Imperial College
London

World Wide Mind: Real-time EEG decoding across 100 brains

Dissertation, MSc Applied Machine Learning

James Teversham, CID 01997171

2 September 2021

Prepared for Prof Tim Constandinou

Department of Electrical and Electronic Engineering, Imperial College London

Statement of Originality

I, James Teversham, hereby certify that:

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signed

2 September 2021

Foreword and Acknowledgements

This project has been a true test of what I believe Engineering is - or at least should be - about; a collection of tools that one acquires and develops to solve challenging inter-disciplinary problems in possibly foreign domains. It has thoroughly tested my technical and problem solving skills across several subject areas, including mathematics and statistics, signal processing, electronics, embedded systems programming, web programming and general patience. In addition, through this project, I have learnt about basic neurophysiology and associated neurotechnologies - fields which I was close to clueless about coming in to it. The brain truly is one of nature's most incredible and fascinating machines and this project has inspired me to try understand my own one a little more.

My most sincere vote of gratitude goes to the members of the Imperial NGNI Lab. Whether I was in Cape Town, Egypt or London, you managed to meet with me nearly every week since project inception to hear of my progress and more commonly, my trials and tribulations. Your advice and patient guidance was indispensable to me and the success of this project. I am most thankful for your time and wish you every success.

Finally, I thank my parents and family back home in Cape Town for their endless interest and support in my endeavours. Life in COVID has been challenging to say the least and I am extremely grateful for your help and encouragement. I hope that I have done you proud.

Abstract

The cost of existing brain-computer interface (BCI) technologies makes them inaccessible to the general public and largely prohibits their use on a mass scale. This project details the development of a novel, ultra low-cost BCI prototype that attempts to start changing this. This prototype, and its future extensions, is hoped to facilitate public engagement and education in the domain of EEG sensing and other neurotechnologies, as well as improving advocacy an encouraging further research funding.

In particular, this study presents the development of real time decoding and communication of raw EEG signals acquired from an ultra low-cost proprietary EEG-based BCI device developed by the Next Generation Neural Interfaces (NGNI) Lab at Imperial College. It forms part of a broader project being coordinated by the NGNI Lab that will be presented in an exhibition hosted by the Royal Society. The BCI prototype developed in this project will be worn by up to 100 different audience members at this exhibit and will serve to decode basic EEG signals concurrently and in real time in order to facilitate collaborative control in a multi-player game using only mental control.

This project was designed under very challenging constraints, including: a budget of around 20 GBP (compared to the price of several hundred GBP for a typical BCI device), availability of only a single EEG channel and the limitation of dry surface electrodes. The steady state visual evoked potential (SSVEP) paradigm is used as the BCI control signal of choice owing to its ease of implementation, relatively high achievable SNR and the fact that it does not require user training. Based on their wide-spread used in the field of EEG signal processing, decoding algorithms based on canonical correlation analysis (CCA) such as Multi-set CCA (MsetCCA) and Generalised CCA (GCCA) are explored most closely. An MQTT client for publishing decoded data to a cloud service using Amazon Web Services (AWS) IoT Core is also implemented in firmware. Towards the objective of creating a *fully mobile* BCI, all operations necessary for signal acquisition, processing, decoding and communication are capable of running on the device itself. This functionality is fully implemented in the firmware of the electronic hardware provided by the NGNI Lab based on the ESP32 SoC by Espressif Systems. All firmware is implemented in Micropython, the cross-architecture Python compiler and runtime for microcontrollers that runs on bare-metal.

Subject to performing a short calibration sequence each time the BCI is put on, optimal results show that decoding accuracy of $95.56 \pm 3.74\%$ with an information transfer rate (ITR) of 102 bits/min ($p = 4$ calibration trials of $T = 0.75$s each) can be achieved by the MsetCCA algorithm. With more modest calibration requirements ($p = 2$ calibration trials of $T = 1$s each), accuracy of $80.56 \pm 4.46\%$ with an ITR of 40 bits/min can be achieved using the same algorithm. All decoding computation occurs on-device in real time.

Although the fact that data was only collected from a single test subject in this study is a significant limitation, the prototype produced nonetheless presents a very encouraging proof-of-concept that warrants further investigation and more stringent testing. Importantly, it has been developed with exclusively open source tools and all source code is freely available here.

Contents

List of Figures

List of Tables

List of Source Code Listings

Chapter 1

Introduction

1.1 Background and Motivation

A brain-computer interface (BCI), also known as brain-machine interface (BMI), is a system that allows computers or other external devices to be controlled by electroencephalographic (EEG) activity alone [12]. The most common application of BCIs is to provide a means of communication to people who have lost the means to do so in a normal capacity, typically due to injury, trauma or preexisting impairments of the neuromuscular system. Particularly in the case of fully paralysed or 'locked in' patients suffering disorders such as amyotrophic lateral sclerosis (ALS), strokes or spinal cord injuries, advances in BCI technology pose a promising means of restoring the ability to communicate with the outside world [4]. This is made possible by measuring and characterising (decoding) brain activity in order to translate raw EEG signals into external control commands without any neuromuscular interaction. External control targets may be in the form of computer-based spellers, speech synthesizers or neural prostheses [12].

In some cases of severe impairment, invasive EEG treatment is required which involves implantation of electrode arrays on the surface of the brain. However, non-invasive EEG is far more prevalent as it only requires surface electrodes placed on the scalp of a subject and is thus safer, more convenient and far more accessible [27]. BCIs using this technology have found applications in smart home appliances, disability assistance and human-computer interaction [31]. For example, a patient bound to their bed due to neuromuscular impairments could interact with a BCI to adjust lighting, heating, television or other appliances.

A challenge with existing BCI technologies is their cost which, in many cases, prohibits their use on a mass scale. In this project, a novel, ultra low-cost[1] EEG-based BCI system will be created in order to investigate the viability of acquiring and decoding EEG activity from an audience of 100 people simultaneously.

Importantly, the success of this project would provide great opportunities for public engagement and education in the domain of EEG sensing and other neurotechnologies. For example, it is conceivable that a future version of this prototype could be used in summer schools or other educational activities to give young learners hands-on experience with this otherwise expensive and largely inaccessible technology. More widespread public demonstrations such as these will increase and improve advocacy for this technology which may, at present, be somewhat stigmatised in some communities due to its opaqueness and potentially, due to the negative association with mind control or other dramatised pop culture analogues.

Increased public interest and accessibility, in turn, will likely encourage further funding for this fascinating field of research. Furthermore, while the initial demonstration of this project will be performed on healthy individuals, it could provide great promise for facilitating large scale, inclusive interaction among people with neuromuscular impairments such as those mentioned above.

1.2 Objectives of the Study

The core focus of this study is to develop real time decoding and communication of raw analogue EEG signals acquired from a proprietary EEG hardware device developed by the Next Generation Neural Interfaces (NGNI) Lab at Imperial College London. These devices will be worn by up to 100 people simultaneously and will be used to facilitate collaborative control in a multiplayer game[2]. Consequently, decoding, communication and visualisation needs to happen concurrently and in real time between devices.

Specific objectives of this project are outlined below:

[1] In the region of 10 to 20 times less expensive than the cheapest comparable commercial model

[2] not included in the scope of this project

1. Review the literature pertaining to EEG signal acquisition and decoding. Perform a brief survey of common EEG paradigms and decide on a particular paradigm for further investigation based on a clear rationale.
2. Review decoding techniques and algorithms for the selected EEG paradigm. Record the most effective algorithms as specified in the literature, as well as their analytical derivations in preparation for implementation.
3. Scope firmware most suitable for the microcontroller used in the EEG hardware supplied by the NGNI Lab.
4. Design and implement embedded software required to sample EEG signals and perform online decoding thereof using previously identified decoding algorithms.
5. Determine the optimal network protocol and structure to communicate decoded data and any other pertinent device-specific information to a cloud service.
6. Devise testing mechanisms to verify the validity of obtained results.
7. Critically evaluate the performance of the system with reference to the objectives of this study
8. Provide a discussion of results obtained and provide reasons for any shortcomings or anomalies. Provide suggestions for further improvement.

1.2.1 Research questions to be investigated

This project aims to address the following core research questions:

- Can an ultra low-cost (in the region of £20) device be used to decode EEG signals of interest in real time? If so, to what degree of accuracy and under what limitations?
- If so, can such a device perform all computation necessary for signal decoding on-device (without the need to outsource computation to the cloud or end client, for example)?

1.2.2 Significance of this work

If able to fulfil its aforementioned objectives, this device would serve as a significant proof-of-concept system in the area of EEG signal analysis. Importantly, as specified by the constraints below, this device is to be built using low cost, easily accessible components. Commercial EEG devices, as explored in Section 2.5, are typically priced from several hundred GBP and upwards. Very few, if any, devices are available for less than 100 GBP. Therefore, while likely slightly less capable than the aforementioned commercial counterparts, the device investigated in this project could pose an interesting low-cost option for large scale EEG data acquisition, testing and experimentation. Furthermore, as alluded to in the motivation in Section 1.1, it could become a valuable pedagogical tool for engaging and educating young learners about basic neurophysiology and neurotechnology; fields that would otherwise be difficult to access below a tertiary education level.

1.3 Scope and Constraints

The task of using this device at scale in a public setting imposes some unique practical constraints. These are outlined below:

- production at scale: 100+ replicas of the BCI device need to be produced.
- very tight budget of $\approx$ £20 per device.
- hardware must be self-contained and so all computation should happen on the hardware if possible.
- there is to be minimal calibration required from the user. If any, it should be done automatically and should not detract significantly from the overall user experience.
- real time decoding and feedback of signals (or as close to real time as possible).
- device should be non-invasive and cannot use 'wet' electrodes that would hamper the user experience.

The EEG hardware provided by the NGNI Lab is based on the low-cost Espressif ESP32 SoC based on the Tensilica Xtensa LX6 MCU. All firmware must be developed on this device.

Finally, every effort must be taken to use exclusively **open source** software, tools and other resources in developing this system. This will create a platform most suitable for the education space as it will afford the ability to easily modify, extend or deconstruct the system as a result of having accessible resources;

both on a physical hardware/firmware level, as well as in the form of vast open source communities and documentation.

1.3.1 Constraint implications

The constraints listed above have substantial implications on my project. One of the largest constraints is the budget of £20; entry level consumer-grade BCIs (brain computer interfaces) typically cost in the region of several hundred USD. As a result, the device in this project will only have **two active electrodes** measured in a differential configuration to yield only a **single active channel**. Although most BCIs have many more active channels, many studies have shown that a viable decoding system can be achieved with only a limited number of channels [10], some with even as few as 2 [32].

The prohibition of wet electrodes is also significant. Wet electrodes are known to significantly improve the quality[3] of the electrical connection to the scalp which would invariably improve SNR.

1.4 Plan of Development

This report begins with a review of the basics of neurophysiology and how electrical signals are used by the human brain. An overview of the mechanics of electroencephalography (EEG) is presented, followed by some of the popular EEG paradigms explored in the literature. A particular paradigm is selected for further analysis. The current state-of-the-art in EEG signal decoding for the selected EEG paradigm is explored both from a technological and theoretical perspective.

Chapter 3 expands on the theory of decoding algorithms that are most applicable to this project. This theory expansion is vital as it lends a far easier understanding of the final implementation in software. Various computational approaches are explored and their advantages and disadvantages analysed.

Chapter 5 explores the design of the embedded software required to execute the decoding algorithms introduced in the previous chapter. Auxiliary components, such as networking and communication infrastructure, are also explored. Computational challenges and the evolution of their solutions are presented.

Results of the implemented BCI system are provided in Chapter 6. Results for preliminary verification and diagnostic tests are provided, followed by those attained in decoding experiments performed using the final system prototype.

A discussion surrounding the system design and methodology, as well as results obtained, is provided in Chapter 7. Finally, conclusions are drawn and suggestions for improvement in further work are provided in Chapter 8.

[3] by reducing contact impedance

Chapter 2

Literature Review

2.1 Basic neurophysiology

2.1.1 Electrophysiology

EEG relies on electrophysiological activity generated by electro-chemical neurotransmitters that exchange signals between neurons in the brain [12]. Roughly speaking, when a neuron is excited by afferent action potentials (rapid changes in cell membrane potentials), postsynaptic potentials (EPSPs) are generated in its apical dendrites [2] (neural branches). This induces a potential difference (dipole) which results in a small flow of current from the nonexcited soma membrane to the apical dendrites where the EPSPs are present. [2]. This process causes both intracellular current flow within the neuron, as well as extracellular current flow. These current flows are termed *primary* and *secondary* or *return* currents respectively. Both such currents contribute to electrical potentials measured on the scalp. However, it is believed that the source of most measurable EEG potentials arises from large collections of cortical pyramidal neurons arranged in macro-assemblies with dendrites orientated perpendicularly to the local cortical surface [2], [3]. The specific spatial positioning and simultaneous activation of these large clusters of neurons is believed to generate signals that can be measured on the scalp. Specifically, [2] suggests that these signals most likely arise from EPSPs in these macro-assemblies and less so due to rapidly firing action potentials that travel along the axons of excited neurons.

2.1.2 Functional neuroimaging

Various techniques exist for measuring and translating brain activity into electrical signals. These techniques are typically grouped into electrophysiological and hemodynamic. Hemodynamic techniques such as functional magnetic resonance (fMRI) and near infrared (NIR) spectroscopy measure brain activity *indirectly* by tracking relative concentrations of oxyhemoglobin. The blood releases glucose to *active* neurons at a greater rate than inactive ones which causes an increase in oxyhemoglobin in these active areas [12], thereby providing a measurable proxy for cerebral activity. While hemodynamic approaches offer superior spatial resolution, they typically offer far lower temporal resolution and require complex, expensive equipment.

EEG, on the other hand, is a electrophysiological technique that relies on the direct measurement of electrical signals generated by neural cell assemblies. Owing to the fact that it offers high temporal resolution[1], far lower cost than most other techniques, good portability and safety, EEG has become the most widely used neuroimaging modality [12]. It is these factors that make EEG most appropriate for this project.

2.2 Electroencephalography

2.2.1 Invasive vs non-invasive techniques

The ability to reliably detect EEG signals is made challenging by the fact that neuron potentials must pass through the skull and scalp and will unavoidably be measured in conjunction with background noise and other undesirable artefacts such as electromyography (EMG) signals. Invasive methods that require surgical implantation of electronic devices are sometimes used in order to circumvent some of these challenges. Electrocorticography or intercranial EEG is an example of an invasive method; an electrode array is placed directly on the exposed surface of the brain to measure cerebral activity. Intracortical signal analysis is another emerging invasive encephalography technique used in BCIs that involves implantation of electrodes in

[1] typically in the order of tens of milliseconds

the grey matter of the brain. Examples of this technique include local field potential (LFP), single/multi=unit activity (S/MUA) and entire spiking activity (ESA).

While invasive methods offer superior signal resolution and quality, they are clearly not suitable for this project. Non-invasive BCIs do not require any surgical intervention and only involve the placement of electrodes on the scalp of the subject. Despite the aforementioned challenges of measuring signals of poorer quality, this has the advantage of convenience, cost effectiveness, safety and minimal invasiveness.

2.2.2 The nature of EEG signals

EEG signals most commonly range in amplitude from 0.5 to 100μV peak-to-peak [3] (in the case of a healthy brain). For reference, this is roughly 100 times lower than the typical amplitude of ECG signals. It is widely believed that EEG signals show varying energy in a few distinct frequency bands depending on mental state and cognitive function of a subject [2], [6]. These frequency bands are summarised below [6], [12]:

Frequency band	Physiological association
delta (0.5 - 4Hz)	Usually only detected in a state of deep sleep. Excessive signal energy in the delta band while awake may suggest neurological disease.
theta (4 - 8Hz)	Cognitive tasks involving association, awareness and meditation. Usually low energy in this band while subject is awake.
alpha (8 - 13Hz)	Typically measured in the occipital region of the brain. Primarily related to visual processing but also memory processes. Induced by closing of the eyes and relaxing and attenuated when eyes are open or by thinking or mental calculation.
beta (12 - 30Hz)	A variety of mental processes such as mathematical computation, planning, high level processing

Table 2.1: Summary of commonly designated EEG signal frequency bands and their broad physiological associations

It is worth noting that different areas of the brain may produce signals with different energy compositions across these frequency bands. Consequently, the particular placement of electrodes in measuring localised signals of interest is an important consideration.

2.2.3 Electrode choice and placement

An EEG is signal is most commonly measured as the potential difference over time between active and reference electrodes. Electrodes be used in conjunction with conductive media such as conductive gel or without and would be termed wet or dry electrodes in these two cases respectively.

Electrodes are most commonly arranged on the scalp according to the International 10-20 system standardised by the American Electroencephalographic Society. An overview of this system is presented in Figure 2.1.

2.2.4 BCI control signals

The core role of a BCI is to interpret the intentions of a subject by making sense of their brain signals. These signals are comprised of a superposition of many different neuronal potentials associated with various mental tasks, most of which are not yet understood or clearly identifiable. However, there are some mental processes that have proved to correspond to identifiable signals that can be decoded by BCIs. These signals are either produced predictably in response to a particular external stimulus, or can be be modulated at will by a subject with suitable conditioning [12]. Some of the most popular BCI control signals are discussed below.

Visual evoked potentials

Visual evoked potentials (VEPs) are modulations in the activity of the brain's visual cortex in response to a visual stimulus [1]. VEPs can be characterised by the nature of the visual stimulus used to evoke them, namely [12]:

- the flicker or reversal frequency of stimulus images or shapes
- the morphology of the stimuli

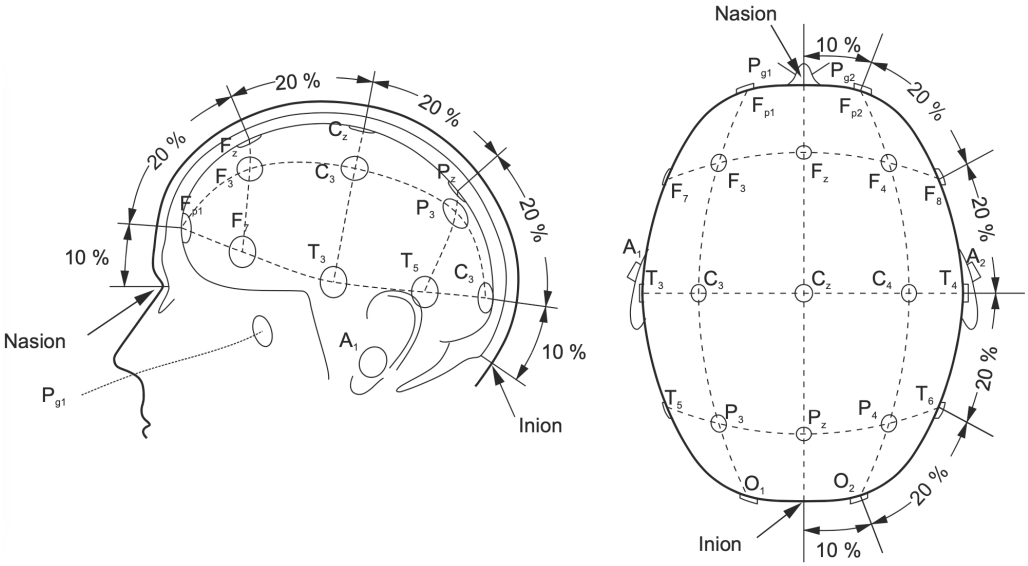


Figure 2.1: Diagram illustrating typical electrode placements using the international 10-20 system. A represents the ear lobe, C the central region, P the parietal, F the frontal and O the occipital. The *naison* and *inion* are used as reference locations.

- the proportion of the visual field occupied by the stimulus

Most commonly, frequency is modulated between different visual stimuli in order to encode different control targets. Transient VEPs, as the name suggests, are short-term potentials that occur in response to visual stimuli below 6Hz. Conversely, steady state VEPs (SSVEPs) are produced at frequencies above 6Hz and are characterised by sinusoidal signals with a fundamental frequency matching that of the visual stimulus [17]. VEPs have the advantage of a relatively high information transfer rate (ITR) and do not require any training from the subject as they are elicited involuntarily. ITR is measured in bits/min and is a measure of how much information can be communicated by a BCI per unit time. Higher ITRs are desirable as these correspond to more responsive BCIs.

Slow cortical potentials

Slow cortical potentials (SCPs) are slow voltage shifts below 1Hz that correspond to changes in the level of cortical activity [12]. Although these signals can be self-regulated in order to control external devices through a BCI, reliable operation often requires training and is dependent on numerous external factors such as a subject's psychological state, motivation and social context [12]. Furthermore, as SCPs occur over several seconds, maximum ITRs attainable are low.

Event related potentials

Event related potentials (ERPs), such as the P300 evoked potential, are positive signal spikes generated in response to infrequent or unexpected auditory, visual or somatosensory stimuli [12]. 'P300' is derived from the fact that these potentials are typically evoked around 300ms after observing an infrequent stimulus after a sequence of common or expected ones. P300-based BCIs are only capable of very low ITRs since considerable number of non-target stimuli need to be presented before the infrequent stimulus in order to preserve its novelty. Furthermore, magnitude of P300 responses may decrease with time as the subject begins to anticipate responses.

Sensorimotor rhythm

Sensorimotor rhythms, such as motor imagery, are modulations in cerebral activity associated with motor tasks, even when overt motor action is not performed. BCI control can be achieved through sensorimotor signals as subjects can be trained to generate modulations voluntarily through mental rehearsal of motor actions. For example, a subject may imagine clenching their fists. However, obtaining reliable signals through self-control proves difficult in practice as patients often imagine *images* of related movements which does not produce sufficiently similar cerebral activity to the thought of performing the actual activity itself [12]. As such, reliable BCI control usually requires special training with an emphasis on kinesthetic experiences.

Paradigm	Physiological phenomena	Training required	ITR (bits/min)
VEP	Signal modulations in the visual cortex in response to a flickering visual stimulus	No	**60-200**
SCP	Slow shifts in cortical potentials due to modulation of cortical activity/concentration	Yes	5-12
ERP	Abrupt signal modulations in response to infrequent or unexpected stimuli	No	20-25
Sensorimotor	Modulations in signals from the motor cortex synchronised to imagined motor actions	Yes	3-35

Table 2.2: A summary of commonly used BCI control signal paradigms. The information transfer rate (ITR) values provided are typical but may vary depending on the type of data acquisition and decoding systems used.

2.3 Steady-state visual evoked potentials (SSVEP)

Several studies suggest that steady-state visual evoked potentials [39], [28], [32] (SSVEPs) offer significant potential for EEG decoding tasks similar to those in this project due to the high information transfer rate (ITR), non-invasiveness and relatively high SNR that can be achieved using even basic BCI devices [37]. Moreover, SSVEP amplitudes change as a function of stimulus intensity (luminance and contrast) [26] which can easily be controlled. Considering these factors and the fact that SSVEP-based BCIs require little to no user training or prior BCI experience, it is clear that this is the most suitable EEG paradigm for this project.

2.3.1 Evoking and measuring SSVEPs

Broadly speaking, in order to evoke SSVEPs, flickering visual stimuli with frequencies of around 7-15Hz are commonly used [32], [19], [27], however, Xie *et al* demonstrated successful decoding with 27.8Hz [17]. As in [32], [19], [26], visual stimuli are usually presented using a computer monitor with an LCD display or discrete LEDs [18]. SSVEPs are measured in the occipital, parietal or parieto-occipital regions [39] for proximity to the visual cortex.

The aforementioned studies do not provide a unanimous set of parameters for the optimal SSVEP stimulus configuration. There is substantial variation in the frequency, colour and source of stimulus used across these studies and indeed, many other studies in the literature [27]. Duart *et al* in [27] sought to investigate these stimulus design parameters more closely and compare their impact on accuracy of frequency discrimination and SNR (signal-to-noise ratio) in SSVEP-based BCIs. Specifically, the following parameters were tested:

- **frequency**: low (5Hz), middle (12Hz) and high (30Hz) frequencies were tested.
- **colour**: red, white and green squares were experimented with.
- **attention**: a measure of attention was tested using to determine correlation with evoked responses.

Duart *et al* found that the middle frequency of 12Hz produced maximal SNR, followed by the low 5Hz frequency. Furthermore, red and green stimuli produced responses with maximum SNR near 5Hz while red and white were optimal at 12Hz. No difference in SNR was observed between colours at 30Hz. Despite red light proving optimal in other studies such as [20], the authors note that red light may produce increased risk of inducing epileptic seizures and should thus be avoided where possible. Moreover, Zhu *et al* noted that lower (flicker) frequencies and light colours of longer wave length tend to produce greater visual fatigue which consequently degrades SSVEP responses over time [7]. Finally, Duart *et al* measured attention of subjects undergoing SSVEP trials through the Conner's Continuous Performance Task version 2 (CPT-II) which measures reaction times, omission errors and commission errors [27]. They found that this measure of attention showed a significant correlation to the SNR of evoked SSVEP signals at low frequencies around 5Hz, but less so with higher frequencies. The authors hypothesised that this may be due to the fact that greater concentration is required at lower frequencies that cause greater fatigue [27].

Taking into account these findings, it would appear that using stimulus frequencies of around 12Hz would be a suitable choice. Furthermore, so as to avoid using red light, white would be a good choice in medium frequency range and green would be suitable for lower frequencies around 5Hz. Using frequencies around 12Hz has the added advantage of causing less visual fatigue and theoretically, should require less active concentration from subjects in order to achieve adequate SNR.

Electrode placement

As alluded to above, SSVEP signals are produced in the visual cortex located in the occipital region of the brain. Accordingly, almost all studies of SSVEP decoding use electrodes in the occipital or parieto-occipital regions [26], [32], [20], [20]. Referring to the 10-20 electrode positions in Figure 2.1, this corresponds to electrodes in the O_z, O_1, O_2, PO_z, $PO1$ and $PO2$ positions. Some studies suggest that when few channels are available, position O_z should be prioritised as it offers a relatively high density of SSVEP signals [26].

Sampling

In practical systems, there is almost always substantial 50Hz or 60Hz mains power interference. Therefore, in order to satisfy the Nyquist criterion, sampling rate of at least $f_s > 100$Hz (or 120Hz for regions with 60Hz) is required. In order to account for higher frequency components that are likely to be non-negligible in measured EEG signals, most studies of BCI systems use sampling rates of 200Hz [29], 250Hz [15], 256Hz [34], [16] or 500Hz [6], [27].

Another important consideration is the length of the recording window used to capture SSVEP signals. Most commonly, sliding windows of fixed length T are used with some degree of overlap between successive windows such that $\tau_{n+1}(0) - \tau_n(0) < T$ where $\tau_n(0)$ marks the start time of window n. It is desirable to *minimise* T since this affords a higher information transfer rate (ITR) which results in a more responsive BCI. However, all studies mentioned in the previous paragraph indicate that SSVEP classification accuracy improves with T. Thus, the question is more about determining the minimum window length T to achieve a desired degree of average accuracy. This varies considerably based on the sophistication of the BCI hardware and decoding algorithm, among other factors. Therefore, a suitable time window for this project should be determined empirically. Based on somewhat comparable studies in [21], [20], [26], [11] and others, suitable values of T are expected to be in the region of 1.5 to 3.5 seconds.

Presentation and arrangement of stimuli

The authors in [31] studied the effects of the arrangement of SSVEP stimuli on the decoding accuracy thereof. As with many such studies, a series of four squares designed to flash at independent stimulus frequencies were arranged on a contrasting background. The four different arrangements tested in [15] were adapted for use in this project and are shown in Figure 2.2. The authors found that for recording windows of length $T > 3$s, there was no significant difference in decoding performance among the different configurations. However, it was noticed that for $T < 3$s, a spatial configuration most similar to (b) was optimal. Configuration (a) was reported to be the most visually frustrating for subjects undergoing the experiment and less so for the more dispersed configurations. Considering that, even for $T < 3$s, configuration (b) was only marginally better, (c) may pose a good balance between decoding performance and user experience.

2.4 Computational Approaches for SSVEP Decoding

2.4.1 Power spectral density and frequency domain

Arguably the most obvious approach for analysing periodic signals in order to distinguish frequency content is through the use of Fourier analysis. Power spectral density analysis (PSDA) is typically used to analyse the power of a signal over a continuous frequency range. For a discrete time deterministic signal $x[n]$ with finite energy $\sum_{n=-\infty}^{\infty} |x[n]|^2 < \infty$, the discrete time Fourier transform (DTFT) is defined as

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} \tag{2.1}$$

for complex variable j and frequency variable ω with $\omega = 2\pi f$. The energy spectral density $S_x(\omega)$ of $x[n]$ is defined as $S_x(\omega) = |X(\omega)^2|$. The power spectral density (PSD) $P_x(\omega)$ of $x[n]$ of finite length N is defined as

$$P_x(\omega) = \lim_{N\to\infty} E\left\{ \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n]e^{-jn\omega} \right|^2 \right\} \tag{2.2}$$

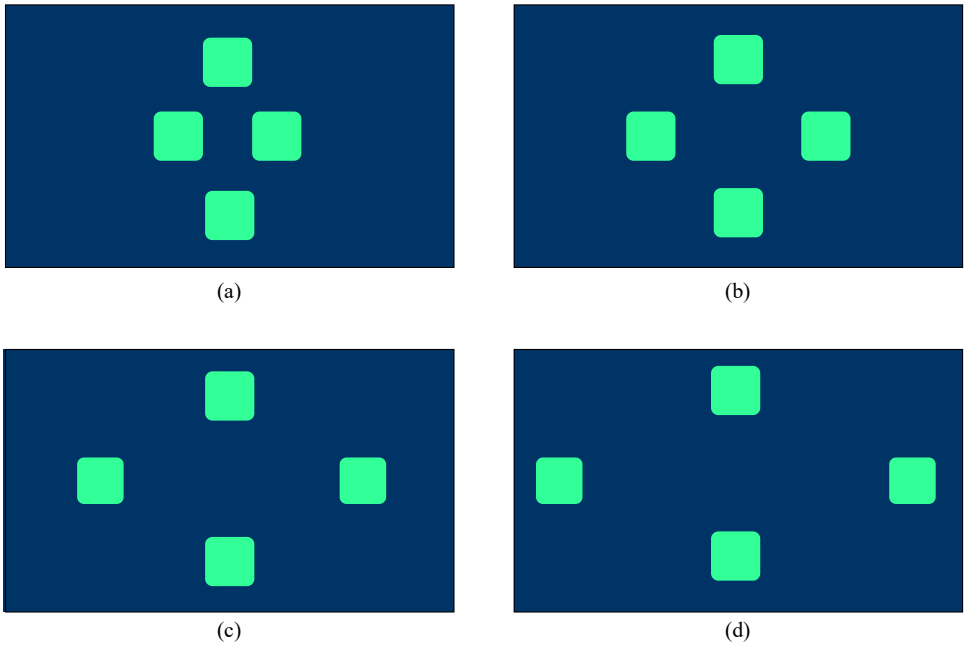


Figure 2.2: Experimental SSVEP stimulus grid configurations adapted from [31]. The absolute dimensions and positions of the stimulus squares and the background are not important here; only their relative positioning is relevant

The PSD represents the distribution of signal power over a frequency range of interest. In practice, one can only consider finite length signals with $N < \infty$ and so, $P_x(\omega)$ must be estimated (albeit arbitrarily closely). Such an estimate of power spectral density is called a **periodogram**.

In the context of SSVEP decoding, it is clear how PSDA could be useful. Simply by computing the periodogram (estimated PSD) of a measured EEG signal $x[n]$, the target SSVEP stimulus frequency $f^* \in \mathcal{F}$ could be inferred from the set of candidate stimulus frequencies $\mathcal{F}$ by choosing $f^* = \text{argmax}_{f \in \mathcal{F}} \widehat{P}'_x(\omega)$ where $\widehat{P}'_x(\omega)$ is the adjusted signal spectrum obtained by subtracting a baseline spectrum measured with no active stimulus. This is necessary as EEG signals tend to follow a $1/f$ power distribution whose magnitude naturally decays with frequency. Therefore, without "zeroing" the active spectrum, the lower, non-target stimulus frequencies may have more power which could produce erroneous results.

Welch's method for PSD estimation

Welch's method is commonly used to reduce noise in PSD analysis. This approach involves dividing the original signal into short overlapping windows of length $N^* \leq N$ which are then windowed (in the time domain). The periodograms of these windows are computed independently and then averaged. While this approach reduces variance in the final periodogram, it comes at the cost of decreased frequency resolution. Furthermore, the type of window used should be chosen carefully as this will further impact the resolution-variance trade off. Hamming, Blackman and Bartlett windows are popular choices.

Efficacy of PSDA for SSVEP decoding

Many studies have demonstrated that PSDA can be used for SSVEP decoding/classification [31], [12], [32]. However, it is widely documented that PSDA is inferior to more specialised statistical techniques that have been developed for this task: both in terms of signal classification accuracy and information transfer rate (ITR) [26], [5], [35], [15], [8]. The authors in [8] performed a study specifically to compare the efficacy of PSDA vs statistical algorithms such as CCA (explored below) in EEG signal decoding. Their findings reinforce those of their colleagues in that CCA and its derivatives produce better decoding performance than PSDA.

2.4.2 Statistical

Canonical correlation analysis (CCA)

CCA is a standard multivariate statistical technique for analysing multiple variables measured on a set of observations. In particular, variables are partitioned into two sets or *views* of the data [22]. Effectively, CCA is a multivariate extension of ordinary correlation. Given variable partitions $\mathrm{X} \in \mathbb{R}^{m \times p}$, $\mathrm{Y} \in \mathbb{R}^{m \times q}$ each with m observations, CCA seeks to find a linear combination of X, Y that maximises the correlation between their images $\mathbf{z}_X = \mathrm{X}\mathbf{w}_X$ and $\mathbf{z}_Y = \mathrm{Y}\mathbf{w}_Y$. These images are known as the *canonical variables* and $\mathbf{w}_X$ and $\mathbf{w}_Y$ are the *canonical weight vectors* that effectively act as spatial filters across signal channels.

With CCA for SSVEP frequency recognition tasks, $\mathrm{X} \in \mathbb{R}^{N_s \times N_c}$ is the set of *measured* EEG signals from N_c channels over N_s observations. The measured signals in X must be compared with each frequency $f_k \in \mathcal{F}$, a finite set of candidate stimulus frequencies. CCA can be used to compute weighted correlations between the measured signals and each candidate frequency f_k by constructing a sinusoidal reference signal set $\mathrm{Y}_k \in \mathbb{R}^{N_s \times 2N_h}$ as follows:

$$\mathrm{Y}_k = \begin{pmatrix} \sin(2\pi f_k t) \\ \cos(2\pi f_k t) \\ \sin(4\pi f_k t) \\ \cos(4\pi f_k t) \\ \vdots \\ \sin(2\pi N_h f_k t) \\ \cos(2\pi N_h f_k t) \end{pmatrix}, \quad t = \frac{1}{f_s}, \frac{2}{f_s}, \ldots, \frac{N_s}{f_s} \tag{2.3}$$

where f_s is the sampling frequency and N_h is the number of harmonics in the reference set, a parameter to be chosen. Since the scalp and other interacting layers between the brain and surface electrodes exhibit low-pass filter dynamics [2], [5], it is rare that more than $N_h = 5$ harmonics are selected in the reference signal set, and most commonly $N_h = 3$ is chosen [5]. The final task of frequency recognition is performed by selecting the frequency f^*

$$f^* = \underset{f_k \in \mathcal{F}}{\operatorname{argmax}}\ \rho_k \tag{2.4}$$

corresponding to the candidate frequency f_k which maximises the canonical correlation ρ_k between X and the the reference set Y_k. While standard CCA is a good starting point for statistical SSVEP decoding as it forms the basis of many similar algorithms, there are several extensions which have been shown to significantly improve signal recognition performance [15], [35], [34]. The most promising of these are explored below.

Task-related component analysis (TRCA)

Task-related component analysis (TRCA) as a decoding technique in biophysical systems was introduced in the seminal paper by Tanaka *et al* [13]. TRCA seeks to extract task-related signal components (as opposed to spurious, unwanted components such as EMG artefacts and background noise) by identifying components that lead to maximum reproducibility across trials. Task-related components are constructed as linear combinations of signals recorded across multiple time blocks (trials) and their weights are optimised so as to maximise their inter-block correlation or covariance [13]. Components with maximum inter-block covariance and thus, reproducibility or consistency, are identified as task-related components (subject to certain significance measures as discussed later). Further advantages of TRCA are that, unlike other more common generalised linear models (GLMs), it assumes no *a priori* knowledge of source signals and is not sensitive to autocorrelation [13]. It also offers concrete measures of task-relatedness unlike some other data-driven methods like independent component analysis.

To illustrate the basic idea of this algorithm, [13] presents a useful toy example. Consider a task-related signal $s(t)$ standardised to have zero mean and unit variance embedded in additive white (uncorrelated) noise $n(t)$: $x(t) = a_1 s(t) + a_2 n(t)$ with some mixing scalars $a_i \in \mathbb{R}$. Then, assume the observable signal $x(t)$ is measured over two distinct time blocks or trials to yield $x_1(t)$ and $x_2(t)$. A linear model of the two observed blocks could be defined as follows:

$$x_1(t) = a_{11} s(t) + a_{12} n(t) \tag{2.5}$$
$$x_2(t) = a_{21} s(t) + a_{22} n(t) \tag{2.6}$$

As mentioned above, TRCA attempts to recover task related components (only $s(t)$ in this case) from a linear combination of observations across trials. Defining this weighted sum as $y(t)$ with

$$y(t) = y_1(t) + y_2(t) = w_1 x_1(t) + w_2 x_2(t) = (w_1 a_{11} + w_2 a_{21})\, s(t) + (w_1 a_{12} + w_2 a_{22})\, n(t) \tag{2.7}$$

Recovering $s(t)$ involves maximising the covariance between $y_1(t)$ and $y_2(t)$:

$$\begin{aligned} \text{Cov}\,(y_1, y_2) &= (w_1 a_{11} + w_2 a_{21})^2 \,\text{Cov}\,(s_1, s_2) + (w_1 a_{12} + w_2 a_{22})^2 \,\text{Cov}\,(n_1, n_2) \\ &\quad + (w_1 a_{11} + w_2 a_{21})\,(w_1 a_{12} + w_2 a_{22})\,[\text{Cov}\,(s_2, n_2) \; + \text{Cov}\,(n_1, s_2)] \\ &= (w_1 a_{11} + w_2 a_{21})^2 \,\text{Cov}\,(s_1, s_2) \end{aligned} \tag{2.8}$$

Since $\text{Cov}(n_1, n_2) = \text{Cov}(s_1, n_2) = \text{Cov}(n_1, s_2) = 0$. In order to bound this quadratic objective, we choose the weights w_1, w_2 so that the variance of y is constrained to 1:

$$\text{var}(y) = (w_1 a_{11} + w_2 a_{21})^2 + (w_1 a_{12} + w_2 a_{22})^2 = 1 \tag{2.9}$$

where $s(t)$ and $n(t)$ are also assumed to be uncorrelated $\forall t$. The solution of this constrained optimisation problem is

$$w_1 a_{11} + w_2 a_{21} = 1, \quad w_1 a_{12} + w_2 a_{22} = 0, \tag{2.10}$$

which, barring the unlikely case that $a_{11} a_{22} = a_{12} a_{21}$, yields $y(t) = s(t)$ as desired. This demonstrates how inter-trial covariance maximisation can be used for recovery of task-related signals.

Although TRCA has been shown to outperform CCA [33], [34], [35], extended and hybrid forms of these algorithms have produced superior results. Some of these extensions are explored below.

Multiset canonical correlation analysis (MsetCCA)

MsetCCA is one extension of standard CCA that takes into account historical data instead of performing inference purely on new observations. Zhang *et al* propose that this is one of the reasons that standard CCA performs poorly on short time windows; it effectively over fits to localised dynamics [15]. Furthermore, the authors suggest that exclusively using the pre-constructed sinusoidal reference set is not optimal since this artificial reference does not include other features from real EEG data [15]. To circumvent this, MsetCCA seeks to optimise the reference signals used in the CCA algorithm by learning multiple linear transforms to maximise overall correlation between canonical variables over *many* sets of EEG data at each candidate frequency $f_k \in \mathcal{F}$ [15]. This optimisation effectively finds optimal joint spatial filters $\mathbf{w}_1, \ldots, \mathbf{w}_{N_t}$ (over N_t trials) using only historical observations ('training' data). The authors claim that MsetCCA outperforms similar techniques, especially in cases with few channels and short time windows.

Generalised canonical correlation analysis (GCCA)

Wong *et al* noted in their study of EEG spatial filtering methods in [30] that using historical observations to estimate spatial filters (canonical weights) $\mathbf{w}_i$ can be combined with harmonic reference signals as in ordinary CCA. Combining these techniques was found to enhance the algorithm's robustness and reduced calibration requirements [30]. Motivated by this, the authors in [35] developed a new algorithm called generalised CCA (GCCA) that aims to simultaneously maximise correlation between three sets of data: historical observations, measured signals in a new sample and the pre-constructed harmonic reference. As interpreted by the authors in [35], the optimal spatial filters obtained through GCCA perform SSVEP signal denoising.

2.5 Existing BCI Technology

Commercial BCI devices are currently available from several manufacturers, including but not limited to: B-Alert X10, NeuroSky, Emotiv and OpenBCI. These devices are typically targeted at consumers but some are also aimed at research applications (such as the Emotiv Epoc Flex). This review will not explore these

devices further as they are all priced in excess of several hundred USD which violates the important budget constraint on this project.

More of interest to this project is the low-cost, lightweight BCI devices being developed in the EEG research community. Several studies have presented promising BCI prototypes that have demonstrated that reliable EEG signal decoding is possible in significantly resource-constrained settings [21], [14], [11], [18], [25]. In particular, these studies have explored devices that are constrained in terms of computing capacity, mobility or form factor, number of channels available and cost.

However, a thorough review of these existing BCI prototypes did not reveal any that satisfied *all* constraints or requirements of this project. Uktveris *et al* demonstrated a very promising and impressive device with wireless capabilities (communication with a host PC or other device using Bluetooth Low Energy (BLE) and/or Wi-Fi) and some on-device digital signal processing ability [21]. However, their device requires final signal decoding to be performed on a host device separate from the actual BCI itself. Furthermore, the cost of the raw materials in their project was around 113 EUR [21] (albeit for a more capable multi-channel BCI system).

The authors in [18] created an embedded BCI system consisting of a small, on-board analogue sampling device that communicates using RF signals with another external main controller. The authors claim this to be a 'wireless' BCI but in fact, all computation and decoding happens on the fixed main controller and the on-board controller only serves to acquire the raw signals.

Acampora *et al* provide a slightly more feasible prototype in [32] and provide an expansion for SSVEP decoding using this device in [25]. This device was able to achieve maximum SSVEP classification accuracy of 74.5% for a 2s recording window, and up to 92.7% over a 4s window (using a support vector machine with Gaussian or RBF kernel) [25]. However, this prototype utilises an off-the-shell Olimex EEG-SMT device for signal acquisition and a Raspberry Pi 3 for computation and decoding. While their device is seemingly effective and fairly mobile, it would certainly not satisfy the £20 price budget in this project. Housing a Raspberry Pi 3 and battery pack along with the rest of the BCI hardware would also be overly cumbersome for this project.

Chapter 3

Theory Development

3.1 Eigenvalue Optimisation

Many, if not all, of the statistical decoding algorithms mentioned so far require a form of eigenvalue optimisation to find a solution. Worded differently, these are optimisation problems that can be conveniently reformulated as eigenvalue problems.

3.1.1 Optimisation form I

The first form, which arises in the CCA algorithm, addresses constrained optimisation problems of the form

$$\begin{aligned} \underset{\mathbf{w}}{\text{maximize}} \quad & \mathbf{w}^\top \mathrm{A}\mathbf{w} \\ \text{subject to} \quad & \mathbf{w}^\top \mathbf{w} = 1, \end{aligned} \tag{3.1}$$

with variable $\mathbf{w} \in \mathbb{R}^d$ and $\mathrm{A} \in \mathbb{R}^{d \times d}$. Employing Lagrange dual theory, the Lagrangian for (3.1) is

$$\mathcal{L} = \mathbf{w}^\top A\mathbf{w} - \lambda \left(\mathbf{w}^\top \mathbf{w} - 1 \right), \tag{3.2}$$

with Lagrange multiplier $\lambda \in \mathbb{R}$ [23]. Referencing the Karush-Kuhn-Tucker (KKT) conditions, *stationary* requires $\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0$:

$$\mathbb{R}^d \ni \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 2A\mathbf{w} - 2\lambda \mathbf{w} \overset{\text{set}}{=} 0 \Longrightarrow A\mathbf{w} = \lambda \mathbf{w}, \tag{3.3}$$

yielding the standard eigenvalue problem in A. Since (3.1) was posed as a maximisation problem, the optimal $\mathbf{w}$ corresponds to the largest eigenvalue in A. For a minimisation problem, it would correspond to the smallest eigenvalue.

3.1.2 Optimisation form II

The second form is an extension of the prior univariate case to a multivariable system $\mathrm{W} \in \mathbb{R}^{d \times d}$. Consider the following constrained optimisation problem:

$$\begin{aligned} \underset{\mathrm{W}}{\text{maximize}} \quad & \mathrm{tr}\left(\mathrm{W}^\top \mathrm{A}\mathrm{W} \right) \\ \text{subject to} \quad & \mathrm{W}^\top \mathrm{W} = \mathrm{I}, \end{aligned} \tag{3.4}$$

where tr(.) represents the matrix trace and $\mathrm{A} \in \mathbb{R}^{d \times d}$. The Lagrangian for this problem, taken from [23], is the following:

$$\mathcal{L} = \mathrm{tr}(\mathrm{W}^\top \mathrm{A}\mathrm{W}) - \mathrm{tr}(\Lambda^\top (\mathrm{W}^\top \mathrm{W} - \mathrm{I})), \tag{3.5}$$

where $\Lambda \in \mathbb{R}^{d\times d}$ is a diagonal matrix whose diagonal entries are the Lagrangian multipliers. Once again, applying the *stationarity* condition, (3.4) can be reformulated to the following eigenvalue problem [23]:

$$\mathbb{R}^{d\times d} \ni \frac{\partial \mathcal{L}}{\partial \mathrm{W}} = 2A\mathrm{W} - 2\mathrm{W}\Lambda \overset{\text{set}}{=} 0 \tag{3.6}$$

$$\implies A\mathrm{W} = \mathrm{W}\Lambda \tag{3.7}$$

The diagonal elements of Λ are the eigenvalues of A and are arranged in descending order for the maximisation problem and ascending order for the minimisation problem. The columns of W are the corresponding eigenvectors.

3.1.3 Power iteration

The power iteration algorithm is a simple iterative algorithm used to find the eigenvector $\mathbf{q}_1 \in \mathbb{R}^d$ corresponding to the largest eigenvalue of a matrix $A \in \mathbb{R}^{d\times d}$ [9]. It relies on the property that if $\mathbf{q}$ is an eigenvector of matrix A with eigenvalue λ, then $\mathrm{A}^k\mathbf{q} = \lambda^k\mathbf{q}$, $k \in \mathbb{N}$. Let unit vector $\mathbf{v}^{(0)} \in \mathbb{R}^d$ be the initial estimate of an eigenvector of A. Assuming that A is full rank with real eigenvalues, its eigenvectors $\mathbf{q}_i$ are orthogonal and form a basis for $\mathbb{R}^d$. Therefore, $\mathbf{v}^{(0)}$ can be expressed as a linear combination of $\mathbf{q}_i$:

$$\mathbf{v}^{(0)} = \sum_{i=1}^{d} c_i\mathbf{q}_i \quad \text{for} \quad c_i \in \mathbb{R},\ \forall i \tag{3.8}$$

Assuming that $c_1 \neq 0$:

$$\mathrm{A}\mathbf{v}^{(0)} = \sum_{i=1}^{d} c_i\lambda_i\mathbf{q}_i \implies \mathrm{A}^k\mathbf{v}^{(0)} = \sum_{i=1}^{d} c_i\lambda_i^k\mathbf{q}_i \tag{3.9}$$

Then, taking λ_1^k as a common factor,

$$\mathrm{A}^k\mathbf{v}^{(0)} = \lambda_1^k\left(c_1\mathbf{q}_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k\mathbf{q}_2 + \ldots + c_n\left(\frac{\lambda_n}{\lambda_1}\right)^k\mathbf{q}_n\right) \tag{3.10}$$

Assuming that the eigenvalues λ_i are assumed to be real, distinct and descending in magnitude, it is clear to see that for $i = 2, \ldots, d$,

$$\lim_{k\to\infty}\left(\frac{\lambda_i}{\lambda_1}\right)^k = 0 \tag{3.11}$$

Thus, over multiple iterations (increasing k), $\mathrm{A}\mathbf{v}^{(0)} \to c_1\lambda_1^k\mathbf{q}_1$ and $\mathbf{v}^{(k)}$ approaches the eigenvector $\mathbf{q}_1$:

$$\mathbf{q}_1 \approx \frac{\mathrm{A}\mathbf{v}^{(0)}}{\|\mathrm{A}\mathbf{v}^{(0)}\|} \tag{3.12}$$

An approximation of the eigenvalue corresponding to $\mathbf{q}_1$ can be found using the Rayleigh quotient for $\mathbf{v}^{(k)}$ defined as

$$r(\mathbf{v}) = \frac{\mathbf{v}^\top\mathrm{A}\mathbf{v}}{\mathbf{v}^\top\mathbf{v}} \approx \frac{\lambda\mathbf{v}^\top\mathbf{v}}{\mathbf{v}^\top\mathbf{v}} = \lambda, \tag{3.13}$$

where the iteration index k has been omitted for readability. Although elegant, this algorithm is limited in that it only returns a single eigenvector and only converges if eigenvalues $\lambda_1, \lambda_2, \ldots$ are *distinct*. However, for problems that only require solving of the largest eigenvalue/eigenvector, this is a useful algorithm that is easy to implement.

3.1.4 Simultaneous iteration

The power iteration algorithm, along with its extensions such as the inverse power iteration and Rayleigh Quotient iteration algorithms, only compute a single eigenvector/eigenvalue at a time. In order to compute several eigenvalues, these methods must be reapplied several times [9]. Simultaneous iteration offers a method of solving all eigenvalues at once.

As in (3.8) in the power iteration algorithm, only non-trivial eigenvectors in the linear sum with $c_i \neq 0$ can be found by the algorithm. Equivalently, only eigenvectors *not orthogonal* to $\mathbf{v}^{(0)}$ can be selected by the algorithm. This suggests that if different, mutually-orthogonal initial vectors $\mathbf{v}^{(0)}$ are selected across runs, there is a chance of finding different eigenvalues [9]. Using this idea, instead of beginning with $\mathbf{v}^{(0)}$, consider an initial basis of d linearly independent vectors arranged as the columns of a matrix $\mathrm{V}^{(0)}$:

$$\mathrm{V} = \left[\begin{array}{c|c|c} \mathbf{v}_1^{(0)} & \dots & \mathbf{v}_d^{(0)} \end{array} \right] \tag{3.14}$$

If we let

$$\mathrm{V}^{(k)} = \mathrm{A}^{(k)}\mathrm{V}^{(0)} = \left[\begin{array}{c|c|c} \mathbf{v}_1^{(k)} & \dots & \mathbf{v}_d^{(k)} \end{array} \right], \tag{3.15}$$

the standard power iteration is effectively applied to all the vectors $\mathbf{v}_1^{(0)}, \dots, \mathbf{v}_d^{(0)}$ at once [9]. However, in order to actually extract different eigenvectors from the set of distinct initial vectors (columns of V), we must orthonormalise the columns of $\mathrm{V}^{(k)}$ at each iteration. This is done in the univariate version of the power iteration algorithm; the multivariate analogue is to ensure the set of eigenvector estimates (columns of $\mathrm{V}^{(k)}$ are orthonormal in each iteration [9]. This is achieved by using the QR decomposition of $\mathrm{V}^{(k)}$ which decomposes a matrix into the product of an orthonormal matrix Q and an upper triangular matrix R. In each iteration, a new matrix W is obtained by multiplying A by the latest eigenvector approximation matrix $\mathrm{V}^{(k-1)}$. W is updated to the orthonormal columns of $\mathrm{Q}^{(k)}$ from the QR decomposition of $\mathrm{W}^{(k-1)}$.

3.1.5 QR iteration algorithm

While the simultaneous iteration is effective under some conditions, it is not commonly used in practice [9]. The QR algorithm is a subtle yet elegant adjustment: in each iteration, A is modified directly. Specifically, factor matrices Q and R from the QR decomposition of $\mathrm{A}^{(k-1)}$ are simply multiplied in reverse order to obtain $\mathrm{A}^{(k)} = \mathrm{R}^{(k)}\mathrm{Q}^{(k)}$. The motivation behind this is not immediately obvious or intuitive but can be explained through a restructuring of the simultaneous iteration algorithm as detailed at length in [9].

It should be noted that both the simultaneous iteration and QR iteration algorithms may fail if the eigenvalues of A are not real or if the eigenvectors thereof do not form a basis for $\mathbb{R}^d$.

3.2 SSVEP Decoding Algorithms

As mentioned in Section 2.4.1, statistical algorithms selected or designed for SSVEP decoding have proven to be significantly more successful than PSDA approaches. Therefore, only the former will be explored in this section.

3.2.1 Canonical correlation analysis (CCA)

Given variable partitions $\mathrm{X} \in \mathbb{R}^{m \times p}$, $\mathrm{Y} \in \mathbb{R}^{m \times q}$ each with m observations, CCA seeks to find a linear combination of X, Y that maximises the correlation between their images. More formally, assuming that the variables (columns) of the two partitions $\mathbf{x}_1, \dots, \mathbf{x}_p \in \mathbb{R}^m$ and $\mathbf{y}_1, \dots, \mathbf{y}_q \in \mathbb{R}^m$ are standardised to have zero mean and unit variance, find

$$\rho = \max_{\mathbf{w}_X, \mathbf{w}_Y} \mathrm{corr}(\mathrm{X}\mathbf{w}_X, \mathrm{Y}\mathbf{w}_Y) \tag{3.16}$$

$$= \max_{\mathbf{w}_X, \mathbf{w}_Y} \frac{\mathbb{E}[\mathrm{X}\mathbf{w}_X\mathbf{w}_Y^\top\mathrm{Y}^\top]}{\sqrt{\mathbb{E}[\mathrm{X}\mathbf{w}_X\mathbf{w}_X^\top\mathrm{X}^\top]\mathbb{E}[\mathrm{Y}\mathbf{w}_Y\mathbf{w}_Y^\top\mathrm{Y}^\top]}} \tag{3.17}$$

where ρ is the *canonical correlation*, $\mathbf{w}_X$ and $\mathbf{w}_Y$ are the canonical weight vectors and the images $\mathbf{z}_X = X\mathbf{w}_X$ and $\mathbf{z}_Y = Y\mathbf{w}_Y$ are the *canonical variables* for X and Y respectively.

Geometric interpretation

The authors in [22] present an intuitive geometric analogy. Consider X and Y as linear transforms of position vectors $\mathbf{w}_X$ and $\mathbf{w}_Y$ onto their images $\mathbf{z}_X$ and $\mathbf{z}_Y$ in $\mathbb{R}^m$. CCA constrains $\mathbf{z}_X$ and $\mathbf{z}_Y$ to have unit norm and that the angle $\theta \in [0, \frac{\pi}{2}]$ between them be minimised so as to maximise their correlation. The canonical correlation ρ is then the inner product of $\mathbf{z}_X$ and $\mathbf{z}_Y$:

$$\rho = \mathbf{z}_X^\top \mathbf{z}_Y = \|\mathbf{z}_X\| \|\mathbf{z}_Y\| \cos(\theta) = \cos(\theta) \tag{3.18}$$

Therefore, CCA aims to find position vectors $\mathbf{w}_X$, $\mathbf{w}_Y$ that, after undergoing linear transforms X and Y, are mapped to a unit sphere in $\mathbb{R}^m$ where θ is minimised [22]. Note that there will be $n = \min(p, q)$ canonical correlations $\rho_1, \ldots, \rho_n$ with $|\rho_i| \geq |\rho_j|$, $\forall\, i < j$. Typically, only the first and largest canonical correlation ρ_1 is considered.

Solving for canonical weight vectors through the eigenvalue problem

Consider the case of solving for the first and largest canonical correlation p_1. Following the geometric interpretation, the objective is to find

$$\cos(\theta_1) = \max_{\mathbf{w}_X, \mathbf{w}_Y} \langle \mathbf{z}_X, \mathbf{z}_Y \rangle \quad \text{with} \quad \|\mathbf{z}_X\| = \|\mathbf{z}_Y\| = 1 \tag{3.19}$$

where θ_1 is the smallest angle between image vectors. Defining the within-set covariance matrices as

$$C_{XX} = X^\top X \quad \text{and} \quad C_{YY} = Y^\top Y \tag{3.20}$$

and the inter-set cross covariance matrix as

$$C_{XY} = X^\top Y, \tag{3.21}$$

the unit variance constraint on the image vectors can be rewritten as

$$\begin{aligned} \|\mathbf{z}_X\| &= \mathbf{z}_X^\top \mathbf{z}_X = \mathbf{w}_X^\top X^\top X \mathbf{w}_X = \mathbf{w}_X^\top C_{XX} \mathbf{w}_X = 1 \\ \|\mathbf{z}_Y\| &= \mathbf{z}_Y^\top \mathbf{z}_Y = \mathbf{w}_Y^\top Y^\top Y \mathbf{w}_Y = \mathbf{w}_Y^\top C_{YY} \mathbf{w}_Y = 1 \end{aligned} \tag{3.22}$$

Substituting (3.22) into (3.17) yields

$$\cos(\theta) = \max_{\mathbf{w}_X, \mathbf{w}_Y} \mathbf{w}_X^\top C_{XY} \mathbf{w}_Y \tag{3.23}$$

Lagrangian dual theory provides a straight forward way to solve the constrained optimisation problem above with unit variance constraints as in (3.22). Consider the Lagrangian for this problem below with Lagrange multipliers λ_1 and λ_2:

$$\mathcal{L}(\mathbf{w}_X, \mathbf{w}_Y, \lambda_1, \lambda_2) = \mathbf{w}_X^\top C_{XY} \mathbf{w}_Y - \frac{\lambda_1}{2}(\mathbf{w}_X^\top C_{XX} \mathbf{w}_X - 1) - \frac{\lambda_2}{2}(\mathbf{w}_Y^\top C_{YY} \mathbf{w}_Y - 1) \tag{3.24}$$

With reference to the KKT conditions, stationary requires

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_X} = \mathrm{C}_{XY}\mathbf{w}_Y - \lambda_1 \mathrm{C}_{XX}\mathbf{w}_X = 0 \tag{3.25}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_Y} = \mathrm{C}_{YX}\mathbf{w}_X - \lambda_2 \mathrm{C}_{YY}\mathbf{w}_Y = 0 \tag{3.26}$$

Since $\mathrm{C}_{YX} = \mathrm{C}_{XY}^\top$. Pre-multiplying (3.25) by $\mathbf{w}_X^\top$ and (3.26) by $\mathbf{w}_Y^\top$ yields:

$$\begin{aligned} \mathbf{w}_X^\top \mathrm{C}_{XY}\mathbf{w}_Y - \lambda_1 \mathbf{w}_X^\top \mathrm{C}_{XX}\mathbf{w}_X = 0 \\ \mathbf{w}_Y^\top \mathrm{C}_{YX}\mathbf{w}_X - \lambda_2 \mathbf{w}_Y^\top \mathrm{C}_{YY}\mathbf{w}_Y = 0 \end{aligned} \tag{3.27}$$

Since $\mathbf{w}_X^\top \mathrm{C}_{XX}\mathbf{w}_X = \mathbf{w}_Y^\top \mathrm{C}_{YY}\mathbf{w}_Y = 1$, it is clear from (3.27) that $\lambda_1 = \lambda_2 = \lambda$. Substituting this result into (3.25) yields

$$\mathbf{w}_X = \frac{\mathrm{C}_{XX}^{-1}\mathrm{C}_{XY}\mathbf{w}_Y}{\lambda} \tag{3.28}$$

Then, substituting into (3.26) produces the following generalised eigenvalue problem:

$$\mathrm{C}_{YX}\mathrm{C}_{XX}^{-1}\mathrm{C}_{XY}\mathbf{w}_Y = \lambda^2 \mathrm{C}_{YY}\mathbf{w}_Y \tag{3.29}$$

If C_{YY} is non-singular, this reduces to a standard eigenvalue problem of the form

$$\mathrm{C}_{YY}^{-1}\mathrm{C}_{YX}\mathrm{C}_{XX}^{-1}\mathrm{C}_{XY}\mathbf{w}_Y = \lambda^2 \mathbf{w}_Y \tag{3.30}$$

The eigenvalues of the matrix $\mathrm{C}_{YY}^{-1}\mathrm{C}_{YX}\mathrm{C}_{XX}^{-1}\mathrm{C}_{XY}$ correspond to the squares of the canonical correlations $\rho_1, \ldots, \rho_n$.

3.2.2 Task-related component analysis (TRCA)

Expanding from the toy example in Section 2.4.2, the TRCA algorithm implicitly assumes that observed signals are generated as a linear combination of task-related and task-unrelated components which implies that task-related components can be recovered by applying an appropriate weighting of observed signals across trials [13]. Inter-trial covariance maximisation is the core objective that aims to extract task-related components with maximal temporal similarity across trials.

Now that multiple trials are to be considered, the signal matrix $\mathrm{X} \in \mathbb{R}^{N_s \times N_c}$ introduced in Section 3.2.1 must be extended to a third order signal tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N_s \times N_c \times N_t}$ with N_t trials, each with N_c channels[1] and N_s samples. Let $\mathbf{y}^{(k)}$ be the k-th trial (block) with $k \in \{1, 2, \ldots, N_t\}$ of the signal $y(t)$ and $\mathbf{x}_i^{(k)}$ be the k-th trial of the input signal from channel i, $i \in \{1, \ldots, N_c\}$. Consider the covariance between all i channels for two trials k and l:

$$\widehat{\mathrm{C}}_{kl} = \mathrm{Cov}\left(\mathbf{y}^{(k)}, \mathbf{y}^{(l)}\right) = \sum_{i,j=1}^{N_c} w_i w_j \,\mathrm{Cov}\left(\mathbf{x}_i^{(k)}, \mathbf{x}_k^{(l)}\right) \tag{3.31}$$

[1] the constraint that the number of variables (width of X_i) for each trial be constant is not strictly necessary. For this application, however, it is assumed that the number of channels will not change between trials of a given experiment.

Furthermore, as with CCA, we must impose the constraint on the designed weights such that the variance of $y(t)$ is bounded:

$$\mathrm{Var}(\mathbf{y}) = \sum_{i,j=1}^{N_c} w_i w_j \,\mathrm{Cov}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \mathbf{w}^{\mathrm{T}}\mathrm{Q}\mathbf{w} = 1 \tag{3.32}$$

Then, all possible combinations of trials $1, \ldots, N_t$ are summed as

$$\begin{aligned}\sum_{\substack{k,l=1,\\ k\neq l}}^{N_t} \widehat{\mathrm{C}}_{kl} &= \sum_{\substack{k,l=1,\\ k\neq l}}^{N_t} \mathrm{Cov}\left(\mathbf{y}^{(k)}, \mathbf{y}^{(l)}\right) \\ &= \sum_{\substack{k,l=1,\\ k\neq l}}^{N_t} \sum_{i,j=1}^{N_c} w_i w_j \,\mathrm{Cov}\left(\mathbf{x}_i^{(k)}, \mathbf{x}_j^{(l)}\right) = \mathbf{w}^{\top}\mathrm{S}\mathbf{w}\end{aligned} \tag{3.33}$$

where S is the following symmetric matrix

$$\mathrm{S}_{i,j} = \sum_{\substack{k,l=1,\\ k\neq l}}^{N_t} \mathrm{Cov}\left(\mathbf{x}_i^{(k)}, \mathbf{x}_j^{(l)}\right) \tag{3.34}$$

As alluded to before, (3.33) yields the sum of covariances of all combinations of trials: a measure of task consistency. This is precisely what we want to maximise. This constrained optimisation problem (recalling the variance constraint on $y(t)$) can be reformulated as a Rayleigh-Ritz eigenvalue problem of the form

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^{\top}\mathrm{S}\mathbf{w}}{\mathbf{w}^{\top}\mathrm{Q}\mathbf{w}} \tag{3.35}$$

The inter-trial weight vectors $\mathbf{w}_k$, $k \in \{1, \ldots, N_t\}$ can be found by computing the eigenvectors of $\mathrm{Q}^{-1}\mathrm{S}$. Specifically, the degree of task-relatedness corresponds to the magnitude of the eigenvalues of $\mathrm{Q}^{-1}\mathrm{S}$ and so the optimal weight $\mathbf{w}^*$ corresponds to the largest eigenvalue.

3.2.3 Multiset CCA (MsetCCA)

MsetCCA is an extension of standard CCA to multiple data sets or partitions. Accordingly, the objective is now to maximise the correlation between canonical variables from *many sets* of observations at a given stimulus frequency f_k. Although several objective functions for MsetCCA exist, the MAXVAR objective is explored in [15] for its intuitive extension to ordinary CCA with multiple variable sets. Assuming all sets of variables in $\boldsymbol{\mathcal{X}}$ are normalised to have zero mean and unit variance, the MAXVAR objective for maximising correlation over canonical variables from multiple sets at a given candidate frequency f_k is defined as follows:

$$\begin{aligned}\max_{\mathbf{w}_1,\ldots,\mathbf{w}_{N_t}} \quad & \rho = \sum_{i\neq j}^{N_t} \mathbf{w}_i^{\top}\mathrm{X}_i^{\top}\mathrm{X}_j\mathbf{w}_j \\ \text{s.t.} \quad & \frac{1}{N_t}\sum_{i=1}^{N_t} \mathbf{w}_i^{\top}\mathrm{X}_i^{\top}\mathrm{X}_i\mathbf{w}_i = 1,\end{aligned} \tag{3.36}$$

where the same conventions for the signal tensor $\boldsymbol{\mathcal{X}}$ as in Section 3.2.2 above apply. Following a similar approach as in the derivation for CCA above, the method of Lagrange multipliers can be used to transform the constrained optimisation problem in (3.36) to a generalised eigenvalue problem of the form:

$$(\mathrm{R} - \mathrm{S})\mathbf{w} = \rho\mathrm{S}\mathbf{w} \tag{3.37}$$

where

$$\mathrm{R} = \begin{bmatrix} \mathrm{X}_1^\top \mathrm{X}_1 & \dots & \mathrm{X}_1^\top \mathrm{X}_{N_t} \\ \vdots & \ddots & \vdots \\ \mathrm{X}_{N_t}^\top \mathrm{X}_1 & \dots & \mathrm{X}_{N_t}^\top \mathrm{X}_{N_t} \end{bmatrix}, \quad \mathrm{S} = \begin{bmatrix} \mathrm{X}_1^\top \mathrm{X}_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \mathrm{X}_{N_t}^\top \mathrm{X}_{N_t} \end{bmatrix} \quad \text{and} \quad \mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_{N_t} \end{bmatrix}$$

Thus, R is the inter-trial block covariance matrix which captures sub-covariance matrices between all pairs of trials $n \in \{1, \dots, N_t\}$. S is a block diagonal matrix which captures within-set covariance matrices for each trial. $\mathbf{w}$ is a matrix of optimal spatial filters (vectors) $\mathbf{w}_1, \dots, \mathbf{w}_{N_t}$ resulting from the largest combined canonical correlation between all canonical variables $\mathbf{z}_i = \mathrm{X}_i \mathbf{w}_i \in \mathbb{R}^{N_s}$, $\forall\, i \in [1,\ N_t]$. As with standard CCA, the largest canonical correlation ρ^* corresponds to the largest generalised eigenvalue in (3.37). The canonical variables $\mathbf{z}_i$ corresponding to ρ^* are indeed the eigenvectors corresponding to the largest generalised eigenvalue [15]. The optimal reference $\mathrm{Y}_k \in \mathbb{R}^{N_s \times N_t}$ for given frequency f_k can be computed using the spatial filters from $\mathbf{w}$ as:

$$\mathrm{Y}_k = \begin{bmatrix} \mathbf{z}_{1,k} & \dots & \mathbf{z}_{N_t,k} \end{bmatrix} = \begin{bmatrix} \mathrm{X}_{1,k}\mathbf{w}_{1,k} & \dots & \mathrm{X}_{N_t,k}\mathbf{w}_{N_t,k} \end{bmatrix} \tag{3.38}$$

After this training or calibration process is complete and Y_k is computed for all candidate frequencies f_k, these optimised reference sets can be used for inference. Given a new set of test signals $\hat{\mathrm{X}} \in \mathbb{R}^{N_s \times N_c}$, ordinary CCA as in Section 3.2.1 can be used to discern the frequency f^* corresponding to the highest canonical correlation between $\hat{\mathrm{X}}$ and the associated referenced set Y^*. The process for finding f^* is identical to that in (2.4). The important difference here is how the pre-computed reference sets Y_k used in the CCA algorithm are calculated.

3.2.4 Generalised CCA (GCCA)

Consider a single candidate frequency f_k: for the rest of this subsection, the k index is ommitted for brevity but all symbols refer to those specific to the single set of trials for a distinct stimulus frequency f_k. As with MsetCCA, GCCA requires a signal tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N_s \times N_c \times N_t}$ that incorporates several trials. The *template matrix* $\overline{\mathrm{X}}$ is obtained by computing the arithmetic mean of all trials in the set: $\overline{\mathrm{X}} = \frac{1}{N_t} \sum_{i=1}^{N_t} \boldsymbol{\mathcal{X}}_i$. The concatenated signal matrix X^c is formed by unrolling $\boldsymbol{\mathcal{X}}$ along the third (trial) axis:

$$\mathrm{X}^c = \begin{bmatrix} \mathrm{X}_1^\top & \mathrm{X}_2^\top & \dots & \mathrm{X}_{N_t}^\top \end{bmatrix} \in \mathbb{R}^{N_c \times (N_s N_t)} \tag{3.39}$$

Then, the template and sinusoidal reference matrices are concatenated similarly to match the dimensions of X^c:

$$\overline{\mathrm{X}}^c = \begin{bmatrix} \overline{\mathrm{X}}^\top & \overline{\mathrm{X}}^\top & \dots & \overline{\mathrm{X}}^\top \end{bmatrix} \in \mathbb{R}^{N_c \times (N_s N_t)} \quad \text{and} \quad \mathrm{Y}^c = \begin{bmatrix} \mathrm{Y}^\top & \mathrm{Y}^\top & \dots & \mathrm{Y}^\top \end{bmatrix} \in \mathbb{R}^{2N_h \times (N_s N_t)} \tag{3.40}$$

where Y follows the definition of the sinusoidal reference in (2.3). Consider the augmented spatial filter vector $\widetilde{\mathbf{w}}$:

$$\widetilde{\mathbf{w}} = \begin{bmatrix} \mathbf{w}_{\mathrm{X}^c} & \mathbf{w}_{\overline{\mathrm{X}}^c} & \mathbf{w}_{\mathrm{Y}^c} \end{bmatrix}^\top, \tag{3.41}$$

where, for example, component weight $\mathbf{w}_{\mathrm{X}^c}$ corresponds to concatenated signal matrix X^c, and so on. Similarly, the augmented signal matrix $\widetilde{\mathrm{X}}$ is defined as:

$$\widetilde{\mathrm{X}} = \begin{bmatrix} (\mathrm{X}^c)^\top & (\overline{\mathrm{X}}^c)^\top & (\mathrm{Y}^c)^\top \end{bmatrix}^\top \tag{3.42}$$

The objective of GCCA can then be expressed as

$$\begin{aligned} \text{maximise} \quad & \text{tr}(\widetilde{\mathbf{w}}^\top \widetilde{\mathrm{X}}\widetilde{\mathrm{X}}^\top \widetilde{\mathbf{w}}) \\ \text{s.t.} \quad & \widetilde{\mathbf{w}}^\top \mathrm{D}\widetilde{\mathbf{w}} = \mathrm{I} \end{aligned} \tag{3.43}$$

where D, the within-set block covariance matrix, is defined as

$$\mathrm{D} = \begin{bmatrix} \mathrm{X}^c(\mathrm{X}^c)^\top & 0 & 0 \\ 0 & \overline{\mathrm{X}}^c(\overline{\mathrm{X}}^c)^\top & 0 \\ 0 & 0 & \mathrm{Y}^c(\mathrm{Y}^c)^\top \end{bmatrix} \tag{3.44}$$

Again, using the method of Lagrange multipliers, (3.43) can be reformulated as a generalised eigenvalue problem of the form

$$\widetilde{\mathrm{X}}\widetilde{\mathrm{X}}^\top \widetilde{\mathbf{w}} = \lambda \mathrm{D}\widetilde{\mathbf{w}} \tag{3.45}$$

If D is non-singular, (3.45) resolves to an ordinary eigenvalue problem:

$$\mathrm{D}^{-1}\widetilde{\mathrm{X}}\widetilde{\mathrm{X}}^\top \widetilde{\mathbf{w}} = \lambda \widetilde{\mathbf{w}} \tag{3.46}$$

Then, the eigenvector of $\mathrm{D}^{-1}\widetilde{\mathrm{X}}\widetilde{\mathrm{X}}^\top$ corresponding to its largest eigenvalue is the optimal spatial filter $\mathbf{w}^* \in \mathbb{R}^{2(N_c+N_h)}$.

Given a new test sample set $\hat{\mathrm{X}} \in \mathbb{R}^{N_s \times N_c}$, two correlations are computed: first between the test data and the historical template, and then between the test data and sinusoidal reference. This can be expressed as

$$\rho_1 = \text{corr}(\hat{\mathrm{X}}\mathbf{w}_{\mathrm{X}^c},\ \overline{\mathrm{X}}\mathbf{w}_{\overline{\mathrm{X}}^c}) \tag{3.47}$$

$$\rho_2 = \text{corr}(\hat{\mathrm{X}}\mathbf{w}_{\mathrm{X}^c},\ \mathrm{Y}\mathbf{w}_{\mathrm{Y}^c}) \tag{3.48}$$

where corr(.) denotes Pearson correlation. Finally, the output correlation for frequency f_k is computed as a combination of ρ_1 and ρ_2:

$$\rho = \text{sign}(\rho_1)\rho_1^2 + \text{sign}(\rho_2)\rho_2^2 \tag{3.49}$$

Chapter 4

Apparatus and Experimental Procedure

4.1 BCI Apparatus

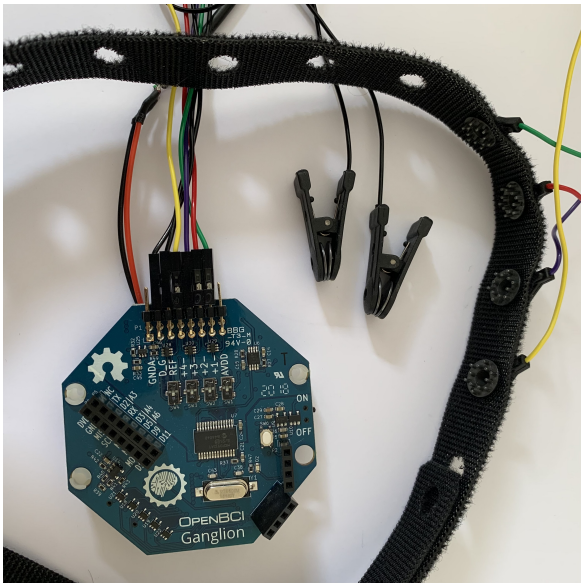
In general, EEG-based BCI systems are comprised of the following core elements [3], [12]:

- electrodes: placed on the scalp of the subject to record raw electrical potentials.
- signal processing elements: amplifiers and filters are typically employed before the signals are digitised
- analogue-to-digital converter (ADC): digitises measured signal for manipulation in a computer or microcontroller
- computer or microcontroller: facilitates data processing, computation and storage

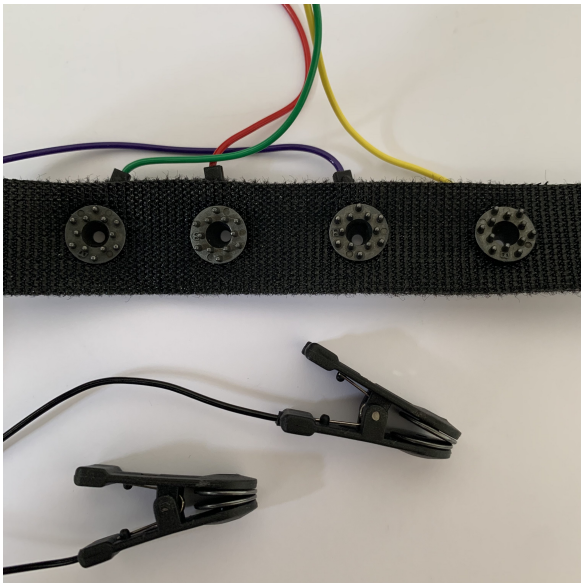
4.1.1 OpenBCI Ganglion

As the Imperial NGNI hardware prototype was still under development for a large part of this project, experimentation was initially done on a Ganglion bio-sensing kit made by OpenBCI (OpenBCI, New York, USA). This kit was chosen due to the fact that it is relatively low-cost at 374.99 USD, open-source and has been scientifically validated as in [26], [29]. Furthermore, it is a relatively simple platform with sensing capabilities that are likely the most comparable to those anticipated for the NGNI prototype.

As depicted in Figure 4.1, the Ganglion board offers four active channels. All four channels were using during experimentation as it was more convenient to exclude data from certain channels in post-processing than only



(a) OpenBCI Ganglion board, electrodes and adjustable Velcro headband



(b) Close-up view of the 4 active electrodes and two reference electrode clips (bottom)

Figure 4.1: Images of the OpenBCI Ganglion bio-sensing device and electrodes. The two black clips are the reference electrodes and the four active channels are received through the coloured wires.

measuring a subset of channels. Although wet electrodes can also be used with this kit, in accordance with the project constraints mentioned in Chapter 1, dry electrodes were used. As seen in Figure 4.1a, the four active electrodes have spiky nodules to increase surface pressure and thus improve contact quality with the scalp. For reference and comparison, some core features of the Ganglion board are provided below:

- Microchip MCP3912 four channel 24-bit Delta-Sigma analogue frontend
- 200Hz sampling rate
- Simblee Bluetooth 4.0 module

4.1.2 NGNI Prototype I

As alluded to in Chapter 1, the hardware to be used in this project was supplied by the Imperial NGNI Lab. All hardware prototypes developed by the Lab were based on the Espressif ESP32; a low-cost, low-power SoC (system-on-chip) based on the Tensilica Xtensa LX6 microprocessor with integrated Wi-Fi and Bluetooth. Features of the ESP32 that are relevant to this project include [38]:

- dual-core, 240MHz CPU
- onboard FPU
- 12-bit successive-approximation (SAR) ADC
- 4x SPI, 2x I2C, 3x UART interfaces
- up to 600 DMIPS performance
- ultra low-power (ULP) co-processor
- 4 MiB SRAM
- integrated Wi-Fi 802.11 b/g/n and BLE

The ESP32 is extremely capable for its low price tag of around 3.6 USD [38]. Its dual-core CPU is also particularly attractive as it could allow decoding-related computation and network communication to happen concurrently.

Figure 4.2 shows the electronic hardware prototype developed by the NGNI Lab. In this design, the active components directly involved in the normal functioning of the system are independently located on a 'target' board. A second programmer board was created to enable serial communication with the target; most commonly in order to flash new firmware to it during development. The programmer board uses six spring-loaded pogo pins to make momentary contact with corresponding pads on the target board during development. These 6 pins, and their corresponding pads on the target board, can be seen directly in the centre of Figure 4.2c. A 3D printed housing was created to facilitate correct contact between the boards during development. As depicted in the image of the programmer board in Figure 4.2b, it also features contact points for probing a few selected pins/junctions on the target board such as: the ADC output, the output of the instrumentation amplifier, the output of the analogue filter and all relevant supply voltage references. In addition, two momentary push-buttons connected to the `BOOT0` and `EN` pins of the ESP32 SoC are included in order to allow hardware resets and to control the entry state upon reset: bootloader mode or normal operating mode.

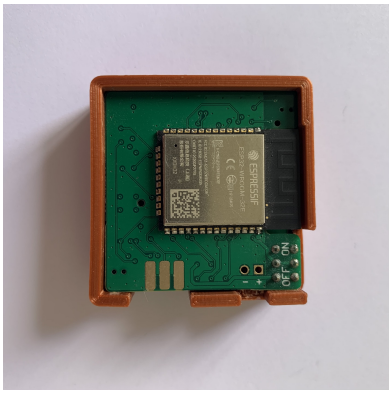
Analogue signal processing system

Figure 4.3 provides an overview of the key analogue signal processing elements in the electronic hardware prototype. This system is primarily responsible for amplifying μV-scale raw EEG signals, filtering out 50Hz mains power interference and offering further amplification that can be adjusted in firmware.

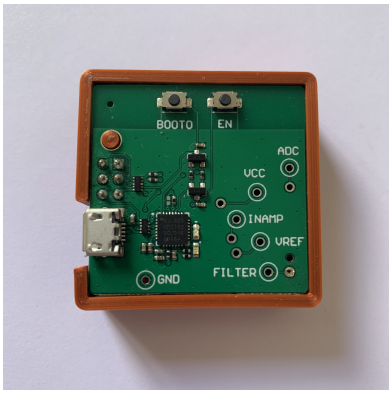
Important details of the components shown in Figure 4.3 are provided below:

(a) differential instrumentation amplifier
  - fixed gain of 1120
  - high-pass filter dynamics: highest corner frequency $f_c = 0.48$Hz

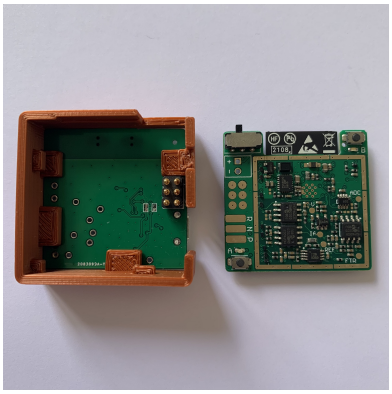
(b) hourglass low-pass filter
  - Q factor 2.17
  - low-pass corner frequency $f_c = 37.4$Hz



(a) Target board with ESP32 SoC and peripheral electronics



(b) Programmer board with serial-to-USB interface



(c) Bottom side of the programmer board (left) and target board

Figure 4.2: Images of the electronic hardware prototype developed in the Imperial NGNI Lab. This prototype includes the target board with ESP32 SoC, as well as a programmer board that is used to flash new firmware on to the microcontroller aboard the target board. The orange 3D-printed housing is used to position the programmer board above the target during firmware updates or other serial communication with an external computer.

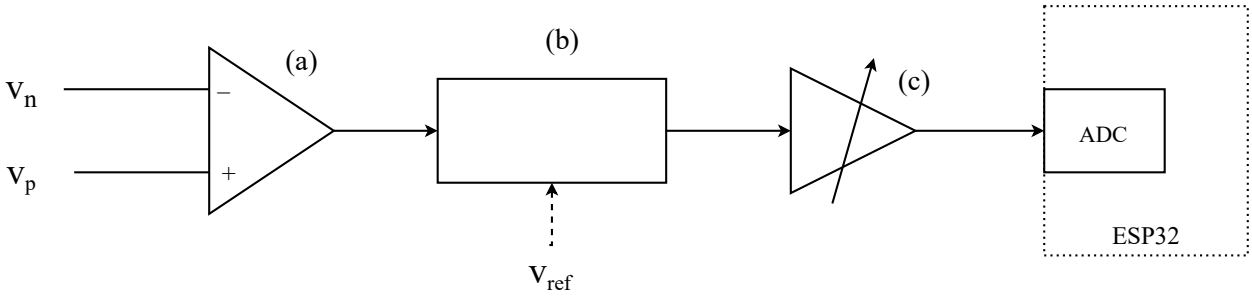


Figure 4.3: Functional overview diagram of the analogue signal processing system in the NGNI electronic hardware prototype. (a) signals captured from the two active electrodes, v_n and v_p, first pass through a differential instrumentation amplifier. (b) a third order hour glass low-pass filter is used for 50Hz mains power rejection. (c) the filtered signal can be further amplified with adjustable gain that is controlled via a digital potentiometer through a SPI interface.

- notch frequency $f_n = 50$Hz
- 50Hz rejection characteristics: minimum 17dB, nominal 35dB and maximum 55dB.

(c) adjustable-gain output amplifier

- adjustable gain between 1.745 and 19.2
- input low-pass filter dynamics: corner frequency of $f_c = 36.35$Hz

Important to note is that the analogue hourglass filter designed for 50Hz rejection has a corner frequency of $f_c = 37.4$Hz. Therefore, EEG signals with harmonics beyond 37.4Hz will start to become attenuated as a side-effect of the dynamics of this filter. Furthermore, as mentioned in Section 2.2.2, raw EEG signals are expected to be in the range of 0.5 to 100μV peak-to-peak. Neglecting other minor sources of gain or attenuation in the system, the ranges of signal amplitudes (peak-to-peak voltages) at the output of the system for various input EEG signal amplitudes and gain configurations is shown in Table 4.1. Note that, with the minimum gain configuration of $g_2 = 1.745$ on the output amplifier, the maximum expected output voltage is only $v_{\text{out}} = 0.132$V which is approximately equal to the full range of amplitudes expected with this configuration.

It should be noted that this first revision of the electronic hardware prototype is designed for coupling with *two* active electrodes but only allows for *differential* signal measurements between these electrodes. Consequently, only a single 'channel' is recorded by the ADC in the ESP32. A third reference electrode is also expected in order to offer a common voltage reference point between the two active channels.

v_{in} (μV)	g_1	g_2	g_1g_2	v_{out} (V)
0.5	1120	1.745	1315.4	**0.000658**
0.5	1120	19.2	21504	0.0108
100	1120	1.745	1315.4	0.132
100	1120	19.2	21504	**2.15**

Table 4.1: Table showing the theoretical range of voltages v_{out} at the input to the ADC of the ESP32 (output of the analogue signal processing system) with varying input signal amplitude v_{in} and gain configurations. Fixed gain g_1 is that of the input stage differential amplifier and g_2 is the adjustable gain of the output amplifier. The expected EEG signal amplitude range, v_{in}, is from [3]. Voltages shown are peak-to-peak amplitudes.

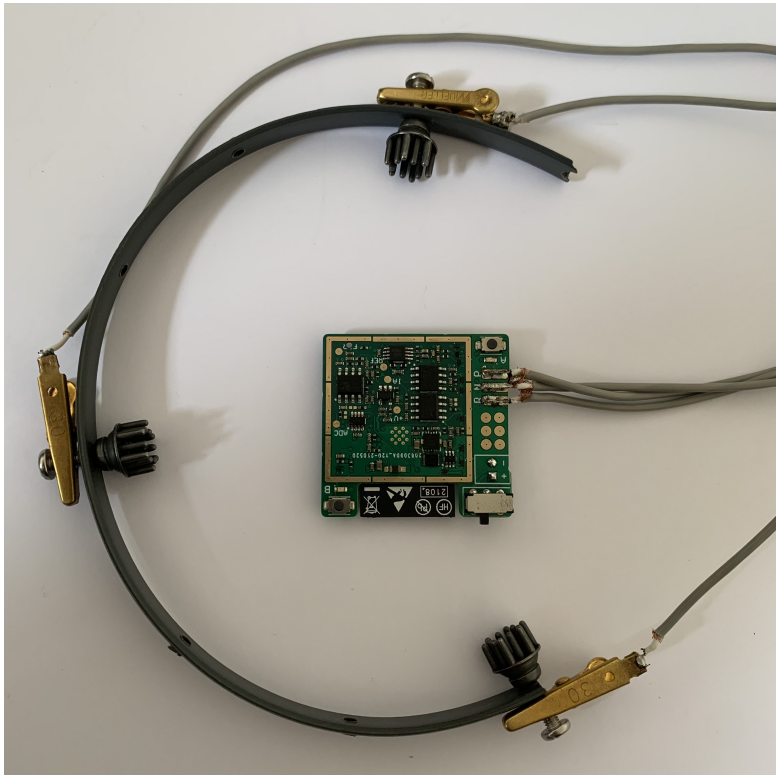


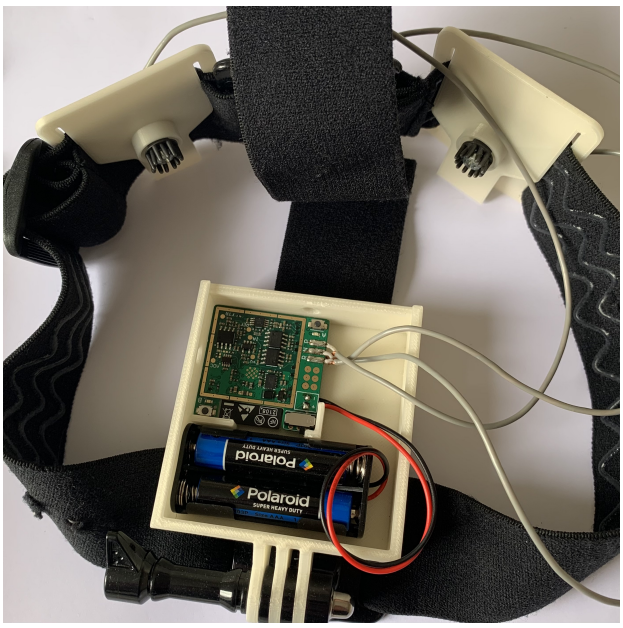
Figure 4.4: The first very rudimentary prototype EEG band developed by the NGNI Lab based on the electronic hardware prototype in Figure 4.2. The active electrodes are those at the two extremes of the band and the middle one is a reference electrode.

Mechanical hardware prototype

The very first complete EEG headband prototype comprising electronic and mechanical hardware is shown in Figure 4.4. This version did not include any means of adjusting the circumference of the band of the electrode positions within it besides having to drill new holes in it. As such, this makeshift prototype was more intended to serve as a preliminary means of gathering real-life data using the electronic hardware. It was not designed (nor expected) to produce viable results for SSVEP decoding.

4.1.3 NGNI Prototype II

Guided by the first EEG headband prototype, a second, improved version was developed by the NGNI Lab. This version, pictured in Figure 4.5, employed a flexible and adjustable strap mechanism. This enabled the band circumference to be adjusted so as to offer greater comfort for the user and ensure appropriate contact pressure with heads of any size or morphology. Furthermore, electrodes could be adjusted more easily thanks to the 3D-printed locator structures in which they were housed. These locator structures, seen in Figure 4.5b, allowed for greatly improved contact angles between the electrodes and the scalp surface compared to the first prototype. In particular, the slight pitch and yaw flexibility introduced by the electrode locators allowed the electrodes to remain perpendicular to the undulating scalp surface, thereby maximising contact surface



(a) View of the 3D-printed headband housing and adjustable elastic strap. The electronic hardware is externally powered by two 1.5V AAA alkaline batteries before being boosted to 3.3V using on-board circuitry.

(b) Close-up view of the two active electrodes on either side of the reference electrode. Electrodes are housed in 3D-printed locator structures that allow for slight pitch and yaw deviations about the axis normal to the scalp.

Figure 4.5: Images of the final complete hardware system comprising electronic and mechanical EEG hardware. All hardware was designed by the NGNI Lab. The band is intended to be worn such that the electrodes make contact with the back of the scalp in the occipital or parieto-occipital region.

area. However, their stiffness also prevented excessive pitch and yaw deviations which may have arisen if the electrodes were housed directly in the flexible headband strap.

4.2 Experimental Procedure

4.2.1 Data acquisition

Owing to limitations imposed by the Coronavirus Pandemic, all experimental data was recorded on the author. The SSVEP stimulus squares interface shown in Figure 5.1 was displayed on an 11" Apple iPad Pro (4^{th} generation, 2020) with Apple A12Z CPU, 2388x1688 px (264 PPI) display resolution and 120Hz refresh rate. The web page was displayed in the native Safari browser with no other apps running simultaneously so as to minimise CPU load and provide consistent flicker frequency across the stimulus squares. With reference to the diagram in Figure 4.6, the iPad was positioned $d = 50$cm away from the subject's visual field at an angle of $\theta = 45$ degrees below the horizontal line-of-sight. It should be noted, however, that this arrangement was followed approximately and is only provided as a rough guideline.

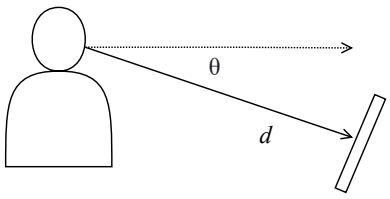


Figure 4.6: Diagram showing the arrangement of the stimulus display device relative to a subject undergoing the experiment. The dotted line represents the line-of-sight normal to the subject's eye line and the solid arrow is the position vector normal to the display device held distance d away from the visual field at angle θ below the line-of-sight.

4.2.2 Testing and verification

In order to verify that signals measured from by the BCI system were valid, several checks were devised.

Basic firmware test

The first very basic test involved verifying the integrity of the firmware on board the ESP32. This involved testing basic peripheral functionality such as toggling an LED connected to a GPIO pin and sampling a known, fixed value from the ADC. Storage and retrieval of data from flash, checking floating point precision and verifying operations from third party modules were also checked.

Alpha band test

As mentioned briefly in Section 2.2.2, EEG signal energy related to visual processing typically occurs in the *alpha* band between 8 and 13Hz and can be measured around the occipital region of the brain. As alpha energy is pronounced when the eyes are closed and attenuated when they are open, this phenomenon can be used as a basic test of the validity of a BCI. If signal energy as measured by the BCI in the occipital region follows this pattern, it suggests that acquired signals are not simply random noise. It should be noted, however, that individuals show varying levels of alpha reactivity and the difference in energy under these two conditions may not be significant in all individuals.

Digital signal processing tests

In order to test the integrity of digital signal processing (DSP) elements in the BCI system (explored more in detail in Chapter 5), the following preliminary checks were employed. In all of the following tests, data was recorded on the electronic hardware detailed above and was analysed offline.

- in order to test the on-device digital low-pass filter, data was recorded with and without the filter active. Analysis was performed to verify that signal components beyond the target frequency band were suitably attenuated and that pass-band distortion was acceptable.
- in order to verify the equivalence between original and down-sampled versions of the same signal, data was collected under three conditions: (i) sampling at $f_s = 256$Hz with low pass filtering but no downsampling, (ii) sampling at $f_s = 256$Hz with low pass filtering and downsampling to $f_s' = 64$Hz and (iii) sampling at $f_s = 256$Hz with *no* low-pass filtering and downsampling to $f_s' = 64$Hz.

Artificial signal decoding test

In order to test the hardware and signal decoding elements, a square wave signal at a known frequency f_0 was produced and applied across the active electrodes. The resulting signal spectrum was measured and compared to the theoretical spectrum of a square wave at frequency f_0. This was performed with and without low-pass filtering to determine if high frequency harmonics were being attenuated correctly. SSVEP signal decoding algorithms were also tested by ensuring that they produced a decoded output matching the artificial stimulus frequency f_0.

4.2.3 Demonstration procedure

During the Royal Society exhibition, audience members attending remotely from various locations will be presented a mobile-friendly, lightweight web page with several flickering squares that will form the SSVEP stimuli. These squares will be programmed to flicker at predetermined stimulus frequencies $f_1, \ldots, f_n$. Each stimulus will correspond to an action - such as 'up' or 'down' - that will control an avatar in a cooperative multiplayer game or simulation. The core objective of the designed BCI system is to decode $f_1, \ldots, f_n$ in order to interpret each user's desired action (i.e. discern which stimulus square they are focused on).

Chapter 5

System Design

5.1 Design of the SSVEP Stimuli

A key consideration in the design of the SSVEP stimuli is the stimulus frequencies $f_1, \ldots, f_n$ to use. As mentioned in Section 2.3.1, similar studies typically use stimulus frequencies between 7Hz and 12Hz for this task. Furthermore, as detailed in Section 2.4.2, CCA and CCA-based decoding algorithms involve a harmonic reference signal set. An important consideration is the *number of harmonics* to include in these reference sets. In order to allow for a fundamental stimulus frequency range of 7Hz to 12Hz, it is only feasible to include one harmonic ($N_h = 1$) in the reference set since the second harmonic of a 12Hz stimulus signal would occur at 36Hz which would likely be attenuated somewhat by the analogue low-pass filter (corner frequency of 37.4Hz) and low-pass filter dynamics of the adjustable output amplifier (corner frequency of 36.35Hz) shown in Figure 4.3. Furthermore, including the second harmonic for each stimulus frequency would have significant memory and computation implications. It is for these reasons that only a single harmonic was used in this system (where applicable). For the experiments conducted in this project, three stimulus frequency at 7, 10 and 12Hz were used.

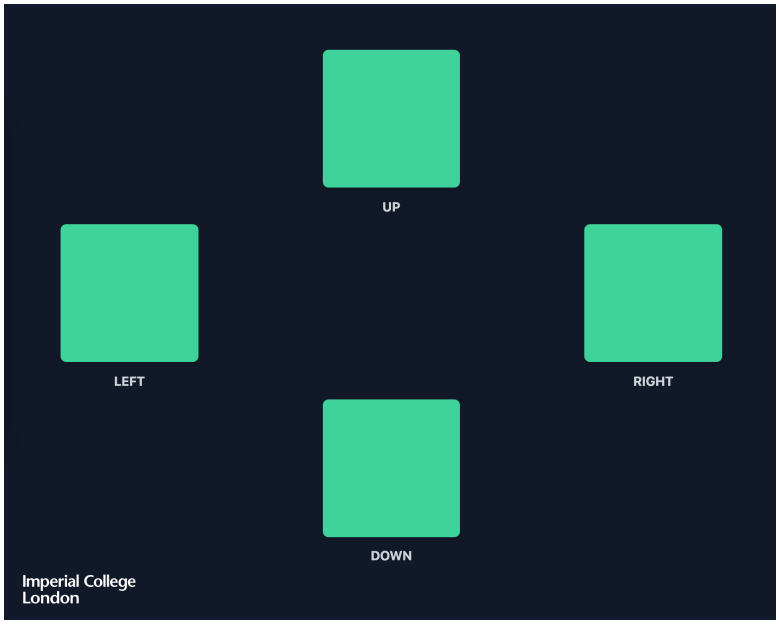


Figure 5.1: Screen capture of the user interface for displaying SSVEP stimuli. The blocks can be independently set to any flicker frequency of interest.

5.1.1 SSVEP stimulus interface

Naturally, an important part of a SSVEP-based BCI system is the SSVEP stimuli which are to be presented to individuals participating in the exhibition (or in general, anyone interacting with the system thereafter). Taking into account design parameters such as stimulus colour, position and contrast mentioned in the literature cited in Section 2.3.1, a basic user interface (UI) was designed with a series of flashing squares as

depicted in Figure 5.1. This UI was implemented[1] as a lightweight HTML page with basic CSS styling and Javascript to handle animation (flickering of the squares). This decision was made to allow for the simplest and most convenient deployment across any device capable of displaying a web page (mobile or otherwise).

Note that external factors related to this project may only require some subset of the stimuli shown in Figure 5.1; for example, the square corresponding to 'down' may be omitted if only the other three actions are required in the game/simulation. The configuration shown is for indicative purposes only.

5.2 Design of the Digital System

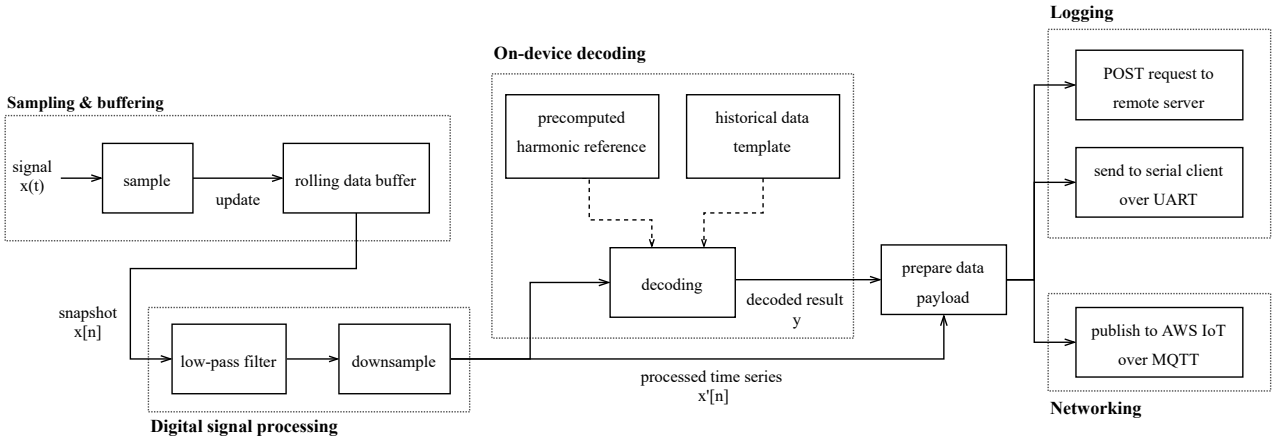


Figure 5.2: Overview diagram of the core components that comprise the digital system designed in this project.

5.2.1 Digital signal processing system

A crucial part in the design of the BCI system is the digital signal processing (DSP) system. The key functions of this system are to digitise, filter and resample the analogue output of the analogue signal processing system presented in Figure 4.3. An overview of the DSP system is shown in Figure 5.3.

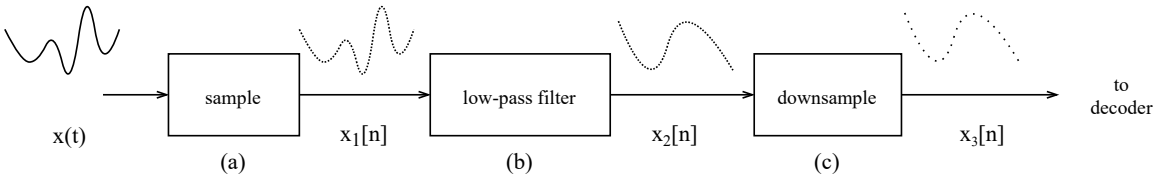


Figure 5.3: Overview diagram of the core components of the digital signal processing system.

Sampling and decimation

The analogue signal $x(t)$ is digitised to $x_1[n]$ using the 12-bit SAR ADC on-board the ESP32. A sampling frequency of $f_s = 256$Hz was selected based for several reasons. First, this is a typical value mentioned in the literature as noted in Section 2.3.1. Second, and more importantly, considering the SSVEP stimulus frequency band of 6 - 12Hz as mentioned in Section 5.1, and allowing for one reference signal harmonic in CCA-based decoding algorithms, the maximum theoretical frequency required is $f_{\text{max}} = 24$Hz. Incorporating a small margin for roll-off of the low-pass filter with $f_c = 26$Hz, sampling at $f_s > 2f_{\text{max}} = 56$Hz is required in order to satisfy the Nyquist sampling criterion and avoid aliasing. 64Hz was identified as an appropriate fit as it is a factor of 256Hz which facilitates decimation by an integer factor.

Figure 5.4 shows a high-level view of the sampling and filtering requirements of the system taking into account EEG and SSVEP dynamics, as well as restrictions specific to this project such as memory constraints.

[1] code implementation available here.

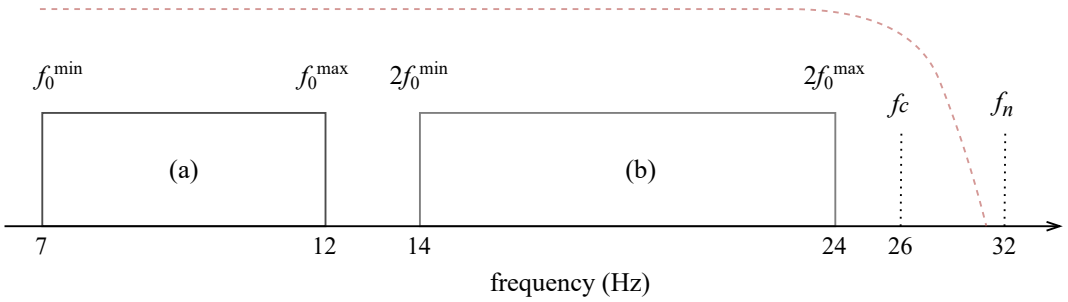


Figure 5.4: Diagram showing SSVEP stimulus frequency bands and other important frequencies. Band (a) represents the fundamental stimulus frequency range and (b) represents the range of first harmonics thereof. The dotted red line is an indicative (idealised) low-pass filter response with corner frequency at $f_c = 26$Hz. Including a small safety margin to allow for filter roll-off, f_n represents the Nyquist frequency for this configuration.

Digital filtering

As indicated in Figure 5.4, an ideal low-pass filter for this system would achieve zero pass-band distortion (ripple) between 7 and 24Hz and steep roll-off after $f_c = 26$Hz to achieve complete signal attenuation before the Nyquist frequency $f_n = 32$Hz. Obviously, this is not physically realisable and so a trade-off between maximising filter roll-off and minimising pass-band ripple must be sought.

Infinite impulse response (IIR) filters are typically more suitable for small, resource-constrained DSP systems. Compared to finite impulse response (FIR) filters, they generally offer:

- the ability to be implemented recursively
- greater computational efficiency
- lower memory requirements
- improved resolution at lower frequencies

Although FIR filters typically offer greater stability and controllability owing to only having zeros in their transfer functions, the aforementioned benefits of IIR filters were deemed to be more important for this application.

Figure 5.5 shows the frequency response of three different commonly used IIR digital filters: type I and II Chebyshev filters, as well as an elliptical filter. Each of these filters were designed to meet the following requirements:

- maximum filter order of $n = 10$
- maximum pass-band ripple (below unity gain) of $r_p = 0.2$dB
- minimum stop-band attenuation of $r_s = 80$dB

Observing the magnitude plot in the to half of Figure 5.5, it is evident that the response of the elliptic filter offers the optimal balance of steep roll-off between pass and stop bands and acceptable ripple in the stop and pass bands. This is intuitive: as r_p approaches zero, the elliptical filter becomes a Chebyshev type II filter. As r_s approaches zero, it becomes a Chebyshev type I filter. The phase response of the elliptical filter is also largely linear in the pass-band. For these reasons, a 10^{th} order elliptical low-pass filter satisfying the aforementioned design requirements was selected.

5.3 Embedded Firmware

As alluded to in Section 4.1.2, all firmware for this system was to be developed for the ESP32 SoC developed by Espressif Systems. Espressif offers a comprehensive open source framework called the Espressif IoT Development Framework (ESP-IDF) that can be used to develop ESP32-based applications on Windows, Linux and macOS using C and C++ programming languages. Furthermore, ESP-IDF tooling has been extended to be used with the popular open source platform Arduino. While the ESP-IDF has arguably become the de facto standard for developing complex embedded applications for the ESP32, a different approach was taken in this project for reasons explored below.

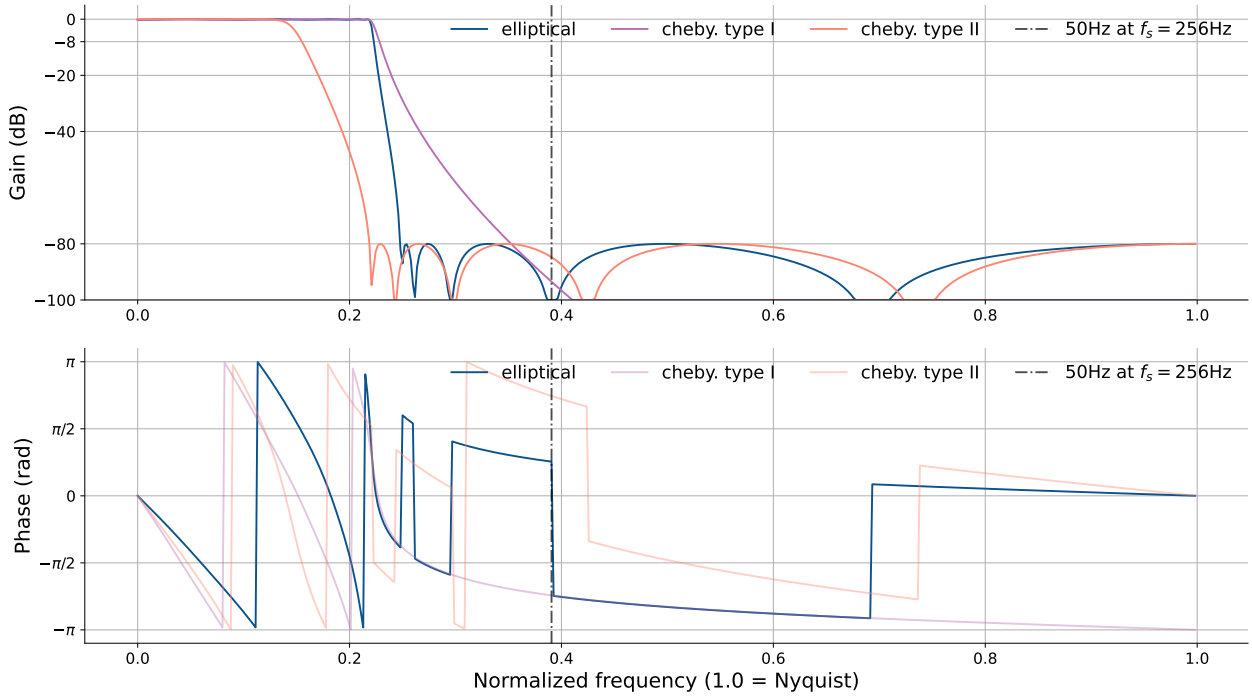


Figure 5.5: Frequency response of the digital low-pass filter implemented in firmware on the ESP32. The pre-filtered Nyquist frequency in this implementation is 128Hz: half the 256Hz sampling frequency.

5.3.1 MicroPython

MicroPython is a lean and highly efficient implementation of the Python 3 language that provides a carefully-selected subset of the Python standard library that is optimised for microcontrollers and other resource-constrained targets [40]. In fact, MicroPython is a full Python compiler and runtime implemented in C (C99) that runs natively (on bare metal) on several architectures. It is an open source project started by Damien George in 2013 that has since gained great traction from the open source community with constant feature updates and fixes. Some of its notable features include [40]:

1. compact enough to fit and run within 256k of ROM (code space) and 16k of RAM.
2. many different compile-time configuration options
3. support for many architectures (both for microcontrollers and fully-fledged microprocessors): x86, x86-64, ARM, ARM Thumb, Xtensa
4. inline assembler for Thumb and Xtensa instruction sets
5. a cross-compiler that allows ordinary Python scripts to be cross-compiled into bytecode that can be loaded from non-volatile flash instead of RAM.
6. explicit memory errors (`MemoryError`) for heap exhaustion
7. explicit run time errors (`RuntimeError`) when reaching the stack limit

The technical features of MicroPython listed above have some very useful practical implications for application development on the ESP32 that are not available when developing using the ESP-IDF in C/C++. Practical advantages of MicroPython include:

- The convenience of using the Python programming language which offers far more high-level functions and constructs over C or C++.
- A real-time interactive Python prompt (REPL) that allows commands to be tested and executed immediately over a serial connection. For some MicroPython ports, including that of the ESP32, a WebREPL is also provided that allows for interactivity over a wireless network connection.
- Support for both microcontroller and desktop architectures such as X86 allows MicroPython code to be tested with exact equivalence on a personal computer without having to interact with a microcontroller. This can be very useful for prototyping and algorithm development.

- A very familiar and intuitive micro-directory and module structure that emulates regular Python modules (explored more below).
- A community of open source developers that can very easily contribute new modules and peripheral functionality

5.3.2 Module structure

As alluded to above, a very convenient feature of MicroPython is its intuitive module/file structure. It allows python modules in the form of `.py` files to be organised in folders that serve as packages as with regular Python. This file structure is stored directly in flash storage on the device and can be read or written to at any time during development. Consider the module structure of the MicroPython application developed in this project below:

```
/
├── boot.py
├── main.py
├── .env
├── lib
│   ├── core.py
│   ├── decoding.py
│   ├── peripherals.py
│   └── ...
├── certificates
│   ├── dab0ac2b5c-private.pem.key
│   ├── dab0ac2b5c-public.pem.key
│   └── dab0ac2b5c-certificate.pem.crt
└── logs
    ├── log-17082021-1.json
    └── log-10072021.txt
```

The root directory contains two important files - `boot.py` and `main.py` - that are run in sequence at boot time. The `boot.py` script is typically used for internal system processes and `main.py` is used for any user functionality that should automatically be run at boot. Also in the root directory is the `.env` file which can be used to store sensitive environment variables such as Wi-Fi passwords or API key strings. The `lib` directory (package) contains the modules that implement the actual functionality of the digital system, including peripheral management, decoding and computation and networking (note that not all modules are shown above). Amazon Web Services (AWS) key files and certificates for secure MQTT communication are stored in the `certificates` folder. Finally, a `logs` directory shows how log files can also be very conveniently stored in arbitrary formats such as `.json` or `.txt` in non-volatile flash memory. Listing 1 below demonstrates how user-specific modules and files can be imported and used within a new script once flashed to the microcontroller's non-volatile memory.

```
# import user-specific functions from non-volatile memory
from lib.decoding import cca
from lib.computation import solve_gen_eig_prob as solve_eig

# MicroPython emulation of the standard Python `os` module
import os

# list root directory
print(os.listdir('/'))

# open and read a file from flash
with open('/logs/log-10072021.txt') as f:
    print(f.read())
```

Listing 1: Basic MicroPython code to import user-specific modules and read from a text file in non-volatile storage. Note that the syntax is identical to ordinary Python.

5.3.3 Numerical computation

An additional important motivation for using MicroPython over the more traditional C/C++ approach is the ability to use an extremely convenient open source MicroPython module called `ulab`[2]. The `ulab` project was created to offer a subset of the very popular NumPy numerical computing library for Python which offers highly performant array computing and linear algebra functionality. It also implements a small subset of functionality offered by SciPy; the equally popular scientific computing library for Python. Some of the most notable features offered by this module include [36]:

1. compact, iterable and sliceable array structures for numerical data up to 4 dimensions
2. extremely frugal with RAM usage
3. efficient, vectorised computations on multi-dimensional arrays
4. basic linear algebra functionality such as matrix inversion, multiplication, determinants, Cholesky and QR decomposition
5. fast Fourier transforms (FFTs)
6. implementation of digital filters using second order sections (SOS)
7. basic numerical approximation and function minimisation

Being able to perform matrix and array operations, filtering and numerical approximation on a microcontroller in an elegant, Pythonic way is clearly an invaluable asset for this project. Indeed, for any project of this sort, it is somewhat amazing. Listing 2 briefly demonstrates just how effortlessly complex numerical computations can be performed using `ulab`.

```
# import ulab numpy module
from ulab import numpy as np

# create an arbitrary positive definite, symmetric 3x3 matrix
# A can be sliced like A[i0:i1, j0:j1]
A = np.array([[25, 15, -5], [15, 18,  0], [-5,  0, 11]])

# compute lower triangular square root or A using Cholesky decomp
A_sqrt = np.linalg.cholesky(A)

# compute determinant of A
det_A = np.linalg.det(A)

# compute inverse of A
A_inv = np.linalg.inv(A)

# compute Moore-Penrose pseudoinverse of A = (A^T.A)^-1.A^T
A_pinv = np.dot(np.linalg.inv(np.dot(np.transpose(A), A)), np.transpose(A))

# many other utility functions such as argmax(), argsort(), convolve()
```

Listing 2: Illustration of the convenience offered by the `ulab` module for linear algebra and general numerical computing.

5.3.4 Networking

Once processed and decoded, data from many BCI devices must be streamed to a central cloud service. The design of this cloud service was beyond the scope of this project. However, it was specified that the Amazon Web Services (AWS) IoT Core service was to be used. This service allows IoT devices to securely exchange small data payloads over the Internet using the MQTT protocol. MQTT is a lightweight, publish-subscribe protocol that has become commonplace in the realm of IoT communication. As mentioned in Section 4.1.2, the ESP32 SoC has integrated Wi-Fi functionality. MicroPython includes generic socket functionality that can be used to implement an MQTT client for interfacing with AWS.

As seen in Figure 5.6, the MQTT protocol defines a message broker (server) and several clients. In this case, the AWS IoT service acts as an MQTT broker that receives messages published by remote ESP32 BCI clients.

[2]Source code and documentation available here.

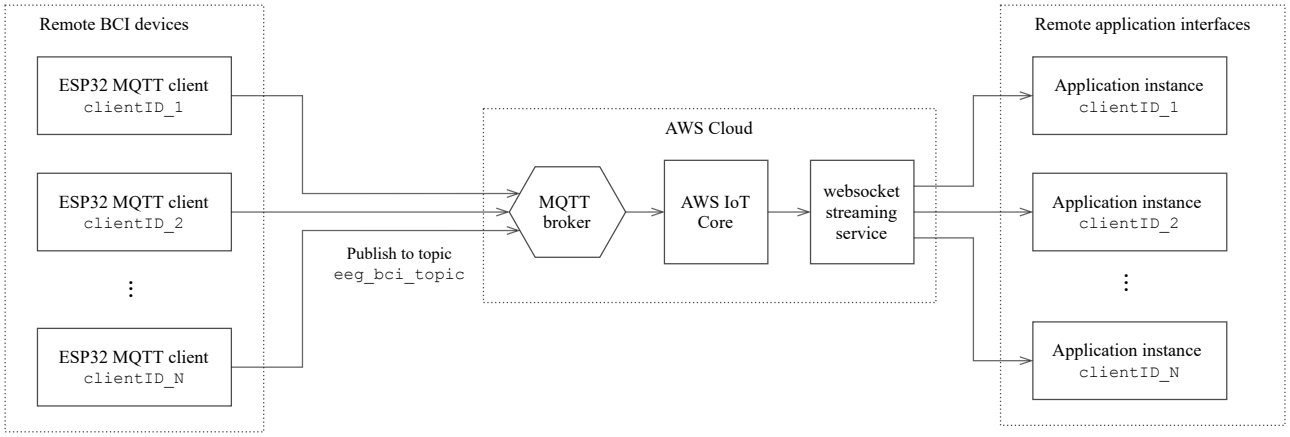


Figure 5.6: Diagram showing the client-broker MQTT interface for relaying data captured by remote BCI devices to an AWS cloud service and thereafter, to a web application for visualisation.

Each of the clients have a unique client ID but subscribe to a common topic that the AWS service expects. Although the design of subsequent elements in the AWS cloud pipeline was not included in this project, a schematic representation is presented in Figure 5.6: data received from BCI clients over MQTT is streamed to a web application which displays EEG data in real time.

5.3.5 Logging

Data logging from the ESP32-based BCI hardware was crucial for this project. This enabled the gathering of experimental data, but also various signals and messages for debugging during development. By leveraging the built-in WiFi functionality on the ESP32, a basic wireless logging system was created.

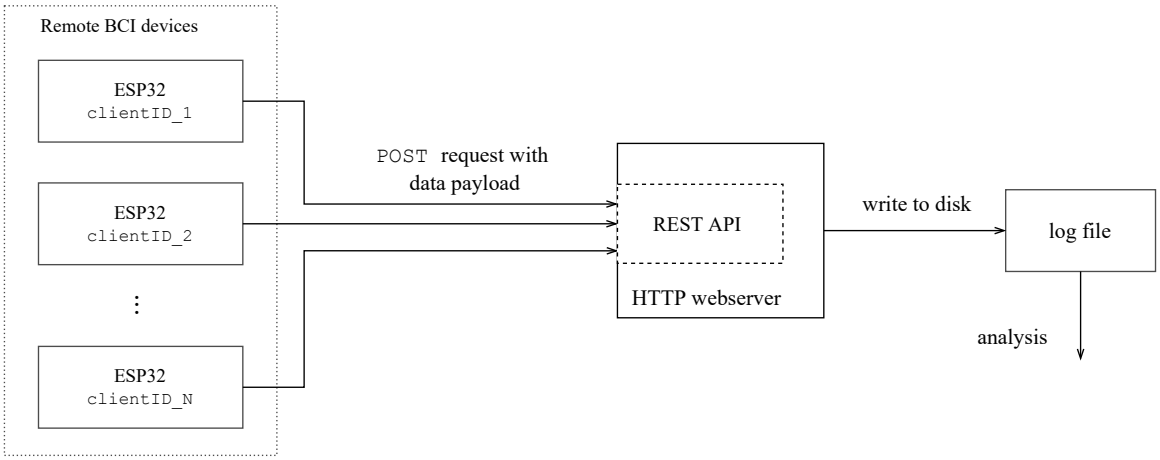


Figure 5.7: Diagram showing the web logger interface

Figure 5.7 provides an overview of the web-based logging system. Potentially many ESP32 devices collect data and periodically log data to a web server. Typically, a REST API (application progamming interface) is deployed on the server which accepts requests from clients and processes and stores request data. In this project, a basic REST API[3] was created using Flask, an open source web micro-framework written in Python.

The JSON object in Listing 3 shows an example of a data payload sent via an ESP32 client to the logging API. Decoded data, as well as raw data used to perform on-board decoding, is compiled into each data payload. Each payload is then sent via an HTTP `POST` request to the logging server. Once received and verified by the API, data is written to a log file stored on the server. Note that in this project, the server and API were run on a local network and data was thus stored on a personal computer. However, the system would work identically if the server was deployed in the cloud. This would allow data to be logged by remote BCI devices from anywhere in the world and would not require users of the devices to run their own local server. This could facilitate a decentralised data acquisition system that could enable wide-scale data collection.

[3]source code available here.

```
{
    "client_id": "esp1", # used to identify which device data was received from
    "timestamp": 1630235886, # used for synchronisation and periodicity monitoring
    "trial_id": "test_123", # used to group trials
    "decoded_data": { # dict of stim. freqs and their decoded outputs
        "7": 0.12,
        "10": 0.56,
        "12": 0.04
    },
    "raw_data": [[ 0.02859, 0.054, ..., 0.345], # raw data for further offline analysis
                 [ 0.412, 0.001, ..., 0.345]]
}
```

Listing 3: Example JSON data payload sent from a remote ESP32 device via `POST` HTTP request to the logging server

5.3.6 Digital filtering

The digital low-pass filter described in Section 5.2.1 whose frequency response is shown in Figure 5.5 was designed using SciPy. In particular, given the filter design requirements, the `scipy.signal.ellip()` function was used to generate filter coefficients. Using the `output=='sos'` argument, the filter coefficients were computed in a cascaded second-order sections (SOS) representation. The SOS representation of an IIR filter with digital transfer function $H(z)$ is as follows:

$$H(z) = K \prod_{n=0}^{N-1} \frac{b_{n,0} + b_{n,1}z^{-1} + b_{n,2}z^{-2}}{1 + a_{n,1}z^{-1} + a_{n,2}z^{-2}}, \tag{5.1}$$

where K is a gain scalar, N is the number of sections and a_n and b_n are the reverse and forward coefficients for section n respectively. Notice that in the form in (5.1), only a highest order of 2 is present for any component. This cascaded lower-order form offers greater numerical stability and less sensitivity to quantisation of coefficients than standard higher order polynomials.

The filter coefficients, once computed offline, were stored as constants in the the ESP32 firmware (in particular, in the `lib.signal` module). As there was no need to recompute or otherwise modify these coefficients, they were stored in read-only flash memory to persist across power cycles and save space in RAM. In order to run digital filtering on the ESP32 itself, the convenient `scipy.signal.sosfilt()` from the `ulab` module was used. This function simply takes the pre-computed SOS coefficients and the input signal and returns the filtered version. Tests were performed to ensure equivalence between this implementation and the standard implementation in the fully-fledged SciPy module.

5.4 Algorithm Implementation

5.4.1 Eigenvalue algorithms

As elaborated on in Chapter 3, the decoding algorithms explored in this project all resolve to constrained optimisation problems that can be reformulated as eigenvalue problems. Consequently, an 'eigen' function capable of finding the eigenvalues (and corresponding eigenvectors) of an arbitrary matrix was crucial in the design of this system. The `ulab` module conveniently includes a basic eigen function of this sort: `numpy.linalg.eig()`. However, the severe limitation with this implementation is that it only works with *symmetric* matrices (i.e. matrices with real eigenvalues). For signal decoding, many of the eigenvalue problems arising in this project involve asymmetric matrices. Thus, an alternative solution needed to be sought.

After much experimentation and difficulty finding an analytical solution that could work[4] under all conditions, it was decided that *iterative* eigenvalue algorithms should be explored.

Power Iteration

The simple but elegant power iteration algorithm explored in Section 3.1.3 can easily be implemented in firmware. The pseudocode for this algorithm is provided in Algorithm 1. The corresponding MicroPython

[4] produce the correct result and maintain numerical stability

implementation is documented in Listing 4.

Algorithm 1 Power iteration

Pick an initial vector $\mathbf{v}^{(0)}$ with $\|\mathbf{v}^{(0)}\| = 1$ ▷ Initial unit vector assigned randomly
for $k = 1, 2, \ldots, K$ **do** ▷ repeat for K iterations
  $\mathbf{w} \leftarrow \mathbf{w} = \mathrm{A}\mathbf{v}^{(k-1)}$ ▷ update eigenvector estimate
  $\mathbf{v}^{(k)} \leftarrow \mathbf{w}/\|\mathbf{w}\|$ ▷ normalise new estimate
end for

Simultaneous Iteration

As discussed in Section 3.1.4, the Simultaneous Iteration algorithm provides an effective extension of Power Iteration by solving for all eigenvectors simultaneously. The pseudocode for the implementation of this algorithm is given in Algorithm 2.

Algorithm 2 Simultaneous iteration

Pick an initial basis for $\{\mathbf{v}^{(0)}, \ldots, \mathbf{v}^{(d)}\}$ for $\mathbb{R}^d$ ▷ initialise set of random basis vectors
Construct the matrix $\mathrm{V} = \left[\begin{array}{c|c|c} \mathbf{v}_1^{(0)} & \ldots & \mathbf{v}_d^{(0)} \end{array} \right]$
Obtain the factors $\mathrm{Q}^{(0)}\mathrm{R}^{(0)} = \mathrm{V}^{(0)}$ ▷ compute QR factorisation of $\mathrm{V}^{(0)}$
for $k = 1, 2, \ldots, K$ **do**
  $\mathrm{W} \leftarrow \mathrm{W} = \mathrm{A}\mathrm{Q}^{(k-1)}$ ▷ update step
  Obtain the factors $\mathrm{Q}^{(k)}\mathrm{R}^{(k)} = \mathrm{W}$ ▷ extract orthonormal column vectors of W from $\mathrm{Q}^{(k)}$
end for

QR Iteration

Finally, as revealed in Section 3.1.5, the QR Iteration algorithm is a more convenient form of the Simultaneous Iteration algorithm that is more commonly used in practice. Pseudocode for its implementation in this project is given in Algorithm 3. The MicroPython implementation of this algorithm is presented in Listing 5 in

Algorithm 3 QR iteration

$\mathrm{A}^{(0)} \leftarrow \mathrm{A}$
for $k = 1, 2, \ldots, K$ **do**
  Obtain the factors $\mathrm{Q}^{(k)}\mathrm{R}^{(k)} = \mathrm{A}^{(k-1)}$ ▷ perform QR factorisation on $\mathrm{A}^{(k-1)}$
  $\mathrm{A}^{(k)} \leftarrow \mathrm{R}^{(k)}\mathrm{Q}^{(k)}$ ▷ update step detailed in [9]
end for

Appendix A.1.

Generalised eigenvalue algorithm

The generalised eigenvalue problem for two symmetric matrices A, B $\in \mathbb{R}^{d \times d}$ is given by the form

$$\mathrm{A}\mathbf{x}_i = \lambda_i \mathrm{B}\mathbf{x}_i, \quad \forall i \in \{1, \ldots, d\}, \tag{5.2}$$

where $\mathbf{x}_i$ and λ_i are the eigenvectors and eigenvalues of the system respectively. This problem is relevant to several decoding algorithms explored in this project, including the GCCA and MsetCCA algorithms explored in Chapter 3. The proof for solving for $\mathbf{x}_i$ and λ_i is involved but well known and is given in [24]. However, understanding the algorithm to solve this problem is crucial for enabling implementation in firmware. The generalised eigenvalue algorithm, extracted from [24], is given in Algorithm 4 below.

The MicroPython implementation of the generalised eigenvalue algorithm can be found in Listing 6 in Appendix A.1.

Algorithm 4 Generalised eigenvalue algorithm

$\Phi_B, \Lambda_B \leftarrow B\Phi_B = \Phi_B\Lambda_B$ ▷ compute eigenvectors Φ_B and eigenvalues Λ_B of B
$\widetilde{\Phi}_B \leftarrow \widetilde{\Phi}_B = \Phi_B\Lambda_B^{-1/2} \approx \Phi_B\left(\Lambda_B^{1/2} + \varepsilon I\right)^{-1}$ ▷ multiply above by $\Lambda_B^{-1/2}$, define intermediate $\widetilde{\Phi}_B$
$\widetilde{A} \leftarrow \widetilde{A} = \widetilde{\Phi}_B^\top A \widetilde{\Phi}_B$ ▷ define intermediate $\widetilde{A}$
$\Phi_A, \Lambda_A \leftarrow \widetilde{A}\Phi_A = \Phi_A\Lambda_A$ ▷ compute eigenvectors Φ_A and eigenvalues Λ_A of $\widetilde{A}$
$\Lambda \leftarrow \Lambda = \Lambda_A$ ▷ notice that eigenvalues of $\widetilde{A}$ are the generalised eigenvalues
$\Phi \leftarrow \Phi = \widetilde{\Phi}_B\Phi_A$ ▷ compute generalised eigenvectors
return Φ, Λ ▷ return the generalised eigenvector and eigenvalue matrices

5.4.2 Decoding algorithms

CCA

The implementation of the CCA algorithm introduced in Section 2.4.2 is demonstrated in Algorithm 5. The ε argument is used for numerical conditioning to ensure matrices C_{XX}, C_{YY} are not singular prior to inversion. Typical values are in the range of $\varepsilon \approx 10^{-7}$. In the case of CCA for SSVEP decoding as in this project, the reference signal argument Y would be the harmonic reference signal as specified in Section 2.4.2.

Algorithm 5 CCA algorithm

function ComputeCCA(X, Y, ε) ▷ Input signal $X \in \mathbb{R}^{N_c \times N_s}$ and ref. signal $Y \in \mathbb{R}^{q \times N_s}$, $q \in \mathbb{N}$
  $C_{XX} \leftarrow C_{XX} = XX^\top$ ▷ Signal covariance matrix
  $C_{YY} \leftarrow C_{YY} = YY^\top$ ▷ Reference covariance matrix
  $C_{XY} \leftarrow C_{XY} = XY^\top$ ▷ Cross covariance matrix
  $M \leftarrow M = (C_{XX} + \varepsilon)^{-1} C_{XY} (C_{YY} + \varepsilon)^{-1} C_{XY}^\top$ ▷ Store intermediate result in M
  $\lambda^* \leftarrow M\mathbf{w} = \lambda\mathbf{w}$ ▷ Find maximum eigenvalue λ^* of M using Algorithm 3
  return $\sqrt{\lambda^*}$ ▷ Maximum canonical corr. is square root of λ^*
end function

The MicroPython implementation of this algorithm is presented in Listing 7 in Appendix A.1.

GCCA

The implementation of the GCCA algorithm introduced in Section 2.4.2 is demonstrated in Algorithm 6. As this is a template-based algorithm that requires calibration or training data prior to inference, there are two distinct functions: FitGCCA() and ComputeGCCA(). The former must be executed before inference using the ComputeGCCA() function can begin. The Python implementation of this algorithm is presented in Listing 9 in Appendix A.1.

MsetCCA

The implementation of the MsetCCA algorithm introduced in Section 2.4.2 is demonstrated in Algorithm 7. As with the GCCA algorithm implementation in Section 5.4.2, the FitMCCA() and ComputeMCCA() functions must be executed in sequence. That is, inference requires the optimised reference signal Y that is computed by FitMCCA(). Note that, for ease of explanation, the single-channel case $N_c = 1$ (as used in this project) has been documented here.

The MicroPython implementation of this algorithm is presented in Listing 8 in Appendix A.1.

Algorithm 6 GCCA algorithm

function FitGCCA($\boldsymbol{\mathcal{X}}$) ▷ Input training/calibration data tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N_c \times N_s \times N_t}$
  $\mathrm{X}^c \leftarrow \mathrm{X}^c = \begin{bmatrix} \mathrm{X}_1^\top & \mathrm{X}_2^\top & \dots & \mathrm{X}_{N_t}^\top \end{bmatrix}$ ▷ Form $\mathrm{X}^c \in \mathbb{R}^{N_c \times (N_s N_t)}$ by concatenating trials X_i of $\boldsymbol{\mathcal{X}}$
  $\overline{\mathrm{X}} \leftarrow \overline{\mathrm{X}} = \frac{1}{N_t} \sum_{i=1}^{N_t} \boldsymbol{\mathcal{X}}_i$ ▷ Form signal template by averaging data across trials X_i, $\forall i$
  $\overline{\mathrm{X}}^c = \begin{bmatrix} \overline{\mathrm{X}}^\top & \overline{\mathrm{X}}^\top & \dots & \overline{\mathrm{X}}^\top \end{bmatrix}$ ▷ Form concat. template $\overline{\mathrm{X}}^c \in \mathbb{R}^{N_c \times (N_s N_t)}$ to match dim. of X^c
  generate Y ▷ Form harmonic reference set $\mathrm{Y} \in \mathbb{R}^{2N_h \times N_s}$
  $\mathrm{Y}^c \leftarrow \mathrm{Y}^c = \begin{bmatrix} \mathrm{Y}^\top & \mathrm{Y}^\top & \dots & \mathrm{Y}^\top \end{bmatrix}$ ▷ Form concat. reference $\mathrm{Y}^c \in \mathbb{R}^{2N_h \times (N_s N_t)}$ to match dim. of X^c
  $\widetilde{\mathbf{w}} \leftarrow \widetilde{\mathbf{w}} = \begin{bmatrix} \mathbf{w}_{\mathrm{X}^c} & \mathbf{w}_{\overline{\mathrm{X}}^c} & \mathbf{w}_{\mathrm{Y}^c} \end{bmatrix}^\top$ ▷ Form augmented spatial filter vector
  $\widetilde{\mathrm{X}} = \widetilde{\mathrm{X}} = \begin{bmatrix} (\mathrm{X}^c)^\top & (\overline{\mathrm{X}}^c)^\top & (\mathrm{Y}^c)^\top \end{bmatrix}^\top$ ▷ Form augmented signal matrix
  $\mathrm{D} \leftarrow \mathrm{D} = \mathrm{diag}\left(\mathrm{X}^c(\mathrm{X}^c)^\top,\ \overline{\mathrm{X}}^c(\overline{\mathrm{X}}^c)^\top,\ \mathrm{Y}^c(\mathrm{Y}^c)^\top \right)$ ▷ Form intermediate block-diagonal matrix D
  $\widetilde{\mathbf{w}}^*, \lambda^* \leftarrow \widetilde{\mathrm{X}}\widetilde{\mathrm{X}}^\top \widetilde{\mathbf{w}} = \lambda \mathrm{D} \widetilde{\mathbf{w}}$ ▷ Find eigen pair with max eigenvalue λ^* (Algorithm 4)
  store $\widetilde{\mathbf{w}}^*$, $\overline{\mathrm{X}}$, Y ▷ Store optimised spatial filter (eigenvector) $\widetilde{\mathbf{w}}^* \in \mathbb{R}^{2(N_c+N_h)}$
end function

function ComputeGCCA(X_{test}) ▷ Test data $\mathrm{X} \in \mathbb{R}^{N_c \times N_s}$
  $\mathbf{w}_{\mathrm{X}^c}$, $\mathbf{w}_{\overline{\mathrm{X}}^c}$, $\mathbf{w}_{\mathrm{Y}^c} \leftarrow \widetilde{\mathbf{w}}^*$ ▷ Extract component filters from optimised $\widetilde{\mathbf{w}}^*$
  $\rho_1 \leftarrow \rho_1 = \mathrm{corr}(\mathrm{X}_{\text{test}}\mathbf{w}_{\mathrm{X}^c},\ \overline{\mathrm{X}}\mathbf{w}_{\overline{\mathrm{X}}^c})$ ▷ Canonical corr. between X_{test} and historical template
  $\rho_2 \leftarrow \rho_2 = \mathrm{corr}(\mathrm{X}_{\text{test}}\mathbf{w}_{\mathrm{X}^c},\ \mathrm{Y}\mathbf{w}_{\mathrm{Y}^c})$ ▷ Canonical corr. between X_{test} and harmonic reference
  $\rho \leftarrow \rho = \mathrm{sign}(\rho_1)\rho_1^2 + \mathrm{sign}(\rho_2)\rho_2^2$ ▷ Combined output correlation
  return ρ
end function

Algorithm 7 MsetCCA algorithm (single channel, $N_c = 1$)

function FitMCCA(X) ▷ Input training/calibration data matrix $\mathrm{X} \in \mathbb{R}^{N_t \times N_s}$
  $\mathrm{R} \leftarrow \mathrm{R} = \mathrm{X}\mathrm{X}^\top$ ▷ Compute inter-trial covariance matrix
  $\mathrm{S} \leftarrow \mathrm{S} = \mathrm{diag}(\mathrm{R})$ ▷ Construct intra-trial covariance matrix using diagonal entries of R
  $\mathbf{w}^*, \lambda^* \leftarrow (\mathrm{R} - \mathrm{S})\mathbf{w}_i = \lambda_i \mathrm{S}\mathbf{w}_i, \quad \forall i \in \{1, \dots, N_t\}$ ▷ Find eig. pair with max eig. val λ^* (Algorithm 4)
  $\mathrm{Y}^* \leftarrow \mathrm{Y} = \begin{bmatrix} w_1^*\mathbf{x}_1 & w_2^*\mathbf{x}_2 & \dots & w_{N_t}^*\mathbf{x}_{N_t} \end{bmatrix}$ ▷ Compute opt. reference $\mathrm{Y} \in \mathbb{R}^{N_t \times N_s}$ with $\mathbf{x}_i$ as rows of X.
  store Y^*
end function

function ComputeMCCA(X_{test}) ▷ Test data $\mathrm{X} \in \mathbb{R}^{N_c \times N_s}$, $N_c = 1$
  $\rho \leftarrow$ ComputeCCA(X_{test}, Y^*) ▷ Compute canonical corr. using CCA with optimised ref. Y^*
  return ρ
end function

Chapter 6

Results

6.1 Hardware Verification and Testing

This section details the results of the various verification tests mentioned in Section 4.2.2. The purpose of these tests was to verify that all core components of the designed system performed as expected. After basic tests to verify firmware integrity and communication with the ESP32-based target board passed, the digital signal processing (DSP) system was investigated. The results of important diagnostic tests are detailed below. Throughout this section, 'the hardware' refers to the electronic hardware introduced in Section 4.1.2 and pictured in Figure 4.2.

6.1.1 DSP system

As explained in Section 5.3, the DSP system is crucial for preparing analogue signals acquired from hardware prior to decoding. In order to test several components of this system simultaneously, a square wave signal $x[n]$ at fundamental frequency $f_x^{(0)} = 12\text{Hz}$ was fed to the ADC of the ESP32. Sampling was set to $f_s = 256\text{Hz}$ and where applicable, downsampling to $f_s' = 64\text{Hz}$ was used. Owing to the lack of access to a signal generator or other laboratory equipment, the input signal $x[n]$ was generated using a hardware timer on board the ESP32. This experiment was designed to test the following components of the DSP system as follows:

1. **sampling**: a basic sanity check to verify that a square wave at 12Hz was indeed measured by the ADC. This also offered a means of inspecting the consistency (both in amplitude and frequency) of the measured signal.
2. **filtering**: the input signal $x[n]$ was filtered by the onboard digital low-pass filter described in Section 5.2.1 to yield $y[n]$. Given the filter's corner frequency of $f_c = 26\text{Hz}$, $y[n]$ should only contain the fundamental frequency $f_x^{(0)} = 12\text{Hz}$ since higher order harmonics of the square wave input at $\{f_x^{(k)} | f_x^{(k)} = (2k-1)f_x^{(0)},\ k = 2, 3, \dots\} = \{36\text{Hz}, 60\text{Hz}, \dots\}$ should be largely attenuated. Furthermore, the measured spectrum of $y[n]$ offers a way to compare the actual and designed behaviour of the filter.
3. **downsampling**: a simple check to confirm that, given all prior design decisions, downsampling does not distort the originally measured signal excessively in the frequency band of interest (around $7 - 12\text{Hz}$). With low-pass filtering in place, there should be no aliasing or artefacts introduced.

All results reported below were computed on the hardware itself, i.e.: all DSP operations (sampling, filtering, resampling) were computed on-device as would happen in a real application. Figures were then plotted using `Matplotlib`, an open source Python library.

Figure 6.1 shows the filtered signal $y[n]$ against the measured input square wave $x[n]$. From visual inspection, $x[n]$ appears to have very consistent frequency and amplitude as desired. $y[n]$ shows marginal phase lag and slight inconsistencies in amplitude compared to the ideal output $y^*[n] = \frac{4}{\pi}\sin(2\pi f_x^{(0)} n)$. However, these slight deviations are certainly tolerable and would make little to no difference to the success of subsequent decoding algorithms.

Figure 6.2 shows PSD estimates $\hat{P}_x(\omega)$ and $\hat{P}_y(\omega)$ of $x[n]$ and $y[n]$ respectively in the form of periodograms. Confirming the time domain representation in Figure 6.1, $\hat{P}_x(\omega)$ shows power peaks at the odd numbered harmonics of $x[n]$. This can be explained by the Fourier series expansion of an ideal square wave at fundamental

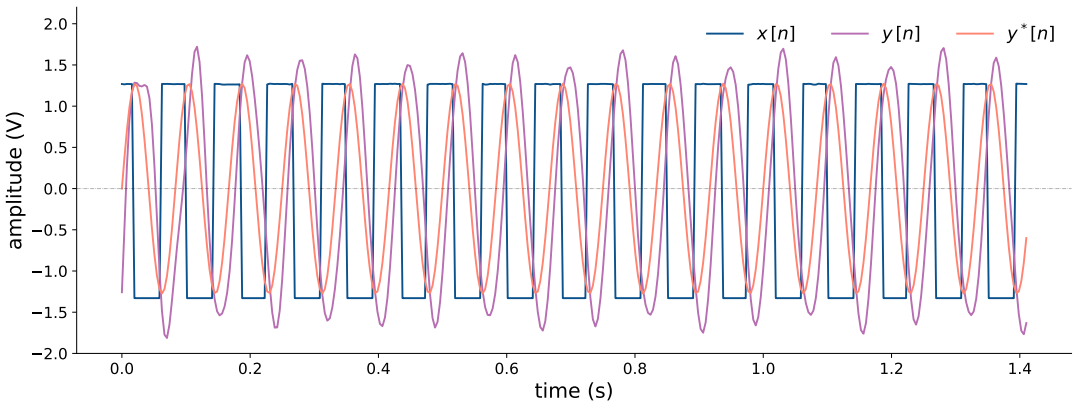


Figure 6.1: Time domain plot showing the measured square wave signal $x[n]$ together with the digitally-filtered output signal $y[n]$ designed to retain only the fundamental frequency $f_x^{(0)}$ of $x[n]$. The ideal output signal $y^*[n]$ represents a sinsuoid at frequency $f_x^{(0)}$: $y^*[n] = \frac{4}{\pi}\sin(2\pi f_x^{(0)} n)$.

frequency $f^{(0)}$ with 50% duty cycle:

$$x[n] = \frac{4}{\pi}\sum_{k=1}^{\infty}\frac{\sin(2\pi(2k-1)f^{(0)}n)}{2k-1}, \tag{6.1}$$

which is nothing but an infinite sum of sinusoids whose amplitudes decay with frequency. Notice that only odd-numbered harmonics in (6.1) are non-zero. Therefore, the ideal spectrum $P_x(\omega)$ of $x[n]$ should be an impulse train at frequencies $(2k-1)f_x^{(0)}$, $k \geq 2$, $k \in \mathbb{Z}$.

While not quite impulses, power peaks of $\hat{P}_x(\omega)$ in Figure 6.2 are clearly visible at the odd-numbered harmonics at 26Hz, 60Hz and so on. The spectrum of the filtered signal, $\hat{P}_y(\omega)$, shows very little distortion in the pass-band between 0 and $f_c = 26$Hz, as well as impressively steep roll-off for $f > f_c$. As a result, only the fundamental frequency of $x[n]$ is captured in $y[n]$, as desired. Attenuation of higher frequency harmonics in $x[n]$ by the digital filter is evidently very effective and meets the design criteria stated in Section 5.2.1: stop-band attenuation is approximately 80dB, as can be seen in the power difference between $\hat{P}_x(\omega)$ and $\hat{P}_y(\omega)$ at 60Hz, for example.

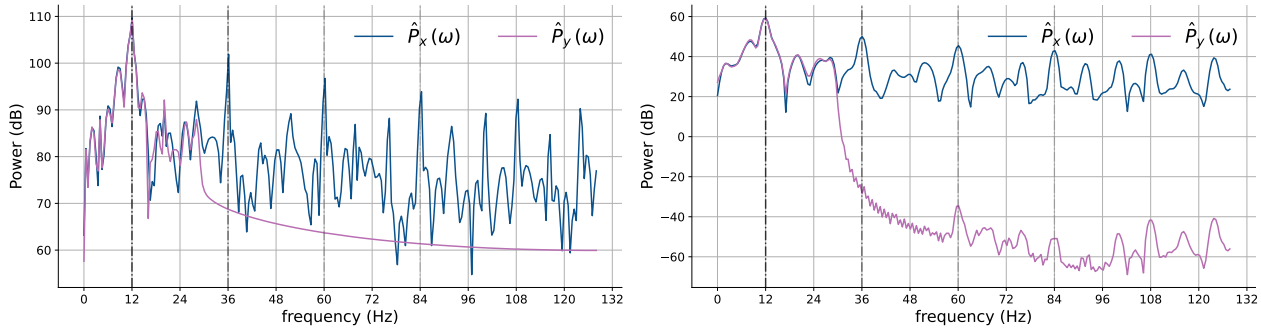


Figure 6.2: Standard periodogram (left) and Welch-averaged periodogram (right) representing PSD estimates of measured input signal $x[n]$ and filtered output $y[n]$. Dashed vertical lines mark $f_x^{(0)}$ and odd-numbered harmonics of $x[n]$.

Along with $\hat{P}_x(\omega)$ as before, Figure 6.3 shows the estimated spectra $\hat{P}_z(\omega)$ and $\hat{P}_{\text{alias}}(\omega)$, where $\hat{P}_z(\omega)$ corresponds to $z[n]$, the downsampled version of the filtered output $y[n]$ and $\hat{P}_{\text{alias}}(\omega)$ is the estimated spectrum of $x[n]$ after being downsampled with *no* prior filtering. Both resampled signals were downsampled by a factor of 4 to $f_s' = 64$Hz. As is particularly evident from the artefacts at 4Hz, 20Hz and 27Hz in the Welch-

averaged periodogram in Figure 6.3, aliasing has occurred in $\hat{P}_{\text{alias}}(\omega)$. The full spectrum of $\hat{P}_x(\omega)$ in Figure 6.2 explains this behaviour: $x[n]$ contains substantial energy at higher frequencies past $f > f_n' = 32$Hz where f_n' is the Nyquist frequency for the downsampled rate of $f_n' = 64$Hz. This demonstrates the necessity for low-pass filtering to isolate the frequency band of interest before downsampling. On the other hand, Figure

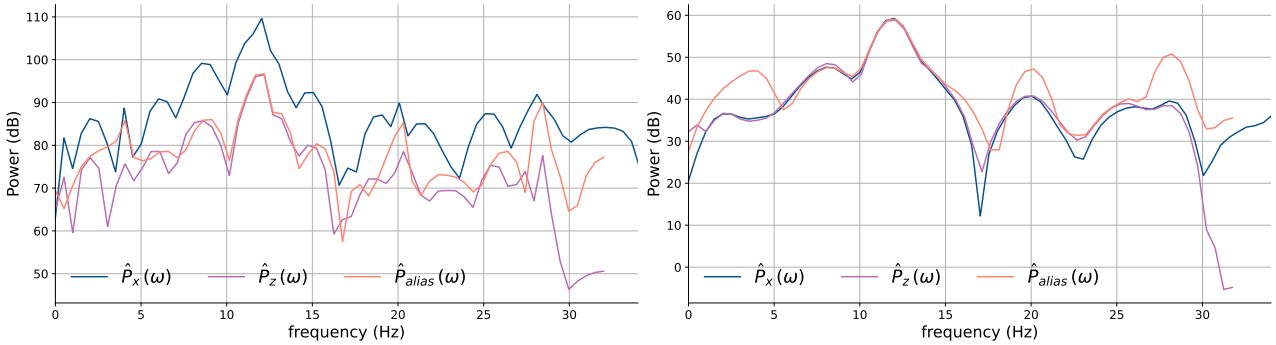
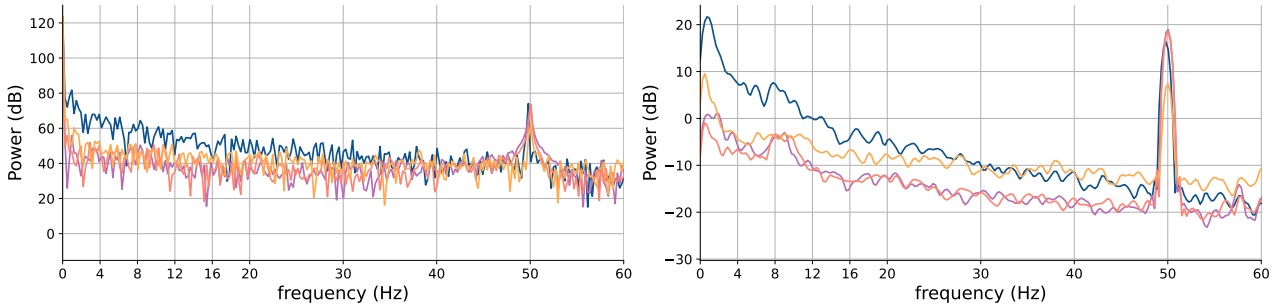


Figure 6.3: Standard periodogram (left) and Welch-averaged periodogram (right) representing PSD estimates $\hat{P}_x(\omega)$ and $\hat{P}_z(\omega)$ of input $x[n]$ and filtered, downsampled output $z[n]$ respectively. $\hat{P}_{\text{alias}}(\omega)$ shows the estimated spectrum of a downsampled version of $x[n]$ *without* prior low-pass filtering

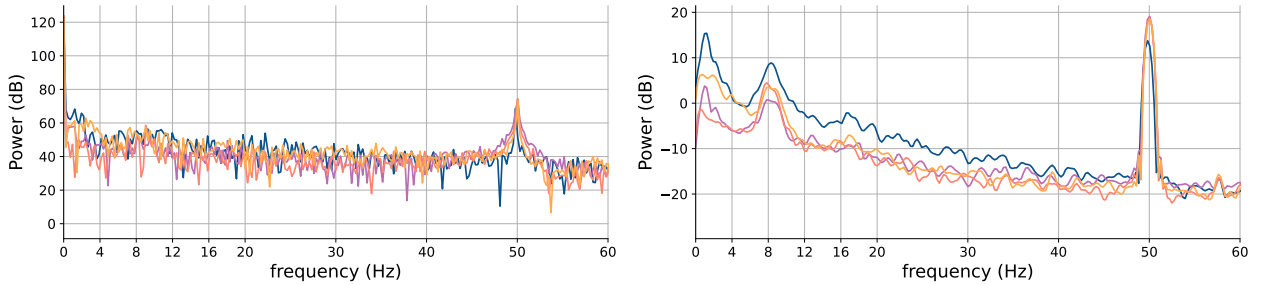
.

6.3 also demonstrates the effectiveness of the DSP system as a whole: with suitable low-pass filtering, the spectrum of the downsampled signal $z[n]$ in the pass-band between 0 and 24Hz agrees very closely with that of the original signal $x[n]$.

6.1.2 Hardware and data acquisition



(a) eyes open



(b) eyes closed

Figure 6.4: **Alpha band test**: periodograms showing $N = 1024$ point PSD estimates for EEG signals measured from a subject in two distinct states: with eyes open and eyes closed. Data was acquired using the early hardware prototype in Figure 4.4. Attention should be given to signal power around $8 - 10$Hz (alpha band). In both (a) and (b), the left subplot shows a standard, non-windowed periodogram accompanied by a Welch-averaged periodogram to the right. Different coloured traces represent independent trials.

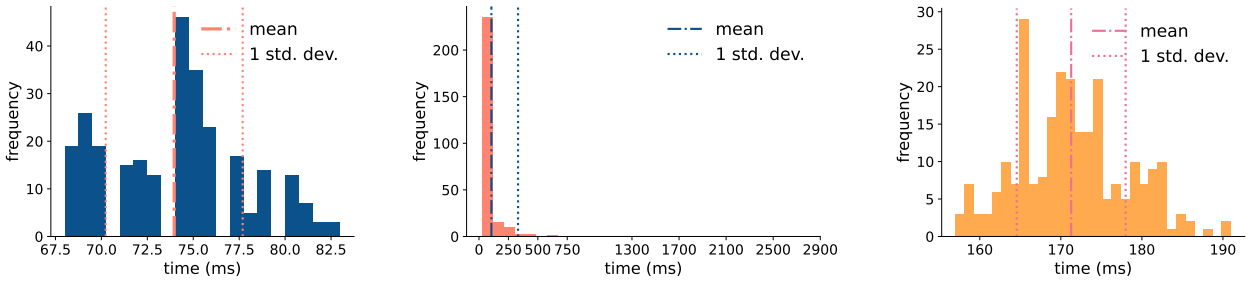
As mentioned in Chapter 4, the rudimentary headband discussed in Section 4.1.2 was created as a means of acquiring and testing real-life data from the electronic hardware prototype introduced in Section 4.1.2. After

verifying the fidelity of the DSP system, a basic BCI test known as the 'alpha test' was performed. This involves measuring the EEG signals generated by the brain's visual cortex during two distinct states: with eyes open and eyes closed. In particular, signal energy in the alpha band between 8-12Hz is observed. As mentioned briefly in Section 2.2.2, a working BCI should be able to discern greater energy in the alpha band when a subject's eyes are closed compared to when open.

Figure 6.4 shows the estimated power spectra recorded over several trials in each of the two aforementioned states. Particularly around 8Hz, Subfigure 6.4b representing the eyes-closed state shows noticeably more signal power in the alpha band relative to the rest of the signal (compared to the eyes-open state). This is suggestive of the presence of valid EEG signals as opposed to just random noise.

6.1.3 Execution time profiling

A number of trials were performed to determine the consistency of execution time across key system processes in the sample-process-decode-publish loop. Figure 6.5 shows the distributions of execution times recorded over several trials. In each trial, all possible functionality was enabled so as to test maximum computational loading. 'Auxiliarly processing time' is defined as the processing time in each loop required for any operations *besides* decoding computation and publishing to AWS over MQTT.



(a) decoding computation (b) MQTT publish (c) auxiliary processing

Figure 6.5: Execution time distributions for key processes in the sample-decode-publish loop. Auxiliary processing refers to any processing besides that which is required for decoding and publishing data to the cloud.

6.2 Experimental Decoding Results

In this section, the experimental results of various decoding algorithms introduced in Chapter 2 are reported. Unfortunately, because only single channel EEG signals were available using the hardware provided in this project, the TRCA algorithm was no longer appropriate. This is due to the fact that it tries to find optimal *spatial* filters for task-related discrimination; effectively optimally-weighted combinations of multiple channels. Clearly, having only a single channel renders this approach ineffective.

Evaluation of results

Worth noting is the method used to evaluate results reported below. For template-based decoding algorithms (GCCA, MsetCCA), initial training or calibration is required using historical template data. The term 'calibration' is preferred since in practice, calibration must happen automatically on the device and the term 'training' is typically associated with offline model fitting.

Leave-p-out cross-validation (LpO CV) was employed in order to minimise the risk of selection bias or overfitting to pathological test cases. Usually, LpO CV refers to *training* a model on $n - p$ observations and *validating* on the held-out p validations (with $p < n$). However, in this project, template-based algorithms were instead *calibrated* on the smaller p-partition and validated on the remaining $n - p$ samples. This more appropriately reflects the practical dynamic of this system where only few calibration trials will be available before inference needs to begin.

LpO CV was selected, as opposed to leave-one-out CV[1], for example, as it provides the ability to vary p in order to investigate the effect of more or fewer calibration trials on decoding performance. For n trials

[1] a special case of LpO CV with $p = 1$ that is more commonly used

or observations, LpO CV will generate $\binom{n}{p}$ unique calibration-validation splits[2]. Validation metrics, such as accuracy, are then averaged across all splits (unless otherwise stated). This is depicted graphically in Figure 6.6. Note that, contrary to Lpo CV, the more well-known k-fold cross-validation technique is a *non-exhaustive*

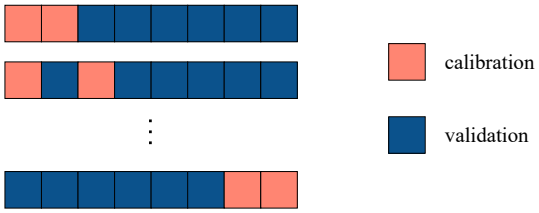


Figure 6.6: Diagram illustrating leave-p-out cross validation (Lpo CV). Individual blocks in each row are different trials, each represented by a data matrix $X_n \in \mathbb{R}^{N_s \times N_c}$ where N_s is the number of samples per trial and N_c is the number of channels ($N_c = 1$ in this project). For n trials, there will be $\binom{n}{p}$ different calibration-validation splits (rows of blocks).

CV method as it does not test every combination of splits in the original data sample: it only selects train-validation splits whose elements are contiguous. In comparison, Lpo CV may select non-contiguous elements in either or both of the train and validation sets. Therefore, k-fold CV is a less computationally-intensive approximation of the more rigorous Lpo CV method.

6.2.1 The effect of recording window length

As alluded to in Section 2.3.1, arguably the most important factor in the decoding system - besides the actual algorithm used - is the length of the recording window T. As cited in Chapter 2, decoding accuracy almost always improved with increasing T. This is expected; longer time windows offer more samples to the decoding algorithm being used. However, the trade off is decreased information transfer rate (ITR) which results in a more sluggish BCI system.

Figures 6.7 and 6.8 show the effect of varying T on decoding accuracy for the GCCA and MsetCCA algorithms respectively. For each algorithm, the effect of varying T is tested with four different calibration scenarios with $p = 1, \ldots, 4$ where p is the number of training or calibration trials. As expected, there is a monotonic increase in *average* decoding accuracy (across stimulus frequencies for each trial) with increasing T.

6.2.2 The effect of varying calibration trials

Another important consideration for template-based algorithms such as GCCA and MsetCCA cited in the literature is the number of calibration or training trials used to 'fit' the algorithms prior to inference. In this context, 'fit' refers to computing optimised reference signals or canonical weights based on historical data.

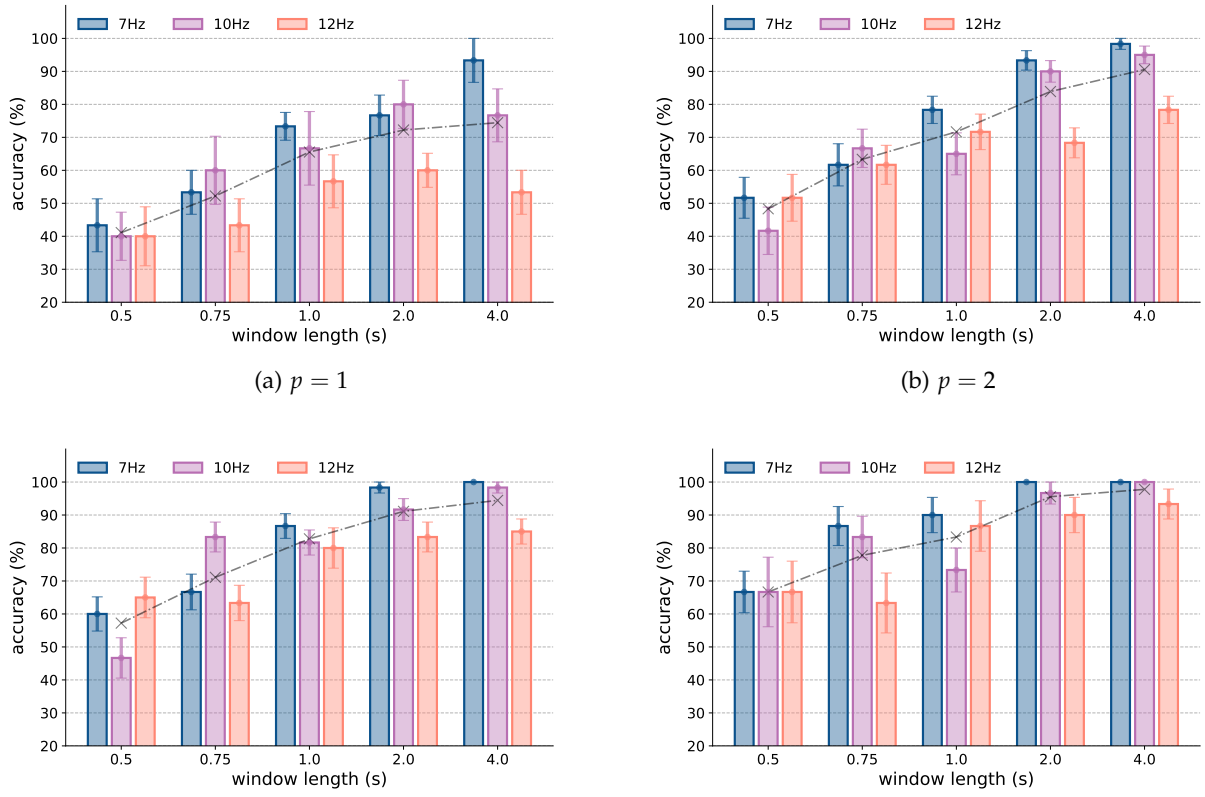
Figures 6.9 and 6.10 show the effect of varying p, the number of calibration trials used to fit the GCCA and MsetCCA algorithms respectively. These tests were performed in four different sets, each with a distinct recording window length T. Similar to the prior experiment for varying T directly, there is a monotonic increase in *average* decoding accuracy with increasing p. Note that the same cross-validation evaluation procedure as explained in Section 6.2 was used to generate these results.

6.2.3 Generalisation testing

Of interest to template-based decoding algorithms that leverage historical 'training' or calibration data is their ability to *generalise* to new sets of data. In order to clarify this idea in context, consider two trial sessions X_a and X_b taken under *different conditions* where the subject would have at least had to remove and reinstall the BCI headband on their head between sets. Thus, X_a and X_b would be recorded with slightly different electrode positions and contact impedances and would therefore exhibit different signal quality.

This experiment was thus designed to investigate whether calibration using data from one session, X_a, say, would allow generalise to another, say, X_b. That is, would calibrating using X_a and testing (validating) on X_b yield comparable performance to calibrating and testing on data recorded from the same session as in prior experiments? If so, the implication is that calibration would only need to be performed once-off, or at least, not *every* recording session which would enhance the user experience. An extension of this experiment was

[2]This would generally become computationally infeasible for even modest n. However, the calibration or 'training' processes for the algorithms in this project were not computationally intensive and the number of trials in each experiment never exceeded 10.



(a) $p = 1$ (b) $p = 2$

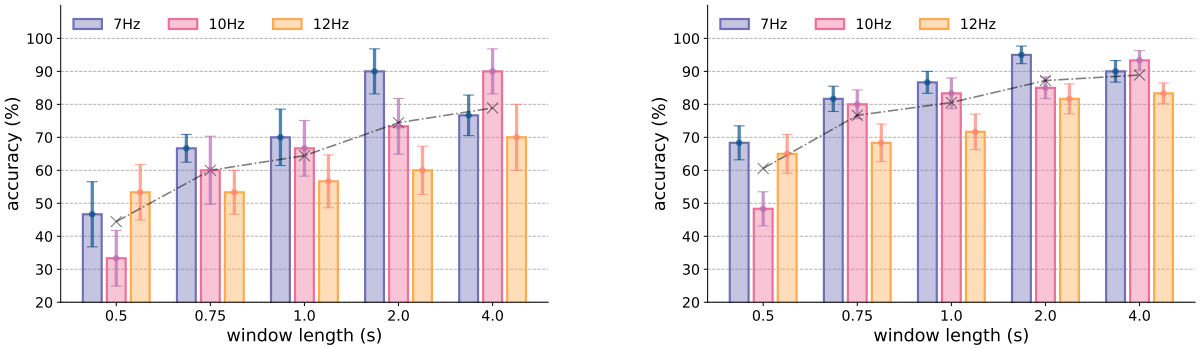
(c) $p = 3$ (d) $p = 4$

Figure 6.7: **GCCA decoding accuracy, varying** T: effect of varying recording window length T on validation accuracy for different numbers of calibration trails N_t. In each subfigure, all results were computed using $p = n, n \in \{1, \ldots, 4\}$ calibration trials. Crosses connected with dashed traces denote average decoding accuracy across stimulus frequencies for a given a given window length T. Error bars denote one standard error of the mean.

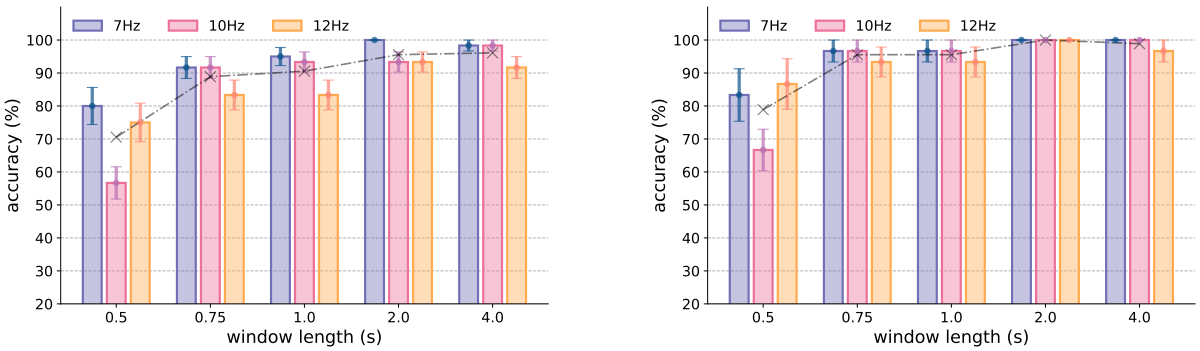
to determine if calibrating using X_a *and* a small subset of X_b and then testing on the remainder of X_b would yield better performance than only calibrating on the subset of X_b as in previous experiments.

The results of such an experiment are shown in Figure 6.12 and Figure 6.13 for the GCCA and MsetCCA algorithms respectively. Continuing the analogy with X_a and X_b above, the number of 'overlapping trials' in these figures refers to the number of trials from X_b included in the calibration set along with the whole of X_a. Accordingly, 0 overlapping trials would correspond to a situation where calibration was performed exclusively on X_a and testing on X_b. This condition tests the initial question in the experiment mentioned above. The case of n overlapping trials with $n > 1$ corresponds to calibration using all of X_a *and* n trials from X_b while still testing using the left-over[3] trials from X_b. Calibration and 'training', as referred to in the figures, are used interchangeably here.

[3] the set difference between all trials in X_b and the n-element subset used for calibration.

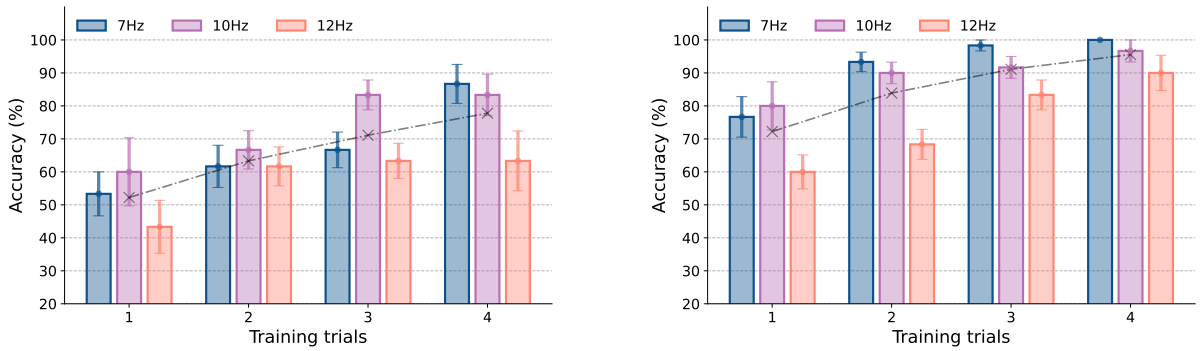


(a) $p = 1$ (b) $p = 2$



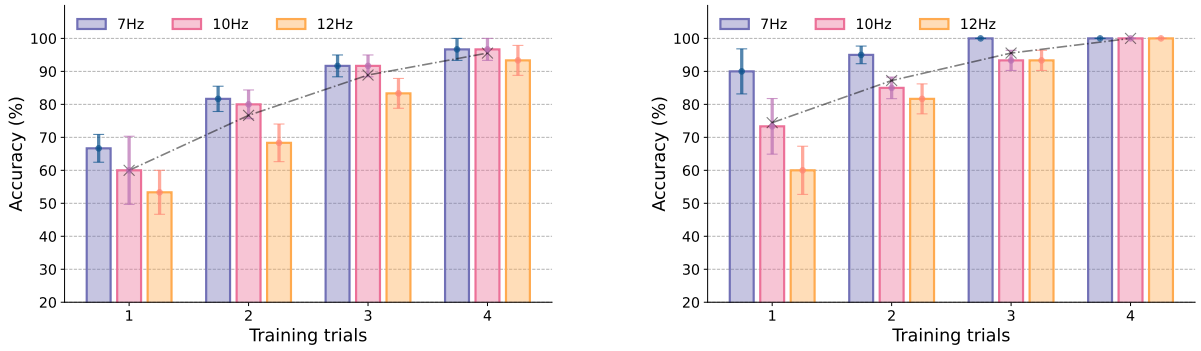
(c) $p = 3$) (d) $p = 4$

Figure 6.8: **MsetCCA decoding accuracy, varying** T: effect of varying recording window length T on validation accuracy for different numbers of calibration trails p. In each subfigure, all results were computed using $p = n$, $n \in \{1, \ldots, 4\}$ calibration trials. Crosses connected with dashed traces denote average decoding accuracy across stimulus frequencies for a given a given window length T. Error bars denote one standard error of the mean.



(a) $T = 0.75$s ($N_s = 192, N'_s = 48$) (b) $T = 2$s ($N_s = 512, N'_s = 128$)

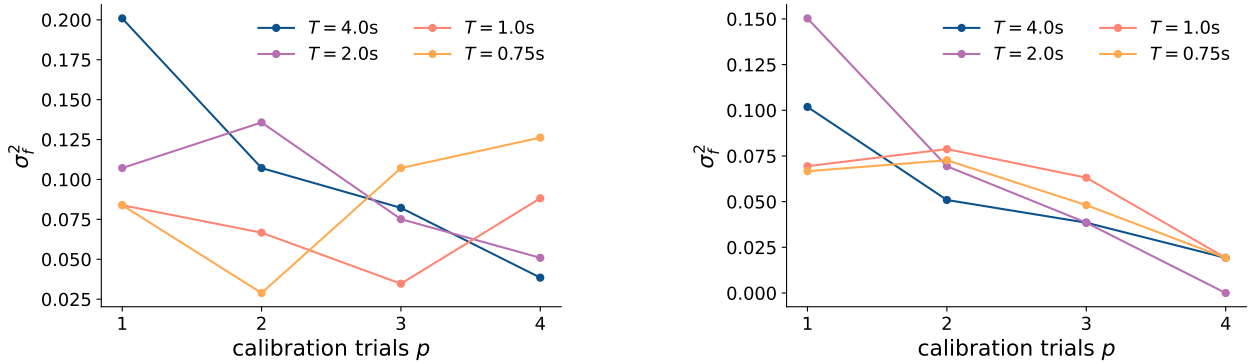
Figure 6.9: **GCCA decoding accuracy, varying** p: effect of varying the number of training (calibration) trials p on validation accuracy for recording windows of varying length T. In each subfigure, all trials are of equal length T seconds which equates to N_s samples at $f_s = 256$Hz and N'_s samples at the downsampled rate of $f'_s = 64$Hz. Crosses connected with dashed traces denote average decoding accuracy across stimulus frequencies for a given $p = n$, $n \in \{1, \ldots, 5\}$. Error bars denote one standard error of the mean.



(a) $T = 0.75$s ($N_s = 192, N_s' = 48$)

(b) $T = 2$s ($N_s = 512, N_s' = 128$)

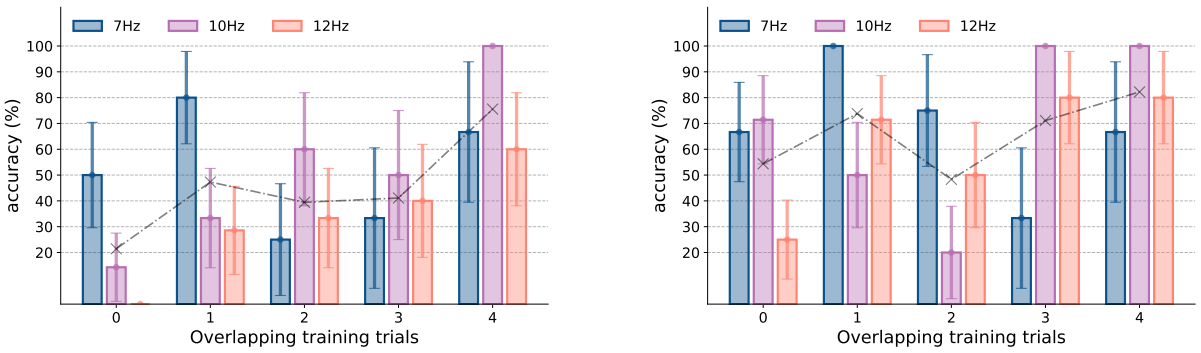
Figure 6.10: **MsetCCA decoding accuracy, varying** p: effect of varying the number of training (calibration) trials p on validation accuracy for recording windows of varying length T. In each subfigure, all trials are of equal length T seconds which equates to N_s samples at $f_s = 256$Hz and N_s' samples at the downsampled rate of $f_s' = 64$Hz. Crosses connected with dashed traces denote average decoding accuracy across stimulus frequencies for a given $p = n$, $n \in \{1, \ldots, 5\}$. Error bars denote one standard error of the mean.



(a) GCCA algorithm

(b) MsetCCA algorithm

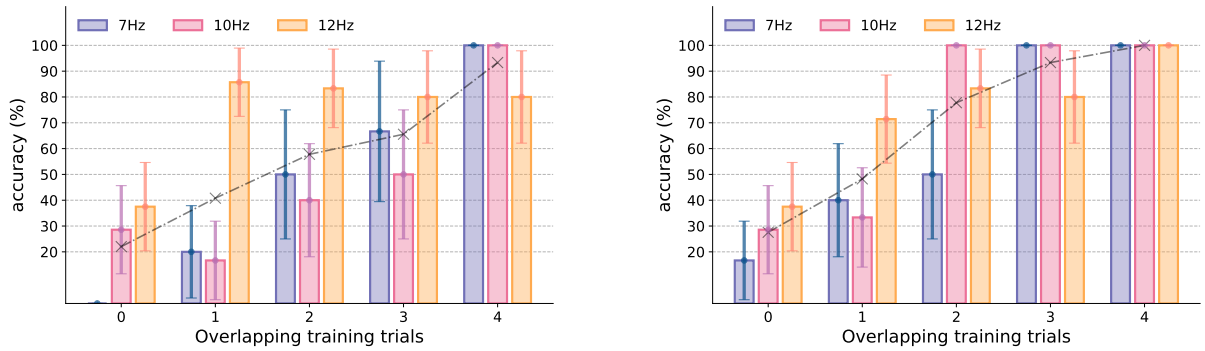
Figure 6.11: Graphs showing variance in decoding accuracy across the triad of stimulus frequencies at each value of p. The standard deviation in accuracy over frequencies σ_f^2 is reported. Different colour traces represent recording windows of different lengths as labelled.



(a) $N_s = 512$, $N_s' = 128 \Leftrightarrow T = 2$s

(b) $N_s = 1024$, $N_s' = 256 \Leftrightarrow T = 4$s

Figure 6.12: **GCCA generalisation performance**: decoding accuracy on data from distinct recording sessions taken under different conditions, X_a and X_b where X_a is a calibration (training) set and X_b a test set. The number of overlapping trials refers to the number of trials in X_b *also* used for calibration. Results using $T = 2$s and $T = 4$s recording windows are shown. Error bars denote one standard error of the mean.



(a) $N_s = 512$, $N'_s = 128 \Leftrightarrow T = 2$s

(b) $N_s = 1024$, $N'_s = 256 \Leftrightarrow T = 4$s

Figure 6.13: **MsetCCA generalisation performance**: decoding accuracy on data from distinct recording sessions taken under different conditions, X_a and X_b where X_a is a calibration (training) set and X_b a test set. The number of overlapping trials refers to the number of trials in X_b *also* used for calibration. Results using $T = 2$s and $T = 4$s recording windows are shown. Error bars denote one standard error of the mean.

p	**1**				**2**				**3**				**4**			
T	**0.75**	**1**	**2**	**4**	**0.75**	**1**	**2**	**4**	**0.75**	**1**	**2**	**4**	**0.75**	**1**	**2**	**4**
$\bar{\mathbf{x}}$	52.22	65.56	72.22	74.44	63.33	71.67	83.89	90.56	71.11	82.78	91.11	94.44	77.78	83.33	95.56	**97.78**
$\mathbf{s}_{\bar{\mathbf{x}}}$	8.341	7.800	6.204	7.120	6.037	5.292	3.590	2.823	5.095	4.556	3.167	1.824	7.098	6.558	2.893	**1.514**
σ_f^2	0.084	0.084	0.107	0.201	**0.029**	0.067	0.136	0.107	0.107	0.035	0.075	0.082	0.126	0.088	0.051	0.039
ITR	8.69	18.70	13.64	7.64	21.61	26.51	23.61	15.59	34.30	45.00	31.90	18.29	**47.89**	46.09	38.35	21.14

Table 6.1: **GCCA results summary**: a compilation of decoding performance metrics for the GCCA algorithm. Results are reported over varying numbers of calibration trials p and recording window lengths T (in seconds). Average accuracy across frequency $\bar{\mathbf{x}}$ its standard error $\mathbf{s}_{\bar{\mathbf{x}}}$ are reported in %. **ITR** is given in bits/min. Accuracy variance over frequency σ_f^2 is dimensionless.

p	**1**				**2**				**3**				**4**			
T	**0.75**	**1**	**2**	**4**	**0.75**	**1**	**2**	**3**	**0.75**	**1**	**2**	**4**	**0.75**	**1**	**2**	**4**
$\bar{\mathbf{x}}$	60.00	64.44	74.44	78.89	76.67	80.56	87.22	88.89	88.89	90.56	95.56	96.11	95.56	95.56	98.89	**98.89**
$\mathbf{s}_{\bar{\mathbf{x}}}$	7.070	8.341	7.522	7.659	4.633	4.462	3.496	3.126	3.716	3.438	2.039	2.215	3.736	3.736	**0.512**	1.111
σ_f^2	0.067	0.069	0.150	0.102	0.073	0.079	0.069	0.051	0.048	0.063	0.039	0.039	0.019	0.019	0.019	**0.019**
ITR	17.12	17.42	15.28	9.45	45.44	40.80	27.17	14.56	77.65	62.37	38.35	19.63	**102.28**	76.71	44.58	22.29

Table 6.2: **MsetCCA results summary**: a compilation of decoding performance metrics for the MsetCCA algorithm. Results are reported over varying numbers of calibration trials p and recording window lengths T (in seconds). Average accuracy across frequency $\bar{\mathbf{x}}$ its standard error $\mathbf{s}_{\bar{\mathbf{x}}}$ are reported in %. **ITR** is given in bits/min. Accuracy variance over frequency σ_f^2 is dimensionless.

Chapter 7

Discussion of Results and Methodologies

7.1 Digital signal processing system

The role of the DSP system in this project was paramount. Without appropriately processed signals, even the most sophisticated decoding algorithms would have been rendered useless.

The time domain plot in Figure 6.1 demonstrates effective sampling; the input square wave is captured with consistent frequency and amplitude. As is more visible in Figure 6.2, the estimated spectrum of the low-pass-filtered signal is almost exactly as desired; pass-band distortion is negligible and extremely steep roll-off is achieved in the transition band. Furthermore, stop-band attenuation is more than sufficient and, encouragingly, is very close to the 80dB design specification. Finally, after establishing effective sampling and filtering, the PSD estimate of the downsampled signal in Figure 6.3 verifies that the theoretical downsampling rate of $f_s' = \frac{1}{4} f_s = 64$Hz is able to maintain signal fidelity in the pass-band. The implication of this is that 4x fewer samples can be used to represent the original signal with negligible loss of information. This is invaluable for a system such as the one in this project that is severely constrained in memory and computational power.

In summary, the DSP system proved very effective and was verified to operate in very close agreement with its design parameters. Worth emphasising is the fact that all necessary DSP operations are able to be performed in firmware on the ESP32. Not only is this a prerequisite for the end-to-end on-device processing objective, but also, it makes for a more flexible and future-proof system. New filter weights or other DSP-related parameters can simply be flashed onto the device, or indeed, uploaded 'over the air' through a network connection if needed.

7.2 Decoding

Broadly speaking, the decoding accuracy results in Chapter 6 demonstrate that the system developed in this project is a very promising first prototype. Before any further analysis of these results, it should be emphasised that while impressive, they were computed using data collected exclusively from the author owing to COVID19-related circumstances. Notwithstanding, several trials were conducted under various conditions and a stringent evaluation procedure, as detailed in Section 6.2, was employed in an attempt to provide more robust results.

Trends in decoding accuracy with varying recording window size T and number of calibration trials p were largely similar across the GCCA and MsetCCA algorithms. Accordingly, these common trends are discussed below, followed by an analysis of their differences in Section 7.2.

The trends in average classification accuracy across frequencies in Figure 6.7 and Figure 6.9 show a strong positive correlation to the length of the recording window T used for measurement. This is a fairly obvious and expected result. It should be noted, however, that this increase in accuracy with T was not exclusively monotonic for each stimulus frequency. For example, in Figure 6.7, decoding accuracy of the 12Hz stimulus was higher for $T = 1$s than $T = 2s$. These infrequent fluctuations could be due to random noise or EMG or other signal artefacts present during one trial but not another.

What is evident from both experiments with varying T and p is that the reliance on the number of calibration trials decreases with increasing T. For example, in Figure 6.9, only $p = 2$ calibration trials are required using $T = 2$s windows to exceed the average accuracy for $p = 4$ calibration trials when using $T = 0.75$s windows. Figure 6.11 shows that, in general, the variance in decoding accuracy across stimulus frequencies decreases when more calibration trials are available. For the GCCA algorithm, this behaviour is only consistent for $T > 1$s, however. The reason for this is likely because with more calibration data on which to fit the decoding

models, inferential/test variance tends to decrease as the reference templates or filters learnt during fitting better represent underlying EEG dynamics and can better reject random, task-unrelated components. As the decoding classifiers effectively use independent instances of their respective decoding algorithm for each frequency, a decrease in test variance within frequency sets results in a decrease in variance across them too.

Comparison of accuracy across algorithms

While trends in decoding accuracy between the GCCA and MsetCCA algorithms are similar, there are some notable differences. With reference to the summarised performance metrics in Table 6.2 and Table 6.2, MsetCCA shows superior quantitative performance characteristics over all metrics considered. Furthermore, the plots in Figure 6.11 show a far more consistent decline in accuracy variance across frequencies with increasing p for MsetCCA compared to GCCA, as well as lower absolute variances. Finally, in the results of the generalisation tests in Section 6.2.3, the MsetCCA results once again show a more consistent trend with substantially less variance than the GCCA counterparts. This behaviour suggests that, with more historical calibration data from across different conditions and test subjects, MsetCCA is more likely to generalise to true out-of-sample data.

7.2.1 Online decoding

One of the biggest challenges of this project was to develop a system capable of end-to-end, real-time on-device decoding. That is, an embedded system capable of performing all operations needed to acquire, process, decode and communicate signals without outsourcing any computation off the device.

As documented in the system design in Chapter 5 and demonstrated by the results in Chapter 6, one can conclude that this objective has been achieved. This is extremely significant as it means that a fully functional (albeit fairly basic) BCI device can be deployed in a small, self-contained package that is both completely mobile and extremely cost-effective. From a technical perspective, the fact that sophisticated decoding algorithms leveraging eigenvalue solvers and other high level linear algebra can be performed on a microcontroller - and with such ease of implementation - is quite remarkable.

7.2.2 Generalisation ability

One of the most significant limitations of this system is the need for calibration during prior to each new recording session. A session, in this context, is defined as a distinct, continuous period in which the BCI headset was not removed or adjusted. Correspondingly, in order to separate sessions and truly test generalisation ability, it was required that the headset be removed between sessions. The results of the generalisation tests in Section 6.2.3 show that using calibration data from one session was not helpful when used in preparation for inference in a different session. Referring to the convention explained in Section 6.2.3, Figure 6.12 and Figure 6.13 show very poor decoding performance for the case of no overlapping trials. This suggests that using calibration and validation data from two completely distinct sessions is not feasible - at least, not with the limited data available at this point. Interestingly, with $n > 0$ overlapping trials, decoding accuracy was *worse* than in the original decoding tests in Section 6.2.1 where calibration and test data sets were drawn from the same session (ultimately leaving less data available for calibration). In the case of overlapping trials, calibration was performed on both the whole first session *and* a subset of the second session on which testing was performed. It is thus surprising that this overlapping case yielded poorer performance as the decoding algorithms had access to a superset of the calibration data available in the initial tests in Section 6.2.1.

These findings suggest that there is dramatic variance in recorded signals across sessions. This may be due to several factors:

1. There was **no standardised positioning** of the BCI headband on the subject's head across sessions. Each time the headband was put on, an attempt was made to estimate consistent positioning but as the electrodes were positioned at the back of the head, this was difficult to do reliably without assistance (which was seldom available). Consequent deviations in the electrode positions across sessions would naturally produce noticeable variance in measured signals.
2. Only dry electrodes could be used and **scalp-contact quality was inconsistent** at best. On some occasions, it was found that electrodes would be lodged in a more favourable orientation with better scalp contact than on others. This was caused by variations in hair partings and angle-of-location relative to the protruding inion at the back of the skull.
3. Albeit a differential measurement across two electrodes, only a **single EEG channel** was available. This poses more of a challenge for decoding algorithms to reject task-unrelated artefacts and random noise owing to having no other independent measurements available.

7.3 Networking and communication

The MQTT protocol for publishing data to the AWS IoT Core service proved very effective. Although it was not formally tested, there were no recorded instances of packet loss. Furthermore, it is a widely-adopted protocol in the Internet of Things (IoT) community and as a result, there are many open source MQTT utilities, guides and sources of documentation. This was convenient during the development of this project and will hopefully continue to be in future revisions if used for educational purposes.

A limitation of having to use this protocol over public Internet is the slightly inconsistent latency experienced in publishing data. The distribution of time required to successfully publish decoded data to AWS ('publish time') in Figure 6.5b demonstrates this phenomenon. The mean publish time is around 100ms with a standard deviation of 210ms which leaves most times below the 350ms mark. However, it was noticed that sporadically, times of over 2s would occur. This is likely due to fluctuations in network traffic - both on the Internet, the AWS service and most likely, within the personal network of the user. This is not a significant issue, however, as all data payloads are sent with timestamps; any readings received after a predetermined tolerance window could simply be discarded before resynchronising the system.

7.4 System Design and Methodologies

7.4.1 Challenges encountered

Several trials and tribulations were encountered in this project - most of which were related to numerical/computation challenges or memory limitations. One of the first of these was a limitation in the eigenvalue solver in the `ulab` module which only allowed for computation on symmetric matrices. Through engagement with the author of `ulab` project, Mr Zoltán Vörös, a way to circumvent this problem was developed. After reading about the QR-iteration algorithm for solving for eigenvalues iteratively, a request for a QR decomposition function was put forth to Mr Vörös. Fortunately, he kindly obliged and soon added the required functionality which can now be found in the latest version of this module. This is one of the beauties of open source software.

Another challenge was encountered upon noticing that, even with identical inputs, certain linear algebra operations on the ESP32 were not producing the same results as their equivalents on a 64-bit computer. After some debugging, this was discovered to be down to precision issues in the MicroPython firmware image running on the ESP32. Fortunately, only one compiler flag had to be adjusted to enable double precision before rebuilding and flashing the updated MicroPython firmware. This solved the earlier numerical equivalence issues.

7.4.2 Limitations of the system

Besides the extremely limited budget, the constraint of having only a single channel likely proved to be the most significant in this project. The literature cited in Section 2.5 suggests that very few, if any, BCI devices rely on a single measurement channel alone. Systems with two to four electrodes are far more common; the studies reviewed in Section 2.2.2 showed that, up until some saturation point, increasing the number of active channels invariably improves decoding accuracy.

Not only would the presence of multiple channels very likely improve decoding performance in the existing decoding algorithms used, but it would also enable other multi-channel algorithms to be used. TRCA, as presented in Section 3.2.2, is one such algorithm that requires multiple channels. This algorithm (and its extensions) has been shown to be very effective and would prove a worthy contender to the existing CCA-based algorithms used in this project.

While less of a concern in a development scenario, the ergonomics of the BCI device will become an important consideration when used in a more public setting. The current hardware prototype, as pictured in Figure 4.5, becomes remarkably uncomfortable after even a modest amount of time. This is likely due to the fact that the electrodes protrude substantially from the headband strap.

As discussed in Section 7.2.2, the current BCI system requires at least a few initial calibration trials to work effectively. However, this is not a very severe limitation since calibration can be performed in just 10 to 15 seconds if only two calibration trials of $T = 2$s are employed, for example. This calibration period can also be reduced if slightly decreased performance can be tolerated. The more significant limitation, however, is that calibration must be performed at the beginning of each new session. That is, each time the BCI headband is removed and reinstalled (or moved around significantly without being completely removed).

7.4.3 Choice of development tools

The tools used to develop this project, including the software, firmware and other technologies, were not only important during the development phase, but will continue to be after completion of this dissertation. The reason for this is that, as specified in the initial project scope and constraints mentioned in Chapter 1, a core objective for this project was to create a platform that can one day be used as an engagement and/or educational tool in the neurotechnology community. This factor strongly guided the choice of technologies in this project and ultimately, a system has been designed that uses exclusively open source tools.

The choice of MicroPython for development with the ESP32 proved to be extremely prudent. MicroPython offers extremely simple yet elegant syntax and useful constructs that not only speed up development, but also make the process far more enjoyable. From an educational perspective, MicroPython is far more intuitive than C/C++ or their derivatives and shares the same syntax as Python; an extremely popular language for programming education. This also allowed for the use of the `ulab` module for linear algebra and other scientific computing that was absolutely indispensable to this project.

Finally, ease of development using the MicroPython ecosystem is undoubtedly the easiest and most convenient of all other options. In particular, a Jupyter Notebook can be used over a serial connection to a MicroPython-compatible board to offer immediate and interactive development, experimentation and debugging. Compared to more traditional embedded ecosystems that require any code updates to be flashed onto the target MCU and rerun, this interactive environment is remarkably efficient. All components of this system - including those responsible for data acquisition, filtering, computation and communication - were developed in an interactive notebook environment before being compiled into frozen bytecode on the device.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

Over the course of this project, an ultra low-cost, fully self-contained BCI prototype was developed. The research questions posed in Section 1.2.1 have been addressed:

1. An ultra low-cost BCI device can be used for the purpose of decoding basic EEG signals such as SSVEPs in real time. Subject to performing a short calibration sequence each time the BCI is put on, decoding accuracy of $95.56 \pm 3.74\%$ with an ITR of 102 bits/min ($p = 4$, $T = 0.75$s). With more modest calibration requirements ($p = 2$, $T = 1$s), accuracy of $80.56 \pm 4.46\%$ with an ITR of 40 bits/min can be achieved.
2. The device can achieve the aforementioned performance when performing *all* operations necessary to do so on the device itself. No computational or other outsourcing is required.

The results presented in Chapter 6 demonstrate that the DSP system, coupled with the decoding algorithms used, proved very effective for SSVEP decoding. It can be concluded that, of the applicable algorithms explored, MsetCCA offers superior performance in terms of accuracy, ITR and robustness. While the results in this study are somewhat limited due to the fact that data was only collected from a single subject, they nonetheless present an encouraging proof-of-concept for further, more stringent investigation.

At the time of writing, to the best of the author's knowledge, there are no fully mobile, self-contained BCI devices in the price range of around £20. Furthermore, unlike many existing research devices that use proprietary software or other tools, this prototype was developed using exclusively open source software. As a result, this project presents an exciting prospect for further research and development as it could become a valuable tool for education and public engagement.

8.2 Future work

While the system developed in this project performed remarkably well considering the constraints imposed on it, there are several areas for improvement. The following observations and suggestions are made for future extensions of this project.

Multi-channel sensing

As mentioned in the prior discussion, the limitation of only having one active EEG channel was significant. It is firmly believed that introducing addition channels would significantly improve decoding performance. Furthermore, it is conceivable that the existing electronic hardware in this project could be adapted for multi-channel support by using a multi-channel analogue frontend (as is used on the OpenBCI Ganglion board, for example). The ESP32 SoC also has several available ADC inputs.

Algorithm expansion

While the algorithms selected in this project are some of the most popular and are close to the state-of-the-art at the time of writing, there are many other slight adjustments and extensions to these algorithms that warrant exploration. For example, if multi-channel sensing is introduced, algorithms related to finding optimal spatial filters *across channels* can be used. TRCA and its extensions (such as ensemble TRCA) are examples of such algorithms that have proved effective in the literature.

A more robust testing framework

As alluded to, the COVID19 Pandemic made it difficult to gather data from different subjects during the course of this project. As a result, data was only collected from the author. An obvious future step would be to extend this experiment over many more test subjects of different ages with differing head morphologies and hair types. Having a broader test group could also be valuable for other reasons; subject to relevant data privacy considerations, the web logger infrastructure documented in Section 5.7 could allow anonymised EEG data to be collected remotely from a wide audience from the comfort of their homes. Optimising decoding algorithms using data from a wider variety of people would likely lead to a much greater chance of achieving effective generalisation across recording sessions with less need for calibration.

Due to time constraints, testing procedures in this project were also fairly limited. More stringent validation tests would be helpful extensions. For example, data was collected under fairly idealised conditions where the subject did not move their head or eyes considerably. In future, the effect of some of the following disturbances on decoding performance could be investigated:

- varying degrees of head and body movement
- eye movement and blinking
- modulation of focus on the stimulus and concentration in general

Another useful test would be to occasionally randomise the order of stimulus frequencies associated with the flashing squares in the SSVEP user interface. This could be used to verify that the decoding algorithms are indeed using valid EEG signals for frequency discrimination and not just other signal components correlated to the fixed position of the stimulus frequencies.

Finally, experimenting with different SSVEP stimuli configurations and analysing the impact on decoding performance (if any) would be interesting. This could include varying the spatial distribution of stimuli as well as their colours and contrast relative to the background colour.

Improved SSVEP squares user interface

As depicted in Figure 5.1, the SSVEP squares user interface created for this project is very basic. A very useful extension would be to interpret signals published from BCI devices in order to provide user feedback of decoded signal results. Potentially, a means of visualising raw data could also be incorporated which would offer a far richer user experience.

Another limitation of the current user interface is that the flicker frequencies of stimulus squares are controlled by vanilla Javascript. Consequently, there is likely to be fluctuations in the consistency of frequencies displayed on different devices and even on the same device during different computational loading conditions. In future, a more robust technology should be investigated to ensure that precise and consistent flicker frequencies are maintained. WebGL or related wrapper libraries such as Three.js could be potential options.

Improved BCI headband ergonomics

As mentioned in Section 7.4.2, the current design of the BCI hardware becomes uncomfortable to wear, even after short periods of time. Before deploying this device in a public setting, work should be done to improve the ergonomics of the design to achieve greater comfort; ideally without any resulting compromises to the performance of the system.

Appendix A

Appendices

A.1 Firmware Implementations

```
from ulab import numpy as np

def power_iteration(A, iterations):
    """
    Iterative algo. to find the eigenvector of a matrix A corresponding to the largest
    eigenvalue.
    """
    # choose random initial vector to reduce risk of choosing one orthogonal to
    # target eigen vector
    b_k = np.array([urandom.random() for i in range(len(A))])

    for _ in range(iterations):
        b_k1 = np.dot(A, b_k)
        b_k1_norm = np.linalg.norm(b_k1)
        # re normalize the vector
        b_k = b_k1 / b_k1_norm

    return b_k1_norm, b_k
```

Listing 4: MicroPython implementation of the Power Iteration algorithm presented in Algorithm 1 for finding the maximum real eigenvalue of a symmetric, real-valued matrix

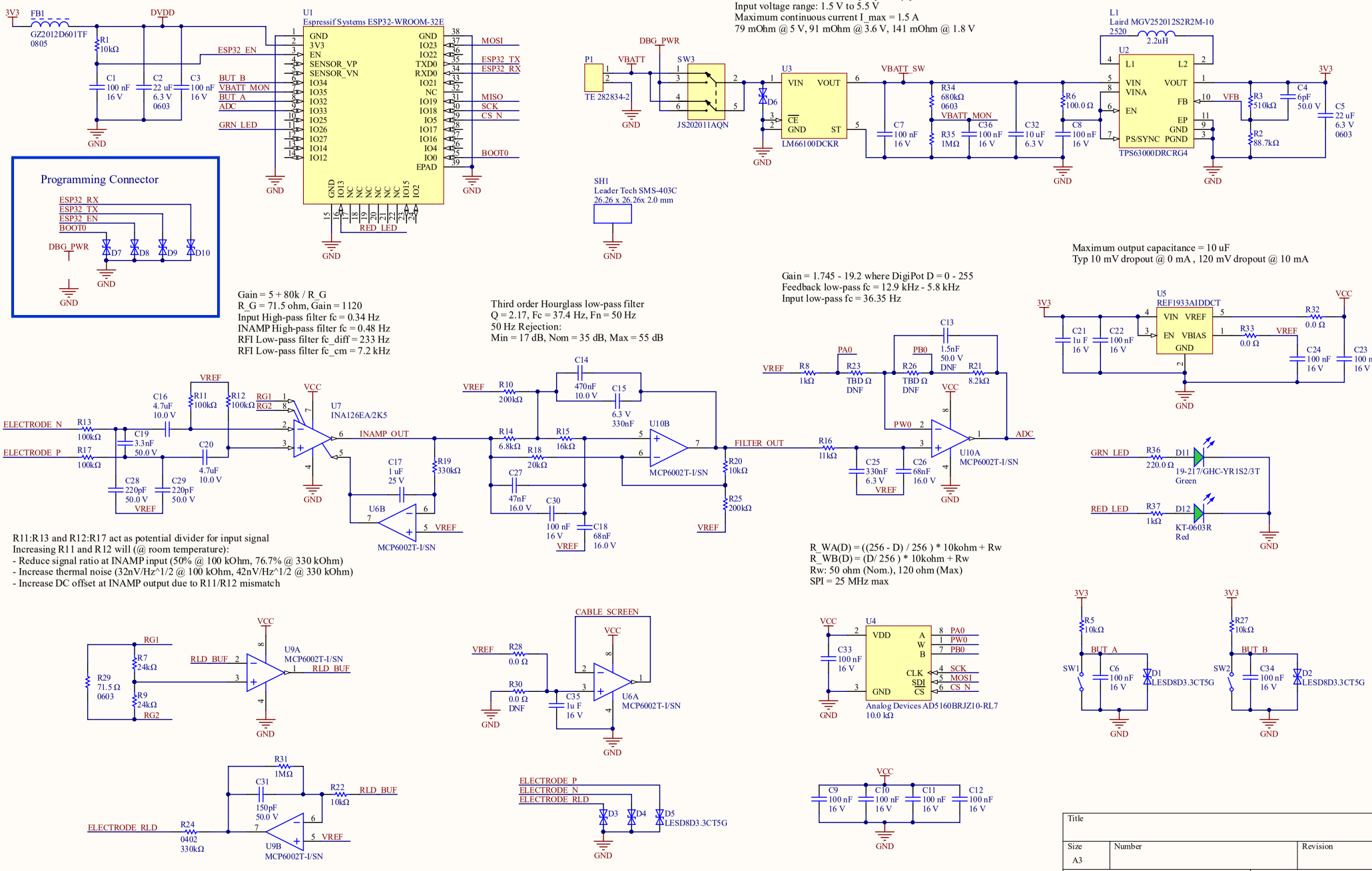
```
from ulab import numpy as np

def qr_iteration(A, iterations=30):

    """
    Use the QR iteration algorithm to iteratively solve for the eigenvectors and eigenvalues
    of a matrix A. Note: only guaranteed to recover exactly for symmetric matrices
    with real eigenvalues. May work partially for asymmetric matrices (no complex support yet).

    Number of iterations is specified by `iterations` and should be tuned empirically. Typically,
    30 is more than enough.
    Returns:
        `lam`: vector of eigenvalues
        `Q_bar`: matrix of eigenvectors (columns)
    """

    Ak = A
    Q_bar = np.eye(len(Ak))

    for _ in range(iterations):
        Qk, Rk = np.linalg.qr(Ak)
        Ak = np.dot(Rk, Qk)
        Q_bar = np.dot(Q_bar, Qk)

    lam = np.diag(Ak)
    return lam, Q_bar
```

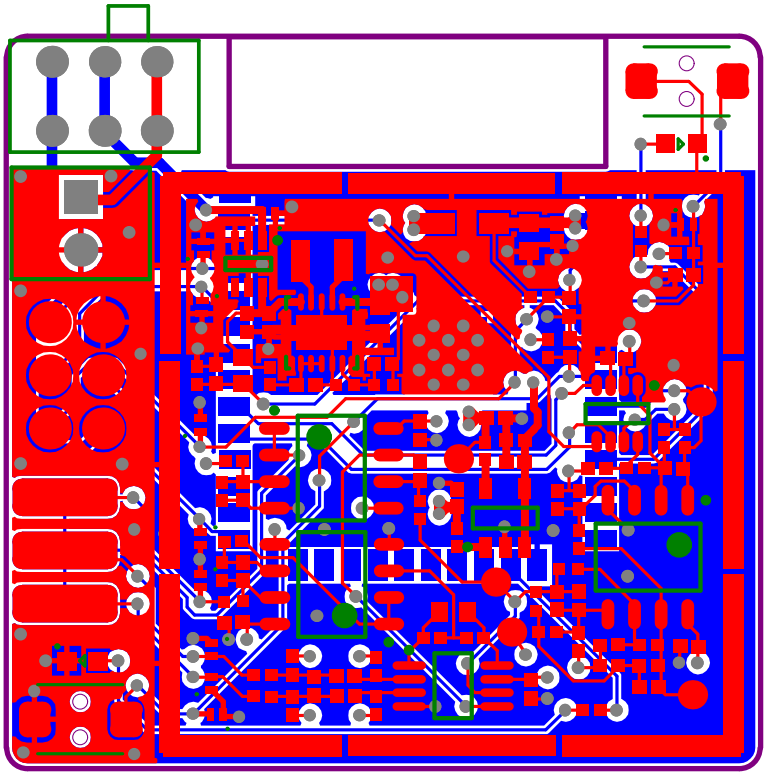
Listing 5: MicroPython implementation of the QR Iteration algorithm presented in Algorithm 3

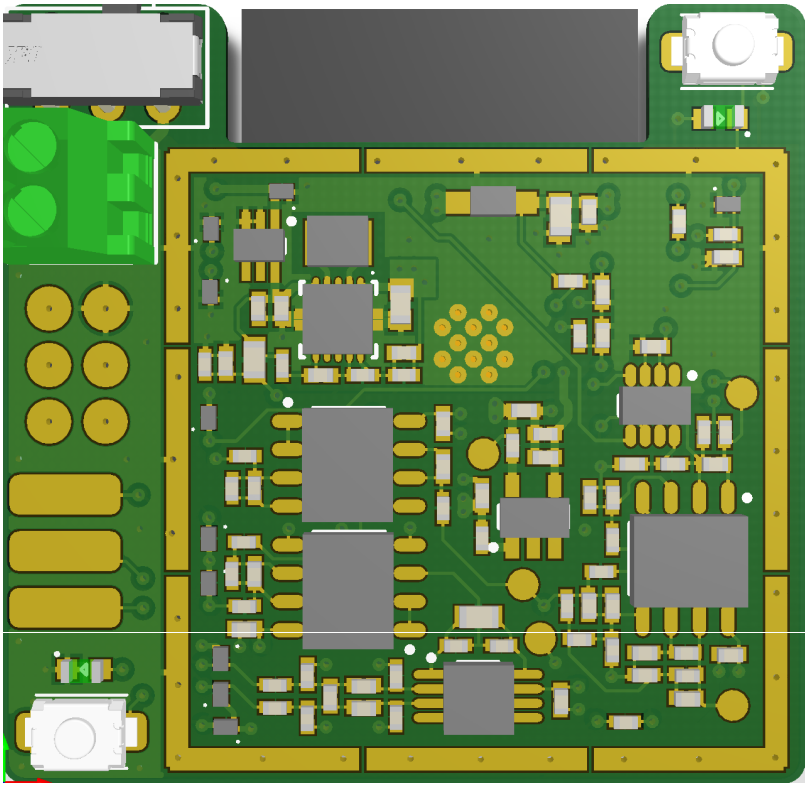
A.2 EEG Hardware Schematics

All schematics in this section were kindly provided by Steven Wong from the Imperial College NGNI Lab who designed them.

All passive components are 0402 unless specified
Resistor tolerance = 1%, Capacitor tolerance = 10%







```
from ulab import numpy as np

def solve_gen_eig_prob(A, B, eps=1e-6):
    """
    Solves the generalised eigenvalue problem of the form:
    Aw = lambda*Bw

    Note: can be validated against `scipy.linalg.eig(A, b=B)`

    Ref:
    'Eigenvalue and Generalized Eigenvalue Problems: Tutorial (2019)'
    Benyamin Ghojogh and Fakhri Karray and Mark Crowley
    arXiv 1903.11240

    """
    Lam_b, Phi_b = np.linalg.eig(B) # eig decomp of B alone
    Lam_b = np.eye(len(Lam_b))*Lam_b # convert to diagonal matrix of eig vals

    Lam_b_sq = replace_nan(Lam_b**0.5)+np.eye(len(Lam_b))*eps
    Phi_b_hat = np.dot(Phi_b, np.linalg.inv(Lam_b_sq))
    A_hat = np.dot(np.dot(Phi_b_hat.transpose(), A), Phi_b_hat)
    Lam_a, Phi_a = np.linalg.eig(A_hat)
    Lam_a = np.eye(len(Lam_a))*Lam_a

    Lam = Lam_a
    Phi = np.dot(Phi_b_hat, Phi_a)

    return np.diag(Lam), Phi
```

Listing 6: MicroPython implementation of the generalised eigenvalue algorithm in Algorithm 4

```
from ulab import numpy as np

class CCA():
    """
    Canonical Correlation Analysis algorithm for SSVEP decoding. Reference
    signal `Y` is a harmonic reference set with `Nh` harmonics.

    Expects a list of `stim_freqs` corresponding to SSVEP stimulus frequencies.
    Requires sampling frequncy `fs` for the harmonic ref.
    """

    def __init__(self, stim_freqs, fs, Nh=2):
        self.Nh = Nh
        self.stim_freqs = stim_freqs
        self.fs = fs

    def compute_corr(self, X_test):
        result = {}
        Cxx = np.dot(X_test, X_test.transpose()) # precompute data auto correlation matrix
        for f in self.stim_freqs:
            Y = harmonic_reference(f, self.fs, np.max(X_test.shape), Nh=self.Nh, standardise_out=False)
            rho = self.cca_eig(X_test, Y, Cxx=Cxx) # canonical variable matrices. Xc = X^T.W_x
            result[f] = rho
        return result

    @staticmethod # define as static to allow non-CCA instances to use this func
    def cca_eig(X, Y, Cxx=None, eps=1e-6):
        if Cxx is None:
            Cxx = np.dot(X, X.transpose()) # signal covariance matrix
        Cyy = np.dot(Y, Y.transpose())  # reference covariance matrix
        Cxy = np.dot(X, Y.transpose()) # cross covariance matrix
        Cyx = np.dot(Y, X.transpose()) # same as Cxy.T

        M1 = np.dot(np.linalg.inv(Cxx+eps), Cxy) # intermediate result
        M2 = np.dot(np.linalg.inv(Cyy+eps), Cyx)

        lam, _ = solve_eig_qr(np.dot(M1, M2), 20) # solve eig. vals using QR iteration
        return np.sqrt(lam)
```

Listing 7: MicroPython implementation of the CCA algorithm from Algorithm 5

```
from ulab import numpy as np

class UnivariateMsetCCA():
    """
    Multiset CCA algorithm for SSVEP decoding.

    Computes optimised reference signal set based on historical observations
    and uses ordinary CCA for final correlation computation given a new test
    signal.

    Note: this is a 1 channel implementation (Nc=1)
    """

    def __init__(self):
        self.Ns, self.Nt = None, None

    def fit(self, X, compress_ref=True):
        """
        Expects a training matrix X of shape Nt x Ns. If `compress_ref=True`, the `Nt` components
        in optimised reference signal Y will be averaged to form a single reference vector. This
        can be used for memory optimisation but will likely degrade performance slightly.
        """
        if X.shape[0] > X.shape[1]:
            print("Warning: received more trials than samples. This is unusual behaviour: check X")
        R = np.dot(X, X.transpose()) # inter trial covariance matrix
        S = np.eye(len(R))*np.diag(R) # intra-trial diag covariance matrix

        lam, V = solve_gen_eig_prob((R-S), S) # solve generalised eig problem
        w = V[:, np.argmax(lam)] # find eigenvector corresp to largest eigenvalue
        Y = np.array([x*w[i] for i, x in enumerate(X)]) # store optimised reference vector Nt x Ns
        self.Y  = Y
        if compress_ref:
            self.Y = np.mean(Y, axis=0).reshape((1, max(Y.shape))) # this will average Nt components in Y: Nc x Nt -> 1 x Nt

    def compute(self, X_test):
        if self.Y is None:
            raise ValueError("Reference matrix Y must be computed using `fit` before computing corr")
        if len(X_test.shape) == 1:
            X_test = X_test.reshape((1, len(X_test)))
        return CCA.cca_eig(X_test, self.Y)[0] # use ordinary CCA with optimised ref. Y
```

Listing 8: MicroPython implementation of the MsetCCA algorithm from Algorithm 7

```
from ulab import numpy as np

class GCCA():
    """
    Generalised canonical component analysis.

    Expects the target frequency at `f_ssvep`. `fs` is the sampling rate used and `Nh` the number of harmonics for the harmonic t

    Ref: 'Improving SSVEP Identification Accuracy via Generalized Canonical Correlation Analysis'
    Sun, Chen et al
    """

    def __init__(self, f_ssvep, fs, Nh=3, name=None):
        self.Nc, self.Ns, self.Nt = None, None, None
        self.Nh = Nh
        self.w_chi_bar_n = None
        self.w_Y_n = None
        self.w_Chi_n = None
        self.fs = fs
        self.f_ssvep = f_ssvep

        self.name = name or "gcca_{0}hz".format(f_ssvep)


    def fit(self, X):
        """
        Fit against training tensor X.

        X should be a 3rd order tensor of dim (Nc x Ns x Nt)
        """
        assert len(X.shape) == 3, "Expected 4th order input data tensor: Nc x Ns x Nt"
        self.Nc, self.Ns, self.Nt = X.shape

        Chi_n = X
        Chi_n_c = Chi_n.reshape((self.Nc, self.Ns*self.Nt))

        Chi_bar_n = np.mean(Chi_n, axis=-1) # mean over trials for each channel with all samples: output shape is Nc x Ns x 1
        Chi_bar_n_c = np.concatenate([Chi_bar_n for i in range(self.Nt)], axis=1) # concat along columns

        Y_n = cca_reference([self.f_ssvep], self.fs, self.Ns, Nh=self.Nh).reshape(-1, self.Ns)
        Y_n_c = np.concatenate([Y_n for i in range(self.Nt)], axis=1)

        # form X and D and find eigenvals
        X = np.c_[Chi_n_c.T, Chi_bar_n_c.T, Y_n_c.T].T

        d1 = Chi_n_c.dot(Chi_n_c.T)
        d2 = Chi_bar_n_c.dot(Chi_bar_n_c.T)
        d3 = Y_n_c.dot(Y_n_c.T)
        D = block_diag(d1, d2, d3)

        lam, W_eig = solve_gen_eig_prob(X.dot(X.T), D) # solve generalised eigenvalue problem

        i = np.argmax(np.real(lam))
        w = W_eig[:, i] # optimal spatial filter vector with dim (2*Nc + 2*Nh)

        w_Chi_n = w[:self.Nc] # first Nc weight values correspond to data channels
        w_Chi_bar_n = w[self.Nc:2*self.Nc] # second Nc weights correspond to Nc template channels
        w_Y_n = w[2*self.Nc:] # final 2*Nh weights correspond to ref sinusoids with harmonics

        self.w_chi_bar_n =  w_Chi_bar_n.T.dot(Chi_bar_n)
        self.w_Y_n = w_Y_n.T.dot(Y_n)
        self.w_Chi_n = w_Chi_n

    def compute(self, X_test):
        if self.w_chi_bar_n is None:
            raise ValueError("call `.fit(X_train)` before performing classification.")

        rho1 = correlation(self.w_Chi_n.T.dot(X_test), self.w_chi_bar_n)[0]
        rho2 = correlation(self.w_Chi_n.T.dot(X_test), self.w_Y_n)[0]

        return np.sum([np.sign(rho_i)*rho_i**2 for rho_i in [rho1, rho2]])
```

Listing 9: Python implementation of the GCCA algorithm from Algorithm 6

Bibliography

[1] B. Bromm, "Human brain electrophysiology. evoked potentials and evoked magnetic fields in science and medicine," English, *Pain (Amsterdam)*, vol. 39, no. 3, pp. 371–372, Dec. 1989. DOI: 10.1016/0304-3959(89)90058-4.

[2] S. Baillet, J. C. Mosher, and R. M. Leahy, *Electromagnetic brain mapping*, ID: 1, 2001. DOI: 10.1109/79.962275.

[3] M. Teplan, "Fundamental of eeg measurement," *MEASUREMENT SCIENCE REVIEW*, vol. 2, Jan. 2002.

[4] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, *Brain–computer interfaces for communication and control*, English, 2002. DOI: 10.1016/S1388-2457(02)00057-3. [Online]. Available: https://dx.doi.org/10.1016/S1388-2457(02)00057-3.

[5] Z. Lin, C. Zhang, W. Wu, and X. Gao, "Frequency recognition based on canonical correlation analysis for ssvep-based bcis," *IEEE transactions on biomedical engineering*, vol. 53, no. 12, pp. 2610–2614, 2006.

[6] A. S. Varnavas, "Signal processing methods for eeg data classification," Ph.D. dissertation, 2008.

[7] D. Zhu, J. Bieger, G. Garcia Molina, and R. M. Aarts, "A survey of stimulation methods used in ssvep-based bcis," *Computational intelligence and neuroscience*, vol. 2010, 2010.

[8] G. Hakvoort, B. Reuderink, and M. Obbink, "Comparison of psda and cca detection methods in a ssvep-based bci-system," *Centre for Telematics & Information Technology University of Twente*, 2011.

[9] M. Panju, "Iterative methods for computing eigenvalues and eigenvectors," *arXiv preprint arXiv:1105.1185*, 2011.

[10] Y. T. Wang, Y. Wang, and T. P. Jung, "A cell-phone-based brain-computer interface for communication in daily life," vol. 8, 2011. DOI: 10.1088/1741-2560/8/2/025018.

[11] L.-D. Liao, C.-Y. Chen, I.-J. Wang, S.-F. Chen, S.-Y. Li, B.-W. Chen, J.-Y. Chang, and C.-T. Lin, "Gaming control using a wearable and wireless eeg-based brain-computer interface device with novel dry foam-based sensors," *Journal of neuroengineering and rehabilitation*, vol. 9, no. 1, pp. 1–12, 2012.

[12] L. F. Nicolas-Alonso and J. Gomez-Gil, "Brain computer interfaces, a review," English, *Sensors (Basel, Switzerland)*, vol. 12, no. 2, pp. 1211–1279, 2012. DOI: 10.3390/s120201211. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/22438708.

[13] H. Tanaka, T. Katura, and H. Sato, "Task-related component analysis for functional neuroimaging and application to near-infrared spectroscopy data," *NeuroImage*, vol. 64, pp. 308–327, 2013.

[14] M. D. Vos, M. Kroesen, R. Emkes, and S. Debener, "P300 speller bci with a mobile eeg system: Comparison to a traditional amplifier," *Journal of neural engineering*, vol. 11, no. 3, p. 036 008, 2014.

[15] Y. Zhang, G. Zhou, J. Jin, X. Wang, and A. Cichocki, "Frequency recognition in ssvep-based bci using multiset canonical correlation analysis," *International journal of neural systems*, vol. 24, no. 04, p. 1 450 013, 2014.

[16] M. Nakanishi, Y. Wang, Y.-T. Wang, and T.-P. Jung, "A comparison study of canonical correlation analysis based methods for detecting steady-state visual evoked potentials," *PloS one*, vol. 10, no. 10, e0140703, 2015.

[17] S. Xie, C. Liu, K. Obermayer, F. Zhu, L. Wang, X. Xie, and W. Wang, "Stimulator selection in ssvep-based spatial selective attention study," *Computational Intelligence and Neuroscience*, vol. 2016, 2016, ISSN: 16875273. DOI: 10.1155/2016/6410718.

[18] R. Chai, G. R. Naik, S. H. Ling, and H. T. Nguyen, "Hybrid brain–computer interface for biomedical cyber-physical system application using wireless embedded eeg systems," *Biomedical engineering online*, vol. 16, no. 1, pp. 1–23, 2017.

[19] J. Chen, D. Zhang, A. K. Engel, Q. Gong, and A. Maye, "Application of a single-flicker online ssvep bci for spatial navigation," *PLoS ONE*, vol. 12, 5 May 2017, ISSN: 19326203. DOI: 10.1371/journal.pone.0178385. [Online]. Available: /pmc/articles/PMC5451069/%20/pmc/articles/PMC5451069/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5451069/.

[20] L. Chu, J. Fernández-Vargas, K. Kita, and W. Yu, "Influence of stimulus color on steady state visual evoked potentials," English, in, ser. Intelligent Autonomous Systems 14. Cham: Springer International Publishing, Feb. 2017, pp. 499–509, ISBN: 9783319480350. DOI: 10.1007/978-3-319-48036-7_36. [Online]. Available: http://link.springer.com/10.1007/978-3-319-48036-7_36.

[21] T. Uktveris and V. Jusas, "Development of a modular board for eeg signal acquisition," *Sensors*, vol. 18, no. 7, p. 2140, 2018.

[22] V. Uurtio, J. Monteiro, J. Kandola, J. Shawe-Taylor, D. Fernandez-Reyes, and J. Rousu, "A tutorial on canonical correlation methods," English, *ACM computing surveys*, vol. 50, no. 6, pp. 1–33, Jan. 2018. DOI: 10.1145/3136624. [Online]. Available: http://dl.acm.org/citation.cfm?id=3136624.

[23] B. Ghojogh, F. Karray, and M. Crowley, "Eigenvalue and generalized eigenvalue problems: Tutorial," *arXiv preprint arXiv:1903.11240*, 2019.

[24] ——, "Eigenvalue and generalized eigenvalue problems: Tutorial," *arXiv preprint arXiv:1903.11240*, 2019.

[25] G. Acampora, P. Trinchese, and A. Vitiello, "Classifying eeg signals in single-channel ssvep-based bcis through support vector machine," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2020, pp. 2305–2310.

[26] P. Autthasan, X. Du, J. Arnin, S. Lamyai, M. Perera, S. Itthipuripat, T. Yagi, P. Manoonpong, and T. Wilaiprasitporn, "A single-channel consumer-grade eeg device for brain-computer interface: Enhancing detection of ssvep and its amplitude modulation," English, *IEEE sensors journal*, vol. 20, no. 6, pp. 3366–3378, Mar. 2020. DOI: 10.1109/JSEN.2019.2958210. [Online]. Available: https://ieeexplore.ieee.org/document/8926475.

[27] X. Duart, E. Quiles, F. Suay, N. Chio, E. García, and F. Morant, "Evaluating the effect of stimuli color and frequency on ssvep," English, *Sensors (Basel, Switzerland)*, vol. 21, no. 1, p. 117, Dec. 2020. DOI: 10.3390/s21010117. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/33375441.

[28] S. Kanoga, M. Nakanishi, A. Murai, M. Tada, and A. Kanemura, "Robustness analysis of decoding ssveps in humans with head movements using a moving visual flicker," vol. 17, 2020. DOI: 10.1088/1741-2552/ab5760.

[29] V. Peterson, C. Galván, H. Hernández, and R. Spies, "A feasibility study of a complete low-cost consumer-grade brain-computer interface system," English, *Heliyon*, vol. 6, no. 3, e03425, Mar. 2020. DOI: 10.1016/j.heliyon.2020.e03425. [Online]. Available: https://dx.doi.org/10.1016/j.heliyon.2020.e03425.

[30] C. M. Wong, B. Wang, Z. Wang, K. F. Lao, A. Rosa, and F. Wan, "Spatial filtering in ssvep-based bcis: Unified framework and new improvements," *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 11, pp. 3057–3072, 2020.

[31] X. Zhao, C. Liu, Z. Xu, L. Zhang, and R. Zhang, "Ssvep stimulus layout effect on accuracy of brain-computer interfaces in augmented reality glasses," English, *IEEE access*, vol. 8, pp. 5990–5998, 2020. DOI: 10.1109/ACCESS.2019.2963442. [Online]. Available: https://ieeexplore.ieee.org/document/8947980.

[32] G. Acampora, P. Trinchese, and A. Vitiello, "A dataset of eeg signals from a single-channel ssvep-based brain computer interface," *Data in Brief*, vol. 35, p. 106 826, 2021.

[33] H. K. Lee and Y.-S. Choi, "Enhancing ssvep-based brain-computer interface with two-step task-related component analysis," *Sensors*, vol. 21, no. 4, p. 1315, 2021.

[34] R. Miao, L. Zhang, and Q. Sun, "Hybrid template canonical correlation analysis method for enhancing ssvep recognition under data-limited condition," in *2021 10th International IEEE/EMBS Conference on Neural Engineering (NER)*, IEEE, 2021, pp. 65–68.

[35] Q. Sun, M. Chen, L. Zhang, X. Yuan, and C. Li, "Improving ssvep identification accuracy via generalized canonical correlation analysis," in *2021 10th International IEEE/EMBS Conference on Neural Engineering (NER)*, IEEE, 2021, pp. 61–64.

[36] Z. Vörös, *The ulab book*, 2021. [Online]. Available: https://micropython-ulab.readthedocs.io/en/latest/index.html.

[37] F. Zhu, L. Jiang, G. Dong, X. Gao, and Y. Wang, "An open dataset for wearable ssvep-based brain-computer interfaces," *Sensors (Switzerland)*, vol. 21, pp. 1–17, 4 Feb. 2021, ISSN: 14248220. DOI: 10.3390/s21041256.

[38] D.-K. Electronics, *Esp32-wroom-32e*. [Online]. Available: https://www.digikey.co.uk/en/products/detail/espressif-systems/ESP32-WROOM-32E-16MB/11613166.

[39] S. M. Fernandez-Fraga, M. A. Aceves-Fernandez, J. C. Pedraza-Ortega, and S. Tovar-Arriaga, "Eeg signal analysis methods based on steady state visual evoked potential stimuli for the development of brain computer interfaces: A review fernandez-fraga et al issn 2349-7238." [Online]. Available: www.pubicon.co.in.

[40] D. P. George, *Micropython - python for microcontrollers*. [Online]. Available: https://micropython.org/.