NTB
Interstaatliche Hochschule
für Technik Buchs

FHO Fachhochschule Ostschweiz

Türklingelanlage mit Standardkomponenten

Federico Crameri, Geo Bontognali

Bachelorarbeit

Studiengang:	Systemtechnik
Profil:	Informations- und Kommunikationssysteme
Referent:	Prof. Dr. Hauser-Ehninger Ulrich, MSc in Electronic Engineering
Korreferent:	Toggenburger Lukas, Master of Science FHO in Engineering

Kurzfassung

In der heutigen Welt entwickelt sich alles mit einer erstaunlichen Geschwindigkeit. Die Gebäudetechnik ist hier keine Ausnahme. Darunter zählen unterschiedliche Bereiche und Systeme. Eins davon ist die Türklingelanlage.

Im Rahmen dieser Bachelorarbeit wurde ein Türklingelanlage-Prototyp mit Standardkomponenten entwickelt. Wichtig für unsere Arbeit war zu überprüfen, inwiefern die heutige Standard- und OpenSource -komponenten für ein solches System geeignet sind. Während der Entwurf-Phase wurden unterschiedliche Anforderungen definiert. Die Kommunikation über die Türklingelanlage muss über Video und Audio Signale erfolgen. Die Gegensprechanlage wurde komplett auf die digitale Ebene realisiert. In einer Welt wo jeder Internetzugriff im Griffweite immer hat, war eine digitale Lösung, der einzige richtige Weg. Das brachte unterschiedliche Herausforderungen mit sich.

Das Endresultat ist eine digitale, flexible und zeitgemässe Türklingelanlage welche mehrere Eingangstüren Steuern kann und die Bedienung mit den Herkömmlichen Handys ermöglicht.

Abstract

In a world where everything is moving forward at the speed of light, home building technology is no exception. There are plenty of systems needed around building a house. 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 comes on the table, also a lot of different complications and challenges comes with it 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

1 Einführung

1.1 Problemstellung

Heutzutage liefern diverse Hersteller verschiedene Lösungen für das Türglockensystem. Diese sind meistens Komplettsysteme, die nicht nur das einfache Klingel ermöglichen, sondern auch Zusatzfunktionen wie das Videostreaming anbieten. Diese Systeme sind aber meistens proprietär und werden 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 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/Technologien, eine kostengünstige und funktionale Türklingelanlage zu entwickeln.
Um den Kostenfaktor zu berücksichtigen, soll die Anlage auf schon vorhandene Technologie/Hardware basieren. Somit fallen die hohen Kosten für die Beschaffung proprietärer Hardware weg.
In der 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 zu sehen ist. Im Rahmen einer Bachelor Arbeit, in der die Anforderungen und Analysen schon im Vorhinein im Fachmodul definiert worden sind, eignet sich am bestens ein lineares V-Modell. Ein solcher Prozess ist sehr schlank, übersichtlich und geeignet für die Grösse des Projekts.
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 erlaubt, trotz der klar definierten Anforderungen, während der Entwurf- und der Implementierungsphase ein agiles Vorgehen mit der Kunde 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/Entscheidungen die in den ersten Phasen getroffen worden sind.

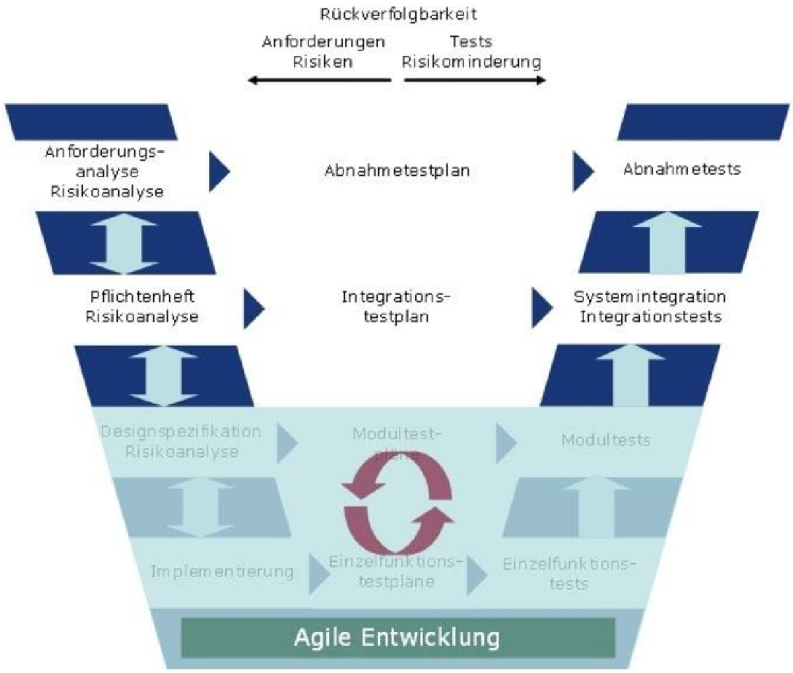

Abbildung 1: Hybrides Vorgehensmodell

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 Hardware Komponenten, die mittlerweile schon bestellt wurden, getestet und zusammengebaut. Die nächste zwei Meilensteine sind Software-Ready Meilensteine. Die Software Programmierung wurde in zwei Teile geteilt.
Bei Part 1 geht es um die Skripts die Serverseitig kleine Aufgaben übernehmen. Part 2 ist der grösste Programmierung teil. Da werden die Webapplikationen entwickelt, die auf die Aussensprechstellen und auf die Mobile Geräte der Bewohner ausgeführt werden sollen.
Die letzte Phase ist für die Optimierung und Reserve gedacht.

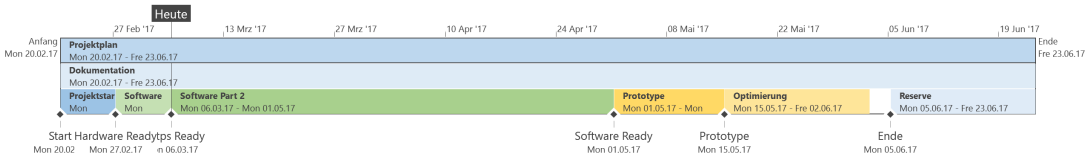

Abbildung 2: Zeitplanung mit Meilensteine

Vorgangsname	Dauer	Anfang	Ende
Dokumentation	**90 Tage**	**Mon 20.02.17**	**Fre 23.06.17**
Dokumentation	18 Wochen	Mon 20.02.17	Fre 23.06.17
Start	0 Std.	Mon 20.02.17	Mon 20.02.17
Projektstart	**5 Tage**	**Mon 20.02.17**	**Mon 27.02.17**
Projektplanung	1 Woche	Mon 20.02.17	Fre 24.02.17
Minor HW & SW Tests	1 Woche	Mon 20.02.17	Fre 24.02.17
Patchkabel Tests	1 Woche	Mon 20.02.17	Fre 24.02.17
Autostart configurieren	1 Woche	Mon 20.02.17	Fre 24.02.17
Hardware Ready	0 Std.	Mon 27.02.17	Mon 27.02.17
Software Part 1	**5 Tage**	**Mon 27.02.17**	**Mon 06.03.17**
Skript: Keymapping für die Schalter	1 Woche	Mon 27.02.17	Fre 03.03.17
Skript: Lautsprecher ON/OFF	1 Woche	Mon 27.02.17	Fre 03.03.17
Skript: Watchdog	1 Woche	Mon 27.02.17	Fre 03.03.17
Skript: Serverdienst für die Relaysteueru	1 Woche	Mon 27.02.17	Fre 03.03.17
Skritps Ready	0 Tage	Mon 06.03.17	Mon 06.03.17
Software Part 2	**40 Tage**	**Mon 06.03.17**	**Mon 01.05.17**
Laravel App für die Türsprechanlage	4 Wochen	Mon 06.03.17	Fre 31.03.17
Laravel App für die Clients	4 Wochen	Mon 06.03.17	Fre 31.03.17
Laravel App für den Managment	5 Wochen	Mon 27.03.17	Fre 28.04.17
Software Ready	0 Tage	Mon 01.05.17	Mon 01.05.17
Prototype	**10 Tage**	**Mon 01.05.17**	**Mon 15.05.17**
Prototype	1 Woche	Mon 01.05.17	Fre 05.05.17
Tests	1 Woche	Mon 08.05.17	Fre 12.05.17
Prototype	0 Tage	Mon 15.05.17	Mon 15.05.17
Optimierung	**15 Tage**	**Mon 15.05.17**	**Fre 02.06.17**
Optimierung	3 Wochen	Mon 15.05.17	Fre 02.06.17
Tests	3 Wochen	Mon 15.05.17	Fre 02.06.17
Ende	0 Tage	Mon 05.06.17	Mon 05.06.17
Reserve	3 Wochen	Mon 05.06.17	Fre 23.06.17

Abbildung 3: Projektplanung

3 Aktueller Stand der Technik

Eine Türklingelanlage, welche Audio und Video 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 bei solche Lösungen aus diesem Grund ausgeschlossen.

Abbildung 4: Analoge Türsprechanlage mit In-House Display

In den letzten Jahren sind die ersten Digitale Lösungen mit IP Videoübertragung auf dem Markt gekommen. Die Digitalisierung in diesem Bereich hat es die gigantische Schritten im Bereich der Miniaturisierung und die immer schnellere Internet Zugänge (xDSL, LTE, usw) zu verdanken.

3.1 Die Herausforderungen der Digitalisierung

Die Digitalisierung bringt nicht nur Vorteile mit sich. Besonders bei der Video und Audioübertragung. Während eine Analoge Videoübertragung ziemlich mühelos erfolgt muss im Fall eine Digitale Lösung das Video zuerst kodiert und dann dekodiert werden.
Die heutige Kodierung-Algorithmen ermöglichen eine ziemlich schnelle Dekodierung. Mittlerweile hat jeder Smartphone genug Leistung um ein Full-DSL Videostream vom Youtube oder Netflix in real-time zu dekodieren. Auf die andere Seite ist die Kodierung ein sehr rechenintensiven Prozess und benötigt sehr viel Leistung.
Jeder der schon mal mit Video-Editing zu tun hatte, weisst, wie viel Zeit die Exportierung eines Video dauern kann.

Die grösste Herausforderung für die real-time Digitale Video/Audio Kommunikation besteht also darin, die Kodierung und Dekodierung der Audio und Video Signal im vernünftigen Zeit durchzuführen.

3.2 Marktsituation

Der Hauptziel dieses Bachelorarbeit ist, eine Kostengünstige Lösung für eine digitale, flexible und skalierbare Türklingelanlage. Tatsächlich ist es so, dass die bestehende Lösungen sehr teuer sind. Viele Produkte basieren auf Drittanbieter, SIP Gateways oder andere Elemente die Zusatzkosten verursachen. Das möchten wir alles vermeiden.

Abbildung 5: Telecom Behnkle MyIntercom

Eine der günstigsten Produkte den wir finden konnten ist das *"MyIntercom"* von Telecom Behnkle (siehe Abb. 5). Diese Türklingelanlage ist ziemlich flexibel und bietet die Möglichkeit, mehrere Türen anzuschliessen. Der Preis liegt hier bei zirka 1'600.- CHF pro Türe bei dem Basic-Modell.

Dank der Aufschwung von Open-Source Hardware wie das Raspberry PI und Real Time Communication Protokolle wie WebRTC muss es möglich sein, kostengünstigere Lösungen zu erarbeiten. Bei den folgenden Kapiteln geht es nun um die effektive Realisierung einem Prototyp, welches die oben genannte Problemen adressiert.

4 Lösungskonzept

Die Abb. 6 bietet ein Überblick über den verschiedenen Hardware-Komponenten, die für den Türklingelanlage notwendig sind.
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.
Das zweite Begriff ist die Aussensprechstelle. Wie in der Abb. 6 ersichtlich handelt sich hauptsächlich um ein Mikrocontroller mit verschiedene Modulen die an den Eingangstüre installiert wird.
Räumlich von der Aussensprechstelle getrennt befindet sich der Server. Diese besteht aus ein Mikrocontroller, die als Server im Einsatz ist, ein Switch die dazu dient die Aussensprechstelle mit Strom und Datenverbindung zu versorgen und die Relais welche den Türöffner und Glocken betätigen.

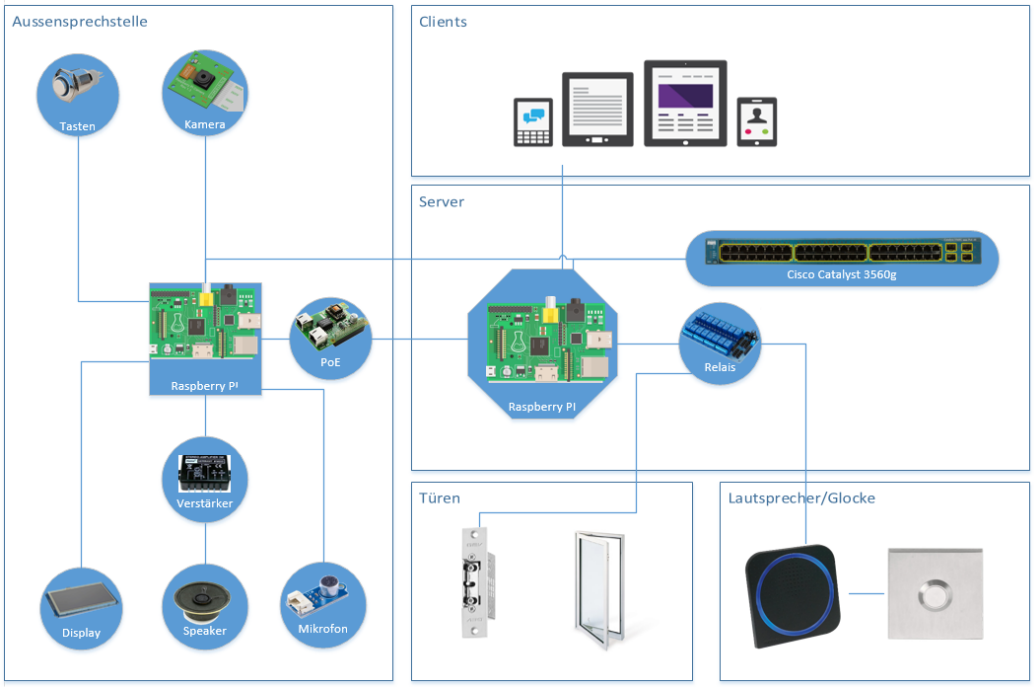

Abbildung 6: Hardware Ecosystem

Das System wird nicht nur aus Hardware bestehen, sondern auch aus viel Software-Elemente die zusammen arbeiten werden. Als erstes, wie in der Projektplanung bereits definiert, wird die Hardware Seite der Lösung realisiert. Sobald alle Hardware Komponenten getestet und auf Kompatibilität geprüft sein werden, wird die Programmierung stattfinden.

Anzahl	Komponent	Preis
1	Raspberry Pi 3 Model B	50.-
1	Raspberry Gehäuse und Netzteil	25.-
2	8-Kanal Relais Modul	15.-
1	*Kleinmaterial*	15.-
Total		**140.-**

Tabelle 1: Server HW Komponenten

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

Tabelle 2: Aussensprechstelle HW Komponenten

5 Hardware

5.1 Komponenten

Das System wird Hardwareseitig hauptsächlich in zwei Teile unterteilt. Der Server und die Aussensprechstelle.
Die Tabelle 1 und die Tabelle 2 zeigen die benötigten HW-Komponenten, welchen an die jeweilige Stellen benötigt werden.
Um den Überblick über die Kosten aller Hardware-Komponenten zu behalten, sind hier auch die Preisen aufgelistet. Wichtig hier ist sicherzustellen, dass die gesamten Hardwarekosten diejenigen der von der Konkurrenz angebotenen Produkte nicht übersteigen.
Die Preise können natürlich leicht abweichen da es sich um Standardkomponenten handelt und die Preise sich schnell ändern. Die Summen sind mehr als Kostenschätzung zu betrachten.

5.2 Stromspeisung

Ein Ziel unser Lösung ist die Installationskosten zu senken und die Montage zu vereinfachen. Aus diesem Grund war für unsere Lösung wichtig, Power over Ethernet zu verwenden. Moderne Hausalte werden meistens mit Ethernet Verkabelung verlegt. Dank PoE ist nur noch ein Kabel notwendig, welches Strom und Konnektivität gewährleistet.

Zusätzlich würde das System noch eine Leitung, um den Türöffner zu steuern. Auch hier kann das Endinstallation vereinfacht werden. Anstatt ein dediziertes Kabel zwischen Server und Türöffner einzuziehen, werden zwei Drähte von dem bereits installierten Ethernet Kabel verwendet.

Cisco Catalyst 3560g welcher für den PoE Stromversorgung zuständigt ist, verwendet das Phantomspeisung oder Mode A. Das heisst, dass die mit Datenübertragung belegten Drähte mit der Stromversorgung überlagert werden. Das ist möglich da Elektrizität mit eine Frequenz von 60 Hz schwingt und die Datenübertragungen im bereich 10-100MHz liegen.

STANDARD	SOURCE								COMMENTS
	Ethernet RJ-45 connector pin number								
	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 **(used by Cisco Catalyst Switches)**

Abbildung 7: Catalyst 3560g PoE Pinbelegung

Wie im Abb. 8 dargestellt werden die Adern 7 und 8 dazu verwendet um der Türöffner zu betätigen. Aus den 3 verbliebenden Adernpaare kann maximal die Ethernet Kategorie 100BASE-T erreicht werden. Da aber WebRTC eine erhebliche kleinere Bandbreite in Anspruch nimmt, stellt für die Aussensprechstelle kein Hinderniss dar.

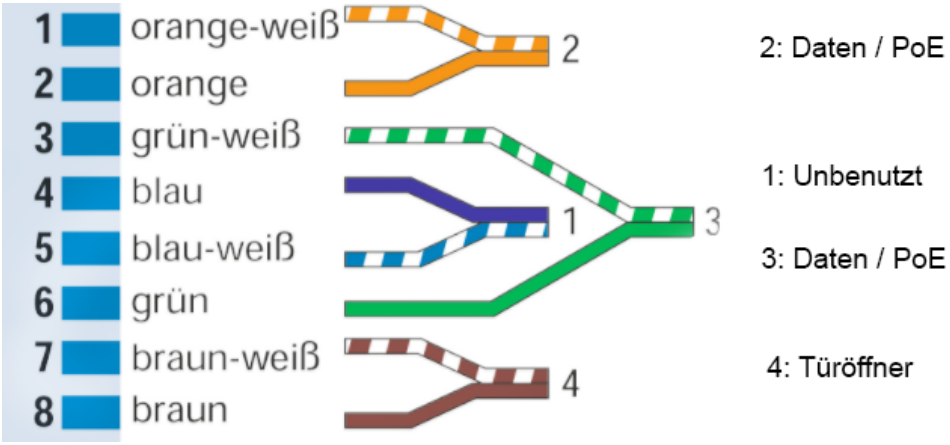

Abbildung 8: Cat. 7 Ethernet Pinbelegung für die Aussensprechstelle

5.3 Server

Der Server wird mit einem Relay-Board verbunden. Diese wird die Gongs und die Türöffner bedienen. An dieser Stelle ist die Hardware-Konfiguration sehr einfach. Mit der jetzige Hardwarekonfiguration könnten bis auf 8 Wohnungen und 8 Aussensprechstellen angeschlossen werden. Die Abb. 9 und die Abb. 10 zeigen die Pinbelegung auf den Pi und auf dem Relay-Board. Die Tabelle 3 zeigt wie die verschiedene Pins miteinander angeschlossen werden.

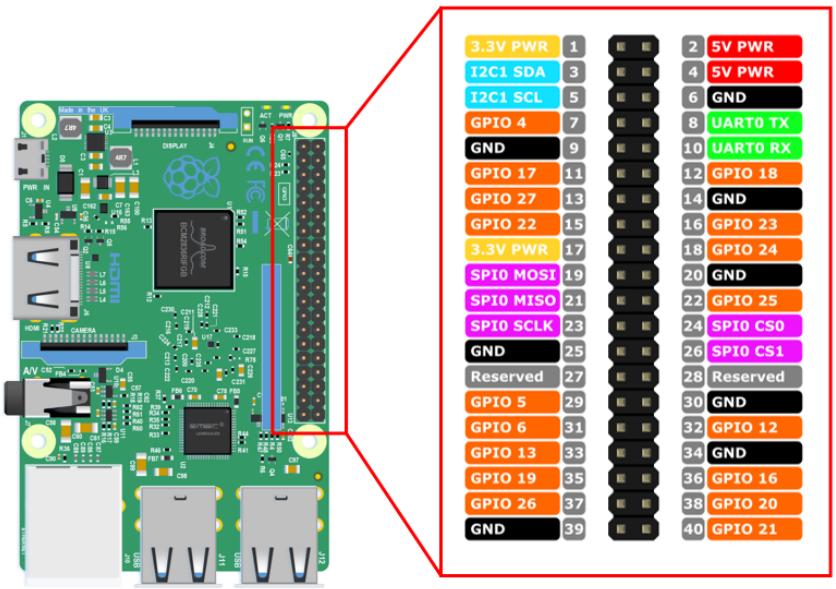

Abbildung 9: Pinbelegung für die Aussensprechstelle

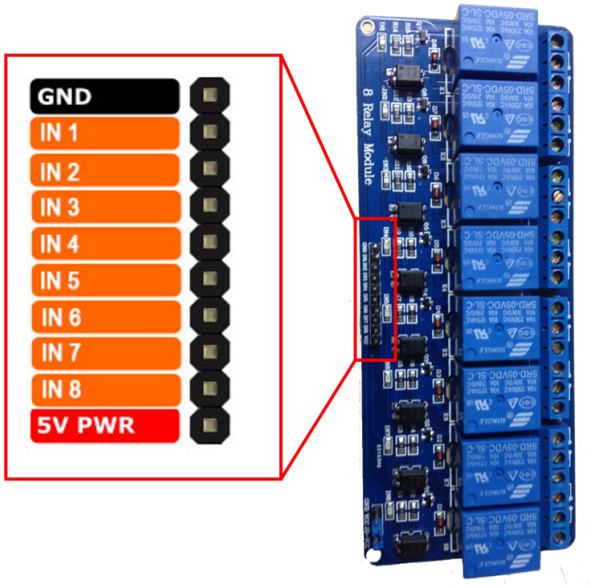

Abbildung 10: Pinbelegung für das Relais-Modul

5.4 Aussensprechstelle

Bei der Aussensprechstelle wird auch eine 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.
Für die Audiowiedergabe sind ein kleines Lautsprecher und ein Verstärker notwendeig. Die Chinch-Anschluss der Raspberry Pi hat eine zu niedrige Ausgangsleistung um den Lautsprecher direkt anschliessen zu können.
Die drei Schalter, die für die Bedienung der Aussensprechstelle notwendig sind, werden an die GPIOs der Raspberry PI angeschlossen. Die Tabelle 4 zeigt die Pinbelegung.

5.4.1 Problemen

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

Audiowidergabe

Die grösste Problematik bei der Audiowiedergabe besteht darin, dass die Massen des Raspberry Pi, der Verstärker und des Audio-Interface alle zusammen

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

gekoppelt sind. Das führt zu Brummschleifen die wiederum Störsignale auf dem Audio-Ausgang erzeugen. Um das zu vermeiden wird ein Massentrennfilter an dieser Stelle eingesetzt.
Die Störsignale sind nun fast komplett verschwunden, ein klein Hintergrundgeräusch ist aber immer noch vorhanden. Um dieses Problem umzugehen, wird ein zusätzliches Relay installiert, welche das Lautsprecher-Stromkreis unterbricht, wenn diese nicht verwendet wird.

Das Mikrophon

Das Problem bei der Audioaufnahme liegt bei der Mikrophon selber. Die Raspberry Pi besitzt kein integriertes Audio-Input. Aus diesem Grund wurde ein USB-Audio Interface verwendet. Es hat sich aber herausgestellt, dass nicht so einfach ist, kostengünstige und qualitatives USB Mikrophone zu finden. Die meisten Produkten sind nicht für den Outdoor-Betrieb gedacht. Für unser Prototyp ist das eingesetzte Mikrophon völlig ausreichen. Für das Endprodukt sollte man hier noch etwas Zeit und Geld in der Evaluation ein besseres Mikrophon investieren.

6 Software

Die Hardware ist nun vollständig und dient als Basis für die Entwicklung der Softwarekomponenten die für das System notwendig sind.

6.1 Programmiersprachen

Das System besteht aus mehrere Programme und Dienste. Für die Entwicklung werden folgende Programmiersprachen eingesetzt:

- Java
- Javascript
- PHP

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

6.1.1 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. Für Java sind auch unzählige Libraries verfügbar, insbesondere für die Hardware Steuerung der Raspberry Pi. Eine zweite Variante wäre Python gewesen, die auch das Raspberry sehr gut unterstüzt. Python ist aber zu wenig typisiert und für eher kleinere Softwarestücke gedacht.

6.1.2 PHP/Javascript

Die Client Applikation sowohl auch die Applikation bei der Aussensprechstelle werden Web-Applikationen sein. Dies ermöglicht eine schnelle und zeitgemässe Softwareentwicklung. Für dieses Projekt ist die System-Eingriffstiefe von Webapplikationen jedenfalls ausreichend. Es muss lediglich Zugriff auf Mikrofon, Lautsprecher und Kamera garantiert werden. Ein weiteres Punkt zugunsten einer Webapplikation ist die Cross-Plattform Kompatibilität.
Aus diesem Grund haben wir uns für PHP (Objektorientiert) im 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 vieles mehr als was dieses Projekt benötigt.

6.1.3 PHP Framework: Laravel

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

6.2 System Übersicht

Das System besteht aus mehrere Softwarekomponenten die zusammenarbeiten müssen (siehe Abb. 11). Die Vertraulichkeit der Kommunikation zwischen den Knoten ist von TLS immer gewährleistet. Die einzelne Komponenten, sowie das Thema Sicherheit, werden in den nächsten Kapiteln genauer beschrieben.

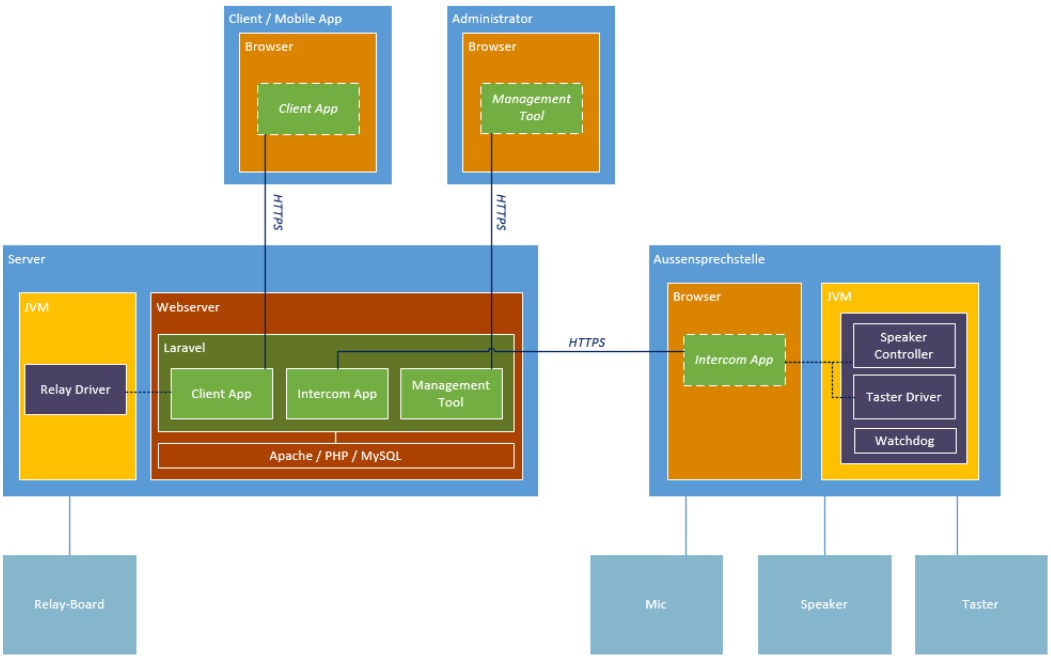

Abbildung 11: Software Ecosystem

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 den Bewohner, die Applikation bei der Aussensprechstelle wo die Bewohner angezeigt werden und das Management Tool.
Die Audio/Video-Kommunikation zwischen die Aussensprechstellen und die Client Applikationen wird mithilfe von WebRTC realisiert.

6.3 Mühsame Security Policies

Die Sicherheit spielt für dieses System eine grosse Rolle. Aus diesem Grund wurde von Anfang an geplant, das ganze Datenverkehr mit TLS zu verschlüsseln. Auch WebRTC selber weigert sich zu funktionieren, wenn keine gültige HTTPS Verbindung vorhanden ist.
Unglücklicherweise für die Entwicklung dieses Prototyp, sind die Security-Policies von den Heutige Browser immer mühsamer. Es ist zum Beispiel nicht mehr Möglich, den Browser so Einzustellen, dass die Zertifikat-Fehler 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 zu einem Hostname oder IP Adresse gebunden. Das hat als folge, dass jedes mal Etwas an dem Netzwerk angepasst wurde, müssten alle Zertifikate nochmals generiert werden.
Zusätzlich hat Google Chrome während der Entwicklung dieses Projekts mit der Version 58 die Security Policies geändert. Nach dem Update brauchten die Self-Signed-Certificates ein Zusätzliches Feld für das Subject Alternative Name (SAN). Im Netz war am anfang sehr wenig Hilfe zu finden und das hat 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]

6.4 WebRTC

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

WebRTC wird hauptsächlich für die Entwicklung von Videokonferenz Programme 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 Signaling-Process, welcher zusammen mit dem ICE (Interactive Connectivity Establishment) für den Verbindungsaufbau 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]

6.4.1 Signaling Process

Ähnlich wie bei VoIP-Telefonie *(SIP)*, brauchen die Sprechpartner ein gemeinsam bekanntes Knoten, um die Verbindung zu initialisieren (siehe Abb. 12). In den meisten Fällen ist einem Partner, die logische Adressierung der andere Partner nicht bekannt. Es besteht also keine Möglichkeit um eine P2P Verbindung auf einmal zu starten.
Im unseren Fall wäre es theoretisch möglich, da die Position der Aussensprechstelle bzw. der Server immer dieselbe sind. Allerdings wurde WebRTC nicht so konzipiert. Die Standard WebRTC API beinhaltet kein Konstrukt um eine Verbindung anhand von Bekannter IP-Adresse aufbauen zu können.
Im Internet sind es mehrere Signaling-Server Libraries verfügbar. Allerdings sind diese für andere Anwendungen gedacht. Im unseren System, wird beispielsweise nie eine Anruf von der Aussensprechstelle zu den Client Applikation gestartet,

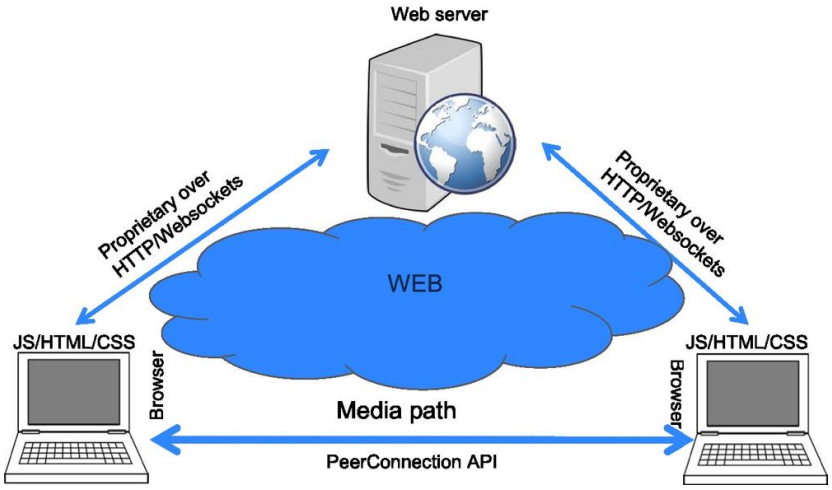

Abbildung 12: Der Signaling Prozess

sondern lediglich umgekehrt.
Für die Zwecke unser Projekt wurde ein eigenes Signaling-Server entwickelt. Dieser wird auf den Server ausgeführt und somit bleibt der Datenverkehr zwischen dem Client Applikation und der Aussensprechstelle, während jeder Schritt der Verbindungsaufbau und Kommunikation, innerhalb des lokales Netzwerkes. Das natürlich nur, solange der Bewohner sich zu Hause befindet.

6.4.2 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. Falls sich ein Sprechpartner hinter ein NAT-Knote befindet, wird für den anderen unmöglich sein eine Verbindung aufzubauen. Hier kommen die STUN-Servers im Spiel. Ähnlich wie bei dem Signalisierungsprozess stehen STUN-Servers als Hilfe für den Verbindungsaufbau da (siehe Abb. 13). STUN-Servers informieren die Clients über jegliche NAT Konfigurationen die sich dazwischen befinden würden. Die beide Sprechpartner erhalten somit Informationen über welche Ports und Öffentliche Adressen die Verbindung initialisiert werden kann. Für die Entwicklung dieses Projektes werden die Google STUN Servers verwendet, welche kostenfrei zur Verfügung stehen.
Falls sich beide Sprechpartner im gleichen lokales Netzwerk befinden, werden kei-

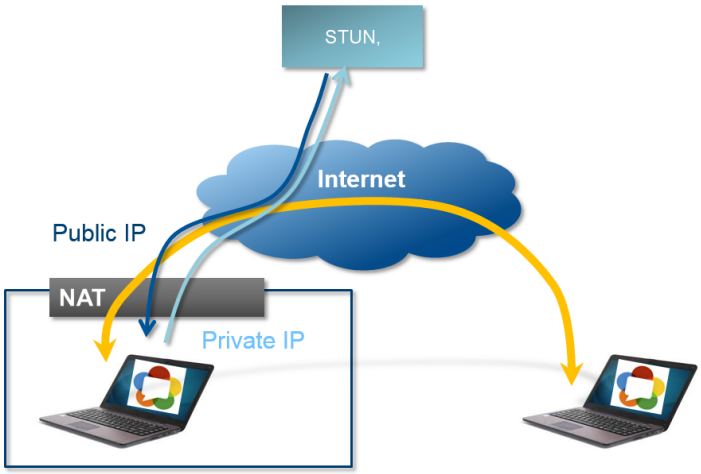

Abbildung 13: STUN Server

ne STUN-Servers benötigt und den gesamten Datenverkehr bleibt innerhalb des Heimnetzwerkes.

6.5 Webapplikationen

Während die verschiedene Java Dienste relativ kleine Programme sind, besteht den Gesamten Quellcode der Webapplikationen aus mehrere tausende Codezeilen. Das PHP-Backend im Zusammenarbeit mit Javascript auf den Client-Seite ist für die meisten Aufgaben der Anlage sowohl auch für ein Teil der Sicherheitsaspekten zuständig.

6.5.1 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 alle Aussensprechstellen verbinden können, ein Video Signal von der Kamera aller Eingänge erhalten, alle Türe öffnen und mit der Person bei der Türe über die Anlage kommunizieren können.

Die Abb. 14 zeigt das Design für die Webapplikation. Hier gezeigt ist die Smartphone Version. Dank ein Responsive-Design wird die selbe Applikation auch auf andere Geräte wie z.B. Tablets oder Computers passend angezeigt.
Bei der Design-Entwurf standen Übersichtlichkeit und Benutzerfreundlichkeit im

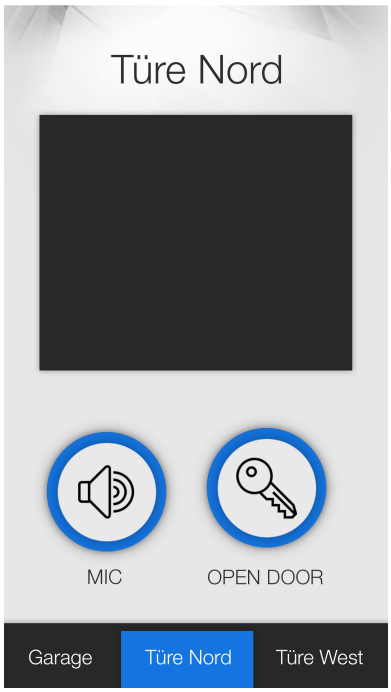

Abbildung 14: Design der Client-Webapp

Vordergrund. Aus diesem Grund werden die Tasten für die Audio-Kommunikation und für die Öffnung der Türe gross Angezeigt. Das Videostream von der ausgewählte Türe wird sofort angezeigt und benötigt keine weitere Interaktion.

6.5.2 Aussensprechstelle Webapplikation

Die Aussensprechstelle ist mit einem Bildschirm ausgestattet. Hier wird die Bewohnerliste angezeigt. Mithilfe von drei Schalter können diesen durchgeblättert und angerufen werden (siehe Abb. 15).

Währen der Bewohner die Möglichkeit hat, die Person an der Türe zu sehen, erhaltet der Besucher an der Türe kein Videosignal. Das wäre Technisch absolut möglich. Als Einwohner will man aber die Möglichkeit haben, die Türe nicht zu öffnen oder dem Besucher unsere Präsenz gar nicht bekanntgeben.

Probleme bei der Videoübertragung

Nach die ersten Tests 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.

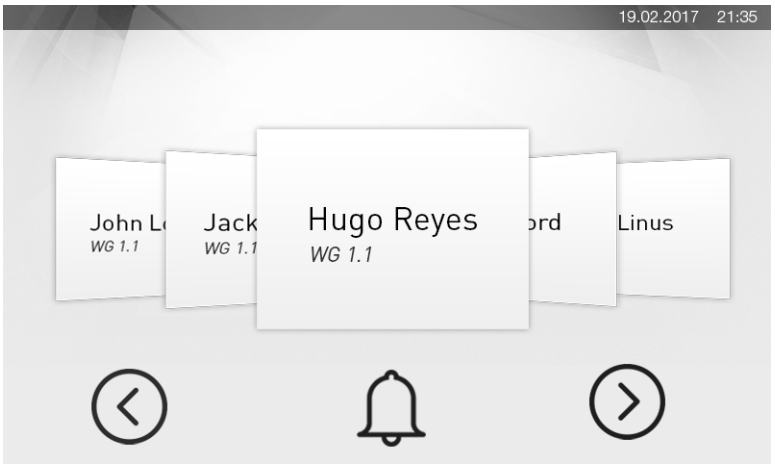

Abbildung 15: Design der Aussensprechstelle-Webapp

Nach eine Problemanalyse konnte man folgendes feststellen:

Für das Video encoding verwendet WebRTC das VP8 codec. Der Codierung im Gegensatz zu den Decodierung, wie bei der Mehrheit solche Systemen ist sehr Leistungsintensiv. Die Aussensprechstelle muss im Stande sein, die Codierung in Real-Time auszuführen. Das bringt die Rapspberry an ihre Grenzen. Obwohl eine Kamera mit hohe Auflösung im Einsatz ist, wird WebRTC im Folge des niedriges Framerates die Qualität des Stream verringern. Sobald die Qualität herabgesetzt ist, ist die Raspberry wieder im Stand die Codierung im Echtzeit durchzuführen. Aufgrund der hohe Überlastung des Prozessor während der Kodierung, tauchen zusätzlich Wärmeabführung Probeme auf. Eine verlängerte Videostreaming-Session mit erhöhte Umgebungstemperaturen, könnte der Raspberry zum Absturz bringen.
Der Raspberry Pi 3 war während der Entwicklungsphase des Prototyp die richtige Entscheidung. Haptgrund war der hohe Kompatibilität, die Standardisierung von ein sehr gut etabliertes Produktund, und die Stabilität. Dazu kommen noch die Unzählige Infos, Dokumentationen die im Interet über diese Microcontroller zu finden sind.

Alternative
Mit den gesammelte Erfahrungen während der Prototyp Entwicklung kann eine bessere Alternative zur Raspberry für eine Weiterenwicklung der Anlage ausgewertet werden.

	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 Vegleich mit die Banana Pi Alternative

Der Microcontroller Banana Pi M3 hat in Gegensatz zum Raspberry erheblich mehr Datenverarbeitungsleistung zu bieten (siehe Tabelle 5). Dazu kommt noch dass diese Microcontroller das H.264 hardware acceleration unterstüzt und somit der Videostream weiterhin optimisiert würde.
Ein weiteres Vorteil des Banana Pi is das der Raspbian OS ebenfalls ünterstützt wird. Die mit dem Projekt mitgeliferte Image des Betriebsistem für die Aussensprechstellen könnte somit auf den neue Microcontroller mit geringere Aufwand aufgespielt werden. Auch die Verkabelung stellt kein Problem dar, da die Pinbelegung eins zu eins die von der Raspberry entspricht.

6.5.3 Management Tool

Um eine schnellere Inbetriebnahme und eine zentrale Verwaltung des Systems zu gewährleisten, wurde der Management Tool etwickelt. Dieses Webapplikation ermöglicht die Erfassung von Aussensprechstellen und Bewohnern. Diese Schnittstelle wurde mit Webtechnologien entwickelt(HTML, PHP, Javascript) und wird zusammen mit den MySQL Database auf den localen Raspberry Server gehostet. Aus Sicherheitsgründen werden alle eingehende und ausgehende Verbindungen mittels TLS abhörsicher aufgebaut.
Grund für das einsetzten von Webtechnologien ist die Plattformunabhängigkeit sowie die Einfachkeit und die Standardisierung der Sprachen. Für den ersten Prototyp lag der Fokus auf die funktionale Eingenschaften der Tool. Bei einer zukünfitige Weiterentwicklung des Produkt kann man, dank der Webtechnologien mit gerigere Aufwand das Tool skalieren bzw. neue Features hinzufühgen.
Das Tool ist mit ein Login versehen, somit ist die Vertraulichkeit garantiert.

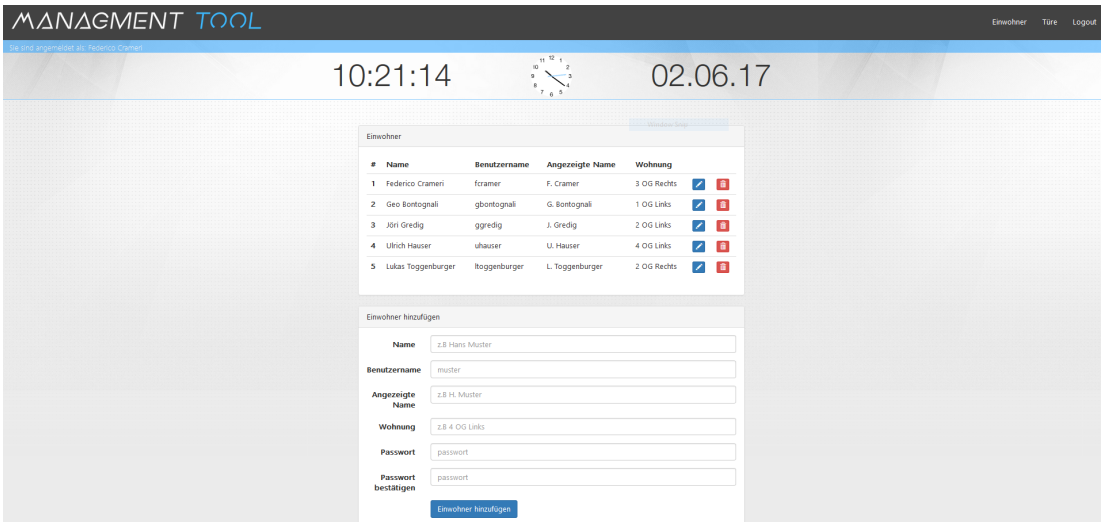

Abbildung 16: Design der Management tool

Bewohner

Unter die Bewohner Seite werden alle Wohnungen, beziehungsweise alle Bewohner aufgelistet. Diese verfühgen über ein Benutzername sowie eine Passwort die von der Client Applikation werwendet wird um sie sich bei den Server zu authentifizieren. Dieser Abschnitt bietet noch die Möglichkeit der Name und die Position der Wohnung welcher an den Aussensprechstelle angezeigt wird, abzuändern.

Türen

Bei der Einbau eine neue Tür, kann diese in dem Management Tool aufgeführt werden. Dabei muss beachtet werden dass der Id mit derjenige die auf den neu installierte Aussensprechstelle übereinstimmt. In diese Sektion sind auch die Namen der Türen definiert, welche denn auf den Client Applikation angezeigt werden.

6.5.4 Remote Verbindung

..

6.6 OS und Dienste

Nun beschrieben sind alle Dienste die auf dem Raspberry laufen werden. Diese Dienste werden benötigt um die Webapplikationen mit dem Hardware zu verbinden.

6.6.1 Raspbian

Auf alle Raspberry Pi wurde den Betriebssystem Raspbian Jessie installiert. Diese wird von Raspberry Pi Foundation mitgeliefert und gilt als besonders hochoptimerte 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. Diese erlaubt der unbeschränkte Weitergabe des Software sowie abgeleitete und modifizierte Werke weiterzugeben. Raspbian enthält Java SE Platform Prudukte welches und dem BCL(Oracle Binary Code License) lizensiert sind. Dieses Lizenz gewährleistet die obengenannten Freiheite ebenfalls.

6.6.2 Taster Controller

Die Aussensprechstelle wird durch 3 Schalter bedient. Die Aufgabe der Taster Controller besteht darin auf die GPIOs der Raspberry welcher an den Schaltern verbunden sind, abzuhöhren. Sobald ein Schalter gedrückt wird, wird eine Tastatureingabe simuliert. Durch die Simulation kann der Javascript Kode, die local im Browser ausgeführt wird, auf Schalterndruck reagieren. Somit kann auf den Aussensprechstelle Webapplikation (siehe Abb. 15) ein Bewohner ausgewählt werden (Schalter rechts und Links), und diese denn auch angerufen werden (Schalter Mitte).
Die ursprüngliche Idee war das Taster Controller, so wie alle andere Dienste, als Daemon auszuführen. Das hätte den Vorteil, dass der Daemon mittels die übliche run, stop und restart Befehl gesteuert werden könnte. Per Definition ist ein Daemon Benutzerunabhängig und genau diese Ansatzt stellte uns ein Problem dar. Eine der eingesetzten Java Library (Robot, zum Keyevent simulieren) benötigt den Zugriff auf den LXDE Desktopumgebung. Aus dem Grund das LXDE ein Benutzerspezifische Prozess ist, konnte der Taster Controller nicht als Daemon ausgeführt werden.
Um das Problem umzugehen bietet LXDE ein Autostart. Im Unix Runlevel 5 wird gewartet bis der Desktop Umgebung initialisiert ist, und erst denn wird der Taster Controller ausgeführt. Somit kann jetzt der eingesetzte Robot Library auf den LXDE zugreifen und die Tastatureingabe simulieren.

6.6.3 Speaker Controller

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

Der Dienst besteht lediglich aus ein Socket-Server, der auf ein Signal wartet und durch die GPIO der Raspberry, ein kleines Relay steuert. Das Signal kommt aus der Javascript Seite derAussensprechstelle-Webpplikation *(localhost)*. So kann den Lautsprecher bei Bedarf ein- und ausgeschaltet werden.

6.6.4 Relay Controller

Das Relay Controller ist sehr ähnlich aufgebaut wie das Speaker Controller. Auch hier handelt sich um ein kleines Socket-Server der auf ein Signal wartet und durch die GPIO der Raspberry, ein oder mehrere Relays steuert.
Das Relay Controller wird auf dem Server ausgeführt und wartet auf die Befehle von den Webapplikationen. Das Relay ist an die Türöffner und an die Gongs in den Wohnungen angeschlossen.
Der Datenaustausch zwischen den Relay-Controller und die Webapplikationen erfolgt im Form eine JSON String.
Währen bei der Speaker-Controller die befehle aus der Client Seite stammen, kommen die Daten bei dem Relay-Controller aus dem JSON-Backend auf dem Server selbst. Somit bleibt den Datenverkehr nur auf dem Localhost und werden Man in the Middle oder Injection Attacke ausgeschlossen. Aus diesem Grund lauscht dieser Server nur auf Verbindungen die aus dem Localhost stammen.

6.7 Logging

Für die Identifikation und Rückverfolgung von Fehlern sowie für den Monitoring sind Logs File von grosse Bedeutung. Diese werden bei allen Services und Dienste konsequent druchgeführt. Das Logrotate wird nicht eingesetzt, statdessen kümmert sich das Java runtime environment um die Grösse des generiertes Log Files. Aus dem Grund dass es sich noch um ein Protoyp handelt wurde das Logging

Stufe auf 7 eingestellt. In diese Stufe werden alle Emergency Nachrichten bis auf die Debug Nachrichten im Log Dateien gespeichert. Gemäss der FHS (Filesystem Hierarchy Standard) werden die Logs unter /var/log/Aussensprechstelle gesichert.

6.8 Watchdog

Die ganze Hardware, die an die Türe installiert wird, ist bei eine Endkunde schwer zugänglich. Sollte nun ein Problem mit dem System auftreten, müsste man Vorort die Anlage zurücksetzen. Die Lösung heisst hier Hardware-Watchdog, die auf dem Raspberry komplett unabhängig vom eigentlichen System läuft. Der Vorteil von ein Hardware-Watchdog ist das wenn der System bzw. der Prozessor steht, führt diese unabhängige Hardware ihre Aufgabe weiterhin aus. Der Watchdog wird als standalone Gerät im Unix erkennt. Wird diese Gerät einmal beschrieben, dann muss diese im eine Zeitintervall von 15 Sekunden erneut beschrieben werden. Ist diese Bedingung nicht erfüllt, denn wird ein Hardware-Reset von Watchdog durchgeführt und das System wird neugestartet. Das Beschrieben von der Watchdog-Gerät wird von eine Watchdog-Daemon übernommen. Durch der Konfigurationsdatei des Daemon können verschiedene Parameter des System wie Temperatur, Auslastung der Prozessor usw. überwacht werden. Besonders relevant für die Türsprechanlage ist das PID-Monitoring. Diese ermöglicht das ständig überprüfen von spezifische Prozesse und Diensten die das System benötigt, um sein Zweck als Aussensprechstelle zu erfüllen. Sobald eine diese Prozesse steht wird das System innerhalb von 15 Sekunden nuegestartet. Ein solches Mechanismus steigert die Verfügbarkeit des Dienst, die für eine Türsprechanlage von grosse Bedeutung ist.

7 Testabnahme

Überprühfung der Anforderungen

8 Ausblick

8.1 Verbesserungs- und Erweiterungsmöglichkeiten

Der aktuelle Stand des System bietet die Standardfunktionalitäten die notwendig sind um den Prototyp in einem Testumgebung einsetzen zu können. Skalabilität und Weiterentwicklung waren deshalb zwei wichtige Begriffe die den Ganzen Arbeit geprägt haben.
Das einsetzen von Webtechnologien hat dazu gebracht, das die verschiedene Benutzerschnittstellen im Zukunft, ohne grosse Aufwand und Konw-How abgeändert oder erweitert werden können.
In Bereich Hardware könnten ebenso verbesserungen vorgenommen werden. Die Komponenten wurden so ausgewählt um eine Schnelle und einfache Implementierung zu ermöglichen. Wie im Bericht schon erwähnt sollte für die Aussensprechstellen eine Leistungstärkere Microcontroller eingesetzt werden. Auch für den Kamera könnten weitere Produkte ewaluiert werden die für den Zweck besser geeinigt sind. Ein Kamerasensor mit integrierte autofokus könnte zum Beispiel die Qualität des Bild witerhin verbessern.
Die eingesetzte Komponenten wie der PoE Splitter, der Massentrennfilter und den Verstärker sind Open Source. Mit den vorhandenen Schltplänen diese Komponenten, könnte man eine einzelne Platine anfertigen lassen, die alle Funktionalitäten beinhaltet. Diese würde den Montage deutlich vereinfachen und gleichzeitig auch den Ausfallsrisiko sinken.

8.2 Einsatzmöglchkeiten

Das entstandene Prototyp ist die richtige Lösung um das machbarkeit eine solche Digitalisierung zu demonstrieren. Die Anlage soll weiterentwickelt werden, und diese Prototyp ist bestens geeignet um mögliche Interessentetn und Investoren aufmerksam zu machen. Die kompakte bauweise des Türklingelanlage und die einfache Verkabelungstechnologien ermöglichen bei Fachmessen oder Presentationen eine rasche Inbetriebname des Anlage.

9 Schlussfolgerung

Das während dem Bachelorarbeit entstandene Produkt hat gezeigt das eine Digitalisierung von ein Türklingelanlage mit den heutige Technologie möglich ist. Was für ein Laie auf eine ersten blick, als relativ einfach System wahrgenommen wird, hat uns bei der Enwicklung grosse Herausforderung bereitgehalten. Im gegensatzt zum herkömmliche analoge Türklingelanlagen werden bei unserem Prototyp alle Viedeo Streams digitalisiert, codiert und decodiert. Diese Schritt führ im gegensatz zur analoge Systemen, welche im echzeit funktionieren, zu kleinere aber spürbare delays. So auch die Qualität des Videoübertragung. Bei den jetztige stand der Technik ist eine bidirektionale im echtzeit funktinierende HD Verbindung kaum realisierbar. Die Qualität die erricht wird nähr diejenigen von den Marktführer wie z.B Skype oder Facebook.
In den letzten Jahren wurden enorme Schritte im Bereich Technologien und Froschung gemacht, und der Trend ist stets positiv. In unseren Augen wird aber, in weniger Jahren in der Welt des Hausautomation, die Kluft zwischen Analog und Digital überwinden.
Im verlauf der Arbeit wurde klar das der Weg zur digitalisierung die richtige Entscheidung war. Das digitalen Welt bring mit sich unzählige Vorteile die im zukünftige Hausautomation nicht mehr wegzudenken sind. So ist zum Beispeil die Interaktion mit den Türklingelanlage über ein Smartphone oder ein Tablet ein erhebliche Vorteil im gegensatz zur analoge Systeme. Weiterhin ermöglicht den digitales System der Zugriff auf die Videoübertragung von Aussen, was mit analoge Technologie bis jetzt kaum realisierbar war.

10 Anhang

Diese Anleitungen sind an den Weiterentwikler des Prototyp gerichtet. Mithilfe von diese Dokumentation und den mitgelieferte Image des Aussensprechstelle, soll ein Entwickler im Stand sein ein fresh installiert Raspbian OS zu einer Aussensprechstelle/Server zu konfigurieren. Alles was Konfiguriert wurde, wurde Dokumentiert und in den Anleitung aufgeführt. Diese soll auch das Hinzufügen von zukünftige Funktionalitäten erleichtern.

10.1 Aussensprechstelle Konfigurationsanleitung

10.1.1 Aktuelle Stand

Betriebssystem: Raspbian jessie with pixel
Version: April 2017
Kernel Version: 4.4

10.1.2 Namen und Passwortkonzept

Hostname: DoorPixxx (x= fortlaufende Nummerierung)
User: pi
Password: bachelor (Einfachheitshalber wurde diese schwach Passwort ausgewählt. Sollte aber bei eine Produktive inbetriebnahme zwingend geändert werden)

10.1.3 Betriebssystem Installation

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

10.1.4 Allgemeine Einstellungen

Der System soll auf dem neuste Stand aktualisieren werden

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

Mit den Terminal Kommando '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 Passwordkonzept setzen.
- Zum schluss soll noch die Zeit-Zone, den Land und die Tastatuslayout angepasst werden.

10.1.5 Bidschirm Konfiguration

Die Display Treiber von waveshare.com herunterladen und auf dem SD Karte in Root Directory speichern. (http://www.waveshare.com/wiki/4inch_HDMI_LCD) Mit folgenden bash Kommandos wird der Treiber Instelliert:

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

Nachdem das der Bildschirm Treiber installiert wurde, müssen die Einstellungen für den Bildschirm angepasst werden. Folgende Code-Zeilen müssen am ende des 'config.txt' Datei der sich in den root directory befindet, hinzugefügt werden.

```
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
```

10.1.6 Browser Kiosk-mode

Als erstes wir die unclutter tool installiert um den Mausepfeil auszublenden.

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

Kios-mode Einstellungen werden im config Datei (/home/pi/.config/lxsession/LXDE-pi/autostart) wie folgendes 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
```

10.1.7 Aussensprechstelle Initialisierung

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

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

Inhalt der 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
```

10.1.8 Taster Controller

Die Tastencontroller die für den Key Mapping zuständig ist wird von dem oben gezeigte AussensprechstelleLauncher.sh unter /home/pi/Aussensprechstelle/TasterController/TasterController.jar gestartet. Also muss die kompilierte Jar Artefakt 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»

10.1.9 Speaker Controller Service

Als erstes muss der mitgelieferte Jar Artefakt SpeakerController.jar unter folgendes Pfad kopiert werden:

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

Um den SpeakerController als Service unter Unix laufen zu lassen muss unter /etc/init.d/ der speakerController Script erzeugt werden. Der Inhalt des Script wird mit den 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 folgende commandos gestartet und gestoppt werden

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

10.1.10 Watchdog/Watchdog deamon

Um die von den Aussensprechstelle benötigte Dienste zu monitorieren die es benötigt wird ein Watchdog verwendet. Raspberry Pi hat ein «stad-alone» Hardware Watchdog die ein Autostart durchführt sobald eine der Dienste oder den OS steht. Mit folgende 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 von den Watchdog überwachen werden muss unter /etc/watchdog.conf die Konfiguratiosdatei abgeändert werden. Der Inhalt des Konfigurationsdatei wird mit dem Projekt mitgeliefert.

10.2 Server Konfigurationsanleitung

10.2.1 Aktuelle Stand

Betriebssystem: Raspbian jessie with pixel
Version: April 2017
Kernel Version: 4.4

10.2.2 Namen und Passwortkonzept

Hostname: SrvPixxx (x= fortlaufende Nummerierung)
User: pi
Password: raspberry (Default Password)

10.2.3 Software Installation

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

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

```
apt-get install nginx
udo 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
```

10.2.4 Erstellung SSL Zertifikate

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

- Erstellung SSL Zertifikat für die Client Webapplikation. Als Hostname wird hier als Beispiel intercom.app verwendet. Zuerst muss eine Konfigurationsdatei (v3.ext) mit dem folgenden 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. Als Password kann immer dieselbe verwendet werden und muss das Keystore Password der Signaling-Server 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 aus der Systemeinstellungen installiert werden.
- keystore.jks -> Das muss später in die Selber Verzeichnis kopiert werden, wo der SignalingServer installiert wird.

10.2.5 Konfiguration von Nginx

Vor dem Deploy den 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. Der Datei */etc/nginx/sites-available/default* muss editiert werden. Im rot markiert sind die Stellen die angepasst werden müssen.

- Nun müssen die zwei VirtualHosts für die zwei WebApps konfiguriert werden. Diese werden unter verschiedene Ports laufen. Für das müssen zwei Konfigurationsdatei unter */etc/nginx/sites-available* erstellt werden. Bsp: intercom.app und management.app. Der Inhalt muss wie folgendes aussehen. Im rot markiert sind die Stellen, die unterschiedlich sein müssen für die beiden Webapplikationen. 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/
sud 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
```

10.2.6 Deploy Webapplikationen

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

- Die beide Webapplikationen müssen zuerst auf dem Server unter den 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 Anmeldedaten 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 von jede 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.

10.2.7 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 in den jeweilige Projekts Verzeichnisse unter /deploy zu finden.
- Nun muss noch für den SignalingServer noch das vorher hergestellte keystore.jks kopiert werden. Der Keystore muss sich in dem gleichen Verzeichnis wie der Signaling Server befinden.
- Für beide Dienste muss noch das autostart Skript unter /etc/init.d/ kopiert werden. Die Skripte sind unter dem 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
```

10.3 Zertifikate

....

Abbildungsverzeichnis

Tabellenverzeichnis

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 Crameri 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 Crameri

Literatur

[1] *35+ jQuery 3D Slider & Carousel Plugin with Examples Demo*. URL: http://www.jqueryrain.com/demo/3d-slider-carousel/ (besucht am 18. 04. 2017).

[2] *4inch HDMI LCD - Waveshare Wiki*. URL: http://www.waveshare.com/wiki/4inch_HDMI_LCD (besucht am 20. 02. 2017).

[3] Sam Dutton Published: November 4th, 2013 Updated: November 4th und 2013 Comments: 2 Your browser may not support the functionality in this article. *WebRTC in the real world: STUN, TURN and signaling - HTML5 Rocks*. HTML5 Rocks - A resource for open web HTML5 developers. URL: https://www.html5rocks.com/en/tutorials/webrtc/infrastructure/ (besucht am 23. 02. 2017).

[4] AboutSSL.org. *How to Create and Import Self Sign SSL Certificate on Android*. AboutSSL.org. URL: https://aboutssl.org/how-to-create-and-import-self-signed-certificate-to-android-device/ (besucht am 19. 06. 2017).

[5] *Adding Push Notifications to a Web App | Web*. Google Developers. URL: https://developers.google.com/web/fundamentals/getting-started/codelabs/push-notifications/ (besucht am 20. 06. 2017).

[6] *Building a signaling server in Java - WebRTC Cookbook*. 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).

[7] *Change default users on Raspberry Pi – Gordon Lesti*. URL: https://gordonlesti.com/change-default-users-on-raspberry-pi/ (besucht am 20. 02. 2017).

[8] *Debian – Debian-Gesellschaftsvertrag*. URL: https://www.debian.org/social_contract.de.html (besucht am 06. 07. 2017).

[9] *Embedded Linux JVM Debugger (Raspberry Pi, BeagleBone Black, Intel Galileo II, and several other IoT Devices) for IntelliJ IDEA :: JetBrains Plugin Repository*. 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).

[10] 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).

[11] *Fixing Chrome 58+ [missing_subjectAltName] with openssl when using self signed certificates | Alexander Zeitler*. URL: https://alexanderzeitler.com/articles/Fixing-Chrome-missing_subjectAltName-selfsigned-cert-openssl/ (besucht am 01. 06. 2017).

[12] Wolfram Gieseke. *Raspberry Pi: Den eingebauten Hardware-Watchdog zur Überwachung nutzen – gieseke-buch.de*. URL: http://www.gieseke-buch.de/raspberrypi/eingebauten-hardware-watchdog-zur-ueberwachung-nutzen (besucht am 20. 03. 2017).

[13] *How To Create a SSL Certificate on Apache for Ubuntu 14.04*. DigitalOcean. URL: https://www.digitalocean.com/community/tutorials/how-to-create-a-ssl-certificate-on-apache-for-ubuntu-14-04 (besucht am 10. 04. 2017).

[14] *Interactive Connectivity Establishment*. In: *Wikipedia*. Page Version ID: 732705039. 2. Aug. 2016. URL: https://en.wikipedia.org/w/index.php?title=Interactive_Connectivity_Establishment&oldid=732705039 (besucht am 23. 02. 2017).

[15] *java - How are SSL certificate server names resolved/Can I add alternative names using keytool? - Stack Overflow*. 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).

[16] *Java Keytool Essentials: Working with Java Keystores | DigitalOcean*. URL: https://www.digitalocean.com/community/tutorials/java-keytool-essentials-working-with-java-keystores (besucht am 28. 02. 2017).

[17] *Lane 6 - Start a fullscreen browser kiosk on the raspberry pi*. URL: https://lane6.de/posts/start-a-fullscreen-browser-kiosk-on-the-raspberry-pi (besucht am 20. 02. 2017).

[18] *Laravel Notification Channels*. URL: http://laravel-notification-channels.com/webpush/ (besucht am 21. 06. 2017).

[19] *Linux Java Service Wrapper Example*. URL: http://www.jcgonzalez.com/linux-java-service-wrapper-example (besucht am 23. 02. 2017).

[20] *Linux Watchdog configuring*. URL: http://www.sat.dundee.ac.uk/psc/watchdog/watchdog-configure.html#Process_Monitoring_by_PID_File (besucht am 20. 03. 2017).

[21] *LSBInitScripts - Debian Wiki*. URL: https://wiki.debian.org/LSBInitScripts (besucht am 01. 03. 2017).

[22] *Raspberry Pi in Kiosk Mode*. OSH Lab. 25. Apr. 2016. URL: https://oshlab.com/raspberry-pi-kiosk-mode/ (besucht am 20. 02. 2017).

[23] *Raspberry Pi in Kiosk Mode - OSH Lab*. URL: https://oshlab.com/raspberry-pi-kiosk-mode/ (besucht am 20. 02. 2017).

[24] *Raspberry Pi System Logging and Loggly*. URL: http://blog.scphillips.com/posts/2015/05/raspberry-pi-system-logging-and-loggly/ (besucht am 15. 03. 2017).

[25] *Raspberry Pi • View topic - Low resolution on Chromium - WebRTC*. URL: https://www.raspberrypi.org/forums/viewtopic.php?f=43&t=159941 (besucht am 15. 03. 2017).

[26] *Raspberry Pi • View topic - Poor audio quality on analog output*. URL: https://www.raspberrypi.org/forums/viewtopic.php?f=38&t=37038 (besucht am 07. 06. 2017).

[27] Deven Rathore. *send push notifications to mobile devices with laravel*. Dunebook.com. 10. Okt. 2015. URL: https://www.dunebook.com/send-push-notifications-to-mobile-devices-with-laravel/ (besucht am 21. 06. 2017).

[28] *Reliable Projects 2: Using the Internal WatchDog Timer for the Raspberry Pi*. SwitchDoc Labs. 20. Nov. 2014. URL: http://www.switchdoc.com/2014/11/reliable-projects-using-internal-watchdog-timer-raspberry-pi/ (besucht am 22. 03. 2017).

[29] *SSL*. URL: https://crsr.net/Notes/SSL.html (besucht am 19. 06. 2017).

[30] *SSLContext (Java Platform SE 7)*. URL: http://docs.oracle.com/javase/7/docs/api/javax/net/ssl/SSLContext.html (besucht am 28. 02. 2017).

[31] *Tutorial: Implement Push Notifications in your PhoneGap Application : Devgirl's Weblog*. URL: http://devgirl.org/2013/07/17/tutorial-implement-push-notifications-in-your-phonegap-application/ (besucht am 20. 06. 2017).

[32] *x11 - How to make /etc/init.d script act like it's launched under X? - Unix & Linux Stack Exchange*. URL: `http://unix.stackexchange.com/questions/98967/how-to-make-etc-init-d-script-act-like-its-launched-under-x` (besucht am 01. 03. 2017).