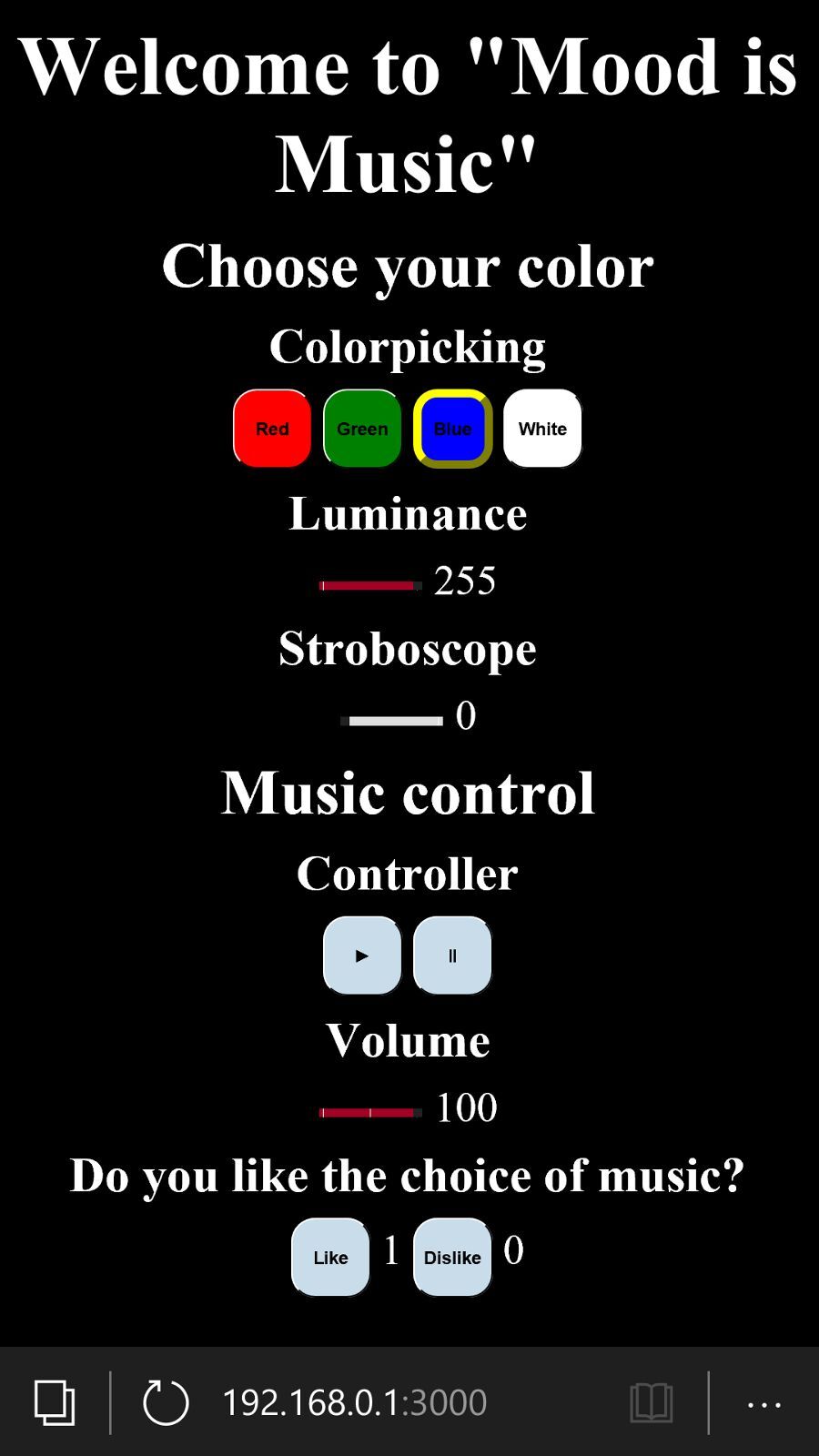
# Dokumentation zum ITS Projekt „Mood is music“

## Die WebApp:



Im Prototyp der Webapp stehen die vier Grundfarben des RGBW-Systems als Auswahlbutton zur Verfügung, die bei Betätigung RGBW Werte per Websocket versendet, sodass die Lampe in der gewünschten Farbe leuchtet. Außerdem gibt es je einen Regler für Luminance und Stroboscope Effekt, die zwischen den DMX Werten 0 und 255 eingestellt werden können.

Des Weiteren kann per „Play“ Button eingebundene Musik abgespielt und per „Pause“ Button angehalten werden. Über einen Volume – Regler kann die Lautstärke der abgespielten Musik zwischen den Werten 0 und 100 geregelt werden.

Für die spätere Funktion wurde schon zu Beginn der Entwicklungsphase sowohl ein „Like“- als auch ein „Dislike“- Button eingebaut, um den Counter zu simulieren, der im Endprodukt die Like / Dislike Verhältnisse der Genres in der Datenbank zählt.

In der finalen Version unserer Webapp wurde statt der vier RGBW Farben ein wabenförmiger Colorpicker mit 127 Farben eingebaut. Jedes Farbfeld enthält seine notwendigen Informationen über seine RGB Werte, die per Websocket als DMX Signal an die Lampe übermittelt werden.

Sobald eine Farbe ausgewählt wurde, wird per Funktion zufällig ein Lied aus einem der fünf Genres aus der Datenbank ausgewählt und abgespielt. Das zu hörende Lied kann per „Pause“ Button weiterhin pausiert und nach dem Pausieren per „Play“ Button weiter abgespielt werden. Wird der „Play“ Button gedrückt während das Lied läuft, wird ein neues Lied zufällig ausgewählt und abgespielt.

In einem Lauftextfeld wird angezeigt, welcher Interpret mit welchem Song gerade zu hören ist.

In der endgültigen Version gibt es den Volume Regler und die Buttons „Like“ und „Dislike“, mit dem Unterschied, dass sich die Gewichtung der zu einem Farbbereich zufällig ausgewählten Gernes innerhalb der Datenbank beim Drücken des „Like“ oder „Dislike“ Button verändert. Wird der „Like“ Button gedrückt spielt die Musik ganz normal weiter, wird jedoch der „Dislike“ Button betätigt wird sofort ein Lied aus einem anderen Genre abgespielt.

Eine weitere Veränderung ist die Hintergrundfarbe der Webapp, welche sich je nach ausgewählter Farbe verändert und diese für den User angezeigt wird.

Wenn mehrere User mit dem Raspberry Pi verbunden sind und die Webapp geöffnet haben, werden alle Veränderung, die ein User macht, den jeweils anderen sofort angezeigt bzw. in ihrer Ansicht mit verändert. So sieht jeder, der die Webapp auf hat, die aktuellen Einstellungen und es können alle Veränderungen bzw. eine Auswahl vornehmen ohne, dass es Probleme in der Ausführung oder Anzeige gibt.

## Intelligente Datenbank:

Die Musikauswahl soll anhand einer intelligenten Datenbank erfolgen. Die Datenbank soll sich merken, welches Genre bei welcher Farbe bevorzugt wird und dementsprechend ein Lied abspielen. Anfangs haben alle Genres dieselbe Wahrscheinlichkeit gewählt zu werden, nämlich 20%. Der User endscheidet mit einem „like“ oder „dislike“, ob ihm das zufällig abgespielte Lied zu der Farbe passt. Die Datenbank speichert die Vorlieben ab und bildet eine prozentuale Verteilung. Bekommen Lieder aus einem Genre bei einer bestimmten Farbe immer dislikes, dann sinkt der Prozentsatz des Genres. Damit alle Genres bei einer Farbe summiert immer 100% ergeben, werden dem gedislikten Genre Prozentpunkte abgezogen und dementsprechend auf die anderen Genres aufgeteilt.

Beispielhafte Datenbank:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id | Farbe | Rock | Pop | Hiphop | Metal | Reggae |
| 1 | Rot | 60 | 2 | 4 | 24 | 10 |
| 2 | Grün | 0 | 20 | 8 | 3 | 69 |
| 3 | Blau | 20 | 47 | 16 | 13 | 4 |
| 4 | Violett | 3 | 4 | 29 | 54 | 10 |
| 5 | Gelb | 43 | 5 | 2 | 37 | 13 |
| 6 | Orange | 12 | 9 | 33 | 17 | 29 |

Wird nun das Genre Metal bei der Farbe Gelb geliked, so werden von allen anderen Genres ein Prozentpunkt abgezogen und dem Genre hinzugefügt. In dem Falle würde Metal +4% und alle anderen Genres -1% bekommen.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id | Farbe | Rock | Pop | Hiphop | Metal | Reggae |
| 5 | Gelb | 41 | 5 | 4 | 37 | 13 |
| Neu |  | -1 | -1 | -1 | +4 | -1 |
| 5 | Gelb | 40 | 4 | 3 | 41 | 12 |

Wird bei der Farbe Grün Reggae geliked, so erhält das Genre nur +3%, da Rock bereits bei 0% ist und hier kein Prozentpunkt mehr abgezogen werden kann. Bei den verbliebenen Genres werden weiterhin 1% abgezogen.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id | Farbe | Rock | Pop | Hiphop | Metal | Reggae |
| 2 | Grün | 0 | 20 | 8 | 3 | 69 |
| Neu |  | +-0 | -1 | -1 | -1 | +3 |
| 2 | Grün | 0 | 19 | 7 | 2 | 72 |

Wird nun eine neue Farbe gewählt, bestimmt ein Zufallsgenerator anhand der aktuellen Prozentsätze, welches Genre gewählt wird. Anschließend wird ein zufälliges Lied aus dem Genre abgespielt.

## Kommunikation:

Die Kommunikation zwischen Webapp und Datenbank läuft über einen Raspberry Pi 3. Um genau zu sein, muss man sagen, dass die Skripte auf dem Raspberry laufen. Der Raspberry Pi bietet ein Wlan Netz in das sich die Benutzer einwählen können. Dadurch bekommen sie Zugriff über ihren Browser auf die oben beschrieben Webapp. Die dort eingegebenen Daten werden an die Datenbank gesendet, alles intern auf dem Raspberry. Die Musikausgabe von der Datenbank wird ebenfalls auf dem Raspberry ausgeführt und wird über die Klinckenbuchse nach außen geführt.  
Nur die DMX Ausgabe wird über externe Bauteile übernommen. Diese werden über die Serielle Schnittstelle versorgt. Die Daten werden von einem ATmega 162 verarbeitet auf dem eine Arduino IDE läuft.

## Hardware:

Verbaut wurden in der Schaltung:

* Raspberry Pi 3
* ATmega 162
* 2x MAX485
* 16Mhz Quarz
* 2x 22pf, 100nf Kondensator
* 10KΩ, 3,3KΩ, 1,6kΩ Widerstand
* DMX male/female Buchse
* Schalter

Am Anfang haben wir uns mit einem Breadboard langsam an die endgültige Schaltung herangetastet. Die ersten Versuche wurden mit Arduino Nanos sowie mit Megas, dem Raspberry und MAX485 Boards durchgeführt. Auf die Arduinos wurden anfangs einfache DMX Bibliotheken aufgespielt, um die Serielle Kommunikation zu testen und zu optimieren. Nachdem es uns gelungen war, wurde eine Schaltung ohne Arduino und nur mit dem ATmega Chip erdacht. Da wir so kostengünstiger und platzsparender arbeiten konnten. Der ATmega 162 wurde genommen, da er genug I/O Pins und Leistung besitzt, aber noch nicht riesen groß wird.  
Die MAX 485 Chips sind für das DMX Signal notwendig. Da dieses aus einem Signal und einem invertierten Signal zusammengesetzt ist. Die Invertierung übernehmen diese Chips.  
Der Quarz sowie die zwei 22pf Kondensatoren dienen als externe Clock für den ATmega.  
Die Widerstände sind als Pullup verbaut. Der 100nF Kondensator dient als Schutz, um etwaige Spitzen o.ä. abzufangen.  
Für Notfallzwecke wurde noch ein Reset Knopf verbaut, damit man den ATmega reseten kann falls nichts mehr geht. Dieser war beim fertigen Projekt, allerdings im Gehäuse versteckt. Aus diesem Grund wurde der Reset Pin des ATmega zusätzlich mit einem GPIO Pin des Raspberrys verbunden. Ein entsprechendes Reset Skript wurde erstellt.  
Die ganze Schaltung ist so konzipiert, dass es möglich ist den ATmega neu zu programmieren und so anders zu nutzen. Daher haben wir auch 2 MAX Chips verbaut. So ist es neben dem DMX empfangen auch möglich DMX zu senden. Die Hardware ist also so ausgelegt, dass man sie für jede DMX Anwendung in einem Universum nutzen kann und so frei in seiner Gestaltung ist.  
Dabei ist jedoch darauf zu achten, dass man das Gerät nicht einfach so in eine DMX Kette einfügen kann. Die beiden DMX Buchsen sind nicht durchgeschleift, da es als Sender benutzt werden kann. Wenn zwei Sender in einer DMX Kette hängen kann es zu Geräteschäden kommen, deshalb ist das gleichzeitige Senden und Empfangen Software mäßig zu regeln.

Die Stromversorgen für den ATmega läuft über die 3,3V Leitung von Raspberry und die Stromversorgung für die MAX Chips über die 5V Leitung.

Die Pins zur Programmierung und für die Serielle Kommunikation sind direkt vom Raspberry zum ATmega gezogen.

Am Ende wurde alle in ein Gehäuse eingepasst und noch ein Lüfter zur Kühlung verbaut.

Schaltplan:

