

|  |
| --- |
| **Berry\_case** |
| Berufsübergreifendes Projektarbeit |

Eine Dokumentation über die Entwicklung ein multifunktionales Gehäuse für der Einplatinencomputer Raspberry Pi 4B.

Autor: Enrico Cirignaco  
Fachvorgesetzte: Matthias Burri   
Auftraggeber: Matthias Burri 05.12.20

**Abstract**

Die im nachfolgenden Dokument beschriebene Arbeit wurde im Ramen einem Berufsübergreifendes Projektarbeit (BüP) durchgeführt:

Die Aufgabe lautete, ein Gerät zu entwickelt, dass ermöglicht unterwegs mit dem Raspberry zu arbeiten und die Allgemeine Produktivität der Benutzer zu steigern. Daher muss eine ansteckbarer Leiterplatte und ein Gehäuse entwickelt werden. Im Gehäuse muss einen Akku integriert werden, sodass das Gerät auch ohne Netzspeisung benutzt werden kann.

Als Vorarbeit zu dieser Aufgabe wurde eine kleine Marktanalyse gemacht, um zu wissen ob ähnliche Lösungen auf dem Markt schon vorhanden sind. Dazu wurde auch die technische Machbarkeit untersucht.

Das Gehäuse ist mit einem OLED Display und ein Navigationsknopf ausgestattet. Damit können Einstellungen gemacht werden und Systeminformationen eingeblendet werden. Um der Rechner Kühl zu behalten wurde im Gerät ein 20mm Lüfter eingebaut. Die Geschwindigkeit des Lüfters kann mit einem automatischen oder einem manuellen Modus angesteuert werden. Die Kerneigenschaft vom berrycase ist die UPS Funktion (aus dem English Unterbrechungsfreie Stromversorgung). Dies ermöglich vom Akkubetrieb auf Netzbetrieb (und umgekehrt) zu wechseln, ohne das System herunterfahren zu müssen.

Am Ende der Arbeit wurden alle grundlegenden Ziele erreicht und es konnte sogar angefangen am optionalen Ziel zu arbeiten. Diese wurde aber nicht abgeschlossen. Verbesserungen, die noch implementiert werden könnten, wären das 3D Gehäuse fertig entwickeln und ausdrucken. Zudem könnte eine Grössere Akku eingebaut werden. Das Kühlungssystem könnte auch so erweitert werden, dass die CPU/GPU übertackt werden kann.

**Inhaltsverzeichnis**

[1 Software 4](#_Toc57710791)

[1.1 Node-RED 4](#_Toc57710792)

[1.1.1 Wake-up Light 4](#_Toc57710793)

[1.1.2 Light Control 7](#_Toc57710794)

[1.1.3 Resources 9](#_Toc57710795)

[1.1.4 Control Panel 10](#_Toc57710796)

[1.1.5 Power Usage 11](#_Toc57710797)

[1.1.6 Plattform abhängiges Interface 12](#_Toc57710798)

[1.1.7 Applikation URL 13](#_Toc57710799)

[1.1.8 WPS Button 14](#_Toc57710800)

**Abbildungsverzeichnis**

Abbildung 1: USB Type-C und Schutzschaltung 11

Abbildung 2: Ausschnitt aus dem PDF «Introduction to USB Type-C™» 12

**Tabellenverzeichnis**

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

**Abkürzungsverzeichnis**

**IPA** Individuelle Projektarbeit

**OLED** Organic Light Emitting Diode

**UPS** Uninterruptible Power Supply

**CPU** Central Processing Unit

**GPU** Graphics Processing Unit

**UX**  User Experience

**IP** Internet Protocol

**GPIO** General-purpose input/output

**BüP** Berufsübergreifendes Projekt

**BMS** Battery management system

**PCB** Printed Circuit Board

**PWM** Pulse-Width Modulation

**SoC** System on a chip

**PIR** Passive Infrared Sensor

**TVS** Transient-voltage-suppression diode

**PTC** Positive Temperature Coefficient  
**DC** Direct Current  
**AC** Alternating Current **ADC** Analog-Digital-Converter  
**USB** Universal Serial Bus  
**I/O** Inputs/Outputs  
**LED** Light Emitting Diode  
**uC** Mikrocontroller  
**GND** Ground  
**V** Volt  
**A** Ampere  
**IC** Integrated Circuit  
**F** Farad  
**GUI** Graphical User Interface  
**RC** Widerstand **+** Kondensator

# Einleitung

Im Rahmen einem Training für die zukünftige Berufsübergreifendes Projektarbeit soll dieses Projekt durchgeführt werden

## Aufgabestellung

Der Raspberry Pi unterwegs zu benutzen ist sehr umständlich. Ein Powerbank ist nötig und ein Netzwerk muss aufgesetzt werden. Um die UX (Nutzererfahrung) zu steigern muss ein Gehäuse für der Einplatinencomputer entwickelt und gebaut werden.

Das Gehäuse muss klein gehalten werden. Eine UPS Funktion muss implementiert werden, um im Minimum zwei Stunden im Akkubetrieb arbeiten zu können. Das Gehäuse muss mit einem Kühlungssystem für die CPU und einem kleinen Display, um die IP-Adresse anzuzeigen ausgestattet werden. Die GPIOs von Raspberry müssen zugänglich bleiben.

## Vorgehen

Während der Dauer der BüP soll das Konzept erarbeitet werden, die Elektronik entwickelt und gefertigt werden.

Zusätzlich muss die Software geschrieben werden. Folgende Arbeiten gehören zur BüP:

• Grobplanung

• Vorstudie

• Konzept erarbeiten

• Hardware Entwicklung

• Firmware / Software Entwicklung

• Testing / Bugfixing

• Dokumentation

Das Projekt wird mit der „Agilen Arbeitsmethodik“ geleitet. Dementsprechend wird in der Startphase nur eine Grobplanung gemacht und die Meilensteine werden festgesetzt. Ein SCRUM Board und das Project Management Tool „Trello“ werden eingesetzt.

Das Projekt wird mit GIT verwaltet und auf Github dokumentiert. Alle Projektunterlagen sind Open-Source. Das Arbeitsjournal wird elektronisch geführt.

## Zielsetzung

Das Projekt wurde in zwei Zielbereiche unterteilt. In einen grundlegenden und in einen optionalen Zielbereich.

### Grundfunktionen

* Batteriebetrieb (Minimum zwei Stunden Autonomie)
* Aktives Kühlungssystem
* Einschaltknopf
* Ausschaltknopf (mit soft Shutdown)
* Unterbrechungsfreie Stromversorgung (UPS)
* Batteriemanagementsystem (BMS)
* Die GPIOs vom Raspberry müssen zugänglich bleiben
* Das Gehäuse darf maximal doppelt so gross wie der Raspberry sein
* Display, um Informationen anzuzeigen
* Stromversorgung über USB C

### Optional

Die optionalen Ziele sollen erst realisiert werden, sobald die Grundfunktionen alle einwandfrei funktionieren. Optional soll mithilfe eines CAD Tools ein passendes Kunststoff Gehäuse gezeichnet und mit einem 3D Drucker ausgedruckt werden.

# Hardware

## Konzept

In diesem Abschnitt sollen die einzelnen Schaltungsteile analysiert und evaluiert werden. Hierbei ist darauf zu Achten, dass einige Teile der Schaltung von einem vorhergegangenen und Open Source Projekten übernommen wurden und daher nicht überall neue Evaluationen durchgeführt wurden.

### Mikrokontroller

Der Mikrocontroller dient in dieser Anwendung zur Ansteuerung der einzelnen Hardwarekomponenten und soll auch die Webapplikation hosten.

#### Vorgaben

Der Kontroller muss folgende Punkten erfüllen:

* Auf dem Kontroller muss ein Webserver laufen können
* Wi-Fi Verbindung
* PWM zu Ansteuerung der LED
* Geringe Leistungsaufnahme da der Kontroller 24/7 laufen wird
* Es werden insgesamt 8 I/Os (Inputs/Outputs) benötigt, um die Hardwareteile anzusteuern.

#### Evaluation

Es wurde entschieden ein Einplatinencomputer (SoC) der britische Hersteller Raspberry Pi einzusetzen, damit die Software auf bestehende Open-Source Projekte aufgebaut werden kann. Alle in Abschnitt 2.1.1.1 erwähnten Punkte werden vom Raspberry Pi Zero W erfüllt. Zudem dieses Model ist der billigste und kleinste im Sortiment.

Der gewählte Kontroller besitzt folgende Eigenschaften:

* 1GHz, single-core CPU
* 512MB RAM
* 802.11 b/g/n Wireless LAN
* Mini HDMI and USB On-The-Go ports
* Micro USB power
* 40-pin I/Os Header

### LED-Treiber

LEDs sind Stromabhängiges Bauteilen und somit können mit einer Stromquelle angesteuert werden. Darum wurde es entschieden die vier-farbige LED mit vier linearem Konstantstromquelle-Regler anzusteuern. Der BCR420UW6 von der Hersteller DIODES erfüllt alle Anforderungen mit eine maximale Ausgang Strom von 350mA und Ansteuerung mit ein PWM Signal.

### Bewegungssensor

Es wurden verschieden Lösungen in Betracht genommen, um eine Bewegung auf kürzeren Distanzen zu detektieren. Man ist zum Schluss gekommen, dass Passive Infrarote Sensoren (PIR) sich am besten eignen für diese Anwendung. Um die Komplexität zu reduzieren wurde ein digitale PIR ausgewählt. Der EKMC1601111 vom Hersteller Panasonic mit Integrierte Auswertung-Schaltung und Fresnel-Linse wurde auswählt.

### Strom- und Spannungsmessung

Die Spannung kann einfach mit einem Spannungsteiler konvertiert werden und mit einem ADC ausgewertet werden. Der Strom muss zuerst in eine Spannung umgewandelt werden und danach analog zu der Spannung Messung weiterverarbeitet werden. Die Konvertierung Strom-Spannung erfolgt mithilfe einem Shunt Widerstand. Da der Raspberry Pi nicht mit einem ADC ausgestattet ist, muss eine Externe evaluiert werden. Für die Kommunikation zwischen ADC und uC soll das Protokoll I2C angewendet. Der MCP3426 vom Hersteller Microchip wurde ausgewählt.

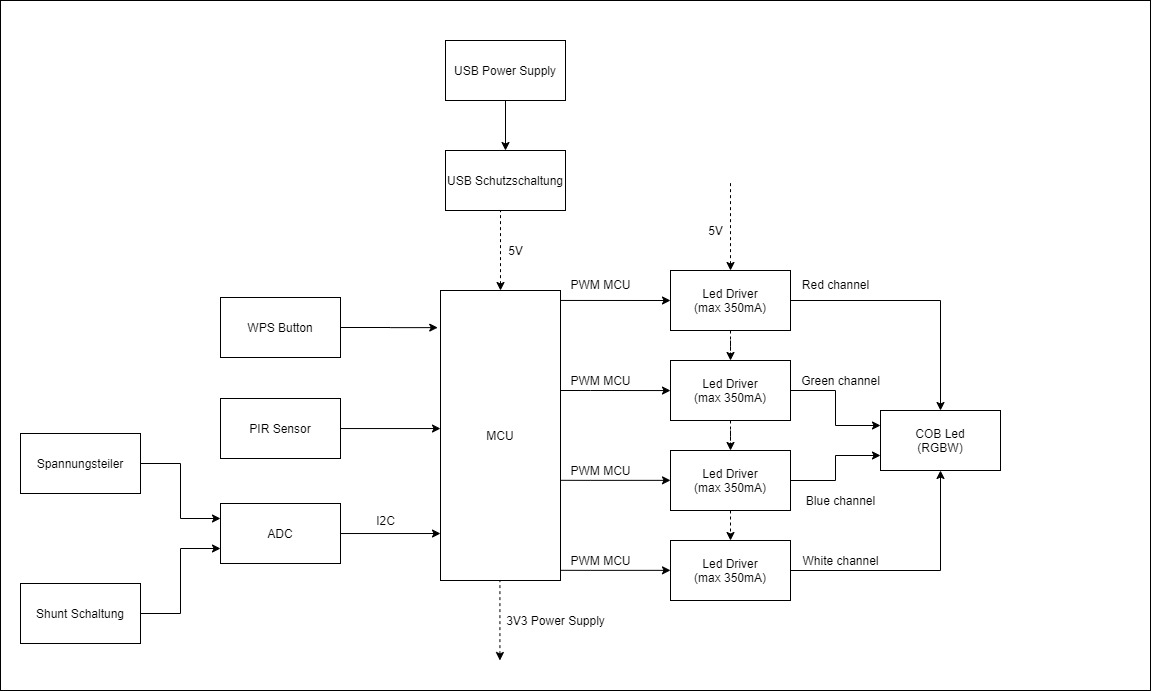
Der gewählte Kontroller besitzt folgende Eigenschaften:

* 16bit Auflösung
* 2 Differenzeingänge
* Delta-sigma verfahren
* I2C Schnittstelle

## Realisierung

In diesem Abschnitt wird auf die Realisierung der Hardware genauer eingegangen und die einzelnen Schaltungsblöcke werden genauer erklärt.

### Blockschaltbild



### Mikrokontroller

Der Mikrocontroller ist der Kern der Hardware und steuert alle anderen Baugruppen. Integriert auf dem Board ist auch die 3V3 Speisung.

### Speisung

Das Gerät ist über USB gespiesen. Folgt die Speisung ist schon vorhanden. Was noch implementiert worden ist, ist eine Schutz Schaltung, um das Gerät gegen Überspannungen und Überlastungen zu schützen. Diese Schaltung wurde mehr oder weniger von dem Open Source Schema von der Raspberry Pi 4B übernommen. Der TVS Diode D2 Schutz gegen Überspannungen. Der Widerstand R11 ist ein PTC Rücksetzbare Sicherung, die bei einem Strom von 2.5A oder höher ihr interne Widerstand verhindert und konsequent die Schaltung trennt.

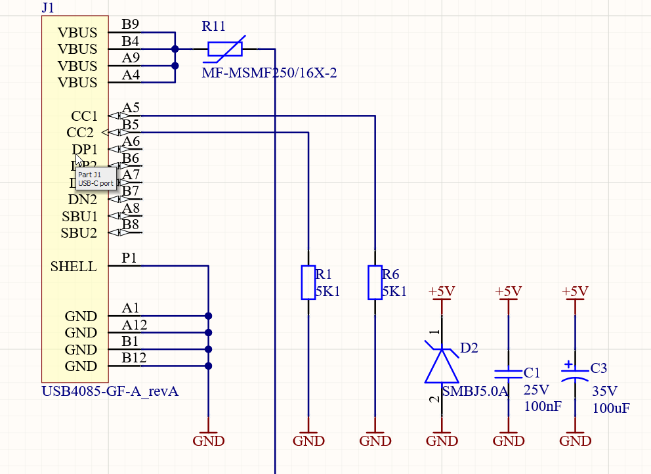


Abbildung 1: USB Type-C und Schutzschaltung

Wie im Dokument «Introduction to USB Type-C™» von Microchip empfohlen wird wurden zwei 5k1 Widerstände zwischen Ground und Pin CC1 und CC2 von der USB Buchse bestückt. Diesen stellen sicher, dass das Gerät mit genügend Leistung von der USB-Speisung versorgt wird. Die Zwei Kondensatoren C1 und C2 sind da, um Allfälliges Störungen von der Speisung herauszufiltern.

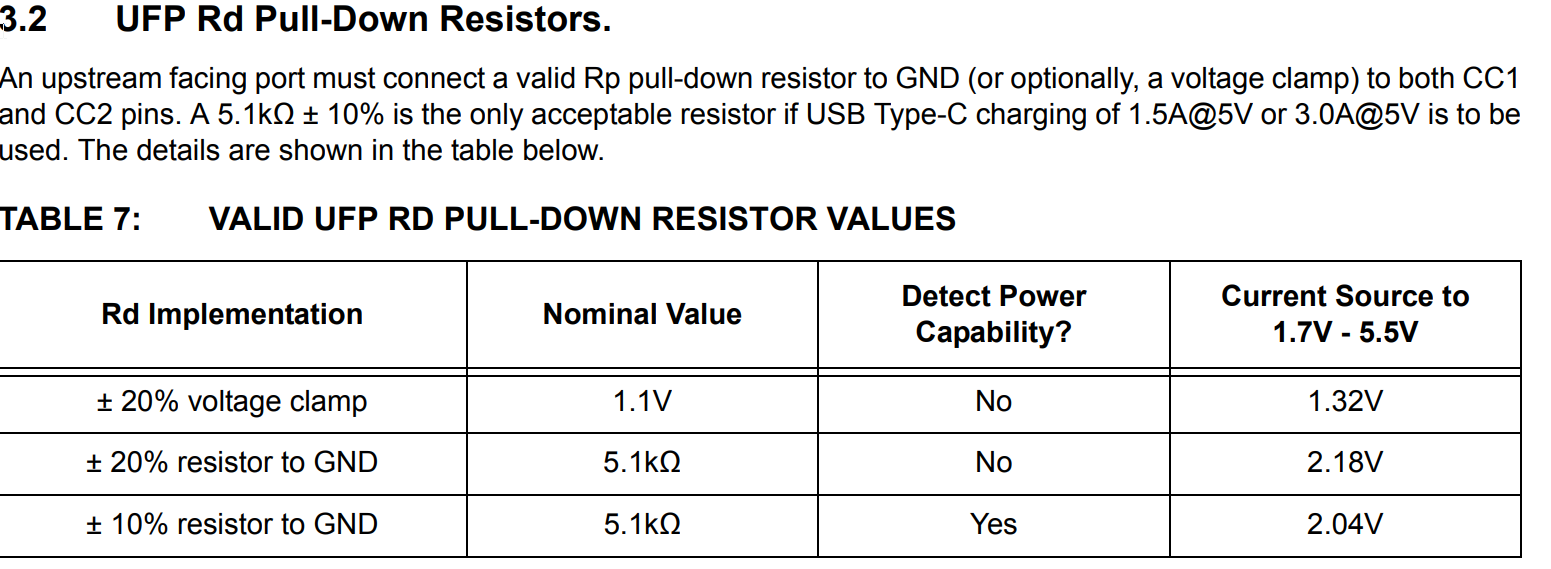
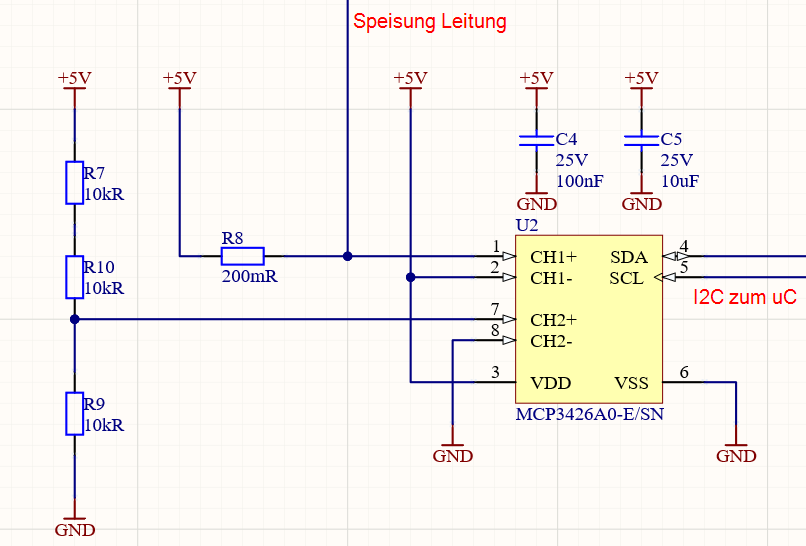


Abbildung 2: Ausschnitt aus dem PDF «Introduction to USB Type-C™»

### Strom und Spannung Messung

Die Messung der Speisespannung erfolgt über dem Spannungsteiler (R7-R9-R10). Da die drei Widerstände genau der gleiche Werte haben (±1%), ist die Ausgangspannung gleich die Eingangspannung geteilt durch drei. Diese Pegel ist dann von der ADC ausgewertet. Der ADC arbeitet mit eine Interne Spannungsreferenz von 2,048V. Die maximale Spannung an den Differenzeingänge darf diese Wert nicht überschritten.

//beschreibung strommessung



# Software

In diesem Abschnitt der Dokumentation sollen die einzelnen Teile der Software analysiert und erklärt werden.

## Node-RED

Die Software wurde mithilfe der grafisches Entwicklungswerkzeug Node-RED realisiert. Die web Applikation so wie die Steuerlogik wurden in diese Umgebung implementiert. Das Projekt ist modular aufgebaut und in verschiedene Flow organisiert. Jede Feature ist in eine eigene Flow implementiert.

Zwei Versionen der Web Applikation sind vorhanden. Ein Desktop und eine Mobile Version. Wenn man auf die Hauptseite kommt, wird es ausgelesen auf was für ein Gerät die Webseite geöffnet wurde und die entsprechend Seite wird geladen.

### Wake-up Light

Die Wake-up light ist eine Feature der smart\_lamp, dass der natürliche Sonnenaufgang simuliert indem die Lichthelligkeit langsam gedimmt wird und die Farbtemperatur geändert wird. Die Wake-up light soll helfen sanfter aufzuwachen und den Tag in besserer Stimmung zu beginnen

#### GUI / Frontend

Die GUI ist so aufgebaut das am Anfang eine kurze Einleitung ist. Hier wird es erklärt wie das Tool funktioniert. Danach sind drei verschiedene Wecker Module vorhanden. Jeder Wecker kann komplett unabhängig eingestellt werden. Jede Wecker Modul bestehet aus ein ein/aus Schalter, ein Time Picker, um die Zeit auszuwählen und ein Dropdown Menü, um die Tage der Woche auszuwählen. Die Einträge werden automatisch in ein lokales Json File gespeichert und bei Bedarf wieder ausgelesen.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

#### Backend

Ein Bild, das Text, Karte enthält.

Automatisch generierte Beschreibung

Der aktuelle Wochentag, Stunde, Minute, Sekunde und Anzahl vergangene Sekunden seit Mittenacht werden in Sekundentakt aktualisiert und in die dazugehörige Variabel gespeichert. Der Funktions-Node «interrupt\_alarm» zurückgibt wie viele Sekunden vergangen sind seit dem Auslöser. Das wird in folgendes weise gemacht:

**Funktion-Node: interrupt\_alarm**

1. // if the manual Overrite switch is on never trigger interrupt
2. if(!manual\_sw)
3. {
4. // check all threee alarm modules
5. for(var i=0; i<3; i++)
6. {
7. // return true if today weekday is contained in the array alarm\_day\_01
8. var isWeekday = alarm\_day[i].includes(weekday);
9. // calculate elapsed time
10. var elapsed\_time = time\_s - (alarm\_time[i]-1800);
12. //check if alarm modul is activated
13. if(alarm\_sw[i])
14. {
15. // trigger interrupt if elapsed time is between 1 and 1800
16. if(elapsed\_time > 0 && elapsed\_time <= 1800 && isWeekday)
17. {
18. msg.payload = elapsed\_time;
19. return msg;
20. }
21. }
22. }
23. }

 Der Output wird von zwei weitere Funktions-Node gefangen und verarbeitet. Dies Kommunizieren direkt mit dem Flow zuständig für das Steuern der Leds (light\_control flow). Der Funktions-Node «control\_switches» ist zuständig für das Steuern der Led-Hauptschalter und die Weise Led-Schalter. Ganz am Anfang muss sichert gestellt werden, dass der Hauptschalter ein ist. Am Schluss, wenn das Wecker läutet sollte, wird auch die Weisse LED eingeschalten. Die Zweite Node «curves\_functions» ist zuständig für das Rechnen der RGB Wert in Funktion der vergangenen Zeit. Jede RGB Farbe wird mit einer Arctan-Funktion berechnet. Nur die Funktion für die rote Farbe bestehet aus einem Polynom aus ein Arctan Funktion und ein eine Funktion der 0 Grades (konstant).

**Funktion-Node: scurve\_function**

1. var time\_elapsed = msg.payload;
2. var rgb\_obj = {"r":0, "g":0, "b":0};
4. // Curve of red led, part 1--> function range between 0 and 1080
5. if(time\_elapsed < 1080)
6. {
7. rgb\_obj.r = Math.round((Math.atan((time\_elapsed\*0.004)-2.15)\*112)+126.5);
8. }
9. // Curve of red led, part 2 --> function range between 1080 and 1800
10. if(time\_elapsed >= 1080 && time\_elapsed <= 1800)
11. {
12. rgb\_obj.r = 255;
13. }
14. // Curve of green led --> function range between 360 and 1800
15. if(time\_elapsed > 360 && time\_elapsed <= 1800)
16. {
17. rgb\_obj.g = Math.round((Math.atan((time\_elapsed\*0.004)-4.3)\*70)+87);
18. }
20. // Curve of blue led --> function range between 1080 and 1800
21. if(time\_elapsed > 1080 && time\_elapsed <= 1800)
22. {
23. rgb\_obj.b = Math.round((Math.atan((time\_elapsed\*0.006)-8.6)\*21)+25);
24. }

27. msg.payload = rgb\_obj;
28. return msg;

#### Kennlinien

Die mathematischen Funktionen würden mithilfe vom Tool Geogebra zusammengesetzt. Gegeben ist, dass der Sonnenaufgang genau 30 Minuten dauern muss und dass die Leds ein Wert zwischen 0 und 255 annehmen können. Die x-Achse beschreibt die vergangene Zeit seit dem Auslöser in Sekunden. Die y-Achse der Led-Wert. Es gibt vier Kurven, eins für jede Grundfarbe und eine für diese Summe alle Farben. Mit Schieben und strecken können die Kurven verändert werden.

Angewendete Formeln:

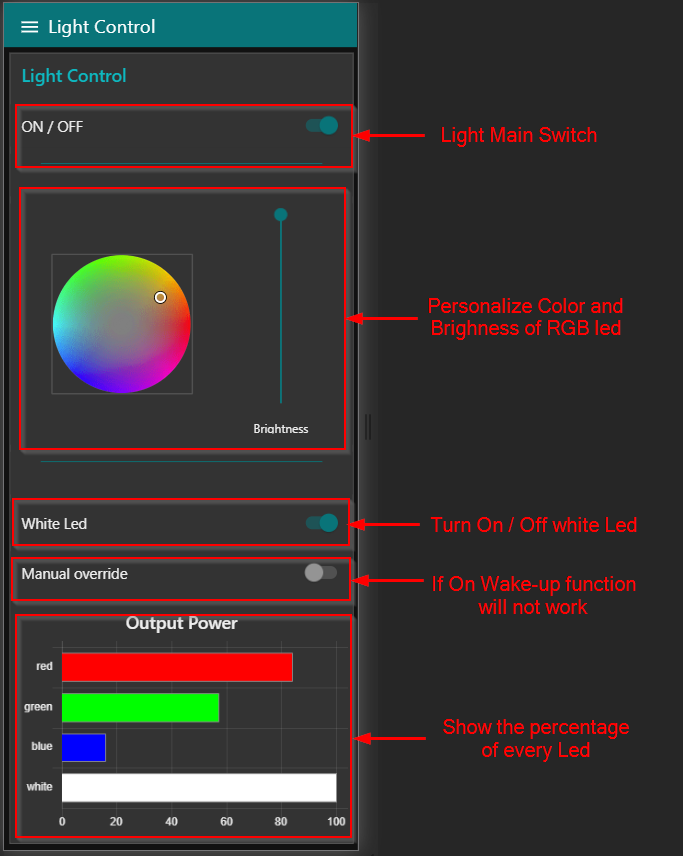
Ein Bild, das groß, Tisch, Wasser, Mann enthält.

Automatisch generierte Beschreibung

### Light Control

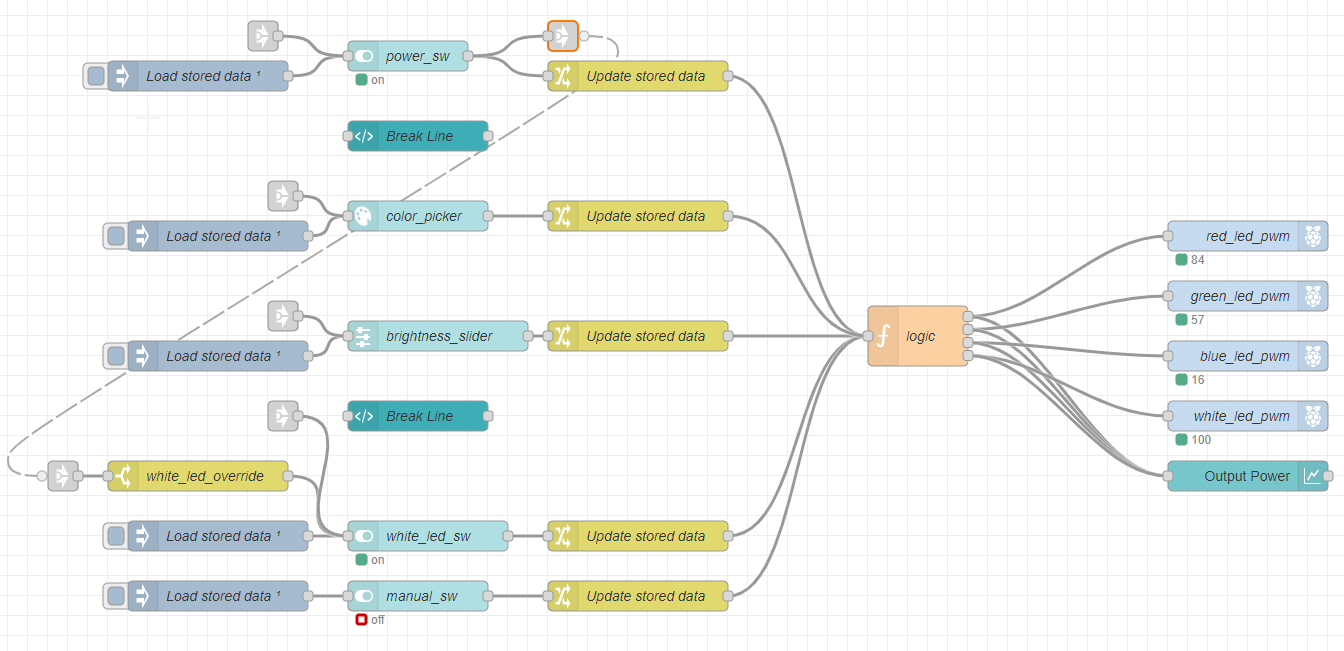
In dieses Panel kann die Lichtquelle angesteuert und personalisiert werden.

#### GUI / Frontend



Die GUI bestehet aus fünf Elemente. Zuerst ist ein Hauptschalter für das Light. Damit kann das Licht schnell ein- und ausgeschalten werden. Dann folgt ein Rundes Color Picker und ein Schieber, um die Helligkeit einzustellen. Die Bedingung ist recht «straightforward». Mit dem zweiten Schalter kann die Weisse Led ein- und ausgeschaltet werden. Die «Override» Schalter ist da um den Zugriff der Wake-up Funktion auf die Led zu verweigern oder gewährlisten. Das kann nützlich sein um die Wake-up Funktion temporär auszuschalten oder die Led während der Sonnenaufgang-simulation manuell kontrollieren zu können. Zuletzt ist ein Graph die zeigt mit wie viel Prozentual-Leistung Jede Led angesteuert wird. Entspricht der Duty Cycle Wert.

#### Backend



Die graue Eingangsschnittstellen ermöglichen das Steuern der Leds via Nodes die in andere Flows sind. In diesem Fall erlauben sie, dass Nodes der Wake-up Flow während der Simulation der Sonnenaufgang die Steuerung übernehmen können. Es gibt auch eine Interne Schnittstelle (die gestreichelte Linie). Zusammen mit dem «white\_led\_override» Node bewirken, dass die Weisse Led sich automatisch ausgeschaltet wird, wenn der Hauptschalter ausgeschaltet ist. Der Aufbau der Bedienelemente ist ähnlich wie in der Wake-up Flow. Die eingetragenen Daten werden Lokal gespeichert und beim Neustart werden vom Backup wieder hochgeladen. Die vier ausgangs-nodes steuern direkt die GPIO von Raspberry. Sie erzeugen ein PWM Signal. Der Eingangswert entspricht der Duty Cycle in Prozent. Die gleichen Werten werden gleichzeitig auf ein Graph angezeigt.

1. if(led\_power\_sw === true)
2. {
3. msg\_red.payload = Math.round(rgb\_led\_color.r\*led\_brightness);
4. msg\_green.payload = Math.round(rgb\_led\_color.g\*led\_brightness);
5. msg\_blue.payload = Math.round(rgb\_led\_color.b\*led\_brightness);
6. }
7. else
8. {
9. msg\_red.payload = 0;
10. msg\_green.payload = 0;
11. msg\_blue.payload = 0;
12. msg\_white.payload = 0;
13. }
15. if(white\_led\_sw === true)
16. {
17. msg\_white.payload = 100;
18. }
19. else
20. {
21. msg\_white.payload = 0;
22. }
24. return [msg\_red, msg\_green, msg\_blue, msg\_white];

 Die Aufgabe der Logic Node ist ziemlich klar definiert. Schalte ein die RGB Led, wenn der Hauptschalter ein ist. Schalte ein der Weisse Led, wenn der Weisse Led-Schalter ein ist. Zurückgibt der Wert von jeder Farbe auf einen separaten Ausgang. Der ausgangswert (PWM zwischen 0 und 100) wird berechnet indem der RGB Wert (0-255) mit dem Helligkeitsfaktor multipliziert wird. Dies liegt zwischen 0 und 0.39 so dass beim ein RGB Wert von 255 am Ausgang 100 ergibt.

### Resources

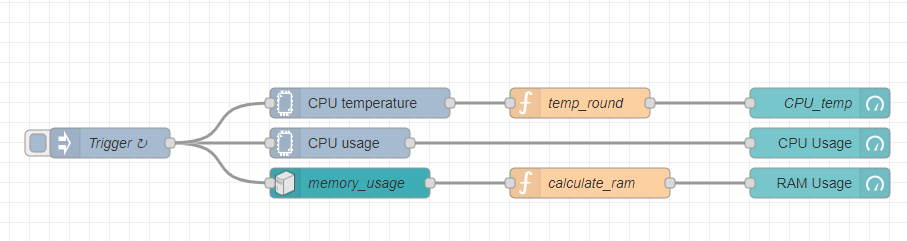
In diese Panel Nützliche System Informationen werden dargestellt.

#### GUI / Frontend

Ein Bild, das Stereo enthält.

Automatisch generierte Beschreibung

#### Backend

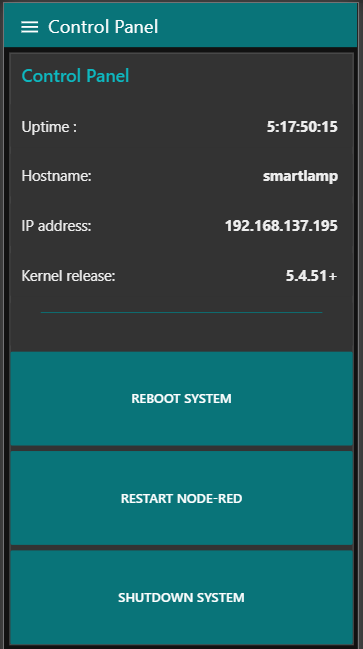


Für die Umsetzung dieser Flow zwei Wichtige externe Module wurden eingesetzt: node-red-contrib-cpu und node-red-contrib-os. Die Informationen wurden dann auf Anzeigen dargestellt. Die Anzahl besetzte Arbeitsspeicher in Mb musste zuerst berechnet werden (gesamte Speicher – freie Speicher).

### Control Panel

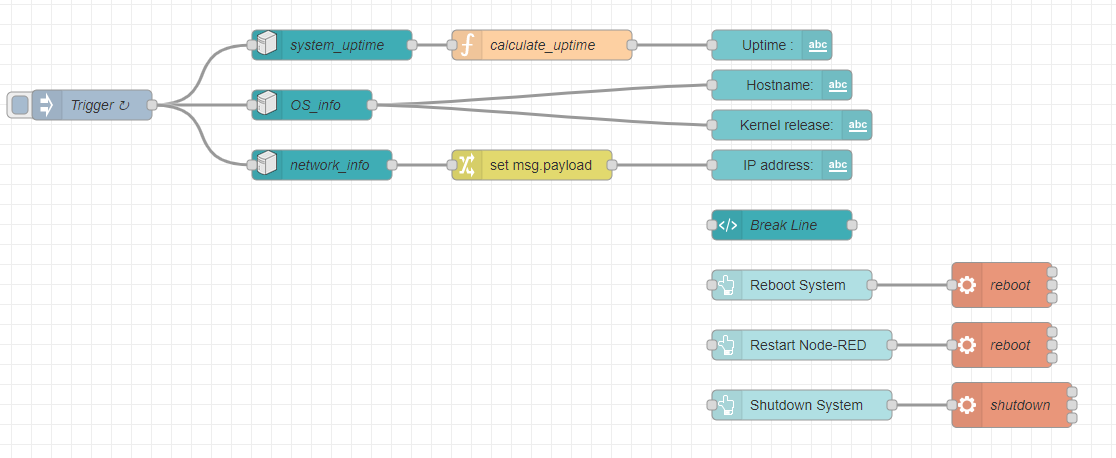
In dieses Panel werden Systeminformationen angezeigt. Das System kann von hier auch gesteuert werden.

#### GUI / Frontend



Die GUI ist selbsterklärend. In dem oberen Teil werden einige Systeminformationen angezeigt. Seit wie lange das System Aktiv ist, Hostname, IP-Adresse, und die aktuelle Linux Version. In dem unteren Teil sind Drei Knopfe vorhanden. Von hier kann der User das System neustarten und herunterfahren sowie auch nur Node-RED neustarten.

#### Backend

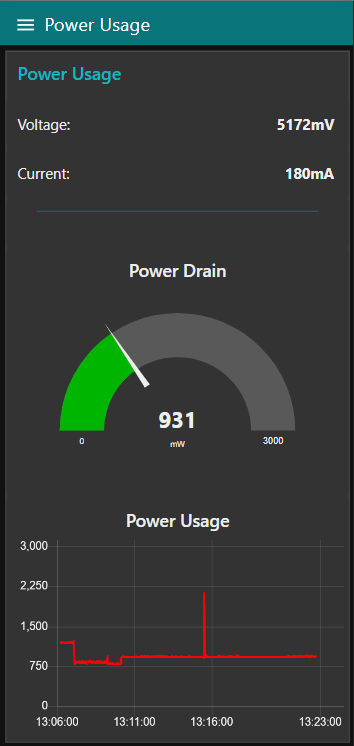


Die Umsetzung wurde durch ein Externe Modul vereinfacht (node-red-contrib-os). Mithilfe von spezifische Nodes werden mit Sekundentakt die Systeminformationen abgefragt und auf das Dashboard angezeigt. Die Uptime wird in Sekunde zurückgegeben und muss darum noch in eine besser lesbare Form umgewandelt werden. Die Drei knöpfe sind mit «Exec» Nodes verbunden. Sie ermöglichen das Ausführen eine Commando in die Linux Bash. Die «Exec» Node ist im Modul node-red-contrib-func-exec zur Verfügung gestellt. Sind respektive angewendete Bash Befehle sind: «sudo reboot», «sudo shutdown now» und «node-red-restart».

### Power Usage

In dieses Panel werden Stromaufnahmen, Speisespannung und Leistungsaufnahmen angezeigt.

#### GUI / Frontend



Zu Oberst sind die gemessene werten für Spannung und Strom angezeigt. Danach ist mit ein die die Berechnete Leistungsaufnahme zeigt. Unten ist den Verlauf der Leistungsaufnahme der letzten Stunde in Milliwatt.

#### I2C AD-Wandler

Um Spannung und Strom zu messen benötigt man ein Analog-digital Wandler. Auf dem Raspberry Pi ist aber keine vorhanden. Um die Messungen trotzdem durchführen zu können wurde ein dual Chanel ADC auf dem PCB eingebaut. Der externe ADC kommuniziert mit dem Raspberry über I2C. Es handelt sich um das MCP3426 von Microchip. Der Chip konvertiert das analoge Signal von zwei verschiedenen Kanälen mit dem «delta-sigma» Verfahren mit bis zu 16bit Auflösung. Eine interne Spannungsreferenz ist auch vorhanden (2.048V ± 0.05%).

Die Inbetriebnahme dieser Chip ist sehr einfach. Um eine Messung zu machen muss eine «Read» Vorgang mit dem I2C Protokoll starten und gleichzeitig ein «Configuration Byte» übertragen. Zurück bekommt man zwei «Data Byte» und der «Configuration Byte».

Für die Messung der Spannung wurden folgende Einstellungen eingesetzt:

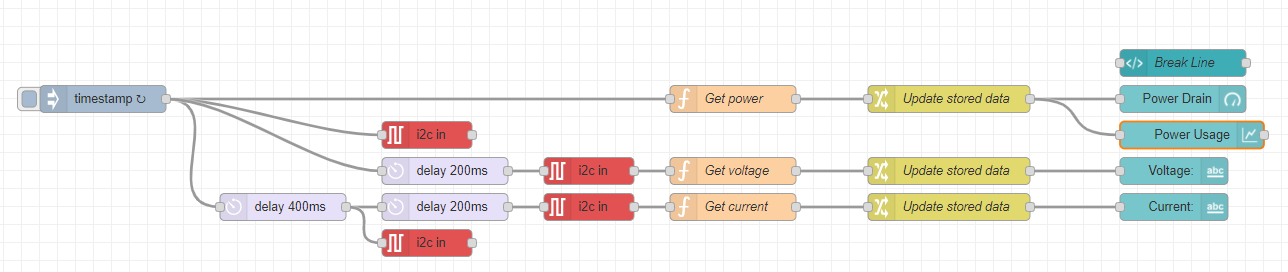
* Channel: 1
* Mode: Continuous
* Sample Rate: 240SPS (12bits Auflösung)
* PGA(Programmable Gain Amplifier) : x1

Für die Messung des Stromes folgende Einstellungen:

* Channel: 0
* Mode: Continuous
* Sample Rate: 15SPS (16bits Auflösung)
* PGA(Programmable Gain Amplifier) : x8

Nachdem die Analoge Signale ausgelesen worden sind, müssen Strom, Spannung und Leistung berechnet werden. Folgende Formel wurden angewendet.

#### Backend



Jede Sekunde wird ein Interrupt ausgelöst, die Messungen werden gestartet und danach werden die Resultate im Dashboard angezeigt. Da die zwei Messungen nicht gleichzeitig erfolgen können, ist eine Verzögerung von 400ms dazwischen. Der Verlauf der Leistungsaufnahme in den Letzte Stunde wird in ein Diagramm angezeigt.

### Plattform abhängiges Interface

Das System soll erkennen ob der User auf eine Mobile oder Desktop Plattform das Dashboard anschauen möchte. Und danach soll die entsprechende Seite laden. Die Mobile Seite wird standardmässig geladen. Folgende Funktion ist im Head von HTML code hinterlegt:

1. <script>
2. if(!( /Android|webOS|iPhone|iPad|iPod|BlackBerry|IEMobile|Opera Mini/i.test(navigator.userAgent)) ) {
3. window.location.href = "http://smartlamp/#!/6";
4. }
5. </script>

Diese IF-statement prüft ob der User auf ein Mobile oder Desktop Browser die Seite geöffnet hat. Wenn es sich um ein Desktop Version handelt, der User wird sofort zur Desktop Seite umgeleitet. Die Umleitung geschieht mehr oder weniger sofort, und darum merkt der User nichts davon.

#### Desktop GUI

Das Node-RED Projekt wurde in der ersten Linie für die Darstellung auf mobile Geräte entwickelt. Die Webseite ist in fünf Tab aufgeteilt. Jeder Tab ist so organisiert, dass den ganzen Inhalt auf dem Bildschirm eines Modernes Smartphones dargestellt werden kann. Dieses Format eignet sich aber nicht gut für die Darstellung auf einen grösseren Bildschirm. Eine alternative Webseite muss also aufgebaut werden. Der ganze Inhalt ist in einen einzigen Tab, in fünf verschieden Gruppe organisiert. Die Bedingung ist genau die gleiche geblieben wie in die Mobile Version.

Ein Bild, das Monitor, Screenshot, Bildschirm, sitzend enthält.

Automatisch generierte Beschreibung

### Applikation URL

Der User soll auf die Webseite zugreifen können auch wenn der IP-Adresse des Geräts nicht bekannt ist. Die Idee ist, dass nur der Hostname (smartlamp) eingetippt werden muss, um auf das Dashboard seit zu kommen. Um dies zu ermöglichen, zuerst muss der TCP Port auf dem Node-RED löst zu der Standard HTTP Port (80) geändert werden. Diese Einstellung kann in das Settings.js File getätigt werden.

1. // the tcp port that the Node-RED web server is listening on
2. uiPort: process.env.PORT || 80,

 Danach muss die relative Path von der Node-RED UI und das Dashboard eingestellt werden.

1. // By default, the Node-RED UI is available at http://localhost:1880/
2. // The following property can be used to specify a different root-path.
3. // If set to false, this is disabled.
4. httpAdminRoot: '/admin',
5. // If you installed the optional node-red-dashboard you can set it's path
6. // relative to httpRoot
7. ui: { path: "" },

 Da die TCP Port 80 zu den Privilegierten Ports gehört, muss noch Node-RED Zugriff zu diesem Port gewährleistet werden. Folgende Command muss in die Bash ausgeführt werden:

1. sudo setcap 'cap\_net\_bind\_service=+ep' $(eval readlink -f `which node`)

Jetzt kann das Dashboard auf die Adresse «http://smartlamp» erreicht werden. Die Noed-RED UI kann auf die Adresse «htt://smartlamp/admin» erreicht werden

### WPS Button

Die Smartlamp muss bei der Inbetriebnahme mit dem Heimnetz verbunden werden. Verschiedene Ansätze wurden in Betracht genommen, um dieses Problem zu lösen. Schlussendlich wurde es entschieden einen sogenannten «Wi-Fi Protected Setup» zu implementieren. Diese Feature ermöglicht Geräte die kein Benutzerschnittstelle (Tastatur und Bildschirm) haben, um die Wifi Password einzugeben, sich trotzdem mit einem Netzwerk zu verbinden. So Funktioniert’s: auf das neue Gerät und auf dem Wifi Router muss innerhalb 2 Minuten der WPS Knopf gedrückt werden. Dann die zwei Geräte tauschen sich die nötigen Informationen aus und die Verbindung wird automatisch hergestellt.

In Linux folgende Befehle müssen ausgeführt werden und die WPS Funktion zu aktivieren:

1. wpa\_cli -i wlan0 scan
2. wpa\_cli -i wlan0 wps\_pbc

Diese zwei Befehle werden in ein Exec Node ausgeführt. Der Exec Node ist entweder von ein Dashboard Button oder von eine GPIO (Hardware Button) getriggert. Beim Drucken der WPS Button (SW oder HW) wird die aktuelle Wifi Verbindung (wenn eine Vorhanden) getrennt und falls möglich eine neue hergestellt.

