Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

Dokumentation

Zur Diplomarbeit

„DMX-Steuerpult mit Motorfader“

Autor:	Spring Roman
Klasse:	15E
Abschlussjahr:	2018
Betreuer:	Nicolas Krauer
Starttermin:	17. Okt. 2017
Abgabetermin:	05. März 2018
Version:	1.00

Historie

Vers.	Wann?	Wer?	Was?
0.0	11.10.17	RSG	Erstellung Dokumentation / Dokulayout
0.1	18.10.17	RSG	Einleitung / Pflichtenheft
0.11	06.11.17	RSG	Pflichtenheft
0.2	19.11.17	RSG	Analyse
0.3	29.12.17	RSG	Analyse / Konzept
0.31	04.01.18	RSG	Konzept
0.32	05.01.18	RSG	Konzept
0.4	13.01.18	RSG	Realisierung Hardware
0.41	14.01.18	RSG	Realisierung Hardware
0.42	16.01.18	RSG	Realisierung Hardware
0.43	23.01.18	RSG	Realisierung Hardware
0.44	26.01.18	RSG	Realisierung Hardware
0.5	23.01.18	RSG	Realisierung Software
0.51	26.01.18	RSG	Realisierung Software
0.52	29.01.18	RSG	Realisierung Software
0.53	10.02.18	RSG	Realisierung Software
0.54	13.02.18	RSG	Realisierung Software
0.6	15.02.18	RSG	Testkonzept
0.61	16.02.18	RSG	Testkonzept
0.62	17.02.18	RSG	Messungen DMX Tester
0.63	18.02.18	RSG	Messungen Oszilloskop
0.64	19.02.18	RSG	Messungen Multimeter
0.65	21.02.18	RSG	Messung
0.7	22.02.18	RSG	Anhang
0.71	24.02.18	RSG	Anhang
0.72	26.02.18	RSG	Anhang
0.73	29.02.18	RSG	Anhang
1.0	04.03.18	RSG	Abgabeversion

Tab. 1 Historie

Kürzel:

RSG **R**oman **S**prin**g**

In der Historie sind nur grosse Änderungen aufgelistet.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Einleitung

Aufgabenstellung:
Praktisch alle DMX-Steuerungen, in den tieferen bis mittleren Preisklassen, besitzen keine Motorfader. Somit müssen bei einer Umstellung der Geräteselektion, die Werte von den Fadern abgeholt werden. Das bedeutet, dass die Fader bewegt werden müssen, bis der Wert des Scheinwerfers erreicht wird. Wenn mehrere Scheinwerfer angewählt sind, werden diese jedoch unterschiedlich schnell abgeholt. Daher werde ich innerhalb meiner Diplomarbeit eine DMX-Steuerung bauen, welche mit Motorfader arbeitet. Diese fahren bei einer Umstellung automatisch an ihre neue Position.
Da ich schon das eine oder andere Projekt mit dem Arduino realisiert habe und mich mit diesen Kontrollern bereits angefreundet habe, werde ich auch dieses Projekt mithilfe eines Arduinos erledigen.

Die Diplomarbeit umfasst mehrere Hauptaufgaben, die erfüllt werden müssen. Folgende Aufgaben müssen erledigt werden:

- Projektplanung
- Erstellung eines Pflichtenhefts mit Prioritäten
- Entwicklung der Hardware
- Entwicklung der Software
- Behandlung möglicher Fehler
- Erstellung eines Testkonzeptes
- Erstellung einer Kurzbedienungsanleitung
- Dokumentation aller Arbeitsschritte

Erklärung:
Ich erkläre hiermit, diese Arbeit (DA) selbstständig erarbeitet und dokumentiert zu haben.

Ort: Horgen

Datum: 05.03.2018

Unterschrift: .....

Deklaration des Aufwandes

⇨ Siehe Kapitel „8.4 Zeitplananalyse“ und „8.6 Stundenanalyse“

1 Pflichtenheft

1.1 Produktübersicht und Ausgangslage

Im Rahmen der Diplomarbeit soll ein DMX-Steuerpult gebaut werden. Das Steuerpult soll bis zu 16 LED-Scheinwerfer, mit jeweils sechs Kanälen (Rot, Grün, Blau, Weiss, Amber, UV), steuern können. Dies bedeutet, dass das Steuerpult 96 Adressen steuern muss. Zusätzlich soll das Pult über mehrere Tasten verfügen, welche verschiedene Farben oder Szenen aufrufen. Auch soll es möglich sein, die aktuelle Farbe oder Szene abzuspeichern und wieder aufzurufen. Die sechs Kanäle sollen per Fader angesteuert werden, welche auch per Motor an eine bestimmte Position gefahren werden können. Daher können die Fader bei einem Wechsel der Scheinwerfer auf die richtige Position fahren.

Zusätzlich zu den obligatorischen Punkten sollen noch optionale Ziele erreicht werden. Das erste optionale Ziel ist, die komplette Steuerung in einem eigenen Gehäuse unterzubringen und nicht auf einer improvisierten Testplatte. Die anderen beiden optionalen Punkte haben das Ziel, die Steuerung mobil zu machen. Dies soll erreicht werden, indem die Steuerung optional über einen Akku betrieben werden kann. Zusätzlich soll das DMX-Signal über Funk übermittelt werden können, damit gar keine Kabel mehr benötigt werden.

1.2 Zweck

Im Pflichtenheft werden die geltenden Anforderungen an die Diplomarbeit festgehalten, die durch den Auftragsgeber (die Lehrperson) festgelegt wurden.

1.3 Geltungsbereich / Gültigkeitsbereich

Die Gültigkeit dieses Pflichtenhefts betrifft die Diplomarbeit des Studenten Spring Roman.

1.4 Referenzierte Dokumente

- DA Aufgabenstellung Spring Roman
- DA Vorschlag DMX-Steuerung Spring Roman
- DA Bewertungsformular 2017 (Hardware/Software)
- Bestimmungen VDA/DA
- Terminplan DA HF 17/18

1.5 Eigesetzte Tools

Betriebssystem:	- Windows 10
Programmiertool:	- Arduino Tool - Notepad++
Dokumentation:	- Microsoft Word 365 - Microsoft Excel 365 - Microsoft Visio 2016
Hardware:	- Easy Eda (Schema) - Circuit Maker (PCB) - CorelDraw X8 (Frontplatte 2D) - FreeCAD (Frontplatte 3D) - Cure Version 3.1 (3D Drucker) - Picoscope 6 (Software für das PC-Oszilloskop)
Datensicherung:	- Lokal (Surface) - Automatisches Backup 1: Dropbox - Automatisches Backup 2: NAS (Privat)

1.6 Informationsbeschaffung

Erfahrung:
Mein Wissen über DMX kommt von meiner Erfahrung mit dem Umgang an diversen Steuerungen und Scheinwerfern. Durch die Arbeit mit vielen verschiedenen Geräten habe ich gelernt, was für eine Steuerung wichtig ist.
Das Programmieren habe ich mir auch während diversen kleineren Projekten angeeignet und somit auch verbessert.

Internet:
Die zusätzlich benötigten Informationen werden durch Recherchen über Google besorgt. Dies sind vor allem Informationen über den Aufbau und die Funktionsweise eines DMX-Signals. Da ich für die Diplomarbeit einzelne Funktionen programmieren muss, welche ich zuvor noch nie angewendet habe, werde ich für diese ebenfalls das nötige Wissen aus dem Internet beziehen.

Datenblätter:
Informationen über die einzelnen Bauteile werden mit der Hilfe von Datenblättern eingeholt.

1.7 Organisatorische Anforderungen

1.7.1 Zeitliche Ressourcen

Wichtige Termine:

Datum:	Was?
10.10.2017	Start der DA
06.11.2017	1. Besprechung
18.12.2017	2. Besprechung
15.01.2018	3. Besprechung
05.03.2018	Ende der DA

Tab. 2 Termine

Stundenaufwand:
Der Stundenaufwand wurde vereinfacht von dem DA-Vorschlag übernommen.

Tätigkeit	Soll-Aufwand [h]
Dokumentation	130
Hardware	25
Software	100
Präsentation	(10)
Total	**255** (265)

Tab. 3 Stundenaufwand

Detaillierte Zeitplanung:
Siehe Anhang: „Zeitplan“

Meilensteine:
Die Meilensteine beinhalten folgende Abschnitte, die erledigt sein müssen, damit der Meilenstein als erreicht gilt. Die folgenden drei Meilensteine sind zu erreichen:

Vorarbeit	Realisierung	Projektende
Zeitplanung	Analyse	Test
Einleitung	Konzept	Anhang
Pflichtenheft	Realisierung	Überarbeitung

1.7.2 Finanzielle Ressourcen

Da die finanziellen Mittel relativ begrenzt sind, sollen die Kosten des Projekts auch dementsprechend klein gehalten werden.
Das Material der Steuerung soll nicht mehr als 400.- SFr. kosten. Da bei den Bestellungen zusätzliche Ersatzteile mitbestellt werden, ist die Kostengrenze für das Projekt höher. Das Projekt soll nicht mehr als 600.- SFr. kosten.

1.7.3 Handhabung von Änderungen

Änderung	Handhabung
Aufgabenstellung	Rücksprache mit Betreuer
Grössere Änderungen am Pflichtenheft	Rücksprache mit Betreuer
Kleinere Änderungen am Pflichtenheft	Eigenverantwortung

Tab. 4 Handhabung von Änderungen

1.8 Zielsetzungen aus Anforderungen

Die Kriterien des Projekts sind in zwei Hauptgruppen unterteilt. Zum einen die **Muss-Kriterien**; diese Kriterien beinhalten unabdingbare Leistungen, die in jedem Fall erfüllt werden müssen. Zum anderen, die **Soll-Kriterien** (Optionale Kriterien); diese Kriterien werden im Laufe des Projekts angestrebt und nach Möglichkeit erfüllt.

1.8.1 Muss-Kriterien:

#			Prio.	Bemerkung
1		**Dokumentation**		
	1	Erstellung einer kompletten Dokumentation	1	
	2	Erstellung eines Testkonzepts	2	
	3	Dokumentiertes Testen	3	
	4	Erstellung einer Kurzbedienungsanleitung	2	
2		**Hardware**		
	1	16 Geräte auswählbar	1	
	2	6 Motorfader	1	
	3	9 Farb- & Szenetasten	2	
	4	1 Blackouttaste	2	
3		**Software**		
	1	Geräte anwählbar	1	
	2	Auswertung und Ansteuerung der Motorfader	1	
	3	Farb- & Szenetasten abspeicher- & abrufbar	2	
	4	Funktion Blackouttaste	3	
	5	Ausgabe eines normgerechten DMX-Signals	1	

Tab. 5 Muss-Kriterien

Dokumentation

1.1 Erstellung einer kompletten Dokumentation

Folgende Hauptkapitel muss die Dokumentation beinhalten: Pflichtenheft, Analyse, Konzept, Realisierung, Kontrolle.

Messwert:

- Sind alle Kapitel abgearbeitet?

1.2 Erstellung eines Testkonzepts

Es muss ein Testkonzept erarbeitet werden, in welchem sinnvolle Testkriterien vorhanden sind.

Messwert:

- Sind min. 80% der Funktionen im Testkonzept abgedeckt?

1.3 Dokumentiertes Testen des Spiels

Die ausgeführten Tests müssen analysiert werden.

Messwert:

- Ist der Test analysiert worden?

1.4 Erstellung einer Kurzbedienungsanleitung

Die Anleitung muss die Bedienung auf 3 – 10 Seiten erklären.

Messwert:

- Hat die Anleitung 3 - 10 Seiten?
- Sind die wichtigsten Funktionen erklärt worden?

Hardware

2.1 16 Geräte auswählbar

Es muss die Möglichkeit bestehen, alle 16 Geräte zu selektieren.

Messwert:

- Können alle 16 Geräte ausgewählt werden?

2.2 6 Motorfader

Das Steuerpult muss sechs Motorfader für die Steuerung der Kanäle aufweisen.

Messwert:

- Sind mindestens sechs Motorfader verbaut?

2.3 9 Farb- & Szenentasten

Für das Abspeichern und Abrufen der Farben und Szenen müssen je neun Tasten vorhanden sein.

Messwert:

- Sind je neun Tasten vorhanden?

2.4 1 Blackouttaste

Für die Blackout-Funktion muss ebenfalls eine Taste vorhanden sein.

Messwert:

- Ist eine Taste für die Blackout-Funktion vorhanden?

Software

3.1 Geräte auswählbar

Die Geräte müssen einzeln auswählbar sein.

Messwert:

- Können einzelne Geräte dazu- oder weggeschaltet werden?

3.2 Auswertung und Ansteuerung der Motorfader

Die Positionen der Fader müssen eingelesen werden können. Zudem müssen die Fader bei der Umschaltung der Geräte in die richtige Position fahren.

Messwert:

- Können die Fader über die ganze Länge ausgelesen werden?
- Fahren die Fader an die richtige Position nach dem Umschalten?

3.3 Farb- & Szenentasten abspeicher- & abrufbar

Die 18 Tasten müssen abrufbar sein, bzw. auch mit einem neuen Wert überspeicherbar sein

Messwert:

- Sind alle neun Farben abrufbar?
- Sind alle neun Szenen abrufbar?
- Können neue Farben bzw. Szenen gespeichert werden?

3.4 Funktion Blackouttaste

Beim Betätigen der Blackouttaste müssen alle Werte der Geräte vorübergehend auf null gesetzt werden. Beim erneuten Betätigen müssen die Geräte wieder die alten Werte erhalten.

Messwert:

- Können die Geräte mit der Blackouttaste aus- und wieder eingeschaltet werden?

3.5 Ausgabe eines normgerechten DMX-Signals

Das auszugebende DMX-Signal muss der Norm entsprechen. Zudem muss das Signal gemäss den Werten angepasst werden.

Messwert:

- Ist das aufzugebende Signal normgerecht?
- Wird das Signal richtig angepasst?

1.8.2 Soll-Kriterien:

#			Prio.	Bemerkung
1		**Dokumentation**		
2		**Hardware**		
	1	Gehäuse	1	
	2	Wireless DMX Modul	1	
	3	Akkubetrieb	2	
3		**Software**		

Tab. 6 Soll-Kriterien

2.1 Gehäuse

Es soll ein geschlossenes Gehäuse gebaut werden, welches die IP20 Norm erfüllt.

Messwert:

- Ist ein vollwertiges Gehäuse vorhanden?
- Erfüllt das Gehäuse die IP20 Norm?

2.2 Wireless DMX Modul

Damit das DMX-Signal nicht nur per Kabel an die Scheinwerfer übertragen werden kann, soll ein Wireless DMX Modul eingebaut werden, mit welchem das Signal ebenfalls per Funk übertragen werden kann.

Messwert:

- Können verschiedene Scheinwerfer mit dem Modul gekoppelt werden?
- Wird das DMX-Signal versendet?

2.3 Akkubetrieb

Um komplett kabellos zu sein, soll das Gerät per Akku oder Batterie betrieben werden können.

Messwert:

- Kann das Gerät ohne Speisung betrieben werden?
- Halten die Akkus lange genug (Siehe Kapitel: „2.3 Akkubetrieb“)?

1.9 Persönliche Ziele

Neben dem eindeutigen Ziel, dass ich eine gute Diplomarbeit abliefern will, gibt es noch weitere Gründe. Zum einen, dass ich meine Kenntnisse über Projektentwicklung auffrischen kann. Zum anderen, dass ich etwas entwickeln kann, was nicht nur für die Diplomarbeit zu gebrauchen ist, sondern mir auch danach noch etwas bringt.

2 Analyse

2.1 Layout Frontpanel

Das Frontpanel wird ähnlich gestaltet, wie die Frontpanels von anderen Steuereinheiten. Als Vorbild wurde der „LED Operator 6“ von Showtec gewählt, welcher ähnlich gross ist. Er hat gleich viele Fader und hat etwa die gewünschte Grösse. Allerdings kann dieser nur sechs Geräte ansteuern und hat keine Szenenfunktion. Dafür hat er mehr Farbtasten zur Verfügung, welche jedoch auch nicht selbst belegt werden können. Zusätzlich hat er noch eine Soundfunktion, auf welche ich bei meiner DMX-Steuerung verzichten werde.

Abb. 1 Led Operator 6

2.2 Bauteileauswahl

Die Auswahl der Hauptkomponenten wurde bereits vor der Konzeptphase festgelegt.

Fader

Die Auswahl der Fader mit Motoren im bezahlbaren Bereich ist relativ beschränkt, da ein Grossteil der Motorfader für die Audioindustrie hergestellt werden. Da diese in der Audioindustrie viel genauer sein müssen, sind diese auch viel teuerer. Schliesslich habe ich mich für den Fader RS60N von Alps entschieden. Um die Motoren in den Fader ansteuern zu können, benötige ich auch einen Treiber pro Motor, da der Arduino den benötigten Strom nicht liefern kann. Als Treiber habe ich mich für den L293B in einem DIP-16 Gehäuse entschieden.

Tasten

Bei der Tastenwahl griff ich auf das Lager meines Betriebes zurück, welcher verschiedene elektronische Bauteile für Prototypen und Reparaturen besitzt. Die ausgewählten Tasten gehören eher zur älteren Generation. Sie funktionieren jedoch hervorragend und kosten nichts. Zusätzlich sind die Tasten mit und ohne Led vorhanden, diese sind von der Grösse und vom Aussehen her gleich.

Wireless-DMX-Modul

Bei dem Wireless-DMX-Modul hatte ich keine grosse Wahl. Ich musste entweder ein chinesisches Produkt bestellen oder eines das von einer Firma in Schweden kommt. Diese Firma produziert einen Grossteil der WDMX-Module, welche in praktisch allen professionellen Geräten eingebaut sind. Da ist die Auswahl relativ einfach gefallen und ich entschied mich für das Modul aus Schweden von der Firma Wireless Solution.

Arduino

Für dieses Projekt benötige ich relativ viele Ports, sowie viel Speicher, da ich die Szenen und Farben abspeichern muss. Die Pins summieren sich durch die Geräte-, Szenen- und Farbtasten, durch die Motorenansteuerung, sowie durch die Ausgabe des DMX-Signals zusammen. Für die Tasten benötige ich im schlimmsten Fall 28 Pins. Für die Fader benötigt der Arduino je einen analogen Pin und für die Treiber noch zusätzlich zwei digitale Pins für die Richtungsangabe. Das ergibt für die Fader noch einmal 22 Pins, wenn man das Enable-Signal noch dazu zählt. Wenn die Ausgabe des DMX-Signals noch dazu gezählt wird, ergibt dies eine Gesamtpinanzahl von 52, welche das Arduinoboard verarbeiten und ansteuern können muss. Daher kommt nur ein Arduinoboard der Mega-Reihe in Frage, welche 54 Digital-IO und 16 Analog-IO haben. Daher entschied ich mich für das Arduino Mega 2560 Board.

2.3 Akkubetrieb

Damit eine Batterie bzw. Akkubetrieb Sinn macht, muss die Steuerung mindestens einen Eventabend ohne Stromanschluss überstehen. Daher muss die Batterie bzw. der Akku eine Zeitspanne von 6 - 8h überbrücken können.
Für die Berechnung der benötigten mAh werden die grössten Stromverbraucher eingerechnet, alle übrigen Stromverbraucher werden vernachlässigt.

Arduino:

Der Arduino Mega 2560 verbraucht im aktiven Zustand 10 mA. Und da dieser durchgehend läuft, ergibt sich eine Stromkapazität von 10 mAh.

Fadermotoren:

Die Motoren der Fader sind mit 0.8 A angegeben. Da diese jedoch nur beim Wechsel der Geräte kurz angesteuert werden, ist der Stromverbrauch dementsprechend kleiner. Auch wenn ein Wechsel unter 0.5 Sek dauert, wird für die Berechnung einfachheitshalber den Wert 0.5 Sek verwendet. Zusätzlich wird angenommen, dass alle Geräte einmal einzeln eingestellt werden. Daher muss jeder der sieben Fader 16-mal die Position ändern. Daher ergibt sich eine Änderungsanzahl von 112.
Das ergibt für die Fader eine Stromkapazität von:

$$800\,mA \cdot \frac{0.5\,Sek}{3600\,\frac{Sek}{Stunde}} \cdot 112\,Bewegungen = 12.4\,mAh$$

Led:

Die Leds in den Tasten benötigen im Betrieb 7 - 9 mA. Da sich auf der Steuerung zehn Leds befinden, wird für die Berechnung der Worstcase angenommen, sprich dass alle zehn Leds für die volle Stunde leuchten.

$$9\,mA \cdot 10\,Stück = 90\,mA => 90\,mAh$$

WDMX-Modul:
Das WDMX-Modul ist der grösste Stromverbraucher. Es benötigt im Sende-Modus 200 mA, was einer benötigten Kapazität von 200 mAh entspricht.

Dies gibt eine totale Stromkapazität von 312.4 mAh. Damit die Steuerung einen sechs stündigen Eventabend übersteht, muss der Akku dem entsprechend eine Stromkapazität von mindestens 1874.4 mAh aufweisen.

Da ein normaler AA-Akku bereits 2000 – 2600 mAh besitzt (6.4 - 8.3 stündiger Eventabend), kann der Akkubetrieb mit AA-Akkus realisiert werden. Für die Fadermotoren wird eine Spannung um die 10 V benötigt, daher werden acht dieser AA-Akkus in Serie geschalten.

2.4 DMX-Signal

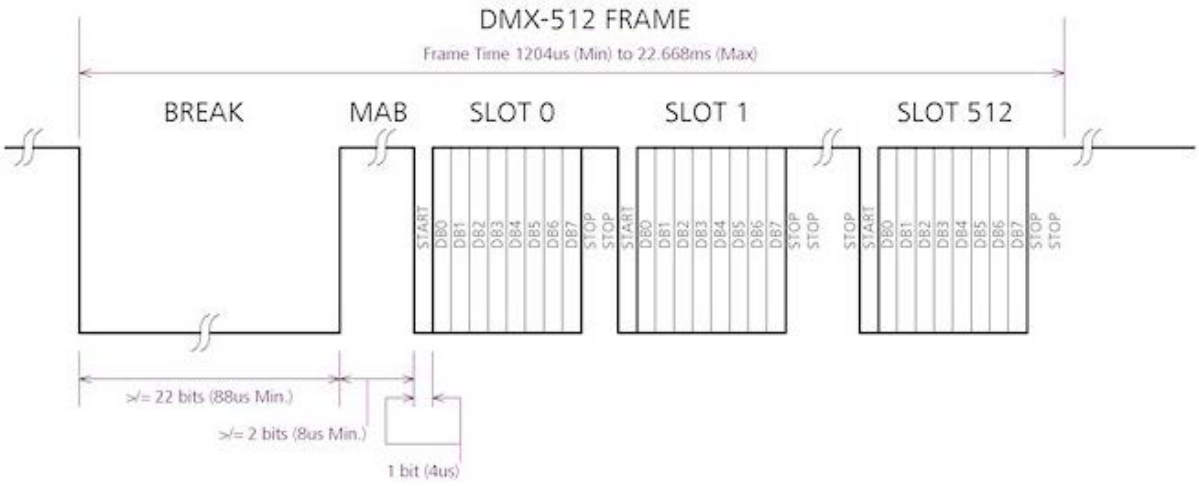

Abb. 2 DMX-Aufbau

Ein DMX-Signal ist ein serielles Signal, welches verwendet wird um die unterschiedlichsten Scheinwerfer in der Event-Branche anzusteuern. Das Signal ist folgendermassen aufgebaut: Jede Adresse benötigt insgesamt elf Bits, zu Beginn ein Startbit, acht Wertebits und für den Abschluss zwei Stoppbits. Diese elf Bits werden auch Slot genannt. Vor jeder Übertragung des Frames wird ein Break-Signal gesendet, welches 22 Bits lang ist. Anschliessend wird das MAB-Signal (Mark After Break) gesendet, dies leitet die Datenübermittlung ein.
Da die DMX-Steuerung 16 Geräte mit je sechs Adressen (Rot, Grün, Blau, Weiss, Amber, UV) steuert, ergibt dies 96 Adressen. Wenn diese 96 Slots auf die Bits runtergerechnet werden, ergibt dies 1056 Bits. Mit dem Break- und MAB-Signal ergibt dies eine Framelänge von 1080 Bits.

3 Konzept

3.1 Konzeptübersicht (Grobkonzept)

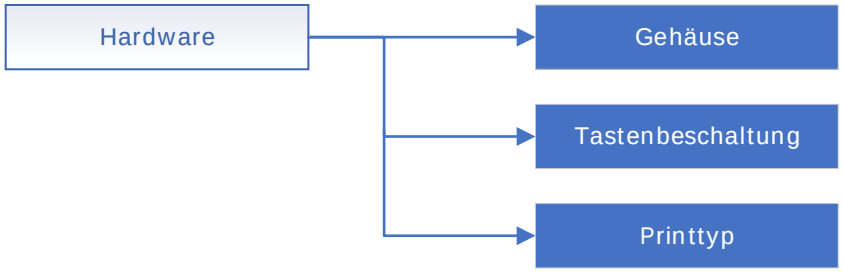

Abb. 3 Grobkonzept, Übersicht der Konzepte

3.2 Konzeptvarianten

Für die folgenden Punkte wird ein Konzept ausgearbeitet.

Hardware:

- Gehäuse -> Wie wird das Gehäuse realisiert?
- Tastenbeschaltung -> Wie werden die Tasten an den Arduino angeschlossen?
- Printtyp -> Wie werden die Prints realisiert?

Die Konzeptvarianten werden in einer Tabelle ausgewertet. Jeder der Vergleichspunkte wird von einer Skala von 1 (schlecht) bis 4 (gut) bewertet. Anschliessend werden die Werte mit ihrer Gewichtung multipliziert. Daraus kann zum Schluss ein Total pro Variante ermittelt werden. Mit dessen Hilfe kann so festgestellt werden, welche Variante optimal für das Projekt ist.

3.3 Konzepterarbeitung

3.3.1 Hardware

3.3.1.1 Gehäuse

Variantenbeschreibung:

Kompletter Eigenbau (Variante 1):

In der ersten Variante soll das komplette Gehäuse selber gebaut werden. Dies könnte zum Beispiel komplett durch den 3D-Drucker geschehen. Eine andere Option wäre, dass das Gehäuse aus Holz oder alternativ aus Metall gebaut wird.

Fertiggehäuse (Variante 2):

Für diese Variante wird ein Gehäuse fertig eingekauft und die Frontplatte wird angepasst. Diese Variante bedingt allerdings, dass ein sehr gut passendes Gehäuse gefunden wird.

Fertiggehäuse mit eigenem Frontpanel (Variante 3):

Die letzte Variante besteht aus einem Fertiggehäuse, welches im Gegensatz zur zweiten Variante nur einigermassen passen muss. Da die Frontplatte entweder aus Metall gebaut oder mit dem 3D-Drucker ausgedruckt wird.

Bewertungskriterien mit Gewichtung (KW):

Bewertung		Variante 1		Variante 2		Variante 3	
Bewertungskriterium	**Gew.**	**KW**	**Tot**	**KW**	**Tot**	**KW**	**Tot**
Aufwand	4	1	4	3	12	3	12
Kosten	2	4	8	2	4	3	6
Ästhetik	1	2	2	4	4	3	3
Total			**14**		**20**		**21**

Tab. 7 Konzepterarbeitung Gehäuse

Variantenbewertung mit definierten Kennwerten:

Aufwand:

4 fast kein — 1 grosser

Kosten:

4 geringe — 1 hohe

Ästhetik:

4 Professionell — 1 Bastelarbeit

Variantenauswahl mit Begründung, Beschreibung soweit noch nötig

Ich werde die Variante 3 weiterverfolgen, da dort das Kosten / Nutzenverhältnis am optimalsten ist. Nach dem die Hardware definiert und gebaut ist, kann ich ein passendes Gehäuse suchen und kaufen. Das Frontpanel werde ich auf dem PC zeichnen und mit dem 3D-Drucker ausdrucken. Dies hat den grossen Vorteil, dass ich so das Frontpanel so gestalten kann wie es am meisten Sinn macht.

3.3.1.2 Tastenbeschaltung

Variantenbeschreibung:

Direktbeschaltung (Variante 1):
In der ersten Variante soll jede einzelne Taste direkt an einen Pin angeschlossen werden.

Matrixbeschaltung (Variante 2):
In der zweiten Variante soll die Tastenbeschaltung mit der Hilfe einer Tastenmatrix geschehen. Dies erhöht den Softwareaufwand, jedoch wird der Hardwareaufwand verringert, da weniger Kabel mit den Pins verbunden werden müssen.

Bewertungskriterien mit Gewichtung (KW):

Bewertung		Variante 1		Variante 2	
Bewertungskriterium	**Gew.**	**KW**	**Tot**	**KW**	**Tot**
Aufwand	4	3	12	2	8
Kosten	2	2	4	3	6
Überprüfung (Software)	2	2	4	3	6
Total			**20**		**20**

Tab. 8 Konzepterarbeitung Tastenbeschaltung

Variantenbewertung mit definierten Kennwerten:

Aufwand:
4 fast keiner — 1 grosser

Kosten:
4 geringe — 1 hohe

Überprüfung (Software):
4 schnell / einfach — 1 langsam / schwierig

Variantenauswahl mit Begründung, Beschreibung soweit noch nötig

Die Tasten in diesem Projekt werde ich mit einer Matrix einlesen, da ich ohne eine Matrix-Beschaltung fast alle Pins des Arduino verbrauchen würde und somit keine Reserven mehr hätte. Da sich die Tasten nicht an einem Ort auf dem Fronpanel befinen, werde ich für dieses Projekt drei kleinere Matrizen bauen. Die Matrixvariante hat zudem noch den Vorteil, dass die Tasten in der Gruppe entprellt werden können und nicht jede Taste einzeln eine Entprellung benötigt.

3.3.1.3 Printtyp

Variantenbeschreibung:

Veroboard (Variante 1):

Bei dieser Variante sollen die einzelnen Prints auf Veroboards (auch Lochrasterprint) realisiert werden. Dadurch könnte der Print jederzeit angepasst werden, falls in der späteren Entwicklung etwas nicht funktionieren würde.

Layout (Variante 2):

In der zweiten Variante sollen die Prints gelayoutet werden. Die Realisierung würde daher jedoch mehr Zeit in Anspruch nehmen. Zusätzlich zur Zeit, welche für das Layouten und Bestücken benötigt wird, kommen noch die Anfertigungs-, sowie die Lieferzeit dazu.

Bewertungskriterien mit Gewichtung (KW):

Bewertung		**Variante 1**		**Variante 2**	
Bewertungskriterium	**Gew.**	**KW**	**Tot**	**KW**	**Tot**
Aufwand	3	2	6	2	6
Kosten	1	3	3	2	2
Ästhetik	2	1	2	4	2
Realisierbarkeit	3	3	9	2	6
Total			**20**		**16**

Tab. 9 Konzepterarbeitung Printtyp

Variantenbewertung mit definierten Kennwerten:

Aufwand:

4 fast kein — 1 grosser

Kosten:

4 geringe — 1 hohe

Ästhetik:

4 Professionell — 1 Bastelarbeit

Realisierbarkeit:

4 einfach — 1 schwierig

Variantenauswahl mit Begründung, Beschreibung soweit noch nötig

Ich habe mich trotz der Tabelle für die Variante 2 entschieden. Da ein Print mit einem Veroboard sehr schnell nach einer Bastelei aussieht. Zudem habe ich im zweiten Lehrjahr das letzte Mal einen Print gelayoutet. Daher sehe ich diese Diplomarbeit als Chance an, mich wieder mit der Materie bekannt zu machen. Der Aufwand ist aus meiner Sicht bei beiden Varianten etwa gleich gross, da das Realisieren mit den Veroboards viele Lötarbeiten bedeuten würde und beim Layouten muss ich mich zuerst wieder in die Materie einarbeiten.

4 Hardware

4.1 Grobdiagramm

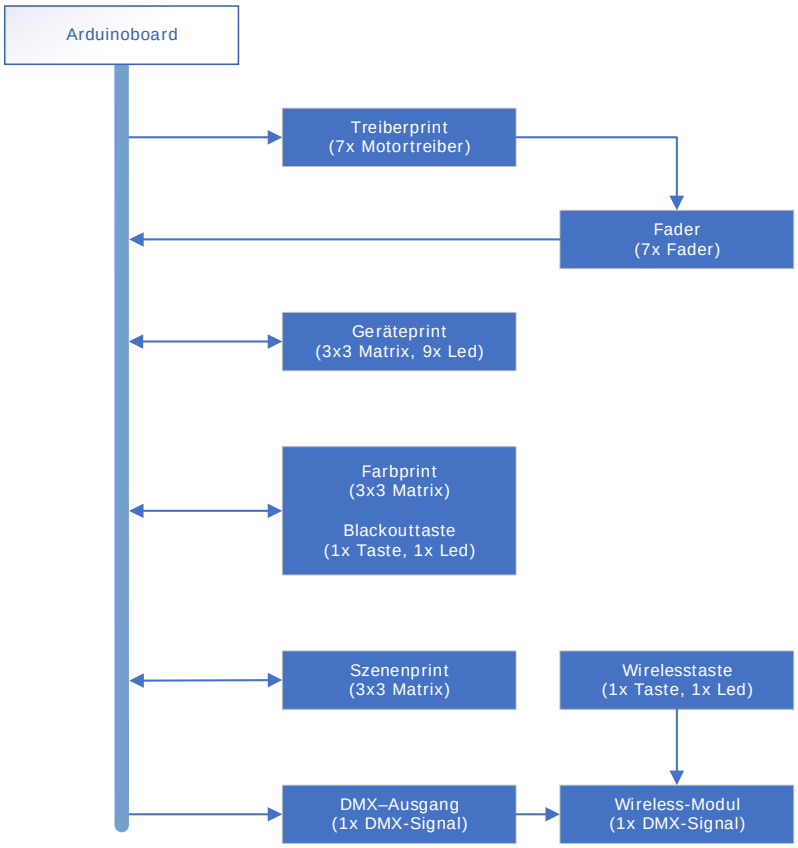

Abb. 4 Grobdiagramm Hardware

Das Grobdiagramm zeigt die Verhältnisse der einzelnen Elemente der DMX-Steuerung auf. Die Hauptplatine der Steuerung bildet das Arduino Mega 2560 Board. Der Arduino ist für die Kommunikation zwischen den einzelnen Modulen verantwortlich. Das Arduinoboard kann den Strom, welcher von den Motoren benötigt wird, nicht liefern. Daher wird ein Treiberprint angefertigt, der die Ansteuerung der Motoren übernimmt. Der Treiberprint weist somit eine Verbindung zu Arduino und den Fadern auf.

Die sieben Fader werden direkt mit dem Frontpanel verschraubt. Sie sind mit den Treibern und dem Arduinoboard verbunden.

Für die Matrizen werden drei kleinere Matrixplatinen gebaut. Durch die Gruppierung der Tasten wird jeweils ein Print für die Auswahl der Geräte, Farben und Szenen angefertigt. Diese Prints haben je eine Verbindung zum Arduino, mit Zu- und Rückleitungen.

Neben dem Arduinoboard gibt es noch ein weiteres Modul, welches fertig eingekauft wird. Das WDMX-Modul der Firma Wireless Solution, welches für die Übermittlung des DMX-Signals per Funk zuständig ist. Dieses Modul ist parallel zum DMX-Ausgang angeschlossen.

4.1.1 Treiberprint

4.1.1.1 Schema

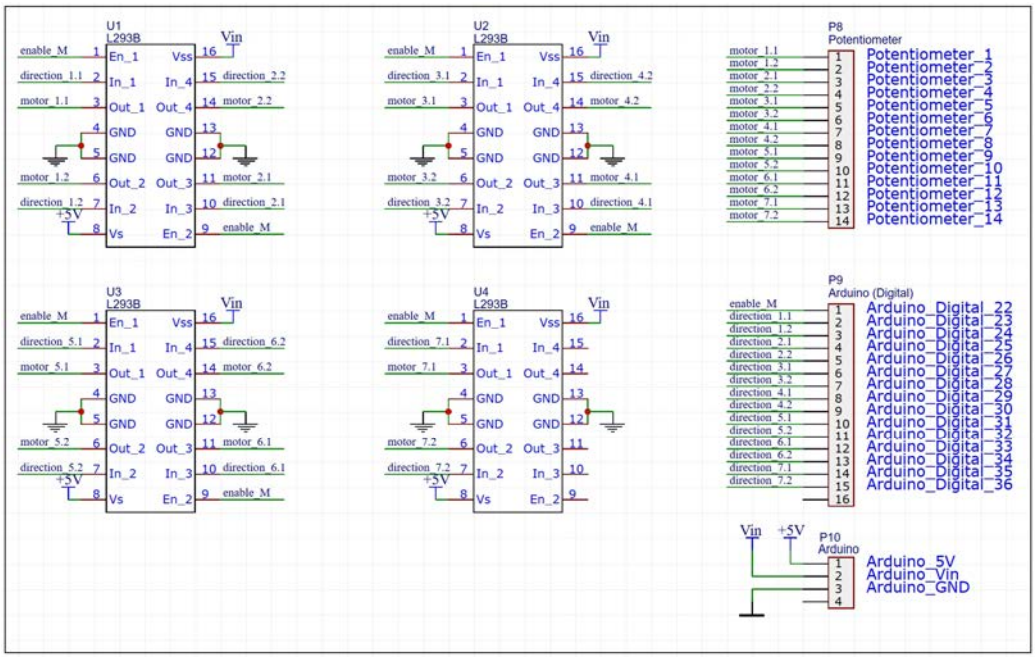

Abb. 5 Treiberprint – Schema

Diese Treiber benötigen jeweils ein Enable- und zwei Richtungssignale. Das Enable-Signal startet die Motoransteuerung und die Richtungssignale bestimmen, in welche Richtung die Motoren angesteuert werden. Um Verbindungen, sowie Pins auf dem Arduino, zu sparen, werden alle Enable-Signale zusammen als ein Signal angeschlossen.
Die beiden Richtungssignale aller Treiber werden mit dem Enable-Signal zusammen auf eine Stiftleiste geführt. Diese Stiftleiste wird anschliessend mit dem Arduino verbunden.
Die Anschlüsse für die Motoren werden ebenso auf eine Stiftleiste geführt, an welcher die Motoren der Fader angeschlossen werden. Die Stromzufuhr wird separat auf eine kleine Stiftleiste geführt.

4.1.1.2 Layout

Bei der Platzierung der Motortreiber wurde darauf geachtet, dass es möglichst wenig Kreuzungen gibt. Um an Höhe zu sparen, sind die Anschlüsse so platziert, dass letztendlich die Stiftleisten liegend angelötet werden. Damit der Print in das Gehäuse eingebaut werden kann, sind in den Ecken jeweils ein Loch für eine M3-Schraube platziert worden.

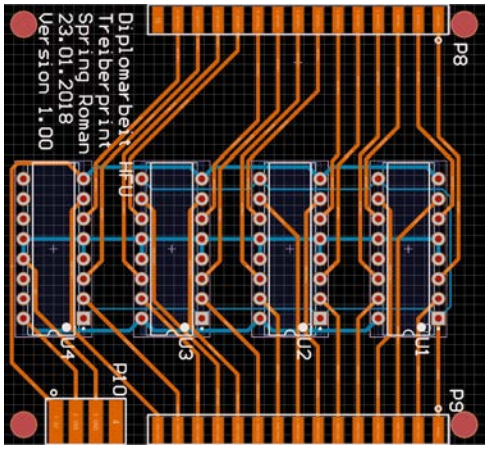

Abb. 6 Treiberprint - Layout

4.1.2 Berechnungen (Matrizen)

4.1.2.1 Berechnung der Hardware-Entprellung

Die Hardware-Entprellung wird mit Hilfe eines RC-Filters erreicht. Das Filter ist ein RC-Tiefpass mit einer Grenzfrequenz von 250 Hz. Durch die Eigenschaft eines solchen Tiefpasses, werden alle Frequenzen, die höher als die Grenzfrequenz sind, gedämpft.

Als Kondensator wird ein 1 uF verwendet, da dieser bereits vorhanden ist und im richtigen Bereich liegt. Durch diesen Kondensator ergibt sich folgende Widerstandsgrösse:

$$f_g = \frac{1}{2 \cdot \pi \cdot R \cdot C} => R = \frac{1}{2 \cdot \pi \cdot C \cdot f_g} = \frac{1}{2 \cdot \pi \cdot 1uF \cdot 250\ Hz} = 636\ \Omega => E24 - Reihe => \underline{\underline{R = 620\ \Omega}}$$

$$f_{gneu} = \frac{1}{2 \cdot \Omega \cdot R \cdot C} = \frac{1}{2 \cdot \pi \cdot 620\ \Omega \cdot 1\ uF} = \underline{\underline{256.7\ Hz}}$$

Bei den Berechnungen wurden die Toleranzen weggelassen, da eine eindeutige Grenzfrequenz für diesen Tiefpass irrelevant ist.
Jede Matrix besitzt drei Tiefpässe, für jeden Strang einen. Ein weiterer Tiefpass sorgt für die Entprellung der Blackouttaste.

4.1.2.2 Berechnung der LED-Widerstände

Um nicht zu viel Strom zu verschwenden, wird der Strom für die Leds auf 7 - 9 mA beschränkt. Dies wird erreicht, indem man die gleichen Widerstände verwendet, welche auch für die Entprellung zum Einsatz kommen.

$$I = \frac{U}{R} = \frac{5\ V}{620\ \Omega} = 8\ mA$$

4.1.3 Farb- & Szenenmatrix

4.1.3.1 Schema

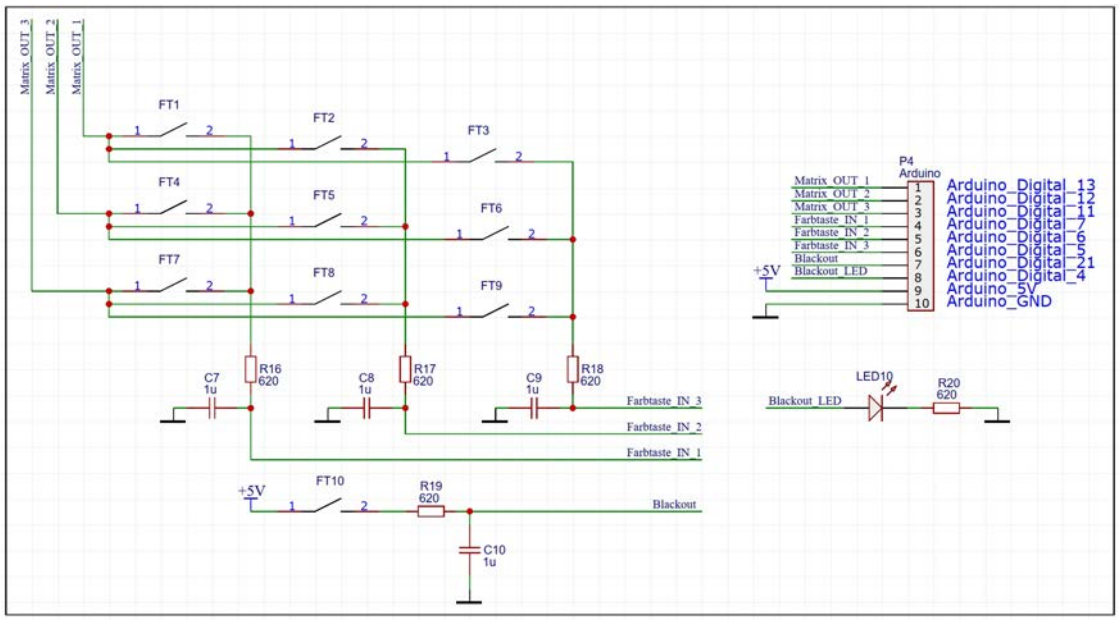

Abb. 7 Farbprint – Schema

In einer Matrix muss nicht jede Taste entprellt werden, es genügt, wenn die einzelnen Ableitungen entprellt werden. Die Platinen für die Farb- und Szenenmatrix sind praktisch identisch, mit dem Unterschied, dass der Szenenprint keine Blackouttaste besitzt. Daher ist hier auch nur die Farbmatrix aufgeführt, welche repräsentativ für den Szenenprint steht.

4.1.3.2 Layout

Da der Farbprint, sowie der Szenenprint praktisch identisch sein müssen, konnte das Layout so gezeichnet werden, dass für beide Matrizen das gleiche Layout verwendet werden kann. Für den Szenenprint wird nur die Blackouttaste (FT10), der Tiefpass (R19/C10) und die Blackoutled (LED10) mit ihrem Vorwiderstand (R20) weggelassen. Wie bei dem Treiberprint, werden hier die Stiftleisten ebenfalls liegend montiert, damit die Höhe möglichst gering bleibt. Da das gleiche Layout verwendet werden kann, wird auch nur dieses Layout für die beiden Prints bestellt. Dies führt jedoch dazu, dass die Beschriftungen bei der Szenenplatine nicht mit den Bezeichnungen auf dem Schema übereinstimmen. Die Übereinstimmung ist jedoch nicht zwingend notwendig, da sowieso die gleichen Bauteile eingesetzt werden.

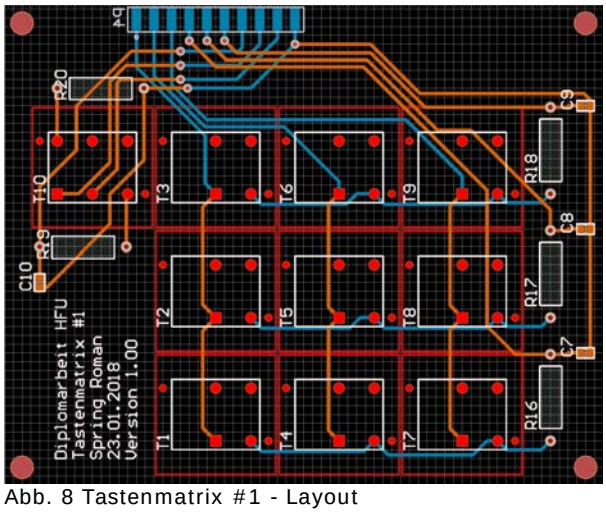

Abb. 8 Tastenmatrix #1 - Layout

4.1.4 Gerätematrix

4.1.4.1 Schema

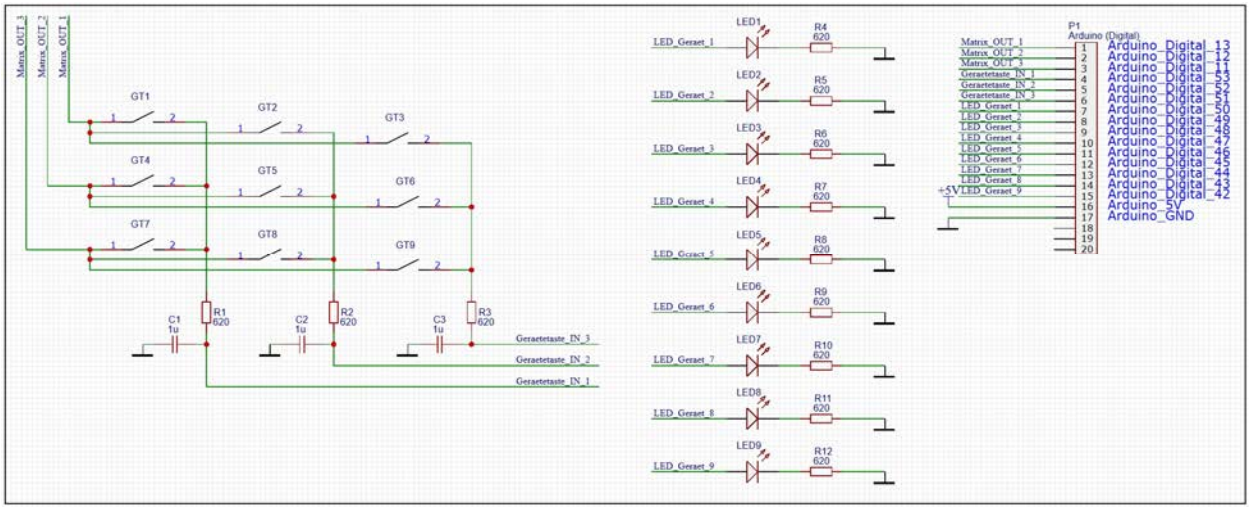

Abb. 9 Gerätematrix - Schema

Der Geräteprint unterscheidet sich nicht gross von dem Farbprint, bzw. dem Szenenprint. Der Unterschied liegt an den Tasten, welche hier alle eine Led besitzen. Zudem müssen die Tasten anders angeordnet werden. Durch die zusätzlichen Pins der Led ist die Stiftleiste, welche für die Verbindung zum Print verwendet wird, grösser.

4.1.4.2 Layout

Die Tasten werden nicht wie bei dem Farbprint in einem Quadrat angeordnet, sondern in zwei Reihen. Die Tasten GT1-GT8 werden für die Auswahl der Geräte verwendet und die Taste GT9 wird die Seite umschalten. Auch bei diesem Print wird die Stiftleise liegend montiert, um Platz in der Höhe zu sparen.

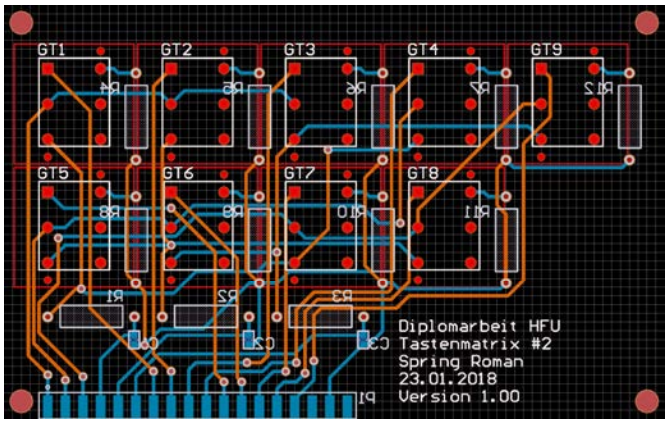

Abb. 10 Gerätematrix - Layout

4.1.5 Faderansteuerung

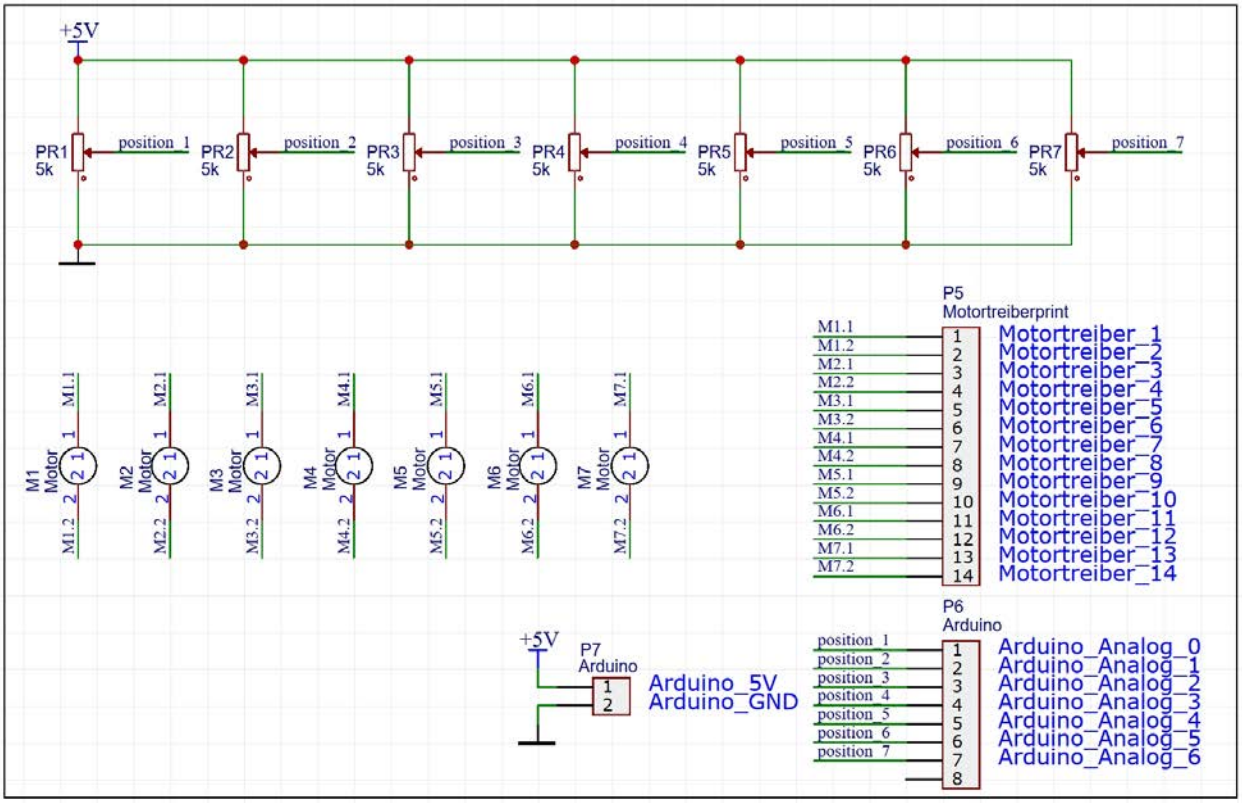

Abb. 11 Fader - Schema

Die Fader besitzen Lötaugen und keine Pins, daher können sie nicht auf einen Print gelötet werden. Die Fader werden mit der Frontplatte verschraubt und werden direkt mit der Verbindung zum Treiberprint und zum Arduino verlötet.
Die Fader werden wie Potentiometer angeschlossen. Der Vorteil an dem Arduino ist, dass man den Kurzschlussfall ignorieren kann. Somit kann der Widerstand, welcher den Kurzschluss verhindert weggelassen werden.
Die Motoren bräuchten theoretisch Freilaufdioden, welche bei der Abschaltung den Strom kontrolliert abfliessen lassen. Da sich diese Motoren in beide Richtungen bewegen, wäre eine kompliziertere Beschaltung nötig. In dieser Steuerung haben die Motoren jedoch genügend Zeit um sich zu beruhigen, daher werden diese Dioden für dieses Projekt weggelassen.

4.2 Umsetzung

4.2.1 Gehäuse

Die Suche nach dem richtigen Gehäuse stellte sich als schwerere Aufgabe heraus als gedacht. Der erste Punkt auf den geachtet wurde war, dass alle Elemente auf dem Frontpanel Platz findet. Diese Bedingung war jedoch bereits der erste Knackpunkt, da sehr viele Gehäuse die 20 cm Grenze nicht überschreiten. Damit alle Elemente auf dem Frontpanel platziert werden können, benötigt dieses jedoch mindestens 30 cm. Wenn ein Gehäuse die nötige Breite hatte, fehlte es jedoch an der Höhe. Da die Höhe des Gehäuses um die 10 cm besitzen muss, um alle Bauteile und Prints unterzubringen.
Kurz bevor der Entschluss gefällt wurde, ein eigenes Gehäuse zu bauen, wurde auf Conrad ein passendes Gehäuse gefunden. Die Frontplatte ist sehr knapp für alle Elemente, genügt jedoch um alle Elemente unterzubringen. Zudem ist die Frontplatte bei diesem Gehäuse leicht abgeschrägt, was die Bedienung der Steuerung komfortabler macht.

4.2.2 Speisung und Akku

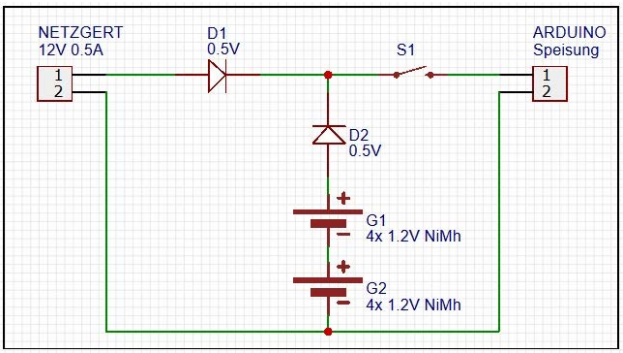

Abb. 12 Speisung und Akku

Die Spezifikationen des Arduino Mega 2560 schlägt eine Spannung von 7 - 12 V vor. Die Motoren der Fader benötigen eine Spannung um die 10 V. Damit das Netzteil nicht an der Anschlagsgrenze läuft, wird für die DMX-Steuerung ein 12 V Netzteil verwendet. Durch die Schottky-Diode (D1), welche dem Verpolungsschutz dient, liegt am Arduino eine Spannung von 11.5 V an.
Durch die serielle Beschaltung der acht NiMh-Akkus wird eine Spannung von 9.6 V erreicht. Davon werden die 0.5 V der Schottky-Diode abgezogen. Dies ergibt eine Spannung von 9.1 V für den Arduino. Da die Spannung über den Akkus kleiner ist als die 11.5 V, werden die Akkus nicht belastet, wenn das Netzteil angeschlossen ist. Die Diode (D2) verhindert, dass die Akkus unkontrolliert über das Netzteil geladen werden. Jedoch werden die Motoren im Akkubetrieb sich langsamer bewegen, da die Spannung kleiner ist.

4.2.3 DMX-Signal (Wire & Wireless)

4.2.3.1 Schema

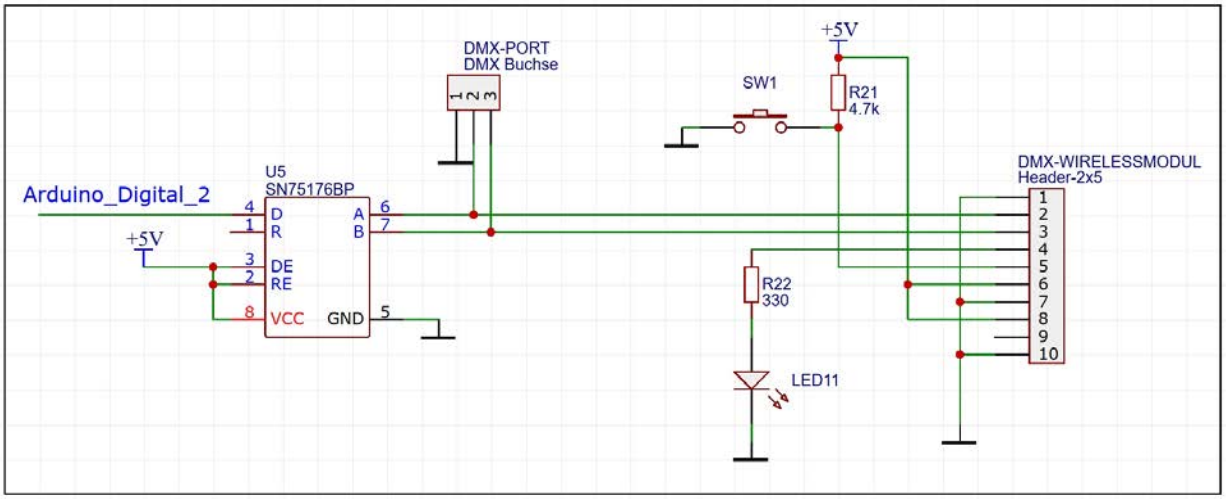

Abb. 13 DMX-Signal Verarbeitung

Dieses Schema zeigt nicht wie die anderen ein Print auf, sondern zwei. Zum einen die Signalverarbeitung mit dem SN75176B und zum anderen das Wirelessmodul. Die DMX-Buchse, die Taste und die Led befinden sich nicht auf einem Print, sondern direkt am Gehäuse.

4.2.3.2 Stromschlaufe (SN75176BP)

Eine Stromschlaufe hat im Gegensatz zur Übertragung mittels Spannungspegel den grossen Vorteil, dass die Übermittlung viel störresistenter ist. Aus diesem Grund wurde damals das DMX-Signal als Stromschlaufe definiert.
Dies führt zu dem Problem, dass das Signal nicht direkt von dem Arduino gesendet werden kann. Um ein DMX-Signal zu generieren, welches von den Scheinwerfern und von den Messgeräten akzeptiert, bzw. erkannt wird, benötigt es noch einen Zusatz-IC.
Für dieses Projekt wird der SN75176B verwendet, welcher in sehr vielen DMX-Produkten zum Einsatz kommt. Dieser wandelt einen Spannungspegel in eine Stromschlaufe um, er könnte auch umgekehrt wirken, was das Empfangen eines Signales über die Stromschlaufe ermöglichen würde. Da die Beschaltung dieses Bauteils nicht viel mehr als das Abgreifen von Pins ist, wurde der Print mit Hilfe eines Veroboards realisiert.
Durch die Beschaltung der Eingänge DE/RE wird definiert, ob der Baustein für das Senden oder Empfangen zuständig ist.

4.2.3.3 Wireless-Modul

Für das Senden des DMX-Signals per Funk wird das Modul Nano von der Firma Wireless Solution aus Schweden verwendet. Der Vorteil an diesem Modul ist, dass viele professionelle Scheinwerfer, welche Wireless DMX verwenden, mit diesem gekoppelt werden können. Die Integrierung dieses Moduls ist sehr einfach, da das DMX-Signal direkt ohne weitere Verarbeitung an das Modul gesendet werden kann. Das WDMX-Modul benötigt zusätzlich noch eine externe Taste, welche zur Steuerung des Moduls verwendet wird. Damit man ein Feedback von dem Modul bekommt, benötig es noch eine Led. Diese Led wird dirtekt über einen Widerstand an das Modul angeschlossen. Für diese DMX-Steuerung wird eine Taste verwendet, in welchem sich auch die Led befindet.

4.2.4 Frontpanel

Die Frontplatte wurde zu Beginn in CorelDraw gezeichnet. Da die einzelnen Elemente und ihre Befestigungslöcher als Gruppe gezeichnet wurden, konnten diese so lange verschoben werden, bis die optimale Position gefunden wurde.

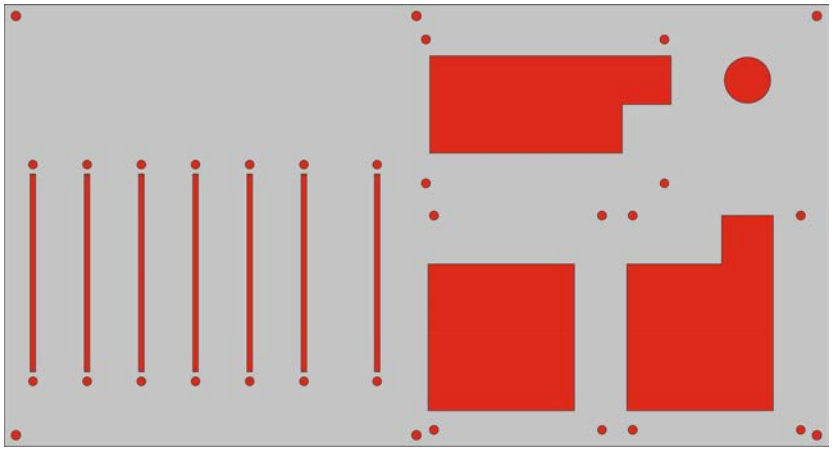

Abb. 14 Frontpanel Gesamt

Das aufgekommene Problem war, dass der 3D-Drucker (Ultimaker 2), welcher wir in unserem Betrieb verwenden, nur Bauteile in der Grösse von 225 mm x 225 mm x 205 mm drucken kann. Die Frontplatte hat jedoch eine Grösse von 161.5 mm x 304,1 mm. Aus diesem Grund wird das Frontpanel in zwei Teilen gedruckt. Diese beiden Teile haben noch eine Grösse von 161.5 mm x 152.05 mm und können somit mit dem Drucker ausgedruckt werden.

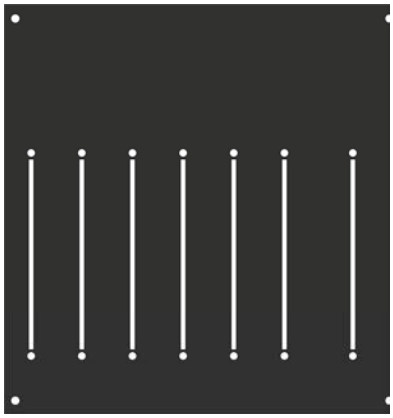

Abb. 15 Druckgrafik Frontpanel Links

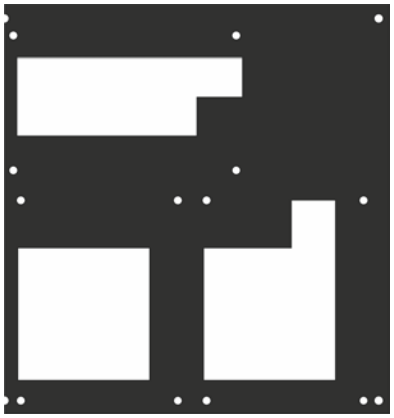

Abb. 16 Druckgrafik Frontpanel Rechts

Im nächsten Schritt werden die beiden vektorbasierten Grafiken im FreeCAD extrudiert. Dies bedeutet, dass die 2D Grafiken zu einem Volumenkörper werden, welcher der 3D-Drucker ausdrucken kann.

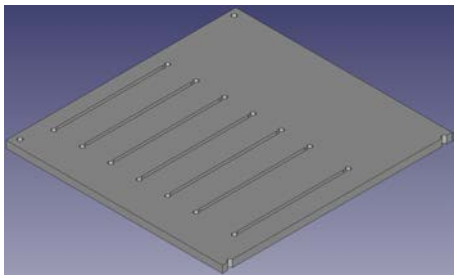

Abb. 17 3D Frontpanel Links

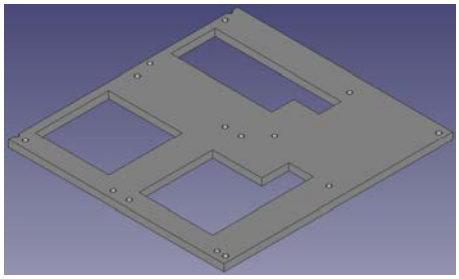

Abb. 18 3D Frontpanel Rechts

Im letzten Schritt werden im Cura (Software für den 3D-Drucker) die G-Codes generiert. Der 3D-Drucker benötigt diese Codes um die Teile zu drucken, da diese alle benötigten Informationen enthalten.
Folgende Informationen sind neben dem eigentlichen Model die wichtigsten:

Schichtdicke:
Die Schichtdicke bestimmt die Dicke jeder einzelnen Schicht und durch die Höhe des Models sind auch die Anzahl Schichten definiert. Für die beiden Frontplatten wurden die Schichtdicke jeweils auf 0.15mm festgelegt. Dies ergibt durch die Höhe von 3 mm eine Schichtanzahl von 20.

Wände:
Ein Produkt aus einem 3D-Drucker ist nicht komplett ausgefüllt oder hohl. Es besitzt im Kern ein Raster aus Quadraten oder Waben, dies erhöht die Stabilität des Produkts enorm. Damit der Drucker weiss, wann dieses Muster beginnt, benötig er Angaben zur Wanddicke. Da der Druckkopf einen Durchmesser von 0.4 mm hat, wurde die Wanddicke auf 0.8 mm festgelegt.

Fülldichte:
Durch Fülldichte wird definiert, wie dicht das Raster gedruckt wird. Mit einer Fülldichte von 100% wird das komplette Bauteil ausgefüllt. Hingegen bei einem Wert von 0% wird das Bauteil hohl gedruckt. Meiner Erfahrung nach ist ein Wert von 20 - 30% am idealsten, daher wurde die Fülldichte dieser Bauteile auf 20% eingestellt.

Geschwindigkeit:
Die Druckgeschwindigkeit des 3D-Druckers wirkt sich direkt auf die Qualität des Produkts aus. Grundsätzlich gilt, je langsamer desto genauer. Da das Drucken bei zu langsamer Geschwindigkeit nie ein Ende findet, muss hier ein Kompromiss gefunden werden. Für die Geschwindigkeit haben meine Erfahrungen gezeigt, dass sie im Bereich von 40 mm/s liegen sollte. Dieser Wert wurde auch für die Frontplatte übernommen.

Material:
Im Betrieb wird primär der Kunststoff PLA verwendet, da dieser relativ gut zu drucken ist. Zudem ist der PLA Kunststoff sehr UV beständig, was in einer längeren Lebensdauer resultiert. Zusätzlich sollte der PLA Kunststoff theoretisch biologisch abbaubar sein, was in der heutigen Zeit durchaus sinnvoll ist. Aus diesen Gründen wird auch die Frontplatte der Steuerung aus PLA gedruckt.

Einer der Nachteile an 3D-Druckern ist die notwendige Zeit, um ein Produkt zu drucken. Im Fall der DMX-Steuerung benötigt die linke Hälfte der Frontplatte 10 h 30 min und die rechte Seite eine Zeit von 7 h (Angabe von Cura).

4.2.5 Diverses

4.2.5.1 Verbindungskabel

Da die Steuerung aus mehreren Platinen besteht, müssen alle Module miteinander verbunden werden. Alle Prints, bis auf das Arduinoboard, sind mit männlichen Stiftleisten ausgerüstet. Durch die Stiftleisten können Kabel angefertigt werden, welche wieder entfernt werden können. Dies ermöglicht das einfache Austauschen bzw. Reparieren von defekten Modulen.
Die verwendeten Litzenkabel haben einen Litzendurchmesser von 0.2 mm^2. Damit die Litzen nicht bei der kleinsten mechanischen Belastung brechen, werden die Pins der Stiftleisten mit Heissleim verklebt.

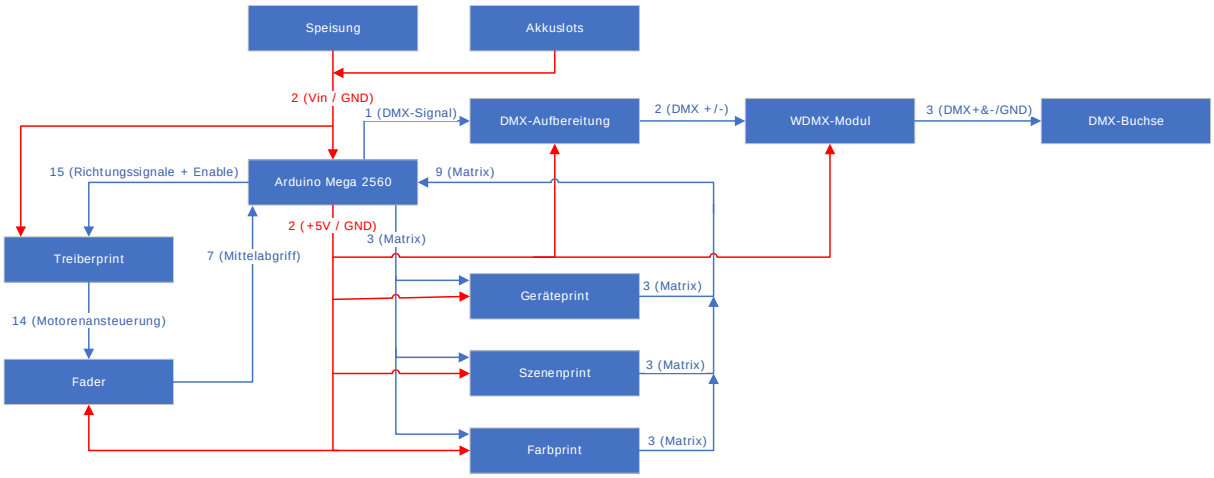

Abb. 19 Übersicht Verbindungskabel

In der Übersicht wird ersichtlich, wie die einzelnen Teile verbunden werden. Die Zahl an den Verbindungen zeigt die Anzahl der Kabel an. In den Klammern steht der jeweilige Verwendungszweck der Kabelgruppe.
Bei der Herstellung der Verbindungskabel wird darauf geachtet, dass möglichst viele Kabel zu einem Strang zusammengenommen werden können. Damit man bei späteren Arbeiten die Kabel auseinanderhalten kann, wurde folgende Farbgebung eingehalten:

Farbgebung der Kabel:

Farbe	Bedeutung
Rot	+5V oder Vin
Schwarz	Mit Rot zusammen -> GND, sonst allgemeines Signal
Weiss	Erster Pin einer Gruppe und DMX+
Blau	DMX-
Grün	Peripherie WDMX

Tab. 10 Farbgebung Verbindungskabel

4.2.5.2 Beschriftung Frontpanel

Da der Ultimaker 2 auf ein Filament (Material) pro Druck begrenzt ist, kann die Frontplatte nur einfarbig hergestellt werden. Jedoch darf die Beschriftung der einzelnen Komponenten auf der Frontplatte nicht fehlen. Für die Beschriftung einer Frontplatte gibt es zwei Varianten: Entweder alle Komponenten einzeln beschriftet oder das ganze Frontpanel auf einmal beschriften.
Die Beschriftung in der ersten Variante könnte per Hand geschehen oder zum Beispiel über einen P-Touch. Jedoch würde hierbei jede Beschriftung, welche nicht im Lot liegt sofort auffallen. Schlussendlich würde die Beschriftung sehr improvisiert wirken.
Daher wurde eine wetterfeste Folien-Etikette ausgedruckt, womit das ganze Frontpanel auf einmal beschriftet wird. Da die Folie grösser als ein A4-Blatt sein müsste, wurde auch die Beschriftung auf zwei Teile gedruckt. Durch die beiden Etiketten ergibt sich zudem der Vorteil, dass die beiden Frontplatten immer noch separat entfernt werden können.

Bei der Beschriftung wurde darauf geachtet, dass ein falsch Interpretieren, bzw. ein Missverstehen ausgeschlossen ist. Auf dem Platz oberhalb der Fader wurde die Steuerung selbst angeschrieben.

4.2.5.3 Modifikation: Tiefpass vs Pulldown

Beim Vorabtesten während der Programmierung fiel auf, dass das Einlesen der Matrizen nicht zuverlässig funktionierte. Nach längerer Suche wurde deutlich, dass die gelesenen Pins der Matrizen keine definierte Low-Position hatten. Durch das Parallelschalten eines hohen Widerstandes zum Kondensator konnte dies behoben werden.
Durch diese Massnahme war das Signal zwar definiert, jedoch entlud sich der Kondensator unregelmässig und nicht nachvollziehbar. Die Entladung konnte über eine Sekunde dauern. Das führte durch die Matrixbeschaltung dazu, dass mehrere Tasten eingelesen wurden.
Nach diversen gescheiterten Versuchen das Problem zu beheben, wurde die ursprüngliche Idee mit der Hardware-Entprellung verworfen. Anschliessend wurde die Entprellung Softwareseitig gelöst. Da die Lötstellen der Kondensatoren nun frei waren, konnten diese direkt für die Widerstände verwendet werden. Als Pulldown-Widerstand wurde ein 1 MΩ Widerstand verwendet.

4.2.5.4 Modifikation: Speisung des Treiberprints

Bei den ersten Versuchen, die Motoren in Bewegung zu setzen fiel auf, dass sich die Fader nur sehr langsam bewegen liessen. Nach verschiedenen Versuchen auf der Softwareseite, konnte diese ausgeschlossen werden. Beim Studieren der Datenblätter von den Treibern wurde der Fehler entdeckt. Bei der Realisierung des Treiberprints, wurde das Datenblatt falsch interpretiert. Die Spannung an Vs sollte gleich der Spannung an Vss sein, jedoch wurde Vs aus unbekannten Gründen an +5 V angeschlossen.
Die Lösung dieses Fehlers war relativ einfach, beim Speisungsanschluss wurde die 5 V Leitung entfernt und der +5 V Pin wurde mit der Vin Leitung verbunden. Somit haben beide Pins an den Treibern die gleiche Spannung.

4.3 Impressionen

4.3.1 Innenleben

Gehäuse:

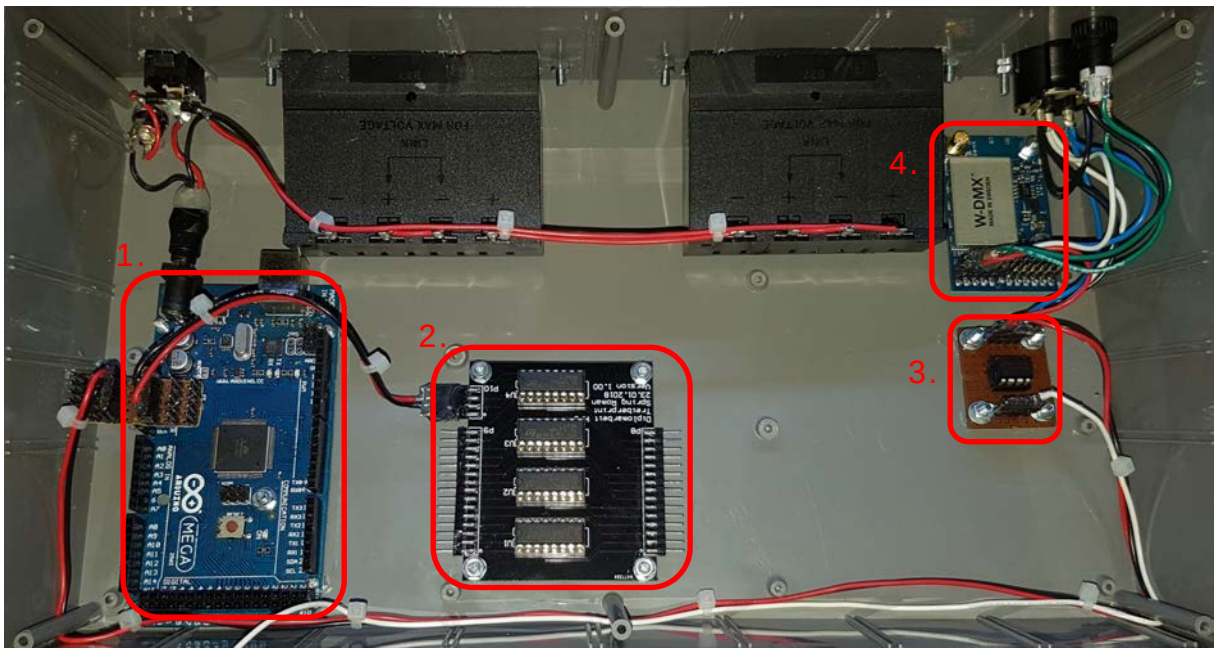

Abb. 20 Innenleben Gehäuse

1. Arduino Mega 2560
2. Motortreiberprint
3. Wandlungsprint (Stromschlaufe) für das DMX-Signal
4. WDMX-Modul

Frontpanel:

Abb. 21 Innenleben linke Frontplatte

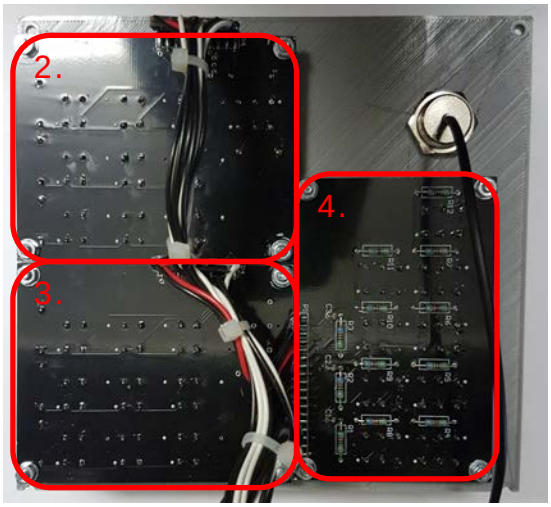

Abb. 22 Innenleben rechte Frontplatte

1. Fader
2. Farbprint
3. Szenenprint
4. Geräteprint

4.3.2 Gehäuse

Oben:

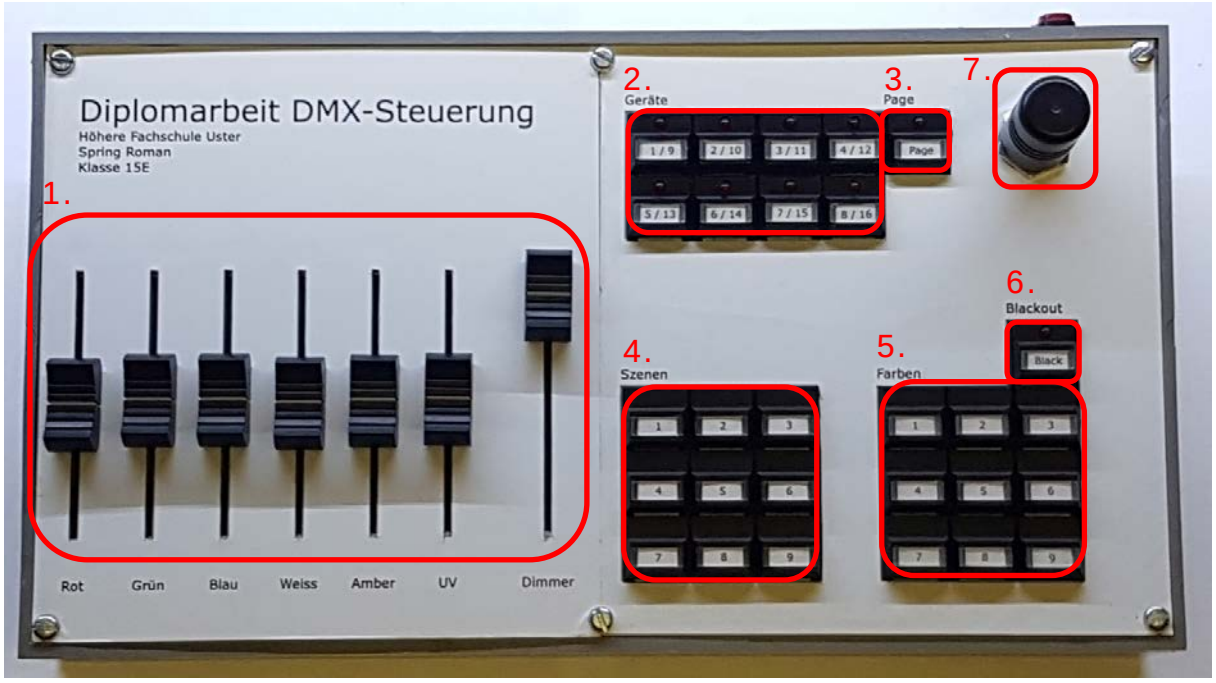

Abb. 23 Gehäuse Oben

1. Fader (Rot, Grün, Blau, Weiss, Amber, UV, Dimmer)
2. Gerätetasten
3. Pagetaste
4. Szenentasten
5. Farbtasten
6. Blackouttaste
7. Antenne

Rückseite:

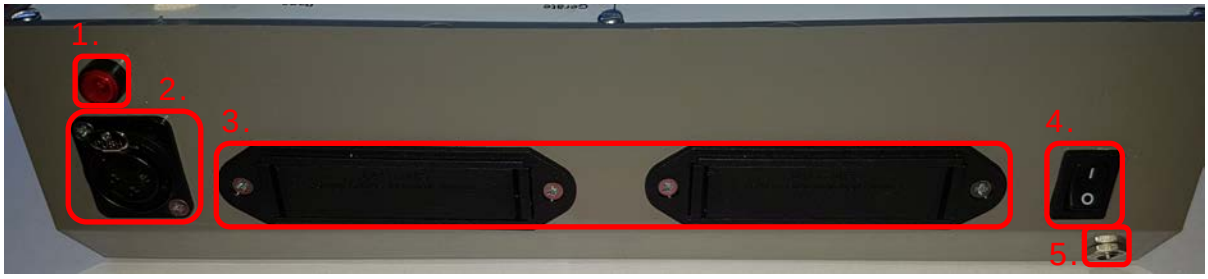

Abb. 24 Gehäuse Rückseite

1. WDMX-Taste / WDMX-Led
2. 3-Pin DMX-Ausgang
3. Akkufach
4. On / Off
5. Speisung

5 Software

5.1 Grobübersicht

Bei der Programmierung eines Arduinos sind zwei Hauptfunktionen bereits gegeben. Die „Setup“-Funktion läuft einmalig durch und wird primär für die Initialisierung des Arduinos verwendet. Nach dieser wird die „Loop“-Funktion, welche unendlich wiederholt wird, aufgerufen. In dieser befindet sich das Hauptprogramm, welches somit immer wieder ausgeführt wird.

„Setup“-Funktion

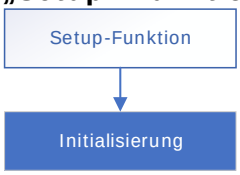

Abb. 25 Grobübersicht, „Setup“-Funktion

In der „Setup“-Funktion werden alle Initialisierungen erledigt, von den Pindefinitionen bis zu den Initialisationen der Interrupts.

„Loop“-Funktion

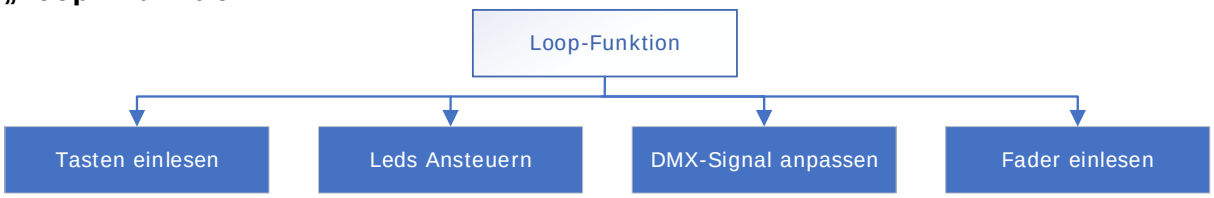

Abb. 26 Grobübersicht, „Loop“-Funktion

Die „Loop“-Funktion wiederspiegelt die Dauerwhile-Schlaufe, welches in irgendeiner Weise in jedem C-Programm anzutreffen ist. In ihr läuft das hauptsächliche Programm ab. Sie ist in diesem Projekt für mehrere Bereiche verantwortlich. Sie überprüft zum Beispiel die Tastenmatrizen und wertet diese aus. Sie steuert zudem die Leds der Tasten (Zum Beispiel signalisieren diese, welche Geräte angewählt sind). Diese Funktion ist ebenfalls für das Anpassen des DMX-Signales verantwortlich. Das Signal wird gemäss den selektierten Tasten und den Positionen der Fader angepasst. Die Überprüfung der Positionen der Fader, sowie das Einlesen der Tasten, werden ebenfalls durch diese Funktion erledigt.

Interrupt

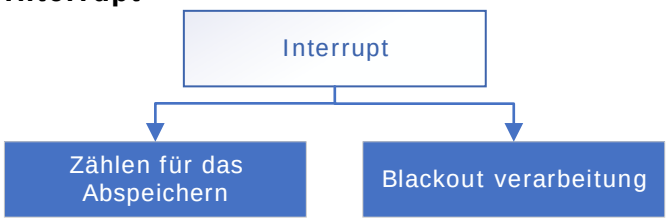

Abb. 27 Grobübersicht, Interrupts

Um dieses Projekt zu realisieren, werden zwei Interrupts verwendet. Ein Timer-Interrupt, welcher für das Auswerten der Druckdauer einer Taste verantwortlich ist. Der zweite Interrupt wird für die Blackouttaste verwendet, damit diese sofort ausgelöst wird.

5.2 Umsetzung

5.2.1 Modulübersicht

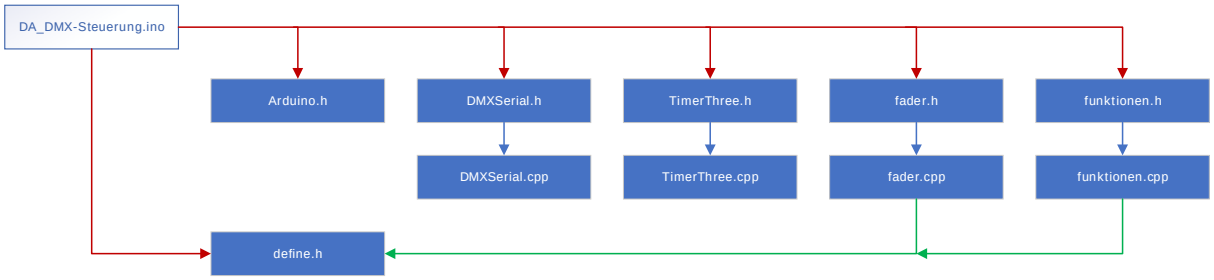

Abb. 28 Modulübersicht

Die Modulübersicht zeigt die Verknüpfung der einzelnen Module auf. Die Farben signalisieren die Hierarchie. Rot wird für die Verknüpfung des Hauptmoduls verwendet und Grün zeigt die Verknüpfung der Untermodule auf.

Die einzelnen Module haben folgende Funktionen, bzw. Inhalt:

Arduino:
Dies ist die Standardbibliothek für die Arduinobefehle.

DMXSerial:
Dieses Modul ist verantwortlich für die Initialisierung der seriellen Schnittstelle, somit kann die Schnittstelle als DMX-Ausgang verwendet werden. Auch werden Funktionen zur Verfügung gestellt, mit dessen Hilfe das DMX-Signal angepasst werden kann.

TimerThree:
Für den Timer-Interrupt wird der dritte Timer benötigt. Damit dieser Timer initialisiert und verwendet werden kann, stellt Arduino dieses Modul zur Verfügung.

Fader:
In diesem Modul werden die Funktionen programmiert, welche für die Fader verantwortlich sind. Die eine Funktion ist für das Auslesen der Position zuständig. Die andere Funktion steuert die Motoren der Fader und fährt die Fader an ihre neue Position.

Funktionen:
Dieses Modul wird für allgemeine Funktionen des Programms verwendet. Zum Beispiel befindet sich dort die Funktion für das Auslesen der Matrizen.

Define:
Damit alle Defines sich an einem Ort befinden und damit keine Defines mehrfach definiert werden müssen, werden alle Defines in diesem Header gemacht.

5.2.2 „Setup“-Funktion

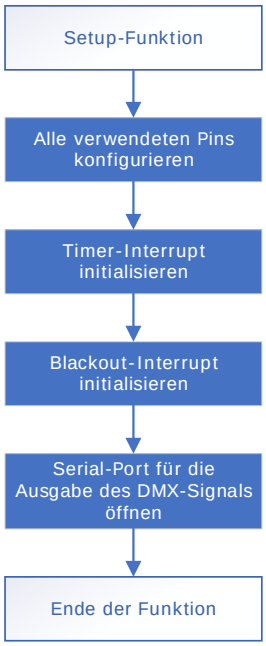

Abb. 29 „Setup“-Funktion

Zu Beginn der „Setup“-Funktion werden alle verwendeten Pins konfiguriert, bzw. als Ausgang oder Eingang gesetzt. Die Arduinobibliothek stellt dafür ein einfacher Befehl zur Verfügung. Durch diesen Befehl kann ein Pin unkompliziert definiert werden.
Nach der Konfiguration der Pins wird der Timer-Interrupt initialisiert. Der Interrupt wird so konfiguriert, dass dieser alle zehntel Sekunde aufgerufen wird. Nach dem Timer-Interrupt wird noch der Tasten-Interrupt so initialisiert, dass die Blackouttaste diesen Interrupt auslöst.

Meine Programmierkenntnisse waren nicht ausreichend um ein genügend stabiles DMX-Signal zu generieren. Deshalb wurde entschieden, dass das DMX-Signal mit der Hilfe einer externen Bibliothek über die serielle Schnittstelle ausgegeben werden soll. Durch die DMXSerial-Bibliothek ist dies möglich. Diese verwendet den Serialport 1 des Arduinoboard, um ein DMX-Signal auszugeben. Durch diese Bibliothek stehen Funktionen für die Initialisierung des Signals zur Verfügung, sowie Funktionen für das einfache anpassen des DMX-Signals. Dank dieser Bibliothek ist es möglich, ein stabiles Signal zu generieren, welches von den Messgeräten und Scheinwerfern erkennt werden kann.
Daher wird nach der Initialisierung der Interrupts mit der Hilfe dieses Moduls die serielle Schnittstelle konfiguriert, damit sie ein gültiges DMX-Signal ausgibt.

5.2.3 Interrupts

5.2.3.1 Timer-Interrupt

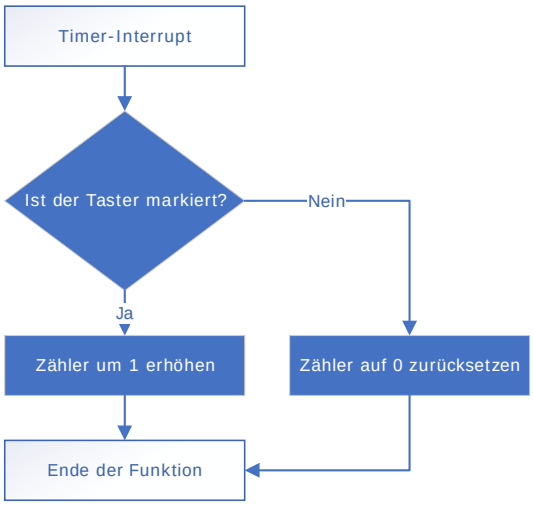

Abb. 30 Interrupt, Timer-interrupt

Der Timer-Interrupt wird alle zehntel Sekunde aufgerufen. Bei jedem Aufruf wird überprüft, ob immer noch die gleiche Taste gedrückt ist, die bei der letzten Überprüfung gedrückt war. Ist dies der Fall, wird der Zähler um eins erhöht. Durch diesen Zähler ist es in der „Loop“-Funktion möglich zu überprüfen, wie lange eine Taste bereits gedrückt wurde. Überschreitet der Zähler einen Wert von 20, wurde die Taste länger als zwei Sekunden gedrückt.
Ist keine Taste oder eine andere gedrückt, wird der Zähler auf null zurückgesetzt.

5.2.3.2 Tasten-Interrupt

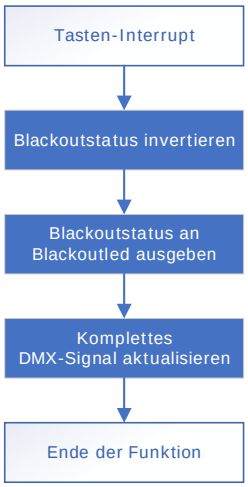

Abb. 31 Interrupt, Tasten-Interrupt

Durch ein Betätigen der Blackouttaste wird dieser Interrupt aufgerufen. Dieser invertiert als erstes den Blackoutstatus welcher bestimmt, wie das DMX-Signal angepasst wird. Damit der Nutzer auch sieht, in welchem Status er sich befindet, wird die Blackoutled ebenfalls invertiert bzw. an den Status angepasst. Anschliessend wird noch das komplette DMX-Signal angepasst.

5.2.4 „Loop“-Funktion

Die „Loop“-Funktion ist grösser und somit kann diese in der Dokumentation nicht zufriedenstellend und übersichtlich dargestellt werden. Daher wurde sie in sinnvolle Teile aufgeteilt. Zu Beginn der Funktion wird das EEPROM verarbeitet. Diese Verarbeitung wird nachfolgend in einem separaten Teil erläutert. Anschliessend entscheidet sich das Programm für eine der drei Sequenzen, die abgearbeitet werden soll. Je nachdem, was das Einlesen der Tasten ergibt wird diese Entscheidung getroffen. Diese drei Sequenzen sind ebenfalls als einzelne Teile dokumentiert. Diese Einzelteile sind jedoch im Sourcecode nicht von der „Loop“-Funktion getrennt. Das Trennen der Teile würde die Verwendung von zu vielen globalen Variablen nach sich ziehen, da die einzelne Teile viele gleiche Variablen verwenden.

5.2.4.1 Einleitung

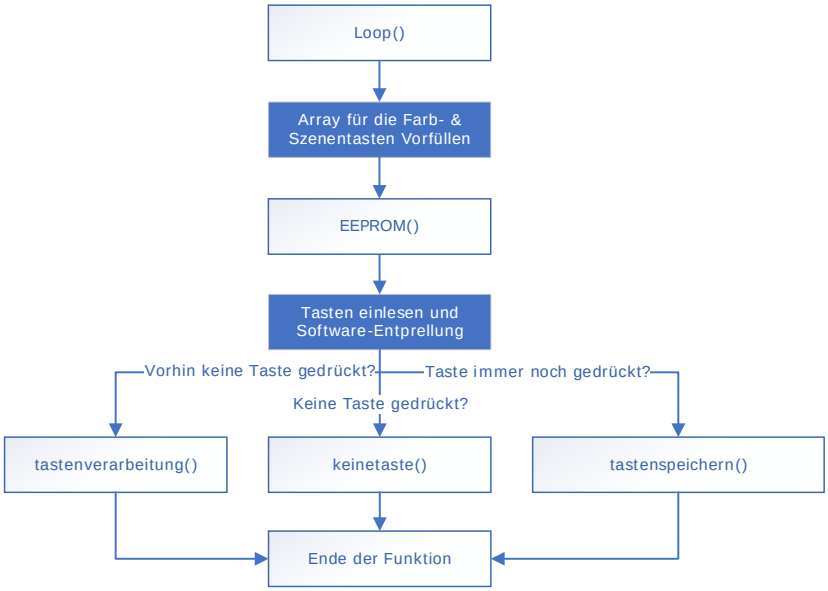

Abb. 32 "Loop"-Funktion

In der ersten Durchführung werden die Arrays für die Farben und für die Szenen erstellt. Bei der Erstellung werden die Arrays mit Defaultwerten vorgefüllt. Für die Farbarrays bedeutet dies, dass zum Beispiel alle Grundfarben abgespeichert werden. In den Arrays für die Szenen werden ebenfalls Szenen mit den verschiedenen Grundfarben gespeichert.
Der erste Schritt in der „Loop“-Funktion ist das verwalten des EEPROMs. Nach der Verwaltung des EEPROMs werden mit der Hilfe von der „matrix“-Funktion die drei Matrizen eingelesen. Das Einlesen wird zusätzlich per Software noch entprellt. Mit dem entprellten Wert wird entschieden, welche Sequenz anschliessend abgearbeitet werden soll.

5.2.4.2 EEPROM

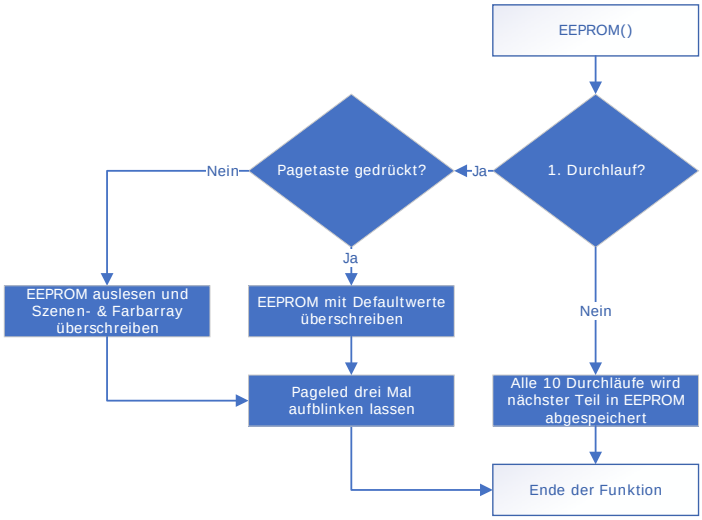

Abb. 33 EEPROM

Beim ersten Durchlauf wird entschieden, ob das EEPROM ausgelesen wird oder mit den Defaultwerten zurückgesetzt werden soll. Dies wird durch das Auswerten der Pagetaste entschieden. Ist sie bei dem Start gedrückt, wird das EEPROM zurückgesetzt. War die Pagetaste nicht gedrückt, wird das EEPROM ausgelesen, somit können die Werte im Programm verwendet werden. Nach dieser ersten Verarbeitung des EEPROMs, wird die Pageled so angesteuert, dass sie dreimal blinkt. Das Blinken signalisiert dem Benutzer, dass die Startup-Routine fertig ist und die Steuerung nun verwendet werden kann.

Nach dem ersten Durchlauf wird alle zehn Durchläufe ein Teil des EEPROM geupdatet. Da das EEPROM für das Schreiben eines Bytes 3.3 ms benötigt, wird zuerst geprüft ob sich der gespeicherte Wert von dem zu speichernden Wert unterscheidet. Da die Routine für das komplette EEPROM viel zu lange benötigen würde, wird immer nur einen Teil geupdatet. Somit wird nicht das komplette Programm aufgehalten
Einer dieser Teile updatet jeweils eine Faderposition einer Farbtaste und eine Faderposition aller Geräte einer Szenentaste. Da es sieben Fader und jeweils neun Tasten sind und alle zehn Durchgänge ein Update gemacht wird, ist das EEPROM alle 490 Durchläufe einmal komplett geupdatet worden. Einer dieser kleineren Updates besteht aus 17 Bytes (Szene: 16 Geräte à einem Byte, Farbe: ein Byte). Im Worstcase dauert das Schreiben eines Teiles 56.1 ms (17 Bytes x 3.3 ms). Das Abspeichern des kompletten EEPROMs benötig daher ca. 2750 ms. Mit dem restlichen Programm ergibt es eine Dauer von 3 s (Angenommener Wert). Da der Benutzer jedoch nie so schnell alle Farb- und Szenentasten überspeichern kann, ist der Speicheraufwand des EEPROMs jedoch immer geringer.

5.2.4.3 Tastenverarbeitung

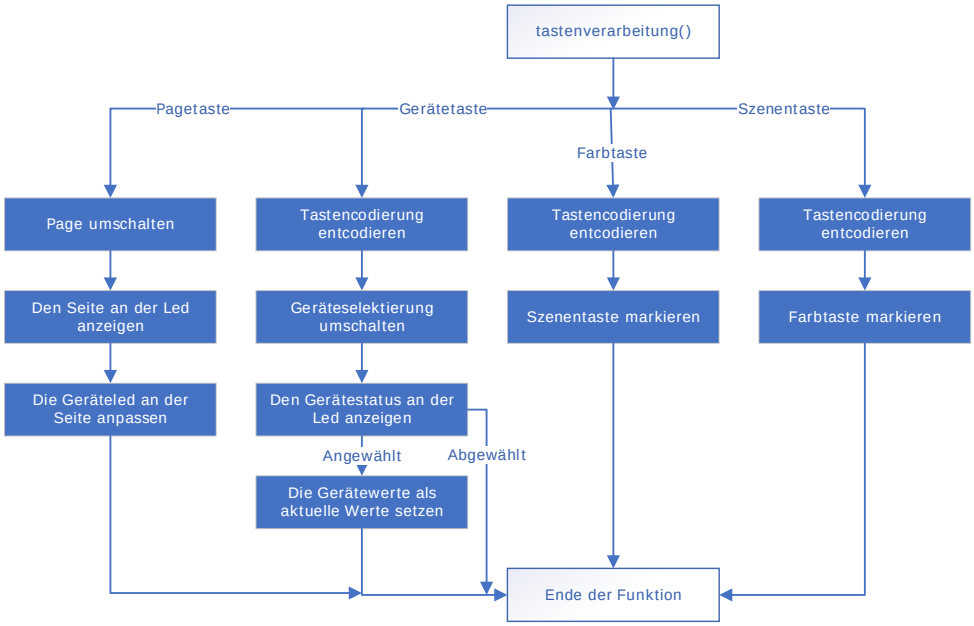

Abb. 34 Tastenverarbeitung

Der erste Teil der „Loop“-Funktion wertet den Tastencode der „matrix“-Funktion aus. Handelt es sich bei der Codierung um ein Farb- oder Szenetaste, wird diese als erstes decodiert. Anschliessend wird diese Taste als unbearbeitet markiert. Mit dieser Markierung wird in den anderen Teilen entschieden, was geschieht oder wie lange die Taste bereits gedrückt wurde.
Ist die Taste in der Codierung eine aus der Gerätematrix wird geprüft, ob es sich um die Pagetaste handelt. Falls die Pagetaste gedrückt wurde, wird die Seite umgeschaltet und es werden die Geräteleds an der Seite angepasst. Wurde eine der acht Gerätetasten gedrückt, wird wie bei den Farb- und Szenentasten die Codierung aufgelöst. Durch das Auflösen ist die genaue Gerätetaste bekannt und das Gerät wird selektiert oder deselektiert. Ebenfalls wird die Geräteled gemäss der Selektierung umgeschaltet. Wenn das Gerät angewählt wurde, werden alle Fader auf die gespeicherten Gerätewerte gefahren. Dieses Aktualisieren der Faderposition wird beim Abwählen übersprungen.

5.2.4.4 Keine Taste gedrückt

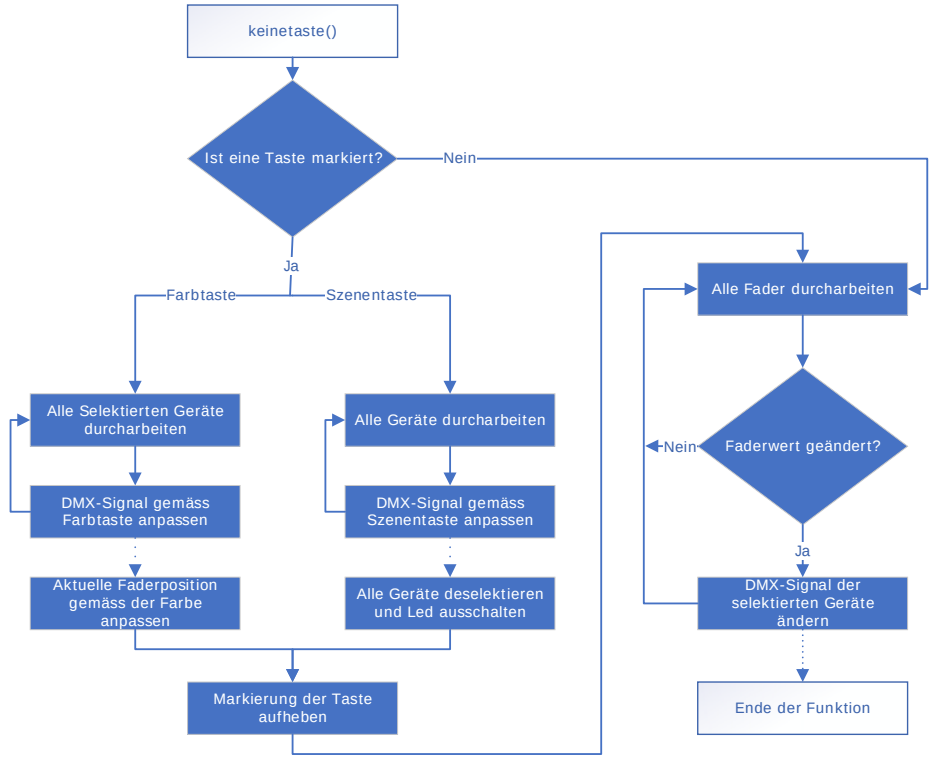

Abb. 35 Keine Taste gedrückt

Der zweite Teil der „Loop“-Funktion wird aufgerufen, wenn bei diesem Durchgang keine Taste eingelesen wurde. Diese Funktion ist in zwei Abschnitte aufgeteilt, der erste Verarbeitet das Betätigen der Farb- und Szenentasten. Dieser wird ausgeführt, wenn eine Taste markiert ist, bzw. wenn im vorgängigen Durchlauf eine Taste gedrückt wurde (Taste wird bei der negativen Flanke verarbeitet). Ist keine Taste markiert, wird der komplette erste Teil übersprungen. Falls jedoch eine Taste markiert ist, wird, je nachdem ob es sich um eine Farbtaste oder eine Szenetaste handelt, die Farbe oder die Szene angepasst. Wird eine Farbe aufgerufen, werden alle Geräte durchgearbeitet. Bei den selektierten Geräten wird der Speicher und das DMX-Signal gemäss der Farbe angepasst. Nach dem Anpassen, werden noch die Fader gemäss der Farbe neu gesetzt.
Handelt es sich um einen Szenenaufruf werden alle Geräte, ob selektiert oder nicht, gemäss den Szenenwerten überschrieben. Nach dem Anpassen, werden alle Geräte deselektiert.
Egal ob eine Farb- oder Szenentaste gedrückt wurde, wird nach dem Anpassen bei der entsprechenden Taste die Markierung aufgehoben.
Der zweite Teil ist für die Verarbeitung der Fader verantwortlich, dieser wird im Gegensatz zum ersten Teil immer ausgeführt. In diesem Teil werden alle Fader durchgearbeitet und ihre Positionen eingelesen. Wenn sich eine Position geändert hat, wird das DMX-Signal entsprechen der selektierten Geräte angepasst.

5.2.4.5 Tasten überschreiben

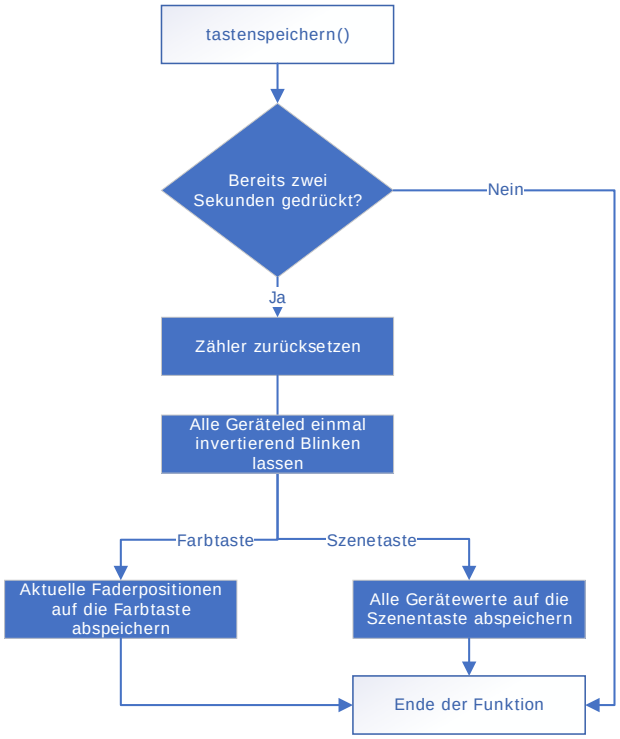

Abb. 36 Tasten überschreiben

Ist eine Taste länger als ein Programmdurchgang gedrückt, wird dieser dritte Teil der „Loop“-Funktion aufgerufen. Mit dem Zähler, welcher im Timer-Interrupt erhöht wird, kann geprüft werden, wie lange die Taste bereits gedrückt ist. Falls die Taste länger als zwei Sekunden gedrückt wurde, wird dieser Teil ausgeführt und nicht übersprungen. Als erstes wird der Zähler, welcher für die Abfrage der Druckdauer benötigt wird, zurückgesetzt. Anschliessend wird dem Benutzer signalisiert, dass das Abspeichern erfolgreich war und somit auch die zwei Sekunden um sind. Diese Signalisation geschieht mit Hilfe der Geräteleds, welche alle invertiert werden und nach einer halben Sekunde wieder auf ihren ursprünglichen Zustand zurückgesetzt werden. Nach der Signalisation werden die Faderwerte abgespeichert, falls die gedrückte Taste eine Farbtaste war. War es eine Szenetaste, wird die komplette Szene abgespeichert, sprich alle Werte von allen Geräten.

5.2.5 Externe Funktionen

5.2.5.1 „setDMX“-Funktion

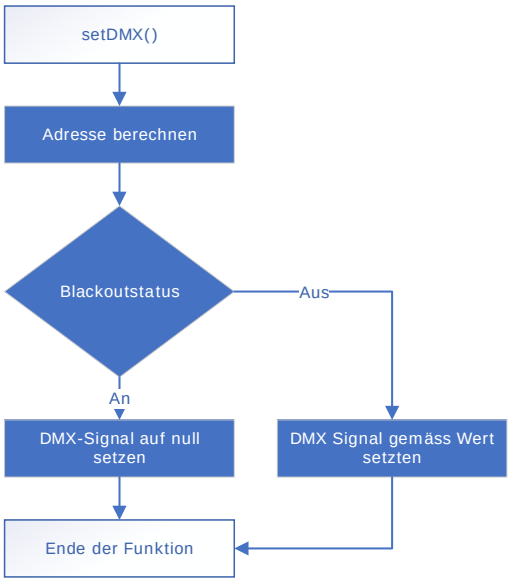

Abb. 37 "setDMX"-Funktion

Diese Funktion ist dafür zuständig, dass ein Slot des DMX-Signals angepasst wird. In einem ersten Schritt werden die Adressen, welche geändert werden sollen ausgerechnet. Anschliessend wird der Wert, falls der Blackoutstatus aus ist, mit dem neuen Wert überschrieben. Ist der Blackoutstatus an, wird statt dem neuen Wert eine Null geschrieben.

5.2.5.2 „matrix“-Funktion

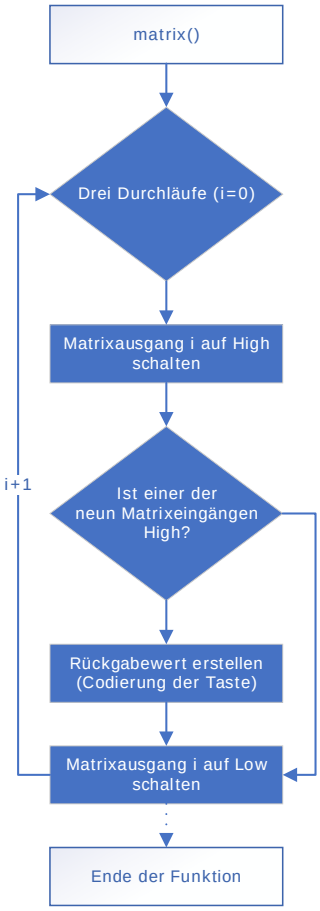

Abb. 38 "matrix"-Funktion

Die „matrix“-Funktion liest die drei Matrizen ein. Sie läuft dreimal durch, in jedem Durchgang wird der nächste Matrixausgang eingeschaltet und die Anderen werden ausgeschaltet. Durch das Verbinden der Matrixausgänge, können nun in einem Durchgang neun Eingänge überprüft werden. Liegt an einem Eingang ein Signal an, wird für diese Taste ein Code generiert. Als Code wird eine Zahl erstellt, welche im späteren Programm wieder decodiert werden kann.

Die Codierung sieht folgendermassen aus:

	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**
00	G1	G2	G3	S1	S2	S3	F1	F2	F3
10	G4	G5	G6	S4	S5	S6	F4	F5	F6
20	G7	G8	G9	S7	S8	S9	F7	F8	F9

G: Gerätetaste
S: Szenentaste
F: Farbtaste

Codierung: Die Zehnerstelle mit der Einerstelle addieren
Beispiel: Szenentaste 5 -> 15

Für diese Art der Einlesung, stellt sich die Frage, ob die erste oder letzte eingelesene Taste verwertet wird. Da keine der beiden Varianten einen relevanten Vorteil hat, wurde gemäss Bauchgefühl bestimmt, dass jeweils die erste erfasste Taste verarbeitet wird.

5.2.5.3 „getfader“-Funktion

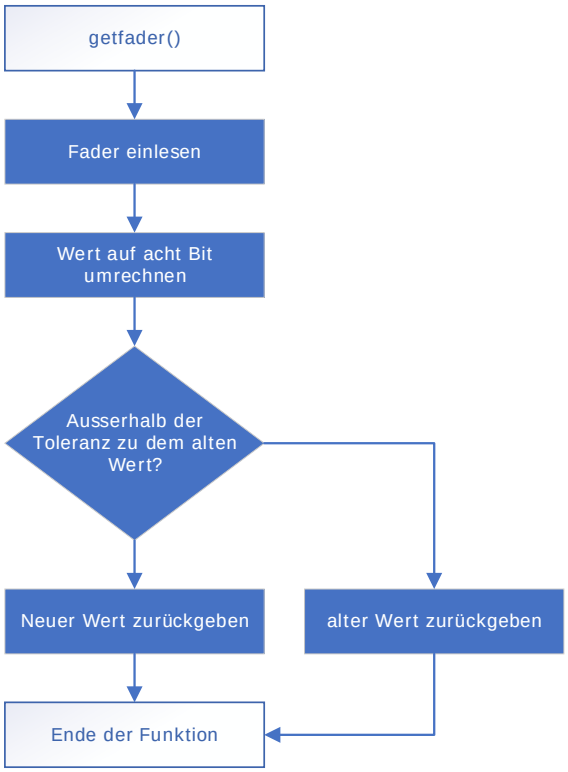

Abb. 39 "getfader"-Funktion

Die erste Funktion, welche auf die Fader zugreift ist die „getfader“-Funktion. Diese liest einen bestimmten Fader aus und rechnet den ausgelesenen 10 Bit Wert in einen 8 Bit Wert um. Dank dieser Umrechnung kann anschliessend im ganzen Programm mit 8 Bit gearbeitet werden. Die Arbeit mit 8 Bit spart zum einen Platz bei dem Abspeichern der Werte. Zum anderen spart das restliche Programm die andauernde Umrechnung, da der Wertebereich des DMX-Signals ebenfalls 8 Bit gross ist.
Nach dem Einlesen und dem Umrechnen wird noch geprüft, ob sich der neue Wert ausserhalb eines Toleranzbereichs befindet. Diese Überprüfung ist notwendig, da der eingelesene Wert sehr schwanken kann. Die Schwankung kommt durch die Potentiometer zustande, da diese ihren Widerstand stufenlos verstellen, haben diese keinen definierte Position. Dies hätte die Auswirkung, dass die Einlesefunktion je nach Position des Faders, bei jedem Durchgang einen anderen Wert einliesst.

5.2.5.4 „setfader“-Funktion

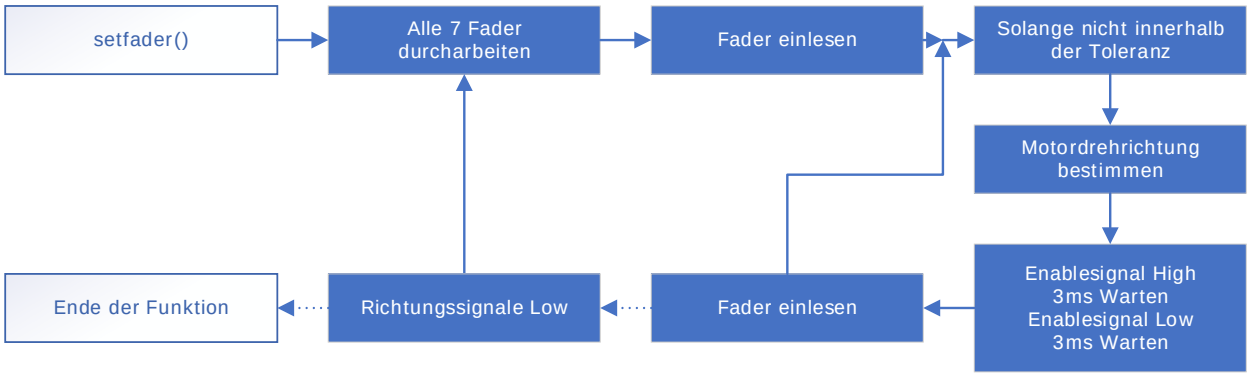

Abb. 40 "setfader"-Funktion

Um die Fader an eine Position zubewegen gibt es zwei Varianten, entweder das sich alle auf einmal bewegen oder nach einander. Durch die Bewegung aller Fader auf einmal entsteht eine enorme Stromspitze, welche nicht erwünscht ist. Daher wurde die zweite Variante realisiert, in welcher einer nach dem anderen platziert wird.
Während dem Programmieren wurde festgestellt, dass die Fadermotoren zu schnell sind, um rechtzeitig gestoppt zu werden. Daher wurde ein PWM programmiert, mit welchem die Fader ein kleines Stück bewegt werden und nach jeder Bewegung wieder überprüft werden. Dies wird solange wiederholt, bis sich der Fader innerhalb der Toleranzgrenze befindet.
Die „Korrekturwiederholung“ ist in dem Flussdiagramm nicht ersichtlich. Diese wiederholt die ganze Funktion viermal hintereinander. In jedem Durchgang wird die Toleranz, in welchem sich der Fader befinden muss, kleiner. Damit diese Toleranz auch eingehalten werden kann und der Fader nicht über sein Ziel hinausfährt, wird auch die Geschwindigkeit jedes Mal verringert. Die Geschwindigkeit wird verkleinert, in dem das Enablesignal kürzer auf High gesetzt wird. Ist der Fader jedoch bereits innerhalb der Toleranzgrenze, wird der Durchgang übersprungen.

6 Kontrolle

6.1 Vergleich Pflichtenheft

Detaillierte Beschreibungen der einzelnen Punkte können unter dem Kapitel „1.8 Zielsetzungen aus Anforderungen“ gefunden werden.

Muss-Kriterien:

#			Prio.	Bemerkung
1		**Dokumentation**		
	1	Erstellung einer kompletten Dokumentation	1	-
	2	Erstellung eines Testkonzepts	2	-
	3	Dokumentiertes Testen	3	-
	4	Erstellung einer Kurzbedienungsanleitung	2	Ok
2		**Hardware**		
	1	16 Geräte auswählbar	1	Ok
	2	6 Motorfader	1	Ok
	3	9 Farb- & Szenentasten	2	Ok
	4	1 Blackouttaste	2	Ok
3		**Software**		
	1	Gerät anwählbar	1	Ok
	2	Auswertung und Ansteuerung der Motorfader	1	Ok
	3	Farb- & Szenentasten abspeicherbar & abrufbar	2	Ok
	4	Funktion Blackouttaste	3	
	5	Ausgabe eines Normgerechten DMX-Signal	1	Ok

Tab. 11 Muss-Kriterien Vergleich mit Pflichtenheft

Soll-Kriterien:

#			Prio.	Bemerkung
1		**Dokumentation**		
2		**Hardware**		
	1	Gehäuse	1	Ok
	2	Wireless DMX Modul	1	Ok
	3	Akkubetrieb	2	Ok
3		**Software**		

Tab. 12 Soll-Kriterien Vergleich mit Pflichtenheft

6.2 Praxistest

Die DMX-Steuerung wird mittels einer im Vorfeld erstellten Excel-Liste (Siehe Anhang) geprüft. Da das Überprüfen gewisser Testpunkte abhängig von anderen Punkten ist, zum Beispiel kann das Verarbeiten der Fader nur mithilfe der DMX-Ausgabe geprüft werden. Daher wird die Steuerung einem Praxistest unterzogen, wobei alle Punkte gemeinsam überprüft werden.

Testaufbau:
Damit das Testen so realitätsnahe wie möglich ist, werden jeweils vier Scheinwerfer von vier verschiedenen Typen verwendet. Davon sind acht Scheinwerfer mit Kabel verbunden und acht Scheinwerfer per Funk.

Scheinwerfer	Anzahl DMX-Kanäle	Anschlussart
Showtec Spot 6 Q5	5	Kabel
Showtec Club PAR 12/6	6	Kabel
Showtec Archi Painter 24/8 Q4	4	Funk
Elation Volt Q3	4	Funk

Tab. 13 Testaufbau

Für die Überprüfung werden die Scheinwerfer so adressiert, dass jede Gerätetaste einen Scheinwerfer wiederspiegelt. Der komplette Test wird in einem Raum durchgeführt, da nicht das WDMX-Modul auf die Reichweite getestet wird, sondern die Steuerung selbst. Das Resultat ist in dem Testkonzept, welches sich im Anhang befindet, ersichtlich.

Es ist neben diesem Test auch ein Feldtest vorgesehen, dieser findet jedoch erst nach der Abgabe der Dokumentation statt.

6.3 Fehler

6.3.1 Ausstehende Fehler

In diesem Projekt stehen keine Fehler aus, es wurden alle Fehler behoben.

6.3.2 Behobene Fehler

6.3.2.1 Letzte DMX-Adressen

Problem:
Durch die Toleranzen im Timing der DMX-Übertragung kommt es vor, dass der letzte Slot zum Teil von den Scheinwerfern bzw. Messgeräten nicht akzeptiert wird. Damit die Steuerung komplett verwendet werden kann bedarf es mindestens 96 Adressen.
Lösung:
Um dieses Problem zu beheben, wurde die Anzahl der gesendeten Slots von 96 auf 100 erhöht. Dadurch sind die ersten 96 sicher stabil.

6.3.2.2 WDMX Modul

Problem:
Es ist nicht möglich, dass Modul von dem Empfangsmodus in den Sendemodus umzustellen. Obwohl die Beschaltung des Moduls richtig ist.
Lösung:
Durch mehrere E-Mails mit dem Hersteller in Schweden kam heraus, dass diese anscheinend in der neusten Generation noch gewisse Probleme haben, welche vereinzelt auftreten. Das Problem konnte jedoch schnell behoben werden, in dem man den Pin 8, welcher eigentlich an +5 V angeschlossen wird, nicht anschliesst. Dadurch hat der Empfangsteil auf dem Modul keinen Strom und das Modul startet in dem Sendemodus.

6.4 Messungen

6.4.1 DMX Signal

DMX-Messgerät:

Um das DMX-Signal auf seine Richtigkeit zu prüfen gibt es spezielle Messgeräte. Diese zeigen an, wie gut das Signal ist, bzw. wie das Signal aufgebaut ist. Auch zeigen diese Messgeräte die einzelnen DMX-Werte bei den Adressen an.

Daten Format:

Abb. 41 DMX-Signal Format

Abb. 42 DMX-Signal Timing

Abb. 43 DMX-Signal Pegel

Abb. 44 DMX-Signal Werte

Diese Messungen wurden mit Hilfe des DMX Tester von Showtec gemacht. Der DMX Tester wird über ein DMX-Kabel mit der Steuerung verbunden. Durch die Messung kamen folgende Werte heraus.

Messwert	Erwartung	Messung	Bemerkung
Anzahl Adressen	100 CH	99 - 100 CH	Letzte Adresse wird zum Teil nicht erkannt.
Start- / Stoppbit	1 Bit/2 Bits	1 Bit/2 Bits	Mit „DMX-512 Pocket Tester“ geprüft (PT-3512A)
Break	>=88 us	100 us	
MAB	>=8 us	15 - 16 us	
Slotdauer	43 - 45 us	44 - 45 us	
Framedauer	~5 ms	5 ms	
Signal Pegel	> 1.5 V, <5 V	3.72 V	
Werte pro Adresse	0 - 255	0 - 255	

Tab. 14 Messresultate DMX Tester

Die Messung zeigt, dass alle Werte in den erwarteten Bereichen liegen. Zwar wird die letzte Adresse zum Teil nicht erkannt, jedoch durch die Korrektur von 96 auf 100 Channel ist dies irrelevant. Die Steuerung benötigt nur die ersten 96 Adressen und diese sind stabil

Oszilloskop:

Um die Dauer eines Frames genau zu messen, bzw. die Dauer der einzelnen Bestandteile des Signales, wird ein Oszilloskop verwendet. Mit dem Oszilloskop wird der Ausgang des Arduinos (Digital Pin 1) gemessen. Das verwendete Oszilloskop ist ein „PicoScope 2206A“, welches mit der PC-Software Picoscope 6 ausgewertet wurde.

Periodenmessung:

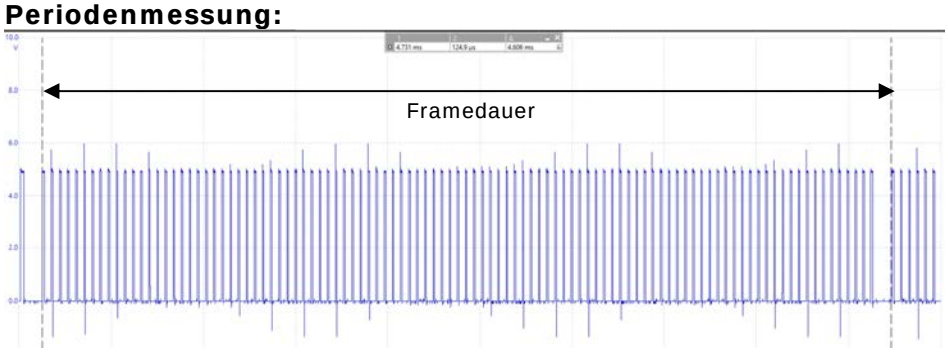

Abb. 45 Periodenmessung Oszilloskop (500 us / Div)

Messung der Bestandteile:

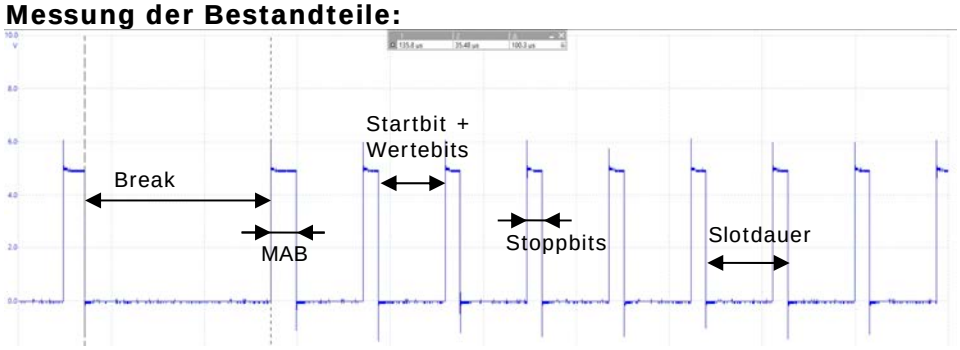

Abb. 46 Messung der Initialisierung Oszilloskop (50 us / Div)

Mit der zweiten Messung, welche nur die Initialisierung und die ersten paar Slots zeigt, konnte die Messung der einzelnen Bestandteile vorgenommen werden.

Messwert	Erwartung / Norm	Messung	Bemerkung
Break	>=88 us	100.6 us	
MAB	>=8 us	13.6 us	
Start- & Wertebits	~36 us	36.3 us	
Stoppbits	>=8 us	8 us	
Slotdauer	43 - 45 us	44 us	
Framedauer	~5 ms	4.6 ms	

Tab. 15 Messresultate Oszilloskop

Auch in dieser Messung sind alle Werte in dem Bereich, welcher durch die Norm vorgegeben wurde. Der Vergleich mit den Normwerten konnte belegen, dass das DMX-Signal normgemäss versendet wird.

6.4.2 Stromverbrauch

Um den gesamten Stromverbrauch der Steuerung zu ermitteln, werden die Akkuzellen entfernt und die gesamte Stromaufnahme der Steuerung gemessen. Für diese Messung wurde ein Multimeter zwischen der Speisung und der Diode angeschlossen. Um den Strom zu messen ohne dass die Spannung absinkt, wird für die Messung ein Netzteil verwendet, welches 1 A liefern kann.
Jede Messung wird fünfmal durchgeführt und am Ende wird das Messresultat gemittelt. Bei den Messungen mit den Motoren wird darauf geachtet, dass sich alle Fader von dem einen Anschlag zum anderen Bewegen.

Messung	Resultat (Mittel)
Standby (Alle Led aus)	264 mA
Standby (Alle Led an)	318 mA
Motorenbewegung (Alle Led aus)	493 mA
Motorenbewegung (Alle Led an)	532 mA

Tab. 16 Messresultate Multimeter

Weil die maximale Stromaufnahme grösser als 0.5 A ist, genügt das angedachte Netzteil mit seinen maximalen 0.5 A nicht aus. Da das grössere Netzteil nur 5.- SFr. mehr kostet, wird das Alte durch dieses ersetzt. Somit kommt das Netzteil nie an seine Grenzen, was zusätzlich dessen Lebensdauer erhöht.

Durch die Messung konnte eine theoretische Akkudauer von 9.6 h ausgerechnet werden. Dieser Wert erreicht man, wenn davon ausgegangen wird, dass ungefähr die Hälfte der Leds durchgehend leuchten. Der Strom für die Motoren wird für diese Berechnungen weggelassen, da dieser nur sehr kurz auftritt.

$$Dauer = \frac{Akkukapazität}{\frac{Standbystrom\ 1 + Standbystrom\ 2}{2}} = \frac{2800\ mAh}{291\ mA} = 9.6\ h$$

Das Ziel, dass der Akku 6 – 8 h hält, wird somit ebenfalls erreicht und zumindest in der Theorie übertroffen. Die tatsächliche Akkulaufzeit, wird sich erst in der Praxis zeigen, da nicht vorhersehbar ist, wie viele der Led leuchten oder wie viele Male die Fader durch die Motoren bewegt werden. Auch wenn die Akkudauer schlechter ausfällt, liegt die Laufzeit immer noch in dem angestrebten Bereich. Wenn man den schlimmsten Fall betrachten würde, sprich alle Leds leuchten und die Motoren fahren durchgehend, ergäbe es immer noch eine Akkudauer von ca. 5.2 h. Dieser Worstcase wird jedoch in der Realität nie eintreffen.

7 Zukünftige Erweiterungen

Folgende Erweiterungen könnten für die Zukunft weiterverfolgt werden:

- Die Möglichkeit, dass der Benutzer die Startadressen der einzelnen Geräte selber bestimmen kann. Um dies Softwareseitig umzusetzen, müsste mindestens eine Taste für das erhöhen der Adresse vorhanden sein. Zusätzlich wären drei 7-Segmentanzeige, für das Anzeigen der Adressen, sehr nützlich.

- Eine weitere sehr interessante Erweiterung wäre die Verknüpfung mehrerer Steuerungen. Der Vorteil wäre, dass wenn bei der einen Steuerung die Fader bewegt werden, die anderen Steuerungen diese Werte ebenfalls übernehmen. Das WDMX-Modul, sowie auch der Stromschlaufen-Baustein können senden wie auch empfangen, daher müsste nur die Beschaltung die Module verändert werden. Durch das Umschalten der Bausteine durch die Software, wäre grundsätzlich eine Kommunikation möglich.

- Da gewisse Events unter freiem Himmel stattfinden, wäre eine wasserfeste oder zumindest eine spritzwasserfeste Steuerung von enormem Vorteil. Dies würde eine andere Art von Frontplatte bedeuten. Zumindest müssten alle Komponente, die nach Aussen führen, ersetzt werden.

- Eine Möglichkeit, dass die Akkuzellen im Netzbetrieb kontrolliert geladen werden. Durch das Laden im Netzbetrieb müssten die Akkus nicht mehr für das Aufladen entfernt werden.

- Das momentane Gehäuse ist ein typisches Conrad-Produkt: schlechtes Material, ungenau und war bereits bei der Ankunft leicht verformt. Daher wäre eine weitere Überlegung für die Zukunft, dass entweder ein besseres Gehäuse eingekauft wird oder ein eigenes Gehäuse gebaut wird.

- Viele DMX-Steuerungen haben eine Funktion, in welcher die Scheinwerfer mit der Hilfe des Sounds geändert werden. Um dies zu realisieren, müsste ein Mikrofon eingebaut werden und dieses über einen analogen Eingang eingelesen werden. Zusätzlich müsste noch ein Potentiometer eingebaut werden, welches für die Einstellung der Empfindlichkeit des Mikrofons wäre.

8 Anhang

8.1 Technisches Fazit

Für mich war es das erste Mal, dass ich ein Projekt absolviert habe, in welchem ein komplettes Gerät von Grund auf gebaut habe. Bis anhin hatte ich höchstens gewisse Modifikationen an bestehenden Produkten vorgenommen. Daher habe ich während der Diplomarbeit vor allem hinsichtlich der Realisierung der Hardware viel gelernt. Zum Beispiel, dass man zuerst alles durchplanen sollte und erst dann die Materialien bestellt. Mir passierte es, dass ich mehrere Bestellungen tätigen musste, da mir bei dem Zusammenbau immer wieder fehlende Komponenten aufgefallen sind. Da mir die nötige Erfahrung fehlte, gingen gewisse Materialien vergessen. Zum Beispiel hatte die erste Zwischenversion keinen Schalter um das Gerät auszuschalten.

Ich wollte zu Beginn die Leiterplatten mit Veroboards realisieren. Zum Glück hatte mich Daniel Ringgenberg eines Besseren belehrt und mir ans Herz gelegt, dass ich die Prints layouten soll. Durch seine Hilfe, bei der Einarbeitung in das Layoutprogramm und das generieren der Gerberdaten, gelang das realisieren der Platinen auch problemlos. Diese Entscheidung war nicht nur für das Projekt von Vorteil, da es nicht nach einer Bastelarbeit aussieht. Sondern war es auch für mich als Repetition von Vorteil, da ich seit meinem zweiten Lehrjahr nicht mehr gelayoutet habe.

Die Programmierung der Software war der leichteste Teil der Diplomarbeit. Durch ein paar Probleme während der Programmierung, zog sich die Realisierung der Software allerdings in die Länge.

Der schwierigste Teil war für mich die Dokumentation. Durch meine Schwäche in Sprache und Schrift fiel mir das Formulieren und das Überarbeiten extrem schwer, auch zog sich die Arbeitszeit enorm in die Länge.

8.2 Persönliches Fazit

Aufgrund von gewissen Schulmodulen kam es zu Verzögerungen der Diplomarbeit. Wegen den Verzögerungen kam ich in den letzten paar Wochen ziemlich in Zeitnot. Zum Glück war in diesen Wochen im Betrieb nicht viel los, somit konnte ich dort einen Teil der Zeit wieder aufholen. Nur durch dies war es mir möglich, dass ich innert vier Woche 180 Stunden aufarbeiten konnte. Unter dieser enormen Investition litt jedoch die Motivation enorm, da viele dieser Stunden bis tief in die Nacht gingen.

Die Diplomarbeit hat mir im Grossen und Ganzen viel Spass gemacht, dennoch bin ich froh, wenn die Arbeit und somit die Schule vorbei sind.

8.3 Kostenanalyse

Durch die Auflistung der effektiven Kosten in der Stückliste ist ersichtlich, dass sich die Materialkosten dieses Projekt auf 357.- SFr. (gerundeter Wert) beziehen. Somit sind die geplanten Kosten von 400.- SFr. nicht erreicht worden.

8.4 Zeitplananalyse

Im Anhang befindet sich die Auflistung der Zeitplanung mit Soll/Ist Vergleich.

Während der Projektzeit gab es einige Abweichungen von der Zeitplanung, die grössten Abweichungen werden in den nachfolgenden Punkten erläutert:

13.11.17 - 08.12.17:
In dieser Zeitspanne musste die verfügbare Zeit in anderen Schulprojekten (Primär Sensorik 2), bzw. in die vorweihnachtlichen Prüfungen eingesetzt werden. Zusätzlich fanden in dieser Zeit praktisch jedes Wochenende Events statt, an welchen ich für meinen Betrieb arbeiten musste.

09.12.17 – 31.12.17:
Über die Weihnachtsferien war für die Diplomarbeit relativ viel geplant. Jedoch kam die Arbeit in dieser Zeit nur langsam voran. Dies kam daher, dass die Energie für die Schule langsam erschöpft war. Zusätzlich kam noch die zweite Projektarbeit (Sensorik 2) dazu, welche über die Weihnachtsferien erledigt werden musste und sehr viel Zeit in Anspruch nahm.
Durch die Freude an der Programmierung konnte ich immerhin der Sourcecode zu einem Grossteil in den Ferien schreiben. Da jedoch noch praktisch nichts von der Hardware vorhanden war, wurde der ganze Code im Freiluftstyle programmiert. Dies bedeutet, dass programmiert wird ohne zu wissen ob das Programmierte überhaupt mit der Hardware zusammenpasst.

27.01.18 – 10.02.18:
Da die Programmierung frühzeitig fertig war und die Hardware ebenfalls früher als geplant fertig wurde, konnten die Tests vorgezogen werden. So konnte die Arbeit am Projekt früher beendet werden. Durch die frühe Beendigung war mehr Zeit für die Überarbeitung der Dokumentation vorhanden.

8.5 Auswertung Meilensteine

Vorarbeit:
Dieser Meilenstein wurde erfolgreich eingehalten.

Realisierung:
Dieser Meilenstein wurde nicht erreicht, da durch die Verzögerungen (Siehe Kapitel „8.4 Zeitplananalyse“) sich alle Arbeiten nach hinten verschoben haben.

Projektende:
Auch das Projektende konnte erfolgreich eingehalten werden.

8.6 Stundenanalyse

Tätigkeit	Soll-Aufwand [h]	Ist-Aufwand [h]
Dokumentation	130	119
Hardware	25	75
Software	100	97
Präsentation	(10)	Noch ausstehend
Total	**255** (265)	**291**

Tab. 17 Stundenanalyse, Aufwandsvergleich

Durch regelmässiges eintragen der Arbeitszeit konnte der Ist-Aufwand über die ganze Projektarbeit genau erfasst werden. Die Einschätzung des Zeitaufwandes für die Dokumentation wurde relativ genau eingehalten. Die geschätzte Zeit für die Programmierung der Software stimmte ebenfalls gut mit der benötigten Zeit überein. Jedoch schoss die Zeit, welche für das Arbeiten an der Hardware verwendet wurde, enorm in die Höhe. Die ursprünglich geplanten 25 h wurden bei weitem überschritten. Dies zeigt, dass bei dem nächsten Projekt der Hardwareaufwand nicht zu unterschätzen ist. Aufgrund des Mehraufwands bei der Hardwarerealisierung, wurde über das ganze Projekt gesehen ca. 15% mehr Zeit benötigt.

8.7 Quellenangabe

Abb. 1: https://www.mikrocontroller.net/attachment/274921/DMX_Frames.jpg
Logo: http://www.hbu.ch/de

Alle Abbildungen, welche in der Quellenangabe nicht aufgeführt wurden, sind selbst erstellt.

8.8 Beilagenverzeichnis

- Glossar
- Anleitung
- Stückliste
- Testkonzept
- Zeitplan
- Diagramme
- Schema & Layout
- Frontplatte
- Diverses:
  - DA Aufgabenstellung Roman Spring
  - DA Vorschlag DMX-Steuerung Roman Spring
  - DA Bewertungsformular 2017 (Hardware/Software)
  - Bestimmungen VDA/DA
  - Terminplan DA HF 17/18
- CD mit Dokumentation, Sourcecode & Anhang

Glossar

Abkürzung	Erklärung
HFU	**H**öhere **F**achschule **U**ster
RSG	Kürzel des Verfassers, **R**oman **S**prin**g**
DA	**D**iplom**a**rbeit
NAS	**N**etwork **A**ttached **S**torage, Netzgebundener Speicher
DMX	**D**igital **M**ultiple**x**
WDMX	**W**ireless **D**igital **M**ultiple**x**
NiMh	**Ni**ckel **M**etall**h**ydrid
UV	**U**ltra **V**iolett

Tab. 18 Glossar – Abkürzungen

Begriffe	Erklärung
Slot (DMX-Signal)	11 Bits welche eine Adresse übermitteln (1 Start-, 8 Werte- & 2 Stoppbits)
Frame (DMX-Signal)	Ein Durchlauf des DMX-Signals
Showtec	Marke verschiedener Eventprodukte (www.highlite.nl/de)
Elation	Marke verschiedener Scheinwerfer (www.elationlighting.com/)
Blackout	Ein Modus, in welchem alle Scheinwerfer aus sind
Freiluftstyle	Das Programmieren ohne die Hardware
Array	Eine Sammlung von gleichwertigen Werten

Tab. 19 Glossar - Begriffe

Farbe vs Szene (aus der Sicht dieses Projekts)**:**
Unter dem Speichern einer Farbe versteht man, dass alle sieben Faderpositionen abgespeichert werden. In der Software werden die sieben Werte der Fader in das Array der Taste abgespeichert. Bei dem Aufruf einer Farbe werden alle selektierten Geräte auf diese sieben Werte gesetzt.

Das Speichern einer Szene hingegen bedeutet, dass alle Geräte mit ihren Werten gespeichert werden. Dies bedeutet in der Praxis, dass insgesamt 16-mal eine Farbe à sieben Werte abgespeichert werden. Das temporäre Abspeichern geht relativ schnell vonstatten. Das Speichern in das EEPROM benötig jedoch viel mehr Zeit, daher wird dies in kleinen Schritten erledigt (Siehe Kapitel „5.2.4.2 EEPROM“). Bei dem Aufrufen einer Szene werden alle Geräte, ob selektiert oder nicht, überschrieben und es entsteht die komplett gleiche Szene, welche zuvor abgespeichert wurde.

Anleitung

Stückliste

Testkonzept

Zeitplan

Diagramme

Schema & Layout

Frontplatte

DA Aufgabenstellung

DA Vorschlag

Bewertungsformular DA

Bestimmungen VDA/DA

Terminplan DA HF 17/18

CD

Inhalt:

1. Dokumentation
2. Anleitung
3. Stückliste
4. Testkonzept
5. Zeitplan
6. Diagramme
7. Schema & Layout
8. Frontpanel
9. Diverses
    1. Datenblätter
    2. DA Aufgabenstellung Roman Spring
    3. DA Vorschlag DMX-Steuerung Roman Spring
    4. DA Bewertungsformular 2017 (Hardware/Software)
    5. Bestimmungen VDA/DA
    6. Terminplan DA HF 17/18
10. Sourcecode