

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

Zusammenfassung

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi.

Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer

Abstract

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi.

Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer

Inhaltsverzeichnis

1	Einführung	1
1.1	Problemstellung	1
1.2	Grundidee	1
2	Projektplanung	2
2.1	Prozess	2
2.2	Zeitplanung	3
3	Aktueller Stand der Technik	4
3.1	Die Herausforderungen der Digitalisierung	4
3.2	Marktsituation	5
4	Lösungskonzept	6
5	Hardware	7
5.1	Komponenten	7
5.2	Stromspeisung	8
5.3	Server	9
5.4	Aussensprechstelle	10
5.4.1	Problemen	10
6	Software	13
6.1	Programmiersprachen	13
6.1.1	Java	13
6.1.2	PHP/Javascript	13
6.1.3	PHP Framework: Laravel	14
6.2	System Übersicht	14
6.3	Mühsame Security Policies	15
6.4	WebRTC	16
6.4.1	Signaling Process	16
6.4.2	STUN Servers & Remote Verbindung	17
6.5	Webapplikationen	18
6.5.1	Client Webapplikation	18
6.5.2	Aussensprechstelle Webapplikation	19
6.5.3	Management Tool	21
6.5.4	Remote Verbindung	22

6.6	OS und Dienste	22
6.6.1	Raspbian	23
6.6.2	Taster Controller	23
6.6.3	Speaker Controller	23
6.6.4	Relay Controller	24
6.7	Logging	24
6.8	Watchdog	25
7	Testabnahme	26
8	Fazit und Ausblick	27
9	Anhang	28
9.1	Aussensprechstelle Konfigurationsanleitung	28
9.1.1	Aktuelle Stand	28
9.1.2	Namen und Passwortkonzept	28
9.1.3	Betriebssystem Installation	28
9.1.4	Allgemeine Einstellungen	28
9.1.5	Bidschirm Konfiguration	29
9.1.6	Browser Kiosk-mode	29
9.1.7	Aussensprechstelle Initialisierung	30
9.1.8	Taster Controller	31
9.1.9	Speaker Controller Service	31
9.1.10	Watchdog/Watchdog deamon	32
9.2	Server Konfigurationsanleitung	32
9.2.1	Aktuelle Stand	32
9.2.2	Namen und Passwortkonzept	33
9.2.3	Software Installation	33
9.2.4	Erstellung SSL Zertifikate	33
9.2.5	Konfiguration von Nginx	35
9.2.6	Deploy Webapplikationen	37
9.2.7	Deploy Dienste und Services	37
9.3	Zertifikate	38
	Literaturverzeichnis	39
	Abbildungsverzeichnis	40
	Tabellenverzeichnis	41

Abkürzungsverzeichnis	42
Eidesstattliche Erklärung	43

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 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 Türsprechanlage 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 Türsprechanlage 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 Türsprechanlage diese Standards in Betracht ziehen. Dieses System soll den Benutzern ermöglichen, Ihre Sprechtüranlage 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 (siehe Abb. 1). 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.

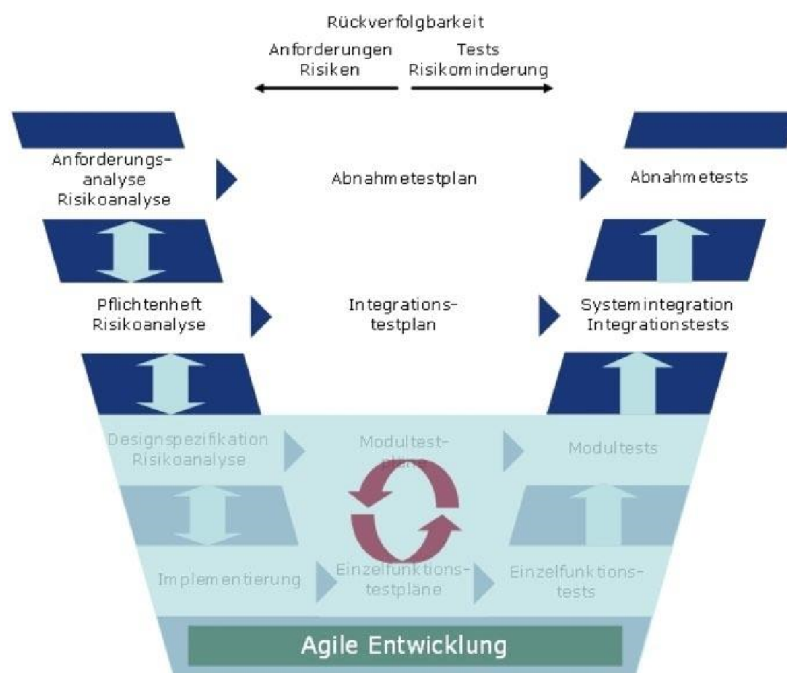


Abbildung 1: Hybrides Vorgehensmodell

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.

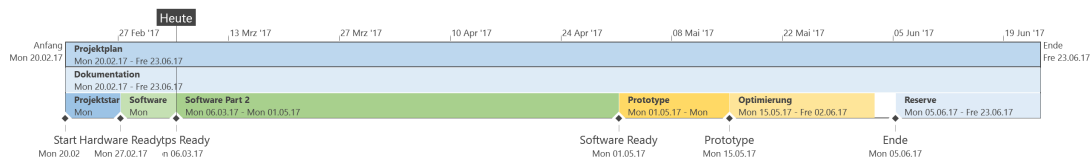


Abbildung 2: Zeitplanung mit Meilensteinen

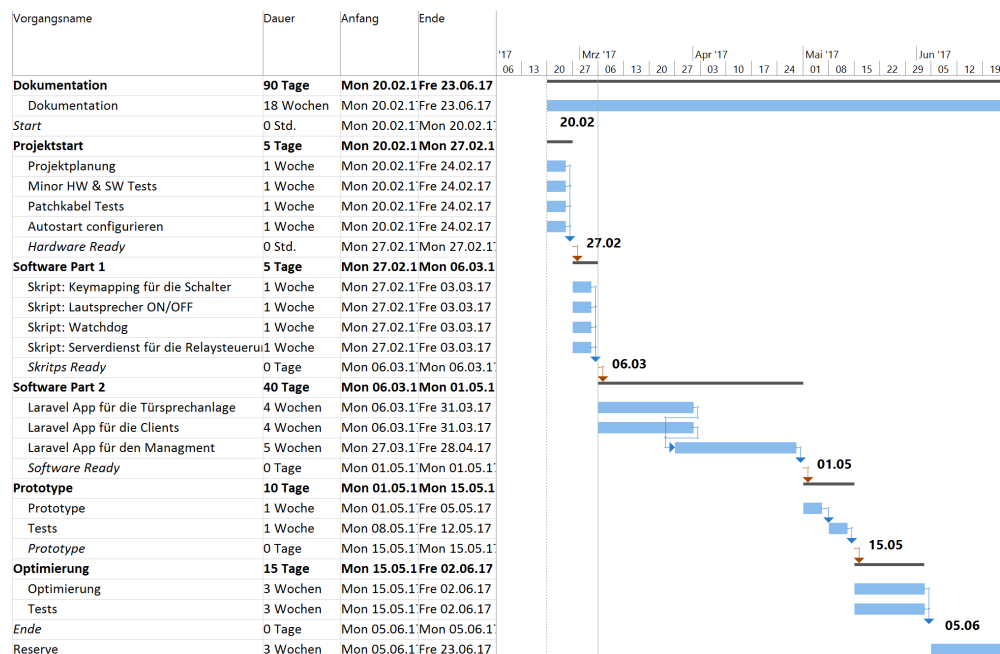


Abbildung 3: Projektplanung

3 Aktueller Stand der Technik

Eine Türsprechanlage, 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-HD 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, weiss, 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 Gegensprechanlage. 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 Türsprechanlage notwendig sind.

Es werden nun zwei Begriffe erklärt, die in diesem Dokument von grosse Bedeutung sind. Das erste ist die Türsprechanlage. 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.

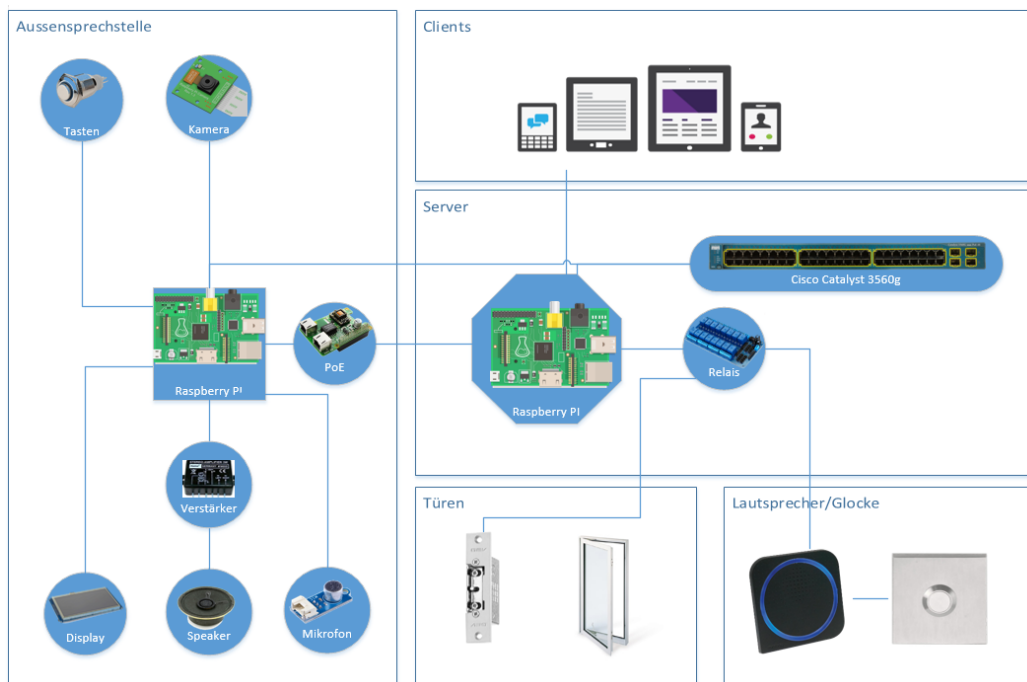


Abbildung 6: Hardware Ecosystem

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	<i>Kleinmaterial</i>	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	<i>Kleinmaterial / Gehäuse</i>	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 Kosten-schätzung zu betrachten.

5.2 Stromspeisung

Ein Ziel unserer 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 Hausalt werden meistens mit Ethernet Verkabelung verlegt. Dank PoE ist nur noch ein Kabel notwendig, welches Strom und Konnektivität gewährleistet.

Zusätzlich braucht das System eine Leitung, welche den Server und der Türöffner verbindet. Auch hier kann die 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 zuständig ist verwendet die Phantomspeisung oder Mode A. Das heisst, dass die mit Datenübertragung belegten Drähte mit der Stromversorgung überlagert werden. Das ist möglich da Elektrizität mit einer 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 in Abb. 8 dargestellt werden die Adern 7 und 8 dazu verwendet um den Türöffner zu betätigen. Aus den 3 verbleibenden Adernpaaren kann maximal die Ethernet Kategorie 100BASE-T erreicht werden. Da aber WebRTC eine erhebliche kleinere Bandbreite in Anspruch nimmt, stellt für die Aussensprechtellen kein Hindernis dar.

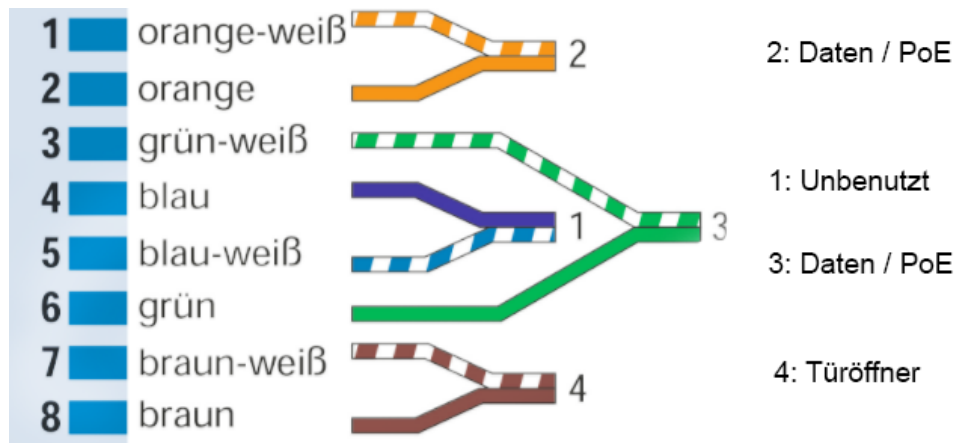


Abbildung 8: Cat. 7 Ethernet Pinbelegung für die Aussensprechstellen

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 PIN-Belegung auf den Pi und auf dem Relay-Board. Die Tabelle 3 zeigt wie die verschiedene Pins miteinander angeschlossen werden.

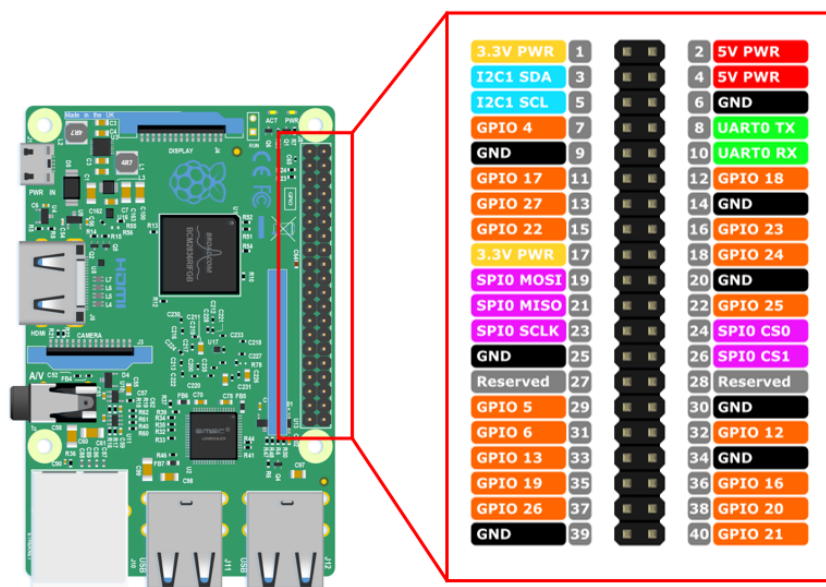


Abbildung 9: Pinbelegung für die Aussensprechstellen

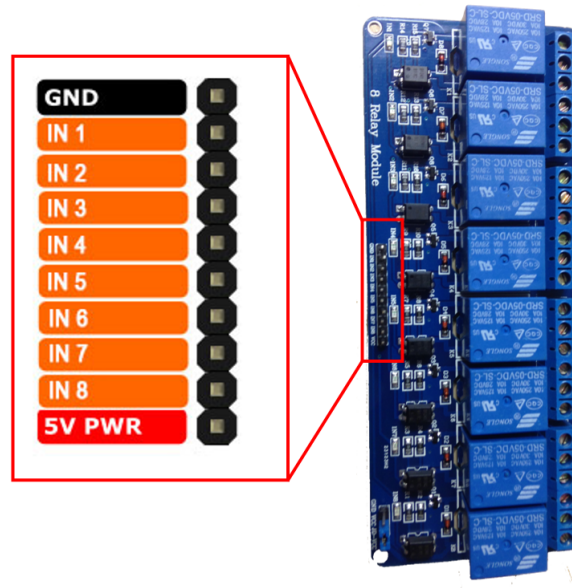


Abbildung 10: Pinbelegung für das Relais-Modul

5.4 Aussensprechstelle

Bei der Aussensprechstelle wird auch eine Raspberry Pi eingesetzt. Hier sind mehrere Zusatzkomponenten notwendig. Die Speisung an dieser Stelle erfolgt nur über PoE, aus diesem Grund ist PoE-Splitter vorhanden.

Für die Audiowiedergabe ist ein kleines Lautsprecher und ein Verstärker nötig. Die Chinch-Anschluss der Raspberry Pi hat eine viel zu kleine Ausgangsleistung um direkt ein solches Lautsprecher 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 PIN-Zuweisung.

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.

Audiowiedergabe

Die grösste Problematik bei der Audiowiedergabe besteht darin, dass die Massen des Raspberry Pi, der Verstärker und des Audio-Interface alle zusammen gekoppelt sind. Das führt zu Brummschleifen die wiederum Störsignale auf dem

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

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 unterstützt. 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.

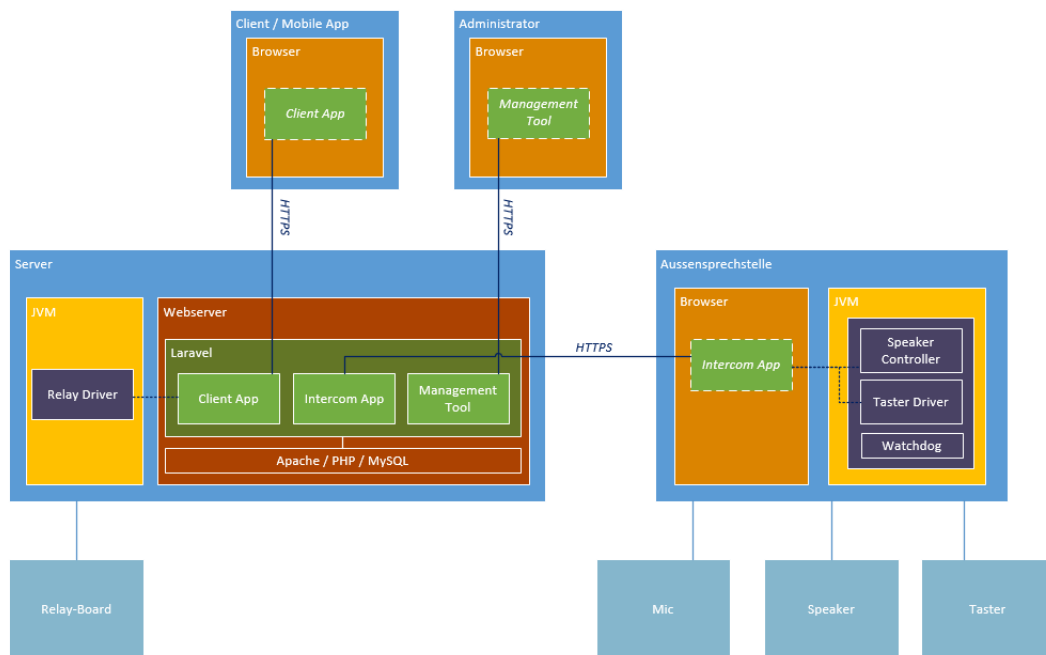


Abbildung 11: Software Ecosystem

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-Apps 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](https://developers.google.com/chrome/updates/dev/rel.chrome/58.0.3021.104/changes/changelog_beta#chrome-58-0-3021-104)]

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 Sprechpartnern (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 Aussensprechstellen 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-App gestartet, sondern

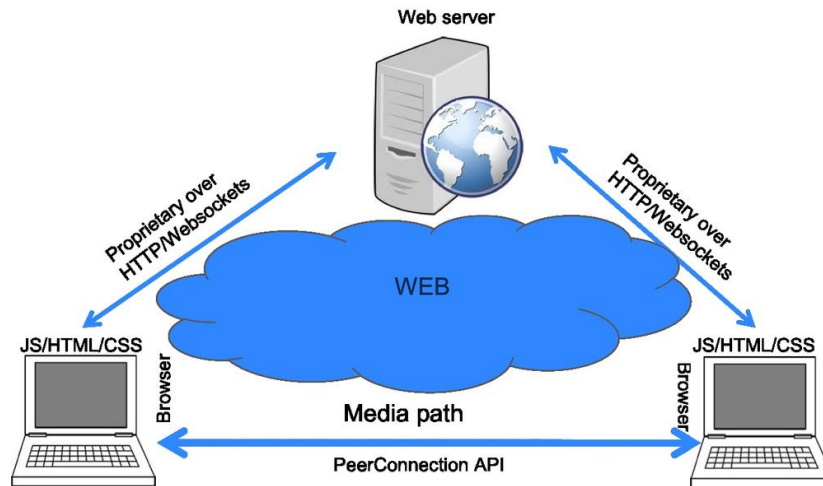


Abbildung 12: Der Signaling Prozess

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-App 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 Aussensprechstellen sich verbinden zu können. Hier stellt das NAT-Protokoll (Network Address 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-

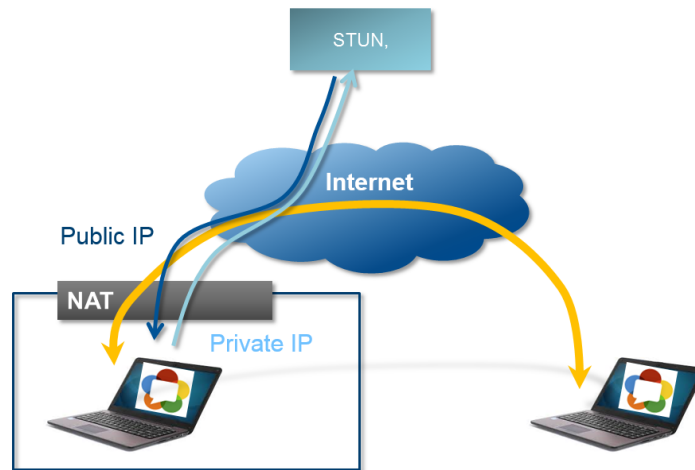


Abbildung 13: STUN Server

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

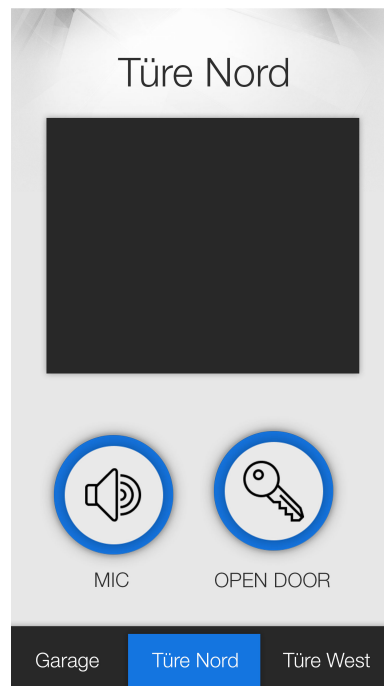


Abbildung 14: Design der Client-Webapp

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).

Während der Bewohner die Möglichkeit hat, die Person an der Türe zu sehen, erhält 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.

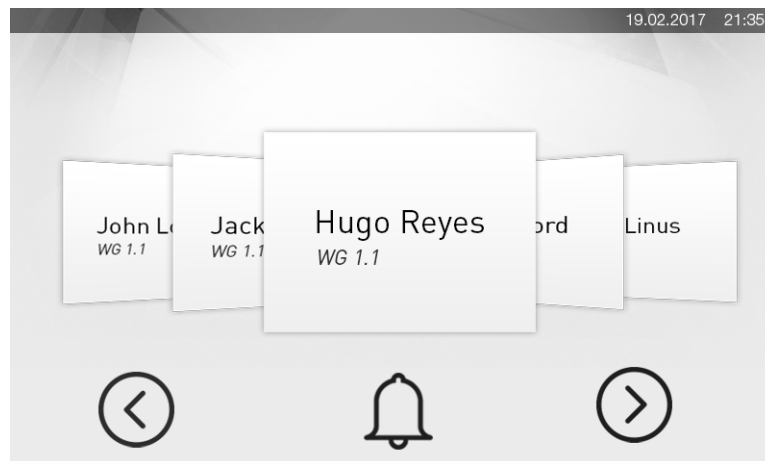


Abbildung 15: Design der Client-Webapp

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 quasi in Real-Time auszuführen. Das bringt die Rapsberry an ihre Grenzen. Obwohl eine Kamera mit hohe Auflösung im Einsatz ist, wird WebRTC im Folge des niedrigen Framerates die Qualität des Stream verringern. Sobald die Qualität herabgesetzt ist, ist die Raspberry wieder im Stand die Codierung im Echtzeit durchzuführen.

Aufgrund der hohen Überlastung des Prozessor während der Kodierung, tauchen zusätzlich Wärmeabführung Probleme auf. Bei einer verlängerten Videostreaming-Session mit einer erhöhten Umgebungstemperatur, könnte der Raspberry zu einem Absturz kommen.

Der Raspberry Pi 3 war während der Entwicklungsphase des Prototyps die richtige Entscheidung. Hauptgrund war die hohe Kompatibilität, die Standardisierung von einem sehr gut etablierten Produkt und die Stabilität. Dazu kommen noch die unzähligen Infos, Dokumentationen, die im Internet über diesen Microcontroller zu finden sind.

Alternative

Mit den gesammelten Erfahrungen während der Prototyp-Entwicklung kann eine bessere Alternative zur Raspberry für eine Weiterentwicklung der Anlage ausge-

	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 der Banana Pi Alternative

wertet werden.

Der Microcontroller Banana Pi M3 hat in Gegensatz zum Raspberry erheblich mehr Datenverarbeitungsleistung zu bieten (siehe ??). Dazu kommt noch dass diese Microcontroller das H.264 hardware acceleration unterstützt und somit der Videostream weiterhin optimisiert würde.

Ein weiteres Vorteil des Banana Pi ist das der Raspbian OS ebenfalls unterstützt wird. Die mit dem Projekt mitgelieferte Image des Betriebssystems für die Aussen-sprechstellen könnte somit auf den neuen Microcontroller mit geringem Aufwand aufgespielt werden. Auch die Verkabelung stellt kein Problem dar, da die Pinbelegung eins zu eins die von der Raspberri entspricht.

6.5.3 Management Tool

Um eine schnellere Inbetriebnahme und eine zentrale Verwaltung des Systems zu gewährleisten, wurde der Management Tool entwickelt. Dieses Portal ermöglicht die Erfassung der Gegensprechanlagen und der Wohnungen. Diese Schnittstelle wurde mit Webtechnologien entwickelt (HTML, PHP, Js) und wird zusammen mit der MySQL Database auf dem lokalen Raspberry Server gehostet. Aus Sicherheitsgründen werden alle eingehende und ausgehende Verbindungen abhörsicher aufgebaut.

Grund dafür das eingesetzten von Webtechnologien ist die Plattformunabhängigkeit sowie die Einfachheit und die Standardisierung der Sprachen. Für den ersten Prototyp lag der Fokus auf die funktionale Eigenschaften der Tool. Bei einer zukünftigen Weiterentwicklung des Produkts kann man, dank der Webtechnologien mit geringem Aufwand das Tool skalieren bzw. neue Features hinzufügen.

Das Tool ist mit einem Login versehen, somit ist sichergestellt das nur der Hausverwalter die Anlage verwalten kann.

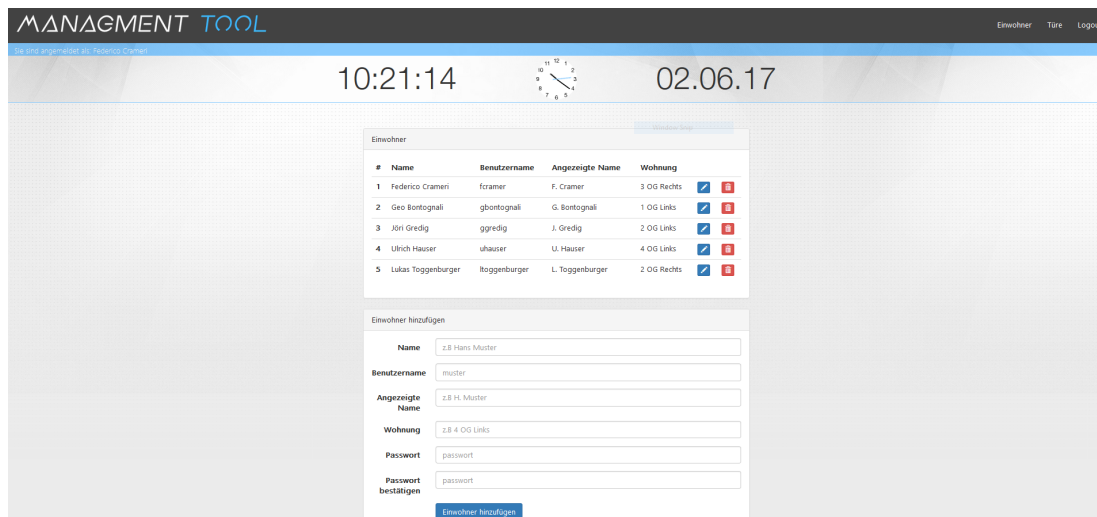


Abbildung 16: Design der Management tool

Bewohner

Unter die Bewohner Seite werden alle Wohnungen, beziehungsweise alle Bewohner aufgelistet. Diese verfügen über ein Benutzername sowie eine Passwort die Von der Client App wervendet 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 App (Siehe Abbildung 16) 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 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. Diese erlaubt der unbeschränkte Weitergabe des Software sowie abgeleitete und modifizierte Werke weiterzugeben. Raspbian enthält Java SE Platform Produkte welches und dem BCL(Oracle Binary Code License) lizenziert sind. Dieses Lizenz gewährleistet die obengenannten Freiheiten ebenfalls.

6.6.2 Taster Controller

Die Aussensprechstelle wird durch 3 Schalter bedient. Die drei Schaltern werden an die GPIO-Pins des Raspberry PI angeschlossen. Die Aufgabe der Taster-Controller besteht darin, die GPIO-Input Signale, als verwendbare Tastatur-Eingaben umzuwandeln. Somit kann die GUI an der Aussensprechstelle gesteuert werden.

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 Befehle gesteuert werden könnte. Eine der eingesetzten Java-Library benötigt aber den zugriff auf dem Graphisches Umgebung. Das Problem besteht darin, dass ein Daemon Benutzer-Unabhängig ist, während der X-Server beim Login einem Benutzer ausgeführt wird. Das ausführen der Deamon erst ab Init 5, da wo auch der X-Server ausgeführt wird, konnte aus diesem Grund das Problem auch nicht lösen. Die verwendete Library hat also keine Möglichkeit, als Deamon eine Verbindung mit dem X-Server aufzubauen.

Die Desktop-Umgebung LXDE welche von Raspbian verwendet wird, bietet aber ein Autostart welches das Taster-Controller nach dem Initialisierung des X-Server, unter dem gleichen Benutzer ausführt.

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 der Aussensprechstelle-Webapplikation (*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.

Während bei der Speaker-Controller die Befehle aus der Client Seite stammen, kommen die Daten bei dem Relay-Controller aus dem PHP-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 durchgeführt. Das Logrotate wird nicht eingesetzt, stattdessen kümmert sich das Java runtime environment um die Grösse des generiertes Log File. 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 einem Endkunde schwer zugänglich. Sollte nun ein Problem mit dem System auftreten, müsste man vor Ort die Anlage zurücksetzen. Die Lösung heisst hier Hardware-Watchdog, die auf dem Raspberry komplett unabhängig vom eigentlichen System läuft. Der Vorteil von einem Hardware-Watchdog ist, dass wenn das System bzw. der Prozessor steht, führt diese unabhängige Hardware ihre Aufgabe weiterhin aus. Der Watchdog wird als standalone Gerät im Unix erkannt. Wird dieses Gerät einmal beschrieben, dann muss dieses in einem Zeitintervall von 15 Sekunden erneut beschrieben werden. Ist diese Bedingung nicht erfüllt, dann wird ein Hardware-Reset von Watchdog durchgeführt und das System wird neugestartet. Das Beschreiben von der Watchdog-Gerät wird von einem Watchdog-Daemon übernommen. Durch die Konfigurationsdatei des Daemons können verschiedene Parameter des Systems wie Temperatur, Auslastung der Prozessor usw. überwacht werden. Besonders relevant für die Türsprechanlage ist das PID-Monitoring. Dieses ermöglicht das ständige Überprüfen von spezifischen Prozessen und Diensten, die das System benötigt, um seinen Zweck als Aussensprechstelle zu erfüllen. Sobald einer dieser Prozesse steht, wird das System innerhalb von 15 Sekunden neugestartet. Ein solches Mechanismus steigert die Verfügbarkeit des Dienstes, der für eine Türsprechanlage von grosser Bedeutung ist.

7 Testabnahme

Überprüfung der Anforderungen

8 Fazit und Ausblick

Der Fazit wir hier noch hinzugefügt.

9 Anhang

Diese Anleitungen sind an den Weiterentwickler 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.

9.1 Aussensprechstelle Konfigurationsanleitung

9.1.1 Aktuelle Stand

Betriebssystem: Raspbian jessie with pixel

Version: April 2017

Kernel Version: 4.4

9.1.2 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)

9.1.3 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

9.1.4 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 Tastaturlayout angepasst werden.

9.1.5 Bildschirm 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 Installiert:

```
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
```

9.1.6 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
```

9.1.7 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

```

9.1.8 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»

9.1.9 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 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 folgende commands gestartet und gestoppt werden

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

9.1.10 Watchdog/Watchdog daemon

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 Konfigurationsdatei abgeändert werden. Der Inhalt des Konfigurationsdatei wird mit dem Projekt mitgeliefert.

9.2 Server Konfigurationsanleitung

9.2.1 Aktuelle Stand

Betriebssystem: Raspbian jessie with pixel

Version: April 2017

Kernel Version: 4.4

9.2.2 Namen und Passwortkonzept

Hostname: SrvPixxx (x= fortlaufende Nummerierung)

User: pi

Password: raspberry (Default Password)

9.2.3 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
```

9.2.4 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 Passwort kann immer dieselbe verwendet werden und muss das Keystore Passwort 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.

9.2.5 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

```

9.2.6 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.

9.2.7 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
```

9.3 Zertifikate

....

Literatur

Abbildungsverzeichnis

1	Hybrides Vorgehensmodell	2
2	Projektplanung Meilensteine	3
3	Projektplanung	3
4	Analoge Türsprechanlage mit In-House Display	4
5	Telecom Behnkle MyIntercom	5
6	Hardware Ecosystem	6
7	Catalyst Pinouts	8
8	EthernetPinbelegung	9
9	EthernetPinbelegung	9
10	EthernetPinbelegung	10
11	Software Ecosystem	14
12	Der Signaling Prozess	17
13	STUN Server	18
14	Design der Client-Webapp	19
15	Design der Client-Webapp	20
16	Design der Management tool	22

Tabellenverzeichnis

1	Server HW Komponenten	7
2	Aussensprechstelle HW Komponenten	7
3	PIN-Zuweisung zwischen den Server und die Relais Module	11
4	PIN-Zuweisung zwischen den Raspberry PI und die Schalter . . .	11
5	Verwendete Raspberry Pi im Vergleich mit die Banana Pi Alternative	21

Abkürzungsverzeichnis

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

