ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΜΣ « Τεχνολογίες και Υπηρεσίες Ευφυών Συστημάτων Πληροφορικής και Επικοινωνιών »

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Εφαρμογή και αξιολόγηση τεχνικών και αλγορίθμων μηχανικής μάθησης σε ενσωματωμένα συστήματα στο χώρο της ορθοπεδικής αποκατάστασης»

ΣκαρμίντΖος Βασίλειος

ΕΠΙΒΛΕΠΩΝ: Χρήστος Αντωνόπουλος

ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΠΑΤΡΑ, ΑΥΓΟΥΣΤΟΣ 2022

University of Peloponnese

Department of Electrical and Computer Engineering

Technologies and Services of Smart Information Systems and Communications

«Application and evaluation of machine learning techniques and algorithms in embedded systems in the field of orthopedic rehabilitation»

Skarmintzos Vasileios

Patras, August 2022

Πάτρα, Αύγουστος 2022

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Χρήστος Αντωνόπουλος,

Επίκουρος Καθηγητής στο τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών στο Πανεπιστήμιο Πελοποννήσου

2. Νικόλαος Βώρος,

Καθηγητής στο τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών στο Πανεπιστήμιο Πελοποννήσου

3. Νίκος Πετρέλλης ,

Αναπληρωτής Καθηγητής στο τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών στο Πανεπιστήμιο Πελοποννήσου

SUMMARY

With a focus on the ideas of machine learning and the use of the techniques offered, this thesis will introduce the basic concepts of machine learning and highlight the idea of edge computing. Everything that will be mentioned aims to form the framework for the creation and evaluation of a machine learning algorithm in order to be used in an application in the field of orthopedic rehabilitation. After becoming familiar with the use of the Edge Impulse platform and the Arduino Nano 33 BLE sense development board, an attempt is made to formulate a machine learning algorithm that will manage to evaluate and classify the flexion-extension movement of the human knee at a specific angle. The separation of the angles of movement is considered important as, based on medical advice, during orthopedic rehabilitation (from surgery), the flexibility of the leg in the specific angles also determines the success of the rehabilitation process. Through explaining the main points of the Edge Impulse platform, we come to 3 cases of algorithms that are evaluated in terms of the accuracy they provide in their predictions. To emphasize the importance of edge computing in resource and energy management, the most efficient algorithm is selected, developed on the Arduino Nano 33 BLE sense development board and with appropriate changes to the source code, the decision of the algorithm is sent to a mobile device, removing the need for another system for the operation of our application.

Key Words:

Machine learning, Edge Computing, Edge Impulse, Arduino Nano 33 BLE, orthopedic rehabilitation

Table Of Contents

Table Of Pictures

Table of Tables

Introduction

In the context of familiarization with the field of edge computing through the application of machine learning techniques, a reference will be made to the concepts of Artificial Intelligence, machine learning and deep learning. The importance of machine learning algorithms in creating solutions for a variety of applications that can improve many aspects of the daily lives of people, businesses and society in general is highlighted.

The problem dealt with in this thesis concerns the application of machine learning algorithms in orthopedic rehabilitation. In other words, it looks forward to the creation of a system for monitoring rehabilitation exercises for orthopedic problems through a specific platform and with the use of modern tools and sensors.

A special reference is made to the machine learning algorithms and the concept of edge computing. Regarding edge computing, it is about the effort to save processing and communication resources in order to create a flexible and at the same time user-friendly application for orthopedic rehabilitation after an accident or operation in the knee area. For this reason, a relevant reference is also made to the anatomical structure of the human knee, in order to understand the use cases of the application as well as the basic metric points that allow an orthopedist to judge the condition and degree of rehabilitation of the patient.

In the next phase, an acquaintance is made with the tools that are recommended in order to start the creation of the above application. In the first phase, the Arduino Nano 33 BLE sense board is presented. It is a board of the Arduino family which, with its small size but the multitude of memory and IMU sensors, creates an ideal system to be used at a practical level for the implementation of the idea. A guided tour of the Edge Impulse platform is also attempted. This is the basic platform from which the board gains its functionality. Through this, the necessary input data will be collected and they will be the trigger for the creation of a machine learning model that gives an implementation of the application.

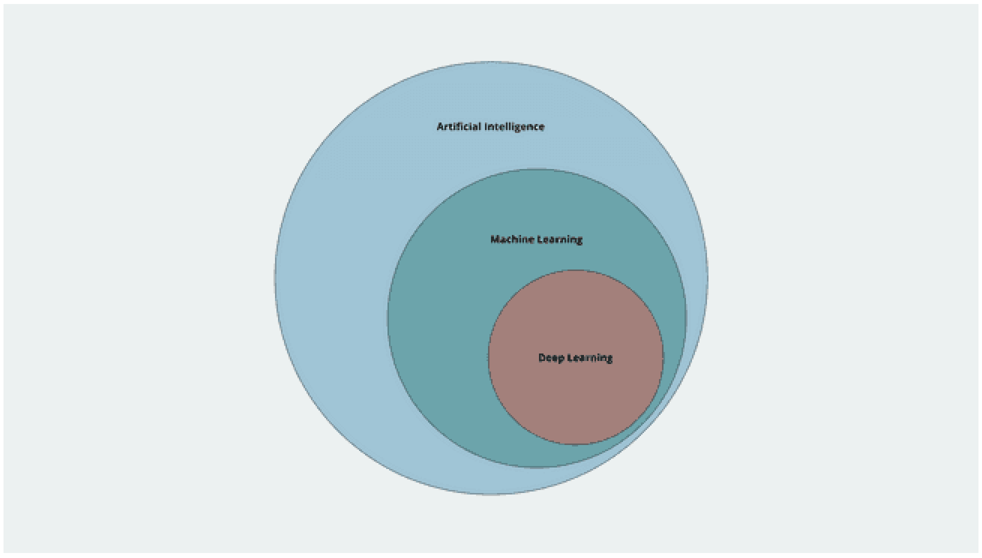
In the implementation of the experimental process, using a goniometer as well as a ROM knee instructor, the Arduino board is placed on it and the recording of movements starts. We are concerned here with the ability of the algorithm to separate the angles during the extension

and flexion of the leg, as this will facilitate the work of the orthopedist through a series of exercises that the patient will perform. After the data have been collected, we proceed with their appropriate processing. Then we enter them into the algorithm to train the model. With proper parameterization and after training cycles we have a model that more or less accurately provides a way to implement the restoration application.

Finally, a consolidated observation of the resulting models is made. We distinguish the accuracy and F1-score metrics that facilitate us in having a clear picture of how well the models created give the correct predictions in the application of interest.

The possibility provided by the edge impulse platform, to export the model and load it on the Arduino device, makes the exploitation of edge computing evident for the creation of functional and efficient algorithms with little processing power and less energy consumption. For that reason, the best algorithm is chosen and the source code of it is extracted. The code then is modified in order to enable the ability Arduino Nano 33 BLE sense offers, to work with Bluetooth Low Energy technology. With BLE enabled, the decision of the algorithm is send to a mobile device while all the “hard” work is executed on device !

1 About Machine Learning



Picture 1.1: Artificial Intelligence, Machine Learning and Deep Learning

1.1 Artificial Intelligence & Machine Learning

Artificial Intelligence and Machine learning are two terms that are increasingly used nowadays. People tend to believe that they describe the same concept in different words. However, this belief is not the reality. In this chapter, an attempt to clarify these two broad parts of computer science will be made. A basic understanding of the key concepts involved is very important, in order to separate them and find the relations between them.

1.1.1 What is Artificial Intelligence

Artificial intelligence (A.I.) describes the field of science which gives a computer system the ability to mimic or even go beyond human capabilities. It refers **to all the ideas** that drive to the simulation of human way of thinking by machines, that are programmed by humans and imitate their actions. The main goal of A.I.is to give the ability to systems to solve complex problems and perform human tasks. Math and logic are used to acquire information from the human environment, properly process them and learn from them in order to make decisions. In general, it is an organized set of actions through which a machine may achieve a specific goal.

1.1.2 What is machine learning?

In an abstract approach, Machine Learning is a branch of Artificial Intelligence, which expresses the capability of machine to imitate human behavior.[2] (Picture 1.1)

Another definition is that it is a field devoted to understanding and building methods that 'learn', that is, methods that leverage data to improve performance on some set of tasks. With Machine Learning, the systems learn automatically from data, and try to give accurate outputs.

Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so. They are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks [14]

While, in general, making a program to execute certain tasks is a basic concept nowadays, there are some cases where programming is time-consuming or maybe impossible/inappropriate to solve a problem. For example, building a program to understand human faces is a difficult problem to solve. The approach of machine learning is to let computers learn themselves -through experience-how they can solve problems like the one mentioned above.

The concept here is that first of all we have to collect data (from various sources as pictures, numbers, records, sensors etc.). These data are properly processed to be used as training and test set meaning the data with which the machine learning will be trained and then be tested for accuracy respectively.[2]

From there on, the user/programmer can choose which learning model (algorithm) is suitable for the problem. The model then will train itself and will make patterns and predictions in order to give solutions and answers to the problem the researcher investigates. Over time the human programmer can also tweak the model, including changing its parameters, to help push it toward more accurate results.

At this point, it would be useful to (also) understand the idea of Deep Learning. Deep learning is a subset of machine learning. It is a very efficient way to implement Machine Learning. We refer to deep learning algorithms, which are set of programmable actions that imitate the way a human brain would work in order to learn. These algorithms use mathematical models to acquire a set of data, properly process the information and bring a final output, which could solve a real-world problem

1.1.3 Main parts of Machine learning

An approach can be done in order to break out the learning system of a machine learning algorithm into three main parts [13]:

- **A Decision Process**: In general, machine learning algorithms are used to make a prediction or classification. Based on some input data, which can be labelled or unlabeled, your algorithm will produce an estimate about a pattern in the data.
- **An Error Function:** An error function serves to evaluate the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model.
- **A Model Optimization Process**: If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this evaluate and optimize process, updating weights autonomously until a threshold of accuracy has been met.

1.1.4 Machine Learning Algorithms

Depending on what answer or feedback the model gives as an output, we can distinguish three broad categories of algorithms [9]:

1. **Supervised Learning**: In this category, the algorithms are fed with set of data that contains both the inputs and desired outputs. In other words, the datasets used are pre-labeled. Each training example contains one or more inputs (in a representation of a vector) and also the desired output. The model adjusts the weights of a neural network until the model is fitted. There are two main areas where supervised learning is useful**: classification problems and regression problems**. The final purpose of this approach is for the machine to try and find the general rules and connections between inputs and outputs, so that it can distinguish or predict any other inputs that comes when the machine is trained.
2. **Unsupervised learning**: These algorithms discover hidden patterns or data groupings without the need for human intervention. The dataset used here is unlabeled, meaning the training set does not contain any desired outcome. The neural network here tries to identify patterns and connections between data without the need of any intervention. Depending on what problem needs to be

solved, the output data can be organized differently. One way is clustering, in which a collection of images for example can be the input dataset. Relying on common features these images have, and without any help from the researcher, the model tries to find similarities in these datasets and groups them together. Another use is this of anomaly detection. The machine can detect any data that do not follow the learning pattern and flag these outliers. A good usage may be in bank transactions or to distinguish inputs that cannot be categorized.

3. **Semi-supervised learning**: The dataset contains structured and unstructured data, which guides the algorithm on its way to making independent conclusions. The combination of the two data types in one training dataset allows machine learning algorithms to learn to label unlabeled data. Common situations for this kind of learning are medical images like CT scans or MRIs.

1.1.5 Machine Learning Models

In order to bring better results by using Machine Learning, a series of machine learning algorithms for supervised learning has been developed. Most common algorithms are explained below [13]:

- **Linear Regression:** The purpose here is to predict continuous outcomes (any numerical outcome). The aim of the algorithm is to observe the variability of the input values in relevance to some independent values, such as time. If the values tend to increase, decrease etc., the model can learn and predict the next possible values.
- **Logistic Regression**: This regression type is more common for classification problems. This logistic model predicts in which given category could a value belong. For example, given certain values, the model can predict if the given characteristics belong to a male or a female person. It works better with binary classification problems.
- **Neural Networks**: A neural network is more complex than logistic regression. This algorithm tries to imitate human brain behavior. Nodes are used here, to make any process and calculation needed, in order to have a final output. Through one input layer, the data are inserted into a neural network. Many hidden layers, with numerous nodes, interconnect. The input data are then “fed-forward” to this network. The so-called hidden layers make all the necessary processes to bring an output (output layer). The weight of a connection between two neurons/nodes in a layer is randomly determined at the beginning of the training process Through repetitive processes and re-evaluation of weights between neurons (nodes) and with a suitable activation function (a process

called back-propagation) the network succeeds to “learn”. So, any new given input can then (successfully or not) be categorized to a class. Deep learning models can be distinguished from other neural networks because deep learning models employ more than one hidden layer between the input and the output. This enables deep learning models to be sophisticated in the speed and capability of their predictions.

- **Decision Trees**: Decision trees are data structures with nodes that are used to test against some input data. The input data is tested against the leaf nodes down the tree to attempt to produce the correct, desired output. They are easy to visually understand due to their tree-like structure and can be designed to categorize data based on some categorization schema.
- **Random Forest**: Random Forest models are capable of classifying data using a variety of decision tree models all at once. Like decision trees, random forests can be used to determine the classification of categorical variables or the regression of continuous variables.

1.1.6 Machine Learning Metrics

In order to evaluate the machine learning model that was generated for a certain application, various metrics are proposed. We can find a wide variety of metrics, which find practical use in a plethora of applications. In fact, depending on the problem that the machine learning model tries to solve, there are different metrics which try to describe in mathematical way how good the model is. The combination of the metrics is very helpful as they can provide a better picture of how the model works and the researcher can proceed to the necessary adjustments to work even better.

Before talking about the various metrics, first a reference to the confusion matrix needs to be made. When we talk about classification problems-where any input needs to be classified in an actual given class- a key element is the confusion or error matrix. Each column of this matrix represents instances of an actual class while each row represents instances of predicted classes.[11]

Four basic characteristics of the confusion matrix define the metrics of a model (Picture 1.2):

- TP (True Positive): TP represents the number of correct predictions of the model. E.g., if we have to predict if an image shows a cat, in a total of some images, then we talk about the correct predictions of images that include a cat.
- TN (True Negative): TN represents the number of correctly negative class. (Meaning, in a total of images, the model correctly spots the non-existence of a cat)
- FP (False Positive): FP represents the number of misclassified predictions. Here we talk about the prediction of cat-existence in images that do not contain one. FP is also known as a Type I error.
- FN (False Negative) : FN represents the number of misclassified predictions. Here we talk about the prediction of NO cat-existence in images that do not contain one. FN is also known as a Type II error.

From the above image it is apparent that the diagonal elements of the matrix show the correct predictions for different classes while false negative and positive elements denote the misclassifications.

For classification problems, the most common metrics are:

- Accuracy: One of the samples metrics defined as the correct predictions divided by total number of predictions
- Precision: When we have an imbalance in the distribution of classes (meaning that one class is more frequent than others) we will have false indications according to accuracy metric. So, by looking a class specific metric we have a better understanding of the model effectiveness. Precision is calculated as below:

Precision POSITIVE= True Positive/ (True Positive+ False Positive)

Or

Precision NEGATIVE= True Negative/ (True Negative+ False Negative)

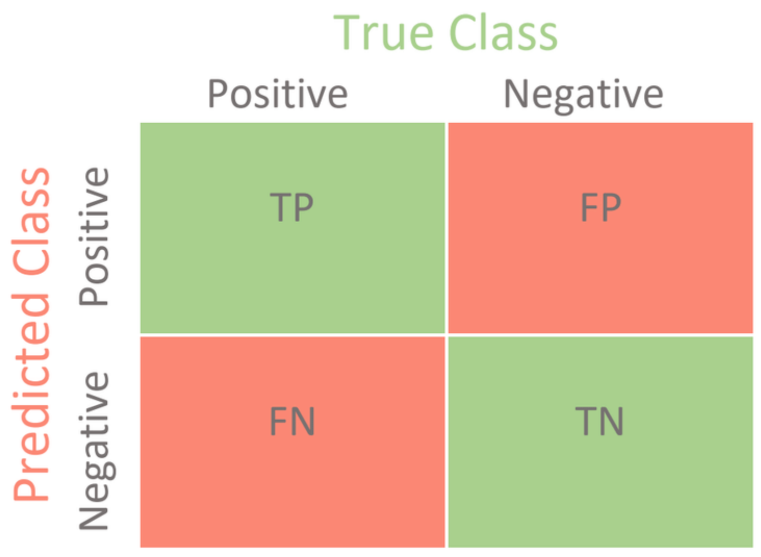
In that way, we can see in which class the model has better precision.

- Recall: the fraction of samples from a class which are correctly predicted by the model

Recall= True Positive/ (True Positive+ False Negative)

- F1-score: It is a metric that combines precision and recall, and is defined as

F1-score= 2*Precision*Recall/ (Precision Recall)



Picture 1.2 Confusion Matrix for Binary Classification

1.1.7 ML Use Cases

Machine Learning algorithms and applications have already started to be used in everyday life activities. Many examples can be encountered:

- **Natural Language Processing (NLP):** It has to do with the ability of the machine to recognize human speech and turn it into written format. Many mobile devices give users the chance to give oral command in order to search something or to give accessibility around texting
- **Chatbots:** Replacing interaction with human in order to provide customer service is another way we use machine learning today. Online chatbots are specially designed machine learning models, which have the ability to provide proper answers to written questions of users. Frequently asked questions around topics or personalized advice for consumers are two of the most basic areas where machine learning (through chatbots) is being used. Virtual assistants or agents are now available in almost any website that offers services to consumers.
- **Computer Vision:** Computers now have the ability to process images to extract useful information or take actions. Digital images, videos and other visual inputs are provided to the machine learning algorithms and many applications arise such as photo tagging in social media, radiology imaging in healthcare, and self-driving cars within the automotive industry.

The value of machine learning has not left most industries and organizations that use large amounts of data unconcerned. By properly collecting data and feeding these data to machine learning algorithms, they are able to work more efficiently and develop the areas of interest:

Financial Area: Machine Learning technology is trusted by banks and other financial institutions as it offers a good analysis of data and also security. For example, investment opportunities and trade suggestions could be extracted using machine learning. In addition, with the use of anomaly detection, any tries of fraud or false information can be spotted fast (cybersurveillance).

Governmental Use: Many ministries and governmental organizations have multiple sources of data. These data are very important because with proper processing many conclusions are drawn. Machine learning is a handy and practical way as efficiency is increased, fraud and identity theft can be detected and in general public money can be saved.

Electronic Retail: Personalized shopping experience is another area where we can find machine learning. When a consumer visits a web site, many information is gathered, regarding his/her choices, buying history and past purchases. By implementing machine learning algorithms and using these data big retail companies can develop a new campaign, optimize the experience of users and also deploy a marketing strategy.

Transportation: Public transportation, delivery companies and in general any kind of transportation industry can take advantage of the solutions machine learning algorithms offer. By trying to find best routes or organize a better delivering pattern can save a lot of money to these kinds of organizations. Analyzing these data is important and with machine learning solutions, the benefits are certain.

Health Care: Wearable devices and sensors have brought a revolution to medical activities, as they assess the acquisition of patients' data in real time. Doctors and Medical staff can now have a completely different approach on how they treat patients and may have improved diagnoses as with machine learning they can use the data collected for applications that make patients' monitor easier and also better rehabilitation techniques.

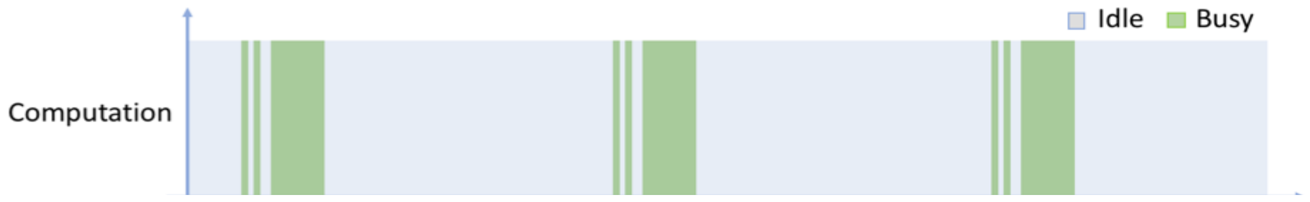
1.2 Machine Learning on the Edge

Indeed, the idea of using machine learning on the edge is of high interest. As it is apparent, in the following years billions of small intelligent devices will be interconnected. These devices will be embedded in buildings, vehicles, cities, factories even in our bodies as wearable devices. The use of these devices will interact with the environment or with other devices. This interaction includes the collection of data through sensors. Examples of such resource-constrained IoT devices include sensors, microphones, smart fridges, and smart lights. IoT devices and sensors continuously generate large amounts of data, which is of critical importance to many modern technological applications.[7]

The best way to use all the data that come from these devices is to ‘feed’ those data to a Machine learning system. The process of such data need high computational capabilities and decision is left to be done to the cloud, somewhere away from these devices. However, many applications demand immediate decision-making and make the use of cloud for machine learning purposes impractical. Another idea is to use those data and perform computations near the data source. That is the idea of Edge Computing.

When we talk about IoT devices, we primarily speak about MCUs (microcontrollers). The devices have both computational and communication capabilities. These chips have low energy needs (meaning they can run for months on small batteries) and also have a low cost. This practically means that we have in our hands, devices with low computational power, clocked at hundreds of MHz and with hundreds of kB of RAM. Their main job is to use any sensors they may have to gather information about their surroundings. [11]

The graph below (Picture 1.3) shows the durations in which an MCU is idle (with blue color) and busy (with green color). Apparently, idle time is especially high compared to the busy time.



Picture 1.3 Idle and Busy time of an MSU

The idea is to use this idle time to our advantage. For example, a camera connected to one MCU could easily do gesture recognition, an IMU could give results about the board orientation by checking the gravity acceleration vector orientation) or to measure shocks, vibration, acceleration and rotation speed. We can create suitable models for our IoT device that can inference simple results. Our attempt is to do as much computation work as possible on device and leave other, less time-sensitive computations for the cloud.

The advantages of performing computations on the edge can be summarized below 0:

- The data gathered by sensors have a great volume. So, the need for bandwidth to transfer those data to cloud servers is high. Edge computing can drastically reduce this bandwidth
- Lower latency is also achieved, as edge devices are close to the data sources, improving the real-time data processing
- User privacy and safety of the data is another thing to keep in mind. Edge devices have the ability to discard information that need to be kept private, as only certain information and with special encoding.

Because one final decision is of our interest, an ML algorithm, running on device could minimize delays, improves privacy and consume minimum bandwidth when connected to cloud. The latter is important, as some layers of a neural network could possibly be calculated on device and the outputs could then be transferred through a gateway to cloud for further processing. So, an amount of payload is transferred to these small devices.

Machine learning techniques that have been mentioned in previous chapters require a great number of resources to work properly. The goal here is to develop appropriate algorithms that are tailored for embedded platforms. What we assume here is that a machine learning model has been trained offline on a prebuild dataset. Usually, data gathered by the edge device would be sent back to a cloud platform. This platform is responsible to perform the training and finally distribute the model that has been built back to the edge device. We can find many deployment platforms available for these purposes.

1.2.1 Machine Learning on the edge with Edge Impulse

In our case, Machine Learning platform “Edge Impulse” is a great asset. This platform let us build advanced edge ML solutions in easy and clear steps. The platform at first asks us to connect a supported edge device (e.g., raspberry pi or Arduino board). After been connected, our device can be used to acquire data and build data sets in a user-friendly way. We can even upload datasets from other sources. The next step is to design our machine learning model.

Edge Impulse gives a wide variety of ready algorithms (called blocks). Two kinds of blocks are available: Processing Blocks and Learning Blocks. In order to extract useful information from the data gathered, we can use a series of processing blocks, where (depending on the kind and source of the data) help us process the data before feeding the algorithm. Learning blocks take the features extracted and with the use of pre-defined algorithms we can train our model.

Following the training of the model, it is time to test it. We can either do a live-classification, validating the model with real time data from the edge device or use the test set, collected from the first step of the procedure.

Finally, and most importantly, Edge Impulse gives the possibility to deploy the model created (called impulse) to any of the supported devices. So, there is no need for connection for the edge device as the model runs ON DEVICE!

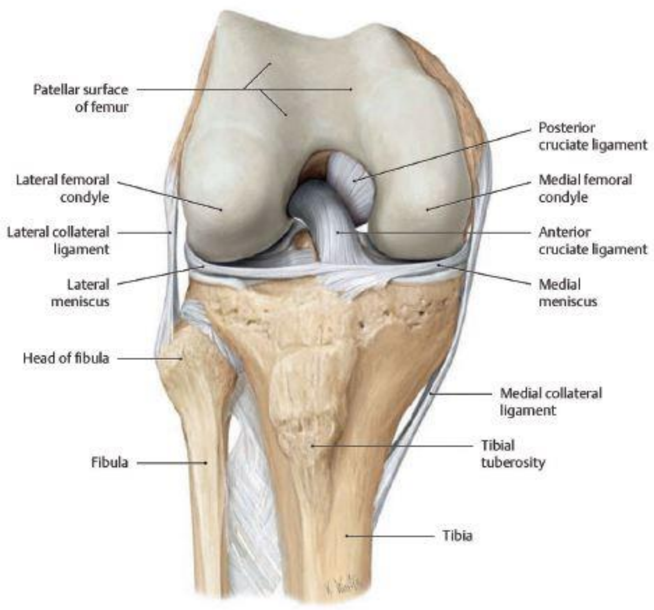
1.3 Medical theory

1.3.1 General Information about Human Knee

The knee joint (Picture 1.4) is a modified hinge joint (ginglymus). The knee joint is not a very stable joint. Knee range of motion is through following movements: [10]

- Flexion
- Extension
- Medial rotation
- Lateral rotation

The femur and lateral meniscus move over the tibia during rotation, while the femur rolls and glides over both menisci during extension-flexion. Thus, the flexion-extension movement occurs in the compartment above the menisci whereas the rotator movement occurs below the menisci.


Patellar surface of femur
Posterior cruciate ligament
Lateral femoral condyle
Medial femoral condyle
Lateral collateral ligament
Anterior cruciate ligament
Lateral meniscus
Medial meniscus
Head of fibula
Medial collateral ligament
Tibial tuberosity
Fibula
Tibia

Picture 1.4 The Knee Joint

Normal range of motion of knee is

- Flexion – 120-150 degrees
- Internal rotation with knee flexed – 10 degrees
- External rotation with knee flexed – 30-40 degrees

The total range of motion is dependent on several parameters such as soft-tissue restraints, active insufficiency, and hamstring tightness.

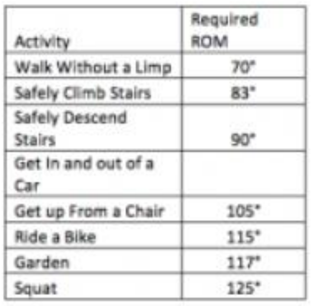
Flexion and extension are the chief movements. These take place in the upper compartment of the joint, above the menisci.

Normally the line of the tibia and femur should coincide with the full extension being recorded as 0^o. Loss of the full extension may be described as 'the knee lacks X degrees of extension'.

The flexion of the knee can be measured using a goniometer. Flexion over 120^o is regarded as normal. Loss of flexion is common after local trauma, effusion and arthritic conditions

They differ from the ordinary hinge movements in two ways:

- The transverse axis around which these movements take place is not fixed. During extension, the axis moves forwards and upwards, and in the reverse direction during flexion. The distance between the center and the articular surfaces of the femur changes dynamically with the decreasing curvature of the femoral condyles.
- These movements are invariably accompanied by rotations (conjunct rotation). Medial rotation of the femur occurs during the last 30 degrees of extension and lateral rotation of the femur occurs during the initial stages of flexion when the foot is on the ground. When the foot is off the ground, the tibia rotates instead of the femur, in the opposite direction.



Activity	Required ROM
Walk Without a Limp	70°
Safely Climb Stairs	83°
Safely Descend Stairs	90°
Get In and out of a Car	
Get up From a Chair	105°
Ride a Bike	115°
Garden	117°
Squat	125°

Table 1.1: Movements and required Range Of Motion

1.3.2 Importance of Knee Range of Motion

In order any human to be able to do certain activities, the Knee Range of Motion (ROM) is an important indication. Following a surgery on the knee or any distortion that may (accidentally) occur, therapists have to take into serious consideration the Range of Motion of the knee. In the table below, it is apparent that when someone needs to do any activity, the degrees of the bending knee play a significant role.

As we have seen, Range of Motion is a composite movement, made up by extension and flexion. The combination of the two movements, creates the ability to practice the movement of the knee. When we talk about extension, we mean that the leg is straight (0^o-completely straight leg) and flexion means a flexed leg (at around 130^o).

Extension of the knee is of great importance as it may lead to instabilities and (dangerous) falls and also, if legs cannot go completely straight then quadriceps muscles are continuously activated and are tired quickly.

Knee flexion is equally important, because it is this exact movement of the knee that lets any activity be executed. Bending the knee is crucial if one needs to stand up from a chair or safely climb stairs.

1.4 Use Case of this thesis

As we have seen in previous chapter, IoT in combination with Machine Learning can find many fields of application. For the purpose of this thesis, health care and more precisely patient rehabilitation is the basic field of interest.

Patients nowadays have the need for personalized interaction with their therapists bringing more effective treatments with cost-efficiency. At the same time, medical staff has to be prepared to offer services of high value and successful treatments to patients in need.

IoT devices have made their appearance in health care systems worldwide. Their arrival has revolutionized the way medical staff interacts with patients. However, these new and rich data sources have started to overwhelm the health systems, bringing new challenges to those who develop them. All this digitalization of the systems has provided many opportunities to transform into systems that continue to treat patients with less suffering and also meet the cost, quality and safety challenges.

Delivering health care services to patients, providing the best experience and in real-time is the goal. In this thesis, the basic idea is to take advantage of all the assets IoT offers, in combination with Machine Learning and Edge Computing. Machine learning will give us all the necessary tools to acquire, process and properly use the data that come from sensors and in that way build models that satisfy special applications for medical use.

Taking into consideration all the above information, with this thesis, we aim to apply and evaluate machine learning techniques and algorithms in integrated systems in the field of orthopedic rehabilitation. We intend to create a machine learning model with the use of Edge Impulse platform and pass this model to an Arduino Nano 33 sense board. The basic idea is to train a model that can distinguish different degrees of angles of a moving knee. By making an extend-flex movement of one’s knee, the model will be able to classify this movement in the given degrees. The importance of this classification is apparent in orthopedic exercises that a therapist may ask a patient to execute. The final purpose is to give Arduino Nano 33 the ability to send wirelessly the predictions made by the model running on device. So, the patient only needs to have a wearable device and practice the “flex-extend” exercises given by the therapist, without the need to be transported to a doctor's office or a medical/rehabilitation center.

2 Edge Impulse & Arduino Nano 33 sense board.

2.1 Edge Impulse

2.1.1 General Description

Edge Impulse is a startup company based on San Jose, USA. The primary purpose of the company is to help users build machine learning applications for embedded systems. Edge Impulse's online platform enables the implementation of ML models on microcontroller embedded devices. It is a cloud-platform, meaning the user can have access from anywhere, while the models created targets microcontrollers. Following a specific sequence of steps, the user can easily train a model and deploy it to an embedded device. With the use of modern techniques (such as TinyML and AutoML) the platform allows the ML models (with ease) to be passed to an edge device. Taking advantage of all the available hardware and sensors, many applications can be developed, including monitoring, sensing, tracking etc. The great power of the created models is that they can perform predictions or classifications without the need of a cloud service (Machine Learning on the Edge).[3]

2.1.2 Building a Machine Learning Model

The Edge Impulse platform is user-friendly and with easily applied steps enables the development of Machine Learning models.

First of all, the user can acquire data by connecting a wide range of boards (such as Arduino) and by using the variety of sensors this board has (such as accelerometers, microphones etc). There is flexibility in the creation of the datasets as the sensors, audio or camera data that come directly from the board are driven the platform

After that, the user can develop ML algorithms using the pre-built ML algorithms and Digital Signal Processing blocks that the platform offers. Feature extraction from the data acquired is another advantage of the platform, giving the user the chance to interfere with parameters that can lead to better and "clearer" data. In addition, Edge Impulse gives helpful visualizations of the datasets, in order to have a better understanding of how the data are distributed and how separatable they are.

Edge Impulse offers also a model performance testing. Before deploying to a physical device, the platform performs an early model accuracy check. Using special virtual simulation tool, can give a first image on how well the model performs and if it offers satisfactory accuracy. Furthermore, there is a live classification tool, where the user can use the device to check if the model works correctly, in live testing. The user also has the ability to test the model by using the Test Set he/she created on the first step

Finally, edge impulse gives the chance to deploy the built model to a variety of embedded devices. An optimized source code is generated, and binaries for a wide variety of development boards is available. Arduino boards and Raspberry Pi are some of the hardware edge impulse supports.

2.1.3 Edge Impulse Solutions

Some use cases suggested by Edge Impulse are the ones below:

- **Agriculture**: With the use of ML, modern agriculture can make a step forward, finding solutions on optimized harvesting, environmental-friendly and cost-effective farming practices and plant growth monitoring. Use cases: remote farm monitoring, animal health monitoring, crop health
- **Health**: New wearables can be created to monitor patient and his/her surroundings. The immediate profits are: real-time monitoring through biometric sensors based on Machine Learning and more privacy as data are processed on device and there is no need to send sensitive data. Use cases: fall detection, rehabilitation exercises monitoring, patient vital signs monitoring
- **Infrastructure**: In the way to build smarter cities, ML on the edge can bring added value. Smart grid and water network monitoring can detect malfunctions in real-time. Also, in rural environment, public safety can be benefited as there are available solutions for smart traffic monitoring to air pollution metering and respond to natural disasters.
- **Environment**: ML on the edge benefits the environment, as low power devices take a great amount of the weight of the process that needs to be made. Automations also bring better efficiency and real-time monitoring is possible with the least power usage and with minimal connectivity. Some practical examples: remote monitoring, smart animal tracking, wildfire detection etc.

- **Industry**: Industries and enterprises can increase the productivity by implementing ML on sensors and device data. Also, monitoring of the equipment is a fact and fault-prediction is crucial. Safety of people and equipment is another filed where ML can be helpful. Use cases: predictive maintenance, parcel tracking, worker protection.

2.2 Arduino Nano 33 Sense

2.2.1 General Description

Arduino Nano 33 is a member of the Arduino family. Its main purpose is to cover basic IoT applications and scenarios. Having a relatively small dimension, it is ideal to use it as an edge device. Through a series of sensors, it provides solutions to develop many IoT ideas that may occur. It also offers increased connectivity with several modules able to send and receive data.

The Arduino Nano 33 BLE Sense is an evolution of the traditional Arduino Nano, but featuring a lot more powerful processor, the nRF52840 from Nordic Semiconductors, a 32-bit ARM® Cortex®-M4 CPU running at 64 MHz It comes with a series of embedded sensors:

- 9 axis inertial sensor: what makes this board ideal for wearable devices
- Humidity and temperature sensor: to get measurements of the environmental conditions
- barometric sensor
- microphone: capture and analyze sound in real time
- gesture, proximity, light color and light intensity sensor: estimate the room’s luminosity, but also whether someone is moving close to the board

Microcontroller	NINA b-3 (nRF52840)
Operating Voltage	3.3V
Power (Input Voltage -Nominal)	4.5-21V

Clock Speed	64 MHz
CPU Flash Memory	1 Mb
SRAM	256 kB
Digital I/O pins	14
Analog Input Pins	8
Analog Output Pins	1
USB	Native in the SAMD21 processor
SENSORS	
IMU	LSM9DS1
Microphone	MP34DT05
Gesture, light, proximity	APDS9960
Barometric Pressure	LPS22HB
Temperature / Humidity	HTS221
Memory	256 KB SRAM, 1MB flash
Dimensions	
Weight	5 gr
Width	18 mm
length	45 mm

The main feature of this board, besides the impressive selection of sensors, is the possibility of running Edge Computing applications (AI) on it using TinyML .

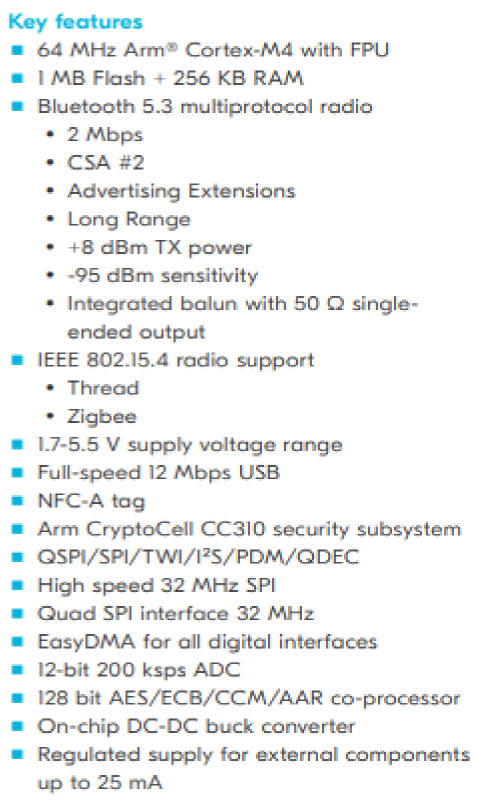
Table 2.1: Arduino Nano 33 sense board Characteristics

2.2.2 The processor

The nRF52840 is an advanced, highly flexible single chip solution for today's increasingly demanding and wireless applications for connected devices on our connected living environments and the IoT at large. It is designed ready for the major feature advancements of Bluetooth LE and takes advantage of increased performance capabilities which include long range and high throughput modes. Inherent industry-grade security is essential in today's applications. The nRF52840 adds best-in-class security for CortexTM-M Series with on-chip ARM® CryptoCell cryptographic accelerator.

The range of applications with this chip is vast. Some special applications are:

- IoT: Smart home, industrial IoT, mesh networks, smart city
- Wearables: connected watches, fitness devices, health monitor wearables, connected health



Key features

- 64 MHz Arm® Cortex-M4 with FPU
- 1 MB Flash + 256 KB RAM
- Bluetooth 5.3 multiprotocol radio
  - 2 Mbps
  - CSA #2
  - Advertising Extensions
  - Long Range
  - +8 dBm TX power
  - -95 dBm sensitivity
  - Integrated balun with 50 Ω single-ended output
- IEEE 802.15.4 radio support
  - Thread
  - Zigbee
- 1.7-5.5 V supply voltage range
- Full-speed 12 Mbps USB
- NFC-A tag
- Arm CryptoCell CC310 security subsystem
- QSPI/SPI/TWI/I²S/PDM/QDEC
- High speed 32 MHz SPI
- Quad SPI interface 32 MHz
- EasyDMA for all digital interfaces
- 12-bit 200 ksps ADC
- 128 bit AES/ECB/CCM/AAR co-processor
- On-chip DC-DC buck converter
- Regulated supply for external components up to 25 mA

2.2.3 Sensors [12]

2.2.3.1 HTS221 Capacitive digital sensor

The HTS221 is an ultra-compact sensor for relative humidity and temperature. It includes a sensing element and a mixed-signal ASIC to provide the measurement information through digital serial interfaces.

2.2.3.2 LPS22HB nano pressure sensor

The LPS22HB is an ultra-compact piezoresistive absolute pressure sensor that functions as a digital output barometer. The device comprises a sensing element and an IC interface that communicates through I2C or SPI from the sensing element to the application.

2.2.3.3 MP34DT05 audio sensor omnidirectional digital microphone

The MP34DT05-A is an ultra-compact, low-power, omnidirectional, digital MEMS microphone built with a capacitive sensing element and an IC interface.

2.2.3.4 APDS-9960 - Digital Proximity, Ambient Light, RGB and Gesture Sensor

The Broadcom APDS-9960 is a digital RGB, ambient light, proximity, and gesture sensor device in a single 8-pin package. The device has an I2C compatible interface providing red, green, blue, clear (RGBC), proximity, and gesture sensing with IR LED. The RGB and ambient light sensing feature detects light intensity under various lighting conditions and through various attenuation materials, including darkened glass. In addition, the integrated UV-IR blocking filter enables accurate ambient light and correlated color temperature sensing.

2.2.3.5 LSM9DS1 inertial module, 3D magnetometer, 3D accelerometer, 3D gyroscope, I2C, SPI

The LSM9DS1 is a system-in-package featuring a 3D digital linear acceleration sensor, a 3D digital angular rate sensor, and a 3D digital magnetic sensor.

This exact sensor is of our interest as we take advantage of the indications it produces in order to measure the changes in the position of the board.

2.2.4 Connectivity

In order to achieve the desired connectivity with Bluetooth, Bluetooth Low Energy and WiFi, the board is equipted with a special microvontroller, the uBlox NINA-W10 series.

The radio supports Wi-Fi 802.11b/g/n connectivity in the 2.4 GHz band and also Bluetooth v4.2 (Bluetooth BR/EDR and Bluetooth low energy) communications.

This module includes a wireless MCU, enough flash memory and all the necessary components to achieve a compact, standalone and multipurpose radio solution for IoT devices, such as Arduino Nano 33. Some interesting fields of application are: Internet of Things (IoT), Wi-Fi networks, Bluetooth and Bluetooth low energy applications, Telematics, Point-of-sales, Medical and industrial networking, mobile phones, home/building automation, Ethernet/Wireless Gateway etc.

3 Description of the Experiment

3.1 General Description

The purpose of this experiment is to create an embedded machine learning model that is able , to a satisfactory degree, distinguish the different positions of an adult leg. To be more precise, the model aims to discriminate between different angles that an adult can bend his/her leg. The degrees of the angles that we want the model to recognize are 20^o, 45^o, 60^o, 90^o and 120^o.

3.2 Equipment and Software

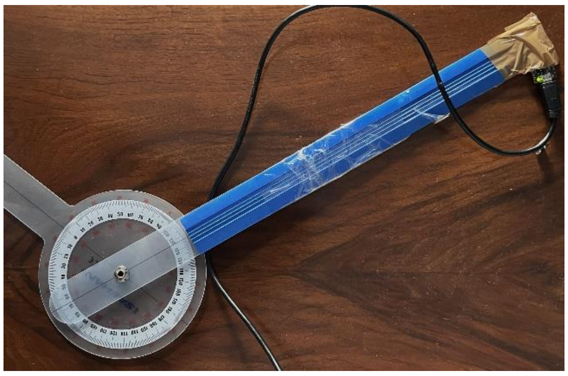
In order to achieve this purpose, we used an Arduino Nano 33 BLE sense Board. This board is considered ideal for IoT solutions because it offers a variety of sensors in small dimensions (18 x 45 mm) as it is fully supported by Edge Impulse. Using the sensors that this board has, we will record the movement of the leg. In other words, the sensors will record the up- and-down movement of ones' leg (from the knee and below) and the angle this movement "writes".

In order to measure the angles of interest we will use two tools: a universal goniometer and a telescopic RangeOfMotion knee brace with built-in goniometer . We adjust the Arduino Nano 33 to each of these devices and we proceed to the measurements.

3.3 Models to be tested- 3 Cases

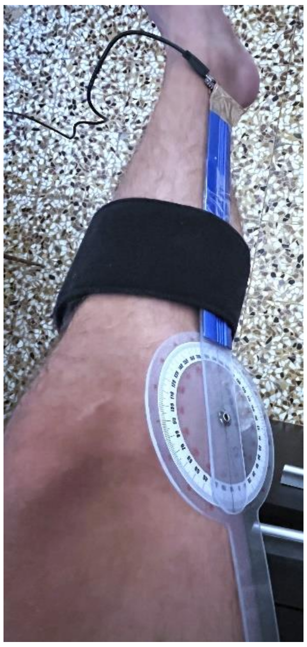
For the purposes of this experiment, three installations of the Arduino Nano 33 board were used.

1. In the first case the board was adjusted on the edge of the goniometer (Picture 3.1). The goniometer was then placed in a surface (writing desk). Then ,while holding the one brace of the goniometer steady, we started moving the other brace (with the Arduino board attached) up-and-down, "writing", in that way, the desired angle.



Picture 3.1: Goniometer with Arduino attached

2. The second case followed the same procedure but this time the goniometer was placed on the knee of an adult person (Picture 3.2). The person (starting from complete stretch of the leg – 0 degrees) started moving the leg up-and-down "writing" the desired angle, according to the indication of the goniometer.



Picture 3.2: Goniometer on knee

3. The third case involved a telescopic RangeOfMotion knee brace (Picture 3.3). This brace is a universal, easily adjustable and secure support to patients recovering from knee surgery, injuries or instabilities. The range of motion settings can be easily adjusted via the spring loaded quick-clips by a simple 'pull and slide' motion. The mechanism this device has, has degree indications so we can easily choose the range of motion of the leg in degrees. For example, by pulling and sliding the regulator to 45o, we have a range of motion (from full extend) of 45o degrees. After wearing this device, we properly stick Arduino Nano to the lowest part of the brace and we start experimenting with the different angles.

Picture 3.3: Knee Brace-Range of Motion

3.4 The Experiment step-by-step

The following chapters explain in detail the steps that were followed in order to achieve the connectivity between Arduino Nano 33 BLE sense and Edge Impulse, the procedure followed to build a machine learning model and also an evaluation of the three cases presented above, in order to result in the most accurate approach.

3.4.1 Arduino Nano 33 Board and Edge Impulse

The first thing that needs to be done is to ensure that our board Arduino Nano 33BLE sense and Edge Impulse platform are connected. So, the necessary software needs to be installed and then ensure that this connection is successful.

3.4.1.1 Installing the necessary software

In order to connect Arduino Nano 33 board to the Edge Impulse Platform the following software is necessary:

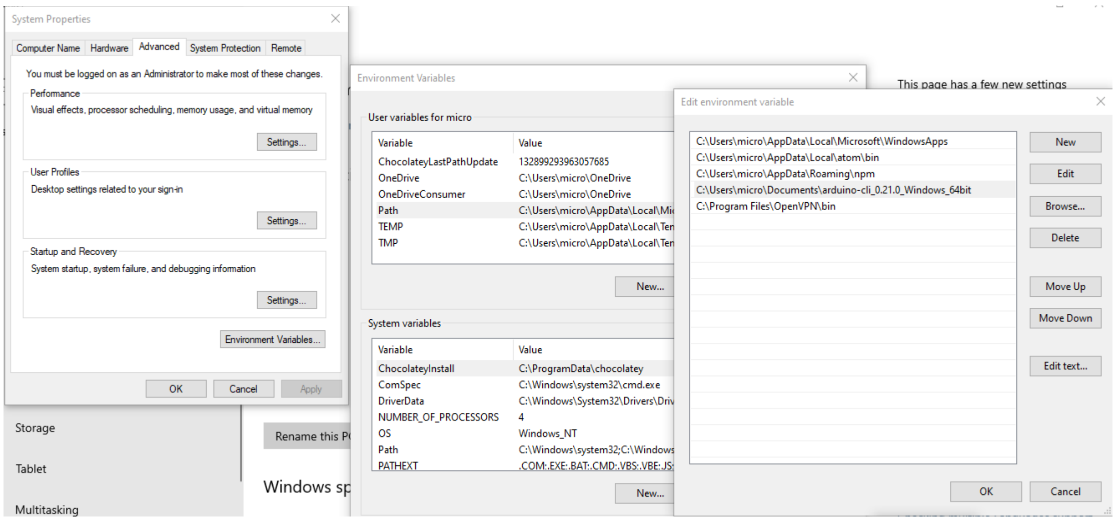
- **Edge Impulse CLI**: It is used to control local devices, acts as a proxy to synchronize data for devices that don't have an internet connection, to upload and convert local files. There are several tools available. One of great interest is **Edge-impulse-daemon.** Edge-impulse-daemon is a tool that includes all the necessary things to make devices work over serial connection, has the role of proxy for these devices that do not have an IP connection.

  To use this daemon, we run in a command prompt the following :

```
$ edge-impulse-daemon
```

The daemon starts and the prompt will ask for a username and a password. Detailed information about the installation of this software can be found in the official documentation page of Edge impulse [https://docs.edgeimpulse.com/docs/edge-impulse-cli/cli-installation]

- Arduino CLI: an all-in-one solution that provides Boards/Library Managers, sketch builder, board detection, uploader, and many other tools needed to use any Arduino compatible board and platform from command line or machine interfaces. Arduino CLI is the heart of all official Arduino development software (Arduino IDE, Arduino Web Editor). We visit the web site : https://arduino.github.io/arduino-cli/0.24/installation/ where we download the installation file. After installation, we take the path of the downloaded file and add this location to Environment Variables as shown in the following picture (Picture 3.4). Detailed information about the installation of this software can be found in the official installation page of Arduino CLI [https://arduino.github.io/arduino-cli/0.23/installation/].



You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles
Desktop settings related to your sign-in

Startup and Recovery
System startup, system failure, and debugging information

User variables for micro

Variable	Value
ChocolateyLastPathUpdate	132899293963057685
OneDrive	C:\Users\micro\OneDrive
OneDriveConsumer	C:\Users\micro\OneDrive
Path	C:\Users\micro\AppData\Local\Mi
TEMP	C:\Users\micro\AppData\Local\Ten
TMP	C:\Users\micro\AppData\Local\Ten

System variables

Variable	Value
ChocolateyInstall	C:\ProgramData\chocolatey
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\Driv
NUMBER_OF_PROCESSORS	4
OS	Windows_NT
Path	C:\Windows\system32;C:\Windows
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;

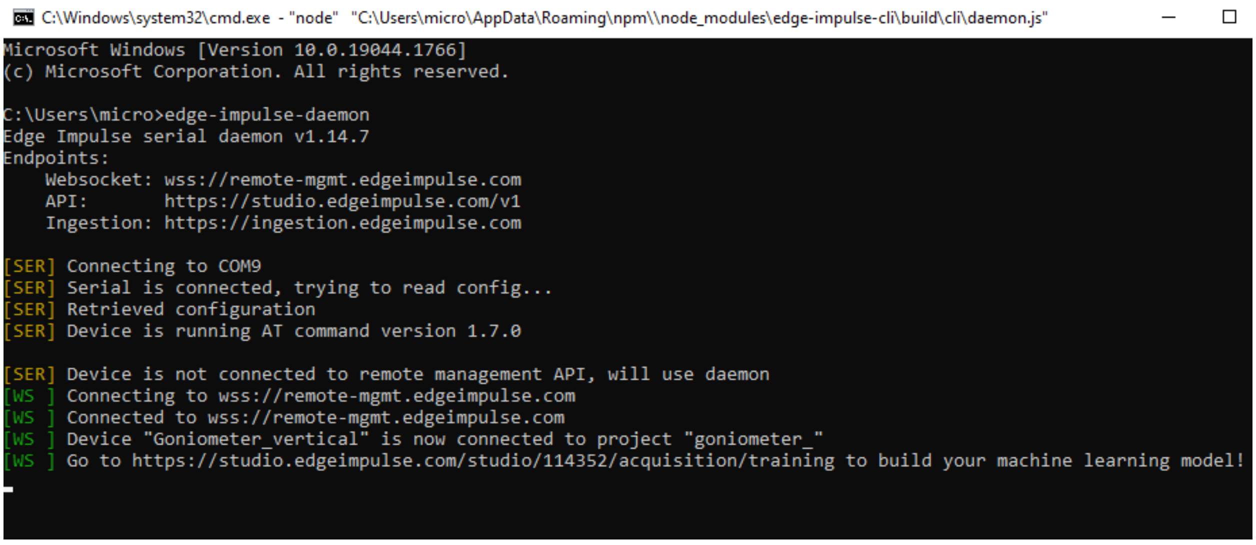
```
C:\Users\micro\AppData\Local\Microsoft\WindowsApps
C:\Users\micro\AppData\Local\atom\bin
C:\Users\micro\AppData\Roaming\npm
C:\Users\micro\Documents\arduino-cli_0.21.0_Windows_64bit
C:\Program Files\OpenVPN\bin
```

Picture 3.4 :Environmental Variables Setup

3.4.1.2 Connect to Edge Impulse Platform

- With the use of a micro-USB cable, we connect the Arduino Board to our system. We then press the RESET button on the board to launch the bootloader.
- After a short firmware update of the board, we run the following command to a terminal/ command prompt (Picture 3.5):

```
$ edge-impulse-daemon
```



```
C:\Windows\system32\cmd.exe - "node"  "C:\Users\micro\AppData\Roaming\npm\\node_modules\edge-impulse-cli\build\cli\daemon.js"
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\micro>edge-impulse-daemon
Edge Impulse serial daemon v1.14.7
Endpoints:
    Websocket: wss://remote-mgmt.edgeimpulse.com
    API:       https://studio.edgeimpulse.com/v1
    Ingestion: https://ingestion.edgeimpulse.com

[SER] Connecting to COM9
[SER] Serial is connected, trying to read config...
[SER] Retrieved configuration
[SER] Device is running AT command version 1.7.0

[SER] Device is not connected to remote management API, will use daemon
[WS ] Connecting to wss://remote-mgmt.edgeimpulse.com
[WS ] Connected to wss://remote-mgmt.edgeimpulse.com
[WS ] Device "Goniometer_vertical" is now connected to project "goniometer_"
[WS ] Go to https://studio.edgeimpulse.com/studio/114352/acquisition/training to build your machine learning model!
```

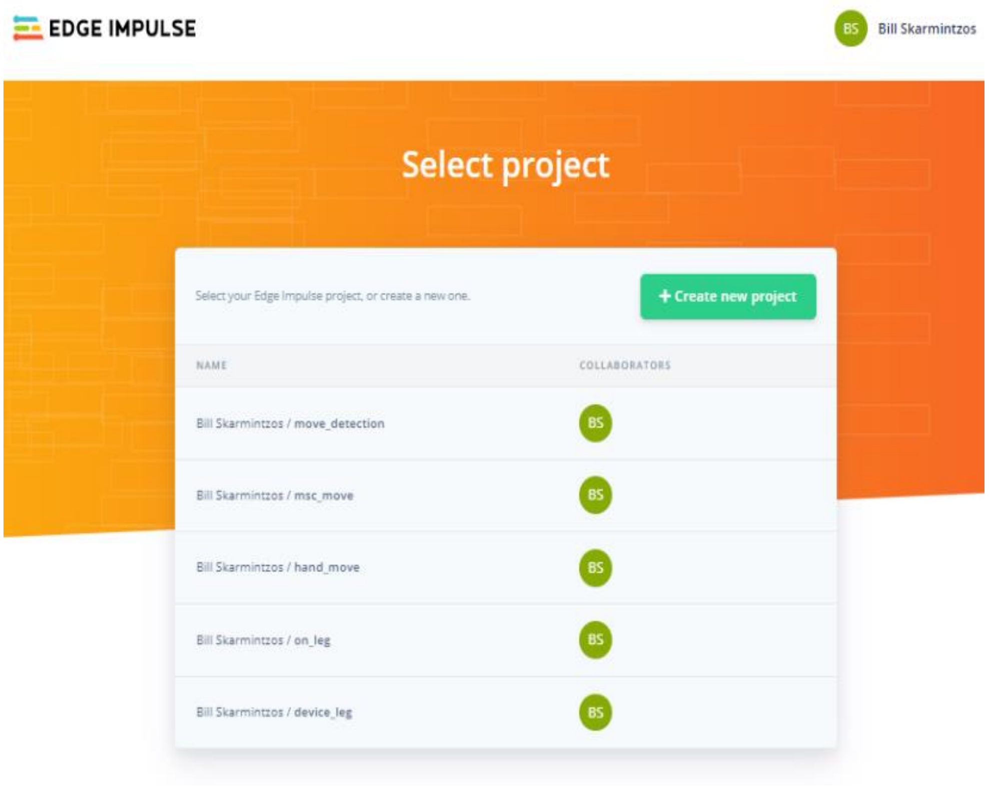
Picture 3.5: Edge Impulse Daemon

- After running the above command, we can now check if the Board is connected to the Edge Impulse Project.

3.4.2 Create a new Edge Impulse Project

In order to create a new project to the Edge Impulse Platform, a new user account is required. By visiting the URL [https://www.edgeimpulse.com/] and pressing the "GET STARTED" button, the user can create a new account and start building projects.

After creating the project, we head to the main page of the project (Picture 3.6)



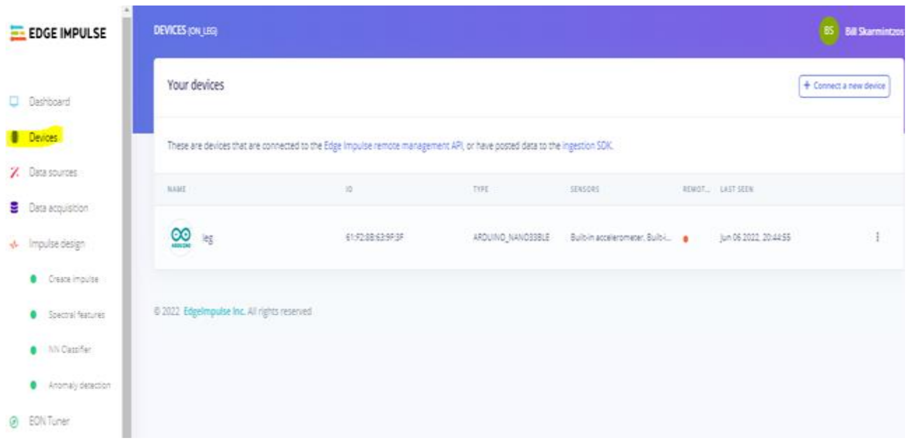
Select your Edge Impulse project, or create a new one.

NAME	COLLABORATORS
Bill Skarmintzos / move_detection	BS
Bill Skarmintzos / msc_move	BS
Bill Skarmintzos / hand_move	BS
Bill Skarmintzos / on_leg	BS
Bill Skarmintzos / device_leg	BS

Picture 3.6: Edge Impulse New Project

3.4.3 Check Device Connectivity

On the left pane of the main page, we can find the steps that need to be followed in order to create the model. The choice **“Devices”,** gives us the chance to check if the Arduino board is connected to our project, in order to be able to collect data. (Picture 3.7)



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
leg	61:F2:88:63:9F:3F	ARDUINO_NANO33BLE	Built-in accelerometer, Built-i...	●	Jun 06 2022, 20:44:55

© 2022 EdgeImpulse Inc. All rights reserved

Picture 3.7: Edge Impulse Device Connectivity Check

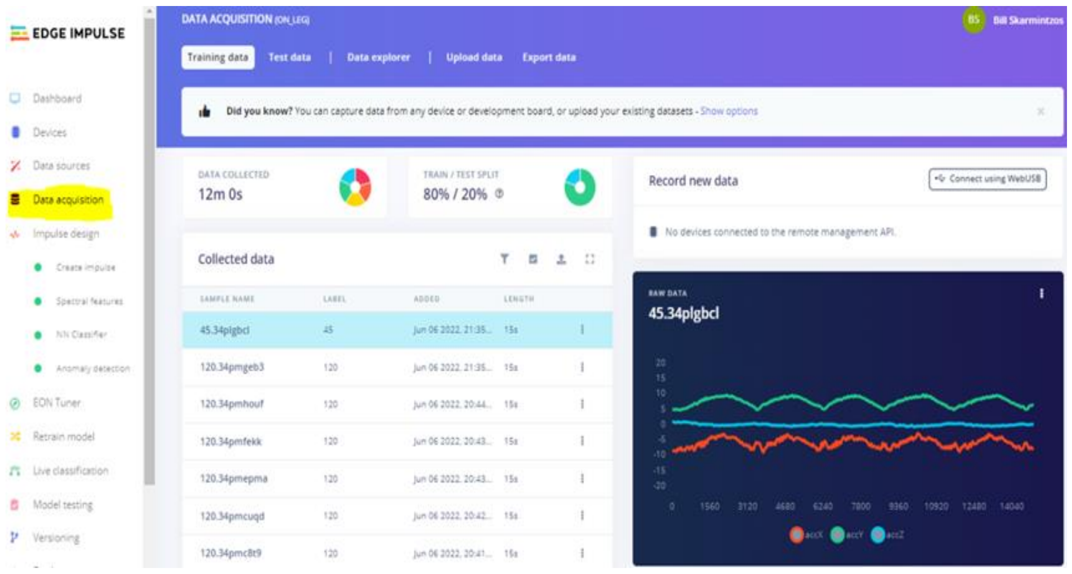
3.4.4 Data Acqisition- Training & Test Set

The choice **“Data Acquisition”** is linked with the collection of data from the board. In our experiment we use this function of Edge Impulse in order to record and properly categorize the results of the moving leg.In our experiment we categorize the data in five categories (20, 45, 60, 90, 120 degrees). In addition, we use some of these records to create a test set (so as to check the accuracy of the model that will be created)

In each of the three installations explained in chapter 3.3, we start moving -either the goniometer or the RangeOfMotion Knee brace, depending on the installation- in the range of the angle degrees sampled.

For each of our three experiments, we have sampled 10 extension-flexion moves on each of the 5 angles of interest (5 X 10 = 50 samples in total). The duaration of each sample is 15sec. We used 2 of the samples gathered to fill the “Test Set”.

The following image (Picture 3.8) shows the page of the TRAINING DATA, where we find a list of the Collected Data. We can also see a graphical indication of the records of the accelerometes.



Did you know? You can capture data from any device or development board, or upload your existing datasets - Show options

DATA COLLECTED
12m 0s

TRAIN / TEST SPLIT
80% / 20%

No devices connected to the remote management API.

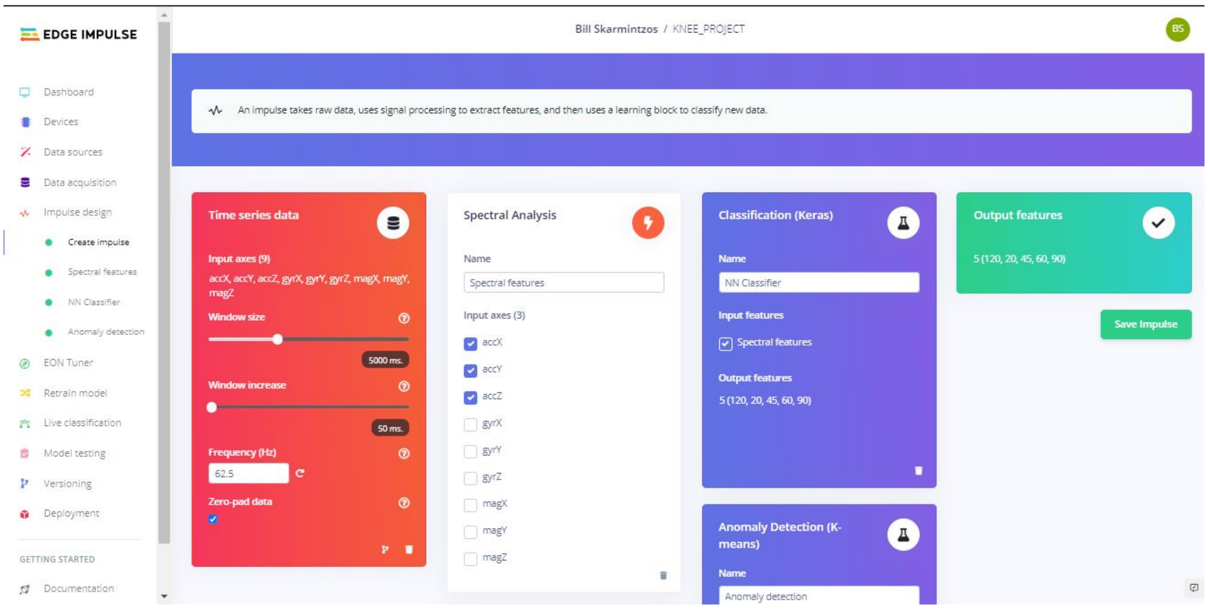
Collected data

SAMPLE NAME	LABEL	ADDED	LENGTH
45.34plgbcl	45	Jun 06 2022, 21:35...	15s
120.34pmgeb3	120	Jun 06 2022, 21:35...	15s
120.34pmhouf	120	Jun 06 2022, 20:44...	15s
120.34pmfekk	120	Jun 06 2022, 20:43...	15s
120.34pmepma	120	Jun 06 2022, 20:43...	15s
120.34pmcuqd	120	Jun 06 2022, 20:42...	15s
120.34pmc8t9	120	Jun 06 2022, 20:41...	15s

Picture 3.8: Edge Impulse Data Acquisition

3.4.5 Impulse Design

After recording and collecting our data, it is time to create a Machine Learning Model. As a first step, Edge Impulse gives the chance to create this with easy steps, through choosing a series of blocks. The image below (Picture 3.9) shows the "Create Impulse" choice of the left-side menu. In this page we can distinguish 3 major blocks.



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 (9)
accX, accY, accZ, gyrX, gyrY, gyrZ, magX, magY, magZ
Window size: 5000 ms.
Window increase: 50 ms.
Frequency (Hz): 62.5
Zero-pad data: ✓

Spectral Analysis
Name: Spectral features
Input axes (3)
✓ accX
✓ accY
✓ accZ
gyrX
gyrY
gyrZ
magX
magY
magZ

Classification (Keras)
Name: NN Classifier
Input features
✓ Spectral features
Output features
5 (120, 20, 45, 60, 90)

Anomaly Detection (K-means)
Name: Anomaly detection

Output features
5 (120, 20, 45, 60, 90)

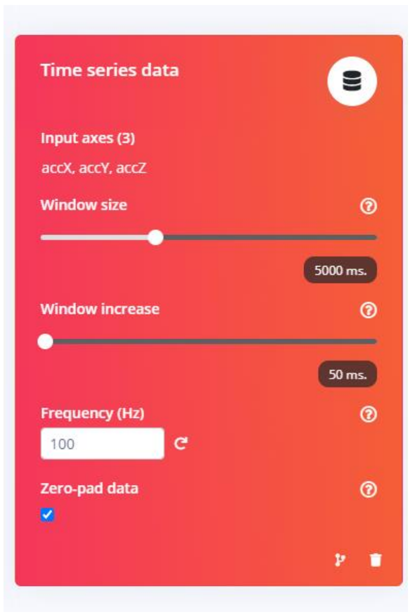
Picture 3.9:Edge Impulse_ Impulse Design

3.4.5.1 Time Series Data

The first block, is the "Time Series data".

This block (Picture 3.10) indicates the type of input data.

- **"The input axes"** field lists all the axis referenced from our training dataset .
- The **window size** is the size of the raw features that is used for the training.
- **The window increase** is used to artificially create more features (and feed the learning block with more information)
- **The frequency** is automatically calculated based on our training samples. You can modify this value but you currently cannot use values lower than 0.000016 (less than 1 sample every 60s).
- **Zero-pad data**: Adds zero values when raw feature is missing
  In every model that we trained, Time Series Data parameters that were used are:
    - Window Size :5000ms
    - Window Increase: 50ms


Time series data

Input axes (3)
accX, accY, accZ

Window size
5000 ms.

Window increase
50 ms.

Frequency (Hz)
100

Zero-pad data
✓

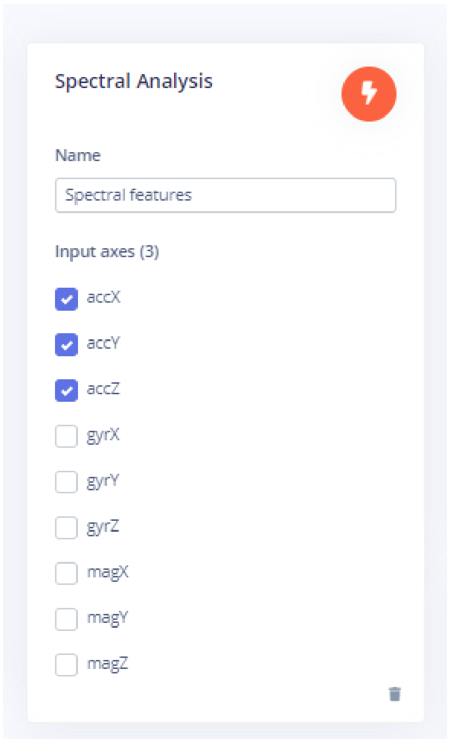
Picture 3.10:Edge Impulse Create Impulse

3.4.5.2 Spectral Analysis Processing Block

A processing block is basically a feature extractor. It consists of DSP (Digital Signal Processing) operations that are used to extract features that our model learns on. These operations vary, depending on the type of data used in our project. **Spectral Analysis (**Picture 3.11**)** is our choice for the experiment, as it is a good solution for analyzing repetitive motion, such as data from accelerometers (data from accelerometers, magnitometers and gyroscopes).

In all our models, in spectral analysis block, we used 3 input Axes (from the given):

accX, accY, accZ


Spectral Analysis

Name

Spectral features

Input axes (3)

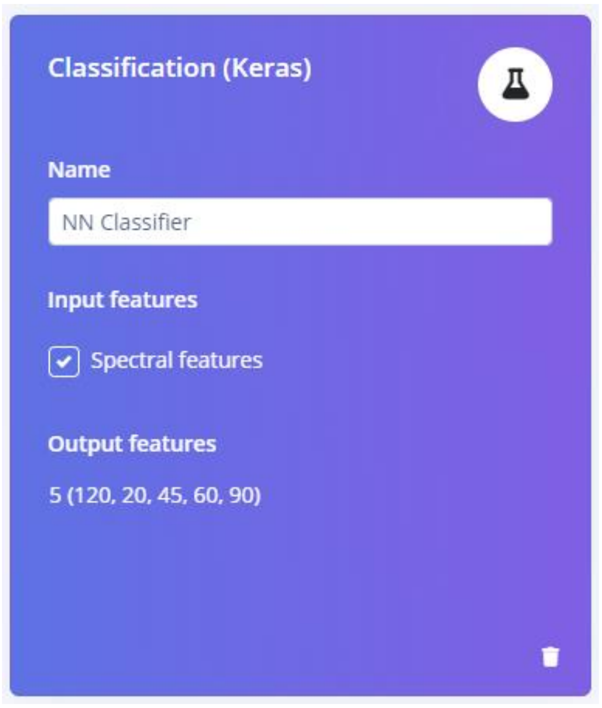
- [x] accX
- [x] accY
- [x] accZ
- [ ] gyrX
- [ ] gyrY
- [ ] gyrZ
- [ ] magX
- [ ] magY
- [ ] magZ

Picture 3.11:Edge Impulse Spectral Analysis Block

3.4.5.3 Classification Block and Anomaly Detection

After adding the processing block, we move to the learning blocks. Learning blocks vary depending on what our model wants to do. It is simply a neural network that it is trained to learn on our data.

- **NN Classifier block** :The basic idea of NN Classifier (Picture 3.12)is that a neural network classifier will take some input data, and output a probability score that indicates how likely it is that the input data belongs to a particular class. The classes we have here are the degrees of the angles of the experiment. **The NN classifier input features are those from the Spectra**l features block and the **Output** features are the 5 labels examined.


Classification (Keras)

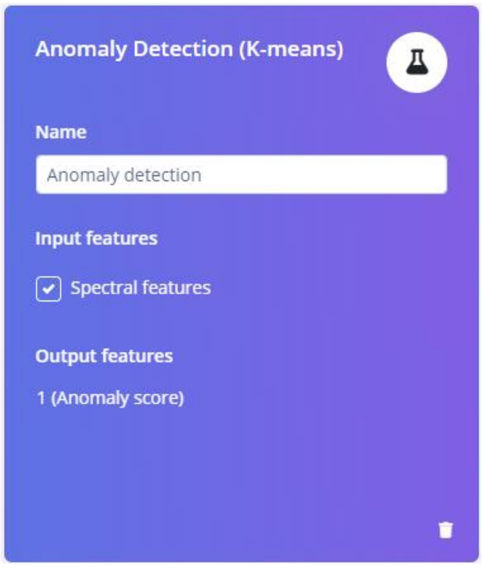
Name
NN Classifier

Input features
✓ Spectral features

Output features
5 (120, 20, 45, 60, 90)

Picture 3.12: NN Classifier block

- **Anomaly Detection block.** The idea behind this block (Picture 3.13) is that neural network do not work flawlessly. In many cases they cannot judge where a result belongs and if they spot something they have never seen before, they will try to classify it to one of the known classes.
  We leave the parameters of this block as it is.


Anomaly Detection (K-means)

Name
Anomaly detection

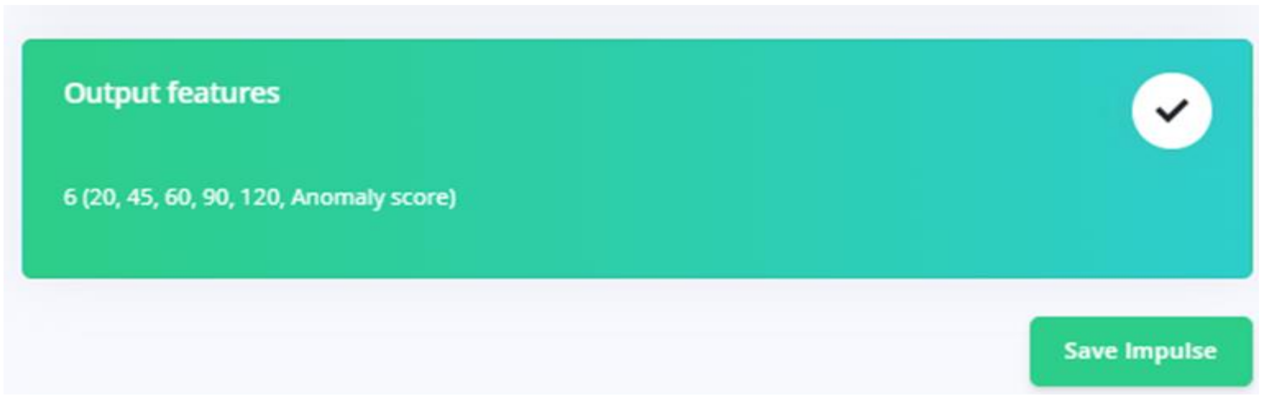
Input features
✓ Spectral features

Output features
1 (Anomaly score)

Picture 3.13 :Edge Impulse Anomaly Detection block

3.4.6 Complete Impulse

After these blocks, we are ready to save the impulse and continue with more parameters on the blocks.


Output features

6 (20, 45, 60, 90, 120, Anomaly score)

Picture 3.14:Edge Impulse Save Impulse

3.5 Explanation of the Blocks

On the left pane of Edge Impulse platform, after completing the Edge Impulse design, new tabs appear – Spectral Features, NN classifier and Anomaly detection.

In the paragraphs below, we attempt to explain in more detail the Processing and Learning blocks used in the creation of the impulse (machine learning model).

3.5.1 Spectral Features (Processing Block)

The Spectral features block (Picture 3.15)extracts frequency and power characteristics of a signal. Low-pass and high-pass filters can also be applied to filter out unwanted frequencies. It has a good use in analyzing repetitive patterns in a signal, such as movements or vibrations from an accelerometer.

The parameters that can be adjusted are:

Scaling

- Scale axes: Multiplies axes by this number to scale data from the sensor

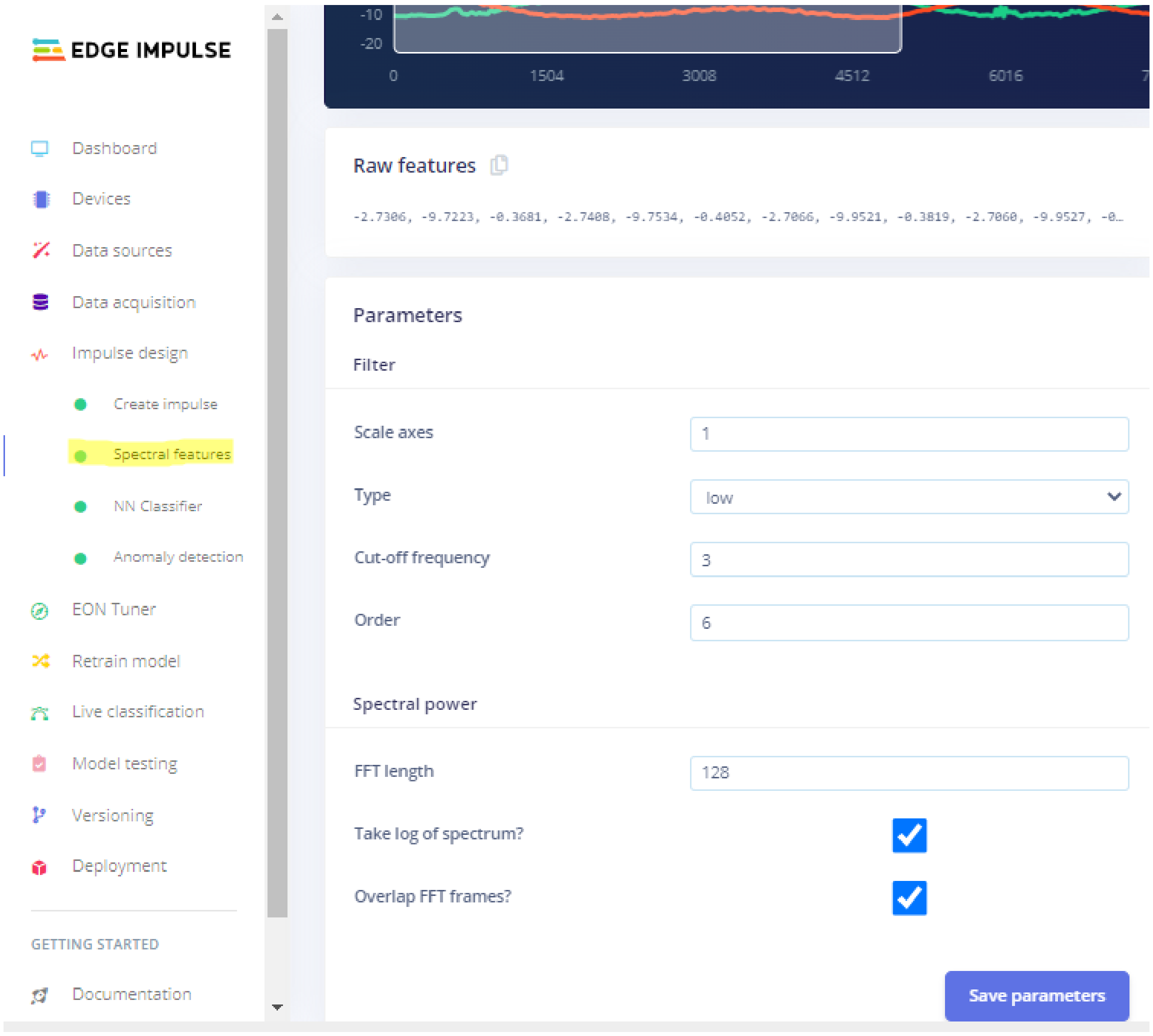
Filter

- Type: Type of filter to apply to the raw data (low-pass, high-pass, or none)
- Cut-off frequency: Cut-off frequency of the filter in hertz
- Order: Order of the Butterworth filter

Spectral power

- FFT length: The FFT size
- No. of peaks: Number of spectral power peaks to extract
- Peaks threshold: Minimum threshold to extract a peak (frequency domain normalized to [0, 1])

Power edges: Splits the power spectral density in various buckets (V**2/Hz unit)



Raw features

```
-2.7306, -9.7223, -0.3681, -2.7408, -9.7534, -0.4052, -2.7066, -9.9521, -0.3819, -2.7060, -9.9527, -0…
```

Parameters

Filter

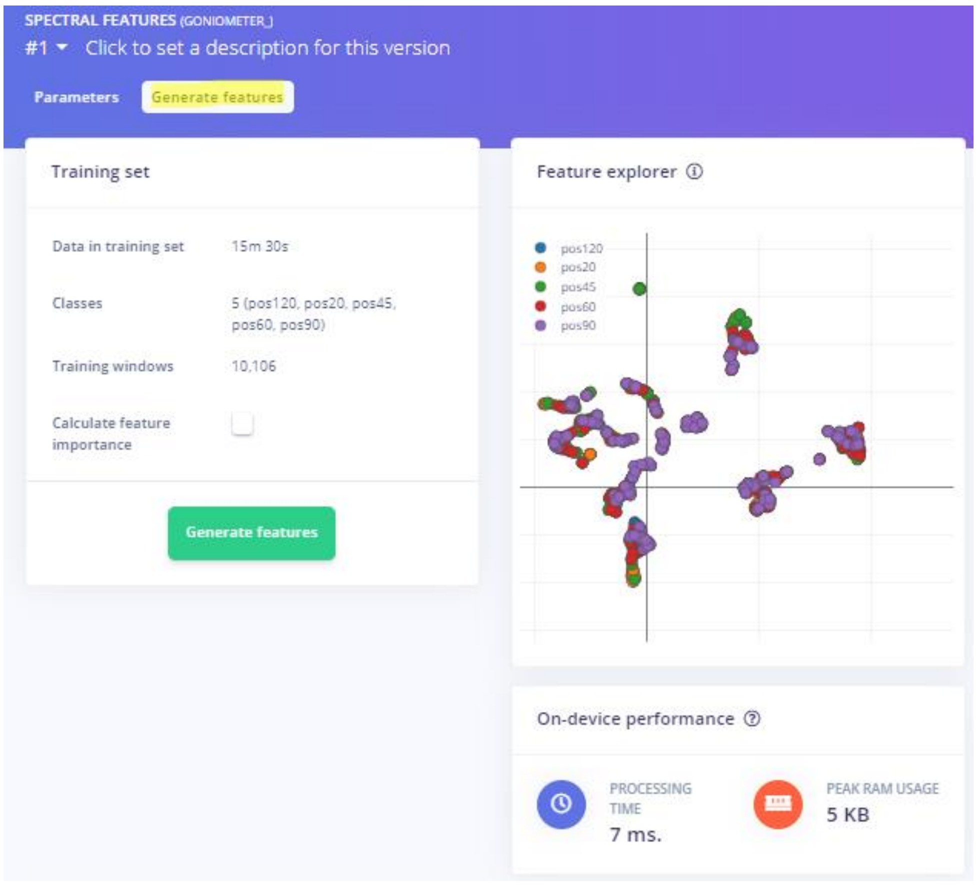
Scale axes	1
Type	low
Cut-off frequency	3
Order	6

Spectral power

FFT length	128
Take log of spectrum?	✓
Overlap FFT frames?	✓

Picture 3.15: Edge Impulse Spectral Features

We leave the parameters of this block unchanged. We press the "Save Parameters" button and we move to Generate Features (Picture 3.16) page to complete the feature generation job.


SPECTRAL FEATURES (GONIOMETER_)
#1 Click to set a description for this version

Training set

Data in training set	15m 30s
Classes	5 (pos120, pos20, pos45, pos60, pos90)
Training windows	10,106
Calculate feature importance	

On-device performance

PROCESSING TIME 7 ms.

PEAK RAM USAGE 5 KB

Picture 3.16: Edge Impulse Spectral Features Generate Features

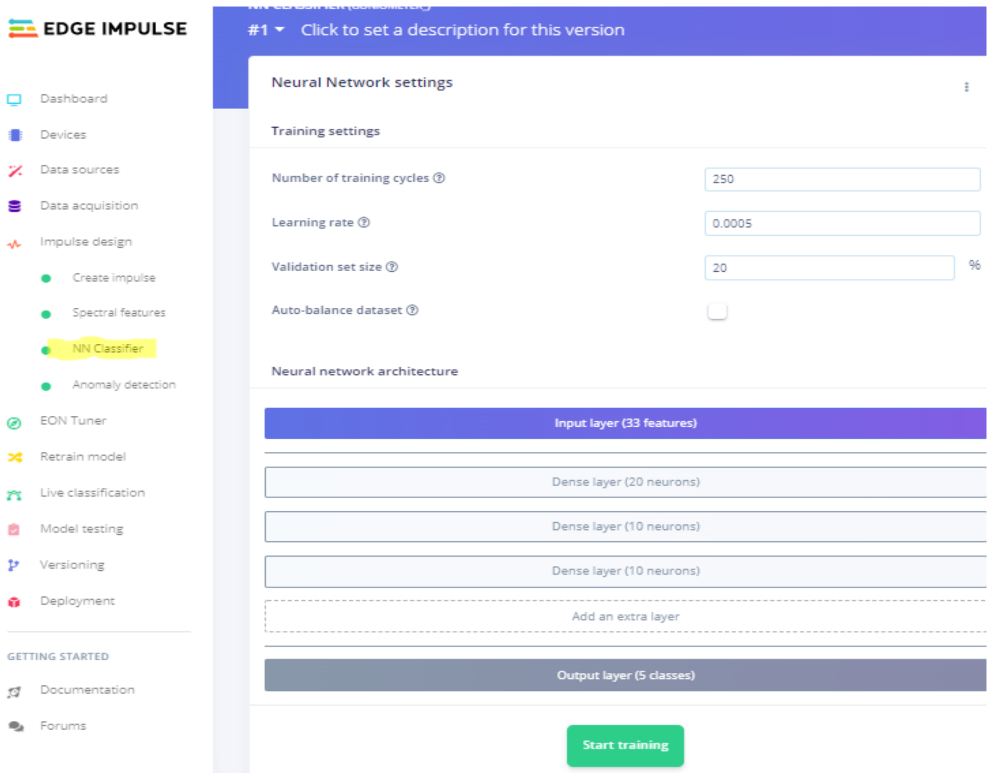
3.5.2 NN Classifier - Classification

The next step is to use the Learning blocks. We choose the NN classifier block. This block (Picture 3.17) is based on the theory of Neural Networks. The neural network classifier will take some input data, and output a probability score that indicates how likely it is that the input data belongs to a particular class.

Adding this block, creates a neural network with the parameters that we as users choose. With our dataset collected and features processed, we can train our machine learning model. The neural network is then given a set of training data- the set of examples that it is supposed to predict. The network's output is compared to the correct answer and, based on the results, the weights of the connections between the neurons in the layer are adjusted.

The architecture of our Neural Network can be arranged through this block. By selecting the NN Classifier on the left pane, we can adjust many parameters of the learning block, such as number of training cycles and also learning rate.

The Training procedure can begin by pressing the “Start Training” button



Neural Network settings

Training settings

Number of training cycles: 250

Learning rate: 0.0005

Validation set size: 20 %

Auto-balance dataset: ☐

Neural network architecture

Input layer (33 features)

Dense layer (20 neurons)

Dense layer (10 neurons)

Dense layer (10 neurons)

Add an extra layer

Output layer (5 classes)

Picture 3.17: Edge Impulse Learning Block Classification

3.5.3 Anomaly Detection (K-means)

The Anomaly Detection (K-Means) block (Picture 3.18) is a method that helps recognize in which of our classes, an input may belong. Because some inputs are not easily recognizable by the neural network, this block will include them in the closest related class.

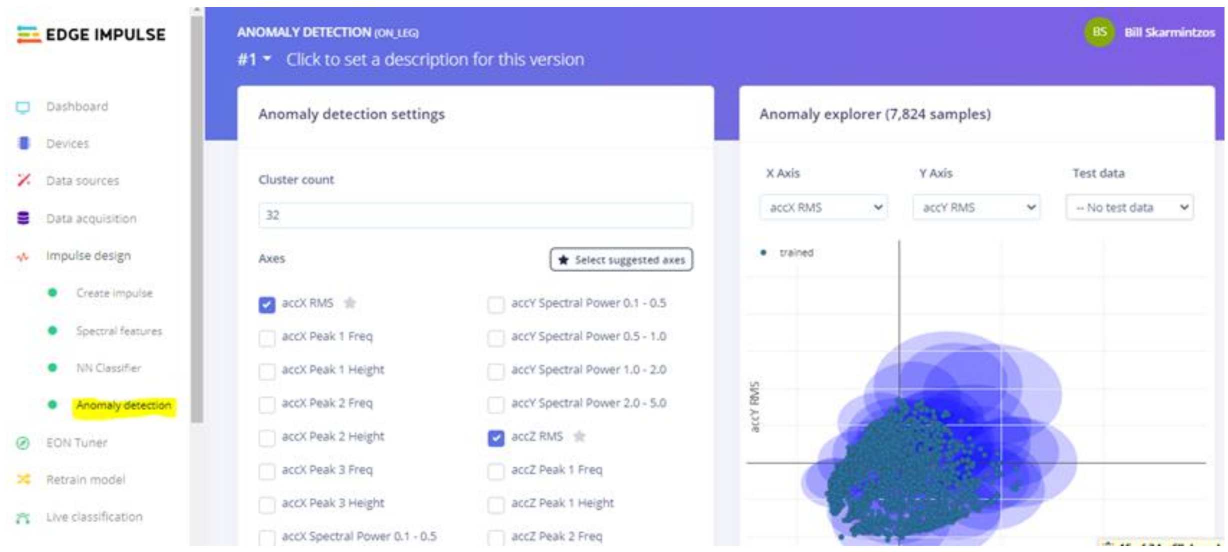
This procedure is based on K-means clustering. A cluster refers to a collection of data that are aggregated together, because they have certain similarities. K is the number of the centroids we will use to group our data.

The algorithm, at first, randomly selects centroids and are the starting points of every cluster. A threshold value can be added to detect anomalies: if the distance between a data point and its nearest centroid is greater than the threshold value, then it is an anomaly.

In our anomaly detection block, we can Select the suggested axes button to harness the value of the feature importance output.

In our anomaly detection block, we can click on the **Select suggested axes** button to harness the value of the feature importance output.

By pressing the “Start Training” button, the training of the Neural Network for this block starts.



ANOMALY DETECTION (ON_LEG)

#1 Click to set a description for this version

Anomaly detection settings

Cluster count: 32

Axes — ★ Select suggested axes

- [✓] accX RMS
- [ ] accX Peak 1 Freq
- [ ] accX Peak 1 Height
- [ ] accX Peak 2 Freq
- [ ] accX Peak 2 Height
- [ ] accX Peak 3 Freq
- [ ] accX Peak 3 Height
- [ ] accX Spectral Power 0.1 - 0.5
- [ ] accY Spectral Power 0.1 - 0.5
- [ ] accY Spectral Power 0.5 - 1.0
- [ ] accY Spectral Power 1.0 - 2.0
- [ ] accY Spectral Power 2.0 - 5.0
- [✓] accZ RMS
- [ ] accZ Peak 1 Freq
- [ ] accZ Peak 1 Height
- [ ] accZ Peak 2 Freq

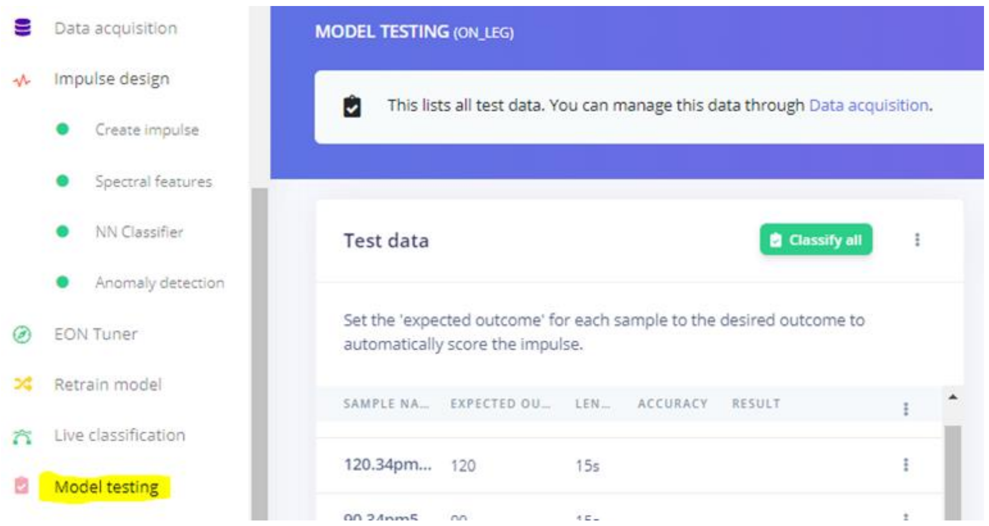
Anomaly explorer (7,824 samples)

X Axis: accX RMS — Y Axis: accY RMS — Test data: -- No test data

Picture 3.18: Edge Impulse Anomaly Detection Block

3.5.4 Model testing

After having trained the above model, we can now test it (Picture 3.19)! By selecting the **“Model testing”** on the left pane, we can use the Test data that we collected on the Data Acquisition tab to test our model and check if it can successfully classify the data.



This lists all test data. You can manage this data through Data acquisition.

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

SAMPLE NA...	EXPECTED OU...	LEN...	ACCURACY	RESULT
120.34pm...	120	15s		

Picture 3.19 Edge Impulse Model Testing

3.5.5 Deploy the model

Now that we have created and tested our model, we can make our model run on device. That is especially important as we have explained. After this step, the model built will run on any given platform supported by Edge impulse (such as Arduino), giving the necessary results without the need of any connection with a computer or other system

In this step, a package will be created, including all the processing, configuration and learning blocks. Edge Impulse platform gives a variety of options to deploy in our board. So, we can deploy our model as a customized library (like an Arduino library), deploy as a pre-built firmware, fully supported by development boards such as Arduino Nano 33 and two more (directly on computer or phone and Linux targets).

To run as an Arduino Library, we only have to choose the corresponding choice in the Platform. A zip file will be downloaded including the Arduino library ready to be deployed on our device.

Edge Impulse gives another option as it can generate a pre-built firmware. In that way we have a ready to go binary for our board. Edge impulse includes a wide variety of development boards that can receive a binary created by the platform. For our case we have Arduino Nano 33 BLE board and a proper binary for this board can be easily built and deployed to our board. Next step is to Flash our binary to the device. Giving one simple command ($ edge-impulse-run-impulse). In the command prompt, the model starts to run on-device and in that way the impulse runner shows the results of the running model.

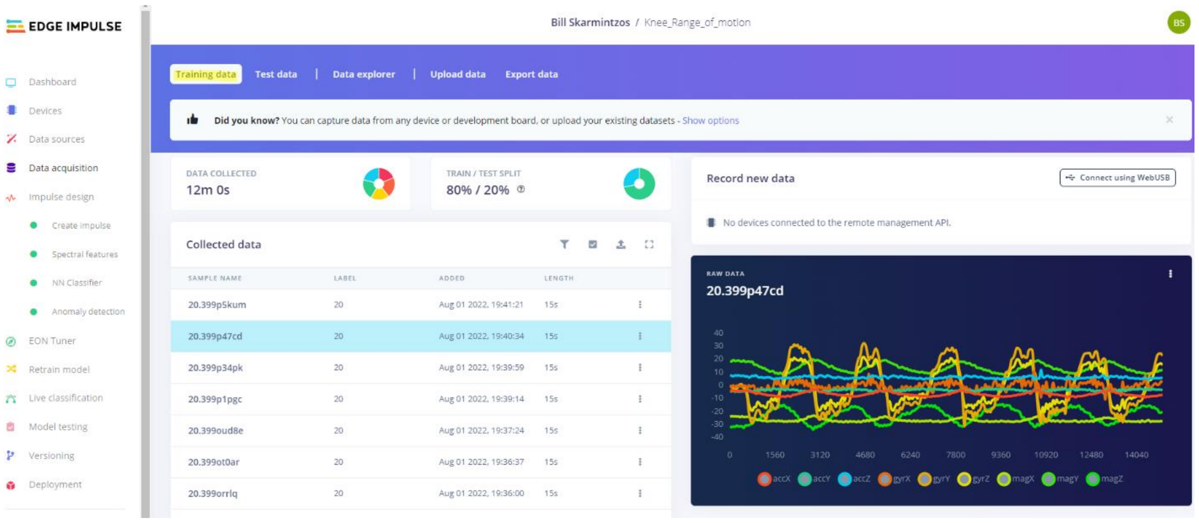
3.6 Presentation of the models

3.6.1 General

In this chapter we are going to present details about the 3 models we created.

As we mentioned, we installed Arduino Nano 33 BLE in three different ways. The first case was the installation of Arduino on the one edge of a goniometer. In the second case, the goniometer with the Arduino attached, was placed in the area of the knee. In the third case we used a RangeOfMotion-Knee-Brace and we placed the Arduino board in the area of the ankle.

For each installation, we followed the steps that were explained in the previous chapter. So, firstly, we have collected a Training and a Test Set. For each case we have collected several samples. In each case the total duration of the samples reaches 10-12 mins. The **Training Data** set represents the angles 20, 45, 60, 90 and 120 ^{o}. An image of how the samples is presented in the platform of Edge Impulse are shown below (Picture 3.20).



Did you know? You can capture data from any device or development board, or upload your existing datasets - Show options

DATA COLLECTED
12m 0s

TRAIN / TEST SPLIT
80% / 20%

Record new data

No devices connected to the remote management API.

Collected data

SAMPLE NAME	LABEL	ADDED	LENGTH
20.399p5kum	20	Aug 01 2022, 19:41:21	15s
20.399p47cd	20	Aug 01 2022, 19:40:34	15s
20.399p34pk	20	Aug 01 2022, 19:39:59	15s
20.399p1pgc	20	Aug 01 2022, 19:39:14	15s
20.399oud8e	20	Aug 01 2022, 19:37:24	15s
20.399ot0ar	20	Aug 01 2022, 19:36:37	15s
20.399orrlq	20	Aug 01 2022, 19:36:00	15s

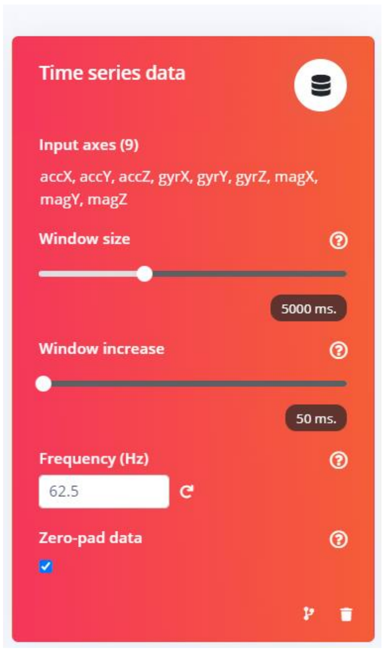
RAW DATA
20.399p47cd

Picture 3.20 : Samples of a Training Set

The **Test Set** consists of collected data that we have captured and we will use them to verify the model we have created. It is recommended, an approximate 80/20 train/test split ratio for the data of every class (or label) in our dataset, although especially large datasets may require fewer testing data.

On the **Impulse Design/ Create Impulse tab**, where we build our model as a pipeline, we have to select the proper parameters in order to have an as better trained model as possible. The parameters chosen are shown below:

- For the **Time Series block**, we use the same in all three cases. Window size is 5000ms and Window increase 50ms.



Time series data

Input axes (9)

accX, accY, accZ, gyrX, gyrY, gyrZ, magX, magY, magZ

Window size: 5000 ms.

Window increase: 50 ms.

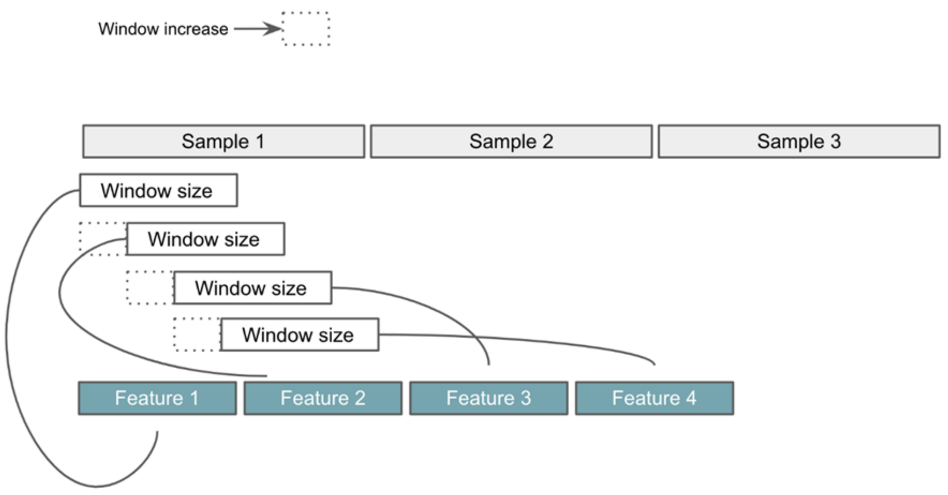
Frequency (Hz): 62.5

Zero-pad data: ✓

The window size is the size of the raw features that is used for the training. In all of our installations we used a window size of 5000 MS (5 sec)

The window increase is used to artificially create more features (and feed the learning block with more information)

The image below (Picture 3.21)summarizes the role of each parameter.

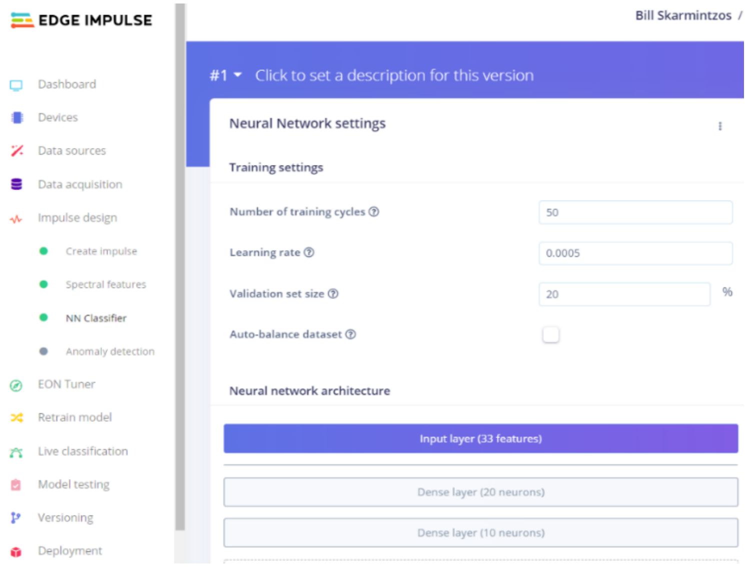


Picture 3.21: Window size and Window increase

- The next parameters that we **interfere with are in the NN_Classifier block**. The following table shows the parameters used in each case

CASE / Parameter	**Case1_Goniometer**	**Case 2_Goniometer on Leg**	**Case 3_Knee Brace-ROM**
Number of training Cycles	50	50	19
Learning Rate	0.0005	0.0005	0.0005

The pictures (Pictures 3.22-3.23-3.24) below also show these parameters, being set on Edge Impulse Platform. On the next chapters, we are going to investigate the results that Edge Impulse produces, when we run NN classifier block and also on Model Testing tab, that Edge Impulse offers. In that way we will have a first look on how good the model is trained and also have the ability to test the models ourselves with live classification results.



Neural Network settings

Training settings

Number of training cycles 50

Learning rate 0.0005

Validation set size 20 %

Auto-balance dataset ☐

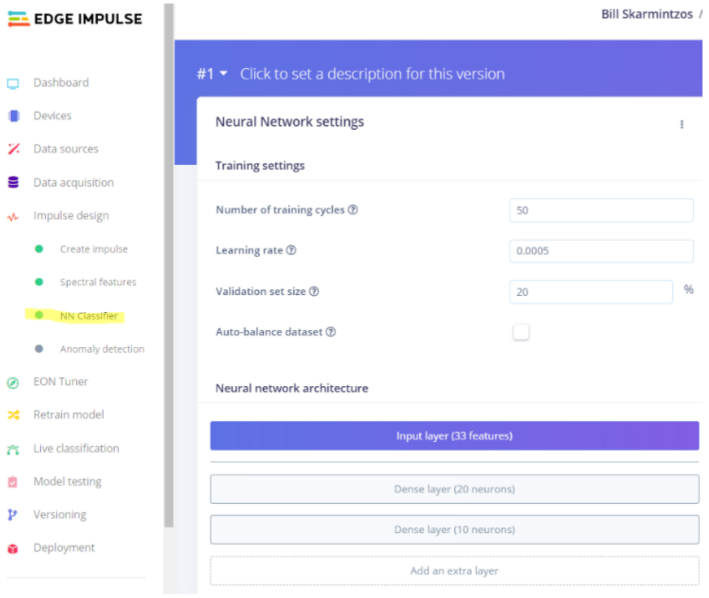
Neural network architecture

Input layer (33 features)

Dense layer (20 neurons)

Dense layer (10 neurons)

Picture 3.23: Goniometer NN parameters



Neural Network settings

Training settings

Number of training cycles 50

Learning rate 0.0005

Validation set size 20 %

Auto-balance dataset ☐

Neural network architecture

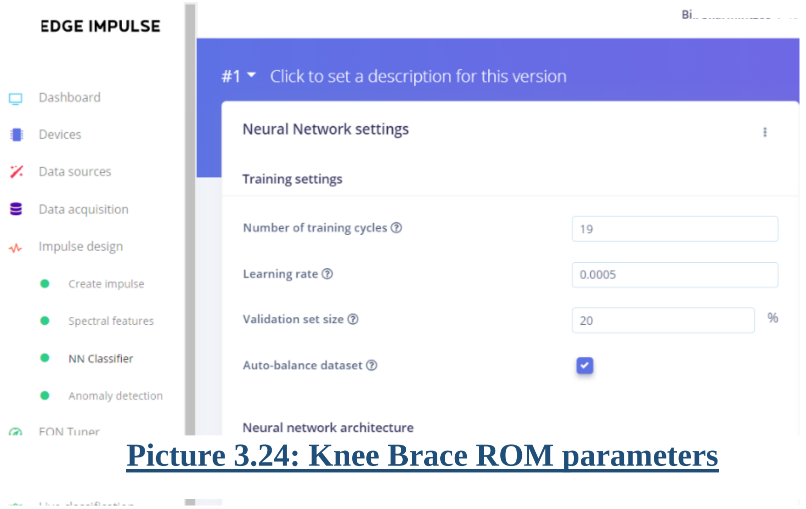
Input layer (33 features)

Dense layer (20 neurons)

Dense layer (10 neurons)

Add an extra layer

Picture 3.22: Goniometer on leg NN Parameters



Neural Network settings

Training settings

Number of training cycles 19

Learning rate 0.0005

Validation set size 20 %

Auto-balance dataset ✓

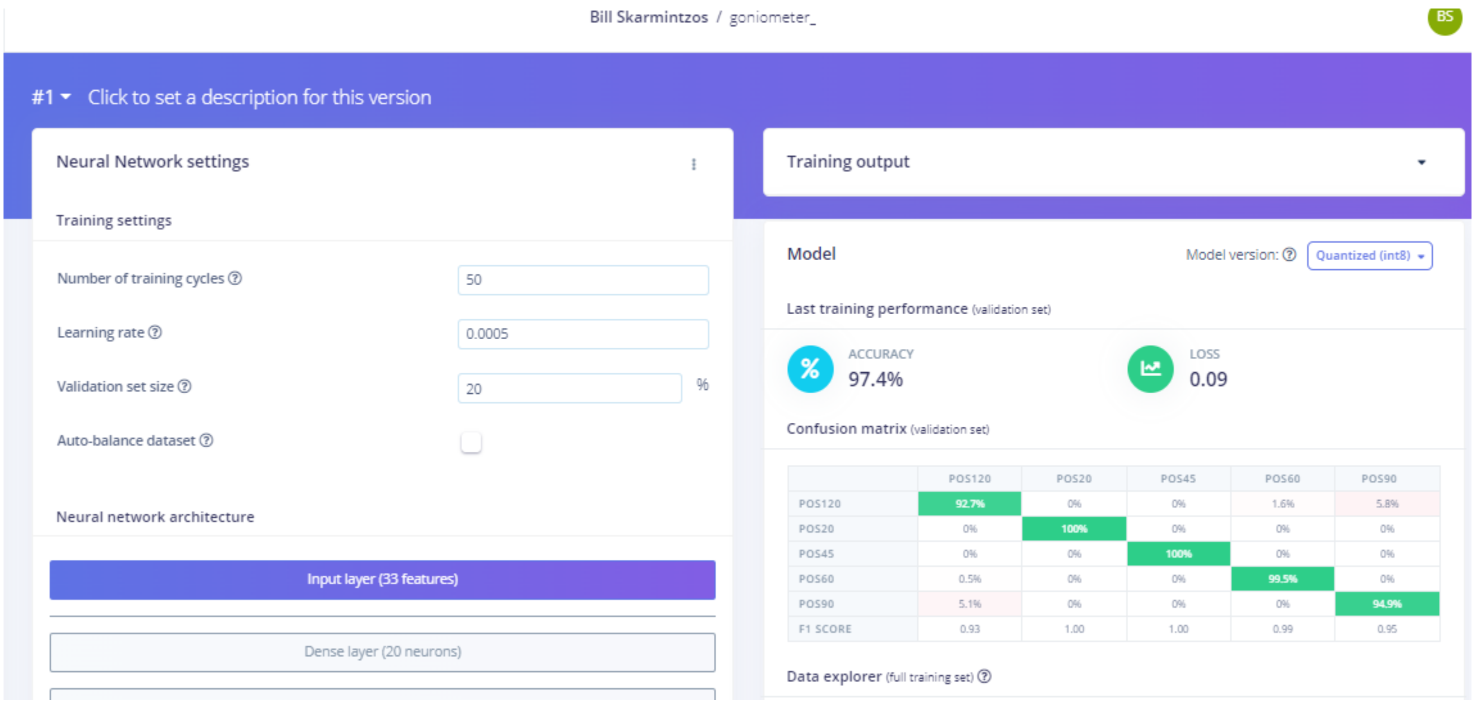
Neural network architecture

Picture 3.24: Knee Brace ROM parameters

3.6.2 Arduino on Goniometer- Results

On the first installation, we used a goniometer and the board was adjusted on the edge of the goniometer. The goniometer was placed on a surface (writing desk). While holding the one brace of the goniometer steadily, we started moving the other brace (with the Arduino board attached) up-and-down, "writing" in that way the desired angle. We tried to simulate the extension and flexion of a human knee, in the angles of interest.

The results of the NN Classifier are shown in the image below (Picture 3.25). This is a first indication of how accurately the model is working and is a special feature that Edge Impulse offers.


Bill Skarmintzos / goniometer_
BS
#1 Click to set a description for this version
Training output
Model version: Quantized (int8)

Neural Network settings

Training settings

Number of training cycles: 50

Learning rate: 0.0005

Validation set size: 20 %

Auto-balance dataset: ☐

Neural network architecture

Input layer (33 features)

Dense layer (20 neurons)

Model

Last training performance (validation set)

ACCURACY 97.4%

LOSS 0.09

Confusion matrix (validation set)

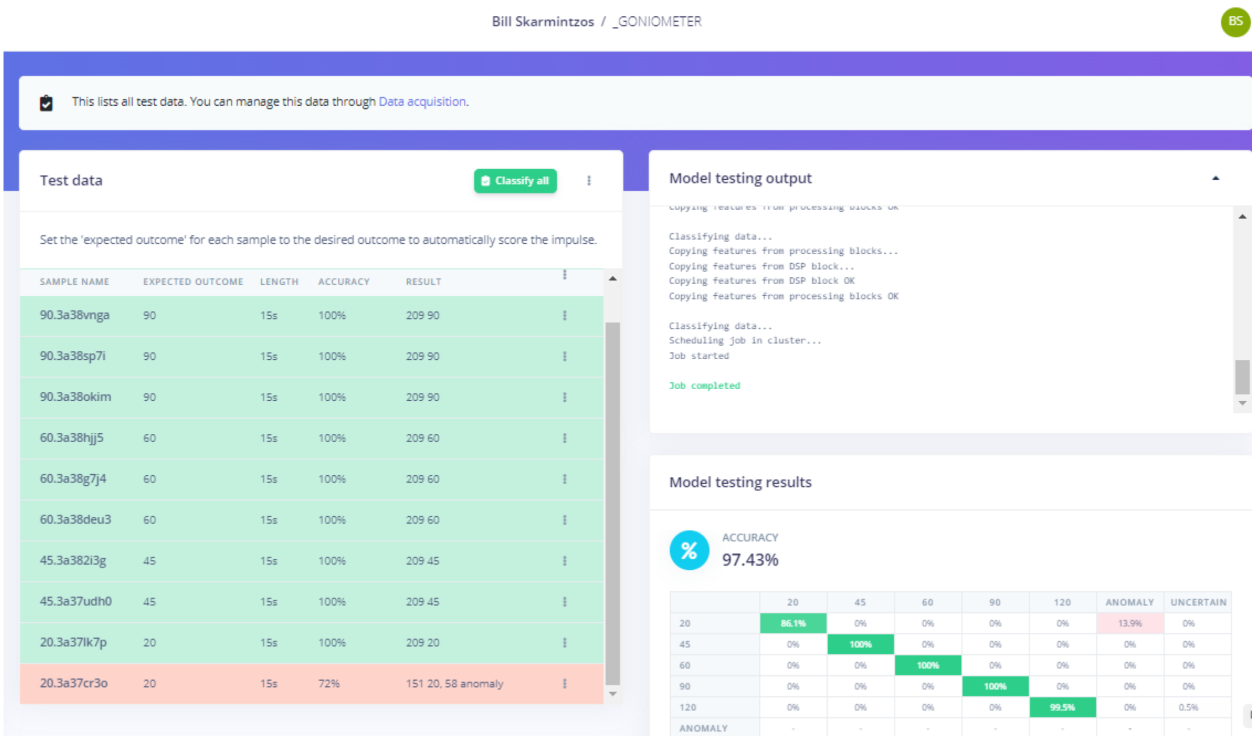
	POS120	POS20	POS45	POS60	POS90
POS120	92.7%	0%	0%	1.6%	5.8%
POS20	0%	100%	0%	0%	0%
POS45	0%	0%	100%	0%	0%
POS60	0.5%	0%	0%	99.5%	0%
POS90	5.1%	0%	0%	0%	94.9%
F1 SCORE	0.93	1.00	1.00	0.99	0.95

Data explorer (full training set)

Picture 3.25: Goniometer _first results

In order to have a well-trained model, we ran 50 training cycles. As it is apparent, the accuracy of this model is very good (accuracy: 97.4%). The F1 score is close to 1.00 except the case of 90 and 120 degrees. As it is apparent in the confusion matrix, there is a small difficulty in discerning these two cases with a 5.1% and 5.8% of the predictions giving the wrong classification.

After running the Anomaly Detection block, we move to the Model Testing tab. On this tab we **can use the Test Set acquired** on the first steps, in order to realistically see how good, the Model is working. The results of this are shown below (Picture 3.26):



This lists all test data. You can manage this data through Data acquisition.

Test data

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

SAMPLE NAME	EXPECTED OUTCOME	LENGTH	ACCURACY	RESULT
90.3a38vnga	90	15s	100%	209 90
90.3a38sp7i	90	15s	100%	209 90
90.3a38okim	90	15s	100%	209 90
60.3a38hjj5	60	15s	100%	209 60
60.3a38g7j4	60	15s	100%	209 60
60.3a38deu3	60	15s	100%	209 60
45.3a382i3g	45	15s	100%	209 45
45.3a37udh0	45	15s	100%	209 45
20.3a37lk7p	20	15s	100%	209 20
20.3a37cr3o	20	15s	72%	151 20, 58 anomaly

Model testing output

```
Classifying data...
Copying features from processing blocks...
Copying features from DSP block...
Copying features from DSP block OK
Copying features from processing blocks OK

Classifying data...
Scheduling job in cluster...
Job started

Job completed
```

Model testing results

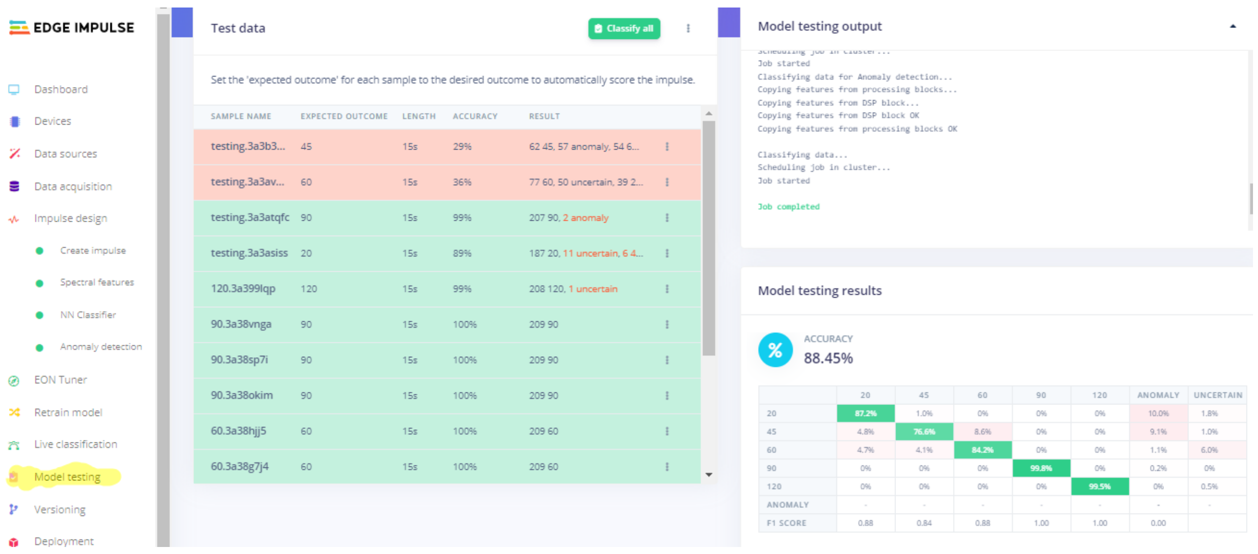
ACCURACY
97.43%

	20	45	60	90	120	ANOMALY	UNCERTAIN
20	86.1%	0%	0%	0%	0%	13.9%	0%
45	0%	100%	0%	0%	0%	0%	0%
60	0%	0%	100%	0%	0%	0%	0%
90	0%	0%	0%	100%	0%	0%	0%
120	0%	0%	0%	0%	99.5%	0%	0.5%
ANOMALY	-	-	-	-	-	-	-

Picture 3.26: Goniometer results after test set

Only by using the Test Set the model seems to work very well. It is well trained with an accuracy percentage of 97.43%. There is a small anomaly percentage on 20 degrees, meaning that the model cannot recognize some of the samples taken. Between angles that differ less than 30 degrees (45 and 60 degrees) the model seems to work very well, as it does not confuse them even though they are very close.

Edge Impulse gives us the choice to do a Live Classification of the model. That practically means that we can give samples to our model and have a prediction instantly. When giving samples that way, we can see that the model has a slightly worse behavior. We sampled some extra data (flex-extend movements) to the model. What we notice here (Picture 3.27) is that it has a higher anomaly rate while its uncertainty in almost every label is apparent (but remains low).



Test data

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

SAMPLE NAME	EXPECTED OUTCOME	LENGTH	ACCURACY	RESULT
testing.3a3b3...	45	15s	29%	62 45, 57 anomaly, 54 6...
testing.3a3av...	60	15s	36%	77 60, 50 uncertain, 39 2...
testing.3a3atqfc	90	15s	99%	207 90, 2 anomaly
testing.3a3asiss	20	15s	89%	187 20, 11 uncertain, 6 4...
120.3a399lqp	120	15s	99%	208 120, 1 uncertain
90.3a38vnga	90	15s	100%	209 90
90.3a38sp7i	90	15s	100%	209 90
90.3a38okim	90	15s	100%	209 90
60.3a38hjj5	60	15s	100%	209 60
60.3a38g7j4	60	15s	100%	209 60

Model testing output

```
Scheduling job in cluster...
Job started
Classifying data for Anomaly detection...
Copying features from processing blocks...
Copying features from DSP block...
Copying features from DSP block OK
Copying features from processing blocks OK

Classifying data...
Scheduling job in cluster...
Job started

Job completed
```

Model testing results

ACCURACY
88.45%

	20	45	60	90	120	ANOMALY	UNCERTAIN
20	87.2%	1.0%	0%	0%	0%	10.0%	1.8%
45	4.8%	76.6%	8.6%	0%	0%	9.1%	1.0%
60	4.7%	4.1%	84.2%	0%	0%	1.1%	6.0%
90	0%	0%	0%	99.8%	0%	0.2%	0%
120	0%	0%	0%	0%	99.5%	0%	0.5%
ANOMALY	-	-	-	-	-	-	-
F1 SCORE	0.88	0.84	0.88	1.00	1.00	0.00	

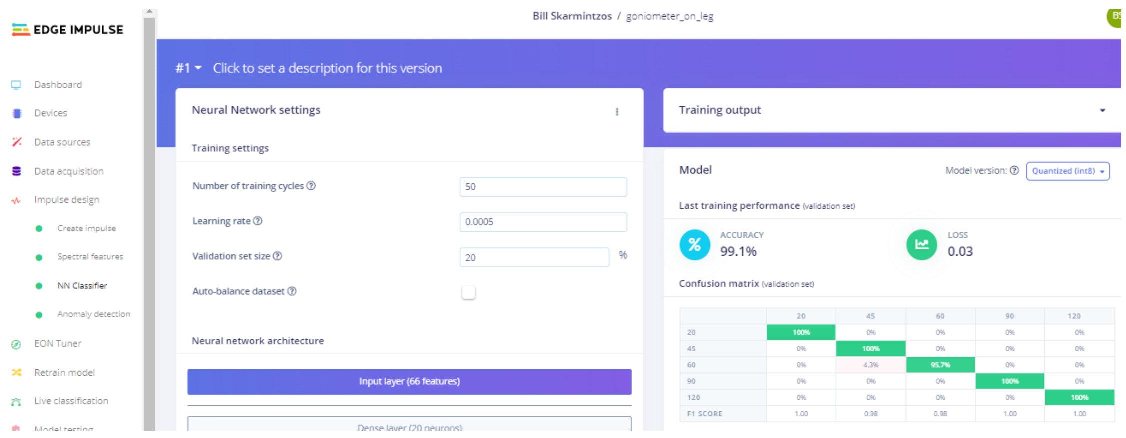
Picture 3.27: Goniometer results with Live Classification

3.6.3 Goniometer on the knee- Results

In the second case, the goniometer was placed on the knee of an adult person. The person (starting from complete extension of the leg) started moving the knee up-and-down, “writing” the desired angle, according to the indication of the goniometer. The angles that were tested here are also 20^o, 45, 60, 90 and 120 ^{o}.

The **Training and Test** sets were collected with the installation we explained.

We can see the results of the NN Classifier (Picture 3.28)that were produced after the Training process of the model. This is a first indication of how good the model is working and is a special feature that Edge Impulse offers.



Neural Network settings

Training settings

Number of training cycles	50
Learning rate	0.0005
Validation set size	20 %
Auto-balance dataset	☐

Neural network architecture

Input layer (66 features)

Dense layer (20 neurons)

Training output

Model

Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY
99.1%

LOSS
0.03

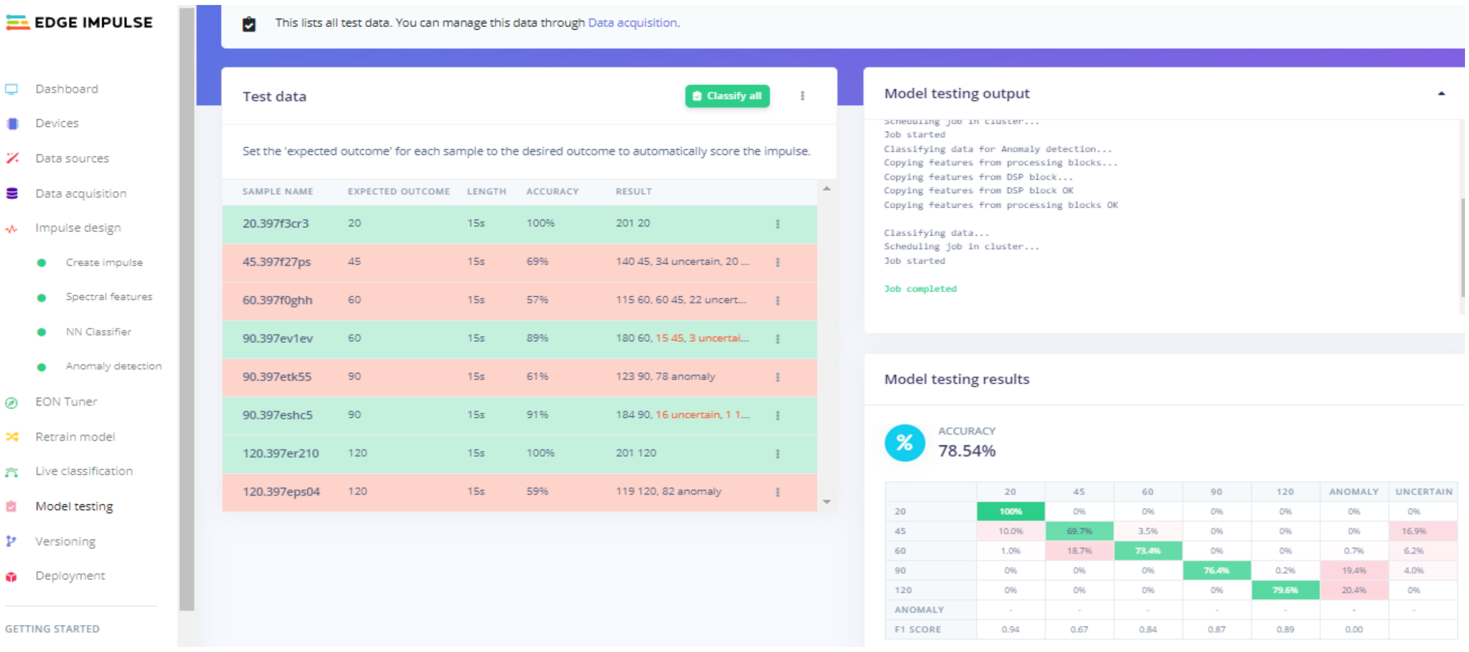
Confusion matrix (validation set)

	20	45	60	90	120
20	100%	0%	0%	0%	0%
45	0%	100%	0%	0%	0%
60	0%	4.3%	95.7%	0%	0%
90	0%	0%	0%	100%	0%
120	0%	0%	0%	0%	100%
F1 SCORE	1.00	0.98	0.98	1.00	1.00

Picture 3.28: Goniometer on leg_ first results

In this situation, the overall accuracy of the model is slightly better than the first case. The training cycles here were also 50. The 20 and 45 degrees give very good results reaching 100%. 60 degrees angle gives also a good percentage of prediction with 95.7%. However, there seems to be some predictions which confuse the model with the 45 degrees angle. Higher angles (90^o and 120^o) have a high percentage (100%) in this installation.

After running the Anomaly Detection block, we move to the Model Testing tab. As with the previous installation (goniometer with Arduino attached) we can realistically see how good the Model is working. The results of this are shown below (Picture 3.29):



This lists all test data. You can manage this data through Data acquisition.

Test data

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

SAMPLE NAME	EXPECTED OUTCOME	LENGTH	ACCURACY	RESULT
20.397f3cr3	20	15s	100%	201 20
45.397f27ps	45	15s	69%	140 45, 34 uncertain, 20 ...
60.397f0ghh	60	15s	57%	115 60, 60 45, 22 uncert...
90.397ev1ev	60	15s	89%	180 60, 15 45, 3 uncertai...
90.397etk55	90	15s	61%	123 90, 78 anomaly
90.397eshc5	90	15s	91%	184 90, 16 uncertain, 1 1...
120.397er210	120	15s	100%	201 120
120.397eps04	120	15s	59%	119 120, 82 anomaly

Model testing output

```
Scheduling job in cluster...
Job started
Classifying data for Anomaly detection...
Copying features from processing blocks...
Copying features from DSP block...
Copying features from DSP block OK
Copying features from processing blocks OK

Classifying data...
Scheduling job in cluster...
Job started

Job completed
```

Model testing results

ACCURACY
78.54%

	20	45	60	90	120	ANOMALY	UNCERTAIN
20	100%	0%	0%	0%	0%	0%	0%
45	10.0%	69.7%	3.5%	0%	0%	0%	16.9%
60	1.0%	18.7%	73.4%	0%	0%	0.7%	6.2%
90	0%	0%	0%	76.4%	0.2%	19.4%	4.0%
120	0%	0%	0%	0%	79.6%	20.4%	0%
ANOMALY	-	-	-	-	-	-	-
F1 SCORE	0.94	0.67	0.84	0.87	0.89	0.00	

Picture 3.29: Goniometer on leg results after Live Classification

The classification of the test data here gives different results from the initial estimation. Only 20^o seem to have good results. The other angles all have anomaly and uncertainty percentages. An overall image is shown below:

- 45^o: correct classification 69.7%. There is a small mis-classification with 20^o (10%) and 60^o (3,5%). Also, uncertainty at around 17%.
- 60^o: correct classification 73.4%. Misclassification with 45^o (18.7%) and uncertainty at around 6,2%.
- 90^o: correct classification 76.4%. There is only a small percentage of anomaly score at 19.4%.
- 120^o: correct classification 79.6%. There is only a percentage of anomaly score at 20.4%.

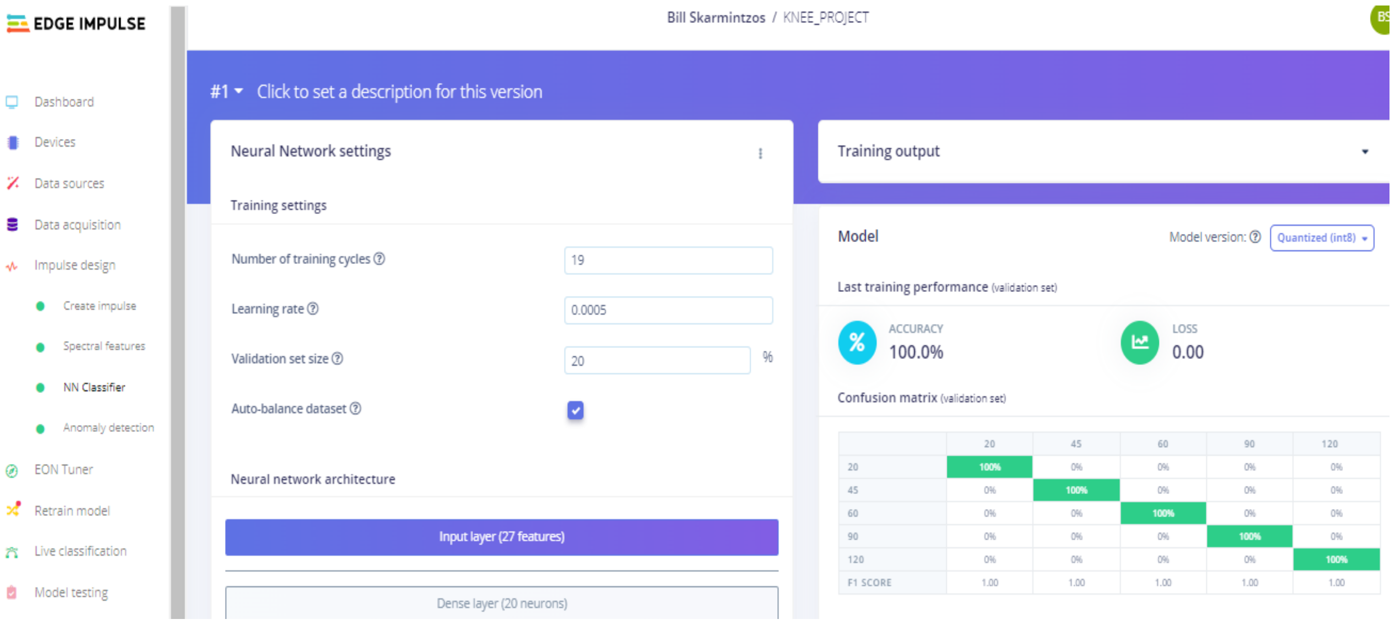
3.6.4 Knee Brace-Range of Motion- Results

In the third case we used a telescopic RangeOfMotion (ROM) knee brace. With the degree indications this device has, we can easily choose the range of motion of the leg in angle degrees. For example, by pulling and sliding the regulator to 45°, we have a range of motion (from full extend) of 45° degrees. After adjusting the device, we properly stick Arduino Nano 33 BLE to the lowest part of the brace and we start experimenting with the different angles.

The angles that were tested here are also 20°, 45, 60, 90 and 120 °.

We followed the procedure explained in the above chapters, concerning the creation of the machine learning models.

The results of the NN Classifier are shown in the image below (Picture 3.30). This is a first indication of how accurately the model is working and is a special feature that Edge Impulse offers.



Training settings

Number of training cycles: 19
Learning rate: 0.0005
Validation set size: 20 %
Auto-balance dataset: ✓

Model — Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 100.0%
LOSS 0.00

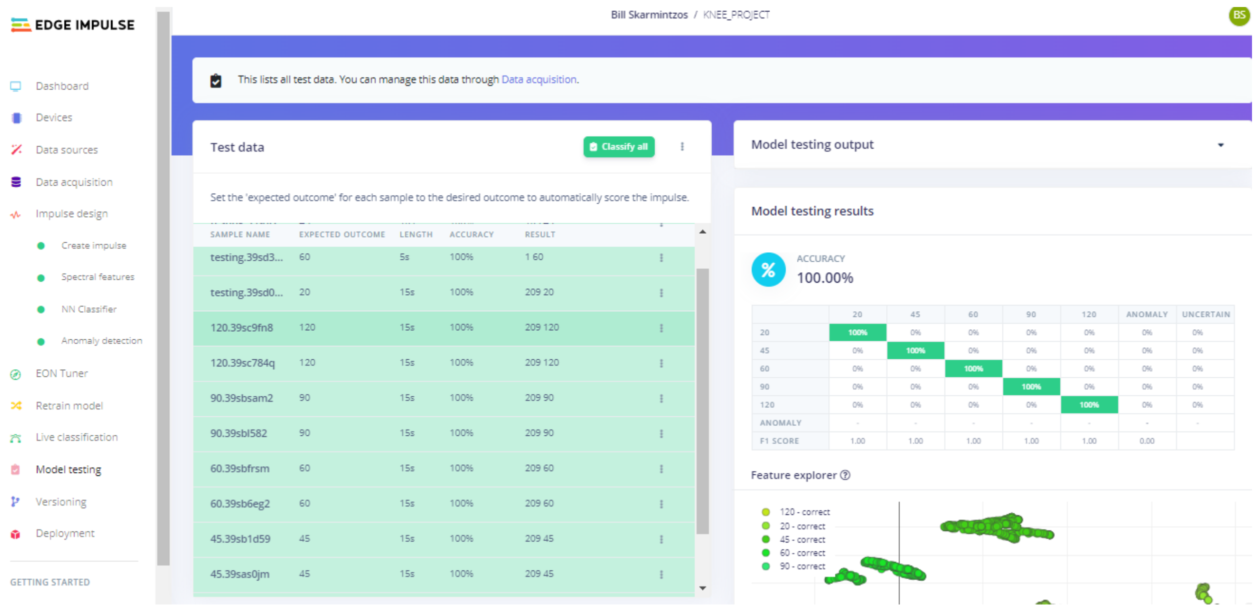
Confusion matrix (validation set)

	20	45	60	90	120
20	100%	0%	0%	0%	0%
45	0%	100%	0%	0%	0%
60	0%	0%	100%	0%	0%
90	0%	0%	0%	100%	0%
120	0%	0%	0%	0%	100%
F1 SCORE	1.00	1.00	1.00	1.00	1.00

Picture 3.30: Knee Brace ROM first results

It is apparent that the accuracy of this method is much better than the other methods. The model created with the installation we explained, seems to have a very good classification ability of the different angles we investigated. The accuracy is 100% with very high F1_score (close to 1.00). The number of training cycles was also low with only 19 cycles. This is a very good result as the model is trained fast and with accuracy.

After running the Anomaly Detection block, we move to the Model Testing tab. As with the previous installations (goniometer with Arduino attached) we can realistically see how good the Model is working. In this phase, all the Test Set is taken into consideration along with some extra testing samples. These samples, as we explained earlier, are taken after the model is trained, in the Live Classification tab. The results of this are shown below (Picture 3.30):



This lists all test data. You can manage this data through Data acquisition.

Test data

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

SAMPLE NAME	EXPECTED OUTCOME	LENGTH	ACCURACY	RESULT
testing.39sd3...	60	5s	100%	1 60
testing.39sd0...	20	15s	100%	209 20
120.39sc9fn8	120	15s	100%	209 120
120.39sc784q	120	15s	100%	209 120
90.39sbsam2	90	15s	100%	209 90
90.39sbl582	90	15s	100%	209 90
60.39sbfrsm	60	15s	100%	209 60
60.39sb6eg2	60	15s	100%	209 60
45.39sb1d59	45	15s	100%	209 45
45.39sas0jm	45	15s	100%	209 45

Model testing output

Model testing results

ACCURACY
100.00%

	20	45	60	90	120	ANOMALY	UNCERTAIN
20	100%	0%	0%	0%	0%	0%	0%
45	0%	100%	0%	0%	0%	0%	0%
60	0%	0%	100%	0%	0%	0%	0%
90	0%	0%	0%	100%	0%	0%	0%
120	0%	0%	0%	0%	100%	0%	0%
ANOMALY	-	-	-	-	-	-	-
F1 SCORE	1.00	1.00	1.00	1.00	1.00	0.00	

Picture 3.31: Knee Brace Live Classification results

The results here are very good. We notice that the model can distinguish the different angles with a very efficient way. There seems to be no anomaly or uncertainty percentage. Even on Live classification samples, the predictions are clear and correct with no anomaly or uncertainty points.

The following table summarizes the results found in previous chapters:

Table 3.1: Accuracy of Models

Model Accuracy (final accuracy with Live Classification samples)		
case1_Goniometer	case2_Goniometer on Leg	case3_Knee Brace
88.45%	78.54%	100%

3.7 Discussion

After having executed the three installations of our experiment and also gathered the model testing results, we can make some comments regarding the final results.

- In the installations with the goniometer and the Arduino Nano 33 attached to it, we can see that the uncertainty is high. One explanation for this, is that the goniometer, except for the 'up-down' move on the vertical axis, it also has some movements in the horizontal axis. The way the goniometer is assembled gives this flexibility. Basically, the goniometer's moving brace is connected with the circular part of the goniometer with a small metallic bolt. These small vibrations of the goniometer in the horizontal axis makes the acquired training set samples more complex to distinguish. Also, goniometer is less stable than the Knee brace. When it is attached, it needs to fit on the same location on the foot as this of when the data were acquired.
- With Live Classification samples added to the Test Set we have a more realistic way of examining the real accuracy of the models. After taking these samples we can see that the accuracy percentages (classification results) show an increase in uncertainty and anomaly. This is an indication that the models, in different environments and with different circumstances have a different response on how they make predictions.
- The angles that are relatively close and "write" almost equal arcs, give a higher percentage of false predictions. This is a normal consequence of the fact that the flexion-extension moves are identical. The only thing that changes is the arc each angle produces. So, it is probable that the Arduino Nano 33 BLE board IMU does not have the ability to distinguish so small differences in movement.
- For the installation in which the knee brace was used, we can conclude that it gives the most realistic environment for the experiment. Knee brace brings stability and efficiency. Arduino is held steadily on the brace and the position of it remains unchanged. The two installations of the experiment, which include the goniometer, give a more typical way to try and create a model. By simply recording the movement of a goniometer, gives a more plastic movement, without taking into consideration the mechanics of the movement of a human knee. By observing the anatomy of a human leg, we can understand that a goniometer cannot express the complexity of the extension and flexion movement. The knee brace on the other hand gives a more natural way of recording the flexion and extension as the device that is planned to be assembled, will be used by a patient that probably has a knee brace adjusted.
- In general vibrations and position of Arduino play an important role on the results of the model. We can infer that even small changes on direction and

position of Arduino or any vibrations could give different results or bring uncertainty to the models.

4 Edge Computing implementation

In this chapter, we intend to implement the idea of Edge Computing. More specifically, we will **use the model with the best accuracy**, created in the previous chapter and deploy it to the Arduino Nano 33 BLE board. Next, with proper modifications to the source code that arises from Edge Impulse Platform, we will activate the Bluetooth Low Energy module of the Arduino board and pass the predictions resulting from the model to a proper mobile application.

4.1.1 Scope Description

In order to have a full picture of the way Edge Impulse can enforce the idea of Edge Computing in collaboration with a development board such as Arduino Nano Sense 33, in this phase, with appropriate modifications to the model exported in previous steps, we try to disconnect the Arduino board from a central system and use its capabilities in order to work independently.

The targets intended to be accomplished here are:

- to use the model created in previous step, in order to have a prediction about the degrees of the angle, by the flexion-extension movement executed by a patient.
- to properly use the features given by the development board of Arduino Nano sense 33 in order to send information to a mobile device or system. This exact package of information will include the result/ prediction made by the model running On Device.

4.1.2 Bluetooth LE

Bluetooth is a radio technology, well-known for its short link communication[1]. The operational frequencies are between 2,400 and 2,485 GHz. With the use of radios, this technology focuses on establishing connections and information exchange between two or more devices.

Many big companies like IBM, Intel and Nokia, saw a great field of development as the idea of creating a short-range wireless link between devices could find many practical uses. A special group was formed in 1998 by those companies in order to standardize the protocol and set the rules in order to be functional. (Bluetooth® Special Interest Group (SIG)). Bluetooth versions that were released through the years are: Bluetooth® 1.0 was released around 1999, version 2.0 in 2004, version 2.1 in 2007, version 3.0 in 2009, version 4.0 in 2010, version 4.1 in 2013, version 4.2 in 2014, version 5.0 in 2016 and version 5.1 on 2019.

In a try to develop low-power wireless communication technology, SIG merged with Wibree (a wireless technology developed by Nokia, Nordic semiconductor and other companies) and the result of this was Bluetooth® Low Energy (BLE). Bluetooth® Low Energy was designed to reduce, significantly, the power consumption by reducing the amount of time that the Bluetooth® radio is on. Classic Bluetooth® and Bluetooth® Low Energy are both included since the Bluetooth® 4.0 specification, but here's the thing: Classic Bluetooth® and Bluetooth® Low Energy work differently, and they are not compatible. Generally speaking, classic Bluetooth® is mainly used for audio applications (wireless headphones, for example) while Bluetooth® Low Energy is more often seen in power-constrained applications (such as wearables and IoT devices, for example).[1]

Bluetooth Low Energy is based on the idea of roles and responsibilities. The main roles given to the devices that want to communicate are central (servers) and peripheral (clients). A peripheral device may be a smartwatch or any other collection of sensors. A central device is a device scanning information like a mobile device while a peripheral device advertises information. Basically, after a successful connection between two devices, a peripheral will broadcast information in nearby devices. Central devices can listen for any device that broadcasts information. When central devices pick up the advertised information, it tries to connect with the advertiser and interact with the advertised information.

Central devices try to categorize the information they receive from peripheral devices. They give a Unique Identification code called UUID. In that way, the information is properly organized and create entities called services. In that way, any peripheral device can easily find, select and interact with the service they need. Within each service will exist a list **of characteristics**. Once the peripheral device discovers these characteristics, it can write information to, request information from, and subscribe to updates from these characteristics

A central device can interfere with information with the following ways:

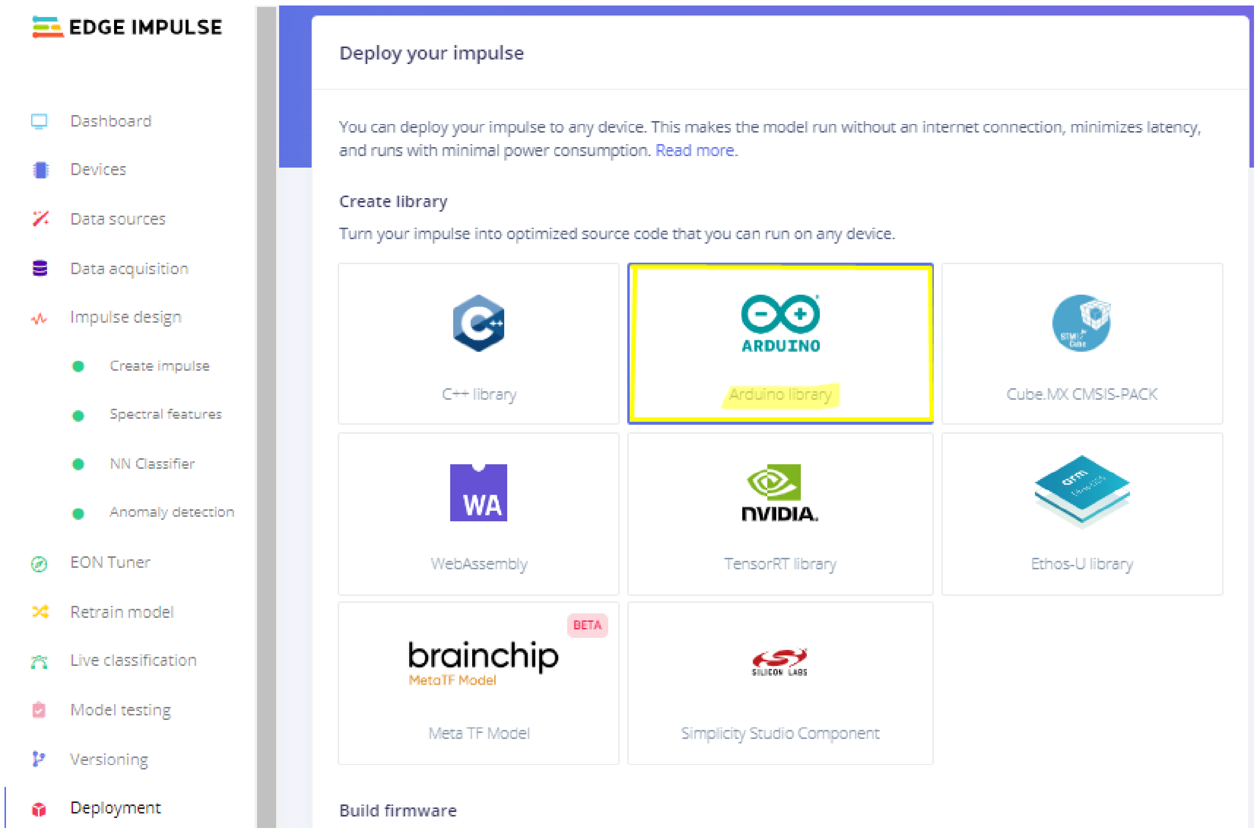
- Reading: the peripheral is asked to send the value of a specific characteristic and it is used mainly for information that do not change that often (e.g., version number etc.)
- Writing: modifies the value of a characteristic. It has to do with commands asking the peripheral to perform a certain task.

- Notifying the peripheral is asked to send continuous and updated information of a characteristic, without the central having to ask for it constantly.

4.1.3 Implementation of the idea

4.1.3.1 Download the Library

The first thing that we need to do is to have access to the script of the model for Arduino Nano 33. To run as an Arduino Library, we only have to choose the corresponding choice in the Edge Impulse Platform (Picture 4.1). A zip file will be downloaded including the Arduino library ready to be deployed on our device.


Deploy your impulse

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.

Create library

Turn your impulse into optimized source code that you can run on any device.

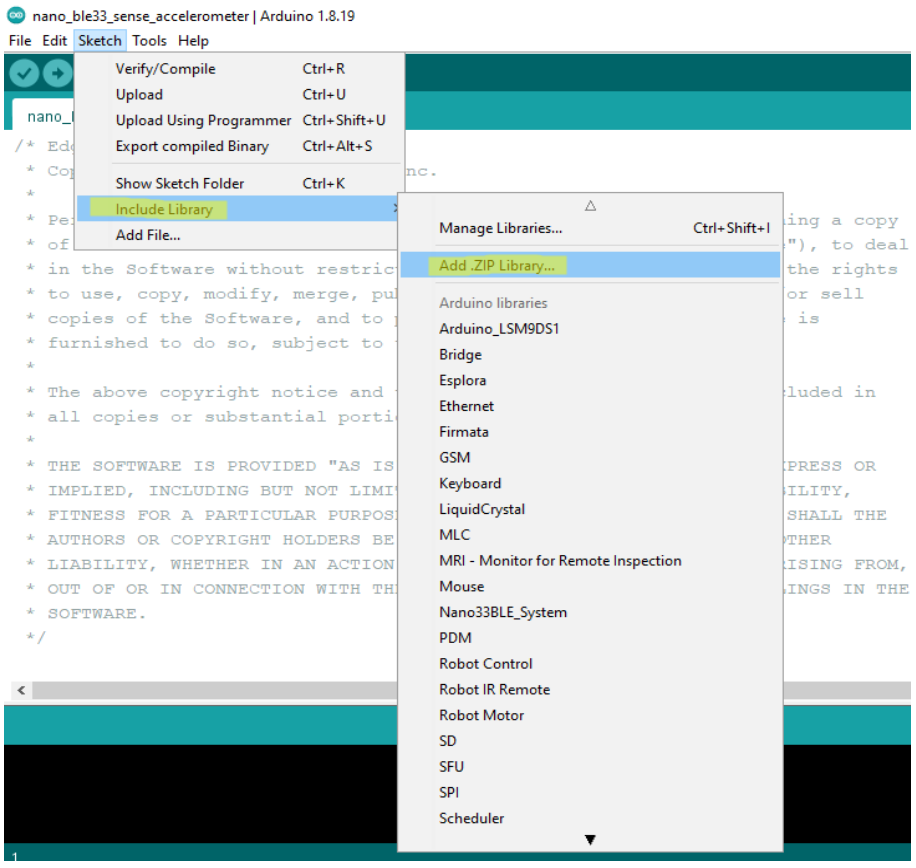
Build firmware

Picture 4.1: Edge Impulse- Arduino Library Download

4.1.3.2 Accessing Source code of the Model

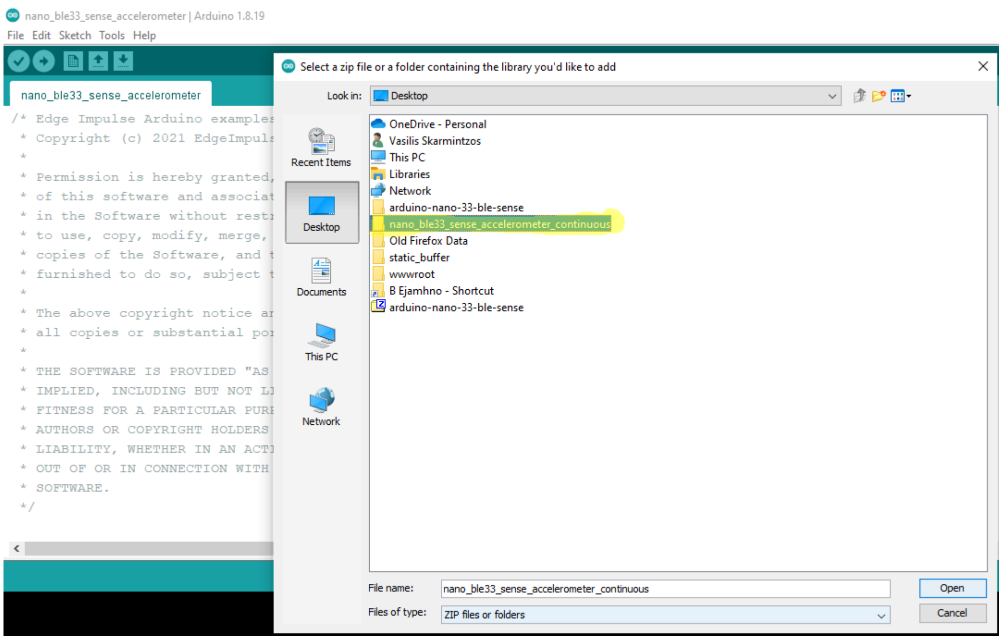
In order to have access to the source code of our model, we need to use Arduino IDE. The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board. We downloaded and installed the wanted software from: https://www.arduino.cc/en/software . After the installation is complete, we proceed to the following steps:

- On the “Sketch” tab, we choose “Add Zip Library” (Picture 4.2)



Picture 4.2: Arduino IDE Add zip Library

- Then we choose the file of the Library and Add it to the IDE (Picture 4.3).



nano_ble33_sense_accelerometer | Arduino 1.8.19

```
/* Edge Impulse Arduino examples
 * Copyright (c) 2021 EdgeImpuls
 *
 * Permission is hereby granted,
 * of this software and associat
 * in the Software without restr
 * to use, copy, modify, merge,
 * copies of the Software, and t
 * furnished to do so, subject t
 *
 * The above copyright notice an
 * all copies or substantial por
 *
 * THE SOFTWARE IS PROVIDED "AS
 * IMPLIED, INCLUDING BUT NOT LI
 * FITNESS FOR A PARTICULAR PURE
 * AUTHORS OR COPYRIGHT HOLDERS
 * LIABILITY, WHETHER IN AN ACTI
 * OUT OF OR IN CONNECTION WITH
 * SOFTWARE.
 */
```

Select a zip file or a folder containing the library you'd like to add

Look in: Desktop

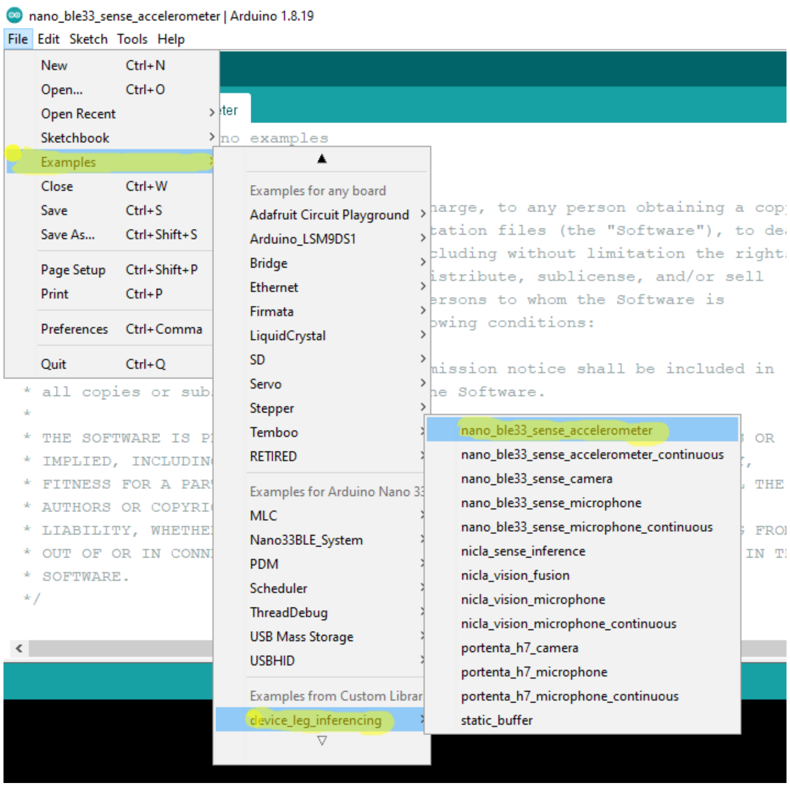
OneDrive - Personal
Vasilis Skarmintzos
This PC
Libraries
Network
arduino-nano-33-ble-sense
nano_ble33_sense_accelerometer_continuous
Old Firefox Data
static_buffer
wwwroot
B Ejamhno - Shortcut
arduino-nano-33-ble-sense

File name: nano_ble33_sense_accelerometer_continuous

Files of type: ZIP files or folders

Picture 4.3: Arduino IDE_ Choosing the library

- After that step, the library is inserted and we can now have access to the source code of the model created. We choose File →Examples→..and then we find the wanted model (Picture 4.4). The source code is then open to the IDE and we can make the necessary modifications and additions to it.



Picture 4.4: Arduino IDE_ Accessing the source code

4.1.3.3 *Modifying the source code*

Now that we have access to the source code of the Model, we can modify the code (according to [4]) in order to make the Arduino Nano 33 board able to send the predictions of the model to a mobile device.

What we intend to do here, is to make proper modifications and additions to the source code of the model that was built through Edge Impulse. We then install a suitable application to a mobile device in order to receive the predictions from the board. In that way, it is apparent how powerful and flexible the idea of edge computing can be.

In the following parts, we present the modifications and additions being made to the source code of the model.

- To enable Bluetooth® on the Nano 33 BLE Sense, we can use the ArduinoBLE library, and include it at the top of our sketch:

```
#include <ArduinoBLE.h>
```

- We then *Set Services and Characteristics*:

```
BLEService inferenceService("2d0a0000-e0f7-5fc8-b50f-05e267afeb67");
```

```
BLEStringCharacteristic inferenceCharacteristic("2d0a0001-e0f7-5fc8-b50f-
05e267afeb67", BLERead | BLENotify, 56); // remote clients will be able to
get notifications if this characteristic changes
```

- With another set of instructions inside setup function, we can make some choices regarding the Arduino Nano 33 Board. So, we can set the desired device name, set some connection intervals, advertise the service (in our case inference service) and add characteristic to the service (inferenceCharacteristic). In this set of instructions we also set the first value of the string as 'inference'. We also set an advertising interval (of 80 ms) and by calling `BLE.advertise();` we can start advertising the service.

Part of the source code from the setup() function is shown below:

```
//BLE_______________________________________
 if (!BLE.begin()) {   // initialize BLE
      Serial.println("starting BLE failed!");
      while (1);
 }
     BLE.setLocalName("ARDUINO_MSC");  // Set device name
    //Set the minimum and maximum desired connection intervals in units of 1.25 ms.
    //Bluetooth LE desired connection Interval 160ms - 200ms. Central has the final
word!
    BLE.setConnectionInterval(160, 200);
    BLE.setAdvertisedService(inferenceService); // Advertise service
    inferenceService.addCharacteristic(inferenceCharacteristic); // Add characteristic
to service
    BLE.addService(inferenceService); // Add service
    inferenceCharacteristic.writeValue("inference"); // Set first value string
    //Set the advertising interval in units of 0.625 ms
    //Bluetooth LE advertising interval around 50ms
    BLE.setAdvertisingInterval(80);
    BLE.advertise();  // Start advertising
#if (DEBUG_FLAG==1)
    Serial.print("Peripheral device MAC: ");
    Serial.println(BLE.address());
    Serial.println("Waiting for connections...");
#endif
```

- Inside the loop() part of the Arduino sketch we have another set of instructions. At first we wait for a central device to connect (our mobile device). If the devices reaches connection, we turn on a LED indication on board to express the successful connection.

Part of the source code from the loop() function

```
void loop()
{
     // wait for a BLE central
```

```
BLEDevice central = BLE.central();
String inferenceResult = "";

// if a central is connected to the peripheral:
if (central.discoverAttributes()) {
  // turn on the LED to indicate the connection:
  digitalWrite(LED_BUILTIN, HIGH);
#if (DEBUG_FLAG==1)
  // print the central's BT address:
  Serial.print("Connected to central: ");
  Serial.println(central.address());
  Serial.print("Switching on accelerometer... ");
#endif


  while (central.connected()){
```

- Another interesting part of the addition made to the source code, has to do with the sending procedure of the prediction. What we are aiming for here is to take the result of the model (the prediction of the model) and by setting it as a parameter to a specific function, to send it in our mobile device.
  For that cause, we call the
  `void sendInferenceOverBLE(String inferenceResult)`
  function. This function uses the -
  `inferenceCharacteristic.writeValue(inferenceResult);`-
  instruction to write the value of the prediction as a characteristic to the service.

Part of the source code from the loop () function where the prediction is sent to the central device:

```
// 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].value);

        //SEND PREDICTION VIA
BLE_____________________________________________________________________________
            if(result.classification[ix].value >0.7 && inferenceResult !=
result.classification[ix].label){
                  inferenceResult = result.classification[ix].label;
                  sendInferenceOverBLE(inferenceResult);
                }
          }
    #if EI_CLASSIFIER_HAS_ANOMALY == 1
        ei_printf("    Anomaly Score: %.3f\n", result.anomaly);
    #endif
  }//WHILE ENDS...
```

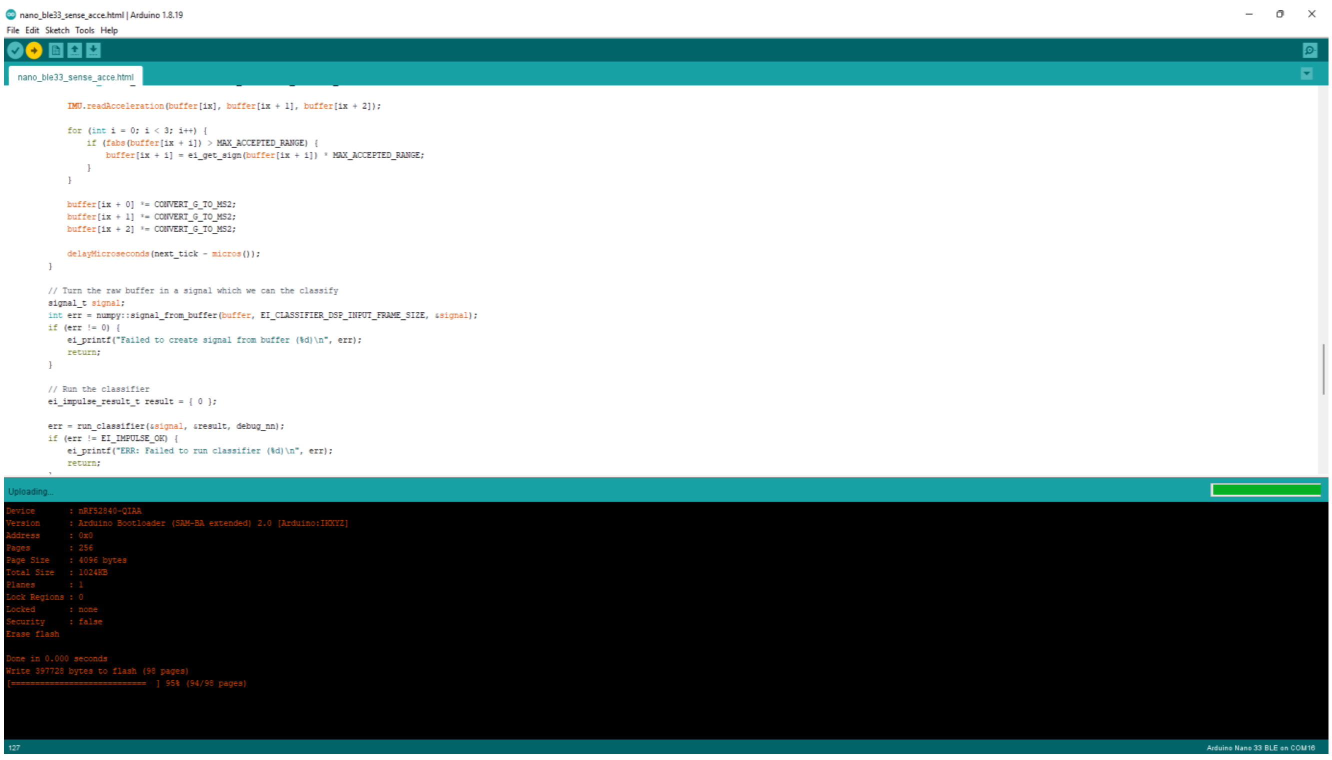
sendInferenceOverBLE() function

```
//BLE

  void sendInferenceOverBLE(String inferenceResult) {

#if (DEBUG_FLAG==1)
     Serial.println("Sending inference result over BLE TO THE MOBILE DEVICE..."); // 
print it to serial consle
#endif
     inferenceCharacteristic.writeValue(inferenceResult); // Write value
}
```

The whole source code can be found in Appendix I of this thesis.

4.1.3.4 Testing the results

The picture below shows the procedure of **Uploading** the source code to the Arduino Nano 33 BLE.(Picture 4.5). By pressing the corresponding button, the Uploading stars.


nano_ble33_sense_acce.html | Arduino 1.8.19
File Edit Sketch Tools Help
nano_ble33_sense_acce.html
Uploading...
127
Arduino Nano 33 BLE on COM16

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

        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;
            }
        }

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

        delayMicroseconds(next_tick - micros());
    }

    // Turn the raw buffer in a signal which we can the classify
    signal_t signal;
    int err = numpy::signal_from_buffer(buffer, EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, &signal);
    if (err != 0) {
        ei_printf("Failed to create signal from buffer (%d)\n", err);
        return;
    }

    // Run the classifier
    ei_impulse_result_t result = { 0 };

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

```
Device       : nRF52840-QIAA
Version      : Arduino Bootloader (SAM-BA extended) 2.0 [Arduino:IKXYZ]
Address      : 0x0
Pages        : 256
Page Size    : 4096 bytes
Total Size   : 1024KB
Planes       : 1
Lock Regions : 0
Locked       : none
Security     : false
Erase flash

Done in 0.000 seconds
Write 397728 bytes to flash (98 pages)
[============================  ] 95% (94/98 pages)
```

Picture 4.5 :Uploading the source code

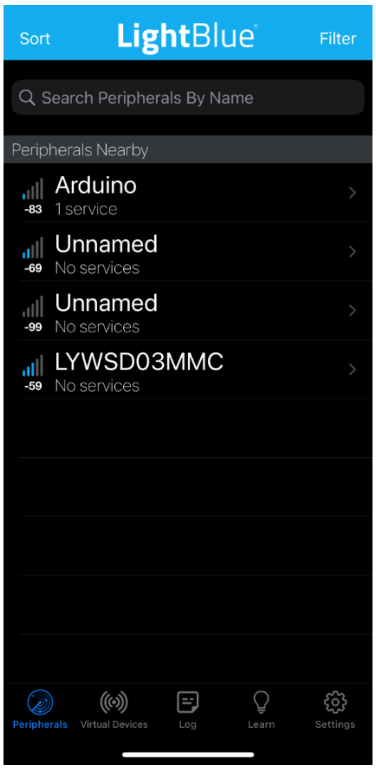
After a short period of time, the code is passed to the Arduino and is up and running.

In order to test the functionality of the code we are going to use a mobile application. In our case we choose to use "LightBlue®App". Both Android and IOS versions exists. In short, LightBlue® can connect us to devices that use Bluetooth Low Energy (also known as Bluetooth Smart, or Bluetooth Light). With LightBlue®, we can scan, connect to and browse any nearby

BLE device. It Full support of read, write, and notify is included to ease BLE firmware development efforts. We can also view the signal strength (RSSI) in real time to get an idea of how close you are to the BLE device, handy for finding BLE devices.

We open the application from our mobile device. The application, automatically searches for Bluetooth devices. The pictures below explain how the predictions are presented to the user through the application LightBlue:

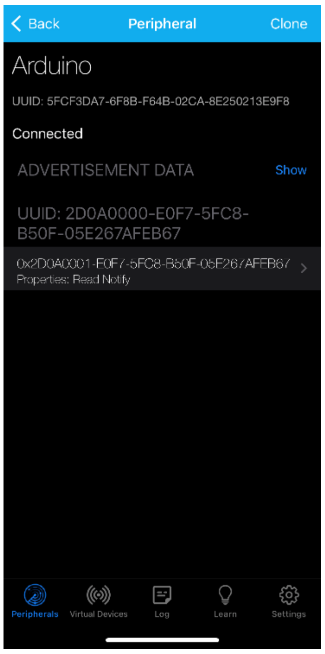
- When we open the application, we can see the nearby Bluetooth devices (Picture 4.6). We choose the Arduino device, with one service running.



Arduino -83 1 service
Unnamed -69 No services
Unnamed -99 No services
LYWSD03MMC -59 No services

Picture 4.6:LightBlue_starting page

- The next page (Picture 4.7) shows more information about the chosen device. We can distinguish information about UUID (service id) and also the properties of the characteristics (read and notify)



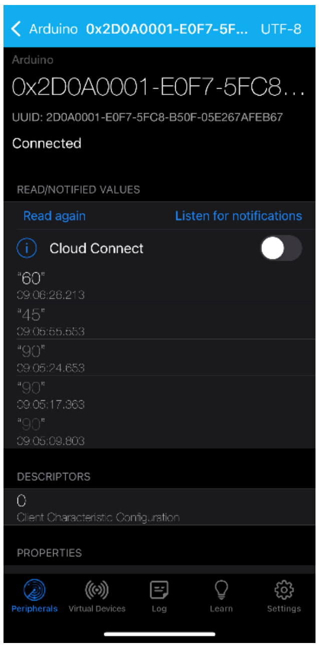
Arduino
UUID: 5FCF3DA7-6F8B-F64B-02CA-8E250213E9F8
Connected

ADVERTISEMENT DATA Show

UUID: 2D0A0000-E0F7-5FC8-B50F-05E267AFEB67
0x2D0A0001-E0F7-5FC8-B50F-05E267AFEB67
Properties: Read Notify

Picture 4.7:LightBlue Peripherals page

- We choose the service advertised by the Arduino board. The next page (Picture 4.8) is the page that shows the records of the service table, with the written characteristics. These characteristics are the predictions made by the model running on Arduino board. Every time a prediction is made, this prediction is sent to the mobile application via BLE and is shown to the user.



Arduino
0x2D0A0001-E0F7-5FC8...
UUID: 2D0A0001-E0F7-5FC8-B50F-05E267AFEB67
Connected

READ/NOTIFIED VALUES

"60"
09:06:26.213
"45"
09:05:55.553
"90"
09:05:24.653
"90"
09:05:17.363
"90"
09:05:09.803

DESCRIPTORS

0
Client Characteristic Configuration

PROPERTIES

Picture 4.8: LightBlue Service Characteristics

The above process shows us emphatically how important the use of edge computing is for the realization of applications. The saving of computing resources and energy is obvious as no central system is used. All actions are accomplished on device and as users we take advantage of the possibilities provided to us to utilize the information, we are interested in.

5 Bibliography

[1] Bagur, J. (2022, 08 08). *Connecting Nano 33 BLE Devices over Bluetooth®*. Retrieved from docs.arduino.cc/tutorials: https://docs.arduino.cc/tutorials/nano-33-ble-sense/ble-device-to-devicehttps://docs.arduino.cc/tutorials/nano-33-ble-sense/ble-device-to-device

[2] Brown, S. (2021, April 21). *Machine learning, explained*. Retrieved from Management Sloan School: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained

[3] Chaitanya, K. (2022, January 10). *Edge Impulse Leverages TinyML and AutoML for Efficient Microcontrollers*. Retrieved from Verdict: https://www.verdict.co.uk/edge-impulse-leverages-tinyml-and-automl-for-efficient-microcontrollers/

[4] Dannegård, B. (2022, 08 08). *Nano 33 BLE Sense Cheat Sheet*. Retrieved from docs.arduino.cc: https://docs.arduino.cc/tutorials/nano-33-ble-sense/cheat-sheet

[5] Education, I. C. (2020, July 15). *Machine Learning*. Retrieved from IBM: https://www.ibm.com/cloud/learn/machine-learning

[6] M. G. Sarwar Murshed, Christopher Murphy, Daqing Hou, Nazar Khan, Ganesh Ananthanarayanan, and Faraz Hussain. 2021. Machine Learning at the Network Edge: A Survey. ACM Comput. Surv. 54, 8, Article 170 (October 2021), 37 pages. https://doi.org/10.1145/3469029.

[7] Microsoft. (n.d.). *Machine Learning on the Edge*. Retrieved from Microsoft Research: https://www.microsoft.com/en-us/research/project/machine-learning-edge/

[8] Minaee, S. (2019, 10 28). *20 Popular Machine Learning Metrics. Part 1: Classification & Regression Evaluation Metrics*. Retrieved from towards science: https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce

[9] Salian, I. (2018, August 2). *SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning?* Retrieved from nvidia.com: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/

[10] Singh, D. A. (n.d.). *Knee Range of Motion and Movements*. Retrieved from boneandspine: https://boneandspine.com/knee-range-of-motion/

[11] Tan, N. (2018, May 1). *Why Machine Learning on The Edge?* Retrieved from towardsdatascience.com: https://towardsdatascience.com/why-machine-learning-on-the-edge-92fac32105e6

[12] Tanner, G. (n.d.). *Arduino Nano 33 BLE Sense Overview*. Retrieved from gilbert tanner: https://gilberttanner.com/blog/arduino-nano-33-ble-sense-overview/

[13] *What Is Machine Learning (ML)?* (2020, June 26). Retrieved from Berkeley School of Information: https:// online.berkeley.edu/blog/what-is-machine-learning/

[14] Wikipedia. (2022, July 27). *Machine Learning*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Machine_learning

Appendix I

Source code loaded to Arduino Nano 33 BLE sense

```
/* Edge Impulse Arduino examples
 * Copyright (c) 2021 EdgeImpulse Inc.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 */

/* Includes ---------------------------------------------------------------- */
#include <KNEE_PROJECT_inferencing.h>
#include <Arduino_LSM9DS1.h>

#include <ArduinoBLE.h>        //Click here to get the library: http://librarymanager/All#ArduinoBLE



#define  DEBUG_FLAG 0 // Flag to control Serial debugging messages

BLEService inferenceService("2d0a0000-e0f7-5fc8-b50f-05e267afeb67");
BLEStringCharacteristic inferenceCharacteristic("2d0a0001-e0f7-5fc8-b50f-05e267afeb67", BLERead |
BLENotify, 56); // remote clients will be able to get notifications if this characteristic changes



/* Constant defines -------------------------------------------------------- */
#define CONVERT_G_TO_MS2    9.80665f
#define MAX_ACCEPTED_RANGE  2.0f        // starting 03/2022, models are generated setting range to +-
2, but this example use Arudino library which set range to +-4g. If you are using an older model,
ignore this value and use 4.0f instead
```

```
/*
 ** NOTE: If you run into TFLite arena allocation issue.
 **
 ** This may be due to may dynamic memory fragmentation.
 ** Try defining "-DEI_CLASSIFIER_ALLOCATION_STATIC" in boards.local.txt (create
 ** if it doesn't exist) and copy this file to
 ** `<ARDUINO_CORE_INSTALL_PATH>/arduino/hardware/<mbed_core>/<core_version>/`.
 **
 ** See
 ** (https://support.arduino.cc/hc/en-us/articles/360012076960-Where-are-the-installed-cores-located-)
 ** to find where Arduino installs cores on your machine.
 **
 ** If the problem persists then there's not enough memory for this model and application.
 */

/* Private variables ------------------------------------------------------- */
static bool debug_nn = false; // Set this to true to see e.g. features generated from the raw signal

/**
* @brief      Arduino setup function
*/
void setup()
{
    // put your setup code here, to run once:
    Serial.begin(115200);
    Serial.println("Edge Impulse Inferencing Demo");

    if (!IMU.begin()) {
        ei_printf("Failed to initialize IMU!\r\n");
    }
    else {
        ei_printf("IMU initialized\r\n");
    }

    if (EI_CLASSIFIER_RAW_SAMPLES_PER_FRAME != 3) {
        ei_printf("ERR: EI_CLASSIFIER_RAW_SAMPLES_PER_FRAME should be equal to 3 (the 3 sensor
axes)\n");
        return;
    }

      //BLE_______________________________________
 if (!BLE.begin()) {   // initialize BLE
      Serial.println("starting Bluetooth® Low Energy failed!");
      while (1);
 }
     // set advertised local name and service UUID:
     BLE.setLocalName("ARDUINO_MSC");  // Set device name
    //Set the minimum and maximum desired connection intervals in units of 1.25 ms.
    //Bluetooth LE desired connection Interval 160ms - 200ms. Central has the final word!
    BLE.setConnectionInterval(160, 200);

    BLE.setAdvertisedService(inferenceService); // Advertise service
    inferenceService.addCharacteristic(inferenceCharacteristic); // Add characteristic to service

    // Add service
    BLE.addService(inferenceService);

     // set the initial value for the characteristic:
    inferenceCharacteristic.writeValue("inference");

    //Set the advertising interval in units of 0.625 ms
    //Bluetooth LE advertising interval around 50ms
    BLE.setAdvertisingInterval(80);

    // Start advertising
    BLE.advertise();

#if (DEBUG_FLAG==1)
    Serial.print("Peripheral device MAC: ");
    Serial.println(BLE.address());
    Serial.println("Waiting for connections...");
#endif


}

/**
 * @brief Return the sign of the number
 *
 * @param number
 * @return int 1 if positive (or 0) -1 if negative
 */
float ei_get_sign(float number) {
    return (number >= 0.0) ? 1.0 : -1.0;
}

/**
* @brief      Get data and run inferencing
*
* @param[in]  debug  Get debug info if true
*/
void loop()
{
     // listen for Bluetooth® Low Energy peripherals to connect:
```

```
BLEDevice central = BLE.central();
String inferenceResult = "";

// if a central is connected to the peripheral:
if (central.discoverAttributes()) {

  // turn on the LED to indicate the connection:
  digitalWrite(LED_BUILTIN, HIGH);

#if (DEBUG_FLAG==1)
  // print the central's BT address:
  Serial.print("Connected to central: ");
  Serial.println(central.address());
  Serial.print("Switching on accelerometer... ");
#endif

  // while the central is still connected to peripheral:
  while (central.connected()){
  ei_printf("\nStarting inferencing in 2 seconds...\n");

  delay(2000);

  ei_printf("Sampling...\n");

  // Allocate a buffer here for the values we'll read from the IMU
  float buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE] = { 0 };

  for (size_t ix = 0; ix < EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE; ix += 3) {
      // Determine the next tick (and then sleep later)
      uint64_t next_tick = micros() + (EI_CLASSIFIER_INTERVAL_MS * 1000);

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

      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;
          }
      }

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

      delayMicroseconds(next_tick - micros());
  }

  // Turn the raw buffer in a signal which we can the classify
  signal_t signal;
```

```
    int err = numpy::signal_from_buffer(buffer, EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, &signal);
    if (err != 0) {
        ei_printf("Failed to create signal from buffer (%d)\n", err);
        return;
    }

    // Run the classifier
    ei_impulse_result_t result = { 0 };

    err = run_classifier(&signal, &result, debug_nn);
    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].value);
        if(result.classification[ix].value >0.7 && inferenceResult !=
result.classification[ix].label){
              inferenceResult = result.classification[ix].label;
                inferenceCharacteristic.writeValue(inferenceResult); // Write value
                 }
    }
#if EI_CLASSIFIER_HAS_ANOMALY == 1
    ei_printf("    anomaly score: %.3f\n", result.anomaly);
#endif
}
  }
}

// #if !defined(EI_CLASSIFIER_SENSOR) || EI_CLASSIFIER_SENSOR != EI_CLASSIFIER_SENSOR_ACCELEROMETER
// #error "Invalid model for current sensor"
// #endif
```