

Studienarbeit von

F.Schiemann & A.Görzen

Android App LED Steuerung

Server Raspberry PI

ALED Controller

Stand: 14.05.2019

[1. Historie 3](#_Toc24623932)

[2. Einleitung 4](#_Toc24623933)

[3. Probleme bei der Umsetzung der Anwendung 4](#_Toc24623934)

[3.1 Angular 5](#_Toc24623935)

[3.1.1 Web-App starten 5](#_Toc24623936)

[3.1.2 Unsere Web-App 5](#_Toc24623937)

[3.1.3 Unsere Problematik mit Angular 5](#_Toc24623938)

[3.2 Android App 5](#_Toc24623939)

[4. Aktuelle App Umsetzung 6](#_Toc24623940)

[4.1 Einleitung Projekt 6](#_Toc24623941)

[4.2 Hardware 7](#_Toc24623942)

[4.2.1 Raspberry PI 3B+ 7](#_Toc24623943)

[4.2.2 MCU Micro Controller 7](#_Toc24623944)

[4.2.3 NodeMCU ESP8266 7](#_Toc24623945)

[4.2.4 WS2812B RGB Strip eingesetzt. 9](#_Toc24623946)

[4.3 Grundaufbau der Anwendung 9](#_Toc24623947)

[4.3.1 Programmiersprache: Dart 10](#_Toc24623948)

[4.3.2 Node-Red Framework 10](#_Toc24623949)

[4.3.3 Wled Framework 10](#_Toc24623950)

[Hier bitte nochmal kürzen 11](#_Toc24623951)

[5. NodeRED Implementierung 11](#_Toc24623952)

[5.1 NodeRED MQTT BROKER 12](#_Toc24623953)

[5.2 NodeRED Flow 12](#_Toc24623954)

[6. Quellen 15](#_Toc24623955)

# Historie

|  |  |  |
| --- | --- | --- |
| **Datum** | **Bearbeitet von:** | **Änderung** |
| 20.11.2018 | A.Görzen | Einleitung |
| 10.12.2018 | A.Görzen | Probleme bei der Umsetzung der Anwendung |
| 07.04.2019 | F.Schiemann | Ergänzungen Einleitung |
| 08.04.2019 | F.Schiemann | Aufbau Projekt |
| 15.04.2019 | F.Schiemann | NodeRED Instanz |
| 20.05.2019 | F.Schiemann | ESP8266 MQTT |
| 20.08.2019 | F.Schiemann | Wled Framework |
| 17.10..2019 | F.Schiemann | Korrektur und Verbesserung |

# 

# Einleitung

Es sollte eine Möglichkeit zur Steuerung einer RGB LED geschaffen werden.

Die Ansteuerung sollte zum einem die Möglichkeit zur Auswahl einer Farbe zulassen und zum anderen unterschiedliche ab und auf Dimm Verläufe anbieten.

Realisiert wird das durch die Verwendung einer Android Applikation. Die Android Anwendung fungiert hier bei, als Client. Dieser die Auswahl der Farbe sowie der Dimm stufen zulässt. Der Gedanke dahinter ist das die Funktion als Wecker bzw. als einschlaf Licht dient. Dem Nutzer soll dabei überlassen werden wie schnell er einschläft oder aufwacht und mit welchem Effekt bzw. welcher Farbe.

Die ausgewählten Farben sowie die ausgewählten Einstellungen (Dimm stufe und/oder Effekte) werden vom Client (Android App) zum Server (Controller) übermittelt. Hier kann durch die Verwendung von Wlan auf Kabelverbindungen verzichtet werden. Der Controller übersetzt die Eingaben des Clients und gibt sie an die Hardware weiter.

Einer der Hauptaspekte dieser Studienarbeit ist es mehr über das Zusammenspiel von Software und Hardware in Erfahrung zu bringen. Sowie die Interaktion verschiedener Programmiersprachen und Techniken besser zu verstehen und kennen zulernen.

# Probleme bei der Umsetzung der Anwendung

Es war schnell klar, dass wir die LED an einen Raspberry Pi anklemmen könnten und über die GPIO Ports die Steuerung übernehmen könnten.

Nach der Definition der Funktionen, sowie der Festlegung der Technik, wurde mit den bekannten Sprachen ausprobiert. (Java, C++ bzw Python)

Dieses noch ziemlich ohne Struktur oder einen konkreten Plan. Nach anfänglicher Zufriedenheit über das schnell voranschreiten, stellte sich die ersten großen Probleme ein.

Wir hatten im Android Studio mit Java begonnen und konnten dann über ein Python Skript auf dem Raspberry die GPIO Ports ansteueren. Allerdings gab es seitens des Skripts keine passable Möglichkeit der Rückmeldung an die App. Alles was über ein einfaches Starten des Skripts hinausging machte Probleme. Und das Kompilieren des Quellcodes der App dauert jeden Mal eine kleine Ewigkeit und führe oft im Kreis. Es Kamm dazu das die Fehleranalyse neuer Versuche bis zu 80% der Gesamtzeit in Anspruch nahm und wir einfach nicht vorankamen.

Aufgrund der Probleme entschieden wir uns, noch einmal mit den gesammelten Erfahrungen von neuem zu Starten.

Wir suchten als noch der Möglichkeit direkt auf dem Raspberry zu Arbeiten um eine direkte Kommunikation zu haben.

Wir brauchen also einen Webserver, der die Oberfläche und die Steuerung vereint. Im Mobilsegemnt kommt man mit diesen Anforderungen eigentlich nicht an Node.Js vorbei. Wir entschiedenen uns für Angular, gerade weil Angular die Möglichkeit einer schnellen Gestaltung modernere Oberflächen bietet. Und wir von Java im Hinterkopf hatten wieviel Zeit es raub eine einigermaßen Hübsche Oberfläche zur Verfügung zu stellen. Positiv war für uns an Angular das es relativ neu auf dem Markt war und die Möglichkeit bietet IOS, Android sowie alles andere mit Browser anzusprechen.

Auch hier kamen wir Anfänglich sehr gut zurecht, Anwendung war zwar etwas aufwändig zu starten. (Skripte für den Webserver, richtige NODE.Js Version) aber es gab kein Zeitraubendes Kompilieren mehr.

Allerdings musst wir mit Entsetzen verstellen das die zur Verfügung stehenden Plugins zur Ansteuerung der GPIO PORTS des Pi’s alle ohne Funktion waren. Wir versuchenden einige Zeit das Problem zu lösen, aber auch hier kam der Punkt wo wir entscheiden noch einmal einem anderen Ansatz zuflogen.

## Angular

TypeScript als Programmiersprache wird von Angular bevorzugt, dadurch ist es möglich objektorientert Programme und wiederverwendebare Bibliotheken als Templates zuschreiben. Angular unterstützt die Entwicklung von Web-Apps. Diese Apps ähneln richtigen Apps zudem sollen diese Web-Apps auch offline bedienbar sein.

Um Angular nutzen zu können wird die JavaScript-Laufzeitumgebung Node.js benötigt. Bei der Installation von Node.js wird automatisch der Node.js-Paket-Manager (NMP) installiert, der auch erforderlich ist. Der NMP ermöglicht den Zugriff auf zahlreiche kostenlose Pakete. Mit NMP wird das Kommandozeilen-Interface Angular CLI installiert, das neue Projekte anlegt, Dateien hinzufügt und andere Aufgaben bei der Entwicklung übernimmt.

Der größte Unterschied zu den gängigen Frameworks und Programmiersprachen, wäre das es nicht mehr notwendig ist das der Quellcode kompiliert wird, sondern die Anwendung in der Kommandozeilen-Interface gestartet werden kann. Alle gespeicherten Änderungen am Quellcode werden sofort bei dem refresh der Web-App-Seite angezeigt.

**Anmerkung:** Seit HTML5 sind Anwendungen im Offline-Modus möglich. Die Logik hierfür ist dann nicht unbedingt mehr auf Seiten des Servers, sondern kann im Client platziert sein. Das erfordert einen besser strukturierten Client-Code als bisher. Dafür ist AngularJS der nächste logische Schritt.

### Web-App starten

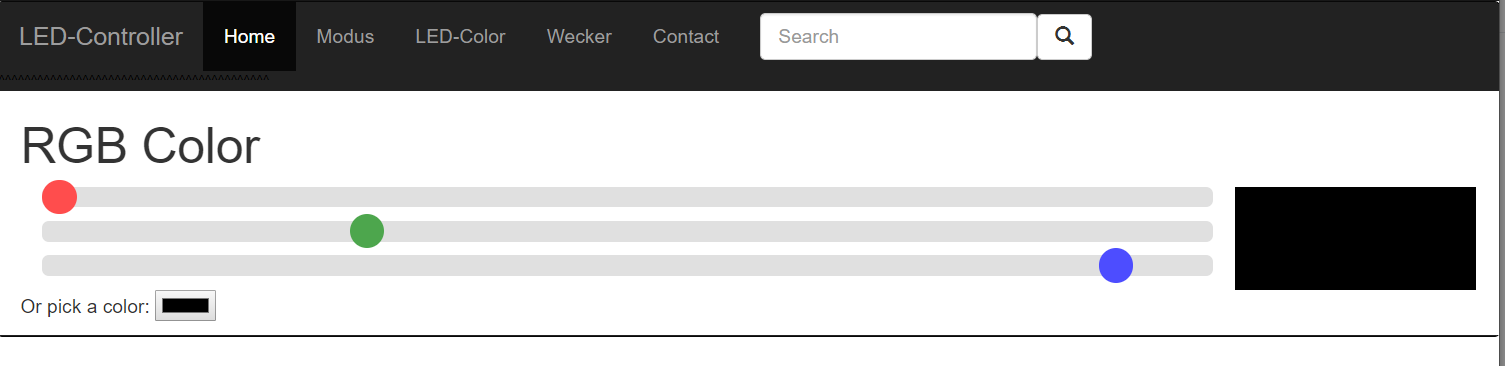
Das Projekt-Set-up wird mit dem CLI realisiert. Das CLI für Angular lässt sich über den Package-Manager npm installieren, der ein Teil von Node.js ist. Nach der npm Installation steht das CLI über den Befehl ng zur Verfügung. Durch das argument new wird dem CLI der Befehl zum generieren eines neuen Projekts im angeführten Ordner

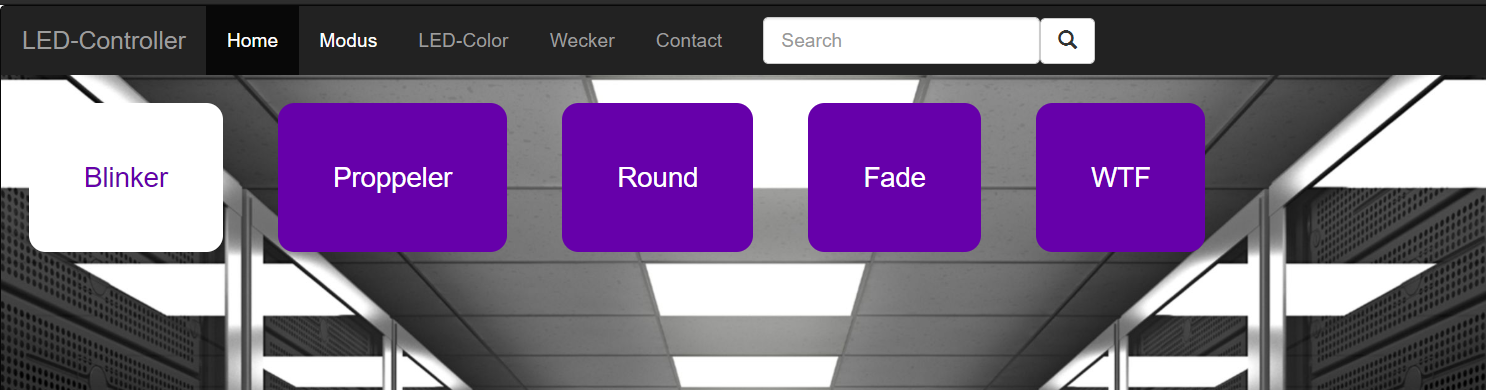
### Unsere Web-App

Unsere App hatte eine menü strucktur. Die LED konnte an und ausgemacht werden sowie einen Colorpicker.

### Unsere Problematik mit Angular

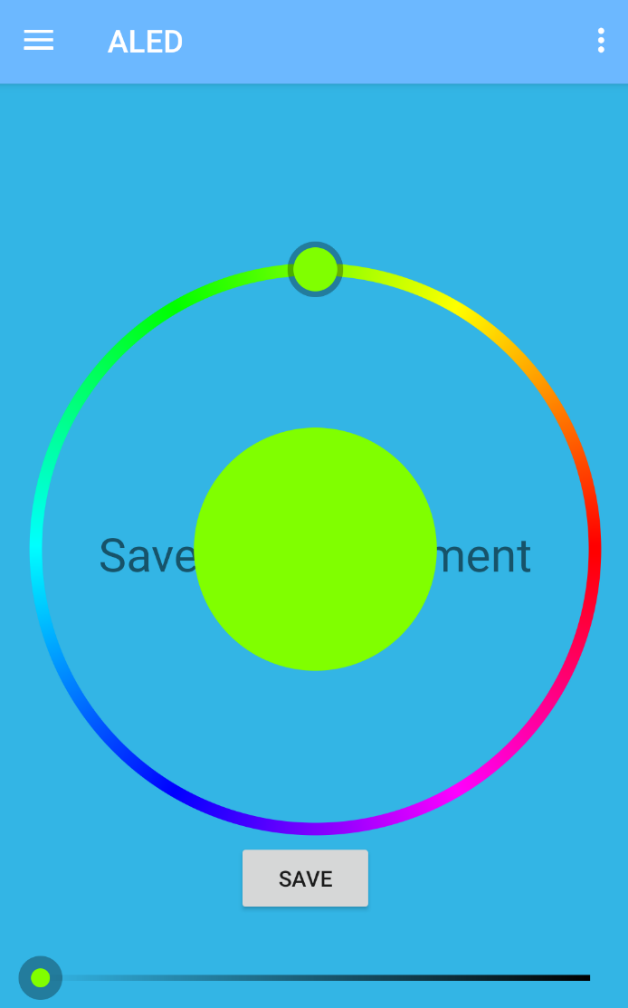
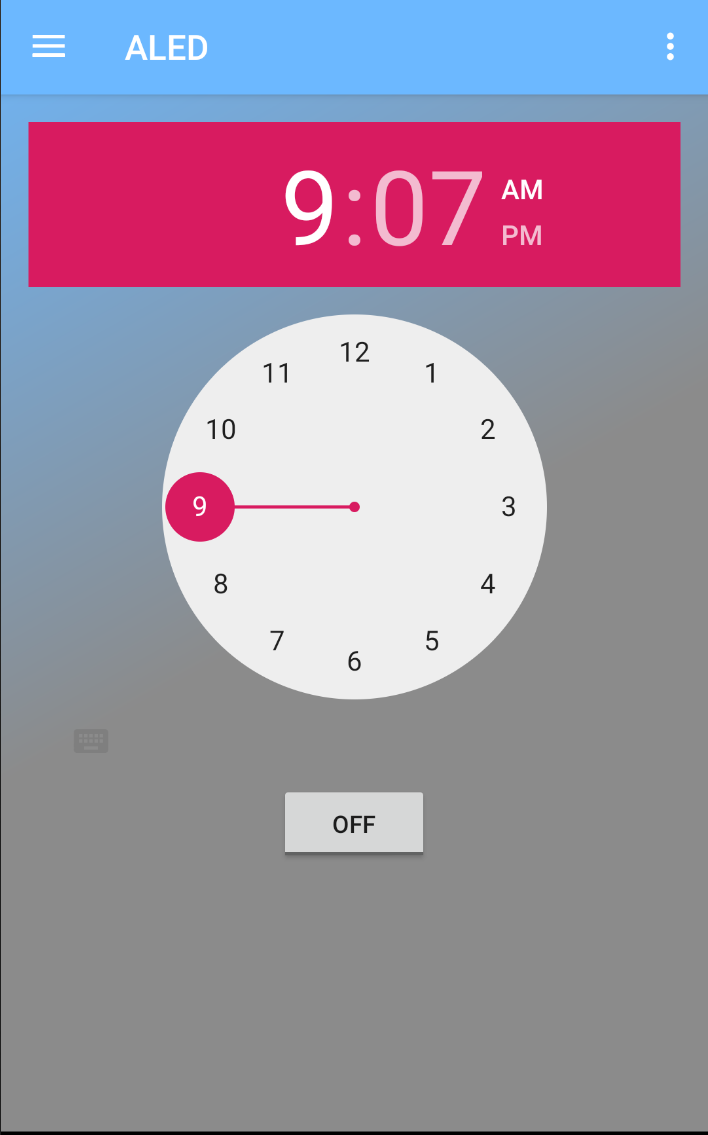
Dudy cycle gpio hat nicht funktioniert zb pwm.





## Android App

Problematik: Die gestalltung der app sehr umfangreich. Ständig neu komepllieren war super langsam und wenn ein fehler im code war ging das kompelieren nicht. Probleme im Team zuarbeiten. Viele abhändigkeiten die sehr viel zeit gekostet haben und uns immer zurück geworfen hat. Mit python probleme das man zwei programmier spracahen abstimmen musste.



# Einleitung Projekt

Die Grundelemente der App sollen kurz erläutert werden, die genau Erklärung folgt dann in den nach folgenden unteren Kapiteln.

Die Benutzer Oberfläche steht in der Smartphone App zur Verfügung, diese in Dart programmiert wurde.

Über das User UI kann der Benutzer, die IP Adresse des Servers angeben. Es wird grafisch der aktuelle Zustand der Verbindung gezeigt.

Um so schnell erkennen zu können, ob noch eine Verbindung besteht oder die Verbindung abgebrochen ist. Zudem biete das UI die Möglichkeiten, die LED an bzw. aus zuschalten. Über einen schiebe Register die Geschwindigkeit des Auf bzw. ab Dimmens zu regeln sowie einen Spezifischen Effekt Zuwahlen.

Es gibt noch einen weiteren Unterpunkt im UI in welchem der User über einen Color Piker eine Farbauswahl treffen kann, dieser beeinflusst dann die Effekte mit der entsprechenden Farbe.

Als letzten Punkt gibt es den Alarmbereich, in dem die Wecker Zeiten hinterlegt werden sollen.

Grundaufbau der Anwendung

## Hardware

Hier nochmal kurz einleitung wie diese hardware.

### Raspberry PI 3B+

Der Raspberry Pi ein Einplatinencomputer, der von der britischen Raspberry Pi Foundation entwickelt wurde. Der Rechner enthält ein Ein-Chip-System von Broadcom mit einem ARM-Mikroprozessor, die Grundfläche der Platine entspricht etwa den Abmessungen einer Kreditkarte. Der Raspberry Pi kam Anfang 2012 auf den Markt; sein großer Markterfolg wird teils als Revival des bis dahin weitgehend bedeutungslos gewordenen Heimcomputer zum Programmieren und Experimentieren angesehen. Der im Vergleich zu üblichen Personal Computern sehr einfach aufgebaute Rechner wurde von der Stiftung mit dem Ziel entwickelt, jungen Menschen den Erwerb von Programmier- und Hardware­kenntnissen zu erleichtern. Entsprechend niedrig wurde der Verkaufspreis angesetzt, der je nach Modell etwa 5 bis 35 Euro beträgt.

### MCU Micro Controller

Als Mikrocontroller (MCU) werden Halbleiterchips bezeichnet, die einen Prozessor und zugleich auch Peripheriefunktionen enthalten. In vielen Fällen befindet sich auch der Arbeits- und Programmspeicher teilweise oder komplett auf demselben Chip. Ein Mikrocontroller ist ein Ein-Chip-Computersystem. Für manche Mikrocontroller wird auch der Begriff System-on-a-Chip oder SoC verwendet.

Auf modernen Mikrocontrollern finden sich häufig auch komplexe Peripheriefunktionen wie z. B. CAN- (Controller Area Network), LIN- (Local Interconnect Network), USB- (Universal Serial Bus), I²C- (Inter-Integrated Circuit), SPI- (Serial Peripheral Interface), serielle oder Ethernet-Schnittstellen, PWM-Ausgänge, LCD-Controller und -Treiber sowie Analog-Digital-Umsetzer. Einige Mikrocontroller verfügen auch über programmierbare digitale und/oder analoge bzw. hybride Funktionsblöcke.

### NodeMCU ESP8266

Es wird ein ESP8266 von espressif verwendet. Hierbei handelt es sich um einen 32-Bit-Mikrocontroller mit Wlan. Der ESP bietet zahlreiche I/O-Ports und diverse serielle Schnittstellen, wie in der Abb. Zusehen ist. Es können alle GPIO-Ports auch genutzt werden, um ein PWM-Signal auszugeben. Das PWM Signal wurde vom Projekt in den Versionen mit einfachen RGB LED verwendend. Mittlerweile werden die I/O-Ports als serielle Schnittstelle betrieben.

Programmiert wird der NodeMCU über Arduino es kann hier C, C++, Python sowie Lua verwendet werden. Das verwendete WLED Framework ist in C++ geschrieben

Wie oben erwähnt werden die GPIO Ports als serielle Schnittstelle verwendet um den Daten Port, des LED Streifens anzusprechen.

Die Strom Versorgung des Led Streifen, sowie des NODEMCU erfolgt externes Netzteil und ist abhängig von der Anzahl der anzusteuernden LEDs.

Für unser Projekt werden aufgrund der 144 Leds zu einem Netzteil greifen welches größer mindestens 10A liefert.

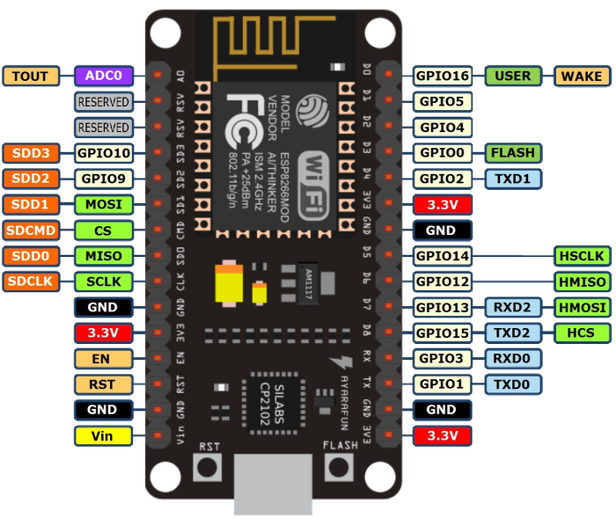


Abbildung 2 https://github.com/lvidarte/esp8266/wiki/NodeMCU

MQTT (Message Queuing Telemetry Transport)

Das MQTT Protokoll bietet viele positive Eigenschaften für den Einsatz in IOT Anwendungen.

„Trotz der Verzögerungen im Standard ist MQTT ein offenes und produktiv einsetzbares Protokoll, dessen Hauptanwendungsfälle im Bereich des Internet der Dinge liegen. Die Skalierung auf viele Tausend gleichzeitig verbundene Geräte und die Effizienz bei Übertragung und Ressourcennutzung der Geräte bieten eine Lösung für Probleme, die durch das altehrwürdige HTTP nicht realisierbar sind.“

Quelle: <https://www.heise.de/developer/artikel/MQTT-Protokoll-fuer-das-Internet-der-Dinge-2168152.html?seite=all> 17.05.2019 21:00 Uhr

**[mqttDeviceTopic]**

Send brightness

**[mqttDeviceTopic]/col**

Send color as HEX

**[mqttDeviceTopic]/api**

 Send an API call

Um unsere Befehle zu verarbeiten greifen wir auf die /api zurück.

Die /api stellt die Kommunikation mit dem LED Strip dem Node-Red zur Verfügung



### WS2812B RGB Strip eingesetzt.

Diese ist 1m lang und hat 144 LEDs. Die LED sind Einzel adressierbare bar. Über die einzelne Ansteuerung können Effekte dargestellt werden.

Die Kommunikation mit dem LED Strip wird über das Framework „Wled“ vom Aircoookie umgesetzt.

Diese steht frei über das GitHub Repository zur Verfügung.

Das Framework erlaubt es die LED anzusteuern, es hierbei möglich die Anzahl, die Helligkeit und Farbe zusetzen. Das Framework sendet seine Befehle, wie oben erläutert, über die GPIO PORTS des ESP8266.

Wir verwenden die MQTT Schnittstelle des Framework um unser Befehle zu empfangen.

Bis hier

## Grundaufbau der Anwendung

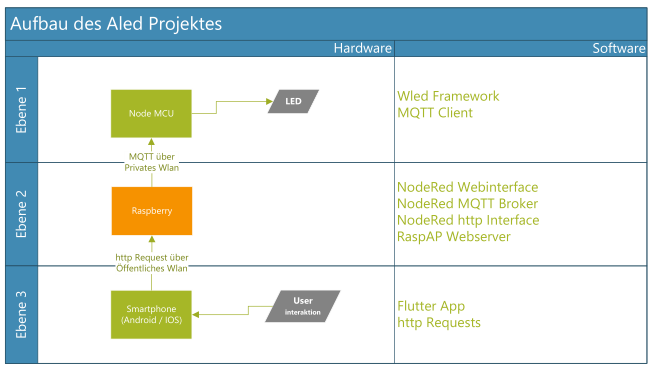


Abbildung 1 Aufbau des Aled Projektes

Wie in der Abbildung 1 zuerkennen ist, greift der Nutzer über die Ebene 3 auf das System zu.

Er kann über die App Befehle an dem Pi übergeben. Das Pi gibt diese dann an den Node MCU weiter welche dann den LED streifen steuert.

Der Node MCU hält eine private Wlan Verbindung zum Raspberry, über diese die beiden Teilnehmer eine MQTT Kommunikation aufbauen.

Der NodeMCU ist hierbei als Client anzusehen, da er nur Befehle vom Pi empfängt.

Die Physikalische Kommunikation findet über Wlan statt, es ist technisch nicht erforderliche ein privates Netz aufzuspannen. Aus Sicherheitsgründen aber durchaus sinnvoll, da so ein unbefugter Zugriff verhindert werden kann. Und nur der Pi geschützt werden muss.

Das private Wlan wird vom Pi aufgespannt, es wurde ein RaspAP Webserver verwendet, neben diesem Dienst läuft ein NodeRED Server.

Der NodeRED Server stellt die für die Kommunikation benötigte Dienste zur Verfügung. Dazu zählen zum einen ein MQTT Broker in Richtung Node MCU, sowie ein http Dienst, in Richtung der Flutter App auf dem Handy.

Die Kommunikation zwischen der App und dem Raspberry findet über das „Öffentliche“ Wlan statt. Bei bedarf kann die App aber auch uns Private Wlan und von dort Kommunizieren.

### Programmiersprache: Dart

Das Framework Flutter ist nur mit der eher selten verwendeten Programmiersprache Dart verbunden. Die neuste Dart Version ist stark typisiert. Die Syntax von Dart wurde stark von JavaScript beeinflusst. Dart unterstützt sowohl Objektorientierung sowie funktionale Programmierung. Diese Programmiersprache hat eine sehr mächtige Standardbibliothek an Funktionen. Zum Vergleich mit Java biete es einen knappen und ausdrucksstarken Code. Asynchronität ist direkt in die Standardbibliothek integriert ist ein Kernpunkt von Dart, während es bei der nativen Entwicklung häufig zu Hilfs Bibliotheken greifen muss.

### Node-Red Framework

„Node-RED ist ein von IBM entwickeltes grafisches Entwicklungswerkzeug.“

Quelle: wikipedia

NodeRed verarbeite Abfragen der Dart Applikation. Und stellt so die Kommunikation zwischen dem Smartphone und des NodeMCU her.

NodeRED biete die Möglichkeit mithilfe von Bausteinen eine Ablaufsteuerung zu generieren.

Ablaufsteuerung haben vordefinierte Verarbeitungswege um REQUEST’s handlen zu können.

Die Ablaufsteuerung werden über einen Webinterface Grafisches erstellt.

### Wled Framework

Auf dem NodeMCU wird, das WLED Framework (WiFi Lighting Effects Driver) eingesetz.um die Kommunikation zwischen LED und Hardware herzustellen.

Das WLED Framework steht unter der MIT-Lizenz zur Verfügung und kann über Github heruntergeladen werden. Und bietet die Möglichkeit bis 1500 Led anzusteuern. Die Bibliothek des verwendeten WS2812FX LED Streifens steht vollständig zur Verfügung und bietet 80 Spezial Effekte. Zudem biete das Framework die Möglichkeit des OTA (Over the air) Updates. Es werden Native JSON und HTTP Request APIs sowie eine MQTT schnittstell zur Verfügung gestellt.

Der größte Vorteil, des Framework ist, das er in neben der Verwendung in unserem Project parallel aus anderen Diensten angesteuert werden kann. Um ein Beispiel zu nennen kann nachdem der Licht Wecker angegangen ist, per Alexa abgeschaltet werden. Oder dazu verwendet werden eine Philips hue lights Steuerung anzuschalten. Um nochmal ein Beispiel zu nennen könnte das Auslösen des Weckers auch das Licht nach Zeit X einschalten umso sicher zugehen das aufgestanden wird.

## Hier bitte nochmal kürzen

# NodeRED Implementierung

Auf Raspberry PI, ist unter der Distribution Debian Stretch das Node-Red Paket installiert.

Das NodeRED Paket steht über die GitHub Seite des NodeRed Projektes zur Verfügung. Für den Verwendeten Pi gibt es ein kompiliertes Paket.

Die Instanz ist nach der Installation des node-red Package, über den Port 1880 erreichbar. Der Zugriff ist über alle aktiven Interfaces möglich. So ist es möglich im „privaten“ Wlan mit dem NodeMCU zu sein und im „öffentlichen“ mit dem Smartphone.

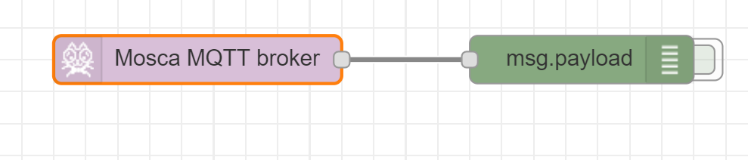
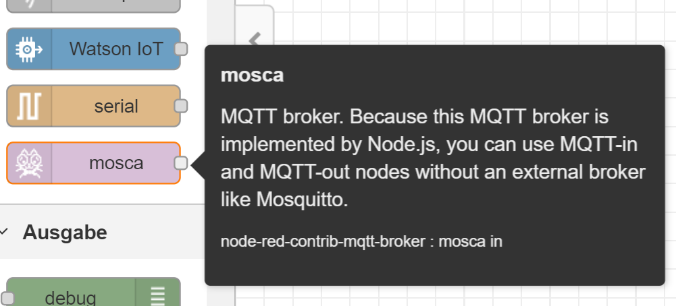
(Von uns wir das Paket von : <https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered> genutzt).

## NodeRED MQTT BROKER

Für unsere Kommunikation verwenden wir einen MQTT Broker, dieser stellt die Kommunikation zwischen Pi und NodeMCU her.

NodeRed stellt über das Mossca Framework ein out of the box Broker zur Verfügung.

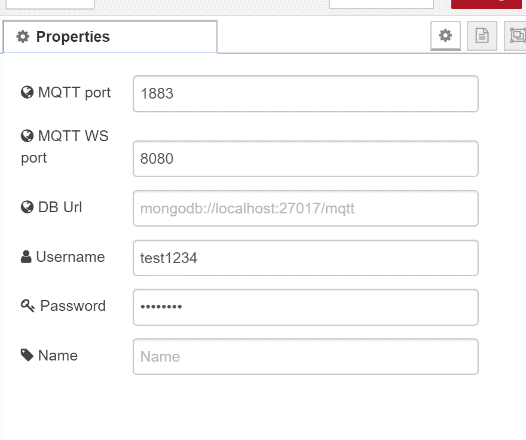
Welchen wir nur noch konfigurieren müssen.



Um MQTT auf dem PI zu verwenden ist es erforderlich einen MQTT Broker zu starten.

Der MQTT Broker stellt die Kommunikation zwischen den Teilnehmern sicher.

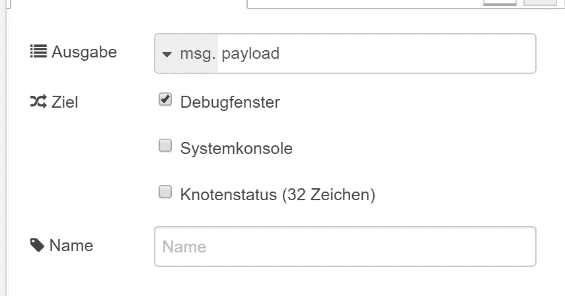
Wir verwenden für die Implementierung des Node-RED Frameworke( Mosca) , hier könnte man auch

 Dem Mosca Framework übergibt man:

Einen Zugriffsport sowie, einen Usernamen und ein

Password. Über den MQTT Port 1883 findet

die Kommunikation mit dem MCU Statt.

Sehr hielfreich ist es die Ausgabe in das Debugfenster um mögliche fehler zufinden. Nach erfolgreicher erst Impelmentierung kann diese Funktion deaktivert werden.

## NodeRED Flow

TEXTTEXTTEXTTEXTTEX

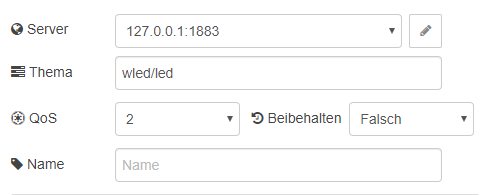
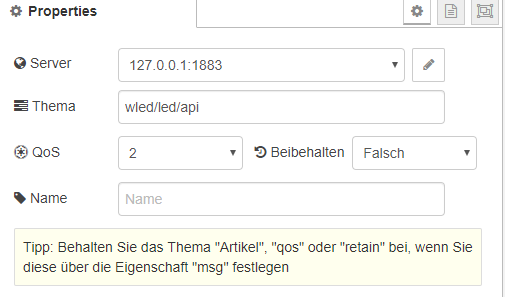
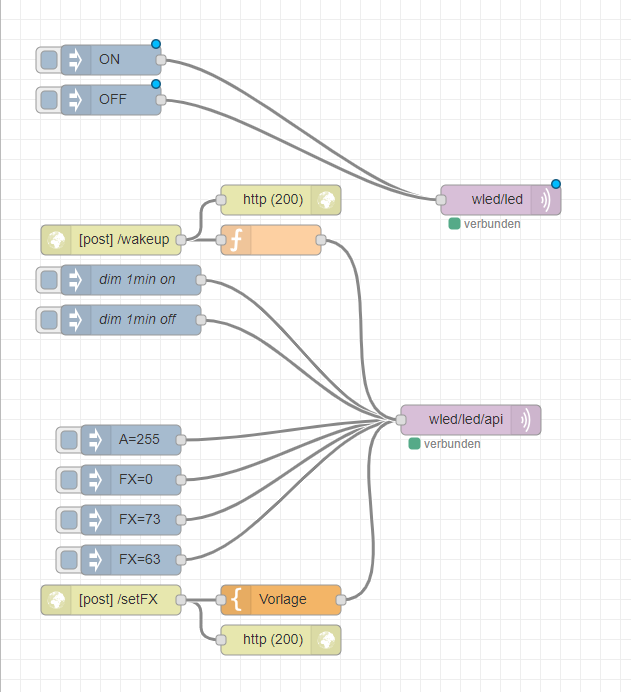


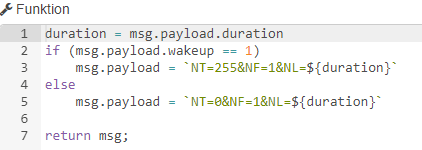
Figure 1wled/led



TEXTTEXTTEXTTEXTTEX

TEXTTEXTTEXTTEXTTEX

TEXTTEXTTEXTTEXTTEX



TEXTTEXTTEXTTEXTTEX

TEXTTEXTTEXTTEXTTEX

TEXTTEXTTEXTTEXTTEX

TEXTTEXTTEXTTEXTTEX

TEXTTEXTTEXTTEXTTEX

TEXTTEXTTEXTTEXTTEX

# Quellen

[1] Android Studio - letzter Aufruf am 10.05.2019 - <https://www.dev-insider.de/was-ist-android-studio-a-605428/>

[2] Raspberry Pi 3B+ - letzter Aufruf am 04.05.2019 - <https://www.elektronik-kompendium.de/sites/raspberry-pi/2102291.htm>

[3] Flutter Dev - letzter Aufruf am 13.05.2019 - <https://flutter.dev/>

[4] Node – RED - letzter Aufruf am 13.05.2019 - <https://nodered.org/docs/>

[5] nodeMCU - letzter Aufruf am 13.05.2019 - <https://www.mikrocontroller-elektronik.de/nodemcu-esp8266-tutorial-wlan-board-arduino-ide/>

[6] Flutter Geschichte - letzter Aufruf am 13.05.2019 - <https://google.fandom.com/wiki/Flutter>

[7] Flutter-Widgets- letzter Aufruf am 14.05.2019 - <https://docs.flutter.io/flutter/widgets/Widget-class.html>

[8] Dart-Programmiersprache- letzter Aufruf am 14.05.2019- <https://de.wikipedia.org/wiki/Dart_(Programmiersprache)>