Beuth Hochschule für Technik Berlin
University of Applied Sciences

Fachbereich VI
Informatik und Medien
Studiengang Technische Informatik – Embedded Systems

Bachelorarbeit

Haptisches Display
Recherche, Modellierung und Aufbau eines 3D-Demonstrators

Zur Erlangung des akademischen Grades „Bachelor of Engineering“

Betreuer:
Prof. Dr. Rozek

Zweitgutachter:
Prof. Dr. Rauchfuß

Steffen Roth
Straße 604 Nr. 18
12355 Berlin
Matrikelnummer: s759414
SteffenRoth31@aol.com

Beuth Hochschule für Technik Berlin
University of Applied Sciences

Fachbereich VI
Informatik und Medien
Studiengang Technische Informatik Embedded Systems

Bachelorarbeit

Haptisches Display
Recherche, Modellierung und Aufbau eines
3D-Demonstrators

Zur Erlangung des akademischen Grades
Bachelor of Engineering

Betreuer:
Prof. Dr. Rozek

Zweitgutachter:
Prof. Dr. Rauchfuß

Steffen Roth
Straße 604 Nr. 18
12355 Berlin
Matrikelnummer: s759414
SteffenRoth31@aol.com

Inhaltsverzeichnis

---

1. Abbildungs- und Quellcodeverzeichnis

Abbildungsverzeichnis

Quellcodeverzeichnis

Hinweis:

Da alle Programmteile mit einer grafischen Programmiersprache geschrieben sind, werden die Quellcodes ebenfalls gelistet, aber zur besseren Übersicht von den Abbildungen getrennt. Viele Quellcodeausschnitte sind zur besseren Übersicht mit Hilfe von Bildbearbeitungssoftware zusammengeschnitten, aber funktionell identisch mit dem Original.

2. Einleitung und Motivation

In der Vergangenheit war es Menschen mit Einschränkungen in der Sensorik, stark erschwert zu kommunizieren. Beispielsweise ist die Gebärdensprache für Gehörlose in Deutschland erst seit 2002 mit dem Inkrafttreten des BGG (Behindertengleichstellungsgesetz) gesetzlich als vollwertige Sprache anerkannt. [BGBl. I, S. 1468-1469] Auch sehbehinderte Menschen haben bisher starke Einschränkungen in ihren Kommunikationsmöglichkeiten hinzunehmen, vor allem im Hinblick auf moderne Kommunikationsmittel. So ist die Nutzung von Computerarbeitsplätzen bis heute lediglich mit sogenannten Braille- Laufzeilen auf Tastaturen oder sehr seltenen und teuren Braille- Displays möglich. Auch diese sind aber lediglich in der Lage, Braille- Schriftzeichen und grobauflösende 2D-Reliefs darzustellen, keine komplexeren Strukturen mit Höhen- und Tiefenverläufen.

Hierdurch sind Betroffene in der Nutzung von Diensten wie dem Internet stark eingeschränkt; die Verwendung dieser Hilfsmittel als mobile Lösungen, z.B. in Tablet PCs oder Mobiltelefonen, ist bislang gar nicht möglich. Dieses Problem zu lösen und sehbehinderten Menschen die Möglichkeit einer besseren Nutzung digitaler Abbildungen im Allgemeinen eröffnen zu können, ist das Ziel dieser Bachelorarbeit. Dazu soll ein Displayprototyp zur Darstellung von dreidimensionalen Objekten konstruiert und ein entsprechender Treiber erstellt werden. Dieser soll es Sehbehinderten, insbesondere vollständig Erblindeten, erstmals ermöglichen, Strukturen aus Computersimulationen zu ertasten.

Um dieses Ziel zu erreichen, wurden mehrere Ideen für verschiedenste Konstruktionen gesammelt, ausgewertet und als Zeichnungen und simple Prototypen-Modelle, teilweise aus einfachem Papier, aufgebaut und ausgewertet. Bei dem Aufbau des Displays ist darauf geachtet worden, dass die Darstellung von Schrift mittels Integration der bereits erwähnten Braille- Displayelemente in einem späteren Entwicklungsstadium weiterhin möglich bleibt, um es beispielsweise zu ermöglichen, eine dreidimensionale Darstellung zu kommentieren.

3. Projektverlaufsplan

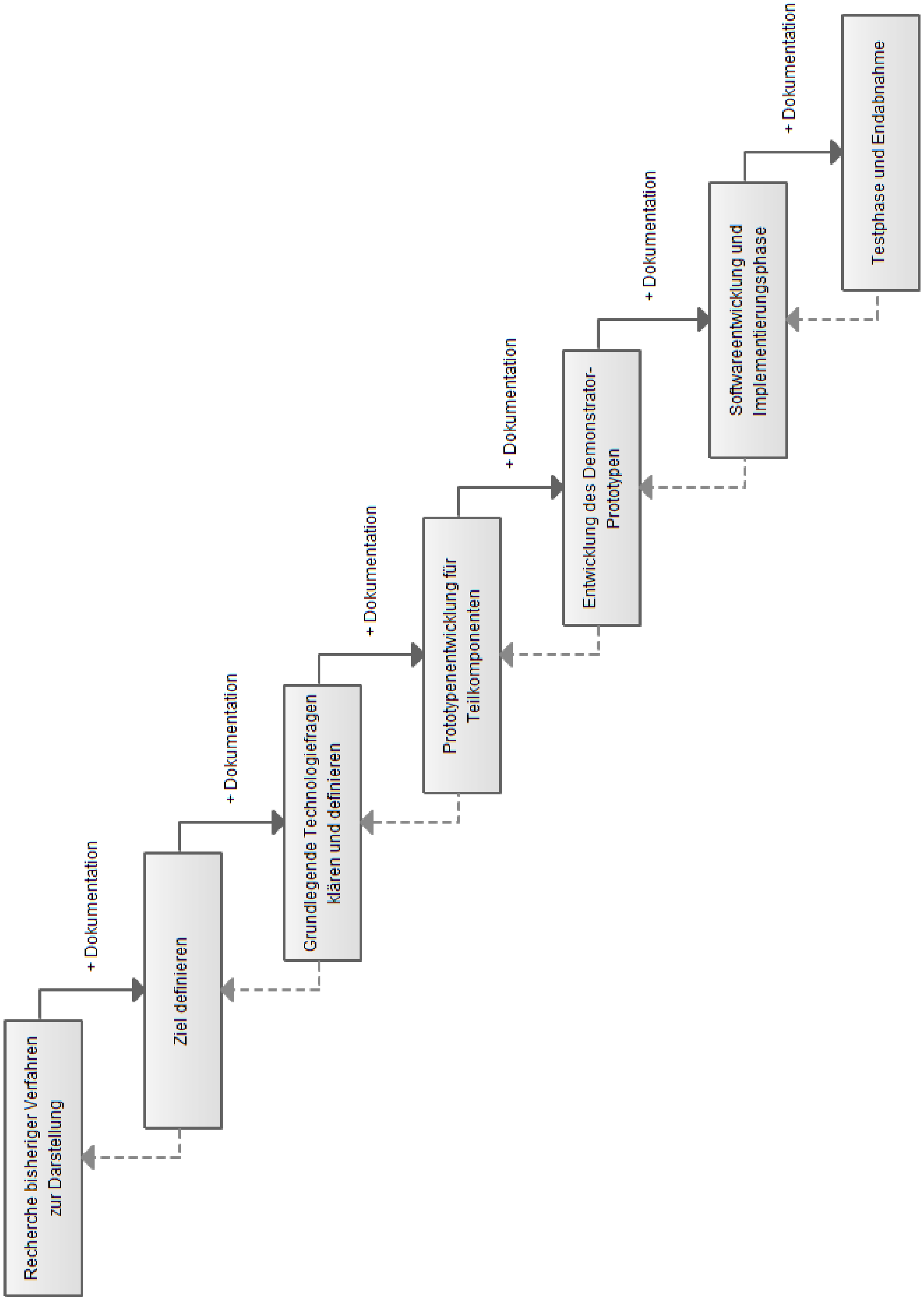


Abbildung 01: Wasserfallmodell des Projektverlaufs

4. Blindheit und Haptik

Es kann zahlreiche Gründe für die Erblindung eines Menschen geben. So kann es durch Unfälle oder Krankheiten zu einer Schädigung des Auges oder des Sehnervs kommen, oder durch Fehlbildungen kann das Sehen von Geburt an eingeschränkt oder unmöglich sein.

Weiterhin besteht die Möglichkeit, dass das Sehen „vergessen“ wird. Diesen Prozess beschreibt Richard L. Gregory als visuelle Agnosie, wobei der Begriff der Agnosie von Freud geprägt wurde als eine Unfähigkeit, sensorische Reize wahrzunehmen. Die Unfähigkeit des Sehens liegt dabei nicht in den Wahrnehmungsorganen selbst begründet, sondern vielmehr in der Verarbeitung dieser sensorischen Reize bzw. deren Zuordnung. So beschreibt Gregory: „Agnosien sind von großem psychologischem Interesse, denn sie kennzeichnen die Unfähigkeit, Objekt zu erkennen, obwohl die Augen, die Ohren oder der Tastsinn normal arbeiten. Allein die Bedeutung der sensorischen Signale ist verloren gegangen.“ [AGPS] S. 208 – 210

Dieses Vergessen tritt auf, wenn Teile des Gehirns geschädigt sind. Dies kann durch Unfälle schlagartig auftreten oder ein - zumeist krankheitsbedingter schleichender Prozess sein. Bei dem schleichenden Prozess, bei dem je nach Verlauf der Blick für „das Ganze“ verloren geht, werden zunehmend Objekte miteinander vertauscht oder nicht mehr erkannt. So kann es beispielsweise dazu kommen, dass Betroffene nicht mehr zwischen ihren Schuhen und ihren Füßen unterscheiden können, oder viele Details eines Bildes erkennen, aber nicht in der Lage sind, die Darstellung des vollständigen Objekts wahrzunehmen.

Um Menschen mit diesen Wahrnehmungsstörungen zu helfen, muss der optische Sinn, genau wie bei vollständig blinden Menschen, durch andere Wahrnehmungsmöglichkeiten unterstützt und in schwersten Fällen sogar vollständig ersetzt werden. Zu diesem Zweck bedient man sich unter anderem der Haptik (Tastsinn).

Das haptische Empfinden beruht nicht ausschließlich auf direktem Druck bzw. Druckunterschieden, sondern auf diversen Empfindungen. Das haptische Empfinden kann nach James J. Gibson in Subsysteme der Wahrnehmung eingeteilt werden, aufgrund z.B. des Unterschiedes zwischen Muskel-, Gelenks- und Hautempfindlichkeit. So kann „ein Geiger die Form seines Instruments bei geschlossenen Augen mit außerordentlicher Genauigkeit spüren. Ein Mann der am Stock geht, kann sogar Steine, Schlamm oder Gras am Ende des Stocks wahrnehmen. Alle diese Wahrnehmungen stammen jedoch von der Berührung der Haut mit den anliegenden Oberflächen und dem Kontakt der Knochen untereinander.“ [DSPW] S. 148

Im Umkehrschluss heißt dies auch, dass es nicht ausreichend ist, nur ein einzelnes dieser Subsysteme anzusprechen, um Strukturen bzw. Objekte deutlich erkennbar und begreiflich zu machen. Es müssen auch in einer künstlich erzeugten Darstellung alle Subsysteme gleichzeitig gereizt werden, damit die Illusion real wirkt.

Daher stellt sich die Frage, ob es bereits ähnliche Systematiken gibt, die Blinden helfen im Alltag wieder „zu sehen“. Die Antwort darauf lässt sich leicht finden und liegt in der Blinden- oder Brailleschrift.

5. Die Brailleschrift

Die Brailleschrift ist ein von dem in frühster Kindheit durch einen Unfall selbst erblindeten Franzosen Louis Braille erfundenes Schriftalphabet. [BRBU]

Dieses beruht auf einem geprägten Punktsystem, das ertastet werden kann, und die lateinischen Buchstaben ersetzt. Die Brailleschrift ist somit auch ein haptisches System bzw. eine haptische Darstellung mit einem gewissen Abstraktionsgrad. Dabei bilden insgesamt sechs Punkte (drei in der Höhe, zwei in der Breite) das gesamte Grundraster zur Darstellung der Buchstaben, Zahlen und Sonderzeichen. Da jeder Punkt im Raster erhaben oder nicht erhaben sein kann, ergibt sich eine Gesamtanzahl von maximal $2^6 = 64$ Zeichen, die aber mit Hilfe von Kurzzeichen noch einmal vereinfacht wurden. So gibt es in der deutschen Braille- Basisschrift Doppelbelegungen für Zahlen, das heißt der Buchstabe A hat auch den Wert 1, B den Wert 2, C den Wert 3 usw.. Dies muss allerdings durch ein entsprechendes Zeichen angekündigt werden. Zur Vereinfachung besteht die Basisschrift zudem lediglich aus Kleinbuchstaben, die ebenfalls durch ein Sonderzeichen zu einem Großbuchstaben erklärt werden können. Eine weitere gebräuchliche Erweiterung der Brailleschrift ist die sogenannte Vollschrift, die für sehr häufig verwendete Buchstabengruppen wie zum Beispiel „sch“, „ei“ oder „eu“ ein einzelnes Zeichen einführt.

Durch diese Maßnahmen wird die Textlänge etwas verkürzt und die Lesbarkeit stark erhöht.

Aus demselben Grund gibt es auch noch die Erweiterungen der Kurzschrift, welche der Stenografie der sogenannten Schwarzschrift (siehe Kapitel 12; Abkürzungs- und Fremdwortverzeichnis) ähnelt und starke Abkürzungen bereitstellt. Damit können Texte um ca. 30 % bis 40% kürzer geschrieben und von geübten Blinden fast genauso schnell wie Schwarzschrift von Sehenden gelesen werden. Allerdings setzt diese Art des Schreibens und Lesens bereits eine große Übung im Umgang mit der Blindenschrift voraus.

Als letzte Verkürzung der Blindenschrift existiert noch die Blindenstenografie. Diese setzt sich aber aus einem sehr komplizierten Regelwerk zusammen, welches das Punktsystem teilweise auf 8 Punkte erweitert und kann daher nur von sehr wenigen, speziell ausgebildeten Personen geschrieben und gelesen werden. [BLST]

5.1. Blindengerechte Darstellungen am Computer

Für die Darstellung von Braille- Schriftzeichen gibt es bereits seit einigen Jahren Laufzeilen auf speziellen Tastaturen und einfache Braille Displays. Letztere weisen jedoch das Problem auf, dass sie sehr kostspielig und in Ihrer Darstellungsfähigkeit auf Schriftzeichen beschränkt sind oder allenfalls bei sehr teuren Varianten zweidimensionale Reliefs grob abbilden können.

Dementsprechend ist es zwar möglich, einen Text, z.B. einen Zeitungsartikel im Internet, zu lesen oder selbst einen Text am PC zu verfassen. Die Verwendung von Bildern, etwa um kompliziertere Sachverhalte zu verdeutlichen, ist aber nicht möglich.

Bücher in Brailleschrift sind in der Herstellung sehr Aufwändig. Dementsprechend sind solche Bücher selten verfügbar und teuer. Darüber hinaus ist der Platzbedarf für einen Text in Brailleschrift um ein Vielfaches höher als für den gleichen Text in Schwarzschrift, da er bei Unterschreitung einer gewissen Mindestgröße nicht mehr erkenn- bzw. ertastbar wäre. Hinzu kommt durch die Prägung bedingte größere Dicke der Seiten. Damit ist ein Medium wie ein PC, mit dem man Platzsparend und kostengünstig viele Informationen speichern und abrufen kann, gerade im Bereich der Kommunikation für Blinde ein immer wichtiger werdendes Element.

Abbildung 02: Braille Tastatur mit geringen Abweichungen zum Standard Tastaturlayout [BAUM]

Eine Besonderheit, die bei dieser Tastatur auffällt ist, dass das Grundraster für weitere computertypische Sonderzeichen um zwei Punkte erweitert wurde.

6. 3D-Drucker

In den letzten Jahren begegnet man immer öfter dem Begriff des „3D-Druckers“. Auch die Beuth Hochschule für Technik Berlin besitzt mehrere dieser Drucker und stellt diese auch für Projekte bereit.

Ein 3D Drucker ist, wie der Name bereits vermuten lässt, in der Lage, aus einem am Computer entworfenen CAD- Modell ein reales dreidimensionales Objekt zu erstellen. Hierfür steht eine große Auswahl an zum Druck verwendbare Materialien zur Verfügung. Die gebräuchlichsten sind Kunststoffe wie z.B. ABS, welcher ein vergleichsweise harter Kunststoff ist, aber auch weichere Materialien wie Nylon können, mit etwas Experimentieren bei der Drucktemperatur und Druckgeschwindigkeit, verwendet werden. Drucke aus Metall anzufertigen ist ebenfalls möglich, allerdings sind dafür spezielle Drucker notwendig auf die an dieser Stelle nicht eingegangen wird, da diese sehr kostenintensiv sind und nach einem grundlegend anderen Verfahren als die an der Beuth Hochschule verfügbaren Drucker arbeiten.

6.1. Der Aufbau

Die in der Beuth Hochschule verwendeten 3D Drucker bestehen aus vier grundlegenden Einheiten:

- Das Heizbett besteht aus einer elektrischen Heizung auf einer Platine und einer darauf liegenden Glasplatte. Die Glasplatte ist der Träger für zu druckenden Objekte und muss aufgeheizt werden, damit die Kunststoffteile auf der Platte besser haften. Anderenfalls würden diese sich zu leicht lösen und während des Druckes verrutschen. Um noch bessere Hafteigenschaften zu erreichen wird die Glasplatte meistens noch mit einer Kunststoff-Klebefolie versehen.

- Der Druckkopf besteht aus dem Extruder und der Spritzdüse, die von einem Heizelement umgeben ist. Der Extruder hat die Funktion, mittels eines Zahnrades das in Fadenform gelieferte Kunststoffrohmaterial von der Rolle in das Heizelement bzw. die Spritzdüse zu drücken. Die Spritzdüsen stehen in unterschiedlichen Stärken zur Verfügung, mit denen der gespritzte Kunststoffaden unterschiedlich dick wird. Bei dem verwendeten Drucker der Hochschule beträgt die Stärke 0,5 mm, was im Vergleich mit anderen 3D-Druckern bereits recht grob ist. Die feinsten Spritzdüsen, die momentan erhältlich bzw. selbst herstellbar sind, haben eine Stärke von 0,15 mm, sind aber selten, da hiermit bereits kleine Objekte damit im Druck sehr viel Zeit benötigen (mehrere Stunden).

- Die Achsen sind bei dem verwendeten Drucker mit vier Motoren verbunden. Zum gleichmäßigen Hochfahren der Spritzdüseneinheit entlang der Höhenachse (Z-Achse) werden zwei Motoren benötigt. Die Spritzdüseneinheit selbst ist über einen Keilriemen mit dem Motor für die Y-Richtung verbunden, das spart Gewicht an den beweglichen Teilen und sorgt für eine größere Genauigkeit. Die X-Richtung wird durch eine Verschiebung des Heizbettes kontrolliert, dies geschieht ebenfalls über einen Keilriemen.

- Die Steuerung besteht aus Arduino-Baugruppen und ist sowohl für die Steuerung der Heizelemente als auch für die Steuerung der Motoren verantwortlich. Die Ansteuerung findet dabei über einen normalen USB-2-Port statt. [ARDU]

6.2. Druckverfahren und Dateiformate

Um einen 3D-Drucker betreiben zu können und ein Modell auszudrucken, wird zunächst ein Programm benötigt, mit dem das Modell als CAD-Modell erstellen werden kann. In diesem Fall wird dafür „SketchUp Pro 8“ der Firma Google verwendet (siehe auch Kapitel 9.1., Die Fertigung der Displayelemente).

Beim Entwerfen der Modelle muss darauf geachtet werden, dass die Teile, die nach oben hin gedruckt werden sollen, nicht zu große Winkel zur Grundfläche des Heizbettes haben dürfen, da sie sonst vor der Aushärtung des Kunststoffs leicht wegknicken könnten. Gleichfalls ist darauf zu achten, dass keine Elemente ohne Befestigungen in der Luft gedruckt werden, daher sind alle nachfolgenden

Modelle in Ihrer CAD Vorschau gegenüber dem späteren Einbau auf den Kopf gestellt, damit ihre größten Flächen zuunterst gedruckt werden.

Als Nächstes muss das Modell mittels eines Exportprogramms, welches SketchUp als Plug-In extra hinzugefügt wurde, in eine STL Datei konvertiert werden. [TRIM]

Diese Modelldatei wird dann in Pfade konvertiert, die der Drucker bzw. der Druckkopf verstehen und abfahren kann, ohne das Druckmodell unregelmäßig zu drucken oder sogar während des Druckens zu rammen und damit gleich wieder zu beschädigen. [AUFA] Hierfür wird das Programm „Slic3er“ (Slicer 3) benutzt. Die dabei entstehende Pfaddatei, die .gcode Datei, ist die entscheidende für den eigentlichen Druckertreiber „Pronterface“, der die Arduino-Steuerung des Druckers steuert und überwacht.

7. Aufbau des ersten Prototyps

Die Konzeptionierung

Während der Konzeptionierungsphase werden viele Ideen zur Umsetzung des Ziels gesammelt und mit verschiedenen Techniken als grobe Prototypen festgehalten. Dazu werden diese teilweise auch als Modelle z.B. aus Papier, Legosteinen oder anderen einfachen Elementen zur Veranschaulichung konstruiert und in ihrer grundlegenden Anwendung simuliert und teilgetestet.

7.1. Konzept 1: Umsetzung mittels dehnbarer Oberflächenstruktur

Die erste Idee für ein Display sieht vor, einen Rahmen mit einem dehnbaren und weichen Stoff (z.B. einem Nylongewebe) zu bespannen, um diesen dann mit Hilfe von Steuerdrähten, die in der Stoffbahn verhakt sind, zu verziehen. Die Drähte werden dabei, um Platz zu sparen, über Umlenkrollen von Bowdenzügen mit Servomotoren verbunden.

Die Vorteile dieser Displayart sind, dass Höhenunterschiede zwischen zwei direkt nebeneinander liegenden Steuerdrähten nicht weiter beachtet werden müssen, da diese Unterschiede direkt von der Stoffbahn ausgeglichen werden. Ein weiterer Vorteil dieser Konstruktion ist, dass das Display eine komplett geschlossene Fläche hat und somit für den Benutzer keine „blinden Stellen“ entstehen.

Durch diese Art des Displays entstehen aber auch starke Einschränkungen, so wird beispielsweise die Stoffbespannung einer hohen Belastung ausgesetzt, die zu einem starken Verschleiß und somit zu einer vergleichsweise kurzen Einsatzfähigkeit führt, was wiederum zur Folge hat, dass die Bespannung häufig gewechselt werden muss. Dies wäre besonders durch die Verhakungen der Steuerdrähte sehr aufwändig.

Der Vorteil einer weichen Bespannung ist auch mit einer weiteren negativen Eigenschaft verbunden: Durch den von der Hand des Benutzers zwangsläufig ausgeübten Druck, verformt sich die Bespannung und die dargestellte Form ist nur noch schwer erkennbar.

7.2. Konzept 2: Umsetzung mittels Druckelement für Einzelfinger

Der zweite Entwurf einer Anzeige besteht aus einem druckerzeugenden Element, das auf einen Finger aufgesetzt wird und durch verschiedene Druckstärken Höhen und Tiefen simulieren soll.

Der große Vorteil dieser Konstruktion ist, dass der Hardwareaufwand auf lediglich ein Element pro Finger beschränkt ist. Doch bei näherer Betrachtung fällt schnell auf, dass diese Umsetzung fehlerhaft ist, da zum einen mittels Sensoren immer die exakte Position des Fingers ermittelt werden müssen. Das betrifft sowohl die räumliche Koordinate des Fingers, als auch den Winkel des Fingers zum Horizont, um die Stellung an einem simulierten dreidimensionalen Objekt exakt wiedergeben zu können. Diese Art der Positionsbestimmung ist allerdings sehr aufwändig und störanfällig.

Darüber hinaus kommt kein Gefühl des Strukturverlaufs auf, da mit dem Finger Höhenverläufe durch das Zusammenspiel von Bewegungen und Oberflächengefühl als Form wahrgenommen werden können, nicht aber allein durch einen veränderten Druck auf einen Finger. (siehe Kapitel 4, Blindheit und Haptik). Dies kann man bereits in einem einfachen Versuch feststellen, indem man zwei oder mehr Objekte nimmt (z.B. eine Tastatur und eine Fernseherfernbedienung). Eine Testperson die einmal mit geschlossenen Augen diese Objekte mit einem einzelnen Finger erfühlt, wird die zwei Objekte problemlos erkennen. Wenn dieser Versuch nun wiederholt wird, indem der Finger der Testperson ein paar Zentimeter über die Objekte bewegt wird und eine zweite Person mit zwei eigenen Fingern nur den Druck - je nach Höhen und Tiefen, über denen sich der Finger der Testperson befindet - verändert, sind die Objekte nicht mehr unterscheidbar.

Daher müsste wiederum eine sehr kleine und entsprechend aufwändige Mechanik konstruiert werden die eine Bewegung im Verlauf simulieren kann. Dies setzt Mikromechaniken und Mikroelektronik voraus, die zu aufwändigen Produktionsverfahren führen. Damit erweist sich dieser Entwurf als untauglich.

7.3. Konzept 3: Umsetzung mittels Verschiebung fester Quaderstrukturen

Der dritte Entwurf entsteht aus einer Idee von Herrn Prof. Rozek für ein Bewegungsgelenk. Der Entwurf besteht aus mehreren Reihen von quaderförmigen Strukturen, die an jeweils zwei gegenüberliegenden Seiten geöffnet bleiben. Alle diese Strukturen werden in Reihen angeordnet und innerhalb der jeweiligen Reihe an den Seiten miteinander verbunden bzw. bekommen nur eine gemeinsame Wand mit beweglichen Achsen. Durch die fehlenden Seitenwände können die Strukturen stark gestaucht werden und dadurch Höhenverläufe simulieren.

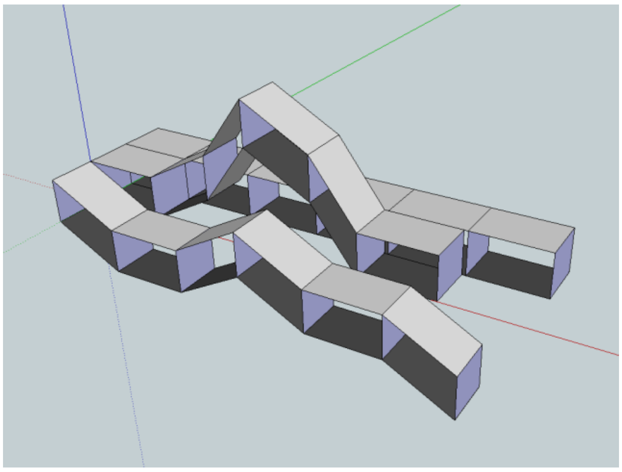


Abbildung 03: Darstellung mittels Quaderstrukturen

Doch es fallen schon anhand der obigen Skizze die massiven Nachteile dieser Konstruktion auf. So besitzen alle drei Reihen sechs Rechteckstrukturen mit den exakt gleichen Ausmaßen. Lediglich die Winkel unterscheiden sich. Dadurch entstehen schon bei geringen Höhenunterscheiden zwischen zwei Reihen, starke Längenunterschiede. So ist die hinterste Reihe ohne Auslenkungen schon mehr als eine komplette Quaderlänge länger als die davor liegende mit stärkeren Auslenkungen. Dadurch ist die Herstellung einer kohärenten Displayfläche nicht möglich. Darüber hinaus ist die Breite jeder Reihe fest und kann auch nicht im Anstellwinkel verändert werden.

7.4. Konzept 4: Umsetzung des finalen Prototyps

Der letzte und umzusetzende Entwurf ist ein Hybrid aus den vorhergehenden Entwürfen. Er besteht aus festen Kunststoffdreiecken, die zusammen eine feste Oberfläche bilden. Diese Dreiecke können über Drähte an ihren Ecken in ihrer Ausrichtung beeinflusst werden und somit komplexere Höhenverläufe bilden.

Damit ist die Stabilität einer festen Oberfläche gegeben, welche ein gutes Druckgefühl gewährleisten soll. Gleichzeitig ist die Formbarkeit einer flexiblen Aufhängung für komplexe Strukturen gegeben. Um den Aufwand an Steuermodulen und damit einen großen Kostenfaktor senken zu können, werden die Ecken der Dreiecke allerdings nicht einzeln mit jeweils einem Steuermotor verbunden, sondern erst in Gruppen zusammengefasst.

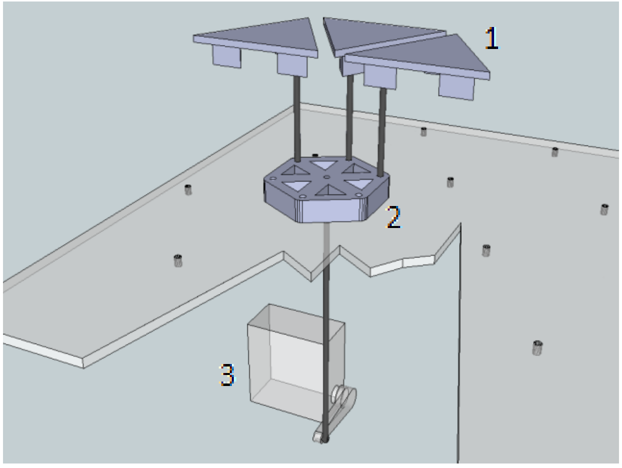


Abbildung 04: Die Displayelemente

Diese Verteilerelemente (2) sind dann mit jeweils einem Servomotor (3) verbunden und können auf diese Art maximal sechs Dreiecke (1) gleichzeitig steuern. Somit werden für eine Auflösung von 25 Dreiecken statt 75 Steuermotoren nur noch 19 Motoren benötigt. Gleichzeitig hat dies den Effekt, dass die Höhenunterschiede zwischen direkt nebeneinander liegenden Dreiecken nicht mathematisch ermittelt bzw. interpoliert werden muss, sondern dies rein durch den Aufbau bereits mechanisch geschieht.

Dabei entstehen allerdings, je nachdem ob die Dreiecke in ihrer Ausrichtung gegenüber der Displayebene einen „Berg“ oder ein „Tal“ bilden, unterschiedliche große Abstände zwischen den Dreiecken, was auch nicht ohne Weiteres verhindert werden kann.

8. Die Pulsweitenmodulation

Die Pulsweitenmodulation (PWM) ist eine zur Kontrolle analoger Schaltkreise durch digitale Ausgänge technischer Anlagen gut geeignete Technik. Die PWM wird in vielen Bereichen wie der Messtechnik, der Kommunikationstechnik und der Steuerung von technischen Anlagen (z.B. durch kontrollierte Stromzufuhr) angewandt. [BARR] Auch die später zur Ansteuerung verwendeten Servomotoren werden über die PWM kontrolliert. Dieser Vorgang wird zwar von Servocontrollern eigenständig übernommen, kann aber prinzipiell auch ohne diese komplexen Komplettbausteine ausgeführt werden. Um das nähere Verfahren der PWM erläutern zu können, müssen zuvor die Begrifflichkeiten der analogen und der digitalen Signale verdeutlicht und definiert werden.

8.1. Analoge Signale

Ein analoges Signal hat die Eigenschaften, dass es kontinuierlich, variabel und mit einer unendlichen Genauigkeit der Auflösung der Zeit und Zustandsgröße vorliegt.

Als einfaches Beispiel für ein typisches Analogsignal kann eine normale 9 Volt Batterie genommen werden, die an einen Widerstand angeschlossen ist. Diese wird im zeitlichen Verlauf immer weiter entladen und ändert damit kontinuierlich ihre Spannung. Dieses Signal der Batterie ist damit variabel und unendlich, da die Batterie immer eine Spannung liefert (nach vollständiger Entladung 0 Volt, aber dieser Wert ist ebenfalls kontinuierlich). Die Spannung unterliegt auch keinerlei Stufung, da sie alle Bereiche zwischen 9 Volt und 0 Volt durchläuft, und (entsprechend genaoue Messung vorausgesetzt) jeder Wert in diesem Bereich zu irgendeinem Zeitpunkt feststellbar wäre.

8.2. Digitale Signale

Ein digitales Signal hat grundsätzlich lediglich zwei Werte, die entsprechend ein Maximum und ein Minimum bilden (davon abweichende Zwischenwerte sind unerwünschte physikalische Nebeneffekte und werden auf Grund der Komplexität hier nicht weiter betrachtet). Dies hat den Vorteil, dass die anliegenden Potentiale eindeutig zugewiesen sind und sich auch kontrolliert sehr schnell ändern können. Als einfaches Beispiel hierfür kann eine LED mit einem Taster genommen werden. Wenn der Taster gedrückt wird, liegt sofort die maximale Spannung (z.B. 1 Volt), die in das System gespeist wird, an, und die LED leuchtet; wird der Taster losgelassen fällt die Spannung wieder auf das Minimum, 0 Volt, zurück und die LED geht aus. Zwischen diesen zwei Werten gibt es keinerlei Zwischenwerte.

8.3. Die Pulsweitenmodulation

Mit Hilfe der Pulsweitenmodulation ist es möglich, analoge Signale aus digitalen Signalen heraus zu simulieren. Um dies zu erreichen, bedient man sich der Trägheit. Dazu werden digitale Signale auf ein Bauteil gegeben, das so langsam reagiert, dass es scheint, als wäre es in einem Zwischenzustand. Ein für den Effekt anschauliches Bauteil, ist eine einfache Glühbirne. Wenn diese mit einer gewissen Frequenz angesteuert wird aber nur bei jedem zweiten Takt Strom bekommt blinkt sie. Bei einer niedrigen Frequenz von z.B. 1 Hz leuchtet sie eine Sekunde lang und ist dann 1 Sekunde aus. Das wäre ein sehr auffälliges Blinken. Wenn man diese Frequenz jedoch erhöht, z.B.100 Hz, würde die Glühbirne 100 Mal in der Sekunde an- und aus geschaltet. Dies hat zum Effekt, dass die Glühbirne so langsam reagiert das es so scheint als würde sie nur halb so viel Spannung bekommen wie auf Volllast und scheinbar auch nur halb so hell leuchtet. Wenn nun noch die Frequenz des An- und Ausschaltzyklus variiert wird, z.B. auf 25 Hz, kann damit auch die Helligkeit präzise eingestellt werden. [RNPW]

Nach diesem Prinzip funktioniert auch die Ansteuerung von Servomotoren. Hierbei ist aber die Besonderheit zu beachten, dass die meisten Servos intern mit einer Frequenz von 50 Hz arbeiten, d.h. das der Interne Regler mit dieser Frequenz angesteuert werden muss. Während der daraus resultierenden 20ms Periode ist es entscheidend, wie lange ein Servo angesprochen wird, um die

Auslenkung des Servos vorzugeben. Dabei sind die Standardwerte, die bei Servomotoren gebräuchlich sind, bei 1ms für die minimale Auslenkung des Servos, 1,5ms für die Mittelstellung und 2ms für maximale Auslenkung. [RNSE]

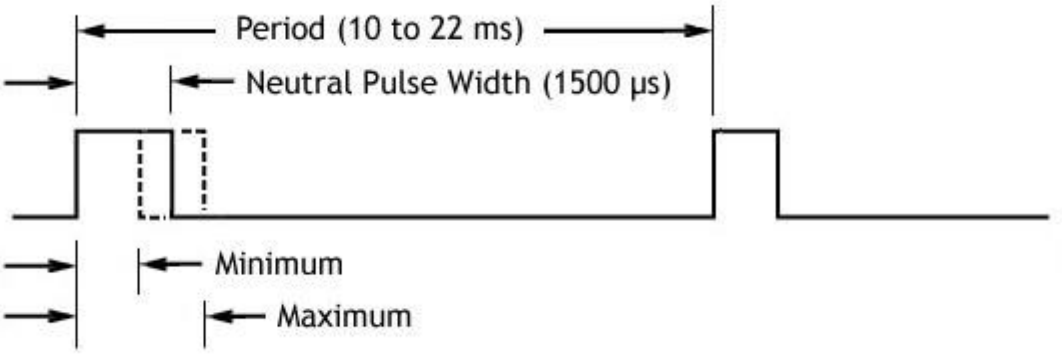


Abbildung 05: PWM Signalverlauf für Servo mit angedeuteter Reichweite für Maxima Stellungen [ACRO]

9. Der Aufbau des Displays

Da die mechanischen Einzelteile des Displays allesamt Spezialanfertigungen und nicht im Handel erhältlich sind, müssen diese von Hand angefertigt werden. Um dabei gleichbleibende Qualität in der Verarbeitung und Präzision erreichen zu können, werden die Einzelteile in einem 3D Modellierungsprogramm erstellt und dann mittels 3D-Druckverfahren aus Kunststoff gefertigt. (siehe Kapitel 6, 3D Drucker)

Die Drähte, die die Servomotoren mit den Verteilerelementen verbinden, und die Kunststoffleiter, die die dreieckigen Frontpaneele mit den Verteilerelementen verbinden, werden komplett per Hand angefertigt. Für diese Aufgaben werden sehr verwindungssteife Metallstäbe und Kunststofffäden mit einer hohen Elastizität benötigt. Daher werden für die Metallstäbe ein 0,8 mm starker Federstahldraht und für die flexiblen Kunststofffäden ein ebenfalls 0,8 mm starker Lichteiter verwendet. Alle mechanischen Elemente werden mit dem „Kraftkleber“ der Firma Tesa fixiert.

9.1. Die Fertigung der Displayelemente

Die Einzelteile, die mittels eines 3D-Druckers entstehen, werden wie bereits erläutert mit Hilfe der CAD-Software „Google SketchUp Pro 8“ entwickelt. SketchUp soll es dem Benutzer möglichst einfach machen, dreidimensionale Objekte zu erstellen und zu bearbeiten. Es gibt dabei sechs grundsätzliche Templates für verschiedene Verwendungszwecke.

Jedes Template ist doppelt vorhanden, um das metrische System und das z.B. in den USA verwendete angloamerikanische Maßsystem (Fuß und Zoll) zur Verfügung zu stellen.

Für das Ingenieurswesen, mit einer Unterscheidung nach Größe der zu entwerfenden Teile gibt es zwei unterschiedliche Templates. Das Engineering Template ist auf größere Maschinenteile oder sogar ganze Maschinenentwürfe ausgelegt, wohingegen das „Product Design und Woodworking Template“ besonders auf kleinere Konstruktionen spezialisiert ist. Das zeigt sich schon in den Maßeinheiten der Templates. So ist die Standardmaßeinheit des Engineering Template ein Meter und die des „Product Design und Woodworking Template“ ein Millimeter. Da die Teile für den Prototyp des Displays ebenfalls sehr klein sind (die größte Strecke ist die Kantenlänge eines der Frontdreieckspaneele, die 30 mm beträgt), wird dieses Template gewählt.

Es werden für das Display insgesamt 4 verschiedene Teile mit Hilfe des 3D Druckers der Beuth Hochschule produziert und entsprechend mit Google SketchUp entworfen:

- Die Dreieckfrontpaneele, welche bei Fertigstellung des Displays erfühlt werden können,

- Die Verteilerelemente, diese verbinden am Ende die Dreiecke mit den Steuermotoren und sind damit ausschlaggebend für den wesentlich geringeren Hardwareaufwand an Motoren, (Kapitel 7.5.; Konzept 4: Umsetzung des finalen Prototyps)

- Eine Grundhalterplatte, die die Aufgabe hat, das für das Display notwendige, mechanische Gestänge in seiner Position zu halten und zu stützen,

- Eine Servohalterplatte, die genau auf die verwendeten Servos abgestimmt wird. Dabei muss auf Kabelanschlüsse und die Standard- Servohalter an den Servos selbst geachtet werden. Da die Servos aber erst vergleichsweise spät in das Display selbst eingesetzt werden, ist diese Platte auch mit einigem zeitlichen Abstand zu den anderen Displayelementen zu entwerfen und zu fertigen.

9.2. Die Entwicklungsstadien

Erster Entwurf: Spinnenförmiger Entwurf

Es gibt in der Herstellung der Dreieck-Frontpaneele für deren Struktur und die damit direkt verbundenen Verteilerelementen insgesamt vier Entwürfe.

Der erste dieser Entwürfe sieht vor, dass an der Unterseite des Dreiecks drei Stege angebracht werden. Diese Stege sollen schräg nach oben verlaufen und sich in der Mitte des Dreiecks treffen, um eine Aufnahme für einen Stabilisierungsdraht zu bilden.

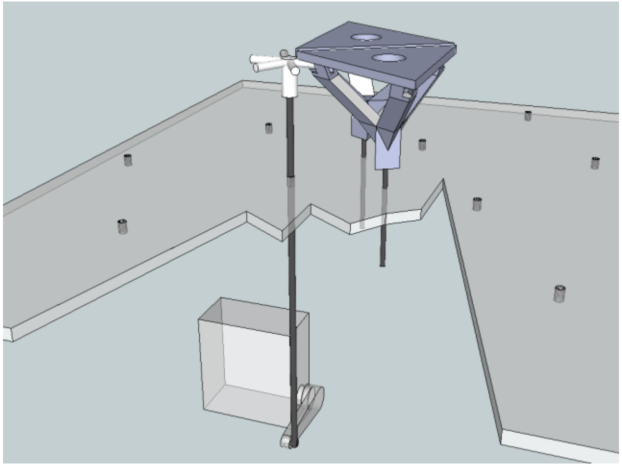


Abbildung 06: Gesamtvorschau erster Entwurf

In die Stege werden die bereits erwähnten Verteiler, welche hier spinnenförmig sind, eingesteckt und an ihren Enden unter Erwärmung breiter gedrückt, um ein Herausrutschen zu verhindern. Dafür verfügen die Stege über Aussparungen. In diesem Entwurf wird ebenfalls eine Aussparung in der Mitte des Displaydreiecks eingelassen, um eine eventuelle spätere Erweiterung für Braille- Elemente zu ermöglichen. Der Draht, der sich zentral unter jedem Dreieck- Paneel befindet, soll dieses zusätzlich stützen.

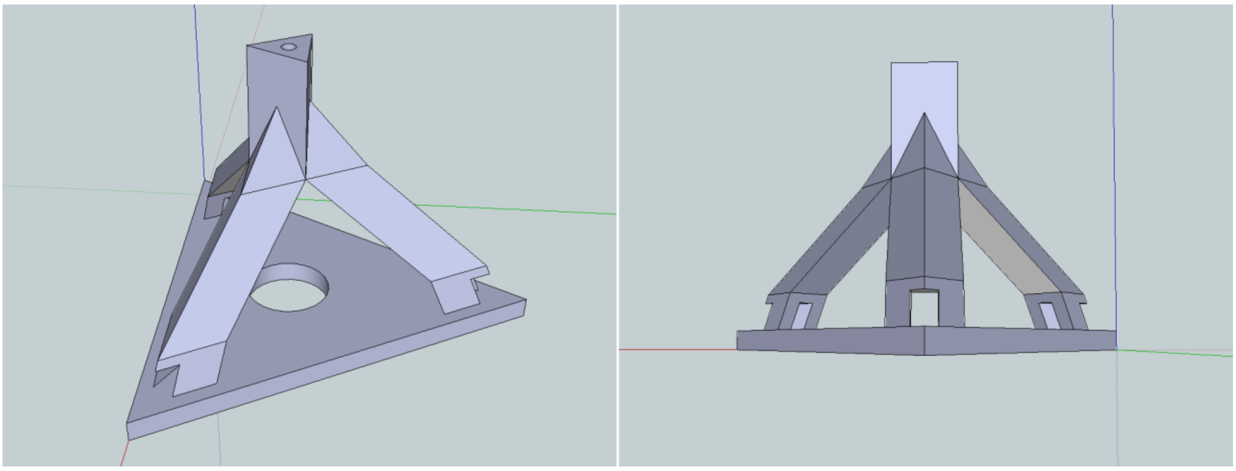


Abbildung 07: erster Entwurf des Dreieck- Frontpaneels CAD Vorschau

In der isometrischen Perspektive sind die Löcher für die Aufnahme des Stützdrahtes an der oberen Spitze der Konstruktion und das Loch für die Braille- Elemente in der Dreiecksfläche gut erkennbar (linkes Bild). In der Seitenansicht ist das Loch für die Verteilerelemente in einem der Stege sichtbar (rechtes Bild)

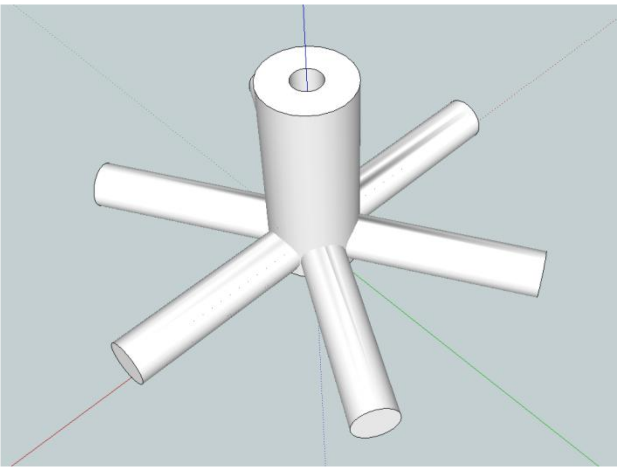


Abbildung 08: spinnenförmiges Verteilerelement CAD Vorschau

Das Verteilerelement besteht aus sechs spinnenförmig angeordneten Stangen mit einem Verbindungsknoten in der Mitte, in den auch der Führungsdraht eingelassen wird. Überflüssige Verbindungsbrücken, z.B. an Displayrändern, können einfach entfernt werden.

Während des Drucks der Dreieck- Frontpaneele kommt es zu Problemen, da die Stege im Bereich der eingelassenen Aussparungen mit 1,5 mm zu dünn und damit zu instabil für den weiteren Druckvorgang sind. Das ist besonders problematisch, solange der Kunststoff beim Drucken noch weich ist, was nur mit einer massiven Kühlung zu verhindern wäre. Eine Kühlung ist aber bei dem verwendeten Drucker momentan nicht verfügbar. Der Rest der Stege kann daher nicht korrekt gedruckt werden, da der Unterbau immer wegsackt. Somit muss der Druck abgebrochen und dieser Entwurf verworfen werden.

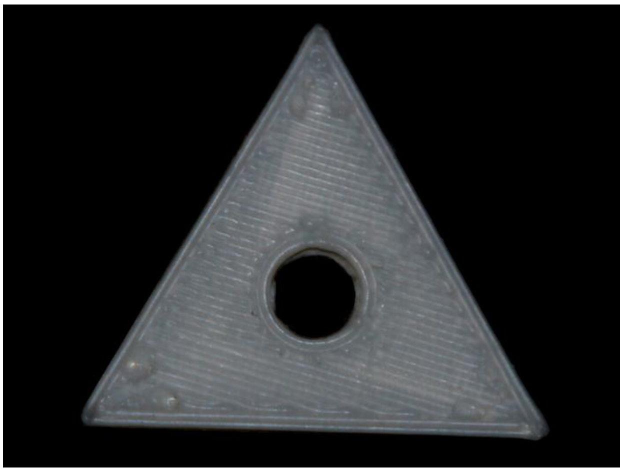


Abbildung 09: erstes fehlerhaft gedrucktes Dreieckspaneel

Es ist deutlich erkennbar, dass die komplette obere Struktur zur Steuerung fehlt. Die Bruchkanten sind am besten an der unteren linken Ecke erkennbar.

Zweiter Entwurf: Turmstruktur

Der zweite Entwurf weicht, wie auch die Späteren, wegen den beschriebenen Stabilitätsproblemen in der Aufhängung der Dreieck- Frontpaneele stark ab. Es werden nun insgesamt vier kleine dreieckige „Türme“ an die Unterseite des Dreieck- Paneels angebracht. In jedem dieser Türme befindet sich ein Loch für die Aufnahme eines Drahtes, bzw. der Kunststoffleiter. Dabei sollen die Kunststoffleiter an den Ecken, zur Ausrichtung des Dreiecks dienen, und der Draht in der Mitte um die Position des Dreiecks insgesamt zu stützen. Auch die Verteiler Bauteile müssen entsprechend angepasst werden. Sie bestehen nicht mehr in der filigranen Stangenkonstruktion, sondern aus einer 12- eckigen Platte, mit Löchern als Gegenstück für die flexiblen Kunststoffleiter die zur Ausrichtung der Dreiecke dienen und einem zentral angebrachten Loch für den Steuerdraht, der mit dem Steuerservo direkt verbunden wird.

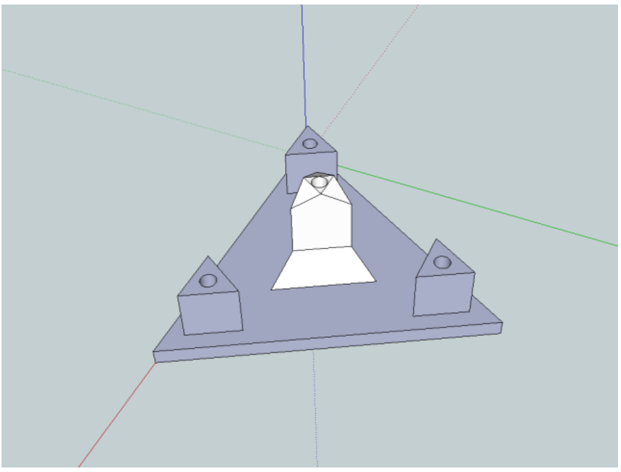


Abbildung 10: zweiter Entwurf mit turmartiger Struktur CAD Vorschau

In der Mitte ist gut sichtbar der erhöhte Turm (weiß) zur Aufnahme des Stützdrahtes zu erkennen, ebenfalls gut erkennbar sind die drei äußeren Aufnahmen der Kunststoffleiter.

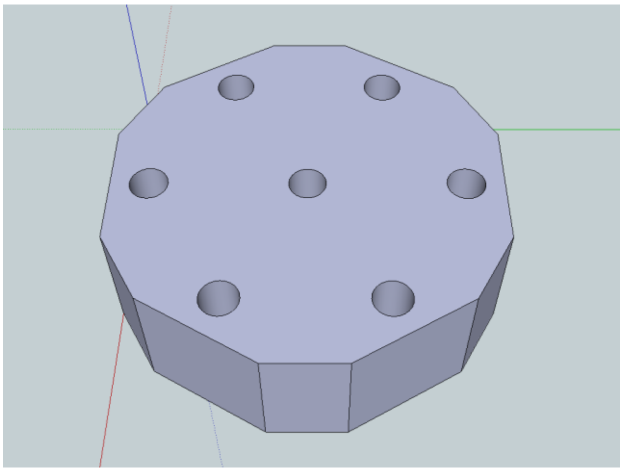


Abbildung 11: massives Verteilerelement CAD Vorschau

Der Druck dieses zweiten Prototyps funktioniert fast perfekt, lediglich eine etwas zu hohe Temperatur der Spritzdüse, zusammen mit der relativ hoch eingestellten Geschwindigkeit, sorgen für leichte äußerliche Deformierungen der mittleren Führungsdrahtaufnahme am Dreieckpaneel, die aber keine weiteren funktionellen Auswirkungen haben.

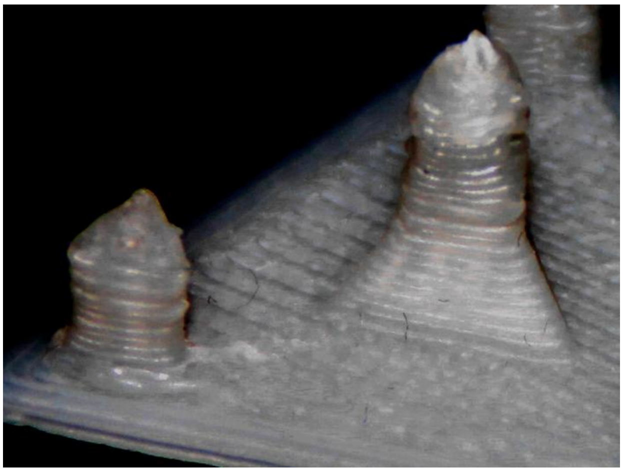


Abbildung 12: Druckergebnis des zweiten Entwurfs

In dem stark vergrößerten Ausschnitt des gedruckten Display- Elementes sind Verformungen an der Spitze der mittleren Halterung sichtbar.

Die Verformungen und besonders das Problem, dass diese Konstruktion mit der mittigen Halterung die Erweiterung mit Braille Displayelementen verhindert, sind ausschlaggebend, diesen Entwurf, obwohl er prinzipiell funktionsfähig ist, erneuert zu überarbeiten.

Dritter Entwurf: fehlender Stützdraht

Der dritte Entwurf besteht aus den gleichen Elementen wie der zweite, es wird jedoch die mittlere Halterung weggelassen. Dadurch soll neben der erwähnten Möglichkeit zur Erweiterung um Braille-Elemente auch die mechanische Belastung der Gesamtkonstruktion verringert werden.

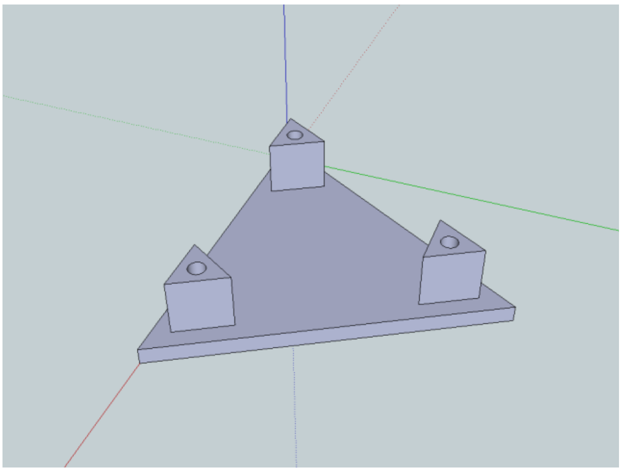


Abbildung 13: dritter Entwurf ohne Turmstruktur CAD Vorschau

Dieser Entwurf besteht lediglich aus der dreieckigen Grundplatte, die seit dem Erstentwurf in ihren Ausmaßen unverändert ist, und den 3 Halterungen für die Kunststoffleiter.

Dieser Entwurf entspricht im Grunde dem final verwendeten Entwurf, doch es stellt sich heraus, dass die Lochdurchmesser an den Halterungen für die Kunststoffleiter von 1,2 mm auf 1,4 mm vergrößert werden müssen. Dies führt zu neuen Problemen im Druck, da die Materialstärke an den Halterungen zu gering für den verwendeten Druckkopf bzw. die verwendete Spritzdüse ist. Hierdurch werden die einzelnen Halterungen statt wie in der CAD Vorschau als stabile dreieckige „Türme“ als jeweils 3 sehr leicht brechende Teilhalterungen gedruckt.

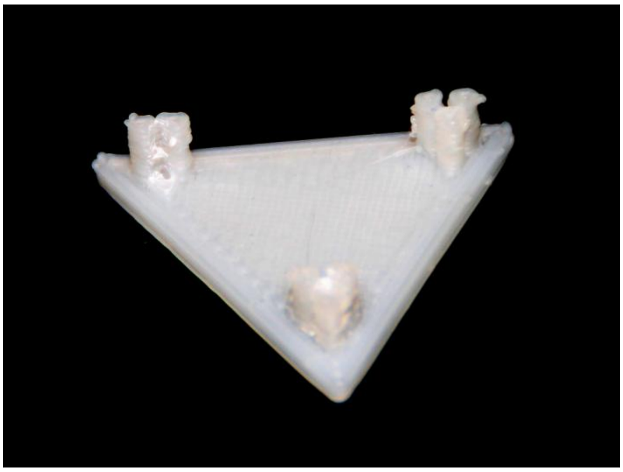


Abbildung 14: Lückenbildung

Die Lückenbildung beim Druck der Halterungen ist an der rechten hinteren Seite sichtbar und teilweise bereits, wie an der linken hinteren Seite, gebrochen.

Vierter Entwurf: finaler Entwurf

Daher wird dieser Entwurf noch ein viertes und letztes Mal überarbeitet, indem die Halterungen massiver gebaut und weiter zur Mitte der Paneele gerückt werden, um die Bewegungsfreiheit auch bei starken Oberflächenverformungen des Displays weiterhin erhalten zu können. Hierfür müssen die Verteiler zur Steuerung ebenfalls überarbeitet werden. So werden die Löcher zur Aufnahme der

Verbindungsleiter und Steuerdrähte weiter nach außen gelegt und die Verteilerelemente damit deutlich größer.

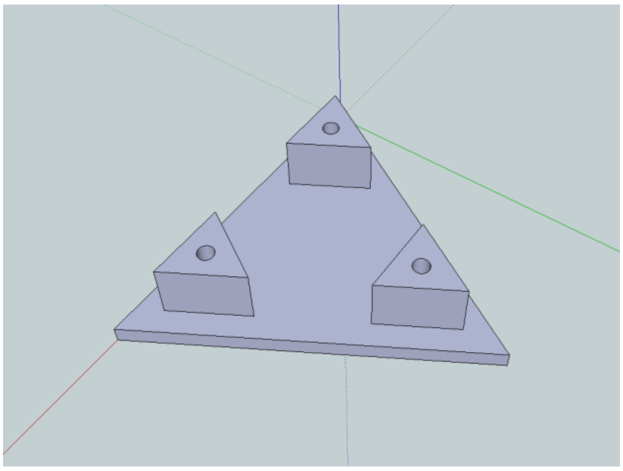


Abbildung 15: Entwurf des finalen Dreieckpaneels mit verstärkten Halterungen CAD Vorschau

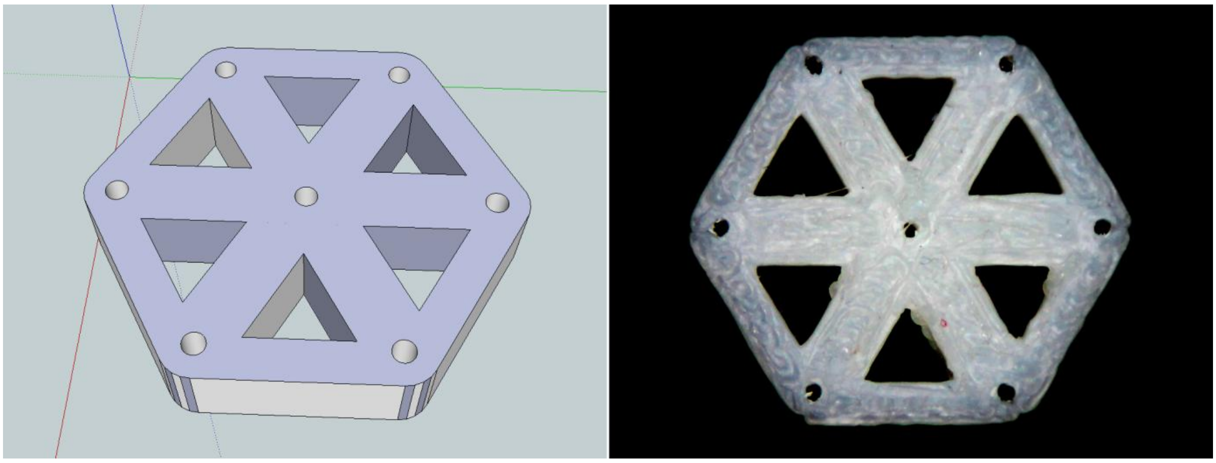


Abbildung 16: Vergleich der Verteilerelemente CAD Vorschau / 3D Druck

Der Vergleich des fertigen Verteilerelementes gegenüber dem CAD Modell zeigt, dass das Druckverfahren bereits sehr gut funktioniert, so ist die Form des Verteilerelementes bereits sehr exakt. Lediglich die Oberfläche der letzten Druckschicht ist nicht frei von optischen Fehlern. Dennoch kann dieses Element einwandfrei für das Display genutzt werden.

9.3. Der Bau des Displays

Nachdem die einzelnen Elemente erfolgreich gedruckt sind, kann mit dem Aufbau des Displays begonnen werden. Dafür wird eine Metallgewindestange in vier Stücke mit einer Länge von jeweils 125 mm geteilt und an den Enden sauber abgeschliffen. Zur Vorbereitungen der bereits erwähnten, und ebenfalls mit Hilfe des 3D-Druckers hergestellten Grundplatte (siehe Kapitel 9.1.; Die Fertigung der Display Elemente) für die Dreieckfrontpaneele wird in jede Ecke ein 3,5 mm dieser Grundplatte großes Loch gebohrt und die Platte mit einem 3mm dicken, feinporigem Schaumstoff beklebt. Dieser hat die Aufgabe, bei dem fertigen Display die Geräusche zu mindern und die Drähte, die die Verteilerelemente mit den Servomotoren verbinden, in ihrer Führung zu stützen.

Zum Schutz vor Verschmutzung und Beschädigungen bei Transporten wird der komplette Display-Demonstrator in einer Transportbox verschraubt.

Als letzter Schritt für die Vorbereitung des Displaygrundstatives werden daher in den Deckel dieser Transportbox, im Abstand der Löcher in der Halterplatte, ebenfalls Löcher gebohrt.

Nun werden die Gewindestangen mit Hilfe von M3 Muttern und dazu passenden Unterlegscheiben an der Halterplatte und danach an dem Deckel verschraubt.

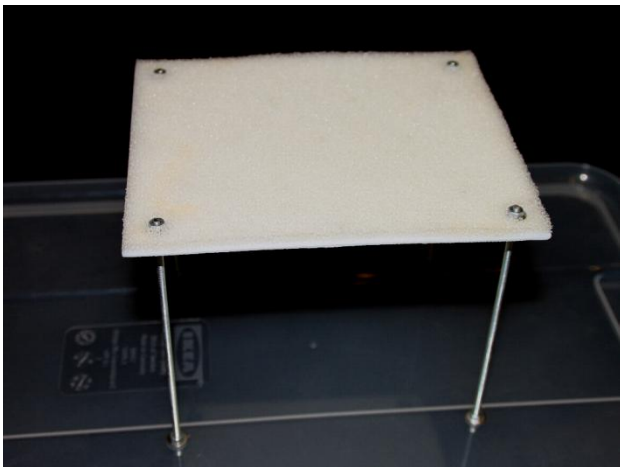


Abbildung 17: Fertiges Displaygrundstativ auf Transportboxdeckel

Als nächster Schritt werden die Stangen zur Führung der Verteilerelemente geschnitten. Dafür wird Federstahldraht gewählt, da dieser besonders hohen Belastungen sehr lange wiederstehen kann, ohne sich zu verbiegen. Zudem ist Federstahldraht auch als absolut gerader gerichteter Draht in einer Stärke von 0,8 mm im Handel leicht erhältlich.

Um den Draht auf eine exakt gleiche Länge von 70 mm zuzuschneiden, wird dieser zunächst mit ca. 2 - 3 mm Überlänge grob vorgeschnitten. Danach schleift man ihn mit Hilfe eines Schleifbocks und einem Vergleichsdraht, der zuvor auf die exakt richtige Länge gebracht wird, auf die 70 mm Ziellänge herunter. Dabei gilt es aufzupassen, das die Drähte nicht zu warm werden und miteinander verschmelzen.

Die Kunststoffleiter, die als flexible Brücke zwischen den Verteilerelementen und den Dreieckpaneelen dienen, werden lediglich mit Hilfe einer Schere auf Länge gebracht, da sie bei anderen Schneidegeräten, wie z.B. Papierschneidern, die auf eine bestimmte Länge einstellbar sind, wegrutschen oder zu dünn sind. Das hat zur Folge, dass diese Brücken auch deutlich am ungenausten in ihrer Länge sind und eine Abweichung von +/- 0,5 mm aufweisen. Das stellt bei einer angepeilten Länge von 30 mm eine vergleichsweise hohe Ungenauigkeit dar.

Um die Verteilerelemente mit den Dreieckpaneelen verbinden zu können, werden zuerst die Drähte mit Hilfe eines Kraftklebers in die Löcher in der Mitte der Verteilerelemente geklebt.

Sobald diese Klebestellen getrocknet und ausgehärtet sind, werden die flexiblen Kunststoffleiter mit Hilfe eines Kunststoffklebers in die äußeren Löcher geklebt. Wenn auch diese Klebestellen ausgehärtet sind werden die Elemente in das Display- Grundstativ gesteckt und parallel zueinander ausgerichtet.

In die Löcher der Dreieckpaneele wird nun jeweils eine kleine Menge des Kunststoffklebstoffs eingefüllt. Dann werden die Paneele jeweils in die Mitte zwischen die Verteilerelemente positioniert und auf die Enden der Kunststoffleiter gesteckt.

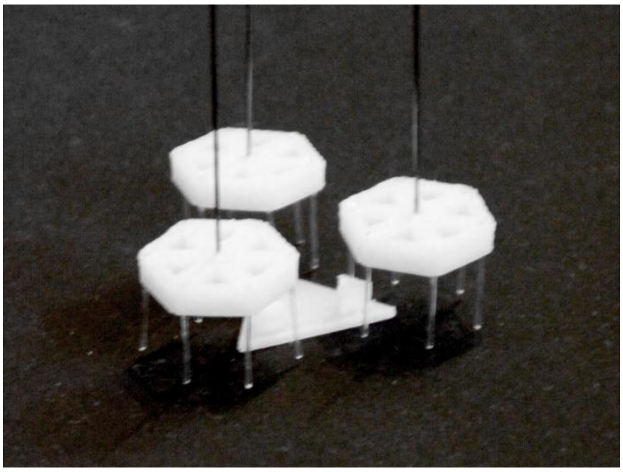


Abbildung 18: einzelnes Dreieckspaneel mit komplettem mechanischem Aufbau

9.4. Die aktiven Steuerelemente

Als aktive Elemente, die die Verteilerelemente anheben bzw. senken, werden als erste Option Schrittmotoren und als zweite Option Servomotoren aus dem Modellbau angedacht. Beide Arten der Motoren benötigen im Gegensatz zu anderen Antrieben, z.B. zu normalen Gleichstrommotoren, keinerlei zusätzliche Messeinrichtung zur Regelung der Position. Damit sinkt der Aufwand der Ansteuerung erheblich und ist zugleich zuverlässiger.

Da Schrittmotoren mit einem Lineargetriebe in der Lage sind, wesentlich höhere Höhenunterschiede, als Servos mit Standard-Ruderhörnen zu erzeugen, wird diese Lösung im Ausgangspunkt favorisiert. Diese Antriebsart muss aber aufgrund zu hoher Kosten pro Einheit und der hohen Anzahl, die für den Demonstrator benötigt wird, verworfen werden. Die preisgünstigeren Digitalservos, die als Ersatz genommen werden, Verfügen über eine hohe Genauigkeit und eine hohe Wiederholgenauigkeit. Darüber hinaus sind sie klein genug, um in einer Schicht komplett unter der Grundplatte des Displays verbaut werden zu können.

Zur Ansteuerung der Servos wird ein Hardwarecontroller benötigt. Als erster Versuch wird der myDAQ von National Instruments verwendet. Dieser ist eine speziell auf die Software LabVIEW abgestimmte Mess- und Steuereinheit und bietet neben analogen auch digitale Ein- und Ausgänge. Diese sollen zur Ansteuerung der Servos mittels Pulsweitenmodulation (siehe Kapitel 8.3., Die Pulsweitenmodulation) genutzt werden.

Nach einer Recherche muss diese Ansteuerungsoption aber verworfen werden, da der myDAQ nur einen System Timing Controller besitzt und darum nicht in der Lage ist, mehr als einen Servo zugleich zu kontrollieren.

Aus diesem Grund wird nun ein SD21 Servocontroller verwendet. Dieser ist nicht speziell auf LabVIEW ausgelegt, bietet aber die Möglichkeit, bis zu 21 Servos gleichzeitig zu kontrollieren. Um den Controller verwenden zu können, benötigt man noch zusätzlich einen „USB zu I²C“ Adapter Modul als Schnittstelle.

Nachdem auch sichergestellt ist, welche Servos verwendet werden und damit die Abmessungen feststehen, wird als letztes zu produzierendes Element eine Halterung für diese entworfen. Diese Halterung besteht aus zwei Platten. Die Servos werden zwischen zwei dieser Platten eingeklemmt, um sicher zu halten. Daher müssen die Platten mit Aussparungen für die Kabel und die an den Enden der Servos angebrachten, im modellbauüblichen Halterungen versehen werden. Es muss ebenfalls genügend Platz gelassen werden, damit die am Servo angebrachten Ruderhörner bei Bewegungen nach oben und unten nicht mit den Halterplatten kollidieren. Um dennoch eine effektive Platzausnutzung und einen geringen Kunststoffverbrauch zu erreichen, wird eine ungleichmäßige, rein funktionelle Form gewählt. Da durch das Einklemmen eine Bewegung der Servos selbst bereits unmöglich ist, reicht es aus, diese beim Einbau mit doppelseitigem Klebeband gegen Verrutschen, zu fixieren.

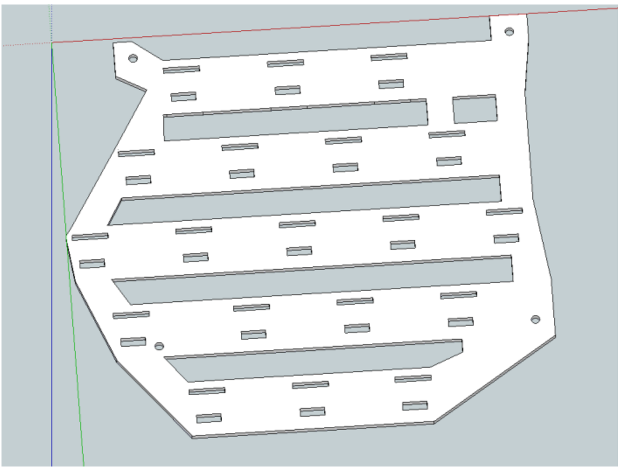


Abbildung 19: Die Servo- Halterplatte CAD Vorschau

Die in den Halterungen befindlichen Servos werden unterhalb der Grundplatte verschraubt, so dass die in den Ruderhörnernern befestigten Drähte bei der „Minimalstellung“ ca. 5 mm oberhalb der Löcher der Grundplatte enden. Hierbei treten allerdings Fehler auf, die darin bestehen, dass die Ruderhörner von insgesamt 3 Servos bei Minimalstellung mit der unteren Servohalterplatte in Berührung kommen und blockieren. Daher werden die Aussparungen in der Halterplatte mittels einer Minifräse an diesen drei Stellen vergrößert.

Als letzter Schritt des mechanischen Aufbaus des Demonstrators werden die Kabel der 19 Servos nach Reihen sortiert und gebündelt und mit Kabelbindern ebenfalls am Transportboxdeckel befestigt. Dadurch können diese nicht mit den Ruderhörnen in Berührung kommen und diese blockieren. Der ausgewählte Servocontroller wird ebenfalls am Deckel verschraubt und alle Servostecker und das USB zu I²C- Adaptermodul mit dem Controller verbunden. Nach der Beschriftung der Servos und dem Anlöten des Stromkabels zur Versorgung des Controllers und der Servos an einen USB- Stecker, kann das Display mit einem normalen USB-Netzteil betrieben werden. Damit ist der Bau der Hardware abgeschlossen.

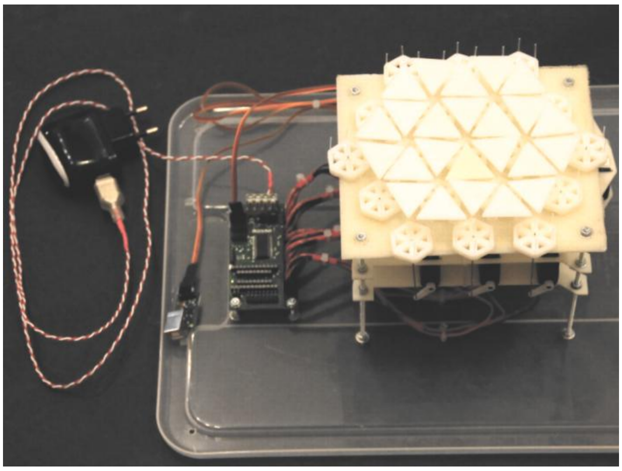


Abbildung 20: Der fertige Display Demonstrator

10. Die Programmierung

10.1. Die Programmiersprache LabVIEW

LabVIEW steht für **Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench und ist eine grafische IDE für die Programmiersprache „G“, die zur 4. Generation der Programmiersprachen gehört.

Das Programmieren in LabVIEW erfolgt mittels zwei verschiedener Oberflächen, dem Frontpanel und dem Blockdiagramm. Im Frontpanel werden grafische Elemente als Benutzerschnittstelle, wie beispielsweise Schalter, Eingabefelder und Anzeigen gesetzt. Im Blockdiagramm wird mittels sogenannten Funktionsblöcken der Programmcode definiert. Dafür stehen vorgefertigte Funktionsblöcke in dem sogenannten Elementefenster zur Auswahl. [PRLA]

Die Programme können jederzeit getestet werden, wobei anzumerken ist, dass es sich bei dem ausgeführten Code nicht um einen interpretierten Code handelt, sondern um einen kompilierten. Das hat zur Folge, dass die Laufzeiten durchaus vergleichbar mit anderen Hochsprachen sind.

Das Speichern des erstellten Programmcodes erfolgt dabei in sogenannten **v**irtual **i**nstruments (.vi). Dabei können auch Programmteile zu einem neuen Funktionsblock, einem SubVI, zusammengelegt werden. Eine Besonderheit dieses Aufbaus ist, neben der beliebig tiefgehenden Komplexität eines Programms durch diverse Verschachtelungen von SubVIs, dass jeder Funktionsblock, mit Ausnahme von Primitives (z.B. mathematischen Grundfunktionen wie addieren, subtrahieren, usw.), lediglich ein SubVI ist. Das hat wiederum zur Folge, dass jedes SubVI und somit auch jeder Funktionsblock einzeln unabhängig von Hauptprogramm ausführbar und auch veränderbar ist.

Die Stärken von LabVIEW liegen besonders in der Aufnahme und Verarbeitung von Messdaten, der Automatisierung und der einfachen und schnellen Prototypen-Implementierung. Durch die Programmierung im Blockdiagramm mittels Symbole ist eine gute Übersichtsmöglichkeit gegeben, die es in herkömmlichen textbasieren Quellcodes auf Grund der Textlänge nicht gibt.

Die Schwächen von LabVIEW liegen in der begrenzten Möglichkeit der Funktionsblöcke, da diese Einschränkungen in der Veränderbarkeit gegenüber herkömmlichen Sprachen und deren Funktionen unterliegen. So liegt auch eine Schwäche in der begrenzten Auswahl an Libraries für externe Hardware vor. Es gibt zwar Messkarten und Steuereinheiten von National Instruments, diese sind aber vergleichsweise teuer und bieten nicht immer alle Möglichkeiten der Ansteuerungen, die für spezifische Aufgaben benötigt werden(z.B. im vorliegenden Fall die Kontrolle mehrerer Servos mittels des myDAQ). Auch bereits vorhandene Libraries für Hardware, wie z.B. Arduino-Komponenten [ARDU], sind meist nur auf Grundfunktionen beschränkt und stoßen schnell an ihre Grenzen. [PRLA]

10.2. Das Treiberprogramm

Da der Treiber wie erwähnt in LabVIEW geschrieben wird und diese Programmierung wie beschrieben rein grafisch erfolgt, werden Quellcodes nicht wie in Programmiersprachen der 2. Generation (z.B. Borland Assembler) oder 3. Generation (z.B. C, C++, Java, etc.) mit Inline-Kommentaren beschrieben, sondern, um die Übersichtlichkeit der Quellcodes weiterhin gewährleisten zu können, mit numerischen Indices in geschweiften Klammern versehen. Diese werden nach den Abbildungen des jeweiligen Quellcodes ausführlich kommentiert, da Kurzkommentare hier ebenfalls eher für Verwirrung sorgen.

Um einen Überblick über die grobe Funktionalität des Treiberprogramms zu bekommen, wird im ersten Schritt ein Programmablaufplan erstellt. Dieser zeigt alle Funktionen, die das Programm haben soll und die Abläufe, die dafür benötigt werden, in einer groben Verlaufsstruktur.

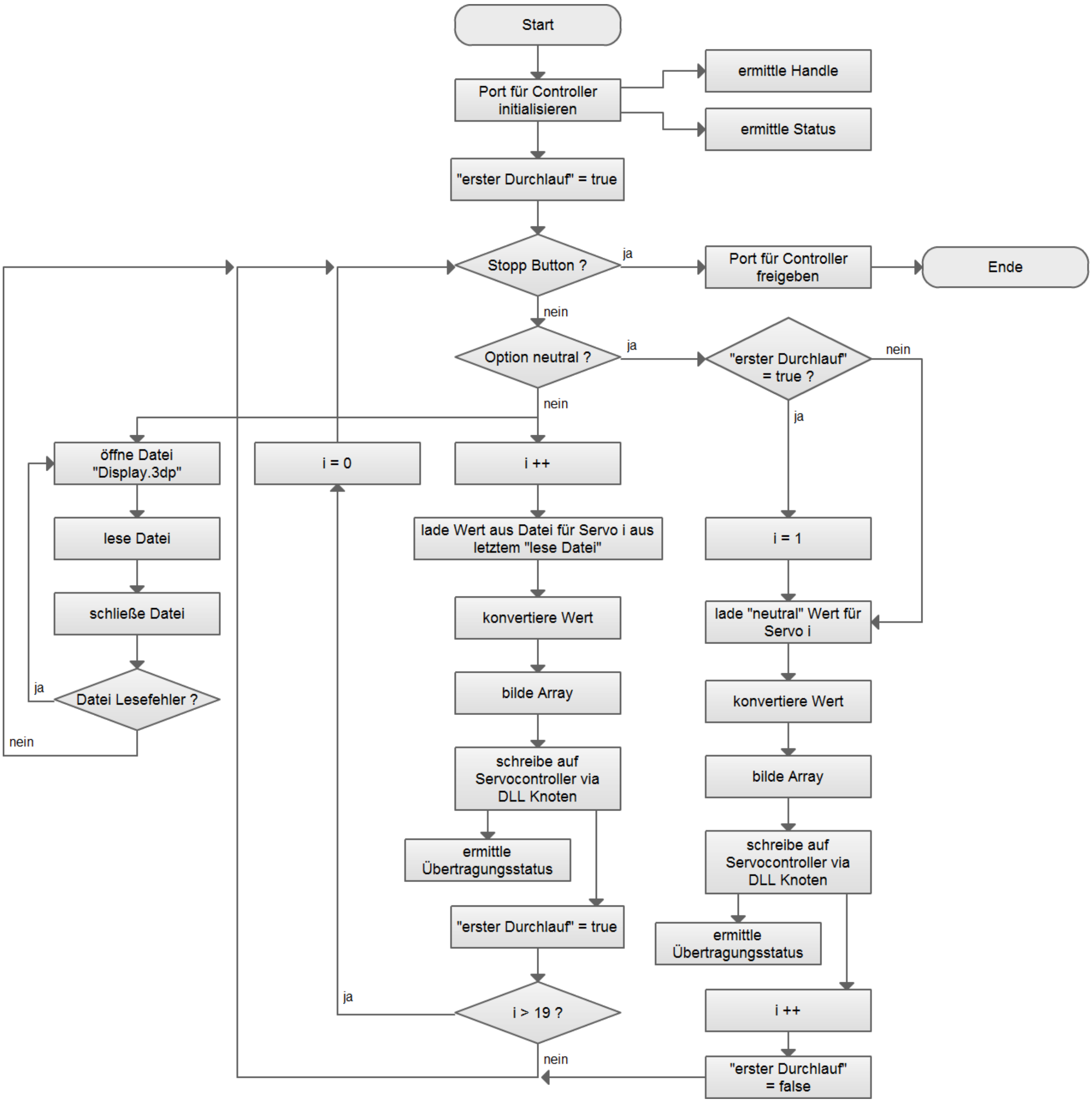


Abbildung 21: Programmablaufplan des Treiberprogramms

Nach der Initialisierung besitzt das Treiberprogramm zwei Optionen für den Dauerbetrieb, nämlich die Funktion „neutral“ und die Funktion „extern“, wobei letztere im Ablaufplan nicht namentlich genannt wird, aber als Gegenstück zur „neutral“-Funktion einen Verlaufszweig besitzt.

Dies hat den Hintergrund, dass der Verwendete Servocontroller „SD21“ alle Servos in eine vordefinierte Stellung bringt, solange er noch keine externen Steuersignale erhält. Gleichzeitig kann der SD21 aber nicht die Servos ansteuern, bzw. Steuersignale auswerten, wenn er Signale erhält, ohne zuvor an den Versorgungsstrom angeschlossen worden zu sein. Somit muss immer erst diese vordefinierte „neutrale“ Stellung erreicht werden, bevor das Display verwendet werden kann. Daher werden Festwerte für jeden Servo in das Treiberprogramm eingegeben, die bei der Auswahl der

Option „neutral“ angefahren werden sollen, um bei jedem Start des Display-Demonstrators die Verzögerungszeit gering zu halten. Die zweite Option besteht in der Ansteuerung des Displays mittels einer Datei, dies ist für externe Programme gedacht. Dabei sind die Werte für die Servo-Stellungen dementsprechend nicht vordefiniert, sondern in einem Bereich von 0 (Minimalstellung) bis 10 (Maximalstellung) frei wählbar.

Das Programm kann mit einem Klick auf einen Stopp Button jederzeit beendet werden, dabei wird die Verbindung zum Servocontroller beendet und der verwendete Port wieder freigegeben.

Insgesamt besteht das Treiberprogramm aus 3 Teilen:

- Die Initialisierung des Controllers
- Das Hauptprogramm, bestehend aus dem Schreibevorgang auf den Servocontroller und den erwähnten Optionen
- Dem Beenden der Verbindung zum Controller und der Freigabe des Ports

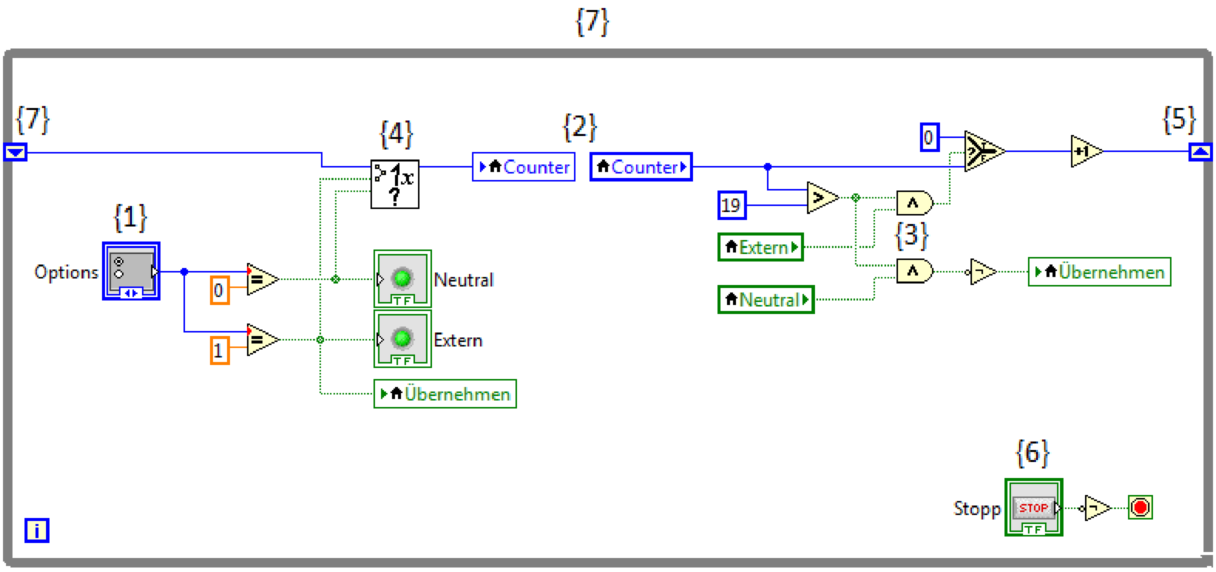
Da der Schreibeprozess der Hauptbestandteil ist wird bei der folgenden Programmanalyse darauf ein besonderes Augenmerk gelegt.

10.3. Optionswahl und Servoindex

Zur Eingabe von Daten für die Stellung der Display-Servos sollen zwei Möglichkeiten bestehen. Die Erste ist ein fest eingestellter Wert für jeden einzelnen Servo für die bereits erwähnte Neutralstellung, der zweite Weg geht über eine externe Datei, damit ist es auch unabhängigen Programmen möglich, das Display als Anzeige zu verwenden. Um den über einen USB-Port verbundenen Controller korrekt mit Werten für alle Servos zu versorgen, ist es notwendig, diese Werte seriell zu senden. Zu diesem Zweck wird ein Index benötigt, der immer auf den Servo verweist, der aktuell mit neuen Stellungswerten versorgt werden soll. Dazu kommt eine Logik zur Auswertung der gewählten Optionen, da diese auch Auswirkungen auf das Verhalten des Zählindex der Servos haben. Wenn die Option der Neutralstellung gewählt wurde, sollen die entsprechenden Servo-Stellungswerte nur einmalig geschrieben werden und die Kommunikation mit dem Servocontroller danach inaktiv sein, um unnötige Schreibvorgänge zu verhindern. Zudem soll dieser Schreibvorgang immer bei dem allerersten Servo (Servo Nummer A0) beginnen. Damit soll sichergestellt werden, dass nach genau einem Durchlauf alle Servos die gewünschte Stellung haben. Beim Schreiben der Werte für die Servo- Stellungen aus der externen Datei spielt die Reihenfolge der Servo- Abarbeitung keine Rolle. Daher wird hier immer mit dem Servo zum aktuellen Zählindex fortgefahren, d.h. wenn Beispielsweise noch während der Ausführung der Neutralstellungsoption, die Option zum Ansteuerung über die externe Datei gewählt wird, springt der Servoindex nicht zurück an den Servo Nummer A0, sondern setzt mit dem nächsten Servo auf den der Index verweist fort (z.B. Servo Nummer C3). Es wird auch permanent der aktuelle Wert jedes Servos neu geladen und immer neu auf den Controller geschrieben, um jederzeit sichergehen zu können, dass die aktuellen Servo-Stellungen auch dem Soll entsprechen.

Für die Optionsauswahl, zwischen der Neutralstellung und der Annahme externer Werte, wird ein boolsches Bedienelement mit einer Radiobutton- Funktion erstellt. Damit wird sichergestellt, dass immer nur eine Option aktiv ist. Gleichzeitig nimmt das Element im Blockdiagramm einen Wert für die gewählte Option an (0 basiert).

Abbildung 22: Options Radiobutton



Quellcode 01: Servoindex und Optionswahl

Das mit dem Erstellen des Options- Frontpanelelements automatisch hinzugefügte Blockelement Options {1} setzt, je nach gewählter Option, nach dem Vergleich mit den Werten 0 und 1 die boolschen Variablen „Neutral“ oder „Extern“ auf true bzw. false. Dies dient vor allem der Übersichtlichkeit, da man diese Variablen auch durch eine Wiederholung des vorhergehenden Vergleichs, ob die Variable Options den Wert 0 oder 1 hat, ersetzen könnte.

Die Variable Counter {2} ist die Variable, die für den zuvor erwähnten Zählindex der 19 Servos verwendet wird. Je nach gewählter Option {3} wird hierbei entschieden, ob nach dem 19. Wert der Index wieder auf 0 gesetzt und entsprechend wieder der erste Servo mit Werten beschrieben werden soll, oder die Wertübernahme für die Servos deaktiviert wird.

Bei der Auswahl der Option für die Neutralstellung wird der Wert von Counter beim ersten Durchlauf der Sequenz immer auf 0 gesetzt, um danach immer den ersten Servo mit einem Stellungswert zu versorgen. Dafür wurde eine entsprechende Logik in dem SubVI „Options(SubVI)“ {4} erstellt. Diese Logik vergleicht mit Hilfe einer Rückkopplung ob die Optionsauswahl sich geändert hat und nun den Wert 1 (Option: „extern“) hat. Wenn dies zutrifft, wird der Zählindex nicht verändert. (siehe Kapitel: 15.3. SubVI’s Übersicht; Options(SubVI).vi) Um den letzten Wert von Counter nicht bei jedem neuen Schleifendurchlauf zu resetten, muss dieser durch ein Schieberegister {5} „weitergereicht“ werden.

Diese Indexschleife soll so lange ausgeführt werden, bis der Stopp Button {6} angeklickt wird, dieser wurde im Schaltverhalten auf „Beim Drücken Schalten“ abgeändert, sodass der Abbruchbefehl der while-Schleife {7} mit negativer Logik erfolgt. Durch Klick auf diesen Button wird die Schleife abgebrochen und automatisch die Controllerverbindung beendet und der Port freigegeben.

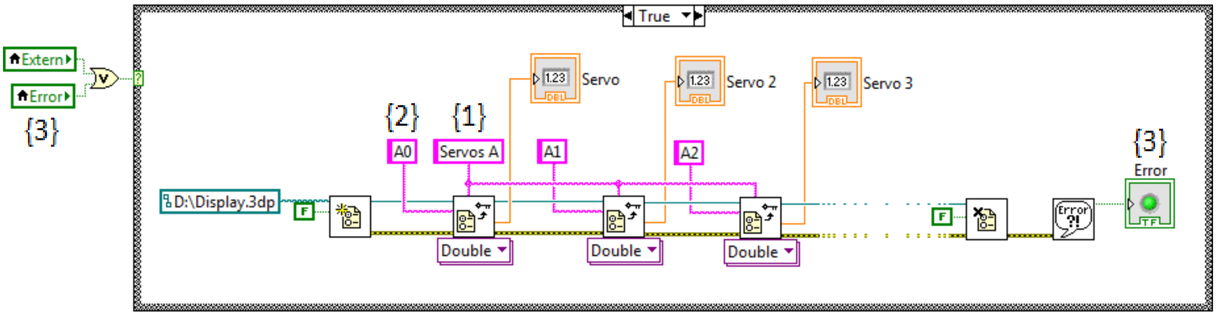
10.4. Externe Dateieinbindung

Für die Option der externen Displayansteuerung muss, wie bereits erwähnt, permanent eine Datei mit den aktuell anzufahrenden Werten vorhanden sein. Hierfür gibt es unter LabVIEW zwei unterstützte Dateiformate. Das erste Format ist das auch von Excel verwendete .CSV Format, mit dem besonders Messwerte tabellarisch erfasst und schnell ausgelesen werden können. Das zweite unterstützte Format ist die Konfigurationsdatei. Dieses Format hat keine spezifische Dateiendung und kann daher frei gewählt werden. Das hat den Vorteil, dass unverwechselbare Dateiendungen verwendet werden die nicht versehentlich gelöscht oder von anderen Programmen überschrieben werden können. Als definierter Ablageort, der immer funktioniert, wird hierbei die Festplattenpartition D: gewählt, da diese keine Administratorenrechte benötigt. Die Administratorenrechtefunktion bei Partition C: ist unter neueren Windows Versionen wie Windows Vista, Windows 7 und Windows 8 ein häufig auftretendes Problem und sorgt dafür, dass die notwendige externe Datei häufig nicht korrekt geschrieben werden kann, da Windows-Warnmeldungen ausgegeben werden, die vor dem Speichern bestätigt werden müssen.

Die dateiinterne Struktur hingegen ist bereits lange von Microsoft standardisiert und führt zu keinen Problemen. Sie folgt dem simplen Schema:

Schema:	**Verwendungsbeispiel:**	**Kommentar:**
[„Sektion 0“]	[Servos A]	Servo-Reihe A
„Schlüssel 0“=Wert	A0=1	Servo-Nummer A0
„Schlüssel 1“=Wert	A1=5	Servo-Nummer A1
„Schlüssel 2“=Wert	A2=4	Servo-Nummer A2
[„Sektion 1“]	[Servos B]	Servo-Reihe B
…	…	…

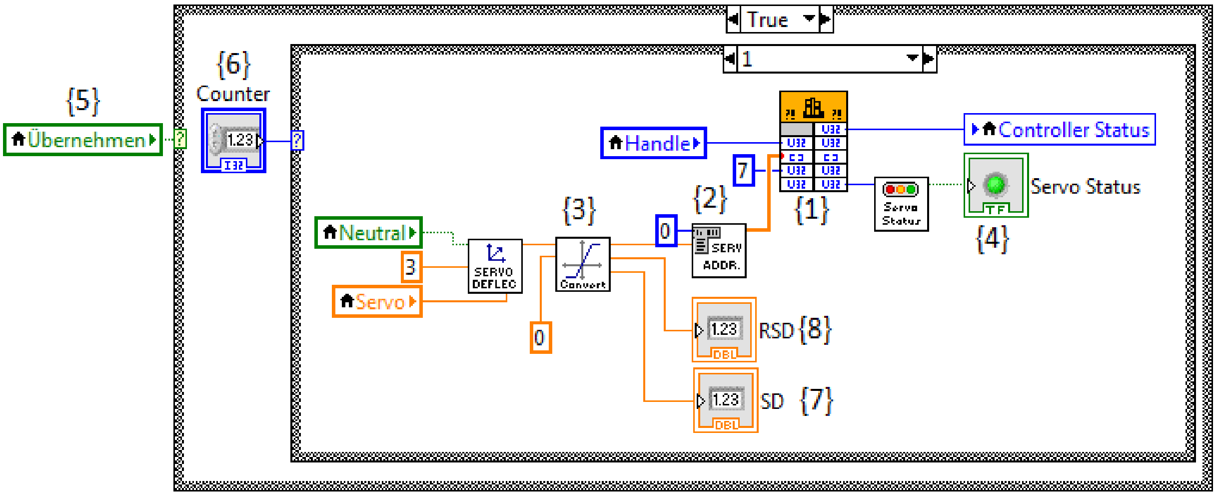
Damit ist es im Gegensatz zum Excel-Dateiformat möglich, besonders einfach und übersichtlich die einzelnen Servos in Gruppen {1} und Einzelelemente {2} zu unterteilen. Auf Grund der angesprochenen Unverwechselbarkeit und Übersichtlichkeit wird dieses Dateiformat gewählt. Damit der Nachteil der langsameren Auslesegeschwindigkeit dieses Dateiformates nicht die Stellgeschwindigkeit des Displays mindert, wird das Auslesen in eine gekapselte while- Schleife gesetzt. Diese wird parallel zum Rest des Programms abgearbeitet und kann daher schneller als der Schreibeprozess auf den Controller abgearbeitet werden, da dieser „gebremst“ werden muss (siehe Kapitel 10.5., Die Servocontroller- Ansteuerung). Das hat auch den Vorteil, dass bei einem eventuellen Dateilesefehler dieser schnell und einfach behandelt werden kann, indem die Datei, nach Setzen der Variable Error auf true {3}, erneut ausgelesen wird.



Quellcode 02: Lesen der Externen Datei (Ausschnitt)

10.5. Die Servocontroller- Ansteuerung

Das Kernstück des Treiberprogramms bildet der Schreibeprozess auf den Servocontroller.



Quellcode 03: Schreiben auf den Servocontroller (Übersicht)

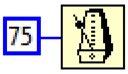
Um diesen Vorgang des Schreibens zu ermöglichen, benötigt man die Standard-Windows-Datei FTD2XX.dll. Diese Library erlaubt die Kommunikation mit dem USB Port und damit mit dem „USB to I²C“ Adaptermodul. Das ist wiederum an den Servocontroller angeschlossen. Diese .DLL Datei kann mit Hilfe der DLL-Knoten Funktion {1} in LabVIEW eingebunden werden und muss dann in der Konfiguration als FT_Write Funktion deklariert werden. Damit ist es, vorausgesetzt, dass der vorhergehende Initialisierungsprozess erfolgreich war, nun möglich, Datenbyte-Arrays mit allen benötigten Nutzdaten an den Controller zu senden. Diesem DLL-Knoten muss jetzt lediglich der Wert vom Handle von der vorhergehenden Initialisierung des Ports (siehe Quellcode 6, FT_Open) übergeben werden. Damit ist sichergestellt, dass das Programm Schreibrechte für den Port hat.

Hinzu kommt das Array {2} zur eigentlichen Ansteuerung des jeweiligen Servos. Die einzelnen Arrayelemente stellen dabei die drei wichtigsten Daten dar, die Adresse des Servos, die Stellgeschwindigkeit und die Auslenkung (näheres siehe Quellcode 14, ServoAddress(SubVI)).

Allerdings müssen alle Auslenkungswerte vor dem Schreiben konvertiert werden, da die Servos „auf dem Kopf stehend“ eingebaut wurden und einen internen Wertebereich von 7 (Minimalstellung) bis 3 (Maximalstellung) haben, der zur Verwendung für Drittanwender schlecht geeignet ist. Daher wird der Wert mittels der linearen Funktion $f(x) = -\frac{4}{10}x + 7$ auf die Werte von 0 (Minimalstellung) bis 10 (Maximalstellung) konvertiert {3} (siehe Quellcode 13: ServoDeflectionConverter(SubVI). Die Ausgabe des Servostatus {4} ist lediglich die Kontrolle, ob alle 7 Datenbytes des Arrays korrekt geschrieben wurden.

Hier greifen auch die boolsche Variable „Übernehmen“ {5} und der Zählindex mit der Variable Counter {6}. Wenn einer der beiden Variablen einen „falschen“ Wert annimmt, also „Übernehmen“ = false oder „Counter“ = 0 bzw. > 19, wird der Schreibeprozess, wie bereits bei der Erläuterung der „neutral“ Option beschrieben, nicht weiter ausgeführt.

Da der verwendete Servocontroller zu schnell gelieferte Werte nicht verarbeiten kann, muss die Ausführung des Schreibevorgangs verlangsamt werden. Dafür wird ein Timer-Objekt verwendet, das immer bis zu einem Vielfachen eines voreingestellten Wertes in Millisekunden wartet, bevor der Schleifeninhalt erneuert ausgeführt wird. Nach mehreren Versuchen hat sich hierbei ein Wert von 75ms als zuverlässig und dennoch ausreichend schnell für Auslenkungsänderungen erwiesen.



Quellcode 04: Timerfunktion: Bis zum nächsten Vielfachen von ms warten

Im letzten Schritt werden die erstellten Elemente des Blockdiagramms, die auf dem Frontpanel auch als Anzeigeelement dienen, möglichst übersichtlich angeordnet. Zu diesem Zweck werden die Anzeigen der Servo- Auslenkungswerte (SD) {7}, der realen Werte nach der Konvertierung (RSD) {8} und der Status der geschriebenen Bytes aller 19 Servos via Drag and Drop Funktion in ein Registerkarten-Objekt verschoben. Dieses Objekt vereinfacht es die Werte übersichtlich anzuordnen. Die Aufteilung auf die einzelnen Karten wird automatisch unterstützt.

Die überflüssigen Anzeigeelemente, wie die Variable Übernahme oder die Neutral bzw. Extern Variable, werden ausgeblendet.

Handle: 19772693

Controller Status: 0

○ Neutral ● Extern

SD	RSD	Servo Status
0,5	6,8	●
1,75	6,3	●
2,78	5,888	●
5,1	4,96	●
4,6	5,16	●
3,4	5,64	●
8	3,8	●
9,2	3,32	●

Abbildung 23: Vollständiges Frontpanel

10.6. Das Testprogramm

Um das Display einfach und komfortabel testen zu können, ohne die Werte der Servoauslenkungen per Hand in die Externe Datei eingeben zu müssen, wird ein externes und vollständig unabhängiges Testprogramm geschrieben.

Dieses Programm besteht lediglich aus einer Aufführung aller einzelnen Servos,

mit jeweils einem „+“ Button und einem „-“ Button, um den Auslenkungswert um jeweils 1 zu erhöhen oder zu senken. Dazu gibt es noch eine Auswahl von zehn vordefinierten Stellungen, die man mit nur einem Mausklick anfahren kann. Um dieses Testprogramm zu schreiben, wurde die Programmierumgebung Multimedia Fusion 2 Developer (MMF2) eingesetzt. Diese Programmierumgebung ist eine Crossplattform für diverse Betriebssysteme und Geräte (z.B. Windows PC, Android, iPhone, Flash, usw.) mit einer Spezialisierung auf mediale Inhalte wie Videos, Audiodateien und Animationen.

Multimedia Fusion 2 ist auch eine größtenteils grafische „Programmiersprache“ und teilt sich in vier Hauptfenster auf. Das erste Fenster, der sogenannte Storyboard Editor, ist für komplexere Programme mit mehreren Frames vorgesehen, um diese zu verwalten. Da das Testprogramm aber lediglich einen Frame besitzt, ist dieses Menü hier eher irrelevant. Das zweite Fenster ist der Frame Editor und mit dem Frontpanel von LabVIEW vergleichbar. Es dient hauptsächlich der Gestaltung der grafischen Oberfläche. Hier können Animationen, Buttons, Hintergründe, usw. erstellt und festgelegt werden. Darüber hinaus kann man hier auch Extensions (diese entsprechen externen Librarys) laden. Diese werden durch Symbole dargestellt, aber im Programm später nicht angezeigt.

Da dieses Testprogramm lediglich Werte in eine Datei (D:/Display.3dp) schreiben und zur Übersicht selbst anzeigen soll, wird hierfür nur die Extension „INI“ benötigt. Diese Extension bietet die Möglichkeit, Textdateien im Microsoft Standard-Konfigurationsdateienformat auf die Festplatte zu schreiben und auszulesen (siehe Kapitel 10.4., Externe Dateneinbindung). Da die in die Datei geschriebenen Auslenkungswerte im Testprogramm auf zwei verschiedene Arten angezeigt werden sollen, müssen entsprechend für jeden Servo zwei Zählerobjekte erstellt werden. Der eine Zähler wird dabei als numerische Anzeige konfiguriert und der Zweite als Balkenanzeige. Damit soll auch grafisch angedeutet werden, wie das Abbild auf dem Display aussehen muss. Die Buttons für die zehn vordefinierten Stellungen können als ein MMF2 Standardobjekt im Frame Editor eingefügt werden und müssen nur in ihrer Größe angepasst werden. Alle anderen Hintergrundgrafiken werden in einem einfachen, in MMF2 integrierten Bildbearbeitungsprogramm erstellt und als sonst funktionslose Hintergrundobjekte in den Frame eingefügt und positioniert.

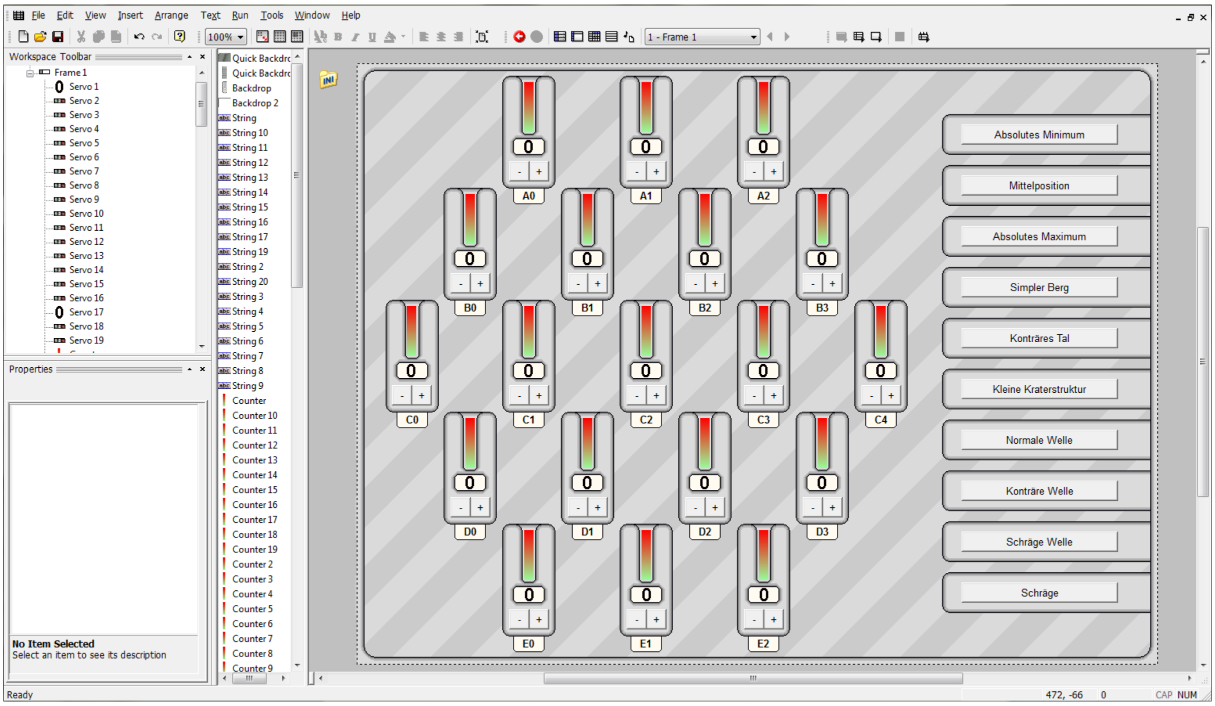
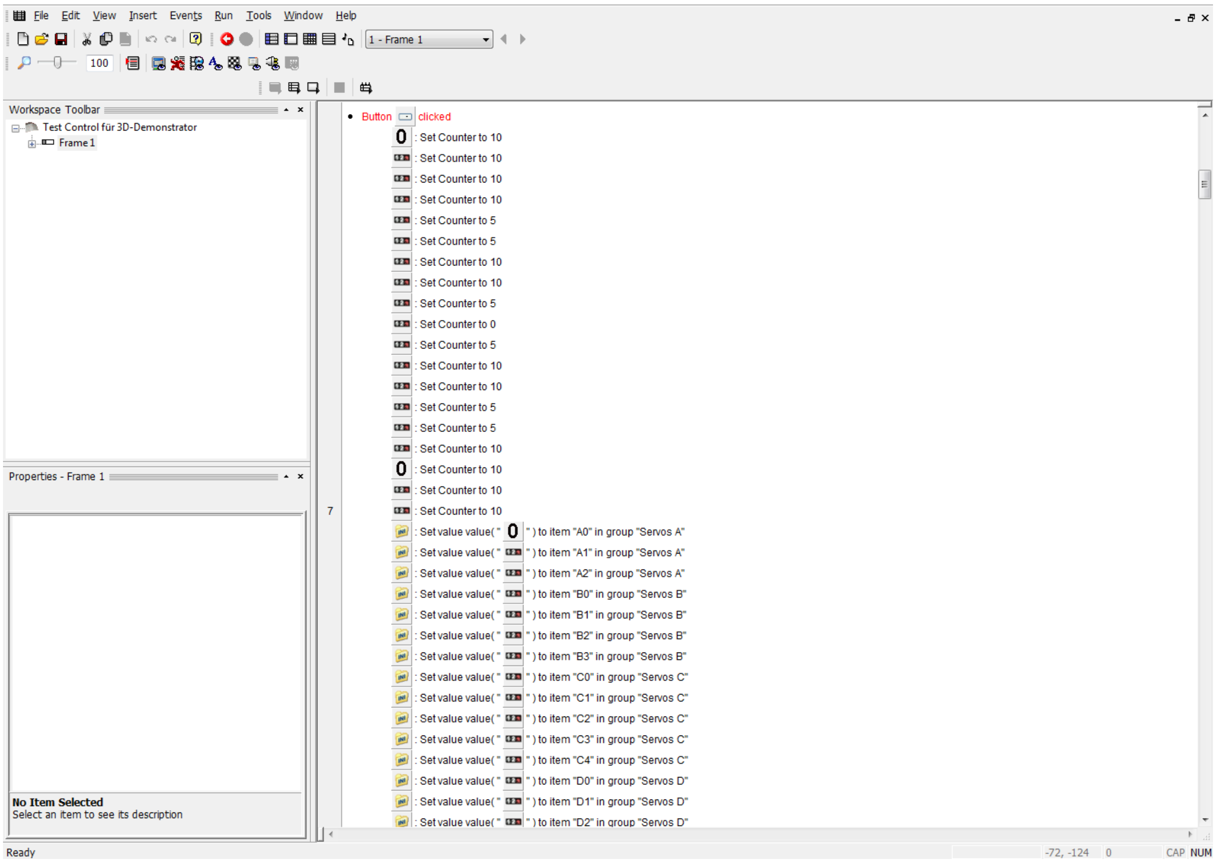


Abbildung 24: Frontpanel des Testprogramms im Frame Editor

Das dritte und vierte Fenster von MMF2 ist jeweils zur Programmierung da. Der Unterschied zwischen diesen Fenstern ist der Detailierungsgrad und damit der Grad der Geschwindigkeit in der der Programmcode hier festgelegt werden kann. So ist der Event Editor eine tabellarische Aufteilung, in der in der obersten Zeile alle Objekte des Frame Editors angezeigt werden. In der ersten Spalte werden Trigger-Events frei definiert, nach deren Eintritt weitere Befehle ausgeführt werden sollen (z.B. if „Mouse Button left click“ on „Button 1“). Daraus ergibt sich eine Gitterstruktur, in der die Folgebefehle festgelegt werden können. Diese definierten Events werden hier lediglich als Haken angezeigt und sind, bei mehreren Folgebefehlen zu einem einzelnen Trigger-Event, diese sind hier nicht in ihrer Abarbeitungsreihenfolge veränderbar. Das vierte Fenster, der Event List Editor, bietet daher die Möglichkeit diese Befehle in einer ausführlichen Liste via Drag and Drop in ihrer Abarbeitungsfolge zu verändern. Durch diese Möglichkeiten der schnellen Programmerstellung eignet sich Multimedia Fusion 2 besonders gut zur Erstellung von Rapid Prototyping Software.



```
• Button    clicked
    0 : Set Counter to 10
      : Set Counter to 10
      : Set Counter to 10
      : Set Counter to 10
      : Set Counter to 5
      : Set Counter to 5
      : Set Counter to 10
      : Set Counter to 10
      : Set Counter to 5
      : Set Counter to 0
      : Set Counter to 5
      : Set Counter to 10
      : Set Counter to 10
      : Set Counter to 5
      : Set Counter to 5
      : Set Counter to 10
    0 : Set Counter to 10
      : Set Counter to 10
7     : Set Counter to 10
      : Set value value( " 0 " ) to item "A0" in group "Servos A"
      : Set value value( "  " ) to item "A1" in group "Servos A"
      : Set value value( "  " ) to item "A2" in group "Servos A"
      : Set value value( "  " ) to item "B0" in group "Servos B"
      : Set value value( "  " ) to item "B1" in group "Servos B"
      : Set value value( "  " ) to item "B2" in group "Servos B"
      : Set value value( "  " ) to item "B3" in group "Servos B"
      : Set value value( "  " ) to item "C0" in group "Servos C"
      : Set value value( "  " ) to item "C1" in group "Servos C"
      : Set value value( "  " ) to item "C2" in group "Servos C"
      : Set value value( "  " ) to item "C3" in group "Servos C"
      : Set value value( "  " ) to item "C4" in group "Servos C"
      : Set value value( "  " ) to item "D0" in group "Servos D"
      : Set value value( "  " ) to item "D1" in group "Servos D"
      : Set value value( "  " ) to item "D2" in group "Servos D"
```

Quellcode 05: Programmausschnitt Testprogramm

Der kleine Ausschnitt des Quellcodes des Testprogramms zeigt, was passiert wenn der Button für die Option „Konträres Tal“ gedrückt wird. So wird in jede Variable ein Wert für die Auslenkung des jeweiligen Servos gesetzt, und diese Werte danach in die externe Displaydatei geschrieben. Die Anzeige über die Zählobjekte mit Balkenansicht in der Oberfläche des Testprogramms geschieht an einer anderen Stelle, da dieser Programmteil immer ausgeführt wird und somit nicht mit dem Button-Event direkt verknüpft werden muss.

Da aber Quellcodes von Multimedia Fusion nicht übersichtlich druckbar sind, wird hier auf weitere Auszüge verzichtet. Der komplette Quellcode, sowie die notwendige Software zur Betrachtung, ist aber auf der beigefügten DVD vorhanden (siehe Kapitel 16, Inhalte der DVD).

11. Fazit der Bachelorarbeit

Zusammenfassung

Mit Hilfe des 3D-Druckers war es möglich, innerhalb kurzer Zeit einen funktionierenden und professionellen Display- Demonstrator zu entwickeln und betriebsfertig aufzubauen. Die Technik ist zwar momentan noch nicht vollständig fehlerfrei, aber dennoch bereits weit entwickelt und einsatzbereit. Die ebenfalls schnelle Entwicklungsmöglichkeit mit LabVIEW, für eine stabile und zuverlässige Treiberlösung, hat sich ebenfalls als praktikabel erwiesen und steht im Hinblick auf Zuverlässigkeit nicht hinter den herkömmlichen Programmiersprachen wie C zurück.

Ergebnis der Bachelorarbeit

Das Ziel dieser Bachelorarbeit war es einen Display Demonstrator zur entwickeln, der es ermöglicht, eine computergenerierte Oberflächenform ertastbar zu machen. Damit soll es zum ersten Mal möglich sein, einer Person mit einer Sehbehinderung auch computergenerierte 3D-Objekte verständlich darzustellen. Dieses Ziel konnte erreicht werden. So entstand ein Display, dessen Auflösung es bereits ermöglicht, grobe Flächen- bzw. Objektverläufe darzustellen. Der Treiber ist ebenfalls voll funktionsfähig und bereits zuverlässig einsetzbar.

Zukunftsausblick

In Zukunft wird es immer wichtiger werden, sehbehinderten Personen eine Möglichkeit zu geben, moderne Kommunikationsmittel, wie einen PC oder Mobillösungen wie Smartphones und Tablets, zu benutzen. Diese Geräte bieten, besonders in Sachen Platzbedarf und Vielfalt, gegenüber den aktuell erhältlichen Büchern mit Reliefs und Brailleschrift deutliche Vorteile.

Die aktuell in dem Demonstrator verwendete Technik ist alles in allem zwar zuverlässig, muss aber besonders im Hinblick auf Platzbedarf und Kosten noch verbessert werden. So könnte in Zukunft beispielsweise auf magnetische Elemente gesetzt werden. Diese werden paarweise angeordnet, und können sich mit Hilfe von unterschiedlich starken Kräften gegenseitig abstoßen und somit die Verschiebung der Displayelemente ermöglichen. Der dafür benötige technische Unterbau müsste dabei wesentlich platzsparender ausfallen. Damit wäre es ebenfalls möglich, die Dreieckspaneele kleiner, zahlreicher und somit das Display hochauflösender zu gestalten. Dieses Ziel wird in Zukunft auch weiter verfolgt und soll in einem ausgereiften Display zur Verwendung kommen.

12. Fremdwort- und Abkürzungsverzeichnis

ABS:	**A**crylnitril-**B**utadien-**S**tyrol; ein thermoplastischer Kunststoff mit guten Eigenschaften für 3D Drucker.
Arduino:	Eine Open-Source Hardware-Plattform, d.h. alle Hardwarepläne und Quellen der Arduino- Baugruppen liegen offen und sind frei über das Internet verfügbar sind. Die Hardware basiert dabei hauptsächlich auf den Atmel AVR- Mikrokontrollern. [ARDU]
CAD:	**C**omputer **A**ided **D**esign; beschreibt das Konstruieren eines Objekts am Computer, heutzutage meist in 3D-Ansicht, um komplizierte Formen vorzudefinieren und evtl. auftretende Probleme frühzeitig erkennen zu können.
CSV:	**C**omma **S**eparated **V**alues; Dateiformat für Textdateien mit einfach strukturierten Inhalten, zumeist Tabellen.
DLL:	**D**ynamic **L**ink Fibrary; eine Programmbibliothek, die nach Bedarf geladen wird.
Handle:	Bezeichnung eines Referenzwertes zu einer Systemressource (Windows).
I^{2}C / I2C:	**I**nter- **I**ntegrated **C**ircuit; serieller Datenbus, der besonders gut für die Kommunikation zwischen einem Controller und angeschlossene Peripheriegeräte geeignet ist.
IDE:	**I**ntegrated **D**evelopment **E**nvironment; beschreibt eine Entwicklungsumgebung, die meist aus einem Quelltexteditor (im Falle von LabVIEW dem Blockdiagramm und dem Frontpanel), einem Compiler, einem Linker und einem Debugger besteht.

LabVIEW:	**Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench; eine Programmieroberfläche für die grafische Programmiersprache „G“.
MMF2:	**M**ulti**m**edia **F**usion **2**; Crossplattform für verschiedene Zielgeräte und Programmiersprachplattformen, mit besonderem Schwerpunkt auf Animationen und mediale Inhalte
PWM:	**P**uls**w**eiten**m**odulation (eng. **P**ulse- **W**idth- **M**odulation); siehe Kapitel 8.3, die Pulsweitenmodulation.
RSD:	**R**eal **S**ervo **D**eflection; Wirkliche Servoauslenkung, beschreibt den echten Winkel, in dem sich ein Servo- Ruderhorn zum Servo befindet auf einer Scala von 7 (Minimum) bis 3 (Maximum).
Schwarzschrift:	Diese Bezeichnung wurde für Schriftzeichen, die auf lateinischen Buchstaben beruhen eingeführt, um eine deutlichere Unterscheidung zur Blindenschrift, die auf einem Punktesystem beruht, zu ermöglichen.
SD:	**S**ervo **D**eflection; Servoauslenkung, beschreibt den Winkel, in dem sich ein Servo-Ruderhorn zum Servo befindet auf einer Scala von 0 (Minimum) bis 10 (Maximum).
STL:	**S**urface **T**essellation **L**anguage; eine computergenerierte Beschreibung von Oberflächen. Diese werden dazu in Dreiecke aufgeteilt. Die entstehenden .STL Dateien können für diverse CAD- Systeme genutzt werden.
VI:	**V**irtual **I**nstrument; Standard Dateiendung für LabVIEW Quellcodes.

13. Quellenverzeichnis

Literaturquellen:

[AGPS] Auge und Gehirn Psychologie des Sehens
Autor: Richard L. Gregory, Ausgabe: April 2001
ISBN: 3-499-60805-7, ro ro ro Verlag

[AUFA] Automated Fabrication
Autor: Ph. D. Marshall Burns, Ausgabe: Juli 1993
ISBN: 978-0131194625, Prentice Hall
Verwendete Auszüge: http://www.ennex.com/~fabbers/StL.asp, Stand: 09.06.2013

[BGBl] Bundesgesetzblatt I
Autor: Bundesanzeiger Verlag GmbH, Stand: 26.05.2013
Verfügbar unter: http://www1.bgbl.de/

[DSPW] Die Sinne und der Prozess der Wahrnehmung
Autor: James J. Gibson, Ausgabe: 1973
ISBN: 3-456-30586-9, Huber Verlag

[PRLA] Praxiseinstieg LabVIEW
Autor: Friedrich Plötzeneder / Birgit Plötzeneder, Ausgabe: 2010
ISBN: 978-3-7723-4039-0, Franzis Verlag GmbH
(Wurde als allgemeine Informationsquelle zu LabVIEW verwendet)

Webquellen

[ACRO] Acroname Robotics (Bildquelle)
Autor: Acroname Inc., Stand: 02.06.2013
http://www.acroname.com/robotics/info/articles/servo/servo_pwm1v3.jpg

[ARDU] Arduino.cc, Stand: 24.06.2013
http://www.arduino.cc/en/

[BARR] BARR group
Autor: Michael Barr, Stand: 14.05.2013
http://www.barrgroup.com/Embedded-Systems/How-To/PWM-Pulse-Width-Modulation

[BAUM] Baum Retec AG (Bildquelle)
Autor: Baum Retec AG, Stand: 30.05.2013
http://www.baum.de/cms/typo3temp/pics/1c5d730167.jpg

[BLST] Deutsche Blindenstudienanstalt e.V.
Autoren: Richard Heuer, Ernst- Dietrich Lorenz, Günther Meier, Thomas Schwyter,
Dr. Wolfgang A. Slaby, Christian Waldvogel
http://www.blista.de/download/druckerei/system_d_blindenschrift_7620.pdf

[BRBU] Braille Bug
Autor: American Foundation for the Blind
http://www.braillebug.org/louis_braille_bio.asp

[RNPW] RoboterNETZ
Autor: Robert Pukshofer, Stand 22.05.2013
http://www.rn-wissen.de/index.php/Bascom_und_PWM

[RNSE] RoboterNETZ
Autor: Frank Brall, Stand: 03.06.2013
http://www.rn-wissen.de/index.php/Servos
weitere Autoren (Übersicht):
http://www.rn-wissen.de/index.php?title=Servos&action=history

[TRIM] Trimble Ltd.
Autor: Trimble Ltd., Stand 24.05.2013
http://extensions.sketchup.com/en/content/sketchup-stl

14. Danksagung

Ich möchte mich nun, am Ende meines Studiums, besonders bei meiner Familie bedanken, die es mir überhaupt erst ermöglich hat, dieses Studium und meine gesamte vorhergehende schulische Laufbahn zu bewältigen und durchzuhalten.

Ich möchte mich auch sehr bei meinem guten Freund Herrn Sebastian Schmidt bedanken, für die Zeit, die er mich so manches Mal aus dem Stress des Studiums losgerissen hat.

Mein Dank gilt weiterhin der Beuth Hochschule für Technik Berlin, für das Engagement in der Weiterentwicklung und Forschung in vielen Bereichen der Informatik und den damit verbundenen Projekten, an denen ich ebenfalls teilhaben durfte.

Meinem Betreuer, Herrn Prof. Dr.-Ing. Alfred Rozek, möchte ich für sein großes Interesse und seine Unterstützung bei meiner Bachelorarbeit und dem behandelten Gebiet der Haptik, sowie für die Hilfsmittel, die mir zur Verfügung gestellt wurden, sehr danken.

Einen besonderen Dank möchte ich an dieser Stelle auch an Gerard Choinka richten, der mir mit seinem großen Wissen und seiner Erfahrung um den 3D Druck während meiner Bachelorarbeit hilfreich zur Seite stand. Ich wünsche ihm auf diesem Wege alles Gute für seine Masterarbeit.

15. Anhang

15.1. Verwendete Bauteile

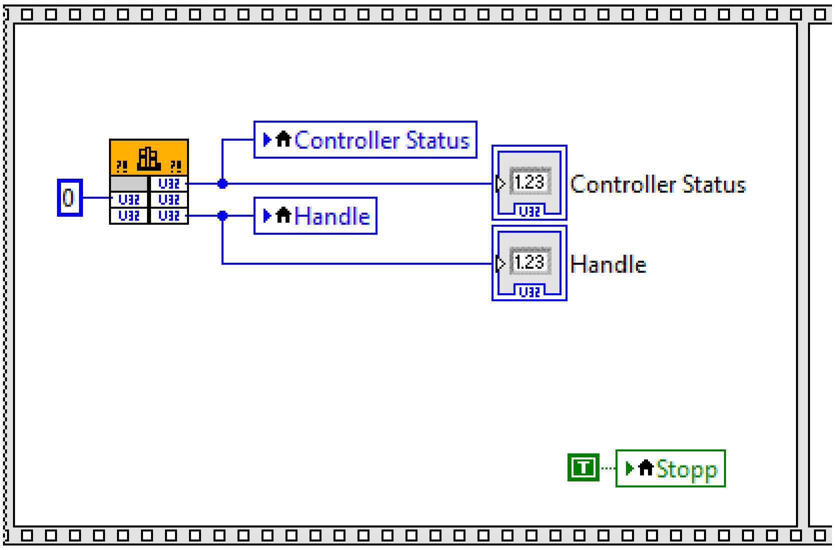
Bezeichnung: SD21 - Servocontroller Modul
Hersteller: Devantech Ltd.
Bezugsquelle: http://www.roboter-teile.de

Bezeichnung: USB zu I^{2}C Adaptermodul
Hersteller: Devantech Ltd.
Bezugsquelle: http://www.roboter-teile.de

Bezeichnung: Mini-Servo Gleitlager Getriebe Kunststoff JR
Hersteller: Modelcraft
Bezugsquelle: http://www.conrad.de
Info: nicht mehr lieferbar (Stand 17.07.2013)

Bezeichnung: USB Netzteil
Hersteller: noname
Bezugsquelle: http://www.reichelt.de
Info: benötigter Mindeststrom 1000 mA

15.2. Quellcodesammlung

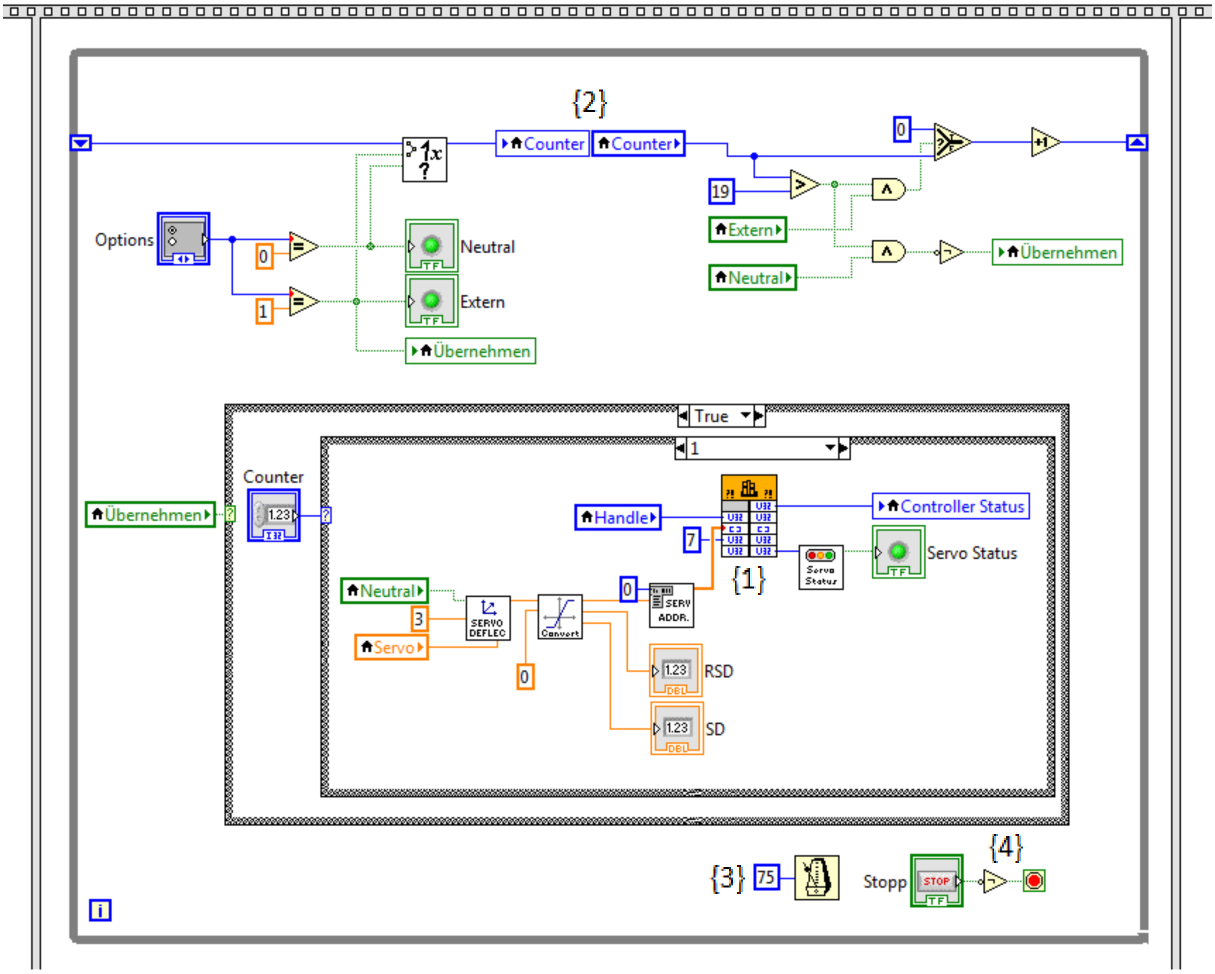


Quellcode 06: FT_Open

Aufgabe: Initialisieren des Servocontrollers mittels der FT_Open Funktion der FTD2XX.dll, Schreiben und Ausgeben des Controllerstatus und des Handle- Wertes, wird immer als Erstes bei Programmaufruf ausgeführt

Mögliche Controller Status- Fehlercodes:

0	Initialization correct
1	Controller blocked
4	Controller not found



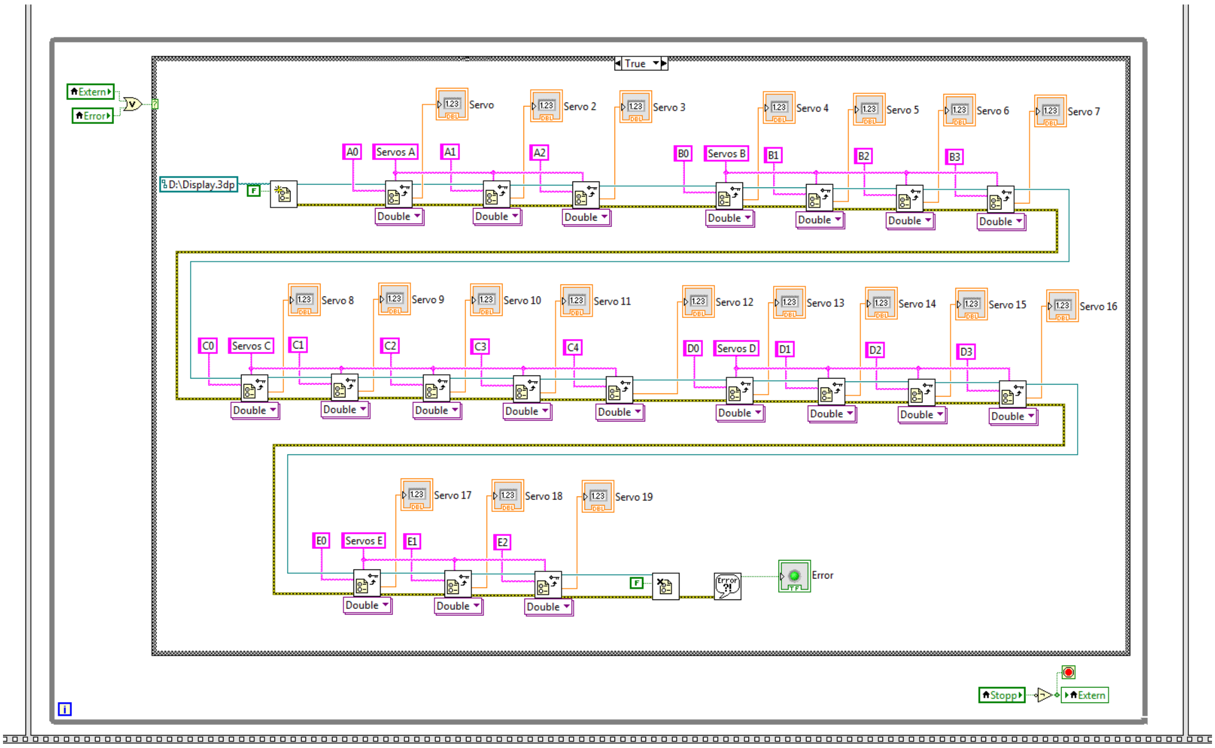
Quellcode 07: Indexdurchlauf und Schreiben auf Servocontroller

Aufgabe: Komplettübersicht des sequenziellen Schreibvorgangs auf den Servocontroller mittels der FT_Write Funktion der FTD2XX.dll {1},

Komplettansicht der Servo- Indexschleife {2},

der Verzögerung um 75ms zur Fehlervermeidung durch zu schnelles Schreiben {3}

und des Stopp Buttons {4}

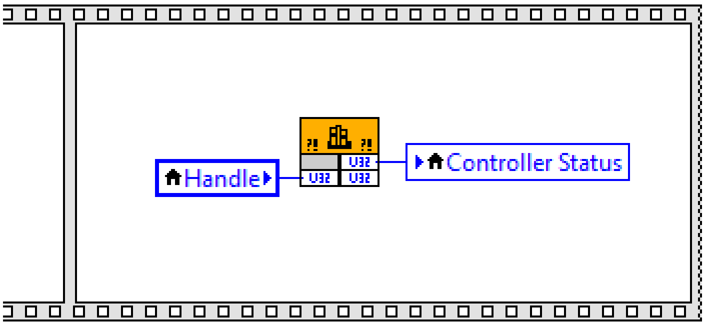


Quellcode 08: Datei lesen

Aufgabe: Vollständige Übersicht über das Einlesen der externen Datei zur Ansteuerung des Displays, automatisches Neustarten des Lesevorgangs bei vorhergehend fehlerhaftem Lesevorgang und automatischer Deaktivierung bei Neutralstellung, läuft parallel zum Schreibvorgang auf den Controller, ohne Timer- Verzögerung für schnelle Fehlerbehandlung

Quellcode 09: Dummyobjekte

Aufgabe: Die Registerkarte dient lediglich als Frontpanelelement zur übersichtlichen Aufteilung und wird im Blockdiagramm daher nicht weiter verwendet, die Variable „Übernehmen“ wird lediglich über die lokale Variable- Funktion gelesen bzw. verändert.



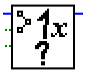
Quellcode 10: FT_Close

Aufgabe: Mittels der Funktion FT_Close der FTD2XX.dll wird die Verbindung zum Servocontroller beendet und der Port wieder freigegeben, der Servocontroller Status aktualisiert und ausgegeben. Die Funktion wird immer nach Drücken des Stopp-Buttons ausgeführt; bei Programmabbruch wird der Servocontroller blockiert: Fehlercode 1 (wird erst bei Neustart des Programms angezeigt)

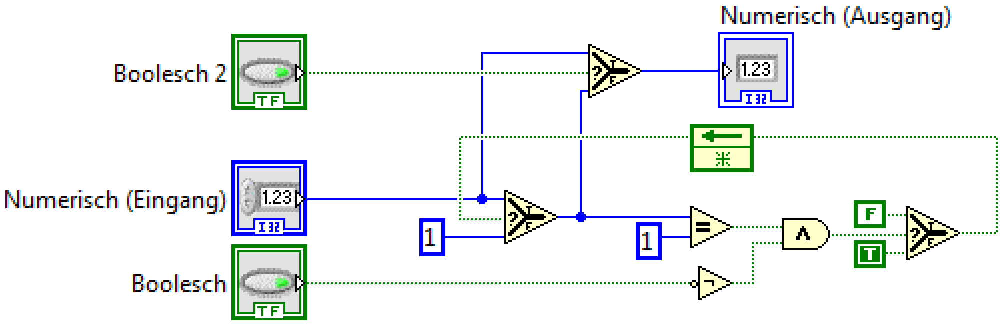
15.3. SubVI's Übersicht

Name: Options(SubVI).vi

Symbol:



Code:



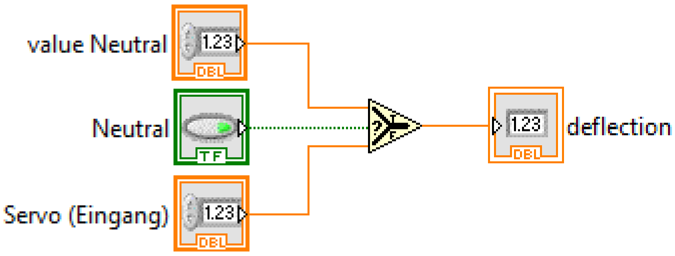
Quellcode 11: Options(SubVI)

Aufgabe: Servoindex wird bei Wahl der Option „neutral“ einmalig beim Start auf 1 gesetzt, bei Option „extern“ unverändert gelassen (siehe Kapitel 10.3. Optionswahl und Servoindex)

Name: ServoDeflectionOptions(SubVI).vi

Symbol:

Code:

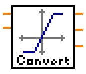


Quellcode 12: ServoDeflectionOptions(SubVI)

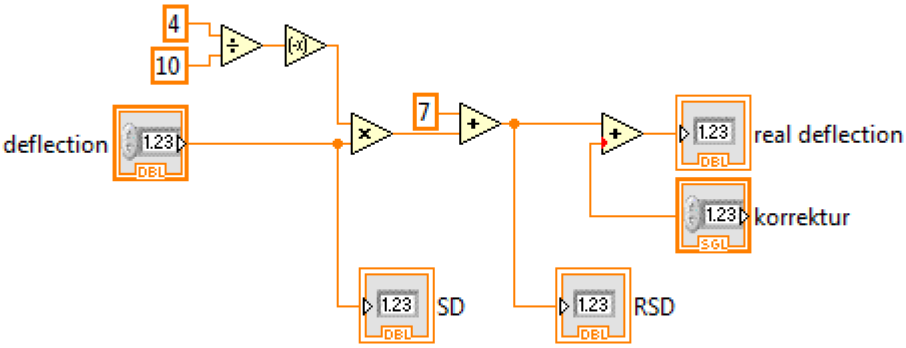
Aufgabe: Werteeingang abhängig von gewählter Eingabeoption steuern; ist ein veraltetes SubVI aus einer frühen Testphase mit Zusatzoptionen, die in der Endfassung unnötig sind. SubVI wurde nur zur Erhaltung der Übersichtlichkeit beibehalten

Name: ServoDeflectionConverter(SubVI).vi

Symbol:



Code:



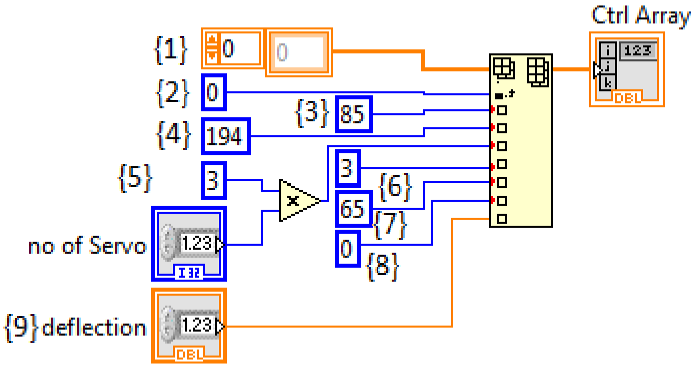
Quellcode 13: ServoDeflectionConverter(SubVI)

Aufgabe: Permanentes Konvertieren und Anzeigen der Eingabewerte aus dem Bereich 0 (Minimum) bis 10 (Maximum) in den Bereich von 7 (Minimum) bis 3 (Maximum) mittels der Linearen Funktion: $f(x) = -\frac{4}{10}x + 7$. Anschließend wird die Korrekturgröße für baubedingte Stellungsfehler des spezifischen Servos addiert.

Name: ServoAddress(SubVI).vi

Symbol:

Code:



Quellcode 14: ServoAddress(SubVI)

Aufgabe: Bilden und Ausgeben des Arrays für die Ansteuerung des Servos mit dem aktuellen Wert für die Servo- Auslenkung. Dabei haben die Elemente die folgende Bedeutung:

{1} das zu bildende Array ist eindimensional

{2} ist die Stelle im Array, an der die Zeile eingefügt wird (Tabellenfunktion)

{3} Zielgerät ist ein I^2C Controller; 85_{10} => $0x55_{16}$;

Datenblatt: „USB to I2C Communications Module”

Kapitel: „Writing to I2C devices with internally addressable registers”

{4} Adresse des SD21 Servocontrollers; $194_{10} = C2_{16}$;

Datenblatt: „SD21 – 21 Channel Servo Driver Module”

Kapitel: „Servo Processor”

{5} Servo- Adresse, erfolgt 0- basiert in 3er Schritten pro Port

{6} unused Port

Datenblatt: „USB to I2C Communications Module”

Kapitel: „USB-I2C Commands“

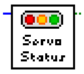
{7} Stellgeschwindigkeit der Servos

{8} Position Low Byte

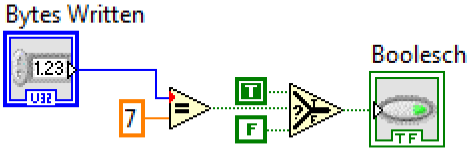
{9} Position High Byte

Name: ServoStatus(SubVI).vi

Symbol:



Code:



Quellcode 15: ServoStatus(SubVI)

Aufgabe: Kontrolle und boolsche Anzeige, ob alle Daten (-Bytes) beim letzten Schleifendurchlauf korrekt geschrieben wurden

16. Inhalte der DVD - Rom

Es wird empfohlen innerhalb dieser DVD über das Menü (start.html) zu navigieren.

Verzeichnis	Inhalt	Kurzbeschreibung
Menü	start.html css html pictures	Menü zur Übersicht über die DVD Inhalte
Bilder	CAD Bildausschnitte Diagramme Fotos Skizzen	Alle in der Bachelorarbeit verwendeten CAD Bilder, Diagramme und Fotos in höherer Auflösung; zusätzliche Fotos
Dokumente	Bachelorarbeit Datenblätter Quellen	Die Bachelorarbeit als PDF Datei, die Datenblätter des SD21 Servocontrollers und des I²C Adapter Moduls, Komplettkopien sämtlicher Webquellen und Buchquellen (lediglich Einbände)
DruckerModelldateien	SKP Dateien STL Dateien	Die CAD Modelle der aktuellsten Displaykomponenten
Programme	Display Demonstrator Treiber Testprogramm	Kompilierte Programmdateien des Displaytreibers und des Testprogramms
Quellcodes	Display Demonstrator Treiber Testprogramm	Quellcode Dateien des Displaytreibers und des Testprogramms
Treiber	I²C USB Port Test LabVIEW runtime Engine 2012 Multimedia Fusion 2 Demo	Testtool zur Prüfung des Ports/der Verbindung zum I²C Adapter Modul, Die LabVIEW runtime Engine 2012 32-Bit (Windows XP, Vista, 7 und 8) wird benötigt falls kein LabVIEW 2011 oder neuer vorinstalliert ist, MMF2 Demo wird zur Betrachtung des Testwerkzeug-Quellcodes benötigt
Videos		Videos von Tests des Displays in verschiedenen Entwicklungsstadien