# Die Verwendung eines SingleBoardComputers als Medienserver

## Chancen und Grenzen anhand eines praktischen Beispiels

## Bachelor-Thesis

**zur Erlangung des akademischen Grades B.Sc.**


## Axel Lodyga

**2121386**

Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Design, Medien und Information
Department Medientechnik


Erstprüfer: Prof. Dr.-Ing. Roland Greule

Zweitprüfer: B.Sc Fabian Oving

Hamburg, 22. August 2016

# Inhaltsverzeichnis

# Abstract

The work at hand describes the conception of a compressed media server. To develop the scope of functions, various media serves are compared and their characteristics established. A SingleBoardComputer is ought to serve as the hardware basis. In addition, a comparison between the underlying SBC and a competing product is drawn and a performance evaluation is provided. Furthermore, the development of an additional clip-on module to the DMX-control is described, as well as the necessary software for the application as a media server. The resulting concept of the media server builds upon a combination of Python, HTML5 and JavaScript. Concluding, derived advantages and disadvantages are examined and an outlook on the projects development possibilities' is presented.



# Zusammenfassung

Die vorliegende Arbeit beschreibt die Konzeptionierung eines komprimierten Medienservers. Um den Funktionsumfang eines solchen Servers zu erarbeiten, werden verschiedene Medienserver miteinander verglichen und deren Merkmale heraus gearbeitet. Als Hardwaregrundlage soll ein SingleBoardComputer dienen. Zusätzlich wird der zugrundeliegende Einplatinenrechner mit einem Konkurrenzprodukt verglichen und auf seine Leistungsfähigkeit hin untersucht. Die Entwicklung eines zusätzlichen Aufsteckmodules zur DMX-Steuerung wird beschrieben, wie auch die für den Einsatz als Medienserver nötige Software. Dabei setzt das entstandene Konzept des Medienservers auf eine Kombination aus Python, HTML5 und JavaScript. Abschließend werden noch Vor- und Nachteile gegenüber gestellt, sowie ein Ausblick auf die Entwicklungsmöglichkeiten des Projekts gegeben.

# 1. Einleitung

Sie sitzen in einem Theater und sind begeistert von dem Bühnenbild. Es besteht aus diversen Leinwänden, die mit scheinbar unabhängig voneinander laufenden Bildabfolgen bespielt werden und so ein lebendiges und vielseitiges Bühnenbild kreieren. Auf den ersten Blick erscheint die Erstellung solch eines Bühnenbildes einfach. Um Videos angepasst auf eine Leinwand zu bringen, benötigt man lediglich einen Projektor und ein passendes Abspielgerät, Laptop oder einen DVD-Player. Um ein Bühnenbild, das Begeisterung hervorruft, zu erstellen, bedarf es jedoch einiges mehr: Wenn man die Leinwand zum Beispiel in einem Theaterstück einsetzten will und in der Mitte soll dann noch eine Tür eingebaut werden, sodass die Leinwand ein gemaltes Bühnenbild ersetzt, wird es schon komplizierter. Sollte ein „experimentierfreudiger“ Bühnenbildner jetzt noch auf die Idee kommen, dass rechts, links und oberhalb der Tür jeweils ein anderes Video abgespielt werden soll und dass sich am besten noch spontan der Inhalt per Knopfdruck ändern möge, so ist es mit einem DVD-Player nicht mehr getan. Auch wenn das Video in der Postproduktion, also von vorn herein so produziert, geschnitten und bearbeitet ist, dass unter der Nichtbeachtung der spontanen Inhaltswechsel alles den Gedanken des Bühnenbildners entspricht, so wird das alles mit einem Laptop ohne die passende Software kaum realisierbar sein. Sollten die Ansprüche noch weiter steigen und man hat nicht nur eine flache Leinwand, auf welche man projizieren möchte, sondern ein abstraktes Konstrukt von Formen und Körpern, so wird spezielle Soft- und eventuell auch Hardware benötigt. Dieses besondere Etwas nennt man in der Event- oder Medientechnik „Medienserver“. Jedoch sind diese Gerätschaften mitunter sehr teuer. Auch wenn für den konkreten Bedarfsfall gar nicht alle Funktionen benötigt werden, so muss man schnell mehrere tausend Euro (vgl. A.3) in die Hand nehmen, um die eben geschilderten Vorstellungen zu realisieren. In Anbetracht dessen stellt sich die Frage, wie ein Medienserver gestaltet werden kann, der kleiner und günstiger ist, sodass dieser für ein größeres Publikum zugänglich und handhabbar ist, da gerade im (Laien-)Theater oft nur sehr wenig bis gar kein Budget vorhanden ist. Die Entwicklung und Umsetzung solch eines Medienservers ist das Thema dieser Bachelor-Thesis. In den folgenden Kapiteln wird darauf eingegangen, was einen Medienserver in seinen Kernfunktionen definiert und inwieweit man die einzelnen Komponenten durch günstige Lösungen ersetzen, aber die Variabilität beibehalten kann.

# 2. Was ist ein Medienserver?

Ein Medienserver ist meistens ein konventioneller Computer mit spezieller Software, welche die Verarbeitung und Verwaltung von Bild- und Videomaterial übernimmt. Somit macht die Software einen PC erst zu einem Medienserver.
Jedoch unterscheidet sich die Verwendung des Begriffes „Medienserver“ im alltäglichen Gebrauch von dem, der in der Event- und Medientechnik üblich ist. Im Verbraucherumfeld wird bei einer Verwaltungslösung für die heimischen Videos, Bilder und Musikdaten von der Funktion eines Medienservers gesprochen (vgl. AVM (2015)). Zudem gibt es den Begriff „Medienserver“ auch im Sprachgebrauch von Bibliothekaren. Hier verwendet man einen derartigen Server für das sichere Verwalten, Katalogisieren und automatische Verbinden mit Zusatzinformationen. In diesem Sinne behandelt auch Seifert den Begriff „Medienserver“:

*„Ein Medienserver ist eine Software, die große Objektmengen unterschiedlicher Herkunft sicher verwalten kann. Die Objekte oder Medien werden mit Metadaten angereichert und kategorisiert gespeichert. Basistypen der Medien bilden Texte oder Bilder, weitere Typen sind aber denkbar.“* (Seifert 2010: 8)

Während die bereits vorgestellten Verwendungen des Begriffes „Medienserver“ definiert sind als eine Applikation, welche Verwaltungsaufgaben bewältigt, so übernimmt der Medienserver im Bereich der Event- und Medientechnik zudem etliche weitere Aufgaben. Er dient nicht nur zu Verwaltung, sondern auch zur Manipulation und Ausgabe dieser Daten. Nachfolgend soll das Augenmerk genau auf der Darstellung der Medieninhalte liegen, da diese einen Medienserver von einem Medienverwalter grundlegend unterscheidet. Ein weiterer Unterschied liegt in der Art und Weise der Steuerung eines solchen Gerätes. Beispielsweise wird die Verwaltungs- und Wiedergabesoftware Plex (vgl. Plex (2016)), sowie ähnliche Produkte, gesteuert vom wiedergebenden Gerät. Der Bibliothekars-Server zum Beispiel von der Weboberfläche, auf welcher die Informationen abgebildet werden kontrolliert. Somit ist der Server physikalisch getrennt von der Ein- und Ausgabe. Da der Medienserver aus der Medientechnik zugleich das ausgebende Gerät beinhaltet, wird dieser anders angesteuert. Er kann sowohl auf der Maschine selbst als auch von externen Quellen gesteuert werden. Ob die Steuerung nun ein Lichtstellpult oder eine Lasererfassungseinheit (vgl. Coolux (2016)) übernimmt, hängt dabei ganz von der eingesetzten Soft- und Hardware ab.

## 2.1. Merkmale

Die Software „PandorasBox“ von Coolox ist unter Berücksichtigung der Systemmindestvoraussetzungen auf jedem PC installier- und anwendbar. Jedoch ist, je nach gegebener Hardwarekomponenten, nicht der volle Funktionsumfang nutzbar. Sofern nur eine Grafikkarte mit zwei Ausgabeschnittstellen verwendet wird, lassen sich nicht ohne Weiteres sechs Projektoren anschließen. Hinzu kommen eventuelle Leistungseinbrüche durch nicht ausreichenden Arbeitsspeicher, ruckelnde Videos aufgrund zu langsamer Festplatte und so weiter. Um solche Problematiken zu vermeiden, bieten die Hersteller konfektionierte Hardware inkl. der Software und der zugehörigen Lizenzen an.
Bei der Betrachtung des aktuellen Marktes und der Vielzahl an Möglichkeiten, die dieser im Bereich der Medienserver im Sinne der Event- und Veranstaltungstechnik bereit hält, kann man schnell den Überblick über die eigentlichen Merkmale eines solchen verlieren. Um eine Basis zur Konzeptionierung eines neuen Medienservers zu schaffen, werden zunächst aus einer Auswahl von aktuell verfügbaren Modellen verschiedener Hersteller, Typen und Hauptanwendungsgebieten die verschiedenen Eckdaten zusammengefasst und eine Schnittmenge der Funktionalitäten gebildet.
Dazu werden folgende Medienserver betrachtet:

| Hersteller | Modell |
|---|---|
| Green Hippo | Amba |
| ArKaos Pro | Stage Server |
| MX Wendler | FXSERVER |
| d3 | d3 4x2pro |
| Green Hippo | Taiga |
| Coolux | Pandoras Box |
| AV Stumpfl | Wings Engine Stage |
| MA | MA VPU basic MK2 |
| Face | ImageCue |

Tabelle 2.1.: Auflistung der zu vergleichenden Medienserver

## 2.2. Grundfunktionalitäten

Alle Medienserver beinhalten die Ausgabe von Video- und Bildmaterial über Standardvideoanschlüsse. Das heißt, von VGA über HDMI zu Mini-Displayport kommt es ganz auf den spezifischen Typ Medienserver an, nämlich mit welcher Art und Anzahl dieser ausgestattet ist. Jedoch stehen immer mindestens zwei Anschlüsse (siehe Tabelle A.3) zur Verfügung, und zwar einer für die Steuerung und Konfiguration sowie einer für den eigentlichen Content. Dieser Inhalt kann zum Beispiel über einen Beamer, Bildschirme, oder sonstige Bildverarbeitungsgeräte wie Pixelmapper dargestellt werden.
Dabei werden Pixelmapper benutzt, um das Videosignal, welches bevorzugt für einen Monitor oder einen Projektor gedacht ist, umzubauen, sodass es auf einer LED-Leinwand oder auf einzelnen Leuchten abgespielt werden kann. Dabei wird das Videosignal Pixel für Pixel in ein DMX-Signal oder anderes Steuersignal, wie zum Beispiel „ArtNet“, umgewandelt. Im Medienserver wird die Videoausgabe, ob nun durch einen Projektor oder einen Pixelmapper, gleich behandelt, es sei denn, dass in dem Medienserver eine solche Funktion bereits direkt integriert ist, sodass das Signal nicht noch einmal umgewandelt werden muss.

### 2.2.1. Layer

In diesen Videoausgaben können sogenannte Layer definiert werden. Das sind Bereiche der Ausgabe, welchen Inhalte zugewiesen werden können. Hierbei spielt es zunächst keine Rolle, ob in diesem Bereich ein Bild, eine grafische Animation (GIF) oder ein Video dargestellt werden soll. Ein solcher Layer muss zudem nicht zwingend zweidimensional sein. Es gibt auch dreidimensionale Layer. Meist definieren sie einfache Körper, wie Würfel oder Kugeln, auf deren Oberfläche dann der Inhalt angepasst dargestellt wird. Ebenfalls ist es mittlerweile auch möglich, Inhalte auf komplexeren Körpern, wie Menschen oder Häusern, abzubilden. Diese Art der Projektion oder Darstellung wird dann 3D-Mapping (siehe Abbildung A.1) genannt, sobald der Bildinhalt auf echte dreidimensionale Formen abgebildet wird.
Dabei besteht jederzeit die Möglichkeit, diese Layer oder Ebenen beliebig dynamisch zu verändern. Sowohl der Inhalt, die Größe oder Position der Bildebene lassen sich steuern, sodass durch die Komposition und Veränderungen eigene neue Inhalte entstehen können.

Abbildung A.1.: 3D-Projection: Mapping auf einfachen Körpern[1]

### 2.2.2. Verarbeitung der Layer

Diese Parameteränderungen können über verschiedene Wege angesprochen werden. Zunächst kann man alle Werte fest am Medienserver einstellen(siehe Abbildung A.2).


Abbildung A.2.: Layersteuerung: Coolux Pandoras Box - Layersteurung[2]

[1] Quelle: Sandra Ciampone http://plusinsight.de/wordpress/wp-content/gallery/bilder_artikel/dynamic/41_sonyps3-1.jpg-nggid03571-ngg0dyn-618x450x100-00f0w010c010r110f110r010t010.jpg Aufruf: 06.08.2016

[2] Quelle: Axel Lodyga 2016 - Screenshot

Zudem lassen sich aber auch fest definierte Abläufe von Änderungen im Server definieren, oder sie werden durch ein externes Signal verändert. In der Eventtechnik gibt es verschiedene, definierte Signalprotokolle, welche je nach Hersteller und Modell unterstützt werden. Das strukturell und technisch simpelste, das bei den betrachteten Medienservern unterstützt wird, ist DMX-512, welches unter Vernachlässigung von RDM ein differenzielles unidirektionales Steuersignal darstellt. Es kann insgesamt 512 Kanäle à 256 (0-255) Parameter 44,1-mal pro Sekunde abbilden.
Somit können mit einem DMX-Signal, ein sogenanntes Universum, 512 verschiedene Parameter à 256 Werten von außen den Medienserver beeinflussen. Die Zuordnung der Kanäle zu den jeweiligen Parametern im Server ist entweder fix definiert (siehe Abbildung A.16) oder beliebig anpassbar. Einer Videoebene sind so zum Beispiel folgende Grundeigenschaften zugewiesen, welche über das Steuersignal dynamisch änderbar sind.

- Inhalt (Bild, Video, etc.)
- Translation (Verschiebung)
- Rotation (Drehung)
- Skalierung (Größe)
- Opazität (Transparenz/Deckkraft)

Zudem ist es je nach Medienserver möglich, viele weitere Eigenschaften eines Layers zu definieren und zu manipulieren, wobei sich diese je nach Inhalt und Typ des Layers nochmal spezifizieren oder ändern können. So kann zum Beispiel ein einfaches Bild keine Abspielgeschwindigkeit definiert bekommen oder Lautstärke. Ebenso sind weitere änderbare Parameter möglich:

- Softedge (auslaufende Kanten)
- Warping (Verzerrung)
- Bildverarbeitung (Sättigung, Helligkeit, ...)
- Abspielgeschwindigkeit (bei Videos)
- Masken (oder Gobos)

Wobei auch hier das Funktionsspektrum ganz vom ausgewählten Server abhängt. Es kann je nach Anzahl der Layer ein einziges DMX-Universum schnell vergeben sein. Dazu eine Beispielrechnung:

Inhalt: 1 Kanal
Translation im dreidimensionalen Raum : 3 Kanäle
Rotation im dreidimensionalen Raum : 3 Kanäle
Skalierung im Raum : min 2 Kanäle (wenn zweidimensional)
Opazität : 1 Kanal
Sättigung je Farbe (RGB) , Helligkeit : min 4 Kanäle
Abspielgeschwindigkeit: 1 Kanal
Softedge pro Seite : min 4 Kanäle
Verzerrung: min 4x2 Kanäle (jeder Eckpunkt mit x/y Koordinate )
Masken : min 6 [nochmals eigene Translation(2D), Rotation(2D), Opazität und Skalierung(2D)]
Insgesamt pro Layer: min 32 Kanäle

Somit kann man trotz des im Beispiel verwendeten überschaubaren Funktionsspektrums mit einem DMX-Universum maximal 16 Layer ansteuern, sofern jeder Layer seine eigenen Steuerkanäle erhält. Sollte man mehr als 16 Ebenen oder einen weitaus größeren Funktionsumfang abbilden wollen, gibt es mehrere Möglichkeiten, dies anzugehen.
Entweder man hat die Möglichkeit, sowohl vom Server als auch vom Lichtstellpult und dem Medienserver her mehrere „Universen“ zu benutzen, oder man weicht auf ein anderes Steuerprotokoll aus, welches nicht so limitiert ist. Häufig wird auf ArtNet oder ein herstellereigenes Protokoll, wie MAnet oder Hognet, zurückgegriffen. Diese Steurprotokolle arbeiten ebenfalls mit Kanälen und den zugehörigen Parameterwerten von 0 bis 255. Jedoch können aufgrund der Übertragung per UDP, also mit Hilfe der Ethernetschnittstelle, gleich mehrere DMX-Universen abgebildet werden. So können nach „ArtNet 3 Spezifikation“ theoretisch bis zu knapp 33.000 DMX-Universen abgehandelt werden (vgl. (Art-Net 2016: 5)). Eine weitere Methode ist das Zusammenlegen verschiedener Funktionen auf einen DMX-Kanal. So steht ein Kanal nur zur Layerauswahl zur Verfügung, um mit weiteren 32 Kanälen die Parameter zu definieren. Es lassen sich theoretisch 256 Ebenen mit nur 33 Kanälen ansteuern. Jedoch ist die simultane Änderung mehrerer Layer-/Parameterkombinationen mit dem letzten Lösungsansatz nicht mehr möglich, da immer erst ein Layer gewechselt werden muss, um den entsprechenden Parameter ansprechen zu können. Dabei entstehen jedoch neue Problematiken, wie die Abfrage der zuletzt gesetzten Einstellung.

## 2.3. Unterschiede

Da verschiedene Medienserver für unterschiedliche Anwendungszwecke konzeptioniert wurden und werden, geht die Ausstattung, die man erhält, von speziell auf den Aufgabenbereich zugeschnittenen Versionen bis hin zu ausgewogenen und „Allround"-Paketen. Je nach Anspruch und Budget kann man zwischen viel und wenig wählen.

| Hersteller | Green Hippo | | MX Wendler | d3 | coolox | AV Stumpfl | MA | Face |
|---|---|---|---|---|---|---|---|---|
| Modell | Amba | Taiga | FX Server | d3 4x4pro | Pandoras Box Server Pro | Wings Engine Stage | MA VPU basic MK2 | ImageCue |
| max. Content Outputs | 1 | 6 | 6 | 4 | 4 | 4 | 3 | 1 |
| Controle Outputs | 1 | 2 | up to 6 | 1 | | 1 | | 0 |
| max. Resolution | 4K | 4K | FULL HD | 4K | 8K | FULL HD | FULL HD | FULL HD |
| max. Layers | 64 | unlimited | 64 | | unlimited | unlimited | 32 | 1 |
| HD Playback Layers | 8 | 40 | | 32 | 24 | | 4 | 1 |
| 4K Playback Layers | 2 | 8 | | 8 | 2 | | 0 | 0 |
| Video Inputs | 1 | 8 | 4 | 4 | up to 20 | 2 | 2 | 0 |
| | keine Infos | | | | | IO immer extern | | |
| DMX | ✓ | ✓ | ✓ | nur sendend | ✓ | ✗ | ✓ | ✓ |
| ArtNet | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ |
| SACN | ! | ! | ✗ | ✗ | ✓ | ✗ | ! | ✓ |
| ADAT | ! | ! | ✗ | ✓ | ! | ✗ | ! | ! |
| OSC | ✓ | ✓ | ✓ | ! | ! | ✗ | ! | ! |
| d3 Net | ! | ! | ! | ✓ | ! | ✗ | ! | ! |
| MIDI | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ! | ! |
| MANet | | ! | ✗ | ! | ✓ | ✗ | ✓ | ✗ |
| HogNet | ! | ! | ✗ | ! | ✗ | ✗ | ! | ✗ |
| Transformation 2D | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ |
| Transformation 3D | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ |
| Transparenz | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Color editing | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Masks | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ |
| | | | | | | | | |
| Splitting | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ! | ✗ |
| Warp | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ |
| Softedge | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ |
| Dome Projection | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ |
| | | | | | | | | |
| Partikeleffects | ✗ | ✗ | ✗ | ✗ | ✓ | ! | ✗ | ✗ |
| 3D-Object Import | ✓ | ✓ | nur .3DS | ✓ | ✓ | ✓ | ✓ | ✗ |
| Video Encoder | ✓ | ✓ | ✓ | ✓ | optional | ! | ✗ | ✗ |
| Imageconverter | ✓ | ✓ | ✓ | ✓ | ✓ | ! | ✗ | ✗ |
| Pixelmapper | ✓ | ✓ | ✗ | ✓ | ✓ | ! | ✓ | ✗ |
| | | | | | | | | |
| Price (exc. VAT) | ca. 7.100 € | ca. 43.000€ | min 10.000€ | ca. 90.000 € | ca. 54.000€ | 1.400€ am Tag mieten | 14.000 € | 826 € |

✗ *nicht vorhanden/ nicht möglich*
! *keine Angaben*
✓ *vorhanden/möglich*

Quellen:
http://www.arkaospro.com/media-servers/arkaos-stage-server
http://shop.lmp.de/de_DE/001000000/001050000/001050100/00182712.html
http://www.eventtechniker.de/hauptbuehne/medientechnik-mainmenu-73/produktvorstellungen/5696-devicecontext-veroeffentlicht-mxwendler-fxserver-und-stage-designer-software-50-mit-umfangreichen-leistungsstarken-features-und-gesteigerter-performance.html
http://www.arkaospro.com/media-servers/arkaos-stage-server
http://www.3dbroadcastsales.com/green-hippo-hippotizer-v4-amba
http://plsn.com/current-issue.html/39-road-tests/15189-d3-technologies-4x4pro-1.html
http://www.coolux.de/index.php?id=dowload-center#file/false/file-252
https://www.huss-licht-ton.de/product_info.php/MA-Lighting-VPU-Basic-MK-II-Video-Processing-Unit/info/12595.html
http://www.coolux.de/index.php?id=dowload-center#file/cat-50/file-188
https://avstumpfl.com/fileadmin/user_upload/downloads/de/MediaControl_Handbuch/Wings_Engine_Stage_Handbuch.pdf
https://avstumpfl.com/de/media-control-systeme/medienserver/wings-engine-stage/technische-details/
http://www.veranstaltungstechnik.com/wp-content/uploads/2016/01/VTG_Preisliste-2016-1.pdf

Abbildung A.3.: Vergleich der verschiedenen Medienserver

Die auffälligsten Unterscheidungsmerkmale sind demnach die Anzahl der Ausgabemöglichkeiten und die Anzahl der Layer sowie deren jeweilige Auflösung (vgl. A.3). Einige der verglichenen Produkte haben noch besondere Möglichkeiten der Signalverarbeitung: Von der Anzahl der Videoeingänge, der Effektmöglichkeiten bis hin zu einem integriertem Pixelmapper. Die Art und Anzahl der verfügbaren Steuersignale unterscheidet sich meistens lediglich in der Implementierung proprietärer Herstellerformate, wie

MANet oder HogNet. Aus der Reihe fällt lediglich das Produkt „ImageCue“ der Firma „Face“. Dieser Medienserver ist, ebenso wie das zu konzipierende Projekt, auf einem SingleBoardComputer gestützt. Er besitz jedoch, wie in Abbildung A.3 zu sehen, nur einen einzelnen Layer, welcher jedoch keine der in Abschnitt 2.2.2 definierten Grundfunktionalitäten bietet. Lediglich der Inhalt und die Transparenz lassen sich durch externe Steursignale beeinflussen. Er bietet also keine richtige Alternative zu den großen Medienservern.


Abbildung A.4.: Medienserver von Face - ImageCue[3]

[3]Quelle: Face 2016 https://store.face.be/imagecue/imagecue-hd-image-server?returnurl=%2fimagecue%2f%2338579 Abruf 17.08.2016

# 3. Was definiert einen SingleBoardComputer?

Der Ausdruck SigleBoardComputer (SBC) beschreibt eine Platine, auf der alle essentiellen Bauteile vorhanden sind, um einen Personal-Computer funktionstüchtig abzubilden. Oftmals handelt es sich jedoch nicht wie bei einem Laptop oder einem Standrechner um austauschbare Einzelkomponenten, sondern um einen „System on a Chip“ (SOC) mit weiteren SMD-Bauteilen. Ein SOC , ist die Fusion einer CPU, der Recheneinheit des Grafikchips und dem BIOS, dem rudimentäre Betriebssystem als auch der Bestandteile zur Steuerung aller Anschlussmöglichkeiten zu einem einzigen Chip. In den meisten Fällen ist auch der Arbeitsspeicher direkt mit in diesen Systemchip integriert. Weitere Bestandteile der Platine sind die Buchsen für alle nötigen Anschlüsse, Chips zur Verarbeitung eines kabelgebundenen Netzwerks oder auch Bluetooth und WLAN. Für die Bildausgabe ist häufig HDMI und ein FBAS-Anschluss vorhanden. Peripheriegeräte wie Maus und Tastatur werden üblicherweise über USB angeschlossen, wovon zwei bis vier je nach SBC vorhanden sind. Der Systemspeicher wird in der Regel über eine (Micro)SD-Karte bereitgestellt oder, sofern vorhanden, auf einem integrierten Speicherbaustein. Ein paar Status-LEDs, ein Anschluss zur Stromversorgung und alles, was man für einen Rechner braucht, ist auf einer kleinen Platine vereint.

## 3.1. Grundlegende Funktionen/Aufbau

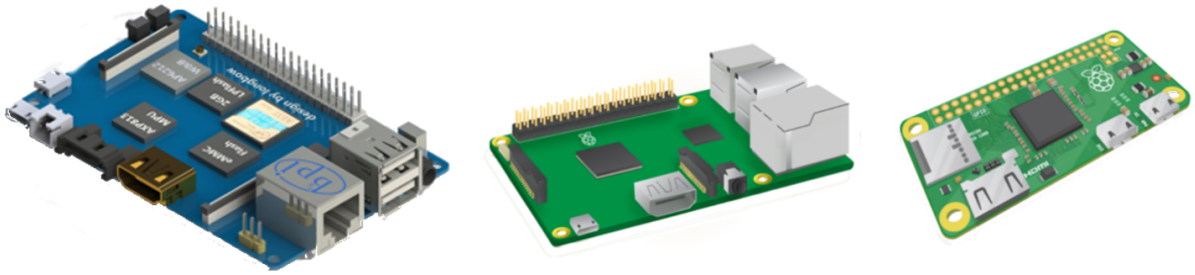

Abbildung A.5.: SingleBoardComuter: V.l.n.r.: Banana Pi M3[1], Raspberry Pi Modell 3B[2], Raspberry Pi Zero[3]

Je nach Modell und Hersteller können die Größe und auch die Ausstattung eines SBCs sehr unterschiedlich sein. So hat die Raspberry Foundation neben dem aktuellen „Top-Modell“, dem „Raspberry Pi 3 B“, einen winzigen Bruder „Raspberry Pi Zero“ im Angebot, welche sich deutlich durch Größe und auch Systemleistung unterscheiden (siehe Abbildung A.5). Ds ist zum einen die unterschiedliche Rechenleistung, die sich aus der Taktrate als auch der Anzahl der Rechenkerne der SOCs ermittelt, als auch die Art und Größe des Arbeitsspeichers. Videotechnisch unterstützen alle gängigen kleinen SingleBoardComputer FullHD, wobei sich auch Leistungsunterschiede in den Grafikeinheiten bemerkbar machen können, sofern diese vom System gut nutzbar sind. Hierbei trägt die Auswahl des Betriebstriebsystems einen erheblichen Anteil zu den Unterschieden bezüglich der Art und Nutzbarkeit eines solchen Kleinrechners bei. Fast alle Hersteller bieten verschiedene angepasste Linux-Distributionen unter anderem auch Android oder, wie beim Raspberry Pi 3, eine Windows 10 IOT Version an. Jedoch lassen sich auch bei gleichen Systemen auf unterschiedlichen Platinen gravierende Leistungsunterschiede aufgrund von inkompatiblen oder schlichtweg nicht entwickelten Treibern feststellen (siehe Abbildung A.6). So ist zum Beispiel die hardwarebeschleunigte Wiedergabe sowie das Decodieren von Videos nicht auf allen Systemen und mit jedem Codec möglich.

## 3.2. Mögliche Typen

Bei der Auswahl eines geeigneten SBC für die Umsetzung eines Medienservers, war zum einen das Preisleistungsverhältnis als auch die Community Unterstützung aus-

[1]Quelle: SinoVoip 2015 - http://forum.banana-pi.org/t/bpi-m3-3d-design-file/715 Abruf: 06.08.2016
[2]Quelle: Raspberry Pi Foundation 2016 - https://www.raspberrypi.org/wp-content/uploads/2016/02/Pi_3_Model_B.png Abruf: 06.08.2016
[3]Quelle: Raspberry Pi Foundation 2016 - https://www.raspberrypi.org/wp-content/uploads/2015/11/Pi_Zero_v1.2.png Abruf: 06.08.2016

schlaggebend. Die von der Raspberry Pi Foundation entwickelten und verkauften Platinen erfreuen sich im Bastler- und Entwicklerbereich großer Beliebtheit und somit ist die Unterstützung durch Foren und Blogs sehr gut vorhanden. Einer der Konkurrenten mit stärkeren Leistungsmerkmalen hat dieses Privileg leider nicht in gleichem Umfang. Da die Raspberry Pis den Standard von kleinen Entwicklerboards darstellen, gibt es diverse Erweiterungsplatinen zu erwerben. Diese Platinen werden auf den GPIO-Header, also die 20-40 Pin Anschlussstiftleiste, gesteckt und können so den Funktionsumfang des scheckkartengroßen Computers mit den unterschiedlichsten Funktionen ausweiten. Um diese Funktionalitäten auch mit den Konkurrenzproduktionen anderer Hersteller zu ermöglichen, integrieren diese eine baugleiche Stiftleiste. Jedoch ist darauf zu achten, dass auch die Belegung der einzelnen Anschlüsse, der Stiftleiste, mit denen der Pi's übereinstimmt, da es sonst im ungünstigsten Fall zu Kurzschlüssen und somit zur Zerstörung der Bauteile kommen könnte.
Zu Beginn des Projekts wurde der Pi 3 grade veröffentlicht und auch der Banana PI M3 feierte sein Debüt. Da der erstgenannte auf den bewehrten Pi Strukturen aufbaut, wurde dieser in die engere Auswahl genommen, während der Banana Pi aufgrund seines Datenblattes in Bezug auf Rechen- und Grafikleistung im Verhältnis zu seinem Preis als eine gute Entwicklungsgrundlage erschien.

### 3.2.1. Raspberry Pi 3

Der als Lern- und Entwicklungsplattform entwickelte Einplatinencomputer wurde konzipiert, um einen einfachen Einstieg in die Soft- und Hardwareprogrammierung für wenig Geld zu ermöglichen. Somit sollte die Forschung und der Ausbau der digitalen Welt vorangebracht werden.

*„Putting the power of digital making into the hands of people all over the world“* (RaspberryPi-Foundation 2016: 1)

Im Vergleich zu seinem Vorgänger „Raspberry Pi 2 Modell B“, welcher im Februar 2015 vorgestellt wurde, hat das Modell 3 B einige spektakuläre Neuerungen in die Familie der Pis gebracht. Der Dreier hat im Gegensatz zu allen anderen der Produktpalette einen 64Bit Armv8 Chipsatz und als erster eine integrierte Möglichkeit, über Funk zu kommunizieren. Dazu steht neben Bluetooth 4.1 & LE auch ein Wireless-LAN Chip zur Verfügung. Somit ist er für alle Projekte zwar noch immer abwärtskompatibel zu seinen Vorgängern, bietet aber deutliche Vorteile für kommende Einsatzzwecke.

### 3.2.2. BananaPi M3

Der erste Banana Pi, zunächst von der chinesischen Bildungsinitiative „Lemaker.org“ entwickelt und vertriebene, ist ein etwas verbesserter „Klon“ des Rasperry Pi 1 bzw. 2. Mit einem Mehrkernprozessor und der zusätzlichen Möglichkeit, Festplatten über SATA anzuschließen, hat sich zum aktuellen Modell nicht viel verändert. Der Preis beläuft sich noch immer knapp unter 100 Euro (vgl. Reichelt (2016)), jedoch ist die Ausstattung gewachsen und der Vertrieb als auch die Entwicklung wird nun von SinoVP bzw. der dahinter stehenden Fuchuang OSS Foundation übernommen. Lemaker.org und die Fuchuang OSS Fundation liegen bezüglich der Markenrechte des Banana Pi Projekts seit 2015 im Rechtsstreit (vgl. Banana-Pi (2016)). Daraus resultierend ist die Unterstützung und Entwicklung der Betriebssysteme in Mitleidenschaft gezogen worden. Zudem ist auch die Unterstützung durch andere Anwender stark zurückgegangen. Nichtsdestotrotz bleiben ein leistungsfähiger Prozessor und eine ebenfalls leistungsfähige Grafikeinheit stark herausragende Merkmale des M 3. Auch die Möglichkeit des Verzichts der SD-Karte durch die Nutzung des internen Speichers als auch der Erweiterung durch vollwertige Festplatten oder SSDs lassen den BananaPi als Grundlage für einen Medienserver günstig erscheinen.

## 3.3. Vergleich und Auswahl

Im folgenden Abschnitt werden verschiedene Parameter der 2 Computer miteinander verglichen, um eine Auswahl für die Grundlage eines komprimierten Medienservers zu bilden. Bei der Betrachtung der Spezifikationen und technischen Eigenschaften (siehe Tabelle A.17), stechen lediglich die starken Unterscheide in der Taktung und der Anzahl der Prozessorkerne hervor. Im Vergleich zum Raspberry Pi besitzt der Banana Pi auch noch ein paar Peripheriegeräte, wie ein Mikrofon und einen Infrarotsensor, mehr. Da der SATA-Anschluss bei genauerer Begutachtung lediglich über einen internen USB-Port implementiert ist, sollte dieser nicht als Auswahlkriterium in Betracht gezogen werden, da eine USB-Festplatte demnach keine Geschwindigkeitseinbußen mit sich bringen würde. Zu vergleichen gilt nun die Performanz und Treiberintegration beider SBCs.
Als Betriebssystem kommen Android und Windows nicht in Frage, da beide auf ein grafisches Grundsystem setzen, welches zusätzlich Leistung und Arbeitsspeicher des SBC ohne zweckmäßigen Nutzen verbrauchen würde. Da sowohl für den Raspberry Pi als auch für dessen Konkurrenten diverse Linux-Derivate zur Verfügung stehen, werden nur die zwei bekanntesten getestet, welche auf beiden Platinen-Computern gleichermaßen verfügbar sind. Das sind demnach zum Ersten „Raspbian“, ein speziell für den Raspberry Pi angepasstes Debian, zum Zweiten „Ubuntu Mate“, ebenfalls

ein Debian Derviat. Auf allen Systemen wird zunächst die Integrationsmöglichkeit von „Open Lighning Architecture“ (OLA) überprüft (vgl. Open-Lighting-Architecture (2016)). Als weitere Alternative kann man noch „Arch Linux“ nennen, welches sich aber aufgrund der umfangreichen Installation und Konfiguration nur bedingt eignet.
Um nun eine Aussage über die Leistungsfähigkeit der Grafikeinheit und deren Unterstützung zu tätigen, werden in jedem System die vorliegen Browser auf ihre HTML5-Videofähigkeiten überprüft. Zusätzlich wird noch versucht den Chromium Browser zu installieren, welcher ebenfalls auf seine Leistung bezüglich der Videoausgabe getestet wird.
Für die Tests wurde jederzeit dieselbe einfache Internetseite (vgl. w3schools (2016))benutzt. Um auch vergleichbare Werte zu erhalten, die mehr aussagen als „geht“ und „geht nicht“, wurde im jeweiligen Browser der FPS Zähler aktiviert (vgl. Chome (2015)). Eine weitere ähnliche Analyse wurde für die WebGL-Unterstützung durchgeführt (vgl. krpano (2015)). Und um eine ganz genaue Einstufung der Werte zu bekommen, wurde anschließend ein Leistungstest, ein sogenannter Benchmark, des Browsers durchgeführt (vgl. Futuremark (2013)). Um die gewonnenen Daten in Relation setzten zu können, wurden dieselben Tests auch auf einem eigenen knapp zwei Jahre alten, aber gut ausgestattetem Laptop durchgeführt (siehe Tabelle A.6).

Chromium wurde als zu verwendender Browser ausgewählt, da er sich durch Setzen von erweiterten Einstellungen optimieren lässt. Zudem ist es der einzige der verglichenen Browser, welcher eine Hardwarebeschleunigung der Videoanzeige ermöglicht. Das macht sich auch sehr deutlich in der Zahl der angezeigten Bilder pro Sekunde bemerkbar. Zudem ermöglicht es die Nutzung von WebGL, die für die Erweiterbarkeit und die Zukunft des Projekts relevant ist.
Aufgrund der besseren Treiberimplementierung und der Möglichkeiten, Chromium zur Hardwarebeschleunigung zu zwingen, wurde das System zunächst auf dem Raspberry Pi 3 gebaut. Zudem wurde als Betriebssystem „Rasbian Jessie“ gewählt, auch wenn Firefox unter „Ubuntu Mate“ eine höhere Benchmark-Leistung erzielte. Das begründet sich unter anderem in der besseren Unterstützung für WebGl. Änderungen für die Entwicklung der Software hat diese Entscheidung nicht zur Folge. Jedoch ist das Gehäuse an das gewählte Modell anzupassen. Da die Abmessungen und Anschlüsse beider Platinen nicht gleich sind, musste sich beim Bau des Gehäuses für eine Platine entschieden werden. Zudem sind während der Testdurchführung starke Temperaturanstiege aufgefallen, was zur Integration einer aktiven Belüftung des Gehäuses geführt hat. Mit dieser sind auch bei weiteren Tests extreme Temperaturspitzen von bis zu 80°C des SOCs ausgeblieben. Unter Dauerbelastung sind nun nur noch 55°C registriert worden.

| Modell | | Banana Pi M3 | | | Raspberry Pi 3 | | | A504 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Hersteller | | *SinoVoip* | | | *Raspberry Pi Foundation* | | | *XMG* | | |
| **Betriebssystem** | **Raspbian Jessie** | Kernel 3.4 | | | Kernel 4.4 | | | Kernel 4.7 | | |
| **Browser** | | HTML5 Video FPS | WebGL FPS | Pacekeeper | HTML5 Video FPS | WebGL FPS | Pacekeeper | HTML5 Video FPS | WebGL FPS | Pacekeeper |
| **Epihany** | | ✗ 12 | ✗ 0 | ✓ 328 | ✗ 12 | ✗ 0 | ✓ 407 | ✓ 60 | ✗ 0 | ✓ 2700 |
| **Firefox** | | ✓ 24 | ✗ 0,3 | ✓ 400 | ! 23 | ✗ 0,4 | ✓ 530 | ✓ 60 | ✓ 60 | ✓ 3250 |
| **Midori** | | ✗ 13 | ✗ 0 | ✓ 244 | ! 20 | ✗ 0 | ✓ 357 | ✓ 60 | ✗ 0 | ✓ 2915 |
| **Chromium** | | ! 23 | ✗ 2 | ✓ 523 | ! 20 | ✗ 0 | ✓ 495 | ✓ 60 | ✓ 60 | ✓ 3540 |
| **Chromium optimiert** | | ✗ 0 | ✗ 0 | ✗ 0 | ✓ 40 | ! 17 | ✓ 576 | ✗ 0 | ✗ 0 | ✗ 0 |
| | **OLA integration** | | ✓ | | | ✓ | | | ✓ | |
| | | | | | | | | **Eigener Laptop als Vergleich für "richtige" Computer** | | |
| **Betriebssystem** | **Ubuntu Mate 16.04** | Kernel 3.4 | | | Kernel 4.19 | | | | | |
| **Browser** | | HTML5 Video FPS | WebGL FPS | Pacekeeper | HTML5 Video FPS | WebGL FPS | Pacekeeper | | | |
| **Epihanie** | | ✗ 0 | ✗ 0 | ✗ 0 | ✗ 0 | ✗ 0 | ✗ 0 | | | |
| **Firefox** | | ! 23 | ✗ 0 | ✓ 611 | ✓ 30 | ✗ 5 | ✓ 624 | | | |
| **Midori** | | ! 23 | ✗ 0 | ✗ 0 | ! 23 | ✗ 0 | ✗ 0 | | | |
| **Chromium** | | ✓ 25 | ✗ 2 | ✓ 551 | ✗ 13 | ✗ 0 | ✓ 497 | | | |
| **Chromium optimiert** | | ✗ 0 | ✗ 0 | ✗ 0 | ✓ 35 | ✗ 1,2 | ✓ 496 | | | |
| | **OLA integration** | | ✓ | | | ✓ | | | | |

Abbildung A.6.: Vergleich der verschiedenen Betriebssysteme und Browser

# 4. Umsetzung

Das Ziel war, einen einfachen, zunächst über DMX, steuerbaren Medienserver auf SingleBoardComputer-Basis zu konstruieren. Dabei sollte er möglichst simpel in seiner Verwendung als auch in der Erweiterbarkeit der Funktionalitäten sein. Auch die Konfiguration und das Benutzen sollten übersichtlich und nicht zu komplex gestaltet sein.
Grundlegende Funktionen und Eigenschaften, die der Medienserver beinhalten soll:

- sehr kleine Abmessungen
- DMX (max 10 channels)
- FullHD Ausgang
- mindestens 2 Ebenen
- Funktionen:
    - Auswahl der Quelle
    - Transformation
    - Skalierung
    - Wiedergabegeschwindigkeit
    - Transparenz

## 4.1. Idee

Alle Komponenten von der Software über ein DMX-Interface wurden neu konzipiert, um allen Anforderungen gerecht zu werden. Bei der Videoausgabe wurde auf ein völlig neues Prinzip gesetzt. Es wurde weder ein eigener Grafikrenderer programmiert, noch wurde auf eine Grafikverarbeitungsbibliothek, wie OpenCL/OpenGl oder Direct3D, gesetzt. Der gesamte Inhalt wird durch eine HTML5-Seite ausgegeben. Somit sind zwar nicht alle Funktionalitäten, die ein 10.000 Euro teurer Medienserver mit sich bringt, umsetzbar; auch ist die Beanspruchung der zugrundeliegenden Hardware nicht optimal, jedoch lässt sich die gesamte Softwarestruktur einfach programmieren und erweitern. Das fertige Programm erfüllt letztlich alle Anforderung, indem es zur Darstellung auf

eine Kombination aus HTML(5), CSS und JavaScript setzt, welches viele Möglichkeiten zur Bild- und Videodarstellung als auch zur Steuerung von Haus aus mit sich bringt.

### 4.1.1. Aufbau

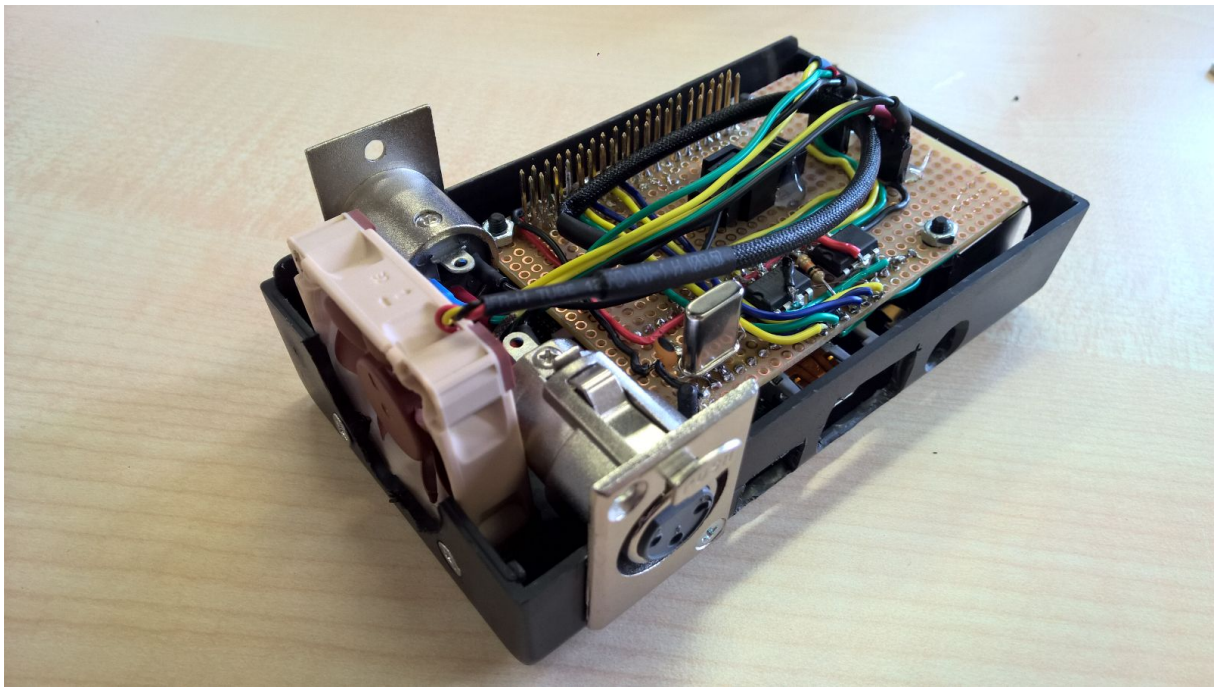

Abbildung A.7.: fertiges Produkt, mit offener Oberschale[1]

Der systematische Aufbau des neuen Medienservers 4.2 unterscheidet sich nicht im Vergleich zu den „großen“. Es gibt die Hardwareebene, welche die Signalumsetzung der Steuersignale, sofern vorhanden, übernimmt, und die Softwareebene, welche das ausgegebene Videosignal generiert. Diese Struktur findet sich auch in der Eigenentwicklung wieder (vgl. A.8).

[1] Quelle: Axel Lodyga 2016

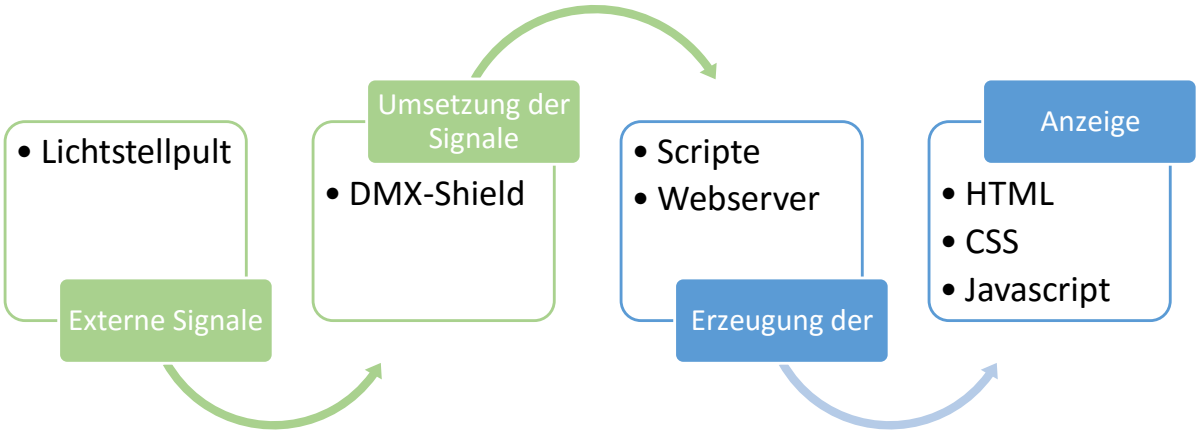


Abbildung A.8.: Systemaufbau : grün: Hardware , blau: Software[2]

Zur Videoausgabe wird beim Raspberry Pi der HDMI-Anschluss genutzt, welcher eine Auflösung von bis zu FullHD unterstützt. Auch der Ton kann über den Anschluss ausgegeben werden. Zusätzlich besitzt der Pi noch einen 3,5mm Klinkenanschluss. Zur Generierung der Anzeige wird eine angepasste Vollbildversion, ein sogenanntes Kiosk-Browsing, des quelloffenen „Chromium Browser“ benutzt. Dieser besitzt im Vergleich die besten Voraussetzungen bezüglich der Leistungsausschöpfung des SBC ( siehe Tabelle A.6). Die Darstellung wird mithilfe von Cascading-Style-Sheets (CSS) definiert. Das CSS beinhaltet alle Information zur Art und Darstellung der einzelnen Elemente in einer deskriptiven Form. Mit Hilfe von JavaScript-Befehlen kann während der Darstellung einer Webseite der Inhalt des CSS geändert werden, ohne dabei die Seite neu laden zu müssen. Unter Zuhilfenahme dieser Möglichkeit lassen sich vollkommen dynamische Seiten gestalten. Das Auslösen der Änderungen erfolgt über eine Web-Schnittstelle namens „Websockets“. Über diese Schnittstelle werden die benötigten Informationen zum JavaScript übertragen, welches die Anpassungen im CSS auslöst. Diese direkten Änderungsmöglichkeiten beziehen sich nicht nur auf das CSS, sondern auch die zugrundeliegende HTML-Seite kann direkt angepasst werden. Diese Möglichkeiten legen die Grundlage, um zum Beispiel in einer Ebene eine Quelle zu wechseln. Auch das Hinzufügen und Löschen von Layern wird so möglich. Die Informationen, welche durch die Websockets an die Webseite gesendet werden, kommen von einem Python-Skript, welches die DMX-Daten von dem selbst konstruierten DMX-Shield erhält. Die DMX-Platine ist dabei so konstruiert, dass sie ohne Ausbau aus dem fertigen Gehäuse mit einer neuen Firmware bespielt werden kann. Das hat zur Folge, dass Optimierungen und Erweiterungen der vorhandenen Hardware ohne Weiteres möglich

[2]Quelle: Axel Lodyga 2016

sind.

## 4.2. DMX-Shield

DMX ist ein differentielles Signal, welches nach DIN-56930-2 über einem 5-POL XLR-Stecker (vgl. feiner lichttechnik (2016)) (häufig wird auf einen 3-Pol zurückgegriffen, da die Steckverbinder und Kabel kostengünstiger sind) übertragen wird.
Da ein normaler Computer jedoch nicht über eine solche Schnittstelle verfügt, muss eine Verarbeitung des Signals über eine separate Schnittstelle erfolgen. Dies lässt sich auf mehreren Wegen lösen.
Zum einen könnte man ein USB DMX-Adapter anschließen, welcher sich entweder über Python oder über eine Zusatzsoftware wie OLA (vgl. (Open-Lighting-Architecture 2016)), ansprechen lässt. Oder man steuert den Differenziell/Seriell-Wandler über die seriellen Anschlüsse des SBCs an, welche dann das differentielle DMX Signal generieren. Aufgrund der fehlenden Implementierung der Empfangsrichtung für DMX über den Wandler in der zusätzlichen Applikation OLA lässt sich so leider nur ein DMX Signal generieren jedoch nicht eingehend verarbeiten. Zu guter Letzt wäre noch auf die bereits erwähnten DMX-Shields, also Erweiterungskarten zur Ergänzung eines DMX-Anschlusses, hinzuweisen. Es gibt bereits im freien Handel fertige DMX-Aufsteckplatinen (vgl. Bitwizard (2016)), welche für den Raspberry Pi entwickelt wurden. Jedoch stellte sich im Test heraus, dass diese als DMX-Interface für das Projekt nicht geeignet sind, da diese Zusatzplatinen nach eben beschriebenen Schema der Variante mit den Differenziell/Seriell-Wandler arbeitet.

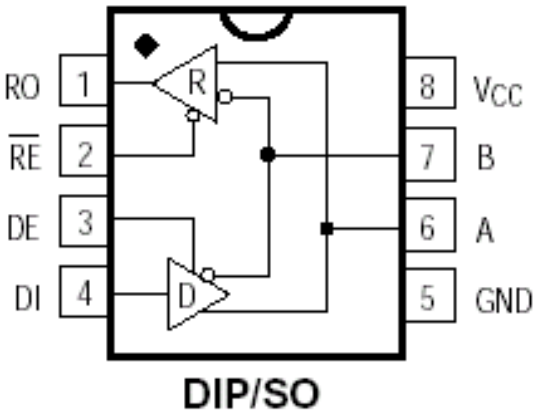


Abbildung A.9.: Schaltbild MAX485 Seriell/Differentiell Wandler[3]

[3]Quelle: http://www.futurlec.com/Maxim/MAX485.gif Abruf: 09.08.2016

Der eingesetzte Wandler, meist ein MAX485-Chip (vgl. Maxim-Integrated (2016)) ermöglicht es, durch Anlegen einer Spannung an zwei der vorhandenen Pins die Datenflussrichtung zu definieren. Der Chip (siehe Abbildung A.9) erwartet bei anliegender Spannung auf den Pins ( „RE“, „DE“) ein serielles Signal auf dem Pin „DI“. Dieses liegt dann umgewandelt als differentielles Signal auf den Pins „A“ und „B“ an. Werden die Stellungsanschlüsse („RE“, „DE“) mit Masse verbunden, so lässt sich umgekehrt ein an „A“ und „B“ vorhandenes Signal an dem Pin „RO“ als serielles Signal empfangen. Jedoch ist eine simultane Umsetzung der Signale in beide Richtungen mit einem einzelnen Baustein nicht möglich.
Es gibt zwar Firmware- und Betriebssystemvarianten für die Zusatzplatine von Bitwizard, welche auch den Empfang bzw. das Auslesen des DMX-Signals ermöglichen, jedoch lässt sich auf diesem System keine andere Software installieren oder um weitere Funktionen ergänzen (vgl. DMX (2015)). Somit blieb nur eine Eigenentwicklung des DMX-Interfaces.

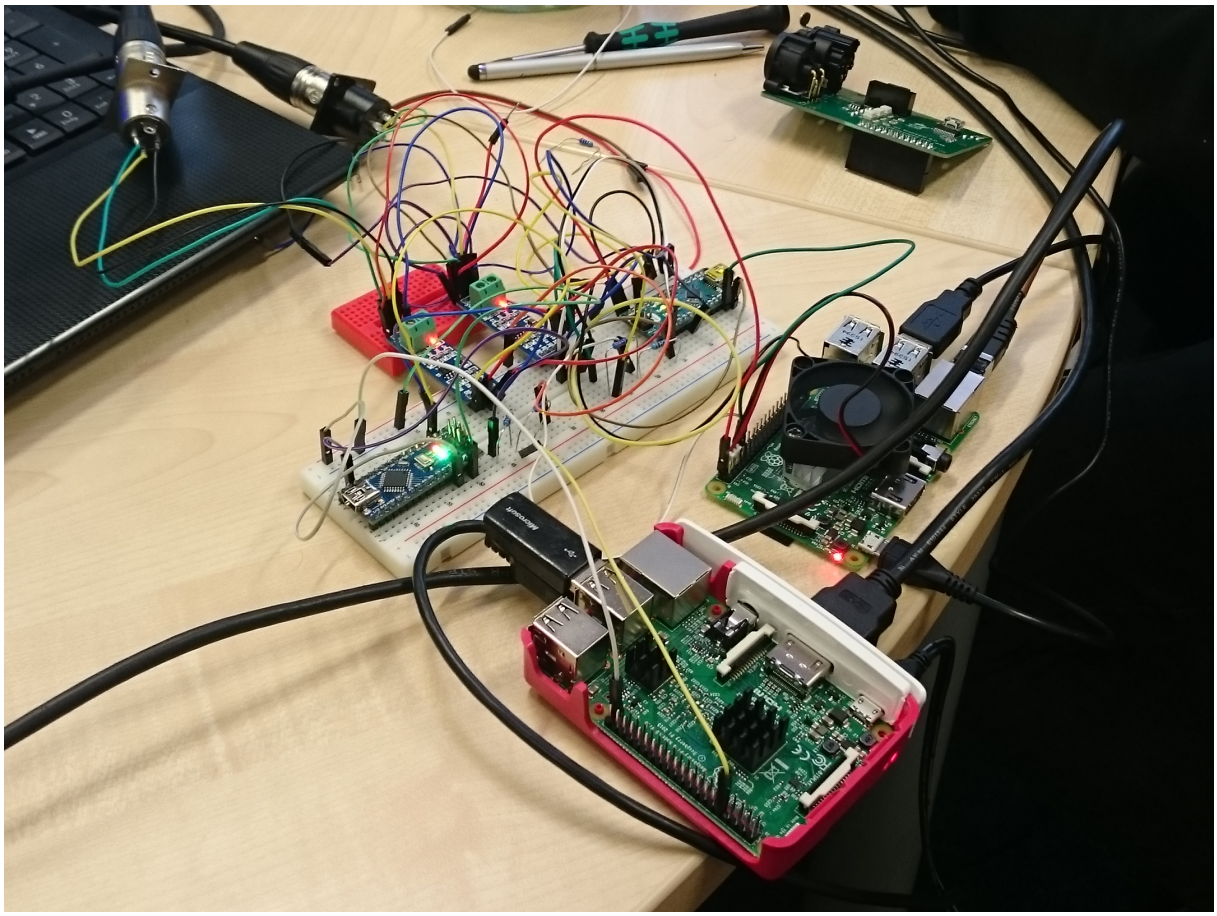

Abbildung A.10.: DMX-Shield : Testaufbau mit zwei Arduino Nano[4]

[4]Quelle: Axel Lodyga 2016

Zunächst wurde die Erweiterungsplatine auf Arduino-Nano (vgl. Arduino (2016c)) (siehe Abbildung A.10), bzw. später Arduino-Mega-Basis (vgl. Arduino (2016b)) aufgebaut, da es für die Arduino-IDE (vgl. Arduino (2016d)) mehrere Bibliotheken, Conceptinetics (vgl. Conceptinetics (2016)) und SimpleDMX (vgl. tinker.it (2016)) gibt, welche das Handling von DMX auf einem Arduino unter Zuhilfenahme von MAX485-Bausteinen implementiert. Ein Arduino ist eine Platine zur Hardwareentwicklung auf Basis eines 8-Bit Mikrocontrollers mit einer dazugehörigen Entwicklungsumgebung (der IDE), welche ohne größere Kenntnisse von Assembler oder C-Programmierung die Konzipierung und Programmierung von Hardware-Projekten ermöglicht. Dabei ist der Einstieg von blinkenden LEDs )(vgl. Arduino (2016a)) bis hin zur sehr komplexen Steuerung von autonomen Drohnen (vgl. MultiWii (2016)) möglich.
Auf Grundlage der DMX-Bibliotheken wurde ein Programm geschrieben, welches dem Mikrocontroller ermöglicht, ein serielles Signal in ein DMX Signal zu wandeln, und das simultan in Empfangs- als auch Senderichtung. Die Software ermöglicht es auch, ein komplettes DMX-Universum abzubilden, jedoch beeinträchtigt die Anzahl der Kanäle den Datendurchsatz. Dadurch findet eine Verzögerung im Programmablauf statt, wodurch die Ausführung in beide Richtungen einen Zeitversatz aufweisen kann.
Die Geschwindigkeit, ein komplettes DMX-512 Signal nach Norm 44,1 pro Sekunde übertragen zu können, ist durch die Begrenzung der Datenrate auf UART Seite nicht möglich. Die maximale Baudrate, welche der Raspberry Pi, oder der BananaPi ausnutzen können, liegt jedoch nur bei 11.520 Bytes pro Sekunde. Dabei könnten sowohl die Differenziell/Seriell-Wandler *„2,5Mbit/sec “*(Maxim-Integrated 2016) als auch der verwendete Microcontroller (Atmel 2016: 186) eine Baudrate von 2,5Mbit mit geringer Fehlertoleranz umgehen.

$$Rate_{DMX} = 8 * 512 * 44,1 = ca.180.000 Bit/Sek$$
$$Rate_{UART} = 115.200 Bit/Sek$$
$$Rate_{möglich} = 250.000 Bit/Sek$$

Somit ist der limitierende Faktor in der Übertragungsgeschwindigkeit der SingleBoard-Computer. Bei Betrachtung der Baudraten ist die Differenz zwischen nötiger Übertragungsrate und praktisch nutzbarer bei ca. 65.000 Bit/Sek. Der Verlust im Bereich der Wiederholungsrate liegt demnach bei ca. ein Drittel und das bei der Nutzung aller DMX-Kanäle. Also werden alle 512 Kanäle nicht 44-mal sondern nur 25-mal pro Sekunde übertragen, was immer noch „Echtzeit“ im Sinne einer Eingabeverzögerung eines ein-

zelnen Videoframes, beziehungsweise einer 25tel Sekunde entspricht. Würde man die Anzahl der Kanäle auf 332 reduziert, so hätte man auch wieder die Möglichkeit, diese 332 Datenpakete 44-mal pro Sekunde zu versenden.

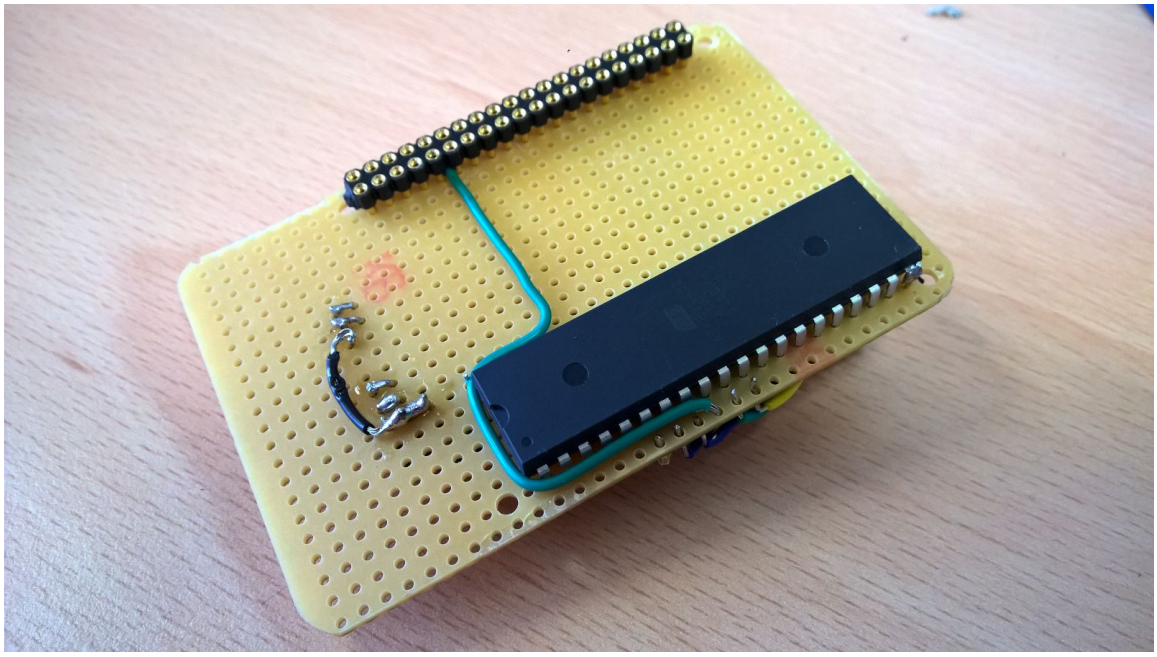

Abbildung A.11.: DMX-Shield : Fertige Aufsteckplatine - Unterseite mit Atmega162[5]

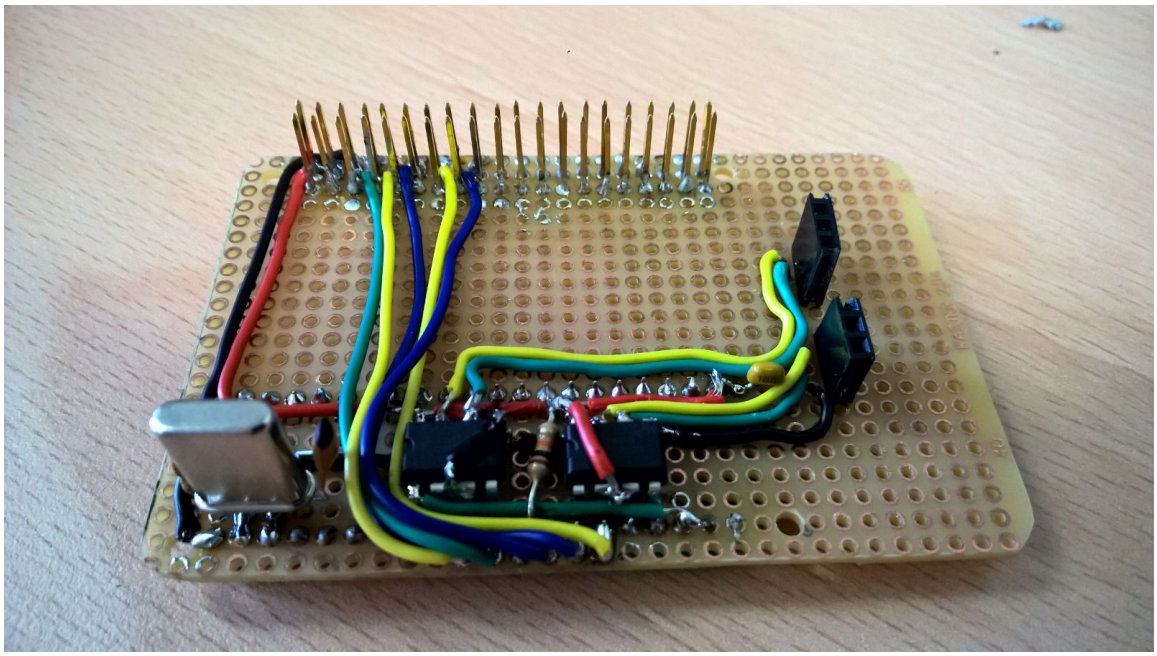

Abbildung A.12.: DMX-Shield : Fertige Aufsteckplatine - Oberseite mit MAX485 Chips und Anschlüssen für XLR Stecker[6]

[5]Quelle: Axel Lodyga 2016
[6]Quelle: Axel Lodyga 2016

### 4.2.1. „Betriebssystem“ ATmega162

Das Herzstück, welches dem Pi die Kommunikation über DMX ermöglicht, ist der ATmega162 in Verbindung mit zwei MAX485 Chips (siehe Abbildungen A.11 & A.12). Dabei übernimmt der Mikrocontroller die Übersetzung der unterschiedlichen Signale. Zunächst wurden die Schaltung und der zugehörige Programmcode mit einem Arduino Nano umgesetzt. Der Nano besitzt jedoch nur eine serielle Schnittstelle, welche allerdings durch die Conceptinetics-Bibliothek besetzt ist und man auf eine simulierte serielle Software-Schnittstelle angewiesen wäre. Diese ist jedoch nur bis zu einer Baudrate von 9.600 wirklich zuverlässig.
Um einen zweiten hardwareseitigen Kommunikationsanschluss zu haben, muss man, wenn man sich bei den Arduino-Produkten umsieht, auf den Arduino Mega 2560 zurückgreifen. Dieser besitzt sogar gleich vier Anschlussmöglichkeiten zur seriellen Kommunikation (vgl. Arduino (2016b)). Der verwendete Mikrocontroller ist, wie der Name schon vermuten lässt, ein ATmega2560, produziert von der Firma Atmel. Leider ist eine DIP-Variante, welche sich einfacher verlöten lässt, bei der Anschlusszahl des Bauteils nicht verfügbar. Aufgrund dessen wurde eine „abgespeckte“ Variante dieses Bausteins, der ATmega162, für das Projekt verwendet.
Das Bauteil wurde deshalb ausgewählt, weil er der „kleinste“ ATmega ist, welcher über die zwei benötigten seriellen Schnittstellen verfügt (vgl. (Atmel 2016: 165)) und sich zudem die Arduino-IDE integrieren lässt (vgl. Silvius (2013)). Eine der Schnittstellen wird von der Bibliothek „Conceptionetics“ benutzt, während die zweite zur Kommunikation mit dem Raspberry Pi dient.
Nachfolgend wird der Programmablauf des Mikrocontrollers betrachtet:

```
1  #include <Conceptinetics.h>
2  #define SERIAL_TX_BUFFER_SIZE 16
3  #define SERIAL_RX_BUFFER_SIZE 16
4  int number_of_channels = 512;
5  int channels_with_vals[512]={0};
6  #define INPUT_SIZE ((number_of_channels*3)+number_of_channels)
7  #define DMX_SLAVE_CHANNELS   number_of_channels
8  #define DMX_MASTER_CHANNELS  number_of_channels
9  #define RXEN_PIN                20
10 DMX_Master        dmx_master ( DMX_MASTER_CHANNELS, RXEN_PIN );
11 DMX_Slave dmx_slave ( DMX_SLAVE_CHANNELS );
12
13 void setup() {
14         dmx_slave.enable ();
15         dmx_slave.setStartAddress (1);
16         dmx_master.enable ();
17         dmx_master.setChannelRange ( 1, 5, 127 );
18         Serial.begin(115200,0x86);
19         Serial.println("HY PI I'm the DMX_Serial Handler Version 4");
20         dmx_master.setChannelRange ( 2, 5, 255 );
21         dmx_master.setChannelValue ( 2,0 );
22         dmx_master.setChannelValue ( 4,0 );
23         dmx_master.setChannelRange ( 1, 10, 0 );
24 }
```

Code C.1: ATmega162 Firmware: Initialisierungsphase

Noch vor der eigentlichen Initialisierungsphase werden diverse Parameter für den Programmablauf definiert. So wird in Zeile eins zunächst die Conceptionetics-Bibliothek eingebunden, während in den darauf folgenden Zeilen 7-11 deren Parameter festgelegt werden. In Zeile sieben wird die Anzahl der zu betrachtenden DMX-Kanäle definiert, während die nachfolgende Zeile ein Array zum Zwischenspeichern der zu sendenden DMX-Werte mit 0 initialisiert. In der Zeile sechs wird ein Bereich im Speicherregister des Mikrocontrollers definiert und geblockt, in welchem die zu empfangenden Daten des Raspberry Pi zwischengespeichert und verarbeitet werden. Dabei ist die Speichergröße abhängig von der Anzahl der zu behandelnden Kanäle.
Die Funktion „setup“ (siehe CodeC.1) wird immer einmalig ausgeführt, wenn der Mikrocontroller startet oder zurückgesetzt wird. Dabei wird der Startwert des DMX-Empfanges gesetzt, aber auch auf den Kanälen eins bis fünf ein „Blinken“ erzeugt, welches das

Initialisieren des Controllers signalisieren soll. Zudem wird in Richtung des Raspberry Pi's eine Startbotschaft gesendet, um die UART-Kommunikation zu initiieren und zu testen.
Nach der Initialisierung springt das Programm in eine Dauerschleife, in welcher der Prozessor entscheidet ob er von DMX zu Seriell (receiving) oder in die entgegengesetzte Richtung übersetzten soll. Dabei überprüft der Code, ob am seriellen Anschluss 1 ein Datenfluss anliegt oder nicht (siehe Code C.2).

```
1 void loop() {
2         if (Serial.available() > 0) {
3                 read_values();
4         }
5         else {
6                 receiving();
7         }
8 }
```

Code C.2: ATmega162 Firmware: Hauptprogrammschleife

Sollte dies nicht der Fall sein, so wird die Routine „receiving“ ausgeführt:
In dieser Funktion wird wieder eine Schleife durchlaufen, in welcher jeder einzelne Kanal auf seinen Wert hin überprüft wird. Dieser ausgelesene Zustand wird als Zahl (0-255) auf die den Serial(1)-Anschluss geschrieben und vom nächsten Wert mit einem Komma getrennt. Sobald die Schleife nun alle Kanäle abgefragt hat, wird ein „Neue-Zeile-Zeichen“ (auch \n) angefügt und der Programmablauf springt wieder in die Hauptschleife (siehe Code C.2).
Nach demselben Schema läuft auch die Senderichtung, und zwar mit dem Unterschied, dass diese Funktion aufgeteilt ist in einen Werte-Auslesen-Teil und dem eigentlichen Werte auf DMX-Senden-Teil.
Dabei wird zunächst ein Zwischenspeicher für die Zeichen, welche der Raspberry Pi an den Mikrocontroller sendet, angelegt. Dieser wird dann mit den empfangenen Daten gefüllt und eine abschließende 0 angefügt. Anschließend wird nach dem nächsten Trennzeichen, in diesem Fall das „Et-Zeichen“ (umgangssprachlich „Kaufmanns-Und“ genannt), gesucht und der Zeichenkettenabschnitt wird unter Vernachlässigung des „&“ in eine Zahl umgewandelt. Diese Zahl wird nun in ein Register unter der Indizierung einer durchlaufenden und immer mit inkrementierender ID gespeichert. Das Suchen und Teilen der empfangenen Daten wird solange wiederholt, bis kein Teilzeichen mehr gefunden wird.
Abschließend wird die für das Übertragen der Kanalinformation zuständige Funktion „sending“ (siehe Code C.4) aufgerufen . In dieser wird das eben gespeicherte Array

```
void read_values() {
        char input[INPUT_SIZE + 1];
        byte size = Serial.readBytes(input, INPUT_SIZE);
        input[size] = 0;
        int channelid = 1;
        char* command = strtok(input,"&");
        while (command != 0){
                int value = atoi(command);
                channels_with_vals[channelid] =  value;
                command = strtok(0,"&");
                channelid++;
        }
        sending();
}
```

Code C.3: ATmega162 Firmware: Auslesen des seriellen Inputs

```
void sending() {
        for (int i = 1; i <= number_of_channels; i++) {
                dmx_master.setChannelValue(i,
                                channels_with_vals[i]);
        }
}
```

Code C.4: ATmega162 Firmware: Senden der DMX-Daten

aufgerufen und alle Werte nach und nach als DMX-Information ausgegeben. Abschließend wird wieder in die Hauptschleife (siehe Code C.2) gewechselt und das Programm beginnt von vorn. So durchläuft der Mikrochip einen Dauerzyklus von Empfangen und Senden von Daten, welche die Software auf dem Raspberry Pi interpretieren bzw. schreiben kann.

## 4.3. Software

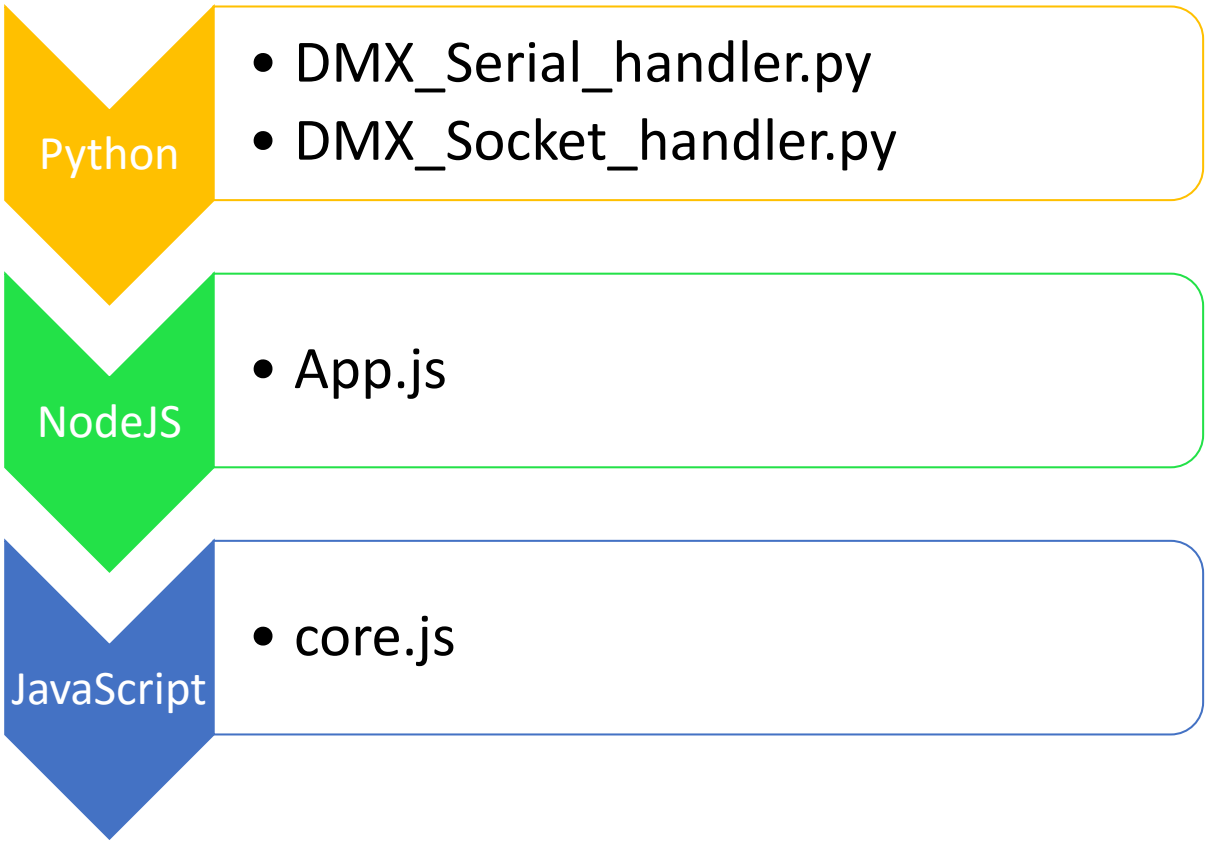


Abbildung A.13.: Software : einzelne Softwarekomponenten[7]

Um auch die Abläufe im Raspberry Pi (siehe Abbildung A.13) zu verstehen, muss man zunächst die Struktur der verwendeten Programme verstehen. Die Software ist aufgeteilt in drei Hauptkomponenten, wovon zwei in JavaScript geschrieben sind und eine in Python umgesetzt wird.
Der Python-Teil übernimmt das Umsetzten der Steuersignale in ein Format, womit sich Webscripte bzw. Webserver steuern lassen. Um die Signale des DMX-Shields zu interpretieren, wird die serielle Schnittstelle des Raspberry Pi von einem in Python geschriebenen Programm genutzt. Dieses wertet die ankommenden Daten aus und die daraus resultierenden Steuerungsdaten werden per Websocket an den Webserver weitergegeben. Dieser gibt sie unbehandelt an den Client, also den Chromium-Browser, weiter. Das dort laufende JavaScriptscript führt die Darstellungsänderungen durch. Um auch DMX-Signale senden zu können, kann der Weg auch gegenläufig betrachtet werden, wodurch die Bidirektionalität geben ist (siehe Abbildung A.14).

[7] Quelle: Axel Lodyga 2016

### 4.3.1. Python

Der Programmteil, welcher in der Programmiersprache Python geschrieben ist, gliedert sich wiederum in zwei Bestandteile. Dabei ist der DMX_data_handler für die Handhabung der seriellen Schnittelle auf dem Raspberry Pi zuständig. Zunächst wird der Anschluss initialisiert und parametrisiert (siehe Code C.5). Dabei wird auch die maximale Baudrate von 115.200 festgelegt und in welcher Art und Weise das Signal behandelt wird; also, ob ein oder zwei Stopbits benutzt werden und ob eine Paritätsprüfung stattfindet.

```
1         import serial
2         import time
3         from conf import *
4         from threading import Thread
5
6         #Initializing
7         channel_buffer = range(0, 128)
8         for x in xrange(0, 128):
9         channel_buffer[x] = 0
10
11        last_received = ''
12
13        ser = serial.Serial(
14        port='/dev/ttyAMA0',
15        baudrate=115200,
16        parity=serial.PARITY_NONE,
17        stopbits=serial.STOPBITS_ONE,
18        bytesize=serial.EIGHTBITS,
19        timeout=1,
20        )
21
22        ...
```

Code C.5: DMX_data_handler.py: Empfängt und generiert die DMX relevanten Daten - Initialisierung

Anschließend werden weitere Funktionen zum Senden der DMX-Daten und zur Generierung der benötigten Zeichenketten definiert. Um den Medienserver jedoch extern steuerbar zu machen, muss die Empfangsrichtung angegeben werden.
Dazu werden zwei Funktionen definiert: Erstere wird durch die den Socket-Handler

aufgerufen (siehe Code C.12), um einen neuen DMX-Datensatz abzufragen, während zweitere für die Erzeugung dieses Datensatzes zuständig ist. Hierfür wird ein eigener Prozess geöffnet (siehe Code C.6:20), welcher den eingehenden Datenstrom überwacht. Da der ATmega162 nach einem kompletten Informationspaket eine neue Zeile schreibt, speichert die Funktion „receiving“ des DMX_data_handler (siehe Code C.6:9) immer die letzten zwei empfangen Zeilen in einen Zwischenspeicher. Durch die Auslagerung in einen separaten CPU-Prozess und die Zwischenspeicherung zweier Zeilen, ist eine echtzeitfähige Kontinuität gewährleistet.

```
1  ...
2
3  def read_serial_data():
4          "Reads the incomming Serial DATA"
5          values = last_received
6          dmx = values.split(',')
7          return dmx
8
9  def receiving(seri):
10         global last_received
11         buffer_string = ''
12         while True:
13                 buffer_string = buffer_string +
                    ↪  seri.read(seri.inWaiting())
14                 if '\n' in buffer_string:
15                         lines = buffer_string.split('\n')
16                         last_received = lines[-2]
17                         buffer_string = lines[-1]
18
19
20 Thread(target=receiving, args=(ser)).start()
```

Code C.6: DMX_data_handler.py: Empfängt und generiert die DMX relevanten Daten - Empfang

Die gesamten Steuersignale werden nachfolgend in einem separaten Python-Programm namens „DMX_socket_handler“ verarbeitet. Die Zuordnung der einzelnen DMX-Kanäle zu ihren Funktionalitäten wird im ersten Schritt nach der Umgebungs-Konfiguration ausgeführt. Aufgrund der Verwendung eines Schlüssel-Werte-Paar-Systems können in allen folgenden Funktionen die Schlüssel aufgerufen werden, ohne den zugeordneten Kanal kennen zu müssen. Das hat zur Folge, dass sich auch die DMX-Konfiguration

ändern lässt. Hierfür ist es lediglich nötig, eine neue Zuweisung der Paare zu veranlassen.

```
1        ...
2        global settings
3        settings={
4        'rotation':6,
5        'selector':0,
6        'source':1,
7        'x':2,
8        'y':3,
9        'size_x':4,
10       'size_y':5,
11       'boarders':9,
12       'opacity':8,
13       'zrotation':7,
14       'playspeed':10,
15       };
16       ...
```

Code C.7: DMX_socket_handler.py: behandelt die DMX-Daten und generiert die Websocket-Informationen - Zuordnung

Die Erzeugung der für die Veranlassung der Inhaltsänderungen notwendigen Websocket-Informationen erfolgt in der Funktion „get_channel_values“ (C.8). Sofern die zugrundeliegenden DMX-Informationen nicht denen der zuvor abgehandelten explizit gleicht, wird eine Iteration über das Schlüssel-Werte-Paar-Konstrukt der Kanalzuordnung gestartet. Hierbei wird jedes einzelne Element auf eine Statusänderung hin überprüft. Im Falle einer Änderung wird ein Websocket-Event ausgelöst (siehe Code C.8:8). Zur Identifikation trägt das Event den Namen der zugeordneten Medienserver-Funktionalität. So sieht zum Beispiel das Element zur Änderung der X-Position so aus:

*{’X’ : ’127’}*

Diese Informationen nimmt nun zunächst der NodeJS-Server entgegen, bevor die Daten die eigentliche Änderung auslösen können. Zu guter Letzt wird der aktualisierte Status der DMX-Informationen als letzter veränderter Stand abgespeichert, um beim nächsten Funktionsaufruf zum Vergleich auf Veränderungen zu dienen.

```
1         ...
2         def get_channel_values():
3         global last_channels
4         dmx_data = read_serial_data()
5         if last_channels != dmx_data :
6                 for (channel, title) in settings.items():
7                         if last_channels[channel] != dmx_data[channel] :
8                         socketIO.emit(title,dmx_data[channel])
9                 last_channels = dmx_data;
10        ...
```

Code C.8: DMX_socket_handler.py: Behandelt die DMX-Daten und generiert die Websocket Informationen

### 4.3.2. Javascript

Um zu verstehen, wie der JavaScript-Code den angezeigten Bildschirminhalt manipulieren kann, muss man zunächst den Aufbau der HTML-Seite anschauen. Im Kopfbereich des Quellcodes wird die Seite mit allen nötigen zusätzlichen Dateien verknüpft (siehe Code C.9 :4-8). Im eigentlichen Körper sind, bis auf ein Audio-Element, lediglich leere Abschnitte definiert. Dieses Audioelement spielt eine „leere“ MP3-Datei in einer Dauerschleife ab, was dazu dient, den Audioausgangs des Raspberry Pi zu starten, da es bei manchen Projektoren zu einer Bildneuinitialisierung kommt, sobald ein Audiosignal über HDMI dazugeschaltet wird. Das würde bei einem Wechsel zwischen Inhalten ohne Tonspur zu einem Inhalt mit Ton zu einer neu Initialisierung der Projektorquelle führen und somit auch zu einem Bild- und Tonausfall von mindestens einer Sekunde. Durch die „lautlose“ MP3 wird dieses Problem umgangen.
Die sonstigen Abschnitte, auch DIV genannt, legen die Grundlage, um Inhalte zu definieren und durch den JavaScript-Code und das CSS ansprechbar zu werden, da beide Codes auf diese DIVs referenziert. Diese haben einen festen Namen, auch ID genannt, um sie als einzelne Layer zu deklarieren. Zudem sind sie mit grundlegenden CSS-Attributen versehen, welche zu einer gezwungenen Hardwarebeschleunigung dienen (siehe Code C.9 :13-15). In den DIVs wird über JavaScript die Quelle platziert. Das Bild oder Video, welches angezeigt werden soll, liegt also in dem jeweiligen DIV.

```
<DOCTYPE HTML>
<html>
<head>
        <title>RasViMap</title>
        <script type="text/javascript"
        ↪ src="/socket.io/socket.io.js"></script>
        <script type="text/javascript"
        ↪ src="./assets/js/new_core.js"></script>
        <script type="text/javascript"
        ↪ src="./assets/js/jquery-3.0.0.min.js"></script>
        <link rel="stylesheet" href="/assets/css/index.css">
</head>
<body onload="initialize()" style="overflow:hidden;">
        <audio autoplay loop><source
        ↪ src='./assets/sounds/blank.mp3'/></audio>

        <div id='layer_0' class='layer' style="transform:
        ↪ translateZ(0);will-change:
        ↪ transform,opacity,border-radius;"></div>
        <div id='layer_1' class='layer' style="transform:
        ↪ translateZ(10);will-change:
        ↪ transform,opacity,border-radius;"></div>
        <div id='layer_2' class='layer' style="transform:
        ↪ translateZ(20);will-change:
        ↪ transform,opacity,border-radius;"></div>
</body>
```

Code C.9: HTML Seite zur Darstellung der Inhalte

Um diese Seite überhaupt im Browser aufrufen und anzeigen zu können, benötigt man einen Webserver. In diesem Fall wird NodeJS unter Verwendung der Pakete „Express“ (vgl. ExpressJs) und „SocketIO“ (vgl. Socket.IO (2016)) als ein solcher Webserver eingesetzt. Er ermöglicht die Verwaltung und Distribution der Quell- und Script-Dateien als auch die Verwendung der Websockets, da eine direkte Kommunikation der in Python generierten Websockets zur Gegenstelle, der Webseite, nicht möglich ist. Dabei übernehmen nach der Einrichtung und Initialisierung (siehe Code C.10) drei Funktionen die eigentliche Arbeit des Abhandelns restlicher Funktionalitäten.

```
var PORT = 3000;
var express = require('express');
defaults = {
        /*Channels*/
        /*1*/         selector:0,
        /*2*/         source:1,
        /*3*/         x:2,
        /*4*/         y:3,
        /*5*/         size_x:4,
        /*6*/         size_y:5,
        /*7*/         rotation:6,
        /*8*/         opacity:7,
        /*9*/         boarders:8,
        /*10*/         playspeed:9,
};
settings = defaults;
var onevent = socket.onevent;
socket.onevent = function (packet) {
        var args = packet.data || [];
        onevent.call (this, packet);
        packet.data = ["*"].concat(args);
        onevent.call(this, packet);
};
var app = express();
var http = require('http')
var server = http.createServer(app);
app.use(express.static(__dirname + '/public/'));
server.listen(PORT);
var io = require('socket.io').listen(server);
        ...
```

Code C.10: NodeJS Server - Initialisierung

Diese Einrichtung beschreibt zum Beispiel auch die Standardzuordnung der DMX-Kanäle zu den einzelnen Funktionen, wie es auch in Python geschieht (C.7). Des Weiteren wird der vom Webserver zu benutzende Port und der Ordner angeben, in welchem sich die HTML-Daten befinden. Lediglich die Initialisierung der Websockets folgt, nachdem der Prozess des Express-Paketes gestartet wird.

```
io.sockets.on("connection",function(socket){
socket.emit('settings',settings);
```

Code C.11: NodeJS Server - Connection

Im Code (siehe Code C.12) sind die Websocket-Funktionen aufgeführt, welche die einzelnen Websockets abhandeln. Erste leitet lediglich alle Websocket-Pakete weiter, welche ankommen. Das kann von der Admin-Seite her sein und betrifft auch alle Pakete, die vom Python-Skript aus gesendet werden. Dabei sind die Rezipienten immer alle die mit dem Server verbundenen Schnittstellen. Die zweite Funktion dient lediglich dem Zurücksetzten der DMX-Kanal Zuordnung auf die zuvor in der app.js (siehe Code C.10) definierten Standardeinstellung. Die dritte und letzte Funktion des Servers sendet alle 50 Millisekunden ein inhaltsloses Websocketpaket mit dem Namen „get_channel_values“. Dieses Paket löst im Python Programm die Abfrage nach neuen DMX Werten aus (siehe Code C.6).

```
socket.on("*",function(event,data) {
        socket.broadcast.emit(event,data)
});
socket.on("defaults",function(socket){
        settings = defaults;
        socket.emit('settings',settings);
        socket.broadcast.emit('settings',settings)
});
setInterval( function(){
        socket.emit('get_channel_values');
},50);
```

Code C.12: NodeJS Server - Funktionen

Die erste Zeile des HTML-Body (siehe Code C.9:10), also der Hauptteil der angezeigten HTML-Seite, startet die nötigen Funktionen in der Javascript Datei „core.js“. Wieder mit einer Initialisierungsphase beginnend, wird anschließend eine Websocket-Verbindung mit dem NodeJS-Server aufgebaut. Als Antwort auf eine erfolgreich hergestellte Verbindung sendet der Server die aktuellen Einstellung der Kanalbelegung (siehe Code C.11). Dies ist für die darstellende Webseite zwar nicht von Relevanz, jedoch zum Beispiel für die administrative Seite, in der die Kanalbelegung geändert werden soll und sie dabei nach den gleichen Kommunikationsschemata mit Websockets arbeitet.

Die „core.js“-Datei ist mit Herstellen der Verbindung ansprechbar für Websocket-Pakete vom Server, also auch für diejenigen, die vom Python-Skript kommen (siehe Code C.8), und hat für jedes ankommende Paket eine spezielle Funktion, welche Veränderungen in der HTML-Seite auslöst.
Es wird nach dem Prinzip der „Layerselection“ (siehe Code 2.2.2) gearbeitet. Also wird zuerst überprüft, welche Ebene aktuell ausgewählt ist und angesprochen werden soll. Alle ankommenden Befehle und Daten, die eine Änderung hervorrufen sollen, werden so lange an der Ebene abgehandelt, bis durch die Änderung des „Selectors-Kanals“ (siehe Code C.7:5) ein anderer Layer ausgewählt wird.
Da jede Kanaländerung ein bestimmtes Websocket-Event auslöst, gibt es für jedes Event eine definierte Funktion, welche eine Manipulation der HTML-Seite zur Folge hat. Nachfolgend wird zum Beispiel die Rotation eines Layers geändert. Dazu wird das Transform-CSS-Attribut ergänzt, um das „rotateY-Element“. Dies beschreibt die Rotation eines Objektes um die Y-Achse. Es lassen sich sowohl der Rotationsursprung, welcher im Beispiel nicht angesprochen wird, als auch der Winkel definieren.

```
1        ...
2        socket.on('rotation',function(dmx_value){
3        layers.Transform({ rotateY: ((dmx_value/255.00)*360) });
4        });
5        ...
```

Code C.13: Beispiel eines Socket-Empfängers in der core.js Datei

Der Übertragene DMX-Wert (0-255) wird nun auf 0-360° umgerechnet und die ausgewählte Ebene lässt sich so um maximal 360° drehen.
Da das CSS fast alles ermöglicht, was man benötigt, lassen sich ohne große Programmierkenntnisse viele Funktion in den Medienserver implementieren. Auch 3D-Verformungen lassen sich so einfach implementieren, ohne die Rechenleistung des Raspberry Pi groß in Anspruch zu nehmen (vgl. Three.js (2015)). Laut Silvia Pfeiffer lässt sich auch ein Video-Element in Bezug auf Abspielgeschwindigkeit und Laufrichtung beeinflussen.

*„If @playbackRate is positive or 0, then the direction of playback is forward. If it’s negative, the direction is backward. .“* (Pfeiffer 2010: 104)

Leider funktioniert das Attribut in etlichen Tests nur mit positiven Werten, welche einen linearen Einfluss auf die Wiedergabegeschwindigkeit in positiver Richtung zeigten.

# 5. Ergebnisse

## 5.1. Chancen und Grenzen

### 5.1.1. Leistung und Hardware

Die bisherige Herangehensweise zeigt, dass es möglich ist, auf einem SBC einen kleinen Medienserver zu betreiben. Die größte Hürde besteht im Moment darin, die Limitierung der Videoauflösung der einzelnen Layer zu erhöhen. Bisher sind alle Quellen H.264-codierte Videos mit einer Auflösung von 360p. Dabei ist die Anzahl der Ebenen zunächst auf vier Stück begrenzt worden. Unter diesen gegeben Umständen läuft die Darstellung mit allen Änderungsmöglichkeiten der Parameter flüssig. Aber noch längst sind nicht alle Möglichkeiten, die das gegebene Umfeld bezüglich CSS und JavaScript bietet, ausgenutzt worden. So lassen sich neben den bisherigen Bildmanipulationen zum Beispiel auch Farbfilter und Verzerrungseffekte über CSS implementieren (vgl. Coyier (2015) und Bennet (2015)). Im Vergleich zu einem „großen“ Medienserver wird aber hardwareseitig immer eine Grenze bestehen. Sei es durch die beschränkte Möglichkeit, nicht mehre Monitore anzuschließen zu können, oder durch die Begrenzung bei der Auflösung des Videoausgangs an sich.

Inwieweit neue Versionen der Treiberimplementierung und gegebenenfalls auch neue Betriebssysteme Verbesserungen schaffen, ist reine Spekulation. Auch die Leistungssteigerung des Prozessors bezüglich seiner 64Bit-Fähigkeiten, bleiben ohne die Entwicklung der erforderlichen Systeme und den entsprechenden Anwendungen, durch den Hersteller oder die Nutzergemeinschaft, nicht abzusehen. Da jedoch die Hardwaredekodierung von Videos durch den Grafikchip möglich, jedoch bisher in keinem Browser nativ integriert ist, kann, sobald verfügbar, von einer Leistungssteigerung bei der Videowiedergabe ausgegangen werden.

Eine weitere Möglichkeit die Leistungsfähigkeit der Anzeige zu steigern, liegt im Streaming. Das heißt, man kann die Videoquellen auch über Websockets leiten, sodass die Steuerung bzgl. der Abspielgeschwindigkeit etc. vom NodeJS-Server übernommen wird. Das hat eine Entlastung des Browsers zur Folge. Zudem wäre eine simultane Wiedergabe auf mehreren Endgeräten möglich. Im konkreten Fall hieße es, dass

man einen Raspberry Pi als Master nutzt, welcher die Steuerung und Verwaltung übernimmt, und einen oder mehrere andere Pi's, welche über Wlan oder Ethernetkabel verbunden sind, als Slaves agieren. Diese Slaves erhalten dann die Steuersignale und Quelldaten für die Anzeige der Webseite vom Master. Somit ließen sich zum Beispiel mehrere Projektoren aus verschiedenen Richtungen, zum Beispiel für dreidimensionales Mapping (siehe Abbildung A.1), steuern ohne das erheblichen Mehrkosten entstünden.

### 5.1.2. Erweiterungen und Software

Als weitere Chancen sind auf jeden Fall eine Implementierung einer ArtNet und SACN Schnittstelle durch eine Python Verknüpfung der OLA (vgl. Open-Lighting-Architecture (2016)) anzusehen. Eine eigene Bibliothek hierfür ist in der Software bereits integriert, jedoch im Medienserver nicht implementiert. Durch eine Erweiterung um diese Protokolle würde eine einfachere Steuerung aller Einstellungen und die eigentliche Bedienung durch ein größeres Lichtstellpult, wie die GrandMA2 von MA Lighting (vgl. MA-Lighting (2015)) , möglich sein.

Um diese mehrfache Ansteuerung jedoch umsetzten zu können, muss zum einen die Latenz der Websockets in Betracht gezogen als auch deren Reaktionen verbessert werden. Im Test kam es bei zu vielen simultanen Änderungen der DMX-Kanäle zu Abstürzen, da zum Beispiel beim Herunterziehen aller 10 Fader des Lichtstellpults im „Worst Case“ innerhalb kürzester Zeit (ca. eine Sekunde) 2550 Websockets mit ihren Änderungen ausgelöst werden.

Ein weiterer limitierender Faktor im Bereich der Signalverarbeitung ist die aktuell vom Webserver gesteuerte Abfrage nach einem neuem DMX-Informationsdatensatz. Durch die zeitgesteuerte Taktung von derzeit 50 Millisekunden könnten immer noch Information verloren gehen.

Zu guter Letzt sei noch die Möglichkeit erwähnt, dass die Bedienung auch ohne ein Lichtstellpult durch die Nutzung der Einstellungsseite, d.h. des „Adminpanels“, funktioniert, da die Webseite neben den Einstellungen bezüglich der DMX-Kanalbelegung zudem die aktuellen Kanalwerte, und zwar mit simulierten Fadern und deren Position, anzeigt. Diese Fader lassen sich ebenso wie die Regler eines Lichtstellpultes bedienen, die Änderungen lösen dieselben Websockets wie die DMX-Daten aus. So bedarf es lediglich eines Computers oder Handys, um die Steuerung des Medienservers zu übernehmen.

# 6. Fazit

In der vorliegenden Arbeit sollte untersucht werden, ob und inwieweit es möglich ist, einen SingleBoardComputer als einen günstigen Ersatz für einen in der Event- und Medientechnik benutzten Medienserver einzusetzen. Die grundlegenden Funktionen eines Medienservers, welche sich zusammenfassen lassen als Verwalten, Darstellen und Manipulieren von (bewegten) Bildinhalten, haben sich bei der Betrachtung verschiedener im Markt gebräuchlicher Geräte herauskristallisiert. Diese galt es auf einem einfachen und erweiterbaren System, nämlich auf einem Einplatinen-Computer, umzusetzen. Unter Berücksichtigung der Leistungsfähigkeit und der Grundlage, welche das Betriebssystem vorgibt, wurde eine Möglichkeit für kleinere und nicht allzu komplexe Anwendungen geschaffen. Mit Hilfe eines eigens entwickelten DMX-Shields können nun DMX-Daten empfangen und gesendet werden. Somit lässt sich der Medienserver mit gebräuchlichen Lichtstellpulten steuern und in das Medientechnikumfeld nahtlos integrieren. Ergo wurden die anfangs aufgestellten Kriterien (siehe 4) umgesetzt. Gerade was die Stabilität und die Leistungsfähigkeit bezüglich der Videoauflösung angeht, ist noch „Luft nach oben“. Als „Proof-of-Concept“ zeigt dieses Projekt jedoch, dass es machbar ist, ein anwender- und entwicklerfreundliches Grundsystem zu schaffen, was in seinem Potential noch lange nicht erschöpft ist. Das Ergebnis ist ein Produkt, welches aufgrund des offenen Quellcodes und der damit verbundenen Entwicklungsmöglichkeiten geeignet ist, von anderen Personen als Alternative zu den herkömmlichen und teuren Produkten gewählt zu werden. Aufgrund des Aufbaus auf HTML, CSS und JavaScript lassen sich eine Menge von Effekten und Manipulationen erzeugen, ohne dass dabei ein tiefgreifendes Verständnis für die mathematischen und computergrafischen Programmiermethoden von Nöten ist.

# A. Material

## A.1. Diagramme

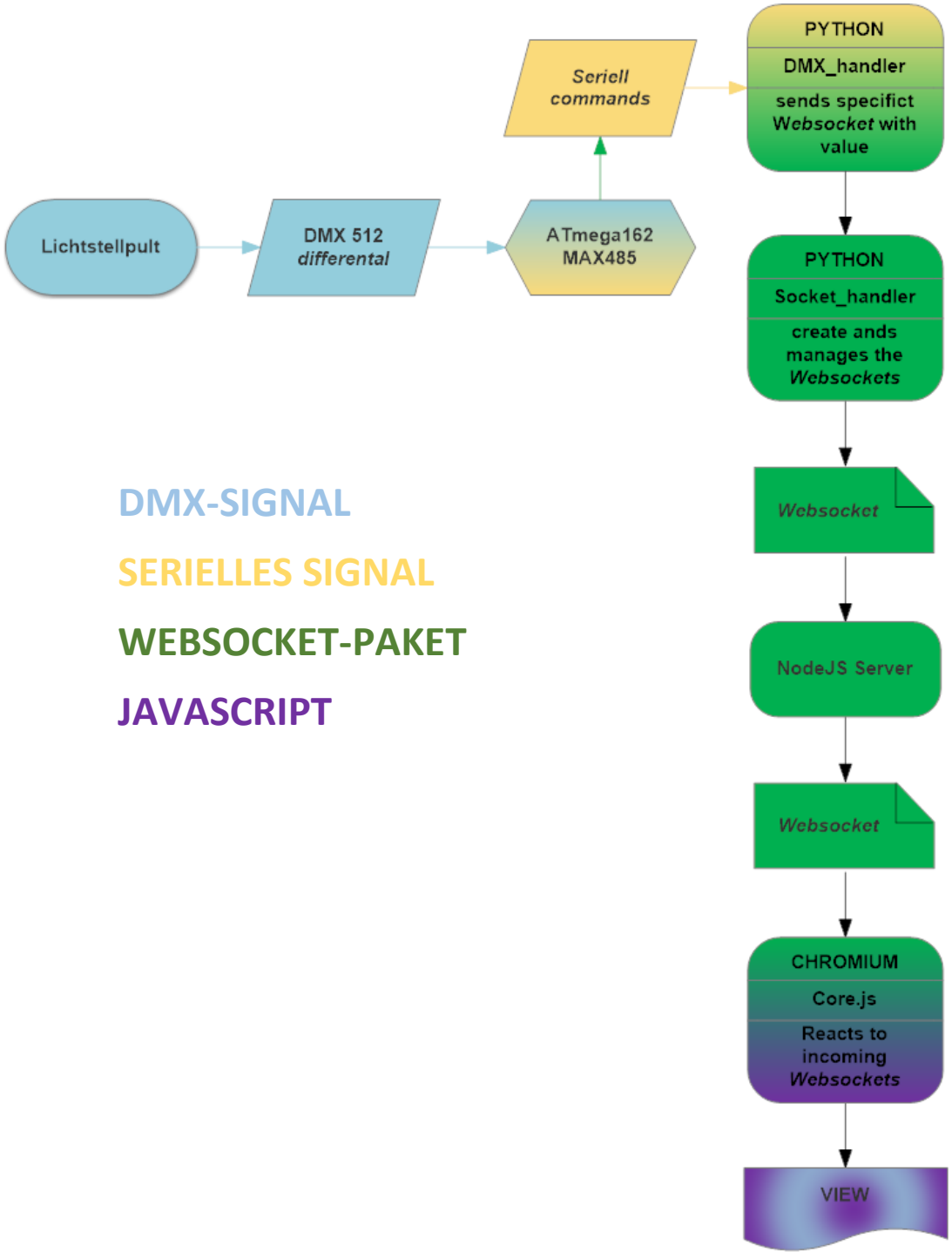


Abbildung A.14.: Laufdiagramm der Steuersignale und ihrer Art

## A.2. Schaltpläne

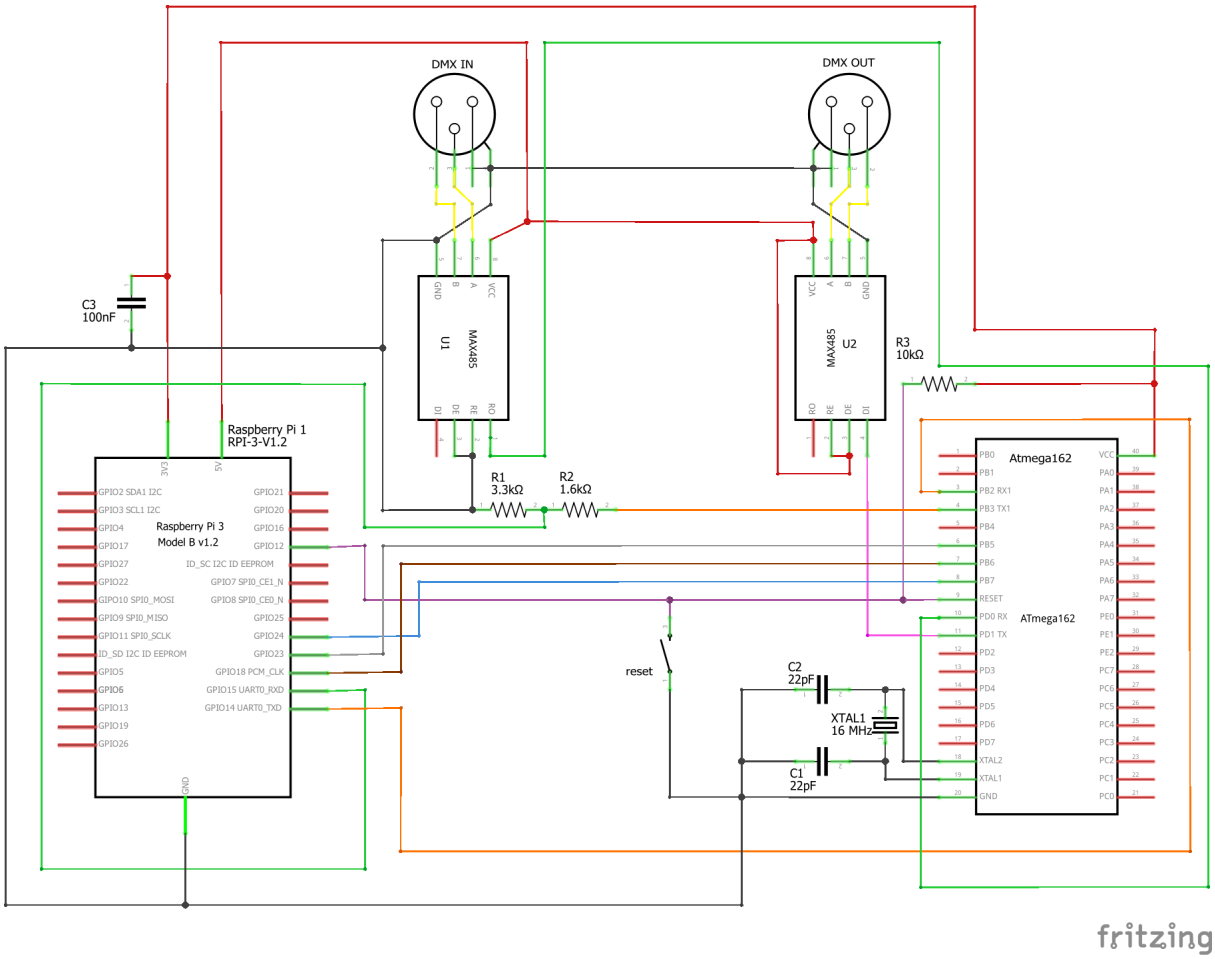


Abbildung A.15.: Schaltplan DMX-Shield

## A.3. Source-Code

Der komplette Quellcode für die Software des Medienservers, als auch für die ATmega Firmware, ist unter https://github.com/Exellent1988/RasViMap/ zu finden. Bei Abgabe dieser Bachelor-Thesis ist die aktuelle Revision : b8ec053

## A.4. Tabellen

**ImageCue™ DMX Profile 1**

| Chan | Parameter | Description | DMX Value | Default Value |
|---|---|---|---|---|
| 1 | Dimmer | Output intensity from blackout (0) to full intensity (255) | 0 - 255 | 255 |
| 2 | Image Source Folder | Stock JPEG and H264 Video | 0 | 0 |
| | | User Folder /imagecue001 | 1 | |
| | | User Folders /imagecue002 - 255 | 2 - 255 | |
| 3 | Image Select | No Selection (OPEN) | 0 | 0 |
| | | First Image File (001) | 1 | |
| | | Additional Image Files (002 - 255) | 2 - 255 | |
| 4 | Fade Time | Fade Time - Coarse (25.6 sec per step) | 0 - 255 | 0 |
| 5 | | Fade Time - Fine (0.1 sec per step) | 0 - 255 | 0 |
| 6 | Color | Red saturation from none (0) to full saturation (255) | 0 - 255 | 255 |
| 7 | | Green saturation from none (0) to full saturation (255) | 0 - 255 | 255 |
| 8 | | Blue saturation from none (0) to full saturation (255) | 0 - 255 | 255 |
| 9 | | Master Color Saturation from none (0) to full saturation (opaque) (255) | 0 - 255 | 0 |
| 10 | Mode Control | No Control Mode Selected | 0 | 0 |
| | | Future Control Modes | 1 - 63 | |
| | | Color in front of image | 64 | |
| | | Future Control Modes | 65 - 127 | |
| | | Video play once and exit | 128 | |
| | | Future Control Modes | 129 - 191 | |
| | | Color in front & Video play once | 192 | |
| | | Future Control Modes | 193 - 255 | |
| 11 | Overlay Select | No Selection (OPEN) | 0 | 0 |
| | | First User Overlay File (001) | 1 | |
| | | Add'l User Overlay Files (002 - 200) | 2 - 200 | |
| | | First Stock Overlay File (201) | 201 | |
| | | Add'l Stock Overlay Files (202 - 255) | 202 - 255 | |
| 12 | Overlay Opacity | Opacity of Overlay Image from transparent (0) to opaque (255) | 0 - 225 | 255 |

Abbildung A.16.: Face ImageCue - DMX Profil[1]

[1] Quelle: ImageCue - QuickStartGuide http://imagecue.lighting/media/downloads/quick_start_guide_web_v3.pdf Aufruf: 06.08.2016

| Modell | Banana Pi M3 | Raspberry Pi 3 |
|---|---|---|
| Hersteller | *SinoVoip* | *Raspberry Pi Foundation* |
| SoC | AllWinner A83T | Broadcom BCM2837 |
| CPU | ARM Cortex-A7 (32-bit)<br>2GHz octa core | ARM Cortex-A53 (64-bit)<br>1.2GHz quad core |
| GPU | PowerVR SGX544MP1 | VideoCore IV |
| RAM size | 2GB | 1GB |
| Built-in RAM | ✓ | ✓ |
| Internal storage | 8GB | ✗ |
| SD card | MicroSD card slot<br>**Max. size: 128GB** | MicroSD card slot<br>**Max. size: 128GB** |
| USB host | 2 | 4 |
| USB OTG | ✓ | ✓ |
| USB version | 2.0 | 2.0 |
| Ethernet | 1 port<br>**Type: Gigabit** | 1 port<br>**Type: 10/100** |
| Wake-on-Lan | ✗ | ✗ |
| HDMI | ✓ | ✓ |
| VGA | ✗ | ✗ |
| CVBS | ✗ | ✓ |
| Display interfaces | MIPI DSI | MIPI DSI |
| CSI camera | ✓ | ✓ |
| Audio output | ✓ | ✓ |
| Audio connector | 3.5mm jack | 3.5mm jack |
| HDMI audio | ✓ | ✓ |
| SPDIF | ✗ | ✗ |
| I2S | ✓ | ✓ |
| Line input | ✗ | ✗ |
| Mic input | ✗ | ✗ |
| Onboard mic | ✓ | ✗ |
| SATA | ✓ | ✗ |
| IR sensor | ✓ | ✗ |
| Wi-Fi | 802.11 b/g/n (AP6212) | 802.11b/g/n |
| Bluetooth | Bluetooth 4.0 | Bluetooth 4.1 |
| RTC | ✗ | ✗ |
| GPIO pins | 40 | 40 |
| PWM pins | 1 | 1 |
| ADC pins | ✗ | ✗ |
| I2C | 1 | 1 |
| SPI | 1 | 1 |
| UART | 2 | 1 |
| RS232 | ✗ | ✗ |
| Arduino pinout | ✗ | ✗ |
| Voltage | 5V | 4.8V - 5.2V |
| Power consumption | 1A - 2A | 600mA - 1.6A |
| Windows support | ✗ | Windows 10 |
| Android support | Android 5.1 | ✗ |
| Size (mm) | 92 x 60 x 17 | 85 x 56 x 20 |
| Weight (g) | 45 | 45 |
| Operating temperature | ? | 0 °C – 45 °C |
| Price | $74 | $35 |

Abbildung A.17.: Vergleich der zwei SingleBoardComputer[2]

[2]Quelle: SBC-DB http://www.board-db.org/compare/115,103/ Aufruf: 06.08.2016

# Abbildungsverzeichnis

# Abkürzungsverzeichnis

**DMX** Digital Multiplex

**HTML** Hypertext marup Language

**CPU** Central Processing Unit

**CSS** Cascading Style Sheets

**GPIO** general purpose Input/Output

**OLA** Open Lighting Architecture

**UART** Universal Asynchronous Receiver Transmitter

**SACN** Streaming Architecture for Control Networks

**SATA** Serial AT Attachment

**SBC** SingleBoardComputer

**SOC** System on a Chip

**vgl.** vergleiche

# Liste der Code-Teile

# Tabellenverzeichnis

# Literaturverzeichnis


**Arduino 2016a**
Arduino: *Arduino - Blink*. https://www.arduino.cc/en/Tutorial/Blink. Version: 2016, Abruf: 24.07.2016

**Arduino 2016b**
Arduino: *Arduino MEGA 2560*. http://www.arduino.org/products/boards/arduino-mega-2560. Version: 2016, Abruf: 24.07.2016

**Arduino 2016c**
Arduino: *Arduino NANO*. http://www.arduino.org/products/boards/arduino-nano. Version: 2016, Abruf: 24.07.2016

**Arduino 2016d**
Arduino: *What is Arduino*. http://www.arduino.org/learning/getting-started/what-is-arduino. Version: 2016, Abruf: 24.07.2016

**Art-Net 2016**
Art-Net: *Art-Net Specification Art-Net 3*. http://www.artisticlicence.com/WebSiteMaster/User%20Guides/art-net.pdf. Version: 2016

**Atmel 2016**
Atmel: *Embedded-Mikrocontroller ATMEGA162-16PU PDIP-40 Atmel 8-Bit 16 MHz Anzahl I/O 35*. http://www.produktinfo.conrad.com/datenblaetter/150000-174999/154285-da-01-en-ATMEGA162_16PU_DIL_40.pdf. Version: 2016, Abruf: 24.07.2016

**AVM 2015**
AVM: *FRITZ!Box-Mediaserver einrichten | FRITZ!Box 7390 | AVM Deutschland*. https://avm.de/service/fritzbox/fritzbox-7390/wissensdatenbank/publication/show/274_FRITZ-Box-Mediaserver-einrichten/. Version: 2015, Abruf: 05.08.2016

**Banana-Pi 2016**
Banana-Pi: *Banana Pi - A Highend Single-Board Computer*. http://www.bananapi.org/. Version: 2016, Abruf: 24.07.2016

**Bennet 2015**
Bennet: *CSS3 Filters applied to HTML5 video*. http://bennettfeely.com/labs/filters-on-video/. Version: 2015

**Bitwizard 2016**

Bitwizard: *DMX interface for Raspberry pi*. http://www.bitwizard.nl/shop/DMX-interface-for-Raspberry-pi. Version: 2016, Abruf: 24.07.2016

**Chome 2015**

Chome: *Rendering Settings - Google Chrome*. https://developer.chrome.com/devtools/docs/rendering-settings. Version: 2015

**Conceptinetics 2016**

Conceptinetics: *DMX Library for Arduino / Wiki / Home*. https://sourceforge.net/p/dmxlibraryforar/wiki/Home/. Version: 2016, Abruf: 24.07.2016

**Coolux 2016**

Coolux: *coolux - Airscan*. http://www.coolux.de/products/airscan/. Version: 2016, Abruf: 01.08.2016

**Coyier 2015**

Coyier, Chris: *SVG Filters on HTML5 Video*. https://codepen.io/chriscoyier/pen/zbakI. Version: 2015

**DMX 2015**

DMX, Open Source Raspberry P.: *Firmware - Open Source Raspberry Pi DMX / RDM / MIDI / OSC / Art-Net*. http://www.raspberrypi-dmx.com/raspberry-pi-rdm-controller/firmware. Version: 2015

**ExpressJs**

ExpressJs: *Express - Node.js-Framework von Webanwendungen*. http://expressjs.com/de/

**Futuremark 2013**

Futuremark: *Peacekeeper - free universal browser test for HTML5 from Futuremark*. http://peacekeeper.futuremark.com/. Version: 2013

**krpano 2015**

krpano: *krpano / HTML5 Video WebGL Performance Test*. http://krpano.com/ios/bugs/ios8-webgl-video-performance/. Version: 2015

**feiner lichttechnik 2016**

lichttechnik feiner: *DMX512 / DIN 56930-2*. http://www.feiner-lichttechnik.de/index.php/white_papers.html?file=tl_files/Download/whitepapers/dmx-din569302.pdf. Version: 2016, Abruf: 24.07.2016

**MA-Lighting 2015**

MA-Lighting: *grandMA2 light: MA Lighting International GmbH*. http://www.malighting.com/de/produkte/control/control/ma-lighting/grandma2-light/120112-grandma2-light.html. Version: 2015

**Maxim-Integrated 2016**

Maxim-Integrated: *Schnittstellen-IC - Transceiver Maxim Integrated MAX485CPA+ RS422, RS485 1/1 PDIP-8 zum Conrad Online-Shop | 001119858*. https://www.conrad.de/de/schnittstellen-ic-transceiver-maxim-integrated-max485cpa-rs422-rs485-11-pdip-8-1119858.html. Version: 2016, Abruf: 24.07.2016

**MultiWii 2016**

MultiWii: *MultiWii*. http://www.multiwii.com/wiki/index.php?title=Main_Page. Version: 2016, Abruf: 24.07.2016

**Open-Lighting-Architecture 2016**

Open-Lighting-Architecture: *Open Lighting Project*. https://www.openlighting.org/ola/. Version: 2016, Abruf: 24.07.2016

**Pfeiffer 2010**

Pfeiffer, Silvia: *The Definitive Guide to HTML5 Video (Expert's Voice in Web Development)*. Apress, 2010 http://www.amazon.com/Definitive-Guide-HTML5-Experts-Development/dp/1430230908%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D1430230908. – ISBN 1430230908

**Plex 2016**

Plex: *How Plex Works | Home Media Server | Media Manager*. https://www.plex.tv/how-it-works/. Version: 2016, Abruf: 01.08.2016

**RaspberryPi-Foundation 2016**

RaspberryPi-Foundation: *Strategy2016-18.pdf*. https://www.raspberrypi.org/files/about/RaspberryPiFoundationStrategy2016-18.pdf. Version: 2016, Abruf: 24.07.2016

**Reichelt 2016**

Reichelt: *BANANA PI M3: Banana Pi M3, 2 GHz Octa-Core, 2 GB DDR3, Wi-Fi bei reichelt elektronik*. https://www.reichelt.de/Einplatinen-Computer/BANANA-PI-M3/3/index.html?ACTION=3&GROUPID=6666&ARTICLE=162280. Version: 2016, Abruf: 01.08.2016

**Seifert 2010**
Seifert, Arne: *Langzeitarchivierung bei Medienservern*, Universität der Bundeswehr München, Dokotorarbeit, 2010

**Silvius 2013**
Silvius: *Using ATmega162 with Arduino IDE - OpenHardware.Ro*. http://openhardware.ro/using-atmega162-arduino-ide/. Version: Juni 2013, Abruf: 25.07.2016

**Socket.IO 2016**
Socket.IO: *Socket.IO*. http://socket.io/. Version: 2016

**Three.js 2015**
Three.js: *three.js / examples*. http://threejs.org/examples/#webgl_materials_video. Version: 2015

**tinker.it 2016**
tinker.it: *Google Code Archive - tinkerit - DmxSimple.wiki*. https://code.google.com/archive/p/tinkerit/wikis/DmxSimple.wiki. Version: 2016, Abruf: 24.07.2016

**w3schools 2016**
w3schools: *Tryit Editor v3.0*. http://www.w3schools.com/html/tryit.asp?filename=tryhtml5_video. Version: 2016

Ich versichere, die vorliegende Arbeit selbstständig ohne fremde Hilfe verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt zu haben. Die aus anderen Werken wörtlich entnommenen Stellen oder dem Sinn nach entlehnten Passagen sind durch Quellenangaben eindeutig kenntlich gemacht.

Hamburg, 22. August 2016 Axel Lodyga