Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering
IMES Institut für Mechanische Systeme

Zurich University
of Applied Sciences

School of Engineering
CAI Centre for Artificial Intelligence

VT2 Biomedical Engineering

Bridging Mind and Machine:
A Python Framework for BCI Applications

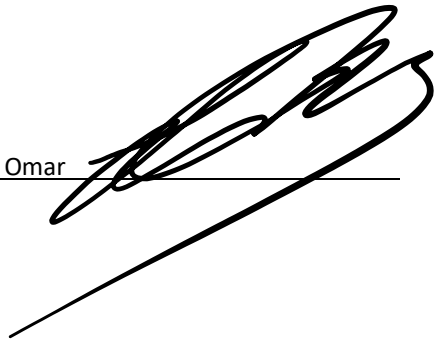
Author	Lone Omar
Supervisor	Prof. Dr. Daniel Baumgartner
Co-Supervisor	Dr. Ricardo Chavarriaga
Date	31.07.2024

Independence of Work

The author(s) confirm(s) that they have carried out this work independently. All external help, aids and the adoption of texts, illustrations and programs from other sources are properly referenced in the work. This is also a binding declaration that the present work does not contain any plagiarism, i.e. no parts that have been taken over in part or in full from another's text or work under pretence of one's own authorship or without reference to the source. In the event of misconduct of any kind, Sections 39 and 40 (Dishonesty and Procedure in the Event of Dishonesty) of the ZHAW Examination Regulations and the provisions of the Disciplinary Measures of the University Regulations shall come into force.

Place, Date: **Name Student:**

Winterthur, 31.07.2024 Lone Omar



Abstract

The growing desire to seamlessly and intuitively interact with technology has driven significant advancements in two key areas, namely Brain-Machine Interfaces (BMIs) and Human-Machine Interfaces (HMIs). BMIs represent the establishment of a direct link between the brain and external devices. Translating thought into action with sensors like EEG, MEG or fMRI. HMIs, on the other hand, encompass a broader range of technologies that facilitate information exchange between humans and machines. These interfaces can be as simple as conventional keyboards and mice or as complex as advanced virtual reality systems with haptic feedback. As these fields continue to evolve, the convergence of human and machine intelligence has the potential to expand various scientific domains, including neurorehabilitation, prosthetic control, and communication for individuals with disabilities.

EEG is a non-invasive neuroimaging technique employed in many BMIs. However, the state-of-the-art techniques are hampered by limitations in current processing tools. Existing solutions either offer predefined processing pipelines with narrow customization options or provide granular control over data handling at the expense of user-friendliness. The interdisciplinary field of neuroengineering, especially within emerging BCI technologies, demands a user-friendly and customizable Python framework that empowers researchers and clinicians to conduct rapid prototyping.

To address the current limitations of current EEG-based BMIs, this project showcases a simplified Python framework for EEG processing in real-time, thereby leveraging established libraries. This approach aims to minimize reliance on third-party graphical software tools and strikes a balance between ease of use and customization. The framework allows for tailored data handling and processing for specific BCI applications.

Development will be a three staged process. The first phase will evaluate existing frameworks and identify suitable components for the technical base layer. The second stage will focus on building the core framework with features for data acquisition, preprocessing, feature extraction, and classification. Finally, the last stage will showcase the framework's capabilities through a real-time motor imagery application controlling a video game and hand prosthesis.

This solution offers an abstraction layer, enabling user-friendly development of customized protocols and facilitating the comparison of algorithms and paradigms on the go, allowing for rapid and customized prototyping.

Keywords:

BCI, EEG, Python, Framework, Customization, Motor Control, Pipeline, Prototyping, Modular, Benchmark

Preface

I would like to express my sincere gratitude to all those who supported me in this project thesis.

I am thankful to Dr. Ricardo Chavarriaga and Prof. Dr. Daniel Baumgartner, who supervised me and gave me a free hand over the project.

Thank you to Prof. Dr. Hans Dermot Doran for his expertise and assistance regarding interfacing and software architecture.

To Khizer Lone and Klotz Van Ziegelstein for their generous offer to proofread this thesis.

To Taquichiri Carlos Rafael Tordoya for his time to pose as photo model.

Table of Contents

0 Glossary, List of Figures and Tables

0.1 Glossary

Abbreviation	Explanation
10-20	The 10-20 system is a standardized method for positioning electrodes on the scalp during EEG recordings
ADHD	Attention Deficit Hyperactivity Disorder Is a neurodevelopmental disorder characterized by inattention, hyperactivity, and impulsivity.
ALS	Amyotrophic lateral sclerosis Is a progressive neurodegenerative disease affecting nerve cells in the brain and spinal cord.
ANN	Artificial Neural Network Is a computational model inspired by the human brain, capable of learning complex patterns.
API	Application Programming Interface A set of programming instructions and standards for accessing a software application.
ASSR	Auditory Steady-State Response A rhythmic brain response generated by continuous presentation of auditory stimuli modulated in amplitude or frequency.
BCI / BMI	Brain-computer interface also called Brain-machine interface Is a direct communication pathway between the brain's electrical activity and an external device
CNN	Convolutional Neural Network Is a deep learning architecture for processing grid-like data.
CSP	Common Spatial Pattern Algorithm for data projection of a signal
ECG	Electrocardiogram Is a recording of the heart's electrical activity over time.
ECoG	Electrocorticography Is an invasive method of placing electrode directly on the surface of the brain to record brain waves.
EEG	Electroencephalography Method for measuring brain signals
EMG	Electromyography Method for measuring muscle activity or muscle potential
ERD / ERS	Event-Related Desynchronization / Synchronization Refers to changes in brainwave patterns associated with specific events or actions.
ERP	Event-Related Potentials Are measurable brain responses directly linked to specific sensory, cognitive, or motor events.
fNIRS	Functional Near-Infrared Spectroscopy Is a non-invasive neuroimaging technique that measures brain activity by detecting changes in blood oxygenation levels using near-infrared
HCI	Human-computer interaction A computer technology, which focuses on the interfaces between users and computers
ICA	Independent Component Analysis Is a statistical method for separating a multivariate signal into its constituent independent components.
IMU	Internal Measurement Unit An electronic device that measures forces acting on it like acceleration or angular rate.
IoT	Internet of Things Is a network of physical devices embedded with sensors, software, and connectivity, enabling data collection and exchange.
LDA	Linear Discriminant Analysis Algorithm for class differentiation of a set
LIS	Locked-In Syndrome Is a neurological condition characterized by complete paralysis of voluntary muscles except for eye movement, with preserved consciousness.

LSL	Lab Streaming Layer Is a software framework for synchronizing and streaming real-time data from various sensors in experimental setups.
MEG	Magnetoencephalography Is a non-invasive neuroimaging technique that measures magnetic fields produced by electrical currents in the brain.
N400	Is an event-related potential (ERP) component reflecting semantic processing difficulties, typically elicited by words or phrases that violate semantic or syntactic expectations.
P300	Is an event-related potential (ERP) component elicited by unexpected or salient stimuli, reflecting cognitive processes related to stimulus evaluation and categorization.
RNN	Recurrent Neural Networks Are neural networks designed to process sequential data.
SCP	Slow Cortical Potential Are low-frequency fluctuations in brain electrical activity that can be voluntarily controlled.
SMR	Sensorimotor Rhythms Refers to rhythmic brain activity patterns associated with the sensorimotor cortex, modulated by movement or motor imagery.
SSVEP	Steady-State Visual Evoked Potentials A rhythmic brain response generated in response to a continuously flashing visual stimulus.
SVM	Support Vector Machine Is a supervised machine learning algorithm used for classification and regression by finding the optimal hyperplane that separates data points into different categories.
ZMQ	ZeroMQ Is a high-performance asynchronous messaging library that provides a socket-like interface for network communication.

0.2 List of Figures

0.3 List of Tables

0.4 List of Equations

1 Introduction

1.1 A Brief History of BCI

Brain-Computer interfaces (BCIs), also known as brain-machine interfaces (BMIs)[1], constitute systems that enable direct communication between the brain and external devices. These interfaces typically operate by acquiring brain signals, processing them to extract relevant information, and translating this information into commands that control external systems. Their main goal is to assist, augment or repair human cognitive or sensory-motor functions. The term was first introduced in 1973 by Jacque J. Vidal in his paper "Toward Direct Brain-Computer Communication" [1]. The technical means to measure brain waves were developed even earlier in 1929, when a German scientist named Hans Berger demonstrated that the human brain was producing electrical currents representative of brain activity. These currents could be measured by placing electrodes on the scalp, which was the first concept of electroencephalography (EEG). The EEG proved to become a major tool in neuroscience and was used to study cognitive functions and understand and diagnose neuropathologies. Figure 1 provides a brief overview of historical events for brain-computer interfaces since the invention of EEG [2].

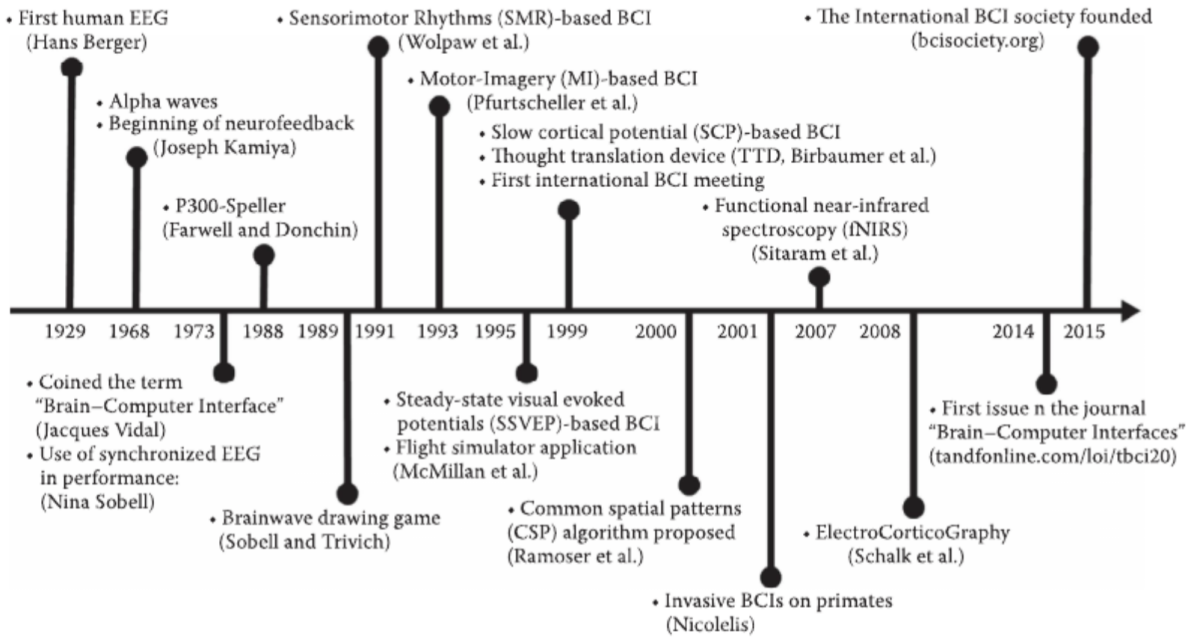


Figure 1: Historical Events for BMI
Showcasing a timeline of major events occurring in the field of BCI applications [3].

As technology has advanced, so too has the processing and storage of big data, and likewise BMI systems and their paradigms. A BCI paradigm is the experimental design or task used to elicit specific brain activity patterns that a BMI system can then interpret and translate into commands [4]. A list of commonly used BCI paradigms is listed in Table 1.

[1] The terms BCI and BMI will be used interchangeably throughout this document and mean the same thing.

Table 1: Overview of Common BCI Paradigms [5]

Name	Description
Event-Related Desynchronization/Synchronization (ERD / ERS)	When a person imagines moving a body part, called motor imagery, the brain's activity in the corresponding motor area changes. ERD reflects a decrease in alpha (8 - 12 Hz) and beta (13 - 30 Hz) rhythms (desynchronization), while ERS indicates an increase (synchronization). These changes can be used to classify intended movements.
Event-Related Potentials (ERP)	These are transient changes in brain electrical activity that occur in response to specific events or stimuli. Examples are P300 or N400. The P300 is a positive deflection occurring around 300 milliseconds after a rare or unexpected stimulus. ERP-based BCIs often use oddball paradigms where infrequent targets are embedded among frequent non-targets.
Steady-State Visual Evoked Potentials (SSVEP)	When a visual stimulus flickers at a specific frequency, it induces a corresponding rhythmic response in the brain. By measuring the frequency of this response, a BCI can determine which stimulus the user is attending to. SSVEP BCIs are often used for menu selection or cursor control.
Auditory Steady-State Response (ASSR)	Similar to SSVEP, but evoked by auditory stimuli. It's a rhythmic response in the brain to a continuous auditory stimulus, modulated in amplitude or frequency.
Slow Cortical Potentials (SCP)	These are very slow fluctuations in brain electrical activity. Users can learn to voluntarily modulate these potentials to control external devices. SCP BCIs often require extensive training.
Sensorimotor Rhythms (SMR)	These are rhythmic brainwaves (8 – 30 Hz) recorded over the sensorimotor cortex. Like ERD/ERS, they can be modulated by motor imagery and used for BCI control.
Hybrid Systems	A combination of multiple paradigms to improve performance, reliability, or user comfort. For example, a system might combine SSVEP for menu selection with motor imagery for cursor control.

BMI systems consist of four components, as can be seen in Figure 2. Signal acquisition captures brain activity through modalities such as electroencephalography (EEG), electrocorticography (ECoG), magnetoencephalography (MEG), or functional near-infrared spectroscopy (fNIRS). Raw neural data is then fed into signal processing pipelines to eliminate artifacts, extract relevant features / patterns, as described in Table 1, and classify brain states. The classification serves as the foundation for generating control signals that interface with external devices. To optimize system performance and user interaction, feedback mechanisms are integrated, providing real-time information about system status and user performance [2], [5] - [8].

BCI systems can be non-invasive or invasive, depending on the signal acquisition device used. Non-invasive devices include EEG, MEG and fNIRS. Where EEG and fNIRS are portable, MEG requires a shielded environment and requires the most expensive equipment. Among the invasive methods are ECoG, where electrodes are placed directly on the brain's surface, and intracortical electrodes. These electrodes are implanted within the brain tissue.

Which method to choose depends on factors such as the desired level of invasiveness, the application, cost and desired signal quality. EEG is the preferred choice for non-invasive, portable applications such as gaming or neurofeedback due to its affordability and safety.

When high temporal resolution is important, as in auditory or visual stimulus studies, MEG is more suitable despite its high cost and the need for a shielded environment. For scenarios demanding portability and safety, fNIRS offers a viable option. In cases of severe motor impairment, ECoG provides superior signal quality compared to non-invasive alternatives, albeit with the added complexity of surgical implantation. Lastly, intracortical electrodes offer the highest signal resolution but demand invasive surgery, limiting their use to research or extreme circumstances where other methods prove insufficient [2], [5], [6], [9].

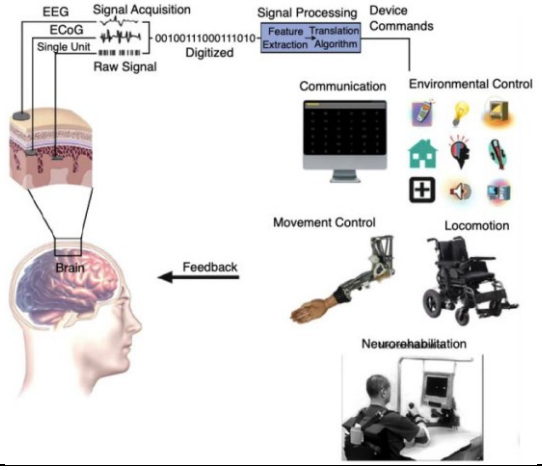


Figure 2: BCI System Components
A BCI system consists of four elements. Signal acquisition, processing, control and feedback. [10]

EEG hardware has become accessible and affordable, like the OpenBCI [11] and Unicorn Hybrid Black [12] EEG devices. The non-invasive nature and reduced cost of EEG devices have solidified their position as the primary choice for BCI applications. [13].

1.2 Brain Waves and the Homunculus

Brain waves are oscillatory patterns generated by the synchronous firing of neuronal ensembles that constitute the foundational building block of brain functions. These electrical waves are categorized into distinct frequency bands, see Table 2. Each band is associated with specific cognitive and physiological states. While delta waves predominate during deep sleep, theta waves are linked to drowsiness and meditative states. Alpha rhythms are characteristic of relaxed wakefulness, whereas beta waves are associated with active attention and cognitive processing. Gamma waves, the fastest of these oscillations, are often linked to higher cognitive functions and consciousness [5], [7], [8].

Table 2: Overview Frequency Bands in Brain Waves
[5], [7]

Name	Range	Correlates to mental condition
Delta	0 – 4 Hz	State of deep sleep
Theta	4 – 8 Hz	Deep relaxation, focus, imagination
Alpha	8 – 12 Hz	Concentration, learning
Beta	12 – 30 Hz	Active thinking, decision-making
Gamma	30 – 70 Hz	Sensing, reading, speaking

The motor cortex is the region responsible for planning and executing voluntary movements. It is situated within the frontal lobe of the brain. A distorted representation of the human body, known as the homunculus, is superimposed on this cortical area. This map illustrates the brain areas responsible for different body parts, see Figure 3. The hands and the face occupy larger cortical regions as they require finer motor and sensory control [14], [15].

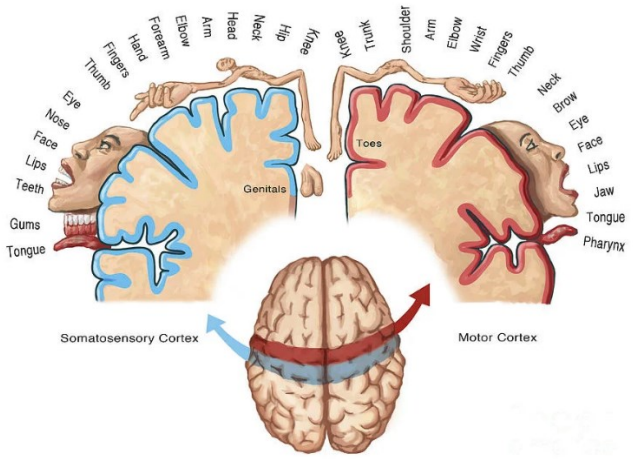


Figure 3: Mapping of the Homunculus and Brain Regions
The map reflects the amount of brain tissue dedicated to each human body part. There are two parts: The sensory homunculus and the motor homunculus. As the name suggest, one is reflective of the somatosensory cortex and the other for the motor. [16]

1.3 Signal Acquisition, Processing and Classification

In the previous chapter, it was mentioned that EEG is typically used for non-invasive BMIs. In this chapter, common standards regarding electrode placement and signal processing are discussed.

To ensure standardized electrode placement and comparability across studies, the 10-20 system is widely adopted. This system divides the scalp into a grid based on percentages of the total skull circumference, with electrodes positioned at specific intervals, see Figure 4. By referencing anatomical landmarks such as the nasion (bridge of the nose) and inion (prominent point at the back of the head), precise electrode locations are determined. This standardized approach enables accurate identification of brain regions associated with different cognitive functions and facilitates data analysis and comparison between studies. The letter in front of the electrode stands for the brain region and the number stands for the distance from the middle line of the brain. The further the electrodes are placed from the that line, the higher the number. Even numbers represent the right hemisphere while odd numbers specify the left. Most EEG devices come with different head caps that have inserts for the electrodes and are labelled [17].

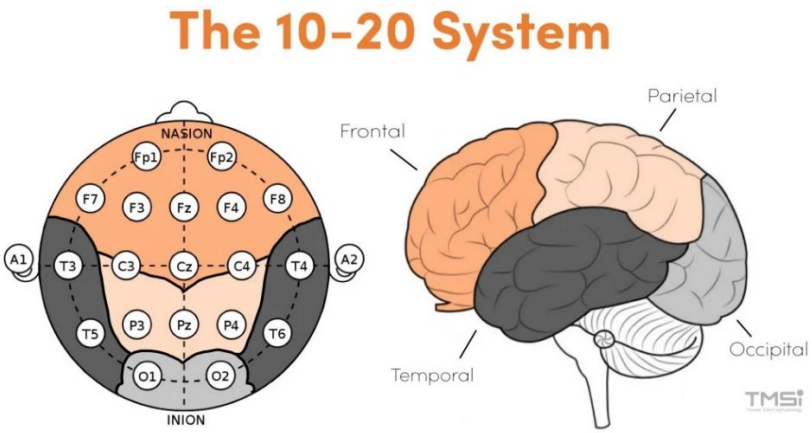


Figure 4: The 10-20 System
A system for standardizing the electrode placement around the head.

Raw EEG signals are small in magnitude and measured in µV. They are contaminated by noise and artifacts such as eye blinks or muscle contractions. Preprocessing techniques, including filtering, mitigate these noise and artifacts. Typically, this involves filtering techniques, such as band-pass filtering to isolate specific frequency bands of interest and notch filtering to remove power-line interference. Additionally, artifact removal methods like Independent Component Analysis (ICA) are employed to eliminate ocular and muscular artifacts. Once the EEG signal is cleaned, feature extraction is performed to identify discriminative patterns indicative of the user's intent. Common feature extraction techniques encompass spectral analysis (e.g., power spectral density), time-frequency analysis (e.g., wavelet transforms), and spatial filtering (e.g., common spatial pattern). These methods extract salient information about brain activity, which is subsequently transformed into a suitable format for classification algorithms [18], [19], [20].

Traditional machine learning methods such as Linear Discriminant Analysis (LDA) and Support Vector Machines (SVM) have been widely employed for their computational efficiency and interpretability. In recent years, the application of neural networks has emerged too. Deep learning models, such as Convolutional Neural Networks (CNNs), Artificial Neural Networks (ANN), and Recurrent Neural Networks (RNNs), have demonstrated promising performance in extracting complex features directly from raw EEG data, bypassing the need for traditional hand-crafted feature engineering. One primary limitation is the requirement for large amounts of training data to achieve optimal results. For motor imagery BCIs most commonly CSP is used for feature extraction and LDA for classification [4], [20], [21].

2 State of the Art

2.1 Clinical Use and BCI Solutions

Non-invasive EEG-based systems have demonstrated potential in neurorehabilitation, enabling individuals with motor impairments, like stroke, amyotrophic lateral sclerosis (ALS), spinal cord injury, locked in syndrome (LIS), or cerebral palsy, to regain control over prosthetic limbs or assistive devices. Additionally, BMIs have found utility in augmentative communication, like speech recognition, providing a means for individuals with severe motor disabilities to interact with computers and communicate through brain signals.

In the realm of neurofeedback, EEG-based systems are employed to train individuals to self-regulate brain activity, demonstrating efficacy in treating conditions such as attention deficit hyperactivity disorder (ADHD), anxiety, and depression [5], [22] - [25].

Invasive BCIs have shown progress in restoring sensory and motor functions. Intracortical implants have enabled paralyzed individuals to control robotic limbs. Furthermore, research into sensory prostheses utilizing cortical implants has yielded results in restoring tactile and visual sensations. Beyond motor and sensory restoration, BCIs are being explored for cognitive enhancement, such as improving memory or attention, and for the treatment of neurological disorders like epilepsy and Parkinson's disease [23] - [29].

The BCI landscape is characterized by a growing number of companies developing innovative technologies [30], [31], [32]. Table 3 provides an overview of some of them.

Table 3: BMI Companies

Name	Description
Non-Invasive	
BrainCo	BrainCo is a cognitive technology company founded at the Harvard Innovation Lab. Leveraging expertise in machine learning, design, and neuroscience, the company develops products that enhance cognitive function and performance [33].
Emotiv	Emotiv is a technology company specializing in EEG. The company develops and commercializes hardware and software for acquiring and processing brain signals. EMOTIV's primary focus is on creating BCIs for applications in various domains like gaming, research, and human-computer interaction [34].
Kernel	Kernel focuses on developing tools for brain measurement. The company developed a fNIRS device called Kernel Flow. This device allows for natural head movement while collecting high-quality brain data. Kernel aims to use this technology to advance brain research, improve patient outcomes, and accelerate the development of new treatments [35].
Muse	Muse specializes in EEG-based meditation and relaxation devices. Muse targets the consumer wellness market. Their products are designed to help users improve focus, sleep, and overall well-being [36].
Neurable	Neurable develops BCI for everyday use. Their current solution is an headphone shaped EEG that monitors the mental state of the user and helps the user focus or calm down with music [37].
Invasive BCI	
Blackrock Neurotech	Blackrock Neurotech has been developing implantable BMIs for decades. Their technology involves arrays of electrodes implanted into the brain to record neural signals. The company provides hardware and software for various BCI solutions [38].
BrainGate	BrainGate develops BMI technologies to assist individuals with neurological conditions or limb loss in regaining communication, mobility, and independence. Their research focuses on providing reliable control over the environment for people with ALS, spinal cord injury, and stroke [39].

Neuralink	Neuralink's goal is to develop a universal brain interface to improve the quality of life for individuals with disabilities and enhance human capabilities. They are developing thin, thread-like electrodes that have to be implanted by a robot [40].
Paradromics	Paradromics is developing a high-performance BCI called Connexus Direct Data Interface (DDI). This technology aims to provide communication and computer control for severely motor-impaired individuals by translating neural signals into speech, text, and cursor movements. The company plans to expand the platform for future applications beyond communication and control [41].
Precision Neuroscience	Precision Neuroscience focuses on developing minimally invasive BCIs. The company's flagship product is the Layer 7 Cortical Interface, a high-resolution device designed to be implanted on the brain's surface without damaging brain tissue. The primary goal is to restore speech and movement for severely paralyzed individuals. Precision Neuroscience is currently conducting clinical trials and aims to launch its first commercial product in 2025. The company also explores potential applications in stroke rehabilitation and depression treatment [42].
Synchron	Synchron is developing a minimally invasive BMI designed to restore touchscreen control for patients with limited hand mobility. The technology involves a stent-based device inserted into the brain via the jugular vein to record neural signals and transmit them to external devices [43].

2.2 Available BMI Toolboxes and Modules

With the rise of BCI research, the development of specialized toolboxes and modules for BCI applications has risen too. These frameworks offer a range of functionalities. Either they focus on one specific function or provide support for all modalities ranging from data acquisition, processing, classification and visualization [44], [45], [46].

BCI++, BioSig, BCI2000, OpenVibe and xBCI are all based on C and C++. BioSig is one of the toolboxes only available for offline analysis. OpenVibe provides a visual way to develop pipelines akin to MathWorks Simulink. While BCILAB, Brainstorm, EEGLAB and FieldTrip are all toolboxes for Matlab. FieldTrip specifies that it is capable of real-time data processing, but the analysis can only be done after the data has been fully recorded [47] - [55].

An overview of the toolboxes is given in Figure 5 and Table 4. The toolboxes described, are stand-alone systems, mostly able to create complete applications from within itself. The column in the table named "Customizable" specifies the option of whether the toolbox provides the option to program custom functions or if only predefined ones are supported. All real-time frameworks, besides EEGSynth support offline processing.

In regard to Python, there are a variety of modules providing an abstraction layer to develop code for BCIs. They can focus on signal acquisition, processing, visualization or marker generation. To interface these different modules with each other, the lab-streaming-layer (LSL) protocol is commonly used. Figure 6 and Table 5 give an overview and provide a description of each module [56] - [72].

Both lists are not complete and focus on toolboxes and modules that are specific to BCI applications. Hence, modules like SciPy, Scikit-Learn or TensorFlow are not included as they target more general use cases.

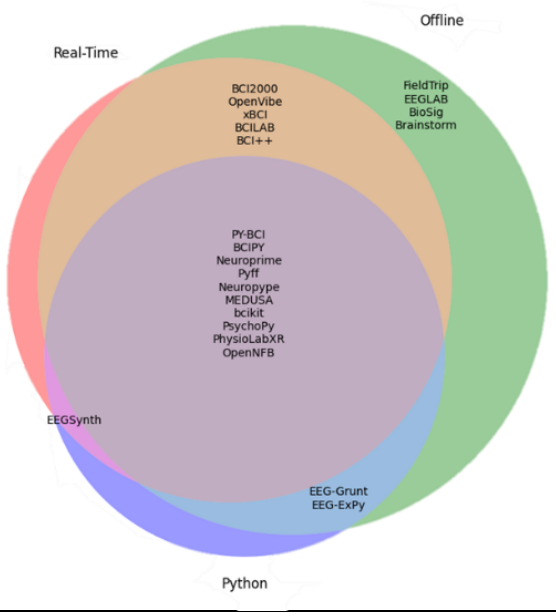


Figure 5: Overview BCI Toolboxes

An overview of major BMI toolboxes sorted between real-time processing capabilities and Python codebase [47] - [55], [73] - [83].

Table 4: Overview BCI Toolboxes
[47] - [55], [73] - [83]

Name	Description	Real-Time Processing	Customizable	Codebase
BCI++	Complete BCI framework in C/C++	YES	YES	C / C++
BCI2000	Complete BCI framework in C++.	YES	YES	C++
bcikit	Modular analysis of biosensor data streams with a visualization component. Only supports OpenBCI hardware	YES	YES	Python
BCILAB	MATLAB Toolbox for BCI. Not actively developed since 2017.	YES	YES	Matlab
BCIPY	BciPy is a library for conducting Brain-Computer Interface experiments in Python	YES	YES	Python
BioSig	Toolbox for offline biosignals analysis	NO	YES	C / C++
Brainstrom	Offline imaging framework for neural activity with Matlab	NO	YES	Matlab
EEG-ExPy	A framework with a collection of classic EEG experiments. Includes generic analyses and visualization	NO	NO	Python
EEG-Grunt	Offline data visualizer and analyzer	NO	NO	Python
EEGLAB	Compete framework, running on Matlab. Can run in standalone mode. No real-time processing	NO	YES	Matlab
EEGSynth	A framework to interface hardware in real-time. There are no offline capabilities.	YES	YES	Python

FieldTrip	Matlab toolbox for BCI. Offline processing	NO	YES	Matlab
MEDUSA	Use premade apps or create your own BCI. Only available on Windows.	YES	YES	Python
Neuroprime	Framework for BCI interfaces. Documentation contains only one example.	YES	YES	Python
Neuropype	Complete Framework to create BCI systems. A visual pipeline designer like OpenVibe. Only an academic license is free	YES	YES	Python
OpenNFB	Complete framework, missing documentation, pure coding	YES	YES	Python
OpenVibe	Framework to create processing pipelines in a visual way similar to Math Works Simulink.	YES	YES	C++
PhysioLabXR	Available for all systems, based on scripts	YES	YES	Python
PsychoPy	Build and run experiments in Python	YES	YES	Python
PY-BCI	A Python package to create real-time experiments. Bound to Pytorch, scikit-learn or TensorFlow	YES	YES	Python
Pyff	Pyff is a Feedback Framework that provides a platform-independent framework to develop BCI feedback applications in Python	YES	YES	Python
xBCI	A generic platform for realizing online BCI	YES	YES	C++

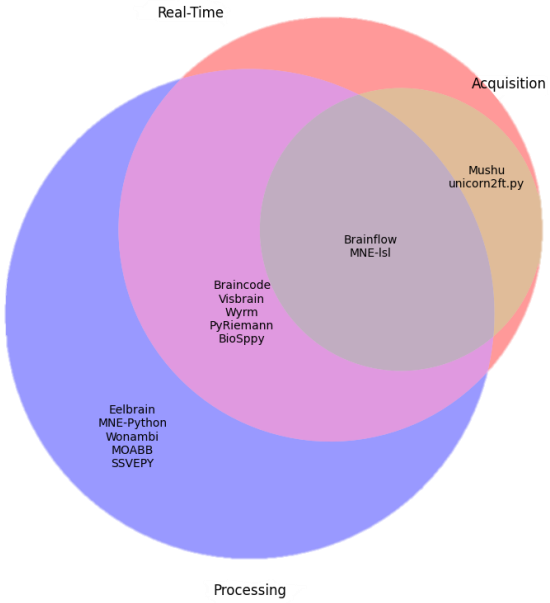


Figure 6: Overview BMI Python Modules

An overview of Python modules to develop BMI applications, split between real-time, signal acquisition and processing capabilities [56] - [72].

Table 5: Overview BMI Python Modules
[56] - [72]

Name	Description	Real-Time	Classification
BioSppy	A toolbox for biosignal processing, written in Python	YES	Processing
Braincode	Braincode is an open-source Python toolbox for decoding raw electrophysiological brain data with deep learning models	YES	Processing
Brainflow	BrainFlow is a library intended to obtain, parse and analyze EEG, EMG, ECG and other kinds of data from biosensors	YES	Acquisition, Processing
Eelbrain	Open-source Python toolkit for MEG and EEG data analysis	NO	Processing
MNE-lsl	A framework for real-time brain signal streaming with MNE-Python	YES	Acquisition, Processing
MNE-Python	Open-source Python package for exploring, visualizing, and analyzing human neurophysiological data	NO	Processing, Visualization
MOABB	Tool for Benchmarking BCI pipelines	NO	Processing
Mushu	A free open-source BCI signal acquisition software written in Python	YES	Acquisition
PyRiemann	PyRiemann is a Python machine learning package based on the scikit-learn API	YES	Processing
SSVEPY	A package to analyze MNE-formatted EEG data for SSVEPs.	NO	Processing
unicorn2ft.py	Python API to get data from Unicorn EEG on other operating systems without the proprietary Python library.	YES	Acquisition
Visbrain	A multipurpose GPU-accelerated open-source suite for brain data visualization	YES	Visualization
Wonambi	Package for the analysis of EEG, ECoG and other electrophysiology modalities	NO	Processing
Wyrm	A Python BCI toolbox for running and processing real-time experiments	YES	Processing, Experiment

2.3 Parallel Frameworks IoT and BCI

The domains of Internet of Things (IoT) and BCIs share a fundamental architecture, despite their distinct application areas. At their core, these systems involve a sequential process of data acquisition, processing, interpretation, and action / control [84]. Figure 7 shows such a pipeline in the Microsoft Azure environment.

IoT systems typically gather data from physical sensors embedded in devices. This data is transmitted to a central system for processing, analysis, and decision-making. Industrial applications of IoT, often aligned with Industry 4.0 principles, focus on optimizing production processes, predictive maintenance, and quality control. For instance, anomaly detection in sensor data can signal equipment malfunctions, while data-driven insights can inform smart maintenance schedules [84] - [87].

- **General Flow:**
  - Data -> Insight -> Action
- **Device:**
  - generating Data
- **Ingestion & provisioning:**
  - insight from data (feature extraction)
- **Hot path:**
  - near real-time, handling is fast, low-latency
- **Warm path:**
  - in batches, hourly, daily
- **Cold path:**
  - timing is not relevant
- **Management & business integration:**
  - Perform an action, do something given insight like store, raise alarm

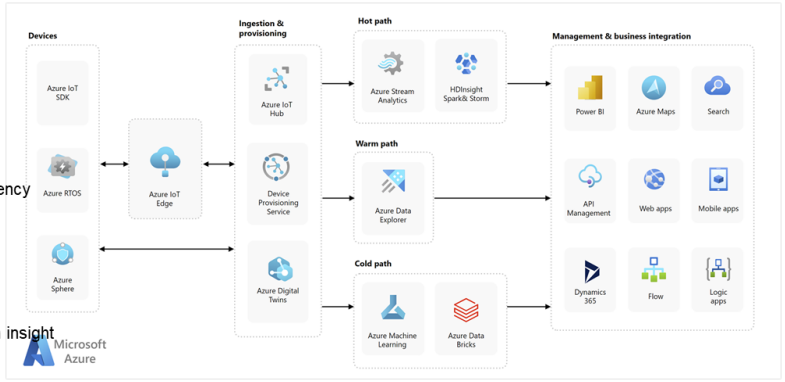


Figure 7: Overview of an IoT Pipeline in Azure

Illustrates an example of an IoT pipeline in the Microsoft Azure cloud environment. Devices collect data. Data is ingested by a hub, and depending on the timing requirement processed over a hot, warm or cold path and finally forwarded to databases and further 3rd party applications [84], [88].

Similarly, BCIs acquire data from the brain and process it to extract relevant features and patterns, as described in chapter 1.1 and 1.3. While the data sources and processing techniques differ, the underlying pipeline is analogous to each other, see Figure 8.

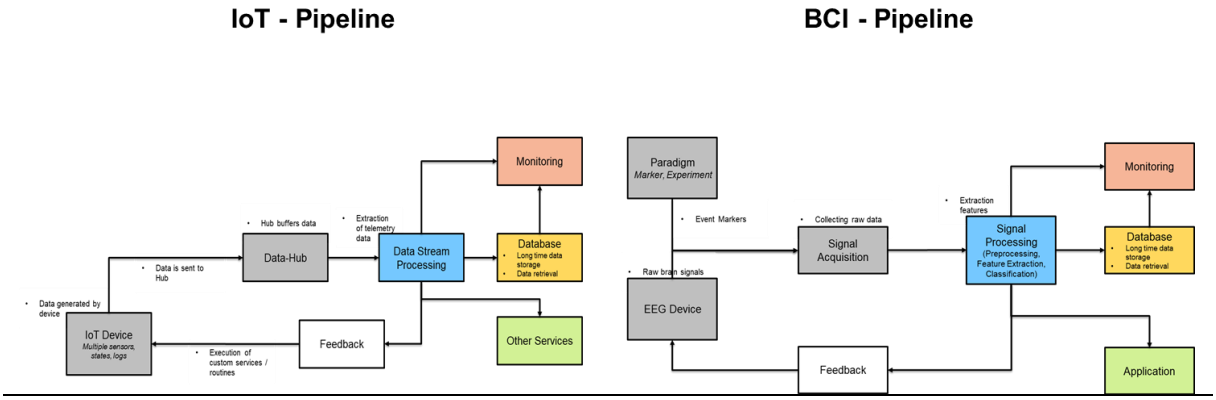


Figure 8: Comparison IoT and BCI Pipeline

A comparison between an IoT and BCI pipeline. Both pipelines deal with signal acquisition, processing, storage and action.

3 Objective

3.1 Aim and requirements

The aim of this thesis is to construct a Python-based framework capable of developing real-time BMI applications. As proof of concept, a motor imagery example will be showcased. The framework should be adaptable and customizable while running in a Python environment. While customizing a specific part of the pipeline, the rest of the pipeline should stay unaffected. Both data scientists and programmers are intended as user groups, and it should provide the tools for data visualization, storage, replay, and processing. To ensure seamless integration within the Python ecosystem, compatibility with existing relevant modules is another wish.

3.2 Motive

For in-house development, a dedicated Python framework for BCI applications is required to accelerate the development of online systems. By providing an abstract and modular structure, the framework facilitates rapid prototyping and customization. Configurable components allow researchers to tailor the system to specific needs, while keeping consistent design principles, enhancing code readability and maintainability. A standardized framework promotes reproducibility and comparison of different realizations for the same BMI paradigms and allows the ability to profile and benchmark the complete pipeline or parts of it to optimize or identify potential bottlenecks.

3.3 Selection

As the overview provided in Chapter 2.2, there are a variety of toolboxes available that can be used as a foundation to create BMI systems. Leveraging these existing tools together, like building blocks and creating the desired framework, is a more pragmatic approach than building everything from the ground up.

Considering the Python toolboxes presented in Figure 5, PY-BCI, BCIPY, MEDUSA, PhysioLabXR and bci-kit offer the tools to create complete BMI pipelines.

Neuroprime [89] and OpenNFB [82] lack documentation and are not considered further. PsychoPy [76] focuses more on the paradigm creation and could be used as an add-on. Neuropype [78] is similar to OpenVibe [50] and provides the user with a similar user interface to MathWorks SimuLink [90]. Neuropype requires a license, and only the academic license is available for free. According to the documentation of PY-BCI [81], the processing module is bound to pytorch, scikitlearn and tensor flow and restricts the user to these modules. Bcikit [79] only supports OpenBCI hardware for real-time acquisition. MEDUSA [77] runs only on Windows machines, which leaves the selection between BCIPY and PhysioLabXR.

Both the tools are similar to each other, allowing the user to create their own pipelines and use their own code with a modular setup. If looking at the activity by the community on improving the tools. BCIPY's last bug fix was in May 2023 [91], while PhysioLabXR last

commit was in July 2024 [92] as of this time of writing. Given that, the selection fell on PhysioLabXR as the backend of the desired BCI framework.

3.4 PhysioLabXR

PhysioLabXR is a software toolkit designed to simplify the real-time processing of multi-modal physiological data in neuroscience and human computer interfaces (HCI). It offers a platform for handling various physiological signals and supports data transfer via Lab Streaming Layer (LSL) [56] and ZeroMQ (ZMQ) [93] protocols. The framework provides functionalities for visualizing, recording, and replaying LSL and ZMQ data streams, enabling users to easily add, inspect, and manage the data. Basic stream processing can be done over the visualization module on the fly. Figure 9 illustrates the backend and data flow in PhysioLabXR. The scripting interface provides an API to handle LSL and ZMQ streams and can handle multiple scripts running. Scripts are run in parallel, and execution time and buffer sizes can be handled via the scripting interface or done in the code itself [80], [94], [95].

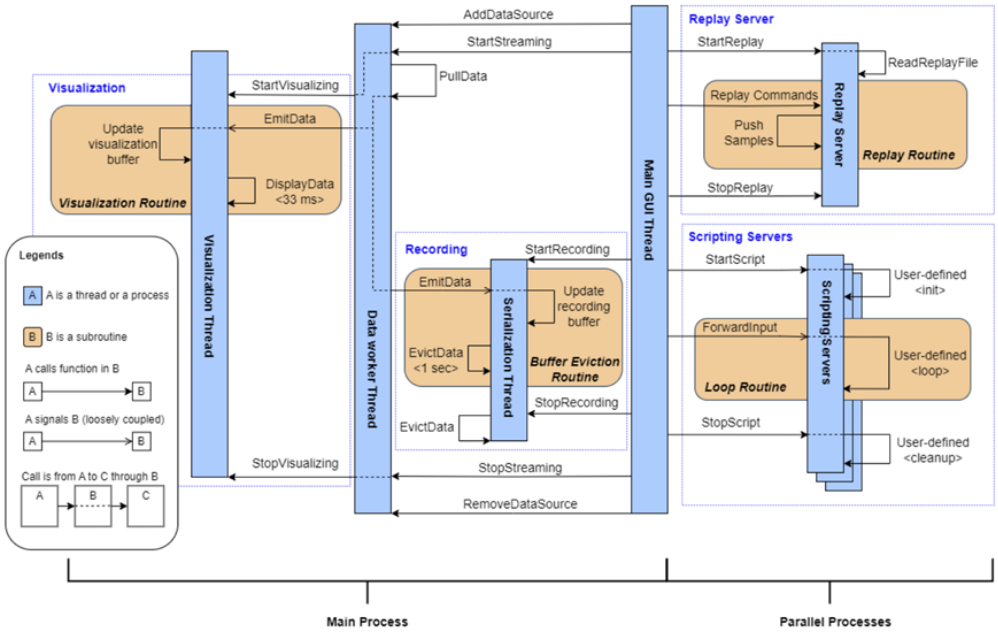


Figure 9: PhysioLabXR Backend

Showcasing a diagram of the backend of PhysioLabXR. The main modules are split between visualization, recording, replay and scripting interfaces [96, p. 4].

Scripts follow a simple sequence. They have an initialize block, where parameters and variables can be defined. After initialization, the loop function is repeatedly called until the process is terminated. Once terminated, the clear block is called and allows deconstructing variables, see Figure 10.

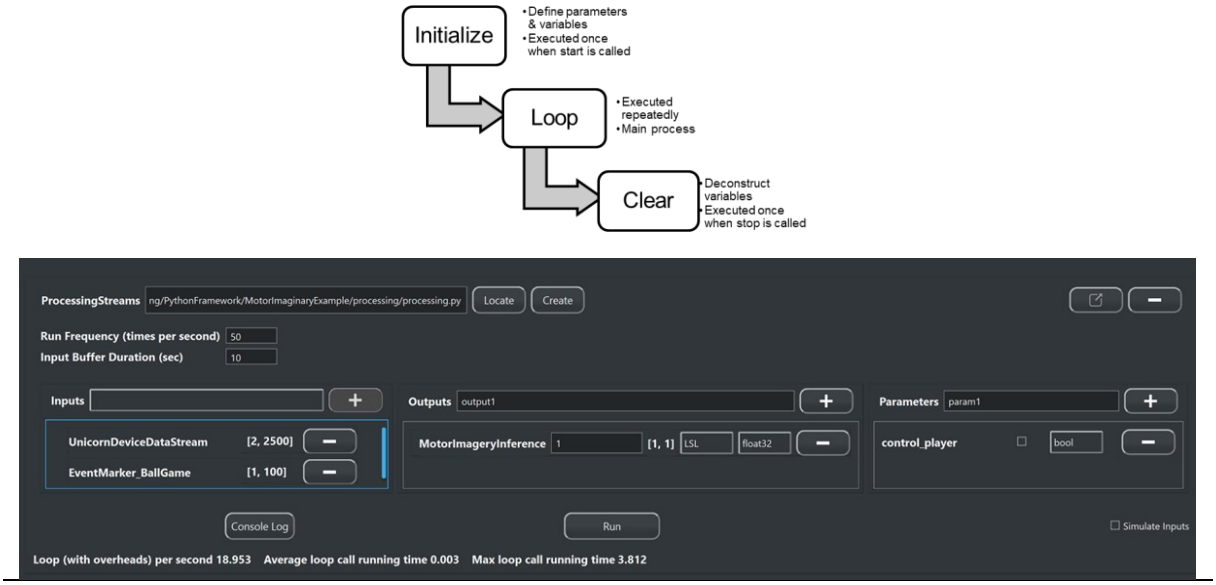


Figure 10: Scripting Interface

At the top: The order in which a script is executed. The loop function is called repeatedly till the script is terminated. Bottom: Showing the interface to add scripts, customize buffer size and run frequency of the script. If desired, input and output streams can be defined, which can then be handled over PhysioLabXR API inside the code. Here, there are two input streams defined and one output stream. Parameters can be defined outside the code too, and can then be accessed inside the program. During run time, these parameters can be changed on the fly. The input streams are coming from the acquisition script and the Unity game. Loop metrics of the script are shown at the bottom during run-time.

4 BCI Framework with PhysioLabXR

4.1 Overview

The goal is to create a modular setup for the system described in Figure 2. One possible realization is shown in Figure 11. The color-coded elements represent a modular pipeline architecture, where each component operates independently, allowing for configuration changes without affecting other modules. To realize this modular pipeline in PhysioLabXR, its scripting interface is used. An acquisition script will focus on acquiring the data from the EEG device, a processing script will handle the time-data-event marker correlation, training and classification. For each application there is to control, there will be a corresponding script written for it. The monitoring / visualization will be done with the internal PhysioLabXR visualization module, where the streams are passed to the corresponding API. The same applies for storage.

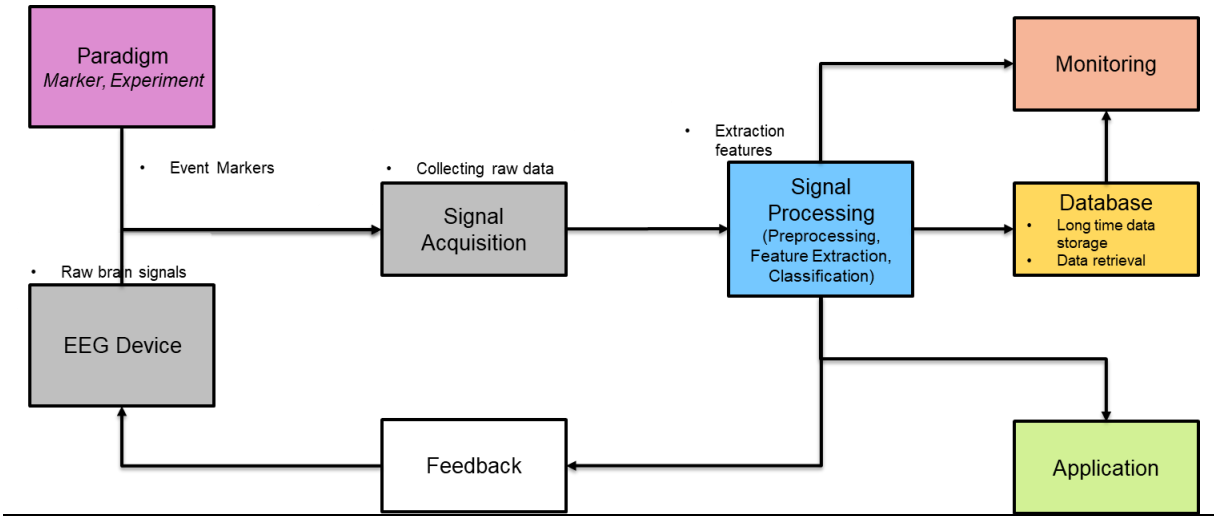


Figure 11: BCI Pipeline

Illustrating the individual parts of a BCI pipeline. The colors show a way to modularize the pipeline, allowing each component to function as an independent process. For instance, the signal processing module requires only EEG data and event markers. EEG device-specific acquisition methods are not of interest in the processing part.

4.2 Motor Imagery Example

To showcase the modularity and customization of this framework. A motor imagery example is demonstrated with the given pipeline architecture illustrated in Figure 11. As mentioned in Chapter 1.3, currently the best classification results are achieved with a CSP filter and an LDA classification. The MNE Python library has an example for decoding motor imagery from EEG data [97]. Further, PhysioLabXR provides some examples BCIs like an ERP classifier, a P300 speller and a motor imagery example [98], [99], [100]. The motor imagery example from PhysioLabXR uses the decoding example from MNE. All the examples are used as a template for the presented motor imagery example. For the event markers and the visual training, the Unity Balance Ball demo is used [101], [102].

As mentioned in the previous chapter, the motor imagery BCI will have a modular setup, see Figure 12. One script will be responsible for the EEG data acquisition. A second script will

implement the correlation, training and classification. For controlling, there will be two additional scripts. One will control the interface to the prosthetic hand and the other will control a virtual gamepad that allows for an interface to general video games.

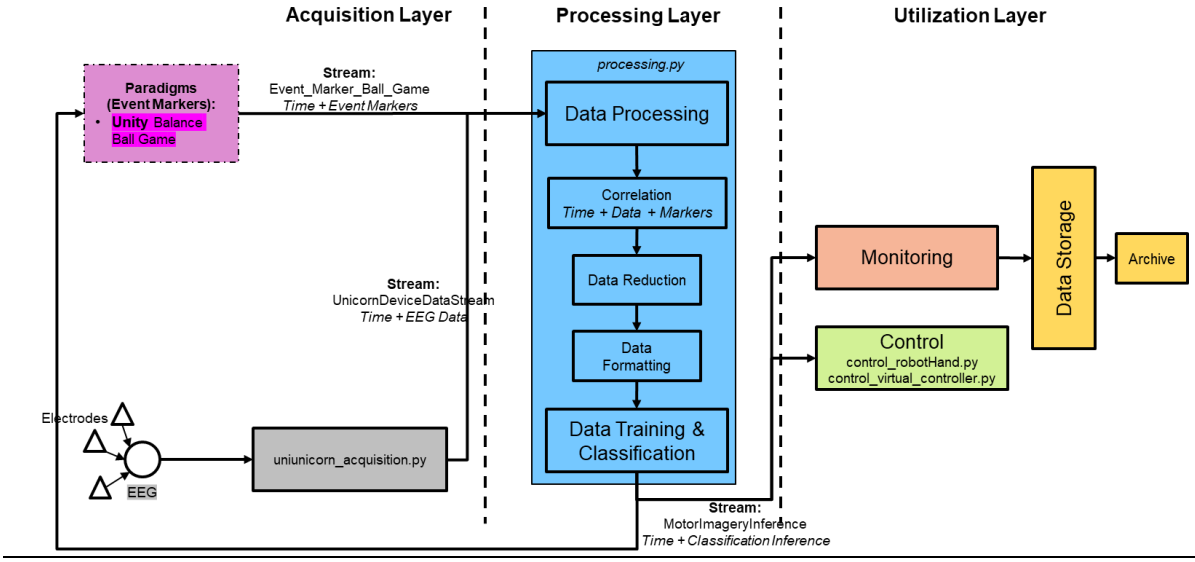


Figure 12: In-Depth Pipeline Overview

For the motor imagery example, the Unity Balance Ball game is used for the event marker (pink). An acquisition script collects the data from the EEG device (gray). In the processing script (blue), time, markers and EEG data are correlated, reduced to only left- and right-hand imagery movement and then formatted so that they can be used by the MNE CSP and Sklearn LDA modules for training and classification. The classification value is then forwarded to three different applications (green), the Balance Ball game, hand prosthesis, and a virtual gamepad. Data storage and monitoring are handled by the built-in functions in PhysioLabXR.

4.3 Acquisition

EEG data acquisition is performed using the Unicorn Black 8-channel EEG system by gtec [12]. The device offers a sampling rate of 250Hz and includes an integrated Inertial Measurement Unit (IMU). The system provides 18 channels in total. It contains EEG signals, acceleration and gyroscope data, the device's battery status, digital I/O, and timestamps. The EEG electrodes are placed according to the 10-20 system standard, with channels located at Fz, C3, Cz, C4, Pz, PO7, Oz, and PO8, see Figure 4 and Figure 13. The data is transmitted via Bluetooth.

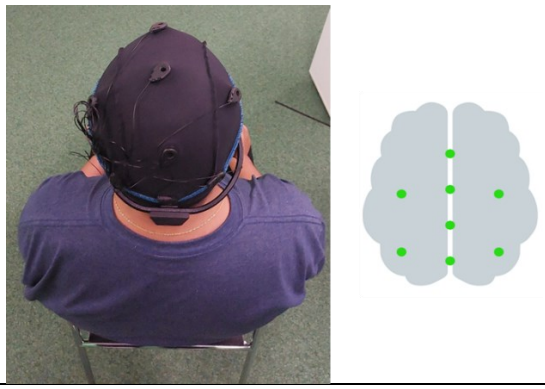


Figure 13: EEG Headset

Showing the EEG headset and the electrode placement according to the 10-20 system.

For accessing the device, gtec provides different APIs. It's possible to acquire data over a C, Python or LSL interface. To decide on the appropriate interface, a profiling was done in Python (cprofile [103]) and in C (Visual Studio [104]). Figure 14 showcases the results. All provided APIs from the manufacturer take about 4ms to acquire 18 channels. When using the BrainFlow API, presented in Chapter 2.2, as described in [105], the acquisition is faster as a sub-process is buffering the data.

Given the results from the profiling and the fact that BrainFlow allows for the configuration of ring buffer sizes [106] and supports a variety of other EEG devices, the BrainFlow API is used to acquire data. For implementation, see the unicorn_acquisition.py script in the digital attachment, in Chapter 9. The script outputs the eight-channel EEG data on an LSL stream called UniCornDeviceDataStream, see Figure 15. The hardware samples at 250Hz the acquisition script is called every 100Hz, and the average loop run-time was 1ms.

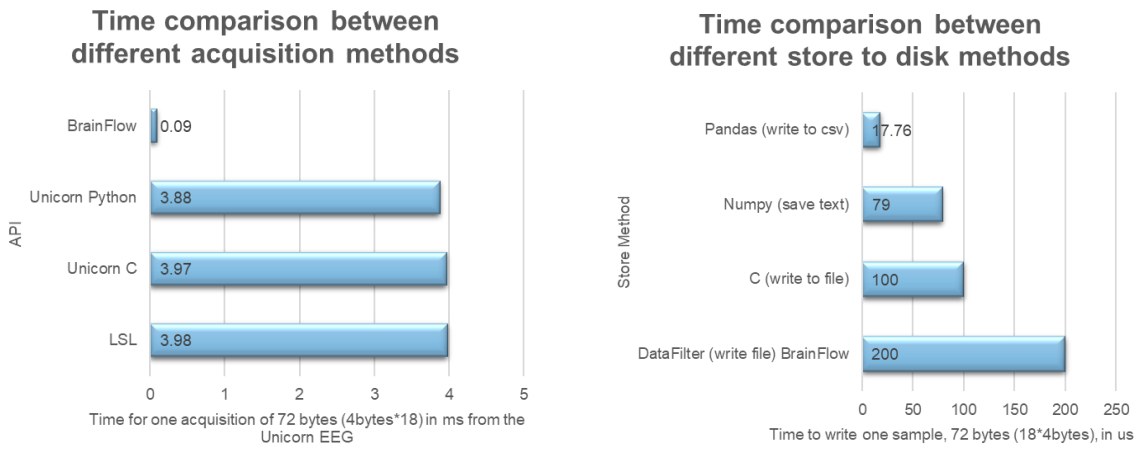


Figure 14: Profiling Acquisition

Results of acquiring data over different APIs. Left for acquisition and right for storage methods. The BrainFlow API is the fastest as it buffers the data in a subprocess while the other APIs directly acquire the data. For storage, the panda data frame stores the fastest as it only opens the file once.

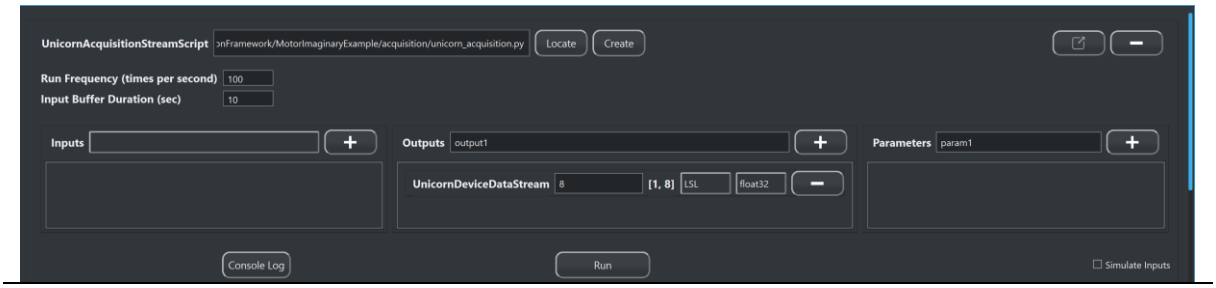


Figure 15: Acquisition Script

The acquisition script will run at 100Hz, has a buffer duration size of 10 seconds, and outputs an eight-channel stream over LSL called UnicornDeviceDataStream. This stream can be used by the visualization module of PhysioLabXR and other scripts.

4.4 Balance Ball Paradigm

For the event markers and training, the Unity balance ball game is used, which is available for download, see [102]. An in-depth description can be found here [101]. The game has a training mode and a play mode. To summarize, the training works as follows: A ball is shown

on top of a bar and randomly goes to the left or right. When going to the left, a marker with the value of two is sent out over LSL. The ball rolls for five seconds in that direction and then gets teleported back to the middle. When the five seconds are over, a marker value of minus two is sent. For the right-hand side, the marker value corresponds to three and minus three. The trials and resting period can be defined before starting the application. 15 and 30 trial runs were used, which means for each side 15 or 30 runs will be made. A resting period of two seconds was chosen. During training, a marker update is sent every seven seconds. In play mode, the game awaits an incoming LSL stream with values between zero and one to balance the ball. Where zero means to tilt the bar down on the left and one means to tilt it completely down on the right, see Figure 16.

The participant is told to imagine squeezing a ball in the left hand or right hand depending in which direction the ball rolls. To simplify the imagination task the user was provided with two physical balls to keep in their hand.

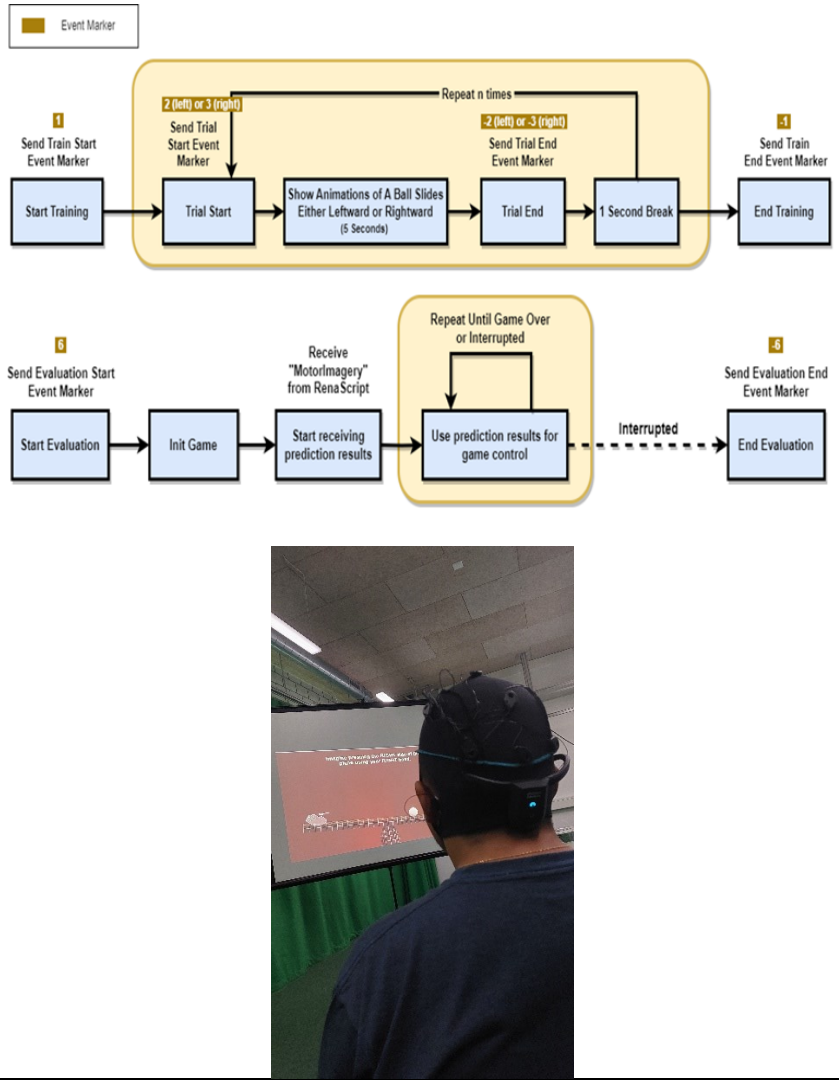


Figure 16: Balance Ball Game

Top: Illustration how the event markers are sent out from the Unity game when training and how an incoming stream controls the game in evaluation mode. Note that by default the break is set to one second [107], [108].
Bottom: Subject doing a training session. The ball randomly goes to the left or right side when in training.

4.5 Processing

The processing script, called processing.py and found in the digital attachment, ingests data streams from both the EEG device and the balance ball game, see Figure 10. On the EEG device, a bandpass filter and notch filter are applied before it is passed on. With the built-in function get_event_locked_data, see [109], the raw EEG data is correlated with the markers. Epoch length can be specified, and samples are created that can be keyed by event value. The data is filtered and segmented to only contain epochs corresponding to left- and right-hand imagery movements. As described in Chapter 4.2, the data is then used to train the CSP filter and LDA classifier according to the MNE documentation [97]. Per default, the CSP computes the power of the epoch and takes the logarithm [110]. Once the training has finished, the trained model is applied to the incoming EEG data in real-time. A classification inference is outputted over LSL. The parameter control_player, see Figure 10, can be set to true and is used to force the loop into decoding mode to control other applications and not rely on the Unity Game. Epoch lengths were varied between three and four seconds. Meaning of the five seconds the ball was moving in training, either the last three or last four seconds of EEG data was taken. To smoothen the inference value a moving average is applied. The processing script is executed at 50Hz and average loop execution takes 3ms. A training section with 100 epochs (50 left and 50 right) was processed in 3.8 seconds, which is done once. The process is illustrated in Figure 17.

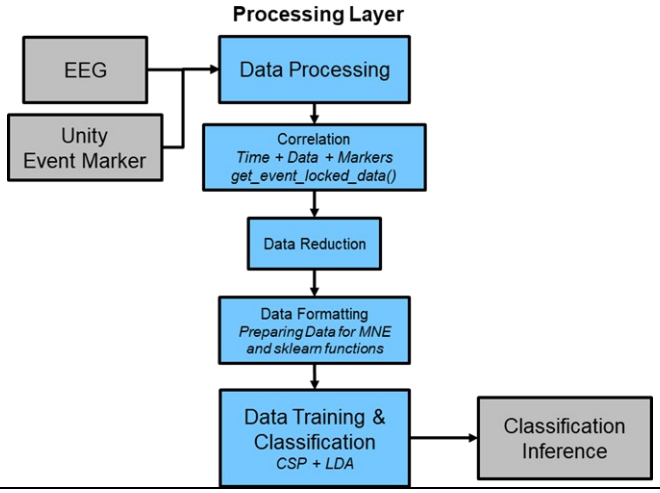


Figure 17: Processing Script

The script takes the EEG data stream, and Event Marker correlates the data before processing and training the CSP and LDA classifier. Once training is finished, an output stream is created, which outputs the classification of detecting a left- and right-hand movement. Zero for left- and 1 for right-hand imagery movement.

4.6 Controlling Applications

Three applications are controlled, see Figure 18. The MotorImageryInference stream forwards the classification values from the processing script. The inference lies between zero and one. Where zero means the epoch was classified as a 100% left-hand imagery movement and one 100% right-hand movement.

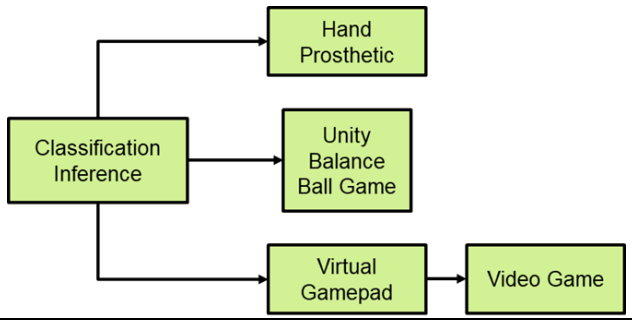


Figure 18: Controlling Applications

The classification inference is used to control three different applications. It can be fed back to the Unity Balance game, a hand prosthesis can be controlled or a virtual gamepad. For the hand prosthesis and gamepad is a separate script that reads the inference value and handles the further processing.

For the Unity Balance game, the inference stream is directly forwarded back to the game to control the bar and balance the ball on top of it. The inference dictates how much the bar tilts to the left or right. The inlet and outlet streams can be defined in Unity as described in Figure 19.

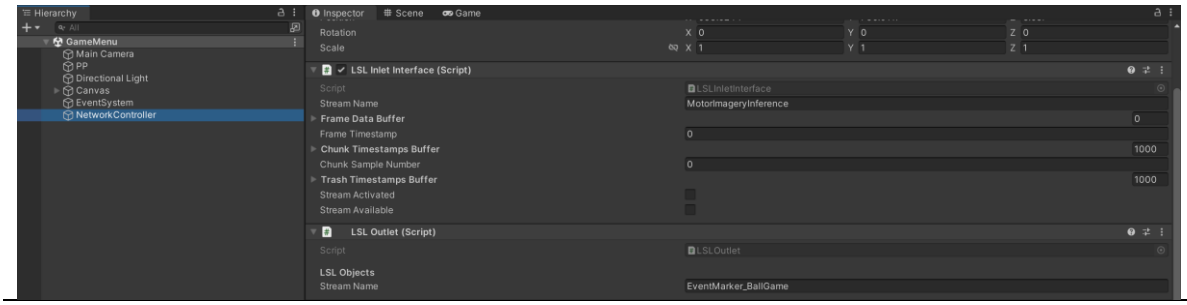


Figure 19: Inlet and Outlet Configuration in Unity

Under Assets -> Scene, select GameMenu. In the Hierarchy tab select NetworkController and in the Inspector are the configurations for the inlet and outlet stream. Per default, the event makers are forwarded over an LSL stream called EventMarker_BallGame and the inlet stream is called MotorImageryInference.

The hand prosthesis was configured in a previous bachelor thesis, see here [111]. The prosthesis is controlled over an Arduino that sends the character zero and one over a Bluetooth connection. It has two modes. In first mode, the hand closes when received a zero and one sequence. It opens again after receiving another sequence of one and zero. In mode 2 the hand closes as long as a one sequence is received and starts to close as long as a zero is in coming.

The control_robotHand.py script, see digital attachment in Chapter 9 and Figure 20, has a configurable parameter called serial_port where the COM port of the Arduino can be set. The Arduino is connected over USB with the workstation. The script reads from the inference stream. If the inference is less than 0.2 it is determined as left imagery movement and a one is sent. If the threshold is higher than 0.8 it is detected as right imagery movement and a zero is sent to the Arduino.

For controlling a gamepad the Python module vgamepad [112] is used. It allows creating a virtual controller and map joysticks and buttons to physiological signals, as done in a previous work described here [113]. Two functions are implemented. One uses the inference signal as a joystick and provides the option to move left and right. If the inference is zero,

the joystick is moved completely to the left and if it is one it is moved fully to the right. The other function maps the signal to the A button of a Xbox360 controller, see Figure 21. Similar to the prosthetic hand, the same threshold is set. If the inference is less than 0.2 the button is pressed, if it is higher than 0.2 the button is released. Both scrips are executed at 20Hz and their average loop duration was below 1ms.

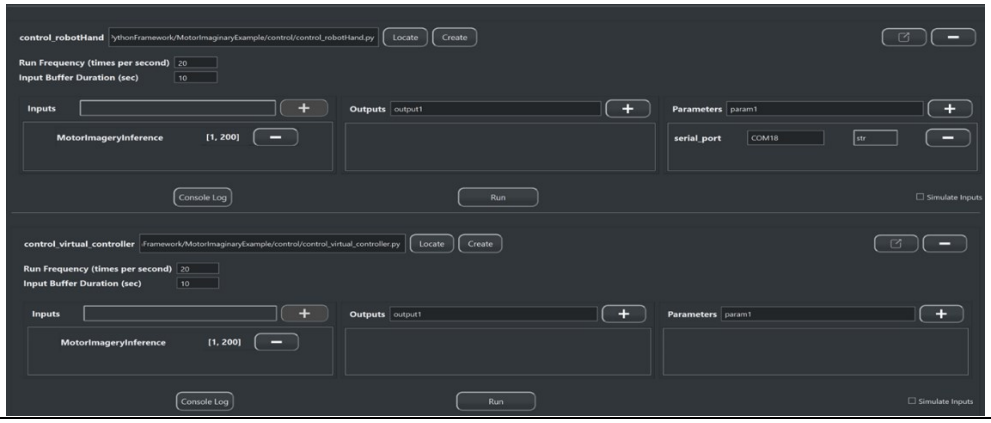
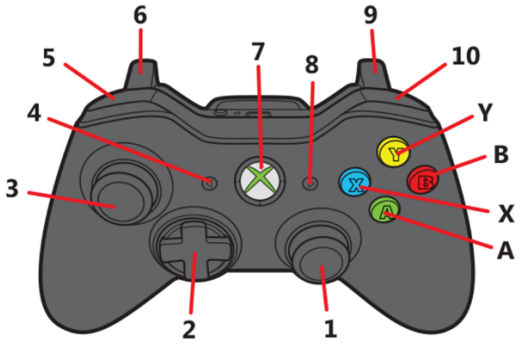


Figure 20: Controlling Scripts

For the hand prosthesis, the serial_port can be configured over the scripting interface. Both scripts take as input the inference stream, called MotorImageryInference, generated from the processing script.



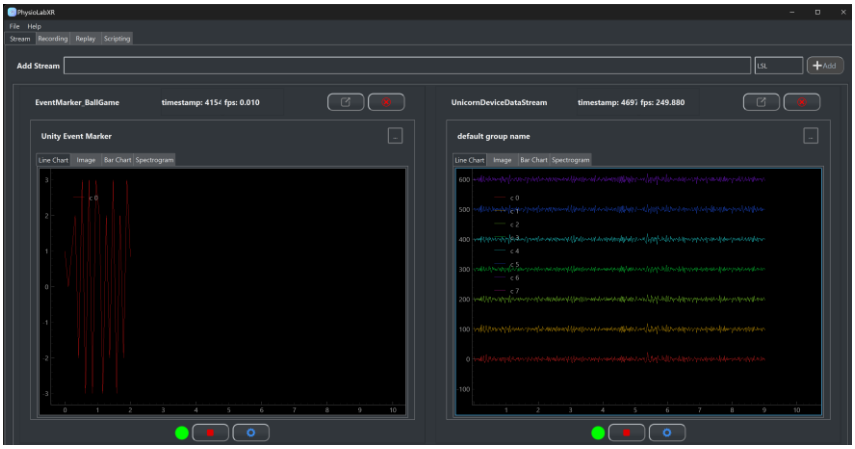
1 Right stick	**8** Start button
2 Directional pad (D-pad)	**9** Right trigger
3 Left stick	**10** Right bumper
4 Back button	**A** A button (green button)
5 Left bumper	**B** B button (red button)
6 Left trigger	**X** X button (blue button)
7 Guide button	**Y** Y button (yellow button)

Figure 21: XBox360 Controller Layout

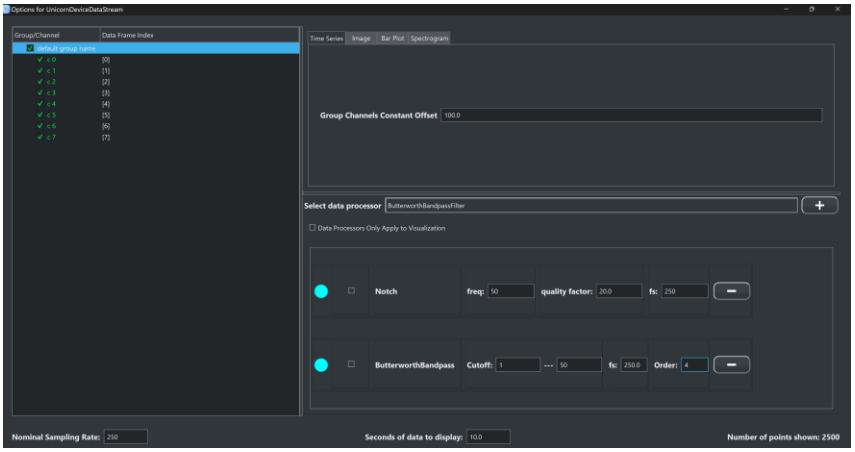
Showing the buttons that are configurable on the virtual controller and how they are mapped. [114]

As mentioned in chapter 4.1, the monitoring, recording and replay functionalities are handled over PhysiolabXR directly. For the recording, the disk location and name can be specified and all the streams get recorded. For the visualization, the LSL streams can be added and get plotted. Plot frequency, type of plot and other plot configurations can be done via the graphical user interface (GUI), see Figure 22, or handled in code.

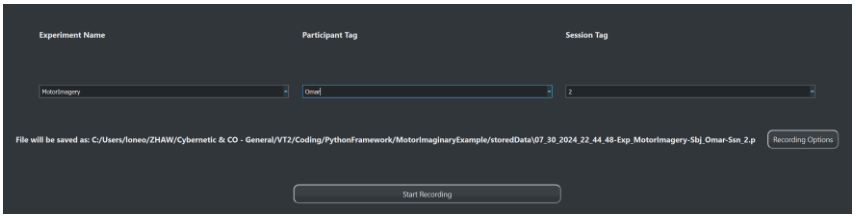
A)



B)



C)



D)

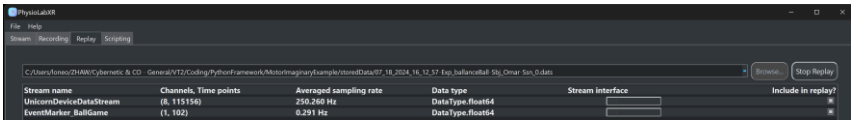


Figure 22: Visualization and Recording Tools

A) Showing the event marker stream and EEG data stream in real-time.
B) Showing the option to customize and preprocessing functionalities of the EEG stream before forwarding it further. A notch filter and bandpass filter are applied.
C) The recording tab allows for the recording of all streams.
D) The replay tab loads an older recording and allows visualizing and processing again.

5 Results

5.1 Complete Framework

Figure 23 illustrates the realization of the framework. A subject is able to complete the training and then control the three applications with motor imagery movement. Illustrated are the examples for the balance game, hand prosthesis and a video game. The modules run in parallel and the different LSL streams can be passed through the pipeline. Visualization and recording are possible while acquiring real-time data. The framework works as intended.

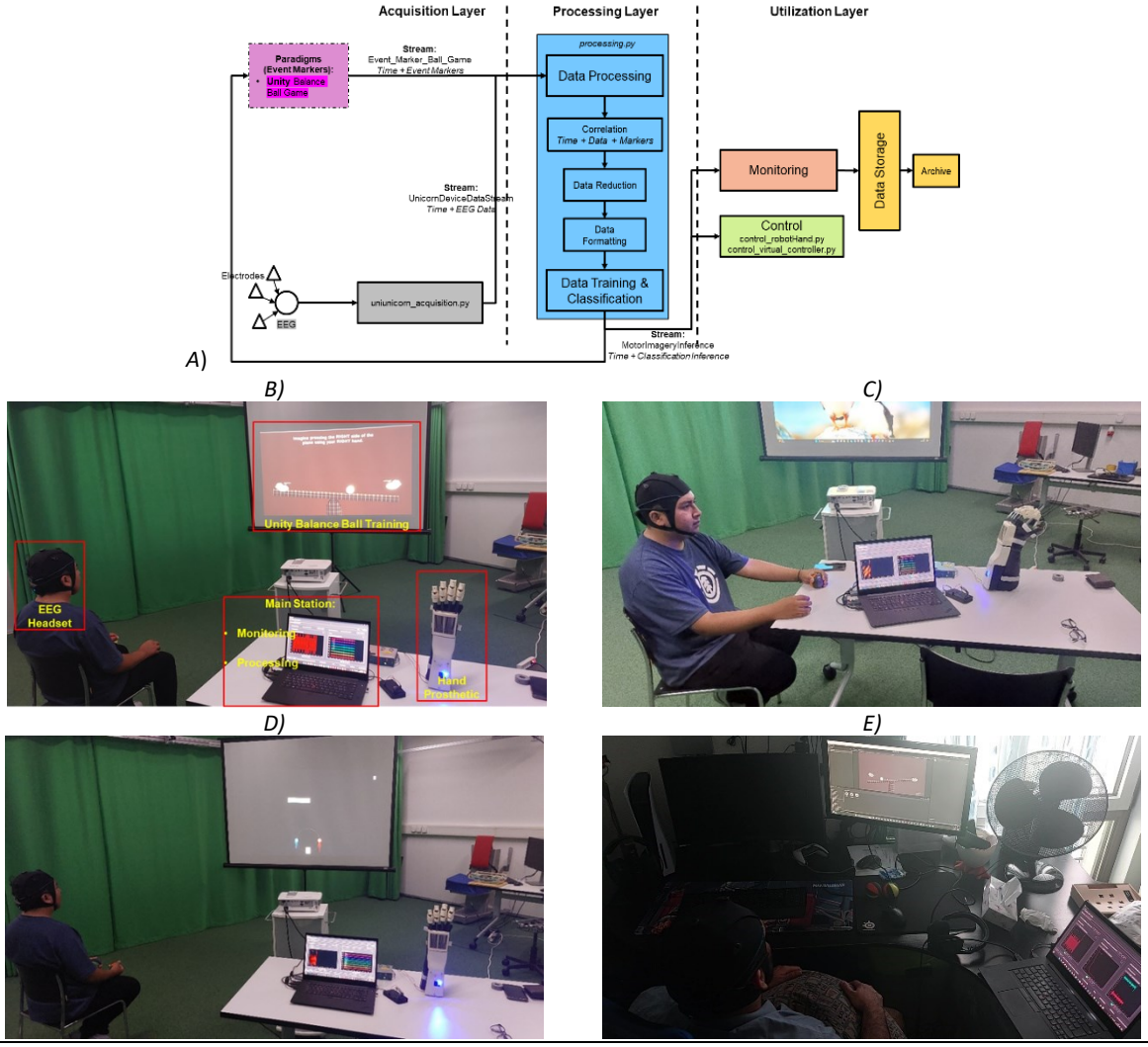


Figure 23: Realization of the Framework

A) Showing the data flow of the framework
B) Showing a subject absolving the training, on the workstation the event marker and the EEG data is monitored
C) Showing a subject controlling the prosthetic hand. The hand just closed.
D) Showing a subject playing the video game Duet [115]. The goal is to maneuver through obstacles while moving the circles. The circles rotate at the same time either clockwise or anti-clockwise.
E) Showing a subject playing the balance ball game.

The longest processing time in the pipeline occurs after training finishes and CSP and LDA are fitted with the data. During training, the data is stored in memory. Once the training is finished, signaled by the event marker with a minus one, see Figure 16. The data is loaded from memory, epoched and fitted. This is only done once during runtime. The longest processing time observed was 3.8 seconds with 50 trials for left- and right-hand imagery movement. The duration of the fitting linearly correlates to the total amount of trials. The more trials, the longer the fitting takes. One trial takes seven seconds, five for the ball rolling on one side and two seconds of rest. A training session of 100 trials in total, 50 for each side, will take around 11 minutes and 40 seconds. After the fitting, the decoding of the signals took on average 3ms. For the acquisition, the longest delay which was observed was 2ms. The processing time for the different controls was less than 1ms. On average, the complete pipeline takes a bit more than four milliseconds.

5.2 Motor Imagery Results

Table 6 shows the results from different motor imagery experiments, where the individual parameters like channels, epoch length and CSP order were changed. The classification accuracy stands for what the classifier determines after the training is completed. The state stands for if the subject was able to play the game and had a feel of control. The classification varies from 50% to 75%. Only on results above 60% were the games playable. Subject 2 never achieved a high accuracy and participated in two experiments. The highest results were achieved when the channels were reduced to C3 and C4. Each parameter has an influence on the classification accuracy.

Table 6: Motor Imagery Results

Subject	EEG Device	Channels	Epoch Length [start, end]	Bandpass Filter (Hz)	CSP Order	Training Runs	Classification Accuracy *Best of*	State
1	Unicorn	C3, C4	[2, 5]	1.0 - 50.0	1	30	75%	Playable
1	Unicorn	C3, C4	[2, 5]	1.0 - 50.0	1	15	70%	Playable
1	Unicorn	Fz, C3, Cz, C4, Pz, PO7, Oz, PO8	[2, 5]	1.0 - 50.0	4	30	65%	Playable
1	Unicorn	C3, C4	[1, 5]	1.0 - 50.0	1	15	65%	Playable
1	Unicorn	Fz, C3, Cz, C4, Pz, PO7, Oz, PO8	[2, 5]	1.0 - 50.0	4	15	61%	Playable
1	Unicorn	Fz, C3, Cz, C4, Pz, PO7, Oz, PO8"	[1, 5]	7.0 - 30.0	4	30	55%	Not Playable
1	Unicorn	Fz, C3, Cz, C4, Pz, PO7, Oz, PO8	[1, 5]	7.0 -30.0	2	30	52%	Not Playable
1	Unicorn	Fz, C3, Cz, C4, Pz, PO7, Oz, PO8	[1, 5]	1.0 - 50.0	2	30	51%	Not Playable
1	Unicorn	Fz, C3, Cz, C4, Pz, PO7, Oz, PO8	[1, 5]	7.0 - 30.0	4	10	50%	Not Playable

2	Unicorn	Fz, C3, Cz, C4, Pz, PO7, Oz, PO8	[1, 5]	7.0-30.0	4	30	50%	Not Playable
2	Unicorn	Fz, C3, Cz, C4, Pz, PO7, Oz, PO8	[1, 5]	7.0 - 30.0	4	15	50%	Not Playable

6 Discussion

6.1 Framework, Memory and Timing

According to the documentation, the LSL stream follows a publisher-subscriber model. Devices that generate data publish to the stream, and subscribers can fetch data from the stream. All data is held in memory for fast retrieval, unless there is insufficient space, in which case the oldest entries are removed from the stream [116].

Given that the used EEG has 18 channels of four bytes each and a sampling frequency of 250Hz, a 1-hour trial run can be expected to generate 64.8 megabytes of data. Most contemporary PCs and laptops have RAM capacities starting at 8 GB. Assuming 70 % of that memory will be reserved / allocated for other applications, that still leaves 2.4 GB available for use. The EEG data stream would occupy 2.7% of that memory after an hour of acquisition, see Equation 1.

Equation 1: Memory Computation

$$\text{EEG data size after 1h: } 18 * 4 \text{ bytes} * 250 \text{ Hz} * 60 \text{ s} * 60 = 64.8 * 10^6 \text{ bytes}$$
$$\text{Available memory space: } 8 \text{ GB} * 0.3 = 2.4 \text{ GB}$$
$$\text{Occupying memory: } \frac{0.0648}{2.4} = 2.7 \%$$

This means EEG data can remain in memory during the training period without loss of data. The buffer size in PhysiolabXR is tied to the internal buffer used by the framework itself and not to that of LSL. This distinction is important, as these are essentially two separate data structures. The PhysiolabXR buffers flush their data in 10 second intervals and (subsequently) query for new samples. The data flushed from PhysiolabXR's buffers is likely still present in the buffers of LSL, however, and can thus still be served by LSL upon request. The only requirement is that there is enough free memory on the system. which is the case here.

The used EEG device samples with 250Hz, the acquisition script samples at a 100Hz frequency and has an average loop duration of 1ms. The longest observed delay was 2ms. At 100 Hz, this delay does not affect the acquisition, as it occurs only in 10ms intervals. The processing loop's completion time averaged at 3ms, with the loop repeating every 50Hz (20ms). The controlling scripts take less than 1ms to run, and their execution reoccurs every 20Hz (50ms). Both the interval for loop repetition as well as the execution time of a single loop are large enough to rule out interference. One potential improvement would be to determine the optimal run frequency for a specific loop time.

Training the CSP and LDA filter takes around 3.8 seconds, but during the fitting new data is stored in the buffer. During the decoding stage the processing script will always fetch the latest data from the buffer, meaning that once training completes successfully, the epoch is built with the latest data, thereby creating a moving sampling window that gets populated with five new entries every 20ms.

Supposing that training of the CSP and LDA were to occur in each loop iteration, the script's execution frequency would need to be adjusted in accordance with the new loop time.

For best performance, the loop frequency and duration would need to be reconfigured in the event of changes to the code or the employed algorithms.

The delay between detection of a motor imagery movement and control output was not benchmarked at this time. The test subject reported a slight lag, though, which could potentially be improved by decreasing the epoch length.

6.2 Motor Imagery Results

The classification accuracies are in line with other research data, see [117], [118], [119]. Figure 24 shows how the electrodes are distributed and which brain region is deemed responsible for hand motor skills. C3 and C4 are closer to the brain region corresponding to the face rather than the hands. Indeed, the placement proved suboptimal for capturing discriminative features associated with motor imagery tasks. Additionally, the short number of trials had a debilitating effect on classification. The use of dry electrodes, while convenient, might have compromised data quality due to increased noise levels. Furthermore, maintaining heightened levels of focus during the experiment proved difficult for the test subjects, potentially also impacting classification performance in a negative way. Subject two was unable to muster the required amount of focus during the training and developed a headache after the two initial runs, which led to the experiment being aborted. The cap has designated places for the electrodes and does not allow for relocation. The EEG data stream was reduced to two channels, C3 and C4. This demonstrated the possibility of classifying brain activity as originating from the left or right brain, while at the same time not providing a definitive way of distinguishing left- and right-hand motor imagery movement from other neural activity. Further, the temporalis muscle started to cause artifacts when the participants started to intensify their imaginary movement. The muscle presses on the cap and moves the electrodes, which results in noise.

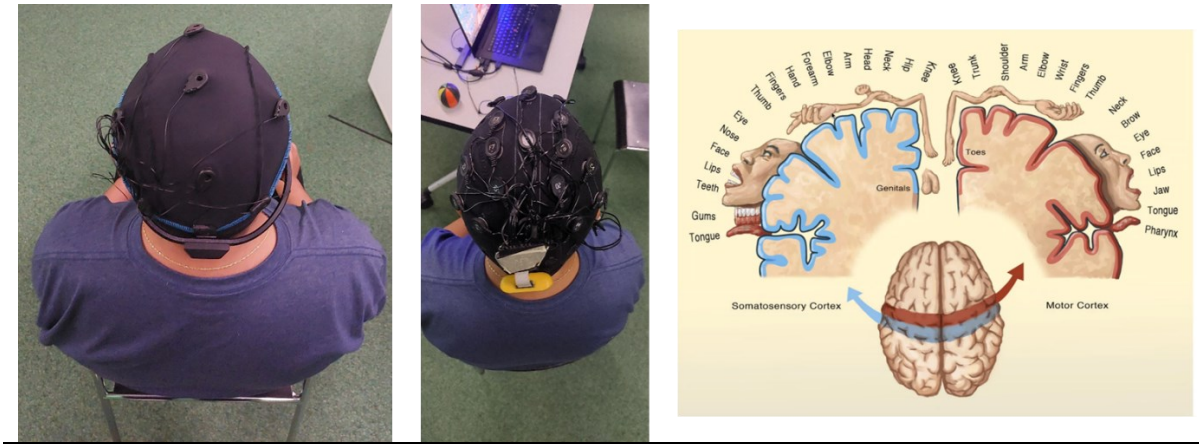


Figure 24: Electrode Placement Comparison

Showing the comparison, how the electrodes are distributed in perspective to the homunculus. On the left, the electrodes are not ideally placed for hand motor-imagery. The ideal place would be between C3 – Cz and C4 - Cz. The gtec cap has no additional holes such that the electrodes could be replaced. In the middle, an Enobio 24 channel device is shown covering most of the motor cortex.

Increasing the epoch length had an inverse effect on the accuracy of the classification. Similarly, keeping the bandpass filter between 8 and 30 Hz was worse than having a broader bandpass filter. Decreasing the CSP filter order had a negative impact on accuracy. The effects of increasing the order were not analyzed. Controlling the hand prosthesis and interacting with the video game appeared to elicit some joy in the test subjects. The Duet video game [115], with its incoming obstacles, proved too fast-paced and there was no configuration option to decrease the speed. For future experiments, it is recommended to stick to a game where all basic actions can be conducted with a single button and / or where the speed can be tweaked. For example, the game “one btn bosses” [120], could be such a candidate.

6.3 Modularity

This experiment did not aim for high classification accuracies. Rather, its purpose was to serve as a stage to demonstrate the modular setup of the framework and the speed with which configuration templates could be developed and benchmarked. Changing the CSP order or the epoch length required only minimal adjustments of the variables in the processing script and did not affect any other modules in the pipeline. Similar trial length and resting period could be set in the Unity game, without modifying any of the other scripts. The virtual game controller could be switched from a joystick-based input control to one based on button presses (and vice versa) by tweaking the code logic. This could be improved by defining a parameter in the scripting tab that could be toggled on and off, just like in the scenario with the processing script and the hand prosthesis example, Figure 10 and Figure 20. Should the need arise to test a new / different EEG device, the Unicorn EEG stream would most certainly have to be replaced with another one. For example, with an Enobio device that only offers LSL, the LSL stream can be directly added to the inputs of the processing script. The script’s internal business logic would of course have to be adjusted, notably the reference to the stream name. This could also be streamlined by providing an external script argument / parameter. Channel names are currently hard-coded in the processing script. This could be improved by the utilization of an LSL EEG device meta stream. By querying the meta stream, the correct channel names could be automatically determined at runtime. Another option would be to set the EEG stream name to something generic, thereby sidestepping the need for re-referencing. Also, in case IMU data has to be forwarded, the changes only happen in the acquisition script.

7 Conclusion & Outlook

In conclusion, this project has successfully leveraged existing Python tools and modules to create real-time BCI applications. PhysioLabXR thereby provides the mold from which a robust backend can be crafted as well as numerous API endpoints for recording, visualization, and scripting tasks. Its API-mediated access to LSL streams simplifies processing and allows for easier interfacing with other applications. The framework's modular setup and its division of the pipeline into the three stages "acquisition", "processing" and "control" elevate end user's flexibility. Even swapping EEG devices will result in minimal disruptions to the overall pipeline. The motor imagery demonstration serves as a proof of concept.

The scripting feature cuts down on programming overhead and renders the design of BMI applications more user-friendly. The framework, by default, does the heavy lifting relating to the lifecycle management of the ring buffer and streams, but users could optionally handle these tasks themselves. Given the framework's compatibility with Windows, macOS and Linux, cross-platform development and interdisciplinary work is featured. There is no restriction pertaining to the usage of other Python modules, and the LSL and ZMQ interface both support the integration of any kind of sensor or biosignal.

Future research shall either focus on the framework itself or explore the potential of cloud-based BCIs. Concerning the framework, the motor imagery BCI could still be optimized by including Independent Component Analysis (ICA) for preprocessing or implementing machine learning algorithms for continuous model improvement. Further, the framework could be expanded with additional BMI paradigms (e.g. P300) and integrated with Python modules like PsychoPy to create more BCIs.

Casting a wider view, the structure of LSL streams may raise privacy and security concerns. Lots of Python modules utilize LSL to interface with each other, but the stream itself is not encrypted. This is negligible in a lab environment, but in a public space (where networks have to be regarded as untrusted), the unencrypted streams may put the BCI applications (and their users) at risk of falling prey to Man-in-the-Middle attacks. Finally, just like in the IoT sector, the industry may one day move towards cloud-based BCI processing, where challenges such as data privacy and latency will present themselves for future research.

8 Bibliography

[1] J. J. Vidal, "Toward Direct Brain-Computer Communication," *Annu. Rev. Biophys.*, vol. 2, no. Volume 2, 1973, pp. 157–180, Jun. 1973, doi: 10.1146/annurev.bb.02.060173.001105.
[2] F. Lotte, C. Nam, and A. Nijholt, "Introduction: Evolution of Brain-Computer Interfaces," 2018, pp. 1–8.
[3] F. Lotte, C. Nam, and A. Nijholt, *BCI Technologies and historical Events*. 2018. [Image]. Available: https://www.researchgate.net/publication/322173712_Introduction_Evolution_of_Brain-Computer_Interfaces
[4] R. Somai, M. Riahi, and F. Moussa, "ALS Recommendation System for BCI User experience evaluation," Sep. 2020.
[5] A. Kawala-Sterniuk *et al.*, "Summary of over Fifty Years with Brain-Computer Interfaces—A Review," *Brain Sci.*, vol. 11, no. 1, Jan. 2021, doi: 10.3390/brainsci11010043.
[6] "What is BCI? | Calgary Pediatric Brain-Computer Interface Program | Cumming School of Medicine | University of Calgary." Accessed: Jul. 25, 2024. [Online]. Available: https://cumming.ucalgary.ca/research/pediatric-bci/bci-program/what-bci
[7] "Introduction To Modern Brain-Computer Interface Design - SCCN." Accessed: May 15, 2024. [Online]. Available: https://sccn.ucsd.edu/wiki/Introduction_To_Modern_Brain-Computer_Interface_Design
[8] "Intro to Brain Computer Interface," NeurotechEDU. Accessed: Jul. 24, 2024. [Online]. Available: http://learn.neurotechedu.com/introtobci/
[9] M. A. Lebedev and M. A. L. Nicolelis, "Brain-Machine Interfaces: From Basic Science to Neuroprostheses and Neurorehabilitation," *Physiol. Rev.*, vol. 97, no. 2, pp. 767–837, Apr. 2017, doi: 10.1152/physrev.00027.2016.
[10] *BCI Setup*. Accessed: Jul. 25, 2024. [Image]. Available: https://www.ncbi.nlm.nih.gov/core/lw/2.0/html/tileshop_pmc/tileshop_pmc_inline.html?title=Click%20on%20image%20to%20zoom&p=PMC3&id=7824107_brainsci-11-00043-g001.jpg
[11] "All-in-One Gelfree Electrode Cap Bundle," OpenBCI Online Store. Accessed: Jul. 25, 2024. [Online]. Available: https://shop.openbci.com/products/all-in-one-gelfree-electrode-cap-bundle
[12] "Unicorn Hybrid Black," g.tec medical engineering GmbH. Accessed: Jul. 25, 2024. [Online]. Available: https://www.gtec.at/product/unicorn-hybrid-black/
[13] K. Värbu, N. Muhammad, and Y. Muhammad, "Past, Present, and Future of EEG-Based BCI Applications," *Sensors*, vol. 22, no. 9, p. 3331, Apr. 2022, doi: 10.3390/s22093331.
[14] D. W. Yip, A. O. Awosika, and F. Lui, "Physiology, Motor Cortical," in *StatPearls*, Treasure Island (FL): StatPearls Publishing, 2024. Accessed: Jul. 25, 2024. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/NBK542188/
[15] C. Voss, "The Homunculus," Voss Feldenkrais. Accessed: Jul. 25, 2024. [Online]. Available: https://www.vossfeldenkrais.com/post/the-homunculus
[16] *Homunculi.* Accessed: Jul. 25, 2024. [Image]. Available: https://static.wixstatic.com/media/9165c4_85b92fddb0a444d1ad8d5dbde7995530~mv2.jpg/v1/fill/w_900,h_682,al_c,q_85,enc_auto/9165c4_85b92fddb0a444d1ad8d5dbde7995530~mv2.jpg
[17] "The 10-20 System for EEG." Accessed: Jul. 26, 2024. [Online]. Available: https://info.tmsi.com/blog/the-10-20-system-for-eeg
[18] A. Chaddad, Y. Wu, R. Kateb, and A. Bouridane, "Electroencephalography Signal Processing: A Comprehensive Review and Analysis of Methods and Techniques," *Sensors*, vol. 23, no. 14, p. 6434, Jul. 2023, doi: 10.3390/s23146434.
[19] "EEG." Accessed: Jul. 26, 2024. [Online]. Available: https://www.medicine.mcgill.ca/physio/vlab/biomed_signals/eeg_n.htm
[20] S. Aggarwal and N. Chugh, "Signal processing techniques for motor imagery brain computer interface: A review," *Array*, vol. 1–2, p. 100003, Jan. 2019, doi: 10.1016/j.array.2019.100003.
[21] P. Wierzgała, D. Zapała, G. M. Wojcik, and J. Masiak, "Most Popular Signal Processing Methods in Motor-Imagery BCI: A Review and Meta-Analysis," *Front. Neuroinformatics*, vol. 12, p. 78, Nov. 2018, doi: 10.3389/fninf.2018.00078.
[22] J. J. Shih, D. J. Krusienski, and J. R. Wolpaw, "Brain-Computer Interfaces in Medicine," *Mayo Clin. Proc.*, vol. 87, no. 3, pp. 268–279, Mar. 2012, doi: 10.1016/j.mayocp.2011.12.008.
[23] A. Dai, "Brain computer interface for the treatment of neurodegenerative diseases," *E3S Web Conf.*, vol. 553, p. 05010, 2024, doi: 10.1051/e3sconf/202455305010.
[24] Z. A. Alkaff *et al.*, *Applications of Brain Computer Interface in Present Healthcare Setting*. IntechOpen, 2024. doi: 10.5772/intechopen.112353.
[25] S. Luo, Q. Rabbani, and N. E. Crone, "Brain-Computer Interface: Applications to Speech Decoding and Synthesis to Augment Communication," *Neurotherapeutics*, vol. 19, no. 1, pp. 263–273, Jan. 2022, doi: 10.1007/s13311-022-01190-2.
[26] G. Papanastasiou, A. Drigas, C. Skianis, and M. Lytras, "Brain computer interface based applications for training and rehabilitation of students with neurodevelopmental disorders. A literature review," *Heliyon*, vol. 6, no. 9, p. e04250, Sep. 2020, doi: 10.1016/j.heliyon.2020.e04250.
[27] V. Maksimenko *et al.*, "Brain-computer interface for the epileptic seizures prediction and prevention," in *2020 8th International Winter Conference on Brain-Computer Interface (BCI)*, Feb. 2020, pp. 1–5. doi: 10.1109/BCI48061.2020.9061655.

[28] X. Qian *et al.*, "Brain-computer-interface-based intervention re-normalizes brain functional network topology in children with attention deficit/hyperactivity disorder," *Transl. Psychiatry*, vol. 8, no. 1, pp. 1–11, Aug. 2018, doi: 10.1038/s41398-018-0213-8.
[29] E. Barraud, "Thought-controlled walking again after spinal cord injury," May 2023, Accessed: Jul. 26, 2024. [Online]. Available: https://actu.epfl.ch/news/thought-controlled-walking-again-after-spinal-co-3/
[30] R. Bajpai, "Top 10 Brain-Computer Interface Companies in the World," ELE Times. Accessed: Jul. 26, 2024. [Online]. Available: https://www.eletimes.com/top-10-brain-computer-interface-companies-in-the-world
[31] "Top brain-computer interface companies | VentureRadar." Accessed: Jul. 26, 2024. [Online]. Available: https://www.ventureradar.com/keyword/brain-computer%20interface
[32] S. Whooley, "7 brain-computer interface companies you need to know," MassDevice. Accessed: Jul. 26, 2024. [Online]. Available: https://www.massdevice.com/brain-computer-interface-bci-companies/
[33] "BrainCo – Train Your Brain for Success." Accessed: Jul. 26, 2024. [Online]. Available: https://brainco.tech/
[34] "About EMOTIV," EMOTIV. Accessed: Jul. 26, 2024. [Online]. Available: https://www.emotiv.com/pages/about
[35] "Kernel | About." Accessed: Jul. 26, 2024. [Online]. Available: https://www.kernel.com/about
[36] "Shop | Muse™ EEG-Powered Meditation & Sleep Headband." Accessed: Jul. 26, 2024. [Online]. Available: https://choosemuse.com/pages/shop, https://choosemuse.com/pages/shop
[37] "Products | Neurable." Accessed: Jul. 26, 2024. [Online]. Available: https://www.neurable.io/products
[38] "Our Story," Blackrock Neurotech. Accessed: Jul. 26, 2024. [Online]. Available: https://blackrockneurotech.com/our-story/
[39] "About Braingate," BrainGate. Accessed: Jul. 26, 2024. [Online]. Available: https://www.braingate.org/about-braingate/
[40] Neuralink, "Neuralink — Pioneering Brain Computer Interfaces," Neuralink. Accessed: Jul. 26, 2024. [Online]. Available: https://neuralink.com/
[41] "Enabling Connection | About Paradromics." Accessed: Jul. 26, 2024. [Online]. Available: https://www.paradromics.com/about
[42] "Precision - About." Accessed: Jul. 26, 2024. [Online]. Available: https://precisionneuro.io/about
[43] "The Technology," Synchron. Accessed: Jul. 26, 2024. [Online]. Available: https://synchron.com/technology
[44] R. K. Das, A. Martin, T. Zurales, D. Dowling, and A. Khan, "A Survey on EEG Data Analysis Software," *Sci*, vol. 5, no. 2, Art. no. 2, Jun. 2023, doi: 10.3390/sci5020023.
[45] Venthur, Bastian and Blankertz, Benjamin, "Towards a Free and Open Source BCI System Written in Python," *Verlag der Technischen Universität Graz*. doi: 10.3217/978-3-85125-260-6-179.
[46] J. S. Brumberg, S. D. Lorenz, B. V. Galbraith, and F. H. Guenther, "The Unlock Project: A Python-based framework for practical brain-computer interface communication 'app' development," *Conf. Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. IEEE Eng. Med. Biol. Soc. Conf.*, vol. 2012, pp. 2505–2508, 2012, doi: 10.1109/EMBC.2012.6346473.
[47] "BCI++ - Brain Computer Interface Wiki." Accessed: Jul. 27, 2024. [Online]. Available: https://bciwiki.org/index.php?title=BCI%2B%2B
[48] "BioSig Technologies, Inc. (BSGM)," BioSig Technologies, Inc. Accessed: Jul. 27, 2024. [Online]. Available: https://www.biosig.com
[49] "BCI2000 Wiki." Accessed: Jul. 27, 2024. [Online]. Available: https://www.bci2000.org/mediawiki/index.php/Main_Page
[50] "Home," OpenViBE. Accessed: Jul. 27, 2024. [Online]. Available: https://openvibe.inria.fr/
[51] I. P. Susila, S. Kanoh, K. Miyamoto, and T. Yoshinobu, "xBCI: A Generic Platform for Development of an Online BCI System," *IEEJ Trans. Electr. Electron. Eng.*, vol. 5, no. 4, pp. 467–473, 2010, doi: 10.1002/tee.20560.
[52] *sccn/BCILAB*. (Jul. 13, 2024). HTML. Swartz Center for Computational Neuroscience. Accessed: Jul. 27, 2024. [Online]. Available: https://github.com/sccn/BCILAB
[53] "Introduction - Brainstorm." Accessed: Jul. 27, 2024. [Online]. Available: https://neuroimage.usc.edu/brainstorm/Introduction
[54] "EEGLAB." Accessed: Jul. 27, 2024. [Online]. Available: https://sccn.ucsd.edu/eeglab/index.php
[55] "Getting started with real-time analysis for BCI/neurofeedback," FieldTrip toolbox. Accessed: Jul. 27, 2024. [Online]. Available: https://www.fieldtriptoolbox.org/getting_started/realtime/
[56] "LabStreamingLayer's Documentation — Labstreaminglayer 1.13 documentation." Accessed: Jun. 25, 2024. [Online]. Available: https://labstreaminglayer.readthedocs.io/index.html
[57] *bbci/mushu*. (Oct. 15, 2023). Python. Berlin Brain-Computer Interface. Accessed: Jun. 18, 2024. [Online]. Available: https://github.com/bbci/mushu
[58] *bbci/wyrm*. (Jun. 07, 2024). Python. Berlin Brain-Computer Interface. Accessed: Jun. 18, 2024. [Online]. Available: https://github.com/bbci/wyrm
[59] P. Bota, R. Silva, C. Carreiras, A. Fred, and H. Plácido da Silva, *BioSPPy: A Python toolbox for physiological signal processing*. (May 2024). Python. doi: 10.1016/j.softx.2024.101712.
[60] "Braindecode — Braindecode 0.8.1 documentation." Accessed: Jun. 19, 2024. [Online]. Available: https://braindecode.org/dev/index.html
[61] B. Reuderink, *breuderink/eegtools*. (Dec. 20, 2023). Python. Accessed: Mar. 11, 2024. [Online]. Available: https://github.com/breuderink/eegtools
[62] *eeglib*. (Oct. 23, 2023). SoftwareX. Accessed: May 15, 2024. [Online]. Available: https://github.com/ElsevierSoftwareX/SOFTX-D-21-00087

[63] *ElsevierSoftwareX/SOFTX-D-21-00087*. (Oct. 23, 2023). SoftwareX. Accessed: Mar. 11, 2024. [Online]. Available: https://github.com/ElsevierSoftwareX/SOFTX-D-21-00087
[64] "GitHub - octopicorn/bcikit: Toolkit and workbench for Brain Computer Interface (BCI) software development, for Python. Modular design built to play well with Machine Learning algorithms that follow python's `scikit-learn` interface." Accessed: Jun. 19, 2024. [Online]. Available: https://github.com/octopicorn/bcikit?tab=readme-ov-file#readme
[65] *gumpy-bci/gumpy*. (Feb. 29, 2024). Python. gumpy-bci. Accessed: Jun. 18, 2024. [Online]. Available: https://github.com/gumpy-bci/gumpy
[66] A. Gramfort *et al.*, *MEG and EEG Data Analysis with MNE-Python*. (2013). Python. doi: 10.3389/fnins.2013.00267.
[67] "MNE — MNE 1.6.1 documentation." Accessed: Mar. 11, 2024. [Online]. Available: https://mne.tools/stable/index.html
[68] "MNE-LSL." Accessed: Jun. 19, 2024. [Online]. Available: https://mne.tools/mne-lsl/stable/index.html
[69] C. Kothe, *pylsl: Python interface to the Lab Streaming Layer*. Python. Accessed: May 15, 2024. [MacOS, Microsoft :: Windows, POSIX :: Linux]. Available: https://github.com/labstreaminglayer/pylsl
[70] *pyRiemann/pyRiemann*. (Jun. 18, 2024). Python. pyRiemann. Accessed: Jun. 19, 2024. [Online]. Available: https://github.com/pyRiemann/pyRiemann
[71] J. Muradeli, *ssqueezepy*. (May 15, 2024). Python. Accessed: May 15, 2024. [Online]. Available: https://github.com/OverLordGoldDragon/ssqueezepy
[72] "Welcome to BrainFlow's documentation! — BrainFlow documentation." Accessed: Jun. 18, 2024. [Online]. Available: https://brainflow.readthedocs.io/en/stable/index.html
[73] *bbci/pyff*. (Oct. 15, 2023). Python. Berlin Brain-Computer Interface. Accessed: Jul. 27, 2024. [Online]. Available: https://github.com/bbci/pyff
[74] *CAMBI-tech/BciPy*. (Jun. 15, 2024). Python. CAMBI-tech. Accessed: Jul. 27, 2024. [Online]. Available: https://github.com/CAMBI-tech/BciPy
[75] "EEGrunt/EEGrunt.py at master · curiositry/EEGrunt · GitHub." Accessed: Jun. 19, 2024. [Online]. Available: https://github.com/curiositry/EEGrunt/blob/master/EEGrunt.py
[76] "Home — PsychoPy®." Accessed: Jul. 27, 2024. [Online]. Available: https://www.psychopy.org/
[77] E. Santamaría-Vázquez *et al.*, "MEDUSA©: A novel Python-based software ecosystem to accelerate brain-computer interface and cognitive neuroscience research," *Comput. Methods Programs Biomed.*, vol. 230, p. 107357, Mar. 2023, doi: 10.1016/j.cmpb.2023.107357.
[78] "NeuroPype - Home." Accessed: Jul. 27, 2024. [Online]. Available: https://www.neuropype.io/
[79] octopicorn, *octopicorn/bcikit*. (May 18, 2024). JavaScript. Accessed: Jul. 27, 2024. [Online]. Available: https://github.com/octopicorn/bcikit
[80] "PhysioLabXR." Accessed: Jul. 27, 2024. [Online]. Available: https://physiolabxr.org/
[81] L. Booth, A. Asghar, and A. Bateson, *PyBCI: A Python Package for Brain-Computer Interface (BCI) Design*. (Dec. 2023). Python. doi: 10.21105/joss.05706.
[82] J. Sieber, *strfry/OpenNFB*. (Jul. 08, 2024). Python. Accessed: Jul. 27, 2024. [Online]. Available: https://github.com/strfry/OpenNFB
[83] "The EEGsynth," The EEGsynth. Accessed: Jul. 27, 2024. [Online]. Available: https://www.eegsynth.org/
[84] falloutxAY, "Azure IoT reference architecture - Azure Architecture Center." Accessed: Jul. 27, 2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/reference-architectures/iot
[85] "IoT Architecture in Detail: Comprehensive Guide," Relevant Software. Accessed: Jul. 27, 2024. [Online]. Available: https://relevant.software/blog/iot-architecture/
[86] A. Mathias and K. Khan, "Internet of Things (IoT) in the Cloud: Connecting and Managing Smart Devices," May 2023.
[87] L. Hou *et al.*, "Internet of Things Cloud: Architecture and Implementation," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 32–39, Dec. 2016, doi: 10.1109/MCOM.2016.1600398CM.
[88] *IoT Pipeline*. Accessed: Jul. 27, 2024. [Image]. Available: https://learn.microsoft.com/en-us/azure/architecture/reference-architectures/iot/images/iot-reference-architecture.svg
[89] N. M. C. Da Costa, E. G. Bicho, and N. S. Dias, "NeuroPrime: a Pythonic framework for the priming of brain states in self-regulation protocols," in *2021 IEEE 9th International Conference on Serious Games and Applications for Health(SeGAH)*, Aug. 2021, pp. 1–8. doi: 10.1109/SEGAH52098.2021.9551893.
[90] "Simulink - Simulation and Model-Based Design." Accessed: Jul. 29, 2024. [Online]. Available: https://ch.mathworks.com/products/simulink.html
[91] "Merge pull request #284 from CAMBI-tech/patch_rc3 · CAMBI-tech/BciPy@02591eb · GitHub." Accessed: Jul. 29, 2024. [Online]. Available: https://github.com/CAMBI-tech/BciPy/commit/02591ebd08399b3d00991abff6f78c3d3768434c
[92] "fix rpc language enuum in RPCWidget · PhysioLabXR/PhysioLabXR-Community@04ff537," GitHub. Accessed: Jul. 29, 2024. [Online]. Available: https://github.com/PhysioLabXR/PhysioLabXR-Community/commit/04ff537cea3e9af33146c0ac4cbb50dfeee35486
[93] "ZeroMQ." Accessed: Jul. 30, 2024. [Online]. Available: https://zeromq.org/
[94] Z. 'Leo' Li *et al.*, "PhysioLabXR: A Python Platform for Real-Time,Multi-modal, Brain–Computer Interfaces and Extended RealityExperiments," *J. Open Source Softw.*, vol. 9, no. 93, p. 5854, Jan. 2024, doi: 10.21105/joss.05854.
[95] "Scripting - PhysioLabXRDocs 1.0.0 documentation." Accessed: Jul. 30, 2024. [Online]. Available: https://physiolabxrdocs.readthedocs.io/en/latest/Scripting.html

[96] Z. 'Leo' Li *et al.*, *Figure 4, Sequence Diagram*. 2024. Accessed: Jul. 23, 2024. [Image]. Available: https://joss.theoj.org/papers/10.21105/joss.05854
[97] "Motor imagery decoding from EEG data using the Common Spatial Pattern (CSP) — MNE 1.7.1 documentation." Accessed: Jul. 30, 2024. [Online]. Available: https://mne.tools/stable/auto_examples/decoding/decoding_csp_eeg.html#references
[98] "Building a multi-modal ERP Classifier based on EEG and Pupil Size - PhysioLabXRDocs 1.0.0 documentation." Accessed: Jul. 30, 2024. [Online]. Available: https://physiolabxrdocs.readthedocs.io/en/latest/tutorials/BuildMultiModalERPClassifier.html
[99] "PhysioLabXR + Unity P300 Speller Tutorial - PhysioLabXRDocs 1.0.0 documentation." Accessed: Jul. 30, 2024. [Online]. Available: https://physiolabxrdocs.readthedocs.io/en/latest/PhysioLabXRP300SpellerDemo.html
[100] "Motor Imagery: Balance Ball Game in Unity - PhysioLabXRDocs 1.0.0 documentation." Accessed: Jul. 14, 2024. [Online]. Available: https://physiolabxrdocs.readthedocs.io/en/latest/BalanceBallMotorImageryTutorial.html
[101] "Motor Imagery: Balance Ball Game in Unity - PhysioLabXRDocs 1.0.0 documentation." Accessed: Jul. 29, 2024. [Online]. Available: https://physiolabxrdocs.readthedocs.io/en/latest/BalanceBallMotorImageryTutorial.html
[102] Z. "Leo" Li, *ApocalyVec/PhysioLabXR_MotorImager_BalancingBall*. (Dec. 05, 2023). C#. Accessed: Jul. 30, 2024. [Online]. Available: https://github.com/ApocalyVec/PhysioLabXR_MotorImager_BalancingBall
[103] "The Python Profilers," Python documentation. Accessed: Jul. 24, 2024. [Online]. Available: https://docs.python.org/3/library/profile.html
[104] Mikejo5000, "First look at profiling tools - Visual Studio (Windows)." Accessed: Jun. 25, 2024. [Online]. Available: https://learn.microsoft.com/en-us/visualstudio/profiling/profiling-feature-tour?view=vs-2022
[105] "Supported Boards — BrainFlow documentation." Accessed: Jul. 30, 2024. [Online]. Available: https://brainflow.readthedocs.io/en/stable/SupportedBoards.html#unicorn
[106] "User API — BrainFlow documentation." Accessed: Jul. 30, 2024. [Online]. Available: https://brainflow.readthedocs.io/en/stable/UserAPI.html
[107] "balanceBall_train_diagram.png (6711×1580)." Accessed: Jul. 23, 2024. [Online]. Available: https://physiolabxrdocs.readthedocs.io/en/latest/_images/balanceBall_train_diagram.png
[108] "balanceBall_eval_diagram.png (6326×1427)." Accessed: Jul. 23, 2024. [Online]. Available: https://physiolabxrdocs.readthedocs.io/en/latest/_images/balanceBall_eval_diagram.png
[109] "PhysioLabXR-Community/physiolabxr/scripting/physio/epochs.py at 698e1e6ac87a12dc33cd52cab4698bc0fcba7534 · PhysioLabXR/PhysioLabXR-Community · GitHub." Accessed: Jul. 30, 2024. [Online]. Available: https://github.com/PhysioLabXR/PhysioLabXR-Community/blob/698e1e6ac87a12dc33cd52cab4698bc0fcba7534/physiolabxr/scripting/physio/epochs.py#L11
[110] "mne.decoding.CSP — MNE 1.7.1 documentation." Accessed: Jul. 30, 2024. [Online]. Available: https://mne.tools/stable/generated/mne.decoding.CSP.html
[111] O. Lone and K. Kuster, "Ansteuerung eines Aktuators mittels EEG." unpublished, Jun. 09, 2022.
[112] *vgamepad: Virtual XBox360 and DualShock4 gamepads in python*. Python. Accessed: Jan. 28, 2024. [Microsoft :: Windows]. Available: https://github.com/yannbouteiller/vgamepad
[113] O. Lone, "Providing a simplified interface for game-based rehabilitation." Jan. 31, 2024.
[114] Microsoft, *Xbox360Controller*. Accessed: Jan. 28, 2024. [Image]. Available: https://compass-ssl.xboxlive.com/assets/d1/04/d10437ca-81da-48db-a7a0-4c1173ad42e1.png?n=xbox-360-controller-layout.png
[115] "Duet on Steam." Accessed: Jul. 31, 2024. [Online]. Available: https://store.steampowered.com/app/292600/Duet/
[116] "LSL Coding Examples — Labstreaminglayer 1.13 documentation." Accessed: Jul. 31, 2024. [Online]. Available: https://labstreaminglayer.readthedocs.io/dev/examples.html
[117] J. Feng, Y. Li, C. Jiang, Y. Liu, M. Li, and Q. Hu, "Classification of motor imagery electroencephalogram signals by using adaptive cross-subject transfer learning," *Front. Hum. Neurosci.*, vol. 16, p. 1068165, Dec. 2022, doi: 10.3389/fnhum.2022.1068165.
[118] T. Mwata-Velu *et al.*, "EEG-BCI Features Discrimination between Executed and Imagined Movements Based on FastICA, Hjorth Parameters, and SVM," *Mathematics*, vol. 11, no. 21, Art. no. 21, Jan. 2023, doi: 10.3390/math11214409.
[119] R. Fu, M. Han, Y. Tian, and P. Shi, "Improvement motor imagery EEG classification based on sparse common spatial pattern and regularized discriminant analysis," *J. Neurosci. Methods*, vol. 343, p. 108833, Sep. 2020, doi: 10.1016/j.jneumeth.2020.108833.
[120] "ONE BTN BOSSES by Brin, Fufer, JonasWG, francescatremulo," itch.io. Accessed: Jul. 31, 2024. [Online]. Available: https://being-brin.itch.io/obb-demo

9 Digital Attachments

All code and other sources were handed in digitally. They have the following structure:

DigitalAttachement_VT2_24_char_baud_loneoma1:

- Code
  - GtecUnicornLib
    - Contains the Python APY for the EEG
  - PythonFramework
    - MotorImageryExample
      - Contains the scripts used for the motor imagery BCI
    - Profiling
      - Contains the scripts and benchmarks from the various profilings
    - PhysioLabXR_MotorImager_BalancingBall-master.zip
      - The Unity balance ball game
    - pybluez-master.zip
      - A Python Bluetooth module for talking with the Bluetooth driver.
- Updates
  - Contains the presentations created for the semester.
- Unicorn Suite Hybrid Black 1.18.00.zip
  - Contains the documentation and software for the EEG
- Benchmarks.xlsx
- Project-Timeline.xlsx
- Python_modules_vennDiagramm.py
  - Used to create the Illustration of Venn-Diagrams.
- PythonPackages.xlsx
- submission_SSBE2024_OmarLone.pdf
- VT1_24_char_baud_lone_thesis.pdf
- VT2_24_char_baud_lone_Presentation.pptx
- VT2_24_char_baud_lone_PresentationHandout.pdf
- VT2_24_init_abstract_lone.pdf