 1 to 0:

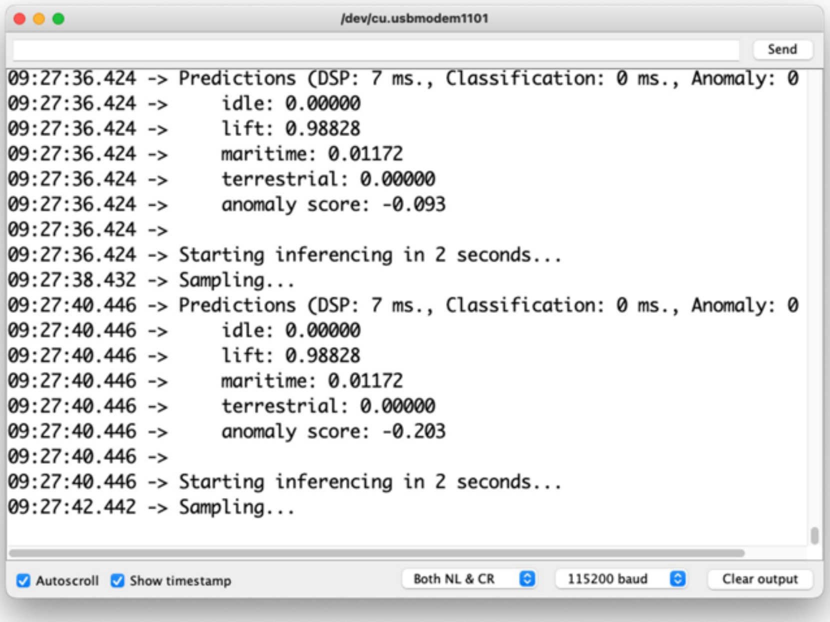


```
#define CMSIS_NN                                  1
#define EI_CLASSIFIER_TFLITE_LOAD_CMSIS_NN_SOURCES   1
#endif

#ifndef EI_CLASSIFIER_TFLITE_ENABLE_ARC
#ifdef CPU_ARC
#define EI_CLASSIFIER_TFLITE_ENABLE_ARC             1
#else
#define EI_CLASSIFIER_TFLITE_ENABLE_ARC             0
#endif // CPU_ARC
#endif // EI_CLASSIFIER_TFLITE_ENABLE_ARC

#ifndef EI_CLASSIFIER_TFLITE_ENABLE_ESP_NN
    #if defined(ESP32)
        #define EI_CLASSIFIER_TFLITE_ENABLE_ESP_NN      0
    #endif // ESP32 check
#endif

// no include checks in the compiler? then just include metadata and then ops_define (optional if on EON model)
#ifndef __has_include
    #include "model-parameters/model_metadata.h"
    #if (EI_CLASSIFIER_INFERENCING_ENGINE == EI_CLASSIFIER_TFLITE) && (EI_CLASSIFIER_COMPILED == 1)
        #include "tflite-model/trained_model_ops_define.h"
    #endif
#else
    #if __has_include("tflite-model/trained_model_ops_define.h")
    #include "tflite-model/trained_model_ops_define.h"
    #endif
#endif // __has_include

// clang-format on
#endif // _EI_CLASSIFIER_CONFIG_H_

```

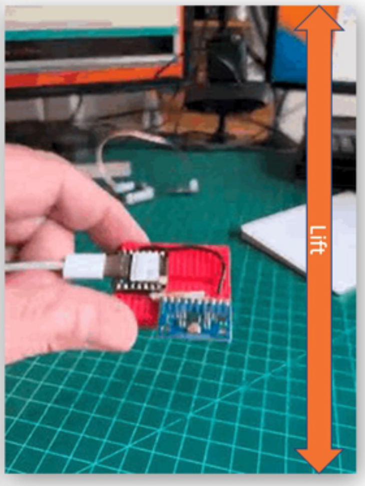
Now you should try your movements, seeing the result of the inference of each class on the images:

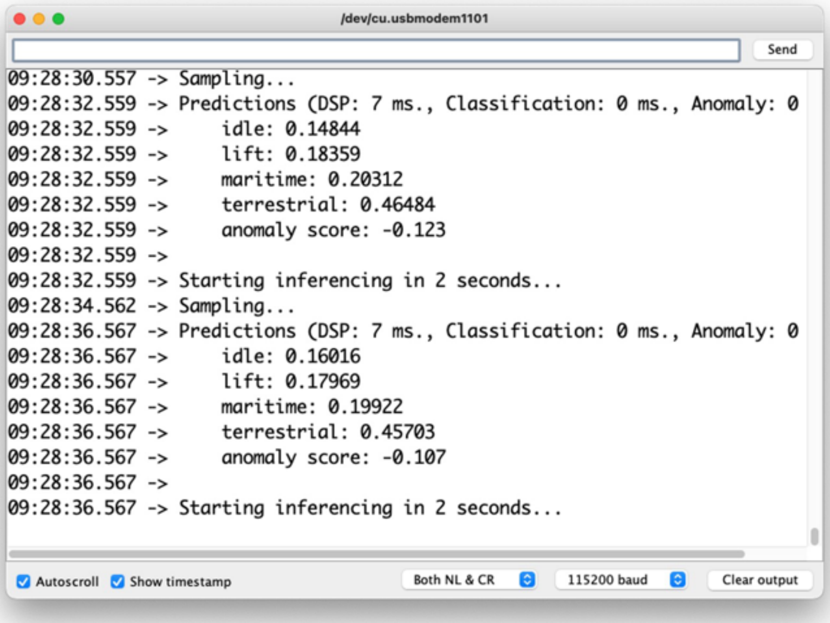


```
09:26:08.258 -> Predictions (DSP: 7 ms., Classification: 0 ms., Anomaly: 0
09:26:08.258 ->     idle: 0.98828
09:26:08.258 ->     lift: 0.00781
09:26:08.258 ->     maritime: 0.00000
09:26:08.258 ->     terrestrial: 0.00000
09:26:08.258 ->     anomaly score: -0.273
09:26:08.258 ->
09:26:08.258 -> Starting inferencing in 2 seconds...
09:26:10.230 -> Sampling...
09:26:12.270 -> Predictions (DSP: 7 ms., Classification: 0 ms., Anomaly: 0
09:26:12.270 ->     idle: 0.99219
09:26:12.270 ->     lift: 0.00391
09:26:12.270 ->     maritime: 0.00000
09:26:12.270 ->     terrestrial: 0.00391
09:26:12.270 ->     anomaly score: -0.345
09:26:12.270 ->
09:26:12.270 -> Starting inferencing in 2 seconds...
09:26:14.262 -> Sampling...
```

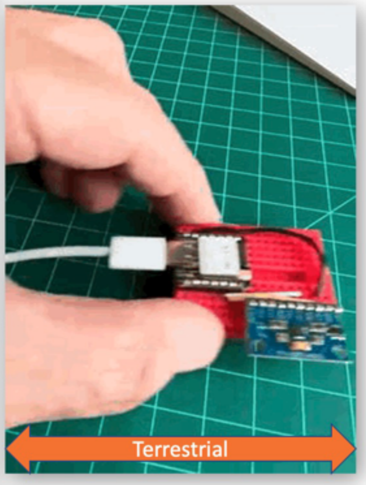


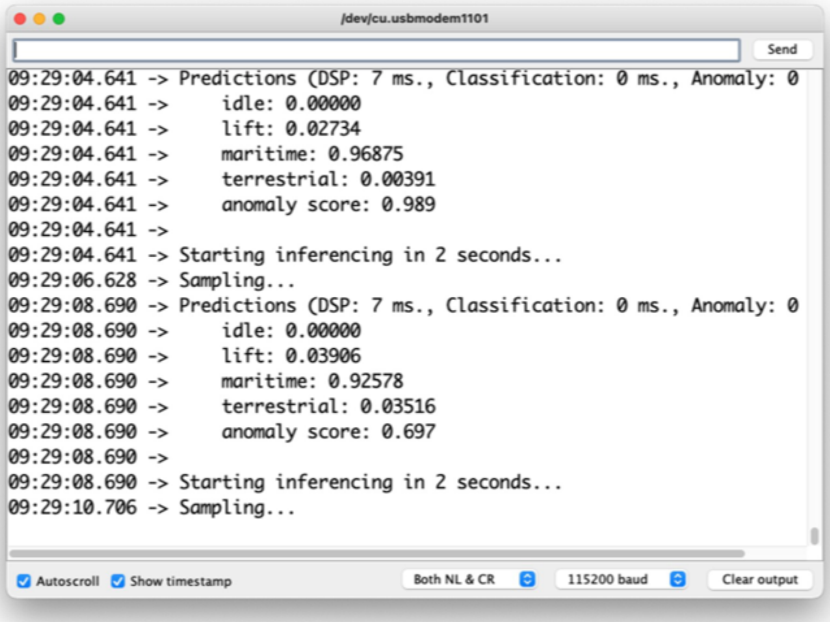
```
09:27:36.424 -> Predictions (DSP: 7 ms., Classification: 0 ms., Anomaly: 0
09:27:36.424 ->     idle: 0.00000
09:27:36.424 ->     lift: 0.98828
09:27:36.424 ->     maritime: 0.01172
09:27:36.424 ->     terrestrial: 0.00000
09:27:36.424 ->     anomaly score: -0.093
09:27:36.424 ->
09:27:36.424 -> Starting inferencing in 2 seconds...
09:27:38.432 -> Sampling...
09:27:40.446 -> Predictions (DSP: 7 ms., Classification: 0 ms., Anomaly: 0
09:27:40.446 ->     idle: 0.00000
09:27:40.446 ->     lift: 0.98828
09:27:40.446 ->     maritime: 0.01172
09:27:40.446 ->     terrestrial: 0.00000
09:27:40.446 ->     anomaly score: -0.203
09:27:40.446 ->
09:27:40.446 -> Starting inferencing in 2 seconds...
09:27:42.442 -> Sampling...
```



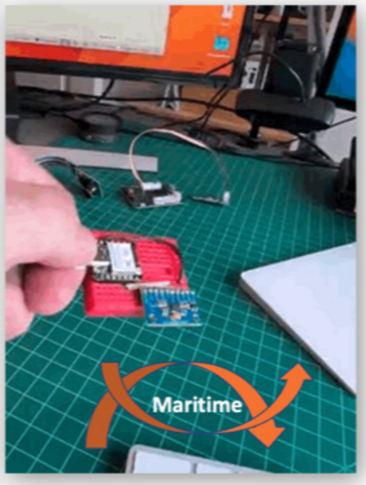


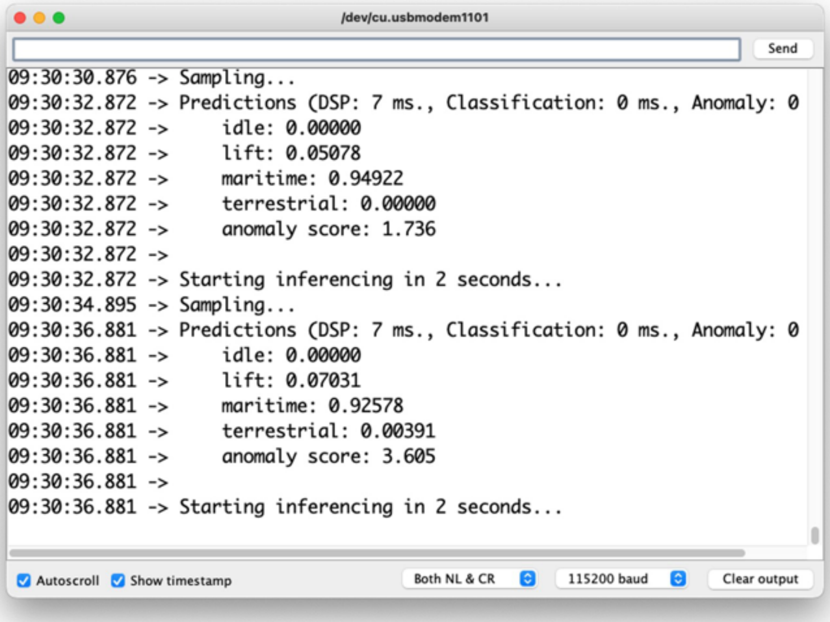
```
09:28:30.557 -> Sampling...
09:28:32.559 -> Predictions (DSP: 7 ms., Classification: 0 ms., Anomaly: 0
09:28:32.559 ->     idle: 0.14844
09:28:32.559 ->     lift: 0.18359
09:28:32.559 ->     maritime: 0.20312
09:28:32.559 ->     terrestrial: 0.46484
09:28:32.559 ->     anomaly score: -0.123
09:28:32.559 ->
09:28:32.559 -> Starting inferencing in 2 seconds...
09:28:34.562 -> Sampling...
09:28:36.567 -> Predictions (DSP: 7 ms., Classification: 0 ms., Anomaly: 0
09:28:36.567 ->     idle: 0.16016
09:28:36.567 ->     lift: 0.17969
09:28:36.567 ->     maritime: 0.19922
09:28:36.567 ->     terrestrial: 0.45703
09:28:36.567 ->     anomaly score: -0.107
09:28:36.567 ->
09:28:36.567 -> Starting inferencing in 2 seconds...
```



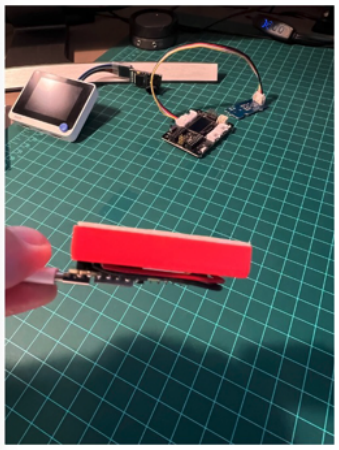


```
09:29:04.641 -> Predictions (DSP: 7 ms., Classification: 0 ms., Anomaly: 0
09:29:04.641 ->     idle: 0.00000
09:29:04.641 ->     lift: 0.02734
09:29:04.641 ->     maritime: 0.96875
09:29:04.641 ->     terrestrial: 0.00391
09:29:04.641 ->     anomaly score: 0.989
09:29:04.641 ->
09:29:04.641 -> Starting inferencing in 2 seconds...
09:29:06.628 -> Sampling...
09:29:08.690 -> Predictions (DSP: 7 ms., Classification: 0 ms., Anomaly: 0
09:29:08.690 ->     idle: 0.00000
09:29:08.690 ->     lift: 0.03906
09:29:08.690 ->     maritime: 0.92578
09:29:08.690 ->     terrestrial: 0.03516
09:29:08.690 ->     anomaly score: 0.697
09:29:08.690 ->
09:29:08.690 -> Starting inferencing in 2 seconds...
09:29:10.706 -> Sampling...
```





```
09:30:30.876 -> Sampling...
09:30:32.872 -> Predictions (DSP: 7 ms., Classification: 0 ms., Anomaly: 0
09:30:32.872 ->     idle: 0.00000
09:30:32.872 ->     lift: 0.05078
09:30:32.872 ->     maritime: 0.94922
09:30:32.872 ->     terrestrial: 0.00000
09:30:32.872 ->     anomaly score: 1.736
09:30:32.872 ->
09:30:32.872 -> Starting inferencing in 2 seconds...
09:30:34.895 -> Sampling...
09:30:36.881 -> Predictions (DSP: 7 ms., Classification: 0 ms., Anomaly: 0
09:30:36.881 ->     idle: 0.00000
09:30:36.881 ->     lift: 0.07031
09:30:36.881 ->     maritime: 0.92578
09:30:36.881 ->     terrestrial: 0.00391
09:30:36.881 ->     anomaly score: 3.605
09:30:36.881 ->
09:30:36.881 -> Starting inferencing in 2 seconds...
```

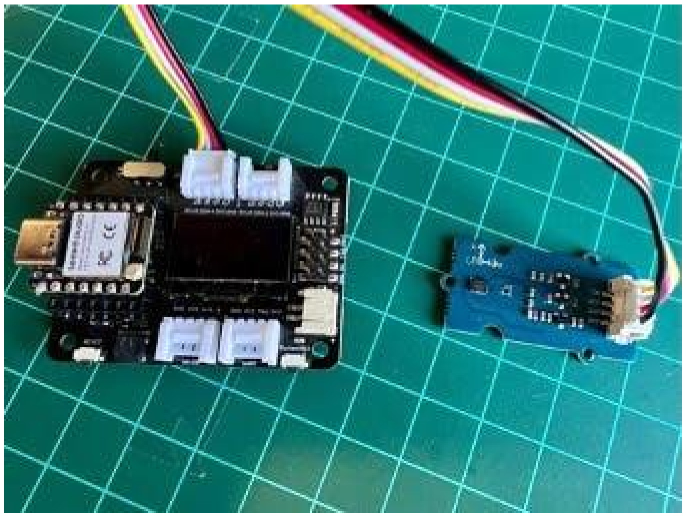


## Conclusion

The Seeed XIAO ESP32S is a giant tiny device! It is powerful, not expensive, low power, and suitable for use on the most common embedded machine learning applications. Even though Edge Impulse does not officially support XIAO BLE Sense, we realized it could be easily connected with the Studio.

Regarding the IMU, this project used the low-cost MPU6050 but could also use other IMUs, for example, the LCM20600 (6-axis), which is part of the Seeed Grove - IMU 9DOF (lcm20600+AK09918).

One advantage of the last device is that it has integrated a Grove connector, which can be helpful in teaching in the case you are using the XIAO with an extension board, as shown below:



You can follow the instruction here to connect the IMU with the MCU. Only note that for using the Grove ICM20600 Accelerometer, it is essential to update the files **I2Cdev.cpp** and **I2Cdev.h** that you will download from the library provided by Seeed Studio. For that, replace both files from this link.

You can find on the GitHub project a sketch for testing the IMU: accelerometer_test.ino.

On the project's GitHub repository, you will find the last version of all codes and other docs: XIAO-ESP32S3 - IMU.

## Knowing more

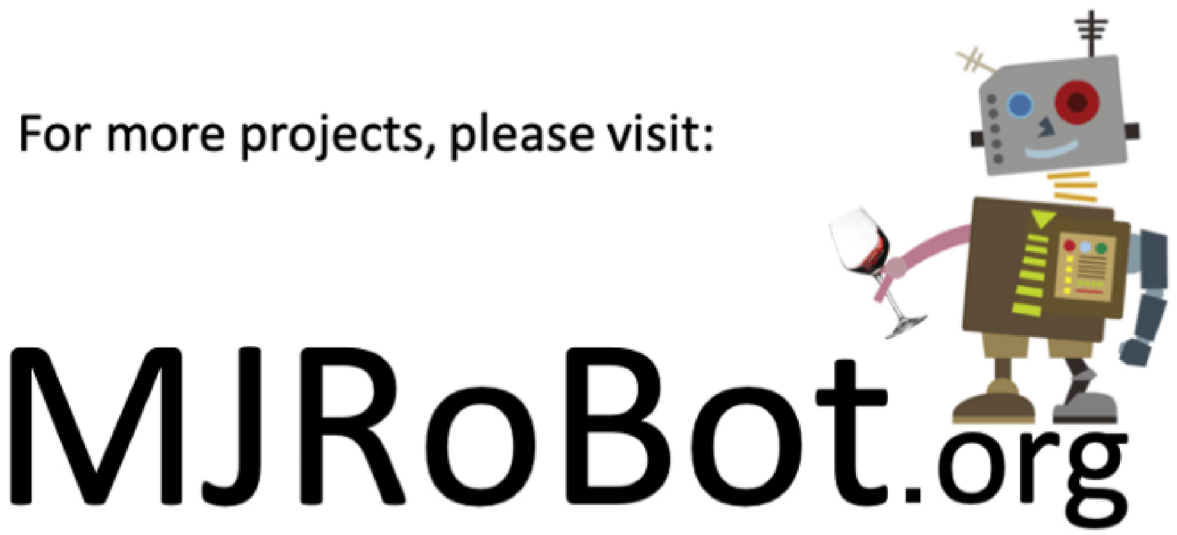
If you want to learn more about Embedded Machine Learning (TinyML), please see these references:

- "TinyML - Machine Learning for Embedding Devices" - UNIFEI
- "Professional Certificate in Tiny Machine Learning (TinyML)" – edX/Harvard
- "Introduction to Embedded Machine Learning" - Coursera/Edge Impulse
- "Computer Vision with Embedded Machine Learning" - Coursera/Edge Impulse
- "Deep Learning with Python" by François Chollet
- “TinyML” by Pete Warden, Daniel Situnayake
- "TinyML Cookbook" by Gian Marco Iodice
- "AI at the Edge" by Daniel Situnayake, Jenny Plunkett

On the TinyML4D website, You can find lots of educational materials on TinyML. They are all free and open-source for educational uses – we ask that if you use the material, please cite them! TinyML4D is an initiative to make TinyML education available to everyone globally.

**That's all, folks!**

As always, I hope this project can help others find their way into the exciting world of AI!



link: MJRoBot.org

Greetings from the south of the world!

See you at my next project!

Thank you

Marcelo