

OMReS

Open Modular Remote System

Diplomarbeit der Technikerschule Zürich

Diplomand: Pascal Keller

Diplomarbeitsbetreuer: Markus Kohlhaupt

Bearbeitungszeitraum: 17.02.2016 - 30.05.2016

Inhaltsverzeichnis

Abbildungsverzeichnis	5
Tabellenverzeichnis.....	5
1 Einleitung.....	6
1.1 Kurzfassung	6
1.2 Zielsetzung	6
2 Management Summary	7
2.1 Aufgabenstellung	7
2.2 Konzept und umgesetzte Lösung	7
2.3 Herausforderungen und Schwierigkeiten	8
3 Projektmanagement.....	9
3.1 Zeitplanung	10
3.2 Risiken.....	11
4 Engineering	12
4.1 Anwendungsszenarien.....	12
4.2 Anforderungen.....	12
4.2.1 Grundsätzliche Anforderungen	13
4.2.2 Anforderungen an die Hardware	13
4.2.3 Anforderungen an die Bedienung	14
4.2.4 Anforderungen an die Datenkommunikation.....	15
4.3 Konzeptübersicht	15
4.4 Architektur	16
4.4.1 Hardware Fernsteuergerät	16
4.4.2 Software	17
4.5 Datenkommunikation	20
4.5.1 Fernsteuergerät - Server.....	20
4.5.2 Frontend - Server.....	22
4.6 Sicherheit	23
4.6.1 Angriffspunkte	23
4.6.2 Umsetzung.....	24
4.7 Testfälle	28
5 Evaluierung	30
5.1 Hardware	30
5.1.1 Arduino Plattform.....	30
5.1.2 Raspberry Pi.....	32
5.1.3 Tinkerforge	33
5.1.4 ausgewählte Plattform	33
5.1.5 Nutzwertanalyse.....	34

5.2	Frontendtechnologien	36
5.3	Servertechnologien.....	37
5.4	Entwicklungswerzeuge.....	37
5.5	Mobilfunkanbieter.....	38
5.6	Serverhosting	39
6	Hardware	40
6.1	Schnittstellen	40
6.2	Zusammenbau Prototyp	42
7	Implementierung.....	43
7.1	Software Mikrokontroller	43
7.1.1	Programmierumgebung	43
7.1.2	Programmstruktur.....	43
7.1.3	Schwierigkeiten und Herausforderungen	51
7.1.4	Programmbibliotheken.....	52
7.2	Server.....	53
7.2.1	Datenbank	53
7.2.2	Programmstruktur.....	54
7.2.3	Schwierigkeiten und Herausforderungen	64
7.2.4	Programmbibliotheken.....	65
7.3	Frontend	66
7.3.1	JavaScript.....	66
7.3.2	Layout und Navigation.....	71
7.3.3	Design der Oberfläche.....	72
7.4	nicht implementierte Funktionalitäten.....	73
8	Bedienung der Benutzeroberfläche	76
8.1	Schalten der Ausgänge.....	76
8.2	Geräteeinstellungen.....	76
8.3	Kanaleinstellungen.....	77
9	Schlusswort und Ausblick.....	79
10	Anhang	80
10.1	Testprotokolle	80
10.2	Arbeitsjournal	82
10.3	Lernjournale.....	84
10.3.1	Woche 1.....	84
10.3.2	Woche 2.....	85
10.3.3	Woche 3.....	86
10.3.4	Woche 4.....	87
10.3.5	Woche 5.....	88
10.3.6	Woche 6.....	89
10.3.7	Woche 7.....	89

10.3.8	Woche 8.....	90
10.3.9	Woche 9.....	90
10.3.10	Woche 10.....	91
10.3.11	Woche 11.....	91
10.4	Besprechungsprotokolle	92
10.5	Disposition und Auftrag	96
10.6	Ausführungsbestimmungen.....	103
Glossar.....		113
Quellenverzeichnis		113
Literaturverzeichnis		114
Softwareverzeichnis.....		115

Abbildungsverzeichnis

Abbildung 1	Vergleich Aufwände.....	11
Abbildung 2	Das System in der Übersicht.....	15
Abbildung 3	Architektur Hardware Fernsteuergerät	16
Abbildung 4	Übersicht Softwarekomponenten des Systems.....	19
Abbildung 5	Socket Verbindungen mit dem Server.....	22
Abbildung 6	Let's Encrypt Zertifikat.....	24
Abbildung 7	Sicherheitsarchitektur Server – Client.....	25
Abbildung 8	Sicherheitsarchitektur Server – Fernsteuergerät.....	27
Abbildung 9	Die Arduino Plattform [5]	30
Abbildung 10	Raspberry Pi 3 Model B [10]	32
Abbildung 11	Mehrere Tinkerforge Module zusammengesteckt [12].....	33
Abbildung 12	Datenverkehr bei einer normalen Serveranfrage.....	38
Abbildung 13	Klemmanschlüsse für die analogen Eingänge	41
Abbildung 14	Programmablauf vom Fernsteuergerät	44
Abbildung 15	Debug-Konsole des Arduino	45
Abbildung 16	Datenbankmodell	53
Abbildung 17	Notation des Sequenzdiagramms.....	55
Abbildung 18	Sequenzdiagramm Fernsteuergerät zu Server.....	56
Abbildung 19	Sequenzdiagramm Authentifizierung und Registrierung	61
Abbildung 20	Sequenzdiagramm Server zu Frontend.....	66
Abbildung 21	Sequenzdiagramm Click-Events	69
Abbildung 22	Übersicht Layout Benutzeroberfläche	71
Abbildung 23	Vergleich Responsive Webdesign	72
Abbildung 24	Geräteeinstellungen	75

Tabellenverzeichnis

Tabelle 1	Projektplan	10
Tabelle 2	Aufbau des Datenblocks für die gesendeten Daten	21
Tabelle 3	Aufbau des Datenblocks für die empfangenen Daten.....	21
Tabelle 4	Technische Eigenschaften Arduino Uno [5]	31
Tabelle 5	Technische Eigenschaften Arduino 101 [6].....	31
Tabelle 6	Auswertung der Nutzwertanalyse	35
Tabelle 7	Testaufbau der Hardware	42
Tabelle 8	Testprotokoll Fernsteuergerät	80
Tabelle 9	Testprotokoll Frontend	80
Tabelle 10	Testprotokoll Server.....	81
Tabelle 11	Testprotokoll Fernsteuergerät - Server	81
Tabelle 12	Testprotokoll Frontend – Server	81
Tabelle 13	Testprotokoll Systemtest.....	82
Tabelle 14	Arbeitsjournal	83

1 Einleitung

1.1 Kurzfassung

In dieser Dokumentation wird die Entwicklung und Umsetzung eines Systems beschrieben, mit dem es möglich ist standortunabhängig elektrische Anlagen fernzusteuern und zu überwachen. Das System besteht aus einem Fernsteuergerät, welches sich über das Mobilfunknetz mit dem Internet verbindet, einem Server welcher ans Internet angeschlossen ist und einer Webapplikation welche vom Endnutzer auf einfache Weise bedient werden kann. Das System basiert auf Open Source Hard- sowie Software.

In der Beschreibung des Systems wird insbesondere auf die Datenkommunikation und das Zusammenspiel der einzelnen Teilsysteme eingegangen. Dabei geht es auch um Verschlüsselungstechnologien und die Sicherheitsaspekte im Bereich des «Internet of Things».

Schlagwörter: Remote System, WebSocket, Mikrokontroller, node.js, Datenkommunikation, Sicherheit, Verschlüsselung.

1.2 Zielsetzung

Das Ziel dieser Diplomarbeit ist es, ein funktionierendes System zu entwickeln sowie einen Prototyp herzustellen. Das Gerät soll in der Lage sein durch Fernsteuerung andere elektrische Geräte oder Anlagen anzusteuern, sowie zu überwachen. Die Fernsteuerung erfolgt über eine Weboberfläche, welche vom Endnutzer mit einem handelsüblichen Computer, einem Tablet oder dem Smartphone bedient werden kann. Die Bedienung soll möglichst einfach sein.

Ein weiteres Ziel ist eine saubere Dokumentation, welche das System verständlich erklärt und auch die Herausforderungen und Schwierigkeiten bei der Umsetzung aufzeigt.

2 Management Summary

2.1 Aufgabenstellung

Es soll ein System entwickelt werden, welches elektrische Anlagen und Geräte standortunabhängig steuern kann. Die verwendete Hard- und Software soll dabei quelloffen sein. Die Bedienung soll für den Endnutzer möglichst einfach und verständlich sein.

Mussziele

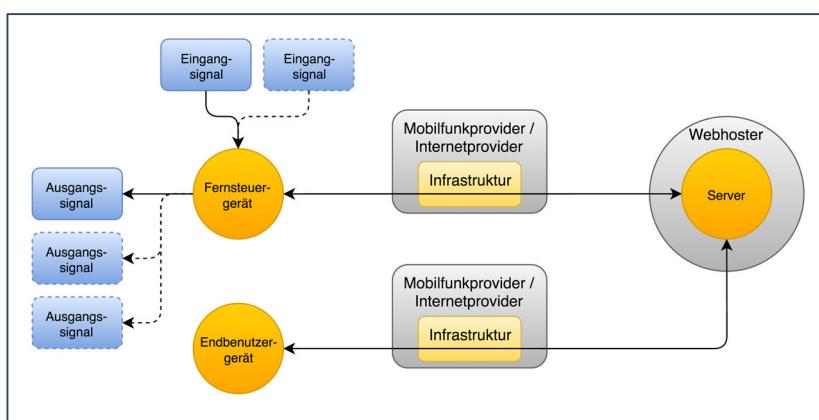
- funktionierender Prototyp mit Datenkommunikation über das Mobilfunknetz
- Bedienung mit Webbrowser auf dem Smartphone möglich
- einlesen von einem analogen Signal möglich
- schalten von mindestens einem elektrischen Gerät möglich

Kannziele

- gut designte Weboberfläche
- Datenkommunikation mittels Ethernet Schnittstelle und Wi-Fi
- einlesen und schalten von mehreren Signalen und Geräten

2.2 Konzept und umgesetzte Lösung

Das System besteht aus drei Teilen: Einem Server, welcher mit node.js umgesetzt ist. Einem Fernsteuergerät, welches auf der Arduino Plattform basiert und sich über das Mobilfunknetz mit dem Server verbindet sowie einer Webapplikation, welche mit JavaScript und dem Bootstrap CSS-Framework umgesetzt wurde.



Der Prototyp erfüllt bis auf die Kommunikation über Wi-Fi alle Muss- und Sollziele. Des Weiteren konnten noch einige weitere Features umgesetzt werden:

- Es können die Werte von 6 analogen Eingängen auf der Weboberfläche angezeigt, sowie 6 digitale Ausgänge geschalten werden.
- Zusätzlich können verschiedene Schaltmodi konfiguriert werden. Dadurch können die Ausgänge je nach Wert der analogen Eingänge automatisch geschalten werden.
- Die Datenkommunikation findet durchgehend verschlüsselt statt und entspricht somit heute gängigen Standards.
- Das übermittelte Datenvolumen zwischen Server und Fernsteuergerät ist so gering, dass ein Dauereinsatz über das Mobilfunknetz mit keinen hohen Kosten verbunden ist.

2.3 Herausforderungen und Schwierigkeiten

Bei dieser Diplomarbeit habe ich einige Herausforderungen meistern können. Die grössten Schwierigkeiten bereitete mir dabei die Implementation des Mikrocontrollers. Insbesondere auch, weil die Entwicklungsumgebung für die Arduino Plattform nicht sehr ausgereift ist und dadurch die Fehlerfindung erschwert wurde. Ausserdem war es für mich das erste wirkliche Projekt bei dem ein Mikrocontroller eingesetzt wurde. In diesem Bereich habe ich aber dadurch auch viel Neues gelernt.

Neue Erkenntnisse konnte ich auch im Bereich der Verschlüsselung und bei den Authentifizierungsmechanismen gewinnen. Eine Unklarheit bei einer Verschlüsselungsmethode im Server hielt mich dabei auch einige Zeit auf.

Im Bereich des Servers habe ich vor allem Erfahrungen mit der asynchronen Programmierung und Callback Funktionen sammeln können. Ich kenne nun einige Tücken und Eigenheiten, welche ein asynchroner Code mit sich bringt und weiss nun wie man damit umgeht.

Schlussendlich habe ich noch den Umgang mit WebSocket gelernt und weiss nun wie man eine bidirektionale Kommunikation zwischen Server und Frontend implementiert.

Es war eine spannende sehr vielseitige Arbeit mit grossem Lerneffekt. Gerade das Zusammenspiel von den einzelnen Komponenten hat mir Freude bereitet.

3 Projektmanagement

Da bei dieser Diplomarbeit in vielen Bereichen Erfahrung fehlte, war es nicht ganz einfach die Aufwände der einzelnen Arbeiten einzuschätzen. Insbesondere gab es einige Unsicherheiten bezüglich der Umsetzung der Software beim Fernsteuergerät. Da es bei Projekten in der Softwareentwicklung nicht sinnvoll ist streng nach Wasserfallprinzip vorzugehen wurde ein Vorgehen mit iterativem Charakter gewählt. Es wurde dabei aber darauf verzichtet auf eines der bekannten Vorgehensmodelle wie Scrum oder Kanban zu setzen. Dies vor allem, weil diese Diplomarbeit ein Ein-Mann-Projekt ist und die Erfahrungen in diesen Vorgehensmodellen fehlen.

Im Groben sah der Ablauf des Projekts wie folgt aus:

- Vorbereitung der Dokumentation und der Planung
- Recherche von Technologien sowie Erarbeitung der grundsätzlichen Konzepte und Architekturen
- Zusammenbau des Prototyps
- Implementierung von Server und Mikrokontroller für einen Kommunikationstest
- weitere Recherche, Evaluierung, Architekturausarbeitung, Implementierung und Dokumentation im Wechsel
- Dokumentation der Implementierung
- Abschluss der Dokumentation

Im Arbeitsjournal, welches sich im Anhang befindet, sind die Arbeitsschritte detaillierter aufgeführt.

3.1 Zeitplanung

Projektplan

Tätigkeit	Aufwand Soll	Aufwand Ist
Planung	58:00	59:30
Allgemeine Vorbereitung	4:00	3:00
Projektplan erstellen	3:00	2:00
Konzept / Architektur entwerfen	15:00	13:00
Informationsbeschaffung Technologien	12:00	14:45
Analyse	6:00	4:30
Evaluation	10:00	10:45
Projektmanagement allgemein	8:00	11:30
Dokumentation	83:00	104:45
Vorbereitung Dokumentation	4:00	6:00
Dokumentation schreiben	70:00	95:00
Zusatzdokumente einpflegen	3:00	1:00
Rechtschreib- und Formatierungsprüfung	4:00	2:00
Vorbereitung zum Druck	2:00	0:45
Hardware	18:00	4:15
Beschaffung Hardware	4:00	1:00
Zusammenbau Prototyp	10:00	1:45
Abklärung und Beschaffung SIM Karte	4:00	1:30
Tests	5:00	5:45
Komponententests	2:00	1:30
Systemtests	2:00	2:45
Akzeptanztest	1:00	1:30
Software	135:00	151:15
Vorbereitung Entwicklungsumgebungen	4:00	3:15
Design Mockup Web Frontend	6:00	4:45
Implementierung Web Frontend	35:00	23:15
Implementierung Server Backend	45:00	62:45
Implementierung Mikrokontroller	45:00	57:15
Total:	299:00	325:30

Tabelle 1 Projektplan

Abweichungen

Im Laufe des Projekts wurde klar, dass für die Implementation im Bereich des Mikrocontrollers und des Servers mehr Zeit als angenommen benötigt wird. Im Bereich Hardware hingegen wurden einige Stunden dafür eingeplant den Testaufbau zu erweitern. Nach Absprache mit dem Diplomarbeitsbetreuer wurde schlussendlich auf die Ausführung dieser Arbeit verzichtet. Da es bei dieser Diplomarbeit vorwiegend um Softwareentwicklung geht hat die Implementation auch eine höhere Relevanz als der Zusammenbau der Hardware.

Der Aufwand für die Dokumentation wurde auch unterschätzt. Es wurde am Schluss noch einmal viel Zeit in die Ausarbeitung der Dokumentation gesteckt und einige Kapitel wurden hinzugefügt, erweitert sowie überarbeitet.

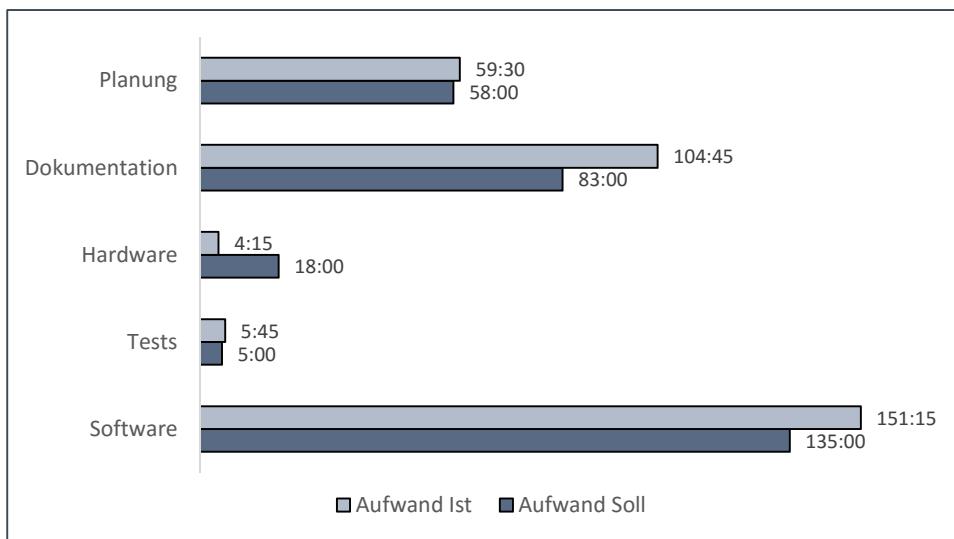


Abbildung 1 Vergleich Aufwände

3.2 Risiken

Für das Projekt bestehen einige bekannte Risiken, welche den Aufwand des Projekts erhöhen können, die Qualität der Lösung beeinträchtigen oder schlimmstenfalls das Fertigstellen des Prototyps im Zeitrahmen der Diplomarbeit verunmöglichen können.

Kommunikation zwischen Fernsteuergerät und Server über das Mobilfunknetz

Eine funktionierende und reibungslose Kommunikation zwischen den beiden Komponenten ist von einigen Faktoren abhängig. Der Mobilfunkanbieter muss entsprechende Angebote anbieten mit denen man sich über die verwendete Hardware in das entsprechende Netz verbinden kann. Welche Angaben dazu benötigt werden und ob sie vom Anbieter zur Verfügung gestellt werden ist unsicher. Es wird auch eine SIM Karte für die verwendete Hardware benötigt, welche ebenfalls vom Anbieter zur Verfügung gestellt werden muss.

Ein weiterer Faktor ist die Kompatibilität der gewählten Technologien wie zum Beispiel die Kommunikationsprotokolle, beziehungsweise die entsprechenden Programmbibliotheken für das Fernsteuergerät und den Server. Wie viel Arbeitsspeicher benötigt die schlussendliche Implementation auf dem Fernsteuergerät? Sind die gewählten Kommunikationstechnologien mit vernünftigen Aufwand implementierbar? Das sind die Unsicherheiten und Fragen welche sich zu diesem Thema stellen.

Schwierigkeiten bei der Implementation

Vor allem im Bereich des Fernsteuergeräts, welches voraussichtlich mit einem Mikrokontroller betrieben wird, fehlt die Erfahrung und das Know-how. Wie gut man in diesem Bereich voran kommt hängt einerseits von der Qualität der Dokumentationen, sowie den eigenen Erfahrungen bei der hardwarenahen Programmierung in den entsprechenden Programmiersprachen ab.

4 Engineering

4.1 Anwendungsszenarien

Ferienhausbesitzer

Ein Vater möchte das Wochenende mit seiner Familie in seinem Ferienhaus im entfernten Wintersportgebiet verbringen. Um Strom zu sparen soll die Heizung im Ferienhaus nicht die ganze Zeit eingeschaltet bleiben. Damit die Familie aber bei Ankunft nicht erst 3 Stunden lang frieren muss will der Vater die Heizung vorgängig standortunabhängig einschalten können. Ausserdem möchte er verhindern das die Sanitäranlagen wegen gefrorenem Wasser Schaden nehmen. Desweiteren soll bei der installierten Solaranlage der Zustand des Akkus sowie die Energieproduktion überwacht werden können.

Produktionsleiter

Ein Produktionsleiter einer kleinen Fabrik möchte den Zustand seiner Maschinen, welche über das Wochenende laufen, überwachen und benachrichtigt werden, falls Maschinen ihren Betriebszustand ändern.

Hausbesitzer

Ein Hausbesitzer möchte von seinem Automobil aus das Garagentor bedienen können und temporär Bekannten und Freunden Zugang zum Haus gewähren.

Geocacher

Ein Geocacher hat nur Zugriff auf den versteckten Schatz, wenn er mit seinem Handy auf einer Webseite die richtige Schalterkombination eingibt.

4.2 Anforderungen

Einige Anforderungen an das Produkt entstanden aus der Disposition der Diplomarbeit. Die anderen Anforderungen entstanden vorwiegend aus der Analyse der Anwendungsszenarien. Ausserdem bestehen einige weitere nicht funktionale Anforderungen, welche für ein Produkt dieser Art möglichst immer gut erfüllt werden sollen. Zu bemerken ist noch, dass man im Rahmen einer Diplomarbeit nicht alle Anforderungen erfüllen kann und man sich auf die Wichtigsten beschränken soll. Auch die Priorisierung einzelner Anforderungen kann anders aussehen wie bei einer echten Produktentwicklung.

Des Weiteren wird nicht explizit zwischen funktionalen und nicht funktionalen Anforderungen unterschieden. Es werden sowohl das Was sowie auch das Wie in den einzelnen Anforderungen berücksichtigt.

4.2.1 Grundsätzliche Anforderungen

Basierend auf Open Source Hardware und Software

Die Layoutpläne und elektrischen Schemas für die Komponenten des Fernsteuergeräts müssen frei verfügbar sein sowie lizenziert benutzt werden dürfen. Dies gilt auch für jeglichen Quellcode der Komponenten sowie für Server- und Frontendcode.

Modulares System

Die einzelnen Komponenten des Systems müssen voneinander optimal entkoppelt sein. Die Schnittstellen müssen klar definiert werden.

Gute Usability

Ein Endnutzer welcher das System nur bedient braucht keine besonderen technischen Kenntnisse. Für den Zusammenbau und die Konfiguration des Systems sind fundierte technische Kenntnisse in den relevanten Bereichen nötig. Programmierkenntnisse sind dafür aber nicht notwendig.

Vernünftiger Anschaffungspreis

Der Preis für die benötigten Komponenten, sowie Kosten für Webhosting und Mobilfunknutzung müssen im vernünftigen Rahmen liegen.

Schutz gegen Zugriff von Unbefugten

Das System muss vor unbefugtem Zugang geschützt sein. Die eingesetzten Technologien entsprechen dem heutigen Stand der Technik.

4.2.2 Anforderungen an die Hardware

Ethernet- oder Mobilfunk-Konnektivität

Die Hardware muss in der Lage sein die Datenkommunikation über eine für den entsprechenden Einsatzzweck geeignete Schnittstelle abwickeln zu können.

Geringe Stromaufnahme

Damit die Hardware mit Hilfe eines Akkus möglichst lange betrieben werden kann, muss die Stromaufnahme möglichst gering gehalten werden.

Erfassen von mehreren Eingängen

Es müssen mehrere Eingangssignale erfasst werden können. Die entsprechenden Schnittstellen sollen sowohl analoge Werte als auch ein digitales Signal verarbeiten können.

Schalten von mehreren Ausgängen

Das System muss mehrere Ausgangssignale schalten können. Die Ausgänge müssen eine Spannung von bis zu 230VAC vertragen.

Notschaltung

Ausgänge müssen automatisch geschaltet werden falls ein bestimmtes Eingangssignal einen festgelegten Wert erreicht.

4.2.3 Anforderungen an die Bedienung

Komfortable Bedienung vom Smartphone aus

Die Bedienung des Systems soll mit dem Smartphone ohne Probleme möglich sein.

Reaktionszeit entspricht Anwendungszweck

Je nach Anwendungszweck muss das System auf Eingaben mehr oder weniger schnell reagieren können.

Sicheres und passwortgeschütztes Login

Die Authentifizierung und Anmeldung entspricht den gängigen Standards.

Abfrage / Anzeige der Eingangswerte

Die gesammelten Werte der Eingänge von der Hardware müssen angezeigt werden können.

Warnungsmeldungen

Warnungen, welche von der Hardware aus gesendet werden, sollen einerseits direkt angezeigt werden können sowie auch per SMS- oder Push-Nachricht an den Endnutzer direkt gesendet werden können. Auch soll der Endnutzer benachrichtigt werden falls die Verbindung zum Fernsteuergerät unterbrochen ist.

Schalten der Ausgänge

Die Ausgänge der Hardware müssen auf einfachste Weise geschalten werden können.

Umbenennung der Labels für die Werteanzeige und Buttons der Ausgänge

Die Beschreibungen für die einzelnen geschalteten Ausgänge müssen editiert werden können. Auch die Beschreibungen zu den angezeigten Eingangswerten müssen angepasst werden können.

Konfigurationsmöglichkeiten

Die Hardware soll von der Bedienoberfläche aus konfiguriert werden können.

4.2.4 Anforderungen an die Datenkommunikation

Zugriff standortunabhängig möglich

Der Zugriff auf die Bedienoberfläche soll standortunabhängig möglich sein.

Geringe Datenmenge für Übertragung

Die Grösse und Menge der übertragenen Datenpakete von der Hardware aus soll möglichst gering gehalten werden.

4.3 Konzeptübersicht

Das gesamte System besteht aus verschiedenen Komponenten, welche miteinander über verschiedene Datenkanäle und Schnittstellen kommunizieren. Schlüsselkomponente ist das Fernsteuergerät, welches über seine Ausgänge externe elektrische Anlagen ein- und ausschalten oder anderweitig ansteuern kann. Des Weiteren können über die verfügbaren Eingänge analoge sowie digitale Werte eingelesen werden. Als weitere Komponente ist ein Server dafür zuständig sowohl Daten vom Fernsteuergerät zu empfangen und zu senden, als auch eine Weboberfläche für den Endnutzer zur Verfügung zu stellen. Der Endnutzer kann über die Weboberfläche die Ausgänge des Fernsteuergeräts fernsteuern, die vom Fernsteuergerät gesendeten Eingangssignale auslesen sowie die Ein- und Ausgänge als auch weitere Einstellungen konfigurieren.

Die Datenkommunikation der verschiedenen Komponenten wird über das Internet abgewickelt. Auf Seite des Fernsteuergeräts wird der Zugang zum Netz über das Mobilfunknetz gewährleistet. Falls am Standort des Fernsteuergeräts schon eine Netzwerkinfrastruktur besteht, kann auch diese als Zugang zum Internet genutzt werden. Der Zugriff zum Server wird von einem Webhost oder Server Provider sichergestellt. Zugriff auf die Weboberfläche findet wie gewohnt über den Zugang des Internet Providers, oder aber auch mit dem Smartphone über das Mobilfunknetz statt.

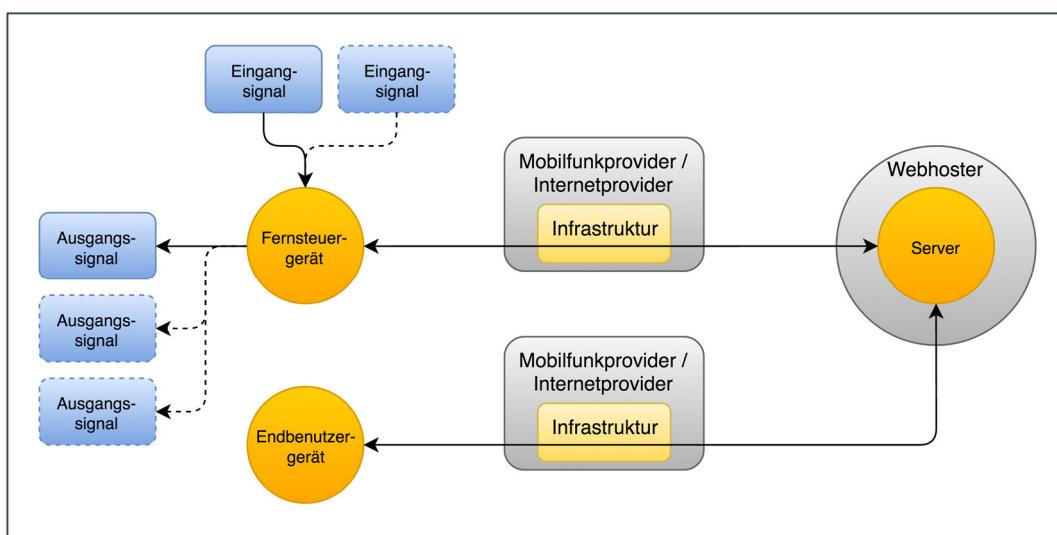


Abbildung 2 Das System in der Übersicht

4.4 Architektur

4.4.1 Hardware Fernsteuergerät

Das Fernsteuergerät besteht aus den folgenden sich voneinander abgrenzbaren Komponenten:

Mikrokontroller

Die Hauptaufgabe des Mikrocontrollers ist nebst der Prozessverarbeitung das einlesen der Eingangssignale, das schalten der Ausgangsignale sowie das empfangen und senden der Daten über die für den Anwendungszweck geeignete Schnittstelle.

Relais

Um externe elektrische Komponenten oder Signale schalten zu können werden Relais benötigt. Über die Anschlüsse der Relais lassen sich unabhängige Signale mit Spannungen bis zu 230VAC schalten.

Klemmenanschlüsse Eingangssignale

Um den Anschluss für die digitalen sowie analogen Werte zu vereinfachen werden diese vom Mikrocontroller auf separate Klemmanschlüsse geführt.

Adapter für die Datenkommunikation:

Da je nach Anwendungszweck für die Datenkommunikation andere Verbindungsadapter benötigt werden, sollen diese modular austauschbar sein. Daher müssen Schnittstellen für die Datenkommunikation sowie die Spannungsversorgung vorhanden sein.

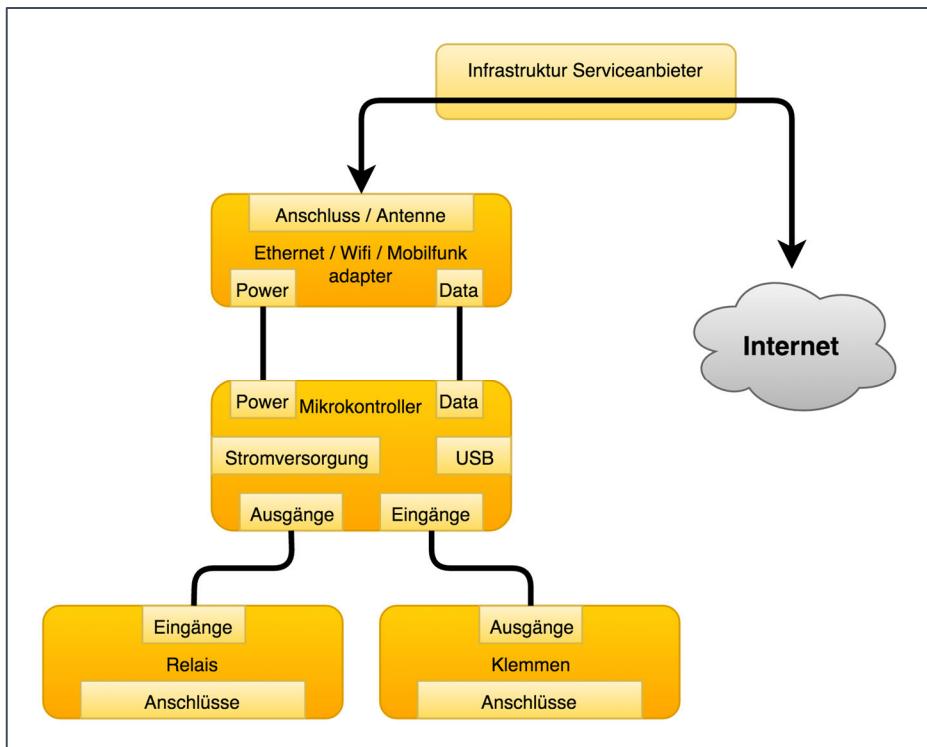


Abbildung 3 Architektur Hardware Fernsteuergerät

4.4.2 Software

4.4.2.1 Aufgaben Fernsteuergerät

Inputwerte verarbeiten

Signale, welche an den Eingangs-Schnittstellen ankommen, müssen in geeignete Werte umgewandelt und zwischengespeichert werden. Falls die Werte einen festgelegten Wert erreichen, müssen entsprechend der Logik, welche in der Konfiguration des Geräts festgelegt ist, Ausgänge geschalten werden.

Ausgänge schalten

Entsprechend der konfigurierten Logik, aber auch entsprechend den empfangenen Befehlen vom Server, werden die Ausgänge ein oder ausgeschalten.

Daten an Server senden und vom Server empfangen

Grundsätzlich sendet nur das Fernsteuergerät Nachrichten an den Server. In der Nachricht sind alle Nutzdaten, welche übermittelt werden sollen enthalten. In den Nutzdaten der Antwortnachricht vom Server befinden sich die Konfigurationsdaten sowie die Steuerbefehle für die Ausgänge.

Die Eingangswerte im Zwischenspeicher, sowie die Zustände der Ausgänge werden an den Server gesendet. Die Häufigkeit der Datenübermittlung hängt von der Konfiguration ab.

Gerät konfigurieren

Beim Startvorgang des Geräts wird die Grundkonfiguration geladen. Diese enthält die Daten für eine korrekte Verschlüsselung sowie die eindeutige ID des Geräts. Auch sind Informationen für den Verbindungsaufbau ins Netz enthalten. Diese Werte sind statisch gespeichert und können nur durch frisches komplizieren und laden des Programmcodes auf das Gerät geändert werden. Konfigurationseinstellungen werden in einem nicht flüchtigen und beschreibbaren Speicher abgelegt. Diese Einstellungen werden auch beim Start des Geräts geladen. Diese Daten enthalten Informationen für die Sendekadenz sowie die Grenzwerte und den Schaltmodus der einzelnen Ein-/Ausgänge. Werden Daten empfangen bei denen geänderte Konfigurationseinstellungen enthalten sind, werden diese übernommen und auch im nicht flüchtigen Speicher gesichert.

Verschlüsselung der Datenkommunikation

Da die Ressourcen auf dem Fernsteuergerät beschränkt sind, ist es nicht möglich ein Kommunikationsprotokoll zu verwenden, welches eine asymmetrische Verschlüsselungsmethode implementiert hat. Die Nutzdaten werden deshalb mit einem symmetrischen Verschlüsselungsverfahren verschlüsselt.

Verifizierung der ankommenden Daten

Um sicher zu stellen, ob die ankommenden Daten nach der Entschlüsselung korrekt sind, werden diese verifiziert. Falls die empfangenen Daten nicht korrekt erfasst wurden, werden die Sendedaten nochmals übermittelt.

4.4.2.2 Aufgaben Server

Senden und empfangen von Daten Fernsteuergerät

Empfangene Daten, welche vom Fernsteuergerät gesendet wurden, müssen verifiziert werden, die Nutzdaten verarbeitet und in die Datenbank gespeichert werden sowie danach die entsprechenden Daten von der Datenbank geladen werden und an das Fernsteuergerät gesendet werden.

Senden und empfangen von Daten Frontend

Sobald eine User Authentifizierung stattgefunden hat werden die Daten zwischen dem Frontend und dem Server über eine bidirektionale stetige Verbindung transportiert. Es werden dabei vor allem aktuelle Messwerte ausgelesen und an das Frontend geschickt. Warnmeldungen, die vom Fernsteuergerät gesendet werden, sollen sofort auf der Weboberfläche angezeigt werden. Außerdem sollen die Schaltzustände der Ausgangssignale beim Fernsteuergerät, welche man über die Weboberfläche setzen kann, an den Server gesendet werden und dort in der Datenbank gespeichert werden.

Versenden von Warnmeldungen

Empfangene Warnmeldungen vom Fernsteuergerät sollen über die Weboberfläche, sowie auch per E-Mail, Push Nachricht oder SMS an den Endnutzer geschickt werden.

Authentifizierung von Fernsteuergerät und Frontend Client

Sowohl die User, welche das Frontend benutzen, als auch die Datenpakete, welche vom Fernsteuergerät gesendet werden, müssen authentifiziert, beziehungsweise verifiziert werden.

Jedes einzelne Datenpaket, welches vom Fernsteuergerät gesendet wird, muss verifiziert werden. Beim Frontend Client hingegen wird nach einer einmaligen Authentifizierung ein privater Übermittlungskanal geöffnet und die Nutzdaten wie auch nutzerspezifische Benachrichtigungen werden über diesen Kanal versendet, beziehungsweise empfangen.

User-Registrierung

Neue User sollen sich über die Weboberfläche registrieren können. Es wird ein neues Nutzerkonto angelegt und dem User werden Login Informationen zur Verfügung gestellt um ihr eigenes Fernsteuergerät mit dem Server verbinden zu können.

speichern der Daten in die Datenbank

Logindaten der User, Konfigurationsdaten für das Fernsteuergerät sowie die Eingangswerte und die aktuellen Zustände der Ausgänge für das Fernsteuergerät werden in einer Datenbank gesichert.

4.4.2.3 Aufgaben Frontend

Anmeldung und Registrierung für die Endnutzer

Auf der Oberfläche müssen Schaltflächen und Formulare dargestellt werden um sich zum einen auf der Seite anmelden zu können, und zum anderen um sich auch als neuen Nutzer registrieren zu können.

Konfiguration des Fernsteuergeräts

Es soll ein Menü aufgerufen werden können mit dem verschiedene Einstellungen für das Fernsteuergerät konfigurieren zu können.

Konfiguration der Weboberfläche

Wenn man auf der Seite eingeloggt ist werden Listen für die Anzeige der Eingangswerte sowie Schaltflächen für die Ausgänge angezeigt. Die Listentitel sowie die Namen der Schaltflächen sollen editierbar sein.

Direktes schalten der Ausgänge

Die angezeigten Schaltflächen sollen beim Aktivieren den Zustand wechseln und der entsprechende Schaltzustand soll an den Server gesendet werden.

Anzeige der Eingangswerte

In den Listen für die Eingänge sollen immer die aktuellen Werte, welche vom Server empfangen werden, angezeigt werden.

Anzeigen von Warnmeldungen

Werden Warnmeldungen vom Server an das Frontend gesendet, sollen diese sofort und gut sichtbar dargestellt werden.

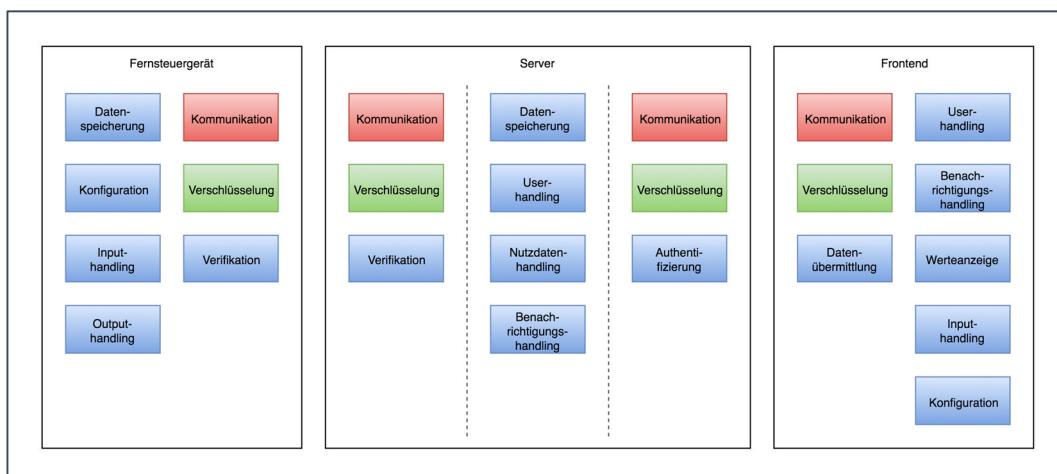


Abbildung 4 Übersicht Softwarekomponenten des Systems

4.5 Datenkommunikation

Die gesamte Kommunikation zwischen den einzelnen Komponenten des Systems wird über das Internet abgewickelt. Genauer gesagt wird zwischen dem Fernsteuergerät und dem Server das HTTP Protokoll eingesetzt. Zwischen dem Frontend und dem Server wird HTTPS eingesetzt um den Content für die Darstellung und die clientseitigen Scripts zu übertragen. Für die weitere Kommunikation zwischen Frontend und Server wird das WebSocket Protokoll eingesetzt.

4.5.1 Fernsteuergerät- Server

Da das Fernsteuergerät je nach Anwendungszweck über eine GSM Verbindung ans Internet angebunden ist, soll die übertragene Datenmenge möglichst gering gehalten werden. Dies wird zum einen dadurch erreicht indem man die Nutzdaten in einzelnen Bytes abbildet, anstatt diese als strukturierter Text in Form von XML / JSON oder ähnlichem zu übertragen. Dies bietet sich sowieso an, da der benutzbare Arbeitsspeicher auf dem Mikrokontroller des Fernsteuergeräts ziemlich eingeschränkt ist. Des Weiteren kann durch die Konfiguration des Fernsteuergeräts bestimmt werden wie oft die Daten an den Server übertragen werden sollen.

Das Fernsteuergerät ist die aktive Komponente, welche in festgelegten Intervallen eine Verbindung zum Server herstellt und die Daten an den Server überträgt. Nachdem die Informationen erfolgreich beim Server angekommen sind, wird vom Server eine Antwortnachricht geschickt. Wurde diese Antwortnachricht korrekt empfangen wird die Verbindung zum Server geschlossen. Es ist also nicht möglich von der Serverseite aus aktiv Informationen zum Fernsteuergerät zu übertragen. Erst beim nächsten Intervall, wenn sich das Fernsteuergerät neu mit dem Server verbindet, können Informationen in der Antwortnachricht wieder übertragen werden. Kann der Server bei der Datenübertragung den Server nicht erreichen, das heisst es folgt keine Antwort auf die Anfrage, wird nach einer bestimmten Zeit eine weitere Anfrage gesendet. Dasselbe gilt auch dann, wenn zwar eine Anfrage empfangen wird, diese aber fehlerhaft ist.

Bei der Übertragung der Daten zwischen dem Fernsteuergerät und dem Server beträgt die Datenmenge der Nutzdaten 64 Byte. Dabei sind die ersten 16 Byte für den Initialisierungsvektor der Verschlüsselung vorgesehen. Bei den Positionen an denen keine relevanten Werte gespeichert werden ist der Wert der entsprechenden Byte Position eingetragen. An den Bytestellen 16, 32 und 48 wird ausserdem geprüft ob die Daten korrekt übertragen wurden. Dies gilt sowohl für die Datenblöcke welche an den Server gesendet werden, sowie auch für die empfangenen Blöcke vom Server.

Bei den gesendeten Datenblöcken folgen danach die Werte der analogen Eingänge, sowie den Zustandsstatus des entsprechenden Ausgangs. Bei den empfangenen Daten sind die unteren wie oberen Grenzwerte für die automatische Ausgangsschaltung, den zugehörigen Schaltmodus, die Schaltbefehle der einzelnen Ausgänge sowie Werte für die Sendekadenz enthalten.

gesendete Daten:

Position:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value:	initialVector															
Position:	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Value:	16	A1 S1 A2 S2 A3 S3 A4 S4 A5 S5														
Position:	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Value:	32	A6 S6 36 37 38 39 40 41 42 43 44 45 46 47														
Position:	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Value:	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

Tabelle 2 Aufbau des Datenblocks für die gesendeten Daten

- **A1 - A6:** Die analogen Werte von den Eingängen an dem Fernsteuergerät. Es werden Werte zwischen 0 und 1023 als unsigned Integer gespeichert und benötigen je 2 Byte Speicherplatz. Die Byte Reihenfolge entspricht Big-Endian.
- **S1 - S6:** Die Ausgangszustände für die entsprechenden Ausgänge. Hat sich der Zustand des Ausgangs durch die automatische Schaltung nicht verändert wird eine 0 übertragen. Wurde ein Grenzwert erreicht und dadurch der Ausgang automatisch ausgeschaltet wird eine 1 übertragen. Wurde ein Grenzwert erreicht und dadurch der Ausgang automatisch eingeschaltet wird eine 2 übertragen.

empfangene Daten:

Position:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value:	initialVector															
Position:	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Value:	16	LT1 HT1 M1 LT2 HT2 M2 LT3 HT3 M3														
Position:	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Value:	32	LT4 HT4 M4 LT5 HT5 M5 LT6 HT6 M6														
Position:	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Value:	48	O1	O2	O3	O4	O5	O6	sendInterval				errorInterval				errCnt

Tabelle 3 Aufbau des Datenblocks für die empfangenen Daten

- **LT/HT1 - LT/HT6:** Der untere (LT / LowTrigger) bzw. obere (HT / HighTrigger) festgelegte Grenzwert für die automatische Schaltung der Ausgänge. Es werden Werte zwischen 0 und 1023 als unsigned Integer gespeichert und benötigen je 2 Byte Speicherplatz. Die Byte Reihenfolge entspricht Big-Endian.
- **M1 - M6:** Der Schaltmodus der entsprechenden Ausgänge für die automatische Schaltung. Es werden Werte zwischen 0 und 7 gespeichert.
- **O1 - O6:** Der Schaltbefehl für die entsprechenden Ausgänge. Wenn der Ausgang nicht geschalten werden soll, wird eine 0 übertragen. Wenn der Ausgang ausgeschalten werden soll, wird eine 1 übertragen. Wenn der Ausgang eingeschalten werden soll, wird eine 2 übertragen.
- **sendInterval:** Die Wartezeit in Millisekunden bevor wieder Daten vom Fernsteuergerät an den Server gesendet werden sollen. Der Wert wird als unsigned long gespeichert welcher 4 Byte Speicherplatz benötigt. Die Byte Reihenfolge entspricht Big-Endian. Der minimal zulässige Wert ist 0.

- **errorInterval:** Die Wartezeit in Millisekunden welche auf eine Antwort vom Server gewartet wird bevor ein weiteres Mal versucht wird die Daten an den Server zu senden. Der Wert wird als unsigned long gespeichert welcher 4 Byte Speicherplatz benötigt. Die Byte Reihenfolge entspricht Big-Endian. Der minimal zulässige Wert ist 4000.
- **errCnt:** Die Anzahl der erfolglosen Sendeversuche an den Server bevor das Fernsteuergerät neu gestartet wird. Es werden Werte zwischen 0 und 255 gespeichert. Der minimal zulässige Wert ist 4.

4.5.2 Frontend- Server

Bei der Kommunikation zwischen Frontend und Server ist die Menge der übertragenen Daten nicht mehr so relevant. Eine direkte stetige Verbindung in beide Richtungen aber schon. Ruft man im Browser die mit dem Server verknüpfte Domain auf werden als erstes die Dateien für die Darstellung des Frontend sowie den dazu gehörigen Programmcode übertragen. Bei der Ausführung des Codes auf der Clientseite wird von Anfang an eine bidirektionale Verbindung mit dem Server aufgebaut. Die weitere Datenübertragung findet dann über diese Verbindung statt. Verbinden sich weitere Web-Clients mit dem Server, nutzen diese denselben sogenannten Socket beim Server. Es ist somit möglich Daten vom Server aus an alle verbundenen Clients zu schicken. Umgekehrt kann natürlich auch jeder Client Daten an den einen Server schicken. Solange sich aber die Clients nicht eindeutig authentifiziert haben, ist es nicht möglich Daten zwischen einem bestimmten Client und dem Server zu übertragen. Erst nach der Authentifizierung kann über die Socket Verbindung der entsprechende User über einen sogenannten Room erreicht werden. So können dann Daten separat über diese einzelnen Rooms übertragen werden. [1]

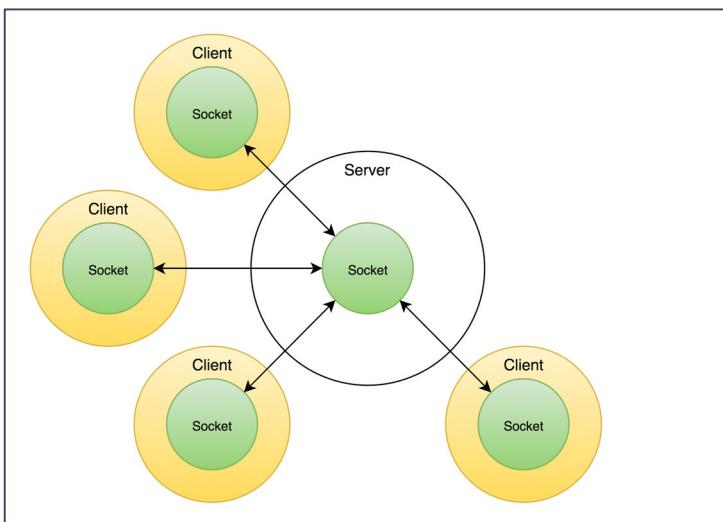


Abbildung 5 Socket Verbindungen mit dem Server

4.6 Sicherheit

Bei diesem Projekt wird die Datenkommunikation über das Internet abgewickelt. Insbesondere wird über das Netz ein Fernsteuergerät angesprochen, welches elektrische Anlagen im eigenen Heim, Ferienhaus oder aber auch unter Umständen Industrieanlagen überwacht und steuert. Gerade wenn zum Beispiel das System verwendet wird um im Eigenheim das Garagentor fernzusteuern, wird das Thema Sicherheit relevant. Auch der Zugang zum Server und die darauf gespeicherten Nutzerdaten sollten gut geschützt sein.

Gerade im Bereich Internet of Things wird leider das Thema Sicherheit öfters vernachlässigt. Nicht selten auch weil es eben gerade bei den schwachen Hardware Komponenten welche in IoT Projekten eingesetzt werden schwierig sein kann Kryptografiealgorithmen zu implementieren. Es ist aber durchaus möglich auch auf schwacher Hardware ein für den Zweck vernünftig hohes Mass an Sicherheit zu erreichen.

4.6.1 Angriffspunkte

Man in the Middle Angriff

Ein Man in the Middle Angriff ist eine Angriffsform welche in Rechnernetzen ihre Anwendung findet. Der Angreifer steht dabei entweder physikalisch oder logisch zwischen den beiden Kommunikationspartnern. Er hat dabei mit seinem System vollständige Kontrolle über den Datenverkehr zwischen zwei oder mehreren Netzwerkteilnehmern und kann die Informationen nach Belieben einsehen und sogar manipulieren. Dabei täuscht er den Kommunikationspartnern vor, das jeweilige Gegenüber zu sein. [2]

Bei diesem System könnte sich beispielsweise ein Angreifer durch einen schwach geschützten Wifi Access Point Zugang ins Heimnetzwerk erhalten. Danach könnte er Datenpakete vom Fernsteuergerät abfangen und eine manipulierte Antwortnachricht zurück an das Fernsteuergerät senden um das Garagentor zu öffnen.

Server Hack

Falls es ein Angreifer schafft sich Zugang zum Server zu verschaffen, kann er die Anmeldedaten der Nutzer aus der Datenbank auslesen. Der Angreifer meldet sich dann mit diesen gestohlenen Nutzerdaten am System an und hat somit vollen Zugriff auf das Fernsteuergerät.

Hardware Hack

Falls ein Angreifer direkten physikalischen Zugriff zum Fernsteuergerät hat, kann dieser ein manipuliertes Programm auf das Gerät laden und hat damit danach auch vollen Zugriff auf das Gerät von aussen.

4.6.2 Umsetzung

Server – Client

Die Hardware, auf dem die Server Software betrieben wird, sollte sich vorzugsweise in einem Datencenter befinden. Die Sicherheit des Betriebssystems auf dem der Server läuft, sowie auch die Sicherheit der Netzwerkinfrastruktur liegt also in der Verantwortung des Betreibers des Datencenters. Falls man sich entscheidet einen kompletten eigenen Server zu mieten oder aber die Server Software im heimischen NAS zu betreiben, muss man selbst dafür sorgen, dass eine grundlegende Sicherheit gewährleistet wird. Unabhängig von Betriebssystem und Standort, sowie deren Sicherheit, gibt es aber trotzdem noch 2 potentielle Angriffspunkte für das System.

Um die Wahrscheinlichkeit eines erfolgreichen Man in the Middle Angriffs stark zu vermindern wird die Datenübertragung zwischen dem Frontend und dem Server verschlüsselt. Dies gilt sowohl für die HTTPS Verbindung als auch für die Übertragung mit dem WebSocket Protokoll. Um einen möglichst reibungslosen Betrieb über die abhörsichere HTTPS Verbindung zu gewährleisten, wird ein gültiges Zertifikat benötigt. Let's Encrypt ist eine noch junge Zertifizierungsstelle welche SSL/TLS Zertifikate ausstellt. Sie unterscheidet sich durch einige Merkmale von anderen üblichen Zertifizierungsstellen. Das beantragen und erstellen von Zertifikaten ist bei Let's Encrypt mit sehr wenig Aufwand möglich. Dabei ersetzt ein automatisierter Prozess die bisher gängigen komplexen händischen Vorgänge bei der Erstellung, Validierung, Signierung, Einrichtung und Erneuerung von Zertifikaten für verschlüsselte Websites. Der Dienst bietet für viele verschiedene Plattformen CLI Tools an um mit ein paar wenigen Kommandozeilenbefehlen ein Zertifikat erstellen zu können. Außerdem, und auch ein Argument welches sehr für den Dienst spricht, sind die ausgestellten Zertifikate völlig kostenlos. Auch ist es möglich mit Let's Encrypt Zertifikate für dynamische Domains ausstellen zu lassen. [3]

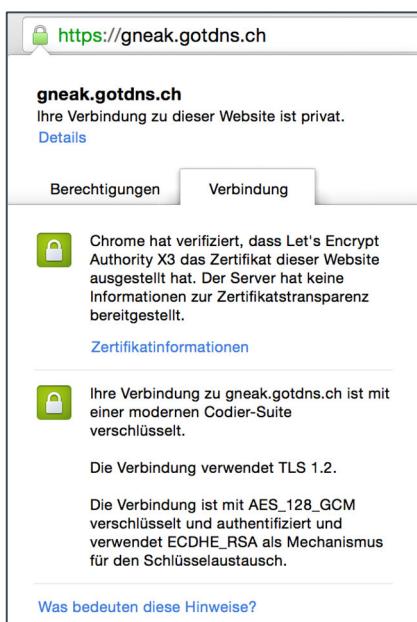


Abbildung 6 Let's Encrypt Zertifikat

Das gleiche Schlüsselpaar, welches für die HTTPS Verbindung verwendet wird, kann auch für eine sichere WebSocket Verbindung verwendet werden. Der gesamte Datenaustausch, welcher zwischen dem Server und dem Frontend stattfindet, ist somit durch eine starke Verschlüsselung abhörsicher.

Im Falle eines Serverhacks bei dem ein Angreifer Zugriff auf den Programmcode sowie die Datenbank erhält, sollte der angerichtete Schaden möglichst geringgehalten werden. Dabei geht es vor allem um die gespeicherten Nutzerdaten. Deshalb werden in diesem System die Passwörter in Form eines salted Hash abgespeichert. Der Salt für den Passwort Hash wird während der Userregistrierung dynamisch

erzeugt und im Datensatz des Users abgespeichert. Zusätzlich zum Salt wird auch noch ein Schlüssel generiert, welcher für die Datenkommunikation zwischen dem Server und dem Fernsteuergerät benötigt wird. Die Nutzdaten, welche Konfigurationsinformation sowie gespeicherte Eingangswerte des Fernsteuergeräts enthalten, sind als Klartext in der Datenbank abgespeichert. Für die eindeutige Identifikation der User wird eine E-Mail-Adresse verwendet. Bei einem erfolgreichen Angriff erhält der Angreifer zwar Zugriff auf gültige Email Adressen und zugehörige Passwort Hashs, kann aber aus den Hashs nur sehr schwer das ursprüngliche Passwort rekonstruieren. Jedoch ist auch der Schlüssel für die Kommunikation mit dem Fernsteuergerät in der Datenbank gespeichert. Nach einem Serverhack müssen also diese Schlüssel alle neu generiert werden und das Fernsteuergerät neu konfiguriert werden. Ausserdem werden auch die Passwörter neu generiert und alle User werden informiert und aufgefordert ein neues Passwort zu wählen. [4]

Das System verwendet einen einfachen aber wirksamen Authentifizierungsmechanismus. Beim Login werden die E-Mail-Adresse sowie das zugehörige Passwort im Klartext über die verschlüsselte WebSocket Verbindung zum Server übertragen. Danach wird in der Datenbank nach dem Datensatz gesucht, welcher der Email Adresse entspricht. Es wird der Salt für den User ausgelesen und dazu verwendet um zusammen mit dem übertragenen Passwort einen Hash zu generieren. Ist der generierte Hashwert äquivalent zu dem welcher im Datensatz des entsprechenden Users gespeichert ist, ist die Authentifizierung erfolgreich. Ansonsten wird dem User eine Fehlermeldung angezeigt. Nur nach einer erfolgreichen Authentifizierung können Meldungen vom Client an den Server gesendet werden, beziehungsweise werden nur dann die Meldungen entgegengenommen und verarbeitet. Auch gilt dies für die andere Richtung.

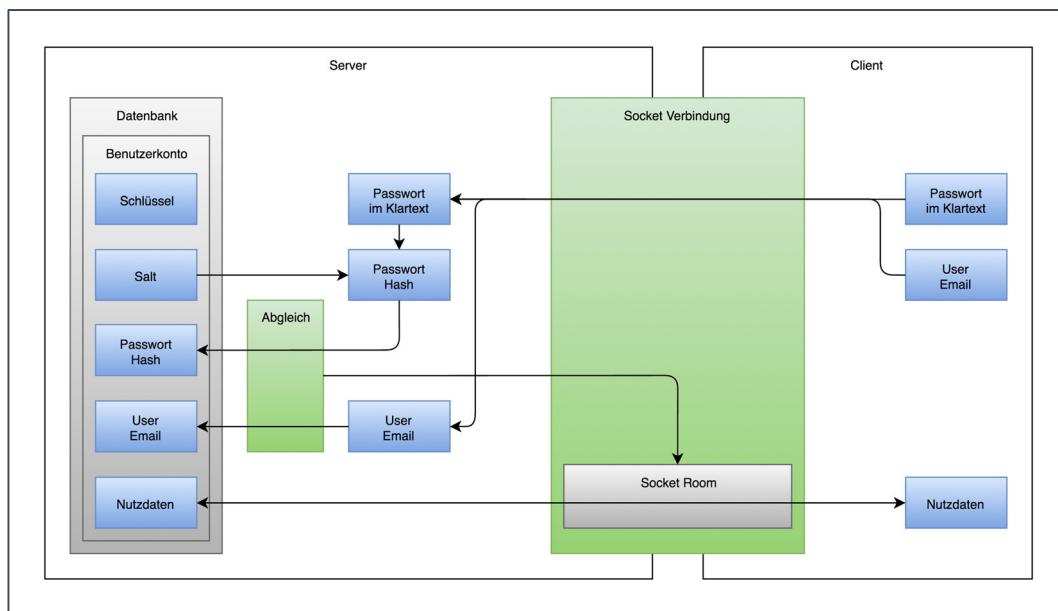


Abbildung 7 Sicherheitsarchitektur Server – Client

Server – Fernsteuergerät

Während zwischen Client und Server ohne Probleme eine sichere Verbindung verwendet werden kann, ist dies beim Fernsteuergerät nicht ohne weiteres möglich. Grund dafür ist hauptsächlich die eingeschränkte Leistungsfähigkeit von dem Mikrokontroller. Es gibt für den gewählten Mikrokontroller keine Programmbibliotheken welche HTTPS und asymmetrische Verschlüsselungsverfahren unterstützen. Diese Verfahren sind zu aufwendig und würden zu viel Arbeitsspeicher benötigen. Trotzdem sollte man nicht auf eine Verschlüsselung bei der Datenübertragung verzichten.

Für einfachere Verschlüsselungsverfahren gibt es durchaus Programmbibliotheken welche nur wenig Speicher und Rechenzeit brauchen. Für dieses System wird die AES-Verschlüsselung im CBC-Modus verwendet. Grundsätzlich wird zwischen dem Server und dem Fernsteuergerät das HTTP Protokoll für die Kommunikation verwendet. Die Nutzdaten werden aber verschlüsselt übertragen. Damit die Verschlüsselung, beziehungsweise die Entschlüsselung korrekt funktioniert, muss dem Server und dem Fernsteuergerät der Schlüssel bekannt sein. Dieser Schlüssel wird beim erstellen eines neuen Users dynamisch erstellt und wird im entsprechenden Datensatz gespeichert. Der Schlüssel muss dann im Programmcode des Fernsteuergeräts hinterlegt werden. Um das Fernsteuergerät eindeutig einem User zuordnen zu können wird die ID, welche auch in der Datenbank gespeichert ist verwendet.

Um die Sicherheit der Verschlüsselung zu verbessern wird im CBC Verfahren des AES-Algorithmus ein Initialisierungsvektor verwendet. Wenn das Fernsteuergerät Daten an den Server senden möchte wird also als erstes zur Laufzeit dieser Vektor generiert. Der Vektor muss unverschlüsselt zum Server übertragen werden und bildet somit den Anfang der Nachricht, welche gesendet wird. Die zu übertragenden Nutzdaten werden danach mit Hilfe des Initialisierungsvektors verschlüsselt und bilden den Rest der Nachricht. Die User ID wird in der URL der HTTP Nachricht übertragen. Der Server kann beim Empfang dann die URL auswerten und die Nachricht eindeutig einem User in der Datenbank zuweisen. Es wird dann der entsprechende Schlüssel aus der Datenbank ausgelesen und die empfangenen Nutzdaten werden zusammen mit dem unverschlüsselt übertragenen Initialisierungsvektor entschlüsselt. Nach der Entschlüsselung findet eine Verifikation der Nutzdaten statt. Nur wenn bestimmte Werte an bestimmten Stellen in der Nachricht korrekt sind werden die Nutzdaten auf dem Server weiterverarbeitet und eine Antwort zurück an das Fernsteuergerät geschickt. Falls die Verifikation der Nutzdaten fehl schlägt wird vom Server nichts weiter unternommen.

Die Nachrichten vom Server an das Fernsteuergerät werden mit den gleichen Methoden verschlüsselt. Auch die Verifikation der Antwort auf dem Fernsteuergerät ist dieselbe. Der Unterschied besteht einzig darin, dass bei einer fehlgeschlagenen Verifikation die ursprüngliche Nachricht noch einmal an den Server gesendet wird.

Durch die Verifikation wird sichergestellt, dass nur korrekt entschlüsselte Nachrichten, und somit auch nur valide Werte weiterverarbeitet werden. Es gibt mehrere Szenarien bei denen eine Verifikation fehl schlägt: Es wurde falsch, oder mit einem falschen Schlüssel verschlüsselt. Durch einen Fehler beim senden, übertragen oder empfangen der Nachricht. Die Nutzdaten konnten richtig entschlüsselt werden, aber Werte liegen ausserhalb des validen Bereichs.

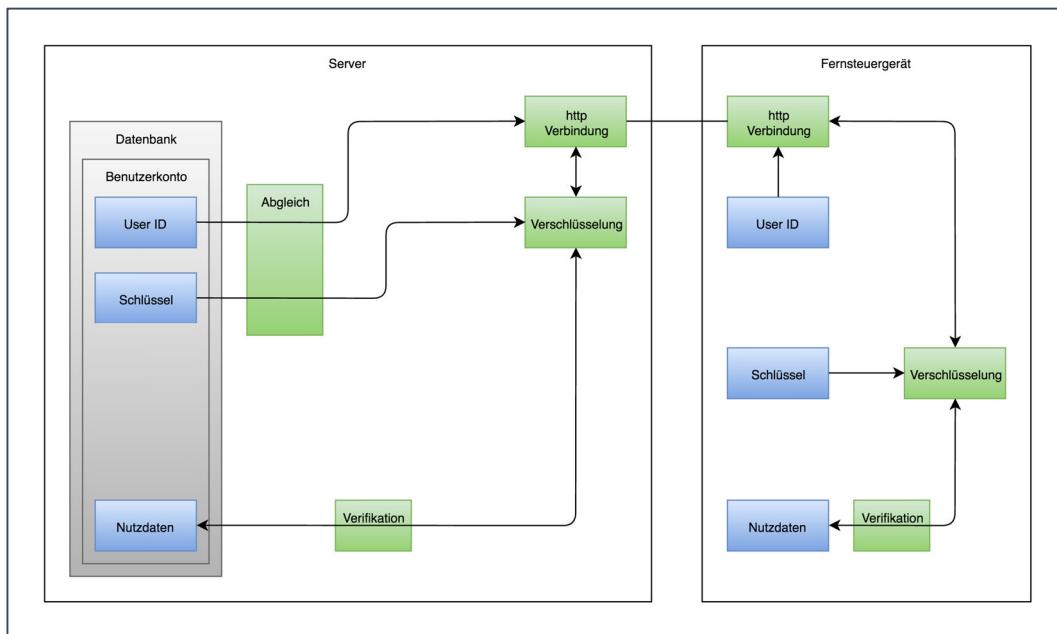


Abbildung 8 Sicherheitsarchitektur Server – Fernsteuergerät

4.7 Testfälle

Im Folgenden werden Testfälle für die einzelnen Komponenten, für die Kommunikation zwischen den Komponenten sowie des Gesamtsystems definiert. Diese Tests widerspiegeln auch wieder die definierten Anforderungen an das System. Diese Tests wurden während der Implementation immer wieder durchgeführt. Die Protokolle zu den Tests sind im Anhang zu finden.

Fernsteuergerät

1. Das Gerät verbindet sich automatisch ins Internet und ist bereit für das Empfangen und Versenden von Daten.
2. Das Gerät konfiguriert sich beim Start automatisch.
3. Die Werte von allen verfügbaren analogen Eingängen werden korrekt gespeichert.
4. Alle verfügbaren digitalen Ausgänge werden entsprechend der gespeicherten Werte geschalten.
5. Werden die Grenzwerte der entsprechenden Eingangssignale erreicht, wird der entsprechend konfigurierte Ausgang automatisch geschalten.

Frontend

1. Die korrekten Einstellungen werden nach Anmeldung angezeigt.
2. Das Betätigen der Buttons für Ausgänge ändert den Zustand.
3. Die Eingangswerte werden korrekt dargestellt und automatisch aktualisiert.
4. Die Buttonlabels können editiert werden.

Server

1. Empfangene Daten werden in der Datenbank im korrekten Datensatz gespeichert.
2. Daten können von der Datenbank aus korrektem Datensatz ausgelesen werden.
3. Passwörter werden sicher und korrekt als Hashs in der Datenbank abgespeichert.
4. Authentifizierung mittels Anmeldedaten funktioniert.

Fernsteuergerät – Server

1. Fernsteuergerät verschlüsselt Nutzdaten.
2. Verschlüsselte Daten werden übertragen und vom Server empfangen.
3. Server entschlüsselt Daten korrekt.
4. Server verschlüsselt entsprechende Nutzdaten für die Antwort.
5. Verschlüsselte Daten werden übertragen und vom Fernsteuergerät empfangen.
6. Fernsteuergerät entschlüsselt Daten korrekt.

Frontend – Server

1. Eindeutige Authentifizierung des Endnutzers an Server funktioniert.
2. Direkte bidirektionale Verbindung zwischen Frontend und Server steht nach Anmeldung.

Systemtest

1. Anmeldung über Webbrowser eines Smartphones erfolgreich.
2. Automatisch aktualisierte Eingangswerte des Geräts werden angezeigt.
3. Das drücken von Buttons schaltet die Ausgänge des Geräts.
4. Konfiguration des Geräts von der Weboberfläche aus erfolgreich.
5. Notfallschaltung reagiert entsprechend der Konfiguration auch ohne Datenverbindung.
6. Beim aufstarten werden keine Ausgangszustände verändert.

5 Evaluierung

In diesem Kapitel wird beschrieben welche Werkzeuge, Technologien und Geräte für dieses Projekt eingesetzt werden. Es ist nicht als vollumfängliche Evaluierung zu verstehen. Da es in diesem Projekt in allen Teilbereichen viele verschiedene Möglichkeiten gibt, und ich persönlich nur mit wenigen Erfahrung habe, hätte eine Evaluierung von all diesen Möglichkeiten zu viel Aufwand benötigt. Trotzdem sind die Entscheidungen nicht unbegründet. Es wird auf die verschiedenen Möglichkeiten eingegangen und es werden Vor- und Nachteile erläutert.

5.1 Hardware

5.1.1 Arduino Plattform

Das Arduino Uno ist die robusteste am besten dokumentierte und meist genutzte Platine der Arduino Familie. Diese Eigenschaften und der günstige Anschaffungspreis machen das Arduino Uno zu einem sehr gut geeigneten Mikrokontroller für dieses Projekt. [5]

Das Arduino 101 verbindet die Einfachheit der klassischen Platinen mit den neuesten Technologien. Die Platine bietet Features wie ein 6 Achsen Accelerometer und Bluetooth Konnektivität. Die bessere Performance und die zusätzlichen Features sind für das Projekt nicht relevant, der grössere Arbeitsspeicher hingegen schon. [6]

Die Arduino Plattform ist so aufgebaut das man die Hauptplatinen mit zusätzlichen Modulen ausstatten kann. Diese sogenannten Shields erweitern die Mikrokontroller um unzählige verschiedene Features. Ansteuerung von Servomotoren, LCD-Displays, Audio Boards und Wifi Konnektivität sind nur einige wenige Beispiele für die vielen Möglichkeiten. Um mit dem Mikrokontroller eine Datenverbindung über das Mobilfunknetz herzustellen wird ein 2G oder 3G Shield benötigt. Alternativ kann auch ein Ethernet Shield eingesetzt werden. Voraussetzung dafür ist eine bestehende kabelgebundene Netzwerkinfrastruktur am entfernten Standort.



Abbildung 9 Die Arduino Plattform [5]

Arduino Uno

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Tabelle 4 Technische Eigenschaften Arduino Uno [5]

Arduino 101

Microcontroller	Intel Curie
Operating Voltage	3.3V (5V tolerant I/O)
Input Voltage (recommended)	7-12V
Input Voltage (limit)	7-20V
Digital I/O Pins	14 (of which 4 provide PWM output)
PWM Digital I/O Pins	4
Analog Input Pins	6
DC Current per I/O Pin	20 mA
Flash Memory	196 kB
SRAM	24 kB
Clock Speed	32MHz
Features	Bluetooth LE, 6-axis accelerometer/gyro
Length	68.6 mm
Width	53.4 mm

Tabelle 5 Technische Eigenschaften Arduino 101 [6]

Cellular Shields

Für die Arduino Plattform gibt es mehrere Hersteller von Shields mit denen man sich ins Mobilfunknetz verbinden kann. Zwei Modelle haben sich nach einiger Recherche als geeignet für dieses Projekt herausgestellt.

Das Unternehmen arduino.org ist der offizielle Hersteller der originalen Arduino Platinen. Nebst den Grundplatten stellt arduino.org auch ein GSM Shield her. Mit dem verbauten Modem M10 von Quectel ist es mit diesem Shield möglich eine 2G Verbindung herzustellen. [7]

Das Unternehmen cooking-hacks.com bietet in seinem Online Shop ein 3G/GPRS Shield für Arduino an. Mit diesem Modul ist es möglich eine Verbindung ins schnelle 3G Netz herzustellen. Dieses Modul kostet aber einiges mehr als das offizielle GSM Shield, und es gibt auch keine Programmbibliothek für dieses Shield. [8]

5.1.2 Raspberry Pi

Der Raspberry Pi ist ein Einplatinencomputer, welcher ein Ein-Chip-System mit einem ARM-Mikroprozessor enthält. Die Grundfläche der Platine entspricht etwa den Abmessungen einer Kreditkarte. Von den Dimensionen her also etwa vergleichbar mit einem Arduino Uno. Die Leistungsfähigkeit des Minicomputers übersteigt ein Mehrfaches von der eines Arduino Boards. Der Raspberry Pi bietet zudem auch eine frei programmierbare GPIO Schnittstelle, worüber LEDs, Sensoren, Displays und andere Geräte angesteuert werden können. Im Gegensatz zum Arduino gibt es aber keine fertigen Mobilfunkmodule und dazu die entsprechenden Programmbibliotheken. Auch ist die Raspberry Pi Plattform nicht quelloffen. [9]



Abbildung 10 Raspberry Pi 3 Model B [10]

5.1.3 Tinkerforge

Tinkerforge ist ein Baukastensystem mit einem breiten Spektrum an vielen verschiedenen Modulen. Sowohl die komplette Software als auch die Hardware aller Module ist Open Source. Tinkerforge setzt noch viel stärker auf Module als es bei Arduino der Fall ist. Die einzelnen sogenannten „Bricks“ haben alle meistens nur eine bestimmte Aufgabe. Die Systeme werden also aus mehreren Bricks zusammengesetzt und über einen separaten Rechner angesprochen. Die Bricks sind also keine vollwertigen Mikrokontroller, sondern sind sozusagen passive Module, welche nur angesprochen werden. Auch bei Tinkerforge setzt man nur auf Ethernet oder Wifi Module. Es gibt kein Modul mit dem man sich ins Mobilfunknetz verbinden kann. Und wie schon beschrieben muss für einen vollwertigen Betrieb eines Tinkerforge Systems zusätzlich ein Computer betrieben werden. [11]

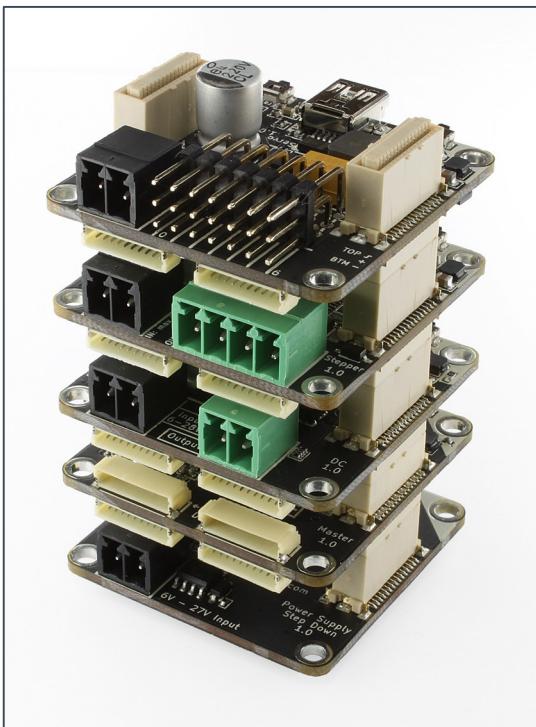


Abbildung 11 Mehrere Tinkerforge Module zusammengesteckt [12]

5.1.4 ausgewählte Plattform

Das Baukastensystem von Tinkerforge sieht zwar sehr vielversprechend aus und ist auch komplett quelloffen, doch bei genauerer Recherche wurde festgestellt, dass dieses System die Anforderungen für dieses Projekt nicht erfüllen kann. Dies vor allem wegen dem Fehlen von Mobilfunkmodulen und dem zusätzlich benötigten Rechner.

Der Raspberry Pi hat viel Leistung und ist auch in der Anschaffung ziemlich günstig. Auch hat man mit dem Minicomputer die Möglichkeit externe Elektronik anzusteuern. Doch fehlt leider geeignete Hardware sowie die entsprechenden Programmbibliotheken für die Mobilfunknetz-Konnektivität. Auch ist das Raspberry Pi keine quelloffene Hardware.

Deshalb wurde die Arduino Plattform ausgewählt. Der relativ günstige Anschaffungspreis, die Möglichkeit verschiedene Shields wählen zu können, der einfache Einstieg als Neuling sowie die grosse Community machen Arduino zu einer idealen Plattform für ein Projekt dieser Art.

5.1.5 Nutzwertanalyse

5.1.5.1 Bewertungskriterien

Es wird mit einer Skala von 0-5 bewertet. Wird die Anforderung am bestmöglichen erfüllt wird die Note 5 vergeben. Wird sie gar nicht erfüllt wird die Note 0 vergeben.

Performance

Es sind damit die technischen Eigenschaften des Mikrocontrollers gemeint. Der wichtigste Wert in dieser Kategorie ist die Menge des Arbeitsspeichers. Die Geschwindigkeit der CPU spielt in diesem Projekt eher eine untergeordnete Rolle.

Zukunftstauglichkeit

Dies ist vor allem auf die Mobilfunktechnik bezogen. Das GSM Netz wird bei einigen Mobilfunkanbietern in ein paar Jahren deaktiviert. [13] Hardware welche nur diese alte Technik unterstützt funktioniert dann nicht mehr.

Reifegrad

Es sind damit Eigenschaften wie die Verbreitung des Produkts, die Qualität der Dokumentationen, die Verfügbarkeit von Tutorials, die Aktivität der Community und Ähnliches gemeint.

Kosten

Die reinen Kosten für die Beschaffung der gesamten Hardware für das Fernsteuergerät.

- **Arduino Uno:** Fr. 28.90 bei play-zone.ch [14]
- **Arduino 101:** Fr. 32.10 bei mouser.ch [15]
- **GSM Shield 2:** Fr. 99.90 bei play-zone.ch [16]
- **3G/GPRS Shield:** Fr. 164.00 bei cooking-hacks.com [8]

5.1.5.2 Variante A

Es wird die Arduino Uno Hauptplatine mit dem offiziellen GSM Shield eingesetzt. Dies ist sicher die günstigere aber auch die leistungsschwächere Variante. Für das GSM Shield stehen offizielle Programmbibliotheken zur Verfügung und im Internet findet man viele Beispiele und Tutorials welche für die gewählte Hardware dieser Variante ausgelegt sind. Da mit dem GSM Shield nur 2G Verbindungen möglich sind, funktioniert diese Variante unter Umständen in ein paar Jahren nicht mehr.

5.1.5.3 Variante B

Es wird die Arduino 101 Hauptplatine mit dem 3G/GPRS Shield eingesetzt. Vorteile dieser Variante sind hauptsächlich die leistungsstärkere Hardware und das 3G Shield welches auch noch nach der Deaktivierung des GSM Netzes funktioniert. Da aber im Arduino 101 ein anderer Prozessor verbaut ist und diese Platine noch ziemlich neu ist kann es unter Umständen zu Problemen bei der Implementierung von Programmcode kommen. Auch fehlt eine Programmbibliothek für das 3G Shield. Die Hardware dieser Variante kostet auch ein wenig mehr.

5.1.5.4 Auswertung

Kriterien	Gewichtung (%)	Note	Note * Gewicht	Note	Note * Gewicht
		Variante A	Variante B		
Performance	15	2	30	4	60
Zukunftstauglichkeit	20	2	40	5	100
Reifegrad	35	4	140	2	70
Kosten	30	3	90	2	60
Total:	100		300		290

Tabelle 6 Auswertung der Nutzwertanalyse

Das wichtigste Kriterium ist der Reifegrad. Deshalb hat dieses auch mit 35% das höchste Gewicht bekommen. Für ein Projekt dieser Art ist es wichtig, dass die Programmierschnittstellen gut dokumentiert sind und die Qualität der Programmbibliotheken stimmt. Auch kann es helfen, wenn man bei Fragen im Internet viele Beispiele und Tutorials findet.

Die Performance hingegen hat mit nur 15% das kleinste Gewicht erhalten. In diesem Projekt spielt die Performance im Sinne von Rechenleistung nicht so eine grosse Rolle. Einzig die Menge des Arbeitsspeichers ist relevant. In Variante A reicht aber der Speicher für das Programm und die Programmbibliothek des GSM Shield aus.

Beim Kriterium Zukunftstauglichkeit hat die Variante A nur 2 von 5 Punkten erhalten. Das GSM Shield ist zwar ausgereift aber leider auch veraltet. In den nächsten Jahren deaktivieren voraussichtlich einige Mobilfunkanbieter das GSM Netz. Falls dies passieren sollte und das Gerät davon betroffen ist kann man als Alternative aber beispielsweise den Anbieter wechseln und somit das Fernsteuergerät weiterhin nutzen. Bis das GSM Netz komplett verschwunden ist wird es wohl doch noch einige Jahre dauern.

Die Hardware der Variante B ist ca. 60 Franken teurer. Auch die Beschaffung der teureren Variante kann sich unter Umständen ein wenig schwieriger gestalten da diese Produkte noch nicht so breitflächig verfügbar sind.

gewählte Variante

Die Variante A hat mit 300 Punkten gegenüber der Variante B mit 290 Punkten bei der Nutzwertanalyse knapp gewonnen. Trotz der Tatsache, dass diese Variante in einigen Jahren nicht mehr funktionieren könnte wird für das Projekt diese Variante ausgewählt. Nicht zuletzt auch weil das Risiko für Komplikationen kleiner ist mit dieser ausgereifteren Hardware. Außerdem kann zu einem späteren Zeitpunkt immer noch die Software weiterentwickelt werden, damit das System auch mit dem 3G Shield funktioniert.

5.2 Frontendtechnologien

Das Frontend wird von den Endnutzern über einen Webbrower aufgerufen. Deshalb kommen natürlich für die Umsetzung der Oberfläche auch Webtechnologien zum Einsatz. Bei den Frontendtechnologien geht es vor allem um den Einsatz von Frameworks, welche die Entwicklung der Webapplikation erleichtern. Man muss dabei zwischen zwei Kategorien von Frameworks unterscheiden, welche beide unglücklicherweise als Frontend Frameworks bezeichnet werden.

Zum einen gibt es Frameworks welche darauf ausgelegt sind sehr schnell Ergebnisse zu erzielen und um damit lauffähige Webanwendungen zu erstellen. Dazu bieten heutige Webframeworks unter anderem einen Datenbankzugriff, Templating-Mechanismen sowie eine saubere Trennung von Präsentation und Code durch Verwendung des Model-View-Controllers als Architekturmuster. [17] Angular, React, Ember oder Backbone sind einige bekannte heutzutage eingesetzte Frameworks in dieser Kategorie.

Die andere Kategorie wird auch manchmal als CSS-Framework bezeichnet. Dieser Name beschreibt auch die Aufgabe dieser Frameworks passender. Beim entwickeln einer Webseite verwendet man immer wieder typische Gestaltungselemente wie zum Beispiel Buttons, Formular-Elemente, Navigations-, Dropdown- und Breadcrumbs-Menüs, Tabs, Slider oder ein Raster-System. Damit man solche Gestaltungselemente nicht jedes Mal von Grund auf neu gestalten muss, gibt es mittlerweile zahlreiche Frameworks, welche Webentwicklern ein Grundgerüst zur Verfügung stellen, und einem damit eine Menge Arbeit abnehmen. [18] Vor allem wenn man eine Oberfläche nach den Prinzipien eines Responsive Webdesign oder Mobile First gestalten will sind solche Frameworks hilfreich. Dabei geht es darum, dass sich der Webinhalt je nach Anzeigegerät entsprechend anpasst.

Bei den meisten Frameworks werden die verschiedenen Inhaltsblöcke in einer Rasteraufteilung (Grid-System) abgebildet. Die einzelnen Gitterelemente bekommen dann spezifische Eigenschaften, welche bestimmen, wie die Dimensionen der Elemente bei verschiedenen Bildschirmgrössen sein sollen. Zusätzlich enthalten die Frameworks viele vordefinierte Stile für HTML Elemente. Somit kann mit wenig Aufwand Webinhalt mit einheitlichem Look-and-feel designt werden. Auch gibt es für einige Frameworks spezielle Elemente um auf einfache Weise zusätzliche Funktionalitäten zu integrieren. Beispielsweise eine Slideshow, welche automatisch die angezeigten Bilder wechselt.

Bootstrap, Foundation und Semantic UI sind 3 der bekanntesten und erfolgreichsten Frameworks in dieser Kategorie. Sie alle funktionieren grundsätzlich nach den gleichen, im vorherigen Absatz beschriebenen Prinzipien. [19] Da mir persönlich die praktischen Erfahrungen bei Foundation und Semantic UI fehlen wird für dieses Projekt Bootstrap als Frontend Framework eingesetzt. Es ist von den drei erwähnten dasjenige mit der grössten Verbreitung und somit auch der grössten Community. Bei Problemen findet man deshalb im Internet leichter Hilfe. Die Integration gestaltet sich relativ simpel und die Dokumentation von Bootstrap erleichtert den Einstieg. Außerdem habe ich schon während einem kleinen Projekt erste Erfahrungen mit dem Framework sammeln können.

Um die Implementierung des HTML zu vereinfachen wird Jade eingesetzt. Jade ist eine Template Engine, welche für node.js auf der Serverseite wie auch auf Clientseite im Webbrower eingesetzt wird. [20] Mit Jade lässt sich HTML einfacher und schneller schreiben, da eine Syntax verwendet wird, welche auf öffnende und schliessende HTML-Tags verzichtet sowie die Struktur des HTML durch Einrückung definiert. Jade kann so eingesetzt werden, dass während der Laufzeit des JavaScript dynamisch Teile der Oberfläche gerendert und angezeigt werden. Es ist aber auch möglich, dass aus einer Jade Datei statisch eine HTML-Datei generiert wird. In diesem Projekt wird die Weboberfläche statisch mit Jade erstellt.

Des Weiteren wird JavaScript und insbesondere die jQuery Bibliothek verwendet um die Logik im Frontend zu implementieren.

5.3 Servertechnologien

Auf Serverseite kommen Technologien in Frage, welche auf PHP, Java, Ruby, Python, JavaScript und noch einigen anderen Sprachen basieren. Ausserdem sind einige der Technologien stark darauf ausgelegt, dass man die Web Applikation nach einer MVC Architektur entwickelt. Beispielsweise das Ruby on Rails oder das Django Framework. Dies setzen auf die MVC Architektur. Daneben gibt es noch unüberschaubar viele Frameworks um die Entwicklung von Web Applikationen zu vereinfachen.

Ich persönlich konnte bisher nur Erfahrungen mit Apache / PHP und node.js sammeln. Zumindest wenn es um die Entwicklung von Web Anwendungen geht. Wobei sich die Erfahrungen mit PHP auch in Grenzen halten. Es bleibt also nur noch das auf JavaScript basierende node.js übrig. In den letzten Jahren konnte ich einige Erfahrungen mit JavaScript auf Client sowie auch auf Serverseite sammeln.

Glücklicherweise ist node.js für dieses Projekt sehr gut geeignet und wird deshalb auch für die Serverseite eingesetzt. Gerade auch weil die Architektur für dieses Projekt nicht wie eine klassische Web Anwendung aufgebaut ist. Der Server soll sowohl als Schnittstelle zum Fernsteuergerät, sowie auch zum Web-Client dienen. Dabei hat er die Aufgabe die ankommenden Daten zu persistieren sowie auch von der einen zur anderen Schnittstelle weiterzuleiten. Ausserdem soll der Web-Client über eine bidirektionale WebSocket Verbindung angebunden sein. Würde man bei einer solchen Ausgangslage versuchen ein Framework einzusetzen welches stark auf die MVC Architektur setzt wäre man ziemlich eingeschränkt und müsste einen Aufwand betreiben damit das System schlussendlich den Anforderungen entspricht. Ausserdem habe ich persönlich noch mit keinem MVC Framework gearbeitet und ich müsste mir erst die Grundlagen derselben erarbeiten.

Bei node.js hingegen hat man durch die vielen verschiedenen Module, welche zur Verfügung stehen, viel mehr Optionen und man erzielt schnell akzeptable Lösungsansätze. Ein weiterer Vorteil ist sicherlich auch, dass man mit node.js auf Serverseite sowie auch auf Clientseite JavaScript verwenden kann.

Als Datenbank würde sich natürlich MongoDB anbieten. Die einfache Integration in node.js sowie die einfache Handhabung im Zusammenhang mit JavaScript sind sicherlich einige Vorzüge von MongoDB. Da es sich dabei aber um eine schemalose Datenbank handelt, und in diesem Projekt die Informationen in einer klar strukturierten Form abgespeichert werden sollen, wird MySQL als klassische relationale Datenbank eingesetzt.

5.4 Entwicklungswerkzeuge

Bei der Wahl der Entwicklerwerkzeugen setzte ich vorwiegend auf Tools mit denen ich schon gearbeitet habe und ich mich schon auskenne.

Als Codeeditor kommt Atom von GitHub zum Einsatz. Atom ist ein Open Source-Texteditor auf Basis von Electron. Electron besteht aus dem Webbrowser Chromium und dem JavaScript-Framework Node.js und erlaubt, beliebige Anwendungen aus JavaScript, HTML und CSS zu erstellen. Atom integriert einen Paketmanager namens apm. Aufgrund der Rendering-Engine als Unterbau bietet Atom Syntaxhervorhebung für viele Programmiersprachen und erlaubt den Anwendern, das Programm beliebig mit Plug-Ins und Themes zu erweitern. [21]

In diesem Projekt kommen unter anderem Technologien wie jade für das generieren von HTML und Templates zum Einsatz. Um die jade-Dateien in valides HTML umzuwandeln kann man ein Command Line Tool einsetzen. Jedoch ist es mühsam jedes Mal aufs Neue einen Kommandozeilenbefehl abzusetzen falls man etwas in einer jade-Datei geändert hat. Um diese und noch andere lästige Aufgaben zu automatisieren werden sogenannte Task-Runners eingesetzt. Grunt, Gulp und Brunch sind drei solcher Hilfswerkzeuge. Brunch ist zwar das unbekannteste von den drei genannten Tools, bietet aber gegenüber den beiden anderen Werkzeugen einige Vorteile. Zum einen muss man bei Brunch nicht viel konfigurieren da das Tool so ausgelegt ist, dass es die gängigen Aufgaben out of the box korrekt

erledigt. Ausserdem ist Brunch um einiges schneller als die Konkurrenz. Insbesondere wenn die Web Projekte etwas grösser geworden sind fällt der Geschwindigkeitsunterschied auf. Wegen diesen genannten Vorteilen und der Tatsache, dass ich persönlich schon Erfahrungen mit Brunch sammeln konnte wird für dieses Projekt dieser Task Runner eingesetzt.

Die Paketverwaltung npm (node package manager) wird verwendet um auf sehr komfortable Weise zusätzliche Funktionalitäten in Form von Modulen in das Projekt für den Server zu integrieren.

Für das Debugging des JavaScript Codes auf der Web-Client Seite kommen die Entwickertools für den Chrome Browser zum Einsatz.

Um das Arduino Projekt zu komplizieren und auf den Mikrokontroller zu laden wird die Arduino IDE verwendet. Da das Programm aber zum schreiben von dem Quellcode nicht sehr komfortabel ist, wird dieser auch mit dem Atom Codeeditor erstellt.

Für das Monitoring des Datenverkehrs sowie für die Analyse und Fehlersuche wird der Netzwerksniffer Wireshark eingesetzt.

5.5 Mobilfunkanbieter

Bei der Wahl des Mobilfunkanbieters und eines entsprechend geeigneten Angebots sind vor allem zwei Faktoren zu berücksichtigen:

- Menge der übertragenen Daten
- Netzabdeckung

Die Summe der übertragenen Daten, welche bei einer normalen Anfrage an den Server entstehen belaufen sich durchschnittlich auf ca. 1600 Bytes. Wenn das Fernsteuergerät so konfiguriert ist, dass alle 15 Minuten eine Anfrage stattfindet, ergibt das 2976 Anfragen pro Monat. Es wird pro Monat also ein Datenvolumen von 4.54 MB verbraucht. Nimmt man an, dass jede zweite Anfrage aufgrund eines Fehlers erneut gesendet werden muss ist man immer noch unter 10 MB an verbrauchtem Datenvolumen pro Monat.

No.	Protocol	Length	Info
168	TCP	74	21755 → 80 [SYN] Seq=0 Wi
169	TCP	78	80 → 21755 [SYN, ACK] Seq
170	TCP	66	21755 → 80 [ACK] Seq=1 Ac
171	HTTP	207	POST /remote/fffffffff HTT
172	TCP	66	[TCP Window Update] 80 →
173	TCP	66	80 → 21755 [ACK] Seq=1 Ac
174	TCP	130	[TCP segment of a reassem
175	TCP	66	80 → 21755 [ACK] Seq=1 Ac
176	HTTP	325	HTTP/1.1 200 OK (applica
177	TCP	66	80 → 21755 [FIN, ACK] Seq
178	TCP	66	21755 → 80 [ACK] Seq=206
179	TCP	66	[TCP Out-Of-Order] 80 → 2
180	TCP	66	21755 → 80 [FIN, ACK] Seq
181	TCP	66	[TCP Out-Of-Order] 80 → 2
182	TCP	66	21755 → 80 [ACK] Seq=207
183	TCP	66	[TCP Dup ACK 181#1] 80 →
184	TCP	78	[TCP Dup ACK 182#1] 21755

Abbildung 12 Datenverkehr bei einer normalen Serveranfrage

In der Schweiz ist die Netzabdeckung bei den drei grossen Mobilnetzbetreiber Swisscom, Sunrise und Salt durchgehend sehr gut. Das Swisscom Netz schneidet dabei am besten ab. [22]

Beim Vergleich der Prepaid Angebote der Mobilfunkanbieter hat sich dasjenige von der Migros als klar das Beste herausgestellt. Im Angebot von M-Budget Mobile sind 10 MB Datenvolumen pro Monat inklusive. [23] Da wohl nur in seltenen Fällen dieses Volumen überschritten wird, ist dieses Angebot für diese Anwendung im Prinzip kostenlos. Ausserdem wird bei M-Budget Mobile das Swisscom Netz verwendet. Die Netzabdeckung ist also auch sehr gut.

5.6 Serverhosting

Bei diesem Projekt wird node.js als Servertechnologie eingesetzt. Zusätzlich kommt eine MySQL Datenbank zum Einsatz. Die meisten klassischen Webhoster bieten in ihren günstigen Angeboten schon eine MySQL Datenbank an. Sucht man aber einen Anbieter bei dem man auch eine node.js Webanwendung hosten lassen kann, findet man jedoch nur noch eine Handvoll Angebote. Meistens handelt es sich dabei um PaaS Angebote, welche meistens ziemlich flexibel und günstig sind. Alternativ kann man sich auch eine virtuelle Serverinstanz in einem Datencenter mieten. Dies hat den Vorteil, dass man die volle Kontrolle hat und den Server nach seinen Bedürfnissen einrichten kann. Dies ist aber natürlich auch teurer und der Aufwand für die Einrichtung ist höher.

Im Rahmen dieser Diplomarbeit, bei der erst einmal nur ein Prototyp der Anwendung entwickelt wird, wird node.js sowie auch die MySQL Datenbank während der Entwicklung auf dem lokalen Rechner ausgeführt. Für die Demonstration während der Präsentation der Arbeit werde ich dann mein persönliches NAS als Server nutzen.

6 Hardware

6.1 Schnittstellen

Während man es beim Server sowie dem Frontend nicht direkt mit physikalischen Schnittstellen zu tun hat, besitzt der Mikrokontroller einige Schnittstellen die nach aussen geführt werden.

Stromversorgung

Der Mikrokontroller kann direkt über die USB Schnittstelle betrieben werden. Alternativ besitzt das Arduino Uno Board eine Schnittstelle für eine Stromversorgung mit einem externen Netzteil. Die maximal zulässige Spannung beträgt 12VDC. Das Netzteil sollte eine Leistung von 300-500mA liefern können. [24]

Ethernet Verbindung

Das Fernsteuergerät kann über eine Ethernet-Schnittstelle mit dem Netzwerk verbunden werden. Dafür wird das Arduino Ethernet Shield 2 benötigt. Dieses besitzt eine handelsübliche RJ-45 Buchse.

Mobilfunknetzverbindung

Das Fernsteuergerät kann sich mit einem GSM Modem ins Internet verbinden. Dafür wird das Arduino GSM Shield 2 benötigt. Um einen möglichst guten Empfang zum GSM Netz zu ermöglichen, muss die Antenne eventuell nach aussen geführt werden. Außerdem wird für den Betrieb mit dem GSM Modem eine Sim-Karte eines Mobilfunknetzbetreibers benötigt.

Digitale Ausgänge

An dem Fernsteuergerät können maximal 6 digitale Ausgänge betrieben werden. Folgende Pins sind als digitale Ausgänge konfiguriert:

- **Kanal 1:** Pin 5
- **Kanal 2:** Pin 6
- **Kanal 3:** Pin 8
- **Kanal 4:** Pin 9
- **Kanal 5:** Pin 0
- **Kanal 6:** Pin 1

Die aktivierte Ausgänge liefern eine Spannung von 5V. Jeder einzelne Pin verträgt eine maximale Last von 40mA. [25] Möchte man grössere Lasten mit dem Fernsteuergerät schalten, können Relais eingesetzt werden.

Analoge Eingänge

An dem Fernsteuergerät können maximal 6 analoge Eingänge betrieben werden. Folgende Pins sind als analoge Eingänge konfiguriert

- **Kanal 1:** Pin A0
- **Kanal 2:** Pin A1
- **Kanal 3:** Pin A2
- **Kanal 4:** Pin A3
- **Kanal 5:** Pin A4
- **Kanal 6:** Pin A5

Wird Kanal 5 und 6 benötigt, muss auf die beiden Status Pins verzichtet werden und die beiden Pins müssen im Arduino Sketch entsprechend entfernt werden.

Die analogen Eingänge erwarten eine Spannung zwischen 0 und 5V. Die Spannung wird durch die AD Wandler in Werte zwischen 0 und 1023 umgewandelt. Für den Anschluss der analogen Signale kann ein Klemmenbrett verwendet werden. Dieses vereinfacht das anschliessen der Sensoren an das Fernsteuergerät.

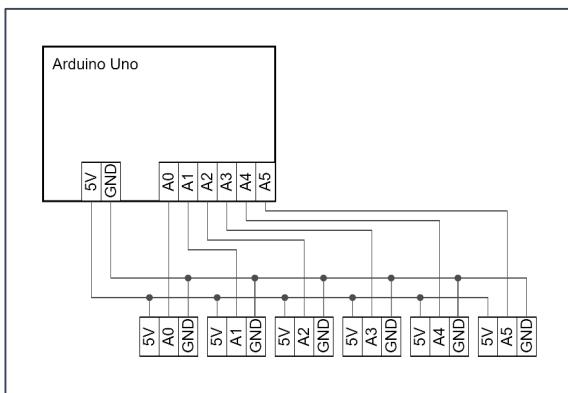


Abbildung 13 Klemmanschlüsse für die analogen Eingänge

6.2 Zusammenbau Prototyp

Der Prototyp besteht aus einem Arduino Board, welches alternativ zusammen mit dem GSM Shield oder dem Ethernet Shield betrieben werden kann. Um die Ausgänge des Fernsteuergeräts zu simulieren wurden an den digitalen Ausgängen vier grüne Leuchtdioden angeschlossen. Außerdem werden an Pin 18 (A4) und 19 (A5) zwei Status LEDs betrieben. Diese geben unter anderem Auskunft darüber wann die Kommunikation stattfindet. An den vier analogen Eingängen sind ein Potentiometer, ein Drucksensor, ein Photowiderstand und ein einfacher Taster angeschlossen.

Alle benötigten Elektronikbauteile, Verbindungsleitung sowie das Steckmodul waren in einem Arduino Starterkit vorhanden. Es mussten also keine zusätzlichen Komponenten für diesen Testaufbau bestellt werden. Natürlich ist der Aufbau in dieser Form nicht für den richtigen Einsatz gedacht. Dafür fehlen vor allem Relais um auch grössere Lasten und Spannungen schalten zu können.

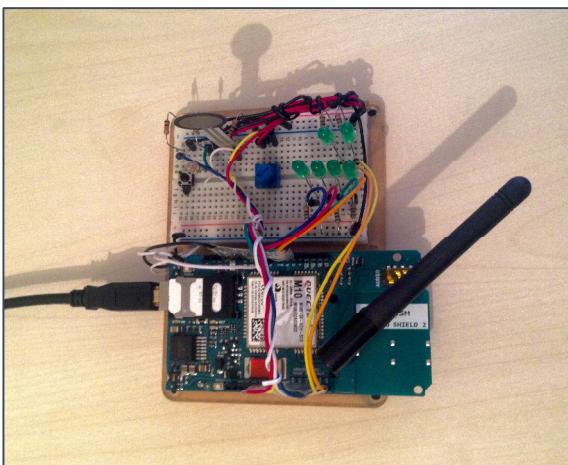


Tabelle 7 Testaufbau der Hardware

7 Implementierung

In diesem Kapitel wird genauer auf die Aspekte der Implementation der Teilsysteme eingegangen. Dazu gehören zum einen Erläuterungen zu einzelnen Funktionen, sowie auch Erklärungen zum Aufbau und der Funktionsweise einzelner Programmbibliotheken beziehungsweise einzelner Module. Außerdem wird auf die Schwierigkeiten während der Implementation eingegangen.

7.1 Software Mikrokontroller

7.1.1 Programmierumgebung

Die Sprache, welche zum programmieren eines Arduino Boards benutzt wird, ist lediglich eine Ansammlung von Funktionen welche in C, C++ oder Assembler geschrieben sind. Der geschriebene Code wird danach mit dem Compiler avr-g++ kompiliert. Es ist also möglich alle Standardfunktionen, welche von dem Compiler unterstützt werden zu nutzen.

Die offizielle Arduino IDE vereinfacht einige Abläufe um welche sich ein C-Programmierer separat kümmern muss. Zum Beispiel werden die Funktionsprototypen automatisch generiert und es entfällt die lästige Vorwärtsdeklaration und das mühsame erstellen von Header Files. Auch geht das kompilieren des Projekt, sowie das nachfolgende übertragen des Maschinencodes auf das Arduino Board leicht von der Hand. Ansonsten bietet die IDE aber nicht sehr viel Komfort und es empfiehlt sich den Programmcode in einem anderen Editor zu schreiben und die IDE lediglich zum kompilieren und übertragen zu verwenden.

Im Bezug zur Entwicklungsumgebung sei hier noch erwähnt, dass es zwei verschiedene Downloads der fast exakt gleichen Software gibt. Zum einen bietet die Seite arduino.cc die Software in der Version 1.6.8 an. Bei der «anderen» Seite arduino.org steht die IDE in der Version 1.7.10 zum Download bereit. (Stand 29.4.16) Dabei handelt es sich bei der Version auf arduino.org um einen Fork der originalen Arduino Software. Neben minimalen offensichtlichen Unterschieden der beiden Entwicklungsumgebungen werden vor allem auch andere Programmbibliotheken mitgeliefert. Das sorgte am Anfang für einige Verwirrung und ich musste mich erst im Wirrwarr zurechtfinden. Insbesondere fehlte mir die Programmbibliothek für das aktuelle Ethernet Shield, weshalb ich dann auf die andere von den beiden Entwicklungsumgebungen wechseln musste. Ein Rechtstreit um die Markennamen und den Vertrieb, sowie die Herstellung der Arduino Boards ist der Grund für die ganze Verwirrung. Dies soll aber nicht Thema dieser Diplomarbeit sein.

7.1.2 Programmstruktur

Da es sich beim Arduino um eine Mikrokontroller-Plattform handelt, gibt es kein vorinstalliertes Betriebssystem, welches Aufgaben wie eine Task- oder Speicherverwaltung übernimmt. Man programmiert also auf einer tiefen, hardwarenahen Ebene. Das bietet zum einen den Vorteil, dass man viel Kontrolle hat, aber auch den Nachteil, dass man gut darauf achten muss keine Speicherüberläufe oder dergleichen zu produzieren.

In einem Arduino Programm gibt es zwei Hauptroutinen welche beim erstellen eines neuen Sketches auch schon vorgegeben sind. Beim Start des Mikrocontrollers wird als erstes die Setup Routine einmalig ausgeführt. Danach wechselt das Programm in eine Endlosschlaufe. Beim programmieren für einen Mikrocontroller muss man ein wenig umdenken, da man sich um viele Sachen kümmern muss, welche ansonsten schon vom Betriebssystem beziehungsweise einer Laufzeitumgebung oder aber über ein API geregelt sind. Ein Beispiel ist das Event Handling. Während man zum Beispiel bei Ja-

vaScript relativ einfach einen Listener für einen Buttonclick implementiert muss man beim Mikrocontroller selber in der Hauptschleife bei jedem Durchlauf den Zustand eines Eingangs überprüfen und dafür sorgen, dass eine dadurch aufgerufene Funktion nur einmalig ausgeführt wird.

Die Abbildung unten zeigt eine vereinfachte Darstellung von dem Programmablauf für dieses Projekt. Im Folgenden werden die einzelnen Aufgaben genauer erläutert.

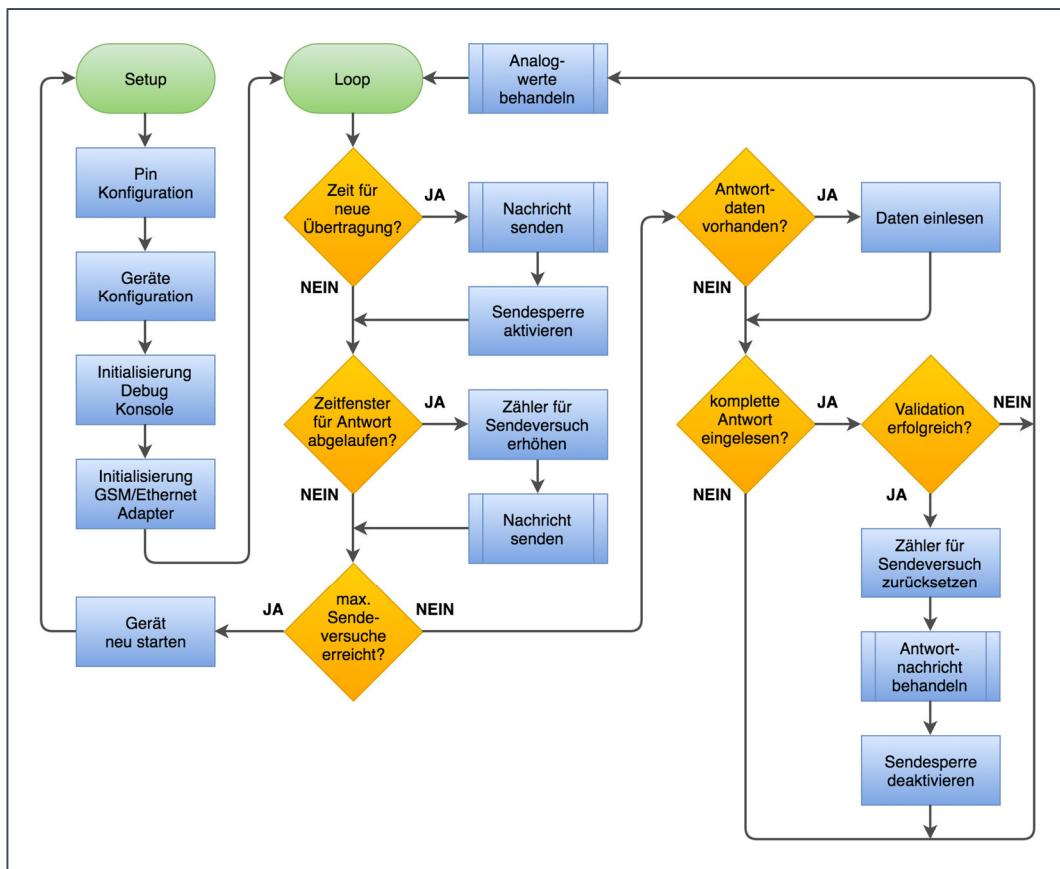


Abbildung 14 Programmablauf vom Fernsteuergerät

Pin Konfiguration

Die Pins, an denen die Eingänge und Ausgänge an das Board angeschlossen sind, müssen richtig konfiguriert werden. Die Positionen dieser Pins werden als globale Konstanten (Bsp. `const byte OUTPUT_PIN1 = 5;`) definiert. Standardmäßig sind alle Pins als Eingänge konfiguriert. Mit `pinMode()` können die Pins umkonfiguriert werden. Mit `digitalWrite()` kann der Zustand eines Pins geändert werden.

```

...
pinMode(OUTPUT_PIN4, OUTPUT);
...
digitalWrite(OUTPUT_PIN1, LOW);
...
  
```

Geräte Konfiguration

Die Konfiguration des Fernsteuergeräts wird bei jeder korrekt empfangenen Antwortnachricht im EEPROM des Mikrocontrollers persistiert. Dabei werden aber nur die geänderten Bytes neu geschrieben. Dies ist vor allem wichtig, da ein EEPROM nur eine begrenzte Anzahl an Speichervorgängen

durchführen kann. Die so gespeicherten Daten sind auch nach einem Neustart des Geräts noch verfügbar. Sie werden beim Start einmalig in den `payloadBuffer` geladen und danach wird die `handleConfigData()` Funktion aufgerufen.

```
..
for(int i = 0; i < PAYLOAD_MAX; i++) {
    payloadBuffer[i] = EEPROM.read(i);
}
handleConfigData();
..
```

Diese Funktion liest die Werte aus dem `payloadBuffer` ein und kopiert diese in dafür vorgesehene Variablen im Arbeitsspeicher. Die Variablen der Konfigurationseinstellungen sind zum Teil in Datentypen abgespeichert, welche mehr als ein Byte Speicherplatz im Arbeitsspeicher benötigen. Der Zwischenspeicher ist aber in Form eines Byte Arrays aufgebaut. Deshalb muss darauf geachtet werden, dass die Daten in der richtigen Reihenfolge in die dafür vorgesehenen Variablen abgespeichert werden. Dabei hilft die Bitshift Funktion der C Programmiersprache.

```
..
sendInterval = (unsigned long)payloadBuffer[39] << 24;
sendInterval |= (unsigned long)payloadBuffer[40] << 16;
sendInterval |= (unsigned long)payloadBuffer[41] << 8;
sendInterval |= (unsigned long)payloadBuffer[42];
..
```

Bei den Konfigurationsvariablen handelt es sich um globale Variablen für Grenzwerte, Schaltmodi, Ausgangszustände sowie Sendekadenzen. Diese Variablen werden von den anderen Funktionen im Programm verwendet.

Initialisierung Debug-Konsole

Viele Debugging Möglichkeiten bieten sich beim programmieren für ein Arduino nicht an. Eine Möglichkeit des Debuggings ist es aber sich bestimmte Werte an bestimmten Positionen im Programm in einer Textkonsole anzeigen zu lassen. Bei der Arduino Software ist ein Serial-Monitor enthalten welcher für solche Aufgaben gut geeignet ist.

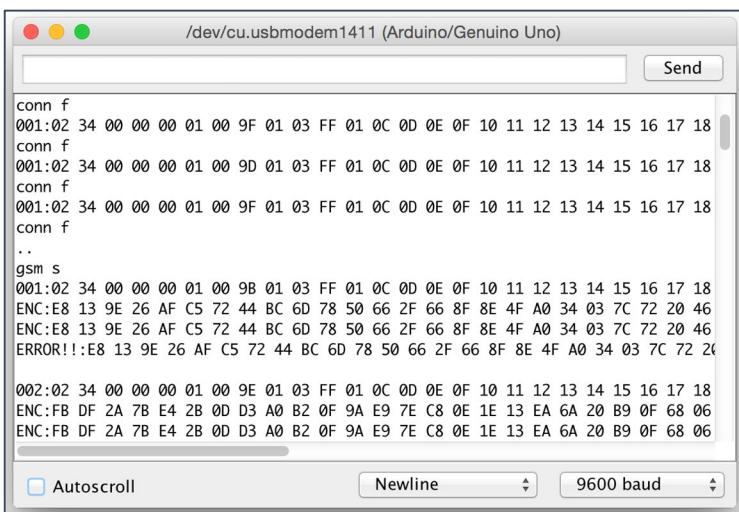


Abbildung 15 Debug-Konsole des Arduino

Um diese ansprechen zu können muss man im Arduino Programm erst die serielle Schnittstelle initialisieren und das Board kann anschliessend über die USB-Schnittstelle mit dem Monitor kommunizieren. Danach lassen sich Texte sowie auch Werte über eine einfache Anweisung auf dem Serial Monitor anzeigen.

```
..
Serial.begin(9600);
Serial.print(F("\n.."));
..
```

Damit auch Datenwerte schön formatiert auf dem Monitor ausgegeben werden, wird eine Hilfsfunktion benutzt. Diese gibt einzelne Bytes eines gespeicherten Wertes in Hexadezimalform inklusive führendem 0 aus.

```
..
void printHex(byte *data, byte length) {
    for (int i=0; i<length; i++) {
        if (data[i] < 0x10) {Serial.print("0");}
        Serial.print(data[i],HEX);
        Serial.print(" ");
    }
}
..
```

Initialisierung GSM/Ethernet Adapter

Beim Fernsteuergerät hat man die Möglichkeit entweder ein Ethernet Shield, oder aber ein GSM Shield für die Verbindung ins Internet zu verwenden. Dies bietet sich an, da beide entsprechenden Bibliotheken und deren Funktionen nach dem gleichen Prinzip aufgebaut sind. So müssen lediglich die Programmbibliothek, einige feste Konstanten sowie die Anweisungen für die Initialisierung der Zusatzmodule geändert werden.

Der Programmcode für die Initialisierung wurde weitgehend aus den offiziellen Beispielen, welche man auf den Arduino-Seiten findet übernommen. die `blinkLED()` Anweisung sorgt dafür, dass durch ein Status-LED angezeigt werden kann ob die Verbindung hergestellt werden konnte oder nicht.

```
..
while(notConnected) { //GSM
    if ((gsmAccess.begin(PINNUMBER) == GSM_READY) & //GSM
        (gprs.attachGPRS(GPRS_APN, GPRS_LOGIN, GPRS_PASSWORD) == GPRS_READY)) { //GSM
        notConnected = false; //GSM
    } //GSM
    else { //GSM
        Serial.print(F("\ngsm f")); //GSM
        blinkLED(STATUS_PIN1, 4, 150, false); //GSM
        delay(1000); //GSM
    } //GSM
} //GSM
..
..
```

Zeit für neue Übertragung?

Eine der einstellbaren Werte in der Konfiguration des Fernsteuergeräts ist die Zeitspanne zwischen zwei Datenübertragungen. Diese Wert entspricht Millisekunden und wird als unsigned long gespeichert. Somit lässt sich eine Zeitspanne zwischen 0 Sekunden und knapp 50 Tagen im Gerät einstellen.

Um mit dem Arduino die Zeit messen zu können, kann man die `millis()` Funktion nutzen. Beim Aufruf dieser Anweisung wird die Anzahl vergangener Millisekunden seit dem Start des Mikrokontrollers zurückgegeben. Es wird zudem eine Variable benötigt, welche den Wert vergangener Millisekunden zum Zeitpunkt der letzten Übertragung enthält. Dieser Wert wird gespeichert, wenn die Antwortnachricht komplett empfangen wurde.

```
..
sendLock = false;
..
previousMillis = millis();
..
```

Um zu testen ob es Zeit für eine neue Übertragung ist, muss der Wert der letzten Übertragung vom Wert des aktuellen Zeitpunkts subtrahiert werden. Ist der berechnete Wert grösser als die konfigurierte Zeitspanne, kann eine neue Übertragung gestartet werden.

```
..
if(millis() - previousMillis >= sendInterval && !sendLock) {
    sendMessage();
}
..
```

Nachdem die Nachricht gesendet wurde wird die Variable `sendLock` auf `true` gesetzt. Dies verhindert ein mehrmaliges senden der Nachricht. Deshalb auch die zusätzliche Bedingung in der if Abfrage.

Zeitfenster für Antwort abgelaufen?

Unter normalen Bedingungen wird die Antwortnachricht nach einer kurzen Zeit korrekt empfangen. Falls es aber Probleme bei der Verbindung oder dem Server gibt, kann es vorkommen, dass keine Antwort an das Fernsteuergerät zurückgesendet wird. Die erwartete Zeitspanne zwischen Sendung und Empfang wird in einer Variable als Millisekunden Wert gespeichert. Wenn diese Zeit abgelaufen ist und keine Antwort empfangen wurde, wird ein weiterer Versuch gestartet und die Nachricht wird erneut gesendet. Dabei wird ein Zähler für fehlgeschlagene Übertragungen erhöht.

```
..
if(millis() - previousMillis >= errorInterval && sendLock) {
    noResponseCount++;
    ..
    sendMessage();
}
..
```

maximale Sendeversuche erreicht?

Wenn das Maximum an Sendeversuchen erreicht wurde wird das Gerät komplett neu gestartet. Dies wird dadurch erreicht indem der Watchdog-Timer des Arduino aktiviert wird und das Programm danach eine Endlosschleife ausführt. [26]

```
..
void softwareReset( uint8_t prescaller) {
    wdt_enable(prescaller);
    while(1) {}
}
..
```

Antwortdaten vorhanden?

Bei jedem Durchgang im Hauptloop wird durch die `client.available()` Funktion überprüft ob ankommende Daten vorhanden sind. Wenn ankommende Daten vorhanden sind wird jedes einzelne Byte eingelesen. Die einzelnen Bytes welche in der Antwortnachricht als ASCII Zeichen codiert sind werden überprüft. Dies, weil bei der Antwortnachricht der HTTP Header nicht relevant ist, und nur die eigentliche Nachricht gespeichert werden soll. Der Header und der Inhalt einer HTTP Nachricht sind durch eine leere Zeile unterteilt.

```
..
if(!hasResponse) {
    data = client.read();

    // payload begins after a blank line
    if (data == 0x0A && blank) {
        hasResponse = true;
        responsePosition = 0;
        return;
    }
    else {blank = false;

    if(data == 0x0A) {
        data = client.read();
        if(data == 0x0D) {blank = true;}
    }
}
..
..
```

Wenn die eigentliche Nachricht erreicht wurde wird die Hilfsvariable `hasResponse` auf `true` gesetzt und die ankommenden Daten werden ab diesem Zeitpunkt zwischengespeichert. Die ersten 16 Bytes der Antwort entsprechen dabei dem Initialvektor, welcher benötigt wird um die nachfolgenden Daten zu entschlüsseln. Der Rest der Nachricht wird im `payloadBuffer` zwischengespeichert.

```
..
if(hasResponse) {
    data = client.read();
    if(responsePosition < 16) {
        initialVector[responsePosition] = data;
    }
    else {
        payloadBuffer[responsePosition - 16] = data;
    }
    responsePosition++;
}
..
..
```

komplette Antwort eingelesen?

Wenn keine neuen Daten mehr ankommen, sprich die komplette Antwortnachricht empfangen wurde, wird die Datenverbindung getrennt. Es wird dann die Hilfsvariable hasResponse auf false gesetzt und danach die Nachricht, welche in den Zwischenspeicher geladen wurde, verarbeitet.

```
..
if(!client.available() && !client.connected() && hasResponse) {
    hasResponse = false;
..
readResponse();
}
..
```

Bei der Verarbeitung werden als erstes die Daten im Zwischenspeicher entschlüsselt. Gleich danach werden die Daten validiert. Dabei wird überprüft ob an bestimmten Stellen der Nachricht die richtigen Werte gespeichert sind. Nur wenn die Validation erfolgreich war wird der Zähler für die Fehlversuche zurückgesetzt, die nun entschlüsselten Daten weiterverarbeitet und die Sendesperre deaktiviert.

```
..
else {
    noResponseCount = 0;
    // update EEPROM
    int written = updateEEPROM();
    Serial.print(F("\nWRITTEN:"));
    Serial.print(written);
    handleConfigData();
    Serial.print(F("\nRES:")); //debug
    sendLock = false;
}
..
```

Bei der Weiterverarbeitung werden alle Werte der Konfiguration, welche sich seit dem letzten Empfang geändert haben im EEPROM Speicher persistiert. Anschliessend werden die Daten aus dem `payloadBuffer` in die entsprechenden Konfigurationsvariablen kopiert.

Analogwerte behandeln

Bei jedem Durchgang der Hauptschleife werden die Eingänge eingelesen und die zugehörigen Ausgänge entsprechend der Konfiguration der einzelnen Kanäle geschalten. Dafür wird die Funktion `checkTrigger(...)` verwendet. In der Konfiguration des Fernsteuergeräts kann ein unterer sowie ein oberer Grenzwert als auch ein Schaltmodus eingestellt werden. Erreicht ein Eingangssignal einer der Grenzwerte wird je nach Schaltmodus der Ausgang automatisch geschalten. Die `checkTrigger(...)` Funktion gibt ausserdem den Status des entsprechenden Ausgangs zurück.

```

..
byte checkTrigger(int value, channel channel, byte outputPin) {
..
    switch (channel.triggerMode) {
        case 0:
            outputStatus = 0;
            break;
        case 1:
            if(lower) {outputStatus = 1;}
            else if(higher) {outputStatus = 2;}
            else {outputStatus = 0;}
            break;
        case 2:
            if(higher) {outputStatus = 2;}
            else {outputStatus = 0;}
            break;
        case 3:
            if(lower) {outputStatus = 1;}
            else {outputStatus = 0;}
            break;
        case 4:
            if(lower) {outputStatus = 2;}
            else if(higher) {outputStatus = 1;}
            else {outputStatus = 0;}
            break;
        case 5:
            if(lower) {outputStatus = 2;}
            else {outputStatus = 0;}
            break;
        case 6:
            if(higher) {outputStatus = 1;}
            else {outputStatus = 0;}
            break;
        case 7:
            if(lower) {outputStatus = 2;}
            else if(higher) {outputStatus = 2;}
            else {outputStatus = 0;}
            break;
    }
    if(outputStatus == 1) {digitalWrite(outputPin, LOW);}
    if(outputStatus == 2) {digitalWrite(outputPin, HIGH);}

    return outputStatus;
}
..

```

7.1.3 Schwierigkeiten und Herausforderungen

Arbeitspeicher sparen mit Folgen

Da ich vor allem am Anfang der Implementierung immer wieder mal komisches unlogisches Verhalten des Programms feststellen musste, vermutete ich, dass der Mikrokontroller ein Problem durch zu wenig verfügbarem Speicherplatz hat. Um Speicher im RAM zu sparen gibt es mehrere Möglichkeiten. Vor allem Strings, welche dazu verwendet werden um Infos in der Debug-Konsole anzuzeigen verbrauchen unnötig viel Speicher. Diese konstanten Strings werden zum einen im Flashspeicher abgelegt und auch noch zusätzlich beim Programmstart in den RAM geladen. Um dies zu verhindern kann ein einfaches Makro eingesetzt werden. Dazu umschliesst man den String einfach mit dem F() Makro. Aus `Serial.print("String");` wird dann einfach `Serial.print(F("String"));`. Damit wird verhindert, dass der String zusätzlich in den Arbeitsspeicher geladen wird und man kann dadurch eine Menge RAM sparen. [27]

Eine weitere Möglichkeit ist die Nutzung von PROGMEM. Variablen welche mit PROGMEM deklariert sind werden auch nur ausschliesslich im Flash Speicher abgelegt. Es muss dabei aber darauf geachtet werden, dass die Daten bei Gebrauch korrekt in den RAM geladen werden. Beim testen dieser Funktionalitäten war mir dies aber noch nicht bewusst und die mit PROGMEM deklarierten Variablen konnten nicht korrekt gelesen werden. Beim kompilieren wurde kein Fehler angezeigt da ein gültiger Pointer auf ein char Array gefunden wurde. Der gefundene Pointer bezieht sich aber auf den Flashspeicher. Die Funktion im Programm referenziert aber auf die Stelle im Arbeitsspeicher wo natürlich keine korrekten Daten gefunden werden. [28] Unglücklicherweise habe ich die Variable für die PIN-Nummer meiner SIM-Karte auch mit PROGMEM deklariert. Das hat schlussendlich dazu geführt, dass mehrmals die PIN-Nummer falsch eingelesen wurde und die SIM-Karte gesperrt wurde.

Ich verwende die PROGMEM Deklaration nicht mehr und spare nun nur noch Arbeitsspeicher bei den Strings durch das F() Makro. Probleme mit fehlendem Arbeitsspeicher gibt es schlussendlich auch keine mehr.

Fehler im GSM Modul

Nach langer Fehlersuche hat sich herausgestellt, dass durch einen Fehler in der GSM Programmbibliothek empfangene Daten unter bestimmten Umständen nicht richtig eingelesen werden. Das Problem tritt immer dann auf, wenn sich unter den empfangenen Daten ein Byte mit dem Wert 255 befindet. Dieser Wert wird schlicht ignoriert und verworfen. Da die Daten in einen Buffer gelesen werden welcher aus einem Byte Array besteht, verschieben sich alle ankommenden Daten nach dem ignorierten Byte um eine Stelle im Array. Die ankommenden Daten sind noch verschlüsselt und müssen zusammen mit dem übertragenen Initialvektor korrekt entschlüsselt werden. Wenn jetzt natürlich die Daten verschoben im Buffer gespeichert sind und einige Werte fehlen, klappt die Entschlüsselung nicht korrekt. Eine anschliessende Validierung der entschlüsselten Daten schlägt infolgedessen natürlich auch fehl. Das bedeutet, dass ein weiteres Mal eine Nachricht an den Server gesendet wird und ein weiteres Mal die Antwort empfangen werden muss.

Zu dem Zeitpunkt als ich dem Problem auf die Spur kam, wurde bei der Verschlüsselung noch kein Initialvektor verwendet. Deshalb traten auch beim erneuten Empfang der verschlüsselten Daten die gleichen Fehler auf. Um dem entgegenzuwirken werden jetzt bei jeder Übertragung Initialvektoren für die Verschlüsselung generiert. Dabei wird der Initialvektor unverschlüsselt übertragen und die restlichen Daten, welche mit dem übertragenen Initialvektor verschlüsselt wurden, werden danach angehängt. Es kann nun immer noch passieren, dass ein Initialvektor erzeugt wird, welcher selbst, sowie aber auch die daraus erzeugten verschlüsselten Daten, ein Byte mit dem Wert 255 enthält. In diesen Fällen ist es immer noch nötig die Nachricht vom Fernsteuergerät aus neu zu übertragen. Die Wahrscheinlichkeit, dass mehrmals innerhalb kurzer Zeit falsche Daten empfangen werden ist aber

relativ klein. Durch dieses Vorgehen kann der Fehler im GSM Modul umgangen werden. Gleichzeitig ist auch die Verschlüsselung durch die dynamisch erzeugten Initialvektoren stärker.

7.1.4 Programmbibliotheken

Bis auf eine Bibliothek, welche für die Verschlüsselung der Daten zuständig ist, wurden ausschliesslich offizielle Bibliotheken der Arduino IDE verwendet. Um das GSM Shield anzusprechen kommt die GSM Bibliothek zum Einsatz. Durch die Verwendung der Ethernet2 Bibliothek kann das Fernsteuergerät auch mit einem Ethernet Shield betrieben werden. Gerade diese beiden Bibliotheken sind glücklicherweise ziemlich ähnlich aufgebaut. Beide verwenden eine Client Klasse bei welchen die Methoden für das Senden und Empfangen implementiert sind. Dadurch kann man durch sehr wenig Aufwand den Programmcode so umschreiben, dass das Fernsteuergerät mit dem Ethernet oder dem GSM Shield betrieben werden kann. Es müssen lediglich einige Variablen Deklarationen und der Code für die Initialisierung ausgetauscht werden.

```
..
// handleAnalogValues(); //ETH
// if (Ethernet.begin(mac) == 0) { //ETH
//   Serial.println(F("dhcp f")); //ETH
//   blinkLED(STATUS_PIN1, 4, 150, false); //ETH
//   Ethernet.begin(mac, 80); //ETH
// } //ETH
// delay(1000); //ETH
// Serial.println(F("eth s")); //ETH
// blinkLED(STATUS_PIN1, 1, 250, true); //ETH

while(notConnected) { //GSM
  handleAnalogValues();
  if ((gsmAccess.begin(PINNUMBER) == GSM_READY) & //GSM
      (gprs.attachGPRS(GPRS_APN, GPRS_LOGIN, GPRS_PASSWORD) == GPRS_READY)) { //GSM
    notConnected = false; //GSM
  } //GSM
  else { //GSM
    Serial.print(F("\ngsm f")); //GSM
    blinkLED(STATUS_PIN1, 4, 150, false); //GSM
    delay(1000); //GSM
  } //GSM
} //GSM
..
..
```

Im Folgenden wird noch auf die verwendete Krypto-Bibliothek eingegangen. Die Arduino AESLib basiert auf der AVR-Crypto-Lib und enthält nur die Assembler Implementationen der AES Verschlüsselung. Die Bibliothek ist so aufgebaut, dass sie ohne Probleme direkt in der Arduino IDE verwendet werden kann. Die Bibliothek stellt Funktionen zur Ver- und Entschlüsselung mit der AES128, sowie der AES256 Methode zur Verfügung. Dies sowohl im Single als auch im Blockchiffre Modus. [29] Im Programmcode des Fernsteuergeräts wird AES128 im Blockchiffre Modus verwendet. Die Implementierung in den Programmcode ist ziemlich simpel. Man muss lediglich auf das Array referenzieren in dem sich die zu ver-/entschlüsselnden Daten befinden. Zusätzlich muss noch der Initialvektor, der eigentliche Schlüssel sowie die Grösse des Arrays angegeben werden.

```
..
aes128_cbc_enc(CIPHER_KEY, initialVector, (void *)payloadBuffer, PAYLOAD_MAX);
..
```

Dadurch, dass in dieser Bibliothek ausschliesslich die AES Methoden in Assembler integriert sind braucht die Bibliothek auch nur wenig Speicher. Für ein Projekt mit einem Arduino Uno, welcher sowieso nur 2KB Ram enthält also sehr gut geeignet.

7.2 Server

Da der Server sowohl mit dem Fernsteuergerät als auch mit dem Frontend kommuniziert ist eine klare Abtrennung zwischen diesen Teilen schwierig. Deshalb wird es vor allem in Bezug auf die Kommunikation über die WebSockets einige Überschneidungen mit Erläuterungen geben, welche eher ins Kapitel Frontend gehören.

7.2.1 Datenbank

An den Server ist eine kleine MySQL Datenbank angebunden. Darin werden alle relevanten Daten der Endnutzer gespeichert.

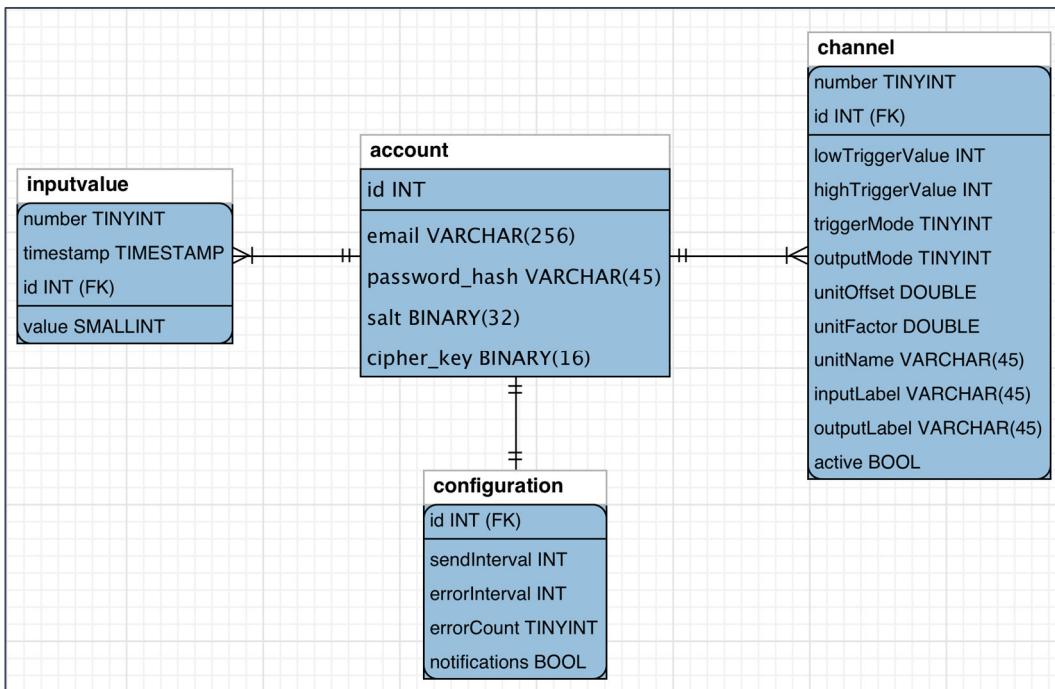


Abbildung 16 Datenbankmodell

account

Bei einer Neuregistrierung eines Benutzers wird auf dem Server aus dem gewählten Passwort ein Hash mit einem zur Laufzeit zufällig generierten Salt erzeugt. Der Chiffrierschlüssel welcher für die Konfiguration des Fernsteuergeräts benötigt wird, wird ebenfalls zur Laufzeit generiert. Diese Informationen werden in einem neuen Eintrag der **account** Tabelle persistiert. Zusätzlich werden bei der Neuregistrierung Einträge mit Standardwerten in der **configuration**, sowie der **channel** Tabelle erzeugt.

configuration

Zu jedem Account gehört genau eine Konfiguration. Der Primärschlüssel besteht deshalb nur aus dem Fremdschlüssel **id** von der **account** Tabelle. Neben den gespeicherten Einstellungen, welche das Kommunikationsverhalten des Fernsteuergeräts bestimmen, kann man im **notifications** Feld bestimmen,

ob bei einem automatischen Zustandswechsel eines Ausgangs des Geräts der Endnutzer benachrichtigt werden soll.

inputvalue

In dieser Tabelle werden die analogen Werte gespeichert, welche vom Fernsteuergerät eingelesen und an den Server übermittelt werden. Demzufolge hat ein Account mehrere dieser Werte gespeichert. Der Primärschlüssel setzt sich aus der Nummer des Kanals, einem Zeitstempel und der `id` vom Account als Fremdschlüssel zusammen. Die analogen Werte von allen Eingangskanälen des Fernsteuergeräts werden gleichzeitig beim Server empfangen. Bevor diese Werte persistiert werden wird ein Zeitstempel erzeugt. Ein kompletter Satz analoger Werte werden also mit dem gleichen Zeitstempel in der Datenbank gespeichert.

channel

Ein Account hat mehrere Kanäle. Deshalb besteht der Primärschlüssel dieser Tabelle aus der `id` als Fremdschlüssel von der `account` Tabelle, sowie der Nummer des einzelnen Kanals. Einige Informationen, welche in dieser Tabelle gespeichert sind, werden beim Fernsteuergerät benötigt. Dies sind die Werte `lowTriggerValue`, `highTriggerValue`, `triggerMode` und `outputMode`. Die Werte `unitOffset`, `unitFactor`, `unitName`, `inputLabel`, `outputLabel` und `active` werden jedoch nur für das Frontend benötigt. Im Wert `active` wird angegeben ob der entsprechende Kanal auf der Benutzeroberfläche angezeigt werden soll.

7.2.2 Programmstruktur

Im Gegensatz zu dem Mikrokontroller ist der Ablauf des Programmcodes beim Server nicht sequentiell. Eine Charakteristik von vielen Serverapplikationen, welche auf node.js basieren, ist die auf Ereignisse (Events) basierende Struktur. Das hat den Vorteil, dass beispielsweise Datenbankabfragen nicht die gesamte Applikation blockieren bis die Ergebnisse zur Verfügung stehen. Doch muss man sich im Klaren darüber sein, dass eben diese Ergebnisse erst zu einem späteren Zeitpunkt geliefert werden, aber das Programm in der Zwischenzeit weiteren Code ausführt. Um diesen asynchronen Programmcode kontrollieren zu können, werden Callback Funktionen eingesetzt.

Darstellung

Um den Ablauf und die Struktur der Applikation aufzuzeigen wird nachfolgend für die Visualisierung ein Sequenzdiagramm verwendet. Es dient hauptsächlich dazu, die Kommunikation und Datenübertragung zwischen den einzelnen Teilsystemen aufzuzeigen. Es werden deshalb auch nur die Funktionsaufrufe dargestellt, welche eine Datenübertragung zu einem anderen Teilsystem zur Folge haben. Auch wird nur der Ablauf aufgezeigt, wenn alles reibungslos funktioniert. Die Fehlerbehandlung bei beispielsweise falsch eingegeben Nutzerdaten ist nicht aufgeführt. Die Namen der Funktionen entsprechen denen, welche auch im Quelltext zu finden sind. Um die Übersicht zu wahren werden die Parameter aber nicht aufgeführt. Zu beachten ist, dass einige Funktionen asynchron sind. Diese Funktionen welche ihren Rückgabewert an eine Callback-Funktion zurückgeben sind mit einem Pfeil in die entgegengesetzte Richtung und einer senkrechten Roten Linie dargestellt. Subroutinen werden durch einen kurzen Pfeil und einer senkrechten roten Linie dargestellt. Falls die Subroutinen einen Rückgabewert zurückliefern, wird dies mit einem kurzen Pfeil in die andere Richtung dargestellt.

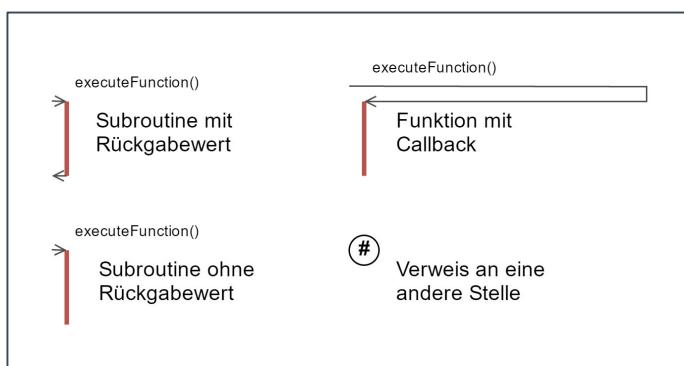


Abbildung 17 Notation des Sequenzdiagramms

Namenskonventionen

Damit klarer wird welche Aufgaben die Funktionen übernehmen, werden in diesem Projekt folgende Namenskonventionen verwendet:

- **get:** Daten welche aus einem Speicherobjekt im eigenen Programm geholt werden.
- **set:** Daten welche in einem Speicherobjekt im eigenen Programm festgelegt werden.
- **load:** Laden von Daten welche in einem nicht flüchtigen Speicher abgelegt sind.
- **save:** Speichern von Daten in einen nicht flüchtigen Speicher.
- **fetch:** Auslesen von Daten in einer Datenbank.
- **store:** Speichern von Daten in eine Datenbank.
- **apply:** Anlegen von Datensätzen in eine Datenbank.
- **send:** Senden von Daten über eine Netzwerkschnittstelle an ein anderes Teilsystem.
- **receive:** Empfangen von Daten über eine Netzwerkschnittstelle von einem anderen Teilsystem.
- **insert:** Einfügen von Inhalt in einer grafischen Benutzeroberfläche.
- **collect:** Sammeln von Inhalt aus einer grafischen Benutzeroberfläche.
- **assign:** Zuweisung eines Zustands an ein Objekt in der Benutzeroberfläche.

Kommunikation mit dem Fernsteuergerät

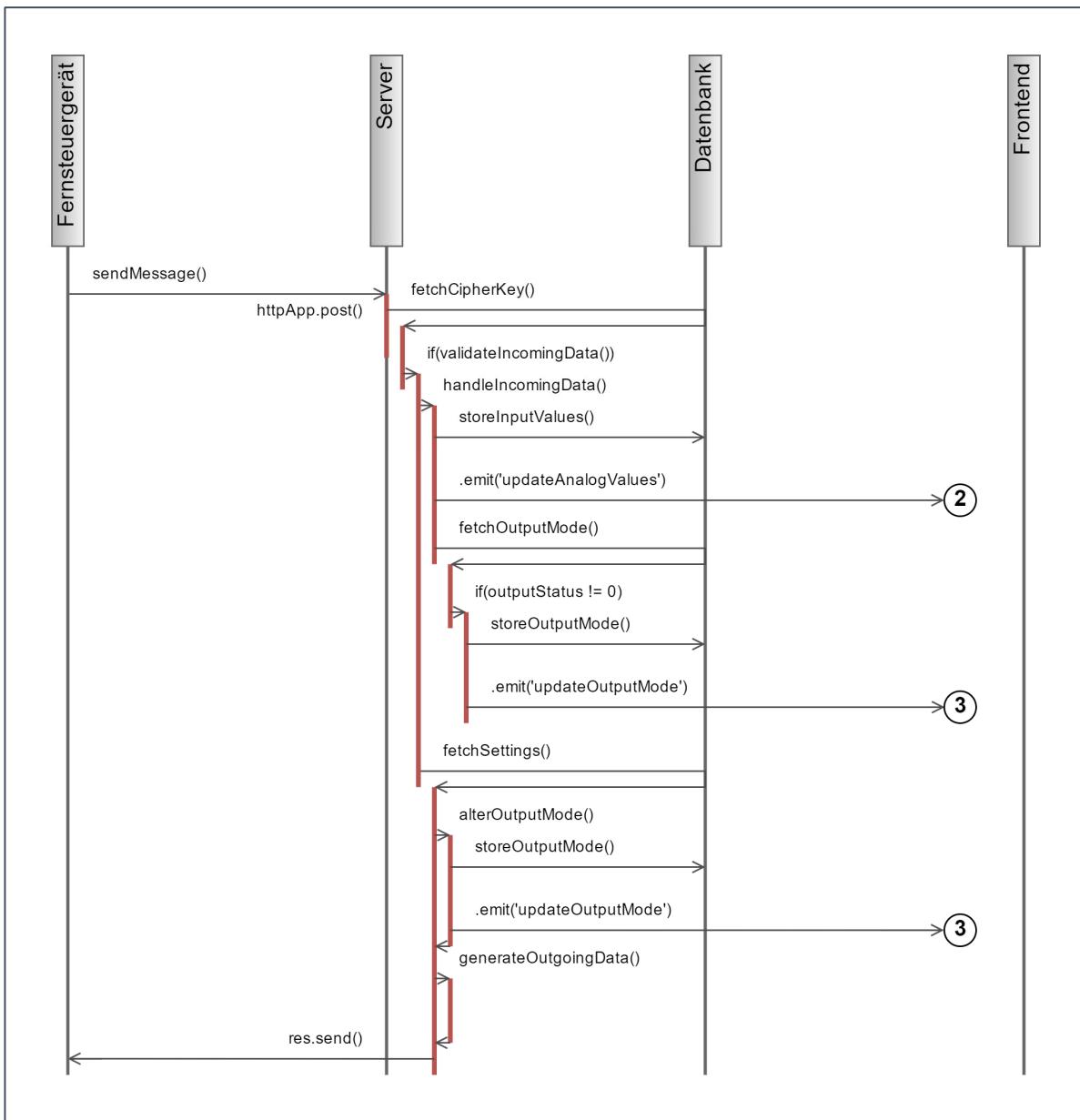


Abbildung 18 Sequenzdiagramm Fernsteuergerät zu Server

Mit der `sendMessage()` Funktion wird vom Fernsteuergerät eine HTTP Nachricht im Post Modus zum Server gesendet. Auf Serverseite wird beim Empfang ein Event ausgelöst, welches vom Listener `httpApp.post()` behandelt wird. Damit die Nachricht identifiziert und korrekt entschlüsselt werden kann wird die `userId`, welche mit der URL der Nachricht übertragen wurde ausgelesen.

```

...
httpApp.post("/remote/id:userId", rawParser, function(req, res) {
  let userId = req.params.userId;
...
  
```

Der zu der ID zugehörige Chiffrierschlüssel wird aus der Datenbank ausgelesen. Hier ist zu erwähnen, dass alle Anfragen an die Datenbank asynchron sind. Die Weiterverarbeitung der Daten findet deshalb in einer Callback Funktion statt welche als Parameter in der `.query()` Funktion definiert ist.

```
..
let query = ` 
  select
    cipherKey
  from
    account
  where
    id = ?
`;
let values = [userId];
dbConn.query(query, values, function(err, result) {
  if(err) {
    console.error(err);
    return;
  }
..
`;
```

Die empfangenen Nutzdaten der HTTP Nachricht werden in einen `payload` und `iv` (Initialvektor) aufgeteilt. Der `payload` wird anschliessend mit dem `iv` und dem `cipherKey` aus der Datenbank entschlüsselt. Der `payload` wird anschliessend validiert. Dabei wird überprüft ob an bestimmten Stellen der Nachricht die richtigen Werte gespeichert sind. Nur wenn diese Validation erfolgreich war, werden die Daten weiterverarbeitet. Ansonsten wird lediglich eine Fehlermeldung in der Konsole des Servers ausgegeben.

Bei erfolgreicher Validierung werden zwei Funktionen mit asynchronem Verhalten aufgerufen. Zum einen ist dies die `handleIncomingData()` Funktion, welche die empfangenen Daten persistiert und an das Frontend weiterleitet. Als zweites werden mit `fetchSettings()` die Konfigurationseinstellungen aus der Datenbank ausgelesen und an das Fernsteuergerät gesendet.

Bei der Behandlung der empfangenen Daten werden als erstes die analogen Werte vom `payload` in einem Array gespeichert. Da die analogen Werte im `payload` je zwei Byte Speicher belegen, und diese im Big-Endian Format abgelegt sind, müssen diese Werte korrekt ausgelesen werden. Die Methoden der Buffer Klasse von node.js bieten dazu einige Möglichkeiten. Mit `.readUInt16BE()` kann ein unsigned Integer, welcher 16bit lang ist und im Big-Endian Format abgelegt ist, bequem ausgelesen werden.

```
..
function handleIncomingData(payload, userId) {
  let inputValue = [];
  inputValue[0] = payload.readUInt16BE(1);
  inputValue[1] = payload.readUInt16BE(4);
  inputValue[2] = payload.readUInt16BE(7);
..
`;
```

Die analogen Werte werden zusammen mit einem Zeitstempel und der zugehörigen Kanalnummer in der Datenbank gespeichert. Die analogen Werte werden zusätzlich über die WebSocket Kommunikation mit `.emit('updateAnalogValues')` an das Frontend gesendet.

Der `outputStatus` von den einzelnen Kanälen vom Fernsteuergerät ist auch im `payload` der empfangenen Nachricht enthalten. Dieser Status gibt an ob ein Ausgang durch die automatische Schaltung geschalten wurde. Um den veränderten Zustand der Ausgänge im Frontend korrekt darstellen zu können, findet ein Vergleich zwischen den momentan in der Datenbank gespeicherten und den vom Fernsteuergerät empfangenen Daten statt. Dabei wird der `outputMode` der Kanäle mit `fetchOutputMode()` aus der Datenbank ausgelesen. Wenn der `outputStatus` vom Fernsteuergerät 0 ist, das heisst sich nicht

verändert hat, passiert nichts weiter. Auch wenn sich der `outputStatus` vom Fernsteuergerät nicht mit dem von der Datenbank unterscheidet passiert nichts weiter. Nur wenn sich die beiden Werte unterscheiden wird zum einen der neue Wert mit `storeOutputMode()` in der Datenbankpersistiert und der neue Zustand mit `.emit('updateOutputMode')` an das Frontend gesendet.

```
..  
fetchOutputMode(userId, function(userId, data) {  
    for(let i = 0; i < outputStatus.length; i++) {  
        if(outputStatus[i] !== 0) {  
            console.log(outputStatus[i]);  
            console.log(data[i]);  
            if(data[i] != outputStatus[i]) {  
                let outputMode = outputStatus[i] + 4;  
                storeOutputMode(userId, {mode: outputMode, channel: i+1}, function(res) {  
                    console.log(res);  
                });  
                app.io.in(userId).emit('updateOutputMode', {channel: i+1, mode: outputMode});  
            }  
        }  
    }  
});  
..
```

Bei der zweiten parallelen Verarbeitung wird mit `fetchSettings()` die Konfiguration für das Fernsteuergerät aus der Datenbank ausgelesen. Bei diesen Konfigurationsdaten wird der `outputMode` vor dem versenden mit `alterOutputMode()` abgeändert. Dies ist nötig, da an das Fernsteuergerät nur die Schaltbefehle 0, 1 und 2 (nichts schalten, ausschalten, einschalten) übertragen werden dürfen. In der Datenbank werden aber vier verschiedene Zustände gespeichert:

- 1:.Ausgang ist ausgeschaltet
- 2:.Ausgang ist eingeschaltet
- 3:.Ausgang soll ausgeschalten werden
- 4:.Ausgang soll eingeschalten werden

```
..
function alterOutputMode(userId, settings) {
  let newSettings = settings;
  for(let i = 0; i < newSettings.channel.length; i++) {
    let outputMode = newSettings.channel[i].outputMode;
    // only send 0, 1 or 2 to the device for further information see documentation
    if(outputMode >= 3 && outputMode <= 4) {
      outputMode -= 2;
      newSettings.channel[i].outputMode = outputMode;

      let data = {channel: i+1 ,mode: outputMode};
      storeOutputMode(userId, data, function(res) {
        console.log(res);
      });

      app.io.in(userId).emit('updateOutputMode', {channel: i+1, mode: outputMode});
    }
    else {
      newSettings.channel[i].outputMode = 0;
    }
  }
  return newSettings;
}
..

```

Falls der `outputMode` verändert wurde wird dieser mit `storeOutputMode()` neu in der Datenbank gespeichert. Auch wird der veränderte Zustand mit `.emit('updateOutputMode')` an das Frontend gesendet.

Anschliessend wird der zu sendende payload mit `generateOutgoingData()` generiert. Dabei kommen wieder die Methoden der Buffer Klasse zum Einsatz.

```
..
function generateOutgoingData(settings) {
  let payload = Buffer.alloc(48);
  let channel = settings.channel;
  ..

  payload.writeUInt32BE(settings.device.sendInterval, 39);
  payload.writeUInt32BE(settings.device.errorInterval, 43);
  payload.writeUInt8(settings.device.maxErrorCount, 47);

  return payload;
}
..
```

Bevor nun aber die Daten an das Fernsteuergerät gesendet werden können, müssen sie verschlüsselt werden. Zusammen mit einem zufällig generierten `iv` und dem `cipherKey` wird der `payload` chiffriert und der `iv` vorne angehängt. Um Fehler zu minimieren, welche durch empfangene Bytes mit dem Wert 255 beim Fernsteuergerät entstehen, wird die Prozedur der Verschlüsselung in einer Endlosschlaufe ausgeführt. Es werden solange neue Initialvektoren generiert und mit diesen verschlüsselt bis im verschlüsselten Payload keine Werte mehr mit 255 vorhanden sind.

```
..  
while(true) {  
    iv = crypto.randomBytes(16);  
    encryptedPayload = encrypt(payload, iv, cipherKey);  
    responseData = Buffer.concat([iv, encryptedPayload]);  
    if(responseData.includes(255)) {  
        continue;  
    }  
    console.log(payload);  
    break;  
}  
..
```

Dieses Stück Code soll als Workaround verstanden werden. Es handelt sich momentan bei dem System noch um einen Prototyp. Wenn der Bug im GSM Modul des Fernsteuergeräts behoben werden kann ist auch dieser Code nicht mehr nötig.

Abschliessend werden mit `res.send()` die Daten an das Fernsteuergerät gesendet.

Authentifizierung und Registrierung

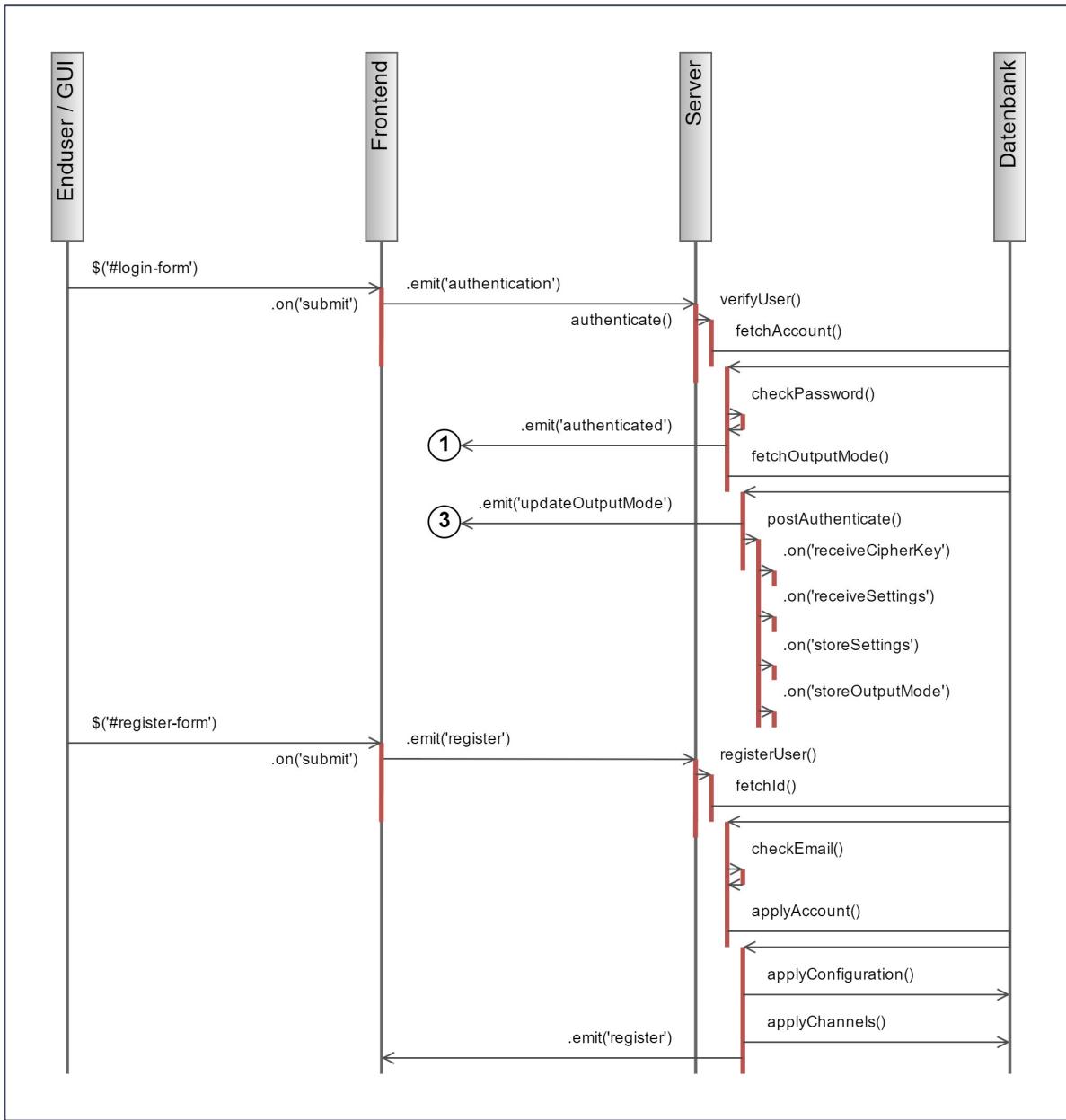


Abbildung 19 Sequenzdiagramm Authentifizierung und Registrierung

Um sich als Endnutzer in das System einzuloggen muss in einem Formular auf der Weboberfläche E-Mail-Adresse und Passwort eingegeben werden. Durch einen Klick auf die Login Schaltfläche wird ein Ereignis ausgelöst, welches von dem `.on('submit')` Event Listener anschliessend behandelt wird. Die Informationen aus den Formularfeldern werden ausgelesen und mit `.emit('authentication')` über den WebSocket zum Server gesendet.

Für die Authentifizierung am Server wird das node Modul `socketio-auth` verwendet. Bei der Initialisierung des Moduls wird ein Event Listener erstellt welcher auf das Event `'authentication'` hört. Kommt ein solcher Event beim Listener an wird die `authenticate()` Funktion ausgeführt. Nach einer erfolgreichen Authentifizierung wird die Funktion `postAuthenticate()` aufgerufen.

```
..
require('socketio-auth')(app.io, {
  authenticate: authenticate,
  postAuthenticate: postAuthenticate,
  timeout: 'none'
});
..
```

Bei der `authenticate()` Funktion werden der Socket, von welchem der Event kam, die Login Informationen und eine Callback Funktion als Parameter übergeben. Die eigentliche Verifizierung passiert dann in der `verifyUser()` Funktion. Dabei werden die Informationen, welche der übermittelten E-Mail Adresse entsprechen, aus der Datenbank ausgelesen. Wenn kein Datensatz mit der entsprechenden E-Mail-Adresse gefunden wurde, wird die Ausführung der Funktion abgebrochen und die Callback Funktion mit den Informationen `data.verified = false` und `data.userId = null` aufgerufen. Im anderen Fall wird aus dem übermittelten `password` und dem aus der Datenbank ausgelesenen `salt` ein `hash` generiert. Wenn dieser generierte `hash` mit dem `passwordHash` aus der Datenbank übereinstimmt wird `data.verified` auf `true` gesetzt. Anschliessend wird noch die `userId` aus der Datenbank dem `data` Objekt zugewiesen und die Callback Funktion aufgerufen. Bei einem falschen Passwort wird zwar trotzdem die Callback Funktion aufgerufen, aber `data.verified` bleibt im `false` Zustand.

```
..
dbConn.query(query, values, function(err, result) {
  //verify
  if(result.length === 0) {
    console.error('verifyUser: user not found');
    return callback(data);
  }
  //check Password
  let hash = createHash(user.password, result[0].salt);
  if(hash === result[0].passwordHash) {
    data.verified = true;
  }
  data.userId = result[0].id;
  return callback(data);
});
```

Zurück in der Callback Funktion der `verifyUser()` Funktion wird im Falle eines falschen Passworts, oder wenn der Benutzer nicht gefunden wurde, ein entsprechender Error an das `socketio-auth` Modul zurückgegeben. Das Modul sendet dann das Event `'unauthorized'` mit dem entsprechenden Error an das Frontend. Ist die Verifikation jedoch erfolgreich sendet das Modul das Event `'authenticated'` an das Frontend. Als weiteres wird der verbundene Socket einem Raum mit dem Namen, welcher der `userId` entspricht, zugewiesen. Somit können Socket Events gezielt über diesen Raum an nur den entsprechenden Benutzer gesendet werden. Zusätzlich wird nach der Authentifizierung der `outputMode` der Ausgänge von der Datenbank ausgelesen und über `.emit('updateOutputMode')` an das Frontend gesendet.

```
..
if(data.verified) {
  socket.join(data.userId);
  console.log(`authenticate: user with id${data.userId} verified`);
  fetchOutputMode(data.userId, function(userId, data) {
    console.log(data);
    for(let i = 1; i <= data.length; i++) {
      app.io.in(userId).emit('updateOutputMode', {channel: i, mode: data[i-1]});
    }
  });
  return callback(null, true);
}
..
..
```

Die `postAuthenticate()` Funktion wird ebenfalls nach der Authentifizierung ausgeführt. Dadurch werden Event Listener für den entsprechenden authentifizierten Socket erstellt. Somit können nur korrekt eingeloggte Nutzer Mit dem Server interagieren und Informationen aus der Datenbank auslesen oder in die Datenbank speichern.

```
..
function postAuthenticate(socket, data) {
  socket.on('receiveCipherKey', function() {
    let userId = getRoom(socket);
    fetchCipherKey(userId, function(data) {
      app.io.in(userId).emit('receiveCipherKey', data);
    });
  });
  ..
  socket.on('storeOutputMode', function(data) {
    let userId = getRoom(socket);
    storeOutputMode(userId, data, function(res) {
      app.io.in(userId).emit('storeOutputMode', res);
    });
  });
}
..
..
```

Registrierung

Die Registrierung funktioniert aus Sicht des Frontend nach dem gleichen Prinzip wie die Authentifizierung. Der Event Handler `.on('register')` führt die Funktion `registerUser()` aus. Dabei wird erst überprüft ob der User nicht schon mit dieser E-Mail-Adresse registriert ist. Wenn die Funktion `fetchEmail()` ein Resultat liefert, ist schon ein User mit dieser Adresse registriert und es wird lediglich eine Meldung an die Callback Funktion zurückgegeben.

```
..
if(result.length !== 0) {
  console.log('registerUser: user already exists');
  return callback({status: 0, info: 'user already exists'});
}
..
```

Ansonsten wird ein `salt` mit zufälligen Werten erzeugt und zusammen mit dem gesendeten Passwort ein `passwordHash` generiert. Diese Werte werden inklusive der übermittelten E-Mail-Adresse und einem zufällig generierten `cipherKey` als neuen Account in der Datenbank gespeichert. Danach werden zusätzlich noch Datensätze für die Konfiguration und die einzelnen Kanäle mit Standardwerten erzeugt. Außerdem wird bei der erfolgreichen Registrierung ein Statuscode und eine entsprechende Info über `.emit('register')` an das Frontend gesendet.

```
..
return callback({status: 1, info: 'user account registration successful'});
..
```

7.2.3 Schwierigkeiten und Herausforderungen

Verwirrung bei den Crypto Methoden

Bei der Implementierung der Verschlüsselungsfunktionen bin ich auf ein Problem gestossen, welches mich einiges an Zeit gekostet hat. Am Anfang des Projekts wurde im Programmcode noch kein Initialvektor für die Verschlüsselung verwendet. Deshalb wurde die Methode `.createDecipher()` von der offiziellen Krypto-Bibliothek von node.js verwendet. Diese benutzt als Parameter einen Algorithmus sowie ein Passwort. Beim Testen der Ver- und Entschlüsselung zwischen dem Fernsteuergerät und dem Server wurden die übermittelten Daten aber nie korrekt ver- und entschlüsselt. Bei der Fehler suche habe ich alles Mögliche versucht und den Fehler an vielen verschiedenen Stellen vermutet. Bei der Recherche im Internet bin ich dann schlussendlich auf den entscheidenden Hinweis gestossen. [30] Aus dem Passwort Parameter wird in der Methode intern ein Initialvektor und der eigentliche Schlüssel erzeugt. Dazu wird die Routine `EVP_BytesToKey` [31] aus der OpenSSL Bibliothek verwendet. Die Ursache war also, dass die Verschlüsselung auf Seite des Fernsteuergeräts nur den rohen `cipherKey` verwendet, aus diesem aber auf dem Server ein anderer Schlüssel erzeugt wird. Nach dieser Erkenntnis wurde dann auf Server sowie auf Seite des Fernsteuergeräts eine Verschlüsselung mit Initialvektor und Schlüssel verwendet. Bei der Methode `.createDecipherIV()` kommt die OpenSSL Routine nicht mehr zum Einsatz und der rohe Schlüssel wird unverändert für den Algorithmus verwendet.

```
..
function decrypt(encryptdata, iv, key) {
  let decipher = crypto.createDecipheriv('aes-128-cbc', key, iv);
  decipher.setAutoPadding(false);
  let cleardata = decipher.update(encryptdata);
  cleardata = Buffer.concat([ cleardata, decipher.final() ]);
  return cleardata;
}
..
```

Asynchronität in der Schleife

An einer Stelle im Servercode werden die Einstellungen für alle Kanäle in der Datenbank gespeichert. Dabei wird eine for Schleife durchlaufen und für jeden Kanal wird eine asynchrone Datenbankabfrage gestartet. Da diese Anfragen alle asynchron sind kann nicht mit Sicherheit angenommen werden, dass die letzte Abfrage in der Iteration auch diejenige ist welche als letztes abgeschlossen wird. Da nun aber nach der erfolgreichen Speicherung von allen Kanälen eine Rückmeldung gesendet werden soll muss das zuverlässig überprüft werden können. Das folgende Beispiel zeigt auf wie eine solche Überprüfung implementiert werden kann:

```
..
let inserted = 0;
for(let i = 1; i <= MAX_CHANNELS; i++) {
  ..
  dbConn.query(query, values, function(err, result) {
    ..
    if (++inserted == MAX_CHANNELS) {
      return callback({status: 1, info: 'settings successful stored'});
    }
  });
}
..
```

Bevor die Schleife durchlaufen wird, wird die Variable `inserted` mit dem Wert `0` initialisiert. In der Callback Funktion der Abfrage wird dann diese Variable um `1` erhöht und anschliessend verglichen ob schon alle anderen Abfragen abgeschlossen sind. Auf diese Weise kann zuverlässig überprüft ob alle asynchronen Abfragen schon abgearbeitet wurden.

7.2.4 Programmbibliotheken

Für die Implementation wurden sowohl einige Module der node Standard Bibliothek, als auch externe Module verwendet, welche über den node package manager (npm) installiert werden können.

integrierte Module

- **http:** Für eine vereinfachte Nutzung von HTTP Nachrichten.
- **https:** Für eine vereinfachte Nutzung von HTTPS Nachrichten.
- **fs:** Methoden für das Schreiben und Laden vom lokalen nicht flüchtigen Speicher. Wird in diesem System zum laden der TLS/SSL Zertifikate benötigt.
- **crypto:** Methoden zum ver- und entschlüsseln. Ausserdem werden Methoden zum erzeugen von zufälligen Werten bereitgestellt.

zusätzliche Module

- **express:** Framework für die einfache Erstellung von Webanwendungen.
- **body-parser:** Middleware für das express Framework. Wird dazu verwendet um die Nutzdaten aus der HTTP Nachricht als Buffer zu speichern.
- **mysql:** Für die Nutzung der MySQL Datenbank.
- **socket.io:** Für die bidirektionale eventbasierte Echtzeitkommunikation.
- **socketio-auth:** Modul für die Authentifizierung mit socket.io.

7.3 Frontend

Die Implementation im Bereich Frontend kann grob in drei Bereiche aufgeteilt werden. Ein Bereich ist der eigentliche Programmcode, welcher in JavaScript implementiert ist. In diesem Bereich geht es um die Kommunikation über die WebSockets sowie um das Auslesen und Einfügen von Bezeichnungen und Werten auf der Benutzeroberfläche. Ein zweiter Bereich ist die Umsetzung des Layouts sowie der Navigation durch die Oberfläche und die Bezeichnung der einzelnen Elemente im DOM. In einem dritten Bereich geht es schlussendlich noch um das Webdesign mit dem generierten HTML und die Umsetzung der Oberfläche mit dem Bootstrap Framework.

7.3.1 JavaScript

Wie auch schon im Kapitel der Serverimplementation erwähnt, kann es in Bezug auf die Kommunikation über die WebSockets einige Überschneidungen mit Erläuterungen geben, welche eher ins Kapitel Server gehören. Auch werden in diesem Kapitel wieder Sequenzdiagramme verwendet, welche im Zusammenhang mit der Serverimplementation stehen. Die nummerierten Kreise verdeutlichen diese direkten Verknüpfungen mit den Sequenzdiagrammen der Serverimplementation.

Socket-Events vom Server

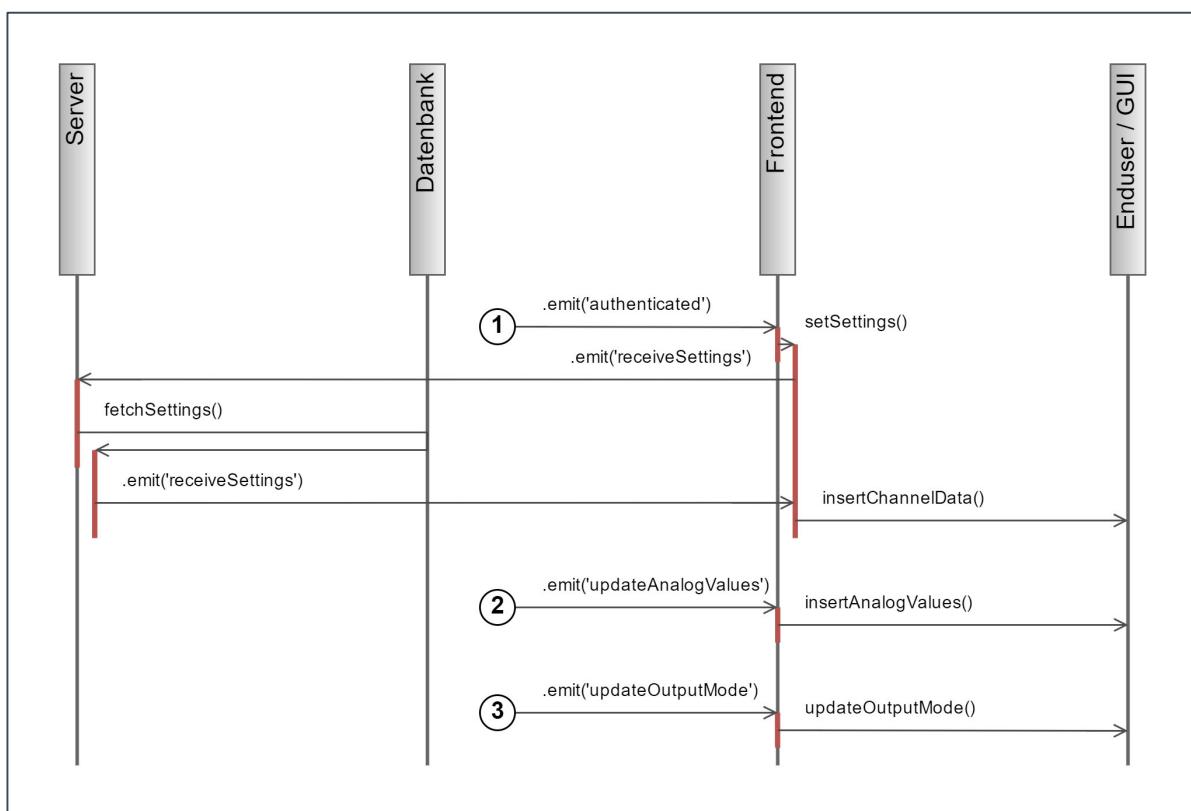


Abbildung 20 Sequenzdiagramm Server zu Frontend

Nach einer erfolgreichen Authentifizierung beim Server wird von diesem aus über die WebSocket Kommunikation das ‘authenticated’ Ereignis ausgelöst. Der Event Listener vom Frontend führt daraufhin die `setSettings()` aus. Diese wiederum schickt ein Ereignis zum Server und verlangt die Einstellungen, welche in der Datenbank gespeichert sind. Nachdem diese beim Server von der Datenbank ausgelesen worden sind, werden sie mit `.emit('receiveSettings')` an das Frontend gesendet. Nach dem Empfang werden die Einstellungen in einem globalen Speicherobjekt gespeichert.

```
..
export function setSettings(callback) {
  let socket = App.getSocket();

  socket.emit('receiveSettings');
  socket.on('receiveSettings', function(data) {
    settings = data;
    console.log(settings);
    //remove listener after one time (because every function-call would instanciate
    another listener)
    socket.off('receiveSettings');
    return callback();
  });
}
..

```

Hier sei noch erwähnt, dass der mit dem Funktionsaufruf erstellte Event Listener nach dem Empfang der Einstellungen wieder entfernt wird. Dadurch wird vermieden, dass bei einem erneuten Aufruf der `setSettings()` Funktion ein zweiter identischer Event Listener erstellt wird und somit auch beide die Einstellungen empfangen und speichern würden.

Nach der Speicherung der Einstellungen wird auf der Benutzeroberfläche mit der Funktion `insertChannelData()` für jeden Kanal die Sichtbarkeit, die Bezeichnungen von den Ein- und Ausgängen sowie die Einheitenbezeichnung der analogen Werte gesetzt.

```
..
export function insertChannelData() {
  for(let i = 1; i <= App.CHANNELS; i++) {
    let channel = settings.channel[i - 1];
    let cstr = `#channel-output${i}-`;
    let qstr;

    qstr = cstr + 'row';
    assignVisibleState(qstr, channel.active);

    qstr = cstr + 'label';
    $(qstr).text(channel.outputLabel);

    cstr = `#channel-input${i}-`;

    qstr = cstr + 'row';
    assignVisibleState(qstr, channel.active);

    qstr = cstr + 'label';
    $(qstr).text(channel.inputLabel);

    qstr = cstr + 'unit';
    $(qstr).text(channel.unitName);
  }
}
..

```

Der Event Listener `.on('updateAnalogValues')` führt bei einem entsprechenden ankommenden Ereignis die `insertAnalogValues()` Funktion aus. Dabei werden für jeden Kanal die übermittelten Werte auf der Benutzeroberfläche überschrieben. Dieses Ereignis wird immer dann ausgelöst, wenn beim Server neue Daten vom Fernsteuergerät ankommen.

```
..  
for(let i = 1; i <= data.length; i++) {  
    let qstr = `#channel-input${i}-value`;  
    $(qstr).text(data[i - 1]);  
}  
..
```

Die Funktion `updateOutputMode()` wird bei einem ankommenden `'updateOutputMode'` Ereignis ausgeführt. Das Ereignis wird nach einer erfolgreichen Authentifizierung, bei neuen Daten vom Fernsteuergerät sowie vor dem Senden von neuen Daten an das Fernsteuergerät ausgelöst. Die Funktion setzt die Anzeige der Buttons für die einzelnen Ausgänge auf den aktuellen Stand. es können 6 verschiedene Zustände angezeigt werden:

- **1:** Ausgang ist ausgeschaltet (weiss)
- **2:** Ausgang ist eingeschaltet (grün)
- **3:** ausstehender Ausschaltbefehl (blau)
- **4:** ausstehender Einschaltbefehl (blau)
- **5:** Ausgang wurde automatisch ausgeschalten (orange)
- **6:** Ausgang wurde automatisch eingeschalten (orange)

Die ausstehenden Befehle wechseln den Zustand auf Aus oder Ein bevor neue Daten an das Fernsteuergerät gesendet werden.

Click-Events

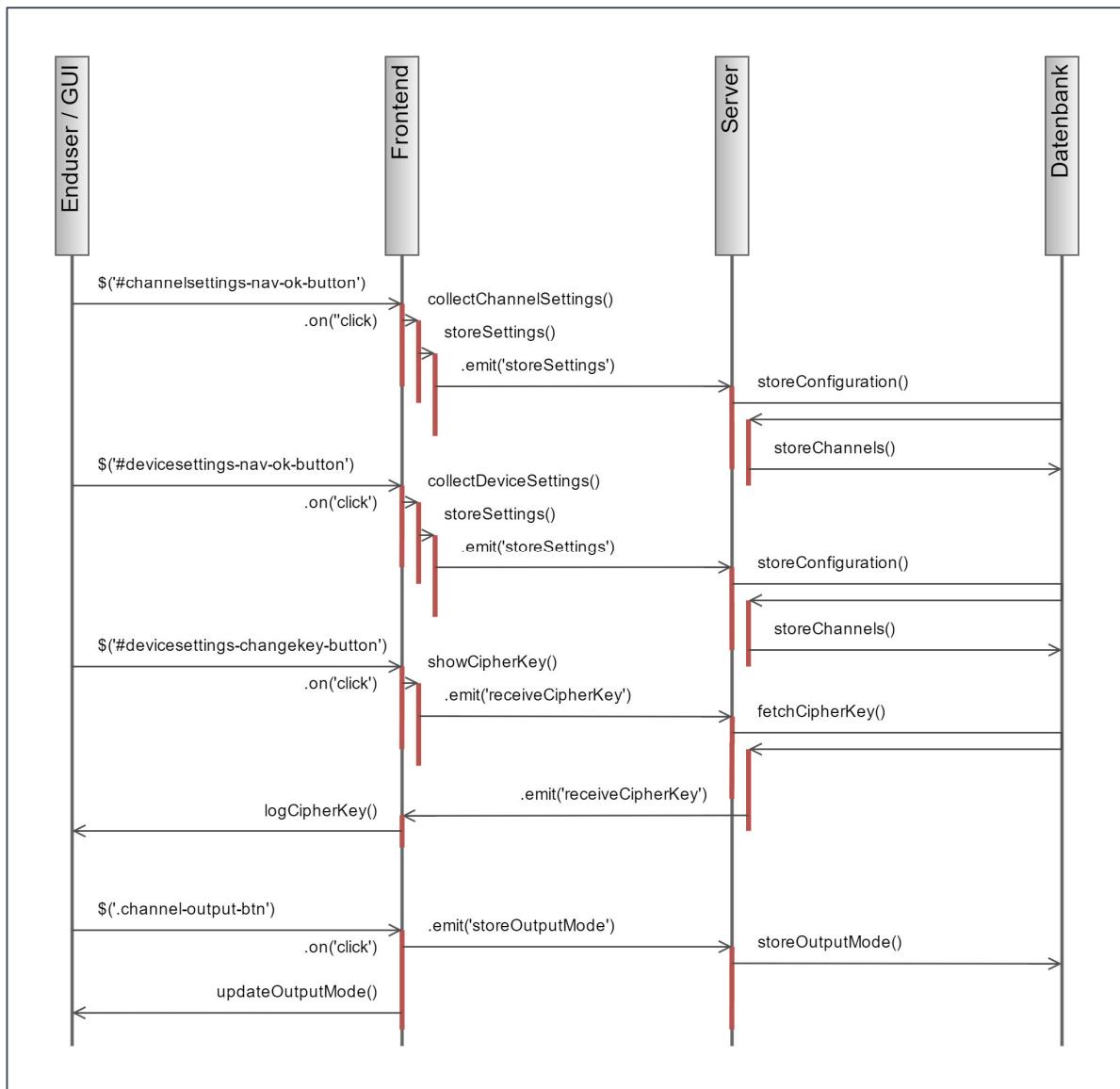


Abbildung 21 Sequenzdiagramm Click-Events

Mit einem Klick auf den Ok Button in den Kanaleinstellungen werden die in den Formularfeldern eingegebenen Daten in der Benutzeroberfläche ausgelesen und im globalen `settings.channel` Speicherobjekt gespeichert. Zusätzlich werden die Einstellungen an den Server übermittelt und dort in der Datenbank gespeichert.

Nach dem genau gleichen Prinzip funktioniert die Sammlung und Speicherung der Geräteeinstellungen.

In den Geräteeinstellungen kann außerdem mit einem Klick auf den Button «Schlüssel anzeigen» der Chiffrierschlüssel angezeigt werden, welchen man für die Konfiguration des Fernsteuergeräts benötigt.

Bei einem Klick auf einen Button der Ausgänge im Hauptmenu wird der Zustand des Buttons ausgelesen, entsprechend angepasst und anschliessend wird der neue Zustand angezeigt. Zusätzlich wird der neue Zustand auch an den Server gesendet und dort in der Datenbank gespeichert. Die Zustände der Buttons verändern sich bei einem Klick wie folgt:

- ausgeschaltet -> ausstehender Einschaltbefehl
- eingeschaltet -> ausstehender Ausschaltbefehl
- ausstehender Ausschaltbefehl -> eingeschaltet
- ausstehender Einschaltbefehl -> ausgeschaltet
- Ausgang wurde automatisch ausgeschalten -> ausstehender Einschaltbefehl
- Ausgang wurde automatisch eingeschalten -> ausstehender Ausschaltbefehl

Einige Funktionen, welche bei einem Klick ausgeführt werden sind im Sequenzdiagramm nicht aufgeführt, da sie keine Kommunikation mit dem Server zur Folge haben. Bei einem Klick auf Ok oder Abbrechen im Hauptmenu der Einstellungen wird die Funktion `insertChannelData()` ausgeführt. Dabei werden veränderte Einstellungen der Kanäle im Hauptmenu aktualisiert bevor durch die Navigation dieses angezeigt wird. Klickt man im Hauptmenu der Einstellungen auf die jeweilige Einstellungskategorie wird entsprechend die Funktion `insertChannelSettings()`, `insertDeviceSettings()` oder `insertAccountSettings()` ausgeführt. Wenn man sich einloggt und zum ersten Mal in die Einstellungen navigiert werden dadurch die Einstellungen, welche vorerst nur im Speicherobjekt `settings` vorhanden sind, in das jeweilige Menu abgefüllt. Außerdem kann man, falls man aus Versehen falsche Werte in den Einstellungen angegeben hat und auf Abbrechen klickt, durch einen erneuten Klick die alten Einstellungen wiederherstellen.

7.3.2 Layout und Navigation

Die Benutzeroberfläche ist in einzelne Panes aufgeteilt. Bei diesen Panes handelt es sich um HTML div Elemente welche alle eine eindeutige ID besitzen. Jedes Pane besitzt eine horizontale Navigationsleiste welche oben angezeigt wird. Je nach Pane werden in der Navigationsleiste andere Texte und Buttons angezeigt. Alle diese Buttons besitzen auch eine eindeutige ID.

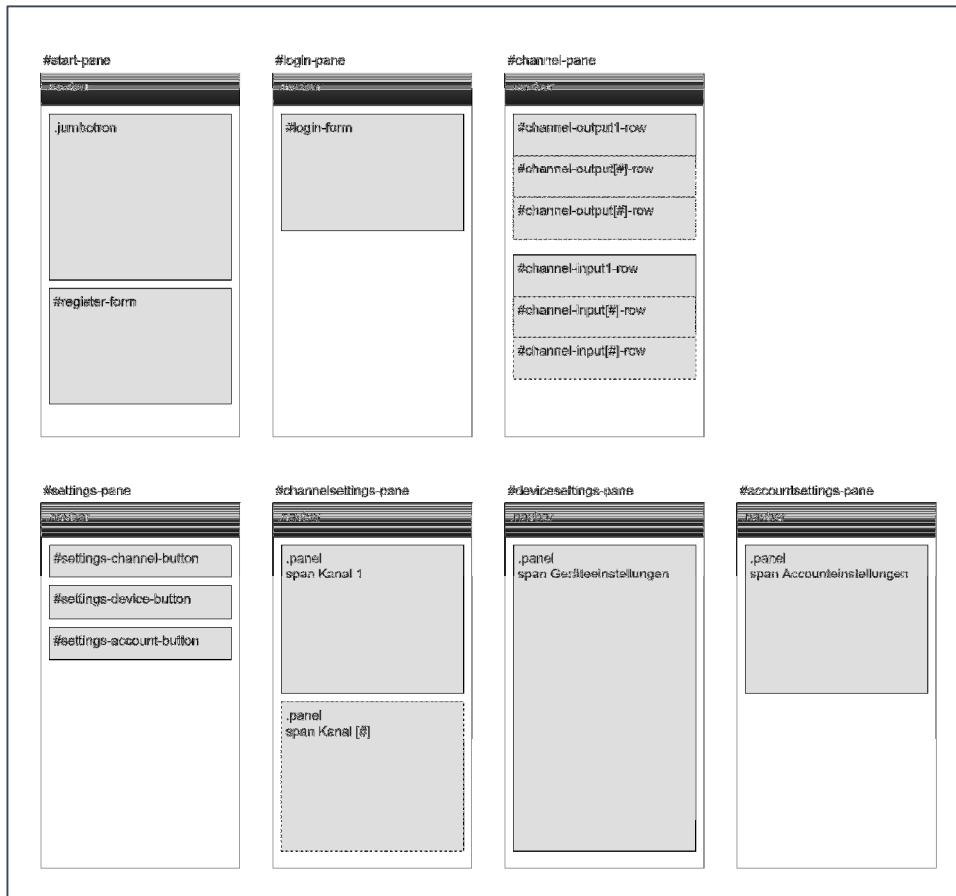


Abbildung 22 Übersicht Layout Benutzeroberfläche

Wird die Weboberfläche aufgerufen werden alle Panes komplett geladen. Auch werden alle Elemente für die Ein- und Ausgänge der Kanäle geladen. Grundsätzlich ist das HTML statisch und es werden zur Laufzeit keine Elemente dem DOM hinzugefügt oder davon entfernt. Jedoch sind die einzelnen Panes mit der CSS-Klasse `.hidden` versehen. Diese setzen die Eigenschaft `display` auf `none` und werden somit auf der Benutzeroberfläche nicht angezeigt.

Die Navigation ist also so realisiert, dass beim Klicken des jeweiligen Navigationsbuttons die `.hidden` Klasse bei den Panes entfernt oder hinzugefügt werden.

```
$( '#start-nav-login-button' ).on( 'click', function() {
  $( '#start-pane' ).toggleClass( 'hidden' );
  $( '#login-pane' ).toggleClass( 'hidden' );
});
```

Zu erwähnen ist noch, dass erst nach einer erfolgreichen Authentifizierung das Hauptmenu angezeigt werden kann.

```
...
if(!authenticated) {
  $('#login-pane').toggleClass('hidden');
  $('#channel-pane').toggleClass('hidden');
}
...
```

7.3.3 Design der Oberfläche

Im Bereich Webdesign geht es vor allem um das Schreiben des HTML Code, sowie die Konfiguration des Aussehens mit CSS. Beim Design von diesem Prototyp musste erfreulicherweise nur sehr wenige eigene CSS Eigenschaften definiert werden. Bootstrap, welches als CSS-Framework in diesem Projekt eingesetzt wird, nimmt da viel Arbeit ab.

Responsive Design

Als Beispiel für die Umsetzung des Designs soll der Aufbau der Kanaleinstellung dienen. Das Design ist Responsive. Das bedeutet, dass sich die Seitenelemente je nach Grösse des Bildschirms anpassen. Bei der Darstellung in einem verkleinerten Browserfenster auf dem Desktop werden die Überschriften die Eingabefelder links dargestellt. Bei der Darstellung auf einem Smartphone werden die Überschriften jedoch über den Eingabefeldern angezeigt.

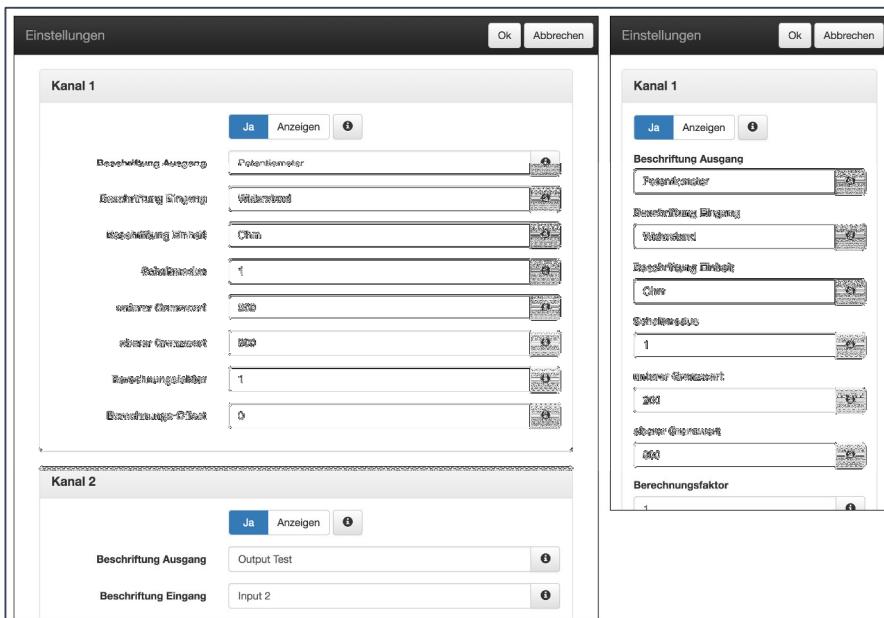


Abbildung 23 Vergleich Responsive Webdesign

Ermöglicht wird diese automatische Anpassung durch das Grid-System von Bootstrap. Die Elemente sind in einem Gitter angeordnet. Das Gitter ist in 12 gleich breite Teile unterteilt. Mit CSS Klassen, welche von dem Bootstrap Framework definiert wurden, kann man bestimmen wie viel Platz für die einzelnen Elemente verbraucht werden sollen. Weist man einem Element beispielsweise die Klasse `.col-sm-4` zu, wird für dieses Element 4/12 der gesamten Breite an Platz zur Verfügung gestellt. Zusätzlich bestimmen diese CSS Klassen auch, für welche Gerätekategorie diese Eigenschaften gelten sollen. Durch die Klasse `.col-sm-4` wird also bestimmt, dass bis zu der Gerätekategorie «small» 4/12 vom Platz verbraucht werden sollen. Die Kategorie «small» entspricht dabei einem Tablet. Wenn jetzt

aber weniger Platz für die Darstellung zur Verfügung steht, ist die CSS Eigenschaft `.col-sm-4` nicht mehr gültig. Es wird dann für jedes Element die gesamte Breite des Bildschirms verwendet. Deshalb unterscheidet sich die Darstellung auf dem Smartphone.

```
..
.form-group
  label.col-sm-4.control-label Beschriftung Ausgang
  .col-sm-8
    .input-group
      input(id = ...).form-control
      .input-group-btn
        button(class = ...).btn.btn-default
          spanglyphicon.glyphicon-info-sign
..
..
```

HTML Code mit jade

Jade ist eine Template Engine welche zum generieren des HTML eine saubere und übersichtliche Syntax benutzt. Mit dieser Syntax kann viel Schreibarbeit gespart werden und der Code ist um einiges übersichtlicher als der entsprechende HTML Code. Die Syntax ist «whitespace sensitive». Verschachtelungen von HTML Elementen werden beispielsweise durch einen Tabulator festgelegt. Außerdem wird auf die schliessenden Tags verzichtet. Das Standardelement `<div>` kann in jade weggelassen werden. So wird beispielsweise aus `.someClass <div class = "someClass"></div>`. Es ist auch möglich JavaScript einzubinden um beispielsweise mit Schleifen mehrere Elemente zu erzeugen.

```
..
- var channels = 6
..
- for(var i = 1; i <= channels; i++) {
..
.input-group
  input(id = 'channelsettings-channel'+i+'-outputlabel-value', type = 'text').form-
control
..
- }
```

In diesem Projekt wurden aus 347 Zeilen jade Code 1034 Zeilen HTML Code generiert. Außerdem konnte durch die bessere Übersicht sicherlich auch einiges an Zeit eingespart werden.

7.4 nicht implementierte Funktionalitäten

In der Aufgabenstellung dieser Diplomarbeit wurde ein funktionierender Prototyp als Mussziel definiert. Dieses Ziel konnte erreicht werden. Es konnten sogar bis auf eine Ausnahme alle aufgeführten Muss- und Sollziele erreicht werden. Es fehlen jedoch noch einige wichtige und wünschenswerte Funktionalitäten im System. Diese konnten mangels Zeit nicht mehr umgesetzt werden und werden hier nun erläutert.

Fernsteuergerät

Die Kommunikation über ein Wi-Fi Modul ist noch nicht möglich. Dies ist auch das erwähnte nicht erreichte Sollziel aus der Aufgabenstellung. Ansonsten ist die Software des Mikrocontrollers für einen Prototypen schon auf einem guten Stand.

Server / Frontend

Der nicht bestandene Test betrifft die Anzeige der analogen Werte auf der Benutzeroberfläche. Nach dem Login werden diese Werte erst angezeigt, wenn sie vom Fernsteuergerät an den Server übermittelt wurden. Damit die aktuellsten Werte direkt angezeigt werden können, müssten sie aus der Datenbank ausgelesen werden und anschliessend an das Frontend gesendet werden.

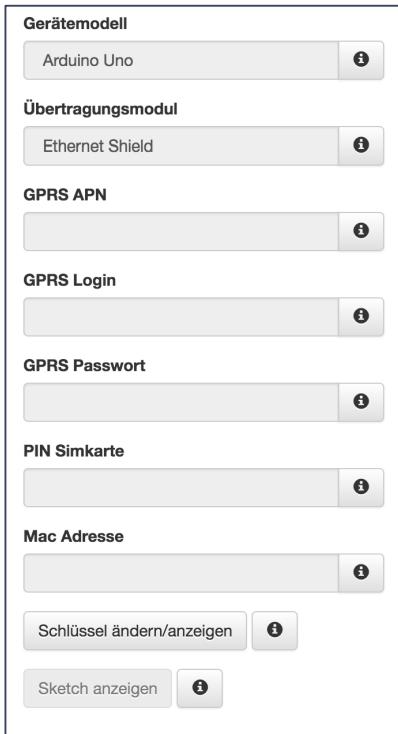
Wenn ein Ausgang automatisch geschalten wurde, wird dies auf der Benutzeroberfläche durch den Status des Buttons visuell dargestellt. Es fehlt aber noch eine zusätzliche Benachrichtigung an den Endnutzer. Dafür sollte es die Möglichkeit der Benachrichtigung per E-Mail, SMS und Push Nachricht geben.

Es sollte die Möglichkeit bestehen alle gespeicherten Eingangswerte für die jeweiligen Kanäle anzuzeigen. Momentan werden diese Werte mit Zeitstempel in der Datenbank gespeichert. Sie müssten jetzt noch in geeigneter Form auf der Weboberfläche angezeigt werden können.

Auf der Weboberfläche können in den Eingabefeldern beliebige Werte eingegeben werden. Es fehlt noch eine Validierung für korrekt eingetragene Werte. Bei nicht zulässigen eingegebenen Werten sollte der Endnutzer mit einer Warnung informiert werden. Insbesondere eine Passwortvalidierung ist wichtig. Auch fehlt noch an einigen Stellen eine Rückmeldung an den Endnutzer. Zum Beispiel bei der Registrierung oder bei einem erfolglosen Loginversuch.

Die Darstellung der analogen Werte ist noch nicht korrekt. Es fehlt noch die Umrechnung für eine korrekte Darstellung. In Kapitel 8.3 wird beschrieben wie die Umrechnung funktioniert.

Damit die Konfiguration des Fernsteuergeräts vereinfacht werden kann, sollte der Programmcode für den Mikrokontroller in den Geräteeinstellungen automatisch erzeugt werden können. Die Eingabefelder dafür sind in der Weboberfläche schon vorhanden. Die Funktionalität dafür muss aber noch implementiert werden.



The screenshot shows a configuration interface for a device. It includes fields for:

- Gerätemodell:** Arduino Uno
- Übertragungsmodul:** Ethernet Shield
- GPRS APN:** (empty field)
- GPRS Login:** (empty field)
- GPRS Passwort:** (empty field)
- PIN Simkarte:** (empty field)
- Mac Adresse:** (empty field)
- Schlüssel ändern/anzeigen:** (button)
- Sketch anzeigen:** (button)

Abbildung 24 Geräteeinstellungen

In den Accounteinstellungen kann die E-Mail-Adresse, und das Passwort noch nicht geändert werden. Die Eingabefelder dafür sind aber schon in der Weboberfläche integriert.

Auf der Weboberfläche gibt es zu allen Eingabefeldern einen Infobutton. Dieser hat momentan noch keine Funktion. Bei einem Klick auf den Button sollte eine Kurzhilfe zu dem jeweiligen Eingabefeld angezeigt werden.

8 Bedienung der Benutzeroberfläche

In diesem Kapitel werden einige Bereiche der Konfiguration und Bedienung der Weboberfläche beschrieben.

8.1 Schalten der Ausgänge

Im Hauptmenü der Oberfläche kann durch einen Klick, oder das Berühren der Buttons, der Zustand der entsprechenden Ausgänge auf dem Fernsteuergerät geändert werden. Eine Betätigung bedeutet aber nicht, dass der Zustand sofort übertragen wird und das Gerät den Ausgang schaltet. Um den effektiven Zustand der einzelnen Ausgänge zu verdeutlichen sind die Buttons farbig markiert.

weisser Button

Der Ausgang auf dem Fernsteuergerät ist ausgeschaltet.

grüner Button

Der Ausgang auf dem Fernsteuergerät ist eingeschaltet.

blauer Button

Ein noch ausstehender Aus- oder Ein-Befehl. Das Fernsteuergerät ist also noch nicht in diesem Zustand. Der Zustand ändert sich auf Aus oder Ein sobald sich das Fernsteuergerät mit dem System verbindet und neue Befehle entgegennimmt.

orangener Button

Das Fernsteuergerät hat durch die automatische Schaltung den Ausgang ein- oder ausgeschalten.

8.2 Geräteeinstellungen

Die Geräteeinstellungen beziehen sich auf das Fernsteuergerät. Geänderte Werte werden erst aktiviert, wenn sich das Fernsteuergerät verbindet und neue Befehle und Einstellungen entgegennimmt.

Sendeintervall

Mit diesem Wert wird bestimmt wie lange das Fernsteuergerät wartet, bis es sich erneut mit dem System verbindet. Der Wert wird in Millisekunden angegeben.

Fehlerintervall

Dieser Wert bestimmt wie lange bei einem Verbindungsversuch des Fernsteuergerätes gewartet werden soll bevor ein weiterer Versuch gestartet wird. Der Wert wird in Millisekunden angegeben.

Fehlversuche:

Hier wird angegeben nach wie vielen erfolglosen Verbindungsversuchen das Fernsteuergerät komplett neu gestartet werden soll.

8.3 Kanaleinstellungen

Einige Kanaleinstellungen beziehen sich auf das Fernsteuergerät. Geänderte Werte werden erst aktiviert, wenn sich das Fernsteuergerät verbindet und neue Befehle und Einstellungen entgegennimmt.

Die nachfolgend erwähnten Einstellungen beziehen sich auf die automatische Schaltung der Ausgänge beim Fernsteuergerät. Das Fernsteuergerät wandelt die ankommende Spannung (0V – 5V) an den analogen Eingängen in einen Wert zwischen 0 und 1023 um. Je nach Konfiguration wird beim überschreiten eines Eingangswerts der entsprechende Ausgang automatisch geschalten.

unterer Grenzwert:

Beim unterschreiten dieses Werts wird der entsprechende Ausgang je nach Schaltmodus geschalten.

oberer Grenzwert:

Beim überschreiten dieses Werts wird der entsprechende Ausgang je nach Schaltmodus geschalten.

Schaltmodus:

Es können 8 verschiedene Schaltmodi eingestellt werden. Die folgende Tabelle benutzt Pfeile für eine klarere Darstellung der einzelnen Modi. « $\uparrow \Rightarrow$ EIN» hat folgende Bedeutung: Beim überschreiten des oberen Grenzwerts wird der Ausgang eingeschaltet. « $\downarrow \Rightarrow$ X» hat folgende Bedeutung: Das unterschreiten des unteren Grenzwerts hat keinen Einfluss auf den Ausgangszustand.

- **Modus 0:** $\downarrow \Rightarrow X$, $\uparrow \Rightarrow X$ (deaktivierter Automatikmodus)
- **Modus 1:** $\downarrow \Rightarrow AUS$, $\uparrow \Rightarrow EIN$
- **Modus 2:** $\downarrow \Rightarrow X$, $\uparrow \Rightarrow EIN$
- **Modus 3:** $\downarrow \Rightarrow AUS$, $\uparrow \Rightarrow X$
- **Modus 4:** $\downarrow \Rightarrow EIN$, $\uparrow \Rightarrow AUS$
- **Modus 5:** $\downarrow \Rightarrow EIN$, $\uparrow \Rightarrow X$
- **Modus 6:** $\downarrow \Rightarrow X$, $\uparrow \Rightarrow AUS$
- **Modus 7:** $\downarrow \Rightarrow EIN$, $\uparrow \Rightarrow EIN$

Die nachfolgend erwähnten Einstellungen beziehen sich auf die Anzeige der analogen Werte auf der Weboberfläche. Dabei geht es um die Umrechnung der übertragenen Werte vom Fernsteuergerät. Wenn angenommen ein Temperatursensor beispielsweise eine Temperatur von 30 °C misst, liefert er am analogen Eingang des Geräts eine Spannung von 3 Volt. Diese Spannung wird vom Gerät in den Wert 615 umgewandelt und auch so übertragen. Hier ist noch anzumerken, dass hier nicht auf die Auswahl von Vorwiderständen für die Sensoren und deren Berechnung eingegangen wird. Mit dem Berechnungsfaktor sowie dem Offset kann auf der Oberfläche der Wert korrekt angezeigt werden.

Berechnungsfaktor

Um den Berechnungsfaktor zu bestimmen kann folgende Formel angewendet werden:

$$(highUnitValue - lowUnitValue) / 1023 = factor$$

Dabei steht `highUnitValue` für den höchsten gemessenen Wert beim Sensor für den eingestellten Bereich. Mit `lowUnitValue` ist der tiefste gemessene Wert gemeint. Der `highUnitValue` Wert sollte also den übertragenen Wert 1023 zur Folge haben. Der `lowUnitValue` Wert einen übertragenen Wert von 0.

Berechnungs-Offset

Mit dem Offset ist die Verschiebung des Wertes für die korrekte Anzeige gemeint. Dieser Offset entspricht ganz einfach dem `lowUnitValue` Wert vom Sensor. Also dem Wert der einen übertragenen Wert von 0 zur Folge hat.

9 Schlusswort und Ausblick

Die Umsetzung dieses Projekts war für mich eine spannende und vielseitige Herausforderung. Manche Stunden bereiteten mir Freude und motivierten mich immer weiter zu machen. Es gab aber auch Momente welche an meinen Nerven gezehrt und meine Geduld auf die Probe gestellt haben. Erfreulicherweise konnte ich alle Hürden überwinden und ich kann nun mit Stolz ein System vorzeigen welches meine vielfältigen Fähigkeiten widerspiegelt.

Neben der eigentlichen Implementierung hat mir vor allem auch das Ausarbeiten der Architekturen Freude bereitet. Gerade auch das Zusammenspiel der einzelnen Komponenten untereinander finde ich sehr spannend. Zu wissen was alles zwischen einem ankommenden 3V Signal am Mikrokontroller und der Anzeige des entsprechenden Werts auf der Weboberfläche passiert, ist ein schönes Gefühl.

Auch habe ich während dieser Arbeit viel Neues dazu gelernt. Es war für mich das erste richtige Projekt bei dem ich für einen Mikrokontroller programmiert habe. Dabei lernte ich den sparsamen Umgang mit dem Arbeitsspeicher und ich konnte neue Erfahrungen im Bereich der Low-Level-Programmierung sammeln. Neue Erkenntnisse konnte ich auch im Bereich der Verschlüsselung und bei den Authentifizierungsmechanismen gewinnen. Gerade im Bereich der Internet of Things ist die sichere Übertragung von Daten nicht immer gegeben. Doch in diesem System werden die Daten durchgängig sicher verschlüsselt. Mein Wissen über die Programmierung mit node.js konnte ich ebenfalls vertiefen. Insbesondere konnte ich im Bereich der asynchronen Programmierung und den Callback Funktionen Erfahrungen sammeln. Ich kenne nun einige Tücken und Eigenheiten welche ein asynchroner Code mit sich bringt und weiss nun wie man damit umgeht. Schliesslich habe ich noch den Umgang mit WebSockets sowie dem CSS-Framework Bootstrap gelernt.

Nach all diesen neuen Erfahrungen würde ich bei einem nächsten ähnlichen Projekt jedoch einiges anders machen. Die Entwicklungsplattform, mit welcher der Arduino Mikrokontroller programmiert werden kann, ist mehr schlecht als recht. Insbesondere fehlen in dem Programm Funktionen fürs Debugging. Der Codeeditor selbst ist auch nicht das Gelbe vom Ei. Bei zukünftigen Arduino Projekten würde ich mich also erst auf die Suche nach einer geeigneten Entwicklungsumgebung machen.

Obwohl das Bootstrap Framework sehr einfach zu lernen ist, fand ich den Umgang mit dem Grid-System teilweise ein wenig verwirrend und unschön. Da es viele Alternativen zu Bootstrap gibt werde ich mir in Zukunft sicher auch mal ein anderes Framework anschauen.

Die Zukunft für dieses Projekt ist auch noch offen. Es wurde ein funktionierender Prototyp umgesetzt dem aber noch einige wichtige Funktionalitäten fehlen. Mit ein wenig Aufwand ist es jedoch möglich das System auf einen Stand zu bringen bei dem es mit gutem Gewissen veröffentlicht werden kann. Neben der Weiterentwicklung der Software stehen beispielsweise Ideen wie das entwickeln eines Gehäuses im Raum. Die Pläne dafür könnten auch veröffentlicht werden. Das Gehäuse könnte dann mit einem 3D-Printer gedruckt werden.

Meine anfängliche Sorge, dass mir dieses Projekt nicht gefallen würde war völlig unbegründet. Im Nachhinein bin ich sehr froh, dass ich mich für dieses Projekt entschieden habe und ich kann diese Diplomarbeit für mich als Erfolg verbuchen.

10 Anhang

10.1 Testprotokolle

Fernsteuergerät	
Das Gerät verbindet sich automatisch ins Internet und ist bereit für das Empfangen und Versenden von Daten.	bestanden
<i>Bemerkung:</i>	
Das Gerät konfiguriert sich beim Start automatisch.	bestanden
<i>Bemerkung:</i>	
Die Werte von allen verfügbaren analogen Eingängen werden korrekt gespeichert.	bestanden
<i>Bemerkung:</i>	
Alle verfügbaren digitalen Ausgänge werden entsprechend der gespeicherten Werte geschalten.	bestanden
<i>Bemerkung:</i>	
Werden die Grenzwerte der entsprechenden Eingangssignale erreicht, wird der entsprechend konfigurierte Ausgang automatisch geschalten.	bestanden
<i>Bemerkung:</i>	

Tabelle 8 Testprotokoll Fernsteuergerät

Frontend	
Die korrekten Einstellungen werden nach Anmeldung angezeigt.	bestanden
<i>Bemerkung:</i>	
Das Betätigen der Buttons für Ausgänge ändert den Zustand.	bestanden
<i>Bemerkung:</i>	
Die Eingangswerte werden korrekt dargestellt und automatisch aktualisiert.	teils bestanden
<i>Bemerkung:</i> Die Eingangswerte werden nach dem einloggen erst angezeigt, wenn das Fernsteuergerät wieder Daten überträgt. Es müssten beim Login die aktuellsten Daten aus der Datenbank eingefügt werden.	
Die Buttonlabels können editiert werden.	bestanden
<i>Bemerkung:</i>	

Tabelle 9 Testprotokoll Frontend



Server	
Empfangene Daten werden in der Datenbank im korrekten Datensatz gespeichert.	bestanden
<i>Bemerkung:</i>	
Daten können von der Datenbank aus korrektem Datensatz ausgelesen werden.	bestanden
<i>Bemerkung:</i>	
Passwörter werden sicher und korrekt als Hashs in der Datenbank abgespeichert.	bestanden
<i>Bemerkung:</i>	
Authentifizierung mittels Anmeldedaten funktioniert.	bestanden
<i>Bemerkung:</i>	

Tabelle 10 Testprotokoll Server

Fernsteuergerät – Server	
Fernsteuergerät verschlüsselt Nutzdaten.	bestanden
<i>Bemerkung:</i>	
verschlüsselte Daten werden übertragen und vom Server empfangen.	bestanden
<i>Bemerkung:</i>	
Server entschlüsselt Daten korrekt.	bestanden
<i>Bemerkung:</i>	
Server verschlüsselt entsprechende Nutzdaten für die Antwort.	bestanden
<i>Bemerkung:</i>	
Verschlüsselte Daten werden übertragen und vom Fernsteuergerät empfangen.	bestanden
<i>Bemerkung:</i> Bei den übertragenen Daten wird sichergestellt das kein Byte mit dem Wert 255 übertragen wird.	
Fernsteuergerät entschlüsselt Daten korrekt.	bestanden
<i>Bemerkung:</i> Der Empfang von Bytes mit Wert 255 funktioniert beim GSM Modul nicht korrekt.	

Tabelle 11 Testprotokoll Fernsteuergerät - Server

Frontend – Server	
Eindeutige Authentifizierung des Endnutzers an Server funktioniert.	bestanden
<i>Bemerkung:</i>	
Direkte bidirektionale Verbindung zwischen Frontend und Server steht nach Anmeldung.	bestanden
<i>Bemerkung:</i> Die Verbindung steht schon vor der Anmeldung aber es können noch keine accountspezifische Daten empfangen oder gesendet werden.	

Tabelle 12 Testprotokoll Frontend – Server

Systemtest	
Anmeldung über Webbrowser eines Smartphones erfolgreich.	bestanden
<i>Bemerkung: Eventuelle Probleme beim Zertifikat auf der Android Plattform.</i>	
Automatisch aktualisierte Eingangswerte des Geräts werden angezeigt.	teils bestanden
<i>Bemerkung: Die Eingangswerte werden nach dem einloggen erst angezeigt, wenn das Fernsteuergerät wieder Daten überträgt. Es müssten beim Login die aktuellsten Daten aus der Datenbank eingefügt werden.</i>	
Das drücken von Buttons schaltet die Ausgänge des Geräts.	bestanden
<i>Bemerkung:</i>	
Konfiguration des Geräts von der Weboberfläche aus erfolgreich.	bestanden
<i>Bemerkung:</i>	
Notfallschaltung reagiert entsprechend der Konfiguration auch ohne Datenverbindung.	bestanden
<i>Bemerkung:</i>	
Beim aufstarten werden keine Ausgangszustände verändert.	bestanden
<i>Bemerkung: Bis zur kompletten Initialisierung sind alle Ausgänge ausgeschaltet.</i>	

Tabelle 13 Testprotokoll Systemtest

10.2 Arbeitsjournal

Tätigkeit	Aufwand (h:min)	Datum
Vorbereitung Dokumentation, Recherche Planungstools und Planungsvorgehen	02:45	07.03.16
Vorbereitung Dokumentation, Recherche Planungstools	04:00	08.03.16
Vorbereitung Dokumentation, Zielbestimmung formuliert	05:45	09.03.16
Recherche und Engineering Technologien / Hardware Auswahl	03:30	15.03.16
Konzept, Dokumentation, Anforderungen	06:00	16.03.16
Recherche Technologien, Dokumentation, Variantenanalyse, Besprechung, Ausarbeitung Protokoll, Risiken beschrieben	05:45	17.03.16
Recherche Technologien, Risiken, Anforderungsanalyse, Dokumentation	06:00	18.03.16
Recherche Technologien, Architektur Fernsteuergerät, Dokumentationsstruktur, Dokumentation	06:00	19.03.16
Recherche Technologien, Architektur Software, Vorbereitung Entwicklungsumgebung	06:45	21.03.16
Recherche Technologien, Architektur Sicherheit, PM	02:00	22.03.16
Recherche Technologien, Architektur Sicherheit, Beschaffung SIM-Karte, Dokumentation, PM	06:45	23.03.16
Architektur Software, Dokumentation	03:30	24.03.16
Recherche Technologien, Zusammenbau Prototyp, Implementation mC und Server	08:15	25.03.16
Implementation IO Test & Kommunikationstest, Zusammenbau Prototyp	04:30	26.03.16
Dokumentation, Architektur, Implementation Verschlüsselung	08:00	28.03.16
Implementation Verschlüsselung, Besprechung	04:00	29.03.16
Implementation Protokoll und Verschlüsselung	10:30	30.03.16
Projektmanagement Allgemein, Dokumentation, Architektur	03:30	01.04.16
Projektmanagement Allgemein, Dokumentation Testfälle, Implementation HTTPS	07:30	05.04.16
Recherche Technologien Session, Implementation Datenbank Session Authentifizierung	07:30	06.04.16
Implementation Socket.io	03:00	07.04.16
Mockup Frontend, Implementation Socket.io, Implementation Authentifizierung, Implementation Fernsteuergerät	12:00	08.04.16
Implementation Server, Implementation Kommunikation Fernsteuergerät	14:00	09.04.16

Implementation Funktionalität Fernsteuergerät	08:00	10.04.16
Fehlersuche Mikrokontroller	04:00	12.04.16
Fehlersuche Mikrokontroller	04:00	13.04.16
Projektmanagement Allgemein, Dokumentation	04:00	15.04.16
Dokumentation	00:15	18.04.16
Architektur Kommunikation, Dokumentation	05:00	19.04.16
Projektmanagement Allgemein, Recherche, Dokumentation	05:30	21.04.16
Evaluierung, Dokumentation	06:00	22.04.16
Evaluierung	01:00	23.04.16
Evaluierung, Dokumentation	05:15	27.04.16
Evaluierung, Dokumentation	05:45	28.04.16
Dokumentation Implementation	05:30	29.04.16
Formatierung Dokumentation, Dokumentation	05:30	30.04.16
Dokumentation Implementation, Implementation Fernsteuergerät und Server	07:00	01.05.16
Implementation Fernsteuergerät, Test Fernsteuergerät, Dokumentation Implementierung	06:45	02.05.16
Projektmanagement Allgemein	02:15	03.05.16
Projektmanagement Allgemein, Dokumentation Implementation	01:00	10.05.16
Dokumentation Implementation	02:15	11.05.16
Dokumentation Implementation. Implementation Server (Datenbankmodell)	07:30	12.05.16
Implementation Server, Implementation Frontend	08:15	13.05.16
Implementation Server, Implementation Frontend, Recherche Technologien	09:00	14.05.16
Implementation Frontend, Implementation Server	10:00	15.05.16
Implementation Frontend, Implementation Server	10:00	16.05.16
Implementation Frontend, Implementation Server	10:00	17.05.16
Implementation Frontend, Implementation Server, Testing	04:00	18.05.16
Dokumentation Implementation Server	06:00	19.05.16
Dokumentation Implementation Server	01:30	20.05.16
Projektmanagement Allgemein, Dokumentation Implementation Server	01:15	25.05.16
Dokumentation Implementation Server	09:30	26.05.16
Dokumentation Implementation Server und Frontend	08:00	27.05.16
Dokumentation Frontend, Management Summary, Glossar, Abschluss der Arbeit	11:00	28.05.16
Abschluss Dokumentation, Abschluss der Arbeit	09:00	29.05.16

Tabelle 14 Arbeitsjournal

10.3 Lernjournale

10.3.1 Woche 1

Arbeits- / Lernziele:

- Dokumentation vorbereitet
- Zielbestimmung formuliert
- Tools zur Planung ausgewählt
- Erste Planung mit den wichtigsten Punkten erstellt
- Konzept erarbeitet und dokumentiert

nicht erreichte Ziele:

Konzept erarbeitet und dokumentiert:

Ich habe zwar eine erste grobe Darstellung des Systems in visueller Form erarbeitet, aber diese ist noch nicht sehr genau. Eine saubere Dokumentation dazu konnte ich auch noch nicht niederschreiben.

Lernerfolg / Bemerkungen:

Bei der Suche eines geeigneten Tools, welches die Planung meiner Diplomarbeit unterstützen sollte, bin ich auf viele verschiedene Lösungen gestossen. Viele Tools sind aber für Teams und je nach dem eher für reine agile Softwareentwicklung ausgelegt. Da ich keine Erfahrungen mit diesen Tools habe, und ich es nicht für sinnvoll erachte während meiner Diplomarbeit ein solches und vermutlich eher ungeeignetes Werkzeug zu erlernen, habe ich mich entschieden einen einfachen Projektplan in Excel zu realisieren.

10.3.2 Woche 2

Arbeits- / Lernziele:

- Varianten erarbeitet und analysiert
- Hardware ausgesucht und bestellt
- Grundgerüst der Dokumentation mit allen wichtigen Kapiteln erstellt
- Konzepte und Lösungsansätze dokumentiert
- Testbetrieb der Hardware
- Anforderungsanalyse abschliessen
- Anwendungsszenarien beschreiben

nicht erreichte Ziele:

diverse:

Aufgrund der Besprechung mit meinem Diplomarbeitsbetreuer musste ich einige Wochenziele zurückstellen, beziehungsweise verwerfen. Insbesondere sind diese: Varianten erarbeitet und analysiert, Konzepte und Lösungsansätze dokumentiert, Hardware ausgesucht und bestellt, Testbetrieb der Hardware.

Lernerfolg / Bemerkungen:

Während der Sitzung mit dem Betreuer wurde mir bewusst, dass ich mein Vorgehen überdenken sollte und ich erst ein paar Schritte zurück machen sollte. Ich habe in einigen Punkten ein wenig zu vorschnell gehandelt. Ich muss mich jetzt vorerst mit der grundsätzlichen Anforderungsanalyse und der gleichen beschäftigen bevor ich Varianten bilden kann und weitere Schritte unternehmen kann.

Die Kritik wegen meinem «falschen» Vorgehen konnte ich gut nachvollziehen. Tatsächlich konnte ich durch die Überlegungen für verschiedene Anwendungsszenarien einige Anforderungen erkennen welche mir davor noch nicht klar waren. Ich bin mir sicher, dass sich während der Arbeit noch weitere Anforderungen herauskristallisieren werden. Ich möchte deshalb im Moment nicht mehr allzu viel Zeit in die Analyse stecken und mich mit der Architektur des Systems auseinandersetzen. Ich sollte eine gute Balance zwischen iterativem und linearem Vorgehen finden.

10.3.3 Woche 3

Arbeits- / Lernziele:

- Sim Karte für GSM Shield besorgen
- Dokumentation für Evaluierung einiger Technologien
- Architektur erarbeitet und dokumentiert
- Ausgangslage dokumentieren
- Kommunikationstest zwischen Arduino und node Server
- Arbeitsumgebung einrichten (Softwareentwicklung)

nicht erreichte Ziele:

Kommunikationstest:

Mein Ziel für den Test war Kommunikation über das GSM Shield mit verschlüsselten Nutzdaten und über das CoAP Protokoll. Mit dem GSM Shield konnte ich nur eine HTTP Verbindung herstellen. Es gibt zwar eine Bibliothek für das CoAP-Protokoll, diese funktioniert aber leider nur zusammen mit dem Ethernet Shield von Arduino. Die Suche nach geeigneten Programmbibliotheken und Informationen über das Protokoll und deren Implementierungen hat mich viel Zeit gekostet. Es ist sehr schade, dass die CoAP-Bibliothek nicht mit dem GSM Shield funktioniert. Mein Know-how in C++ und das Verständnis der Architektur von den Arduino Bibliotheken reichen leider auch nicht aus um eine eigene Implementation in vernünftiger Zeit programmieren zu können.

Dokumentation Architektur / Evaluierung:

Da der Kommunikationstest einiges länger als erwartet gedauert hat konnte ich meine Ziele für die Dokumentation nicht umsetzen. Vor allem waren und sind auch weiterhin einige Fragen bei der Architektur auf Mikrokontrollerseite offen, welche ich erst erklärt haben will.

Lernerfolg / Bemerkungen:

Nachdem ich stundenlang erfolglos nach Lösungen für den Kommunikationstest gesucht habe war ich dann doch ein wenig frustriert. Da ich dort nicht weiter kam habe ich kurzfristig einen Arbeitsschritt erledigt der nicht auf meinem Wochenprogramm stand. Den Zusammenbau und die Implementierung eines IO Test für den Mikrokontroller konnte ich ohne Probleme fertigstellen. Dies war nach den frustrierenden Stunden zuvor eine willkommene Abwechslung und ein kleines Erfolgserlebnis.

10.3.4 Woche 4

Arbeits- / Lernziele:

- Implementierung Verschlüsselung Kommunikationstest
- Dokumentation für Evaluierung einiger Technologien
- Architektur erarbeitet und dokumentiert
- Testfälle dokumentieren

nicht erreichte Ziele:

Dokumentation Evaluierung:

Diese Woche blieb leider keine Zeit mehr für die Dokumentation der Evaluierung. Neue Erkenntnisse ergaben sich während der Implementierung der Verschlüsselung bei der ich mir verschiedene Algorithmen und Übertragungsprotokolle angesehen ausgetestet und verglichen habe. Dies hat mehr Zeit als geplant benötigt. Ich weiss nun welche Technologien ich aus welchen Gründen einsetzen werde. Dies muss nun noch in der Dokumentation der Evaluierung niedergeschrieben werden.

Lernerfolg / Bemerkungen:

Nach der Besprechung mit meinem Betreuer beschäftigte ich mich nochmals mit der Implementation des CoAP Protokolls auf Seite des Mikrocontrollers. Ich habe nach einiger Zeit festgestellt, dass eine Implementierung viel zu kompliziert wird und zu viel Zeit in Anspruch nehmen würde. Deshalb findet die Kommunikation nun über das HTTP Protokoll statt. Außerdem konnte ich noch eine wichtige Hürde meistern und habe nun eine funktionierende Verschlüsselung zwischen Fernsteuergerät und Server. Die Implementierung bereitete mir einiges Kopfzerbrechen. Mitunter auch weil die Dokumentation und die Qualität einiger Bibliotheken für Kryptoalgorithmen zu wünschen übriglassen.

10.3.5 Woche 5

Arbeits- / Lernziele:

- Implementierung Authentifizierung Server - Frontend
- Implementierung bidirektionale Verbindung Server - Frontend
- Architektur Server – Frontend angepasst / verfeinert

nicht erreichte Ziele:

Architektur Server – Frontend angepasst / verfeinert:

Die Dokumentation und Visualisierung der verfeinerten Architektur fehlt noch. Ich konnte aber eine Lösung erarbeiten und habe nun eine sichere bidirektionale Verbindung so wie auch eine Authentifizierung implementieren können.

Lernerfolg / Bemerkungen:

Diese Woche konnte ich einige Probleme lösen und bin grundsätzlich gut vorwärts gekommen. Die ganze Architektur und die Kommunikation der verschiedenen Komponenten funktioniert grundsätzlich. Diese Herausforderung konnte ich meistern! Die Implementierung auf Seite des Mikrocontrollers bereitet mir aber immer wieder ein wenig Kopfzerbrechen. Immer wieder hat sich das Programm komisch verhalten und ich tat mich auch schwer mit dem Debugging in diesem Bereich. Ich habe auch mehr Zeit als geplant in die Implementierung beim Mikrocontroller gesteckt und weil es sich anbot angefangen die Funktionalität zu implementieren. Die Dokumentation habe ich deshalb leider vernachlässigt und muss mich nun wirklich mal intensiver damit beschäftigen. Ich kann nun auch aber durch meine Erfahrungen viel mehr dokumentieren und aus dem Vollen schöpfen was an und für sich gut ist. Nur bin ich durch die Umstände ein wenig aus dem Konzept gekommen.

10.3.6 Woche 6

Arbeits- / Lernziele:

- Dokumentation Architektur
- Dokumentation Implementation
- Dokumentation Evaluierung

nicht erreichte Ziele:

Dokumentationen:

Ein fieser Bug beim Empfang der Nutzdaten des Fernsteuergeräts hat mich viel Zeit Energie und Nerven gekostet. Ich habe deshalb die Analyse und das Beheben des Bugs priorisiert und die Dokumentation nach hinten geschoben.

Lernerfolg / Bemerkungen:

Leider konnte ich den Bug nicht beheben da er wohl in der Programmbibliothek des GSM Moduls zu finden ist. Viel Zeit ist für das analysieren und eingrenzen des Fehlers verloren gegangen. Schlussendlich habe ich mich entschieden durch einen kleinen Workaround das Problem zu umgehen. Natürlich ist dies nicht ideal, aber im Rahmen der Diplomarbeit muss ich mich wohl damit zufriedengeben und diesen Weg gehen. Der fiese Bug hat auch ein wenig an meiner Motivation gezeihrt und ich bin deshalb mit der Dokumentation nicht viel weitergekommen.

10.3.7 Woche 7

Arbeits- / Lernziele:

- Dokumentation Architektur
- Dokumentation Implementation
- Dokumentation Evaluierung

nicht erreichte Ziele:

Dokumentationen:

Die letzte Woche bin ich mit den Dokumenten nicht so weit wie geplant gekommen. Ich habe zwar sowieso mehr als eine Woche für das nachdokumentieren eingeplant doch wollte ich ein wenig mehr schaffen.

Lernerfolg / Bemerkungen:

Das mir das dokumentieren nicht so liegt wusste ich schon vorher. Ich habe vor allem teilweise Schwierigkeiten meine Gedanken fachsprachlich korrekt und verständlich nieder zu schreiben.

10.3.8 Woche 8

Arbeits- / Lernziele:

- Dokumentation Evaluierung
- Dokumentation Implementation

nicht erreichte Ziele:

Dokumentation Implementation:

Als ich mit der Dokumentation der Implementation für das Fernsteuergerät begonnen habe, war die Implementation an sich noch nicht komplett abgeschlossen. Deshalb bin ich dann beim dokumentieren angestanden und habe mich dann entschieden zuerst die Implementation komplett abzuschliessen.

Lernerfolg / Bemerkungen:

Ich glaube die Vorgehensweise, dass man zuerst die Implementierung eines Teilsystems abschliesst und erst danach diese dokumentiert ist sinnvoller als immer nur ein wenig zu implementieren und dies danach zu dokumentieren. Vor allem wenn sich dann bei der Implementierung zeigt, dass man gewisse Sachen bei anderen Funktionen nochmals ändern muss. Dies stimmt dann je nach dem nicht mehr mit der Dokumentation überein.

10.3.9 Woche 9

Arbeits- / Lernziele:

- Dokumentation Implementation
- Implementation Server

nicht erreichte Ziele:

Implementation Server:

Während der Implementierung auf Serverseite wurde es zunehmend mühsam, die Funktionalitäten zu testen. Deshalb habe ich angefangen das Frontend zu Implementierung damit ich direkt Daten eingeben kann und das System vom Frontend her testen kann. Dadurch konnte ich natürlich die Implementation auf Serverseite nicht komplett abschliessen.

Lernerfolg / Bemerkungen:

Gerade wenn es um Frontend zu Server Kommunikation geht wechselt man beim implementieren immer zwischen Frontend und Server hin und her. Das ist einerseits auch okay so, aber man läuft Gefahr, dass man sich verzettelt da man zu viele verschiedene Teile gleichzeitig programmiert. Ich stellte mir auch immer wieder Fragen bezüglich Best Practice. Gerade auch bei der Kommunikation der Sockets bezüglich Aufbau und Bezeichnung der Events und deren Daten.

10.3.10 Woche 10

Arbeits- / Lernziele:

- Implementierung komplett abschliessen
- Dokumentation Implementierung

nicht erreichte Ziele:

Dokumentation Implementierung:

Die Dokumentation für die Serverimplementation ist noch nicht fertig. Die Visualisierung hat dabei mehr Zeit als gedacht benötigt. Insbesondere auch, weil mein Anspruch an die Ästhetik der Visualisierungen hoch ist. Ich habe deshalb für das Einrichten von Vorlagen für die Visualisierung einiges an Zeit investiert

Lernerfolg / Bemerkungen:

Die Implementation ist komplett abgeschlossen und ich habe was ich wollte erreicht. Das System funktioniert stabil und es ist aus meiner Sicht schlussendlich ein brauchbarer und guter Prototyp geworden. Es fehlen noch einige Konfigurationsmöglichkeiten im Frontend sowie die Validation für die eingetragenen Werte. Doch aus meiner Sicht sind diese Funktionalitäten für einen Prototyp erstmal nicht relevant.

10.3.11 Woche 11

Arbeits- / Lernziele:

- Dokumentation Implementierung
- Dokumentation abschliessen
- Arbeit komplett abschliessen

nicht erreichte Ziele:

Keine

Lernerfolg / Bemerkungen:

In dieser letzten Woche stand noch einmal viel Dokumentationsarbeit an. Bei der Dokumentation der Implementierung kam ich auch gut vorwärts. Die Arbeit, welche ich in die Visualisierung für den Ablauf des Programms zwischen den einzelnen Komponenten gesteckt habe, hat sich meiner Meinung nach gelohnt und diente mir als gute Gedankenstütze beim niederschreiben des Ablaufs.

10.4 Besprechungsprotokolle

Besprechungsprotokoll 1

Datum: 17.03.2016 **Zeit:** 13:00 **Ort:** Bahnstrecke Ziegelbrücke/Zürich

Teilnehmer: Markus Kohlhaupt, Pascal Keller

Thema: Grundsätzliche Diskussion, Aufbau und Planung

Diskussion:

Meine definierten Meilensteine sind nicht terminbezogen und in dem Sinn nicht als solche zu bezeichnen.

- ⇒ Die Meilensteine besser als Risiken für das Projekt ansehen und Ausweichmöglichkeiten beziehungsweise alternative Lösungen dokumentieren.

Das Arbeitsjournal im Sinne einer Auflistung der geführten Arbeiten ist in Ordnung. Das Lernjournal sollte gesetzte Ziele wie auch die tatsächlich umgesetzten Ziele beinhalten. Das Arbeitsjournal sollte in den Anhang.

- ⇒ Arbeitsjournal anpassen und in den Anhang verschieben.

Die Kapitel für die Softwareumsetzung sind noch ein wenig schwammig und unpassend betitelt.

- ⇒ Wenn es soweit ist diese Kapitel anpassen. Stichworte: Software Design, gewählte und eingesetzte Technologien, Umsetzung der technischen Lösung (Implementation)

Ein Konzept für das gesamte System kann erst erläutert und umgesetzt werden, wenn vorher eine saubere Analyse / das Engineering erarbeitet wurde. Es müssen zuerst die Anforderungen an das System klar sein. Grundsätzliche Fragen sind: Welches Problem löst mein System? Welche Anwendungszwecke gibt es?

- ⇒ Zuerst sich wirklich intensiv mit dem Engineering auseinandersetzen und dieses abschliessen. Mit den Erkenntnissen und den Anforderungen die daraus entstehen kann man sich anschliessend konkret um die weiteren Schritte kümmern. Die Anforderungen durch möglichst generische Anwendungszwecke abbilden. Anwendungsszenarios beschreiben welche die Lösung abdecken kann.

Das Vorgehen ist nicht optimal und sollte so schnell und gut es geht verbessert werden.

- ⇒ Die gesetzten Ziele anpassen und erreichen. Bis Montag (21.3.16) schicke ich eine Mail an Herr Kohlhaupt mit meinen neuen generischen Anforderungen.

Unterschriften:

Markus Kohlhaupt
Diplomarbeitsbetreuer

Pascal Keller
Diplomand

Besprechungsprotokoll 2

Datum: 29.03.2016 **Zeit:** 16:00 **Ort:** Bahnstrecke Ziegelbrücke/Zürich

Teilnehmer: Markus Kohlhaupt, Pascal Keller

Thema: Architektur und Technologien

Diskussion:

Ich stecke bei der Implementation des CoAP Protokolls fest da es keine angepasste Programmbibliothek für das Arduino gibt welches mit dem GSM Shield funktioniert. Ich kann mich noch nicht entscheiden ob ich versuchen soll die Bibliothek für das GSM Shield anzupassen oder ob ich auf das schon funktionierende HTTP Protokoll zurückgreifen soll.

- ⇒ Ich setze mich nun intensiver mit dem Protokoll und deren Implementierung auseinander und nehme mir dafür einige Stunden Zeit.

Die Authentifizierung des Fernsteuergeräts bei dem Server scheint ein wenig zu aufwendig zu sein. Wir besprachen verschiedene Möglichkeiten für eine einfachere Umsetzung.

- ⇒ Ich finde eine Lösung die einfacher aber trotzdem sicher ist.

Ich habe nur sehr wenig Erfahrung im Bereich der Benutzeranmeldung für Webserver. Wir diskutierten die Anmeldung und Authentifizierung, sowie der Verbindungsaufbau zwischen Server und Frontend. Einige Lösungsansätze und Ideen kamen uns schnell in den Sinn.

- ⇒ Die Umsetzung kann schon noch einige Knacknüsse beinhalten. Die konkreten Lösungen zu Teilproblemen kommen wohl erst bei der Implementierung zum Vorschein. Wenn ich mich näher damit beschäftige werde ich sicher eine vernünftige Lösung erarbeiten können.

Die Beschreibungen der allgemein gehaltenen Anforderungen sind soweit in Ordnung.

- ⇒ Ich werde diese Anforderungen so lassen und auch keine neuen Anforderungen an das System stellen.

Unterschriften:

Markus Kohlhaupt

Diplomarbeitsbetreuer

Pascal Keller

Diplomand

Besprechungsprotokoll 3

Datum: 03.05.2016 **Zeit:** 14:00 **Ort:** Café Müller Näfels

Teilnehmer: Markus Kohlhaupt, Pascal Keller

Thema: Fortschritt, Lösungen, Probleme, Dokumentationsinhalt

Diskussion:

Fehleranalyse/Fehlerbehebung

- ⇒ Wenn noch Zeit übrig bleibt sollte ich den Fehler bei der Datenübertragung noch mal genauer analysieren.

Ich habe zu viele Begriffe im Glossar welche nicht beschrieben werden müssen.

- ⇒ Relevant sind Begriffe welche in dieser Diplomarbeit eine kurze Beschreibung nötig haben um Beispielsweise Verwirrungen zu vermeiden. Zum Beispiel was mit dem Endnutzer gemeint ist, und was das „Fernsteuergerät“ ist.

Bei der Dokumentation der Implementierung muss nicht jede Funktion beschrieben sein.

- ⇒ Ich beschreibe die wichtigen und interessanten Funktionen des Programmcodes.

Eine Bedienungsanleitung für das System ist ein Punkt der zur Dokumentation gehört.

- ⇒ Funktionen des Systems welche eine genauere Beschreibung nötig haben und nicht selbsterklärend sind werde ich in einer Anleitung dokumentieren.

Wie die Hardware schlussendlich aussieht ist nicht so relevant.

- ⇒ Der schlussendliche Prototyp sollte zwar nicht gerade auseinanderfallen, wenn man ihn berührt, aber es reicht wenn man damit die Funktionsweise gut demonstrieren kann.

Die zu unterschreibenden Dokumente werden nicht unterschrieben abgegeben.

- ⇒ Vor der Abgabe werde ich Herr Kohlhaupt die Dokumentation digital schicken. Falls dann was nicht in Ordnung mit den zu unterschreibenden Dokumenten ist, kriege ich Feedback. Die Dokumente werden schlussendlich nach der Abgabe in der Originaldokumentation unterschrieben.

Unterschriften:

Markus Kohlhaupt
Diplomarbeitsbetreuer

Pascal Keller
Diplomand

10.5 Disposition und Auftrag

Disposition und Auftrag für die

Diplomarbeit
von **Pascal Keller**

der **Technikerschule HF Zürich**



Versionskontrolle:

Erstellt am:	11.1.2016
Geprüft von:	Markus Kohlhaupt Studiengangleiter Informatik
Geprüft am:	14.1.2016
Nachbesserung:	
Freigegeben von:	Markus Kohlhaupt Studiengangleiter Informatik
Freigegeben am:	15.2.2016

Inhaltsverzeichnis:

1 Aufgabenstellung der Diplomarbeit	3
2 Disposition zur Diplomarbeit.....	4
2.1 Ausgangslage und Projektbeschreibung	4
2.2 Relevanz	4
2.3 Fragestellung und Zielsetzung	4
2.4 Abgrenzung.....	5
2.5 Risikoabschätzung.....	5
2.6 Hilfsmittel.....	5
3 Fremdbeteiligung	5

1 Aufgabenstellung der Diplomarbeit

Auftrag: Umsetzung der Diplomarbeit gemäss der bewilligten Disposition..

- Umfang:**
- Analyse der Aufgabenstellung
 - Definition des Sollzustandes
 - Bewertung allfälliger Lösungsvarianten
 - Entwurf der ausgewählten Variante
 - Realisierung der ausgewählten Variante
 - Dokumentation und Test
 - Der Umfang der Arbeit wird laufend mit dem Betreuer abgestimmt

Betreuer: Markus Kohlhaupt

Email: markus@kohlhaupt.ch

Telefon: +41 79 813 98 63

Diplomandin / Diplomand: Pascal Keller

Email: pascalkeller@gmail.com

Telefon: +41 79 476 55 20

Ausgabe der Arbeit: Mittwoch, 17. Februar 2016, 18:00 Uhr

Abgabe der Arbeit: **Dienstag, 31. Mai 2016, 18.00 Uhr**

Sekretariat Technikerschule HF Zürich (Zimmer ZL 1.12) oder Poststempel

Mitgelieferte Dokumente: Folgende Dokumente sind integraler Bestandteil von Diplomarbeiten an der Technikerschule HF Zürich

- Ausführungsbestimmungen für Diplomarbeiten vom August 2014
- Prozesse und Richtlinien für Qualifikationsverfahren vom September 2015

Unterschriften

Pascal Keller
Diplomand

Markus Kohlhaupt
Diplomarbeitsbetreuer

Markus Kohlhaupt
Studiengangleiter

2 Disposition zur Diplomarbeit

Titel der Arbeit:	Open Remote (Arbeitstitel)
Vorname, Nachname:	Pascal Keller
Telefon:	+41 79 476 55 20
Email:	pascalkeller@gmail.com
Studiengang:	Informatik
Semester:	6
Bevorzugter Betreuer:	Markus Kohlhaupt

Beschreiben Sie Ihren Vorschlag nach den folgenden Gesichtspunkten.

2.1 Ausgangslage und Projektbeschreibung

Welches Phänomen liegt dem Thema zugrunde? Was fällt Ihnen auf? Welche Bedürfnisse erkennen Sie? Was genau ist die jetzige Ist-Situation bzw. die Ausgangslage, die Sie mit Ihrer Diplomarbeit verbessern bzw. für die Sie im Rahmen Ihrer Diplomarbeit eine Lösung erarbeiten wollen? In welchem Umfeld ist das Thema angesiedelt?

Verschiedene Hersteller bieten Lösungen an um Heizungen oder andere elektrische Geräte im Ferienhaus oder an anderen entfernten Standorten fernzusteuern. Bei einigen der Lösungen wird das Mobilfunknetz benutzt um Steuer-Signale per SMS an die Steuergeräte zu senden. Andere funktionieren über das Festnetztelefon oder über das Internet. Allesamt sind es kommerzielle proprietäre Lösungen. Die Produkte unterscheiden sich einerseits im Umfang der Funktionen und vor allem sind einige nicht sehr bedienerfreundlich.

Ich möchte mit meiner Diplomarbeit ein System entwickeln welches auf Open Source Hard- sowie Software basiert. Die Lösung soll modular aufgebaut und einfach bedienbar sein.

Relevante Themenfelder dieser Diplomarbeit sind Mikroprozessortechnik, Datenkommunikation, Internet und Web Technologien sowie Elektrotechnik und Elektronik.

2.2 Relevanz

Weshalb ist das Thema für die Allgemeinheit oder bestimmte Wirtschaftszweige oder für einen bestimmten Personenkreis relevant? Ist das Thema finanziell relevant? Welche Entwicklungen hängen vom Thema ab?

Eine modular aufgebaute nicht proprietäre offene Lösung ist für technikbegeisterte Hobbyisten interessant welche eine eigene anpassbare Lösung für ihre Zwecke realisieren möchten.

2.3 Fragestellung und Zielsetzung

Welche Frage soll beantwortet werden? Formulieren Sie eine konkrete Frage (Formulierungen wie »...will mich beschäftigen mit...« oder »... will meine Kenntnisse vertiefen in ...« sind nicht ausreichend). Welche technische Herausforderung soll gelöst werden? Was soll als Ergebnis herauskommen (Verbesserungen, Veränderungen, Gewinn)? Welches ist der konkrete Nutzen aus Ihrer Diplomarbeit? Wie wertvoll ist Ihre thematische Erarbeitung? Welches sind die messbaren Resultate? Mit welchen Kenngrößen wollen Sie den Erfolg messen? Welche Erwartungen haben Sie an das Resultat? Welches sind Mussziele, die erreicht werden müssen und welches sind und Kannziele, die möglicherweise erreicht werden können?

Wie können die Anforderung an ein offenes, modulares, zuverlässiges und einfach zu bedienendes Fernsteuerungssystem optimal erfüllt werden?

Mussziele:

- funktionierender Prototyp mit Datenkommunikation über das Mobilfunknetz
- Bedienung mit Webbrower auf dem Smartphone möglich
- Einlesen von einem analogen Signal möglich
- Schalten von mindestens einem elektrischen Gerät möglich

Kannziele:

- Gut designtes Web Interface
- Datenkommunikation mittels Ethernet Schnittstelle und Wi-Fi
- Einlesen und schalten von mehreren Signalen und Geräten

2.4 Abgrenzung

Welche einzelnen Themen (Schwerpunkte) müssen für die Gesamtlösung bearbeitet werden? Wie soll die Lösung pro Schwerpunkt aussehen und was bzw. wie trägt sie zur Gesamtlösung bei? Welche Schwerpunkte werden in der Diplomarbeit ausgeklammert und weshalb?

Wie die Datenkommunikation im Sinne von geografischem Standort oder anderen Einflüssen sicher gestellt wird soll nicht Thema dieser Diplomarbeit sein. Auch wie das Gerät von elektrischer Seite her versorgt wird, (Akkubetrieb) und wie es verbunden ist, (Störeinflüsse, Blitzschlag) wird ausgeklammert.

2.5 Risikoabschätzung

Welche Risiken birgt die Umsetzung? Welche Vorbereitungen werden getroffen? Welche Auswirkungen bewirken unerwartete Änderungen?

Für diese Diplomarbeit bestehen keine besonderen Risiken. Die Verfügbarkeit von Material und Technologien ist gewährleistet. Gesetzliche Auflagen sind bei dieser Arbeit nicht relevant. Es verbleiben die üblichen Risiken wie Verzögerungen durch Krankheit oder technische Herausforderungen. Durch eine entsprechende Planung können diese jedoch minimiert werden.

2.6 Hilfsmittel

Welche Hilfsmittel planen Sie einzusetzen? In welcher Umgebung soll die Arbeit stattfinden?

Die Werkstatt der oswald electric ag darf und werde ich benutzen um den Hardware Teil meiner Diplomarbeit umsetzen zu können. Weil die Datenkommunikation unter anderem auch über das Mobilfunknetz abgewickelt wird bin ich auf einen oder mehrere Mobilfunknetz-Anbieter angewiesen. Diese stellten mir die Sim-Karten sowie das Netz zur Verfügung. Ansonsten bin ich nur noch auf mein MacBook mit funktionierendem Internetzugang angewiesen.

3 Fremdbeteiligung

Falls die Arbeit für eine Firma erstellt wird, bitte Name der Firma, zuständige Person und Abteilung, Emailadresse, Telefonnummer und die vollständige Postadresse angeben:

Den Auftrag, welchen ich im Rahmen der Diplomarbeit erledige, habe ich mir selbst erteilt. Es sind keine Firmen oder andere kommerzielle Institutionen an dieser Arbeit beteiligt.

Falls von Seiten der Firma, für welche die Diplomarbeit erstellt wird, Geheimhaltungserklärungen verlangt werden, diese bitte mit der Disposition vorlegen.

Keine Geheimhaltung

10.6 Ausführungsbestimmungen

Ausführungsbestimmungen für Diplomarbeiten

der **Technikerschule HF Zürich**



Inhaltsverzeichnis:

1	Zweck dieser Bestimmungen	3
2	Allgemeine Bestimmungen	3
3	Diplomarbeitsthema.....	3
4	Ausgabe der Aufgabenstellung der Diplomarbeit	4
5	Abgabe der Diplomarbeit	4
5.1	Abzugebende Dokumente und Datenträger	4
5.1.1	Protokoll der Besprechungen mit dem Diplomarbeitsbetreuer	4
5.1.2	Zusammenfassung	4
5.1.3	Lebenslauf.....	4
6	Ausführung der Arbeit.....	4
6.1	Allgemeines.....	4
6.2	Schriftliche Arbeit.....	5
6.2.1	Ergänzungen Studiengang Elektrotechnik (Vertiefungsrichtung Elektronik)	6
6.2.2	Ergänzungen Studiengang Informatik.....	6
6.2.3	Ergänzungen Studiengang Maschinenbau	6
6.3	Lernjournal- und Arbeitsjournal	6
6.4	Sicherheitskopie.....	7
7	Betreuung und Überwachung	7
8	Bewertung der Diplomarbeit.....	7
9	Taxation der Arbeit	7
9.1	Vortaxation	7
9.2	Taxation.....	7
10	Inkrafttreten	8

1 Zweck dieser Bestimmungen

Die vorliegenden Bestimmungen haben den Zweck, die Ausgabe, die Ausarbeitung und die Abgabe sowie die Betreuung der Diplomarbeit in Ergänzung der Prüfungsbestimmungen des Reglements der Technikerschule HF Zürich im Detail zu regeln.

2 Allgemeine Bestimmungen

Für die Ausführung der Diplomarbeit gelten die folgenden allgemeinen Bestimmungen:

- Die Arbeit kann entweder als Einzel- oder als Gruppenarbeit ausgeführt werden.
- Jede Diplomarbeit wird von einem vom Studiengangleiter zugewiesenen Betreuer begleitet. Seine Aufgaben sind unter »7 Betreuung und Überwachung« erläutert.
- Die Ausführung der Diplomarbeit darf weder von Behörden noch von anderen Auftraggebern in irgendeiner Form entschädigt werden.
- Die Arbeit muss von den Studierenden in allen Belangen selbständig ausgeführt werden. Eine diesbezügliche Erklärung ist im Anhang der vorliegenden Bestimmungen zu finden. Diese muss von der / vom Studierenden unterschreiben und mit der Arbeit abgegeben werden.
- Die Diplomarbeit ist grundsätzlich Eigentum des Diplomanden/der Diplomandin und darf von Drittpersonen nicht weiter verwendet werden. Die Technikerschule HF Zürich behält sich jedoch unter schriftlicher Ankündigung das uneingeschränkte und kostenlose Recht vor, die Arbeit oder Teile davon zu schulischen oder Werbezwecken zu nutzen.
- Erfolgt die Aufgabenstellung durch Dritte und haben diese massgebliche Unterstützung geleistet, so haben sie während der Erstellung und nach der Diplomierung Recht auf Einsicht.
- Erfolgt nach der Fertigstellung der Diplomarbeit eine kommerzielle Auswertung der Arbeit oder von Teilen der Arbeit, werden diesbezügliche Vereinbarungen grundsätzlich zwischen dem Aufgabensteller und dem Verfasser der Arbeit geregelt und abgeschlossen.

3 Diplomarbeitsthema

Grundsätzlich wird das Thema der Diplomarbeit von den Studierenden gewählt. Bei der Wahl des Themas sind folgende Regeln zu beachten:

- Für den Themenfindungsprozess ist der Studiengangleiter zuständig. Themenvorschläge können bis Ende der 14. Woche des 5. Semesters in schriftlicher Form an den Studiengangleiter eingereicht werden. Die Themenfreigabe erfolgt durch den Studiengangleiter.
- Die Diplomarbeit kann entweder in rein theoretischer Form oder kombiniert in theoretischer und praktischer Form ausgeführt werden. In beiden Fällen soll sich der Aufwand im Rahmen von ca. 250 Arbeitsstunden bewegen.
- Es muss grundsätzlich ein neues, im bisherigen Fachunterricht in Form von Semesterarbeiten nicht behandeltes Thema gewählt werden.
- Das gewählte Thema muss den wesentlichen Teilen des Fachunterrichts des 4. und des 5. Semesters entsprechen. Als richtungweisend für die Themenwahl gelten folgende Bereiche:

Elektrotechnik (Elektronik)	Elektrotechnik (Energietechnik)	Informatik	Maschinenbau
<ul style="list-style-type: none"> ▪ Mikroprozessortechnik ▪ Informations- und Kommunikationstechnik ▪ Digitaltechnik ▪ Analogtechnik 	<ul style="list-style-type: none"> ▪ Elektrische Maschinen ▪ Leistungselektronik ▪ Elektrische Anlagen / Elektroplanung 	<ul style="list-style-type: none"> ▪ Datenbanken ▪ Kommunikation und Netzwerke ▪ Informatiksicherheit ▪ Softwareentwicklung ▪ Systemadministration 	<ul style="list-style-type: none"> ▪ Konstruktions- und Gestaltungslehre

- Wo die Beschaffung allfälliger Unterlagen zu bestimmten Vereinbarungen führen könnte, ist es ausschliesslich Sache des Betreuers, die erforderlichen Verhandlungen zu führen.

4 Ausgabe der Aufgabenstellung der Diplomarbeit

Die Ausgabe der Arbeiten erfolgt in der ersten Unterrichtswoche des 6. Semester gemäss Terminplan durch die Schulleitung. Die Aufgabenstellung erfolgt in schriftlicher Form und muss von den Studierenden persönlich entgegengenommen und in 4-facher Ausführung unterzeichnet werden. Über Ausnahmen entscheidet grundsätzlich die Schulleitung.

5 Abgabe der Diplomarbeit

Die Abgabe der Diplomarbeit muss an dem im Stundenplan unter »Abgabe der Diplomarbeiten« aufgeführten Datum erfolgen. Als Ort der Erfüllung gilt das Sekretariat der Technikerschule HF Zürich (Lagerstrasse 45). Die Diplomierenden erhalten bei der Abgabe ihrer Arbeit vom Sekretariat der Technikerschule HF Zürich eine schriftliche Quittung. Die Arbeit kann entweder persönlich abgegeben werden oder auf postalischem Wege. In diesem Falle gilt das Datum des Poststempels. Zu spät abgegeben Arbeiten werden nicht angenommen und sind vollumfänglich zu wiederholen.

Wird eine Diplomarbeit nur unvollständig abgegeben, so entscheidet die Schulleitung gemäss Reglement und nach Rücksprache mit dem zuständigen Betreuer über die Zulassung zu den Diplomprüfungen.

5.1 Abzugebende Dokumente und Datenträger

Folgenden Dokumente und Datenträger müssen am Abgabetermin im Sekretariat der Technikerschule HF Zürich oder auf einer Poststelle abgegeben werden:

- Der schriftliche Bericht der Arbeit als Gesamtwerk in Papierform in zweifacher Ausführung
- Zusätzlich sämtliche Dokumente auf einem handelsüblichen Datenträger (CD, DVD, Memory Stick) ebenfalls in zweifacher Ausführung
- Der Datenträger soll in geeigneter Form dem Bericht beigefügt werden
- Für die Lesbarkeit der Datenträger ist der Diplomand/die Diplomandin zuständig
- Nicht lesbare Datenträger gelten als nicht abgegeben
- Vom Bericht getrennt die Zusammenfassung gemäß Punkt 6.5
- Ebenfalls vom Bericht getrennt der Lebenslauf gemäß Punkt 6.6

Zudem sind abzugeben:

5.1.1 Protokoll der Besprechungen mit dem Diplomarbeitsbetreuer

Jede Besprechung mit dem Diplomarbeitsbetreuer muss vom Diplomanden / von der Diplomandin kurz protokolliert werden (Ort, Datum, Zeit, Grund der Besprechung, gefasste Beschlüsse). Diese Protokolle sind vom Diplomarbeitsbetreuer jeweils zu unterzeichnen und gelten als integraler Bestandteil der schriftlichen Arbeit (Siehe unter 6.2 Schriftliche Arbeit)

5.1.2 Zusammenfassung

Zusätzlich zum schriftlichen Bericht ist eine Zusammenfassung in Form eines Management Summary auf einer, maximal auf zwei DIN A4 Seiten zu erstellen. Dabei sollen die wesentlichen Punkte der Aufgabenstellung, des Konzeptes und der Lösung beschrieben, sowie ein kurzer Überblick bezüglich Schwierigkeiten und Lernerfolge gegeben werden.

5.1.3 Lebenslauf

Zur Diplomarbeit gehört ebenfalls ein Lebenslauf in Kurzform. Das zu verwendete Formular ist auf dem Sekretariat der Technikerschule HF Zürich zu beziehen.

6 Ausführung der Arbeit

6.1 Allgemeines

Die Diplomarbeit ist im 6. Semester des Studiums während der im Stundenplan bekannt gegebenen Zeit parallel zum Unterricht auszuführen. Es gelten dazu die folgenden Zielsetzungen:

- Anwenden einer effizienten Arbeitssystematik
- Praxisnahe Arbeiten unter erhöhtem Zeitdruck
- Wertvolle Selbsteinschätzung der erlangten Fähigkeiten als Grundlage für die Neubestimmung der beruflichen Laufbahn

Die im Stundenplan des 6. Semesters reservierte Zeit ist hauptsächlich für die Besprechungen mit dem Betreuer vorgesehen, wobei Zeit und Ort der Treffen in Absprache mit dem Betreuer frei gewählt werden können.

Alle benutzten Quellen (Internet, Literatur, Drittpersonen), auch jene, die der Inspiration oder als Vorlage dienten, sind im Anhang der Diplomarbeit lückenlos aufzuführen. Zudem ist die Verwendung der Quellen in der Arbeit (Bericht, Code) eindeutig zu kennzeichnen. Fehlen diese Angaben oder sind sie unvollständig, so kann dies unter Umständen zu einer Nichtbewertung der Diplomarbeit führen. Insbesondere dann, wenn die Eigenleistung nicht klar feststellbar ist.

6.2 Schriftliche Arbeit

Ein wesentlicher Bestandteil der Diplomarbeit ist der schriftliche Bericht. Darin werden sämtliche Teile der Arbeit ausführlich und vollständig dokumentiert. Insbesondere sind dies:

- Die Analyse des Ist-Zustandes und die Zielsetzungen (Soll-Zustand) sind knapp und verständlich zu beschreiben.
- Vorgehen, Abläufe und Berechnungen sind in logischer Folge knapp, deutlich und nachvollziehbar darzustellen.
- Konzeptansätze, erarbeitete Lösungsvarianten und eingesetzte Werkzeuge sind zu begründen sowie ausführlich und verständlich zu dokumentieren.
- Sind mehrere Varianten zur Lösung erarbeitet worden, so sind diese mit Hilfe geeigneter und nachvollziehbarer Kriterien zu bewerten.
- Die Wahl der endgültigen Lösungsvariante ist anhand der Auswahlkriterien zu begründen.
- Wo nötig sind die Beschreibungen mit geeigneten Illustrationen wie Zeichnungen, Diagrammen, Schemata, etc. zu ergänzen.
- Der Bericht muss ein persönliches Fazit der Arbeit beinhalten (welches waren die Lerneffekte, wo lagen die Schwierigkeiten, was könnte man anders oder besser machen, wie könnte die Arbeit erweitert werden, etc.).
- Eine Planung der Arbeiten (Projektplan) muss ersichtlich sein.

Die Form des schriftlichen Berichtes muss den nachfolgend aufgeführten Punkten Rechnung tragen:

- Der Schriftliche Bericht muss im Format DIN A4 erstellt werden.
- Der schriftliche Bericht ist klar zu gliedern (Inhaltsverzeichnis, Abbildungsverzeichnis, Anhänge) und wo notwendig mit aussagekräftigen Illustrationen zu ergänzen.
- Sämtliche erstellten Dokumente müssen die Bezeichnung der Schule tragen. Entsprechende Schriftzüge und Logos sind beim Sekretariat der Technikerschule HF Zürich zu beziehen.
- Die einzelnen Blätter sind zu nummerieren und in geeigneter Form zu binden (Ordner, Schnellhefter, Plastikrücken, Leimrücken, etc.).

Zum schriftlichen Bericht gehören zudem folgende Dokumente, welche in geeigneter Form (eingebunden oder elektronisch integriert) dem Bericht hinzugefügt werden müssen:

- Die Aufgabenstellung im Original
- Die Bestimmungen zur Ausführung von Diplomarbeiten mit ausgefüllter und unterzeichneter Erklärung hinsichtlich der selbständigen Ausführung der Arbeit
- Kurzprotokolle der Besprechungen mit dem Diplomarbeitsbetreuer

6.2.1 Ergänzungen Studiengang Elektrotechnik (Vertiefungsrichtung Elektronik)

Für den Studiengang Elektrotechnik (Elektronik) gelten bezüglich der schriftlichen Arbeit zusätzlich die folgenden Punkte:

- Schemata, Stücklisten, Schaltungslayouts und Datenblätter von verwendeten elektronischen Bauelementen sind auf dem Datenträger so zu speichern, dass sie einerseits lesbar sind und andererseits den urheberrechtlichen Bestimmungen genügen.
- Zusätzlich verwendete Werkzeuge und deren Dokumentation sind nach Möglichkeit und ohne Urheberrechtsverletzungen ebenfalls auf dem Datenträger abzugeben.
- Das Format und der Inhalt des Datenträgers sind zu beschreiben. Der Inhalt muss mit den handelsüblichen Werkzeugen lesbar sein. Allenfalls müssen die Daten "lesbar" exportiert auf dem Datenträger vorliegen.

6.2.2 Ergänzungen Studiengang Informatik

Für den Studiengang Informatik gelten bezüglich der schriftlichen Arbeit zusätzlich die folgenden Punkte:

- Für Erläuterungen kann der Quellcode auszugsweise in den Bericht übernommen werden.
- Zusätzlich eingesetzte Werkzeuge und deren Dokumentation sind nach Möglichkeit und ohne Urheberrechtsverletzungen ebenfalls auf dem Datenträger abzugeben.
- Das Format und der Inhalt des Datenträgers sind zu beschreiben. Der Inhalt muss mit den Standardwerkzeugen lesbar sein. Allenfalls müssen die Daten "lesbar" exportiert auf dem Datenträger vorliegen.
- Quellcode und Konfigurationsdateien müssen in Form von editierbarem Text gespeichert werden.
- Ausführbare Programme und deren Quellcode sind auf dem Datenträger abzugeben. Sie müssen eindeutig, unmissverständlich und mit entsprechender Quellenangaben ausgezeichnet sein und lückenlos die Rückschlüsse: »selbst erzeugt, durch Tool erzeugt, ohne Änderung übernommen, übernommen aber abgeändert« ermöglichen.

6.2.3 Ergänzungen Studiengang Maschinenbau

Für den Studiengang Maschinenbau gelten bezüglich der schriftlichen Arbeit zusätzlich die folgenden Punkte:

- Die Aufgabenstellung ist in einer Übersicht (Disposition, Zusammenstellung) darzustellen.
- Mindestens ein wichtiges Detail – vom Betreuer bestimmt – soll gemäss Aufgabenstellung detaillierter ausgearbeitet werden.
- Der Zeichnungs- und Berechnungsaufwand wird vom Betreuer bestimmt.
- Sämtliche Zeichnungen sind normgerecht und fertigungsgerecht zu erstellen und müssen rechts unten mit einem DIN-Zeichnungskopf »Technikerschule HF Zürich« versehen werden.
- Kopien der Originalzeichnungen sind gefaltet, jedoch nicht eingehefbt abzugeben.
- Dispositionen und Maschinenzusammenstellungen müssen die wesentlichen Hauptmasse enthalten.
- Erzeugnisse aus zusätzlich eingesetzten Werkzeugen wie beispielsweise CAD müssen auf dem Datenträger in lesbarer Form gespeichert werden.

6.3 Lernjournal- und Arbeitsjournal

Als integraler Bestandteil des schriftlichen Berichts erstellt jede / jeder Studierende ein Lern- und Arbeitsjournal. Im Wesentlichen enthält dies:

- Lernziele zur Diplomarbeit (werden wöchentlich von der Diplanddin oder dem Diplanden formuliert und mit dem Betreuer oder der Betreuerin besprochen)
- Lernzielkontrolle (wie sind die Lernziele erreicht worden, welche Lernerfolge sind gemacht worden)
- Die wichtigsten Arbeitsschritte sollen in einem Arbeitsjournal festgehalten werden. Dies muss auch dem Betreuer oder der Betreuerin auch zu den Zwischenbesprechungen vorgelegt werden.

6.4 Sicherheitskopie

Diplanden sind verpflichtet, jederzeit Sicherheitskopien ihrer Arbeit zu erstellen und diese auf Verlangen der Schulleitung der Technikerschule HF Zürich zur Verfügung zu stellen.

7 Betreuung und Überwachung

Jede Diplomarbeit wird von einem vom Studiengangleiter bestimmten Betreuer begleitet. Der Betreuer ist verpflichtet, den Fortschritt der Arbeit anhand der vom Diplanden / von der Diplanddin erstellten Planung und des Lern- und Arbeitsjournals zu überwachen und regelmässig zu überprüfen. Er unterzeichnet die Kurzprotokolle der Besprechungen. Im Rahmen dieser Besprechung sind gleichzeitig Unterlagen für die endgültige Bewertung zu sammeln. Besprechungstermine werden zwischen Betreuer und Dipland / Diplanddin laufend vereinbart.

Hat der Betreuer auf Grund der gelieferten Zwischenresultate Grund zur Annahme, dass eine Fertigstellung der Arbeit auf den festgesetzten Termin mangels Einsatz des Diplanden / der Diplanddin fragwürdig erscheint, so erfolgt eine schriftliche Ermahnung seitens des Betreuers. Tritt trotz dieser Ermahnung keine deutliche Verbesserung der Leistungen ein, so kann der Betreuer nach Absprache mit dem zuständigen Studiengangleiter sein Mandat niederlegen, wodurch die Diplomarbeit automatisch als nicht abgegeben gilt.

8 Bewertung der Diplomarbeit

Nach Abgabe der Diplomarbeit wird diese vom zuständigen Betreuer eingesehen und anhand der Vorlage »Bewertung von Diplomarbeiten« bewertet. Der Bewertungsvorschlag ist mindestens eine Woche vor der Taxation der Arbeiten, dem dafür zuständigen Studiengangleiter zuzustellen. Die endgültige Note der Diplomarbeit wird im Anschluss an die Taxation festgelegt.

Die für die Bewertung massgeblichen Kriterien sind dem mitgelgenden Dokument »Bewertungsvorlage für Diplomarbeiten vom Februar 2016 zu entnehmen.

9 Taxation der Arbeit

9.1 Vortaxation

Als Hauptprobe zur Taxation organisiert die Schulleitung eine Vortaxation. Diese ist obligatorisch und findet 1 – 2 Wochen nach Abgabe der Diplomarbeit statt. Studierende zeigen einen Auszug von 10 Minuten Ihrer Präsentation der Diplomarbeit. Mitglieder der Schulleitung geben Ihnen Tipps und Anregungen für die definitive Präsentation der Arbeit anlässlich der Taxation.

9.2 Taxation

Zum Ende des Diplomsemesters findet eine Taxation der Diplomarbeit statt. Die jeweiligen genauen Daten sind den von der Schulleitung publizierten Prüfungsplänen zu entnehmen. Die Taxation unterliegt folgenden Regeln und Bestimmungen:

- Die Taxation wird vom jeweiligen Studiengangleitern in Anwesenheit von Prüfungsexperten und des zuständigen Betreuers durchgeführt.
- Die Taxation besteht zum einen in der Präsentation der Arbeit und einer Vorführung von allfälligen praktischen Teilen der Arbeit durch den Diplanden / die Diplanddin.
- Bei Gruppenarbeiten ist die Vorstellung der Diplomarbeit unter den Gruppenmitgliedern zu gleichen Teilen aufzuteilen.
- Für die Präsentation stehen ein Hellraumprojektor und ein Beamer zur Verfügung.
- Werden zusätzliche Hilfsmittel benötigt, so müssen diese durch die Diplanden vorgängig selbst beschafft werden.
- Im Anschluss an die Vorstellung der Arbeit, haben Studiengangleiter und Experten die Gelegenheit, Fragen zur Diplomarbeit zu stellen.
- Für die Präsentation und die Vorführung der Arbeit sowie für die Beantwortung der Fragen stehen jedem Dipland / jeder Diplanddin 20 bis 30 Minuten zur Verfügung. Die genaue Zeit ist den von der Schulleitung publizierten Prüfungsplänen zu entnehmen.

- Diplomanden sollen auf Fragen der Prüfungsorgane knapp und kompetent antworten können. Bei Gruppenarbeiten muss jeder über die gesamte Diplomarbeit Auskunft geben können.
- Den Diplomanden wird vor der Taxation genügend Zeit eingeräumt, ihre Arbeiten zu installieren und auf Funktionsfähigkeit hin zu überprüfen.

10 Inkrafttreten

Die vorliegenden Bestimmungen zur Ausführung von Diplomarbeiten ersetzen sämtliche früheren Bestimmungen vollumfänglich und treten ab dem 1. Februar 2016 in Kraft.

Zürich, 31. Januar 2016



Peter Jost, Schulleiter Technikerschule HF Zürich

Anhang 1: Bestätigung und ergänzende Bemerkungen

Bestätigung

Hiermit bestätigt der / die Unterzeichnete, seine / ihre Diplomarbeit zum Thema

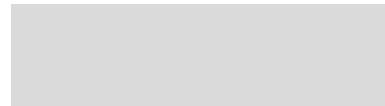
OMRes - Open Modular Remote System (Arbeitstitel Open Remote)

mit Ausnahme der zitierten Quellen vollumfänglich selbstständig und gemäss der geltenden Reglemente und Bestimmungen ausgeführt zu haben.

Ort und Datum:

Mollis, 31.5.2016

Unterschrift:



An der Aufgabenstellung beteiligte Unternehmen:

Firma:

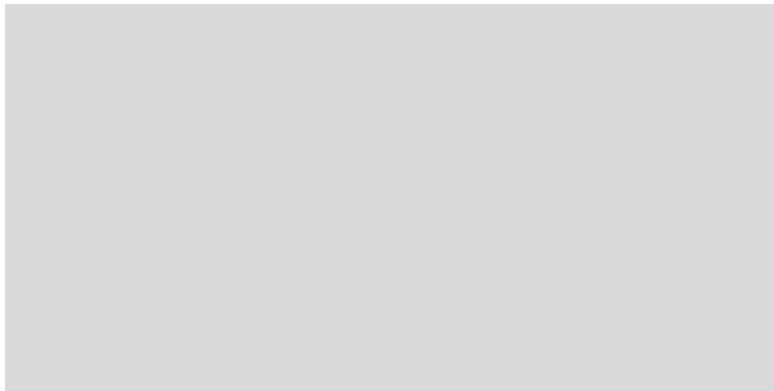
Adresse:

Kontaktperson:

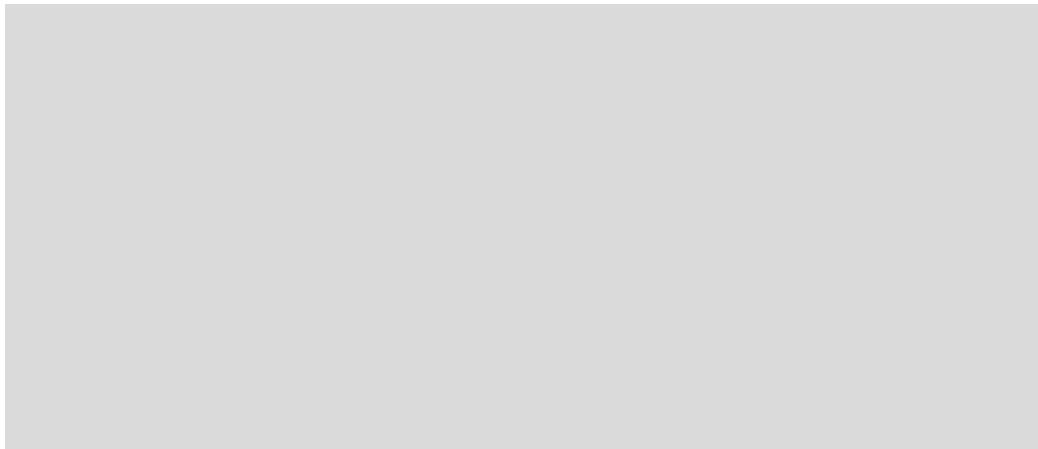
Telefonnummer:

Ort und Datum:

Unterschrift:



Ergänzende Bemerkungen des Betreuers:



Glossar

- **Endnutzer:** Ein Nutzer des Systems der sich ausschliesslich über die Weboberfläche einloggt und die Ausgänge schaltet und die übermittelten Werte überwacht.
- **Fernsteuergerät:** Das Gerät, welches am entfernten Standort installiert ist.
- **Kanal:** Ein Kanal ist eine zusammengehörende Einheit eines analogen Eingangs und einem entsprechenden digitalen Ausgang im Fernsteuergerät. Ein- und Ausgänge können immer nur Kanalweise gesteuert werden.
- **Frontend:** Die Komponente des Systems, welche lokal beim Endnutzer im Webbrowser läuft. Dazu gehört sowohl die JavaScript Applikation, als auch die grafische Oberfläche.

Quellenverzeichnis

1. **socket.io.** Rooms and Namespaces. [Online] <http://socket.io/docs/rooms-and-namespaces/>.
2. **wikipedia.org.** Man-in-the-Middle-Angriff. [Online] <https://de.wikipedia.org/wiki/Man-in-the-Middle-Angriff>.
3. **letsencrypt.org.** Getting Started. [Online] <https://letsencrypt.org/getting-started/>.
4. **Raffi.** Grundlagen: Sicheres Passwort Hashing mit Salts. [Online] <http://codebude.net/2015/03/30/grundlagen-sicheres-passwort-hashing-mit-salts/>.
5. **arduino.cc.** ArduinoBoardUno. [Online] <https://www.arduino.cc/en/Main/ArduinoBoardUno>.
6. —. ArduinoBoard101. [Online] <https://www.arduino.cc/en/Main/ArduinoBoard101>.
7. **arduino.org.** Arduino GSM Shield 2. [Online] <http://www.arduino.org/products/shields/5-arduino-shields/arduino-gsm-shield-2>.
8. **cooking-hacks.com.** 3G/GPRS shield for Arduino. [Online] <https://www.cooking-hacks.com/3g-gprs-shield-for-arduino-3g-gps>.
9. **wikipedia.org.** Raspberry Pi. [Online] https://de.wikipedia.org/wiki/Raspberry_Pi.
10. **pi-supply.com.** picture. [Online] https://www.pi-supply.com/wp-content/uploads/2016/02/pi_angled_noobs-new.jpg.
11. **tinkerforge.com.** Was ist Tinkerforge? [Online] http://www.tinkerforge.com/de/home/what_is_tinkerforge/.
12. —. picture. [Online] <http://www.tinkerforge.com/de/home/press/>.
13. **netzwoche.ch.** Swisscom beendet die 2G-Ära. [Online] <http://www.netzwoche.ch/de-CH/News/2015/10/08/Swisscom-macht-das-Funknetz-fit-fuer-die-Zukunft.aspx>.
14. **play-zone.ch.** Original Arduino Uno R3 (Atmega328) *RETAIL*. [Online] <http://www.play-zone.ch/de/elektronik-kit-zubehoer/avr-arduino-freeduino/boards-original/original-arduino-uno-r3-atmega328.html>.
15. **mouser.ch.** Genuino 101. [Online] <http://www.mouser.ch/ProductDetail/Intel/GENUINO101>.
16. **play-zone.ch.** Arduino GSM Shield 2 (Antenna connector). [Online] <http://www.play-zone.ch/de/elektronik-kit-zubehoer/avr-arduino-freeduino/arduino-shields/arduino-gsm-shield-2-antenna-connector.html>.
17. **wikipedia.org.** Web Application Framework. [Online] https://de.wikipedia.org/wiki/Web_Application_Framework.

18. **nodecode.de.** 4 Open-Source CSS-Frameworks im Überblick. [Online] <http://nodecode.de/open-source-css-frameworks>.
19. **slant.co.** What is the best CSS framework? [Online] http://www.slant.co/topics/150/compare/~bootstrap_vs_foundation_vs_semantic-ui.
20. **pugjs.** Jade - robust, elegant, feature rich template engine for Node.js. [Online] <https://github.com/pugjs/pug>.
21. **wikipedia.org.** Atom (Texteditor). [Online] [https://de.wikipedia.org/wiki/Atom_\(Texteditor\)](https://de.wikipedia.org/wiki/Atom_(Texteditor)).
22. **comparis.ch.** Netzabdeckung: So finden Handynutzer den besten Empfang. [Online] <https://www.comparis.ch/telecom/mobile/news/2014/12/netzabdeckung-schweiz.aspx>.
23. **migros.ch.** Tarife & Bestimmungen in Detail. [Online] <https://shop.m-budget.migros.ch/de/mobile-prepaid/tarife/details>.
24. **obiwian.** Das "richtige" Netzteil für Arduino UNO. [Online] <http://forum.arduino.cc/index.php?topic=162277.0>.
25. **arduino.cc.** ARDUINO PIN CURRENT LIMITATIONS. [Online] <http://playground.arduino.cc/Main/ArduinoPinCurrentLimitations>.
26. **codeproject.com.** Arduino Software Reset. [Online] <http://www.codeproject.com/Articles/1012319/Arduino-Software-Reset>.
27. **adafruit.com.** Optimizing SRAM. [Online] <https://learn.adafruit.com/memories-of-an-arduino/optimizing-sram>.
28. **abcminiuser.** GCC and the PROGMEM Attribute. [Online] <https://github.com/abcminiuser/avr-tutorials/blob/master/Progmem/Output/Progmem.pdf>.
29. **DavyLandman.** Arduino Library for AES Encryption. [Online] <https://github.com/DavyLandman/AESLib>.
30. **recalll.co.** NodeJS AES CBC Encryption [Node.js and crypto]. [Online] <https://recalll.co/app/?q=%20%20NodeJS%20AES%20CBC%20Encryption%20%5BNode.js%20and%20crypto%5D%0A%20%20>.
31. **nodejs.org.** crypto_createcipher_algorithm_password. [Online] https://nodejs.org/api/crypto.html#crypto_crypto_createcipher_algorithm_password.

Literaturverzeichnis

- C als erste Programmiersprache;** Buch
 Vieweg + Teubner; 7. Auflage; ISBN 978-3-8348-1221-6
- Arduino Language Reference;** Online Dokumentation
<https://www.arduino.cc/en/Reference/HomePage>
- stackoverflow;** Online Hilfeportal
<http://stackoverflow.com/>
- DevDocs;** Online Dokumentation
<http://devdocs.io/>
- Adafruit Learning System;** Online Lernportal
<https://learn.adafruit.com/>
- Docs | Node.js;** Online Dokumentation
<https://nodejs.org/en/docs/>

Socket.IO - Docs; Online Dokumentation

<http://socket.io/docs/>

jQuery API Documentation; Online Dokumentation

<http://api.jquery.com/>

Bootstrap; Online Dokumentation

<http://getbootstrap.com/>

Softwareverzeichnis

MySQL Workbench; Applikation

<http://dev.mysql.com/downloads/workbench/>

Wireshark; Applikation

<https://www.wireshark.org/download.html>

Atom; Applikation

<https://atom.io/>

Arduino Software; Applikation

<https://www.arduino.cc/en/Main/Software>

draw.io; Online App

<https://www.draw.io/>

Brunch; Entwicklungstool

<http://brunch.io/>

Let's Encrypt certbot; Entwicklungstool

<https://letsencrypt.org/getting-started/>

AESLib; Programmbibliothek

<https://github.com/DavyLandman/AESLib>

express; Programmbibliothek

<https://github.com/expressjs/express/>

body-parser; Programmbibliothek

<https://github.com/expressjs/body-parser>

mysql; Programmbibliothek

<https://github.com/felixge/node-mysql>

socket.io; Programmbibliothek

<https://github.com/socketio/socket.io>

socketio-auth; Programmbibliothek

<https://github.com/facundoolano/socketio-auth>

jQuery; Programmbibliothek

<https://jquery.com/download/>

Bootstrap; Programmbibliothek

<http://getbootstrap.com/getting-started/#download>

Bootstrap Switch; Programmbibliothek

<http://www.bootstrap-switch.org/>