Gliederung der schriftlichen Arbeit - ALT

1. Einleitung
   1. Aufgabestellung und Rahmenbedingung
   2. Motivation
2. Grundlagen
   1. Stand der Technik
   2. Raspberry Pi
      1. Hardware
      2. Software
   3. Android Smartphone
      1. Android Betriebssystem
      2. Konnektivität
   4. Android Applikation
      1. Android Manifest
      2. Android Activity
      3. Android Ressource
      4. Android Layout
      5. Android View
   5. Lösungsansätze
      1. Android Applikation
      2. Übertragungsart
      3. Signalempfang und Auswertung
      4. Ausführung der Übertragene Signal
3. Vorbereitung
   1. Raspberry Pi
      1. Inbetriebnahme und Vorbereitung
      2. Werkzeugen
      3. Installation des Webservers
      4. Einrichtung des Webservers
   2. App Entwicklungsumgebung
      1. Anlegen des Projekts
      2. Einbindung Externe Library
      3. Einstellung der Version Kontrollsystem
4. Implementierung
   1. Raspberry Pi Software
      1. Kommunikation Komponente
      2. Interpreter Komponente
   2. Android App (Pi$Control)
      1. Oberfläche
      2. Kommunikation Komponente
5. Test
6. Schlussbetrachtung
   1. Erreichter Stand und Ausblick
   2. Fazit & Ausblick
7. Weiterentwicklung
8. Quellenverszeichnis
9. Anhang

Gliederung der schriftlichen Arbeit - NEU

Inhaltsverzeichnis

[1. Einleitung 6](#_Toc506881642)

[1.1. Problemstellung/Zielsetzung 6](#_Toc506881643)

[1.2. Motivation 7](#_Toc506881644)

[2. Grundlagen 8](#_Toc506881645)

[2.1. Stand der Technik 8](#_Toc506881646)

[2.2. Raspberry Pi 10](#_Toc506881647)

[2.2.1. Hardware 10](#_Toc506881648)

[2.2.2. Software 14](#_Toc506881649)

[2.3. Android Smartphone 16](#_Toc506881650)

[2.3.1. Hardware 16](#_Toc506881651)

[2.3.2. Software 16](#_Toc506881652)

[2.3.3. Android-Anwendung 19](#_Toc506881653)

[3. Lösungsansatz 24](#_Toc506881654)

[3.1. Konzept 25](#_Toc506881655)

[3.2. Entscheidungen über Implementierung 26](#_Toc506881656)

[3.2.1. Übertragung 26](#_Toc506881657)

[3.2.2. Raspberry Pi Module 28](#_Toc506881658)

[3.2.3. Android Anwendung 29](#_Toc506881659)

[4. Entwurf 29](#_Toc506881660)

[4.1. Pflichtenheft 30](#_Toc506881661)

[4.2. Raspberry Pi Anwendung 30](#_Toc506881662)

[4.2.1. API und Seine Bibliothek 30](#_Toc506881663)

[4.2.2. Kommunikation Schnittstelle 34](#_Toc506881664)

[4.3. Android Anwendung 35](#_Toc506881665)

[4.3.1. Steuerelement(*Control*) 36](#_Toc506881666)

[4.3.2. Anzeige von Steuerelement 38](#_Toc506881667)

[4.3.3. Signalübertragung 38](#_Toc506881668)

[4.3.4. Einfügen von Steuerelement (*Control*) 41](#_Toc506881669)

[5. Implementierung 47](#_Toc506881670)

[5.1. Raspberry Pi 47](#_Toc506881671)

[5.1.1. Inbetriebnahme 47](#_Toc506881672)

[5.1.2. WLAN Access Point: Einrichtung 48](#_Toc506881673)

[5.1.3. Webserver: Apache2 – Installation und Einrichtung 48](#_Toc506881674)

[5.1.4. Module 48](#_Toc506881675)

[5.2. Android Anwendung 49](#_Toc506881676)

[6. Test 51](#_Toc506881677)

[6.1. Test Aufbau 51](#_Toc506881678)

[6.2. Bewertung der Ergebnisse 53](#_Toc506881679)

[7. Erreichter Stand & Ausblick 54](#_Toc506881680)

[7.1. Erreichter Stand 54](#_Toc506881681)

**Abkürzungsverzeichnis**

API Application Programming Interface

HTTP Hypertext Transfer Protocol

XML Extensible Markup Language

GUI Graphical User Interface

GPIO General Purpose Input Output

RasPi Raspberry Pi

PHP Hypertext Preprocessor

NFC Near Fields Communication

ISR Interrupt Service Routine

WLAN Wireless Local Area Network

AP Access Point

CGI Common Gateway Interface

PWM Pulse Width Modulation

FQN Fully qualified Name

SDK Software development kit

# Einleitung

Eingebettet Systeme sind in den letzten Jahrzehnten sehr preiswert geworden. Dies beschreibt einen Computer, welcher in einem umgebenden technischen System eingebettet ist und mit diesem in Wechselwirkung steht. Der Computer übernimmt komplexe Steuerungs-, Regelungs-, Überwachungs- und Datenverarbeitungsaufgaben und verleiht damit dem umgebenden System oft einen entscheidenden Wettbewerbsvorsprung. Diese eingebetteten Systeme haben sich heutzutage in allen Bereichen der Technik etabliert wie zum Beispiel in der in der Industrie, Automobilindustrie, Medizin und in der Telekommunikation. In Moderne Personenkraftwagen der Oberklasse sind beispielsweise zwischen 70 und 80 integrierte und miteinander vernetzte Steuergeräte enthalten (1). Eingebetteten System erfordern in der meistens fällen ein Bedienungs- oder Steuerungssystem.

Heutzutage besteht ein Trend eingebettete Systeme mit Smartphones zu steuern. Dies kann besonders in der Hausautomatisierung und Robotik beobachtet werden. In der Robotik-Branche wird die Steuerung von Drohnen, Saugrobotern und Quadrocoptern mit Smartphones immer beliebter. Des Weiteren können, solange eine Internetverbindung besteht, Haushaltsgeräte mit eingebetteten Systemen unterwegs oder zuhause mithilfe eines Smartphones gesteuert werden. Bei einem Smartphone handelt es sich um ein Mobiltelefon, dass neben dem Telefonieren noch zahlreiche weitere Computertechnische Funktionen aufweist.

Die Steuerung von eingebetteten Systemen mit Hilfe des Smartphones erfolgt über vordefinierte Steuerelemente. Dies hat den Nachteil, dass bei der Entwicklung eines eingebetteten Systems, die Steuerung neu zu Implementierung ist. Um die Entwicklung des Steuerungssystems zu erleichtern, ist es sinnvoll, Ein allgemein einsetzbares Steuerungssystem bereitzustellen. Diese Bachelor Arbeit umfasst den Entwurf und die Implementierung eines Solchen Systems für den Raspberry Pi. Sie dient auch als ein Leitfaden, damit Nachbau und Weiterentwicklung ermöglicht wird.

# Problemstellung/Zielsetzung

Ziel dieser Bachelorarbeit, ist die Entwicklung einer universellen Android Applikation für die Bedienung eines Raspberry Pi basierten eingebettet Systems. Die Applikation überträgt Signale zum Raspberry Pi über eine drahtlose Schnittstelle. Der Raspberry muss vorher mit einer kommunikationsschnittstelle und einer API ausgestattet werden.

Aufgabe der Kommunikationsschnittstelle ist das Empfangen und Weiterleiten von Signalen zur Benutzer Anwendung. Letzteres ist die Nutzer Baustelle, die Ihm ermöglicht durch einbinden der API auf übertragene Signale zuzugreifen und die auszuführende Aktion selber zu bestimmen. Die Schnittstelle, die API und die Benutzer Anwendung sollen möglich getrennt sein. Die gleiche Philosophie soll auch bei der Android Applikation verwendet werden.

Die Android App besteht aus einer konfigurierbaren graphische Oberfläche und einer Kommunikationsschnittstelle. Die Oberfläche ist das Benutzerfenster; sie biete unter anderen Steuerelemente zur Verfügung. Der Nutzer soll je nach Wunsch Steuerelemente einfügen oder löschen können. Beim Einfügen bestimmt er die Eigenschaften des Elements. Aufgabe der Kommunikationsschnittstelle ist die Übertragung von Steuerelemente Signale an dem Raspberry Pi. Eine weitere Aufgabe ist das Mitteilen von Raspberry Pi Ausgaben an der Oberfläche.

Die Module, die oben erläutern wurden sollen sowohl auf dem Android Smartphone und auf dem Raspberry Pi nicht vermischt werden. Zweck ist, dem Nutzer eine Flexibilität zu bieten und Ihm den Austausch zu erleichtern.

Das Bedienungssystem (Die Applikation, die Raspberry Pi Schnittstelle und die API) soll einen idealen Einstiegspunkt für die Entwicklung eines Steuerungssystems für eingebettet System mit Raspberry Pi über Android Smartphone bieten.

Weiterhin werden alle Schritte beim Entwurf, sowie der Implementierung dokumentiert, damit zukünftige Nachbauten und Erweiterungen unkompliziert gemacht werden können.

# Motivation

Die Motivation sich mit dem Thema dieser Bachelorarbeit auseinanderzusetzen, ist die Anwendung und die Erweiterung der während der Studienzeiten erworbenen Wissen in der Soft- und Hardware. Es wird Praktisch umgesetzt, wie man ein Projekt von der Grundidee bis hin zur Implementierung einschließlich Test gestaltet. Die Hauptvoraussetzung für die Implementierung der Android Applikation, sind die Kenntnisse der im Fach „Projekt Software Entwicklung“ kennengelernte Android Framework. Die Entwicklung der Raspberry Pi Module benötigt Kenntnissen vom Raspberry Pi Umgebung, sprich Linux Betriebssysteme und seine bekannten GPIO-Pins und deren Programmierung. Grundlagen aus der Elektrontechnik werden beim Test benötigt. LEDs werden an den GPIO-Pins angeschlossen, um die Funktionalität der Applikation zu testen.

Von der Grundidee bis zur Implementierung und Test wird nicht nur das im Studium erworbene Wissen benötigt, sondern auch die Ergebnisse aus eigener Recherche eingesetzt um diese Bachelorarbeit nach besten Wissen und Gewissen abzuschließen.

# Grundlagen

Dieses Kapitel befasst sich mit den verwendeten Komponenten für das Thema dieser Bachelorarbeit. Um das Ziel zu erreichen ist es wichtig, dass das Umfeld beschrieben und die Komponenten kennengelernt werden. Die Eigenschaften des Raspberry Pis und seine Schnittstellen werden näher betrachtet. Besonderes Augenmerk wird auf den „General Purpose Input Output“ (GPIO) gelegt. Schließlich werden Funkschnittstelle betrachtet. Nach dem Raspberry Pi wird Android Smartphone auch näher betrachten. Besonders werden hier die seine Funkschnittstelle, sein Betriebssystem, der Aufbau seiner Anwendungen beschrieben.

# Stand der Technik

Vor Anfang dieser Bachelorarbeit, wurde nach Projekten gesucht die das das Thema schon behandeln haben. In diesem Kapitel werden einige davon vorgestellt.

**PiRelay[[1]](#footnote-1)**

PiReplay ist eine Android App zur Steuerung von Raspberry Pi GPIO-Ports. Die Applikation ist für das Ein- und Ausschalten von GPIO-Pins und kann bis zu fünf Raspberry Pi steuern. Zweck der Entwicklung dieser App, ist die Steuerung Haushaltsgeräten wie Lichter, Ventilatoren, Motoren, Türen und Heizung. Davor muss aber der Raspberry Pi mit einem Webserver und ein PHP Skript ausgestattet sein. Die Oberfläche ist konfigurierbar und kann bis zu 100 Steuerelemente „Relay“ (vgl. Abbildung 1).

Wenn ein „Relay“ auslöst sendet es dem Webserver eine HTTP Anfrage über WLAN. Der Server bearbeitet die Anfrage und leitet die Informationen an das PHP Skript weiter. Die Informationen enthalten die Pin-Nummer und das Signal. Mittels dieser Informationen, schaltet das Skript der Pin Ein oder Aus (2).



Abbildung 1: Graphische Oberfläche PiReplay (2)

**BlueTerm[[2]](#footnote-2)**

BlueTerm ist eine Android App zur Kommunikation mit allen Bluetooths serieller Adapter (4). Das Projekt besteht aus BlueTerm und zwei Python Skripten. Das erste Skript ist zuständig für die Bluetooth Kommunikation und das zweite für die GPIO-Pins. Im Code ist schon festgelegt welche Pins angesprochen werden sollen.



Abbildung 2: Konsolenanwendung BlueTerm

Im Gegensatz zu PiReplay besitzt BlueTerm keine Graphische Oberfläche, sondern eine Konsole für die Eingabe. Durch eingeben von Ein (1) oder Null (0) in der Konsole kann der Pin Ein- oder Ausgeschaltet werden.

**Zusammenfassung**

Aus der Suche ergab sich, dass unsere Thema nicht neue ist. Es existieren bereits viele Projekte die dies behandelt haben; sie erfüllen jedoch nicht komplett unsere am Anfang festgelegte Voraussetzungen. Zum einen hat der Nutzer keinen direkten Zugriff auf die übertragenen Signale und kann sie also nicht nach seinem Wunsch implementieren und zum anderen sind Projektmodule nicht austauschbar, entweder weil der Quellcode nicht Open Source ist oder keine sichtbaren Grenzen zwischen den Komponenten existiert.

# Raspberry Pi

Raspberry Pi ist ein Einplatinencomputer in Kreditkarten Format. Er wurde von Videospiel-schöpfer David Braden für die Raspberry Pi Stiftung entwickelt.

Die Intention war es, ein günstiges und einfach zu programmierendes Produkt zu entwickeln. Der Raspberry Pi kostet ungefähr 35 Euro und bietet mit seinem günstigen Preis Leuten, die nicht über die finanziellen Mittel verfügen und sich fürs Programmieren interessieren und begeistern, die Möglichkeit einen Computer zu kaufen und sich darin einzuarbeiten (5).

Das Thema dieser Bachelorarbeit wurde mit dem Raspberry Pi 3 Modell B bearbeitet. Es ist seit Februar 2016 verfügbar. Im Lieferumfang befinden sich der Computer und der Stromadapter. Weitere Zubehör muss man sich extra kaufen.

# Hardware

In diesem Kapitel werden physische Komponente des verwendet en Modells näher kennengelernt. Besonderes Augenmerk wird auf seine GPIO-Ports und seine Funkschnittstelle gelegt.



Abbildung 3: Komponentenübersicht des Raspberry-Pi 3 Modell B (5)

Maße (Länge x Breite x Höhe): 85,6 mm x 56,0 mm x 20mm

Gewicht: 40g

SoC: Broadcom-BCM2837

CPU

* Typ: ARM Cortex-A53
* Kerne: 4
* Takt: 1200 MHz
* Architektur: ARMv8-A (64 Bit)

GPU: Broadcom Dual Core Videocore IV

Arbeitsspeicher: 1024 MB

Netzwerk

* Ethernet: 10/100 Mbit/s
* WLAN: Broadcom BCM43143 2,4 GHz WLAN b/g/n
* Bluetooth: 4.1 Low Energy

Schnittstellen

* GPIO-Pins: 40
* CSI, DSI, I²C, SPI, UART, microSD-Slot
* 4 x USB 2 Port
* DSI Display Port
* CSI Kamera Port

Videoausgabe: HDMI (Typ A), Composite Video

Audioausgabe: HDMI (digital), 3,5-mm-Klinkenstecker (analog)

Betriebsspannung: 5 Volt Micro-USB-Anschluss (Micro-USB-B)

**GPIO-Pins**

Sie sind auf Abbildung 3 mit der Nummer 1 repräsentiert. Sie werden verwendet, um Peripherie Geräte oder Messinstrumente anzuschließen, um Daten zu verarbeiten oder Funktionen zu steuern. Die Funktionen dahinter ermöglichen es Signale mit High- und Low-Pegel zu generieren, um einfache Bauelemente, Wie LEDs anzusteuern. Sie bieten auch Pulsweitenmodulation (PWM) an, um eine LED zu dimmen oder einen Motor mit unterschiedlich schnellen Radumdrehungen anzusteuern.

Das Model 3 B besitzt 40 Pins. Sie sind in zwei Reihen mit jeweils 20 Pins mit einem Abstand von 2,54 mm angeordnet; 28 davon können als GPIO verwendet werden. Die restlichen Pins sind entweder für einen Erdanschluss oder für die Stromversorgung. Zu den Stromversorgunganschlüssen zählen die 5 V und die 3.3V (7).



Abbildung 4: Raspberry Pi GPIO-Pins (7)

Es gibt zwei Nummerierungssysteme zur Bezeichnung der Pins. Das BOARD System bezieht sich auf die Physischen Position Pins auf dem Board. Die Nummern gehen von 1 bis 40 und der Pin mit Nummer 1 steht direkt neben die Bezeichnung J8 auf der Platine. Das BCM System bezieht sich auf die offizielle Dokumentation des auf dem Raspberry Pi verbauten BCM837-Chips. Es ist auf Abbildung 4 mit der Bezeichnung „GPIOxx“ repräsentiert. wobei „xx“ die Nummer bezeichnet.

Ferner haben die Entwickler eine weitere Bezeichnung eingeführt, welche in der Abbildung 4 nicht weiter deklariert ist, aber zum Teil den Pin in seiner Funktion ausweist. Zur Illustrierung sind die Pins farblich in Gruppen markiert, damit Zusammenhängende Funktionen der Pins deutlicher ausgemacht werden können.

Der maximale Strom ist 50 mA bei den 3.3V Pins und 1 A bei den 5.5V Pins. Die 5V Pins sollten sorgfältig behandelt werden, da sie das Board beschädigen können, wenn Sie mit andere Pins direkt verbinden sind. Sie sollten am besten vor jeder Manipulation isoliert werden.

Die Software Umsetzung der GPIO Pins ist in verschiedenen Programmiersprachen möglich. Python und C zählen zu den bekanntesten. Es existiert bereit Bibliotheken in diesen Sprachen, die die Implementierung erleichtern; beispielweise RPi.GPIO, Piggpio und WiringPi.

Es ist wichtig zu wissen, dass die Raspberry Pi GPIO für Echtzeit- oder Zeitnahreagierende Systeme nicht geeignet sind, da das Betriebssystem jederzeit einem anderen Prozess priorisieren kann.

**Funkschnittstellen**

Das Model 3 B beinhaltet eingebautes Bluetooth und WLAN.

Der Bluetooth Chip arbeitet mit BTLE, auch bezeichnet als Bluetooth 4.1 Low Energy. Damit bringt der Raspberry Pi 3 Model B.

Das WLAN Modul unterstützt die Standards 802.11b, g und n und arbeitet im 2.4 GHz Band (BCM43143). Das Modul hat eine maximale übertragungsrate von 150 Mbit/s (8).

# Software

**Betriebssystem**

Das Betriebssystem ist eine Zusammenstellung von Computerprogrammen, die Systemressourcen eines Computers wie Arbeitsspeicher, Festplatten, Ein- und Ausgabegeräte verwaltet (8). Es gibt zwei Hauptkategorien vom Betriebssystem für den Raspberry Pi: Die eingebetten und die Mehrzwecksysteme.

Die Eingebetten sind für bestimmte Zwecke entwickelt. Sie besitzen meistens keine graphische Oberfläche. Durch Ihre geringe Anforderungen in Bezug auf Energiebedarf sowie Arbeits- und Massenspeicher eignen sich eingebettete Betriebssysteme für Zeitnah reagierende System wie Industrieanlagen, Drohnen, Antiblockiersystem und Roboter. Ein Beispiel ist Minoca OS (10).

Die Mehrzwecksysteme dagegen brauchen viel Speicher und Energie und haben längere Reaktionszeit. Sie haben aber den Vorteil, dass sie einfacher zu bedienen sind und meisten eine graphische Oberfläche besitzen. Im Internet sind mehrere Mehrzwecksysteme zu finden. Die Üblichen sind Ubuntu, Noobs und Raspbian.

Das beliebteste Betriebssystem ist der kostenlose, auf Debian Linux basierte und für Raspberry Pi Hardware optimierte Raspbian. Die Beliebtheit entsteht durch die kostenlose Lizenz, die schnelle Leistung, und die vorinstallierte Software und Tools.

Raspbian wird offiziell unterstützt von der Raspberry Stiftung und beinhaltet mehr als 35.000 Software Pakete. Dazu zählen unter anderen Textverarbeitungsprogramme wie LibreOffice, Browsers wie Chromium und Firefox, Entwicklungsprogramme wie Geany und BlueJ.

**Anwendung**

Hier werden die Programmiersprachen der Raspberry Pi Anwendungen erläutert.

Der Name des Raspberry Pi knüpft an der Tradition an, Computer nach Früchten zu benennen. Bekannte Vertreter sind Apple und Blackberry. Der Zusatz „Pi“, ausgesprochen wie das englische Wort „Pie“, übersetzt für Tortenstück, wird oft fälschlicherweise als Solches interpretiert, steht jedoch für Python Interpreter, eine Andeutung für die Hauptprogrammiersprache des Raspberry Pis.

Bei Python handelt es sich um eine Höhere Programmiersprache, die sich durch ihre Komplexität von der Maschinensprache entfernt. Erst der Einsatz von Interpreter oder Compiler übersetzt Befehle des Programmiercodes in Maschinensprache, die der Mikroprozessor bzw. Mikrocontroller versteht.

Der Raspberry Pi kann aber nicht nur mit Python programmiert werden. Er unterstützt auch Scratch, C, C++, Java, Perl, HTML5 und Skriptsprachen wie, PHP, JavaScript. Da der Raspberry Pi auf einem Linux-Kernel operiert, können über die angebotenen Paketquellen nach Belieben weitere Programmiersprachen hinzugefügt werden.

# Android Smartphone

Wie eingangs erwähnt, ist ein Smartphone ein Mobiltelefon, das umfangreiche Computer-Funktionalitäten und Konnektivitäten zur Verfügung stellt. Bei Android Smartphones handelt es sich um Smartphones, die mit dem Android Betriebssystem ausgestattet sind.

# Hardware

Wegen der Vielfalt von Android Smartphones, können ihre physische Komponenten nicht genau beschrieben. Laut Statista[[3]](#footnote-3) befindet sich weltweit bereit 2,58 Mrd. Android Smartphone (Stand 2017) im Gebrauch. Sie sind je nach Hersteller und Modell unterschiedlich gebaut. Meistens sind sie unter anderen mit Prozessor, Speicher, Touchscreen, Batterie, Kamera, WLAN und Bluetooth ausgestattet.

Moderne Smartphone besitzen neue Technologie wie NFC, A-GPS, Glonass, und Galileo. Sie sind auch mit vielen Sensoren ausgerüstet wie Gyroskope, Barometer und Magnetometer (8).

# Software

**Betriebssystem**

Android ist das Open-Source-Betriebssystem von Google, das die meistens Smartphones und Tablets auf dem Markt antreibt. Die erste Android-Version kam im September 2008 auf dem Markt (10). Es besitzt heutzutage weltweit den Größten Marktanteil von Smartphone Betriebssystemen.



Abbildung 5: Weltweiter Marktanteil der Smartphone-Betriebssysteme (13)

Das Betriebssystem beinhaltet vier Schichten und Jede abstrahiert die darunterliegende. Es besteht aus einem angepassten Linux-Kernel. Das Kernel bildet die Schicht zwischen der Hardware und der restlichen Architektur. Er enthält unter anderem Hardwaretreiber, Prozess- Speicherverwaltung und Sicherheitsverwaltung (siehe Abbildung 6).



Abbildung 6: Architektur Android (11)

Da Android ein Multiprozess Betriebssystem ist, läuft jede Anwendung in ihrer eigenen Instanz der virtuellen Maschine. Dieser Umstand ist vor allem für die Sicherheit und die Anwendungsentwicklung von Bedeutung. Die Virtuelle Maschine wurde speziell dafür entwickelt mehrere virtuelle Maschinen effizient parallel ausführen zu können. Die ausgeführten Dateien sind in einem für minimalen Speicherverbrauch optimierten dex-Format. Dieses Format wird, nachdem es von Java kompiliert wurde, mit dem dx-Werkzeug in das nötige Format umgewandelt. Die Virtuelle Maschine benötigt weniger Zwischenschritte um den Bytecode auf dem Prozessor auszuführen, weil das Register nicht stapelbasiert ist. Die Maschine nutzt die Low-Level-Speicherverwaltung und das Threading des optimierten Kernels aus (12).

Das Anwendungsframework ist die Schicht, die für die Entwicklung von Android Applikationen von Bedeutung ist. Sie stellt den Rahmen für eine einheitliche Anwendungsarchitektur bereit. Ziel ist es, Anwendungen nach einheitlichen Richtlinien zu entwickeln und somit die Integration und Wiederverwendung von Anwendungen unter Android zu vereinfachen (12). Es bietet Entwicklern Elementen und Methoden zur Erstellung von GUIs oder Nutzung von Ressourcen an. Hintergrunddienste, Nachrichtenaustausch mit anderen Applikationen und das Nutzen von Hardware Ressourcen wird ermöglich. Der Zugrifferlaubnis auf Ressourcen muss ebenfalls bei Installation der Applikation vom Nutzer akzeptiert werden. So werden unerlaubte Zugriffe vermieden.

Das Android-Betriebssystem beinhaltet grundlegende Apps wie E-Mail, SMS-, Kalender-, Karten-, Kontakte-, und Browseranwendungen. Diese Anwendungen wurden mit der Programmiersprache Java implementiert.

# Android-Anwendung

In diesem Kapitel werden die wichtigsten Komponenten eine Android-Anwendung und deren Zusammenhang kennengelernt. Die App besteht grundsätzlich aus einem Manifest, den Java Code und Ressourcen. Der Java Code beinhaltet Activities. Ressourcen bestehen aus Layouts, und notwendige Bestandteile des Programms wie Zeichenfolgen, und Binärdateien. Der Java Code und die Ressourcen sind im Projekt unter den Ordner java beziehungsweise res zu finden. Der Java Code beinhaltet die Programmlogik, die Ressourcen dagegen die graphische Darstellung.



Abbildung 7: Aufbau Android-Anwendung

**Android Manifest**

Es beinhaltet grundlegende Informationen, die das System braucht, um die Applikation zu starten. Drinnen befinden sich Angaben über Komponenten und Zugriffsrechte der Applikation (siehe Abbildung 8). Zu den Komponenten zählen Activities und Hintergrund-Services. Die Manifest-Datei ist XML-formatiert.



Abbildung 8: Beispiel Android Manifest

**Activity**

Eine Android-App besteht wesentlich aus Activities (3). Sie ist ein Fenster mit dem der Nutzer mittels Tasten- oder Touchscreen Berührungen interagiert. Das Aussehen der Benutzerfenster wird mit Layouts definiert. Es kann immer nur eine Activity gleichzeitig aktiv sein, bei einem Wechsel wird die zuvor laufende pausiert. Das Aufrufen erfolgt über Intents[[4]](#footnote-4). Intents sind Objekte, die Informationen über eine Operation beinhalten und das Bindeglied zwischen mehreren Komponenten von Android bilden.

Android greift im Vergleich zu anderen Betriebssystemen stärker in den Lebenslauf einer Anwendung ein. Aus dem Benutzerverhalten und dem Activity-Management von Android resultieren für eine Aktivität verschiedene Zustände, wobei die Übergänge durch den Aufruf bestimmter Methoden gekennzeichnet sind (16).



Abbildung 9: Zustandsdiagramm einer Android Activity (16)

Das aufrufen dieser Methoden wird bis auf die *onCreate-*Methode vom System übernommen. Beim Starten einer Anwendung und somit der ersten Activity, werden gleich drei Methoden Aufgerufen: *onCreate, onStart,* und *onResume* (siehe Abbildung 9)*.* Die wesentliche Logik der Activity ist in der *onCreate*-Methode implementiert. Dawird auch mithilfe von der *setContentView*-Methode bekannt gegeben, welches Layout (GUI) zu der Activity gehört (siehe Abbildung 10). Dieser Methode muss ein Layout übergeben werden.



Abbildung 10: Deklaration von Activity in Java

**Layout**

Ein Layout definiert die Struktur einer GUI. Es gibt verschiedene Typen von Layouts: LinearLayout, RelativeLayout, TableLayout, GridLayout*.* Der wesentliche Unterschied ist die Anordnung der GUI Elementen (Views). Ein LinearLayout positioniert beispielweise Elemente in einer einzigen vertikal oder horizontal Reihe. Zu jedem Layout-Typ gehört eine Java-Klasse.

Layouts können sowohl in XML als auch in Java deklariert werden. Android bietet ein einfaches XML Vokabular zur Erstellung von Layouts. Die Deklaration in Java erfordert die Vererbung der Klasse vom Layout-Typ. Die Positionierung des GUI Elements und die Große des Layouts werden mithilfe der LayoutParamsbestimmt (17). Layouts können auch verschachtelt sein.



Abbildung 11: Deklaration vom RelativeLayout in Java

**View**

Views sind Steuerelemente der Android GUI. Sie reagieren auf Touch- oder Tatstatur-Ereignissen und ermöglichen somit dem Nutzer mit der Anwendung zu kommunizieren. Android bietet verschiedene Typen von Views, die sich von ihrem Aussehen und ihrer Bedienungsform unterscheiden. Zu jedem View-Typ gehört eine Java-Klasse.



Abbildung 12: Views Klassen in Android (18)

Views können in XML oder in Java (siehe Abbildung 13) deklariert werden. Die Gestaltung von Ereignissen erfolgt über Interfaces und Methoden von View-Klassen.



Abbildung 13: Deklaration vom Android View in Java

# Lösungsansatz

Dieses Kapitel umfasst die Entscheidungen über Aufbau und Entwurf der Android Applikation und der RasPi Anwendung. Die Entscheidungen beziehen sich auf die Grundlagen und das damit einhergehende, erlangte Wissen. Der Komplette Aufbau wird mithilfe von Bildern und Graphen skizziert.

# Konzept

Die wesentliche Herausforderung dieser Bachelorarbeit ist die Entwicklung eines modular aufgebauten und universellen (wiederverwendbaren) Steuerungssystem. Beim Entwurf wurde auf diesen Kriterien besonders geachtet und ein Konzept entwickelt, das diese erfüllt.

**Modularität**

Die Modularität (auch Baustein- oder Baukastenprinzip) ist die Aufteilung eines Ganzen in Teile, die als Module, Komponenten, Bauelemente oder Bausteine bezeichnet werden und zusammen interagieren können (19). Um die Modularität des Systems zu gewährleisten, wurden Vollständig implementierten Module definiert. Bei Module handelt es sich, um Elemente, die als Bestandteil eines Systems einen bestimmten Zweck erfüllen.

Die Android Applikation besteht aus zwei Modulen: die Benutzeroberfläche (GUI) und die Kommunikationsschnittstelle. Die Benutzeroberfläche beinhaltet Steuerelemente. Ein Steuerelement beschreibt eine View, die durch Touch-Ereignisse Signale and den RasPi sendet. Die Kommunikationsschnittstelle sendet Signale über die Funkschnittstelle an den RasPi und leitet seine Nachtrichten an der Benutzeroberfläche weiter.

Die RasPi Anwendung besteht aus der Kommunikationsschnittstelle und der API. Die Kommunikationsschnittstelle steuert den Empfang und die Weiterleitung von Signalen an die Nutzer Anwendung. Nachrichten an die Android Applikation werden ebenfalls über die Kommunikationsschnittstelle übertragen. Die API ist die Schnittstelle zwischen den Signalen und der Nutzer Anwendung. Sie bietet dem Nutzer Methoden zur Verwaltung von Signalen und zur Erstellung von Nachrichten, die an die Android Applikation versendet werden. Die API ist in Nutzer Anwendung einzubinden.

**Universalität**

Im Kontext dieser Bachelorarbeit besteht die Universalität darin, die App für unterschiedliche Anwendungsszenarien einzusetzen. Die universelle Einsetzbarkeit des Bedienungssystems ist durch die API gewährleistet, die dem Nutzer die Möglichkeit bietet die Signale individuell zu implementieren.



Abbildung 14: Aufbau des Steuerungssystems

# Entscheidungen über Implementierung

Wir haben sowohl bei dem Android-Smartphone, als auch bei dem Raspberry Pi erfahren welche Funkschnittstelle sie anbieten, welche Programmiersprachen sie unterstützen. In diesem Kapitel werden wir begründete Entscheidungen für den Entwurf treffen.

# Übertragung

Bei der Signalübertragung geht aus um zwei Aspekte: Die Schnittstelle und der Übertragungsmodus.

**Schnittstelle**

Das Android-Smartphone und der RasPi besitzen Bluetooth und WLAN. Die Übertragung kann über eine der beiden Schnittstellen erfolgen. Beide Schnittstellen besitzen Vor- und Nachteile.



Abbildung 15: Vergleich Bluetooth und WLAN (20)

Die Entscheidung hängt von den Anforderungen an das System und den Zielen des Nutzers ab.

Für diese Arbeit wurde die WLAN-Schnittstelle verwendet. Die Gründe dafür sind die höhere Reichweite und die Unterstützung mehrerer Geräte. Die in Abbildung 15 erwähnte Nachteile der WLAN-Schnittstelle treffen in unserem Kontext nicht zu, da sowohl das Smartphone, wie auch der RasPi integriertes WLAN besitzen.

**WLAN-Übertragungsmodus**

Die WLAN-Schnittstelle bietet wiederum zwei übertragungsarten: Wi-Fi Direct WIFI Access Point.

Wi-Fi Direct ist ein Standard zur Datenübermittlung zwischen zwei WLAN-Endgeräten ohne zentralen Wireless Access Point (21). Jedoch unterstützen nicht alle Android-Smartphones diese Technologie und laut (22) unterstützt der im Raspberry Pi Modell 3 B integrierte WLAN-Modul diese Technologie nicht. Ein separates WLAN-Modul lässt sich allerding nachträglich hinzufügen.

Ein Access Point ist ein Gerät, das ein drahtloses lokales Netzwerk bereitstellt, mit dem andere Geräte (Clients) verbinden können. Das Hauptziel besteht häufig darin dem Client der Internetzugang zu ermöglichen. Clients können über das Netzwerk aber auch Nachrichten austauschen. In unseren Fall fungiert der RasPi als der Access Point und das Smartphone als Client. Das Integriertes WLAN-Modul des RasPi kann als Access Point konfiguriert werden und jedes Smartphone unterstützt den Client-Modus. Nachteil ist, dass der Client keinen Internetzugang erhält, wenn der Access Point keiner Internetverbindung aufbauen kann.

Bei dieser Bachelorarbeit wurde die Access Point-Variante verwendet. Diese Variante hat den Vorteil, dass sie mit vielen Geräten kompatibel ist. Ein weiterer Vorteil ist, dass mehrere Geräte sich gleichzeitig verbinden können. Dies ist beispielsweise für Anwendungen des Steuerungssystems im Bereich Hausautomatisierung von Vorteil.

# Raspberry Pi Module

Dieser Abschnitt umfasst die Entscheidung über Komponenten, die den Raspberry Pi umfassen. Es wird erläutern wie Signale empfangen werden sollen. Weiterhin werden Entscheidungen über die Programmiersprachen der Module getroffen.

* **Webserver**

Wie Oben erläutern, ist Hauptziel ein Access Point die Erstellung eines lokalen Netzes und Somit die Verteilung von Internetzugang. Es kann auch für andere Zwecke verwendet werden dafür braucht man aber der richtigen Software. Damit der Access Point Signale vom Client bearbeitet kann, muss er mit einem Webserver ausgestattet sein. Für diese Arbeit wurde sich für Apache Web Server[[5]](#footnote-5) entschieden. Die Signale werden dann von Android Anwendung als HTTP Anfrage gesendet. HTTP Anfragen ermöglichen generell den Client Informationen zum Webserver zu senden, um Formular abzusenden, Datei hochzuladen oder Dateien abzurufen (23). In unseren Fall werden wir Kommunikation Schnittstelle der Raspberry Pi mit Parametern aufrufen.



Abbildung 16: Beispiel HTTP Anfrage mit Parametern

* **Module**

„File\_to\_call.cgi“ auf Abbildung 16 ist die Schnittstelle auf dem Server, die die Anfrage Empfängt. Es handelt sich um ein CGI-Skript. Das Common Gateway Interface (CGI) ist ein Standard zum Schreiben von Programmen, die über einen Webserver mit einem Client interagieren können, auf dem ein Webbrowser ausgeführt wird. Grundsätzlich interpretiert, verarbeitet er die an ihn gesendeten Informationen und generiert die Antwort, die an den Client zurückgesendet wird. Meistens werden diese Informationen über Umgebungsvariable[[6]](#footnote-6) in das CGI-Skript eingegeben (24).

Wie in den Grundlagen erläutert, kann der Raspberry Pi Python, C/C++ und Java Programmiert werden. Außer C sind anderen Sprachen sogenannten Höhere Programmiersprachen. Der Quellcode dieser Programmiersprachen wird erst in Byte-Code umgewandelt der dann in einer Virtuelle Maschine ausgeführt wird. Dadurch ergibt sich eine längere Laufzeit.

Auf (24) wurde dieser Unterschied bewiesen. Dafür wurde ein Programm für dauerhaft Ein-/Ausschalten von LED in Python und C geschrieben und auf dem Raspberry Pi ausgeführt.



Abbildung 17: Leistung Python und C (24)

Die Zeitachse ist in Mikrosekunde und Nanosekunde bei Python beziehungsweise C. letztgenannt ist also Ca. 1000-mal schneller als Python. Jedoch hat er der Nachteil, dass er Komplexer ist.

Jedoch Reicht die Geschwindigkeit von Höhere Sprache für unseren Anwendungsfall vollkommen aus. Deswegen Wurde die Kommunikation Schnittstelle und die API beziehungsweise in Java und Python Implementiert. Ein Grund Dafür beide Module in verschiedene Sprachen zu implementieren, ist die Modularität des Systems hervorzuheben.

# Android Anwendung

Hier ist der Auswahl begrenz; Android Anwendungen können nur in Java programmiert werden. Die Kommunikation Schnittstelle wird dann in einem separaten Paket abgelegt, um seinen Austausch zu erleichtern.

# Entwurf

Dieses Kapitel umfasst den Praktischen Teil der Bachelorarbeit. Das erste Unterkapitel „Pflichtenheft“ gliedert sich in drei Kategorien „Muss“, „Wunsch“ und „Abgrenzung“ auf. In Stichpunkten wird aufgezählt, welche notwendigen Schritte in der Realisierung vorhanden sein müssen, welche Wunschkriterien eventuell realisiert werden, aber nicht zwingend erforderlich sind und explizite Abgrenzungen, die nicht umgesetzt werden.

Im Anschluss daran folgt die Implementierung von auf Abbildung 14 definierten Module. Schrittweise wird die Implementierung der Bibliothek und der API mithilfe von Bilder und Graphen beschrieben. Jedoch wird nur die wesentlichen Aspekte betrachtet. Danach folgt die Implementierung der Android Anwendung. Für bessere Verständnis wird die Implementierung mithilfe von Klassendiagramm erklärt. Die Wichtigsten Klassen, Interface und Komponenten werden dargestellt und deren Zusammenhangt verdeutlich.

# Pflichtenheft

* Musskriterien
  + Implementierung der Android App mit sinnvollen Steuerelemente
  + Implementierung der RasPi Schnittstelle und der API
  + Entwurf eines Modular und Universelle Bedienungssystem.
* Wunschkriterien
  + Implementierung einer 8Bit-Wert PWM Steuerelement
  + Einfügen von Steuerelement auf Android App implementieren
  + Löschen von Steuerelement auf Android App implementieren
  + Anzeige der RasPi Ausgabe auf Android App
* Abgrenzungskriterien
  + Test auf einem Praktischen Projekt

Ausblick

* Log auf Android anzeigen
* BaseControl class implementieren
* ControlRowView in BaseControl umfassen
* Implementierung von blinkControl
* Mit sigterm pwm prozess beeinflussen ohne es zu beenden und neuzustarten
* App Logik die entsprechen RPI-COm Nachricht Kategorie reagiert.
* Optimiert get() in API; param in ein Named Value speichern statt jedes mal die argument durchzusuchen
* AppLib: port beim schliessen der Anwendung freigeben und nicht bein jeder ausmachen
* HandleBlinkControl() optimieren
* Ähnlichkeiten zwischen Button und switchcontrol nutzen um code zu optimieren
* Echtzeit ausgabe auf app anstatt warten das alles fertig ist (Beim Blinken z.b.)
* Andere RequestHandler implementieren (Android app)

# Raspberry Pi Anwendung

# API und Seine Bibliothek

<Der Begriff API stammt aus dem englischen Sprachraum und ist die Kurzform von „Application-Programming-Interface“. Frei ins Deutsche Übersetzt bedeutet dies so viel wie „Schnittstelle zur Anwendungsprogrammierung“.>?

In unseren Kontext ist die API die Schnittstelle zur Verwaltung übertragene Signale. Somit ermöglich sie die gewünschte Universalität. Sie ist in der Programmiersprache Python geschrieben. Die Signal Informationen werden beim Aufruf des Skriptes als Argumente übergeben. Die Übergabe geschieht in Form von Name-Wert-Paare (vgl. Abbildung 18).



Abbildung 18: Beispiel Aufruf API mit Argumenten

Der Zugriff auf diese Argumente in Skripte wird durch die Funktion sys.argvermöglich. Es handelt sich um eine Python Methode, die an einem Skript übergebene Parametern auflistet (25).

Die Bibliothek ist auch in Python geschrieben und ihre Aufgabe ist die die Manipulation der Parameter zu vereinheitlichen und zu erleichtern. Weiterhin bietet Sie grundlegende Methoden zur Verwaltung GPIO-Ports. Sie umfasst unter anderen Methoden zur Ein-/Ausschalten, Blinken, und PWM-Ausgabe der Ports. Sie ist in der API einzubinden/eingebunden.

**Methoden zur Verwaltung übertragene Signale**

Die wesentlichen Methoden dieser Kategorie sind die get-, und log-Methoden:

* Die get-Methode durchsucht die Argumentliste und gibt den werden der Übergebene Argumentname zurück. Der Methode kann Eins bis Drei Parametern übergeben werden:
  + key: ist eine Zeichenkette (String[[7]](#footnote-7)), die den Argument-Name beschreibt. Er muss immer übergeben werden.
  + isRequired: vom Typ *boolean* ist False wenn vom Nutzer nicht übergeben. Er Bestimmt, ob das Argument erforderlich ist, wenn True*,* beendet sich das Programm mit Fehler Ausgabe auf die Konsole, falls der Wert nicht gefunden wurde.
  + isString: ist auch vom Typ *boolean* und seines Wert ist Standartmäßig False. Der Parameter legt fest ob der Wert als String behandelt werden soll oder nicht; beim False wird er als Integer behandelt.
* Die log-Methode wurde für eine strukturierte und einheitliche Konsole Ausgabe implementiert. Die Ausgabe mit dieser Methode beinhaltet das Datum, die Uhrzeit an denen die Ausgabe generiert wurde und die Nachricht selber. Sie werden in Form von HTML Element (span) angezeigt und mittels der HTML Klassen-Attribut unter verschiedene Kategorien zugeordnet. Zwecks ist die Android Anwendung zu ermöglichen, die Nachtricht-Kategorie zu erkennen und entsprechend darauf zu reagieren. Beim Aufruf der Funktion muss zusätzlich zur Nachricht auch die Abkürzung der Nachricht-Kategorie (als Character[[8]](#footnote-8)) übergeben werden. Es existiert schon folgende Kategorien (mit Abkürzungen):
  + ERROR (e) zum Ausgeben einem Fehler
  + WARNING (w) zum Ausgeben einer Warnung
  + TODO (t) zum Ausgeben einen noch nicht implementierte feature
  + INFO (i) zum Ausgeben von Standard Nachrichten

**Methoden Zur Verwaltung GPIO-Ports**

Außer handlePwmControl-Methode handelt es sich um Wrapper Methoden, die Ausgabe von Signale auf GPIO-Port vereinfachen indem sie eine Reihe von Methoden der RPi.GPIO[[9]](#footnote-9) Bibliothek einkapselt, die für die Ausgabe ein bestimmtes Signal notwendig sind. Die RPi.GPIO Bibliothek wurde in App Bibliothek eingebunden. Zu diesen Methoden zählen handleSwitchControl, handleButtonControl, handeBlinkControl, und handlePwmControl. Jede Methode wurde aufgebaut um das Signal eine bestimmte Control zu behandelt. handleBlinkControl behandelt beispielweise Signale vom BlinkControl.

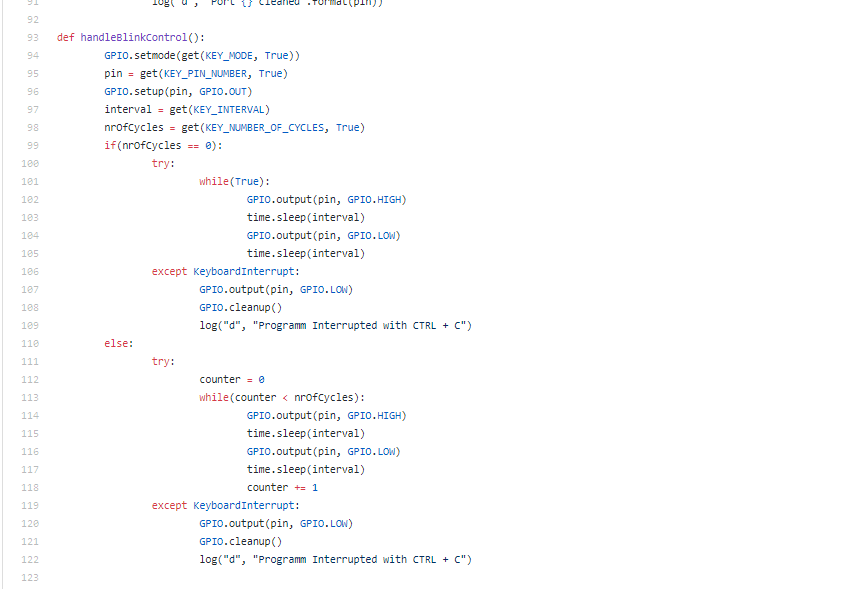


Abbildung 19: Methode handleBlinkControl der App Bibliothek

Bevor dieser Methode aufgerufen wird soll erst geprüft werden, ob der Signal vom Typ BlinkControl ist. Dies geschieht mithilfe der getName-Methode.

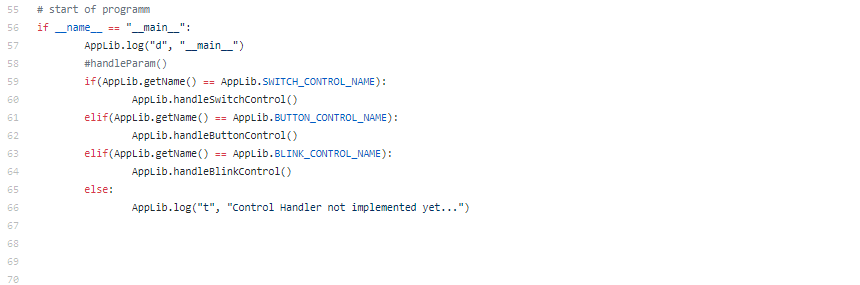


Abbildung 20: Beispiel Signal-Typ Überprüfung

Außer hanlePwmControl Funktionieren andere Methoden nach demselben Schema.

Wie eingangs erwähnt ist die handlePwmControl*-*Methode anders implementiert. Die Methode ist zur Behandlung PWM Signale. Der Raspberry Pi unterstützt keine Hardware PWM, daher wird er mit einer Software-Schleife emuliert. In einem getrennten Thread wird das Signal dauerhaft gesetzt und zurückgesetzt. Beim beenden des Programms wird das Thread auch beendet und somit das PWM ausgeschaltet (26).

Eine Schleife wäre aber in unseren Kontext ungünstig, weil die Anwendung würde ja kein Signal mehr empfangen bis die Schleife beendet wird. Damit unseren PWM an bleibt wurde mithilfe dieses Beispiels[[10]](#footnote-10) ein Daemon[[11]](#footnote-11) implementiert. Gemäß dem Beispiel wurde das Daemon in eine andere Datei ausgelagert. Das Daemon Paket muss erst installiert werden.



Abbildung 21: PWM Daemon Skript

Die Datei muss beim Aufruf in der Reihenfolge der Port Nummer, die Frequenz und der Duty Cycle der PWM übergeben werden. Der Aufruf kann von ein Python Skript mithilfe der Python Methode os.system ausgeführt werden.

# Kommunikation Schnittstelle

Die Aufgabe dieses Modul ist die Verarbeitung von Signale und die Bereitstellung der Konsole Ausgabe der API. Es handelt sich Grundsätzlich um eine Java Anwendung die zusätzlich zur Verarbeitung und Weiterleitung von Signale an den API auch eine HTML Seite generiert, die Informationen über das verarbeitete Signal und die Konsole Ausgabe der API in einer bestimmten Form beinhaltet. Die HTML Seite kann dann in Browser angeschaut werden oder wie in unseren Fall durch eine weitere Anwendung (Android Applikation) verarbeite. Die Anwendung besteht aus eine Java-klasse, deren wichtigste Methoden buildCmd und callApi sind. buildCmd generiert das Kommando, die zum Aufrufen des API Notwendig sind. Das Kommando besteht unter anderen aus der API Umgebung (Python – API ist in Python), sein Name, sein Absoluter Path, und den Parametern. callApi ruft API mittels generierte Kommando auf und fügt die vom API generierte Ausgabe in der HTML-Seite ein.

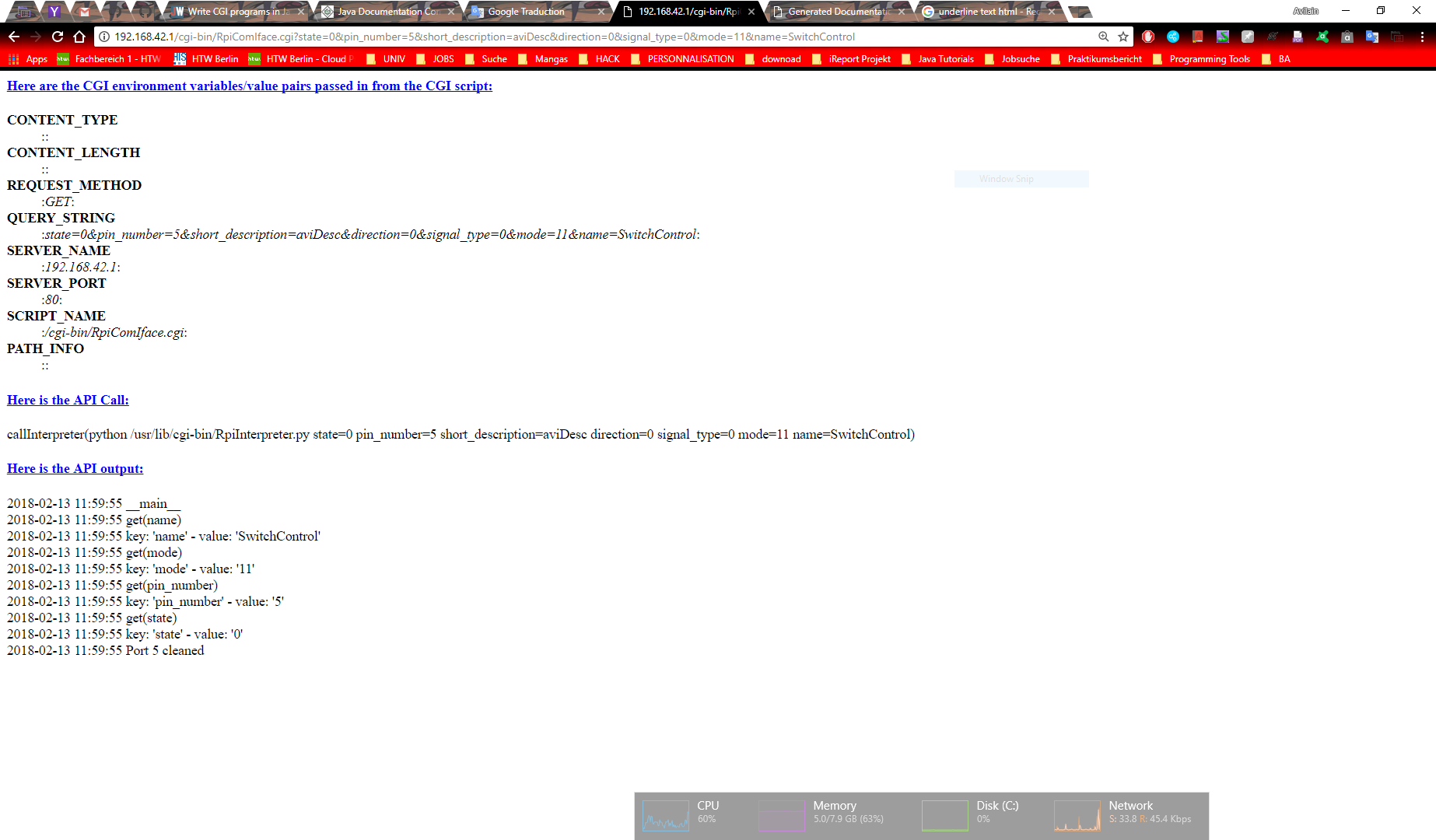


Abbildung 22: Vom COM Schnittstelle der Raspberry Pi generierte HTML Seite nach einem Signaldurchlauf

Die Anwendung wird vom CGI-Skript über Umgebungsvariable mit Signal Informationen versorgt und ausgeführt. Damit der Zugriff und die Verarbeitung von Signal Informationen leichter werden, wurde die Hier[[12]](#footnote-12) implementierte Bibliothek verwendet.

# Android Anwendung

Die Anwendung besteht aus einem Activity; Sein Layout besteht im wesentlich aus ein ListView der die Steuerelemente (Control) und deren Eigenschaften nach ein anders auflistet. Unter Control Eigenschaften versteht man, welche Port wird mit dieser Control gesteuert, welche Signal Type wird ausgegeben oder eingelesen, und zu welchem Zweck wurde der Control angelegt. Berührt man das Bildschirm vom linken Rand nach rechts, taucht das Menu auf. Hier kann man unter „Configure Manager“ kann man bestimmte Eigenschaften der Anwendung ändern. Beispielweise die URL der RasPi und das Nummerierungssystem. Das Einfügen und Löschen von Controls geschieht über „Add Control“ beziehungsweise „Remove Control“ (vgl. Abbildung 23).

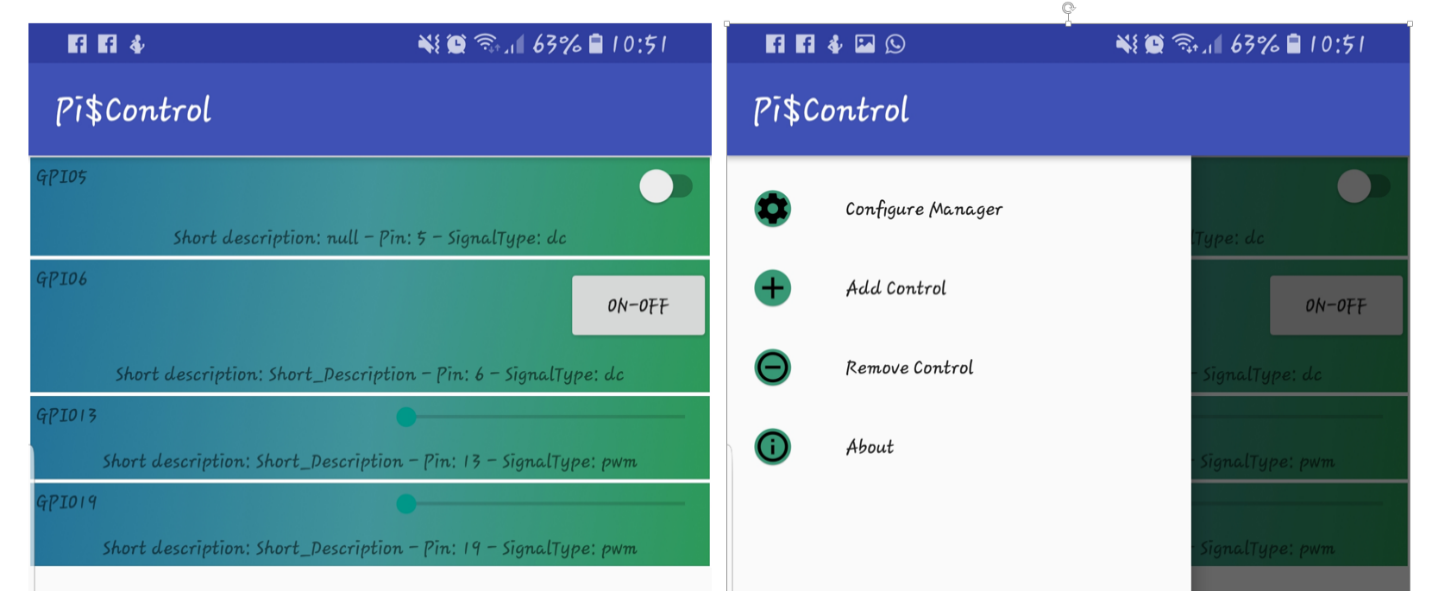


Abbildung 23: Android Anwendung GUI

# Steuerelement(Control)

Wie bereits erwähnt, ist ein Control Ein Android View der beim Touch-Ereignisse Signale an dem RasPi sendet. Es ist eigentlich ein java-Klasse, die folgende Kriterien erfüllt (siehe Abbildung 24).

* Implementierung eines Unterinterface von Control
* Vererbung einer Unterklasse von Android View
* Registrierung eines Touch-Events auf Android View



Abbildung 24: SwitchControl Klassendiagramm

Die Notwendigkeit jedes Kriteriums wird in den nächsten Absätzen erläutern.

**Implementierung eines Unterinterface von Control**

Control Unterinterface umfassen alle notwendigen Methoden für die Darstellung von Control und die Übertragung deren Signal. Die Wichtigsten sind:

* getViewDescription: liefert die Beschreibung des Control als zeichenkette (String) zurück. Diese Beschreibung beinhaltet Die Kurzbeschreibung, Der Signal Typ, und die Portnummer
* getRequestParams: gibt Parametern zurück, die für die HTTP Anfrage notwendig sind. Sie werden als Name-Wert-Paare gespeichert. Die Namen befinden sind schon als Konstant im Interface.
* setChangeListener: Hier wird festgelegt auf welchem Ereignis das View reagieren Soll und wie er reagieren soll. In unseren Fall wird eine HTTP Anfrage an dem RasPi gesendet.
* getView: Gibt View zurück, der zum Control gehört. Hier handelt es Sich um die Android View, das von der Klasse geerbt wurde.

**Vererbung einer Unterklasse von Android View**

Wie in den Grundlagen schon erläutert sind Oberfläche Elemente in Android Unterklassen von View. Die zu vererbte Unterklasse hängt vom Signal typ ab. Beispielsweise für ein PWM und DC Signal wird aus der Klasse Seekbar (Schieber) beziehungsweise Switch (Schalter) vererbt.

**Registrierung eines Touch-Events auf Android View**

Wie schon oben erläutert, soll es über setChangeListener-Methode geschehen. Der Registrierte Event hängt vom View ab. Auf Switch zum Beispiel wurde OnCheckedStateChanged-Listener registriert werden.

# Anzeige von Steuerelement

Die getView-Methode vom Control gibt nur das View für die Anzeige zurück. Es ist jedoch auch wichtig zu wissen, welche Port auf dem RasPi der Control Steuert und Welche Signal Typ wird ausgegeben. Diese Aufgabe übernimmt die ControlView-Klasse. Es handelt sich um ein Relativelayout, dessen Aufgabe die Anzeige von Control und seine Eigenschaften bezogen auf dem RasPi darstellt.



Abbildung 25: Control View vom SwitchControl

Die Eigenschaften werden vom getPort- und getViewDescription-Methode der Control Interface zurückgeliefert und deren Position im Layout mittels RelativeLayout.LayoutParams festgelegt (siehe Abbildung 25).

# Signalübertragung

Klassen die für die Kommunikation zuständig sind, wurden im Paket communication zusammengestellt, um den Austausch zu ermöglichen und vereinfachen (Modularität).



Abbildung 26: Klassendiagramm Kommunikation Schnittstelle Android

Wie bereits erwähnt werden Signale als HTTP Anfragen über WLAN and den RasPi übertragen. Es gibt verschiedene Anfragemethoden unter anderen GET, POST, PUT, DELETE. Sie haben jedoch Gemeinsamkeiten, die in der ControlRequest-Klasse zusammengefasst sind. Hier werden beispielweise Methoden zur Generierung des URLs und Empfang der RasPi Rückmeldung implementiert. Weiterhin beinhaltet sie auch Konstanten, die zur Signalübertragung und Verarbeitung der RasPi Rückmeldung Nötig sind (siehe Abbildung 27). Damit die Oberfläche während der Übertragung nicht blockiert ist, werden Anfragen in einem separaten Thread[[13]](#footnote-13) ausgeführt. Die Klasse AsyncTask*[[14]](#footnote-14)* vom Android Framework erleichtert die Gestaltung von separaten Thread.



Abbildung 27: ControlRequest Klassenvariable für die Übertragung

Die „CLASS\_NAME“ Konstanten bezeichnen die HTML-Klassen, die in der API log-Methode vergeben sind.

Die Übertragung übernimmt eine Unterklasse von ControlRequest; Die Entscheidung über eine Unterklasse bestimmt die Anfragemethoden die verwendet werden soll. PostRequestHandler-Klasse wird beispielweise für POST Methode verwendet; Seinem Konstruktor wird eine Unterinterface von Control Übergeben. Sie beinhalten die getRequestParam-Methode, die Anfrage Parameter zurückliefern. im doInBackground-Methode von AsyncTask-Klasse wird die Verbindung eingerichtet, die URL generiert, das Signal an dem RasPi übertragen, und die Rückmeldung empfangt (siehe Abbildung 28).



Abbildung 28: Implementierung doInBackground in PostRequestHandler-Klasse

# Einfügen von Steuerelement (Control)

Die Implementierung dieser Funktionalität erhält zwei zwischenschritte. Zuerst wurden GPIO-Port der RasPi auf Android Anwendung abgebildet und danach Ihre Gestaltung implementiert. Schließlich konnte das Einfügen von Steuerelement implementiert.

**Abbildung GPIO-Ports auf Android**

Zweck dieser zwischenschritt ist die Abbildung der Port Eigenschaften; darunter versteht man unter anderen:

* Welche Signale Sie ausgeben können
* Wie viele Control einem Port gleichzeitig steuern dürfen
* Welche Pin Nummer sie besitzen je nach Nummerierungssystem
* Um Welche Port Type handelt es sich (GPIO, POWER oder GROUND)

Diese Eigenschaften wurden in Port-Klasse und ihre Unterklassen nachgebildet.



Abbildung 29: Klassendiagramm von Port Typen

Port ist eine Abstrakte Klasse, die gemeinsame Port Eigenschaften und Methoden zusammenfasst. PortType ist ein Enum[[15]](#footnote-15); er bestimmt der Port Type (GPIO, GROUND, POWER).

**Port Gestaltung**

Diese Aufgabe übernimmt die ControlManager-Klasse. Sie umfasst GPIO-Ports Informationen. Darunter versteht man wie viele Port gibt’s, welche Signal kann welchen Port ausgeben, wie viel davon sind frei, welche Nummerierungssystem angewendet ist. Sie beinhaltet auch die URL Adresse der RasPi. Die Klasse wurde nach dem Singelton[[16]](#footnote-16) Entwurfsmuster implementiert, damit es nur eine einzige Instanz existiert.



Abbildung 30: Klassendiagramm ControlManager

Zu den Wichtigsten Methoden dieser Klasse gehören:

* generatePortList: Hier wird die Port Liste gemäß Abbildung 4 deklariert und zurückgegeben. Für andere RasPi Modelle sollte diese Methode überschrieben werden. Sie liefert eine ArrayList[[17]](#footnote-17) von Port
* PortFilter: diese Generic[[18]](#footnote-18) Methode Filtert die Port Liste nach dem Port Typ. Der Methode muss ein PortType übergeben werden.
* getServerUrl und getMode liefern der RasPi URL beziehungsweise das Nummerierungssystem zurück.

**Einfügen-Prozess**

Der Einstiegspunkt dieses Prozesses ist das Interface Editable; es liefert Informationen, die Später bei der Control Instanziierung gebraucht werden und besitzt zwei Methoden:

* getClazz: gibt das Objekt des Control zurück.
* getEditableAttributes: Diese Methode liefert Control Attributen, die für Erstellung des Steuerelements notwendig sind. Die Variablen werden als Liste von Name-Wert-Paare zurückgegeben. Als Wert wird den Datentyp des variable als FQN[[19]](#footnote-19) zurückgegeben (vgl. Abbildung 31).



Abbildung 31: Editable Implementierung in PwmControl-klasse

Control Interface ist ein Editable (Vererbung) und somit besitzen alle Control beide Methoden.

Die Übergabe von Attributen auf Abbildung 31 generiert folgende Dialog[[20]](#footnote-20).



Abbildung 32: Dialog für Neue PWM Control

Die Erstellung von Dialog auf Abbildung 32 Übernimmt EditableDialog-Klasse. Sein Konstruktor wird dem Editable übergeben und mittels getEditableAttributes-Methode die Liste der Name-Werte-paare geholt. Die getContentView-Methode von EditableDialog-Klasse macht eine Schleife auf die Liste und erzeugt für jeden Attribut-Eintrag ein Horizontales LinearLayout, der aus zwei Views besteht: Das erste beinhaltet den Eintragsnamen und das Zweite das Eingabefeld für den Eintragswert. Der Horizontale LinearLayout wird vom EditableAttributEntryView generiert.



Abbildung 33: EditableAttributEntryView Layout

Eigentlich besteht das Dialog Layout aus ein Vertikales LinearLayout, der Layouts von EditableAttributEntryView (Abbildung 33) zusammenfasst.

Beachtenswert ist, dass das Eingabefeld bei pin\_number ein Spinner[[21]](#footnote-21) ist. Hier werden nur Nummern von verfügbaren GPIO-Ports aufgelistet. Die Liste befindet sich im ControlManager (vgl. Abbildung 34).



Abbildung 34: Behandlung pin\_nummer Eingabefeld

Weitere Eingabefeldern mit Besonderheiten werden auch in setupValueView-Methode behandelt. Beispielweise wird bei Enum auch ein Spinner Angezeigt; die Auswertung erfolgt über Wert des Attribut-Eintrag



Abbildung 35: Behandlung von Enum Eingabefeld

Bestätigt man seine Eingabe mit „Create“ (Abbildung 32), werden Eingegebene Werte den ControlFactory-Klasse übergeben. Die getControl-Methode dieser Klasse liefert das Entsprechende Control zurück falls alle Parameter eingegeben wurden. Das Control wird dann auf Layout des Activity eingefügt. ControlFactory wurde nach dem Factory[[22]](#footnote-22) Entwurfsmuster implementiert.

# Implementierung

Dieser Kapitel umfasst Software, Entwicklungsumgebungen und Bibliotheken, die diese Bachelorarbeit verwendet wurden. Verfolgte Anleitungen werden auch referenziert und die Abweichungen auch erläutern. Der Kapitel unterteil sich in Raspberry Pi Unterkapitel und Android Unterkapitel.

# Raspberry Pi

# Inbetriebnahme

Für die Inbetriebnahme des RasPi ist ein Betriebssystem erforderlich. Es muss vorerst auf einer microSD-Karte installiert werden.

Das Betriebssystem dieser Bachelorarbeit (2017-07-05-raspbian-jessie) wurde auf Herstellerseite[[23]](#footnote-23) heruntergeladen, aus dem Archiv entpackt und mit einem zusätzlichen Programm (Etcher[[24]](#footnote-24)) auf die microSD-Karte kopiert (installiert).

Eine SSH-Verbindung mit dem Laptop wurde dann eingerichtet, sodass der Raspberry Pi selbst auf Tastatur und Monitor verzichten kann. Die Anleitung befindet sich auf (28).

Nach der Startvorgang muss man sich anmelden. Der Default Benutzername ist „pi“ und das Passwort ist „raspberry“. Die Auflösung wurde für eine bessere Darstellung erhöht

Nach dem Startvorgang erflogt die Anmeldung unter dem Benutzernamen „pi“ und das Passwort „raspberry“. Für eine bessere Darstellung wurde in Grundeinstellung unter **Advanced Option > Resolution** die Auflösung auf 1920\*1080 geändert. Die Grundeinstellungen erhält man durch Eingabe des Befehls **sudo raspi-config** in der Konsole.

# WLAN Access Point: Einrichtung

Die Installation und Einrichtung des Access Point erfolgte nach der Anleitung auf (29). Es handelt sich um ein PDF-Datei; Eine bearbeitete Version befindet sich im Anhang unter **RasPi/Anleitungen/wlanAP.pdf**; die Datei beinhaltet Kommentare, die Besonderheiten oder Unterschiede zu dieser Bachelorarbeit kennzeichnen.

# Webserver: Apache2 – Installation und Einrichtung

Apache Server (Version 2.4.29) wurde durch Ausführen folgender Kommando in Terminal installiert: sudo apt-get install apache-y (30). Die Installation muss man noch bestätigen. Dies erfolgt durch Eingabe von „y“ und drücken auf Enter-Taste.

Der Default Pfad für die Server Skripten ist /usr/lib/cgi-bin/. Da wurden die RasPi Module abgelegt. Damit GPIO-Port vom Smartphone ansprechbar sind, musst der Client Nutzergruppe (www-data) zur GPIO (gpio) eingefügt werden. Zu diesem Zweck führt man folgende Kommando in der Konsole aus: sudo adduser www-data gpio.

# Module

Die Entwicklungsumgebung der Kommunikation Schnittstelle und der API ist ein Texteditor namens Geany[[25]](#footnote-25). Sie bietet viele Vorteile wie Syntaxhervorhebung, Autovervollständigung und Code-Folding.

Die API wird mit der Programmiersprache Python in der Version 2.7.9 umgesetzt. Sie erfordert folgende Bibliotheken:

* **RPI.GPIO** (Version 0.6.3): vereinfacht den Zugriff auf GPIO-Pins; kann mit dem Befehl sudo apt-get install python dev python-rpi.gpio installiert werden.
* **sys**: ermöglicht den Zugriff auf Skript Argument
* **datetime**: für Datum und Zeit Ausgabe
* **time**: zum Anhalten der Anwendung beim Blinken
* **runner**: für PWM Behandlung (siehe Abbildung 21); Sie befindet sich in **daemon[[26]](#footnote-26)** Paket.

Die Kommunikation Schnittstelle wurde in der Programmiersprache Java in der Version … umgesetzt. Sie benötigt folgende Dateien:

* CgiLib.java: vereinfacht der Zugriff auf HTTP Anfrage Parametern; Sie befindet sich im Anhänge unter dem Verzeichnis **/RasPi/Software**. Die Originale Bibliothek befindet sich hier (24).
* RpiComIface.cgi: Empfängt die Anfrage und leitet sie an die Kommunikation Schnittstelle weiter; sie ist im Verzeichnis /RasPi/Software

Die Module befinden sich im Anhang unter /RasPi/Anwendung/.

# Android Anwendung

Die Anwendung wurde mit der Entwicklungsumgebung Android Studio (Version 2.3.3) entworfen. Die Umgebung wurde von Google und für Android veröffentlich und integriert schon alle Komponenten, die Für die Entwicklung notwendig sind. Zusätzliche Bibliotheken kann man immer im build.gradle Datei einbinden (31). Für diese Bachelorarbeit wurden folgende Bibliotheken eingebunden:

* **Jsoup[[27]](#footnote-27)** (Version 1.11.2):vereinfach dieBearbeitung der RasPi Rückmeldung.
* **Design Support Library[[28]](#footnote-28)** (Version 25.3.1): Für App Menü

Die Anwendung wurde mit Android SDK Version 25 kompiliert kann unter Android Version 4.4 oder früher laufen.

Die Anwendung wurde zuerst zeitnah mit einem Android Emulator namens Genymotion[[29]](#footnote-29) entwickelt und getestet. Die Verwendung dieser Emulator im Android Studio erfordert der Genymotion Plugin. Der Plugin kann man sich auf Android Repository (**File > Setting > Plugins**) holen und installieren. Nach der Installation wird das Genymotion Symbol in Symbolleiste angezeigt. Beim erst Start ist der Emulator einzurichten.



Abbildung 36: Emulator Konfiguration

Auf dem Emulator wurde Android 4.4.2 installiert

Anschließend wurde die Anwendung auf ein Samsung Galaxy S8 Plus (Android Version 7 und 8) getestet.

Der Quellcode befindet sich im Anhang unter **/Smartphone /Pi$Control/**

# Test

Die vollständig implementierten Funktionalitäten der Android App und seine API werden in diesem Kapitel getestet. Hier wird geprüft, inwieweit die Anwendung unseren am Anfangt definierten Voraussetzungen und Bedienungen erfüllt. Dieser Kapitel beinhaltet zwei Schritten: der Aufbau und die Bewertung der Ergebnisse.

# Test Aufbau

Sowohl die Android Anwendung als auch der Raspberry Pi werden eingerichtet.



Abbildung 37: Mein Bild Kommt noch HAHAH

**Android Smartphone**

Die Anwendung wurde auf ein Samsung Galaxy S8 Plus installiert und beinhaltet vier Control für den Test: einen Schalter, einen Knopf, und zwei Schieber (vgl. Abbildung 38). Weiterhin wird auch das Einfügen von Control getestet.



Abbildung 38: Android App für den Test

Der Letzte Control (GPIO9) liefert ein 8 Bit PWM Signal (Skala von 0 bis 255) aus und wird nicht komplett getestet. Es wird nur geprüft, ob der 8 Bit wert übertragen wird. Hier wurde das BCM Nummerierungssystem verwendet.

**Raspberry Pi**

Hier wird mit LEDs getestet, ob Signale richtig an den GPIO-Ports ankommen. Zu diesem Zweck werden LEDs am GPIO-Ports angeschlossen, die bei der Android App Angesprochen sind (siehe Abbildung 37). Die Betroffene Ports sind: GPIO5, GPIO6, GPIO13. Jeder LED wird wie auf Abbildung 37 verdrahtet.



Abbildung 39: LED Verdrahtung auf GPIO-Port

Schließlich wird auf RasPi eine kleine Anwendung zum Test der API Funktionalitäten geschrieben. Die Anwendung nutzt API Funktionen um die Signale zu bewerten und darauf zu reagieren.



Abbildung 40: Raspberry Pi Test Anwendung

# Bewertung der Ergebnisse

Die LED von GPIO5 geht an und aus, wenn man den Schalter Ein-/Ausschaltet.

Die LED von GPIO6 geht an, wenn der Knopf gedrückt ist. Beim Loslassen geht sie wieder aus

Die LED von GPIO13 wird auch richtig gedimmt; je Großer der Schieberwert ist, desto heller wird die LED und umgekehrt. Jedoch geht die LED erst aus und dann an. Die Erklärung liegt bei der handlePwmControl-Methode. Wie im 4.2.1 erläutern, werden PWM mit Daemon gehandelt. Daemon werden aber nach Ihrem Start im Prozess umgewandelt und man kann also auf seine Variablen nicht mehr zugreifen. Beim Änderung der Schiebewert wird zuerst der laufende Daemon (Prozess) gestoppt und ein Neuen mit dem Aktuellen Schiebewert gestartet.

Der 8 Bit PWM Wert wird auch richtig übertragen.

Die Steuerelemente verhalten sie also wie erwartet; das Einfügen eines Steuerelements durch App Menü wurde auch erfolgreich durchgeführt.

# Erreichter Stand & Ausblick

# Erreichter Stand

Anhand der Kriterien, die im Pflichtheft in Kapitel 4.1 aufgeführt wurden, wird der erreichte Stand aufgezählt.

Die Android Anwendung wurde mit Sinnvollem Steuerelemente implementiert. Zur sinnvollen Steuerelemente zählen ein der Schalter, die Taste und der Schieber. Die Sinnvolle Steuerelemente sind reine Aus

Die RasPi Schnittstelle und die API wurden implementiert und deren Funktionalitäten im Kapitel 6 getestet.

Die Musskriterien des Pflichtenhefts konnten während der Bearbeitungszeit umgesetzt werden.

Einige Wunschkriterien konnten auch erfüllt werden.

Der 8Bit-Wert PWM Steuerelement könnte auch in der Bearbeitungszeit implementiert werden. Es wurde auch geprüft, ob der Wert Korrekt umgewandelt wird.

Das Einfügen von Steuerelemente konnte bereit implementiert und Funktionsfähig. Der Nutzer bestimmt beim Einfügen die Eigenschaften des Steuerelements. Jedoch konnte aus Zeitlichen Gründen das Löschen von Steuerelemente nicht implementiert werden.

Aufgrund verbliebene Zeitdauer konnten die restlichen Wunschkriterien nicht umgesetzt werden. Dies betrifft die Anzeige der RasPi Ausgabe auf Android App und das Löschen von Steuerelemente.

Aus den Wunschkriterien konnten zwei von vier umgesetzt werden.

Die Abgrenzungskriterien wurden als nichterreichbar deklariert. Am Anfang wurde kein verfügbares Projekt gefunden, um die Android Anwendung zu testen.

# Fazit & Ausblick

Das Bachelorthema zu wählen, um ein modular und wiederverwendbar Android Anwendung für die Steuerung auf Raspberry Pi basierten eingebettet System zu entwerfen, war eine sinnvolle Entscheidung. Während der Entwurf wurden neuen Wissen erworben, die in der Zukunft sicherlich von Vorteil sein werden.

Überlegungen mussten angestellt werden, damit am Ende ein sinnvolles Projekt entstehen entsteht. Vieles aus dem Studium erworbenen Kenntnissen wurden angewendet und halfen bei der Umsetzung.

Der Grund sich mit dem Thema dieser Bachelorarbeit auseinanderzusetzen war einen sinnvollen Einstiegspunkt für die Entwicklung Bedienungssystem von Raspberry Pi basierten eingebettet System anzubieten. Durch das Modul Konzept ist der Nutzer möglich, Teile des Bedienungssystems auszutauschen. Die Konfigurierbare Oberfläche bietet den Nutzer die Möglichkeit, seine eigene Steuerelemente zu erstellen und die Übertragenen Signale mithilfe der API nach seinem Wunsch implementieren.

Als Übertragungsschnittstelle wurde WLAN für seine längere Reichweite gewählt. Ein weiterer Grund war, dass sowohl der Raspberry Pi 3 Modell als auch das Smartphone schon ein eingebautes WLAN Modul besitzen. Als Übertragungsmodus wurde der Access Point Variante gewählt, weil er Einfach mit vielen Android Geräten Kompatibel und vermeidet, dass der Raspberry Pi von anderen Netze abhängig ist.

Das Bedienungssystem bietet noch viele Erweiterungsmöglichkeiten.

Die restlichen Wunschkriterien können noch implementiert werden. Dazu gehören die Anzeige der Konsole Ausgabe der Raspberry Pi auf der Android Applikation und das Löschen von Steuerelementen.

Die Speicherung einer eingerichtete Bedienoberfläche wäre noch eine sinnvolle Funktionalität. Die gespeicherte Oberfläche könnte später geladen oder als Datei exportiert werden. Interessant wäre auch, das Verhalten der Applikation während Zustandsänderung der Activity zu implementieren, Damit beispielweise die Freigabe von RasPi Ressourcen erfolgt, wenn die Applikation vom System beendet wird.

Beim Erfolglose Signal Übertragung wird der Nutzer vom App informiert. Der Control Zustand ändert sich aber, obwohl den Signal auf RasPi Port nicht ausgegeben wurde. Die Behandlung dieses Problem wäre auch eine Sinnvolle Funktionalität.

1. <https://play.google.com/store/apps/details?id=com.jasonfindlay.pirelaypro> [04.02.2018] [↑](#footnote-ref-1)
2. <https://play.google.com/store/apps/details?id=es.pymasde.blueterm&hl=fr> [04.02.2018] [↑](#footnote-ref-2)
3. <https://de.statista.com/themen/581/smartphones/> [06.02.2018] [↑](#footnote-ref-3)
4. <https://developer.android.com/reference/android/content/Intent.html> [06.02.2018] [↑](#footnote-ref-4)
5. <https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md> [10.02.2018] [↑](#footnote-ref-5)
6. <https://de.wikipedia.org/wiki/Umgebungsvariable> [13.02.2018] [↑](#footnote-ref-6)
7. <https://en.wikipedia.org/wiki/String_(computer_science)> [12.02.18] [↑](#footnote-ref-7)
8. <https://en.wikipedia.org/wiki/Character_(computing)> [12.02.18] [↑](#footnote-ref-8)
9. <https://pypi.python.org/pypi/RPi.GPIO> [12.02.18] [↑](#footnote-ref-9)
10. <https://stackoverflow.com/a/9047339/7614256> [13.02.2018] [↑](#footnote-ref-10)
11. <https://pypi.python.org/pypi/python-daemon/> [13.02.18] [↑](#footnote-ref-11)
12. <https://www.javaworld.com/article/2076863/java-web-development/write-cgi-programs-in-java.html> [13.02.18] [↑](#footnote-ref-12)
13. <https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html> [16.02.18] [↑](#footnote-ref-13)
14. <https://developer.android.com/reference/android/os/AsyncTask.html> [16.02.2018] [↑](#footnote-ref-14)
15. <https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html> [17.02.18] [↑](#footnote-ref-15)
16. <https://de.wikipedia.org/wiki/Singleton_(Entwurfsmuster)> [17.02.18] [↑](#footnote-ref-16)
17. <https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html> [17.02.18] [↑](#footnote-ref-17)
18. <https://docs.oracle.com/javase/tutorial/java/generics/types.html> [17.02.18] [↑](#footnote-ref-18)
19. <https://en.wikipedia.org/wiki/Fully_qualified_name#Examples> [17.02.18] [↑](#footnote-ref-19)
20. <https://developer.android.com/guide/topics/ui/dialogs.html> [17.02.2018] [↑](#footnote-ref-20)
21. <https://developer.android.com/guide/topics/ui/controls/spinner.html> [17.02.2018] [↑](#footnote-ref-21)
22. <https://www.tutorialspoint.com/design_pattern/factory_pattern.htm> [17.02.2018] [↑](#footnote-ref-22)
23. <https://downloads.raspberrypi.org/raspbian/images/raspbian-2017-07-05/> [18.02.18] [↑](#footnote-ref-23)
24. <https://etcher.io/?ref=etcher_update> [18.02.18] [↑](#footnote-ref-24)
25. <https://www.geany.org/Main/HomePage> [18.02.2018] [↑](#footnote-ref-25)
26. <https://pypi.python.org/pypi/daemon-runner> [18.02.2018] [↑](#footnote-ref-26)
27. <https://github.com/jhy/jsoup> [18.02.2018] [↑](#footnote-ref-27)
28. <https://developer.android.com/topic/libraries/support-library/packages.html#design> [18.02.2018] [↑](#footnote-ref-28)
29. <https://www.genymotion.com/> [18.02.2018] [↑](#footnote-ref-29)