Final Report

Gesture Recognition using Surface EMG and Machine Learning

Student: Alejandro Avila Carrión
alejandro.carrion.19@ucl.ac.uk

Primary Supervisor: Prabhav Nadipi Reddy
p.reddy@ucl.ac.uk

Secondary Supervisor: Henry Lancashire
h.lancashire@ucl.ac.uk

Date: 17/05/2022

Department of Medical Physics and Biomedical Engineering of
University College London

Abstract

The use of Machine Learning (ML) for hand gesture recognition has become increasingly common in modern day myoelectric prosthetics. These myoelectric prosthetics use the Surface Electromyography (sEMG) signals of the muscles remaining after amputation to control the movements and grasps of the mechanical parts of robotic hands. This process involves the extraction of features from sEMG signals to carry out the detection of everyday hand gestures which are intended by the amputee. sEMG features contain important information about the sEMG signal and can be fed into ML algorithms to detect the relationship between the signal and the desired gesture.

Gesture recognition can be performed using limited sEMG data thanks to the combined effect of feature extraction and ML. In addition, the implementation of sEMG feature extraction and ML into microcontrollers for convenient and portable gesture recognition is limited by the memory constraint of microcontroller themselves. However there are available tools such as the TensorFlow Lite Converter to decrease the size of ML algorithms. The aim of this project was to prove the principle of using feature extraction, ML and microcontrollers to detect the hand gestures corresponding to different sEMG. This was achieved successfully using a simple ML model, but the implementation of additional ML methods, such as Deep Learning, would be more appropriate for the task.

Table of contents

1. Introduction

1.1 Background on the use of sEMG for Gesture Recognition

Amputations are devastating injuries; they involve the loss of a limb and are commonly performed through surgery when a heavily affected limb is not treatable or possible to salvage (1). Trauma, infection, peripheral vascular disease (e.g. diabetes, which accounts for 50% of US amputations), cancer and congenital abnormalities are amongst the most common causes of amputations (1–3). In 2005, 1.6 million people in the US suffered from limb loss, of which 20-25% had their upper arm affected, and this number has been growing yearly at a rate of 50,000-100,000 ever since (4, 5). The loss of the hand linked to upper limb amputations has significant effects in daily functional performance and has proven to have a negative impact on the mental health of amputees (1, 5).

The hand is an essential and highly proficient part of the human body. It can adopt a wide range of shapes and motions, which allows it to interact with its environment with great versatility to carry out different tasks (6, 7). Therefore, losing a hand is coupled with a loss of ability to perform daily activities, including work, self-care, and hobbies, which significantly reduces the quality of life of the amputee (8). There are also important psychological implications associated with hand mutilation, including anxiety and depression (8). Furthermore, the impact of hand loss expands to a societal level in terms of productivity costs (9). To overcome these problems, prosthetic hands and arms now are produced and are accessible to amputees in most parts of the world, allowing some return of lost function and improving everyday living (7, 10).

In recent years, with advancements on physiology, software and mechatronics, increasingly advanced prosthetic hands have been developed; with the ultimate aim of completely emulating the functionality of the human hand. Some of these prosthetic hands include the BeBionic Hand by Ottobock and the iLimb by Össur, which offer 14 and 36 different grips and hand positions, respectively (11, 12). These technologies make use of what is known as surface electromyography (sEMG) for voluntary control of the mechanical hand, in order to adopt the available grips and hand shapes offered by the manufacturer (11–13). The use of sEMG gives the amputee an easier and more intuitive way to manipulate their prosthetic, mimicking in some way how limb motion is controlled naturally (7).

sEMG involves the reading of electric potentials on the surface of the skin generated during skeletal muscle contractions (14). These electric potentials are specific to particular muscle movements, which allows their use for controlling prosthetics (15). Just as prior to limb loss, when a hand amputee thinks of moving a missing finger, the muscles that remain post-amputation will contract involuntarily in a specific way; this generates specific sEMG signals at the skin surface associated to those muscle movements (16). Different sEMG signals are equally produced when the patient thinks of realising other hand motions, each associated to its respective muscle

contractions (16). sEMG-controlled prosthetics make use of this distinct nature of sEMG to translate the movements the amputee intended to make from available muscles to the prosthetic hand (7).

Advantages of sEMG involve non-invasiveness, and the possibility to record the sEMG signal(s) at many locations over the available muscles with the use of surface electrodes (17). The versatility of sEMG acquisition is highly advantageous due to the different shapes and forms upper limb amputations can take (5). However, detecting the intended movements of the user using sEMG requires an accurate signal recording, processing, and classification (7); sEMG signals are known to be highly noisy and variable, and therefore it is essential to carry out these steps correctly (18). While the more basic myoelectric (or sEMG-controlled) prosthetics operate using non-pattern recognition (i.e. are limited to an on/off, threshold or proportional control), proficient myoelectric hands like the iLimb detect underlying patterns in the sEMG signals to control the prosthetic (7, 12).

Non-pattern recognition greatly restricts the number of different movements the prosthetic can make and therefore pattern recognition is superior in offering a wider variety of hand movements (7). While non-pattern recognition entirely relies on sEMG amplitude, pattern recognition extracts multi-dimensional (including time, frequency, space and fractal domain) features from the sEMG signals to detect the intended motions of the user (7). As mentioned earlier, just like the sEMG signals that are produced, these features are highly specific to the muscle movements triggered when the amputee thinks of moving the missing extremity in different ways (7). sEMG features contain important information about the sEMG signal and are then used to recognise the desired user hand gestures (7). The process of transforming sEMG signal information (through features) into an electrical signal that can be used to control the robotic hand most commonly involves the use of machine learning (ML) algorithms (19).

ML algorithms involve mathematical models that make predictions based on previous observations (20, 21). They are widely used to analyse the features extracted from sEMG signals, in order to detect (predict) the corresponding gestures intended by the user (22). They have proven to give highly accurate results in the area of hand gesture recognition on a number of different studies (22); however, there are yet some challenges to be overcome. As mentioned earlier, sEMG signals are very noisy; their amplitude is random in nature, their position is unstable and they vary between individuals (7, 23). In addition, due to the small muscle mass normally present in the upper limb, the muscle contractions are relatively weak, meaning the sEMG produced is quite small (7). This makes the analysis of sEMG signals harder and limits the degrees of freedom with which the prosthetic hand can be controlled (7).

Ways to increase the available information for gesture recognition for more diverse outcomes involve paring sEMG with other sensors (e.g. an accelerometer) or using a large number of electrode pairs to record sEMG at many locations (24). The later method is not very feasible practically, where a robust and compact robotic arm

with integrated sEMG sensors is usually desired by the user (11, 12). In addition, having too much information may add complexity to the data processing process, which may limit its own practical application (24). Therefore, commercial prosthetic devices have had to compromise and can only offer control for a limited number of degrees of freedom (7).

Another method to increment the number of predicted movements in myoelectric prosthetics is to perform a more precise analysis of the sEMG signals (7). For this, better ML, signal processing and feature extraction methods must be developed to distinguish between the noisy sEMG signals (7). Taking into consideration the practical aspect mentioned earlier, the aim of this project is to perform an sEMG analysis and develop a simple ML algorithm to prove the principle that accurate hand gesture recognition with ML is indeed possible with a limited amount of sEMG information. The ultimate goal of this project is to contribute to the vast number of studies which use sEMG and ML for hand gesture recognition (22). In addition, this project will take a step further aiming to implement the model to a microcontroller. Hopefully, this paper will give insightful information that can be used to increase the degrees of freedom of natural control in prosthetics to amputees.

1.2 Background on Machine Learning

Machine learning (ML), particularly the branch of supervised learning, involves the use mathematical models in order to map underlying relations between pairs of data (20). These pairs of data are obtained from real life examples through experiments, and ML models resemble in some way how scientists find the relationship between variables of interest from experimental data (21). To create a ML model, the programmer gives the data of interest to a special type of algorithm and lets this algorithm discover the relationships in the data by itself (21). ML models are useful in situations where finding the exact connection between variables is difficult, costly or time-consuming (20). The mapped relations made by ML models can then be used to detect (predict) what would be the corresponding data pair when giving to the ML model only one of the pieces of data within a pair (20).

The piece of data that you input into the ML model is called the feature vector (or features) and the corresponding data pair that the ML model detects (predicts) is called the label (20). The feature vector contains in each dimension a value that the describes the real-life phenomenon that you want to predict somehow, and the label describes the real-life phenomenon by itself (20). For example, when detecting speech with ML [like voice assistants do (e.g. Alexa)], the feature vector could include, but is not limited to, the frequency spectrum of the voice signal, the signal length, or the energy in the signal; meanwhile, the label would be the actual word/sentence that produced that particular signal frequency spectrum, length, and energy (21). Gesture recognition with sEMG is analogous to this case and uses its own sEMG features to detect gestures (7, 22).

There are many different algorithms that can be used to create ML models that predict gestures from sEMG. However, as found out by the literature review by Jaramillo-Yánez et al. (2020) exploring 65 hand gesture recognition studies, the most common algorithms used are support vector machines (SVM) and convolutional neural networks (CNN) (22). SVMs involve the drawing of a boundary line between features in order to distinct the corresponding labels (20). When the feature vectors lie above this boundary line, they are given one label, and when they lie below this boundary line, they are given another label; hence, this algorithm is only able to relate the features to two different phenomena (20). On the other hand, CNNs involve a network of 'neurons', each containing its own parameters, which are 'activated' or updated to link our input features to their matching labels (20).

The parameters within neurons that are tweaked to give the correct predictions for a given feature vector are called the weight and bias values (20). The process of 'activating' or updating these parameters is called training and involves feeding feature vectors which have been priorly labelled by the programmer (meaning each feature has a matching label) to the CNN (21). The algorithm then uses these labelled features (analogous to the data pairs described earlier) during training to establish the relationships between the features and the labels (20). The degree by which the neuron's weights and bias values are 'activated' depends upon an activation function (20). In addition, contrary to SVMs, CNNs are able to do multi-class classification, which means the features can be used for predicting multiple labels (20).

2. Methods

2.1 Machine Learning Algorithm Selection

While SVMs occupy less disk space and are generally faster than CNNs, they are not able to classify more than two gestures, which greatly limits the future possibilities of this study (20). Therefore, CNNs were utilised to create the ML model in this project. In addition, we made use of the simplicity offered by the Keras tool in TensorFlow for building CNNs. TensorFlow is an open-source platform initially developed by Google used to build, train, evaluate, and deploy ML models, with plenty of different examples online. The way we interacted with TensorFlow later on in the Methods was by using the Python programming language.

2.2 Steps in Gathering a Dataset

2.2.1 Sample Collection

As defined by The Hundred-Page Machine Learning Book by Burkov A., "a dataset is a collection of labelled examples" (20). This involves the labelled feature vectors described earlier, which are necessary to train our CNN. In order to have an sEMG dataset appropriate for gesture detection it is first necessary to acquire a

significantly big database of sEMG signals from which to extract the features (21). The size of the dataset is very important for giving accurate predictions, particularly when the data samples are quite different (i.e. for being able to make correct predictions amongst highly variable features) (21). There is no golden rule as to how big a dataset should be, but ideally, you would want it to be as big as possible for better ML outcomes (21). In this case a database of 450 different sEMG signals recorded at two different locations in the arm was utilised for feature extraction.

The database included the sEMG information for 6 different common hand movements, as shown in Figure 1 (25). It was divided into 2 folders; in one of the folders (Database 1) you had the sEMG gesture data for 5 different subjects (3 females and 2 males) and in the other folder (Database 2) you had the sEMG gesture data for only one subject (male), taken for 3 consecutive days (25). For Database 1, the signals were taken over a duration of 6 seconds, and each subject was asked to perform each gesture 30 times (so 30 pieces of sEMG data per gesture per subject) (25). On the other hand, for Database 2, the signals were measured for 5 seconds, and the subject conducted the grasps 100 times for each gesture each day (so 100 pieces of sEMG data per gesture per day) (25). The sEMG data was provided as .mat files, one file per subject in Database 1 and one file per day in Database 2.

Within each .mat file, the sEMG data was neatly organised into matrices. Each matrix had the different sEMG samples (trials) organised in its rows, with the columns holding the time-dependent (sampled) data points for each sample. The matrices were named according to the gesture for and the recording location at which the sEMG signals were taken. The sEMG signals were obtained at the Flexor Capri Ulnaris and Extensor Capri Radialis, Longus and Brevis muscles, with the Flexor Capri Ulnaris signals labelled as 'Channel 1' and the Extensor Capri Radialis, Longus and Brevis signals labelled as 'Channel 2'. For example, the sEMG signals taken for a Spherical grasp at the Flexor Capri Ulnaris for one of the subjects in Database 1 was saved in a 30-row matrix called 'spher_ch1'. Likewise, the sEMG signals taken for a Hook grasp at the Flexor Capri Radialis, Longus and Brevis for one of the days of Database 2 was saved in a 100-row matrix called 'hook_ch2'.

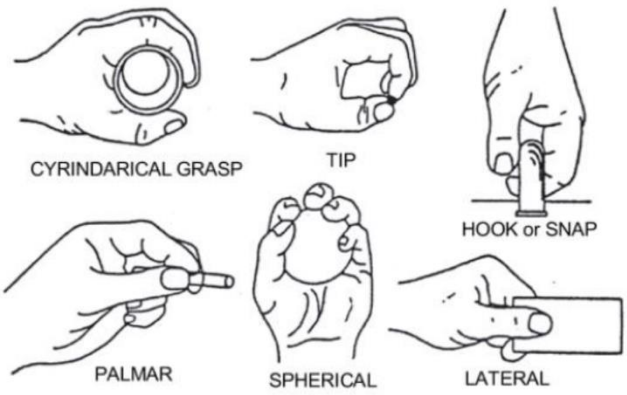


Figure 1. Hand gestures for which the sEMG data was available. From the UCI ML repository website (25).

The sEMG database was found in an online ML repository hosting website (25). The samples were taken using the National Instruments (NI) Labview as a programming kernel, with a NI Analog-to-Digital (NI USB-009) Converter connected to a PC (25). Two differential sEMG sensors were used to take the sEMG signals and the signals were then passed on to the Handheld EMG System by Delsys Bagnoliâ (a two channel EMG system) before getting digitised (25). In each of the experiments/days, the subjects executed the hand gestures repeatedly, grasping different objects in order to perform these gestures, and they were allowed to carry out the task with whichever force they wanted (25). The surface electrodes used for sEMG recording were held in position with elastic bands and an additional electrode was placed in between as the reference ground (25).

The sEMG was sampled with a 500 Hz sampling rate (so a 1/500 = 2ms sampling period), bandpass-filtered between 15Hz and 500 Hz, and notch-flitered at 50Hz to remove mains noise (25). The frequency of EMG commonly lies between 0-500 Hz, with the main energy component of the signal happening from 50-150 Hz (26). Therefore this sampling frequency seemed appropriate for meeting the 2×sampling frequency requirement needed in Nyquist's criterion. In addition, the signals' amplitude came measured as a voltage (25). In order to carry out the project, two of the offered gestures in the database were arbitrarily chosen and used for gesture recognition using ML and sEMG. These were the Hook grasp (used for heavy load support in practice) and the Spherical grasp (used for spherical tool holding in practice).

2.2.2 Sample Pre-processing

The sEMG data from Database 1 and Database 2 was imported into MATLAB (see Code 1) for the 2 chosen gestures. The Hook grasp signals were referred to as 'Gesture 1' within the code while the Spherical grasp signals were referred to as 'Gesture 2'. MATLAB was the coding platform used to carry out the feature extraction of the sEMG signals of the database. We worked with Database 1 and Database 2 separately as the sEMG signals within both data subsets had different lengths [or total number of data points (For Database 1, there were (6s/2ms =) 3000 sample points, vs (5s/2ms =) 2500 sample points for Database 2)]. As the signals from different subjects/days were merged into a large matrix before working with them, it was not possible to combine differently sized signals from the two (Database 1 and Database 2) data subsets.

Before directly extracting sEMG features, the sEMG signals had to be modified and processed; this was to improve the quality of the features and the performance of gesture recognition down the line. Sample pre-processing involved offset correction and endpoint detection. Removing the offset of the signal was essential as sEMG features are heavily dependent on the zero-crossing line; this was done by subtracting the mean value of the signal. After that, the signal was cut to only maintain the higher

energy part of the signal; this was done through a coding process which involved obtaining the energy in the signal, quantising the signal and using some threshold values to locate the endpoints. The code looped through all the samples of each data subset, performing the mentioned data pre-processing, and can be found in Code 2.

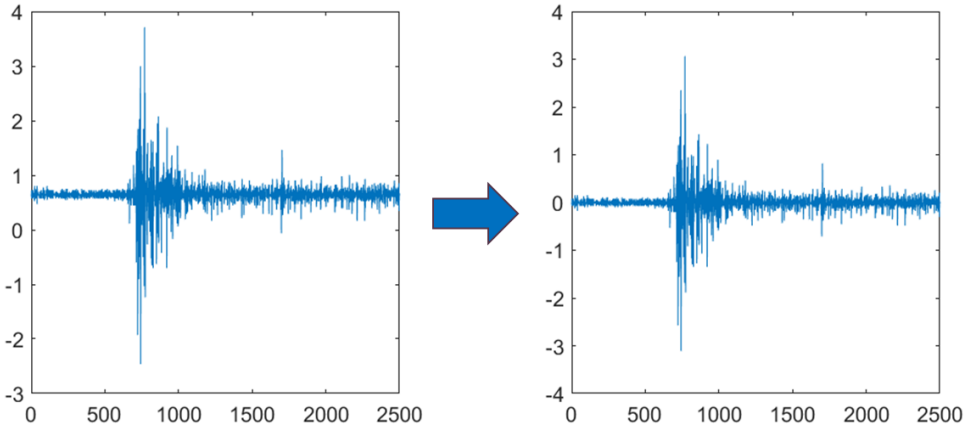


Figure 2. Example of offset correction. On the y-axis you have signal amplitude and on the x-axis you have length along the signal [number of sample points]

Code 3 contains the endpoint detection code, which additionally to cutting the signal at the region of interest, retrieves the length of this region. After the endpoints of the signal were detected, the values outside the endpoints were set to zero – This was done instead of directly cutting the signal (vector) in order to keep the size of the main matrix (containing all the different samples of each data subset) consistent. Adding zeros outside the endpoints did not affect the sEMG feature values. The length of the cut signal (in number of sample points) was directly outputted by the function as 'wl' to calculate sEMG features later on.

The initial threshold 'thr' value set in Code 1 for endpoint detection was found after looping through all of the sEMG signals from Database 2 [where you can appreciate the ground noise of the acquisition (as compared to Database 1, whose signals were purely sEMG in most cases)] and getting the mean minimal quantised energy value. This mean minimal value represented the amplitude of the ground noise in the sEMG signal recordings. The code utilised to find the threshold is in Code 4 and Code 5 in the Appendix. The 'thr' value was then multiplied by a factor of 1.875 to set the start point threshold and by a factor of 2.25 to set the end (or finish) point threshold. These multiplication factors were set manually by visually inspecting the Database 2 sEMG signals for the two different gestures for the two different channels. After gradually going through all the sEMG signals using Code 6, the multiplication factors were tweaked to only maintain the main part of the signal.

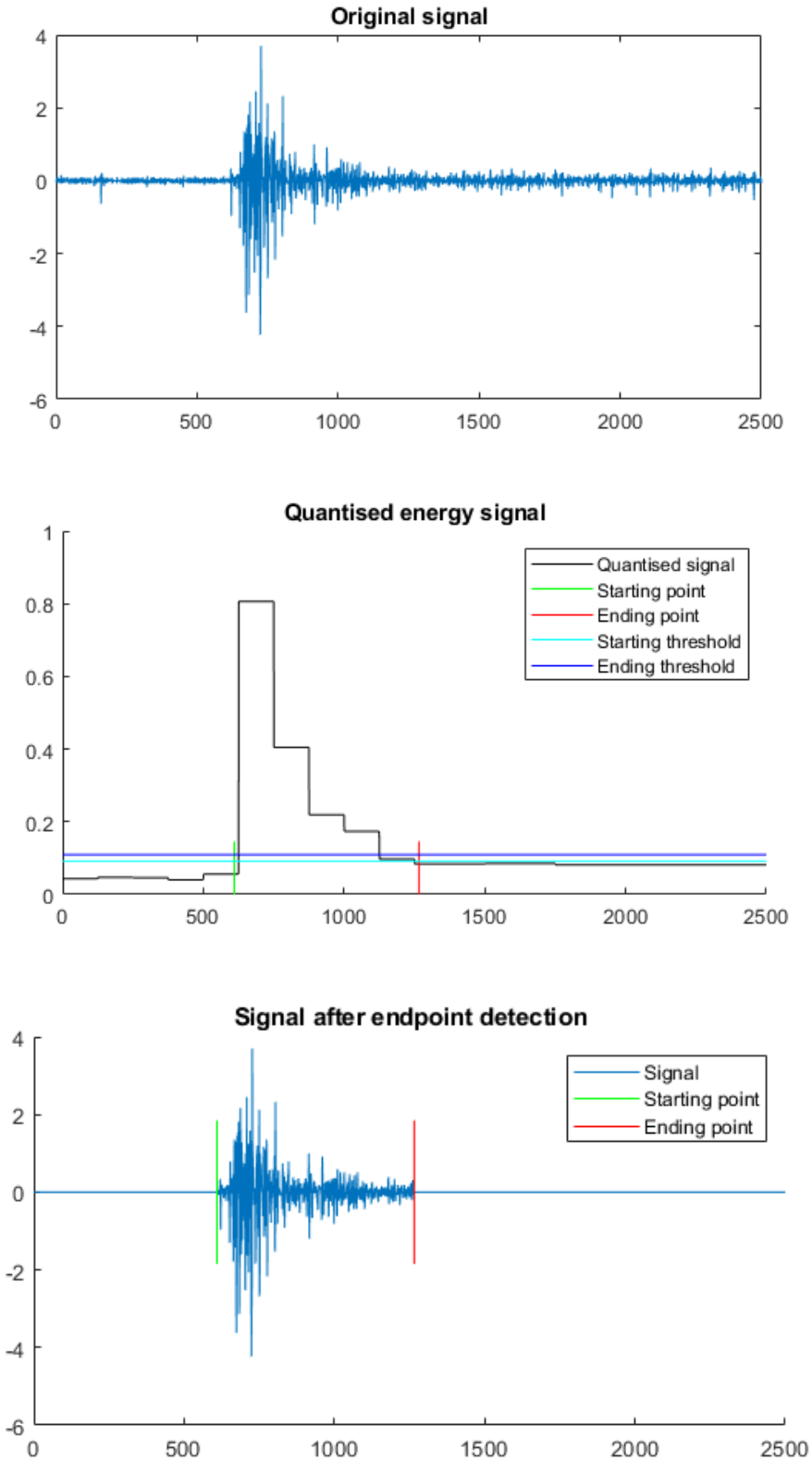


Figure 3. Example of the output obtained from Code 6 to manually inspect the sEMG signals before, during and after endpoint detection. On the y-axis you have signal amplitude and on the x-axis you have length along the signal [number of sample points]

2.2.3 Feature Extraction

Once the sEMG signals were appropriately pre-processed, their features were extracted. The code used for this can be found in Code 7, which calls on many other functions to extract the selected features, and then combines the features from each channel into a single vector. There are many possibilities and domains from which to extract sEMG features, including the time, frequency, space and fractal domains, as depicted by Jaramillo-Yánez et al. (22). However, the following time-domain features were chosen for their simplicity and the ability to easily construct functions for their calculation (as feature extraction was implemented into the microcontroller later on):

1. Mean Absolute Value (MAV)
2. Number of Zero Crossings (ZC)
3. Number of Slope (gradient) Sign Changes (SSN)
4. Waveform Length (WL)

- The MAV was obtained by first getting the absolute value of the signal and then calculating its mean (see in Code 8) using the waveform length from endpoint detection (which was in number of sample points). It represents the mean energy of the signal.
- The ZC involves the number of times the signal crosses the zero (or x-axis) line. It was obtained by getting the sign (+ve, -ve or 0) of all of the data points in the signal and counting the number of times the sign changed from the start to the end of the signal (see Code 9).
- The SSC is the number of times the gradient of the signal changes (between +ve, -ve and 0). To get it, the gradient at each point of the signal was first calculated using the values of the two adjacent points (using central differences). The sign of all the gradient points in the signal was then retrieved and the number of times the gradient sign changed from the start to the end of the signal was counted (see Code 10).
- The WL is the time length of the signal (in this case of the main signal part as obtained from endpoint detection). It was determined by using multiplying the 'wl' value (which was in number of sample points) by the sampling period (which was 2 ms) (see Code 11).

2.2.4 Even Mixing of the samples

The sEMG feature vectors of Database 1 and 2 containing our chosen features, calculated during feature extraction, were then combined into big matrix (each row of this new matrix having the feature vector for each sEMG sample). Since the location of the sEMG samples for each subject/day was already known (from when the two data subsets were imported into MATLAB), a code was designed to equally spread the data from the different subject/days (Code 12). It is important to mix the data evenly in this

way to improve the outcomes of the ML model and to avoid overfitting, which will be discussed later (20).

2.2.5 Exporting MATLAB features to Python

After evenly mixing the feature vectors within the matrix, the feature matrices for Gestures 1 and 2 were exported as .txt files to be used and processed by Python, which is where the main ML code was developed. The resulting .txt files contained 8 columns, with the MAV, ZC, SSC and WL from the Channel 1 signal in columns 1, 2, 3 and 4, respectively, and the MAV, ZC, SSC and WL from the Channel 2 signal in columns 5, 6, 7 and 8, respectively. The code utilised for this can be found at the end of Code 1. In addition, the sEMG signal from just one sample (post pre-processing) was exported to test the implementation of feature extraction and of the ML model into the microcontroller.

2.2.6 Labelling Feature Vectors

The main ML code was written as a Jupyter notebook (which is a special document format that enables you to mix Python code, writing and graphics together) and executed in Google Colaboratory (which allows you to run Jupyter notebooks in a virtual machine). Running the code in the cloud was an advantage as it did not use resources of the local PC. The Jupyter notebook used in this case can be found in https://colab.research.google.com/drive/1znmGRG4JW81ADdCLYPzFru6a4pT622Iw?usp=sharing; a duplicate of the python part of the notebook was also copied into Code 13. The notebook was constructed upon one of the TensorFlow Lite examples found in the main TensorFlow Github page (27).

The first parts of the notebook involve importing TensorFlow and additional Python libraries, such as numpy and pyplot, which are necessary for manipulating data and to produce figures that display the data, respectively. After this, the sEMG feature vectors were extracted from the .txt files created by MATLAB and assigned a label to create the dataset required to build the ML model. In our case, we used one-hot encoding for labelling, where each bit represents a probability from 0 to 1 that the features belong to a particular gesture. For Gesture 1 samples, it was chosen to be a probability of 1 in the first bit and a probability of 0 in the remaining bits, so resulting the label was [1, 0].

Similarly, for Gesture 2 samples, a probability of 1 in the second bit and a probability of 0 in the remaining bits was set, so the corresponding label was [0, 1]. Advantages of using one-hot encoding for labelling include the easiness with which the code can be expanded to include additional gestures in future studies. After labelling, the sEMG feature matrices from Gestures 1 and 2 were mixed and combined into a large new labelled feature matrix (which is analogous to the required dataset). The Gesture 1 and 2 data had to be mixed to ensure the ML model received equal

information from both gestures during training. Subsequently, the dataset was split back into x values (features) and y values (labels) to shape it into the correct format to appropriately feed it into the ML model.

2.3 Splitting dataset for Training, Validation and Testing

To evaluate how well a ML model performs, it is required to compare its predictions to new (unused) data during and after training (20). The evaluation of the model accuracy during training is referred to as validation, while the same evaluation just after training is referred to as testing (21). In both the training and testing cases, it is essential to use fresh data that has not been already used by the model to map the relations between the features and the labels (21). To have this fresh (unused) data, the dataset is commonly divided between training, validation, and testing on a 60%-20%-20% basis (21). Our code employed the same splitting ratio mentioned to assess model performance.

2.4 Model Architecture

There are many different factors to be considered when building the model architecture of a CNN. These include the number of neuron layers in the model, the number of neurons in each layer, and the activation function used to update the neurons' weight and bias values (21). When choosing the primary model architecture, the initial CNN created copied the structure of the model used in Hailong's et al. (2018) study on 2-channel sEMG gesture recognition (24). This model had 3 layers of neurons, with the input layer 26 neurons long, a hidden layer with 17 neurons, and the output layer consisting of only two neurons (in order to output the 2-long vector of label probabilities) (24). In addition, the activation function was originally set to 'Relu'.

The notation used to depict model architecture in this paper will of the form AAxBBxCCx…, where AA is the size of the input layer, and BB, CC, etc. are the consecutive hidden layers (so for the example above you would have a 26x17 CNN, as defined). After setting the model architecture, the ML model was compiled with Categorical Crossentropy as the loss parameter and (Categorical) Accuracy as the performance metric. These parameters are normally used to measure the error in multi-class classification problems which use one-hot encoding, like it is our case (28). Ideally, you want the Categorical Crossentropy to be as close to 0 as possible and the (Categorical) Accuracy near to 1 for a highly performing model (20). Using these loss parameters, different model architectures were tested and evaluated (see in the Results section).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 26)                182
_________________________________________________________________
dense_1 (Dense)              (None, 17)                459
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 36
=================================================================
Total params: 677
Trainable params: 677
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. A depiction of the original CNN architecture used, as outputted by the code in the Jupyter notebook.

2.5 Training the Model

Just like how diversly CNN can be built, there are many different possibilities when training the ML model. As mentioned earlier, training involves passing the labelled feature vectors into the neural network and updating the weight and bias values of neurons to establish the underlying relations (21). In order to do this, the model checks how far its predictions deviate from the expected labels, and adjusts the weights and biases accordingly to increase the likeability of being correct in the next predictions (21). During training, this correcting process is run multiple times on the whole dataset, with each complete run through the dataset known as an epoch (21).

Within each epoch, the data runs through the neural network in batches (21). Each batch involves the aggregation of several pieces of data, which are passed into the model at the same time to produce prediction values; the correctness of these predictions is then assessed in conjunction to update the weights and biases (so parameter tuning only occurs once per batch) (21). In practice, the chosen epoch number is a compromise between training speed, overfitting and overall model accuracy (20). Overfitting occurs when the model is able to predict very well the training dataset by not so well the validation and test datasets, and normally occurs as a result of high epochs numbers (20).

Also, when training the ML model, you want the batch size to be as small as possible; this adds a higher variability (noise) in the learning process, forcing the model to adjust its parameters more often and avoiding generalised predictions (29). However, small batch sizes also considerably increase training time, and therefore, just like with the epoch number, it is all about the compromise between the factors (29). Taking account the previous points, a batch size of 8 and an epoch number of 1000 were chosen to train the model; the outputs of different batch size-epoch number combinations were also explored (see in the Results section). After training the model, it was saved into different TensorFlow formats for the next part of the Methods.

2.6 Implementation to Microcontroller

2.6.1 Compressing and Exporting the Model

Due to their limited memory, when incorporating ML models to microcontrollers, you want to reduce the model size as much as possible (21). For this, you can convert the trained model into a memory-efficient format using the TensorFlow Lite Converter provided by TensorFlow (21). One additional technique for reducing the size of the model is called quantisation (21). Here, the precision of the model's weights and bias is reduced, saving memory space, commonly without an effect on accuracy (21). In addition, quantised models run faster, due to simpler involved calculations (21). Converting to model to TensorFlow Lite format, with and without quantisation was evaluated, in terms of impact on memory space and model accuracy.

After transforming the model to a space-efficient format, it was converted into a C source file, which can be used by microcontrollers to load the ML model. In our case, we deployed the model to an Arduino Nano 33 BLE microcontroller. For this, the resulting array in the C source file was copied and pasted into one of the examples of the Arduino_TensorFlowLite library in Arduino, and the corresponding array length was substituted as well. The Arduino files created to implement the ML model into the microcontroller can be seen in Code 14, and particularly, the array was substituted into the 'model.cpp' file.

2.6.2 Arduino Microcontroller Code

The Arduino code used contained many files. All of the files, except the 'main.ino' file had an associated header file (with an .h extension) in order to call them from the 'main.ino' file. The first part of the 'main.ino' file involved setting up TensorFlow Lite and error reporter dependencies and calling the other files from the code. The code is then able to perform feature extraction on the pre-processed channel signals and use these features and the previously defined ML array to make its predictions. The output after inputting the feature vectors into the model and obtaining some predictions (process known as inference), was sent to an output handler function to display the run results in the Serial Plotter.

Most of the code involved in setting up the ML model was left as it was in the example provided by Arduino. However, the Tensor Flow Arena size was a variable of interest. When creating the interpreter that makes the predictions in the Arduino, it was necessary to allocate a space of working memory for the model, which is used during model running. This memory space is named the Tensor Arena. It is used to store the inputs, outputs and intermediate arrays of the ML model during inference, and in the given TensorFLow Lite example, had an allocated 2000 bytes. Usually, it is difficult to determine the exact appropriate size of the Tensor Arena, as it varies with different model architectures, numbers of inputs and outputs (21) – However, ideally,

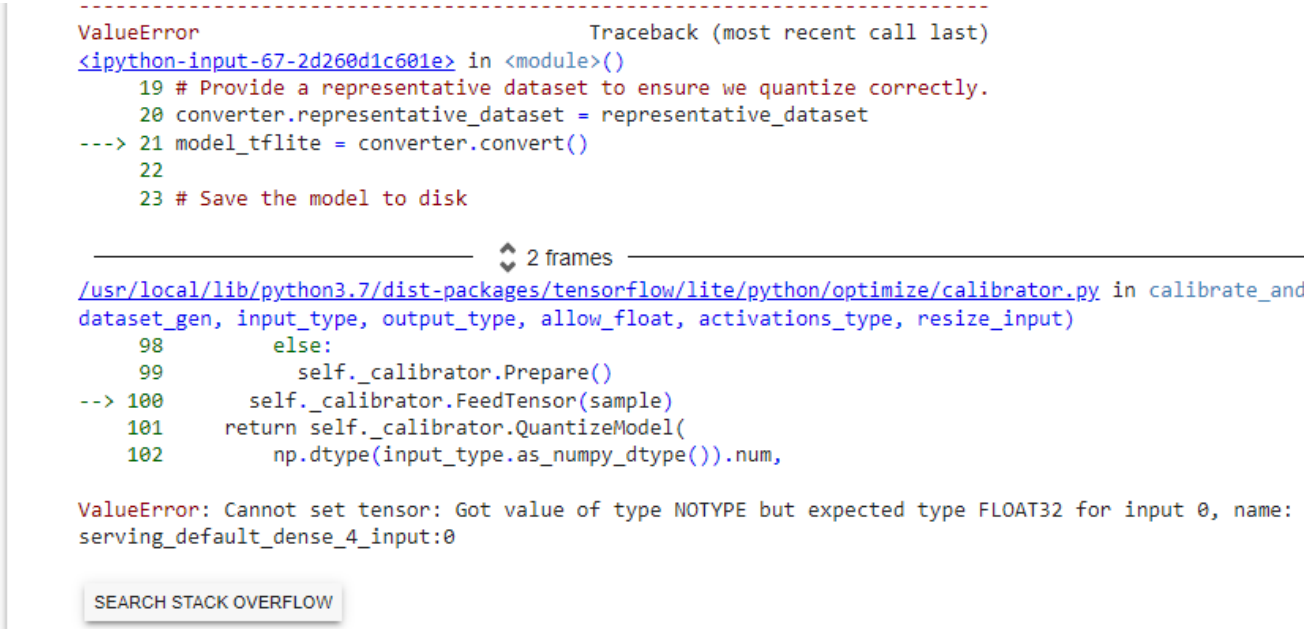
you want it to be as small as possible due to the limited memory in microcontrollers (21). For this project, the Tensor Arena was left at 2000 bytes since the model ran perfectly with this memory allocation.

The feature extraction code can principally be found in the 'feature_provider.cpp' file in Code 14 and involved the construction of functions using for loops and if statements to extract the MAV, ZC and SSC features. The pre-processed Channel 1 and Channel 2 signals for a single trial (exported to .txt files earlier by MATLAB) were substituted (copied and pasted) into the 'ch1_sample' and 'ch2_sample' variables, as well as their corresponding waveform lengths (in number of sample points). The code then performed feature extraction on these pre-defined arrays. Note that the length of 'ch1_sample' and 'ch2_sample' was calculated before jumping into the feature extraction functions – This was done as calling the array from within the function only gave a pointer to the array, from which it is not possible to obtain the array length directly. The number of sample points was necessary in order to perform the for loops that calculate the different features.

The features were then displayed in the Serial port, fed into the ML model (using part of the template code) and the results of inference were given to the 'HandleOutput' function in the 'output_handler.cpp' file. When the prediction for a given run exceeded a threshold value in the 'HandleOutput' function, the Serial port printed the corresponding gesture (Gesture 1/Gesture 2) and one of the built-in LEDs of the main Arduino board was lit (with a blue colour when Gesture 1 was recognised and with red colour when Gesture 2 was identified). An example of the resulting Serial port output of Arduino is depicted further on (see in Results).

2.7 Data Types

The main data type utilised was 'single'. This was because the ML TensorFlow Lite model required single data types as an input for running inference (see Figure 5). The majority of the code in MATLAB and Python was built with single data types as there was no memory constraint in these cases. In Arduino, 8-bit signed integers ('int_8') and unsigned 16-bit integers ('unsigned int') where additionally used whenever possible to reduce the code size. Particularly, 'int_8' was used to save the +ves, -ves and 0s in feature extraction, and 'unsigned int' was used to store the lengths of the sEMG arrays and the waveform lengths (in number of sample points).



```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-67-2d260d1c601e> in <module>()
     19 # Provide a representative dataset to ensure we quantize correctly.
     20 converter.representative_dataset = representative_dataset
---> 21 model_tflite = converter.convert()
     22
     23 # Save the model to disk

                                  2 frames
/usr/local/lib/python3.7/dist-packages/tensorflow/lite/python/optimize/calibrator.py in calibrate_and
dataset_gen, input_type, output_type, allow_float, activations_type, resize_input)
     98         else:
     99           self._calibrator.Prepare()
--> 100       self._calibrator.FeedTensor(sample)
    101     return self._calibrator.QuantizeModel(
    102         np.dtype(input_type.as_numpy_dtype()).num,

ValueError: Cannot set tensor: Got value of type NOTYPE but expected type FLOAT32 for input 0, name:
serving_default_dense_4_input:0

SEARCH STACK OVERFLOW
```

Figure 5. Output of the Jupyter notebook when trying to run inference on the TensorFlow Lite model with a 'double' data type.

3. Results

3.1 Feature Testing

For testing model performance (also done in the consecutive sections of Results) the ML model constructed in the Methods was run 6 different times, slightly adjusting some of its paraments. The loss and accuracy for the whole dataset were written down for each run and for the best run (with the lowest loss and highest accuracy), the training, validation and testing accuracies were additionally noted. As mentioned earlier, the Categorical Crossentropy was the loss parameter and the (Categorical) Accuracy was the performance metric. A Categorical Crossentropy close to 0 and a (Categorical) Accuracy near to 1 gives a higher performing model (20).

To test which of the features were most/less important for gesture recognition, the ML model was run with all the features vs with one of the features missing. The results are displayed in Table 1:

For: • 26x17 • Batch size 8, 1000 epochs • ReLu	**Run number**						**Mean**
Featured used	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Run 6**	
All of the features							
Accuracy	0.8944	0.9233	0.9033	0.9144	0.9144	0.9111	0.9102
Loss	n. r.	0.21243	n. r.	n. r.	n. r.	n. r.	n. r.
No MAV							
Accuracy	0.8944	0.9022	0.9	0.9189	0.92	0.8822	0.9030
Loss	0.26757	0.26929	0.26406	0.24919	0.26653	0.31734	0.2723
No ZC							
Accuracy	0.8678	0.8611	0.8667	0.8689	0.8744	0.8478	0.8645
Loss	0.30635	0.30006	0.28686	0.30339	0.27999	0.31197	0.2981
No SSC							
Accuracy	0.8833	0.8889	0.8967	0.9156	0.8933	0.8844	0.8937
Loss	0.25178	0.23359	0.23577	0.24747	0.24088	0.24578	0.2425
No WL							
Accuracy	0.9078	0.9189	0.8822	0.9011	0.8978	0.91	0.9030
Loss	0.22051	0.21369	0.27596	0.2524	0.28688	0.24771	0.2495

Table 1. Run results of ML model when using different combinations of features. Note that n.r. means the data was not recorded in that instance. The best runs are highlighted in blue, and the mean results are highlighted in red.

Remark that for comparison of the different cases, the best run results were principally used as they are a clear indicator of how proficient a model can get. The mean results were also important as they highlight the average performance of the model. For lower mean accuracies and losses, you would have to run the model more times to obtain a run with good results (the best run), so high mean results are important for saving time during training. In this case, lowering the number of features fed into the model decreased performance.

The resulting impact from removing each feature, from high to low, was listed as follows: 1. removing ZC, 2. removing SSC, 3. removing MAV, 4. removing WL. It can be seen how the results from the no WL runs do not differ that much from the all-feature runs. Therefore, considering model speed and size, the WL was discarded from the final version of the model. A lower number of inputs means the model has to map less relationships during training and perform less mathematical operations during inference, so this gives lower model sizes and greater speeds (21).

3.2 Model Architecture Testing

The next parameters to be tested involved the number of neuron layers in the CNN, and the corresponding number of neurons in each layer. The following model

architectures where tested, parting from the original 26x17 model used in Hailong's et al. (2018) study (24): (using the notation defined in the Methods →) 64x32x16, 52x34, 26x17, 13x9. The obtained results can be seen in Tables 2 and 3.

For: • All features • Batch size 8, 1000 epochs • ReLu	**Run number**						**Mean**
Architecture used	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Run 6**	
64x32x16							
Accuracy	0.9122	0.9067	0.9117	0.9089	0.9122	0.92	0.9120
Loss	n. r.	n. r.	n. r.	n. r.	n. r.	0.22700	n. r.
52x34							
Accuracy	0.9089	0.9222	0.9133	0.9156	0.9067	0.8967	0.9106
Loss	n. r.	0.20711	n. r.	n. r.	n. r.	n. r.	n. r.
26x17							
Accuracy	0.8944	0.9233	0.9033	0.9144	0.9144	0.9111	0.9102
Loss	n. r.	0.21243	n. r.	n. r.	n. r.	n. r.	n. r.
13x9							
Accuracy	0.86	0.8878	0.8511	0.9138	0.9111	0.8489	0.8788
Loss	n. r.	n. r.	n. r.	0.21690	n. r.	n. r.	n. r.

Table 2. Run results of ML model when using different model architectures. Note that n.r. means the data was not recorded in that instance. The best runs are highlighted in blue, and the mean results are highlighted in red.

For: • All features • Batch size 8, 1000 epochs • ReLu	**For best run**					
Architecture used	**Training set**		**Validation set**		**Testing set**	
	Loss	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Accuracy**
64x32x16	0.14324	0.9481	0.39564	0.8611	0.30962	0.8944
52x34	0.13536	0.95	0.35223	0.8611	0.27725	0.9
26x17	0.15641	0.9389	0.35418	0.8778	0.23872	0.9222
13x9	0.17832	0.9286	0.36441	0.85	0.18515	0.9333

Table 3. Run results of ML model for the best run, when using different model architectures.

It can be seen from Table 2 that the largest and more complex model architectures (64x32x16 & 52x34) don't yield an increase in accuracy or a significant decrease in loss. In addition, they were more prone to overfitting by having larger training set performances and lower testing set performances compared to the other

two smaller models. Therefore, they were discarded for the final ML model. The 26x17 and 13x9 models gave similar best run results (including overfitting), but the mean results for the 13x9 case were inferior (meaning it took a larger number of tries to obtain such best run), so the 13x9 model was discarded for time-saving purposes.

3.3 Model Training Testing

The next factors tested were the batch size and epoch number in conjunction. As described earlier, you want to have a small batch size to prevent overfitting, but this comes at the cost of training time. The epoch number depends on how well the model is performing during training, and in this case was set up to the point where the training and validation losses and accuracies stopped improving. Figure 6 shows visually how the model performance increases every epoch and how this performance stops increasing at a certain point. The results obtained when running the model with different batch sizes and epochs are in Tables 4 and 5.

For: • All features • 26x17 • ReLu	**Run number**						**Mean**
Training parameters used	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Run 6**	
Batch size 4, 1000 epochs							
Accuracy	0.9011	0.9111	0.8956	0.8944	0.9144	0.9156	0.9054
Loss	n. r.	n. r.	n. r.	n. r.	n. r.	0.24244	n. r.
Batch size 8, 1000 epochs							
Accuracy	0.8944	0.9233	0.9033	0.9144	0.9144	0.9111	0.9102
Loss	n. r.	0.21243	n. r.	n. r.	n. r.	n. r.	n. r.
Batch size 16, 2000 epochs							
Accuracy	0.8756	0.9056	0.9089	0.9089	0.8989	0.9133	0.9019
Loss	n. r.	n. r.	n. r.	n. r.	n. r.	0.24988	n. r.

Table 4. Run results of ML model when using different batch size-epoch number combinations. Note that n.r. means the data was not recorded in that instance. The best runs are highlighted in blue, and the mean results are highlighted in red.

For: • All features • 26x17 • ReLu	**For best run**					
Training parameters used	**Training set**		**Validation set**		**Testing set**	
	Loss	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Accuracy**
Batch size 4, 1000 epochs	0.1966	0.9315	0.39009	0.8556	0.23232	0.9278
Batch size 8, 1000 epochs	0.15641	0.9389	0.35418	0.8778	0.23872	0.9222
Batch size 16, 2000 epochs	0.18536	0.9315	0.39698	0.8722	0.29633	0.9

Table 5. Run results of ML model for the best run, when using different batch size-epoch number combinations.

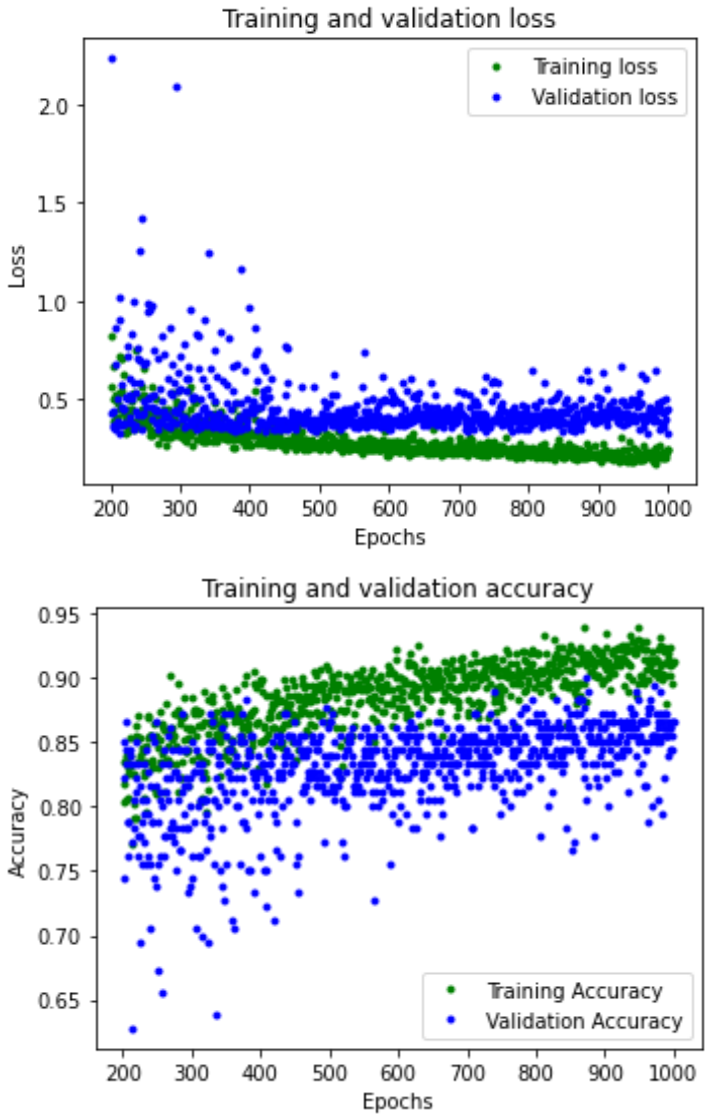


Figure 6. Model performance during training shown as the training and validation set losses and accuracies per epoch.

It is worth noting again that the epoch number was manually set for each batch size up to the point where the model accuracies and losses stopped increasing and decreasing, respectively. Tables 4 and 5 show that the different combinations of parameters used yielded similar results. However, the 'batch size 4, 1000 epochs' case took longer to train; this was because the training time is proportional to the inverse of the batch size times the epoch number and this case gives a larger number for the multiplication than the other two cases (1/4 * 1000 = 250 vs 1/16*2000 = 125 vs 1/8 * 1000 = 125). 'Batch size 8, 1000 epochs' was chosen over 'batch size 16, 2000 epochs' as it gave better training, validation and testing losses and a better testing accuracy in Table 5.

3.4 Model Activation Function Testing

The activation function defines the way in which the weight and bias values inside the neurons are updated or 'activated' during training (20). In this case we tested the model with the 'ReLu' and 'Sigmoid' activation functions (in the input and hidden layers, as the output layer has its own different activation function). The results obtained can be seen in Table 6; Figure 7 shows sketches of these activation functions. It can be seen how ReLu gave significantly better results than Sigmoid, so it was chosen as the main activation function.

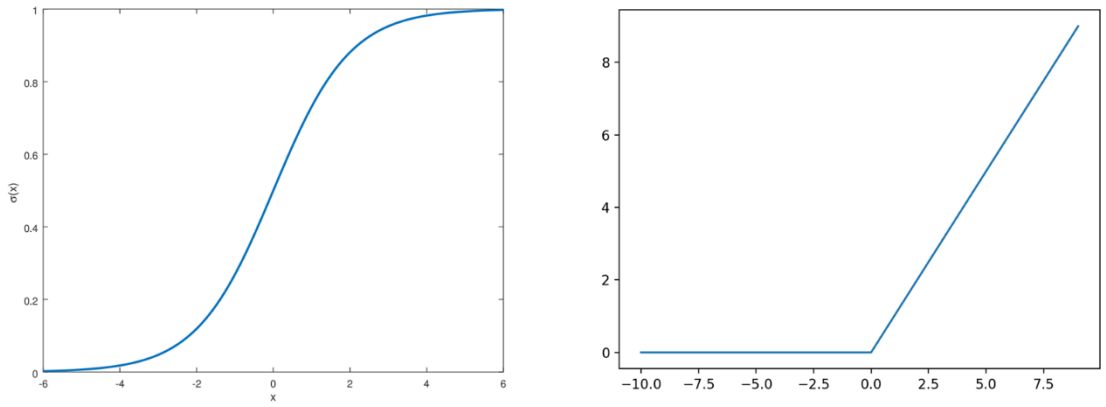


Figure 7. Sketches of the Sigmoid (to the left) and ReLu (to the right) activation functions. Sources: (30, 31).

For: • All features • 26x17 • Batch size 8, 1000 epochs	**Run number**						**Mean**
Activation function used	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Run 6**	
Sigmoid							
Accuracy	0.7767	0.7933	0.7933	0.7978	0.7944	0.7941	0.79165
Loss	n. r.	n. r.	n. r.	n. r.	n. r.	n. r.	n. r.
ReLu							
Accuracy	0.8944	0.9233	0.9033	0.9144	0.9144	0.9111	0.9102
Loss	n. r.	0.21243	n. r.	n. r.	n. r.	n. r.	n. r.

Table 6. Run results of ML model when using different activation functions. Note that n.r. means the data was not recorded in that instance. The best runs are highlighted in blue, and the mean results are highlighted in red.

3.5 TensorFlow Lite Conversion and Quantisation Testing

After the model was compressed into a space-efficient format using the TensorFlow (TF) Lite Converter and further quantised to reduce its size, it was tested against the original TensorFlow model. This was to evaluate if the compression had had any effect on the model accuracy and loss. To do this, the original, TF Lite and quantised TF Lite model predictions for the test set were plotted, the associated mean losses (Categorical Crossentropy) and accuracies (Categorical) calculated, and the memory space of each model measured. The results are shown in Figures 8-10:

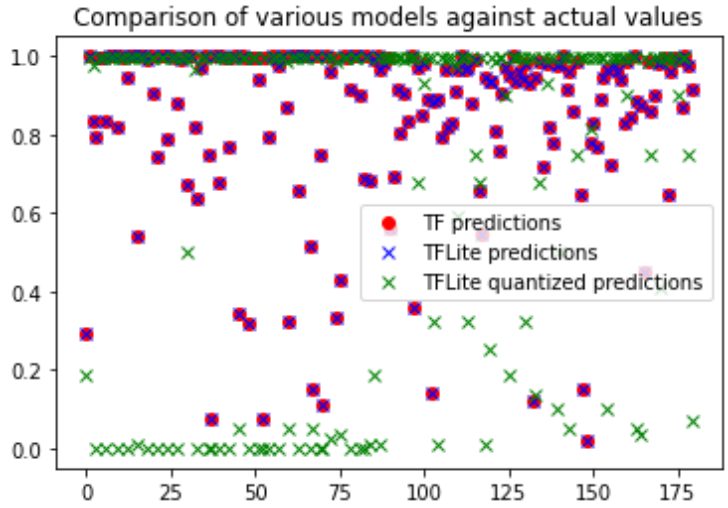


Figure 8. The plotted test set predictions of the original TF, TF Lite and quantised TF Lite models.

Model	Loss	Accuracy
TensorFlow	0.246759	0.911111
TensorFlow Lite	0.246759	0.911111
TensorFlow Lite Quantized	2.648698	0.716667

Figure 9. The test set mean loss and accuracy for the original TF, TF Lite and quantised TF Lite models, as outputted by the Jupyter notebook.

Model	Size	
TensorFlow	41144 bytes	
TensorFlow Lite	4384 bytes	(reduced by 36760 bytes)
TensorFlow Lite Quantized	3488 bytes	(further reduced by 896 bytes)

Figure 10. The size of the original TF, TF Lite and quantised TF Lite models (in bytes), as outputted by the Jupyter notebook.

Figures 8 and 9 show how there is no appreciable change in model performance when converting from TF to TF Lite format. On the other hand, when the ML model was quantised, it suffered a considerable loss of model performance. Since the additional reduction in model size (896 bytes) was not necessary in our case to implement the model to the Arduino Nano 33 BLE, quantisation was discarded. In addition, in Figure 8, it can be observed how most of the model predictions lie above the 80% (or 0.8) accuracy point – Therefore, this number was chosen to be threshold in the Arduino output handler function for detecting if a given prediction score really corresponds to a gesture or not.

3.6 Arduino code Output

The Arduino code correctly performed the feature extraction of the pre-processed pre-defined sEMG signals. This can checked by comparing the output in the Arduino Serial monitor (with the Arduino-extracted features) (see Figure 11) to the MATLAB extracted features (see Figure 12) for the same pre-processed signal. Figures 11 and 12 show that the features extracted by Arduino were identical to those extracted by MATLAB for the same pre-processed signal. In addition, the Arduino code was successful in performing the ML predictions as is displayed in Figure 11.

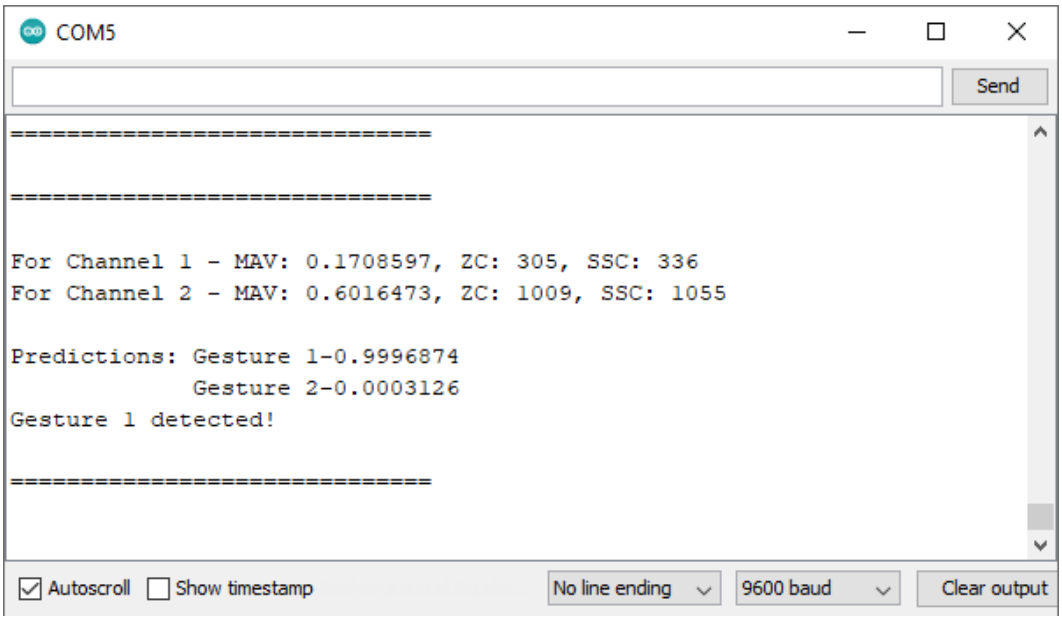


Figure 11. Output in the Serial monitor after uploading the Arduino code (Code 14) to the Arduino Nano 33 BLE.

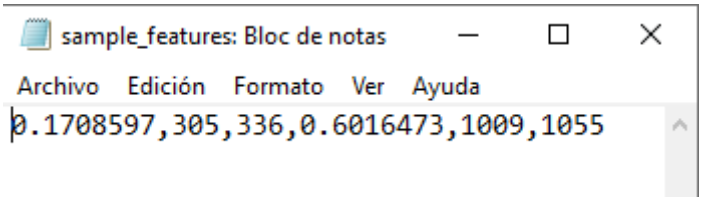


Figure 12. MATLB extracted features for the same pre-processed signal as used by the Arduino code (Code 14).

3. Discussion

The project proved the principle that ML learning can indeed be used to detect 2 different hand gestures from 2-channel sEMG data, and with quite accurate results. A summary of the best model parameters as calculated in the Results section for this specific application is as follows:

- Features fed into the model – MAV, ZC and SSC
- Model architecture – 26x17
- Model training – Batch size 8, 1000 epochs
- Input and hidden layers activation function – ReLu
- Model compression – TF Lite (not quantised)

The final model which used all these factors (before TF Lite conversion) gave a Categorical Crossentropy of 0.23384 and an accuracy of 0.9067 for the whole dataset. Even though these results are quite good, they still fall short compared to some of the accuracies obtained in the sEMG gesture detection studies analysed in (22) (> 95%).

For final model							
Training set		**Validation set**		**Testing set**		**Whole dataset**	
Loss	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Accuracy**
0.19842	0.9185	0.32720	0.8667	0.24676	0.9111	0.23384	0.9067

Table 7. Summary of run results for the final ML model.

Reasons for this inferior performance may be due to the fact that the sEMG signals were recorded from many different subjects. As explained by Li et al. (2021), ML algorithms are only suitable to analyse steady state signals, while sEMG signals are transient in nature (7). This transient state of sEMG occurs not only between subjects but also within the same subject; therefore sEMG analysis by ML models is not ideal (7). In recent years, the application of Deep Learning (DL) for gesture recognition has increased to take hold of the challenging task of inspecting transient sEMG signals (7). DL is a branch of ML in which the model learns by itself (contrary to ML, where the programmer must manually train the model) (7). As DL models can perform feature extraction and model training at the same time, they are more suitable for these kind of tasks (7).

An additional problem with the model was overfitting as can be seen in Figure 6 and Table 7. Despite having this overfitting between the training and validation datasets, the model was kept on trained (for a larger number of epochs) as this meant an increase in training performance while the validation performance stayed almost the same (see Figure 6). In addition, this increase in epoch number did not have an effect on the testing set performance, which displayed similar results to the training set and was actually superior to the validation set (see Table 7).

4. Conclusion

To conclude, this project has demonstrated that it is possible to use a simple ML model for hand gesture recognition using sEMG. Specifically, for sEMG signals coming from the Flexor Capri Ulnaris and Extensor Capri Radialis, Longus and Brevis muscles and to distinguish between a Hook and a Spherical hand grasp. In addition, it has shown the possibility of implementing feature extraction and the ML model into an Arduino 33 BLE microcontroller. For possible future work, one could increase the number of gestures recognised (using the data already available within the utilised database) or the number of databases from which the sEMG signals are retrieved (e.g. the Ninapro database (7)). Also, the implementation of the pre-processing code to Arduino could be done, moving a step closer to detect gestures in real-time with the microcontroller. Furthermore, one could explore the application of Deep Learning to this specific problem to account for the transient nature of sEMG. Finally, it is hoped that this paper has contributed to the development and improvement of new ML methods to increase the available number of gestures in prosthetics to amputees.

5. References

1. Sahu A, Sagar R, Sarkar S, Sagar S. Psychological effects of amputation: A review of studies from India. Industrial Psychiatry Journal [Internet]. Wolters Kluwer -- Medknow Publications; 2016 [cited 2022 May 16]; 25(1):4. Available from: /pmc/articles/PMC5248418/.

2. Kuiken TA, Feuser AE, Barlow AK. Targeted Muscle Reinnervation: A Neural Interface for Artificial Limbs. Targeted Muscle Reinnervation: A Neural Interface for Artificial Limbs. CRC Press; 2013.

3. Esquenazi A, Maitin I, Cruz E. Current Diagnosis & Treatment: Physical Medicine & Rehabilitation (Chapter 27). McGraw Hill [Internet]. McGraw Hill; 2014 [cited 2022 May 16]. Available from: https://accessmedicine.mhmedical.com/content.aspx?bookid=1180&sectionid=70380726.

4. Ziegler-Graham K, MacKenzie EJ, Ephraim PL, Travison TG, Brookmeyer R. Estimating the prevalence of limb loss in the United States: 2005 to 2050. Arch Phys Med Rehabil [Internet]. Arch Phys Med Rehabil; 2008 [cited 2022 May 16]; 89(3):422–9. Available from: https://pubmed.ncbi.nlm.nih.gov/18295618/.

5. Fahrenkopf MP, Adams NS, Kelpin JP, Do VH. Hand Amputations. Eplasty [Internet]. HMP Global; 2018 [cited 2022 May 16]; 18:ic21. Available from: /pmc/articles/PMC6173827/.

6. Elkoura G, Singh K. Handrix: Animating the Human Hand. 2003.

7. Li W, Shi P, Yu H. Gesture Recognition Using Surface Electromyography and Deep Learning for Prostheses Hand: State-of-the-Art, Challenges, and Future. Frontiers in Neuroscience. Frontiers Media S.A.; 2021; 15:259.

8. Meyer TM. Psychological aspects of mutilating hand injuries. Hand Clinics [Internet]. Elsevier; 2003 [cited 2021 Nov 30]; 19(1):41–9. Available from: http://www.hand.theclinics.com/article/S0749071202000562/fulltext.

9. Putter CE de, Selles RW, Polinder S, Panneman MJM, Hovius SER, Beeck EF van. Economic impact of hand and wrist injuries: Health-care costs and productivity costs in a population-based study. Journal of Bone and Joint Surgery - Series A [Internet]. Journal of Bone and Joint Surgery Inc.; 2012 [cited 2022 May 16]; 94(9):e56(1). Available from: https://journals.lww.com/jbjsjournal/Fulltext/2012/05020/Economic_Impact_of_Hand_and_Wrist_Injuries_.14.aspx.

10. Global Artificial Limbs Market Will Reach USD 2,758 Million [Internet]. [cited 2022 May 16]. Available from: https://www.globenewswire.com/news-release/2019/02/18/1733678/0/en/Global-Artificial-Limbs-Market-Will-Reach-USD-2-758-Million-By-2025-Zion-Market-Research.html.

11. bebionic Hand EQD | The most lifelike prosthetic hand [Internet]. [cited 2022 May 16]. Available from: https://www.ottobock.com/en-us/product/8E70.

12. Precision, power, and intelligent motion. Ossur.com [Internet]. [cited 2022 May 16]. Available from: https://www.ossur.com/en-gb/prosthetics/arms/i-limb-quantum.

13. Khokhar ZO, Xiao ZG, Menon C. Surface EMG pattern recognition for real-time control of a wrist exoskeleton. BioMedical Engineering Online [Internet]. BioMed Central; 2010 [cited 2021 Nov 30]; 9(1):1–17. Available from: https://link.springer.com/articles/10.1186/1475-925X-9-41.

14. Cram's Introduction to Surface Electromyography - Eleanor Criswell - Google Books [Internet]. [cited 2021 Nov 30]. Available from: https://books.google.co.uk/books?hl=en&lr=&id=ADYm0TqiDo8C&oi=fnd&pg=PP1&dq=surface+electromyography&ots=RxqfMtNt0P&sig=f4HCshMfY2JTtUIVcX2M4j-_i3Q&redir_esc=y#v=onepage&q=surface%20electromyography&f=false.

15. Disselhorst-Klug C, Williams S. Surface Electromyography Meets Biomechanics: Correct Interpretation of sEMG-Signals in Neuro-Rehabilitation Needs Biomechanical Input. Frontiers in Neurology. Frontiers Media S.A.; 2020; 11:1644.

16. Rehman MZU, Gillani SO, Waris A, Jochumsen M, Niazi IK, Kamavuako EN. Performance of Combined Surface and Intramuscular EMG for Classification of Hand Movements. Annu Int Conf IEEE Eng Med Biol Soc [Internet]. Annu Int Conf IEEE Eng Med Biol Soc; 2018 [cited 2022 May 16]; 2018:5220–3. Available from: https://pubmed.ncbi.nlm.nih.gov/30441515/.

17. Chowdhury RH, Reaz MBI, Mohd Ali MA bin, Bakar AAA, Chellappan K, Chang TG. Surface Electromyography Signal Processing and Classification Techniques. Sensors (Basel) [Internet]. Multidisciplinary Digital Publishing Institute (MDPI); 2013 [cited 2021 Nov 30]; 13(9):12431. Available from: /pmc/articles/PMC3821366/.

18. Electrode placement over the forearm and upper arm muscles. | Download Scientific Diagram [Internet]. [cited 2021 Nov 30]. Available from: https://www.researchgate.net/figure/Electrode-placement-over-the-forearm-and-upper-arm-muscles_fig3_309174529.

19. Ciancio AL, Cordella F, Barone R, Romeo RA, Bellingegni AD, Sacchetti R, et al. Control of prosthetic hands via the peripheral nervous system. Frontiers in Neuroscience. Frontiers Research Foundation; 2016; 10(APR):116.

20. Burkov A. The Hundred-Page Machine Learning Book [Internet]. [cited 2022 May 16]. Available from: http://themlbook.com/wiki/doku.php.

21. Warden P, Situnayake D. TinyML [Internet]. O'Reilly Media; 2019 [cited 2022 May 16]. Available from: https://learning.oreilly.com/library/view/tinyml/9781492052036/.

22. Jaramillo-Yánez A, Benalcázar ME, Mena-Maldonado E. Real-Time Hand Gesture Recognition Using Surface Electromyography and Machine Learning: A Systematic Literature Review. Sensors 2020, Vol. 20, Page 2467 [Internet]. Multidisciplinary Digital Publishing Institute; 2020 [cited 2022 May 16]; 20(9):2467. Available from: https://www.mdpi.com/1424-8220/20/9/2467/htm.

23. Ciancio AL, Cordella F, Barone R, Romeo RA, Bellingegni AD, Sacchetti R, et al. Control of prosthetic hands via the peripheral nervous system. Frontiers in Neuroscience. Frontiers Research Foundation; 2016; 10(APR):116.

24. Yu H, Fan X, Zhao L, Guo X. A novel hand gesture recognition method based on 2-channel sEMG. Technology and Health Care [Internet]. IOS Press; 2018 [cited 2022 May 16]; 26(Suppl 1):205. Available from: /pmc/articles/PMC6004976/.

25. UCI Machine Learning Repository: sEMG for Basic Hand movements Data Set [Internet]. [cited 2022 May 16]. Available from: https://archive.ics.uci.edu/ml/datasets/sEMG+for+Basic+Hand+movements.

26. (PDF) Online EMG Signal Analysis for diagnosis of Neuromuscular diseases by using PCA and PNN. [Internet]. [cited 2022 May 16]. Available from: https://www.researchgate.net/publication/232905752_Online_EMG_Signal_Analysis_for_diagnosis_of_Neuromuscular_diseases_by_using_PCA_and_PNN#pf2.

27. tflite-micro/train_hello_world_model.ipynb at main · tensorflow/tflite-micro · GitHub [Internet]. [cited 2022 May 17]. Available from: https://github.com/tensorflow/tflite-micro/blob/main/tensorflow/lite/micro/examples/hello_world/train/train_hello_world_model.ipynb.

28. How to solve Classification Problems in Deep Learning with Tensorflow & Keras? | by Murat Karakaya | Deep Learning Tutorials with Keras | Medium [Internet]. [cited 2022 May 17]. Available from: https://medium.com/deep-learning-with-keras/how-to-solve-classification-problems-in-deep-learning-with-tensorflow-keras-6e39c5b09501.

29. Deep Learning [Internet]. [cited 2022 May 17]. Available from: https://www.deeplearningbook.org/.

30. The sigmoid function (a.k.a. the logistic function) and its derivative [Internet]. [cited 2022 May 17]. Available from: https://hvidberrrg.github.io/deep_learning/activation_functions/sigmoid_function_and_derivative.html.

31. A Gentle Introduction to the Rectified Linear Unit (ReLU) [Internet]. [cited 2022 May 17]. Available from: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

6. Appendix

Code 1. (below) Main MATLAB file for the feature extraction of sEMG signals. This file calls many other files and associated functions to achieve its purpose.

```
clear; clc;
%This code will extract the time domain features from recorded sEMG
signals and appropriately format them to be exported
%into the main ML code

%This threshold is used in the pre-processing of the signals later on
thr = single(0.0488);

%
%--- Database 1 ---%
%Importing and combining Database 1 data
load('../Database 1/female_1.mat');
```

```
gest1_ch1 = hook_ch1; gest1_ch2 = hook_ch2; gest2_ch1 = spher_ch1;
gest2_ch2 = spher_ch2;

load('../Database 1/female_2.mat');
gest1_ch1 = [gest1_ch1; hook_ch1]; gest1_ch2 = [gest1_ch2; hook_ch2];
gest2_ch1 = [gest2_ch1; spher_ch1]; gest2_ch2 = [gest2_ch2;
spher_ch2];

load('../Database 1/female_3.mat')
gest1_ch1 = [gest1_ch1; hook_ch1]; gest1_ch2 = [gest1_ch2; hook_ch2];
gest2_ch1 = [gest2_ch1; spher_ch1]; gest2_ch2 = [gest2_ch2;
spher_ch2];

load('../Database 1/male_1.mat')
gest1_ch1 = [gest1_ch1; hook_ch1]; gest1_ch2 = [gest1_ch2; hook_ch2];
gest2_ch1 = [gest2_ch1; spher_ch1]; gest2_ch2 = [gest2_ch2;
spher_ch2];

load('../Database 1/male_2.mat')
gest1_ch1 = [gest1_ch1; hook_ch1]; gest1_ch2 = [gest1_ch2; hook_ch2];
gest2_ch1 = [gest2_ch1; spher_ch1]; gest2_ch2 = [gest2_ch2;
spher_ch2];

%Turning data into 'single' data type
gest1_ch1 = single(gest1_ch1); gest1_ch2 = single(gest1_ch2);
gest2_ch1 = single(gest2_ch1); gest2_ch2 = single(gest2_ch2);

%Clearing unused variables
clearvars -except thr gest1_ch1 gest1_ch2 gest2_ch1 gest2_ch2

%Pre-processing samples from Database 1
if size(gest1_ch1,1) ~= size(gest1_ch2,1)   %Gesture 1
    disp('Channel 1 and Channel 2 sample size does not match for
Gesture 1')
else
    [gest1_ch1, wl_gest1_ch1] = pre_processing(gest1_ch1, thr);
    [gest1_ch2, wl_gest1_ch2] = pre_processing(gest1_ch2, thr);
end
if size(gest2_ch1,1) ~= size(gest2_ch2,1)   %Gesture 2
    disp('Channel 1 and Channel 2 sample size does not match for
Gesture 2')
else
    [gest2_ch1, wl_gest2_ch1] = pre_processing(gest2_ch1, thr);
    [gest2_ch2, wl_gest2_ch2] = pre_processing(gest2_ch2, thr);
end

%Extracting features from Database 1
data_gest1 = feature_function(gest1_ch1, gest1_ch2, wl_gest1_ch1,
wl_gest1_ch2);
data_gest2 = feature_function(gest2_ch1, gest2_ch2, wl_gest2_ch1,
wl_gest2_ch2);
%}

%
%--- Database 2 ---%
%Importing and combining Database 2 data
load('../Database 2/male_day_1.mat');
gest1_ch1 = hook_ch1; gest1_ch2 = hook_ch2; gest2_ch1 = spher_ch1;
gest2_ch2 = spher_ch2;
```

```
load('../Database 2/male_day_2.mat');
gest1_ch1 = [gest1_ch1; hook_ch1]; gest1_ch2 = [gest1_ch2; hook_ch2];
gest2_ch1 = [gest2_ch1; spher_ch1]; gest2_ch2 = [gest2_ch2;
spher_ch2];

load('../Database 2/male_day_3.mat');
gest1_ch1 = [gest1_ch1; hook_ch1]; gest1_ch2 = [gest1_ch2; hook_ch2];
gest2_ch1 = [gest2_ch1; spher_ch1]; gest2_ch2 = [gest2_ch2;
spher_ch2];

%Turning data into 'single' data type
gest1_ch1 = single(gest1_ch1); gest1_ch2 = single(gest1_ch2);
gest2_ch1 = single(gest2_ch1); gest2_ch2 = single(gest2_ch2);

%Clearing unused variables
clearvars -except data_gest1 data_gest2 thr gest1_ch1 gest1_ch2
gest2_ch1 gest2_ch2

%Pre-processing samples from Database 2
if size(gest1_ch1,1) ~= size(gest1_ch2,1)    %Gesture 1
    disp('Channel 1 and Channel 2 sample size does not match for
Gesture 1')
else
    [gest1_ch1, wl_gest1_ch1] = pre_processing(gest1_ch1, thr);
    [gest1_ch2, wl_gest1_ch2] = pre_processing(gest1_ch2, thr);
end
if size(gest2_ch1,1) ~= size(gest2_ch2,1)    %Gesture 2
    disp('Channel 1 and Channel 2 sample size does not match for
Gesture 2')
else
    [gest2_ch1, wl_gest2_ch1] = pre_processing(gest2_ch1, thr);
    [gest2_ch2, wl_gest2_ch2] = pre_processing(gest2_ch2, thr);
end

%Extracting features from Database 2
% data_gest1 = feature_function(gest1_ch1, gest1_ch2, wl_gest1_ch1,
wl_gest1_ch2);
% data_gest2 = feature_function(gest2_ch1, gest2_ch2, wl_gest1_ch1,
wl_gest1_ch2);
data_gest1 = [data_gest1; feature_function(gest1_ch1, gest1_ch2,
wl_gest1_ch1, wl_gest1_ch2)];
data_gest2 = [data_gest2; feature_function(gest2_ch1, gest2_ch2,
wl_gest2_ch1, wl_gest2_ch2)];
%}

%-- Mixing the data from all the different datasets equally
data_gest1 = mix_dataset(data_gest1);
data_gest2 = mix_dataset(data_gest2);

%--- Exporting the data ---%
%Exporting the data as .txt files, which will be saved in the current
directory
writematrix(data_gest1,'gest1.txt','Delimiter','space');
writematrix(data_gest2,'gest2.txt','Delimiter','space');

%Exporting data from only one sample, to use on Arduino
index1 = 1;
index2 = 2;
writematrix(gest1_ch1(index1,:),'sample1.txt','Delimiter','comma');
```

```
writematrix(gest1_ch2(index1,:),'sample2.txt','Delimiter','comma');
writematrix(data_gest1(index2,:),'sample_features.txt','Delimiter','co
mma');
writematrix([wl_gest1_ch1(index1),
wl_gest1_ch2(index1)],'sample_wl.txt','Delimiter','comma');
```

Code 2. (below) Main MATLAB file for sEMG signal pre-processing. This file calls associated functions to achieve its purpose.

```
function [gest_ch, wl] = pre_processing(gest_ch, thr)
%This function pre-processes the sEMG signals before their features
are extracted. It does this through offset
%correction and endpoint detection

samples = size(gest_ch,1);      %Number of samples in gest_ch
(corresponding to the number of rows in the matrix)

%Calculating the WL
wl = single(zeros(samples,1));     %Pre-allocating size for speed

for n=1:samples
    %Offset correction - Offset in the signal is removed
    gest_ch(n,:) = gest_ch(n,:) - mean(gest_ch(n,:));

    %Endpoint detection - Main segment of the signal is retrieved
along with its corresponding Waveform Length using
    %the set threshold value
    [gest_ch(n,:),wl(n,1)] = endpoint_detection(gest_ch(n,:), thr);
end

end
```

Code 3. (below) MATLAB function used for sEMG endpoint detection.

```
function [gest_ch,wl] = endpoint_detection(gest_ch, thr)
%This function detects the main segment of the sEMG signal and sets
the rest of the signal to zero. The energy of the
%signal is obtained (by calculating its absolute value) and this
energy wave is then transformed into a step function
%(using the mean values of the wave for each step). The starting and
ending points of the signal are then found using
%different thresholds and the signal is cut at these points

sample_points = length(gest_ch);    %Number of sample points in
gest_ch (corresponding to the length of gest_ch)
quantization_step = 125;        %This value is used for 'quantising'
the signal later on
gest_ch_quantised = zeros(1, sample_points);    %Pre-allocating size
for speed

thr_start = thr*1.875;      %Starting threshold to calculate the
starting point
thr_finish = thr*2.25;      %Ending threshold to calculate the ending
point

%--- Getting the quantised energy signal ---%
```

```
%First half of the signal is quantised using the quantisation_step set
earlier.
%--> Every 125 points, the MAV of these points is calculated and all
of these points are set to this MAV value
for n=1:quantization_step:(1/2)*sample_points
    gest_ch_quantised(n:n+(quantization_step-1)) =
mean(abs(gest_ch(n:n+(quantization_step-1))));
end

%Third quarter of the signal is quantised using the quantisation_step
× 2.
%--> Every 250 points, the MAV of these points is calculated and all
of these points are set to this MAV value
for m=(1/2)*sample_points+1:quantization_step*2:(3/4)*sample_points
    gest_ch_quantised(m:m+(quantization_step*2-1)) =
mean(abs(gest_ch(m:m+(quantization_step*2-1))));
    last_quantisation_point = m;
end

%Last quarter of the signal is quantised all at once
%--> The MAV of the remaining last points is calculated and all of
these points are set to this MAV value.
gest_ch_quantised(last_quantisation_point:sample_points) =
mean(abs(gest_ch(last_quantisation_point:sample_points)));
%}

%--- Endpoint detection ---%
%Starting point detection
start_point = [];
for counter1=1:1:sample_points      %Looping through all of the points
in the sample (from start to end)
    if gest_ch_quantised(counter1) >= thr_start     %If the value of
the quantised energy signal point is equal or larger than the starting
threshold, the starting point is equal to counter1
        start_point = counter1;     %Starting point is stored in
start_point
        break
    end
end

%Ending point detection
finish_point = [];
for counter2=sample_points:-1:1     %Looping through all of the points
in the sample (from end to start)
    if gest_ch_quantised(counter2) >= thr_finish    %If the value of
the quantised energy signal point is equal or larger than the ending
threshold, the ending point is equal to counter2
        finish_point = counter2;    %Ending point is stored in
finish_point
        break
    end
end

%Starting point exceptions
%--> If no starting point is detected using the current starting
threshold, this threshold is lowered and another
%    attempt to find the starting point is carried out.
if isempty(start_point) == 1
    thr_start = thr*1.75;
    for counter1=1:1:sample_points
```

```
        if gest_ch_quantised(counter1) >= thr_start
            start_point = counter1;
            break
        end
    end
end
if isempty(start_point) == 1
    thr_start = thr*1.625;
    for counter1=1:1:sample_points
        if gest_ch_quantised(counter1) >= thr_start
            start_point = counter1;
            break
        end
    end
end
if isempty(start_point) == 1
    thr_start = thr*1.5;
    for counter1=1:1:sample_points
        if gest_ch_quantised(counter1) >= thr_start
            start_point = counter1;
            break
        end
    end
end

%Ending point exceptions
%--> If no ending point is detected using the current ending
threshold, this threshold is lowered to the current
%    starting threshold and another attempt to find the ending point
is carried out.
if isempty(start_point) == 0 && isempty(finish_point) == 1
    thr_finish = thr_start;
    for counter2=sample_points:-1:1
        if gest_ch_quantised(counter2) >= thr_finish
            finish_point = counter2;
            break
        end
    end
end

%--- Preparing endpoints and gest_ch signal for output ---%
if isempty(start_point) == 0 && isempty(finish_point) == 0  %If the
endpoints have been found, the following code is executed

    %Decreasing and increasing the starting and ending points,
respectively, to account for some of the error associated
    %with quantisation (in order to not loose important information in
the signal)
    start_bias = round(0.125*quantization_step);        %Starting
point is decreased by 0.125 × quantisation_step
    finish_bias = round(1.125*quantization_step);       %Ending point
is increased by 1.125 × quantisation_step
    start_point = start_point-start_bias;
    finish_point = finish_point+finish_bias;
    if start_point < 1      %If after the adjustment, the starting
point is smaller than 1, it is set equal to 1
        start_point = 1;
    end
    if finish_point > sample_points     %If after the adjustment, the
ending point is greater than the total number of sample points, it is
set equal to the last sample point
```

```
        finish_point = sample_points;
    end

    %Zeroing the points in the signal outside the endpoints
    gest_ch(1:start_point) = 0;
    gest_ch(finish_point:sample_points) = 0;
    wl = finish_point - (start_point-1);    %Storing the Waveform
Length [number of sample points] in wl

elseif isempty(start_point) == 1 && isempty(finish_point) == 1  %If
the endpoints have not been found, the following code is executed
    start_point = 1;        %Setting the starting point equal to 1
    finish_point = sample_points;   %Setting the ending point equal to
the last sample point

    %Zeroing all the points in the signal
    gest_ch = zeros(1, sample_points);
    wl = 0;     %Storing the Waveform Length [number of sample points]
in wl
end

% To output the following data, along with the modified 'gest_ch'
signal and WL, uncomment the following code
%{
wl = [wl start_point finish_point thr_start thr_finish];
gest_ch = [gest_ch; gest_ch_quantised];
%}
end
```

Code 4. (below) MATLAB code used to find the mean minimal quantised energy value in the sEMG signals of Database 2. This file calls associated functions to achieve its purpose.

```
%This code retrieves the average minumum value of the quantised energy
signals for all the signals in a single database
clear;

%--- Database 2 ---%
%Importing and combining Database 2 data
load('../Database 2/male_day_3.mat');
gest1_ch1 = hook_ch1; gest1_ch2 = hook_ch2; gest2_ch1 = spher_ch1;
gest2_ch2 = spher_ch2;

load('../Database 2/male_day_2.mat');
gest1_ch1 = [gest1_ch1; hook_ch1]; gest1_ch2 = [gest1_ch2; hook_ch2];
gest2_ch1 = [gest2_ch1; spher_ch1]; gest2_ch2 = [gest2_ch2;
spher_ch2];

load('../Database 2/male_day_3.mat');
gest1_ch1 = [gest1_ch1; hook_ch1]; gest1_ch2 = [gest1_ch2; hook_ch2];
gest2_ch1 = [gest2_ch1; spher_ch1]; gest2_ch2 = [gest2_ch2;
spher_ch2];

min_gest_ch1 = zeros(size(gest1_ch1,1),2);  %Pre-allocating size for
speed
min_gest_ch2 = zeros(size(gest1_ch2,1),2);  %Pre-allocating size for
speed
```

```
for n=1:size(gest1_ch1,1)
    %Channel 1
    gest1_ch1(n,:) = gest1_ch1(n,:) - mean(gest1_ch1(n,:));    %Offset
correction
    min_gest_ch1(n,1) = find_minimum(gest1_ch1(n,:));   %Storing the
minimum quantised energy value of Gesture 1 Channel 1 in the first
instance of min_gest_ch1
    gest2_ch1(n,:) = gest2_ch1(n,:) - mean(gest2_ch1(n,:));    %Offset
correction
    min_gest_ch1(n,2) = find_minimum(gest2_ch1(n,:));   %Storing the
minimum quantised energy value of Gesture 2 Channel 1 in the second
instance of min_gest_ch1

    %Channel 2
    gest1_ch2(n,:) = gest1_ch2(n,:) - mean(gest1_ch2(n,:));    %Offset
correction
    min_gest_ch2(n,1) = find_minimum(gest1_ch2(n,:));   %Storing the
minimum quantised energy value of Gesture 2 Channel 1 in the first
instance of min_gest_ch2
    gest2_ch2(n,:) = gest2_ch2(n,:) - mean(gest2_ch2(n,:));    %Offset
correction
    min_gest_ch2(n,2) = find_minimum(gest2_ch2(n,:));   %Storing the
minimum quantised energy value of Gesture 2 Channel 2 in the second
instance of min_gest_ch2
end

mean(mean(min_gest_ch1))
mean(mean(min_gest_ch2))

thr = (mean(mean(min_gest_ch1)) + mean(mean(min_gest_ch2))) / 2
```

Code 5. (below) MATLAB function called by Code 4.

```
function [min_gest_ch] = find_minimum(gest_ch)
%This function calculates the minumum value of a signal after being
quantised

sample_points = length(gest_ch);    %Number of sample points in
gest_ch (corresponding to the length of gest_ch)
quantization_step = 125;        %This value is used for 'quantising'
the signal later on
gest_ch_quantised = zeros(1, sample_points);    %Pre-allocating size
for speed;

%--- Getting the quantised energy signal ---%
%First half of the signal is quantised using the quantisation_step set
earlier.
%--> Every 125 points, the MAV of these points is calculated and all
of these points are set to this MAV value
for n=1:quantization_step:(1/2)*sample_points
    gest_ch_quantised(n:n+(quantization_step-1)) =
mean(abs(gest_ch(n:n+(quantization_step-1))));
end

%Third quarter of the signal is quantised using the quantisation_step
× 2.
%--> Every 250 points, the MAV of these points is calculated and all
of these points are set to this MAV value
```

```
for m=(1/2)*sample_points+1:quantization_step*2:(3/4)*sample_points
    gest_ch_quantised(m:m+(quantization_step*2-1)) =
mean(abs(gest_ch(m:m+(quantization_step*2-1))));
    last_quantisation_point = m;
end

%Last quarter of the signal is quantised all at once
%--> The MAV of the remaining last points is calculated and all of
these points are set to this MAV value.
gest_ch_quantised(last_quantisation_point:sample_points) =
mean(abs(gest_ch(last_quantisation_point:sample_points)));

%Retrieving minimum value
min_gest_ch = min(gest_ch_quantised);
end
```

Code 6. (below) MATLAB code used to visually inspect the sEMG quantised energy signals to find the multiplication factors for the threshold in Code 3. This file calls associated functions to achieve its purpose.

```
%This code displays graphically the process of signal endpoint
detection
clear;
load('../Database 2/male_day_3.mat')    %Loading dataset (each row in
the matrix corresponds to the data for a sample)
gest_ch_complete = hook_ch1;    %Choosing the gesture and channel to
analyse

thr = 0.0488;   %Setting the threshold to be used for endpoint
detection

Fs = 500;       %Sampling frequency [Hz]
t = 1/Fs;       %Time step [s]

for n=1:size(gest_ch_complete,1)    %Looping through all of the
samples in the dataset and plotting the respective graphs
    clf;
    gest_ch = gest_ch_complete(n,:);
    gest_ch = gest_ch - mean(gest_ch);    %Offset correction
    lines_thr = max(gest_ch)/2;     %lines_thr is used for plotting
later on

    %Plotting the original signal
    figure(1);
    subplot(2,2,1);
    plot(gest_ch);
    title('Original signal');

    %Doing endpoint detection and retrieving all the corresponding
relevant variables/values
    [gest_ch, data] = endpoint_detection(gest_ch, thr);
    start_point = data(2);
    finish_point = data(3);
    thr_start = data(4);
    thr_finish = data(5);
    wl = data(1);
    gest_ch_quantised = gest_ch(2,:);
    gest_ch = gest_ch(1,:);
```

```
    %Plotting the quantised energy signal along with the calculated
endpoints and thresholds
    subplot(2,2,2);
    hold on;
    plot(gest_ch_quantised,'k');
    plot(repelem(start_point,3),linspace(0,thr*3,3),'g');
    plot(repelem(finish_point,3),linspace(0,thr*3,3),'r');
    plot(1:length(gest_ch),repelem(thr_start,length(gest_ch)),'c');
    plot(1:length(gest_ch),repelem(thr_finish,length(gest_ch)),'b');
    hold off;
    legend('Quantised signal','Starting point','Ending
point','Starting threshold','Ending threshold');
    title('Quantised energy signal');

    %Plotting the signal after endpoint detection along with the
calculted endpoints
    subplot(2,2,3);
    hold on;
    plot(gest_ch);
    plot(repelem(start_point,3),linspace(-lines_thr,lines_thr,3),'g');
    plot(repelem(finish_point,3),linspace(-
lines_thr,lines_thr,3),'r');
    hold off;
    legend('Signal','Starting point','Ending point');
    title('Signal after endpoint detection');
    fprintf('%i ',n);
    input('Continue?');
end
```

Code 7. (below) Main MATLAB file for sEMG signal feature extraction. This file calls associated functions to achieve its purpose.

```
function [data_gest] = feature_function(gest_ch1, gest_ch2,
wl_gest_ch1, wl_gest_ch2)
%This function extracts the features for all of the samples in the
data set (using the pre-defined functions) and puts
%them together into a matrix (where each row of the matrix contains
the features for each sample)

%--- Time domain features ---%
%Feature 1 - Mean Absolute Value (MAV)
mav_gest_ch1 = mean_absolute_value(gest_ch1, wl_gest_ch1);
mav_gest_ch2 = mean_absolute_value(gest_ch2, wl_gest_ch2);

%Feature 2 - Zero Crossings (ZC)
zc_gest_ch1 = zero_crossings(gest_ch1);
zc_gest_ch2 = zero_crossings(gest_ch2);

%Feature 3 - Slope Sign Changes (SSC)
ssc_gest_ch1 = slope_sign_changes(gest_ch1);
ssc_gest_ch2 = slope_sign_changes(gest_ch2);

%Feature 4 - Waveform Length (WL) (in seconds)
wl_gest_ch1_s = waveform_length(wl_gest_ch1);
wl_gest_ch2_s = waveform_length(wl_gest_ch2);
```

```
%--- Combining the data ---%
%Combining the data from each channel
data_gest_ch1 = [mav_gest_ch1 zc_gest_ch1 ssc_gest_ch1 wl_gest_ch1_s];
data_gest_ch2 = [mav_gest_ch2 zc_gest_ch2 ssc_gest_ch2 wl_gest_ch2_s];

%Combining the Channel 1 and Channel 2 data into one
data_gest = [data_gest_ch1 data_gest_ch2];
end
```

Code 8. (below) MATLAB function used for retrieving sEMG's Mean Absolute Value.

```
function [mav] = mean_absolute_value(gest_ch, wl_gest_ch)
%This function calculates the Mean Absolute Value (MAV) of the samples

if size(gest_ch,1) ~= length(wl_gest_ch)
    disp('Channel and Waveform Length arrays are not the same size')
else
    samples = size(gest_ch,1);      %Number of samples in gest_ch
(corresponding to the number of rows in the matrix)
end

%Calculating the MAV
mav = single(zeros(samples,1));     %Pre-allocating size for speed
for n=1:samples             %Looping through all of the samples and
storing their MAV in mav
    if wl_gest_ch(n) ~= 0
        sum = 0;
        for m=1:length(gest_ch(n,:))
            sum = sum + abs(gest_ch(n,m));      %Getting the sum of
the absolute values of all the points in the signal
        end
        mav(n,1) = sum / wl_gest_ch(n);     %Dividing by the number of
sample points to get the MAV
    else
        mav(n,1) = 0;
    end
end
end

% Using MATLAB's sum function to get the sum of the array - Gives
different values
%mav(n,1) = sum(abs(gest_ch(n,:))) / wl_gest_ch(n);
```

Code 9. (below) MATLAB function used for retrieving sEMG's Number of Zero Crossings.

```
function [zc] = zero_crossings(gest_ch)
%This function calculates the Zero Crossings (ZC) of the samples (i.e.
how many times the signal crosses the time axis)

samples = size(gest_ch,1);      %Number of samples in gest_ch
(corresponding to the number of rows in the matrix)

%Calculating the ZC
zc = single(zeros(samples,1));     %Pre-allocating size for speed
for n=1:samples            %Looping through all of the samples and
storing their ZC value in zc
```

```
    signs = sign(gest_ch(n,:));    %Getting the sign of each sample
point
    previous = signs(1);       %previous and counter are used in the
for loop to get the ZC
    counter = 0;
    for m=2:length(signs)      %Looping through signs and counting the
number of times the values change
        current = signs(m);
        if previous ~= current
            counter = counter + 1;      %Storing the ZC in counter
        end
        previous = signs(m);
    end
    zc(n,1) = counter;
end
end
```

Code 10. (below) MATLAB function used for retrieving sEMG's Slope Sign Changes.

```
function [ssc] = slope_sign_changes(gest_ch)
%This function calculates the Slope Sign Changes (SSC) of the samples
(i.e. how many times the gradient of the signal
%changes between 0, +ve and -ve)

samples = size(gest_ch,1);      %Number of samples in gest_ch
(corresponding to the number of rows in the matrix)

%Calculating the SSC
ssc = single(zeros(samples,1));     %Pre-allocating size for speed
for n=1:samples             %Looping through all of the samples and
storing their SSC values in ssc
    gradient_signs = sign(gradient(gest_ch(n,:)));    %Getting the
sign of the gradient for each sample point
    previous = gradient_signs(1);       %previous and counter are used
in the for loop to get the SSC
    counter = 0;
    for m=2:length(gradient_signs)      %Looping through
gradient_signs and counting the number of times the values change
        current = gradient_signs(m);
        if previous ~= current
            counter = counter + 1;      %Storing the SSC in counter
        end
        previous = gradient_signs(m);
    end
    ssc(n,1) = counter;
end
end

% Using diff function to get the gradient by forward differences
[dx(i)= x(i+1)-x(i)] - Gives different values
%gradient_signs = sign(diff(gest_ch(n,:)));    %Getting the sign of
the gradient for each sample point
```

Code 11. (below) MATLAB function used for retrieving sEMG's Waveform Length [s].

```
function [wl] = waveform_length(wl_gest_ch)
%This function calculates the Waveform Length (WL) of the samples
(i.e. the time width of the signal) in seconds
```

```
samples = length(wl_gest_ch);      %Number of samples in wl_gest_ch
(corresponding to the number of rows in the matrix)
fs = 500;       %Sampling frequency [Hz]
t = 1/fs;       %Time step [s]

%Calculating the WL (in seconds)
wl = single(zeros(samples,1));     %Pre-allocating size for speed
for n=1:samples            %Looping through all of the samples and
storing their WL (in seconds) in wl
    wl(n,1) = wl_gest_ch(n) * t;
end
end
```

Code 12. (below) MATLAB function used for equal mixing of sEMG sample data.

```
function [data_gest_mixed] = mix_dataset(data_gest)
%This function shuffles the features from the samples of each
person/day equally

%Sanity check to see the numbers are mixed correctly
%(uncomment to see the original position of the samples as a last
column in the matrix)
%data_gest = [data_gest, (1:size(data_gest,1))'];

data_gest_mixed = single(zeros(size(data_gest,1),size(data_gest,2)));
%Matrix with the shuffled features. Pre-allocating size for speed

counter1 = 1;   %counter1 is used to fill the first 14 instances of
the shuffled feature matrix with each loop iteration later on
counter2 = 1;   %counter2 is used to track the cummulative number of
loop iterations in order to determine with value to fill in the 15th
instance of the shuffled feature matrix with each loop iteration later
on
counter3 = 1;   %counter3, counter4 and counter5 are used to fill in
the 15th instance of the shuffled feature matrix with each loop
iteration later on
counter4 = 1;
counter5 = 1;

%With each loop iteration, 15 instances of the shuffled feature matrix
are defined
for n=1:30      %30*15 = 450
    %Filling the first 14 instances
    data_gest_mixed(counter1,:) = data_gest(n,:);    %Dataset 1,
Female 1
    data_gest_mixed(counter1+1,:) = data_gest(n+150,:);    %Dataset 2,
Male Day 1
    data_gest_mixed(counter1+2,:) = data_gest(n+250,:);    %Dataset 2,
Male Day 2
    data_gest_mixed(counter1+3,:) = data_gest(n+30,:);    %Dataset 1,
Female 2
    data_gest_mixed(counter1+4,:) = data_gest(n+350,:);    %Dataset 2,
Male Day 3
    data_gest_mixed(counter1+5,:) = data_gest(n+180,:);    %Dataset 2,
Male Day 1
    data_gest_mixed(counter1+6,:) = data_gest(n+60,:);    %Dataset 1,
Female 3
```

```
    data_gest_mixed(counter1+7,:) = data_gest(n+280,:);     %Dataset 2,
Male Day 2
    data_gest_mixed(counter1+8,:) = data_gest(n+380,:);     %Dataset 2,
Male Day 3
    data_gest_mixed(counter1+9,:) = data_gest(n+90,:);     %Dataset 1,
Male 1
    data_gest_mixed(counter1+10,:) = data_gest(n+210,:);     %Dataset
2, Male Day 1
    data_gest_mixed(counter1+11,:) = data_gest(n+310,:);     %Dataset
2, Male Day 2
    data_gest_mixed(counter1+12,:) = data_gest(n+120,:);     %Dataset
1, Male 2
    data_gest_mixed(counter1+13,:) = data_gest(n+410,:);     %Dataset
2, Male Day 3

    %Filling the 15th instance
    if rem(counter2,3) == 0 && counter3 <= 10    %If counter2 is a
multiple of 3 this code is executed
        data_gest_mixed(counter1+14,:) = data_gest(counter3+240,:);
%Dataset 2, Male Day 1
        counter3 = counter3 + 1;
    elseif rem(counter2,2) == 0 && counter4 <= 10    %If counter2 is a
multiple of 2 this code is executed
        data_gest_mixed(counter1+14,:) = data_gest(counter4+340,:);
%Dataset 2, Male Day 2
        counter4 = counter4 + 1;
    elseif counter5 <= 10    %If counter2 is neither a multiple of 2 or
3 this code is executed
        data_gest_mixed(counter1+14,:) = data_gest(counter5+440,:);
%Dataset 2, Male Day 3
        counter5 = counter5 + 1;
    end

    counter1 = counter1 + 15;
    counter2 = counter2 + 1;
end
end
```

Code 13. (below) Python code from the Jupyter notebook used to build, train and export the ML model. Note that the Jupyter notebook was build from one of the given ML examples of TensorFlow Lite (can be accessed here: https://github.com/tensorflow/tflite-micro/blob/main/tensorflow/lite/micro/examples/hello_world/train/train_hello_world_model.ipynb) meaning that some of the code/comments may be exactly the same as in the example.

```
1 # -*- coding: utf-8 -*-
2 """Final.ipynb
3
4 Automatically generated by Colaboratory.
5
6 Original file is located at
7 https://colab.research.google.com/drive/1znmGRG4JW81ADdCLYPzFru6a4pT622Iw
8
9 # Train of a TensorFlow Lite model for Microcontrollers
```

```
10
11 This notebook demonstrates the process of training a model using TensorFlow and
converting it for use with TensorFlow Lite for microcontrollers. The application is
gesture recognition using features extracted from sEMG signals.
12
13 ## Configure Defaults
14 """
15
16 # Define paths to model files
17 import os
18 MODELS_DIR = 'models/'
19 if not os.path.exists(MODELS_DIR):
20 os.mkdir(MODELS_DIR)
21 MODEL_TF = MODELS_DIR + 'model'
22 MODEL_NO_QUANT_TFLITE = MODELS_DIR + 'model_no_quant.tflite'
23 MODEL_TFLITE = MODELS_DIR + 'model.tflite'
24 MODEL_TFLITE_MICRO = MODELS_DIR + 'model.cc'
25
26 # Defining path to save the entire keras model to a HDF5 file (for size
27 # comparison with the TensorFlow Lite models later)
28 MODEL_TF_H5 = MODELS_DIR + 'model.h5'
29
30 """## Setup Environment
31
32 Install Dependencies:
33 """
34
35 ! pip install tensorflow==2.4.0
36
37 """Import Dependencies:"""
38
39 # TensorFlow is an open source machine learning library
40 import tensorflow as tf
41
42 # Keras is TensorFlow's high-level API for deep learning
43 from tensorflow import keras
44 # Numpy is a math library
45 import numpy as np
46 # Pandas is a data manipulation library
47 import pandas as pd
48 # Matplotlib is a graphing library
49 import matplotlib.pyplot as plt
50 # Math is Python's math library
51 import math
52
53 """## Dataset
54
55 ### 1. Import and Process Data
56
57 In order to train a ML model, it is necessary to have a dataset. A dataset is a
collection of labelled feature vectors, where a feature is a value that describes a
sample of data somehow. In our case, the features are the MAV, ZC, SSC and WL of the
```

```
Channels 1 and 2 data. A label is an informative value (or values) used to identify
a particular feature vector. In our case, we use one-hot encoding for labelling,
where each bit represents the probability from 0 to 1 the features belong to a
particular gesture. For example, for Gesture 1 samples, there would be a probability
of 1 in the first bit and a probability of 0 in the remaining bits, so the label
would be [1, 0].

The code in the following cell will import the feature vectors of each sample
(obtained from MATLAB), assign them a label ([1, 0] for Gesture 1 and [0, 1] for
Gesture 2), and then mix the Gesture 1 and 2 datasets together. After that, the code
splits the dataset into `x` (features) and `y` (labels) to appropriately feed it
into the ML algorithm.
"""

#Importing .txt files containing the combined Channel 1 and Channel 2 sEMG
#features for each gesture; each row corresponding to a different sample
data_gest1 = np.loadtxt('gest1.txt')
data_gest2 = np.loadtxt('gest2.txt')

#Creating labels for each gesture using one-hot encoding. Each gesture has a
#label of 1 for its corresponding column and a label of 0 for all other
#columns. Make sure the number of columns in each label is the same.
label_gest1 = np.tile(np.array([1,0], dtype=float), (len(data_gest1), 1))
label_gest2 = np.tile(np.array([0,1], dtype=float), (len(data_gest2), 1))

#Combining data from each gesture with its corresponding label
gest1 = np.concatenate((data_gest1, label_gest1), axis=1)
gest2 = np.concatenate((data_gest2, label_gest2), axis=1)

#Mixing the Gestures 1 and 2 data
if len(gest1) != len(gest2): #Checking gest1 and gest2 have the same length
print("Data for gestures 1 and 2 isn't the same size")
else:
counter = 0 #Counter to be used in the for loop
gestCombined = np.zeros((2*len(gest1),len(gest1[0]))) #Pre-allocating size for
speed
for n in range(0,len(gest1)): #Combining data into array gestCombined
gestCombined[counter] = gest1[n]
counter = counter + 1
gestCombined[counter] = gest2[n]
counter = counter + 1

#Getting x and y values to feed into the algorithm. x has our features and y
#has the corresponding labels
gest_num = len(label_gest1[0]) #Number of gestures
x_values = np.zeros((len(gestCombined),len(gestCombined[0])-gest_num))
#Pre-allocating size for speed
y_values = np.zeros((len(gestCombined),gest_num)) #Pre-allocating size for speed
for k in range(0,len(gestCombined)):
x_values[k] = gestCombined[k][0:len(gestCombined[n])-gest_num]
y_values[k] = gestCombined[k][len(gestCombined[n])-gest_num:len(gestCombined[n])]
#Finally converting y to intergers
```

```
y_values = y_values.astype(int)

"""### 2. Split the Data

To evaluate the accuracy of the model, it is necessary to compare its predictions to
real data and check how well they match up. This evaluation happens during training
(where it is referred to as validation) and after training (referred to as testing).
It is important in both cases to use fresh data that was not already used to train
the model.

The data is split as follows:
1. Training: 60%
2. Validation: 20%
3. Testing: 20%

"""

# Number of sample datapoints
samples = len(x_values)

# We'll use 60% of our data for training and 20% for testing. The remaining 20%
# will be used for validation. Calculating the indices of each section
train_split = int(0.6 * samples)
test_split = int(0.2 * samples + train_split)

# Choping x and y data into three parts
x_train = np.float32(x_values[0:train_split])
x_validate = np.float32(x_values[train_split:test_split])
x_test = np.float32(x_values[test_split:len(x_values)])

y_train = np.float32(y_values[0:train_split])
y_validate = np.float32(y_values[train_split:test_split])
y_test = np.float32(y_values[test_split:len(y_values)])

type(x_train[1][1])

"""## Training

### 1. Design the Model

We build a neural network model that takes `x` as an input value and uses it to
predict the probability scores for each gesture (i.e. the `y`). This type of problem
is called a regression, and the algorithm uses the layers of neurons to attempt to
learn any patterns underlying the training data, so it can make predictions.

There are 3 layers of neurons. The first layer takes an input vector of length 6
(the `x` feature vector) and runs it through 26 neurons. Based on this input, each
neuron will become activated to a certain degree based on its internal state (its
_weight_ and _bias_ values). A neuron's degree of activation is expressed as a number.

The activation numbers from the first layer are then fed as inputs to the second
layer, 17 neurons long, which applies its own _weights_ and _biases_ to these inputs
to calculate its own activation. The activation numbers from the second layer are
```

```
then fed as inputs to the third layer, and the process repeats until the final layer
is reached. The last layer outputs a vector of length 2 (the `y` vector of
probabilities).
138
139 The code in the following cell defines the model using
[Keras](https://www.tensorflow.org/guide/keras), TensorFlow's high-level API for
creating deep learning networks. Once the network is defined, it is compiled with
Categorical Crossentropy as the loss parameter and (Categorical) Accuracy as the
performance metric. These parameters are normally used to measure the error in
multi-class classification problems which use one-hot encoding, like it is our case
([How to solve Classification Problems in Deep Learning with Tensorflow &
Keras?](https://medium.com/deep-learning-with-keras/how-to-solve-classification-proble
ms-in-deep-learning-with-tensorflow-keras-6e39c5b09501)).
140 """
141
142 #--- Creating the model architecture using Keras ---#
143 # Defining the model
144 model = tf.keras.Sequential()
145
146 # Adding the first layer. The neurons decide whether to activate based on the
147 # 'relu' activation function.
148 model.add(keras.layers.Dense(26, activation='relu', input_shape=(6,)))
149
150 # Including additional layers. These layers will help the network learn more
151 # complex representations
152 model.add(keras.layers.Dense(17, activation='relu'))
153
154 # Incorporating the final layer, 2 neurons long, in order to output a vector of
155 # length 2. The activation function used is 'softmax'
156 model.add(keras.layers.Dense(2, activation='softmax'))
157
158 # Compiling the model using the standard 'adam' optimizer, the
159 # 'categorical_crossentropy' loss function, and the 'accuracy' metric for
160 # regression.
161 model.compile(optimizer='adam', loss="categorical_crossentropy", metrics=["accuracy"])
162
163 # Printing some summary information about the model's architecture
164 model.summary()
165
166 """### 2. Train the Model
167 Once the model is defined, we can use our data to train it. Training involves
passing an `x` feature vector into the neural network, checking how far the
network's output deviates from the expected `y` value, and adjusting the neurons'
_weights_ and _biases_ so that the output is more likely to be correct the next time.
168
169 Training runs this process on the full dataset multiple times, and each full
run-through is known as an epoch. During each epoch, data is run through the network
in multiple batches. Each batch involves several pieces of data, which are passed
into the network, producing output values. These outputs' correctness is measured in
aggregate and the network's _weights_ and _biases_ are adjusted accordingly, once
per batch.
170
```

```
The code in the following cell uses the `x` and `y` from the training data to train
the model. It runs for 1000 epochs, with 8 pieces of data in each batch. The
validation data is also passed in for validation.

"""

# Train the model on our training data while validating on our validation set
history = model.fit(x_train, y_train, epochs=1000, batch_size=8,
validation_data=(x_validate, y_validate))

# Save the model to disk
model.save(MODEL_TF) # SavedModel format (to create the TensorFlow Lite models)
model.save(MODEL_TF_H5) # HDF5 format (to compare with the TensorFlow Lite models
size)

"""### 3. Plot Performance Metrics (Categorical Crossentropy and Categorical Accuracy)

The code in the following cell draws a graph of the loss (Categorical Crossentropy)
and the performance metric (Categorical Accuracy) during training and validation
(i.e. the distance between the predicted and actual values).
"""

# Calculate and print the loss on the training dataset
train_loss, train_accuracy = model.evaluate(x_train, y_train, verbose=0)
print('\033[1m' + 'For training set:' + '\033[0m' + '\tLoss =',format(train_loss,
".5f"),'\tAccuracy =',format(train_accuracy, ".4f"),'\n')

# Calculate and print the loss on the validation dataset
val_loss, val_accuracy = model.evaluate(x_validate, y_validate, verbose=0)
print('\033[1m' + 'For validation set:' + '\033[0m' + '\tLoss =',format(val_loss,
".5f"),'\tAccuracy =',format(val_accuracy, ".4f"),'\n')

#--- Plotting the graphs ---#
# Exclude the first few epochs so the graph is easier to read
SKIP = 200

# Plotting the loss
train_loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(train_loss) + 1)

plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)

plt.plot(epochs[SKIP:], train_loss[SKIP:], 'g.', label='Training loss')
plt.plot(epochs[SKIP:], val_loss[SKIP:], 'b.', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

# Plotting the accuracy
```

```
train_accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']

plt.subplot(1, 2, 2)

plt.plot(epochs[SKIP:], train_accuracy[SKIP:], 'g.', label='Training Accuracy')
plt.plot(epochs[SKIP:], val_accuracy[SKIP:], 'b.', label='Validation Accuracy')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()

"""### 4. Plot Predicted vs Actual values

In the following cells, the metrics of the test set are measured and the predicted and actual values of the whole dataset are plotted for comparison.
"""

#Function to prepare y values for the plot. It separates the Gesture 1
#labels from the Gesture 2 labels as they are mixed in vector y.
def prepareY4Plot(y):
counter = -1
y_gest1 = np.zeros((int(len(y)/gest_num),gest_num))
y_gest2 = np.zeros((int(len(y)/gest_num),gest_num))
for n in range(0,int(len(y)/gest_num)):
counter = counter + 1
y_gest1[n] = y[counter]
counter = counter + 1
y_gest2[n] = y[counter]
return y_gest1, y_gest2

# Calculate and print the loss on the test dataset
test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=0)
print('\033[1m' + 'For test set:' + '\033[0m' + '\tLoss =',format(test_loss, ".5f"),'\tAccuracy =',format(test_accuracy, ".4f"),'\n')

# Make predictions based on the whole dataset
y_dataset_pred = model.predict(x_values)

# Preparing variables for the plot
y_dataset_gest1, y_dataset_gest2 = prepareY4Plot(y_values)
y_dataset_pred_gest1, y_dataset_pred_gest2 = prepareY4Plot(y_dataset_pred)

# Graph the predictions against the actual values
x_plot_gest1 = range(0,int(len(x_values)/2))
x_plot_gest2 = range(int(len(x_values)/2),len(x_values))
plt.clf()
plt.title('Comparison of predictions and actual values for the whole dataset')
#plt.plot(x_plot_gest1, y_dataset_gest1[:,0], 'bo', label='Actual - Gesture 1')
plt.plot(x_plot_gest1, y_dataset_pred_gest1[:,0], 'bx', label='Predicted - Gesture 1')
```

```
#plt.plot(x_plot_gest2, y_dataset_gest2[:,1], 'ro', label='Actual - Gesture 2')
plt.plot(x_plot_gest2, y_dataset_pred_gest2[:,1], 'rx', label='Predicted - Gesture 2')
plt.legend()
plt.show()

# Calculate and print the loss on the whole dataset
dataset_loss, dataset_accuracy = model.evaluate(x_values, y_values, verbose=0)
print('\n','\033[1m' + 'For whole dataset:' + '\033[0m' + ' Loss
=',format(dataset_loss, ".5f"),' Accuracy =',format(dataset_accuracy, ".4f"),'\n')

"""## Generate a TensorFlow Lite Model

### 1. Generate Models with or without Quantization
Now that we have a model, we will use the [TensorFlow Lite
Converter](https://www.tensorflow.org/lite/convert) to convert it into a special,
space-efficient format for use on memory-constrained devices (such as
microcrocontrollers). Since the model is going to be deployed on a microcontroller,
you want it to be as tiny as possible.

One additional technique for reducing the size of the model is called
[quantization](https://www.tensorflow.org/lite/performance/post_training_quantization)
. It reduces the precision of the model's _weights_, and possibly the activations
(output of each layer) as well, which saves memory, often without much impact on
accuracy. Quantized models also run faster, since the calculations required are
simpler.

In the following cell, the model will be converted to TensorFlow Lite format twice,
once with quantization and once without.
"""

# Convert the model to the TensorFlow Lite format without quantization
converter = tf.lite.TFLiteConverter.from_saved_model(MODEL_TF)
model_no_quant_tflite = converter.convert()

# Save the model to disk
open(MODEL_NO_QUANT_TFLITE, "wb").write(model_no_quant_tflite)

# Convert the model to the TensorFlow Lite format with quantization
x_values = np.float32(x_values) #It is required to use float data type format
def representative_dataset():
for i in range(len(x_values)):
yield([x_values[i]])
# Set the optimization flag.
converter.optimizations = [tf.lite.Optimize.DEFAULT]
# Provide a representative dataset to ensure we quantize correctly.
converter.representative_dataset = representative_dataset
model_tflite = converter.convert()

# Save the model to disk
open(MODEL_TFLITE, "wb").write(model_tflite)

"""### 2. Compare Model Performance
```

```
307
308 In the following cells, to prove these models are accurate even after conversion and
quantization, we'll compare their predictions and performance metrics on our test
dataset.
309
310 ***Note:***
311 *The `predict` (for predictions) and `evaluate` (for loss and accuracy) functions
are defined to be able to use them on the TFLite models. (These are already included
in a TF model, but not in a TFLite model.)*
312 """
313
314 def predict_tflite(tflite_model, x_test):
315 # Prepare the test data
316 x_test = np.float32(x_test) #It is required to use float data type format
317 x_test_ = x_test
318
319 # Initialize the TFLite interpreter
320 interpreter = tf.lite.Interpreter(model_content=tflite_model)
321 interpreter.allocate_tensors()
322
323 input_details = interpreter.get_input_details()[0]
324 output_details = interpreter.get_output_details()[0]
325
326 # If required, quantize the input layer (from float to integer)
327 input_scale, input_zero_point = input_details["quantization"]
328 if (input_scale, input_zero_point) != (0.0, 0):
329 x_test_ = x_test_ / input_scale + input_zero_point
330 x_test_ = x_test_.astype(input_details["dtype"])
331
332 # Invoke the interpreter
333 y_pred = np.empty((len(x_test_),gest_num), dtype=output_details["dtype"])
334 for i in range(len(x_test_)):
335 interpreter.set_tensor(input_details["index"], [x_test_[i]])
336 interpreter.invoke()
337 y_pred[i] = interpreter.get_tensor(output_details["index"])[0]
338
339 # If required, dequantized the output layer (from integer to float)
340 output_scale, output_zero_point = output_details["quantization"]
341 if (output_scale, output_zero_point) != (0.0, 0):
342 y_pred = y_pred.astype(np.float32)
343 y_pred = (y_pred - output_zero_point) * output_scale
344
345 return y_pred
346
347 def evaluate_tflite(tflite_model, x_test, y_true):
348 global model
349 y_pred = predict_tflite(tflite_model, x_test)
350 #Loss
351 loss_function = tf.keras.losses.get(model.loss)
352 loss = loss_function(y_true, y_pred).numpy()
353 #Accuracy
354 accuracy_function = tf.keras.metrics.CategoricalAccuracy()
```

```
355 accuracy_function.update_state(y_true, y_pred)
356 accuracy = accuracy_function.result().numpy()
357 return loss, accuracy
358
359 """**1. Predictions**"""
360
361 # Calculate predictions
362 y_test_pred_tf = model.predict(x_test)
363 y_test_pred_no_quant_tflite = predict_tflite(model_no_quant_tflite, x_test)
364 y_test_pred_tflite = predict_tflite(model_tflite, x_test)
365
366 # Preparing variables for the plot
367 y_test_gest1, y_test_gest2 = prepareY4Plot(y_test)
368 y_test_pred_tf_gest1, y_test_pred_tf_gest2 = prepareY4Plot(y_test_pred_tf)
369 y_test_pred_no_quant_tflite_gest1, y_test_pred_no_quant_tflite_gest2 =
prepareY4Plot(y_test_pred_no_quant_tflite)
370 y_test_pred_tflite_gest1, y_test_pred_tflite_gest2 = prepareY4Plot(y_test_pred_tflite)
371
372 # Compare the predictions
373 x_plot_gest1 = range(0,int(len(x_test)/2))
374 x_plot_gest2 = range(int(len(x_test)/2),len(x_test))
375 plt.clf()
376 plt.title('Comparison of various models against actual values')
377 #plt.plot(x_plot_gest1, y_test_gest1[:,0], 'ko', label='Actual values')
378 #plt.plot(x_plot_gest2, y_test_gest2[:,1], 'ko')
379 plt.plot(x_plot_gest1, y_test_pred_tf_gest1[:,0], 'ro', label='TF predictions')
380 plt.plot(x_plot_gest2, y_test_pred_tf_gest2[:,1], 'ro')
381 plt.plot(x_plot_gest1, y_test_pred_no_quant_tflite_gest1[:,0], 'bx', label='TFLite
predictions')
382 plt.plot(x_plot_gest2, y_test_pred_no_quant_tflite_gest2[:,1], 'bx')
383 plt.plot(x_plot_gest1, y_test_pred_tflite_gest1[:,0], 'gx', label='TFLite quantized
predictions')
384 plt.plot(x_plot_gest2, y_test_pred_tflite_gest2[:,1], 'gx')
385 plt.legend()
386 plt.show()
387
388 """**2. Loss (Categorical Crossentropy) and Accuracy (Categorical)**"""
389
390 # Calculate loss
391 loss_tf, accuracy_tf = model.evaluate(x_test, y_test, verbose=0)
392 loss_no_quant_tflite, accuracy_no_quant_tflite =
evaluate_tflite(model_no_quant_tflite, x_test, y_test)
393 loss_tflite, accuracy_tflite = evaluate_tflite(model_tflite, x_test, y_test)
394 import statistics
395 loss_no_quant_tflite = statistics.mean(loss_no_quant_tflite)
396 loss_tflite = statistics.mean(loss_tflite)
397
398 # Compare loss
399 df = pd.DataFrame.from_records(
400 [["TensorFlow", loss_tf, accuracy_tf],
401 ["TensorFlow Lite", loss_no_quant_tflite, accuracy_no_quant_tflite],
402 ["TensorFlow Lite Quantized", loss_tflite, accuracy_tflite]],
```

```
403 columns = ["Model", "Loss", "Accuracy"], index="Model")
404 df
405
406 """**3. File Size**"""
407
408 # Calculate size
409 size_tf = os.path.getsize(MODEL_TF_H5)
410 size_no_quant_tflite = os.path.getsize(MODEL_NO_QUANT_TFLITE)
411 size_tflite = os.path.getsize(MODEL_TFLITE)
412
413 # Compare size
414 pd.DataFrame.from_records(
415 [["TensorFlow", f"{size_tf} bytes", ""],
416 ["TensorFlow Lite", f"{size_no_quant_tflite} bytes ", f"(reduced by {size_tf -
size_no_quant_tflite} bytes)"],
417 ["TensorFlow Lite Quantized", f"{size_tflite} bytes", f"(further reduced by
{size_no_quant_tflite - size_tflite} bytes)"]],
418 columns = ["Model", "Size", ""], index="Model")
419
420 """## Generate a TensorFlow Lite for Microcontrollers Model
421 The following cell converts the TensorFlow Lite model into a C source file
that can be loaded by TensorFlow Lite for microcontrollers.
422 """
423
424 # Install xxd if it is not available
425 !apt-get update && apt-get -qq install xxd
426 # Convert to a C source file, i.e, a TensorFlow Lite for Microcontrollers model
427 !xxd -i {MODEL_NO_QUANT_TFLITE} > {MODEL_TFLITE_MICRO}
428 # Update variable names
429 REPLACE_TEXT = MODEL_TFLITE.replace('/', '_').replace('.', '_')
430 !sed -i 's/'{REPLACE_TEXT}'/g_model/g' {MODEL_TFLITE_MICRO}
431
432 """## Deploy to a Microcontroller
433
434 In our case, we are interested in deploying the model to an Arduino Nano 33 BLE. To
do this, the array in the C source file is copied and pasted into one of the
examples of the Arduino_TensorFlowLite library in Arduino, and the corresponding
array length is substituted as well.
435 """
436
437 # Print the C source file
438 !cat {MODEL_TFLITE_MICRO}
```

Code 14. (below) Combined Arduino files used to implement the trained ML TensorFlow Lite model (and its associated feature extraction code) into an Arduino Nano 33 BLE microcontroller. Note that the files were built from one of the given ML implementation examples of the Arduino_TensorFlowLite library in Arduino, meaning that some of the code/comments may be exactly the same as in the example. The name of each separate file is shown prior to its corresponding code.

main.ino

```
/* Copyright 2020 The TensorFlow Authors. All Rights Reserved.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

==============================================================================*/

//----- Setting up dependencies and calling other files -----//
#include <TensorFlowLite.h>
#include "main_functions.h"
#include "tensorflow/lite/micro/all_ops_resolver.h"

#include "endpoint_detection.h"
#include "feature_provider.h"
#include "model.h"
#include "output_handler.h"

#include "tensorflow/lite/micro/micro_error_reporter.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/schema/schema_generated.h"
#include "tensorflow/lite/version.h"


//----- Setting up the model -----//
// Globals, used for compatibility with Arduino-style sketches.
namespace {
tflite::ErrorReporter* error_reporter = nullptr;
const tflite::Model* model = nullptr;
tflite::MicroInterpreter* interpreter = nullptr;
TfLiteTensor* input = nullptr;
TfLiteTensor* output = nullptr;
int inference_count = 0;

constexpr int kTensorArenaSize = 2000;
uint8_t tensor_arena[kTensorArenaSize];
} // namespace

void setup() {
// Set up logging. Google style is to avoid globals or statics because of
// lifetime uncertainty, but since this has a trivial destructor it's okay.
```

```
// NOLINTNEXTLINE(runtime-global-variables)
static tflite::MicroErrorReporter micro_error_reporter;
error_reporter = &micro_error_reporter;

// Map the model into a usable data structure. This doesn't involve any
// copying or parsing, it's a very lightweight operation.
model = tflite::GetModel(g_model);
if (model->version() != TFLITE_SCHEMA_VERSION) {
TF_LITE_REPORT_ERROR(error_reporter,
"Model provided is schema version %d not equal "
"to supported version %d.",
model->version(), TFLITE_SCHEMA_VERSION);
return;
}

// This pulls in all the operation implementations we need.
// NOLINTNEXTLINE(runtime-global-variables)
static tflite::AllOpsResolver resolver;

// Build an interpreter to run the model with.
static tflite::MicroInterpreter static_interpreter(
model, resolver, tensor_arena, kTensorArenaSize, error_reporter);
interpreter = &static_interpreter;

// Allocate memory from the tensor_arena for the model's tensors.
TfLiteStatus allocate_status = interpreter->AllocateTensors();
if (allocate_status != kTfLiteOk) {
TF_LITE_REPORT_ERROR(error_reporter, "AllocateTensors() failed");
return;
}

// Obtain pointers to the model's input and output tensors.
input = interpreter->input(0);
output = interpreter->output(0);

//Opening up Arduino Serial port
Serial.begin(9600);

//Setting the LED pins to output
pinMode(LED_BUILTIN, OUTPUT);
pinMode(LEDB, OUTPUT);
pinMode(LEDR, OUTPUT);

//Ensuring the LED is off by default (on when the pin is LOW, off when HIGH)
digitalWrite(LEDB, HIGH);
digitalWrite(LEDR, HIGH);
}

//----- Main loop -----//
void loop() {
Serial.println("==============================");
Serial.println();
```

```
//--- Providing signals and relevant values ---//
// Directly defining the Channel 1 and Channel 2 signals after being pre-processed
and the endpoints detected
float ch1_sample[] = {
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,-0.02420636,-0.02420636,-0.1007114,-0.02420636,-0.07520935,-0.04970735,-
0.04970735,0.05229865,0.001295641,0.02679765,0.07780065,0.05229865,-0.02420636,0.001295641,0.02679765,-
0.07520935,0.001295641,0.05229865,0.07780065,-0.02420636,-0.02420636,-0.07520935,0.001295641,-0.02420636,0.05229865,0.001295641,-
0.02420636,-0.02420636,-0.1007114,-0.07520935,0.001295641,-0.04970735,0.001295641,0.05229865,-0.02420636,0.001295641,0.001295641,-
0.02420636,-0.1772154,-0.04970735,0.4348236,0.2563117,0.05229865,-0.1262124,-0.3047243,-
0.1772154,0.02679765,0.001295641,0.1288036,0.07780065,0.05229865,-0.02420636,-0.04970735,-0.2027173,0.4348236,0.1033027,-
0.1517144,0.05229865,0.2308106,-0.7892554,0.1543056,0.05229865,0.3073156,-0.02420636,-0.2282194,-0.1007114,0.3328167,0.7153416,-
0.07520935,-0.9932684,-0.02420636,0.05229865,0.07780065,0.05229865,0.02679765,0.02679765,0.1288036,0.07780065,-0.02420636,-
0.1262124,-0.1007114,0.02679765,-0.1517144,0.2308106,0.2563117,0.02679765,-0.3557274,0.3328167,0.07780065,-1.248284,-
0.2282194,0.9448556,0.02679765,-0.3557274,0.3328167,0.1288036,0.2053086,0.05229865,-0.07520935,-0.2282194,-
0.1262124,0.07780065,0.2308106,0.1543056,-0.3302253,0.1033027,0.4603246,0.02679765,-0.3812284,-0.2537203,-0.1772154,-
0.1007114,0.001295641,-0.04970735,0.1798066,0.2563117,0.2308106,0.1288036,0.1033027,0.001295641,-0.1007114,0.1543056,-
0.07520935,0.1798066,-0.3812284,-0.6107433,0.1543056,0.07780065,0.1033027,-0.02420636,-0.04970735,0.3838207,0.1543056,-0.1007114,-
0.2792224,0.1288036,0.02679765,-0.07520935,0.1288036,0.1798066,-0.2537203,-0.2537203,-0.2792224,0.7408426,-0.4322324,0.4093217,-
0.1772154,0.2053086,0.07780065,0.05229865,-0.1262124,-0.3557274,0.5113286,0.1798066,0.1033027,-0.9422644,-
0.3302253,0.1288036,0.2308106,0.4093217,0.3328167,0.3583186,-0.3047243,-0.2792224,-0.2537203,-0.1772154,-0.07520935,0.05229865,-
0.04970735,0.1543056,0.02679765,0.2053086,0.3328167,-0.07520935,0.02679765,-0.07520935,-
0.7637534,0.1798066,0.07780065,0.1798066,0.1288036,0.1798066,0.05229865,-0.2027173,-0.07520935,0.02679765,0.07780065,0.1288036,-
0.07520935,-0.1007114,-0.2027173,0.2818137,0.07780065,-0.02420636,-0.2282194,-0.02420636,-0.1772154,-0.2027173,-
0.3302253,0.4603246,0.6898397,0.3328167,0.02679765,-0.2792224,-0.2537203,-0.1262124,0.1288036,-0.07520935,-0.2282194,-
0.04970735,0.02679765,0.001295641,0.02679765,0.2053086,-0.2282194,0.02679765,0.001295641,0.07780065,0.07780065,0.02679765,-
0.02420636,-0.1007114,-0.3557274,0.3583186,0.2818137,-0.07520935,-0.1517144,-0.1262124,-0.07520935,0.001295641,0.1798066,-0.1007114,-
0.2282194,0.07780065,0.2053086,0.1543056,-0.02420636,0.1288036,0.02679765,-0.7382513,-
0.2282194,0.3328167,0.2308106,0.1033027,0.1033027,0.07780065,-0.04970735,0.07780065,0.02679765,-0.04970735,-0.2027173,-0.02420636,-
0.1262124,-0.04970735,-0.2282194,0.4093217,0.1798066,-0.1262124,0.4858267,-0.1262124,0.07780065,0.1288036,-
0.8147563,0.05229865,0.3328167,0.1033027,0.1033027,0.05229865,0.2053086,0.2818137,0.1288036,-0.02420636,-0.1007114,-0.02420636,-
0.07520935,-0.02420636,0.07780065,0.07780065,0.02679765,0.1543056,-0.2282194,0.07780065,-0.04970735,-
0.1262124,0.001295641,0.02679765,-0.3812284,0.1288036,0.001295641,0.02679765,-0.3047243,0.2308106,-0.07520935,0.001295641,-
0.2282194,-0.04970735,-0.3302253,1.505892,-0.2282194,-1.350291,-0.4577334,0.05229865,0.2563117,0.6133347,0.1288036,-
0.2537203,0.07780065,0.2308106,-1.197281,0.5623317,1.862915,-0.1517144,0.001295641,-0.5852424,-0.6362454,-0.1262124,-
0.04970735,0.1543056,0.1033027,0.1288036,-0.1007114,0.1798066,0.5368297,0.07780065,-0.1517144,-0.02420636,-0.04970735,-0.1262124,-
0.07520935,0.001295641,0.05229865,-0.1262124,0.1033027,0.02679765,-0.02420636,-0.04970735,-0.02420636,-0.04970735,-
0.3302253,0.8938527,0.5623317,-0.3812284,-1.222782,0.8173476,0.2308106,0.001295641,-0.1262124,-0.02420636,-0.07520935,-
0.9422644,1.888417,-0.3812284,-1.069773,0.1798066,-0.07520935,0.1543056,-0.1007114,0.001295641,0.001295641,-0.1007114,-
0.04970735,0.1798066,0.02679765,-0.02420636,-0.02420636,-0.02420636,0.02679765,0.001295641,0.1288036,-0.07520935,0.2308106,-
0.1262124,-0.4067304,-0.2792224,0.5113286,-0.07520935,0.3328167,0.1798066,-0.1517144,-0.3812284,0.2308106,0.2818137,-0.4577334,-
0.3812284,0.5368297,0.3073156,-0.6362454,0.02679765,0.3583186,0.2563117,0.001295641,-0.07520935,-0.02420636,-0.1007114,0.1288036,-
0.1517144,-0.07520935,-0.1007114,-0.1517144,0.1543056,0.1798066,0.1288036,-0.2027173,-0.1262124,0.001295641,0.05229865,0.1543056,-
0.1007114,0.1543056,0.3328167,0.3073156,-0.1262124,0.2053086,-0.1262124,-0.04970735,-0.1262124,-
0.2282194,0.001295641,0.1798066,0.02679765,0.05229865,-0.07520935,-0.6872483,1.021361,0.2563117,-0.2537203,-0.1517144,0.05229865,-
0.2282194,0.1288036,0.001295641,-0.02420636,0.1288036,0.1288036,-0.4577334,-0.2537203,0.3073156,-0.02420636,0.2053086,0.1798066,-
0.1517144,-0.2282194,-0.1772154,-0.07520935,-0.02420636,-0.3302253,0.5878336,-0.04970735,-0.2027173,0.02679765,0.05229865,-0.1007114,-
0.1772154,-0.07520935,-0.07520935,0.1543056,0.1798066,-0.1772154,0.2818137,-0.02420636,0.001295641,0.07780065,-
0.1517144,0.02679765,0.02679765,0.02679765,0.2818137,0.1288036,0.1033027,-0.1262124,-0.2027173,-0.1772154,-0.07520935,-0.07520935,-
0.1262124,-0.4577334,0.5878336,0.2818137,-0.2792224,-0.07520935,-0.07520935,0.05229865,-0.1007114,0.1798066,0.05229865,-
0.1262124,0.02679765,0.1033027,-0.04970735,0.05229865,-0.02420636,0.001295641,-0.07520935,0.02679765,-
0.04970735,0.1288036,0.1798066,0.02679765,-0.02420636,0.001295641,-0.1262124,-0.1007114,0.001295641,-0.04970735,-
0.07520935,0.02679765,0.001295641,-0.07520935,0.2308106,0.3838207,-0.4832354,0.2563117,0.1543056,-0.8147563,0.05229865,0.5113286,-
0.04970735,0.1543056,-0.07520935,-0.07520935,0.001295641,0.05229865,0.02679765,-0.02420636,0.001295641,-0.02420636,-0.02420636,-
0.02420636,0.07780065,-0.02420636,0.001295641,-0.02420636,0.05229865,0.001295641,-0.2027173,0.02679765,0.02679765,-
0.02420636,0.02679765,0.05229865,-0.02420636,0.001295641,-0.2282194,0.1543056,0.001295641,-0.1262124,0.3328167,-
0.3557274,0.2818137,0.05229865,-0.07520935,-0.2027173,-0.1262124,-0.04970735,0.1798066,-0.07520935,-
0.07520935,0.02679765,0.1798066,0.1033027,-0.1007114,-0.2027173,0.4093217,0.2818137,-0.2027173,-0.1772154,-0.07520935,-0.1007114,-
0.1262124,-0.2282194,0.05229865,0.1033027,0.07780065,0.07780065,0.02679765,-0.04970735,-0.07520935,-0.1772154,0.2563117,0.02679765,-
0.04970735,0.001295641,-0.02420636,-0.1262124,0.4603246,0.07780065,-0.07520935,-0.1262124,-0.1007114,-0.04970735,0.1543056,-
0.02420636,-0.02420636,-0.02420636,0.001295641,-0.02420636,-0.02420636,-0.02420636,0.07780065,0.02679765,-0.02420636,-0.1517144,-
0.1517144,-0.04970735,0.07780065,0.07780065,0.02679765,-0.02420636,-0.3302253,0.7408426,0.2818137,-0.2792224,-0.2282194,-
0.1772154,0.001295641,-0.07520935,-0.07520935,0.07780065,-0.02420636,0.02679765,0.05229865,0.1288036,-
0.02420636,0.001295641,0.1033027,-0.02420636,-0.02420636,-0.1007114,-0.02420636,-0.04970735,0.001295641,0.07780065,-
0.04970735,0.001295641,0.02679765,-0.04970735,0.1288036,0.1033027,-0.07520935,-0.1517144,0.6643386,-0.2537203,-
0.4322324,0.05229865,0.07780065,-0.1517144,-0.1517144,0.07780065,-0.04970735,-0.1007114,-0.04970735,0.07780065,-
0.3047243,0.1798066,0.2308106,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

```
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
};
unsigned int ch1_wl = 657; //Length of cut part of the Channel 1 signal
float ch2_sample[] = {
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-0.0003079474,-0.1023389,-0.07683194,-
0.0003079474,0.3823081,0.254769,-0.1788629,-0.280894,-0.05132395,0.1017231,0.2037541,-0.561479,-0.1278469,-
0.1023389,0.05070706,0.3568001,0.2292611,0.1272301,0.05070706,-0.0003079474,-0.02581595,-0.02581595,-0.1023389,-0.1278469,-
0.07683194,0.05070706,-0.5104629,-0.07683194,0.2802771,0.05070706,-0.0003079474,0.02519906,-0.05132395,0.07621506,0.02519906,-
0.0003079474,-0.05132395,0.1017231,0.1272301,0.02519906,0.152738,-0.586987,-0.0003079474,-
0.07683194,0.1017231,0.2037541,0.152738,0.02519906,-0.02581595,-0.2043699,0.07621506,0.1272301,-0.1278469,-0.1023389,-
0.0003079474,0.2037541,0.4078161,0.02519906,-0.4339399,-0.280894,-0.0003079474,-0.0003079474,0.02519906,-0.1023389,-0.07683194,-
0.1278469,0.3568001,0.2802771,0.05070706,-0.1533549,-0.2043699,0.2037541,-0.0003079474,0.07621506,0.1017231,0.152738,-
0.280894,0.05070706,0.2037541,0.05070706,-0.0003079474,-0.1023389,-0.1533549,-0.3319089,-
0.02581595,0.02519906,0.2802771,0.3568001,0.254769,-0.1278469,-0.02581595,-0.07683194,-0.586987,-
0.1278469,0.2292611,0.1017231,0.1272301,0.1782461,0.2292611,-0.1788629,-0.02581595,-0.05132395,-0.07683194,-0.1278469,-0.0003079474,-
0.1788629,-0.1023389,-0.1788629,-0.306401,0.1017231,0.152738,0.152738,-0.3829249,-0.2298779,-
0.280894,0.02519906,0.02519906,0.1782461,0.1017231,0.2802771,-0.0003079474,-0.02581595,-0.1023389,-0.1023389,-0.05132395,-
0.1023389,0.05070706,0.2037541,-0.561479,-0.0003079474,0.07621506,0.07621506,0.331292,0.1782461,0.02519906,-0.0003079474,-0.1533549,-
0.1514779,-0.1514779,-0.1514779,-0.05132395,0.331292,0.2292611,0.5098461,-0.4084319,-1.122649,-
0.0003079474,0.254769,0.02519906,0.152738,0.1017231,0.254769,0.02519906,0.2292611,0.4078161,-0.586987,0.02519906,-0.9185869,-
0.8675719,0.3823081,0.8924631,0.02519906,0.4588311,0.4333231,0.1782461,-0.07683194,-0.1023389,-0.07683194,-0.3319089,-
0.07683194,0.1782461,0.331292,-0.484956,-0.0003079474,-0.5104629,-0.05132395,-0.8930799,-
0.4084319,0.688401,0.866955,0.3823081,0.3823081,0.2037541,0.5353541,-0.3574169,-0.4594479,-0.07683194,0.1272301,-0.0003079474,-
0.2553859,-0.2553859,-0.6124939,0.2802771,0.1782461,0.1272301,0.4078161,0.1272301,-1.734835,0.9179701,-0.280894,-0.1533549,-
0.7910489,0.9179701,1.071017,0.4843391,0.688401,-0.6380019,-1.479758,-1.071634,-0.2043699,0.6118771,0.1017231,-
0.1278469,0.4843391,0.4843391,0.4333231,-0.6635099,1.096525,-0.1533549,-0.3574169,-0.3829249,-0.2553859,-2.040928,-
0.1533549,0.4078161,0.7904321,0.4588311,0.1782461,0.688401,-0.02581595,-0.1788629,-0.8165559,-0.07683194,0.4843391,1.37711,0.866955,-
0.4084319,-3.112254,-2.602099,1.249571,1.734218,1.453633,1.198556,0.764924,-0.689018,-0.4084319,0.4588311,-0.586987,0.8414471,-
1.734835,-1.556281,-2.040928,0.6118771,0.9434781,1.198556,0.7139081,0.5608621,0.07621506,-0.3829249,0.3823081,0.6373851,-0.1278469,-
0.8930799,0.1272301,-3.545885,0.688401,0.968986,2.218866,-3.290808,0.1272301,0.6628931,0.1017231,1.198556,1.657695,-
0.2298779,0.6373851,-3.622409,-0.1533549,0.5353541,0.5608621,0.4843391,0.3057851,0.866955,-1.22468,0.4588311,-0.5104629,-0.765541,-
0.4339399,0.02519906,0.7394161,0.688401,-0.484956,0.5863701,1.555664,-0.2553859,0.2292611,-2.806161,-0.07683194,1.402618,0.7904321,-
0.280894,-0.02581595,0.688401,1.224063,-1.377727,-1.45425,0.254769,-0.05132395,-0.2043699,0.9434781,0.3823081,0.764924,0.6628931,-
0.1533549,-0.2043699,-1.760343,1.530156,-2.678622,0.4843391,0.764924,0.866955,1.045509,-1.887882,0.5353541,0.5098461,-
0.0003079474,0.254769,0.9944941,-0.6635099,-3.316316,-0.0003079474,0.5353541,1.096525,0.2037541,0.688401,-0.280894,-
0.5104629,0.331292,1.224063,-0.3319089,0.05070706,-0.0003079474,0.1017231,-0.3319089,-0.944095,-0.306401,0.6628931,0.6118771,-
1.403235,0.6628931,1.096525,1.198556,0.4333231,0.8414471,-3.061238,0.2292611,1.14754,-0.2298779,-0.05132395,-
0.3574169,0.1272301,0.331292,0.2292611,-0.1023389,0.1782461,0.3568001,0.2292611,-1.709327,0.2037541,-
1.097142,0.7394161,0.1017231,0.1017231,-0.280894,0.152738,0.152738,1.173048,0.2037541,0.7139081,-0.7400329,-0.9696029,-0.3574169,-
0.6124939,-0.0003079474,0.7904321,1.632187,0.764924,-1.632804,0.331292,-1.071634,0.866955,0.152738,-
0.8165559,0.4588311,1.173048,0.07621506,0.2292611,0.1017231,0.3568001,0.2037541,-0.4084319,-2.066436,-0.689018,2.550466,-
0.9185869,0.4843391,-2.627606,0.05070706,0.254769,2.091327,-2.47456,1.530156,1.785234,0.331292,-1.22468,0.5098461,0.5863701,-1.887882,-
0.765541,-0.561479,0.152738,-1.173665,0.1272301,-0.7400329,0.8924631,0.764924,0.3823081,0.02519906,1.300587,-1.428742,0.8924631,-
0.689018,0.07621506,0.3568001,0.7394161,-0.05132395,-0.8930799,-3.724439,0.3568001,1.428125,-1.250188,-
1.148157,1.632187,2.550466,0.6628931,0.5353541,-2.47456,-0.4084319,0.866955,0.1017231,-0.0003079474,0.2292611,0.02519906,0.07621506,-
0.5104629,-0.6124939,0.9434781,-4.84678,-0.07683194,0.9944941,2.065819,1.632187,0.6118771,0.1782461,-1.607296,-
0.586987,0.5098461,0.152738,-0.0003079474,-0.4084319,0.1017231,0.02519906,-0.1788629,-0.2043699,0.2292611,0.07621506,0.331292,-
0.9696029,-1.097142,-0.7400329,1.555664,1.734218,-1.403235,-0.05132395,1.37711,0.8924631,0.5353541,0.4333231,0.4078161,-
0.1533549,0.07621506,-2.449052,-0.4594479,-0.765541,0.9434781,-0.8930799,0.7139081,0.5098461,0.1272301,-0.4594479,0.968986,1.020001,-
2.040928,0.5608621,-0.0003079474,-0.02581595,0.1272301,-0.5104629,-0.5104629,0.764924,0.968986,0.3057851,-1.199173,-1.326711,-
0.280894,0.8414471,-0.4594479,-0.944095,1.428125,0.7139081,-0.4594479,-1.326711,0.8414471,0.05070706,0.3823081,-0.02581595,0.05070706,-
0.3829249,-0.306401,-0.0003079474,-0.3574169,0.07621506,0.3568001,0.2292611,0.05070706,0.4078161,0.1017231,-0.1278469,-0.8675719,-
0.280894,0.4078161,0.254769,1.173048,1.581172,-1.581789,-1.658312,0.07621506,-0.05132395,1.224063,0.2037541,-
0.1278469,0.6118771,0.5353541,-0.8420639,-0.2043699,-1.785851,0.1272301,0.6118771,-0.2043699,0.7904321,1.989296,-0.9951109,0.1782461,-
1.428742,-0.689018,-0.1788629,0.866955,0.3057851,0.764924,0.4843391,-0.3574169,-1.632804,0.8159391,0.3568001,0.2037541,-0.5104629,-
0.02581595,0.6628931,0.4078161,0.5353541,-0.306401,-0.05132395,0.2037541,-0.306401,0.5098461,-0.306401,-0.07683194,-
0.2553859,0.5098461,0.3057851,-0.9696029,-0.05132395,-0.5359709,-0.2298779,0.6118771,1.020001,0.866955,-1.020618,-0.3319089,0.3823081,-
0.6124939,0.331292,-0.6380019,0.02519906,-0.3319089,0.152738,0.6118771,-0.3319089,-0.7910489,-0.02581595,0.9179701,-
0.9185869,0.8924631,0.6373851,0.8159391,-0.280894,-1.862374,-0.9696029,1.020001,0.5098461,0.7139081,-0.4594479,0.2037541,-
0.0003079474,-1.352219,0.3057851,0.7139081,0.5863701,-0.07683194,-0.1023389,0.2292611,-0.484956,0.02519906,0.5608621,0.2037541,-
2.270498,0.07621506,2.040311,-0.5359709,-0.8930799,0.866955,0.2292611,-0.2298779,0.5353541,0.3568001,-0.8165559,-0.586987,-
0.1788629,0.02519906,0.2802771,0.7139081,-0.1023389,-0.05132395,0.2802771,0.152738,-1.428742,0.2292611,-
0.02581595,0.3057851,0.7139081,0.4588311,0.5863701,-1.938897,-0.2553859,0.4333231,0.2037541,0.6373851,0.1017231,-2.091944,-0.5104629,-
0.0003079474,0.5863701,0.3823081,0.688401,0.3568001,0.5098461,-0.9185869,0.2037541,0.05070706,-1.071634,1.93828,-2.423544,1.300587,-
0.765541,0.5863701,0.152738,0.3823081,-0.1023389,-0.5104629,0.4588311,-0.8675719,-0.4594479,-0.306401,-0.561479,1.37711,0.2802771,-
1.097142,-1.22468,1.198556,0.6118771,-0.1278469,0.2037541,1.14754,0.3057851,0.3057851,0.764924,-1.22468,-0.1788629,0.4078161,-
```

0.1023389,-1.071634,-0.7910489,0.07621506,-0.765541,0.7904321,0.9179701,-
0.8675719,1.096525,0.1017231,0.7139081,0.4333231,0.5098461,0.2037541,-2.47456,0.5353541,-
0.1023389,0.07621506,0.3057851,0.152738,0.688401,-0.5104629,-0.280894,0.07621506,0.866955,0.1782461,-1.097142,-1.403235,-
0.9696029,1.402618,-0.02581595,0.3568001,0.5863701,-0.05132395,-1.148157,0.4078161,0.4588311,0.9944941,-1.632804,-0.2298779,0.7139081,-
0.2298779,0.5353541,1.555664,-1.938897,-0.561479,0.6628931,-2.142959,0.688401,0.5098461,1.14754,0.8414471,0.6118771,-1.913389,-
0.4594479,0.5353541,0.02519906,-0.2553859,-0.9696029,1.020001,-1.22468,-1.071634,0.9179701,1.810742,-0.7910489,-
2.193975,0.6118771,0.254769,1.224063,0.5098461,-0.02581595,0.07621506,0.3057851,-0.9185869,0.1017231,-0.6380019,0.7904321,-
0.2043699,0.4078161,0.254769,-0.7145249,-2.449052,0.5098461,2.065819,-0.280894,-1.020618,1.683203,0.6118771,-0.1023389,-
0.8165559,0.1272301,-0.6124939,0.9179701,-0.07683194,-0.6635099,-1.607296,0.8924631,-1.020618,1.020001,0.764924,0.866955,-0.9185869,-
1.709327,0.07621506,1.402618,1.428125,0.254769,-1.122649,0.7139081,-0.5104629,0.5098461,-0.2043699,0.2037541,1.60668,-1.020618,-
2.627606,0.5098461,0.1782461,0.07621506,0.866955,0.3823081,0.1272301,0.254769,0.7904321,-0.484956,-
0.02581595,0.3568001,0.02519906,0.9944941,-0.3319089,-0.2043699,-1.581789,1.14754,-1.22468,1.734218,-1.122649,0.2802771,-1.301204,-
1.530773,2.091327,0.7904321,0.2802771,0.1017231,0.2037541,-0.0003079474,0.7394161,0.152738,1.734218,-1.581789,-
1.734835,0.152738,1.300587,-0.05132395,0.2037541,-0.8165559,-3.188777,0.1782461,0.3823081,3.315699,-0.8165559,0.6373851,0.7394161,-
1.352219,-0.306401,1.657695,0.2037541,-0.1023389,-0.2553859,-0.3319089,-2.040928,-0.02581595,-0.3829249,0.331292,0.5608621,0.2292611,-
0.3829249,0.7139081,0.5353541,-0.484956,0.1782461,0.1017231,0.6373851,-0.5359709,0.2292611,-0.2298779,-0.4084319,0.07621506,-
0.0003079474,-0.0003079474,0.1017231,0.152738,0.1272301,1.020001,-0.689018,-0.2043699,-1.326711,-0.4339399,0.7139081,-
0.07683194,0.4078161,0.5608621,0.4588311,-0.484956,0.9944941,-0.7400329,-0.7400329,-0.4594479,-0.9185869,1.020001,-
0.7145249,1.224063,0.5098461,0.7139081,-0.05132395,-0.4339399,0.02519906,-0.944095,0.8414471,0.3057851,-1.071634,-0.0003079474,-
1.122649,0.1272301,-0.1533549,0.9434781,0.1017231,-0.0003079474,-0.7145249,1.045509,-0.0003079474,-0.561479,-
0.3574169,0.5353541,0.4588311,0.6373851,0.1782461,-1.22468,0.02519906,0.8159391,-1.250188,1.020001,-1.734835,-0.3319089,1.861757,-
0.6380019,0.4843391,0.2037541,-0.944095,1.581172,-0.07683194,-1.887882,-0.2043699,1.122032,-
0.3829249,0.6628931,0.5098461,0.7139081,0.4843391,-0.7400329,-1.581789,0.8924631,0.331292,-
0.9696029,0.1017231,0.331292,0.152738,0.4843391,-0.280894,-1.989913,-0.07683194,0.7904321,1.479141,1.096525,-0.07683194,-1.326711,-
0.9951109,0.5863701,0.6373851,0.7139081,0.3568001,1.020001,0.9434781,-0.4594479,-1.556281,-0.7910489,-0.561479,0.05070706,-0.1023389,-
0.0003079474,0.3057851,0.6628931,0.2802771,-0.1533549,0.3823081,0.4078161,-0.944095,0.4843391,-1.862374,1.096525,-0.2553859,-
0.2553859,0.7394161,0.1272301,0.7904321,-0.05132395,-1.071634,0.6628931,-0.8165559,1.402618,0.05070706,-0.02581595,-1.68382,-
0.05132395,-1.097142,1.300587,0.3568001,1.122032,0.152738,-0.2298779,-0.3319089,0.2292611,-0.1023389,-1.734835,-1.020618,-
0.05132395,0.7904321,1.020001,-0.1023389,0.3568001,-0.9951109,0.02519906,0.5608621,0.5353541,0.5098461,-1.301204,-
0.5359709,0.1017231,0.4078161,0.152738,-0.02581595,1.224063,0.4588311,-0.0003079474,-1.22468,0.02519906,0.2802771,0.4333231,-1.22468,-
1.326711,0.1017231,1.581172,1.045509,-1.632804,-1.964405,0.02519906,0.9179701,-0.3574169,-0.05132395,0.2037541,0.6118771,-0.4594479,-
0.2298779,0.331292,-1.020618,1.428125,-0.1278469,-0.8165559,0.254769,0.8924631,0.3568001,0.3823081,-0.0003079474,-0.2043699,-
0.1788629,-0.05132395,-0.484956,0.07621506,-0.9696029,1.020001,-0.7145249,-0.944095,0.8159391,-1.964405,2.065819,-
0.8930799,0.8924631,1.071017,0.8414471,0.3568001,-0.9185869,0.764924,-0.1533549,-1.097142,-0.9185869,0.4078161,0.5863701,0.4078161,-
0.765541,-0.7400329,0.3568001,0.9179701,0.02519906,-1.326711,0.3568001,0.8414471,0.3823081,0.8414471,0.7139081,-
1.148157,0.2802771,0.3823081,-1.199173,-0.1278469,-0.05132395,0.3057851,-0.07683194,0.6118771,0.2292611,-1.148157,-
1.250188,0.2037541,1.14754,0.3057851,-0.4339399,0.1272301,1.453633,-1.760343,0.254769,0.5863701,-
0.944095,0.1017231,0.3568001,1.045509,0.254769,-1.785851,-0.5104629,0.3057851,-0.2043699,0.254769,0.688401,-0.5104629,-0.5359709,-
0.2043699,-0.944095,-0.9951109,0.968986,2.116835,-0.2298779,-1.097142,0.7394161,0.764924,0.6628931,0.5608621,-1.173665,-
0.306401,0.4078161,-0.280894,-1.632804,0.3568001,0.2292611,0.7139081,-
0.6635099,0.6118771,0.3057851,0.3057851,0.02519906,0.4333231,1.198556,1.249571,-4.617211,-
1.046126,0.7904321,1.020001,1.351602,1.428125,-1.709327,-0.4339399,0.5098461,-0.306401,0.5608621,0.1017231,-0.280894,0.5608621,-
0.3574169,-0.0003079474,1.224063,-0.689018,0.2802771,0.254769,0.1017231,-0.765541,-0.8420639,0.3057851,0.1017231,-
0.2043699,0.5863701,0.9179701,-0.5104629,0.5608621,0.254769,0.1272301,-1.887882,0.7139081,0.5353541,-1.479758,-0.07683194,0.4333231,-
0.4594479,1.224063,-1.887882,0.7139081,0.2802771,0.7139081,0.4078161,0.4588311,-0.280894,0.07621506,-0.2298779,-0.0003079474,-
0.1023389,-0.4339399,-0.2298779,-0.0003079474,-0.3829249,-1.122649,-0.484956,1.122032,0.5353541,1.045509,1.581172,-1.377727,-0.9696029,-
0.689018,0.5863701,-0.0003079474,0.3568001,0.2292611,0.4078161,1.14754,-0.5359709,-1.964405,-2.24499,0.02519906,-0.4339399,-
0.280894,1.93828,0.1782461,0.5608621,-0.2553859,-0.05132395,0.1782461,0.7394161,0.1782461,0.07621506,-0.7145249,0.02519906,-
0.05132395,0.331292,-0.586987,-0.2298779,0.331292,-0.3574169,-0.3829249,-0.05132395,1.020001,-0.9185869,-0.9185869,-
1.352219,1.173048,1.198556,1.963788,-1.071634,0.7394161,-0.6124939,-0.3829249,1.708711,-1.148157,-
1.250188,0.1272301,0.1017231,0.07621506,-0.586987,0.2037541,0.152738,-0.02581595,0.4588311,-0.3319089,-
0.3319089,0.6373851,0.5098461,0.3823081,-0.5359709,0.4588311,-0.944095,0.1782461,0.3568001,-0.2553859,0.07621506,-0.6124939,-
0.280894,0.2292611,0.3057851,0.05070706,0.5353541,-0.02581595,0.5098461,0.4078161,-0.1023389,-1.071634,-0.2298779,-1.122649,-
0.1788629,1.326094,0.4588311,1.300587,-1.403235,-0.6380019,-0.0003079474,1.020001,0.7139081,0.5608621,-1.122649,-
1.505266,0.331292,0.1782461,0.331292,-0.1533549,0.7139081,0.8159391,-2.984715,0.1272301,-0.7145249,1.326094,1.504649,-0.9185869,-
0.0003079474,-0.6124939,0.4333231,1.071017,-0.6124939,0.4078161,0.3568001,0.02519906,-0.7145249,-0.5359709,-0.07683194,-
1.301204,1.708711,1.249571,-0.6124939,-0.1278469,0.6628931,0.4333231,-1.479758,-0.6380019,0.8924631,0.688401,0.6628931,0.3568001,-
1.199173,-0.1533549,-0.05132395,-0.8930799,0.4078161,0.254769,0.5863701,0.254769,-0.3829249,-0.8420639,-0.5359709,0.6118771,0.4078161,-
0.8165559,0.1782461,1.37711,0.2037541,0.1017231,-0.4339399,-0.280894,0.4078161,-0.4594479,0.8159391,-0.07683194,-0.6124939,0.5863701,-
0.7145249,0.5608621,-0.2553859,-0.1788629,-0.6380019,-0.8165559,0.7394161,0.05070706,0.4078161,2.422928,-0.765541,-3.341823,-
0.2553859,0.4588311,0.2292611,1.912773,-1.658312,-1.097142,-0.3829249,0.764924,0.4588311,0.4843391,-0.2553859,0.1782461,-
0.1023389,0.4333231,-0.1278469,-1.479758,0.4843391,0.254769,0.331292,-0.0003079474,1.020001,-
3.137761,0.152738,1.096525,0.8924631,1.020001,0.9434781,-2.296006,-0.9696029,0.1017231,0.4078161,0.07621506,0.254769,-0.5104629,-
0.3574169,0.1272301,0.6118771,0.7394161,0.02519906,-0.02581595,-0.5359709,-0.4339399,0.3823081,0.3823081,-0.6635099,-
0.6380019,0.688401,-2.040928,0.7139081,0.3057851,0.2802771,0.3057851,0.4078161,-0.1533549,-0.02581595,-0.4339399,-0.1278469,0.331292,-
0.2553859,1.326094,-1.45425,1.020001,-1.020618,0.2802771,-0.1788629,-0.02581595,0.968986,0.8159391,0.5353541,0.6118771,0.07621506,-
0.6635099,-0.4594479,-0.765541,-1.148157,-0.2553859,0.3057851,0.331292,1.632187,0.6118771,-0.8675719,-1.275696,-1.275696,1.326094,-
0.2043699,0.3568001,0.5098461,0.2292611,0.764924,0.3568001,-0.2043699,0.2292611,-0.561479,-
1.530773,0.1017231,0.254769,0.7139081,0.3568001,-0.8675719,0.4588311,0.152738,0.4333231,-1.530773,-
1.22468,0.3823081,1.198556,0.7139081,0.152738,1.14754,-1.326711,0.331292,-0.2298779,0.1272301,-0.9185869,0.1782461,0.254769,-
1.020618,0.8414471,-0.6635099,0.2802771,0.3568001,0.02519906,0.3057851,-1.326711,0.764924,-0.05132395,0.3057851,-
0.6380019,0.5863701,0.2037541,-0.6380019,-0.7400329,1.224063,0.4078161,-0.05132395,0.4333231,-0.1278469,-1.530773,-
0.5104629,0.6118771,1.020001,1.096525,0.1017231,-0.8930799,0.4078161,0.764924,0.4843391,-1.148157,-0.2298779,0.152738,0.3057851,-
0.0003079474,0.9434781,-1.581789,-1.68382,0.6628931,0.4588311,1.275079,1.351602,0.3568001,-0.4084319,-0.944095,-
0.1788629,0.4078161,0.6118771,-0.4594479,-1.301204,-0.4339399,0.4588311,0.7904321,0.5608621,-0.2043699,-0.9951109,-
0.07683194,0.05070706,0.8159391,0.2037541,-0.05132395,1.020001,0.1017231,0.2037541,-0.1023389,0.6118771,0.2292611,-
1.071634,0.3057851,-0.6124939,-0.07683194,-0.1788629,-0.9696029,0.5098461,-0.7145249,0.6118771,1.224063,0.331292,-0.07683194,-1.68382,-
0.2553859,1.020001,0.6628931,0.5863701,-0.02581595,-0.9185869,1.071017,-0.306401,-0.3574169,0.8159391,-0.4084319,-
2.219482,0.5863701,0.1272301,-0.1788629,-0.3319089,1.60668,1.096525,-0.1023389,-0.5104629,-0.1023389,0.5608621,1.122032,-0.2298779,-
0.6124939,-0.1023389,0.254769,-0.4339399,-0.05132395,-0.4084319,0.4588311,-0.7145249,-0.2043699,-0.1023389,-0.561479,-0.1278469,-
0.05132395,0.7394161,0.8159391,1.122032,-1.199173,-1.326711,-0.0003079474,0.764924,0.8414471,0.8414471,-0.944095,0.2802771,0.1782461,-

```
0.2043699,0.02519906,0.7904321,-0.765541,-1.887882,-0.4084319,0.4588311,0.3823081,-0.2298779,0.6118771,0.254769,0.6118771,0.6373851,-
0.8165559,-0.1023389,-0.7400329,0.152738,1.045509,-0.2043699,0.7904321,0.1782461,0.1782461,-1.607296,0.07621506,0.7139081,-0.280894,-
0.280894,-0.4084319,0.3823081,0.7904321,0.4333231,-1.530773,-0.0003079474,1.020001,0.07621506,-
1.632804,0.254769,1.224063,1.300587,0.2292611,-0.7145249,0.152738,-0.4594479,0.2292611,-0.02581595,-0.1278469,-0.7145249,-
0.8165559,0.5098461,0.1782461,-0.1788629,0.764924,-0.3574169,0.8414471,-2.500068,-0.306401,0.5863701,1.453633,0.4078161,-
0.0003079474,0.5098461,0.2802771,-0.4084319,-0.02581595,-0.7400329,-0.1788629,0.05070706,-0.05132395,-0.05132395,0.1017231,-
0.944095,0.4843391,-1.020618,0.4588311,-0.05132395,0.8159391,-0.4339399,1.37711,-1.734835,-0.306401,1.071017,0.02519906,0.3823081,-
0.02581595,0.331292,-0.3574169,0.3568001,-0.8165559,-0.07683194,0.6118771,-0.689018,0.1782461,-0.02581595,-0.306401,0.331292,-
0.306401,0.688401,-1.22468,-0.1788629,1.071017,-0.3574169,0.1272301,1.020001,0.7394161,0.152738,0.2037541,1.275079,-2.372529,-
1.122649,0.688401,-0.7400329,0.3057851,-0.1278469,-0.02581595,0.05070706,-0.0003079474,0.05070706,0.1017231,0.5608621,0.05070706,-
0.5104629,-1.505266,0.7139081,0.254769,0.2037541,1.37711,-0.280894,0.764924,-2.423544,-0.4339399,1.122032,1.045509,-0.4594479,-
1.046126,0.2802771,1.224063,0.4588311,0.3057851,-0.484956,0.152738,0.5098461,-0.3829249,-0.7145249,-0.4084319,-0.2553859,-
0.2043699,0.07621506,0.1017231,-0.07683194,0.152738,0.7904321,-1.22468,0.4078161,1.351602,0.1782461,0.4078161,-0.2043699,-
0.8930799,0.3823081,-0.6124939,-0.9696029,0.6118771,1.071017,0.9179701,0.7139081,-0.9185869,-0.0003079474,0.07621506,-0.2043699,-
0.586987,-0.0003079474,-1.071634,-0.05132395,0.07621506,0.2802771,0.4333231,-0.02581595,-0.4339399,-0.7145249,-
0.7910489,1.198556,0.4333231,0.5098461,-0.8420639,0.7394161,-1.071634,0.1782461,-0.280894,0.9179701,0.1017231,-0.1278469,-
0.02581595,0.254769,-0.9185869,-0.05132395,0.1782461,-0.1023389,-0.4339399,-0.0003079474,0.968986,-1.607296,-
0.9185869,1.122032,0.688401,-0.306401,1.122032,-0.6380019,0.2037541,0.7394161,-0.765541,0.5098461,-0.5359709,-0.0003079474,-0.3829249,-
0.2298779,0.07621506,0.05070706,0.07621506,-0.8420639,0.9434781,-0.6635099,0.02519906,0.8159391,-0.9185869,0.4333231,-
0.3829249,0.4333231,0.1272301,-0.280894,0.1782461,-0.02581595,0.1017231,0.1272301,0.2037541,0.5098461,-1.428742,-
0.2298779,0.1272301,0.07621506,-0.765541,-0.3574169,0.688401,0.9944941,-0.3319089,0.968986,0.07621506,0.05070706,-0.3574169,-
0.1278469,-1.785851,-0.2553859,1.300587,1.300587,-1.148157,-0.4084319,-0.1533549,0.4333231,0.4078161,0.3057851,0.02519906,0.254769,-
0.5359709,-0.1788629,0.3823081,-0.1023389,-0.4339399,0.5863701,-0.1023389,-2.296006,0.7139081,0.7904321,-
0.3829249,0.4333231,0.6373851,-0.3574169,0.3057851,0.07621506,-0.8165559,-1.45425,-0.02581595,0.5098461,0.968986,-0.4339399,0.3823081,-
0.1788629,0.2802771,0.2802771,-0.7910489,0.4588311,-0.4594479,0.7139081,-0.3574169,0.3568001,0.05070706,-0.2043699,1.020001,-
1.122649,0.152738,0.254769,-0.1788629,0.02519906,0.2802771,0.2292611,-0.1023389,-1.22468,0.02519906,-
0.4339399,0.3057851,0.1017231,0.2802771,-0.7145249,-0.1788629,-0.306401,0.3823081,1.173048,0.9179701,0.2802771,0.02519906,0.2037541,-
0.2043699,-0.4594479,-0.586987,-0.2298779,-0.0003079474,-0.2043699,0.4588311,-0.0003079474,-0.3574169,-0.1023389,1.326094,-3.214285,-
2.933699,1.428125,1.504649,0.968986,0.688401,0.968986,1.045509,-0.1278469,0.2292611,0.07621506,-0.3574169,-0.484956,-0.3829249,-
0.7400329,-0.1533549,-0.9951109,-0.0003079474,-
0.1023389,0.2802771,0.2802771,0.1782461,0.5353541,0.152738,0.1272301,0.1017231,0.1017231,-0.0003079474,0.02519906,0.02519906,-
0.05132395,-0.1533549,-0.1533549,-0.689018,0.331292,0.3057851,1.887265,-0.1278469,-4.132563,-
1.377727,1.708711,2.295389,0.6373851,0.02519906,-0.8675719,0.2037541,0.5863701,-0.3319089,0.4588311,0.02519906,-1.046126,0.05070706,-
0.2553859,-0.1533549,-0.4084319,0.1017231,0.2292611,-0.280894,-0.1023389,0.1782461,-0.4084319,0.254769,-
0.1788629,0.5863701,0.968986,0.3057851,0.4333231,-0.484956,0.5863701,-1.020618,-0.4084319,0.1272301,0.968986,-2.831668,-
0.4594479,0.688401,1.122032,0.331292,0.07621506,0.02519906,-0.3319089,0.1272301,-0.5104629,-0.7910489,0.764924,1.60668,-1.479758,-
0.6635099,-1.199173,0.5863701,1.020001,0.6628931,-0.689018,0.254769,-0.4084319,-0.4339399,0.6118771,-0.05132395,-0.689018,-
0.3319089,0.5863701,-0.07683194,1.249571,-0.8930799,0.07621506,0.9434781,0.4078161,-2.117451,-0.6380019,0.5608621,0.2802771,-
0.6380019,0.2802771,0.866955,0.2037541,0.3057851,-1.122649,0.968986,-0.6635099,-0.05132395,-0.1533549,0.5098461,0.2037541,-0.07683194,-
0.05132395,0.05070706,0.688401,-1.530773,-0.7910489,-0.2553859,0.6373851,0.8414471,0.1782461,0.05070706,-
1.479758,0.764924,0.1272301,0.02519906,0.3057851,-2.01542,-1.403235,0.7139081,1.020001,0.4588311,0.688401,0.7904321,-
1.275696,0.5098461,-0.561479,-0.1278469,0.3568001,0.3823081,-0.4339399,-0.3319089,-0.4084319,-0.8420639,0.7904321,0
};
unsigned int ch2_wl = 2141; //Length of cut part of the Channel 2 signal

// Determining the total number of sample points in the Channel 1 and Channel 2
signals (to use them to extract the features)
unsigned int ch_len = 0;
if (sizeof(ch1_sample)/sizeof(float) == sizeof(ch2_sample)/sizeof(float)) {
ch_len = sizeof(ch1_sample)/sizeof(float); //Length of the whole Channel 1
and 2 signals
}


//--- Extracting sEMG features ---//
//Feature 1 - Mean Absolute Value (MAV)
float ch1_mav = MeanAbsoluteValue(ch1_sample, ch1_wl, ch_len);
float ch2_mav = MeanAbsoluteValue(ch2_sample, ch2_wl, ch_len);

//Feature 2 - Zero Crossings (ZC)
unsigned int ch1_zc = ZeroCrossings(ch1_sample, ch_len);
unsigned int ch2_zc = ZeroCrossings(ch2_sample, ch_len);

//Feature 3 - Slope Sign Changes (SSC)
unsigned int ch1_ssc = SlopeSignChanges(ch1_sample, ch_len);
unsigned int ch2_ssc = SlopeSignChanges(ch2_sample, ch_len);

```

```
131 //Printing features on the Serial port
132 Serial.print("For Channel 1 -");
133 Serial.print(" MAV: "); Serial.print(ch1_mav, 7);
134 Serial.print(", ZC: "); Serial.print(ch1_zc);
135 Serial.print(", SSC: "); Serial.println(ch1_ssc);
136 Serial.print("For Channel 2 -");
137 Serial.print(" MAV: "); Serial.print(ch2_mav, 7);
138 Serial.print(", ZC: "); Serial.print(ch2_zc);
139 Serial.print(", SSC: "); Serial.println(ch2_ssc);
140
141
142 //----- Running inference -----//
143 // Organising the features (x values) to feed into the model
144 float x[6] = {ch1_mav, ch1_zc, ch1_ssc, ch2_mav, ch2_zc, ch2_ssc};
145
146 // Placing the input into the model's input tensor
147 input->data.f[0] = x[0];
148 input->data.f[1] = x[1];
149 input->data.f[2] = x[2];
150 input->data.f[3] = x[3];
151 input->data.f[4] = x[4];
152 input->data.f[5] = x[5];
153 // input->data.f[6] = x[6];
154 // input->data.f[7] = x[7];
155
156 // Run inference, and report any error
157 TfLiteStatus invoke_status = interpreter->Invoke();
158 if (invoke_status != kTfLiteOk) {
159 TF_LITE_REPORT_ERROR(error_reporter, "Invoke failed");
160 return;
161 }
162
163 // Obtaining the output from the model's output tensor
164 float y[2];
165 y[0] = output->data.f[0];
166 y[1] = output->data.f[1];
167
168 // Output the results
169 HandleOutput(error_reporter, y);
170
171 delay(2000);
172 }
```

model.cpp

```
// Automatically created from a TensorFlow Lite flatbuffer using the command:
// xxd -i model.tflite > model.cc

// This is a standard TensorFlow Lite model file that has been converted into a
// C data array, so it can be easily compiled into a binary for devices that
// don't have a file system.
```

```
#include "model.h"

// Keep model aligned to 8 bytes to guarantee aligned 64-bit accesses.
alignas(8) const unsigned char g_model[] = {
 0x1c, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x14, 0x00, 0x20, 0x00,
0x04, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x10, 0x00, 0x14, 0x00, 0x00, 0x00,
0x18, 0x00, 0x1c, 0x00, 0x14, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00,
0x18, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x40, 0x01, 0x00, 0x00,
0x20, 0x00, 0x00, 0x00, 0xe8, 0x00, 0x00, 0x00, 0x50, 0x00, 0x00, 0x00,
0x02, 0x00, 0x00, 0x00, 0xd4, 0x03, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00,
0x01, 0x00, 0x00, 0x00, 0x44, 0x01, 0x00, 0x00, 0x0d, 0x00, 0x00, 0x00,
0xc8, 0x10, 0x00, 0x00, 0xc4, 0x10, 0x00, 0x00, 0xf8, 0x0f, 0x00, 0x00,
0x64, 0x0f, 0x00, 0x00, 0xb8, 0x0e, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00,
0xd0, 0x04, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0xa8, 0x10, 0x00, 0x00,
0xa4, 0x10, 0x00, 0x00, 0xa0, 0x10, 0x00, 0x00, 0x9c, 0x10, 0x00, 0x00,
0xcc, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00,
0x0c, 0x00, 0x14, 0x00, 0x10, 0x00, 0x0c, 0x00, 0x08, 0x00, 0x04, 0x00,
0x0c, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00,
0x28, 0x00, 0x00, 0x00, 0x2c, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00,
0x73, 0x65, 0x72, 0x76, 0x65, 0x00, 0x00, 0x00, 0x0f, 0x00, 0x00, 0x00,
0x73, 0x65, 0x72, 0x76, 0x69, 0x6e, 0x67, 0x5f, 0x64, 0x65, 0x66, 0x61,
0x75, 0x6c, 0x74, 0x00, 0x01, 0x00, 0x00, 0x00, 0x14, 0x00, 0x00, 0x00,
0x01, 0x00, 0x00, 0x00, 0x24, 0x00, 0x00, 0x00, 0x08, 0x00, 0x0c, 0x00,
0x08, 0x00, 0x04, 0x00, 0x08, 0x00, 0x00, 0x00, 0x0a, 0x00, 0x00, 0x00,
0x04, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x64, 0x65, 0x6e, 0x73,
0x65, 0x5f, 0x38, 0x00, 0xb2, 0xf0, 0xff, 0xff, 0x04, 0x00, 0x00, 0x00,
0x0d, 0x00, 0x00, 0x00, 0x64, 0x65, 0x6e, 0x73, 0x65, 0x5f, 0x36, 0x5f,
0x69, 0x6e, 0x70, 0x75, 0x74, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
0x0c, 0x00, 0x00, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x04, 0x00, 0x08, 0x00,
0x08, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00,
0x13, 0x00, 0x00, 0x00, 0x6d, 0x69, 0x6e, 0x5f, 0x72, 0x75, 0x6e, 0x74,
0x69, 0x6d, 0x65, 0x5f, 0x76, 0x65, 0x72, 0x73, 0x69, 0x6f, 0x6e, 0x00,
0x02, 0xf1, 0xff, 0xff, 0x04, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00,
0x31, 0x2e, 0x35, 0x2e, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0f, 0x00, 0x00, 0x00, 0x4d, 0x4c, 0x49, 0x52,
0x20, 0x43, 0x6f, 0x6e, 0x76, 0x65, 0x72, 0x74, 0x65, 0x64, 0x2e, 0x00,
0x00, 0x00, 0x0e, 0x00, 0x18, 0x00, 0x04, 0x00, 0x08, 0x00, 0x0c, 0x00,
0x10, 0x00, 0x14, 0x00, 0x0e, 0x00, 0x00, 0x00, 0x14, 0x00, 0x00, 0x00,
0x40, 0x00, 0x00, 0x00, 0x44, 0x00, 0x00, 0x00, 0x48, 0x00, 0x00, 0x00,
0x58, 0x00, 0x00, 0x00, 0x0b, 0x00, 0x00, 0x00, 0x14, 0x0f, 0x00, 0x00,
0xc8, 0x0e, 0x00, 0x00, 0x60, 0x0e, 0x00, 0x00, 0xd8, 0x0d, 0x00, 0x00,
0x28, 0x0d, 0x00, 0x00, 0x70, 0x0a, 0x00, 0x00, 0x40, 0x03, 0x00, 0x00,
0x5c, 0x02, 0x00, 0x00, 0xa0, 0x01, 0x00, 0x00, 0x08, 0x01, 0x00, 0x00,
0x80, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x01, 0x00, 0x00, 0x00, 0x0a, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00,
0xe0, 0x01, 0x00, 0x00, 0x48, 0x01, 0x00, 0x00, 0xb4, 0x00, 0x00, 0x00,
0x1c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x6d, 0x61, 0x69, 0x6e,
0x00, 0x00, 0x0e, 0x00, 0x18, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x10, 0x00,
0x07, 0x00, 0x14, 0x00, 0x0e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x09,
0x01, 0x00, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00,
0x04, 0x00, 0x00, 0x00, 0xde, 0xf1, 0xff, 0xff, 0x00, 0x00, 0x80, 0x3f,
0x01, 0x00, 0x00, 0x00, 0x0a, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
0x09, 0x00, 0x00, 0x00, 0x3c, 0xfe, 0xff, 0xff, 0x19, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x19, 0x01, 0x00, 0x00, 0x00, 0xa8, 0xf1, 0xff, 0xff,
0x14, 0x00, 0x00, 0x00, 0x0b, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00,
0x40, 0x00, 0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
0x01, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x19, 0x00, 0x00, 0x00,
0x53, 0x74, 0x61, 0x74, 0x65, 0x66, 0x75, 0x6c, 0x50, 0x61, 0x72, 0x74,
0x69, 0x74, 0x69, 0x6f, 0x6e, 0x65, 0x64, 0x43, 0x61, 0x6c, 0x6c, 0x3a,
0x30, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
0x02, 0x00, 0x00, 0x00, 0x8c, 0xf1, 0xff, 0xff, 0x7e, 0xff, 0xff, 0xff,
0x00, 0x00, 0x00, 0x08, 0x18, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00,
0x04, 0x00, 0x00, 0x00, 0xa4, 0xf1, 0xff, 0xff, 0x01, 0x00, 0x00, 0x00,
0x09, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00,
0x06, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x2c, 0xf2, 0xff, 0xff,
0x14, 0x00, 0x00, 0x00, 0x0a, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00,
0x3c, 0x00, 0x00, 0x00, 0x2c, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
0x01, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x14, 0x00, 0x00, 0x00,
0x74, 0x66, 0x6c, 0x2e, 0x66, 0x75, 0x6c, 0x6c, 0x79, 0x5f, 0x63, 0x6f,
0x6e, 0x6e, 0x65, 0x63, 0x74, 0x65, 0x64, 0x32, 0x00, 0x00, 0x00, 0x00,
0x02, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0x02, 0x00, 0x00, 0x00,
0xe0, 0xf2, 0xff, 0xff, 0x00, 0x00, 0x0e, 0x00, 0x14, 0x00, 0x00, 0x00,
0x08, 0x00, 0x0c, 0x00, 0x07, 0x00, 0x10, 0x00, 0x0e, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x08, 0x1c, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00,
0x04, 0x00, 0x00, 0x00, 0x6a, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x01,
0x01, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00,
0x07, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
0xc0, 0xf2, 0xff, 0xff, 0x14, 0x00, 0x00, 0x00, 0x09, 0x00, 0x00, 0x00,
```

```
0x18, 0x00, 0x00, 0x00, 0x3c, 0x00, 0x00, 0x00, 0x2c, 0x00, 0x00, 0x00,
0x02, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x11, 0x00, 0x00, 0x00,
0x14, 0x00, 0x00, 0x00, 0x74, 0x66, 0x6c, 0x2e, 0x66, 0x75, 0x6c, 0x6c,
0x79, 0x5f, 0x63, 0x6f, 0x6e, 0x6e, 0x65, 0x63, 0x74, 0x65, 0x64, 0x31,
0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
0x11, 0x00, 0x00, 0x00, 0x74, 0xf3, 0xff, 0xff, 0x00, 0x00, 0x0e, 0x00,
0x16, 0x00, 0x00, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x07, 0x00, 0x10, 0x00,
0x0e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x24, 0x00, 0x00, 0x00,
0x18, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x00,
0x08, 0x00, 0x07, 0x00, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
0x01, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00,
0x0c, 0x00, 0x10, 0x00, 0x0b, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x04, 0x00,
0x0c, 0x00, 0x00, 0x00, 0x09, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x09,
0x01, 0x00, 0x00, 0x00, 0x78, 0xf3, 0xff, 0xff, 0x14, 0x00, 0x00, 0x00,
0x08, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00,
0x28, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
0x1a, 0x00, 0x00, 0x00, 0x13, 0x00, 0x00, 0x00, 0x74, 0x66, 0x6c, 0x2e,
0x66, 0x75, 0x6c, 0x6c, 0x79, 0x5f, 0x63, 0x6f, 0x6e, 0x6e, 0x65, 0x63,
0x74, 0x65, 0x64, 0x00, 0x02, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
0x1a, 0x00, 0x00, 0x00, 0x54, 0xf3, 0xff, 0xff, 0x22, 0xf4, 0xff, 0xff,
0x04, 0x00, 0x00, 0x00, 0x88, 0x00, 0x00, 0x00, 0xd8, 0xa3, 0x81, 0x3e,
0x35, 0xf2, 0x05, 0xbf, 0x09, 0xd1, 0x09, 0x3f, 0xa9, 0x05, 0x9a, 0xbd,
0xc1, 0x39, 0x86, 0x3d, 0xb7, 0xe9, 0xd8, 0xbe, 0xb5, 0xee, 0xb7, 0xbb,
0x25, 0x08, 0x5e, 0x3c, 0xde, 0x36, 0xc9, 0x3e, 0xb0, 0xdd, 0xcb, 0x3d,
0x1b, 0xb3, 0x95, 0xbe, 0x85, 0x5b, 0x38, 0xbd, 0x2a, 0x3e, 0xfd, 0xbe,
0xb9, 0xbc, 0xc2, 0xbe, 0xce, 0x15, 0x0c, 0xbe, 0x27, 0x9f, 0xa1, 0x3d,
0x28, 0xab, 0xca, 0x3e, 0xb2, 0x8a, 0x57, 0xbd, 0x79, 0xca, 0x11, 0xbf,
0x90, 0x92, 0xb4, 0x3d, 0x1a, 0xd3, 0x3a, 0x3d, 0xda, 0xa6, 0xc2, 0xbc,
0xde, 0x14, 0x82, 0x3e, 0x71, 0x95, 0x8a, 0xbe, 0x5e, 0xf7, 0x08, 0x3e,
0x34, 0x43, 0x59, 0x3e, 0x50, 0x36, 0x5e, 0x3e, 0x82, 0xae, 0x4e, 0x3d,
0xff, 0x0a, 0x2b, 0xbe, 0x20, 0xd5, 0x8d, 0xbc, 0x1d, 0x17, 0x91, 0xbe,
0xb1, 0xae, 0xc6, 0x3e, 0x70, 0xff, 0x3b, 0x3e, 0x61, 0x96, 0xac, 0xbe,
0x9a, 0xf4, 0xff, 0xff, 0x10, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00,
0x14, 0x00, 0x00, 0x00, 0x24, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
0x02, 0x00, 0x00, 0x00, 0x11, 0x00, 0x00, 0x00, 0x0d, 0x00, 0x00, 0x00,
0x73, 0x74, 0x64, 0x2e, 0x63, 0x6f, 0x6e, 0x73, 0x74, 0x61, 0x6e, 0x74,
0x32, 0x00, 0x00, 0x00, 0x20, 0xf4, 0xff, 0xff, 0xee, 0xf4, 0xff, 0xff,
0x04, 0x00, 0x00, 0x00, 0xe8, 0x06, 0x00, 0x00, 0xc6, 0xfb, 0xe9, 0xbd,
0x44, 0x76, 0xcf, 0x3d, 0x7e, 0x7b, 0xb3, 0x3e, 0x8a, 0xce, 0x1d, 0xbe,
0xec, 0x2f, 0xb1, 0x3d, 0x32, 0x90, 0x85, 0x3e, 0x76, 0x21, 0x90, 0xbd,
0x40, 0xc5, 0x93, 0xbe, 0x05, 0x74, 0x0a, 0xbe, 0x62, 0x81, 0x38, 0xbe,
0x12, 0xe2, 0x04, 0xbe, 0xc1, 0xa0, 0x8e, 0xbd, 0x80, 0xfe, 0x61, 0xbb,
0xa0, 0xe0, 0xb9, 0xbe, 0xe6, 0x84, 0xf8, 0x3c, 0xf1, 0xfb, 0xdf, 0x3d,
0x2a, 0x2e, 0xb7, 0xbd, 0x28, 0xa9, 0x98, 0xbe, 0x70, 0x43, 0x46, 0x3d,
0xc0, 0xc0, 0x03, 0xbc, 0x5a, 0x57, 0x52, 0xbe, 0x60, 0x02, 0x7b, 0x3d,
0x18, 0x45, 0xfa, 0xbd, 0x9b, 0xe8, 0x32, 0x3d, 0x43, 0xac, 0x2b, 0x3e,
0xb9, 0x28, 0xf4, 0x3d, 0x29, 0xda, 0x9e, 0x3e, 0x70, 0x6e, 0x5e, 0x3d,
0xdc, 0x9f, 0xa1, 0xbe, 0xe4, 0x5c, 0x1d, 0x3e, 0xa8, 0x4f, 0x28, 0xbe,
0x69, 0xe8, 0x6b, 0xbe, 0x69, 0x94, 0x50, 0xbe, 0xc8, 0xce, 0x4b, 0xbe,
0x48, 0x0f, 0x8f, 0x3e, 0x54, 0x6d, 0x28, 0x3e, 0x09, 0xe1, 0x84, 0xbe,
0x89, 0x7b, 0xc0, 0x3e, 0xa2, 0x1c, 0x66, 0xbe, 0xf8, 0xc6, 0x2b, 0x3e,
0xaa, 0x05, 0x8d, 0xbe, 0x10, 0xad, 0x95, 0xbe, 0x8a, 0x2c, 0x19, 0xbe,
0xdc, 0x28, 0xab, 0x3e, 0x42, 0x26, 0xad, 0xbe, 0x3c, 0x7a, 0xa3, 0x3e,
0xbe, 0x7e, 0xb3, 0x3e, 0xa1, 0x06, 0x68, 0xbe, 0xa2, 0x57, 0x49, 0xbe,
0xf4, 0xe8, 0x97, 0xbe, 0x7a, 0x2d, 0x90, 0xbe, 0xfa, 0xcd, 0x75, 0xbe,
0x2f, 0xbf, 0x00, 0xbe, 0x7b, 0x37, 0x4a, 0xbe, 0x03, 0x26, 0x53, 0xbe,
0x50, 0x10, 0xc6, 0xbd, 0xa8, 0x12, 0x42, 0x3e, 0xe0, 0xdf, 0x72, 0x3d,
0xe0, 0x77, 0xe1, 0xbd, 0xb7, 0x80, 0xb7, 0xbe, 0xe7, 0xd6, 0x14, 0xbe,
0x96, 0x12, 0x9f, 0xbe, 0x8e, 0x91, 0xa9, 0x3e, 0x8f, 0x6d, 0x9b, 0xbe,
0x07, 0xcd, 0x1c, 0xbe, 0xf7, 0x8f, 0x32, 0xbe, 0x08, 0x62, 0x73, 0x3e,
0x28, 0x0f, 0x3f, 0xbd, 0x02, 0x77, 0x9a, 0xbe, 0x66, 0xaf, 0xa1, 0x3e,
0x8c, 0x8a, 0x19, 0x3e, 0xf8, 0x32, 0x34, 0x3d, 0xb8, 0x6c, 0x4f, 0x3d,
0x6c, 0x2b, 0xf5, 0x3d, 0x1c, 0xf9, 0xe6, 0x3d, 0xf4, 0x1e, 0x93, 0x3e,
0x9a, 0xf0, 0x41, 0xbe, 0x1c, 0xf5, 0x0a, 0x3e, 0x83, 0x75, 0x8e, 0x3c,
0x85, 0x2a, 0xc7, 0xbe, 0x5d, 0xce, 0x4b, 0xbe, 0x46, 0x72, 0xa9, 0x3e,
0x18, 0x00, 0x97, 0x3e, 0x76, 0xd1, 0x0b, 0x3e, 0x59, 0x17, 0x21, 0x3e,
0x03, 0xdd, 0x5e, 0xbe, 0x35, 0xdd, 0x94, 0xbe, 0xb8, 0xe9, 0x41, 0x3e,
0xa4, 0x16, 0xbc, 0xbe, 0xc7, 0x8c, 0xdb, 0xbe, 0xda, 0xc5, 0xa4, 0x3d,
0xff, 0x12, 0x0a, 0x3e, 0xea, 0x65, 0x96, 0x3e, 0x4f, 0x08, 0xd3, 0xbe,
0x21, 0xf9, 0xcd, 0xbe, 0x30, 0x66, 0xaa, 0x3e, 0x10, 0x47, 0xe8, 0x3d,
0x40, 0xb4, 0xac, 0xbd, 0xe8, 0x08, 0x2e, 0x3e, 0x12, 0x90, 0xba, 0xbe,
0x66, 0x17, 0xbf, 0x3e, 0x23, 0xc4, 0x8c, 0xbd, 0x81, 0xb6, 0x65, 0x3d,
0x5a, 0x1d, 0xfb, 0x3c, 0xb8, 0x7b, 0xef, 0x3d, 0x19, 0xb6, 0xc0, 0x3e,
0xec, 0xbb, 0xda, 0xbd, 0xc8, 0x75, 0xc4, 0xbd, 0xc0, 0x44, 0xa5, 0x3e,
0x5f, 0x7b, 0xc7, 0xbe, 0xb2, 0xaf, 0x52, 0xbc, 0xd9, 0xe2, 0x87, 0x3e,
0xb8, 0xf5, 0x65, 0x3e, 0xe7, 0xdc, 0x8e, 0xbe, 0x2e, 0x01, 0x5f, 0x3e,
0x30, 0x3e, 0x9c, 0x3e, 0x17, 0x93, 0x86, 0x3e, 0x1a, 0x8d, 0xbe, 0xbd,
0x3c, 0xe0, 0xc2, 0x3d, 0x17, 0xad, 0x97, 0x39, 0x68, 0xa3, 0x42, 0x3e,
0xc4, 0x17, 0x28, 0xbb, 0xb4, 0xff, 0x7d, 0x3e, 0x40, 0xc5, 0xb5, 0x3c,
0xbc, 0x2a, 0x34, 0xbe, 0x40, 0x68, 0x74, 0x3c, 0xe0, 0x1c, 0xbc, 0xbc,
```

0x35, 0x1f, 0x4f, 0x3e, 0x3b, 0xf2, 0x0a, 0xbe, 0x7a, 0xd7, 0x1f, 0xbe,
0xc2, 0x4a, 0x65, 0x3e, 0x99, 0xb8, 0xb7, 0xbe, 0x40, 0xfc, 0xa5, 0xbc,
0xf3, 0x06, 0xb5, 0xbe, 0x04, 0x33, 0x57, 0xbe, 0xdc, 0xc1, 0x78, 0xbe,
0x8d, 0x79, 0x89, 0xbe, 0x70, 0xd7, 0x15, 0xbd, 0xd4, 0x6c, 0x6f, 0x3e,
0xd4, 0x1e, 0xa2, 0xbe, 0x05, 0x13, 0x3d, 0xbe, 0xa6, 0x41, 0x1c, 0x3e,
0x52, 0x87, 0x8a, 0xbe, 0xf4, 0xeb, 0xb1, 0x3e, 0x68, 0xcd, 0xfc, 0x3d,
0xaf, 0xf9, 0x1c, 0xbe, 0x24, 0x8f, 0x9c, 0x3e, 0xf0, 0xc2, 0xf1, 0xbc,
0x96, 0x43, 0x28, 0xbe, 0x40, 0x06, 0x0a, 0xbd, 0x48, 0xe1, 0x31, 0xbd,
0x60, 0xe0, 0x22, 0x3e, 0x5c, 0x69, 0x94, 0x3d, 0x2f, 0xea, 0xa3, 0xbd,
0x58, 0x0c, 0x26, 0x3e, 0xf0, 0xa9, 0x7d, 0xbd, 0xca, 0x20, 0xd0, 0xbe,
0x84, 0x0d, 0x43, 0x3e, 0xc4, 0x21, 0x0b, 0x3e, 0xfc, 0xd6, 0xa6, 0x3d,
0xa7, 0x36, 0x40, 0xbe, 0xb6, 0xeb, 0x32, 0x3d, 0x00, 0xcf, 0xb9, 0x3d,
0x99, 0x2d, 0x2c, 0x3e, 0x42, 0xd1, 0x07, 0xbe, 0x80, 0x64, 0x11, 0xbe,
0xe0, 0x13, 0xa2, 0xbd, 0x1b, 0x62, 0x07, 0xbe, 0xcf, 0x81, 0x4f, 0xbe,
0x78, 0x3a, 0x23, 0x3e, 0x60, 0x44, 0xec, 0x3c, 0x2e, 0xbb, 0xc9, 0xbe,
0x04, 0xb2, 0x5c, 0xbe, 0x04, 0x1a, 0xb9, 0x3e, 0xa0, 0x91, 0x5f, 0x3c,
0xfb, 0x98, 0x79, 0xbe, 0x9d, 0x25, 0x25, 0x3e, 0x92, 0xf9, 0xb6, 0xbe,
0x74, 0x98, 0xb5, 0x3e, 0x78, 0xca, 0x90, 0xbe, 0xa0, 0x80, 0x12, 0x3d,
0x39, 0x90, 0xbb, 0xbe, 0x3f, 0x1f, 0xa1, 0xba, 0xbe, 0xc2, 0xd2, 0xbe,
0x48, 0xe5, 0xd4, 0x3d, 0xe2, 0x71, 0x00, 0xbe, 0x90, 0x18, 0xb7, 0x3e,
0x25, 0xa6, 0x72, 0x3e, 0xe9, 0x68, 0x3d, 0xbf, 0xfb, 0xce, 0xb3, 0x3d,
0x65, 0x9c, 0x4c, 0xbe, 0x80, 0x79, 0x66, 0x3d, 0x1d, 0x19, 0x0e, 0x3d,
0x09, 0xcb, 0xfc, 0x3d, 0x3f, 0x44, 0xef, 0xbe, 0xc9, 0x9d, 0x2e, 0x3d,
0x20, 0x26, 0x23, 0xbe, 0x52, 0x3b, 0x2f, 0xbc, 0x21, 0x2e, 0xa4, 0x3d,
0x0c, 0x81, 0xa7, 0xbe, 0x2a, 0x6e, 0x83, 0xbe, 0x3c, 0xb9, 0x1c, 0x3e,
0x20, 0xa1, 0xef, 0xbe, 0xf4, 0xd2, 0xad, 0xbd, 0xe7, 0x5a, 0x51, 0xbe,
0x77, 0x3d, 0x1d, 0xbe, 0xf2, 0x5c, 0x00, 0x3e, 0x58, 0xe3, 0xd8, 0x3d,
0x70, 0x0c, 0x5b, 0x3e, 0xb2, 0xe0, 0x8a, 0x3d, 0x44, 0xa0, 0x75, 0xbe,
0x20, 0x74, 0xb7, 0x3c, 0xe0, 0xd6, 0xc0, 0x3d, 0xc9, 0x39, 0xbc, 0xbe,
0xe9, 0x33, 0xac, 0x3e, 0x65, 0x09, 0x92, 0x3e, 0x89, 0x5e, 0x37, 0xbe,
0x7a, 0x1c, 0x9b, 0x3e, 0x04, 0x55, 0x60, 0x3e, 0x50, 0x2b, 0xc9, 0xbd,
0x0b, 0xe9, 0x1a, 0xbf, 0x2a, 0xb6, 0xdb, 0xbd, 0x28, 0xde, 0x3a, 0x3d,
0x8c, 0x44, 0x0f, 0xbc, 0x12, 0xad, 0xb5, 0xbf, 0xc4, 0x53, 0x47, 0xbe,
0x8a, 0x85, 0x90, 0xbe, 0xd4, 0x99, 0x58, 0x3e, 0xfe, 0xef, 0x0b, 0xbf,
0xc8, 0x79, 0xdb, 0xbd, 0xc6, 0x61, 0x86, 0x3e, 0x0e, 0xa2, 0x75, 0xbe,
0x32, 0xa4, 0xad, 0xbe, 0x1f, 0xef, 0x86, 0x3d, 0xc0, 0xda, 0x17, 0x3c,
0x94, 0x83, 0x85, 0x3d, 0xe8, 0x56, 0xb6, 0xbe, 0xa6, 0xf2, 0xb2, 0xbe,
0xc1, 0xff, 0xaf, 0xbe, 0x40, 0x45, 0xbd, 0xbe, 0x08, 0x8f, 0x3a, 0x3e,
0x8c, 0x58, 0x31, 0xbe, 0x1c, 0x4b, 0x73, 0x3e, 0x70, 0xe9, 0x87, 0x3e,
0x26, 0x2a, 0xaa, 0x3e, 0x44, 0xd4, 0xae, 0x3e, 0xb9, 0x2a, 0x80, 0xbe,
0x78, 0x81, 0x55, 0x3e, 0x88, 0x6c, 0xcc, 0x3d, 0xc0, 0x9e, 0xa5, 0x3e,
0x90, 0x6f, 0x91, 0xbd, 0xf3, 0xf1, 0x16, 0xbe, 0x40, 0xd1, 0x9e, 0x3e,
0x15, 0x1f, 0x5a, 0xbe, 0xe8, 0xb9, 0x1b, 0x3e, 0x4c, 0x93, 0x42, 0x3e,
0x22, 0x96, 0xab, 0x3e, 0xd4, 0x39, 0x44, 0x3e, 0x1a, 0x23, 0xbf, 0x3e,
0x1c, 0x03, 0x5f, 0xbe, 0x54, 0x0f, 0xa5, 0x3d, 0xf6, 0x9e, 0x60, 0xbe,
0xd4, 0x60, 0x84, 0x3e, 0x79, 0x4a, 0x88, 0xbe, 0xb9, 0x41, 0x1f, 0xbe,
0x0d, 0x42, 0x37, 0x3e, 0x9b, 0x86, 0x36, 0xbe, 0xe0, 0xf8, 0x02, 0xbe,
0xbd, 0x0a, 0x1c, 0xbe, 0x94, 0x57, 0xa1, 0x3e, 0xed, 0xe9, 0x99, 0xbe,
0xce, 0xf9, 0x0d, 0xbe, 0x89, 0xb2, 0xf4, 0xbc, 0x73, 0xad, 0x22, 0x3e,
0x67, 0x99, 0x87, 0x3e, 0x30, 0xab, 0xaa, 0x3e, 0x37, 0x37, 0x72, 0x3e,
0x05, 0x82, 0x0d, 0x3d, 0x1e, 0xa1, 0x9b, 0x3e, 0x90, 0xae, 0xb4, 0xbd,
0x08, 0x8d, 0x3d, 0xbd, 0x08, 0x0f, 0x0d, 0x3e, 0xa3, 0x4a, 0x82, 0xbe,
0x67, 0x3a, 0xae, 0xbd, 0x8e, 0xcd, 0x45, 0xbd, 0xaf, 0x8b, 0x33, 0x3d,
0x13, 0x98, 0x44, 0xbf, 0x55, 0x08, 0x81, 0x3f, 0x48, 0x55, 0x39, 0x3e,
0xc0, 0xe1, 0x6d, 0x3e, 0xd0, 0xef, 0x9a, 0x3e, 0x82, 0xac, 0x82, 0xbf,
0x55, 0x33, 0x9c, 0xbe, 0x5c, 0x8c, 0x8a, 0x3e, 0x1c, 0x5a, 0x1e, 0x3e,
0xa4, 0xd2, 0x35, 0x3e, 0x43, 0x65, 0x74, 0x3e, 0x75, 0x92, 0xb2, 0xbe,
0x3d, 0xe6, 0x91, 0xbd, 0xaf, 0x91, 0xb9, 0xbd, 0x8b, 0x4f, 0x88, 0xbd,
0xf9, 0x27, 0x7c, 0xbc, 0x2e, 0xe3, 0x6d, 0x3e, 0xe8, 0x84, 0xcd, 0x3d,
0x14, 0xae, 0x16, 0x3e, 0xb0, 0x5b, 0x6d, 0xbd, 0xad, 0x97, 0x32, 0x3e,
0x7b, 0x19, 0x1e, 0xbe, 0xe4, 0xee, 0x4e, 0x3e, 0x58, 0xd8, 0x32, 0x3e,
0xdb, 0xa3, 0x5a, 0x3e, 0x53, 0x52, 0x4f, 0xbe, 0xe3, 0x14, 0x83, 0xbe,
0xd0, 0xe1, 0x1b, 0x3e, 0xd8, 0x88, 0x19, 0x3e, 0x00, 0x5d, 0xf3, 0x3b,
0x14, 0xe8, 0x8f, 0xbe, 0x10, 0xf0, 0xd1, 0xbd, 0x0a, 0xed, 0x5b, 0xbe,
0x38, 0xe0, 0x88, 0xbd, 0xf4, 0x1a, 0x1e, 0xbe, 0x53, 0xbc, 0x8a, 0xbe,
0x30, 0xae, 0x2c, 0x3d, 0x38, 0x65, 0x42, 0x3e, 0x60, 0xd7, 0x29, 0xbc,
0x42, 0x41, 0x95, 0x3e, 0x24, 0xd7, 0x87, 0xbd, 0xaf, 0x8f, 0xaa, 0xbe,
0x20, 0x45, 0x9c, 0xbe, 0x60, 0xbf, 0x97, 0xbe, 0x46, 0x1d, 0x30, 0xbe,
0xad, 0x36, 0x92, 0xbe, 0x0a, 0x00, 0xaf, 0xbe, 0x73, 0xb0, 0x28, 0xbe,
0x80, 0x00, 0x94, 0xbe, 0x58, 0x4b, 0xf4, 0xbd, 0x10, 0x44, 0xad, 0x3c,
0xf0, 0xc7, 0x46, 0xbd, 0xbc, 0x66, 0x2a, 0xbc, 0x9e, 0x5e, 0x33, 0xbd,
0xa5, 0xc7, 0x46, 0xbe, 0x98, 0x1e, 0xeb, 0x3d, 0x89, 0xc6, 0x24, 0xbe,
0x35, 0xbc, 0x92, 0x3e, 0x14, 0xc3, 0x7c, 0xbb, 0x1c, 0x34, 0xb0, 0xbe,
0xfc, 0x31, 0x09, 0x3e, 0x14, 0x22, 0x1d, 0x3e, 0xa7, 0x1f, 0x8e, 0xbe,
0x4d, 0xbb, 0x4f, 0xbd, 0xdf, 0x34, 0x74, 0xbe, 0xa2, 0x88, 0x04, 0x3e,
0xc8, 0xde, 0x6c, 0x3e, 0xe4, 0x9e, 0x97, 0xbe, 0x0c, 0xd6, 0x9c, 0x3e,
0x44, 0x6a, 0x7d, 0x3e, 0x0c, 0x76, 0xf3, 0xbd, 0x98, 0xee, 0x95, 0xbd,
0xf9, 0x9a, 0xaa, 0xbe, 0xc4, 0x8b, 0x8b, 0x3e, 0x3f, 0x8c, 0x89, 0xbe,
0xfc, 0x03, 0x23, 0x3e, 0xd2, 0x4e, 0x87, 0xbe, 0x4b, 0xec, 0xf9, 0x3d,
0xd9, 0xf7, 0x59, 0xbd, 0x87, 0xd2, 0x61, 0xbe, 0x80, 0x2e, 0xb6, 0xbb,
0x80, 0xc2, 0xe4, 0x3c, 0x10, 0xd6, 0x8e, 0x3e, 0x37, 0xba, 0xab, 0xbe,

```
0x2e, 0xfb, 0x09, 0x3e, 0xdb, 0x3d, 0x41, 0xbe, 0xf0, 0x42, 0x34, 0x3e,
0xc0, 0x28, 0x43, 0xbc, 0xc0, 0x8f, 0x0b, 0x3e, 0x89, 0x12, 0xae, 0xbe,
0xb6, 0x05, 0x85, 0x3e, 0x6d, 0x8a, 0xce, 0xbe, 0xcf, 0x80, 0x13, 0xbe,
0x08, 0x9d, 0xb5, 0xbe, 0x7b, 0x87, 0x5f, 0x3e, 0x68, 0x2d, 0x24, 0x3e,
0xda, 0xcd, 0x2f, 0xbe, 0x98, 0xc7, 0x6b, 0xbe, 0x0a, 0x36, 0x97, 0x3e,
0x98, 0x22, 0x52, 0xbd, 0x70, 0xdc, 0x0b, 0xbd, 0xb0, 0xed, 0x2f, 0xbd,
0xd4, 0x85, 0xa6, 0x3d, 0x85, 0x4d, 0xa7, 0xbe, 0x57, 0xc9, 0x88, 0x3d,
0x00, 0x07, 0x87, 0xbf, 0x19, 0x4b, 0xad, 0xbe, 0xce, 0x4d, 0xbb, 0x3e,
0x7f, 0xa8, 0xad, 0xbe, 0x63, 0x74, 0x84, 0x3f, 0x98, 0x3f, 0x7a, 0xbf,
0x32, 0xc0, 0x1b, 0x3d, 0x84, 0x05, 0x88, 0xbd, 0xf0, 0xe2, 0x06, 0x3e,
0x3c, 0xac, 0x53, 0xbe, 0x3b, 0x0b, 0xf6, 0xbd, 0x98, 0x56, 0xb3, 0xbd,
0x11, 0x8e, 0xf9, 0x3c, 0x0b, 0x6a, 0x39, 0xbe, 0xc6, 0x88, 0xea, 0x3e,
0x5b, 0x2b, 0x2a, 0xbf, 0x57, 0x20, 0xbc, 0x3e, 0x78, 0xf1, 0x41, 0x3e,
0x0c, 0x58, 0x9d, 0x3e, 0xcc, 0x84, 0x6a, 0xbe, 0xc8, 0x0f, 0x60, 0xbe,
0xb8, 0x67, 0x69, 0x3e, 0x13, 0x03, 0x95, 0xbe, 0x2e, 0xf4, 0xb7, 0x3e,
0xc2, 0x73, 0x54, 0x3e, 0xae, 0x27, 0x7d, 0x3d, 0xa8, 0xb7, 0xe7, 0x3c,
0xb2, 0x38, 0xa8, 0x3e, 0xde, 0x9f, 0x9d, 0xbe, 0x30, 0x74, 0xaf, 0xbc,
0x91, 0xa2, 0x50, 0x3e, 0xff, 0x95, 0xa0, 0x3e, 0x87, 0x2c, 0x6c, 0x3e,
0x40, 0x07, 0x79, 0x3e, 0xc0, 0x90, 0x46, 0xbd, 0x96, 0x59, 0x0a, 0x3e,
0x3b, 0x96, 0xf6, 0xbd, 0x4c, 0xb9, 0x2d, 0x3e, 0x72, 0xfb, 0x27, 0xbe,
0x06, 0xf7, 0xf3, 0xbb, 0xdf, 0x21, 0x19, 0xbd, 0xff, 0xe5, 0xc5, 0xbe,
0xce, 0x0b, 0x75, 0x3c, 0x98, 0xa7, 0xb4, 0x3e, 0x6a, 0xb6, 0xa5, 0x3e,
0xb7, 0xbf, 0xed, 0xbc, 0x18, 0x16, 0xbc, 0xbe, 0x08, 0x49, 0x2d, 0x3d,
0xcd, 0x7e, 0x80, 0x3e, 0x03, 0x51, 0xa8, 0x3e, 0x31, 0x5d, 0xe0, 0xbe,
0xc6, 0xfb, 0xff, 0xff, 0x10, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00,
0x14, 0x00, 0x00, 0x00, 0x24, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
0x11, 0x00, 0x00, 0x00, 0x1a, 0x00, 0x00, 0x00, 0x0d, 0x00, 0x00, 0x00,
0x73, 0x74, 0x64, 0x2e, 0x63, 0x6f, 0x6e, 0x73, 0x74, 0x61, 0x6e, 0x74,
0x31, 0x00, 0x00, 0x00, 0x4c, 0xfb, 0xff, 0xff, 0x1a, 0xfc, 0xff, 0xff,
0x04, 0x00, 0x00, 0x00, 0x70, 0x02, 0x00, 0x00, 0x3f, 0x17, 0x02, 0x41,
0xd6, 0x1a, 0xcd, 0x3e, 0xbc, 0xe1, 0xbf, 0xbe, 0x3d, 0x12, 0x97, 0x40,
0x64, 0x48, 0xae, 0xbe, 0x0f, 0x8d, 0xb1, 0x3e, 0x9e, 0x8d, 0x81, 0x3f,
0x5d, 0x05, 0x2f, 0xbf, 0x05, 0x9c, 0x49, 0x3e, 0xe6, 0x13, 0x86, 0x40,
0xe7, 0x0e, 0xc2, 0x3c, 0xc7, 0x26, 0x22, 0x3e, 0xf0, 0xc3, 0xd0, 0x3c,
0xd2, 0x46, 0x0f, 0x3e, 0x9a, 0x88, 0x45, 0xbe, 0x30, 0x91, 0x34, 0x3d,
0x62, 0xdd, 0x6a, 0x3e, 0x2a, 0xc9, 0xa9, 0xbe, 0xd3, 0x38, 0xcb, 0x3e,
0xc4, 0x7d, 0x82, 0xbe, 0x61, 0xaf, 0xaa, 0xbe, 0xec, 0xd7, 0xc2, 0x3d,
0x1e, 0xb2, 0xae, 0xbe, 0x94, 0x1c, 0xc7, 0x3d, 0xca, 0x35, 0x45, 0x3e,
0x20, 0x43, 0x7f, 0x3d, 0xf4, 0xce, 0x9d, 0x3d, 0xc0, 0x49, 0x99, 0x3c,
0x00, 0xf6, 0x75, 0xbe, 0xca, 0x17, 0x91, 0xbe, 0x4b, 0xac, 0x12, 0x40,
0x20, 0x39, 0xa5, 0xbe, 0xe7, 0x05, 0xb9, 0x3e, 0xf6, 0x57, 0x01, 0x41,
0xdc, 0xed, 0xcc, 0xbe, 0x80, 0x59, 0x86, 0x3e, 0xb5, 0x57, 0x40, 0xc0,
0x60, 0x8e, 0x8c, 0x3e, 0xa3, 0x8e, 0x10, 0xbf, 0x8f, 0xff, 0x21, 0x40,
0x93, 0x56, 0x58, 0xbe, 0x72, 0x12, 0xc6, 0x3e, 0x80, 0x4b, 0x4b, 0xc1,
0x66, 0xdb, 0x8b, 0x3d, 0x03, 0xb3, 0x62, 0x3d, 0xc9, 0x9a, 0x0d, 0xc1,
0x8d, 0xd4, 0x88, 0xbc, 0x0b, 0xf9, 0x48, 0x3e, 0xf2, 0xd7, 0x3f, 0x3e,
0x90, 0x17, 0x71, 0xbd, 0x04, 0xdd, 0xb1, 0xbd, 0x23, 0xaf, 0xa5, 0xbe,
0xf8, 0xb9, 0xae, 0xbe, 0x4c, 0xa3, 0xdc, 0x3d, 0x4b, 0x68, 0xc6, 0xbe,
0xb8, 0x11, 0x03, 0xbe, 0x0a, 0xb9, 0x95, 0xbe, 0xb0, 0x79, 0x80, 0xbc,
0xa0, 0xc7, 0x9b, 0xbc, 0x6c, 0x48, 0x02, 0xbe, 0xc5, 0xfc, 0xdf, 0xbe,
0x8d, 0x41, 0xdf, 0xbd, 0xae, 0x44, 0x87, 0xbc, 0x2b, 0x66, 0xda, 0x3d,
0x2c, 0xc6, 0xe1, 0xbe, 0x03, 0xfd, 0xa2, 0x3e, 0xc9, 0x63, 0x11, 0x41,
0x88, 0xd6, 0xdb, 0x3e, 0xb6, 0x75, 0x9a, 0xbe, 0x2b, 0x29, 0x15, 0x41,
0xbf, 0xec, 0xaa, 0x3e, 0x9f, 0xf8, 0xb0, 0xbe, 0xca, 0x80, 0x13, 0x41,
0x21, 0x7b, 0xba, 0xbe, 0xa6, 0x07, 0x04, 0xbe, 0xd8, 0xb9, 0xa3, 0x3f,
0x1a, 0x41, 0xb6, 0x3e, 0x35, 0xd4, 0x8b, 0xbb, 0x4c, 0x8b, 0xe8, 0xbf,
0x10, 0x30, 0xa8, 0x3c, 0x96, 0x19, 0xbc, 0xbe, 0xca, 0x98, 0x5b, 0xbf,
0x85, 0x57, 0x18, 0xbe, 0x6e, 0x93, 0x2a, 0x3e, 0x95, 0xd7, 0x2b, 0xbe,
0x0e, 0x29, 0x25, 0x3c, 0xb1, 0x48, 0x27, 0xbe, 0xdf, 0x74, 0x13, 0x3e,
0xfc, 0x85, 0xb0, 0xbe, 0xf5, 0x36, 0x8e, 0x3e, 0x86, 0x48, 0xe0, 0xbd,
0x41, 0x88, 0xa3, 0xbd, 0x1c, 0xfd, 0xfb, 0x3d, 0xe4, 0xb0, 0xb5, 0x3f,
0x75, 0xa5, 0xa7, 0xbe, 0x59, 0x50, 0x43, 0x3e, 0xca, 0xf1, 0x49, 0xbf,
0xec, 0xb1, 0x05, 0xbf, 0xa7, 0x21, 0xb8, 0xbe, 0x67, 0xb1, 0x83, 0xbd,
0x68, 0xff, 0x86, 0x3e, 0x9c, 0x9c, 0xf1, 0x3d, 0x01, 0xee, 0x51, 0xbe,
0x49, 0xf5, 0xbd, 0x3e, 0x79, 0xe5, 0x95, 0xbe, 0xec, 0x38, 0x5c, 0x3e,
0xe4, 0xa9, 0x0c, 0xbe, 0x9d, 0x30, 0x34, 0x3d, 0x02, 0xd1, 0x31, 0x3e,
0x46, 0x01, 0x50, 0x3e, 0x66, 0x37, 0x8f, 0xbe, 0xd1, 0xfb, 0xd0, 0x3e,
0x90, 0xce, 0xc8, 0xbe, 0xbc, 0x29, 0xc5, 0x3d, 0x30, 0x36, 0x1f, 0xbe,
0xca, 0x6a, 0x10, 0x3e, 0x08, 0x0d, 0xe9, 0xbd, 0x80, 0xfc, 0x88, 0xbd,
0x00, 0xa2, 0x8b, 0xbb, 0xc4, 0xda, 0x1b, 0xbe, 0x1b, 0x6d, 0xd6, 0x40,
0x37, 0xc5, 0x44, 0x3e, 0x17, 0x2b, 0x51, 0x3e, 0xed, 0x54, 0xc7, 0x40,
0xb3, 0x1e, 0x67, 0x3e, 0x8f, 0x74, 0xd0, 0xbe, 0x66, 0xa3, 0x65, 0x3e,
0x45, 0xe7, 0x69, 0xbe, 0xfa, 0xd3, 0x18, 0x3e, 0x6c, 0x19, 0x01, 0xbe,
0x18, 0x4e, 0x58, 0x3d, 0xb8, 0x22, 0x80, 0xbd, 0xd2, 0x46, 0x6c, 0x3e,
0xd8, 0xf0, 0xb9, 0xbe, 0xb8, 0x4c, 0x70, 0x3d, 0x26, 0xfb, 0x4a, 0xbe,
0x02, 0xbf, 0x13, 0x3e, 0x88, 0xe0, 0xae, 0xbe, 0xf1, 0x2c, 0x76, 0xbe,
0xe4, 0x16, 0xb6, 0xbe, 0xac, 0xd4, 0x3e, 0x3e, 0xcf, 0x90, 0x59, 0x3e,
0x1f, 0xc0, 0xe2, 0xbe, 0x66, 0x5d, 0xb3, 0x3e, 0x65, 0x8a, 0xdc, 0xbe,
0xc9, 0xa6, 0x03, 0x3d, 0x41, 0xf3, 0x4c, 0x3d, 0x28, 0xb5, 0x32, 0x3d,
0xcd, 0x17, 0xa9, 0xbe, 0x96, 0x3b, 0x30, 0x3e, 0xef, 0xbe, 0x3e, 0x41,
0x66, 0x64, 0xab, 0xbe, 0x8e, 0xf6, 0x95, 0x3e, 0xa6, 0x7d, 0xfc, 0x40,
```

```
0xa9, 0x57, 0x3a, 0x3a, 0x59, 0x4b, 0x86, 0x3e, 0x7a, 0xfe, 0xff, 0xff,
0x10, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x14, 0x00, 0x00, 0x00,
0x24, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x1a, 0x00, 0x00, 0x00,
0x06, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x73, 0x74, 0x64, 0x2e,
0x63, 0x6f, 0x6e, 0x73, 0x74, 0x61, 0x6e, 0x74, 0x00, 0x00, 0x00, 0x00,
0x00, 0xfe, 0xff, 0xff, 0xce, 0xfe, 0xff, 0xff, 0x04, 0x00, 0x00, 0x00,
0x68, 0x00, 0x00, 0x00, 0x9d, 0x39, 0xa5, 0x3f, 0x4e, 0x5e, 0x5a, 0x40,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0xa2, 0xa8, 0xd4, 0x3f, 0x22, 0x77, 0x4e, 0x40, 0x85, 0x5a, 0x22, 0xc0,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7d, 0x24, 0x4b, 0xbd,
0xa4, 0x7c, 0x62, 0x40, 0x88, 0x69, 0xeb, 0x3e, 0x0c, 0x98, 0xa1, 0x3d,
0x71, 0xa6, 0x21, 0xbd, 0x85, 0xe9, 0x19, 0x3e, 0xdc, 0x56, 0x5e, 0xbe,
0xe0, 0x8d, 0x87, 0xbc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x57, 0xc1, 0xe8, 0x3f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0xb0, 0xd4, 0x91, 0xbd, 0x09, 0xc3, 0x23, 0xbc, 0xed, 0xf1, 0x05, 0x40,
0x26, 0xff, 0xff, 0xff, 0x10, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00,
0x10, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
0x1a, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x64, 0x65, 0x6e, 0x73,
0x65, 0x5f, 0x36, 0x2f, 0x62, 0x69, 0x61, 0x73, 0x00, 0x00, 0x00, 0x00,
0xa8, 0xfe, 0xff, 0xff, 0x76, 0xff, 0xff, 0xff, 0x04, 0x00, 0x00, 0x00,
0x44, 0x00, 0x00, 0x00, 0xa4, 0x55, 0x43, 0xbd, 0xfd, 0xb3, 0x11, 0xbc,
0x00, 0x00, 0x00, 0x00, 0xd3, 0xd5, 0x21, 0xbe, 0x44, 0xeb, 0xf8, 0xbf,
0x7d, 0xda, 0xaf, 0xbb, 0x1f, 0x52, 0xec, 0xbd, 0x2a, 0x5a, 0x02, 0x40,
0xf6, 0x7b, 0x0d, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x57, 0x5e, 0x0c, 0x3f,
0x72, 0xe8, 0x2c, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x74, 0x5b, 0x14, 0xbd,
0xab, 0xf8, 0x8f, 0xbc, 0xce, 0x01, 0x08, 0x40, 0xef, 0xe1, 0x8b, 0xbd,
0xaa, 0xff, 0xff, 0xff, 0x10, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00,
0x10, 0x00, 0x00, 0x00, 0x24, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
0x11, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x64, 0x65, 0x6e, 0x73,
0x65, 0x5f, 0x37, 0x2f, 0x62, 0x69, 0x61, 0x73, 0x00, 0x00, 0x00, 0x00,
0x04, 0x00, 0x06, 0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x00,
0x08, 0x00, 0x04, 0x00, 0x06, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00,
0x08, 0x00, 0x00, 0x00, 0x70, 0xe3, 0xee, 0xbf, 0x70, 0xe3, 0xee, 0x3f,
0x00, 0x00, 0x0e, 0x00, 0x14, 0x00, 0x04, 0x00, 0x00, 0x00, 0x08, 0x00,
0x0c, 0x00, 0x10, 0x00, 0x0e, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00,
0x02, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00,
0x01, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00,
0x64, 0x65, 0x6e, 0x73, 0x65, 0x5f, 0x38, 0x2f, 0x62, 0x69, 0x61, 0x73,
0x00, 0x00, 0x00, 0x00, 0x90, 0xff, 0xff, 0xff, 0x14, 0x00, 0x18, 0x00,
0x04, 0x00, 0x00, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x10, 0x00, 0x00, 0x00,
0x00, 0x00, 0x14, 0x00, 0x14, 0x00, 0x00, 0x00, 0x14, 0x00, 0x00, 0x00,
0x01, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x44, 0x00, 0x00, 0x00,
0x34, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
0x06, 0x00, 0x00, 0x00, 0x1f, 0x00, 0x00, 0x00, 0x73, 0x65, 0x72, 0x76,
0x69, 0x6e, 0x67, 0x5f, 0x64, 0x65, 0x66, 0x61, 0x75, 0x6c, 0x74, 0x5f,
0x64, 0x65, 0x6e, 0x73, 0x65, 0x5f, 0x36, 0x5f, 0x69, 0x6e, 0x70, 0x75,
0x74, 0x3a, 0x30, 0x00, 0x02, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff,
0x06, 0x00, 0x00, 0x00, 0xfc, 0xff, 0xff, 0xff, 0x04, 0x00, 0x04, 0x00,
0x04, 0x00, 0x00, 0x00
};
const int g_model_len = 4384;
```

feature_provider.cpp

```
#include "feature_provider.h"
#include "Arduino.h"

// Designing a 'sign' function like in MATLAB
static inline int8_t sign(float val) {
if (val > 0) return 1;
if (val < 0) return -1;
return 0;
}

// Creating a function to find the Mean Absolute Value (MAV) of an array
float MeanAbsoluteValue(float* gest_ch, unsigned int wl, unsigned int len) {
float sum = 0;

for (int m = 0; m < len; m++) { //Finding the sum of the absolute values of
```

```
all of the entries of the array
30 sum = sum + abs(gest_ch[m]);
31 }
32
33 float mav = sum/wl; //Calculating the mean using only the length of the
non-zero part of the sEMG signal (or 'wl')
34 return mav;
35 }
36
37 // Creating a function to find the Zero Crossings (MAV) of a signal. To do this,
38 // the function gets the sign of all of the points in the signal and counts the
39 // number of times the sign changes from the start to the end of the signal
40 unsigned int ZeroCrossings(float* gest_ch, unsigned int len) {
41 int8_t previous = sign(gest_ch[0]);
42 int8_t current;
43 unsigned int counter = 0;
44
45 for (int m = 1; m < len; m++) {
46 current = sign(gest_ch[m]);
47 if (previous != current) {
48 counter = counter + 1;
49 }
50 previous = sign(gest_ch[m]);
51 }
52
53 unsigned int zc = counter;
54 return zc;
55 }
56
57 // Creating a function to find the Slope Sign Changes (SSC) of a signal. To do
58 // this, the function calculates the gradient at each point using the values of
59 // the two adjacent points (using central differences) and then gets the sign
60 // of all of the gradient points in the signal. It then counts the number of
61 // times the gradient sign changes from the start to the end of the signal
62 unsigned int SlopeSignChanges(float* gest_ch, unsigned int len) {
63 float first_diff = gest_ch[1] - gest_ch[0]; //It is necessary to manually define
the gradient at the first point as there is no point to the left for adequately
calculating the gradient (using central differences)
64 int8_t previous = sign(first_diff);
65 float gradient;
66 int8_t current;
67 unsigned int counter = 0;
68
69 for (int m = 1; m < len-1; m++) {
70 gradient = (gest_ch[m+1] - gest_ch[m-1])/2;
71 current = sign(gradient);
72 if (previous != current) {
73 counter = counter + 1;
74 }
75 previous = sign(gradient);
76 }
77
```

```
78 float final_diff = gest_ch[len-1] - gest_ch[len-2]; //It is necessary to manually
define the gradient at the last point as there is no point to the right for
adequately calculating the gradient (using central differences)
79 current = sign(final_diff);
80 if (previous != current) {
81 counter = counter + 1;
82 }
83
84 unsigned int ssc = counter;
85 return ssc;
86 }
```

output_handler.cpp

```
18 const float thr = 0.8;
19
20 // Called by the main loop to produce some output based on the x and y values
21 // -->
22 void HandleOutput(tflite::ErrorReporter* error_reporter, float* y_val) {
23
24 Serial.println();
25 Serial.print("Predictions: Gesture 1-");
26 Serial.println(y_val[0], 7);
27 Serial.print(" Gesture 2-");
28 Serial.println(y_val[1], 7);
29
30 if (y_val[0] > thr) {
31 Serial.println("Gesture 1 detected!");
32 digitalWrite(LEDB, LOW);
33 } else if (y_val[1] > thr) {
34 Serial.println("Gesture 2 detected!");
35 digitalWrite(LEDR, LOW);
36 }
37
38 Serial.println();
39 Serial.println("==============================");
40 Serial.println();
41
42 //Keep light on for 1 second, then turn it off
43 delay(1000);
44 if (y_val[0] > thr) {
45 digitalWrite(LEDB, HIGH);
46 } else if (y_val[1] > thr) {
47 digitalWrite(LEDR, HIGH);
48 }
49 }
```

endpoint_detection.h

```
#ifndef ENDPOINT_DETECTION_H_
```

```
#define ENDPOINT_DETECTION_H_

// Defining the functions in the header file
float EndpointDetectionFunction(float* gest_ch);

#endif  // ENDPOINT_DETECTION_H_
```

endpoint_detection.cpp

```
#include "endpoint_detection.h"
#include "Arduino.h"

// This function detects the endpoints of a sEMG signal given a threshold value
// and also calculates the distance between these points (or Waveform Length)
float EndpointDetectionFunction(float* gest_ch) {

}
```

feature_provider.h

```
#ifndef FEATURE_PROVIDER_H_
#define FEATURE_PROVIDER_H_

// Defining the functions in the header file
float MeanAbsoluteValue(float* gest_ch, unsigned int wl, unsigned int len);

unsigned int ZeroCrossings(float* gest_ch, unsigned int len);

unsigned int SlopeSignChanges(float* gest_ch, unsigned int len);

#endif  // FEATURE_PROVIDER_H_
```

main_functions.h

```
#ifndef MAIN_FUNCTIONS_H_
#define MAIN_FUNCTIONS_H_

// Expose a C friendly interface for main functions.
#ifdef __cplusplus
extern "C" {
#endif

// Initializes all data needed for the example. The name is important, and needs
// to be setup() for Arduino compatibility.
void setup();

// Runs one iteration of data gathering and inference. This should be called
// repeatedly from the application code. The name needs to be loop() for Arduino
// compatibility.
```

```
void loop();

#ifdef __cplusplus
}
#endif

#endif  // MAIN_FUNCTIONS_H_
```

main.cpp

```
#include "main_functions.h"

// Arduino automatically calls the setup() and loop() functions in a sketch, so
// where other systems need their own main routine in this file, it can be left
// empty.
```

model.h

```
// Automatically created from a TensorFlow Lite flatbuffer using the command:
// xxd -i model.tflite > model.cc

// This is a standard TensorFlow Lite model file that has been converted into a
// C data array, so it can be easily compiled into a binary for devices that
// don't have a file system.

#ifndef MODEL_H
#define MODEL_H

extern const unsigned char g_model[];
extern const int g_model_len;

#endif  // MODEL_H
```

output_handler.h

```
#ifndef OUTPUT_HANDLER_H_
#define OUTPUT_HANDLER_H_

#include "tensorflow/lite/c/common.h"
#include "tensorflow/lite/micro/micro_error_reporter.h"

// Defining the functions in the header file
void HandleOutput(tflite::ErrorReporter* error_reporter, float* y_val);

#endif  // OUTPUT_HANDLER_H_
```