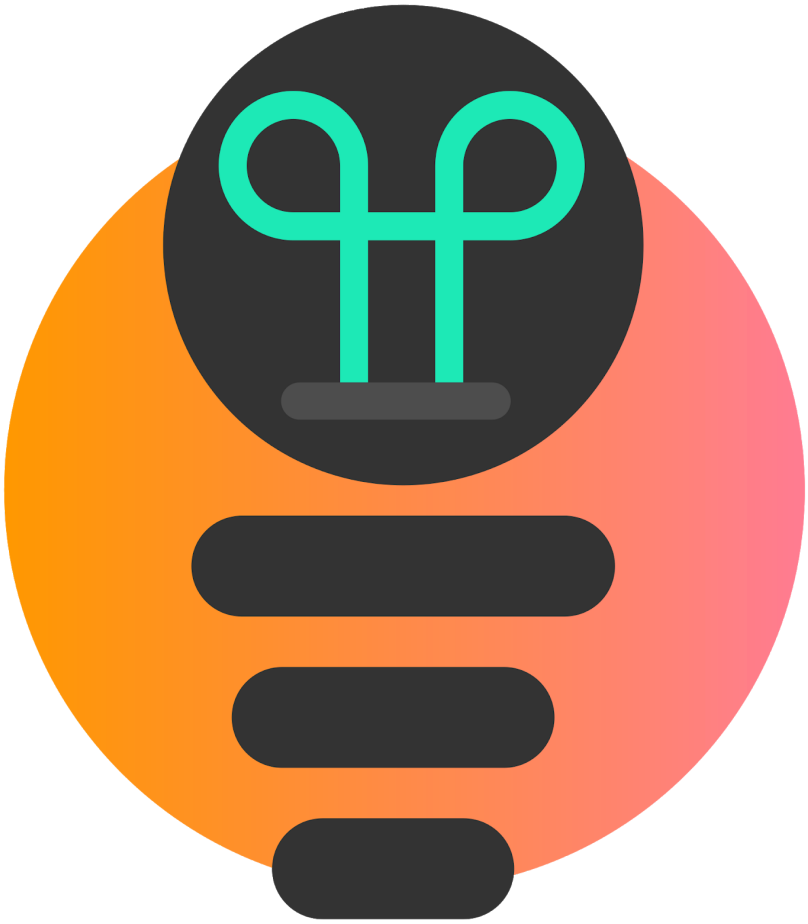
Projekt DevLights

Smarte LED-Streifen für Entwickler



Joshua Slaar

Graf Adolf Gymnasium der Stadt Tecklenburg

Projektkurs Informatik

Schuljahr 2020 / 21

Herr Spratte / Herr Paaschen

Inhaltsverzeichnis

Vorwort **3**

1. Die Projektidee **4**

2. Umsetzung **4**

3. Aufbau **6**

3.1 Das Licht 6

3.1.1 Hardware 6

3.1.2 Programmierung des ESP (Software) 7

3.2 Die API (Schnittstelle) 9

3.2.1 Web API / Server 9

3.2.2 Raspberry Pi 4 12

3.3 Software / Anwendungen 14

3.3.1 Die Smartphone Applikation (App) 14

3.3.2 Die Spielmodifikation 15

4. Zukunftsaussichten **16**

5. Fazit **17**

6. Quellenverzeichnis **18**

Erklärung **19**

Vorwort

“Und Gott sprach: Es werde Licht. Und es wurde Licht”.
Die Welt begann laut der Bibel mit Licht, auch die Theorie des Urknalls, beginnt durch eine Explosion, durch die das Licht und alles weitere, wie Materie entstand. Alles beginnt mit Licht und alles benötigt Licht.
Licht ist für die Menschen und den ganzen Planeten eine der wichtigsten Dinge, die es überhaupt benötigt damit es Leben gibt. Licht trägt zur Produktion von Sauerstoff bei. Für den Menschen ist Licht auch schon immer wichtig gewesen und wird es auch immer sein, da ohne Licht kein Leben möglich wäre.
Bereits seit Beginn der Menschheitsgeschichte benutzen die Menschen Licht, um ihren Alltag zu erleichtern. Bei den Höhlenmenschen in der Form eines Lagerfeuers, um bei Dunkelheit ihr Leben zu erhellen. Und bei Tag das Sonnenlicht.
Mit der Erfindung von künstlichen Lichtquellen, wie der Öllampe im 16. Jahrhundert und verschiedenen anderen Gaslichtern in den darauffolgenden Jahrhunderten, entstanden immer mehr Möglichkeiten für die Menschen auch für längere Zeiten im Dunkeln und auch am Tag, wenn die Sonne durch Wolken verdeckt war, sehen zu können .
Mit der Entdeckung der Elektrizität und der Erfindung von elektrischen Lampen, wie der Glühbirne durch Thomas Edison, wurden weitere, meist weniger gefährliche Möglichkeiten geschaffen, da es durch Öllampen schneller zu Bränden kam als durch Glühbirnen.
Mit der Erfindung der farbig leuchtenden LED im Jahre 1972 konnte man Licht auch künstlich in Millionen verschiedenen Farben leuchten lassen.
Licht wurde dadurch nicht nur mehr zur Erleuchtung, sondern auch für eine schönere Darstellung der Umgebung benutzt. Auch Lichtspielereien und Lichtshows wurden möglich.
Die vielen Möglichkeiten boten zum Beispiel das eigene Haus oder die eigene Wohnung in schöne Farben zu hüllen.
Dieses Projekt setzt bei der Idee der Verschönerung an, näher erklärt wird alles mit der Projektidee.

1. Die Projektidee

Heutzutage gibt es viele verschiedene LED-Streifen mit unterschiedlichen Steuerungsmodulen. Selbst Discounter wie Lidl und Aldi verkaufen LED-Streifen. Doch bei den meisten dieser LED-Streifen gibt es viele verschiedene Steuerungen, mit Fernbedienungen, Smartphone etc..
Das Problem dabei ist, dass jeder Hersteller beziehungsweise Anbieter seinen eigenen Weg zur Steuerung hat, und sich diese Steuerung nicht bearbeiten oder erweitern lässt. Man hat also ein unveränderbares System.
Dieses Projekt versucht da anzusetzen und dieses Problem zu beheben in dem es ein gewisses Grundsystem bestehend aus LED-Streifen, einer Kontrolleinheit und einer API (“Schnittstelle”, “Knotenpunkt”) bieten. Dem Entwickler wird die Möglichkeit geboten diese API zu nutzen um verschiedene Anwendungen zu entwickeln und dadurch seine LED-Streifen nach seinen Wünschen zu steuern. Steuern ließe sich das System zum Beispiel durch eine Smartphone Applikation.

2. Umsetzung

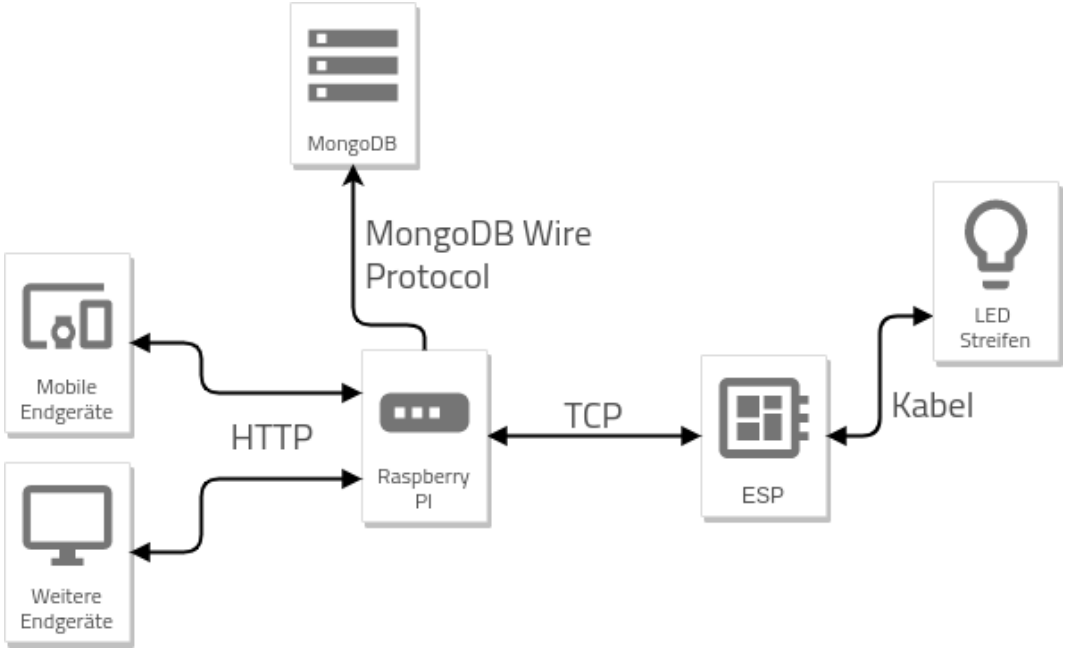


Abbildung: Verbindungsdiagramm

Aus der grundlegenden Idee der Komponenten oben in der Projektidee wurde dann der in der Abbildung dargestellte Aufbau.

Zu dem Aufbau gehören folgende Komponenten, die nun näher in Bezug auf die Funktionsweise und die Kommunikation erläutert werden.

Als LED-Streifen kamen verschiedene LED-Streifen in Frage die einen WS2812B Controller enthalten

LED-Streifen mit diesem Controller bieten den Vorteil, dass diese “programmierbar” sind, also jede LED sich über ein Programm ansteuern lässt. Dafür gibt es eine gut dokumentiere C++ Biblothek, die den Namen Adafruit Neopixel hat, auf die später näher eingegangen wird[1].

Für dieses Projekt wurde das Modell von BTF-LIGHTNING[2] mit dem WS2812B Controller verwendet.

Der LED-Streifen ist mittels Kabel, zu Beginn auf einem Breadboard, mit dem ESP-12F verbunden. Der ESP-12F ist ein Wifi Modul, welches einerseits netzwerkfähig ist, was für die Kommunikation mit der API wichtig ist, da diese netzwerkbasiert läuft, und gleichzeitig auch die Kommunikation mit dem LED-Streifen ermöglicht, er dient als dessen Kontrolleinheit. Ebenfalls ist der ESP-12F Modul relativ kostengünstig und es gibt verschiedene “Entwicklerboards” für dieses, welche zum Beispiel einen USB-Anschluss bieten. Für dieses Projekt wird das Entwicklerboard ESP8266[3] mit dem darauf befestigten ESP-12F Modul verwendet.

Die Kommunikation zwischen ESP-12F geschieht mit der API geschieht durch eine Netzwerkverbindung. Das bedeutet logischerweise, dass auch das Gerät auf dem die API läuft, die Möglichkeit bieten muss sich mit einem Netzwerk zu verbinden. Dafür wird in diesem Projekt ein Raspberry Pi[4] verwendet. Der Raspberry Pi ist ein Mikrocomputer. Dieser verarbeitet dann die Anfragen von Anwendungen, und gibt diese via Transfer Control Protocol (kurz. TCP, dt. Übertragungssteuerungsprotokoll) an den ESP-12F weiter.

Als Anwendungen kann in der Theorie alles eingesetzt werden, beispielsweise eine Smartphone Applikation, eine Web-App, oder auch eine Spielmodifikation . Diese Anwendungen müssen dann via Hypertext Transfer Protocol (kurz. HTTP) mit der API kommunizieren, um die Lichter steuern zu können.

---

[1] Adafruit Neopixel
[2] BTF-LIGHTNING WSB2812B
[3] ESP8266 NodeMCU
[4] Raspberry Pi 4 Model b

3. Aufbau

3.1 Das Licht

3.1.1 Hardware

Damit der LED-Streifen überhaupt funktionieren kann, braucht er wie jedes andere Gerät Strom. Dafür ist wichtig zu wissen, welche Spannung und welche Stromstärke, dieser benötigt, damit dieser funktioniert. Auch ist es wichtig zu wissen, welche Spannung der ESP-12F und sein Entwicklerboard benötigen, da es bei unterschiedlichen Spannungen etc. auch schnell zu Problemen kommen kann, wenn das eine Gerät zu viel und das andere zu wenig Spannung/Stromstärke bekommt. Und das war auch eines der ersten Probleme, die auftraten

Nämlich der LED-Streifen, welchen wir benutzen, der BTF-Lightning WS2812BALL, benötigt 5 Volt (V) Spannung und 9 Ampere (A) Stromstärke.

Die Spannung des Streifens darf um maximal 0,5V abweichen, also muss die Spannung mindestens 4,5V und maximal 5,5V betragen.

Das Entwicklerboard arbeitet sowohl mit einer Spannung von 5V als auch 3,3V. Das ESP-12F Modul kann jedoch nur mit 3,3V funktionieren und gibt auch nur soviel Spannung zurück. Um die Funktionsweise zu gewährleisten, musste dieser Spannungsunterschied von 1,2V beziehungsweise 1,7V ausgeglichen werden. Dafür wurde eine extra Komponente in den Stromkreis eingebaut.

Diese Komponente nennt sich Pegelumsetzer (engl. Levelshifter), der dafür sorgt, dass die Eingangsspannung von 3,3V an die benötigte Ausgangsspannung von 5V angepasst wird, um so trotz der unterschiedlichen Spannungen für ein gutes Funktionieren zu sorgen.[5]

Dazu kommt noch ein anderes Problem, dass mit der Stromstärke zusammenhängt. Der LED-Streifen in seiner Ausführung benötigt nicht nur 5V Spannung, sondern empfiehlt auch eine Stromstärke von 9 Ampere (A). Doch herkömmliche Netzteile bieten entweder eine viel zu hohe Spannung und die passende Stromstärke, oder eine passende Spannung und eine zu niedrige Stromstärke. Beispielsweise ein herkömmliches Netzteil für Ladekabel von Smartphones hat 5V Spannung und 2A Stromstärke. Es gibt zwar auch einige Industrienetzteile, die eine fast passende Stromstärke und Spannung liefern,

[5] Pegelwandler

aber die sind einerseits relativ unhandlich und aus Kostengründen wurde entschieden, solche nicht in diesem Projekt zu nutzen.

Soweit funktioniert der Aufbau, mit ein paar Problemen, die in unregelmäßigen Abständen für falsches Leuchten sorgen, was mit der zu geringen Stromstärke zusammenhängt.

Während der Entwicklung des Projektes wurde die Stromversorgung des ESPs und des LED-Streifens und deren Verbindung auf einem Breadboard aufgebaut. Das ist jedoch ziemlich platzintensiv und somit nicht gerade praktikabel. Deswegen wurde überlegt, den Aufbau durch die Verbindung der Komponenten auf einer Platine zu ersetzen, Zuerst noch mit dem ESP8266 Entwicklerboard, da dieses die einfache Möglichkeit des Hochladens von Programmcode bietet. Da dies aber auch nicht gerade praktikabel war, da das Entwicklerboard selbst gelötet werden musste, wurde beschlossen, nur noch den ESP-12F und unter anderem den Pegelumsetzer und die Leiterbahnen auf die Platine zu setzen. Das erschwert zwar das Hochladen von Programmcode auf den ESP-12F aber macht es noch lange nicht unmöglich, Es muss dafür entweder ein extra Raspberry Pi oder ein extra Entwicklerboard, wie den ESP8266 verwendet werden.

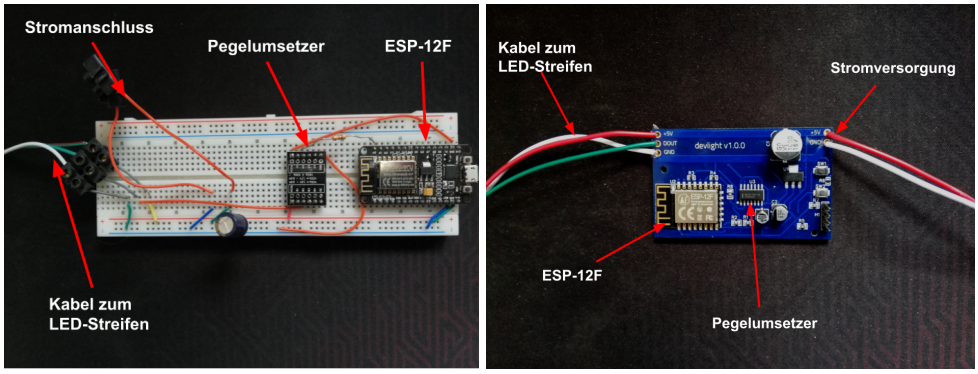


Abbildung Links der Aufbau mit einem Breadboard, rechts die Platine

3.1.2 Programmierung des ESP (Software)

Für die Kontrolle des LED-Streifens wird eine Kontrolleinheit benötigt, die dem LED-Streifen sagt, wie die einzelnen LEDs leuchten sollen.

Hier kommt der ESP8266 und sein ESP-12F Modul ins Spiel. Diese beiden sind zwar am Anfang erstmal “dumm”, aber sind programmierbar und genau das wird auch gemacht, um die Funktionen einzubauen, die gewünscht sind. Dafür nutzen wir die Programmiersprache C++ und die Entwicklungsumgebung Arduino IDE.

Die Ansteuerung der einzelnen LEDs geschieht, wie bereits in der Umsetzung beschrieben, mittels Open-Source Bibliothek Adafruit NeoPixel[6].

[6]Adafruit Neopixel

Adafruit Neopixel stellt verschiedene Methoden zur Ansteuerung von unterschiedlichen LED-Streifen durch verschiedene Mikrocontroller zur Verfügung, zu denen auch das Modell von BTF-Lightning und das ESP8266 Entwicklerboard gehören. Neopixel lässt sich sowohl für LED-Streifen nur in Verbindung mit bestimmten Mikrocontrollern benutzen[7]

Der ESP dient als Empfänger und Verarbeiter der Daten, die er seitens der API gestellt bekommt.

Damit er diese Daten überhaupt bekommen kann, muss er erstmal eine Verbindung zur API beziehungsweise zum Raspberry Pi, auf welchem der Server läuft, herstellen. Dies geschieht durch eine TCP Verbindung, die über das Wifi Modul des ESPs aufgebaut wird. Dafür stellt der Raspberry Pi (kurz. Pi) ein eigenes WLAN-Netzwerk zu Verfügung. (TCP steht für Transfer Control Protocol)

In dieses WLAN Netzwerk des Pis loggt sich der ESP-12F dann ein.

Beim ersten Einrichten sendet der ESP eine Anfrage an den Server, die zur Einrichtung des ESP auf dem Server dient, bei der die IP-Adresse des ESPs mitgegeben wird, damit der Server weiß, wie er Daten an den ESP senden soll.

Die Daten, die der ESP seitens der API gesendet bekommt, werden im JavaScript Object Notations (JSON)[8] Format übertragen. Das JSON Format wird mehrheitlich zum Austausch und zur Speicherung von Daten benutzt. Da C++ kein solches Format wie JSON zur Verfügung stellt, wird die Bibliothek ArduinoJson[9] benutzt, die dieses zur Verfügung stellt.

```
1  {
2      "command": "leds",
3      "data": {
4          "pattern": "plain",
5          "colors": ["#00FF00"]
6      }
7  }
```

Abb: Beispielaufbau eines JSON Objektes, welches der ESP vom Server bekommt, um die Farbe auf Gelb zu ändern.

Das JSON Objekt enthält immer einen Befehl (“command”) und die für den Befehl benötigten Daten (“data”), womit der programmierte ESP dann die passende Aktion ausführt.

im Programmcode wird dann überprüft, welcher Befehl gesendet wurde.

---

[7] Adafruit Neopixel - Supported Chipsets
[8] JSON
[9] ArduinoJSON

Entspricht der Befehl einem der implementierten Befehle (z.B. Einschalten (“on”), Farbe ändern (“leds”))[10] wird der entsprechende zugeordnete Teil des Programmcodes ausgeführt. Sollte der Befehl nicht einem der Befehle, die einprogrammiert wurden entsprechen, passiert nichts.

Der ESP kann aber nicht nur die Farbe ändern und starten/neu starten. Es gibt auch die Möglichkeit die Anzahl der LEDs auszuwählen, für z.B. längere oder kürzere Streifen, oder wenn die Stromstärke nicht für den Betrieb ausreicht. Auch das Einstellen der Helligkeit ist möglich. Für die verschiedenen Muster, die der LED-Streifen darstellen können soll, werden verschiedene mathematische Berechnungen durchgeführt, um die richtige Farbe oder richtige Position auf dem LED-Streifen zu ermitteln.

Alle Aktionen im Programmcode, die den LED-Streifen betreffen, werden durch Methoden der Bibliothek Adafruit NeoPixel (s.o.) ausgeführt,

Ein Beispiel hierbei wäre `fill(RGB color)`, was allen einzelnen LEDs sagen würde, nehme diese Farbe `color` an.

Weiterhin muss der ESP-12F auch mit verschiedenen Situationen klar kommen die auftreten. Wird der ESP-12F neugestartet oder die Stromversorgung unterbrochen, so hat er vorher bereits das eingestellte Muster gespeichert. Für die Speicherung dieses Muster auf dem ESP gibt es einen Flashspeicher[11] auf den mittels EEPROM Bibliothek[12] zugegriffen und Daten gespeichert / ausgelesen werden können.

Damit der Programmcode überhaupt auf den ESP-12F kommt, wird der ESP, über das Entwicklerboard ESP8266, welches ein USB-Port besitzt, mit dem Computer verbunden, und mittels der Hochladen Funktion auf den ESP gebracht.

In der erweiterten Version, in der das ESP8266 Entwicklerboard, durch die Platine ersetzt wird, fällt der USB-Port weg und das Hochladen erfolgt auf einem anderen, bereits erklärtem Wege (siehe 3.1.2. Hardware).

3.2 Die API (Schnittstelle)

3.2.1 Web API / Server

Die API (“Schnittstelle), genauer gesagt die Web API, basiert auf NestJS.

NestJS ist ein “progressives NodeJS Framework zum Erstellen effizienter, zuverlässiger, und skalierbarer serverseitiger Anwendungen”[13].

---

[10] ProjektDevlights Hardware
[11] Flash-Speicher
[12] EEPROM Library
[13] NestJS a progressive NodeJS framework

NodeJS[14] ist eine serverseitige JavaScript Laufzeitumgebung. NestJS erweitert diese durch verschiedene Funktionen.

Eine dieser Funktionen ist, dass es einen Webserver zur Verfügung stellt. Normalerweise ist dieser Webserver von Express, doch hier wird fastify[15] genutzt, was den Vorteil bietet, dass Anfragen von Nutzern in beinahe Echtzeit verarbeitet werden. Express[16] hingegen braucht mehr Zeit zum Verarbeiten der Anfragen Deswegen haben wir uns für fastify entschieden, da es eben den Vorteil bietet, Anfragen schnell zu verarbeiten, da es für dieses Projekt sehr wichtig ist, dass Anfragen schnell verarbeitet werden, damit zum Beispiel auf Aktionen in Videospielen reagiert werden kann.

Ein weiterer Vorteil den NestJS bot, ist dass es hundertprozentig kompatibel ist, mit der typisierten JavaScript Variante, Typescript[17], was den Vorteil bietet das man ein typisiertes Projekt erstellen kann. Da Typescript auf JavaScript basiert, ist es auch möglich sowohl in JavaScript als auch TypeScript geschriebene Bibliotheken zu benutzen.

Der Server ist via des MongoDB Wire Protokolls mit einer entweder im lokalen Netzwerk oder in der Cloud liegenden MongoDB[18] Datenbank beziehungsweise deren Datenbankserver verbunden. Das MongoDB Wire Protokoll sorgt kurz gesagt für die Kommunikation von dem NestJS Server mit dem Datenbankserver via TCP.[19]

Die Datenbank ist zur Speicherung der Lichter, und allem was dazu gehört, wie Wecker, Labels (Tags) etc. da. MongoDB arbeitet im Gegensatz zu MySQL mit Dokumenten, die dem JSON Standart folgen, während MySQL mit Tabellen arbeitet. Für den Anschluss über das MongoDB Wire Protokoll gibt es die JavaScript Bibliothek mongoose. Mongoose ist somit dafür zuständig die Daten, die in der Datenbank gespeichert werden sollen, an die Datenbank zusenden und auch für die Verarbeitung der Datenbank Anfragen von und zur Datenbank.[20]

Der Server stellt als Grundlage verschiedene HTTP-Anfragen, die der Kommunikation dienen. Das Prinzip funktioniert so: Etwas sendet eine Anfrage an eine Route und kriegt eine Antwort. Diese sind unterteilt in vier verschiedene Routen, für die jeweiligen Komponenten. Dies dient der Übersicht, man könnte auch alles ohne die Unterrouten aufbauen.

---

[14] Node.js
[15] Fastify Fast and low overhead web framework, for Node.js
[16] Express
[17] TypeScript language
[18] MongoDB The application data platform
[19] MongoDB Wire Protocol
[20] Mongoose

Diese vier Unterrouten sind, für Lichter (/lights), für Alarme (/alarm), für Labels (eng. tags, /tags) und für den ESP (/esp).
Die Unterrouten für die Lichter, Labels und Alarme sind für Nutzeranfragen da. Der Nutzer sendet eine Anfrage an den Server, um sein Licht einzuschalten, dessen Muster oder Farbe zu ändern. Die Daten, welche der Nutzer dabei mitgibt sind denen der die der ESP bekommt zwar ähnlich, jedoch wird kein Befehl mitgegeben. Ein Beispiel hierbei wäre das Ändern einer Farbe auf einem bestimmten Licht, dafür sendet der Nutzer eine Anfrage an /lights/{id}/color, (id=ID des ESPs) die Daten, die dabei mitgegeben würden, sähen wie folgt aus:

```
1  {
2      "pattern": "plain",
3      "colors": ["#FF0000"]
4  }
```

Abbildung: Daten die zur Farbänderung (hier rot) übergeben werden.

Um zu überprüfen ob es ein Licht, welches man bearbeitet oder abfragt überhaupt gibt, wird zwischen die Anfrage und der Verarbeitung eine Middleware gesetzt, die eine Anfrage an die Datenbank schickt, und sollte diese nichts zurückgeben, ist davon auszugehen, dass das Licht nicht in der Datenbank existiert und dem Sender der Anfrage wird dies zurückgegeben. Das gleiche gilt auch für die Wecker und die Labels.
Die Wecker Routen sind für die sog. Weckfunktion des Projektes. Damit wird es dem Nutzer ermöglicht, sich durch sein Licht, wie ein Wecker, um eine bestimmte Uhrzeit wecken zu lassen. Dabei leuchtet das Licht 5 Minuten lang in einer bestimmten ausgewählten Farbe, bis es seine volle Helligkeit erreicht hat. Da hierbei für die sich ständig ändernden Farben besonders viele Anfragen an den ESP gesendet werden, da die Funktionen des Weckers auf dem Server und nicht auf dem ESP implementiert wurden. Das wurde so gemacht, da es in dieser Ausführung einfacher ist dem Benutzer, der den Wecker erstellt hat, mitzuteilen wann der Wecker zu Ende ist. Hätte man das auf dem ESP entwickelt, so hätte dieser dem Server per Anfrage mitteilen müssen, dass der Wecker fertig ist, und dieser hätte es dann erst an den Nutzer weitergegeben.
Um einen Wecker zu erstellen übergibt der Benutzer eine Farbe, eine Uhrzeit und optional Wochentage an denen der Wecker ausgeführt werden soll. Gibt er keinen Wochentag an, läuft der Wecker jeden Tag. Wird ein Wecker erstellt überprüft der Server nicht nur die ankommenden Daten auf ihre Korrektheit, sondern überprüft auch, ob jener mit einem anderen Wecker in die Quere kommt.

Wenn dies der Fall ist, wird der Wecker nicht in der Datenbank abgelegt, und dem Benutzer wird mitgeteilt, dass der Wecker so einem anderen in die Quere kommt.

Damit der Wecker um eine bestimmte Uhrzeit ausgeführt werden kann, wird das ganze geplant. Für diese Planung gibt es ein universelles Format zur Planung von “zeitbasierten Ausführungen von Prozessen”. Dieses Format nennt sich “cron pattern”.[21] Über das Zeitplanungsmodul von NestJS werden die Wecker dann anhand des gegebenen Formates gespeichert und zum richtigen Zeitpunkt ausgeführt.

Bei den Esp Routen ist das etwas anders, das sind nur Routen für den ESP, zum Einrichten (/esp/setup) und falls sich die IP-Adresse des ESPs (/esp/update) ändert. Damit diese nicht von anderen Geräten benutzt werden, wurde eine Middleware eingebaut, die überprüft anhand des User-Agents ob der Sender der Arduino Plattform entstammt. Eine weitere Besonderheit bei den Anfragen des ESPs sind, dass diese obwohl Daten vom ESP an den Server übertragen werden, HTTP-GET Anfragen sind. Das liegt daran, dass es auf dem ESP einfacher ist eine GET-Anfrage zu implementieren als andere Anfragen, wie POST oder PATCH. Im Normalfall sollte man Daten nicht via GET übertragen.

Beim ersten Einrichten, wird dem ESP eine ID seitens des Servers zugewiesen, damit das Licht eindeutig identifizierbar ist. Wenn sich dann die IP-Adresse des ESPs ändert, dann teilt dieser dem Server die neue IP-Adresse mit, und gibt zur Identifizierung ebenfalls diese ID mit. Daten, die an den ESP gesendet werden, werden mittels TCP übertragen. Bei der Übertragung via TCP werden nur einzelne Bytes übertragen, und nicht wie bei HTTP, was zwar auf TCP basiert, noch weitere Daten, wie Header et cetera. Für die Kommunikation mit dem ESP via TCP gibt es von Node.js selbst ein eigenes Modul net. Da NestJS auf Node.js basiert ist eine normale Nutzung von net möglich. net stellt in diesem Fall einen TCP Server mit dem sich der ESP verbindet. net kann sowohl als Server als auch als Client benutzt werden. Hier in diesem Projekt fungiert der Web Server als TCP Server mit dem sich der ESP als Client verbindet, um sich nicht ständig wieder neu mit dem Server verbinden zu müssen. Das ermöglicht eine schnelle Kommunikation zwischen Server und ESP.

3.2.2 Raspberry Pi 4

[21] Cron Wikipedia

Der Server läuft auf einem Raspberry Pi, genauer gesagt momentan auf dem Raspberry Pi 4 Model B[22]. Die Verwendung eines anderen Raspberry Pi Model ist ebenfalls möglich. Der Raspberry Pi bietet den Vorteil, dass er einerseits nicht allzu viel Platz wegnimmt, da er relativ klein ist und trotzdem eine gute Leistung hervorbringt, um so die Möglichkeit zu bieten, den NestJS Server auf jenem laufen zu lassen. Ein weiterer Vorteil den der Raspberry Pi bietet ist, dass man diesen durchgängig laufen lassen kann, und der Stromverbrauch deutlich niedriger ist, als wenn man einen herkömmlichen PC durchgängig laufen lassen würde. Das auf dem Raspberry Pi laufende Betriebssystem Raspberry Pi OS Lite[23], sorgt dadurch, dass es lediglich eine Konsole und keine Desktop GUI Umgebung zur Verfügung stellt, für eine gute Performance und den niedrigen Stromverbrauch. Würde man die Desktop Version benutzen könnte ein höherer Stromverbrauch entstehen, die Auswirkungen auf diesen Webserver wären jedoch gering.

Der Raspberry Pi besitzt zusätzlich, wie fast jedes Gerät heutzutage, ein WLAN Modul und einen LAN-Anschluss. Dies machen wir uns in diesem Projekt zunutze um das WLAN Modul so umzufunktionieren, dass diese nicht ein WLAN Netzwerk empfängt sondern, ein eigenes WLAN Netzwerk ausstrahlt, um bei der Entwicklung des ESPs nicht ständig das WLAN Netzwerk auf das jeweilige Netzwerk beim Benutzer zuhause anpassen zu ermöglichen. Da der Raspberry Pi in diesem Falle als Access Point funktioniert, überlässt er dem oberen Router, also dem Router des Nutzers die IP-Adressen Verwaltung etc, und strahlt nur ein zusätzliches Netzwerk aus. Das ändert nichts an der bestehenden Struktur des Heimnetzes, fügt aber ein neues WLAN Netzwerk hinzu. Dafür muss der Raspberry Pi jedoch via LAN-Anschluss oder eventuellen zweiten WLAN-Modul an das Netzwerk angeschlossen werden, da es nicht möglich ist, sich über das umfunktionierte WLAN-Modul in ein anderes WLAN einzuloggen, da nur eine Funktion gleichzeitig auf einem Modul möglich ist. Verbindet man den Raspberry Pi nicht mit dem Netzwerk des Nutzers, funktioniert der ganze Projektaufbau nicht.

[22] Raspberry Pi Model 4 B
[23] Raspberry Pi Operating systems

3.3 Software / Anwendungen

3.3.1 Die Smartphone Applikation (App)

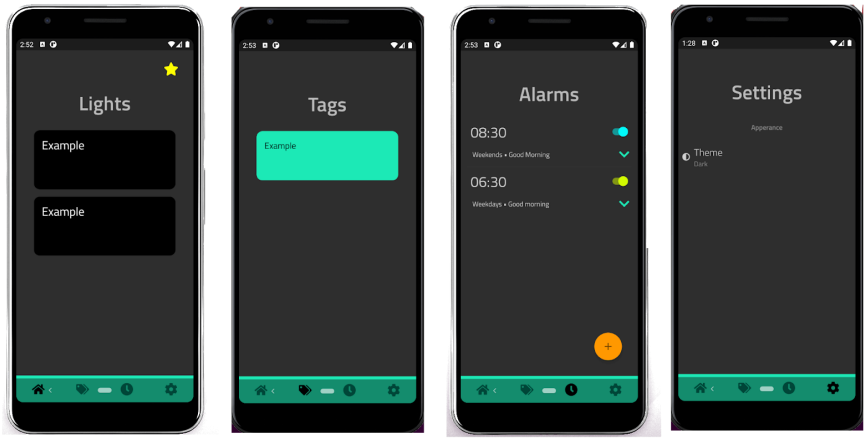


Abbildung: Die vier Reiter der DevLights App in einem Android Emulator

Heutzutage laufen viele Anwendungen ganz einfach per App auf dem Smartphone. Deswegen haben wir uns dafür entschieden eine Smartphone Applikation (App) für das Projekt DevLights zu entwickeln.
Für die Entwicklung der App wurde das Framework React Native[24] benutzt.
Dieses dient zur plattformübergreifenden Entwicklung von mobilen Applikationen. Entwickelt wurde es Facebook, wird aber zur Zeit als Open-Source-Projekt behandelt, und kann theoretisch von allen weiterentwickelt werden[25].
Das Framework baut auf dem ebenfalls von Facebook entwickelten React[26] Framework auf, einer JavaScript Bibliothek / Framework, die häufig zur Entwicklung von Benutzeroberflächen (eng. User Interfaces, kurz. UI) verwendet wird. Der Programmcode läuft auf dem Endgerät auf einem extra JavaScript Thread, der mit den nativen Komponenten, wie zum Beispiel einem Textfeld (TextView), kommuniziert. Das ermöglicht eine Entwicklung für verschiedene mobile Betriebssystem wie iOS und Android. Für die Entwicklung und Testung auf iOS ist jedoch ein Mac notwendig. Theoretisch ließe sich React Native auch auf Windows zu benutzen, praktisch wird das eher selten gemacht, da React Native hauptsächlich für mobile Betriebssysteme entwickelt wurde.

---

[24] React Native - learn once, write anywhere
[25] React Native Wikipedia.
[26] React A JavaScript library for building user interfaces

Ebenfalls lassen sich auf React Native basierende Applikationen auch in TypeScript entwickeln, da sowohl React Native als auch React kompatibel mit TypeScript sind.

Der Aufbau der App ist relativ simpel. Es gibt vier Teile der App. Diese sind zuerst die Startseite mit den Lichtern, wo sich die Lichter steuern lassen, die Kontrolle der Labels, eine Seite für Erstellung und Bearbeitung von Weckern, und die App Einstellungen. Mittels einer unteren Navigationsleiste lässt sich zwischen den einzelnen Teilen navigieren.

Die Daten, die die App benötigt, wie Lichter, Wecker und Labels, werden durch Anfragen an die API abgefragt und durch die BIbliothek React Redux[27] im “Speicher” gespeichert. Für die Anfragen an die API wird die Bibliothek axios benutzt, welche eben die Möglichkeit bietet HTTP-Anfragen zusenden[28].

Der Nutzer wählt auf der Startseite, sofern vorhanden, ein Licht aus, und kann dort den Namen, die Länge und Helligkeit des LED-Streifens ändern, das Licht ein und ausschalten, sowie die Farbe ändern. Bei den Labels gibt es zur Zeit die gleichen Funktion, mit Ausnahme der Namensänderung, da dies bei Labels nicht viel Sinn macht und der Längenänderung.

Auf dem Reiter der Wecker gibt es die Möglichkeit einen Wecker zu erstellen. Dabei wird man durch ein interaktives Menü geleitet, indem zuerst gefragt wird, auf welche Uhrzeit man den Wecker stellen möchte und man danach das Licht auswählen kann. Ebenfalls gibt es die Möglichkeit bereits vorhandene Wecker zu editieren. Dabei lassen sich der Name, die Uhrzeit, die Farbe, die Wochentage und auf welchem Licht, der Wecker läuft, editieren.

In den Einstellungen gibt es momentan nur ein Feature, welches einem ermöglicht die App entweder im Licht- oder Dunkelmodus erscheinen zu lassen.

Insgesamt bietet die Smartphone App eine gute Möglichkeit seine Lichter zu bearbeiten und zu verwalten.

3.3.2 Die Spielmodifikation

Eine weitere Anwendung, die entwickelt wurde, ist eine Modifikation für das Computerspiel Minecraft. Kurz gesagt, Minecraft ist ein Open-Sandbox Spiel, in dem man die Möglichkeit hat Sachen zu bauen und seiner Kreativität freien Lauf zu lassen[29].

---

[27] React Redux
[28] Axios Promise based HTTP client for the browser and node.js
[29] Minecraft.net

Der Vorteil den Minecraft bietet, ist dass es bei diesem Spiel eine deutlich einfachere Variante gibt, Modifikationen für das Spiel zu entwickeln, als bei den meisten heutigen Computerspielen

Für die Entwicklung bot sich die Forge Programmbibliothek[30] und deren eigener Minecraft Client, der die Modifikationen in das Spiel lädt, an, da es für diesen relativ einfach erscheint eine eigene Modifikation zu schreiben. Forge bietet den Vorteil, dass es möglich ist sowohl clientseitige Modifikationen, als auch serverseitige Modifikationen zu entwickeln.

Die Mod wurde in der Programmiersprache Java[31] geschrieben, in der auch Minecraft zu einem großen Teil geschrieben wurde.

Reagieren tut die Mod auf verschiedene Interaktionen im Spiel. Ein Beispiel hierbei wäre, dass der LED-Streifen, wenn man ein bestimmtes Biom (eine bestimmte Region) betritt, eine bestimmte Farbe annimmt, um so eine angepasste Atmosphäre zu bieten. Stirbt der Spieler im Spiel, blinkt der LED-Streifen kurz rot auf. Schläft der Spieler, geht das Licht aus, wacht er auf, geht es wieder an. Und fährt der Spieler, zum Beispiel eine Lore, dann läuft der LED-Streifen im Runnermodus durch.

Damit all diese Funktionen so auf die verschiedenen Aktionen reagieren kann, gibt es bei Forge verschiedene Events, für verschiedene Dinge die passieren, das Event für den Tod des Spielers ist das LivingDeathEvent[32], welches auf jeden möglichen Tod eines Lebenwesens im Spiel reagiert. Da wird dann noch überprüft ob der gestorbene ein Spieler ist.

Für die Kommunikation mit der API wurde eine eigene Java Klasse geschrieben, die sich nur mit Anfragen an den Server beschäftigt. Dafür wird die Bibliothek okhttp von Google benutzt, die die Möglichkeit bietet HTTP-Anfragen an einen Server zu senden.

4. Zukunftsaussichten

Für das Projekt DevLights gibt es noch viele aussichtsreiche Ideen, die man auch nach Ende des Projektes noch umsetzen könnte. Es stehen dazu momentan einige Ideen im Raum, die umgesetzt werden könnten.

Die Entwicklung einer eigenen Desktop Applikation, als Frontend Anwendung ist auch eine Überlegung, damit man die DevLights auch über seinen Computer oder Laptop steuern kann, wobei das eine eher kleinere Idee ist. Auch die

[30] Mcforge Documentation
[31] Java Oracle
[32] Class LivingDeathEvent - non official forge documentation

Erweiterung der Smartphone App um einen Entwicklermodus, mit dem man unter anderem andere eigene Muster an den LED-Streifen senden kann ist eine eher kleine Idee für die Zukunft des Projektes.
Viel größer ist dann die Idee der Implementierung eines Nutzer Systems auf dem Server, damit die Lichter einer oder mehreren Personen individuell gesteuert werden können, und einzelne Personen auch einfacher favorisierte Muster und Farben speichern können. Ein Problem, was dabei zusätzlich behoben werden würde wäre folgendes, momentan kann man, wenn man einer anderen Person im Netzwerk drin ist, alle Lichter steuern, die in diesem enthalten sind.

Nochmal übersteigen würde die Idee, das Projekt weg vom lokalen Netzwerk rein in die Cloud zu legen, um so eine Steuerung über Smart-Home Speaker, wie den Amazon Echo Dot oder den Google Assistant zu ermöglichen.
Der Amazon Echo Dot funktioniert nämlich kurz gesagt nur über die Cloud und kommuniziert fast gar nicht mit Geräten im lokalen Netzwerk.
Die DevLights wären so nämlich über diesen steuerbar und können bequem per Sprachbefehl gesteuert werden.

Das sind nur unsere Ideen für die Zukunft, gleichzeitig wollen wir das ganze Projekt Open Source auf der Entwicklerplattform GitHub zur Verfügung stellen, um die Idee der selbstständigen Weiterentwicklung durch andere Entwickler zu ermöglichen.

Das Projekt DevLights blick somit auf eine aussichtsreiche Zukunft, die scheinbar unendliche Möglichkeiten zulässt, wie sich die Lichter entwickeln.

5. Fazit

Das gesamte Projekt funktioniert im jetzigen Zustand sehr gut. Gleichzeitig gibt es noch ein großes Weiterentwicklungspotential. Es lässt sich wie bereits erwähnt noch um viele Punkte erweitern, und das sind nur wenige die genannt worden sind.
Die API bietet die Möglichkeit das bestehende System an viele bestehende Systeme anzubinden und auf bestimmte Aktionen reagieren zu lassen.

6. Quellenverzeichnis

(Die Webseiten wurden am 16.05.2021 besucht)

1. https://github.com/adafruit/Adafruit_Neopixel
2. https://www.btf-lighting.com/products/ws2812b-led-pixel-strip-30-60-74-96-100-144-pixels-leds-m
3. https://www.az-delivery.de/en/products/nodemcu
4. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/
5. https://www.mikrocontroller.net/articles/Pegelwandler
6. https://github.com/adafruit/Adafruit_NeoPixel
7. https://github.com/adafruit/Adafruit_NeoPixel#supported-chipsets
8. https://www.json.org/json-en.html
9. https://arduinojson.org/
10. https://github.com/ProjektDevLights/Hardware/blob/07297f300d8a18d569f2163201ee1f92cee5ba61/Control.cpp#L26-L99
11. https://www.computerweekly.com/de/definition/Flash-Speicher
12. https://www.arduino.cc/en/Reference/EEPROM
13. https://nestjs.com/
14. https://nodejs.org/en/
15. https://www.fastify.io/
16. https://expressjs.com/
17. https://www.typescriptlang.org/
18. https://www.mongodb.com/1
19. https://docs.mongodb.com/manual/reference/mongodb-wire-protocol/
20. https://mongoosejs.com/
21. https://de.wikipedia.org/wiki/Cron
22. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/
23. https://www.raspberrypi.org/software/operating-systems/
24. https://reactnative.dev/
25. https://en.wikipedia.org/wiki/React_Native#:~:text=React%20Native%20is%20an%20open,along%20with%20native%20platform%20capabilities.
26. https://reactjs.org/
27. https://react-redux.js.org/
28. https://github.com/axios/axios
29. https://minecraft.net
30. https://mcforge.readthedocs.io/en/1.15.x/
31. https://www.java.com/en/
32. https://skmedix.github.io/ForgeJavaDocs/javadoc/forge/1.9.4-12.17.0.2051/net/minecraftforge/event/entity/living/LivingDeathEvent.html

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen als die im Literaturverzeichnis angegeben Hilfsmittel verwendet habe. Insbesondere versichere ich, dass ich alle wörtlichen oder sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Ort, Datum	Unterschrift