Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Projektarbeit (Informatik)

Speak your mind! Brain Computer Interfaces for Communication

Autoren	Lucian Nicca Michael Häseler
Hauptbetreuung	Thilo Stadelmann
Nebenbetreuung	Ricardo Chavarriaga
Datum	23. Dezember 2022

Abstract

Communication with patients suffering from locked-in syndrome, a condition where most if not all motor functions of the body have been lost, has been and still is a tremendous challenge in medicine. In recent decades there has been a surge in the development of brain computer interfaces (BCIs) that can be used as a means to overcome said challenge by recording and interpreting the electrical activity of the user's brain. This project's goal was to design and implement a BCI communication system with a flexible, decoupled architecture so that future changes and enhancements can be easily developed. This system consists of three main parts: The data acquisition hardware, a preprocessing & classification pipeline and a front end application with which the end user interacts. As a proof of concept two different front ends, a single stimulus visualiser and a typewriter application, based on steady-state visually evoked potentials (SSVEPs) have been developed and used to test the system and its architecture. Using a canonical correlation analysis (CCA) based classification approach the system was able to achieve a high accuracy when tested using the single stimulus visualiser. This high accuracy however did not carry over to the full system test in conjunction with the typewriter application.

Preface

Nearly every day we interact with computers and various electronic devices. Being able to control such devices with just our minds is such a fascinating concept. Being able to use our love and skills in computer science to achieve something so seemingly futuristic drove us to this project.

We would like to express our sincere gratitude to our supervisors Prof. Dr. Thilo Stadelmann and Dr. Ricardo Chavarriaga for giving us the opportunity to conduct this project, for their valuable guidance and patience throughout the entire project, and for their profound knowledge and insights in the field of brain-computer interfaces and neuroscience.

Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Erklärung betreffend das selbständige Verfassen einer Projektarbeit an der School of Engineering

Mit der Abgabe dieser Projektarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Projektarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:	Unterschriften:
Winterthur, 23. Dezember 2022	[illegible]
	[illegible]

Das Original dieses Formulars ist bei der ZHAW-Version aller abgegebenen Projektarbeiten zu Beginn der Dokumentation nach dem Abstract bzw. dem Management Summary mit Original-Unterschriften und -Datum (keine Kopie) einzufügen.

Contents

1 Introduction

1.1 Motivation

Typical interaction with a computer or electronic device happens through some input device like a touchscreen, mouse and keyboard or even our voice. But what if it isn't possible for a person to use those types of input devices due to a disability or missing limbs? That is where brain-computer interfaces (BCIs) try to provide a solution by allowing people to interact with computers or electronic devices using only their thoughts. BCIs can therefore aid people with disabilities in their day to day life by letting them control prosthetic limbs [1], wheelchairs [2] or typewriting applications [3].

But Brain-computer interfaces also have use cases outside their utilities in the medical field. For example is it possible to enhance attention and memory capabilities through playing a neurofeedback computer game [4]. Another BCI based project provides a headset, which can be used to reduce stress and improve engagement of employees [5]. It is even possible to control drones with BCIs and perform races between participants [6].

All of this can be achieved by measuring brain activity, extracting features from that activity and converting those to outputs with which a device can be controlled. There are two types of BCI, invasive and non-invasive. The focus of this project is on the non-invasive BCIs, in particular Electroencephalography (EEG)-based BCIs. Which is a method to measure spontaneous electrical brain by attaching electrodes to the scalp. In the recorded data it is possible to perceive visual and auditory responses induced by generated stimuli. By carefully selecting the presented stimulus or stimuli control sequences can be defined.

In this project EEG-based BCIs are used in a context of writing letters or words by generating a visual stimulus for each letter and evaluating the measured brain activity to find out what stimuli a user was looking at. It is discussed what solutions for said application already exist, how they work and what solution was implemented.

1.2 Project objectives

The scope of this project comprises of literature study on the current state of BCI communication and typewriting implementations, an implementation of a typewriting application and of testing the accuracy, ease of use and performance of that implementation.

The literature research provides knowledge about current implementations and limitations of BCI interfaces and what kind of typewriting implementations have been done. The focus there is a comparison of the different typing paradigms weighing their advantages and disadvantages. Using that knowledge a typewriter application implementing one of the paradigms is developed, which main objective it is to be a expandable, adaptable demonstrator of the functionalities a BCI can achieve in this context.

2 Theoretical background

This chapter serves as an overview of the relevant topics that are required to understand the domain this project finds itself in.

2.1 Brain-computer interface

A brain-computer interface (BCI) is a system that records and interprets electrical signals produced by neurological activities in a brain. The recorded signal is usually sent through a processing chain that includes preprocessing, feature extraction and classification. The classified signal is then usually utilized to manipulate an application or a hardware device which in turn provides the user of the BCI with feedback. [7]. This sequence of steps is generally known as the *BCI Loop* (see Figure 2.1).

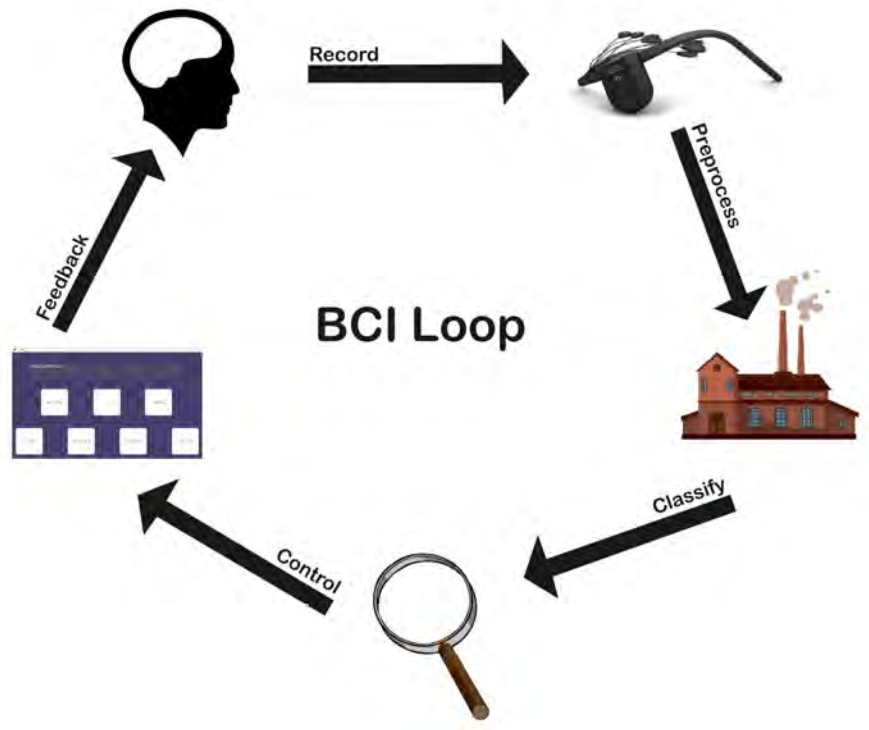

Figure 2.1: Depiction of the BCI Loop.

There is a wide range of various BCI types, ranging from invasive devices, that need to be inserted into the body via surgery, to non-invasive ones. This project made use of a non-invasive electroencephalography (EEG) — *enképhalos* meaning brain, *gráphein* meaning writing — based device to record signals. EEG is a method with which voltage fluctuations on the scalp can be measured using a set of methodically placed electrodes. These electrodes can be used dry but generally it is preferred to use conductive gel to increase the signal quality.

Even though it is commonly preferred to cover the whole scalp with electrodes [7], one might only be interested in the activity confined to a certain area of the brain. For example, if a BCI application is stimulating brain activity using visual events, only the occipital lobe might be of interest. In such a case the unwanted electrodes can be easily filtered out during preprocessing.

2.2 Typewriter Applications

Typewriter applications are one of numerous different kinds of BCI applications that are being developed nowadays. As one of the project's requirements was to create such a typewriter application, this section serves as an overview of the various typewriter implementations, also called paradigms, that have been developed.

2.2.1 P300 paradigm

The P300 paradigm relies on the fact that when a subject is given a set of elements to look at, for example alphanumeric symbols, which randomly light up, the subjects brain produces an event-related potential (a response to an external stimulus) with a latency of about 300 milliseconds when the gazed at element of the set lights up [8]. This potential is typically most evident in the EEG signals measured by electrodes covering the parietal and occipital lobe.

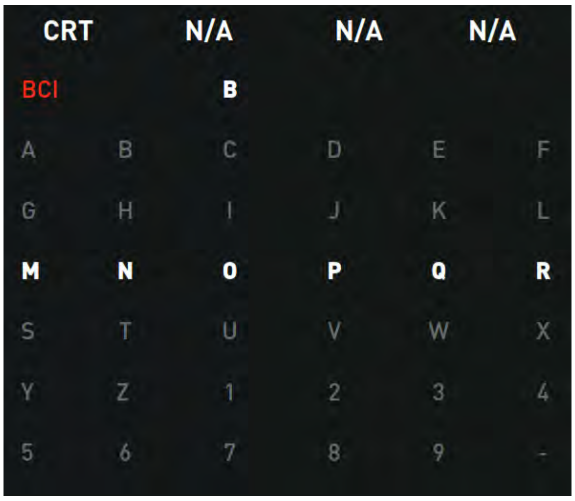

Figure 2.2: Graphical user interface for a P300 typewriter [9].

Figure 2.2 displays one of the earliest designs of a graphical user interface (GUI) used in conjunction with the P300 paradigm. The idea is that the rows and columns of the displayed matrix flash in a random order and if the subjects target letter is among those lit up letters a P300 response is evoked and measured. Based on the timing of the response one can determine which letter the subject targets and subsequently write it to the display, as is the case with the letter *B* at the top of the depiction [9].

2.2.2 SSVEP paradigm

Steady-state visual evoked potentials (SSVEPs) is, as the name implies, a technique to stimulate brain activity by supplying a subject with a visual stimulus that changes its state with a constant frequency, for example a blinking light-emitting diode (LED) [10].

As depicted in Figure 2.3, the application makes use of target surfaces akin to clickable buttons in conventional GUI applications. Each of these target surfaces flicker with a distinct frequency, often called *target frequency*, that evokes a measurable response in the subject's brain in the same frequency band as the target surface's flicker frequency. Similarly to the P300 paradigm the evoked potential is most prominent in electrodes the covering the parietal and occipital lobe.

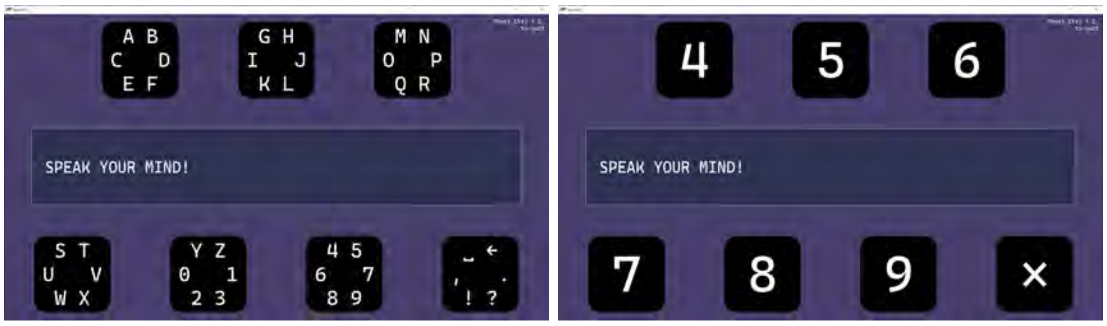

Figure 2.3: Graphical user interface for an SSVEP typewriter.

The application utilizes a two-stage selection process where in the first stage the user has to select one of the seven symbol groups by gazing at it. When a group gets selected the group's symbols are distributed over six of the seven target surfaces — a back button appears on the seventh — which in turn can be selected again by the subject to either write a symbol or go back to stage one.

2.2.3 Motor imagery paradigm

The motor imagery paradigm differs from P300 and SSVEP in that it does not require any external stimulation in order to work. Subjects imagine moving a body part which activates areas of the brain that are responsible for movement [11]. This is be measured using EEG, magnetoencephalography (MEG), and functional Magnetic Resonance Imaging (fMRI) based methods of which the former is more applicable due to its lower experimental cost and spontaneous and evoked signals are used in EEG-based BCIs [12].

The GUI depicted in Figure 2.4 was designed in a similar fashion to the SSVEP example in Section 2.2.2 where symbols can be chosen in a two-stage process. The subject could then for example control the green arrow by imagining right-hand movement and select the target the arrow points to by imagining left-foot movement [9].

2.3 Steady-state visually evoked potential

In this section SSVEP is further elaborated due to it being applied in the project which will be covered in Section 3.3. As the P300 and motor imagery paradigms are not applied in the

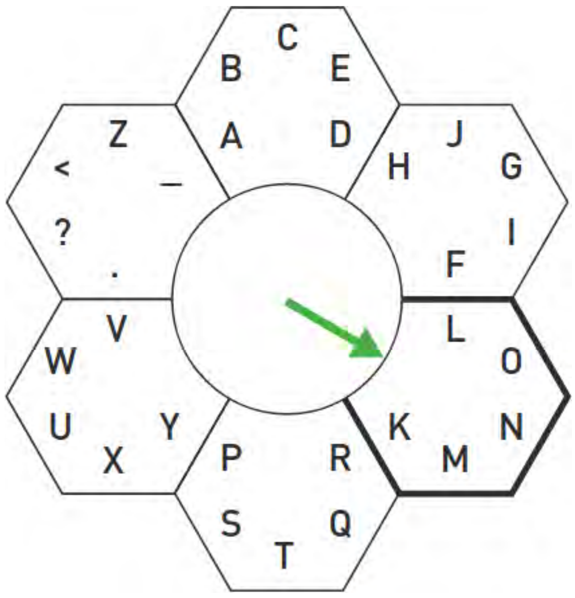

Figure 2.4: Graphical user interface for a motor imagery typewriter [9].

project no detailed explanation is done.

As already touched upon in Section 2.2.2, SSVEP is a phenomenon where the brain responds to a perceived visual stimulus oscillating at a steady frequency with an increase in amplitude in the EEG signal in the same frequency band as well as in its harmonics [7] — a harmonic being a multiple of a base frequency. For example, if a brain perceives a light flashing at a frequency of 15 Hz, one can observe amplitude spikes in the EEG signal at the frequencies 15 Hz, 30 Hz etc.

When designing SSVEP based BCI systems some limitations concerning the number of usable target frequencies arise. Due to that circumstance that SSVEPs are reflected not only in the base frequency, but also in its harmonics, it is undesirable to have target frequencies being harmonics of each other as this might lead to misclassifications with certain classification algorithms (see Figure 2.5).

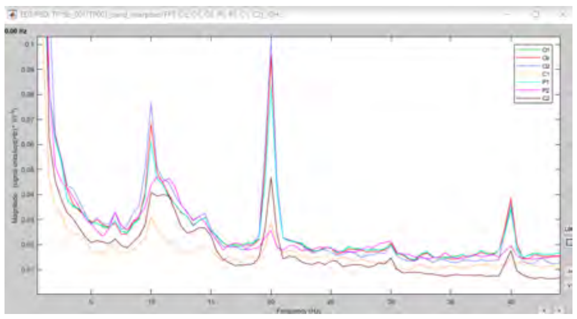

Figure 2.5: Display of harmonics in frequency band for 10 Hz SSVEP [10].

Furthermore, one needs to consider the facts that EEG signals of awake subjects reside in the frequency band between 1 Hz to 30 Hz and that the frequencies need to be integer fractions of the computer monitors refresh rate.

This limits the amount of usable target frequencies drastically. However, some of these limitations can be circumvented by applying techniques like phase shifting, which allows the system to use the same frequency for different stimuli at the same time [13].

Due to the flickering nature of SSVEP acting as a trigger stimulus is ineligible for people with photosensitive epilepsy due to the high risk of seizures.

2.4 Canonical correlation analysis

In order to detect SSVEP in EEG signals, frequency detection algorithms can be utilized. A common choice is the canonical correlation analysis (CCA) method [14], [15], which was employed for developing this project's typewriter application.

Generally, CCA tries to find correlations between two sets of variables, $x(t)$ and $y(t)$, rather than only two variables in classical correlation analysis.

Lin, Zhang, Wu, *et al.* [14] exploits this method to detect SSVEPs in EEG signals. The idea is to let $x(t)$ be the EEG signal and $y(t)$ the Fourier decomposition of the target frequency's harmonics:

$$y(t) = \begin{pmatrix} \sin(2\pi f t) \\ \cos(2\pi f t) \\ \sin(4\pi f t) \\ \cos(4\pi f t) \\ \sin(6\pi f t) \\ \cos(6\pi f t) \\ \vdots \end{pmatrix}, \qquad t = \frac{1}{S}, \frac{2}{S}, \ldots, \frac{T}{S} \tag{2.1}$$

Equation 2.1 shows said Fourier decomposition where f is the target frequency, T is the amount of sampling points and S the sampling rate.

CCA will be able to find a significantly higher correlation to $y(t)$ when an SSVEP occurs as the amplitude in the EEG signal spikes at the target frequency and at the harmonic frequencies as opposed to when no SSVEP occurs.

3 Methods

This chapter discusses the applied tools for building the system, employed algorithms and implementation details of the system, as well as how the system tests were set up in order to test the system performance.

3.1 Prerequisites

The main requirement for this project was to build a modular implementation of a typewriter BCI application with the *Python* programming language. Other than that there were no constraints set on how the system should be realised.

The time required to get the system up and running should be as short as possible as the goal is to use it in demonstrations. Furthermore, it is preferable to not need extensive training for both the subject using the system as well as the classification algorithm. All this ensures that the system is as accessible and easy to use as possible.

3.2 Tools

This section describes all tools used during the project's development.

3.2.1 OpenViBE

OpenViBE [16] is a software suite that enables a fast set up of BCI experiments thanks to its intuitive graphical pipeline programming interface. It has wide driver support for a multitude of EEG devices out of the box, including the *Unicord Hybrid Black* (see Section 3.2.3) EEG headset. All relevant steps of acquiring data, filtering, processing, classifying and visualizing EEG signals can be done in real time.

This tool was employed to quickly experiment with new ideas that arose during development as well as to acquire and verify the EEG signal captured by the EEG device.

3.2.2 Lab Streaming Layer

Lab Streaming Layer (LSL) is a software that enables its users to write sensor data to a stream from which consumers can read. It supports a wide variety of programming languages, such as Python, C# and Java, and runs as an independent process on the operating system. Processes can read from and write to streams locally or over a network [17].

This software was used as a communication channel between all parts of the developed BCI system.

3.2.3 Unicorn Hybrid Black

The EEG device used for measuring brain activity is a commercial headset by *g.tec medical engineering GmbH*, specifically the *Unicorn Hybrid Black* product. The headset measures EEG signals sampled with a bit depth of 24 bit and a frequency of 250 Hz [18]. For this project only channels 5, 6, 7, and 8 were of interest as they are positioned on the parietal and occipital lobe as the application uses visual events.

3.3 Choice of BCI paradigm

When planing a BCI system, the decision as to what kind of paradigm it should follow is the foundation for all further development. During this project's planing phase three paradigms were considered: the P300, the SSVEP and the motor imagery paradigm.

Motor imagery was quickly ruled out as it requires subject training [19] which would not be always feasible, especially when one wants to demonstrate the system during an exhibition.

P300 and SSVEP both require no subject training, however SSVEP has been shown to have greater performance concerning the information transfer rate [19]. Another positive aspect of SSVEP is that, due to its nature of not being a reaction based paradigm, it does not rely on correct timing between what is happening on the subject's screen and the time of signal classification.

Due to the aforementioned reasons it was decided to create a system with SSVEP in mind.

3.4 System architecture

As modularity was an important requirement for the system architecture, it was decided to design it in an object-oriented way. This allows the decomposition of the system into small decoupled components, which promotes the ability to introduce component-local changes that do not affect the system as a whole as well as replaceability of components as long as the interfaces of the components stay the same.

To achieve high modularity, the architecture was split up into two high level components: a *back end* component, comprising of submodules concerned with the EEG signal acquisition, preprocessing and classification, and a *front end* component that would be the actual application that reacts to the classifications, a typewriter, for example. The two components would communicate via a separate communication channel, where the back end would notify the front end as soon as a successful classification has been made.

3.4.1 Back end architecture

The highest level component of the back end system is a *Pipeline*, whose responsibility is to conduct the flow of the EEG data throughout the system, i.e. pass it to the correct subcomponents in the correct order.

For the implementation of a BCI typewriter system, four subcomponents were identified: a *Signal Aggregator*, responsible for aggregating the EEG signal and pass it to the back end system in a usable format; a *Preprocessor*, responsible for preprocessing the EEG signal to remove noise from the signal and remove unwanted channels; a *Classifier* whose purpose is to classify the EEG signal; and a *Notifier* that communicates asynchronously with the front end component.

Figure 3.1 depicts how the Pipeline component controls the data flow between all subcomponents. Furthermore, one can see how the Signal Aggregator component is the system's ingress boundary, handling incoming EEG signals, and the Notifier component is the system's egress boundary, handling the communication with listening front end systems.

The instantiation of all components is delegated to the back end system's main class *Application*. This serves to keep the business logic as free of dependencies as possible, thus leading to a more loosely coupled system which in turn improves the system's modularity.

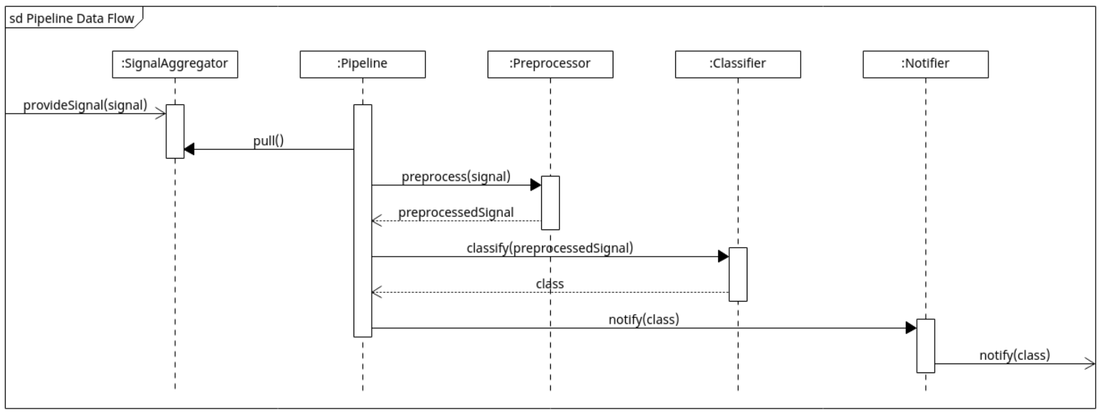

Figure 3.1: A depiction of the Pipeline sequence diagram.

The subcomponents have been designed to be replaceable as long as the interfaces that the Pipeline component expects are adhered to. As seen in Figure 3.2 the subcomponents' consist of only one method for each subcomponent. This is by design to promote ease of use and decoupling.

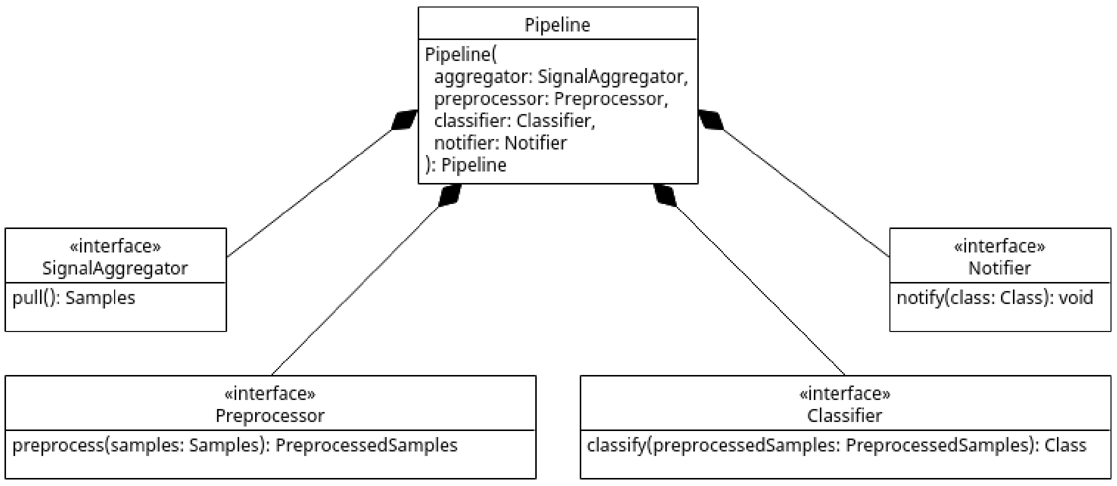
Pipeline
Pipeline(
aggregator: SignalAggregator,
preprocessor: Preprocessor,
classifier: Classifier,
notifier: Notifier
): Pipeline
«interface»
SignalAggregator
pull(): Samples
«interface»
Notifier
notify(class: Class): void
«interface»
Preprocessor
preprocess(samples: Samples): PreprocessedSamples
«interface»
Classifier
classify(preprocessedSamples: PreprocessedSamples): Class

Figure 3.2: A depiction of the class diagram of the Pipeline and its subcomponents' interfaces.

3.4.2 Front end architecture

The front end architecture loosely follows the common *Model-View-Controller* (MVC) pattern. The *Experiment Controller* is in charge of managing the whole application and its program flow. It houses an instance of the *Experiment View* and the *Experiment Model* and reads data from the separate communication channel.

The structure of the Experiment View is based on components, meaning that, for example, a square shaped component can be combined with a text component to create a new combined component. By employing this technique, one can create visual stimuli for evoking SSVEPs by additionally adding a flickering behaviour to said component.

The layout of the components as well as their content is managed by the Experiment Controller. It continuously listens on the communication channel for classifications sent by the back end system, which it then forwards to the Experiment Model, that in turn updates its internal state based on the received classification. This way the Experiment Model always holds the current state of the application that is mirrored by the Experiment View.

3.5 System implementation

This section explores how the concepts described in Section 3.4 have been implemented and explains why certain approaches were favoured over others.

At the project's inception it was decided to implement the back end as well as the front end system with Python due to its highly portable nature — since its executable artifacts run in a virtual machine [20] — and good library support for BCI development.

3.5.1 Back end implementation

This section discusses how each individual component of the back end system has been implemented and reasons choices made.

Pipeline

The Pipeline component is the heart of the back end system and thus implements the main program loop. Thanks to the narrow interfaces of the components it handles its implementation is rather compact, only managing the the communication between its subcomponents.

LSL Signal Aggregator

The Signal Aggregator's responsibility is to capture the recorded EEG signal and supply the back end system with that data.

It is implemented as a component that reads a predefined amount of samples from an LSL stream that is populated by OpenViBE with the recorded EEG signal. The amount of samples is calculated by multiplying the EEG headset's sampling frequency by a time window in seconds, which both are defined during the components instantiation. The ability to overlap the EEG signal instead of pulling separate chunks of samples each cycle has not yet been implemented but is a plausible future addition for this component.

The decision to use LSL to transfer the EEG data to the back end application was a natural one as it had already been successfully employed that way in conjunction with OpenViBE and the Unicorn Hybrid Black EEG headset during the knowledge acquisition phase of the project.

Identity Preprocessor

The Preprocessor's responsibility is to clean the incoming signal from any noise and highlight important features.

It's current implementation does not manipulate the signal in any way as the preprocessing is done through OpenViBE. This decision was made in order to allocate more time to the development of other system parts. This means that the back end system currently already receives preprocessed signals instead of raw ones.

However, if one were to implement the Preprocessor component in-system in the future, a sensible approach would be to implement multiple small cohesive preprocessing components using the *pipes-and-filters* architectural pattern [21], where e.g. multiple spectral and spatial filters could be chained together in a flexible way in order to form a complete processing chain.

CCA Classifier

The Classifier's responsibility is to receive an EEG signal — possibly preprocessed — and classify it using a classification algorithm.

Due to the front end system being an SSVEP typewriter, a statistical method was used for classifying the signals, as such classifiers have shown great success with such systems in the past [14]. The canonical correlation analysis method was chosen in the end as it seemed like a robust choice due to many articles and papers covering its application in BCI settings [14], [15].

The Classifier component makes use of a subcomponent called *Canonical Correlation Analysis* that maps the EEG signal to a list of correlation coefficients. Each correlation coefficient corresponds to one of the front end system's target frequencies. This allows the Classifier component to detect SSVEP peaks in the EEG signal by looking for correlation coefficients that exceed a certain adjustable threshold and pick the target frequency with the largest one.

For future performance enhancements of the Classifier component's implementation one could develop a more sophisticated CCA based algorithm. See Nakanishi, Wang, Wang, *et al.* [15] for a comparison of different CCA based algorithms.

LSL Notifier

The Notifier's responsibility is to send classifications to the front end system.

Three different implementation options were identified that could be used to implement this component. The first option was to build it as a TCP server that communicates over IP. Front end systems could then establish a connection with this server and subsequently receive classification notifications. This solution would allow communication on a local machine as well as over a network, which would allow the back end and front end systems to run on different machines.

The second option was to develop the Notifier component in a way where it would communicate with front end systems through interprocess communication on a single machine. This would allow very fast communication speeds at the cost of tighter coupling between the back end and front end systems.

The third option was to use LSL and write the classifications into a stream from which front end systems could read.

Eventually, after exploring all of the above approaches, the LSL variant was chosen. The two main reasons for this decision were ease of implementation and that no new technology needed to be added to the back end system, keeping its complexity at bay.

3.5.2 Front end implementation

Implementation of the front end is done with the help of *PsychoPy*, an application for the creation of experiments in behavioural science [22]. The application also provides a Python interface to build the experiment in a Python application.

The decision to use PsychoPy was mainly because of its automatic synchronization of displayed frames with the refresh rate of the monitor. With this feature in place it is guaranteed that there are no skipped frames or stutters in the front end GUI which otherwise would disrupt the smooth display of frequencies.

An alternative to PsychoPy was *pygame* [23], a Python library for writing video games. While testing the front end it was not possible to achieve a synchronous display of the GUI frame rate and the screen refresh rate. There were randomly skipped frames, which were very noticeable when compared with the otherwise consistent flickering of the stimulus target.

Typewriter

The typewriter application uses a multilayer design, as discussed in Section 2.2.2. The application can either be started as a version with seven stimuli which uses two layers as seen in Figure 2.3 or as a version with four stimuli and three layers pictured in Figure 3.3. Implementation of the flickering is achieved by only drawing the white version of the stimulus component on every nth frame, e.g. drawing the white stimulus every 5th frame on a 60 Hz monitor would result in a 12 Hz flickering frequency.

The application frequently reads on the LSL stream for incoming classifications. Behind each frequency a control action is defined for the current state, with which the action on selection and transition to the next state is defined, e.g. selecting of a character group or writing a character. Such an action gets triggered when the same classification is received two times in a row. All the actions also trigger a small transitional period, which shows the made selection and then the next state without flickering to give the user time to decide on a new target.

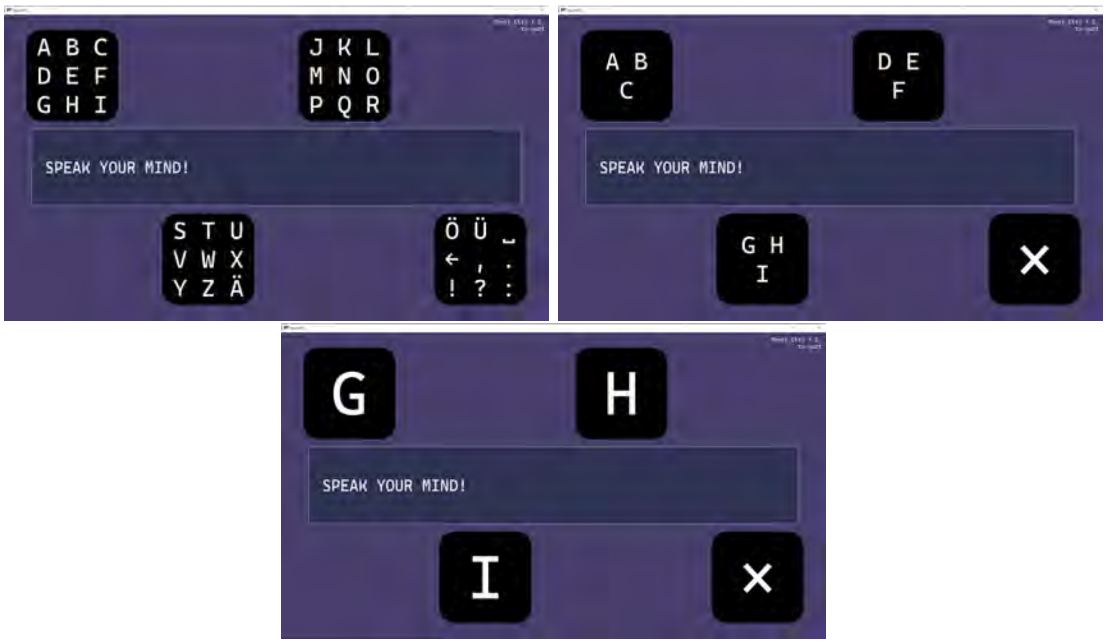

Figure 3.3: Three layers of typewriter interface with four stimuli.

Single stimulus presenter

To test single frequencies a simple GUI was developed which only displays one flickering stimulus as seen in Figure 3.4. The main use of this application is to test the classification algorithm by continuously presenting a steady frequency and being able to change that frequency in the running application.

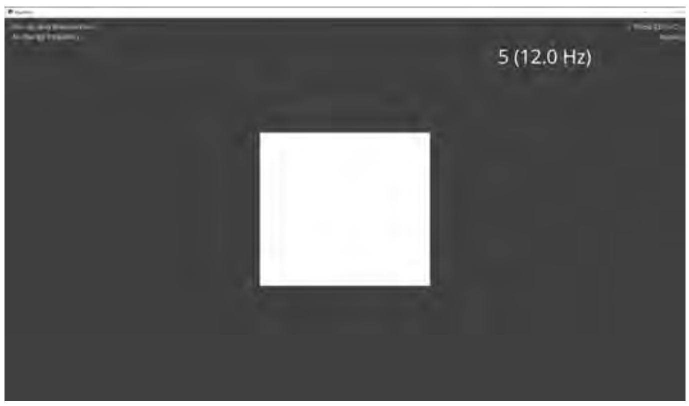

Figure 3.4: Simple GUI with one flickering stimulus.

3.6 System Test

The goal of the system test is to test the accuracy and reliability of the BCI typewriter system in an online scenario. By defining a sentence to type, the number of needed inputs can be determined. When a signal gets misclassified and a wrong action gets executed, the test subject should try to correct it by either going back to the previous view or by deleting the input character with the backspace functionality of the application. The reason for this approach is to represent a real world use of the application as accurately as possible.

The accuracy of the system is then calculated by taking the number of correctly classified inputs and dividing it by the number of total inputs:

$$accuracy = \frac{\text{number of correct inputs}}{\text{number of total inputs}} \tag{3.1}$$

3.6.1 Setup

To achieve consistency and reproducibility the following prerequisites need to be followed.

Firstly, the sentence for the subject to write must be agreed on, so the inputs can be accurately recorded. The test environment should be void of unnecessary distractions. However, the test should still be comparable with a real world scenario so it is to be conducted in normal lighting conditions. The application is to be run in full screen mode. The proper functioning of the EEG headset must be verified by letting the subject clench their teeth and looking for the appropriate response in the EEG signals.

3.6.2 Execution

For the test execution, four pieces of software are required: *OpenViBE Acquisition Server* to connect to the EEG headset and receive data, *OpenViBE Designer* to run the scenario specific to this test, and finally the back end and front end systems developed in this project.

After each prerequisite has been met, the test conductor signals the subject to start writing. While the subject is writing, the conductor keeps track of all commands that are executed and marks which were successful inputs and which ones were missed. When the subject has finished writing all required symbols, the test execution is complete.

3.6.3 Back end system test

To give more insight on certain problems or errors of the system, the back end implementation is additionally tested separately with a simple, single frequency stimulation application. The subject will look at a single stimulus blinking at a certain frequency until a defined amount of classifications were made by the back end system. This is used to investigate the accuracy of the back end system.

4 Results

This chapter discusses the results achieved while testing the front- and back end system following the outline described in Section 3.6.

4.1 Classification accuracy

First tests were conducted with the simple GUI (see Figure 3.4) with a single stimulus to test performance and accuracy of the classification algorithm. Tested were the frequencies 8.57, 10, 12 and 15 Hz. All frequencies were tested with 4 and 2 seconds worth of EEG samples. Each of the tests ran until 20 classification samples were collected.

The test results as shown in Figure 4.1 display a high accuracy for 10 Hz with 100 % accuracy for both 2- and 4-second time windows. Similar high results with the 4-second time window were found for 12 Hz and 8.67 Hz with 100 % accuracy and 90 % accuracy respectively. The worst performance was seen with 15 Hz with only 40 % accuracy in the 4-second time window and only 25 % accuracy in the 2-second time window.

The results of the 15 Hz tests do not align with previous unrecorded tests, where the accuracy was much higher.

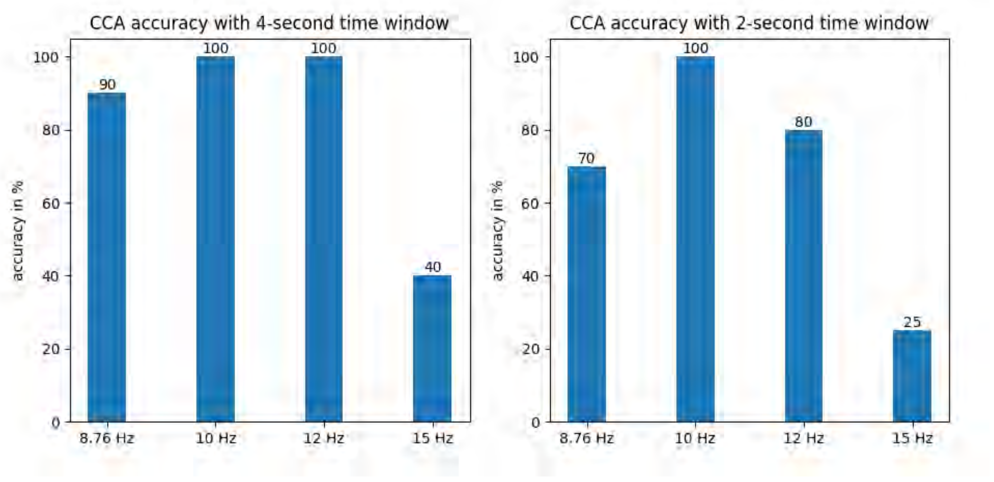

Figure 4.1: Accuracy of CCA classification.

4.2 Typewriter

It was not possible to conduct the complete system tests as defined in Section 3.6 due to the performance of the typewriter system not being consistent enough.

Starting with the main problems faced in the seven stimuli version of the front end application,

The main problem in the seven stimuli version of the front end application was the limitation of usable frequencies in combination with a 60 Hz monitor. This limitation made it only possible to use frequencies which were subharmonics of some smaller frequencies, e.g. 15 Hz and 7.5 Hz. This led to frequent misclassifications of those frequency pairs. When retesting the single frequencies on their own with the single stimulus GUI the classification worked consistently again as shown in Section 4.1.

Concerning the four stimuli version of the front end application which uses fewer frequencies and with that expanding the number of possible frequency combinations, harmonics can be avoided. But even then the classifications were not consistent enough to write something effectively and three layers of selection make each misclassification a costly error to fix, i.e. needing up to three correct classifications to correct a misclassification. Changing the positioning or the frequency allocation of the stimuli in the GUI did not show any significant improvement or deterioration to the performance.

The subpar performance of the typewriter application could have multiple causes. The near proximity of the stimuli may be causing some unwanted interfering evoked potential in the peripheral vision of the subject. There may be an error in the implementation of the CCA or how the stimuli are generated. During testing subjects have noted that they saw stutters in the frequencies they were looking at. This might have been a visual illusion but might point to an error in the front end GUI. but might point to an error in the front end GUI.

5 Discussion

The goal of the project was to develop a highly modular BCI typewriting system that could be employed as a demo at exhibitions as well as act as a framework for future BCI experiments.

We created a decoupled BCI system with SSVEP as the leading paradigm in mind. However it is designed in a way that it not limited to SSVEP alone. Developing applications following other paradigms, such as P300 or motor imagery, with this system as a basis should work just as well. This level of modularity also present in the system's subcomponents: one can easily swap out preprocessors or classifiers to suit one's needs.

However, we are disappointed that we could not get the typewriter itself to work properly. We are certain that we were rather close to a working solution but due to time constraints we were not able to reach it. Our main suspect for the misclassifications discussed in Chapter 4 is that the front end GUI is not capable of displaying multiple flickering stimuli without small stutters.

Furthermore, during the first few weeks of the project we were rather overwhelmed with all the information we had to absorb and make sense of, as we were completely new to this field. This led to situations where we had to experiment with a multitude of different concepts and ideas in order to finally understand them and figure out, if that particular concept or idea is of any use for the project.

Nevertheless we do consider the delivered product a success. It represents a good foundation for future development of BCI systems and due to its modularity supports effortless experimentation with different preprocessing chains and classifiers.

Bibliography

[1] M. Vilela and L. R. Hochberg, “Applications of brain-computer interfaces to the control of robotic and prosthetic arms,” eng, *Handbook of Clinical Neurology*, vol. 168, pp. 87–99, 2020, ISSN: 0072-9752. DOI: 10.1016/B978-0-444-63934-9.00008-1.

[2] K. Matsuzawa and C. Ishii, “Control of an electric wheelchair with a brain-computer interface headset,” in *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, 2016, pp. 504–509. DOI: 10.1109/ICAMechS.2016.7813500.

[3] L. Junwei, S. Ramkumar, G. Emayavaramban, *et al.*, “Brain computer interface for neurodegenerative person using electroencephalogram,” *IEEE Access*, vol. 7, pp. 2439–2452, 2019. DOI: 10.1109/ACCESS.2018.2886708.

[4] K. P. Thomas, A. P. Vinod, and C. Guan, “Design of an online eeg based neurofeedback game for enhancing attention and memory,” in *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2013, pp. 433–436. DOI: 10.1109/EMBC.2013.6609529.

[5] muse. “Muse corperate package.” (2021), [Online]. Available: https://choosemuse.com/corporate/ (visited on 12/20/2022).

[6] H.-E. R. Lab. “Brain-drone race.” (2022), [Online]. Available: http://hxr.cise.ufl.edu/BrainDroneRace/ (visited on 12/20/2022).

[7] M. Clerc, L. Bougrain, and F. Lotte, “Brain-computer interfaces 2: Technology and applications,” in 2016, ISBN: 9781119332428.

[8] L. Farwell and E. Donchin, “Talking off the top of your head: Toward a mental prosthesis utilizing event-related brain potentials,” *Electroencephalography and Clinical Neurophysiology*, vol. 70, no. 6, 1988, ISSN: 0013-4694. DOI: 10.1016/0013-4694(88)90149-6.

[9] A. Rezeika, M. Benda, P. Stawicki, F. Gembler, A. Saboor, and I. Volosyak, “Brain–computer interface spellers: A review,” *Brain Sciences*, vol. 8, no. 4, 2018, ISSN: 2076-3425. DOI: 10.3390/brainsci8040057.

[10] I. Volosyak, H. Cecotti, D. Valbuena, and A. Graser, “Evaluation of the bremen ssvep based bci in real world conditions,” in *2009 IEEE International Conference on Rehabilitation Robotics*, 2009.

[11] R. Abiri, S. Borhani, E. W. Sellers, Y. Jiang, and X. Zhao, “A comprehensive review of eeg-based brain–computer interface paradigms,” *Journal of Neural Engineering*, vol. 16, no. 1, 2019. DOI: 10.1088/1741-2552/aaf12e.

[12] S. N. Resalat and V. Saba, “A study of various feature extraction methods on a motor imagery based brain computer interface system,” *Basic and clinical neuroscience*, vol. 7, no. 1, pp. 13–19, Jan. 2016, ISSN: 2008-126X.

[13] M. A. Lopez-Gordo, A. Prieto, F. Pelayo, and C. Morillas, “Use of phase in brain–computer interfaces based on steady-state visual evoked potentials,” *Neural Processing Letters*, vol. 32, no. 1, 2010, ISSN: 1573-773X. DOI: 10.1007/s11063-010-9139-8.

[14] 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. 54, no. 6, 2007. DOI: 10.1109/TBME.2006.889197.

[15] 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, Oct. 2015. DOI: 10.1371/journal.pone.0140703.

[16] I. Rennes. "Openvibe | software for brain computer interfaces and real time neurosciences." (2015), [Online]. Available: http://openvibe.inria.fr/ (visited on 12/20/2022).

[17] "Lab streaming layer." (2022), [Online]. Available: https://labstreaminglayer.org/ (visited on 12/22/2022).

[18] g.tec medical engineering GmbH. "Technology | unicorn hybrid black." (2022), [Online]. Available: https://www.unicorn-bi.com/brain-interface-technology/ (visited on 12/20/2022).

[19] N. Masoodhu Banu, T. Sujithra, and S. M. Cherian, "Performance comparison of bci speller stimuli design," *Materials Today: Proceedings*, vol. 45, 2021, ISSN: 2214-7853. DOI: 10.1016/j.matpr.2020.11.804.

[20] "Glossary — python 3.11.1 documentation." (2022), [Online]. Available: https://docs.python.org/3/glossary.html#term-bytecode (visited on 12/21/2022).

[21] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, "Pattern-orientierte softwarearchitektur. ein pattern system.," in 1998, ISBN: 9783827312822.

[22] J. Peirce, J. R. Gray, S. Simpson, *et al.*, "PsychoPy2: Experiments in behavior made easy," vol. 51, Feb. 2019. DOI: 10.3758/s13428-018-01193-y. [Online]. Available: https://doi.org/10.3758%2Fs13428-018-01193-y.

[23] "Pygame." (2022), [Online]. Available: https://www.pygame.org/ (visited on 12/22/2022).

List of Figures