RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

HOCHSCHULE HEILBRONN
TECHNIK WIRTSCHAFT INFORMATIK

Bachelorarbeit

Drahtlose PC-Anbindung des Muskelstimulators MotionStim8 und Realisierung einer Steuersoftware zur Kopplung mit dem Thalmic MYO Armband

Autorin:	Corinna Simon
Studiengang:	Medizinische Informatik Ruprecht-Karls-Universität Heidelberg / Hochschule Heilbronn
Matrikelnummer:	185370
Abgabe:	10. März 2017
Referent:	Prof. Dr. Rolf Bendl
Korreferent:	Dr. Christoph Maier

Zusammenfassung

In dieser Arbeit wird eine Software entwickelt, um das Koppeln des Thalmic MYO-Armbands mit dem Stimulator MotionStim8 zu vereinfachen. In einer vorherigen Arbeit [15] wurden für die Kopplung MATLAB Skripte benutzt. Diese sollen durch die Software vollständig ersetzt werden. Außerdem soll der Stimulator eine drahtlose Anbindung über zwei British Broadcasting Corporation (BBC) micro:bit Systeme erhalten.

Dazu wird zuerst eine Anforderungsanalyse durchgeführt, woraus konkrete UseCases der Software formuliert werden konnten. Hierzu werden aus der Problemstellung und den daraus resultierenden Zielen dieser Arbeit Funktionale und nicht-Funktionale Anforderungen extrahiert. Es müssen Gesten, sowie deren Myo- und Stimulations-Daten verwaltet werden können. Auch müssen sowohl Myo-Armband als auch MotionStim8 drahtlos angebunden, sowie gekoppelt werden.

Nachdem die Anforderungen feststehen, wird ein Konzept zur Umsetzung entwickelt. Dazu wird eine Entwicklungsumgebung gewählt und eine Softwarearchitektur ausgearbeitet. Es wird ein Model-View-Controller (MVC)-Modell angestrebt. Für die Entwicklungsumgebung der Steuersoftware wird VisualStudio, bzw. C#, und für die drahtlos Anbindung Mu, bzw. MicroPython, verwendet. Auch wird eine Übersicht über die Komponenten angefertigt. Daraus sind die Hardware-Schnittstellen, PC zu micro:bit, micro:bit zu micro:bit, micro:bit zu MotionStim8, sowie Myo-Armband zu PC, ableitbar. Für das Myo-Armband existieren bereits Software-Bibliotheken, welche eingebunden werden können und somit das Entwickeln einer eigenen Schnittstelle nicht notwendig machen. Zum Schluss der Konzeption werden Ideen für eine Benutzeroberfläche erarbeitet. Dafür werden die Nicht-Funktionalen Anforderungen, modularer Aufbau sowie Benutzerfreundlichkeit und Übersichtlichkeit, aus der Anforderungsanalyse aufgegriffen. In dieser Phase werden bereits erste MockUps erstellt.

Die Implementierung findet in zwei Schritten statt.

Die Drahtlosfunktion, bzw. der Python-Teil, kann separat entwickelt werden. Dafür werden die seriellen Schnittstellen, PC zu micro:bit und micro:bit zu MotionStim8, sowie die Übertragung zwischen den beiden micro:bits implementiert. Außerdem werden die Befehle im micro:bit an die benötigte Command Structure des Stimulators angepasst.

Die Steuersoftware beinhaltet eine Graphical User Interface (GUI) mit modulübergreifenden Funktionen. Die einzelnen Module, Myo Controller, Stimulation Controller und Gesture Controller besitzen ihre eigene Oberfläche und können in die GUI modular integriert werden. Die Gesten-, Myo- und Stimulations-Daten können über das jeweilige Modul verwaltet werden. Über das Beenden der Anwendung hinaus werden alle Daten in einer Extensible Markup Language (XML)-Datei gespeichert. Die Kopplung des Myo-Armbands mit dem MotionStim8 findet über einen Mapper statt, welcher eine Mapping-Matrix für eine Geste errechnet. Dafür müssen vorab Myo- und Stimulations-Daten für

jede Position dieser Geste gesetzt werden. Die Echtzeitfunktion wird übergeordnet über die GUI gestartet und berechnet für eingehende Myo-Daten die Pulsweiten für die Stimulation. Für diese Berechnung wird die Mapping-Matrix benötigt.

Hiernach kann die entwickelte Steuersoftware für die definierten Anforderungen, bzw. UseCases, verwendet werden.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich bei meiner Bachelorarbeit unterstützt haben.

Dabei danke ich Allen voran zuerst Dr. Christoph Maier, welcher diese Arbeit betreut, mich bei dem technischen Teil der Arbeit sehr unterstützt und mir immer wieder mit Ideen weitergeholfen hat.

Ebenfalls möchte ich mich bei meinem Kommilitonen und Freund Steffen Fröhlich für das Korrekturlesen meiner Arbeit und für einige nützliche Tipps über selbige bedanken.

Besonderer Dank gilt meiner Familie, speziell meinem Vater Rainer Simon, der mir meinen Studiengang vorgestellt und mein Studium ermöglicht hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

Codeverzeichnis

Abkürzungsverzeichnis

BBC British Broadcasting Corporation

BCI Brain-Computer-Interface

CID Command ID

COM Component Object Model

DL Data Lenght

EEG Elektroenzephalografie

EMG Elektromyographie

EMS Elektrische Muskelstimulation

GUI Graphical User Interface

LED Light-emitting diode

MEG Elektromyografie

MVC Model-View-Controller

PC Personal Computer

RAM Random-Access Memory

SDK Software Development Kit

SOF Start of Frame

UART Universal Asynchronous Receiver Transmitter

USB Universal Serial Bus

XML Extensible Markup Language

1. Einleitung

Muskeln sind kontraktive Organe des Körpers. Ihre Aufgabe ist es sich zusammenzuziehen und zu strecken [11]. Was bei gesunden Menschen ihr eigener Körper managen kann, kann bei Querschnittsgelähmten teilweise von Elektrostimulation übernommen werden.

In der Neurologischen Klinik Bad Aibling wird 2007 an Patienten die elektrische Reizung von Muskeln erprobt. Diese Patienten sind alle mindestens ab dem Becken abwärts vollständig gelähmt. Der Stromimpuls lässt die Muskeln ihre ursprüngliche Funktion wieder ausführen. Die Beine des Patienten können somit wieder bewegt werden. Dafür werden an jedes Bein acht Elektroden angebracht, welche mit ca. 90 bis 120 Milliampere, die dort sitzenden Muskeln stimulieren. Anstatt der chemischen Botenstoffe, bei einem gesunden Menschen, übernimmt die Muskelreizung hier der Strom. Wichtig hierbei ist, dass die Muskelnerven und Muskeln des Probanden noch intakt sein müssen. Das Training der Muskeln ist, aufgrund der zu großen Maschinen, jedoch für die Patienten nur in der Klinik möglich. [26]

Dass das Thema immer noch aktuell ist zeigt eine Forschung eines Teams der Berliner Charité von 2014. Diese brachten durch Elektrostimulation querschnittsgelähmte Ratten wieder zum laufen. [12] [25]

Diese Arbeit beschäftigt sich mit der Übertragung von Muskelreizen, bzw. mit Abfolgen von Muskelreizen, wodurch Gesten zustande kommen. Dafür wird das Thalmic Myo-Armband [17] und der Stimulator MotionStim8 [14] gekoppelt. Dass dies für einfachere Gesten möglich ist wurde bereits in einer vorangegangenen Arbeit [15] gezeigt. Die Grundlagen der Arbeit wurden dabei in MATLAB programmiert. Dabei muss vor jeder Ausführung eine Kalibrierung ausgeführt werden, welche mit Hilfe einer Skizze auf einem Blatt Papier durchgeführt wurde. Dies ist soweit nicht benutzerfreundlich und aufwendig.

Im Folgenden wird eine Software entwickelt, um die Kopplung des Myo-Armbands und des MotionStim8 zu vereinfachen. Außerdem soll durch eine drahtlos Anbindung des Stimulators an den PC mehr Bewegungsfreiheit ermöglicht werden.

1.1. Problemstellung

In einer vorausgehenden Arbeit [15] wurde die prinzipielle Möglichkeit der Elektrostimulation einfacher Handgesten durch Kopplung des Thalmic Myo-Armbands mit dem Stimulator MotionStim8 gezeigt. Hierfür wurde bisher hauptsächlich mit MatLab gearbeitet. Um dieses Vorgehen zu vereinfachen und benutzerfreundlicher zu gestalten, soll in dieser Arbeit eine Software für die Kopplung und Konfiguration des Thalmic Myo-Armbands und des Stimulators MotionStim8 entwickelt werden. Auch soll der Stimulator MotionStim8 drahtlos über den PC gesteuert werden können, dadurch muss sich der Proband nicht mehr direkt neben dem PC befinden.

1.2. Ziele

Das Ziel der Arbeit ist eine benutzerfreundliche Software zur Verwaltung von Gesten und deren Elektromyographie (EMG) und Stimulator-Daten. Damit die Software mit Erweiterungen ergänzt werden kann, muss der Aufbau modular erfolgen. Es sollen unterschiedliche Gesten behandelt und mit mehreren MYO- und Stimulator-Datensätzen zusammen gespeichert werden können. Dafür sollen die Daten des MYO-Armbands in der Software ausgelesen, angezeigt und einer Geste zugewiesen werden können. Auch Stimulationsparameter sollen dargestellt werden und veränderbar sein. Außerdem soll es möglich sein Stimulationsparameter vom PC aus zu kontrollieren. Hierfür muss eine Drahtlose Anbindung des MotionStim8 über zwei BBC micro:bit Systeme realisiert werden.

2. Grundlagen

In diesem Kapitel wird auf die Grundlagen dieser Arbeit eingegangen. Neben der vorgestellten Hardware, dem MotionStim8, dem Myo-Armband, sowie dem micro:bit, wird Grundlegendes über die Elektrostimulation und die Elektromyographie erläutert. Des Weiteren werden in diesem Abschnitt die verwendeten Programmiersprachen, Entwicklungsumgebungen und Software-Bibliotheken kurz beschrieben.

2.1. MotionStim8 - Elektrostimulation

Die Elekrostimulation [10] wird vor allem im Sport, bei Elektrischem Muskelstimulations (EMS)-Training und der Rehabilitation, sowie bei muskuloskelettalen Schmerzen, eingesetzt. Hierbei werden gezielt Muskeln oder Nerven gereizt.

Abbildung 2.1: Der MotionStim8 der Firma Krauth + Timmermann [14]

Der MotionStim8 [14] ist ein Elektrostimulator mit acht Kanälen und wird sowohl im ambulanten als auch in klinischen Bereich eingesetzt. Da mit dem MotionStim8 1, 2, 4 oder alle 8 Kanäle genutzt werden können, ist er flexibel einsetzbar und bietet eine sehr große Anwendungsvielfalt. Der MotionStim8 ermöglicht zwei verschiedene Betriebsarten, den Patientenbetrieb und den Programmierbetrieb.

Über eine serielle Schnittstelle lässt sich das Gerät extern steuern. Es lassen sich Stromstärke (zwischen 1-125mA), Frequenz (zwischen 1-99Hz) und Impulsbreite (zwischen 10-500μsec) für jeden Kanal einstellen.

2.2. Thalmic Myo-Armband - Elektromyographie

Die Elektromyographie (EMG) [9] misst die elektrische Spannung in einem Muskel und ist eine Untersuchungsmethode aus der Neurobiologie, um bspw. zu Überprüfen, ob eine Erkrankung des Muskels vorliegt.

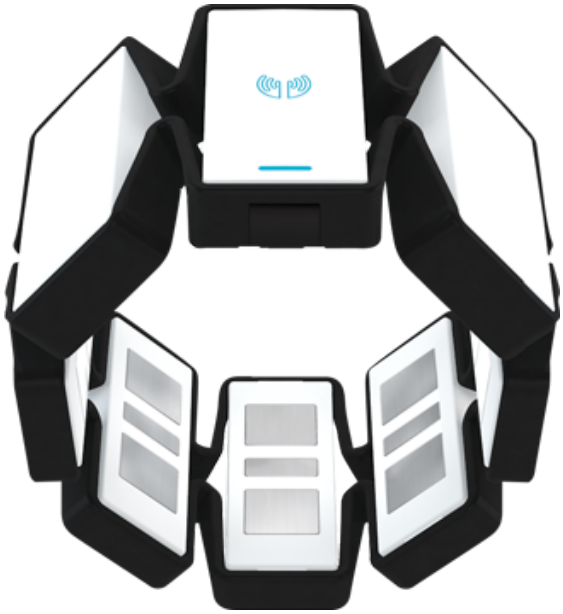


Abbildung 2.2: Das Thalmic Myo-Armband [17]

Das Thalmic Myo-Armband [17] ist ein Gesten-Control-Armband. Es ist mit acht Sensoren ausgestattet, welche die elektrischen Signale der Muskeln messen (EMG) und aufgrund dessen auch vordefinierte Gesten erkennen können. Das Gerät wird über einen Bluetooth 4.0 Adapter mit dem Computer verbunden und ist mit Betriebssystemen ab Windows 7 kompatibel.

Mittels Myo-Connect [18], einem Programm von Thalmic welches das Myo-Armband mit dem PC verbindet und bereits einige Steuerfunktionen bietet, können Anwender die Gestensteuerung direkt für die aktuell fünf erkennbaren Gesten, "Fist", "Wave Left", "Wave Right", "Fingers Spread", "Double Tab"nutzen.

Für Entwickler liefert Thalmic außerdem ein C++ Software Development Kit (SDK) [19], welches Funktionen für die Bluetooth Verbindung und die Datenübertragung liefert. Dieses SDK kann jeder über die Thalmic Website herunterladen und mit dessen Hilfe eigene Scripts für das Myo-Armband schreiben.

2.3. BBC micro:bit

Der BBC micro:bit [4] ist ein auf ARM basierendes Embedded-System, welches von der BBC in Großbritannien für Schüler entwickelt wurde. Schüler sollten anhand des micro:bit spielend leicht das Programmieren lernen.

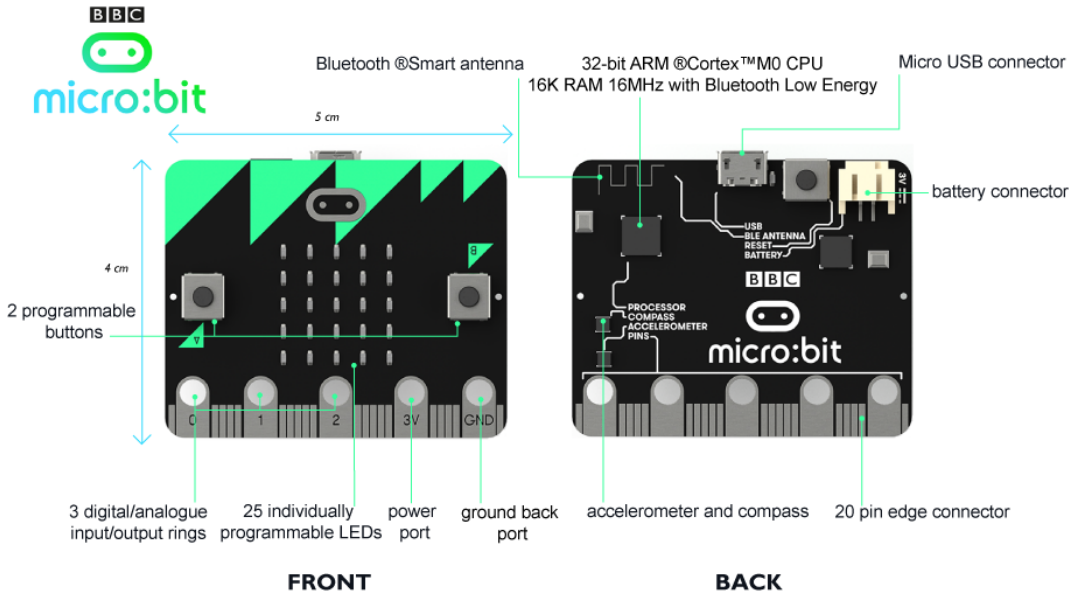


Abbildung 2.3: Die Hardware des BBC micro:bit [4]

Der BBC micro:bit [13] verfügt über eine 5x5 Light-emitting diode (LED)-Matrix, zwei programmierbaren Buttons sowie einer Bluetooth-Antenne und einer Universal Serial Bus (USB) Steckverbindung zum Verbinden des Gerätes mit dem PC oder dem Smartphone. Außerdem hat der micro:bit einen 3-Achsen-Beschleunigungssensor und ein Magnetometer. Des Weiteren besitzt der micro:bit 20 Input/Output-Pins, 6 davon analog.

Um Skripte für den micro:bit entwickeln zu können, werden einige Programmieroberflächen [6] angeboten. Microsoft PXT, MicroPython, Microsoft Block Editor, Code Kingdoms JavaScript und Microsoft Touch Develop sind direkt über die BBC micro:bit Website nutzbar.

2.4. Mu - MicroPython

MicroPython

MicroPython [8] ist eine speziell für Micro-Controller und ähnliche Umgebungen entwickelte und optimierte Version von Python. Sie basiert auf Python 3 und enthält nur einen Teil der Python-Standard-Library. Sie beinhaltet dennoch komplexere Funktionen, wie Exception handling und Listenverwaltung. Trotzdem ist MicroPython kompakt genug, um mit nur 256kB Code Speicherplatz auszukommen und mit nur 16kB Random-Access Memory (RAM) zu funktionieren.

Mu

Mu [16] ist ein sehr einfach gehaltener Code-Editor, um MicroPython Scripts für den BBC micro:bit zu schreiben. Er kann das geschriebene Script direkt auf den angeschlossenen micro:bit flashen, damit dieses dort sofort ausgeführt werden kann.

2.5. Visual Studio 2015 - C#

C#

C# [21] [1] ist eine Systemprogrammiersprache von .NET. Somit können viele Softwarebibliotheken die .NET beinhaltet direkt in C# eingebunden und genutzt werden. C# ist eine Objektorientierte Programmiersprache und gehört zu der Sprach-Familie C/C++, enthält jedoch auch Teile aus den Programmiersprachen Delphi und Java.

Visual Studio 2015

Visual Studio 2015 [23] ist eine, von Microsoft vertriebene, Entwicklungsumgebung für verschiedene Programmiersprachen. Zu diesen gehören Visual Basic, C++, C, C#, JavaScript und einige mehr. Mit diesen kann Code für iOS-, Windows- und Android-Anwendungen erstellt werden. Außerdem gibt es eine große Menge an optionalen Paketen, um die Funktionen von Visual Studio 2015 zu erweitern und genauer an die eigenen Bedürfnisse anzupassen.

2.5.1. FieldTrip

FieldTrip [27] ist eine Open Source Software zur Analyse von Elektromyografie (MEG), Elektroenzephalografie (EEG) und anderen elektrophysiologischer Daten. Die Software wurde in MATLAB, einem Programm für Daten-Analyse und -Visualisierung, implementiert. Die FieldTrip Toolbox besteht aus fast 1000 Funktionen, deren Hauptaugenmerk auf der Analyse elektrophysiologischer Daten liegt.

Zudem enthält FieldTrip ein Echtzeit-Modul, dessen Kernkomponente der FieldTrip-Buffer ist. Ein multithreaded Netzwerk, welches dem Client erlaubt Daten in kleinen Blöcken zu streamen und gleichzeitig die Datenanalyse in MATLAB durchzuführen.
Dieses Modul ermöglicht es dem Benutzer BCI Systeme zu erstellen.

Bereits in der vorausgehenden Arbeit [15] entwickelte Dr. Christoph Maier, auf Grundlage der FieldTrip Toolbox [27] und dem Myo-SDK [19], ein C++ Programm, welches die EMG-Daten des Thalmic Myo-Armbands in einen FieldTrip-Buffer-Server schreibt. Dieses Programm wird in dieser Arbeit wiederverwendet.

Da es von den Entwicklern von FieldTrip keinen FieldTrip-Buffer in C# gibt muss eine Alternative gefunden werden.

2.5.2. Buffer BCI

Der Buffer BCI [3] ist ein Plattform- und Sprachunabhänges Framework zum Bau von BCI-Schnittstellen. Hierbei können Clients Daten auf dem Server ablegen und auf die Daten und Events auf selbigem zugreifen. Der Server basiert auf dem FieldTrip-Buffer-Server. Zudem liefert der Buffer BCI für C#, Java, Python und C Funktionen, um auf Daten und Events, welche auf dem Server liegen, zugreifen zu können. Für Octave und MATLAB liefert der Buffer BCI weiter reichenden Support.

Mittels dem in Buffer BCI enthaltenem Buffer für C# sollte somit auf den von Dr Christoph Maier erstellten FieldTrip-Buffer-Server zugegriffen werden können.

2.5.3. Math.NET Numerics

Die Numerics Library [30] ist eine in C# entwickelte Sammlung von einer Vielzahl von Methoden und Algorithmen von numerischen Berechnungen. Lineare Algebra, Interpolation und Integration sind nur ein Teil davon. Numerics ist Teil der Math.NET initiative [29], einer Open-Source-Initiative die sich mit Toolkits rund um Mathematik beschäftigt.

3. Anforderungsanalyse

In diesem Kapitel werden die Ziele, siehe Kapitel 1.2 Seite 2, in Anforderungen an das Projekt umformuliert. Zusätzlich werden anhand dieser einige UseCases für das Programm entworfen.

3.1. Funktionale Anforderungen

Realisierung eines Gestenmanagers

Als Grundlage der Anwendung dient ein Gestenmanager, zum Speichern von Gesten, mit ihrer Bezeichnung, einer Beschreibung und fünf Positionen der Geste. Hierfür müssen Gesten hinzugefügt, gelöscht und bearbeitet werden können. Des Weiteren müssen Myo- und Stimulations-Daten der jeweiligen Gestenposition zugewiesen werden können.

Speichern der Daten

Um alle im Gestenmanager enthaltenen Daten auch nach dem Beenden des Programms zu erhalten, müssen diese in einer Datenbank, bzw. einer XML-Datei, abgespeichert werden. Beim Start der Anwendung können alle Daten anhand dieser Datei wieder eingelesen werden.

Anbindung des MotionStim8

Setzen der Stimulationsparameter
In der Anwendung wird ein Modul benötigt, um die Stimulationsparameter setzen zu können. Hierbei handelt es sich um acht Kanäle jeweils mit Amplitude, Frequenz und Pulsweite. Insgesamt also 24 Regler, welche dargestellt werden müssen. Außerdem soll es für die jeweils acht Regler einer Art eine Kopplungsfunktion geben, um diese gleichzeitig auf den selben Wert einstellen zu können.

Senden der Stimulationsparameter über die BBC micro:bits an den MotionStim8
Die Stimulationsparameter müssen über eine Serielle Schnittstelle(USB) dem 'Sender' micro:bit übergeben werden. Dieser schickt die Werte direkt weiter an den 'Empfänger' micro:bit und übergibt dem MotionStim8 ebenfalls über eine Serielle Schnittstelle(Pins) die, zu einem für den MotionStim8 gültigen Befehl veränderten, Stimulationsparameter. Um eine Echtzeit-Funktion der Kopplung ermöglichen zu können ist es essentiell, dass dieser Vorgang schnell abläuft.

Anbindung des Thalmic Myo-Armbands

Lesen der Myo-Daten in Echtzeit
Um das Thalmic Myo-Armband in die Anwendung einzubinden muss mittels eines FieldTrip-Buffers auf den FieldTrip-Buffer-Server (siehe Kapitel 2.5.1 Seite 6) zugegriffen werden.
Myo-Daten anzeigen
Die Myo-Rohdaten werden in einem regelmäßigen Abstand ausgelesen und ein Mittelwert für jeden der acht ausgelesenen Kanäle bestimmt. Eine genaue Darstellung der Myo-Daten kann mittels eines Diagramms realisiert werden, welches den Mittelwert jedes Kanals anzeigt. Dieses wird im gleichem Abstand aktualisiert wie die Rohdaten ausgelesen werden.
Aufnehmen/Abspielen der Daten
Um Myo-Daten speichern zu können wird eine Aufnahme-Funktion benötigt. Hierfür werden die Mittelwerte der Kanäle über einen bestimmten Zeitraum gesammelt und können dann einer Gestenposition zugewiesen werden.

Realisierung der Kopplung von MotionStim8 und Myo-Armband

Mapping
Durch das Mapping wird die Kopplung des Myo-Armbands mit dem MotionStim8 realisiert. Hierbei werden die Myo-Daten und Stimulationsparameter für alle fünf Positionen einer Geste benötigt. Die Berechnung der Mapping-Matrix kann erst bei vollständigen Daten gestartet werden. Sie berechnet eine Matrix, welche für die Echtzeit-Funktion benötigt wird. Damit dort die Berechnung der Stimulationsparameter erfolgen kann.
Loop
Der Loop entspricht der Echtzeit-Funktion der Anwendung. Er berechnet für ankommende Myo-Daten, anhand der im Mapping erstellten Matrix, passende Stimulationsparameter. Diese müssen dann an den Stimulator geschickt werden.

3.2. Nicht-Funktionale Anforderungen

Modularer Aufbau

Um die Anwendung erweitern zu können, wird ein modularer Aufbau angestrebt. Später soll dadurch auch das Target, aktuell der MotionStim8, oder die Source, aktuell das Myo-Armband, gewechselt werden können. Dafür soll nur das jeweilige Modul mit dem des neuen Targets, bzw. der neuen Source, getauscht werden müssen. **Benutzerfreundliche und übersichtliche GUI**

Da alleine der Stimulator 24 Regler benötigt, um alle Kanäle abzudecken, ist ein übersichtlicher Aufbau der GUI wichtig, damit der Benutzer nicht überfordert wird. Auch die verschiedenen Module sollen für den Benutzer klar ersichtlich sein.

3.3. UseCases

3.3.1. Hinzufügen von Gesten

UseCase-Name:	Hinzufügen von Gesten
Subsystem:	Steuersoftware
Akteure:	User, GestureController, XML-Datei
Kontext und Vorbedingungen:	Programm wurde gestartet

Normal-Ablauf:

1. Der User wählt die Funktion 'Geste hinzufügen'
2. Der User kann nun
   - die Bezeichnung und
   - die Beschreibung
   - die Schmerzgrenze

   der neuen Geste eingeben
3. Der User bestätigt seine Eingaben
4. Das System meldet: 'Die Geste wurde erfolgreich erstellt'

Normal-Ergebnis:
Es wurde erfolgreich eine neue Geste erstellt.

Alternativ-Ablauf:
User bricht den Vorgang ab
Anstatt zu Bestätigen kann der User zu jeder Zeit auch abbrechen.

a Der User wählt die Funktion 'abbrechen'

b Die aktuelle neue Geste, die aktuellen Änderungen oder die zu löschende Geste wird auf den Ausgangszustand zurückgesetzt.

Alternativ-Ergebnis:
Der Vorgang wurde abgebrochen.

3.3.2. Löschen/Ändern von Gesten

UseCase-Name:	Löschen von Gesten
Subsystem:	Steuersoftware
Akteure:	User, GestureController, XML-Datei
Kontext und Vorbedingungen:	Programm wurde gestartet und es existiert bereits mindestens eine Geste

Normal-Ablauf:

1. User wählt die zu löschende Geste aus
2. User wählt die Funktion 'Geste löschen'
3. User bestätigt das Vorhaben
4. System meldet: 'Die Geste wurde erfolgreich gelöscht'

Normal-Ergebnis:
Eine ausgewählte Geste wurde gelöscht.
Alternativ-Ablauf:
Geste ändern

1. Der User wählt die zu ändernde Geste aus
2. Der User wählt die Funktion 'ändern'
3. Der User kann nun
   - die Bezeichnung und
   - die Beschreibung
   - die Schmerzgrenze

   der Geste ändern
4. Der User bestätigt seine Eingaben
5. Das System meldet: 'Die Geste wurde erfolgreich geändert'

Alternativ-Ergebnis:
Eine ausgewählte Geste wurde geändert.

User bricht den Vorgang ab
Anstatt zu Bestätigen kann der User zu jeder Zeit auch abbrechen.

a Der User wählt die Funktion 'abbrechen'

b Die aktuelle neue Geste, die aktuellen Änderungen oder die zu löschende Geste wird auf den Ausgangszustand zurückgesetzt.

Alternativ-Ergebnis:
Der jeweilige Vorgang wurde abgebrochen.

3.3.3. Anzeigen/Aufnahme der MYO-Daten

UseCase-Name:	Anzeigen der MYO-Daten
Subsystem:	Steuersoftware
Akteure:	User, MyoController, Myo-Armband
Kontext und Vorbedingungen:	Programm wurde gestartet und das Thalmic Myo-Armband ist mit dem Computer verbunden

Normal-Ablauf:

1. Der User wählt die Funktion 'Start Myo-Armband'
2. Das System zeigt die Myo-Daten an
3. Der User wählt die Funktion 'Stop Myo-Armband'
4. Das System zeigt keine Myo-Daten mehr an

Normal-Ergebnis:
Die Myo-Daten wurden erfolgreich angezeigt.

Alternativ-Ablauf:
Aufnahme der Myo-Daten

3.a Der User wählt die Geste und deren Position zu welcher er Myo-Daten erfassen möchte

4.a Der User wählt die Funktion 'Aufnahme starten'

5.a Das System nimmt die Myo-Daten auf

6.a Der User wählt die Funktion 'Aufnahme beenden'

7.a Das System speichert die Myo-Daten für die ausgewählte Geste und Position ab

8.a Das System meldet 'Die Myo-Daten wurden erfolgreich gespeichert'

Alternativ-Ergebnis:
Myo-Daten wurden aufgenommen und Gesten- sowie Positionsabhängig gespeichert.

3.3.4. Speichern/Anzeigen von Stimulationsparametern

UseCase-Name:	Speichern von Stimulationsparametern
Subsystem:	Steuersoftware
Akteure:	User, StimController, GestureController
Kontext und Vorbedingungen:	Programm wurde gestartet

Normal-Ablauf:

1. Der User wählt eine Geste und Position aus
2. Der User stellt die gewünschten Stimulationsparameter für alle Kanäle
   - Amplitude
   - Frequenz
   - Pulsbreite

   ein.
3. Der User wählt die Funktion 'Speichern'
4. Das System bestätigt, dass die Stimulationsparameter gespeichert wurden.

Normal-Ergebnis:
Die Stimulationsparameter wurden erfolgreich Gesten- und Positionsabhängig gespeichert.

Alternativ-Ablauf:
Anzeigen bereits vorhandener Stimulationsparameter

2.a Der User wählt die Funktion 'Stimulationsparameter anzeigen'

3.a Das System zeigt die Stimulationsparameter an

Alternativ-Ergebnis:
Die Stimulationsparameter wurden erfolgreich angezeigt.

3.3.5. Mapping der MYO-Daten auf die Stimulationsparameter

UseCase-Name:	Mapping der MYO-Daten auf die Stimulations-parameter
Subsystem:	Steuersoftware
Akteure:	User, Mapper, GestureController
Kontext und Vorbedingungen:	Programm wurde gestartet und es wurden für, die zu mappende Geste, Myo- und Stimulations-Daten für alle fünf Positionen gespeichert.

Normal-Ablauf:

1. Der User wählt die zu mappende Geste
2. Der User wählt die Funktion 'Mapping'
3. Das System gibt aus, dass das Mapping erfolgreich war

Normal-Ergebnis:
Für die ausgewählte Geste wurde erfolgreich eine Mapping-Matrix erstellt und gespeichert.

Alternativ-Ablauf:
Nicht genügend Myo- oder Stimulations-Daten vorhanden

3.a Das System gibt aus, dass das Mapping nicht durchgeführt werden kann.

Alternativ-Ergebnis:
Das Mapping wird abgebrochen.

3.3.6. Stimulation der Muskeln mit dem MotionStim8

UseCase-Name:	Stimulation der Muskeln mit dem MotionStim8
Subsystem:	Steuersoftware, Drahtlos Anbindung des MotionStim8
Akteure:	User, StimController, micro:bit Systeme, MotionStim8
Kontext und Vorbedingungen:	Programm wurde gestartet, beide micro:bit Systeme und der Stimulator sind aktiv und verbunden.

Normal-Ablauf:

1. Der User wählt eine Geste aus
2. Der User lässt sich zu der Geste die Stimulationsparameter anzeigen ODER wählt eigene Parameter
3. Der User wählt die Funktion 'Start Stimulation'
4. Das System startet die Stimulation über die micro:bit Systeme am MotionStim8 mit den gewählten Parametern
5. Der User passt die Stimulationsparameter an
6. Das System passt auch die Stimulation des MotionStim8 mit den veränderten Parametern an

Normal-Ergebnis:
Die Muskeln eines mit dem MotionStim8 verbundenen Probanden werden mit den gewählten Parametern stimuliert.

Alternativ-Ablauf:
/

3.3.7. Echtzeitfunktion

UseCase-Name:	Echtzeitfunktion
Subsystem:	Steuersoftware, Drahtlos Anbindung des MotionStim8
Akteure:	User, StimController, GestureController, MyoController, micro:bit Systeme, MotionStim8, Myo-Armband
Kontext und Vorbedingungen:	Programm wurde gestartet, beide micro:bit Systeme und der Stimulator sind aktiv und verbunden. Außerdem ist das Myo-Armband angelegt und angeschlossen. Des Weiteren muss für die Geste die Mapping-Matrix vorliegen.

Normal-Ablauf:

1. Der User wählt eine Geste aus
2. Der User startet das Myo-Armband über die Funktion 'Start Myo-Armband'
3. Der User wählt die Funktion 'Echtzeitfunktion starten'
4. Das System lädt die Stimparameter in den StimController und startet die Stimulation über die micro:bit Systeme am MotionStim8
5. Der User passt die Stimulationsparameter alle 50ms an die durch die Mapping-Matrix und gemessenen Myo-Daten neu berechneten Pulsweiten an
6. Das System passt auch die Stimulation des MotionStim8 mit den veränderten Parametern an

Normal-Ergebnis:
Die gemessenen Myo-Daten werden in Pulsweiten umgewandelt, welche dann stimuliert werden. Es entsteht eine Echtzeitübertragung der Geste.

Alternativ-Ablauf:
/

4. Konzeption

In diesem Kapitel wird anhand der Anforderungsanalyse(siehe Kapitel 3 Seite 8) ein Konzept zur Entwicklung eines Programmes entworfen, welches die Anforderungen erfüllt. Dazu wird zuerst eine Entwicklungsumgebung, sowohl für die Software als auch für die Drahtlos-Anbindung des micro:bits, festgelegt. Danach wird detaillierter auf den Aufbau der Komponenten und der angestrebten Softwarearchitektur eingegangen. Des Weiteren werden die Schnittstellen zwischen den einzelnen Komponenten und die Benutzeroberfläche beschrieben.

4.1. Entwicklungsumgebung

Mu-Micropython

In dieser Arbeit wurde sich für MicroPython als Programmiersprache für die BBC micro:bits und Mu als Code-Editor entschieden. MicroPython beinhaltet bereits einige nützliche Softwarebibliotheken [32] [33] für unsere Funktionen und bietet, vergleichbar mit JavaScript, eine bessere Performance. Da ein auf der BBC micro:bit Website integrierter MicroPython-Editor nicht aktuell genug ist [6](Stand Dezember 2016), um die für uns relevanten Module verwenden zu können, wurde der empfohlene Einstiegs-Editor Mu [16] verwendet.

VisualStudio 2015 - C#

Die Anbindung an das Thalmic Myo-Armband wurde mit C++ realisiert (siehe Kapitel 2.5.1 Seite 6), jedoch bietet C# mehr Komfort bei der Programmierung. Es müssen bspw. keine Header Dateien erzeugt werden. Auch können im Vergleich zu C++ mehr Grundklassen genutzt werden, welche in C++ selbst erstellt werden müssten.

Da der Leistungsvorteil von C++ gegenüber C# in dieser Anwendung nicht genug ins Gewicht fällt, wird für die Entwicklung der Steuersoftware in dieser Arbeit C# verwendet. Dies auch aufgrund meines Studiums, in dem größtenteils Java gelehrt wurde, wodurch C# einen leichteren Einstieg als C++ bietet.

VisualStudio 2015 wurde gewählt, da es eine empfohlene Entwicklungsumgebung für C# ist und einige zusätzliche Funktionen, bspw. den Designer, bietet, um das Erstellen der Software und vor allem der GUI zu erleichtern.

4.2. Architektur

Die Architektur lässt sich in zwei Teile spalten. Zum Einen in die Komponentenübersicht der Drahtlos-Anbindung des Stimulators und des Myo-Armbands, aus dieser ergeben sich die benötigten Schnittstellen. Zum Anderen in die Softwarearchitektur der Steuersoftware.

4.2.1. Komponentenübersicht

Für die drahtlose PC-Anbindung des MotionStim8 müssen zwei micro:bit Systeme zwischen geschaltet werden. Der 'Sender' micro:bit wird dafür mit dem PC verbunden und der 'Empfänger' mit dem MotionStim8. Die Kommunikation der beiden micro:bits erfolgt dabei drahtlos. (siehe Abb. 4.1)

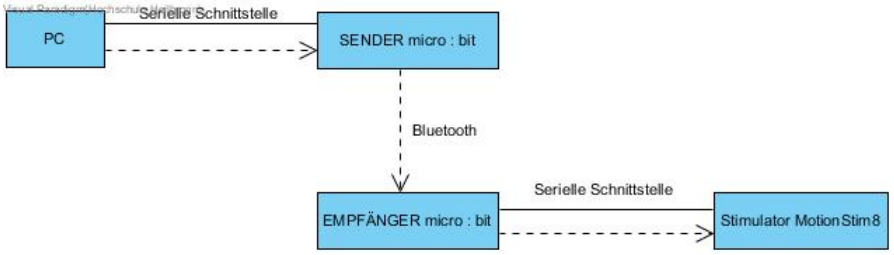


Abbildung 4.1: Aufbau der drahtlosen PC-Anbindung des MotionStim8

Für die Anbindung des Thalmic Myo-Armbands existiert bereits eine Bluetooth Verbindung über einen Bluetooth-Adapter am PC. Diese wird dafür genutzt.

4.2.2. Softwarearchitektur

Für die Steuersoftware wurde sich für eine etwas abgewandelte Form des MVC-Modells (siehe Abb. 4.2) entschieden.

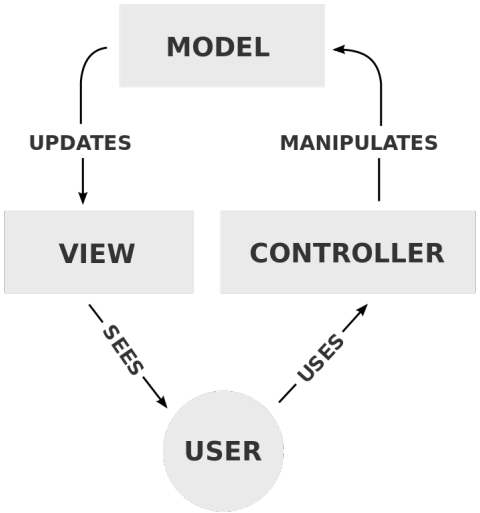


Abbildung 4.2: Das Model-View-Controller Modell [7]

Hierbei können in dieser Arbeit, aufgrund des Designers in VisualStudio, View und Controller nicht klar getrennt werden. Die einzelnen UserControls (StimulationController, MyoController und GestureController) erzeugen jeweils ihre eigene GUI und beherbergen gleichzeitig auch ihre Funktionalität. Auf den Designer wurde in dieser Arbeit allerdings nicht verzichtet, da dieser, durch das Darstellen der aktuellen GUI, das Positionieren von Elementen auf der Oberfläche enorm erleichtert. Zur Verdeutlichung siehe Abb. A.3 auf Seite 49.

4.3. Schnittstellenbeschreibung

In diesem Abschnitt werden die im Kapitel 4.2.1 auf Seite 18 ermittelten Schnittstellen beschrieben.

4.3.1. PC zu micro:bit

Verwendung
Es sollen vom PC Stimulationsdaten an den micro:bit geschickt werden können, um diese von dort aus weitersenden zu können. Eine einseitige Kommunikation ist dafür ausreichend.
Beschreibung
Es handelt sich um eine Serielle Schnittstelle über eine USB-Verbindung. Der BBC micro:bit wird vom PC als COM-Schnittstelle erkannt. Dies kann man über den Gerätemanager des PC's herausfinden.
Besonderheiten
Unverständlicher Weise entsteht auf meinem Computer ein Problem sobald das computereigene Bluetooth aktiviert ist. Dann lassen sich keine COM-Ports mehr öffnen.

4.3.2. micro:bit zu micro:bit

Verwendung
Die Stimulationsdaten müssen drahtlos von einem micro:bit an den anderen gesendet werden.
Beschreibung
Durch die Bluetooth-Antenne des micro:bit Systems und dem Radio-Modul von MicroPython können Daten zwischen zwei micro:bits übertragen werden.
Besonderheiten
/

4.3.3. micro:bit zu MotionStim8

Verwendung
Es soll das Stimulationscommando von dem micro:bit an den MotionStim8 gesendet werden.

Beschreibung
Hierbei handelt es sich ebenfalls um eine Serielle Schnittstelle. Diesmal allerdings über die Pins des micro:bits. Hierfür muss die Verbindung der beiden Geräte gelötet werden. Danach kann der micro:bit über die Pins das Stimmulationscommando an den Stimulator senden.
Besonderheiten
Der BBC micro:bit bereitet bei der Umsetzung Schwierigkeiten, da er die gesendete Bit-Folge invertiert. Dadurch kann der MotionStim8 die Commandos nicht verstehen. Dies kann nur durch die Zwischenschaltung eines Hardware-Inverters gelöst werden.

4.3.4. Myo-Armband zu PC

Verwendung
Es sollen die Rohdaten des Myo-Armbands ausgelesen werden, um weiterverwendet und verarbeitet werden zu können.
Beschreibung
Durch das von Dr. Christoph Maier erstellte Programm (siehe Kapitel 2.5.1 Seite 6) werden die Rohdaten des Myo-Armbands in einen Buffer-Server eingelesen. Diese können dann mittels Buffer BCI in die Software eingelesen werden.
Besonderheiten
Es existiert eine Softwarebibliothek namens MyoSharp [24], welche bereits viel Funktionalität für das Myo-Armband in C# bietet. Diese konnte jedoch aufgrund von Kompatibilitätsproblemen nicht in dieses Projekt eingebunden werden.

4.4. Benutzeroberfläche

Wichtig für die Realisierung der Benutzeroberfläche sind die Nicht-Funktionalen Anforderungen.

Für einen modularen Aufbau des Programms ist es wichtig, dass die GUI aus auswechselbaren Modulen besteht. Somit kann das Target, aktuell der Stimulator, direkt ausgetauscht werden indem das Modul mit einem anderen ersetzt wird. Die Module benötigen dafür ihre eigene Oberfläche und ergeben zusammengesetzt auf der GUI das komplette Programm.

Da Übersichtlichkeit vor allem bei den 24 Reglern des Stimulators sehr wichtig ist, muss ein Weg gefunden werden, diese Darzustellen ohne den Benutzer mit den vielen Reglern zu überfordern.

In einem ersten MockUp (siehe Anhang Abb. A.1 Seite 48) wurde diese Unterteilung durch Module bereits versucht darzustellen. Später wurde ein etwas detaillierteres MockUp (siehe Anhang Abb. A.2 Seite 48) entworfen. In diesem wird bereits dargestellt, dass für den Stimulator einzelne Tabs verwendet werden sollen, um maximal acht Regler auf einmal anzuzeigen.

5. Implementierung

In diesem Kapitel wird die in dem Kapitel 3(Seite 8) und Kapitel 4(Seite 17) entworfene Software umgesetzt. Dazu wurde die Implementierung in zwei Teile getrennt, der Anbindung des MotionStim8 und der Entwicklung der Steuersoftware.

5.1. Drahtlose PC-Anbindung an den MotionStim8

5.1.1. Vorgehensweise

Zuerst wird die Kommunikation der beiden BBC micro:bit Systeme mittels der Radio-Funktion realisiert. Nachdem die beiden miteinander kommunizieren können, wird sich um die seriellen Schnittstellen mittels Universal Asynchronous Receiver Transmitter (UART) gekümmert. Die PC-Anbindung kann dabei direkt über die USB-Verbindung vorgenommen werden. Für die Anbindung des Stimulators MotionStim8 muss dieser zuvor mit den Pins des micro:bit Systems verbunden werden. Das dafür Notwendige löten übernimmt Dr. Christoph Maier.

5.1.2. BBC micro:bit

Die BBC micro:bit Systeme werden unterteilt in Sender und Empfänger. Der Sender ist direkt via USB-Verbindung am PC angeschlossen, während der Empfänger mit dem MotionStim8 verbunden ist.

5.1.2.1. Radio - Drahtlose Netzwerkfunktion

Das Radio-Modul [32] ermöglicht die direkte Kommunikation zwischen micro:bit Systemen. Hierfür muss lediglich die Radio Funktion importiert und angeschaltet werden.

Hiernach können Daten bereits Gesendet und Empfangen werden. Um dies zu Testen, wurde ein Python-Script geschrieben, welches beim Sender mit Drücken des A-Buttons eine Nachricht schickt. Während der Empfänger vorerst dauerhaft auf eine Nachricht wartet und, sofern er eine erhält, diese auf seiner LED-Matrix darstellt.

Code 1: Test der Radio-Funktion - Sender

```
1 import radio
2 from microbit import *
3
4 radio.on()
5
6 while True;
7   if button_a.was_pressed():
8     radio.send('Put your message here');
```

Code 2: Test der Radio-Funktion - Empfänger

```
import radio
from microbit import *

radio.on()

while True:
    incoming = radio.receive()
    if incoming:
        display.scroll(incoming)
```

5.1.2.2. UART - Serielle Schnittstelle

Das UART-Modul [33] ermöglicht dem micro:bit die Kommunikation mit Geräten über eine serielle Schnittstelle. Hierfür muss die UART-Funktion initialisiert werden. Es können die Baudrate, die Länge der übertragenen Bytes, die Parität, sowie die benutzten 'tx' und 'rx' Pins festgelegt werden.

```
uart.init(baudrate=9600, bits=8, parity=None, stop=1, *, tx=None, rx=None)
```

Der BBC micro:bit unterstützt Baudrates von 9600 bis zu 115200 und dies ist der einzige Parameter der gesetzt werden muss, um eine UART-Verbindung aufzubauen. Die Länge der übertragenen Bytes ist standardmäßig auf Acht festgelegt. Soll die Parität überprüft werden, kann diese auf ODD oder EVEN festgelegt werden. Als 'tx' und 'rx' Parameter können beliebige der 20 Pins gewählt werden, siehe Abb. 5.1.

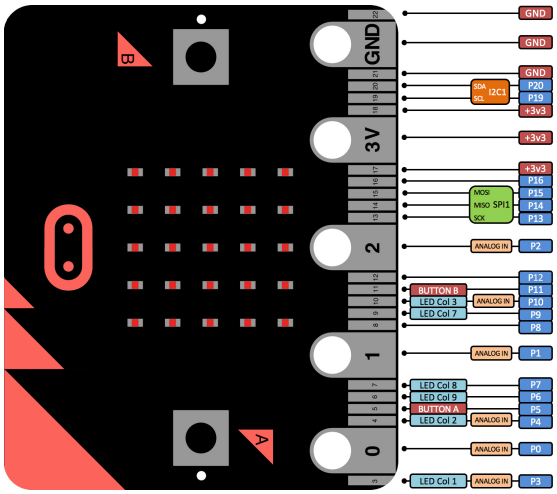


Abbildung 5.1: Die Pinbelegung des BBC micro:bit [5]

Sind die 'tx' und 'rx' Pins bei der Initialisierung nicht angegeben, werden die internen USB-UART 'tx'/'rx' Pins verwendet, womit man mit der USB-Schnittstelle des micro:bit Systems kommunizieren kann.

5.1.3. Kommunikation mit dem BBC micro:bit

5.1.3.1. PC-Anbindung

Die PC-Anbindung mit dem Sender wird mit UART und einer Baudrate von 115200 initialisiert. Da der PC via USB mit dem micro:bit verbunden ist muss kein 'tx'/'rx' definiert werden. Um zu testen ob der PC mit dem micro:bit verbunden ist, wird mit Button B des mirco:bits eine Nachricht gesendet.

Zum Überprüfen der Verbindung wird PuTTY [31] verwendet. Das Open-Source Programm kann Verbindungen über Serielle Schnittstellen, Telnet uvm. herstellen.

Auf dem PC kann somit über PuTTY [31] mit der festgelegten Baudrate auf den, dem micro:bit zugewiesenen COM-Port, gehorcht werden. Nachrichten die der micro:bit mit der selben Baudrate an die USB-UART Verbindung absendet, erscheinen, sofern die Verbindung funktioniert, in dem Programm. Der COM-Port kann über den Geräte-Manager von Windows herausgefunden werden.

Code 4: Test der UART-Schnittstelle - micro:bit

```
1 from microbit import *
2
3 uart.init(115200)
4
5 while True:
6   if button_b.was_pressed():
7     uart.write("Hallo, ich bin der BBC micro:bit!")
```

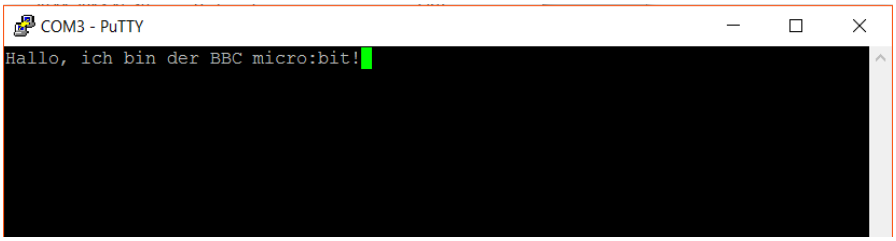


Abbildung 5.2: PuTTY [31] lauscht auf COM3 mit einer Baudrate von 115200

Da wir nun wissen, dass der micro:bit über den COM-Port mit dem Computer kommunizieren kann, verwenden wir dies in die andere Richtung. Hierfür wird von dem PC über C# eine Verbindung mit dem seriellen COM-Port geöffnet und eine Nachricht versendet. Dazu wird die Klasse SerialPort [22] benutzt.

Code 5: UART-Schnittstelle - PC

```
String microBitPort = "COM3"
SerialPort serial = new SerialPort(microBitPort);
serial.BaudRate = 115200;
serial.Open();

if(serial.IsOpen)
{
  serial.Write("Hallo, ich bin der PC")
}
```

Um diese mit dem micro:bit empfangen zu können muss dieser mittels UART mit der selben Baudrate auf Nachrichten warten. Diese Nachricht kann der Sender dann an den Empfänger weiterleiten.

Code 6: UART-Schnittstelle - micro:bit

```
import radio
from microbit import *

uart.init(115200)
radio.on()

while True:
  text = uart.readline()

  if text:
    display.scroll(text)

    radio.send(text)
```

5.1.3.2. Generierung und Übertragung der Befehlsbytes

Die Befehle, die der MotionStim8 versteht, müssen die vorgegebene 'Command structure' [20] einhalten.

SOF DL CID Data Checksum

SOF	Start of Frame	'255 255'
DL	Data Lenght in Bytes	
CID	Command ID	1-12
Data		
Checksum	Summe des kompletten Commands ohne SOF modulo 256	

Tabelle 1: Command Structure

Der Sender erhält vom PC einen String welcher die CID und die Data enthält. Diesen leitet der Sender an den Empfänger micro:bit weiter. Der Rest des Commands wird dort mittels eines Bytearrays zusammengesetzt. Dafür wird der erhaltene String mit dem Trim-Befehl in die einzelnen Werte aufgeteilt. Die DL ist dann die Länge der erhaltenen

Liste minus eins. Diese wird mit den Listeneinträgen in ein Bytearray geschrieben. Somit besteht das Bytearray bereits aus DL + CID + Data. Damit kann die Checksum errechnet werden, indem das Bytearray aufsummiert wird und das Ergebnis modulo 256 gerechnet wird. Zuletzt wird dem Bytearray vorne das SOF angehängt.

Code 7: Generierung der Command Structure - micro:bit

```
        #Generierung der Befehlsbytekette
        list = text.split()

        array = bytearray([0x00])
        lenght = len(list)
        array[0] = lenght-1

        for i in range(0,lenght):
          data = list.pop(0)
          array = array + bytearray([0x00])
          array[i+1] = int(data)

        s = 0
        for i in array:
          s += i
        checksum = s % 256

        array = array + bytearray([0x00])
        pos = len(array)-1
        array[pos] = checksum

        array = bytearray([0xFF,0xFF]) + array
```

5.1.3.3. MotionStim8-Anbindung

Die Anbindung an den MotioStim8 wird ebenfalls mit dem UART-Modul geregelt. Dafür muss UART mit einer Baudrate von 38400 und den belegten rx/tx Pins initialisiert werden. Wir verwenden als tx Pin8 und als rx Pin12.

```
uart.init(baudrate = 38400, tx=pin8, rx=pin12)
```

Bei Realisierung der Echtzeitfunktion trat ein Überlauf Problem auf. Dabei wurden die gesendeten Strings so schnell geschickt, dass die Queue des micro:bits voll war. Durch diesen Fehler sind, zum Einen einige Parameter verloren gegangen, zum Anderen erkannte der micro:bit den String nicht mehr als String (anscheinend ein Bug der Mu-Version).

Diese Beiden Fehler wurden durch ein erhöhen der Queue-Länge über

```
radio.config(queue = 10)
```

und dem einbauen eines Sleeps bei dem 'Sender' micro:bits.

5.2. Steuersoftware

5.2.1. Vorgehensweise

Es soll ein Programm zur Kopplung des Thalmic Myo-Armbands und dem Stimulator MotionStim8 entwickelt werden. Das Programm soll außerdem Gesten verwalten können und diesen sowohl Stimulationsparameter, als auch aufgezeichnete Myo-Daten zuweisen können.

Dafür wird in VisualStudio ein neues Windows-Forms-Projekt erstellt. Um einen modularen Aufbau zu gewährleisten werden nun die einzelnen Gesten-, Stimulations-, Myo-Controller, sowie der Mapper, als UserControls erstellt. Somit kann jeder Controller anhand eines eigenen Designers gestaltet werden. Als Erstes muss der Gesten-Controller seine Funktion und zu Grunde liegende XML-Datei erhalten. Diese Datei ersetzt für den Umfang dieser Arbeit eine Datenbank. Danach kann sich um die Umsetzung des Myo- und Stimulations-Controller, deren Funktionen zum Teil auf dem Gesten-Controller aufbauen, gekümmert werden. Der Mapper muss als letztes realisiert werden, da dieser Funktionen aller anderen Controller benötigt um die Kopplung des Myo-Armbands und dem MotionStim8 durchführen zu können.

Sind die Controller gestaltet und mit Funktion versehen, können sie auf dem Main-Window platziert werden. Controller-Übergreifende Funktionalitäten werden direkt über das Main-Window realisiert, dazu gehören bspw. das ausführen des Mappings oder des Echtzeit-Modus.

5.2.2. Graphical User Interface GUI

Neben der Darstellung der WindowsForm mitsamt der einzelnen Module, beherbergt die GUI die Buttons für Controllerübergreifende Funktionen, wie dem Mapping, der Echtzeitfunktion und dem Zuweisen von Daten/Parametern an eine Geste.
Diese übergreifenden Funktionen werden nach dem entsprechend gedrückten Button in den einzelnen Modulen aufgerufen. Für den Benutzer ersichtlich wird dies durch Benachrichtigungen per MessageBox, siehe Abb. 5.3. Über diese kann der User bspw. über fehlende Daten oder anderweitige Fehler informiert werden.

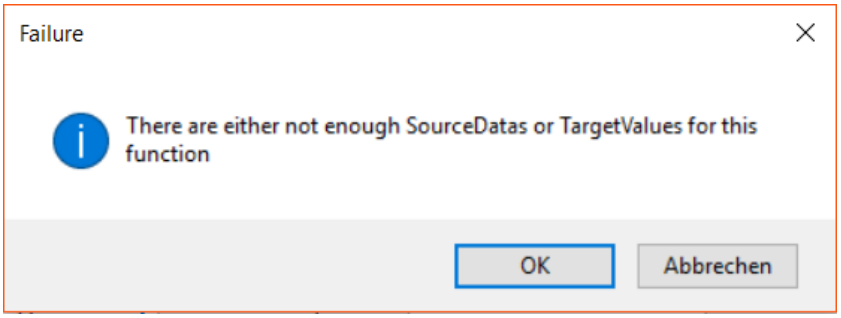


Abbildung 5.3: MessageBox mit Fehlermeldung

Über das Menü (MenuStrip) lässt sich der COM-Port des angeschlossenen BBC micro:bit Systems beliebig ändern. Der als default eingestellte COM ist der COM-3 Port.

5.2.2.1. Controllerübergreifende Funktionen

Mapping
Ruft die Funktion Mapping des Mappers auf, dafür werden die Source-Values (float[,]) und die Pulsweiten(float[,]) über den GestureController übergeben. Es wird die Mapp-Matrix zurückgegeben und zum Speichern an den GestureController übergeben.

Show Target Values
Übergibt dem TargetController eine Liste(List<String>) der Werte und die aktuell ausgewählte Position(integer), um die passenden Stimulationsparameter anzuzeigen. Dabei werden die Liste und die Position vom GestenManager abgefragt.

Record Source-Data
Startet und stoppt im SourceController die Aufnahme. Leitet Aufnahme(float[,]) an den GestureController zum Speichern weiter.

Loop
Startet die Echtzeitfunktion. Dafür werden Myo-Daten, Stimulationsparameter sowie die Mapping-Matrix, welche bereits vorher berechnet worden sein muss, benötigt.

In Intervallen von 50ms werden die Myo-Daten mit Hilfe der Mapping-Matrix in Pulsweiten umgewandelt. Überschreiten errechnete Werte die angegebene Schmerzgrenze eines Probanden so wird maximal diese als Pulsweite gesetzt.

D.h. die Amplitude und Frequenz einer Geste wird bereits zu Beginn der Funktion im Stimulator gesetzt. Während die Pulsweite in 50ms Intervallen angepasst wird.

Start Target
Diese Funktion ruft die StartTarget-Methode des Target Controllers auf. Dabei wird der Port des micro:bits Übergeben und die Stimulation mit den aktuellen Parametern des MotionStim8 gestartet. Werden Parameter verändert so werden diese Änderungen direkt an den Stimulator weitergegeben

5.2.3. GestureController

Der GestureController ist das wohl größte Modul, in ihm werden unter anderem alle Daten zusammengeführt und gespeichert, des Weiteren werden mit ihm alle Gesten verwaltet. Er bildet die Grundlage des Programms.
Zum Darstellen der einzelnen Gesten wird eine ListView verwendet. Diese bietet eine Übersicht über alle Gesten. Nach dem Auswählen einer bestimmten Geste werden mehr Informationen zu dieser angezeigt.
Die Informationen werden neben der ListView in TextBoxen angezeigt. Außerdem befindet sich dort ein Schieberegler, um die Position der Geste wählen zu können.

5.2.3.1. Speichern und Einlesen der Daten

Die Daten der Gesten mitsamt ihrer Myo-Daten und ihrer Stimulationsparameter werden in einer XML-Datei abgespeichert. Diese wird beim Starten des Programms eingelesen und beim Beenden aktualisiert. Dies wird in C# mittels eines DataSets ermöglicht. Die Methoden

```
dataset.ReadXML(String XMLDatei)
```

sowie

```
dataset.WriteXML(StringXMLDatei)
```

werden hierfür verwendet. Dieses DataSet kann aus mehreren Tabellen(DataTables) bestehen. In dieser Arbeit wird nur eine Tabelle, siehe Abb. 5.4, genutzt.

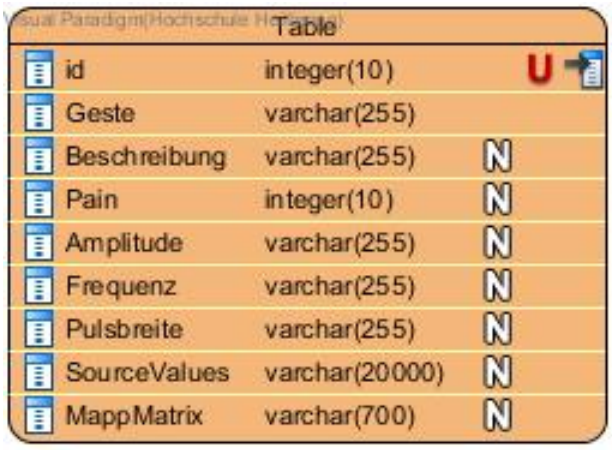


Abbildung 5.4: Datenbank-Aufbau

Speichern
Zum Speichern der Daten reicht es die Daten in die richtige Reihe der DataTable zu schreiben, dafür muss die id und die Spalte angegeben werden. Für die Geste mit der verwendeten ID wird der Wert der angegebenen Spalte dann gesetzt.

```
db.Rows[item.Index]["Geste"] = gestenName;
```

Wird hiernach die Methode WriteXML genutzt sind die Änderungen in der XML-Datei sichtbar.

Einlesen
Hierfür wird beim Start des Controllers die Methode ReadXML angewendet. Danach kann die ListView des Controllers mit den Gesten aus der DataTable gefüllt werden. Hierfür wird mit Schleifen die Tabelle durchlaufen und für jeden Eintrag in der Tabelle die ID mit dem Namen der Geste in die ListView geladen.

5.2.3.2. Anzeigen einer Geste
Wählt man in der ListView eine Geste aus, so werden der Name der Geste, die Beschreibung und ein Regler für fünf Positionen angezeigt. Erst hier wird die Beschreibung aus dem DataTable gelesen.

5.2.3.3. Hinzufügen, Löschen oder Ändern von Gesten
Für diese Funktionen gibt es im GestureController Buttons. Diese Buttons sind nicht immer aktiv. D.h. die Buttons Ändern und Löschen sind erst aktiv, wenn auch eine Geste ausgewählt ist.

Hinzufügen
Wird der Button 'Geste hinzufügen' gedrückt, so aktivieren sich die TextBoxen neben der ListView. Hier kann nun Name, Beschreibung und Schmerzgrenze der neuen Geste eingetragen werden. Wird dies mit dem Drücken des 'Speichern' Buttons bestätigt befindet sich die neue Geste im ListView. Außerdem erhält der Benutzer eine Information über eine MessageBox, dass die Geste erfolgreich gespeichert wurde.

Löschen
Wird der Button 'Geste löschen' gedrückt, so muss erst noch explizit bestätigt werden, dass die Geste gelöscht werden soll. Wird bestätigt, so wird die ausgewählte Geste mittels ihrer eindeutigen ID aus der DataTable gelöscht. Danach wird der ListView aktualisiert, damit die Geste in diesem nicht mehr fälschlicherweise angezeigt wird. Auch hier wird der Benutzer mittels MessageBox über den abgeschlossenen Vorgang informiert.

Ändern
Wird der Button 'Ändern' gewählt, aktivieren sich erneut die TextBoxen für den Namen, die Beschreibung und die Schmerzgrenze der Geste. Diese können dann beliebig geändert werden. Werden die Änderungen gespeichert, so wird der für die ID bereits vorhandene Eintrag in der DataTable überschrieben.

5.2.3.4. Speichern und Ausgeben der Stimulationsparameter

Speichern

Die Stimulationsparameter werden einzeln gespeichert, d.h. Amplitude, Frequenz und Pulsweite. Hierfür werden die Parameter(integer) in einen String umgewandelt und dann in die DataTable geschrieben. Diese Parameter sind Positionsabhängig und müssen auch so gespeichert werden. Außerdem muss der String auch erweiterbar sein, sowie veraltete Werte für Positionen durch neue ersetzen können.
Unterteilt wird dieser String wie folgt:

Schema:
"Position:Wert1.Wert2.Wert3.Wert4.Wert5.Wert6.Wert7.Wert8-Position: (...)"
Beispiel:
"3:17.15.12.11.13.23.42.22-2: (...)"

Ausgeben

Um die Werte ausgeben zu können muss der String wieder in seine Werte aufgespalten werden. Dazu wird die Split-Methode in Schleifen verwendet. Zuerst müssen die einzelnen Positionen getrennt werden, hier unterteilt durch '-'. Danach kann nach einer gewünschten Position gesucht werden. Ist diese vorhanden werden die Werte abgespalten, getrennt mittels ':'. Um die einzelnen Werte, welche noch mit '.' unterteilt sind, zu erhalten müssen diese nun nur noch auseinander genommen werden.

5.2.3.5. Speichern und Ausgeben der Myo-Daten

Ähnelt dem Prinzip der Stimulationsparameter. Hier wird ebenfalls ein String erzeugt, diesmal aus einem float[,]. Einzige Neuerung, die Werte der acht Sensoren werden noch zusätzlich durch ein '_' getrennt.

Schema:
"Position:Wert.Wert.Wert_Wert.Wert.Wert_ (... für acht Sensoren)-Position: (...)"

5.2.3.6. Speichern und Ausgeben der Mapping-Matrix

Ebenfalls nicht grundlegend anders als bei dem Vorangegangenen. Hier jedoch einfacher unterteilt. Die MappMatrix wird lediglich Zeilenweise, die Werte jeweils durch '.' unterteilt und die Zeilen durch '_', in einen String geschrieben.

Schema:
"Wert1.Wert2.(...).Wert8_(...)"

5.2.4. Stimulation Controller

Der Stimulation Controller ist das für den MotionStim8 verantwortliche Modul. In ihm werden die Parameter und auch die Anbindung des Stimulators gesteuert.

Da der Controller dafür viele Regler benötigt, wurde um ihn möglichst übersichtlich zu halten eine TabPage verwendet. Somit sind nur acht der insg. 24 Trackbars auf einen Blick zu sehen.
Des Weiteren wird eine Kopplungsfunktion für die einzelnen Parameter, Amplitude, Frequenz und Pulsweite, benötigt. Diese ist über eine CheckBox aktivierbar und somit lassen sich alle Regler für den aktuellen Parameter gleichzeitig steuern.

5.2.4.1. Parameter laden

Übergibt eine List<String> welche alle Stimulationsparameter beinhaltet, der Aufbau ist zu sehen in Abschnitt 5.2.3 auf Seite 30 Abschnitt 'Speichern und Ausgeben der Stimulationsparameter'. Aus dieser werden die Werte der aktuell ausgewählten Position ausgelesen.
Die jeweiligen Werte werden in die entsprechende TrackBox und TextBox eingetragen.

5.2.4.2. Start/Stop Target

Start

Erstellt und öffnet via SerialPort eine Verbindung zum micro:bit. Dafür wird der über die GUI erhaltene Port genutzt. Die Baudrate kann beliebig gesetzt werden, beachtet werden muss nur, dass die Gleiche bei micro:bit und PC verwendet werden muss.
Außerdem wird ein Timer mit einem Interval von 1000ms gestartet, um einen Überlauf zu vermeiden. Dieser Timer sendet alle Kommandos an den micro:bit, um alle Parameter(Amplitude, Frequenz und Pulsweite) zu übertragen. Hiernach wird der Timer beendet. Solange jedoch nicht die Methode StopTarget ausgeführt wurde werden jegliche Veränderungen der Parameter direkt an den micro:bit gesendet. Dazu wird eine Boolean Variable auf 'true' gesetzt.

Stop

Beendet den Timer und schließt den SerialPort. Außerdem setzt es die Boolean Variable wieder auf 'false'.

5.2.5. MYO Controller

Der MYO Controller ist der für das Thalmic Myo-Armband verantwortliche Modul. Es stellt die Myo-Daten in einem Diagramm(Chart) dar und besitzt ansonsten nur zwei Buttons zum Starten und Beenden des MYO-Armbands.

5.2.5.1. Start/Stop Source

Start

Startet das von Dr. Christoph Maier entwickelte Programm myo2ft.exe, siehe Kapitel 2.5.1 Seite 6. Um sich danach mit dem BufferClient von BufferBCI, siehe Kapitel 2.5.2 Seite 7, auf den Buffer-Server zu verbinden. Dieser kann über 'localhost' mit dem Port '1972' erreicht werden.
Außerdem wird der Header ausgelesen und zum Einen die Labels der einzelnen Kanäle in die Chart eingelesen. Zum Anderen wird der Header-Start-Wert gespeichert. Dies ist notwendig, da das Myo-Armband die Werte fortlaufend zählt, und somit nicht bei 0 anfängt wenn man es startet.
Zuletzt wird ein Timer mit einem Intervall von 50ms gestartet welcher die Chart auf die neuen Werte updated. Dazu wird der Header erneut ausgelesen und die neuen Werte, die mittels des Header-Start-Werts errechnet werden können, erhalten werden. Für jeden Kanal wird dann ein Mittelwert über den Betrag der Werte ausgerechnet. Die für jeden Kanal ermittelten Werte können dann in die Chart eingetragen werden, nachdem die alten entfernt wurden.

Stop

Stoppt den Timer und beendet die Verbindung zum Myo-Armband.

5.2.5.2. Start/Stop/Cancel Record

Diese Funktionen setzen das Starten des Myo-Armbands vorraus. Falls dieses nicht gestartet wurde, erscheint eine passende Meldung für den Benutzer.

Start

Startet die Record-Funktion, indem ein Boolean auf 'true' gesetzt wird. Dieses Boolean führt dazu, dass der aktuell laufende Timer die gemittelten Werte jedes Kanals in ein Array speichert.

Stop

Setzt das Boolean auf 'false' und stoppt somit die Aufnahme. Gibt außerdem die aufgenommenen Daten als Array zurück.

Cancel

Beendet ebenfalls die Aufnahme verwirft jedoch das Array.

5.2.5.3. Record abspielen/Abspielen beenden

Abspielen

Leert die aktuelle Chart und startet einen Timer. Dieser Timer geht in einem 50ms Intervall die als Array übergebene Aufnahme durch. Dabei werden immer die acht Kanal-Werte gesetzt und die vorherigen gelöscht. Ist die Aufnahme abgearbeitet wiederholt der Timer diese bis das Abspielen beendet wird.

Abspielen beenden

Beendet den Timer und damit auch das Abspielen der Aufnahme.

5.2.6. Mapper

Der Mapper bietet zwei grundlegende Funktionen. Zum Einen kann er eine 8x8 Mapping-Matrix aus den Myo-Daten aller Positionen einer Geste und den dazu gehörenden Pulsweiten erstellen. Zum Anderen enthält er eine Loop-Funktion, welche aus den aktuellen Myo-Daten anhand der Mapping-Matrix die passenden Pulsweiten errechnet.

5.2.6.1. Mapping

Die Mapping-Funktion benötigt eine Nx8 Matrix der Myo-Daten für alle Positionen. Hierbei können für jede Position n-Daten vorliegen, diese besitzen jeweils die acht Sensoren des Myo-Armbands. N entspricht allen vorliegenden Datensätzen für alle Positionen.

$$
MyoMatrix = \begin{array}{c} \\ Position1 \\ \vdots \\ Position1 \\ Position2 \\ \vdots \\ \vdots \\ Position5 \\ \vdots \\ Position5 \end{array}
\begin{array}{c}
\begin{array}{cccc} Sensor1 & Sensor2 & \dots & Sensor8 \end{array} \\
\begin{pmatrix}
a_{1.1} & a_{2.1} & \dots & a_{8.1} \\
\vdots & \vdots & \dots & \vdots \\
a_{1.n1} & a_{2.n1} & \dots & a_{8.n1} \\
b_{1.1} & b_{2.1} & \dots & b_{8.1} \\
\vdots & \vdots & \dots & \vdots \\
\vdots & \vdots & \dots & \vdots \\
e_{1.1} & e_{2.1} & \dots & e_{8.1} \\
\vdots & \vdots & \dots & \vdots \\
e_{1.n5} & e_{2.n5} & \dots & e_{8.n5}
\end{pmatrix}
\end{array}
\tag{1}
$$

Außerdem benötigt das Mapping für jede Position die passenden Pulsweiten. Diese werden passend zu den für die jeweilige Position n-mal untereinander in eine Matrix geschrieben. Es entsteht ebenfalls eine Nx8 Matrix

$$
StimMatrix = \begin{array}{c} \\ Position1 \\ \vdots \\ Position1 \\ Position2 \\ \vdots \\ \vdots \\ Position5 \\ \vdots \\ Position5 \end{array}
\begin{array}{c}
\begin{array}{cccc} Kanal1 & Kanal2 & \dots & Kanal8 \end{array} \\
\begin{pmatrix}
plw_{1.1} & plw_{1.2} & \dots & plw_{1.8} \\
\vdots & \vdots & \dots & \vdots \\
plw_{1.1} & plw_{1.2} & \dots & plw_{1.8} \\
plw_{2.1} & plw_{2.2} & \dots & plw_{2.8} \\
\vdots & \vdots & \dots & \vdots \\
\vdots & \vdots & \dots & \vdots \\
plw_{5.1} & plw_{5.2} & \dots & plw_{5.8} \\
\vdots & \vdots & \dots & \vdots \\
plw_{5.1} & plw_{5.2} & \dots & plw_{5.8}
\end{pmatrix}
\end{array}
\tag{2}
$$

Um die Mapping-Matrix zu erhalten wird die Pseudoinverse der MyoMatrix mit der StimMatrix multipliziert.

```
Matrix<float> mapMatrix = myoMatrix.PseudoInverse()*stimMatrix;
```

Somit entsteht eine 8x8 Matrix, welche für jeden Kanal eine Art Umrechnungsvektor beinhaltet. Diese Matrix wird für den Loop, also die Umrechnung von Myo- auf Stimulations-Werte (Pulsweiten) benötigt.

$$MapMatrix = \begin{array}{c} \begin{array}{cccc} Kanal1 & Kanal2 & \dots & Kanal8 \end{array} \\ \begin{pmatrix} a_1 & a_2 & \dots & a_8 \\ b_1 & b_2 & \dots & b_8 \\ \vdots & \vdots & \dots & \vdots \\ h_1 & h_2 & \dots & h_8 \end{pmatrix} \end{array} \tag{3}$$

5.2.6.2. Loop

Der Loop ist das Kernstück der Echtzeitfunktion. In ihm werden die gemessenen Myo-Werte in Pulsweiten umgerechnet und können direkt an den Stimulator weiter gesendet werden.
Um die Pulsweiten errechnen zu können benötigt der Loop zum einen die 8x8 MapMatrix und zum Anderen die 1x8 MyoDaten.

$$MyoDaten = \begin{array}{l} Sensor1 \\ Sensor2 \\ \vdots \\ Sensor8 \end{array} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_8 \end{pmatrix} \tag{4}$$

Wie bereits in Kapitel 5.2.5 auf Seite 31 erwähnt, messen wir diese im 50ms Intervall und berechnen für jeden Sensor den Mittelwert für die in der Zeit ermittelten Werte. Die Pulsweiten lassen sich dann durch Multiplikation der MapMatrix mit den MyoDaten errechnen.

```
Vector<float> plw = myoDaten * mapMatrix;
```

Somit erhält man einen Vektor mit den Pulsweiten für jeden Kanal.

$$Pulsweiten = \begin{matrix} Kanal1 \\ Kanal2 \\ \vdots \\ Kanal8 \end{matrix} \begin{pmatrix} plw_1 \\ plw_2 \\ \vdots \\ plw_8 \end{pmatrix} \tag{5}$$

6. Ergebnis

In diesem Kapitel wird auf die entwickelte Software und ihre Funktionalität eingegangen. Das hierfür entwickelte Konzept(Kapitel 4 Seite 17), die erhobenen Anforderungen(Kapitel 3 Seite 8) und die Implementierung(Kapitel 5 Seite 21) sind dafür essentiell und als Teil des Ergebnisses anzusehen.

6.1. Steuersoftware

Die in dieser Arbeit entwickelte Software beinhaltet sowohl die in der Anforderungsanalyse definierten Funktionalitäten und erfüllt ebenfalls die gestellten Nicht-Funktionalen Anforderungen.

6.1.1. Graphical User Interface (GUI)

Die Nicht-Funktionalen Anforderungen können hauptsächlich subjektiv bewertet werden. Die benutzerfreundliche und übersichtliche GUI wurde dabei, nach eigenem ermessen, bestmöglich umgesetzt, siehe Abb. 6.1.

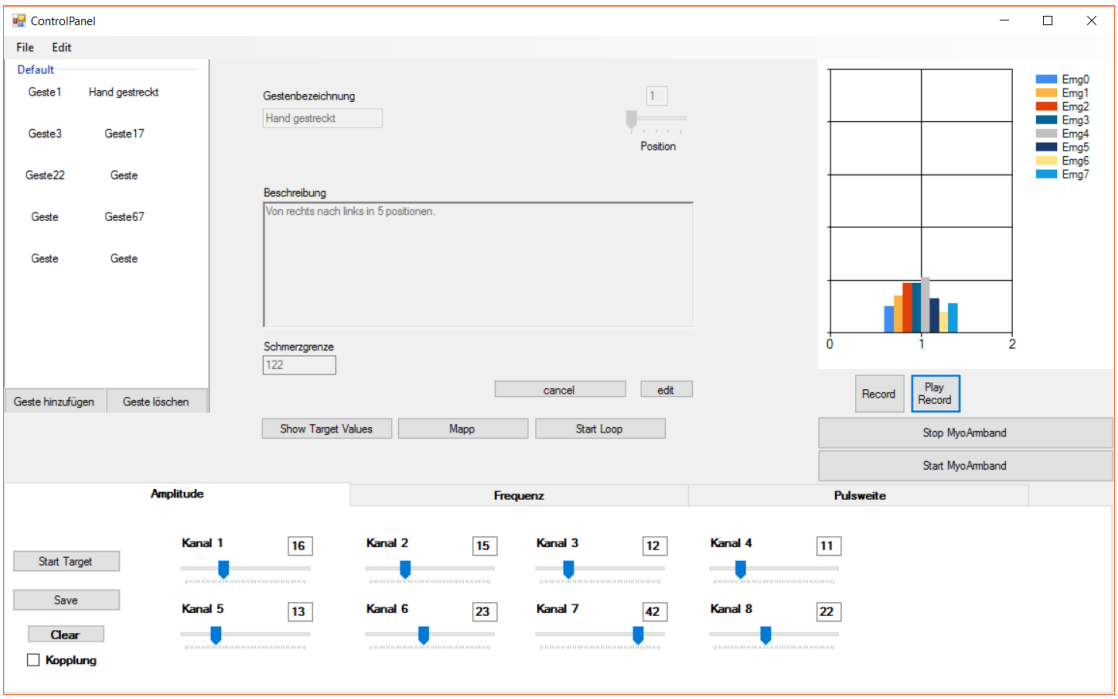


Abbildung 6.1: Screenshot der Anwendung

6.1.2. Klassenarchitektur

Auch die modulare Umsetzung wurde angestrebt und mittels Interfaces realisiert. D.h. es existieren übergeordnete Interfaces Target und Source von denen in dem aktuellen Fall die Controller des MotionStim8 und des Myo-Armbands erben. Somit können andere Targets oder Sources eingesetzt werden, indem diese das jeweilige Interface implementieren und somit die für übergeordnete Funktionen essentiellen Methoden beinhalten müssen. Die Komponenten können somit komplett getauscht werden indem ein neuer Controller eingefügt wird, siehe Abb. 6.2.

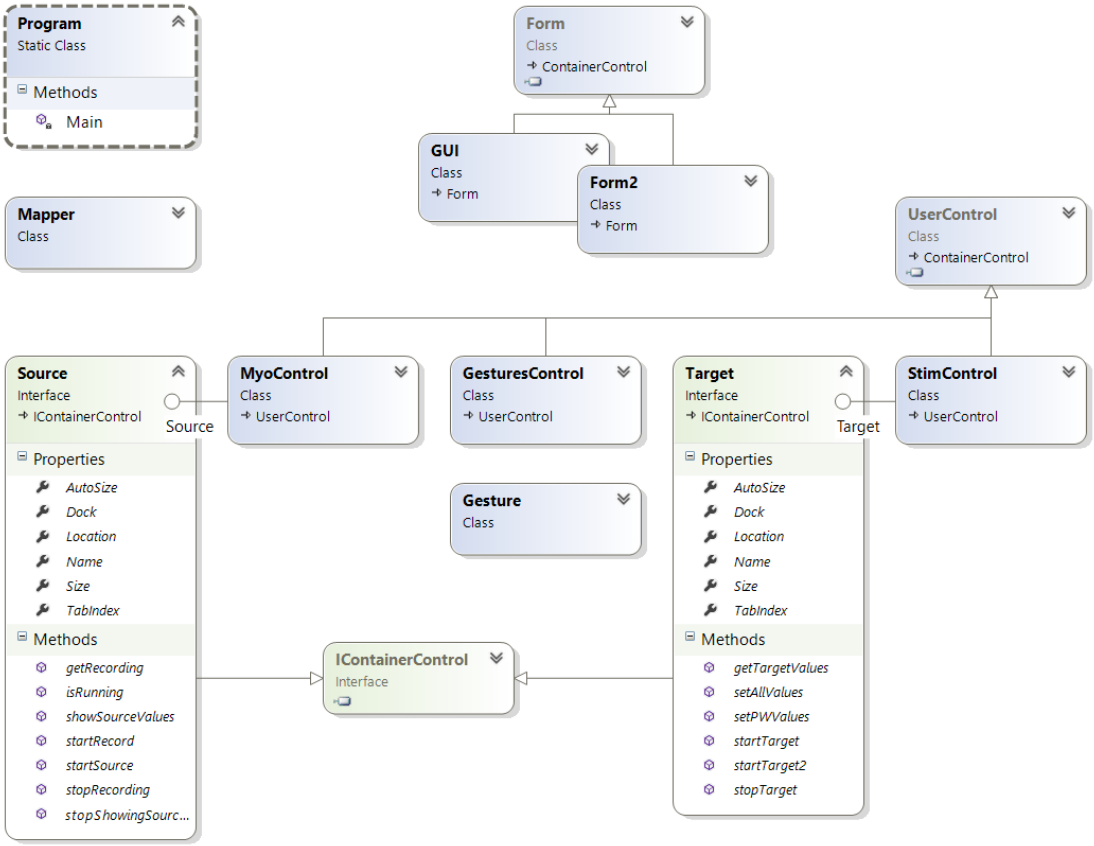


Abbildung 6.2: Klassendiagramm der Anwendung

6.1.3. Funktionalität

6.1.3.1. Verwalten von Gesten

Es können Gesten hinzugefügt, gelöscht und geändert werden. Jeder Vorgang wird dabei bei Bestätigung eine Nachricht per MessageBox ausgeben, um den Benutzer über eine Erfolgreiche Ausführung zu informieren.

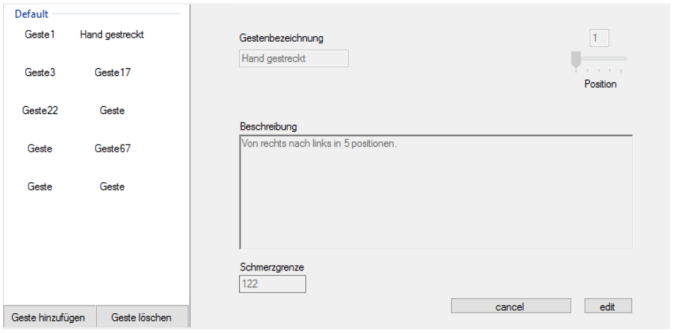


Abbildung 6.3: Oberfläche des GestureControllers

6.1.3.2. Anzeigen und Umgang mit MYO-Daten

Myo-Rohdaten werden von dem Programm empfangen und der Mittelwert über 50ms für den Benutzer ersichtlich in einer Chart angezeigt, siehe Abb. 6.4. Auch können Myo-Daten über einen bestimmten Zeitraum aufgenommen und einer Gestenposition zugewiesen werden.

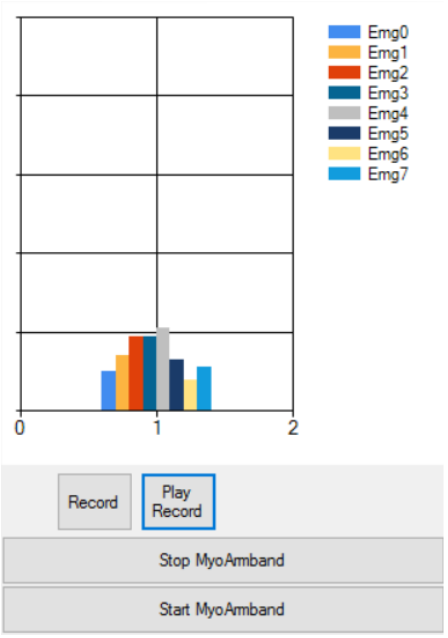


Abbildung 6.4: Oberfläche des Myo Controllers

6.1.3.3. Anzeigen und Umgang mit Stimulationsparametern

Stimulationsparameter können in ihrem jeweiligen Rahmen verändert werden. Außerdem gibt es eine Kopplungsfunktion mit welcher alle Parameter einer Art gleichzeitig verändert werden können. Des Weiteren können Stimulationsparameter wie Myo-Daten bestimmten Gestenpositionen zugewiesen und wieder abgefragt werden.

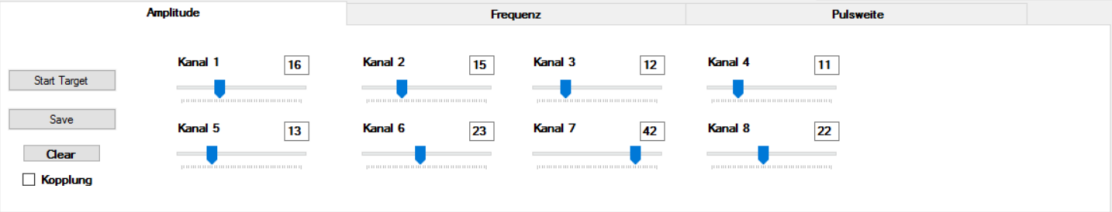


Abbildung 6.5: Oberfläche des Stimulation Controllers

6.1.3.4. Drahtlose Stimulation

Zum Zuweisen der Stimulationsparameter kann die drahtlose Stimulation aktiviert werden. Hierbei werden die aktuellen Stimulationsparameter an den MotionStim8 übertragen. Die Parameter des MotionStim8 werden bei jeglicher Veränderung der Parameter im StimController aktualisiert.

6.1.3.5. Kopplung von MotionStim8 und Thalmic MYO-Armband

Die Kopplung der beiden Geräte findet durch das Mapping statt. Hierbei wird für eine ausgewählte Geste die Mapping Matrix errechnet und gestenabhängig abgespeichert. Falls diese Funktion mit der ausgewählten Geste aufgrund von fehlenden Daten nicht möglich ist, wird der Benutzer darüber informiert.

6.1.3.6. Echtzeitfunktion

Für die Echtzeitfunktion müssen MotionStim8 und Myo-Armband angeschlossen, sowie das Mapping für die Geste durchgeführt worden sein. Ist dies nicht der Fall wird der Benutzer darüber informiert, dass die Funktion nicht ausgeführt werden kann. Die Funktion ermittelt für die, alle 50ms errechneten, Mittelwerte des Myo-Armbands die Pulsweiten für die Stimulation. Dies ist durch die berechnete Mapping-Matrix der Geste möglich. Die Pulsweiten werden dann drahtlos an den Stimulator gesendet. Einfache Gesten können somit in Quasi-Echtzeit, d.h. mit geringer Verzögerung, übertragen werden.

6.2. Fazit

Die in der Anforderungsanalyse definierten Anforderungen an die Anwendung, vor Allem die ausformulierten UseCases, wurden vollständig umgesetzt. Alle UseCases können mit der Steuersoftware ausgeführt werden.
Die anfänglich gesetzten Haupt-Ziele, wie eine Steuersoftware zur Kopplung von MotionStim8 und Myo-Armband, sowie die drahtlos Anbindung des Stimulators, konnten erfüllt werden. Die dafür benötigten Teil-Ziele, Verwaltung von Gesten, sowie Myo- und

Stimulations-Daten, wurden ebenfalls bewältigt.
Wie bereits erwähnt kann die Benutzerfreundlichkeit der Software nur subjektiv bewertet und dieser Punkt somit aus meiner persönlichen Sicht als erfüllt betrachtet werden.

Die Steuersoftware ersetzt die in der vorherigen Arbeit [15] verwendeten MATLAB Scripts (siehe Sequenzdiagramme Abschnitt A.1.2.2 Seite 50) und vereint die Funktionalitäten in einer einzigen Anwendung. Um die Funktionalität zu überprüfen wird zum Ende dieser Arbeit ein Testdurchlauf(siehe Anhang Abschnitt A.2.3 Seite 57) durchgeführt. Dafür wird unter Anderem die drahtlose Stimulation(siehe Anhang Abschnitt A.2.3.1 Seite 57) und die Echtzeitfunktion (siehe Anhang Abschnitt A.2.3.2 Seite 61) durchgeführt.

7. Diskussion und Ausblick

In diesem Kapitel sollen die Grenzen der in dieser Arbeit entwickelten Software aufgezeigt, sowie die Entscheidungen im Entwicklungsprozess diskutiert werden. Ebenfalls soll ein Ausblick darauf gegeben werden, inwieweit diese Anwendung in Zukunft erweitert werden könnte.

7.1. Diskussion

Bisher gab es nur eine Sammlung an MATLAB scripts, um sowohl die Kopplung von MotionStim8 und Myo-Armband, als auch die Echtzeitfunktion zu realisieren. Vollwertige Programme für diese Funktionen sind nicht aufzufinden. Die entwickelte Software kann diese und weitere Funktionen bieten. Insgesamt betrachtet lässt sich somit sagen, dass das Ergebnis dieser Arbeit als Erfolg verbucht werden kann.

Schwierigkeiten im Entwicklungsprozess

Während der Anbindung des Myo-Armbands wurde nach einer Software-Bibliothek gesucht, um diese zu realisieren ohne sie selbst zu implementieren. Dabei fand ich unter anderem die Bibliothek MyoSharp [24]. Diese bietet umfangreiche Funktionen für das Myo-Armband. Jedoch erzeugte das einbinden dieser Bibliothek in meine Arbeit mehrere Fehler aufgrund von verschiedenen Versionen, welche sich zwar teilweise beheben ließen, jedoch nicht vollständig. Deswegen nutze ich in der Arbeit den Buffer BCI [3], welcher zwar weniger Funktionen liefert, dafür jedoch gut eingebunden werden konnte.

Die drahtlose Anbindung des MotionStim8 verlief auch nicht ohne Schwierigkeiten. Obwohl die 'Command Structure' eingehalten wurde, reagierte dieser zuerst nicht. Die gesendeten Daten des micro:bits wurden invertiert. Dies zu beheben ging nicht im Code, um das Problem zu lösen wurde ein Hardware-Inverter zwischen geschaltet. Dank Dr. Christoph Maier wurde dieses Problem identifiziert und die Lösung gefunden.

Bei der restlichen Implementierung der Software bestand das größte Problem am Designer in VisualStudio. Dieser verursachte ein paar mal einen Absturz des Programms. Dieses Problem ist bekannt und kann nur durch entfernen des Designers gelöst werden. Da dieser jedoch das Erstellen der Benutzeroberfläche enorm vereinfacht und das Problem sehr selten auftrat, wurde er behalten.

Stärken und Schwächen der Software

Eine Schwäche des Programms ist allerdings die Aufnahme der Myo-Daten. Hier kann kein beliebiger Zeitraum aufgenommen werden. Der reservierte Speicherplatz reicht für ein Recording von maximal acht Minuten. Für die angestrebte Verwendung des Programmes für Handgesten sollte dies jedoch kein Problem darstellen.

Eine Weitere Schwäche zeigt sich im Echtzeitmodus. Dieser ist zeitverzögert, da unter anderem die Myo-Daten nicht in Echtzeit, sondern in einem 50ms Intervall ausgelesen werden. Außerdem verzögert die Übertragung der Parameter an den MotionStim8 zusätzlich nochmals um einige Millisekunden.

Die Stärken des Programmes äußern sich vor Allem in der Bedienbarkeit im Vergleich zum vorherigen. Regler und andere Anzeigen bieten Übersichtlichkeit. Die Möglichkeit des Speicherns von Gesten bzw. Parametern bietet eine bessere Verwendbarkeit, auch über das Beenden der Anwendung hinaus. Beim erneuten Starten des Programms sind alle gespeicherten Daten wieder vorhanden. Eine neue Kalibrierung, d.h. das Erstellen einer Mapping Matrix, ist somit nicht mehr vor jeder Anwendung notwendig.

Die drahtlose Anbindung des MotionStim8 ermöglicht es, bei der Anwendung des Programmes mehr Abstand zwischen den zwei möglichen Probanden zu haben. Der MotionStim8 ist jetzt portabel und handlicher. Er benötigt nur noch eine Kabelverbindung zu dem sehr kompakten BBC micro:bit.
Diese Anbindung kann durch die micro:bit Systeme kostengünstig realisiert werden.

Der modulare Aufbau des Programms bietet Flexibilität. Das Programm kann dadurch einfach mit neuen Targets oder Sources erweitert werden. Somit können bspw. Stimulatoren mit mehr oder weniger Kanälen wie der MotionStim8 ohne großen Aufwand eingebunden werden.

7.2. Ausblick

Wie bereits erwähnt, bietet die Software aufgrund des modularen Aufbaus eine gute Grundlage für Erweiterungen.

Denkbar wären andere Stimulatoren oder auch EMG Messgeräte. Aber auch das Anbinden eines Roboterarms als Target. Hierfür kann bspw. die mittels Myo-Armband aufgezeichnete Geste an den Roboterarm übertragen werden. Sobald dabei eine bestimmte Geste erkannt wird könnte der Roboterarm entweder vorgegebene Bewegungen abspielen, oder auch der weiteren Armbewegung des Probanden folgen.

Bei der Erweiterung des GestureControllers bieten sich auch weitere Möglichkeiten.

Beispielsweise durch einen weiteren Faktor, dem spezifischen User. Dabei kann für einzelne Benutzer eine Art Profil erstellt und spezifische Daten für die Stimulation hinterlegt werden. Dies wäre bspw. die Schmerzgrenze. Dieser Benutzer wird dann zusätzlich zur Geste ausgewählt. Die Geste ist ohne die Schmerzgrenze nicht mehr benutzerabhängig, sondern mit allen Benutzern kombinierbar.

Auch die Erweiterung der Beschreibung einer Geste wäre möglich. Diese erfolgt aktuell schriftlich. Denkbar wäre hier Bilder oder sogar Video-Sequenzen hochladen und anzeigen zu lassen. Dabei können komplexere Gesten besser dargestellt werden, als diese nur schriftlich zu beschreiben.

Des Weiteren kann die aktuell genutzte XML-Datei durch ein vollwertiges Datenbanksystem ersetzt werden. Dadurch wäre die Anwendung nicht mehr computerspezifisch und die Daten von mehreren PCs aus zu erreichen.

8. Eidesstattliche Erklärung

Ich erkläre an Eides Statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe und dass ich alle Stellen, die ich wörtlich oder sinngemäß aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe. Die Arbeit hat bisher in gleicher oder ähnlicher Form oder auszugsweise noch keiner Prüfungsbehörde vorgelegen.

Heilbronn, den 10. März 2017

Corinna Simon

Literaturverzeichnis

[1] CSharp-Station. C-Sharp. http://csharp-station.com/. Zuletzt Eingesehen am 22.02.2017.

[2] Evolus. Pencil, Programm zum Erstellen von MockUps. http://pencil.evolus.vn/. Zuletzt Eingesehen am 06.03.2017.

[3] J. Farquhar. Buffer BCI. https://github.com/jadref/buffer_bci. Zuletzt Eingesehen am 12.02.2017.

[4] Micro:bit Educational Foundation. BBC micro:bit. http://microbit.org/hardware/. Zuletzt Eingesehen am 12.02.2017.

[5] Micro:bit Educational Foundation. BBC micro:bit. https://www.microbit.co.uk/device/pins. Zuletzt Eingesehen am 24.02.2017.

[6] Micro:bit Educational Foundation. BBC micro:bit. http://microbit.org/code/. Zuletzt Eingesehen am 12.02.2017.

[7] R. Frey. Model-View-Controller Modell. https://commons.wikimedia.org/wiki/File:MVC-Process.svg. Zuletzt Eingesehen am 24.02.2017.

[8] D. George. MicroPython. http://micropython.org/. Zuletzt Eingesehen am 22.02.2017.

[9] DocCheck Medical Services GmbH. Elektromyographie. http://flexikon.doccheck.com/de/Elektromyographie. Zuletzt Eingesehen am 22.02.2017.

[10] DocCheck Medical Services GmbH. Êlektrostimulation. http://flexikon.doccheck.com/de/Elektrostimulation. Zuletzt Eingesehen am 22.02.2017.

[11] DocCheck Medical Services GmbH. Muskel. http://flexikon.doccheck.com/de/Muskel. Zuletzt Eingesehen am 06.03.2017.

[12] DocCheck Medical Services GmbH. Muskelstimulation bei Querschnittsgelähmten durch Implantat. http://news.doccheck.com/de/130495/der-laehmung-beine-machen/. Zuletzt Eingesehen am 06.03.2017.

[13] EXP GmbH. Deutsche Website welche den BBC micro:bit vertreibt. http://www.exp-tech.de/bbc-micro-bit. Zuletzt Eingesehen am 12.02.2017.

[14] Krauth + Timmermann GmbH. MotionStim8. http://www.krauthtimmermann.de/169-0-Motionstim-8.html. Zuletzt Eingesehen am 12.02.2017.

[15] S. Hannß. Untersuchungen zur Kopplung des Myo-Armbands mit dem Elektrostimulator MotionStim8. Bachelorarbeit, Hochschule Heilbronn, 2016.

[16] N. H.Tollervey. MicroPython Code Editor Mu. https://codewith.mu/. Zuletzt Eingesehen am 12.02.2017.

[17] Thalmic Labs Inc. Myo-Armband. `https://www.myo.com/techspecs`. Zuletzt Eingesehen am 12.02.2017.

[18] Thalmic Labs Inc. Myo-Connect Software. `https://www.myo.com/start`. Zuletzt Eingesehen am 12.02.2017.

[19] Thalmic Labs Inc. Myo SDK für Entwickler. `https://developer.thalmic.com/docs/api_reference/platform/getting-started.html`. Zuletzt Eingesehen am 12.02.2017.

[20] Martin Rohm. Testbox RS232.

[21] Microsoft. C-Sharp im Detail. `https://msdn.microsoft.com/de-de/library/bb979023.aspx`. Zuletzt Eingesehen am 22.02.2017.

[22] Microsoft. SerialPort-Klasse. `https://msdn.microsoft.com/de-de/library/system.io.ports.serialport`. Zuletzt Eingesehen am 12.02.2017.

[23] Microsoft. VisualStudio 2015. `https://msdn.microsoft.com/de-de/library/dd831853.aspx`. Zuletzt Eingesehen am 22.02.2017.

[24] T. Uzun N. Cosentino. MyoSharp. `https://github.com/tayfuzun/MyoSharp`. Zuletzt Eingesehen am 24.02.2017.

[25] J. Gandar und Weitere N. Wenger, E. M. Moraud. Muskelstimulation bei Querschnittsgelähmten durch Implantat Artikel in Nature Medicine. `http://www.nature.com/nm/journal/v22/n2/full/nm.4025.html`. Zuletzt Eingesehen am 06.03.2017.

[26] 3Sat online. Interview mit Dr. Jochen Quintern von der Neurologischen Klinik Bad Aibling. `http://blog.stand-up-initiative.de/wp-content/uploads/pdf/medizinisch/msfg.pdf`. Zuletzt Eingesehen am 06.03.2017.

[27] Oostenveld, R., Fries, P., Maris, E., Schoffelen, JM. FieldTrip: Open Source Software for Advanced Analysis of MEG, EEG, and Invasive Electrophysiological Data. *Computational Intelligence and Neuroscience*, 2011.

[28] Visual Paradigm. Visual Paradigm, Programm zum Erstellen von Diagrammen. `https://www.visual-paradigm.com/`. Zuletzt Eingesehen am 06.03.2017.

[29] C. Rüegg. Math.NET Initiative. `https://www.mathdotnet.com/`. Zuletzt Eingesehen am 22.02.2017.

[30] C. Rüegg. Math.NET Numerics. `https://numerics.mathdotnet.com/`. Zuletzt Eingesehen am 22.02.2017.

[31] B. Harris J. Nevins S. Tatham, O. Dunn. PuTTy. `http://www.putty.org/`. Zuletzt Eingesehen am 24.02.2017.

[32] Mehrere unbekannte Authoren. BBC micro:bit MicroPython Dokumentation. https://microbit-micropython.readthedocs.io/en/latest/tutorials/radio.html. Zuletzt Eingesehen am 12.02.2017.

[33] Mehrere unbekannte Authoren. BBC micro:bit MicroPython Dokumentation. http://microbit-micropython.readthedocs.io/en/latest/uart.html. Zuletzt Eingesehen am 12.02.2017.

A. Appendix

A.1. Steuersoftware

A.1.1. MockUps

Alle folgenden MockUps wurden mit Pencil [2] erstellt.

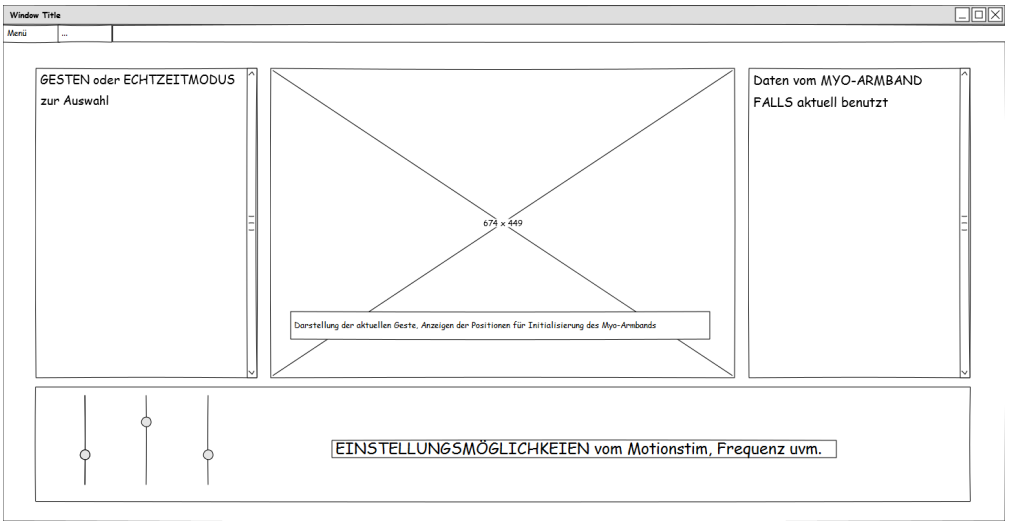


Abbildung A.1: Erstes MockUp der Steuersoftware

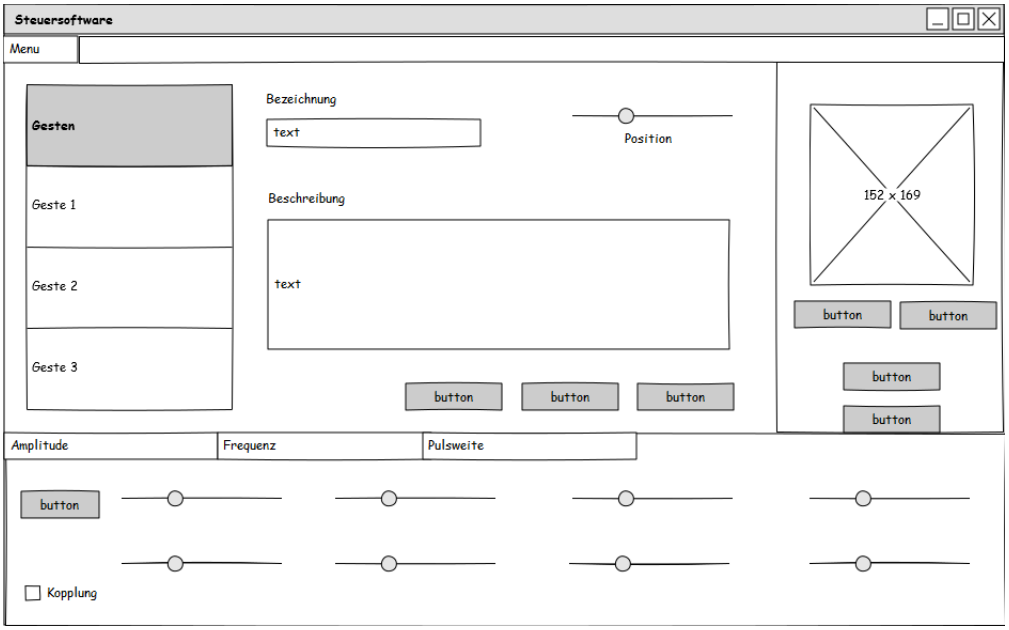


Abbildung A.2: Fortgeschrittenes MockUp der Steuersoftware

A.1.2. Diagramme

Die folgenden Diagramme wurden mit Visual Paradigm [28] erstellt.

A.1.2.1. Komponentendiagramm

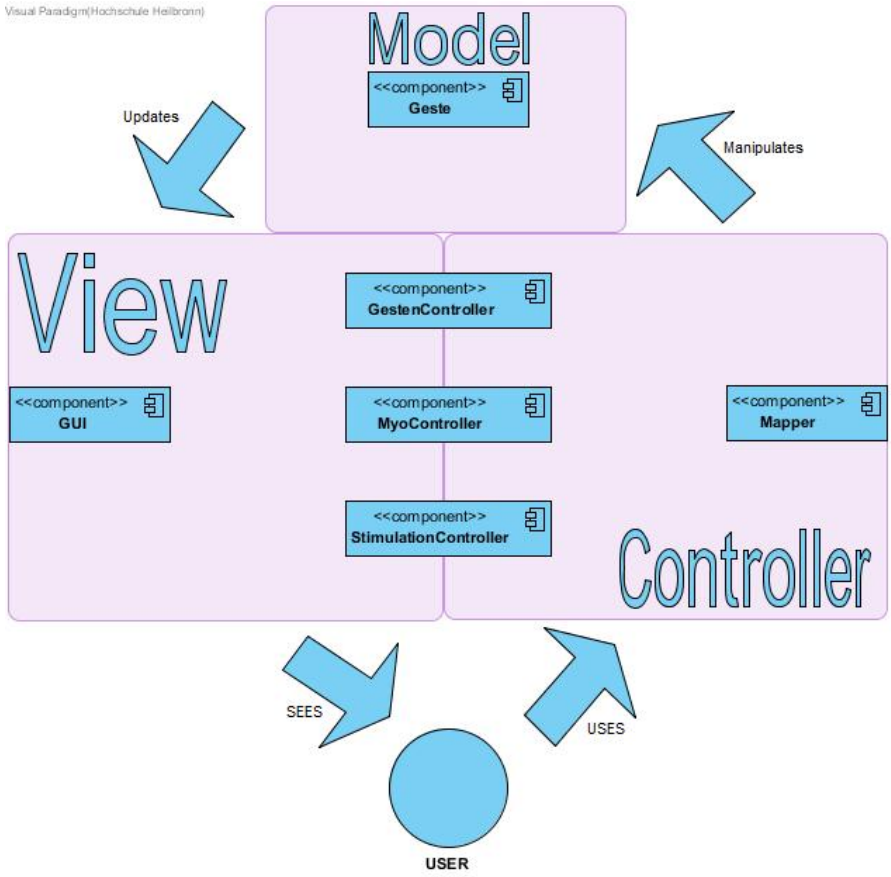


Abbildung A.3: Komponentendiagramm der Steuersoftware

A.1.2.2. Sequenzdiagramme

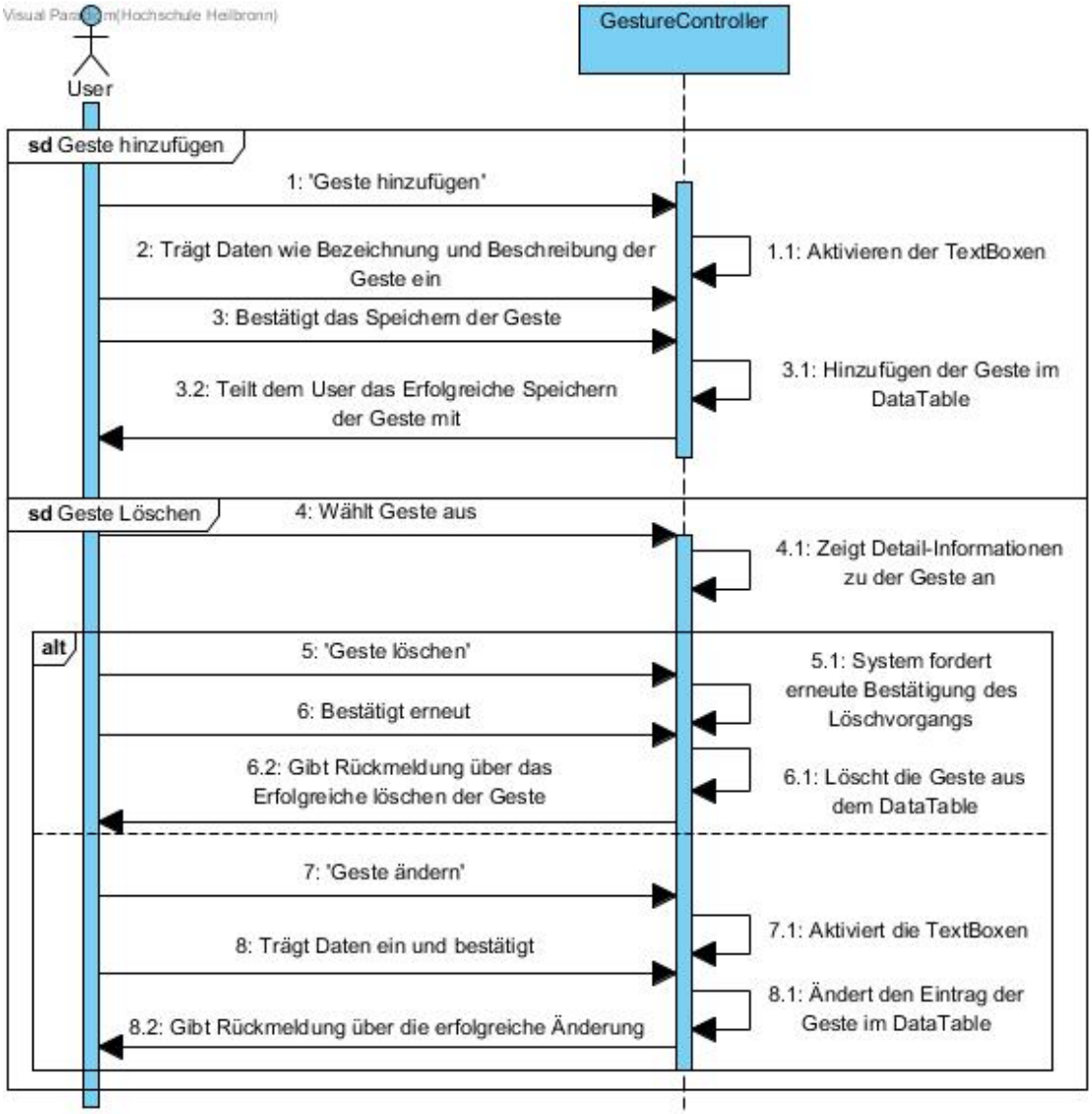


Abbildung A.4: Sequenzdiagramm der gestenverwaltenden UseCases

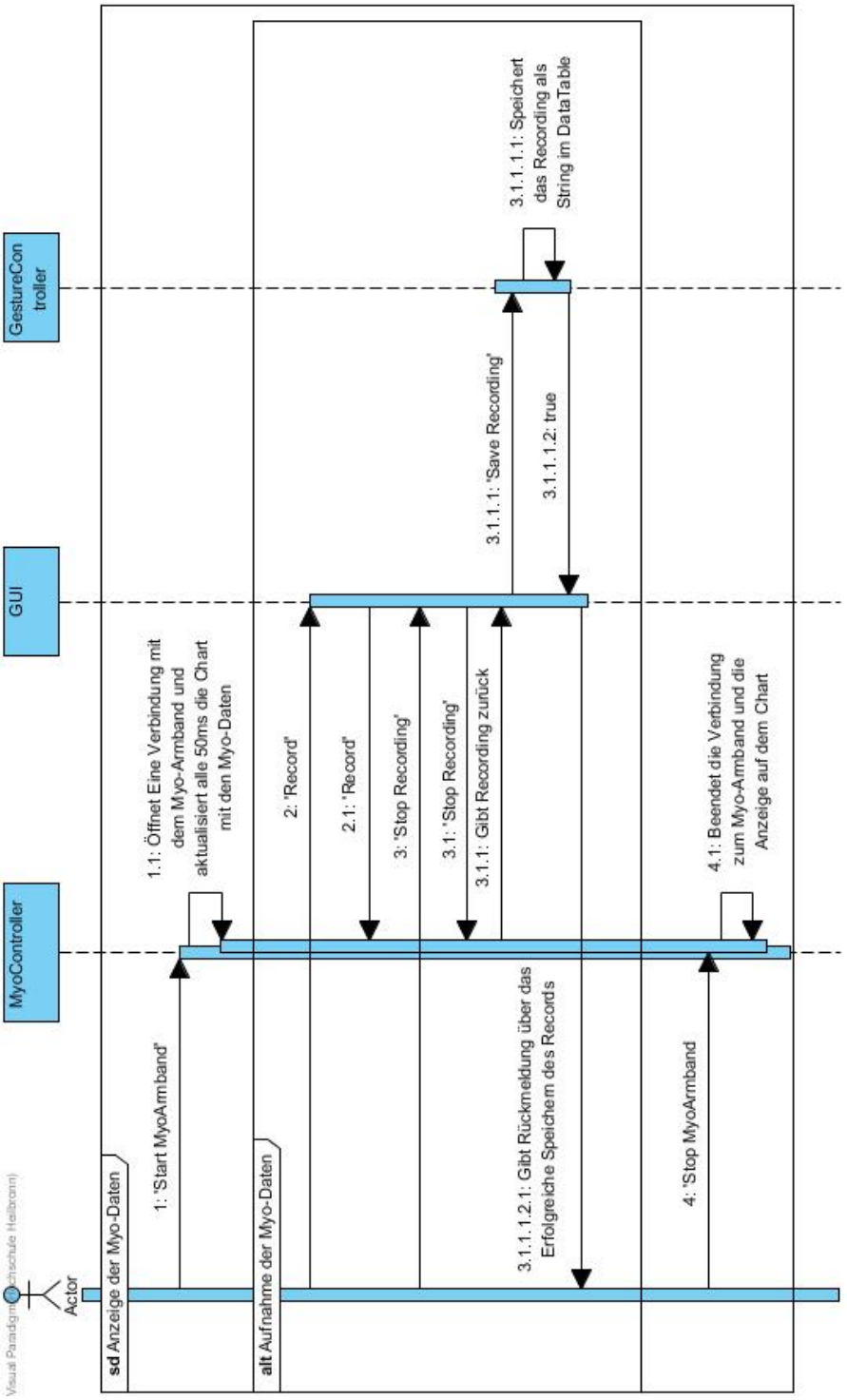


Abbildung A.5: Sequenzdiagramm der Abläufe der Myo-Armband UseCases

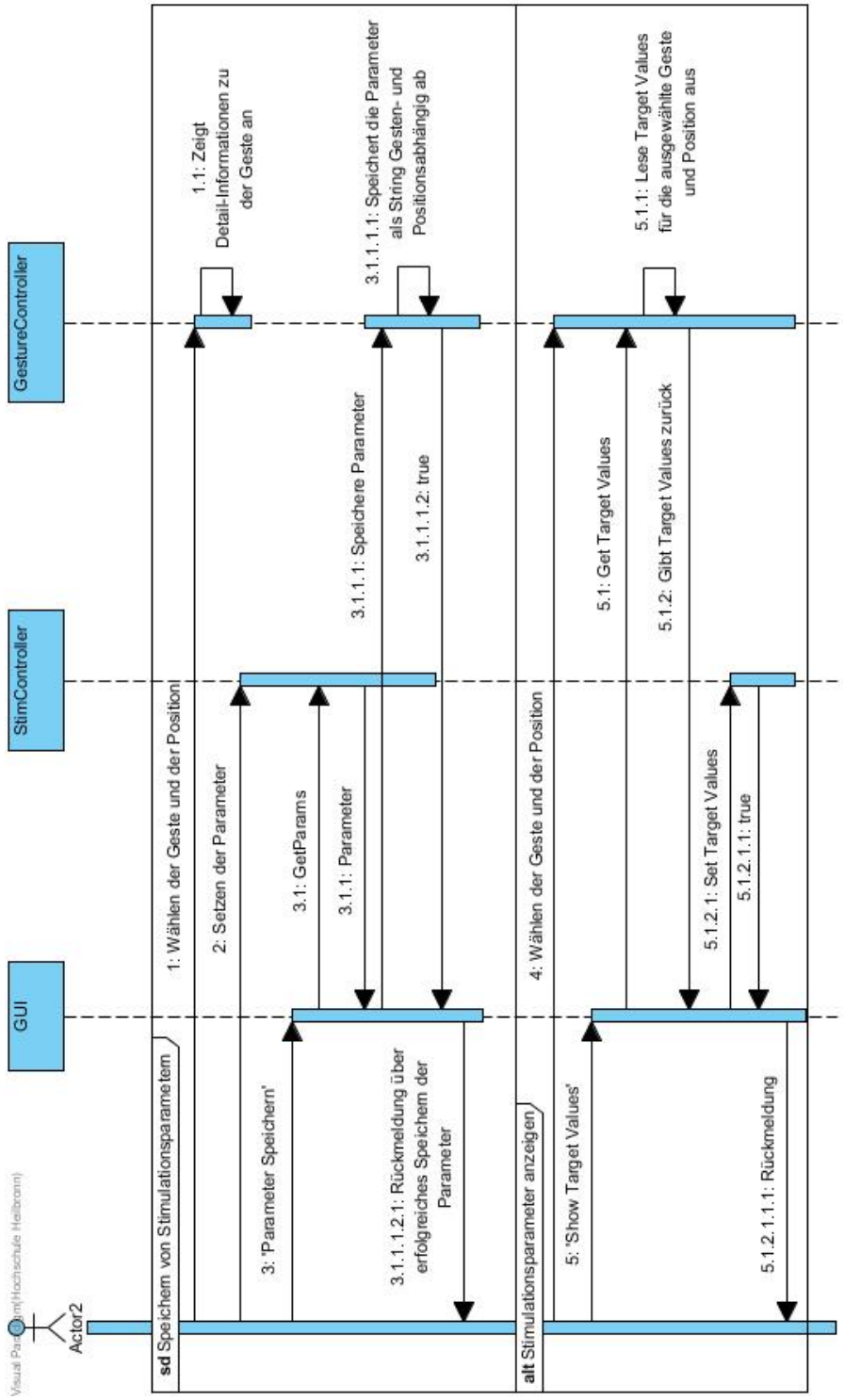


Abbildung A.6: Sequenzdiagramm der Abläufe der stimulatorabhängigen UseCases

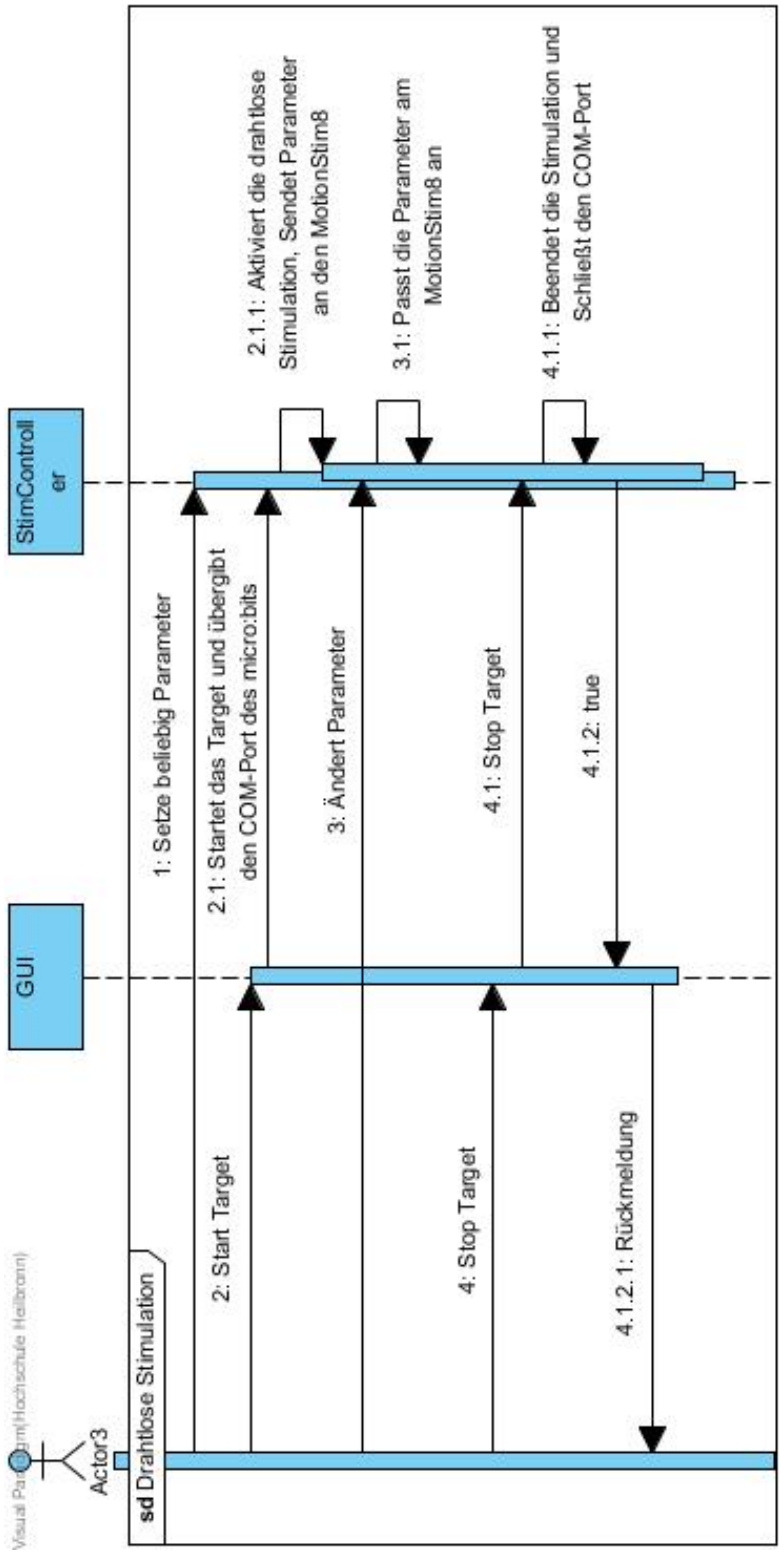


Abbildung A.7: Sequenzdiagramm der drahtlosen Stimulation

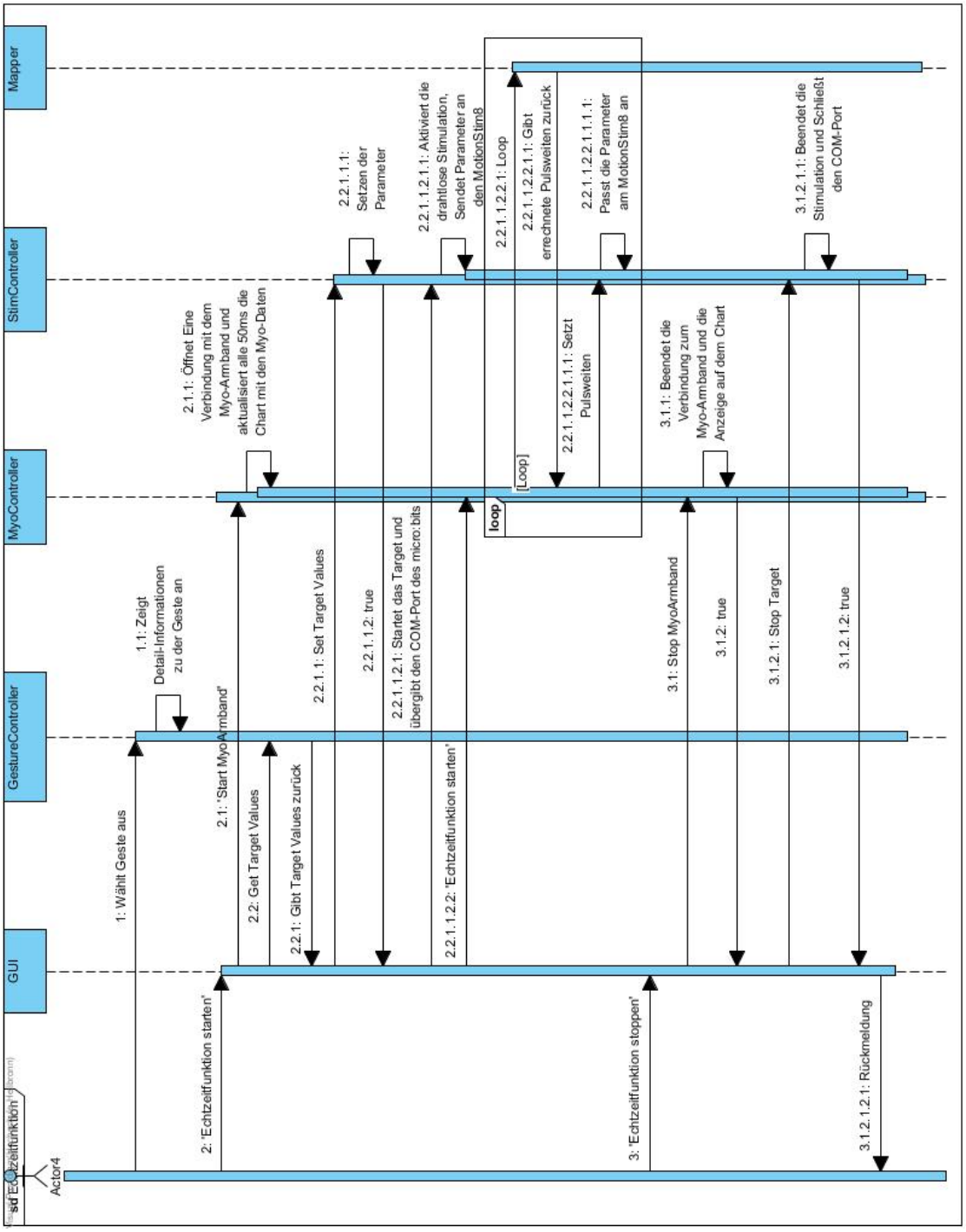


Abbildung A.8: Sequenzdiagramm der Echtzeitfunktion

A.2. Drahtlosfunktion

A.2.1. Python Code

Code 13: Python Code des Senders

```
# Add your Python code here. E.g.
import radio
from microbit import *

#UART wird initialisiert
#Baudrate = 115200
#tx/rx nicht gesetzt -> Standard USB-Port wird verwendet
uart.init(115200)
#Radio-Modul wird gestartet
radio.on()

#Dauerhaft ausgefuehrter Code
while True:
    #Zum Unterscheiden der beiden micro:bit Systeme
    display.show(Image.HAPPY)
    #'hoert' der seriellen Schnittstelle zu
    text = uart.readline()
    #sobald er etwas liest
    if text:
        #sendet den Text an den Empfaenger
        radio.send(text)
        sleep(50)
```

Code 14: Python Code des Empfängers

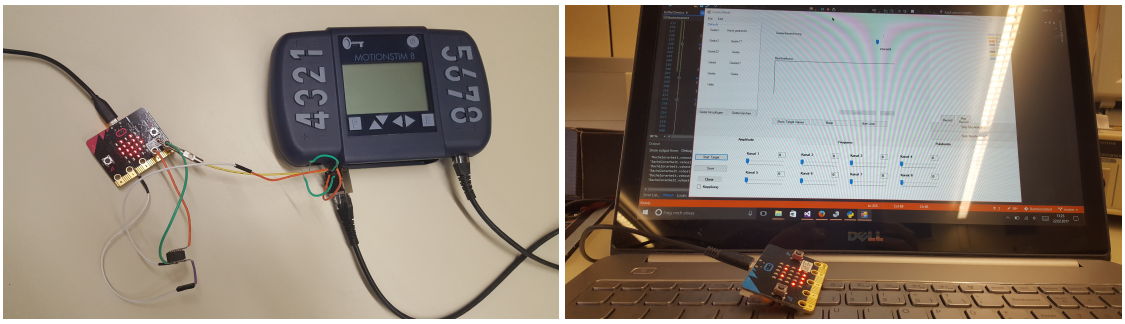
```
# Add your Python code here. E.g.
import radio
from microbit import *

#UART wird initialisiert
#Baudrate = 38400
#tx = pin8, rx = pin12
uart.init(baudrate=38400, tx=pin8, rx=pin12)
#Radio-Modul wird gestartet und konfiguriert
radio.on()
radio.config(queue=10)

#Dauerhaft ausgefuehrter Code
while True:
    #Zum Unterscheiden der beiden micro:bit Systeme
    display.show(Image.HEART)
    #Wartet auf Nachricht des Senders
    text = radio.receive()
    #Sobald eine Nachricht ankommt
    if text:
        if isinstance(text, str):
            #Generierung der Befehlsbytekette
            list = text.split()
```

```
24
25              #DL
26              command = bytearray([0x00])
27              dl = len(list)
28              command[0] = dl-1
29
30              #CID und Data
31              for i in range(0, dl):
32                  x = list.pop(0)
33                  command = command + bytearray([0x00])
34                  command[i+1] = int(x)
35
36              #Checksum
37              s = 0
38              for i in command:
39                  s += i
40              checksum = s % 256
41
42              command = command + bytearray([0x00])
43              length = len(command)-1
44              command[length] = checksum
45
46              #Ergaenzung um SOF
47              command = bytearray([0xFF, 0xFF]) + command
48
49              #Ausgabe an den MotionStim8
50              uart.write(command)
51              sleep(20)
```

A.2.2. Aufbau



(a) MotionStim8-Anbindung (b) PC-Anbindung

Abbildung A.9: Aufbau der Drahtlosverbindung des MotionStim8 mit dem PC

A.2.3. Testdurchlauf

A.2.3.1. Drahtlose Stimulation

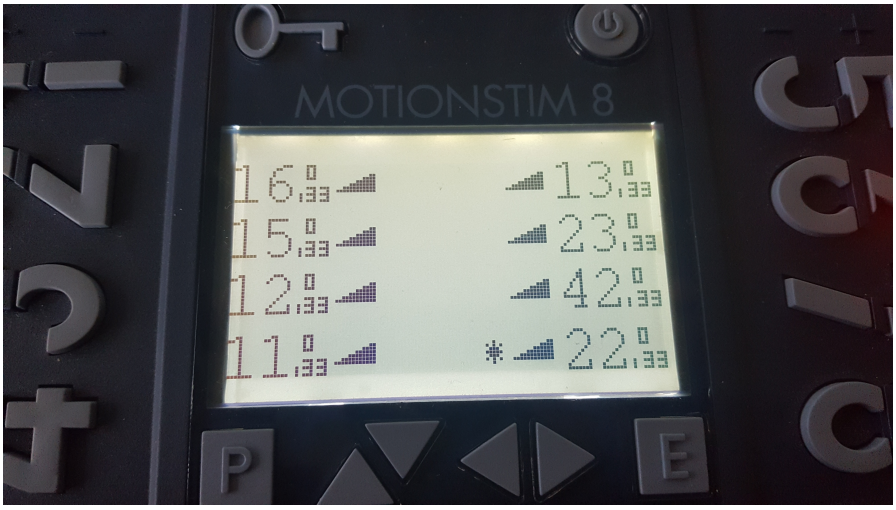


Abbildung A.10: MotionStim8 nach Start der drahtlosen Stimulation

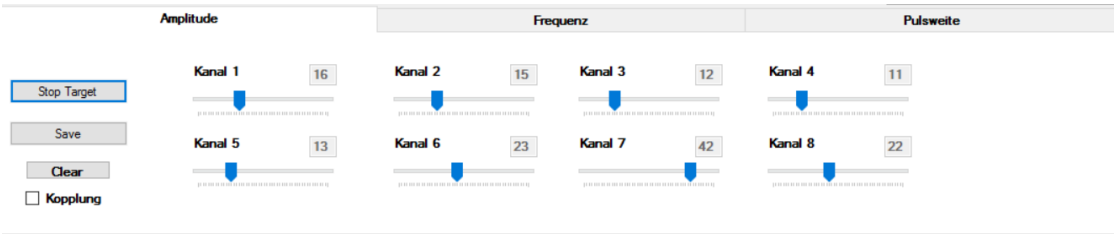


Abbildung A.11: Stimulations Parameter in der Steuersoftware bei Beginn der drahtlosen Stimulation

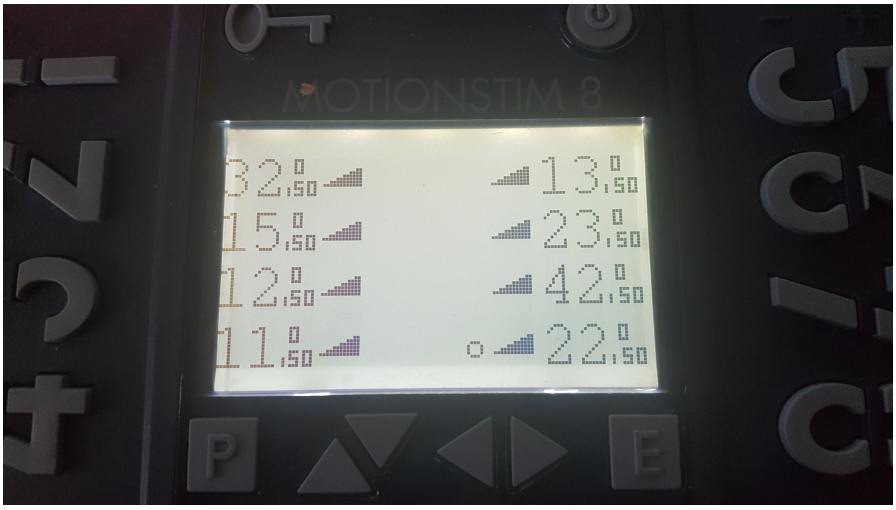


Abbildung A.12: MotionStim8 nachdem die Frequenz in der Steuersoftware verändert wurde

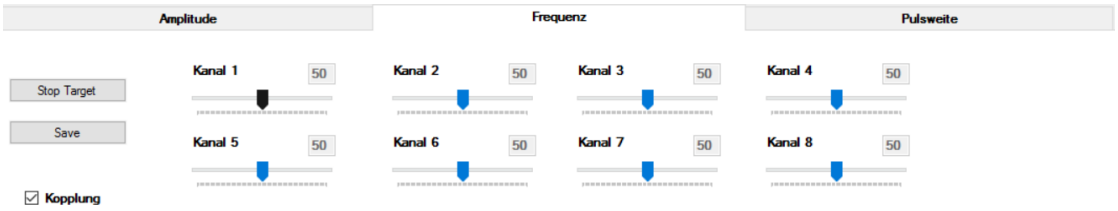


Abbildung A.13: Veränderung der Frequenz über die Kopplungsfunktion bei aktivierter drahtloser Stimulation

Abbildung A.14: MotionStim8 nachdem die Amplitude von Kanal 1 verändert wurde

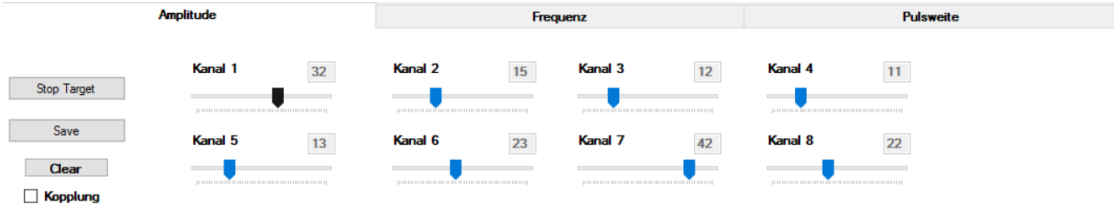


Abbildung A.15: Veränderung der Amplitude des Kanal 1 bei aktivierter drahtloser Stimulation

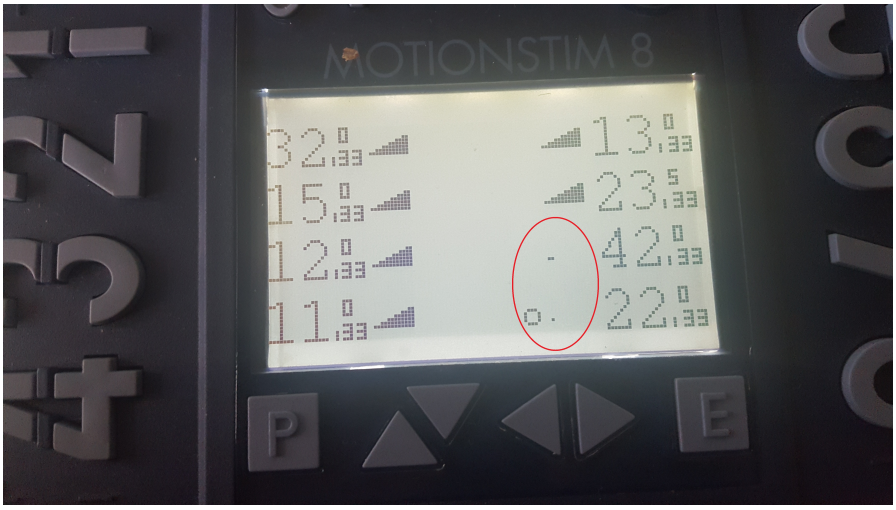


Abbildung A.16: MotionStim8 nachdem Kanäle 7 und 8 deaktiviert wurden

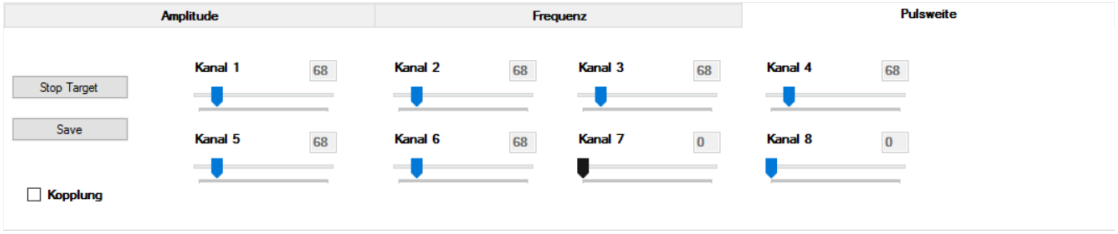


Abbildung A.17: Deaktivieren der Kanäle 7 und 8 durch setzen von 0 als Pulsweite

A.2.3.2. Loop - Echtzeitfunktion

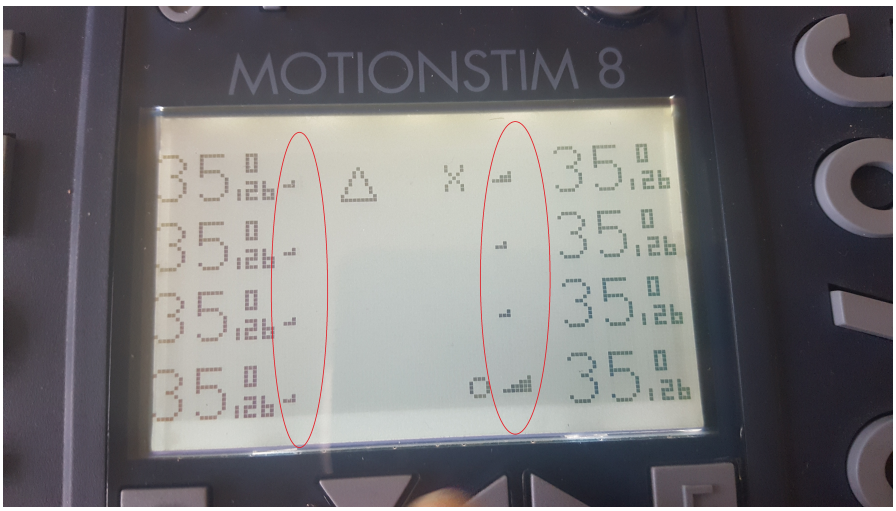


Abbildung A.18: MotionStim8 nach gestarteter Echtzeitfunktion

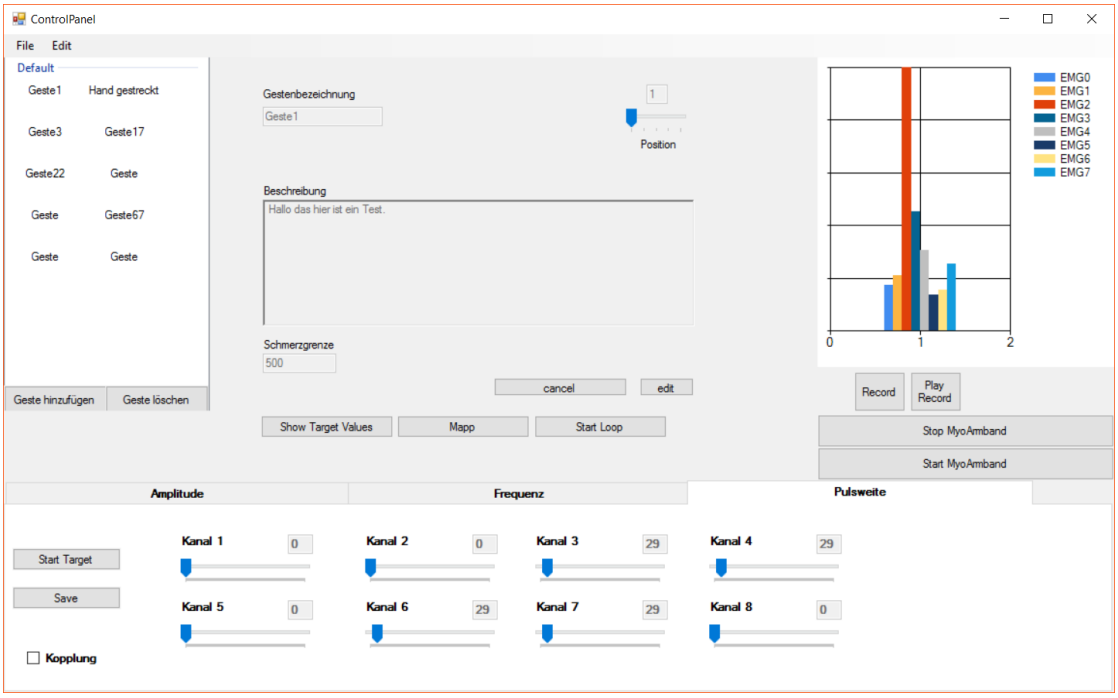


Abbildung A.19: Gestartete Echtzeitfunktion

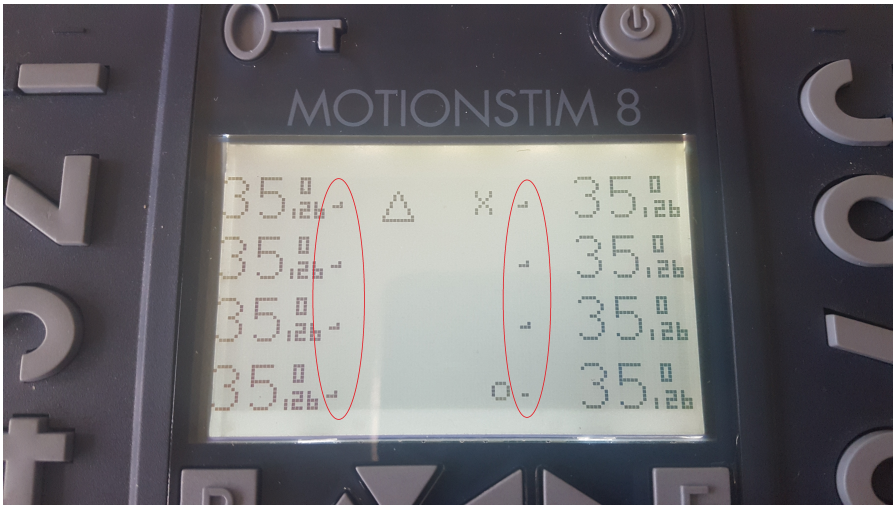


Abbildung A.20: MotionStim8 während der Echtzeitfunktion(nur veränderte Pulsweiten)

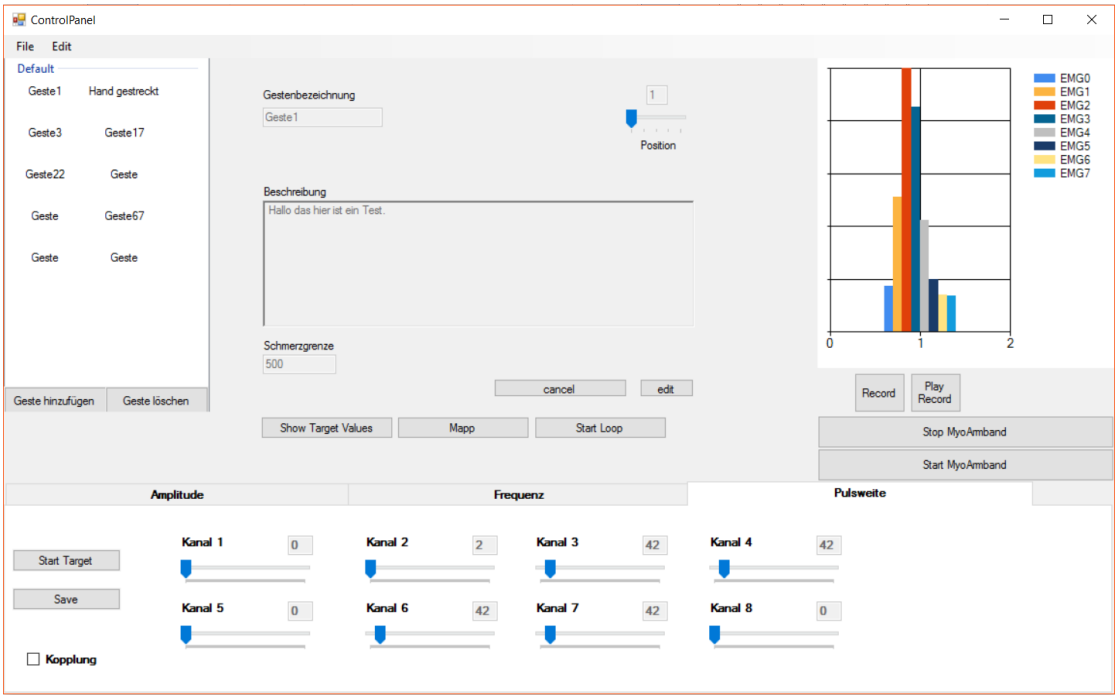


Abbildung A.21: Durch Mapping Matrix und MyoDaten berechnete Pulsweiten in der Steuersoftware