

PROJEKTDOKUMENTATION

Teamorientiertes Projekt

SOMMERSEMESTER 2022

PROF. DR.-ING. MANFRED STRAHNEN

PROF. DR.-ING. PHILIPP GRAF





I. Formalia	5
II. Abkürzungsverzeichnis	6
1. Überblick zu Motivation und Projektziel	7
1.1 Motivation	7
1.2 Projektziel	7
2. Anforderungen	8
2.1 Projektplanung	8
2.1.1 Kunde	8
2.1.2 Teammitglieder	8
2.1.3 Beschreibung des Problems / Vision	8
2.1.4 Stakeholder	9
2.1.5 Features	9
2.2 Funktionale User Stories	10
2.2.1 Funktionale User Stories der Testperson	11
2.2.2 Funktionale User Stories von dem oder der Wissenschaftler:in	11
2.3 Nicht-Funktionale User Stories	13
3. Ergebnis aus Nutzersicht	15
3.1 Ergebnis aus Nutzersicht: Testperson	15
3.1.1 Spiel	15
3.2 Ergebnis aus Nutzersicht: Wissenschaftler:in	16
3.2.1 Bedienung der Grafischen Oberfläche	16
3.2.2 Datenerfassung	19
3.2.3 Spiel	21
3.2.4 Datenanalyse	23
3.2.5 Live-Plot	25
3.2.6 Datenextraktion	26
3.2.7 Datensatz Loader	28
4. Technisches Konzept	29
4.1 Gesamtübersicht	29
4.2 Komponenten Diagramm	30
4.3 Übersicht der Schnittstellen zwischen den Komponenten	31



4.4 Trial Aufnahme und Erfassung der Sliding Windows	32
4.5 Cursor Control Algorithmus	34
4.6 Strategy-Pattern zur Verwaltung der Game Control Strategien	37
4.7 Publish-Subscribe-Pattern zur Realisierung der Algorithmus-Strategie	38
4.8 Datenextraktion	39
4.9 Datenvisualisierung	39
4.10 Model-View-Controller Pattern	41
4.10.1 Allgemeiner MVC-Prozess	41
4.10.2 Kommunikation zwischen zwei MVC-Komponenten	42
4.10.3 Gesamtüberblick und Ownership	43
4.10.4. Modul-/Klassendiagramm mit Minimalbeispielen	44
4.10.5 Minimale Code-Ausschnitte	45
5. Projektdokumentation	48
5.1 Beschreibung des Vorgehensmodells und der Teamorganisation	48
5.1.1 Gewähltes Vorgehensmodell	48
5.1.2 Rollen im Team	48
5.1.3 Allgemeine zugrundeliegenden Entscheidungen	48
5.1.4 Teamorganisation	49
5.1.6 Verwendete Software Tools	49
5.1.7 Verwendete Hardware	49
5.2 Dokumentation Projektmanagement	50
5.2.1 Burndown-Charts	52
5.2.2 Velocity-Charts	55
5.2.2 Backlogs	56
6. Beiträge der einzelnen Teammitglieder	61
7. Diskussion des Ergebnisses	64
7.1 Welche notwendigen Ziele wurden erreicht?	64
7.2 Welche optionalen Ziele wurden erreicht?	64
7.4 Retrospektive	64
7.4.1 Projektübergreifende Retrospektive	64
7.4.2 Retrospektive nach den einzelnen Sprints	67



7.5 Impediment Backlog	69
7.6 Fazit des Projektes	72
8. Aufnahme von EEG-Daten	73
9. Ausblick	76
10. Quellen	77



I. FORMALIA

Hiermit versichern wir, dass wir diese Projektdokumentation selbstständig verfasst und keine anderen als die von uns angegebenen Quellen und Hilfsmittel benutzt wurden.

Vorname, Nachname	Matrikelnummer	Unterschrift
Marcel Roth	3130163	
Firat Demir	3137518	
Christian Dittrich	3134786	
Larissa Willibald	3130800	
Charlotte Wengler	3129558	
Pascal Halmer	3134867	



II. ABKÜRZUNGSVERZEICHNIS

z.B.	zum Beispiel
sog.	sogenannte
bspw.	beispielsweise
bzw.	beziehungsweise
EEG	Elektroenzephalogramm
KI	Künstliche Intelligenz
BCI	Brain Computer Interface
FIFO	First In First Out
FFT	Fast Fourier Transformation



1. ÜBERBLICK ZU MOTIVATION UND PROJEKTZIEL

1.1 MOTIVATION

Brain-Computer-Interfaces wandeln die gemessenen Aktivitäten des Zentralen Nervensystems in eine künstliche Ausgabe um. Sie dienen somit als Verbindung zwischen Umwelt und Gehirn. Verwendung finden diese Schnittstellen beispielsweise bei der Steuerung eines Rollstuhls für ganzkörper gelähmte Personen rein durch Impulse des Gehirns. Die erfassten EEG-Ströme werden von einer KI interpretiert. Zum Training dieser KI benötigt man sogenannte Trials, welche mithilfe eines EEG-Geräts erfasst werden. Ein Trial ist ein aufgenommener Gedankenstrom mit einer bestimmten Dauer, währenddessen die Testperson denselben Gedankenimpuls gibt (z.B. nach rechts denken).

1.2 PROJEKTZIEL

Das Ziel unseres Projektes ist eine spielerische Erfassung der EEG-Trials, ohne die Testperson zu ermüden. Dies wird durch ein einfach gestaltetes Spiel, bei dem die Spielfigur ein Ziel erreichen muss, realisiert.



2. ANFORDERUNGEN

2.1 PROJEKTPLANUNG

2.1.1 KUNDE

Technische Hochschule Ulm (Prof. Dr.-Ing. Manfred Strahnen)

2.1.2 TEAMMITGLIEDER

- Marcel Roth
- Pascal Halmer
- Christian Dittrich
- Firat Demir
- Larissa Willibald
- Charlotte Wengler

2.1.3 BESCHREIBUNG DES PROBLEMS / VISION

Zum Trainieren einer KI sind meist eine große Anzahl von EEG-Trials notwendig, um eine sinnvolle Qualität zu erreichen. Die Gewinnung dieser Trials ist bislang auf die herkömmliche Art und Weise sehr ermüdend für die Testperson. Diese muss auf einen Bildschirm schauen und bekommt monoton zufällig nach links oder rechts zeigende Pfeile eingeblendet, die vorgeben, welche Bewegung sie sich vorstellen muss. Dies fördert das Abschweifen der Testperson, umso länger die Trial Aufnahme andauert, was zu einem verfälschten Ergebnis führen kann. Deshalb ist das Ziel unseres Projektes eine spielerische Erfassung der EEG-Trials. Dies wird mit einem einfach gestalteten Spiel, bei dem auf der horizontalen Achse eine Spielfigur zu einem unbewegten Ziel gesteuert werden soll, realisiert, um eine Ablenkung und Ermüdung der Testperson weitestgehend zu vermeiden. Ein großer Vorteil eines solchen Spiels ist auch, dass die Testperson sofort Rückmeldung über den gegebenen Impuls erhält, indem sie sieht, wohin und ob sich die Spielfigur bewegt, wodurch eine höhere Qualität der Trials erreicht werden kann.



2.1.4 STAKEHOLDER

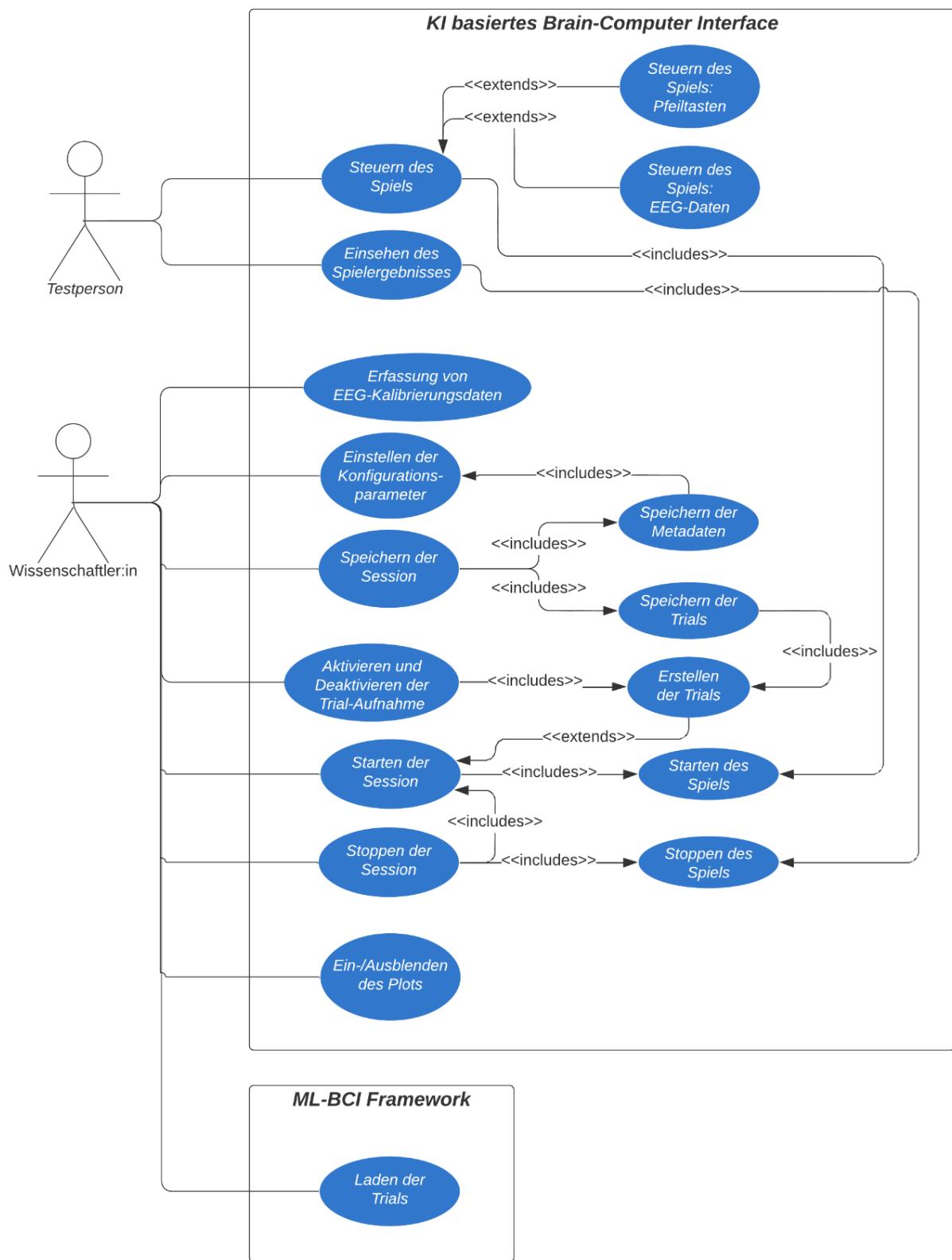
- Kunde (Prof. Dr.-Ing. Manfred Strahnen)
- Testperson
- Wissenschaftler:innen
- Entwickler:innen

2.1.5 FEATURES

- Notwendig
 - Einlesen der EEG-Daten
 - Interpretieren der EEG-Daten
 - Steuerung des Spiels durch Echtzeit-EEG-Daten
 - Spielerische Erfassung der EEG Trials
 - Extrahieren der Trials
 - Live-Plot der EEG-Daten
- Optional
 - Vergleich verschiedener Methoden zur Erhebung der EEG-Daten
 - Laden der Trials in das ML-BCI Framework
 - Wiedergabe von bereits aufgenommenen Sessions



2.2 FUNKTIONALE USER STORIES





2.2.1 FUNKTIONALE USER STORIES DER TESTPERSON

MIN-4 - Steuerung des Spiels: Pfeiltasten

Als eine Testperson möchte ich das Spiel mit den Pfeiltasten (links/rechts) steuern können, um damit die Grundlage für eine weiterführende Implementierung zu legen.

MIN-164 - Steuerung des Spiels: EEG-Daten

Als eine Testperson möchte ich das Spiel mit den EEG-Daten steuern können.

MIN-165 - Einsehen des Spielergebnisses

Als eine Testperson möchte ich das Spielergebnis einsehen können.

2.2.2 FUNKTIONALE USER STORIES VON DEM ODER DER WISSENSCHAFTLER:IN

MIN-8 - Laden der Trials in das ML-BCI Framework

Als ein:e Wissenschaftler:in möchte ich die Möglichkeit haben, die gespeicherten Trials für die weitere Verarbeitung in das ML-BCI Framework zu laden.

MIN-7 - Speichern der Trials

Als Wissenschaftler:in möchte ich die Möglichkeit haben, die erzeugten Daten für die spätere Verwendung im ML-BCI Framework zu speichern.

MIN-27 - Erfassung von EEG-Kalibrierungsdaten: Front-End

Als Wissenschaftler:in möchte ich vor dem Beginn des eigentlichen Spiels dazu in der Lage sein, die zwingend erforderlichen EEG-Kalibrierungsdaten assistiert aufzunehmen, damit die EEG-Daten, welche im Anschluss aufgezeichnet werden, hinreichend korrigieren zu können.

MIN-43 - Erstellen der Trials

Als Wissenschaftler:in möchte ich vom Programm automatisch Trials erfasst bekommen, um diese später im ML-BCI Framework weiter verwenden zu können.



MIN-90 - Ein- und Ausschalten der Trial-Aufnahme über die GUI

Als ein:e Wissenschaftler:in möchte ich die Möglichkeit haben, die Trial Aufnahme über die GUI ein- und auszuschalten.

MIN-83 - Ein- und Ausblenden des Plots

Als Wissenschaftler:in möchte ich in der Lage sein, die benötigten Plots ein- und auszublenden, um jederzeit einen Einblick auf die benötigten Daten haben zu können.

MIN-69 - Einstellung der Konfigurationsparameter

Als Wissenschaftler:in möchte ich in der Lage sein, für das Spiel verschiedene Ansichten anzeigen und die Einstellungen dafür ändern zu können, damit jederzeit die richtigen Einstellungen eingestellt und die dazugehörigen benötigten EEG-Daten angezeigt werden können.

MIN-166 - Starten der Session

Als Wissenschaftler:in möchte ich die Session starten können.

MIN-167 - Stoppen der Session

Als Wissenschaftler:in möchte ich die Session stoppen können.

MIN-168 - Speichern der Session

Als Wissenschaftler:in möchte ich die Session speichern können, um wieder Zugriff auf diese haben zu können.

MIN-169 - Starten des Spiels

Als Wissenschaftler:in möchte ich, dass das Spiel durch das Starten der Session beginnt.

MIN-170 - Stoppen des Spiels

Als Wissenschaftler:in möchte ich, dass das Spiel stoppt und den Punktestand anzeigt, wenn die Session beendet wird.



MIN-171 - Speichern der Metadaten

Als Wissenschaftler:in möchte ich die Möglichkeit haben, die eingegebenen Metadaten speichern zu können, um jederzeit zu wissen, welche Einstellungen vorgenommen wurden und welche Aufzeichnungen zu welcher Testperson gehören.

2.3 NICHT-FUNKTIONALE USER STORIES

MIN-3 - Kontinuierliches Auslesen der EEG-Daten in Python

Als ein:e Entwickler:in möchte ich die EEG-Daten kontinuierlich auslesen, damit sie für weitere Verarbeitungsschritte zur Verfügung stehen.

MIN-5 - Prototypischer Algorithmus Teil 1

Als Entwickler:in möchte ich aus den EEG-Daten aller Channels den kleinen Laplacian auslesen und mithilfe dessen Durchschnittswert die Werte der Elektroden C3 und C4 korrigieren, um damit das Rauschen zu entfernen und die Daten nutzbar zu machen.

MIN-6 - Prototypischer Algorithmus Teil 2

Als ein:e Entwickler:in möchte ich das Zeitsignal mithilfe von FFT oder Multitaper in ein Power Spectral Density Signal umwandeln, um daraus mittels Integration die Energie des für uns interessanten Frequenzbereichs zu berechnen. Diese Werte möchte ich in eine interpretierbare Richtung umwandeln.

MIN-24 - Projektdokumentation: Grundstruktur

Als Product Owner möchte ich bestimmte Kriterien für die Projektdokumentation erfüllen.

MIN-44 - Definition der Schnittstellen der Komponenten

Als Entwickler:in möchte ich eine klar definierte Übersicht der Schnittstellen zwischen den einzelnen Softwarekomponenten, um diese korrekt implementieren zu können.



MIN-86 - Optimierung des Algorithmus

Als Entwickler:in möchte ich die Genauigkeit des Algorithmus verbessern, um eine bessere Bestimmung der Labels zu erhalten.

MIN-72 - Anpassung des Pong Spiels

Als Entwickler:in möchte ich das bisher implementierte Spiel vereinfachen, um die Aufnahme der Trials zu verbessern.

MIN-47 - Überarbeitung des EEG-Daten-Einlesens

Als Kunde möchte ich, dass die EEG-Daten kontinuierlich und ohne eine Latenz eingelesen und übertragen werden, um im späteren Verlauf Echtzeitdaten zur Verfügung zu haben.

MIN-40 - Verbesserung prototypischer Algorithmus

Als ein:e Wissenschaftler:in möchte ich sinnvolle Daten aus dem Algorithmus für eine spätere Weiterverarbeitung bekommen. Diese sollten eine mit einer Korrektheit der Vorhersage von min. 80 % aufweisen.



3. ERGEBNIS AUS NUTZERSICHT

3.1 ERGEBNIS AUS NUTZERSICHT: TESTPERSON

3.1.1 SPIEL

Spielobjekte



-
- **Spieler** (Player): Dargestellt durch ein blaues Quadrat. Dieses ist beweglich und kann von der Testperson gesteuert werden.
 - **Ziel** (Target): Dargestellt durch einen roten unbeweglichen Kreis. Dieser wird in einem zufälligen Abstand nach dem Zufallsprinzip entweder links oder rechts vom Spieler platziert.

Spielidee

Der Spieler muss das Ziel einsammeln, indem er sich auf der x-Achse (Horizontale) zum Zielobjekt bewegt, bis er mit diesem kollidiert. Bei einer Kollision pausiert das Spiel für einen kurzen Moment, damit die Testperson sich sammeln kann. Außerdem wird während der kurzen Pause die Zeit über dem Zielobjekt angezeigt, welche der Spieler benötigt hat, um dieses einzusammeln. Nach der Pause verschwindet das Zielobjekt und wird nach dem Zufallsprinzip neu positioniert.

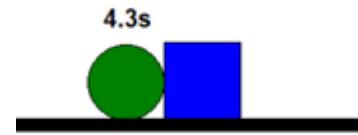
Spielbeginn

Zu Beginn der Session sieht die Testperson einen weißen Bildschirm für 30 Sekunden, sodass sich die Testperson beruhigen kann. Anschließend erscheint das Spiel und die Testperson kann sofort beginnen, das Spiel zu steuern.



Features

- Die Bewegung des Spielerobjektes auf der horizontalen Achse (x-Achse) nimmt bis zum Stillstand linear ab, solange keine neue Richtungsaktualisierung angestoßen wird.
- Kollidiert das Spielerobjekt mit dem Zielobjekt, wird das Zielobjekt grün eingefärbt, um der Testperson eine positive Resonanz zu vermitteln. Außerdem wird die benötigte Zeit bis zum Erreichen des Zielobjektes eingeblendet.
- Erreicht der Spieler das Ziel in der dafür vorgesehenen entfernungsabhängigen Zeit nicht, verschwindet das Ziel und erscheint an einer neuen zufälligen Position wieder.



Endansicht und Punktestand

Beendet der oder die Wissenschaftler:in die Session, erscheint für die Testperson ein weißes Fenster, das ihr Rückmeldung über ihre Spielleistung gibt. Dieses Fenster enthält folgende Informationen:

- Wie viele Zielobjekte erschienen sind und wie viele davon getroffen wurden.
- Prozentuale Anzahl der getroffenen Ziele
- Mittlere Zeit, die man nach einem Treffer noch gehabt hätte, um das Ziel zu treffen (in Prozent)

3.2 ERGEBNIS AUS NUTZERSICHT: WISSENSCHAFTLER:IN

3.2.1 BEDIENUNG DER GRAFISCHEN OBERFLÄCHE

Alle Werte, welche im Konfigurationsfenster eingegeben werden, werden beim Starten der Session validiert. Sobald der Wert eines Eingabefeldes invalide ist, verfärbt sich dessen Titel rot und die Session kann nicht gestartet werden.



Bis auf das Kommentarfeld sind alle Eingabefelder ausschließlich vor dem Starten einer Session veränderbar. Sie sind jedoch, bis zum Speichern oder Verwerfen einer Session deaktiviert (ausgegraut), im Konfigurationsfenster einsehbar.

The screenshot displays the 'Settings' configuration window. It includes sections for 'Subject' (with fields for ID, Age, and Sex), 'Algorithm' (with fields for Threshold, f_min, and f_max), 'Sliding Window Size' (1000), 'Sliding Window Offset' (200), 'Trial' (Trial Min-Duration: 1000), and a 'Comment' text area. On the right, there's a 'General' panel with two toggle switches: 'Plot' (off) and 'Trial Recording' (on). At the bottom is a 'Start Session' button.

Testperson-Sektion (Subjekt): Erfasst Testperson-spezifische Daten

Name	Beschreibung	Eingabewert
ID	Identifikationsnummer (ID), welche der Testperson zugeordnet ist	ganzzahliger Wert; größer als Null
Age	Alter der Testperson	ganzzahliger Wert; zwischen eins und 100
Sex	Geschlecht der Testperson	'D' für divers; 'F' für weiblich; 'M' für männlich



Algorithmus-Sektion: Algorithmus-spezifische Daten, welche für die Datenanalyse relevant sind

Name	Beschreibung	Eingabewert	Schrittweite
Threshold	Grenzwert zur Berechnung des Labels	Dezimalzahl; [0; 3]	-
f_min	minimale Frequenz	Dezimalzahl; [0; f_max[-
f_max	maximale Frequenz	Dezimalzahl; [f_min; ∞[-
Sliding Window Size	Größe des Sliding Windows (in ms)	Ganze Zahl; [200; 2000]	200
Sliding Window Offset	Größe der zeitlichen Differenz zwischen zwei Sliding Windows (in ms)	Ganze Zahl; [40; 400]	40

Trial-Sektion: Trial-spezifische Daten, welche für die Erfassung von Trials relevant sind

Name	Beschreibung	Eingabewert	Schrittweite
Trial Min-Duration	Mindestdauer eines Trials (in ms)	Ganze Zahl; [800; 1500]	100

Kommentar-Sektion: Kommentarfeld für den oder die Wissenschaftler:in, welches zu jedem Zeitpunkt bis zum Speichern der Session bearbeitet werden kann

General-Sektion: Ein- und Ausschalten verschiedener Funktionen mit Checkbuttons (False/True)

Name des Checkbuttons	Funktion	Verfügbarkeit
Plot	Schaltet den Live-Plot an und aus	Kann jederzeit ein- und ausgeschaltet werden
Trial	Schaltet das Aufnehmen	Kann nur vor dem Starten



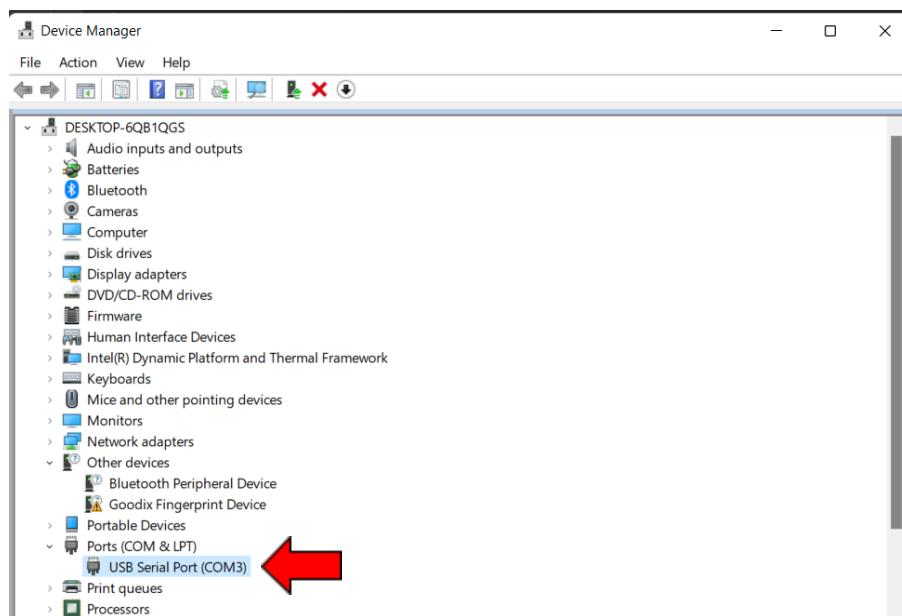
Recording	von Trials an und aus	der Session ein- und ausgeschaltet werden
------------------	-----------------------	-------------------------------------------

3.2.2 DATENERFASSUNG

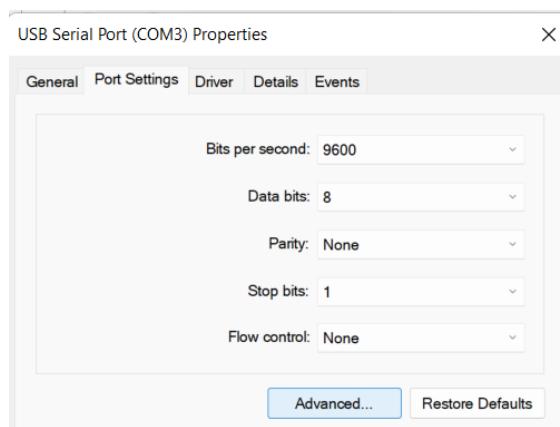
Porteinstellungen

Der Latenz Timer des verwendeten seriellen Ports muss korrekt eingestellt werden, um einen konsistenten Datenstrom zu erhalten.

- Unter Windows:
 - Gerätetmanager öffnen → Rechtsklick auf den seriellen Port des OBCI-Boards

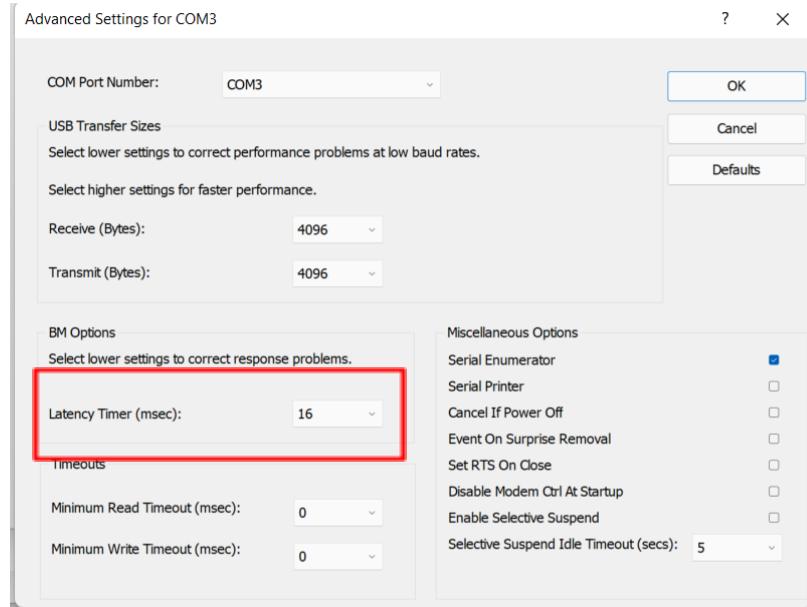


- Auswählen von Properties > Port Settings > Advanced





- Einstellen des “Latency Timer” auf 1 ms



- Unter Linux:

Die Latenz wird automatisch vom Programm auf 1 ms eingestellt

Verbindung zum Board herstellen

1. Die Verbindung zum Board wird durch das Klicken des Buttons “Connect Board” hergestellt
2. Weiteres mögliches Vorgehen:

a. Verbindung konnte hergestellt werden:

Der Button wird zum Button “Start Session”. Nun können die Einstellungen für die Session eingegeben und die Session gestartet werden. Nach dem Starten der Session werden die EEG-Daten des Boards automatisch eingelesen.

b. Verbindung konnte nicht hergestellt werden:

Ein Informationsfenster bezüglich des fehlgeschlagenen Verbindungsbaus erscheint und es kann erneut versucht werden, eine Verbindung herzustellen. Mögliche Ursachen für das Scheitern des Verbindungsbaus:



- Die Pegel des OpenBCI Headsets sind nicht auf null gesetzt:
Das OpenBCI Headset wurde im Rahmen einer Bachelorarbeit hardwaremäßig erweitert und beim Starten des Headsets muss darauf geachtet werden, dass ein bestimmter Stecker kurzzeitig ausgesteckt wird (bei Fragen an Prof. Strahnen wenden)
- Headset ist ausgeschaltet: Headset einschalten und erneut versuchen, die Verbindung herzustellen
- Der Dongle ist nicht an den PC angeschlossen: Dongle anschließen, Headset einschalten und erneut versuchen, die Verbindung herzustellen
- Der Dongle verbindet sich mit einem anderem Bluetooth-Gerät, als mit dem Headset: Am Besten alle anderen Bluetooth-Geräte, die sich mit dem Gerät verbinden wollen aus dem Raum entfernen oder vom Strom entfernen (Tipp: Dies am Besten vor dem Einsticken des Dongles tun)

3.2.3 SPIEL

Wird im Folgenden von dem Spieler gesprochen, ist damit die Spielfigur gemeint und nicht die Testperson.

Starten des Spiels

1. Durch das Drücken des Buttons “Start Session” wird das Spielfenster gestartet.
2. Zunächst erscheint im Spielfenster für 30 Sekunden (Kalibrierungsdauer) ein weißes Bild zum Sammeln von Kalibrierungsdaten. Während dieser Zeit wird die Testperson gebeten, sich zu entspannen. Die EEG-Daten dieser Zeitspanne werden als Kalibrierungtrial erfasst, sofern “Trial Recording” aktiviert ist, und können später gemeinsam mit den aufgenommenen Trials gespeichert werden.
3. Nach dem Ablauen der 30 Sekunden startet das Spiel automatisch.



Steuerung

- Pfeiltasten
- Algorithmus (EEG-Headset)

Ändern der Steuerungsstrategie

- Öffnen der `config.py` Datei
- Im Bereich "Game" befindet sich die Konstante `USED_STRATEGY_CLASS`. Dieser Konstante muss die verwendete Strategie zugewiesen werden.
- Möglichkeiten:
 - `strategy.KeyStrategy` → Steuerung durch die Pfeiltasten
 - `strategy.AlgorithmsStrategy` → Steuerung durch die EEG-Daten

Trial-Erfassung

- Die Aufnahme eines Trials wird gestartet, wenn der Spieler sich in die Richtung des Ziels bewegt.
- Die Aufnahme wird beendet, wenn:
 - der Spieler das Ziel erreicht
 - der Spieler die Richtung ändert
 - der Spieler das Ziel nicht in der gegebenen Zeit erreicht und das Ziel an einer anderen Stelle neu erscheint
- Durch die verschiedenen Kriterien, die zum Beenden einer Trialaufnahme führen, wird erreicht, dass nur valide Trials erfasst werden.
- Als Trial werden nur die Trials markiert, die auch mindestens so lang sind wie die in der GUI eingestellte Trial-Mindestdauer.



Veränderbare Konstanten in der Konfigurationsdatei

Auswählen der Konfigurationsdatei: `config.py`

- `TARGET_RESPAWN_TIME` → Zeitspanne, für welche der Spieler sich nicht mehr bewegen kann, sobald das Ziel getroffen wurde (in ms). Hierdurch erhält die Testperson eine Rückmeldung (Farbwechsel, Zeiteinblendung) und kann sich gleichzeitig kurz ausruhen
- `MIN_DISTANCE_TARGET` → Mindestabstand (in Pixeln), den das Ziel beim Wiedererscheinen zum Spieler haben muss, um eine Mindestlänge des Trials garantieren zu können.
- `TIME_TO_STOP_PLAYER` → Zeit (in ms), die der Spieler ab dem letzten Richtungsimpuls benötigt, um zum Stillstand zu kommen
- `TIME_TO_CATCH_PER_PIXEL` → Zeit pro Pixel (in ms), die der Spieler hat, um das Ziel zu erreichen
- `SHOW_SCORE` → Einblendung der benötigten Zeit bei einem Treffer
 - True: Punktestand wird eingeblendet
 - False: Punktestand wird nicht eingeblendet
- `OBJECT_SIZE` → Konstante zur Berechnung der Größe des Spielers und des Ziels, um diese an die Bildschirmgröße anzupassen
- `CALIBRATION_TIME` → Dauer (in s) für welche der Kalibrierungsbildschirm vor dem Spiel zu sehen ist

3.2.4 DATENANALYSE

Ändern der benutzten PSD-Methode

1. Öffnen des Skriptes `cursor_control_algorithm.py`
2. Der Konstante `USED_METHOD` muss die gewünschte PSD-Methode zugewiesen werden, welche in dem Enum `PSD_METHOD` aufgelistet werden
3. Verschiedene Möglichkeiten:
 - `PSD_METHOD.fft` → FFT
 - `PSD_METHOD.multitaper` → Multitaper-Methode



- `PSD_METHOD.burg` → Burg-Algorithmus (für die Burg Methode muss die `spectrum` Bibliothek importiert werden)
- `PSD_METHOD.periodogram` → Periodogram-Methode

Signalbezogene Werte

Auswählen der Konfigurationsdatei: `config.py`

- `BCI_CHANNELS` → Array mit den Namen der verbundenen Messpunkte
An die Position des Kanals wird die Bezeichnung des jeweiligen Messpunkts geschrieben.
Mit dieser Auswahl an Kanälen, die auch konkrete Steckplatzpositionen am Headset repräsentieren, können sowohl der large als auch der small laplacian abgebildet werden.

Unterschieden wird zwischen dem small Laplacian

```
BCI_CHANNELS = ['C3', 'Cz', 'C4', 'P3', 'Pz', 'P4', 'O1', 'O2', 'FC5',
    'FC1', 'FC2', 'FC6', 'CP5', 'CP1', 'CP2', 'CP6']
```

und dem large Laplacian:

```
BCI_CHANNELS = ['C3', 'Cz', 'C4', 'P3', '?', 'P4', 'T3', '?', '?', 'F3',
    'F4', '?', '?', '?', 'T4']
```

- `CH_NAMES_WEIGHT` → wird ein Array übergeben, dass die jeweilige Gewichtung des Signals enthält. Diese befindet sich an der Stelle, wo der jeweilige Name des Kanals (vom Signal) äquivalent zum `BCI_CHANNELS` Array steht.
 - Ist der Kanal an der Position mit einem Signal verbunden, so beträgt die Gewichtung den Wert 1, wenn nicht, entspricht sie 0

Unterschieden wird zwischen dem small Laplacian

```
CH_NAMES_WEIGHT = [1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1]
```

und dem large Laplacian:

```
CH_NAMES_WEIGHT = [1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1]
```



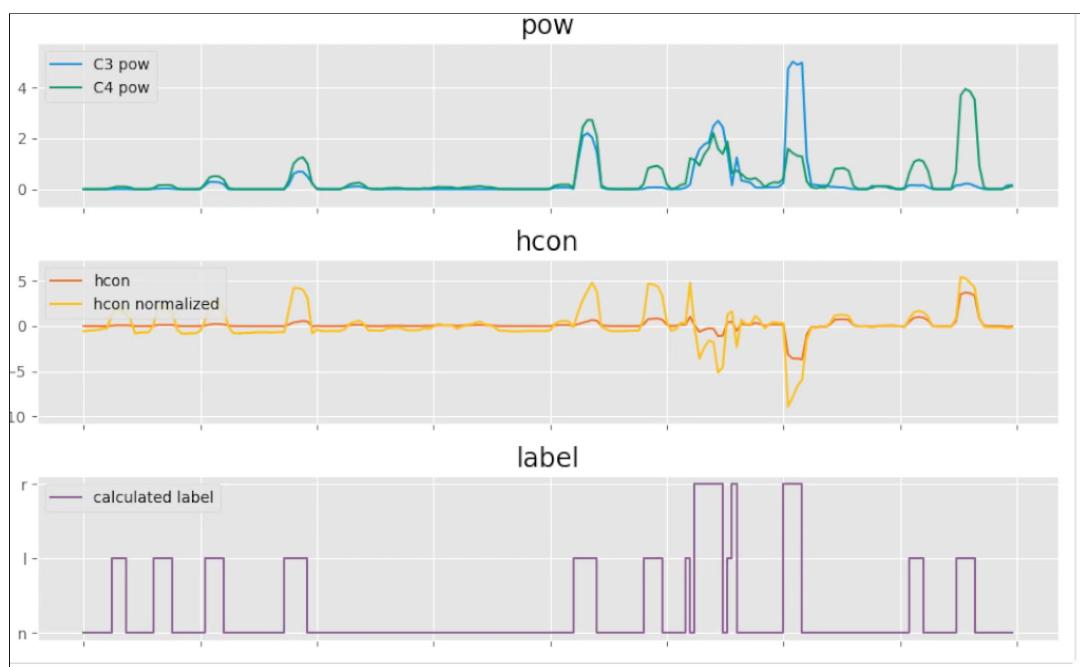
- **WEIGHT** → Ganzzahliger Wert. Beschreibt die Gewichtung des C4-Signals bei der Berechnung der Energiedifferenz der Signale C4 und C3.

3.2.5 LIVE-PLOT

Durch Einschalten des Plots im Konfigurationsfenster, erweitert sich dieses. Die verschiedenen Plots, welche die resultierenden Signalwerte nach den diversen Verarbeitungsschritten der Datenanalyse (siehe Kapitel 4.5) zeigen, werden im Konfigurationsfenster dann auch dargestellt.

Folgende Verarbeitungsschritte werden grafisch live dargestellt:

- **pow**: Plot der Energien der Bandbreiten von C3 und C4
(Also die Signale nach dem vierten Verarbeitungsschritt der Datenanalyse)
- **hcon**: Plot des horizontalen Steuerungswerts, welcher sich aus der Energiedifferenz von C4 und C3 berechnen lässt. Ergibt sich aus dem fünften bzw. sechsten Verarbeitungsschritt der Datenanalyse (mit und ohne Normalisierung)
- **label**: Plot des berechneten Labels, wobei 'r' für rechts, 'l' für links und 'n' für nicht definiert steht





3.2.6 DATENEXTRAKTION

Durchführen einer Session

1. Starten der Verbindung zum Board (siehe Kapitel 3.2.2)
2. Einstellen der Parameter in dem Konfigurationsfenster (siehe Kapitel 3.2.1)
3. Starten der Session durch Drücken des Buttons "Start Session", welcher danach zum "Abort"-Button wird
4. Kalibrierung
 - a. Während der Kalibrierung erscheint neben dem "Abort"-Button ein blauer Ladebalken, welcher den Kalibrierungsfortschritt anzeigt
 - b. Mit dem "Abort"-Button kann die Kalibrierung abgebrochen werden. Die Session wird dann verworfen und der "Abort"-Button wird wieder zum "Start Session"-Button
 - c. Ist die Kalibrierung beendet, verschwindet der blaue Ladebalken und der "Abort"-Button wird zum "Stop Session"-Button
5. Erfassung und Weiterverarbeitung der EEG-Daten
 - a. Fall: Trial-Aufnahme ist eingeschalten
 - i. Wird ein valider Trial aufgenommen, wird der Start- und Endzeitpunkt sowie das Label und die enthaltenen EEG-Daten auf der Konsole ausgegeben
 - b. Fall: Trial-Aufnahme ist ausgeschalten
 - i. Keine Konsolenausgabe
6. Stoppen der Session durch das Drücken des "Stop Session"-Buttons

Hierdurch wird die Datenübertragung beendet und der "Stop Session"-Button wird durch die Buttons "Save Session" und "Discard Session" ersetzt. Außerdem erscheint ein Informationsfenster, das darüber informiert, wie viele Trials gespeichert wurden.
7. Weiteres Vorgehen mit der aufgenommenen Session
 - a. Fall: Verwerfen der Session durch das Drücken des "Discard Session"-Buttons



b. Fall: Speichern der Session durch das Drücken des “Save Session”-Buttons

i. Der aufgenommene Datenstrom, die Markierungen der Trials, die Labels der Trials und die Metadaten zu der Session werden in einer Datei mit folgendem Namensschema im Verzeichnis session gespeichert:

```
session-<subject-id>-<ddmmYYYY>-<HHMMSS>.npz
```

ii. Ausgabe der Metadaten der gespeicherten Session auf der Konsole

Mögliche Labels

- INVALID = 99 (Nullwert)
- LEFT = 0 (links)
- RIGHT = 1 (rechts)
- CALIBRATION = 2 (Kalibrierungs-Trial)

Erfasste Metadaten

Metadaten	Erfassungsart
ID der Testperson	Über das Einstellungsfenster
Geschlecht der Testperson	Über das Einstellungsfenster
Alter der Testperson	Über das Einstellungsfenster
Aufnahmedatum der Session	Wird im MetaData Konstruktor erfasst
Uhrzeit beim Start der Session	Wird im ConfigController erfasst
Abtastrate in Hz	Die Abtastrate des OpenBCI Headsets (125 Hz) ist als Defaultwert im Konstruktor hinterlegt
EEG-Ausrüstung	‘BCI’ ist als Defaultwert im Konstruktor hinterlegt
Art der Datenerfassung	‘game’ ist als Defaultwert im Konstruktor hinterlegt



Kanalbelegung	In der Datei config.py änderbar (siehe 3.2.4)
Anzahl der erfassten Trials	Wird automatisch im Code erfasst
Anzahl der verschiedenen erfassten Labels	Wird automatisch im Code erfasst
Kommentar von dem oder der Wissenschaftler:in	Wird im Konfigurationsfenster erfasst

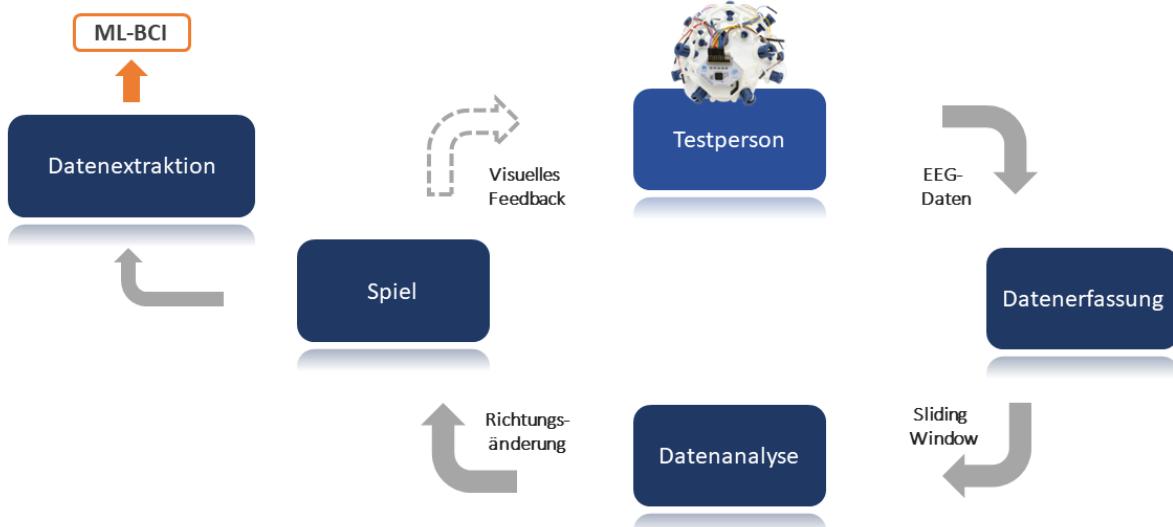
3.2.7 DATENSATZ LOADER

Mit der Funktion `get_channel_rawdata()` im Skript `game_dataset_loader.py` ist es später möglich, die Session-Daten des Spiels zu laden. Dieses Skript basiert auf `bcic_data_loading.py` vom ML-BCI Framework und sollte somit leicht im ML-BCI Framework integrierbar sein. Die Funktion benötigt die Speicheradresse der npz-Datei und optional eine Liste, nach welchen Channels gefiltert werden soll. Des Weiteren gibt es die Möglichkeit eines Notch Filters für die rohen Daten, welchen man in `config.py` ein- und ausschalten kann. Als Rückgabewert gibt die Funktion je ein NumPy-Array mit den rohen Daten und das zugehörige Label für jeden Zeitpunkt zurück.



4. TECHNISCHES KONZEPT

4.1 GESAMTÜBERSICHT



Die Grundarchitektur ist zirkular aufgebaut, wobei der Anfang bei der Testperson liegt, deren EEG-Ströme mithilfe des BCI-Headset erfasst werden. Diese EEG-Daten werden von der Datenerfassung eingelesen und zu sogenannten Sliding Windows verarbeitet, welche an die Datenanalyse weitergegeben werden. Die Datenanalyse berechnet aus diesen Sliding Windows den darin gegebenen Richtungsimpuls und gibt diesen an das Spiel weiter, welches damit eine Spielfigur steuert. Durch die Bewegung dieser Spielfigur erhält die Testperson ein visuelles Feedback.

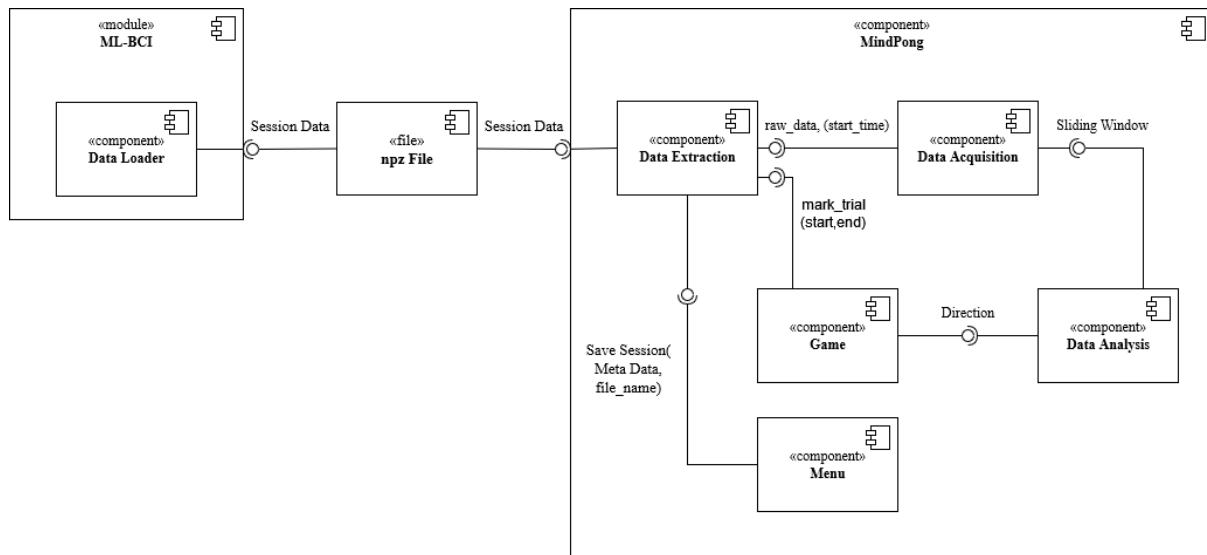
Die Speicherung der wie oben beschrieben erfassten Trials erfolgt durch die Datenextraktion, indem ihr das Spiel mitteilt, welche Teile des aufgenommenen Datenstroms valide Trials sind und diese markiert. Die Datenextraktion speichert zum Schluss die Session.

Das Endziel dieses Projekts ist es, diese gespeicherten Sessions in dem ML-BCI Framework zu speichern. Dies wird mithilfe eines Loaders realisiert.



4.2 KOMPONENTEN DIAGRAMM

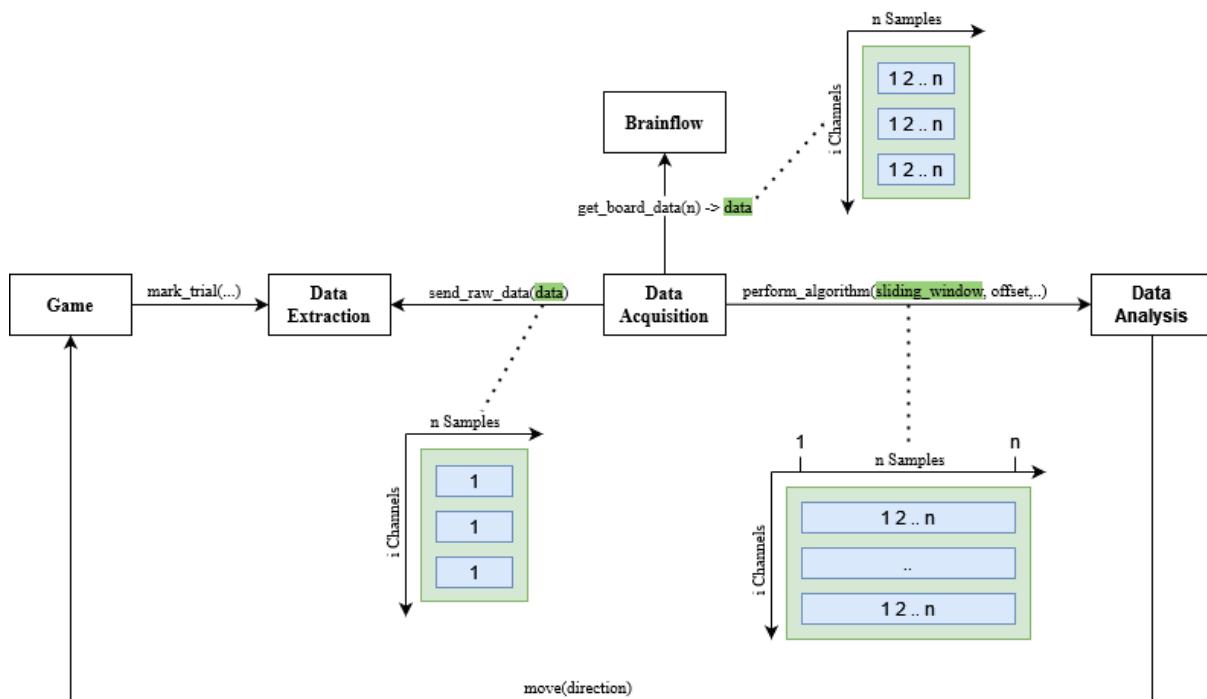
Im Folgenden ist ein Komponentendiagramm der Software abgebildet. Dieses zeigt die Schnittstellen zwischen den einzelnen Modulen und wie diese miteinander zusammenhängen. Außerdem ist der Zusammenhang mit dem ML-BCI Framework ebenfalls abgebildet.





4.3 ÜBERSICHT DER SCHNITTSTELLEN ZWISCHEN DEN KOMPONENTEN

Das folgende Diagramm soll verdeutlichen, wie die Daten zwischen den einzelnen Komponenten übergeben werden.



Die Datenerfassung (Data Acquisition) übergibt der Datenanalyse (Data Analysis) mit der Funktion `perform_algorithm()` ein Sliding Window. Dieses Sliding Window ist ein zweidimensionales Array, welches für jeden Channel ein eigenes Array enthält. Die Arrays der einzelnen Channels enthalten alle Samples, die zwischen Zeitpunkten t_1 und t_n erfasst wurden.

Der Datenextraktion (Data Extraction) übergibt die Datenerfassung die Daten mit der Funktion `send_raw_data()` als zweidimensionales Array. Dieses Array enthält für jeden Channel ein Array, indem jeweils das aktuellste Sample dieses Channels gespeichert ist.

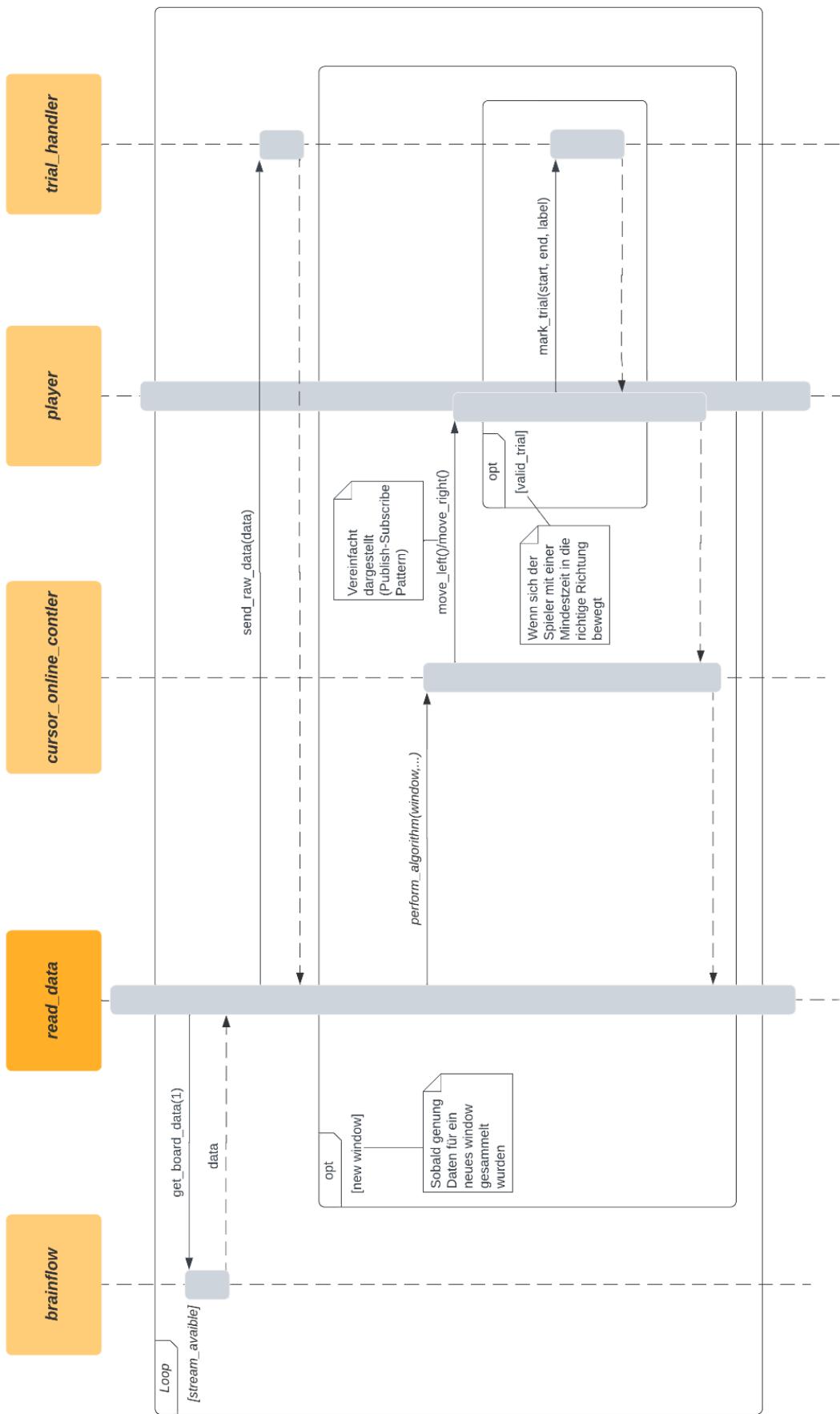
Die Datenerfassung fordert die EEG-Daten vom Headset mithilfe der Brainflow-Funktion `get_board_data(n)` an. Diese Funktion gibt die Daten in demselben Format wie Sliding Windows zurück. In den Unterarrays sind die n zuletzt erfassten Samples der jeweiligen Channels gespeichert.



Die Datenanalyse übergibt das aus den Sliding Windows berechnete Label, mit der Methode move, dem Spiel (Game). Dieses teilt der Datenextraktion Startzeitpunkt, Dauer und Label eines Trials mit der Funktion `mark_trial()` mit.

4.4 TRIAL AUFNAHME UND ERFASSUNG DER SLIDING WINDOWS

Folgendes Sequenzdiagramm zeigt den generellen Ablauf der Aufnahme von Trials, wenn eine Session gestartet und die Option Trial aufnehmen gewählt wurde. In der Datenerfassung wird mithilfe der Brainflow Bibliothek über die komplette Dauer der Session Daten eingelesen. Die Datenerfassung puffert diese Daten einerseits in einem Ringpuffer und sendet dieselben Daten gleichzeitig an die Datenextraktion. Auch die Datenextraktion besitzt einen Puffer in Form einer Liste, welcher zur Speicherung des gesamten Datensatzes beim Speichern einer Session erforderlich ist. Für den Algorithmus werden sogenannte Sliding Windows benötigt, diese sind sich zeitlich überlappende Datenfenster. Hat die Datenerfassung ausreichend Daten für ein neues Sliding Window erhalten, sendet sie diese an den Algorithmus, welcher aus den gesammelten Daten ein Label berechnet. Wenn der Algorithmus links oder rechts erkannt hat, sendet dieser dann einen Steuerungsbefehl an das Spiel, um die Spielerfigur zu bewegen. Hat das Spiel einen gültigen Trial erkannt, ruft dieses die Datenextraktion zum Markieren des Trials im entsprechenden Datenpuffer auf.





4.5 CURSOR CONTROL ALGORITHMUS

Die eingelesenen EEG-Daten werden in der Datenanalyse, deren Herzstück der Cursor Control Algorithmus ist, in ein Steuerungssignal transformiert, mit dem schließlich das Spiel gesteuert werden soll.

Der nachfolgend beschriebene Algorithmus basiert auf dem Cursor Control Algorithmus nach Stieger¹ und Wolpaw², besteht hier aber aus 7 Verarbeitungsschritten der Daten. Der Algorithmus läuft immer über ein Sliding Window.

1. Standardisierung der Daten:

Die Daten werden in der Funktion `standardize_data(in_data)` für jedes Sliding Window standardisiert.

$$X_{\text{standardized}} = \frac{X - \mu}{\sigma}$$

X = input values
 μ = $\text{mean}(X)$
 σ = $\text{standard deviation}(X)$

Somit wird erreicht, dass die in Schritt 4 berechneten integrierten Energiewerte nicht zu groß werden. Folglich können sie beim späteren Darstellen im Live-Plot zusammen mit anderen Werten wie hcon geplottet werden.

Der durch die Standardisierung verursachte Datenverlust, aufgrund von möglichen Rundungsfehlern, kann hier als nicht relevant betrachtet werden. Dies wurde in Versuchen auch überprüft.

2. Spatial Filtering:

Im Spatial Filtering werden die Signalwerte von C3 und C4 mithilfe der jeweils umgebenden Elektroden gefiltert.

Zur Vorbereitung werden zuerst die Elektroden um die C3 und C4 Elektrode der linken und rechten Gehirnhälfte zugeordnet. Über beide Mengen wird jeweils ein durchschnittliches Signal mittels der Methode `calculate_laplacian(samples: np.ndarray)` berechnet.

¹ Continuous sensorimotor rhythm based brain computer interface learning in a large population

² Conversion of EEG activity into cursor movement by a brain-computer interface (BCI)



Im weiteren Verlauf kann von den Signalen der C3 und C4 Elektrode der jeweils berechnete Laplacian abgezogen werden.

3. Spectral Analysis:

In der Spektralanalyse wird ermittelt, zu welchen Anteilen die einzelnen Frequenzen in einem Signal stecken. Die Auswahl ist hier zwischen FFT-basierten Methoden wie, Periodogram, Multitaper und FFT und Burg, einer parametrischen Methode möglich. Die Multitaper Methode birgt den Vorteil, dass die Rückgabeparameter bereits einen gewünschten Frequenzbereich umfassen.

Um eine sinnvolle Aussage über die Anteile von Frequenzen in Signalen zu tätigen, sollten mindestens zwei vollständige Wellen von `FMIN` enthalten sein (zum Beispiel sollte bei einem unteren Frequenzbereich von 8 Hz die Mindestgröße des Sliding Windows $2 / (8 \text{ Hz}) = 250 \text{ ms}$ betragen).

4. Band-Power Calculation:

Als nächsten Schritt wird über einen gewünschten Frequenzbereich integriert. Dabei müssen die benötigten Frequenzen (jene, die innerhalb des Intervalls liegen, der durch die Konstanten `[FMIN, FMAX]` ausgedrückt werden kann) herausgefiltert werden, sofern die verwendete Spektralanalyse nicht Multitaper war.

Anschließend wird durch Integration über die Frequenzen, mithilfe der Trapezregel, die Gesamtenergie des Signals für C3 und C4 über diesen bestimmten Frequenzbereich berechnet ($C3_{\text{pow}} / C4_{\text{pow}}$). Wir nutzen die Trapez- und nicht die Quadratregel, da sich bei unseren Daten Problem ergaben. Signalwerte können oft annähernd Null sein, was zu Fehlern bei der Quadratur innerhalb der `quad()` Methode von `scipy` führte.

5. Differenz der Energie-Werte von C3 und C4:

Die Signale von C3 und C4 verhalten sich kontralateral für rechts und links. Somit kann durch die Differenz der berechneten Werte $C3_{\text{pow}}$ und $C4_{\text{pow}}$ ein Steuerungswert `hcon` abgeleitet werden. Über die Zeit können die berechneten `hcon` durchaus als ein Steuerungssignal `hconsignal` betrachtet werden.



$$hcon = C3_{pow} - C4_{pow}$$

6. Standardisierung des berechneten Steuersignals:

Das berechnete Steuerungssignal $hcon_{signal}$ wird zu Beginn über 30 Sekunden in einem Ringpuffer gesammelt.

Dies geschieht in der EEG-Kalibrierung-Phase, währenddessen sich die Testperson noch in Ruhe befindet und noch kein Spiel sehen kann.

Über die gesammelten Werte werden nun zum einen der Mittelwert, aber auch die Standardabweichung berechnet. Die Standardisierung (siehe Formel in 1.) setzt sich aus dem aktuellen $hcon$ als X und den zuvor berechneten Mittelwert und Standardabweichung zusammen.

7. Berechnung des Labels:

Zur Berechnung des Labels wird ein Threshold (Grenzwert) t genutzt. Dieser sorgt dafür, dass nur deutliche Ausschläge als (vorgestellte) Bewegung gewertet werden. Andere Einflüsse werden hier zusätzlich herausgefiltert.

Das Label L oder auch der Steuerungsbefehl setzt sich daher folgendermaßen zusammen (l = links, n = nicht definiert, r = rechts):

$$\begin{aligned} L \in \{n, l, r\} \\ n \in \{x \mid -t \leq x \leq t\} \\ l \in \{x \mid x > t\} \\ r \in \{x \mid x < -t\} \end{aligned}$$

Im Anschluss wird das Publish-Subscribe-Pattern (siehe Kapitel 4.7) verwendet, um den berechneten Steuerungsbefehl an das Spiel weiterzuleiten.

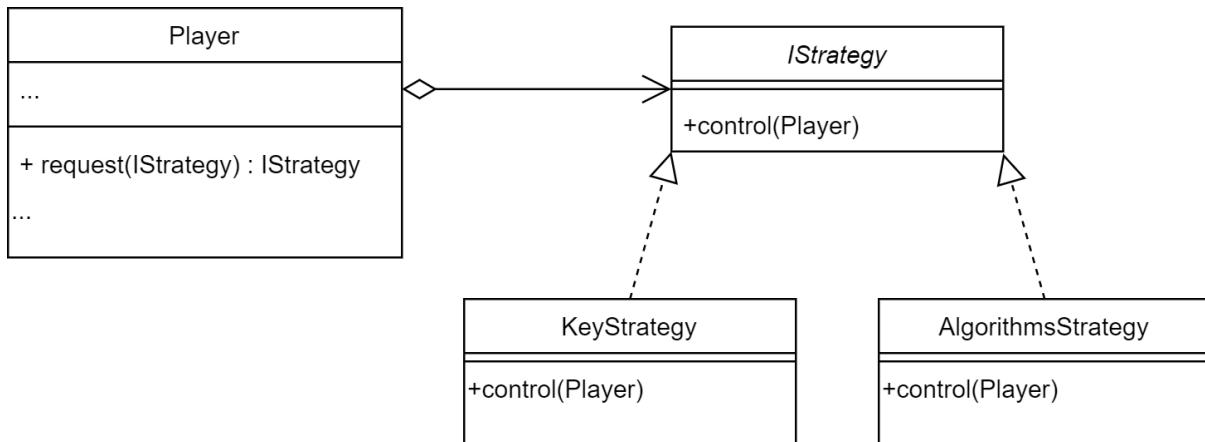
8. Füllen der Plot-Queues:

Die FIFO-Queues stellen dem Live-Plot (siehe Kapitel 4.9) bestimmte Datenwerte zur Verfügung.

Am Ende des Algorithmus werden das berechnete Label, sowie andere Werte aus Zwischenschritten des Algorithmus wie $C3_{pow}/C4_{pow}$ und $hcon$ / standardisierte $hcon$ in jeweils eine eigene Queue geschrieben. Herausgenommen werden diese dann bei der Ausführung des Live-Plots.



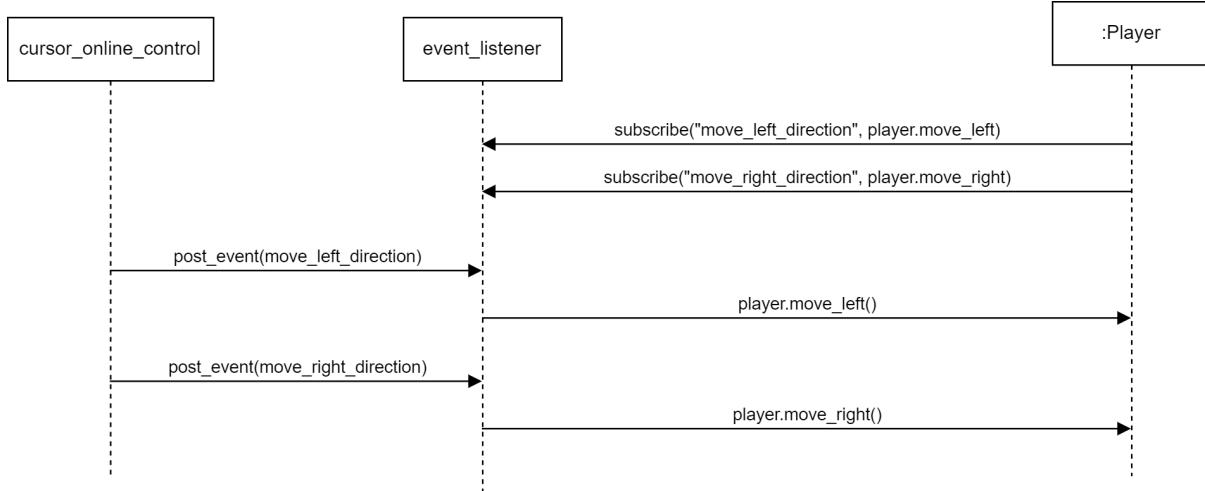
4.6 STRATEGY-PATTERN ZUR VERWALTUNG DER GAME CONTROL STRATEGIEN



Die Auswahl der Steuerungsstrategie des Player-Objekts im Spiel wird mithilfe des Strategy Patterns realisiert. Hierfür gibt es die Schnittstelle **IStrategy**, von welcher die konkreten Strategy-Klassen erben. Programmiersprachen wie Java und C++ verfügen über ein Schlüsselwort für Schnittstellen, Python hingegen nicht. Schnittstellen werden in Python durch eine Klasse mit abstrakten Methoden dargestellt. Python unterscheidet sich noch in einem weiteren Aspekt von anderen Sprachen. Es verlangt von der Klasse, die die Schnittstelle implementiert, nicht, dass sie alle abstrakten Methoden der Schnittstelle definiert. In unserem Fall ist die abstrakte Methode `control()`. Dessen Definition wird in den jeweiligen Strategy-Klassen konkret implementiert. Die Klasse **KeyStrategy** enthält die Steuerung durch die Pfeiltasten der Tastatur und **AlgorithmsStrategy** enthält die Steuerung durch den Algorithmus. Dem Player-Objekt wird über die `request`-Methode mitgeteilt, welche Strategy-Klasse verwendet wird.



4.7 PUBLISH-SUBSCRIBE-PATTERN ZUR REALISIERUNG DER ALGORITHMUS-STRATEGIE



Die Steuerung des Player-Objekts mithilfe des Algorithmus wurde durch das Publish-Subscribe Pattern realisiert. Der Algorithmus (`cursor_control_algorithm.py`) fungiert als Publisher und die Strategy-Klasse (`AlgorithmsStrategy` siehe 4.3) als Subscriber. Das Skript `event_listener.py` dient hierbei als Schnittstelle zwischen dem Algorithmus und der Strategy-Klasse. Es stellt die Funktionen `subscribe()` und `post_event()` zur Verfügung.

Mit der `subscribe()`-Funktion erstellt der Subscriber, also die Strategy-Klasse, einen Event-Typ und ordnet diesem eine Funktion zu, die ausgeführt werden soll, wenn das erstellte Event eintritt. In diesem Modell werden die Events "move_left_direction" und "move_right_direction" erstellt und ihnen die jeweiligen Move-Methoden des Players zugeordnet. Das obige Diagramm ist vereinfacht und stellt das Strategy Pattern nicht mit dar. Deswegen ist der Player in dem Diagramm der Subscriber.

Mit der `post_event()`-Funktion übergibt der Algorithmus dem "event_listener" den Eventtyp und der "event_listener" ruft dann die jeweilige zugeordnete move-Methode auf. Hierbei ist zu beachten, dass in Python der Subscriber nicht, wie in anderen Programmiersprachen, benachrichtigt wird, sondern die Methode des Subscribers einfach aufgerufen wird, wenn ein Event gepostet wird.



4.8 DATENEXTRAKTION

Die Datenextraktion ist für das Puffern und Speichern der Daten und Trials verantwortlich. Sie puffert die rohen Daten, welche von der Datenerfassung kommen, in einer Liste.

Sobald die Datenextraktion einen Befehl zum Markieren eines Trials, mit dem Label, Start- und Endzeit, bekommt, berechnet sie den Startindex des Trials in der Datenliste und die Länge des Trials in Anzahl der Samples. Die Datenextraktion speichert diese Informationen zusammen mit dem Label in Listen ab. Des Weiteren wird sich die Anzahl der Trials und die Anzahl der unterschiedlichen Labels gemerkt.

Wird nun eine Session gespeichert, konvertiert die Datenextraktion alle Daten und Listen mit Informationen zu den Trials in NumPy-Arrays und speichert diese, gemeinsam mit den Metadaten, in einem npz-File ab. Die npz-File sind folgendermaßen benannt: „session-<subject-id>-<ddmmYYYY>-<HHMMSS>.npz“.

4.9 DATENVISUALISIERUNG

Die Funktion des Live-Plots in dem Konfigurationsfenster für den oder die Wissenschaftler:in wird in dem Skript `liveplot_matlab.py` implementiert. Wichtig ist, dass zu Anfang des Skriptes das Backend (“TkAgg”) gesetzt wird, damit es schon beim Erstellen aufgerufen wird. Dieses sorgt für ein stabiles, geräteunabhängiges Verhalten beim Anzeigen der Plots. Auch am Anfang muss das selbst erstellte Theme (“`liveplot_light.mplstyle`”) gesetzt werden, da Themes in Matplotlib nicht zur Laufzeit geändert werden können.

Um die Figure für die Plots in diesem Skript zu speichern, wird zuerst die Methode `start_live_plot()` im Controller des Konfigurationsfensters einmalig beim Starten der Session, mit einer dort erstellen Figure als Übergabeparameter, aufgerufen.

Der Vorteil dieses Aufbaus ist die Möglichkeit, die zu plottenden Queues zur Laufzeit hinzufügen zu können. Dazu kann die Methode `connect_queue(queue: queue.Queue, plot_label, row: int, color: str, name: str, column: int, position: int, y_labels:list = None)` aufgerufen und folgende Parameter mitgeben werden:



queue	FIFO Queue, die Daten enthält, welche geplottet werden sollen
plot_label	Klassenname, PlotData Objekte mit demselben Klassennamen werden in einem Subplot gezeichnet
row	Anordnung des Subplots in der entsprechenden Zeile
column	Anordnung des Subplots in der entsprechenden Spalte
position	Position vom Subplot, hier wird von rechts nach links und von oben nach unten durchgezählt
color	Farbe der gezeichneten Linie
name	Name des Graphens
y_labels	Benutzerdefinierte y_labels

Zudem sollte beim Hinzufügen von Queues beachtet werden, dass diese nicht mehrfach im Live-Plot Skript gespeichert werden. Dazu kann die Methode `remove_all_plots()` vor allen `connect_queue()`-Aufrufen genutzt werden, um alle Queues aus dem Skript zu entfernen und die Figure zurückzusetzen. Mit dem nachfolgenden Aufruf der Methode `initial_draw()` kann die Anzeige dann aktualisiert werden. Das wird zum Beispiel gemacht, wenn eine neue Session gestartet wird.

Um die Plots zu aktualisieren, wird der Gameloop im Controller des Konfigurationsfensters genutzt, der periodisch die Methode `perform_live_plot()` aufruft. So kann sichergestellt werden, dass der Plot oder das Konfigurationsfenster in der Mainloop läuft.

Beim Aufruf der Methode `perform_live_plot()` wird für jede gespeicherte Queue überprüft, ob sich in dieser neuen Daten befinden. Sollte dies der Fall sein, werden alle Werte der Queue entnommen und an das Ende der in Arrays verwalteten y-Werte, durch shiften der alten Werte, gehängt. Die x-Werte werden um die Anzahl der neuen Werte, auch durch shiften, erweitert. Danach wird die Methode `live_plotter(plot_data: PlotData)` aufgerufen, welche den Graph anschließend neu zeichnet und skaliert.

Falls die Plots sich verändert haben, wird zuletzt die Methode `figure.canvas.draw()` aufgerufen, um die Canvas zu aktualisieren.



4.10 MODEL-VIEW-CONTROLLER PATTERN

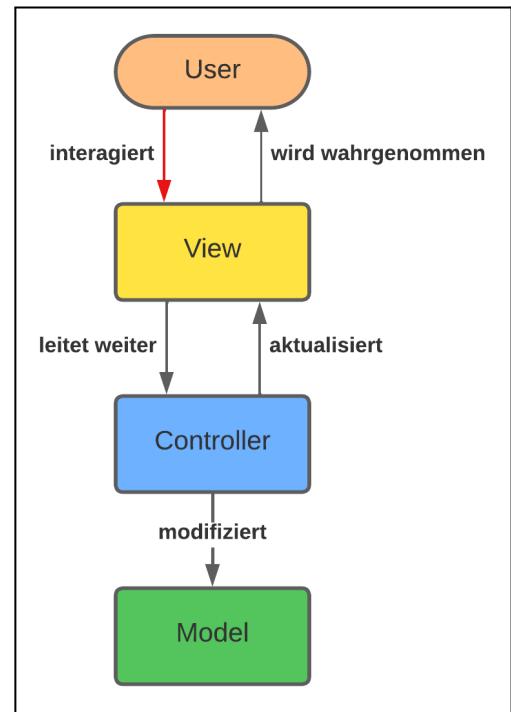
4.10.1 ALLGEMEINER MVC-PROZESS

Als Model-View-Controller Pattern, kurz **MVC**, bezeichnet man ein Architekturmuster, bei dem die Anwendung in drei eigenständige Module unterteilt wird: Model (Modell), View (Ansicht, Darstellung) und Controller (Steuerung).

Das MVC-Modell dient zur Speicherung bestimmter Daten, d. h. zur Speicherung von Teilen des aktuellen Zustands der Anwendung.

MVC-Views sind die grafischen Schnittstellen der MVC-Anwendung. Sie “visualisieren” Daten, die im Model der Anwendung enthalten sind.

Der MVC-Controller dient zur Steuerung der MVC-Anwendung.



In den meisten MVC Implementierungen interagieren das Model und die View direkt miteinander, indem das Model bei Änderungen die View aktualisiert. Meist wird dies durch ein Observer-Pattern realisiert. In unserem Fall wurde gezielt darauf geachtet, dass die Kommunikation zwischen dem Model und der View zwingend über den Controller erfolgt (Siehe Abbildung).

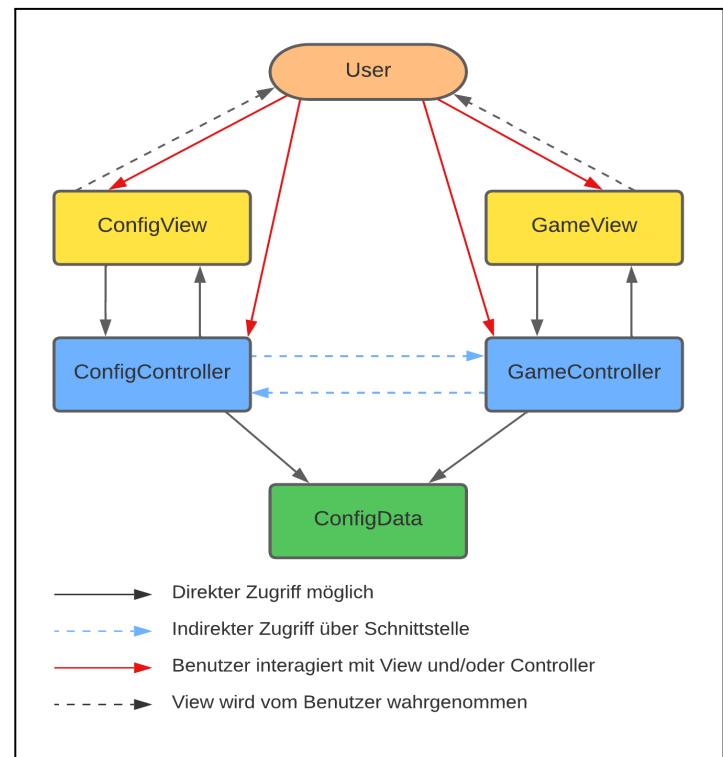
Der oder die Entwickler:in möchte Fehler gern an einer Stelle im Code finden und beheben, ohne dass andere Teile des Codes betroffen sind. Durch die lose Kopplung der einzelnen Module haben lokale Änderungen des Programms auch nur lokale Auswirkungen und es ist nicht erforderlich etwas an den anderen Modulen zu verändern, geschweige denn Kenntnisse über diese zu haben. Außerdem können die einzelnen Module leicht ausgetauscht werden. Hierdurch wird der Wunsch des Entwicklers bzw. der Entwicklerin nach **Änderbarkeit** erfüllt.



4.10.2 KOMMUNIKATION ZWISCHEN ZWEI MVC-KOMPONENTEN

In der rechten Abbildung ist die Kommunikation zwischen den beiden MVC-Komponenten dargestellt. Beide Komponenten besitzen eine Referenz auf dasselbe MVC-Modell, sodass nur eine Instanz der `ConfigData` Klasse existiert, welche den derzeitigen Zustand der Anwendung enthält.

Eine Kommunikation des Users ist sowohl mit den Views, als auch mit den beiden Controllern möglich (bspw. die Steuerung des Spiels durch die Tastatur oder das EEG-Headset).



Der gegenseitige Zugriff der beiden Controller erfolgt ausschließlich indirekt über die zur Verfügung gestellten Schnittstellen dessen Eigentümers. Der Eigentümer ist das Objekt, welches den Controller erzeugt hat und die Referenz auf diesen besitzt. (siehe: blau gestrichelte Linie in der Abbildung).

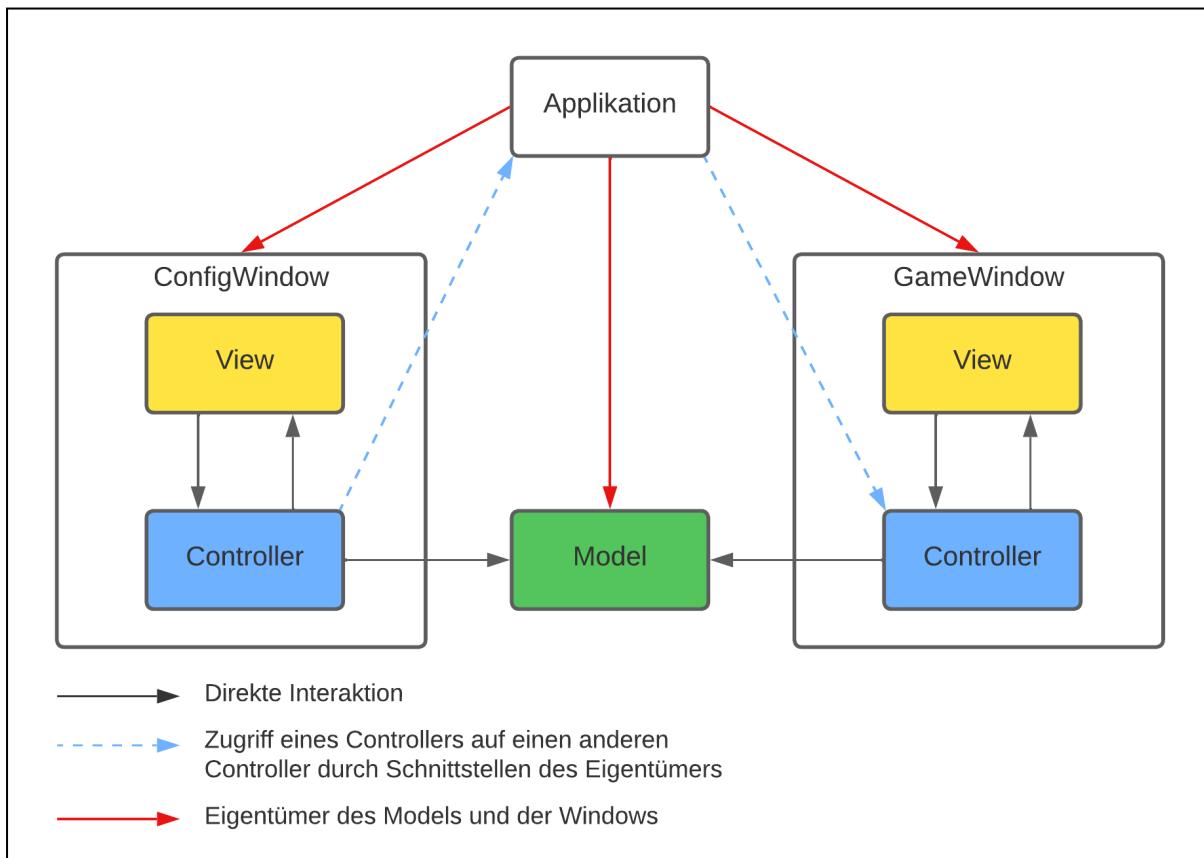
Beispielsweise kann von der `ConfigView` aus über einen Button die aktuelle Session gestoppt werden. Gleichzeitig soll das Spiel beendet und die erreichte Punktzahl auf der `GameView` angezeigt werden. Hierfür stellt der `GameController` eine Methode zur Verfügung, welche vom `ConfigController` aufgerufen werden muss.

Um dem MVC-Pattern treu zu bleiben und eine maximale Entkopplung zu gewährleisten, erfolgt dieser Methodenaufruf über die Schnittstellen der Eigentümer.

Im nächsten Abschnitt wird anhand des soeben beschriebenen Beispiels näher auf die Eigentümer der einzelnen MVC-Komponenten und das Architektur-Gesamtbild eingegangen.



4.10.3 GESAMTÜBERBLICK UND OWNERSHIP



Im Allgemeinen müssen die einzelnen MVC-Komponenten beim MVC-Pattern an beliebigen Zeitpunkt im Code initialisiert werden. Außerdem müssen sowohl die View, als auch das Modell nach ihrer Erstellung an den Controller gebunden werden.

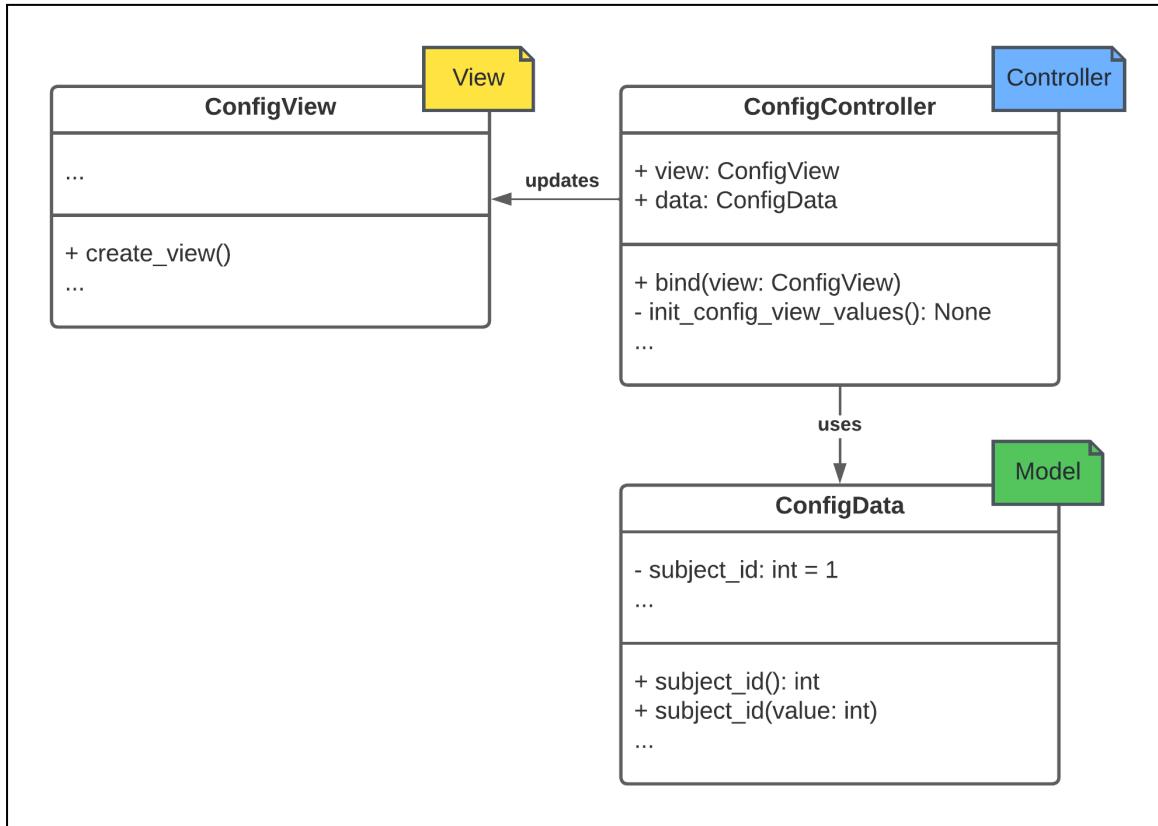
Die gesamte Konfiguration diesbezüglich befindet sich in diesem Projekt jeweils in einem separaten Fenster-Objekt (`ConfigWindow` und `GameWindow`). Da beide Fenster nicht immer angezeigt werden sollen, müssen diese dynamisch steuerbar sein. Dies erfolgt durch deren Eigentümer (Owner), dem Applikationsobjekt. In der obigen Abbildung ist zu sehen, dass Applikation der Eigentümer des Models und der Windows (Fenster) ist.

Möchte der Controller von `ConfigWindow` nun auf den Controller von `GameWindow` zugreifen, so sieht der Pfad folgendermaßen aus:

`ConfigController>ConfigWindow>Applikation>GameWindow>GameController`



4.10.4. MODUL-/KLASSEN-DIAGRAMM MIT MINIMALBEISPIELEN





4.10.5 MINIMALE CODE-AUSSCHNITTE

Minimalbeispiel eines Controllers:

Im nachfolgenden Code-Auszug ist einerseits die Definition der abstrakten Controller Klasse und deren konkrete Implementierung enthalten. Mit der abstrakten Methode `bind(view: View)` kann einer Controller-Instanz nach seiner Erstellung eine View Instanz zugewiesen werden.

```
class Controller(ABC):
    @abstractmethod
    def bind(self, view: View):
        raise NotImplementedError

class ConfigController(Controller):
    def __init__(self, master=None) -> None:
        self.master = master
        self.view = None

    def bind(self, view: View):
        self.view = view
        self.view.create_view()
```



Minimalbeispiel einer View:

Im unteren Code-Auszug ist einerseits die Definition der abstrakten View Klasse und deren konkrete Implementierung enthalten. Eine View erbt hierbei von `tk.Frame` und stellt somit ein Fenster dar. Mit der abstrakten Methode `create_view()` wird die View initial erstellt.

```
class View(tk.Frame):
    @abstractmethod
    def create_view(self):
        """Creates the view"""
        raise NotImplementedError

class ConfigView(View):
    def __init__(self, master):
        super().__init__(master)

    def create_view(self):
        control_frame = tk.Frame(master=self)
        control_frame.grid(row=0, column=0, sticky='nsew')
```



Minimalbeispiel eines Models:

Im nachfolgenden Code-Auszug ist ein Model inklusive einer privaten Variable, einem entsprechenden Getter, Setter und Defaultwert zu sehen. Außerdem erfolgt das Miteinbeziehen der Business-Logik im Setter der Variable.

```
class ConfigData(object):
    def __init__(self, subject_id: int = 1):
        self.__subject_id = subject_id

    @property
    def subject_id(self):
        return self.__subject_id

    @subject_id.setter
    def subject_id(self, value):
        """
        Validate the subject id
        :param int value: the new value for the subject id
        :raise ValueError: if the given value is incorrect
        :return: None
        """

        if int(value) > 0:
            self.__subject_id = value
        else:
            raise ValueError(f'Invalid subject id: {value}')
```



5. PROJEKTDOKUMENTATION

5.1 BESCHREIBUNG DES VORGEHENSMODELLS UND DER TEAMORGANISATION

5.1.1 GEWÄHLTES VORGEHENSMODELL

Bei der Wahl des Vorgehensmodells kam für uns im Team nur Scrum infrage, da unsere Aufgabenstellung stark wissenschaftlich geprägt ist und demzufolge das genaue Vorgehen zu Beginn und auch während der ersten Sprints nie ganz eindeutig war.

5.1.2 ROLLEN IM TEAM

	Rolle	Person
1	Product Owner & Entwickler	Firat Demir
2	Scrum Master & Entwickler	Marcel Roth
3	Entwickler	Christian Dittrich
4	Entwicklerin	Charlotte Wengler
5	Entwickler	Pascal Halmer
6	Entwicklerin	Larissa Willibald

5.1.3 ALLGEMEINE ZUGRUNDELIEGENDEN ENTSCHEIDUNGEN

Wir haben uns im Team bewusst für die Spezialisierung der Entwickler:innen auf gewisse Epics entschieden, um zu Beginn des Projektes den größtmöglichen Parallelitätsgrad zu erzielen.

Zur Planung unserer Sprints haben wir uns für die Kombination aus User Stories mit Tasks und somit gegen Subtasks entschieden. Da wir nur acht Wochen Zeit hatten, haben wir uns für zweiwöchige Sprints entschieden, um unsere Zeit mehr für die konkrete Realisierung anstatt für organisatorische Meetings zu nutzen.

Aufgrund der wissenschaftlich geprägten Themenstellung und den sich schnell ändernden Anforderungen, haben wir jeweils zur Mitte der Sprints ein Sprint Refinement durchgeführt. Dies hat uns stark dabei geholfen, dass alle



Teammitglieder mit genügend Aufgaben ausgelastet waren und gab uns die Gelegenheit spontane Kundenwünsche einplanen zu können.

Da uns somit insgesamt nur drei Sprints zur Verfügung standen und eine zeitliche Abschätzung von Tasks erst nach der Durchführung von circa drei Sprints sinnvoll ist, haben wir uns für die Komplexitätsabschätzung mit Story Points entschieden.

5.1.4 TEAMORGANISATION

Das gesamte Team hat sich aufgrund der wissenschaftlichen Themenstellung und der benötigten Hardware trotz Covid-19 darauf geeinigt, die wesentliche Arbeit des Projektes vor Ort (im Raum C24 an der THU) abzuhalten. Daraus ergaben sich direkt qualitativ hochwertigere Daily Scrums.

Ein zweiwöchiger Sprint enthielt bei uns die folgenden Scrum-Meetings:

- Daily Scrum Meeting
- Sprint Planning Meeting 1 (Festlegen der User Stories)
- Sprint Planning Meeting 2 (Ausarbeiten der User Stories)
- Sprint Refinement Meeting
- Sprint Review Meeting
- Sprint Retrospective Meeting

5.1.6 VERWENDETE SOFTWARE TOOLS



5.1.7 VERWENDETE HARDWARE

- Ultracortex Mark IV EEG Headset Pro-Assembled Medium 16-channel³
- Linux Notebook

³ OpenBCI Online Store (2022): All-in-One Biosensing R&D Bundle.



5.2 DOKUMENTATION PROJEKTMANAGEMENT

Wie bereits in Kapitel 5.1.3 beschrieben, haben wir uns zum Abschätzen der Aufgaben für Story Points entschieden. Im allerersten Program Increment war noch alles neu. Weder die einzelnen Spezialistenteams, noch das Team im Gesamten, haben jemals in exakt dieser Konstellation gearbeitet. Die Teams befanden sich in der “Storming Phase”. Das bedeutete wiederum, dass die Working Agreements, Definition of Done (DoD) und Definition of Ready (DoR) noch unklar waren. Wir haben uns für den Einsatz von DoD entschieden. Die DoD besteht aus vorab festgelegten Kriterien, die ein Produkt erfüllen muss, um als done – also fertig – angesehen zu werden.⁴ Außerdem war sowohl die Arbeitsumgebung, als auch die verwendete Programmiersprache Python neu und unbekannt. Somit ist jede Schätzung eine Kaffeesatzleserei.⁵ Hierbei haben wir uns auf die Empfehlung von Herrn Balser hin für SAFe (Scaled Agile Framework) entschieden. SAFe vermeidet das Schätzen anhand einer Referenz-User-Story und schlägt folgendes vor: **Relativ Schätzen**

Hierbei wählt man beim ersten Program Increment das kleinste Item aus dem Backlog und weist diesem eine ‘1’ zu. “Dem nächstgrößeren Item weist man eine ‘2’ zu, und hangelt sich unter Nutzung einer an die Fibonacci-Zahlen angelehnten Reihe (1, 2, 3, 5, 8, 13, 20, 40, 100) vorwärts”⁶.

Dann stellte sich noch die Frage, wie sich die Velocity über normalisierte Story Points berechnet. Also, mit wie vielen Story Points wir den ersten Sprint starten. Aufgrund der zweiwöchigen Sprintdauer besitzt ein Sprint 10 Arbeitstage. Multiplizieren wir diese Arbeitstage mit der Anzahl der Teammitglieder, so kommen wir auf:

$$\text{Base Capacity} = 10 * \text{Teammitglieder} = 60 \text{ SP.}$$

Abwesenheits- und Feiertage werden ebenfalls mit der Anzahl der Teammitglieder multipliziert und von der Base Capacity subtrahiert. Im ersten Sprint war dies aber nicht der Fall. Hiervon ziehen wir dann noch mal 25 Prozent ab, denn die Planung auf 100 Prozent Auslastung ist nicht realistisch. Außerdem haben sowohl der Scrum Master, als auch der Product Owner nur teilweise mitentwickelt. Somit erhalten wir

⁴ Salimi, Sohrab: Definition of Done.

⁵ Agile Academy (2022): Normalisierte Story Points in SAFe – wie und warum?

⁶ Vgl. ebd.



letzten Endes:

Initiale Velocity = 60 SP * 0.75 = 45 SP.

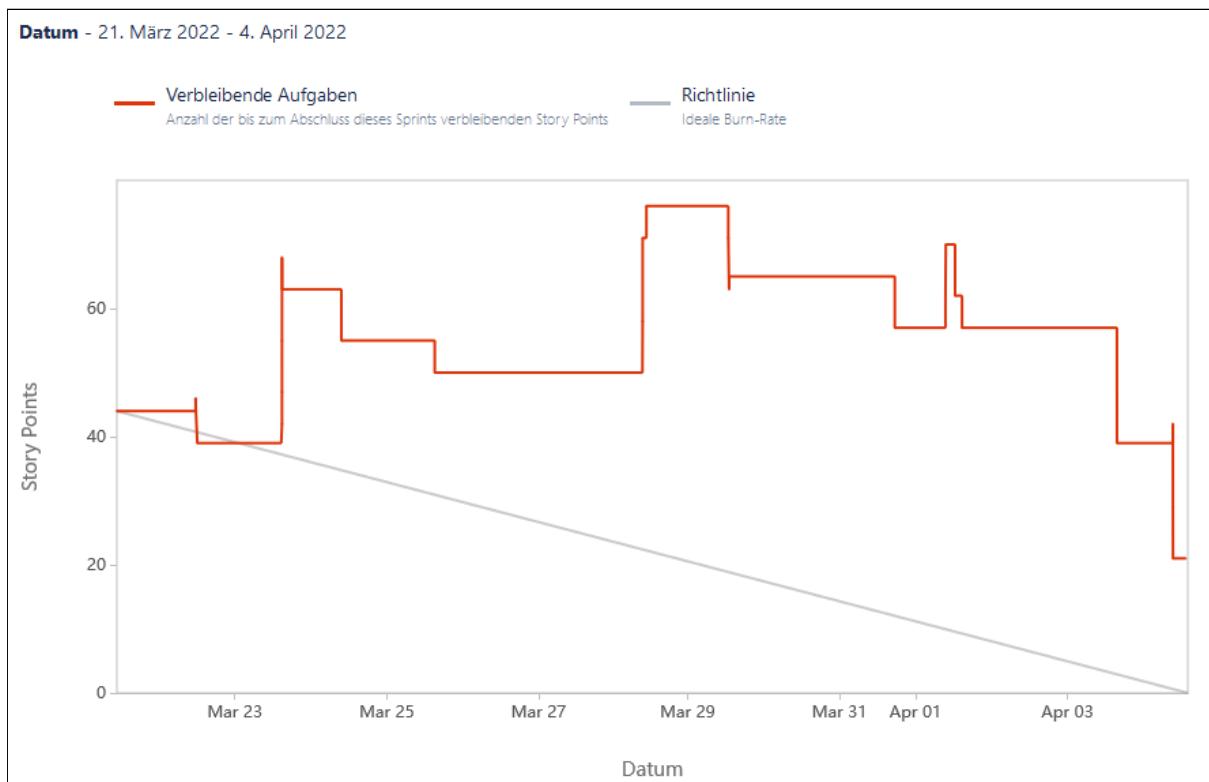
Somit wurde die erste Iteration mit 45 SP geplant. Im Folgenden sind die Burndown-Charts der einzelnen Sprints aufgeführt.



5.2.1 BURNDOWN-CHARTS

Sprint 1

Im Burndown-Chart des ersten Sprints ist zu sehen, dass, wie in Kapitel 5.2 beschrieben, mit einer Schätzung von circa 45 SP geplant wurde. Da das zugrundeliegende Thema enorm viele Unbekannte mit sich brachte, gab es im Laufe des Sprints mehrere abrupte Anstiege der Kurve. An diesen Tagen entstanden neue Items, deren Erforderlichkeit dem Team jedoch erst durch das Einarbeiten in die jeweilige Thematik bewusst wurde. Diese wurden dann direkt mit in den Sprint einbezogen und manipulieren dementsprechend das Endergebnis. Die größten Unbekannten waren mit Sicherheit die neue Programmiersprache Python und der neurowissenschaftliche bzw. physikalische Kontext der Themenstellung. Jeweils zur Mitte des Sprints wurde ein Sprint Refinement durchgeführt, um agil Anpassungen am Sprint vornehmen zu können. Dies geht aus der Kurve ebenfalls durch einen Anstieg der Story Points einher. Letzten Endes wurden nicht alle Sprint-Items erfolgreich umgesetzt. Das liegt unter anderem an den Items, welche durch agile Anpassung hinzugekommen sind.





Sprint 2

Im zweiten Sprint haben wir die zugrundeliegende Velocity durch die Erkenntnisse des ersten Sprints auf knapp 60 SP angehoben. Aus dem Burndown-Chart geht hervor, dass sich unser Team bis zur Mitte des Sprints nur seltenst oberhalb der Richtlinie befand. In der zweiten Hälfte des Sprints wurden kaum Items abgeschlossen, da das Team zu diesem Zeitpunkt an größeren Storys mit teilweise 13 Storypoints arbeitete und diese erst kurz nach Sprintende fertiggestellt wurden. Alles in allem verlief die Schätzung im zweiten Sprint jedoch einwandfrei.





Sprint 3

Der letzte Sprint erstreckte sich bis kurz vor die endgültige Präsentation und ging somit etwas länger als zwei Wochen. Da dieser Sprint außerdem alle Aufgaben bezüglich der Dokumentation enthielt, und diese teilweise sehr agil hinzukamen, lagen die Story Points während des gesamten Sprints über der Richtlinie. Auch fielen in den letzten Sprint einige Bug Behebungen, welche natürlich nicht vorhersehbar waren.





5.2.2 VELOCITY-CHARTS

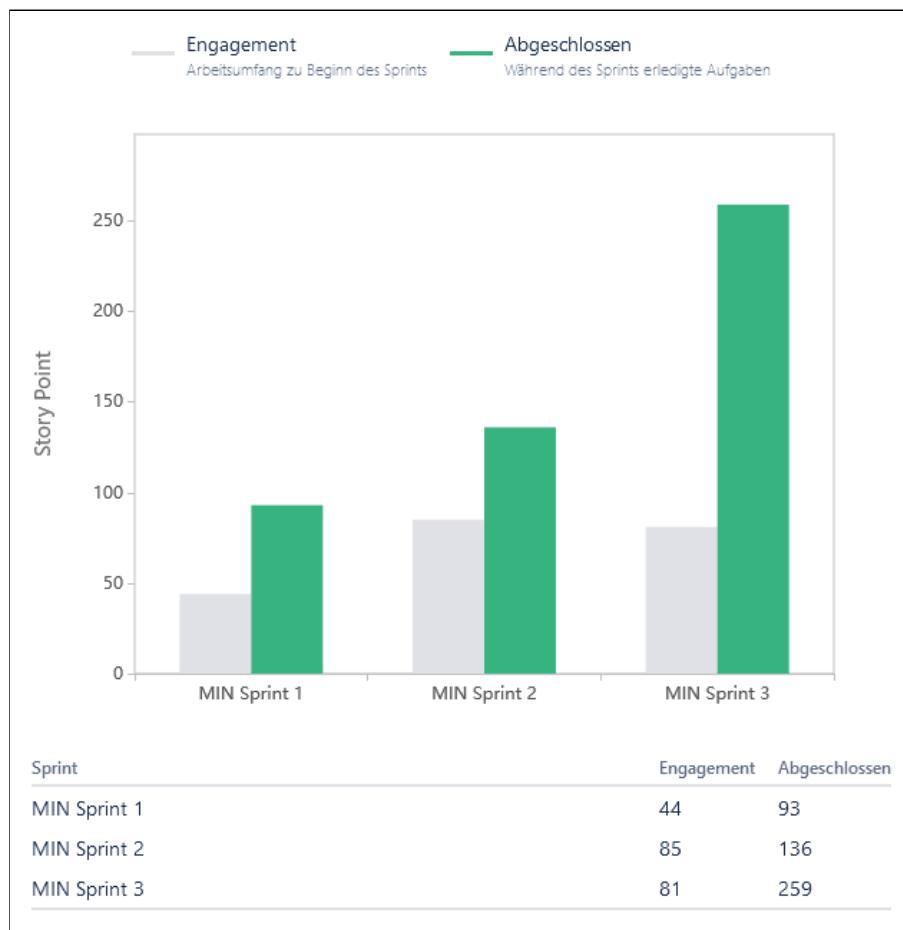
Setzt man die geplanten Story Points zum Beginn eines Sprints mit den tatsächlich abgeschlossenen Story Points ins Verhältnis, so kann eine Aussage darüber getroffen werden, wie gut die Aufgabenplanung im jeweiligen Sprint umgesetzt wurde.

Verhältnis Sprint 1 = $44 / 93 = 47\%$

Verhältnis Sprint 2 = $85 / 136 = 62,5\%$

Verhältnis Sprint 3 = $81 / 259 = 31\%$

Das Verhältnis gibt Aufschluss darüber, wie viele der nach Sprintende erledigten Aufgaben zum Sprintbeginn auch eingeplant waren. Vergleicht man den ersten mit dem zweiten Sprint, so fällt direkt auf, dass sich die Planung stark verbessert hat. Der dritte Sprint besitzt im Vergleich zu den anderen Sprints ein katastrophales Verhältnis, da hier die gesamte Dokumentation und diverse Bug Behebungen mitwirkten und wir diese nicht bereits zu Beginn des Sprints exakt abschätzen konnten.





5.2.2 BACKLOGS

Sprint 1

Vorgangstyp	Vorgangs schlüssel	Zusammenfassung
Story	MIN-3	Kontinuierliches Auslesen der EEG-Daten in Python
Story	MIN-4	Steuerung des Spiels: Pfeiltasten
Story	MIN-5	Prototypischer Algorithmus Teil 1
Story	MIN-6	Prototypischer Algorithmus Teil 2
Task	MIN-9	Auslesen der Daten: Vorarbeiten
Task	MIN-10	Auslesen der Daten: Implementierung
Task	MIN-11	Auslesen der Daten: Konfiguration und Dokumentation
Task	MIN-12	Prototyp für das Spiel: Recherche GUI mit Tkinter in Python
Task	MIN-13	Prototyp für das Spiel: Implementierung
Task	MIN-14	Prototypischer Algorithmus Teil 1: Einlesen der BCIC Testdaten
Task	MIN-15	Prototypischer Algorithmus Teil 1: Implementierung des Spatial Filtering
Story	MIN-24	Projektdokumentation: Grundstruktur
Story	MIN-27	Erfassung von EEG-Kalibrierungsdaten: Front-End
Task	MIN-28	Prototypischer Algorithmus Teil 2: FFT und Multitaper
Task	MIN-29	Prototypischer Algorithmus Teil 2: Alpha Band Integration
Task	MIN-30	Prototypischer Algorithmus Teil 2: Richtungsableitung
Task	MIN-33	Auslesen der Daten: Implementierung ohne GUI und Verzögerung
Task	MIN-34	Architektur mit Schnittstellen visualisieren
Task	MIN-36	Einbinden des Pong Spiels in die GUI
Task	MIN-38	Verbesserung prototypischer Algorithmus: Variabler Sliding Window Offset (ML-BCI)
Task	MIN-39	Verbesserung prototypischer Algorithmus: Live Plot der Daten (ML-BCI)
Story	MIN-40	Verbesserung prototypischer Algorithmus



Sprint 2

Vorgangstyp	Vorgangs schlüssel	Zusammenfassung
Story	MIN-7	Speichern der Trials
Task	MIN-36	Einbinden des Pong Spiels in die GUI
Task	MIN-38	Verbesserung prototypischer Algorithmus: Variabler Sliding Window Offset (ML-BCI)
Task	MIN-39	Verbesserung prototypischer Algorithmus: Live Plot der Daten (ML-BCI)
Story	MIN-40	Verbesserung prototypischer Algorithmus
Task	MIN-42	Implementierung des Sliding Windows ohne GUI und Delay
Story	MIN-44	Definition der Schnittstellen der Komponenten
Task	MIN-46	Schnittstellen: Erstellen eines Komponenten Diagramms
Story	MIN-47	Überarbeitung des EEG-Daten Einlesens
Task	MIN-48	Implementierung einer Schnittstelle zum Anfragen von rohen EEG-Daten
Task	MIN-49	Steuerung mit Strategy Pattern
Task	MIN-50	Portierung des ML-BCI Frameworks auf andere Geräte
Task	MIN-51	Integration des Skriptes zum vereinfachten Laden der EEG-Daten von Prof. Strahnen
Task	MIN-53	Erstellen einer Code-Guideline in Confluence
Task	MIN-54	Code Refactoring nach Code-Guideline (auf Master-Branch)
Task	MIN-55	Definieren von Metadaten, Trialgröße usw.
Task	MIN-57	Recherche: Performanter Array Zugriff in Python (Row-Major oder Column-Major)
Task	MIN-59	Schnittstellen: Erstellen eines übersichtlichen Diagramms
Task	MIN-62	Schnittstellen: Erweiterung des Komponentendiagramms um Data Extraction und Data Loader
Task	MIN-63	Speichern der Trials in einem Buffer
Task	MIN-64	Metadaten-Klasse erstellen
Task	MIN-65	Speichern der Session
Task	MIN-66	Analyse der Ergebnisse des Algorithmus
Story	MIN-69	Einstellung der Konfigurationsparameter
Task	MIN-70	Anpassung der Data Extraction nach Kundenwünschen
Task	MIN-71	Anpassung der Data Acquisition nach Kundenwünschen



Story	MIN-72	Anpassung des Pong Spiels
Task	MIN-73	Erstellen der Spielerklasse
Task	MIN-74	Erstellen der Zielklasse
Task	MIN-76	Erstellen der Settings Ansicht
Task	MIN-77	Initiales MVC Konstrukt
Task	MIN-78	Sicheres Laden von Channels
Task	MIN-80	Kommentierung des Spielcodes
Task	MIN-81	Endscreen und Score des Spiels
Task	MIN-82	Modernes Styling

Sprint 3

Vorgangstyp	Vorgangsschlüssel	Zusammenfassung
Story	MIN-8	Laden der Trials in das ML-BCI Framework
Story	MIN-43	Erstellen der Trials
Task	MIN-56	Extrahieren der Trials: Allgemeine Implementierung
Task	MIN-58	Erweiterung des Komponentendiagramms um die Trial Extraction Komponente
Story	MIN-72	Anpassung des Pong Spiels
Task	MIN-75	Speichern der Session über die GUI
Task	MIN-81	Endscreen und Score des Spiels
Story	MIN-83	Ein- und Ausblenden des Plots
Bug	MIN-84	Game Strategy in der Config einstellbar
Task	MIN-85	Verbinden des Trialhandlings mit dem Spiel
Story	MIN-86	Optimierung des Algorithmus
Task	MIN-87	Verschieben aller Konstanten in das Config Model Objekt
Task	MIN-88	Erweiterung um Kommentarfeld
Task	MIN-89	Umstrukturierung der Config View
Story	MIN-90	Steuerung der Trial-Aufnahme über die GUI
Task	MIN-91	Steuerung der Trial-Aufnahme: Back-End
Task	MIN-92	Steuerung der Trial-Aufnahme: Front-End
Task	MIN-93	Integration des neuen Spiels in die MVC GUI
Task	MIN-94	Evaluation Matplotlib Live-Plot



Task	MIN-95	Einstellen der Konstanten auf bestimmte Person für gutes Endergebnis
Task	MIN-96	Evaluierung verschiedener Handbewegungen
Task	MIN-97	Integration des Algorithmus in den Master
Task	MIN-98	Implementierung des Publishers für das Strategy Pattern (Algorithmus)
Task	MIN-99	Starten von ReadData beim Start des Spiels
Task	MIN-100	Poster Entwurf skizzieren
Task	MIN-101	Diagramm zum MVC Pattern
Task	MIN-102	Diagramm und Beschreibung zum Strategy Pattern in Python
Task	MIN-103	Diagramm und Beschreibung zum Publish-Subscribe Pattern in Python
Task	MIN-104	Use-Case Diagramm zu den User Stories
Task	MIN-105	Ergebnis aus Nutzersicht: Testperson
Task	MIN-106	Projektübergreifende Retrospektive
Task	MIN-107	Dokumentation zum Kapitel Projektmanagement
Task	MIN-108	Screenshots für das Poster aufnehmen
Task	MIN-109	Poster realisieren
Task	MIN-110	Automatische Latenzeinstellung unter Linux
Task	MIN-111	Implementierung des Live-Plots mittels Matplotlib
Bug	MIN-112	Player überlappt mit dem Target
Task	MIN-113	Meta Daten im Trialhandler erfassen
Task	MIN-114	Variable Größe des Targets und Spielers
Bug	MIN-116	Steuerung des Spiels wieder ermöglichen
Task	MIN-117	Löten der Kabel am openBCI Headset
Task	MIN-118	Deaktivieren der Eingabefelder nach Starten des Spiels
Bug	MIN-119	Fehler bei nicht verbunden Headset
Task	MIN-122	Ausblick Cursor Control Algorithmus
Task	MIN-123	Präsentation erstellen
Task	MIN-124	Stoppen des Threads zum Daten Lesen
Task	MIN-126	Refactoring (Algorithmus-Team)
Task	MIN-128	Erstellen eines Sequenzdiagramms
Task	MIN-129	Print Methode für die Metadaten
Task	MIN-130	Kalibrierung mit Progressbar



Task	MIN-131	Integration vom Live-Plot in die GUI
Task	MIN-132	Live-Plot bei Stop Session stoppen und ausblenden
Task	MIN-133	Toggles anpassen
Task	MIN-134	Starten des Game Windows auf einem auswählbaren Monitor
Bug	MIN-136	Board beenden bei beendeten Thread
Task	MIN-137	Kalibrierung Trial erfassen
Task	MIN-138	Optimierung des Live-Plots
Task	MIN-139	Starten/Stoppen des Spiels durch Leertaste ausbauen
Bug	MIN-140	Konstanter Dark Mode beim Spiel
Task	MIN-141	Connect-Button mit Funktionalität erstellen
Task	MIN-142	Stoppen der Session während der Kalibrierung ermöglichen
Task	MIN-143	Fotos vom Probanden mit Headset für Präsentation und Poster
Task	MIN-144	Minimal Code-Ausschnitt zu View, Model und Controller erstellen (MVC)
Task	MIN-145	Ergebnis aus Nutzersicht: Wissenschaftler:in
Task	MIN-148	Laden einer aufgezeichneten Session
Task	MIN-150	Dark Mode ausbauen
Task	MIN-151	User Storys anpassen
Bug	MIN-152	Exception bei handle samples
Bug	MIN-153	Hit Abstand zwischen Player und Target
Bug	MIN-154	Discard Session bei Trial Aufnahme
Bug	MIN-155	Board-Session beenden beim Beenden des Session
Bug	MIN-158	Session Replay: Das Spielfenster friert beim Abbrechen ein
Task	MIN-159	Beschreibung zum MVC Diagramm
Bug	MIN-160	Startzeitpunkt des Kalibrierungs Trials ist negativ
Bug	MIN-161	Gamewindow Name ist "Calibration"
Bug	MIN-162	Daten werden bei Discard nicht gecleared
Bug	MIN-163	Player übertritt die Wall Border
Bug	MIN-172	Channel Belegung in Metadaten erfassen



6. BEITRÄGE DER EINZELNEN TEAMMITGLIEDER

Detaillierte Beschreibung der Beiträge der einzelnen Teammitglieder zum Projektergebnis

- Marcel Roth

- Koordinierung der Kommunikation innerhalb des Entwicklerteams als Scrum Master
- Sichert gestellt, dass jedes Teammitglied zur Fertigstellung des Softwareproduktes gleichermaßen wichtig und involviert war
- Erstellen der Branching und Commit Guideline
- Erstellen des Technischen Konzepts
- Erstellen und Erweitern der gesamten grafischen Oberfläche nach Kundenwünschen unter Verwendung des MVC-Pattern in Python
- Erstellen des Use-Case-, Komponenten- und MVC-Diagramms und der dazugehörigen Dokumentation
- Entwicklung des anfänglichen Pong-Spiels inklusive Gameloop etc.

- Firat Demir

- Organisatorische Aufgaben und Kommunikation mit dem Kunden als Product Owner
- Entwicklung der GUI für die Erfassung von EEG-Kalibrierungsdaten
- Erstellen und Pflege der Dokumentation (Confluence)
- Erstellen der Code Guideline
- Durchsetzen der Code Guideline durch Code Refactorings
- Erstellen einer Skizze des Posters
- Erstellen des Posters und der Präsentation
- Terminplanung mit dem Kunden und Herrn Balser
- Pflege des Sprint, Product und Impediment Backlogs



- Wesentlicher Beitrag zur endgültigen Dokumentation
- Charlotte Wengler
 - Entwicklung der Datenerfassung
 - Entwicklung des Speicherns von Sessions
 - Refactoring des Pong Spiels zu einem einfacheren Spiel
 - Entwicklung des Strategy-Pattern für die Steuerung des Spielers
 - Erstellung des Strategy-Pattern Diagramm
 - Verbindung des Algorithmus mit dem Steuern des Spiels mithilfe eines Publish-Subscribe Patterns und Erstellung des entsprechenden Diagramms
 - Entwicklung des Anstoßens der Trial-Erfassung durch das Spiel
 - Wesentlicher Beitrag zur endgültigen Dokumentation
- Pascal Halmer
 - Entwicklung der Datenerfassung
 - Entwicklung des Speicherns der Session
 - Refactoring des Pong Spiels zu einem einfacheren Spiel
 - Erweiterung des Spiels um einen Endscreen und um das Anzeigen der Zeit, die der Spieler benötigt, um das Ziel im Spiel zu erreichen
 - Entwicklung des ML-BCI Loaders, um erzeugte Session-Daten in das ML-BCI Framework laden zu können
 - Erstellen des Use-Case-, Komponenten-, Sequenzdiagramm



- Larissa Willibald & Christian Dittrich
 - Wissenschaftlerin/Proband
 - Entwicklung der Datenanalyse (Cursor Control Algorithmus) und dessen Dokumentation
 - Implementierung der einzelnen Verarbeitungsschritte
 - Recherche zu FFT und anderen Spektralanalyse Methoden
 - Recherche zu verschiedenen Frequenzbändern
 - Recherche zur genaueren Implementierung des Algorithmus
 - Recherche zum Aufbau des Gehirns zum optimalen Platzieren des EEG-Headsets
 - Evaluation von verschiedenen Einstellungsmöglichkeiten
 - Schreiben eines Testskriptes zum Laden und Evaluieren von gegebenen BCIC-Testdaten
 - Live-Plot mittels PyQt5 Framework
 - Live-Plot mittels matplotlib für Tkinter GUI und dessen Dokumentation
 - Verbinden des Algorithmus mit dem Einlesen der Daten
 - Dokumentation von Ausblick und Erkenntnisse in Einlesen von EEG Daten



7. DISKUSSION DES ERGEBNISSES

7.1 WELCHE NOTWENDIGEN ZIELE WURDEN ERREICHT?

- ✓ Einlesen der EEG-Daten
- ✓ Interpretieren der EEG-Daten
- ✓ Steuerung des Spiels durch Echtzeit-EEG-Daten
- ✓ Spielerische Erfassung der EEG-Trials
- ✓ Extrahieren der Trials
- ✓ Live-Plot der EEG-Daten

7.2 WELCHE OPTIONALEN ZIELE WURDEN ERREICHT?

- ✓ Vergleich verschiedener Methoden zur Erhebung der EEG-Daten
- ✓ Laden der Trials in das ML-BCI Framework
- ✓ Wiedergabe von bereits aufgenommenen Sessions

7.4 RETROSPEKTIVE

Die Retrospektiven der einzelnen Sprints wurden jeweils zum Ende der jeweiligen Sprints mit dem gesamten Team und einer Dauer von einer Stunde abgehalten. Des Weiteren wurde am Ende des Projektes eine abschließende Projektübergreifende Retrospektive gehalten.

7.4.1 PROJEKTÜBERGREIFENDE RETROSPEKTIVE

Was lief gut?

Durch das Arbeiten direkt vor Ort an der Hochschule ergab sich ein gutes Teamwork und eine schnelle Problemlösung, da man bei einem Problem schnell einen Ansprechpartner:in im Team gefunden hat und auch unseren Experten schneller hinzugezogen werden konnte. Zudem wurde, dadurch, dass alle Teammitglieder sich



am selben Ort befanden, der Teamgeist gestärkt, weil man sich besser kennenlernen konnte, als beispielsweise verglichen beim Arbeiten von zuhause aus.

Wegen des forschungsbezogenen Themas haben sich einzelne Teammitglieder auf gewisse Arbeitsschwerpunkte spezialisiert. Somit wurde konzentriertes Wissen erreicht und paralleles Arbeiten ermöglicht.

Positiv anzumerken ist auch, dass aus dem streng umgesetzten Git-Workflow eine gute Merge Request Verteilung auf das Team resultierte. Hierbei sorgten auch mehrere erstellte Guidelines für mehr Übersichtlichkeit. Die erstellten Merge Request wurden zügig gereviewt, meist innerhalb eines Tages. Der allgemeine gute Workflow im Team ergab sich auch durch die gute Umsetzung des Vorgehensmodell Scrum.

Der Overhead für das Team wurde reduziert, weil die Sprint-Reviews auf Wunsch des Kunden schlicht gehalten wurden. Die erzielten Fortschritte wurden direkt am entstehenden Produkt präsentiert, anstatt eine PowerPoint-Präsentation vorzuführen.

Der letzte positive Punkt ist, dass das Team das gesamte Projekt über durch einen Berater bei den einzelnen Scrum Artefakten unterstützt wurde. Dies war hilfreich, da die Umsetzung nicht immer klar war, weil dies für die meisten eins der ersten Projekte dieser Art war.

Was lief schlecht?

Obwohl die Spezialisierung der einzelnen Teammitglieder, wie oben erläutert, mehrere positive Effekte mit sich brachte und alle sich einig waren, dass dies notwendig war, haben sie doch auch zu einer ungleichen Arbeitsverteilung geführt. Wenn Probleme auftraten, konnten nicht beteiligte Teammitglieder manchmal nicht helfen, da ihnen spezielles Wissen fehlte. Ebenso konnten zu diesem Themenbereich erstellte Tasks auch nur von den entsprechend spezialisierten Mitgliedern bearbeitet werden. Dies hatte unter anderem zur Folge, dass das Zweierteam, welches an der Datenanalyse arbeitete, einen sehr großen Arbeitsumfang hatte. Jedoch konnten andere Entwickler:innen es nicht unterstützen, weil hierfür sehr viel Fachwissen notwendig war, welches sich das Zweierteam im Laufe des Projekts angeeignet hatte.



Problematisch war auch, dass bestimmte Risikofaktoren nicht ausreichend evaluiert wurden und daraus ein großer Zeitverlust resultierte. So wurde anfangs an einem Pong-Spiel gearbeitet, bis man nach einem Sprint feststellte, dass dies nicht zur Trialerfassung geeignet ist, weil aus dem Spiel schwer zu erkennen ist, welches Label erzielt werden soll. Außerdem wurde nicht evaluiert, welche Bibliotheken zur Erstellung des Live-Plots und der GUI miteinander kompatibel sind. So wurde beides mit verschiedenen Bibliotheken parallel realisiert und erst nach zwei Sprints festgestellt, dass diese nicht kompatibel sind. Eine frühzeitige Evaluierung der verschiedenen Möglichkeiten hätte in beiden Fällen Zeit gespart.

Was würden wir beim nächsten Mal anders machen?

Wie vorher schon ausgeführt, wurde die Evaluation verschiedener Risikofaktoren in manchen Punkten versäumt. Dies würden wir bei einem neuen Projekt nicht mehr vernachlässigen.

In Bezug auf die Organisation wäre es besser gewesen, wenn wir fachliche Problem mit einzelnen Personen besprochen hätten, anstatt im gesamten Team, beispielsweise im Daily Scrum. So hätten Mitglieder, die nicht zur Problemlösung beitragen konnten, in dieser Zeit andere Dinge bearbeiten können.

Was haben wir gelernt?

Gerade das Team, das für die Datenanalyse zuständig war, hat sich viel Wissen im Bereich Neurowissenschaften und Physik erarbeitet.

Alle Teammitglieder haben sich durch das Projekt die Programmiersprache Python angeeignet oder ihre Kenntnisse dort erweitert. Für die meisten von uns war es auch das erste Mal eine umfangreiche Software zu erstellen, dadurch haben wir viele neue Kenntnisse und Erfahrungen gewinnen können. Besonders hervorzuheben ist hier die Anwendung verschiedener Software-Patterns und wann man diese einsetzt.

Für mehrere Teammitglieder war auch der Umgang mit Git und Scrum in diesem Umfang neu. Das ganze Team ist jetzt in der Lage einen Branch zu erstellen, einen Merge Request zu stellen und einen Branch in einen anderen zu mergen. Wir konnten uns auch gut mit der Vorgehensweise zur Bearbeitung von Tasks vertraut



machen (Bearbeitung, Review und anschließend Mergen). Auch fühlen wir uns jetzt sicherer in Bezug auf den verschiedenen Artefakten, Rollen und Meetings bei Scrum. Doch vermutlich eine der wichtigsten Lektionen war für uns von Problemen oder erstellter Software zu trennen. Gerade durch das fehlende Evaluieren mussten sich Teammitglieder von ihrer erstellten Software, in die viel Zeit geflossen ist, trennen und akzeptieren, dass diese im fertigen Produkt nicht enthalten sein wird, wie zum Beispiel das Pong-Spiel. Auch mussten wir lernen, von einem Problem abzulassen, wenn wir nicht weiter kamen oder den Stand auf dem sich die Software war zu akzeptieren. Hervorzuheben ist, dass nicht, wie lange angenommen, die Umsetzung des Cursor Control Algorithmus die Ursache des Problems war, weswegen wir rechte und linke Bewegungen nicht unterscheiden konnten. Vielmehr lag es an der exakten Platzierung des Headsets auf dem Motor Cortex, sodass wir letztendlich das Spiel äußerst zuverlässig steuern konnten.

7.4.2 RETROSPEKTIVE NACH DEN EINZELNEN SPRINTS

Keep: (Beibehalten)

1. Sprintlänge ziemlich angenehm (Erst recht durch Refinement in der Sprint Mitte) - Sprint 1
2. User Stories und Tasks werden teilweise sehr kurzfristig und agil in den Sprint aufgenommen - Sprint 1
3. Arbeitszeiten und zeitliche Flexibilität im Team - Sprint 1
4. Aktive Nutzung des Impediment Backlogs durch das Team - Sprint 1
5. Sofern möglich Arbeit von Zuhause → Produktiver - Sprint 2
6. Refinement in der Mitte des Sprints ziemlich gut - Sprint 2
7. Code Guideline und diese auch durchgesetzt - Sprint 2
8. Daily Länge gut (15 min; manchmal länger, falls Arbeit ausging; Zeit gut genutzt) - Sprint 2
9. Aufteilung Dokumentation und Programmierung gegen Ende hin → Starke Entlastung - Sprint 3



10. Hervorragender Git Workflow. Gute Verteilung der Merge Reviews und einige Merge Requests - Sprint 3

11. Gutes Verknüpfen der Arbeit von Zuhause und vor Ort innerhalb des Teams - Sprint 3

Add: (Verbessern)

1. Bei Arbeitslosigkeit ggf. schon Tickets beginnen und fertig spezifizieren, welche Teil der späteren Meilensteine sind - Sprint 1
2. Confluence übersichtlicher strukturieren - Sprint 1
3. Code Guidelines einführen und im gesamten Team umsetzen - Sprint 1
4. Mehr Tasks auf To-do bzw. im Backlog in Reserve - Sprint 2
5. Tasks unabhängiger voneinander (falls möglich) - Sprint 2
6. Ausführlicheres Sprint Planning - Sprint 3
7. Festgesetzte Sprint Ziele - Sprint 3

Less: (Weniger machen)

1. Kürzere Dailys (zu detailliert und technisch) - Sprint 1
2. Kunde nicht in Sprint Planning einbeziehen, sondern weiteres Vorgehen als Team aus dem Review Gespräch ableiten - Sprint 2
3. Spontane Feature-Wünsche nicht auf eigene Hand in den Sprint nehmen, sondern nur beim Refinement/Sprint Planning - Sprint 3

More: (Mehr machen)

1. Intensivere Nutzung und Pflege des Impediment Backlogs - Sprint 1
2. Bei Problemen und sonstigem mehr Kontakt im Team - Sprint 1
3. Mehr Branches bzw. Tasks feiner unterteilen, um mehr Parallelität zu gewährleisten - Sprint 2



4. Mehr Evaluierung vor größeren Arbeiten/Umsetzungen (Pyplotlib mit tkinter) - Sprint 3
5. Arbeit gleichmäßiger verteilen auf alle Teammitglieder - Sprint 3
6. Auf dem End- bzw. Zielsystem öfter testen - Sprint 3
7. Mehr mit dem OpenBCI Headset testen - Sprint 3
8. Kürzere Dailys und technische Aspekte in kleineren Gruppen besprechen - Sprint 3

7.5 IMPEDIMENT BACKLOG

In unserem Impediment Backlog haben wir alle Punkte erfasst, welche ein reibungsloses Arbeiten behindert und somit das Erreichen der einzelnen Ziele in den einzelnen Sprints verzögert haben. Dazu zählen alle Probleme und Hindernisse, die während des gesamten Projekts aufgetreten sind. Die jeweiligen Probleme wurden stets von den einzelnen Teammitgliedern sofort im Impediment Backlog dokumentiert und bei Lösungsvorschlägen aktualisiert.

Datum	Frage / Problemstellung	Antwort / Lösung
16.03.2022	Sliding Windows: Welche Größe (zeitlich) müssen diese haben? Welcher Offset zwischen den Windows?	Länge des Sliding Windows sollte mindestens eine Welle der Frequenz enthalten, bei 1 Hz Welle müsste das Sliding Window also bspw. mindestens 1 Sekunde groß sein. → Guter Startwert 1 Sekunde Sliding Windows sollten sich weitestgehend überlappen. Parameter müssen erst herausgearbeitet werden. → Mit Offset bei 100ms starten
17.03.2022	Warum kann keine Verbindung zum Cyton Board hergestellt werden?	Liegt an dem Anbau. Spezieller Stecker muss herausgezogen werden, um die Pegel auf Null zu setzen.
18.03.2022	Was sollen wir alles grafisch darstellen? Spiel, EEG Ströme?	Visualisierung muss abschaltbar sein! TK (Toolkit) für die GUI verwenden Auf jeden Fall zwei verschiedene Fenster, damit es auf den anderen Monitor geschoben werden kann. Score nach dem Verlieren anzeigen und automatisch neu starten. Während des Spiels Score ausblenden.



18.03.2022	Welche Pins gehören zu den Channels?	Pins können anhand der Kabelleitungen abgelesen werden. Dokument der PIN Zuteilung erfolgt per E-Mail durch Professor Strahnen.
21.03.2022	Haben Labornotebook mit den BCIC-Testdaten und dem ML-BCI Framework noch nicht erhalten und können daher nicht weiterarbeiten.	Laptop am 22.03.2022 erhalten.
21.03.2022	Der Ringpuffer überträgt die Daten erst, wenn er 60 Einträge hat. Dadurch entsteht ein Delay von ca. 0,5 sec.	Lösungsansatz: Streaming der Open Source Bibliothek LSL, anstatt dem von Chunks mit Brainflow.
21.03.2022	Entscheidung: 1. Kontinuierliche lineare Bewegung des Pong-Paddle nach einmaligem setzen der Richtung durch einen Tastendruck 2. Einmalige Bewegung des Pong-Paddle um gewissen Wert X pro Tastendruck	Zunächst wurde sich im Prototypen des Spiels für die Implementierung der Variante 1 entschieden.
21.03.2022	Matplotlib Bibliothek Installation und Pip Package Update funktionieren nicht	Pip erst lokal updaten und dann im PyCharm Projekt. Danach funktioniert die Installation von Matplotlib.
24.03.2022	Unter MacOS wird die GUI in PyCharm immer als schwarzes Fenster angezeigt	Problem nur mit Python Version 3.10.4 behoben, jedoch fordert Kunde Version 3.8. Bei der Version 3.8 besteht der Fehler jedoch weiterhin unter MacOS. Code wird unter MacOS mit Version 3.10.4 geschrieben, aber kann auch für Version 3.8 genutzt werden.
28.03.2022	Der erste Sprint war anfangs nicht exakt abschätzbar und es konnten zu Beginn noch nicht alle benötigten Tasks erstellt und abgeschätzt werden, da sich diese teilweise erst im Laufe des ersten Sprints ergeben haben. Aus diesem Grund ist die Sprint-Burndown Chart von Sprint 1 nicht linear und enthält Peaks.	
28.03.2022	Bei Versuchen die OpenBCI-Stream Bibliothek zu verwenden, wird der Zugriff auf den USB-Port verweigert.	
30.03.2022	Bei Versuchen mit dem LSL-Stream zu arbeiten, wurde festgestellt, dass immer bei der Datenübertragung ein Delay von ca. 0.5 s auftritt, genau so wie wenn wir ohne GUI und mit der Brainflow-Bibliothek arbeiten. Daher vermuten wir, dass dies an der Hardware liegt.	Das Delay entsteht durch die default-Einstellung des Serial Ports des verwendeten PCs. Meistens ist dort eine Latenz von 16 ms eingestellt, stellt man sie auf 1ms, gibt es einen kontinuierlichen Datenstrom.



06.04.2022	PyQt5 ließ sich nicht importieren, da dll beschädigt sei.	- Neuinstallation von Python 3.8 - Setzen aller Path-Variablen von Python und PyCharm - Anlegen eines neuen lokalen Repositories unter Pycharm Projects
08.04.2022	Zum Erstellen von sinnvollen Trials ist eine gewisse Mindestzeitdauer notwendig, in welcher klar vorgegeben ist, in welche Richtung sich das Paddle gerade bewegt. Um diese Information dann mit den Trials abzuspeichern. Im Pong Spiel ist der Startzeitpunkt der Trial Aufnahme bzw. die vorgegebene Richtung leider nicht immer vorherzusagen. Das liegt unter anderem daran, dass sich der Ball bewegt. Aus diesem Grund ist das Pong Spiel ungeeignet für die Trial Aufnahme. Diese Problematik hatten wir zu Beginn leider nicht bedacht. Demzufolge ist es erforderlich, die Spielidee zu adaptieren und einfacher zu gestalten, sodass vom Spiel zu jedem Zeitpunkt die Zielrichtung bekannt ist.	Wechsel zu einem einfacheren Spiel, wo der Spieler sich nur rechts und links bewegen kann. Das Ziel des Spiels ist, ein Objekt/Ziel in einer bestimmten Zeit zu erreichen. Erreicht er das Ziel nicht in der vorgegebenen Zeit, erscheint ein neues Ziel.
09.04.2022	Korrelation der EEG-Daten zwischen Vorstellung und echter Ausführung der rechten und linken Handbewegung (stärkerer Ausschlag, wo befinden sich die Ausschläge, etc.)	
09.04.2022	Welche Filterung von EEG-Daten ist am optimalsten (zur Herausfilterung von Artefakten)	
10.04.2022	Errechnete Labels scheinen verschoben zu sein!!!	
14.04.2022 -19.04.2022	Die Dokumentation und Kommunikation der Projektarbeit auf Jira und Confluence war über 5 Tage lang nicht erreichbar, weil die Seiten down und die Bearbeitung des Projektes deshalb nicht möglich war.	Am 19.04.2022 waren beide Seiten wieder erreichbar
21.04.2022	FuncAnimation wurde auf verschiedenen Rechnern nur einmal ausgeführt und hat teilweise nicht einmal geplottet	Liegt daran, dass je nach Rechner als matplot backend ein Anderes als Standard zugewiesen sein kann. Das richtige Backend ist TkAgg. Der Befehl zum Setzen lautet <code>matplotlib.use('TkAgg')</code>
21.04.2022	Aufnahme von EEG Werten stellt sich als schwierig und inkonsistent heraus	Erkenntnisse: Proband sollte unter keinen Medikamenteneinfluss, auch keinen Einfluss von Kopfschmerztabletten



		stehen, da sonst die EEG Werte schwach ausfallen können Proband sollte auf einer bequemen Sitzgelegenheit, für uns hat ein Sofa gut funktioniert, sitzen in einer ruhigen und für ihn entspannten Umgebung langhaarige Probanden sind eher weniger für die Trockenelektroden geeignet <u>Vorsicht:</u> Headset kann sich über die Zeit verstellen (schon ab 10 min) lange Sessions sind nicht empfohlen, da das Headset nach einer Zeit als sehr schmerhaft empfunden wird
22.04.2022	Soll die fertige Software als .exe vorliegen oder als .py scripte	Alle Skripte ohne Git Verlinkung müssen auf dem Linux Rechner vorliegen.
24.04.2022	<ul style="list-style-type: none"> • Kabel der Channels sind zu dünn und brechen nach öfterem Einstellen einfach ab (trotz der hohen Vorsicht beim Drehen) • Vorrat an Steckern ist leer 	
24.04.2022	Kabel von den Elektroden brechen vorne sehr schnell ab	Versuch die Kabel wieder hin zu löten

7.6 FAZIT DES PROJEKTES

Abschließend ist zum Projekt zu sagen, dass wir alle sehr viel aus dem Projekt mitnehmen können und an uns gewachsen sind.

Alle Teammitglieder waren mit dem organisatorischen Ablauf und dem Arbeitsklima innerhalb der Gruppe sehr zufrieden. Dass in einem Softwareprojekt Probleme und Fehleinschätzungen geschehen, ist völlig normal. Jedoch haben wir daraus viel gelernt und sind in der Lage, diese in weiteren Softwareprojekten zu vermeiden.

Des Weiteren war das Aneignen der komplett neuen Programmiersprache Python sehr spannend und für die Zukunft sicherlich hilfreich.

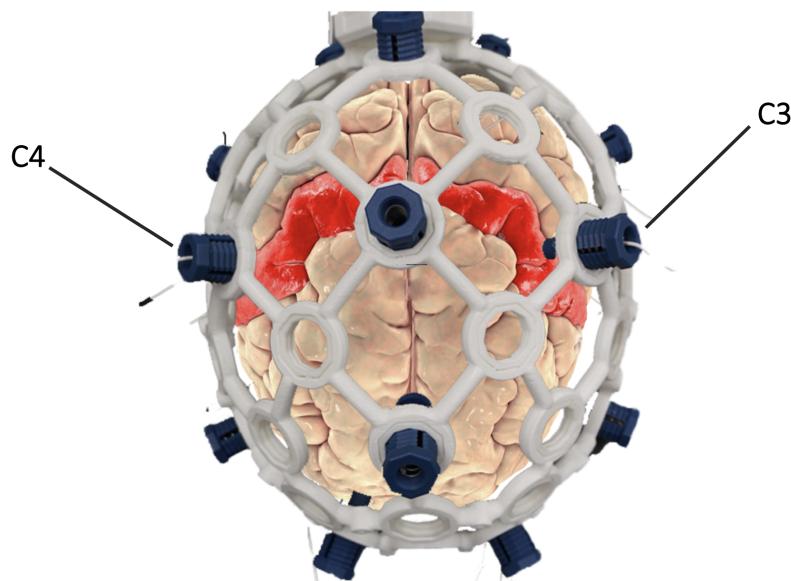
Wir fanden alle das Thema sehr interessant, jedoch war es etwas überdimensioniert und knapp bemessen für acht Wochen. Daraus folgte, dass teilweise auch an Wochenenden gearbeitet werden musste, um zum Projektabschluss fertig zu werden.



8. AUFNAHME VON EEG-DATEN

In diesem Kapitel werden zuletzt noch die Erkenntnisse notiert, die während des Projektes beim Entwickeln und Testen bezüglich der Aufnahme und Analyse der EEG-Daten gewonnen wurden.

Einer der wichtigsten Punkte ist das richtige Platzieren des EEG-Headsets auf dem Kopf der Testperson. Die Elektroden C3 und C4 müssen direkt auf dem Motor Cortex liegen, sodass der Cursor Control Algorithmus richtige Ergebnisse liefern kann. In der folgenden Abbildung ist der Motor Cortex (roter Bereich auf dem Gehirn) und das Headset mit den Elektroden C3 und C4 zu sehen:



78

Erkenntnisse aus neurowissenschaftlicher Sicht:

- Der Sensorimotor Cortex (verantwortlich für Sensorik) liegt direkt hinter dem Motor Cortex und kann daher für Artefakte sorgen⁹
- Da der Motor Cortex bei jeder Testperson unterschiedlich ausgeprägt ist, sollte dieser bei der Testperson eine ausreichend große Oberfläche haben, um diesen zu treffen

⁷ <https://www.alamy.de/fotos-bilder/motorischer-kortex.html>

⁸ https://docs.openbci.com/assets/images/MarkIV_electrode_map-aa945ecc6171e61a19bd9dc5176a25c8.png

⁹ Räumliche und zeitliche Veränderungen kortikaler elektrischer Aktivität bei spontan induzierter Handmotorik



Erkenntnisse beim Aufzeichnen einer Session:

- Die einzelnen benutzten Channels sollten über OpenBCI-GUI richtig eingestellt werden, sodass alle einen guten Kontakt haben
- Die Testperson sollte gesund sein und keine Medikamente nehmen, da insbesondere der Einfluss durch Kopfschmerztabletten im EEG sichtbar sein kann und die Ausschläge drastisch reduziert werden können.
- EEG Signale verhalten sich umso ruhiger, je weniger Körperspannung die Testperson hat. Daher wird zu einem bequemen Stuhl oder Sofa an einem ruhigen und am besten vertrauten Ort geraten. Die Arme können zusätzlich noch mit Kissen unterstützt werden und wenn die Sitzgelegenheit zu hoch ist, sollte die Testperson die Füße hochlegen.¹⁰
- Da das Headset vor allem nach längerer Tragezeit als unangenehm empfunden wird, sollte die Testperson dieses empfohlener weise nicht länger als eine Stunde aufbehalten.
- Von Sessions an mehreren aufeinanderfolgenden Tagen mit der gleichen Testperson ist zudem auch aufgrund des Tragekomforts abzuraten. Kopfschmerzen stellen dann nämlich schneller ein.
- Da sich die Haare der Testperson beim Einstellen der einzelnen Elektroden (und vor allem bei den Abstandshaltern) verwickeln, sind kurze bzw. keine oder wenig Haare empfehlenswert.
- Ob die nötige Konzentration und die Ruhe der Testperson vorhanden ist, ist schnell anhand des Graphens (vor allem C3_{pow} / C4_{pow}) ersichtlich, Aspekte wie eine ausreichende Oberfläche des Motor Cortex sind hingegen schwer zu validieren.

Erkenntnisse beim Einstellen des Cursor Control Algorithmus:

- Frequenzbänder mu, alpha und beta sind vielversprechend. Eine Kombination von mu/alpha und beta sind daher auch denkbar.¹¹

¹⁰ Classifying EEG signals preceding right hand, left hand, tongue, and right foot movements and motor
imageries

¹¹ Negative covariation between task-related responses in alpha/beta-band activity and BOLD in
human sensorimotor cortex: An EEG and fMRI study of motor imagery and movements



- Multitaper erzielte bei unserer Testperson die besten Resultate. Burg schien zu sensibel zu sein, da zu viele willkürliche Ausschläge sichtbar waren.
- Eine Gewichtung der integrierten PSD Werte von C3 und C4 war, unter Berücksichtigung der oben genannten Punkten zur Aufzeichnung einer Session und bei korrekter Positionierung nicht mehr notwendig.
- Hinweise auf parallele sensorische Wahrnehmung im C3 und C4 Bereich wurden gefunden, aber in ersten Tests, indem die Testperson an den Händen berührt wurde nicht bestätigt
- Bei physisch ausgeführten Bewegungen muss darauf geachtet werden, dass die Elektroden C3 und C4 mittig auf dem Motor Cortex und nicht zu nah am Sensorimotor Cortex platziert werden. Dies führt zu schwer differenzierbaren Ausschlägen der $C3_{pow}$ und $C4_{pow}$ Kurven. Dieses Problem sollte jedoch nicht bei imaginären Bewegungen auftreten.
- Imaginäre Bewegungen führten zu viel schwächeren Ausschlägen der $C3_{pow}$ / $C4_{pow}$ Kurven und konnten nicht zuverlässig erkannt werden. Der Verdacht lag hierbei, dass dies an den Trockenelektroden des BCI Headsets lag, da in der Forschung bei derartigen Versuchen nach unseren Recherchen Nasselektroden verwendet wurden.
- Bessere Ergebnisse erreichten wir mit einem small Laplacian. Bei dem large Laplacian war die Platzierung noch schwieriger, da schnell der Hörbereich im Gehirn sich dominant in den Ausschlägen widerspiegelte. Das liegt daran, dass die äußeren rechten und linken Elektroden auf der Kopfseite der Testperson sich auf dem Bereich des Hörens befinden. Dies hat dann bei Geräuschen das Ergebnis stark verfälscht.
- Für physische Bewegungen hat das Handheben im Gegensatz zum Greifen deutlichere und differenzierbare Ausschläge produziert.
- Über imaginäre Bewegungen können wir leider keine zuverlässige Aussage treffen. Bewegungen wurden nicht immer erkannt und wir konnten die Trockenelektroden s.o. nicht als Grund ausschließen. Dennoch hat es der Testperson geholfen, zuvor einen Schaumstoffball zu drücken und sich diese Bewegung dann vorzustellen. Wir vermuten, das liegt daran, dass die



Testperson den Ball besser greifen und stärker zudrücken kann, als wenn sie nur die Hand schließt.



9. AUSBLICK

Mit der in dem Projekt entstandenen Software zur spielerischen Erfassung von Trials, wurde eine solide Basis für weitere Forschungen im Bereich KI an der THU geschaffen. Im Stil von Open Source wurde geschaut, dass die einzelnen Komponenten, insbesondere der Cursor Control Algorithmus, einfach erweiterbar und ausführlich kommentiert sind. Zudem wird eine umfassende Dokumentation bereitgestellt. Dies soll insbesondere weitere Forschungen und Optimierungen, durch ein besseres Verständnis mit der vorhandenen Software und der zugehörigen Dokumentation, im Bereich der KI unterstützen.

Ziel dieser Software wird es dann im späteren Verlauf sein, qualitativ hochwertigen Trials zum Trainieren einer KI bereitzustellen, wodurch diese eine hohe Trefferwahrscheinlichkeit bei der Erkennung von Links- und Rechtsbewegungen erlernen wird. Die Höhe der Trefferwahrscheinlichkeit im Gegensatz zum alten Verfahren zum Erfassen von Trials muss jedoch noch evaluiert werden.

Sollte die KI, und davon gehen wir aus, eine sehr hohe Trefferwahrscheinlichkeit aufweisen, kann diese dann bei querschnittsgelähmten Menschen eingesetzt werden, um Prothesen und Rollstühle präzise mit Gedanken zu steuern.



10. QUELLEN

1. Stieger, James R.; Engel, Stephen A.; He, Bin (2021): Continuous sensorimotor rhythm based brain computer interface learning in a large population; In Sci Data 8 (1); p. 98. Online verfügbar unter: <https://www.nature.com/articles/s41597-021-00883-1#further-reading>
2. Conversion of EEG activity into cursor movement by a brain-computer interface (BCI) - PubMed. Online verfügbar unter: <https://pubmed.ncbi.nlm.nih.gov/15473195/>
3. OpenBCI Online Store (2022): All-in-One Biosensing R&D Bundle. Online verfügbar unter: <https://shop.openbci.com/products/all-in-one-biosensing-r-d-bundle?variant=13043151994952>
4. Salimi, Sohrab: Definition of Done. Online verfügbar unter: <https://www.agile-academy.com/de/agiles-lexikon/definition-of-done/>
5. Agile Academy (2022): Normalisierte Story Points in SAFe – wie und warum?. Online verfügbar unter: <https://www.agile-academy.com/de/skalierung/normalisierte-story-points-wie-und-warum/>
6. Motorischer Kortex Stockfotos und -bilder Kaufen - Alamy (2022). Online verfügbar unter: <https://www.alamy.de/fotos-bilder/motorischer-kortex.html>
7. Ultracortex Mark IV | OpenBCI Documentation (2022). Online verfügbar unter: <https://docs.openbci.com/AddOns/Headwear/MarkIV>
8. Knauth, Astrid (2020): Räumliche und zeitliche Veränderungen kortikaler elektrischer Aktivität bei spontan induzierter Handmotorik. With assistance of Helmut Laufs, Paul Christian Baier. Online verfügbar unter: https://macau.uni-kiel.de/receive/macau_mods_00000700?lang=de
9. Morash, Valerie; Bai, Ou; Furlani, Stephen; Lin, Peter; Hallett, Mark (2008): Classifying EEG signals preceding right hand, left hand, tongue, and right foot movements and motor imageries. In Clinical neurophysiology : official journal of the International Federation of Clinical Neurophysiology 119 (11), pp. 2570–2578. Online verfügbar unter: <https://www.sciencedirect.com/science/article/pii/S138824570800922X>
10. Yuan, Han; Liu, Tao; Szarkowski, Rebecca; Rios, Cristina; Ashe, James; He, Bin (2010): Negative covariation between task-related responses in alpha/beta-band activity and BOLD in human sensorimotor cortex: an EEG and fMRI study of motor imagery and movements. In Neurolimage 49 (3), pp. 2596–2606. Online verfügbar unter: https://www.sciencedirect.com/science/article/pii/S1053811909010994?dgcid=api_sd_search-api-endpoint