IMPERIAL

# Electromyography (EMG) Signal Pattern Classification with a Modular 3D-Printed Signal Acquisition Armband

Author
A. Osman
CID: 02026214

Supervised by
Prof. K. Fobelets

A Thesis submitted in fulfillment of requirements for the degree of
**Master of Engineering in Electrical and Electronic Engineering with Management**

Department of Electrical and Electronic Engineering
Imperial College London
2025

# Abstract

This project presents the design and development of a real-time Surface Electromyography (sEMG) classification system for recognising hand and wrist movements using a custom-built, 3D-printed forearm armband with modular bipolar electrodes. The system employs a wireless OpenBCI Cyton biosensing board to acquire multi-channel sEMG signals, which are processed and classified using Machine Learning (ML) models. Data was collected from multiple participants across a range of physical activity levels and gender, with both left and right arm configurations. The data pipeline included pre-processing with real-time causal filters, windowed segmentation with 128 ms windows and 75% overlap, and feature extraction encompassing time-domain metrics. Feature selection was performed using Random Forest Importance scores, Recursive Feature Elimination (RFE), and feature correlation analysis to reduce redundancy. Multiple classifiers were evaluated, with the best-performing model, an Multi-Layer Perceptron (MLP), achieving 97.49% test accuracy. A lightweight Graphical User Interface (GUI) was developed to display live predictions with minimal latency, using a majority-voting buffer to improve temporal stability. This work demonstrates the feasibility of an accurate and responsive real-time sEMG-based gesture recognition system using a low-cost hardware add-on to the Cyton Board and open-source tools, with potential applications in rehabilitation, prosthetic control, and human-computer interaction.

# Declaration of Originality

I hereby declare that the work presented in this thesis is my own unless otherwise stated. To the best of my knowledge the work is original and ideas developed in collaboration with others have been appropriately referenced. I acknowledge that I used ChatGPT as an assistive tool for refining written text and for code syntax suggestions.

# Copyright Declaration

# Acknowledgments

I would like to express my sincere gratitude to my supervisor, Professor Kristel Fobelets, for her invaluable expertise, encouragement, and extensive guidance throughout this project. Her insightful feedback and unwavering support were instrumental in shaping this work from its initial concept to its final form.

I am also deeply grateful to my friends and family for their constant support and understanding during this demanding year. A special thank you is owed to those who generously volunteered their time to participate in the data collection sessions; this project would not have been possible without your help.

Finally, I wish to extend my thanks to the staff and my peers in the Department of Electrical and Electronic Engineering. I have thoroughly enjoyed my four years at Imperial College London, and I am thankful for the knowledge and experiences I have gained.

# Contents

**Abstract** i

**Declaration of Originality** iii

**Copyright Declaration** v

**Acknowledgments** vii

**List of Acronyms** xiii

**List of Figures** xv

**1 Introduction** 1

1.1 Key Questions and Significance . . . . . 2

1.2 Outline of Report Structure . . . . . 3

**2 Background** 5

2.1 Overview of sEMG signals . . . . . 6

2.1.1 What is sEMG . . . . . 6

2.1.2 Signal Characteristics . . . . . 6

2.1.3 Key Challenges with using sEMG . . . . . 7

2.1.4 Primary Applications of sEMG . . . . . 8

2.1.5 sEMG Machine Learning Pipeline . . . . . 10

2.2 Review of Existing sEMG systems . . . . . 15

2.2.1 Commercial Wearables . . . . . 15

2.2.2 Research Projects & Platforms . . . . . 16

2.2.3 The OpenBCI Platform . . . . . 18

2.3 Case for a Modular sEMG Wearable Add-On for the Cyton Board . . . . . 19

2.4 Hardware and Software Utilised in this Project . . . . . 19

2.4.1 Hardware . . . . . 19

2.4.2 Software . . . . . 21

2.5 Chapter Summary . . . . . 21

**3 Requirements Capture** **23**

3.1 Functional Requirements . . . 23

3.2 Non-Functional Requirements . . . 24

3.3 Chapter Summary . . . 25

**4 Analysis and Design** **27**

4.1 Final Design . . . 28

4.1.1 Final Hardware Design . . . 28

4.1.2 Final Software Design . . . 31

4.1.3 Summary of Python Applications Developed . . . 33

4.2 Design Constraints . . . 34

4.3 Hardware Design Decisions . . . 35

4.3.1 Armband Design Decisions . . . 35

4.3.2 Modular Sensor Design and Integration . . . 38

4.3.3 Bias Placement . . . 38

4.3.4 Characterisation of the Sensor Modules . . . 39

4.3.5 Cost of Materials . . . 39

4.4 Software Design Decisions . . . 40

4.4.1 Data Collection Protocol and GUI . . . 40

4.4.2 Adding Labels to Raw sEMG Data . . . 41

4.4.3 Signal Preprocessing . . . 41

4.4.4 Data Augmentation . . . 44

4.4.5 Feature Selection Process . . . 44

4.4.6 Machine Learning Model Evaluation . . . 46

4.4.7 Real-Time Classification Application . . . 48

4.5 Chapter Summary . . . 50

**5 Implementation Details** **51**

5.1 Dataflow Overview . . . 51

5.2 Data Collection and Labelling . . . 52

5.3 Signal Processing and Feature Extraction . . . 54

5.3.1 Signal Filtering . . . 54

5.3.2 Feature Extraction . . . 55

5.4 Classification System . . . 55

5.4.1 Offline Model Training . . . 55
5.4.2 Real-Time Classification System . . . 56
5.5 Chapter Summary . . . 57

**6 Test Plan and Verification** **59**
6.1 Hardware Verification . . . 59
6.1.1 Sensor Signal Integrity . . . 60
6.1.2 Wiring and Connectivity . . . 60
6.1.3 Wearability and Fit . . . 60
6.2 Software Verification . . . 61
6.2.1 Signal Processing Verification . . . 61
6.2.2 Real-Time Classification Application . . . 62
6.3 Chapter Summary . . . 62

**7 Results** **65**
7.1 Hardware Evaluation . . . 66
7.1.1 3D-Printed sEMG Armband Signal Quality . . . 66
7.1.2 Motion Artifact Quantification . . . 67
7.2 ML Model Results . . . 68
7.2.1 Final MLP Offline Results . . . 68
7.2.2 Final MLP Real-Time Results . . . 70
7.3 Comparison to Existing Solutions . . . 71
7.4 Computational Performance . . . 72
7.5 Chapter Summary . . . 73

**8 Conclusions** **75**
8.1 Evaluation . . . 75
8.2 Project Success and Achievements . . . 76
8.3 Limitations and Further Work . . . 76
8.4 Remarks on Design Decisions . . . 77

**9 User Guide** **79**

9.1 Hardware Setup and Assembly . . . . . 79

9.1.1 3D Printing . . . . . 79

9.1.2 Assembly . . . . . 80

9.2 Data Collection Protocol . . . . . 80

9.3 Training a Classification Model . . . . . 81

9.4 Using the Real-Time Application . . . . . 82

**A Appendix: sEMG Feature Mathematical Definitions** **83**

**B Appendix: Window Size and Overlap Grid Search Results** **87**

**Bibliography** **91**

# List of Acronyms

**ADC** Analogue-to-Digital Converter

**ApEn** Approximate Entropy

**AR** Augmented Reality

**BCI** Brain Computer Interface

**CAD** Computer Aided Design

**CNN** Convolutional Neural Network

**CPU** Central Processing Unit

**DL** Deep Learning

**ECG** Electrocardiogram

**EEG** Electroencephalogram

**EMG** Electromyography

**FFT** Fast Fourier Transform

**FT** Fourier Transform

**FD** Frequency Domain

**GUI** Graphical User Interface

**GPU** Graphics Processing Unit

**HD-sEMG** High Density Surface Electromyography

**HMM** Hidden Markov Model

**IC** Integrated Circuit

**IMU** Inertial Measurement Unit

**KNN** K-Nearest Neighbours

**LDA** Linear Discriminant Analysis

**LSTM** Long Short-Term Memory

**MAV** Mean Absolute Value

**MFL** Maximal Fractal Length

**ML** Machine Learning

**MLP** Multi-Layer Perceptron

**MUAP** Motor Unit Action Potential

**MU** Motor Unit

**NN** Neural Network

**PIP** Print-In-Place

**PLA** Polylactic Acid

**RF** Random Forest

**RFE** Recursive Feature Elimination

**RNN** Recurrent Neural Network

**RMS** Root Mean Square

**sEMG** Surface Electromyography

**SampEn** Sample Entropy

**SNR** Signal to Noise Ratio

**SVM** Support Vector Machine

**TPU** Thermoplastic Polyurethane

**Var** Variance

**WT** Wavelet Transform

# List of Figures


1.1 The six hand and wrist gestures targeted for classification. . . . . 2

2.1 Example of High Density Surface Electromyography (HD-sEMG), figure from [14]. 8

2.2 Comparison of achievable level of detail between Electromyography (EMG) methods, figure from [12]. . . . . 9

2.3 sEMG armband used to control surgical camera, Figure from [18] . . . . . 10

2.4 sEMG armband used to control (A) flying drones, (B), (C), and (D) robot movements, Figure from [18] . . . . . 10

2.5 Illustration of sample entropy and approximate entropy, figure from. [28] . . . . . 12

2.6 Performance of various classifiers, figure from [33]. ICA: Independent component analysis [35]. CSP: Common spatial patterns [36]. Features defined in Appendix A. 13

2.7 MYO Thalmic armband worn on forearm of a user. . . . . 16

2.8 MYO armband sensor tear down, figure from [48]. . . . . 16

2.9 The Meta Orion hardware. . . . . 17

2.10 Cyton biosensing board, figure from [3]. . . . . 20

2.11 Screenshot of OpenBCI GUI while sample sEMG data is loaded. Top left: time domain plotting, top right: frequency domain plotting, bottom left: Inertial Measurement Unit (IMU) measurements. . . . . 21

4.1 Final 3D-printed system with Cyton board. . . . . 28

4.2 Fusion 360 render of final Thermoplastic Polyurethane (TPU) sleeve design. . . . . 29

4.3 Fusion 360 render of a sensor module with wiring route highlighted. . . . . 29

4.4 Assembled sensor module with pin-header for Cyton board connectivity. . . . . 30

4.5 Hardware integration of the bias/reference electrode and Cyton board. . . . . 30

4.6 Bipolar sensor configuration. . . . . 31

4.7 Hand movements to classify. . . . . 32

4.8 OpenBCI GUI (left) and Tkinter prompting GUI (right). . . . . 34

4.9 Real-Time classification application screen grab while user is performing wrist flexion (down). At the top is the current predicted output and at the bottom of the interface is the real-time filtered signals from the 8 channels. . . . . 35

4.10 Fusion 360 visualisation of the modular chain link armband design. . . . . 36

4.11 Fusion 360 render of Polylactic Acid (PLA) Print-In-Place (PIP) chain link armband with tolerance fit. . . . . 37

4.12 Fusion 360 render of TPU armband with sensor module attached. . . . . 37

4.13 Comparison of raw sEMG signals under different fitting conditions. Each sub-figure displays the sEMG waveform (left) and its corresponding Fast Fourier Transform (FFT) transform (right) for all 8 channels. . . . . 39

4.14 Example 8 channel waveforms and frequency spectra. . . . . 42

4.15 Final feature set correlation matrix, higher correlation in red and lower correlation in blue. . . . . 45

5.1 High level data flow diagram. . . . . 52

5.2 Block diagram showing data flow from acquisition hardware to raw sEMG dataframe. 52

5.3 State machine diagram of Tkinter data collection prompting app. . . . . 53

5.4 Block diagram showing logical flow of signal processing and feature extraction. . . 54

5.5 Real-Time classification application high level design flowchart. . . . . 56

6.1 Plot of single sEMG channel showing effects of filtering. . . . . 62

7.1 Different perspectives of the sensor module placement on the forearm. . . . . 66

7.2 Training curves of final MLP architecture. . . . . 70

7.3 Confusion matrix of final MLP model on test data. Each row represents the true class and each column represents the predicted class, with diagonal values indicating correct classifications. . . . . 70

# 1 Introduction

## Contents

1.1 Key Questions and Significance . . . . . 2
1.2 Outline of Report Structure . . . . . 3

sEMG is a non-invasive technique used to measure the electrical activity produced by skeletal muscles [1]. By placing electrodes on the skin surface, sEMG captures the muscle activation patterns associated with different movements, making it a powerful tool for applications in rehabilitation, prosthetic control, and human-computer interaction.

This project focuses on the development of a low-cost, 3D-printable armband equipped with modular passive bipolar sEMG electrodes designed to work seamlessly with the OpenBCI Cyton biosensing board [2]. The proposed design improves on prior wearable sEMG systems by offering better ease of fabrication, adaptability and sensor modularity. Alongside the hardware, the project also focusses on the development of software systems capable of classifying muscle activity using the acquired sEMG signals. Both offline and real-time classification pipelines are implemented, enabling responsive gesture recognition and movement analysis.

Current commercial sEMG signal acquisition systems are often expensive and proprietary. These solutions tend to use fixed-position electrodes with limited open-source integration. OpenBCI provides powerful open-source biosensing solutions [3], but there is a lack of well-designed wearable hardware accessories that integrate seamlessly with it. The low-cost and accessible nature of this project means that users can easily 3D print the sleeve and use off-the-shelf electrodes to reproduce this setup.

1

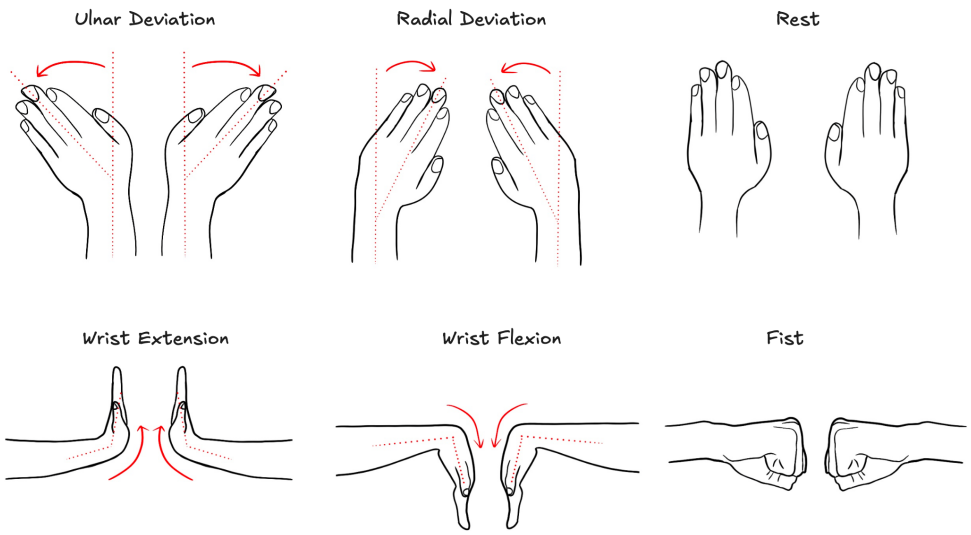



Figure 1.1: The six hand and wrist gestures targeted for classification.

## 1.1 Key Questions and Significance

This project seeks to answer several key questions at the intersection of accessible hardware design and applied machine learning:

1. Can a low-cost, 3D-printable, and modular sEMG armband be designed to be comfortable, easy to assemble, and capable of acquiring high-quality signals with the OpenBCI Cyton board [2]?

2. Can forearm muscle activity, captured using this custom hardware, be reliably classified both offline and in real-time to distinguish between a set of functional hand and wrist gestures?

3. Does a modular, adjustable sensor layout provide tangible benefits in adapting the system to different users and improving signal consistency?

These questions are significant in a wider context. In the field of Human-Computer interaction sEMG is being increasingly explored to provide gesture based control of devices, particularly in virtual / augmented reality [4], assistive devices [5], and prosthetics. A prominent recent example is the Meta Orion project which utilises an sEMG wrist band as its main input device [4]. Furthermore, a low-cost, easily reproducible sEMG system makes sEMG signal acquisition more open to students, researchers, and developers who may have access to OpenBCI hardware, encouraging the exploration of ML and wearable technologies.

## 1.2 Outline of Report Structure

This report documents the entire project lifecycle, from background research to final evaluation. The chapters are structured as follows:

- **Chapter 2: Background** provides a review of sEMG principles, existing classification techniques, and an analysis of related commercial and academic systems.
- **Chapter 3: Requirements Capture** outlines the functional and non-functional requirements that guided the system's design.
- **Chapter 4: Analysis and Design** details the iterative design process for both the hardware and software components, justifying the key engineering decisions made.
- **Chapter 5: Implementation Details** describes the interesting technical implementation of the final hardware and software, with illustrative diagrams and code segments.
- **Chapter 6: Test Plan and Verification** presents the test plan used to ensure the functional correctness and reliability of each system component.
- **Chapter 7: Results** contains a comprehensive evaluation of the system's performance, including hardware signal quality and ML classification metrics.
- **Chapter 8: Conclusions** provides a critical evaluation of the project's outcomes, discusses its successes and limitations, and suggests directions for future work.
- **Chapter 9: User Guide** serves as a practical manual for replicating and using the developed system.

1

# 2

# Background

## Contents


**2.1 Overview of sEMG signals . . . . . 6**

2.1.1 What is sEMG . . . . . 6

2.1.2 Signal Characteristics . . . . . 6

2.1.3 Key Challenges with using sEMG . . . . . 7

2.1.4 Primary Applications of sEMG . . . . . 8

2.1.5 sEMG Machine Learning Pipeline . . . . . 10

**2.2 Review of Existing sEMG systems . . . . . 15**

2.2.1 Commercial Wearables . . . . . 15

2.2.2 Research Projects & Platforms . . . . . 16

2.2.3 The OpenBCI Platform . . . . . 18

**2.3 Case for a Modular sEMG Wearable Add-On for the Cyton Board . . 19**

**2.4 Hardware and Software Utilised in this Project . . . . . 19**

2.4.1 Hardware . . . . . 19

2.4.2 Software . . . . . 21

**2.5 Chapter Summary . . . . . 21**


This chapter provides the necessary context for the project. It begins by explaining the fundamentals of sEMG signals and the standard ML pipeline used for gesture recognition. It then reviews the landscape of existing sEMG systems to identify the key limitations that this project addresses, making the case for a low-cost, modular hardware add-on for the OpenBCI platform [3]. Finally, it introduces the tools used throughout the project.

2

## 2.1 Overview of sEMG signals

### 2.1.1 What is sEMG

EMG involves measuring the potential difference between two electrodes on a skeletal muscle to record its electrical activity [6]. Processing these signals involves using a device to sense the electrical potentials generated by muscles, typically through one of two methods: needle EMG or sEMG. sEMG is non-invasive and typically non medical procedure and involves placing electrodes on the skin surface to take readings [6]. This technique is commonly used by professionals such as physiotherapists, kinesiologists, and biomedical engineers. Processing these signals can give insights into the activation strength of the measured muscle and the biomechanics of human movement. Needle EMG involves using needle electrodes inserted into muscle tissue [7]. This technique is commonly used by neurologists in medical settings to diagnose neuromuscular disorders.

When you contract a muscle, an electrical impulse starts in your brain and then travels down the central and peripheral nerves and ends at a Motor Unit (MU), which refers to a motor neuron and the muscle fibres it innervates. The electric potential is called a Motor Unit Action Potential (MUAP). As the motor neuron fires, it triggers the muscle fibres within its motor unit to contract, producing a burst of electrical activity. When many motor units are activated together, especially during voluntary contractions, these individual MUAPs overlap in time and space, creating a complex, composite signal. This signal can be detected and measured using EMG techniques. In the case of sEMG, electrodes placed on the skin surface detect the superimposed activity from multiple nearby motor units. This makes the resulting signal rich in information but also more challenging to interpret due to crosstalk and the influence of factors like electrode placement, muscle depth, and tissue conductivity.

Unlike some other biomedical signals that can be detected using surface electrodes, such as Electrocardiogram (ECG) [8], visual inspection of sEMG signals can only reveal the timing of muscle activation, indicating when a muscle is active or inactive [6]. Interesting information is obtained through advanced processing methods such as ML.

### 2.1.2 Signal Characteristics

The amplitude of sEMG signals typically ranges from a few tens of $\mu$V to 1 mV - 2 mV, depending on the specific muscle, electrode placement, and recording setup [6]. In our case, as will be discussed

in later sections, the recorded signals generally fell within a range of 0 to 600 $\mu$V, which is consistent with expectations for low-force, surface-level muscle activity.

Most of the power in an EMG signal is concentrated below 400–500 Hz [9], with the most informative components typically found in the 50–150 Hz range. This frequency band is often used to distinguish active muscle contractions from background noise and baseline activity.

### 2.1.3 Key Challenges with using sEMG

The quality of an sEMG signal is characterised by three main types of contaminants: instrumentation noise from the equipment itself, interference from external electrical sources, and measurement artifacts from the electrode-skin interface [10]. The following sources are especially relevant:

- **Power-line interference:** Caused by interference between the sEMG acquisition system and the power mains, this introduces sinusoidal noise at 50 Hz (Europe) or 60 Hz (North America), along with higher harmonics. *Mitigation:* High common-mode rejection, notch filtering [10].
- **Crosstalk:** sEMG activity from adjacent or nearby muscles contaminates the signal from the muscle of interest. *Mitigation:* Good electrode placement, smaller electrodes, reduced inter-electrode distance [10].
- **Measurement artifacts:** These arise from poor electrode contact due to motion, skin deformation, or impedance changes at the electrode-skin interface. *Mitigation:* Ensuring firm and consistent electrode placement, discarding contaminated segments, and applying signal filtering [10].
- **Motion artifacts:** These occur due to movement-induced disruption at the electrode-gel interface, producing low-frequency noise components (typically <20 Hz). *Mitigation:* High-pass filtering, stable electrode attachment, and minimizing cable movement [10].

Proper signal acquisition protocols, robust hardware, and targeted preprocessing techniques are essential to attenuate these noise sources and ensure high-quality sEMG recordings.

Other relevant challenges in using sEMG include the difficulty of obtaining accurately labelled data and the high degree of inter-subject variability [11]. Supervised machine learning methods, which are commonly used for gesture or movement classification, rely on precisely labelled training

data. Generating such labels often requires the use of synchronized video recordings, prompting software, or motion tracking systems to establish a reliable ground truth. Without accurate temporal alignment between the labels and the corresponding sEMG data, model training can become inconsistent and error-prone. Furthermore, inter-subject variability presents a significant obstacle to the generalizability of learned models. Differences in skin impedance, subcutaneous fat distribution, muscle morphology, and individual movement strategies lead to substantial variation in recorded signals across users. As a result, classifiers trained on one individual's data may perform poorly when applied to another without additional calibration or personalisation steps.

### 2.1.4 Primary Applications of sEMG

In clinical settings, sEMG can be used as a non-invasive technique to measure muscle activity. Intramuscular EMG is needed to diagnose neuromuscular disorders, although in patients like children where needle EMG is less feasible sEMG could be valuable for other disorders [12]. Needle EMG provides more detailed data at the motor unit level, whereas sEMG is more suited to quantify larger volumes of muscle activity data. It cannot offer the precision of needle EMG for tasks such as measuring spontaneous single-fibre movements for fibrillations, positive spikes, myotonia [13], etc. Conventional sEMG is of little / no use for clinical neurophysiology although advancements in the technology such as HD-sEMG can provide clearer motor unit level signals [12]. In HD-sEMG the electrodes contacting the skin consist of an array of small electrodes. An example of what this looks like is shown in Figure 2.1. In contrast, sEMG uses relatively large electrodes. A diagram showing the achievable level of detail between EMG acquisition methods is shown in Figure 2.2.

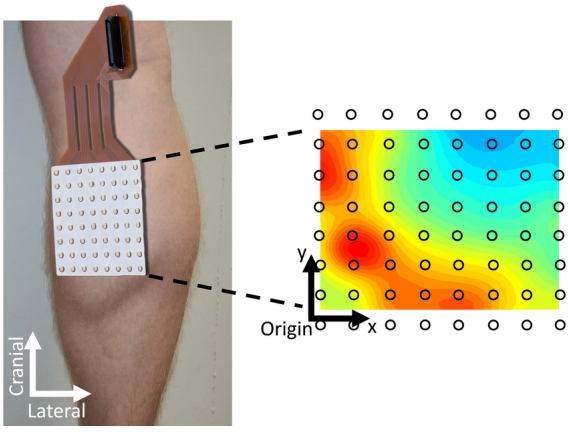



Figure 2.1: Example of HD-sEMG, figure from [14].

Physical therapists often use sEMG when studying biomechanics in the context of muscle activation during different exercises, different physical therapy treatments and rehabilitation. A

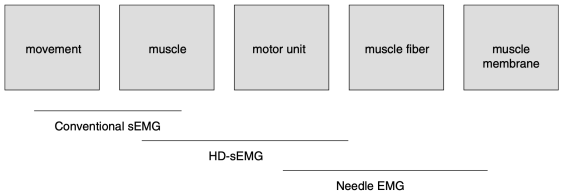



Figure 2.2: Comparison of achievable level of detail between EMG methods, figure from [12].

study comparing muscle activation between the two weight lifting exercises “Barbell Bench Press” and “Dumbbell Flys” used sEMG to compare the Root Mean Square (RMS) voltages of the signals produced by four main muscles during the two different exercises [15]. This proved to be a reliable and systematic way to compare muscle activation during the different phases of the exercises and demonstrates the usefulness of sEMG in this area.

In the past few decades, much research has occurred in the realm of myoelectric control of assistive devices with many of them producing promising results [16]. Success has been found in developing control methods based on sEMG for applications such as Electric Power Wheelchairs [5] and control of prosthetic hands [16]. However, sEMG analysis seems quite restricted in capability, especially when only using a few electrodes. The recorded signals do not provide a level of detail sufficient for highly complex control tasks, which are essential for applications such as controllable upper-limb prostheses for amputees [16]. Furthermore, pure sEMG techniques have no way of providing feedback to the user, which would be useful in these contexts [16]. That being said, sEMG is still widely used in human-computer interaction applications in commercial / clinical upper limb prosthetic control due to its non-invasiveness, while still being able to provide signals good enough for some complexity of control [17].

In healthcare and prosthetics, armband-based sEMG systems have been widely employed to control upper-limb prostheses by translating muscle activity into corresponding movements [18]. These systems have also been used during rehabilitation to assist patients in regaining motor function, showing improvements in both range of motion and dexterity [19]. In surgical settings, hands-free control of medical equipment, such as endoscopic cameras, has been enabled using gesture-based commands derived from sEMG signals, allowing surgeons to adjust views without physical contact [18]. This is shown in Figure 2.3.

Beyond medical applications, sEMG-based gesture recognition has found use in gaming and entertainment, enabling users to control gameplay through natural muscle movements. It has also been integrated into interactive stage performances, where gestures trigger audiovisual effects such as lighting cues. Additionally, sEMG control has been explored for operating external devices

Figure 2.3: sEMG armband used to control surgical camera, Figure from [18]

such as robotic arms, drones, and other consumer electronics, offering an intuitive interface for human–machine interaction. Examples of such are shown in Figure 2.4.

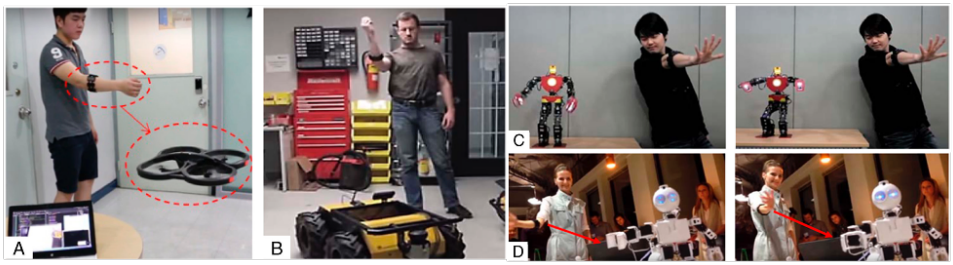



Figure 2.4: sEMG armband used to control (A) flying drones, (B), (C), and (D) robot movements, Figure from [18]

A highly advanced use of EMG signals is shown in "The Mind Controlled Prosthesis" [20] initiative lead by Cindy Chestek and Paul Cederna has pushed the capabilities of prosthetic hand control using EMG through the development of "Regenerative Peripheral Nerve Interfaces" [21]. This is a surgical procedure performed during amputation of an upper limb or after, where a piece of muscle is attached to the end of nerves towards the point of amputation and this muscle amplifies the EMG signals 10-100 times [20]. They then used indwelling EMG electrodes and two ML techniques (Kalman Filtering for continuous variable tasks and a Naive Bayes classifier for discrete). The prosthetic hand trails produced impressive results for the natural control of prosthetic hands for upper limb amputees and these surgical and ML techniques could be the way forward for improving the quality of life of such patients.

### 2.1.5 sEMG Machine Learning Pipeline

The general flow to take sEMG signals and convert these to gesture classifications is as follows. Signal extraction from the sensors, filtering and processing, feature extraction, training on data, and then finally classification using an ML model [22].

There are two fundamental design choices when developing an sEMG-based classification sys-

tem: the selection of features to extract from the raw signal, and the choice of classifier used to interpret those features. Both decisions have a critical impact on the system's overall performance, particularly in terms of classification accuracy, computational efficiency, and real-time responsiveness.

The first decision—feature extraction—involves identifying meaningful characteristics from the raw signal that capture relevant muscle activity patterns. This process must balance discriminative power with computational cost and robustness to noise [23]–[25]. The second decision relates to classifier selection. The chosen machine learning model must be well-suited to the nature of the extracted features, the number of classes, and the variability in the input data, all while maintaining low latency for real-time applications [24].

**Feature Extraction**

Generally, features can be extracted from the time domain or time frequency domain of the data [26]. The time domain features are easy to implement and require lower computation as no transformation is needed. The features from the frequency domain tend to be statistical properties of the power spectral density of an sEMG signal. Frequency Domain (FD) features involve using methods such as Fourier Transform (FT) and Wavelet Transform (WT) which are popular methods but much more computationally demanding and are therefore only suitable for high performance embedded systems [27].

The authors of [27] decided to only investigate time domain features as they worked better with Linear Discriminant Analysis (LDA) classifiers which they found to be a more robust classifier than more traditional Support Vector Machine (SVM)s for sEMG classification.

Phinyomark et al. [23] provide a comprehensive quantitative comparison of 50 different sEMG features extracted from sEMG signals when used in a variety of different classifiers. Here, they used fluctuating sEMG data taken from trials taken over 21 days to take into account the effect of 'old' training. The best performing features (highest mean test classification) were found to be Sample Entropy (SampEn), Approximate Entropy (ApEn) and Maximal Fractal Length (MFL) [23]. In this context Entropy refers to the level of complexity or regularity and is a useful non-linear feature often applied to biological data which are inherently noisy [28] [29]. A visual of both can be seen in Figure 2.5.

Many other authors, most of whom did not consider the long-term effects of pattern matching

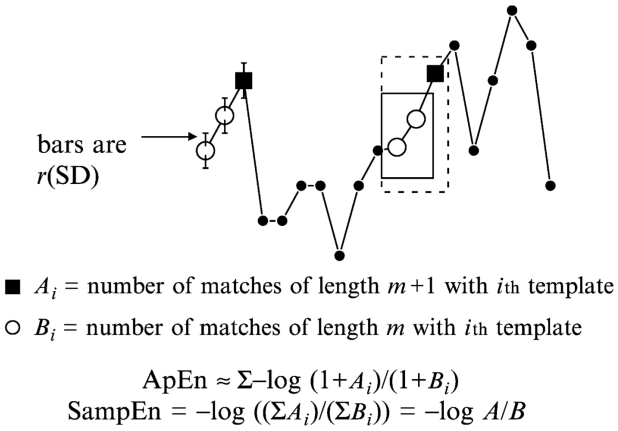



FIG. 1. Schematic demonstration of entropy estimation using approximate entropy (ApEn) and sample entropy (SampEn). The time series begins with the $i$th template. In this example, $m$ is 2. The tolerance for accepting matches is $r$ times the standard deviation, and is shown by the error bars. Here, the template is matched by the 11 and 12th points (solid box), and the $m+1$st points also match (dashed box). Thus quantities $A$ and $B$ both increment by 1.

Figure 2.5: Illustration of sample entropy and approximate entropy, figure from. [28]

biological data, used common simpler time domain features (as expected for a wearable context) and achieved high classification accuracies [26], [30], [31]. Some of the simplest features here are: RMS, Mean Absolute Value (MAV), Variance (Var), etc. Much success has been found with a wide array of features + classifier combinations and this will need to be explored for our particular use case and quality of sensors.

### Classifier Selection

The second important choice is which classifier(s) to use. There are many ML classifiers that could work for this use case and produce good results, some simple classifiers would be SVM, MLP, LDA and K-Nearest Neighbours (KNN). Some more complex classifiers would be Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and even pre-trained foundation models [32]. Trade-offs here include: the number of poses we wish to be able to tell apart, computation time (latency of the system), the number of input channels, the complexity and time taken to train [24].

Various authors have found that the use of a group of the best performing time domain characteristics along with a LDA classifier gave the best classification accuracy [23], [27], [33], [34]. Many authors also opted to use SVMs [31]. Using multiple different classifiers and then performing a post-processing majority vote step also gave very good results [33]. A comparison from [33] can be seen in Figure 2.6. Using multiple models increases computational costs. This trade-off

**Table 1.** Summary of various individual classifiers and combined classifiers tested on amputee data [1].

| Pre-Processing | Segmentation/Window Length | Feature Extraction/DR | Classification | Post-Processing | Classes/EMG Channel | Accuracy |
|---|---|---|---|---|---|---|
| N/A | 256 ms overlapping 32 ms | TD, 6AR, RMS/PCA, ULDA | KNN, LDA [69] | Majority vote | 7/57 | >97% |
| N/A | 200 ms length with 50 ms increment | 6AR, RMS, IAV, ZC, WL, SSC/OFNDA | LDA [37] | N/A | 12/11 | 90% |
| N/A | 200 ms with 5 ms increment window | MAV, ZC, WL, SSC/N/A | LDA [64] | N/A | 7/7 | 95.64% |
| ICA | 250 ms overlapping window with 64 ms increment | 4AR, RMS, MAV, ZC, VAR, WL/ULDA | LDA [12] | N/A | 12/11 | >90% |
| CSP | 300 ms with 75 ms of delay between the overlapped window | M, RMS, WA, SSC/PCA | ANN [65] | N/A | 5/6 | 92.04%(PCA) 93.4%(CSP) |
| N/A | Window set to 4500 and window shift 50 | Variogram/N/A | SVM [67] | N/A | 7/48 | 81.6% |
| N/A | 256 ms with window shift 32 ms | WL/N/A | NN [70] | N/A | 4/6 | An average RMS error=0.16 for 4 patterns |
| N/A | 200 ms sliding window | RMS, log(rms)/N/A | Fuzzy c- means clustering [71] | N/A | 4/3 | 87.5±13% |
| N/A | 200 ms with an increment of 75 ms | RMS, WL, ZC, SSC/N/A | LDA [72] | N/A | 6/8 | >91% |

[1] Table omit the results from able-bodied subjects.

Figure 2.6: Performance of various classifiers, figure from [33]. ICA: Independent component analysis [35]. CSP: Common spatial patterns [36]. Features defined in Appendix A.

needs to be evaluated based on the particular models being used and the potential improvement in classification accuracy they provide.

Deep Learning (DL) methods are a subset of ML methods that are characterised by neural networks with many layers (hence they are ”deep”) [37]. They can automatically extract features from raw data during training, reducing, or sometimes eliminating, the need for manual feature engineering. DL models use large multilayer networks with many parameters that can include convolutional layers, recurrent layers, and fully connected layers. They require large data sets and a large amount of computational resources, like Graphics Processing Unit (GPU)s for training. [38] There are also Transformer-based DL models which are highly scalable models which use attention mechanisms which can be shown to be superior in tasks with spatial or sequential dependencies (like language or time series data tasks). These models also have many parameters which means they are also computationally demanding to train and run. [39] The increased complexity of these models allows them to typically achieve higher classification accuracies than those achieved by traditional ML methods. The trade-offs and differences between the two approaches are summarised in Table 2.1.

| **Aspect** | **Deep Learning** | **Non-Deep Learning** |
|---|---|---|
| **Feature Extraction** | Features learned during training | Manually extracted |
| **Data Requirements** | Large datasets | Small datasets |
| **Model Complexity** | Up to billions of parameters | Much fewer parameters |
| **Computational Power** | High (requires GPUs) | Low (CPU is often sufficient) |
| **Flexibility** | Can handle raw, unprocessed data | Needs preprocessed data |

Table 2.1: Comparison of DL and non-DL methods.

Below is a non-exhaustive list of common types of classifier used in sEMG classification.

- **Multi-Layer Perceptron (MLP) [40]**
  - The simplest type of neural network that consists of multiple layers of neurons with non-linear activation functions.
  - MLPs are good at capturing complex relationships in data and can perform very pattern recognition tasks compared to traditional ML models.
  - No longer commonly used as more advanced Neural Network (NN) models such as CNNs and RNNs can achieve superior performance in handling high dimensional data.
- **Convolutional Neural Network (CNN) [41]**
  - CNNs are particularly effective for data with spatial dependencies. Although used mainly for visual data, they can also be applied to time series EMG data.
  - They use convolutional layers that automatically learn and detect important features, making them highly effective for tasks requiring feature extraction and classification.
- **Self-Organising Maps (Fuzzy) [42]**
  - A type of unsupervised learning that results in a map of input features organised in such a way that similar features are clustered together on the map.
  - Fuzzy variants incorporate concepts from fuzzy logic, providing a way to handle uncertainty and imprecision in EMG data classification.
- **Naive Bayes Classifier [43]**
  - This model applies the Bayes theorem, assuming independence between features and that each feature contributes equally to an input belonging to a class. The model chooses the class to which the input most likely belongs to given the training data.
  - It is simple and effective, especially when the assumption of independence holds reasonably true. This assumption tends to be unrealistic in many scenarios.
- **Support Vector Machine (SVM) [44]**
  - SVMs are linear classifiers that work by finding the hyperplane that best separates the data into classes.
  - They are effective in high-dimensional spaces and are robust against over-fitting, especially in cases where the number of dimensions exceeds the number of samples (this could be the case here).

  - SVMs are very commonly found in sEMG classification literature.

- **Linear Discriminant Analysis (LDA) [44]**
  - LDA works by projecting the data onto a lower-dimensional space where the classes are as distinct as possible.
  - LDA is also very commonly used for this task, with many authors claiming that it is the best method.

- **Hidden Markov Model (Hidden Markov Model (HMM)) [45]**
  - HMMs model the sequence and time series data as a Markov process with hidden states.
  - They are suitable for gesture recognition from sequential EMG signals where the signal time properties are important.

- **K-Nearest Neighbour (KNN) [44]**
  - KNN operates by finding the most similar recorded instances (neighbours) in the training data.
  - It is a very simple model that does not need to be trained, although it has a high space complexity because it needs to store all training examples.
  - KNN may not be feasible in a real time context due to its high memory requirements to compare to all training examples, particularly if the data set is large.

## 2.2 Review of Existing sEMG systems

### 2.2.1 Commercial Wearables

#### MYO Armband

The MYO Thalmic Armband is an sEMG sensing device that wraps around the user's forearm [18]. It is composed of eight fixed position sections, each of which has three electrodes and read-out electronics. It also contains a 9-axis IMU to measure spatial orientation. The band is elastic so that it fits snug on the user's forearm. The device's main purpose is to act as an alternative input device to computers to allow for gesture control. Examples of usage include controlling presentation slides, interacting with smart home devices and video game control for a more immersive experience (although a small number of games support it). An image of the unit can be seen in Figure 2.7.

The device allows for five basic gestures: wave right, wave left, fist, open hand, and double tap index and thumb [46]. In an experiment that compared the experience of playing "Need for Speed", a famous racing video game, more people reported that playing with the MYO armband was more exciting compared to playing with a keyboard [46].

Figure 2.7: MYO Thalmic armband worn on forearm of a user.

Each of the eight sEMG sensing modules incorporates a bipolar sEMG electrode arrangement. In a bipolar configuration, two electrodes are placed on the muscle and a reference electrode is placed in an area with little activity [47]. The voltage difference between the two electrodes on the muscle is measured [18]. This setup will capture the differential signal, which should reduce noise as it cancels common-mode noise. The bipolar sensors are shown in Figure 2.8. The signals are then amplified using amplifier circuits in each sensor module. The armband samples data at a frequency of 200 Hz. The price of this armband is $149.

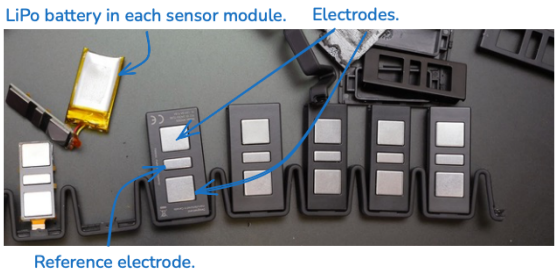



Figure 2.8: MYO armband sensor tear down, figure from [48].

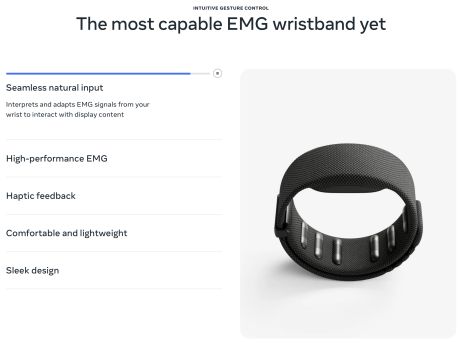
### 2.2.2 Research Projects & Platforms

#### sEMG Bracelet in Meta Orion Project

The Meta Orion project is a prototype Augmented Reality (AR) glasses project, that is not available to consumers [4], see Figure 2.9a. It cost millions in research and development and the unit cost is around $10,000. The main input device for AR glasses is their sEMG wristband (Figure 2.9b)

which had its first research prototype produced in 2020 and the prototype product produced in 2024 [49]. Meta has not released technical specifications for the sEMG electrodes, inspection shows that there are six modules. The electrodes appear to be in a bipolar configuration similar to the MYO sEMG sensing module discussed above, but much smaller.


(a) Meta Orion components. Image from [4].




(b) Meta Orion sEMG bracelet. Image from [4].

Figure 2.9: The Meta Orion hardware.

The claimed capabilities are very impressive. It allows for a detailed estimation of pose, at the individual finger level. It also allows for typing on a virtual keyboard completely from finger pose estimation. Meta have open-sourced the large datasets for both tasks, called ”emg2pose” and ”emg2qwerty”, respectively [50]. The data set consists of 301 individuals and 716 hours of recorded sEMG data.

### The ”3DC” Armband

The **3DC Armband** is a low-cost, wireless, 3D-printed sEMG acquisition system developed by the Biomedical Microsystems Laboratory at Laval University (BML-UL) [51]. The project's primary goal was to create an affordable device for personal use that does not sacrifice signal quality or wearability, addressing the high cost of many existing sEMG systems. The estimated cost to assemble the armband is approximately $150 USD [51]. However, it is important to note that this price is equal to the retail price of the MYO armband, which was a complete, consumer-grade product. This equivalence challenges the ”low-cost” claim of the 3DC system, suggesting its primary advantages lie in its open-source design and superior signal quality rather than affordability when compared to previous commercial offerings.

The 3DC Armband features **10 sEMG fixed position channels** that record at a sampling rate of 1000 Hz using dry electrodes. The device is lightweight at only 62g and also includes a 9-axis IMU for tracking orientation [51].

To validate its performance, the creators conducted a direct comparison with the MYO armband. A new dataset was recorded for this purpose, featuring 22 participants performing 11 distinct hand and wrist gestures while wearing both armbands simultaneously. The results showed that the 3DC Armband systematically and significantly outperformed the MYO Armband across three different ML classifiers when sufficient training data was used [51].

A significant contribution of this project is its open-source nature. The complete dataset, along with the source code, Altium project files for the PCB, and the 3D models for the armband, were made publicly available in a GitHub repository to encourage further research [51].

### 2.2.3 The OpenBCI Platform

OpenBCI (Open-Source Brain-Computer Interface) is a versatile research platform and company dedicated to democratizing neurotechnology by providing low-cost, open-source hardware and software for biosensing [3]. The ecosystem is designed to be highly flexible, enabling researchers, developers, and hobbyists to sample various biopotential signals, including Electroencephalogram (EEG), ECG, and EMG.

The core of the hardware offering is a series of Arduino-compatible data acquisition boards, most notably the **Cyton** [2] and **Ganglion** boards [52]. The Cyton board, built around the high-quality Texas Instruments ADS1299 Analogue-to-Digital Converter (ADC), provides 8 high-resolution (24-bit) channels sampling at 250 Hz. This can be expanded to 16 channels using a ”Daisy” extension module, which halves the sampling rate to 125 Hz. The platform is compatible with a wide range of standard electrodes and includes an onboard accelerometer for motion tracking.

A key strength of the OpenBCI platform is its commitment to open-source principles. All hardware schematics, firmware, and software are publicly available. This allows for extensive customisation and integration with other popular research tools like Python and MATLAB. Furthermore, the community shares designs for peripherals, such as the 3D-printable Ultracortex headset [53], making it a highly adaptable and cost-effective solution for prototyping and conducting research in sEMG and other Brain Computer Interface (BCI) applications.

## 2.3 Case for a Modular sEMG Wearable Add-On for the Cyton Board

While the OpenBCI platform provides powerful and accessible data acquisition capabilities through the Cyton board, it currently lacks a dedicated, integrated wearable for forearm sEMG data acquisition. The platform's ecosystem would be significantly enhanced by a reusable and convenient hardware solution specifically for this application.

At present, users wishing to conduct forearm sEMG research using the OpenBCI platform are directed to use disposable, gel-based snap electrodes sold on the openBCI shop [3]. While these provide great signal quality [47], this approach presents several practical limitations. These electrodes are single-use, leading to recurring costs and environmental waste. Furthermore, the process of manually placing numerous individual electrodes is time-consuming and prone to inconsistency, especially when utilizing all eight available channels. Inconsistent electrode placement is a critical issue in sEMG research, as it can significantly affect signal quality and the repeatability of experiments.

The development of a modular, wearable sEMG armband designed as an add-on for the Cyton board would directly address these challenges. Such a device would provide a reusable, convenient, and standardised method for acquiring high-density forearm muscle activity. This solution would greatly benefit students, researchers, and developers within the OpenBCI community who wish to explore ML applications with sEMG data, such as real-time gesture recognition.

## 2.4 Hardware and Software Utilised in this Project

The development of this project's system was made possible by leveraging a combination of specialised off-the-shelf components, materials, and software platforms.

### 2.4.1 Hardware

The physical system combines the core data acquisition electronics with custom-fabricated parts and readily available consumables.

- **OpenBCI Cyton Board:** As introduced previously, the Cyton board an Arduino compatible 8-channel neural interface with a 32-bit processor and can be used to sample EEG, EMG, and ECG activity. The board is shown in Figure 2.10.

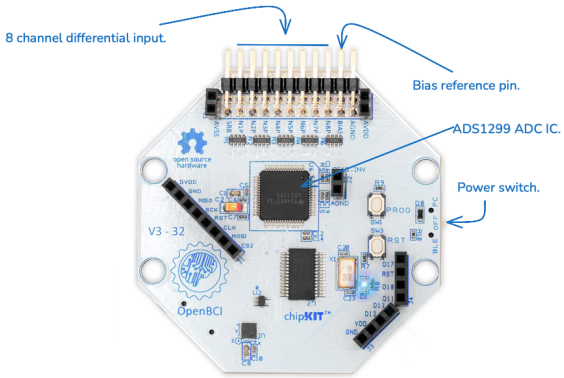



Figure 2.10: Cyton biosensing board, figure from [3].

- **RFDuino Dongle:** The wireless communication between the Cyton board and the host computer was handled by the RFDuino USB dongle, which is supplied with the Cyton board and ensures reliable data transmission. Together with the Cyton board this offers 8-channel data at a sampling frequency of 250 Hz to be transmitted. This may be a limitation as typically the highest frequencies in an EMG signal are between 400-500 Hz. Therefore, in line with the Nyquist-Shannon sampling theorem [54], there would be a benefit to increasing the sampling frequency up to 1 kHz (2*500 Hz) to ensure the entire signal spectrum is captured [55]. As discussed above, the MYO armband only samples data at 200 Hz and achieves gesture control. The ”3DC” system, also discussed above, samples at 1000 Hz and claims to achieve better results than that of the MYO armband [51]. This increased sampling rate may be a factor in the better performance observed in the ”3DC” armband, as they are able to take advantage of the higher frequency information present in sEMG.

- **SKINTACT ECG Electrodes:** Standard, commercially available SKINTACT F-301 Ag/AgCl hydrogel electrodes [56] were used for signal acquisition. The gel was removed when the electrodes were used, so they were used as dry electrodes.

- **3D Printer and Filaments:** A Bambu Lab P1S 3D printer [57] was used for all custom component fabrication. Two specific materials were chosen for their distinct properties:

  - **PLA:** Used for the rigid components due to its stiffness and ease of printing.
  - **TPU:** Used for the flexible armbands and straps.

- **Assembly Components:** Standard male-to-female jumper wires were used to connect the electrodes to the Cyton board's input pins. For fastening, heavy-duty Velcro was used in the final design to create an adjustable mechanism. Neodymium magnets were also utilised in earlier prototypes before the design was iterated upon.

### 2.4.2 Software

The software toolchain includes established platforms for design and data acquisition, which were used alongside the custom Python applications developed as part of this work.

- **OpenBCI GUI:** The official OpenBCI GUI was used during the data collection phase for two primary purposes: real-time visualisation to ensure signal integrity during setup, and recording the raw, timestamped sEMG data into `.txt` files. This GUI is shown in 2.11.

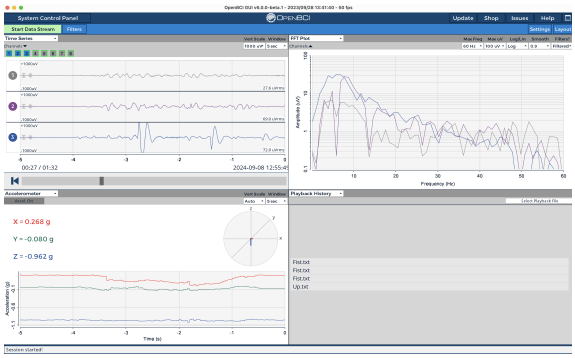


Figure 2.11: Screenshot of OpenBCI GUI while sample sEMG data is loaded. Top left: time domain plotting, top right: frequency domain plotting, bottom left: IMU measurements.

- **Autodesk Fusion 360:** All 3D models for the wearable components were designed in Autodesk Fusion 360 [58]. This professional-grade Computer Aided Design (CAD) software enabled the precise parametric modelling required to create the modular sensor housings, the flexible armband, and other custom parts detailed in Chapter 4.

## 2.5 Chapter Summary

This chapter has provided a comprehensive background on the principles and applications of sEMG-based gesture recognition. It began by detailing the physiological origins and characteristics of sEMG signals, along with the inherent challenges in acquiring high-quality data. The standard

ML pipeline for classifying these signals, including the stages of preprocessing, feature extraction, and classifier selection, was reviewed to establish the technical context for this project.

A survey of existing systems, from the commercial MYO armband to the Meta Orion project and the academic ”3DC” armband, revealed a landscape of powerful but often inaccessible, inflexible, or proprietary solutions. This analysis identified a specific ”peripheral gap” within the popular, open-source OpenBCI ecosystem, which lacks a dedicated, reusable, and adaptable forearm wearable. This project is positioned to fill that gap by creating a low-cost hardware add-on for the Cyton board. Finally, the enabling technologies, particularly 3D printing, that make such a custom and accessible solution feasible were introduced. This foundational work provides the clear rationale for the system requirements and design choices detailed in the subsequent chapters.

# 3

# Requirements Capture

Contents

3.1 Functional Requirements . . . . . 23
3.2 Non-Functional Requirements . . . . . 24
3.3 Chapter Summary . . . . . 25

This chapter outlines the functional and non-functional requirements that guided the design and implementation of the hardware and software developed in this project. These requirements were established based on the analysis of existing wearable sEMG systems and the specific opportunities identified in the background research detailed in Chapter 2. They serve as the core objectives and constraints against which the final system is evaluated.

## 3.1 Functional Requirements

Functional requirements specify the core operations that the system must perform to achieve its intended goals. For this project, the system was required to:

1. **Acquire Multi-Channel sEMG Signals:** The system must be capable of capturing raw sEMG signals from up to eight bipolar electrode channels using the OpenBCI Cyton biosensing board and transmitting them wirelessly to a host computer.

2. **Design and Fabricate a Modular Wearable Armband:** A 3D-printable forearm armband must be designed and fabricated. This hardware must:

(a) Maintain secure and consistent skin contact for all electrodes during a range of forearm and wrist movements.

(b) Support the modular attachment and repositioning of sensor modules to accommodate different forearm sizes and to allow for optimised placement over target muscle groups.

3. **Develop a Structured Data Collection Protocol:** A systematic protocol, supported by a GUI, must be developed to guide participants through data collection sessions. This system must generate accurately timestamped labels corresponding to the raw sEMG recordings.

4. **Pre-process sEMG Signals:** The system must implement digital signal processing pipelines for both offline and real-time use cases. This includes:

   (a) Band-pass filtering to remove baseline drift and noise.

   (b) Notch filtering to suppress 50 Hz power-line interference (UK).

   (c) Windowing the continuous signal into overlapping segments for feature computation.

5. **Extract and Select Relevant Features:** An effective set of features must be computed from each signal window to capture the discriminative characteristics of muscle activity. The feature selection process should aim to maximise class separability while maintaining computational efficiency.

6. **Classify Forearm Movements Using Machine Learning:** The system must be able to train and evaluate machine learning models capable of recognising a set of distinct hand and wrist gestures. This capability must be demonstrated in two modes:

   (a) *Offline mode:* Train and test models using pre-recorded data to establish a baseline for maximum classification accuracy.

   (b) *Real-time mode:* Continuously acquire, process, and classify incoming sEMG data with minimal latency to enable responsive, interactive control.

## 3.2 Non-Functional Requirements

Non-functional requirements address the broader qualities and constraints of the system, such as its usability, reliability, and cost-effectiveness. The following quality goals were established for this project:

1. **Low Cost of Add-on Components:** While the OpenBCI Cyton board is a significant investment, all custom-designed hardware components, including the 3D-printable parts, sensors, and wiring, must be low-cost and use commercially available materials to ensure the armband is an accessible and easily replicable add-on.

2. **High Signal to Noise Ratio (SNR):** The physical design of the armband and sensor modules must aim to maximise the SNR of the acquired signals. This is to be achieved through stable, high-quality electrode contact and customisable sensor placement that minimises motion artefacts and crosstalk.

3. **Modularity and Scalability:** The system must be inherently modular. The hardware design should support reconfigurable electrode layouts, and function correctly with up to the eight differential inputs provided by the Cyton board.

4. **Comfort and Wearability:** The armband must be lightweight, flexible, and adjustable to ensure user comfort during extended use, preventing skin irritation or excessive pressure.

5. **Low Latency:** The real-time classification pipeline must be computationally efficient. The total time from signal acquisition to a prediction being displayed must be low enough to facilitate smooth and responsive user interaction.

6. **Robustness to User Variability:** While a universal, user-independent model is beyond the scope of this project, the system must be designed to handle data from a diverse set of users. This is to be addressed by collecting a dataset from participants with varied physical characteristics and ensuring the machine learning models do not over-fit to specific individuals.

## 3.3 Chapter Summary

This chapter has defined the key functional and non-functional requirements that serve as the blueprint for the project. The functional requirements outline the necessary capabilities of the hardware and software, from signal acquisition to real-time classification. The non-functional requirements establish the critical quality attributes, including the low cost of the custom hardware, signal quality, modularity, and low-latency performance. Together, these requirements form the criteria against which the final system's design and performance, as detailed in the subsequent chapters, are measured.

3

# 4 Analysis and Design

## Contents


**4.1 Final Design . . . . . 28**
4.1.1 Final Hardware Design . . . . . 28
4.1.2 Final Software Design . . . . . 31
4.1.3 Summary of Python Applications Developed . . . . . 33
**4.2 Design Constraints . . . . . 34**
**4.3 Hardware Design Decisions . . . . . 35**
4.3.1 Armband Design Decisions . . . . . 35
4.3.2 Modular Sensor Design and Integration . . . . . 38
4.3.3 Bias Placement . . . . . 38
4.3.4 Characterisation of the Sensor Modules . . . . . 39
4.3.5 Cost of Materials . . . . . 39
**4.4 Software Design Decisions . . . . . 40**
4.4.1 Data Collection Protocol and GUI . . . . . 40
4.4.2 Adding Labels to Raw sEMG Data . . . . . 41
4.4.3 Signal Preprocessing . . . . . 41
4.4.4 Data Augmentation . . . . . 44
4.4.5 Feature Selection Process . . . . . 44
4.4.6 Machine Learning Model Evaluation . . . . . 46
4.4.7 Real-Time Classification Application . . . . . 48
**4.5 Chapter Summary . . . . . 50**


This chapter presents a comprehensive overview of the system's final architecture, alongside the engineering decisions and iterative design processes that shaped its development. It begins with a summary of the complete system, both hardware and software, followed by an outline of the objectives and constraints that guided the project. Subsequent sections detail the progression of hardware design, including the development of the modular sensor modules and 3D-printed wearable sleeve, as well as the design trade-offs and rationale behind each iteration. On the software side, the data acquisition pipeline, preprocessing techniques, feature selection, and ML

model design are thoroughly discussed. Each stage is critically evaluated in the context of real-time applicability and classification accuracy, culminating in the integration of a responsive, low-latency classification application. The chapter serves to justify and contextualise the system's final design in alignment with the project's goals.



## 4.1 Final Design

We begin this chapter presenting the final implementation before moving to the design process.

### 4.1.1 Final Hardware Design

The final 3D-Printed sEMG acquisition system is shown being worn in Figure 4.1.

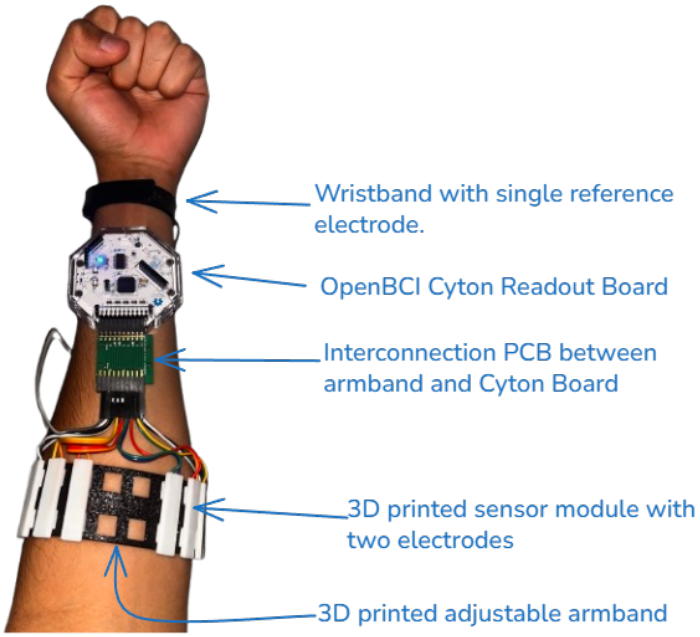



Figure 4.1: Final 3D-printed system with Cyton board.

Around the forearm, there is a TPU (a flexible 3D-printable plastic) armband with square cutouts arranged around the length of the armband to which the sensor modules can attach. A render of the TPU armband is shown in 4.2.

Attached to the armband are eight modular PLA (a rigid 3D-printable plastic) bipolar sensors that can be positioned any where around the armband, all eight sensors do not need to be attached,

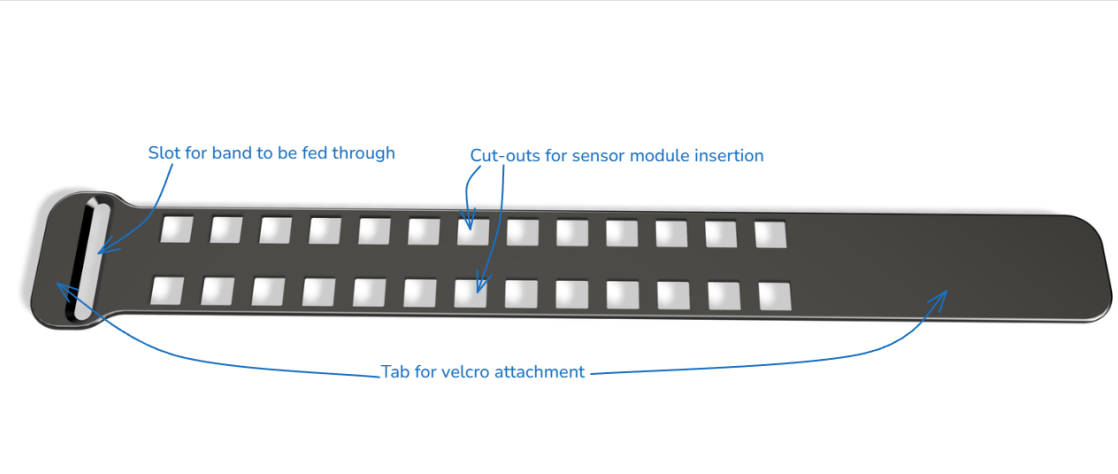



Figure 4.2: Fusion 360 render of final TPU sleeve design.

and the Cyton board can support up to a maximum of 8 channels. The two passive electrodes on each PLA sensor are taken from inexpensive "SKINTACT ECG Electrodes" which are made from silver with a silver chloride top layer [56]. The wiring is fed through the wire routes (highlighted in Figure 4.3) and the ends are stripped. These passive electrodes are placed on top and snap fit into the sensor module, the tolerance is intentionally very tight and this requires significant force to press in. This approach yields a highly reliable, low-resistance connection between the electrodes and wiring. Using a standard multimeter, the resistance between each electrode and the corresponding wire terminal was consistently measured at 0.1 Ω across all sensor modules. These modules attach to the TPU armband using a tolerance snap fit from the square cut out on the armband and the two tapered notches on the sensor. A render of the sensor module is shown in Figure 4.3. An image of an assembled sensor module with electrodes and wires is shown in 4.4.

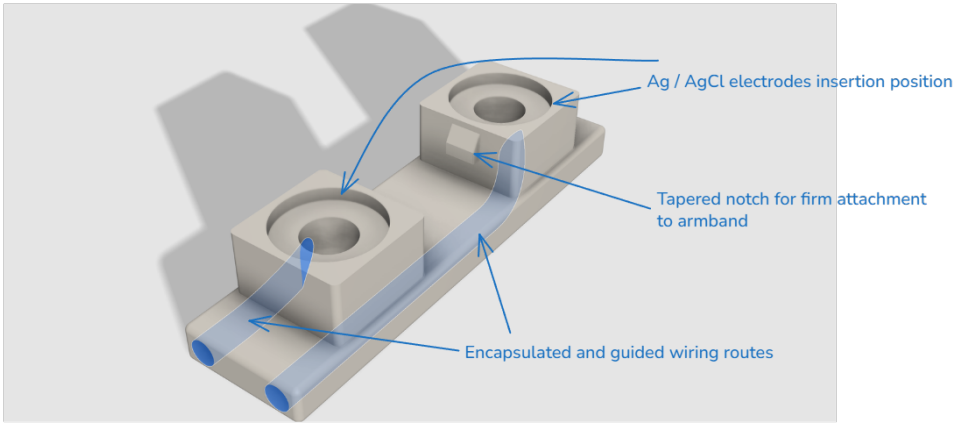


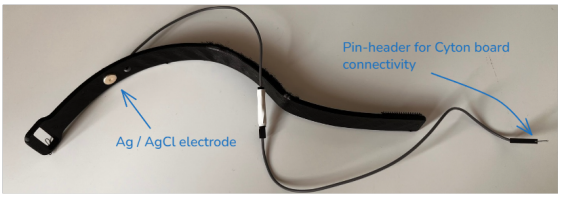

Figure 4.3: Fusion 360 render of a sensor module with wiring route highlighted.

The two wires coming from the sensors connect to the corresponding channel pin on the Cyton

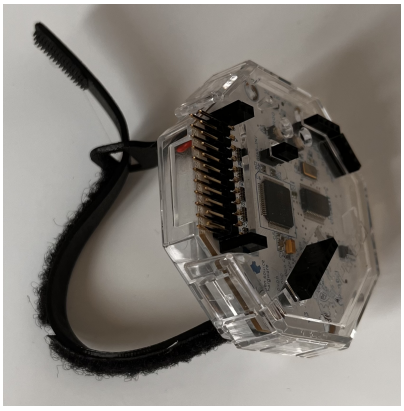
Figure 4.4: Assembled sensor module with pin-header for Cyton board connectivity.



board. For practicality, a simple PCB with two 10x2 pin-out headers is used for ease of connection to the board, but this is not necessary. A TPU band, Figure 4.5a is also worn around the wrist. This contains an electrode that is placed over the pisiform bone to provide the board with a reference signal with little to no muscular activity, this goes to the pin labelled 'bias' on the Cyton board. Another similar TPU band integrates with the plastic enclosure of the Cyton board, allowing it to be worn on the forearm. This is shown in Figure 4.5b.




(a) Bias band with reference electrode inserted.



(b) Cyton board with plastic enclosure and TPU band attached.

Figure 4.5: Hardware integration of the bias/reference electrode and Cyton board.

Each channel on the Cyton board measures the voltage difference between the two electrodes attached to the differential channel pins on the Cyton board:

$$V_{Channel_i} = V^{+}_{Channel_i} - V^{-}_{Channel_i} \tag{4.1}$$

Measuring differential signals reduces common-mode noise and should improve the SNR. The bias pin of the ADS1299 [59] Integrated Circuit (IC) used in the board is also used for noise cancelling. It establishes a common ground between the Cyton board and your body, and also injects a compensating signal back into the body to further cancel the noise. This setup is illustrated in Figure 4.6

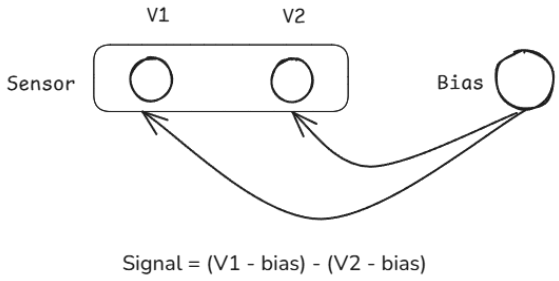



Figure 4.6: Bipolar sensor configuration.



### 4.1.2 Final Software Design

An objective of this project was to develop a system capable of classifying sEMG signals into six distinct hand and wrist movement classes. This was formulated as a multi-class classification problem, where the goal is to assign short time windows of sEMG data to one of the predefined movement classes.

Let $x \in \mathbb{R}^d$ denote a feature vector extracted from a window of sEMG data. The length of this vector, d, will be determined at the feature selection phase and d will be equal to the number of features extracted from each window. Let $y \in \{1, 2, 3, 4, 5, 6\}$ denote the corresponding class label, representing: Rest (1), Fist (2), Wrist Flexion (3), Wrist Extension (4), Radial Deviation (5), and Ulnar Deviation (6).

The objective is to learn a function

$$f : \mathbb{R}^d \to \{1, 2, 3, 4, 5, 6\}$$

that maps an input feature vector $x$ to its correct class label $y$. This function is trained using a labelled dataset $\{(x_i, y_i)\}_{i=1}^{N}$, where $N$ is the number of training samples.

The classifier $f$ is optimised to minimise a loss function, typically the categorical cross-entropy:

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{k=1}^{6} \mathbb{1}(y_i = k) \log \hat{p}_{i,k}$$

where $\hat{p}_{i,k}$ is the predicted probability of class $k$ for the $i$-th input, and $\mathbb{1}(\cdot)$ is the indicator function.



Raw sEMG data was acquired using the 3D-printed armband connected to an OpenBCI Cyton board. Data was transmitted wirelessly via the RFduino dongle and recorded using a Python-based application developed with `tkinter` and the OpenBCI GUI, which provided participants with visual cues and movement timing.

Data was collected from 10 participants using both left and right arms. Each participant performed the six different movements (*Rest*, *Fist*, *Flexion*, *Extension*, *Radial Deviation*, and *Ulnar Deviation*), with five repetitions per movement, each lasting four seconds. These hand movements are shown in Figure 4.7. Participants included both males and females across a range of ages and levels of physical activity to ensure dataset diversity and robustness.

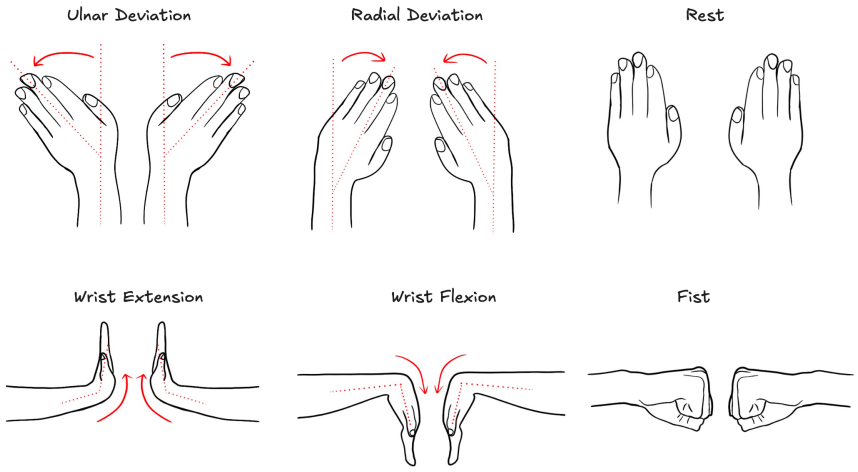



Figure 4.7: Hand movements to classify.

The collected data underwent preprocessing steps involving filtering and normalisation. Filtering was used to remove motion artifacts and powerline interference, while normalisation mitigated inter-subject amplitude differences. The best filtering configuration was found to be a band pass filter with a low-cut at 30 Hz and 124 Hz as well as a notch filter at 50 Hz (UK powerline frequency). Data was then segmented into overlapping windows of fixed duration. The optimal settings here were found to be a 128 ms window with a 75% overlap. Each window was assigned a label based on a majority-vote strategy, requiring at least 75% of samples in the window to correspond to the same class.

Feature extraction was performed on each window using a set of time-domain features and pairwise correlations between the channel signals. These features formed the input to the classification model. After preprocessing and feature extraction, the dataset was split into training (70%), validation (15%), and test (15%) sets using a stratified approach to preserve class distribution. An MLP neural network was trained on the extracted features, with hyperparameters tuned using the validation set.



The trained model was then also integrated into the real-time classification application. During real-time inference, incoming data streams were filtered, segmented, normalized, and classified continuously. The application displayed the predicted movement to the user, completing the full system.

The final MLP model achieved a test accuracy of **97.49%** in offline classification, demonstrating strong generalisation performance and validating the effectiveness of the full machine learning pipeline.

### 4.1.3 Summary of Python Applications Developed

Several Python-based tools were developed over the course of this project for data collection, labelling, and real-time classification.

- **Data Collection GUI:** A Tkinter-based graphical user interface that prompts the user to perform specific hand and wrist movements while recording timestamped labels. These are saved as `.csv` files containing movement class annotations with precise Unix timestamps. This is shown alongside the OpenBCI GUI in Figure 4.8.
- **Labelling Script:** A stand-alone Python script that automatically pairs each OpenBCI sEMG recording (saved as `.txt` files) with its corresponding label file generated by the Data Collection GUI. It synchronises the timestamps, assigns class labels to each sEMG sample by matching to the nearest label time, and exports the resulting labelled dataset for model training.
- **Real-Time Classification GUI:** A second Tkinter-based GUI for real-time inference. This interface visualises the incoming multichannel EMG signals from the OpenBCI board and displays the current predicted pose using a pre-trained model. A rolling prediction buffer is used to smooth output, improving usability and reducing jitter. This is shown in Figure 4.9.

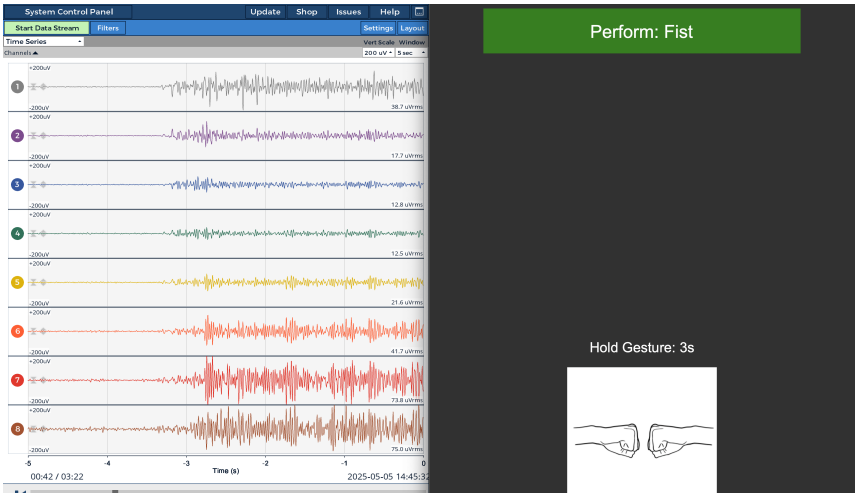



Figure 4.8: OpenBCI GUI (left) and Tkinter prompting GUI (right).

Together, these tools form a complete software pipeline for supervised sEMG gesture recognition, covering the full workflow from data acquisition to real-time feedback.

## 4.2 Design Constraints

In the following sections of this chapter, the design decisions that lead to the final design presented above are outlined. Here, the design constraints that lead to these decisions are discussed.

1. **OpenBCI Cyton Board**

   The OpenBCI Cyton Biosensing Board allows for 8 channels with a limited sampling rate of 250 Hz. This low sampling rate is a limitation, as the highest frequencies that contain useful information in an EMG signal are about 400-500 Hz. So according to the Nyquist theorem the sampling frequency should ideally be around 1 kHz [55]. Furthermore, the board in the plastic enclosure is rather large at around 60 mm x 60 mm and needs to be physically connected to the 3D-Printed system and worn on the body.

2. **sEMG Signal Quality and Noise**

   sEMG signals tend to be noisy due to the inherent method of acquisition. High or unstable impedance at the electrode-skin interface leads to variability in signal quality and noise. Motion artifacts, such as movement of the skin or electrodes during muscle contractions, are also a main source of noise. Electrical noise from power lines (50 Hz in UK) will be picked up and sEMG signals are particularly susceptible to this the signals tend to be in the 0-600 $\mu V$ range.

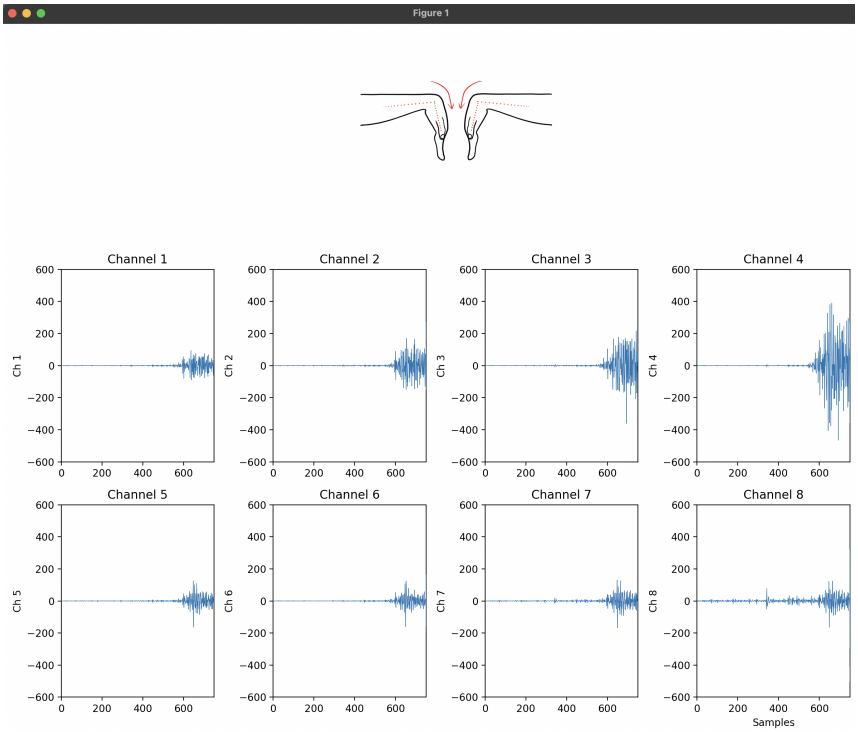



Figure 4.9: Real-Time classification application screen grab while user is performing wrist flexion (down). At the top is the current predicted output and at the bottom of the interface is the real-time filtered signals from the 8 channels.

3. **Real-Time Processing Requirements**

   Processing and classification must be performed in a streaming fashion with minimal delay, placing limits on model complexity and pre/post-processing time.

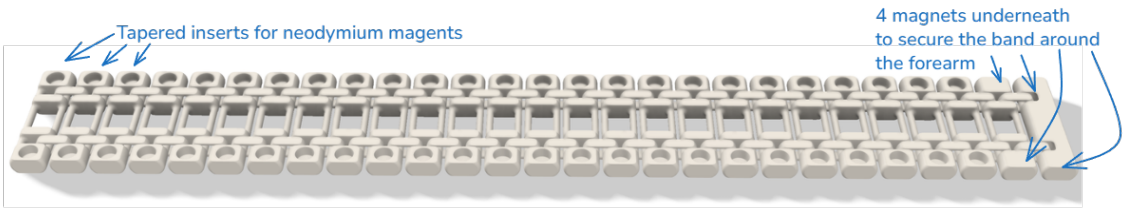
## 4.3 Hardware Design Decisions

### 4.3.1 Armband Design Decisions

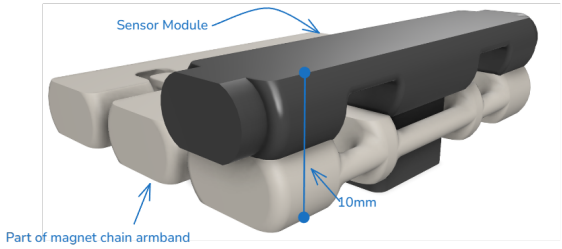
#### PLA Print-in-Place Band with Magnetic Fasteners

Many armbands of different designs were printed and tested. The main goals to be achieved by the armband was flexibility, wearability, and adjustability. Initially, the armband was made from rigid PLA plastic. The flexibility was achieved through a PIP chain design. PIP is a 3D printing technique that allows flexible 3D prints to be printed containing many separated rigid parts that interlink during the printing process. The method for fastening the band around the forearm was neodymium magnets that insert along the edges of the band, allowing the band to close at any of

the sections hence making it adjustable. A render of this design is shown in Figure 4.10a.




(a) Fusion 360 render of PLA PIP chain link armband with neodymium magnet fastening.




(b) Fusion 360 render of PLA PIP chain link armband with sensor module.

Figure 4.10: Fusion 360 visualisation of the modular chain link armband design.

This version was easy to print with a generic PLA printing profile even with the intricate PIP chain links, however assembly with the neodymium magnets was found to be difficult and time consuming as they are strong magnets placed close together. Once assembled, the armband was very easy to fasten due to the magnetic fastening system and the magnets provided enough strength so that the armband stays fastened. The print time for this design is 2 hours and 7 minutes on the ”Generic PLA” BambuLab print profile.

#### PLA Print-in-Place Band with Tolerance Fit Fasteners

The significant drawback of the earlier design was the difficulty of assembly after printing. Over fifty small neodymium magnets would need to be placed and secured, which is difficult as when placing strong magnets near each other it is very tedious to keep them apart. Furthermore, polarity needs to be kept track of and typical neodymium magnets do not indicate their polarity. The fastening solution was switched to a tolerance fit where tapered pegs at the end of the band would fit into cut out holes along the side. This eliminated any need for assembly after printing, however, securing the band around the forearm one handed become a more challenging task. The print time, using the same print profile as before, was similar at 1 hour and 56 minutes. At this point two draw backs of the PLA PIP chain design became clear. Firstly, it did not offer enough adjustability as the change in size between two fastening points is too large. This is important for an sEMG application, as the fit needs to be as flush as possible with the skin. Secondly, the

thickness of this band is 5 mm and the sensor modules protrude by 5 mm. So the total thickness is 1 cm which could be reduced. This is illustrated in Figure 4.10b


Figure 4.11: Fusion 360 render of PLA PIP chain link armband with tolerance fit.



### TPU Band with Velcro Fasteners

TPU is a flexible 3D printable plastic. Flexible filaments need to print slower than rigid filaments such as PLA, hence print times will increase when using such filaments. The first key design change was using TPU for the band material for flexibility without chain links. This allowed for the band to be only 2 mm thick. The second key design change was using Velcro and a strap design for fastening the band. This increases the customisability of the fit around the forearm as the user is no longer restricted to discrete snap positions. Despite the slower print speed that comes with printing in TPU, the much thinner and simpler design of this band reduced the print time to 1 hour and 12 minutes on the "Generic TPU" print profile. This is the armband that is shown in the final system in Figure 4.1. A 3D render of the armband with a sensor attached is shown in Figure 4.12.

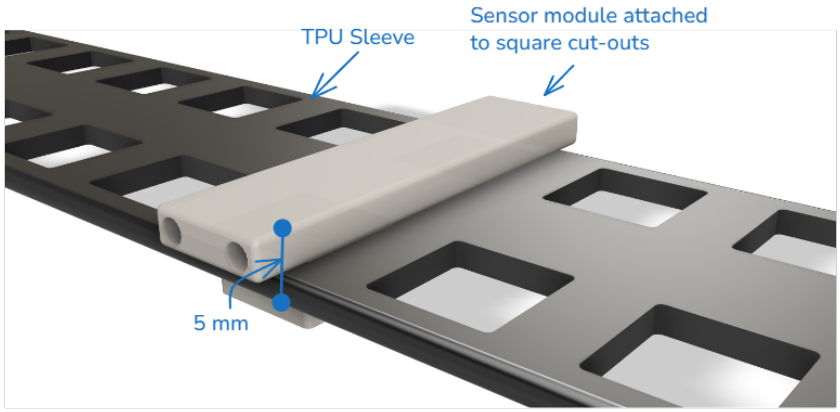



Figure 4.12: Fusion 360 render of TPU armband with sensor module attached.

A comparison of all designs is shown in Table 4.1

Table 4.1: Comparison of Armband Designs

| Design | Fastener | Print Time | Assembly | Wearability |
|---|---|---|---|---|
| PLA Chain | Magnets | 2 hr 7 min | Difficult | Good, magnets are convenient |
| PLA Chain | peg-and-hole fit | 1 hr 56 min | None | Difficult to put on |
| TPU | Velcro strap | 1 hr 12 min | Easy | Great, thin and more adjustable |

### 4.3.2 Modular Sensor Design and Integration



The design of the sensor modules focused on achieving reliable signal acquisition, modularity, and wearability. To ensure a low-resistance and secure electrical connection between the sensor and the electrode wire, the electrode housing was designed with a very tight tolerance, allowing it to fit snugly over the stripped end of the wire without additional fastening components. Modularity was a key requirement; each sensor unit was engineered to snap onto the sleeve and be re-positionable to any location, enabling flexible placement according to the subject's forearm anatomy and ensuring consistent muscle coverage. Finally, to maintain comfort and wearability during use, the total thickness of the TPU Armband and attached sensors was reduced to just 5 mm, contributing to a lightweight and wearable solution suitable for extended sessions. The key advantage with: modular sensors, an adjustable armband and the ability to position those sensors in any position in the armband, is to be able to adapt the signal acquisition system to various users to get the best possible signal quality. The largest source of noise in sEMG systems are measurement artifacts due to poor electrode contact and motion artifacts due to the electrode moving on the skin during muscle contraction. Focussing on making a highly adjustable system lead to great signal quality, as quantified in Chapter 7.

### 4.3.3 Bias Placement

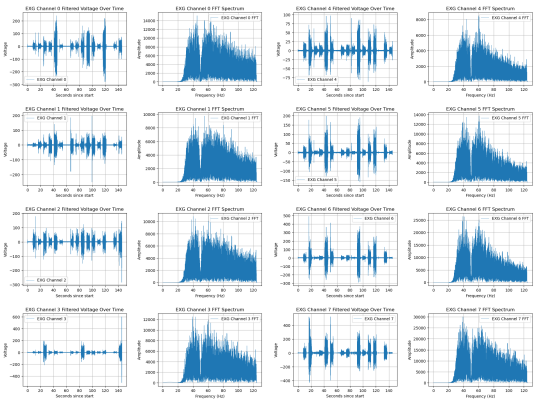
A suitable placement for the bias electrode in sEMG systems typically requires a bony, electrically neutral region to ensure stable reference signals. On the forearm, the two main options are the bony prominence near the elbow and the pisiform bone at the wrist. Extending a wearable all the way to the elbow introduces significant challenges in terms of fit, comfort, and adjustability. Preliminary testing revealed that positioning the bias electrode over the pisiform bone at the wrist provided stable, high-quality signals without introducing noticeable artefacts or degradation. This finding enabled the use of a wrist-based bias placement in the final design, allowing for a more compact, TPU-based armband rather than a full-length sleeve, while still ensuring reliable signal acquisition. However, the drawback to this approach is that now a wrist band must be worn with

the reference electrode. The ”Bias Band” can be seen worn in the full system in Figure 4.1 and a render of the model is shown in Figure 4.5a.

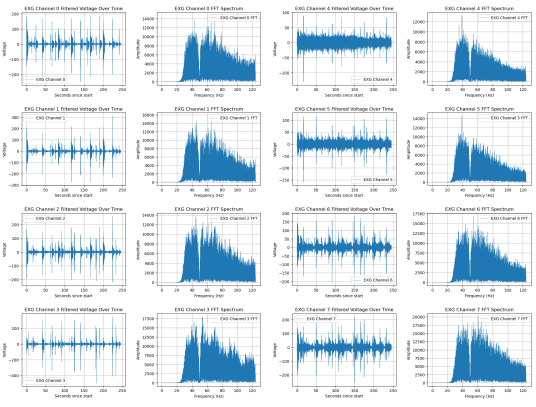
### 4.3.4 Characterisation of the Sensor Modules

The signal quality obtained from each sensor module is highly dependent on the quality of its physical contact with the skin. However, this dependency is acceptable in our system as the wearable has been designed to be highly customisable, allowing for consistent and secure placement across a range of forearm geometries. Quantitative SNR values are presented in the results chapter (Chapter 7) in Table 7.1, while qualitative signal quality can be observed from raw waveforms.

Figure 4.13a shows an example where the fit was optimal across all channels, resulting in clean signals with minimal baseline drift or artefacts. In contrast, Figure 4.13b depicts a scenario where some sensor modules exhibited poor contact due to local electrode lift, which introduced visible noise and degraded the signal quality on affected channels. These examples highlight the importance of careful sensor placement adjustment and illustrate the system's ability to maintain high signal fidelity under good fit conditions.



(a) Good electrode fit — low noise across all channels.



(b) Worse electrode fit — elevated noise on channels 4–8.

Figure 4.13: Comparison of raw sEMG signals under different fitting conditions. Each sub-figure displays the sEMG waveform (left) and its corresponding FFT transform (right) for all 8 channels.

### 4.3.5 Cost of Materials

A breakdown of the cost of materials used to print and assemble the final armband is provided in Table 4.2.

Table 4.2: Material Cost Breakdown for One 8-Channel Armband Assembly

| Item | Unit Price | Amount Used | Cost |
|---|---|---|---|
| 95A TPU (for armband) | £18.00 / kg | 14.15 g | £0.25 |
| PLA (for sensor modules) | £10.00 / kg | 13.44 g (8 × 1.68 g) | £0.13 |
| ECG Electrodes | £6.80 / 30 | 16 (8 × 2) | £3.63 |
| Male/Female Jumper Wires | £3.99 / 40 | 16 (8 × 2) | £1.60 |
| Heavy-duty Velcro | £6.48 / m | 10 cm | £0.65 |
| **Total Cost** | | | **£6.26** |



This estimate assumes the armband makes full use of all eight channels available on the Cyton board. The total cost may be reduced if fewer sensor modules are used.

## 4.4 Software Design Decisions

### 4.4.1 Data Collection Protocol and GUI

To enable effective training of machine learning models for gesture classification, a structured data collection protocol was essential. A GUI was developed using `Tkinter` to guide users through the data acquisition process. This prompting interface displayed movement instructions and a count-down timer, ensuring consistency across recording sessions. The GUI also allowed for configuration of trial duration, rest intervals, and the number of repetitions per gesture. Gestures are presented to the user in a randomised order to avoid any biases arising from the ordering of the gestures. Simultaneously, sEMG signals from the 3D printed sleeve were recorded using the OpenBCI GUI software and stored as timestamped CSV files. A post-processing Python script was implemented to synchronise and merge the labels generated by the GUI with the corresponding EMG signal recordings, this is further discussed below. A screen capture of using the Data Collection GUI and OpenBCI GUI is presented in Figure 4.8.

The data collection protocol is that the user is prompted to perform and hold a gesture for a specified period of time, by default 4 seconds. Then the user rests and is told what the following gesture will be with a countdown. This is then repeated for each gesture in a random order. Then the entire process is repeated according to the specified repetitions set, by default 3 repetition but 5 repetitions was used in this case.

### 4.4.2 Adding Labels to Raw sEMG Data

To enable supervised training of the classification model, raw sEMG signals collected from the OpenBCI board were labelled according to the prompted movements. The OpenBCI GUI outputs a `.txt` file containing time-stamped multichannel sEMG signals, while the custom prompting application developed as part of this project simultaneously generates a `.csv` file containing Unix timestamps and the associated movement classes. A Python labelling script was developed to synchronise these two data sources. It recursively scans through all subfolders within the `Output Files` directory, pairing each sEMG file with its corresponding label file based on file extensions. For each matched pair, the script trims the sEMG recording to only include samples within the time range covered by the label file. Each sEMG row is then assigned a class label by matching its timestamp to the nearest label timestamp using an absolute time difference. The resulting labelled data is saved as a new `.txt` file within the same subdirectory. This approach automates the preprocessing of multiple sessions, allowing alignment of raw data with ground truth annotations by simply running the script.



### 4.4.3 Signal Preprocessing

#### Signal Filtering

In order to prepare the raw sEMG data for classification, a series of signal preprocessing steps were applied. Firstly, filtering was carried out to improve signal quality and remove unwanted artefacts. A band-pass filter was used to eliminate low-frequency drift and motion artefacts, while a notch filter was applied to suppress power-line interference, in our case in the UK, at 50 Hz. Example waveforms and corresponding frequency spectra after filtering are provided in Figure 4.14.

Offline filtering was performed using zero-phase filtering via the `filtfilt` function from `scipy.signal`, which applies the filter forward and backward to avoid phase distortion. The primary filter was an $8^{\text{th}}$-order Butterworth band-pass filter, supplemented by a notch filter centred at 50 Hz (with a quality factor of 30) to remove power-line interference.

The passband was configured from 30 Hz to 124 Hz. The 30 Hz low-cut frequency was chosen to attenuate low-frequency motion artifacts, a common source of noise. The high-cut frequency was intentionally set at 124 Hz, just below the Nyquist limit of 125 Hz for the system's 250 Hz sampling rate. This decision effectively turns the filter into a wide-band filter that removes low-frequency

noise while retaining nearly the entire usable spectrum of the recorded signal. As useful sEMG information exists up to 400-500 Hz [9], this approach ensures that all information available within the hardware's sampling limit is preserved. This design choice was adopted after it was empirically observed that the classifier's performance benefited significantly from having access to this broader range of frequency information compared to using a narrower filter.



In contrast, real-time filtering requires causal filters that operate on streaming data without knowledge of future samples. For this, the `lfilter` function from `scipy.signal` was used in combination with `lfilter_zi` to initialise filter state. The real-time band-pass filter used the same order and cutoff frequencies as offline, but due to its causal nature, introduces phase delay.

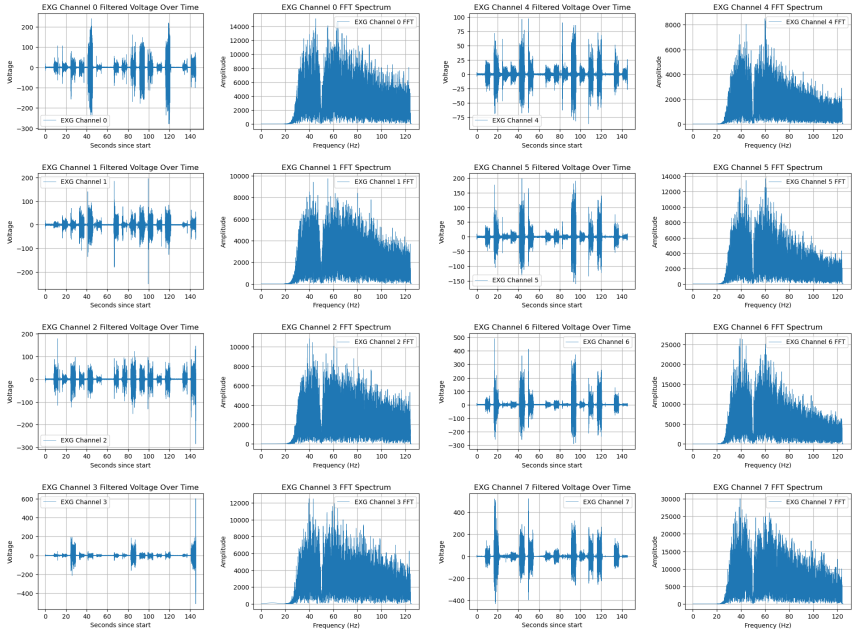



Figure 4.14: Example 8 channel waveforms and frequency spectra.

### Normalisation

Following this, normalisation was performed to mitigate inter-subject and inter-session amplitude variations, before combining recording sessions into a single dataset. Firstly, each channel's amplitude was clipped to $\pm 600\,\mu$V to suppress large artefacts before normalisation or classification. Then, each channel was normalised using z-score standardisation, defined as:

$$x' = \frac{x - \mu}{\sigma} \tag{4.2}$$

where $x$ is the raw signal, $\mu$ is the mean, and $\sigma$ is the standard deviation of the channel over the session. This ensures each feature has zero mean and unit variance, improving training stability and comparability across sessions.

At this point, recordings obtained from the right arm were relabelled with reversed channel order to ensure consistency when combined with those from the left arm.



### Windowing

The preprocessed signal was then segmented using a windowing approach, in which fixed-duration windows were extracted from the recording for feature computation and classification. The choice of window duration, sampling rate, and overlap percentage significantly affects temporal resolution and model performance. The size of the resulting windowed dataset can be calculated as follows.

Let $D$ be the total recording duration (s), $f_s$ the sampling rate (Hz), $W$ the window duration (s), and $O$ the window overlap (as a decimal, e.g., 0.25 for 25%).

Define:

$$N = D \cdot f_s \quad \text{(total number of samples)} \tag{4.3}$$

$$W_s = W \cdot f_s \quad \text{(window size in samples)} \tag{4.4}$$

$$S = W_s \cdot (1 - O) \quad \text{(step size in samples)} \tag{4.5}$$

Then, the number of windows is given by:

$$\text{Number of windows} = \left\lfloor \frac{N - W_s}{S} \right\rfloor + 1 \tag{4.6}$$

A smaller window size results in lower real-time latency, as decisions can be made more frequently and with less delay. It also increases the number of windows extracted from a given recording, effectively enlarging the training dataset and potentially improving generalisation. However, smaller windows contain fewer signal samples, which may degrade classifier performance due to reduced information per window. This can lead to noisier features and poorer separability between classes. Therefore, selecting an appropriate window size involves balancing the trade-off between latency, dataset size, and classification accuracy.

### 4.4.4 Data Augmentation

Data augmentation was explored to improve the generalisation ability of the trained classifiers and to reduce over-fitting, especially given the limited amount of labelled sEMG data that could be collected from each subject. By artificially expanding the training dataset with plausible variations of the recorded signals, the model learns to become more robust to minor variations in signal amplitude, noise, and electrode placement. Several augmentation techniques were implemented, including the addition of Gaussian noise to simulate measurement variability, amplitude scaling to account for differences in muscle contraction strength, and temporal jittering to simulate timing inconsistencies. These augmentations preserve the underlying class semantics while introducing enough variability to regularise training.

### 4.4.5 Feature Selection Process

To identify the most informative features for EMG signal classification, a multi-step feature selection process was applied. Initially, a broad set of time-domain, statistical, and non-linear features were extracted, these are all outlined in Appendix A. Random Forest feature importance scores were computed to evaluate each feature's individual contribution to classification accuracy. RFE was used as a wrapper-based method to obtain a feature set from these scores. RFE works by recursively training a model (in this case a Random Forest), ranking features by their importance, and removing the least important ones in each iteration until a desired number of features is reached. This process helps identify a compact subset of features that optimally support the classifier's performance without redundancy. Features that involved high computational complexity—such as those requiring iterative procedures or expensive transforms—were excluded to ensure compatibility with real-time applications. Following this, pairwise correlations between features were analysed. Features in the reduced subset exhibiting high collinearity were considered redundant, and in such cases, one feature from each highly correlated pair was removed to reduce redundancy and improve generalisability. The final features selected after this process were:

- Log Root Mean Square (ln_rms)
- Average Amplitude Change (aac)
- Mean Absolute Value Slope (mavs)
- Slope Sign Changes (ssc)

- Wilson Amplitude (wamp)
- Skewness
- Simple Square Integral (ssi)

The correlation matrix of the final features from the dataset are shown in Figure 4.15. The lower triangle and diagonal are omitted.



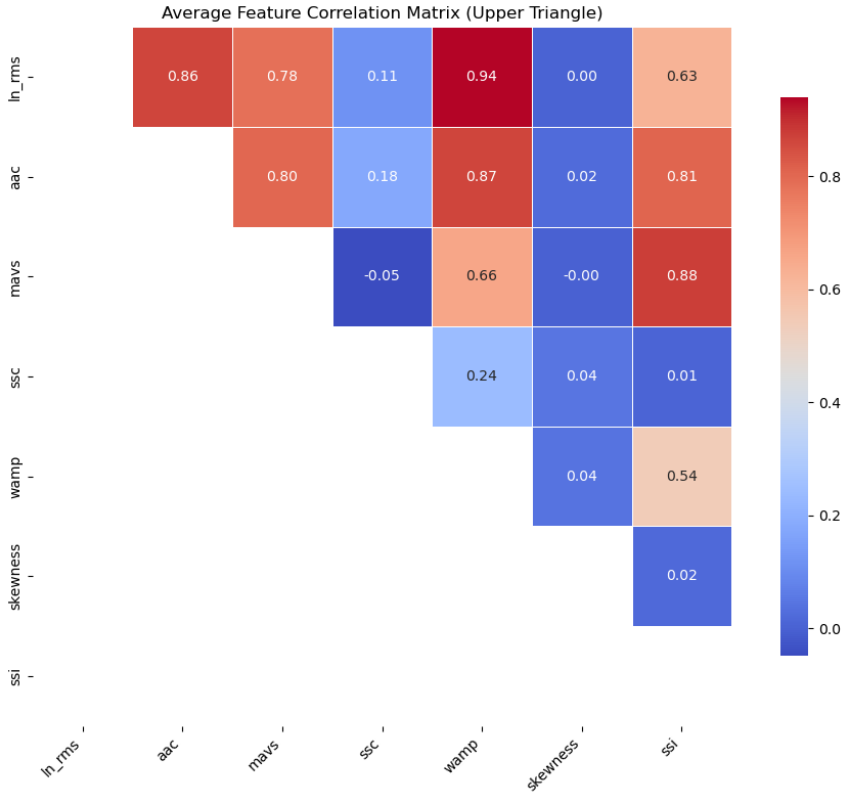



Figure 4.15: Final feature set correlation matrix, higher correlation in red and lower correlation in blue.

The pairwise correlations between the 8 channels are also added to the feature set, adding 28 features for 8 channels. Including the correlations between EMG channels in the feature set is valuable because it captures spatial dependencies and muscle coordination patterns that per-channel features alone cannot reflect. sEMG signals recorded from multiple channels are often not independent, as muscles tend to activate in coordinated groups during specific movements. Correlation features quantify the degree to which pairs of muscles co-activate, providing insights into these synergies. This becomes particularly useful for distinguishing gestures that produce similar individual channel features but differ in how muscles interact spatially. Additionally, correlation coefficients are scale-invariant, making the model more robust to variations in amplitude caused

by differences in electrode placement or skin conductance. Overall, incorporating inter-channel correlations helps encode richer spatial information that can improve classification performance.

The length of the feature vector for each data point is 84 (7 time domain features x 8 channels + 24 pair wise correlations).

To test whether this adds more useful information, a simple SVM classifier with linear kernel was trained on the original feature set without the correlations and then with the correlations. The results are shown in Table 4.3. These results show a significant increase in test accuracy when the pairwise correlations are added to the feature set, which strongly suggests they add significantly useful information.

| | Test Accuracy |
|---|---|
| Without Correlations | 78.76% |
| With Correlations | 88.19% |

Table 4.3: Effect of adding pairwise channel correlations to feature set.

### 4.4.6 Machine Learning Model Evaluation

#### Overview of Model Selection and Training Process

A range of machine learning models were trained and evaluated to identify the most effective approach for classifying hand movements based on sEMG signals. The process began with traditional models such as SVM, KNN, Random Forest (RF), and LDA, which provided a strong baseline and enabled rapid iteration over different feature extraction and selection strategies. Subsequently, more complex models including MLPs and deep learning approaches such as Long Short-Term Memory (LSTM) networks, CNN, and hybrid CNN-LSTM architectures were explored to better capture the temporal and spatial dynamics of the sEMG signals. The highest classification performance was achieved using an MLP, which attained a test accuracy of 97.49%. The deep learning approaches do not use a feature set. Instead, they operate directly on the filtered sEMG signal windows, learning relevant spatial and temporal representations automatically through their layered architectures. This enables the models to capture complex patterns and dependencies in the data without relying on handcrafted features. However, despite their expressive power, these models typically require more data and computational resources to train effectively. In this project, the MLP outperformed the more complex deep learning models, likely due to the relatively small dataset size (discussed below) and the effectiveness of the extracted features, which provided a compact and informative representation of the sEMG signals.

Model performance was optimised through systematic parameter tuning. A 70-15-15 split was used for the training, validation, and test sets respectively. All hyper-parameter selection and tuning were conducted using the training and validation sets only; the test set was held out and used once at the end for final evaluation to ensure an unbiased performance estimate. This is best practise in ML classification. Parameters that were optimised include the model architecture, training settings such as batch size and learning rate, and level of regularisation through methods such as L2 regularisation and dropout layers.

The window size and overlap percentage, key parameters in the segmentation of the sEMG signal, were optimised via grid search, with the best results obtained using a 128 ms window and 75% overlap. This configuration provided the optimal balance between capturing sufficient temporal context and maintaining a high number of training samples. The results of this grid search are given in Table 4.4. Interestingly, this trend of benefitting from a greater number of training examples (due to a smaller window size and higher overlap percentage) was not observed when training simpler ML models. These results, provided in Appendix B Tables B.1, B.2, B.3, show that the average validation accuracy decreases as window size decreases when simpler models are utilised.

| | **Overlap (%)** | | | |
|---|---|---|---|---|
| **Window Size (ms)** | **0%** | **25%** | **50%** | **75%** |
| 0.128 | 94.38 | 95.30 | 96.33 | **97.50** |
| 0.256 | 93.82 | 95.69 | 95.64 | 96.84 |
| 0.512 | 93.40 | 95.50 | 96.25 | 96.48 |

Table 4.4: MLP validation accuracy (%) for different combinations of window size and overlap percentage.

We found that adding augmented examples to the training set did not improve training performance by any significant amount, so it was not used in the final system. A **total of 40.3 minutes** of data was gathered for this dataset. The characteristics of the volunteers in this data set is discussed above in Section 4.1.2. This yielded 75,558 windows, as calculated using Equation 4.6. Only windows with at least 75% overlap with a single class were retained. This criterion led to the exclusion of 1.39% of the windows, resulting in **74,511 usable samples**, as outlined in Table 4.6.

### Highest Performing Model

The model that performed the best was an MLP model utilising Dropout layers for regularisation. Regularisation in ML involves techniques that prevent complex models from over-fitting to a dataset

in hopes of generalising well to unseen data. Dropout is a regularisation technique used in neural networks [60]. It works by randomly setting a fraction of neurons to zero during training. This prevents individual neurons from becoming overly reliant on specific inputs, improving the model's generalization ability and robustness. The model architecture is presented in Table 4.5. The final hyperparameters are presented in Table 4.6.



| Layer | Type | Output Size | Activation / Dropout |
|---|---|---|---|
| 1 | Dense | 256 | ReLU |
| 2 | Dropout | - | 30% |
| 3 | Dense | 128 | ReLU |
| 4 | Dropout | - | 20% |
| 5 | Dense (Output) | *6 classes* | Softmax |

Table 4.5: Architecture of the MLP model.

| Parameter | Value |
|---|---|
| Final Test Accuracy | 97.49% |
| Window Duration | 128 ms |
| Overlap Percentage | 75% |
| Sampling Rate | 250 Hz |
| Dataset size (windows) | 74,511 |
| Trainable parameters | 55,430 |
| Train / Validation / Test Split | 70% / 15% / 15% |
| Loss Function | Categorical Cross-Entropy |
| Optimizer | Adam |
| Learning Rate | 0.001 |
| Batch Size | 16 |
| Epochs | 200 (Early stopping with patience = 10) |
| Early Stopping Metric | Validation Loss |

Table 4.6: Training configuration and hyperparameters for MLP model.

### 4.4.7 Real-Time Classification Application

A Python application was developed for real-time classification of sEMG signals using the custom 3D-printed hardware and adapted versions of the trained ML models. The best performing MLP architecture from earlier analysis was used, with two changes to the training pipeline. Firstly, digital signal filtering techniques used in offline processing—such as non-causal filters—are not suitable for real-time applications, as they require access to future data points in the signal. These filters (e.g., zero-phase filters like `filtfilt`) offer high accuracy but rely on the entire signal being available beforehand. In contrast, real-time filtering must use **causal filters**, where the output at any time depends only on current and past inputs. This allows the filter state to be updated incrementally as new samples arrive, making it appropriate for live signal processing. Secondly, as we have already evaluated the models, we now wish to train on all available data. Rather than

having a training, validation and test split as before, we now have only a training and validation split. The split used for testing the application was 85% training and 15% validation. The reason for keeping a validation set is that it is used for the early stopping callback in the training process, which is important to catch over-fitting when the validation accuracy is no longer improving. The final validation accuracy after this model was trained was 97.83%. It was trained with the same diverse user dataset of ten subjects as before.



The application is multi-threaded to ensure non-blocking operation of both the data stream and the graphical user interface (GUI). The main thread is responsible for updating the GUI at defined intervals using data stored in shared buffers, which are continuously updated by the data streaming thread. This streaming thread manages the connection to the OpenBCI board, populates the sEMG channel buffers, performs predictions, and updates a dedicated predictions buffer. An image of the application GUI while in use is shown in Figure 4.9.

#### Application Configuration Options

The application provides several configuration options to enable flexible deployment across different setups. It establishes a connection to the Cyton board using the OpenBCI Python library, which handles receiving the raw data from the Cyton Board via the RFDuino dongle. Users can select and load a pre-trained model and corresponding scaler, with models specifically trained using the same real-time preprocessing pipeline to ensure compatibility. Additionally, the application allows the user to specify whether the device is worn on the left or right arm; in the case of the right arm, the sEMG channel order is reversed.

#### Data Streaming Thread

The data streaming thread is responsible for real-time signal processing and classification. Unlike offline preprocessing, which used non-causal zero-phase filters such as `filtfilt` from `scipy.signal` which apply the filter forwards and backwards in time—the real-time implementation uses causal filters to ensure that filtering only relies on current and past samples, which is necessary for live signal processing. A buffer is maintained for each sEMG channel, with its size corresponding to the model's required window duration. This buffer is implemented using a queue data structure to efficiently remove the oldest samples as new data arrives. Classifications are made every 64 ms and each prediction is appended to a rolling prediction buffer of size 8. When the graphical

interface is refreshed, it displays the most frequent class within this buffer, effectively applying a majority-voting scheme to smooth out jittery predictions and provide a more stable user experience.

#### GUI Thread

The GUI thread runs on the main thread and is designed to provide a simple and clear visualisation of the real-time classification results. It displays the currently predicted pose alongside the real-time sEMG waveforms from all 8 channels. The GUI is refreshed at regular intervals of 50 ms.



## 4.5 Chapter Summary

In summary, this chapter has outlined the complete design and engineering process behind the real-time sEMG-based gesture classification system. After evaluating several iterations, the final hardware design adopted a flexible TPU armband with Velcro fastening and modular PLA sensor modules, allowing customisable sensor placement. These design choices lead to high signal quality while ensuring adaptability across users. On the software side, a robust signal processing and feature extraction pipeline was developed, used in a MLP classifier trained on both individual channel features and inter-channel correlations. The final model achieved a test accuracy of **97.49%**, validating the pipeline's effectiveness. This model was then used to in a multi-threaded real-time classification application, which integrated causal signal filtering and prediction smoothing to deliver responsive and stable performance. The outcomes presented here form the basis for the evaluation and validation of system performance in subsequent chapters.

# 5

# Implementation Details



## Contents

**5.1 Dataflow Overview . . . . . 51**
**5.2 Data Collection and Labelling . . . . . 52**
**5.3 Signal Processing and Feature Extraction . . . . . 54**
5.3.1 Signal Filtering . . . . . 54
5.3.2 Feature Extraction . . . . . 55
**5.4 Classification System . . . . . 55**
5.4.1 Offline Model Training . . . . . 55
5.4.2 Real-Time Classification System . . . . . 56
**5.5 Chapter Summary . . . . . 57**

This chapter details the technical implementation of the sEMG gesture recognition system, bridging the design concepts outlined in Chapter 4 with the final working hardware and software. The discussion covers the complete system architecture: the software pipelines for data collection, signal processing, model training, and real-time classification. Key code segments are provided to illustrate the core logic of the implementation, focusing on the most challenging or novel aspects of the project. The presented code has been edited from its original form to improve clarity, and only the most relevant and insightful fragments are included. The complete, unabridged scripts for all software components are available in the project's public GitHub repositories, with access details provided in the Chapter **??**.

## 5.1 Dataflow Overview

The implemented system is architected as an end-to-end pipeline that transforms raw sEMG recordings into discrete gesture classifications. The data flows through two broad stages: hardware ac-

quisition and software processing. A high level diagram outlining the entire data flow is given in Figure 5.1.

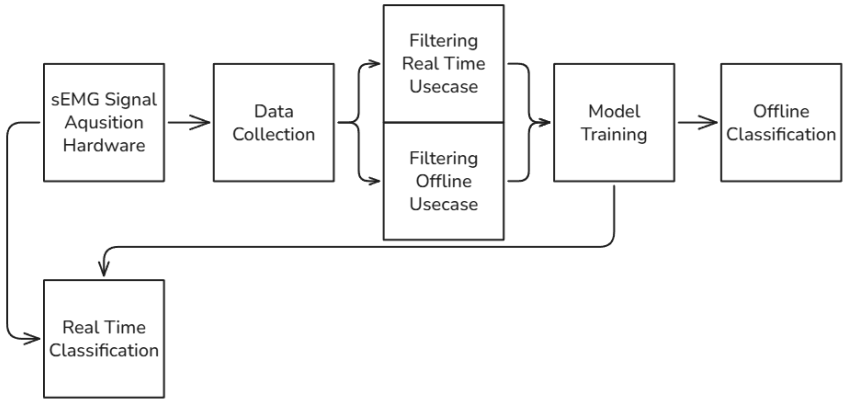



Figure 5.1: High level data flow diagram.

## 5.2 Data Collection and Labelling

A structured and precisely timed data collection process was critical for building a high-quality training dataset. This was managed by two separate GUIs and a subsequent processing script. The custom `Tkinter` application guided participants through the protocol, displaying the gesture to perform, a countdown timer, and the next gesture during rest periods. The application allows for the configuration of gesture duration, rest duration, and the number of repetitions. A block diagram, which illustrates the flow of data from the developed acquisition hardware to a labelled raw sEMG dataframe, is given in Figure 5.2.

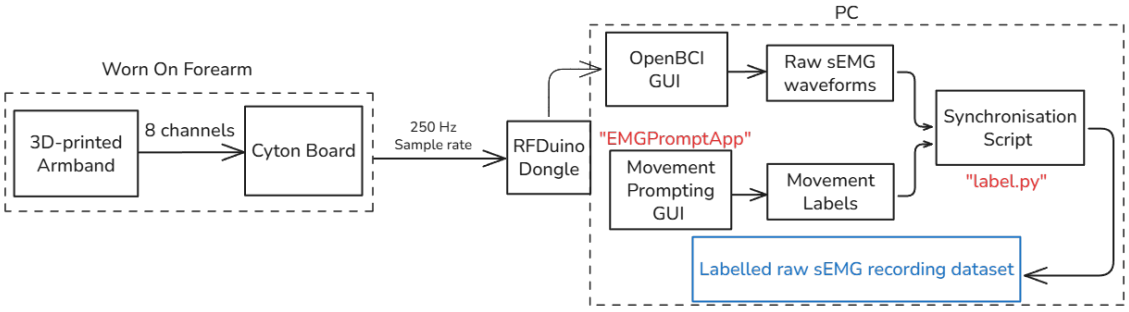



Figure 5.2: Block diagram showing data flow from acquisition hardware to raw sEMG dataframe.

The user is guided through prompts shown to them, which are presented in a random order every repetition. While the user followed the prompts, raw sEMG data was recorded using the OpenBCI GUI, which saved the data as a `.txt` file with Unix timestamps. The prompting GUI saved a corresponding `.csv` file containing the sequence of gestures performed and their precise

start and end timestamps. A detailed state diagram that outlines the control flow is presented in Figure 5.3.

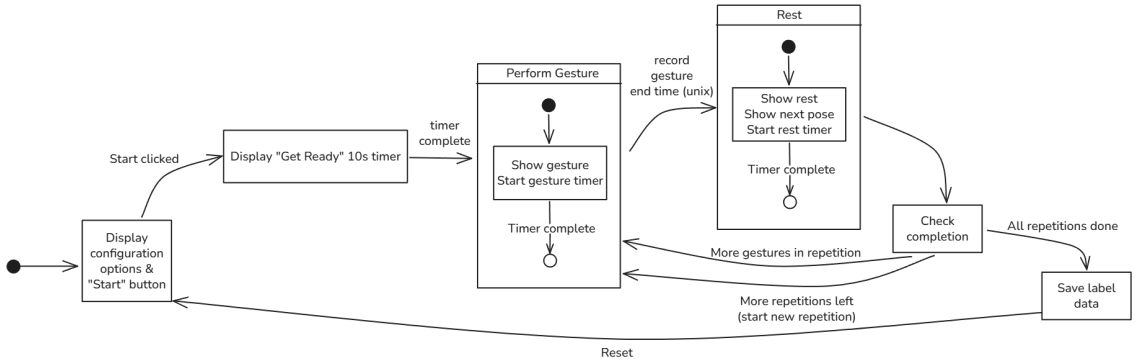



Figure 5.3: State machine diagram of Tkinter data collection prompting app.

A Python script was then used to automatically ”zip” these two files together. The script recursively scans a directory, identifies corresponding sEMG and label files, and merges them. The core of the labelling logic, shown in Listing 5.1, involves assigning a class to each sEMG sample by finding the label with the nearest timestamp. This automated approach ensures accurate and efficient annotation of large datasets.

```
# Snippet from the automated labelling script

def label_emg_df(emg_df, label_df, filename=None): #
    # Filter EMG data to the time range covered by the labels
    min_label_time = label_df['time_unix'].min() #
    max_label_time = label_df['time_unix'].max() #
    emg_df = emg_df[(emg_df[' Timestamp'] >= min_label_time) &
                    (emg_df[' Timestamp'] <= max_label_time)].copy() #

    # Assign a class to each EMG row by finding the closest timestamp
    # in the label file. The .idxmin() method finds the index of the
    # minimum value, which corresponds to the closest time point.
    emg_df['class'] = emg_df[' Timestamp'].apply(
        lambda t: label_df.loc[(label_df['time_unix'] - t).abs().idxmin(), 'class']
    ) #

    # Save the newly labelled DataFrame with raw sEMG Data
    emg_df.to_csv((filename or 'emg_labeled') + ".txt", index=False) #
    print(f"Labeled EMG data saved to {(filename or 'emg_labeled') + '.txt'}") #
```

Listing 5.1: Core function from the automated labelling script. It aligns sEMG data with timestamped labels by finding the nearest time point for each sample.

## 5.3 Signal Processing and Feature Extraction

The raw sEMG signals were processed to enhance quality and extract discriminative features suitable for machine learning. A detailed block diagram that illustrates the logical flow of this process is provided in Figure 5.4.

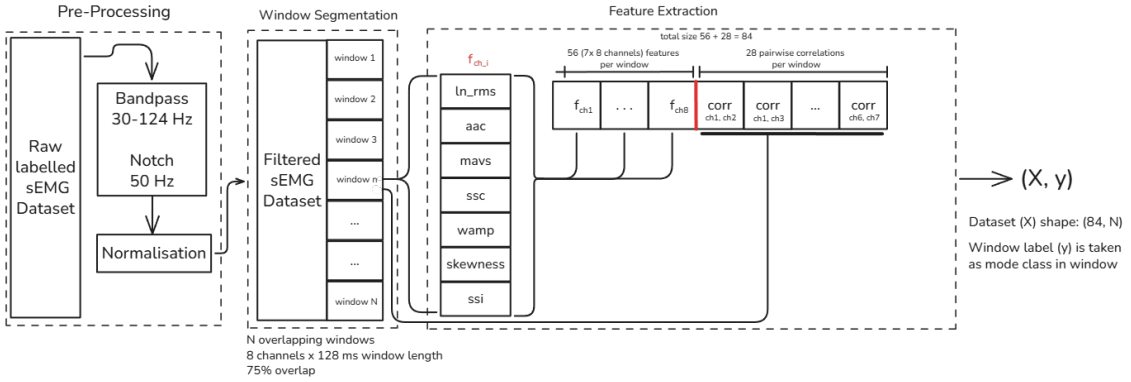



Figure 5.4: Block diagram showing logical flow of signal processing and feature extraction.

### 5.3.1 Signal Filtering

As discussed in the design chapter, a band-pass filter (30–124 Hz) and a 50 Hz notch filter were applied. For offline model training, a zero-phase filter (`filtfilt`) was used to prevent signal distortion. For the real-time application, a causal filter (`lfilter`) was necessary, as it operates on data streams without requiring future samples. So in the real-time model training pipeline, these filters were used instead. The filters were imported from SciPy's signal library [61], as shown in the filtering example given in Listing 5.2.

```
# Example filter design for both offline and real-time use
from scipy.signal import filtfilt # offline filter used
from scipy.signal import lfilter, lfilter_zi # real-time filters used
from scipy.signal import butter, iirnotch # both require these to define bands

# Design offline bandpass and notch filters
b_band, a_band = butter(order, [low, high], btype='band') #
b_notch, a_notch = iirnotch(notch_freq) #
filtered_data = filtfilt(b_notch, a_notch, filtfilt(b_band, a_band, data) ) # example of
    how offline filtering can be applied
# ...
# ...
# Design real-time filter that has a state and updates that state
```

```
filter_state = lfilter_zi(b, a)
x_band, band_states[i] = lfilter(b_band, a_band, [latest_data_sample], zi=band_states[i
    ]) # returns the output and the filter state given a new data sample
x_notch, notch_states[i] = lfilter(b_notch, a_notch, x_band, zi=notch_states[i]) # does
    the same but notch rather than band
filtered_data = x_notch
```

Listing 5.2: Filter design using SciPy for both offline and real-time processing.

### 5.3.2 Feature Extraction

After filtering, the signal was segmented into 128 ms windows with 75% overlap. From each window, a set of seven time-domain features was calculated for each of the eight channels. In addition, the pairwise Pearson correlation coefficient was computed for every combination of the eight channels, capturing information about muscle co-activation. This resulted in a feature vector of length 84 (7 features $\times$ 8 channels + 28 correlations) for each window. As discussed in Chapter 4, 40.3 minutes (excluding rest states) of labelled data was included in the data set used. After windowing, this resulted in 74,511 labelled windows. As illustrated in Figure 5.4, this means that the data matrix used for training, validation, and testing has shape (74,511, 84).

## 5.4 Classification System

The core of the software is the machine learning pipeline, implemented for both offline training and real-time inference.

### 5.4.1 Offline Model Training

An MLP model was implemented using the TensorFlow [62] and Keras [63] Python libraries. The architecture was defined as a `Sequential` model, consisting of two dense hidden layers with ReLU activation, followed by dropout layers for regularisation to prevent overfitting. The final layer is a dense layer with a softmax activation function to output a probability distribution over the six gesture classes. As discussed in Chapter 4, the architecture and hyperparameters were tuned on the validation set. The final parameters are shown in Tables 4.5, 4.6.

### 5.4.2 Real-Time Classification System

The real-time application loads the trained Keras model and the Scikit-learn scaler saved during training. The application is multi-threaded to separate the continuous data acquisition from the GUI updates. A key feature for improving usability is the prediction buffer, which smooths the classifier's output. This was implemented using a `deque` from Python's `collections` library, a highly efficient double-ended queue data structure with a fixed maximum length. The maximum length used was 8. New predictions are added to one end, and old ones are automatically discarded from the other. A `Counter` is used to find the most common prediction in the buffer, implementing a majority vote. A high level diagram of the logic and interaction between the two threads is shown in Figure 5.5. This implementation is shown in Listing 5.3.

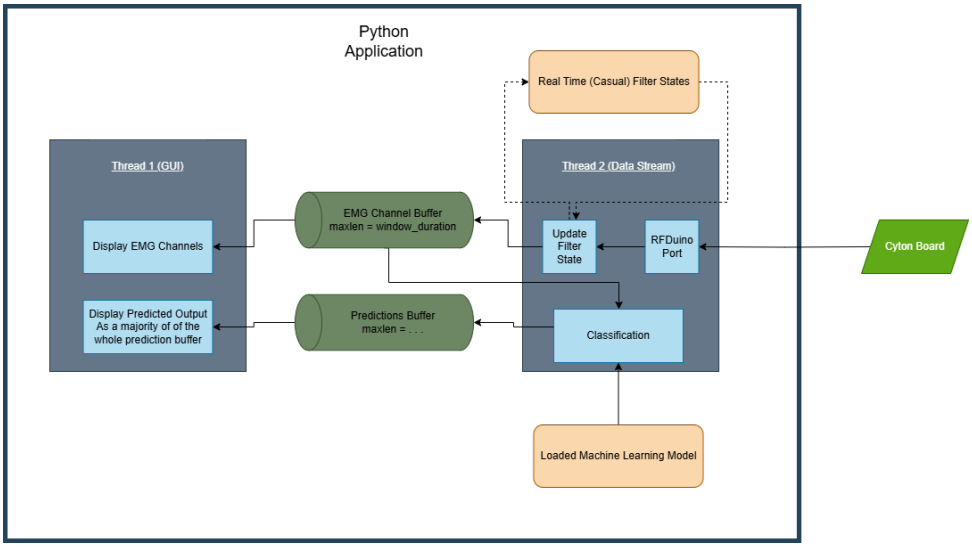



Figure 5.5: Real-Time classification application high level design flowchart.

```
# Snippet showing the prediction smoothing implementation
# A deque of fixed size to store the last N predictions
prediction_buffer = deque(maxlen=8) #

# --- In the data streaming thread ---
def handle_stream(sample):
    # ... (filtering and buffer population logic) ...
    # When a full window of data is ready for classification
    if all(len(buf) == buffer_size for buf in channel_buffers): #
        window = pd.DataFrame({f" EXG Channel {i}": list(channel_buffers[i]) for i in range(8)}) #
        prediction = classify_real_time(window) # returns a single integer prediction (e.g., 0-5)
        prediction_buffer.append(prediction) #
```

```
13
14
15 # --- In the GUI update function (main thread) ---
16 def animate(i):
17     current_pose = Counter(prediction_buffer).most_common(1) # Find the most common
       element in the buffer (majority vote)
18
19     img_display.set_data(POSE_IMAGES[current_pose]) # Update the GUI with the image
20     return [img_display]
```

Listing 5.3: Implementation of the prediction smoothing buffer using a `deque` and `Counter` for majority voting in the real-time application.



This implementation detail was crucial for creating a stable and responsive user experience, mitigating the jitter that can arise from single, instantaneous predictions.

## 5.5 Chapter Summary

This chapter has detailed the technical implementation of the end-to-end sEMG classification pipeline, with the system's logic and data flow illustrated through a series of diagrams and key code listings. The process begins with a structured data collection protocol, which uses a custom `Tkinter` application and the OpenBCI GUI to gather synchronised signal and label data, followed by an automated script to produce a fully annotated dataset.

The implementation of the signal processing and feature extraction pipeline was then presented. This stage transforms the raw, labelled data into a feature matrix suitable for machine learning, employing distinct offline and real-time filtering strategies and calculating a final 84-dimension feature vector for each 128 ms signal window. Finally, the chapter detailed the implementation of the classification system itself. This included the definition and training of the MLP model using Keras, as well as the architecture of the multi-threaded real-time application. An interesting component of this application, the majority-voting prediction buffer, was implemented using a `deque` data structure to ensure a stable user experience. This complete implementation provides the practical foundation for the system evaluation discussed in the subsequent chapters.

5

# 6

# Test Plan and Verification



Contents

**6.1 Hardware Verification . . . . . 59**
6.1.1 Sensor Signal Integrity . . . . . 60
6.1.2 Wiring and Connectivity . . . . . 60
6.1.3 Wearability and Fit . . . . . 60
**6.2 Software Verification . . . . . 61**
6.2.1 Signal Processing Verification . . . . . 61
6.2.2 Real-Time Classification Application . . . . . 62
**6.3 Chapter Summary . . . . . 62**

This chapter outlines the test plan and verification procedures used to ensure the functional correctness and reliability of the system's hardware and software components. Whereas the Results chapter (Chapter 7) focuses on quantifying the system's performance, this chapter details the steps taken to verify that each component was implemented correctly and operated as intended. The test plan was designed to be systematic, covering component-level validation before proceeding to full system integration tests.

## 6.1 Hardware Verification

Verification of the physical hardware focused on three key areas: the ability of the sensors to capture high-quality signals, the integrity of the electrical connections, and the overall wearability of the custom-designed armband.

### 6.1.1 Sensor Signal Integrity

The primary test for the sensor modules was to verify that they could acquire clear and distinguishable sEMG signals during muscle activation. This was tested both qualitatively and quantitatively.

- **Qualitative Test:** Raw sEMG signals from all eight channels were visualised in real-time using the OpenBCI GUI. Figure 4.8 displays an example of this. During a series of muscle contractions, the plots were visually inspected to confirm that signal amplitude increased significantly during a gesture and returned to a low-noise baseline during rest. This provided initial confirmation that the sensors were responding correctly to muscle activity.
- **Quantitative Test:** The SNR was calculated to objectively verify signal quality. The test procedure involved computing the ratio of the average signal power during an active movement to the average signal power during a relaxed rest state. A high SNR value confirms that the signal contains significantly more information than the baseline noise, indicating a successful acquisition. Detailed results of this testing is provided in Chapter 7. The code used for this calculation is presented in the project source files.



### 6.1.2 Wiring and Connectivity

To ensure a reliable electrical path from the electrode surface to the OpenBCI board, the connectivity of each assembled sensor module was physically verified. In total, 32 correctly assembled sensor modules were produced for use with the 3D-printed armbands produced.

- **Test Procedure:** A standard digital multimeter was used to measure the electrical resistance between the metal stud of the snap-on ECG electrode and the corresponding female pin-header that connects to the Cyton board.
- **Acceptance Criterion:** A reading close to zero Ohms was required to pass the test. All assembled sensor modules consistently showed a resistance of $0.1\,\Omega$, confirming a solid, low-resistance connection suitable for capturing microvolt-level bio-potentials.

### 6.1.3 Wearability and Fit

The usability of the armband was tested through subjective feedback from users to ensure it was comfortable, adjustable, and secure.

- **Test Procedure:** Multiple users with different forearm sizes wore the complete assembly, including the TPU armband, sensor modules, bias wrist band, and the Cyton board holder. They were asked to perform the full range of six gestures repeatedly.
- **Evaluation Criteria:** Users provided qualitative feedback on comfort during movement, the security of the Velcro fastening, and whether the sensors remained in firm contact with the skin without causing irritation. The design was considered successful as it was reported to be comfortable and adjustable, a key requirement for reliable signal acquisition. However, some users noted that achieving the optimal fit and sensor module placement for the best signal quality could be a tedious process of fine adjustments. Furthermore, initial testing revealed that the standard armband was not suitable for users with significantly larger or smaller forearms. To address this, scaled versions of the TPU band were created and printed. This was a practical solution, as the armbands are quick to print and consume very little material, which aligns with the project's low-cost and adaptable design goals.



## 6.2 Software Verification

Each stage of the software pipeline was independently verified to ensure its correctness before integration.

### 6.2.1 Signal Processing Verification

The correct implementation of the signal filters was crucial for noise removal. This was verified by analysing their effect on both the time and frequency domains of the signal.

- **Test Procedure:** A sample recording was passed through the implemented band-pass and notch filters. The output was then plotted against the original signal. The FFT of the signal was also computed before and after filtering.
- **Expected Outcome:** The time-domain plot was expected to show a reduction in baseline drift and noise. The FFT plot was used for objective verification, confirming that signal power was significantly attenuated at the 50 Hz power-line frequency and outside the 30–124 Hz passband, thus verifying the correct implementation of the filters. An example of filtering on a single sEMG channel is shown in Figure 6.1.

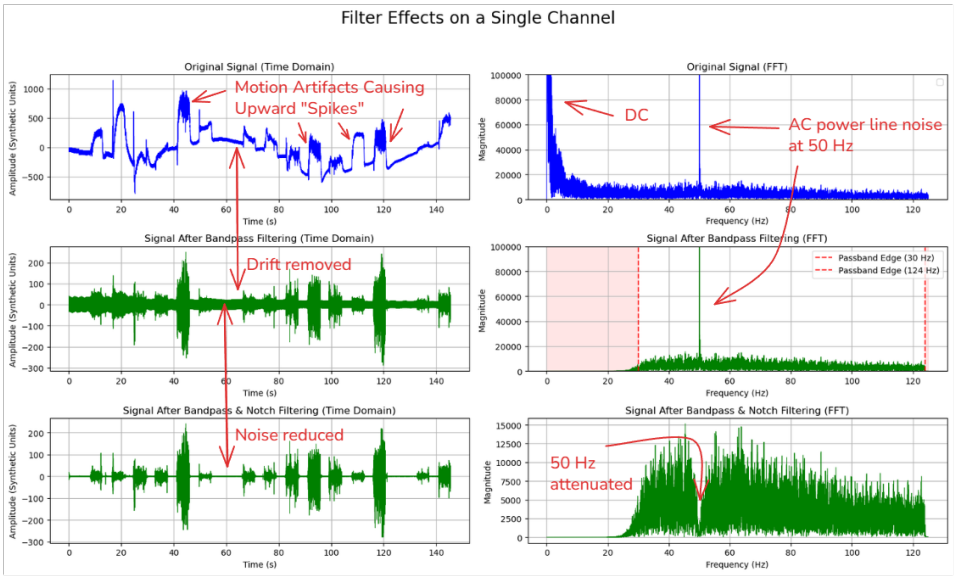



Figure 6.1: Plot of single sEMG channel showing effects of filtering.

### 6.2.2 Real-Time Classification Application

The final integrated application was tested qualitatively to verify its end-to-end functionality and user experience.

- **Test Procedure:** A user wore the device and performed each of the six gestures. The application's output was verified against the user's actions.
- **Verification Criteria:**
  1. **Correctness:** The GUI must display the correct predicted pose corresponding to the gesture being performed.
  2. **Stability:** The output prediction must be stable, without excessive flickering between classes. This directly tests the effectiveness of the majority-voting prediction buffer.
  3. **Configuration:** The model selection and arm selection (left/right channel flipping) features were tested to ensure they functioned as expected.

## 6.3 Chapter Summary

This chapter has described the comprehensive testing plan executed to verify the functional correctness of the entire sEMG classification system. Systematic tests were applied at each stage,

from hardware connectivity and signal integrity to software component validation and final system integration. The hardware was verified to be comfortable, reliable, and capable of acquiring high-quality signals. The software pipeline was validated step-by-step, including filter implementation, feature extraction correctness, and robust model training procedures. Finally, the real-time application was tested to confirm its stability and accuracy in a live setting. These verification steps ensure that the system is implemented correctly, providing a solid foundation for the quantitative performance analysis presented in Chapter 7.

6

# 7

# Results



## Contents

**7.1 Hardware Evaluation . . . . . 66**
7.1.1 3D-Printed sEMG Armband Signal Quality . . . . . 66
7.1.2 Motion Artifact Quantification . . . . . 67
**7.2 ML Model Results . . . . . 68**
7.2.1 Final MLP Offline Results . . . . . 68
7.2.2 Final MLP Real-Time Results . . . . . 70
**7.3 Comparison to Existing Solutions . . . . . 71**
**7.4 Computational Performance . . . . . 72**
**7.5 Chapter Summary . . . . . 73**

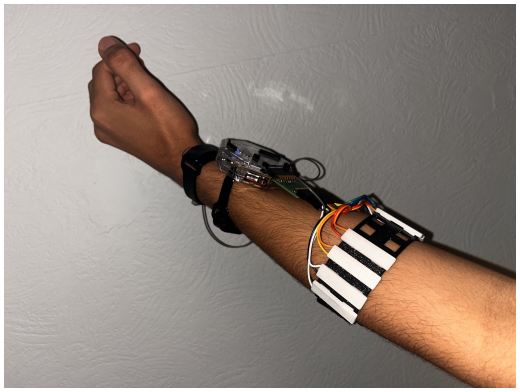
This chapter presents a comprehensive, multi-faceted evaluation of the developed sEMG gesture recognition system, validating its performance from hardware signal acquisition through to real-time software execution. The chapter begins with a detailed hardware evaluation, where the signal quality of the 3D-printed armband is quantified using SNR analysis and a characterisation of motion artifacts. Subsequently, the chapter assesses the classification performance of the final MLP model in both offline and real-time contexts, supported by standard metrics and a confusion matrix analysis. To contextualise the project's contributions, the system is then compared against existing academic and commercial solutions. Finally, the computational performance of the real-time pipeline is measured to verify its suitability for low-latency interactive applications. Collectively, these results provide a quantitative and qualitative validation of the system's design and effectiveness.

## 7.1 Hardware Evaluation

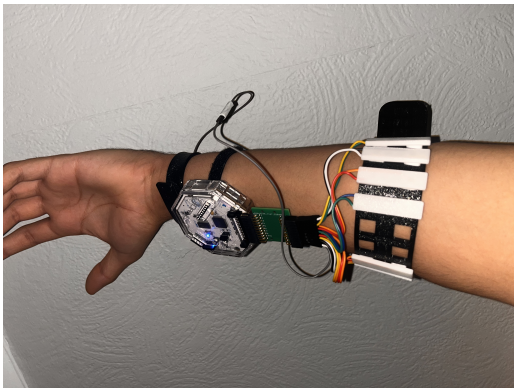
### 7.1.1 3D-Printed sEMG Armband Signal Quality

The sleeve was fitted to a user and adjusted carefully to fit the specific user's arm geometry, this was done to get the best possible electrode to skin contact. The sensor modules were placed in positions where forearm muscle contractions could be felt the most. In this case the eight sensors were split into two groups of four. One group was placed over the Flexor Carpi Radialis, Figure 7.1a. The second group was placed over the Flexor Carpi Ulnaris, Figure 7.1b. As the ML models never see the unfiltered signal, so the SNR was reported using the **filtered** signal with the processing discussed in Chapter 4.





(a) Sensor module placement perspective 1.



(b) Sensor module placement perspective 2.

Figure 7.1: Different perspectives of the sensor module placement on the forearm.

The SNR was calculated by taking the ratio of the **filtered** signal power during active movement to the noise power during rest. The filtering applied is visually illustrated in Figure 6.1.

$$\mathrm{SNR}_{\mathrm{dB}} = 10 \log_{10} \left( \frac{P_{\mathrm{signal}}}{P_{\mathrm{noise}}} \right) \tag{7.1}$$

Here, $P_{\mathrm{signal}}$ represents the average power of the sEMG signal during each specific movement, while $P_{\mathrm{noise}}$ corresponds to the average power recorded during the rest condition. The average power for any given segment was calculated as the mean square of the signal's amplitude values within that segment.

This method operates on the assumption that the noise power measured during the rest state is a valid approximation of the noise present during an active contraction. The reasoning behind

this assumption is that the SNR is calculated on the *filtered* signal. Since noise sources like motion artifacts are predominantly low-frequency ($< 20$ Hz, see Chapter 2), they are significantly attenuated by the 30 Hz high-pass filter applied during preprocessing. To ensure validity, care was taken during the experiment to allow the subject's arm to fully relax during rest periods, minimising any involuntary muscle activity. This was best achieved with the arm hanging down by the subject's side.

An attempt was made to further investigate the effect of movement on noise levels by having participants perform gestures while holding light weights. However, this approach was abandoned as it was impossible to isolate the noise from the muscle activation required to simply grip and hold the weight.

It was observed that the noise power remained relatively constant. As a result, the variation in SNR across channels and movements was primarily influenced by the relative strength of the muscle activation and the extent to which each movement engaged the muscle groups closest to the respective sEMG channels.



Table 7.1 summarises SNR values across channels and poses.

| Channel | Fist | Flexion | Extension | Radial | Ulnar |
|---|---|---|---|---|---|
| Channel 0 | 33.64 | 25.22 | 16.37 | 20.37 | 22.62 |
| Channel 1 | 16.21 | 10.60 | 6.38 | 3.61 | 16.03 |
| Channel 2 | 15.50 | 20.10 | 10.10 | 15.41 | 13.93 |
| Channel 3 | 7.17 | 20.97 | 4.57 | 8.33 | 7.19 |
| Channel 4 | 21.54 | 9.01 | 20.74 | 17.94 | 13.66 |
| Channel 5 | 20.19 | 7.27 | 16.44 | 14.57 | 9.78 |
| Channel 6 | 26.05 | 8.96 | 19.72 | 14.34 | 7.96 |
| Channel 7 | 22.85 | 4.47 | 21.90 | 15.23 | 8.93 |

Table 7.1: SNR (dB) for each EMG channel across hand movements.

### 7.1.2 Motion Artifact Quantification

To demonstrate the necessity of the signal preprocessing pipeline, the level of motion artifact present in the raw, unfiltered signal as a result of the hardware setup was quantified for each gesture. This was achieved by calculating the signal's amplitude range, which we assumed to be a proxy for the low-frequency drift caused by electrode and cable movement during a contraction. These motion artifacts can be visually observed in Figure 6.1.

The methodology involved programmatically identifying all continuous segments corresponding to each of the five active gestures using the dataset's class labels. The time window for each segment

was then expanded by one second on each side to ensure the analysis included the onset and offset of the movement, where artifacts are typically most pronounced. Within these expanded windows, the signal range (maximum value - minimum value) was computed for each of the eight channels. The final values, representing the average range in microvolts ($\mu$V) for each gesture on each channel, are presented in Table 7.2.

| **Pose** | **Ch 0** | **Ch 1** | **Ch 2** | **Ch 3** | **Ch 4** | **Ch 5** | **Ch 6** | **Ch 7** |
|---|---|---|---|---|---|---|---|---|
| Fist | 889.32 | 592.41 | 567.35 | 688.79 | 278.24 | 619.64 | 827.65 | 997.22 |
| Flexion | 938.63 | 540.09 | 562.68 | 1,804.36 | 533.11 | 927.83 | 1,252.27 | 691.81 |
| Extension | 633.57 | 237.96 | 524.03 | 632.57 | 2,319.90 | 2,898.93 | 1,131.23 | 1,224.04 |
| Radial | 328.51 | 202.38 | 414.61 | 517.70 | 261.53 | 588.70 | 506.76 | 430.46 |
| Ulnar | 435.72 | 643.53 | 381.45 | 920.32 | 427.79 | 796.12 | 563.21 | 571.21 |

Table 7.2: Average motion artifact drift per channel and gesture, quantified by the mean signal range ($\mu$V) on the raw signal.

The results show that dynamic movements involving significant wrist displacement, such as 'Extension' and 'Flexion', produce substantially larger signal ranges on certain channels compared to more isometric contractions like 'Fist'. For example, the 'Extension' gesture resulted in average drift values exceeding 2000 $\mu$V on channels 4 and 5. This analysis quantitatively confirms the presence of large, low-frequency motion artifacts in the raw signal. It underscores the critical importance of the 30 Hz high-pass filter, which is applied during preprocessing to remove this drift and provide a clean signal for the subsequent feature extraction and classification stages.

## 7.2 ML Model Results

The classification performance of the final MLP architecture is discussed in detail below in both an offline and real-time context. The results of training simpler models with many differing parameters is summarised in Tables B.1, B.2, B.3 in Appendix B.

### 7.2.1 Final MLP Offline Results

The final classifier was evaluated on a held-out test data set. The model demonstrates strong generalisation capabilities, with training and validation curves closely tracking each other and no indication of over-fitting as shown in Figure 7.2. The classification performance was further assessed using a confusion matrix (Figure 7.3) and summary metrics (Table 7.3). The architecture and training parameters of this model were discussed in Chapter 4 can be found in Tables 4.5 and

4.6.

The final model achieved an overall accuracy of **97.49%** in offline classification, with similarly high averages for precision, recall, and F1-score. This indicates that the classifier performs consistently across all classes. The slight variation in metrics across classes can be attributed to differences in the intensity and distinctiveness of muscle activations associated with each gesture, as well as sensor proximity to the most active muscle groups.

A summary of the meanings of each metric is given below:

- *Define:*
  - **TP** (True Positive): Correctly predicted positive class
  - **TN** (True Negative): Correctly predicted negative class
  - **FP** (False Positive): Incorrectly predicted as positive
  - **FN** (False Negative): Incorrectly predicted as negative



- **Accuracy**:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{7.2}$$

  Measures the proportion of correctly classified samples out of all predictions. It provides an overall sense of model performance.

- **Precision**:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{7.3}$$

  Indicates how many of the predicted positive instances were actually correct. High precision means few false positives.

- **Recall** (Sensitivity or True Positive Rate):

$$\text{Recall} = \frac{TP}{TP + FN} \tag{7.4}$$

  Measures the proportion of actual positive instances that were correctly identified. High recall means few false negatives.

- **F1-score**:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7.5}$$

  Combines precision and recall into a single metric by taking the harmonic mean.

The confusion matrix, Figure 7.3, supports these findings, showing minimal misclassification between movements. Overall, the results demonstrate that the trained model is highly effective at discriminating between the six target poses using sEMG signals, making it well-suited for real-time myoelectric control applications.

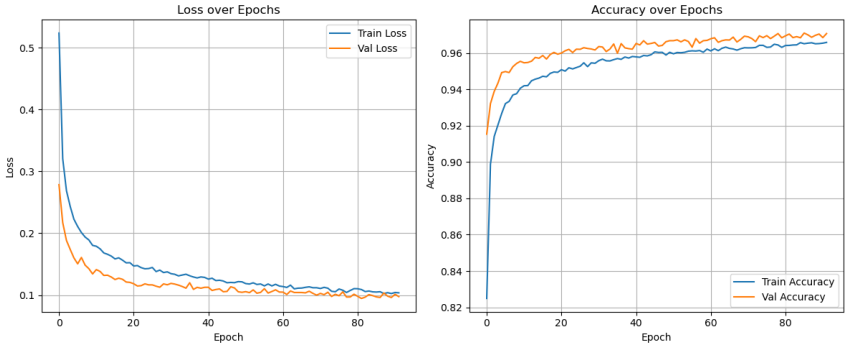



Figure 7.2: Training curves of final MLP architecture.



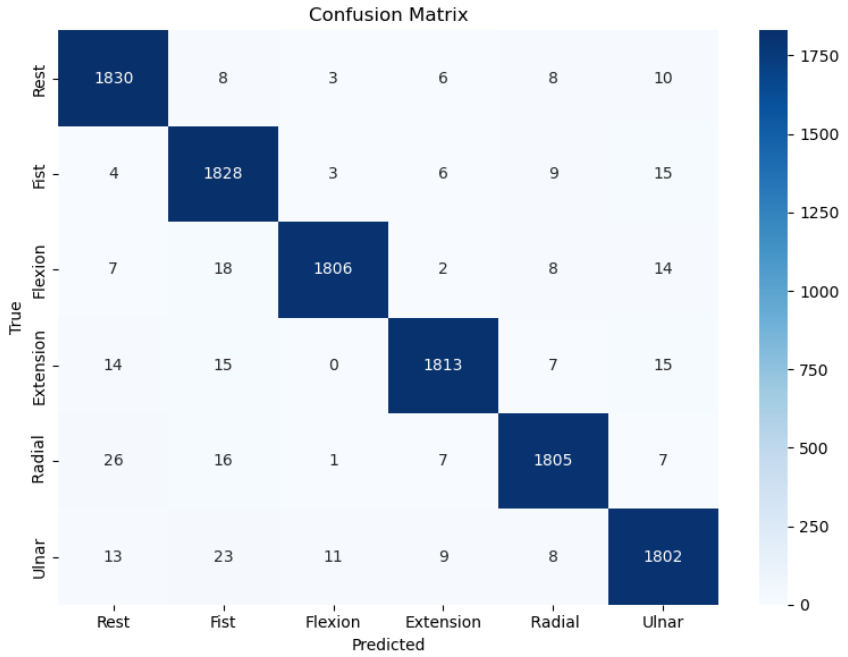



Figure 7.3: Confusion matrix of final MLP model on test data. Each row represents the true class and each column represents the predicted class, with diagonal values indicating correct classifications.

### 7.2.2 Final MLP Real-Time Results

The model achieved a validation accuracy of **97.83%**, demonstrating strong generalisation despite the constraint of causal filtering. Unlike earlier training iterations, no separate test set was used at this stage since the model architecture and hyperparameters had already been tuned during offline

| Class | Precision | Recall | F1-score | Count |
|---|---|---|---|---|
| Rest | 0.97 | 0.98 | 0.97 | 1865 |
| Fist | 0.96 | 0.98 | 0.97 | 1865 |
| Flexion | 0.99 | 0.97 | 0.98 | 1855 |
| Extension | 0.98 | 0.97 | 0.98 | 1864 |
| Radial | 0.98 | 0.97 | 0.97 | 1862 |
| Ulnar | 0.97 | 0.97 | 0.97 | 1866 |
| **Accuracy** | | **97.49%** | | |
| **Average** | 0.97 | 0.97 | 0.97 | 11177 |

Table 7.3: Classification metrics summary.

development. Instead, the entire dataset was leveraged for training, reserving 15% exclusively for validation to enable early stopping.

In the live classification application, a prediction buffer of size 8 was used. At each update cycle, the most frequent prediction within this buffer was displayed, effectively smoothing the output and reducing classification jitter. This strategy significantly enhanced the user experience by mitigating abrupt label switches and presenting stable predictions. Furthermore, the system exhibited low latency, with new classifications rendered promptly following each movement, confirming its suitability for responsive and interactive real-time use.



## 7.3 Comparison to Existing Solutions

To contextualise the contributions of this project, it is useful to compare the final system against the key commercial and academic solutions discussed in Chapter 2. The primary systems for comparison are the consumer-grade MYO armband [46] and the open-source academic "3DC" armband [51]. Table 7.4 provides a summary of their key features alongside those of the system developed in this work.

Table 7.4: Comparison of the developed system with existing sEMG armbands.

| Feature | This Project | MYO Armband | 3DC Armband |
|---|---|---|---|
| **Cost** | ~**£6** (Add-on only)<br>Requires OpenBCI Cyton (~£900) | ~$149<br>(Complete System) | ~$150<br>(Complete System) |
| **Sensor Modularity** | **Fully Modular**<br>(User-positioned sensors) | Fixed Position | Fixed Position |
| **Number of Channels** | 8 | 8 | 10 |
| **Sampling Rate (Hz)** | 250 | 200 | **1000** |
| **Open Source** | **Yes** (Hardware & Software) | No | **Yes** (Hardware & Software) |

The comparison highlights the unique position of this project. While the MYO and 3DC armbands are complete, self-contained systems, the hardware developed in this work serves as an

very low-cost **add-on** for an existing, powerful research platform. For researchers and students who already have access to an OpenBCI Cyton board, the cost to fabricate the custom wearable is negligible.

The most significant advantage of this system is its **full sensor modularity**. Unlike the fixed-position sensors of the other armbands, this design allows users to precisely place electrodes over specific muscle groups, adapting the hardware to their unique anatomy. This customisability is a key factor in achieving the high signal quality reported in this chapter and addresses a major limitation of ”one-size-fits-all” commercial designs.

While the sampling rate of 250 Hz is lower than that of the 3DC armband, it proved more than sufficient for achieving high classification accuracy with time-domain features. The key value proposition of this project is therefore not in creating the cheapest or highest-specification complete system, but in providing an open-source, highly adaptable, and virtually zero-cost peripheral that significantly enhances the capabilities of a widely used research tool.



## 7.4 Computational Performance

To ensure the system’s viability for real-time applications, the computational performance of the classification pipeline was benchmarked on a desktop PC [1]. A dedicated Python script was developed to measure the time required to process a single 128 ms window of data. The script loaded the final trained Keras model and its corresponding scikit-learn scaler. It then programmatically generated a synthetic data window of the correct size (32 samples $\times$ 8 channels) to simulate one unit of input from the live stream.

The benchmark was executed in a loop for 1,000 iterations to obtain a stable average, with an initial ”warm-up” run performed to exclude any one-time initialisation latency from TensorFlow. Within each iteration, the script measured the execution time of two distinct stages: first, the feature extraction process to compute the 84-dimension feature vector, and second, the model inference time required to generate a prediction from that vector. The average times for each stage, along with the total pipeline latency, are presented in Table 7.5. Given that the real-time application is configured to make a new classification every 64 ms, this measured total latency is well within the required processing budget, ensuring that the system can operate without computational bottlenecks.

[1] All benchmarks were ran on a system with a Ryzen 7 9800X3D Central Processing Unit (CPU), 64 GB RAM, running Ubuntu 24.04.2 LTS.

| Metric | Average Time (ms) |
|---|---|
| Feature Extraction Time | 2.6592 |
| Model Inference Time | 29.3083 |
| **Total Pipeline Time** | **31.9675** |

Table 7.5: Benchmark results for real-time classification pipeline.

## 7.5 Chapter Summary

The results presented in this chapter validate the effectiveness of the designed system for accurate, real-time hand gesture classification. The hardware evaluation confirmed that the custom 3D-printed armband delivers reliable signals with a high SNR, indicating excellent signal quality suitable for discriminating between movements. The final machine learning model was shown to be highly effective, achieving a test accuracy of **97.49%** in the offline evaluation, with the confusion matrix and per-class metrics confirming robust performance across all six gestures.



Furthermore, the system's suitability for live deployment was verified. A model trained with a real-time-compatible pipeline achieved a high validation accuracy of **97.83%**. The computational performance benchmark demonstrated that the total pipeline latency is approximately 32 ms, which is well within the 64 ms interval between classifications in the live application, meeting the system's real-time constraints. In summary, the system performs strongly across all evaluation criteria—from hardware signal acquisition to software performance and computational efficiency—demonstrating its potential for responsive and reliable myoelectric control.

7

# 8

# Conclusions

Contents


8.1 Evaluation . . . . . 75
8.2 Project Success and Achievements . . . . . 76
8.3 Limitations and Further Work . . . . . 76
8.4 Remarks on Design Decisions . . . . . 77




This project set out to design, build, and validate a complete system for pattern classification using sEMG as well as a highly adjustable, low-cost 3D-printed wearable for sEMG signal acquisition. This concluding chapter provides a critical evaluation of the project's outcomes against its initial objectives, reflects on the successes and limitations of the work, and discusses key design decisions and directions for future development.

## 8.1 Evaluation

A core part of this project was the successful fulfilment of the initial requirements laid out in Chapter 3. All functional and non-functional requirements were met, demonstrating the project's success in achieving its stated goals. The system successfully acquires multi-channel sEMG signals using the custom-designed, 3D-printable modular sleeve, which was proven to be both low-cost and comfortable. The software pipeline, from the data collection GUI to the final real-time classification application, was fully implemented and verified. Most importantly, the system achieved its primary goal of accurate gesture classification, with the final model yielding a test accuracy of 97.49%.

Compared to existing solutions, this project offers a distinct set of advantages. Unlike commer-

cial, closed-source products like the MYO armband, this system is fully open-source, replicable, and customisable. The modularity of the sensor placement allows for adaptation to individual users in a way that fixed-sensor devices cannot, directly contributing to the high signal quality reported in Chapter 7. For the student or researcher with access to OpenBCI hardware and a 3D printer, this proves as a great starting point for developing an easily producible and fully open-source system that offers a low-cost, modular peripheral for sEMG sensing—complementing the OpenBCI platform with a practical, research-ready solution for gesture recognition and bio-signal experimentation.

## 8.2 Project Success and Achievements

Overall, this project has been highly successful. An end-to-end system was developed from the ground up, encompassing hardware design, data collection, and machine learning software, culminating in a functional real-time application. The most important outcomes of this work are:



- **The development of a well performing, low-cost peripheral for the Cyton board:** The 3D-printed armband, costing just over £6 to produce, proved capable of acquiring high-quality signals suitable for robust classification. This represents a significant contribution to making sEMG technology more accessible.
- **The achievement of high classification accuracy:** The final offline model's accuracy of 97.49% demonstrates that a well-designed feature set combined with a relatively simple MLP architecture and diverse dataset can be highly effective for this task, rivalling the performance of more complex systems.
- **The implementation of a complete real-time system:** The final application successfully integrates signal acquisition, processing, and classification with low latency, proving the viability of the approach for interactive control applications.

## 8.3 Limitations and Further Work

While the project was successful, it is important to acknowledge its limitations, which in turn provide clear directions for future work.

- **Dataset Scope:** The training data used was collected from ten participants (although more data than this was collected, not all of it was usable). While diverse, a larger and more varied dataset would allow for the use of more expressive deep learning models capable of capturing complex spatio-temporal patterns in the signal. This could open the door to more advanced machine learning tasks beyond discrete classification, such as continuous wrist pose estimation or even fine-grained finger pose tracking, enabling richer and more nuanced control interfaces.
- **Gesture Set:** The system is currently limited to classifying six static hand and wrist poses. Future work could expand this vocabulary to include more complex, dynamic, or continuous pose estimation.
- **Hardware Dependence:** The system is designed as an add-on for the OpenBCI Cyton board. A significant next step would be to develop a custom, all-in-one embedded system that integrates microcontrollers, ADCs, and embedded ML capabilities into a single, stand-alone wearable device. The modular sensor system provides a unique foundation for this, with its detachable sensor modules enabling flexible placement, easy replacement, and scalability. Such a design could result in a more compact and convenient wearable that is both lower in cost compared to other 3D-printed sEMG armband solutions and more versatile in functionality, paving the way for on-device inference and real-time gesture recognition without the need for a tethered host system.



Future work should focus on addressing these limitations. An immediate goal would be to expand the dataset dramatically while maintaining the quality of data gathered. Following this, the system could be integrated with a target application, such as controlling a virtual prosthetic hand or a video game, to evaluate its real-world usability and performance. Further goals could be to explore deep learning methods for continuous pose estimation tasks rather than discrete classification tasks, or to explore designing an embedded system that does all processing and ML on device.

## 8.4 Remarks on Design Decisions

The journey of this project involved navigating a series of key design decisions and technical challenges. One of the most critical design choices was in the hardware, moving from an initial rigid, chain-link armband to the final flexible TPU design. This decision, driven by user feedback, prioritised attaining the highest level of adjustability and signal integrity over the novelty of the

first design and was fundamental to the project's success. On the software side, the decision to focus on a robust feature engineering process for an MLP, rather than immediately turning to more complex deep learning models, proved to be highly effective and pragmatic for the scale of the available dataset.

The most challenging aspect of the project was the integration of the hardware and software into a seamless, real-time system. Ensuring that the data stream and processing pipeline did not block the GUI for the user required thoughtful multi-threaded software architecture. This project provided invaluable experience in the complete lifecycle of an engineering project, from conceptual design and rapid prototyping to systematic testing and deployment. It underscored the importance of an iterative, user-centred design process and demonstrated the powerful results that can be achieved by thoughtfully combining custom hardware with intelligent software.

# 9 User Guide

Contents


**9.1 Hardware Setup and Assembly . . . . . 79**
9.1.1 3D Printing . . . . . 79
9.1.2 Assembly . . . . . 80
**9.2 Data Collection Protocol . . . . . 80**
**9.3 Training a Classification Model . . . . . 81**
**9.4 Using the Real-Time Application . . . . . 82**




This chapter serves as a practical guide for users who wish to replicate, use, or build upon the system developed in this project. **All project files, including the CAD models for 3D printing, the full source code for all Python applications, the collected datasets, and the machine learning notebooks, are publicly available in the project's GitHub repository**[1] The repository also contains video demonstrations illustrating the hardware assembly and software usage described in this guide.

## 9.1 Hardware Setup and Assembly

This section details the steps required to fabricate and assemble the physical armband.

### 9.1.1 3D Printing

The custom hardware components are designed to be printed on a standard 3D printer. The `.stl` files for all parts are available in the repository.

[1] The repository can be found at `https://github.com/LilOz/sEMG-Forearm-Classification`.

- **Armband, Cyton Strap, and Bias Band:** These components should be printed using a flexible filament, such as 95A TPU. A standard TPU print profile is sufficient.
- **Sensor Modules:** The eight sensor module housings should be printed using a rigid filament, such as PLA.

Scaled versions of the TPU bands are also provided in the repository to accommodate different forearm sizes.

### 9.1.2 Assembly

Once printed, the components must be assembled. This process is solder-free. Refer to the assembly video in the GitHub repository for more detailed instructions.

1. **Attach Wires to Sensor Modules:** For each sensor module, take two standard jumper wires with one end being a female pin. Feed the non pin end through the wire routes of the PLA housing. Once fed through, strip the end of the wire.
2. **Insert Electrodes:** Take a standard disposable ECG electrode, with the gel and adhesive removed, and press it firmly into the housing. The metal stud of the electrode will make secure contact with the stripped wire due to the tight tolerance of the snap-fit design, as detailed in Chapter 5. Repeat for all sensor modules.
3. **Apply Velcro:** Cut strips of heavy-duty Velcro. Attach the hook side to the end of the main TPU armband, the Cyton strap, and the bias band. Attach the corresponding soft side to the body of the bands to allow for an adjustable fit.
4. **Connect to Board:** Attach the sensor modules to the TPU armband. Connect the female ends of the jumper wires from the sensor modules to the differential input pins (N1P to N8P) on the OpenBCI Cyton board. Connect the bias electrode to the BIAS pin.

## 9.2 Data Collection Protocol

While various methods for data collection are possible, the recommended approach uses the custom `EMGPromptApp` application developed for this project, in conjunction with the official OpenBCI GUI. An example of this procedure is shown on the GitHub repository.

1. **Setup:** Launch both the OpenBCI GUI and the `EMGPromptApp.py` script. In the OpenBCI GUI, connect to the Cyton board and start a recording session.

2. **Data Acquisition:** In the `EMGPromptApp`, enter the name, gesture duration, and rest duration then click "Start". The application will guide the user through a randomised sequence of timed gestures and rest periods, as illustrated by the state machine in Figure 5.3. The app automatically saves a `.csv` file containing the gesture labels and their precise timestamps.

3. **Labelling:** After the session, place the recorded `.txt` file (from OpenBCI) and the `.csv` file (from `EMGPromptApp`) into the 'Output Files' directory. Run the provided labelling script (`label.py`), which will automatically merge the two files to produce a single, labelled dataset ready for processing. This process is described in detail in Chapter 5.

## 9.3 Training a Classification Model

The repository contains Jupyter notebooks that provide a step-by-step walk-through of the recommended model training pipeline. The key stages are:



- **Signal Preprocessing:** The first step is to combine then filter the raw, labelled data. The recommended offline approach uses a zero-phase band-pass filter (30–124 Hz) and a 50 Hz notch filter, as detailed in Chapter 5.

- **Feature Extraction:** The filtered signal should then be segmented into overlapping windows (128 ms duration, 75% overlap is recommended). The final feature set, comprising 7 time-domain features and 28 inter-channel correlations per window, should be extracted. The rationale for this feature set is discussed in Chapter 4.

- **Model Training:** An MLP classifier, with the architecture specified in Table 4.5, can then be trained on the extracted features. It is crucial to split the data into training, validation, and test sets to ensure robust evaluation. There is also another notebook with a differing preprocessing phase for a real-time model to be trained and saved for later use with the real-time application.

Once trained, the model (`model.h5`) and the feature scaler (`scaler.pkl`) should be saved for use in the real-time application.

## 9.4 Using the Real-Time Application

The live classification application provides real-time feedback using a pre-trained model.

1. **Setup:** Place your trained `model.h5` and `scaler.pkl` files into a subdirectory within the `models/` folder.
2. **Launch:** Run the `real_time_classification.py` script.
3. **Configuration:** The application will first prompt you to select the model you wish to load from the `models/` directory. It will then ask you to specify whether the armband is being worn on the left or right arm.
4. **Operation:** Once configured, the application will connect to the OpenBCI board and display a GUI showing the live, filtered sEMG signals and the currently predicted gesture. The implementation details of this application, including the multi-threaded architecture and prediction smoothing, are covered in Chapter 5.

# A

# Appendix: sEMG Feature Mathematical Definitions

- **Integrated EMG (IEMG)**:

$$\text{IEMG} = \sum_{i=1}^{N} |x_i|$$

The total magnitude of the EMG signal over the window.

- **Variance (VAR)**:

$$\text{VAR} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2$$

Measures signal power by quantifying its dispersion.

- **Root Mean Square (RMS)**:

$$\text{RMS} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} x_i^2}$$

Represents the effective magnitude of the signal.

- **Log RMS (lnRMS)**:

$$\text{lnRMS} = \log(\text{RMS} + \varepsilon)$$

Logarithmic form of RMS, with a small constant $\varepsilon$ to prevent $\log(0)$.

- **Kurtosis**:

$$\text{Kurtosis} = \frac{\frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^4}{\left( \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2 \right)^2} - 3$$

Measures the peakedness of the signal's distribution.

- **Skewness**:

$$\text{Skewness} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^3}{\left(\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2\right)^{3/2}}$$

Quantifies the asymmetry of the signal distribution.

- **Average Amplitude Change (AAC)**:

$$\text{AAC} = \frac{1}{N-1}\sum_{i=1}^{N-1}|x_{i+1} - x_i|$$

Average change in amplitude between adjacent points.

- **Mean Absolute Value (MAV)**:

$$\text{MAV} = \frac{1}{N}\sum_{i=1}^{N}|x_i|$$

Average of the absolute values in the signal.

- **Modified Mean Absolute Value 1 (MAV1)**:

$$\text{MAV1} = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}|x_i|, & \text{if } |x_i| > \alpha \cdot \max(|x|)\\ 0, & \text{otherwise}\end{cases}$$

A variant of MAV considering only large signal magnitudes.

- **Modified Mean Absolute Value 2 (MAV2)**:

$$\text{MAV2} = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}|x_i|, & \text{if } |x_i| < \alpha \cdot \max(|x|)\\ 0, & \text{otherwise}\end{cases}$$

Similar to MAV1 but includes small magnitude values.

- **Mean Absolute Value Slope (MAVS)**: Let the signal be divided into $S$ equal segments. Then,

$$\text{MAVS} = \frac{1}{S-1}\sum_{i=1}^{S-1}|\text{MAV}_i - \text{MAV}_{i+1}|$$

Measures change in MAV across signal segments.

- **Slope Sign Changes (SSC)**:

$$\text{SSC} = \sum_{i=2}^{N-1} \delta[(x_i - x_{i-1})(x_i - x_{i+1}) < 0]$$

  Counts the number of sign changes in the slope of the signal.

- **Simple Square Integral (SSI)**:

$$\text{SSI} = \sum_{i=1}^{N} x_i^2$$

  Energy of the signal over the window.

- **Absolute Temporal Moment (TM)**:

$$\text{TM}_p = \frac{1}{N} \sum_{i=1}^{N} |x_i|^p$$

  The $p^{\text{th}}$ order absolute moment of the signal.

- **Willison Amplitude (WAMP)**:

$$\text{WAMP} = \sum_{i=1}^{N-1} \delta(|x_{i+1} - x_i| \geq T)$$

  Counts how often the absolute difference between adjacent samples exceeds a threshold $T$.

- **Myopulse Percentage Rate (MYOP)**:

$$\text{MYOP} = \sum_{i=1}^{N} \delta(|x_i| > T)$$

  Percentage of samples whose magnitude exceeds threshold $T$.

- **Zero Crossings (ZC)**:

$$\text{ZC} = \sum_{i=1}^{N-1} \delta((x_i \cdot x_{i+1} < 0) \wedge |x_i - x_{i+1}| \geq T)$$

  Counts the number of times the signal crosses zero with a minimum difference threshold $T$.

Where $\delta(\cdot)$ is the indicator function, $N$ is the number of samples in the window, and $\bar{x}$ is the mean of the signal.

# B

# Appendix: Window Size and Overlap Grid Search Results

Table B.1: Classifier Performance for Window Duration (wd) = 0.512

| op | Classifier | Train Acc (%) | Validation Acc (%) |
|---|---|---|---|
| 0 | LDA | 84.41 | 83.51 |
| | Random Forest | 100.00 | 93.03 |
| | SVM | 94.60 | 89.43 |
| | KNN (k=5) | 94.99 | 93.78 |
| | KNN (k=10) | 93.22 | 92.28 |
| | Logistic Regression | 90.84 | 87.41 |
| | Ridge Classifier | 82.87 | 81.86 |
| | Perceptron | 82.87 | 82.01 |
| | SGD Classifier | 86.63 | 83.43 |
| | Naive Bayes | 63.87 | 66.34 |
| | Decision Tree | 100.00 | 82.83 |
| | **Average** | | **85.08** |
| 0.25 | LDA | 85.14 | 82.80 |
| | Random Forest | 100.00 | 94.32 |
| | SVM | 94.97 | 89.04 |
| | KNN (k=5) | 94.89 | 92.97 |
| | KNN (k=10) | 93.45 | 92.30 |
| | Logistic Regression | 90.73 | 87.63 |
| | Ridge Classifier | 83.57 | 80.94 |
| | Perceptron | 81.59 | 80.44 |
| | SGD Classifier | 87.76 | 85.27 |
| | Naive Bayes | 62.64 | 62.79 |
| | Decision Tree | 100.00 | 84.71 |
| | **Average** | | **84.84** |
| 0.5 | LDA | 84.93 | 84.15 |
| | Random Forest | 100.00 | 95.31 |
| | SVM | 94.22 | 89.81 |
| | KNN (k=5) | 95.93 | 94.19 |
| | KNN (k=10) | 94.57 | 93.89 |
| | Logistic Regression | 91.02 | 88.04 |
| | Ridge Classifier | 83.21 | 82.01 |
| | Perceptron | 82.55 | 81.11 |
| | SGD Classifier | 88.41 | 86.32 |
| | Naive Bayes | 65.45 | 65.37 |
| | Decision Tree | 100.00 | 87.63 |
| | **Average** | | **86.17** |
| 0.75 | LDA | 85.04 | 82.50 |
| | Random Forest | 100.00 | 96.29 |
| | SVM | 94.18 | 91.23 |
| | KNN (k=5) | 97.28 | 95.50 |
| | KNN (k=10) | 95.67 | 94.64 |
| | Logistic Regression | 90.79 | 88.17 |
| | Ridge Classifier | 83.43 | 81.82 |
| | Perceptron | 84.33 | 83.00 |
| | SGD Classifier | 89.60 | 87.46 |
| | Naive Bayes | 64.28 | 62.78 |
| | Decision Tree | 100.00 | 91.04 |
| | **Average** | | **86.77** |

Table B.2: Classifier Performance for Window Duration (wd) = 0.256

| op | Classifier | Train Acc (%) | Validation Acc (%) |
|---|---|---|---|
| 0 | LDA | 83.49 | 81.51 |
| | Random Forest | 100.00 | 92.99 |
| | SVM | 92.51 | 87.58 |
| | KNN (k=5) | 94.63 | 91.50 |
| | KNN (k=10) | 92.98 | 91.17 |
| | Logistic Regression | 89.00 | 85.22 |
| | Ridge Classifier | 81.90 | 80.35 |
| | Perceptron | 82.23 | 79.00 |
| | SGD Classifier | 86.50 | 83.98 |
| | Naive Bayes | 62.56 | 62.22 |
| | Decision Tree | 100.00 | 83.87 |
| | **Average** | | **83.58** |
| 0.25 | LDA | 82.82 | 82.21 |
| | Random Forest | 100.00 | 93.89 |
| | SVM | 92.07 | 88.57 |
| | KNN (k=5) | 95.31 | 93.48 |
| | KNN (k=10) | 94.06 | 92.94 |
| | Logistic Regression | 88.73 | 86.42 |
| | Ridge Classifier | 81.53 | 80.52 |
| | Perceptron | 80.65 | 79.00 |
| | SGD Classifier | 85.81 | 84.10 |
| | Naive Bayes | 62.77 | 61.87 |
| | Decision Tree | 100.00 | 86.20 |
| | **Average** | | **84.47** |
| 0.5 | LDA | 82.77 | 81.65 |
| | Random Forest | 100.00 | 95.09 |
| | SVM | 91.69 | 89.33 |
| | KNN (k=5) | 96.28 | 94.40 |
| | KNN (k=10) | 94.87 | 93.68 |
| | Logistic Regression | 88.15 | 86.79 |
| | Ridge Classifier | 81.32 | 80.48 |
| | Perceptron | 80.24 | 79.52 |
| | SGD Classifier | 86.14 | 85.08 |
| | Naive Bayes | 64.14 | 64.00 |
| | Decision Tree | 100.00 | 87.59 |
| | **Average** | | **85.24** |
| 0.75 | LDA | 82.55 | 82.34 |
| | Random Forest | 100.00 | 96.48 |
| | SVM | 91.23 | 89.90 |
| | KNN (k=5) | 97.70 | 95.78 |
| | KNN (k=10) | 96.26 | 94.96 |
| | Logistic Regression | 88.12 | 87.20 |
| | Ridge Classifier | 81.28 | 80.56 |
| | Perceptron | 80.11 | 79.33 |
| | SGD Classifier | 86.43 | 85.83 |
| | Naive Bayes | 62.66 | 62.32 |
| | Decision Tree | 100.00 | 90.97 |
| | **Average** | | **85.97** |

Table B.3: Classifier Performance for Window Duration (wd) = 0.128

| op | Classifier | Train Acc (%) | Validation Acc (%) |
|---|---|---|---|
| 0 | LDA | 79.92 | 80.48 |
| | Random Forest | 100.00 | 93.15 |
| | SVM | 87.98 | 86.74 |
| | KNN (k=5) | 92.94 | 89.89 |
| | KNN (k=10) | 91.69 | 90.07 |
| | Logistic Regression | 85.37 | 84.54 |
| | Ridge Classifier | 78.52 | 79.21 |
| | Perceptron | 75.66 | 75.61 |
| | SGD Classifier | 81.40 | 81.50 |
| | Naive Bayes | 59.76 | 59.01 |
| | Decision Tree | 100.00 | 83.31 |
| | **Average** | | **82.14** |
| 0.25 | LDA | 79.85 | 80.45 |
| | Random Forest | 100.00 | 93.63 |
| | SVM | 87.75 | 87.27 |
| | KNN (k=5) | 93.89 | 91.14 |
| | KNN (k=10) | 92.43 | 90.84 |
| | Logistic Regression | 84.90 | 85.32 |
| | Ridge Classifier | 78.50 | 78.65 |
| | Perceptron | 77.09 | 77.04 |
| | SGD Classifier | 82.96 | 83.16 |
| | Naive Bayes | 62.79 | 62.46 |
| | Decision Tree | 100.00 | 85.15 |
| | **Average** | | **83.19** |
| 0.5 | LDA | 80.19 | 80.08 |
| | Random Forest | 100.00 | 94.70 |
| | SVM | 88.19 | 87.60 |
| | KNN (k=5) | 95.27 | 92.51 |
| | KNN (k=10) | 93.59 | 92.22 |
| | Logistic Regression | 85.11 | 85.10 |
| | Ridge Classifier | 78.70 | 78.82 |
| | Perceptron | 77.54 | 77.50 |
| | SGD Classifier | 82.58 | 82.08 |
| | Naive Bayes | 59.43 | 59.22 |
| | Decision Tree | 100.00 | 87.37 |
| | **Average** | | **83.38** |
| 0.75 | LDA | 80.16 | 80.15 |
| | Random Forest | 100.00 | 96.18 |
| | SVM | 87.94 | 87.50 |
| | KNN (k=5) | 97.65 | 95.50 |
| | KNN (k=10) | 95.56 | 94.12 |
| | Logistic Regression | 85.29 | 84.86 |
| | Ridge Classifier | 78.85 | 78.40 |
| | Perceptron | 75.55 | 75.11 |
| | SGD Classifier | 83.38 | 83.09 |
| | Naive Bayes | 61.02 | 61.08 |
| | Decision Tree | 100.00 | 90.13 |
| | **Average** | | **84.19** |

# Bibliography


[1] G. Robertson, *Research Methods in Biomechanics*. Human Kinetics Publishers, 2013.

[2] *Cyton biosensing board (8-channels)*, `https://shop.openbci.com/products/cyton-biosensing-board-8-channel`, Accessed: 17/01/2025.

[3] `https://shop.openbci.com/collections/frontpage`, Accessed: 17/01/2025.

[4] Meta, *Introducing orion, our first true augmented reality glasses*, `https://about.fb.com/news/2024/09/introducing-orion-our-first-true-augmented-reality-glasses/`, Accessed: 28/10/2024.

[5] A. Phinyomark, P. Phukpattaranont, and C. Limsakul, "A review of control methods for electric power wheelchairs based on electromyography signals with special emphasis on pattern recognition," *IETE Technical Review*, vol. 28, no. 4, pp. 316–326, 2011.

[6] R. Merletti and G. Cerone, "Tutorial. surface emg detection, conditioning and pre-processing: Best practices," *Journal of Electromyography and Kinesiology*, vol. 54, p. 102 440, 2020.

[7] J. R. Daube and D. I. Rubin, "Needle electromyography," *Muscle & Nerve: Official Journal of the American Association of Electrodiagnostic Medicine*, vol. 39, no. 2, pp. 244–270, 2009.

[8] S. K. Berkaya, A. K. Uysal, E. S. Gunal, S. Ergin, S. Gunal, and M. B. Gulmezoglu, "A survey on ecg analysis," *Biomedical Signal Processing and Control*, vol. 43, pp. 216–235, 2018.

[9] M. d. Olmo and R. Domingo, "Emg characterization and processing in production engineering," *Materials*, vol. 13, no. 24, p. 5815, 2020.

[10] E. Farago, D. MacIsaac, M. Suk, and A. D. Chan, "A review of techniques for surface electromyography signal quality analysis," *IEEE Reviews in Biomedical Engineering*, vol. 16, pp. 472–486, 2022.

[11] C. Nordander, J. Willner, G.-Å. Hansson, *et al.*, "Influence of the subcutaneous fat layer, as measured by ultrasound, skinfold calipers and bmi, on the emg amplitude," *European journal of applied physiology*, vol. 89, pp. 514–519, 2003.

[12] G. Drost, D. F. Stegeman, B. G. van Engelen, and M. J. Zwarts, "Clinical applications of high-density surface emg: A systematic review," *Journal of Electromyography and Kinesiology*, vol. 16, no. 6, pp. 586–602, 2006.

[13] C. Lossin and A. L. George Jr, "Myotonia congenita," *Advances in genetics*, vol. 63, pp. 25–55, 2008.

[14] P. A. Pincheira, E. Martinez-Valdes, R. Guzman-Venegas, *et al.*, "Regional changes in muscle activity do not underlie the repeated bout effect in the human gastrocnemius muscle," *Scandinavian journal of medicine & science in sports*, vol. 31, no. 4, pp. 799–812, 2021.

[15] T. E. Solstad, V. Andersen, M. Shaw, E. M. Hoel, A. Vonheim, and A. H. Saeterbakken, "A comparison of muscle activation between barbell bench press and dumbbell flyes in resistance-trained males," *Journal of sports science & medicine*, vol. 19, no. 4, p. 645, 2020.

[16] M. Zecca, S. Micera, M. C. Carrozza, and P. Dario, "Control of multifunctional prosthetic hands by processing the electromyographic signal," *Critical Reviews™ in Biomedical Engineering*, vol. 30, no. 4-6, 2002.

[17] D. Farina, N. Jiang, H. Rehbaum, *et al.*, "The extraction of neural information from the surface emg for the control of upper-limb prostheses: Emerging avenues and challenges," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 4, pp. 797–809, 2014.

[18] P. Visconti, F. Gaetani, G. A. Zappatore, and P. Primiceri, "Technical features and functionalities of myo armband: An overview on related literature and advanced applications of myoelectric armbands mainly focused on arm prostheses," *International Journal on Smart Sensing and Intelligent Systems*, vol. 11, no. 1, pp. 1–25, 2018.

[19] S. Marcos-Antón, M. D. Gor-García-Fogeda, and R. Cano-de-la-Cuerda, "An semg-controlled forearm bracelet for assessing and training manual dexterity in rehabilitation: A systematic review," *Journal of Clinical Medicine*, vol. 11, no. 11, p. 3119, 2022.

[20] C. Chestek and P. Cederna, *Mind control prosthesis*, `https://spotlight.engin.umich.edu/mind-control-prosthesis/`, Accessed: 12/11/2024.

[21] M. G. Urbanchek, Z. Baghmanli, J. D. Moon, K. B. Sugg, N. B. Langhals, and P. S. Cederna, "Quantification of regenerative peripheral nerve interface signal transmission," *Plastic and reconstructive surgery*, vol. 130, no. 5S-1, pp. 55–56, 2012.

[22] W.-T. Shi, Z.-J. Lyu, S.-T. Tang, T.-L. Chia, and C.-Y. Yang, "A bionic hand controlled by hand gesture recognition based on surface emg signals: A preliminary study," *Biocybernetics and Biomedical Engineering*, vol. 38, no. 1, pp. 126–135, 2018.

[23] A. Phinyomark, F. Quaine, S. Charbonnier, C. Serviere, F. Tarpin-Bernard, and Y. Laurillau, "Emg feature evaluation for improving myoelectric pattern recognition robustness," *Expert Systems with applications*, vol. 40, no. 12, pp. 4832–4840, 2013.

[24] Z. Abass, W. Meng, S. Q. Xie, and Z. Zhang, "A robust, practical upper limb electromyography interface using dry 3d printed electrodes," in *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, IEEE, 2019, pp. 453–458.

[25] G. Purushothaman and K. Ray, "Emg based man–machine interaction—a pattern recognition research platform," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 864–870, 2014.

[26] A. Phinyomark, R. N. Khushaba, and E. Scheme, "Feature extraction and selection for myoelectric control based on wearable emg sensors," *Sensors*, vol. 18, no. 5, p. 1615, 2018.

[27] P. Kaufmann, K. Englehart, and M. Platzner, "Fluctuating emg signals: Investigating long-term effects of pattern matching algorithms," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, IEEE, 2010, pp. 6357–6360.

[28] J. S. Richman, D. E. Lake, and J. R. Moorman, "Sample entropy," in *Methods in enzymology*, vol. 384, Elsevier, 2004, pp. 172–184.

[29] X. Zhang and P. Zhou, "Sample entropy analysis of surface emg for improved muscle activity onset detection against spurious background spikes," *Journal of Electromyography and Kinesiology*, vol. 22, no. 6, pp. 901–907, 2012.

[30] S. A. Raurale, J. McAllister, and J. M. del Rincon, "Real-time embedded emg signal analysis for wrist-hand pose identification," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2713–2723, 2020.

[31] M. Tavakoli, C. Benussi, P. A. Lopes, L. B. Osorio, and A. T. de Almeida, "Robust hand gesture recognition with a double channel surface emg wearable armband and svm classifier," *Biomedical Signal Processing and Control*, vol. 46, pp. 121–130, 2018.

[32] C.-C. M. Yeh, X. Dai, H. Chen, *et al.*, "Toward a foundation model for time series data," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 4400–4404.

[33] N. Parajuli, N. Sreenivasan, P. Bifulco, *et al.*, "Real-time emg based pattern recognition control for hand prostheses: A review on existing methods, challenges and future implementation," *Sensors*, vol. 19, no. 20, p. 4596, 2019.

[34] A. J. Young, L. H. Smith, E. J. Rouse, and L. J. Hargrove, "Classification of simultaneous movements using surface emg pattern recognition," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 5, pp. 1250–1258, 2012.

[35] T.-W. Lee, "Independent component analysis," in *Independent component analysis: Theory and applications*, Springer, 1998, pp. 27–66.

[36] Y. Wang, S. Gao, and X. Gao, "Common spatial pattern method for channel selelction in motor imagery based brain-computer interface," in *2005 IEEE engineering in medicine and biology 27th annual conference*, IEEE, 2006, pp. 5392–5395.

[37] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[38] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electronic Markets*, vol. 31, no. 3, pp. 685–695, 2021.

[39] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[40] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Transactions on Circuits and Systems*, vol. 8, no. 7, pp. 579–588, 2009.

[41] U. Côté-Allard, C. L. Fall, A. Drouin, *et al.*, "Deep learning for electromyographic hand gesture signal classification using transfer learning," *IEEE transactions on neural systems and rehabilitation engineering*, vol. 27, no. 4, pp. 760–771, 2019.

[42] P. Vuorimaa, "Fuzzy self-organizing map," *Fuzzy sets and systems*, vol. 66, no. 2, pp. 223–231, 1994.

[43] T. Bayes, "Naive bayes classifier," *Article Sources and Contributors*, pp. 1–9, 1968.

[44] C. Seyidbayli, F. Salhi, and E. Akdogan, "Comparison of machine learning algorithms for emg signal classification," *Periodicals of Engineering and Natural Sciences (PEN)*, vol. 8, no. 2, pp. 1165–1176, 2020.

[45] S. R. Eddy, "What is a hidden markov model?" *Nature biotechnology*, vol. 22, no. 10, pp. 1315–1316, 2004.

[46] S. Rawat, S. Vats, and P. Kumar, "Evaluating and exploring the myo armband," in *2016 International Conference System Modeling & Advancement in Research Trends (SMART)*, IEEE, 2016, pp. 115–120.

[47] Z. Li, *Signal processing of multichannel knitted emg sensors*, 2024.

[48] B. Stern and L. Fried, *Myo armband teardown*, https://learn.adafruit.com/myo-armband-teardown/inside-myo, Accessed: 17/01/2025.

[49] Meta, *A look at our surface emg research focused on equity and accessibility*, https://www.meta.com/en-gb/blog/quest/surface-emg-wristband-electromyography-human-computer-interaction-hci/?srsltid=AfmBOopw7dFjrF6NA23-gVByfnokPfG3tXP2EP2C70nP1Eb8HPegnscV, Accessed: 28/10/2024.

[50] Meta, *Advancing neuromotor interfaces by open sourcing surface electromyography (semg) datasets for pose estimation and surface typing*, https://ai.meta.com/blog/open-sourcing-surface-electromyography-datasets-neurips-2024/, Accessed: 28/10/2024.

[51] U. Côté-Allard, G. Gagnon-Turcotte, F. Laviolette, and B. Gosselin, "A low-cost, wireless, 3-d-printed custom armband for semg hand gesture recognition," *Sensors*, vol. 19, no. 12, p. 2811, 2019.

[52] *Ganglion board (4-channels)*, https://shop.openbci.com/products/ganglion-board, Accessed: 17/01/2025.

[53] *3d-printable ultracortex headset*, https://github.com/OpenBCI/Ultracortex, Accessed: 17/01/2025.

[54] E. Por, M. van Kooten, and V. Sarkovic, "Nyquist–shannon sampling theorem," *Leiden University*, vol. 1, no. 1, pp. 1–2, 2019.

[55] C. Larivière, A. Delisle, and A. Plamondon, "The effect of sampling frequency on emg measures of occupational mechanical exposure," *Journal of Electromyography and Kinesiology*, vol. 15, no. 2, pp. 200–209, 2005.

[56] https://www.wessex-medical.com/shop/ecg-recorders/ecg-electrodes/skintact-monitoring-electrodes/, Accessed: 17/01/2025.

[57] *Bambulab p1s 3d printer*, https://bambulab.com/en-us/p1, Accessed: 17/01/2025.

[58] *Autodesk fusion 360*, https://www.autodesk.com/uk/products/fusion-360/overview, Accessed: 17/01/2025.

[59] https://www.ti.com/product/ADS1299, Accessed: 17/01/2025.

[60] S. Wager, S. Wang, and P. S. Liang, "Dropout training as adaptive regularization," *Advances in neural information processing systems*, vol. 26, 2013.

[61] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[62] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `https://www.tensorflow.org/`.

[63] F. Chollet *et al.* "Keras." (2015), [Online]. Available: `https://github.com/fchollet/keras`.