

# Diplomarbeit

**Mediatrix**  
**Ausgeschriebener Titel der Diplomarbeit**

ausgeführt an der  
Höheren Abteilung für Informationstechnologie/Medientechnik  
der Höheren Technischen Lehranstalt Wien 3 Rennweg

im Schuljahr 2017/2018

durch

**Nußbaumer Dominik**  
**Scharwitzl Clemens**  
**Steiner Florian**

unter der Anleitung von

Fink Andreas  
Stimpfl Franz

Wien, 3. April 2018



# Kurzfassung

Unser Team plant eine Webapplikation, die alle benötigten Parameter und Einstellungsmöglichkeiten der AV-Installation in einer übersichtlichen und intuitiv zu bedienenden Benutzeroberfläche vereint.

Die Steuerung aller Geräte erfolgt mittels einer gemeinsamen Webapplikation über jegliche Smartphones, Tablets oder PCs. Diese ermöglicht jedem Lehrer und Schüler, die Verwendung der AV-Installation im "LIZ"(Lern- und Informationszentrum), ohne jegliche technische Vorkenntnisse. Weiters können Voreinstellungen gespeichert werden, um sie während einer Präsentation schnell zu aktivieren. Das Ziel ist es, die Vorbereitungszeit für eine Multimedia-Präsentation zu minimieren und die Bedienung der Geräte zu erleichtern.

Zusätzlich entwickeln wir die Schnittstellen zwischen Webapplikation und den Geräten unter Verwendung eines Raspberry Pi's.



# Abstract

Our team is planning to create a web application which will enable users to control all of the necessary parameters of an AV-installation in an intuitive and easy-to-use Interface.

All devices can be controlled using one central web application running on any smartphone, tablet or desktop. This enables every teacher or student to use the AV-installation present in the “LIZ”(Lern-und Informationszentrum) without having to acquire any additional technical knowledge. Furthermore configurations can be saved as presets. These are easily activated during a presentation with the click of a button. The goal is to minimize the time wasted before a presentation is even started. This not only makes the experience more seamless and professional but also reduces the pressure put on the presenters when something does not go according to plan.

Additionally we are developing Interfaces between the web application and the other devices using a Raspberry Pi mini computer.



# Ehrenwörtliche Erklärung

Ich erkläre an Eides statt, dass ich die individuelle Themenstellung selbstständig und ohne fremde Hilfe verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Wien, am 3. April 2018

---

Mitarbeiter Eins

---

Mitarbeiter Zwei

---

Mitarbeiter Drei



# Inhaltsverzeichnis

<b>Tabellenverzeichnis</b>	<b>xiii</b>
<b>Abbildungsverzeichnis</b>	<b>xv</b>
<b>1      Einleitung</b>	<b>1</b>
1.1    Problemstellung . . . . .	1
1.2    Ziel der Arbeit . . . . .	1
1.3    Abgrenzung und Voraussetzungen . . . . .	1
1.4    Aufbau . . . . .	1
<b>2      Hardware</b>	<b>3</b>
2.0.1    Abgrenzung und Voraussetzungen: . . . . .	3
2.0.2    Aufbau: . . . . .	4
2.0.3    Ziele der Arbeit: . . . . .	4
2.0.4    Gehäuse . . . . .	4
2.0.4.1    Bestandteile des Systems . . . . .	4
2.0.4.1.1    19" Racks . . . . .	6
2.0.4.1.2    Mediatrix-Modul . . . . .	6
2.0.4.2.1    Das Anschlusspanel . . . . .	7
2.0.4.2.2    Anschlüsse für Lautsprecher . . . . .	9
2.0.4.2.3    Anschlüsse für Gehäuseelektronik . . . . .	9
2.0.4.2.4    Anschlüsse für Netzwerkschnittstellen . . . . .	9
2.0.4.2.5    Anschlüsse für Lichtsteuerung . . . . .	9
2.0.4.2.6    Anschlüsse für Infrarotsteuerung . . . . .	10
2.0.4.2.7    Anschlüsse für Zustandserkennung . . . . .	10
2.0.4.2.8    Anschlüsse für Stromversorgung . . . . .	10
2.0.4.2.9    Anfertigung des Anschlusspanels . . . . .	10
2.0.4.2.10    Das Frontpanel . . . . .	10
2.0.4.2.11    Einbau der Geräte in das Mediatrix-Modul . . . . .	10
2.0.4.2.12    Anfertigung der Hauptplatine . . . . .	11
2.0.4.2.13    Anschluss der Relais-Platine . . . . .	11
2.0.5    Raspberry Pi . . . . .	11
2.0.5.1    Funktionen und Spezifikationen . . . . .	12
2.0.5.2    Eignung für dieses Projekt . . . . .	13
2.0.5.3    mögliche Alternativen . . . . .	13
2.0.6    Anschlüsse für Anwender . . . . .	13
2.0.6.1    Anforderungen . . . . .	13

2.0.6.2	Nutzerfreundlichkeit . . . . .	14
2.0.6.3	technische Umsetzung . . . . .	14
2.0.7	Ein- und Ausschaltverzögerung der Lautsprecher . . . . .	14
2.0.8	Stromversorgung . . . . .	16
2.0.8.1	Stromzufuhr zum System . . . . .	16
2.0.8.2	Stromverteilung im Mediatrix-Modul . . . . .	17
2.0.8.3	Spannungswandlung . . . . .	17
2.0.8.4	Die Hauptplatine . . . . .	17
2.0.8.4.1	MOSFET . . . . .	18
2.0.8.5	Einschalten des Systems . . . . .	18
2.0.9	Verkabelung . . . . .	19
2.0.9.1	Vorgang . . . . .	19
2.0.9.2	Lautsprecherleitungen . . . . .	19
2.0.9.3	Anschlüsse für User . . . . .	19
2.0.10	Gehäusebelüftung . . . . .	20
2.0.10.1	Digitaler Temperatursensor . . . . .	20
2.0.10.2	Integration . . . . .	20
2.0.10.3	Aufgetretene Probleme: . . . . .	21
2.0.10.4	Lüftersteuerung mit PWM . . . . .	24
2.0.10.5	Generieren eines PWM-Signals mit dem Raspberry Pi . . . . .	26
2.0.11	Quickstart-Guide . . . . .	27
2.0.12	Hardware für Infrarotfunktionalität . . . . .	27
2.0.13	Beleuchtungskonzept für Konferenzsaal . . . . .	27
2.0.13.1	Herausforderungen . . . . .	28
2.0.13.2	Rednerpositionen . . . . .	28
2.0.13.3	Zusätzliche Beleuchtung . . . . .	29
2.0.13.4	Scheinwerfer: . . . . .	29
2.0.13.5	Mögliche Konfigurationen: . . . . .	30
2.0.13.6	Aufhängung: . . . . .	30
2.0.13.6.1	EUTRAC-Stromschiene: . . . . .	31
2.0.13.6.2	1-Punkt Traverse: . . . . .	31
2.0.13.6.3	Einbau der Traverse: . . . . .	32
2.0.13.7	Gesamtpreise der Konfigurationen: . . . . .	32
2.0.14	Status-Erkennung der Geräte . . . . .	33
<b>3</b>	<b>Betriebssystem</b>	<b>35</b>
3.1	Raspbian . . . . .	35
3.2	Sicherheit . . . . .	36
3.2.1	SSL und TLS . . . . .	36
3.2.2	Firewall . . . . .	36
3.3	Webserver . . . . .	37
3.3.1	Aufbau von Apache . . . . .	38
3.3.2	Verwendete Module und Konfiguration . . . . .	39
3.4	OLA . . . . .	40
3.4.1	C++-Schnittstelle . . . . .	40

3.5	WiringPi . . . . .	41
3.5.1	Serielle Steuerung . . . . .	41
3.5.2	Pin Steuerung . . . . .	43
<b>4</b>	<b>Backend</b>	<b>45</b>
4.1	PHP Extension . . . . .	45
4.1.1	PHP-CPP . . . . .	45
4.1.1.1	C++-Datei . . . . .	45
4.1.1.2	INI-Datei und Makefile . . . . .	47
4.2	Websocket . . . . .	48
4.2.1	Client - Server Kommunikation . . . . .	48
4.2.1.1	Technische Spezifikation - Nuss . . . . .	48
4.2.1.2	Kommunikation mit dem Mischpult - Nuss . . . . .	49
4.2.1.3	Ratchet . . . . .	50
4.2.1.3.1	Application-Klasse . . . . .	51
4.3	DMX . . . . .	52
4.3.1	Funktionsweise des Protokolls . . . . .	52
4.3.2	Anwendung in diesem Projekt . . . . .	53
4.4	Infrarot . . . . .	53
4.4.1	Funktionsweise des Protokolls (RC-5) . . . . .	54
4.4.2	Anwendung in diesem Projekt . . . . .	55
4.5	JSON Web Token . . . . .	55
4.5.1	Firebase-JWT . . . . .	56
4.6	SQLite . . . . .	57
4.6.1	Vorteile für diese Projekt . . . . .	58
4.7	Schnittstelle . . . . .	58
<b>5</b>	<b>Frontend</b>	<b>59</b>
5.1	Responsives Design . . . . .	59
5.1.0.1	Vorteile von anpassungsfähigen Webseiten . . . . .	60
5.2	CSS . . . . .	60
5.2.1	Flexbox . . . . .	60
5.2.1.1	Das Konzept . . . . .	61
5.2.1.2	Technische Spezifikation . . . . .	61
5.2.1.3	Erklärung anhand eines realen Beispiels . . . . .	61
5.2.1.4	Weitere Möglichkeiten von Flexbox . . . . .	64
5.2.2	CSS-Grid . . . . .	64
5.2.2.1	Das Konzept . . . . .	65
5.2.2.2	Technische Spezifikation . . . . .	65
5.2.2.3	Erklärung anhand eines Beispiels . . . . .	65
5.2.2.4	Weitere Möglichkeiten von CSS-Grid . . . . .	68
5.2.3	Gegenüberstellung von CSS-Grid und Flexbox . . . . .	68
5.2.3.1	Eine versus zwei Dimensionen . . . . .	68
5.2.3.2	Dynamisch versus statisch . . . . .	69
5.2.3.3	Erklärung der Unterschiede anhand eines Beispiels . . . . .	70
5.2.3.4	Kombination von CSS-Grid mit Flexbox . . . . .	71

5.2.3.5	Flexbox oder CSS-Grid? . . . . .	73
5.2.4	Normalize.css . . . . .	73
5.2.4.1	Technische Spezifikation . . . . .	74
5.3	SASS . . . . .	74
5.3.1	Was ist ein CSS-Präprozessor? . . . . .	74
5.3.1.1	Partielle CSS-Dateien . . . . .	75
5.3.1.2	Variablen in SASS . . . . .	75
5.3.1.3	Sauberer Code dank Verschachtelung der Regeln . . . . .	76
5.3.1.4	Wiederholungen vermeiden . . . . .	76
5.3.1.5	Funktionen, um Farben zu modifizieren . . . . .	77
5.3.1.6	Nachteile von CSS-Präprozessoren . . . . .	77
5.3.1.7	Verwenden oder nicht verwenden? . . . . .	77
5.4	CSS-Variablen . . . . .	78
5.4.1	Was sind CSS-Variablen? . . . . .	78
5.4.1.1	Technische Spezifikation . . . . .	78
5.4.1.2	Erklärung anhand eines Beispiels . . . . .	79
5.5	jQuery . . . . .	80
5.5.1	Was ist jQuery? . . . . .	80
5.5.1.1	Mehr schreiben in weniger Zeit . . . . .	80
5.5.1.2	Browser Kompatibilität . . . . .	80
5.5.1.3	Simplifizierung von Javascript-Funktionen . . . . .	80
5.5.1.4	Gewappnet für die Zukunft . . . . .	81
5.5.1.5	Potential für Zeitersparnis . . . . .	81
5.5.1.6	Nachteile . . . . .	81
5.6	Design . . . . .	82
5.6.1	Fokus auf technisch unversierte Benutzer . . . . .	82
<b>6</b>	<b>Zusätzliches</b>	<b>83</b>
6.1	Bedienungsanleitung . . . . .	83
6.2	Beleuchtungskonzept Konferenzsaal . . . . .	83
<b>A</b>	<b>Anhang 1</b>	<b>85</b>
<b>Literaturverzeichnis</b>		<b>87</b>

# Tabellenverzeichnis



# Abbildungsverzeichnis

2.1	Innenansicht des Mediatrix-Moduls . . . . .	7
2.2	Anschlussblende des Mediatrix-Moduls . . . . .	8
2.3	Schaltplan DS1820 Raspberry Pi . . . . .	20
2.4	Putty-Ausgabe inkorrekt . . . . .	21
2.5	Putty-Ausgabe korrekt . . . . .	21
2.6	Putty-Ausgabe Temperatur wird angezeigt . . . . .	22
2.7	Pulsweite . . . . .	25
2.8	Duty-Cycle . . . . .	26
2.9	Infrarotplatine . . . . .	28
2.10	EUTRAC Montage . . . . .	31
2.11	1-Punkttravere Montage . . . . .	32
3.1	Die konfigurierten UFW-Regeln . . . . .	37
4.1	Darstellung des DMX-Signals . . . . .	53
4.2	Darstellung des elektromagnetischem Spektrums . . . . .	54
4.3	Darstellung eines RC-5-Codes . . . . .	55
5.1	Flexbox Beispiel Funktionalität . . . . .	62
5.2	Ein dreispaltiges CSS-Grid-Layout mit einer Zeile . . . . .	66
5.3	Ein einspaltiges CSS-Grid-Layout mit drei Zeilen . . . . .	67
5.4	grundlegender Unterschied zwischen CSS-Grid und Flexbox . . . . .	69
5.5	derzeitige Darstellung ohne Flexbox . . . . .	70
5.6	Layout mit "display: flex" . . . . .	70
5.7	Layout-Kombination von Flexbox und CSS-Grid . . . . .	72



# 1 Einleitung

## 1.1 Problemstellung

Heutzutage werden immer höhere technische Anforderungen an Multimediainstallation gesetzt. Es muss einen Beamer, Scheinwerfer, Mikrophone und diverse andere Geräte vorhanden sein. Dabei wird oft vergessen, dass all diese Geräte von einem einfachen Anwender bedient werden müssen, der möglicherweise keine Ahnung von dem integrierten AV-System hat. Meist wird die Person, die diese Installation nutzen will, mit einer Hand von Fernsteuerungen allein gelassen. Wirklich Ahnung von dem System hat meist nur eine Handvoll Personen, die oft keine Zeit haben Hilfe bei der Verwendung zu leisten. Diese Kombination führt oft dazu, dass die Geräte vollständig verstellt oder beschädigt werden. Oft wird auch aus Unwissenheit vergessen die Geräte nach der Verwendung auszuschalten, was zu Überhitzungen und sogar zu Brandgefahr führen kann. All das resultiert in einem Kostenaufwand, sei es durch den Ankauf neuer Geräte, da die Alten beschädigt wurden, oder durch Zeit, die ein Techniker aufwenden muss, um das System wieder in Gang zu bringen.

Hier ist es dringen notwendig den Personen, die eine AV-Installation verwenden möchten, Abhilfe zu schaffen. Es braucht ein System, das die Steuerung von AV-Geräten vereinfacht und gegen Bedienfehler absichert.

## 1.2 Ziel der Arbeit

## 1.3 Abgrenzung und Voraussetzungen

## 1.4 Aufbau



## 2 Hardware

### **2.0.1 Abgrenzung und Voraussetzungen:**

Da das Projekt auf die bereits vorhandenen Geräte aufgebaut wurde, sind diese zusätzlich notwendig. Folgende Komponenten müssen vorhanden sein um das Mediatrix-System einzurichten:

- AV-Receiver
- Digitalmischpult
- Beamer
- Lautsprecher
- Verstärker
- Funkmikrofone
- Server-Rack

Weiters werden diverse Werkzeuge und ein Budget von 300€ benötigt.

In dem Projekt wurden Schnittstellen zwischen der vorhandenen Infrastruktur entwickelt. Diese wurden sowohl hardware- als auch softwareseitig umgesetzt. Die Webapplikation beinhaltet nur die notwendigsten Funktionen der Geräte. Um alle Funktionen des jeweiligen Geräts steuern zu können, muss nach wie vor die dazugehörige Fernbedienung verwendet werden.

Das System kann nur schulintern verwendet werden, da der Webserver ist nur innerhalb des Schulnetzwerks erreichbar ist. Zur Vermeidung von Konflikten kann immer nur ein Gerät mit dem System verbunden sein.

## 2.0.2 Aufbau:

Die Hauptkomponenten des Systems sind zum einen der Webserver mit der darauf laufenden Webapplikation und zum anderen die Elektronik. Als Schnittstelle zwischen Software und Hardware ist ein Raspberry Pi zuständig.

## 2.0.3 Ziele der Arbeit:

Mediatrix soll die Bedienung der AV-Installation im Multimediasaal unserer Schule vereinfachen und jedem ermöglichen. Der User benötigt ausschließlich ein WLAN-fähiges Gerät und muss im Schulnetz eingeloggt sein. Nach Anmeldung in der Webapplikation hat der User die Möglichkeit, grundlegende Funktionen der Installation, die für eine Präsentation wichtig sind einzustellen. Die Bedienoberfläche ist so angelegt, dass sie selbst während einer Präsentation unauffällig bedient werden kann.

Auch außerhalb der Webapplikation soll sich der User gut zurechtfinden. Bedienfehlern soll durch verschiedene Schutzmechanismen vorgebeugt werden.

Änderungen der Infrastruktur sollen leicht zu bewerkstelligen sein. Aus diesem Grund wurde im Zuge der Konzeption darauf geachtet, eine möglichst hohe Kompatibilität des Systems mit verschiedenen Geräten zu erzielen. Beispielsweise wurden Branchenstandards wie Infrarot und DMX zur Steuerung von AV-Geräten, beziehungsweise lichttechnischer Geräte implementiert.

## 2.0.4 Gehäuse

### 2.0.4.1 Bestandteile des Systems

Das „Mediatrix-System“ besteht aus folgenden Komponenten:

- Soundcraft UI16 Digitales Tonsmischpult

dient dem abmischen von verschiedensten Audioquellen, wie Mikrofonen oder Line-Signalen.

- AKG SR420 Funkmikrofone

erleichtern die Verständlichkeit des Redners bei Präsentationen vor großem Publikum.

- Denon AV-Receiver

verteilt, die über HDMI ankommende Signale auf Lautsprecher und Beamer.

- Behringer PA-Endstufe

versorgt die vorderen Lautsprecher mit Leistung.

- Botex 4-Kanal Lichtdimmer

regelt den Strom der Scheinwerfer, um diese in ihrer Helligkeit steuern zu können.

- Kleinrechner

dient zum Abspielen von Präsentationen und Multimediainhalten jeglicher Art.

Weiter Komponenten:

- Raspberry Pi 2 Mikrocontroller

ist der Mittelpunkt des Systems. Er fungiert als Webserver der Webapplikation und ist für die Steuerung der Elektronik zuständig.

- 12V – Relais-Platinen

dienen zur Unterbrechung der Lautsprecherleitungen und der Stromversorgung der Geräte.

- 12V - Netzteil

wandelt 230V AC in 12V DC um.

- Stepdown-Converter

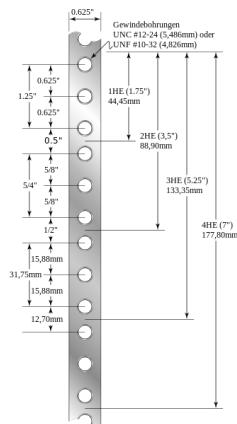
konvertiert 12V DC auf 5V DC, um den Raspberry Pi zu betreiben.

- ENTTEC DMX Interface

bietet die Schnittstelle zwischen Raspberry Pi und den Scheinwerfern.

Um das System möglichst kompakt und modular zu gestalten, bietet sich ein Rack-Schrank mit Montageschienen nach 19"- Standard an. Die Abstände zwischen den Bohrungen an Gerät und Schiene sind genormt, ebenso wie der Abstand zwischen den Schienen. Das ermöglicht den einfachen Einbau oder Austausch von 19"- konformen Geräten.

**2.0.4.1.1 19" Racks** Der horizontale Abstand zwischen zwei Montageleisten beträgt genau 19 Zoll (= 48,26 Zentimeter).



Die minimale Höhe eines 19"- konformen Geräts beträgt 1,75" oder 44,45mm. Dieser Abstand wird als eine Höheneinheit (1HE) bezeichnet. Darum werden bei Geräten aus den Bereichen Tontechnik oder Großrechner und Server die Maße der Geräte oftmals in 19" und Anzahl der Höheneinheiten angegeben. Die Tiefe der Geräte kann jedoch variieren und wird darum meist in Millimetern angegeben.

In unserem Fall haben das Tonschischpult, die Funkmikrofone und die Endstufe konforme Maße, um in ein 19"- Rack eingebaut zu werden. Der AV-Receiver ist mittels Rack-Wanne im Rack platziert. [? , S.101]

## 2.0.4.2 Mediatrix-Modul

Die elektronischen Komponenten, wie der Raspberry Pi, die Relais und das 12V-Netzteil, galt es ebenfalls sicher und kompakt in dem Rack zu platzieren. Dazu fiel die Entscheidung auf ein 19"- Gehäuse mit drei Höheneinheiten. Es wurde ein Leergehäuse angeschafft und mit allen notwendigen Buchsen bestückt. An der Rückseite des Gehäuses sind alle benötigten Anschlüsse herausgeführt.

Innerhalb des Gehäuses sind alle Komponenten befestigt, um sie gegen Verrutschen und sich daraus ergebende Beschädigungen, zu schützen. Die Platinen sind mit Abstandhaltern vom Gehäuseboden abgehoben, um Kurzschlüsse zu vermeiden.

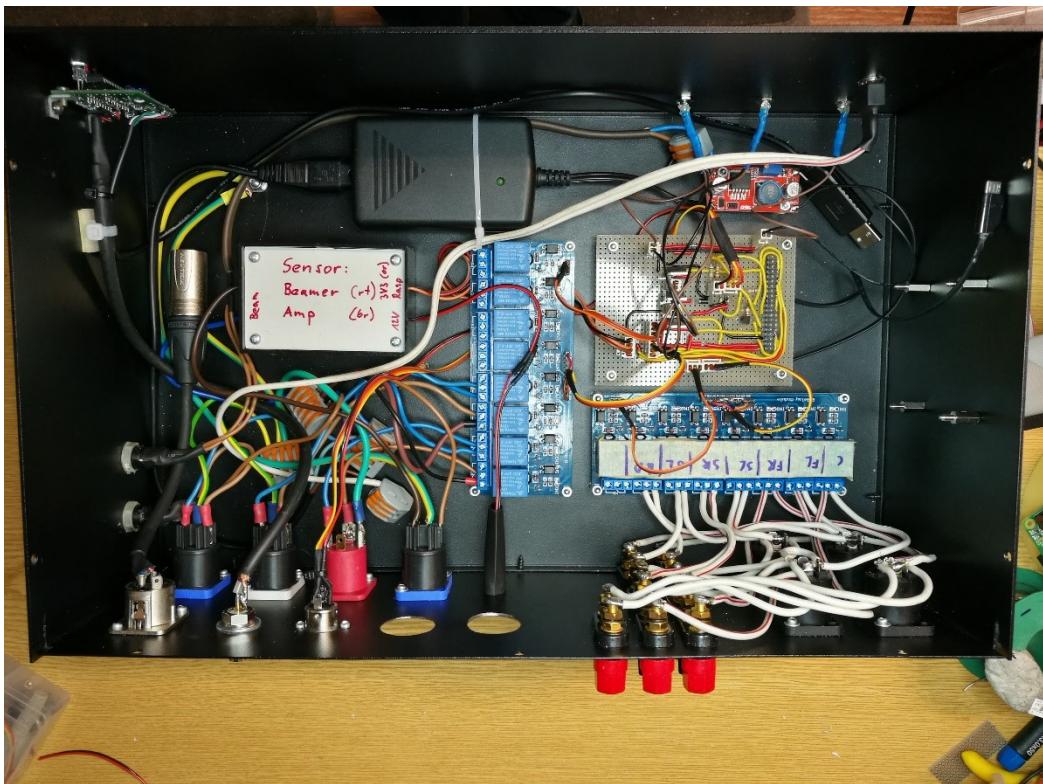


Abbildung 2.1: Innenansicht des Mediatrix-Moduls

**2.0.4.2.1 Das Anschlusspanel** Da alle Lautsprecherleitungen mittels Relais unterbrochen werden können sollen, müssen alle Ausgänge des AV-Receiver mit dem Mediatrix-Modul verbunden sein. Um das vom AV-Receiver bereitgestellte 7.2. – Signal auch ausgeben zu können, sind insgesamt sieben, bereits verstärkte Lautsprecherausgänge vorhanden.

Diese werden mit Patchkabeln an das Mediatrix-Modul angeschlossen. In diesem befindet sich die geregelte Unterbrechungsschaltung. Die beiden unverstärkten Subbass-Ausgänge sind mit einer Stereo-DI-Box verbunden. Die DI-Box symmetriert das asymmetrische Signal und gibt dieses mittels XLR-Leitungen an die Subwoofer weiter. Durch die Symmetrierung des Signals werden Einstreuungen abgeschwächt.

### Symmetrierung von Audiosignalen

„Bei der symmetrischen Kopplung führen beide Adern einer Verbindungsleitung eine amplitudengleiche, jedoch aber gegenphasige Signalspannung gegenüber der Erde (bzw. Masse). Beide Adern sind gegenüber der Erde galvanisch getrennt. Durch die Gegenphasigkeit wird bei der symmetrischen Kopplung eine hohe Störsicherheit erreicht. Bei idealen Voraussetzungen, d.h. bei vollkommener Symmetrie und gleicher Einwirkung eines Störfeldes auf die beiden Adern wird die Störspannung durch Kompensation



Abbildung 2.2: Anschlussblende des Mediatrix-Moduls

aufgehoben."

\cite[S.100]{wirsum\_praktische\_1991} Die Kompensation, oder Auslöschung ergibt sich aus der  $180^\circ$  Phasenverschiebung zwischen den beiden Tonadern. [? ]

Für diese Art der Übertragung ist also eine Leitung mit zwei Signalleitern notwendig. Masse und Schirm sind voneinander getrennt. Nicht so bei der asymmetrischen Übertragung: „Bei der asymmetrischen, meist übertragerlosen Kopplung ist eine Ader geerdet, so daß Fremdfelder Störspannungen erzeugen und bei nicht einwandfreien Erdungsverhältnissen Brummschleifen entstehen können.“ [? , S. 100] Bei kurzen Leitungslängen treten solche Störspannungen selten auf. Allerdings sollte bei langen Strecken eine symmetrische Verbindung verwendet werden.

Die Leitungen zu den Subwoofern sind zwar in diesem Fall nicht lang, jedoch parallel zu Stromleitungen, die Einstreuungen verursachen können verlegt. Aus diesem Grund ist die Entscheidung auf symmetrische Signalübertragung gefallen.

Ein weiterer Vorteil von symmetrischen XLR-Leitungen, gegenüber asymmetrischer Cinch-Leitungen ist die Möglichkeit, Kabel ohne jeglichen Adapter zusammenzustecken und somit die Leitung zu verlängern.

**2.0.4.2.2 Anschlüsse für Lautsprecher** In diesem Fall sind als Ein- und Ausgangsschnittstelle der Lautsprecherleitungen Bananenbuchsen im Mediatrix-Modul verbaut. Diese werden ebenfalls bei gängigen AV-Receivern verwendet. Zum Verbinden können daher handelsübliche Patch-Kabel mit Bananensteckern an beiden Enden verwendet werden. Das bietet mehr Kompatibilität mit verschiedenen AV-Receivern.

Weiters haben die verbauten Buchsen einen Schraubanschluss, mit dem alternativ zu den Bananensteckern auch ein blankes Kabel an den Kontakt geklemmt werden kann.

Die Ausgänge zu den Lautsprechern sind als vierpolige Speakon-Buchsen ausgeführt. So lassen sich jeweils 2 Lautsprecher mit einer Buchse verbinden. Die Lautsprecherpaare sind in die Ausgänge „Center, Front, Side und Back“ unterteilt.

Ein separater Power-Ausgang ermöglicht die Anwendung der Lautsprecherschutzschaltung auf die Aktiv-Subwoofer. Das Unterbrechen des unverstärkten Signals würde zu Störgeräuschen führen. Aus diesem Grund wird die Stromversorgung der Subwoofer geschalten. Die Verbindung ist zudem eindeutig von üblichen Schutzkontaktsteckern zu unterscheiden, was das korrekte Anschließen erheblich erleichtert und Fehlern vorbeugt.

**2.0.4.2.3 Anschlüsse für Gehäuseelektronik** Zur Verbindung von Lüfter, Power-Taster und Öffnungskontakten mit dem Raspberry Pi ist eine DIN-Buchse verbaut. Die Anschlusskabel dieser Komponenten laufen in einen DIN-Stecker zusammen und können somit alle gleichzeitig an- und abgesteckt werden. Vertauschen von Anschläßen ist folglich unmöglich.

**2.0.4.2.4 Anschlüsse für Netzwerkschnittstellen** Der Raspberry Pi stellt den Webserver des Systems zur Verfügung und muss darum über eine Netzwerkverbindung verfügen. Zudem führt eine Netzwerkleitung direkt zum Soundcraft UI16, um die systeminterne Kommunikation gegen böswillige Einwirkungen durch Hacker zu schützen. Diese beiden Leitungen sind als RJ-45 Buchsen aus dem Mediatrix-Modul herausgeführt.

**2.0.4.2.5 Anschlüsse für Lichtsteuerung** Bei der Lichtsteuerung ist auf den Branchenstandard, das DMX-Protokoll gesetzt worden. Dieses wird mittels drei- oder fünfpoliger XLR-Steckern übertragen. Im semiprofessionellen Sektor sind dreipolige Anschlüsse üblicher, darum ist ein solcher eingebaut. Intern ist diese Buchse mit dem ENTTEC DMX-Interface verbunden.

**2.0.4.2.6 Anschlüsse für Infrarotsteuerung** Um die Richtung des Infrarotsignals gezielt auf die zu steuernden Geräte ausrichten zu können, kann eine IR-Diode mittels BNC-Connector angeschlossen und außerhalb des Racks platziert werden. Eine weitere IR-Diode ist in das Frontpanel des Mediatrix-Moduls eingebaut, um Rack-internen Infrarotgeräte, wie den AV-Receiver ansteuern zu können. Oberhalb von ihr befinden sich zudem noch eine Status LED, die blinkt, wenn Signale gesendet werden und eine Diode zum einlesen von neuen IR-Befehlen.

**2.0.4.2.7 Anschlüsse für Zustandserkennung** Der Einschaltzustand des AV- Receivers wird über den Trigger-Output überprüft. Für den Anschluss an das Mediatrix- Modul ist eine 3,5mm Klinkenbuchse verwendet worden. Da der Beamer über keinen Trigger-Output verfügt, wird die Induktionsspannung an einem PowerCon-Ausgang überwacht.

**2.0.4.2.8 Anschlüsse für Stromversorgung** Das Mediatrix-Modul wird mittels PowerCon-Kabel mit 230V versorgt. Diese werden im Gehäuse intern verteilt und an z.B. den steuerbaren PowerCon-Ausgang an die Geräte im Rack weitergegeben.

**2.0.4.2.9 Anfertigung des Anschlusspanels** Um zu garantieren, dass es keine Verwechslungen beim Anschließen der Geräte an das Mediatrix-Modul gibt sind verschiedene Typen von Buchsen verwendet worden. Zur Bohrung der Löcher in das Stahlblech des Gehäuses wurde ein Stufenbohrer verwendet. Anschließend wurden alle Buchsen eingebaut. Die BNC und die Klinkenbuchse verfügen über Muttern und Gewinde, mit denen sie befestigt wurden. Die restlichen Anschlüsse wurden mit Blindnieten vernietet.

**2.0.4.2.10 Das Frontpanel** Das Frontpanel ist sehr schlicht gehalten. Es beinhaltet einen Schalter, der die Stromversorgung des Mediatrix-Moduls unterbricht und Infrarot Sender, Empfänger und Status LED. Weiters sind noch drei Status LEDs für „Stromversorgung AN“, „Raspberry Pi hochgefahren“ und „Lautsprecherleitungen freigegeben“ verbaut. Das Frontpanel kann für Wartungsarbeiten, genauso wie der Deckel leicht abgenommen werden. Die Kabel zum Schalter sind durch Öffnen zweier Federklemmen vom Mediatrix-Modul trennbar. Die Status LEDs können durch Lösen der Steckverbindung von der Platine separiert werden.

**2.0.4.2.11 Einbau der Geräte in das Mediatrix-Modul** Der Großteil der Elektronik befindet sich im Mediatrix-Modul. Aus Platzgründen sind der Raspberry Pi und die Infrarotplatine an den Seitenteilen des Gehäuses montiert.

Am Boden befinden sich zwei Relais-Platinen mit jeweils acht Relais. Eine Platine für die Lautsprecherleitungen, die Andere für die Stromverteilung. Weiters sind auf einer

Lochrasterplatine sind alle notwendigen Schaltungen und Verbindungen zum Raspberry Pi aufgebaut. So wie auch das Netzteil und der Stepdown-Converter sind alle bereits genannten Komponenten fest mit der Bodenplatte verschraubt. Das DMX-Interface ist mit doppelseitigem Klebeband befestigt, da es nahezu kein Gewicht hat und so platziert ist, dass nichts beschädigt werden kann.

**2.0.4.2.12 Anfertigung der Hauptplatine** Für die Hauptplatine wurde eine Lochrasterplatine verwendet. Die Schaltungen wurden als Prototypen bereits auf Laborsteckbrettern aufgebaut und getestet. Schließlich wurden alle auf die Lochrasterplatine übertragen und verlötet.

**Steckverbindungen** Um die Platine oder andere angeschlossene Komponenten möglichst einfach ausbauen oder austauschen zu können, sind die Verbindungen als verpolungssichere Stecker ausgeführt. Alle Pins des Raspberry Pis sind mit einem IDE-Kabel an eine Stifteleiste auf der Platine angeschlossen. Das System kann somit je nach Bedarf um zusätzliche Funktionen erweitert werden.

**Vorteile** Im Gegensatz zu einer geätzten Platine kann eine Lochrasterplatine nachträglich um Bauteile erweitert werden. Weiters ist das Anfertigung nicht so zeitaufwendig wie beim Ätzen.

**2.0.4.2.13 Anschluss der Relais-Platine** Da die Relais eine Schaltspannung von 12V DC benötigen, erfolgt die Versorgung über ein 12V-Netzteil. Diese sind mit JD-VCC und GND verbunden. Der VCC Pin an dieser Steckleiste dient der Stromversorgung der Optokoppler.

Die zweite Steckleiste verfügt über einen GND, ein VCC 5V und 8 Pins für die jeweiligen Relais. Bezogen wird die 5V-Spannung vom Raspberry Pi. Jeder Relais-Pin kann mit einem GPIO-Pin des Raspberry Pis verbunden und somit einzeln angesteuert werden. In diesem Anwendungsfall werden für die Lautsprecherabschaltung 7 Relais mit einem GPIO-Pin geschalten, da eine Differenzierung der Leitungen nicht notwendig ist. Gleichzeitig wird der Strom zu den Subwoofern über zwei Relais auf der anderen Platine geschalten. Ein weiterer GPIO-Pin ist für die Steuerung der Versorgungsspannung der anderen Geräte im Rack zuständig.

## 2.0.5 Raspberry Pi

Der Raspberry Pi ist ein Mikroprozessor, der von „The Raspberry Pi Foundation“ entwickelt wurde, um Menschen auf der ganzen Welt einen Zugang zur digitalen Welt

zu ermöglichen. Der Mikroprozessor ist für nahezu jeden leistbar, verfügt aber über alle Funktionen eines vollwertigen PCs. Zusätzlich sind Dokumentationen und Anleitungen zum Raspberry Pi gratis online verfügbar.

[? ]

Mittlerweile finden sich im Internet verschiedenste Blogs und Videos zu Projekten mit Raspberry Pis, die von jedem nachgebaut werden können. Aufgrund seines Preises und seiner geringen Größe wird der Mikroprozessor gerne für Internet-Of-Things verwendet.

#### **2.0.5.1 Funktionen und Spezifikationen**

Je nach Modell verfügt der Raspberry Pi über verschiedene Spezifikationen. Grundlegend sind jedoch alle ähnlich aufgebaut.

Raspberry Pi 2 Model B:

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core

[? ]

### **2.0.5.2 Eignung für dieses Projekt**

Durch den Preis, die Größe und Möglichkeit, elektronische Schaltungen und Komponenten anschließen zu können, ist der Raspberry Pi 2 Model B sehr gut auf die Anforderungen des Projekts zugeschnitten. Mittels USB-Schnittstellen können die Infrarotplatine und das USB-DMX-Interface angeschlossen werden. Die restliche Elektronik lassen sich mit den „General-Input-Output-Pins“ (GPIO-Pins) verbinden. Da eine vollwertige CPU verbaut ist, können auch komplexere Programme ausgeführt werden.

### **2.0.5.3 mögliche Alternativen**

Am Markt gibt es zahlreiche Mikroprozessoren mit verschiedensten Vor- und Nachteilen. Der „Banana Pi M2“ oder der „Odroid C1+“ verfügen beispielsweise über eine Gigabit-Ethernet-Schnittstelle. Das „Cubieboard“ ist dem Raspberry Pi leistungsmäßig deutlich überlegen, kostet allerdings dreimal so viel. Für dieses Projekt passt das Preis/Leistungsverhältnis des Raspberry Pis am besten.

[? ]

## **2.0.6 Anschlüsse für Anwender**

### **2.0.6.1 Anforderungen**

Der Hauptanwendungszweck des Systems sind Präsentationen von Schülerinnen und Schülern und Lehrerinnen und Lehrern. Deren Anforderung ist es, mit geringstem Aufwand, alle technischen Geräte, die für die Präsentation benötigt werden, verwenden zu können.

Im Regelfall wird ein Anschluss für Bildübertragung von Laptops auf den Beamer, ein Audioanschluss und in manchen Fällen auch ein Mikrofon benötigt.

Für die bestmögliche Kompatibilität mit gängigen Laptops und anderen Abspielgeräten, sind eine HDMI-Buchse, zwei Cinchbuchsen und eine XLR-Female-Buchse verbaut. Zusätzlich sind eine Schutzkontaktsteckdose, für mitgebrachte Geräte und eine USB – Buchse für den, im Rack verbauten Kleinrechner vorhanden.

Die Variante, Buchsen statt Kabeln zu verbauen, hat sich als praktikabler erwiesen als die temporäre Lösung mit Kabeln, die sich nicht bewährt hat.

Die Kabel waren im Rack an die Geräte angeschlossen und durch eine Öffnung an der Oberseite des Racks herausgeführt. Unglücklicherweise haben Schüler mehrmals, kräftig an ihnen angezogen. Beschädigungen an Kabeln, sowie an Geräten waren die Folge.

Um solche Fälle in Zukunft zu vermeiden, sind die Buchsen für Anwender gut zugänglich montiert. Die benötigten Kabel befinden sich im Präsentationskoffer, der von den Anwendern beim Portier ausgeborgt werden kann.

### **2.0.6.2 Nutzerfreundlichkeit**

Zur Vermeidung von lauten Geräuschen, die beim Anstecken auf die Lautsprecher übertragen werden, wird mittels Reedkontakt festgestellt, ob die Plexiglasabdeckung der Buchsen oder die Rack-Tür geöffnet sind. Wenn dies der Fall ist, greift die Lautsprecherabschaltung und der Signalfluss zu den Lautsprechern wird unterbrochen bis die Abdeckung und die Tür wieder geschlossen sind. Dies ist nur bei der XLR- und der Klinkenbuchse notwendig, da bei HDMI-Schnittstellen keine Störgeräusche auftreten, ebenso bei der USB-Schnittstelle. Auch der Einschalttaster befindet sich unter der Abdeckung, um unbeabsichtigtes Ausschalten des Systems zu verhindern.

Die Buchsen sind an der Seitenwand des Racks eingebaut, da auf der Oberseite Platz für Laptops oder einen Bildschirm vorhanden sein soll. Damit der Seitenteil für Wartungsarbeiten vollständig ausgebaut werden kann, sind alle Elemente durch Steckverbindungen vom Rest des Racks trennbar.

### **2.0.6.3 technische Umsetzung**

Der Reedkontakt in der Abdeckung ist mit dem zweiten Kontakt in der Rack-Tür seriell verbunden. Via GPIO-Pin werden beide vom Raspberry Pi ausgelesen. Wenn einer der beiden Kontakte offen ist, erhalten die Relais ein Signal und die Lautsprecherleitungen werden unterbrochen.

Der Signalaustausch zwischen Mediatrix-Modul und Anschlussfeld erfolgt über eine fünfpolige Leitung mit DIN-Steckverbindern an beiden Enden. An der Innenseite des Anschlussfeldes befindet sich ein DIN-Female-Stecker. Das, durch diesen Stecker ankommende Signal, wird auf einer Lochrasterplatine in einem Kunststoffgehäuse auf die verschiedenen Komponenten verteilt.

## **2.0.7 Ein- und Ausschaltverzögerung der Lautsprecher**

Audio ist ein wesentlicher Teil von Multimedia. Darum wird es auch sehr gerne in Präsentationen eingebunden. Produktvideos, Soundeffekte oder Musik tragen oftmals zum Erfolg einer Präsentation bei. Das darf durch Störgeräusche beim Einschalten von Audiogeräten in keinem Fall zunichtegemacht werden. Aus diesem Grund ist ein Lautsprecherschutz implementiert. Dieser unterbindet solche Geräusche, durch

## Unterbrechung der Lautsprecherleitungen.

Beim Einschalten von Verstärkern und Mischpulten kommt es zu knackenden Geräuschen, die auf die Lautsprecher übertragen werden. Diese können dadurch beschädigt werden. Zudem ist es für Zuhörende sehr unangenehm. Gängige Verstärker unterbinden dies, indem sie erst nachdem sie vollständig hochgefahren sind, die Lautsprecherleitungen freigeben. Mischpulte verfügen in der Regel über keine ähnliche Funktion. Hier liegt es in der Verantwortung des Anwenders, die Ausgangslautstärke des Pults auf das Minimum einzustellen, bevor das Pult eingeschalten wird. Im Fall von Schülerinnen und Schülern kann nicht davon ausgegangen werden, dass dies beachtet wird. Darum muss das Unterbrechen zu gewissen Ereignissen folgendermaßen passieren:

Beim Druck des Einschalttasters wird dem Raspberry Pi ein Signal übermittelt. In Folge schaltet dieser das Relais ein, das die anderen Geräte im Rack mit Strom versorgt. Nach 20 Sekunden werden die Lautsprecherleitungen freigegeben. Zuletzt wird die Stromzufuhr für die Subwoofer geöffnet.

1. Einschalter wird gedrückt
  - a. Raspberry Pi bekommt Signal
  - b. Raspberry Pi schaltet Strom für Geräte ein
  - c. Raspberry Pi wartet 20sek.
  - d. Raspberry Pi schaltet Lautsprecher ein
  - e. Raspberry Pi schaltet Sub Strom ein
2. Buchsenabdeckung wird geöffnet
  - a. Raspberry Pi schaltet Lautsprecher ab
  - b. Raspberry Pi schaltet Sub Strom ab
3. Buchsenabdeckung wird geschlossen
  - a. Raspberry Pi schaltet Lautsprecher ein
  - b. Raspberry Pi schaltet Sub Strom ein
4. Rack-Tür wird geöffnet

- a. Raspberry Pi schaltet Lautsprecher ab
  - b. Raspberry Pi schaltet Sub Strom ab
5. Rack-Tür wird geschlossen
- a. Raspberry Pi schaltet Lautsprecher ein
  - b. Raspberry Pi schaltet Sub Strom ein
6. Ausschalter wird gedrückt
- a. Raspberry Pi bekommt Signal
  - b. Raspberry Pi schaltet Lautsprecher ab
  - c. Raspberry Pi schaltet Sub Strom ab
  - d. Raspberry Pi wartet eine Sekunde
  - e. Raspberry Pi schaltet Strom für Geräte ab

## 2.0.8 Stromversorgung

### 2.0.8.1 Stromzufuhr zum System

Die, im System verbauten Komponenten benötigen entweder 230V, 12V oder 5V. Die Stromzuleitung zum Rack erfolgt mit 230V mittels PowerCon – Kabel und versorgt das Mediatrix-Modul dauerhaft mit Strom. Das Kabel ist durch die Kabelöffnung in der Rückwand zum Mediatrix-Modul geführt. Vom Mediatrix-Modul ist eine, durch den Raspberry Pi gesteuerte PowerCon-Buchse zu einer zu einer Steckerleiste verbunden. An dieser Leiste sind die Rack-internen Geräte und die Steckdose an der Seitenwand des Racks angeschlossen.

Die Variante, eine PowerCon-Verbindung zu verwenden bietet sich hervorragend dafür an, da sich der Stecker nicht unbeabsichtigt lösen kann. Um diesen abzustecken muss ein Sicherungsschieber am Stecker nach hinten gezogen und der Stecker nach links gedreht werden.

### 2.0.8.2 Stromverteilung im Mediatrix-Modul

Die Phase der ankommenden 230V wird direkt durch eine Sicherung geschickt, bevor sie in eine Federklemme geleitet wird. Der Nullleiter und die Erdung werden jeweils direkt in eine Federklemme geleitet. Von den Federklemmen der Phase und des Nullleiters gehen jeweils zwei Leitungen in ein Relais. An der anderen Seite der Relais sind die PowerCon-Ausgänge für die Subwoofer und die Steckerleiste der Geräte angeschlossen. Die Erdung wird nicht geschalten und ist von der Federklemme zu den Buchsen verbunden. Das Netzteil ist an die Nullleiter-Klemme und an die Phasen-Klemme angeschlossen, da es über keinen Kaltgerätestecker verfügt. Das Gehäuse ist mit einem Kabel mit Öse, die am Boden verschraubt ist, geerdet.

Mit dem Schalter an der Front des Mediatrix-Moduls sollte der Raspberry Pi, im Falle eines Fehlers, durch Abschaltung des Stroms neugestartet werden können. Aus diesem Grund geplant, die Phase direkt nach der Sicherung über den Schalter zu leiten. Bei einem Test stellte sich jedoch heraus, dass das Netzteil noch für eine halbe Minute nach Betätigung des Schalters Spannung liefert. Diese Wartezeit wäre zu lang gewesen. Bei einem erneuten Versuch mit angehängter Last verkürzte sich diese Zeit jedoch auf unter fünf Sekunden. Falls sich die Wartezeit trotz Last nicht verkürzt hätte, wäre der Schalter zwischen Netzteil und Stepdown-Converter gehängt worden.

### 2.0.8.3 Spannungswandlung

Da der Raspberry Pi allerdings eine Versorgungsspannung von 5V benötigt, müssen die 12V DC nochmals gewandelt werden. Dies geschieht mit Hilfe eines Stepdown-Converters. Parallel dazu ist ein Kabel, das zur Hauptplatine führt an den Gleichrichter angeschlossen.

Aufgrund des benötigten Stromes von mindestens 3A war eine Versorgung mittels Transformator geplant. Der Transformator sollte 12V Wechselspannung liefern, die mittels Brückengleichrichter auf 12V Gleichspannung umgewandelt werden sollten. Im Zuge von Testmessungen stellte sich jedoch heraus, dass die Spannung nach dem Gleichrichter mit 17V zu hoch war. In einem Versuch wurde eine Last von 0,6A angeschlossen, woraufhin die Spannung auf 14V zurück ging. Mit diesem Verhalten konnte das System nicht ordnungsgemäß betrieben werden. Aus diesem Grund wurde schlussendlich ein getaktetes Netzteil mit 4,1A bei 12V eingebaut.

### 2.0.8.4 Die Hauptplatine

Auf der Hauptplatine werden die 12V DC auf verschiedene Komponenten verteilt, beispielsweise die Relais-Platinen, Status LEDs und den Lüfter. Da der Raspberry Pi

nur in der Lage ist, Spannungen von 3,3V zu schalten, werden für die Status LEDs und der Lüfter MOSFETs verwendet.

**2.0.8.4.1 MOSFET** Bei einem MOSFET handelt es sich um einen Metall-Oxid-Halbleiter-Feldeffekttransistor.

„Ein Transistor ist ein Halbleiter-Bauelement aus drei aufeinanderfolgenden Halbleiter-schichten, also npn- oder pnp-Schichten. Er hat somit zwei pn-Übergänge, in deren Grenzgebieten sich Sperrsichten ausbilden.“ [? , S. 372]

Der MOSFET verfügt über drei Pins:

- Gate
- Drain
- Source

Angeschlossen ist der MOSFET wie folgt:

Gate ist mit dem IO-Pin des Raspberry Pis verbunden. Zwischen Drain und 12V ist die Last angeschlossen. Weiters ist eine Freilaufdiode in Flussrichtung von GND zu Drain eingebaut, um den MOSFET vor Überspannungen zu schützen. Source ist auf GND geführt.

## 2.0.8.5 Einschalten des Systems

Per Druck auf den, am Rack angebrachten Taster, schaltet der Raspberry Pi ein 12V Relais, das den anderen Geräten die Stromzufuhr öffnet. Jenes 12V Netzteil betreibt die Relais und muss daher, wie der Raspberry Pi, zu jeder Zeit mit Strom versorgt werden. Wenn alle Geräte den Startvorgang abgeschlossen haben, gibt der Mikroprozessor auch die Lautsprecherleitungen frei (siehe Ein- und Ausschaltverzögerung).

Die Verwendung von Relais für diese Anwendung hat sich bewährt, da der Raspberry Pi nicht in der Lage ist Spannungen von 230V zu schalten, aber mit 3,3V Steuerbefehle an ein Relais senden kann.

## 2.0.9 Verkabelung

### 2.0.9.1 Vorgang

Die Verkabelung innerhalb des Racks ist auf die eingebauten Geräte angepasst und optimiert. Alle Kabel sind etwas länger bemessen, als benötigt, um nach dem Austausch eines Geräts oder einer Neuanordnung der Geräte im Rack weiterhin verwendbar zu sein.

### 2.0.9.2 Lautsprecherleitungen

Für die Leitungen zwischen AV-Receiver, dem Mediatrix-Modul und den Lautsprechern sind Lautsprecherkabel mit einem Leitungsquerschnitt von 2,5mm<sup>2</sup> verbaut. Die zwei jeweils zusammengehörigen Leitungen sind miteinander verschweißt.

Auch im Mediatrix-Modul sind alle Lautsprecherleitungen von den Buchsen zu den Relais und zu den Ausgangsbuchsen mit diesen Kabeln verbunden.

### 2.0.9.3 Anschlüsse für User

Die für den User notwendigen Anschlussbuchsen sind in der Seite des Rack-Schranks eingebaut.

Die HDMI-Buchse ist an der Rückseite, wie an der Vorderseite mit einer HDMI-Buchse ausgestattet. Dies ermöglicht die Verwendung eines üblichen HDMI-Kabels, das an einen HDMI-Input des AV-Receivers angeschlossen ist.

Die XLR-Buchse ist mit einem 90cm langen XLR-Kabel mit einem Mikrofoneingang des Mischpults verbunden.

Die Cinch-Buchsen führen zum Line-Input des Mischpults. Das Kabel ist mit den Buchsen verlötet.

Die USB-Buchse ist mit einem USB 2.0 Druckerkabel an den Rechner angeschlossen.

Eine Schutzkontaktsteckdose ist an den Verteiler der anderen Geräte angeschlossen und wird nach Einschalten des Systems mit Strom versorgt. Zur Verkabelung wurden handelsübliche 1,5mm<sup>2</sup> Stromkabel verwendet.

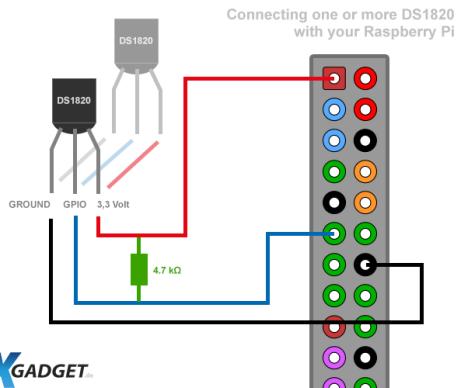


Abbildung 2.3: Schaltplan DS1820 Raspberry Pi

## 2.0.10 Gehäusebelüftung

Da die im Rack verbauten Geräte im Betrieb Wärme produzieren, ist ein Kühlssystem vorhanden. Ein 12V Lüfter sorgt für eine Zirkulation der Luft im Rack. Durch eine, von der Temperatur abhängigen Drehzahlregelung werden Störgeräusche durch den Lüfter minimiert. Es wird dafür gesorgt, dass der Lüfter nur bei Bedarf eingeschalten ist und auch nur mit einer angemessenen Drehzahl betrieben wird.

Im Zeitabstand von zwei Sekunden misst der Raspberry Pi die Temperatur im Inneren des Gehäuses und regelt demnach die Drehzahl des Lüfters. Für die Messung wird ein digitaler Temperatursensor mit der Bezeichnung „DS1820“ eingesetzt.

### 2.0.10.1 Digitaler Temperatursensor

Ein digitaler Temperatursensor bietet die Möglichkeit, im Vergleich zu einem temperaturabhängigen Widerstand, die Temperatur direkt in Grad Celsius auszulesen. Somit sind keine Umrechnungen von Ohm-werten auf Grad Celsius mehr notwendig.

### 2.0.10.2 Integration

Damit der Temperatursensor ausgelesen werden kann, muss er zunächst mit dem Raspberry Pi verbunden werden. Dazu werden ein GPIO-Pin, eine 3,3V Versorgung und der GND folgendermaßen an die Konnektoren des DS1820 angeschlossen:

```
pi@raspberrypi:~$ cd /sys/bus/w1/devices
[1]+  Done                      sudo nohup ./bins.sh &> /dev/null  (wd: /tmp)
(pi@raspberrypi:/sys/bus/w1/devices)
pi@raspberrypi:/sys/bus/w1/devices$ ls
10-000801a96106  w1_bus_master1
pi@raspberrypi:/sys/bus/w1/devices$ █
```

Abbildung 2.4: Putty-Ausgabe inkorrekt

```
pi@raspberrypi:/sys/bus/w1/devices$ ls
10-000801a96106  w1_bus_master1
pi@raspberrypi:/sys/bus/w1/devices$ cd 10-000801a96106
pi@raspberrypi:/sys/bus/w1/devices/10-000801a96106$ ls
driver  id  name  power  subsystem uevent  w1_slave
pi@raspberrypi:/sys/bus/w1/devices/10-000801a96106$ cat w1_slave
2a 00 4b 46 ff ff 0a 10 bf : crc=bf YES
2a 00 4b 46 ff ff 0a 10 bf t=21125
pi@raspberrypi:/sys/bus/w1/devices/10-000801a96106$ █
```

Abbildung 2.5: Putty-Ausgabe korrekt

### 2.0.10.3 Aufgetretene Probleme:

Laut des Tutorials sollte eine ID angezeigt werden, die dem Format 28-00000XXXXXXX entspricht. Doch in dem Versuchsaufbau erschienen mehrere IDs, die das Format 00-XX0000000000 haben. Keine von diesen IDs beinhaltete das File "w1 – slave". Dies hatte zur Folge, dass der Messwert des Temperatursensors nicht ausgelesen werden konnte.

Bei einem erneuten Versuch wurde nach folgendem Tutorial gearbeitet:

<https://www.raspberrypi-spy.co.uk/2013/03/raspberry-pi-1-wire-digital-thermometer-sensor/>

Zudem wurde die Schaltung neu aufgebaut, um sicherzustellen, dass diese kein Problem darstellt.

In dieser Anleitung wird eine notwendige Konfiguration im /boot/config.txt beschrieben. Folgende Codezeile muss eingefügt werden:

dtoverlay=w1-gpio,gpiopin=4

Nach der Änderung im /boot/config.txt File und einem darauffolgenden reboot, schien zum ersten Mal eine plausible Seriennummer auf.

Das Wechseln in das Verzeichnis der Seriennummer hat ohne Probleme funktioniert und das **w1-slave** File ließ sich anzeigen. Nun kann man mit der Variable „t“ die

```
pi@raspberrypi:/sys/bus/w1/devices/10-000801a96106$ cat w1_slave
32 00 4b 46 ff ff 0b 10 05 : crc=05 YES
32 00 4b 46 ff ff 0b 10 05 t=25062
pi@raspberrypi:/sys/bus/w1/devices/10-000801a96106$ █
```

Abbildung 2.6: Putty-Ausgabe Temperatur wird angezeigt

Temperatur in Grad Celsius ablesen. Der Wert  $t=21125$  entspricht  $21,125^{\circ}\text{C}$ .

Auch Erwärmung durch Reiben wird erkannt und der Temperaturwert änderte sich auf  $25,062^{\circ}\text{C}$ .

Nachdem die Temperatur nun manuell ausgelesen werden kann, gilt es dies auch automatisch zu bewerkstelligen. Dazu benötige ich ein Programm, das diese Aufgabe übernimmt und in gewissen Zeitabständen die Temperatur abfragt.

Dies kann in den Programmiersprachen „C“ oder „PYTHON“ gelöst werden.  
Aufgrund des Rates von Herrn Professor August Hörandl fiel die Entscheidung bei der Wahl der Programmiersprache auf PYTHON:

„Raspberry Pi“ steht für „Raspberry Python Interpreter“, d.h. der „Raspberry Pi“ verfügt über einen integrierten Interpreter für die Programmiersprache Python. Das spart zusätzlichen Aufwand, wie das Kompilieren von „C“-Programmen.

[?]

Weiters gibt es im Internet zahlreiche Anleitungen, wie ein Programm für diesen Anwendungsfall aussehen kann. Als Grundlage wurde das Beispielprogramm, aus dem bereits erwähnten Tutorial zur Hand genommen und angepasst.

```
def gettemp(id):
    try:
        mytemp = ''
        filename = 'w1_slave'
        f = open('/sys/bus/w1/devices/' + id + '/' + filename, 'r')
        line = f.readline() # read 1st line
        crc = line.rsplit(' ',1)
        crc = crc[1].replace('\n', '')
        if crc=='YES':
            line = f.readline() # read 2nd line
            mytemp = line.rsplit('t=',1)
        else:
            mytemp = 99999
        f.close()
    return int(mytemp[1])
```

```
except:  
    return 99999
```

In Folge wurde die Logik für die Umwandlung von der gemessenen Temperatur in einen Prozentwert zur Bestimmung des PWM-Duty-Cycles integriert.

```
def fanCon(mt):  
    st = 35.0 #Solltemperatur in Grad Celsius (Temperaturraum 35-55 Grad)  
    maxspeed = 3800 #Maximale Geschwindigkeit des Luefters in RPM  
    prozent = 0  
    if mt > st:  
        rpm=(mt-st)*190 #Umdrehungen Pro Minute  
        prozent=(mt-st)/5 #Prozent der Drehzahl  
        if prozent > 100:  
            prozent = 100  
  
    return prozent
```

Dieser Prozentwert wird an die PWM-Methode weitergegeben.

```
def pwm():  
  
    import RPi.GPIO as GPIO\  
    from time import sleep\  
  
    PWMin = 33 # PWM pin zum Anschluss des Luefters (PWM1 33,35)  
    GPIO.setwarnings(False)  
    GPIO.setmode(GPIO.BOARD) #Pinnummerierung setzen  
    GPIO.setup(PWMin,GPIO.OUT)  
    fan_pwm = GPIO.PWM(PWMin,100) #PWM Instanz mit Frequenz von 100Hz  
    erzeugen  
    fan_pwm.start(0)  
    while True:  
  
        #Auslesen der Temperatur  
        id = '10-000801a96106'  
        mt = gettemp(id)/float(1000) #Momentanttemperatur  
        print "Momentanttemperatur : " + '{:.3f}'.format(mt)  
        prozent = fanCon(mt)  
  
        print "Prozent: " + '{:.3f}'.format(prozent)  
  
        if prozent > 20:  
            fan_pwm.ChangeDutyCycle(prozent)
```

```
sleep(2)

if prozent < 20:
    fan_pwm.ChangeDutyCycle(0)
    sleep(2)

return
```

Bei einem Duty-Cycle-Wert unter 15-20% erhält der Motor des Lüfters nicht genug Leistung um zu starten. Zur Vermeidung von Schäden am Lüfter, ist der minimale Duty-Cycle-Wert bei 20% der maximalen Drehzahl angesetzt. Somit startet und stoppt der Lüfter bei 20%. Die maximale Drehzahl wird bei 55 Grad Celsius erreicht. Bei höheren Temperaturen dreht der Lüfter mit dieser Drehzahl weiter.

#### 2.0.10.4 Lüftersteuerung mit PWM

Um den Geräuschpegel möglichst gering zu halten, sollen die Lüfter nur so schnell laufen, wie es zu dem Zeitpunkt notwendig ist. Dafür muss eine Drehzahlregelung vorgenommen werden.

Dafür gibt es im Allgemeinen zwei Möglichkeiten:

1. Regelung durch vermindern und erhöhen der Spannung am Lüfter
2. Pulsweitenmodulation

Die Regelung über die Spannung wäre wiederum auf verschiedene Weisen möglich:

1. Die einfachste Variante ist sicherlich die Versorgungsspannung des Lüfters mittels temperaturabhängigen Widerstands zu regulieren. Dafür wäre kein Microprozessor notwendig. Allerdings können Zusatzfunktionen, wie eine zeitliche Temperaturstatistik oder Benachrichtigungen an den Techniker, bei zu hoher Temperatur nicht ergänzt werden.

Falls der Temperaturbereich nachträglich geändert werden soll, bringt das einen großen Aufwand mit sich.

1. Wenn man die Versorgungsspannung allerdings mit Hilfe eines digitalen Potentiometers reguliert, das aufgrund der Messung, des digitalen Temperatursensors vom Raspberry Pi konfiguriert wird, besteht die Möglichkeit, die oben genannten Funktionen zu implementieren.

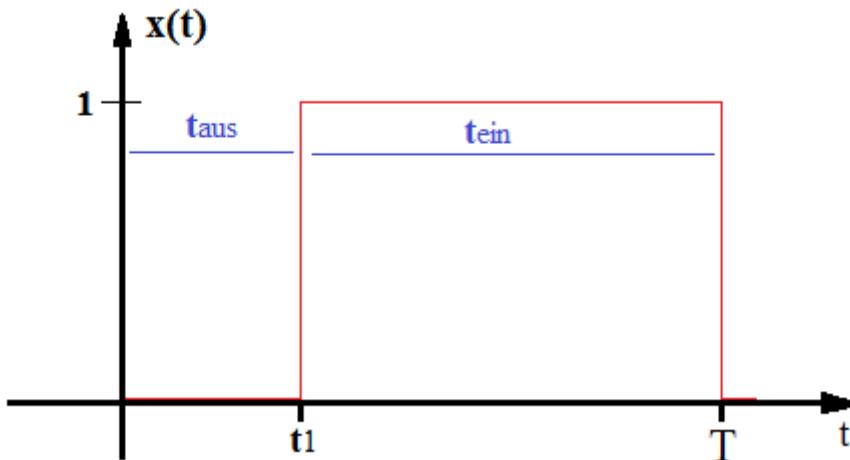


Abbildung 2.7: Pulsweite

Nachdem der verwendete Lüfter allerdings nur via PWM regelbar ist, fallen die anderen Varianten in diesem Fall weg.

PWM, oder auch Pulsweitenmodulation, ist ein Verfahren, bei dem

„[...] das Verhältnis zwischen der Einschaltzeit und Periodendauer eines Rechtecksignals bei fester Grundfrequenz variiert wird. Das Verhältnis zwischen der Einschaltzeit  und der Periodendauer  wird als das Tastverhältnis  $p$  bezeichnet. (laut DIN IEC 60469-1: Tastgrad) (engl. Duty Cycle, meist abgekürzt DC, nicht zu verwechseln mit Direct Current = Gleichstrom). “

[? ]

Das Ganze kann mit einem üblichen Taster und einem Elektromotor verglichen werden. Ein Druck auf den Taster sorgt dafür, dass der Motor die maximale Drehzahl erreichen kann. Wird der Taster wieder losgelassen, stoppt die Stromversorgung und der Motor wird nicht mehr angetrieben. Allerdings benötigt der Motor eine gewisse Zeit, um auf Höchstgeschwindigkeit zu beschleunigen und auch um wieder bis zum Stillstand abzubremsen. Dieses Verhalten kann genutzt werden, um die Drehzahl zu regulieren.

Betätigt man den Taster schneller und öfter hintereinander, also mit einer bestimmten Frequenz, so wird der Motor nie seine Höchstgeschwindigkeit erreichen, aber auch nicht zum Stillstand kommen. Diese Taster-Betätigungen können auch als High und Low oder Ein und Aus betrachtet werden. Wenn man den Betätigungsverlauf visualisieren möchte, eignet sich eine Rechteckschwingung.

Der Bereich zwischen zwei Hochpunkten ist die Periodendauer  $T$ . Innerhalb dieser

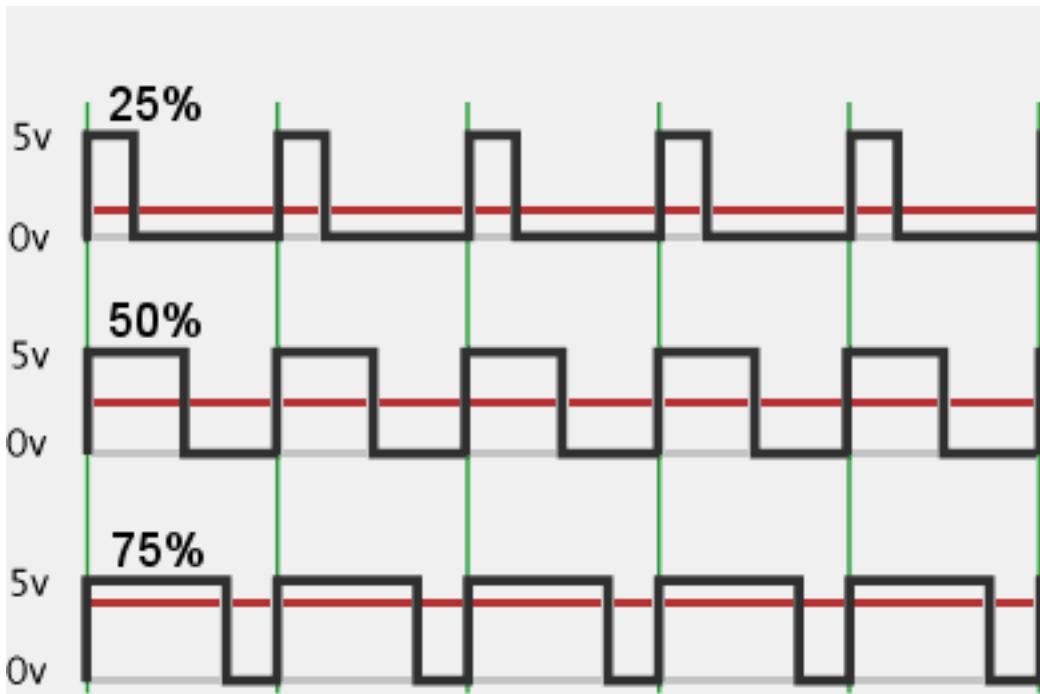


Abbildung 2.8: Duty-Cycle

Periodendauer kann nun, durch verschieben des Ausschaltzeitpunkts bestimmt werden, wieviel Prozent, der Spannung an den Motor weitergegeben werden (Siehe Abb.). Der Prozentwert wird als Duty-Cycle angegeben. 100% des Duty-Cycles entsprechen dem Betrieb mit der gesamten vorhandenen Spannung und somit der maximalen Drehzahl. Über den Mikroprozessor kann mit einer hohen Frequenz ein – und ausgeschaltet werden. Somit dreht sich der Motor (fast) gleichmäßig, jedoch in der gewünschten Geschwindigkeit.

#### 2.0.10.5 Generieren eines PWM-Signals mit dem Raspberry Pi

Mit PYTHON ist das Generieren eines PWM-Signals recht simpel.

```
GPIO.setup(PWMpin,GPIO.OUT) #Pin als Ausgang definieren
my_pwm = GPIO.PWM(PWMpin,100) #PWM Instanz mit Frequenz von 100Hz erzeugen
my_pwm.start(0) #PWM Instanz starten

my_pwm.ChangeDutyCycle(prozent) #einstellen des Duty-Cycles
```

Nach Eingabe dieser vier Befehle wird ein PWM-Signal mit fixem Prozentwert erzeugt. Mit Hilfe von Schleifen kann der Wert, je nach Bedarf variiert werden.

## 2.0.11 Quickstart-Guide

User, die das System verwenden wollen, können mit dem Handy oder Tablet-PC über einen QR-Code zum Login-Dialog navigieren. Für Personen, die das System auf einem Gerät ohne QR-Code Scanner verwenden wollen, befindet sich zusätzlich ein Link auf dem Rack. Die ersten Schritte in der Anwendung werden zudem kurz beschrieben.

## 2.0.12 Hardware für Infrarotfunktionalität

Um wirklich alle Geräte des Systems aus einer Hand bedienen zu können bedarf es einer Steuerung mittels Infrarot. Infrarot ist nach wie vor Standard für Fernbedienungen für AV-Geräte. Darum war es naheliegend diesen zu integrieren. Um Infrarotsignale senden zu können, wird eine Infrarotdiode und ein Konverter benötigt. Der Konverter wandelt den gespeicherten Befehl in elektrische Impulse um, die von der IR-Diode ausgegeben werden. Realisiert wurde die Einbindung dieser Komponenten durch ein Gerät, das vor einigen Jahren, von einem Professor unserer Schule entwickelt wurde. Ein entscheidender Vorteil dieses Gerätes ist die Möglichkeit, neue Infrarotbefehle über eine Lesediode einlernen und abspeichern zu können. Bei Anschaffung eines neuen Beamers können die Codes in wenigen Minuten ausgetauscht werden. Weiteren Änderungen sind nicht notwendig.

Bislang wurde das IR-Device noch via Serial-Connector angesteuert und die Stromversorgung mittels 9V-Batterie gewehrleistet. Da diese Technologien veraltet sind, wurde auf eine USB Schnittstelle umgerüstet, die die Signal- und Stromversorgung übernimmt. Die Platine ist mit einem PIC bestückt, der die Infrarotdioden ansteuert. Um Infrarotsignale in alle Richtungen senden zu können, ist eine der beiden Infrarotdioden außerhalb des Racks platziert. Somit kann die Signalrichtung an die Standorte der zu steuernden Geräte angepasst werden. Die zweite Diode befindet sich innerhalb des Racks, um Geräte, die im Rack verbaut sind ansprechen zu können.

## 2.0.13 Beleuchtungskonzept für Konferenzsaal

Der Konferenzsaal ist der größte Präsentationsraum unserer Schule.

Dort finden hauptsächlich Präsentationen und Vorträge statt. Bei diesen Arten von Veranstaltungen ist eine Ausleuchtung der vortragenden Personen besonders wichtig, vor Allem wenn, während des Vortrags Fotos von diesen Personen gemacht werden.

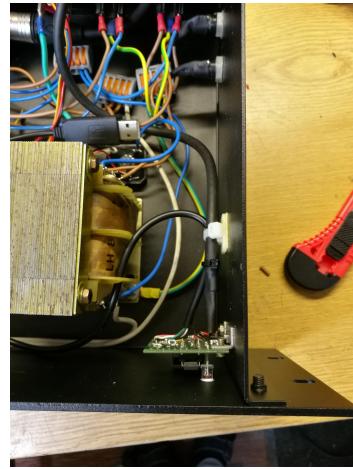


Abbildung 2.9: Infrarotplatine

### 2.0.13.1 Herausforderungen

In den meisten Fällen sind Folienpräsentationen mittels Beamer Teil solcher Vorträge. Um bestmögliche Lesbarkeit der Folien zu gewährleisten, darf nahezu kein Licht von anderen Lichtquellen als dem Beamer auf die Leinwand treffen. Der Lichtkegel sollte demnach formbar sein. Dies geschieht in der Regel durch eine Zoom-Funktion, Framing-Shutter oder Torblenden am Scheinwerfer.

Da der Raum nicht besonders hoch ist, ist die Montage eines Traversensystems, wie es in großen Sälen üblich ist, nicht möglich. Dementsprechend dürfen die verbauten Scheinwerfer auch kein hohes Eigengewicht haben.

Die installierten Scheinwerfer müssen je nach Bedarf auch auf unterschiedliche Positionen eingeleuchtet werden können. Folglich ist eine ausreichende Lichtstärke notwendig, um weiter entfernte Positionen ebenfalls mit genügend Licht versorgen zu können.

Grundlegend sind zwei Rednerpositionen vorgesehen.

### 2.0.13.2 Rednerpositionen

Zur Beleuchtung der Rednerpositionen eignen sich Profiler am besten. Diese lassen sich genau auf Positionen einrichten und mit Funktionen wie Framing Shutter und Fokus kann der Lichtkegel nach Bedarf justiert werden. Durch die Optik im Inneren der Scheinwerfer und den geringeren Abstrahlwinkel ist die Lichtausbeute höher als bei Flutern mit der selben Leistung.

Tabelle 2.1: Profiler

Profiler	Variante 1	Variante 2	Variante 3
Produkt	ADJ Saber Spot WW	ADJ Ikon Profile Pearl	Eurolite LED PFE-100 RGBW
Lichtquelle	LED	LED	LED
Leistung	15W	32W	100W
Farbtemperatur	WW 3000K	CW 7200K	CW
Multi-Color	Nein	Nein	RGBW
Farbfolie	Nein	Ja	Nein
Gobo	Nein	Ja	Nein
Zoom	Nein	Ja (manuell)	Ja (manuell)
Abstrahlwinkel	4°, 10°, 45°	15°- 30°	14°- 41°
Framing-Shutter	Nein	Ja	Ja
DMX	Ja	Ja	Ja
Stromversorgung	PowerCon	Kaltgeräte	PowerCon
Farbe	Schwarz	Weiß	Schwarz
Gewicht	1,2kg	2,7kg	10kg
Größe LBH	88 x 170 x 87 mm	448 x 150 x 127 mm	615 x 280 x 420 mm
Händler	Thomann.at	Thomann.at	Thomann.at
Preis/Stück	105€	248€	579€

### 2.0.13.3 Zusätzliche Beleuchtung

Falls mehr als zwei Personen präsentieren, muss auch die Fläche neben den Rednerpositionen beleuchtet werden. Statt weiteren Profilern bieten sich Fluter an. Aufgrund ihres breiten Abstrahlwinkels, decken sie große Flächen mit weichem Licht ab. Die Farbtemperatur ist bei Flutern üblicherweise eher warm.

Die folgenden Geräte sind für diesen Zweck geeignet. Je nach gewünschter Leistung und Budget kann ein Beleuchtungssystem zusammengestellt werden.

Variante 1: Variante 2: Variante 3:

Preiswert Semi-Professionell Professionell

### 2.0.13.4 Scheinwerfer:

### 2.0.13.5 Mögliche Konfigurationen:

### 2.0.13.6 Aufhängung:

Benötigte Gesamtlänge: 4m

Tabelle 2.2: Fluter

	Variante 1	Variante 2
Fluter		
Produkt	Stairville Mini Stage PAR CW/WW/A	Stairville Revueled 120 Cob 3200k Dmx
Lichtquelle	LED	LED
Leistung	42W	120W
Farbtemperatur	CW/WW/A 2800K-6000K	WW 3200K
Multi-Color	Nein	Nein
Farbfolie	Nein	Nein
Gobo	Nein	Nein
Zoom	Nein	Nein
Abstrahlwinkel	30°	50°
Framing-Shutter	Nein	Torblenden
DMX	Ja	Ja
Stromversorgung	Schuko	PowerCon
Farbe	Schwarz	Schwarz
Gewicht	1,7kg	5,5kg
Größe LBH	140 x 140 x 225 mm	557 x 258 x 164 mm
Händler	Thomann.at	Thomann.at
Preis/Stück	98€	277€

Tabelle 2.3: Preisoptimiert

Preisoptimiert	Klein	Mittel	Groß
Profiler	ADJ Saber Spot WW	ADJ Ikon Profile Pearl	ADJ Ikon Profil
Anzahl	4	2	4
Fluter	/	Stairville Mini Stage PAR CW/WW/A	Stairville Mini S
Anzahl	/	2	4
Gesamtgewicht	4,8kg	8,8kg	17,6kg
Gesamtleistung	60W	148W	296W
Gesamtpreis	420€	692€	1384€

Tabelle 2.4: Leistungsoptimiert

Leistungsoptimiert	Klein	Mittel	Groß
Profiler	ADJ Ikon Profile Pearl	ADJ Ikon Profile Pearl	Eurolite LE
Anzahl	4	4	2
Fluter	Stairville Revueled 120 Cob 3200k Dmx	ADJ Encore FR150z	ADJ Encor
Anzahl	2	2	2
Gesamtgewicht	21,8kg	24,6kg	33,8kg
Gesamtleistung	368W	388W	460W
Gesamtpreis	1546€	1968€	2164€

Tabelle 2.5: Eutrac

Montage: Preisoptimiert	Preis/Stück	Klein	Mittel	Groß
3-Phasen Aufbau Stromschiene 2m	36,70€	2	2	2
Pendelabhangung 60cm	20,93€	4	4	4
Schienenadapter	14,90€	4	4	8
Befestigungsklammer	5,59€	4	4	4
Einspeiser	10,43€	1	1	1
Längsverbinder	10,43€	1	1	1
Endkappe	2,09€	1	1	1
Gesamtpreis		262,03€	262,03€	321,63€

Aufbauschiene Type: 25.../225...

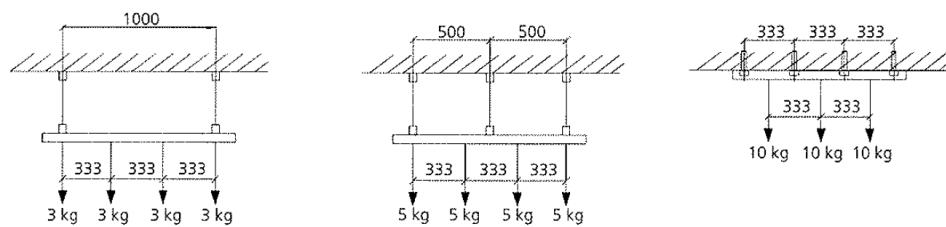


Abbildung 2.10: EUTRAC Montage

**2.0.13.6.1 EUTRAC-Stromschiene:** Die Aufbauanleitung kann von der EUTRAC-Website heruntergeladen werden.

Alle Produkte und Preise sind am 03.04.2018 von amazon.at entnommen worden.

EUTRAC-Aufbauschienensystem 25.../225... 3 Phasen 230V

4 Hängepunkte/Meter bei 3x 10kg Punktlast

3 Hängepunkte/Meter bei 4x 5kg Punktlast

2 Hängepunkte/Meter bei 4x 3kg Punktlast

**2.0.13.6.2 1-Punkt Traverse:** Aufgrund der höheren Last sind EUTRAC-Stromschienen nicht mehr verwendbar. Alternativ dazu eignet sich eine 1-Punkt Traverse. Zur Befestigung können Half-Coupler und Stahlseile verwendet werden. Die 1-Punkt-Traverse verfügt über keine interne Stromführung. Es ist eine 230V-Schutzkontaktleitung zur Traverse notwendig. Eine Serienschaltung von Geräten mit Power In und Out ist mit geeigneten Patchkabeln möglich.

Alle Produkte und Preise sind am 03.04.2018 von thomann.at entnommen worden.

Tabelle 2.6: Traverse

Montage: Leistungsoptimiert	Preis/Stück	Klein	Mittel	Groß
Global Truss F31200 Traverse 2,0 m	44€	2	2	2
Global Truss 812 Halbschelle (Geräte)	5,90€	4	4	8
Global Truss 81702 Half Coupler Small Eye	5,90€	4	4	4
Global Truss Konischer Verbinder für F31-F34 Traverse	7,40€	1	1	1
Stairville Mirror Ball Chain 100cm DIN	5,90€	4	4	4
Stairville Locking Carabina 6mm	1,49€	8	8	8
the box Flugöse M8 x 30mm	4,88€	4	4	4
Gesamtpreis		197,64€	197,64€	221,24€

Tabelle 2.7: PreisPreisoptimiert

Preisoptimiert	Klein	Mittel	Groß
Preis Geräte	420€	692€	1384€
Preis Aufhängung	262,03€	262,03€	321,63€
Gesamtpreis	682,03€	954,03€	1705,63€

#### 2.0.13.6.3 Einbau der Traverse:

Anmerkung:

Die Konstruktion muss nach dem Einbau von befugten Personen geprüft und abgenommen werden.

#### 2.0.13.7 Gesamtpreise der Konfigurationen:

### 2.0.14 Status-Erkennung der Geräte

Wenn Geräte mittels Infrarot eingeschaltet werden, erhält der Sender keine Information, ob das Signal angekommen ist und ob das Gerät wirklich läuft. Um dies festzustellen verfügt das System über einen Stromausgang, an dem gemessen wird, ob ein gewisser Strom fließt. Wenn das der Fall ist, erhält der Raspberry Pi ein Signal. Somit kann der User über den Status der Geräte informiert werden.

In dieser Installation wird so der Einschaltzustand des Beamers bestimmt.

Anders als der Beamer, besitzt der AV-Receiver einen „Trigger-Output“. An jenem

Tabelle 2.8: PreisLeistungsoptimiert

Leistungsoptimiert	Klein	Mittel	Groß
Preis Geräte	1546€	1968€	2164€
Preis Aufhängung	197,64€	197,64€	221,24€
Gesamtpreis	1743,64€	2165,64€	2385,24€

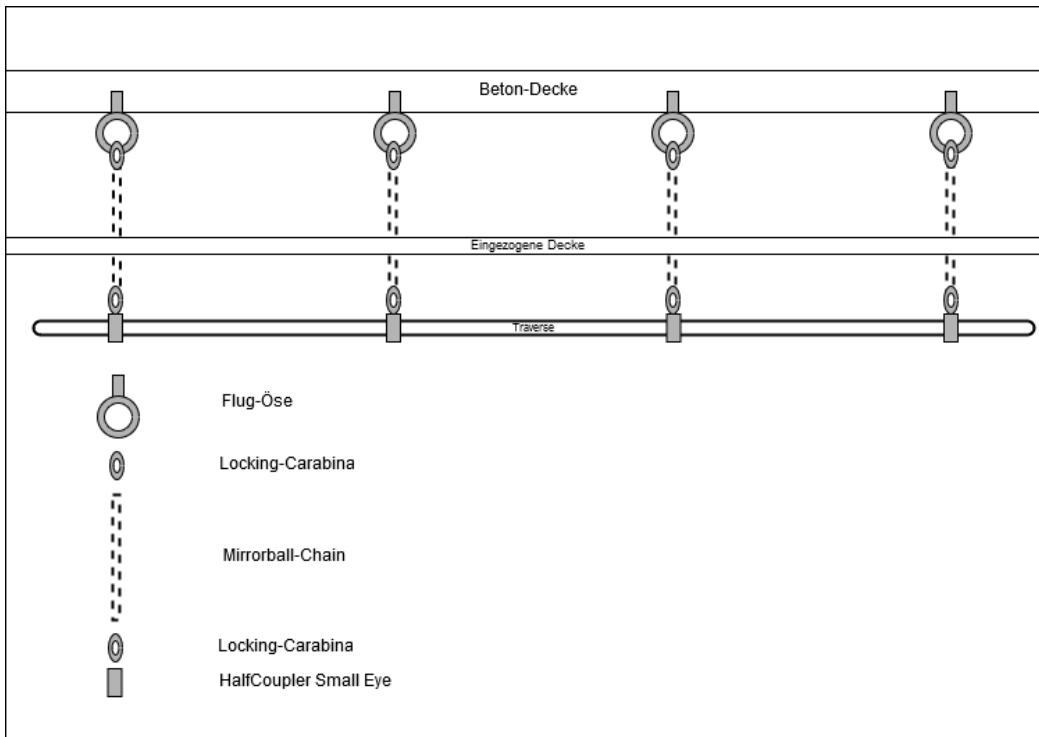


Abbildung 2.11: 1-Punkttraverse Montage

liegen 12V an, sobald das Gerät einsatzbereit ist. Via 3,5mm Klinkenbuchse wird diese Spannung in das Mediatrix-Modul geleitet und dort auf 3.3V umgewandelt. Der Raspberry Pi ist so in der Lage, die Spannung auszuwerten.



# 3 Betriebssystem

Ein Betriebssystem (BS) bezeichnet eine Sammlung von Programmen, die

".. die Grundlage der mögl. Betriebsarten bildet und bes. die Abwicklung von Benutzerprogrammen steuert und überwacht." [noa91, S. 288]

Im folgenden Kapitel wird das verwendete Betriebssystem, sowie der Webserver und die Sicherheitsmaßnahmen erläutert. Weiters wird die Software, die zum Betrieb der Hardware (siehe Kapitel 2, Seite 3) notwendig ist, beschrieben.

## 3.1 Raspbian

Bei der Auswahl eines geeigneten BS für den Raspberry Pi ist zu beachten, dass der verwendete ARM-Prozessor nur von wenigen Betriebssystemen unterstützt wird. Dadurch beschränkt sich das Feld der Möglichkeiten auf beinahe ausschließlich Linux-Distributionen, da der Linux-Kernel den Prozessortyp unterstützt. Für den Einstieg beim Arbeiten mit dem Raspberry Pi eignet sich das auf Debian basierte und auf den Pi zugeschnittene Betriebssystem Raspbian. Weitere kompatible Betriebssysteme sind: Arch Linux ARM, RISC OS oder Coder.

Raspbian wurde in der Version 9 in diesem Projekt verwendet, da es schon einen Grundstock an wichtige Funktionen für die Verwendung des Pis, wie zum Beispiel einen leistungsfähigen Paketmanager, integriert hat. [Sch14, S. 13 bis 16] Jedoch wurde nicht die Voll-Version, sondern eine Lite-Version verwendet. In dieser Version ist kein grafisches Userinterface (GUI) installiert, was für den Anwendungsfall in diesem Projekt nicht von nötzen ist und somit nur zusätzliche Leistung verbraucht hätte. [noad] Ein weiterer Vorteil von Raspbian ist, dass die gesamte Hardware des Raspberry Pis unterstützt wird. Die Raspberry Pi Foundation bezeichnet Raspbian auch als ihr offiziell unterstütztes Betriebssystem. [noad]

## 3.2 Sicherheit

Die Sicherheit spielt in diesem Projekt natürlich einen zentralen Rolle. Dafür wurde auf Ebene des Betriebssystems der Fokus auf verschlüsselte Verbindungen und eine Firewall gelegt.

### 3.2.1 SSL und TLS

SSL bzw. TLS (im weiteren als SSL bezeichnet) wurde in diesem Projekt durch HTTPS (Hypertext Transfer Protokoll Secure) und WSS (WebSocket Secure) integriert.

SSL (Secure Sockets Layer) beschreibt ein Verfahren, bei dem unsichere Protokolle wie HTTP (Hypertext Transfer Protokoll) und WS (WebSocket) in eine verschlüsselte Kommunikation eingebettet werden und weiters auch eine Authentifizierung von Server und Client möglich ist. Das Verfahren arbeitet mit öffentlichen und privaten Schlüsseln. Das Prinzip der Schlüsselpaare funktioniert folgendermaßen: Wollen zwei oder mehrere Geräte miteinander über ein öffentliches Netz verschlüsselt kommunizieren, erstellt sich jeder Teilnehmer ein Schlüsselpaar, bestehend aus privatem und öffentlichem Schlüssel. Will nun ein Gerät mit dem Server verschlüsselt kommunizieren, wird ihm vom Server sein öffentlicher Schlüssel zugeschickt. Alle weiteren Nachrichten werden mit diesem öffentlichen Schlüssel verschlüsselt. Die Nachrichten, die gesendet werden, sind nun nicht mehr lesbar und können nur mit dem privaten Schlüssel vom Server entschlüsselt werden. Um die Authentifizierung des Servers zu gewährleisten, sollten die öffentlichen Schlüssel (Zertifikate) von einer offiziellen Stelle beglaubigt werden.[WH17, S. 675 und f.]

### 3.2.2 Firewall

"Eine Paketfilter-Firewall analysiert den Datenverkehr im Netzwerk, indem sie die Header der Pakete betrachtet. Anhand von festgelegten Filterregeln wird entschieden, ob ein Paket geblockt und verworfen wird oder seinen Weg fortsetzen darf." [Jur13, S. 715]

In Linux wird bereits eine Firewall namens Netfilter/IPTables mit ausgeliefert. Während Netfilter die Kernel-Komponente, die die Filterung vornimmt darstellt, ist IPTables das Programm, das diese steuert.

Die Konfiguration von IPTables gliedert sich in Tabellen und Listen. Generell können die Tabellen *filter*, *nat* und *mangle* bearbeitet werden. Die beiden Letzteren sind für die Grundfunktion der Firewall nicht notwendig und werden deshalb auch nicht weiter ausgeführt. Jede Tabelle enthält Listen, in denen Regel stehen, wie Pakete

```

pi@raspberrypi:~ $ sudo ufw status
Status: active

To           Action    From
--           ----     ---
22/tcp        ALLOW     Anywhere
443/tcp       ALLOW     Anywhere
80/tcp        ALLOW     Anywhere
22/tcp (v6)   ALLOW     Anywhere (v6)
443/tcp (v6) ALLOW     Anywhere (v6)
80/tcp (v6)   ALLOW     Anywhere (v6)

pi@raspberrypi:~ $

```

Abbildung 3.1: Die konfigurierten UFW-Regeln

behandelt werden sollen. Die Regeln werden nacheinander abgearbeitet und sobald eine Regel zutrifft wird keine weitere mehr angewandt. Die *filter*-Tabelle gliedert sich in 3 vordefinierte Listen: INPUT, OUTPUT und FORWARD.

- INPUT ist eine Liste von Regeln, die bei ankommenden Paketen angewandt werden.
- Die Regeln der Liste OUTPUT werden bei ausgehenden Paketen verwendet.
- FORWARD wird zusätzlich angewandt, wenn der Server als Router fungiert.

Um das Verwalten der Firewall zu vereinfachen, wurde in diesem Projekt ufw (Uncomplicated Firewall) verwendet. Ufw ist ein Programm, das ein Kommandozeilen-Programm für die Erstellung von Regeln für IPTables bereitstellt.[noam]

Es wurden Regeln für die Verwendung von HTTPS, SSH und HTTP erstellt, alle anderen Ports wurden geblockt. (siehe Abbildung 3.1, Seite 37)

### 3.3 Webserver

Ein Webserver ist eine Anwendung, die Inhalte eines Computers einem Netzwerk zur Verfügung stellt, unter Verwendung diverser Netzwerkprotokolle, wie zum Beispiel HTTP. Die Software hat weiters die Aufgabe die Datenablage und den Zugriff zu regeln. Als Webserver wurde das Opensource-Projekt Apache verwendet. Apa-

che ist eine weitverbreitete Anwendung und auf fast allen Plattformen verfügbar. Viele Linux-Distributionen, auch Raspbian, haben Apache bereits standardmäßig vorinstalliert.[Jur13, S. 663 bis 664]

### 3.3.1 Aufbau von Apache

Der Apache Webserver ist in Modulen aufgebaut. Jedes Modul stellt dabei eine Funktion des Webservers dar. Standardmäßig werden einige Module beim Installieren des Webservers aktiviert, um die Grundfunktionalität zu gewährleisten. Weitere Module werden durch die Directive *LoadModule* in der Konfigurationsdatei geladen. Meist wird diese Anweisung in eine eigene .load-Datei ausgelagert.

Ein weiteres Feature von Apache sind Virtual-Hosts. Ein Webserver kann dadurch mehrere Webseiten gleichzeitig aktiviert haben. Dabei gibt es zwei Arten von Virtual-Hosts: namensbasierte und Ip-Adressen-basierte Hosts. Die Konfiguration eines solchen Virtual Hosts wird in der Konfigurations-Datei mit der Direktive *VirtualHost* vorgenommen. Meist wird diese in eine extra Datei ausgelagert, die dann in dem Ordner *sites-enabled* abgelegt wird.

Die Konfiguration ist in mehrere Konfigurationsdateien aufgeteilt. Die Hauptkonfigurationsdatei ist */etc/apache2/apache2.conf*. In dieser befinden sich Voreinstellungen und Erklärungen über die Konfigurationsmöglichkeiten. Weiters werden am Ende die weiteren Dateien eingebunden.[Jur13, S. 665 bis 672]

Nachfolgend werden die wichtigsten Dateien und Ordner im Konfigurationsverzeichnis */etc/apache2* beschrieben.

```

/etc/apache2
├── apache2.conf
├── envvars
├── mods-available
└── mods-enabled
    ├── ...
    ├── php7.0.conf
    ├── php7.0.load
    ├── proxy.conf
    └── proxy.load
    └── ...
├── ports.conf
├── sites-available
└── sites-enabled
    ├── 000-default.conf
    └── 000-default-ssl.conf
└── ssl
    └── server.crt

```

```

└─ server.key

```

- **mods-available:** Enthält alle verfügbaren Module in der Form von .load Dateien.
- **mods-enabled:** Enthält symbolische Links auf die Modul-Dateien in *mods-available*. Nur die in diesem Ordner verlinkten Module werden auch beim Start des Webservers geladen.
- **ports.conf:** In dieser Datei wird definiert auf welchen Ports der Webserver ange-sprochen werden kann.
- **sites-available:** Enthält alle verfügbaren Virtual-Hosts, in der Form von Konfigu-rationendateien.
- **sites-enabled:** Enthält symbolische Links auf die Virtual-Hosts-Dateien in *sites-available*. Nur die in diesem Ordner verlinkten Hosts werden auch beim Start des Webservers geladen.
- **ssl:** Dieser Ordner enthält die Zertifikate für die SSL-Verbindung (siehe Kapitel 3.2.1, Seite 36).

### 3.3.2 Verwendete Module und Konfiguration

In diesem Projekt wurden neben den Standard-Modulen, die Module: SSL, Proxy mit der Erweiterung Proxy\_WS und Rewrite verwendet.

**SSL:** Dieses Modul stellt die Verwendung von SSL-gesichertem Http zur Verfügung (Https). Für die Verwendung wird ein SSL-Zertifikat benötigt, dieses kann entweder mit Hilfe der Software OpenSSL selbst erstellt werden, oder von einer offiziellen Stelle verifiziert werden. [noag] Bei diesem Projekt wurde das Zertifikat selbst erzeugt, da offizielle registrierte Zertifikate nur kostenpflichtig zu erhalten sind.

**Proxy und Proxy WS:** Dieses Modul stellt die Funktion eines Proxy-Servers zur Verfügung. Unter einem Proxy-Server versteht man einen Server, der als eine Art Vermittler auftritt. Er leitet Anfragen von einem Netz in ein anderes und erledigt somit die Anfrage an den Ziel-Host für den anfragenden Client.[noan] Die Erweiterung Proxy WS ermöglicht es Websocket-Verbindungen über einen Proxy in ein anderes Netz aufzubauen. Dies war in diesem Projekt notwendig, da Ratchet (siehe Kapi-tel 4.2.1.3, Seite 50) keinen verschlüsselte Websocket-Verbindungen (WSS) unterstützt. Um trotzdem die Verbindung zu verschlüsseln, wurde ein Proxy auf dem localhost eingerichtet, dadurch ist die Verbindung zum Webserver verschlüsselt und wird nur Server-Intern auf eine nicht verschlüsselte Verbindung umgelegt.

**Rewrite:** Dieses Modul ermöglicht es, die URL der Anfrage während der Laufzeit zu ändern. Dies geschieht mittels eines Regulärenausdrucks und einer “RewriteRule” (Umschreibungsregel).[noaf] Dieses Modul wurde eingesetzt, um jede Anfrage, die eine ungesicherte Verbindung herstellen will, automatisch auf einen gesicherten Verbindung umzuleiten.

## 3.4 OLA

OLA (Open Lighting Architecture) ist Teil des Open Lighting Projects, welches von Simon Newton 2004 gegründet wurde. Das Ziel des Projekts ist es, qualitativ hochwertige Open-Source Software für die Licht- und Unterhaltungsindustrie zu entwickeln. OLA vereint verschiedene DMX-Interfaces und kümmert sich um die Kommunikation mit diesen. Weiters werden eine Vielzahl von DMX- und DMX-over-IP-Protokollen unterstützt. Ein Vorteil der Software ist, dass sie auch auf dem im Raspberry Pi verbauten ARM-Prozessor läuft. Des Weiteren bietet OLA auch eine C++-Schnittstelle, diese wird im folgenden Kapitel beschrieben.[noai]

### 3.4.1 C++-Schnittstelle

In diesem Projekt wurde das DMX-Signal über die C++-Schnittstelle der OLA gesteuert. Um OLA über C++ zu steuern werden zwei Klassen benötigt. Einerseits der *StreamingClient* und andererseits der *DmxBuffer*.

Der *StreamingClient* stellt die eigentliche Kommunikationskomponente zu OLA dar. Zum Senden von DMX512 muss der Client indiziert und ein Setup durchgeführt werden. Hierbei können Optionen, wie der automatische Start der OLA Software angegeben werden. Über die *StreamingOptions*-Klasse werden schon alle Default-Einstellungen geliefert, welche für unser Projekt ausreichen. Während des Setups wird die Verbindung zu OLA hergestellt. Die Methode *sendDmx* sendet einen *DmxBuffer* an das gegebene Universum.[noah]

```
// Create a new client.
ola::client::StreamingClient ola_client(
    (ola::client::StreamingClient::Options()));

// Setup the client, this connects to the server
if (!ola_client.Setup()) {
    cerr << "Setup failed" << endl;
    exit(1);
}

//send DMX-Buffer to universe
if (!ola_client.SendDmx(UNIVERSE, buffer)) {
    cerr << "Send DMX failed" << endl;
    exit(1);
}
```

Der *DmxBuffer* representiert alle Channels des Universums. Der Wert eines Channels

kann mit der Methode *SetChannel* geändert werden. Beim Erstellen einer Instanz werden alle Kanäle automatisch auf Null gesetzt. In diesem Projekt werden über die PHP-Extension (siehe Kapitel 4.1, Seite 45) mehrere Channels mit deren Werten übergeben. Diese werden dann im Buffer festgeschrieben und mit der *SendDmx*-Methode an das DMX-Universum gesendet.[noah]

```
//Create new DmxBuffer
ola::DmxBuffer buffer;

//set all channels which are not 0 in buffer
for(unsigned int i = 0; i < sizeof(channels)/sizeof(channels[0]); i++){
    if(channels[i] > 0){
        buffer.SetChannel(i,channels[i]);
    }
}
```

## 3.5 WiringPi

WiringPi ist eine C-Library, mit der die Pins am Raspberry Pi (auch GPIO) gesteuert werden können. Die Befehlsstruktur ist sehr der Programmierung eines Mikroprozessors nachempfunden und macht es somit einfach aus einem C++-Programm die GPIOs zu steuern und zu lesen. Weiters wird auch ein Kommandozeilentool mit ausgeliefert, welches ebenfalls erlaubt Pins des Raspberry Pis zu bedienen. Das erlaubt die Steuerung der GPIOs aus höheren Programmiersprachen wie PHP, was in diesem Projekt für die Überprüfung der Leistung an AV-Receiver und Beamer notwendig ist. Eine weitere Funktionalität, die WiringPi mit sich bringt, ist eine Library zur seriellen Kommunikation, welche für das Steuern des Infrarot-Moduls verwendet wird.[noao] Nachfolgend wird das Ansteuern der Pins und die serielle Kommunikation näher erläutert.

### 3.5.1 Serielle Steuerung

Zur Kommunikation mit dem Infrarot-Modul wurde die *Serial Library* verwendet. Um diese Library zu verwenden, muss zuerst die Header-Datei wiringSerial.h eingebunden werden. Mit der Funktion *serialOpen* wird die Verbindung zu dem Gerät hergestellt und initialisiert. Danach kann mit den Methoden *serialPrintf*, *serialFlush*, *serialDataAvail* und *serialGetchar* mit dem Gerät kommuniziert werden. Geschlossen wird die Verbindung mit *serialClose*.

Um Daten an das Infrarot-Modul zu senden, wird zuerst einen Verbindung aufgebaut und anschließend werden mit Hilfe von *serialPrintf* Daten an das Gerät gesendet. *SerialPrintf* ist eine Nachbildung der *printf*-Funktion des Systems und nimmt somit

*strings* und *char* entgegen. Um sicher zu stellen, dass die Daten an das Gerät gesendet werden und nicht in einem Zwischenspeicher liegen, muss nach jedem Kommando die Methode *serialFlush* ausgeführt werden. Weiters ist zu beachten, dass die Übertragung der Daten und das Auswerten durch das serielle Gerät Zeit benötigt. Das muss auch im Programm am Raspberry Pi berücksichtigt werden. Hierfür bietet sich die Methode *sleep* aus der *WiringPi*-Library an. Um diese zu nutzen muss der Header *wiringPi.h* inkludiert werden.[noao] Der nachstehende Code zeigt einen einfachen Sendevorgang, wie er in diesem Projekt integriert ist.

```
#include <wiringPi.h>
#include <wiringSerial.h>

//open serial connection to IR-Device
int fd = serialOpen(IR::dev, 9600);

delay(200);

//reset the IR-Device
serialPrintf(fd, ":~:");
serialFlush(fd);
delay(200);

//send code to IR-Device
serialPrintf(fd, ("p"+code+"]").c_str());
serialFlush(fd);
delay(300);

//close the connection
serialClose(fd);
```

Zum Empfangen von Daten werden die Methoden *serialDataAvail* und *serialGetchar* verwendet. Wenn *SerialDataAvail* aufgerufen wird, wird ein Boolean zurückgegeben, welcher anzeigt, ob Daten vom Gerät zum Lesen bereitstehen. Mit der Methode *serialGetchar* wird ein *char* von der seriellen Verbindung eingelesen. Wenn kein Zeichen vorhanden ist, blockiert diese Methode das Programm um bis zu zehn Sekunden. Somit muss vor jedem Versuch ein Zeichen zu lesen, geprüft werden, ob überhaupt eines zur Verfügung steht.[noao] Der nachstehende Code zeigt einen einfachen Lesevorgang, wie er in diesem Projekt integriert ist.

```
#include <wiringSerial.h>

//open serial connection to IR-Device
int fd = serialOpen(IR::dev, 9600);

//initialise read string
```

```

string read = "";

//as long as data is available, add char to string
while (serialDataAvail (fd))
{
    read += serialGetchar (fd);
}

```

### 3.5.2 Pin Steuerung

Um festzustellen, ob der AV-Receiver und der Beamer eingeschalten sind und nicht im Standby-Modus laufen, wurde eine Leistungsmessung integriert, welche über einen GPIO-Pin abgefragt werden kann. Liegt an einem Pin Spannung an (HIGH), dann ist das Gerät eingeschalten. Liegt keinen Spannung an (LOW), ist es im Standby-Modus. Vor dem Senden von Infrarotbefehlen kann somit festgestellt werden, ob das Gerät auch eingeschalten ist und diese Empfangen kann. Um den Status, HIGH oder LOW, eines Pins festzustellen, wird das Kommandozeilentool von WiringPi über PHP gestartet. Das Tool wird mit dem Befehl *gpio* aufgerufen und bietet einen sehr großen Funktionsumfang.[noao] Die für diese Projekt wichtigen Optionen sind:

- **gpio -g:** Die Option -g stellt die Nummerierung der Pins auf die GPIO-Nummern ein.[noao]
- **gpio mode <pin> in/out:** Mit dem Befehl *mode* wird definiert, ob ein Pin ein Eingangs- (in) oder eine Ausgangspin (out) ist. Einem Pin muss immer zuerst ein Mode zugewiesen werden, bevor er verwendet werden kann[noao]
- **gpio read <pin>:** Mit dem Befehl *read* wird der aktuelle Status eines Eingangspins festgestellt. Liegt Spannung an (HIGH), wird Eins zurück geliefert. Wenn keine Spannung an liegt (LOW), liefer *read* Null.

Der *gpio*-Befehl wird mit der Methode *exec* aufgerufen. Im nachfolgenden Codesegment wird ein Pin zuerst auf “in” gesetzt und dann der aktuelle Wert gelesen. Diese Logik ist auch im Mediatrix-System integriert.

```

//Set Pin mode
exec('gpio -g mode '.$gpio.' in');

//Read status of Pin and check if high
if(exec('gpio -g read '.$this->gpio) == 1){
    ...
}
```



# 4 Backend

## 4.1 PHP Extension

Um die in C++ programmierte Steuerung der DMX-Schnittstelle, sowie das Infrarot-Modul in PHP verwenden zu können, wurde in diesem Projekt je eine PHP-Extension für DMX und Infrarot entwickelt. PHP bietet die Möglichkeit Erweiterungen (Extensions) zu entwickeln und einzubinden. Bei Extensions müssen zwei Typen unterschieden werden PEAR- und PECL-Pakete. PEAR steht für “PHP Extension and Application Repository” und enthält Extension, die in PHP programmiert wurden. Hingegen steht PECL für “PHP Extension Community Library”, deren Extensions in C programmiert sind. PECL-Erweiterungen müssen auf dem Server vor der Verwendung kompiliert und in der Konfigurationsdatei von PHP eingebunden werden.[WH17, S. 74] Da die Steuerung der DMX-Schnittstelle und des Infrarot-Moduls in C++ programmiert ist, war bei diesem Projekt eine PECL-Extension notwendig. Diese wurde mit Hilfe der Library PHP-CPP erstellt, da diese den Aufwand eine Erweiterung zu entwickeln stark verringert.

### 4.1.1 PHP-CPP

PHP-CPP bieten den Vorteil, dass Extensions entwickelt werden können, ohne ein detailliertes Wissen über PHP-Extensions und die Zend-Engine zu haben. Diese Library erlaubt es ganze C++-Klassen über Extensions auch in PHP aufrufbar zu machen. Hierbei sind drei Dateien essentiell: die *C++-Datei*, die *INI-Datei* und das *Makefile*[noaj]

#### 4.1.1.1 C++-Datei

Diese Datei enthält die C++-Klasse und die *get\_module*-Methode, die jede PHP-Extension besitzen muss. Durch das Einbinden der Library wird die Klasse *Php* verfügbar. Diese stellt Methoden, sowie zusätzliche Datentypen zur Verfügung. Wenn nun eine C++-Klasse für eine PHP-Extension verwendet werden soll, muss diese von der Klasse *Php::Base* abgeleitet werden. So werden einer Methode immer Parameter des Typen *Php::Parameters* übergeben und als Rückgabewert wird *Php::Value* erwartet.

Die `get_module`-Methode muss immer in einem `extern "c"` Block definiert werden, da sie sonst nicht von PHP, das in C programmiert ist, verwendet werden kann. Innerhalb der `get_module`-Methode erstellt man eine neue statische Instanz der `Php::Extension`-Klasse und der `Php::Class`-Klasse. `Php::Extension` repräsentiert die Extension und `Php::Class` die selbst programmierte C++-Klasse. Der `Php::Class` werden nun der Reihe nach die Methoden der C++-Klasse zugewiesen. Weiters wird auch spezifiziert, wie diese Methode in PHP heißen soll und welche Parameter von welchem Datentyp sie haben soll. Am Ende wird die `Php::Class` an die `Php::Extension` angehängt und die `Php::Extension` an die Library zurück geliefert.[noaj] Nachstehend ist ein Code-Beispiel für eine C++-Extension für PHP dargestellt, wie sie auch in diesem Projekt integriert ist.

```
#include <phpcpp.h>

//create new class DMX , which is a child of Php::Base
class DMX : public Php::Base {

public:

    //define the methode with sendChannel
    static Php::Value sendChannel(Php::Parameters &params){ . . . }

    //define the methode with blackout
    static Php::Value blackout(){ . . . }
}

/**
 * tell the compiler that the get_module is a pure C function
 */
extern "C" {

    //define the get_module methode
    PHPCPP_EXPORT void *get_module()
    {
        //create new Extension object
        static Php::Extension extension("DMX", "1.0");

        //create new Php::Class-object
        Php::Class<DMX> dmx("DMX");

        /**
         * add the Methode sendChannel to the dmx object,
         * with the name sendChannel
         * the methode has one paramter with the name channels
         * and is an Array
    }
}
```

```

        */
    dmx.method<&DMX::sendChannel> ("sendChannel", {
        Php::ByVal("channels", Php::Type::Array)
    });

    /**
     * add the Methode blackout to the dmx object,
     * with the name blackout
     */
    dmx.method<&DMX::blackout> ("blackout");

    // add the class to the extension
    extension.add(std::move(dmx));

    // return the extension
    return extension;
}
}

```

#### 4.1.1.2 INI-Datei und Makefile

Die INI-Datei enthält eine Zeile, in der wie in der php.ini die Extension geladen wird. Hierfür wird der im Makefile definierte Name verwendet. Diese Datei wird dann in das Konfigurationsverzeichnis von PHP geladen und bei jedem Start von PHP aufgerufen. Nachstehend ist der Inhalt der INI-Datei dargestellt, wie er in diesem Projekt verwendet wurde.[noaj]

```
extension=DMX.so
```

Das Makefile enthält alle Anweisungen, die für den Kompilervorgang wichtig sind. Hier kann der Name der Extension, sowie das Konfigurationsverzeichnis von PHP eingestellt werden. Weiters können auch noch Optionen, wie die Verwendung von anderen Libraries, wie Ola (siehe Kapitel 3.4, Seite 40) oder WiringPi (siehe Kapitel 3.5, Seite 41), fürs Kompilieren hinzugefügt werden. Dieses Makefile wird dann verwendet, um die C++-Datei zu kompilieren und die so entstandenen Dateien, sowie die INI-Datei, an die richtigen Stellen in der PHP-Konfiguration zu kopieren.[noaj]

## 4.2 Websocket

### 4.2.1 Client - Server Kommunikation

Das Internet wurde um das Konzept, dass ein Client Daten vom Server anfordert und ein Server diese Anfragen bearbeitet, konstruiert. Daran hat sich für lange Zeit nichts geändert bis 2005 Webseiten durch AJAX dynamischer wurden. Mit AJAX können Client und Server eine bidirektionale Verbindung aufbauen, das heißt beide Seiten haben die Möglichkeit gleichzeitig Daten zu senden und zu empfangen. [W3Cd]

Webapplikationen wurden stetig komplexer und benötigten mehr Daten als jemals zuvor. Eine herkömmliche HTTP-Anfrage hat einen sehr großen Overhead, da bei jeder Anfrage ein header und cookies zum Server übermittelt werden müssen. Dadurch vergrößert sich die Latenz und der Benutzer muss länger warten. Das größte Problem ist, dass die meisten der Header und Cookies nicht nötig sind, um die Anfrage zu bearbeiten. [UE]

Diese Probleme können mit WebSockets gelöst werden. [UE]

#### 4.2.1.1 Technische Spezifikation - Nuss

Die WebSocket-Spezifikation ermöglicht eine persistente "socket"-Verbindung zwischen einem Webbrower und einem Server. Beide Seiten können, wann sie wollen, mit dem Senden von Daten beginnen. Der Client baut eine Verbindung durch einen Prozess namens "WebSocket-Handshake" auf. Dieser Prozess beginnt mit einer normalen HTTP-Anfrage an den Server. Ein Upgrade-Header muss in der Anfrage enthalten sein, damit der Server weiß, dass der Client eine WebSocket-Verbindung aufbauen möchte. [W3Cd]

```
var verbindung = new WebSocket("ws://mediatrix.at/ws");

verbindung.onopen = () => {
    verbindung.send("Das WebSocket ist offen");
};

verbindung.onmessage = nachricht => {
    console.log("Nachricht" + nachricht + "erhalten");
};
```

Eine WebSocket-Verbindung wird geöffnet, indem man den WebSocket-Konstruktor aufruft um ein neues Objekt zu erstellen. [W3Cd]

Das URL-Schema fängt im Gegensatz zu herkömmlichen URLs nicht mit "http://" sondern mit "ws://" an. Allerdings gibt es auch bei WebSockets eine verschlüsselte Variante mit dem Schema "wss://". Damit können sichere Verbindungen zwischen Browsern und Servern aufgebaut werden. Weiters stehen verschiedene Ereignis-Handler zur Verfügung, um auf das Eintreten bestimmter Ereignisse reagieren zu können. Dadurch weiß der Client zum Beispiel, ob die Verbindung erfolgreich geöffnet werden konnte oder ob eine Nachricht eingegangen ist. [tut]

Sobald eine Verbindung zum Server erfolgreich aufgebaut wurde, kann der Client sofort anfangen mit der ".send()" Methode Nachrichten an den Server zu übermitteln. Bisher konnten nur Strings versendet werden, aber seit der neuesten Spezifikation ist es auch möglich, binäre Daten zu senden. Der Server hat ebenfalls die Möglichkeit jederzeit Mitteilungen zu senden. Tritt dieser Fall ein, wird das ".onmessage"-Event ausgelöst. Dieses Ereignis übergibt ein Ereignisobjekt, mit dem auf die Nachricht zugegriffen werden kann. [wha]

#### 4.2.1.2 Kommunikation mit dem Mischpult - Nuss

Im Lern- und Informationszentrum wird ein Soundcraft Ui16 als Mischpult für die Mikrofone und den Audio-Eingang verwendet. Dieses verfügt über eine proprietäre Benutzeroberfläche. Der Hersteller hat sich sehr viel Mühe gegeben, dieses System abzusichern, sodass es nur mit der hauseigenen Software bedient werden kann. [? ]

Ich habe den Aufbau und die Verhaltensweise des Mischpults analysiert, um herauszufinden, wie ich darauf zugreifen kann, um es über die Mediatrix-Oberfläche ansteuern zu können. Dabei habe ich erkannt, dass auf dem Mischpult ein Socket.io-Server läuft. Socket.io ist eine JavaScript Bibliothek für Echtzeit-Webapplikationen. Das Mischpult verwendet die von Socket.io implementierte Funktion zur Generierung von Session-Ids. Eine Session-Id ist eine einzigartige Kennzahl, die der Server benötigt, um die Anfragen den jeweiligen Clients zuordnen zu können. Weiters weiß der Server mithilfe der Identifikationsnummern, wieviele einzigartige Clients verbunden sind. Dadurch ist es möglich, dass mehrere Clients gleichzeitig mit dem Server verbunden sind. Genauso funktioniert das Soundcraft Ui16. [? ]

Um eine Verbindung aufzubauen, muss zuerst eine Session-Id angefragt werden. Sobald der Server diese bereitstellt, kann die Verbindung geöffnet werden. Der Server wartet ein paar Sekunden nach dem Senden der Id, ob ein Client sich verbinden möchte. Ist dies nicht der Fall, wird die erstellte Session-Id wieder verworfen. Weiters muss die Verbindung mit dem Mischpult am Leben gehalten werden. Das bedeutet, dass alle paar Sekunden eine Nachricht, im Falle des Soundcraft Ui16 lautet die Nachricht "3::::ALIVE", gesendet werden muss. Ansonsten wird die Verbindung vom Server geschlossen.

Bevor ich damit anfangen konnte, Befehle an das Mischpult zu senden, musste ich

erstmal die Befehlsstruktur analysieren. Dabei ist mir aufgefallen, dass Befehle des Clients mit "3:::" beginnen müssen. Nachrichten vom Server fangen hingegen mit "2:::" an. Die Befehle sind logisch aufgebaut. sie fangen mit "3:::" an anschließend folgt der Befehl. Um Werte wie Lautstärke zu setzen ist folgender Befehl notwendig "3:::SETD^i.0.mix^Wert". Nach dem i steht die Nummer des Kanals und nach dem mix^ wird der gewünschte Wert platziert. Die Lautstärke kann einen Dezimalwert zwischen Null und Eins haben.

Zuerst habe ich mit JavaScript einen Prototypen entwickelt um die Befehle auszuprobieren. Allerdings sollten alle Geräte über den Raspberry Pi gesteuert werden, deswegen musste die Kommunikation mit dem Mischpult am Server umgesetzt werden. Dieser Server baut eine Client-Verbindung zum Socket.io-Server auf. Zuerst wird eine Session-Id mit der cURL-Funktion von PHP angefragt. Mit cURL können HTTP-Anfragen in PHP gemacht werden. Der Server sendet eine Session-Id zurück und die WebSocket-Verbindung auf das Mischpult kann geöffnet werden. Wenn nun auf der Benutzeroberfläche zum Beispiel ein Regler eines Mikrofons betätigt wird, sendet der Client dem Server die Daten und dieser generiert das passende Kommando, um sie weiter an den Server zu senden. [? ]

Dank WebSockets ist die Latenz minimal, obwohl die Befehle über zwei Verbindungen geschickt werden müssen. Mit herkömmlichen HTTP-Anfragen wäre die gesamte Kommunikation wesentlich langsamer.

#### 4.2.1.3 Ratchet

Um einen Websocket-Server auf Seiten des Backends aufzusetzen, wurde die PHP Library Ratchet verwendet. Ratchet erlaubt es sehr einfach einen ereignisbasierten (event based) Websocket-Server aufzusetzen. Für die Erstellung eines Server auf Ratchet basis ist die Application-Klasse das wichtigste Element, diese enthält die Methoden *onOpen*, *onMessage*, *onClose* und *onError*. Die Application-Klasse wird in Kapitel 4.2.1.3.1, Seite 51, näher beschrieben. Diese vier Methoden enthalten die eigentliche Logik des Servers. Wenn nun ein Websocket-Server initialisiert werden soll, muss die nachstehende Struktur eingehalten werden.

```
$server = IoServer::factory(
    new HttpServer(
        new WsServer(
            new Application()
        )
    )
);
```

Wie der obige Code zeigt, muss zuerst ein *IoServer* erstellt werden, welcher einen *HttpServer* benötigt. Der *HttpServer* wiederum benötigt einen *WSSErver*, welchem

die Application-Klasse übergeben wird.[noak] Diese Server-Klassen werden nachfolgen kurz beschrieben:

- **IoServer:** Diese Klasse stellt die Basis dar. Sie kümmert sich um den Verbindungsaubau und -abbau und um alle Fehler, die im Programm auftreten. Weiters sendet und empfängt sie auch Daten vom Client.[noak]
- **HttpServer:** Diese Klasse behandelt die Http-Request, die der Server empfängt. Sie wartet bis ein kompletter Request übertragen wurde und gibt ihn dann erst weiter.[noak]
- **WsServer:** Diese Komponente verarbeitet die Websocket-Verbindungen mit den Browsern nach dem W3C Websocket Standard.[noak]

**4.2.1.3.1 Application-Klasse** Diese Klasse beinhaltet die eigentliche Logik des Websocket-Servers. Das *MessageComponentInterface* gibt hier die vier Methoden, die von Ratchet verlangt werden, vor und muss implementiert werden.[noak] Diese vier Methoden sind:

- **onClose:** Diese Methode wird von Ratchet aufgerufen, wenn eine Websocket-Verbindung geschlossen wird. Als Argument wird die Websocket-Verbindung übergeben.[noak]
- **onOpen:** Diese Methode wird aufgerufen, wenn eine Verbindung mit dem Websocket-Server hergestellt wird. Als Argument wird die Websocket-Verbindung übergeben.[noak]
- **onError:** Wenn ein Error mit der Websocket-Verbindung auftritt wird diese Methode von Ratchet aufgerufen. Hier wird zusätzlich auch die aufgetretene Exception mit übergeben.[noak]
- **onMessage:** Immer wenn eine Nachricht über Websocket von einem Client an den Server übermittelt wird, wird diese Methode aufgerufen. Als Argumente werden die Websocket-Verbindung, von der die Nachricht kommt, und die Nachricht übergeben.[noak]

Nachstehend folgt ein Beispiel für eine Application-Klasse, nach deren Prinzip auch die Application-Klasse in diesem Projekt gestaltet ist.

```
class Application implements MessageComponentInterface {
    private $client;

    public function onOpen(ConnectionInterface $conn) {
        //save the connection
    }
}
```

```

        $this->client = $conn;
    }

    public function onMessage(ConnectionInterface $from, $msg) {
        //json_decode the Message of the client
        $commands = json_decode($msg);

        //check if DMX command is set
        if (isset($commands["dmx"])) { . . . }
    }

    public function onClose(ConnectionInterface $conn) {
        // remove the connection
        $this->client = null;
    }

    public function onError(ConnectionInterface $conn, \Exception $e) {
        //close the connection
        $conn->close();
    }

}

```

## 4.3 DMX

Das DMX (Data Multiplexed) Protokoll wurde erstmalig durch das USITT (United States Institute for Theatre Technology) definiert. Es beschreibt die Steuerung von bis zu 512 Dimmern über eine serielle Verbindung. Das Protokoll findet hauptsächlich in der Theater- und Bühnenbeleuchtungstechnik Anwendung. Hierbei werden die Scheinwerfer über ein Busnetzwerk mit einem Wertebereich von 8-bit gesteuert. Es gilt als State-of-the-art und ist durch die DIN 56930-2 Norm definiert.[noac]

### 4.3.1 Funktionsweise des Protokolls

Die Informationen werden im DMX Protokoll digital übertragen, wobei hier zwischen einer positiven und negativen Spannung von ungefähr 2,5 Volt unterschieden wird. Ein Einser entspricht einer positiven Spannung für 4 µs und ein Nuller einer negativen Spannung für die selbe Zeitspanne. DMX verwendet eine 8-bit Datenlänge. Der Wertebereich eines Kanals liegt also zwischen Null und 255. Die jeweiligen Werte für jeden Kanal werden nacheinander gesendet. Am Anfang jedes Signals wird eine Reset-Sequenz gefolgt von einem Startbyte gesendet (Abbildung 4.1, Seite 53) .[noab].

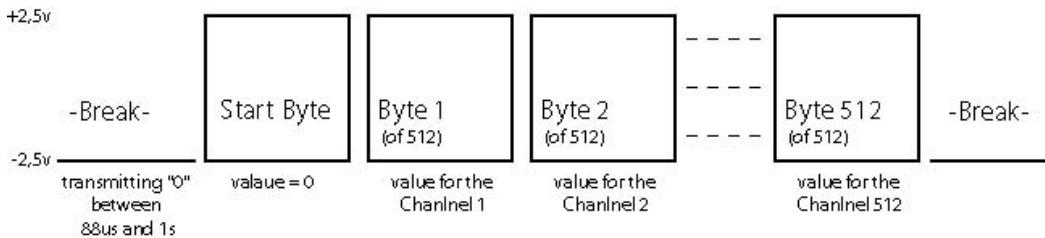


Abbildung 4.1: Darstellung des DMX-Signals

Der Vorteil dieses Protokolls ist, dass alle Empfänger nur an ein Kabel angeschlossen werden müssen und die meist schon vorhandene XLR-Verkabelung genutzt werden kann.[noac]

Ein Nachteil ist, dass bei vielen genutzten Kanälen das Signal und somit auch die Refreshrate sehr gering wird. Das bedeutet, dass in der Praxis DMX mit weniger Geräten betrieben werden sollte.[noac]

### 4.3.2 Anwendung in diesem Projekt

In diesem Projekt wurde DMX als Schnittstelle zur Beleuchtung integriert. Dadurch bleibt das System flexibel. Der Administrator bei der Installation nur an welcher Kanal, welchem Scheinwerfer zuzuordnen ist. Weiters wird dadurch auch garantiert, dass das System mit allen DMX-Stanadard konformen Scheinwerfern funktioniert. Ein weiterer Vorteil, der sich durch die Integration des Protokolls ergibt, ist, dass nicht nur Dimmer, sondern auch RGB- und RGBW-Scheinwerfer gesteuert werden können. Die Art des Scheinwerfers wird gemäß der Anzahl der durch den Administrator angegebenen Kanäle pro Scheinwerfer bestimmt. So ist es auch möglich, dass verschiedene Typen gemischt werden können.

Wenn vom User der Befehl zur Änderung einer Einstellung an einem der Scheinwerfer gesendet wird, wird im Hintergrund das zu diesem Scheinwerfer passende Scheinwerfer-Objekt aufgerufen, welches über die PHP-Extension (siehe Kapitel 4.1, Seite 45) und OLA (siehe Kapitel 3.4, Seite 40) den entsprechenden Wert des DMX-Kanals ändert.

## 4.4 Infrarot

Infrarot bezeichnet elektromagnetische Wellen im Bereich zwischen 780 Nanometer und einem Millimeter.[noaa] Der Name leitet sich von dem lateinischen Präfix "Infra", was "unterhalb" bedeutet, und dem Wort "Rot" ab und bedeutet somit "unter Rot". Das soll beschreiben, dass die Wellenlänge unterhalb der von Rot, also dem niedrigsten noch sichtbaren Bereich, liegt (Abbildung 4.2, Seite 54). Da Infrarot vom Menschlichen

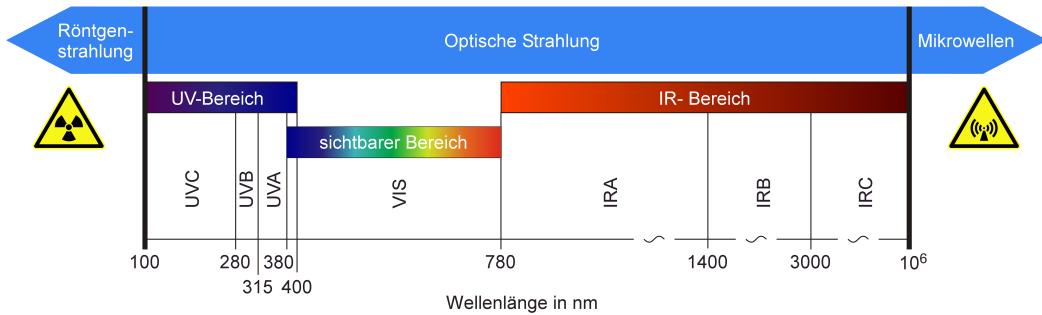


Abbildung 4.2: Darstellung des elektromagnetischen Spektrums

Auge nicht wahrgenommen wird, wird es zur Steuerung von unterschiedlichsten Geräten verwendet. Die Steuerung über Infrarot ist in Produkten der Elektronikbranche sehr verbreitet, jedoch wurde kein einheitlicher Standard geschaffen. Die verschiedenen Standards funktionieren jedoch nach ähnlichen Prinzipien. Einer der verbreitetsten ist der RC-5 Standard, auf welchen im folgenden Kapitel näher eingegangen wird.[noal]

#### 4.4.1 Funktionsweise des Protokolls (RC-5)

Der RC-5-Code hat eine Gesamtlänge von 14 Bit. Ursprünglich waren diese Bit folgendermaßen aufgeteilt: zwei Startbit, ein Togglebit, fünf Adressbits und sechs Kommandobits. Die Startbits dienen zur Synchronisation zwischen Sender und Empfänger. Das Togglebit wird bei jedem Tastendruck geändert, somit kann zwischen dem Gedrückthalten und dem Öftersdrücken eines Knopfs der Infrarotfernbedienung unterschieden werden. Die Adressbits enthalten die Geräte-Adresse. Mit den 5 Adressbits können somit 32 Geräte gesteuert werden. Abschließend werden noch die Kommandobits gesendet, welche den Code des Kommandos, das an das Gerät übermittelt wird, enthält. Mit den 6 Bit können 64 verschiedene Kommandos an ein Gerät geschickt werden. Da 64 Bit für die Steuerung von Geräten oft nicht ausreicht, wurde der RC-5 Standard nachträglich geändert. So kann das zweite Startbit ebenfalls als Kommandabit verwendet werden.

Für die Übertragung wird eine 36 kHz Trägerfrequenz verwendet. Um ein Bit mit dem Wert "1" zu übertragen, wird für 889 µs wiederholt ein ungefähr 6 µs langer Impuls, gefolgt von einer 20 µs langen Pause, übertragen. Anschließend wird 889 µs nichts gesendet. Um den Wert "0" zu übertragen, wird genau das Gegenteil gesendet. Zuerst wird 889 µs nicht gesendet und erst dann wird der Impuls wiederholt für 889 µs übertragen. Daraus ergibt sich für ein Bit eine Übertragungsdauer von 1,778 ms und somit einen Gesamtübertragungs dauer eines RC-5-Codes von 24,889 ms (Abbildung 4.3, Seite 55).[noae]

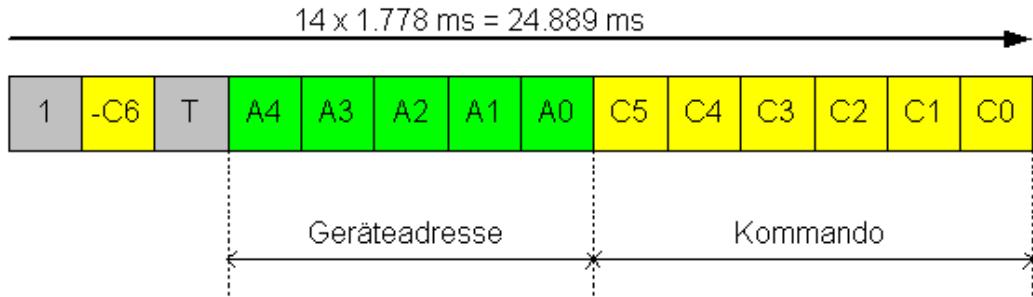


Abbildung 4.3: Darstellung eines RC-5-Codes

#### 4.4.2 Anwendung in diesem Projekt

Die Kommunikation mit dem AV-Receiver und dem Beamer wird in diesem Projekt mittels Infrarot gesteuert. Infrarot bietet die größte Kompatibilität, da sowohl bei Beamern als auch bei AV-Receiver die Bedienung mittels Infrarotfernbedienung Branchenstandard ist. Zum Senden der Signale wird der bereits erwähnte Infrarotsender (siehe Kapitel ??, Seite ??) über eine C++-Schnittstelle angesprochen. Der Administrator muss bei der Installation des Systems, alle benötigten Infrarotcodes über das in PHP programmierte Konsolenprogramm einlesen.

Der Vorteil von Infrarot für dieses Projekt besteht darin, dass sich die Steuerung nicht herstellerspezifisch ändert, sondern nur die verwendeten Codes und Protokolle. Dadurch kann jeder Beamer und AV-Receiver in das System integriert werden.

Ein großer Nachteil des Infrarotprotokolls ist, dass nur eine einseitige Kommunikation stattfindet und somit die Anwendung keine Rückmeldung über den Status und den Erfolg eines Befehls auslesen kann.

## 4.5 JSON Web Token

Ein JSON Web Token (JWT) wird in Webapplikationen eingesetzt, um die fortlaufende Authentifizierung zu gewährleisten. Er besteht aus drei Teilen: Header, Payload und Signature.

- Der **Header** enthält den Verschlüsselungsalgorithmus und die Art des Tokens
- Die **Payload** enthält beliebig viele Key-Value-Paare, auch Claim genannt, im JSON-Format. Hier sind drei Arten von Claims zu unterscheiden. Die registrierten Claims sind durch den RCF 7519 Standard, welcher JWTs beschreibt, definiert. Öffentliche Claims müssen bei der IANA (Internet Assigned Numbers Authority) gemeldet werden. Der dritte Typ sind private Claims, diese können beliebig vergeben und

genutzt werden.

- Die **Signature** enthält das Ergebnis, der Verschlüsselung des Headers und der Payload mit einem geheimen Schlüssel. Für die Verschlüsselung wurden beide Teile base64-Encodiert.

Durch die Signatur des JWTs durch den Server wird sichergestellt, dass jede Änderung der Claims im Payload erkannt und der JWT dadurch als ungültig angesehen wird.[Hoo16] In diesem Projekt wurden JWTs für die Authentifizierung der Clients verwendet. Nach der Anmeldung wird dem Client ein JWT geschickt, welchen er bei jeder weiteren Kommunikation mit schicken muss, da sonst die Verbindung abgebrochen wird. Dadurch wird sichergestellt, dass nur vom Server autorisierte Clients Zugriff auf das Backend haben. Um JWTs zu integrieren wurde die Library Firebase-JWT verwendet.

#### 4.5.1 Firebase-JWT

Durch die Integration der Library wird die *JWT*-Klasse verfügbar. Um einen JWT zu erstellen müssen zuerst die Claims in einem Array definiert. In diesem Projekt wurden fünf registrierte Claims und ein Privater verwendet. Das Array wird anschließend gemeinsam mit dem Schlüssel und dem zuverwendenden Algorithmus an die Methode *encode* übergeben, welche den JWT zurück liefert. Nachfolgend ist eine Integration der *JWT*-Klasse, wie sie in diesem Projekt getätigter wurde.[noa18]

```
$data = [
    // Issued at: time when the token was generated
    'iat' => time(),

    // Json Token Id: an unique identifier for the token
    'jti' => base64_encode(random_bytes(32)),

    // Issuer
    'iss' => "Mediatrix",

    // Not before
    'nbf' => time(),

    // Expire
    'exp' => time() + $this->expireSec,

    // Data related to the signer used
    'data' => [
        ...
    ]
]
```

```

    // User name
    'userName' => $username
]
];

//genreate JWT
$jwt = JWT::encode($data, $this->key, 'HS256');

```

Um zu verifizieren, ob ein JWT gültig ist und um die Daten der Payload zu bekommen, wird die Methode *decode* verwendet. Diese Methode wird der JWT, der Schlüssel und der zuverwendenden Algorithmus übergeben. Zurückgeliefert wird die Payload als Objekt. Nachstehen ist ein Beispiel für das Decodieren eines JWTs, wie es auch in diesem Projekt integriert ist.[noa18]

```

try {
    //decode JWT
    $jwt = JWT::decode($jwt, $this->key, array("HS256"));

    //Print user
    echo "User: {$jwt->data->userName}";

} catch (ExpiredException $ex) {

    //Session Expired
    echo 'JWT expired: ' . $ex->getMessage() . "\n";
}

```

## 4.6 SQLite

Als Datenbank wurde in diesem Projekt SQLite gewählt. Bei SQLite handelt es sich um eine Datei-basierte Datenbank, das bedeutet, dass im Hintergrund eine C-Library arbeitet und die Daten in einem File im Dateisystem ablegt. Im Vergleich zu anderen Datenbanksystem wie MySql ist hier kein zusätzlicher Datenbank-Server notwendig. Der Zugriff auf die Datenbank ist, wie bei anderen Datenbanken, mit SQL möglich. Ein Problem von SQLite ist, dass bei jedem Schreibvorgang die gesamte Datenbank-Datei neu geschrieben wird. Somit ist das Einfügen von Daten in die Datenbank sehr langsam. Weiters sollte auch beachtet werden, dass Dateien auf dem Webserver meist öffentlich einsehbar sind. Das sollte bei SQLite zumindest für das Datenbankverzeichnis verhindert werden, da sonst die gesamte Datenbank von einem Client heruntergeladen werden kann. Ein Vorteil hingegen entsteht beim Lesen von Daten. Da im Hintergrund keine Verbindung hergestellt werden muss, ist der Lesevorgang schneller als bei herkömmlichen Datenbanken.[WH17, S. 617]

#### 4.6.1 Vorteile für diese Projekt

SQLite bringt für dieses Projekt einen großen Vorteil, da die Datenbank als Datei abgelegt werden kann und kein eigener Datenbank-Server betrieben werden muss. Das bringt eine Reduktion der verwendeten Ressourcen, die durch den Raspberry Pi nur sehr knapp bemessen zur Verfügung stehen. Weiters finden auch sehr selten Schreibzugriffe statt, weswegen die etwas langsamere Geschwindigkeit hierbei keine Rolle spielt. Ein Vorteil gegenüber einem einfachen JSON-Files, in das die Daten abgespeichert werden, ist, dass auch Relationen abgebildet werden können.

# 5 Frontend

Ein Frontend ist die höchste Schicht beim Entwickeln von Software, da sie am nächsten zur Eingabe des Benutzers ist. Sie stellt die visuelle Benutzeroberfläche da und wird verwendet, um das System zu bedienen. Da wir uns entschieden haben, eine Webapplikation zu schreiben, wurde die Oberfläche mit den Webstandards HTML5, CSS3 und JavaScript umgesetzt.

**HTML:** kurz für Hypertext Markup Language, ist eine Markup-Sprache, die verwendet wird, um digitale Dokumente zu strukturieren. Diese Dokumente sind das Fundament auf dem das World Wide Web basiert. [htm]

**CSS:** kurz für Cascading Style Sheets, ist eine Stylesheet-Sprache, die entwickelt wurde, um Darstellungsvorgaben vom Inhalt abzutrennen. Zusammen mit HTML können Webseiten entworfen und gestaltet werden. CSS verfügt über verschiedene Layout-Module, wie Flexbox oder CSS-Grid. Im nachstehenden Kapitel werde ich diese beiden Technologien etwas genauer erläutern und aufzeigen, warum ich welches verwendet habe. [cssa]

**JS:** kurz für JavaScript, ist eine dynamische Skript-Sprache. Obwohl die meisten Leute sie nur als Sprache für Webseiten kennen, gibt es auch andere Umgebungen von Javascript, wie zum Beispiel node.js oder Apache CouchDB. Diese Sprache läuft auf der Client-Seite im Webbrower. Sie kann verwendet werden um das Verhalten einer Webseite beim Eintreten von verschiedenen Ereignissen zu definieren. [jav]

## 5.1 Responsives Design

Mit dem Anstieg an Smartphones und anderen mobilen Geräten ist die Zahl der Benutzer von mobilen Seiten stark gestiegen. Webseiten müssen quer über alle Browser, Bildschirmauflösungen und Seitenverhältnisse funktionieren. Der grafische Aufbau einer Seite, insbesondere die Anordnung der einzelnen Elemente, muss angepasst werden, um auf weniger Platz zu funktionieren. Weiters müssen andere Eingabemöglichkeiten wie Touchscreens beachtet werden. Responsives Design ist eine aktuelle Technik, die es ermöglicht mit HTML und CSS3-Media-Queries das einheitliche Anzeigen von Inhalten auf Webseiten zu gewährleisten. Das Layout einer Webseite wird flexibel gestaltet, sodass dieses auf Desktop-Geräten sowie auf Tablets oder Smartphones eine

gleichbleibende Benutzerfreundlichkeit bietet. [Sha, S. 21 - S. 38]

Modernes responsives Design wurde erst durch die Einführung von CSS3-Media-Queries möglich. Diese ermöglichen es, verschiedene Eigenschaften der Geräte zu erkennen, um andere CSS-Eigenschaften zu laden. Zum Beispiel könnte eine Navigation, die auf Desktops die gesamte Breite des Bildschirms einnimmt, auf Smartphones als "Hamburger"-Menü implementiert werden. Dadurch spart man Platz, ohne Funktionen einsparen zu müssen. Beim responsiven Design gibt es zwei verschiedene Ansätze. Der Mobile-First Ansatz baut, wie der Name schon sagt, auf der mobilen Darstellung auf. Das heißt man fängt an die Seite für Smartphones zu entwerfen und verändert diese mit Media Queries so, dass sie auf Tablet- und Desktop-Geräten gut angezeigt wird. Der zweite Ansatz ist, Desktop-First und das genaue Gegenstück zu Mobile-First. In diesem Projekt habe ich mich dafür entschieden, mit der Desktop-Ansicht anzufangen, da ich dadurch schnellere funktionierende Prototypen der Oberfläche herstellen konnte. [Sha? , S. 38]

#### **5.1.0.1 Vorteile von anpassungsfähigen Webseiten**

- Es muss nur eine Version erstellt werden, die sowohl für mobile-Geräte als auch Desktop-Geräte funktioniert.
- Nutzergruppen der verschiedenen Geräte können optimal bedient werden.
- Es kann besser auf die Bedürfnisse der mobilen Nutzer eingegangen werden.

[Sha]

CSS bietet verschiedene Werkzeuge um Layouts auf Webseiten umzusetzen. Die zwei neuesten Modelle sind Flexbox und CSS-Grid. Diese sind mit Hinblick auf responsives Design entwickelt worden. Wie diese Technologien funktionieren und wie sie eingesetzt werden können, werde ich in den nächsten Kapiteln genauer erläutern. [W3Cb, W3Cc]

## **5.2 CSS**

### **5.2.1 Flexbox**

Flexbox, offiziell "CSS Flexible Box Layout Module Level 1", ist eine neue Art und ein neues Konzept, um eindimensionale Layouts auf Webseiten umzusetzen. Früher hat man allen Elementen mit dem Klassen-Selektor fixe Positionen, Maße und Eigenschaften zugewiesen. Mit Id-Selektoren wurden einzelne Elemente weiter modifiziert. Doch bei Flexbox werden grundlegende Regeln festgelegt, wie sich Elemente innerhalb eines

Containers zu verhalten haben. Dies macht das Verhalten der Seite auch bei einer Änderung der Bildschirmauflösung vorhersagbar. Anschließend ist es dem Browser überlassen, die Breite, Höhe, Position und Anordnung, entsprechend den vordefinierten Regeln, zu wählen. Damit wird die Implementierung von Webseiten, die ihr Design an verschiedene Bildschirmauflösungen anpassen müssen, plattformübergreifender und effizienter. [W3Cb]

### 5.2.1.1 Das Konzept

Die Grundidee ist es, dem Flex-Container die Möglichkeit zu geben, die Maße der Elemente so zu verändern, dass der Platz bei unterschiedlichen Bildschirmauflösungen bestmöglich ausgenutzt ist. Um das zu erzielen, lässt das Elternelement die Kindelemente je nach Bedarf wachsen oder schrumpfen. Es werden bestimmte Regeln festgelegt, wie z.B. die Mindestbreite der Elemente, die Achse, an der die Objekte ausgerichtet werden oder, ob die Elemente in die nächste Zeile wandern sollen, wenn es in einer zu eng wird. [CSSb]

### 5.2.1.2 Technische Spezifikation

Innerhalb eines <div> Tags können die einzelnen Elemente ihre Größe "flexibel" verändern. Sie wachsen, um freien Platz zu verwenden oder schrumpfen, damit mehr Elemente pro Zeile platziert werden können. Weiters achtet Flexbox darauf, dass Elemente innerhalb des Elternobjekts bleiben und nicht darüber hinauswandern. Der große Vorteil des Flexbox-Layouts ist die Möglichkeit, die Achse, an der die Elemente ausgerichtet werden, bei einer Änderung der Auflösung anzupassen. Dadurch ist das Layout sehr flexibel, was Orientierungsänderungen bei mobilen Geräten oder Auflösungsänderungen auf Desktop-Geräten betrifft. [MDNa]

### 5.2.1.3 Erklärung anhand eines realen Beispiels

Die Aufgabenstellung: auf der Webseite soll eine Navigation auf der linken Seite angezeigt werden, die auf mobilen Geräten an den unteren Rand des Bildschirms wandert (siehe Abbildung 5.1). Bei diesem Layout ist entscheidend, dass die Reihenfolge der Elemente unabhängig vom Markup geändert werden kann.

Mithilfe von Flexbox ist dieses Verhalten einfach zu erzielen. Wie bereits erwähnt, gibt es bei Flexbox Elternelemente und Kindelemente. Die Elternelemente, auch "Container" genannt, agieren als Rahmen, in dem die Kindelemente, auch items genannt, enthalten sind. Für das aktuelle Beispiel erstelle ich zunächst einen Container mit den Eigenschaften "`display:flex`". Dadurch weiß der Browser, dass dieses Element mit Flexbox positioniert werden soll. [MDNa]

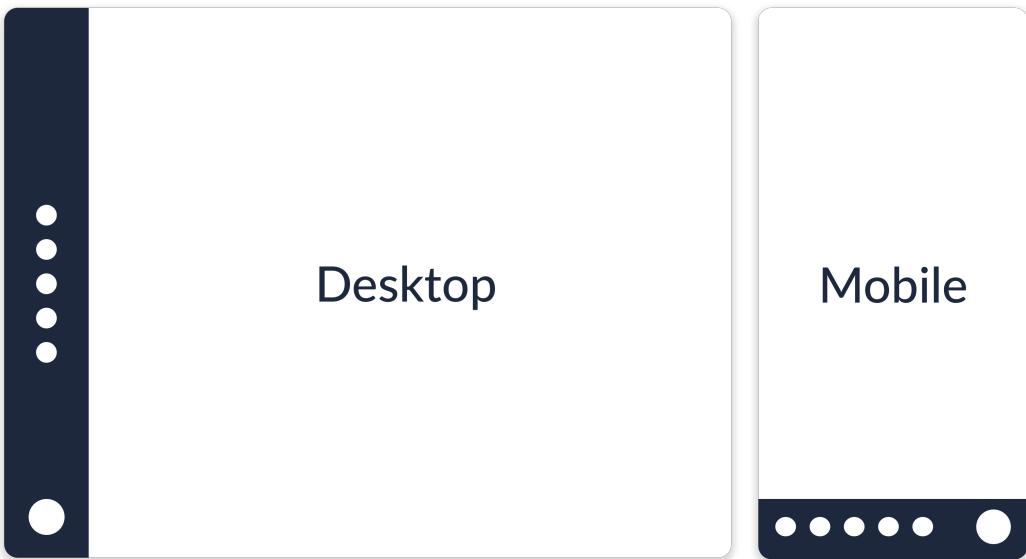


Abbildung 5.1: Flexbox Beispiel Funktionalität

Die Kindelemente dieses Flex-Containers werden standardmäßig auf der horizontalen X-Achse ausgerichtet. Wenn der Inhalt eines Elements mehr Platz einnimmt als vorhanden, dann läuft er über die Grenzen hinaus. Damit dieser Zustand nicht eintritt, wird der Überlauf des Flex-Containers mit "overflow: hidden" auf der X- und Y- Achse ausgeblendet. Die Navigation auf der Seite ist in folgendem Code-Block beschrieben. [W3Cb]

```
.parent {
  display: flex;
  overflow: hidden;
}
```

Dieses Element ist durch "order: 1" das erste Element in der Flexbox, allerdings muss es nicht das Erste im HTML-Markup sein. Mit Flexbox ist es möglich, die Reihenfolge unabhängig vom HTML-Code zu verändern. Damit die seitliche Navigationsleiste eine fixe Position behält, wird ein Hack angewendet. Denn durch das Setzen des Attributs "overflow-y: hidden" scrollt die Seite nur den restlichen Teil der Flexbox und nicht die Navigationsleiste. Weiters werden die Elemente innerhalb des Containers mit "justify-content: center" und "align-items: center" horizontal und vertikal zentriert und sind durch "flex-direction: column" entlang der Y-Achse positioniert. [W3Cb]

```
.side-nav {
  display: flex;
  order: 1;
  justify-content: center;
  align-items: center;
```

```
flex-direction: column;
}
```

Das Inhaltselement wird wegen des Attributes "order: 2" neben dem ersten Element auf der X-Achse positioniert. Weiters wird innerhalb dieses Elements auch mit Flexbox positioniert, wobei der Kontext ein anderer ist. Alle Kindelemente des Inhaltselements werden horizontal mit "justify-content: center" zentriert. In diesem Rahmen ist die Y-Achse die Hauptachse, das heißt, die Elemente werden vertikal verteilt. [W3Cb]

```
.content {
    overflow-y: hidden;
    display: flex;
    justify-content: center;
    flex-direction: column;
    order: 2;
}
```

Damit die Navigation auf mobilen Geräten am unteren Rand positioniert ist, benötigen wir eine Media Query, also eine Abfrage von bestimmten Eigenschaften. Mithilfe dieser können CSS-Stile anhand von verschiedenen Eigenschaften, wie z.B. Bildschirmauflösung oder Seitenverhältnis, manipuliert werden. Im untenstehenden Code-Block wird dies veranschaulicht. Indem wir die Y-Achse des Flexbox-Elternelements mit dem Attribut "flex-direction: column" zur Hauptachse machen, werden die beiden Kindelemente vertikal verteilt. Damit nun auch die Navigation unter dem Inhalt positioniert ist, ändern wir die Reihenfolge mit dem Attribut "order: 2". Dadurch ist dieses Element das zweite in dem Container, obwohl es im HTML-Markup das erste ist. Weiters müssen Höhe und Breite angepasst werden, damit das Element die gesamte Breite des Bildschirms einnimmt und nicht mehr die gesamte Höhe, wie bei der Desktopversion. [W3Cb, Sha]

```
@media (max-width: 576px) {
    .parent {
        flex-direction: column; //Die Y-Achse des Elternelements wird zur Hauptachse
    }

    .side-nav {
        order: 2; //Reihenfolge des Elements verändern
        width: 100vw; //gesamte Breite des Bildschirms ausnutzen
        height: 66px; //Die Höhe fix setzen
    }
}
```

#### 5.2.1.4 Weitere Möglichkeiten von Flexbox

“Das Flexible Box Layout Module” eignet sich äußerst gut für die Umsetzung von einzelnen Komponenten, die in sich geschlossen funktionieren müssen. Die Regeln, wie sich Elemente innerhalb eines Containers zu verhalten haben, werden einmal festgelegt. Das erleichtert das Erstellen von Bereichen, in denen dynamisch mit Javascript Inhalts-elemente hinzugefügt werden. Generell ist Flexbox die beste Technik um Teile einer Webseite zu schreiben, bei der die Anzahl der Elemente unbekannt ist, beziehungsweise die Anzahl der Kind-Elemente nicht statisch ist. [W3Cb]

Ein weiteres Einsatzgebiet von Flexbox ist die vertikale und horizontale Zentrierung von Inhaltselementen. Das Zentrieren ist in Flexbox eine knifflige Angelegenheit. Text kann mit dem Attribut "text-align" zentriert werden. Block-Elemente kann man mit margins oder Transformationen zentrieren. Aber das sind alles in gewisser Weise Hacks, da man Probleme und Bugs von CSS ausnutzt. Das ist natürlich nicht zukunftssicher, da diese Probleme in neuen Versionen behoben werden könnten. Mit Flexbox wird das vertikale und horizontale Zentrieren ein Kinderspiel. Innerhalb eines Flex-Containers können Elemente mit der Eigenschaft "justify-content: center" horizontal und "align-items: center" vertikal zentriert werden. Diese Lösung ist wesentlich eleganter und funktioniert quer über alle Browser. [W3Cb]

Flexbox ist eines der wichtigsten Module von CSS und mittlerweile zu einem fundamentalen Bestandteil moderner Frontend-Entwicklung geworden.

#### 5.2.2 CSS-Grid

CSS Grid Layout, offiziell “Grid Layout Module Level 1”, ist ein zweidimensionales Raster-System und wurde vom World Wide Web Consortium(W3C) entwickelt, um die Art, Webseiten zu schreiben, komplett zu verändern. Mittlerweile sind viele verschiedene Bildschirmauflösungen und Geräte im Umlauf. Die Anforderungen an Webseiten sind gestiegen und Webapplikationen werden zunehmend komplexer und umfangreicher. Derartige Seiten müssen plattformunabhängig auf jedem Gerät und Browser funktionieren. Diese Vielfalt wurde mit CSS immer schwerer umzusetzen. Zuerst wurden Tabellen für Layouts verwendet, danach stieg man auf “floats” um, anschließend folgten “inline-block” und “positioning”, doch all diese Methoden waren in Wirklichkeit nichts anderes als Hacks, denen wichtige Funktionen fehlten, wie zum Beispiel die vertikale Zentrierung. Wie bereits in Kapitel 5.1.1 Flexbox erwähnt, wurde mit Flexbox eine neue Methode eingeführt, um moderne, anpassungsfähige Webseiten umzusetzen. Allerdings ist diese Anwendung nur für simple, eindimensionale Layouts gedacht. Für komplexe zweidimensionale Layouts ist Flexbox umständlich und ineffizient. CSS-Grid hingegen ist das erste CSS-Modul, das einzig und allein für die Lösung jeglicher Layout-Probleme entwickelt wurde. Den Fokus haben die Entwickler auf grafische Benutzeroberflächen und Webapplikationen gelegt. [W3Cc]

### 5.2.2.1 Das Konzept

CSS-Grid kontrolliert die Größe und Position der Elemente und hält dabei vordefinierte Regeln ein. Im Gegensatz zu Flexbox positioniert das Elternelement die Kindelemente auf zwei Achsen anstatt auf einer. Dadurch kann die Position der Elemente genauer bestimmt werden. Dies trägt dazu bei, dass Grid besser für komplette Layouts von Webseiten geeignet ist. Weiters sind große Veränderungen, je nach Bildschirmauflösung, ohne Adaptierung des Markups möglich, da Objekten eine explizite Position zugewiesen werden kann. Man kann somit die gesamte Seite für mobile Geräte umbauen, ohne das HTML-Markup zu berühren. Alles passiert ausschließlich mit den Cascading Style Sheets(CSS). [MDNb]

### 5.2.2.2 Technische Spezifikation

Für einen CSS-Grid muss zuallererst ein Elternelement mit dem Attribut "display: grid" erstellt werden. Anschließend wird die Anzahl an Reihen mit "grid-template-rows" und die Anzahl an Spalten mit "grid-template-columns" festgelegt. Ähnlich zu Flexbox ist, dass die Reihenfolge der Elemente im HTML-Code egal ist. Mit CSS kann diese Anordnung beliebig geändert werden. Dadurch ist es sehr einfach, Elemente mithilfe von Media-Queries neu anzurufen. Deswegen können gesamte Layouts mit minimalem CSS vollkommen umgestaltet werden, um den verfügbaren Platz so effizient wie möglich auszunutzen. Dies macht Grid zu einem der mächtigsten Werkzeuge aller Zeiten von CSS. [W3Cc]

CSS Grid ist in den meisten Browsern nativ unterstützt und hat bereits eine globale Kompatibilität von über 87% erreicht. [Canb]

### 5.2.2.3 Erklärung anhand eines Beispiels

Um die Fähigkeiten von CSS-Grid etwas besser zu erläutern, habe ich ein Beispiel vorbereitet. Die Aufgabe ist es, eine dreispaltige Webapplikation für Twitter zu erstellen. Es gibt eine Spalte für Tweets, eine für Nachrichten und eine Suchfunktion. Das Layout sollte aussehen wie in Abbildung 5.2 dargestellt. [W3Cc]

```
<div class="twitter">
  <div class="tweets">Tweets</div>
  <div class="nachrichten">Nachrichten</div>
  <div class="suche">Suche</div>
</div>
```

Das HTML-Markup ist recht simpel, was auch einer der Vorteile von CSS-Grid ist. Es

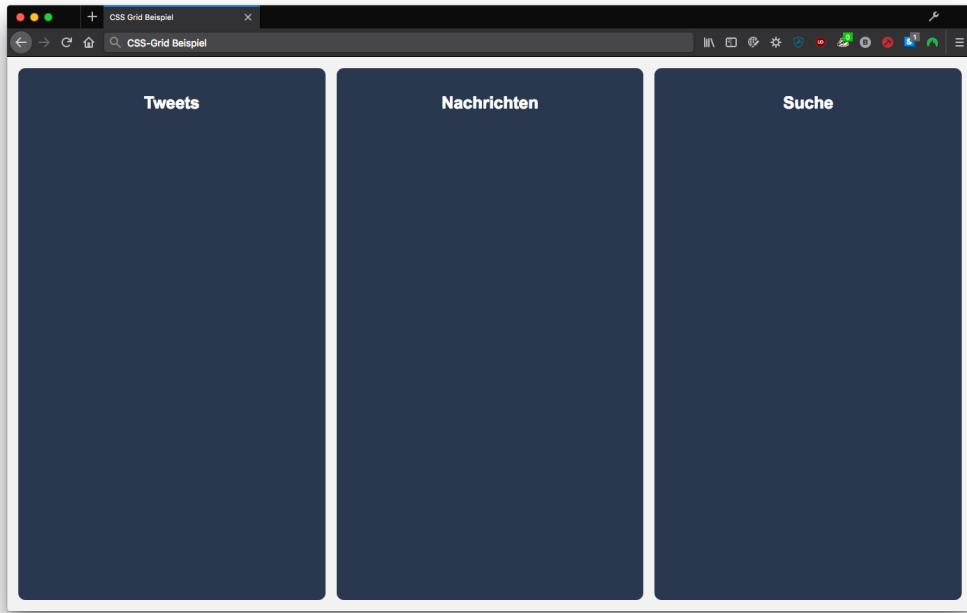


Abbildung 5.2: Ein dreispaltiges CSS-Grid-Layout mit einer Zeile

gibt nur einen Grid-Container. Dieser beinhaltet drei weitere <div> Elemente. Der spannende Teil ist allerdings im Stylesheet. [W3Cc]

```
.twitter {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-rows: 100vh;
}
```

Zuallererst wird das <div> Element mit der Klasse ".twitter" selektiert und mit dem Attribut "display: grid" als ein Grid-Container definiert. Die Grid-Eigenschaft legt außerdem einen neuen Kontext für das Grid-Layout fest. Dieser wird benötigt, um mit Grid ein Layout aufzubauen. Dieses Elternelement unterteilt den verfügbaren Raum mit dem Befehl "grid-template-columns: 1fr 1fr 1fr" auf drei gleich große Spalten. "1 fr" bedeutet, dass jedes Kindelement einen Teil des verfügbaren Platzes zugewiesen bekommt. Da drei Spalten definiert sind, ist jede Spalte 33.33% breit. Zuletzt müssen die Reihen definiert werden. Da unsere Spalten die gesamte Höhe des Browsers einnehmen sollen, muss mit "grid-template-rows" eine Zeile mit 100% der Höhe definiert werden. [W3Cc]

Dieses Layout kann mit Grid sehr einfach verändert werden, um es in zu einem einspaltiges zu verändern.

```
.twitter {
```

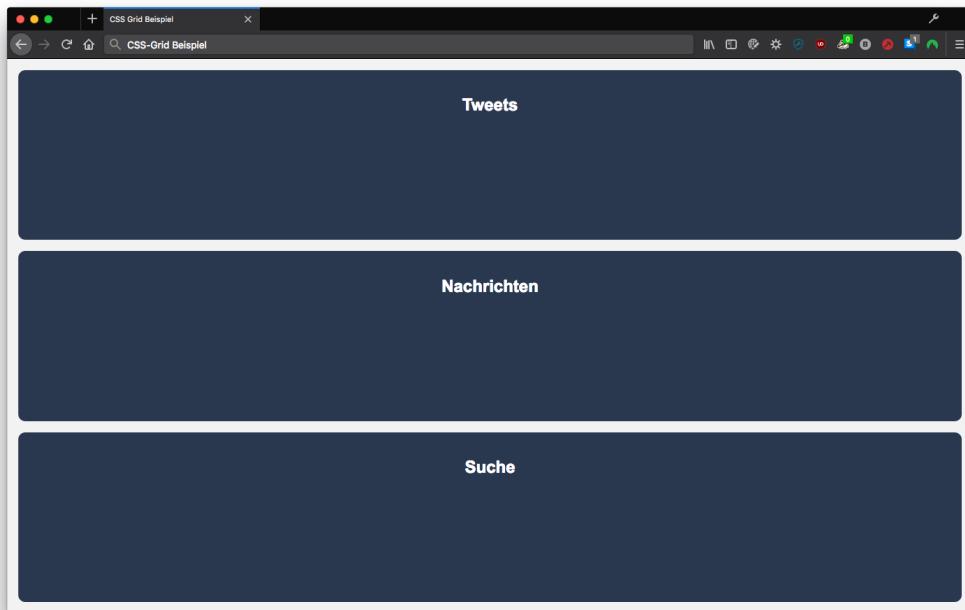


Abbildung 5.3: Ein einspaltiges CSS-Grid-Layout mit drei Zeilen

```

display: grid;
grid-template-columns: 1fr;
grid-template-rows: 1fr 1fr 1fr;
}

```

Man muss lediglich die Anzahl der Spalten auf eine ändern. Weiters fügt man zwei weitere Zeilen hinzu. Der vorhandene Platz soll gleichmäßig aufgeteilt werden(siehe Abbildung 5.3).

Damit das Layout auf mobilen Geräten optimal angezeigt wird, können Media-Queries verwendet werden.

```

@media screen and (max-width: 576px) {
    .twitter {
        grid-template-columns: 1fr 1fr;
        grid-template-rows: 50vh 50vh;
    }
}

```

Diese Media-Query wird aktiviert, wenn die Breite des Bildschirms unter 576 Pixel beträgt. Diese Ansicht wird somit nur auf Smartphones verwendet. [Sha, S.21]

#### 5.2.2.4 Weitere Möglichkeiten von CSS-Grid

Die größte Stärke von CSS Grid ist die explizite Positionierung auf zwei Achsen. Deswegen eignet sich dieses Modul hervorragend für komplexe Webseitenlayouts. Die Grundstruktur einer Seite sollte mit CSS-Grid aufgebaut sein. Solange die Anzahl der Elemente statisch ist, funktioniert das Raster-System sehr gut. Die Kindelemente können bei einer Änderung der Bildschirmauflösung vollkommen neu angeordnet werden. Innerhalb der einzelnen Elemente kann man Flexbox verwenden, da es sehr gut für die Umsetzung von Komponenten eingesetzt werden kann. [W3Cc]

Im Großen und Ganzen ist CSS-Grid eines der spannendsten und mächtigsten Module, das CSS derzeit zu bieten hat. Das W3C hat sich hohe Ziele für diese neue Positionierungsart gesetzt und ich glaube, dass CSS Grid diese übertreffen kann.

### 5.2.3 Gegenüberstellung von CSS-Grid und Flexbox

Vor ein paar Jahren wurde Flexbox eingeführt. Es ist speziell für anpassungsfähige Webseiten entwickelt worden. Flexbox macht das Ausrichten von Elementen und deren Inhalt einfacher, sodass flüssige, flexible und dynamische Seiten erstellt werden können. Diese funktionieren mit wenig CSS in einem breiten Katalog von Geräten. Dank Flexbox sind Webseiten, die sich an verschiedene Geräte anpassen müssen, machbarer und effizienter geworden. Ohne diese Technik würden Webseiten auch heute noch nicht perfekt in allen Bildschirmauflösungen dargestellt werden können.

Allerdings ist ein neuer Konkurrent am Spielfeld erschienen - CSS-Grid. Diese Technik hat einige ähnliche Eigenschaften wie Flexbox. Obwohl so gut wie jedes Layout sowohl mit Flexbox als auch CSS Grid umsetzbar ist, haben beide Techniken ihre Spezialgebiete. Deswegen stellt sich natürlich die Frage, welche Positionierungstechnik gerade besser ist, beziehungsweise wann welche zum Einsatz kommen sollte. [Bor]

#### 5.2.3.1 Eine versus zwei Dimensionen

Der wohl gravierendste Unterschied zwischen Flexbox und Grid ist die Anzahl an Richtungen, die gleichzeitig von der “Rendering Engine” des Browsers beachtet werden können (siehe Abbildung 5.4). Beim “Flexible Box Layout” kann nur eine Richtung beachtet werden, entweder die X-Achse oder die Y-Achse. Dadurch ist es sehr gut geeignet für Anwendungen, in denen Elemente nur auf einer Achse positioniert werden sollen. Ein häufiges Beispiel hierfür wäre eine Navigationsleiste, solange diese nur in einer Richtung ausgerichtet wird. Es ist egal, ob am oberen Rand - entlang der X-Achse oder am linken Rand - entlang der Y-Achse. Flexbox macht diese Leiste bei Veränderungen flexibler, da die Elemente beliebig auf der Achse bewegt werden

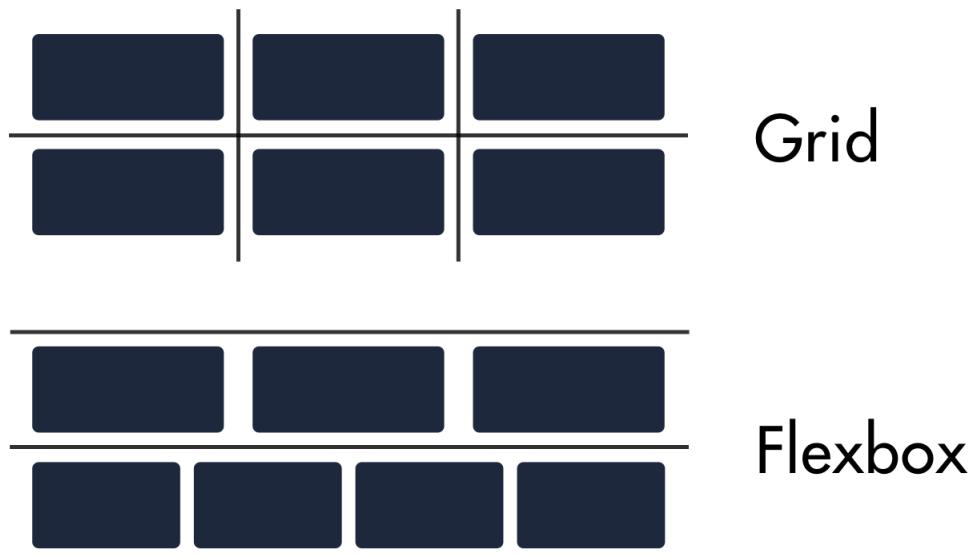


Abbildung 5.4: grundlegender Unterschied zwischen CSS-Grid und Flexbox

können. Außerdem ist weniger Code für die Umsetzung notwendig, somit ist dieses Element simpler und leichter zu warten. Wie bereits in Kapitel 5.1.1 Flexbox erwähnt, ist Flexbox ausgelegt für Container, bei denen die Anzahl an Kindelementen unbekannt ist. [Sha, S.65, S.66]

Bei CSS-Grid wird hingegen eine fixe Anzahl von Reihen und Spalten definiert. Die Maße und die Anordnung dieser können variieren, aber die Anzahl sollte beibehalten werden. [W3Cb]

### 5.2.3.2 Dynamisch versus statisch

Da bei Flexbox im CSS für den Browser genau definiert wird, wie Elemente innerhalb eines Containers anzurichten sind, können beliebig viele Elemente dynamisch hinzugefügt werden. Diese Eigenschaft habe ich bei der Umsetzung der Benutzeroberfläche mehrmals verwendet. Erstens befinden sich die Module für die Steuerung der einzelnen Geräte in einem Flex-Container. Da diese per Klick dynamisch hinzugefügt werden können, hat sich hier Flexbox angeboten. Beim Betätigen eines Knopfes wird das entsprechende Modul an den `<div>` angehängt. Wenn die Gesamtbreite der Elemente die Breite des Bildschirms übersteigt, wird eine Scrollbar auf der X-Achse eingeblendet. Weiters habe ich Flexbox bei der Darstellung der vordefinierten Presets verwendet. Obwohl diese Elemente in Form eines Rasters angeordnet sind, war CSS-Grid nicht geeignet. Denn bei diesem werden fixe Reihen und Spalten festgelegt. Allerdings hat mir das Probleme bereitet, da die Anzahl der Presets zum Zeitpunkt der Entwicklung nicht bekannt war. /cite{flexbox\_vs\_grid}

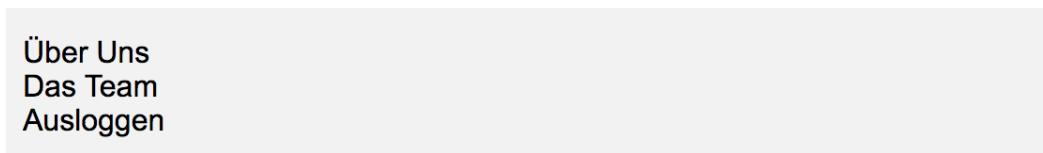


Abbildung 5.5: derzeitige Darstellung ohne Flexbox

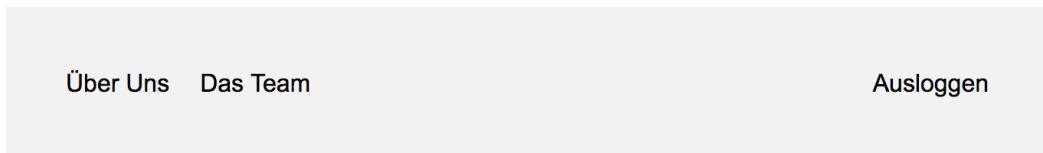


Abbildung 5.6: Layout mit "display: flex"

### 5.2.3.3 Erklärung der Unterschiede anhand eines Beispiels

Ein weiterer Unterschied zwischen Flexbox und CSS-Grid ist, dass die Basis von Flexbox der Inhalt der Seite ist, aber bei Grid die Basis das Layout ist. Um dies genauer zu erläutern, habe ich ein Beispiel vorbereitet. [fle]

```
<header>
  <div>Über Uns</div>
  <div>Das Team</div>
  <div>Ausloggen</div>
</header>
```

Bevor das <header> Element durch das Attribut "display: flex" zu einem Flex-Container wird, sieht das derzeitige Markup aus wie in Abbildung 5.5. [W3C b]

Damit der Ausloggen-Knopf auf der rechten Seite des Bildschirms positioniert ist, wird dieser mit einem CSS-Selektor ausgewählt und durch das Setzen von "margin-left: auto" an das Ende des Containers geschoben (siehe Abbildung 5.6).

Bei diesem Beispiel ist zu erwähnen, dass der Browser selber entscheidet, wie die Elemente zu positionieren sind. Es wurde lediglich das Kommando erteilt, Flexbox als Positionierungstechnik zu verwenden. Das ist einer der Kernunterschiede zwischen Flexbox und CSS-Grid. Obwohl Grid nicht für eindimensionale Layouts wie einen Header gedacht ist, werde ich dieses Element nun mit Grid nachbauen. Der HTML-Code kann beibehalten werden. Im CSS wären dafür mehrere Änderungen notwendig.

```
header {  
    display: grid;  
    grid-template-columns: repeat(10, 1fr);  
}  
div:<math>:nth-child(1)</math> {
```

```

    grid-column: 1 / 2;
}
div:nth-child(2) {
    grid-column: 2 / 3;
}
div:nth-child(3) {
    grid-column: 10 / 11;
}

```

Durch das Setzen von "display: grid" wird das <header> Element zu einem CSS-Grid. Die Anzahl der Spalten wird mit dem Attribut "grid-template-columns" definiert. Der Hauptunterschied mit diesem Ansatz ist, dass die Spalten, also das Layout, zuerst definiert werden müssen. Man benötigt Spalten um Inhalt dort zu platzieren. In diesem Fall erstellt man zehn Spalten zu je einem fr, also einem Bruchteil der Seite. Diese Einheit signalisiert dem Browser, dass er den verfügbaren Platz auf alle Spalten gleich aufteilen soll. Nun weisen wir jedem einzelnen <div> Element eine Spalte zu. Damit der Ausloggen-Knopf am rechten Rand positioniert ist, wird er explizit in die zehnte Spalte mit "grid-column: 10 / 11" geschoben. Dieser Ansatz zwingt uns die Anzahl der Spalten festzulegen. Sofern der Grid nicht verändert wird, hat er zehn Spalten. Eine eindeutige Begrenzung, die in Flexbox nicht vorhanden wäre. [W3Cb]

#### 5.2.3.4 Kombination von CSS-Grid mit Flexbox

Wie bereits erwähnt, haben beide dieser Technologien ihre Vor- und Nachteile. Demnach gibt es verschiedene Anwendungsfälle für sie. Allerdings sollten diese Module zusammen verwendet werden. Mithilfe von CSS-Grid kann ein Raster-System für das Layout der gesamten Webseite erstellt werden. Innerhalb der einzelnen Inhaltselemente hat man mit Flexbox mehr Möglichkeiten, wie zum Beispiel das dynamische Hinzufügen von Elementen. Um zu zeigen, wie so ein Anwendungsfall aussehen könnte, habe ich die beiden obigen Beispiele verbunden. [W3Cc]

```

<div class="container">
    <header>Kopfzeile</header>
    <aside>Menü</aside>
    <main>Inhalt</main>
    <footer>Fußzeile</footer>
</div>

```

Das HTML-Markup ist weiterhin sehr simpel. Die Kopfzeile soll die gesamte Breite der Seite einnehmen und am oberen Rand platziert sein. Das Menü ist schmal und am linken Rand der Seite. Der Inhalt soll den restlichen horizontalen Platz einnehmen. Am unteren Rand soll es noch eine Fußzeile geben. Das Layout ist in Abbildung 5.7 zu

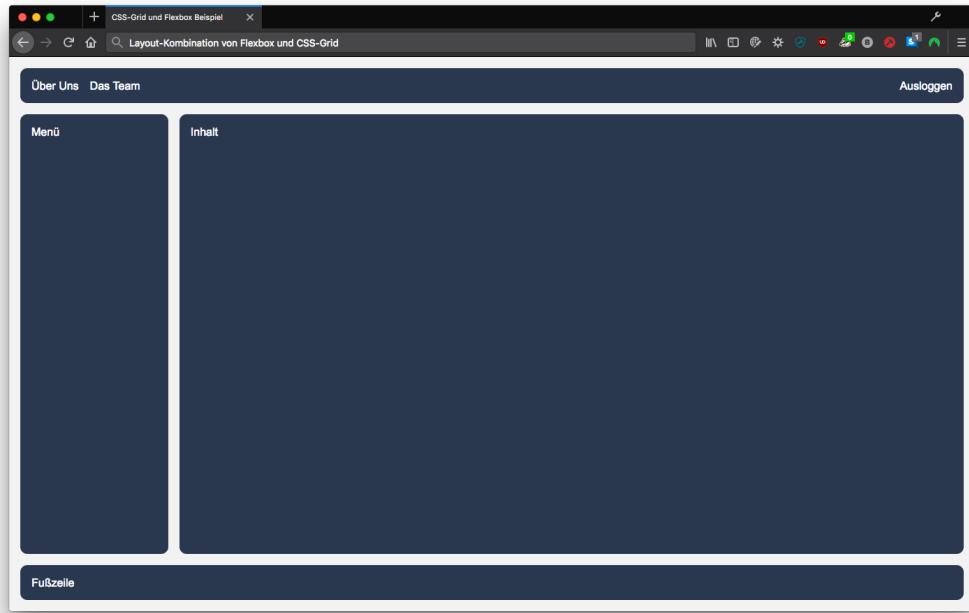


Abbildung 5.7: Layout-Kombination von Flexbox und CSS-Grid

sehen.

```
.container {
    display: grid;
    grid-template-columns: repeat(12, 1fr);
    grid-template-rows: 50px 350px 50px;
}
header {
    grid-column: span 12;
    display: flex;
}
header > div:nth-child(3) {
    margin-left: auto;
}
aside {
    grid-column: span 2;
}
main {
    grid-column: span 10;
}
footer {
    grid-column: span 12;
}
```

In diesem CSS-Code passiert eine Menge. Das Elternelement wird zu einem Grid-Container gemacht. Weiters werden zwölf Spalten mit einem gleichgroßen Anteil der verfügbaren Platzes. Außerdem gibt es drei Reihen mit unterschiedlichen Höhen. Die Kopfzeile streckt sich wegen der Eigenschaft "grid-column: span 12" über alle zwölf Spalten. Wegen des Attributes "display: flex" ist der Header ein Flex-Container. Dieser ist aufgebaut wie im obigen Beispiel zu Flexbox erläutert. Das Menü nimmt nur zwei Spalten ein wegen "grid-column: span 2". Dem Inhalt wird der restlichen Platz zugewiesen. Die Fußzeile ist aufgebaut wie die Kopfzeile und ist ebenfalls so breit wie der Bildschirm. [W3Cc]

### 5.2.3.5 Flexbox oder CSS-Grid?

Welches dieser beiden Module besser ist, ist die falsche Frage. Sie sollte eher lauten, wann verwende ich Flexbox und wann CSS-Grid. Am besten werden sie zusammen eingesetzt. Man kann die Vorteile von Flexbox, wie zum Beispiel Dynamik oder Flexibilität, mit den Vorteilen von CSS-Grid, wie zum Beispiel explizite Positionierung und Layout-Unabhängigkeit, miteinander verbinden.

Da auf dem Dashboard von Mediatrix verschiedenste Daten dynamisch geladen werden, habe ich mich dafür entschieden, die komplette Seite mit Flexbox aufzubauen. Ein weiterer Grund für diese Entscheidung war die Browser-Kompatibilität, da Flexbox bereits zu über 95% unterstützt wird und CSS-Grid erst 87% erreicht hat. Außerdem habe ich Flexbox im Unterricht gelernt und hatte deswegen bereits ein grundsätzliches Verständnis der Technologie. [? Canb]

### 5.2.4 Normalize.css

Jeder Browser wendet standardmäßig bestimmte Stile auf jede Seite an. Dadurch werden Elemente wie Formulare oder Scrollbars auf jedem Browser anders dargestellt. Damit das Verhalten und die Darstellung von Webseiten konsistenter wird, hat Nicolas Gallagher Normalize.css entwickelt. Diese Datei überschreibt einige Stile auf allen Browsern, um die voreingestellten Stile gewissermaßen zurückzusetzen.

Nicolas Gallagher zufolge ist die Aufgabe von Normalize.css:

- sinnvolle Browsereigenschaften zu bewahren
- Stile für eine breite Auswahl an HTML Elementen zurückzusetzen
- Bekannte Fehler und Browser-Inkonsistenzen zu beheben
- die Benutzerfreundlichkeit zu verbessern
- die Änderungen simpel mithilfe von Kommentaren zu erklären

[Nic]

#### 5.2.4.1 Technische Spezifikation

Im Gegensatz zu anderen CSS Reset Dateien überschreibt Normalize.css nicht alle Standardstile der Browser, da sonst alle typographischen Elemente wie Überschriften und Listen neu definiert werden müssten. Allerdings werden Elemente je nach Browser eine andere Erscheinung haben, deshalb versucht Normalize.css diese so konsistent wie möglich zu gestalten.

Auch bekannte Fehler von Browsern werden behoben. Diese beinhalten unter anderem die Darstellung von SVGs im Internet Explorer, Positionierungs-Eigenschaften von HTML5-Elementen, Schriftgröße bei bereits formatierten Texten und viele weitere Probleme im Bezug auf Formulare.

Außerdem ist Normalize.css modular. Somit können nur die benötigten Module importiert werden, um die Ladegeschwindigkeit weiter zu beschleunigen. [Nic]

Zusammenfassend kann man sagen, dass durch die Verwendung von Normalize.css das Arbeiten mit verschiedenen Browsern und Versionen angenehmer wird. Man kann auf eine vereinheitlichte Basis aufbauen und garantiert eine konsistente Darstellung quer über alle gängigen Browser.

## 5.3 SASS

### 5.3.1 Was ist ein CSS-Präprozessor?

Ein Browser verwendet CSS um DOM Elemente visuell zu bearbeiten und zu rendern. CSS selbst hat seine eigene Auswahl an Funktionen, welche manchmal allerdings nicht ausreichen um saubere und wiederverwendbare Regeln zu schreiben. Aufgrund dieser Einschränkungen ist das Konzept eines CSS Präprozessors entstanden. Diese bieten einen erweiterten Funktionsumfang, der die Grundfunktionalität erweitert. Anschließend wird die Datei zu herkömmlichen CSS kompiliert, da der Browser nur reines CSS interpretieren kann. [Sas]

Es gibt verschiedene CSS-Präprozessoren wie SASS, LESS oder Stylus. All diese bieten verschiedene Funktionen an, ich persönlich habe mich aus mehreren Gründen für Verwendung von SASS entschieden.

Um die Vorteile von CSS-Präprozessoren zu verstehen, zeige ich zuerst ein paar der Mängel von reinem CSS auf.

- keine Möglichkeit, allgemeine Stilregeln wiederzuverwenden

- Dateien werden schnell sehr groß
- ordentliches Arbeiten erfordert großen Zeitaufwand

Präprozessoren haben für all diese Probleme eine Lösung parat. [Sas]

### 5.3.1.1 Partielle CSS-Dateien

Aufgrund der ständig komplexer werdenden Frontend-Entwicklung werden CSS-Dateien immer größer und enthalten oft tausende Zeilen an Code. Umso länger diese Dateien sind, desto unhandlicher und verwirrender werden sie. Dafür bietet SASS bereits eine Lösung. CSS-Dateien können auf mehrere partielle Dateien aufgeteilt werden. Dadurch kann man seine Stylesheets besser organisieren und modularisieren. Es ist wesentlich einfacher, mehrere kleine Dateien zu schreiben, zu warten und zu erweitern, als eine große Datei mit tausenden Zeilen. Mithilfe der erweiterten "@import" Funktion des Präprozessors können die einzelnen Dateien beim Speichern zu einer zusammengefasst werden. [Sas]

Die "@import"-Regel ist schon sehr lange ein Teil von CSS. Allerdings ist sie nicht besonders beliebt, da jeder Import eine eigene HTTP-Anfrage ausführt, was zu einer langsamen Webseite führt. Was passiert, wenn man diese Funktion mit SASS verwendet? Hoffentlich haben Sie nie damit aufgehört, über den Namen "Präprozessor" nachzudenken. [Sas]

"Ein Präprozessor ist ein Computerprogramm, das Eingabedaten vorbereitet und zur weiteren Bearbeitung an ein anderes Programm weitergibt." — Wikipedia [? ]

Wenn man dieses Konzept nun auf die "@import"-Regel anwendet, fällt einem auf, dass der Import von SASS damit erledigt wird. Die CSS- und SASS-Dateien werden zusammengefasst und am Ende bleibt nur eine Datei über. Der Browser des Benutzers muss nur eine Anfrage ausführen und bloß eine Datei hinunterladen. Das Projekt könnte aus einer Vielzahl an CSS-Dateien zusammengesetzt sein. [Sas]

### 5.3.1.2 Variablen in SASS

Die meisten Artikel über SASS beginnen damit, das System wegen der Implementation von Variablen zu loben. Der häufigste Einsatzzweck ist die wiederverwendbare Farbpalette. Es passiert immer wieder, dass Farben mehrmals definiert werden, aber man trifft meistens nicht den selben Farbton. Das resultiert in vielen verschiedenen Abstufungen einer Farbe. Aber in SASS können Variablen mit fast jeder Einheit definiert werden. Ein großer Vorteil dieser Funktion ist, dass alle Farben an einem Ort gesammelt sind. Das erleichtert zum Beispiel die Veränderung der Palette bei einer Veränderung des

Corporate-Designs. Außerdem sparen Variablen Zeit, da man nicht die gesamte Datei nach bestimmten Farben oder Werten durchsuchen muss. [Sas]

### 5.3.1.3 Sauberer Code dank Verschachtelung der Regeln

Verschachtelung ist wahrscheinlich die zweitbekannteste Funktion von SASS. Durch sie kann ein Stylesheet besser organisiert werden. Dadurch wirkt es ordentlicher, übersichtlicher und ist leichter zu lesen. Weiters wird die Hierarchie der Regeln auf einen Blick ersichtlich und die Abhängigkeiten der Elemente sind eindeutig. [Sas]

### 5.3.1.4 Wiederholungen vermeiden

Beim Schreiben von CSS ist das mehrfache Definieren von Elementen fast unumgänglich. Allerdings bietet SASS auch hier Abhilfe. "mixins" und "extends" sind zwei mächtige Funktionen. Die Möglichkeiten dieser scheinen fast endlos zu sein. Mit "mixins" können parametrisierte CSS-Funktionen erstellt werden, die im gesamten Dokument wiederverwendbar sind. Hier ein kurzes Beispiel, um "mixins" genauer zu beleuchten. [Sas]

```
@mixin box-shadow($top, $left, $blur, $color, $inset: false) {  
    @if $inset {  
        -webkit-box-shadow: inset $top $left $blur $color;  
        -moz-box-shadow: inset $top $left $blur $color;  
        box-shadow: inset $top $left $blur $color;  
    } @else {  
        -webkit-box-shadow: $top $left $blur $color;  
        -moz-box-shadow: $top $left $blur $color;  
        box-shadow: $top $left $blur $color;  
    }  
}
```

Man stelle sich den Schatten eines Elements vor. So ein Schatten hat mehrere Eigenschaften, die bei jeder Verwendung gesetzt werden müssen. Hier bietet sich ein "mixin" an. Eine Funktion wird zuerst mit den einzelnen Eigenschaften als Parameter definiert. Diese kann nun im gesamten Dokument eingesetzt und angepasst werden. [Sas]

"extends" sind sehr ähnlich, da man mit ihnen Eigenschaften mit anderen Selektoren teilen kann. Anstatt jedoch mehrere Deklarationen auszugeben, gibt man eine Liste von Klassen aus, ohne die Eigenschaften zu wiederholen. Dadurch werden Code-Wiederholungen in der Ausgabedatei vermieden. [Sas]

### 5.3.1.5 Funktionen, um Farben zu modifizieren

SASS erweitert CSS mit zusätzlichen Funktionen. Wenn man Funktionen hört, denkt man ans Programmieren. Die in SASS implementierten Funktionen bieten eine einfache Möglichkeit, um Farben abzustufen oder zu modifizieren. [Sas]

```
lighten($farbe, $faktor)
darken($farbe, $faktor)
grayscale($farbe)
complement($farbe, $alpha)
```

In all diesen Funktionen müssen Parameter übergeben werden. Alle benötigen eine Farbe, dies hexadezimal, rgb oder hsl sein kann. Weiters benötigt man einen Faktor, um die Farbe zu verändern. Mit der "lighten()" Funktion können Farben um einen Faktor aufgehellt werden. Die "darken()" Funktion macht genau das Gegenteil. Mithilfe von "grayscale()" können Farben in Graustufen konvertiert werden. Eine weitere hilfreiche Funktion ist "complement()". Diese gibt die Komplementärfarbe zurück. [Sas]

### 5.3.1.6 Nachteile von CSS-Präprozessoren

Wie beim Lernen jeder anderen Technologie werden Designer auch bei SASS mit einigen Hürden konfrontiert.

- Für Designer ist der Terminal nicht immer ein beliebtes Werkzeug. Um Präprozessoren zu verwenden, wird jedoch ein gewisses Grundwissen vorausgesetzt.
- Da SASS einige Konzepte aus der Programmierung ausborgt, kommt man um das Erlernen dieser nicht herum.
- Die Syntax kann für einen Anfänger etwas verwirrend erscheinen, da sie stark von der normalen CSS-Syntax abweicht.
- Wenn man nicht vorsichtig mit der Gruppierung von Selektoren umgeht, können diese schnell komplex und tief werden, was die Wartbarkeit verschlechtert.

[Sas]

### 5.3.1.7 Verwenden oder nicht verwenden?

Wenn man sich noch nicht sicher ist, ob man einen CSS-Präprozessor verwenden sollte, habe ich ein paar Tipps parat. Der beste Weg, das herauszufinden, ist, es einfach zu verwenden. Man muss die Argumente dafür und dagegen im Hinblick auf die eigenen Ansprüche abwägen und sich seine eigene Meinung bilden. [Sas]

Der Umstieg kann schwierig und zeitraubend werden, aber ich glaube, dass er es wert ist.

## 5.4 CSS-Variablen

### 5.4.1 Was sind CSS-Variablen?

Wenn man eine Webseite entwickelt ist es üblich, eine einheitliche Farbpalette zu erstellen, um das Erscheinungsbild der Seite konsistent zu halten. Unglücklicherweise ist das andauernde Wiederholen der unterschiedlichen Farben quer durch die CSS-Datei nicht nur anstrengend, sondern auch sehr fehleranfällig. Falls eine dieser Farben verändert werden muss, könnte man versuchen mithilfe von Suchen-und-Ersetzen alle Werte zu aktualisieren. Jedoch kann dies bei großen Dateien schnell zu Fehlern führen. [W3Ca]

Wie bereits in Kapitel 5.3 erwähnt, hat man mithilfe von CSS-Präprozessoren die Möglichkeit, Variablen zu definieren. Obwohl diese Funktion sehr praktisch für Frontend-Entwickler ist, hat die Implementation von SASS einen großen Nachteil. Sie ist nämlich statisch, d.h. beim Kompilieren der SASS-Datei werden die Werte der Variablen ausgelesen und an die Stellen, wo sie verwendet werden, eingesetzt. Damit verliert man die Fähigkeit, die Variablen dynamisch auf der Webseite zu ändern. Mit den neuen "CSS Custom Properties" ist es jedoch möglich, globale Variablen für den Browser zu definieren, die dynamisch auf der Seite geändert werden können. Das ermöglicht Funktionen wie zum Beispiel das Ändern der Farbpalette oder der Zeilenhöhe. [W3Ca]

#### 5.4.1.1 Technische Spezifikation

Variablen fügen CSS zwei neue Funktionen zu.

- Einen Behälter für einen Wert zu definieren, der im Quelltext durch einen Namen bezeichnet wird.
- Die "var()" funktion, die es dem Entwickler erlaubt, die Werte in jeglichen Eigenschaften zu verwenden.

[W3Ca]

#### 5.4.1.2 Erklärung anhand eines Beispiels

```
:root {
    --Primär-Farbe: #ed4e53;
}

h1 {
    color: var(--Primär-Farbe);
}
```

In diesem CSS-Block haben wir eine Variable mit dem Namen "Primär-Farbe" und dem Hexwert "#ED4E53" erstellt. Wichtig ist hier, dass diese Farbe im "root" Kontext definiert wurde. Deswegen ist sie im ganzen Dokument verfügbar. Weiters ist anzumerken, dass der Name einer Variable zwei Bindestrichen am Anfang haben muss. [W3Ca]

Mithilfe der "var()" Funktion wird der Wert ausgelesen und eingesetzt. Im Beispiel wird die Schriftfarbe einer Überschrift dementsprechend gesetzt. [W3Ca]

Bei Mediatrix habe ich CSS-Variablen verwendet, um verschiedene Designs des Dashboards anzubieten. Der Benutzer kann zwischen verschiedenen Farbpaletten wechseln, je nach dem, welche ihm besser gefällt. Ohne diese Spezifikation, wäre die Umsetzung dieser Funktion sehr umständlich gewesen.

Auf die CSS-Variablen kann im Javascript zugegriffen werden. Sie können ausgelesen und gesetzt werden. Der Benutzer kann aus mehreren Farbpaletten auswählen. Beim Klick auf eine der Paletten werden alle Variablen der Seite überschrieben.

```
let root = document.querySelector(":root");

root.style.setProperty("--Primär-Farbe", "#000000");
```

Man muss zuerst alle Farben des Root Elements laden. Anschließend können alle Variablen neu gesetzt werden.

CSS-Variablen sind bereits in den aktuellen Versionen der gängigen Browsern verfügbar. [Cana]

## 5.5 jQuery

### 5.5.1 Was ist jQuery?

jQuery ist eine Bibliothek, die von John Resig erfunden wurde, um das Programmieren mit Javascript zu vereinfachen. Diese vereinfacht Frontend Entwicklung durch das Simplifizieren von Javascript-Funktionen, Animationen, AJAX(Asynchronous Javascript and XML), Dom-Manipulation, sowie das Arbeiten mit Events. Dadurch spart man sich Zeit und muss wenige Code-Zeilen schreiben. [jqu]

#### 5.5.1.1 Mehr schreiben in weniger Zeit

Zeit ist Geld. Mit jQuery spart man sich hier und da 30 Sekunden an Arbeit. Diese zeitliche Ersparnis summiert sich zu vielen Stunden an gesparter Zeit auf. Man kann seine Arbeitszeit effizienter und produktiver nutzen, da diese Bibliothek Teile der Arbeit übernimmt und somit für uns arbeitet. Selbst wenn man nicht für einen Kunden arbeitet, lohnt es sich etwas Zeit zu sparen, die man besser nutzen könnte, als Funktionen zu programmieren, die bereits jemand anderer für uns ausformuliert hat. [jqu]

#### 5.5.1.2 Browser Kompatibilität

Ein weiterer Vorteil von jQuery ist die Browser-Unabhängigkeit. Da eine Bibliothek eine Datei ist, die beim Öffnen der Webseite geladen wird, sind alle Funktionen automatisch in allen Browsern verfügbar. Dadurch kann man sicherstellen, dass der Code quer über alle Browser funktioniert. Es gibt einige Javascript-Funktionen, die nativ, also unveränderbar, für manche Browser sind. Das umgeht man mit jQuery. [jqu]

#### 5.5.1.3 Simplifizierung von Javascript-Funktionen

Die Skriptsprache Javascript ist sehr umfangreich und enthält eine Menge an komplizierten Funktionen. Die Entwickler von jQuery haben einige dieser vereinfacht. So wird zum Beispiel eine gut geschriebene und browserunabhängige Implementierung von HTTP-Anfragen(Hypertext Transfer Protokoll) in Form von AJAX-Anfragen bereitgestellt. Schon alle diese Funktion ist ein Grund, um jQuery zu importieren. [jqu]

#### 5.5.1.4 Gewappnet für die Zukunft

Mit jQuery ist man zukunftssicher unterwegs. Funktionen wie ".indexOf()" oder ".bind()" sind zwar bereits nativ in Javascript implementiert, allerdings noch nicht in allen Browsern verfügbar. Jedoch können die dementsprechenden jQuery-Alternativen bereits heute in allen Browsern verwendet werden. [jqu]

#### 5.5.1.5 Potential für Zeitersparnis

Um das Potential von jQuery näher zu erläutern, habe ich ein gängiges Beispiel vorbereitet. Im untenstehenden Code-Block ist eine "Fade-In-Animation" mit Javascript umgesetzt.

```
function fadeIn() {
    var element = document.getElementById("element");
    var transparenz = parseFloat(element.style.opacity);
    var timer = setInterval(function() {
        if (transparenz >= 1.0) clearInterval(timer);
        transparenz += 0.1;
        element.style.opacity = transparenz;
    }, 50);
}
fadeIn(element);
```

Diese Vorgehensweise ist sehr aufwendig. Zuerst muss man das Element selektieren. Dann benötigt man einen Timer, der die Transparenz langsam anhebt, bis das Element vollkommen sichtbar ist. Nun im Gegensatz dazu die Umsetzung mithilfe von jQuery.

```
$(element).fadeIn();
```

In jQuery sind bereits herkömmliche Animationen implementiert. Dadurch können komplizierte Code-Segmente wie im obigen Beispiel oft auf eine Zeile gekürzt werden.

#### 5.5.1.6 Nachteile

Allerdings hat alles auch Nachteile, bei jQuery ist es nicht anders. Der erste negative Aspekt dieser Bibliothek ist der leichte Einstieg und die niedrige Lernkurve. Dadurch hat sich über Jahre eine gigantische Gemeinschaft aufgebaut. Darunter hat zum Beispiel die Qualität von Open Source Plugins im Internet stark gelitten. Der zweite Nachteil ist, dass es einfach ist, ineffizienten Code zu schreiben, wenn das Javascript Wissen begrenzt ist. Die Performanz von jQuery kann verbessert werden, wenn man sich besser

in Javascript auskennt. Manchmal ist es schneller, eine normale Schleife zu verwenden, als auf die jQuery Alternative zurückzugreifen. Das letzte und wohl größte Problem von jQuery ist der Overhead. Man muss eine 97kB große Datei importieren, verwendet aber meistens nur eine kleine Auswahl der verfügbaren Funktionen. [jqu]

Zu guter Letzt kann ich sagen, dass so gut wie jedes größere Projekt durch die Einbindung von jQuery profitieren würde. Nichtsdestotrotz sollte man zuerst reines Javascript lernen, um eine grundlegende Basis aufzubauen. Denn jQuery ist keine Garantie für sauberen Code, denn um das Aneignen von Programmierkenntnissen führt kein Weg vorbei.

## 5.6 Design

### 5.6.1 Fokus auf technisch unversierte Benutzer

Schon am Anfang des Projekts war die Zielgruppe klar. Wir wollten, dass jeder, egal ob Techniker oder nicht, die AV-Geräte im LIZ verwenden kann. Deswegen war es uns wichtig, dass die Benutzeroberfläche schlicht und intuitiv ist. Es werden nur die Informationen angezeigt, die während dem Halten einer Präsentation relevant sind, angezeigt. Weiters gibt es einen Modus, in dem aus vordefinierten Presets ausgewählt werden kann. Somit muss der Benutzer nur noch einen Knopfdruck tätigen und alle Geräte sind einsatzbereit. Dadurch benötigt der Benutzer keine technischen Vorkenntnisse um die AV-Installation im LIZ nutzen zu können.

Bild des Präsentationsmodus -> kommt am Mittwoch

Wenn man im Präsentationsmodus ist, stehen einem nur Presets zur Verfügung. Eine Box repräsentiert ein Preset. Es wird die Anzahl der Scheinwerfer, die Anzahl der Mikrofone, in welchem Modus sich der AV-Receiver befindet und der Beamer-Eingang angezeigt. So sieht man auf einen Blick, welche Parameter mit den verschiedenen Presets gesetzt werden.

Bild der Oberfläche -> kommt am Mittwoch

Wenn man jedoch etwas mehr einstellen möchte, kann man in den erweiterten Modus wechseln. In diesem können die verschiedenen Module zur Steuerung der Geräte eingeblendet werden. Weiters hat man die Möglichkeit die derzeitige Konfiguration als Preset abzuspeichern. Außerdem gibt es verschiedene Farbpaletten für die Benutzeroberfläche.

# **6 Zusätzliches**

## **6.1 Bedienungsanleitung**

## **6.2 Beleuchtungskonzept Konferenzsaal**



# A Anhang 1

was auch immer: technische Dokumentationen etc.

Zusätzlich sollte es geben:

- Abkürzungsverzeichnis
- Quellenverzeichnis (hier: Bibtex im Stil plaindin)



# Literaturverzeichnis

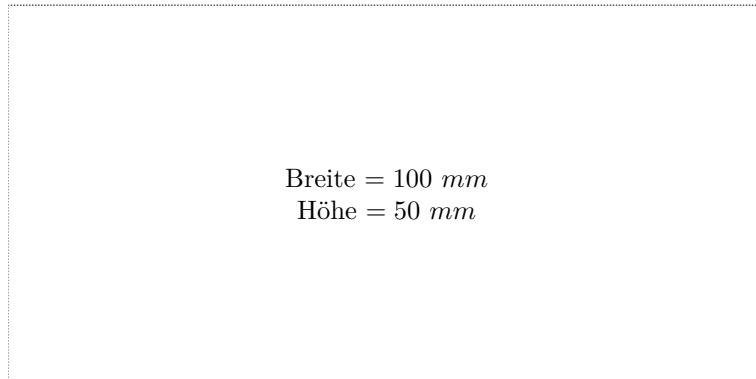
- [Bor] Per Harald Borgen. The ultimate CSS battle: Grid vs flexbox. URL: <https://hackernoon.com/the-ultimate-css-battle-grid-vs-flexbox-d40da0449faf>.
- [Cana] CanIuse. Browser support von css-variablen. URL: <https://caniuse.com/#search=custom%20properties>.
- [Canb] CanIuse. Can i use... support tables for HTML5, CSS3, etc. URL: <https://caniuse.com/#search=grid>.
- [cssa] CSS snapshot 2017. URL: <https://www.w3.org/TR/CSS/>.
- [CSSb] CSS-Tricks. A complete guide to flexbox. URL: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>.
- [fle] Flexbox vs CSS grid: A practical example • silo creativo. URL: <https://www.silocreativo.com/en/flexbox-vs-css-grid-practical-example/>.
- [Hoo16] Oliver Hoogvliet. JSON Web Token (JWT) im Detail, November 2016. URL: <https://blog.codecentric.de/2016/11/json-web-token-jwt-im-detail/>.
- [htm] HTML 5.2. URL: <https://www.w3.org/TR/html52/>.
- [jav] JavaScript | MDN. URL: <https://developer.mozilla.org/en-US/docs/Web/Javascript>.
- [jqu] jQuery Foundation jquery.org. jQuery. URL: <https://jquery.com/>.
- [Jur13] Heike Jurzik. *Debian GNU/Linux: das umfassende Handbuch ; [von der Installation bis zur Administration ; Office, Internet, Audio, Video und Shell ; Debian als Server nutzen, Netzwerk und Sicherheit ; aktuell zu "Wheezy"]*. Galileo Computing. Galileo Press, Bonn, 5., aktualisierte aufl edition, 2013. OCLC: 861209633.
- [MDNa] MDN. Basic concepts of flexbox. URL: [https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_Flexible\\_Box\\_Layout/Basic\\_Concepts\\_of\\_Flexbox](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox).

- [MDNb] MDN. CSS grid layout. URL:  
[https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_Grid\\_Layout](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout).
- [Nic] Gallagher Nicolas. About normalize.css – nicolas gallagher. URL:  
<http://nicolasgallagher.com/about-normalize-css/>.
- [noaa] BAuA - Optische Strahlung - Bundesanstalt für Arbeitsschutz und Arbeitsmedizin. URL:  
[https://www.baua.de/DE/Themen/Arbeitsgestaltung-im-Betrieb/Physikalische-Faktoren-und-Arbeitsumgebung/Optische-Strahlung/\\_functions/BereichsPublikationssuche\\_Formular.html](https://www.baua.de/DE/Themen/Arbeitsgestaltung-im-Betrieb/Physikalische-Faktoren-und-Arbeitsumgebung/Optische-Strahlung/_functions/BereichsPublikationssuche_Formular.html).
- [noab] DMX german. URL: <http://www.theater-technisch-lab.nl/dmxdu.htm>.
- [noac] DMX512/1990. URL:  
<http://www.soundlight.de/techtips/dmx512/dmx512.htm>.
- [noad] Download Raspbian for Raspberry Pi. URL:  
<https://www.raspberrypi.org/downloads/raspbian/>.
- [noae] Elektronik: IR-Fernbedienung, RC-5. URL:  
<http://www.sprut.de/electronic/ir/rc5.htm>.
- [noaf] mod\_rewrite - Apache HTTP Server Version 2.4. URL:  
[http://httpd.apache.org/docs/current/mod/mod\\_rewrite.html](http://httpd.apache.org/docs/current/mod/mod_rewrite.html).
- [noag] mod\_ssl - Apache HTTP Server Version 2.4. URL:  
[https://httpd.apache.org/docs/2.4/mod/mod\\_ssl.html](https://httpd.apache.org/docs/2.4/mod/mod_ssl.html).
- [noah] Open Lighting Architecture: Open Lighting Architecture Developer Documentation. URL: <http://docs.openlighting.org/ola/doc/latest/>.
- [noai] Open Lighting Project | Open Source Lighting Control Software. URL:  
<https://www.openlighting.org/>.
- [noaj] PHP-CPP - A C++ library for developing PHP extensions. URL:  
<http://www.php-cpp.com>.
- [noak] Ratchet - PHP WebSockets. URL: <http://socketo.me/>.
- [noal] SB-Projects - IR Index. URL:  
<https://www.sbprojects.net/knowledge/ir/index.php>.
- [noam] Uncomplicated Firewall (ufw) - Debian Wiki. URL:  
<https://wiki.debian.org/Uncomplicated%20Firewall%20%28ufw%29>.

- [noan] Was ist Proxy-Server? - Definition von WhatIs.com. URL: <http://www.searchnetworking.de/definition/Proxy-Server>.
- [noao] WiringPi. URL: <http://wiringpi.com/>.
- [noa91] *Der Neue Brockhaus: Lexikon und Wörterbuch in fünf Bänden und einem Atlas. 1: A - EK.* Brockhaus, Mannheim, 7., mit nachträgen versehene aufl edition, 1991. OCLC: 257736382.
- [noa18] php-jwt: PEAR package for JWT, April 2018. original-date: 2012-12-31T22:30:39Z. URL: <https://github.com/firebase/php-jwt>.
- [Sas] Sass-Lang. File: SASS\_reference — documentation by YARD 0.9.12. URL: [https://sass-lang.com/documentation/file.SASS\\_REFERENCE.html](https://sass-lang.com/documentation/file.SASS_REFERENCE.html).
- [Sch14] Maik Schmidt. *Raspberry Pi: Einstieg - Optimierung - Projekte.* C't Hardware Hacks Edition. dpunkt.verl, Heidelberg, 2., aktualisierte und erw. aufl edition, 2014. OCLC: 876883730.
- [Sha] Craig Sharkie. *Jump start responsive web design.* OCLC: 987006602. URL: <http://proquest.safaribooksonline.com/?fpi=9781492020615>.
- [tut] tutorialspoint. Websocket events und actions. URL: [https://www.tutorialspoint.com/websockets/websockets\\_events\\_actions.htm](https://www.tutorialspoint.com/websockets/websockets_events_actions.htm).
- [UE] Malte Ubl and Kitamura Eiji. Einführung zu WebSockets: Sockets im web - HTML5 rocks. URL: <https://www.html5rocks.com/de/tutorials/websockets/basics/>.
- [W3Ca] W3C. CSS custom properties for cascading variables module level 1. URL: <https://www.w3.org/TR/css-variables-1/>.
- [W3Cb] W3C. CSS flexible box layout module level 1. URL: <https://www.w3.org/TR/css-flexbox-1/>.
- [W3Cc] W3C. CSS grid layout module level 1. URL: <https://www.w3.org/TR/css-grid/>.
- [W3Cd] W3C. Websockets official. URL: <https://www.w3.org/TR/websockets/>.
- [WH17] Christian Wenz and Tobias Hauser. *PHP 7 und MySQL: das umfassende Handbuch.* Number 4082 in Rheinwerk Computing. Rheinwerk Verlag GmbH, Bonn, 2., aktualisierte auflage, 1. korrigierter nachdruck edition, 2017. OCLC: 986530008.

[wha] whatwg. Websocket official spec. URL:  
<https://html.spec.whatwg.org/multipage/web-sockets.html#network>.

— Druckgröße kontrollieren! —



Breite = 100 mm  
Höhe = 50 mm

— Diese Seite nach dem Druck entfernen! —

Diese  
Seite  
nach dem  
Druck  
entfer-  
nen!