

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

In dieser Bachelorarbeit geht es darum ein Werkzeug/Tool zu entwickeln, das Propeller-Motor-Systeme von Leicht- bis Ultraleichtflugzeugen, sowie von Motorschirmen analysieren bzw. optimieren kann. Das Werkzeug soll weiterführende Erkenntnisse über Propeller und deren Einsatzbereich zur Verfügung stellen. Dazu werden Geometrieinformationen des Propellers von einem CAD-Programm ins Werkzeug geladen und ausgewertet. Gegebenenfalls können auch Geometrieänderungen, im Wesentlichen Änderungen an der Verdrillung des Propellers, vorgenommen werden. Anschliessend kann das Rotorsystem durch Berechnung und graphische Darstellung der Ergebnisse für verschiedene Flugsituationen ausgelegt bzw. optimiert werden.

Ebenfalls wurde eine Verifikation mit Messdaten der Firma Helix Carbon GmbH durchgeführt, welche Hinweise auf die Richtigkeit der Berechnung gab. Auch wurden einige Beispiele zur Optimierung und zur Auslegung durchgeführt, um mögliche Anwendungsfälle aufzuzeigen.

Abstract

This bachelor's thesis is about to develop a tool, which can analyze, respectively optimize propeller-engine-systems from light to ultralight aircrafts and from paramotors. The goal was to create a tool, which provides further findings about propellers and their application area. Therefore, the given propeller geometry information from a CAD program is loaded and analyzed in the tool. There can also be made optionally geometry changes, substantially changes made to the twist of the propeller. Subsequently, the rotor system can be designed and optimized by calculation and graphical representation of the results for different flight situations.

There was also carried out verification with measurements of Helix Carbon GmbH, which gave evidence on the accuracy of the calculation. Some examples for optimization and analysis were explained in order to highlight possible applications.

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	4
3.1	Die Herausforderungen der Digitalisierung	4
3.2	Marktsituation	5
4	Hardware	6
4.1	Server	6
4.2	Aussensprechstelle	6
4.3	Microcontroller Problematik	7
4.3.1	Alternative	7
4.4	PoE	8
5	Software	10
5.1	Programmiersprachen	10
5.1.1	Java	10
5.1.2	PHP/Javascript	10
5.1.3	PHP Framework: Laravel	11
5.2	System Übersicht	11
5.3	Raspbian	12
5.4	Dienste	12
5.4.1	Taster Controller	12
5.4.2	Speaker Controller	13
5.4.3	Relay Controller	13
5.4.4	Signaling Server	13
5.5	Logging	13
5.6	Watchdog	14
5.7	Webapplikationen	14
5.7.1	Client Webapplikation	14
5.7.2	Aussensprechstelle Webapplikation	15

5.8	Management Tool	15
5.8.1	Bewohner	16
5.8.2	Türen	16
5.9	Remote Verbindung	16
5.10	WebRTC	17
5.10.1	Signaling Process	17
5.10.2	STUN Servers & Remote Verbindung	18
Literaturverzeichnis		20
Abbildungsverzeichnis		21
Tabellenverzeichnis		22
Abkürzungsverzeichnis		23
Eidesstattliche Erklärung		24

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.

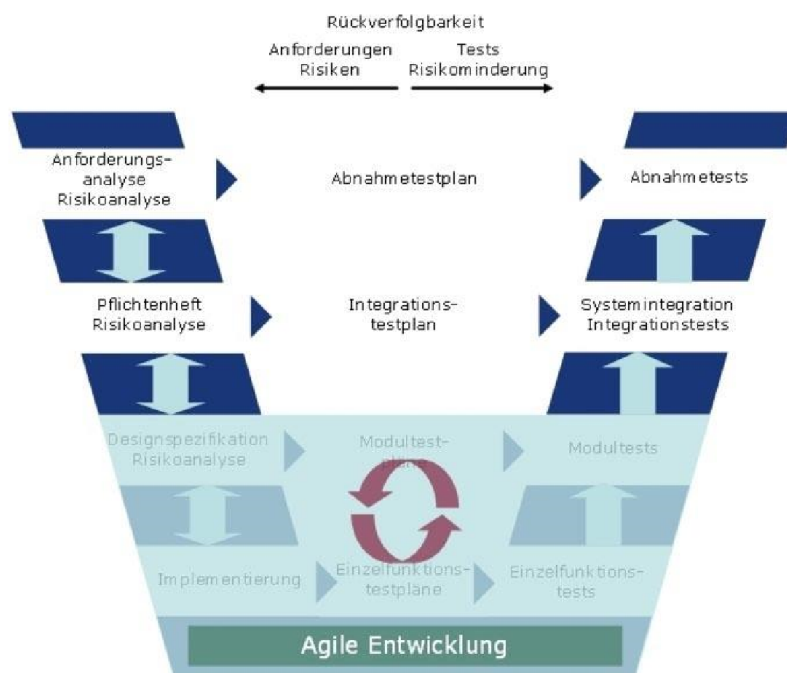


Abbildung 1: Hybrides Vorgehensmodell

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.

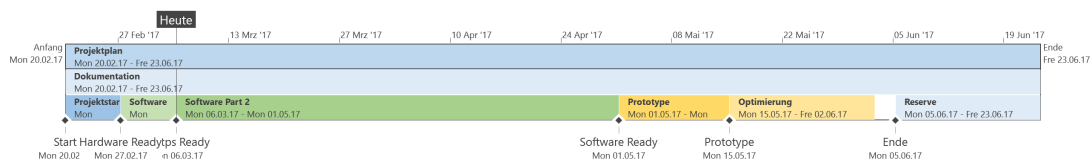


Abbildung 2: Zeitplanung mit Meilensteinen

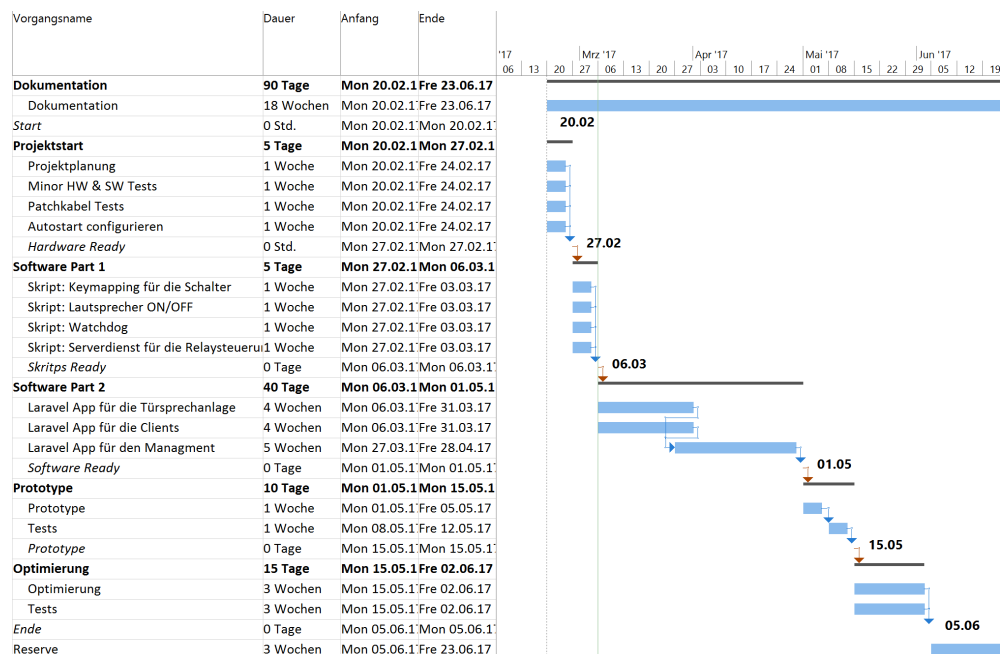


Abbildung 3: Projektplanung

3 Aktueller Stand

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 Hardware

Das System wird Hardwareseitig in zwei Teile unterteilt. Der Server, die zentrale Einheit und die Aussensprechstelle. An beide Orte wird eine Raspberry Pi 3 und das nötige Hardware eingesetzt.

4.1 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. Je nach wie viele Gongs und Türe verbunden werden müssen, könnten bis zwei 8-Channel Relayboards verbunden werden.

<- ABBILDUNG CON LA PI E I RELAY E I PIN A CUI SONO COLLEGATI

4.2 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 zu kleine innere Widerstand um direkt ein solches Lautsprecher anschliessen zu können. Die Hauptproblematik nun besteht darin, dass die Massen des Raspberry Pi, der Verstärker und des Audio-Interface alle zusammen gekoppelt sind. Das führt zu Brunschleifen die wiederum Störsignale auf dem Audio-Ausgang erzeugen. Um das zu vermeiden ist eine Massentrennfilter an dieser Stelle notwendig.

<- ABBILDUNG CON LA PI E COMPONENTI COLLEGATI

	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

Tabelle 1: My caption

4.3 Microcontroller Problematik

WebRTC basiert, für den Video encoding auf den von Google Inc. offengelegte VP8 codec. Der Codierung im Gegensatz zu den Decodierung, wie bei der Mehrheit solche Systemen ist sehr Leistungsintensiv. Die Situation kommt bei der Aussprechstelle genau so vor, dort wird der Videostream auf den Raspberry Codiert und am Server für die Decodierung weitergeleitet. Diese bringt, was der Rechenkapazität anbelangt, die Raspberry an ihre Grenzen. Die Problematik stellt für End-Qualität des Streaming eindeutig den Flaschenhals dar. Obwohl eine Kamera mit hoher Auflösung im Einsatz ist, wird WebRTC anhand der niedrigen Framerate die Qualität des Streams verringern. Sobald die Qualität herabgesetzt ist, ist die Raspberry im Stand die Codierung in Echtzeit durchzuführen.

Aufgrund der hohen Überlastung des Prozessors, die während der Kodierung unterworfen ist, zeigen sich Wärmeabfuhrprobleme. Bei einer verlängerten Videostreaming-Session, was normalerweise bei einer Türsprechanlage nicht der Fall ist, könnte der Raspberry zu einem Absturz bringen.

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

4.3.1 Alternative

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

I know that there is only partial hardware acceleration for vp8 on RPi but I thought on the right bitrate it may fit my requirements.

HW acceleration for 1080p video decoding in Linux

H.264 in WebRTC is only available on devices that support H.264 hardware acceleration.

Wieso wurde raspberry pi eingesetzt (Kompatibilität)

Konkurrenz Produkte (LattePanda) -> mehr leistung beim encoding/decoding Prozesse

4.4 PoE

Moderne Hausalte werden meistens mit ethernet Verkabelung verlegt. Ziel des Aussensprechstelle ist die Installationskosten zu senken und die Montage zu vereinfachen. Drei Anschlüsse werden von den Aussensprechstelle benötigt um sein Ziel zu erreichen und zwar Strom, Internetverbindung und eine Leitung der den Türöffner betätigt. Alle diese Fünktionalität können in einem Kat 7 Ethernet Kabel zusammengeführt werden.

Cisco Catalyst 3560g welcher für den PoE Stromversorgung zuständig ist verwendet das Phantomspeisung oder Mode A. Das heisst dass die mit Datenübertragung belegten Adern mit der Stromversorgung überlagert werden. Diese ist möglich da Elektrizität eine niedrige Frequenz von 60 Hz hat und 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 6: Catalyst 3560g PoE Pinbelegung

Wie im Abbild nr.5!!? 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 er-

hebliche kleinere Bandbreite in Anspruch nimmt, stellt für die Aussensprechtellen kein Hinderniss dar.

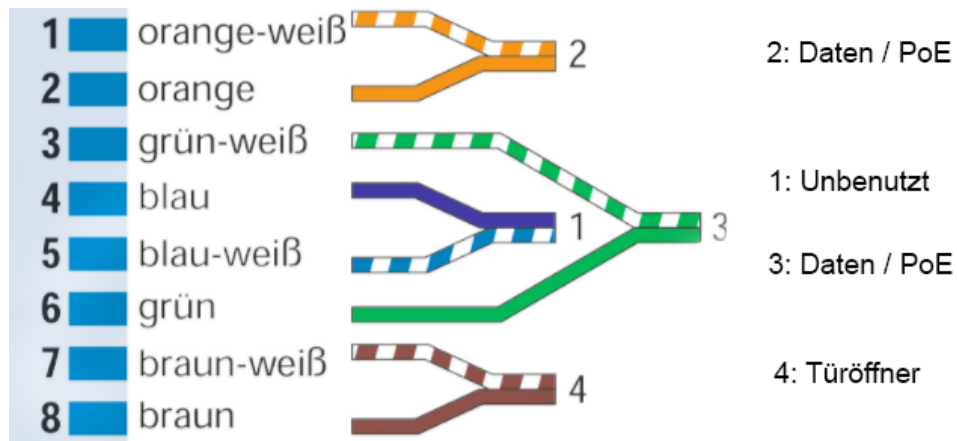


Abbildung 7: Cat. 7 Ethernet Pinbelegung für die Aussensprechtellen

5 Software

5.1 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.

5.1.1 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.

5.1.2 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.

5.1.3 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.

5.2 System Übersicht

Das System besteht aus mehrere Hardware- und Softwarekomponenten die zusammenarbeiten müssen (siehe Abb. 8). 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.

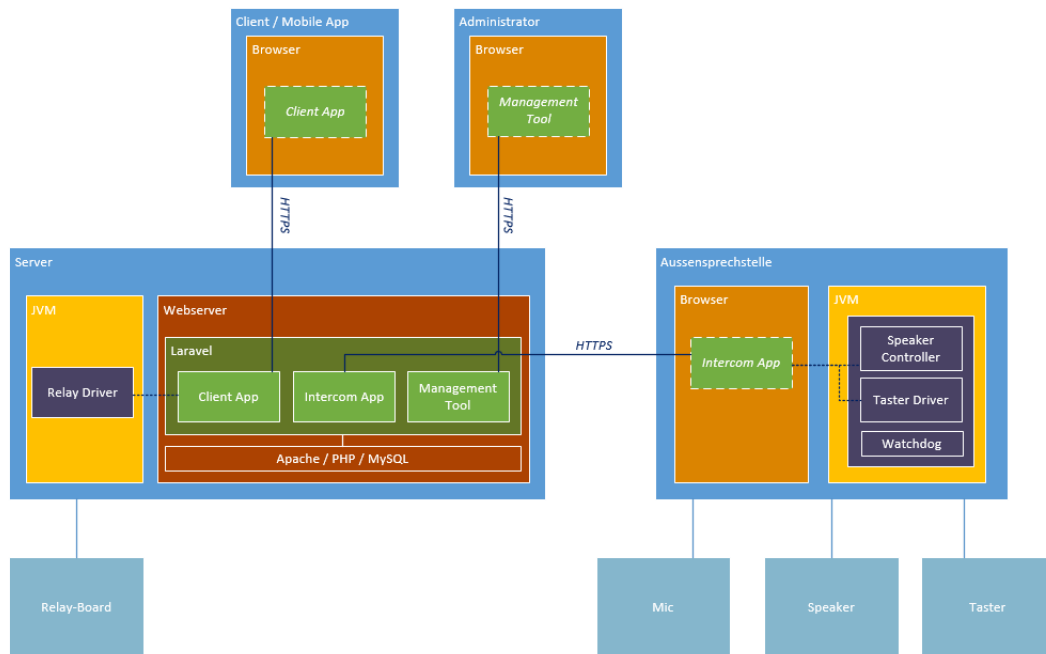


Abbildung 8: Software / Hardware Ecosystem

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. Diese hat eine gewisse Komplexität und wird in ein eigenes Kapitel (siehe Abschnitt 5.10) behandelt.

5.3 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.

5.4 Dienste

5.4.1 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.

5.4.2 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.

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 von der Aussensprechstelle-Applikation (*localhost*). So kann den Lautsprecher bei Bedarf ein- und ausgeschaltet werden.

5.4.3 Relay Controller

...

5.4.4 Signaling Server

Der Signaling-Server ist ein bestandteil von WebRTC und wird in ein eigenes Kapitel ausführlich beschrieben (siehe Abschnitt 5.10.1).

5.5 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 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.

5.6 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 erkannt. 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.

5.7 Webapplikationen

5.7.1 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. 9 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 Vordergrund. Aus diesem Grund werden die Tasten für die Audio-Kommunikation und für die Öffnung der Türe gross Angezeigt. Das Videostream von der ausge-

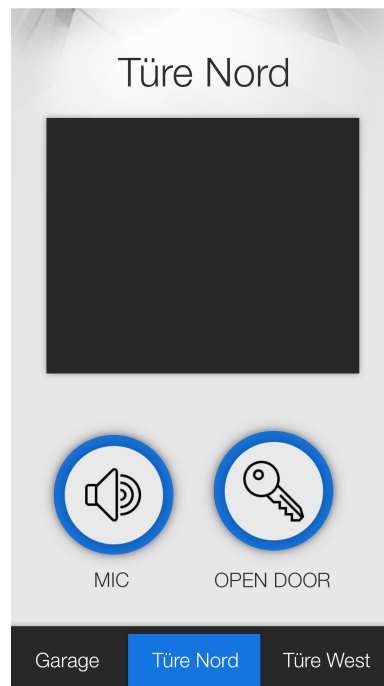


Abbildung 9: Design der Client-Webapp

wählte Türe wird sofort angezeigt und benötigt keine weitere Interaktion.

5.7.2 Aussensprechstelle Webapplikation

..

5.8 Management Tool

Das Management Tool wurde entwickelt um die Installation und Konfigurazionsprozess zu vereinfachen. Der Hauswart beziehungsweise der Mieter soll über eine Schnittstelle verfügen um die verschiedene Bewohner/Türe editieren oder erfassen zu können. Diese Schnittstelle wurde mit webtechnologien entwickelt (HTML, PHP, Js) und wird zusammen mit den MySQL Database auf den lokalen Raspberry Server gehostet. Aus Sicherheitsgründen werden alle eingehende und ausgehende Verbindungen abhörsicher aufgebaut.

Grund dafür das einsetzen 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 zukünftige Weiterentwicklung des Produkt kann man, dank der Webtechnologien mit geringerem Aufwand das Tool skalieren bzw. neue Features hinzufügen.

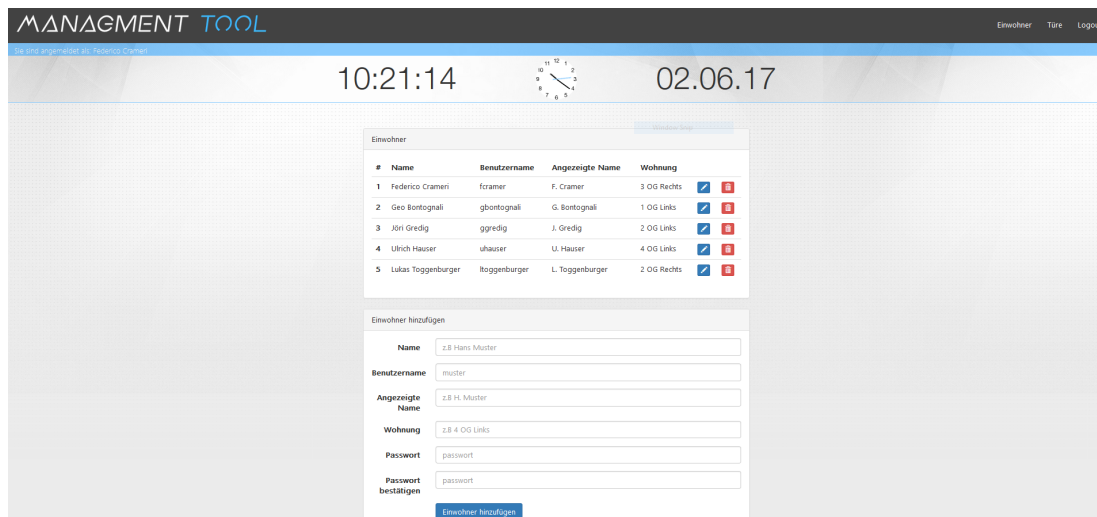


Abbildung 10: Design der Management tool

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

5.8.1 Bewohner

Unter der Bewohner-Seite werden alle Wohnungen, beziehungsweise alle Bewohner aufgelistet. Diese verfügen über einen Benutzernamen sowie ein Passwort, die von der Client App verwendet werden, um sie sich bei dem Server zu authentifizieren. Dieser Abschnitt bietet noch die Möglichkeit, den Namen und die Position der Wohnung, welche an der Aussensprechstelle ?? angezeigt wird, abzuändern.

5.8.2 Türen

Bei der Einbau einer neuen Tür, kann diese in dem Management Tool aufgeführt werden. Dabei muss beachtet werden, dass die ID mit derjenigen übereinstimmt, die auf der neu installierten Aussensprechstelle übereinstimmt. In dieser Sektion sind auch die Namen der Türen definiert, welche dann auf der Client App (Siehe Abbildung 10) angezeigt werden.

5.9 Remote Verbindung

..

5.10 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]

5.10.1 Signaling Process

Ähnlich wie bei VoIP-Telefonie (*SIP*), brauchen die Sprechpartner ein gemeinsam bekanntes Knoten, um die Verbindung zu initialisieren (siehe Abb. 11). 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

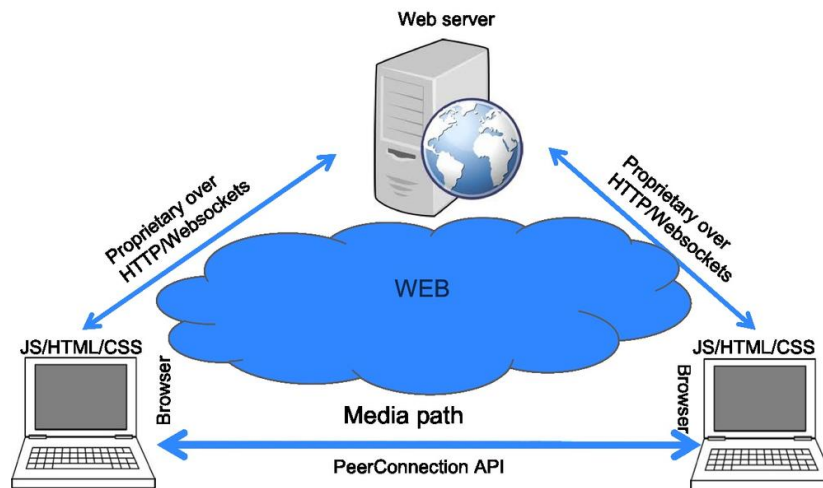


Abbildung 11: Der Signaling Prozess

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.

5.10.2 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. 12). 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-

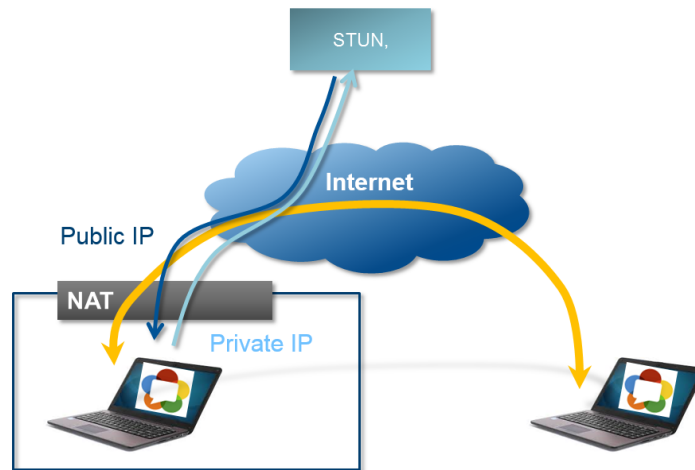


Abbildung 12: STUN Server

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

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	Catalyst Pinouts	8
7	EthernetPinbelegung	9
8	Software / Hardware Ecosystem	11
9	Design der Client-Webapp	15
10	Design der Management tool	16
11	Der Signaling Prozess	18
12	STUN Server	19

Tabellenverzeichnis

1	My caption	7
---	----------------------	---

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

