LERNERSATZLEISTUNG

im Fach Informatik an der Klaus-Groth-Schule Neumünster

vorgelegt von
Tobias Schramm, Florian Siewers, Jasper Masekowsky,
Tom- Calvin Haak & Finn Werft

Klasse: 12 Physik A
Lehrer: Herr Krause
Datum: 20.01.2015

INHALTSVERZEICHNIS

QUELLENVERZEICHNIS

Abbildung 1:
www.de.wikipedia.org/wiki/Open_Source#mediaviewer/File:Opensource.svg
abgerufen am 19.12.2014

Abbildung 2:
www.roboter-bausatz.de/media/image/thumbnail/FZ0550-2_720x600.jpg
abgerufen am 10.01.2015

Abbildung 3:
www.de.wikipedia.org/wiki/Datei:Raspberry_Pi_Logo.svg
abgerufen am 10.01.2015

ABBILDUNGSVERZEICHNIS

I. Motivation & Idee

Wer kennt das nicht?

Müde und geschafft kommt man nach getaner Arbeit nach Hause, schaltet das Licht im Flur an, anschließend im Wohnzimmer – vier neue, stimmungsvoll dimmbare Ikea Lampen (Nachdem man dreimal ums Sofa gekrochen ist, sind sie dann auch alle gleich hell.) jetzt noch den Fernseher und die Stereoanlage an machen. Geschafft!

Und jetzt? Gleich aufs Sofa? Nein! Lieber schnell noch was zu essen holen. Also ab in die Küche, Licht an, Radio an, Schnittchen schmieren und eine Tüte Chips mitnehmen.

[...]

Nach dem zweiten durchlauf der Tagesschau, einer Folge TV Total und einer Wiederholung der Heute Show ist es dann so weit: Am liebsten möchte man direkt im Bett liegen!

Daraus wird leider nichts. Denn erstmal muss man gefühlt eine halbe Stunde durchs Haus laufen, um jede Lampe und jedes Gerät einzeln wieder auszuschalten.

Sobald der Rundgang beendet ist und man dann doch endlich im Bett liegt, hört man ein Geräusch... Die Nachbarn? Oder habe ich mich getäuscht? Nein, da ist es wieder! Laut und deutlich!

„Ding Dang Dong! Hier ist das erste deutsche Fernsehen mit der Tagesschau!“

Verdammt! Fernseher vergessen... Also wieder aufstehen und ins Wohnzimmer laufen! Oder war’s der Fernseher im Bad?

Egal – nach dem dritten Durchlauf der Tagesschau liegt man dann doch endlich wieder im Bett und kann mit etwas Glück sogar durchschlafen.

Möglicherweise schlagen Sie sich seit längerem mit ähnlichen Problemen rum und dachten diese mit einem Arsenal an Funksteckdosen beheben zu können.

Nun, entweder haben Sie ihr Auto verkauft, sämtliche Sparverträge aufgelöst, eine Hypothek aufs Haus aufgenommen und lassen Ihre Kinder arbeiten, um ein „Premium Home Controlling Solution System“ (oder so ähnlich) mit elektrischen Velux Fenstern finanzieren zu können oder aber Sie schlagen sich mit den folgenden Problemen herum.

Immer dann, wenn Ihr Chef Ihnen einen Einlauf verpasst und Ihr Gehalt gekürzt hat, Ihr Sohn eine 5 in Mathe mit nach Hause bringt und Ihre Frau das Auto gegen die Wand gesetzt hat, Sie also bei bester Laune sind, ist die Batterie Ihrer Fernbedienung leer.

Ein anderer Fall... Sie haben neue Lampen und Steckdosen für Ihr Arbeitszimmer gekauft und möchten diese schnell aufbauen und ausprobieren. Schade! Alle verfügbaren Frequenzen der

Fernbedienung sind belegt. Entweder das Arbeitszimmer wird beleuchtet, wenn Sie im Wohnzimmer TV und Stereoanlage einschalten oder aber wenn sie das Licht im Schlafzimmer an machen – oder doch lieber zusammen mit dem TV im Bad?
Klingt alles nicht nach einer zufriedenstellenden Lösung! Und die Fernbedienung liegt sowieso immer am falschen Ende Ihres Hauses. Jetzt könnten Sie natürlich für jeden Raum eine Fernbedienung anschaffen, dann haben Sie allerdings bald genug Systeme, um sie über fünf Generationen an sämtliche Nachfahren zu vererben... Diese werden sich sicher freuen!
Eines kann Sie aber noch schlimmer treffen! Der Super GAU!
Ihr Nachbar ist neuerdings auch stolzer Besitzer einer Sammlung von Funksteckdosen... Er funkt auf derselben Frequenz.
Der folgende Wettbewerb um die Beleuchtungshoheit ist für Außenstehende mit Sicherheit lustig anzusehen, für Sie jedoch einfach nur nervtötend.

Wir von OpenHC[1] dachten: „Es muss eine Lösung geben!“ und schon war die Idee unseres Projektes geboren. Wir wollten ein in jeder Hinsicht besseres System zur Kontrolle der Elektrogeräte im Haushalt entwerfen und konstruieren.

[1] Open(Open Source)HC(House Control)

II. Anforderungen

Zu Beginn der Projektarbeit müssen wir uns natürlich zunächst über die Anforderungen an unser Vorhaben Gedanken machen und dementsprechende Zielsetzungen entwerfen.
Diese sind zum einen durch die Vorgaben aus dem Unterricht gegeben (Arbeitsauftrag: Programmieren Sie eine Android Applikation), zum anderen aber auch durch unsere Kritikpunkte an bisherigen Systemen zur Fernsteuerung von Haushaltsgeräten.

Da es nicht unsere Absicht ist aus dem Projekt Profit zu schlagen, ist das erste Kriterium schnell gefunden. Um das System so kostengünstig wie möglich zu gestalten, aber auch um jedem die Möglichkeit zu lassen, es nach seinen eigenen Wünschen zu erweitern und zu optimieren, steht unsere Entscheidung fest, den gesamten Code als Open Source Software offenzulegen.

Abbildung 1: Open Source

Ein weiterer Punkt ist die erforderliche Hardware. Diese besteht zum einen aus dem Android Gerät des jeweiligen Anwenders, zum anderen aber auch aus zwei Geräten, die wir im Rahmen des Projektes entwickeln müssen. Eines der beiden Geräte ist im Prinzip eine handelsübliche Funksteckdose, die zwischen das Stromnetz und Endgerät geschaltet wird und sobald sie ein entsprechendes Funksignal empfängt, die Stromzufuhr ein- oder ausschaltet. Das andere Gerät stellt hingegen die Schnittstelle zwischen dem Android Gerät und der „Steckdose“ dar. Es ist entweder über WLAN oder aber per LAN Kabel in das Heimnetzwerk des Anwenders integriert und empfängt Signale von der App des Android Gerätes, welches sich entweder im selben WLAN befindet oder aber über einen VPN[2] Zugang von außerhalb zugreift. Diese Signale leitet es dann per Funk an die App weiter. Hierbei legen wir großen Wert auf die Verwendung entsprechender Funkstandards, um einen sicheren und zuverlässigen Betrieb zu gewährleisten.
Natürlich entschließen wir uns auch dazu, die Schaltpläne der Hardware offenzulegen, um beispielsweise die Option zu erhalten, die „Funksteckdose“ in anderer Bauart an Stelle eines Lichtschalters für die Kontrolle der Deckenbeleuchtung in die Wand zu integrieren.

[2] Virtual Private Network – Schnittstelle eines privaten, geschlossenen Netzwerks zu einem, sich nicht in diesem befindenden, Gerät

Bezüglich des Designs[3] sind wir uns schnell einig, dass es modern, schlicht und elegant sein soll, um die Geräte relativ neutral in den Wohnraum einpassen zu können und eine einfache, übersichtliche Handhabung auch für unerfahrene Anwender zu gewährleisten.

Das System soll sich nach Möglichkeit jedoch nicht nur einfach sondern sogar intuitiv bedienen lassen, was in Hinblick auf den von uns angestrebten Funktionsumfang eine penible Planung der UI[4] der Applikation erfordert. So soll es möglich sein neue Geräte hinzuzufügen und individuell zu benennen, sie in Gruppen einzuteilen und nach einem selbst erstellten Plan zeitzusteuern.

Um das Projekt abzurunden, sollen zusätzlich eine Website, die vorliegende Dokumentation und ggf. Marketingprojekte wie ein Werbespot angefertigt werden.

[3] Einschließlich der Oberfläche der Applikation
[4] User Interface – Schnittstelle zwischen Gerät und Benutzer

III. Organisation & Zeitplan

Die Organisation unserer Projektarbeit stützt sich auf die vorhergegangenen Unterrichtsinhalte[5]. Demnach fangen wir mit der Projektdefinition und -planung an, indem wir uns auf die genannten „Anforderungen" einigen und sie als Zielsetzung festhalten. Um das gesamte Projekt übersichtlicher zu strukturieren, teilen wir die Arbeit in Verantwortungsbereiche auf.

Tobias - Programmierung & Hardware

Florian & Jasper - Website

Tom - Design & Marketing

Finn - Organisation & Dokumentation

Trotz dieser Aufteilung arbeiten wir auch Resort-übergreifend zusammen, besprechen bei unseren wöchentlichen Treffen im Informatikunterricht bzw. in den Ferien die Fortschritte und Probleme und entwerfen beispielsweise das Logo und den Namen (OpenHC) in Teamarbeit.

Während der anschließenden Arbeitsphase erfolgt die Projektkontrolle aufgrund des langen Zeitraums nur an Hand von groben Deadlines[6], welche im Zeitplan festgehalten werden, im Rahmen des wöchentlichen Informationsaustausches, der auch der Dokumentation dient.

In der Projektabschlussphase in den Weihnachtsferien treffen sich Florian und Jasper, um die letzten Details an der Website zu besprechen und diese letztendlich fertigzustellen, während Tobias und Finn sich mit dem letzten Feinschliff an Code und Hardware befassen und die Informationen für die Dokumentation zusammenstellen.

---

[5] Thema: Projektarbeit/Softwareprojekte

[6] Für Zwischenziele

KW	Datum	Aufgaben
35-36	25.08.-05.09.14	Im Unterricht bei Frau Dr. Kröger werden die Grundlagen des Projektmanagements in Bezug auf Softwareprojekte erarbeitet.
37-38	08.09.-19.09.14	Wir einigen uns darauf eine Android Applikation als Projektarbeit zu erstellen. Ziel ist es, das Projekt an Hand des Gelernten erfolgreich zu strukturieren und bis zur Deadline nach den Weihnachtsferien 2014/15 abzuschließen. Des Weiteren teilen wir uns in Gruppen ein.
39-41	22.09.-10.10.14	Wir sammeln Ideen für unser Projekt und arbeiten diese in Form von Anforderungen und Zielsetzungen aus. Des Weiteren klären wir organisatorische Dinge wie die Arbeitsaufteilung in unterschiedliche Resorts und erste Zwischenziele. So sollen die Konzepte von Applikation und Website bis nach den Ferien soweit ausgearbeitet sein, dass mit dem Programmieren begonnen werden kann. Außerdem entwerfen wir das Logo und den Projektnamen. Erste Idee für die Website ist ein Onepager, aufgrund des teils Text-lastigen Inhalts entscheiden wir uns jedoch für ein klassisches Design.
42-43	13.10.-24.10.14	Herbstferien Die Anforderungen an die Applikation werden in ein Konzept umgewandelt, auf dessen Grundlage die Programmierungsphase beginnen kann. Nach letzten Änderungen wird eine Vektorgrafik des Logos erstellt.
44-46	27.10.-14.11.14	Um die Arbeit auch von zu Hause aus übersichtlicher zu strukturieren und unserem Open Source Konzept nachzukommen stellen wir die weitere Projektarbeit auf GitHub[7] um. Wie geplant beginnt die Programmierungsphase an Website und Applikation.

[7] www.github.com/OpenHC

47–49	17.11.–05.12.14	Das Arbeitspensum wird zu Gunsten der Probeabiturklausuren und weiterer Projekte reduziert
50–51	08.12.–19.12.14	Der Informatikkurs wird aufgrund des längeren Ausscheidens Frau Dr. Krögers von Herrn Krause übernommen. Florian und Jasper machen was auch immer an der Website, während Tobias und Finn sich mit der Planung des UI's der Applikation befassen. Nebenher unterstützt Tobias verstärkt andere Gruppen beim Programmieren.
52–2	22.12.14–06.01.15	Weihnachtsferien, In den Weihnachtsferien treffen sich Tobias und Finn um Hardware und Dokumentation fertigzustellen, während Jasper und Florian die Website vollenden.
2–3	07.01.–13.01.15	Deadline, Vor der Abgabe werden letzte Änderungen an den Kommentaren im Code vollzogen, um diesen besser verständlich zu machen. Außerdem werden die Ergebnisse aus den einzelnen Resorts präsentiert und mit den Zielsetzungen verglichen. Wir schließen die Projektarbeit zufrieden und erfolgreich ab.

IV. Website

Wie auch bei der App beginnt unsere Arbeit an der Website mit dem Sammeln von Ideen, der Äußerung von Anforderungen und der Formulierung von Zielsetzungen.
Wichtig ist uns vor allem, dass unsere Website ausschließlich auf unserem Code beruht und keine Java Script Bibliotheken wie jQuery oder Ähnliches verwendet, was wir weder selbst entworfen haben, noch vollständig verstehen. So begründet sich unsere Entscheidung die Seite nur mit Hilfe von HTML und CSS aufzubauen.
Da wir in einem Zeitalter leben, in dem nahezu jeder mit mindestens einem internetfähigen Mobilgerät ausgestattet ist, steht außerdem sehr schnell fest, dass wir ein responsives Layout entwerfen möchten.
Ebenfalls einig sind wir uns darüber, dass wir wie auch bei dem Entwurf der Hardware unseres Systems auf ein schlichtes aber elegantes und modernes Design setzen möchten, welches die Inhalte anschaulich und nach Möglichkeit barrierefrei präsentiert.
Unser erster Gedanke – einen One-Pager zu programmieren – ist wie bereits eingangs erwähnt schnell erledigt, weil wir zunächst die Dokumentation unserer Projektarbeit auf der Website zeigen möchten und einen One-Pager für sehr Text-lastigen Inhalt als ungeeignet erachten.
In Folge dessen entscheiden wir uns für ein eher klassisches Design, was sich durch den sorgfältigen Umgang mit CSS von anderen Websites abheben soll.

Das finale Ergebnis kann unter folgendem Link eingesehen werden:
http://openhc.github.io/

V. Funktionsweise

a. Hardware

Um die verwendete Hardware so zuverlässig und günstig wie möglich tu halten, wird sie in zwei Geräte aufgeteilt. Jedes dieser Endgeräte verfügt hierbei über ein Funkmodul und einen Microcontroller.

Als Gateway[8] zum LAN[9] des Nutzers dient ein kleiner Einplatinencomputer, auf welchem Debian[10] läuft. Mittels dieses Computers ist folglich jedes netzwerkfähige Gerät in der Lage, Komponenten im Haushalt steuern zu können.

Unsere Entscheidung zur Besetzung der Hardware ist die Folgende:

Einplatinencomputer: Raspberry PI

Microcontroller: ATMega 88

Funkmodul: NRF24L01+ (2,4 GHz)

Die NRF24L01+ Module sind nicht nur sehr günstig, sondern haben sich in der Vergangenheit auch als äußerst zuverlässig herausgestellt. Zudem ermöglichen sie eine Übertragung mit Datenraten von bis zu 2 Mbit, wodurch sie auch für anspruchsvolle Aufgaben wie das Übertragen von Musik geeignet sind und der Entstehung von Engpässen im Rahmen zukünftiger Applikationen entgegenwirken.

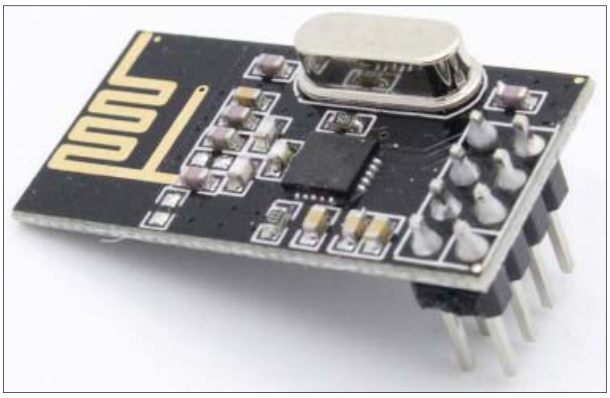

Abbildung 2: Funkmodul

Der ATMega 88 eignet sich als Microcontroller für unser Projekt, da alle vorgestellten Aktoren dank des Gateways nur sehr wenig Rechenleistung besitzen müssen. Des Weiteren gehört er zu einer Familie von untereinander kompatiblen Controllern, weshalb der nachträgliche Wechsel zu einem leistungsfähigeren Modell – zur Bewältigung anspruchsvollerer Aufgaben – problemlos möglich ist. Zusätzliche Features, wie Watchdog und Brown Out Detection garantieren dabei die Zuverlässigkeit der Komponente.

[8] Schnittstelle
[9] Local Area Network
[10] Linux Distribution

Der Raspberry PI fungiert als Herzstück des Gateways und überzeugt auf ganzer Linie. Er ist günstig, kann mit freier Software betrieben werden und verfügt über die nötigen Schnittstellen, um mit den anderen Komponenten kommunizieren zu können. Außerdem ist die Option, den enthaltenen ARM Prozessorkern auf nur 1 GHz zu takten, in Hinblick auf die Energieeffizenz - welche bekanntlich eng mit Stromrechnung und Umwelt verknüpft ist - ein weiterer Pluspunkt.

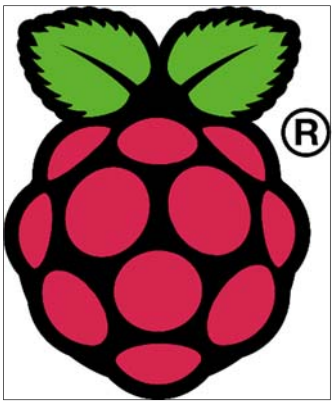

Abbildung 3: Raspberry Pi

Die Verschaltung des Raspberry Pi ist recht simpel, da dieser die für das NRF24L01+ Funkmodul benötigte Spannung direkt bereitstellen kann:

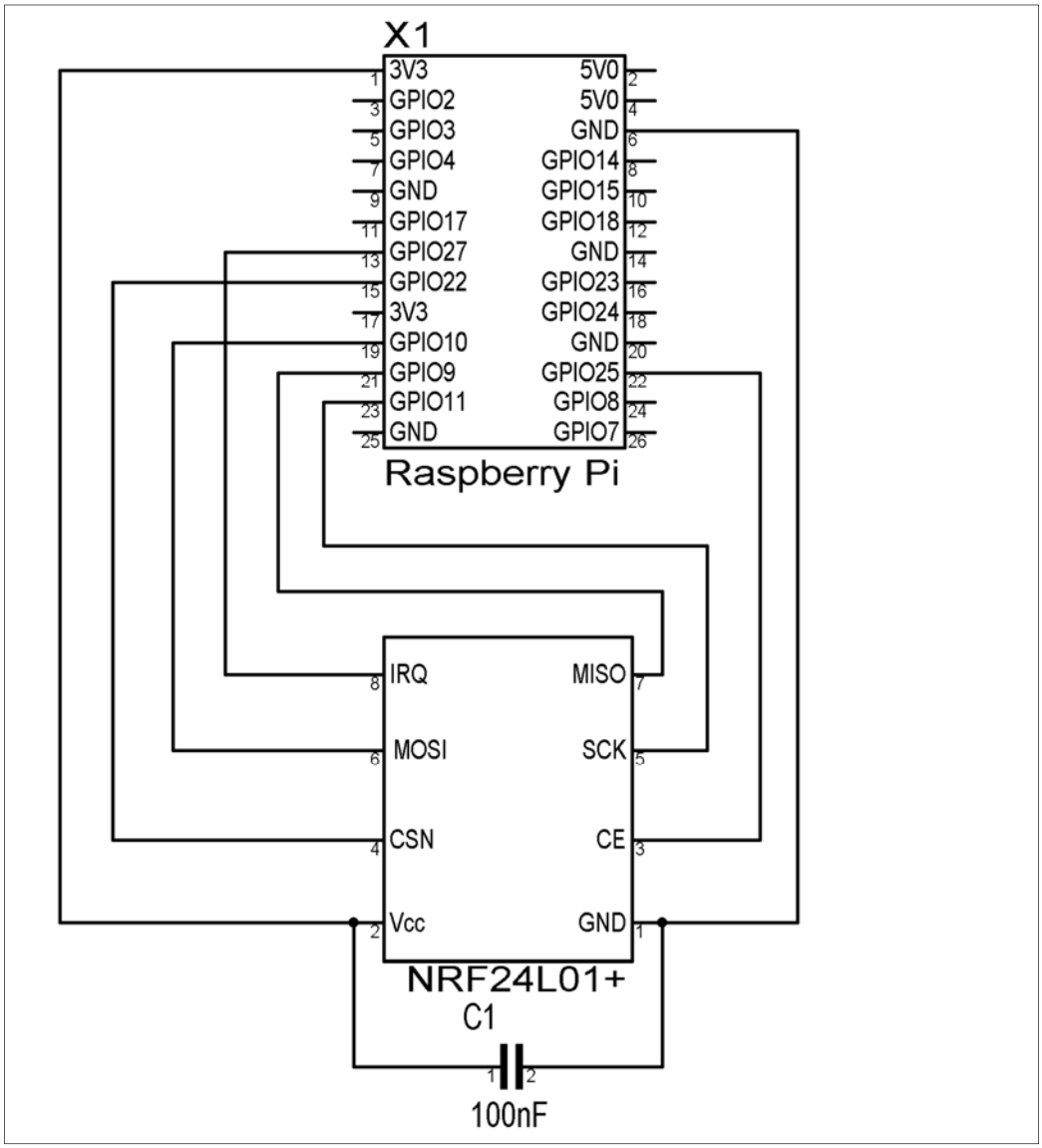

Abbildung 4: Schaltplan des Raspberry PI's mit Funkmodul

Der Aufbau eines Aktors ist etwas komplexer, da er in der Regel mit 230 Volt Netzspannung betrieben wird und somit ein Schaltnetzteil mit integriert werden muss.

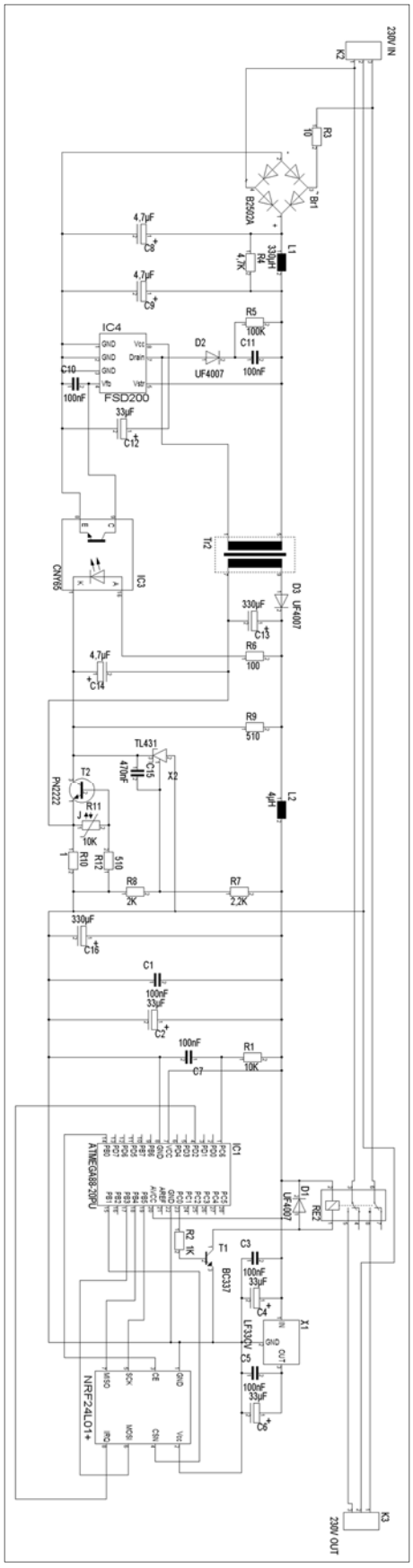

Abbildung 5: Schaltplan einer ferngesteuerten Steckdose

Allerdings lässt sich der Schaltplan für eine Demonstration etwas vereinfachen, indem Netzteil und Relais entfernt werden.

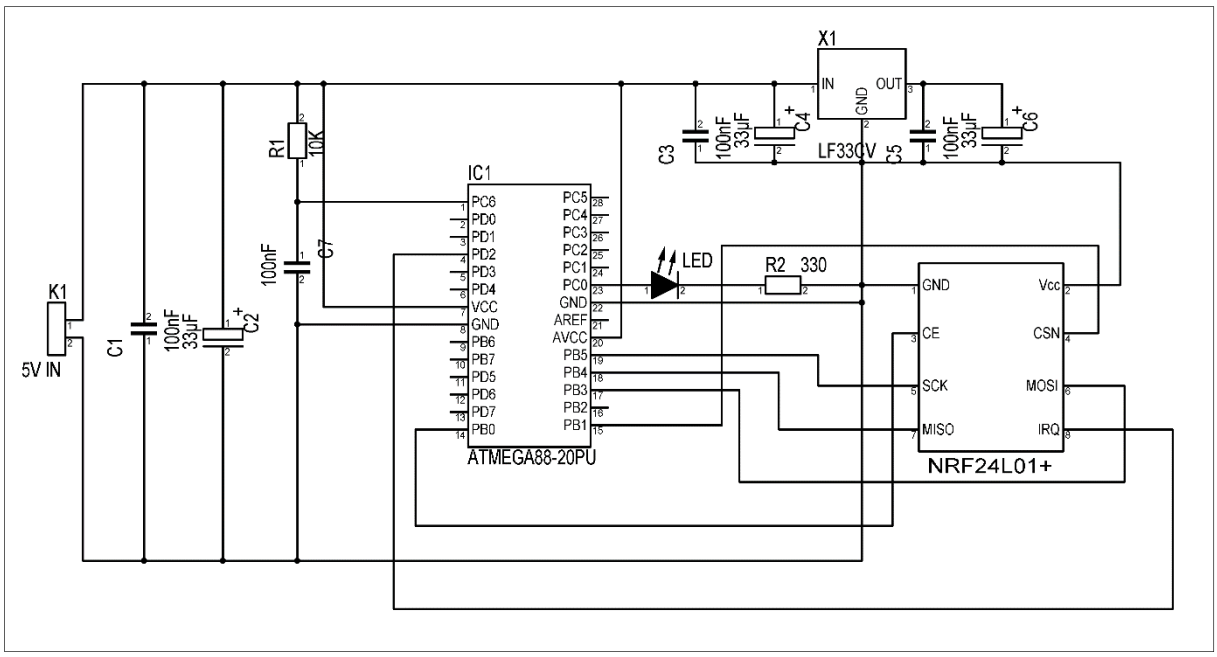

Abbildung 6: Schaltplan ohne Netzteil & Relais

b. Software

Nachdem die Frage der Hardware abschließend geklärt ist, stellt sich die Frage, welche Programmiersprachen für die Bewältigung welcher Aufgaben am besten geeignet sind.
Bei den Microcontrollern der ATMega Familie ist diese schnell beantwortet, da sich die Optionen im Wesentlichen auf Assembler und C beschränken.
Aufgrund unserer Intention das gesamte Projekt offen zu halten und die bestmögliche Verwendbarkeit für Entwickler sicherzustellen, fällt die Entscheidung auf C. Das hat den einfachen Grund, dass es kaum möglich ist, portierbare Programmbibliotheken in Assembler zu schreiben und somit die Hürde für die Entwicklung neuer Komponenten deutlich höher läge. Jeder, der Software für eine eigene Komponente entwickeln wollte, müsste dann nämlich seine eigene Programmbibliothek entwickeln.
Im Fall des Raspberry Pi's kommt im Gegensatz eine wesentlich größere Vielfalt an Programmiersprachen für das Projekt in Frage. Anders, als bei den Mikrocontrollern, welche für just-in-time (JiT) Kompilierung nicht die nötigen Ressourcen haben, kann hier auf eine JiT kompilierte Sprache gesetzt werden. Dies bietet den Vorteil, dass zwischen Architekturen gewechselt werden kann, ohne den Sourcecode verändern zu müssen, da sich dieser erst bei seiner Ausführung passend für die jeweilige Prozessorarchitektur kompiliert.
Aus der großen Menge an nutzbaren Programmiersprachen sticht allerdings Javascript heraus. Dank der Software NodeJS, welche Javascript Code JiT kompilieren kann, ist es möglich sehr stark asynchron zu arbeiten. Das heißt, dass viele Teile des Programms komplett unabhängig voneinander laufen können. Da es sich bei dem Gateway um ein Gerät handelt, das Daten, welche zu einer unbestimmten Zeit aus dem Netzwerk kommen, möglichst schnell per Funk an die richtigen Geräte weiterleiten muss, ist NodeJS für diese Aufgabe perfekt. Bei vielen andern Lösungen würden sich Pakete zunächst aufstauen und zwischengespeichert werden, um im Anschluss ein paar hundert Millisekunden später verarbeitet zu werden. Somit verlängerte sich die Reaktionszeit der Aktoren jedoch deutlich und das Gefühl, alle befohlenen Aktionen würden in Echtzeit umgesetzt werden, ginge verloren.
Für die zur Steuerung der Komponenten entwickelte Android App es wiederum keine große Auswahlmöglichkeit an Programmiersprachen. Hier kommt nur Java infrage.
Sobald die Wahl der Software getroffen ist, steht die Wahl eines Kommunikationsprotokolls bevor. An dieser Stelle muss sorgfältig zwischen den Vor- und Nachteile verschiedener Übertragungsverfahren abgewogen werden. Grundsätzlich steht die Entscheidung über ein Konzept für die Datenübertragung im Vordergrund. Aufgrund der Tatsache, dass bei unserem System jede Veränderung des Status' in Echtzeit angezeigt werden soll, war klar, dass für diese Aufgabe

nur Remote Procedure Calls (RPC) in Frage kommen. Bei Verwendung eines RPC's sprechen über das Netzwerk übertragene Daten in einem Programm direkt eine Funktion an. Die logische Konsequenz der beschriebenen Prozedur sind folglich besonders kurze Latenzzeiten bei der Steuerung von Aktoren.
Als Format für die Übertragung der RPCs steht eine Vielzahl von Formaten zur Verfügung. Unsere Entscheidung fällt auf JSON, da alle von uns - auf im Netzwerk befindlichen Geräten verwendeten - Programmiersprachen einen integrierten Decoder für JSON enthalten, der die Umwandlung in Objekte der jeweiligen Sprache übernimmt. Damit ist sichergestellt, dass der verwendete Decoder immer optimal mit der Runtime der Programmiersprache zusammenarbeitet.
Da jetzt feststeht, was übertragen werden soll, stellt sich die wichtige Frage ob UDP[11] oder TCP[12] verwendet werden soll. Der Unterschied dieser beiden Verfahren ist, dass bei UDP immer einzelne Datenpakete mit einer festen Maximallänge übertragen werden, während bei TCP ein kontinuierlicher Strom von Daten[13] übertragen werden kann.
Abhängig davon, ob von zu Hause oder aber von unterwegs geschaltet werden soll, haben jedoch beide Verfahren Vor- und Nachteile. Um nur die Vorteile zu erhalten und die Nachteile vollständig zu eliminieren, setzen wir daher beide Verfahren ein. Wenn von zu Hause lokal gesteuert werden soll und dafür TCP-Streams verwendet würden, wäre dies sehr ineffizient. Für das Öffnen einer TCP-Verbindung müssen nämlich zuerst mehrere Pakete über das Netzwerk übertragen werden. Im laufenden Betrieb wird die minimale Latenz für die Übertragung kleiner Datenmengen dann durch den Nagle-Algorithmus beschränkt. Wieder ergäbe sich eine spürbare Verzögerung zwischen dem Senden einer Anweisung und der Ausführung dieser durch den entsprechenden Aktor. UDP bietet hingegen die Möglichkeit viele kleine Pakete direkt verschicken zu können, da anders als bei TCP keine Kontrollstruktur vorhanden ist, die sicherstellt, dass Pakete(in der richtigen Reihenfolge) ankommen.
Intern im eigenen Heimnetzwerk stellt das Nichtankommen von Paketen aber ohnehin nur selten ein Problem dar. Daher kann eine eigene Struktur zur Bestätigung von Datenpaketen ohne spürbaren Geschwindigkeitsverlust implementiert werden.
Wenn man sich dagegen außerhalb des lokalen Netzwerks bewegt, tritt gerade bei der Verwendung von Hotspots oder des Mobilfunknetzes die Problematik hoher Latenzzeiten und langer Routen ein. Die Verwendung vieler kleiner UDP Pakete würde das Arbeiten mit der App folglich nahezu unmöglich machen. Demnach ist an dieser Stelle die Verwendung von

[11] User Datagram Protocol
[12] Transmission Control Protocol
[13] Stream

TCP Streams anzustreben. Die interne Erkennung für verlorene Pakete macht TCP deutlich robuster gegen schlechte Internetverbindungen und eröffnet die Möglichkeit größere Mengen an Daten „in einem Rutsch“ zu versenden. Da der dominierende Faktor die Latenz der Verbindung ist, lässt sich die Arbeit durch das Übertragen von mehreren Änderungen in einem Stream deutlich beschleunigen. Eine einzelne Übertragung kann ohne weiteres 20 RPCs zusammenfassen, wodurch bei einer Latenz von 200ms in Mobilfunknetzwerken statt 20 * 200ms = 4s nur einmal 200ms Verzögerung auftreten.

Jetzt braucht es nur noch eine organisierte Möglichkeit auf jedem Endgerät von OpenHC Informationen bereitzustellen und Befehle entgegenzunehmen. Dafür sind sogenannte 'Fields' verantwortlich. Ein Field ist immer mit einem Datentyp verknüpft. In einem Field kann entweder ein Integer, eine Fließkommazahl, ein boolescher Wert oder ein String abgelegt werden. Dabei geben die Attribute 'min_value' und 'max_value' den minimalen und maximalen Wert des Feldes oder aber die minimale und maximale Länge des Strings an. Minimale String-Längen können genutzt werden, um feste Präfixe vor - durch den Nutzer veränderbare - Strings zu schreiben. Außerdem können Fields gegen ungewolltes auslesen geschützt werden, was sinnvoll sein kann, um intern Keys für Verschlüsselungen abzulegen und diese nur neu beschreib-, aber nicht auslesbar zu machen, um bisher aufgezeichnete Daten nicht entschlüsselbar zu machen. Das Gegenstück dazu ist das nicht beschreibbare Field. Dieses kann verwendet werden, um z.B. Messwerte von Sensoren zur Verfügung zu stellen, diese aber nicht manipulierbar zu machen.

Die RPCs enthalten immer ein Feld mit dem Namen 'method'. Dieses Feld spezifiziert die Methode/Funktion, welche auf der Seite des Empfängers aufgerufen werden soll. Zur Erkennung von verlorenen Paketen gibt es außerdem noch das Feld 'transaction_uuid'. Dieses Feld enthält einen einzigartigen String, anhand dessen die Antwort eindeutig einem RPC zugeordnet werden kann. Um nur autorisierte Personen das Gateway bedienen zu lassen, ist in allen Paketen auch noch ein session token im feld 'session_token' enthalten, welcher beim Login durch das Gateway vergeben wird. Alle weiteren Felder sind vom Kontext der Übertragung abhängig. So enthält jede Antwort vom Gateway ein Feld 'success', welches angibt, ob der Aufruf der Methode erfolgreich war. Ein weiteres Feld namens 'login' zeigt dabei an, ob der Client autorisiert war, den RPC auszuführen.
Zur Kommunikation mit den Endgeräten verwendet das Gateway aktuell ein sehr simples Protokoll, welches immer aus 32 Byte langen Paketen besteht:

0 – 4, 5 Byte	5, 1 Byte	6 – 7, 2 Byte	8 – 31, 24 Byte
Adresse	Flags	Field Index	Daten

Die Flags sind wie folgt aufgebaut:

5.0 – 5.4, 5 bit	5.5, 1 bit	5.6, 1 bit	5.7, 1 bit
Länge der Daten	Lesen/Schreiben	Fortsetzung	Letztes Paket

Die endgültige Version des Protokolls wird allerdings, wie nachfolgend beschrieben, aus Sicherheitsgründen anders aussehen.

Zwar ist zurzeit noch keine Verschlüsselung der Datenübertragung implementiert, aber das Protokoll sieht sie in der fertigen Version als zwingend vor. Alle Anfragen an das Gateway über TCP werden mit TLS 1.2 ECDHE AES 256 verschlüsselt sein und somit über perfect forward secrecy verfügen. Das heißt, dass auch bei einer nachträglichen Entwendung des Zertifikats alle bisher zum Gateway aufgebauten Verbindungen nicht entschlüsselt werden können. Außerdem werden sowohl die App, wie auch ein in Zukunft verfügbares Webinterface certificate pinning betreiben, wodurch ein Austausch des Zertifikats durch ein anderes erkannt und die Verbindung mit einem Angreifer verweigert wird.
Für UDP Verbindungen wird jedes Gateway mit einem 256 Bit AES Key ausgeliefert. Dieser wird direkt nach der ersten Verbindungsaufnahme per App oder per Webinterface invalidiert und ein neuer Key angezeigt. Gleichzeitig wird für das Gerät ein Zertifikat ausgestellt, mit welchem es jede weitere Anfrage signieren muss, damit diese akzeptiert wird. Die Apps werden einen RSA Key mit einer Länge von 4096 Bit generieren, von welchem der public Key über die mit dem AES 256 Key verschlüsselte Verbindung an das Gateway übermittelt wird. Über diesen Schlüssel erfolgt dann bei jedem Login ein Schlüsseltausch, wodurch ein AES 256 Key generiert wird, mit welchem alle Daten für diesen Login verschlüsselt werden. Dieses Verfahren ähnelt MOSH[14], wird aber zur Übertragung von RPCs verwendet werden.
Die Verbindung vom Gateway zu den Endgeräten (Sensoren, Aktoren usw.) erfolgt wie bereits erwähnt über NRF24L01+. Da die Endgeräte nicht über ein integriertes Kryptografiemodul verfügen, wird die Verschlüsselung komplett vom Microcontroller übernommen werden. Auch hier wird AES 256 zum Einsatz kommen. Jedes Endgerät und auch das Gateway selbst werden über eine RJ11 Buchse verfügen, über welche sie verbunden werden können. Sobald eine Verbindung hergestellt wurde sendet das Endgerät über die serielle Verbindung (USART) seine Adresse zusammen mit einer Definition seiner Fields an das Gateway. Dieses generiert dann einen AES 256 Key und sendet diesen zusammen mit seiner eigenen Adresse zurück an das Endgerät. Damit ist das Peering abgeschlossen und die beiden Geräte können miteinander verwendet werden.

[14] Version von SSH, die UDP statt TCP verwendet

Dafür wird allerdings das Protokoll, welches über die NRF24L01+ gesprochen wird verändert werden müssen. Die Verbindungen werden mehrere Sicherheitsfeatures erhalten:

I. Eine innerhalb einer Übertragung fortlaufende Nummer, welche die Pakete sequenziert. Das erlaubt gleichzeitig auch das erneute Anfordern verlorengegangener Pakete

II. Einen zufälligen challenge code: Dieser wird für jedes Paket, das verschickt wird generiert und mit dem Paket versendet. Dabei muss das nächste Paket von der Gegenstelle dann diesen Code enthalten, damit es akzeptiert wird. Der Code sollte dabei eine Länge von mindestens 3 Byte (→ 16777216 Möglichkeiten) haben, damit es zusammen mit der Sequenznummer nahezu unmöglich wird, durch Replay-Attacken zufällig die richtige Sequenznummer und den richtigen Challenge Code zu erwischen.

III. Um brute forcing zu unterbinden wird die maximale Menge von Paketen, bei denen entweder die Sequenznummer oder die Challenge falsch sein dürfen auf 20 Stück pro Minute begrenzt. Wenn ein Gerät aber danach mit der korrekten Challenge um die Wiederholung eines Paketes bittet, wird dieser Counter zurückgesetzt. Damit sind Brute Force Attacken in unter 1000 Jahren nicht machbar.

Diese Maßnahmen werden die Verschlüsselung auch für versierte Angreifer heutzutage unknackbar machen.

VI. Bedienungsanleitung

a. Login

Sobald Sie die App öffnen, werden Sie dazu aufgefordert sich anzumelden. Geben Sie dazu einfach Ihren Benutzernamen und Ihr Kennwort ein und drücken Sie anschließend auf „Verbinden“.

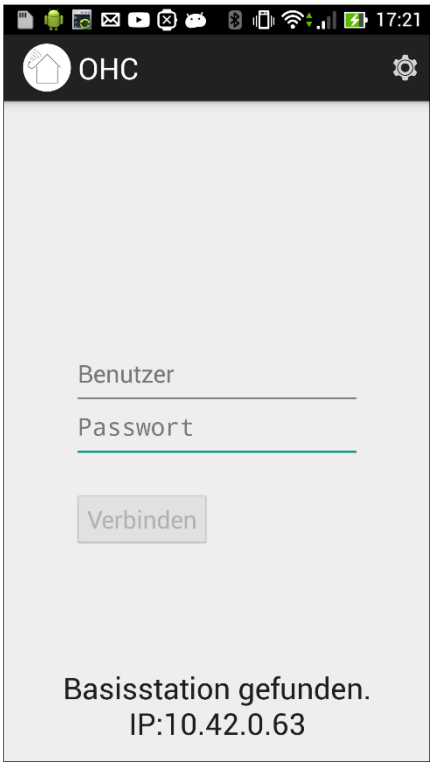

Abbildung 7: Login

Unterhalb des Logins erscheint außerdem eine kurze Statusmeldung, die Sie darüber in Kenntnis setzt, ob die Basisstation gefunden werden konnte und welche IP Adresse diese verwendet.

b. Gerät hinzufügen

Um ein neues Gerät hinzuzufügen verbinden Sie dieses zunächst an den dafür vorgesehenen Steckplätzen mit der Basisstation. Dadurch werden automatisch alle relevanten Informationen übertragen und Ihr Gerät ist anschließend einsatzbereit. Wenn Sie sich das nächste Mal in der App einloggen, erscheint es automatisch in der Übersichtsliste.

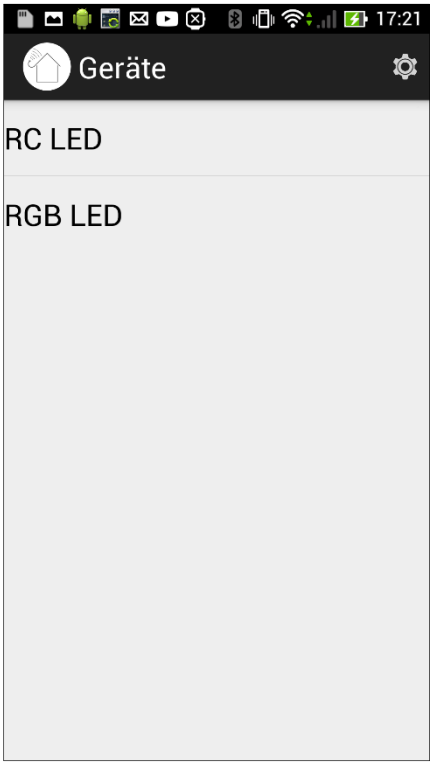

Abbildung 8: Übersichtsliste

c. Gerät bedienen

Wenn Sie ein dimmbares Gerät angeschlossen haben, können Sie die Helligkeit der einzelnen Leuchtmittel über die Schieberegler verändern.

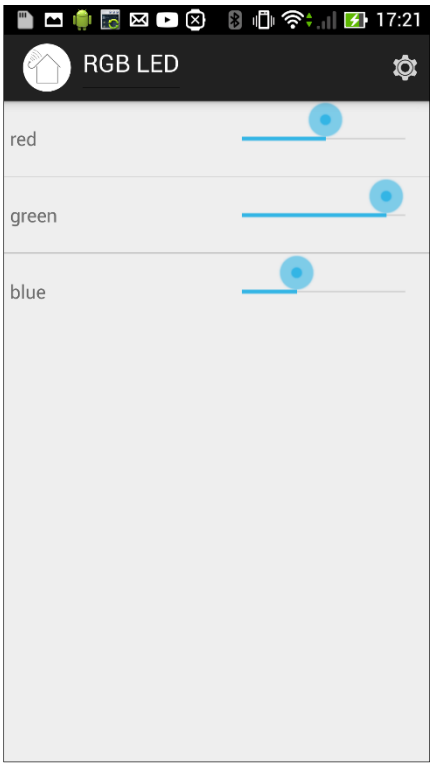

Abbildung 9: Dimmbares Gerät

So sieht der zugehörige Prototyp aus. Die Helligkeit der LEDs ist auf dem Bild nur schwer zu erkennen.

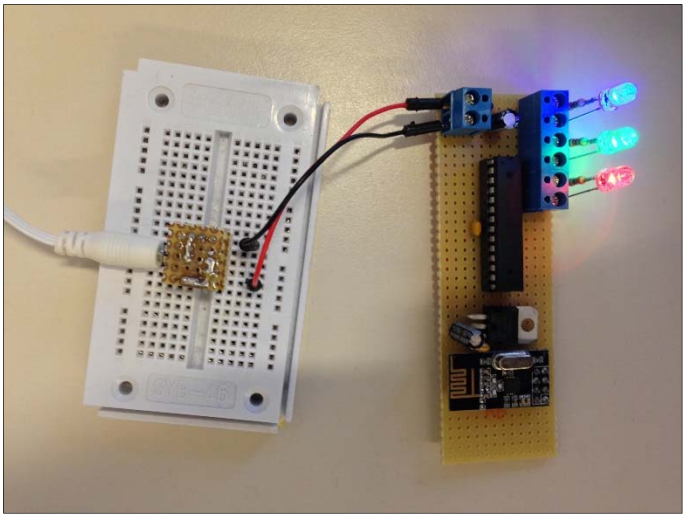

Abbildung 10: Dimmbarer Prototyp

Schließen Sie ein Gerät an, das nur an- oder ausgeschaltet werden kann, zeigt Ihnen die App automatisch nur einen Schalter an.

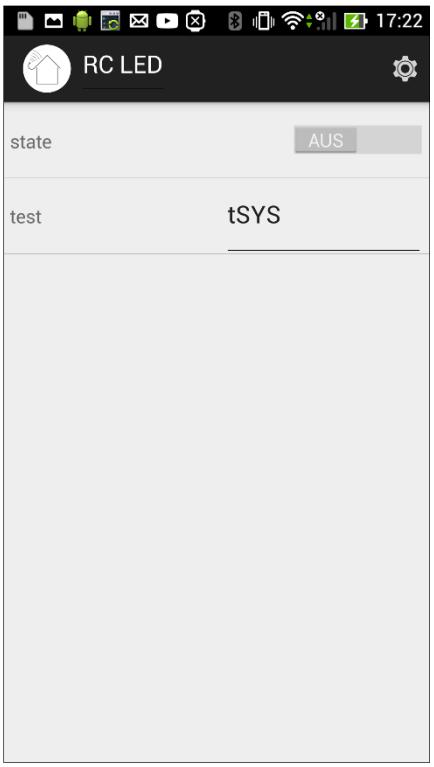

Abbildung 11: Nicht dimmbares Gerät

In dieser Ansicht können Sie außerdem den Gerätenamen ändern, indem Sie einfach in der Kopfzeile auf diesen tippen.

d. IP Adresse konfigurieren

Sollten Sie der Basisstation in Ihrem Heimnetzwerk eine feste IP Adresse zuweisen, können Sie diese manuell in der App angeben. Tippen Sie dazu auf das Zahnrad in der oberen, rechten Ecke.

Abbildung 12: IP Adresse manuell setzen

Abbildung 13: IP Adresse eingeben

VII. Verbesserungen & Ideen für die Zukunft

Selbstverständlich zählt zum vollständigen Abschluss eines (Software-)Projektes auch die intensive Auseinandersetzung mit Ablauf und Ergebnis des selbigen, um im Anschluss ein Resümee ziehen zu können und Ideen für neue Projektarbeiten aber auch für Verbesserungen am abgeschlossenen Projekt festzuhalten.

Insgesamt sind wir mit dem Verlauf des Projekts sehr zufrieden, da wir durch die häufige Absprache in der Gruppe immer Kompromisse gefunden haben, mit denen im Ergebnis alle zufrieden sind und durch die gute Organisation und Einteilung der Arbeit Stress in den letzten Wochen vor der Deadline vermieden werden konnte.

Dennoch stellte sich unser Vorhaben gerade in Bezug auf die Programmierung der Hardware als deutlich komplexer heraus als zu Beginn angenommen. Deshalb haben wir es leider nicht geschafft die ursprünglich geplante Zeitsteuerung in die App bzw. in das Gateway zu integrieren, sodass diese jetzt auf der Liste unserer Ideen für weitere Verbesserungen landet.

Dort stehen an erster Stelle Applikationen/Programme für andere Betriebssysteme. Wenn unser System tatsächlich zum einsatzkommen soll, wären Apps fürs iPhone/iPad und fürs Windowsphone wünschenswert. Ebenso wären Anwendungen für Macs, Linux und Windows Systeme denkbar, wobei ersatzweise natürlich auch auf das Webinterface des Gateways zugegriffen werden kann.

Abseits davon könnte es auch interessant sein einen Stromzähler zu integrieren und eine Möglichkeit zu entwickeln, die einzelnen Geräte in Gruppen zu gliedern und auch gruppenweise zu steuern.

EIGENSTÄNDIGKEITSERKLÄRUNG

Wir haben diese Arbeit selbstständig verfasst und keine anderen Hilfsmittel als die angegebenen Hilfsmittel verwendet.

Neumünster, den 19.01.2015

_____

Tobias Schramm

Neumünster, den 19.01.2015

_____

Florian Siewers

Neumünster, den 19.01.2015

_____

Jasper Masekowsky

Neumünster, den 19.01.2015

_____

Tom-Calvin Haak

Wattenbek, den 18.01.2015

_____

Finn Werft