Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Fabian Behnke

Hard- und Softwareentwicklung einer ereignis-gesteuerten Open-Source-Haussteuerung mit webbasiertem Zugriff

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Fabian Behnke

Hard- und Softwareentwicklung einer ereignisgesteuerten Open-Source-Haussteuerung mit webbasiertem Zugriff

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informations- und Elektrotechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Robert Fitz
Zweitgutachter : Prof. Dr. rer. nat. Henning Dierks

Abgegeben am 09. Juli 2014

Fabian Behnke

Thema der Bachelorthesis

Hard- und Softwareentwicklung einer ereignisgesteuerten Open-Source-Haussteuerung mit webbasiertem Zugriff

Stichworte

Hardwareentwicklung, Softwareentwicklung, Haussteuerung, Hausautomatisierung, Raspberry Pi, Einplatinencomputer, I2C, 1-Wire, Z-Wave, drahtlose Kommunikation

Kurzzusammenfassung

Diese Arbeit befasst sich mit der Analyse, dem konzeptionellen Entwurf und der Entwicklung einer Open-Source basierten Hard- und Softwarelösung zur Steuerung des Eigenheims. Dabei sollen Technologien, wie die drahtlose Kommunikation verwendet werden, die es ermöglichen, dieses System sowohl in Neu- als auch in Altbauten verwenden zu können.

Fabian Behnke

Title of the paper

Development and construction of hard- and software for an event-driven open-source home control system with web-based access

Keywords

Hardware development, Software development, Home control, Home automation, Raspberry Pi, Singe board computer, I2C, 1-Wire, Z-Wave, Wireless communication

Abstract

The focus of this thesis is on analysis, conceptual design and development of an open-source hard- and software solution to control your home. To use this system in new or existing buildings, technologies like wireless communications are used to make this possible.

Hinweise

Fachbegriffe wurden, wenn es sinnvoll erschien, ins Deutsche übertragen. Wurde der Begriff aus der englischsprachigen Fachliteratur übernommen oder die deutsche Übersetzung eines Fachbegriffs ist unüblich, wurden die Begriffe in der Originalsprache belassen.

Eigennamen und (englische) Fachbegriffe sind in der Arbeit *kursiv* gesetzt.

Der bezüglich dieser Arbeit produzierte Quellcode sowie die hier verwendeten Datenblätter sind bei Herrn Prof. Dr.-Ing. Robert Fitz an der Hochschule für Angewandte Wissenschaften in Hamburg in Form einer DVD hinterlegt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
BSD	Berkeley Software Distribution
CA	Certificate Authority
CPU	Central Processing Unit
ECMA	European Computer Manufacturers Association
FQDN	Fully qualified domain name
GPIO	General Purpose Input/Output
GPU	Graphics Processing Unit
HF	High Frequency
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IC	Integrated Circuit
JSON	JavaScript Object Notation
MIT	Massachusetts Institute of Technology
NDA	Non-Disclosure Agreement
NOSQL	Not only SQL
REST	Representational State Transfer
ROM	Read Only Memory
RTC	Real Time Clock
SPI	Serial Peripheral Interface
SQL	Structured Query Language
SSL	Secure Sockets Layer
STP	Shielded Twisted Pair
TCP	Transmission Control Protocol
TTL	Transistor-Transistor-Logik
UART	Universal Asynchronous Receiver Transmitter
URL	Uniform Resource Locator
USB	Universal Serial Bus
UTP	Unshielded Twisted Pair

1 Einführung

1.1 Motivation

In der heutigen Zeit werden Themen wie Energie sparen, Sicherheit schaffen, Heizung steuern, Lichtstimmungen herstellen und die Vernetzung des Hauses immer bedeutender.

Es sind zurzeit viele Smart-Home-Systeme auf dem Markt, die diese Leistungen erzielen. Doch was kosten solche Systeme? Inwieweit lassen sie sich noch nachträglich in eine bestehende Elektroinstallation integrieren? Und lassen sich diese Systeme den eigenen Wünschen entsprechend anpassen?

Die sichersten und besten Systeme, sind drahtgebundene Systeme wie EIB[1] bzw. KNX[2]. Doch sind solche Systeme sehr kostspielig (Kosten beim Einfamilienhaus zwischen 15.000 und 30.000 €, je nach Ausstattung) und lassen sich in eine bestehende Installation nur schlecht integrieren.

Die Hard- und Software der meisten Systeme ist proprietär und lässt sich deshalb nicht nach eigenen Wünschen anpassen.

1.2 Zielsetzung dieser Arbeit

Ziel dieser Arbeit soll es sein, eine kostengünstige Open-Source-Lösung zur Hausautomatisierung zu entwickeln, die von „jedermann“ angepasst oder weiterentwickelt werden kann. Um eine größtmögliche Anzahl von Personen anzusprechen, soll diese Lösung einfach zu bedienen und sowohl in Neuinstallationen als auch in bestehende Installationen integriert werden können.

---

[1] EIB – Der Europäischer Installationsbus ist ein nach EN 50090 Standardisiertes Bussystem zur intelligenten Vernetzung moderner Haus- und Gebäudesystemtechnik .

[2] KNX ist die Weiteeentwicklung des EIB-Standards durch eine Erweiterung um Konfigurationsmechanismen und Übertragungsmedien. KNX in der Norm ISO/IEC 14543-3 als Internationaler und in der Norm EN50090 al Europäischer Standard anerkannt. [1]

2 Technische Grundlagen

2.1 Hardware

2.1.1 Raspberry Pi

Der *Raspberry Pi* ist ein von der britischen *Raspberry Pi Foundation* entwickelter Einplatinencomputer in Kreditkartengröße. Ziel der Entwicklung war es, einen preisgünstigen Computer zu bauen, um jungen Menschen den Erwerb von Programmier- und Hardwarekenntnissen zu erleichtern.

Abbildung 1: Raspberry Pi Model B - Foto

Geschichte

Die ersten Konzepte wurden 2006 vorgestellt und basierten zu diesem Zeitpunkt auf einem *Atmel ATmega664 Mikrocontroller*. [2] Die Leistung dieses Gerätes überzeugte die Entwickler bis dato aber noch nicht. Durch den damals immer größer werdenden Smartphone-Markt fand sich nach kurzer Zeit der passende Prozessor. Der von der Firma *Broadcom* entwickelte *BCM2835*, ein Prozessor mit sogenanntem „System On A Chip“. Er war kostengünstig, hatte eine verhältnismäßig große Leistung und beherbergte zudem eine Grafikeinheit, Arbeitsspeicher und diverse Schnittstellen. Die Idee des neuen *Raspberry Pi* war geboren. Die ersten 50 *Alpha-Boards* wurden im August 2011 ausgeliefert. Sie waren funktional aufgebaut, wie das spätere Model B, aber größer um diverse Messpunkte zur Fehlersuche darzubieten. [3] Seit September 2012 wird

die Version 2 des *Raspberry Pis* verkauft. Sie besitzt nun zwei Befestigungslöcher und eine etwas veränderte Belegung des GPIO-Ports. Etwa einen Monat später erhielt diese Version 2 einen auf 512 MB verdoppelten Arbeitsspeicher. Durch die unerwartet hohen Verkaufszahlen konnte die Produktion sehr schnell aus China nach Wales verlegt werden. [4] Seither wurden mehr als 2.5 Millionen Geräte verkauft (über 1 Millionen produziert in Wales). (Stand: März 2014) [5] [6]

Hardwarebeschreibung

Wie oben bereits erwähnt, ist der *BCM2835* das Herzstück des *Raspberry Pi*. Er beinhaltet eine mit 700MHz getakteten ARM11 CPU, eine mit 250 MHz getaktete Dual-Core GPU mit integriertem 1080p H.264 Video En- bzw. Decoder. Des Weiteren sind 256 MB bzw. 512 MB SD-RAM, eine USB-, eine frei programmierbare GPIO und eine Kameraschnittstelle integriert. [7]

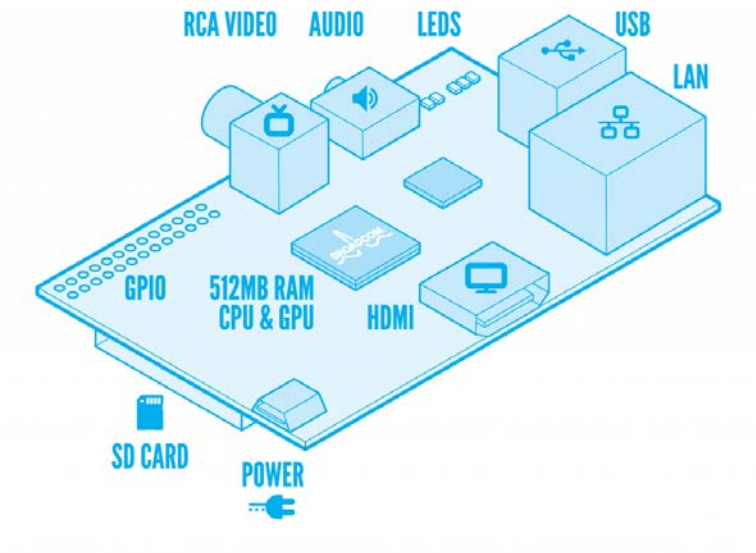


Abbildung 2: Raspberry Pi Model B – Schema [8]

Der *Raspberry Pi* verzichtet auf die direkte Möglichkeit eine Festplatte anzuschließen. Stattdessen wurde ein SD-Karten-Leser integriert, der es ermöglicht, das Betriebssystem von einer handelsüblichen SD-Karte zu booten. Ebenso verzichtet der Raspberry Pi auf eine Real Time Clock (RTC), da diese das Platinenlayout durch eine zusätzliche Batterie und weitere Komponenten um einiges vergrößert hätten und der von der *Raspberry Pi Foundation* angepeilte maximale Verkaufspreis überschritten worden wäre. [8]

Verfügbare Betriebssysteme

Seit der Vorstellung des *Raspberry Pi* sind diverse Betriebssysteme für den *Einplatinencomputer* entwickelt bzw. portiert worden. Bei den meisten Betriebssystemen handelt es sich um Linux-Distributionen, wie z.B. Raspbian (eine für den *Raspberry Pi* optimierte Version von *Debian*). [9] Es sind aber auch Versionen des Handybetriebssystems Android [10], sowie des für *embeddet Systems* entwickelte Betriebssystem *Microsoft Windows CE* verfügbar.

2.1.2 I^2C-Bus

I^2C steht für „Inter-Integrated Circuit“ und wurde Anfang der 1980er Jahre von *Philips Semiconductors* (heute *NXP Semiconductors*) entwickelt. Es handelt sich um einen seriellen Datenbus zur Kommunikation zwischen mindestens einem Mikrocontroller *(Master)* und mehreren ICs (Slaves) auf nur einem *Broadcast-Kanal*. Aus lizenzrechtlichen Gründen führte die Firma Atmel die Bezeichnung TWI (Two-Wire-Interface) ein, technisch gesehen sind TWI und I^2C aber identisch. [11] [12] [13 S. 133]

Versionshistorie

Jahr	Version	Wichtigste Fakten
1992	Spezifikation 1.0	Die erste standardisierte Spezifikation des I^2C-Busses ergänzte den ursprünglichen Standard-Mode (100 kbit/s) um den Fast Mode (400 kbit/s). Der Adressraum wurde um einen 10-Bit-Modus erweitert. Es es können nun bis zu 1136 I^2C-Geräte angesprochen werden.
1998	Spezifikation 2.0	Der Highspeed-Mode (3,4Mbit/s) wurde eingeführt. Die Strom- und Spannungsanforderungen wurden für diesen Modus gesenkt.
2000	Spezifikation 2.1	Kleinere Modifikationen am Highspeed-Mode
2007	Spezifikation 3.0	Der Fast-Mode Plus (1 Mbit/s) wurde eingeführt, basierend auf dem gleichen Protokoll, wie der Standard- und auch Fast-Mode.
2012	Spezifikation 4.0	Der Ultrafast-Mode (5 Mbit/s) wurde eingeführt; ein Modus mit Unidirektionaler Verbindung, für I^2C Geräte, die keine Rückmeldung senden.
2012	Spezifikation 5.0	Berichtigungen redaktioneller Natur an Spezifikation 4.0.
2014	Spezifikation 6.0	Korrekturen an 2 Graphen in Spezifikation 5.0

Tabelle 1: Versionshistorie - I^2C Spezifikationen [14] [15]

Technische Beschreibung

Bei I^2C handelt es sich um einen Zweidrahtbus mit Master-Slave-Konzept und je nach verwendetem Protokoll mit bidirektionaler oder unidirektionaler Übertragung. Der Controller (Master) initiiert die Übertragung und ist für Generierung des Taktsignals zuständig. Möchte der Master mit einem Slave kommunizieren, so fordert er durch eine entsprechende Adressierung den jeweiligen Busteilnehmer dazu auf. Slaves können niemals Verbindungen von sich aus initiieren. Mehrere Master (Multimaster-Mode) in einem System sind möglich. Dazu muss der Master Kontrollmechanismen wie die Abitrierung (Zugriffsregelung) und Kollisionsvermeidung unterstützen. Die Arbitrierung wird durch Synchronisation des Taktsignals und dem verwendeten Protokoll ermöglicht. Daten auf dem Bus werden mit positiver Logik übertragen (*Most Significant Bit* first). I^2C ist nicht als Feldbus geeignet und beschränkt sich durch die maximal erlaubte Buskapazität von 400pF auf nur 2 bis 3 Meter. [12] [13] [15]

Die gesamte Kommunikation wird über nur zwei Signalleitungen abgewickelt. Davon ist eine Takt- (SCL) und eine Datenleitung (SDA). Beide Leitungen sind über Pull-Up-Widerstände mit der Versorgungsspannung (V_{CC} - typischerweise 5 V) verbunden. Sämtliche am Bus angeschlossene I^2C-Geräte besitzen einen *Open-Collector-Ausgang*. Im Zusammenhang mit den Pull-Up-Widerständen ergibt dies eine sogenannte *„Wired-And-Schaltung“* (ermöglicht unter anderem die *Arbitrierung*). [16] [17] Der High-Pegel sollte mindestens 0,7 x V_{CC} betragen, der Low-Pegel höchstens 0,3 x V_{CC}. Der High-Pegel entspricht dabei einer logischen 1, der Low-Pegel einer logischen 0, das heißt der I^2C-Bus arbeitet mit einer positiven Logik. [13 S. 134-135] [15 S. 9]

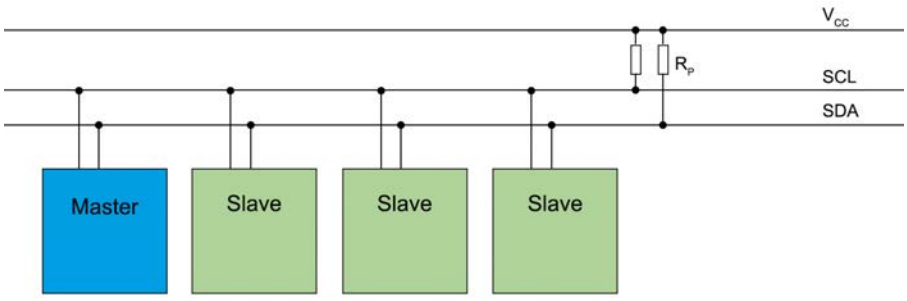


Abbildung 3: I^2C Bus mit einem Master und 3 Slaves [15 S. 8]

Wie oben erwähnt wird das Taktsignal auf dem I^2C-Bus von dem Master vorgegeben. Für die verschiedenen Modi wird die jeweils in der Spezifikation vorgegebene Taktrate verwendet. Benötigt einer der Slaves mehr Zeit, kann er zwi-

schen der Übertragung der einzelnen Bytes die SCL Leitung auf Low ziehen (Clock-Stretching) und den Master dadurch ausbremsen. Der Bus-Master reagiert und richtet sich nun fortan mit seinem Takt nach dem langsamsten Busteilnehmer. Datenbits auf der SDA-Datenleitung dürfen ihren logischen Pegel nur verändern, wenn sich die SCL-Leitung auf LOW Pegel befindet. Ist dies nicht der Fall, so ist das übertragene Bit ungültig. Ausgenommen von dieser Regelung sind das Start-, Stop- und das Repeated-Start-Signal. Sowohl das Start-Signal und als auch das Repeated-Start-Signal sind eine fallende Flanke auf der SDA Datenleitung, während SCL auf High-Level ist. Das Stop-Signal entspricht einer steigenden Flanke auf SDA während sich SCL auf High-Level befindet. [15]

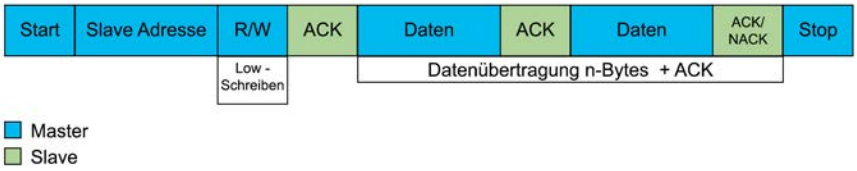


Abbildung 4: I²C Master sendet an Slave [15 S. 15]

Das Übertragungsprotokoll sieht vor, dass der Start einer Übertragung mit dem durch den Master gesendetem Start-Signal beginnt. Danach folgt die in der Regel sieben Bit lange Adresse des Slaves und ein Bit zur Lese- oder Schreibanforderung. Wurde das Adressbyte korrekt empfangen, bestätigt der jeweilige Slave dieses mit einem ACK-Bit (*Acknowledged*). Danach werden die Daten byteweise vom Slave gelesen bzw. zum Slave gesendet. Jedem übertragenen Byte folgt ein ACK-Bit, je nach Modus gesendet vom Slave oder Master. [13]

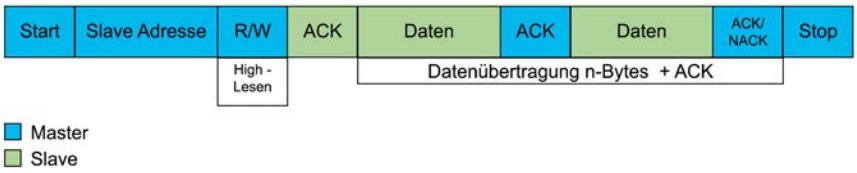


Abbildung 5: I²C Master fordert Daten von Slave an [15 S. 15]

Das letzte Byte im Lesezugriff wird vom Master durch ein NACK-Bit (*Not Acknowledged*) quittiert. Ist die Übertragung beendet, so sendet der Master das Stop-Signal und der Bus ist wieder freigegeben, z.B. für eine weitere Übertragung oder im Multimaster-Mode für die Übertragungsinitiierung eines weiteren Masters. Ist während der Übertragung etwas schief gelaufen, so kann der Mas-

ter ohne senden des Stop-Signals, das Repeated-Start-Signal senden und so die Übertragung wiederholen. [13] [15]

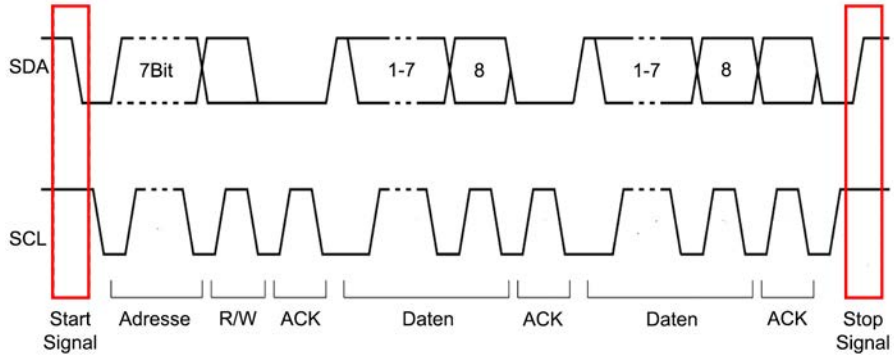


Abbildung 6: I²C Kompletter Übertragungsverlauf [15 S. 13]

Ansprechen eines Bausteins mit einer 10-Bit-Adresse:

Soll ein Baustein mit einer 10-Bit-Adresse angesprochen werden, so ist der Ablauf zu der regulären Adressierung etwas anders. Nach dem Startsignal sendet der Master die Bitfolge 11110, gefolgt von den ersten beiden Adressbits und dem R/W-Bit. Fühlen sich ein oder mehrere Bausteine angesprochen, so senden sie ein ACK-Bit. Darauf sendet der Master die restlichen 8 Bit der Adresse, ist ein Slave mit dieser Adresse vorhanden, so sendet dieser ein ACK-Bit. Der restliche Verlauf der Datenübertragung gleicht der Kommunikation mit einem 7-Bit adressierten Slave. [15]

2.1.3 1-Wire-Bus

Bei *1-Wire®* handelt es sich um eine von der Firma *Dallas Semiconductors Inc.* (heute *Maxim Integrated*) entwickelte serielle Schnittstelle. 1-Wire wurde als günstige Alternative zu I²C entwickelt und benötigt nur eine Datenleitung. Diese Datenleitung kann zusätzlich als Spannungsversorgung vieler 1-Wire-Geräte genutzt werden, sodass man im Idealfall mit nur einer Ader und einer Masseverbindung für eine Kommunikation mit einem oder mehreren 1-Wire-Geräten auskommt.

Es ist eine Vielzahl integrierter Bausteine mit 1-Wire-Bus-Anschluss verfügbar unter anderem zur Temperatur- und Luftfeuchtemessung, zur Akkuüberwachung oder als Echtzeituhr. Der 1-Wire-Bus lässt eine sehr lange Leitungslänge zu. So ist es z.B. möglich, ein Bussystem über mehrere hundert Meter aufzubauen (abhängig von der Dämpfung und der Qualität bzw. Schirmung der Leitung und der verwendeten Netzwerktopologie). [13] [18]

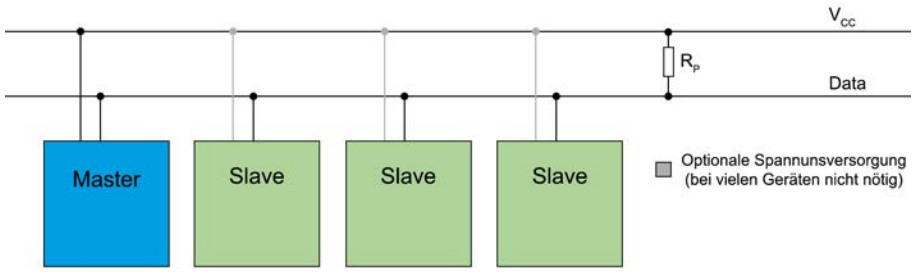


Abbildung 7: 1-Wire-Bus mit einem Master und 3 Slaves

Technische Beschreibung

Wie auch schon beim I^2C-Bus handelt es sich bei 1-Wire® um einen Master-Slave-Bus, jedoch mit dem Unterschied, dass es auf dem Bus nur einen Master geben darf (Single-Master-Bus). Die maximale Anzahl an Busteilnehmern ist auf 100 begrenzt. Das 1-Wire-Protokoll ist bidirektional ausgelegt und übertragene Daten werden mit negativer Logik gesendet (*Least Significant Bit, first*). Die Spezifikation des 1-Wire-Bus sieht zwei Geschwindigkeitsmodi vor: 15,4 kbit/s (Standard Mode) und 125 kbit/s (Overdrive Mode). Jedes 1-Wire-Gerät wird bei Fertigung mit einer einzigartigen 64-Bit-Adresse versehen. Diese Adresse setzt sich aus 8-Bit-Family-Code, 48 Bit Seriennummer und einer 8-Bit-CRC-Checksumme zusammen. [19]

Wie schon erwähnt, benötigt 1-Wire® nur eine Ader. Geräte wie z.B. der Temperatursensor DS18S20 besitzen einen integrierten Pufferkondensator, der geladen wird, solange sich die Datenleitung gerade im High-Zustand befindet. Dieser versorgt den Sensor mit Spannung. Dennoch ist es empfehlenswert (gerade bei langen Leitungslängen), eine separate Ader zur Spannungsversorgung vorzusehen und diese anzuschließen. [13 S. 155] [19]

Die Datenleitung wird über einen Pull-Up-Widerstand mit der Versorgungsspannung 3 V bzw. 5 V (je nach angeschlossener 1-Wire-Geräte) verbunden. Die Technischen Anforderungen an den 1-Wire-Master sind sehr gering, so ist es z.B. möglich, einen simplen Mikrocontroller, der einen bidirektionalen *Open-Drain*-Anschluss besitzt, als Master zu nutzen. Des Weiteren sind diverse Bausteine verfügbar, die andere Protokolle in das 1-Wire-Protokoll übersetzen, z.B. von I^2C zu 1-Wire®. Diese Bausteine übernehmen in diesem Fall die Master-Funktionalität. [19]

Da bei 1-Wire® keine separate Leitung für das Taktsignal vorgesehen ist, ist es wichtig, dass vorgegebene Timings eingehalten werden. Die Timing-

Synchronisation erfolgt bei jedem Bit mit der durch den Master erzeugten fallenden Flanke. Möchte der Master eine logische 1 senden, zieht er den Bus für 6 µs (1 µs im Overdrive-Mode) auf Low-Pegel, bei einer logischen 0 für 60 µs (7.5 µs im Overdrive-Mode). [19] [20]

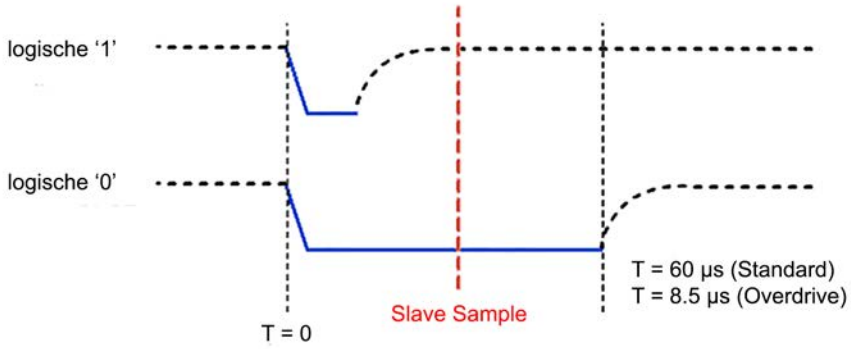


Abbildung 8: 1-Wire-Master schreiben

Möchte der Master ein Bit von einem Busteilnehmer lesen, so zieht er ebenfalls, wie auch beim schreiben der logischen 1 den Bus für 6 µs (1 µs im Overdrive-Mode) auf Low-Pegel. Der Slave hält daraufhin für die Übertragung einer logischen 0 den Bus für mindestens 9 µs (1 µs im Overdrive-Mode) auf Low, für die Übertragung einer logischen 1 auf High. Während dieser Zeit (nach insgesamt 15 µs im Standard bzw. 2 µs im Overdrive-Mode) tastet (engl. sampled) der Master den übertragenen Wert ab. [19] [20]

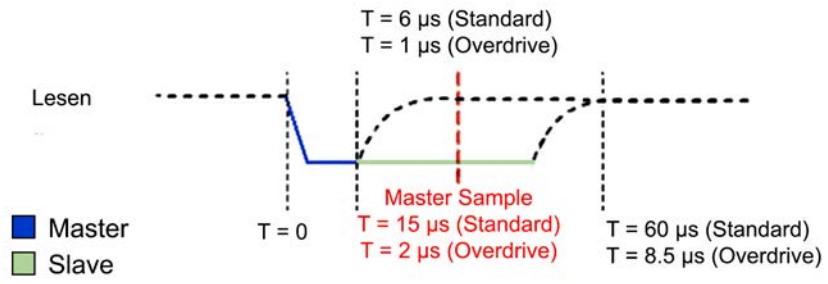


Abbildung 9: 1-Wire-Master lesen (Slave sendet '0')

Das 1-Wire-Protokoll ist in drei Phasen aufgeteilt. Es beginnt mit einem durch den Master initiiertem Rücksetzen des Busses (Master zieht den Bus-Pegel für im Standard-Mode für mindestens 480 µs auf Low, 70 µs im Overdrive-Mode). Alle 1-Wire-Geräte befinden sich nun in einem definierten Zustand und melden ihre Präsenz (Slaves ziehen den Bus für mindestens 60 µs auf Low, im Overdrive-Mode sind es 7.5 µs). Darauf folgen die ROM-Funktion, die Adresse des jeweiligen Gerätes und der Aufruf der gewünschten Funktion. Danach können die Daten je nach Funktion vom Master gelesen oder geschrieben werden. [20]

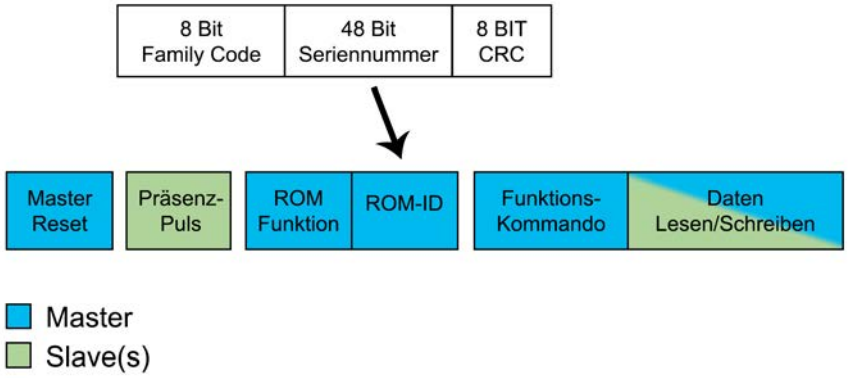


Abbildung 10: 1-Wire-Übertragungsschema

ROM-Funktionen

Nachdem der Slave den Präsenzpuls gesendet hat, sendet der Master die gewünschte ROM Funktion. Zu den ROM-Funktionen zählen:

- Skip-ROM – Auf dem Bus befindet sich nur ein Slave, der Master braucht keine ID zu senden, um das gewünschte Gerät zu adressieren. Er kann direkt nach dem Ausführen dieser Funktion mit dem Senden des Funktionskommandos beginnen.
- Read-ROM – Auf dem Bus befindet sich ebenfalls nur ein Slave, dieser sendet nach dem Ausführen dieser Funktion seine ROM-ID an den Master
- Match-ROM – Auf dem Bus befinden sich mehrere Slaves. Mit dieser Funktion kann der Master direkt ein Gerät nach dem Senden der ROM-ID ansprechen.
- Resume-ROM – Wurde bereits ein Gerät ausgewählt, so kann der Master nach dem Senden dieser Funktion ein weiteres Funktionskommando an den jeweiligen Slave senden, ohne eine Verbindung erneut aufzubauen.
- Overdrive-Skip-ROM – Auf dem Bus befindet sich nur ein Slave, der Master braucht keine ID zu senden, um das gewünschte Gerät zu adressieren. Er kann direkt nach dem Ausführen dieser Funktion mit dem Senden des Funktionskommandos beginnen. Nach dem Ausführen dieser Funktion wird der Bus im Overdrive-Mode betrieben.
- Search-ROM – Diese Funktion bereitet alle Slaves darauf vor, dass der Master im Anschluss einen Algorithmus zum Auffinden von Geräten und deren ROM-IDs ausführen wird. [19]

2.1.4 Z-Wave

Z-Wave ist ein proprietäres, drahtloses Kommunikations-Protokoll, das von der Firma Sigma Designs und der Z-Wave Alliance speziell für die Steuerung und Überwachung von Geräten entwickelt wurde.

Z-Wave-Alliance

Die Z-Wave-Alliance ist eine im Jahr 2005 von führenden Herstellern gegründete Vereinigung, die sich der Entwicklung und Erweiterung von Z-Wave als Schlüsseltechnologie für Smart-Home- und Businessanwendungen gewidmet hat. Die Mitglieder hatten das gemeinsame Ziel, einen hochentwickelten Standard auf dem Markt zu etablieren, der es ermöglicht, dass unterschiedlichste drahtlose Geräte einwandfrei zusammenarbeiten, unabhängig von einem bestimmten Hersteller. Um diese Interoperabilität zu gewährleisten, muss jedes Z-Wave-Produkt einen strengen Konformitätstest bestehen. Dieser stellt sicher, dass sich das Gerät vollständig an den Z-Wave-Standard hält und somit eine reibungslose Kommunikation mit anderen Geräten gewährleistet.

Mittlerweile haben sich der Vereinigung über 250 unabhängige Firmen angeschlossen. Mit fast 1000 verschiedenen Produkten ist Z-Wave eines der größten Ökosysteme interoperabler funkbasierter Produkte. [21] [22]

Spezifikation

Die Spezifikation von Z-Wave beschreibt im Folgenden:

- Es handelt sich um einen HF-Funkstandard mit geringer Sendeleistung über den sogenannte vermaschte Netzwerke (engl. mesh network) aufgebaut werden können.
- Z-Wave arbeitet mit einer *FSK-Modulation* unterhalb des 1-GHz-Bandes (in Europa um 868 MHz, in den USA um 908,42 MHz) um Interferenzen mit anderen Funktechnologien im 2,4 GHz-Band wie *WLAN, Bluetooth, ZigBee*, etc. zu vermeiden.
- Es wurde speziell für Anwendungen zur Steuerung und Überwachung entwickelt und unterstützt Datenraten bis zu 100 kbit/s.
- Es ist eine Verschlüsselung des Funkverkehrs mit dem Verschlüsselungsstandard *AES125* vorgesehen.
- Die Hardwarenahen Protokollschichten MAC und PHY sind von der *International Telecom Union* (ITU-T) als Standard G.9959 [23] definiert [24]

Hinweis:

Da es sich bei Z-Wave um einen proprietären Standard handelt, sind weitere Technische Details nur Z-Wave-Entwicklern vorbehalten und werden hier nicht weiter behandelt.

2.2 Software

2.2.1 JavaScript

Geschichte

Die Skriptsprache *JavaScript* wurde gegen Ende 1995 erstmals unter dem Namen *LiveScript* in der damals aktuellen Vorabversion des Webbrowsers *Netscape Navigator 2.0* veröffentlicht. [25] Entwickelt wurde sie durch den Programmierer Brandan Eich, der später maßgeblich an der Gründung der *Mozilla Foundation* beteiligt war. [26] Einige Monate später kündigte *Microsoft* an, im *Internet Explorer 3.0* ebenfalls eine Variante von *JavaScript* (aus lizenzrechtlichen Gründen unter dem Namen *JScript*) zu unterstützen. [27]

Im November 1996 begann die Entwicklung des Industriestandards *EMCA-262* auch bekannt als *ECMAScript*, das im Juni 1997 in der ersten Version fertig gestellt wurde. Dieser Standard wurde nun schlussendlich im April 1998 als internationaler Standard unter der ISO/IEC 16262 genehmigt und veröffentlicht. Im Juni 1998 veröffentlichte *Ecma* Version 2 des *EMCA-262* – die Veränderungen an der Version waren aber nur redaktioneller Natur. Die dritte Version des Standards, erhielt einen massiven Funktionsumfang und wurde im Juni 2002 veröffentlicht. [28]

Sowohl JavaScript als auch JScript sind in den heutigen Versionen kompatibel zu ECMAScript, wurden aber durch die jeweiligen Hersteller durch eigene Funktionen erweitert.

Verwendungszweck

JavaScript wurde früher hauptsächlich clientseitig eingesetzt. Typische Anwendungsgebiete waren und sind dynamische Manipulation von Webseiten über das *DOM* [29], Plausibilitätsprüfung von Formulareingaben, Anzeige von Dialogfeldern, Nachladen von Daten per *Ajax* [30 S. 291]. Heutzutage findet *JavaScript* bzw. *ECMAScript* auch Anwendung auf serverbasierten Lösungen wie *Node.js* oder als *JScript.ASP* auf dem *Microsoft IIS* (*Microsoft Internet Information Services)* [31]. Teilweise wird die Sprache *JavaScript* in einigen Spiele- und Anwendungsprogrammen als Skriptsprache verwendet [32]. Die NoSQL-Datenbank MongoDB setzt ebenfalls Javascript ein und kann unter anderem für Abfragen und Aggregationsfunktionen verwendet werden. [33]

In der Spezifikation von *Ecma* wird die Verwendung von *ECMAScript* als Skriptsprache wie folgt erläutert:

> *„A scripting language is a programming language that is used to manipulate, customise, and automate the facilities of an existing system. In such systems, useful functionality is already available through a user interface, and the scripting language is a mechanism for exposing that functionality to program control...“ [28 S. 2]*

Die Verwendung einer serverseitigen Anwendung findet in dieser Spezifikation ebenfalls eine Erwähnung:

> *„A web server provides a different host environment for server-side computation including objects representing requests, clients, and files; and mechanisms to lock and share data. By using browser-side and server-side scripting together, it is possible to distribute computation between the client and server while providing a customised user interface for a Web-based application.“ [28 S. 2]*

2.2.2 Google V8

Google V8 ist eine auf Ausführungsgeschwindigkeit optimierte Open-Source-Javascript-Laufzeitumgebung. Sie befindet sich in der Entwicklung bei *Google Inc.* und wurde erstmals am 3. Juli 2008 als Quellcode veröffentlicht. [34] *Google V8* implementiert *ECMAScript* nach Spezifikation *ECMA-262 Version 5* und wird in der Programmiersprache C++ entwickelt. [35] Mittlerweile ist Google V8 Teil des *Google Chrome*, kann aber in jede andere in C++ programmierte Anwendung integriert werden.

Wie schon erwähnt, ist *Google V8* auf Geschwindigkeit optimiert, genauer genommen auf eine schnelle Ausführung von großen JavaScript Anwendungen. *Google* schreibt dazu:

> *„In several benchmark tests, V8 is many times faster than JScript (in Internet Explorer), SpiderMonkey (in Firefox), and JavaScriptCore (in Safari). If your web application is bound by JavaScript execution speed, using V8 instead of your current JavaScript engine is likely to improve your application's performance. How big the improvement is depends on how much JavaScript is executed and the nature of that JavaScript. For example, if the functions in your application tend to be run again and again, the performance improvement will be greater than if many different functions tend to run only once.“ [36]*

Doch wie schafft es *Google Inc.* schneller als die Konkurrenz zu sein?

Fast Property Access

In *JavaScript* können Eigenschaften von Objekten jederzeit geändert, hinzugefügt oder wieder gelöscht werden. Die meisten *JavaScript*-Engines benutzen eine wörterbuch-ähnliche Struktur als Speicher für die Objekteigenschaften. Jeder Zugriff benötigt eine Suche nach dem objektzugehörigen Speicherbereich. Dieser Ansatz macht den Zugriff auf die Eigenschaften eines Objektes in der Regel viel langsamer als z.B. auf die Instanzvariablen der Programmiersprache JAVA.

Google V8 hingegen erzeugt dynamisch im Hintergrund sogenannte „Hidden Classes“. Diese werden für jedes bekannte Objekt angelegt und eine Informationsstruktur, die fortan mit der Objektinstanz verknüpft ist, hinterlegt. Die Informationsstruktur, die unter anderem die Speicherstelle der einzelnen Objekt-Eigenschaften enthält, dient dem in V8 integrierten *Just-in-Time Compiler* im folgenden zur Optimierung der Objektzugriffe. Da es in JavaScript jederzeit möglich ist, den Objekten neue Eigenschaften hinzuzufügen, muss bei Bedarf die jeweilige Hidden Class ersetzt werden. Dieser Ansatz führt zwar zu erhöhtem Rechenaufwand, wird aber dadurch ausgeglichen, dass einmal erzeugte Hidden Classes bei Bedarf wieder verwendet werden können. [36]

Dynamic Machine Code Generation

Sobald der *JavaScript Source Code* das erste Mal ausgeführt wurde, wird dieser direkt in Maschinencode übersetzt. Es ist kein Zwischen-Byte-Code und kein Interpreter nötig. Der Zugriff auf die Objekteigenschaften wird über ein spezielles Caching-Verfahren abgewickelt. Dieses Verfahren nennt sich „Inline-Caching“ und wurde für die Programmiersprache *Smalltalk* entwickelt.

Bei der ersten Ausführung des Source Codes wird für den Zugriff auf eine bestimmte Objekteigenschaft die aktuelle *Hidden Class* des Objekts bestimmt. Der Zugriff auf die Objekteigenschaft wird durch die Vorhersage optimiert, dass die dazugehörige *Hidden Class* auch für alle zukünftigen Objekte im selben Abschnitt des Codes abgerufen wird. Diese Informationen in der *Hidden-Class* werden dazu genutzt, um den Inline-Cache-Code um die Hidden-Class zu erweitern. Ist diese Vorhersage korrekt, so ist der Wert der Eigenschaft in nur einem Arbeitsschritt zugewiesen bzw. abgerufen. Ist die Vorhersage nicht korrekt, so wird diese Optimierung wiederrufen und aus dem Inline-Cache-Code entfernt. [36]

Efficient-Garbage-Collection

Der Speicher eines Prozesses eines ungenutzten Objektes wird zurückgewonnen, indem dieser ungenutzte Speicher durch den Garbage-Collector wieder freigegeben wird. Um diesen Vorgang möglichste effizient zu gestalten, wird die Programmausführung während eines Garbage-Collection-Zyklus gestoppt. Dabei wird immer nur ein Teil der nicht mehr benötigten Objekte bearbeitet, um größere Ausführungspausen des Programms zu minimieren. Die Laufzeitumgebung weiß zu jeder Zeit, an welcher Stelle im Speicher sich alle verwendeten Objekte und Pointer befinden. Dies vermeidet eine falsche Identifizierung von Objekten als Zeiger. Überlagern sich die Erzeugung eines neuen Objektes sowie die Garbage Collection eines nicht mehr benötigten Objektes, so wird dieser Speicherbereich bevorzugt für die Erzeugung des neuen Objektes genutzt und es werden alle Zeiger auf das neue Objekt aktualisiert. [36]

2.2.3 Node.js

Bei *Node.js* handelt es sich um ein serverseitiges JavaScript-Framework basierend auf der von Google Inc. entwickelten Laufzeitumgebung Google V8 zum Betrieb von Netzwerkanwendungen.

Beschreibung von der Node.js Webseite:

> *„Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.“ [37]*

Threadbasierende Netzwerkapplikation sind relativ ineffizient und schwierig zu programmieren. Die in Node.js eingesetzte ereignisgesteuerte JavaScript-Architektur bietet den Vorteil, dass pro bestehender Verbindung weniger Arbeitsspeicher verbraucht wird als bei Lösungen, die für jede neue Verbindung einen separaten Thread starten [38]. Entwickelt wurde Node.js ursprünglich von dem amerikanischen Entwickler Ryan Dahl mit dem Ziel Webseiten mit Push-Technologie zu versehen [39 S. VII].

Da alle *Node.js* Applikationen unabhängig vom Betriebssystem in JavaScript programmiert werden, ist es möglich, den selben Source-Code in der Node.js-Umgebung sowohl unter MacOS, Windows als auch unter Linux ohne Anpassungen laufen zu lassen.

Module

Node.js sieht die Möglichkeit vor Applikationen modular aufzubauen. Einige dieser Module sind bereits in der Node.js-Binary enthalten und können in entwickelten Applikation verwendet werden. Darüber hinaus ist es möglich, eigene Module zu entwickeln und diese in die eigene Applikation einzubinden. Diese können vorkompilierte Dateien mit der Dateiendung .node, JavaScript (Dateiendung .js) oder JSON Dateien (Dateiendung .json) sein. [40] Das node.js Modul-System folgt der CommonJS-Konvention, einem Standard für JavaScript-Systeme, die außerhalb von Browsern betrieben werden und stellt über eine Variable namens *exports* Zugriff auf Funktionen und Variablen des entsprechenden Moduls her. [41]

Reichen die in JavaScript gegebenen Möglichkeiten nicht aus, so können Module auch um C++ Code erweitert werden. Dazu werden die in C++ implementierten Funktionen mittels Bindings dem Namensraum von JavaScript zur Verfügung gestellt.

3 Analyse und Konzeptentwurf

3.1 Spezifikation und Anforderungsanalyse

3.1.1 Vorüberlegung

Der Raspberry Pi schien als ideale Plattform für diese Arbeit, er ist kostengünstig, hat viele Anschlussmöglichkeiten und bietet genügend Rechenleitung zum Betrieb einer Haussteuerung. Neben dem Raspberry Pi werden aber noch weitere Bauteile benötigt, die auf eine Platine aufgelötet werden sollen. Für den ersten Prototypen sollte hier eine 100 x 160 mm große Lochrasterplatine ausreichen. Damit ein versehentliches Berühren der Anschlüsse oder der auf der Platine befindlichen Bauteile nicht zu einem elektrischen Schlag führt, soll eine Spannung von 12–24 V verwendet werden können, um die Steuerung und eventuell angeschlossene Relais zu versorgen. Da sowohl Raspberry Pi als auch die restlichen Bauteile mit einer Versorgungsspannung von 5 V arbeiten, wurde zur Spannungskonvertierung ein Step-Down-Wandler-Modul gewählt. Dieses ist zwar etwas teurer als eine Schaltung mit einem Linearregler, aber wesentlich effizienter. Da die meisten Funktionen auch über Funk abgewickelt werden können, wurden die Anzahl der digitalen Ein- und Ausgänge auf jeweils 8 beschränkt. Als Funkstandard wurde Z-Wave gewählt. Er bietet viele Vorteile, wie z.B. ein vermaschtes Netzwerk und unterstützt eine Vielzahl von Geräten. Dennoch sollte es eine Möglichkeit geben, kabelgebunden Temperaturen zu erfassen. Der gewählte 1-Wire-Bus ist zwar nicht direkt als Feldbus konzipiert, bietet aber eine vielfältige und kostengünstige Auswahl an digitaler Temperatursensoren und unterstützt Leitungslängen mehrerer hundert Meter.

Da es unter anderem das Ziel war, dem Benutzer der Steuerung ein ansatzweise echtzeitfähiges Webinterface zu bieten, wurde Node.js mit seiner Ereignisgesteuerten Architektur, seiner Möglichkeit mit dem Webbrowser über Socket.IO zu kommunizieren und der Unterstützung des Betriebssystems Linux gewählt.

3.1.2 Spezifikation

Die folgende Spezifikation wurde bei der Entwicklung als Ziel gesetzt und daraufhin dementsprechend umgesetzt.

Hardwarespezifikation	Prototyp 1
Platinengröße	Europlatine (100x160mm)
Spannungsversorgung V_{in}	12 – 24 V DC
Hardware zur Steuerung und Auslieferung des Webinterface	Raspberry Pi
Netzwerkanbindung	über in Raspberry Pi integrierte Netzwerkschnittstelle, per WLAN mit USB-WLAN-Stick
Möglichkeit der Spannungsversorgung des Raspberry Pis über V_{in}	5 V Step-Down-Wandler, über USB-Buchse auf der Platine
Anzahl digitaler Ausgänge (DO)	8 (auf Schraubkontakten)
Spannung an DO	Abh. von Versorgungsspannung V_{in}
max. Ausgangsstrom pro DO	500 mA
Anzahl digitaler Eingänge (DI)	8 (auf Schraubkontakten)
Spannung an DI	Tolerante Eingänge 5–24 V
Anzahl analoger Eingänge	4 (auf Schraubkontakten)
Schnittstelle zur Kommunikation mit drahtlosen Geräten	Z-Wave
Schnittstelle zur Kommunikation mit Temperatursensoren	1-Wire
Anschlussmöglichkeit der 1-Wire-Sensoren	RJ45, Schraubkontakte
Schutz des 1-Wire-Bus gegen elektrostatische Aufladung	Bis 27 kV

Tabelle 2: Hardwarespezifikation - Prototyp 1

Aufgrund einer weiterführenden Entwicklung wurde ein zweiter Prototyp entworfen. Dieser Prototyp sollte in erster Linie die Abmessungen des Raspberry Pis annehmen. Da sich dadurch die Platinengröße knapp halbiert, musste die Spezifikation leicht angepasst werden.

Hardwarespezifikation	Prototyp 2
Platinengröße	Raspberry Pi (85,60 × 56 mm)
Möglichkeit der Spannungsversorgung des Raspberry Pis über V_{in}	5 V Pull-Down-Wandler, direkt über die den Pin-Header am Raspberry Pi
Anzahl analoger Eingänge	entfallen

Tabelle 3: Hardwarespezifikation - Prototyp 2

Die Vorab-Softwarespezifikation wurde wie folgt definiert:

Softwarespezifikation	
Betriebssystem	Linux (Raspbian)
Serverplatform	Node.js
Grafischer Benutzerzugriff	Webbasiert / Browsergestützt

Tabelle 4: Vorab-Softwarespezifikation

3.1.3 Anforderungsanalyse

Zu einer genaueren Spezifikation der Software wurde eine Anforderungsanalyse durchgeführt.

Ermittlung der Stakeholder[3]*.*

In diesem Fall ist die Liste der Stakeholder relativ kurz. Erstens bin es ich, in der Rolle als Entwickler des Systems (stellvertretend für spätere Entwickler) und zweitens der Nutzer des Systems. Weitere Stakeholder könnten ebenfalls in Frage kommen: z.B. Schulungspersonal, Projektgegner, Kulturkreise und andere Meinungsführer. Letztgenannte Stakeholder wurden aber in dieser Anforderungsanalyse nicht berücksichtigt.

[3] Stakeholder sind Personen, die Einfluss auf das betroffene System haben (Systembetroffene)

Definition der Ziele

Die zu definierenden Ziele müssen vollständig, korrekt, konsistent gegenüber anderen Zielen, in sich konsistent, testbar, verstehbar für alle Stakeholder, realisierbar, notwendig, eindeutig und positiv definiert sein.

Ziele des Systems:

- Jegliche Verbindung mit dem System soll verschlüsselt ablaufen.
- Die grafische Benutzeroberfläche soll modern und Grafisch ansprechend aussehen, einfach zu bedienen und sowohl auf mobilen Geräten als auch auf PCs nutzbar sein.
- Benutzer müssen sich mit einem Benutzernamen und Passwort in das System einloggen können.
- Benutzer müssen im eingeloggtem Zustand ihren Benutzernamen und ihr Passwort ändern können.
- Benutzer landen nach erstmaligen Erstellen ihres Benutzerkontos auf einer Seite, die ihnen kurz und knapp die Bedienung des Systems erklärt.
- Benutzer landen nach dem Einloggen auf dem Dashboard, einer Startseite, die der Benutzer mit seinen meistbenutzen Funktionen belegen kann.
- Benutzer sollen Räume anlegen, diese mit Szenen, Temperaturwerten und Steuerungselementen belegen und später aufrufen können.
- Benutzer sollen Z-Wave Geräte in das System integrieren und wieder entfernen können.
- Benutzer sollen den Temperatursensoren Bezeichnung und Ort zuweisen können.
- Benutzer sollen die digitalen Ein- und Ausgänge konfigurieren können.
- Benutzer sollen Geräte oder Dienste z.B. für Push-Notifikationen per HTTP REST API in das System integrieren können.
- Benutzer sollen Szenen (einem vorher definiertem Zustand von einem oder mehreren Geräten – z.B. Alle Geräte aus) anlegen können. Diese Szenen sollen zusätzlich über eine einmalige und sichere URL aufrufbar sein.
- Benutzer sollen Regeln erstellen können, dass z.B. bei Auslösen eines Rauchmelders oder der Betätigung der Klingel bestimmte Geräte eingeschaltet werden oder eine Benachrichtigung per Push-Notifikation an ein Mobilgerät gesendet wird.
- Benutzer sollen Timer (Zeitschaltuhren) einrichten können, die vorher definierte Szenen zu bestimmten Tagen und Uhrzeit auslösen.

- Benutzer sollen Webcam-Streams in das System eintragen können. Diese sollen später über die Benutzeroberfläche aufrufbar sein
- Die Webcam-Streams sollen über die bestehende verschlüsselte Verbindung übertragen werden.
- Die Software soll unter Open-Source Lizenz veröffentlicht werden
- Nachfolgenden Entwicklern soll der Einstieg durch sauberen Code und gute Kommentare vereinfacht werden.

Alle diese Ziele wurden tabellarisch erfasst und um mögliche Auswirkungen auf Stakeholder, Randbedingungen der Entwicklung, Abhängigkeiten von anderen Zielen und sonstige wichtige Informationen erweitert.

Nachfolgend wird es am Beispiel des Zieles „Benutzer müssen sich mit einem Benutzernamen und Passwort in das System einloggen können“ aufgeführt.

Benutzerauthentifizierung

Ziel	Benutzer muss sich mit einem Benutzernamen und Passwort in das System einloggen können.
Stakeholder	Benutzer, Entwickler
Auswirkungen auf Stakeholder	Entwickler muss eine sichere Lösung implementieren, die die Daten des Benutzers schützt und dem Benutzer ermöglicht, sich an dem System anzumelden. Benutzer muss sich zum Einloggen eine Benutzername/Passwort-kombination merken
Randbedingungen	Es muss nach einem geeigneten Verschlüsselungsverfahren gesucht werden.
Abhängigkeiten	Ziel 1 (Verschlüsselte Verbindung)
Sonstiges	-

Tabelle 5: Tabellarische Zieldarstellung – Beispiel Benutzerauthentifizierung

Use Cases

Aus den vorher definierten Zielen lassen sich nun konkrete Use Cases (Anwendungsfälle) erstellen. Das Erstellen der Use-Cases ist ein iterativer Prozess, angefangen mit der Findung der Hauptaufgaben über eine grobe Beschreibung des Anwendungsfalls hin zu einer konkreten Beschreibung des Inhalts.

Eine Definition eines Use-Cases wurde hier am Beispiel der Benutzerauthentifizierung mit folgendem Ergebnis durchgeführt:

Benutzerauthentifizierung	
Nummer des Usecase	U2
Autor	Fabian Behnke
Version	1.0, 10.01.2014, Erstellung
Kurzbeschreibung	Benutzer müssen sich mit einem Benutzernamen und Passwort in das System einloggen.
Stakeholder	Benutzer, Entwickler
Fachverantwortliche(r)	Fabian Behnke (Entwickler, Idee)
Referenzen	Handbuch der Haussteuerung
Vorbedingungen	Benutzerkonto muss eingerichtet sein.
Nachbedingungen	Anzeigen des Dashboards (Startseite)
typischer Ablauf	1. Benutzer öffnet Website der Steuerung. 2. Benutzer meldet sich mit Benutzernamen und Passwort an. 3. Benutzer erhält Systemzugriff.
Alternative Abläufe	Benutzer erhält keinen Zugriff.
Kritikalität	sehr hoch, Grundvoraussetzung für Anwendung des Systems
funktionale Anforderungen	Funktion ermöglicht Zugriff auf System

Tabelle 6: Use-Case - Beispiel Benutzerauthentifizierung

Use-Case-Diagramm

Um die Use-Cases übersichtlich darzustellen und zu verwalten, gibt es spezielle Projektierungssoftware. Dazu zählen z.B. *Microsoft Visio* oder *Visual Paradigm*. Mit ihnen lassen sich unter anderem komplette Softwareprojekte abwickeln und planen.

Aus den vorher definierten Use-Cases lässt sich ein sogenanntes Use-Case Diagramm mit allen zugehörigen Stakeholdern und Abhängigkeiten erstellen.

Hier eine beispielhafter Auszug des Use-Case-Diagramms für diese Anwendung:

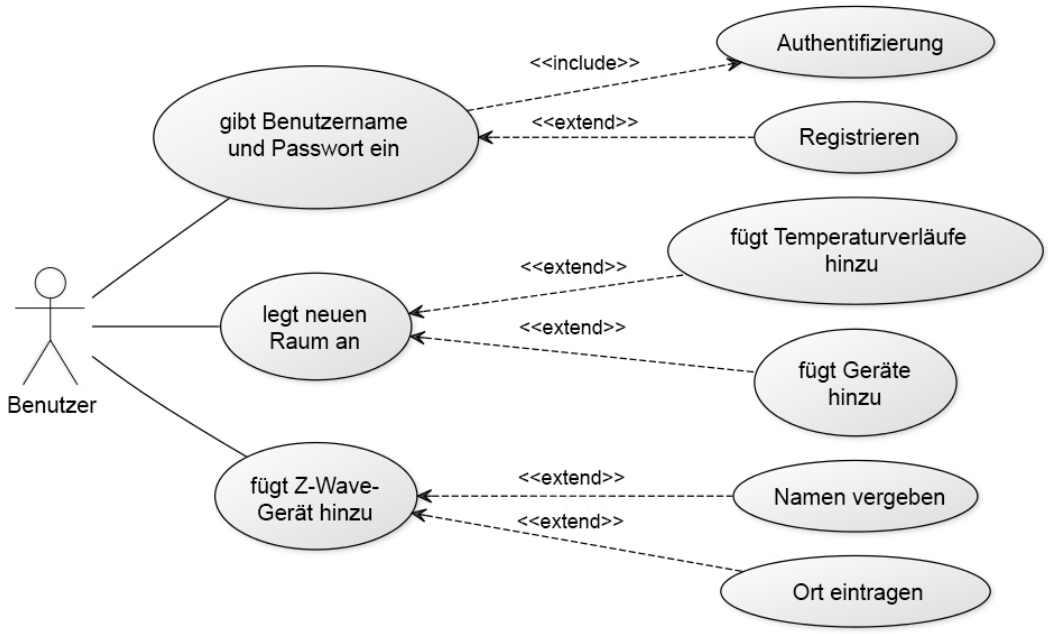


Abbildung 11: Use-Case-Diagramm zur besseren Darstellung (Beispiel)

3.2 Eignung von Node.js als Serverplattform

Node.js wurde speziell für den Serverbetrieb entwickelt. Seine ereignisgesteuerte JavaScript-Architektur bietet den Vorteil, dass pro bestehender Verbindung weniger Arbeitsspeicher verbraucht wird als bei Lösungen, die für jede neue Verbindung einen separaten Thread starten [38]

Dieser *„Single-Threading“*-Betrieb hat zur Folge, dass Kommandos nacheinander abgearbeitet werden und im ersten Schritt keine Parallelisierung erlaubt.

Um dennoch zu gewährleisten, dass Prozesse nicht durch Anfragen blockiert werden, besitzt Node.js ein ausgelagertes I/O-System. Dies bedeutet, dass Schreib- und Leseoperationen in Node.js asynchron durchgeführt werden. [38]

Alle Anfragen an I/O und deren *Callbacks*[4] werden in einer Event-Loop gespeichert und durch einen externen Prozess abgearbeitet. Ist die jeweilige I/O-Anfrage fertiggestellt, werden die abgerufenen Daten an die vorher definierte Callback-Funktion übergeben und die Bearbeitung läuft regulär weiter. [42]

Ein Nachteil an dieser Architektur ist, dass eine Verwendung von Mehrkernprozessoren, wie es bei Servern in der Regel üblich ist, kaum Einfluss auf die Ausführungsgeschwindigkeit von Node.js hat. Diese Einschränkung stört bei dem Einsatz auf dem *Raspberry Pi* nicht, da dieser nur einen Prozessorkern besitzt.

Dennoch bietet Node.js weitere Vorteile gegenüber anderen Systemen. Es sind unter anderem viele Open-Source Module verfügbar, die für den Betrieb unserer Serverplattform hilfreich sind, z.B.:

Express

Ein Web Application Framework, das eine komfortable Möglichkeit bietet, Web Applikationen in Node.js zu realisieren. (Mehr im Kapitel 4.2.7)

Socket.IO

In traditionellen Webapplikationen antwortet der Server nur auf Anfrage des Clients. Socket.IO bietet die Möglichkeit diese Grenze zu durchbrechen. Durch Technologien wie Websockets, *Adobe Flash* Socket, AJAX long polling, AJAX multipart streaming, Forever Iframe und JSONP Polling wird sichergestellt, dass es in jedem derzeit verfügbaren Browser möglich ist, eine Verbindung zum Server aufzubauen und dieser direkt Änderungen an den Browser senden kann. [43]

[4] Callbacks sind Funktionen, die ausgeführt werden sollen, wenn eine Bearbeitung der vorherigen Funktion fertiggestellt ist

3.3 Wahl einer geeigneten Datenbank

Für das vorliegende Projekt wurden drei Open-Source Datenbanksysteme miteinander verglichen. Die Wahl fiel auf das relationale Datenbanksystem *MySQL* sowie auf die beiden NoSQL[5] Datenbanksysteme *MongoDB* und *CouchDB*.

Bei dem Vergleich der drei Datenbanken wurde in erster Linie auf Geschwindigkeit und Entwicklungsaufwand geachtet.

Geschwindigkeit

Um die Geschwindigkeit der Datenbanken vergleichen zu können, wurde ein Testprogramm mit folgendem Ablauf entwickelt:

1. JavaScript Objekt mit bestimmter Anzahl X von Elementen erzeugen. Jedes Element erhält ID, SHA512-Hashwert dieser ID und einem String mit einem von 8 deutschen Städtenamen.
2. Array mit der selben Anzahl X an Elementen erzeugen. Jedes Element besteht aus einer ganzzahligen Zufallszahl zwischen 0 und der Anzahl X minus 1.
3. Zeitmessung „Schreiben“ starten.
4. Alle Elemente des JavaScript Objekts sequenziell in Datenbank schreiben.
5. Zeitmessung „Schreiben“ stoppen.
6. Zeitmessung „Lesen“ starten.
7. X Objekte aus Datenbank mit ID=Zufallszahlen aus Array holen.
8. Zeitmessung „Lesen“ stoppen.

Das beschriebene Testprogramm wurde für alle Datenbanken für die Anzahl der Lese-/Schreibzugriffe X=[10, 20, 50, 100, 500, 1000, 2000] einhundert Mal ausgeführt und die Ergebnisse über die Anzahl der Durchläufe gemittelt.

Um die Testbedingungen so nahe an der Realität wie möglich zu gestalten, wurden die Tests auf der selben Systemumgebung durchgeführt, auf dem die Datenbank später betrieben werden soll (Raspberry Pi, Node.js). Als Datenbanktreiber für Node.js kamen zum Einsatz:

- Mysql (MySQL)
- Mongoose (MongoDB)
- Nano (CouchDB)

[5] NoSQL Systeme zeichnen sich dadurch aus, dass sie nicht den Ansatz einer rationalen Datenbank verfolgen, die ihre Daten in Tabellen mit einem bestimmten Schema abspeichern, sondern in Dokumenten ohne vordefiniertes Schema.

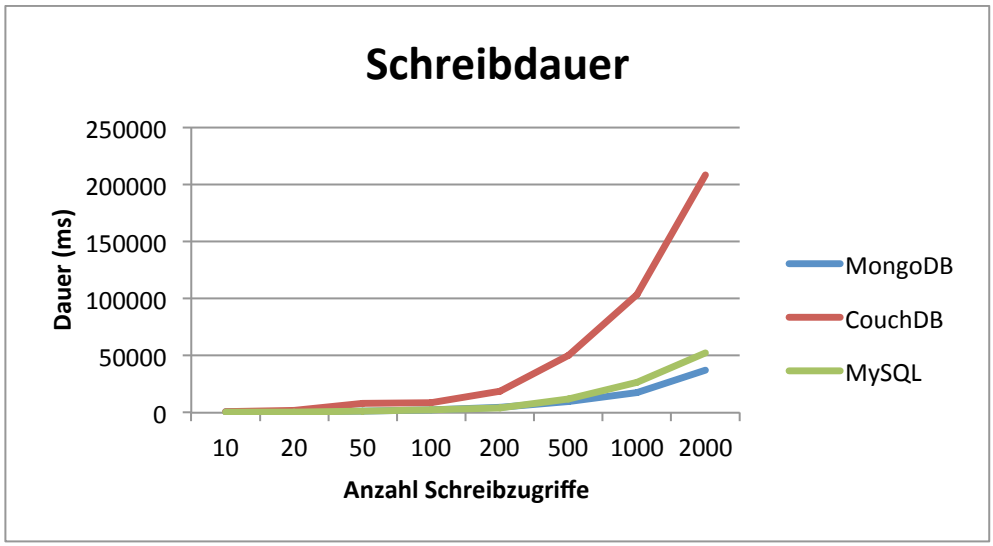


Abbildung 12: Schreibdauer – Datenbankenvergleich

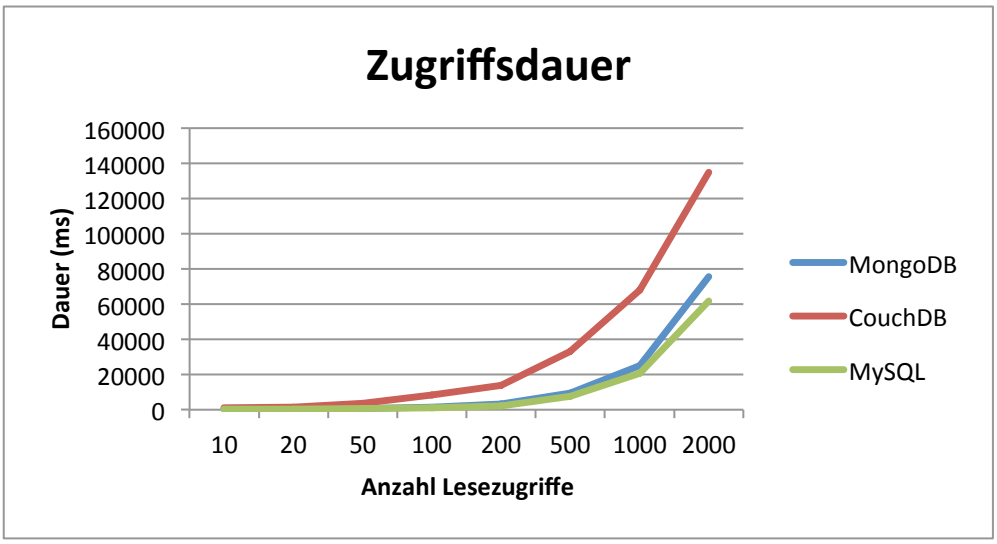


Abbildung 13: Zugriffsdauer – Datenbankenvergleich

Die gemessenen Werte zeigen einen deutlichen Geschwindigkeitsvorteil bei MongoDB und MySQL gegenüber CouchDB, sowohl im Schreibzugriff als auch im Lesezugriff. Im Schreibzugriff ist MongoDB etwas schneller als MySQL, im Lesezugriff ist es umgekehrt. Insgesamt gesehen sind MySQL und MongoDB von der Geschwindigkeit her (bei dieser geringen Anzahl von Zugriffen) in etwa gleich schnell.

Enwicklungsaufwand

Gerade beim Entwicklungsaufwand sind die beiden NoSQL-Datenbanken im Vorteil. Es müssen keine Schemata für einzelne Tabellen und deren Relationen zu einander entwickelt werden. Für alle drei Datenbanken sind Treibermodule

für Node.js erhältlich, sodass hier kein weiterer Entwicklungsaufwand notwendig ist. Der Zugriff auf die NoSQL Datenbanken wird über JavaScript Funktionen zum Speichern und Abrufen der Daten gesteuert. Der Zugriff auf die MySQL Datenbank funktioniert ebenfalls über JavaScript Funktionen, allerdings müssen hier SQL-Statements übergeben werden. An verfügbaren Open-Source Modulen zur Nutzung der Datenbanken als Session Store mangelt es bei allen drei Datenbanken nicht.

Ergebnis

Allgemein wären alle Datenbanken für dieses System geeignet gewesen. Die Wahl viel schlussendlich auf MongoDB. Der etwas geringere Entwicklungsaufwand und der Geschwindigkeitsvorteil führten zu dieser Wahl.

3.4 Wahl einer geeigneten Open-Source Lizenz

Der komplette Quellcode der im Zusammenhang mit dieser Arbeit entwickelten Software soll nach Fertigstellung im Internet veröffentlicht und kostenlos zur Verfügung gestellt werden.

Um mich als Entwickler vor möglichen rechtlichen Konsequenzen zu schützen, sollte nun eine geeignete Open-Source-Lizenz gefunden werden.

Da es mir bei dieser Arbeit darum geht, dass jede Person (einschließlich juristischer Personen wie Firmen) die Möglichkeit haben soll, meinen Quellcode uneingeschränkt nutzen zu können, sollte die verwendete Lizenz diese Klausel beinhalten.

Wird mein Quellcode (oder Teile dessen) in einem anderen Projekt verwendet, so sollte es möglich sein, dieses Projekt auch unter einer anderen Lizenz zu veröffentlichen, als der verwendete Quellcode.

Auch wenn der Quellcode uneingeschränkt nutzbar ist, so sollte es nicht möglich sein, dass z.B. Firmen behaupten können, sie hätten den Quellcode komplett selber geschrieben. Das heißt ein Urheberrechtsvermerk mit meinem Namen sollte weiterhin bestehen bleiben.

Des Weiteren sollte mich die gewählte Open-Source-Lizenz vor Datenverlust Dritter und jeglicher Rechtsverletzung schützen.

Alle vorher genannten Punkte werden z.B. durch die MIT-Lizenz oder BSD-Lizenz abgedeckt. Der Unterschied beider Lizenzen liegt darin, dass die BSD-Lizenz ausschließt den Namen des Urhebers zu Werbezwecken zu verwenden.

Da ich kein Problem darin sehe, dass mit meinem Namen geworben wird und ich die größtmögliche Anzahl an Entwickler erreichen möchte, wird die hier entstandene Software unter MIT-Lizenz veröffentlicht.

Lizenztext

3.5 Vorauswahl der benötigten Bauteile

Bei der Auswahl der Komponenten habe ich mir im Vorfeld Gedanken gemacht und hier in einer Tabelle zusammengefasst. Eine nähere Betrachtung einiger Komponenten ist im Kapitel 4.1(Hardware) zu finden.

Prototyp 1

Anzahl	Bauteil	Beschreibung
1	Raspberry Pi	Modell B 512 MB Ram
1	Europlatine 100x160mm	Lochraster 2,54 mm
1	Step-Down-Wandlermodul	basierend auf dem LM2596
1	FDC6301N	Mosfet N Kanal Dual 25 V 0.22 A
1	MCP23017-E/SP	I^2C 16 Bit I/O-Erweiterung
1	ULN2803AN	Darlington Transistor Netzwerk
1	MCP3423-E/SL	4 Kanal Delta-Sigma-Wandler
1	DS2482S-100+	I^2C-1-Wire Bridge
1	DS9503P+	1Wire-Protection-Diode
1	Funkmodul „RaZberry“	Z-Wave-Funkmodul
1	SOIC14 Adapterplatine	SOIC14 auf 2,54 mm Raster
2	SOIC8 Adapterplatine	SOIC8 auf 2,54 mm Raster
1	SOT23-6 Adapterplatine	SOT23-6 auf 2,54mm Raster
3	Kondensator 100 nF	Keramik, 16 V, 10 % Toleranz
1	Kondensator 10 µF	Tantal, 16 V, 10 % Toleranz
1	Widerstand 100 Ohm	¼ W 5 % Toleranz
4	Widerstand 10 kOhm	¼ W 5 % Toleranz
8	Widerstand 22 kOhm	¼ W 5 % Toleranz
8	Widerstand 47 kOhm	¼ W 5 % Toleranz
8	Zenerdiode 4,7 V	5 W 5 % Toleranz
12	Schraubkontakt 1,5 mm^2	5,08 mm Raster, 3 Anschlüsse
1	RJ45 Buchse	UTP
1	USB Buchse	Platinenmontage
2	Stapelbare Buchsenleiste	2,54 mm Raster, 8 Pins
2	Pin Header Leiste	2,54 mm Raster 5 Pins
1	Abstandsbolzen	15 mm, M3, Nylon
1	Sechskantmutter	M3, Nylon
1	Schraube	M3, Nylon

Tabelle 7: Bauteilliste - Prototyp 1

Aufgrund der Entwicklung eines zweiten Prototypen (weitestgehend basierend auf SMD-Technologie) wurde die Auswahl leicht verändert.

Prototyp 2

Anzahl	Bauteil	Beschreibung
1	Raspberry Pi	Modell B 512 MB Ram
1	TSR-1-2450	Step-Down-Schaltregler
1	FDC6301N	Mosfet N Kanal Dual 25V 0.22A
1	MCP23017-E/SO	I^2C 16-Bit I/O Erweiterung
1	ULN2803ADW	Darlington Transistor Netzwerk
1	DS2482S-100+	I^2C-1-Wire-Bridge
1	DS9503P+	1Wire-Protection-Diode
1	Funkmodul „RaZberry“	Z-Wave-Funkmodul
2	Kondensator 100nF	Keramik, 16 V, SMD 0805, 10 %
1	Widerstand 100Ohm	¼ W, SMD 0805, 5 % Toleranz
1	CAY16-103J4LF	Widerstandsnetzwerk 10 kOhm
1	4816P-T01-223LF	Widerstandsnetzwerk 22 kOhm
1	4816P-T01-473LF	Widerstandsnetzwerk 47 kOhm
4	DZ4J047K0R	Dual Zenderdiode 4.7 V 0.2 W
1	Schraubkontakt 1,5 mm^2	5 mm Raster, 2 Anschlüsse
1	Schraubkontakt 1,5 mm^2	5 mm Raster, 3 Anschlüsse
1	Schraubkontakt 1,5 mm^2	5 mm Raster, 4 Anschlüsse
2	Schraubkontakt 1,5 mm^2	5 mm Raster, 9 Anschlüsse
1	RJ45 Buchse	STP
1	ESQ-113-13-T-D	Female Header 2x13 Pins 2,54 mm Raster erhöht
2	826925-5	Male Header 2x5 Pins 2,54 mm Raster
1	Abstandsbolzen	15 mm, M3, Nylon
1	Sechskantmutter	M3, Nylon
1	Schraube	M3, Nylon

Tabelle 8: Bauteilliste - Prototyp 2

3.6 Unterstützte Z-Wave-Geräteklassen

Wie im Kapitel 2.1.4 erwähnt, gibt es eine Vielzahl verschiedener Z-Wave-Geräte. Dazu zählen z.B.:

- Schaltgeräte
- Dimmer
- Leistungsmessgeräte
- Rollladensteuerung
- Schlösser
- Alarmgeber
- Türkontakte
- Temperaturfühler
- Bewegungsmelder
- Rauchmelder

Um den Entwicklungsaufwand der Software in einem gewissen Rahmen zu halten, wurde die Unterstützung auf folgende Geräteklassen begrenzt:

- Schaltgeräte
- Dimmer
- Leistungsmessgeräte
- Rauchmelder

Zur Entwicklung standen folgende Z-Wave-Geräte zur Verfügung:

Everspring AN158

Ein kombiniertes Schalt- und Leistungsmessgerät für die Steckdose.

Es kann Geräte mit einer Stromaufnahme bis zu 16 A schalten und dessen Stromverbrauch messen.

Abbildung 14: Everspring AN158 - Schalt- und Leistungsmessgerät

Z-Wawe.me-Dimmer

Dieser Dimmer ist eine softwareoptimierte Version des Düwi ZW EDAN 300. Er dimmt Lasten von bis zu 300 W und ist für einen Wandeinbau geeignet. Es wird kein Neutralleiter benötigt, somit können vorhandene Schalter einfach durch diesen Dimmer ersetzt werden.

Es sind Abdeckrahmen und Schaltwippen für die Schalterprogramme Düwi Everlux und Busch Jäger Duro 2000 erhältlich.

Abbildung 15: Z-Wave.Me - Dimmer

Vision Security ZS 6101

Ein optischer Rauchmelder, der bei Detektierung von Rauch, ein sowohl optisches als auch akustisches Signal von sich gibt. Des Weiteren besitzt dieser Rauchmelder eine Z-Wave-Funk-Schnittstelle, über die eine Benachrichtigung an eine Z-Wave-Steuerung möglich ist.

Abbildung 16: Vision Security ZS 6101 - Rauchmelder

3.7 Testkonzept zur Überprüfung der Hardware

Nach der Fertigstellung der Prototypen werden alle Anschlüsse auf korrektes Potential überprüft und mit dem Schaltplan abgeglichen.

Darüber hinaus wurde ein Testprogramm geschrieben, das alle Bausteine anspricht, um Ein- und Ausgänge, 1-Wire-Temperatursensoren, und analoge Eingänge (nur 1. Prototyp) auf die jeweilige Funktion zu überprüfen.

Die jeweiligen Prototypen werden daraufhin an einen Testaufbau, bestehend aus 8 LED mit Vorwiderständen, 8 Tastern, einem 1-Wire-Temperaturfühler und 4 Drehpotentiometern (nur Prototyp 1), angeschlossen und das Testprogramm gestartet.

Nun werden alle Funktionen der einzelnen Bauteile anhand dieses Testaufbaus überprüft.

Die Kommunikation mit dem Z-Wave-Funkmodul kann in der mitgelieferten Controllersoftware überprüft werden.

3.8 Interaktion von Hard- und Software

Der Raspberry Pi kommuniziert über diverse Hard- und Softwareschnittstellen mit den jeweiligen Komponenten.

Die nachfolgende Grafik (Abbildung 15) verdeutlicht diese Kommunikation zwischen Hard- und Software.

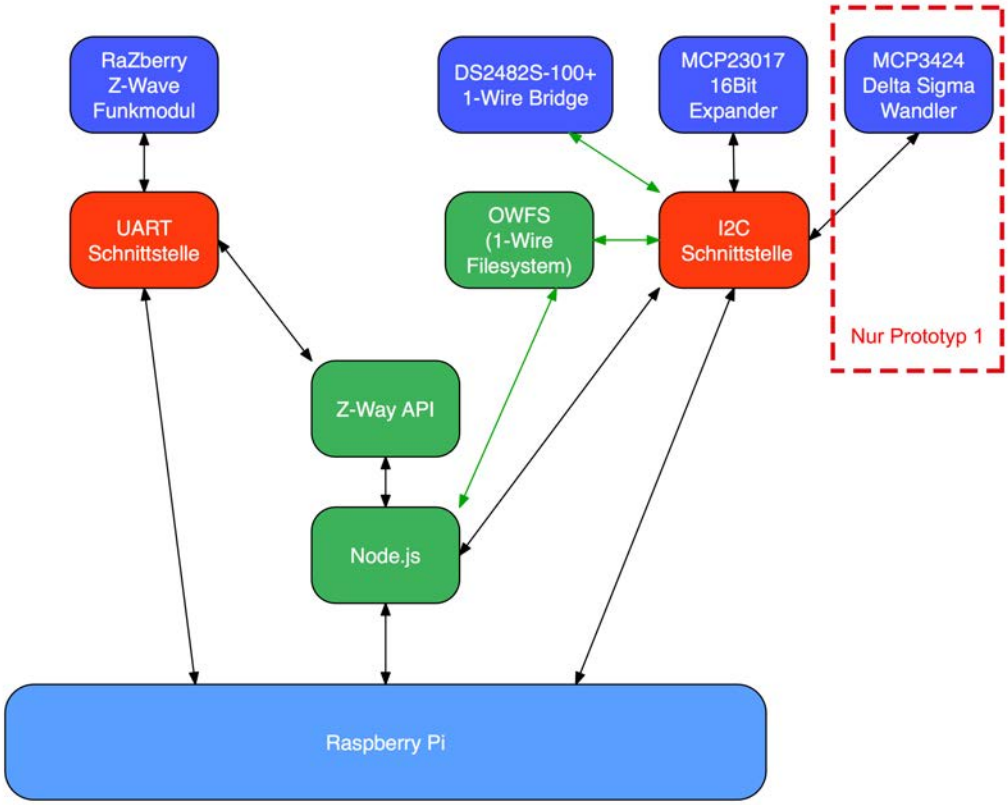


Abbildung 17: Interaktion von Hard- und Software

4 Entwicklung und Umsetzung

4.1 Hardware

4.1.1 Spannungsversorgung

Als Spannungsversorgung soll ein handelsübliches Netzteil mit 12–24 V DC mit mindestens 1A dienen. Darüber versorgt werden sollen:

- Raspberry Pi
- alle sich auf der Platine befindlichen IC's und Module
- optional: Relais zum schalten von höheren Spannungen/Strömen

Da der Raspberry Pi sowie die sich auf der Platine befindlichen IC's eine Spannungsversorgung von 5 V benötigen, muss die Eingangsspannung von 12–24 V möglichst effizient in 5 V gewandelt werden.

Ein günstiger Linearregler würde bei einer Eingangsspannung von 12 V, einer Ausgangsspannung von 5 V und einem Ausgangsstrom von 1 A eine Abwärme mit einer Leistung von 7 Watt erzeugen.

Berechnung der Abwärme eines Linearreglers:

Aus der Spannungsdifferenz

$$\Delta U = U_{Eingang} - U_{Ausgang} = 12\,V - 5\,V = 7\,V$$

und der dem maximalem Ausgangsstrom von 1 A kommen wir auf eine Verlustleitung von

$$P_{Verlust} = \Delta U * I_{max} = 7\,V * 1\,A = 7\,W$$

Ein LM7805 mit TO-220 Bauform besitzt einen Wärmewiderstand von 50 °C/W, somit würde die Verlustleitung am Gehäuse eine Temperatur von

$$T = P_{Verlust} * 50\,° \frac{C}{W} = 7\,W * 50\,° \frac{C}{W} = 350\,°C$$

verursachen.

Diese beachtliche Temperatur könnte man nur durch einen großen Kühlkörper kompensieren.

Bessere Methode - Schaltregler

Eine deutlich effizientere Spannungswandlung kann man durch einen Step-Down Schaltregler erzielen. Diese meist integrierten Bausteine erzielen Wirkungsgrade bis zu 96 % und werden dadurch weniger warm.

Prototyp 1:

Hier wurde ein Step-Down-Wandlermodul basierend auf dem LM2596 Spannungswandler verwendet. Dieses lässt eine maximale Eingangsspannung von bis zu 35 V zu und kann am Ausgang über ein Präzisionspotentiometer von 1,23–30 V einstellt werden. Das Modul liefert einen maximalen Ausgangsstrom von bis zu 3 A.

Abbildung 18: LM2596 Wandlermodul

Prototyp 2:

Aus Platzgründen musste auf ein Bauteil zurückgegriffen werden, das um einiges kleiner ist als das Modul des ersten Prototypen. Das verwendete Bauteil TSR-1-2450 von der Firma Traco Electronic ist nur geringfügig größer als ein Linearregler mit TO-220-Gehäuse, liefert aber mit seiner Funktion als Schaltregler eine Spannung von 5 V mit bis zu 1 A an Ausgang bei einem Wirkungsgrad mit bis zu 94 Prozent. Am Eingang lassen sich 6,5–36 V DC anschließen.

Des Weiteren ist es pinkompatibel zum oben genanntem Linearregler und benötigt keine weitere äußere Beschaltung. Somit ist dieses Bauteil ideal, um ineffiziente Linearregler durch effiziente Schaltregler zu ersetzen.

Abbildung 19: TSR-1-2450 Schaltregler

Nähere Informationen können dem Datenblatt [44] entnommen werden.

4.1.2 Pegelwandlung am I²C - Bus

Die an dem Raspberry Pi vorhandene I²C-Schnittstelle arbeitet mit einem Pegel von 3,3 V. Da die verwendeten Bausteine MCP23017, DS2482S-100 und der im ersten Prototypen verwendete ΔΣ-Wandler MCP3423 am zuverlässigsten mit einem Pegel von 5 V arbeiten, musste eine bidirektionale Pegelwandlung vorgesehen werden.

Dazu wurde eine Schaltung aus einen integriertes Bauteil, bestehend aus zwei n-Kanal-MOSFETs und 4 Pull-Up-Widerständen, aufgebaut.

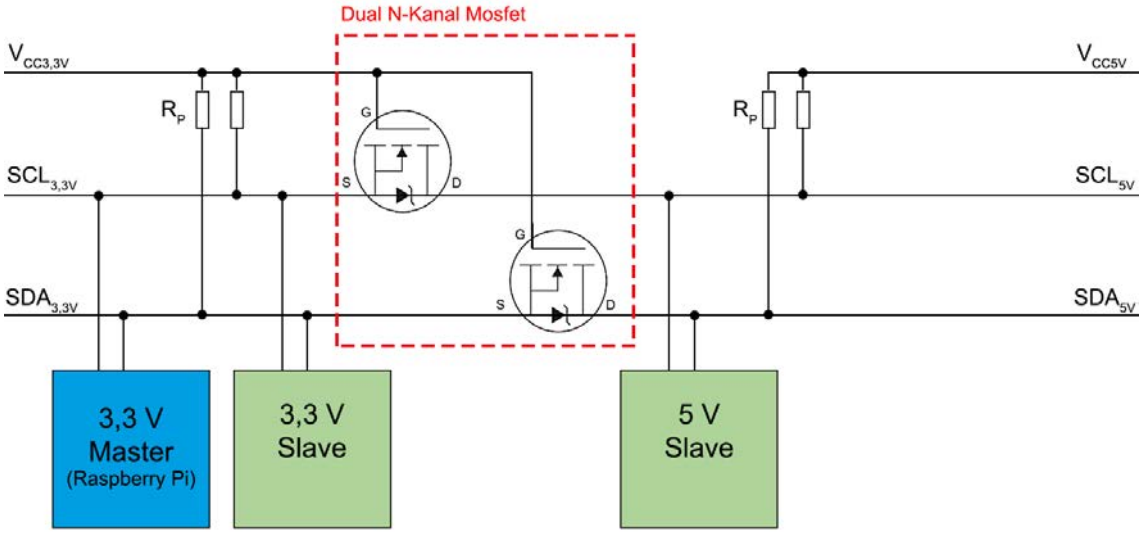


Abbildung 20: Pegelwandlung mit Dual-Mosfet

Wird auf der 3,3-V-Seite eine logische ’0’ angelegt (0 V), so befindet sich der Source-Anschluss des N-Kanal-MOSFETs auf Massepotential. Daraus resultiert eine positive Gate-Sourcespannung, wodurch der Transistor durchschaltet und die 5-V-Seite ebenfalls auf Massepotential zieht. Dies entspricht wiederum einer logischen ’0’.

Wird auf der 3,3-V-Seite eine logische ’1’ angelegt (3,3 V), so liegen Gate- und Source-Anschluss auf demselben Potential. Der Transistor sperrt und die 5-V-Seite wird über den Pull-Up-Widerstand(R_{P5V}) auf 5 V gezogen, was wiederum einer logischen ’1’ entspricht.

Betrachtet man den Pegelwandler nun aus der anderen Richtung, ergibt sich folgendes Szenario:

Wird auf der 5-V-Seite eine logische ’0’ angelegt (0 V), beginnt durch die herstellungsbedingte parallelgeschaltete Substratdiode zu leiten und zieht die 3,3 V

Seite auf annäherndes Massepotential (abzüglich Durchlassspannung der Diode). Dies ergibt ein positive Gate-Sourcespannung, wodurch der Transistor durchschaltet und die 3,3-V-Seite endgültig auf Massepotential (logisch ’0’) zieht.

Wird auf der 5-V-Seite eine logische ’1’ angelegt (5 V), so wird der Source-Anschluss des N-Kanal-MOSFETs über den Pull-Up-Widerstand R_{P3V3} auf 3,3 V gezogen. Source- und Gate Anschluss liegen auf demselben Potential und der Transistor sperrt. Gleichzeitig liegt durch den Widerstand auf der 3,3 V Seite die Versorgungsspannung von 3,3 V an. Dies entspricht wiederum einer logischen ’1’.

Nähere Informationen können der Application Note 10441 von NXP Semicoductors entnommen werden [45].

4.1.3 GPIO Erweiterung

Der Raspberry Pi verfügt über 17 GPIO Pins, von denen insgesamt 9 für Anwendungen wie I^2C, UART und SPI reserviert sind. Es bleiben also insgesamt 8 Anschlüsse, die als „echte“ GPIO-Anschlüsse genutzt werden können. Da sich aber in den Vorüberlegungen als Ziel gesetzt wurde, 8 digitale Eingänge sowie 8 Ausgänge zu realisieren, musste eine Erweiterung dieser Anschlüsse vorgenommen werden.

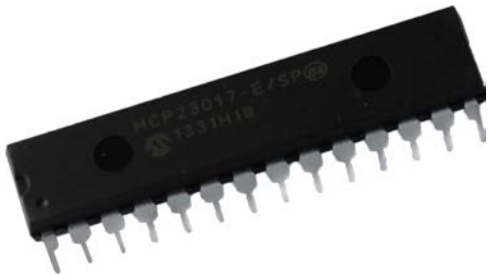


Abbildung 21: MCP23017 im PDIP *Package*

Dazu wurde das Bauteil MCP23017 der Firma Microchip Technology Inc. verwendet. Es ermöglicht eine Erweiterung um insgesamt 16 zusätzliche als Ein- oder Ausgang programmierbare Anschlüsse. Das Bauteil ist in zwei verschiedenen Ausführungen, sowohl für den SPI (MCP23S17) als auch für den I^2C-Bus (wurde hier verwendet) erhältlich. Die Bus-Adresse dieses Gerätes lässt sich über 3 Adress-Pins einstellen, sodass bis zu 8 solcher Geräte an einem Bus betrieben werden können. Des Weiteren verfügt es über zwei Anschlüsse, die als Interrupts genutzt und über das Konfigurationsregister programmiert werden können.

Nähere Informationen können dem Datenblatt [46] entnommen werden.

4.1.4 Tolerante Eingänge

In der Spezifikation wurde festgelegt, dass die Eingänge des Systems eine Eingangsspannung von 5–24 V zulassen sollen. Daher wurde eine einfache Schaltung aus zwei Widerständen und einer Zender-Diode entwickelt, die vor jeden der Eingänge geschaltet wurde.

Der MCP23017 detektiert am Eingang Spannungen unterhalb von 0,2 V als LOW- und oberhalb von 0,8 V als HIGH-Pegel. Das Ziel dieser Schaltung ist es also sicher, die gerade genannten Spannungspegel zu erreichen, ohne aber zu viel Verlustleistung in Form von Wärme zu erzeugen.

Die Schaltung wurde mit Hilfe der Software P-Spice simuliert und auf die eben genannten Punkte optimiert.

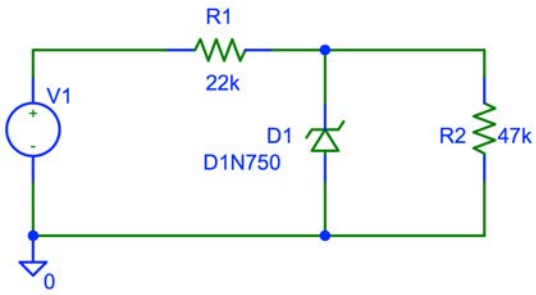


Abbildung 22: P-Spice Simulation - Schaltung (toleranter Eingang)

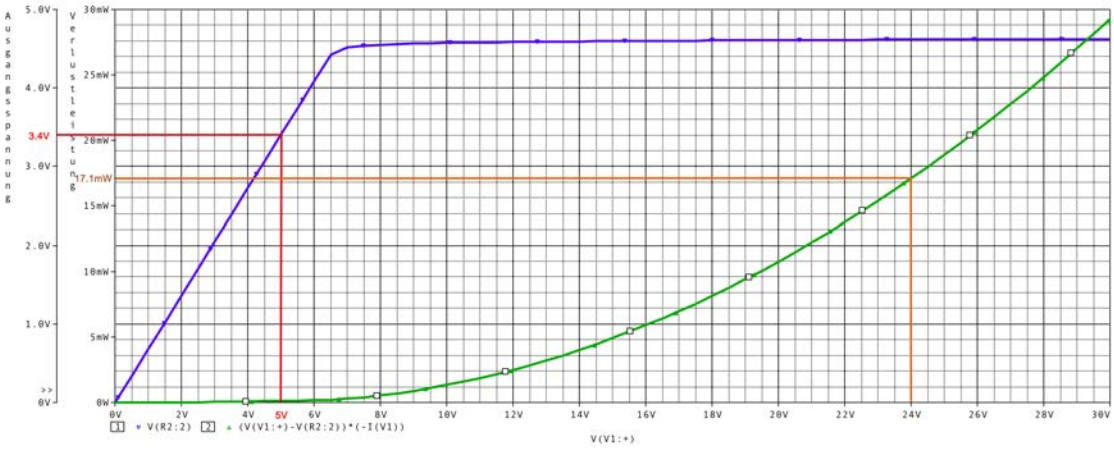


Abbildung 23: P-Spice Simulation – Diagramm (toleranter Eingang)

Als Zenerdiode wurde eine D1N750 Standard-Zenerdiode mit einer Durchbruchspannung von 4,7 V verwendet. Für den Vorwiderstand R1 stellte sich ein Wert von 22 kOhm als ideal heraus. Als Pull-Down-Widerstand wurde ein 47 kOhm Widerstand verwendet, dieser zieht den Eingang des MCP23017 sicher auf Masse. Der Eingangswiderstand des MCP23017 ist sehr gering und konnte deshalb bei der Simulation vernachlässigt werden.

Insgesamt stellt sich für eine Eingangsspannung von 5 V nun am Ausgang eine Spannung von 3,4 V ein. Dies stellt sicher, dass der MCP23017 diese Spannung sicher als High-Level detektiert. Bei einer Eingangsspannung von 24 V tritt eine Verlustleistung von 17,1 mW auf, was bei eben genannter Spannung einen sehr geringen Strom von 712,5 µA entspricht.

4.1.5 Ausgänge zur Relaisansteuerung

In der Spezifikation wurde festgelegt, dass die Ausgänge des Systems eine Spannung in Höhe der Versorgungsspannung V_{in} bei einem maximalen Strom von 500 mA liefern sollen, um z.B. handelsübliche Relais ansteuern zu können.

Da die Ausgänge des MCP23017 eine Ausgangsspannung in Höhe von 5 V bei einem Strom von nur 25 mA liefern, musste eine Möglichkeit gefunden werden, um die vorher spezifizierten Werte zu erreichen. Des Weiteren musste sichergestellt werden, dass die Schaltung nicht durch ein eventuell angeschlossenes Relais durch die beim Abschalten der Spule induzierte Spannung zerstört wird.

Um dieses zu erreichen, wurde das ULN2803A von Texas Instruments verwendet und direkt an die Ausgänge des MCP23017 angeschlossen. Es handelt sich um ein integriertes Bauteil, bestehend aus einem Darlington Array (8 Darlington Schaltungen) und integrierter Freilaufdioden, um induktive Bauteile wie Relais zu schalten. An den insgesamt 8 Eingängen (1B-8B) des Bauteils lassen sich Spannungen bis zu 30 V anschließen. Die 8 Ausgänge (1C-8C) schalten je nach angelegter Spannung am COM-Pin jeweils bis zu 50 V bei einem Maximalstrom von 500 mA.

Hinweis: Die Ausgänge sind durch die Darlington Schaltung bedingt nach Masse geschaltet.

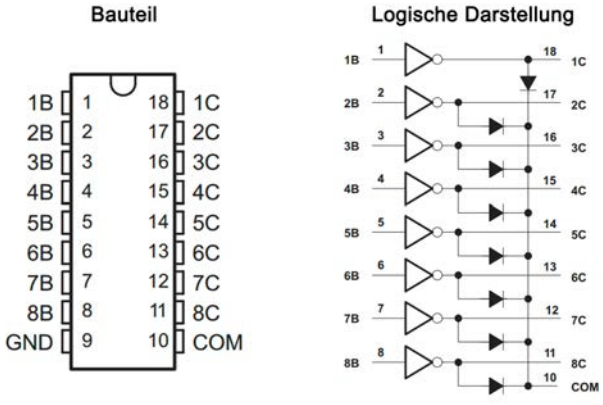


Abbildung 24: ULN2803 - Bauteil und logische Darstellung [47]

Nähere Informationen zu diesem Bauteil können dem Datenblatt [47] entnommen werden.

4.1.6 Funkmodul

In der vorher definierten Spezifikation wurde festgelegt, dass das System über eine Schnittstelle zur Kommunikation mit drahtlosen Z-Wave Geräten verfügen soll.

Dazu wurde das von der Fima Z-Wave.Me produzierte Funkmodul „RaZberry“ verwendet. Dieses Modul wurde speziell für den Raspberry Pi entwickelt und kommuniziert mit diesem über die integrierte UART-Schnittstelle. Es beinhaltet alle Komponenten, um aus einen Raspberry Pi einen voll funktionsfähigen Z-Wave Gateway zu bauen.

Abbildung 25: RaZberry - Z-Wave-Funkmodul [48]

Das Modul besteht im Grunde aus drei Komponenten:

- Der **Hardware**, die „RaZberry Daugter Card“. Eine kleine Platine, die auf den GPIO Anschluss des Raspberry Pis gesteckt wird. Sie wird über die vom Raspberry Pi bereitgestellte Spannung von 3,3 V gespeist und kommuniziert über die UART-Schnittstelle mit dem Einplatinencomputer. Auf der Platine befindet sich ein Sigma Designs 3102 Z-Wave-Transceiver-Modul, ein externer 32K SPI Flash für Netzwerkdaten und eine PCBA Antenne. Zwei LEDs zeigen den Status des Z-Wave-Controller-Chips. [49]
- Der **Firmware** – Die auf dem *Sigma Designs Transceiver Chip* enthaltene Firmware basiert auf den original Designempfehlungen von Sigma Designs, die im System Development Kit 4.54 unter *NDA* veröffentlicht wurden. Verglichen mit anderen Standard-Firmware-Designs, wie sie in

den meisten Z-Wave-USB-Sticks und anderer Z-Wave-Host-Schnittstellen-Hardware verwendet wird, wurden an der RaZberry Firmware diverse Erweiterungen und Verbesserungen vorgenommen. [50]

- Die **Controllersoftware** Z-Way – Die Firmware kommuniziert mit dem Z-Way Communication Stack über das serielle Interface /dev/ttyAMA0. Der Z-Way Communication Protokoll Stack organisiert und verwaltet alle im Z-Wave-Netzwerk befindlichen Geräte. Die Z-Way API erlaubt es, eigene Softwareapplikationen, ohne Hintergrundwissen des komplizierten Z-Wave Netzwerkaufbaus zu entwickeln. [50]

Schematischer Aufbau:

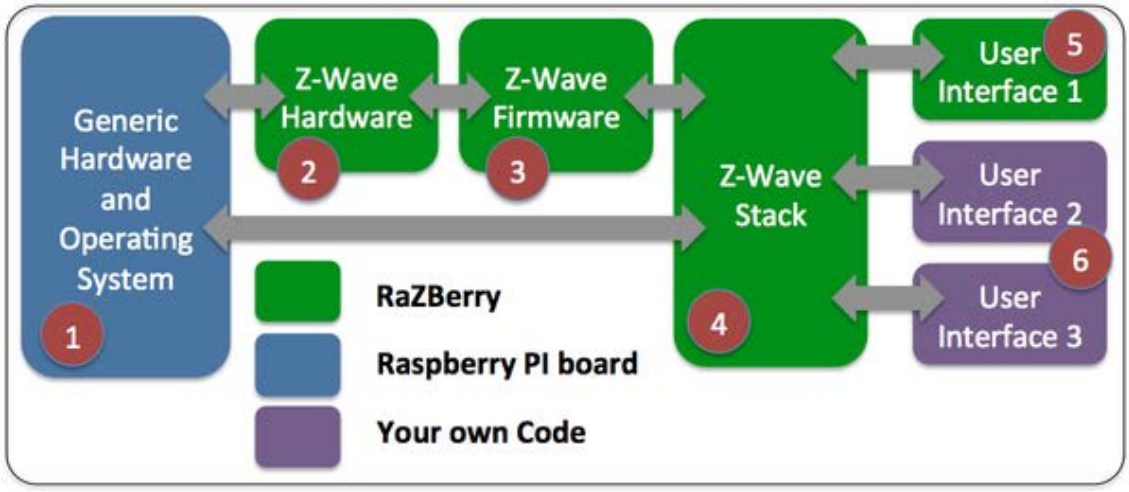


Abbildung 26: RaZberry Schema [48]

1. Raspberry Pi mit installiertem Raspbian OS
2. die RaZberry Daughter Card, wird mit dem Raspberry Pi über die GPIO-Pins verbunden
3. die Z-Wave Low Level Firmware auf dem *Transceiver*
4. Z-Way, der erste zertifizierte Z-Wave Communication Stack, steuert die komplette Z-Wave Netzwerkkommunikation, beinhaltet eine JavaScript-Automatisierungs-Engine und einen Webserver
5. das webbasierte Demo User Interface.
6. die eigene Applikation.

4.1.7 1-Wire-Anbindung

Die zuvor definierte Spezifikation sieht vor, dass das System über eine 1-Wire-Schnittstelle verfügen soll, um den Anschluss diverser digitaler Temperatursensoren, wie den DS18B20 oder DS18S20 zu ermöglichen.

Dazu wurde das Bauteil DS2482S-100 der Firma Maxim Integrated verwendet. Dieses integrierte SMD-Bauelement ist eine sogenannte I^2C-to-1-Wire-Bridge. Das bedeutet, dass dieses Gerät eine bidirektionale Protokollkonvertierung zwischen dem I^2C-Master und dem 1-Wire-Slave durchführt. Alle zeitkritischen Aufgaben auf dem 1-Wire-Bus werden durch dieses Bauteil übernommen. Der DS2482S-100 unterstützt als 1-Wire-Master sowohl Standard Speed als auch Overdrive Speed. Durch die zwei Adresspins können dem Gerät eine von vier möglichen Adressen zugewiesen werden. Somit können bis zu vier von diesen Bauteilen auf einem I^2C-Bus betrieben werden. [51]

Da der 1-Wire-Bus darauf ausgelegt sein soll, durch das komplette Haus gelegt werden zu können, muss die Schaltung zusätzlich gegen eventuell auftretende elektrostatische Aufladung geschützt werden.

Um diesen Schutz zu gewährleisten, wurde das Bauelement DS9503 der Firma Maxim Integrated verwendet. Ein integriertes SMD-Bauteil bestehend aus einer Zenerdiode und zwei 5-Ω-Widerständen zur Isolation der Anode und Kathode. Diese geringen Widerstände sind während der Kommunikation vernachlässigbar, doch stellen sie eine hohe Impedanz in Bezug auf die leitende Diode während eines ESD-Ereignisses dar. Kommt es zu einer Entladung, so absorbiert die Zenerdiode die überschüssige Energie, während die Widerstände die Schaltung isolieren und schützen. Wird dieses Bauteil an I/O Ports verwendet, die schon einen ESD-Schutz besitzen, so können Entladungen oberhalb von 27 kV abgefangen werden. [52]

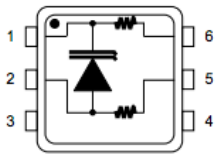


Abbildung 27: DS9503 im TSOC-Package

Nähere Informationen zu den oben genannten Bauteilen können den Datenblättern [51] [52] entnommen werden.

Hinweis: Die Anschlüsse des 1-Wire-Bus wurden auf der Platine zum einen auf Schraubanschlussbuchsen und zum anderen auf eine RJ45-Anschlussbuchse ausgeführt. (Belegung siehe Abbildung 34, Kapitel 4.1.11)

4.1.8 Schaltplan

Anhand der Spezifikation und der verwendeten Bauteile wurde mithilfe der PCB-Design-Software Eagle ein Schaltplan erstellt. Der Schaltplan des ersten Prototypen ist im Anhang A1 zu finden. Der Schaltplan zum Prototypen der zweiten Generation befindet sich im Anhang A2.

4.1.9 Prototypenentwicklung

Bei der Entwicklung des ersten Prototypen wurde ein sogenanntes *„Breadboard“* verwendet. Hierbei handelt es sich um ein Steckbrett zur prototypischen Entwicklung von Schaltungen ohne Lötaufwand. Da dieses *Breadboard* ein Rastermaß von 2,54 mm besitzt, mussten einige Bauteile (unter anderem die in SMD Bauweise), die nicht diesem Rastermaß entsprechen, auf Adapterplatinen gelötet werden. Alle anderen Bauteile, die diesem Maß entsprechen, konnten direkt auf dem Steckbrett platziert werden und nach dem zuvor entwickelten Schaltplan (Anhang A1) mit Verbindungsleitungen versehen werden.

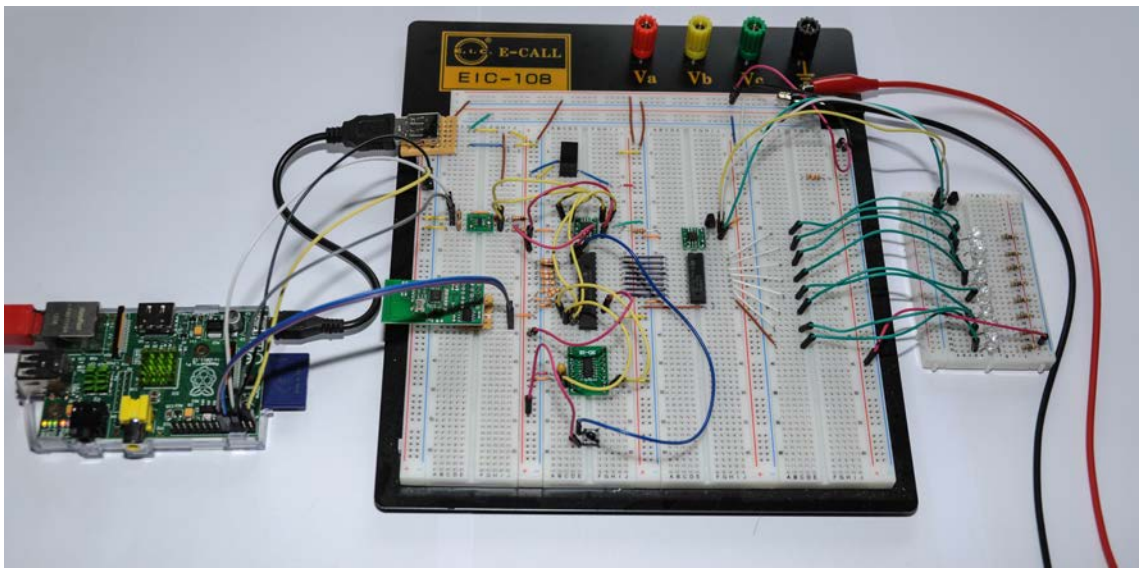


Abbildung 28: Schaltungsaufbau auf einem Breadboard

Anhand dieses Aufbaus konnten die Funktionen des Systems sowie der Schaltplan auf Richtigkeit überprüft und eventuelle Fehler beseitigt werden.

Die weitere Entwicklung des ersten Prototypen fand in erster Linie am Computer statt. Mithilfe der Software *Lochmaster 4.0* der Firma *Abacom* wurde ein Platinenlayout für die in der Spezifikation vorgesehenen 100 x 160 mm große Lochrasterplatine erstellt.

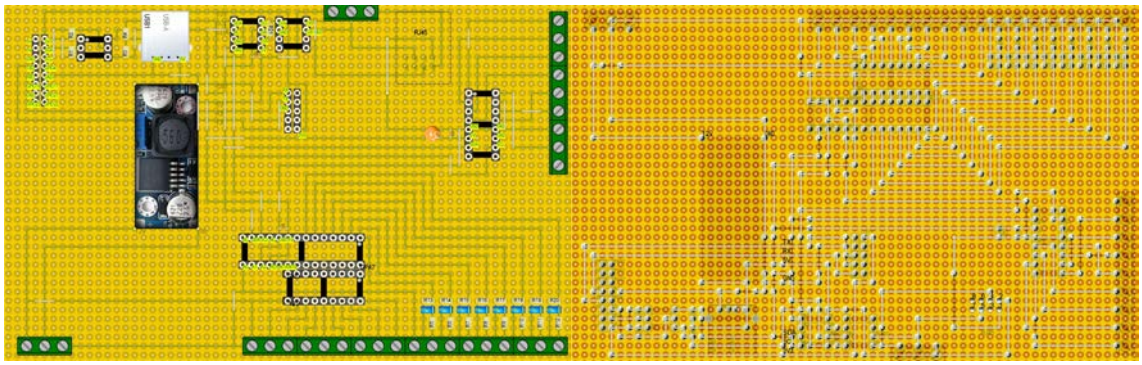


Abbildung 29: Platinenlayout Prototyp 1 – links (Vorderseite), rechts (Rückseite)

Anhand dieses Entwurfs konnten nun Bauteile und Verbindungsleitungen auf der Lochrasterplatine platziert und verlötet werden.

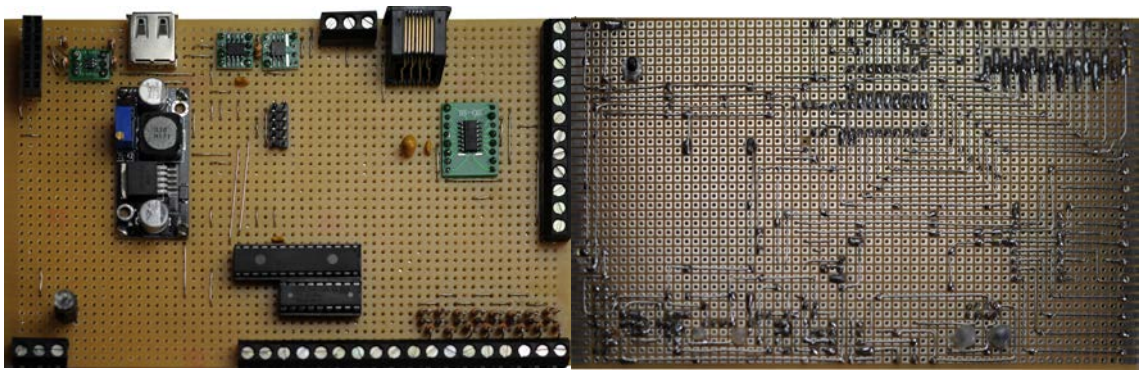


Abbildung 30: Prototyp 1 - Alle Bauteile auf Lochrasterplatine verlötet

Daraufhin wurde der Protoyp nach dem im Kaptitel 3.7 erwähnten Testkonzept überprüft.

4.1.10 Platinenlayout des zweiten Protoypen

Anhand der Spezifikation des 2. Prototypen sowie der verwendeten Bauteile, wurde mithilfe der PCB-Design-Software Eagle ein Platinenlayout angefertigt.

Bei der Verlegung der Leiterbahnen wurde darauf geachtet, dass, sofern möglich keine rechtwinkligen Abknickungen vorkommen, da diese zu einer enormen Stromdichteerhöhung in den Leiterbahnecken führen. [53]

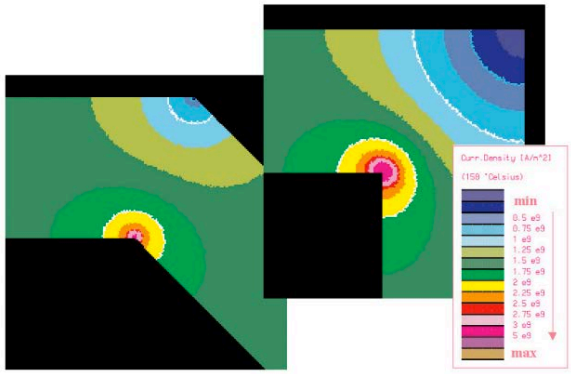


Abbildung 31: Stromdichteerhöhung in Leiterbahnecken, insbesondere bei rechtwinkligen Abknickungen [53]

Ebenso wurde versucht, so wenig wie möglich mit Durchkontaktierungen zu arbeiten, da jede dieser Kontaktierungen eine Schwachstelle darstellen kann.

Die Massekontakte wurden durch Ausfüllen der freien Flächen als Masseflächen verbunden, dieses sorgt für eine niederohmige Verbindung.

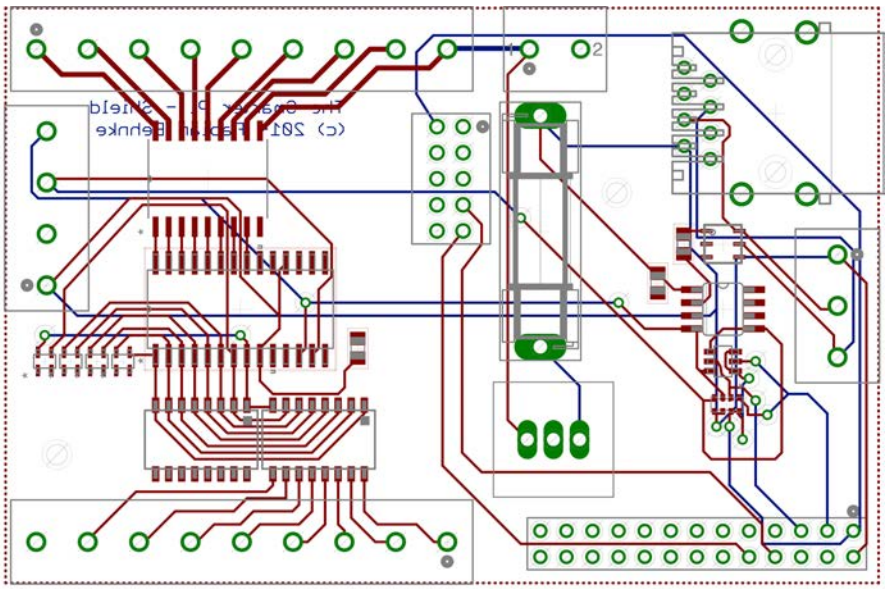


Abbildung 32: Platinenlayout Prototyp 2 – rot (Vorderseite), blau (Rückseite), Masseflächen ausgeblendet

4.1.11 Fertigstellung des zweiten Prototypen

Anhand des erstellen Plartinenlayouts konnte nun die Leiterplatte in die Fertigung gegeben werden. Das Ergebnis der produzierten Platine kann der Abbildung 33 entnommen werden.

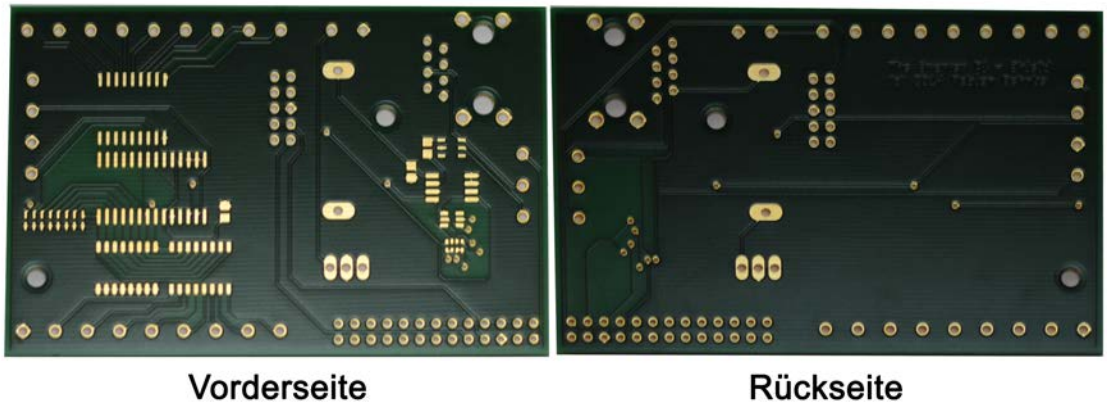


Abbildung 33: Unbestückte Platine

Die Platine wurde nun von Hand mit den entsprechenden Bauteilen bestückt und verlötet. Abbildung 34 zeigt die fertig bestückte Platine mit allen Anschlussmöglichkeiten und deren Belegungen.

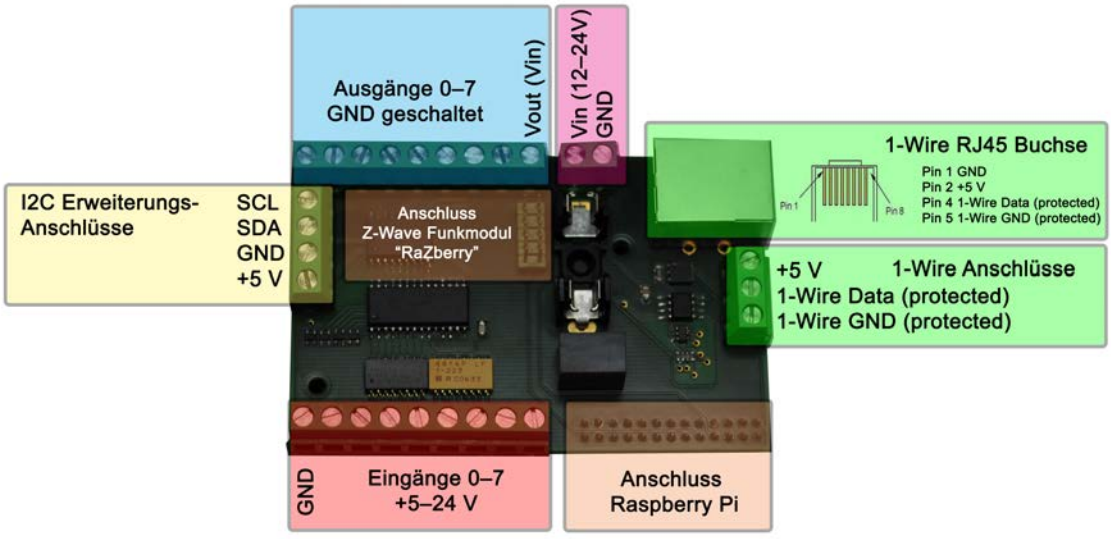


Abbildung 34: Fertig bestückte Platine mit Anschlussbelegung

4.2 Software

4.2.1 Angewandtes Vorgehensmodell

Der dieser Arbeit bezügliche Softwareentwicklungsprozess wurde nach dem V-Modell[6] abgewickelt.

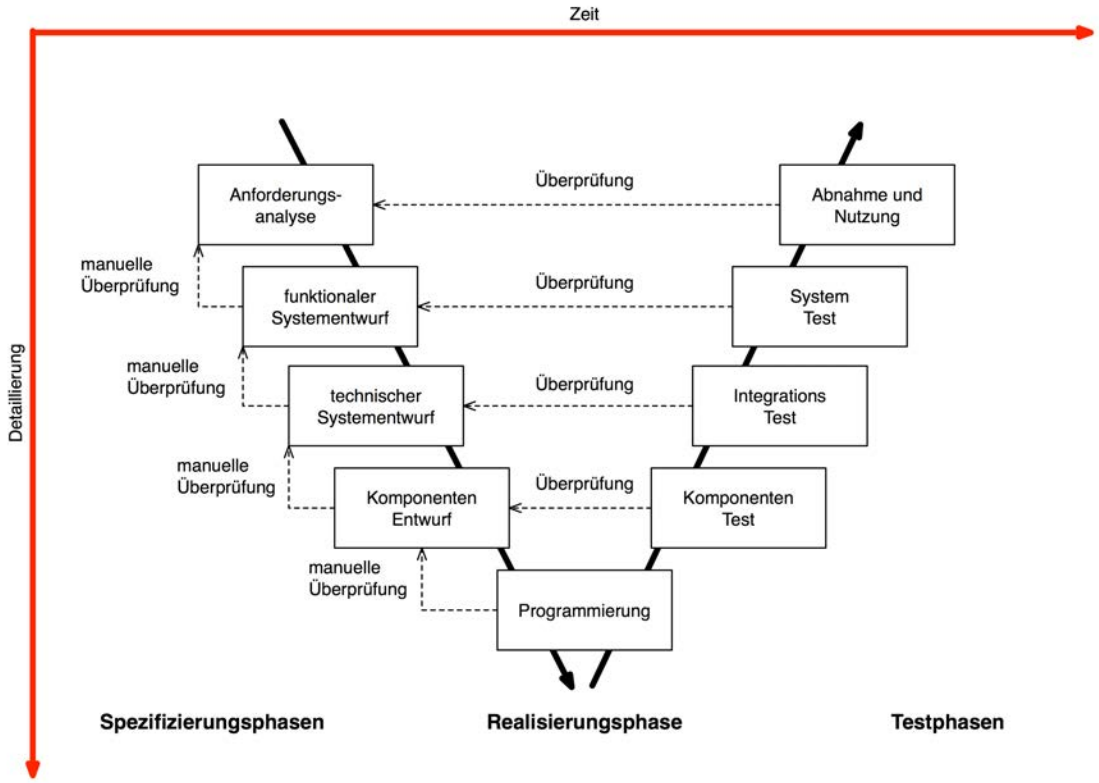


Abbildung 35: verwendetes Vorgehensmodell - V-Modell

Die vorherigen Spezifizierungsphasen Anforderungsanalyse (Kapitel 3.1.3), funktionaler Systementwurf, technischer Systementwurf (Kapitel 3.2 - 3.3) führten zur endgültigen Realisierungsphase. Während der Spezifizierungsphasen wurden die einzelnen Punkte mehrfach verifiziert, um Fehler im Vorhinein ausschließen zu können. Nach der Realisierungsphase folgten mehrere Testphasen, um die vorher spezifizierten Punkte zu überprüfen. Sind während dieser Phasen Fehler entdeckt worden, wurden diese beseitigt und während der Testphasen erneut überprüft.

[6] Das V-Modell ist ein Vorgehensmodell in der Softwareentwicklung, bei dem der Entwicklungsprozess in Phasen aufgeteilt ist. Es basiert auf dem Wasserfallmodell und erweitert es um mehrere Testphasen zur Qualitätssicherung.

4.2.2 Verwendete Frameworks

Im Zuge dieser Arbeit wurden verschiedenste Software Frameworks verwendet. In diesem Kapitel werden die wichtigsten Frameworks näher erläutert.

jQuery Mobile

jQuery Mobile ist ein touch-optimiertes Open-Source-Framework (MIT) zum Erstellen von HTML5-basierten Benutzeroberflächen. Mithilfe dieses Frameworks lassen sich sehr einfach responsive Webseiten und Webapplikationen erstellen, die sowohl auf mobilen Geräten als auch auf Desktop-Geräten gleichermaßen gut bedienbar sind. jQuery Mobile beinhaltet eine Vielzahl grafischer Elemente, wie z.B. Schalter, Datepicker, Icons und viele mehr. Jedes dieser Elemente lässt sich per JavaScript ansprechen, um seinen Zustand von außen verändern zu können. Mit jQuery Mobile erstellte Webapplikationen laufen auf einer Vielzahl derzeit auf dem Markt befindlicher Browser wie z.B. Chrome, Firefox, Safari und Internet Explorer (sowohl in der mobilen Variante als auch in der Desktop-Variante).

Verwendung in dieser Arbeit:

jQuery Mobile wurde verwendet, um die grafische Benutzeroberfläche (Webinterface) der Haussteuerung zu gestalten. Die gute Browserunterstützung stellt sicher, dass das System später auf einer Vielzahl von Geräten, unabhängig vom Betriebssystem, genutzt werden kann.

Express

Express ist ein leichtgewichtiges und flexibles Open-Source (MIT) Framework zur Erstellung von Webapplikationen mithilfe von Node.js. Es bietet eine Vielzahl an Funktionen und eine sehr gut dokumentierte API.

Verwendung in dieser Arbeit:

Express wird verwendet, um die (dynamisch erstellten) Webseiten an den Browser auszuliefern. Des Weiteren ist es zuständig für Weiterleitungen zu weiterführenden Seiten (z.B. nach dem Senden eines Formulars) und die korrekte Abwicklung von Sessions.

EJS

EJS ist ein Open-Source-JavaScript Rendering-Framework (MIT) zur Erstellung von Templates. Es ermöglicht das Rendern von HTML-Seiten aus einem JavaScript basiertem Web-Template, ohne den Server kontaktieren zu müssen.

Verwendung in dieser Arbeit:

EJS wird verwendet um JavaScript-basiertem Web-Templates auf dem Server zu rendern (ähnlich dem Konzept von PHP) und durch Express an den Browser auszuliefern.

4.2.3 Verwendete Node.js Module

Bei dieser Arbeit wurden diverse Node.js-Module verwendet. Diese Module greifen teilweise auf weitere Submodules (Unter-Module) zurück. Diese Submodules werden hier nicht weiter betrachtet.

Coffee-Script

Open-Source (MIT) Modul, kompiliert *CoffeScript*[7] in JavaScript.

Connect-Mongo

Open-Source (MIT) Modul, ermöglicht MongoDB als SessionStore zu verwenden.

Cron

Open-Source (MIT) Modul, ermöglicht die Erstellung von Cron-Jobs für Node.js code. Die Syntax ist kompatibel mit dem von Linux bekannten Cron-Daemon.

Express

Siehe 4.2.2

EJS

Siehe 4.2.2

I2C

Open-Source (BSD) Modul, dass native Bindings an die I^2C-Schnittstelle des Raspberry Pi (funktioniert auch mit dem BeagleBone) enthält und so ermöglicht, die I^2C Schnittstelle direkt in JavaScript anzusprechen.

[7] CoffeScript ist eine Programmiersprache, angelehnt an Ruby, Python und Haskell um die Lesbarkeit und Prägnanz von JavaScript zu verbessern. Typischerweise weisen in CoffeScript geschriebene Programme 30% weniger Programmzeilen auf als in JavaScript. [31]

Mjpeg-Proxy

Open-Source (MIT) Modul, ist ein Proxy für MJPEG-Video-Streams. Dieser ermöglicht das Weiterreichen eines MJPEG-Streams an eine Vielzahl von Clients.

Mongoose

Open-Source (MIT) Modul, ist ein Object Modeling Tool (OMD) für MongoDB, es ermöglicht einen objektbasieren Zugriff auf die in der Datenbank enthaltenen Daten.

Needle

Open-Source (MIT) Modul, ist ein HTTP-Client für Node.js mit einer Auswahl der meistgenutzten Funktionen zum Aufbau von HTTP-Verbindungen.

Socket.IO

Open-Source (MIT) Modul, es ermöglicht durch Technologien wie Websockets, *Adobe Flash* Socket, AJAX long polling, AJAX multipart streaming, Forever Iframe und JSONP Polling eine direkte Kommunikation per JavaScript mit dem Browser.

Socket.IO-Session

Open-Source (MIT) Modul, erweitert das Modul Socket.IO in Verbindung mit Express, um die Funktionalität Sessions zu verwenden.

Underscore

Open-Source (MIT) Modul, beinhaltet eine Vielzahl hilfreicher Funktionen für den Umgang mit Array, Collections, etc.

4.2.4 Im Zuge dieser Arbeit entwickelte Module

Alle im Zuge dieser Arbeit entwickelten Module wurden wegen der besseren Lesbarkeit in CoffeScript programmiert. Im nachfolgenden Abschnitt werden diese Module kurz erklärt und durch Anwendungsbeispiele verdeutlicht.

HttpPseudoDevice

Dieses Modul ermöglicht die Einbindung von Geräten oder Diensten (z.B. zur Versendung von Push Notifikationen) über eine HTTP Rest API.

Anwendungsbeispiel:

```
/* Modul einbinden */
var HttpPseudo = require('httpPseudoDevice');
/* Neue Instanz eines Bausteins erstellen */
var httpPseudo = new HttpPseudo();
/* Das data-Objekt setzt sich wie folgt zusammen:*/
var data = {
            url: "http://meineurl.de/api",
            reqmode: "post", //HTTP GET (get) oder HTTP POST (post)
            username: "meinBenutzername",
            password: "meinPasswort",
            args: [{argument: 'user', value: '12345'}, //Argumente
                  { argument: 'token', value: '12345'}, ...]
}
/* Daten absenden */
httpPseudo.fire(data, callback)
```

MCP23017

Dieses Modul ermöglicht die objektbasierte Ansteuerung des MCP23017 16 bit GPIO Extenders. Intern nutzt es das Submodule I2C, um den Baustein direkt über die I^2C-Schnittstelle ansprechen zu können. Digitale Eingänge werden alle 10 ms abgefragt, Zustandsänderungen werden per Callback an eine zuvor definierte Funktion übergeben.

Anwendungsbeispiel:

```
/* Modul einbinden */
var MCP23017 = require('mcp23017');
/* I2C Adresse des Bausteins */
var address = 0x20;
/* Neue Instanz eines Bausteins erstellen */
/* Argument 1 - I2C Adresse */
/* Argument 2 - Linux Device File */
var mcp = new MCP23017(address, '/dev/i2c-1');
/* Ausgänge Schalten */
mcp.setGpioAPinValue(0,1); //GPIO A Pin 0 auf HIGH schalten
mcp.setGpioAPinValue(0,0); //GPIO A Pin 0 auf LOW schalten
/* Eingänge Abfragen */
mcp.setCallback(meineCallbackFunktion) //Callback setzten
console.log(mcp.getGpioBPinValue(0)); //manuelle Abfrage GPIO B Pin 0
```

MCP3424

Dieses Modul wurde für den ersten Prototypen entwickelt und kam später nicht zum Einsatz. Es ermöglicht das Auslesen des MCP3424 Delta-Sigma-Wandlers über die I^2C-Schnittstelle. Wie auch beim MCP23017-Modul wird hier zur Kommunikation intern das im Kapitel 4.2.3 vorgestellte I2C-Modul verwendet.

Anwendungsbeispiel:

```
/* Modul einbinden */
var MCP3424 = require('mcp3424');
/* I2C Adresse des Bausteins */
var address = 0x68;
/* Verstärkung und Auflösung einstellen */
var gain = 0; //{0,1,2,3} repräsentiert {x1,x2,x4,x8}
var resolution = 3; //{0,1,2,3} repräsentiert {12,14,16,18} bits
/* Neue Instanz eines Bausteins erstellen */
/* Argument 1 - I2C Adresse */
/* Argument 2 - Verstärkung */
/* Argument 2 - Auflösung */
/* Argument 4 - Linux Device File */
var mcp = new MCP3424(address, gain, resolution, '/dev/i2c-1');
/* Erste Abfrage der Spanunnungswerte - Ausgabe auf die Konsole */
setTimeout(function(){
  console.log(mcp.getMv(0)); //channel 0
  console.log(mcp.getMv(3)); //channel 3
}, 2000); //Umsetzung benötigt Zeit...(kleinere Auflösung -> schneller)
```

Owfs_temp

Dieses Modul greift auf das von OWFS[8] bereitgestellte 1-Wire-Dateisystem zurück, um 1-Wire-Temperatursensoren aufzufinden und Temperaturwerte kontinuierlich auszulesen. Sensordaten wie Sensor-ID, Sensor-Typ und Temperatur werden in einem voreingestellten Intervall an eine Callback-Funktion übergeben.

Anwendungsbeispiel:

```
/* Modul einbinden */
var OWFS = require('owfs_temp');
/* Neue Instanz erzeugen */
/* Argument 1 - OWFS Mountpount */
/* Argument 2 - Intervall ms */
/* Argument 3 - Callback-Funktion */
var owfs = new OWFS('/mnt/1wire/', intervall, callback(data));
```

[8] OWFS ist eine unter GNU GPLv2 veröffentlichtes Open-Source Software, die es getreu nach dem Motto „Everything is a file“ ermöglicht, 1-Wire Geräte über das Linux Dateisystem anzusprechen und auszulesen. [32]

RuleHelper

Dieses Modul stellt diverse Funktionen zur Verfügung, die zum Ausführen von Regeln benötigt werden. Es greift auf die Datenbank zurück, um z.B. aus einer Geräte-ID die dazugehörige Regel ausfindig zu machen und daraufhin auszuführen.

ZWave

Dieses Modul ermöglicht im aktuellen Entwicklungsstadium den objektbasierten Zugriff auf Z-Wave-Geräte, wie Schalter, Dimmer, Rauchmelder und Strommessgeräte. Es kommuniziert über die JSON basierte HTTP API von Z-Way mit dem RaZberry-Funkmodul. Zustandsänderungen werden kontinuierlich abgefragt und einer Callback-Funktion übergeben.

Anwendungsbeispiel:

```
/* Modul einbinden */
var ZWAVE = require('zwave');
/* Neue Instanz erzeugen */
/* Argument 1 - IP oder Hostname des Z-WAY Servers */
/* Argument 2 - Port des Z-WAY Servers */
/* Callback Funktion */
var zwave = new ZWAVE('localhost',8083, callback(data));
/* Neues Z-Wave Gerät hinzufügen */
zwave.includeDevice(true)
/* Schalter mit der Node-ID 1 einschalten */
zwave.setSwitchValue(1, true)
/* Dimmer mit der Node-ID 2 auf 50% Dimmen */
zwave.setSwitchMultilevelValue(2, 50);
/* Batterystatus von Gerät mit der Node-ID 3 abfragen */
zwave.getLastBatteryValue(3)
/* Z-Wave Gerät auf dem System entfernen */
zwave.excludeDevice(true)
```

4.2.5 Projektstruktur

Zur besseren Übersicht des Software-Projekts wurden die einzelnen Programmteile ausgegliedert und strukturiert in Ordnern abgelegt. Anhand dieser Struktur (Abbildung 34) lassen sich die verschiedenen Programmteile im Programmcode wesentlich besser nachvollziehen.

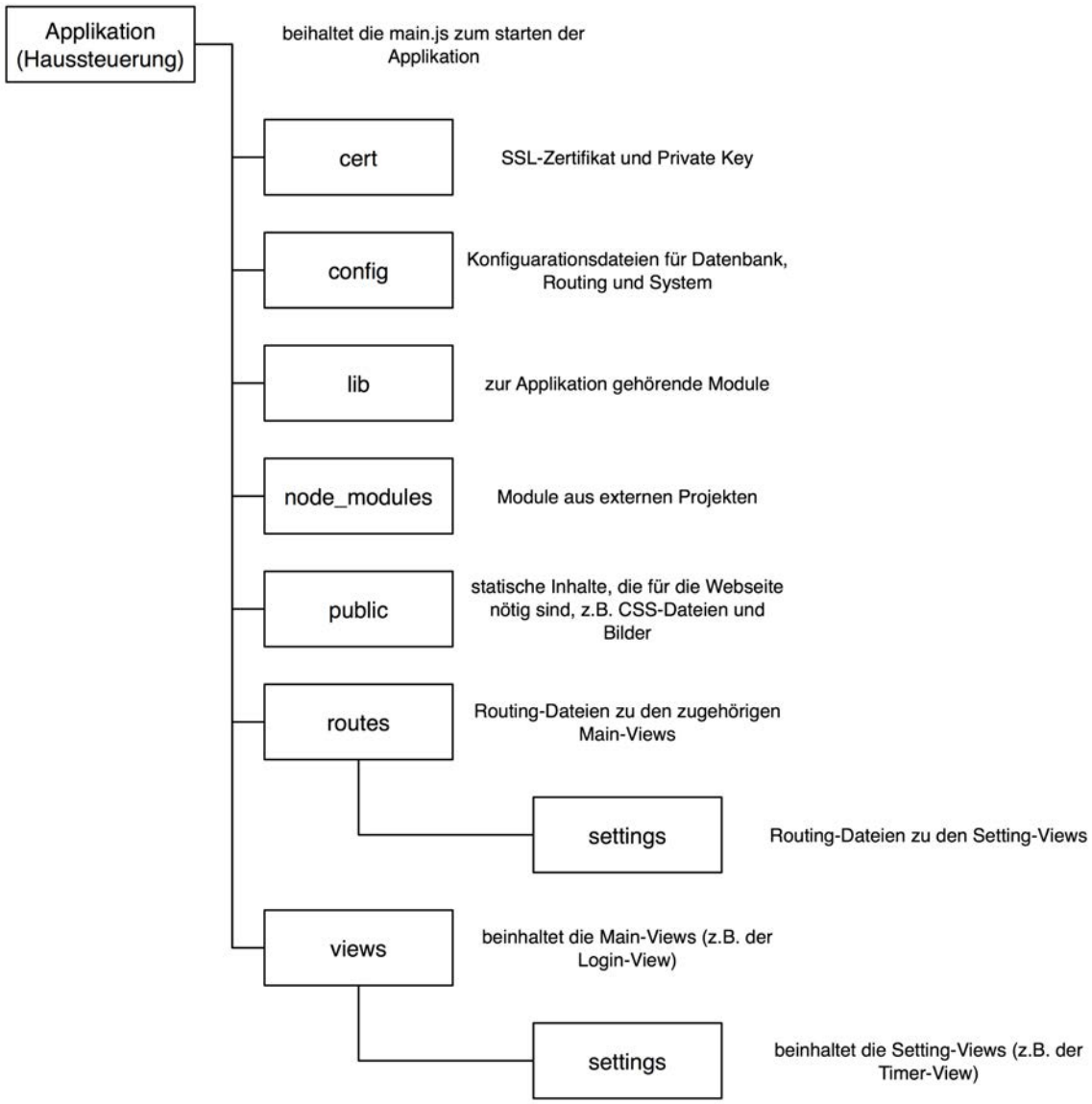


Abbildung 36: Projekt-Struktur

4.2.6 Source-Code Dokumentation mit YUIDoc

Der Source-Code dieser Node.JS Anwendung wurden so programmiert, dass er ohne Kommentare auskommen würde. Dazu wurden Funktionsnamen verwendet, die sofort zu verstehen geben sollen, was diese Funktionen tun. Dennoch wurde der Programmcode zum besseren Verständnis komplett kommentiert.

Die zu der Anwendung gehörigen Module wurden darüber hinaus in einer speziellen Kommentarsyntax geschrieben, aus der sich mithilfe der Open-Source-Software YUIDoc[9], eine komplett webbasierte Dokumentation erstellen lässt. Die YUIDoc Kommentarsyntax bietet dabei eine Vielzahl an Schlagwörtern, um den Programmcode mit verständlichen Kommentaren zu versehen.

Beispiel
Kommentar einer JavaScript Methode:

```
/**
* Meine Methode – sie macht dies und das
*
* @method Methodenname
* @param {String} abc Der Paramater abc erwartet einen String
* @param {Object} config Erwartet ein Konfigurationsobjekt
* @return {Boolean} Gibt true zurück, wenn alles geklappt hat
*/
```

YUIDoc durchsucht das Projektverzeichnis nach Dateien, die mit dieser Kommentarsyntax kommentiert wurden und erstellt daraufhin eine übersichtliche und komplett durchsuchbare Dokumentation.

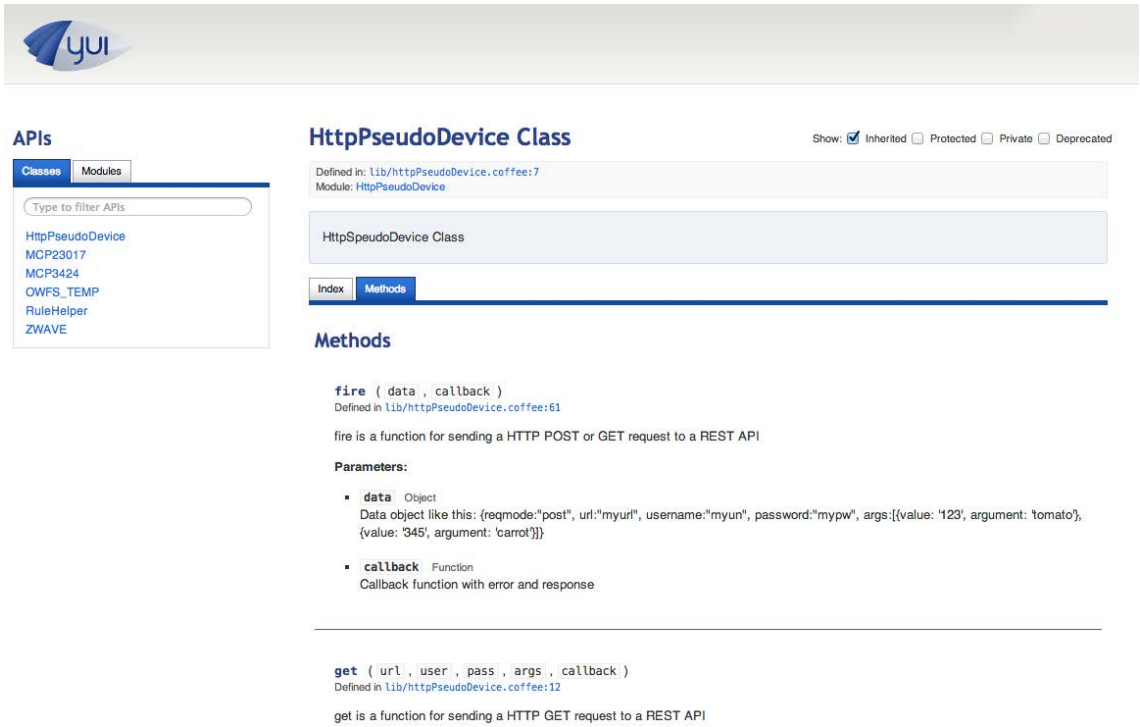


Abbildung 37: Screenshot der YUIDoc Source-Code-Dokumentation

[9] YUIDoc Projekt Webseite - http://yui.github.io/yuidoc/

4.2.7 Dynamische Webseitengenerierung mit „Express“ und „EJS“

Das Zusammenspiel der Module Express und EJS ermöglicht das dynamische Generieren von Webseiten nach einem ähnlichen Konzept wie bei PHP.

Das nachfolgende Beispiel soll das verwendete Konzept näher verdeutlichen.

Aufruf der Seite „Login“

Wird im Webbrowser die Loginseite (http://node.js-server/login) angefordert, so springt der Node.JS Server an die folgende Stelle im Code:

```
app.get('/login', login);
```

Diese Funktion bewirkt den Aufruf der vorher definierten Route-Funktion:

```
login = function(req, res){
    User.count({}, function( err, count){
        if(count == 0){
            res.redirect('/1stStart');
        }else{
            res.render('login', {title: "Login", auth:"ok"});
        }
    })
};
```

Doch was tut diese Route-Funktion?

Zuerst wird die Anzahl der am System registrierter Benutzer gezählt. Ist die Anzahl 0, bedeutet es, dass dieses System zum ersten Mal aufgerufen wurde und der Browser wird aufgefordert, zur Seite „1stStart“ (http://node.js-server/1stStart) umzuleiten. Ist das Ergebnis ungleich 0 wird die Loginseite von EJS gerendert und durch das Modul Express an den Webbrowser ausgeliefert.

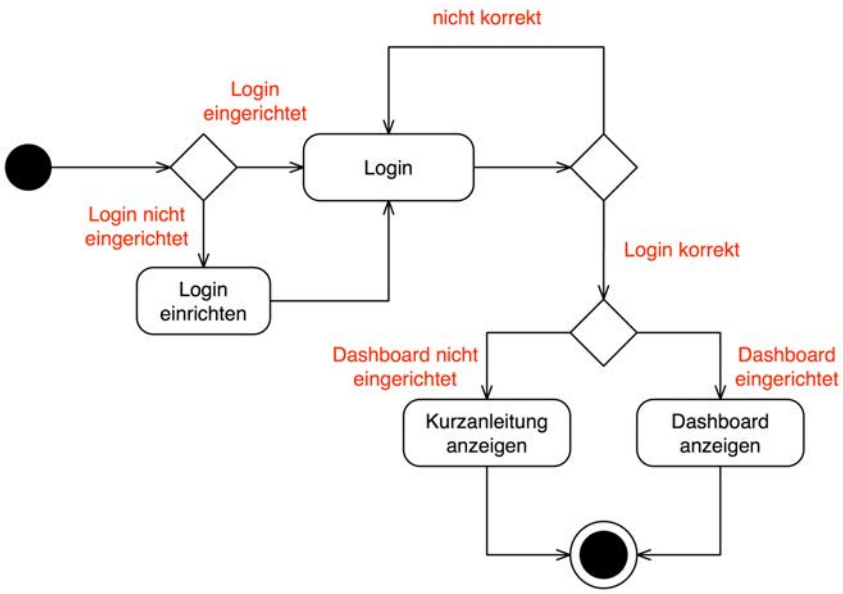


Abbildung 38: Programmablaufplan – Login

Dabei wurden über die Funktion **render()** folgende Parameter, bzw. JavaScript Objekte an EJS übergegeben: Zum einen der String „login“ sowie ein JavaScript Objekt, bestehend aus zwei Strings.

Der erste Parameter ist der Name des Views, der zweite können diverse JavaScript Objekte oder Referenzen auf diese sein, die beim Rendern der Webseite benötigt werden.

EJS sucht daraufhin im Ordner „views“ nach der entsprechenden View-Datei mit dem Namen „login.ejs“.

Diese View- oder auch Template-Dateien werden in erster Linie in HTML programmiert. JavaScript-Code, der mit <% %> umschlossen ist, wird ausgeführt. Wird dieser Code jedoch mit <%= %> umschlossen, so wird das Ergebnis dem HTML-Code an der demensprechenden Stelle angefügt. Die zuvor durch die Funktion **render()** übergebenen Objekte stehen im JavaScript-Code ebenfalls zur Verfügung. Des Weiteren ist es möglich, durch <% include Viewdatei %> den HTML-Code durch weitere View-Dateien zu erweitern.

Vereinfachtes Beispiel der Datei login.ejs

```
<% include header %>
<body>
   <h1><%= title %></h1>
   <% if (auth == "nok"){%>
      <!—HTML Element wird angezeigt, wenn JavaScript Bedingung erfüllt-->
      <p>Username or password you entered is wrong!</p>
  <%}%>
</body>
<% include footer %>
```

Bei folgenden Randbedingung

```
{title: "Login", auth:"ok"}
```

header.ejs

```
<html>
   <head></head>
```

footer.ejs

```
</html>
```

wird die login.ejs von EJS zu folgendem HTML-Code gerendert und durch Express an den Browser ausgeliefert.

```
<html>
   <head></head>
   <body>
      <h1>Login<h1>
   </body>
</html>
```

4.2.8 Grafische Benutzeroberfläche

Bei der grafischen und technischen Gestaltung der Benutzeroberfläche (GUI) mittels jQuery Mobile (Kapitel 4.2.2) wurde darauf geachtet, dass die Bedienung sowohl auf Desktop-Geräten als auch auf mobilen Geräten gleichermaßen gut aussieht und ebenso gut funktioniert.

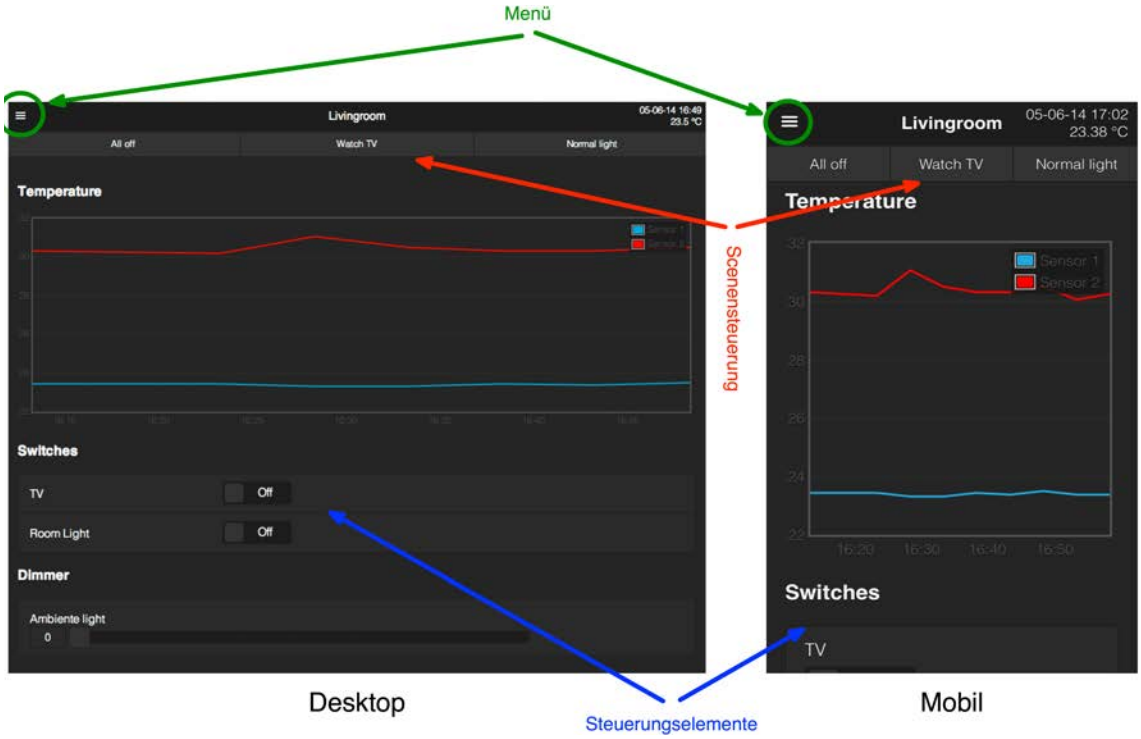


Abbildung 39: Benutzeroberfläche Desktop und Mobil

Benutzeroberfläche

In der Mitte am schwarzen oberen Rand befindet sich der Titel der aktuellen Seite. Oben rechts, ebenfalls auf dem schwarzen Rand, befinden sich aktuelle Uhrzeit sowie, falls vorhanden, Temperatur des dargestellten Raumes. Unterhalb der schwarzen Leiste werden eventuell konfigurierte Szenen des dargestellten Raumes angezeigt. Mit einem Klick oder Fingertab auf eine dieser Szenen, wird diese ausgeführt. Sind einem Raum ein oder mehrere Temperatursensoren zugewiesen, werden diese in Form einer Temperaturkurve dargestellt.

Zur Steuerung der einzelnen Geräte wurden graphische Bedienelemente (Flip-Switches und Slider) verwendet, die ebenfalls auf die Benutzung mit dem Finger (Touch) oder per Maus optimiert sind.

Menü

Das Menü wird über das Symbol (drei waagerechte Striche) am oberen linken Rand der Applikation aufgerufen. Es schiebt die restliche Webseite über den rechten Bildschirmrand hinaus und erscheint im linken Bereich des Bildschirms.

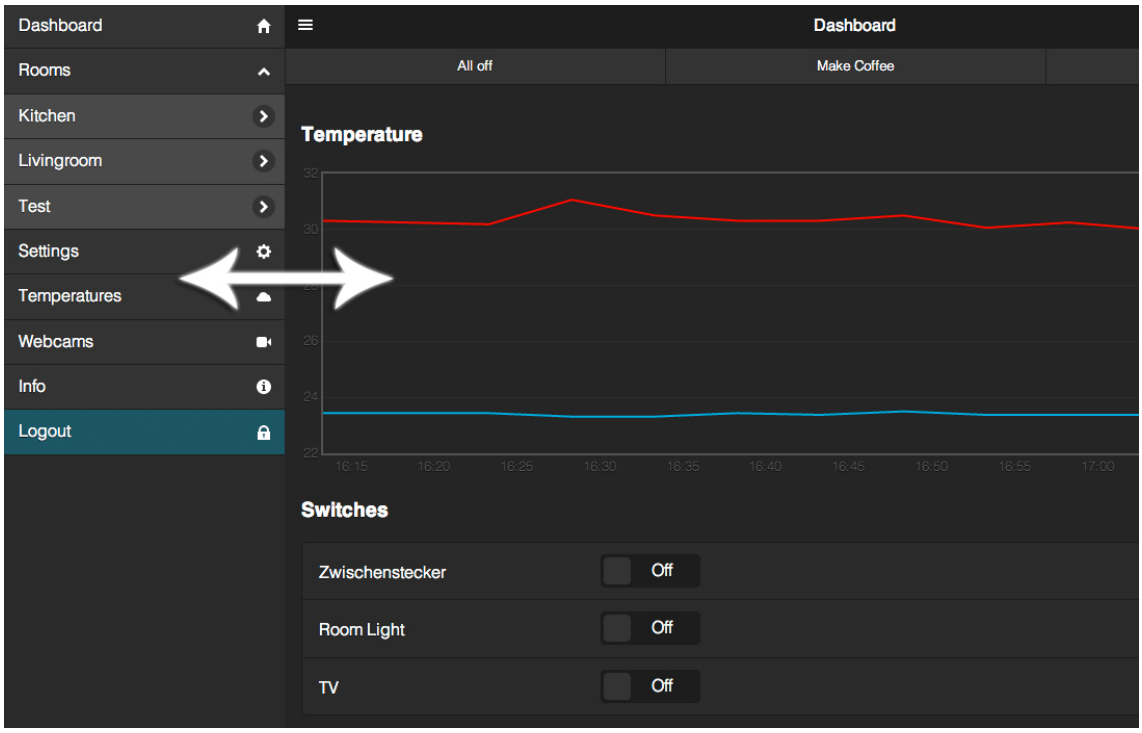


Abbildung 40: Benutzeroberfläche - Menü

Grafische Elemente sowie Icons sollen das Erscheinungsbild auflockern und dem Benutzer ein schnelleres Verständnis über den jeweiligen Menüpunkt ermöglichen.

4.2.9 Die wichtigsten Funktionen im Überblick

In diesem Kapitel werden die wichtigsten Funktionen der entwickelten Haussteuerung kurz erklärt.

Unterstützte Geräte und Dienste

Über die Steuerung können eine Vielzahl an Geräten und Diensten angesteuert werden. Im derzeitigem Entwicklungsstadium sind dies die 8 internen (auf der Platine befindlichen) Ein- und Ausgänge, Geräte und Dienste die über eine HTTP REST API verfügen, sowie Z-Wave Geräte der folgenden Geräteklassen:

- schaltbare Steckdosen (optional: mit integrierter Strommessung)
- Dimmer
- Rauchmelder

Des Weiteren werden Temperatursensoren, die über eine 1-Wire-Schnittstelle verfügen, unterstützt.

Szenen

Es können diverse Szenen angelegt werden, über die ein oder mehrere Geräte in definierte Zustände gesetzt werden können.

Beispiel-Scene: (Wohnzimmer - Alles Aus)

Deckenlicht aus
Ambiente-Beleuchtung 0%
TV aus

Beispiel-Scene: (Wohnzimmer - Fernsehen)

Deckenlicht aus
Ambiente-Beleuchtung 25%
TV ein

Zu diesen Szenen lassen sich sogenannte Webhook-URLs generieren, über die die jeweilige Szene aufgerufen werden kann. So ist es Drittanbieter-Applikationen ebenfalls möglich, auf diese Szenen per URL zuzugreifen. Ein Anwendungsbeispiel wäre eine Applikation zur Anwesenheitserkennung - verlässt man das Haus, so werden alle Geräte im Haus abgeschaltet.

Regeln und Benachrichtigungen

Es lassen sich Regeln anlegen, die die Zustände der einzelnen Geräte betreffen. Löst z.B. ein ans System angemeldeter Rauchmelder aus, so kann hier definiert werden, was daraufhin passieren soll. In der jetzigen Ausbaustufe der Software kann jeweils EIN Gerät auch EINE Szene auslösen. Wurde in dieser Szene ein Push-Dienst hinterlegt, der über eine HTTP REST API angesprochen wird, so lassen sich z.B. Push-Nachrichten an das Mobiltelefon senden, sofern der Rauchmelder (oder ein anderes Gerät) ausgelöst hat.

Räume

In der Software lassen sich virtuelle Räume anlegen. Diesen Räumen können dann Z-Wave-Geräte wie schaltbare Steckdosen, Dimmer oder die auf der Platine befindlichen Ausgänge zugewiesen werden. Des Weiteren ist es möglich, einem Raum ein oder mehrere Temperatursensoren zuzuweisen. Diese werden als Temperaturkurve auf der Seite des jeweiligen Raumes dargestellt.

Timer

Es lassen sich wiederkehrende Timer anlegen, die eine zuvor definierte Szene ausführen. Der Benutzer kann auswählen, an welchen Wochentagen und zu welcher Uhrzeit dieser Timer aktiv werden soll.

Webcam

Zur Raum- und Hofüberwachung können in das System Netzwerkkameras, die einen MJPG-Stream[10] liefern, integriert werden. Eine Vielzahl dieser Kameras bietet keine Möglichkeit, diesen Stream über eine verschlüsselte Verbindung auszuliefern. Daher wird dieser Stream über einen in das System integrierten Proxy geleitet und mit über dieselbe (verschlüsselte) Verbindung ausgeliefert, wie das Webinterface.

4.2.10 Konzept zur Generierung von Zufalls-URLs

Der Computer ist ein Gerät, das deterministisch aus den gleichen Eingaben immer dieselben Ausgaben erzeugt. Da die meisten dieser Computer, so auch der Raspberry Pi, über keine externe Quelle (wie z.B. ein Mikrofon) verfügt, über die man Zufall erzeugen könnte, bleibt den meisten Anwendungen nichts anderes übrig, als Zufallszahlen aus der Uhrzeit zu berechnen.

Bei der Entwicklung dieser Anwendung wurde ein anderes Konzept verwendet.

Zufall in Kombination aus Uhrzeit und Maus- bzw. Fingerbewegung.

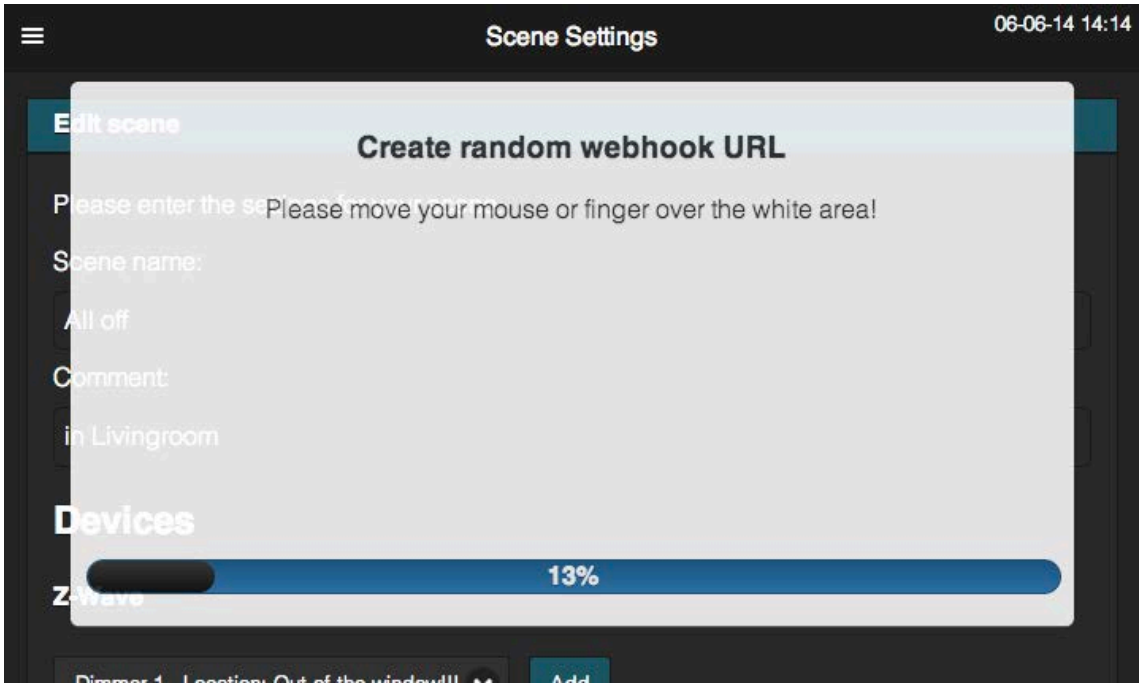


Abbildung 41: Zufalls-URL generieren

Der Benutzer fährt über die weiße Fläche (Abbildung 41) mit einer zufälligen Bewegung der Maus oder seines Fingers. Dabei werden alle 20 ms die X und Y Koordinaten innerhalb dieser Fläche aufgenommen. Diese werden zusammen

[10] MJPG-Stream – Motion JPG Stream ist ein Videostream, bestehend aus komprimierten JPEG Einzelbildern

mit der Uhrzeit hintereinander weg in einen String geschrieben und daraufhin ein SHA256-Hashwert dieses Strings gebildet. Der entstandene aus 64 Buchstaben und Zahlen bestehende Hexadezimal-String wird dann als Zufalls-URL zur Ausführung einer Szene durch Drittanbieter-Anwendungen verwendet. Der 256 bit lange Hashwert hat insgesamt 2^{256} = 1,158 x 10^{77} Möglichkeiten. Will ein Angreifer all diese Möglichkeiten durch Brute-Forcing ausprobieren, so würde er (davon Ausgegangen der Raspberry Pi schafft 10 Anfragen/1 ms) ca. 3,67 x 10^{65} Jahre benötigen.

4.2.11 Sichere Datenübertragung per SSL

Um einen unberechtigten Zugriff auf das Webinterface durch Dritte auszuschließen, wurde das Verschlüsselungsverfahren SSL (Secure Socket Layer) verwendet.

> *„Mit dem SSL-Protokoll steht für Netzwerkanwendungen ein Sicherheitsprotokoll zur sicheren Datenübertragung über ein unsicheres Netz zur Verfügung. SSL stellt Vertraulichkeit, Authentizität und Integrität in der Transportschicht bereit und baut dabei auf ein zuverlässiges Transportprotokoll wie TCP.“ [54 S. 313]*

Verbindungsaufbau einer SSL-verschlüsselten Verbindung mit dem Webinterface

Wird das Webinterface über ein SSL-verschlüsselte Verbindung durch den Browser aufgerufen, so starten Client (Browser) und Server (Node.js) zur gegenseitigen Begrüßung ein Handshake-Verfahren, das entsprechend dem *TLS Handshake Protocol* abgewickelt wird. Der Client sendet dem Server eine Begrüßungsformel *ClientHello*, daraufhin antwortet der Server mit einem *ServerHello*. Beide Verbindungspartner senden sich gegenseitig 28 Byte an zufällig erzeugten Daten. Diese werden später für die Berechnung des *Master-Secrets* benötigt. Daraufhin einigen sich beide auf eine für die weitere Kommunikation zu verwendende Protokollversion und eine sogenannte *Cipher Suite*.

> *„Die Cipher Suite ist ein Tripel aus Schlüsselaustauschverfahren, Verschlüsselungsalgorithmus und Message Authentication Code. Der Client offeriert dabei die von ihm unterstützten Cipher Suiten und der Server wählt daraus entsprechend seiner eigenen Fähigkeiten die bestmögliche Kombination aus. Sollten sich die beiden Seiten nicht auf eine Cipher Suite einigen können, bricht das Handshake mit einem Fehler ab.“ [54 S. 314]*

Ist dies abgeschlossen, authentifiziert sich der Server gegenüber dem Client mit einem zuvor erstellten Zertifikat. Dieses enthält diverse Informationen, unter anderem Domainname (*FQDN*), Eigentümer des Zertifikats, Gültigkeitsdauer, Informationen über die ausstellende Zertifizierungsstelle (CA) und den Public Key (öffentlichen Schlüssel). Das Zertifikat wurde zuvor mit dem Private Key (privaten Schlüssel) der Zertifizierungsstelle signiert. Der Server und die Zertifizierungsstelle sind die einzigen, die diesen privaten Schlüssel kennen. Er darf niemals weitergegeben werden. Im weiteren Verlauf kann nun der Server ebenfalls ein Zertifikat zur Authentifizierung anfordern, dies ist aber optional. Daraufhin versucht der Client, die zuvor vom Server per Zertifikat übermittelte Identität zu überprüfen. Der Server muss zur Bestätigung nachweisen können, dass er im Besitz des zum Public-Key passenden privaten Schlüssels ist. Dazu sendet der Client dem Server 48 Byte zufälliger Daten (der Pre-Master-Secret), die zuvor mit dem Public Key verschlüsselt wurden. Der Server muss nun versuchen, diese Daten mit dem zum Public Key passenden Private Key zu entschlüsseln. Mithilfe dieser korrekt entschlüsselten Daten wird in der letzten Handshake-Phase zusammen mit der Anfangs ausgetauschten Zufallsdaten ein 48 Byte langes *Master-Secret* berechnet. Der Server sendet zum Abschluss anschließend eine mit dem MasterSecret verschlüsselte *ServerFinished* Meldung. Diese Meldung beinhaltet alle bisher ausgetauschten Nachrichten als MD5 sowie SHA-1 Hashwerten. Der Client berechnet nun ebenfalls das Master-Secret und kann dadurch feststellen, ob der Server im Besitz des zum Public Key passenden Private Key ist. Ist diese Überprüfung positiv verlaufen, so wird die weitere Verbindung serverseitig mit den Private Key und Clientseitig mit dem Public Key ver- bzw. entschlüsselt. *[54 S. 313-316]*

Neben der eigentlichen Verschlüsselung wird nun durch den Browser geprüft, ob das Zertifikat „echt“ ist und durch eine vertrauenswürdige Zertifizierungsstelle (CA) ausgestellt wurde. Ist dies nicht der Fall (passiert auch bei selbst signierten Zertifikaten), so wird dies durch eine auffällige Meldung im Browserfenster sichtbar gemacht.

Erstellung eines selbstsignierten Zertifikats

In der Regel sind durch Zertifizierungsstellen ausgestellte Zertifikate nicht kostenlos erhältlich. Daher wurde zu Testzwecken in dieser Arbeit ein selbst signiertes Zertifikat verwendet. Dieses Zertifikat steht in Sachen Verschlüsselung einem durch eine Zertifizierungsstelle ausgestelltes Zer-

tifikat in nichts nach. Nachteil ist nur, dass der Webbrowser nicht in der Lage ist herauszufinden, ob der Inhaber des Zertifikats auch der rechtmäßige Besitzer ist.

Zur Erstellung des Zertifikats wurde das unter Linux verfügbare OpenSSL-Toolkit verwendet.

In der Konsole wird dazu folgender Befehl eingegeben:

```
openssl req -newkey rsa:2048 -new -nodes -x509 -days 3650 -keyout
key.pem -out cert.pem
```

Dieser Befehl erzeugt einen 2048 bit langen Private Key *(key.pem)*, sowie ein 10 Jahre gültiges selbst signiertes Zertifikat *(cert.pem)*.

Nach Ausführen des Befehls werden folgende Daten abgefragt:

```
//2-stelliger Ländercode
Country Name (2 letter code) [AU]:DE
//Staat oder Provinz – Für Deutschland Bundesland
State or Province Name (full name) [Some-State]:Niedersachsen
//Stadt
Locality Name (eg, city) []:Stade
//Firmenname
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Firma
Mustermann
//Abteilungsname
Organizational Unit Name (eg, section) []:IT Abteilung
//FQDN (Fully Qualified Domain Name) des Servers
Common Name (e.g. server FQDN or YOUR name) []:www.smarterpi.org
//Email-Adresse
Email Address []:it@smarterpi.org
```

Diese Daten werden im Zertifikat hinterlegt und können später vom Benutzer im Browser eingesehen werden.

Probleme mit dem Webbrowser Safari in Verbindung eines selbstsignierten Zertifikats

Während der Entwicklung sind Probleme bei der Verwendung von selbstsignierten Zertifikaten in Verbindung des Node.js-Moduls Socket.IO aufgefallen. Dieses Problem betrifft sowohl Desktop-Geräte als auch mobile Geräte und äußert sich dahingehend, dass versucht wird, eine SSL-verschlüsselte Websockets-Verbindung aufzubauen, diese aber mit der Fehlermeldung *„WebSocket network error: OSStatus Error -9807: Invalid certificate chain“* in der JavaScript-Konsole abgebrochen wird. Als Fallback-Übertragungsmethode wird durch Socket.IO auf eine Polling-Verbindung umgeschaltet. Dieser Vorgang dauert im

Extremfall bis zu 2 Sekunden. Das Problem ist bekannt. Als Abhilfe ist zum Zeitpunkt der Erstellung dieser Arbeit nur die Nutzung eines durch eine Zertifizierungsstelle signiertes Zertifikat bekannt.

4.2.12 Fehlerbehandlung

Fehler, die den eigentlichen Programmablauf betreffen werden mithilfe des Node.js Logging-Moduls Winston aufgenommen. Dies ist eine simple und universell einsetzbare Bibliothek die zum Loggen diverser Informationen, z.B. Fehler genutzt werden kann. Winston unterstützt dabei mehrere Transportwege für diese Informationen. Je nach Konfiguration lassen sich diese z.B. in einer Datenbank ablegen, einer Datei speichern oder auf die Konsole ausgeben. Das Modul verfügt über verschiedene Dringlichkeitsstufen, denen auf der Konsole Farben zugewiesen werden können (z.B. Hinweise in Gelb, Fehler in Rot).

Verwendung in diesem Projekt:

Alle Fehler werden zusammen mit einer kurzen Beschreibung und der Modulbezeichnung an Winston übergeben. So lässt sich später schnell herausfinden, an welcher Stelle und in welchem Modul dieser Fehler aufgetreten ist. Im Unterschied zum Produktivbetrieb verfügt man während der Entwicklungszeit in der Regel über eine Konsole. Winston wurde so konfiguriert, dass die Umgebungsvariable `process.env.NODE_ENV` dahingehend geprüft wird, ob es sich um ein Produktivsystem oder ein Entwicklungssystem handelt. Während des Entwicklungsbetriebs wird die Ausgabe in der Konsole vorgenommen, im Produktivbetrieb werden die Fehler im JSON-Format zusammen mit Uhrzeit, Fehlerbeschreibung und diverser Metainformationen in eine Datei geschrieben.

5 Erprobung

5.1 Performance Tests

Verschiedenste Browser wurden bezüglich des Seitenaufbaus sowie der Reaktionsgeschwindigkeit bei Systemänderung getestet.

Der Testrechner besitzt folgende Hardware- und Softwarekonfiguration:

Prozessor:	Intel Core i7-2600K (4 x 3,4 GHz)
RAM:	16 GB DDR3 (1333 MHz)
Festplatte:	SSD - 240 GB OCZ Vertex 2
Betriebssystem:	Windows 8

Getestet wurden folgende Browser:

- Mozilla Firefox Version 29.0.1
- Google Chrome Version 35.0.1916.114
- Apple Safari Version 7.0.3
- Internet Explorer 11.0.9000.16384

Webseitenaufbau

Zur Zeitmessung des Webseitenaufbaus wurden die browsereigenen Messwerkzeuge verwendet. Jede Messung wurde insgesamt zehn Mal durchgeführt, tabellarisch erfasst und über die Messwerte gemittelt. Um auszuschließen, dass sich noch Dateien im Zwischenspeicher befinden, wurde nach jeder einzelnen Messung der Browsercache geleert.

Getestet wurde anhand der Startseite (dem Dashboard), auf der sich während des Tests folgende Elemente befanden:

- 2 Temperaturkurven mit je 100 Messwerten (diese haben am meisten Einfluss auf die Dauer des Webseitenaufbaus)
- 3 Schalter
- 3 Slider (Dimmer)
- 4 Scene-Buttons
- sowie das Menü

Die Verbindung zum Server wurde über eine gesicherte SSL-Verbindung hergestellt.

Ergebnis

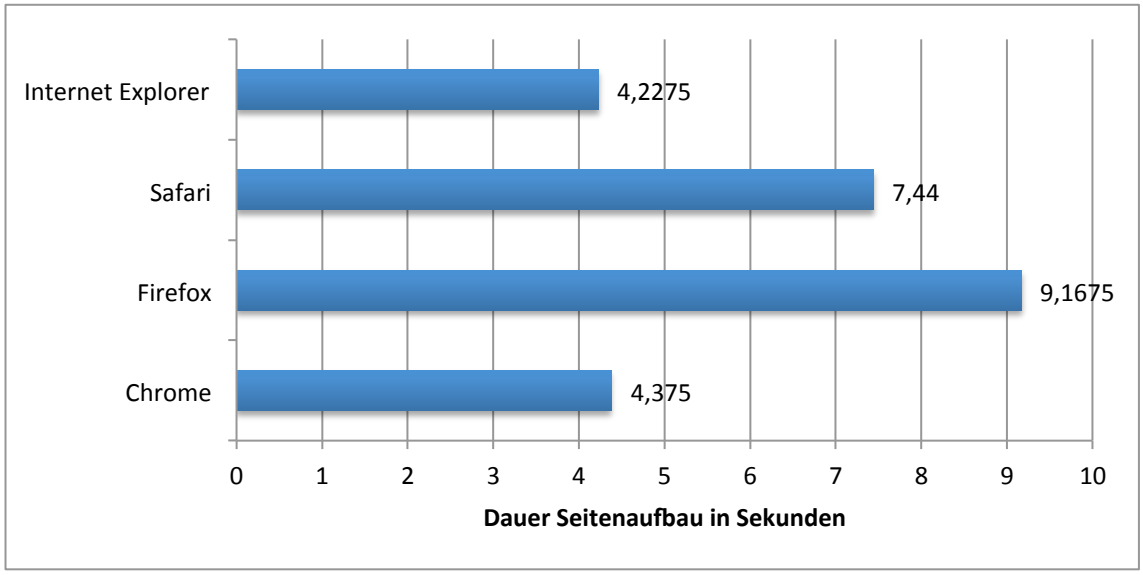


Abbildung 42: Ergebnis: Geschwindigkeitstest Webseitenaufbau

Wie man gut erkennen kann, sind Internet Explorer und Chrome beim Webseitenaufbau mit am schnellsten. Safari sowie Firefox benötigen in etwa doppelt so lange für den Seitenaufbau des Dashboardes.

Reaktionsgeschwindigkeit

Beim Reaktionstest sollte die Zeit zwischen Änderung und Reaktion gemessen werden. Dazu wurde auf zwei Computern das Dashboard aufgerufen und auf einem der Rechner diverse Schalter sowie Slider betätigt. Die Zeit zwischen Mausklick auf dem ersten Computer und Reaktion auf dem zweiten wurde versucht mit einer Stoppuhr zu messen. Wie erwartet, stellte sich dieses Messverfahren als sehr schwierig heraus, da die Reaktionszeiten aller Browser unterhalb einer Sekunde lagen. Trotz des ungenauen Messverfahrens sind diese Werte schon durch die augenscheinliche Betrachtung der Reaktionszeit mehr als zufriedenstellend.

5.2 Z-Wave-Reichweitentest am Beispiel eines Einfamilienhauses

Dieser Test wurde in einem Einfamilienhaus Baujahr 1964 (Massivbauweise) mit 4 Etagen (Keller, Erdgeschoss, Obergeschoss und Dachboden) durchgeführt.

Testbedingung 1 (ein Z-Wave-Gerät im selben Raum wie die Steuerung)

Bei diesem Test wurde überprüft, ob die Reichweite der kleinen Antenne auf dem Z-Wave-Modul ausreichend ist, um mit einem einzelnen (aktiven) Gerät im selben Raum zu kommunizieren. Dazu wurde die Steuerung in einer Ecke des Raumes platziert und die Kommunikation mit dem Gerät in allen 3 anderen Ecken des Raumes überprüft. Obwohl das Gerät teilweise durch Möbel, wie Schränke und einem Sofa verdeckt war, funktionierte die Kommunikation hier reibungslos.

Testbedingung 2 (ein Z-Wave-Gerät im Keller, Steuerung in 1. Etage)

Bei diesem Test stellte sich heraus, dass zwar ab und zu eine Kommunikation zwischen Steuerung und Z-Wave-Gerät stattfand, diese aber nicht zuverlässig funktionierte.

Testbedingung 3 (ein Z-Wave-Gerät im Keller, ein zweites Gerät im Erdgeschoss, Steuerung in 1. Etage)

Dieser Test verlief reibungslos; das zweite Gerät im Erdgeschoss reichte die zur Kommunikation benötigten Daten sicher an das im Keller platzierte Gerät weiter (und umgekehrt). Wie erwartet, funktionierte die Kommunikation mit dem im Erdgeschoss platzierten Gerät reibungslos.

Testbedingung 4 (ein akives Z-Wave-Gerät im Erdgeschoss, ein passives - batteriebetriebenes Gerät im Erdgeschoss, Steuerung in 1. Etage)

Auch dieser Test verlief ohne Probleme, die Kommunikation mit dem passiven Gerät funktionierte problemlos.

Fazit

Die Reichweite des Z-Wave-Moduls ist zwar begrenzt, stellt aber bei einer Konfiguration mit mehreren aktiven Geräten, die zusammen ein vermaschtes Netzwerk herstellen, keinerlei Probleme dar.

6 Fazit und Ausblick

6.1 Fazit

Im Rahmen dieser Arbeit ist eine solide Hard- und Softwarelösung zur Heimautomatisierung entstanden. Durch den Einsatz einer grafischen Benutzeroberfläche basierend auf einer Webapplikation, kann das System von vielen mobilen Geräten und Desktop-Geräten aus gesteuert werden. Bedingt durch die Verwendung des Websocket-Protokolls, kann eine bidirektionale Verbindung mit dem Browser hergestellt und durch Ereignisse hervorgerufene Änderungen direkt an diesen übertragen werden.

Die entstandene Software bietet eine gute Grundlage, um in weiteren Open-Source-Projekten (vielleicht auch nur in Teilauszügen) Anwendung zu finden. Schon in der jetzigen Software-Entwicklungsstufe lassen sich viele Szenarien im Bereich der Heimautomatisierung abdecken. Durch eine gute Dokumentation des Source-Codes ist es anderen Entwicklern möglich, die Software nach eigenen Wünschen anzupassen und den Funktionsumfang der Software zu erweitern.

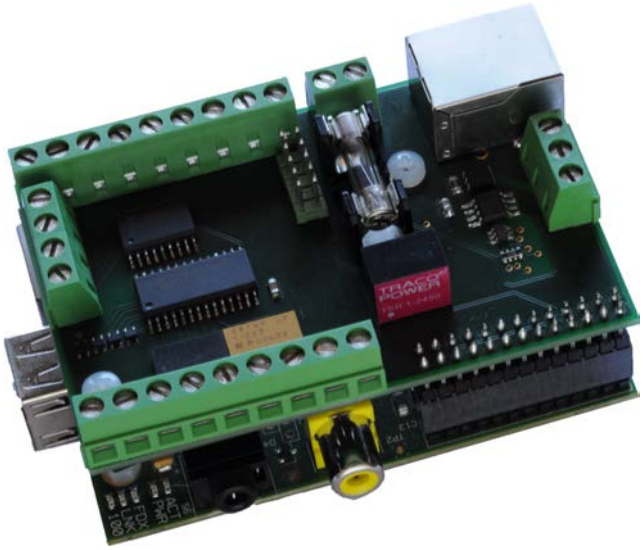


Abbildung 43: fertiggestellte Haussteuerung (Prototyp 2 + Raspberry Pi)

Die Entwicklung der Software verlief recht schnell. Nicht ganz 8 Wochen wurden von der ersten Überlegung bis zur fertigen Umsetzung benötigt. Zu verdanken ist dies der guten Dokumentation von Node.js, sowie der verwendeten Open-Source-Module. Trotz vieler Kritik, die an der Programmiersprache Ja-

vaScript (z.B. JavaScript, die Callback Hölle[11]) ausgeübt wird, habe ich viel Freude bei der Entwicklung gehabt. Nicht zuletzt ist dies unter Anderem der JavaScript IDE Webstorm der Firma JetBrains zu verdanken, die mir während des Programmierens viel Arbeit abgenommen hat. Obwohl die Entwicklung zu großen Teilen problemlos verlief, so gab es dennoch einige Schwierigkeiten. Das Anfangs in Node.js verwendete I^2C-Modul verursachte bei Ansprache des MCP3424 Delta-Sigma-Wandlers sporadisch einen Segmentation Fault[12] und führte zu einem Absturz des Programms. Trotz intensiver Fehlersuche konnte keine Ursache für dieses Problem gefunden werden. Auch dies war einer der Gründe, warum der Delta-Sigma-Wandler im 2. Prototypen entfallen ist.

Das Evaluieren der Schaltung auf einem Breadboard[13] führte schnell zu ersten motivierenden Erfolgserlebnissen. Der Aufbau des ersten Prototypen auf einer Lochrasterplatine stellte eine Herausforderung dar, führte aber aufgrund guter Vorarbeit schnell zu einem voll funktionstüchtigem Prototypen. Der Entwurf des Platinenlayouts mit Hilfe der PCB-Design-Software Eagle war mit der Schwierigkeit verbunden, dass solche Entwicklungsaufgaben bisher nicht Bestandteil des Studiums waren. Daher musste sich die Bedienung der Software vollständig selbst erarbeitet werden. Diese Herangehensweise spiegelt jedoch den Grundgedanken eines Studiums wieder und wird einem im späteren Arbeitsleben sicher erneut begegnen.

Die Stabilität der Hard- und Software zeigte in Testläufen sehr gute Ergebnisse. Allerdings konnten aus Zeitgründen keine Langzeittests durchgeführt werden. Über einem Zeitraum von 6 Wochen traten hier jedoch keinerlei Probleme auf.

6.2 Ausblick

Die Entwicklung der Hard- und Software dieses Open-Source-Projektes ist noch lange nicht abgeschlossen. So gibt es noch eine Vielzahl an Z-Wave Geräten, die derzeit nicht unterstützt werden, darunter sind z.B. Rollladensteuerungen und Heizungsthermostate. Außerdem muss eine Möglichkeit gefunden werden, das System bei einem Stromausfall sicher herunterfahren zu können, um Dateninkonsistenzen in der Datenbank zu vermeiden. Durch eine spätere Veröffentlichung dieses Projektes auf der Webseite http://www.smarterpi.org, sowie auf der Open-Source Hosting-Platform GitHub, können andere Entwickler mithelfen, an Lösungen dieser Probleme zu arbeiten.

---

[11] JavaScript Callback Hell - http://callbackhell.com

[12] Der Segmentation Fault (Schutzverletzung) tritt auf, wenn auf geschützte Ressourcen, insbesondere Speicherbereiche zugegriffen wird.

[13] Das Breadboard ist ein Steckbrett zur prototypischen Entwicklung von Schaltungen ohne Lötaufwand.

7 Literaturverzeichnis

Hinweis

Aufgrund des Themenfeldes dieser Arbeit, die sich mit jungen Technologien im Fachbereich der Informations- und Elektrotechnik beschäftigt, handelt es sich bei der Vielzahl der angegebenen Quellen um Internetseiten. Für Seiten, bei denen die Autoren nicht oder nicht eindeutig identifizierbar waren, wird die Entwicklergruppe oder das Unternehmen hinter der jeweiligen Internetpräsenz genannt.

[1]. **KNX.** [Online] [Abgerufen am: 2014. Juni 2014.] http://www.knx.org/de/was-ist-knx/knx-was-ist-das/.

[2]. **Upton, Eben - Raspberry Pi Foundation.** RASPBERRY PI – 2006 EDITION. [Online] 23. Oktober 2011. [Abgerufen am: 7. Juni 2014.] http://www.raspberrypi.org/raspberry-pi-2006-edition/.

[3]. —. ALPHA BOARDS IN MANUFACTURE. [Online] 25. Juli 2011. [Abgerufen am: 7. Juni 2014.] http://www.raspberrypi.org/alpha-boards-in-manufacture/.

[4]. **Upton, Liz - Raspberry Pi Foundation.** MADE IN THE UK! [Online] 6. September 2012. [Abgerufen am: 7. Juli 2014.] http://www.raspberrypi.org/made-in-the-uk/.

[5]. —. 1.75 MILLION SOLD SO FAR – AND 1 MILLION MADE IN THE UK. [Online] 8. Oktober 2013. [Abgerufen am: 7. Juli 2014.] http://www.raspberrypi.org/1-75-million-sold-and-1-million-made-in-the-uk/.

[6]. **Upton, Eben - Raspberry Pi Foundation.** A BIRTHDAY PRESENT FROM BROADCOM. [Online] 28. Februar 2014. [Abgerufen am: 7. Juli 2014.] http://www.raspberrypi.org/a-birthday-present-from-broadcom/.

[7]. **Broadcom.** [Online] [Abgerufen am: 18. Mai 2014.] http://www.broadcom.com/products/BCM2835.

[8]. **Rapberry Pi.** [Online] [Abgerufen am: 17. Mai 14.] http://www.raspberrypi.org/help/faqs/.

[9]. **Raspberry Pi Foundation.** Downloads. [Online] [Abgerufen am: 7. Juni 2014.] http://www.raspberrypi.org/downloads/.

[10]. **Upton, Eben - Raspberry Pi Foundation.** ANDROID 4.0 IS COMING! [Online] 31. Juli 2012. [Abgerufen am: 7. Juni 2104.] http://www.raspberrypi.org/android-4-0-is-coming/.

[11]. **Atmel.** AVR315: Using the TWI module as I2C master. [Online] 2010. [Abgerufen am: 7. Juni 2014.] http://www.atmel.com/Images/doc2564.pdf.

[12]. **Irazabal, Jean-Marc und Irazabal, Steve.** AN10216-01 - I2C Manual. [Online] 24. März 2003. [Abgerufen am: 7. Juni 2014.] http://www.nxp.com/documents/application_note/AN10216.pdf.

[13]. **Schröder, Joachim, Gockel, Tilo und Dillmann, Rüdiger.** *Embedded Linux - Das Praxisbuch, ISBN 978-3-540-78619-1.* Berlin Heidelberg : Springer, 2009.

[14]. **Embedded Systems Academy.** Overview of the Different Versions of the I2C Specification. [Online] [Abgerufen am: 11. Juni 2014.] http://www.esacademy.com/en/library/technical-articles-and-documents/miscellaneous/i2c-bus/miscellaneous-information/overview-of-the-different-versions-of-the-i2c-specification.html.

[15]. **NXP Semiconductors.** [Online] 04. April 2014. [Abgerufen am: 19. Mai 2014.] http://www.nxp.com/documents/user_manual/UM10204.pdf.

[16]. **Philips Semiconductors.** The I2C-bus and how to use it. [Online] 1995. [Abgerufen am: 11. Juni 2014.] http://www.i2c-bus.org/fileadmin/ftp/i2c_bus_specification_1995.pdf.

[17]. **Elektronik Kompendium.** Open-Collector (OC). [Online] [Abgerufen am: 11. Juni 2014.] http://www.elektronik-kompendium.de/sites/slt/1206121.htm.

[18]. **Maxim Integrated.** Guidelines for Reliable Long Line 1-Wire Networks. [Online] 22. August 2008. [Abgerufen am: 19. Mai 2014.] http://www.maximintegrated.com/app-notes/index.mvp/id/148.

[19]. —. 1-Wire Tutorial. [Online] [Abgerufen am: 19. Mai 2014.] http://www.maximintegrated.com/products/1-wire/flash/overview/index.cfm.

[20]. —. 1-Wire Communication Through Software. [Online] 08. April 2009. [Abgerufen am: 20. Mai 2014.] http://www.maximintegrated.com/app-notes/index.mvp/id/126.

[21]. **Z-Wave Alliance.** Alliance overview. [Online] [Abgerufen am: 20. Mai 2014.] http://www.z-wavealliance.org/alliance-overview.

[22]. **Sigma Designs.** About Z-Wave. [Online] 2014. [Abgerufen am: 11. Juni 2014.] http://www.z-wave.com/what_is_z-wave.

[23]. **International Telecom Union.** Recommendation G.9959. [Online] [Abgerufen am: 20. Mai 2014.] http://www.itu.int/rec/T-REC-G.9959.

[24]. **Z-Wave Alliance.** Technology. [Online] [Abgerufen am: 20. Mai 2014.] http://www.z-wavealliance.org/technology.

[25]. **Wenz, Christian.** Galileo Computing - <openbook>. [Online] 2002. [Abgerufen am: 15. Mai 2014.] http://openbook.galileocomputing.de/javascript/javascript01.htm.

[26]. **Mozilla Presse.** Gründungsmitglied Brendan Eich wird neuer CEO. [Online] [Abgerufen am: 10. Juni 2014.] https://blog.mozilla.org/press-de/2014/03/24/mozilla-grundungsmitglied-brendan-eich-wird-neuer-ceo/.

[27]. **Microsoft.** Microsoft News Center. [Online] 29. Mai 1996. [Abgerufen am: 15. Mai 2014.] http://www.microsoft.com/en-us/news/press/1996/may96/ie3btapr.aspx.

[28]. **ECMA Intarnational.** [Online] 2011. [Abgerufen am: 15. Mai 2014.][Zitat] http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf.

[29]. **SELFHTML e.V.** Einführung in JavaScript und DOM. [Online] [Abgerufen am: 12. Juni 2014.] http://de.selfhtml.org/javascript/intro.htm.

[30]. **Bewersdorf, Jörg.** *Objektorientierte Programmierung mit JavaScript, ISBN 978-3-658-05443-4.* Wiesbaden : Springer Verlag, 2014.

[31]. **Microsoft.** Using JavaScript Along with ASP.NET. [Online] [Abgerufen am: 12. Juni 2014.] http://msdn.microsoft.com/en-us/library/aa479011.aspx.

[32]. **Unity Technologies.** Scripting API. [Online] 2014. [Abgerufen am: 12. Juni 2014.] http://docs.unity3d.com/ScriptReference/.

[33]. **MongoDB.** Server-side JavaScript. [Online] [Abgerufen am: 12. Juni 2014.] http://docs.mongodb.org/manual/core/server-side-javascript/.

[34]. Google V8 ChangeLog. [Online] [Abgerufen am: 12. Juni 2014.] https://code.google.com/p/v8/source/browse/branches/0.1/ChangeLog.

[35]. **Google Inc.** Google Developers - Introduction. [Online] [Abgerufen am: 12. Juni 2014.] [Zitat] https://developers.google.com/v8/intro.

[36]. —. Google Developers. [Online] [Abgerufen am: 15. Mai 2014.] [Zitat] https://developers.google.com/v8/design.

[37]. **Node.js.** Node.js. [Online] [Abgerufen am: 17. Mai 2014.] http://nodejs.org.

[38]. —. *Node.js - About.* [Online] [Abgerufen am: 12. Juni 2014.] http://nodejs.org/about/.

[39]. **Hughes-Croucher, Tom und Wilson, Mike.** *Node: Up and Running - ISBN 978-1-4493-9858-3.* Sebastopol : O'Reilly Media.

[40]. **Node.js.** *Node.js v0.10.29 Manual & Documentation - Modules.* [Online] [Abgerufen am: 12. Juni 2014.] http://nodejs.org/api/modules.html.

[41]. **CommonJS.** Getting CommonJS. [Online] [Abgerufen am: 21. Mai 2014.] http://www.commonjs.org/impl/.

[42]. Understanding the node.js event loop. [Online] [Abgerufen am: 12. Juni 2014.] http://blog.mixu.net/2011/02/01/understanding-the-node-js-event-loop/.

[43]. **Socket.IO.** Socket.IO. [Online] [Abgerufen am: 10. Mai 2014.] http://socket.io/.

[44]. **Traco Electronic.** TSR-1 Serie Datenblatt. [Online] [Abgerufen am: 27. Mai 2014.] http://www.tracopower.com/datasheet_g/tsr1-d.pdf.

[45]. **NXP Semiconductors.** *AN10441 - Level shifting techniques in I2C-bus design.* [Online] 18. Juni 2007. [Abgerufen am: 26. Mai 2014.] http://www.nxp.com/documents/application_note/AN10441.pdf.

[46]. **Microchip Technology Inc.** [Online] 2007. [Abgerufen am: 26. Mai 2014.] http://ww1.microchip.com/downloads/en/DeviceDoc/21952b.pdf.

[47]. **Texas Instruments.** ULN2803A - Datenblatt. [Online] Februar 1997. [Abgerufen am: 27. Mai 2014.] http://www.ti.com/lit/ds/symlink/uln2803a.pdf.

[48]. **Z-Wave.Me.** RaZberry. [Online] [Abgerufen am: 27. Mai 2014.] http://razberry.z-wave.me/.

[49]. —. The RaZberry Daughter Card. [Online] [Abgerufen am: 27. April 2014.] http://razberry.z-wave.me/index.php?id=9.

[50]. —. Software. [Online] [Abgerufen am: 27. Mai 2014.] http://razberry.z-wave.me/index.php?id=10.

[51]. **Maxim Integrated.** DS2482-100 Datenblatt. [Online] 2012. [Abgerufen am: 28. Mai 2014.] http://datasheets.maximintegrated.com/en/ds/DS2482-100.pdf.

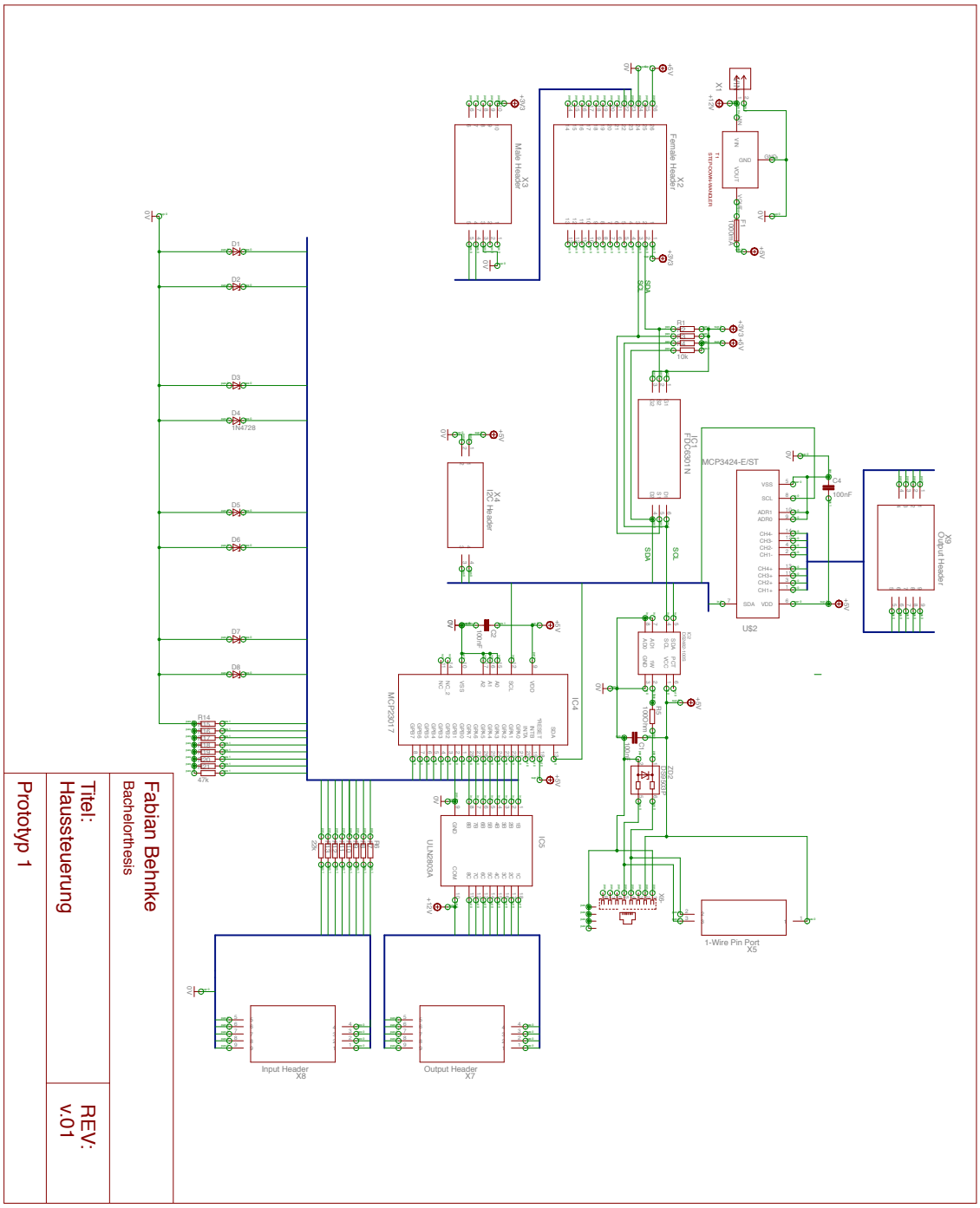
[52]. —. DS9503 Datenblatt. [Online] 2009. [Abgerufen am: 28. Mai 2014.] http://datasheets.maximintegrated.com/en/ds/DS9503.pdf.

[53]. **Prof. Dr. Jens Lienig, Dipl.-Ing. Göran Jerke.** Elektromigration. [Online] 2002. [Abgerufen am: 29. Mai 2014.] http://www.ifte.de/mitarbeiter/lienig/fm_part1.pdf.

[54]. **Zahn, Markus.** *Unix Netzwerkprogrammierung mit Threads, Sockets und SSL - ISBN-10 3-540-00299-5.* Berlin, Heidelberg, New York : Springer-Verlag, 2006.[Zitat]

Anhänge

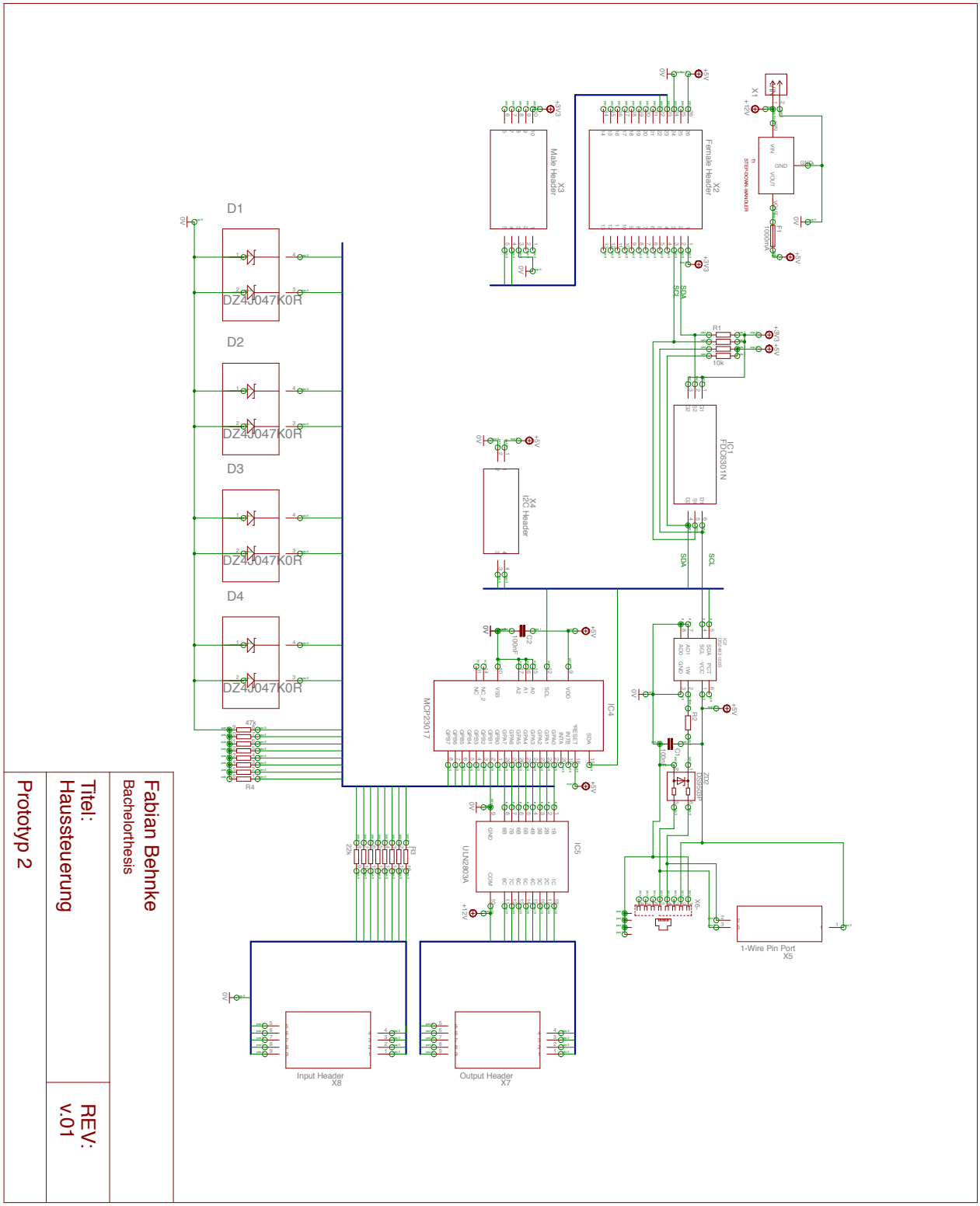
A1) Schaltplan Prototyp 1



Hinweis:

Die Eagle-Datei, sowie der Schaltplan im PDF-Format sind ebenfalls auf der CD/DVD enthalten.

A2) Schaltplan Prototyp 2



Hinweis:

Die Eagle-Datei, sowie der Schaltplan im PDF-Format sind ebenfalls auf der CD/DVD enthalten.

A3) Bedienungsanleitung (23 Seiten)

Bedienungsanleitung (Webinterface)

The Smarter Pi

Manual - German v1.0

Bedienungsanleitung (Webinterface)

Inhaltsverzeichnis

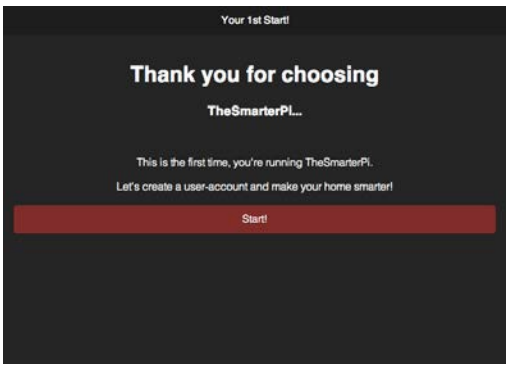
Bedienungsanleitung (Webinterface)

1 Erster Start

The Smarter Pi unterstützt eine Vielzahl aktuell verfügbarer Webbrowser, dazu zählen z.B. Google Chrome, Microsoft Internet Explorer, Mozilla Firefox und Apple Safari (alle sowohl in Mobiler-, als auch in Desktop-Ausführung).

Rufen Sie das Webinterface in Ihrem Lieblingsbrowser auf:

https://<<IP des Raspberry Pi>>:3000 (z.B. https://192.168.1.25:3000)
(Ist bei der vorherigen Konfiguration SSL deaktiviert worden, so ersetzen Sie https:// durch http://.)

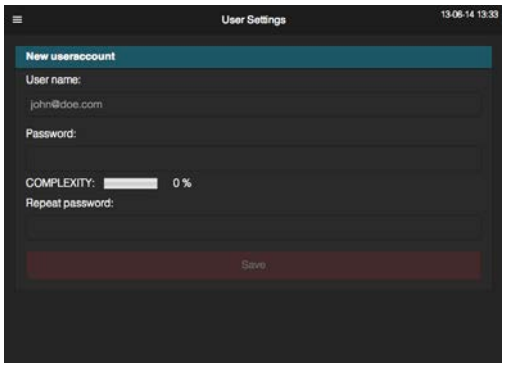


Sie werden daraufhin Begrüßt und darauf hingewiesen, dass Sie TheSmarterPi zum ersten Mal starten.

Drücken Sie auf **[Start]** zum fortfahren.

Sie werden nun aufgefordert einen neuen Benutzer anzulegen, geben Sie dazu im Feld unterhalb von **[User name]** einen gewünschten Benutzernamen ein (Dies kann aber z.B. auch eine E-Mail-Adresse sein). Daraufhin tragen im Feld unterhalb von **[Password]**, Ihr gewünschtes Passwort ein. **[COMPLEXITY]** zeigt an, ob Ihr gewähltes Passwort sicher genug ist. Es muss mindestens eine Komplexität von 20% erreicht werden. Verwendenden Sie am besten eine Kombination aus Buchstaben (in Groß und Kleinschreibung), Zahlen und Sonderzeichen. Wiederholen Sie die Eingabe des Passworts im Feld Unterhalb von **[Repeat Password].**

Schließen sie den Vorgang durch drücken auf **[Save]** ab.

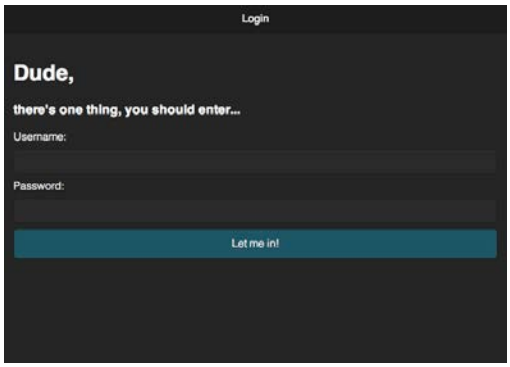


Bedienungsanleitung (Webinterface)

Sie werden nun zur Login-Seite weitergeleitet.
(Diese Seite wird Sie in Zukunft begrüßen, sofern Sie nicht am System eingeloggt sind)

Tragen Sie im Feld unterhalb von **[Username]** Ihren zuvor festgelegten Benutzernamen ein. Ihr Passwort tragen Sie in das Feld unterhalb von **[Password]** ein.

Drücken Sie auf **[Let me in!]** um sich am System anzumelden.



Sofern das System noch nicht eingerichtet ist, wird Ihnen diese Seite angezeigt, Sie soll einen kurzen überblich darüber verschaffen was Sie nun noch zu tun haben.

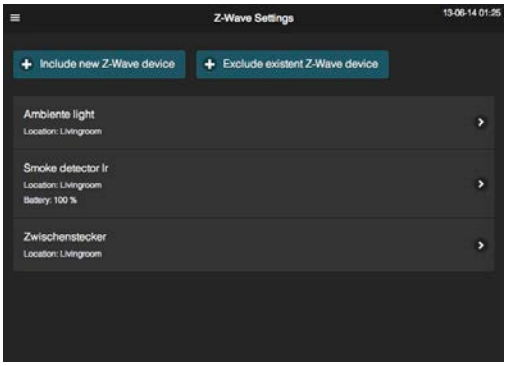
Der rote Pfeil weist darauf hin, dass sich das Menü unter den drei waagerecht verlaufenden Strichen verbirgt.

Bedienungsanleitung (Webinterface)

2 Einstellungen

2.1 Z-Wave

Um in die Einstellungen für Z-Wave Geräte zu gelangen drücken Sie im Menü auf **[Settings]** -> **[Z-Wave]**.



In dieser Oberfläche können Sie Z-Wave Geräte in das System integrieren, wieder auf dem System entfernen und bearbeiten.

Hinweis: Zum bearbeiten eines Geräts drücken Sie auf die jeweilige Schaltfläche.

2.1.1 Z-Wave Gerät ins System integrieren

Um ein Z-Wave Gerät in das System zu integrieren drücken Sie auf die auf **[Include new Z-Wave device]**.

Folgender Hinweis erscheint nun auf Ihrem Bildschirm:

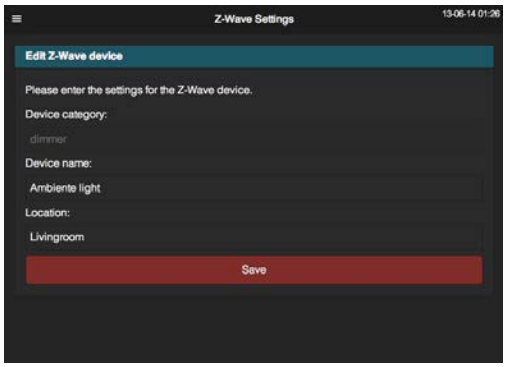
„Press a button on the device to be included…“

Hinweis: Zum abbrechen dieses Vorgangs drücken Sie auf **[Stop]**.

Drücken Sie nun auf den Include Taster auf Ihrem Z-Wave Gerät. Bitte entnehmen Sie der Bedienungsanleitung des jeweiligen Gerätes, wo sich dieser befindet. (Oftmals ist es ein 3 maliges drücken des Einschaltknopfes).

Wurde das Gerät erfolgreich dem System hinzugefügt, werden Sie nun in die Bearbeitungs-Maske des jeweiligen Gerätes weitergeleitet.

Bedienungsanleitung (Webinterface)



Sie bekommen nun unterhalb von **[Device category]** angezeigt, zu welcher Kategorie dieses Gerät gehört.

Geben Sie nun im Feld unterhalb von **[Device name]** einen Namen für das Gerät ein.

Unter **[Location]** können Sie eintragen wo sich das Gerät befindet: z.B. „Wohnzimmer, links hinter Sofa“

Hinweis: Der Eintrag unter **[Location]** hat keinerlei Einfluss auf die spätere Zuweisung eines Raumes und dient lediglich als Kommentarfeld.

Drücken Sie **[Save]** zum Abschließen des Vorgangs.

2.1.2 Z-Wave Gerät aus System entfernen

Möchten Sie ein Z-Wave Gerät auf dem System entfernen So drücken Sie im Z-Wave Menü auf **[Exclude existent Z-Wave Device]**.

Es erscheint folgender Hinweis auf Ihrem Bildschirm:

„Press a button on the device to be excluded...“

Hinweis: Zum abbrechen dieses Vorgangs drücken Sie auf **[Stop]**.

Drücken Sie nun auf den Exculde Taster auf Ihrem Z-Wave Gerät. Bitte entnehmen Sie der Bedienungsanleitung des jeweiligen Gerätes, wo sich dieser befindet. (Oftmals ist es ein 3 maliges drücken des Einschaltknopfes).

Wurde das Gerät erfolgreich aus dem System entfernt verschwindet dieser Hinweis und der Vorgang ist abgeschlossen

Bedienungsanleitung (Webinterface)

2.2 Interne Geräte

Um in die Einstellungen für Interne Geräte zu gelangen, drücken Sie im Menü auf **[Settings]** -> **[Intern Devices]**.

Hier können Sie Ein- und Ausgänge hinzufügen, diese bearbeiten, oder durch drücken auf das [X] wieder entfernen.

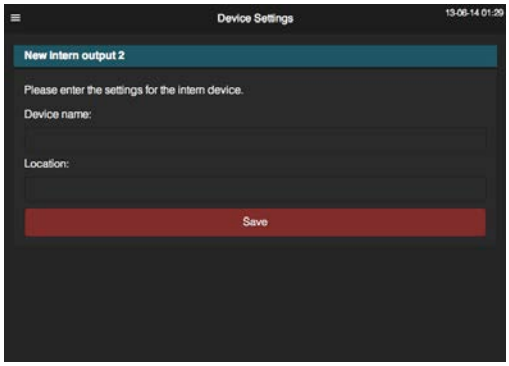
Außerdem können Sie hier ein sogenanntes HTTP Pseudo Device hinzufügen. Dies können z.B. Geräte und Dienste sein, die über eine HTTP REST API aktiviert werden können.

2.2.1 Ausgänge hinzufügen/bearbeiten

Um einen Ausgang hinzuzufügen, wählen Sie in dem Dropdown Feld unterhalb von **[Intern Outputs]** den gewünschten Ausgang **[Output 0-7],** drücken Sie daraufhin auf **[Add]**.

Zum Bearbeiten eines zuvor hinzugefügten Ausgangs drücken Sie auf die Schaltfläche (Grau) des Ausgangs.

Sie gelangen nun in die Maske zum Bearbeitern des jeweiligen Ausgangs.



Geben Sie nun im Feld unterhalb von **[Device name]** einen Namen für das Gerät ein.

Unter **[Location]** können Sie eintragen wo sich das Gerät befindet: z.B. „Wohnzimmer, links hinter Sofa“

Hinweis: Der Eintrag unter **[Location]** hat keinerlei Einfluss auf die spätere Zuweisung eines Raumes und dient lediglich als Kommentarfeld.

Drücken Sie **[Save]** zum Abschließen des Vorgangs.

Bedienungsanleitung (Webinterface)

2.2.2 Eingänge hinzufügen/bearbeiten

Um einen Eingang hinzuzufügen, wählen Sie in dem Dropdown Feld unterhalb von **[Intern Outputs]** den gewünschten Ausgang **[Output 0-7],** drücken Sie daraufhin auf **[Add]**.

Zum Bearbeiten eines zuvor hinzugefügten Eingangs drücken Sie auf die Schaltfläche (Grau) des Eingangs.

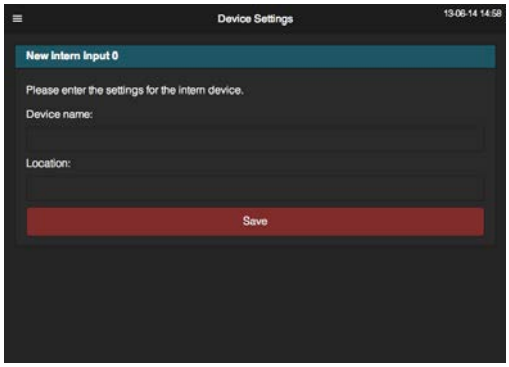
Sie gelangen nun in die Maske zum Bearbeitern des jeweiligen Eingangs.

Geben Sie nun im Feld unterhalb von **[Device name]** einen Namen für das Gerät ein.

Unter **[Location]** können Sie eintragen wo sich das Gerät befindet: z.B. „Wohnzimmer, links hinter Sofa“

Hinweis: Der Eintrag unter **[Location]** hat keinerlei Einfluss auf die spätere Zuweisung eines Raumes und dient lediglich als Kommentarfeld.

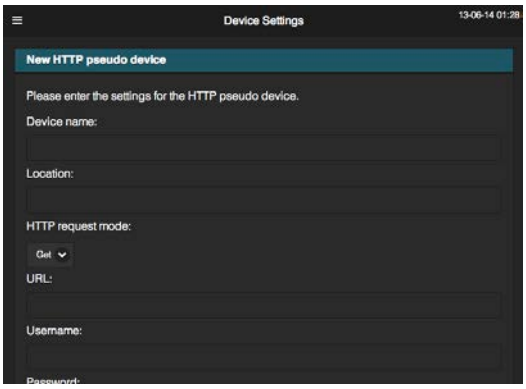
Drücken Sie **[Save]** zum Abschließen des Vorgangs.



2.2.3 HTTP Pseudo device hinzufügen/bearbeiten

Zum hinzufügen eines HTTP Pseudo devices drücken Sie unterhalb von **[HTTP pseudo device]** auf **[Add].**

Zum Bearbeiten eines zuvor hinzugefügten HTTP Ppeudo Devices drücken Sie auf die Schaltfläche (Grau) des jeweiligen Eintrags.



Sie gelangen nun in die Maske zum Bearbeiten des HTTP Pseudo Devices.

Geben Sie nun im Feld unterhalb von **[Device name]** einen Namen für das Gerät ein.

Unter **[Location]** können Sie eintragen wo sich das Gerät befindet: z.B. „Wohnzimmer, links hinter Sofa“

Wählen Sie [HTTP request mode]: POST oder GET

Falls benötigt tragen Sie nun Benutzername und Passwort ein (HTTP Basic Authentication).

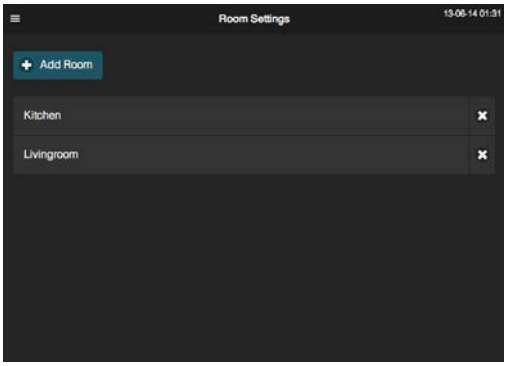
Fügen Sie falls nötig weitere Argumente durch drücken des Buttons **[Add]** hinzu.

Schließen Sie den Vorgang durch drücken auf **[Save]** ab.

Bedienungsanleitung (Webinterface)

2.3 Räume und Dashbaord

Um virtuelle Räume anzulegen, bearbeiten oder zu löschen drücken Sie im Menü auf **[Settings]** -> **[Rooms]**.



Hier können Sie Räume hinzufügen, diese bearbeiten, oder durch drücken auf das **[X]** wieder entfernen

2.3.1 Raum hinzufügen/bearbeiten

Zum Hinzufügen eines Raumes drücken Sie auf **[Add Room]**.

Zum Bearbeiten drücken sie auf die jeweilige Schaltfläche des zuvor angelegten Raumes.

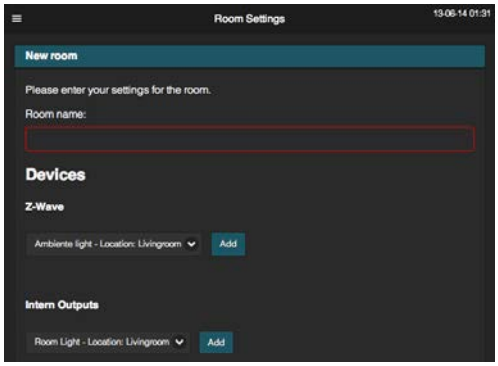
Sie gelangen in die Maske zum bearbeiten des Raumes.

Hier könne Sie dem jeweiligen Raum: Geräte, Temperaturfühler und Szenen zuweisen.

Geben Sie im Feld unterhalb von **[Room name]** einen gewünschten Namen für diesen Raum ein.

Fügen Sie dem Raum durch drücken auf **[Add]** in der jeweiligen Kategorie: Z-Wave Geräte, Interne Ausgänge, 1-Wire Temperatursensoren oder Szenen hinzu.

Drücken Sie **[Save]** zum Abschließen des Vorgangs.



2.3.2 Dashbaord bearbeiten

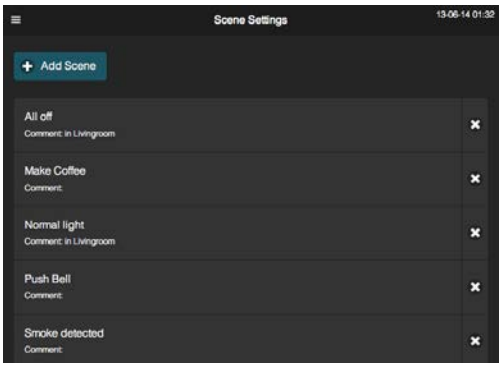
Zum bearbeiten des Dashboardes (die Startseite mit den meistgenutzten Funktionen) drücken Sie im Menü auf **[Settings]**-> **[Dashbaord]**. Danach verfahren Sie wie in 2.3.1 beschrieben.

Bedienungsanleitung (Webinterface)

2.4 Szenen

Szenen können dazu genutzt werden um bestimmte licht Stimmungen herzustellen, oder mehrere Geräte ein- oder auszuschalten.Eine Szene könne sein: „Alle Geräte im Wohnzimmer aus“ oder „Sende eine Push-Nachricht“. Diese Szenen können in den jeweiligen Räumen durch Schaltflächen oder durch festgelegte Regeln und Timer aufgerufen werden.

Um in die Einstellungen dieser Szenen zu gelangen drücken Sie im Menü auf **[Settings]** -> **[Scenes]**



Hier können Sie Szenen hinzufügen, diese bearbeiten, oder durch drücken auf das **[X]** wieder entfernen

2.4.1 Szene hinzufügen/bearbeiten

Um eine Szene hinzuzufügen drücken Sie auf **[Add Scene]**.

Zum Bearbeiten drücken sie auf die jeweilige Schaltfläche der zuvor angelegten Szene.

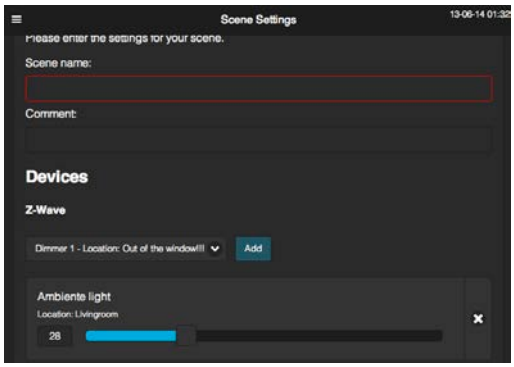
Sie gelangen nun in die Maske zum bearbeiten der Szene.

Geben Sie im Feld unterhalb von **[Scene name]** einen gewünschten Namen für diese Scene ein.

Optional können Sie diese Scene unter **[Comment]** mit einem Kommentar versehen.

Sie können der Scene nun durch drücken auf [Add] in der jeweiligen Kategorie: Z-Wave Geräte, Interne Ausgänge oder HTTP Psudo Devices hinzu.

Stellen Sie die gewünschten Werte der Geräte ein, die bei Aufruf dieser Szene angenommen werden sollen.



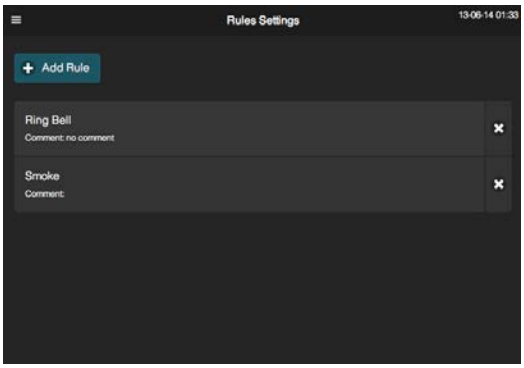
Es lässt sich außerdem durch drücken auf **[Create webhook URL]** eine sogenannte Webhook URL generieren, die genutzt werden kann die ausgewählte Szene in Drittanbieter-Anwendungen ausführen zu lassen. Drücken Sie **[Save]** zum Abschließen des Vorgangs.

Bedienungsanleitung (Webinterface)

2.5 Regeln

Regeln können dazu genutzt werden Szenen in Abhängigkeit von Gerätezuständen auszuführen.

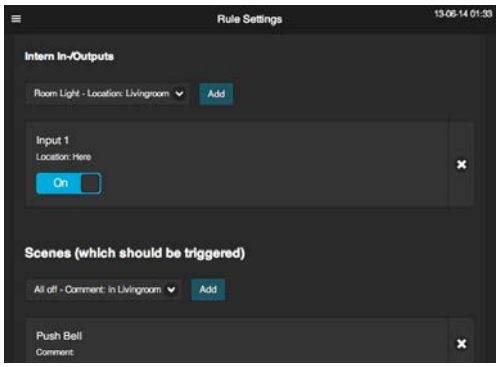
Um in die Einstellungen dieser Regeln zu gelangen drücken Sie im Menü auf **[Settings]** -> **[Rules]**.



Hier können Sie Regeln hinzufügen, diese bearbeiten, oder durch drücken auf das **[X]** wieder entfernen

2.5.1 Regeln hinzufügen/bearbeiten

Um eine Regel hinzuzufügen drücken Sie auf **[Add Rule]**.



Zum Bearbeiten drücken sie auf die jeweilige Schaltfläche der zuvor angelegten Regel.

Sie gelangen nun in die Maske zum bearbeiten der Regel.

Geben Sie im Feld unterhalb von **[Rule name]** einen gewünschten Namen für diese Regel ein.

Optional können Sie diese Regel unter **[Rule Comment]** mit einem Kommentar versehen.

Sie können der Regel nun durch drücken auf **[Add]** in der jeweiligen Kategorie: ein Z-Wave Gerät oder einen Interne Ein- oder Ausgäng zuweisen, der diese Regel ausführen soll.

Währen Sie nun darauf unter **[Scenes]** die gewünschte Szene, die ausgeführt werden soll. Durch drücken auf [Add] fügen Sie diese hinzu.

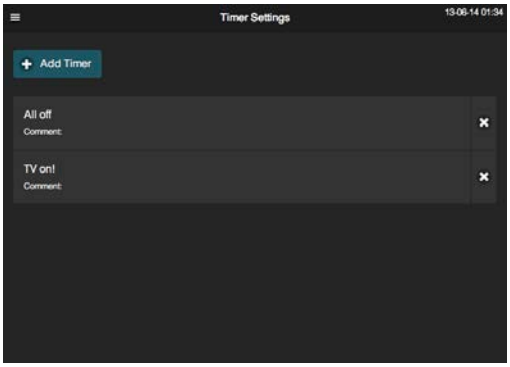
Drücken Sie **[Save]** zum Abschließen des Vorgangs.

Bedienungsanleitung (Webinterface)

2.6 Timer

Timer können dazu genutzt werden um Szenen Zeitgesteuert auszuführen.

Um in die Timer-Einstellungen zu gelangen wählen Sie im Menü auf **[Settings]** -> **[Timer]**.



Hier können Sie Timer hinzufügen, diese bearbeiten, oder durch drücken auf das **[X]** wieder entfernen

2.6.1 Timer hinzufügen/bearbeiten

Um einen Timer hinzuzufügen drücken Sie auf **[Add Timer]**.

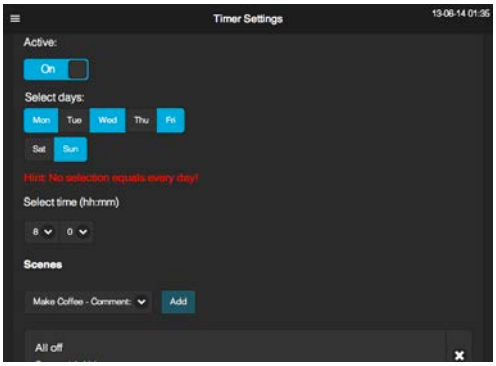
Zum Bearbeiten drücken sie auf die jeweilige Schaltfläche des zuvor angelegten Timer.

Sie gelangen nun in die Maske zum bearbeiten des Timers.

Geben Sie im Feld unterhalb von **[Timer name]** einen gewünschten Namen für diese Regel ein.

Optional können Sie diesen Timer unter **[Timer Comment]** mit einem Kommentar versehen.

Mit der Schaltfläche [Active], können Sie Timer aktiv oder inaktiv Schalten.



Wählen Sie unter **[Select days]** aus, an welchen Tagen dieser Timer aktiv werden soll. Wird kein Tag gewält, wird dieser Timer täglich ausgeführt.

Unter **[Select time]** kann nun die Uhrzeit gewählt werden, wann dieser Timer ausgeführt werden soll.

Zu guter Letzt wählen Sie die gewünscht Szene, die durch diesen Timer ausgeführt werden soll. Durch drücken auf **[Add]** wird die gewählte Scene hinzugefügt.

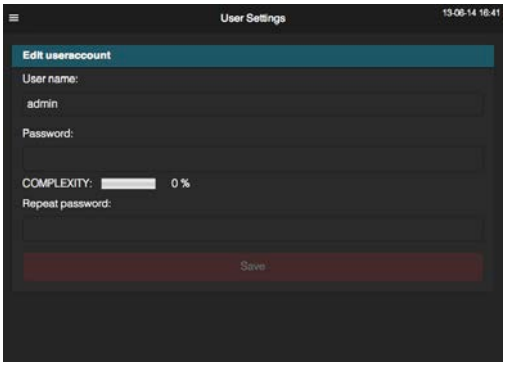
Drücken Sie **[Save]** zum Abschließen des Vorgangs.

Bedienungsanleitung (Webinterface)

2.7 Benutzeraccount

2.7.1 Benutzeraccount bearbeiten

Um Ihren Benutzernamen und/oder das Passwort zu ändern wählen Sie im Menü [Settings] -> [Useraccount].



Hier können Sie nun Ihren Benutzernamen und Ihr Passwort ändern.

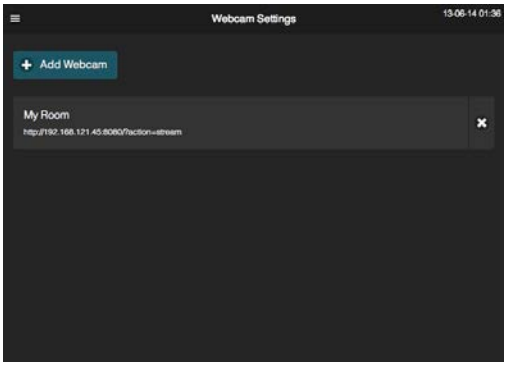
Geben Sie nun im Feld unterhalb von **[User name]** einen gewünschten Benutzernamen ein.

Daraufhin tragen im Feld unterhalb von **[Password]**, Ihr gewünschtes Passwort ein. **[COMPLEXITY]** zeigt an, ob Ihr gewähltes Passwort sicher genug ist. Es muss mindestens eine Komplexität von 20% erreicht werden. Verwendenden Sie am besten eine Kombination aus Buchstaben (in Groß und Kleinschreibung), Zahlen und Sonderzeichen. Wiederholen Sie die Eingabe des Passworts im Feld Unterhalb von **[Repeat Password].**
Schließen sie den Vorgang durch drücken auf **[Save]** ab.

2.8 Webcams

Besitzen Sie eine Webcam, die MJPG-Codierte Video-Streams liefern kann, so haben Sie die Möglichkeit, diese durch das System verschlüsseln zu lassen. Der Stream wird dann über die selbe verschlüsselte Verbindung durchgereicht, über die auch ihr Webinterface verbunden ist.

Um in die Webcam-Einstellungen zu gelangen rufen Sie im Menü [Settings] -> [Webcams] auf.



Hier können Sie Webcam-Steams hinzufügen, diese bearbeiten, oder durch drücken auf das **[X]** wieder entfernen

Bedienungsanleitung (Webinterface)

2.8.1 Webcam hinzufügen/bearbeiten

Um einen Webcam-Stream hinzuzufügen, drücken Sie daraufhin auf **[Add Webcam]**.

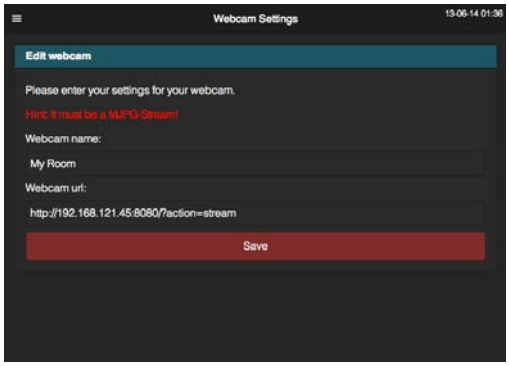
Zum Bearbeiten eines zuvor hinzugefügten Eingangs drücken Sie auf die Schaltfläche (Grau) des Webcam-Streams.

Sie gelangen nun in die Maske zum Bearbeitern des jeweiligen Eintrags.

Geben Sie nun im Feld unterhalb von **[Webcam name]** einen Namen für die Webcam ein.

Unter **[URL]** tragen Sie die Adresse des MJPG Webcam Streams ein, der über die verschlüsselte Verbindung durchgereicht werden soll.

Drücken Sie **[Save]** zum Abschließen des Vorgangs.

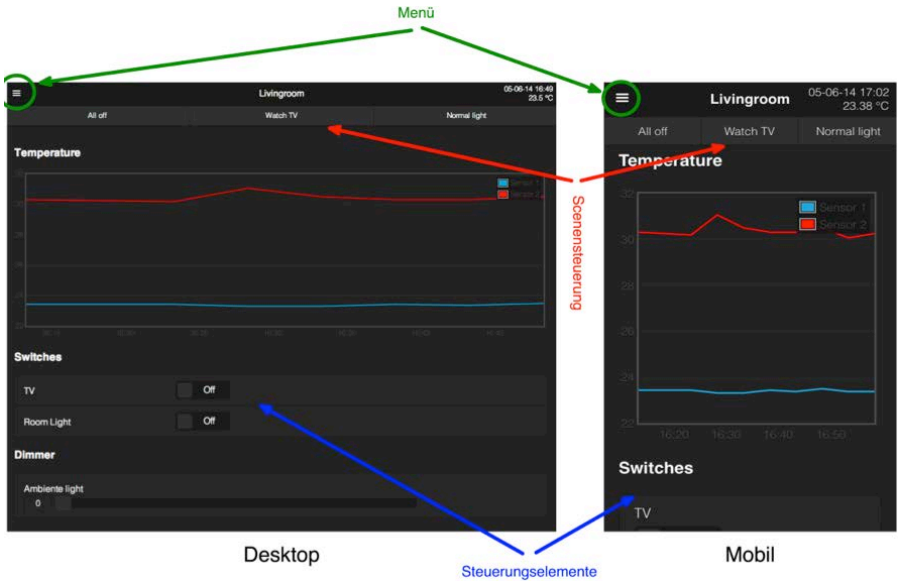


Bedienungsanleitung (Webinterface)

3 Benutzung

3.1 Benutzeroberfläche

In der Mitte am schwarzen oberen Rand befindet sich der Titel der aktuellen Seite. Oben Rechts, ebenfalls auf dem Schwarzen Rand, befinden sich aktuelle Uhrzeit, sowie falls vorhanden Temperatur des dargestellten Raumes. Unterhalb der Schwarzen Leiste werden eventuell konfigurierte Szenen, des dargestellten Raumes angezeigt. Mit einem klick oder fingertab auf eine dieser Szenen, wird diese ausgeführt. Sind einem Raum ein oder mehrere Temperatursensoren zugewiesen, werden diese in Form einer Temperaturkurve (der letzten 8 Stunden) dargestellt.



Zur Steuerung der einzelnen Geräte wurden graphische Bedienelemente (Flip-Switches und Slider) verwendet, die ebenfalls auf die Benutzung mit dem Finger (Touch) oder per Maus optimiert sind.

Bedienungsanleitung (Webinterface)

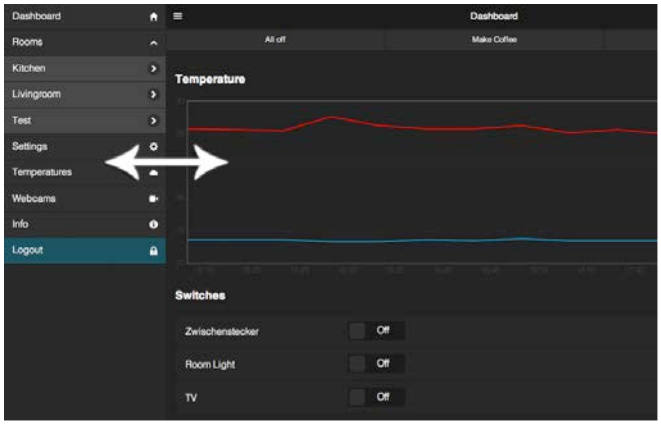
3.2 Dashboard

Das Dashboard ist die Startseite des Systems. Hier können die meistgenutzten Funktionen abgelegt werden.

Hier können Sie dem Dashboard zugewiesene Geräte und Szenen bedienen und sich die Temperaturverläufe von den zugewiesenen Temperatursensoren anzeigen lassen.

3.3 Menü

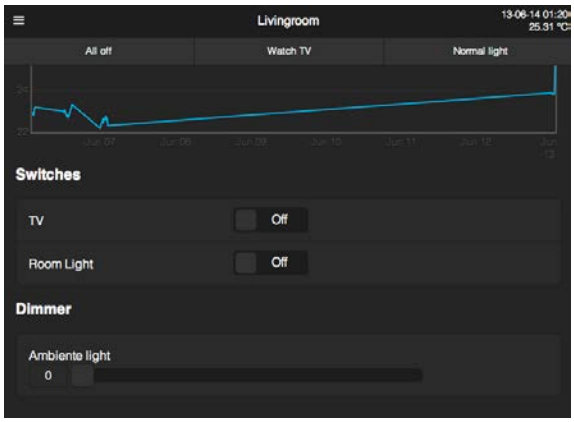
Das Menü wird über das Symbol (drei waagerechte Striche) am oberen linken Rand der Applikation aufgerufen. Es schiebt die restliche Webseite über den rechten Bildschirmrand hinaus und erscheint im linken Bereich des Bildschirms.



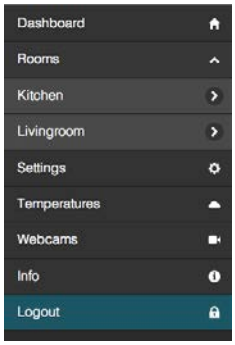
Bedienungsanleitung (Webinterface)

3.4 Räume

Die Virtuellen Räume in TheSmarterPi spiegeln das digitale Abbild der Geräte im Realen Raum wieder. Diese virtuellen Räume sind genauso aufgebaut, wie das auf der Startseite befindliche Dashbaord.

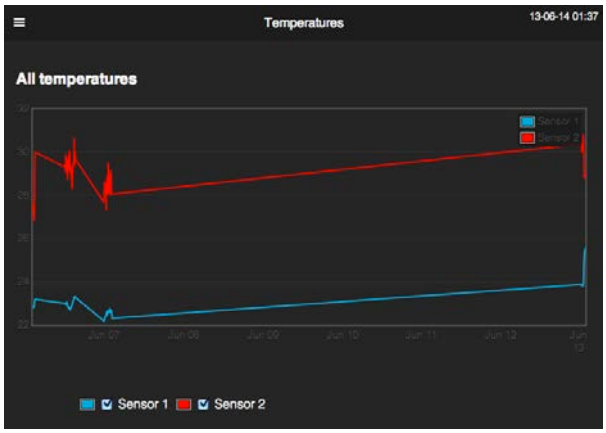


Hier können Sie dem Raum zugewiesene Geräte und Szenen bedienen und sich die Temperaturverläufe (max. 8Std.) von den zugewiesenen Temperatursensoren anzeigen lassen.



Um zwischen den Räumen zu wechseln drücken Sie Im Menü auf **[Rooms]** und wählen den gewünschten Raum.

3.5 Alle Temperaturverläufe



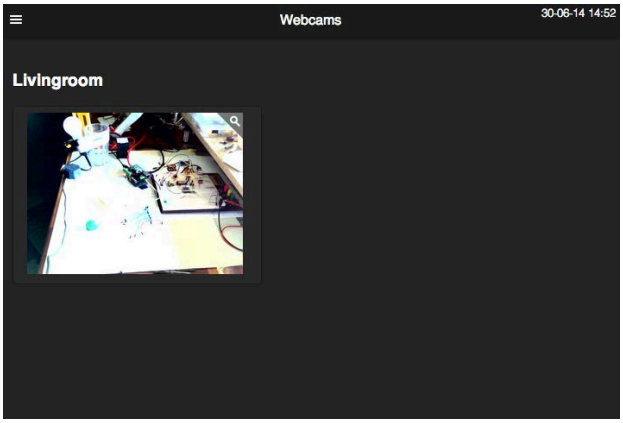
Wählen Sie im Menü [All temperatures] um sich alle Temperaturverläufe, der am System angeschlossenen 1-Wire Sensoren anzuzeigen.

Durch An- oder Abhaken des jeweiligen Sensors, wird dieser aus der Grafik entfernt oder hinzugefügt.

Bedienungsanleitung (Webinterface)

3.6 Webcams

Besitzen Sie eine Webcam, die MJPG-Codierte Video-Streams liefern kann, so haben Sie die Möglichkeit, diese durch das System verschlüsseln zu lassen. Der Stream wird dann über dieselbe verschlüsselte Verbindung durchgereicht, über die auch ihr Webinterface verbunden ist.



Um sich Ihre Webcams anzeigen zu lassen, wählen Sie im Menü **[Webcams].**

All Ihre Webcams werden nun angezeigt.

Fahren Sie mit der Maus oder Ihrem Finger über das Webcam-Bild um es zu vergrößern.

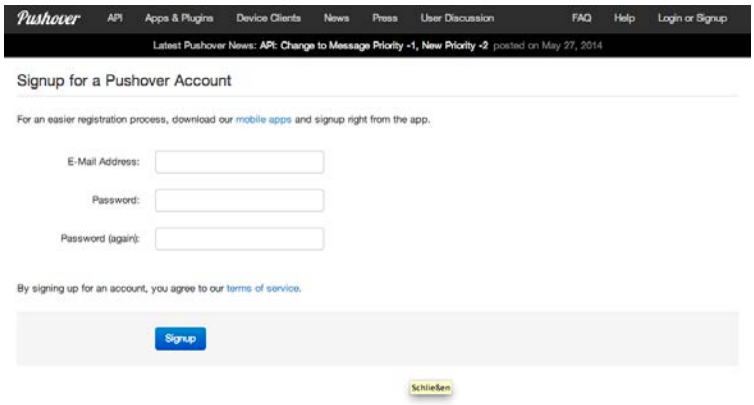
Bedienungsanleitung (Webinterface)

4 Beispielkonfigurationen

4.1 Push Benachrichtigung per Pushover (Android/iOS)

Anlegen eines Benutzerkontos:

Rufen Sie die Webseite www.pushover.net in einem Webbrowser Ihrer Wahl auf. Klicken Sie auf **[Login or Signup]**.



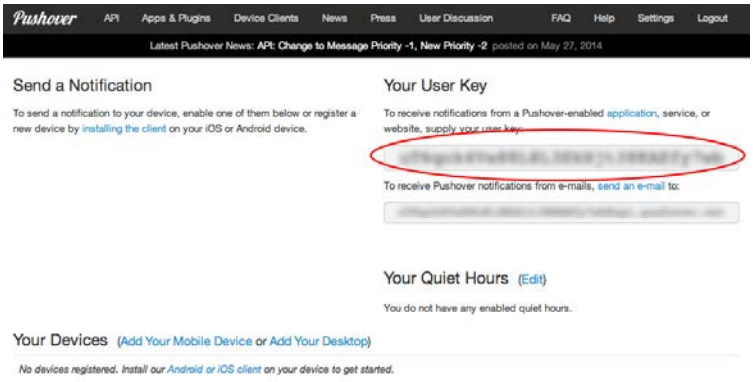
Tragen Sie Ihre E-Mailadresse und Ihr gewünschtes Passwort ein.

Drücken Sie zum Abschluss auf **[Signup]**.

Sie erhalten nun eine E-Mail, mit einem Bestätigungslink. Öffnen Sie diese E-Mail und klicken auf den enthaltenen Link um Ihre E-Mailadresse zu bestätigen.

Ihr Benutzerschlüssel (Userkey)

Im nächsten Fenster (Kontoeinstellungen) sehen Sie Ihren Benutzerschlüssel (rot markiert).

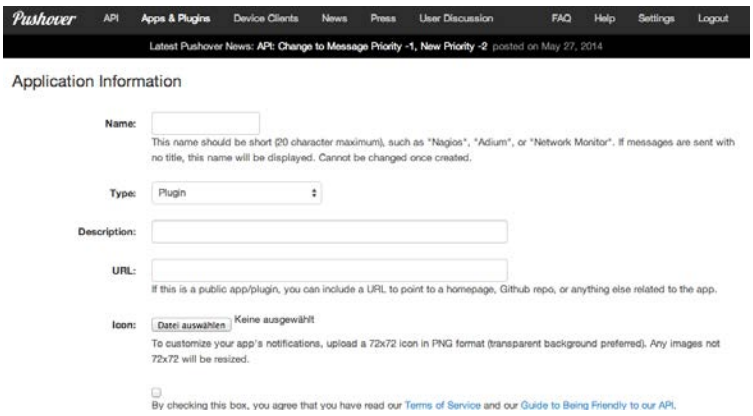


Merken Sie sich den Schlüssel, dieser wird später, bei der Konfiguration des Http Pseudo Devices benötigt.

Bedienungsanleitung (Webinterface)

Anlegen einer Applikation

Klicken Sie im selben Fenster (Kontoeinstellungen), wie eben auf **[Register an Application]** (etwas weiter unten).



Tragen Sie nun den Namen Ihrer Applikation (z.B. TheSmarterPi) ein.

Bei Type wählen Sie „Application“.

Unter Description können Sie eine kurze Beschreibung Ihrer Applikation eintragen.

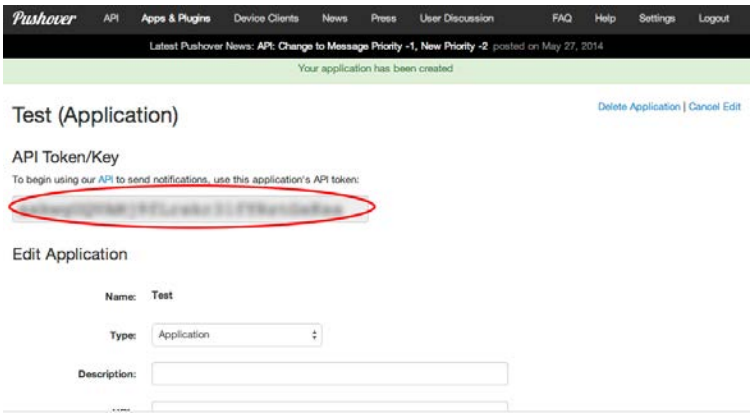
Das Feld URL lassen Sie bitte leer.

Bestätigen Sie mit klicken auf die Checkbox, dass Sie die Nutzungsbedingungen und die API Bestimmungen gelesen haben.

Klicken Sie zum Abschluss auf **[Create Application].**

API Token/Key

Im nächsten Fenster sehen Sie Ihren API Token bzw. Key (rot markiert).



Merken Sie sich den API Token, dieser wird im nächsten Schritt, bei der Konfiguration des Http Pseudo Devices benötigt.

Bedienungsanleitung (Webinterface)

Konfiguration HTTP Pseudo Device
Fügen Sie wie unter 2.2.3 beschrieben ein neues HTTP Pseudo Device hinzu.

Geben Sie nun im Feld unterhalb von **[Device name]** einen Namen für das Gerät ein (hier z.B. Push Benachrichtigung – Klingel).

Unter **[Location]** können Sie den Dienstenamen eintragen (hier z.B. „Pushover“)

Wählen Sie bei [HTTP request mode]: POST

Tragen Sie im Feld URL folgende URL ein:
https://api.pushover.net/1/messages.json

Lassen Sie die Felder Username und Password leer.

Fügen Sie durch drücken des Buttons **[Add]** 3 weitere Argumente hinzu.

Tragen Sie beim 1. Argument folgendes ein:
Argument: **user**
Value: **der zuvor aufgeschriebene Benutzerschlüssel (Userkey)**

Tragen Sie beim 2. Argument folgendes ein:
Argument: **token**
Value: **der zuvor aufgeschriebene API Token/Key**

Tragen Sie beim 3. Argument folgendes ein:
Argument: **message**
Value: **die Nachricht, die auf Ihrem Smartphone angezeigt werden soll.**

Schließen Sie den Vorgang durch drücken auf **[Save]** ab.
Fügen Sie nun das eben erstellte HTTP Pseudo Device wie unter 2.4. beschreiben einer gewünschten Szene hinzu. Wird diese Szene aufgerufen, so wird eine Push Benachrichtigung mit der zuvor eingestellten Nachricht an Ihre Smartphone gesendet.

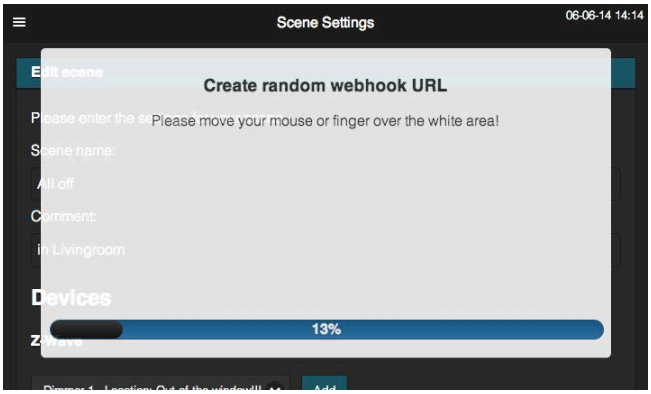
Installieren Sie sich die Pushover App auf Ihrem iOS oder Android Smartphone und loggen sich mit den zuvor erstellten Benutzerdaten ein.

Bedienungsanleitung (Webinterface)

4.2 Anwesenheitserkennung per Geofency (iOS)

Erstellen einer Webhook URL

Fügen Sie wie unter Punk 2.4.1 beschrieben eine neue Szene hinzu. Diese Scene wird später durch Geofency bei Ankunft oder Verlassen des Standorts aufgerufen.



Drücken Sie auf **[Create webhook URL]** und generieren durch bewegen der Maus oder des Fingers innerhalb der weißen Fläche eine Webhook URL.

Schreiben Sie sich die URL auf, diese benötigen Sie später.

Drücken Sie **[Save]** zum Abschließen des Vorgangs.

Einstellung von Geofency

Starten Sie die Applikation Geofency.

Drücken Sie auf **[+]** um einen neuen Ort hinzuzufügen.

Wählen Sie nun, ob Sie die aktuelle Adresse hinzufügen wollen oder wählen Sie eine der anderen Möglichkeiten (Adresssuche, Adressbuch, Geo-Koordinaten oder iBeacon Micro-Location.

Bedienungsanleitung (Webinterface)

Drücken Sie auf das **[Zahnrad-Symbol]** unterhalb der Standortbezeichnung.

Drücken Sie auf **[Webhook]** am unteren Bildschirmrand.

Drücken Sie auf [**Event->URL Einstellungen]**.

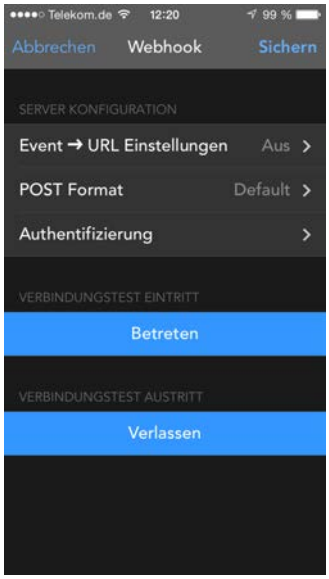
Soll die zuvor erstellte Szene bei Ankunft des Standorts aufgerufen werden, so tragen Sie unter **[Eintritt]** im Feld **[URL]** die zuvor erstellte Webhook-URL ein.

Soll die zuvor erstellte Szene bei Verlassen des Standorts aufgerufen werden, so tragen Sie unter **[Austritt]** im Feld **[URL]** die zuvor erstellte Webhook-URL ein.

Ersetzen Sie **[<your ip or domain>]** durch Ihre IP bzw. Ihre (DynDNS) Domain. Wenn Sie SSL Verschlüsselung nutzen ersetzen sie **http** durch **https**.

Klicken Sie auf **[< Webhook]** um auf die vorherige Seite zu kommen.

Durch drücken auf **[Betreten]** oder **[Verlassen]** können Sie ihre Einstellungen Testen.



Rechtliche Hinweise

H1) Urheberrechtserklärung

Alle in dieser Arbeit verwendeten Fotos und Grafiken wurden eigens erstellt oder erhielten Nutzungserlaubnis durch die jeweiligen Rechteinhaber.

H2) Markenschutzrechtliche Hinweise

Node.js™ ist eine Marke der Joyent Inc.

Chrome™ ist eine Marke von Google Inc.

Pushover ist eine Marke von Superblock, LLC

Firefox® ist ein eingetragenes Markenzeichen der Mozilla Foundation.

Internet Explorer® ist ein eingetragenes Markenzeichen der Microsoft Corporation.

Windows® ist ein eingetragenes Markenzeichen der Microsoft Corporation.

Visio® ist ein eingetragenes Markenzeichen der Microsoft Corporation.

1-Wire® ist ein eingetragenes Markenzeichen von Maxim Integrated Products Inc.

Z-Wave® ist ein eingetragenes Markenzeichen von Sigma Designs und ihren Tochtergesellschaften in den Vereinigten Staaten und anderen Ländern.

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Stade, 07. Juli 2014 _____

Ort, Datum Unterschrift