



**NTB**



Interstaatliche Hochschule  
für Technik Buchs

FHO Fachhochschule Ostschweiz

# Türklingelanlage mit Standardkomponenten

Federico Crameri, Geo Bontognali



## Bachelorarbeit 2017

<b>Studiengang:</b>	Systemtechnik
<b>Profil:</b>	Informations- und Kommunikationssysteme
<b>Referent:</b>	Prof. Dr. Hauser-Ehninger Ulrich, MSc in Electronic Engineering ulrich.hauser@htwchur.ch, +41 (0)81 286 39 97
<b>Korreferent:</b>	Toggenburger Lukas, Master of Science FHO in Engineering lukas.toggenburger@htwchur.ch, +41 (0)81 286 27 22
<b>Industriepartner:</b>	SRM-Projects, Joeri Gredig joeri.gredig@srm-projects.ch, +41 (0)79 340 44 85



## Kurzfassung

In der heutigen Gesellschaft entwickelt sich alles mit erstaunlicher Geschwindigkeit. Diese rasante Entwicklung macht auch im Bereich der Gebäudetechnik keinen Halt. Die **Türklingelanlagen**, als Teil der Gebäudetechnik, sind auch davon betroffen, und haben sich in den letzten Jahren technologisch aber auch preislich weiterentwickelt.

Im Rahmen dieser Bachelorarbeit haben wir einen Türklingelanlage-Prototyp mit Standardkomponenten entwickelt. Wichtig für unsere Arbeit war zu überprüfen, inwiefern die heutigen Standard- und Open Source Komponenten für ein solches System geeignet sind. Während der Entwurfsphase wurden unterschiedliche Anforderungen definiert. Die Kommunikation über die Türklingelanlage muss durch Video und Audio Signale erfolgen, die Gegensprechanlage wurde komplett auf der digitalen Ebene realisiert. In einer modernen Welt, wo jeder Mensch ständig mit dem Internet verbunden ist, ist eine digitale Lösung wohl der einzige richtige Weg. Dieser Weg hat unterschiedliche Herausforderungen mit sich gebracht.

Das Endresultat ist eine digitale, flexible und zeitgemäße Türklingelanlage, welche mehrere Eingangstüren steuert und mit herkömmlichen Handys bedient werden kann

## **Abstract**

In a world where everything moves forward at the speed of light, home building technology is also not excluded from this rapid development.

Today, plenty of systems are built into new homes. One of them is the intercom.

During our bachelor thesis, we developed a prototype for an intercom, based on open source software and hardware components. One of the aims of this project was to evaluate and proof the ability of such components to handle this kind of application.

During the design phase, many different requirements were defined. The intercom needed to be able to provide an audio and video stream. Nowadays, everyone is always connected to the internet, thanks to the power of modern communication systems like Tablets and Smartphones. So, there was no doubt about the need of the intercom to be fully digital. As soon as things like digital real-time video- and audio transmissions come on the table, also a lot of different complications and challenges come with them, too.

As a result, we came up with a prototype that provides a flexible, up-to-date, and reasonably inexpensive solution for a modern house intercom system.



# Inhaltsverzeichnis

<b>1 Einführung</b>	<b>1</b>
1.1 Problemstellung . . . . .	1
1.2 Grundidee . . . . .	1
<b>2 Projektplanung</b>	<b>2</b>
2.1 Prozess . . . . .	2
2.2 Zeitplanung . . . . .	2
2.3 Versionierung . . . . .	2
2.4 Risikoanalyse . . . . .	4
2.4.1 Identifikation Analyse und Bewertung . . . . .	5
2.4.2 Bewertung Projekt Risiken . . . . .	6
2.5 Bewertung Technische Risiken . . . . .	6
2.5.1 Risikosteuerung & Projekt Massnahmen . . . . .	6
<b>3 Aktueller Stand der Technik</b>	<b>7</b>
3.1 Die Herausforderungen der Digitalisierung . . . . .	7
3.2 Marktsituation . . . . .	8
<b>4 Anforderungen</b>	<b>10</b>
4.1 Anforderungen . . . . .	10
4.2 Wunschanforderungen . . . . .	11
<b>5 Lösungskonzept</b>	<b>12</b>
<b>6 Umsetzung der Hardware</b>	<b>13</b>
6.1 Komponenten . . . . .	13
6.2 Stromspeisung . . . . .	14
6.3 Server . . . . .	15
6.4 Aussensprechstelle . . . . .	15
6.4.1 Problemen . . . . .	17
<b>7 Umsetzung der Software</b>	<b>19</b>
7.1 Programmiersprachen . . . . .	19
7.1.1 Java . . . . .	19
7.1.2 PHP/Javascript . . . . .	19
7.1.3 PHP Framework: Laravel . . . . .	20
7.2 System Übersicht . . . . .	20

7.3	Mühsame Security Policies . . . . .	21
7.4	WebRTC . . . . .	22
7.4.1	Signaling Process . . . . .	22
7.4.2	STUN Servers & Remote Verbindung . . . . .	23
7.5	Webapplikationen . . . . .	24
7.5.1	Client Webapplikation . . . . .	24
7.5.2	Aussensprechstelle Webapplikation . . . . .	26
7.5.3	Management Tool . . . . .	28
7.5.4	Remote Verbindung . . . . .	29
7.6	OS und Dienste . . . . .	29
7.6.1	Raspbian . . . . .	30
7.6.2	Taster Controller . . . . .	30
7.6.3	Speaker Controller . . . . .	31
7.6.4	Relay Controller . . . . .	31
7.7	Logging . . . . .	31
7.8	Watchdog . . . . .	32
<b>8</b>	<b>Prototyp Testplan</b>	<b>33</b>
8.1	Abnahme-Testplan . . . . .	33
8.2	Resultate . . . . .	34
8.2.1	Test 12: Verpasste Besuche sind in der Client-App ersichtlich . . . . .	34
8.2.2	Test 8: Die Auflösung des Videosignales wird evaluiert . . . . .	34
<b>9</b>	<b>Ausblick</b>	<b>36</b>
9.1	Verbesserungs- und Erweiterungsmöglichkeiten . . . . .	36
9.2	Einsatzmöglichkeiten . . . . .	36
<b>10</b>	<b>Schlussfolgerung</b>	<b>37</b>
<b>11</b>	<b>Anleitungen</b>	<b>38</b>
11.1	Aussensprechstelle Konfigurationsanleitung . . . . .	38
11.1.1	Aktuelle Stand . . . . .	38
11.1.2	Namen und Passwortkonzept . . . . .	38
11.1.3	Betriebssystem Installation . . . . .	38
11.1.4	Allgemeine Einstellungen . . . . .	38
11.1.5	Bildschirm Konfiguration . . . . .	39
11.1.6	Browser Kiosk-mode . . . . .	39
11.1.7	Aussensprechstelle Initialisierung . . . . .	40

11.1.8 Taster Controller . . . . .	41
11.1.9 Speaker Controller Service . . . . .	41
11.1.10 Watchdog/Watchdog deamon . . . . .	42
11.2 Server Konfigurationsanleitung . . . . .	42
11.2.1 Aktuelle Stand . . . . .	42
11.2.2 Namen und Passwortkonzept . . . . .	43
11.2.3 Software Installation . . . . .	43
11.2.4 Erstellung SSL Zertifikate . . . . .	43
11.2.5 Konfiguration von Nginx . . . . .	45
11.2.6 Deploy Webapplikationen . . . . .	47
11.2.7 Deploy Dienste und Services . . . . .	47
11.3 Installation Client-Webapplikation . . . . .	48
11.3.1 Installationsschritte . . . . .	48
<b>12 Anhang</b>	<b>49</b>
12.1 Messungsresultate . . . . .	49
12.2 Präsentation Prototyp . . . . .	50
<b>Abbildungsverzeichnis</b>	<b>51</b>
<b>Tabellenverzeichnis</b>	<b>52</b>
<b>Eidesstattliche Erklärung</b>	<b>55</b>

# 1 Einführung

## 1.1 Problemstellung

Heutzutage liefern diverse Hersteller verschiedene Lösungen für das Türglockensystem. Diese sind meistens Komplettsysteme, die nicht nur das einfache Klingeln ermöglichen, sondern auch Zusatzfunktionen wie das Video-Streaming anbieten. Diese Systeme sind aber meistens proprietär und werden, gemäß Abschnitt 3.2, für sehr hohe Preise verkauft.

Die Komponenten, die für solche Systeme notwendig sind, sind aber heutzutage kostengünstig auf dem Markt erhältlich. Das Erarbeiten preiswerter Lösungen müsste somit möglich sein.

Natürlich spielen die Kosten einer **Türklingelanlage** auf die Investitionen eines Neubaus keine so grosse Rolle. Sicher besteht aber in diesem Bereich eine Marktlücke und somit die Möglichkeit neue, bessere und günstigere Lösungen zu entwickeln.

## 1.2 Grundidee

Die Grundidee dieser Arbeit ist es, durch das Zusammenspiel verschiedener Systemen und Technologien, eine kostengünstige und funktionale **Türklingelanlage** zu entwickeln.

Um den Kostenfaktor zu berücksichtigen, soll die Anlage auf schon vorhandene Technologien und Hardware basieren. Somit fallen die hohen Kosten für die Beschaffung proprietärer Hardware weg.

In einer Zeit, in der die Hausautomation und das «Internet of things» immer mehr Bedeutung gewinnen, soll die **Türklingelanlage** diese Standards in Betracht ziehen. Dieses System soll den Benutzern ermöglichen, Ihre **Türklingelanlage** durch herkömmliche Smartphone oder Tablet zu bedienen.

Klingelt ein Besucher an der Eingangstüre, soll der Wohnungsbesitzer über sein Smartphone darauf aufmerksam gemacht werden. Über eine am Eingang installierte Kamera bekommt er auch die Möglichkeit den Besucher im Streaming zu sehen und die Türe, falls erwünscht, durch einen Handybefehl zu öffnen.

## 2 Projektplanung

### 2.1 Prozess

Als Entwicklungsprozess wird ein hybrides Vorgehensmodell eingesetzt, welcher in Abbildung 1 dargestellt wird. Im Rahmen einer Bachelorarbeit, in der die Anforderungen und Analysen schon im voraus im Fachmodul definiert worden sind, eignet sich am bestens ein lineares V-Modell. Ein solcher Prozess ist sehr schlank, übersichtlich und für diese Projektgrösse geeignet.

Was das V-Modell nicht erlaubt, ist eine ständige Iteration mit dem Kunden während der Entwurf/Implementierungsphase. Daraus ergibt sich, wie im Abbild unten gezeigt, ein hybrides Modell welches uns zulässt, trotz der klar definierten Anforderungen, während der Entwurf- und der Implementierungsphase ein agiles Vorgehen mit dem Kunden durchzuführen.

Die im Fachmodul geleistete Arbeit gehört zu den ersten zwei Phasen des Modells. Wie im linearen Vorgehensmodell vorgegeben, beginnt die nächste Phase der Arbeit sobald die vorherige Phase abgeschlossen ist. Die ganze Bachelorarbeit basiert auf Evaluationen und Entscheidungen, die in den ersten Phasen des Projekts getroffen worden sind.

### 2.2 Zeitplanung

Die folgenden Abbildungen stellen die Projektplanung und die Meilensteine zeitlich dar (siehe Abb. 2 & Abb. 3). In die erste Woche werden die Hardwarekomponenten, die mittlerweile schon bestellt wurden, getestet und zusammengebaut. Die nächsten zwei Hauptpunkte betreffen die Programmierung der Software, die in zwei Teile geteilt wurde.

Beim Teil 1 geht es um die Skripts die serverseitig kleine Aufgaben übernehmen, beim Teil 2 geht es um die Programmierung der Software. Da werden die Webapplikationen entwickelt, die auf den Aussensprechstellen und auf den mobilen Geräten der Bewohner ausgeführt werden sollen.

Die letzte Phase ist für die Optimierung und als Reserve gedacht.

### 2.3 Versionierung

Für die Versionierung und die gesamte Entwicklung wird das etablierte Open-Source Version-Control Software GIT verwendet. Das gesamte Quellcode, alle Bilder und Dokumente werden in einem Repository gespeichert und versioniert.

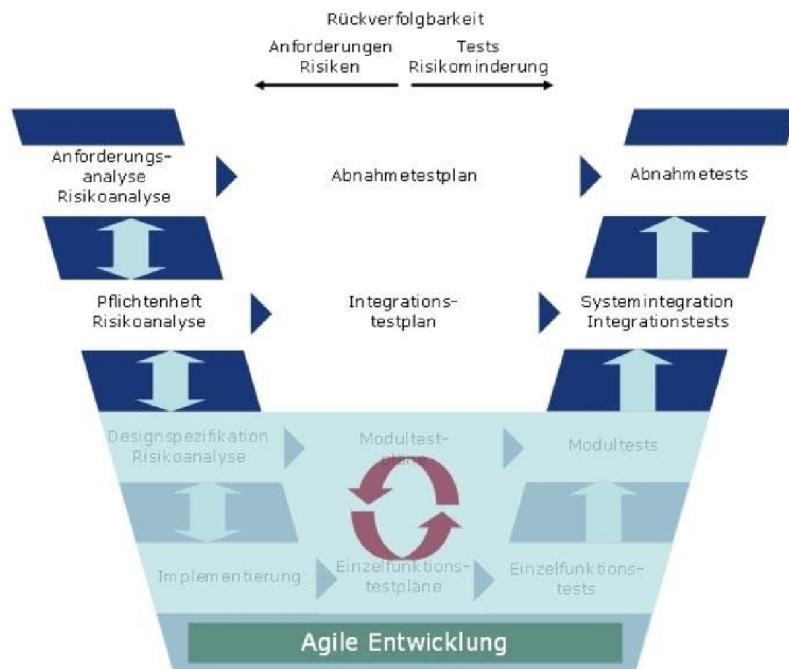


Abbildung 1: Hybrides Vorgehensmodell

(Quelle: <https://www.eckelmann.de/en/services/development-process-models>)

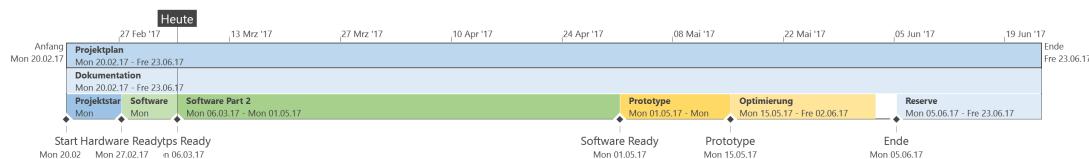


Abbildung 2: Zeitplanung mit Meilensteine

Während die Entwicklung wird der Quellcode aber nicht Open-Source sein. Das Quellcode und die gesamte Entwicklungsdokumentation für das Projekt wird Vertraulich gehalten und nur für die Entwickler und Projektteilnehmer verfügbar sein. Für die Repository und das Backup wird also den Consumer Dienst Bitbucket verwendet.

Bitbucket (siehe Abb. 4) ist ein webbasierter Filehosting-Dienst für Software-Entwicklungsprojekte, der die Versionsverwaltungssysteme Git und Mercurial unterstützt. Bitbucket ermöglicht auch die Zusammenarbeit von mehreren Benutzern

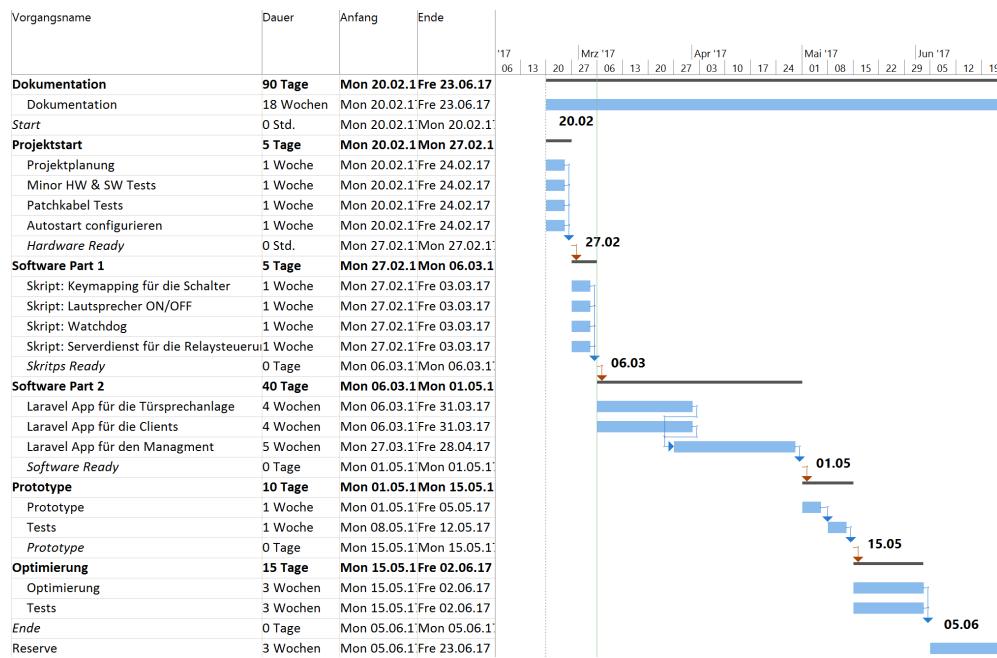


Abbildung 3: Projektplanung

am gleichen Projekt. Bitbucket ist für ein Projekt dieser grosse kostenfrei.



Abbildung 4: Bitbucket ist ein Filehosting und ein Dienst für die Versionskontrolle von Softwareprojekten

## 2.4 Risikoanalyse

Inhalt der Risikoanalyse ist die frühzeitige Identifikation, das Bewerten von Problemen und das Definieren von Massnahmen die zu der Risikominimierung führen. Der Risiko-Management Prozess nach ISO 31000:2009 umfasst folgende Hauptpunkte:

- Risikoidentifikation: Liefert eine Liste von möglichen Risiken die während der Implementierungsphase auftreten könnten.
- Aufgrund der Risikoidentifikation werden Zusammenhänge zwischen den Risiken und die Auswirkungen auf dem Projekt beurteilt.
- Risikobewertung: Zu jedem Risiko werden die Eintrittswahrscheinlichkeit sowie die Auswirkung auf das Gesamtprojekt abgeschätzt.
- Risikobewertung: Zu jedem Risiko werden die Eintrittswahrscheinlichkeit sowie die Auswirkung auf das Gesamtprojekt abgeschätzt.
- Risikosteuerung: Maßnahmen planen, um die gemessenen und analysierten Risiken zu steuern.

#### 2.4.1 Identifikation Analyse und Bewertung

Die Identifikation der Risiken erfolgte durch Brainstorming, aber auch durch Probleme die während den Sitzungen und der Projektplanerfassung aufgetaucht sind.

Um die aufgelisteten Risiken zu bewerten, wurde eine Risikomatrix eingesetzt. Diese soll eine visuelle Darstellung von Risikobewertungen geben.

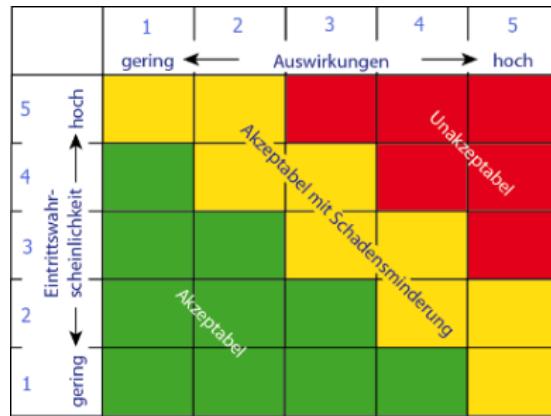


Abbildung 5: Risikomatrix

Die Auswirkungen sowie die Eintrittswahrscheinlichkeit werden mit einem Index (0 bis 5) von gering bis hoch eingestuft. Das Risiko ergibt sich durch die Multiplikation der beiden Achsen. Die Risiken die sich im roten Bereich der Matrix

befinden, müssen bei der Risikosteuerung/Maßnahmen sehr intensiv und detailliert behandelt werden, damit einer der beiden Faktoren minimiert werden kann.

### 2.4.2 Bewertung Projekt Risiken

Die folgende Tabellen stammen aus dem Fachmodul.

ID	Beschreibung	Wahrscheinlichkeit	Auswirkung	Risiko	Maßnahmen
PR1	Verspätete Hardware Lieferung	2	2	4	PM1
PR2	Ungenaue Zeiteinschätzung	2	5	10	PM2
PR3	Kostenüberschreitung / Teurer als Konkurrenzprodukte	1	5	5	PM3
PR4	Projekt entspricht nicht den Erwartungen des Kunden	2	3	6	PM4

1 Farbige Einstufung der Risiken anhand der Risikomatrix

Abbildung 6: Bewertung Projekt Risiken

### 2.5 Bewertung Technische Risiken

ID	Beschreibung	Wahrscheinlichkeit	Auswirkung	Risiko	Maßnahmen
TR1	Übertragene Bildqualität entspricht nicht den Erwartungen der Kunde	3	3	9	TM1
TR2	Web Applikation nicht funktionsfähig auf IOS Plattform	5	1	5	TM2
TR3	Ungenügende Hardware-Leistung um den Video-Stream im Betrieb zu nehmen	2	1	2	TM3
TR4	Sicherheitslücken	3	5	15	TM4
TR5	Latenz der Video/Audio Übertragung ist zu hoch	3	3	9	TM5

2 Farbige Einstufung der Risiken anhand der Risikomatrix

Abbildung 7: Bewertung Technische Risiken

### 2.5.1 Risikosteuerung & Projekt Massnahmen

ID	Massnahme Beschreibung
PM1	Nach Verfügbarkeit des Produkts suchen oder Ersatzprodukte bestellen. Hardware Komponenten werden so früh wie möglich bestellt.
PM2	Im Zeitplan genügend Reserve-Zeit einplanen. Modulanmeldung so ausstatten, dass im zweiten Semester wenige ECTS Punkten zu machen sind.
PM3	Konkurrenzprodukte nach Preis evaluieren. Eine Kostenschätzung der Hardware Komponenten muss im Fachmodul vorhanden sein.
PM4	Durch das Einsetzen eines hybriden Entwicklungsprozesses ist der Kunde immer über den aktuellen Stand des Projekts informiert und kann dementsprechend ihre eigenen Konzepte rechtzeitig einbringen.

Abbildung 8: Projekt Massnahmen

### 3 Aktueller Stand der Technik

Eine **Türklingelanlage** welche Audios und Videos überträgt, ist keine neue Erfindung. Auf dem Markt existieren bereits verschiedene Lösungen und das schon seit mehreren Jahren. Diese sind aber meistens analoge Systeme und verfügen über die Vorteile der Digitalisierung nicht.

Die Steuerung über eine Mobileapplikation ist aus diesem Grund bei solchen Lösungen ausgeschlossen.

Des Weiteren ermöglicht das digitale System den Zugriff auf die Videoübertragung von aussen, was mit analogen Technologien bis jetzt kaum realisierbar war. Die Integration der Digitalisierung im Projekt ermöglicht uns, alle Komponenten dynamisch zu vernetzen. Auch bezüglich Sicherheit bringt eine digitale Lösung Vorteile mit sich. Zum Beispiel, das Mitlauschen von Signalen, eine der häufigsten Angriffsarten, kann durch eine verschlüsselte Verbindung verhindert werden.

In den letzten Jahren sind die ersten, digitalen Lösungen mit **IP** Videoübertragung auf dem Markt gekommen. Die Digitalisierung in diesem Bereich ist den gigantischen Schritten im Bereich der Miniaturisierung und den immer schnelleren Internetzugängen (**xDSL**, **LTE**, usw) zu verdanken.

#### 3.1 Die Herausforderungen der Digitalisierung

Die Digitalisierung bringt, besonders bei den Video- und Audioübertragungen, nicht nur Vorteile mit sich. Während eine analoge Videoübertragung ziemlich mühelos erfolgt, muss im Falle einer digitalen Lösung das Video zuerst kodiert und anschliessend wieder dekodiert werden.

Die heutigen Kodierungsalgorithmen ermöglichen eine ziemlich schnelle Dekodie-



Abbildung 9: Analoge Türsprechanlage mit In-House Display

(Quelle: <https://fr.aliexpress.com/item/1-set-Smart-Home-Door-Intercom-System-One-to-One-Video-Door-Phone-7-inch/32812135796.html?spm=a2g0w.search0304.4.172.SNvC3o>)

rung. Mittlerweile hat jeder Smartphone genug Leistung um ein Full-**HD** Videostreaming von Youtube oder Netflix in Real Time zu dekodieren. Auf der anderen Seite ist die Kodierung ein sehr rechenintensiver Prozess und benötigt sehr viel Rechenleistung.

Jeder der schon mal mit Video-Editing zu tun hatte, weiss wie viel Zeit das Exportieren eines Videos dauern kann.

Die grösste Herausforderung für die Real Time digitale Video-, Audiokommunikation besteht also darin, die Kodierung und Dekodierung des Audios und Videosignals in vernünftiger Zeit durchzuführen.

Im Abschnitt 7.5.2 wird das Thema „Hardwareacceleration bzw. die Decoder/Encoder-Chips“ noch vertieft untersucht und dargestellt, welche Rolle Kodierung und Dekodierung spielen.

### 3.2 Marktsituation

Das Hauptziel dieser Bachelorarbeit ist die Entwicklung einer kostengünstigen Lösung für eine digitale, flexible und skalierbare **Türklingelanlage**. Tatsächlich ist es so, dass die bestehende Lösungen sehr teuer sind. Viele Produkte basieren auf Lösungen von Drittanbietern, **SIP** Gateways oder andere Elemente die Zusatzkosten verursachen. Das möchten wir alles vermeiden.

Eines der günstigsten Produkte das wir finden konnten ist das "*MyIntercom*" von Telecom Behnkle (siehe Abb. 10). Diese **Türklingelanlage** ist ziemlich flexibel und bietet die Möglichkeit, mehrere Türen anzuschliessen. Der Preis liegt beim



Abbildung 10: Telecom Behnkle MyIntercom

(Quelle: <http://www.myintercom.de/en/tradesmen/products/door-intercom-devices/one/myi0001>)

Basic-Modell, bei ungefähr 1'600.- CHF pro Türe.

Dank des Aufschwungs von Open Source Hardware, wie z.B. das Raspberry PI, Real Time Communication Protokolle wie **WebRTC**, sind wir der Meinung, dass es möglich sein muss, kostengünstigere Lösungen zu finden. In den folgenden Kapiteln geht es nun um die effektive Realisierung eines Prototyps, welcher die oben genannte Problemen adressiert.

## 4 Anforderungen

Für den Bachelorarbeit wurden die Anforderungen bereits in dem Fachmodul definiert.

### 4.1 Anforderungen

- A1. Es soll möglich sein, die Haustüre durch ein Signal zu öffnen.
- A2. Es soll möglich sein, ein Videosignal von der Aussensprechstelle zum Client zu streamen.
- A3. Es soll möglich sein, ein Audiosignal zwischen der Aussensprechstelle und der Client App bidirektional zu streamen.
- A4. Ein digitaler Bildschirm zeigt die Informationen der Bewohner (Name, Vorname, usw) an der Aussensprechstelle an.
- A5. Nach einem Stromunterbruch soll die Anlage automatisch wieder Starten und Funktionsbereit sein.
- A6. Den Datenverkehr zwischen den Endknoten muss Verschlüsselt sein.
- A7. Die Komponenten sollten zwischen -20C und +40C funktionsfähig sein.
- A8. Die Komponenten sollten auch im Fall hoher Feuchtigkeit funktionsfähig sein. (80%)
- A9. Die Kamera für das Videosignal muss eine Auflösung von mind. 1280x720 Pixel aufweisen.
- A10. Die Materialkosten pro Aussensprechstelle sollten 400.- nicht überschreiten.
- A11. Die Aussensprechstelle soll auch mit nasse/bedeckte Hände bedienbar sein.
- A12. Bei der Innenstelle ist es möglich das Mikrofon auszuschalten, um die Über-

tragung des Audiosignales zu unterdrücken.

## **4.2 Wunschanforderungen**

W1. Die Komponenten sollten die Speisung durch PoE erhalten.

W2. Die Kamera für das Videosignal muss eine Auflösung von 1920x1080 Pixel aufweisen.

W3. Verpasste Besuche sollten aufgezeichnet werden und in der Client App in Form von einem Foto und Notifikation sichtbar sein.

## 5 Lösungskonzept

Die Abb. 11 zeigt einen Überblick über die verschiedenen Hardwarekomponenten, die für die **Türklingelanlage** benötigt werden.

Es werden nun zwei Begriffe erklärt, die in diesem Dokument von grosse Bedeutung sind. Das erste ist die **Türklingelanlage**. Damit gemeint ist die Gesamtheit der Komponenten die denn Zusammen den Endprodukt darstellen.

Als **Aussensprechstelle** ist die Gesamtheit aller Komponenten des Endproduktes gemeint, die an der Eingangstüre installiert werden. Diese umfasst ein Mikrocontroller und die dazugehörige Module.

Räumlich von der **Aussensprechstelle** getrennt befindet sich der Server. Dieser besteht aus einem Mikrocontroller, der als Server im Einsatz steht, aus einem Switch der dazu dient die **Aussensprechstelle** mit Strom und Datenverbindung zu versorgen und aus einem Relais welches den Türöffner und die Glocke betätigt. Das System besteht aber nicht nur aus Hardware. Das Zusammenarbeiten

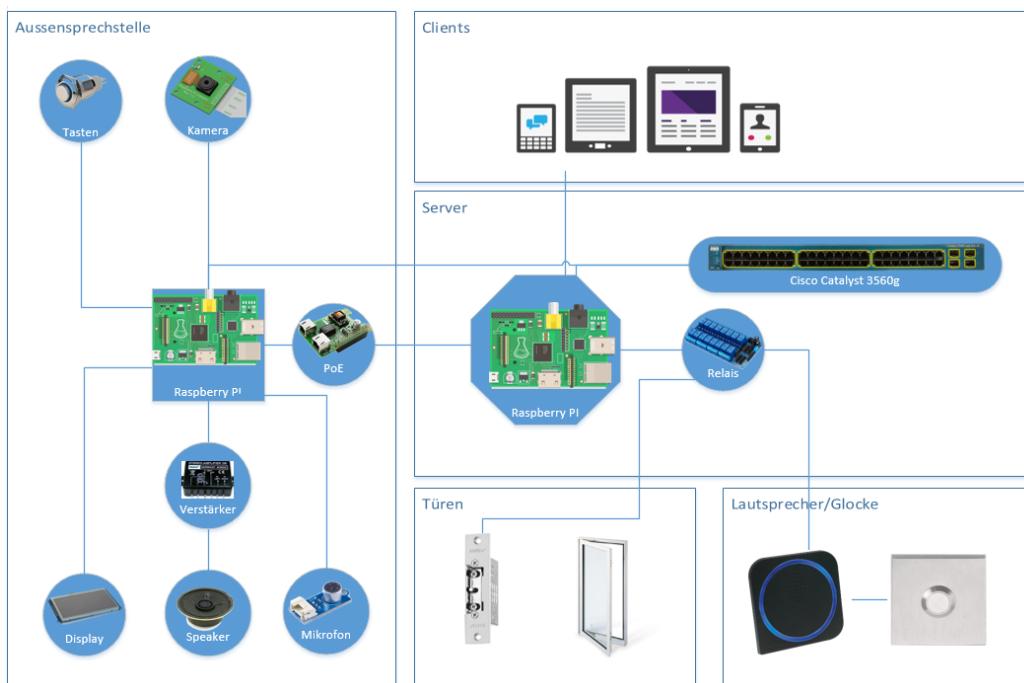


Abbildung 11: Hardware Ecosystem

der Hardware wird von viel Softwareelemente geregelt. Als erstes, wie bereits in der Projektplanung definiert, wird die Hardwareseite der Lösung realisiert. Sobald alle Hardwarekomponenten getestet und auf Kompatibilität geprüft worden sind, wird die Programmierung stattfinden.

Anzahl	Komponente	Preis
1	Raspberry Pi 3 Model B	50.-
1	Raspberry Gehäuse und Netzteil	25.-
2	8-Kanal Relais Modul	15.-
1	<i>Kleinmaterial</i>	15.-
<b>Total</b>		<b>140.-</b>

Tabelle 1: Server **HW** Komponenten

Anzahl	Komponente	Preis
1	Raspberry Pi 3 Model B	50.-
1	4" Bildschirm	64.-
1	Raspberry Kamera	59.-
1	<b>PoE</b> Adapter	50.-
3	Schalter	25.-
1	Mikrophon	12.-
1	Lautsprecher	9.-
1	Audio Verstärker	10.-
1	<i>Kleinmaterial / Gehäuse</i>	50.-
<b>Total</b>		<b>329.-</b>

Tabelle 2: Aussensprechstelle **HW** Komponenten

## 6 Umsetzung der Hardware

### 6.1 Komponenten

Das System wird hardwareseitig grob in zwei Teile unterteilt, den Server und die **Aussensprechstelle**.

Die Tabelle 1 und die Tabelle 2 zeigen die benötigten Hardwarekomponenten, welchen an den jeweiligen Stellen eingebaut werden.

Um den Überblick über die Kosten aller Hardwarekomponenten zu behalten, sind hier auch die Preisen aufgelistet. Dabei ist es wichtig sicherzustellen, dass die gesamten Hardwarekosten diejenigen der von der Konkurrenz angebotenen Produkte nicht übersteigen (siehe Abschnitt 4.1 Anforderung A10).

Die Einkaufspreise sind nur Richtpreise, da es sich um Standardkomponenten handelt und die Marktpreise sich ständig und schnell ändern können. Die Summen sind als Kostenschätzung zu betrachten. (Stand Frühjahr 2017).

## 6.2 Stromspeisung

Ein Ziel unserer Lösung ist die Installationskosten zu senken und die Montage zu vereinfachen. Aus diesem Grund war für unsere Lösung wichtig, **PoE** zu verwenden. In modernen Haushalte werden meistens Ethernet Verkabelungen verlegt und dank PoE ist nur noch ein Kabel, welches Strom und Konnektivität gewährleistet, notwendig.

Zusätzlich benötigt das System noch eine Leitung die den Türöffner steuert. Auch diese Endinstallation kann vereinfacht werden wenn man, anstatt ein dediziertes Kabel zwischen Server und Türöffner einzuziehen, zwei Drähte des bereits installierten Ethernet Kabels verwendet.

Für den Projekt verwendete Cisco Catalyst 3560g, welcher für den **PoE** Stromversorgung zuständig ist, verwendet die Phantomspeisung oder Mode A [20]. Das heisst, dass die mit der Datenübertragung belegten Drähte mit der Stromversorgung überlagert werden. Dies ist möglich da die Frequenz der Elektrizität 50 Hz beträgt und die der Datenübertragungen im Bereich von 10-100MHz liegt.

Bei einer zukünftige Beschaffung von einem aktuellere Switch muss speziell auf die Stromspeisungstyp (Phantom/Spare Pairs Speisung) geachtet werden.

STANDARD	SOURCE								COMMENTS
	1	2	3	4	5	6	7	8	
IEEE 802.3af using data pairs	RX DC+	RX DC+	TX DC-	spare	spare	TX DC-	spare	spare	Industry Standard for Embedded POE  <b>(used by Cisco Catalyst Switches)</b>

Abbildung 12: Catalyst 3560g **PoE** Pinbelegung

Wie im Abb. 13 dargestellt werden die Adern 7 und 8 dazu verwendet um den Türöffner zu betätigen. Aus den 3 verbliebenen Adernpaaren kann maximal die Ethernetkategorie 100BASE-T erreicht werden. Da aber **WebRTC** eine erhebliche kleinere Bandbreite in Anspruch nimmt, stellt es für die **Aussensprechstelle** kein Hindernis dar.

Durch eine Messung auf das Interface des Switches, an welches die Aussensprechstelle angeschlossen ist, konnte die exakte Bandbreite festgestellt werden. Die Messung wurde mit einem leistungsfähigen Prozessor durchgeführt. Damit wird verhindert, dass die Auflösung der Videoübertragung von den Raspberries gedrosselt wird. (Diese Problematik wird im Abschnitt 7.5.2 genau erläutert)

Mit einer hochauflösenden Videoübertragung wurde eine Datenübertragungsrate auf das Interface von 551Kbit/s festgestellt. Die detaillierten Resultate der Messungen sind im Anhang 12.1 zu finden.

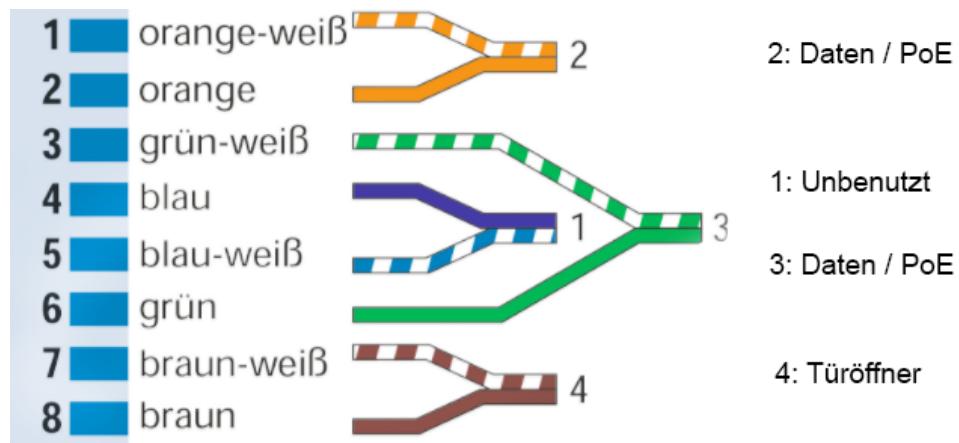


Abbildung 13: Cat. 7 Ethernet Pinbelegung der **Aussensprechstelle**

### 6.3 Server

Der Server wird mit einem Relais-Board verbunden um die Gongs und die Türöffner zu bedienen. An dieser Stelle ist die Hardwarekonfiguration sehr einfach. Mit der aktuellen Hardwarekonfiguration könnten bis 8 Wohnungen und 8 Aussensprechstellen angeschlossen werden. Die Abb. 14 und die Abb. 15 zeigen die Pinbelegung auf dem Pi und auf dem Relais-Board. Die Tabelle 3 zeigt wie die verschiedenen Pins miteinander verbunden werden.

### 6.4 Aussensprechstelle

Bei der **Aussensprechstelle** wird auch ein Raspberry Pi eingesetzt. Hier sind mehrere Zusatzkomponenten notwendig. Die Speisung, wie oben schon erwähnt, erfolgt an dieser Stelle über **PoE**. Aus diesem Grund ist ein **PoE-Splitter** vorhanden.

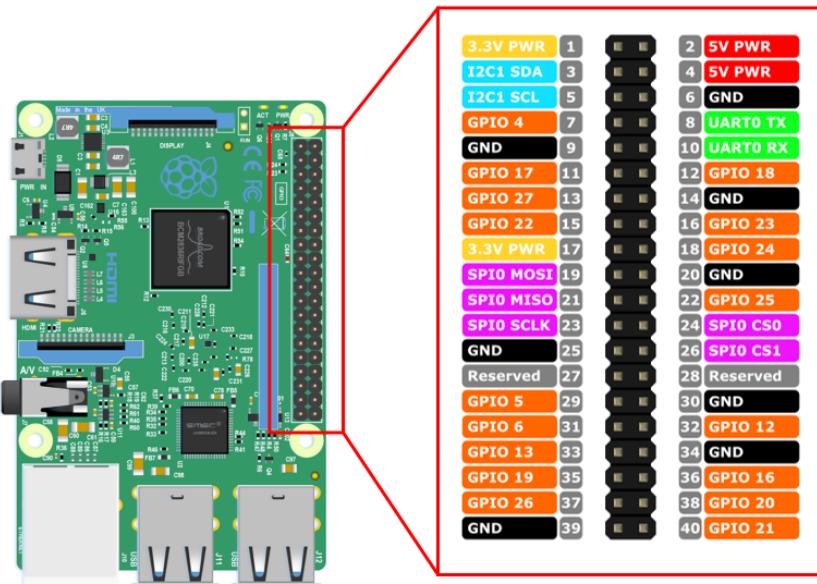


Abbildung 14: Pinbelegung der **Aussensprechstelle**

(Quelle: <http://fablabromagna.org/blog/seminario-iot-presso-corso-di-laurea-ingegneria-e-scienze-informatiche-alma-mater-polo-di-cesena/>)

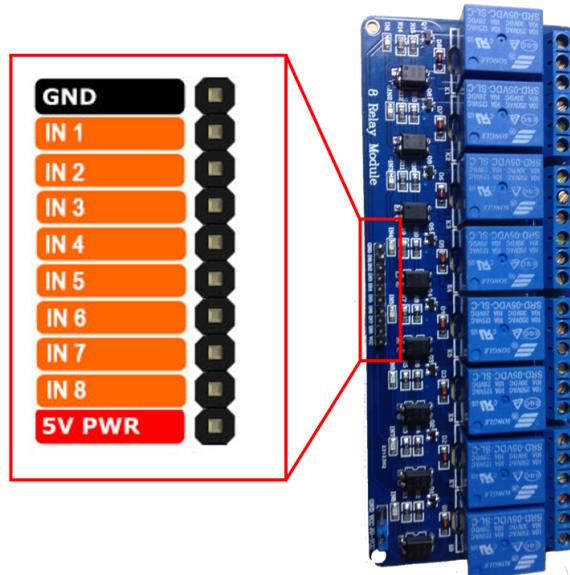


Abbildung 15: Pinbelegung für das Relais-Modul

Für die Audiomeldung sind ein kleiner Lautsprecher und ein Verstärker notwendig. Der Chinch Anschluss des Raspberries Pi hat eine zu niedrige Ausgangsspannung.

Pi GPIO (PIN)	Relais IN (Board Nr)	Funktion
GPIO4 (7)	IN1 (1)	Gong WG.1
GPIO17 (11)	IN2 (1)	Gong WG.2
GPIO27 (13)	IN3 (1)	Gong WG.3
GPIO22 (15)	IN4 (1)	Gong WG.4
GPIO5 (29)	IN5 (1)	Gong WG.5
GPIO6 (31)	IN6 (1)	Gong WG.6
GPIO13 (33)	IN7 (1)	Gong WG.7
GPIO19 (35)	IN8 (1)	Gong WG.8
GPIO18 (12)	IN1 (2)	Türöffner Türe 1
GPIO23 (16)	IN2 (2)	Türöffner Türe 2
GPIO24 (18)	IN3 (2)	Türöffner Türe 3
GPIO25 (22)	IN4 (2)	Türöffner Türe 4
GPIO12 (32)	IN5 (2)	Türöffner Türe 5
GPIO16 (36)	IN6 (2)	Türöffner Türe 6
GPIO20 (38)	IN7 (2)	Türöffner Türe 7
GPIO21 (40)	IN8 (2)	Türöffner Türe 8

Tabelle 3: PIN-Zuweisung zwischen den Server und die Relais Module

Pi GPIO (PIN)	Schalter	Funktion
GPIO16 (36)	Schalter Links	Nach Links Scrollen
GPIO20 (38)	Schalter Mitte	Glocke läuten
GPIO21 (40)	Schalter Rechts	Nach Rechts Scrollen

Tabelle 4: PIN-Zuweisung zwischen den Raspberry PI und die Schalter

leistung um den Lautsprecher direkt anschliessen zu können.

Die drei Schalter, die für die Bedienung der **Aussensprechstelle** notwendig sind, werden an die **GPIOs** des Raspberries PI angeschlossen. Die Tabelle 4 zeigt die Pinbelegung.

Eine Übersicht über die verwendeten Komponenten ist im Anhang 12.2 ersichtlich.

#### 6.4.1 Problemen

Während der Zusammenstellung der Aussensprechstelle sind die erste unvorhergesehene Problemen aufgetaucht. Die Audiowiedergabe und Audioaufnahme stellten eine grössere Herausforderung als geplant dar.

#### Audiowiedergabe

Die grösste Problematik bei der Audiowiedergabe besteht darin, dass die Mas-

sen des Raspberrys Pi, des Verstärkers und des Audio-Interface gekoppelt sind. Das führt zu Brummschleifen, die wiederum Störsignale auf dem Audio-Ausgang erzeugen. Um das zu vermeiden, muss an dieser Stelle ein Massentrennfilter eingesetzt werden.

Die Störsignale sind nun fast komplett verschwunden, nur ein winziges Hintergrundgeräusch ist immer noch vorhanden. Um dieses Problem umzugehen, wird ein zusätzliches Relais installiert, welches den Lautsprecherstromkreis bei Nichtnutzung unterbricht.

### Das Mikrofon

Das Problem der Audioaufnahme liegt beim Mikrofon selber. Der Raspberry Pi besitzt kein integriertes Audio-Input. Aus diesem Grund wurde ein **USB**-Audio-Interface verwendet. Es hat sich aber herausgestellt, dass es nicht so einfach ist, kostengünstige und qualitatives **USB** Mikrophone zu finden. Die meisten Produkte sind nicht für den Outdoor Betrieb gedacht, und oft ist die Empfindlichkeit zu gering, um ein qualitatives Audiosignal aufzunehmen. Für unseren Prototyp wird das eingesetzte Mikrofon völlig ausreichen, für ein gut funktionierendes Endprodukt sollte man ein besseres Mikrofon einbauen.

## 7 Umsetzung der Software

Die Hardware ist nun vollständig und dient als Basis für die Entwicklung der Softwarekomponenten die für das System notwendig sind. Die verwendete Software und Programmiersprachen wurden im Fachmodul evaluiert.

### 7.1 Programmiersprachen

Das System besteht aus mehreren Programmen und Diensten. Für die Entwicklung werden folgende Programmiersprachen eingesetzt:

- Java
- Javascript
- **PHP**

In Verbindung mit **PHP** kommt natürlich die Markup-Languages **HTML5/CSS** zur Anwendung, welche für die graphische Darstellung der Webapplikationen notwendig ist.

#### 7.1.1 Java

Alle Dienste die serverseitig und ohne Interaktion mit dem Enduser ausgeführt werden, werden in Java programmiert. Als stark typisierte und objektorientierte Programmiersprache eignet sich Java für dieses Projekt bestens. Für Java sind auch unzählige Libraries verfügbar, insbesondere für die Hardwaresteuerung des Raspberry Pi. Eine zweite Variante wäre Python gewesen, die auch den Raspberry sehr gut unterstützt. Python ist aber zu wenig typisiert und eher für kleinere Softwarestücke gedacht.

#### 7.1.2 PHP/Javascript

Sowohl die **Client Applikation** als auch die Applikation an der **Aussensprechstelle** werden Web-Applikationen sein. Diese ermöglichen eine schnelle und zeitgemäße Softwareentwicklung. Für dieses Projekt ist die Systemeingriffstiefe von Webapplikationen jedenfalls ausreichend. Lediglich der Zugriff auf Mikrofon, Lautsprecher und Kamera muss garantiert werden. Ein weiterer Punkt zugunsten einer Webapplikation ist die Kompatibilität der Cross-Plattform.

Aus diesem Grund haben wir uns für **PHP** (objektorientiert) in Kombination mit

Javascript/**HTML/CSS** entschieden. Eine zweite Variante wäre Java EE gewesen. Java EE eignet sich aber vor allem für grosse Softwarelösungen und bietet als gesamten Framework viel mehr als das was dieses Projekt benötigt.

### 7.1.3 PHP Framework: Laravel

Für die Entwicklung der Webapplikationen wird Laravel als **PHP** Framework eingesetzt. Laravel ist ein Open Source **PHP** Web-Application-Framework, das sich für kleine bis zu mittelgrosse Projekte eignet. Laravel beruht auf dem ModelView-Controller-Muster und ermöglicht eine objektorientierte Programmierung in **PHP**.

## 7.2 System Übersicht

Das System besteht aus mehreren Softwarekomponenten die zusammenarbeiten müssen (siehe Abb. 16). Die Vertraulichkeit der Kommunikation zwischen den Knoten ist dank **TLS** immer gewährleistet. Die einzelnen Komponenten, sowie das Thema Sicherheit, werden in den nächsten Kapiteln genauer beschrieben.

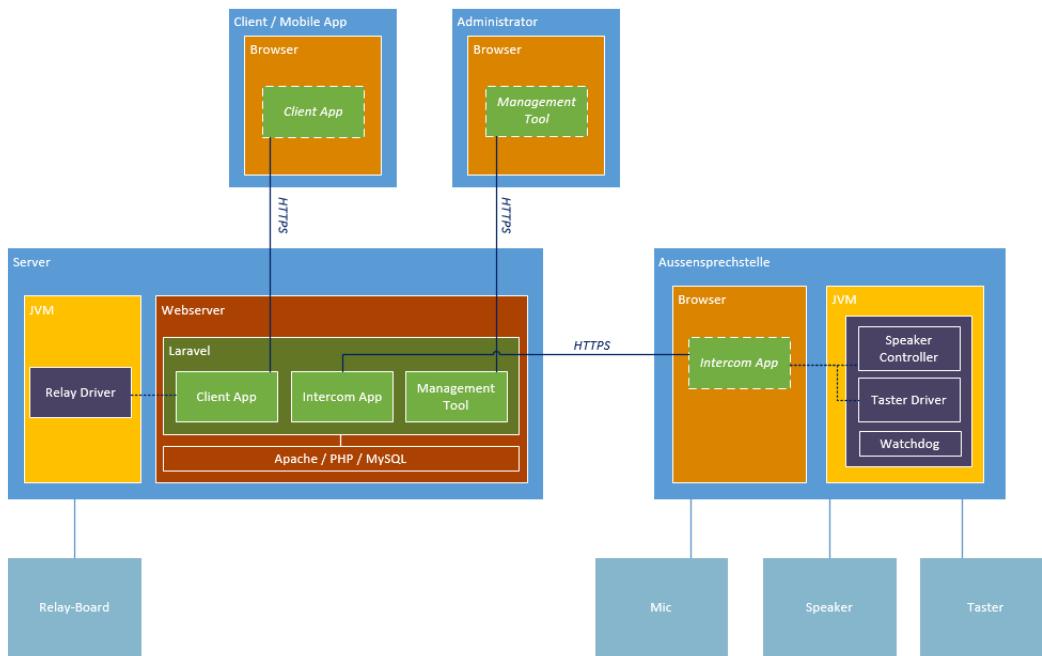


Abbildung 16: Software Ecosystem

Die Software wird in zwei Gruppen unterteilt. Einerseits gibt es alle Dienste/Daemons (*Violett*) die Lokal ausgeführt werden und quasi das Backend des Systems

darstellen.

Die zweite Gruppe beinhaltet die Webapplikationen (*Grün*), die eine **GUI** besitzen und für die Interaktion mit dem System gedacht sind. Darunter zählen die Client-App für die Bewohner, die Applikation bei der **Aussensprechstelle** wo die Bewohner angezeigt werden und das Management Tool.

Die Audios/Videoskommunikation zwischen der **Aussensprechstelle** und den **Client Applikationen** wird mithilfe von **WebRTC** realisiert.

### 7.3 Mühsame Security Policies

Die Sicherheit spielt für dieses System eine grosse Rolle. Aus diesem Grund wurde von Anfang an geplant, den ganzen Datenverkehr mit **TLS** zu verschlüsseln. Auch **WebRTC** selber weigert sich zu funktionieren, wenn keine gültige **HTTP**-**PS** Verbindung vorhanden ist.

Die heutigen Security-Policies der moderne Browser waren eine Hürde für die Entwicklung dieses Prototyps. Es ist zum Beispiel nicht mehr möglich, den Browser so einzustellen, dass die Zertifikatfehler ignoriert werden. Das hat als Folge, dass auch während der Entwicklung das Zertifikat gültig und signiert sein muss, ansonsten funktioniert **WebRTC** nicht.

Dies hat uns während der Entwicklung sehr viel Zeit gekostet. Zertifikate sind immer an einem Hostname oder an einer **IP** Adresse gebunden. Folge dessen mussten wir bei jeder Netzwerkanpassung alle Zertifikate nochmals generieren. Zusätzlich hat Google Chrome während der Entwicklung dieses Projekts mit der Version 58 die Security Policies geändert. [8] Nach dem Update brauchten die Self-Signed-Certificates ein zusätzliches Feld für den Subject-Alternative-Name (SAN). Im Netz war am Anfang sehr wenig Hilfe zu finden und das hat auch nochmals viel Zeit gekostet.

*"[...] RFC 2818 describes two methods to match a domain name against a certificate: using the available names within the subjectAlternativeName extension, or, in the absence of a SAN extension, falling back to the commonName. The fallback to the commonName was deprecated in RFC 2818, but support remains in a number of TLS clients, often incorrectly. [...]"*

[ Deprecations and Removals in Chrome 58, developers.google.com ]

## 7.4 WebRTC

**WebRTC** ist ein offener Standard, der eine Sammlung von Kommunikationsprotokollen und API beinhaltet. Die Standardisierung wird mehrheitlich betrieben und von Google, Mozilla Foundation und Opera Software unterstützt. **WebRTC** basiert auf **HTML5** und Javascript und die Audio/Video Übertragung erfolgt über eine direkte Verbindung zwischen den Sprechpartnern (Peer-to-Peer).

**WebRTC** wird hauptsächlich für die Entwicklung von Videokonferenzprogrammen verwendet. Die Natur dieses Projekt ist allerdings nicht dieselbe wie die herkömmliche Real-Time-Communication Applikationen. Glücklicherweise wurde **WebRTC** so entwickelt, um möglichst viel Flexibilität zu garantieren. Aus diesem Grund beinhaltet der **WebRTC**-Standard keine Definition für den SignalingProcess, welcher zusammen mit dem **ICE** (Interactive Connectivity Establishment) für den Verbindungsauftbau zwischen den Sprechpartnern zuständig ist.

*"The thinking behind **WebRTC** call setup has been to fully specify and control the media plane, but to leave the signaling plane up to the application as much as possible. The rationale is that different applications may prefer to use different protocols, such as the existing SIP or Jingle call signaling protocols, or something custom to the particular application, perhaps for a novel use case. [...]"*

[ Sam Dutton, HTML5Rocks.com ]

### 7.4.1 Signaling Process

Ähnlich wie bei **VoIP**-Telefonie (**SIP**), brauchen die Sprechpartner, um die Verbindung zu initialisieren, einen gemeinsam bekannten Knoten (siehe Abb. 17). In den meisten Fällen ist einem Partner die logische Adressierung des anderen Partners nicht bekannt. Es besteht also keine Möglichkeit um eine **P2P** Verbindung auf einmal zu starten.

Im unseren Fall wäre dies theoretisch möglich, da die Position der **Aussensprechstelle** bzw. des Servers immer dieselbe sind. Allerdings wurde **WebRTC** nicht so konzipiert. Die Standard **WebRTC API** beinhaltet kein Konstrukt um eine Verbindung anhand von bekannten **IP**-Adressen aufzubauen zu können.

Im Internet sind es mehrere Signaling-Server-Libraries verfügbar, diese sind allerdings für andere Anwendungen gedacht. Im unseren System wird beispielsweise

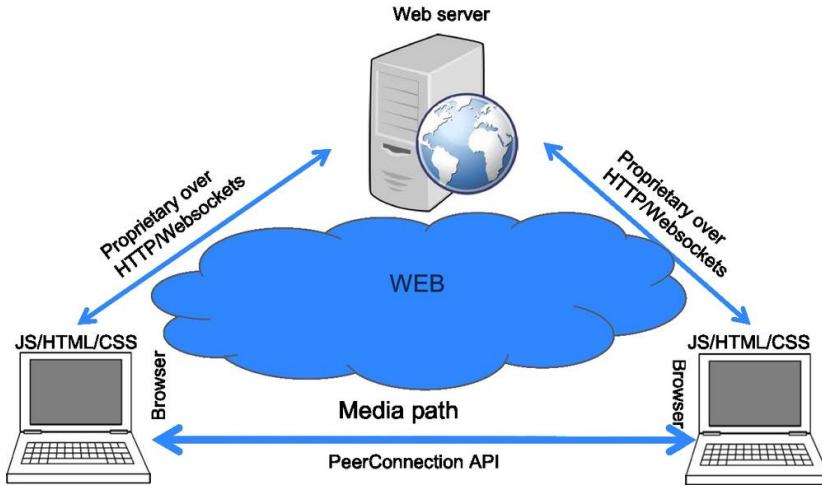


Abbildung 17: Der Signaling Prozess  
 (Quelle: <http://sangigi-fuchsia.fr/le-webrtc/>)

nie ein Anruf von der **Aussensprechstelle** zur **Client Applikation** gestartet, sondern lediglich umgekehrt.

Für die Absichten unseres Projekts wurde ein eigener Signaling-Server entwickelt, der auf dem Server ausgeführt wird. Somit bleibt der Datenverkehr zwischen der **Client Applikation** und der **Aussensprechstelle** während des ganzen Ablaufes innerhalb des lokalen Netzwerkes. Das aber natürlich nur, solange der Bewohner sich zu Hause befindet.

#### 7.4.2 STUN Servers & Remote Verbindung

Eine Anforderung des Systems ist die Möglichkeit, auch ausserhalb des Heimnetzes mit den **Aussensprechstelle** sich verbinden zu können. Hier stellt das **NAT**-Protokoll (Network Adress Translation) ein Problem dar.

Nach dem Signaling-Prozess wird das **ICE**-Prozess gestartet. Hier tauschen sich die zwei Partner Informationen über die eigene Adressierung und den *best path* aus [13]. Falls sich ein Sprechpartner hinter einem **NAT**-Knoten befindet, wird für den anderen unmöglich sein eine Verbindung aufzubauen. Hier kommen die **STUN**-Server im Spiel. [2] Ähnlich wie beim Signalisierungsprozess stehen **STUN**-Server als Hilfe für den Verbindungsaufbau da (siehe Abb. 18).

**STUN**-Server informieren die Clients über jegliche **NAT** Konfigurationen die sich dazwischen befinden würden. Die beiden Sprechpartner erhalten somit Informationen über welche Ports und öffentliche Adressen die Verbindung initialisiert werden kann. Für die Entwicklung dieses Projektes werden die Google-**STUN**-

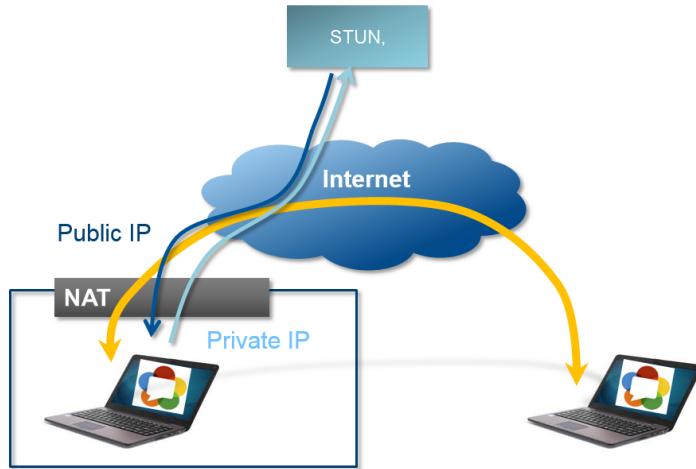


Abbildung 18: STUN Server

(Quelle: <https://realtimecommunication.wordpress.com/2015/05/29/crack-the-nat/>)

Server verwendet, welche kostenfrei zur Verfügung stehen.

Falls sich beide Sprechpartner im gleichen lokalen Netzwerk befinden, werden keine **STUN**-Server benötigt und der gesamte Datenverkehr bleibt innerhalb des Heimnetzwerkes.

## 7.5 Webapplikationen

Während die verschiedene Java Dienste relativ kleine Programme sind, besteht den gesamten Quellcode der Webapplikationen aus mehreren tausende Codezeilen.

Das **PHP**-Backend im Zusammenarbeit mit Javascript auf der Clientseite ist für die meisten Aufgaben der Anlage sowohl auch für einen Teil der Sicherheitsaspekte zuständig.

### 7.5.1 Client Webapplikation

Der Bewohner muss über eine Applikation verfügen, die auf dem Tablet oder Handy ausführbar sein muss. Mithilfe dieser App muss der Enduser folgendes können: Sich mit allen **Aussensprechstellen** verbinden können, ein Videosignal von der Kamera aller Eingänge erhalten, alle Türe öffnen und mit der Person bei der Türe über die Anlage kommunizieren können.

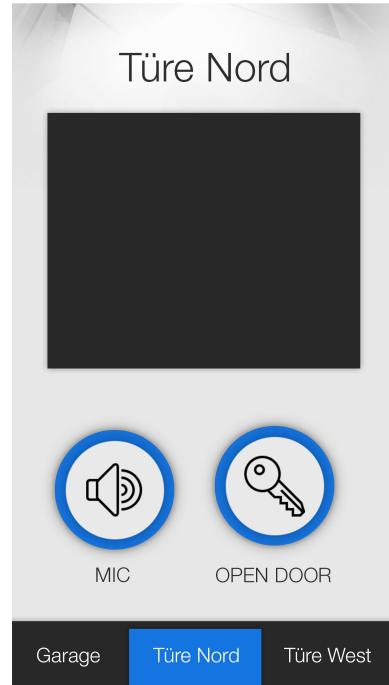


Abbildung 19: Design der Client-Webapp

Die Abb. 19 zeigt das Design der Webapplikation, hier speziell die Smartphone Version. Dank einem Responsive-Design wird die selbe Applikation auch auf andere Geräte wie z.B. Tablets oder Computers passend angezeigt.

Beim Design-Entwurf standen Übersichtlichkeit und Benutzerfreundlichkeit im Vordergrund. Aus diesem Grund werden die Tasten für die Audio-Kommunikation und für die Öffnung der Türe gross Angezeigt. Das Videostream der ausgewählten Türe wird sofort angezeigt und benötigt keine weitere Interaktion.

### Notifications

Wenn jemanden bei der Türe klingelt, wird eine Notification angezeigt. Um das zu erfolgen, muss die Webapplikation mindestens einmal gestartet werden. Dank den Notifications können auch Verpasste besuche an einem Späteren Zeitpunkt gesehen werden [16].

### Anmeldung & Multi-User

Aus Zeitgründen könnten wir die Anmeldeseite für die Client Applikation nicht vervollständigen. Momentan wird die Identität der Anwender lediglich über eine GET Variable definiert (siehe Abschnitt 11.3). Eine Anmeldeseite, welche die Identität der Anwohner überprüft, ist für das Endprodukt natürlich enorm wich-

tig. Die jetzige Lösung weist noch eine grosse Sicherheitslücke auf und ist nur für Test- und Vorführzwecke gedacht. Eine Login-Page wurde von Anfang an vorgesehen, und die Implementierung kann nun ohne Anpassung des bestehenden Sourcecodes vorgenommen werden.

### 7.5.2 Aussensprechstelle Webapplikation

Die **Aussensprechstelle** ist mit einem Bildschirm ausgestattet, der die Bewohnerliste anzeigt. Mithilfe von drei Schaltern kann man durchblättern und die Bewohner können angerufen werden (siehe Abb. 20).

Während der Bewohner die Möglichkeit hat, die Person an der Türe zu sehen, erhält der Besucher an der Türe kein Videosignal, auch wenn dies technisch absolut möglich wäre. Als Bewohner will man aber die Möglichkeit haben die Türe nicht zu öffnen oder dem Besucher die eigene Präsenz gar nicht bekannt zu geben.

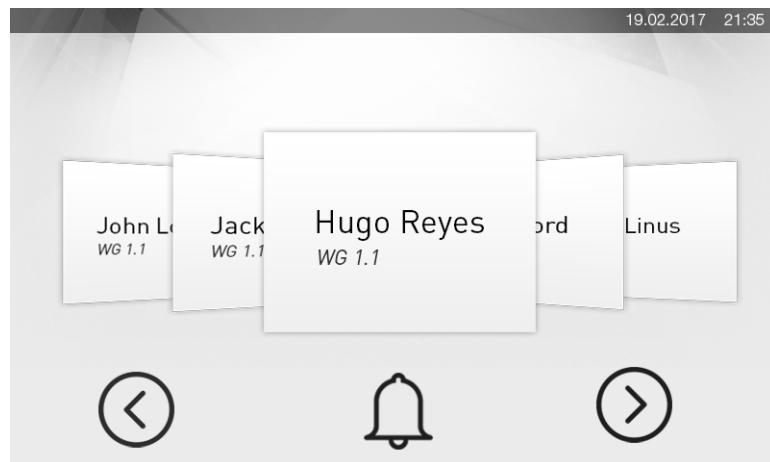


Abbildung 20: Design der Aussensprechstelle-Webapp

### Probleme bei der Videoübertragung

Nach dem ersten Test der Webapplikationen ist ein weiteres Problem aufgetaucht. Die Qualität der Videoübertragung war nicht immer befriedigend. Das Problem ist aber erst aufgetaucht, nach dem Deploy der Webapplikationen auf die endgültige Hardware installiert wurde.

Nach einer Problemanalyse konnte man folgendes feststellen:

Für das Video-Encoding verwendet **WebRTC** das VP8 Codec [22]. Leider unter-

	Raspberry Pi Model 3	Banana Pi M3
CPU Cores	4	8
CPU Design	Cortex A53	Cortex A7
CPU Frequenz	1.2GHz	1.8GHz
Memory	1GB DDR2	2GB DDR3
Memory Frequenz	400MHz	672MHz
H264 Decoding	1080P30	1080P60
H264 Encoding	1080P30	1080P60
Preis	CHF 50.0	CHF 99.00

Tabelle 5: Verwendete Raspberry Pi im Vergleich mit Banana Pi als Alternative

stützt den Raspberry Mikrocontroller keine Hardware acceleration für den VP8 codec. Unterstützt wird aber das "vector acceleration" welches aber softwaremässig erfolgt. Die **Aussensprechstelle** muss im Stande sein, die Codierung in Real-Time auszuführen, was den Raspberry Pi an seinen Grenzen bringt. Obwohl eine Kamera mit hoher Auflösung im Einsatz ist, wird **WebRTC** im Folge des niedrigen Framerates die Qualität des Stream verringern. Sobald die Qualität herabgesetzt ist, ist der Raspberry wieder im Stand die Codierung in Echtzeit durchzuführen.

Aufgrund der hohen Überlastung des Prozessors während der Kodierung, tauchen zusätzlich Wärmeabfuhrprobleme auf. Eine verlängerte Videostreaming-Session mit erhöhten Umgebungstemperaturen, könnte den Raspberry zum Absturz bringen.

Der Raspberry Pi 3 war während der Entwicklungsphase des Prototyps die richtige Entscheidung. Hauptgrund waren die hohe Kompatibilität, die Standardisierung eines sehr gut etablierten Produktes und die Stabilität. Dazu kommen noch die unzähligen Infos, Dokumentationen die im Internet über diese Micro Controller zu finden sind.

## Alternative

Mit den gesammelten Erfahrungen während der Prototypentwicklung könnte eine bessere Alternative zur Raspberry für eine Weiterentwicklung der Anlage ausgewertet werden.

Der Microcontroller Banana Pi M3 bietet im Gegensatz zum Raspberry erheblich mehr Datenverarbeitungsleistung (siehe Tabelle 5)).

Ein weiterer Vorteil des Banana Pi ist, dass der Raspbian **OS** ebenfalls unterstützt wird. Das mit dem Projekt mitgelieferte Image des Betriebssystems für die **Aussensprechstellen** könnte somit auf den neuen Microcontroller mit ge-

ringrem Aufwand aufgespielt werden. Auch die Verkabelung sollte kein Problem darstellen, da die Pinbelegung eins zu eins die des Raspberrys entspricht.

### 7.5.3 Management Tool

Um eine schnellere Inbetriebnahme und eine zentrale Verwaltung des Systems zu gewährleisten, wurde das **Management Tool** entwickelt. Diese Webapplikation, die die Erfassung von **Aussensprechstellen** und Bewohnern ermöglicht, wurde mit Webtechnologien entwickelt (**HTML**, **PHP**, Javascript) und wird zusammen mit den **MySQL**-Datenbank auf dem lokalen Raspberry-Server gehostet. Aus Sicherheitsgründen werden alle eingehenden und ausgehenden Verbindungen mittels **TLS** abhörsicher aufgebaut.

Grund für das Einsetzen von Webtechnologien sind die Plattformunabhängigkeit sowie die Einfachheit und die Standardisierung der Sprachen. Beim ersten Prototyp lag der Fokus auf die funktionalen Eigenschaften des Tools. Bei einer zukünftigen Weiterentwicklung des Produkts kann man, dank der Webtechnologien, mit geringerem Aufwand das Tool skalieren bzw. neue Features hinzufügen.

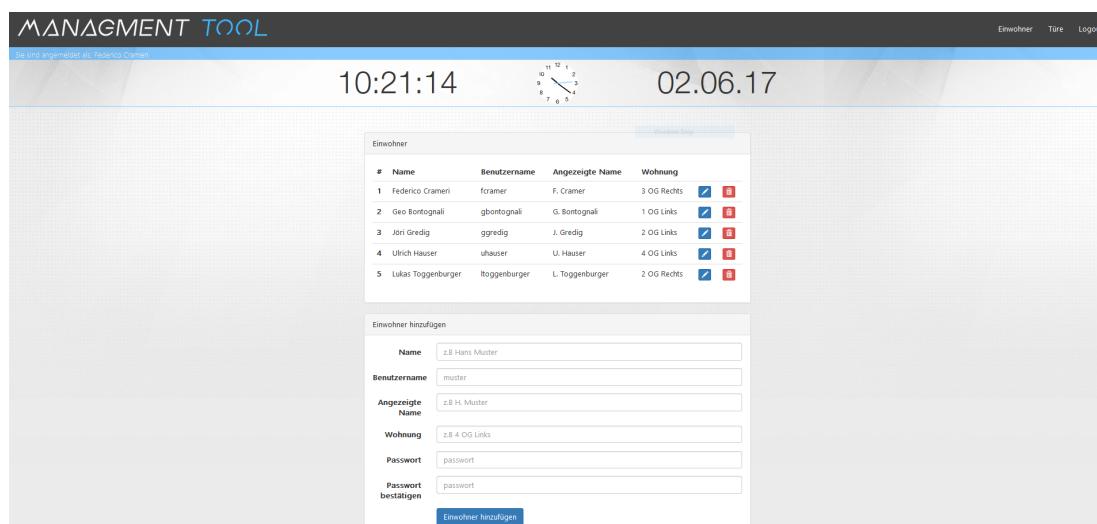


Abbildung 21: Design der Management tool

Das Tool ist mit einem Login versehen und die Vertraulichkeit ist somit garantiert.

### Bewohner

Unter der Bewohnerseite werden alle Wohnungen, beziehungsweise alle Bewohner,

aufgelistet. Diese verfügen über einen Benutzernamen und ein Passwort, die von der **Client Applikation** verwendet werden um eine sichere Authentifizierung beim Server zu gewährleisten. In diesem Bereich hat man die Möglichkeit sowohl der Name als auch die Position der Wohnung, welche an der **Aussensprechstelle** angezeigt werden, abzuändern.

## Türen

In diesem Abschnitt sind die Namen der Türen definiert, welche dann auf der **Client Applikation** angezeigt werden. Der Einbau einer neuen Türe muss im **Management Tool** definiert werden. Dabei ist zu beachten, dass der ID mit demjenigen der auf der neu installierten **Aussensprechstelle** übereinstimmt.

### 7.5.4 Remote Verbindung

Die Remote Verbindung auf der **Türklingelanlage** ist eine Feature, die leider aus Zeitgründen nicht implementiert werden konnte.

Stellt man sich folgende Situation vor: Der Wohnungsbesitzer ist unterwegs, niemand befindet sich in der Wohnung, und nun soll der Briefträger ein Päckchen liefern. Hier kommt die Remote Funktionalität ins Spiel. Über die **Client Applikation** kann die Identität der Person, welche vor dem Eingang steht, überprüft werden. Der Wohnungsbesitzer kann jetzt entscheiden, ob er die Tür öffnen will oder nicht. Bei einer zukünftigen Implementation der Remote Funktion soll besonders Wert auf die Sicherheit gelegt werden. Grund dafür ist, dass die Verbindung zwischen den Komponenten an dem lokalen Netz überlassen wurde. Das heisst, die Anlage ist Zeitweise möglichen bösartigen Angriffe aussetzen.

Diese Funktionalität, welche mit einem analogen System kaum realisierbar ist, würde die Vorteile der Digitalisierung noch deutlicher hervorheben. Aus Marketing Sicht stellt eine solche Funktionalität ein erheblicher Mehrwert dar.

## 7.6 OS und Dienste

Folgend beschrieben wir alle Dienste die auf dem Raspberry laufen werden. Diese werden benötigt um die Webapplikationen mit der Hardware zu verbinden.

### 7.6.1 Raspbian

Auf allen Raspberry Pi wurde der Betriebssystem Raspbian-Jessie installiert. Dieser wird von der Raspberry-Pi-Foundation mitgeliefert und gilt als besonders hochoptimierte **OS** für die mit niedriger Leistung und geringem Stromverbrauch **ARM** Prozessoren.

Raspbian basiert auf Debian welche unter der DFSG (Debian Free Software Guidelines) Lizenz steht [6]. Diese erlaubt der unbeschränkten Weitergabe der Software sowie abgeleitete und modifizierte Werke weiterzugeben. Raspbian enthält Java SE Plattformprodukte welche unter dem BCL(Oracle Binary Code License) lizenziert sind. Diese Lizenz gewährleistet die obengenannten Freiheiten ebenfalls.

### 7.6.2 Taster Controller

Die **Aussensprechstelle** wird durch 3 Schalter bedient. Die Aufgabe der Taster-Controller besteht darin, die **GPIOs** der Raspberry, welcher mit den Schaltern verbunden sind, abzuhören. Sobald ein Schalter gedrückt wird, wird eine Tastatureingabe simuliert. Durch die Simulation kann der Javascript-Code, der lokal im Browser ausgeführt wird, auf den Schalterdruck reagieren. Somit kann auf der **Aussensprechstelle**, dank der Webapplikation (siehe Abb. 20), einen Bewohner ausgewählt werden (Schalter Rechts und Links) und diesen dann auch angerufen werden (Schalter Mitte).

Ursprünglich wollten wir den Taster Controller, sowie alle anderen Dienste, als Daemon ausführen. Das hätte den Vorteil gehabt, dass der Daemon mittels eines üblichen Run-, Stop- oder Restart-Befehls gesteuert werden konnten. Per Definition ist ein Daemon benutzerunabhängig und genau dieser Ansatz war problematisch. Eine der eingesetzten Java-Library (Robot, um Keyevent zu simulieren) benötigt den Zugriff auf die **LXDE**-Desktopumgebung. Aus dem Grund, dass **LXDE** ein benutzerspezifischer Prozess ist, konnte der Taster-Controller nicht als Daemon ausgeführt werden.

Um das Problem umzugehen bietet **LXDE** einen Autostart. Im Unix Runlevel 5 wird gewartet bis die Desktopumgebung initialisiert ist und der Taster-Controller wird erst dann ausgeführt. Somit kann jetzt die eingesetzte Robot-Library auf den **LXDE** zugreifen und die Tastatureingabe simulieren.

### 7.6.3 Speaker Controller

Der Speaker-Controller ist ein kleiner Dienst, welcher den Lautsprecher ein- und ausschalten kann. Trotz einem Massentrennfilter sind immer noch leise Störsignale auf dem Audio-Ausgang vorhanden. Die Aufgabe des Speaker-Controllers besteht darin, die Stromspeisung des Speakers zu trennen, wenn er nicht verwendet wird. Somit ist das System energieeffizienter und unnötige Geräusche können vermieden werden. Der Speaker-Controller wird auf der **Aussensprechstelle** als Daemon ausgeführt.

Der Dienst besteht lediglich aus einem Socket-Server, der auf einem Signal wartet und durch die **GPIO** der Raspberry ein kleines Relais steuert. Das Signal kommt von der Javascriptseite der **Aussensprechstelle**-Webapplikation (*localhost*) und kann so bei Bedarf den Lautsprecher ein- bzw. ausschalten.

### 7.6.4 Relay Controller

Der Relais-Controller ist sehr ähnlich aufgebaut wie der Speaker-Controller. Auch hier handelt sich um einen kleinen Socket-Server, der auf ein Signal wartet und durch die GPIO des Raspberries ein oder mehrere Relais steuert.

Der Relais-Controller wird auf dem Server ausgeführt und wartet auf die Befehle der Webapplikationen. Das Relais ist am Türöffner und an den Gongs der Wohnungen angeschlossen.

Der Datenaustausch zwischen dem Relais-Controller und den Webapplikationen erfolgt in Form eines **JSON**-Strings.

Während beim Speaker-Controller die Befehle aus der Clientseite stammen, kommen die Daten beim Relais-Controller aus dem **JSON**-Backend vom Server selbst. Somit bleibt der Datenverkehr auf dem Localhost und werden Man-in-the-Middle oder Injection-Attacke ausgeschlossen. Aus diesem Grund lauscht dieser Server nur auf Verbindungen die vom Localhost stammen.

## 7.7 Logging

Für die Identifikation und Rückverfolgung von Fehlern sowie für das Monitoring sind Logs-File von grosser Bedeutung [23]. Diese werden bei allen Services und Dienste konsequent durchgeführt.

Das Logrotate wird nicht eingesetzt, stattdessen kümmert sich das Java-Runtime-

Environment um die Grösse des generiertes Log-Files. Da es sich noch um einen Prototyp handelt, wurde die Loggingstufe auf 7 eingestellt. Auf dieser Stufe werden alle Emergencynachrichten bis auf die Debugnachrichten in Logdateien gespeichert.

Gemäss dem FHS (Filesystem Hierarchy Standard) werden die Logs unter /var/log/Aussensprechstelle gesichert.

## 7.8 Watchdog

Die ganze Hardware, die an der Türe installiert wird, ist beim Endkunden schwer zugänglich. Sollte nun ein Problem mit dem System auftreten, müsste man die Anlage vor Ort zurücksetzen. In solchen Fällen hilft der Hardware-Watchdog, der auf dem Raspberry komplett unabhängig vom eigentlichen System läuft [11].

Der Vorteil eines Hardware-Watchdogs ist, dass wenn das System bzw. der Prozessor blockiert, diese unabhängige Hardware ihre Aufgabe weiterhin ausführt. Der Watchdog wird als standalone Gerät im Unix erkannt. Wird dieses Gerät einmal beschrieben, dann muss diese mit einem Zeitintervall von 15 Sekunden erneut beschrieben werden.

Ist diese Bedingung nicht erfüllt, wird dann einen Hardware-Reset von Watchdog durchgeführt und das System wird neugestartet. Das Beschreiben vom Watchdog-Gerät wird von einer Watchdog-Daemon übernommen. Durch die Konfigurationsdatei des Daemons können verschiedene Parameter des Systems wie Temperatur, Auslastung der Prozessor usw [18]. überwacht werden. Besonders relevant für die Türsprechanlage ist das PID-Monitoring. Diese ermöglicht das ständige Überprüfen von spezifischen Prozessen und Diensten die das System benötigt, um seinen Zweck als **Aussensprechstelle** zu erfüllen.

Sobald einer dieser Prozesse anhält wird das System innerhalb von 15 Sekunden neugestaltet. Ein solches Mechanismus steigert die Verfügbarkeit des Dienstes und ist für eine Türsprechanlage von grosser Bedeutung.

## 8 Prototyp Testplan

Überprüfung der Anforderungen.

### 8.1 Abnahme-Testplan

T1. Die Tür bei der Aussensprechstelle muss, nach betätigen der «Türe öffnen» Taste in der Client-App, geöffnet werden können.

T2. In der Client-App wird ein Videosignal von der Kamera der Aussensprechstelle erhalten.

T3. Durch bedienen der Client-App wird ein Audiosignal von der Aussensprechstelle erhalten und umgekehrt.

T4. An der Aussensprechstelle kann man die Namen der Bewohner lesen und auswählen.

T5. Die Stromversorgung der Anlage wird aus- und dann wieder eingeschaltet. Die Anlage muss ohne externen Eingriff wieder Starten.

T6. Mithilfe von Wireshark wird der Netzwerkverkehr analysiert. Der gesamte Datenverkehr aus der Anlage ist verschlüsselt.

T7. Feuchtigkeit, Temperatur und Betriebsbedingungen aus dem Datenblatt mit den Anforderungen vergleichen und überprüfen.

T8. Durch eine Analyse wird die effektive Auflösung des Videosignals evaluiert. Die Auflösung muss mindestens 720p (ev. 1080p) aufweisen.

T9. Sämtliche Materialkosten zusammenstellen und überprüfen. Die Kosten pro Aussensprechstelle müssen das in den Anforderungen definierte Maximum nicht überschreiten.

T10. Mit bedeckte (Skihandschuh) und/oder nassen Hände muss man bei der Aussensprechstelle Klingeln können.

T11. Bei der Aussensprechstelle ist nur das Netzwerkkabel zu finden, keine zusätzliche Speisung.

T12. Bei Klingeln und nicht Öffnen der Türe, muss in der Client-App der verpasste Besuch sichtbar sein.

T13. Durch Bedienen der Client-App ist es möglich, frei zu sprechen, ohne dass das Audiosignal an die Aussensprechstelle weitergeleitet wird.

Ergebnisse werden in einer Test-Traceability Matrix visuell festgehalten. Jede Anforderung wird durch einen Test überprüft. Beziehungsweise, jeder Test verifiziert nur die betroffene Anforderung. Damit werden Überschneidungen vermieden.

## 8.2 Resultate

Bis auf zwei wurden sonst alle Tests bestanden. (siehe Abb. 22)

Es werden nun die Anforderungen und Tests aufgelistet, welche nicht bestanden wurden.

### 8.2.1 Test 12: Verpasste Besuche sind in der Client-App ersichtlich

Bei Test 12 wird eine Wunschanforderung geprüft: (*W13. Verpasste Besuche sollten aufgezeichnet werden und in der Client App in Form von einem Foto und Notifikation sichtbar sein*). Diese Funktionalität wurde aus Zeitgründen nicht vollständig implementiert. Die Anforderung wurde daher nur teilweise erfüllt. Wenn jemand an der Türe klingelt, wird auf dem mobilen Gerät eine Benachrichtigung angezeigt. Diese Benachrichtigung bleibt solange sichtbar, bis der Benutzer sie gelesen und gelöscht hat. Folgerichtig sind verpasste Besuche trotzdem ersichtlich, auch wenn sie im Web-App nicht erscheinen

Weil es sich um eine Webbapplikation handelt, kann diese Funktionalität ohne grosses Know-How von anderen beteiligten Technologien nachträglich implementiert werden.

### 8.2.2 Test 8: Die Auflösung des Videosignales wird evaluiert

Test 8 prüft zwei Anforderungen. Es handelt dabei um:

- A9: Die Kamera für das Videosignal muss eine Auflösung von mind. 1280x720 Pixel aufweisen.

- W2: Die Kamera für das Videosignal muss eine Auflösung von 1920x1080 Pixel aufweisen.

Im Nachhinein ist uns klar geworden, dass diese zwei Anforderungen und ihre dazugehörigen Tests besser definiert werden könnten.

Der Test lautet: (*Durch eine Analyse wird die effektive Auflösung des Videosignales evaluiert. Die Auflösung muss mindestens 720p (oder 1080p) aufweisen.*) Die eingesetzte Kamera weist effektiv eine Auflösung von 1920x1080 Pixel auf. Das Problem liegt bei der Digitalisierung des Videosignales. Während der Fachmodul-Phase wurde die Komplexität der Video-Encoding unterschätzt.

Demzufolge enthält unser System die Hardware, um diese Anforderungen zu erfüllen, aber nicht die Rechenkapazität.

Wie im Abschnitt 7.5.2 bereits diskutiert, wäre eine mögliche Lösung der Einsatz einer besseren CPU.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13
A1	Bestanden												
A2		Bestanden											
A3			Bestanden										
A4				Bestanden									
A5					Bestanden								
A6						Bestanden							
A7							Bestanden						
A8								Nicht bestanden					
A9									Bestanden				
A10										Bestanden			
A11											Bestanden		
A12												Bestanden	
W1											Bestanden		
W2								Nicht bestanden				Bestanden	
W3												Bestanden	



Abbildung 22: Test Traceability Matrix

## 9 Ausblick

### 9.1 Verbesserungs- und Erweiterungsmöglichkeiten

Der aktuelle Stand des Systems bietet die Standardfunktionalitäten die notwendig sind um den Prototyp in einer Testumgebung einsetzen zu können. Skalierbarkeit und Weiterentwicklung waren deshalb zwei wichtige Bedingungen die die ganze Arbeit geprägt haben.

Das Einsetzen von Webtechnologien hat dazu geführt, dass die verschiedenen Benutzerschnittstellen in Zukunft ohne grossen Aufwand und Know-How abgeändert oder erweitert werden können.

In Bereich Hardware könnten ebenso Verbesserungen vorgenommen werden. Die Komponenten wurden so ausgewählt, dass eine schnelle und einfache Implementierung ermöglicht wird. Wie im Bericht schon erwähnt, sollte für die **Aussen-sprechstellen** einen leistungsstärkeren Micro-Controller eingesetzt werden. Auch für die Kamera könnten weitere Produkte evaluiert werden, die für den Zweck besser geeinigt wären. Ein Kamerasensor mit integriertem Autofokus könnte zum Beispiel die Qualität des Bildes weiter verbessern.

Die eingesetzten Komponenten wie der **PoE**-Splitter, der Massentrennfilter und der Verstärker sind Open Source Hardware. Mit den vorhandenen Schaltplänen dieser Komponenten, könnte man eine einzelne Platine anfertigen lassen, die alle Funktionalitäten beinhaltet. Diese würde die Montage deutlich vereinfachen und gleichzeitig auch das Ausfallsrisiko senken.

### 9.2 Einsatzmöglichkeiten

Der entstandene Prototyp ist die richtige Lösung um die Machbarkeit einer solchen Digitalisierung zu demonstrieren. Die Anlage soll weiterentwickelt werden, und dieser Prototyp ist bestens geeignet um mögliche Interessenten und Investoren aufmerksam zu machen. Die kompakte Bauweise der **Türklingelanlage** und die einfachen Verkabelungstechnologien ermöglichen bei Fachmessen oder Präsentationen eine rasche Inbetriebnahme der Anlage.

## 10 Schlussfolgerung

Das, während der Bachelorarbeit, entstandene Produkt hat gezeigt, dass die Digitalisierung einer **Türklingelanlage** mit der heutigen Technologie möglich ist. Was für einen Laien auf den ersten Blick als relativ einfaches System wahrgenommen wird, hat uns bei der Entwicklung grosse Herausforderung bereitgehalten.

Der grösste Fehler aus unsere Seite war, teilweise viel zu *Endprodukt-Orientiert* zu arbeiten. Beispielsweise haben wir sehr viel Zeit für das Design der Webapplikationen investiert. Für uns war es wichtig, eine Zeitgemässe und vor allem Benutzerfreundliche Benutzeroberfläche zu entwickeln. Im Nachhinein ist uns klar gewesen, dass wir uns mehr Zeit einplanen sollten, für die Kernfunktionalitäten.

Im Gegensatz zur herkömmlichen, analogen Türklingelanlagen werden bei unserem Prototyp alle Video-Streams digitalisiert, codiert und decodiert. Diesen Schritt führt, im Gegensatz zu analogen Systemen welche in Echtzeit funktionieren, zu kleineren aber spürbaren Delays. So auch die Qualität der Videoübertragung. Nach dem aktuellen Stand der Technik und mit der für das Projekt beschaffene Hardware ist eine bidirektionale, in Echtzeit funktionierende HD-Verbindung kaum realisierbar. Die Qualität die erreicht wird, nährt sich denjenigen der Marktführer wie z.B Skype oder Facebook.

In den letzten Jahren wurden enorme Schritte im Bereich der Technologien und der Forschung gemacht und der Trend ist stets positiv. Unserer Meinung nach wird aber in wenigen Jahren in der Welt der Hausautomatisierung den Gap zwischen analoge und digitale Technologie überwunden werden.

Im Verlauf der Arbeit wurde uns klar, dass der Weg zur Digitalisierung die richtige Entscheidung war. Die digitale Welt bring unzählige Vorteile mit sich, die in der zukünftigen Hausautomation nicht mehr wegzudenken sind. So ist zum Beispiel die Interaktion der **Türklingelanlage** über ein Smartphone oder ein Tablet ein erheblicher Vorteil im Gegensatz zu analoge Systeme. Weiter ermöglicht das digitale System der Zugriff auf die Videoübertragung von aussen, was mit analoge Technologie bis jetzt kaum realisierbar war.

## 11 Anleitungen

Diese Anleitungen sind an dem zukünftigen Entwickler dieser Prototyp gerichtet. Mithilfe dieser Dokumentation und des mitgelieferten Images der Aussensprechstelle, soll ein Entwickler im Stand sein eine neu installierte Raspbian **OS** einer Aussensprechstelle/Server zu konfigurieren. Alles was konfiguriert wurde, wurde dokumentiert und in der Anleitung aufgeführt. Diese soll auch das Hinzufügen zukünftiger Funktionalitäten erleichtern.

### 11.1 Aussensprechstelle Konfigurationsanleitung

#### 11.1.1 Aktuelle Stand

Betriebssystem: Raspbian jessie with pixel

Version: April 2017

Kernel Version: 4.4

#### 11.1.2 Namen und Passwortkonzept

Hostname: DoorPi<sup>x</sup> (x= fortlaufende Nummerierung)

User: pi

Password: bachelor (Einfachheitshalber wurde dieses schwache Passwort ausgewählt. Sollte aber bei einer produktiven Inbetriebnahme zwingend geändert werden)

#### 11.1.3 Betriebssystem Installation

- Das Image von raspberry.com herunterladen und extrahieren (<https://www.raspberrypi.org/downloads/raspbian/>)
- Um die Image auf der SD Karte zu bringen benutzt man Etcher. (<https://etcher.io/>)
- Mit den Standard-Anmelddaten Anmelden. User: pi Password: raspberry

#### 11.1.4 Allgemeine Einstellungen

Das System soll auf dem neusten Stand aktualisieren werden

```
apt-get update  
sudo apt-get upgrade
```

Mit dem Terminalkommando 'sudo raspi-config' können durch eine grafische Oberfläche folgende allgemeine Einstellungen angepasst werden:

- Unter 'Interfacing Options' muss die SSH Server aktiviert werden.
- Hostname gemäss Namenskonzept anpassen
- Neue Passwort für den Pi Benutzer gemäss Password Konzept setzen.
- Zum Schluss sollen auch die Zeitzone, das Land und das Tastaturlayout angepasst werden.

### 11.1.5 Bildschirm Konfiguration

Die Displaytreiber von waveshare.com herunterladen und auf die SD Karte in Root Directory speichern. ([http://www.waveshare.com/wiki/4inch\\_HDMI\\_LCD](http://www.waveshare.com/wiki/4inch_HDMI_LCD)) Mit folgenden Bash-Kommandos wird der Treiber installiert:

```
tar xzvf /boot/LCD-show-YYMMDD.tar.gz
cd LCD-show/
chmod +x LCD4-800x480-show
./LCD4-800x480-show
```

Nachdem dass der Bildschirmtreiber installiert wurde, müssen die Einstellungen für den Bildschirm angepasst werden. Folgende Code-Zeilen müssen am Ende der 'config.txt' Datei, die sich in der root-directory befindet, hinzugefügt werden [1].

```
hdmi_group=2
hdmi_mode=87
hdmi_cvt 480 800 60 6 0 0 0
dtoverlay=ads7846,cs=1,penirq=25,penirq_pull
=2,
speed=50000,keep_vref_on=0,swapxy=0,pmax=255,
xohms=150,xmin=200,xmax=3900,ymin=200,ymax
=3900
display_rotate=3
```

### 11.1.6 Browser Kiosk-mode

Als erstes wir das Unclutter-Tool installiert um den Mausepfeil auszublenden.

```
sudo apt-get install unclutter
```

Die Kiosk-Mode-Einstellungen werden in der config Datei (/home/pi/.config/lxsession/LXDE-pi/autostart) wie folgt angepasst

```
# Chromium auto start in kiosk mode
# path: /home/pi/.config/lxsession/LXDE-pi/
#       autostart
@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
#@xscreensaver -no-splash
@point-rpi
@xset s off
@xset s noblank
@xset -dpms
@chromium-browser --noerrdialogs --kiosk --
incognito https://172.16.111.99/server
```

### 11.1.7 Aussensprechstelle Initialisierung

Im Homeverzeichnis unter .config/autostart wird die Datei Aussensprechstelle.desktop erstellt.

```
touch /home/pi/Aussensprechstelle/Startup/
AussensprechstelleLauncher.sh
```

Inhalt des Script:

```
#!/bin/bash
# This script executes the needed commands on
# startup to initialize the
# Aussensprechstelle
#/home/pi/Aussensprechstelle/Startup/
# AussensprechstelleLauncher.sh
#
# Activates the Camera Driver (Safe mode
# because of the chrome resolution bug)
sudo modprobe bcm2835-v4l2
gst_v4l2src_is_broken=1
#
# Clears the old TasterController PID of the
# process (In case of system shutdown)
```

```

file="/var/run/TasterController.pid"
if [ -f $file ] ; then
    rm $file
fi
#
# Starts the TasterController
sudo java -jar /home/pi/Aussensprechstelle/
TasterController/TasterController.jar &
#
# Creates the PID for the taster controller
sudo echo $! > /var/run/TasterController.pid
#
# Starts the watchdog service
sudo service watchdog start

```

### 11.1.8 Taster Controller

Der Tastencontroller, der für den Key Mapping zuständig ist, wird vom oben gezeigten AussensprechstelleLauncher.sh unter /home/pi/Aussensprechstelle/TasterController/TasterController.jar gestartet. Die kompilierte Jar-Artefakt muss also dorthin kopiert werden.

Folgende GPIO Pins werden von den 3 Tasten benötigt um die Aussensprechstelle zu steuern:

- GPIO17(16) simuliert den Tastendruck J «Links navigieren»
- GPIO27(20) simuliert den Tastendruck K «Anrufen»
- GPIO22(21) simuliert den Tastendruck L «Rechts navigieren»

### 11.1.9 Speaker Controller Service

Als Erstes muss der mitgelieferte Jar Artefakt SpeakerController.jar unter folgenden Pfad kopiert werden:

```
/home/door/Aussensprechstelle/
SpeakerController/SpeakerController.jar
```

Um den Speaker-Controller als Service unter Unix laufen zu lassen, muss unter /etc/init.d/ das Speaker-Controller-Script erzeugt werden. Der Inhalt des Scripts

wird mit dem Projekt mitgeliefert. Um es ausführbar zu machen, muss noch die «execute» Berechtigung gegeben werden.

```
touch /etc/init.d/speakerController
chmod +x /etc/init.d/speakerController
```

Damit der SpeakerController-Service auch automatisch beim Systemstart ausgeführt wird, muss noch folgendes Kommando ausgeführt werden:

```
sudo update-rc.d speakerController defaults
```

Der Speaker Controller kann nun mit folgenden Befehlen gestartet und gestoppt werden

```
sudo service speakerController start
sudo service speakerController stop
sudo service speakerController restart
sudo service speakerController status
```

### **11.1.10 Watchdog/Watchdog deamon**

Um die von der Aussensprechstelle benötigte Dienste zu überwachen, wird ein Watchdog verwendet. Raspberry Pi hat ein «stand-alone» Hardware Watchdog die ein Autostart durchführt sobald eine der Dienste oder den OS still steht. Mit folgenden Kommandos wird der Watchdog installiert:

```
sudo modprobe bcm2835-wdt
sudo apt-get install watchdog chkconfig
sudo chkconfig watchdog on
sudo /etc/init.d/watchdog start
```

Damit die SpeakerController und die TasterController vom Watchdog überwacht werden, muss unter /etc/watchdog.conf die Konfigurationsdatei abgeändert werden. Der Inhalt der Konfigurationsdatei wird mit dem Projekt mitgeliefert.

## **11.2 Server Konfigurationsanleitung**

### **11.2.1 Aktuelle Stand**

Betriebssystem: Raspbian jessie with pixel

Version: April 2017

Kernel Version: 4.4

### 11.2.2 Namen und Passwortkonzept

Hostname: SrvPi<sup>x</sup>xx (x= fortlaufende Nummerierung)

User: pi

Password: raspberry (Default Password)

### 11.2.3 Software Installation

Nun werden die benötigten Dienste und Tools installiert, die vom Server benötigt werden.

- Installation der Webserver. (PHP, Nginx, MySQL, Java SDK, Composer, Utils)

```
apt-get install nginx
sudo apt-get install mysql-server
apt-get install php5-fpm php5-mysql
sudo apt-get install mysql-server mysql-client
sudo apt-get install oracle-java8-jdk
sudo apt-get install curl php5-cli git
curl -sS https://getcomposer.org/installer |
    sudo php -- --install-dir=/usr/local/bin --
    filename=composer
```

### 11.2.4 Erstellung SSL Zertifikate

Bevor die Webapplikationen installiert werden können, müssen die Zertifikate generiert werden (Self-Signed).

- Erstellung SSL Zertifikat für den Client Webapplikation. Als Hostname wird hier als Beispiel intercom.app verwendet. Zuerst muss eine Konfigurationsdatei (v3.ext) mit folgendem Inhalt generiert werden:

```
authorityKeyIdentifier=keyid , issuer
basicConstraints=CA:TRUE
keyUsage = digitalSignature , nonRepudiation ,
keyEncipherment , dataEncipherment
subjectAltName = @alt_names
```

```
[ alt_names ]
DNS.1 = intercom.app
```

Falls ein IP als hostname verwendet wird, kann man @alt\_names mit IP:192.168.0.18 ersetzen.

- Nun müssen folgende Befehle eingegeben werden. Wenn gefragt, muss der Hostname oder IP als Common Name (CN) eingegeben werden. Es kann immer das gleiche Passwort verwendet werden und muss dem Keystore-Password des Signaling-Servers entsprechen.

```
sudo openssl genrsa -des3 -out rootCA.key 2048

sudo openssl req -x509 -new -nodes -key rootCA.
key -sha256 -days 1024 -out rootCA.pem

sudo openssl req -new -sha256 -nodes -out server
.csr -newkey rsa:2048 -keyout server.key

sudo openssl x509 -req -in server.csr -CA rootCA
.pem -CAkey rootCA.key -CAcreateserial -out
server.crt -days 500 -sha256 -extfile v3.ext

sudo openssl pkcs12 -export -in server.crt -
inkey server.key -out cert.p12

sudo keytool -importkeystore -srckeystore cert.
p12 -srcstoretype PKCS12 -destkeystore
keystore.jks -deststoretype JKS

sudo openssl x509 -inform PEM -outform DER -in
server.crt -out phone.der.crt
```

Somit wurden diverse Dateien generiert. Die folgenden werden später gebraucht.

- server.cert und server.key -> SSL Zertifikate für Apache2
- rootCA.pem -> Root CA. Das muss in den Client-Browser importiert werden, damit die Clients den Server als vertraulich erkennen.

- phone.der.crt -> Root CA für Mobilegeräte. Bei Mobilegeräte kann es per E-Mail verschickt werden und dann in die Systemeinstellungen installiert werden.
- keystore.jks -> Das muss später in das selbe Verzeichnis kopiert werden, wo der SignalingServer installiert wird.

### 11.2.5 Konfiguration von Nginx

Vor dem Deploy der Webapplikationen muss der Webserver noch konfiguriert werden.

- In der Datei `/etc/php5/fpm/php.ini` muss die folgende Zeile auskommentiert und editiert werden:

```
cgi.fix_pathinfo=0
```

- Nun muss Nginx so konfiguriert werden, dass PHP als compiler verwendet wird. Die Datei `/etc/nginx/sites-available/default` muss editiert werden. Die Stellen die angepasst werden müssen sind rot markiert.
- Nun müssen die zwei VirtualHosts für die zwei WebApps konfiguriert werden. Diese werden unter verschiedene Ports laufen. Dafür müssen zwei Konfigurationsdateien unter `/etc/nginx/sites-available` erstellt werden. Bsp: intercom.app und management.app. Der Inhalt muss wie folgt aussehen. Die Stellen, die für die beiden Webapplikationen unterschiedlich sein müssen, sind rot markiert. Der Path zu den SSL-Zertifikate muss auch angepasst werden.

```
# Default server configuration
server {
# SSL configuration
listen 443 ssl; # 444 for the second host
listen [::]:443 ssl;

ssl_certificate /path/to/the/certificate/server.crt;
ssl_certificate_key /path/to/the/certificate/server.key;
```

```

root /var/www/intercom/public; # /management/public for
the second host

# Add index.php to the list if you are using PHP
index index.php index.html index.htm index.nginx-debian.
html;

server_name __;

location / {
# First attempt to serve request as file , then
# as directory , then fall back to displaying a 404.
try_files $uri $uri/ /index.php?$query_string;
}

location ~ \.php$ {
include snippets/fastcgi-php.conf;
fastcgi_pass unix:/var/run/php5-fpm.sock;
}
location ~ /\.ht {
deny all;
}
}

```

- Zum abschliessen noch die folgenden Befehle eingeben:

```

sudo ln -s /etc/nginx/sites-available/intercom.app /etc/
nginx/sites-enabled/
sudo ln -s /etc/nginx/sites-available/management.app /etc
/nginx/sites-enabled/
sudo service nginx reload
sudo service nginx restart
sudo service php5-fpm restart
sudo reboot

```

### 11.2.6 Deploy Webapplikationen

Vor dem Deploy der Webapplikationen muss der Webserver noch konfiguriert werden.

- Die beide Webapplikationen müssen zuerst auf dem Server in die passenden Verzeichnissen kopiert werden.

```
/var/www/management  
/var/www/intercom
```

- Rechte anpassen

```
sudo chmod -R 775 /var/www  
sudo chmod -R 777 /var/www/management/storage  
sudo chmod -R 775 /var/www/intercom/storage  
sudo chgrp -R www-data /var/www/
```

- MySQL Database erstellen, dann Anmelddaten und Database Name in der Datei: .env eingeben. (Falls .env nicht vorhanden: cp .env.example .env)
- Webapplikation installieren: (Diese Befehle müssen in der Root Dir jeder Webapp eingegeben werden).

```
composer install  
php artisan migrate (MNGMT Tool Only)  
php artisan key:generate
```

Die zwei Webapps müssten nun unter die ports 443 und 444 aufrufbar sein.

### 11.2.7 Deploy Dienste und Services

- Zuerst die benötigten Pfade erstellen

```
mkdir /home/pi/server/signalingServer  
mkdir /home/pi/server/relayController
```

- Die beide kompilierte JARs in den entsprechenden Verzeichnissen kopieren. Die kompilierten JARs sind im jeweiligen Projektverzeichniss unter /deploy zu finden.

- Nun muss noch für den SignalingServer das vorher erstellte keystore.jks kopiert werden. Der Keystore muss sich im gleichen Verzeichnis wie der Signaling Server befinden.
- Für beide Dienste muss noch der **Autostart\_Skript** unter /etc/init.d/ kopiert werden. Die Skripte sind im Script-Verzeichnis gespeichert. Auch diese Skripte müssen ausführbar sein. Folgende Befehle müssen noch eingegeben werden:

```
sudo chmod +x /etc/init.d/signalingServer
sudo chmod +x /etc/init.d/relayController
sudo update-rc.d signalingServer defaults
sudo update-rc.d relayController defaults
```

## 11.3 Installation Client-Webapplikation

Es wird nun beschrieben, wie die Client-Webapplikation auf ein mobiles Gerät installiert werden kann. Momentan wurde die Applikation hauptsächlich auf Android Smartphones mit Google Chrome getestet. Da es sich um eine Webapplikation handelt, ist das Betriebssystem von kleinerer Bedeutung. Wichtig ist, dass WebRTC vom Browser unterstützt wird.

Die folgende Anleitung gilt für Android 7.0 Nougat. Die Installation wird auf andere Mobile Geräte auf jeden Fall nur leicht abweichen.

### 11.3.1 Installationsschritte

#### Schritt 1: Installation der Zertifikate

- Die vorher generierte Mobile Zertifikat-Datei (siehe Abschnitt 11.2.4) auf das Smartphone kopieren. Das kann via USB oder E-Mail erfolgen. Wichtig ist, dass die Datei auf dem lokalen Speicher des Mobilgeräts vorhanden ist.
- Unter Options -> Security auf *Install from phone storage* klicken und die Zertifikat-Datei auswählen.
- Gerät neu starten.

#### Schritt 2: Einbindung am Homescreen

Nun kann die Webapplikation bereits verwendet werden. Es reicht die folgende Adresse einzugeben: <https://SERVER/client?id=1>

\* SERVER -> Die IP Adresse oder Hostname des Servers.

\*\* id=1 -> Diese ID definiert, welcher Einwohner sich gerade anmeldet. Welcher Einwohner zu welcher ID gehört, wird in dem Management Tool definiert. Wie in den Abschnitt 7.5.1 bereits erwähnt, ist dies keine endgültige Lösung und nur für Vorführzwecke gedacht.

Die Webapplikation besitzt ein *Manifest-File*, welcher die Applikation als Standalone App konfigurieren kann. Um das zu aktivieren, muss die Webseite zur Homescreen hinzugefügt werden. Um das zu machen, auf das Overflow-Menü (drei kleine Punkte oben rechts) tippen und *Zum Startbildschirm hinzufügen* auswählen.

Nun ist die Webapplikation als Standalone App verfügbar und kann somit auch in Fullscreen benutzt werden.

## 12 Anhang

### 12.1 Messungsresultate

```
Switch#sh interfaces g0/43
GigabitEthernet0/43 is up, line protocol is up (
    connected)
Hardware is Gigabit Ethernet, address is 0023.05e2.c82b
    (bia 0023.05e2.c82b)
MTU 1500 bytes, BW 100000 Kbit/sec, DLY 100 usec,
    reliability 255/255, txload 1/255, rxload 1/255
Encapsulation ARPA, loopback not set
Keepalive set (10 sec)
Full-duplex, 100Mb/s, media type is 10/100/1000BaseTX
input flow-control is off, output flow-control is
    unsupported
ARP type: ARPA, ARP Timeout 04:00:00
Last input never, output 00:00:00, output hang never
Last clearing of "show interface" counters never
Input queue: 0/75/0/0 (size/max/drops/flushes); Total
    output drops: 0
```

```
Queueing strategy: fifo
Output queue: 0/40 (size/max)
5 minute input rate 507000 bits/sec, 108 packets/sec
5 minute output rate 44000 bits/sec, 70 packets/sec
51357 packets input, 33613725 bytes, 0 no buffer
Received 130 broadcasts (82 multicasts)
0 runts, 0 giants, 0 throttles
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
0 watchdog, 82 multicast, 0 pause input
0 input packets with dribble condition detected
32616 packets output, 6365458 bytes, 0 underruns
0 output errors, 0 collisions, 1 interface resets
0 unknown protocol drops
0 babbles, 0 late collision, 0 deferred
0 lost carrier, 0 no carrier, 0 pause output
0 output buffer failures, 0 output buffers swapped out
```

## 12.2 Präsentation Prototyp

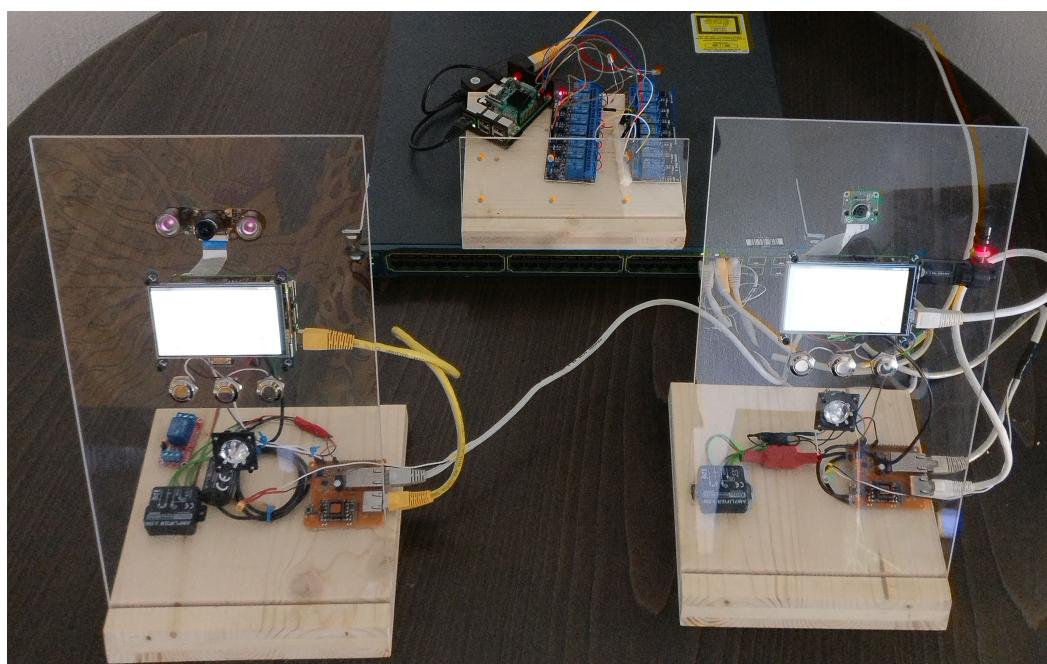


Abbildung 23: Präsentation Prototyp

## **Abbildungsverzeichnis**

1	Hybrides Vorgehensmodell . . . . .	3
2	Projektplanung Meilensteine . . . . .	3
3	Projektplanung . . . . .	4
4	Bitbucket ist ein Filehosting und ein Dienst für die Versionskontrolle von Softwareprojekte . . . . .	4
5	Risikomatrix . . . . .	5
6	Bewertung Projekt Risiken . . . . .	6
7	Bewertung Technische Risiken . . . . .	6
8	Projekt Massnahmen . . . . .	7
9	Analoge Türsprechanlage mit In-House Display . . . . .	8
10	Telecom Behnkle MyIntercom . . . . .	9
11	Hardware Ecosystem . . . . .	12
12	Catalyst Pinouts . . . . .	14
13	EthernetPinbelegung . . . . .	15
14	EthernetPinbelegung . . . . .	16
15	EthernetPinbelegung . . . . .	16
16	Software Ecosystem . . . . .	20
17	Der Signaling Prozess . . . . .	23
18	STUN Server . . . . .	24
19	Design der Client-Webapp . . . . .	25
20	Design der Client-Webapp . . . . .	26
21	Design der Management tool . . . . .	28
22	Test-Traceability Matrix . . . . .	35
23	Prototyp Foto . . . . .	50

## **Tabellenverzeichnis**

1	Server <b>HW</b> Komponenten . . . . .	13
2	Aussensprechstelle <b>HW</b> Komponenten . . . . .	13
3	PIN-Zuweisung zwischen den Server und die Relais Module . . . .	17
4	PIN-Zuweisung zwischen den Raspberry PI und die Schalter . . . .	17
5	Verwendete Raspberry Pi im Vergleich mit Banana Pi als Alternative	27

## Glossar

**API** Application programming interface. 16

**Aussensprechstelle** Mikrocontroller mit verschiedeneModulen die an den Eingangstüre installiert wird. 6–10, 13, 15–25

**Client Applikation** Die Web Applikation welches auf dem Smartphone oder Tablet des Bewohner ausgeführt wird. 13, 15–17

**CSS** Cascading Style Sheets. 13

**DSL** Digital Subscriber Line. 4

**GPIO** General purpose input/output. 10

**GUI** Graphical user interface. 15

**H.264** Standard zur Videokompression. 21

**HTML** Hypertext Markup Language. 13, 16

**HTTPS** Hypertext Transfer Protocol Secure. 15

**HW** Hardware. 7, 41

**ICE** Interactive Connectivity Establishment. 16, 17

**IP** Internet Protocol. 4, 15, 16

**LTE** Long Term Evolution. 4

**NAT** Network address translation. 17

**OS** Operating system. 21

**P2P** Peer to Peer. 16

**PHP** Hypertext Preprocessor. 13, 14, 18

**PoE** Power over Ethernet. 7, 8, 10

**SIP** Session Initiation Protocol. 5, 16

**STUN** Session Traversal Utilities. 17, 18

**TLS** Transport Layer Security. 14, 15

**Türklingelanlage** Gesamtheit der Komponenten die denn Zusammen den Endprodukt darstellen. I, 1, 4–6, 40

**USB** Universal Serial Bus. 12

**VoIP** Voice over IP. 16

**WebRTC** Web Real-Time Communication. 5, 8, 15, 16, 20

## **Eidesstattliche Erklärung**

Die Verfasser dieser Bachelorarbeit, Federico Cramer und Geo Bontognali, bestätigen, dass sie die Arbeit selbstständig und nur unter Benützung der angeführten Quellen und Hilfsmittel angefertigt haben. Sämtliche Entlehnungen sind durch Quellenangaben festgehalten.

Ort, Datum

Geo Bontognali

Ort, Datum

Federico Cramer



## Literatur

- [1] *4 inch LCD configuration.* URL: [http://www.waveshare.com/wiki/4inch\\_HDMI\\_LCD](http://www.waveshare.com/wiki/4inch_HDMI_LCD) (besucht am 08.08.2017).
- [2] Sam Dutton Published 4th. *WebRTC STUN, TURN and signaling.* URL: <https://www.html5rocks.com/en/tutorials/webrtc/infrastructure/> (besucht am 23.02.2017).
- [3] *Adding Push Notifications to a Web App.* Google Developers. URL: <https://developers.google.com/web/fundamentals/getting-started/codelabs/push-notifications/> (besucht am 20.06.2017).
- [4] *Building a signaling server in Java.* ISBN: 978-1-78328-445-0. URL: <https://www.packtpub.com/mapt/book/Application-Development/9781783284450/1/ch01lvl1sec10/Building%20a%20signaling%20server%20in%20Java> (besucht am 23.02.2017).
- [5] *Create a SSL Certificate on Apache for Ubuntu.* URL: <https://www.digitalocean.com/community/tutorials/how-to-create-a-ssl-certificate-on-apache-for-ubuntu-14-04> (besucht am 10.04.2017).
- [6] *Debian Gesellschaftsvertrag.* URL: [https://www.debian.org/social\\_contract.de.html](https://www.debian.org/social_contract.de.html) (besucht am 06.07.2017).
- [7] *Embedded Linux JVM Debugger.* URL: <https://plugins.jetbrains.com/idea/plugin/7738-embedded-linux-jvm-debugger-raspberry-pi-beaglebone-black-intel-galileo-ii-and-several-other-iot-devices-> (besucht am 20.02.2017).
- [8] ericlaw. *Chrome Deprecates Subject CN Matching.* text/plain. 10. März 2017. URL: <https://textslashplain.com/2017/03/10/chrome-deprecates-subject-cn-matching/> (besucht am 01.06.2017).
- [9] Geo Bontognali Federico Cramer. *Fachmodul - Türklingelanlage mit Standardkomponenten.* 1. Juni 2017.
- [10] *Fixing Chrome 58+ openssl using self signed certificates.* URL: [https://alexanderzeitler.com/articles/Fixing-Chrome-missing\\_subjectAltName-selfsigned-cert-openssl/](https://alexanderzeitler.com/articles/Fixing-Chrome-missing_subjectAltName-selfsigned-cert-openssl/) (besucht am 01.06.2017).
- [11] Wolfram Gieseke. *Hardware Watchdog.* URL: <http://www.gieseke-buch.de/raspberrypi/eingebauten-hardware-watchdog-zur-ueberwachung-nutzen> (besucht am 20.03.2017).

- [12] *How are SSL certificate server names resolved*. URL: <https://stackoverflow.com/questions/8443081/how-are-ssl-certificate-server-names-resolved-can-i-add-alternative-names-using/8444863> (besucht am 19.06.2017).
- [13] *Interactive Connectivity Establishment*. In: *Wikipedia*. 2. Aug. 2016. URL: [https://en.wikipedia.org/w/index.php?title=Interactive\\_Connectivity\\_Establishment&oldid=732705039](https://en.wikipedia.org/w/index.php?title=Interactive_Connectivity_Establishment&oldid=732705039) (besucht am 23.02.2017).
- [14] *Java SE 7 Class SSLContext*. URL: <http://docs.oracle.com/javase/7/docs/api/javax/net/ssl/SSLContext.html> (besucht am 28.02.2017).
- [15] Daniel L. *Create and import Self Signed Certificate to Android*. URL: <https://aboutssl.org/how-to-create-and-import-self-signed-certificate-to-android-device/> (besucht am 19.06.2017).
- [16] *Laravel Notification Channels*. URL: <http://laravel-notification-channels.com/webpush/> (besucht am 21.06.2017).
- [17] *Linux Java Service Wrapper*. URL: <http://www.jcgonzalez.com/linux-java-service-wrapper-example> (besucht am 23.02.2017).
- [18] *Linux Watchdog*. URL: [http://www.sat.dundee.ac.uk/psc/watchdog/watchdog-configure.html#Process\\_Monitoring\\_by\\_PID\\_File](http://www.sat.dundee.ac.uk/psc/watchdog/watchdog-configure.html#Process_Monitoring_by_PID_File) (besucht am 20.03.2017).
- [19] *LSB Init Scripts Debian Wiki*. URL: <https://wiki.debian.org/LSBInitScripts> (besucht am 01.03.2017).
- [20] *Power over Ethernet*. URL: <https://www.elektronik-kompendium.de/sites/net/0807021.htm> (besucht am 07.08.2017).
- [21] *Push Notifications in your PhoneGap Application*. URL: <http://devgirl.org/2013/07/17/tutorial-implement-push-notifications-in-your-phonegap-application/> (besucht am 20.06.2017).
- [22] *Raspberry Codecs*. URL: <https://www.raspberrypi.org/blog/libraries-codecs-oss/>.
- [23] *Raspberry Pi System Logging*. URL: <http://blog.scphilips.com/posts/2015/05/raspberry-pi-system-logging-and-loggly/> (besucht am 15.03.2017).
- [24] *Raspberry Poor audio quality on analog output*. URL: <https://www.raspberrypi.org/forums/viewtopic.php?f=38&t=37038> (besucht am 07.06.2017).

- [25] Deven Rathore. *Laravel Push notifications to mobile devices*. 10. Okt. 2015.  
URL: <https://www.dunebook.com/send-push-notifications-to-mobile-devices-with-laravel/> (besucht am 21.06.2017).
- [26] *WebRTC Low resolution on Chromium*. URL: <https://www.raspberrypi.org/forums/viewtopic.php?f=43&t=159941> (besucht am 15.03.2017).
- [27] *Working with Java Keystores*. URL: <https://www.digitalocean.com/community/tutorials/java-keytool-essentials-working-with-java-keystores> (besucht am 28.02.2017).