EEGsuperMap

Ein Instrument auf Audiorate

Anselm Weber

August 2024

Prof. Dr. Marc Bangert
Michele Samarotto
Institut für Musikinformatik und Musikwissenschaft
Hochschule für Musik Karlsruhe

0.1 Einleitung

Digitale Musikinstrumente (DMIs) bestehen aus einer Kombination von Software- und Hardware-Komponenten. Damit ein solches Instrument als Echtzeitsystem funktioniert, ist nicht nur die Funktionsweise der einzelnen Komponenten entscheidend, sondern auch, wie diese untereinander kommunizieren.

In dieser Arbeit wird EEGsuperMap als Beispiel für die Umsetzung eines solchen Systems vorgestellt. EEG-Elektroden erfassen Rohdaten, die über eine grafische Benutzeroberfläche (GUI) in Echtzeit auf Klangparameter abgebildet und somit sonifiziert werden. Dies führt zu einer Verkettung verschiedener Prozesse, darunter das Audio-Processing und der User-Input zur Laufzeit mittels GUI. Das ursprünglich erfasste EEG-Signal wird dabei sonifiziert und durchläuft von Elektrode bis zu Lautsprecher verschiedene Zustände und Übertragungsprotokolle.

Das Instrument operiert daher in einem interdisziplinären Bereich, der Elektroenzephalographie, Daten-Sonifikation und Musikinformatik umfasst. Zu Beginn werden grundlegende Funktionsweisen und Konzepte erläutert, die für die Funktionsweise des DMIs von Bedeutung sind. Anschließend werden schrittweise die einzelnen Komponenten von EEGsuperMap und deren zugrunde liegende Software erklärt, um den gesamten Signalfluss des Systems nachvollziehbar darzustellen.

Chapter 1

Theorie und Hintegrund

1.1 EEG-Daten

Wenn Regionen im Gehirn aktiv sind, feuern die darin befindlichen Neuronen über ihre Dendriten, wobei "Feuern" allgemein die Kommunikation zwischen den Gehirnzellen bezeichnet. Ionen, also geladene Teilchen, werden mithilfe des Membrantransport-Proteins durch die Membran der Neuronen befördert. Somit werden diese elektrisch geladen beziehungsweise polarisiert. Damit bestimmte physische Zustände wie Entspannen oder Schlafen aufrechterhalten werden können, müssen die Neuronen Ionen mit dem Extrazellularraum austauschen. Gelangen diese Ionen zu einem benachbarten Ion der gleichen Ladung, stoßen sie sich ab. Wenn viele Ionen aus vielen Neuronen herausgedrückt werden, bewegen diese ihre Nachbarionen und diese wiederum ihre Nachbarn. Dadurch entsteht eine Welle von Ionen, und ein Ionenleiter im Gehirn entsteht. Bei der Messung werden also elektrische Felder erfasst, die durch die neuronale Aktivität und Kommunikation in spezifischen Hirnregionen erzeugt werden.

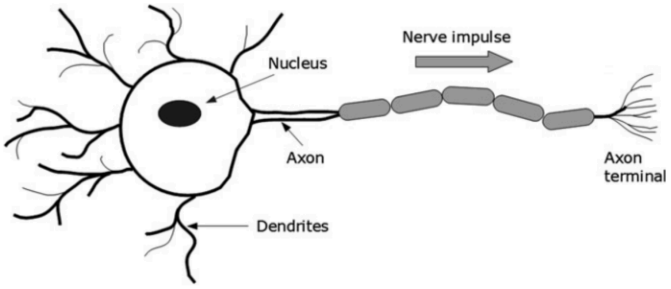

Figure 1.1: Aufbau einer Nervenzelle (Zaitcev et al. 2015a, s. 128)

Mittels EEG-Elektrode wird der Ionenleiter im Gehirn in einen Elektronenleiter innerhalb des EEG-Kabels umgewandelt. Ein elektrisches Signal entsteht (Zaitcev et al. 2015b, s. 128). Um jedoch die Hirnaktivität innerhalb dieses elektrischen Signals zuverlässig identifizieren zu können, müssen noch folgende Aspekte beachtet werden.

1.1.1 Signalstärke

Zunächst ist das elektrische Potential eines einzelnen Neurons, das individuell agiert, viel zu schwach, um es mit einer auf dem Schädel befindlichen Elektrode zu messen. Hinzu kommt, dass die elektrischen Felder bei unterschiedlicher räumlicher Ausrichtung der Hirnzellen, sich gegenseitig aufheben können. Messbare Hirnaktivität beruht also auf der Summe der Signale synchron agierender Neuronen mit gleicher räumlicher Orientierung innerhalb des Gehirns. Ein Beispiel für eine derartige Anordnung findet sich bei den Pyramidenzellen:

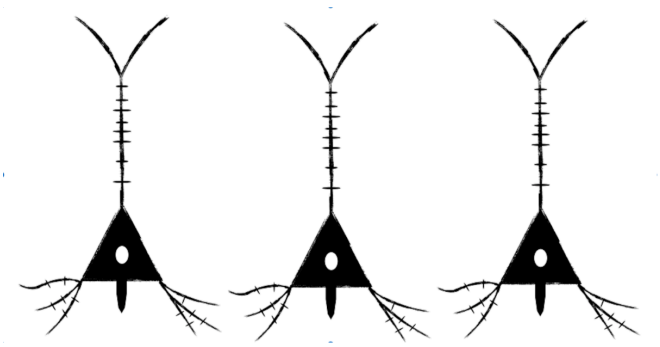

Figure 1.2: Schematische Darstellung von Pyramidenzellen

Diese Neuronen sind im Cortex aufgereiht. Sie feuern zusammen und produzieren somit das deutlich messbares EEG-Signal. Ihnen wird eine entscheidende Rolle in der Verarbeitung räumlicher Informationen sowie der Gedächtnisbildung zugeschrieben. Die Pyramidenneuronen stehen mit rund zehntausend anderen Zellen über ihre Dendriten in Verbindung (Stangl 2021). Hinzu kommt, dass sie sich relativ nah an der Schädeldecke, also weiter außen im Gehirn, befinden. Die Positionierung ist hierbei durchaus von Bedeutung, da elektrische Felder relativ zur Entfernung des jeweiligen Messgerätes exponentiell abnehmen. Daraus folgt im Umkehrschluss, dass Aktivität tief im Inneren des Gehirns und somit weiter entfernt von der Elektrode, kaum mehr bis gar nicht messbar ist. (vgl. Bera 2015, s. 61).

1.1.2 Messung der Signale

Die Messung eines EEG-Signals erfolgt immer in Abhängigkeit von einer Referenz. Ein Aufbau wäre, ein EEG-Signal zwischen mindestens zwei Elektroden zu messen, wobei eine davon als Referenzelektrode dient. Diese sollte an einer relativ inaktiven Zone angebracht werden, damit Ausschläge der jeweiligen Mess-Elektrode im Vergleich sichtbar werden können. Gewöhnlich wird dazu, wie in den Beispielen von Abb.3 und Abb.4, das Ohrläppchen eines oder beider Ohren gewählt. Alternativ kann auch der Durchschnitt aller sich auf dem Schädel befindlichen Elektroden als Referenzwert genutzt werden.

Unabhängig von der gewählten Methode wird immer der Unterschied vom Nutzsignal zum Referenzsignal gemessen.

1.1.3 Störfaktoren

EEG-Signale sind aufgrund der Schwäche des Nutzsignals verrauscht. Ein naheliegender Störfaktor sind andere elektrische Felder in unmittelbarer Nähe der Elektroden. Denn durch feuernde Neuronen erzeugte elektrische Felder sind nicht die einzigen elektrischen Potenziale, die zur Elektrode gelangen. Messbare Signale befinden sich im Mikrovolt (µV)-Bereich und sind daher anfällig gegenüber einer Reihe von Störfaktoren wie Aktivitäten des Körpers oder anderen elektrischen Feldern im Raum. All das beeinflusst das Signal und erschwert die Suche nach messbarer Hirnaktivität. Analog zur Audiotechnik spricht man daher von einer hohen Signal-to-Noise-Ratio im Bezug auf EEG-Signale (Fouad et al. 2015, s. 15). Nichtsdestotrotz gibt es viele Herangehensweisen, um dieses Problem so gut es geht in den Griff zu bekommen. Um ein Signal von etwaigen Störungen zu unterscheiden, ist es hilfreich zu wissen, wonach überhaupt gesucht wird. Deshalb konzentriert sich der folgende Abschnitt auf Muster der Hirnaktivität zu bestimmten Zuständen des Gehirns.

1.1.4 Rhythmische Aktivität

Rhythmische oder oszillierende EEG-Aktivität wird allgemein auch als "Hirnwellen“ bezeichnet. Um bestimmte Zustände wie Schlaf oder Entspannung aufrechtzuerhalten, erzeugen Millionen von Neuronen messbare Aktivität in regelmäßigen Abständen. Diese manifestiert sich in Form von periodischen Oszillationen im EEG-Signal und lässt sich als Anhäufungen in bestimmten Frequenzbändern beobachten (vgl. Deore/Mehrotra, 2015, S. 345). Gängigerweise unterscheidet man unter folgenden Frequenzbändern:

- Delta-Wellen befinden sich im Bereich von 0.1 bis 5 Hertz. Diese tendieren zur größten Amplitude sowie niedrigster Frequenz. Messbar sind sie bei Säuglingen und bei Erwachsenen während des Tiefschlafs.
- Theta-Wellen treten bei 8 bis 13 Hertz auf. Sie werden mit "Ineffizienz“ und Tagträumen in Verbindung gebracht. Die langsamen, also niederfre-

quenten Theta-Wellen repräsentieren den Übergang zwischen Schlaf und Wachzustand.

- Alpha-Wellen befinden sich im Bereich von 8 bis 13 Hertz und sind am deutlichsten in der dominanten Gehirnhälfte. Sie sind messbar während Entspannungsphasen. Tatsächlich führt bereits ein Schließen der Augen zu mehr Alpha Bandstärke. Ein vermehrtes Auftreten äußerer Reize und deren damit einhergehende Verarbeitung kann diese wiederum reduzieren.
- Beta-Wellen reichen von 14 bis 30 Hertz. Sie werden des Öfteren nochmals unterteilt in ß1 und ß2 und haben eine geringere Amplitude. Beta-Wellen werden im Gegensatz zu den zuvor genannten Alpha-Wellen mit bestimmten Aufgaben wie dem Lösen von Mathematikaufgaben oder dem Unterdrücken von Bewegung in Verbindung gebracht.
- Gamma-Wellen schwingen bei einer Frequenz von 31 Hertz oder mehr. Sie können bei Prozessen wie Ideenfindung, Sprach- und Gedankenverarbeitung erscheinen. In der Narkose verschwinden Gamma-Wellen völlig und tauchen daraufhin im wachen Zustand wieder auf (vgl. Bera 2015, s. 64).

Insbesondere die Alpha-Wellen stellten sich in der Vergangenen Musikprojekten und bis heute als brauchbar für Echtzeit Kontrolle heraus. Das kommt unter anderem daher, dass sie während des Verlaufs einer Messung mit etwas Übung aktiv beeinflussbar bleiben.

1.1.5 EEG-Daten in EEGsuperMap

Neben der rhythmischen Aktivität gibt es weitere Hirnaktivität, die durch EEG-Messungen erfasst werden kann. Viele dieser Methoden sind jedoch aufgrund der Notwendigkeit, mehrere Messungen durchzuführen, nicht für die Echtzeitanwendung geeignet.

Wie im späteren Abschnitt Datenfunktionen (2.12) detaillierter beschrieben, bezieht sich EEGsuperMap daher auf die soeben beschriebenen Frequenzbänder. Um Features von Interesse zu extrahieren, wird die Bandstärke einzelner Frequenzbänder in Echtzeit berechnet. Diese Daten werden dann zur Laufzeit auf verfügbare Klangparameter gemappt und somit hörbar gemacht. Der Prozess, bei dem Daten akustisch dargestellt werden, wird als Sonifikation bezeichnet.

1.2 Sonifikation

Laut Kramer et al. (2010, S. 3-4) bezeichnet Sonifikation die Verwendung von nicht-sprachlichem Audio zur Übermittlung von Informationen. Genauer gesagt ist sie die Umwandlung von Datenbeziehungen in wahrgenommene Beziehungen in einem akustischen Signal, um die Kommunikation oder Interpretation zu erleichtern.

Obwohl in der Theorie Sonifkation primär als auditive Informationsdarstellung und -exploration dient, Bruce N. Walker 2011, s. 12, kommt sie auch als künstlerische Praxis zum Einsatz.

Aufbauend darauf geht Alberto de Campo in "Science by Ear" Campo 2009, s. 41 auf Datensonifikation zum Zweck der Exploration von 'auditiven Gestalten' (interessante Klangobjekte) ein. Dazu werden zunächst drei Ansätze unterschieden:

- Kontinuierliche Datenrepräsentation:

  Diese Methode behandelt Daten als quasi-analoge, kontinuierliche Signale und basiert auf einer ausreichend hohen Abtastrate, sodass das Signal frei von Abtastartefakten ist und die Interpolation zwischen den Datenpunkten reibungslos erfolgt. Sowohl einfache Audifikation als auch Parameterzuordnung, die kontinuierliche Klänge beinhaltet, gehören in diese Kategorie.

- Diskrete Punktdatenrepräsentation:

  Diese Methode erstellt individuelle Ereignisse für jeden Datenpunkt. Hier kann man die Daten leicht in unterschiedlichen Reihenfolgen anordnen, sowie Untergruppen basierend auf speziellen Kriterien auswählen (z. B. basierend auf Navigationseingaben).

- Modellbasierte Datenrepräsentation:

  Diese Methode nutzt eine komplexere Beziehung zwischen Daten und Audio, indem ein Modell eingeführt wird, dessen Eigenschaften mit den Daten verknüpft werden.

Die Grenzbereiche zwischen diesen Herangehensweisen sind mitunter fließend. So zeigt de Campo in der folgenden Dartellung wie eindeutig diskrete Noten über "Grain Clouds" in die kontinuirliche Datenrepäsentation übergehen. Bestimmend hierbei ist die Anzahl der Datenpunkte die es benötigt um eine auditive Gestalt wahrzunehmen.

In dieser Arbeit liegt der Fokus der Sonifikation nicht primär auf der Erleichterung der Interpretation. Vielmehr dient Sonifikation als Grundlage zur künstlerischen Datenexploration. Einkommen EEG Daten werden mit 200Hz gesampelt (mehr dazu in 2.1) um einen kontinuirlichen Datenstrom zu erzeugen. Auf dem Schaubild von de Campo würde die Gestalten also in der überlappenden Zone zwischen diskreter und kontinuirlicher Datenrepräsentation liegen.

Daher wurde entschieden die Interaktion mit EEG Daten in Echt-Zeit mit sowohl auditivem als auch visuellem Feedback zu ermöglichen. Um diese Interaktion mit den Daten vielschichtiger zu gestalten, werden Daten zusätzlich zur Audio Sonifkation als auditives Feedback auch visualisiert. Im gegensatz zur reinen Sonifikation der Rohdaten, werden EEG-Daten im Rahmen ihrer klanglichen Möglichkeiten innerhalb des Instrumenten Parameteraums sonifiziert. Ein wichtige Bestandteil dieses Prozesses ist - wie bei Sonifikation im allgemeinen auch - das Mapping.

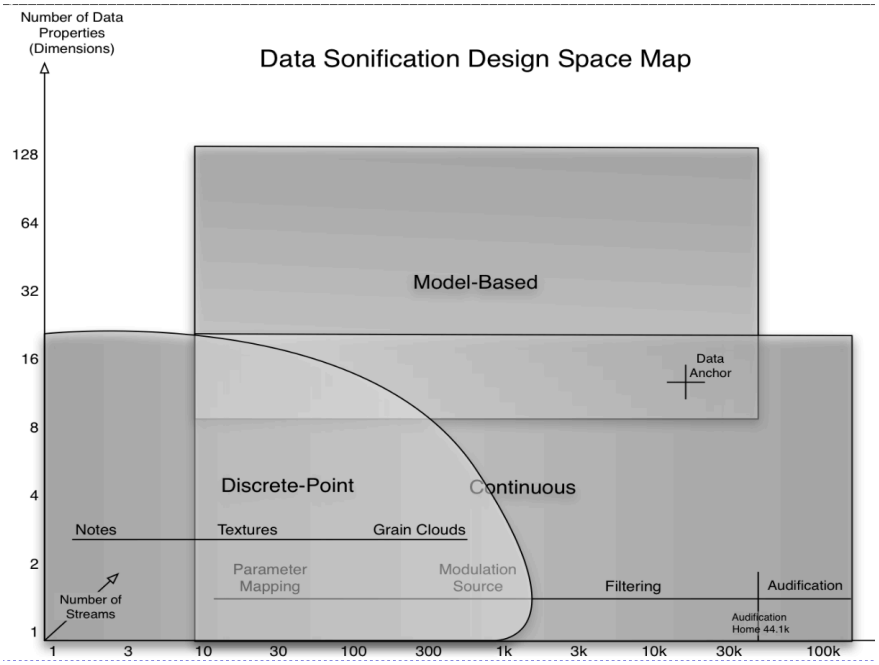

Figure 1.3: Ausschnitt aus "The Sonification Design Space Map" von Alberto de Campo aus Science by Ear S. 42

1.3 Mapping

Das Mapping oder genauer die Mappingfunktion bestimmt, wie Datenparameter und Klangparameter zueinander in Bezug stehen. Je nachdem was sonifizert werden soll, können unterschiedliche Mapping funktionen von direktem Mapping bishin zu komplexen Transferfunktionen zum Einsatz kommen. Grundsätzlich geht es beim Mapping bezogen auf Sonifikation aber immer darum, wie Daten im Audio Output repräsentiert werden sollen, sei es über direkte Sonifikation als Audiosamples oder als Kontrollsignal für Audioparameter. In ihrem Artikel "Parameter Mapping Sonification" aus "Sonfication Handbook" stellen Florian Grond und Jonathan Berger sich daraus ergebende Möglichkeiten und Herausforderungen fest: "The opportunities and challenges of PMSon are complicated by the inherent perceptual entanglement of sound synthesis and processing parameters, which constitute the two big challenges in the formalization of the mapping function" (Florian Grond 2011, S. 392). Diese formalisierung der Mappingfunktion nimmt bei vielen Digitalen Instrumenten eine elementare Rolle ein. Denn je nach Mapping können sich Datenparameter sehr vielfältig auf den letztendlichen Sound Output auswirken. Daher wurde in diesem Projekt entschieden dass Mapping als wichtiger Teil der Echt-Zeit Datensonifikation direkt in die Benutzeroberfläche des Instrumentes zu integrieren. Somit kann das

Mapping während einer Perfomance jederzeit angepasst und neu veradrahten werden.

1.4 Affordances und Ergodynamics

Über das Mapping und die zugrunde liegenden Daten entstehen bei digitalen Musikinstrumenten Parameterräume, die von uns bespielt werden können. Thor Magnusson spricht in diesem Zusammenhang von "Affordanzen"(Magnusson 2006, S. 443). Der Begriff stammt ursprünglich aus der Psychologie und beschreibt die wahrgenommenen Möglichkeiten, die ein Objekt bietet. Diese können jedoch subjektiv variieren. Ein Cello bietet beispielsweise, je nach Praxiserfahrung und kulturellem Hintergrund der Spielerin oder des Spielers, unterschiedliche Affordanzen. Dazu gehören Pizzicato, Tremolo, Scordatura oder Mikrotonalität (t. m. t. 2019, S. 42).

Darüber hinaus spricht Magnusson in "Ergodynamics And A Semiotics of Instrumental Composition" von "Ergodynamics" eines Instruments. Damit ist das Konzept gemeint, wie ein Instrument gespielt werden kann, quasi das "Gameplay" eines Instruments Magnusson 2019, S. 11. Dabei geht der Begriff über feste Gebenheit beziehungsweise die "Constraints" eines Instruments hinaus. Zwar bezieht sich Magnusson dabei auf das Instrument und seine Materialität an sich, jedoch beschreiben Ergodynamics auch, wie wir darauf spielen, und sind daher ebenso wie dessen Affordanzen subjektiv. Im Unterschied dazu beschreiben die Ergodynamics nicht direkt die Möglichkeiten, die ein Instrument bietet, sondern vielmehr, wie intuitiv diese genutzt werden können.

Bezogen auf ein Instrument, das mittels Hirnaktivität gesteuert wird, werfen diese Begriffe unweigerlich zusätzliche Fragen auf. So unterscheidet sich die Interaktion über EEG grundlegend von traditionellen - aber auch anderen digitalen - Musikinstrumenten. Dennoch können diese Begriffe auch im Bezug auf EEG Instrumente hilfreich sein. So stellt ein EEG Instrument wie EEG-superMap zusätzliche Herausforderungen im Bezug auf dessen Ergodynamics, aufgrund felhender physischer Interaktion. Ausgehend davon kann jedoch die Interaktion mit der dessen Software intuitiv gestaltet werden, um Daten zu explorieren und interessante Einstellungen zu finden.

Chapter 2

Das EEG-Instrument

Aufbauend auf den Konzepten des vorhergehenden Kapitels werden im Folgenden zunächst die verwendete Software von EEGsuperMap beschrieben und anschließend das Instrument als Gesamtsystem vorgestellt. Darauf werden dessen Kernaspekte im detail besprochen.

2.1 OpenBCI Hardware

Um EEG-Daten mit Software zu verarbeiten, müssen diese zunächst mittels Hardware gemessen und die elektrischen Signale anschließend in digitale Signale umgewandelt werden. Viele Geräte, die dazu in der Lage sind, sind jedoch extrem teuer und nicht quelloffen. OpenBCI liefert Hardware und Open-Source-Software, um in Forschung, Softwareentwicklung oder auch künstlerischen Anwendungen den Zugang zu EEG und anderen Biosignalen zu erleichtern.

Das Hardwareangebot teilt sich in zwei Boards zur Erfassung von Biosignalen auf:

- Das Ganglion-Board für 4 bis 8 Kanäle, einer Auflösung je nach Konfiguration von 19 Bit oder 18 Bit und einer Sampling-Rate von 200 Hz (OpenBCI 2022b Doku).
- Das Cyton-Board mit bis zu 16 Kanälen, einer Auflösung von 24 Bit bei einer Sampling-Rate von 250 Hz (OpenBCI 2022a Doku).

Beide Boards verbinden sich drahtlos über einen USB-Dongle mit der eigenen Softwareschnittstelle, wobei der Cyton-Dongle über den seriellen Port und der Ganglion-Dongle über Bluetooth Low Energy (BLE) mit dem jeweiligen Board kommuniziert. Gegenüber den Vorteilen, die eine drahtlose Verbindung mit sich bringt, wie Bewegungsfreiheit und Vermeidung elektrophysischer Risiken, beschränkt sie jedoch auch die Übertragungsrate. Im Fall des in diesem Projekt verwendeten Ganglion-Boards können über BLE etwa 100 Pakete pro Sekunde übertragen werden. Durch Datenkompression und einen akzeptablen Verlust

der Auflösung von 24 Bit auf 19 Bit werden 2 Samples pro Paket geschickt. Daraus ergibt sich wiederum die Samplerate von 200 Hz des Ganglion-Boards.

Beide Boards konnten mit einem Wifi-Shield für einen verbesserten Datendurchsatz ausgestattet werden. Allerdings wurde diese Erweiterung aufgrund ihrer Instabilität vorerst aus dem Verkauf genommen (OpenBCI 2022c Doku).

2.2 OpenBCI Software

Als Open-Source-Software hat OpenBCI es ermöglicht, dass sich online eine beständige Entwickler-Community gebildet hat. Dadurch ist bereits eine Vielzahl an Software entstanden, sowohl zur Messung von EEG-Daten als auch für künstlerische Anwendungen.

Als Einstieg können über die mitgelieferte Software von OpenBCI Biodaten beider Boards gefiltert und visualisiert werden. Diese teilt sich auf in eine vollumfängliche grafische Benutzeroberfläche (GUI) sowie die Brainflow-Bibliothek.

Brainflow Brainflow bietet als Datenerfassungs-API für Biosensordaten verschiedene, boardunabhängige Optionen zur Verarbeitung von Biosensordaten. Zusätzlich können über Brainflow auch Boards über synthetische Daten simuliert werden, sodass Software auch ohne Hardware — zumindest bis zu einem gewissen Grad — entwickelt und getestet werden kann. Aufgrund dessen ist es möglich, dass mehrere Boards verschiedener Hersteller mit demselben Code ausgelesen werden können. Die Core-Library ist in C/C++ geschrieben und kann als vorkompilierte Bibliothek von Paketmanagern wie NuGet oder Pip in mehreren Hochsprachen verwendet werden. Dazu gehören Python, Java und TypeScript. Grundsätzlich werden hier die vom Board gesendeten Bits ausgelesen und als Float64 weiterverarbeitet. Unabhängig von der Datenerfassungs-API kommt Brainflow auch mit einer eigenen API für digitale Signalverarbeitung (DSP). So können auf die erfassten Daten verschiedene Filter wie Bandpass und Notch oder Berechnungen wie Bandstärke und Denoising angewandt werden. Dies macht Brainflow über die Verarbeitung von Bio-Daten hinaus auch für klassische DSP-Anwendungen attraktiv, da hierbei dieselben optimierten C++-Algorithmen genutzt werden können.

OpenBCI GUI In der OpenBCI GUI können die durch Brainflow verarbeiteten Daten gefiltert und angezeigt werden. Die Oberfläche lässt sich aus verschiedenen Fenstern (Widgets) zusammenstellen. Auf diese Weise können verschiedene Datenströme wie Alpha-Bandstärke oder Kanal-Amplitude in entsprechenden Visualisierungen in der GUI simultan verfolgt werden. Zusätzlich können einige Parameter über das Networking-Widget per Standard-UDP, OSC oder seriellen Port versendet werden. Darüber hinaus ist es möglich, Daten als .txt-Dateien aufzuzeichnen und über die GUI abzuspielen. Die OpenBCI GUI ist Open-Source und wurde in Processing geschrieben.

Die GUI bietet einen sehr guten Einstiegspunkt zum Testen des EEG-Aufbaus, Visualisieren der Daten und deren Versenden an andere Software per Netzwerk.

Allerdings ist sie in diesem Projekt nicht direkt integriert, da es auch möglich ist, mithilfe des Brainflow-Backends EEG-Daten direkt an die Sound-Software zu schicken. Somit wird ein Zwischenschritt innerhalb der Echtzeitverarbeitungskette eingespart, zumal die Processing-Zeichenaufrufe der Echtzeitgrafen zusätzliche CPU-Leistung kosten.

2.3 Software-Architektur

Die bisher genannten Komponenten sind die Basis für die Verarbeitung von EEG-Daten mit OpenBCI. Sie lassen sich daher in vielen Arbeiten in unterschiedlichsten Anwendungen wiederfinden. Bevor die besonderen Merkmale dieser Arbeit im Detail erörtert werden, wird zunächst eine Übersicht gegeben, die den gesamten Signalfluss des Instruments zeigt, um diesen besser nachzuvollziehen.

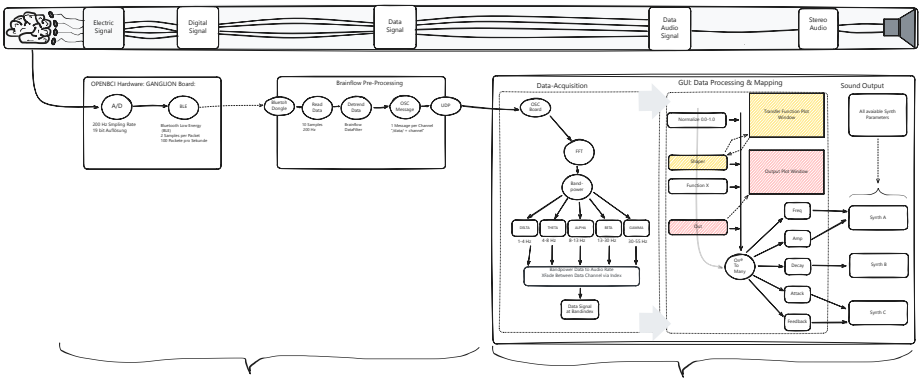

Figure 2.1: Signalfluss und Aufbau des Instruments

Die Software wurde so konzipiert, dass folgende Anforderungen erfüllt werden:

- Das Instrument soll adaptiv auf verschiedene Input-Ranges reagieren.
- Die Klangerzeugung ist unabhängig von der Datenverarbeitung und austauschbar.
- Einzelne Verarbeitungsschritte von Datenfilterung bis Mapping werden in einer grafischen Benutzeroberfläche (GUI) dargestellt und können in Echtzeit rekonfiguriert werden.

Betrachtet man zunächst den Kopfteil der Darstellungen, ist der Zustand des Signals sichtbar, welches die einzelnen Softwarekomponenten bis hin zum Audio-Output durchläuft.

Bevor die Daten spezifisch für Audio weiterverarbeitet und gemappt werden können, werden sie zunächst über Brainflow ausgelesen und aufgearbeitet.

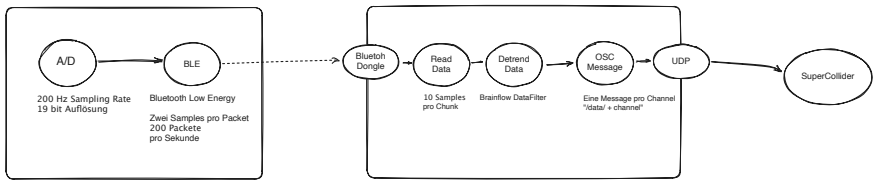

Figure 2.2: Datenverarbeitung in Brainflow

Innerhalb von SuperCollider kommt das grundlegende Konzept von EEGsuperMap zum Tragen. Eingehende EEG-Daten werden als digitale Audiosignale effizient mit DSP-Algorithmen weiterverarbeitet. Alle EEG-Daten auf Audiorate werden fortan als Datensignale bezeichnet. Datensignale sind quasi Audiosignale, werden im Unterschied zu ihnen jedoch niemals über den Hardware-Audio-Output ausgegeben, zumindest nicht als direktes Signal. Vielmehr werden sie als Signal verrechnet und dienen zur Kontrolle von Klangeinheiten, welche wiederum hörbare Audiosignale erzeugen. Der Unterschied zwischen Daten- und Audiosignal liegt also nicht direkt in technischer Natur, sondern vielmehr in ihrer Rolle innerhalb der EEGsuperMap-Software.

Daten auf Audiorate zu verrechnen hat den Vorteil, dass Algorithmen aus der Signal- und Audioverarbeitung auf den EEG-Datenstrom angewandt werden können. Empfangene Daten werden vergleichbar wie bei einem Instrument und einer dazugehörigen Effektkette behandelt: Das Eingangssignal durchläuft eine Abfolge von Verarbeitungsschritten, die das Signal transformieren. Über einen Dry/Wet-Regler kann zusätzlich eingestellt werden, wie stark das Signal vom jeweiligen Verarbeitungsschritt beeinflusst wird. Je nachdem, wie das Signal durch diese Abfolge geleitet wird, ergeben sich unterschiedliche Outputs.

Konkret auf die Software der Arbeit bezogen, resultiert daraus eine Effektkette auf empfangene EEG-Daten, eine Input-Chain, wobei jeder Effekt einen einzelnen Datenverarbeitungsschritt zur Folge hat. Dadurch können verschiedene Funktionen der Datenverarbeitung wie Datenglättung oder -skalierung unabhängig voneinander in Echtzeit auf das Signal angewendet werden.

Wie bei anderer Audio-Software können Datensignale mittels Crossfading auch untereinander gemischt werden. So können mehrere Input-Parameter als Datensignale verfügbar bleiben, zwischen denen jederzeit gefadet werden kann. Datenströme bleiben dadurch an verschiedenen Stellen austauschbar und editierbar, ohne unterbrochen zu werden.

Beispielsweise könnten zunächst die Bandstärke im Alpha-Bereich auf die Frequenz eines Klanges gemappt sein. Je nach Echtzeitdatenlage kann nun nahtlos zu anderen Frequenzbändern gewechselt werden, da alle Bandstärken als Multichannel-Datensignal bereits beim Start der Software zur Verfügung stehen.

Um dieses Konzept zu ermöglichen, wird zur Datenverarbeitung SuperCol-

lider verwendet, und entsprechende Algorithmen werden in Audiorate auf dem SuperCollider-Server ausgeführt.

2.4 SuperCollider

SuperCollider ist in zwei Prozesse unterteilt: die Sprache (sclang, auch als Client bezeichnet) und die Klangrendering-Engine (scsynth, auch als Server bekannt). Diese beiden Systeme verbinden sich über das Netzwerkprotokoll OpenSound-Control (OSC).

SuperCollider ist eine interpretierte, voll ausgestattete Programmiersprache. Während ihre Architektur an Smalltalk angelehnt ist, ähnelt ihre Syntax eher C++. Zu den Hauptmerkmalen der Sprache gehören die präzise Zeitsteuerung, die schnelle Prototypisierungsmöglichkeiten und die algorithmischen Bausteine für musikalische und andere zeitbasierte Kompositionen.

Im Gegensatz zu sclang ist scsynth ein Programm mit einer festen Architektur, das für hoch effizientes Echtzeit-Sound-Rendering entwickelt wurde. Klangprozesse werden durch Synthesegraphen erstellt, die aus einer dynamisch geladenen Bibliothek von Unit-Generatoren (UGens) bestehen. Signale können auf Audio- und Kontrollbussen geroutet werden, und Soundfiles sowie andere Daten können in Buffern gespeichert werden Till Bovermann 2011, S. 244.

SuperCollider bietet als Software mit Live-Coding-Funktionen bereits eine eigene Infrastruktur, um unterschiedliche Signale in Echtzeit verarbeiten zu können. Das beinhaltet klassische Audioverarbeitungsmethoden wie Filtering, Mapping oder Fading. Zusätzlich können aber auch beliebige Transformationen auf Kontroll- oder Audiosignale angewandt werden, wie eigens definierte mathematische Funktionen. Daher kommt in dieser Arbeit SuperCollider nicht nur als Klangsynthese-Engine zum Einsatz, sondern übernimmt auch das Processing und Mapping der EEG-Daten. Denn getreu dem Konzept des Datensignals laufen all diese Prozesse in Audiorate auf dem SuperCollider-Server.

In den folgenden Kapiteln sind die enthaltenen Kernkomponenten aus SuperCollider und deren Funktionen innerhalb des Instruments aufgeführt und erklärt.

2.5 Server Nodes und Proxy Nodes

In SuperCollider existieren Audioprozesse als Nodes (einzelne Klang-Einheiten) auf dem SuperCollider-Server. Sobald ein Audioprozess in der SuperCollider-Sprache gestartet wird, kommuniziert er über OSC (Open Sound Control) mit dem Server, um dort als Node repräsentiert zu werden.

Zusätzlich bietet SuperCollider über die integrierte Erweiterungsbibliothek "JITLib" die Möglichkeit, Platzhalter für diese Nodes auf dem Server zu verwenden. Diese Platzhalter, sogenannte Proxy Nodes, bringen Funktionalitäten aus dem Bereich des Live-Codings mit sich. Sie können in Echtzeit dynamisch ausgetauscht und rekonfiguriert werden. Darüber hinaus ermöglichen sie es,

einzelne Audioverarbeitungsprozesse modular voneinander abzukapseln und neu zusammenzustellen.

Die Modularität ergibt sich entweder aus der Verwendung vieler einzelner Nodes oder durch die Nutzung verschiedener "Slots" innerhalb einer Node. Diese Slots ermöglichen es, mehrere "Source"-Signale innerhalb einer Node zu verwenden oder Signale in vorhergehenden Slots nach bestimmten Regeln weiterzuverarbeiten. Eine solche Regel könnte beispielsweise eine Filterfunktion sein: Eine Filterfunktion in Slot Nummer [30] greift das Signal, das möglicherweise bereits in Slots [0] bis [29] verändert wurde, auf, wendet eine Funktion darauf an und gibt es an die folgenden Slots weiter (siehe Abb.). Nach dem letzten Slot wird das Signal standardmäßig an den Hardware-Audioausgang geroutet. Über einen vordefinierten Dry/Wet-Parameter kann reguliert werden, wie stark die Filterfunktion das Signal beeinflusst.

All dies geschieht innerhalb einer einzelnen Node Proxy Role und mit den Möglichkeiten des Just-in-Time-Programming: Alle Slots können in Echtzeit ausgetauscht und rekonfiguriert werden.

Im EEG-Instrument kommen Node Proxy Roles als Funktionen auf EEG-Datensignale zum Einsatz. Hierbei werden sie jedoch in Form von ProxyChains verwendet.

2.6 ProxyChains und Proxy Roles

Mit der JITlib-Erweiterung ProxyChain lassen sich Slots einer Node bzw. deren Node Proxy Roles zusätzlich unter einem Namensschlüssel ablegen und individuell hinzufügen oder entfernen. ProxyChains verfügen über eine eigene GUI durch die ProxyChainGUI-Klasse. Sie ermöglichen die Steuerung der Parameter der einzelnen Slot-Funktionen über Slider und die Aktivierung oder Deaktivierung von Slots.

In der Klangsynthese würde man eine ProxyChain beispielsweise als Effektslots eines einzelnen Instruments einsetzen. Über die Filter-Node Proxy Role wird ein Quellensignal durch verschiedene Effekte wie Low-Pass-Filter oder Delays geroutet, um einen Output zu erzeugen. Bei dieser Anwendung liegt der Schwerpunkt auf dem Ergebnis einer Effektkette.

Für das EEG-Datensignal ist der Output jedoch nur der letzte Schritt einer Verarbeitungskette, deren Zwischenergebnisse ebenso interessant sein können wie der finale Output. Daher wurde für diesen Fall eine spezielle Node Proxy Role definiert. Diese erlaubt es, Signale zu filtern und zusätzlich aufzuteilen. Jede Operation auf dem Signal oder jede Funktion innerhalb einer ProxyChain erhält ihren eigenen Output-Bus. Auf diese Weise kann jeder Einzelschritt auf Klangparameter geroutet werden, ebenso wie der finale Output am Ende der Verarbeitungskette.

Für die Software der DMIs bedeutet dies, dass eine ProxyChain Datenverarbeitungsfunktionen auf EEG-Daten anwendet. Der Code dieser Funktionen kann zur Laufzeit abgeändert oder ausgetauscht werden. Innerhalb dieser Chain ist nicht festgelegt, auf welche Parameter die verarbeiteten Daten gemappt wer-

den sollen. Die Chain arbeitet unabhängig von der eigentlichen Klangverarbeitung. Beim Start der Mapping-Software werden alle verfügbaren Klangparameter aller Instrumente der aktuellen SuperCollider-Session erfasst. Über die Split-Node Proxy Role können diese im Anschluss mit den Chain-Outputs belegt werden. Dies erlaubt es, Klangsynthese-Konzepte zunächst autonom zu entwickeln. Eine SuperCollider-Session, die ursprünglich nicht für EEG-Daten entwickelt wurde, kann auf diese Weise über die Mapping-Software gesteuert werden, um auf unkomplizierte Weise eventuell interessante Wechselwirkungen zu entdecken.

2.7 OSC

Open Sound Control (OSC) ist ein Netzwerkprotokoll für die Kommunikation zwischen verschiedenen Hard- und Softwareelementen innerhalb eines Netzwerks. Musiksoftware wie Csound, Max/MSP oder SuperCollider können über OSC sogenannte OSC-Messages austauschen und mit visuellen Schnittstellen wie Processing oder Unity kommunizieren.

Diese Messages werden als OSC-Bundles verschickt und können Daten vom Typ int32, float32 und string enthalten. Die Nachrichten haben eine eigene Adresse, die beispielsweise wie folgt aufgebaut sein kann: "/board/data/eeg".

Im Gegensatz zu Protokollen wie MIDI ermöglicht OSC eine deutlich höhere Auflösung durch die größere Größe der versendeten Daten, sodass Parametersprünge weniger hörbar sind.

OSC ist unabhängig vom Transportprotokoll, jedoch wird in der Regel UDP verwendet. Da häufig ein Stream von Echtzeit-Kontrollparametern gesendet wird, ist die schnellere Datenübertragung von UDP gegenüber dem anderen populären Transportprotokoll TCP zu bevorzugen.

In diesem Projekt wird OSC hauptsächlich als Schnittstelle zwischen Brainflow, das ein Python-Skript zur Auslesung des Boards verwendet, und der Mapping-Software in SuperCollider genutzt. Das Skript sendet Daten mit der maximalen Samplerate des Boards als Datenstrom an SuperCollider, wo die EEG-Rohdaten weiterverarbeitet werden. Die Nachrichten sind so strukturiert, dass jeder EEG-Kanal separat gesendet wird, sodass SuperCollider entscheiden kann, welche Kanäle empfangen werden sollen.

Zusätzlich können von SuperCollider aus Nachrichten an das Python-Backend zurückgesendet werden, beispielsweise zum Laden oder Speichern von EEG-Rohdaten. OSC-Nachrichten spielen auch eine wichtige Rolle bei der Kommunikation zwischen den Softwarekomponenten. Die Mapping-Logik der Software wurde in mehrere Klassen aufgeteilt. Um die Modularität der Gesamtstruktur zu gewährleisten, erfolgt die Kommunikation der Softwarekomponenten untereinander über OSC-Nachrichten. Ein Button in der GUI der Software sendet Werte per OSC-Nachricht, ohne zu wissen, wie diese weiterverarbeitet werden. Andere Komponenten können die Nachrichten empfangen und entsprechend reagieren.

Dies verbessert die Austauschbarkeit der einzelnen Softwarebausteine. Da

keine direkte Referenz auf die Klasse zur Weiterverarbeitung der Daten besteht, können diese einfacher verändert oder entfernt werden.

2.8 GUI

Die Interaktion mit den bisher beschriebenen Komponenten erfolgt in SuperCollider klassischerweise über Code innerhalb eines Editors. Zusätzlich können SuperCollider-GUI-Klassen Parameter und Prozesse aus dem Code in einem grafischen User Interface (GUI) visualisieren und manipulierbar machen. Dies umfasst klassische GUI-Elemente wie Fader oder Knobs sowie speziellere Elemente wie das Auslesen oder Zeichnen einer Wellenform.

Die Einbindung einer GUI ist für dieses Projekt wichtig, da sie den im Abschnitt 1.2 (Sonifikation) genannten Aspekt der Datenexploration unterstützt und die Interaktion vereinfacht. Entscheidungen können schnell und intuitiv getroffen, visualisiert und abgehört werden, ohne Code schreiben zu müssen. Dies ist besonders relevant für EEG-Daten, da die Konzentration des Interpreten bereits auf die Erzeugung der Daten fokussiert ist.

Zusätzlich verbessert die GUI den Zugang zum Instrument. Mit einem Basiswissen über SuperCollider und Python kann auch ohne detaillierte Kenntnis des Source Codes EEG-Daten sonifiziert und exploriert werden. Die bereits zuvor beschriebene ProxyChain kommt mit einer eigenen ProxyChainGUI, die in die Oberfläche des Instruments integriert wurde. Innerhalb der ProxyChainGUI können alle Parameter der Slot-Funktionen über Regler angepasst werden. Außerdem können Funktionen aktiviert oder deaktiviert werden. Welche Funktionen genutzt werden und deren Reihenfolge müssen jedoch zuvor im Code festgelegt werden.

2.9 Mapping des Datensignals auf Input-Busse

Um das Mapping der Output-Busse der ProxyChain auf Input-Parameter von Instrumenten zu ermöglichen, wurden der SC-Library die Klassen ListGrid.sc und Rangebox.sc hinzugefügt.

ListGrid fasst mehrere Listview.sc-GUI-Elemente in einer Mapping-Matrix zusammen. Zunächst werden die verfügbaren Parameter aller Instrumente der aktuellen Session abgefragt. Anschließend wird für jedes Instrument ein Listview mit den zugehörigen Parametern erstellt. Über den Listview können dann einzelne oder, durch Mehrfachauswahl, auch mehrere Parameter auf die ProxyChain, also die verarbeiteten Input-Parameter, gemappt werden. Da die ProxyChain, wie bereits erwähnt, mit Signalen arbeitet, könnte man auch sagen, dass ein Output-Signal der ProxyChain über einen Bus auf einen Input-Bus eines Instruments geroutet wird.

Das Routing übernimmt auf der untersten Ebene die Rangebox.sc, eine Klasse, die aus einer Mapping-Funktion und zwei NumberBoxes besteht. Über die NumberBoxes kann in der GUI dem Parameter eine Output-Range zugewiesen

werden. *(HIER Abbildung ListGrid)* Dies funktioniert mit jedem Input-Parameter, da sich diese durch die bereits erwähnte Standardisierung immer im Wertebereich zwischen 0 und 1 befinden.

2.10 Mapping und Crossfade

Damit diese Form des Echtzeit-Mappings ohne Artefakte möglich ist, wird der Output-Bus der Chain nicht direkt auf den Input-Bus eines verfügbaren Sound-Nodes geroutet. Stattdessen wird er zunächst auf den Input-Bus einer temporären ProxyNode geroutet. Dies hat den Vorteil, dass der Chain-Output über die xset-Methode der NodeProxy auf die Sound-Parameter gemappt werden kann. Im Gegensatz zur set-Methode kann xset eine Fade-Zeit übergeben werden, um zwischen altem oder keinem Input und neuem Input zu crossfaden.

```
// Bus vom Split-Node-Proxy-Roll
bus = BusDictManager().getOrSet(busName);

// Für die Reihenfolge der Ausführung
Ndef(key).clear;

// Temporärer Node-Proxy mit Bus als Input und Range von der GUI
tempProx = Ndef(key, {
LinLin.ar(In.ar(bus.index, 1), 0.0, 1.0, range[0], range[1])
});

// Fade-Zeit vom Button
tempProx.fadeTime = fadeTime;
node.fadeTime = fadeTime;

// Mini-Delay-Randomize-Button +
// viele Synths (zu viele xset zur gleichen Zeit)
schedlist = schedlist.rotate;

AppClock.sched(schedlist[0], {
node.xset(param, tempProx);
});
```

Dieser Methode wird standardmäßig eine kurze Fade-Zeit übergeben, um Artefakte durch abruptes Mapping zu vermeiden. Allerdings kann die Fade-Zeit auch höher eingestellt werden, um während einer Session lange Übergänge zwischen verschiedenen Mappings zu ermöglichen. Im Gegenzug ist es möglich, die Fade-Zeit auf 0 zu setzen, um Artefakte zuzulassen und den Mapping-Prozess transparent (mit allen Knacksern) zu sonifizieren.

Ursprünglich war dieser Parameter als fest kodierter Wert innerhalb der Black Box des Instruments gedacht, um Artefakte beim Mapping zu verhindern. Während der Implementierung und des Testens wurde jedoch festgestellt,

wie stark der Parameter die Eigenschaften des Instruments beeinflussen kann. Daher wurde entschieden, den Parameter in der GUI offenzulegen und ihn als Teil des Interfaces zu betrachten.

2.11 Modifikation der ProxyChainGUI

Die ProxyChainGUI wurde so erweitert, dass bei der Interaktion mit einer Slot-Funktion ein entsprechendes ListGrid mit verfügbaren Klangparametern angezeigt wird. Wählt man einen Parameter in der Liste aus, wird der über die Split-Rolle abgesplittete Output-Bus auf diesen gemappt. Auf welche Range der Verarbeitungsschritt gemappt wird, bestimmt das Rangebox.sc-Element, das zwei NumberBoxes für Minimal- und Maximalwerte enthält.

Da alle angezeigten ListViews im Mehrfachauswahlmodus instanziiert werden, können auch mehrere Parameter gleichzeitig mit einem Mouse-Drag ausgewählt werden.

Jeder Slot-Funktion der ProxyChain ist somit eine unabhängige Mapping-Matrix zugeordnet, in der Datensignale Klangparametern (neu) zugewiesen werden können. Diese Matrix durchläuft mehrere One-to-Many-Mappings simultan und in Echtzeit. Als Ergebnis ist diese Art des Mappings ein integraler Bestandteil des digitalen Instruments. Wichtige Entscheidungen in Bezug auf Sound und Ergonomie werden innerhalb der Echtzeit-Mapping-Matrix getroffen und können nach Belieben wieder verworfen werden.

Thor Magnusson verweist in "Sonic Writing“ auf die rechnerische Natur digitaler Musikinstrumente und leitet daraus das Instrument als epistemisches Modell ab:

"The core quality of digital musical instruments is their computational nature, and this involves arbitrary physical materials, mappings, and sound synthesis. The instrument becomes an epistemic model into which we write our music theory, and we test that theory through active engagement and performance, equally in the personal lab of the studio, as well as on the social lab of the stage."

2.12 Datenfunktionen

Bevor EEG-Daten in der ProxyChain zur Verarbeitung und zum Mapping als Audiosignal ankommen, werden zunächst Features aus den EEG-Daten analysiert. Im Fall von EEGsuperMap handelt es sich um die Bandstärke der EEG-Daten in den folgenden Frequenzbändern:

- Delta (1-4 Hz)
- Theta (4-8 Hz)
- Alpha (8-13 Hz)
- Beta (13-30 Hz)

- Gamma (30-55 Hz)

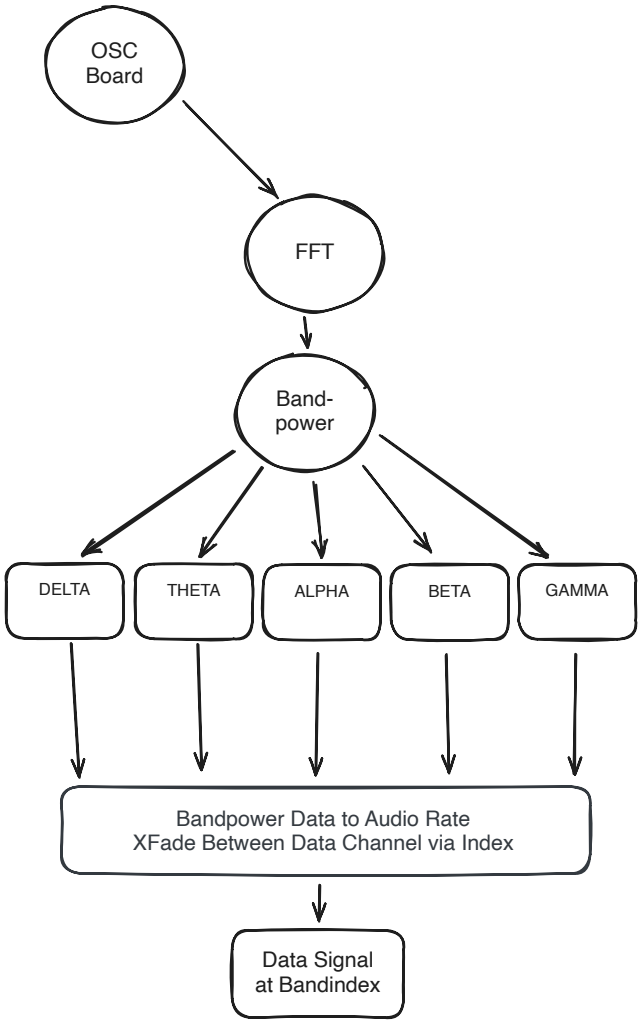

Figure 2.3: Verarbeitungskette in SuperCollider

Um diese Bandstärken zu berechnen, werden die Datenfunktionen des OpenBCI SuperCollider-Ports verwendet (Hofstadter 2024). Dank diesem Projekt wurde einige Datenfunktionen aus Brainflow, sowie GUI Widgets nach SuperCollider portiert.

2.12.1 Berechnung der Bandstärke

Mithilfe einer Fast Fourier Transformation werden Frequenz-Indizes ermittelt, um die Leistungsspektraldichte (Power Spectral Density, PSD) zu berechnen.

Dies geschieht für jeden EEG-Kanal sowie jedes Frequenzband der multidimensionalen EEG-Daten.

```
channels.do{|c, i|
    if(c.active and:{c.data.size>0}, {
        c.data = fft.fft(c.data);
        if(pop1.value > 0, {
            c.data = smoothingFilters[i].filterLog(c.data.max(0.01));
        });
        bands.do{|b, j|
            var psdArr = c.data.copyRange(indices[j][0], indices[j][1] - 1)**2 * factor / 4
            ~bandsBuffer[i][j].pop;
            ~bandsBuffer[i][j].insert(0, psdArr.sum);
        };
    });
};
```

Die berechneten Bandstärken werden im zweidimensionalen bandsBuffer zwischengespeichert. Im nächsten Schritt wird daraus ein Multichannel-Datensignal erstellt, das alle Bandstärken in allen vier EEG-Kanälen enthält, also in diesem Beispiel 20 Kanäle. Diese Werte werden im Format [Band -> Kanal] einem Ndef (ProxyNode auf dem Server) zusammen mit einem Trigger übergeben.

2.12.2 Bandstärke und Trigger

```
~bands.do({|band, band_index|
    var channelArr;
    // bandsBuffer.flop: shape -> [5, 4, 60]
    channelArr = ~bandsBuffer.flop[band_index].collect({|channel| channel.mean});
    Ndef(\bandpower).set(band, channelArr);
});

Ndef(\bandpower).set(\trigger, 1);
```

Alte Werte werden über Demand.ar so lange auf Audiorate gehalten, bis über den Trigger neue Werte abgefragt werden. Somit sind die Bandstärken bereits auf die jeweilige Audiorate des Servers hochskaliert.

Der nächste Schritt profitiert davon, dass die EEG-Daten nun in Form von Audiosignalen vorliegen. Im ersten Slot der ProxyChain, der das 20-Kanal-Datensignal als Input erhält, kann stufenlos in Echtzeit zwischen Frequenzband und Kanal gepannt werden. Dazu wird das Signal so umgeformt, dass über SelectX zunächst ein Band angefadet werden kann. Danach kann mit dem Ugen SelectXFocus zwischen einzelnen EEG-Kanälen überblendet werden. Sowohl SelectX als auch SelectXFocus sind Ugens, um zwischen Audio- oder Kontrollsignalen zu crossfaden. SelectXFocus bietet die zusätzliche Option, über dessen

Focusparameter die ”Fuzziness“, also den Scope, der Auswahl stufenlos zu beeinflussen. Bei einem sehr geringen Wert von 0 werden alle EEG-Kanäle gleichwertig einbezogen. Ein Focuswert ermöglicht ein gezieltes Panning zwischen einzelnen EEG-Kanälen.

```
Ndef(\chPan, { |band, channel, focus|
    var sig, input, chs;

    input = Ndef(\bandpower).ar;
    input = input.reshape(5, 4);

    chs = SelectX.ar(band, input);
    sig = SelectXFocus.ar(channel, chs, focus);

});
```

Damit EEGsuperMap unmittelbar und ohne Kenntnisse von sclang genutzt werden kann, sind weitere Slot-Funktionen der ProxyChain vordefiniert. Aus einer Auswahl wird eine Verarbeitungskette erstellt, die folgende Grundbausteine enthält:

- Input-Signal
- Band / Channel Panning + Scaling
- Rohsignal + Split-Rolle
- Shaping Signal + Split-Rolle
- Outbuf für Echtzeit-Signalplotfenster

Im folgenden exemplarischem Codebeispiel werden aus einer kleinen Auswahl an Funktionen zwei InputChains generiert, die in der GUI unabhängig voneinander eingestellt und gemappt werden können. Über die out Filterfunktion wird das Signal an der jeweiligen Stelle abgegriffen und in einen Buffer für das Plotfenster in der GUI geschrieben.

2.12.3 ProxyChain

```
// Buffer for Waveshaping input signal
t = Buffer.alloc(s, 200, 1, bufnum: 70);

ProxyDictChain.add(
    \sig, \mix -> { LFNoise1.ar(1).range(0, 1) },

    \raw, \split -> { |in, raw| in.asArray[0] },

```

```
    \shaper, \split -> { |in, amp = 1, shaper|
        // Index into the table with raw Signal
        Index.ar(t, in.asArray[0].abs * BufFrames.kr(t));
    },

    \out, \filter -> { |in|
        var phase, rate;
        in = in.asArray[0];
        rate = 44100 / (1024 * 0.5);
        phase = Phasor.ar(0, 1 / rate, 0, 1024);
        BufWr.ar(in, 50, phase);
        in;
    },

    \out2, \filter -> { |in|
        var phase, rate;
        in = in.asArray[0];
        rate = 44100 / (1024 * 0.5);
        phase = Phasor.ar(0, 1 / rate, 0, 1024);
        BufWr.ar(in, 51, phase);
        in;
    },
);

// Clear and set up proxy chain
d.clear;
d = ProxyDictChain(\test3, [\sig, \raw, \shaper, \out]);
c = ProxyDictChain(\test4, [\sig, \raw, \out2]);

~app.(proxychains: [d, c], tbuf: t);
```

Der Aufbau der GUI und die verfügbaren Parameter hängen davon ab, wie die modifizierten ProxyChains der app-Funktion der Software übergeben werden. Änderungen an den Funktionen und dem Aufbau der Chain können auch zur Laufzeit vorgenommen werden. Damit diese Änderungen in der Mapping-Logik und GUI korrekt übernommen werden, sollte der app-Funktion die neue Chain übergeben werden.

```
// Alte ProxyChain
d.clear;

// Neuer Aufbau
d = ProxyDictChain(\basic, [\inputsig, \shaper, \out]);
~app.(proxychains: d, tbuf: t);
```

Dadurch ist es möglich, die Verarbeitungskette auf dem Input-Parameter

jederzeit neu zu konfigurieren. Um diese Flexibilität beizubehalten, ist der Aufbau der ProxyChain vom Code getrennt, der die Basisfunktionalität der Mapping-Software beinhaltet. Grundlegende Funktionen der Software sind in SuperCollider-Klassen implementiert, die im Extension-Ordner der SC Class Library gespeichert sind. Diese können zur Laufzeit nicht mehr verändert werden und werden von der app-Funktion aufgerufen.

Während einer EEG-Performance stellt sich die Frage, ob es sinnvoll ist, die InputChain live umzustrukturieren oder ihre Funktionen neu zu schreiben. Stattdessen können einzelne Parameter der Funktionen über die GUI angepasst und gemappt werden. Allerdings besteht auch die Möglichkeit, zuvor aufgezeichnete Rohdaten abzuspielen, um die ProxyChain "offline" mit vergleichbaren Daten zu optimieren.

2.13 Aufzeichnen und Abspielen von Rohdaten

Ähnlich wie in der OpenBCI GUI können EEG-Rohdaten als .txt- oder .csv-Datei aufgezeichnet und zu einem späteren Zeitpunkt wieder eingelesen werden. Die Aufzeichnung übernimmt Brainflow, bevor die Daten von SuperCollider weiterverarbeitet werden. So lassen sich unterschiedliche DSP-Funktionen für EEG-Rohdaten durchtesten. Dateiname sowie Lese- oder Schreibmodus werden in der GUI in SuperCollider festgelegt. Diese Informationen werden an das Brainflow-Skript übermittelt.

2.13.1 Switch-Modus

```
switch_lock = threading.Lock()
write_lock = threading.Lock()

while True:
with switch_lock:
current_switch = switch

arduino

if current_switch:
    end = count + chunk_len
    chunk = data[count:end].transpose()
    count = (count + chunk_len) % (data_len - chunk_len)
else:
    chunk = board.get_current_board_data(chunk_len)
    with write_lock:
        if write:
            DataFilter.write_file(chunk, filename, "a")

```

```
20  # Process and send to frontend
21  process_chunk(chunk, filename, client)
22  time.sleep(1 / sr)
23
```

Im Skript kann mittels threading.Lock() zur Laufzeit zwischen Lese-/Schreibmodus oder Live-EEG-Daten geswitcht werden, ohne die Software neu starten zu müssen.

2.14 Save System und Presets

Eine wichtige Funktion innerhalb der Software übernimmt die eigens angelegte Utility Klasse in SuperCollider, auf die jedes Objekt Zugriff hat, um eigene Werte oder sich selbst abzuspeichern oder zu laden. Dies geschieht mithilfe der Archive Klasse von SuperCollider. Dort können ganze Objekte abgespeichert und anschließend wieder geladen werden. Dies vereinfacht den komplexen Vorgang, den Zustand der Mapping-Software jederzeit auf der Festplatte zu speichern und bei Bedarf wieder abzurufen.

Zusätzlich wird dieser Prozess durch Modularität vereinfacht. Sobald das Programm beendet wird, muss nicht verfolgt werden, wo sich etwas verändert hat, da jede Klasse ihre eigenen Änderungen unter einem einzigartigen Pfad abspeichern und bei Start wieder laden kann.

2.14.1 Auszug aus dem Save-System von Utility

```
1   load { |key, empty|
2       var item;
3
4       if(Archive.global.at(path).isNil, {
5           ^empty;
6       });
7       if(Archive.global.at(path)[key].isNil, {
8           ^empty;
9       });
10      item = Archive.global.at(path)[key];
11      ^item;
12  }
13  save { |key, item|
14
15      item = item.copy;
16      if(Archive.global.at(path).isNil, {
17          Archive.global.put(path, ())
18      });
19      Archive.global.at(path)[key] = item;
20  }
```

Dieser Pfad setzt sich aus Namen bzw. Keys der ProxyChain, Slotfunktion und Synth-Parametern zusammen. Sobald einer dieser Keys im Code manuell geändert wird, ergibt sich daraus ein neuer Pfad. Bleiben die Keys jedoch gleich und ihre zugeordneten Funktionen ändern sich, können alte Objekte über den unveränderten Pfad wieder geladen werden. So speichert eine Rangebox ab, welchen Slot-Output der ProxyChain über ihre Range auf welchen Synth-Parameter gemappt wird. Sowohl die Funktion zur Datenverarbeitung im jeweiligen Slot als auch die Klangsynthesefunktion können ausgetauscht werden. Sofern der Name der Slot-Funktion sowie der Name der Klangsyntheseparameter unverändert bleiben, können das Mapping und die Range wieder geladen werden.

Unabhängig davon verfügt die Software über vier Presetslots, um zwischen Voreinstellungen ad hoc zu wechseln. Beim Wechseln wird der Mapping-Prozess resimuliert. Daher wirkt sich auch die zuvor angesprochene FadeTime auf den Presetswitch aus. Längere FadeTimes können dadurch auch als performatives Mittel eingesetzt werden, um zwischen Presets zu crossfaden.

```
BusDictManager {
    classvar bus_path = \bus_data5;
    *new {
        ^super.newCopyArgs()
    }
    put{ |key, bus|
        if(Archive.global.at(bus_path).isNil, {
            Archive.global[bus_path] = ();
        });
        Archive.global.at(bus_path)[key] = bus;
    }
    get{|key|
        ^Archive.global.at(bus_path)[key.asSymbol];
    }
    getOrSet{|key|
        if(Archive.global.at(bus_path).isNil, {
            Archive.global.put(bus_path, ());
        });
        if(Archive.global.at(bus_path)[key.asSymbol].isNil, {
            Archive.global.at(bus_path)[key.asSymbol] = Bus.audio(Server.local, 1);
        });
        ^Archive.global.at(bus_path)[key.asSymbol];
    }
}
```

Neben der Benutzerfreundlichkeit ist die Verwendung der Archive Klasse auch essenziell für die Verbindung zwischen ProxyChain-Slots und Klangparameter. Da ProxyChain und Mapping-Funktion nicht voneinander wissen, erfolgt die Verbindung indirekt über Busse, die unter einzigartigen Keys mithilfe

eines Busmanagers abgelegt werden. Nur die Mapping-Funktion der passenden Rangebox kann über ihren eigenen Key auf den korrekten Bus zugreifen. Dadurch wird durch den Key garantiert, dass für die Mapping-Verbindung immer derselbe Bus mit demselben Busindex auf dem Server verwendet wird.

2.15 Randomize Button

Aus der soeben beschriebenen computational nature von digitalen Musikinstrumenten (DMIs) ergibt sich auch die Implementierung der Randomize Funktion. Per GUI-Button wird das Routing der Datensignale sowie die Mapping-Range in Echtzeit neu gewürfelt. Zu Beginn einer Session können dadurch einige Setups durchgewürfelt werden, um einen Eindruck des Parameteraums zu erhalten. Zusätzlich kann diese Funktion durch das Setzen längerer FadeTimes beim Remapping auch als performatives Feature eingesetzt werden.

Ein interessanter Aspekt dabei ist, dass ein Gefühl der Kontrolle entstehen kann, obwohl keine der Kontrollparameter auf explizite Werte eingestellt wurde. Sobald ein Setup mit Live-EEG-Daten sich als "spielbar" anfühlt, können durch minimale Veränderungen Anpassungen vorgenommen werden. Bestenfalls lassen sich damit neue Mappings und Ranges finden, die besonders gut für Brainwave-Kontroll-Daten geeignet sind.

2.16 Plotter

Innerhalb der Oberfläche der DMIs nehmen die Plot-Fenster eine besondere Rolle ein. Die Klasse Plotter in SuperCollider dient dazu, verschiedene Daten und Kurven zu visualisieren. So können sowohl Audio- als auch Kontrollsignale sowie Envelopes, Arrays und Buffersamples dargestellt werden.

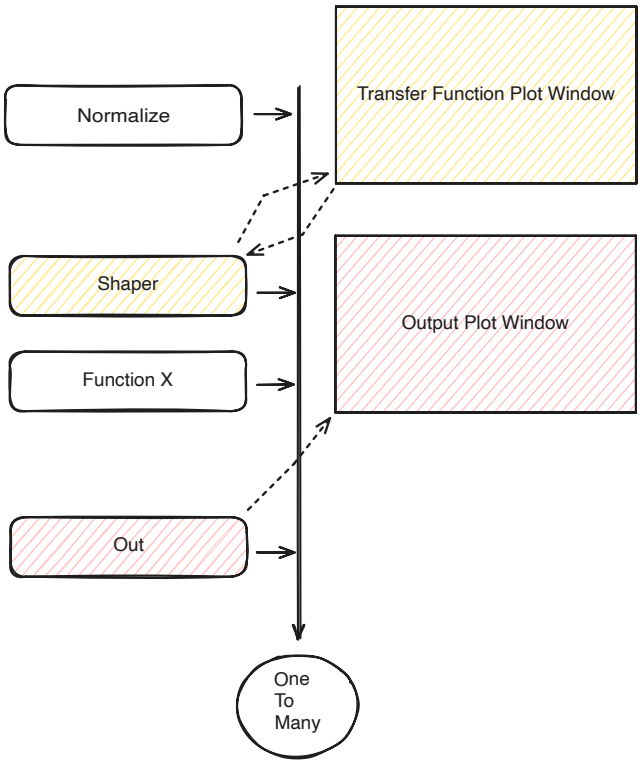

Figure 2.4: Plotter in der GUI

Durch diese zusätzliche Visualisierung können komplexe Signalverarbeitungen und deren mathematischer Hintergrund besser nachvollzogen werden. Für jede Input-ProxyChain dient daher ein Output-Plot als visueller Signal-Monitor. Das verarbeitete Datensignal wird über die Klasse BufWr in einem eigenen Verarbeitungsschritt der Chain in einen Audio-Buffer geschrieben. Dieser Buffer wird mithilfe eines Loops in der SuperCollider-Sprache ausgelesen, um ein Plot-Fenster in Echtzeit zu aktualisieren.

An welcher Stelle das Signal abgegriffen wird, hängt davon ab, an welcher Position innerhalb der `ProxyChain` der Verarbeitungsschritt platziert wird. Das Signal wird nach den verschiedenen Verarbeitungsschritten der `ProxyChain` noch einmal visualisiert. Die Visualisierung zeigt somit immer das Ergebnis der vorherigen Signalverarbeitung.

Über die `Plotter`-Klasse können eigene Kurven gezeichnet werden. In einem weiteren Plot-Fenster dienen diese als Transferfunktion für Input-Parameter. Auf das Datensignal wird also Waveshaping angewandt, dessen Transferfunktion in Echtzeit neu gezeichnet werden kann.

Chapter 3

Einordnung

3.1 Kontext

Aufgrund der interdisziplinären Natur von EEGsuperMap bewegt sich das Projekt sowohl im künstlerischen als auch im wissenschaftlichen Kontext. Angefangen bei den Grundlagen der Elektroenzephalographie über Meilensteine der digitalen Musikinstrumentforschung bis hin zu neueren Projekten mit OpenBCI, baut EEGsuperMap auf grob einhundert Jahre Forschung und Musikkultur auf.

Eine der prominentesten und wohl auch einflussreichsten Musikstücke mit EEG ist "Music for Solo Performer" von Alvin Lucier. In `Music for Solo Performer` werden Hirnwellen auf denkbar direkte Weise sonifiziert: Über Lautsprechermembranen, die die zuvor gefilterten und verstärkten Alpha-Wellen wiedergeben. Damit die für den auditiven Bereich ab ca. 20 Hz zu tiefen Brainwaves dennoch hörbar werden, sind die Membranen auf verschiedenen im Raum verteilten Schlagwerken aufgelegt und übertragen ihre Schwingungen auf deren Felle.

Das Stück zeigt den wirkungsvollen Effekt, dass höhere Bandstärken im Alpha-Bereich auch mehr akustischen Output erzeugen. Je entspannter Alvin Lucier auf der Bühne ist, desto mehr Schlagwerk erklingt, was wiederum für weniger Alpha-Wellen in der Bandstärke sorgt. So verdeutlicht `Music for Solo Performer` diesen Widerspruch und den Feedback-Loop, der entsteht, wenn es die eigenen Hirnwellen sind, die sonifiziert werden.

Durch die Verwendung der elektrischen EEG-Signale ohne deren Digitalisierung unterscheidet sich der Aufbau von Lucier grundlegend von EEGsuperMap. Zur Zeit der Uraufführung war digitale Signalverarbeitung noch nicht relevant. Anstatt Daten zu digitalisieren und eine Bandstärke im Alpha-Bereich als Parameter für Klangsyntese zu verwenden, sind es die bandgefilterten elektrischen EEG-Signale, die die Percussions ansteuern. Mit EEGsuperMap hingegen lassen sich aufgrund der Beschränkungen innerhalb der Sample-Rate die rohen Hirnwellen nur annähernd sonifizieren. Während EEGsuperMap durch die "computational" - also die offene Struktur beim Mapping - charakterisiert

wird, macht Luciers Werk die direkte Sonifikation der Alpha-Wellen und das daraus resultierende Feedback-Loop aus.

3.2 Ausblick

Im Entstehungsprozess von EEGsuperMap haben es einige Features nicht in den aktuellen Stand der Software geschafft, die jedoch für kommende Updates interessant sein könnten.

PyAudio Bereits implementiert war die direkte Sonifikation der Rohdaten über PyAudio. Ein Problem dabei war die für die direkte Sonifikation zu geringe Sample-Rate des Boards. Um die ohnehin schon geringe Auflösung nicht weiter zu verzerren, wurde entschieden, Daten für die direkte Sonifikation über vier Audiokanäle pro EEG-Kanal zu versenden. Diese werden unmittelbar nach dem Pre-Processing über die Brainflow DataFilter-Klasse in Python mithilfe von numpy interpoliert und auf 44100 Samples hochgesampelt. Anschließend kann jeder EEG-Kanal separat mittels PyAudio in einen Audio-Stream geschrieben werden. Das Auslesen und Aufbereiten der EEG-Daten geschieht hierbei in einer Callback-Funktion, die von PyAudio gehandhabt wird. Dies ermöglicht die Unterteilung in kleinere Daten-Chunks, da PyAudio immer dann neue Daten anfordert, sobald ein Daten-Chunk in den Audiostream geschrieben wurde. Über Blackhole kann das Audiosignal dann zu SuperCollider geroutet werden.

Dieses Feature wurde bereits im vollen Umfang getestet, bot jedoch keine hörbar bessere Alternative zum Versenden der Daten per OSC und dem Aufbau des Audiosignals in SuperCollider. Vermutlich ist die Sample-Rate von 200 Hz zu gering, um den Paketverlust von OSC negativ zu beeinflussen. Da die Daten zur Weiterverarbeitung ohnehin schon mit der geringen Frequenz von 200 Hz nach SuperCollider versendet werden, würde es mehr Sinn machen, Audio direkt in SuperCollider zu schreiben. Bei einer direkten Sonifikation könnte man die Daten beispielsweise mit dem `BufWr`-UGen in einen Audiobuffer schreiben. Hier hätte man noch zusätzlich einige Optionen zur Interpolation.

Weitere Datenfunktionen Die bisher implementierten Verarbeitungsfunktionen in der `ProxyChain` bilden eine grundlegende Basis zum Verarbeiten und Mapping von Inputdaten. Für Zukünftige Projekte wäre es förderlich, wenn sich der Pool an Funktionen noch Erweitern könnte. Insbesonders sei hier "Composing Interactions" von Marije Baalman genannt (Baalman 2022), welches auch als Inspirationsquelle für EEGsuperMap diente. In diesem Buch werden verschiedene Algorithmen zur Datenverarbeitung und -glättung detailliert vorgestellt und miteinander verglichen.

Codebuffer innerhalb der GUI Zwar lässt sich die ProxyChain bereits während der Laufzeit umstrukturieren und neukonfigurieren. Allerdings erfordert dies ein wechsel zwischen Codeeditor und der GUI. Daher wäre es ein

interessantes Feature wenn einzelne Funktionen in der ProxyChainGUI ausgewhält werden können. Daraufhin könnte sich ein Input Fenster öffnen in dem die Funktion abgerufen wird, um diese zu tewaken oder neu zu schreiben. So könnte man schnelle Änderungen am Code vornehmen ohne das Fenster zu Wechseln und zur richtige Zeile zu scrollen.

3.3 Fazit

SuperMap hat gezeigt, dass es durchaus sinnvoll sein kann, Datenverarbeitung auf Audiorate zu betreiben. Ab dem Ausführen der Software findet der Großteil der Datenverarbeitung auf dem SuperCollider-Server statt. Somit wird das Problem der Instabilität durch die Erstellung neuer Prozesse oder Threads bei der Datenverarbeitung in Echtzeit vermieden. Die dynamische Erstellung und Verwaltung von Prozessen kann ansonsten zu unerwarteten Leistungsproblemen und Instabilität führen. Der Kern der Datenverarbeitung von EEGsuperMap läuft daher auf einer konstanten CPU-Nutzung und gewährleistet, dass keine neuen Prozesse zur Laufzeit gestartet werden.

Speziell auf EEGsuperMap bezogen ist dies relevant, da GUI-Prozesse zusätzliche Leistungsanforderungen stellen. Daher ist die Trennung der rechenintensiven Datenverarbeitung auf dem Server von der GUI-Interaktion ein wichtiger Aspekt für die Aufrechterhaltung der Gesamtstabilität und Leistungsfähigkeit der Software.

Allerdings hat sich während der Entwicklung der Umfang der Software derart erweitert, dass sich die Frage stellt, ob SuperCollider für eine vollumfängliche Datenverarbeitungssoftware geeignet ist. Hätte man einige Features zur einfachen Anpassung des Codes außer Acht gelassen, könnte man die Software komplett als robuste Audiolösung in C++ entwickeln, anstatt Python und SuperCollider per OSC zu verbinden. Doch so sehr die Aufteilung auf beide Entwicklungsumgebungen Komplexität innerhalb der Softwarestruktur bringt, schafft sie wiederum Ansatzpunkte innerhalb des Datenverarbeitungsprozesses. So bleibt EEGsuperMap an vielen Stellen offen, um eigene Verarbeitungsfunktionen zu testen. Darüber hinaus lassen sich sowohl die Klangsyntese als auch die Datenquelle oder der Sensor komplett austauschen.

Am Ende ist es wie bei vielen digitalen Musikprojekten eine Entscheidung, die vom persönlichen Hintergrund abhängt. Deshalb wurde entschieden, die Flexibilität, einzelne Komponenten auszutauschen, als wichtig genug empfunden, um eigenwillige Softwarelösungen zu entschuldigen. Zur Zeit ist EEGsuperMap ein Tool, das in seiner jetzigen Form auf EEG-Daten optimiert ist. EEG-Features sind aufgrund ihrer Beschaffenheit hochdimensional und erfordern daher eine andere Verarbeitung als der Input eines MIDI-Controllers. Der Aufwand, die Software auch für andere Daten zu verwenden, ist jedoch gering, da lediglich ein anderer Endpunkt für das Empfangen der OSC-Daten geschaffen werden muss. Daher wurden im Hauptteil der Arbeit einzelne Komponente beschrieben, um diese auf eigene Bedürfnisse und Daten anzupassen.

Letztendlich ist EEGsuperMap jedoch nicht nur eine Softwarelösung, son-

dern auch ein Musikinstrument. Daher hat die theoretische Auseinandersetzung mit anderen Projekten sowie fotwährendes anspielen während der Entwicklung auch dazu geführt, dass Parameter aus dem Innenleben des Instruments, in die GUI integriert wurden. Ein Beispiel dafür ist die Verwendung der `FadeTime` als performativer Parameter.

3.4 Grüße und Dank

Mein erster Dank gilt meiner Familie und meinen Freunden, die mich während der Masterarbeit unterstützt haben und damit eigentlich den größten Teil der Arbeit geleistet haben. Ich bedanke mich insbesondere bei Marc Bangert und Michele Samarotto. Ohne deren Unterstützung und Input wäre EEGsuperMap nicht realisierbar gewesen. Dank gilt auch Sierk Schnalzriedt für die Visualisierung der Softwarestruktur, deren Gliederung und die Ordnung meiner Gedanken.

Bibliography

Baalman, Marije (2022). *Composing Interactions. An Artists Guide to Building Expressive Interactive Systems*. English. Includes 256 images, 3 color print. Superposition, p. 608. ISBN: 978-90-828935-4-0. URL: https://store.v2.nl.

Bera, Tushar Kani (2015). "Noninvasive Electromagnetic Methods for Brain Monitoring: A Technical Review". In: *Brain-Computer Interfaces*. Ed. by A. E. Hassanien and A. T. Azar. Department of Computational Science and Engineering, Yonsei University, Seoul 120749, South Korea. Schweiz: Springer International Publishing, pp. 51–85.

Bruce N. Walker, Michael A. Nees (2011). "Theory of Sonification". In: *Sonification Handbook*. Logos Publishing House. Chap. 2.

Campo, Alberto de (2009). "Science By Ear. An Interdisciplinary Approach to Sonifying Scientific Data". PhD thesis. Institute of Electronic Music, Acoustics - IEM, University for Music, and Dramatic Arts Graz.

Florian Grond, Jonathan Berger (2011). "Parameter Mapping Sonification". In: *Sonification Handbook*. Logos Publishing House. Chap. 15.

Fouad, Mohamed Mostafa et al. (2015). "Brain Computer Interface: A Review". In: *Brain-Computer Interfaces*. Ed. by A.E. Hassanien and A.T. Azar. Schweiz: Springer International Publishing, pp. 3–30.

Hofstadter, K. (2024). *OpenBCI-SuperCollider*. GitHub repository. URL: https://github.com/khofstadter/OpenBCI-SuperCollider.

Magnusson, Thor (2006). "Affordances and Constraints in Screen-Based Musical Instruments". PhD thesis. Creative Systems Lab, Dept. of Informatics University of Sussex.

— (2019). "Ergodynamics And A Semiotics of Instrumental Composition". In: *Tempo* 73.41.

OpenBCI (2022a). *Cyton Data Format*. URL: https://docs.openbci.com/Cyton/CytonDataFormat/ (visited on 01/2024).

— (2022b). *Ganglion Data Format*. URL: https://docs.openbci.com/Ganglion/GanglionDataFormat/ (visited on 01/2024).

— (2022c). *OpenBCI Wifi*. URL: https://docs.openbci.com/ThirdParty/WiFiShield/WiFiLanding/ (visited on 05/2024).

Stangl, W. (2021). *Stichwort: 'Pyramidenzellen – Online Lexikon für Psychologie und Pädagogik'*. Abgerufen am 31.03.2024. URL: https://lexikon.stangl.eu/5826/pyramidenzellen.

, thor magnusson thor (2019). *sonic writing - technologies of material. symbolic and signal inscrioptions*. Bloomsbury Publishing Inc. ISBN: 978-1-5013-1385-1.

Till Bovermann, Julian Rohrhuber Alberto de Campo (2011). "Laboratory Methods for Experimental Sonification". In: *Sonification Handbook*. Logos Publishing House. Chap. 10.

Zaitcev, Aleksandr et al. (2015a). "Source Localization for Brain-Computer Interfaces". In: *Brain-Computer Interfaces*. Springer International Publishing.

— (2015b). "Source Localization for Brain-Computer Interfaces". In: *Brain-Computer Interfaces*. Ed. by A.E. Hassanien and A.T. Azar. Schweiz: Springer International Publishing, pp. 125–154.