***Neural Signal Interpretation for Prosthetics***

Alexander Hughes

N00190256

Report submitted in partial fulfilment of the requirements for the BSc (Hons) in Creative Computing at the Institute of Art, Design and Technology (IADT).

***Declaration of Authorship***

*The incorporation of material without formal and proper acknowledgement (even with no deliberate intent to cheat) can constitute plagiarism.*

*If you have received significant help with a solution from one or more colleagues, you should document this in your submitted work and if you have any doubt as to what level of discussion/collaboration is acceptable, you should consult your lecturer or the Program Chair.*

*WARNING: Take care when discarding program listings lest they be copied by someone else, which may well bring you under suspicion. Do not leave copies of your own files on a hard disk where they can be accessed by others. Be aware that removable media, used to transfer work, may also be removed and/or copied by others if left unattended.*

*Plagiarism is considered to be an act of fraudulence and an offence against Institute discipline.*

*Alleged plagiarism will be investigated and dealt with appropriately by the Institute. Please refer to the Institute Handbook for further details of penalties.*

*The following is an extract from the B.Sc. in Computing (Hons) course handbook. Please read carefully and sign the declaration below.*

*Collusion may be defined as more than one person working on an individual assessment. This would include jointly developed solutions as well as one individual giving a solution to another who then makes some changes and hands it up as their own work.*

**Declaration**

*I am aware of the Institute’s policy on plagiarism and certify that this thesis is my own work.*

*Signed: \_\_\_\_\_\_\_\_\_Alexander Hughes\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_03/05/2024\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Failure to complete and submit this form may lead to an investigation into your work.*

**Abstract**

Electromyography (EMG) technology has revolutionized prosthetic design by enabling the translation of muscle activity into functional outputs for prosthetic devices. Despite its potential, the high cost and complexity of advanced EMG-based prosthetics limit their accessibility. This thesis explores innovative methods to simplify the signal acquisition and processing stages of EMG systems, aiming to reduce costs and enhance accessibility for a broader population. Through a comprehensive study integrating both electroencephalogram (EEG) and EMG techniques, this research employs non-invasive methods to optimize signal fidelity and user comfort. The work includes the development of a prototype using Arduino and Grove systems for real-time signal processing, tested against a dataset provided by Mendeley Data. The findings suggest that simplifying these systems without sacrificing functionality is feasible, paving the way for more affordable and accessible prosthetic solutions. This thesis contributes to the field of biomedical engineering by demonstrating how cutting-edge technology can be adapted to meet the needs of diverse users, potentially enhancing the quality of life for individuals with limb loss.

**Acknowledgements**

I would first like to thank my family and friends without whom I would not have had the overwhelming support which kept me going when I was faced with what seemed like an impossible task, they lent me the confidence that this project was not just achievable but also worth the effort.

Secondly, I want to thank both my first supervisor, Cyril Connolly, and my second supervisor, Joachim Pietsch, for their incredible understanding and patience as I developed and took time to make sure I understood my project area. Without them I do not believe my initial project idea would have been fulfilled in such a way and in such great a quality. They kept me on track and accountable for my progress and for that I am ever grateful.

Lastly, I extend thanks and gratitude to all my lecturers and over my five years in this course as they have given me the skills and knowledge to pursue and adapt to any challenges that I faced in my thesis.

Contents

[***1.*** ***Introduction*** 6](#_Toc165671198)

[***2.*** ***Research*** 8](#_Toc165671199)

[***2.1*** ***EEG Acquisition*** 8](#_Toc165671200)

[***2.2*** ***EEG Processing*** 9](#_Toc165671201)

[***2.2.1*** ***Pre-Processing*** 9](#_Toc165671202)

[***2.2.2*** ***Artifacts*** 10](#_Toc165671203)

[***2.2.3*** ***Filtering*** 12](#_Toc165671204)

[***2.2.4*** ***Different Methods of Pre-Processing*** 13](#_Toc165671205)

[***2.2.5*** ***Feature Extraction*** 14](#_Toc165671206)

[***2.2.6*** ***Classification*** 15](#_Toc165671207)

[***2.2.7*** ***Python Libraries*** 16](#_Toc165671208)

[***2.2.8*** ***EEG User Safety*** 17](#_Toc165671209)

[***2.2.9*** ***Consent*** 17](#_Toc165671210)

[***2.2.10*** ***EEG Pivot to EMG*** 18](#_Toc165671211)

[***2.3*** ***EMG Acquisition*** 19](#_Toc165671212)

[***2.4*** ***EMG Processing*** 23](#_Toc165671213)

[***3. Requirements*** 27](#_Toc165671214)

[***3.1 Functional*** 27](#_Toc165671215)

[***3.1.1 Signal Acquisition*** 27](#_Toc165671216)

[***3.1.2 Signal Processing*** 28](#_Toc165671217)

[***3.1.3 Signal Output*** 29](#_Toc165671218)

[***3.2 Non-Functional*** 30](#_Toc165671219)

[***3.2.1 Usability*** 30](#_Toc165671220)

[***4. Design*** 31](#_Toc165671221)

[***4.1 Dataset*** 31](#_Toc165671222)

[***4.2 Flowchart*** 32](#_Toc165671223)

[***5. Implementation/Construction*** 33](#_Toc165671224)

[***5.1 Initial Setup and Configuration*** 33](#_Toc165671225)

[***5.2 Signal Acquisition and Pre-Processing*** 34](#_Toc165671226)

[***5.3 Integration and Control System Development*** 43](#_Toc165671227)

[***5.4 Feature Extraction*** 48](#_Toc165671228)

[***6. Testing & Analysis*** 50](#_Toc165671229)

[***6.1 Acquisition*** 51](#_Toc165671230)

[***6.2 Pre-Processing and Processing*** 51](#_Toc165671231)

[***6.3 Feature Extraction and Output*** 51](#_Toc165671232)

[***7. Discussion*** 52](#_Toc165671233)

[***8. Conclusion*** 52](#_Toc165671234)

[***9. References*** 53](#_Toc165671235)

[***9.1 Image References*** 55](#_Toc165671236)

[***10. Appendices*** 56](#_Toc165671237)

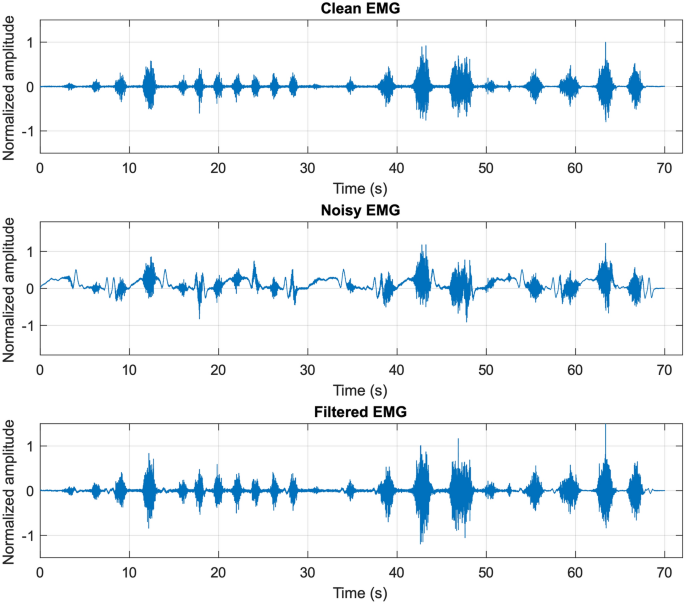
[***10.1 Appendix A: Development Diary*** 56](#_Toc165671238)

# ***Introduction***

Electromyography (EMG) plays a pivotal role in modern prosthetic design, translating electrical activity generated by muscle contractions into actionable data that can drive prosthetic limbs. Prosthetics significantly enhance the quality of life for individuals affected by limb loss due to injury, disease, or congenital conditions. However, the high cost of advanced prosthetics, ranging from $20,000 to $100,000, limits their accessibility (The Complete Guide to arm &amp; hand amputations and prosthetics: MCOP 2020). This thesis explores innovative methods to reduce the complexity and cost of EMG-based prosthetic systems, aiming to make such technology more accessible to a broader audience.

The journey of prosthetic development spans from ancient artifacts like the wooden toe of an Egyptian mummy to today's sophisticated myoelectric arms that integrate seamlessly with human physiology. (AJ, 2007). Initially designed more for aesthetic purposes, advancements in the mid-20th century pivoted towards functional restoration with the introduction of myoelectric prosthetics. These devices harness EMG signals to create intuitive limb movements, a significant leap in prosthetic technology.

Yet, the integration of EMG in prosthetic systems is not without challenges. Signal quality can suffer due to various factors, including interference from external noise and the inherent variability of biological signals. Recent advancements have begun to address these issues through enhanced sensor technologies and sophisticated signal processing algorithms, but the high cost and complexity of these solutions continue to be prohibitive.



*Figure 1: Signal Clarification (Esposito et al., 2023)*

Fig. 1 shows us signal clarification which (Esposito et al., 2023) illustrates the current state of signal processing techniques employed in EMG-based prosthetics. As we can see the noisy EMG graph represents the raw signals that are acquired while the clean EMG graph is shown to set the resting signal rate to a base 0 value. This then goes through the process of filtering which brings us the filtered EMG graph which has amplified the clean signals so that the active periods can be recognized easier by feature extraction methods.

By reducing the barriers to advanced prosthetic technology, we can not only enhance individual autonomy but also inspire further innovation in a field where engineering meets human need.

This thesis aims to tackle these persistent challenges of signal interference, high cost and component complexity by simplifying the signal acquisition and processing stages, ultimately making prosthetic technology both more affordable and functional. Through detailed research into both Electroencephalography (EEG) which is the study of signals collected from the brain, EMG, the development of tailored designs, and the application of novel processing techniques, this work will contribute to the broader goal of democratizing access to advanced prosthetic devices.

# ***Research***

When research commenced it was focused on EEG systems, such as a headset that could be worn to pick up neural signals directly from the brain. EEG is purely non-invasive which seemed a promising avenue for prototyping the acquisition process and it avoids most complicated ethical considerations.

## ***EEG Acquisition***

EEG in essence is a neuroimaging technique that records the electrical activity of the brain, providing valuable insights into brain function and activity. First developed by Hans Berger in the early 20th century, EEG has since evolved significantly in terms of technology and applications.

There are only a few areas where active neurons can generate electrical activity that is recordable on the head’s surface, and even so these signals can still be weak. Amplification is required when signals are detected by scalp electrodes to allow them to be more readable which are then displayed on a computer or on paper.

Signals found on scalp-mounted electrodes are usually at the range of 1 to 150 microvolts over a bandwidth of 0.1 to 60 Hz. These signals have both temporal and spatial variations, and multiple electrodes are commonly used. The positioning of these electrodes is usually determined with the use of the 10-20 standard. This is when the distance between electrodes is measured as 10% or 20% of the skull’s dimensions, with an even number of electrodes on the right, and odd number of electrodes on the left.

An image of this configuration can be seen in Fig. 2.

A diagram of a face with points and lines

Description automatically generated

*Figure 2: General Setup, (Casson et al., 2017)*

An EEG device is normally used to easily place these electrodes on the head in a reproducible fashion. There are different types of devices including the EEG cap, dry EEG device (electrodes), saline solution electrodes, and gel-based electrodes.

Some of the different types can be seen in Fig. 3.

A black headphones with wires

Description automatically generated

*Figure 3: Examples of EEG Recording Devices, (Habibzadeh et al., 2020)*

The techniques explained here are the industry standard for acquiring EEG data then allowing it to be passed from analog to digital data.

## ***EEG Processing***

### ***Pre-Processing***

After the signal acquisition step has been accomplished, the signal processing stage will then commence, starting with the pre-processing of the signal (Hu & Zhang, 2019).

EEG signals often contain unwanted recordings of noise and artifacts, for example eye blinking or movement, and these noise artifacts tend to obscure the signals desired signals.

The purpose behind pre-processing is to minimise the number of noise signals, it is essential to apply pre-processing and de-noising algorithms to the recorded data.

Pre-processing generally allows the removal of artifact-filled data without changing the data itself and can be accomplished in a variety of ways. The desired pre-processing step usually depends on the goal of the study, the details of the experiment, and the recording equipment of raw EEG signals.

Before discussing the algorithms and methods that can be used for pre-processing however, the different types of artifacts should first be reviewed.

### ***Artifacts***

Artifacts in EEG can generally be divided into two different categories. These are physiological and non-physiological artifacts (Hu & Zhang, 2019).

Physiological artifacts stem from biological activity that are close to the head. E.g. Eye movement, or muscle contractions. Non-physiological artifacts come from the interactions between the electrodes and the scalp, the equipment used, or the surrounding electrical devices that can potentially disrupt the EEG signals. These will be discussed in greater detail in the next section.

**Physiological Artifacts:** As stated in previous section, physiological artifacts come from different sources in the body. Common examples of this are eye blinks, eye movement, head movement, heartbeats, and muscular noise. These artifacts can be detected with other forms of biometric data, for example, an electrooculogram or eye-tracking data to detect eye blinks (Hu & Zhang, 2019).

Ocular related artifacts can easily be spotted on a conscious participant during EEG testing. It normally occurs during eye blinks or eye movement. Ocular related artifacts usually don’t ruin brain-generated EEG signals, but they can be minimized by placing electrodes above and below the eye, as well as other methods to help minimize these artifacts.

Electromyography artifacts can also occur. They are high frequency activities and are usually very spiky but are too fast to be epileptic discharge. They can occur during jaw clenching or chewing. Examples of these kinds of artifacts can be seen in Fig. 4

A close-up of several waves

Description automatically generated

Figure 4: Ocular and EMG Artifacts, (Hu & Zhang, 2019)

Artifacts from the heart are normally present during EEG recordings. Electrocardiographic artifacts are easier to be seen in overweight patients, or patients with shorter necks or within babies. This is because the dipole is located closer to the recording electrodes and can transmit the current better. These artifacts are crucial in interpreting physiologic functions that can occur during recording (Hu & Zhang, 2019).

Precipitation artifacts from the scalp can also produce electrical connections between electrodes that are unwanted and are usually low frequency.

These artifacts can be handled by being avoided, rejected, or removed. Eye movements for example can be suppressed. Pre-processing techniques have been created to remove artifacts from EEG recordings which can improve signal-to-noise ratio. Examples of these are independent component analysis (ICA), and more algorithms which will be discussed in a later section.

**Non-Physiological Artifacts:** Non-physiological artifacts are generally from outside sources. These can cause prominent issues during EEG recordings. A common non-physiological artifact is electrical interferences. They can occur from alternating main power supplies at 60Hz in the US and 50Hz in Europe. This can be prevented by shielding recording rooms or relocating subjects from the interference source. It is best to ground the subject and apply notch filters at 50 or 60 Hz to help combat the power line interference.

Electrode artifacts are also common and occur during poor electrode placements. This can lead to disturbances in the electric double layer that can result in an electrode “pop”. Regularly cleaning and inspecting electrodes for corrosion, damaged insulation or broken lead wires is essential in minimizing the effect of these artifacts.

Additionally, malfunctions within the EEG recording system can also occur. The equipment itself can cause artifacts, for example amplifier noise can be caused by thermal agitations of electrons, loose wiring or circuit board connections can lead to artifacts. Digital artifacts such as aliasing, analog to digital conversion errors, and multiplexing artifacts can also disrupt the EEG signal. This is prevented by adjusting the environment, ensuring that electrodes are secured properly, and use electrodes with low-noise amplifiers inside (Hu & Zhang, 2019).

### ***Filtering***

Filtering is essential in pre-processing EEG data. It helps to reduce the different kinds of noises that are available in the recordings, for example 50Hz or 60 Hz line noise, high and low frequency noises. Most of these noises may come from different sources, however they can all be reduced with the use of band-pass filtering. Digital filters can be applied to EEG recordings to improve the interpretations and remove the undesired artifacts found within the EEG signals (Hu & Zhang, 2019).

There are four different kinds of filters that can be applied. These are the low-pass filter, high-pass filter, band-pass filter, and band-stop filter.

* Low pass filter filters out signals within a certain value. Frequencies below that value is kept and frequencies above that value are removed.
* High pass values are the opposite, where frequencies greater than a certain value are kept, and signals that are below that value are removed.
* Band-pass filters keep frequencies between a lower and upper range, while signals that go above or too low that range are removed.
* Band-stop filters eliminate signals within a particular frequency band while being at rest.

An image of the different filters can be seen in Fig. 5.

A group of blue lines

Description automatically generated

Figure 5: Different filters, (Hu & Zhang, 2019)

It is crucial to consider the frequency ranges of the artifacts that are posing problems in EEG recordings. For example, to remove a low-frequency drift, a high pass filter with a limit of 0.1Hz can be applied, while a low-pass filter with the limit of 30 Hz can reduce high-frequency noise that is caused by muscle activity.

### ***Different Methods of Pre-Processing***

As stated in the previous section, there are many different pre-processing methods, that can be completed. The method to be used will depend on the goal of the application that is being created. This section will be discussing the various methods of pre-processing techniques (Lakshmi et al., 2014).

**Independent Component Analysis (ICA):** This is the separation of artifacts from EEG signals into independent components, which are based on the different characteristics of the data that do not depend on the reference channels. During the artifact removal process, ICA preserves the data in the recording trails, each channel’s data, as well as the frontal data. The algorithm decomposes multi-channel EEG data into temporarily independent and spatially fixed components. It is computationally efficient but requires more computations to decompose signals. It shows high performance when dealing with large datasets and EEGLAB shows various types of ICA algorithms. A common algorithm is the Join Approximate Decomposition of Eigen matrices. These algorithms can aid in removing artifacts from EEG Data.

**Common Average Referencing (CAR):** CAR removes noise by subtracting common activity from an area of interest. Noise present in the EEG signal can be the common activity. The referencing method is crucial in enhancing signal-to-noise ratio. CAR can remove the mean within all electrodes and result in clear, and less noisy signals.

**Surface Laplacian (SL):** SL is the estimation of current density entering or leaving the scalp through the skull. The outer shape of the volume conductor is focused, which eliminates the need for in-depth volume conduction details, which can allow for a high-resolution viewing of EEG data.

**Common Spatial Patterns (CSP):** CSP detects abnormal EEG activities. CSP transforms EEG signals into a variance matrix, maximizing discrimination between classes. It applies spatial filtering and utilizes spatial information and can effectively identify patterns of EEG data.

**Principal Component Analysis (PCA):** PCA is a statistical method that transforms correlated vectors into linearly uncorrelated ones, which is known as “Principal Components”. It is based on Second Order Statistics and relies on the decomposition of the covariance matrix. PCA has provided exceptional classification results when used in Brain-Computer Interfaces (BCI).

**Adaptive Filtering:** Adaptive Filtering changes signal properties depending on the specific properties of the signals being analysed. Adaptive filters allow the removal of noise without removing important information. The Least Mean Square algorithm (LMS) removes EEG artifacts effectively.

These methods all provide a way to create noise-free signals that is much clearer and more easily used. Feature extraction is the next step and will be discussed in the next section.

### ***Feature Extraction***

Once noise-free signals are obtained, essential features from the brain signals are then extracted. It is a crucial step as these extracted features are what is used for classification. There are many algorithms that can also be used for feature extraction (Lakshmi et al., 2014). These can be seen below.

**ICA:** ICA creates components that are independent from each other. These components contain essential features that were extracted with the use of ICA. This doubles as a method of feature extraction, and noise separation from brain signals.

**PCA:** PCA can also be used in feature extraction. It can be used for analysis and dimension reduction of the data, without losing any information. With the use of PCA, the information that is seen in the time series multi-channel is used as principal components.

**WT:** Wavelet Transform includes a potential method of feature extraction in B-Spline parameters. The function acts as a low-pass and high-pass filter and with the filtering properties, it acts stands as B-spline clients. Multi-resolution analysis allows filter coefficients to be obtained.

**Autoregressive (AR):** The AR method is used as feature extraction seen within the time-domain analysis. With the use of a shorter duration of data records, the method allows for better frequency resolution and reduces problems of spectral loss. It’s a commonly used method for non-stationary signals. Examples of autoregressive models are: Bilinear AAR, Adaptive AR parameters (AAR), multivariate AAR (MVAAR).

**Wavelet Packet Decomposition:** This method extracts features in the time and frequency domain using the coefficients mean of WT. The initial features are used as the power at special subsets. Fisher’s Criterion was used to measure the separability. Decomposing low frequency waves are shown using wavelet packet tree.

**Fourier Transform (FT):** FT extracts features by transforming signals found within the time domain to frequency domain, this works well for signals that are stationary and have a linear random process. It cannot measure both time and frequency domains. Signals are split into one second windows that overlaps a half second window. The second overlap results are for large numbers of data training for the classifier.

### ***Classification***

Finally, after feature extraction has been accomplished, classification is the final step in the signal processing phase (Lakshmi et al., 2014). It uses the extracted features seen within the feature extraction phase and classifies it to get the specific observations (Kumar & Bhuvaneswari, 2012). There are many methods and algorithms in achieving classification and some of these include Linear Classifiers, Artificial neural networks, Non-linear Bayesian classifiers, and Nearest Neighbour classifiers.

**Linear Classifiers:** Linear functions are used to classify the signals into classes. Examples of frequently used classifiers are Linear Discriminant Analysis (LDA), and Support Vector Machines (SVM).

* **LDA** builds models of the probability density function. It is simple to implement, low on computational requirements, and generally provides good results. A downside is that it might not preserve complex structure if a non-gaussian distribution is used.
* **SVM** are linear classifiers that are used for BCI applications. It uses statistical learning theory and uses the principal of structural risk minimization. Hyper planes are found by SVM and used to separate data sets. These data sets are separated with a wide gap in between to classify the sets into the relevant categories. This method is high in performance and uses many computational complexities.

**Artificial Neural Networks (ANN):** These are non-linear classifiers that contain many interconnected elements known as neurons. A single neuron in ANN is supposed to simulate a biological neuron and can perform simple computational tasks. A commonly used neural network is the Multi-Layer Perceptron Neural Network (MLPNN). It works by separating the network into three layers, which are the input layers, hidden layers, and output layers. MLPNN is fast in operation, easy to implement, and uses small training data. The complexity of ANN is estimated by the number of hidden layers, as with more hidden layers, the more complex it is but with less classification errors.

**Non-linear Bayesian Classifiers (NBC):** These create non-linear decision boundaries. They are usually generative and are made to be more effective in rejecting uncertain samples over discriminative classifiers. The Bayesian Classifier give a feature vector to the class with the highest probability. Hidden Markov Model is an example of a dynamic and NBC. This kind of classification is suitable for the classification of the time series.

**Nearest Neighbour Classifier (NNC):** This classifier gives a feature vector to the class that is based on its nearest neighbour. A feature vector that is from the training data is known as the k-Nearest Neighbour Classifier. It is a non-parametric method and is simple to understand, easy to implement and debug.

With the number of classifiers found, it is essential to choose the correct one for the specific requirements of this project. One that can be the most effective in performance with the least computational drawbacks. Fortunately, different libraries can be used to help with the processing steps and can be seen in the next section.

### ***Python Libraries***

EEG Python libraries are used to help during the processing phase of EEG. This section will be discussing the different EEG Python libraries.

**NeuroKit2**

NeuroKit2 is open source and community driven Python package for neurophysiological signal processing. It can be used for a range of bodily signals and can also be used for EMG. The processing can be accomplished with a few lines of code and contains examples for most-used scenarios, as well as tools for specific tools seen in the processing steps. It is transparent, easily recreated, user-centred and accessible to novice and advanced users. (Makowski et al., 2021)

**MNE-Python**

MNE-Python is part of MNE software suit. It analyses weak signals coming from the brain and is measured by Magnetoencephalography and electroencephalography (M/EEG). It is an open-source package with advanced algorithms in Python for tasks such as data pre-processing, source localization and estimating brain-region connectivity. The code is consistent, well-documented and integrates well with different python libraries for scientific computation and visualization. It has easy access to pre-processed datasets and is user-friendly with comprehensive documentation (Gramfort et al., 2013).

**PySpace**

PySpace is designed to make the task of dealing with complex datasets easier. It automates signal processing and uses machine learning to analyse data which helps in distinguishing noise. Various signal processing algorithms are applied and compared by PySpace and can be used to find suitable preprocessing methods or train supervised algorithms for data classification. The key features of PySpace are automated data handling, modular construction of signal processing chains, disturbed processing, performance evaluation, and visualisation tools. It also has many algorithms for the preprocessing and postprocessing stage (Krell et al., 2013).

**PyEEG**

PyEEG is an open-source Python module for EEG feature extraction, and targets programmers who are working on computational neuroscience. It focuses on extracting features from EEG/MEG segments; however, it doesn’t contain functions to import data of different formats or exporting features to a classifier (Bao et al., 2011).

**BioSig**

BioSig is open source and is used in biomedical signal processing. It is free and contains many tools in different application areas, including BCI and EEG research. This library can provide solutions for data acquisition, artifact processing, quality control, classification, and feature extraction. It can also allow to visualise the data and work effectively with biomedical signals (Vidaurre et al., 2011).

**Grumpy**

Grumpy is an advanced toolbox with high-quality algorithms for processing EMG and EEG bio signals. There are many signal processing and classification methods, including traditional machine learning and deep learning neural network models. Grumpy is versatile, and supports EMG and EEG analysis, visualization, and real-time streaming and decoding (Tayeb et al., 2023).

### ***EEG User Safety***

As this project and area is centred around performing human testing and using them to read EEG signals, it is crucial to identify the different safety procedures that must be performed to ensure that the user comes to no potential harm, whether it is physically, mentally or ensuring that their data doesn’t get leaked and is secured and protected. The first step in ensuring user safety is providing consent and is discussed further in the next section.

### ***Consent***

Giving consent is crucial when performing user testing, to ensure that the participant is aware of what they will be doing, informing them in detail of the tasks that they will be doing is essential.

**Informed Consent:** The British Psychological Society guidelines state that when researchers (students included) are conducting studies that involve human participants, the participant needs to be fully aware of what is being performed in the research (Moore, 2007).

Researchers must explain clearly to the participants in detail what will happen to them during the testing. In this scenario, as participants will have electrodes on them for EEG, the researchers should explain the process and show the equipment that will be used on them, such as the electrode cap, the gel solutions, etc, ensuring that they are aware of the equipment and what they will be used for (Moore, 2007).

Informing the participants of potential side effects that can occur after the experiment is also worth explaining. The knowledge of these side effects can help the participant in deciding whether they should proceed with the testing or not. In the case of using gel-based EEG, the participant should be informed that gel can potentially stay on their hair and would need to be washed out after the testing.

A document for detailed information can be provided for the participant to fully inform them of the testing process. This document can include the procedure, how the researcher will do the tests, the equipment seen within the procedure and potential risks that can occur.

In the case of EEG testing, the user can be informed during different steps in the procedure.

During the testing;

* The skin of the face will be cleaned to ensure maximum signal accuracy.
* The location of the electrode placement, being on the face, scalp, and earlobes, informing them of the EEG cap and other equipment used.
* The user will experience a slight scratching sensation when electrodes are being applied on the hair.

After the procedure;

* The electrodes will be removed but some conducting gel will remain.
* There may be some red marks on where the electrodes were.

These potential points could be placed on the document for EEG testing on a consent form and be given to the participant to read before the test proceeds.

**Right to Withdraw:** Once the information is provided to the participant, their right to withdraw from the experiment should always be made clear. They must be informed that they are allowed to discontinue the experiment at any point even after they have given their consent and should be emphasized that it is okay to do so. This ensures that the participants do not feel guilty if they withdraw and that there will be no consequences if they choose to do so (Moore, 2007).

While consent will not be entirely relevant to due to self-testing and this being a non-clinical project it is still important to observe proper procedure as good practice.

### ***EEG Pivot to EMG***

While a large amount of research was done covering EEG methodology including the acquisition techniques, signal processing, literature reviews, data analysis/collection, user safety/ethical considerations and experiment design, it was eventually evident that EEG would not be the pathway worth pursuing as through a price comparison report conducted during in December it was found that the hardware required (EEG Headsets) would prove to be costly and complex to use, which is one of the problems that this project aims to improve upon.

A black screen shot of a black text

Description automatically generated

*Figure 6: EEG Research folder.*

This led to a pivot in the research focus towards EMG as a suitable methodology for neural signal acquisition. The broad range of research done in EEG before the pivot can be seen in Figure 4 above.

## ***EMG Acquisition***

The research undertaken in EMG followed an almost identical structure to that of its EEG counterpart, with the folders and reports covering the same topics, as can be seen in figure 5, except now in relation to how they function with EMG signals.

A black screen with white text

Description automatically generated

*Figure 7: EMG Research folder.*

EMGs are a very crucial step when creating prosthetic limbs. Surface EMG (sEMG) are the most effective and popular method of judgement on muscle activity. EMG methods measure the skin’s electrical activity during repolarization of muscles when electrodes are activated. The EMG signals are linked to the physical and anatomical factors for example, the number of motor units, types of muscle fibres, diameter, depth etc (Wan et al., 2018).

EMG methods for signal acquisition can involve both invasive and non-invasive techniques which will be explained. For prototype development and ethical considerations, a non-invasive approach to this project will be adopted. This would ensure that there are the least number of external factors to consider when testing and implementing sensor functionality.

Invasive techniques are essentially electrodes contacting the inner muscles to record the signals being produced. This can lead to the results being more accurate as it is in direct contact with the muscle membrane, however due to the invasiveness the end-result can lead the user feeling great discomfort when performing movements, so although it is accurate, it might not be the best process when creating an every-day use prosthetic. (Avilés-Mendoza, 2023),

This leads to what was chosen for the project, non-invasive or surface techniques. This method consists of using surface-electrodes and placing them on top of the skin rather than inserting it into the skin. These electrodes can be gel-based which allows the signals to be read without direct insertion into the skin. The main downside of non-invasive techniques is that the signal is going to be more susceptible to noise or become worn-out (Avilés-Mendoza, 2023), although it provides a much more comfortable user-experience with the prosthetic for day to day-use. As can be seen in figure 6 below (a) represents invasive EMG and (b) represents non-invasive EMG (sEMG).

A close-up of a person's arm

Description automatically generated

*Figure 8: Invasive & Non-invasive EMG (Sharma, 2013)*

Once these electrodes are connected, they can now begin the process of properly acquiring the signals. Signal acquisition is the process of acquiring the signals present in the muscles from the electrodes that have been placed. These signals need to be amplified using a bio-signal amplifier as the signal is usually very faint. While invasive EMG is particularly useful for diagnosis of muscle atrophy and genetic defects, non-invasive electrodes are better suited towards muscle function and providing bio feedback.

Understanding skeletal muscle structure is important in learning the optimal positioning of electrodes to maximize the clarity of signals that can be read. The different muscles and various arrangements thereof are explained in Jamal, 2012 and listed below:

* **Parallel Muscles** - are muscles with fibers that are extending parallel to a force-generating attachment point (axis). Examples of this include the bicep muscles, and the groin.
* **Unipennate Muscles** - These are muscles that are angled in one direction, which are relative to the force-generating axis. An example of this muscle is the Digitorum Longus, which can be seen in figure 7.
* **Multipennate Muscles** - Are muscles, where the muscle fibers and force generating axis are oriented at various angles, generally ranging from 0 degrees to 30 degrees. An example is the Rectus Femoris (bipennate), and Deltoid (multipennate)
* **Circular Muscles** - These are muscles that surround an opening to form a closed shape, and an example of this is the mount muscle.
* **Convergent Muscles** - These muscle fibers come together to an insertion point to maximize their force of contraction. Examples of this is the Pectoralis Major

A better view of examples of these muscles can be seen in the figure 7 below:

A diagram of a human body

Description automatically generated

*Figure 9: Muscle Examples Diagram. (Jamal, 2012)*

The reason behind understanding muscle groups is to better understand the muscles within the forearms, where the EMG electrodes can be placed.

An example of electrodes being placed in the ideal location can be seen in figure 8.

Diagram of a human arm with electrodes and an electrical device

Description automatically generated with medium confidence

*Figure 10: Ideal Electrode Placement. (Martinek et al., 2021)*

The electrical signal provided by EMG electrodes is dependent on where the electrode placement is, and because of this, it is important to ensure that the placement of these electrodes is in the same area over consecutive recording sessions.

Once the EMG signals are acquired and sorted into their respective columns from each electrode, the process of filtering can begin to clarify the signals and make sure the fall between expected parameters. Various filtering techniques can be used to remove unwanted noise from EMG signal readings such as High-Pass Filtering, Low-Pass Filtering, and Band-Pass Filtering as follows:

* **High Pass Filter** - High pass filters are used to remove low-frequency components from the seen electrical signal. The term “Cut off Frequency” is the frequency where all frequencies below the cut off are eliminated, and all frequencies above are allowed to move forward. An example of a high-pass filter can be seen in figure 9.

A diagram of a high pass filter

Description automatically generated

*Figure 11: High pass filter (Jamal, 2012)*

* **Low Pass Filter** - A low pass filter is the opposite of a high pass filter, in which the frequencies less than the cut-off point are removed. A diagram of a low pass filter response can be seen in figure 10.

A diagram of a low pass filter

Description automatically generated

*Figure 12: Low Pass Filter Response. (Jamal, 2012)*

* **Band Pass Filtering** - A combination of both high and low pass filtering will create band pass filtering. This is usually the most common method of filtering in terms of EMG Readings, due to its simple and fast implementation and efficiency when clearing out noise. A diagram of the band pass filter response can be seen in figure 11.

A diagram of a band pass filter

Description automatically generated

*Figure 13: Band Pass Filter Response. (Jamal, 2012)*

## ***EMG Processing***

Signal Processing is the period after acquisition where signal cleaning and clarification is performed. This allows for more concise and meaningful data to be gleaned from the processed signal, which in turn translates into more predictable and straightforward outputs/commands to be taken from it. This is required as the acquired signal contains natural interference and noise. This cleaning and clarification can be conducted using various software algorithms and libraries such as SciPy, Butterworth and the EMG-Analysis toolkit.

There are various types of noise and interferences that can come bundled with an acquired signal, however, each of these can be targeted and removed by a different algorithm which has been calibrated to seek them out.

Wavelet Analysis allows for the transition from a time-value system to a timescale (frequency) system. This is a popular way of removing abnormalities from a signal as there are multiple methods which can be employed depending on which part of the signal requires analysis.

These methods are namely the Discrete Wavelet Transform method (DWT) and the Continuous Wavelet Transform method (CWT).

The DWT method is a technique to transform image pixels into wavelets, which are then used for wavelet-based compression and coding, particularly useful when you are performing sEMG.

A black and white math equation

Description automatically generated with medium confidence

*Figure 14: General equation for DWT.*

The CWT method uses inner products to measure the similarity between a signal and an analyzing function, which unlike the DWT method is more consistent and less time-consuming.

A math equations on a white background

Description automatically generated

*Figure 15: General equation for CWT.*

**Disclaimer:** The two equations shown above are not fully understood by the author, however, research was done into DWT & CWT and their origins thus meriting a heuristic inclusion.

There are various ways of applying WT as different Wavelets have different time-frequency structures and contain several factors that should be considered when deciding which wavelet to choose. Some of the other most popular options are the Mexican Hat (low-oscillation, complex-valued, wavelet) (Complex Mexican hat wavelet 2021) and the Morlet Wavelet (wavelet composed of a complex exponential (carrier) multiplied by a Gaussian window (envelope)) (Morlet wavelet 2024). Wavelets need to be pre-processed and de-noised for sEMG algorithms for upper and lower limb movement recognition.

Artificial Neural Networks (ANN) are adept at handling non-linear data using non-linear activation functions, such as the sigmoid or ReLU functions and offer various methods for utilization. Among these, the Back Propagation Neural Network (BPNN) stands out as a highly favored approach for training self-learning neural networks.

However, a common challenge associated with NN implementation is the substantial requirement of training data and extensive learning iterations.

Another NN option is the Cascade Correlation Network, employing a supervised learning algorithm for ANN. It boasts several advantages including rapid learning, elimination of the need to predefine size, depth, and connectivity patterns, and suitability for large training sets.

Additional noteworthy algorithms leveraging ANN include:

* Radial Basis Function (RBF): Effective for artifact removal and noise reduction.
* Multi-Layer Perceptron (MLP)
* Focused Time-Lagged Recurrent Neural Network (FLRNN): Offers noise reduction and enhanced accuracy.

Python Libraries are an integral part of EMG signal processing due to their expansive and well thought out features that have become standard QoL (Quality of Life) necessities. They are used in nearly every project that contains EMG signals now, for good reason as can be seen below:

**NumPy (Numerical Python)**

NumPy is a cornerstone numerical computing library for Python, offering support for large arrays and matrices. Its efficient mathematical functions enable seamless operations, making it indispensable for scientific and engineering tasks. With intuitive syntax and extensive documentation, NumPy caters to users of all levels. Its compatibility with other tools fosters collaboration, solidifying its place in the Python scientific computing ecosystem. (NumPy, NumPy)

**SciPy (Scientific Python)**

SciPy, built on NumPy, enhances Python's scientific computing capabilities with modules for optimization, integration, and more. It simplifies complex tasks with user-friendly interfaces and well-tested algorithms. Integration with NumPy creates a cohesive environment, empowering researchers in diverse fields like physics and biology. (SciPy, SciPy)

**Matplotlib**

Matplotlib is a versatile 2D plotting library known for its flexibility and support for various plot types. Its MATLAB-like syntax and extensive customization options make it ideal for creating publication-quality visuals. Matplotlib's integration with NumPy and object-oriented design facilitates complex plot layouts, making it a staple in scientific and data analysis communities. (Matplotlib, Matplotlib)

**Seaborn**

Seaborn, built on Matplotlib, focuses on aesthetically pleasing statistical visualizations. It simplifies complex plotting tasks with built-in themes and color palettes, ideal for quick data exploration. Integration with Pandas Data Frames enhances its utility for structured data analysis. (Seaborn, Seaborn)

**BioPython**

BioPython is a comprehensive library tailored for bioinformatics tasks, offering tools for sequence analysis, annotation, and more. Its modular design and compatibility with other Python tools make it essential for interdisciplinary research in genomics and molecular biology. (BioPython, BioPython)

**MNE-Python**

MNE-Python specializes in analyzing neurophysiological data, particularly MEG and EEG signals. Its user-friendly interface and comprehensive toolset cater to both novice and expert users, supporting the entire data analysis pipeline. (MNE-Python, MNE-Python)

**EMG-Analysis Toolkit**

The EMG-Analysis Toolkit provides a range of tools for processing EMG signals, simplifying tasks like signal segmentation and feature extraction. Its user-friendly interface and modular architecture make it valuable for researchers in biomechanics and rehabilitation. (EMG-Analysis Toolkit, EMG-Analysis Toolkit)

**TensorFlow**

TensorFlow, developed by Google, is a robust machine learning library for building and deploying deep learning models. Its efficient computation representation and support for high and low-level APIs make it popular for various applications. (TensorFlow, TensorFlow)

**PyTorch**

PyTorch, developed by Facebook, offers dynamic computation graphs and an imperative programming style, making it flexible and intuitive for deep learning tasks. Its seamless integration with Python promotes rapid prototyping and experimentation. (PyTorch, PyTorch)

**BioSig**

BioSig is a comprehensive open-source software library tailored for biomedical signal processing. It supports a wide range of signals, including EMG, EEG, ECoG (Electrocorticogram), ECG (Electrocardiogram), EOG (Electro-oculogram), and respiratory signals, offering solutions for various stages of signal analysis. BioSig provides tools for data acquisition, artifact processing, quality control, classification, feature extraction, data visualization, and modelling. It incorporates multiple filtering methods and algorithms for artifact detection, enhancing signal quality. While primarily MATLAB and Octave-based, it offers compatibility with Python and C++, ensuring accessibility across different programming environments. (Vidaurre et al., 2011)

**Pyemgpipeline**

Pyemgpipeline is a user-friendly Python package designed for surface and intramuscular EMG processing. It adheres to internationally accepted EMG processing standards, ensuring consistency in parameter values and processing steps. With a straightforward interface, it simplifies EMG processing into seven steps, including bandpass filtering, full wave rectification, and segmentation. Additionally, it incorporates safety features to prevent muscle activity from exceeding safe limits, promoting safe and reliable EMG analysis. (Wu, 2022)

**NeuroKit2**

NeuroKit2 is an open-source Python package facilitating biomedical signal processing, including EMG and ECG. It simplifies signal processing with high-level functions and validated pipelines, enabling quick implementation with minimal code. Transparent and reproducible, NeuroKit2 encourages exploration and innovation, catering to both experienced and novice users. Its user-friendly nature is augmented by comprehensive documentation, installation guides, and step-by-step resources. Offering a wide range of functions, NeuroKit2 supports data management, signal simulation, event extraction, epoch extraction, and signal processing. (Makowski et al., 2021)

**Grumpy**

Grumpy is a versatile Python toolbox designed for hybrid Brain-Computer Interfaces (BCIs), offering a rich selection of signal processing methods accumulated over 20 years by the BCI community. It encompasses various classification methods, from machine learning algorithms to deep neural networks, facilitating diverse BCI applications. Grumpy supports EEG and EMG bio signal analysis, visualization, and real-time streaming. Notably, it has been utilized for controlling robot arms and prosthetic hands using steady-state visually evoked potentials (SSVEP) and surface electromyography (sEMG), highlighting its applicability in advanced BCI systems. (Tayeb, 2023)

Each of these libraries has very specific uses that make them valuable tools during development of an EMG processing application such as this. Some of them overlap in certain ways and fulfill tasks in roughly the same area. The large number of libraries suggests keeping an accurate list of their specific application areas is crucial to being efficient. Of the above, SciPy, Matplotlib and NumPy are the most relevant to the project.

This section highlights the pivotal decision to pivot from EEG to EMG after identifying the limitations and complexities associated with EEG in the context of prosthetic development. This transition is supported by a detailed examination of EMG's suitability for the project's goals, focusing on non-invasive techniques to optimize user comfort and signal fidelity. The research undertaken provides a solid foundation for subsequent development phases, ensuring that the project is built on rigorous scientific understanding and is tailored to meet the specific needs of prosthetic users. This conclusion serves to reiterate the importance of the initial research phase in setting the direction for the project and ensuring that the methodologies chosen to align with the project's objectives and constraints.

# ***3. Requirements***

## ***3.1 Functional***

### ***3.1.1 Signal Acquisition***

The signal acquisition side of the code should be able to record and display signals in a clear and concise manner. These signals, while not filtered, should be organized appropriately. For example, if there are three muscle signals being recorded, there should be 3 columns of data either coming in live or organized in a dataset. This allows for easier and more efficient processing as the data can be pulled from pre-established locations and accurately segmented. Utilizing surface electromyography (sEMG) sensors due to their non-invasive nature, ease of use, and ability to provide sufficient signal resolution for prosthetic control would be the best solution for real time acquisition. These sensors must have a high signal-to-noise ratio (SNR) to minimize the need for extensive filtering and to ensure the fidelity of the muscle activity being recorded, they then should be strategically placed on the forearm to capture the dominant muscle activities that correlate with the intended limb movements. The design of the acquisition system to reject interference from external sources such as power lines and wireless devices is a must. Implementing shielding techniques and differential amplification to enhance the robustness of the signal against environmental noise would be furthered by using electronics such as a Grove shield which is compatible with an Arduino to minimize noise and enhance the signal. This is called pre-amplification to boost the EMG signals immediately at the source, reducing the risk of signal degradation over transmission paths.

Specifying the requirements for analog-to-digital converters (ADCs) that transform analog EMG signals into digital form is essential also. The ADCs should have a resolution of at least around 12 bits and a sampling rate sufficient to capture the highest frequency components of EMG signals, typically around 2000 Hz.

The data must then be displayed in a readable manner, allowing for segmentation and gesture recognition to be tested with discernable results upon iterations. This means that clear visual graphs must be implemented that allow for important parts of the acquisition process to be observed as it goes. This would include parts such as comparing each muscle’s wavelength against each other, looking for outlying spikes or data drops, comparison of pre-processing vs. raw signals and most importantly, separated parts of the data that contain gestures. Including a calibration phase where the signal acquisition system is tuned to each user’s specific EMG signal characteristics would improve the accuracy of signal interpretation. Sensors must be capable of functioning effectively in the presence of moisture and sweat, which are common during regular use by an individual.

Alongside this efficiency and time constraints mean that the data must be retrieved or acquired with as little delay as possible so that examination and investigation of the results is not hindered. In doing so, proper modularity and flexibility procedures should be observed wherein thresholds can be changed or set values swapped out, depending on the requirements of a user or dataset.

### ***3.1.2 Signal Processing***

Real time processing is a crucial element in the pipeline of acquisition to processing to output as without it, a prosthetic limb couldn’t function with the required efficiency or standard as those that it is designed to replicate.

* **Throughput:** The processing unit must handle data streams from multiple sensors simultaneously, supporting at least a 2000 Hz sampling rate per channel to capture the full dynamics of muscle activity.

The very definition of EMG signal processing lies in its ability to enhance the signals features and reduce noise and inconsistencies that are inherent when using sEMG. Implementing digital bandpass filters with cut-off frequencies specifically tailored to the typical frequency range of human muscle signals, approximately 20 to 450 Hz, will help in attenuating noise and irrelevant frequencies that do not contribute to the muscle signal.

Clarity of the data can also be impeded by impurities that are carried through outside factors; this can seriously hinder the processing of data due to an unexpected data flow. Developing algorithms capable of identifying and mitigating artifacts caused by motion or electrical interference will aid in ensuring the purity of the EMG data.

When looking at EMG data and finding gestures or movements it is called feature extraction, this is typically handled by algorithms and when developing these processing systems, it is imperative to remember that each users’ signals can be different and so adaptability is a must in the parameters that are set for these algorithms. Employing algorithms that efficiently extract meaningful features from the EMG signals, such as root mean square (RMS), zero crossings, and waveform length, are crucial for determining the intensity and type of muscle activity. The system should then adapt to individual user variations in muscle activity patterns through a calibration process that personalizes the feature extraction algorithms to the user's specific muscle properties and signal characteristics. This ensures the best possible outcome and carry-through of the original muscle intentions. While this may not be entirely possible given the timeframe and work required to create a personalized system it can also be archived as a potential future addition to the functionality to further increase user experience and efficiency.

Once the signal is finished being cleaned and prepared it should then be converting the processed EMG features into commands that precisely control the movement of the prosthetic limb. This involves mapping specific patterns of muscle activity to corresponding actions in the intended prosthetic device. Incorporating a feedback mechanism that adjusts the processing parameters in real-time based on the user's control accuracy and comfort is ideal, as it proves to act enhancing the user experience and effectiveness of the prosthetic control. As was mentioned in the previous paragraph, this will most likely require work past the deadline for this project, however it is good practice to be able to look into the future at ideal scenarios while maintaining base functionality.

To be fully reliable the system would have to achieve a minimum accuracy of 51% in the translation of EMG signals into prosthetic movements under typical usage conditions. The measurement of this could be done through a testing phase by listing and executing movements or commands and then recording if this had the desired effect on the output and what extra observations could be seen. Maintaining consistent performance over time, with the system capable of self-diagnosing and alerting users to potential issues in the signal processing would ensure trust and reliability in the system. The reason for aiming to 51% accuracy is that the system would then be showing a pattern of predictable outcomes which is exactly the point of a prosthetic in mimicking the original limb. Once this minimum goal is achieved work can then begin on gradually increasing reliability through efficiency and innovation of the functions.

### ***3.1.3 Signal Output***

The output of the signal processing stage in an ideal world is an actual prosthetic that would mimic whatever muscle movements are acquired and then sent through the system. If this is deemed not feasible it would also be appropriate to display results through a microcontroller such as an Arduino, which could then light up LED’s that correspond to each finger or muscle that the signal is being sent from. This would work as proof of concept as from there a full prosthetic would simply be a matter of time and finance.

If a prosthetic is achieved it would most likely be through the medium of 3D printing alongside motors attached to strings. An Arduino could then be attached to carry out actionable commands within the prosthetic.

If the Arduino route is used it would then be a matter of using a breadboard, alongside transistors, jumper wires and multiple LEDs to display and prove that the signals are being accurately translated throughout the whole system.

For the final step of the project pipeline, a machine learning algorithm/model would need to be used to execute the classification algorithm and form actionable commands based off the processed signals. Extreme Gradient Boosting (XGBoost) is an ensemble learning algorithm that has emerged as one of the most effective and versatile tools in machine learning. Originally proposed by (Chen and Guestrin, 2016), XGBoost has gained widespread adoption in both academic research and industrial applications due to its scalability, efficiency, and superior performance compared to traditional machine learning algorithms. This would make it a good candidate for use in the classification algorithm.

## ***3.2 Non-Functional***

### ***3.2.1 Usability***

Efficiency within the processing code is extremely important as it means the least amount of latency possible when running signals through the entire process. The system should process input signals and execute corresponding actions within a few seconds to ensure real-time responsiveness. While not immediately necessary during development the prosthetic should optimize power consumption, with a battery life capable of supporting at least 18 hours of continuous use on a single charge to accommodate daily activities without frequent recharging.

The overall system latency, from signal acquisition through processing to action, must not exceed a few seconds to ensure that the prosthetic feels like a natural extension of the user’s body. Implementing immediate feedback mechanisms, both tactile and visual, would help to inform the user of the prosthetic’s status and any required interactions.

Achieving a control accuracy rate of 95% or higher ensures that the prosthetic movements accurately match the user’s intended motions as interpreted through EMG signals, movements performed by the prosthetic should be repeatable with minimal deviation, ensuring consistent operation under similar conditions.

The system must be able to detect and manage operational errors automatically, accurately giving feedback on what went wrong and where, with as much detail as possible. This can be done through error handlers spread at important intersections throughout the code, whether in acquisition, processing, or the output.

This requirements section meticulously establishes the foundational criteria for the development of an EMG-based prosthetic system, prioritizing clarity, accuracy, and user-centric design. It outlines the essential specifications for signal acquisition to ensure that raw data is captured efficiently and with high fidelity, setting the stage for robust processing algorithms that minimize noise and enhance signal quality. The section further highlights the critical importance of real-time processing capabilities, which are vital for the seamless operation of prosthetic devices. By addressing both functional and non-functional requirements, this section not only sets clear expectations for system performance but also underscores the necessity of a user-friendly interface that can adapt to real-world conditions. Overall, these requirements serves as a blueprint that will drive the engineering and testing of a prosthetic system aimed at providing users with reliable and intuitive assistive technology.

# ***4. Design***

The design section of this project was not overly complex as due to the nature of the area it focuses more heavily on research and development/implementation alongside continuous testing. To explain the initial testing data used, however, and the pipeline it follows, details on the testing dataset and flowchart for the pipeline have been included.

## ***4.1 Dataset***

The EMG dataset that was used to test most functions in this project was published by Mendeley Data and is declared accessible and free to use by all if it is cited appropriately. The reference to the dataset can be found in the references section below in case any copyright infringement is questioned.

It is best to use the creator’s own words to explain the structure of this dataset as it provides a comprehensive and complete overview of how it can be translated into the code used for this project. “This dataset contains electromyography (EMG) signals for use in human-computer interaction studies. The dataset includes 4-channel surface EMG data from 40 participants with an equal gender distribution. The gestures in the data are:

* rest or neutral state.
* extension of the wrist.
* flexion of the wrist.
* ulnar deviation of the wrist.
* radial deviation of the wrist.
* Grip.
* abduction of all fingers.
* supination.
* pronation.

Data were (was) collected from 4 forearm muscles when simulating 10 unique hand gestures and recorded with the BIOPAC MP36 device using Ag/AgCl surface bipolar electrodes. Each participant's data contains five repetitive cycles of ten hand gestures. A demographic survey was applied to the participants before the signal recording process. This data can be utilized for recognition, classification, and prediction studies to develop EMG-based hand movement controller systems. The dataset can also be useful as a reference to create an artificial intelligence model (especially a deep learning model) to detect gesture-related EMG signals. Additionally, it is encouraged to use the proposed dataset for benchmarking of current datasets or for validation of machine learning and deep learning models created with different datasets in accordance with the participant-independent validation strategy.” (Ozdemir, 2021)

A screenshot of a computer

Description automatically generated

Figure 16: Dataset main directory with included resources.

Within this dataset a PowerPoint file was provided (Reference Fig. 16 for all resources provided in the dataset) giving a detailed breakdown of the process a participant would have followed when recording the data, including breaks and gifs of movements required. The folder structure of the data itself is split between raw and filtered versions. This was especially useful as it meant that the processing and cleaning algorithm/method outcomes in this project could be directly correlated and compared to those already present in this dataset. This comes into play in later chapters as it meant there was a pre-designated goal/outcome that could be strived towards without any hints being given away.

Alongside this a file was also present that helped clarify authorship and breakdown of specific timings within this project such as a cycle of 4 second rest periods followed by 6 second active periods.

## ***4.2 Flowchart***

A diagram with white text

Description automatically generated with medium confidence

Figure 17: Flowchart of signal interpretation process.

In Fig. 17 above we can see the process that the acquired signals take depending whether they have been gleaned from live electrodes or through a static dataset. These follow their own paths based on their requirements, for example in the dataset where the functions for pre-processing and graphing need to be able to run through the whole CSV and perform their operations on command whereas with live data from the electrodes the signals must be pre-processed and graphed actively as they come in with as little delay as possible.

The design section effectively outlines the structured approach taken to set up the experimental framework for EMG data acquisition and processing. By selecting a well-documented, accessible dataset, the project ensures transparency and reproducibility in testing the designed algorithms and models. The detailed description of the dataset provides a solid foundation for understanding the characteristics of the EMG signals associated with different gestures, which is crucial for the development of accurate classification and prediction systems. Additionally, the incorporation of a flowchart clarifies the sequence of operations—from data acquisition to signal processing—highlighting the project's systematic approach to handling both live and pre-recorded data. This visual representation aids in comprehending the complex process flow and ensures that the project's design phase is aligned with the functional requirements set forth in earlier sections.

# ***5. Implementation/Construction***

## ***5.1 Initial Setup and Configuration***

To begin the setup and configuration of the project it is important to note the code environment setup that all the functionality of this project was carried out in. The IDE chosen was Visual Studio Code (VSC) for its reliability and its integrated extension installer which was made use of for this project. The extensions installed within VSC were as follows:

* Python for support in the code language used.
* Python Debugger to perform stop point testing.
* Jupyter notebook emulator with Jupyter cell tags, these allowed the code to be ran within cells in VSC like regular Jupyter Notebook. By using this technique each cell was able to be tested and ran independently of the rest of the code file meaning individual graphs could be displayed or whole sections could be skipped in the execution process.
* Rainbow CSV to view large CSVs in a well formatted, color coded and clear file within VSC.

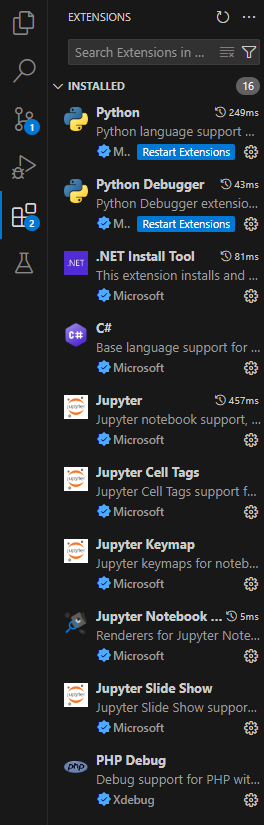


Figure 18: Extensions within VSC.

Another benefit of VSC is its integration with GitHub where you can directly push and pull from a local repository and manage branches. This was used for the project to keep track of progress within different sections of functionality and to have a list of old versions/iterations to look back upon.

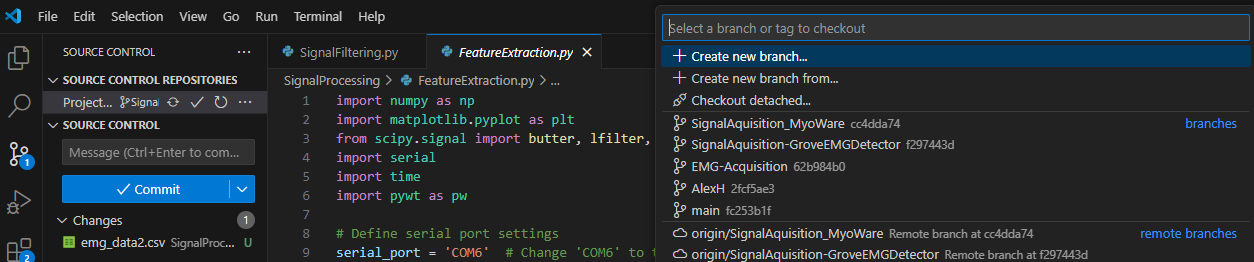


Figure 19: Access to GitHub functionality inside VSC.

Alongside VSC, when communicating with hardware like the Arduino Uno/Grove circuit board, the Arduino IDE is required for a one-time upload of the sketch (script) to the Arduino where it is then stored until changed. This will be explained in greater detail in the latter end of the implementation section.

## ***5.2 Signal Acquisition and Pre-Processing***

In the beginning of the implementation phase a lot of work was put into browsing valuable datasets that could match the requirements of this project. The requirements demanded a signal set that featured multiple channels of EMG data coming in which was then sorted into columns for separate graphing and analysis. Each prospective dataset that was found was brought into VSC and through examples of signal analysis, the beginning of the signal acquisition code began to form. Out of this came the display of raw data that could then be visually analyzed to see if it fit the requirements. Finally, after searching through many different sets the most suitable one was found as will be talked about in the next paragraph.

The dataset used during this phase is that which was mentioned previously, of Mendeley Data. This dataset was comprised of EMG data stored in CSV format, which was sourced from an existing repository of sEMG recordings. This choice was driven by the need for a reliable and consistent data set that could simulate real-world data acquisition in the absence of live experimental data. The following (Fig. 20) is a brief overview of the initial code used to import and inspect the dataset:

A screen shot of a computer

Description automatically generated

*Figure 20: Code for importing and inspecting dataset.*

This code imports the libraries needed for signal acquisition and pre-processing while also reading in a specified CSV from the dataset path. While working with the dataset the signals were primarily taken from one participant that came from the file “1\_raw.csv”. By testing new functionality and methods on the same signals over and over it was easier to judge actual progress in clarity and standard practice.

The functions to graph the raw data that was coming in were created using the matplotlib library, as this in conjunction with the Jupyter Notebook extensions mentioned above allow for enclosed and independent testing of specific graphs without running the whole script.

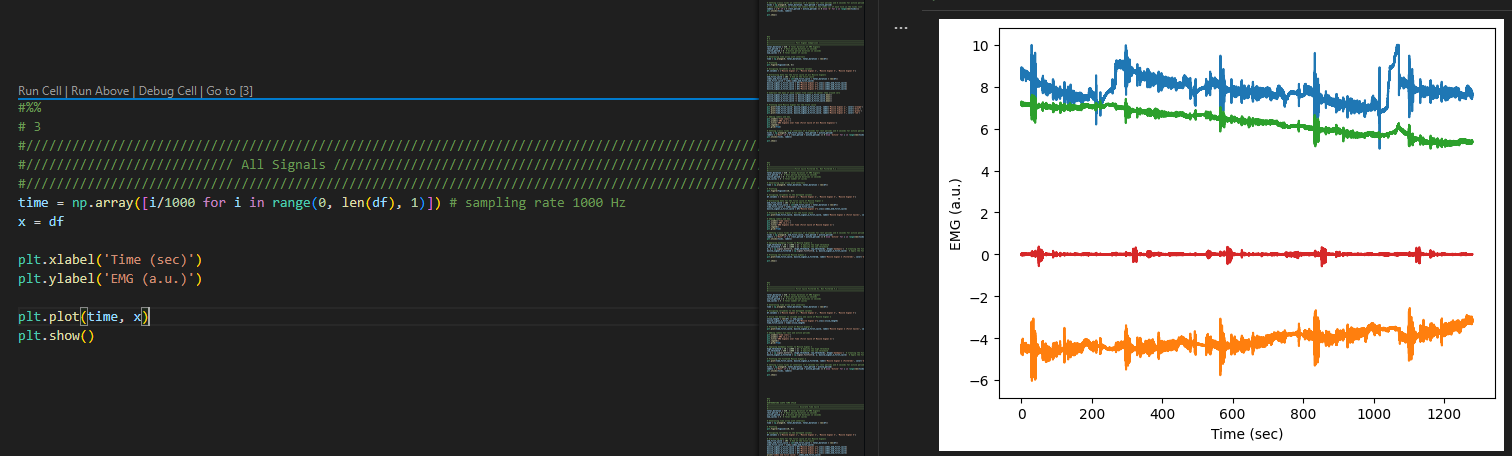


Figure 21: All raw signals from 1\_raw.csv.

As can be seen above in Fig. 21 all the raw signals are being brought into VSC and plotted on a matplotlib graph. This was the first attempt in signal acquisition through a dataset and while the timescale and values were not accurate it was the first step in visual analysis of the data.

Once the raw signals were being brought in and graphed it was then time to fully understand what was happening in the graph. This meant studying the documentation that came with the dataset and included, a PowerPoint detailing the gesture cycles of each participant and their signals, a VSC file containing specifications of timing so that times could be accurately judged when graphing.

In summary, the time cycle of the dataset consisted of:

* 10 gestures per cycle of which there was 5 cycles.
* Each cycle lasted 104 seconds.
* Each cycle began and ended with a 4 second rest period.
* In addition to the 4 second rest periods, after every rest period would come a 6 second active period during which one of the 10 gestures was performed by the participant.
* These 4 and 6 second periods would alternate for the whole cycle until 104 seconds was reached which by then the 10 gestures were complete, meaning the entire 5 cycles took 520 seconds to run their course.
* When the shape of the dataset/csv was gotten it read out as 1,279,999 rows in 4 columns which represent the electrodes.

The columns in the dataset did not come with labels, instead they were simply assigned the averages of their signal values, to counteract this and make it easier to understand variables were assigned to the data path columns under muscle signals 1,2,3 and 4.

The next steps after gaining an understanding of the dataset were to continue progressing with the graphing so that it could be mapped properly, and the cycles and gestures can be easily distinguishable. The first new feature to add was time labels on the graph.

Adding these time labels was done by setting x-axis ticks at intervals of 4 seconds for rest periods and 6 seconds for active periods.

As can be seen in Fig. 21 it is very hard to decipher the individual cycles of the data which led to the decision to isolate the first cycle and experiment with filtering and graphing on that which could then be applied to the whole length after.

To compare the first cycles of all the raw signals together the mean of each signal was subtracted from itself which then centred them around zero. In addition to this since it was known that one cycle was 104 seconds that meant a cutoff point could be defined so that only 0-104 seconds would be displayed as can be seen in Fig. 22.

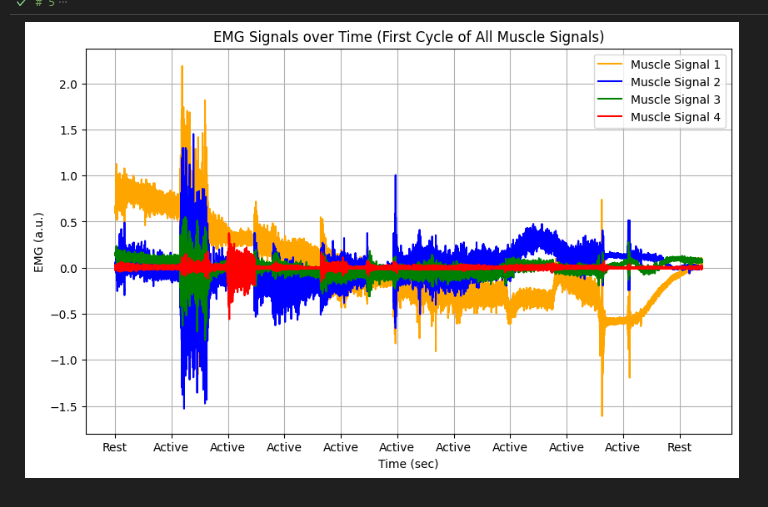


Figure 22: All raw signal first cycles compared.

Once the raw signals were isolated to the first cycle and compared the next step was pre-processing by doing basic filtering. This was done by applying a bandpass filter to Muscle signal 2 with high threshold 20 and low threshold 450. Once this was created and applied the signal began to look far more organized and linear meaning that the filtering was working to create a resting average that would allow for clarification between resting and active periods later in feature extraction. As can be seen in Fig. 23, the values of the graph are being read accurately in Millivolts (mV) on the y-axis despite the labelling of what measurement is being used wrong. Here we can also see that the timeframe labels for resting and active periods are not accurately alternating, issues with the timeframe proved to be a consistent issue early in the acquisition stage.

A graph of a blue and orange signal

Description automatically generated

Figure 23: Filtered vs nonfiltered first cycle.

Through continuous revision of the tick system and generation of the time array eventually the inconsistency of the rest and active periods was resolved as can be seen in Fig. 24. However, it seemed a slight misalignment remained which would not be understood or solved until the end of the acquisition stage.

A graph showing a blue wave

Description automatically generated with medium confidence

Figure 24: Alternating rest and active periods.

Throughout the research of EMG graphing standard practice there was a trend of words such as EMG rectification and enveloping that kept appearing. This turned out to be the best method way to graph EMG signals in preparation for eventual feature extraction and classification. EMG rectification is the term used for when only the positive values are being read into a graph so that a classification algorithm can tell between a rise in activity and a low resting rate. Enveloping the data then means taking out the mean of the data so that only the average remains. This provides a smoother signal wavelength for visual analysis or for a classification algorithm to follow and execute feature extraction as all less valuable data has been removed leaving only the median values as can be seem below in Fig. 25.

A graph with blue lines

Description automatically generated

Figure 25: Filtered, Rectified and Enveloped signal.

Details on how signals were captured and pre-processed using improved methods and refined code. (EMG\_Finalised)

Upon further investigation into the documentation of the timeframe of the dataset it was found that there had been a side note that was missing which was crucial to the timeline of the signals. It stated that after each of the first 4 cycles in the signals there was a 30 second rest period in addition to the standard 104-second-long cycle. This meant that a revision of the time array was needed so that this new addition could be inserted successfully. As the time frame was being re-worked this also provided opportunities to reformate the code.

This new reformat meant standardization of time variables at the start of the code which was clear good coding practice that had been bypassed in favor of being inserted within each code block. The reason for this was as the code had always been organized into blocks that also represented Jupyter Notebook cells it became important to have snippets of variables in each so that they could be run independently. However, as was seen because of this the file would suffer in consistency.

Once the reformat was complete a finalized version of the progress of the dataset acquisition functions was created. This included three graphs detailing the time frame of the rest and active periods (Fig. 26), the first cycle of all raw signals overlapped for reference (Fig. 27) and a filtered vs nonfiltered first cycle signal (Fig. 28).

A graph of colored lines

Description automatically generated

Figure 26: Timeframe of accurate rest and active periods on signals.

A graph of a diagram

Description automatically generated with medium confidence

Figure 27: First cycle of all raw signals overlapped.

A graph showing a number of muscle signals

Description automatically generated

Figure 28: Filtered vs. nonfiltered first cycle of signal 2.

Once the finalised version was formatted correctly and all bugs/errors taken care of it was then time to put the final capstone on the dataset acquisition section. This included combining all functionality that was worked on and could possibly be useful moving forward with processing and the output.

The completed version of the acquisition file contains 5 code blocks, each containing similarly grouped functionality for clarity and good practice. These blocks are organised as follows:

* Block 1: Data Import and Preprocessing.
* Block 2: Signal Processing.
* Block 3: Signal Analysis.
* Block 4: Signal Plotting.
* Block 5: Utilities.

Each of these blocks (except block 1) is followed by a sequence of testing code which serves to call the functions located within the blocks and graph their output.

The outputted graphs for each block are displayed below alongside an explanation of their function.

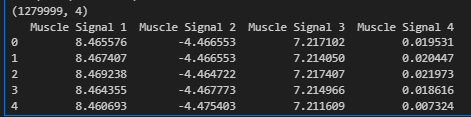


Figure 29: Block 1 output data.

Fig. 29 above shows the shape of the signal CSV that is being imported along with the labels that the code assigns the columns to bring a standardized naming convention to the data path. For a numerical example of the raw data the code extracts the first ten CSV entries from the data frame that the signal is now in and displays them in the output for visual analysis of the values that can be expected.

A group of colorful lines

Description automatically generated with medium confidence

Figure 30: Block 2 Processed signals.

Block 2 is the culmination of the jump between basic pre-processing into full processing to prepare for classification and feature extraction. As can be seen in Fig. 30, this block showcases the raw EMG signal as it is received, then goes through bandpass filtering using butterworth and filtfilt, is rectified to retain positive values and then in packaged as the Envelope of the signal displaying the averages for future analysis and manipulation.

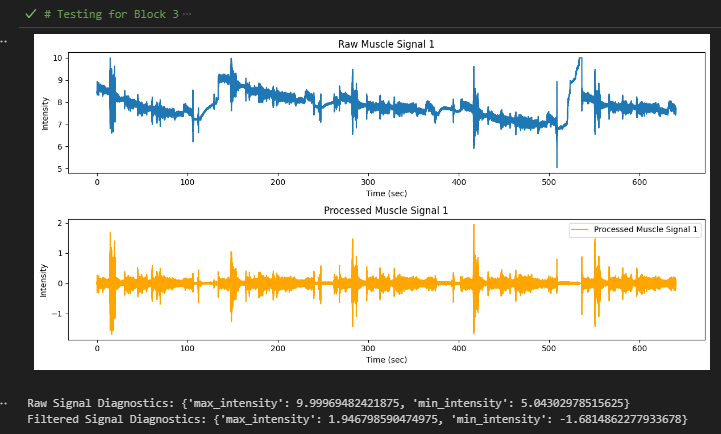


Figure 31: Signal Analysis with diagnostics.

For Block 3 an emphasis was placed in visual and diagnostic analysis. This meant that from a glimpse you can see in seconds the exact time each part of the signal is taking along with exact mV values after processing. Below this graph the block also prints out diagnostics for the exact maximum and minimum intensity of the raw and filtered signals for reference.

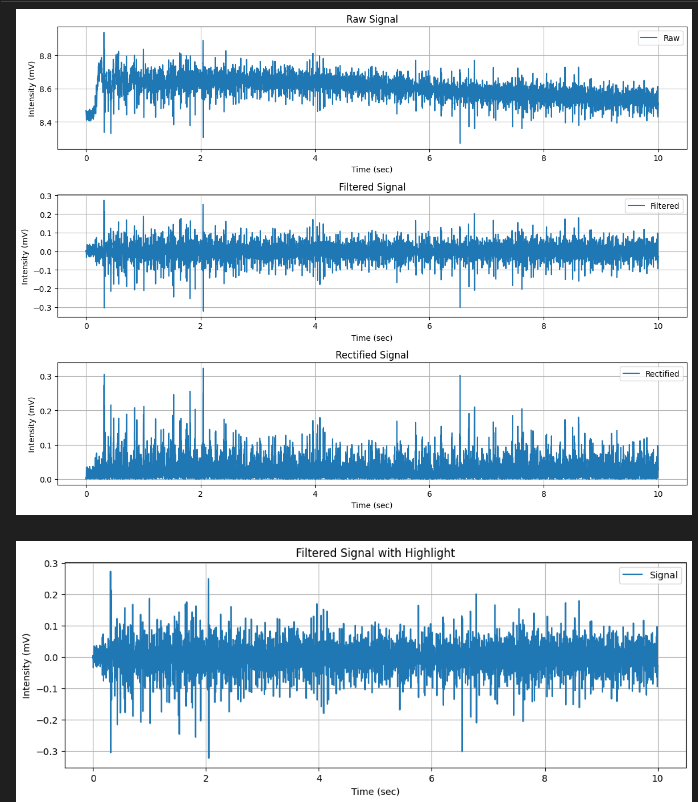


Figure 32: Block 4 Signal Plotting.

Block 4 was focused on showing the product of the graph plotting functionality that made up a large part of acquisition. As can be seen in Fig. 32 the graph goes through the same stages as that of block 2 however the difference is that now there is specific control over what portion of the signal should be displayed for analysis. In the example the seconds show that this section is the first 10 seconds of the signal and shows in detail the signal strength/consistency variation. In this graph however there is also a function to highlight a section of the bottom graph for illustration purpose however it seems it is not functioning as expected.

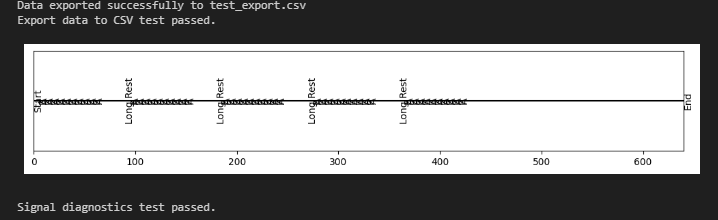


Figure 33: Block 5 saving and time display utilities.

Block 5 is where any functionality that is useful but does not fit reasonably within the other blocks goes. Specifically, it contains the signal section save function and the example time array function. The signal section save function works by taking in a hardcoded definition of a start time and end time within a signal and saves that potion to an external CSV which could then be imported again for separate analysis or experimentation. This would be particularly useful if a portion of a signal would be needed for clinical analysis. The example time array function unfortunately is not fully functional. It includes alternation of rest and active periods alongside a 30 second rest period. The 5 cycles can also be seen represented however it does not reach the full way to the end of the 640 second period. This time function is intended to explain the layout of the alternating periods without needing a signal to attach it to.

This is the culmination of the acquisition, pre-processing, and processing code for the dataset. While getting this far with it proved enlightening about techniques and methods the original goal of this project was to graph and process live data for an output. Towards the latter end of the project when the dataset functionality was satisfactory, focus was shifted towards the same pipeline as the dataset except now using live data from electrodes.

## ***5.3 Integration and Control System Development***

There were two options used for live acquisition and when processing these signals. These were the Myoware Sensor and the Grove Shield Sensor. The Myoware Sensor was the first sensor tested with bringing in live data. As can be seen in Fig. 34 below a sensor is a microcontroller that amplifies signals to be readable.

A red circuit board with black and yellow components

Description automatically generated

Figure 34: Myoware Sensor.

The Myoware Sensor is connected to electrodes that are adhesive and stick to the forearm muscles on top of the skin (sEMG) like those that can be seen in Fig. 35.

Several electrodes with wires

Description automatically generated

Figure 35: Myoware Electrodes.

The sensor is powered by 2 9v batteries which are connected to it by 3 stripped and twisted together 9v battery connectors. The batteries can be seen in Fig. 36 below while the connecters can be seen in Fig. 37.

A battery with a black cover

Description automatically generated with medium confidence

Figure 36: Two 9v batteries to power Myoware sensor.

Several wires with a pen

Description automatically generated with medium confidence

Figure 37: 3 Stripped and twisted 9v battery connectors for Myoware.

Once the sensor can be powered it is then connected to an Arduino Uno (Fig. 38) which can transfer the signals to the computer through a USB B (Fig. 39)

A black circuit board with many small chips

Description automatically generated with medium confidence

Figure 38: Arduino Uno (Elegoo

A close-up of a cable

Description automatically generated

Figure 39: USB B connector.

Once the sensor was connected to the computer using these components combined, it was then time to receive the signals and begin graphing them.

To accomplish this, first the serial port needed to be initialized to where the sensor was connected which for the project was usually defined as COM6. After this it was important to limit the speed at which the hardware and software was communicating and so the Baud rate had to be set to 9600. As it was important to see if the correct live data was being received a save function was implemented in the new live code which would automatically save incoming sensor data to a newly created CSV as can be seen below in Fig. 40.

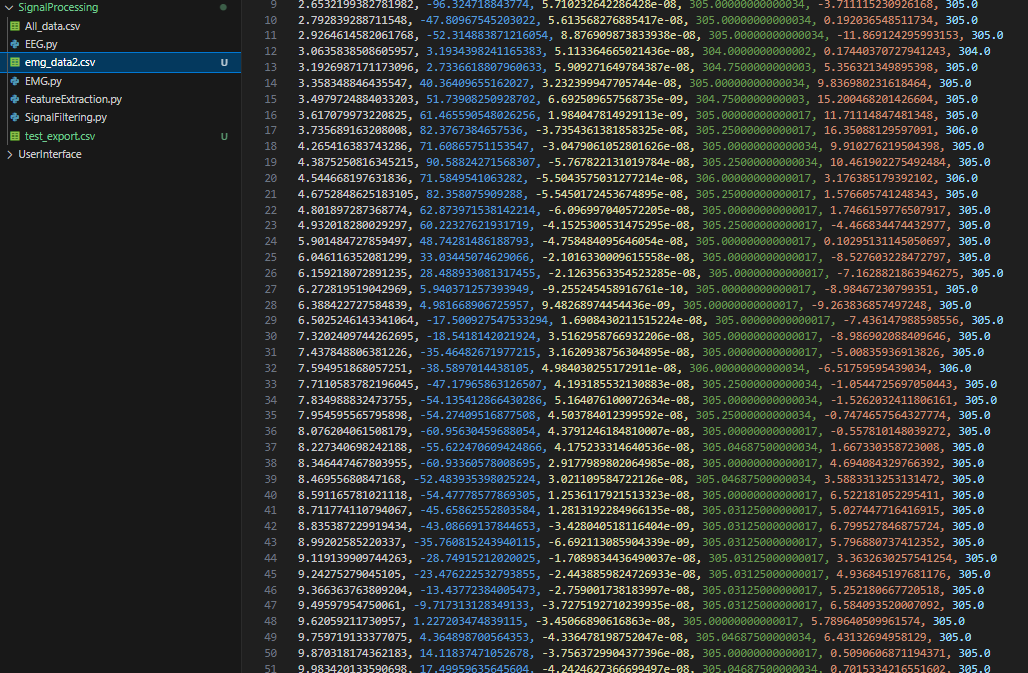


Figure 40: Live data being saved to CSV.

Using the Rainbow CSV extension that was installed before its clear that the data is being organised into the columns/categories provided for it based in the stage of filtering it is at.

Due to the work largely already being done from the dataset it was a similar process to implement the existing project filtering processes to the live data. It works by rapidly handling the raw data coming in from the sensor and passing it through different filters while recording the values after each.

From Fig. 40 the columns are as follows from left to right:

* Time.
* Bandpass.
* Buttered Filtering.
* DWT (Discrete wavelet transform).
* Mix of all filters together.
* Raw data.

As these filters were calibrated already from before the functionality that allows it to filter the live data could simply be copied to each.

Once sufficient data had been collected the live data stream could then by stopped by pressing Ctrl + C.

While Myoware proved to be useful for taking in data and performing the basic functionality, the data still needed to be graphed. This is where the Myoware sensor was swapped out for the Grove Shield.

The Grove Shield is a sensor that comes coupled with a large circuit board which contains various functional components pre-attached. However, unlike the Myoware, neither an external battery source nor Arduino Uno is required. This is due to the Grove technology directly connecting to the computer to draw power while also having an integrated Arduino Uno already within the circuit.

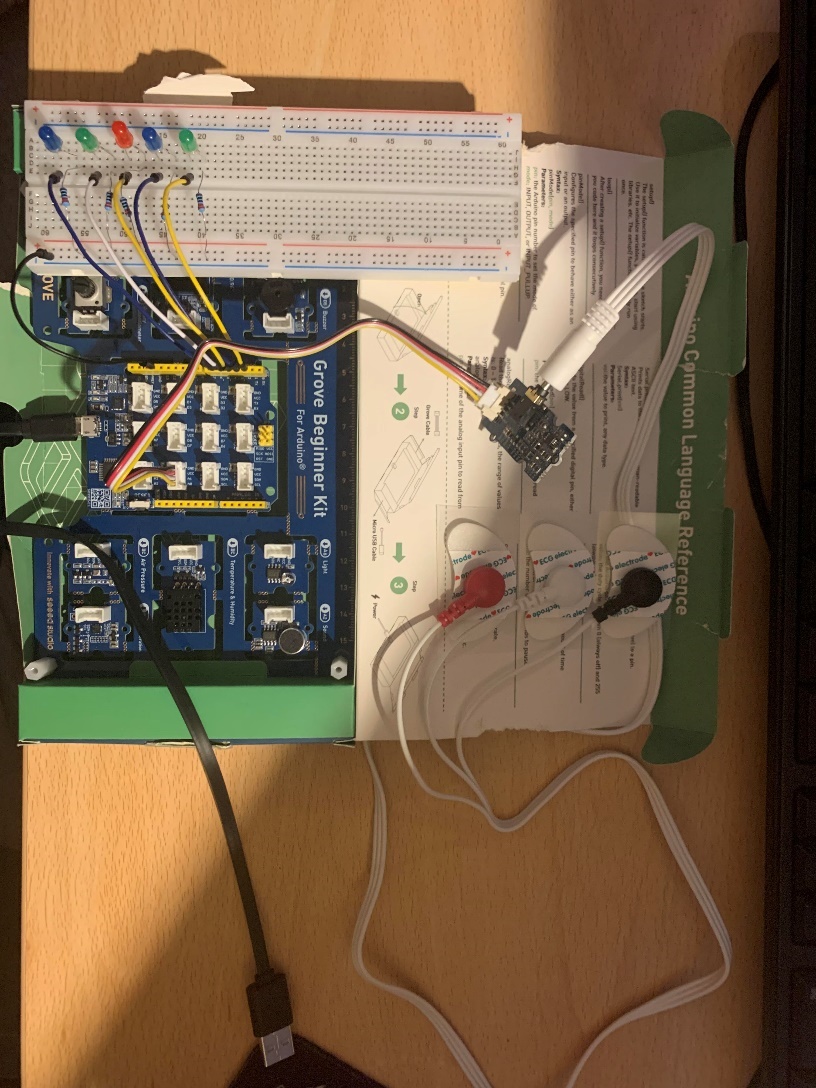


Figure 41: Grove Circuit for live signals.

In Fig. 41 above, there are multiple components connected to the main blue Grove circuit board. From the bottom up they are:

* USB Computer Connector for power and data.
* 3 Electrodes to detect signals.
* Grove Shield microcontroller to amplify signals.
* Main Grove circuit board.
* Testing LED Circuit.

Everything in Fig. 41 is for the purpose of live signal acquisition except for the testing LED circuit, whose purpose will be explained in Feature Extraction and Data Analysis in the next section.

Since the Grove is all connected and ready for the input of live signals all that remained was to tailor the code to the Grove instead of the Myoware sensor, by indicating that the signals would be coming from connection A0 on the grove circuit board to COM6 in the computer and then add the graphing data. The graphing data was designed to automatically move the x-axis forward to keep up with the rate of new data coming in but also to scale the y-axis roof based on the largest value that was last to come in, this ensures that all data can be graphed properly without any cut-offs. While the graph would be moving live as data comes in, Fig. 42 below shows the structure of the graphed data in relation to the signals columns as discussed before.

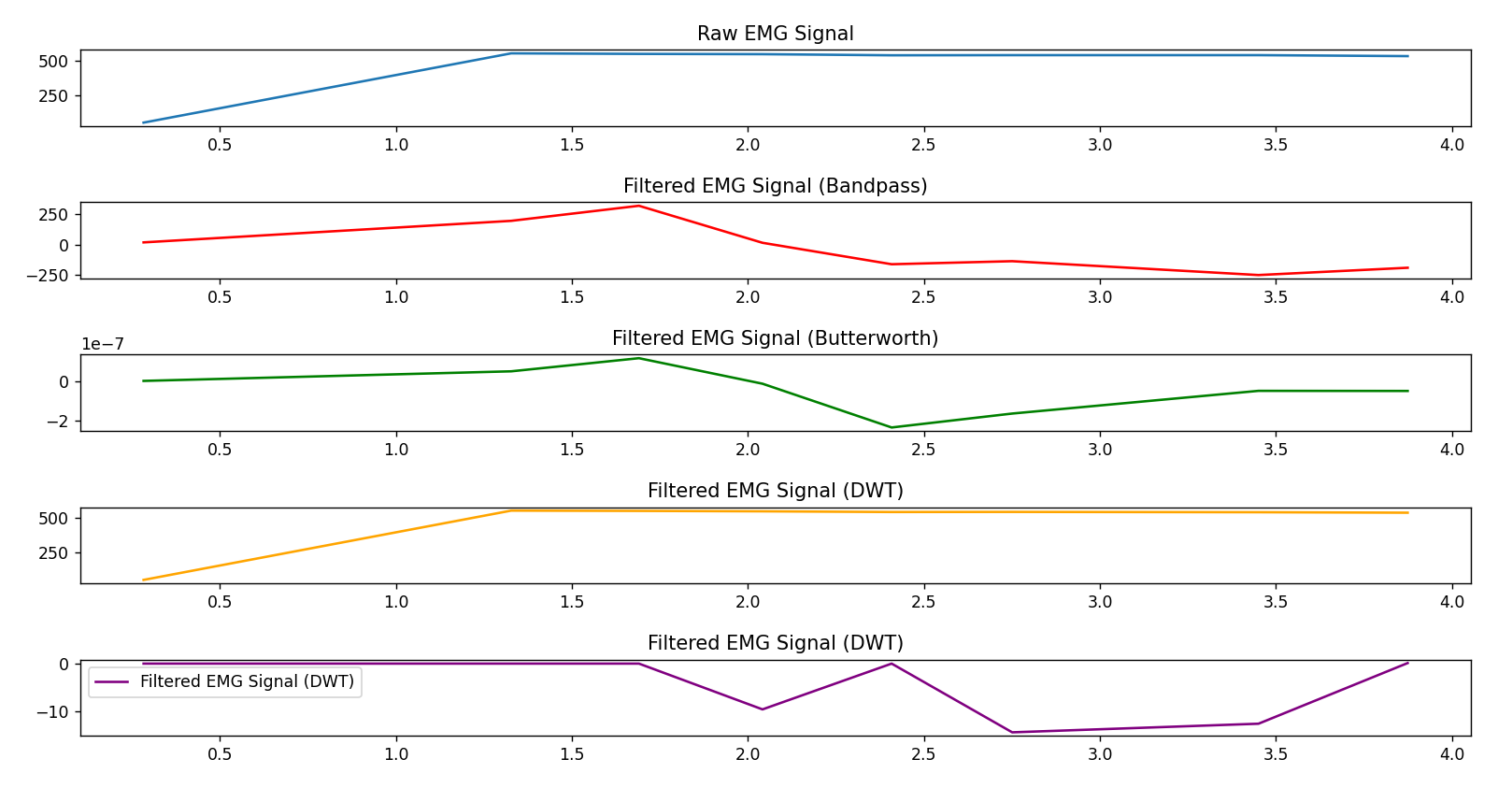


Figure 42: Live EMG data graphing.

Since the live signal has been acquired and processed all that remained to complete the requirements for the project was feature extraction using a classification algorithm to give the processed signals the ability to issue actionable commands to the hardware output circuit.

## ***5.4 Feature Extraction***

The output that the live processed signals would be controlling was the circuit seen earlier in Fig. 41. This circuit had been prototyped using the dataset before live data was available by connecting it to a simple Arduino Uno and an RGB. Using a function that ran through the signal in the dataset automatically the code would send commands to the Arduino to enhance the brightness of an RGB based off signal intensity. Once that worked the prototype was moved on to include a breadboard, jumper wires, 220 Ω (ohm) resistors and 5 LEDs as can be seen in Fig. 43 below.

A circuit board with wires and wires

Description automatically generated

Figure 43: New prototype circuit.

The purpose of this new circuit was to prepare to receive actionable commands from feature extraction functionality which would tell the Arduino which finger has moved and then order it to light up the corresponding LED for that finger (Arduino eventually replaced with the Grove circuit board). While this circuit was prepared before feature extraction was attempted the connection and theory of the circuit still needed testing, to accomplish this a test mode for the Arduino was created. Once the script for the Arduino was ran and communicated with VSC it prompts the user to enter a letter based on what mode they wanted to initiate, if the user entered any letter other than T it would continue its normal function waiting for commands from the feature extraction. However, if T is pressed, the Arduino switches to test mode in which the LEDs can now be directly controlled via the number keys 0-4, with 9 being used to exit the program. This functionality helped ensure that the circuit worked as expected and was prepared for the live data commands that were intended for it.

For all Arduino Uno/Grove circuit boards talked about, they required a one-time upload of their function code through the Arduino IDE to work. Once this code was uploaded it was stored in the microchip on the board meaning all that’s needed thereafter is the VSC script unless ports are modified.

The feature extraction is the only part of the project that remains to be fully completed. Currently the Extreme Gradient Boosting (XGBoost) machine learning algorithm functionality is implemented however there is an error to do with the classification algorithm needed to decipher the actionable commands from the live signals that has not been solved currently.

XGBoost builds upon the principles of gradient boosting, a machine learning technique that sequentially combines weak learners to create a strong learner. It minimizes a loss function by adding new models that predict the residual errors of the previous models. The key innovation of XGBoost lies in its optimization algorithm, which uses second-order Taylor expansions to approximate the loss function and performs gradient descent to find the optimal model parameters (Friedman, 2001).

XGBoost consists of several key components, including:

* Regularization: XGBoost employs L1 and L2 regularization techniques to prevent overfitting and improve generalization performance (Chen & Guestrin, 2016).
* Tree Pruning: It applies tree pruning techniques to control the complexity of individual trees and mitigate overfitting (Friedman, 2001).
* Parallelization: XGBoost implements parallel and distributed computing to accelerate training and inference on large datasets (Chen & Guestrin, 2016).

In the context of real-time EMG signal classification, XGBoost offers several advantages:

* Efficiency: XGBoost's efficient implementation enables real-time processing of incoming EMG signals, allowing for responsive control of robotic prosthetics (Chen & Guestrin, 2016).
* Accuracy: Its ability to capture complex patterns in high-dimensional data makes XGBoost well-suited for accurately classifying muscle activities and motion intentions based on EMG signals (Huang et al., 2019).
* Robustness: XGBoost's robustness to noise and overfitting ensures reliable performance in challenging real-world environments, where EMG signals may be subject to various sources of interference (Chen & Guestrin, 2016).

The implementation/construction section details the setup and use of Visual Studio Code, neatly integrated with essential Python and data handling extensions, to streamline the coding and testing process. This section effectively explained the choice and application of an EMG dataset, illustrating the methods used for data visualization and preprocessing. This setup was pivotal for testing and refining the signal processing algorithms in a controlled environment. The transition from using static datasets to live data acquisition with hardware like the Arduino Uno and Grove circuit board was highlighted, showcasing the practical side of the project. The detailed steps taken to integrate and test hardware components underpin a solid practical application of the project's theoretical framework.

Overall, this section bridges the design and practical execution of the project, laying down a strong foundation for the real-time operation and future enhancements in processing and machine learning.

# ***6. Testing & Analysis***

While a specific testing phase was not performed at the end of this project’s timeline this is because continuous testing was necessary throughout the project so that functionality and methods could be marked as completed.

While that means that user testing and the likes of surveys will not appear in this section it will be used to record bugs or errors that were encountered and then overcome, detailing first the problem then the solution. This will be in addition to their mention in the implementation section.

During the time when using Jupyter Notebook extensions for VSC the formatting became sloppy and no longer follow good practice. This was due to needing instances of variables within each cell/code block for independent testing. Once this was realized a code reformat was conducted now organizing global variables at the top of the script and calling them from inside the cells so that results would stay consistent and could be expanded upon.

## ***6.1 Acquisition***

Within the early acquisition phase of the project, applying accurate time stamps and labels for rest and active periods when graphing proved to be a huge issue that seemed to have no solution. The reason for this, as was later found out, was due to a misunderstanding about how long the signals were captured for. It turned out that while it had been thought that the 5 cycles altogether made up 520 seconds there was in fact an additional 30 second rest period after each cycle except for the 5th one. This meant that the new length of the whole 5 cycles was in fact 640 seconds, once allowing for these 30 second rests. This eventual discovery allowed then for perfect understanding of the timeframe of the signals which solved the issues with labelling and misaligned graphing.

## ***6.2 Pre-Processing and Processing***

Within Block 4 of the completed acquisition code there is an error with the highlighted section of the last graph. This has not yet been solved; however, it is likely that this was caused by a naming convention overlap with another function as it deals with specific signal sections inside a larger signal section.

In Block 5 the time array example function does not work as intended as while the periods are alternating and factoring in the 30 second rest it does not reach the full length of 640 seconds. This is likely due to the sectioning of signals and miscommunication between what length the timeframe should be overall.

## ***6.3 Feature Extraction and Output***

The XGBoost machine learning algorithm is currently stuck when deciphering the commands to be taken from the processed live signals that are being read in. While it can be solved it was not done so by the completion of this thesis.

The testing and analysis section outlines some of the most major issues that occurred during the development process, as the project did not require user testing this section was instead used as a point for showcasing the problems and solutions found during implementation.

# ***7. Discussion***

Through this project I have learned many skills and gained a great understanding of the neural interpretation pipeline for prosthetics.

As this is a field I, as the author, am extremely interested in working in, I believe this has been an invaluable experience as I have had the opportunity to immerse myself in research that directly relates to Bio-Tech/Med-Tech.

I am proud to have progressed to such a complete point within the project and the thesis and feel a great sense of achievement at the depth and detail I have included here.

# ***8. Conclusion***

As we wrap up this exploration of EMG-based prosthetic systems, it's hard not to be inspired by the potential impact this technology can have on improving the lives of those affected by limb loss. From the ancient prosthetics of Egypt to the sophisticated myoelectric arms of today, the journey of prosthetic development has been marked by significant leaps, particularly with the integration of EMG technology. However, the challenges of cost, complexity, and signal interference persist, often making these advanced solutions inaccessible to many.

Throughout this thesis, we've tackled these challenges head-on, aiming to democratize access to prosthetic technology by simplifying the signal acquisition and processing stages. The journey took us through the intricacies of EEG and EMG systems, exploring non-invasive methods that prioritize user comfort without compromising signal integrity. We've navigated through datasets, delved into code optimizations, and experimented with real-time data acquisition setups that bridge the gap between static data and dynamic, real-world applications.

The integration of practical hardware like the Arduino Uno and the Grove circuit board, alongside the theoretical frameworks established, showcases a project that is not only about overcoming technical hurdles but also about ensuring these solutions reach those who need them most. By focusing on reducing costs and enhancing user accessibility, we are opening doors for further innovation in a field where technology meets humanity.

In closing, this thesis isn't just a collection of research and findings; it's a step towards a future where advanced prosthetic limbs are not just a luxury for a few but a widely accessible solution that enhances the quality of life for many. With continued development and a focus on user-centric design, the potential for EMG-based prosthetics is boundless. Here's to pushing forward, innovating, and making the impossible possible for everyone. Thank you for your time, focus and concentration in reading this thesis.

# ***9. References***

The Complete Guide to arm & hand amputations and prosthetics: MCOP. MCOP Prosthetics. (2020, April 1). <https://mcopro.com/blog/resources/arm-hand-prosthetics/#:~:text=How%20much%20does%20a%20prosthetic,the%20latest%20myoelectric%20arm%20technology>.

AJ;, T. (2007, December). Paré and prosthetics: The early history of artificial limbs. ANZ journal of surgery. <https://pubmed.ncbi.nlm.nih.gov/17973673/#:~:text=One%20of%20the%20earliest%20examples,manufactured%20from%20leather%20and%20wood>.

Casson , A. J., Abdulaal, M., Dulabh, M., Kohli, S., Krachunov, S., & Trimble, E. (2017). Electroencephalogram. Seamless Healthcare Monitoring. <http://web.khu.ac.kr/~tskim/HICT%20Lec01-4%20What%20is%20EEG.pdf>

Wan, Y., Han, Z., & Zhong, J. (2018). Pattern recognition and bionic manipulator driving by surface ... Sage Journals. <https://journals.sagepub.com/doi/full/10.1177/1729881418802138>

Avilés-Mendoza, K. (2023). Principles of clinical ethics and their application to practice - PMC. National Library of Medicine. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7923912/>

Jamal, M. Z. (2012, October 17). Signal acquisition using surface EMG and circuit design considerations for robotic prosthesis. IntechOpen. <https://www.intechopen.com/chapters40131>

Vidaurre, C., Sander, T. H., & Schlögl, A. (2011, March 8). BioSig: The Free and open source software library for Biomedical Signal Processing. Computational Intelligence and Neuroscience. <https://www.hindawi.com/journals/cin/2011/935364/>

Wu, T.-L. (2022, April 12). Pyemgpipeline: A python package for electromyography processing. Journal of Open Source Software. <https://joss.theoj.org/papers/10.21105/joss.04156.pdf>

Makowski, D., Pham, T., Lau, Z. J., Brammer, J. C., Lespinasse, F., Pham, H., Sch&ouml;lzel, C., & Chen, S. H. A. (2021, February 2). NeuroKit2: A python toolbox for Neurophysiological Signal Processing - Behavior Research Methods. SpringerLink. <https://link.springer.com/article/10.3758/s13428-020-01516-y>

Tayeb, Z. (2023). Gumpy: A python toolbox suitable for hybrid brain-computer interfaces - DLR. Gumpy: A Python Toolbox Suitable for Hybrid Brain-Computer Interfaces Gumpy: A Python Toolbox Suitable for Hybrid Brain-Computer Interfaces 2. <https://elib.dlr.de/121742/1/Gumpy.pdf>

NumPy. (n.d.). NumPy. <https://numpy.org/>

Scipy. SciPy. (n.d.). <https://scipy.org/>

Visualization with python. Matplotlib. (n.d.). <https://matplotlib.org/>

Statistical Data Visualization#. seaborn. (n.d.). <https://seaborn.pydata.org/>

Biopython. Biopython · Biopython. (n.d.). <https://biopython.org/>

Python homepage#. MNE. (n.d.). <https://mne.tools/stable/index.html>

Tensorflow. TensorFlow. (n.d.). <https://www.tensorflow.org/>

Membership available. PyTorch. (n.d.). <https://pytorch.org/>

Dataset:

Ozdemir, M. A. (2021, December 22). Dataset for multi-channel surface electromyography (SEMG) signals of hand gestures. Mendeley Data. <https://data.mendeley.com/datasets/ckwc76xr2z/2>

**(Reference layout changes due to an update in the citation software used)**

Bao, F. S., Liu, X., & Zhang, C. (2011, March 29). *PyEEG: An open source python module for EEG/Meg Feature Extraction*. Computational Intelligence and Neuroscience. <https://www.hindawi.com/journals/cin/2011/406391/>

Gramfort, A., Luessi, M., Larson, E., Engemann, D. A., Strohmeier, D., Brodbeck, C., Goj, R., Jas, M., Brooks, T., Parkkonen, L., & Hämäläinen, M. (2013, December 9). *Meg and EEG data analysis with MNE-Python*. Frontiers. <https://www.frontiersin.org/articles/10.3389/fnins.2013.00267/full?source=post_page-----58aae90351e5---------------------->

Hu, L., & Zhang, Z. (2019). *EEG signal processing and feature extraction*. Springer.

Krell, M. M., Straube, S., Seeland, A., Wöhrle, H., Teiwes, J., Metzen, J. H., Kirchner, E. A., & Kirchner, F. (2013, December 9). *PySPACE-a signal processing and classification environment in python*. Frontiers. <https://www.frontiersin.org/articles/10.3389/fninf.2013.00040/full>

Kumar, S., & Bhuvaneswari, P. (2012, September 5). *Analysis of electroencephalography (EEG) signals and its categorization–A study*. Procedia Engineering. <https://www.sciencedirect.com/science/article/pii/S1877705812022114?ref=pdf_download&fr=RR-2&rr=823e730f9ab4369a>

Lakshmi, R., Prasad, T. V., & Prakash, C. (2014). *Survey on EEG signal processing methods - researchgate*. Research Gate. <https://www.researchgate.net/profile/Rajyalakshmi-Tvk/publication/328419840_Survey_on_EEG_Signal_Processing_Methods/links/5bcd89b8299bf17a1c662583/Survey-on-EEG-Signal-Processing-Methods.pdf>

Makowski, D., Pham, T., Lau, Z. J., Brammer, J. C., Lespinasse, F., Pham, H., Sch&ouml;lzel, C., & Chen, S. H. A. (2021, February 2). *NeuroKit2: A python toolbox for Neurophysiological Signal Processing - Behavior Research Methods*. SpringerLink. <https://link.springer.com/article/10.3758/s13428-020-01516-y>

Tayeb, Z., Waniek, N., Fedjaev, J., Ghaboosi, N., Rychly, L., Widderich, C., Richter, C., Braun, J., Cheng, G., & Conradt, J. (2023). *Gumpy: A python toolbox suitable for hybrid brain-computer interfaces - DLR*. Grumpy. <https://elib.dlr.de/121742/1/Gumpy.pdf>

Vidaurre, C., Sander, T. H., & Schlögl, A. (2011, March 8). *BioSig: The Free and open source software library for Biomedical Signal Processing*. Computational Intelligence and Neuroscience. <https://www.hindawi.com/journals/cin/2011/935364/>

*Ethics for researchers - die Europäische Kommission*. European Comission. (2013). <https://ec.europa.eu/research/participants/data/ref/fp7/89888/ethics-for-researchers_en.pdf>

Moore, R. A. (2007). *Ethical considerations for Psychophysiology Studies - Sage Journals*. Sage Journals. <https://journals.sagepub.com/doi/pdf/10.1177/174701610700300203>

*Official Legal Text*. General Data Protection Regulation (GDPR). (2022, September 27). <https://gdpr-info.eu/>

St, E. K., Frey, L. C., Britton, J. W., Frey, L. C., Hopp, J. L., Korb, P., Koubeissi, M. Z., Lievens, W. E., Pestana-Knight, E. M., & St, E. K. (2016). *Appendix 3. principles of electrical safety - electroencephalography ...* NCBI. <https://www.ncbi.nlm.nih.gov/books/NBK390355/>

Wikimedia Foundation. (2024, April 25). Morlet wavelet. Wikipedia. <https://en.wikipedia.org/wiki/Morlet_wavelet>

Wikimedia Foundation. (2021, December 3). Complex mexican hat wavelet. Wikipedia. <https://en.wikipedia.org/wiki/Complex_Mexican_hat_wavelet>

## ***9.1 Image References***

Esposito, D., Centracchio, J., Bifulco, P., & Andreozzi, E. (2023, May 12). A smart approach to EMG envelope extraction and powerful denoising for human–machine interfaces. Nature News. <https://www.nature.com/articles/s41598-023-33319-4>

Casson , A. J., Abdulaal, M., Dulabh, M., Kohli, S., Krachunov, S., & Trimble, E. (2017). Electroencephalogram. Seamless Healthcare Monitoring. <http://web.khu.ac.kr/~tskim/HICT%20Lec01-4%20What%20is%20EEG.pdf>

Habibzadeh Tonekabony Shad, E., Molinas, M., & Ytterdal, T. (2020). Impedance and noise of passive and active dry EEG electrodes: A Review. IEEE Sensors Journal, 20(24), 14565–14577. <https://doi.org/10.1109/jsen.2020.3012394>

Sharma, S. (2013). On board EMG Signal Compression for portable devices - A study | IEEE conference publication | IEEE Xplore. On Board EMG Signal Compression for Portable Devices - A Study. <https://ieeexplore.ieee.org/document/6918899>

Jamal, M. Z. (2012, October 17). Signal acquisition using surface EMG and circuit design considerations for robotic prosthesis. IntechOpen. <https://www.intechopen.com/chapters40131>

Martinek, R., Ladrova, M., Sidikova, M., Jaros, R., Behbehani, K., Kahankova, R., & Kawala-Sterniuk, A. (2021, September 10). Advanced bioelectrical signal processing methods: Past, present, and future approach-part III: Other biosignals. MDPI. <https://www.mdpi.com/1424-8220/21/18/6064>

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM. https://dl.acm.org/doi/abs/10.1145/2939672.2939785

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics, 29(5), 1189-1232. https://www.jstor.org/stable/2699986

Huang, C., Zhou, W., Cao, Y., Gao, W., Yang, J., & Zhang, Y. (2019). A Novel Anomaly Detection Method for EMG Signals Based on XGBoost. Journal of Healthcare Engineering, 2019, 1-9.

# ***10. Appendices***

## ***10.1 Appendix A: Development Diary***

This section is simply for holding and referencing the various stages I went through in my development process from January through to the present. It is taken from the diary I kept as I went through development, updating every day and for every major step.

It is structured alongside the timeframe of when it was done so a roadmap can be followed of the progress of the project.

**January 15th to 30th:**

* Browsing valuable datasets for one that matched my requirements.
* Followed examples of signal analysis and attempted the start of my own code, achieved raw data showing, written in Kaggle for testing.
* Found the Mendeley Dataset that would become my choice and listed algorithms I could use.
* Imported the dataset and set up the connection in VSCode.
* Created methods to graph the raw signals and display them using Matplotlib and Jupyter notebook extensions for VS.

**February 1st to 15th:**

* Went fully through the dataset documentation marking down important notes and structure of the dataset, gaining a better understanding.
* Attempted filtering but ran into issues, range was too high, and the values were out of reach.
* Created a GitHub issues tracker for backlog tasks and to track progress of requirements.
* Added timeframe labels to the graphs to accurately depict rest and active periods.
* Separated the time into a single cycle for visual analysis.
* Applied a bandpass filter (process of testing and comparing to dataset filtered signal. Had to use the first cycle to see actual change as the full signal is too large.
* Isolated the single cycle fully and tested pre-processing filtering alongside timeframe adjustments.
* Ran into major issues with timeframe labelling and adjustments as some specifications were missing from dataset documentation.
* Integrated new pre-processing methods to filter while passively fixing the timeframe.
* Timeframe (rest and active periods) now fixed, shows a line between these sections for clarity and visual segmentation.
* Pulled the dataset filtered signal into a graph alongside my pre-processed version to compare differences and make accurate adjustments.
* Determined that there is still an abundance of noise in my data.

**February 15th to 29th:**

* Little functionality was added - mainly bug fixing and testing existing features.
* Considerable experimentation on processing techniques such as classification and filtering.
* Took a 3-day break from new functionality to completely review my code, re-format it professionally and clearly documents each important section and function to make sure I understood every detail.
* Time frame needed some more accurate adjustments as it was not lining up with clearly active periods.
* Implemented accurate millivolt (mV) measurements of the signals and added a rectify function for the signals in case it was decided to implement training for a model.

**March 1st to 15th:**

* Attempted full implementation of rectifying and enveloping the EMG signal.
* Alongside code development began familiarizing myself with Arduino documentation, components, functions, and electronics to show a proof-of-concept outcome from the signals.
* EMG envelope working roughly, needed fine-tuning.
* Fine-tuned the EMG envelope so now it is a clear show of the average signals.
* Watched EMG machine learning tutorials/courses to prepare in case a model was needed.
* Put the code into clear functions that grouped roughly similar code outputs.
* Implemented automatic labelling of the timeframe instead of manual using ticks.
* Began the process of connecting the Arduino to the script for testing inputs.
* Extracted instances of signals for in-depth analysis in a separate file.
* Implemented appropriate sample rate of the signals for testing.
* Perfected the rectified signal, sampling, and intervals. Combined the working functions from my experimental working python file into my production file. The code is now organized into blocks that carry out designated functions. Outlined and began finishing touches for acquisition and prepared to move into processing.
* Updated GitHub to reflect this progress and plan for the next stages.

**March 15th to 31st:**

* Had PC issues and some errors with the Arduino that delayed progress.
* Continued finding appropriate methods of displaying data on the Arduino, homed in on LEDs.
* Outliers in visual analysis of the signal prompted me to re-evaluate labelling, signal accuracy and the timeframe.
* Did a deep dive into the documentation and found a brief mention of a 30-second interval that took place between each cycle that wasn’t stated anywhere else.
* Began working on integrating this interval after each cycle in the timeframe.
* Signal strength now represented accurately (mV).
* Finalized function that allows for exporting data of a specified timeframe into a separate .csv.
* Labelling of the data can be altered when exporting it for better understanding and clarity.
* 30 second intervals are added automatically at the end of each cycle however it is rough and needs improvement.
* Turned the tick counter into a function and made it clearer with additional comments.
* Improved documentation of the whole code.
* Continued progress on improving the 30 second interval.
* Decided that while the function that defaults the signal to the average at the start and names it as 0 is useful, a function that preserves the original starting value is also required.
* Implemented methods to preserve original values as intended and made sure these were also represented accurately when graphed.
* Implemented the ability to comment out the labelling (Rest/Active) function in case just the actual seconds are required to display.
* Put in function to graph the extracted timeframe data in case separate data analysis/experimentation needs to be performed.
* While rectifying the signals, it may also be useful to retain the negative values too for which I put in the ability to comment out the rectifying function and instead just use the filtered signal when graphing.
* Added comments to explain the reasons for rectifying, enveloping and the high/low pass filtering.
* Fixed bugs with cell variables disturbing graphs in latter end cells, also worked through fixing some data values in the graphs such as mV readings and negative values displaying.
* 30 second interval perfected and completed. Working on making sure ticks are completely accurate, reflected this within GitHub.
* Separated some sections of code and added more comments to give a general idea of how to use some of the functions and included optional lines to change the graph.
* Finalized the production acquisition code and the experimental code with re-formatting so that the two could be combined into a finalized version that only needs minor modifications in one file.
* Created a code review video which explains the code – created as a test for when a screencast of the functionality is required.
* Updated comments on the finalized acquisition file so that it flows well between each section/block of functionality.
* Arduino is displaying and RGB light in accordance with signal intensity as it goes through the signal showing integration with hardware/output.
* Commented serial communication methods with the Arduino.

**April 1st to 15th:**

* Wi-fi/power outage for 5 days seriously halted progress and only research on implementing classification algorithms could be done.
* Created a parts list to ensure components were able to display an accurate output of signals, including parts for live acquisition and Arduino outputs.
* Continued processing methods for the signals to make them as clear as possible.
* Began experimenting with live data using a Myoware sensor, encountered difficulties when receiving data from the sensor. Had to fix the Arduino IDE rather than VSCode. The problem was fixed and can solely use VSCode now.
* Created a dataset using live data for testing finger movement functionality, proved ineffective and was redone. Used separate functions to specify resting and active activity, then combine them into one full dataset.
* Dataset now takes in four signals instead of one in columns a, b, c and d alongside Activity and id, which were filtered and rectified. Updated my GitHub to reflect this.
* Obtained the 9v connector for the battery power source. Researched implementation of direct live signals and focused on filtering again.
* Read through Grove and Myoware documentation.
* Created comments explaining functionality used in the classification section of the code.

**April 15th to Present:**

* Read through resources on feature extraction, where to extract time-domain features using the numpy library.
* Researched and examined documentation for Scipy for Butterworth, wavelet transform and adaptive filtering techniques.
* Researched and examined documentation for PyWaveletes for wavelet transform and wavelet-based signal processing.
* Implemented a basic feature extraction but it is not up to standard.
* Attempted to implement more complex filters, having issues with filtering and having the signal at baseline 0.
* Implemented a basic discrete wavelet transform (DWT) filter.
* Signals can now be filtered using both bandpass and DWT filtering.
* Implemented a notch filter.
* Arduino plotting is faster than python plotting. Should review how to improve python speed.
* Attempted filtering of live signals and saving to a csv file, had issues.
* Fixed the saving of signals, can now save signal values after the graph is closed.
* Still fixing live filtering.
* Created a fingers dataset after the save function was fixed and the plotting is more accurate.
* Fixed live filtering.
* Made the plotting more accurate.
* Took time to work on the output of the script to Arduino side to display results.
* Added a simple moving average filter for the signal LED’s give immediate feedback based on the detected finger, put in exception/error handler messages, and maintained a modular approach to the script. Do not have the Grove Shield yet for the Arduino so cannot properly test it yet.
* Added a function that allows you to manually toggle each LED by entering T to test connection (Test mode in the console as an option). Cleaned up commenting.
* Made sure the Arduino sketch lined up flawlessly with the modular approach of the python script with new error handling.
* Updated the python script as I realized some of the functions subtly relied upon a Raspberry Pi Grove library, modified it to make sure it is targeted for an Arduino connection. Testing of the 5 LEDs for each finger is fully functional in test mode. Removed old working code to make it more production ready.
* Researched into a circuit that would allow for LEDs at the end of fingertips to show signals are being received.

Each entry in my development diary may not be separate as it seems here, as they may be grouped based on time and similarity of what was implemented, however this is a complete breakdown of my process throughout the project. I think it would be good to include as much detail as possible here such as snippets where relevant and pictures of hardware progression. It would allow anyone to have a step-by-step roadmap of what happened and when.