Christian Krammer, BSc

Softwareentwicklung zur Ansteuerung eines Therapiegerätes für Säuglinge

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium Biomedical Engineering

eingereicht an der

Technischen Universität Graz

Betreuer

Assoc. Prof. Dipl.-Ing. Dr.techn. Jörg Schröttner

Institut für Health Care Engineering

Graz, Juli 2020

Eidesstattliche Erklärung
AFFIDAVIT

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Datum	Unterschrift

Die Technische Universität Graz übernimmt mit der Betreuung und Bewertung einer Masterarbeit keine Haftung für die erarbeiteten Ergebnisse: Eine positive Bewertung und Anerkennung (Approbation) einer Arbeit bescheinigt nicht notwendigerweise die vollständige Richtigkeit der Ergebnisse.

Danksagung

Zunächst möchte ich mich natürlich bei Thomas und Dominik, dem „Team Babyreha“, für die hervorragende Zusammenarbeit bei diesem Projekt bedanken. Ich bin mir sicher, dass unser Teamwork einen entscheidenden Einfluss auf das Resultat hatte.

Allgemein möchte ich mich beim Team des Instituts für Health Care Engineering für den reibungslosen Ablauf, sowie die Möglichkeit bedanken, bei einem solch besonderen Projekt mitwirken zu dürfen.

Die Entwicklung, das Tüfteln, das Improvisieren und selbst das gelegentliche Kopfzerbrechen, haben mir bei dieser sehr praktisch orientierten Arbeit viel Freude bereitet.

Auch bei Frau Doktor Karin Prechtl möchte ich mich für das sehr ehrliche und hilfreiche Feedback nach der ersten Vorführung des Therapiegerätes bedanken, in dessen Folge noch einige Dinge optimiert und angepasst werden konnten.

Prim. Dr. Peter Grieshofer möchte ich meinen Dank aussprechen, da er die finanziellen Mittel für das Therapiegerät bereitstellte.

Zu guter Letzt möchte ich mich noch bei meinen Eltern und bei meiner gesamten Familie bedanken, die mich während meines gesamten Studiums in jeder erdenklichen Weise unterstützt haben.

Zusammenfassung

Aufgrund neurologischer Erkrankungen weisen Säuglinge und besonders Frühgeborene häufig anormale Bewegungsmuster auf, die in weiterer Folge zu späteren neurologischen Defiziten, wie zum Beispiel Zerebralparesen führen können. Um dem entgegen zu wirken, gibt es den experimentellen Ansatz in den ersten Lebenswochen normal physiologische Bewegungen zu simulieren und durch diese externen Stimuli den Krankheitsverlauf zumindest zu mildern. Basierend auf einem bestehenden Konzept wurde sowohl ein Ansteuerungskonzept und die zugehörige Software zur Ansteuerung des Therapiegeräts unter Berücksichtigung der Normen EN 62304 und EN 14971 entwickelt. Realisiert wurde dies durch ein Softwaresystem, bestehend aus dem Mikrocontroller „Arduino Mega“, dem GUI „MegunoLink“ und acht Steuergeräten „igus Dryve D1“ des Herstellers, der in den Hubsäulen verbauten Schrittmotoren. Im Rahmen der Risikoanalyse konnten weitere, die Software betreffende Risiken identifiziert und minimiert werden. Die Software wurde erfolgreich verifiziert und zusätzlich teilweise validiert, womit die grundlegenden Anforderungen an Funktionalität und Sicherheit als erfüllt angesehen werden können.

Schlüsselwörter:
EN 62304, Mikrocontroller, Schrittmotor, Risikoanalyse, Bewegungsmuster

Abstract

Due to neurological diseases, some infants, especially preemies, show abnormal movement pattern, that may cause neurological deficits, like cerebral paresis in the future. To counter that, there is an experimental approach to simulate normal, physiological movements in the first postnatal weeks to mild the course of the disease by this external stimuli. Based on an existing concept a controlling concept should be developed as well as a software to control the therapy device considering the standards EN 62304 and EN 14971. The developed software system consists of an „Arduino Mega“ microcontroller, the GUI „MegunoLink“ and eight control devices „igus Dryve D1“ by the stepper motors manufacturer, that are used in the lifting columns. By updating the risk analysis, several software affecting risks could be identified and minimized. The software has been verified successfully and partially validated, what means that the basic requirements to functionality and safety can be seen as fulfilled.

Keywords:
EN 62304, microcontroller, stepper motor, risk analysis, movement pattern

Inhaltsverzeichnis

Abkürzungsverzeichnis

$ALARA$:	As low as reasonably achievable	So niedrig, wie vernünftigerweise erreichbar
$ALARP$:	As low as reasonably practicable	So niedrig, wie vernünftigerweise praktikabel
DC:	Direct current	Gleichstrom
DM:	...	Durchgeführte Maßnahme
$EEPROM$:	Electrically erasable programmable read-only memory	Elektrisch löschbarer programmierbarer Speicher
EW:	...	Eintrittswahrscheinlichkeit
ge:	...	Gering
gl:	...	Gelegentlich
GUI:	Graphical user interface	Grafische Benutzeroberfläche
GS:	...	Gesamtrisiko
h:	...	Häufig
IDE:	Integrated development environment	Integrierte Entwicklungsumgebung
I/O:	Input/Output	Eingang/Ausgang
ma:	...	Manchmal
mi:	...	Mittel
NC:	Normal Case	Normalfall
$NEMA$:	National Electrical Manufacturers Association	Baugrößennorm für Schrittmotoren
PWM:	...	Pulsweitenmodulation
QM:	...	Qualitätsmanagement
RS :	...	Risikostufe
RW:	...	Rückwirkung
S:	...	Schaden
se:	...	Selten
SFC:	Single Fault Condition	Erster Fehlerfall
$SOUP$:	Software of Unknown Provenance	Software unbekannter Herkunft
sw:	...	Schwer
ug:	...	Unglaublich
uw:	...	Unwahrscheinlich

1 Einleitung

Bereits ab der 7. Gestationswoche lassen sich beim menschlichen Fötus erste Bewegungen beobachten, die sich bis zur 12. Gestationswoche zu einer Vielfalt von endogenen Bewegungsmustern entwickeln. Diese rein endogenen Bewegungen bestehen auch bis nach der Geburt weiter, unabhängig vom Geburtszeitpunkt. Aufgrund ihrer Komplexität, langen Dauer, ihres recht häufigen Vorkommens und nicht zuletzt der guten Beurteilbarkeit, werden diese sogenannten „General Movements“ für diagnostische Zwecke herangezogen. Diese Bewegungen dauern spätestens bis zum 5. Lebensmonat an. Zusätzlich entwickelt sich ab dem 3. Lebensmonat die willkürliche Motorik. [5]

1.1 Motivation

General Movements umfassen den ganzen Körper des Kindes, wodurch sich ein hoher diagnostischer Wert zur Erkennung von Erkrankungen des kindlichen Nervensystems ergibt. Durch diverse neurologisch pathologische Erscheinungen, wie zum Beispiel Verletzungen aufgrund von Blutungen und hypoxisch ischämischen Läsionen, aber auch bei Frühgeborenen sind diese Bewegungsmuster deutlich erkennbar verändert. Diese pathologischen Muster lassen sich in „poor repertoire“, „cramped synchronized“ und „chaotic“ einteilen. Nach den sogenannten „Writhing Movements“, welche innerhalb der ersten zwei Monate nach der Geburt auftreten, setzen im Alter von 6-9 Wochen die Fidgety Movements ein. Diese General Movements unterscheiden sich von den Writhing Movements durch eine kleinere Amplitude, niedrigere Geschwindigkeiten und variable Beschleunigung. In diesem Stadium treten die oben genannten pathologischen Muster nicht mehr auf. Die Fidgety Movements können lediglich normal, abnormal sein, oder zur Gänze fehlen. [4]

Die Prechtl-Methode dient dazu, bereits pränatal oder früh postnatal die Bewegungsmuster des Kindes zu beobachten und dadurch potenziell auftretende Krankheiten, wie Zerebralparesen zu diagnostizieren. Fehlende oder abnormale Fidgety Movements führen häufig zu späteren neurologischen Defiziten bzw. Zerebralparesen, während dies bei Kindern mit normalen Fidgety Movements selten beobachtet werden konnte. [5]

1.2 Therapieansatz

Das Ziel des Projektes ist es nun in Kooperation mit der Klinik Judendorf Strassengel und basierend auf der Prechtl-Methode ein experimentelles Therapiegerät zu entwickeln, das genau in diesem Zeitraum Anwendung finden soll, in dem sich diese pathologischen Bewegungsmuster manifestieren. Das Gerät wird dann zwischen dem errechneten Geburtstermin und der 14. Lebenswoche zum Einsatz kommen. Durch extern gesetzte motorische Stimuli soll hierbei die Entwicklung gesunder Bewegungsmuster des Patienten unterstützt werden. Diese Stimuli sollen in Form von simulierten gesunden Bewegungen durch das Therapiegerät erfolgen um die Entwicklung von Fidgety Movements zu begünstigen. Als erhofftes Resultat dieser Therapie sollen die aus fehlenden oder abnormalen Fidgety Movements folgenden neurologischen Defizite zumindest gemildert werden.
[10]

2 Aufgabenstellung

Basierend auf einem bereits entwickelten Konzept [10] und sich in Entwicklung befindender Hardware [11], soll eine entsprechende Software, sowie die Ansteuerung für ein Therapiegerät für Säuglinge konzipiert und implementiert werden. Dieses Therapiegerät dient der Verbesserung und Anregung frühkindlicher Bewegungsmuster durch mechanische taktile Reize zur Stimulation von Neuronen und Muskulatur. Ebenso soll bei dieser Arbeit ein besonderer Fokus auf die zutreffenden Normen und Richtlinien gelegt und diese so weit wie möglich umgesetzt werden. Des Weiteren hat die Entwicklung der Software, insbesondere der Benutzeroberfläche entsprechend den Anforderungen seitens der Klinik zu erfolgen.

Folgende Normen und Richtlinien sollen bei der Entwicklung von Ansteuerung und Software berücksichtigt werden:

- OVE EN 62304
  Medizingeräte-Software - Software-Lebenszyklus-Prozesse
- EN ISO 14971
  Medizinprodukte - Anwendung des Risikomanagements auf Medizinprodukte
- 93/42/EWG - Medizinproduktdirektive
- 2006/42/EG - Maschinenrichtlinie

Im Rahmen der Softwareentwicklung an sich, ist die Umsetzung der vorgegebenen Bewegungsmuster aus der Masterarbeit von Herrn Mauerhofer [10] in Form von zyklischen Bewegungen unter Berücksichtigung unterschiedlicher Therapieparameter und einer entsprechenden Fehlerverarbeitung zu bewerkstelligen. Des Weiteren sollen sicherheitsrelevante Parameter berücksichtigt und eine Aktualisierung der Risikoanalyse, sowie eine Funktionsüberprüfung der Ansteuerung und des Sicherheitskonzeptes vorgenommen werden.

Nach Abschluss der praktischen Arbeiten soll eine technische Dokumentation entsprechend der Normvorgaben, sowie die erste Fassung einer Gebrauchsanweisung entworfen werden.

3 Methoden

3.1 Ausgangssituation

Zum Startzeitpunkt, dieser Arbeit existierte bereits ein erstes grundlegendes Konzept des Therapiegerätes, damals noch mit sechs Hubsäulen (Abbildung 3.1), welches von den Kollegen Mauerhofer und Steindorfer im Rahmen ihrer Masterarbeiten entwickelt wurde.

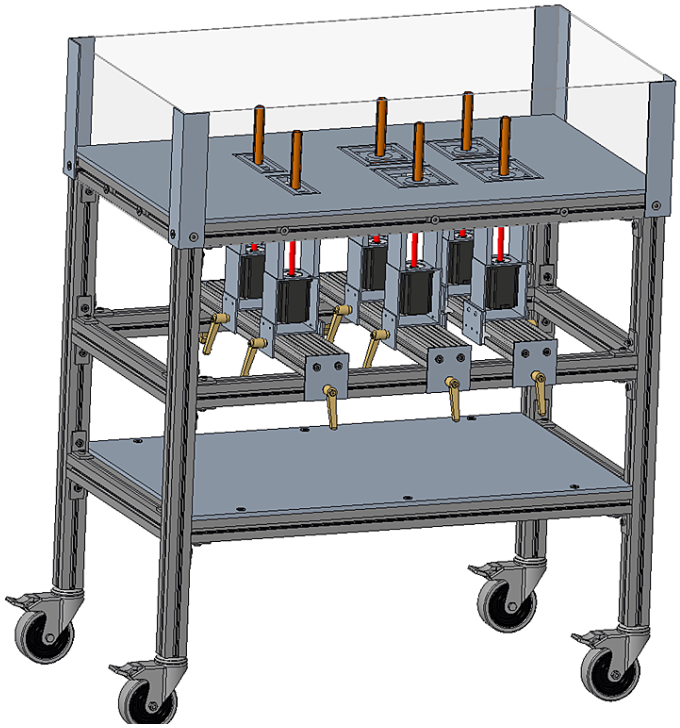


Abbildung 3.1 Erstes Grundkonzept des Therapiegerätes [10]

Im Laufe der Entwicklung wurden mehrere Varianten dieses Konzeptes mit unterschiedlichen Anzahlen an verwendeten Hubsäulen (6-12) ausgearbeitet und bewertet. [10]

3.1.1 Aufbau des Therapiegerätes

Nach Abwägung von Aufwand und Nutzen, bzw. Absprache mit der Klinik wurde letztendlich ein Konzept mit acht Hubsäulen gewählt.
Bei dem final gewählten Konzept des Therapiegerätes (Abbildung 3.2) werden die in Kapitel 2 erwähnten motorischen Stimuli durch 8 Hubsäulen durch eine Auf-und Ab-Bewegung

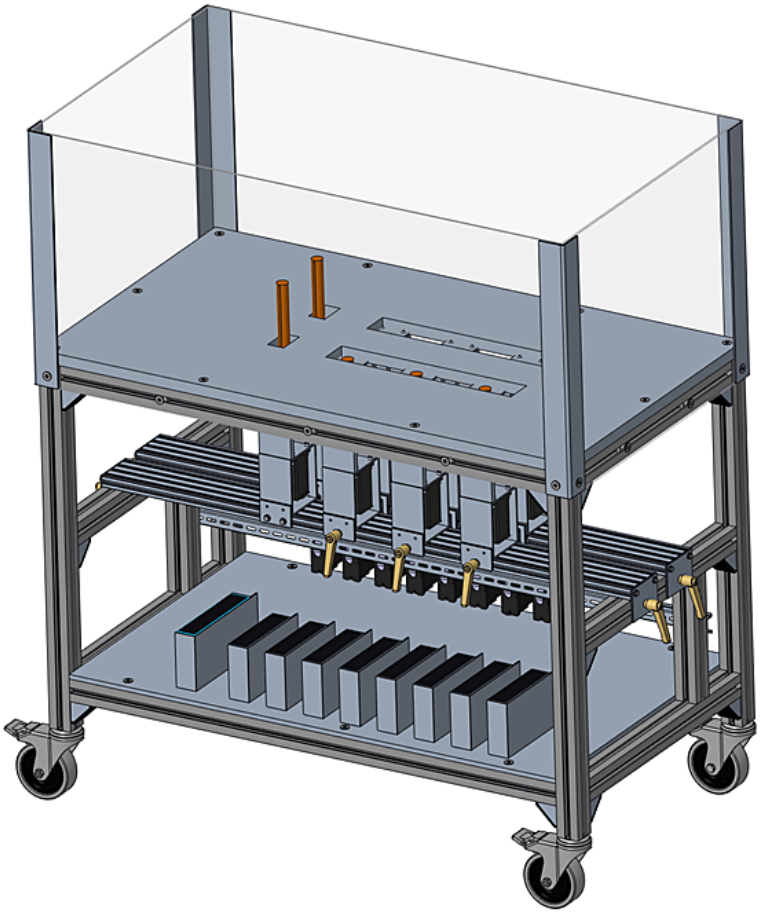


Abbildung 3.2 Finales Konzept des Therapiegerätes mit 8 Hubsäulen und Technikebene und erster Einschätzung der Netzteile [10]

eines Hubaufsatzes erzeugt. Diese Hubsäulen werden jeweils durch einen Schrittmotor der Firma „igus GmbH“ der Type NEMA-23 und einem Nennstrom von 4,2 A angetrieben. Für die folgende Konzeptfindungsphase hinsichtlich der Ansteuerung waren die Daten des Schrittmotors sowie die Spezifikation, dass das Therapiegerät über eine möglichst einfach gestaltete grafische Benutzeroberfläche (GUI) angesteuert werden soll, die beiden ausschlaggebenden Kriterien. Ebenso sollten die Bewegungssequenzen so automatisiert wie möglich ablaufen, also jede Säule sollte nicht manuell angesteuert werden müssen.

3.2 Konzeptfindung

Zu Beginn der Arbeit wurden mehrere Konzepte/Möglichkeiten zur Ansteuerung und Programmierung des Therapiegerätes ausgearbeitet und deren Vor- und Nachteile bewertet.

Allgemeines zu „Arduino“[3]:
Arduino ist eine aus Hardware und Software bestehende Plattform mit freier Lizenz rund um den gleichnamigen Mikrocontroller, der in unterschiedlichen Ausführungen erhältlich ist. Sie erlaubt es, die Mikrocontroller mit einer Programmiersprache, die C und C++ sehr ähnlich ist, über ein zugehöriges IDE auf einfachem Wege zu programmieren. Die Arduino Plattform wird von vielen Hobbyentwicklern für unterschiedlichste Anwendungsbereiche genutzt und ist darum unter anderem weit verbreitet, weshalb eine Vielzahl von Informationen und Erfahrungsberichten online zur Verfügung steht. Außerdem gibt es sogenannte „libraries“, also Bibliotheken, in denen komplexere Funktionen oder Schnittstellen anderen Benutzern frei zur Verfügung gestellt werden.

Aufgrund der hier genannten Vorteile und Eigenschaften dieser Plattform, ist die Entscheidung gefallen, eine Ansteuerung auf Basis eines Arduino-Mikrocontrollers zu entwickeln.

Die grafische Benutzeroberfläche „MegunoLink“[9]:
MegunoLink ist eine Software von der Firma „Number Eight Innovation Limited“, die für die Erstellung und den Betrieb von Benutzeroberflächen von Arduino-Programmen eingesetzt wird. Mit dieser, auf einem normalen Windows PC installierten Software ist es möglich sowohl Daten aus einem laufenden Programm auszulesen, als auch Daten und Befehle an den Mikrocontroller in Echtzeit zu senden. Die laufende Kommunikation findet hierbei über die serielle Schnittstelle USB statt, über welche der Arduino auch über die IDE programmiert wird. Über vom Hersteller bereitgestellte Bibliotheken kann die Benutzeroberfläche einfach und unkompliziert in den Programmcode eingebunden werden. Die Schaltflächen, Textfelder, andere Bedienelemente und Anzeigen werden direkt in Megunolink über einen Editor erstellt und angepasst. In dem später erstellten Programm zur Ansteuerung des Therapiegerätes wird MegunoLink primär für die Eingabe von Befehlen an den Mikorcontroller genutzt.

3.2.1 Verworfene Konzepte

Ansteuerung über „LabVIEW“

Wie schon beim „Lokomaten“, der ebenfalls am Institut für Health Care Engineering im Rahmen von mehreren Masterarbeiten entwickelt wurde, gab es Anfangs die Überlegung das Therapiegerät mittels der Systementwicklungssoftware, „LabVIEW“ anzusteuern. Die Vorteile an dieser Methode wären die recht einfache Implementierung einer grafischen Benutzeroberfläche direkt über LabVIEW und das am Institut vorhandene Know-How. Dem gegenüber standen aber eine begrenzte Anzahl an kompatiblen Schrittmotortreibern, welche zusätzlich nicht genug Strom für die gewählten Schrittmotoren liefern würden, als auch sehr beschränkte Informationen online bezüglich dieser Ansteuerungsmethode.

Ansteuerung durch „Arduino“ und „Motor-Shields“

Die erste Option zur Realisierung einer Ansteuerung des Therapiegerätes mittels Arduino bestand aus einer Kombination des Arduinos mit einem sogenannten „Motor-Shield“. Shields sind die Bezeichnungen für Hardware-Erweiterungen des Arduino Systems, welche in der Regel aus einer Platine bestehen, die auf die Hauptplatine des Mikrocontrollers aufgesteckt werden. Unter anderem existieren Shields für Sensoren, zur Kommunikation via WiFi oder Bluetooth und eben zur Ansteuerung von DC- oder Schrittmotoren. Zu den meisten dieser Shields gibt es passende Bibliotheken für die Arduino IDE zu deren einfachen Verwendung. Der einzige, dafür aber ausschlaggebende Nachteil dieser Ansteuerungsmethode ist die für unsere Zwecke zu niedrige elektrische Leistung. Die verwendeten Schrittmotoren werden mit einem Nennstrom von 4,2 A und einer Spannung zwischen 24V und 48V betrieben. Keiner der im Rahmen der Recherchen gefundenen Motor-Shields konnte diesen Anforderungen gerecht werden.

Ansteuerung durch „Arduino“ und Schrittmotortreiber

Als weitere Möglichkeit zur Ansteuerung des Therapiegerätes wurden herkömmliche Schrittmotorentreiber in Betracht gezogen. Diese Schrittmotorentreiber werden direkt über Pulsweitenmodulation (PWM) durch einen Mikrocontroller (Arduino) angesteuert, was aber einen deutlich höheren Programmieraufwand zur Folge hätte. Des Weiteren besteht bei den Standardmodellen, wie auch bei den Arduino-Shields das Problem einer unzureichenden Strom- und Spannungsversorgung. Lediglich professionelle Modelle im höheren Preissegment würden den Anforderungen an die elektrische Versorgung gerecht werden. Aufgrund der ausschlaggebenden Vorteile der letztendlich gewählten Variante, wurde auch dieses Konzept verworfen.

3.3 Gewähltes Ansteuerungskonzept

Nach Abwägung der Vor- und Nachteile wurde für die Ansteuerung des Therapiegerätes eine Kombination aus Arduino und einem Steuergerät der Firma „igus“ dem „igus Dryve D1“ [7] gewählt. Bei dieser Ansteuerungsvariante wird je ein Schrittmotor von einem Steuergerät (im weiteren auch einfach nur als „Dryve“ bezeichnet) angesteuert. Alle Steuergeräte werden wiederum über einfache Signalleitungen vom Arduino kontrolliert, der via USB mit einem Notebook verbunden ist.

Gegenüberstellung der Vor- und Nachteile

Vorteile:

- Motor und Steuergerät sind vom gleichen Hersteller:
  Da Motor und Steuergerät vom gleichen Hersteller sind und diese, laut dessen Angaben, auch garantiert kompatibel sind, ist sowohl eine optimale Funktionalität, als auch die korrekte Spannungsversorgung der Schrittmotoren gewährleistet, sofern das Steuergerät entsprechend versorgt wird. Langfristig gesehen birgt dies auch Vorteile für die Wartung und eventuelle Fehlerbehebungen, sollte der Hersteller igus hinzugezogen werden müssen.

- Einfache Verbindung und Programmierung:
  Arduino und Steuergerät werden über sogenannte Kabelbrücken, auch „Jumper Wires“ genannt, gebündelte Signalkabel und eine Platine (während der Entwicklungsphase über ein Steckbrett) miteinander verbunden. Im Arduino-Code werden dann die im Dryve vorprogrammierten Fahrprofile gestartet. Somit müssen die einzelnen Bewegungen nicht extra im Arduino Code programmiert werden.

- Zusätzliche Sicherheit durch Beschränkungen am Steuergerät:
  Am Steuergerät selbst können Grenzwerte für Auslenkung, Geschwindigkeit und Beschleunigung festgelegt werden. Die Bewegungsprofile können ebenso wie die Grenzwerte nur über ein Passwortgeschütztes Webinterface verändert werden. In Kombination mit ihrer hohen Zuverlässigkeit tragen die Steuergeräte somit deutlich zur Gesamtsicherheit bei.

Nachteile:

- Höhere Kosten:
  Verglichen mit den in Kapitel 3.2.1 genannten Alternativen sind die Kosten für dieses Ansteuerungskonzept deutlich höher. Dies liegt hauptsächlich am Stückpreis der Steuergeräte von 338,25 €.

- Beschränkung auf vorprogrammierte Fahrprofile:
  Aus dem Vorteil der Beschränkung auf vorprogrammierte Fahrprofile (im Folgenden auch als Bewegungen bezeichnet) ergibt sich somit auch der Nachteil, dass eben nur vorprogrammierte Bewegungen im Programmcode des Arduino aufgerufen werden können. Da diese Bewegungen aber durch Anpassung von Position, Geschwindigkeit und Beschleunigung in einem für unsere Ansprüche absolut ausreichenden Grad konfigurierbar sind, kann dieser Nachteil somit vernachlässigt werden.

- Limitierung auf 32 Fahrprofile:
  Auch die Limitierung auf 32 (5-Bit) vorprogrammierbare Fahrprofile fällt nicht weiter ins Gewicht, da wie folgend in Tabelle 4.3 gezeigt, für die implementierten Bewegungssequenzen nur einige wenige Fahrprofile benötigt werden.

- Kein Zugriff auf den Quellcode der Steuergeräte:
  Da der Hersteller keinen Zugriff auf die Steuerungssoftware gewährt, sind die Steuergeräte laut EN 62304 als SOUP zu klassifizieren. Daraus resultierende potenzielle Gefahren wurden später im Rahmen des Risikomanagements auf ein akzeptables Niveau reduziert. (siehe Kapitel 4.5)

3.3.1 Aufbau und Hardwarekomponenten

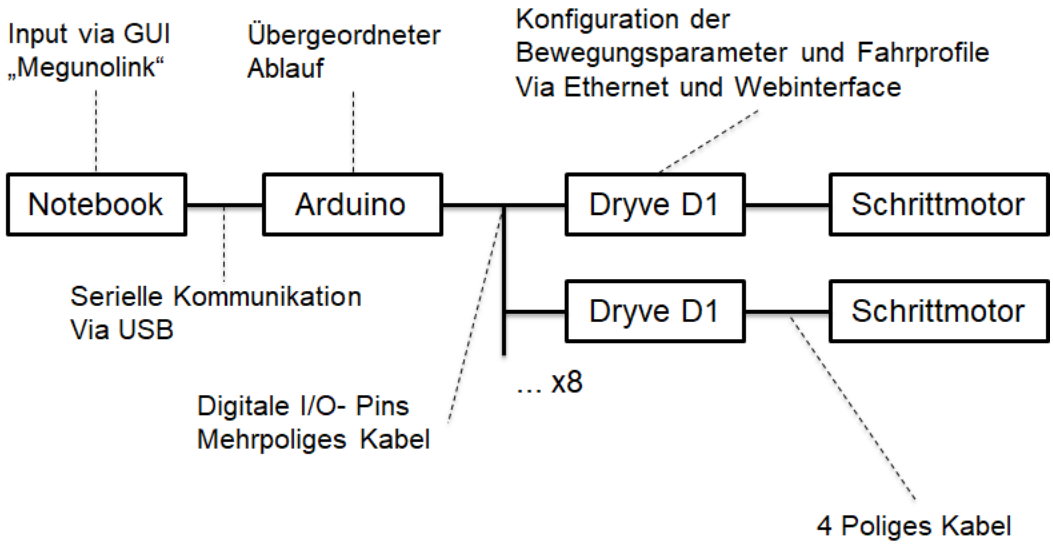


Abbildung 3.3 Aufbau der Ansteuerung

Abbildung 3.3 stellt den grundlegenden Aufbau der Ansteuerung des Therapiegerätes dar. Die acht Schrittmotoren sind jeweils über ein vierpoliges Kabel mit einem Steuergerät (igus Dryve D1) verbunden. Jedes Steuergerät wird über ein medizinisches 5V- und 24V-Netzteil mit Spannung versorgt. Über mehrpolige Kabel und eine Platine sind die digitalen Ein- und Ausgänge der Steuergeräte des Weiteren mit dem Microcontroller (Arduino Mega 2560 Rev3 [3]) verbunden. Auf dem Arduino selbst läuft der übergeordnete Programmcode, die eigentliche Ansteuerungssoftware und somit der wichtigste und komplexeste Bestandteil der gesamten Ansteuerung. Die Ein- und Ausgaben an/durch das Programm geschehen über die grafische Benutzeroberfläche, MegunoLink [9] auf einem handelsüblichen Windows 10 Notebook. MegunoLink kommuniziert über die serielle Schnittstelle USB mit dem Arduino Mega.

Hardware Komponenten der Ansteuerung:

- Windows 10 Notebook (nicht weiter spezifiziert):
  Primär als Plattform zur Ausführung von MegunoLink. Wird auch zur Programmierung des Arduino und der Vorkonfiguration der Steuergeräte verwendet.

- Arduino Mega 2560 Rev 3 :
  Der Mikrocontroller auf dem ein Großteil Ansteuerungssoftware läuft. Aufgrund des meisten verfügbaren Speichers und der höchsten Anzahl an I/O Pins wird die „Mega“ Ausführung des Arduino verwendet.

- igus dryve D1 Motorsteuerung:
  Dient der direkten Ansteuerung der Schrittmotoren, sowie der Konfiguration und Speicherung der Fahrprofile.

- Medizinische Schaltnetzteile der Firma „Mean Well“:
  Sie dienen der Spannungsversorgung der beiden Spannungsebenen (5V - „MSP 100-5“ und 24V - „MSP 200-24“) der Steuergeräte und somit auch der Schrittmotoren.

3.3.2 Funktionsprinzip

Ein Therapieprogramm besteht aus bis zu 20 Sequenzen, welche aus einem Pool von 22 vorgegebenen Bewegungssequenzen über die Benutzeroberfläche ausgewählt werden können. Diese Bewegungssequenzen, die laut den Vorgaben aus der Arbeit von Herrn Mauerhofer [10] implementiert wurden, beinhalten eine Kombination aus mehreren unterschiedlichen aufeinander abgestimmten Fahrprofilen/Bewegungen der Hubsäulen. (siehe Anhang F) Die Eingaben vom User Interface MegunoLink werden mittels USB seriell in Echtzeit an den Arduino Mikrocontroller im Therapiegerät übertragen. Rückmeldungen bezüglich auftretender Fehler, diverse Texthinweise, die ausgewählten Sequenzen und Patienten, sowie der aktuelle Stand des Therapieprogramms werden ebenso via USB vom Arduino an die Benutzeroberfläche gesendet.

Der Mikrocontroller selbst ist durch seine Pins über ein mehradriges Signalleitungskabel mit den Digitalen Ein- und Ausgängen des Steuergeräts „Dryve“ zusätzlich über eine Platine verbunden. Das Steuergerät erhält auf diesem Weg seine Steuereingaben und retourniert gegebenenfalls ein Feedback.

Ansteuerung des Schrittmotorensteuergeräts

Die folgenden Informationen stammen von der Firma igus bzw. dem Handbuch des Steuergeräts [6] und dem Webinterface des igus Dryve D1. Das Steuergerät wird über ein Webinterface konfiguriert, indem es zunächst via Ethernet Kabel mit einem PC verbunden wird. Über einen Webbrowser wird das Interface mit der IP Adresse „192.168.0.10“ aufgerufen. Hier werden grundlegende Einstellungen, wie Motorspezifikation/Type, Vorschub, Grenzwerte der Bewegungsparameter (Verfügbarer Hub, max. Beschleunigung, max. Geschwindigkeit) festgelegt. (Nähere Details sind in Anhang D einzusehen)

Digitale Ein- und Ausgänge:
Nur die verwendeten Ein- und Ausgänge werden beschrieben.

- Digital Eingänge 1-5: Bit 0 - Bit 4
  Durch entsprechendes Setzen der 5 Bit wird aus den bis zu 32 vorprogrammierten Fahrprofilen ausgewählt.

- Digital Eingang 6: Start
  Das gewählte Fahrprofil wird gestartet.

- Digital Eingang 7: Freigabe
  Aktiviert die Bestromung des Schrittmotors

- Digital Eingang 9: Endlagenschalter negativ
  Wird nicht mit dem Arduino verbunden. Dient der Referenzierung der Hubsäule mit Hilfe eines „normaly- open"- Endlagenschalters.

- Digital Eingang 10: Stopp/Reset
  Das aktive Fahrprofil wird unmittelbar gestoppt. Fehler und Alarme werden quittiert.

- Digital Ausgang 1: Bereit
  Die jeweilige Hubsäule ist im Stillstand und kann einen (neuen) Fahrbefehl annehmen.

- Digital Ausgang 4: Alert/ Warnung
  Ein vom Hersteller als „Warnung" vordefinierter Sonderfall ist am jeweiligen Motor/Steuergerät aufgetreten. (Siehe D1 Handbuch V2.0 S.88 [6])

- Digital Ausgang 5: Error/ Fehler
  Ein vom Hersteller als „Fehler" vordefinierter Sonderfall ist am jeweiligen Motor/-Steuergerät aufgetreten. (Siehe D1 Handbuch V2.0 S.88.f [6])

Zur Reduktion der Anzahl an verwendeten Pins am Arduino wurde auf einer Platine eine Art Bussystem für die 5-Bit Binär Pins, sowie die Freigabe Pins hergestellt. Bei diesem Pseudo Bussystem werden alle 8 x 5-Bit Pins, sowie 8x1 Freigabe Pin, je miteinander verbunden und von einem Ausgang am Arduino angesteuert. Somit werden für acht Steuergeräte nur fünf Binär Pins + ein Freigabe Pin am Mikrocontroller benötigt. Funktionell ergeben sich dadurch keine Einschränkungen, da nun entweder alle Motoren, oder keiner bestromt werden. Die Bestromung einzelner Motoren wäre in der verwendeten Konfiguration ohnehin nicht von Bedeutung. Zur Ansteuerung bzw. Ingangsetzung der Fahrprofile wird nun an allen Steuergeräten das gleiche Profil über die 5 Bit gewählt, aber nur am gewünschten Steuergerät ein Start-Signal gesetzt, woraufhin sich lediglich diese Hubsäule in

Bewegung setzt.

Aufbau der Fahrprofile:
Durch Eingabe des verwendeten Motors, sowie der Steigung der in der Hubsäule verwendeten Spindel im Webinterface des Steuergerätes werden die Parameter für die resultierenden Linearbewegungen der Hubsäule direkt im Steuergerät berechnet. Somit ist es möglich die linearen Bewegungsparameter direkt ins Steuergerät einzuprogrammieren, ohne sie zuvor in eine rotatorische Bewegung umrechnen zu müssen.

Prinzipiell setzt sich jedes Fahrprofil aus den folgenden Parametern zusammen:

- Modus:
  „HOM“ zur Durchführung einer Referenzierung,
  „ABS“ um eine Position anzufahren.
- Ziel (mm):
  Zielposition der Bewegung
- Beschleunigung (mm/s^2)
- Geschwindigkeit (mm/s)
- Verzögerung (mm/s^2)

Beispiele für die in ein Steuergerät eingespeicherten Fahrprofile befinden sich in Tabelle 4.3.

3.3.3 Spannungsversorgung

Laut Datenblatt wird der verwendete „igus“ NEMA 23 Schrittmotor mit einem Nennstrom von 4,2 A bei einer Spannung von 24V - 48V betrieben. Bei den zugehörigen Steuergeräten müssen unterschiedliche Funktionseinheiten auf mehreren Spannungsebenen versorgt werden (D1 Handbuch V2.0 S.14 [6]):

- Logik: 12V - 24V (75mA - 225mA)
- Last: 12V - 48V (5mA - 22A)
- Digitale I/O: 5V - 24V (35mA - 1,1A)

Die Pins des Arduino Mega arbeiten in einem Bereich von 0V - 5V (LOW-HIGH). Die Low/High Schwelle der Digitalen Ein- und Ausgänge des Steuergeräts hängt nun von der entsprechenden Versorgungsspannung ab (U_{LOW} = 10%U, U_{HIGH} = 60%U; D1 Handbuch V2.0 S.15 [6]). Folglich wird eine Versorgungsspannung von 5V benötigt, um den Arduino mit dem Steuergerät gemeinsam betreiben zu können. Des weiteren werden sowohl Last, als auch Logik mit 24V versorgt, um diese beiden Funktionsgruppen auf eine Spannungsebene zusammenzufassen und somit Aufwand und Komplexität der Spannungsversorgung zu reduzieren.

Aus Gründen der Sicherheit fiel die Wahl der Netzteile auf medizinische Schaltnetzteile der Firma „Mean Well“. Die Netzteile erfüllen unter anderem die Vorgaben der Normen EN 60601-1 und der EMV-Richtlinie EN 61000-4-2 und besitzen automatische Schutzeinrichtungen gegen Überlast und Überhitzung. [12], [13]

Eingesetzte Schaltnetzteile:

- **Mean Well MSP 100-5** [12]: 5V (17A) - Digitale I/O
- **Mean Well MSP 200-24** [13]: 24V (8,4A) - Logik, Last

Die Schaltnetzteile werden primärseitig mit 230V über Reihenklemmen versorgt. Sekundärseitig werden die Steuergeräte ebenso über ein System aus Reihenklemmen mit Spannung versorgt. (Siehe Stromlaufpläne in Anhang A) Wie aus Gleichung 3.1 hervorgeht ist es möglich die 5V Spannungsebene aller Steuergeräte mit einem Netzteil bei maximaler Last zu versorgen, da der Gesamtstrom I_{ges} niedriger ist, als der maximale vom Netzteil zur Verfügung gestellte Strom.

$$I_{ges} = 8 \cdot 1,1A = 8,8A < 17A \tag{3.1}$$

Für die 24V Netzteile war es hingegen schwierig einzuschätzen, wie viel Strom im Betrieb tatsächlich benötigt werden würde. Daher wurde in der Planungsphase sicherheitshalber ein 24V Netzteil pro Steuergerät einkalkuliert.

Durchgeführte Messungen zur Abschätzung der maximalen Stromaufnahme

Bereits in Phase 1 der Entwicklung (siehe Kapitel 3.5.2) wurde mit einem Leistungsmessgerät die Leistungsaufnahme an der Primärseite (230V) gemessen, um den Leitungsquerschnitt entsprechend dimensionieren zu können. An einem 24V Netzteil + 5V Netzteil + Steuergerät + Motor und Spindel wurde im Betrieb an der Primärseite eine Stromaufnahme von 400mA gemessen.

$$I_{prim} = 8 \cdot 500mA = 4A \tag{3.2}$$

Mit einem zusätzlichen Sicherheitsfaktor wurde die voraussichtliche Stromaufnahme von allen Motoren mit je einem Netzteil ausgerechnet (Gleichung 3.2.) Entsprechend ÖVE-EN 1 Teil 3 wurde der Kabelquerschnitt für die Primärseite auf 1,5mm^2 dimensioniert. Ebenso wurde die Stromaufnahme an der Sekundärseite eines 24V Netzteils an der V- Leitung mit einem digitalen Multimeter (Fluke 87) gemessen. Bei der ersten Messreihe wurde ein Schrittmotor + Steuergerät versorgt, bei der zweiten Messung je zwei Schrittmotoren + Steuergeräte. In Entwicklungsphase 3 (Kapitel 3.5.2) wurde nach Fertigstellung der Hubsäulen und vor der Endmontage des Therapiegerätes eine weitere Messung an der V- bzw. der 0V Leitung an Klemme X1.2 der Steuergeräte vorgenommen. Diesmal mit je vier Motoren und Steuergeräten sekundärseitig an einem 24V Schaltnetzteil.

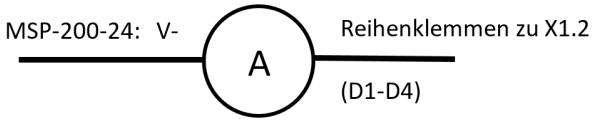


Abbildung 3.4 Messschaltung zur Ermittlung des Stroms an der Sekundärseite. Links das 24V Netzteil und rechts die Reihenklemmen, über welche die Steuergeräte „Dryve 1“- „Dryve 4“ versorgt werden.

3.4 Fortsetzung des Risikomanagements nach EN ISO 14971

Im Zuge dieser Arbeit wurde die Risikoanalyse laufend ergänzt. Als Grundlage hierfür dient das bestehende Risikomanagement der Vorgängerarbeiten ([10], [11]), welches hier aber nicht aufgeführt wird. Die Risikoanalyse wurde entsprechend der Norm EN 14971 [1] durchgeführt. Detailliertere Informationen zu diesen Vorgängen sind der Arbeit von Herrn Mauerhofer [10] oder direkt der Norm EN ISO 14971 zu entnehmen. Bei den folgenden Schritten im Risikomanagement (Risikoanalyse, -bewertung und -beherrschung) wird nur ein grundlegender Überblick über die Vorgänge gegeben.

3.4.1 Risikoanalyse

Bei der Risikoanalyse der Software wurden mittels Brainstorming bzw. laufend während des Entwicklungsprozesses (Abbildung 3.6) produktspezifische und in diesem Falle speziell durch die Software/Ansteuerung entstehende Risiken identifiziert. Die potenziellen Gefährdungen werden sowohl für für Normalbedingungen, also auch Fehlerbedingungen bzw. den vorhersehbaren Missbrauch berücksichtigt.

3.4.2 Risikobewertung

Zur Einteilung der Risiken ($Risiko = Eintrittswahrscheinlichkeit \cdot Schaden$) [8] wurde die bereits aus den vorlaufenden Arbeiten ([10], [11]) vorhandene Risikomatrix (Tabelle 3.2) verwendet. Dies umfasst auch die Klassifizierung der Eintrittswahrscheinlichkeit und der entstandenen Schadenshöhe.

Tabelle 3.1 Eingetretener Schaden

Bezeichnung	**Beschreibung**
Kein/ geringer Schaden	Keine / geringe Verletzung
Mittel	Verletzung
Schwer	Schwere Verletzung bis Tod
Katastrophal	Schwere Verletzung oder Tod mehrerer Personen

Die Auswirkungen der eingetretenen Schäden sind in Tabelle 3.1 aufgeschlüsselt.
Tabelle 3.2 stellt die Risikomatrix zur Risikobewertung der Software und Ansteuerung dar. Sie wurde schon für die Risikobewertung des Konzepts und Aufbaus des Therapiegerätes verwendet [10], [11].

Tabelle 3.2 Risikomatrix

	Eingetretener Schaden				
Eintrittswahrscheinlichkeit	gering	mittel	schwer	katastrophal	Risikostufe
häufig					1
manchmal					1
gelegentlich					1
selten					2
unwahrscheinlich					3
unglaublich					4

Tabelle 3.3 Beschreibung der Risikostufen [8]

Risikostufe	**Beschreibung**
1	**Nicht akzeptierbar große Risiken**
2	**Hohe Risiken** (ALARA-Prinzip): Akzeptierbar, wenn der Nutzen überwiegt und die Risikominimierung mit einem vernünftigen Aufwand bewerkstelligt werden kann.
3	**Tolerierbare Risiken** (ALARP-Prinzip): Akzeptierbar, wenn das Risiko mit wirtschaftlich vertretbarem Aufwand so weit wie möglich reduziert wurde.
4	**Akzeptierbar niedrige Risiken**

Die einzelnen Risikostufen werden in Tabelle 3.3 näher beschrieben. Hierbei werden die Risikostufen 1 + 2 in Summe als „nicht akzeptabel“ und die Stufen 3 + 4 als „akzeptabel“ bewertet.

3.4.3 Risikobeherrschung

Bei nicht akzeptablen Risiken müssen Maßnahmen zur Minimierung der Risiken durchgeführt werden, bis diese schließlich als akzeptabel eingestuft werden können. Diese Maßnahmen können entsprechend des Prinzips der integrierten Sicherheit (mit sinkender Effektivität) konstruktiver (unmittelbar, Stufe 1), unterstützender (mittelbar, Stufe 2) und hinweisender Art (Stufe 3) sein [8]. Zusätzlich kann der bestimmungsgemäße Gebrauch eingeschränkt werden. Durch Umsetzung dieser Maßnahmen können wiederum neue Gefährdungen und Risiken, auch als „Rückwirkungen“ bezeichnet, entstehen. Somit muss eine weitere Risikoanalyse und -bewertung, sowie gegebenenfalls eine Risikominimierung durchgeführt werden.

3.4.4 Bewertung des Gesamtrisikos

Final müssen sowohl die verbleibenden Einzelrisiken, als auch ein kumulatives Gesamtrisiko akzeptierbar sein. So kann es sein, dass einige gleichzeitig auftretende akzeptierbare Einzelrisiken zu einem nicht mehr akzeptablen Gesamtrisiko führen. Somit sind auch die Wechselwirkungen zwischen den einzelnen Risiken zu berücksichtigen und analysieren.

3.5 Berücksichtigung der Norm: EN 62304 -Medizingeräte-Software

Dieses Kapitel gibt einen kurzen Überblick zur Methodik, die der Berücksichtigung der Norm für Medizingeräte Software diente. Die detaillierten Resultate sind Kapitel 4 bzw. der technischen Dokumentation zu entnehmen.

3.5.1 Anforderungen an die Software

Die Anforderungen an die Software zur Ansteuerung des Therapiegerätes ergaben sich aus den (recht groben) grundlegenden Anforderungen des Klinikums, aus der Konstruktion des Gerätes, sowie den Bewegungssequenzen, die in der vorausgehenden Masterarbeit [10] erstellt wurden. Natürlich soll die Software auch den zugehörigen (sicherheitstechnischen) Normen entsprechen. (EN 60601, EN 62304, Maschinenrichtlinie)

Anforderungen seitens des Klinikums

Die ursprünglichen Anforderungen an die Ansteuerung/Software waren im Grunde recht simpel gehalten:

- Möglichst automatisierter Ablauf des Therapieprogramms
- Einfache Bedienung (ohne technisches Vorwissen)
- Übersichtliche, intuitive, gut leserliche Benutzeroberfläche
- Unterbrechung des Programms jederzeit über das GUI möglich
- Bei Unterbrechung des Programms sollen alle Hubsäulen sofort einfahren
- Maximaler Hub auf 60mm beschränkt, beim Rücken auf 40mm
- Möglichkeit mehrere fertige Therapieprogramme dauerhaft abzuspeichern

Weitere Anforderungen durch Funktionalität des Geräts

- Therapieprogramme, die sich aus Sequenzen individuell zusammenstellen lassen
- Bewegungssequenzen nach Vorgabe aus der Arbeit von Herrn Mauerhofer [10]
- Zyklische, möglichst physiologisch anmutende Bewegungen
- Ordnungsgemäße Ansteuerung eines Schrittmotors (NEMA23)

Sicherheitstechnische Anforderungen an das Gerät

- Die Hubbewegungen dürfen zu keiner Gefährdung des Patienten führen
- Beschränkung des maximalem Hubs auf 100mm und der Hubgeschwindigkeit auf 100mm/s
- Größtmögliche Sicherheit für den Patienten
- Anzeige von auftretenden Fehlern der Steuergeräte / Motoren
- Unterbrechung der Bewegungen jederzeit und unverzüglich durch Software/Hardwarelösungen
- Fehler in der Steuerung/Software dürfen nicht zu Gefährdungssituationen führen
- Keine unkontrollierten Bewegungen des Geräts nach der auf einen Spannungsausfall folgenden Spannungsrückkehr
- Die Bewegungsparameter dürfen nicht unbeabsichtigt geändert werden

3.5.2 Software Entwicklungsprozess

Bei der Softwareentwicklung fiel die Wahl des Entwicklungsmodells auf das V-Modell (Abbildung 3.5), da die Entwicklungsstufen einerseits linear angeordnet sind und jeder Entwicklungsstufe eine Testphase zugeordnet ist. Des Weiteren kommt dieses Modell einer unfangreichen und vollständigen Dokumentation zu Gute.

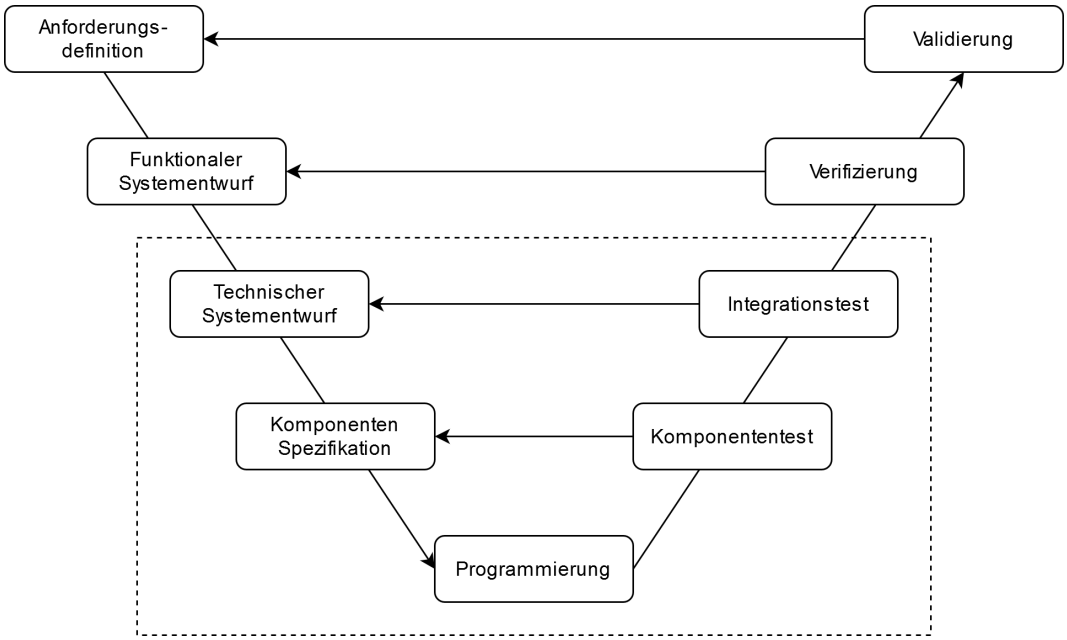


Abbildung 3.5 Visualisierung des verwendeten Softwareentwicklungsprozess-Modells: Dem V-Modell

Im Laufe der Produkt-/Softwareentwicklung hat sich ein gewisses Schema etabliert, nach dem die einzelnen Entwicklungsschritte abgewickelt wurden. Abbildung 3.6 veranschaulicht den grundlegenden Ablauf einer jeden Entwicklungsphase. Diese Entwicklungsphasen fanden im V-Modell in Abbildung 3.5 in den Entwicklungsstufen statt, die durch ein Rechteck markiert wurden.

Zu Beginn jeder Phase stand eine hardwareseitige Erweiterung oder Veränderung des Systems. Diese Änderung konnte durch das Hinzufügen, Entfernen oder Modifizieren bestehender Komponenten, Verbindungen oder Schnittstellen geschehen, die die grundlegende Funktion beeinflussen konnten. Entsprechend dieser hardwareseitigen Erweiterungen wurde der Programmcode des Arduino angepasst, die Benutzeroberfläche adaptiert und gegebenenfalls die Fahrprofile am Steuergerät abgeändert. Im Anschluss an diese Anpassungen wurde die modifizierte Softwareeinheit oder -komponente auf Funktionalität getestet (Komponententest). Zusätzlich wurde überprüft, ob die bestehenden Funktionen und das gesamte System noch fehlerfrei arbeiteten (Integrationstest). Sollten diese Zwischentests nicht zufriedenstellend ausgefallen sein, wurde eine dem Fehler entsprechende software-

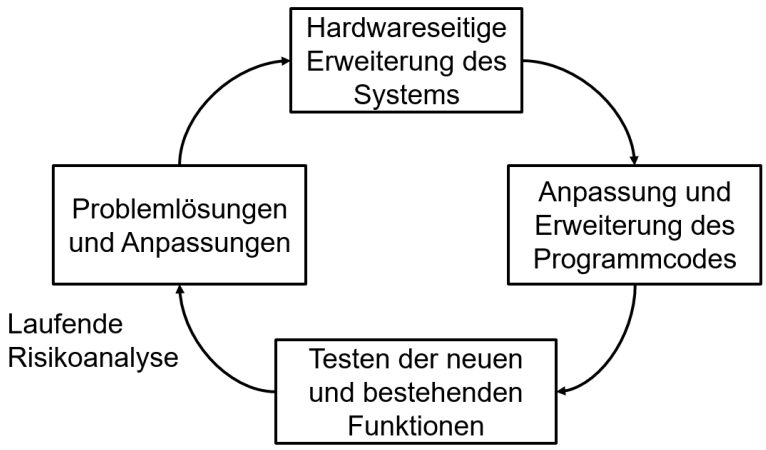


Abbildung 3.6 Ablauf der einzelnen Entwicklungsphasen

und hardwareseitige Problemlösung durchgeführt um diese Fehler zu beheben und deren Ursachen zu identifizieren. Erst mit einem erfolgreichen Abschluss einer Phase wurde dann dieser Kreislauf erneut gestartet und die nächste Entwicklungsphase mit einer Erweiterung des Systems eingeleitet. Parallel zu diesem Prozess wurde die Risikoanalyse laufend aktualisiert. Neue Funktionen und Komponenten können logischerweise sowohl neue Gefahrenquellen, aber auch Möglichkeiten zur Reduktion von bestehenden Risiken darstellen.

Im folgenden Abschnitt werden die durchlaufenen Entwicklungsphasen und deren Schwerpunkte näher beschrieben:

Entwicklungsphase 1: 1 Schrittmotor + Spindel

In der ersten Entwicklungsphase wurden sehr grundlegende Funktionen erarbeitet, getestet und mit ihnen experimentiert, bis ein zufriedenstellendes Ergebnis vorlag. So wurde unter anderem erstmalig ein „dryve D1“ Steuergerät mit zwei Schaltnetzteilen (5V + 24V) versorgt, über das Webinterface programmiert und damit ein Schrittmotor testweise angesteuert. Außerdem wurde mit Fahrprofilen experimentiert, indem die einzelnen Bewegungsparameter variiert wurden. Die Möglichkeit des Hinzufügens einer S-Kurve zu den Bewegungen wurde ausgetestet, um so die Bewegungen fließender zu gestalten und ruckartige Beschleunigungen zu vermindern. Eine erste Version der Benutzeroberfläche MegunoLink wurde erstellt, um manuell einzelne Fahrprofile via Arduino anzusteuern. Parallel dazu wurde ein erster Programmcode am Arduino implementiert, um zum Beispiel eine Dezimalzahl in einen Binärcode umzuwandeln, der dann an den Ausgangs Pins in Form von HIGH und LOW gesetzten Ausgängen dargestellt wurde (`decimalToBinary`). Des Weiteren wurde der Arduino durch einfache Jumper Kabel mit dem Steuergerät verbunden, um so eine erste Ansteuerung des Schrittmotors zu realisieren. Ebenso wurden, wie in Ka-

pitel 3.3.3 und Tabelle 4.1 dargestellt bereits erste Messungen der Stromaufnahme des Motors + Steuergerät an Primär- und Sekundärseite durchgeführt. Die Spannungsversorgung geschah hier über einen provisorischen Aufbau aus einem Netzstecker und Blockklemmen, später Reihenklemmen, zur Stromverteilung. Außerdem wurde eine Steckplatine (bis zur letzten Entwicklungsphase) verwendet, um die Signalleitungen zwischen Arduino und Steuergerät zu führen und zu verbinden. Zusammenfassend wurde diese Entwicklungsphase mit dem den Eingabebefehlen entsprechenden Auf- und Abfahren der Flanschmutter entlang der Spindel beendet.

Entwicklungsphase 2: 2 Hubsäulen

Die Schwerpunkte dieser Entwicklungsphase bestanden unter Anderem im koordinierten Ablauf der Bewegungen zweier Hubsäulen, sowie weiteren Tests der vorgesehenen Funktionen. Nachdem zu diesem Zeitpunkt bereits der Prototyp einer Hubsäule fertiggestellt war und diese die volle Funktionalität aufwies, konnte ebenso die automatische Referenzierung durch den in die Säule integrierten Endlagenschalter getestet werden. Softwareseitig wurde die erste Version des Arrays implementiert, in dem später die vom Benutzer ausgewählten Bewegungssequenzen gespeichert werden sollen (`sequences`). Mit diesem Array wurden die beiden Hubsäulen erstmalig koordiniert bewegt. Ebenso wurde die Stromaufnahme von zwei Hubsäulen sowie deren Steuergeräten an einem 24V + 5V Schaltnetzteil gemessen (siehe Kapitel 3.3.3 und Tabelle 4.1). Die Spannungsversorgung erfolgte im gleichen Provisorium, wie in Phase 1. Parallel dazu wurde die sogenannte Technikebene designt und mit deren Aufbau begonnen [11]. Die Technikebene enthält den Arduino Mikrocontroller, alle Schaltnetzteile und Steuergeräte, sowie Kabelkanäle, Signal- und Versorgungsleitungen und weitere hardwareseitige Maßnahmen zur Ansteuerung des medizinisch-elektrischen Systems. Sie befindet sich im Therapiegerät direkt unterhalb der acht Hubsäulen und deren Halterungen.

Entwicklungsphase 3: 4 Hubsäulen

Die teilweise Fertigstellung der Technikebene, sowie der vier von acht Hubsäulen leitete in die nächste Entwicklungsphase über. In Phase 3 wurde hauptsächlich die Spannungsversorgung von allen Netzteilen durch das Reihenklemmensystem in der Technikebene überprüft. Die Spannungsversorgung erfolgte nun also erstmalig nicht durch das Provisorium, wie zuvor in den Phasen 1 und 2. Ebenso wurde das Netzwerk an Signalleitungen zur Ansteuerung der Steuergeräte erweitert und mit Hilfe von mehradrigen Signalkabeln in eine nun geordnetere Form gebracht. Weitere Strommessungen, wie in Kapitel 3.3.3 und Tabelle 4.2 beschrieben wurden durchgeführt, um die letztendlich notwendige Anzahl an Schaltnetzteilen zu bestimmen. Für die Strommessungen wurde eine temporäre Bewegungssequenz am Arduino programmiert, bei der sich alle vier Hubsäulen synchron mit

hoher Auslenkung, Beschleunigung und Geschwindigkeit dauerhaft auf- und ab bewegen.

Entwicklungsphase 4: 8 Hubsäulen

Mit der vollständigen Fertigstellung der Technikebene und aller acht Hubsäulen, sowie deren finaler Montage in das Therapiegerät begann die vorletzte Entwicklungsphase. Alle acht Steuergeräte + Schrittmotoren wurden, wie vorgesehen über drei Schaltnetzteile ($1 \cdot 5V + 2 \cdot 24V$) und ein System aus Reihenklemmen mit Strom versorgt. Für die Ansteuerung der Steuergeräte wurde ebenfalls das Signalleitungssystem erweitert. Der Arduino Code wurde angepasst, um alle acht Hubsäulen anzusteuern. Ebenso wurde die Funktionalität des Codes und der Benutzeroberfläche stark erweitert (z.B.: Auswahl der Sequenzen, Endlosschleife, Anzeige auftretender Fehler, Redundanzen). Erstmals wurde auch die Gelmatte in Kombination mit den Hubsäulen an einer Therapiepuppe getestet. Der eigentliche Schwerpunkt in dieser Phase bestand unter anderem in der Programmierung der finalen Bewegungssequenzen laut Vorgabe [10]. Bei Tests der ersten Sequenz kam der Fehler auf, dass das Start Signal nicht korrekt vom Arduino an das Steuergerät übermittelt und somit die Bewegung von zwei Säulen (7 und 8) nicht gestartet wurde. Nach einigen weiteren Versuchen und Messungen der Output Pins via Multimeter, stellte sich heraus, dass die Spannung dieser Pins nicht zwischen 0-5V, sondern 2,5-5V liegt. Somit wurde die HIGH/LOW-Grenze von 3V permanent überschritten. Durch einen Wechsel der Pins wurde dieses Problem behoben.

Entwicklungsphase 5: Finale Hardwarekonfiguration

Vor der letzten Entwicklungsphase fand keine für die Ansteuerung relevante Hardwareänderung statt. Es wurde lediglich Kabelmanagement zwischen den Steuergeräten und Schrittmotoren betrieben, sowie Schutzleiter zu relevanten Bauteilen verlegt. Des Weiteren wurde der Not-Aus Taster provisorisch am Aluminium Rahmen fixiert, da das Gehäuse noch nicht montiert war. Nach einer Vorführung des Therapiegerätes für das Klinikum wurden entsprechend des Feedbacks einige Anpassungen vorgenommen. Diese umfassten Beschränkungen des Hubs der Hubsäulen auf maximal 60mm bzw. 40mm für die Rückenpartie, das sofortige Einfahren der Hubsäulen bei einer Therapieunterbrechung, sowie die Möglichkeit Therapieprogramme (unter pseudonymisierten Patienten 01-10) dauerhaft zu speichern. Letzteres wurde durch Speichern auf dem im Arduino integrierten EEPROM realisiert. Des Weiteren wurden entsprechend des Feedbacks kleinere Änderungen an der Benutzeroberfläche durchgeführt, wie z.B. Anpassung der Schriftgröße. Ansonsten wurden in dieser Phase diverse Optimierungen am Programmcode, sowie einige Funktions- und Sicherheits- sowie die finalen Verifizierungs- und Validierungstests durchgeführt. Unter Anderem wurden elektrische Sicherheitsparameter und die Temperatur der elektrischen Komponenten gemessen. Ebenso wurden Worst-Case-Tests durchgeführt, um zu erörtern

wie die Therapiepuppe auf gefährlich schnelle, ruckartige Bewegungen mit hoher Auslenkung reagiert.

3.5.3 Verwendete Werkzeuge

Im folgenden Abschnitt werden die drei wichtigsten Werkzeuge beschrieben, die bei der Entwicklung der Ansteuerungssoftware zum Einsatz kamen.

Arduino IDE

Die Arduino Entwicklungsumgebung in der Version 1.8.9 wurde während des gesamten Entwicklungsprozesses für die Programmierung des Programms am Mikrocontroller genutzt. Der Code wurde dann durch den integrierten Compiler (AVRISP mkII) kompiliert und über via USB Datenkabel an den Mikrocontroller übertragen.

MegunoLink

MegunoLink stellt sowohl die verwendete Benutzeroberfläche, als auch das Werkzeug zur Programmierung ebendieser dar (Version 1.31.19269.926). Des Weiteren kommuniziert dieses Programm während des Betriebs über die USB Schnittstelle mit dem Mikrocontroller. Folglich wird dieses Programm auch als Bestandteil der medizinischen Software gesehen.

igus Dryve D1 Webinterface

Wie auch schon MegunoLink wird das Webinterface des igus Dryve D1 (mit der Firmware Version „dryve-D1-1-20190226“) zum Programmieren der Fahrprofile/Bewegungen der Hubsäulen, sowie zum Setzen der Bewegungslimits verwendet. Da kein Zugriff auf den Quellcode der Steuergeräte besteht und sie einen essentiellen Bestandteil zur Ansteuerung der Hubsäulen darstellen, wird die integrierte Software der Steuergeräte als SOUP klassifiziert.

3.5.4 Verifizierungs- und Validierungstests

Während es bei diesem Projekt an die Integrations- und Systemtests keine besonderen Anforderungen hinsichtlich der Dokumentation und Durchführung gibt, müssen die finalen Verifizierungs- und Validierungstests entsprechend engmaschig dokumentiert und streng nach Protokoll durchgeführt werden. Zu diesem Zweck wurde zunächst aus den Anforderungen an die Software bzw. das Gerät, sowie dessen spezifische Funktionalität eine Liste mit verschiedenen Tests erstellt (Anhang E). Diese Tests wurden dann in funktionelle Gruppen gegliedert (z.B.: Inbetriebnahme, Patientenmanagement, Bewegungssequenzen,

etc.) und einzeln, Schritt für Schritt, entsprechend des Ablaufs, der Funktion und der erwarteten Ergebnisse ausgearbeitet.

Somit wurden Verifizierungstests und Validierungstests in gewisser Weise zusammengefasst, da zumindest eine teilweise Erfüllung der Anforderungen aus dem positiven Ergebnis der Funktionstests gegeben ist. Die hier durchgeführte Validierung wird also im Sinne der Überprüfung der User-Anforderungen an die Software gesehen. Es handelt sich hierbei um keine komplette Validierung des Medizinproduktes, sondern um einen ersten Schritt. Die weiteren Schritte einer Validierung müssen auch die usability und die klinische Bewertung, welche im Rahmen dieser Arbeit nicht berücksichtigt werden, beinhalten.

Bei der Durchführung der Tests wurden die vorbereiteten Testprotokolle akribisch abgearbeitet. Beispiele hierfür befinden sich ebenfalls in Anhang E. Die Ergebnisse der einzelnen Funktionstests (FT) wurden im Protokoll festgehalten (Erwartetes Ergebnis: „JA“ / „NEIN“) und ein Gesamtergebnis, welches am Testprotokoll und der zugehörigen Übersichtsliste ersichtlich ist, notiert („Bestanden“ / „Nicht bestanden“). Zur besseren Rückverfolgung der Vorgänge wurden die Protokolle mit den Namen der Testautoren, Freigebenden und Testern, sowie den zugehörigen Daten versehen.

4 Ergebnisse

4.1 Dimensionierung der Spannungsversorgung

Tabelle 4.1 Stromabgabe eines 24 V Schaltnetzteils an der Sekundärseite bei unterschiedlichen Bedingungen

Messung Nr.	Bedingung	1 Motor [A]	2 Motoren [A]
1	Motoren nicht bestromt (Freigabe inaktiv)	0,08	0,133
2	Motoren bestromt (Freigabe aktiv)	0,190	0,38
3	Referenzierung (niedriger Vorschub: 10mm/s)	0,525	1,05
4	Hoher Vorschub (100mm/s)	0,85	1,75
5	Hoher Vorschub + Gegendruck auf Flanschmutter	1,2	2,088

Die in Tabelle 4.1 angeführten Messwerte stellen jeweils Spitzenwerte dar. Der Gegendruck auf die Flanschmutter wurde händisch ausgeübt um eine Belastung der Hubsäule zu simulieren.

Tabelle 4.2 Stromaufnahme von vier Motoren + Steuergeräten an einem 24 V Schaltnetzteil bei unterschiedlichen Bedingungen

Messung Nr.	Bedingung	Gemessener Strom [A]
1	Motoren nicht bestromt (Freigabe inaktiv)	0,235
2	Motoren bestromt (Freigabe aktiv)	0,770
3	Referenzierung (niedriger Vorschub: 10mm/s)	2,52
4	Hoher Vorschub (100mm/s)	3,9
5	Hoher Vorschub + Gegendruck auf Hubsäule	4,48

Die in Tabelle 4.2 angeführten Messwerte stellen jeweils Spitzenwerte dar. Der Gegendruck auf die Hubsäulen von geschätzt jeweils mehreren Kilo wurde händisch ausgeübt, um eine nicht zu erwartende, starke Überbelastung der Hubsäule zu simulieren.

Änderungen während der Entwicklung

Anhand der Messergebnisse aus Tabelle 4.1 (2,088A $<<$ 8,4A) wurde die Anzahl an verwendeten Netzteilen zunächst von acht 24V + ein 5V Netzteilen auf vier 24V + ein 5V Netzteile reduziert. Nach der Durchführung weiterer Messungen und Berücksichtigung von deren Ergebnissen (Tabelle 4.2, 4,48A $<$ 8,4A) wurden die Netzteile final auf zwei 24V Netzteile und ein 5V Netzteil reduziert, also insgesamt drei, statt wie ursprünglich geplant neun Schaltnetzteile.

4.2 Software-Architektur

Das Therapiegerät wird von einem Softwaresystem gesteuert, welches sich in folgende übergeordnete Komponenten einteilen lässt:

- Die grafische Benutzeroberfläche (GUI) MegunoLink
- Der Programmcode, welcher auf dem Arduino Mikrocontroller läuft
- Die Software der Steuergeräte (klassifiziert als SOUP)

4.2.1 Struktur

Überblick über das übergeordnete Programm am Mikrocontroller

Abbildung 4.1 stellt den vereinfachten Programmaufbau des übergeordneten Codes dar, der auf dem Arduino Mikrocontroller die Steuergeräte entsprechend der Eingaben vom GUI ansteuert. Aus Gründen der Übersicht wurden nur die für den Ablauf wichtigsten Variablen dargestellt und die einzelnen Rückmeldungen der Funktionen an Megunolink mit einem * gekennzeichnet.

Beschreibung der Blöcke in den Flussdiagrammen (Abbildung 4.1, 4.3 und 4.4):

Weiß:	Funktionen
*:	Funktionen mit Feedback an Megunolink
Blau:	Speicher
Grün:	Schnittstellen
Orange:	Variablen
Rot:	Sonstige

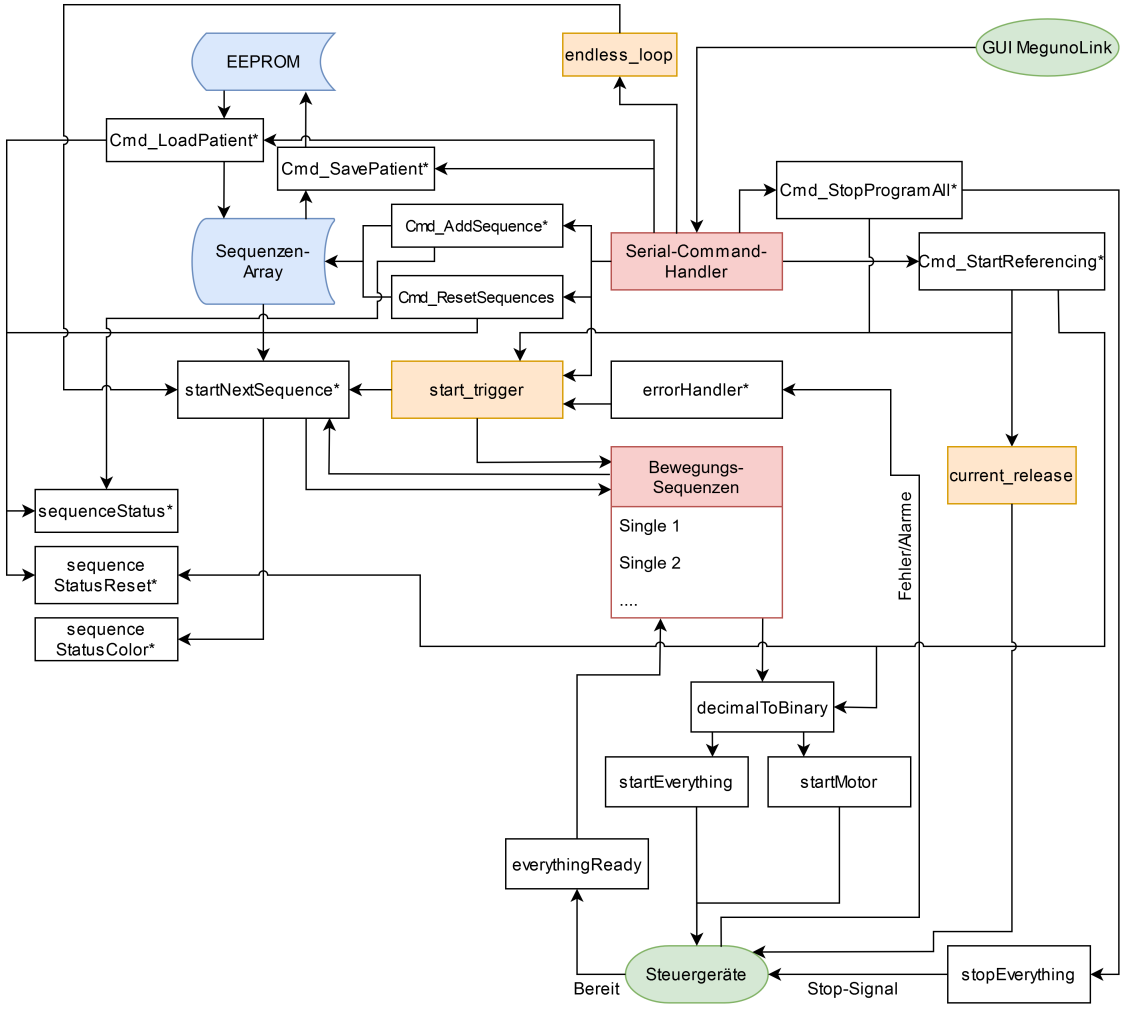


Abbildung 4.1 Aufbau des Arduino Programmcodes und Zusammenhänge der wichtigsten Funktionen

GUI: MegunoLink

Abbildung 4.2 zeigt die grafische Benutzeroberfläche „MegunoLink“ nach Deaktivierung der Bestromung mitten im Therapieprogramm, mit allen ausgelösten Warnungen und Fehlern. Die in rot eingefügten Zahlen dienen der nachfolgenden Beschreibung der Schaltflächen, Textfelder und Ausgaben.

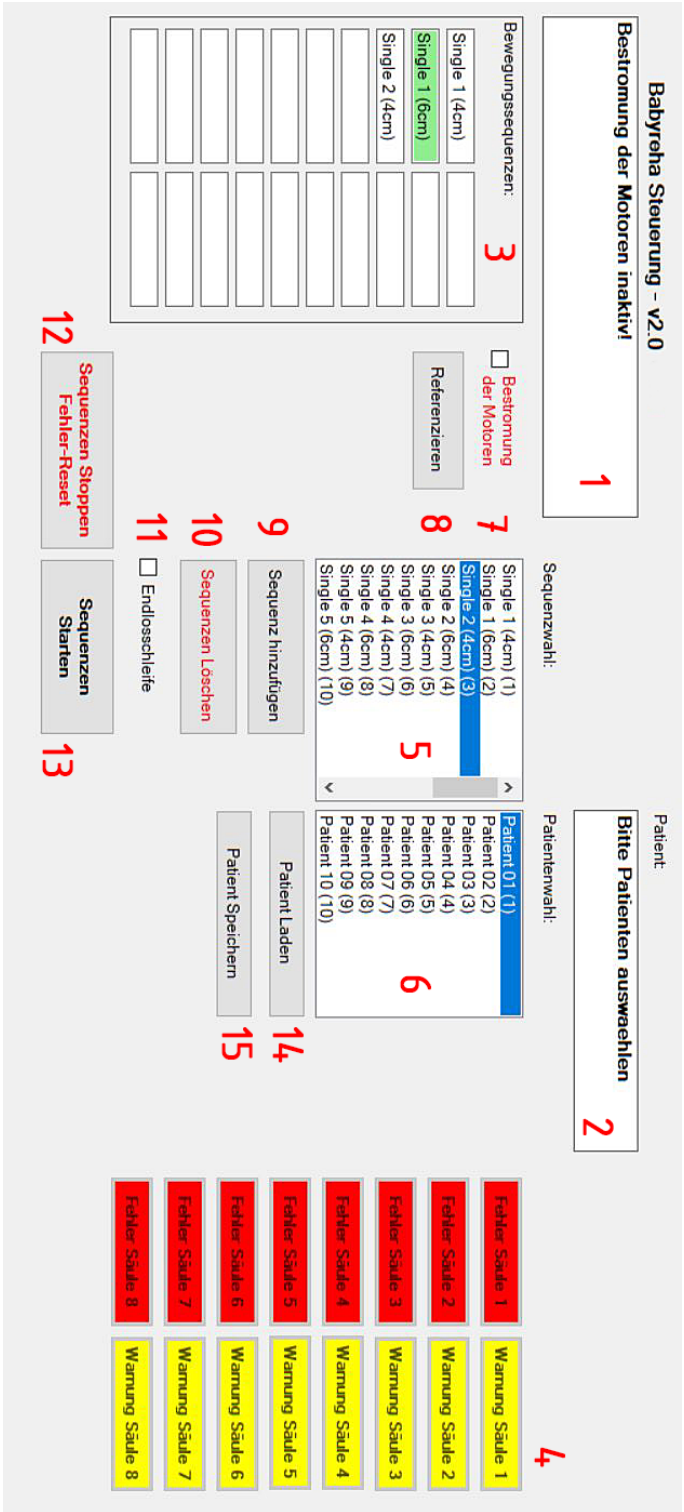


Abbildung 4.2 Die grafische Benutzeroberfläche „MegunoLink“.

Beschreibung der einzelnen Komponenten der grafischen Benutzeroberfläche (Abbildung 4.2):

1. Allgemeines (linkes) Textfeld:
   Dient der Kommunikation mit dem Anwender/Benutzer. Grundlegende Anweisungen und Rückmeldungen werden über dieses Textfeld ausgegeben.

2. Textfeld „Patient“:
   Zeigt bei einem Lade-/ Speichervorgang an, welcher Patient aus dem/in den EEPROM geladen bzw. gespeichert wurde.

3. Textfelder „Bewegungssequenzen“:
   Sie zeigen die zum Therapieprogramm hinzugefügten Bewegungssequenzen der Reihenfolge nach an. Während des laufenden Programms wird die gerade aktive Bewegungssequenz grün hinterlegt.

4. Fehler-/Warnungen-Anzeigen:
   Zeigen Warnungen bzw. Fehler, die an der jeweiligen Hubsäule bzw. Steuergerät aufgetreten sind an.

5. Liste „Sequenzenwahl“:
   Diese Liste dient der Auswahl der vorgegebenen Bewegungssequenzen.
   $(10 \cdot \text{Single} + 12 \cdot \text{Dual} + \text{Test} + \text{Wartung})$

6. Liste „Patientenwahl“:
   Hier wird ausgewählt, welcher pseudonymisierte „Patient“ aus dem EEPROM geladen, oder in den EEPROM gespeichert wird.

7. Checkbox „Bestromung der Motoren“: Durch diese Checkbox kann die Bestromung der Motoren (für Bewegungen bzw. Haltestrom) manuell aktiviert oder deaktiviert werden.

8. Button „Referenzieren“:
   Muss nach jeder Deaktivierung oder jedem Ausfall der Bestromung durchgeführt werden.
   Die Checkbox „Bestromung der Motoren“ wird durch Betätigen dieses Buttons automatisch aktiviert.

9. Button „Sequenz hinzufügen“:
   Fügt die (blau) markierte Bewegungssequenz aus der Liste „Sequenzenwahl“ dem Therapieprogramm hinzu, welche danach im nächsten freien Textfeld von „Bewegungssequenzen“ aufgelistet wird.

10. Button „Sequenzen Löschen“:
    Löscht alle dem Therapieprogramm hinzugefügten Bewegungssequenzen aus dem temporären Speicher/Array.

11. Checkbox „Endlosschleife“:
    Aktiviert eine dauerhafte Wiederholung des gesamten Therapieprogramms.

12. Button „Sequenzen Stoppen / Fehler-Reset“:
    Sofortige Unterbrechung des laufenden Therapieprogramms. Alle Hubsäulen stoppen und fahren danach komplett ein.
    Dient auch der Quittierung von aufgetretenen Fehlern und der anschließenden Reaktivierung der jeweiligen Hubsäule.

13. Button „Sequenzen Starten“:
    Startet das Therapieprogramm bzw. setzt es nach einer Unterbrechung fort.

14. Button „Patient Laden“:
    Lädt entsprechend der in der Liste „Patientenwahl“ getroffenen Auswahl einen zuvor gespeicherten Patienten aus dem EEPROM in das aktuelle Therapieprogramm, die entsprechenden Bewegungssequenzen werden direkt unter „Bewegungssequenzen“ angezeigt.

15. Button „Patient Speichern“:
    Speichert entsprechend der Auswahl der Liste „Patientenwahl“ das aktuelle Therapieprogramm im EEPROM des Mikrocontrollers ab.

Fahrprofile der Steuergeräte

Der Mikrocontroller steuert dem Programmcode entsprechend die in Kapitel 3.3.2 aufgelisteten digitalen Eingänge der Steuergeräte an. Dies geschieht in Abhängigkeit von den Eingaben durch das GUI und den Rückmeldungen der Steuergeräte. Neben den sicherheitsrelevanten Bewegungslimits und allgemeinen Konfigurationsparametern sind die abgespeicherten, den Bewegungssequenzen entsprechenden Fahrprofile der in diesem Fall wichtigste Bestandteil der Software der Steuergeräte. Jedem Fahrprofil ist ein Dezimalwert von 1-32 zugeordnet, der über den in Kapitel 3.3.2 erwähnten 5-Bit Bus ausgewählt wird.

Tabelle 4.3 Übersicht der Fahrprofile

Nr.	Modus	Ziel	Beschleunigung	Geschwindigkeit	Verzögerung
-	-	(mm)	(mm/s^2)	(mm/s)	(mm/s^2)
1	ABS	0	30	30	100
2	ABS	0	40	40	40
3	ABS	0	60	60	60
4	ABS	0	100	100	100
12	ABS	40	40	40	40
13	ABS	30	60	60	60
14	ABS	40	100	100	100
15	ABS	50	40	40	40
15 *	-	-	-	-	-
16	ABS	60	40	40	40
16 *	ABS	40	40	40	40
32	HOM	LSN	10	20	150

Tabelle 4.3 Zeigt die Bewegungsparameter aller Fahrprofile der Hubsäulen, wie sie in die Steuergeräte eingespeichert sind. Hervorzuheben sind die Bewegungen 15 und 16, die einen Hub von über 4cm zur Folge haben. Aus Sicherheitsgründen beträgt der maximale Hub der Säulen (3+4) der Rückenpartie 4cm. Somit ist Fahrprofil Nr. 15 (*) bei diesen Steuergeräten (3+4) nicht angelegt. Fahrprofil 16 ist bei den Steuergeräten 3 und 4 so abgeändert, dass der maximale Hub 4cm beträgt (16*). Alle Hubsäulen können also bei den Bewegungssequenzen, die eine Auslenkung von 6cm vorsehen vom Mikrocontroller gleich angesteuert werden. Der Modus „ABS“ bezeichnet das Anfahren einer absoluten Endposition (Ziel) und „HOM“ eine Referenzfahrt mit einem Endlagenschalter (LSN) als Ziel. Die Fahrprofile 3, 12, 13, 15 werden in der aktuellen Version (v1) nicht verwendet und sind lediglich als Vorbereitung auf eine etwaige Erweiterung der Software angelegt.

4.3 Softwarekomponenten im Detail

Eine Übersicht der wichtigsten im Programmcode verwendeten Variablen und aller Funktionen befindet sich in den Anhängen G und H.

4.3.1 Übernahme der Eingaben von Megunolink

Zur besseren Veranschaulichung der Interaktionen zwischen Benutzeroberfläche „MegunoLink“ und dem Arduino Code stellen die Abbildungen 4.3 und 4.4 die Abläufe im Arduino Code durch Eingaben in das GUI dar. Als Schnittstelle dient die zu MegunoLink zugehörige Funktion „Serial-Command-Handler“, welche die Eingaben vom GUI an den Code am Mikrocontroller weiterleitet und verarbeitet. Für eine niedrige Latenz wird der Serial-Command-Handler in der Main Loop des Arduino Codes periodisch ausgeführt. In den Ablaufdiagrammen (Abbildung 4.3 und 4.4) wird der Serial-Command-Handler der Übersicht halber durch rote Pfeile dargestellt. Ebenso werden alle Funktionen, die ihren Input durch den Serial-Command-Handler erhalten durch den Prefix `Cmd_` gekennzeichnet.

Die beiden Ablaufdiagramme (Abbildung 4.3 und 4.4) zeigen nicht den gesamten Programmcode, sondern nur die für die Eingabe relevanten Funktionen, Speicher (blau) und Variablen (orange).

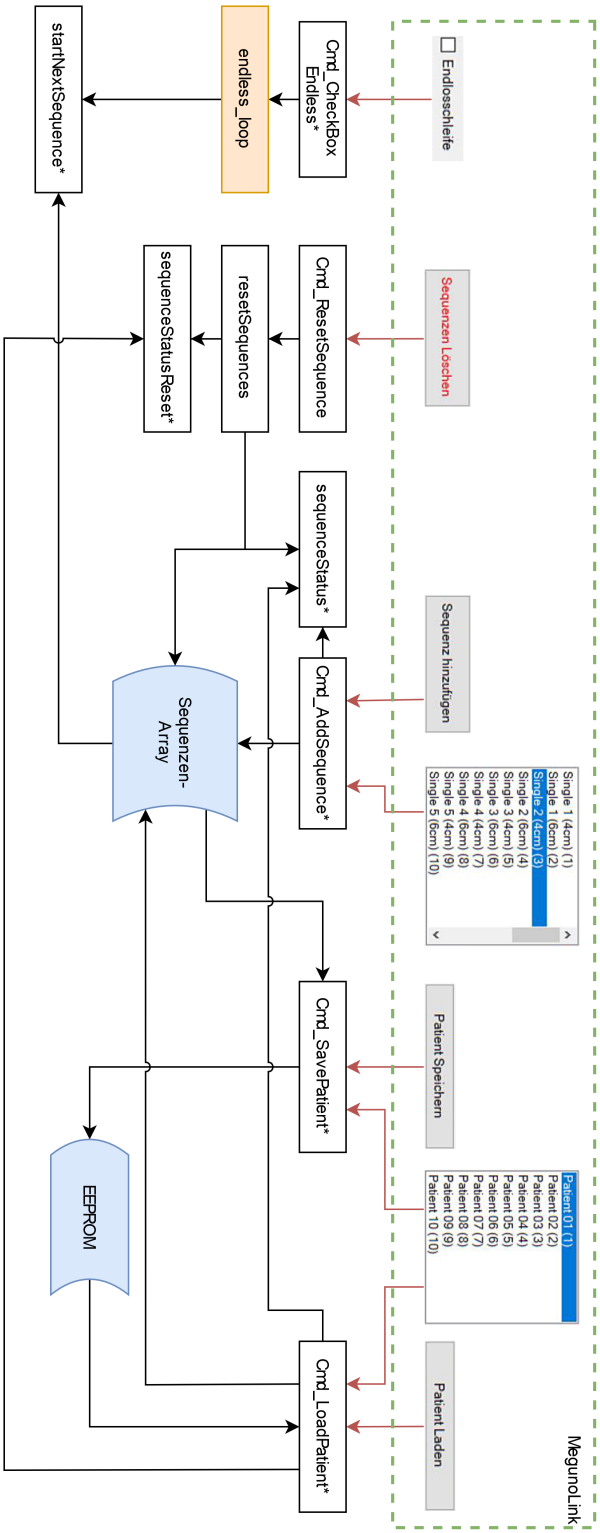


Abbildung 4.3 Zusammenhang zwischen der Benutzeroberfläche und den Funktionen des Arduino Codes im Bezug auf die Verwaltung der Bewegungssequenzen.

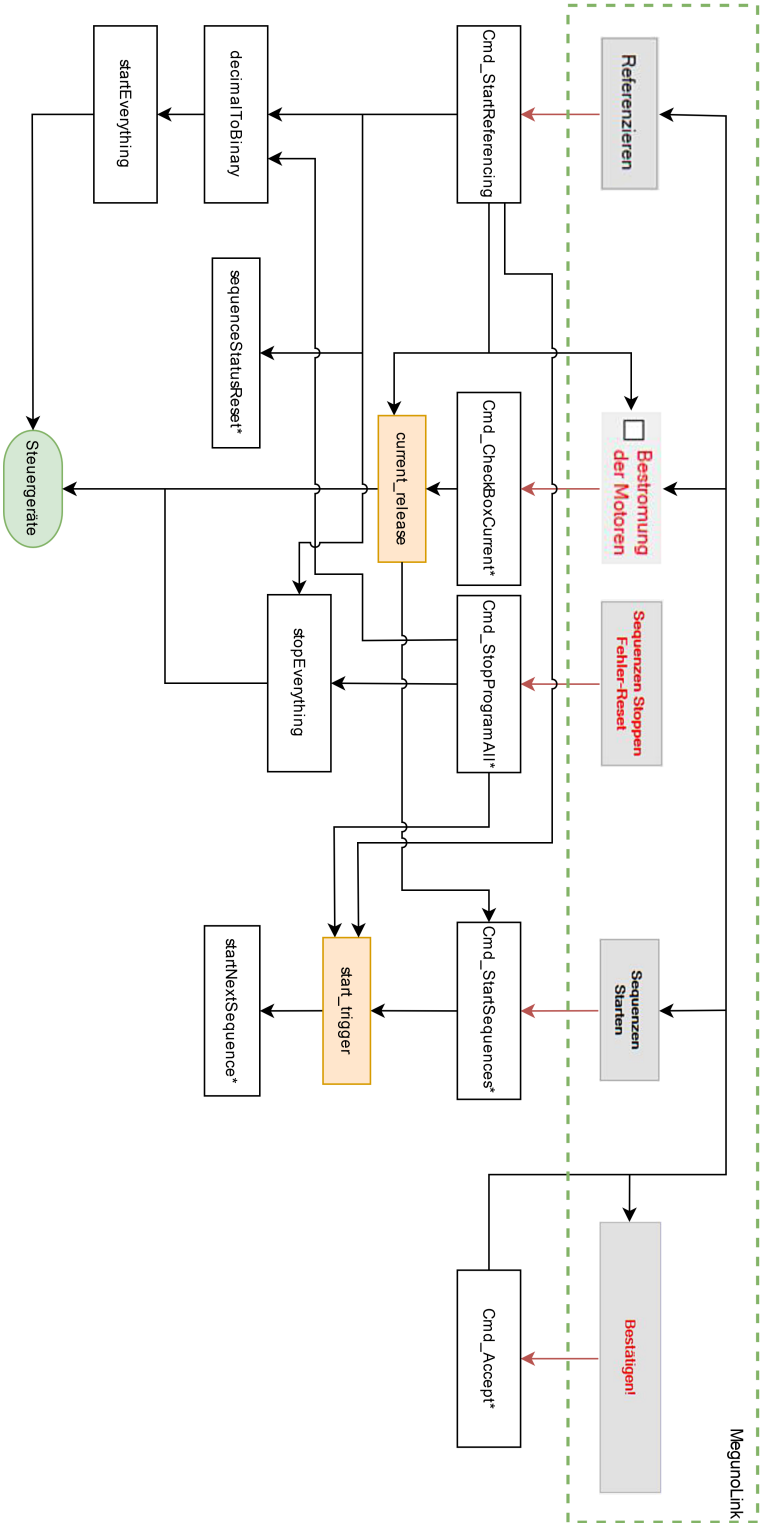


Abbildung 4.4 Zusammenhang zwischen der Benutzeroberfläche und den Funktionen des Arduino Codes im Bezug auf die Ansteuerung des Softwaresystems.

4.3.2 Ausgabe an die Steuergeräte

Die folgenden Funktionen setzen direkt die in Kapitel 3.3.2 genannten, digitalen Eingänge der Steuergeräte über die zugehörigen Ausgänge (Pins) des Arduino Mikrocontrollers. (Die Pin Belegung des Arduino Mega ist in Anhang B ersichtlich)

Auswahl der Fahrprofile: „decimalToBinary“

Diese Funktion bildet den Kern der Ansteuerung der Hubsäulen, da mit ihr die in Tabelle 4.3 aufgelisteten Fahrprofile angesteuert werden. Hierzu werden zunächst alle fünf Output Pins am Mikrocontroller auf „LOW“ gesetzt. Anschließend wird eine in einer Variable (z.B.: `D0_40, D60_40, up_01/02, down_01/02`, etc.) gespeicherte dezimale Zahl in einen 5-Bit Binärcode konvertiert, welcher dann in einem fünfstelligen Array gespeichert wird. Im letzten Schritt werden die fünf Output Pins entsprechend des Arrays bei dem Wert „1“ auf „HIGH“ gesetzt.

Bewegung einzelner Hubsäulen: „startMotor“

Als Eingabe dient dieser Funktion ein Integer Wert von 1-8 (entsprechend der Hubsäule, die angesteuert werden soll). Über einen Switch-Case wird dann laut diesem Eingabewert der zugehörige Pin auf „HIGH“ gesetzt, der mit dem „Start“ Eingang (6) des entsprechenden Steuergeräts verbunden ist. Nach einer vorgegebenen Verzögerung von 30ms wird der Pin wieder auf „LOW“ gesetzt. Die Verzögerung ist notwendig, da ein Signal bzw. eine Änderung des Eingangs für mindestens 10ms am Steuergerät anliegen muss, um korrekt erfasst werden zu können.

Bewegung aller Hubsäulen: „startEverything“

Mit dieser Funktion werden alle Hubsäulen synchron in Bewegung gesetzt. Hierfür werden die Pins, die mit den Digitaleingängen „Start“(6) aller Steuergeräte verbunden sind zeitgleich für eine Dauer 30ms auf „HIGH“ gesetzt. Diese Funktion wird bei der Referenzierung, dem automatischen Einfahren aller Hubsäulen und den Bewegungssequenzen „Test“ und „Wartung“ ausgeführt.

Anhalten aller Hubsäulen: „stopEverything“

Mit dieser Funktion werden alle Hubsäulen zeitgleich gestoppt. Hierfür werden die Pins, die mit den Digitaleingängen „Stopp/Reset“(10) aller Steuergeräte verbunden sind gleichzeitig für eine Dauer 30ms auf „HIGH“ gesetzt.

Bestromung der Hubsäulen: „Cmd_CheckBoxCurrent“

Wie aus dem Ablaufdiagramm 4.4 ersichtlich erhält diese Funktion ihre Eingabe direkt durch die Checkbox „Bestromung der Motoren“ im GUI über den „Serial-Command-Handler“. Entsprechend dem Status der Checkbox wird der Pin am Mikrocontroller, der mit den digitalen Eingängen „Freigabe“ (7) aller acht Steuergeräte verbunden ist auf „HIGH“ oder „LOW“ gesetzt. Mit der Freigabe wird die Bestromung aller Schrittmotoren der Hubsäulen aktiviert. Beim Referenzierungsvorgang wird die Checkbox automatisch aktiviert und die Motoren somit bestromt.

4.3.3 Sequenzen-Management

Speicherort des aktuellen Therapieprogramms: Das Array „sequences“

Das Therapieprogramm wird, sowie alle weiteren Bewegungs- und Testsequenzen als Zahlenwert in dem Integer-Array `sequences` abgespeichert. Dieses Array wird wiederum temporär am Arbeitsspeicher (SRAM) des Mikrocontrollers gespeichert. Es hat die Dimensionen [2]x[20], wobei aber nur die erste Reihe [0] genutzt wird. Die zweite Reihe war ursprünglich für die Anzahl an Wiederholungen der jeweiligen Sequenz vorgesehen, was im Laufe der Entwicklung aber verworfen wurde. Trotzdem wurde das [2]x[20] Layout beibehalten, als Vorbereitung für etwaige zukünftige Erweiterungen des Programmcodes. Jeder Bewegungs- und Testsequenz ist eine Integer Zahl von 1-24 zugeordnet, die in den maximal 20 Positionen des Arrays gespeichert werden können. `sequences` wird bei Programmstart als Nullarray initialisiert.

Hinzufügen von Sequenzen zum Therapieprogramm: „Cmd_AddSequence“

Durch Betätigen des Buttons „Sequenz hinzufügen“ im GUI wird die in der Liste „Sequenzenwahl“ blau hinterlegte Bewegungssequenz bzw. der ihr zugeordnete Integer Wert dem Array `sequences` hinzugefügt. Das Array wird hierbei der Reihenfolge nach von Position 1-20 befüllt. Sind alle 20 Positionen mit Bewegungssequenzen belegt, wird eine entsprechende Meldung an das allgemeine Textfeld im GUI gesendet. Es werden keine weiteren Sequenzen hinzugefügt oder überschrieben. Am Ende jedes Durchlaufs von `Cmd_AddSequence` wird die Funktion `sequenceStatus` aufgerufen um die Änderungen am Therapieprogramm am GUI in den Textfeldern „Bewegungssequenzen“ sichtbar zu machen.

Löschen des Therapieprogramms: „Cmd_ResetSequence“ und „resetSequences“

Wird der Button „Sequenzen Löschen“ im GUI betätigt, führt die Funktion `Cmd_ResetSequence` die Funktion `resetSequences` aus. Diese Staffelung ist notwendig, da `resetSequences` bei jedem Programmstart ausgeführt werden soll und deshalb auch in der Setup Funktion des Arduino Codes aufgerufen wird. `resetSequences` durchläuft das Array `sequences` und setzt alle Positionen auf „0“. Zuletzt werden die Funktionen `sequenceStatus` und `sequenceStatusReset` ausgeführt um die Textfelder „Bewegungssequenzen“ zu aktualisieren.

4.3.4 Visualisierung des Therapieprogramms

Mit den folgenden Funktionen wird die Ausgabe der Textfelder „Bewegungssequenzen“ (im Code als `status_box` bezeichnet) im GUI aktualisiert, damit der Inhalt des Sequenzenarrays und der Stand des Therapieprogramms mit der Anzeige übereinstimmen.

Anzeige der Namen der Sequenzen: „sequenceStatus“

Die 20 Textfelder „Bewegungssequenzen“ sind der Reihe nach durchnummeriert. `sequenceStatus` durchläuft das Sequenzenarray und ordnet die dort abgespeicherten Sequenzen über Switch-Cases den entsprechenden Bezeichnungen zu, welche dann am Textfeld der jeweiligen Position ausgegeben werden.
(z.B.: `case 1: MyPanel.SetText(status_box[status_counter], F(„Single 1 (4cm)“))`)

Anzeige der aktiven Sequenz: „sequenceStatusColor“

Die Variable `sequence_counter` durchläuft im laufenden Therapieprogramm das Sequenzen Array und gibt an, welche Sequenz gerade ausgeführt wird. `sequenceStatusColor` ändert die Hintergrundfarbe des Textfeldes, dessen Nummer dem Wert des `sequence_counter`, also der aktiven Bewegungssequenz entspricht in einen hellgrünen Farbton. Alle weiteren Textfelder, auf die das nicht zutrifft, werden bei jedem Durchlauf wieder weiß hinterlegt.

Rücksetzen der aktiven Sequenz in der Anzeige: „sequenceStatusReset“

Diese Funktion hinterlegt alle Textfelder mit einem weißen Hintergrund. Sie wird bei der Referenzierung, beim Löschen des Arrays und beim Laden eines Patienten/Arrays aus dem EEPROM aufgerufen.

4.3.5 Speichern und Laden von Patienten

Im EEPROM des Mikrocontrollers können Therapieprogramme bzw. die zugehörigen Arrays dauerhaft abgespeichert werden. Pro Position kann maximal 1 Byte (8 Bit) in dem EEPROM abgespeichert werden. Die Gesamtgröße des EEPROM beträgt beim Arduino Mega 4096 Byte. Zum Speichern aller 10 vorgesehenen Patienten Positionen werden also in Summe 200 Byte (= 10 Patienten · 20 - Maximale Sequenzen-Arraylänge) benötigt.

Speichern eines Patienten in den EEPROM: „Cmd_SavePatient“

Die Funktion `Cmd_SavePatient` wird durch Betätigen des Buttons „Patienten Speichern“ ausgeführt. Dabei wird die dem Patienten zugeordnete Nummer (1-10) aus der Liste „Patientenwahl“ übernommen und mit dem Faktor 20 multipliziert, als initiale Speicheradresse verwendet. Ausgehend von dieser initialen Speicheradresse wird nun das aktuelle Array `sequences` der Reihe nach in die folgenden 20 Speicherpositionen des EEPROM kopiert. Nach Abschluss dieses Vorgangs wird eine Meldung, welcher „Patient“ entsprechend der Liste abgespeichert wurde im Textfeld „Patient“ ausgegeben (z.B.: „Patient 01 gespeichert“). Zusätzlich wird im linken Textfeld die Meldung „Patient wurde gespeichert“ angezeigt.

Laden eines Patienten aus dem EEPROM: „Cmd_LoadPatient“

Zum Laden eines im EEPROM abgespeicherten Patienten dient die Funktion `Cmd_LoadPatient` die durch Betätigen des Buttons „Patient laden“ ausgeführt wird. Zu Beginn wird die Funktion `sequenceStatusReset` ausgeführt um die Sequenzen Anzeige zurückzusetzen. Analog zum Speichervorgang wird der EEPROM der dem Patienten zugeordneten Nummer und der somit errechneten initialen Adresse ausgelesen und in das Array `sequences` kopiert. Dabei wird auch die Variable `add_sequence_counter` pro geladener Bewegungssequenz erhöht um nach Beendigung des Ladevorgangs weitere Sequenzen dem Array, an dessen Ende hinzufügen zu können. Nachdem der eigentliche Ladevorgang beendet ist, werden wie nach dem Speichervorgang entsprechende Meldungen in den beiden Textfeldern angezeigt („Patient wurde geladen“ bzw. z.B.: „Patient 01 geladen“). Zusätzlich werden durch Ausführen der Funktion `sequenceStatus` die Textfelder „Bewegungssequenzen“ aktualisiert und die Variable `sequence_counter` auf 0 gesetzt um das Therapieprogramm nach einem eventuellen vorherigen Abbruch von vorne beginnen zu lassen.

4.3.6 Starten und beenden des Therapieprogramms

Die bool'sche Variable `start_trigger` ist eine zusätzliche Sicherheitsmaßnahme, die von mehreren Funktionen aktiviert, oder deaktiviert wird. Eine Bewegungssequenz bzw. deren einzelne Bewegungsphasen werden nur ausgeführt, wenn der `start_trigger` aktiviert ist, also den Wert „1“, oder `true` hat.

Überprüfung der Fahrbereitschaft der Hubsäulen: „everythingReady“

Diese bool'sche Funktion dient der Überprüfung aller digitalen Ausgänge der acht Steuergeräte mit der Bezeichnung „Bereit“ (Digitaler Ausgang 1). Der Ausgang wird vom Steuergerät auf „HIGH“ gesetzt, sobald eine Bewegung fertig ausgeführt wurde bzw. neue Fahrbefehle angenommen werden können. Haben alle digitalen Ausgänge den Wert „HIGH“, so gibt die Funktion den Rückgabewert `true` aus, andernfalls `false`.

Referenzieren der Hubsäulen: „Cmd_StartReferencing“

Nach jeder Unterbrechung der Stromversorgung der Hubsäulen bzw. Steuergeräte müssen diese neu referenziert werden. Der Referenzierungsvorgang wird in Gang gesetzt, indem der Button „Referenzieren“ betätigt wird. Zunächst wird die Bestromung der Schrittmotoren aktiviert, für den Fall, dass dies noch nicht geschehen ist. Anschließend wird die Funktion `stopEverything` ausgeführt, falls sich Hubsäulen noch in Bewegung befinden sollten. Die in Tabelle 4.3 aufgelistete Referenzierungsbewegung (32) wird ausgeführt und die Anzeige der Textfelder „Bewegungssequenzen“ wird zurückgesetzt
(`sequenceStatusReset`). Final werden alle Buttons zur Betätigung freigegeben, die Variable `start_trigger` auf „0“ gesetzt, um eine unbeabsichtigte Bewegung der Hubsäulen zu verhindern und die Variable `sequence_counter` ebenfalls auf „0“ gesetzt, um das Therapieprogramm in weiterer Folge von Anfang an beginnen zu lassen.

Starten/Fortsetzen des Therapieprogramms: „Cmd_StartSequences“

Um ein Therapieprogramm zu starten, oder nach einer Unterbrechung fortzusetzen wird der Button „Sequenzen Starten“ betätigt. Nach einer Überprüfung, ob die Motoren bestromt werden, wird ebenfalls kontrolliert, ob an einem der Steuergeräte ein Fehler aufgetreten ist. Sollte einer der beiden Fälle zutreffen, wird eine entsprechende Meldung im Textfeld ausgegeben. Der `start_trigger` wird auf „1“ gesetzt und die Funktion `startNextSequence` ausgeführt.

Folgende Buttons werden deaktiviert, um unzulässige Eingaben während des laufenden Therapieprogramms zu vermeiden:

- Sequenz hinzufügen
- Sequenzen Löschen
- Sequenzen Starten
- Patient Laden
- Patient Speichern

Unterbrechung des Therapieprogramms, Quittierung von Fehlern: „Cmd_StopProgrammAll“

Das Therapieprogramm wird durch Betätigen des Buttons „Sequenzen Stoppen / Fehler-Reset“ gestoppt. Hierzu wird zunächst der `start_trigger` deaktiviert, also auf „0“ gesetzt und die Funktion `stopEverything` ausgeführt. Anschließend werden alle Hubsäulen mit einer Geschwindigkeit von 30 mm/s eingefahren. Die in der Funktion `Cmd_StartSequences` deaktivierten Buttons, werden reaktiviert und eine entsprechende Meldung wird im Textfeld ausgegeben. Die Quittierung aufgetretener Fehler geschieht direkt im Steuergerät durch ein entsprechendes Signal (Funktion `stopEverything`) am digitalen Eingang 10 („Stop/Reset“).

Wiederholen des Therapieprogramms: „Cmd_CheckBoxEndless“

Durch Aktivierung/Deaktivierung der Checkbox „Endlosschleife“ wird die bool’sche Variable `endless_loop` auf „1“ oder „0“ gesetzt und eine Meldung am Textfeld ausgegeben. Die Funktion `startNextSequence` fragt diese Variable dann am Ende des Therapieprogramms ab. Ist die Endlosschleife aktiv, wird das Therapieprogramm wieder von vorne gestartet, ist sie inaktiv, wird das Therapieprogramm wie vorgesehen beendet.

4.3.7 Aufbau der Bewegungssequenzen

Im Grunde existieren zwei Arten von Bewegungssequenzen: Einfache („Single“) und duale Sequenzen. Bei den einfachen Bewegungssequenzen wird immer nur eine Hubsäule gleichzeitig aus- und eingefahren, während bei den dualen Sequenzen bis zu zwei Hubsäulen gleichzeitig aus- und eingefahren werden. Jede Bewegungssequenz setzt sich aus mehreren Bewegungsphasen zusammen, welchen wiederum eine Enumeration (`sequences_t`) zugeordnet ist. Alle Bewegungssequenzen beginnen mit der Startphase, welche zum Beispiel bei „Single 01“ folgendermaßen benannt ist: `SEQUENCE_SINGLE_01_START = 10`. Wie bei Enumerationen gefordert ist hier jeder Phase ein Wert zwischen 0 und 255 zugeordnet. Die maximale Anzahl an Bewegungsphasen bzw. Stufen pro Sequenz beträgt neun. Die Bewegungssequenzen starten und enden mit komplett eingefahrenen Hubsäulen. Um eine niedrige Latenz von Benutzereingaben während der laufenden Sequenzen zu gewährleisten, sind die Bewegungsphasen in der Main-Loop abgelegt.

Ingangsetzen der Bewegungssequenzen: „startNextSequence“

Die Hauptaufgabe dieser bool’schen Funktion ist es das Array `sequences` auszulesen und entsprechend dessen Inhalts die zugehörigen Bewegungssequenzen zu starten. Bei jedem Durchlauf der Funktion wird zu Beginn `sequenceStatusColor` aufgerufen, um die Anzeige zu aktualisieren. Anschließend wird überprüft, ob die erste Stelle des Arrays leer ist, also den Wert „0“ beinhaltet. Ist dies der Fall, wird eine Meldung an das Textfeld ausgegeben und der `start_trigger` deaktiviert. Die beim Start deaktivierten Buttons (siehe 4.3.6) werden reaktiviert und die bool’sche Funktion gibt den Wert „false“ aus. Des Weiteren wird überprüft, ob das Ende des Inhalts des Arrays erreicht ist. Dies kann entweder daraus resultieren, dass bereits alle 20 Positionen beschrieben sind, oder dass die letzte Sequenz erreicht ist, also die nächste Stelle im Array den Wert „0“ hat. Ist keine Endlosschleife aktiviert, so wird das Programm beendet und `sequence_counter` wird auf „0“ für einen zukünftigen Programmstart gesetzt, der `start_trigger` wird deaktiviert und eine Meldung wird über das Textfeld ausgegeben. Im Übrigen werden auch alle deaktivierten Buttons aktiviert und der Rückgabewert der Funktion ist „false“. Ist die Endlosschleife aktiviert, wird lediglich `sequence_counter` auf „0“ gesetzt, um das Array wieder von vorne zu durchlaufen und der Rückgabewert der Funktion wird als „true“ ausgegeben. Entsprechend des im Array `sequences` abgelegten Wertes wird über eine if-Abfrage die Laufvariable `sequence_` der zugehörigen Enumeration zugeordnet. Ebenso werden die für diese Bewegungssequenz vorgesehenen Fahrprofile (siehe Tabelle 4.3) zugeordnet und ein Rückgabewert „true“ ausgegeben.

Ablauf der Bewegungssequenzen

Nachdem die Funktion `startNextSequence` entsprechend der im Array abgespeicherten Zahl der Laufvariable `sequence_` die zugehörige Enumeration zugeordnet hat wird in der Main-Loop eine „Switch"-Verzweigung durchlaufen (`switch(sequence_)`, bei der nach der in `sequence_` abgespeicherten Enumeration unterschieden wird. Ist die letzte Sequenz des Programms durchlaufen und der Rückgabewert von `startNextSequence` „false" geht die Verzweigung zunächst zu `SEQUENCE_DONE`, final zu `SEQUENCE_NONE` und somit in den Leerlauf. Anhand des Beispiels `SEQUENCE_SINGLE_01_START` wird bei der Start-Phase jeder Sequenz abgefragt, ob der `start_trigger` aktiviert ist, sowie die Funktion `everythingReady` ausgeführt und abgefragt, um festzustellen ob alle Hubsäulen fahrbereit sind. Die erste Bewegung wird ausgeführt, indem eine Hubsäule bei Single- bzw. zwei Hubsäulen bei Dual-Sequenzen ausgefahren werden. Am Ende der Start-Phase wird durch `everythingReady` überprüft, ob die Bewegungen fertig ausgeführt wurden und anschließend die nächste Phase initialisiert, indem die Enumeration der folgenden Bewegungsphase in die Laufvariable gespeichert wird (`sequence_ = SEQUENCE_SINGLE_01_STAGE1`). In der darauffolgenden Phase werden wieder `start_trigger` und Bewegungsbereitschaft abgefragt. Anschließend werden/wird die folgende(n) Säule(n) ausgefahren und die in der vorherigen Phase ausgefahrene(n) Hubsäule(n) wieder eingefahren. Sind diese Bewegungen abgeschlossen wird wieder an die nächste Bewegungsphase übergeben, usw. . In der finalen Phase der Bewegungssequenz werden nur noch die letzten ausgefahrenen Hubsäulen eingefahren. Zusätzlich wird `sequence_ = SEQUENCE_DONE` in die Laufvariable gespeichert. Um zur nächsten Stelle des Arrays `sequences` voranzuschreiten wird der `sequence_counter` um 1 erhöht und die Funktion `startNextSequence` ausgeführt um die nächste Bewegungssequenz zu initialisieren.

4.3.8 Warnungen und Fehler

Fehler und Warnungen werden von den Steuergeräten produziert und allgemein als Warnung oder Fehler im GUI angezeigt. Um welche Warnung/Fehler es sich genau handelt ist an der Benutzeroberfläche nicht ersichtlich und muss direkt am jeweiligen Steuergerät ausgelesen werden. Somit dient die implementierte Warnungs-/Fehleranzeige dem Anwender als Hinweis, ob ein Fehler/Warnung, an welchem Steuergerät/Motor aufgetreten ist und nicht um welchen genau es sich handelt. Bei dem Therapiegerät sind folgende Warnungen und Fehler möglich (D1 Handbuch V2.0 S.88 f [6]):

Warnungen (Alerts):

- A10: Temperatur des Leistungsteils im Steuergerät über 85°C

Fehler (Errors):

- E01: Falsche Parameterkonfiguration
- E02: Motor Überstrom, möglicher Kurzschluss der Motorphasen
- E05: Zu geringe Spannung an Klemme X2.11-12
- E06: Zu geringe Spannung an Klemme X1.2-3
- E07: Zu hohe Spannung an Klemme X1.2-3
- E08: Zu geringe Spannung an Klemme X1.1-2
- E09: Zu hohe Spannung an Klemme X1.1-2
- E10: Leistungsteil im Steuergerät ist überhitzt
- E12: Endlagenschalter wurde ausgelöst

Anzeige der aufgetretenen Fehler und Warnungen: „errorHandler“

Diese Funktion fragt periodisch alle Fehler(5)- und Warnungsausgänge(4) der Steuergeräte ab und visualisiert das aufgetretene Ereignis durch Anzeigen eines entsprechenden Textfeldes im GUI. Ist ein Fehler am Steuergerät aufgetreten wird eine von acht roten Textboxen im GUI angezeigt und der `start_trigger` wird deaktiviert um das laufende Therapieprogramm anzuhalten. Zusätzlich wird in die Laufvariable `sequence_ = SEQUENCE_ERROR` gespeichert um das Therapieprogramm zu unterbrechen. Im Gegensatz dazu wird bei einer auftretenden Warnung lediglich eine von acht gelben Textboxen im GUI angezeigt, das Therapieprogramm wird jedoch nicht unterbrochen. Anhand der roten und gelben Textboxen ist sofort ersichtlich, an welchem Steuergerät/Motor der Fehler bzw. die Warnung aufgetreten ist. Tritt bei einem Steuergerät ein Fehler auf, so stoppt die Software des Steuergeräts laufende Bewegungen sofort. Der `errorHandler` ist in der „Main-Loop“ des Programmcodes implementiert und wird periodisch ausgeführt. Da es sich bei der Anzeige der Fehlermeldungen um keine unmittelbar sicherheitskritische Funktion handelt, wird sie alle 5 Sekunden ausgeführt um die Ressourcen des Mikrocontrollers zu schonen. Das Therapieprogramm stoppt also spätestens 5 Sekunden nach dem Auftreten eines Fehlers.

4.3.9 Sonstige Ausgaben

Quittierung des Warnhinweises zu Programmstart: „Cmd_Accept“

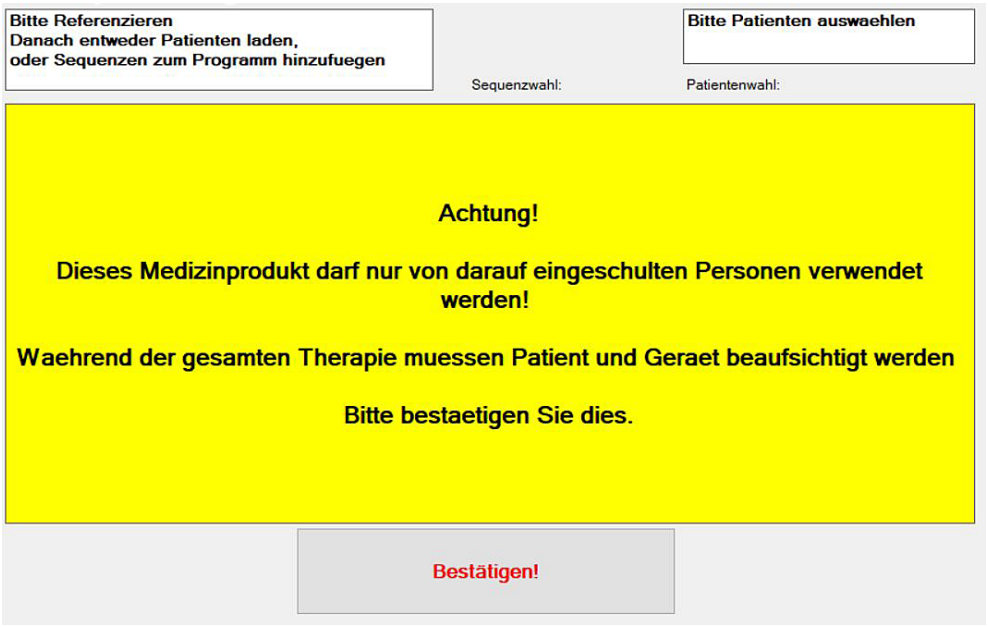


Abbildung 4.5 Warnhinweis bei Programmstart

Im Sinne der hinweisenden Sicherheit wird bei Programmstart der in Abbildung 4.5 gezeigte Warnhinweis ausgegeben, der absichtlich alle Bedienelemente im GUI überdeckt. Erst mit der Bestätigung dieses Hinweises durch den Button „Bestätigen!“ verschwindet diese Meldung, ebenso wie der Button selbst. Zusätzlich werden dadurch erst die von Beginn an deaktivierten Buttons „Referenzieren“, „Sequenzen Starten“, sowie die Checkbox für die Bestromung der Motoren aktiviert. (siehe auch Abbildung 4.4) Da diese drei Bedienelemente direkt Bewegungen oder eine Bestromung der Schrittmotoren auslösen, werden sie beim Start des Therapiegerätes deaktiviert.

4.4 Resultate der Risikoanalyse

In weiterer Folge der Risikoanalyse laut EN 14971 [1] bezogen auf die Software des Therapiegeräts wurden während des Entwicklungsprozesses (siehe Abbildung 3.5 und 3.6) in Summe 15 Risiken identifiziert und klassifiziert. Wie in Tabelle 4.4 dargestellt, sind drei Risiken ohne Abhilfe- bzw. Sicherheitsmaßnahmen als „nicht akzeptabel“ zu bewerten. Weitere neun Risiken haben mit der Eintrittswahrscheinlichkeit „selten“ zusätzlich eine gewisse Wahrscheinlichkeit während des Betriebs aufzutreten.

Tabelle 4.4 Risikomatrix: Verteilung der Risiken ohne Sicherheitsmaßnahmen

	Eingetretener Schaden				
Eintrittswahrscheinlichkeit	gering	mittel	schwer	katastrophal	Risikostufe
häufig			2		1
manchmal		1			1
gelegentlich	1				1
selten	2	3	3		2
unwahrscheinlich	1	2			3
unglaublich					4

Nach Umsetzung der vorgesehenen Sicherheitsmaßnahmen sind alle 15 Risiken als „akzeptabel“ einzustufen. (Tabelle 4.6) Ebenso weisen sie nun eine Eintrittswahrscheinlichkeit von maximal „unwahrscheinlich“ auf. Im Zuge der Risikobeherrschung wurden alle auftretenden Risiken, ob „akzeptabel“ oder „nicht akzeptabel“ weiter reduziert. Da es sich bei dem Therapiegerät um einen Prototypen handelt, werden Risiken der Stufe Risikostufe 3 (siehe Tabelle 3.3) im Sinne des ALARP-Prinzips nicht weiter reduziert, da der Nutzen weiterer Maßnahmen in keinem annehmbaren Verhältnis zu den Kosten stehen würde. Bei den umgesetzten Sicherheitsmaßnahmen konnten keine Rückwirkungen festgestellt werden, für die weitere Abhilfemaßnahmen ergriffen hätten werden müssen. Eine detaillierte Liste mit allen identifizierten Risiken im Hinblick auf die Software, deren Klassifizierung sowie der zugehörigen Abhilfemaßnahmen ist Tabelle 4.5 zu entnehmen.

Tabelle 4.5 Detaillierte Resultate der Risikoanalyse

Detaillierte Resultate der Risikoanalyse

Ursache	NC	SFC	Mögliche Auswirkungen	EW	S	RS	Abhilfemaßnahmen	RW	SS	DM	EW	S	RS	GS
Übertragungsfehler USB zu Arduino		X	Funktion des Therapigeräts beeinträchtigt	uw	mi	III	Verwendung von sicheren, vorprogrammierten Bewegungen	N	1	J	uw	g	IV	
Übertragungsfehler Arduino zu Steuergerät		X	Funktion des Therapigeräts beeinträchtigt	uw	mi	III	Verwendung von sicheren, vorprogrammierten Bewegungen	N	1	J	uw	g	IV	
Gelockerte 5-Bit Binärleitung		X	Ausführung einer flaschen Bewegung	se	mi	III	Risikoreichere Bewegungen mit höherwertigen Bits/Zahlen belegen	N	1	J	uw	mi	III	uw*m=III
							Referenzierung vor Therapiestart -> Referenzierungsbefehl so abgespeichert, dass alle 5 Binärleitungen auf High gesetzt werden müssen und somit auch alle 5 Leitungne überprüft werden	N	1	J	uw	mi	III	
Signalleitung löst sich		X	Funktionsbeeinträchtigung	se	ge	III	Verwendung von Adernendhülsen	N	1	J	uw	ge	IV	uw*g = IV
							Verwendung von Kabelbindern zur Fixierung	N	1	J	uw	ge	IV	
Start- / Freigabe- / Bereitschaftsleitung löst sich		X	Therapiestart nicht möglich	se	ge	III	Verwendung von Adernendhülsen	N	1	J	uw	ge	IV	uw*g = IV
							Verwendung von Kabelbindern zur Fixierung	N	1	J	uw	ge	IV	
Stopleitung löst sich		X	Therapieabbruch nicht möglich	se	mi	III	Verwendung von Adernendhülsen	N	1	J	uw	mi	IV	UW*M=III
							Verwendung von Kabelbindern zur Fixierung	N	1	J	uw	mi	IV	
							Programmierung einer Redundanz in Form des Start Triggers	N	1	J	uw	mi	IV	

Detaillierte Resultate der Risikoanalyse

Alarm- oder Fehlerleitung löst sich		X	Fehler und Alarme werden nicht angezeigt -> Therapie stoppt (Entsprechende Hubsäule bewegt sich nicht mehr)	se	mi	III	Verwendung von Adernendhülsen	N	1	J	uw	mi	IV	uw*m=III
							Verwendung von Kabelbindern zur Fixierung	N	1	J	uw	mi	IV	
							Überwachung der Therapie	N	1	J	se	mi	III	
Falsche Eingabe des Benutzers	X		Verletzung des Patienten	h	sw	I	Auswahl der Sequenzen aus einem vorgegebenen Bewegungspool	N	1	J	ug	sw	IV	
Sicherheitskritische Bewegungen hinsichtlich Geschwindigkeit, Beschleunigung und Auslenkung		X	Verletzung des Patienten	h	sw	I	Ausschließliche Verwendung von vorgegebenen Bewegungssequenzen	N	1	J	ug	sw	IV	ug*sw=IV
							Hardwaremäßige Begrenzung der Auslenkung durch einen Anschlag	N	1	N	ug	sw	IV	
Unzulässige Änderungen der vorprogrammierten Bewegungen	X		Verletzung des Patienten	se	sw	III	Zugangsbeschränkung (Passwort, externe Verbindung notwendig)	N	1	N	ug	sw	IV	
Ausfall durch Fehler bei Steuergerät oder Motor		X	Therapie nicht mehr möglich	uw	ge	IV	Visuelles Feedback an GUI	N	1	J	uw	ge	IV	
Hubsäule fährt unkontrolliert aus		X	Verletzung des Patienten	se	sw	III	Steuergerät mit hoher Zuverlässigkeit verwenden	N	1	J	uw	sw	III	uw*s=III
				se	sw	III	Hardware Anschlag gegen max. Auslenkung	N	1	N	ug	sw	IV	

Detaillierte Resultate der Risikoanalyse

Unbeabsichtigte Bewegung nach Spannungsausfall		X	Verletzung des Patienten	ma	mi	II	Referenzierung nach Spannungsrückkehr notwendig	N	1	J	ug	mi	IV	
Falsche Auslenkung der Hubsäulen	X	X	Verletzung des Patienten	se	sw	III	Referenzierung vor Therapiestart notwendig	N	1	J	ug	sw	IV	uw*s=III
							Verwendung eines Schrittmotors zur genauen Positionierung	N	1	J	uw	sw	III	
							Verwendung eines überdimensionierten Motors zur Vermeidung von Schleppfehlern	N	1	J	uw	sw	III	
USB-Verbindung löst sich während der Therapie		X	Therapie kann nicht mehr über GUI gestoppt werden	gl	ge	III	Fixierung des USB-Kabels	N	1	N	uw	ge	IV	uw*g = IV
							Not-Aus Schalter	N	1	J	ug	ge	IV	

Tabelle 4.6 Risikomatrix: Verteilung der Risiken nach Anwendung der Sicherheitsmaßnahmen

	Eingetretener Schaden				
Eintrittswahrscheinlichkeit	gering	mittel	schwer	katastrophal	Risikostufe
häufig					1
manchmal					1
gelegentlich					1
selten					2
unwahrscheinlich	6	3	2		3
unglaublich		1	3		4

Bewertung des Gesamtrisikos

Da die einzelnen akzeptablen Risiken, sowie deren Ursachen und Abhilfemaßnahmen, keinerlei Wechselwirkungen untereinander aufweisen, ist das verbleibende Gesamtrisiko der Software des Therapiegerätes, wie die Einzelrisiken ebenfalls als „akzeptabel“ zu bewerten.

4.5 Sicherheitsvorkehrungen

Grundlegendes Sicherheitskonzept

Das grundlegende (nicht rein softwareseitige) Sicherheitskonzept besteht aus der Trennung der Programmierung der einzelnen Bewegungen vom restlichen Programmcode, sowie deren Beschränkung an einem vorprogrammierbaren Steuergerät mit hoher Zuverlässigkeit. Wie in Kapitel 3.3.2 erwähnt wird das Steuergerät über ein Webinterface vorprogrammiert, indem es mit einem Ethernetkabel zu einem Notebook verbunden wird. Dieses Webinterface lässt sich darüber hinaus mit einem Passwort sichern. Die Grenzwerte für maximalen Hub, Geschwindigkeit und Beschleunigung beschränken die Fahrprofile zusätzlich. Durch diese Trennung des übergeordneten Programmcodes des Arduinos von den eigentlichen Bewegungen lässt sich das Risiko einer versehentlich umprogrammierten risikoreichen Bewegung einer Hubsäule deutlich vermindern. Ebenso lässt sich dadurch eine beabsichtigte Manipulation der Fahrprofile verhindern (vorhersehbarer Missbrauch). Ein Nachteil, der sich dadurch ergibt, ist ein minimal erhöhter Aufwand für die Programmierung, da jedes Steuergerät einzeln mit dem Notebook verbunden werden muss. Der größte Nachteil, der zusätzlich aus sicherheitstechnischer Sicht entsteht, ist dass mit dem Steuergerät eine sogenannte SOUP, also Software unbekannter Herkunft dem System hinzugefügt wird. In Rücksprache mit dem Hersteller wurde via eMail Korrespondenz (Stand Jänner 2020) versichert, dass bei korrekter Beschaltung des Steuergerätes keinerlei kritische Fehlfunktion

aufgetreten sind. Laut Hersteller wurden seit Markteinführung des Steuergerätes im April 2017 keine für unsere Anwendung kritische Fehlfunktionen gemeldet.

Die im Folgenden beschriebenen, weiteren Sicherheitsvorkehrungen wurden bei der Grundkonzeption der Ansteuerung bzw. im Zuge des Risikomanagements erarbeitet.

Beschränkung der Benutzereingaben

Ein Sicherheitskonzept war von Anfang an eine Beschränkung der Benutzereingaben um so risikoreiche Bewegungen der Hubsäulen zu verhindern, die dem Patienten Schaden zufügen könnten. Zu diesen Beschränkungen zählen die Zusammenstellung des Therapieprogramms aus vorgegebenen Bewegungssequenzen, sowie die Beschränkung der in den Steuergeräten abgespeicherten Fahrprofile auf sichere Bewegungen. Diese Fahrprofile können nur geändert werden, indem das jeweilige Steuergerät durch ein Netzwerkkabel mit einem PC verbunden und über ein (noch nicht implementiertes) Passwort freigeschaltet wird.

Ordnungsgemäße Fixierung der Signalleitungen

Viele in der Risikoanalyse aufgeführte Gefahren haben ihre Ursache in lockeren bzw. losgelösten Signalleitungen (USB und Binärleitungen zwischen Mikrocontroller und Steuergeräten (siehe Kapitel 3.3.2). Diese Risiken lassen sich mit wenig Aufwand durch eine Fixierung der Leitungen mit Kabelbindern und der Verwendung von Aderendhülsen bei Litzenkabeln minimieren.

Programmierte Redundanz: „start_trigger“

Für den Fall, dass sich die am ehesten sicherheitskritische Signalleitung für das Stop-Signal dennoch löst, wurde die Variable `start_trigger` in das Programm eingefügt, die vor jeder Bewegung der Hubsäulen abgefragt und zeitgleich mit einem gesendeten Stop-Signal deaktiviert wird. Ist eine dieser Signalleitungen unterbrochen, stoppt die entsprechende Hubsäule zwar nicht unmittelbar, aber nach deren Bewegung, werden keine weiteren Bewegungen mehr in Gang gesetzt und das Therapiegerät kommt zum Stillstand.

Verwendung von Steuergeräten mit hoher Zuverlässigkeit

Um unkontrollierten Bewegungen der Hubsäulen vorzubeugen und eine Ansteuerung der Schrittmotoren mit hoher Genauigkeit zu gewährleisten werden besonders hochwertige und zuverlässige Steuergeräte verwendet. Aus der Programmierung der Steuergeräte ergibt sich die Eigenschaft, dass die Hubsäulen nach jeder Deaktivierung von deren Spannungsversorgung neu referenziert werden müssen, um weitere Fahrbefehle annehmen zu können.

Hierdurch werden unkontrollierte Bewegungen nach einem Spannungsausfall am System verhindert.

Anordnung der Fahrprofile

Die bis zu 32 Fahrprofile (Tabelle 4.3) werden über binäre 5-Bit Leitungen durch das Anlegen von 5V als „1“ bzw. 0V als „0“ ausgewählt. Löst sich eine dieser fünf Leitungen, besteht die Gefahr, dass ein falsches Fahrprofil ausgewählt wird, das zu einer potenziell gefährlichen Bewegung der Hubsäule führt. Wie in Tabelle 4.3 ersichtlich wurden als Abhilfemaßnahme risikoreichere Fahrprofile, die dem Ausfahren der Hubsäulen dienen, höherwertigen Zahlen zugeordnet (Nr.12-Nr.16). Zusätzlich ist die Referenzierbewegung dem Wert 32 zugeordnet, bei dem alle 5 Bit den Wert „1“ haben müssen, also an allen Leitungen 5V anliegen muss. Ist nun eine der fünf Signalleitungen unterbrochen, kann die Hubsäule nicht referenziert und somit in Folge dessen auch keine Fahrbefehle angenommen werden.

Mechanische Schutzmaßnahmen gegen Fehler im SOUP

Sollte der Fall eintreten, dass eine Hubsäule entgegen aller getroffenen Sicherheitsmaßnahmen unkontrolliert ausfährt, kann ein mechanischer Anschlag an den Hubsäulen eine unzulässig starke Auslenkung verhindern. Der Schrittmotor dreht dann durch und das Gerät kann deaktiviert werden, ohne dass Schäden an Mensch und Maschine entstehen. Zusätzlich tritt an der Spindel ab Beschleunigungen von ca. 150mm/s^2 und Geschwindigkeiten von 150mm/s eine Selbsthemmung auf, sodass diese Bewegungsparameter ebensfalls mechanisch beschränkt sind. Letzteres war nicht in der Konstruktion vorgesehen, wurde aber in Extremfalltests wiederholt nachgewiesen.

4.6 Verifizierung und Validierung der Software

Während sich durch die Verifizierungstests objektive Anforderungen, wie eine Programmunterbrechung über das GUI, oder der maximale Hub der Hubsäulen sehr gut quantitativ bestätigen lassen, müssen subjektive Anforderungen, wie zum Beispiel eine „einfache Bedienung“ und Benutzerfreundlichkeit im Rahmen von Usability-Tests validiert werden. Ebenso lässt sich eine „größtmögliche Sicherheit des Patienten“ nicht direkt in solchen Tests festhalten. Dieser Thematik widmet sich deshalb der gesamte Prozess des Risikomanagements. Dennoch zeigt das Resultat dieser Tests, dass das Therapiegerät den objektiven Anforderungen gerecht wird.

Nur einige Validierungsschritte im Bezug auf die Anforderungen der Software wurden durchgeführt. Weitere Schritte, wie die Validierung der Usability und der klinischen Bewertung sind noch nachzuholen. Alle 16 Checklisten, sowie alle 57 Testprotokolle haben mit dem 05.05.2020 den Status „Bestanden“, womit folglich auch alle Funktionstests erfolgreich durchgeführt wurden.

Verifizierung der Bewegungssequenzen

Für die Verifizierung der von Herrn Mauerhofer vorgegebenen Bewegungssequenzen [10] wurde die Liste mit den einzelnen Bewegungsabläufen aufbereitet und in ein Testprotokoll integriert (Anhang E und F bzw. Abbildung 4.6). Bei den Funktionstests wurde jeder Schritt der Bewegungssequenzen kontrolliert und dokumentiert. Zusätzlich wurde die maximale Auslenkung der Hubsäulen überwacht und ebenfalls im Testprotokoll festgehalten. In Summe entsprechen alle programmierten Bewegungssequenzen den Vorgaben, wie der Übersicht der Testprotokolle in Anhang E zu entnehmen ist. Des Weiteren sind zwei Protokolle für je eine Art von Bewegungssequenzen (Single und Dual) als Beispiel dem Anhang E beigefügt. Abbildung 4.6 zeigt einen Verifizierungstest für eine duale Bewegungssequenz.

Testprotokoll: **TP-5.12** **Dual 1 (6cm)** **V1.0**

Test Autor:	Christian Krammer	**Datum:**	04.05.2020
Freigabe:	Christian Krammer	**Datum:**	04.05.2020
Tester:	Christian Krammer	**Datum:**	05.05.2020

Testergebnis Bestanden

Kurzfassung des Tests:
Bei diesem Test wird der korrekte Ablauf der Bewegungssequenz "Dual1 (6cm)" überprüft.
Ebenso muss die maximale Auslenkung der Hubsäulen geprüft werden.
Der Test kann nach der Vorlage von TP-4.1 mit der entsprechenden Bewegungssequenz durchgeführt werden.

Nr.	Eingabe/Ablauf		Erwartetes Ergebnis		Ja/Nein	Anmerkung
FT-5.12.1	Start der Bewegungssequenz "Dual1 (6cm)"		Ablauf der Bewegungssequenz:			
Bewegung Nr.	Partie 1	Säule Nr.	Partie 2	Säule Nr.		
1	Schulter Links	2	Gesäß Rechts	5	JA	
2	Gesäß Links	6	Schulter Rechts	1	JA	
3	Bein Links	8	Bein Rechts	7	JA	
4	Gesäß Links	6	Gesäß Rechts	5	JA	
5	Schulter Links	2	Gesäß Rechts	5	JA	
6	Gesäß Links	6	Schulter Rechts	1	JA	
7	Bein Links	8	Bein Rechts	7	JA	
8	Gesäß Links	6	Gesäß Rechts	5	JA	
			Gesamter Ablauf laut Vorgabe:		JA	
FT-5.12.2			Maximale Auslenkung aller Hubsäulen mit Ausnahme der Rückenpartie (3 & 4) muss 6cm betragen		JA	
FT-5.12.3			Maximale Auslenkung der Hubsäulen am Rücken (3 & 4) darf nicht höher, als 4cm sein		JA	
FT-5.12.4			Alle Säulen kommen im eingefahrenen Zustand zum		JA	

Abbildung 4.6 Beispiel des Verifizierungstests einer dualen Bewegungssequenz

Konfiguration der Steuergeräte (SOUP)

Im Zuge des Verifizierungs- und Validierungsvorgangs wurden die Steuergeräte, welche als SOUP zu klassifizieren sind, ebenfalls im Bezug auf die Anforderungen geprüft. Im Gegensatz zu dem eigentlichen Programmcode wurden hierbei keine Funktionstests durchgeführt, sondern Checklisten abgearbeitet, um zu gewährleisten, dass die in den Steuergeräten gespeicherten Parameter und Fahrprofile keine Gefahrensituationen herbeiführen können. Eine Übersicht aller geprüften Steuergeräte und den zugehörigen Checklisten, sowie die Ergebnisse der Tests und jeweils ein Beispiel für einen Checklisten Typ befinden sich in Anhang D.

4.7 Softwarespezifikationen laut OVE EN 62304

In diesem Abschnitt werden die wichtigsten Spezifikationen der Software zur Ansteuerung des Therapiegerätes im Bezug auf die Norm OVE EN 62304 [2] aufgelistet. Hierbei werden allerdings nur die auf dieses Projekt zutreffenden Punkte berücksichtigt. Ein von der Norm gefordertes QM-System wurde nicht umgesetzt.

OVE EN 62304-2016, 4.3: Software Sicherheitsklassifizierung

Die Software wird der Norm entsprechend unter Berücksichtigung der Risikobeherrschungsmaßnahmen der **Sicherheitsklasse B** zugeordnet, da das Therapiegerät noch ein gewisses Verletzungspotential birgt, aber zu keinen schweren Verletzungen, oder gar dem Tod führen kann. Ohne die entsprechenden Maßnahmen wäre die Software der Sicherheitsklasse C zuzuordnen, da das Therapiegerät dann das Potential hätte, dem Patienten schwere Verletzungen zuzufügen.

OVE EN 62304-2016, 5.1.1: Planung der Softwareentwicklung

Die Planung der Softwareentwicklung zur Ansteuerung des Therapiegerätes geschah anhand des **V-Modells**. (Kapitel 3.5.2) Ein konkreter Softwareentwicklungsplan, wie von der Norm vorgesehen, wurde allerdings nie erstellt. Lediglich die wesentlichen Eckpunkte/ Meilensteine der Softwareentwicklung wurden notiert:

- Erzeugung der einzelnen Bewegungen durch die Steuergeräte
- Ansteuerung der Steuergeräte
- Erstellung des GUI in MegunoLink
- Eingabe der Therapieprogramme in das GUI
- Implementierung von Sicherheitsmaßnahmen

- Ausgabe von Fehlern und Warnungen
- Implementierung der Bewegungssequenzen
- Visualisierung des Therapieprogramms
- Dauerhaftes Speichern der Therapieprogramme
- Verifizierungstests

Ebenso wurden die zu liefernden Ergebnisse aus den Anforderungen an die Software bestimmt. (siehe Kapitel 3.5.1)

OVE EN 62304-2016, 5.2.2: Inhalt der Software-Anforderungen

Ein Dokument zur Erfassung der Software Anforderungen wurde erstellt und ist als Anhang C beigefügt.

OVE EN 62304-2016, 5.3: Design der Software-Architektur

Die Architektur der Software wird durch Abbildung 4.1 wiedergegeben. Zusätzlich zu den Softwarekomponenten bzw. -Einheiten innerhalb des Arduino Codes werden auch die weiteren Komponenten in Form des GUIs MegunoLink und der SOUP der Steuergeräte abgebildet. Im Gegensatz zu den Komponenten innerhalb des Arduino Codes sind die externen Komponenten über die Schnittstellen „USB“ (MeguniLink) und nicht näher genormte, binär verwendete Signalleitungen (igus Steuergerät Dryve D1) mit dem Mikrocontroller verbunden. Als Leistungsanforderung an die SOUP ist eine hohe Zuverlässigkeit zu nennen. Da die SOUP nur auf der entsprechenden Hardware läuft ist eine weitere Hardwarespezifikation der SOUP nicht notwendig.

OVE EN 62304-2016, 5.4: Detailtieres Software Design

Die Aufzählung und Beschreibung der Softwarekomponenten und deren weitere Unterteilung in Softwareeinheiten wurde bereits in Kapitel 4.3 vorgenommen. Beispielsweise besteht die Softwarekomponente zum Speichern und Laden (Abschnitt 4.3.5) von Patienten aus den beiden Funktionen (= Softwareeinheiten) `Cmd_SavePatient` und `Cmd_LoadPatient`. Abbildung 4.1 seigt also alle Softwareeinheiten im Arduino Code, ohne sie nach Zugehörigkeit zu einer Komponente zu gruppieren.

Im Rahmen des technischen Systementwurfs (siehe Abbildung 3.5) wurden die Funktionalität für die einzelnen Softwarekomponenten festgelegt, welche darauf folgend in der Phase der Komponentenspezifikation in weitere Softwareeinheiten (bzw. Funktionen) aufgeteilt und schließlich implementiert wurden. Die daraus resultierenden Softwareeinheiten und

-Komponenten wurden anschließend durch Komponenten- und Integrationstests auf ihre Funktionalität überprüft.

OVE EN 62304-2016, 5.7: Prüfung des Software-Systems

Die Software, sowie das Gerät an sich wurden im Rahmen der Verifizierung diverser Funktionstests unterzogen, die alle Funktionalitäten und besondere Eigenschaften des Software-Systems abdecken und überprüfen.(Kapitel 3.5.4 und 4.6)
Die Ergebnisse dieser Prüfungen sind in der technischen Dokumentation enthalten, Beispiele hierfür befinden sich in Anhang D und E bzw. Abbildung 4.7.

Testprotokoll: **TP-1.4** **Testsequenz** **V1.0**

Test Autor:	Christian Krammer	**Datum:**	30.04.2020
Freigabe:	Christian Krammer	**Datum:**	04.05.2020
Tester:	Christian Krammer	**Datum:**	04.05.2020

Testergebnis Bestanden

Kurzfassung des Tests:
Nach der Referenzierung wird mit der Testsequenz die Funktionalität aller Hubsäulen überprüft

Nr.	Eingabe	Erwartetes Ergebnis	Ja/Nein	Anmerkung
FT-1.4.1	Aus der Liste "Sequenzwahl" (unten) die Sequenz "Test: Alle gleichzeitig" auswählen	Blaue hinterlegung der Auswahl	JA	
FT-1.4.2	Button "Sequenz hinzufügen" betätigen	"Funktionstest" erscheint im obersten linken Feld unter "Bewegungssequenzen"	JA	
FT-1.4.3	Button "Sequenzen starten" betätigen	Alle 8 Hubsäulen bewegen sich synchron einmal auf - und ab	JA	
		Die Hubsäulen bei der Rückenpartie fahren auf 4cm aus	JA	
		Alle weiteren 6 Hubsäulen fahren auf 6cm Hub aus	JA	
		Das Textfeld "Funktionstest" wird grün hinterlegt	JA	
		Alle Säulen kommen im eingefahrenen Zustand zum Stillstand	JA	
FT-1.4.4	Button "Sequenzen Löschen" betätigen	Der Inhalt aller Textfelder "Bewegungssequenzen" ist gelöscht und weiß hinterlegt	JA	

Abbildung 4.7 Beispiel eines funktionellen Verifizierungstests

OVE EN 62304-2016, 5.8.2-5.8.3: Dokumentation und Bewertung restlicher bekannter Anomalien

Im Verlauf der Entwicklung des Therapiegeräts sind zwei Anomalien aufgetreten, deren genauer Ursprung nicht bekannt ist und somit auch nicht gelöst werden konnten.

1. Fehler bei erster Referenzierung nach Inbetriebnahme des Therapiegeräts:
   Bei der Inbetriebnahme / Spannungswiederkehr kommt es selten vor, dass die Motorsteuergeräte eine Fehlermeldung ausgeben, laut der es ein Problem mit deren Spannungsversorgung vorliegt. Dieser „Fehler“ ist durch eine Quittierung und eine anschließende Referenzierung über das GUI behebbar. Diese Anomalie wird als unkritisch bewertet.

2. Fortsetzung des Therapieprogramms bei Unterbrechung der USB-Verbindung von Laptop zu Mikrocontroller:
   Wird während eines laufenden Therapieprogramms die Verbindung zwischen Mikrocontroller und Laptop getrennt, wird dieses weiterhin fortgesetzt und kann folglich nicht mehr durch das GUI beendet werden. Durch das Wiederherstellen der USB Verbindung geht der Mikrocontroller bzw. das darauf laufende Programm in den „Start“-Zustand und das Programm wird sofort angehalten. Alternativ lässt sich das laufende Programm durch Betätigen des Not-Aus-Tasters am Therapiegerät beenden. Auch diese Anomalie ist als unkritisch anzusehen.

OVE EN 62304-2016, 7: Software-Risikomanagement-Prozess

Ein Risikomanagementprozess nach EN ISO 14971 [1] wurde während des Entwicklungsprozesses angewandt (Kapitel 3.4 und 4.4). Im Rahmen dessen wurde auch das Steuergerät „igus Dryve D1“ als SOUP identifiziert, das zu einer Gefährdungssituation beitragen könnte. Durch entsprechende Hardwaremaßnahmen wurden diese auf ein vertretbares Maß reduziert. Ebenso gibt es laut dessen Hersteller keinerlei bekannte Anomalien bei dessen korrekter Beschaltung.

OVE EN 62304-2016, 8.2.2: Implementierung von Änderungen

Während im Bezug auf die Wartung der Software keine Vorkehrungen getroffen wurden, liegt der technischen Dokumentation eine Anleitung zur Modifikation bzw. Erstellung neuer Bewegungssequenzen bei.

5 Diskussion

5.1 Risikomanagement und Softwareentwicklung

Im Zuge der Aufgabenstellung dieser Masterarbeit, bei der eine Ansteuerung für das sich in Entwicklung befindliche Therapiegerät zu entwickeln war, wurde von Anfang an besonderes Augenmerk auf die Sicherheit des Patienten gelegt. Folglich wurden die zutreffenden Normen OVE EN 62304 „Medizingeräte-Software - Software-Lebenszyklus-Prozesse“, sowie OVE EN 60601-1 „Medizinisch elektrische Geräte - Allgemeine Festlegungen für die Sicherheit einschließlich der wesentlichen Leistungsmerkmale“, so weit es möglich war berücksichtigt. Da die Entwicklung nicht im Rahmen eines von der Norm EN 62304 geforderten QM-Systems stattfand, konnten viele der vorgesehenen Punkte nicht berücksichtigt werden. Ebenso wurde die erstellte technische Dokumentation (insbesondere der Software) auf die wesentlichen Punkte beschränkt.

In dem von der Norm EN 62304 geforderten Risikomanagement entsprechend der Norm EN 14971 für die Anwendung an Medizinprodukten, wurden basierend auf den Arbeiten von Herrn Mauerhofer [10] und Herrn Steindorfer [11] 15 weitere Risiken identifiziert, die aus dem Ansteuerungskonzept resultierten. Alle Risiken wurden als Folge der Abhilfemaßnahmen auf die Risikostufen 3 (5 Risiken) und 4 (10 Risiken) reduziert und somit als „akzeptabel“ eingestuft.

5.2 Auslegung und Hardware

Da die Software laut EN 62304:2016 4.3 ohne Abhilfemaßnahmen zu einer Gefährdungssituation beitragen kann, deren resultierender Schaden möglicherweise als „schwer“ einzuschätzen ist, wurde die Software an sich zunächst mit Sicherheitsklasse C eingestuft. Gerade für Frühchen und Säuglinge birgt die unkontrollierte Bewegung einer, oder mehrerer Hubsäulen ein hohes Gefahrenpotenzial. Mit den gesetzten Maßnahmen lässt sich die Software als Klasse B einstufen, da so keine schweren Verletzungen des Patienten mehr möglich sein sollten.

Begonnen wurde die praktische Arbeit an diesem Projekt mit der Konzeptionsphase, bei der ein den Anforderungen entsprechendes Ansteuerungskonzept entwickelt werden sollte. Aus diesen Anforderungen und den Vorgaben des anzusteuernden Schrittmotors wurde nach Abwägen der Vor- und Nachteile die Variante ausgewählt, bei der Steuergeräte des Motorherstellers „igus“ durch einen „Arduino Mega“ Mikrocontroller über eine grafische Benutzeroberfläche „MegunoLink“ angesteuert werden. Der Vorteil dieser Konfiguration ist die Kompatibilität zwischen Motor und Steuergerät, sowie einige Sicherheitsaspekte. Alle Bewegungen der Hubsäulen bzw. des Schrittmotors werden auf den Steuergeräten vorprogrammiert und gespeichert. Sie können vom Anwender selbst nicht nachträglich verändert werden und so gegebenenfalls den Patienten gefährden. Des Weiteren ist durch die Software der Steuergeräte, die als SOUP klassifiziert worden ist eine präzise Positionierung der Hubsäulen gewährleistet.

Bei der Ansteuerung der Hubsäulen handelt es sich um eine reine Steuerung. Es gibt also keine Rückmeldung der Position der Hubsäulen, da dies einen Encoder am Schrittmotor voraussetzen würde, auf welchen aus Kostengründen verzichtet wurde. Da die Schrittmotoren für den Anwendungszweck an Säuglingen bis zur 14. Lebenswoche stark überdimensioniert wurden, sind Schrittfehler durch deren Überlastung nicht zu befürchten. Bei Experimenten, bei denen händisch ein Druck von mehreren Kilogramm auf eine Hubsäule ausgeübt wurde, konnte keine Überlast festgestellt werden. Somit ist eine Überbelastung der Motoren im bestimmungsgemäßen Gebrauch äußerst unwahrscheinlich.

Ursprünglich war aufgrund der Herstellerangaben ein medizinisches 24V Netzteil pro Steuergerät vorgesehen. Nach der Durchführung einiger Strommessungen (Kapitel 3.3.3) konnte entsprechend deren Resultate die Anzahl auf zwei 24V Schaltnetzteile reduziert werden. (Tabelle 4.1 und 4.2) Es werden also vier Steuergeräte über ein 24V Schaltnetzteil mit Spannung versorgt.

Die Arduino IDE verfügt über keinerlei integrierte Debugging-Möglichkeiten. Deshalb wurden während der gesamten Softwareentwicklung Debugging-Informationen direkt über den in MegunoLink integrierten „Serial-Monitor“ mit den Befehlen `Serial.print` und `Serial.println` ausgegeben. Dies diente hauptsächlich der Überwachung von Reihenfolge und Ablauf der Bewegungssequenzen. Da sie nur wenig Speicher benötigen, den Programmablauf selbst nicht beeinflussen und sie folgenden Problemlösungen von Nutzen sein könnten, existieren diese Ausgaben auch in der vorläufig finalen Version der Ansteuerungssoftware.

Die Entwicklung der Ansteuerung und der Software fand in enger Zusammenarbeit mit den Kollegen statt. Wie aus Kapitel 3.5.2 hervorgeht wurden Hardware und Software par-

allel entwickelt, was eine gewisse Flexibilität verlangte. Während die Hardware ständig erweitert wurde, mussten Ansteuerung und Software daran angepasst werden um das Therapiegerät schrittweise in seiner Funktionalität zu erweitern. Die im V-Modell (Abbildung 3.5) dargestellten Komponenten- und Integrationstests wurden parallel dazu durchgeführt, jedoch wurden nicht alle dokumentiert, da eine Erstellung von Testprotokollen hierfür den Rahmen gesprengt hätten.

5.3 Verifizierung und Validierung

Da die finalen Funktionstests (Verifizierungstests) direkt aus den Anforderungen an die Ansteuerung erstellt wurden, lassen sich diese zumindest teilweise als Validierungstests darstellen. Bei den hier abgeleiteten Validierungstests konnten lediglich die Anforderungen an die Software überprüft werden. Dies ist möglich, da Funktionen überprüft wurden, die sich aus den Anforderungen ergaben. Aus den positiv absolvierten Funktionstests ergibt sich somit wieder eine Erfüllung der Anforderungen. Validierungstests (Überprüfung auf die Anforderungen) und Verifizierungstests (Überprüfung auf die Funktion) wurden somit kombiniert. Die subjektiven Anforderungen an die Software hinsichtlich Benutzerfreundlichkeit und einfacher Bedienung müssen in weiterer Folge im Rahmen von Usabilitytests evaluiert werden. Ebenso muss die Effektivität der Therapie durch eine klinische Studie validiert werden.

5.4 Aufgetretene Probleme

Häufige Ursachen für Fehler waren vertauschte Signalleitungen und Programmierfehler im Arduino-Code.

Das in Kapitel 3.5.2 in Entwicklungsphase 4 aufgetretene Problem bezüglich der inkorrekten Spannung an den Output Pins des Mikrocontrollers ist vermutlich auf dessen innere Verschaltung zurückzuführen. Die verwendeten Pins (20, 21) können auch für einen i2c Bus verwendet werden, was die Theorie einer unvorteilhaften internen Verschaltung für den Verwendungszweck im Therapiegerät bekräftigt. Nach einem Wechsel dieser Pins auf die digitalen Pins 12 und 13 war dieses Problem vollständig behoben.

Die ursprünglich vorgesehene Platine zur Verbindung der Signalleitungen zwischen Mikrocontroller und Steuergeräten konnte aufgrund des aus der Covid-19 Pandemie im Frühjahr 2020 resultierenden Zeitmangels nicht mehr umgesetzt werden. Stattdessen wird das voll funktionsfähige Provisorium in Form einer Steckplatine weiterhin verwendet. Da alle Signalleitungen mit Kabelbindern gesichert sind, ist deren Lockerung nicht zu befürchten.

Die beiden im Rahmen diverser Funktionstests festgestellten verbleibenden Softwareanomalien werden ebenfalls als nicht sicherheitskritisch angesehen. Zum einen tritt selten bei der ersten Referenzierung der Hubsäulen nach der Inbetriebnahme des Therapiegeräts ein Fehler an einem zufälligen Steuergerät auf. Werden die Steuergeräte ausgelesen, wird entweder eine Spannungsunterversorgung oder eine Betätigung des Endlagenschalters angezeigt. Eine mögliche Erklärung wäre es, dass die Einschaltzeiten der Schaltnetzteile minimal von einander abweichen und das Steuergerät somit eine kurzzeitige Unterversorgung feststellt. Das Auslösen des Endlagenschalters könnte seinen Ursprung darin haben, dass sich die Spindeln langsam absenken, wenn die Hubsäulen über einen langen Zeitraum nicht bestromt werden. Da dieser Fehler durch dessen Quittierung behoben werden kann und dessen Ursache nicht relevant für Funktion und Sicherheit ist, kann diese Anomalie als unkritisch angesehen werden.

Die zweite Anomalie tritt in Folge eines ersten Fehlers und zwar durch die Unterbrechung der USB Verbindung zwischen Laptop und Mikrocontroller auf, während sich die Schrittmotoren im bestromten Zustand befinden. Der Mikrocontroller, dessen Spannungsversorgung nur über USB erfolgen sollte, bleibt so allerdings weiterhin betriebsbereit. Läuft zum Zeitpunkt der Unterbrechung ein Therapieprogramm wird es in weiterer Folge nicht gestoppt, sondern weiterhin ausgeführt. Ist zuvor eine Endlosschleife aktiviert worden, würde das Therapieprogramm für eine unbegrenzte Dauer weiterlaufen. Die Folgen dieser Anomalie lassen sich auf zwei Arten beseitigen. Kurzfristig kann der Not-Aus-Taster betätigt werden und das Therapiegerät somit deaktiviert werden. Auf jeden Fall aber muss USB Verbindung wiederhergestellt werden. Geschieht dies während des laufenden Therapieprogramms, wird der Mikrocontroller in dessen Start-Modus zurückversetzt und die Bewegungen der Hubsäulen stoppen umgehend. Die Ursache dieser Anomalie ist nach wie vor unklar. Vermutlich wird der Mikrocontroller durch die interne Verschaltung der I/O Pins, die laufend Signale in Form einer Spannung von den Steuergeräten erhalten (zum Beispiel durch das Bereitschaftssignal) weiterhin versorgt. Da diese Anomalie durch Sicherung des USB Kabels vorgebeugt werden kann und sie, sollte sie eintreten durch Betätigen des Not-Aus Tasters rasch und einfach eliminiert werden kann und von ihr keine Gefahr für den Patienten ausgeht, ist sie ebenfalls als „unkritisch“ zu bewerten. Bis auf diese beiden (auch in Kapitel 4.7 genannten) verbleibenden Anomalien konnten alle im Verlauf des Entwicklungsprozesses aufgetretenen Fehler und Probleme restlos behoben werden.

Der Vorteil einer Geräteentwicklung in einem kleinen Team, so wie sie hier stattgefunden hat, besteht ganz klar in den Überschneidungen des Knowhows der jeweiligen Schwerpunkte, was einer effizienten Problemlösung zugute kommt.

5.5 Getroffene Maßnahmen

Eine der sicherheitskritischsten Aufgaben besteht darin, das Therapieprogramm zuverlässig zu deaktivieren. Neben der Betätigung des Not-Aus-Tasters, welcher die Stromversorgung des gesamten Therapiegeräts unterbricht, gibt es auch die Möglichkeit das Programm über das GUI zu unterbrechen. Wird der Button „Sequenzen Stoppen / Fehler-Reset“ betätigt, so wird ein Stoppsignal an alle Steuergeräte gesendet, welches dafür sorgt, dass alle Hubsäulen sofort zum Stillstand kommen. Sollte sich entgegen der getroffenen Sicherheitsmaßnahmen (Aderendhülsen und Kabelbinder) doch eine Signalleitung lösen, wurde zur Abhilfe eine Redundanz in Form der Variable `start_trigger` programmiert, die gleichzeitig mit dem Aussenden des Stoppsignals deaktiviert wird. Diese Variable wird vor dem Start jeder Phase der Bewegungssequenzen abgefragt. Ist die Variable deaktiviert, so wird die Bewegungssequenz nicht fortgeführt. Zusätzlich wurden alle Bewegungssequenzen einzeln in die Main-Loop eingefügt, welche permanent durchlaufen wird, um eine Eingabe durch das GUI über den „Serial-Command-Handler“ mit einer geringen Latenz zu übernehmen.

Da die einzelnen Sequenzen im Prinzip alle gleich aufgebaut sind, wäre es auch möglich gewesen diese durch Schleifen zu implementieren. Dadurch ginge aber eben diese geringe Latenz verloren, da die Schleife zuerst beendet hätte werden müssen, bevor Eingaben über den „Serial-Command-Handler“ registriert worden wären. Somit werden alle Hubsäulen zuverlässig angehalten, auch wenn das Stoppsignal nicht zu allen Steuergeräten durchdringt.

Die von den Steuergeräten erzeugten Fehler und Warnungen sind mehr funktioneller Natur, als sicherheitsrelevant. (siehe Kapitel 4.3.8) Tritt ein Fehler auf, ist die Konsequenz daraus, dass die betreffende Hubsäule zu diesem Zeitpunkt nicht einsatzfähig ist und die Therapie somit nicht durchgeführt werden kann. Die Anzeige am GUI dient dazu, dem Anwender bzw. dem Betreiber zu zeigen, dass ein Fehler aufgetreten ist und wo er aufgetreten ist. Aus diesem Grund und um die Rechenkapazität des Mikrocontrollers nicht zu sehr zu beanspruchen wird die Funktion `errorHandler` nur alle 5 Sekunden ausgeführt.
Sollte ein Fehler angezeigt werden, kann der Anwender den Fehler quittieren und die Therapie starten, sofern die Ursache dafür behoben ist (z.B.: versehentliches Berühren des Endlagenschalters oder Spannungsschwankungen). Besteht der Fehler weiterhin, muss entsprechendes Personal den Fehler am Steuergerät auslesen und Maßnahmen setzen.

Die einzige SOUP in dem Softwaresystems des Therapiegerätes ist die Software der Steuergeräte „igus Dryve D1“. Um Gefahren zu reduzieren, die aufgrund einer Fehlfunktion dieser Software entstehen könnten, wurden mehrere Maßnahmen getroffen. Für den Ein-

satz im Therapiegerät werden hochwertige Steuergeräte verwendet, die sich schon mehrere Jahre am Markt (seit April 2017) bewährt haben und bei denen laut Korrespondenz mit dem Hersteller noch keine Fehlfunktionen beobachtet bzw. gemeldet wurden (Stand Jänner 2020). Die für diese Anwendung schwerwiegendste Fehlfunktion wäre ein unkontrolliertes Ausfahren der Hubsäulen. Um einer Verletzung des Patienten durch eine zu hohe Auslenkung entgegenzuwirken kann ein mechanischer Anschlag an der Hubsäule angebracht werden. Bei zu hohen Geschwindigkeiten und Beschleunigungen tritt beim Spindelantrieb der Säulen eine Selbsthemmung auf und der Schrittmotor dreht lediglich durch.

5.6 Verbesserungen und weitere Schritte

Da es sich bei diesem Therapiegerät um einen völlig neuen Therapieansatz handelt, muss in Zukunft im Rahmen einer klinischen Prüfung belegt werden, ob die mit dem Therapiegerät erzeugten Bewegungsmuster tatsächlich die erwünschte Wirkung bzw. Therapieerfolg erzielen. Zuvor müssen noch einige Konstruktive Maßnahmen getroffen werden, wie die Montage einer Laptophalterung an das Gerät bzw. die Konstruktion einer Führung für das USB-Kabel und Netzkabel des Notebooks aus dem Gerät. Des Weiteren ist die bereits entworfene Platine wie vorgesehen zu fertigen und anstatt der Steckplatine in das Gerät zu integrieren. Die Software ist so ausgelegt, das die bestehenden Bewegungssequenzen ohne großen Aufwand modifiziert und erweitert werden können. Eine Anleitung hierfür befindet sich in der technischen Dokumentation.

Für die Wartung der Software wurden keine weiteren Vorkehrungen getroffen.

6 Schlussfolgerung

Im Laufe dieser Arbeit wurden Software und Ansteuerung für ein Therapiegerät zur Behandlung neurologischer Defizite bei Neugeborenen durch die Simulation physiologischer Bewegungsmuster entwickelt. Es wurde ein Ansteuerungskonzept gewählt, welches am besten umzusetzen war und gleichzeitig den Anforderungen am ehesten gerecht wurde.

Um das Gerät so seriennahe wie möglich zu entwickeln, wurden von Anfang an die entsprechenden Normen EN 62304 und EN 60601-1 und aus deren Konsequenz ebenso die Norm EN 14971 berücksichtigt. Im Rahmen des Risikomanagements konnten insgesamt 105 Risiken identifiziert, davon 15 rein softwareseitig, und auf ein akzeptables Niveau reduziert werden. Das Nutzen Risiko-Verhältnis der SOUP der Steuergeräte „igus Dryve D1“ konnte durch entsprechende Maßnahmen auf ein akzeptables Niveau gebracht werden.

Bei der Entwicklung von Ansteuerung und Software stand die höchst mögliche Sicherheit bei optimaler Funktionalität im Fokus. Zusätzlich wurde Wert auf ein einfaches, intuitives Konzept gelegt. Das Ergebnis ist eine Ansteuerungssoftware die acht Hubsäulen steuert und somit realitätsnahe Bewegungsmuster simuliert, mit deren Hilfe mehrere Körperpartien des Patienten gezielt und sicher in Bewegung gesetzt werden können. Die aus bis zu 20 unterschiedlichen, vorgegebenen Bewegungssequenzen bestehenden Therapieprogramme können dauerhaft abgespeichert und einem pseudonymisierten Patienten zugewiesen werden. Die Bedienung geschieht durch eine Benutzeroberfläche auf einem Notebook, welche außerdem die aktive Bewegungssequenz des Therapieprogramms anzeigt.

Bereits während der Softwareentwicklung wurden nach der Implementierung jeder Änderung und am Ende jeder Entwicklungsphase ausführliche Funktionstests durchgeführt (Komponententests und Integrationstests, Abbildung 3.5 und 3.6). Nach Abschluss der Geräte- und Softwareentwicklung wurden Integrationstests durchgeführt und entsprechende Dokumente nach EN 62304 erstellt.

Im Rahmen der hervorragenden Zusammenarbeit konnte im Zuge des Projektes ein Gerät entwickelt werden, das alle Erwartungen seitens der Klinik, des Instituts für Health Care Engineering und nicht zuletzt die eigenen erfüllt, wenn nicht sogar etwas übertroffen hat.

Literaturverzeichnis

[1] EN ISO 14971:2013 Medizinprodukte - Anwendung des Risikomanagements auf Medizinprodukte. 2013.

[2] OVE EN 62304:2016 Medizingeräte-Software - Software-Lebenszyklus-Prozesse. 2016.

[3] Arduino AG. Arduino Mega 2560 Rev3. https://store.arduino.cc/arduino-mega-2560-rev3, 2020.

[4] Tobias Bergerhoff. General movements - Eine funktionelle Diagnostik des jungen Nervensystems. 2008.

[5] Christa Einspieler, Peter B. Marschik, and Heinz F.R. Prechtl. Was erzählen uns frühkindliche Bewegungen? 2010.

[6] igus. D1 Handbuch V2.0. https://technical-documents.igus.com/dryve.php, 2019.

[7] igus. dryve Motorsteuerung. https://www.igus.at/info/drive-technology-dryve-motor-control-system, 2019.

[8] Norbert Leitgeb. Sicherheit von Medizingeräten. Springer Verlag. ISBN: 978-3-662-44656-0, 2015.

[9] Number Eight Innovation Limited. Megunolink. https://www.megunolink.com, 2020.

[10] Thomas Mauerhofer. Entwicklung eines Therapiegerätes zur Unterstützung der frühkindlichen Spontanmotorik bei Säuglingen. Master's thesis, Technische Universität Graz - Institut für Health Care Engineering, 2019.

[11] Dominik Steindorfer. Entwicklung und Aufbau eines Therapiegerätes für Säuglinge. Master's thesis, Technische Universität Graz - Institut für Health Care Engineering, 2020.

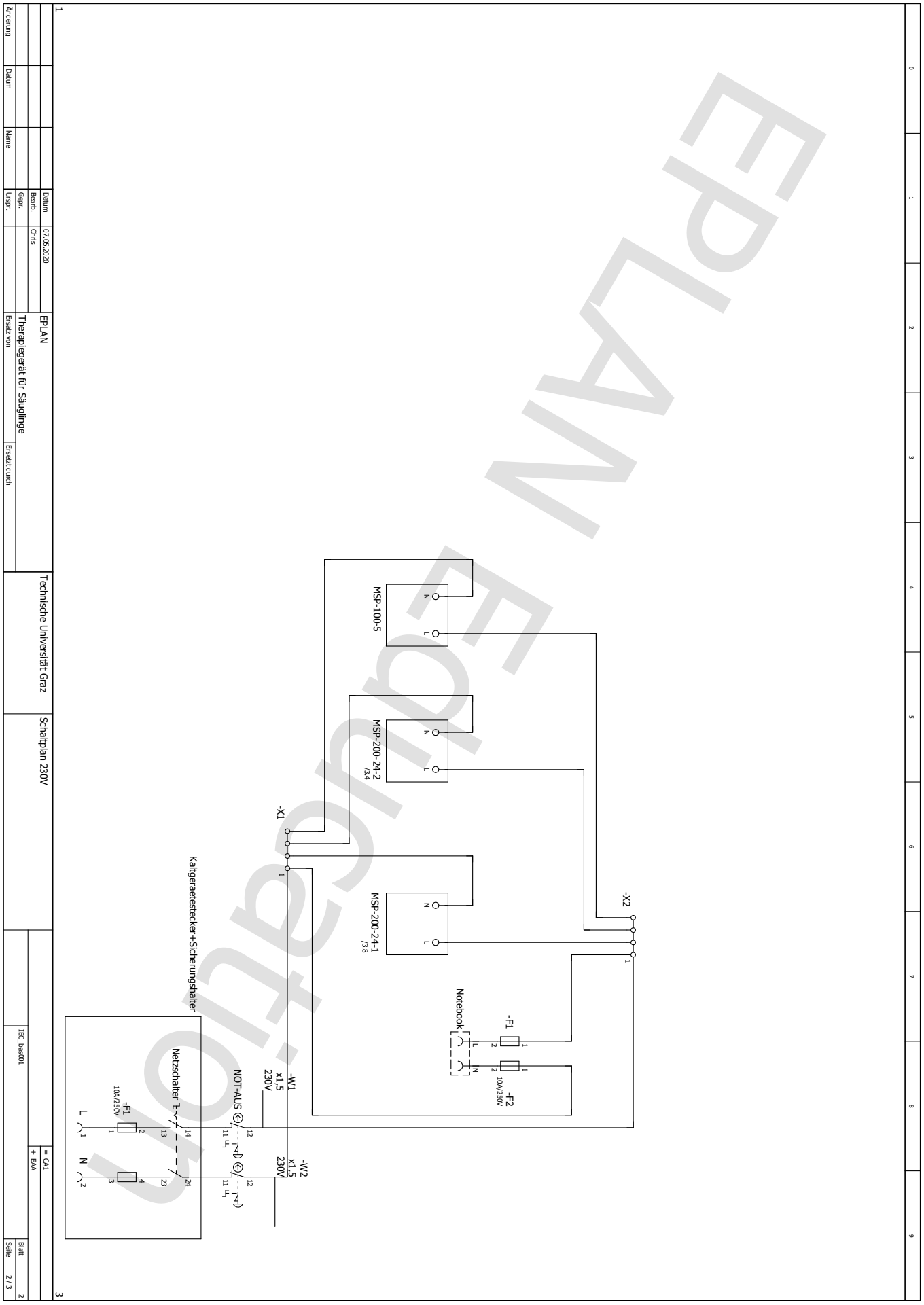
[12] Mean Well. MSP-100-5. https://www.meanwell-web.com/en-gb/ac-dc-single-output-medical-enclosed-power-supply-msp--100--5, 2020.

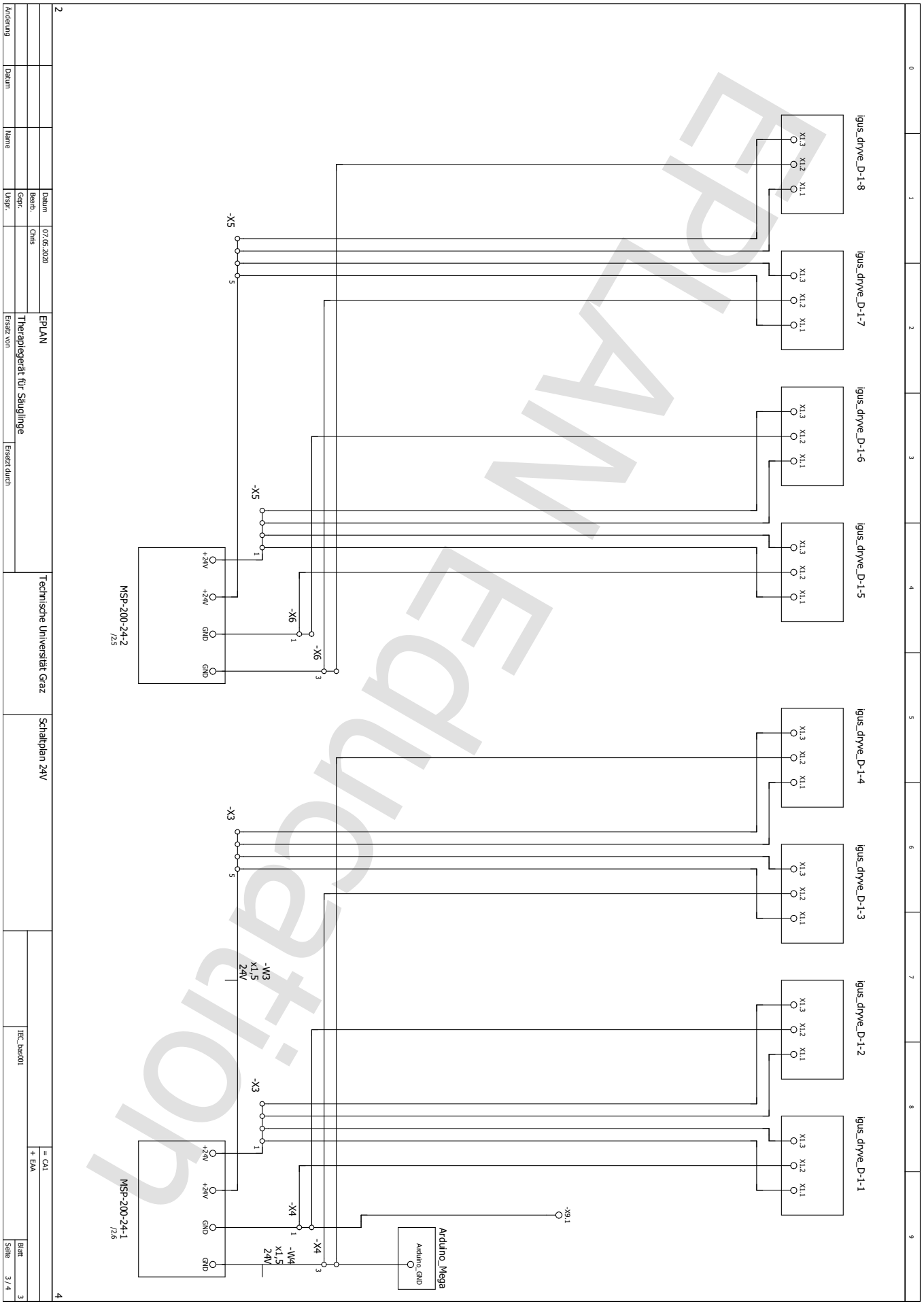
[13] Mean Well. MSP-200-24. https://www.meanwell-web.com/en-gb/ac-dc-single-output-medical-enclosed-power-supply-msp--200--24, 2020.

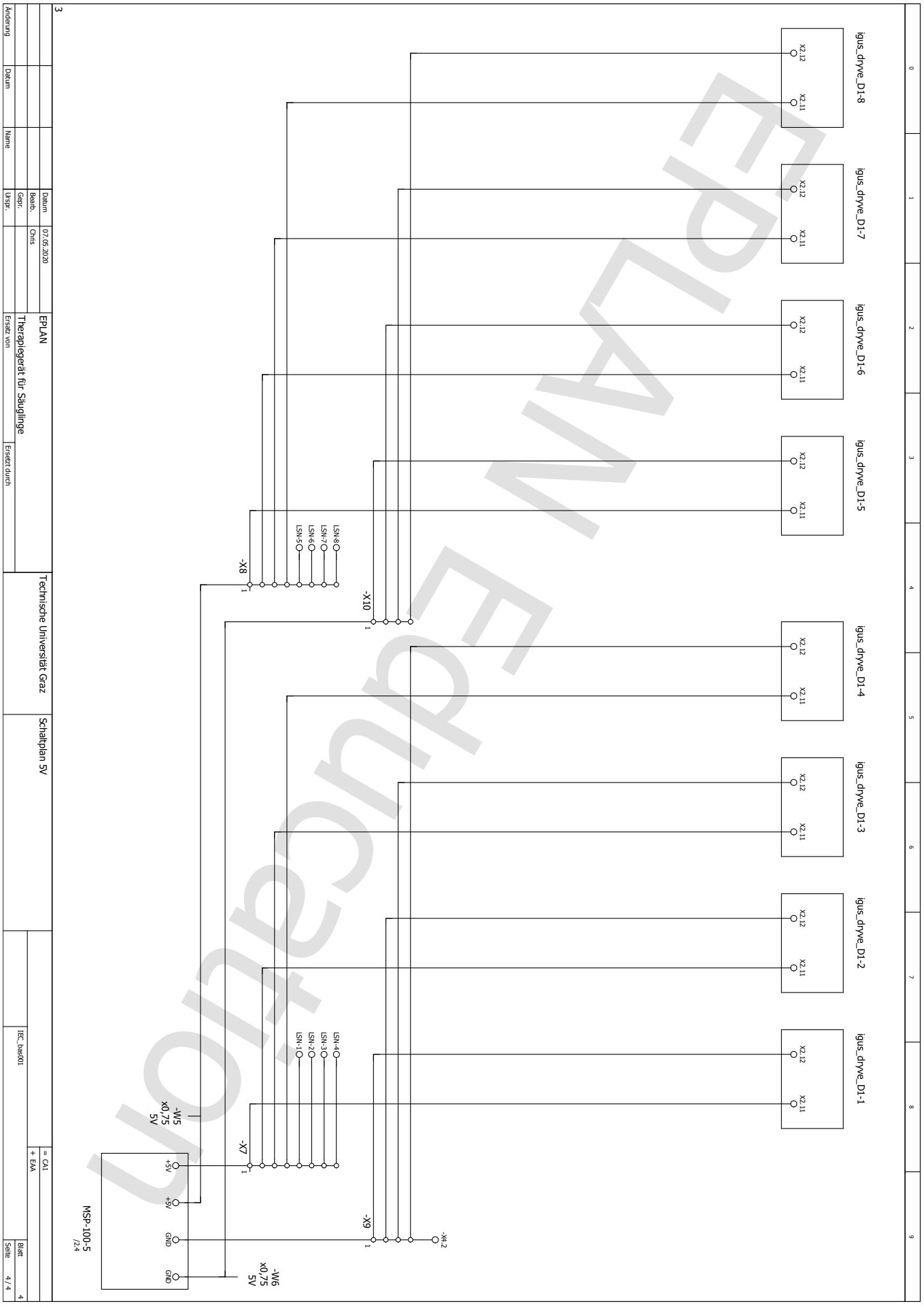
Abbildungsverzeichnis

Tabellenverzeichnis

7 Anhang


EPLAN
MSP-100-5
MSP-200-24-2
/3.4
MSP-200-24-1
/3.8
-X1
-X2
Notebook
-F1
-F2
10A/250V
-W1
x1,5
230V
-W2
x1,5
230V
NOT-AUS
Kaltgeraetestecker+Sicherungshalter
Netzschalter
-F1
10A/250V
L
N
EPLAN
Technische Universität Graz
Schaltplan 230V
Therapiegerät für Säuglinge
Datum 07.05.2020
Bearb. Chris
Gepr.
Urspr.
Name
Datum
Änderung
Ersatz von
Ersetzt durch
IEC_bas001
= CA1
+ EAA
Blatt 2
Seite 2 / 3


EPLAN Education
igus_dryve_D-1-8
igus_dryve_D-1-7
igus_dryve_D-1-6
igus_dryve_D-1-5
igus_dryve_D-1-4
igus_dryve_D-1-3
igus_dryve_D-1-2
igus_dryve_D-1-1
X1.3
X1.2
X1.1
-X5
-X6
-X3
-X4
-W3
x1,5
24V
-W4
MSP-200-24-2
/2.5
MSP-200-24-1
/2.6
+24V
GND
-X9.1
Arduino_Mega
Arduino_GND
EPLAN
Technische Universität Graz
Therapiegerät für Säuglinge
Schaltplan 24V
Datum
07.05.2020
Bearb.
Chris
Gepr.
Urspr.
Ersatz von
Ersetzt durch
Name
Änderung
IEC_bas001
= CA1
+ EAA
Blatt
3
Seite
3 / 4


igus_dryve_D1-8
igus_dryve_D1-7
igus_dryve_D1-6
igus_dryve_D1-5
igus_dryve_D1-4
igus_dryve_D1-3
igus_dryve_D1-2
igus_dryve_D1-1
X2.12
X2.11
-X8
-X10
-X9
-X7
LSN-8
LSN-7
LSN-6
LSN-5
LSN-4
LSN-3
LSN-2
LSN-1
-W5
x0,75
5V
-W6
x0,75
5V
MSP-100-5
/2.4
+5V
GND
-X6.2
EPLAN
Technische Universität Graz
Schaltplan 5V
Therapiegerät für Säuglinge
IEC_bas001
= CA1
+ EAA
Blatt 4
Seite 4 / 4
07.05.2020
Chris
EPLAN Education

Anhang B Arduino Pin-Belegung

Arduino Pin	Steuergerät Nr.	Steuergerät Pin	Bezeichnung
Ausgänge			
3	1-8	X2.1	Bit 0
4	1-8	X2.2	Bit 1
5	1-8	X2.3	Bit 2
6	1-8	X2.4	Bit 3
7	1-8	X2.5	Bit 4
8	1-8	X2.7	Freigabe /Bestromung
22	1	X2.6	Start 1
24	2	X2.6	Start 2
26	3	X2.6	Start 3
28	4	X2.6	Start 4
30	5	X2.6	Start 5
32	6	X2.6	Start 6
34	7	X2.6	Start 7
36	8	X2.6	Start 8
23	1	X2.10	Stop/Reset 1
25	2	X2.10	Stop/Reset 2
27	3	X2.10	Stop/Reset 3
29	4	X2.10	Stop/Reset 4
31	5	X2.10	Stop/Reset 5
33	6	X2.10	Stop/Reset 6
35	7	X2.10	Stop/Reset 7
37	8	X2.10	Stop/Reset 8

Eingänge			
14	1	X3.1	Bereit 1
15	2	X3.1	Bereit 2
16	3	X3.1	Bereit 3
17	4	X3.1	Bereit 4
18	5	X3.1	Bereit 5
19	6	X3.1	Bereit 6
12	7	X3.1	Bereit 7
13	8	X3.1	Bereit 8
10	1	X3.5	Error 1
40	2	X3.5	Error 2
42	3	X3.5	Error 3
44	4	X3.5	Error 4
46	5	X3.5	Error 5
48	6	X3.5	Error 6
50	7	X3.5	Error 7
52	8	X3.5	Error 8
11	1	X3.4	Alert 1
41	2	X3.4	Alert 2

Anhang B Arduino Pin-Belegung

43	3	X3.4	Alert 3
45	4	X3.4	Alert 4
47	5	X3.4	Alert 5
49	6	X3.4	Alert 6
51	7	X3.4	Alert 7
53	8	X3.4	Alert 8

Anhang C Inhalt der Software Anforderungen
OVE EN 62304-2016, 5.2.2

Anforderungen an die Funktionalität und die Leistungsfähigkeit

Leistungsmerkmale

- Zweck der Software:
  Diese Software dient der Ansteuerung des Therapiegeräts für Säuglinge. Schrittmotoren (NEMA23) werden hierbei über Steuergeräte der Type „igus Dryve D1“ angesteuert. Über das GUI „MegunoLink“ werden Benutzereingaben getätigt und Rückmeldungen über den Status von Therapie und Gerät durchgeführt.
- Anforderungen bezüglich Zeitverhalten:
  Ein Programmabbruch über das GUI soll unverzüglich geschehen. Die Software soll so beschaffen sein, dass die Bewegungssequenzen möglichst flüssig und physiologisch ablaufen. Nicht kritische Fehlermeldungen und Warnungen sollten spätestens nach wenigen Sekunden am GUI erscheinen.

Physikalische Merkmale

- Programmiersprache:
  Keine besondere Anforderung
- Plattform:
  Die Software soll auf einem handelsüblichen, dem Stand der Technik (2019) entsprechenden Notebook ausgeführt werden können
- Betriebssystem:
  Die Software soll auf dem Betriebssystem „Windows 10“ lauffähig sein

Rechnerumgebung

- Bezüglich des Rechners gibt es keine gesonderten Anforderungen, sofern der Rechner einigermaßen dem Stand der Technik 2019 (8GB RAM, Mehrkern Prozessor) entspricht.

Ein- und Ausgaben des Softwaresystems

Merkmale von Daten

Nicht zutreffend, da kein Einlesen/Ausgabe von externen Daten in die Software vorgesehen ist

Bereiche, Grenzen, Werte der Voreinstellungen

Ebenfalls nicht zutreffend, da der Benutzer am GUI nur aus einem vorgegebenen Pool an Werten wählen kann.

Schnittstellen zwischen dem Softwaresystem und anderen Systemen

Schnittstellen zu anderen Systemen sind nicht vorgesehen.

Schnittstellen innerhalb des Systems

- Notebook-Arduino Mega-Microcontroller: USB 2.0
- Arduino Mega – igus Dryve D1 Steuergerät: Mehradrige Signalleitungen
- Vorprogrammierung und Auslesen von Fehlern am Steuergerät (Notebook - DryveD1) : Standard Belegung nach TIA-568A und TIA-568B

Software gesteuerte Alarme, Warnungen und Anwender-Meldungen

- Am Steuergerät auftretende Warnungen, Fehler sollen am GUI angezeigt werden.
- Rückmeldungen bezüglich des Therapieprogrammablaufs, der Bedienung, etc. sollen am GUI angezeigt werden
- Alle Anzeigen und Rückmeldung sind rein optisch, es sind keine akustischen Alarme vorgesehen

Anhang C Inhalt der Software Anforderungen
OVE EN 62304-2016, 5.2.2

Anforderungen an die Datensicherheit

Anforderungen im Zusammenhang mit der Beeinträchtigung vertraulicher Informationen

Da Patienten nur pseudonymisiert gespeichert werden ist gibt es hier keine Anforderungen.

Autorisierung

Eine Autorisierung via Passwort ist später vorgesehen um die Bewegungsparameter im Steuergerät zu ändern.

Authentifizierung, Systemsicherheit, Schutz gegen Schadsoftware

Da das System für einen Offline Betrieb vorgesehen ist gibt es hier keine Anforderungen

Anforderungen an die Benutzerschnittstelle, die durch Software implementiert ist

Unterstützung für manuelle Tätigkeiten

Ist nicht vorgesehen

Interaktion zwischen Mensch und Gerät

Sowohl die Eingabe, also die Auswahl von Bewegungssequenzen, Starten, Stoppen von Therapieprogrammen, etc. als auch die Ausgabe des Status der Software, Motoren, Meldungen, Warnhinweise, etc. finden ausschließlich über das GUI statt.

Einschränkungen für das Personal

- Anwender: Nutzung des Therapiegeräts via GUI
- Techniker, Wartung, besonders geschultes Personal: Erweiterter Zugriff mittels Passwort auf die Steuergeräte

Bereiche, die konzentrierte menschliche Aufmerksamkeit erfordern

Während der gesamten Anwendung ist die Aufmerksamkeit des Anwenders erforderlich. Ein Entsprechender Warnhinweis wird beim Start der Software ausgegeben

Datendefinition und Datenbank-Anforderungen

Da keinerlei Datenbanken verwendet werden: Nicht zutreffend

Anforderungen für die Installation an Standorten der und Abnahme und Wartung

Da sich die Software in jedem dieser Fälle auf der zugehörigen Hardware befindet, gibt es hierfür keine gesonderten Anforderungen

Anforderungen hinsichtlich Methoden des Betriebs und der Wartung

Betrieb und Wartung der Software/des Therapiegeräts sind nur von geschultem und qualifiziertem Fachpersonal durchzuführen

Anforderungen bezüglich IT-Netzwerk Aspekten

Da der Anschluss des Therapiegeräts /der Steuerungssoftware an ein IT-Netzwerk nicht vorgesehen ist, gibt es keine Anforderungen/nicht zutreffend.

Anforderungen hinsichtlich der Wartung durch den Betreiber

Hinsichtlich der Wartung gibt es keine Anforderungen. Die Software soll allerdings ohne größeren Aufwand erweiterbar sein um ggf. neue Bewegungssequenzen bzw. Therapien implementieren zu können.

Anhang D

Verifizierung und Validierung:
Checklisten - Übersicht

CL - Nr.	Bezeichnung	Datum	Status	Anmerkung
1	**Parameter**			
1.1	Dryve D1 - Nr. 1	05.05.2020	Bestanden	
1.2	Dryve D1 - Nr. 2	05.05.2020	Bestanden	
1.3	Dryve D1 - Nr. 3	05.05.2020	Bestanden	
1.4	Dryve D1 - Nr. 4	05.05.2020	Bestanden	
1.5	Dryve D1 - Nr. 5	05.05.2020	Bestanden	
1.6	Dryve D1 - Nr. 6	05.05.2020	Bestanden	
1.7	Dryve D1 - Nr. 7	05.05.2020	Bestanden	
1.8	Dryve D1 - Nr. 8	05.05.2020	Bestanden	
2	**Fahrprofile**			
2.1	Fahrprofile- Dryve D1 - Nr. 1	05.05.2020	Bestanden	
2.2	Fahrprofile- Dryve D1 - Nr. 2	05.05.2020	Bestanden	
2.3	Fahrprofile- Dryve D1 - Nr. 3	05.05.2020	Bestanden	
2.4	Fahrprofile- Dryve D1 - Nr. 4	05.05.2020	Bestanden	
2.5	Fahrprofile- Dryve D1 - Nr. 5	05.05.2020	Bestanden	
2.6	Fahrprofile- Dryve D1 - Nr. 6	05.05.2020	Bestanden	
2.7	Fahrprofile- Dryve D1 - Nr. 7	05.05.2020	Bestanden	
2.8	Fahrprofile- Dryve D1 - Nr. 8	05.05.2020	Bestanden	

Anhang D

Verifizierung und Validierung:
Checklisten - Beispiel

Checkliste **CL-1.1** **Steuergerät: igus Dryve D1 - Nr. 1** **V1.0**

Test Autor:	Christian Krammer	**Datum:**	05.05.2020
Freigabe:	Christian Krammer	**Datum:**	05.05.2020
Tester:	Christian Krammer	**Datum:**	05.05.2020
Firmware:	dryve-D1-1-20190226		
Konfigurationsdatei	Full_Setup_V1		

Ergebnis:

Kurzfassung des Tests:
Checkliste der Parameter der Steuergeräte

Nr.	Parameter	Wert	Ja/Nein	Anmerkung
CL-1.1.1	Motortyp	ST (Schrittmotor)	JA	
CL-1.1.2	Motorstrom/Booststrom	3,2A	JA	
CL-1.1.3	Haltestrom	1,6A	JA	
CL-1.1.4	Schrittmodus	Auto	JA	
CL-1.1.5	Schrittwinkel	1,8°	JA	
CL-1.1.6	Verfügbarer Hub	60mm	JA	
CL-1.1.7	Vorschub	12mm	JA	
CL-1.1.8	Max. Geschwindigkeit	100 mm/s	JA	
CL-1.1.9	Max. Beschleunigung	150 mm/s²	JA	
CL-1.1.10	S-Curve	25%	JA	
CL-1.1.11	Schleppfehler	5mm	JA	
CL-1.1.12	Endlagenschalter	Negativ	JA	
CL-1.1.13	Referenzierungsmethode	LSN	JA	
CL-1.1.14	Offset	-5mm	JA	
CL-1.1.15	Manuelle IP	192.168.0.10	JA	
CL-1.1.16	Subnetzmaske	255.255.255.0	JA	
CL-1.1.17	Standardgateway	192.168.0.1	JA	
CL-1.1.18	Alle Digitalen Ein/Ausgänge	"H" High	JA	

Anhang D

Verifizierung und Validierung:
Checklisten - Beispiel

Checkliste **CL-2.1** **Fahrprofile: igus Dryve D1 - Nr. 1** **V1.0**

Test Autor:	Christian Krammer	**Datum:**	05.05.2020
Freigabe:	Christian Krammer	**Datum:**	05.05.2020
Tester:	Christian Krammer	**Datum:**	05.05.2020
Firmware:	dryve-D1-1-20190226		
Konfigurationsdatei:	Full_Setup_V1		
Betriebsart:	Binär		

Ergebnis:

Kurzfassung des Tests:
Checkliste der Fahrprofile der Steuergeräte

CL Nr.	Profil - Nr.	Modus	Ziel	Beschleunigung	Geschwindigkeit	Verzögerung	Ja/Nein
			(mm)	(mm/s²)	(mm/s)	(mm/s²)	
CL-2.1.1	1	ABS	0	30	30	100	JA
CL-2.1.2	2	ABS	0	40	40	40	JA
CL-2.1.3	3	ABS	0	60	60	60	JA
CL-2.1.4	4	ABS	0	100	100	100	JA
CL-2.1.5	12	ABS	40	40	40	40	JA
CL-2.1.6	13	ABS	30	60	60	60	JA
CL-2.1.7	14	ABS	40	100	100	100	JA
CL-2.1.8	15	ABS	50	40	40	40	JA
CL-2.1.9	16	ABS	60	40	40	40	JA
CL-2.1.10	32	HOM	LSN	10	20	150	JA

Anhang E

Verifizierung und Validierung:
Testprotokolle - Übersicht

TP - Nr.	Bezeichnung	Datum	Status	Anmerkung
1	**Inbetriebnahme**			
1.1	Inbetriebnahme	04.05.2020	Bestanden	
1.2	Programmstart	04.05.2020	Bestanden	
1.3	Referenzieren	04.05.2020	Bestanden	
1.4	Testsequenz	04.05.2020	Bestanden	
2	**Sequenzen und Anzeige**			
2.1	Hinzufügen der korrekten Sequenzen (Single)	04.05.2020	Bestanden	
2.2	Hinzufügen der korrekten Sequenzen (Dual)	04.05.2020	Bestanden	
2.3	Löschen von Bewegungssequenzen	04.05.2020	Bestanden	
2.4	Anzeige der aktiven Sequenzen	04.05.2020	Bestanden	
2.5	Rücksetzung der Anzeige durch Referenzierung	04.05.2020	Bestanden	
3	**Patientenmanagement**			
3.1	Speichern und Laden von Patienten	04.05.2020	Bestanden	
3.2	Erweiterung eines geladenen Patienten	04.05.2020	Bestanden	
3.3	Löschen eines angelegten Patienten	04.05.2020	Bestanden	
4	**Bewegungsprogrammablauf**			
4.1	Start und Ende des Therapieprogramms	04.05.2020	Bestanden	
4.2	Stoppen des Programms durch den Benutzer	04.05.2020	Bestanden	
4.3	Fortsetzung des Programms	04.05.2020	Bestanden	
4.4	Endlosschleife	04.05.2020	Bestanden	
5	**Bewegungssequenzen**			
5.1	Single 1 (4cm)	05.05.2020	Bestanden	
5.2	Single 1 (6cm)	05.05.2020	Bestanden	
5.3	Single 2 (4cm)	05.05.2020	Bestanden	
5.4	Single 2 (6cm)	05.05.2020	Bestanden	
5.5	Single 3 (4cm)	05.05.2020	Bestanden	
5.6	Single 3 (6cm)	05.05.2020	Bestanden	
5.7	Single 4 (4cm)	05.05.2020	Bestanden	
5.8	Single 4 (6cm)	05.05.2020	Bestanden	
5.9	Single 5 (4cm)	05.05.2020	Bestanden	
5.10	Single 4 (6cm)	05.05.2020	Bestanden	
5.11	Dual 1 (4cm)	05.05.2020	Bestanden	
5.12	Dual 1 (6cm)	05.05.2020	Bestanden	
5.13	Dual 2 (4cm)	05.05.2020	Bestanden	
5.14	Dual 2 (6cm)	05.05.2020	Bestanden	
5.15	Dual 3 (4cm)	05.05.2020	Bestanden	
5.16	Dual 3 (6cm)	05.05.2020	Bestanden	
5.17	Dual 4 (4cm)	05.05.2020	Bestanden	
5.18	Dual 4 (6cm)	05.05.2020	Bestanden	
5.19	Dual 5 (4cm)	05.05.2020	Bestanden	
5.20	Dual 5 (6cm)	05.05.2020	Bestanden	
5.21	Dual 6 (4cm)	05.05.2020	Bestanden	
5.22	Dual 6 (6cm)	05.05.2020	Bestanden	
5.23	Test (Alle)	05.05.2020	Bestanden	

Anhang E

Verifizierung und Validierung:
Testprotokolle - Übersicht

5.24	Wartung	05.05.2020	Bestanden	
6	**Fehler, Alarme und Abbrüche**			
6.1	Fehlersimulation Hubsäule 1	05.05.2020	Bestanden	
6.2	Fehlersimulation Hubsäule 2	05.05.2020	Bestanden	
6.3	Fehlersimulation Hubsäule 3	05.05.2020	Bestanden	
6.4	Fehlersimulation Hubsäule 4	05.05.2020	Bestanden	
6.5	Fehlersimulation Hubsäule 5	05.05.2020	Bestanden	
6.6	Fehlersimulation Hubsäule 6	05.05.2020	Bestanden	
6.7	Fehlersimulation Hubsäule 7	05.05.2020	Bestanden	
6.8	Fehlersimulation Hubsäule 8	05.05.2020	Bestanden	
6.11	Alarmsimulation Hubsäule 1	05.05.2020	Bestanden	
6.12	Alarmsimulation Hubsäule 2	05.05.2020	Bestanden	
6.13	Alarmsimulation Hubsäule 3	05.05.2020	Bestanden	
6.14	Alarmsimulation Hubsäule 4	05.05.2020	Bestanden	
6.15	Alarmsimulation Hubsäule 5	05.05.2020	Bestanden	
6.16	Alarmsimulation Hubsäule 6	05.05.2020	Bestanden	
6.17	Alarmsimulation Hubsäule 7	05.05.2020	Bestanden	
6.18	Alarmsimulation Hubsäule 8	05.05.2020	Bestanden	
6.20	Spannungsausfall und Wiederkehr	05.05.2020	Bestanden	

Anhang E

Verifizierung und Validierung:
Testprotokolle - Beispiel

Testprotokoll: **TP-1.4** **Testsequenz** **V1.0**

Test Autor:	Christian Krammer	**Datum:**	30.04.2020
Freigabe:	Christian Krammer	**Datum:**	04.05.2020
Tester:	Christian Krammer	**Datum:**	04.05.2020

Testergebnis Bestanden

Kurzfassung des Tests:
Nach der Referenzierung wird mit der Testsequenz die Funktionalität aller Hubsäulen überprüft

Nr.	Eingabe	Erwartetes Ergebnis	Ja/Nein	Anmerkung
FT-1.4.1	Aus der Liste "Sequenzwahl" (unten) die Sequenz "Test: Alle gleichzeitig" auswählen	Blaue hinterlegung der Auswahl	JA	
FT-1.4.2	Button "Sequenz hinzufügen" betätigen	"Funktionstest" erscheint im obersten linken Feld unter "Bewegungssequenzen"	JA	
FT-1.4.3	Button "Sequenzen starten" betätigen	Alle 8 Hubsäulen bewegen sich synchron einmal auf - und ab	JA	
		Die Hubsäulen bei der Rückenpartie fahren auf 4cm aus	JA	
		Alle weiteren 6 Hubsäulen fahren auf 6cm Hub aus	JA	
		Das Textfeld "Funktionstest" wird grün hinterlegt	JA	
		Alle Säulen kommen im eingefahrenen Zustand zum Stillstand	JA	
FT-1.4.4	Button "Sequenzen Löschen" betätigen	Der Inhalt aller Textfelder "Bewegungssequenzen" ist gelöscht und weiß hinterlegt	JA	

Anhang E

Verifizierung und Validierung:
Testprotokolle - Beispiel

Testprotokoll: **TP-5.1** **Single 1 (4cm)** **V1.0**

Test Autor:	Christian Krammer	**Datum:**	04.05.2020
Freigabe:	Christian Krammer	**Datum:**	04.05.2020
Tester:	Christian Krammer	**Datum:**	05.05.2020

Testergebnis Bestanden

Kurzfassung des Tests:
Bei diesem Test wird der korrekte Ablauf der Bewegungssequenz "Single1 (4cm)" überprüft.
Ebenso muss die maximale Auslenkung der Hubsäulen geprüft werden.
Der Test kann nach der Vorlage von TP-4.1 mit der entsprechenden Bewegungssequenz durchgeführt werden.

Nr.	**Eingabe/Ablauf**		**Erwartetes Ergebnis**	**Ja/Nein**	**Anmerkung**
FT-5.1.1	Start der Bewegungssequenz "Single1 (4cm)"		Ablauf der Bewegungssequenz:		
Bewegung Nr.	Partie	Säule Nr.			
1	Schulter Rechts	1		JA	
2	Schulter Links	2		JA	
3	Rücken Rechts	3		JA	
4	Rücken Links	4		JA	
5	Gesäß Rechts	5		JA	
6	Gesäß Links	6		JA	
7	Bein Rechts	7		JA	
8	Bein Links	8		JA	
			Gesamter Ablauf laut Vorgabe:	JA	
FT-5.1.2			Maximale Auslenkung aller Hubsäulen muss 4cm betragen	JA	
FT-5.1.3			Alle Säulen kommen im eingefahrenen Zustand zum	JA	

Anhang E

Verifizierung und Validierung:
Testprotokolle - Beispiel

Testprotokoll: **TP-5.12** **Dual 1 (6cm)** **V1.0**

Test Autor:	Christian Krammer	**Datum:**	04.05.2020
Freigabe:	Christian Krammer	**Datum:**	04.05.2020
Tester:	Christian Krammer	**Datum:**	05.05.2020

Testergebnis Bestanden

Kurzfassung des Tests:
Bei diesem Test wird der korrekte Ablauf der Bewegungssequenz "Dual1 (6cm)" überprüft.
Ebenso muss die maximale Auslenkung der Hubsäulen geprüft werden.
Der Test kann nach der Vorlage von TP-4.1 mit der entsprechenden Bewegungssequenz durchgeführt werden.

Nr.	**Eingabe/Ablauf**		**Erwartetes Ergebnis**		**Ja/Nein**	**Anmerkung**
FT-5.12.1	Start der Bewegungssequenz "Dual1 (6cm)"		Ablauf der Bewegungssequenz:			
Bewegung Nr.	Partie 1	Säule Nr.	Partie 2	Säule Nr.		
1	Schulter Links	2	Gesäß Rechts	5	JA	
2	Gesäß Links	6	Schulter Rechts	1	JA	
3	Bein Links	8	Bein Rechts	7	JA	
4	Gesäß Links	6	Gesäß Rechts	5	JA	
5	Schulter Links	2	Gesäß Rechts	5	JA	
6	Gesäß Links	6	Schulter Rechts	1	JA	
7	Bein Links	8	Bein Rechts	7	JA	
8	Gesäß Links	6	Gesäß Rechts	5	JA	
			Gesamter Ablauf laut Vorgabe:		JA	
FT-5.12.2			Maximale Auslenkung aller Hubsäulen mit Ausnahme der Rückenpartie (3 & 4) muss 6cm betragen		JA	
FT-5.12.3			Maximale Auslenkung der Hubsäulen am Rücken (3 & 4) darf nicht höher, als 4cm sein		JA	
FT-5.12.4			Alle Säulen kommen im eingefahrenen Zustand zum		JA	

Anhang F Übersicht aller Bewegungssequenzen

Single 01		
Bewegung	Partie	Nr.
1	Schulter Rechts	1
2	Schulter Links	2
3	Rücken Rechts	3
4	Rücken Links	4
5	Gesäß Rechts	5
6	Gesäß Links	6
7	Bein Rechts	7
8	Bein Links	8

Single 02		
Bewegung	Säule	Nr.
1	Schulter Links	2
2	Rücken Links	4
3	Gesäß Links	6
4	Bein Links	8
5	Schulter Rechts	1
6	Rücken Rechts	3
7	Gesäß Rechts	5
8	Bein Rechts	7

Single 03		
Bewegung	Säule	Nr.
1	Schulter Links	2
2	Schulter Rechts	1
3	Schulter Links	2
4	Schulter Rechts	1
5	Gesäß Links	6
6	Gesäß Rechts	5
7	Gesäß Links	6
8	Gesäß Rechts	5

Single 04		
Bewegung	Säule	Nr.
1	Rücken Links	4
2	Rücken Rechts	3
3	Rücken Links	4
4	Rücken Rechts	3
5	Bein Links	8
6	Bein Rechts	7
7	Bein Links	8
8	Bein Rechts	7

Single 05		
Bewegung	Säule	Nr.
1	Schulter Links	2
2	Gesäß Rechts	5
3	Schulter Rechts	1
4	Gesäß Links	6
5	Rücken Rechts	3
6	Bein Links	8
7	Rücken Links	4
8	Bein Rechts	7

Anhang F

Übersicht aller Bewegungssequenzen

Dual 01				
Bewegung	Partie 1	Nr.	Partie 2	Nr.
1	Schulter Links	2	Gesäß Rechts	5
2	Gesäß Links	6	Schulter Rechts	1
3	Bein Links	8	Bein Rechts	7
4	Gesäß Links	6	Gesäß Rechts	5
5	Schulter Links	2	Gesäß Rechts	5
6	Gesäß Links	6	Schulter Rechts	1
7	Bein Links	8	Bein Rechts	7
8	Gesäß Links	6	Gesäß Rechts	5

Dual 02				
Bewegung	Partie 1	Nr.	Partie 2	Nr.
1	Schulter Links	2	Gesäß Links	6
2	Schulter Rechts	1	Gesäß Rechts	5
3	Gesäß Links	6	Rücken Links	4
4	Gesäß Rechts	5	Rücken Rechts	3
5	Schulter Links	2	Gesäß Links	6
6	Schulter Rechts	1	Gesäß Rechts	5
7	Gesäß Links	6	Rücken Links	4
8	Gesäß Rechts	5	Rücken Rechts	3

Dual 03				
Bewegung	Partie 1	Nr.	Partie 2	Nr.
1	Schulter Links	2	Bein Rechts	7
2	Bein Links	8	Schulter Rechts	1
3	Schulter Links	2	Rücken Rechts	3
4	Rücken Links	4	Schulter Rechts	1
5	Schulter Links	2	Bein Rechts	7
6	Bein Links	8	Schulter Rechts	1
7	Schulter Links	2	Rücken Rechts	3
8	Rücken Links	4	Schulter Rechts	1

Dual 04				
Bewegung	Partie 1	Nr.	Partie 2	Nr.
1	Schulter Links	2	Bein Links	8
2	Schulter Rechts	1	Bein Rechts	7
3	Bein Links	8	Bein Rechts	7
4	Gesäß Links	6	Gesäß Rechts	5
5	Schulter Links	2	Bein Links	8
6	Schulter Rechts	1	Bein Rechts	7
7	Bein Links	8	Bein Rechts	7
8	Gesäß Links	6	Gesäß Rechts	5

Dual 05				
Bewegung	Partie 1	Nr.	Partie 2	Nr.
1	Schulter Links	2	Gesäß Rechts	5
2	Gesäß Links	6	Schulter Rechts	1
3	Bein Links	8	Bein Rechts	7
4	Gesäß Links	6	Gesäß Rechts	5
5	Schulter Links	2	Gesäß Links	6
6	Schulter Rechts	1	Gesäß Rechts	5
7	Rücken Links	4	Gesäß Links	6
8	Rücken Rechts	3	Gesäß Rechts	5

Dual 06				
Bewegung	Partie 1	Nr.	Partie 2	Nr.
1	Schulter Links	2	Schulter Rechts	1
2	Rücken Links	4	Rücken Rechts	3
3	Gesäß Links	6	Gesäß Rechts	5
4	Bein Links	8	Bein Rechts	7
5	Gesäß Links	6	Gesäß Rechts	5
6	Rücken Links	4	Rücken Rechts	3
7	Schulter Links	2	Schulter Rechts	1
8				

Anhang G Übersicht aller Variablen im Arduino-Code

Name	Typ	Beschreibung	Gesetzt in	Abgefragt in
binary_adress	int Array [5]	5 Bit Binärzahl zur Ansteuerung der 32 Fahrprofile	decimalToBinary	decimalToBinary
sequences	int Array [2][20]	20 Stelliges Array, die das Gesamte Therapieprogramm darstellen, eine Stelle im Array bezeichnet eine Bewegungssequenz	resetSequences, Cmd_AddSequence, Cmd_LoadPatient	startNextSequence, sequenceStatusColor, sequenceStatus, Cmd_SavePatient,
sequence_counter	int	Durchläuft das Bewegungssequenzenarray und gibt die akutell ausgeführte Sequenz wieder	startNextSequence, resetSequences, Cmd_AddSequence, Cmd_LoadPatient, Cmd_StartReferencing, Setup	startNextSequence, sequenceStatusColor,
add_sequence_counter	int	Stelle des Bewegungssequenzen arrays, die gerade neu hinzugefügt wird	resetSequences, Cmd_AddSequence, Cmd_LoadPatient	resetSequences, Cmd_AddSequence,
error_strings	char* []	Char String Array in dem die Namen der Fehlermeldungen gespeichert sind		errorHandler
alert_strings	char* []	Char String Array in dem die Namen der Alarmmeldungen gespeichert sind		errorHandler
status_box	char* []	Char String Array in dem die Namen der Textboxen zur Darstellung der Bewegungssequenzen gespeichert sind	sequenceStatusColor, sequenceStatusReset	sequenceStatus
patients_save	char* []	Char String Array in dem die Meldungen an die Textbox gespeichert sind, die bei erfolgreichem Speichern eines Patienten ausgegeben werden		Cmd_SavePatient
patients_load	char* []	Char String Array in dem die Meldungen an die Textbox gespeichert sind, die bei erfolgreichem Laden eines Patienten ausgegeben werden		Cmd_LoadPatient
command	int	Dezimalzahl des Fahrprofils, dass in eine Binärzahl umgewandelt werden soll	Cmd_StartReferencing, Cmd_StopProgramAll, Loop	decimalToBinary
start_trigger	bool	Flag, die "true" sein muss, dass das Therapieprogramm gestartet bzw. weiterhin ausgeführt wird	StartNextSequence, errorHandler, Cmd_StartReferencing, Cmd_StartSequences, Cmd_StopProgramAll, Setup,	Main Loop
release_state	bool	Status der Bestromung aller Motoren	Cmd_StartReferencing, Cmd_CheckboxCurrent, Setup	Cmd_StartSequences, Cmd_CheckboxCurrent
endless_loop	bool	Gibt an, ob eine Endlosschleife des Durchlaufs des Therapieprogramms gesetzt wurde	Cmd_CheckboxEndless	startNextSequence
last_millis	unsigned long	Variable in der der letzte Zeitpunkt einer Abfrage der vergangenen Millisekunden gespeichert werden. Zur Periodischen Ausführung des "errorHandler"	manuell im Code	Main Loop
motor	int	Der Motor (1-8) der gestartet werden soll	Loop in den Sequenzen	startMotor
patient_number	int	Ausgewählter Patient aus Liste im GUI zum Speichern/Laden	GUI	Cmd_SavePatient, Cmd_LoadPatient
patient_adress	int	Aktuelle Adressenposition des zu speichernden/ladenden Patienten im EEPROM des Arduino (1-20)	Cmd_SavePatient, Cmd_LoadPatient	Cmd_SavePatient, Cmd_LoadPatient
initial_patient_adress	int	Erste Adressenposition des zu speichernden/ladenden Patienten im EEPROM des Arduino	Cmd_SavePatient, Cmd_LoadPatient	Cmd_SavePatient, Cmd_LoadPatient
sequences_t	enum	Enumeration, in der ALLE Schritte der einzelnen Bewegungssequenzen abgelegt sind	Hardcoded	startNextSequence, Main Loop
errors_t	enum			
up_01	int	Enthält die Nummer der Aufwärtsbewegung für alle "Single-Sequenzen"	startNextsequence	Main Loop
down_01	int	Enthält die Nummer der Abwärtsbewegung für alle "Single-Sequenzen"	startNextsequence	Main Loop

Anhang G Übersicht aller Variablen im Arduino-Code

up_02	int	Enthält die Nummer der Aufwärtsbewegung für alle "Dual-Sequenzen"	startNextsequence	Main Loop
down_02	int	Enthält die Nummer der Abwärtsbewegung für alle "Dual-Sequenzen"	startNextsequence	Main Loop

Anhang H Übersicht aller Funktionen im Arduino-Code

Typ	Name	Beschreibung	Aufgerufene Funktionen	Variablen	MgLnk
Bool	startNextSequence	Durchläuft das Sequence Array und Startet die gespeicherten Bewegungssequenzen in der Main, Zuweisung des maximalen Hubs	sequenceStatusColor	sequences, start_trigger, sequence_counter, sequence_maxlenght, endless_loop, ENUM sequence_, up/down_01/02	J
Void	resetSequences	Löscht das gesamte Sequenz-Array (setzt es auf 0)	sequenceStatus, sequenceStatusReset	add_sequence_counter, sequence_counter	N
Void	sequenceStatusColor	Hinterlegt die ausgeführte Bewegungssequenz im GUI mit grüner Farbe		sequences, sequence_counter, status_counter, status_box	J
Void	sequenceStatusReset	Hinterlegt wieder alle Sequenzboxen mit weiß		status_counter, status_box	J
Void	sequenceStatus	Gibt nach der Sequenz im Array den entsprechenden Namen jener in der Sequenzbox aus		sequences, status_counter, status_box	J
Bool	everythingReady	Überprüft, ob alle Steuergeräte bereit für die nächste Bewegung sind			N
Void	startEverything	Setzt alle hubsäulen gleichzeitig in Gang		flank_delay	N
Void	stopEverything	Stoppt alle Hubsäulen gleichzeitig		flank_delay	N
Void	startMotor	Startet die gewählte Hubsäule (1-8)		motor, motor_delay	N
Void	decimalToBinary	Wandelt die Dezimalzahl der gewählten Bewegung in einen Binärcode um und Setzt die 5-Bit Binärpins entsprechend		command, pin_number, bit_counter, binary_adress, rest	N
Void	Cmd_AddSequence (CommandParameter)	Button: Fügt die Bewegungssequenzen aus einer Liste via Knopfdruck zum Sequenzenarray hinzu	sequenceStatus	add_sequence_counter, sequence_counter, sequence_maxlength	J
Void	Cmd_ResetSequence (CommandParameter)	Button: Löschen des Sequenzenarrays	resetSequences	sequence_counter	N
Void	Cmd_SavePatient	Button: Speichert das aktuelle Sequenzenarray im EEPROM des Arduino laut Auswahl von GUI in der Patienteliste (1-10)		save_sequence_counter, patient_number, initial_patient_adress, patient_adress, sequences, patients_save	J

Anhang H Übersicht aller Funktionen im Arduino-Code

Void	Cmd_LoadPatient	Button: Lädt das unter Patient 1-10 gespeicherte Sequenzenarray und schreibt es in das aktuelle Sequenzenarray	sequenceStatusReset, sequenceStatus	load_sequence_counter, patient_number, initial_patient_adress, patient_adress, sequences, add_sequence_counter, sequence_counter, patients_load	J
Void	errorHandler	Überprüft die Fehler/Alarm Outputs der Dryves alle 5 Sekunden, deaktiviert den Start_trigger entsprechend und gibt eine visuellen Hinweis am GUI		error_counter, alert_counter, error_pin, alert_pin, error_subtractor, alert_subtractor, error_strings,alert_strings, sequence_, start_trigger	J
Void	Cmd_StartReferencing	Button: Bestromt alle Motoren und startet die Referenzierungsbewegung, stoppt vorher ggf laufende Bewegungen	stopEverything, sequenceStatusReset, startEverything, decimalToBinary	current_release, reference, start_trigger, sequence_counter	J
Void	Cmd_StartSequences	Button: Startet die ausgewählten Sequenzen, sofern Bestromung aktiv und kein Fehler gemeldet wurde, deaktiviert diverse Buttons	startNextSequence	release_state, start_trigger	J
Void	Cmd_Accept	Button: Bestätigung des Warnhinweises zu Beginn, aktiviert auch kritische Buttons des GUI			J
Void	Cmd_StopProgramAll	Button: Stoppt die laufenden Sequenzen: start_trigger auf 0 gesetzt, Stop Signale zu den Steuergeräten, danach werden alle Säulen eingefahren, Buttons wieder aktiviert	stopEverything, decimalToBinary, startEverything	start_trigger, flank_delay	J
Void	Cmd_CheckBoxCurrent	Checkbox: Aktiviert/Deaktiviert die Bestromung aller Motoren		release_state, current_release	J
Void	Cmd_CheckBoxEndless	Checkbox: Aktiviert/Deaktiviert die Option, das Sequenzenarray endlos zu wiederholen		endless_loop	J
Void	Setup	Arduino Standard: Zustand bei Reset/Start/Verbinden des Arduino, Initialisierung der I/O und der GUI Schaltflächen, Warnhinweis	resetSequences	last_millis, current_release, start_trigger, endless_loop, add_sequence_counter,	J

Anhang H Übersicht aller Funktionen im Arduino-Code

Void	Loop	Arduino Standard: Dauerschleife wird permanent durchlaufen, Eingaben abgerufen, Fehler periodisch ausgelesen und Sequenzen abgelegt	errorHandler, SerialCommandHandler, startNextSequence, everythingReady, decimalToBinary, startEverything, startMotor	last_millis, sequence_, start_trigger, flank_delay, sequence_counter, up_01, up_02, down_01. down_02	