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

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 Einleitung

List Example:

- Maximaler Schub beim Starten z.B. wegen kurzer Startbahn
- Maximale Fluggeschwindigkeit erreichen z.B. bei Rettungsflügen oder Ähnlichem
- Maximale Effizienz z.B. um möglichst lange Flugzeiten zu ermöglichen

Bild Beispiel:

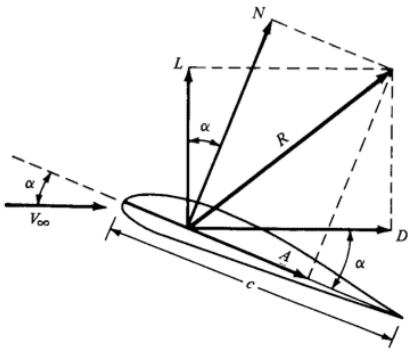

Abbildung 1: Wirksame aerodynamische Kräfte an einem Profil [**?**]. Die Auftriebskraft L wirkt beim Propeller als Schub und die Widerstandskraft D als Drehmoment.

Unterkapittel:

1.1 Subsection test

Wenn die Propellerberechnung beendet ist, werden alle Resultate in eine Baumstruktur gespeichert. Somit sind die Resultate schnell wieder aufrufbar und auch sauber in einer einzigen Datei geordnet und gespeichert.

1.1.1 Subsubsection test

Wenn die Propellerberechnung beendet ist, werden alle Resultate in eine Baumstruktur gespeichert. Somit sind die Resultate schnell wieder aufrufbar und auch sauber in einer einzigen Datei geordnet und gespeichert.

1.1.2 Subsubsubsection test

Wenn die Propellerberechnung beendet ist, werden alle Resultate in eine Baumstruktur gespeichert. Somit sind die Resultate schnell wieder aufrufbar und auch sauber in einer einzigen Datei geordnet und gespeichert.

2 Einführung

2.1 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 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.

2.2 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.

3 Projektplanung

3.1 Prozess

Als Entwicklungsprozess wird ein hybrides Vorgehensmodell eingesetzt (siehe Abb. 2). 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.

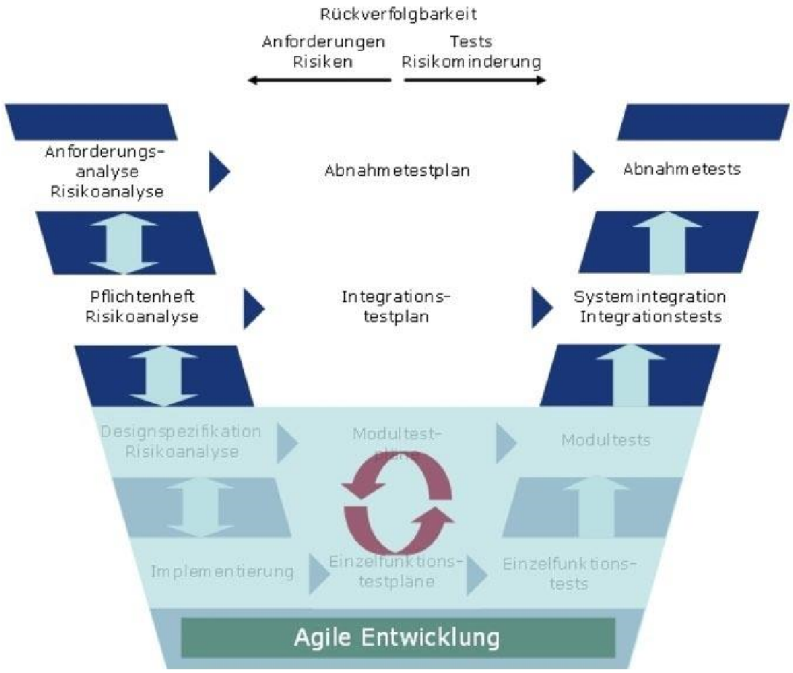

Abbildung 2: Hybrides Vorgehensmodell

3.2 Zeitplanung

Die folgenden Abbildungen stellen die Projektplanung und die Meilensteine zeitlich dar (siehe Abb. 3 & Abb. 4). 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.

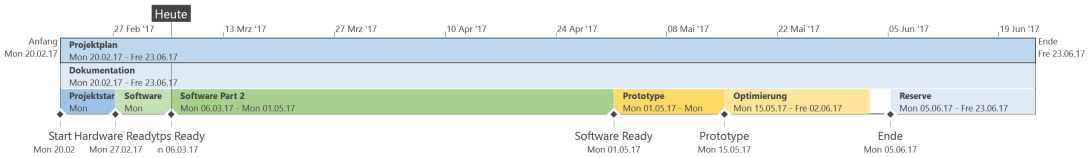

Abbildung 3: Zeitplanung mit Meilensteine

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 4: Projektplanung

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

4.3 PoE

Moderne Hausalte werden meistens mit ethernet Verkabelung verlegt. Ziel des Aussensprechstelle ist die Installationskosten zu senken und die Montage zu vereinfachen. Drei Anschlusse werden von den Aussensprechstelle benötigt um sein Ziel zu erreichen und zwar Strom, Internetverbindung und eine Leitung der für den Türöffner zuständig ist. Alle diese Fünktionalität können in einem Kat 7 Ethernet Kabel zusammengeführt werden.
Cisco Catalyst 3560g welcher für den PoE Stromversorgung zuständigt 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 hat eine niedrige Frequenz von 60 Hz und Datenübertragungen im bereich 10-100MHz liegt.

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 5: Catalyst Pinouts

Die einzige Nachteil bei dieser Konfiguration
<– ABBILDUNG CON IL CAVO ETHERNET

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 unterstüzt. 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. 6). 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.

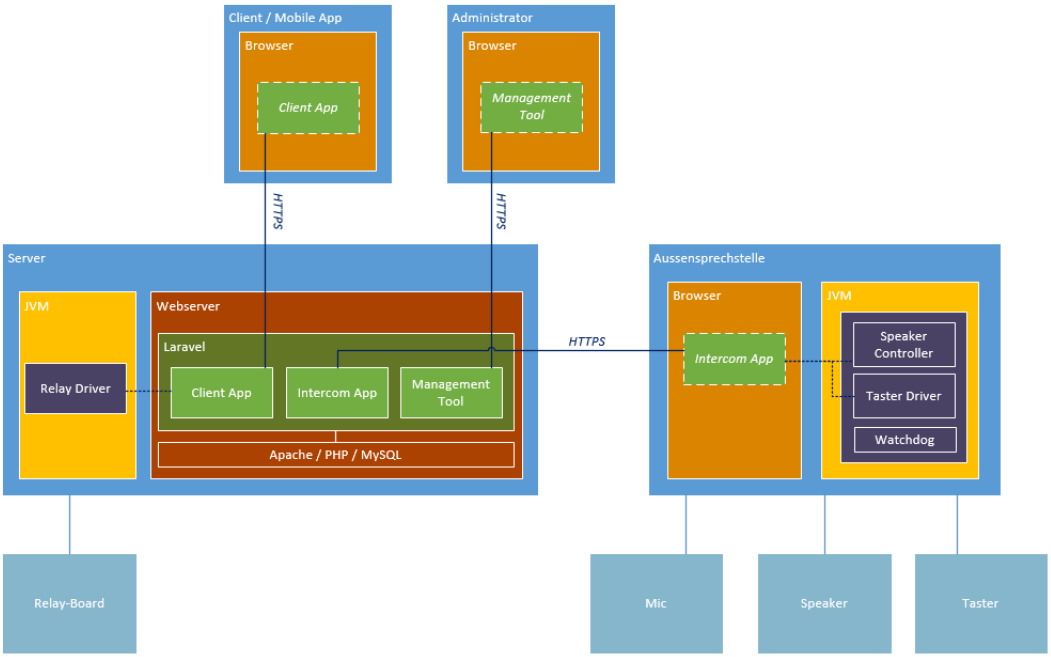

Abbildung 6: Software / Hardware Ecosystem

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 5.8) 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 hochoptimerte OS für die mit niedriger Leistung und geringem Stromverbrauch ARM Prozessoren. Der Raspbian Jessie Betriebssystem basiert auf Debian welch unter der DFSG (Debian Free Software Guidelines) Lizent 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.

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 5.8.1).

5.5 Logging

....

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 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.

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. 7 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 ausgewählte Türe wird sofort angezeigt und benötigt keine weitere Interaktion.

5.7.2 Aussensprechstelle Webapplikation

..

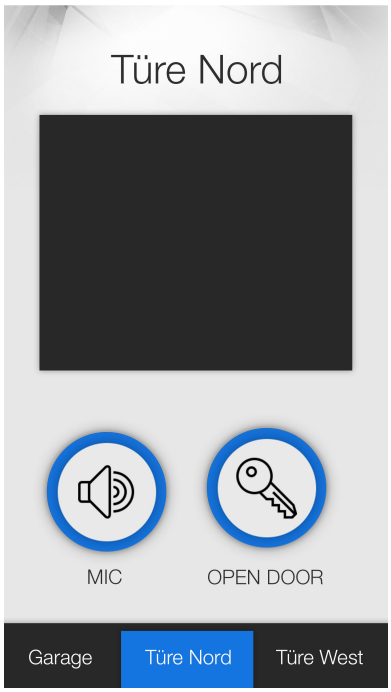

Abbildung 7: Design der Client-Webapp

5.8 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]

5.8.1 Signaling Process

Ähnlich wie bei VoIP-Telefonie *(SIP)*, brauchen die Sprechpartner ein gemeinsam bekanntes Knoten, um die Verbindung zu initialisieren (siehe Abb. 8). 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.

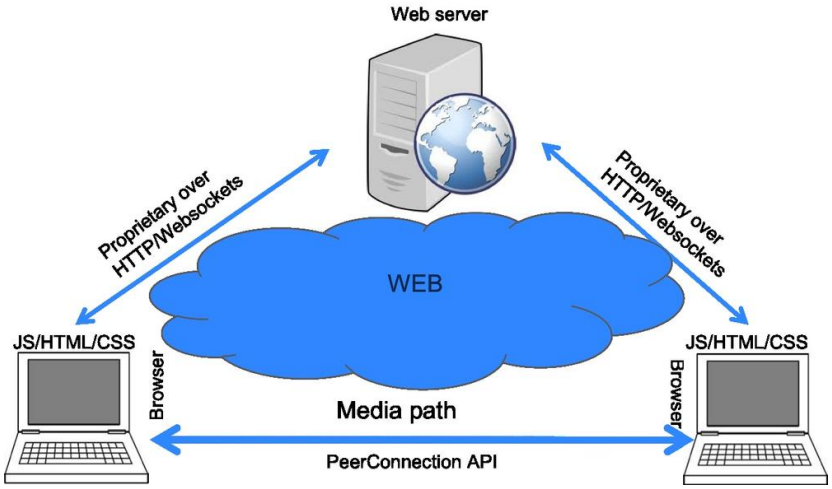

Abbildung 8: Der Signaling Prozess

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 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.

5.8.2 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 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. 9). STUN-Servers informie-

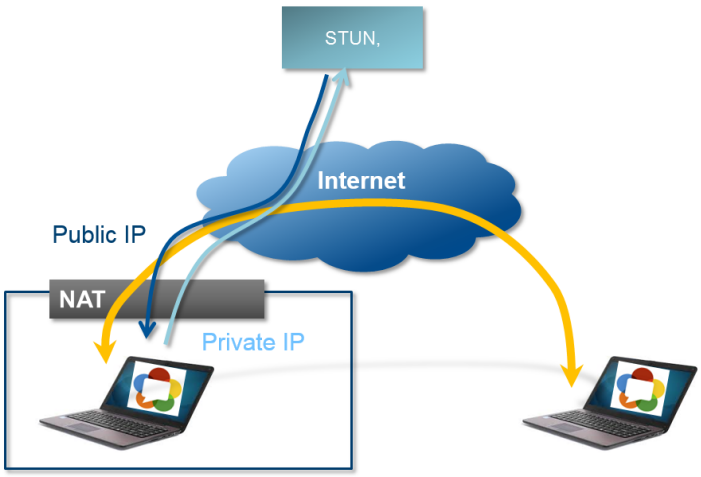

Abbildung 9: STUN Server

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

Literatur

Abbildungsverzeichnis

Tabellenverzeichnis

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