<Deckblatt>

Hier ist das deckblatt

**Kurzfassung**

**Ich fasse was kurz**

**Danksagung**

**Danke**

Inhaltsverzeichnis

[1. Einleitung 1](#_Toc507405250)

[1.1. Problemstellung/Zielsetzung 1](#_Toc507405251)

[1.2. Motivation 2](#_Toc507405252)

[2. Grundlagen 3](#_Toc507405253)

[2.1. Stand der Technik 3](#_Toc507405254)

[2.2. Raspberry Pi 5](#_Toc507405255)

[2.2.1. Hardware 5](#_Toc507405256)

[2.2.2. Software 9](#_Toc507405257)

[2.3. Android Smartphone 10](#_Toc507405258)

[2.3.1. Hardware 10](#_Toc507405259)

[2.3.2. Software 11](#_Toc507405260)

[2.3.3. Android Applikation 13](#_Toc507405261)

[3. Lösungsansatz 20](#_Toc507405262)

[3.1. Konzept 20](#_Toc507405263)

[3.2. Entscheidungen über den Entwurf 21](#_Toc507405264)

[3.2.1. Übertragung 21](#_Toc507405265)

[3.2.2. Raspberry Pi-Anwendung 23](#_Toc507405266)

[3.2.3. Android Applikation 25](#_Toc507405267)

[4. Entwurf 26](#_Toc507405268)

[4.1. Pflichtenheft 26](#_Toc507405269)

[4.2. Raspberry Pi Anwendung 26](#_Toc507405270)

[4.2.1. API 26](#_Toc507405271)

[4.2.2. Kommunikationsschnittstelle 28](#_Toc507405272)

[4.3. Android Applikation (Pi$Control) 30](#_Toc507405273)

[4.3.1. Steuerelement 31](#_Toc507405274)

[4.3.2. Anzeige von Steuerelement 34](#_Toc507405275)

[4.3.3. Signalübertragung 34](#_Toc507405276)

[4.3.4. Einfügen von Steuerelement (*Control*) 37](#_Toc507405277)

[5. Implementierung 44](#_Toc507405278)

[5.1. Raspberry Pi 44](#_Toc507405279)

[5.1.1. Inbetriebnahme 44](#_Toc507405280)

[5.1.2. WLAN Access Point - Einrichtung 44](#_Toc507405281)

[5.1.3. Webserver: Apache2 – Installation und Einrichtung 44](#_Toc507405282)

[5.1.4. Module 45](#_Toc507405283)

[5.2. Android Applikation 45](#_Toc507405284)

[6. Test 48](#_Toc507405285)

[6.1. Test Aufbau 48](#_Toc507405286)

[6.2. Bewertung der Ergebnisse 51](#_Toc507405287)

[7. Erreichter Stand & Ausblick 53](#_Toc507405288)

[7.1. Erreichter Stand 53](#_Toc507405289)

[7.2. Fazit & Ausblick 53](#_Toc507405290)

**Abbildungsverzeichnis**

[Abbildung 1: Graphische Oberfläche PiReplay (2) 4](#_Toc507406616)

[Abbildung 2: Projekt mit BlueTerm 4](#_Toc507406617)

[Abbildung 3: Komponentenübersicht des Raspberry-Pi 3 Modell B (5) 6](#_Toc507406618)

[Abbildung 4: Raspberry Pi GPIO-Pins (7) 7](#_Toc507406619)

[Abbildung 5: Marktanteil der Smartphone-Betriebssysteme weltweit (14) 11](#_Toc507406620)

[Abbildung 6: Architektur Android (11) 12](#_Toc507406621)

[Abbildung 7: Aufbau Android-Anwendung 13](#_Toc507406622)

[Abbildung 8: Beispiel Android Manifest 14](#_Toc507406623)

[Abbildung 9: Zustandsdiagramm einer Android Activity (16) 15](#_Toc507406624)

[Abbildung 10: Code Beispiel einer Android Activity 16](#_Toc507406625)

[Abbildung 11: Deklaration von einem RelativeLayout in Java 17](#_Toc507406626)

[Abbildung 12: Views Klassen in Android (18) 18](#_Toc507406627)

[Abbildung 13: Deklaration vom Android View in Java 19](#_Toc507406628)

[Abbildung 14: Aufbau des Steuerungssystems 21](#_Toc507406629)

[Abbildung 15: Vergleich Bluetooth und WLAN (20) 22](#_Toc507406630)

[Abbildung 16: Beispiel HTTP Anfrage mit Parametern 23](#_Toc507406631)

[Abbildung 17: Leistung Python und C (26) 24](#_Toc507406632)

[Abbildung 18: Aufruf der Nutzer Anwendung mit Argumenten 27](#_Toc507406633)

[Abbildung 19: Formatierte Ausgabe mit *log*-Methode 27](#_Toc507406634)

[Abbildung 20: Rückantwort der Raspberry Pi Kommunikationsschnittstelle 29](#_Toc507406635)

[Abbildung 21: Android Applikation GUI 31](#_Toc507406636)

[Abbildung 22: *SwitchControl* Klassendiagramm 32](#_Toc507406637)

[Abbildung 23: Klassendiagramm von *Control* 33](#_Toc507406638)

[Abbildung 24: Anzeige eines Steuerelements (*SwitchControl*) 34](#_Toc507406639)

[Abbildung 25: Klassendiagramm *communication*-Paket 35](#_Toc507406640)

[Abbildung 26: Konstanten der *ControlRequest*-Klasse 36](#_Toc507406641)

[Abbildung 27: Überschreibung der *doInBackground*-Methode in *PostRequestHandler*-Klasse 37](#_Toc507406642)

[Abbildung 28: Klassendiagramm von *Ports* 38](#_Toc507406643)

[Abbildung 29: Klassendiagramm *ControlManager* 39](#_Toc507406644)

[Abbildung 30: Implementierung von *Editable-*Methoden in *PwmControl*-klasse 40](#_Toc507406645)

[Abbildung 31: Dialog zur Erstellung eines *PwmControl* 41](#_Toc507406646)

[Abbildung 32: Generiertes Layout fürs Attribut Pin Nummer 42](#_Toc507406647)

[Abbildung 33: Behandlung des Eingabefeldes fürs Pin Nummer 42](#_Toc507406648)

[Abbildung 34: Behandlung von Enum Eingabefeldern 42](#_Toc507406649)

[Abbildung 35: Eigenschaften des virtuellen Gerätes 47](#_Toc507406650)

[Abbildung 36: Test Aufbau 48](#_Toc507406651)

[Abbildung 37: Android App für den Test 49](#_Toc507406652)

[Abbildung 38: Verbindung einer LED mit einem GPIO-Port 50](#_Toc507406653)

[Abbildung 39: Raspberry Pi Test Anwendung 51](#_Toc507406654)

**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

BLTE Bluetooth Low Energy

# Einleitung

Eingebettete Systeme haben in den letzten Jahren eine größere Reichweite erlangen. 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 Industrie, Automobilindustrie, Medizin und Telekommunikation. In Moderne Personenkraftwagen der Oberklasse sind beispielsweise zwischen 70 und 80 integrierte und miteinander vernetzte eingebettete Systeme enthalten (1). Sie erfordern in den meisten Fällen ein 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 eingebetteten Systems. Die Applikation überträgt Signale zum Raspberry Pi über eine drahtlose Schnittstelle.

Für den RasPi soll eine Anwendung entwickelt werden, die in der Lage ist, mit der Applikation zu kommunizieren. Sie soll Signale der Android Applikation empfangen und dem Nutzer zur Verfügung stellen. Dazu gehört auch eine API. Seine Aufgabe ist dem Nutzer Methoden für den Zugriff und die Verwaltung von Signalen anzubieten. Die Signale soll der Nutzer nach seinem Wunsch implementieren können. Nach Bearbeitung eines Signals soll die Anwendung eine Antwort an die Applikation senden.

Das Steuerungssystem besteht aus der Android Applikation und der RasPi Anwendung. Es soll modular aufgebaut werden, um ein späteren Austausch der Module zu ermöglichen. Das System soll einen sinnvollen Einstiegspunkt für die Entwicklung eines Steuerungssystems für eingebettete Systeme mit Raspberry Pi über Android Smartphone anbieten.

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 dem Testen gestaltet. Die Hauptvoraussetzung für die Entwicklung der Android Applikation, sind Kenntnisse von der Programmiersprache Java und das Android Framework. Die Entwicklung der Raspberry Pi Anwendung benötigt Kenntnisse in seiner Umgebung, unter anderen das Linux Betriebssystem und den Umgang mit den GPIO-Pins. Grundlagen aus der Elektrontechnik werden für das Testen benötigt. LEDs werden an den GPIO-Pins angeschlossen, um die Applikation zu testen.

Von der Grundidee bis zum Entwurf und Test wird nicht nur das im Studium erworbene Wissen benötigt, sondern auch Ergebnisse aus eigener Recherche 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 GPIO-Pins gelegt. Schließlich werden seine Funkschnittstellen betrachtet. Nach dem Raspberry Pi wird das Android Smartphone auch näher betrachten. Besonders werden seine Funkschnittstellen, sein Betriebssystem, der Aufbau seiner Applikation beschrieben.

## Stand der Technik

Vor Anfang dieser Bachelorarbeit, wurde nach Projekten gesucht, die 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“ enthalten (vgl. Abbildung 1).

Die Steuersignale werden als HTTP Anfragen über WLAN gesendet. 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 : 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 fürs Bluetooth Kommunikation und das zweite für die GPIO-Pins. Im Code ist schon festgelegt welche Pins angesprochen werden sollen.



Abbildung : Projekt mit BlueTerm

BlueTerm besitzt keine Graphische Oberfläche, sondern eine Eingabekonsole. Durch Eingabe von Ein (1) oder Null (0) in wird der Pin Ein- oder Ausgeschaltet.

**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 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 die Projektbestandteile nicht austauschbar, weil keine sichtbaren Grenzen dazwischen 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 verwendeten Modells kennengelernt. Besonderes Augenmerk wird auf seine GPIO-Pins und seine Funkschnittstelle gelegt.



Abbildung : 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 in der Abbildung 3 mit der Nummer 1 markiert. Sie ermöglichen den Anschluss von Peripherie Geräten und Messinstrumenten, um Daten zu verarbeiten und bestimmte Signale Auszugeben. 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 : Raspberry Pi GPIO-Pins (7)

Es gibt zwei Nummerierungssysteme zur Bezeichnung der Pins: BOARD und BCM. Das BOARD System bezieht sich auf die Physischen Position des Pins auf dem Board. Die Nummern gehen von 1 bis 40 und der Pin mit Nummer 1 steht direkt neben der Board Bezeichnung J8. 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 (8).

Der maximale Strom ist 50 mA bei den 3.3V Pins und 1 A bei den 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 (7).

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. Zum Beispiel 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.

Das WLAN Modul unterstützt die Standards 802.11b[[3]](#footnote-3), 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 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, 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 für den RasPi zu finden. Die Üblichen sind Ubuntu, Noobs und Raspbian.

Das beliebteste Betriebssystem für RasPi ist das kostenlose, Debian-basierte und für seine Hardware optimierte Raspbian. Seine Beliebtheit entsteht durch die kostenlose Lizenz, die schnelle Leistung, und die vorinstallierte Software und Tools. Sie gehört zu den Mehrzweckbetriebssystemen.

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, Texteditoren mit grundlegenden Entwicklungsumgebung Funktionen wie Geany und BlueJ.

**Anwendung**

Hier werden die unterstützten RasPi Programmiersprachen erläutert.

Der Name 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 **P**ython **I**nterpreter, eine Andeutung für die Hauptprogrammiersprache des Raspberry Pis (8).

Bei Python handelt es sich um eine Höhere Programmiersprache, die sich von Komplexität der Maschinensprachen entfernt. Erst der Einsatz von Interpreter oder Compiler übersetzt Befehle des Programmiercodes in Maschinensprache, die der Mikroprozessor beziehungsweise 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 werden. Laut Statista[[4]](#footnote-4) befand sich in 2017 weltweit 2,58 Mrd. Android Smartphone 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 meistens Smartphones und Tablets auf dem Markt antreibt. Die erste Android-Version kam im September 2008 auf dem Markt (10).



Abbildung : Marktanteil der Smartphone-Betriebssysteme weltweit (14)

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 : 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). 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 Applikation

In diesem Kapitel werden die wichtigsten Komponenten eine Android Applikation und deren Zusammenhang kennengelernt. Die Applikation 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 seine graphische Darstellung.



Abbildung : Aufbau Android-Anwendung

**Android Manifest**

Es beinhaltet grundlegende Informationen, die das System braucht, um die Applikation zu starten. In der Datei 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 : Beispiel Android Manifest

**Activity**

Eine Android Applikation besteht wesentlich aus Activities (3). Eine Activity ist ein Fenster mit dem der Nutzer mithilfe von 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.

Android greift im Vergleich zu anderen Betriebssystemen stärker in den Lebenslauf einer Applikation 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 : 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 zu der Activity gehört (siehe Abbildung 10). Dieser Methode muss ein Layout übergeben werden.



Abbildung : Code Beispiel einer Android Activity

**Layout**

Ein Layout definiert die Struktur einer graphischen Oberfläche (GUI). Es gibt verschiedene Typen von Layouts: LinearLayout, RelativeLayout, TableLayout, GridLayout*.* Der wesentliche Unterschied ist die Anordnung der GUI Elementen. Ein LinearLayout positioniert beispielweise Elemente in einer einzigen vertikalen oder horizontalen 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 an zur Erstellung von Layouts. Die Deklaration in Java erfordert die Vererbung der Klasse vom Layout-Typ (siehe Abbildung 11). Die Positionierung des GUI Elements und die Größe des Layouts werden mithilfe von LayoutParamsbestimmt (17). Layouts können auch verschachtelt sein.



Abbildung : Deklaration von einem RelativeLayout in Java

**View**

Views sind Elemente 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 an, die sich von ihrem Aussehen und ihrer Bedienungsform unterscheiden. Zu jedem View-Typ gehört eine Java-Klasse.



Abbildung : Views Klassen in Android (18)

Views können in XML oder in Java. In Java muss die Klasse von einer View-Klasse vererben. Die Gestaltung von Ereignissen erfolgt über Interfaces und Methoden von der View-Klasse (siehe Abbildung 13).



Abbildung : Deklaration vom Android View in Java

# Lösungsansatz

Dieses Kapitel umfasst die Entscheidungen des Aufbaus und Entwurfs 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 an den RasPi sendet. Die Kommunikationsschnittstelle sendet Signale über die Funkschnittstelle an den RasPi und leitet seine Nachtrichten an die Benutzeroberfläche weiter.

Die RasPi Anwendung besteht aus seiner 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 der 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 : Aufbau des Steuerungssystems

## Entscheidungen über den Entwurf

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 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 : 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 das 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-Anwendung

Dieser Abschnitt behandelt die Entscheidung über Komponenten, die den Raspberry Pi umfassen. Es wird erläutert wie Signale empfangen werden sollen. Des Weiteren werden Entscheidungen über die Programmiersprachen der Module getroffen.

**Webserver**

Wie oben erläutert, besteht das Hauptziel eines Access Point darin, ein lokales Netzwerke zu erstellen und den Internetzugang bereitzustellen. Andere Anwendungsszenarien sind ebenfalls möglich und haben entsprechende Softwareanforderungen. Damit der Access Point Anfragen von Client bearbeiten kann, muss er mit einem Webserver ausgestattet sein. Für diese Arbeit wurde Apache Web Server[[5]](#footnote-5) verwendet. Die Signale werden dann von der Android Applikation als HTTP-Anfrage gesendet. HTTP-Anfragen ermöglichen generell dem Client Informationen zum Webserver zu senden, um Formulare abzusenden, Dateien hochzuladen oder Dateien abzurufen (23). In unseren Fall rufen wir die Kommunikationsschnittstelle des RasPi mit auf und übergeben Signale als Parameter.



Abbildung : Beispiel HTTP Anfrage mit Parametern

**Module**

„File\_to\_call.cgi“ in der 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 Umgebungsvariablen[[6]](#footnote-6) in das CGI-Skript eingegeben (25).

Wie in den Grundlagen erläutert, lassen sich als Programmiersprachen Python, C/C++ und Java im RasPi verwenden. Außer C sind die anderen Sprachen sogenannte interpretierte Sprachen. Sie haben den Nachteil, dass sie langsamer sind als Maschinensprachen.

Auf (26) wurde dieser Unterschied bewiesen. Dafür wurde ein Programm in Python und C geschrieben, das dauerhaft die LED ein- beziehungsweise ausschaltet und auf dem Raspberry Pi ausgeführt. Die Ergebnisse befinden sich in der unteren Abbildung.



Abbildung : Leistung Python und C (26)

Die Zeitachse ist in Mikrosekunde und Nanosekunde bei Python beziehungsweise C. Letzteres ist ca. 100-mal schneller als Python. Jedoch besteht der Nachteil von C in der Komplexität.

Die Geschwindigkeit einer höheren Programmiersprache reicht für unseren Anwendungsfall allerdings vollkommen aus. Deswegen wurden die Kommunikationsschnittstelle und die API in Java beziehungsweise Python implementiert. Eine Ursache dafür beide Module in verschiedenen Sprachen zu implementieren besteht darin, den modularen Aufbau des Systems zu beweisen.

### Android Applikation

Bei Android-Anwendungen ist die Auswahl an Programmiersprachen begrenzt. Diese können nur in Java programmiert werden. Die Kommunikationsschnittstelle wird dann in einem separaten Paket abgelegt, um einen späteren 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 den Entwurf notwendig sind, welche Wunschkriterien gegebenenfalls realisiert werden, aber nicht zwingend erforderlich sind und explizite Abgrenzungen, die nicht umgesetzt werden.

Im Anschluss daran folgt der Entwurf der in Abbildung 14 definierten Module. Schrittweise wird der Entwurf der RasPi Anwendung mithilfe von Bilder und Graphen beschrieben. Danach folgt der Entwurf der Android Anwendung. Für ein besseres Verständnis wird der Entwurf mithilfe von Klassendiagrammen erklärt. Die wichtigsten Klassen, Interfaces und Komponenten werden dargestellt und deren Zusammenhang verdeutlicht.

## Pflichtenheft

* Musskriterien
  + Implementierung der Android Applikation mit sinnvollen Steuerelementen
  + 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

## Raspberry Pi Anwendung

### API

Wie bereits erwähnt, ist die API in der Programmiersprache Python geschrieben und muss in der Nutzer Anwendung eingebunden werden. Die wichtigsten Methoden werden in der folgenden Tabelle aufgelistet:

|  |  |
| --- | --- |
| Methode Name | Beschreibung |
| get | Gibt das Argumentwert zurück |
| log | Formatiert eine Meldung |
| getName | Gibt den Name des Steuerelements zurück |
| handleSwitchControl | Gibt das Signal eines SwitchControl aus |
| handleButtonControl | Gibt das Signal eines ButtonControl aus |
| handleBlinkControl | Gibt das Signal eines BlinkControl aus |
| handlePwmControl | Gibt das Signal eines PwmControl aus |

Tabelle : Wichtigsten Methoden der API

Die Argumente der Nutzer Anwendung beinhalten das Signal und werden als Name-Wert-Paaren übergeben (siehe Abbildung 18). Der get-Methode muss der Argument Name übergeben werden.



Abbildung : Aufruf der Nutzer Anwendung mit Argumenten

Die log-Methode Formatiert die Meldung in einem HTML Paragraph Element. Die Formatierte Ausgabe beinhaltet auch weitere Informationen wie das Datum, die Uhrzeit und die Kategorie. Die Kategorie wird mithilfe der HTML Klassen-Attribut angegeben (siehe Abbildung 19).



Abbildung : Formatierte Ausgabe mit log-Methode

SwitchControl, ButtonControl, BlinkControl und PwmControl sind Steuerelemente der Android Applikation.

Die handle-Methoden verwenden Methoden aus der Python Bibliothek RPi.GPIO um die Signale Auszugeben.

Der Raspberry Pi unterstützt keine Hardware-PWM, daher wird dies mit einer Software-Schleife emuliert. In einem getrennten Thread wird das Signal dauerhaft gesetzt und zurückgesetzt. Beendet sich das Programm, wird der Thread auch beendet und somit das PWM Signal ausgeschaltet (26).

Eine Schleife wäre aber in unserem Kontext ungünstig, weil die Anwendung kein Signal mehr empfängt bis die Schleife beendet wird. Damit die PWM eingeschaltet bleibt, wurde ein Daemon[[7]](#footnote-7) in PwmDaemon.py-Datei implementiert, der die PWM in einem separaten Thread behandelt.

### Kommunikationsschnittstelle

Die Schnittstelle leitet Signale an die Nutzer Anwendung weiter und stellt seine Ausgabe zur Verfügung (siehe Abbildung 19). Es ist in der Programmiersprache Java geschrieben. Nach Empfang und Weiterleitung eines Signals, generiert das Modul eine HTML-Seite als Rückantwort, die unter anderen die Ausgabe der Nutzer Anwendung enthält.



Abbildung : Rückantwort der Raspberry Pi Kommunikationsschnittstelle

In der folgenden Tabelle werden die Wichtigsten Methoden dieser Schnittstelle aufgeführt.

|  |  |
| --- | --- |
| Methode Name | Beschreibung |
| generateCmd | Generiert das Kommando zum Ausführen der Nutzer Anwendung |
| runUserSoftware | Führt Nutzer Anwendung mittels generierten Kommando aus |
| getProcessOutput | Stellt die Ausgabe der Nutzer Anwendung auf HTML Seite dar |

Tabelle : Wichtigsten Methoden der Raspberry Pi Kommunikationsschnittstelle

Die Schnittstelle erhält Signale von CGI-Skript namens RpiComIface.cgi. Letzteres übergibt das Signal über Umgebungsvariablen der Schnittstelle und ruft sie auf. Um die Manipulation von Signalen zu erleichtern, wurde die Java Bibliothek namens CgiLib.java[[8]](#footnote-8) bereitgestellt.

## Android Applikation (Pi$Control)

Die Applikation besteht aus einem Activity. Sein Layout ist ein vertikales ListView. Er beinhaltet die Steuerelemente und ihre Beschreibung. Unter Beschreibung versteht man, welcher Port wird mit dem Element gesteuert und welches Signal wird ausgegeben, sowie zu welchem Zweck wurde es angelegt.

Wird der Bildschirm, vom linken Rand nach rechts betätigt, taucht das Menu auf. Unter „Configure Manager“ können bestimmte Eigenschaften der Applikation geändert werden. Das Einfügen und Löschen von Steuerelementen geschieht über „Add Control“ beziehungsweise „Remove Control“ (vgl. Abbildung 21).



Abbildung : Android Applikation GUI

### Steuerelement

Ein Steuerelement (Control) ist eine View, die bei einem Touch-Ereignis eine oder mehrere Signale an den RasPi sendet. Es ist eigentlich ein java-Klasse, die folgende Kriterien erfüllt (siehe Abbildung 23).

* Implementierung eines Control-Unterinterface
* Vererbung einer View-Unterklasse.
* Registrierung eines Touch-Events auf der geerbten View-Unterklasse.

In der folgenden Abbildung ist das Klassendiagramm eines SwitchControl dargestellt. Die vererbte Unterinterface von Control ist OutputControl. Die Klasse erbt auch vom View-Unterklasse Switch. Die Vererbung von Interface OnCheckedChangeListner hilft dabei, den Touch-Event auf der vererbten View-Unterklasse zu registrieren. Die einzelnen Kriterien werden in folgenden Abschnitten erklärt.



Abbildung : SwitchControl Klassendiagramm

**Implementierung von** Control**-Unterinterface**

Control-Unterinterfaces beinhalten, notwendige Methoden für die Anzeige eines Steuerelements und die dazugehörige Übertragung von Signalen.



Abbildung : Klassendiagramm von Control

Die wichtigsten Methoden diesen Unterinterfaces werden in der folgenden Tabelle aufgeführt.

|  |  |
| --- | --- |
| Methode Name | Beschreibung |
| getView | Gibt der View des Steuerelements zurück |
| getViewDescription | Gibt die Beschreibung des Steuerelements zurück |
| getRequestParams | Gibt Parameter für die Sendung des Signals zurück |
| setChangeListener | Hier wird festgelegt, auf welchem Ereignis der View reagieren soll |

Tabelle : Wichtigsten Methoden von Control Interface

**Vererbung einer View-Unterklasse**

Durch Vererbung einer View-Unterklasse wird festgelegt, mit welchem View der Nutzer interagieren wird. Die zu vererbte Unterklasse hängt vom Signal typ ab. Für ein DC Signal wird beispielweise von der Switch-Klasse (Schalter) vererbt. Der View wird vom getView-Methode zurückgegeben.

**Registrierung eines Touch-Events auf der geerbten View-Unterklasse**

Wie schon oben erläutert, soll es über setChangeListener-Methode geschehen. Das Registrierte Event hängt vom geerbten View ab. Auf Switch zum Beispiel wurde OnCheckedStateChangedListener-Interface registriert werden (siehe Abbildung 22).

### Anzeige von Steuerelement

Ein Steuerelement wird immer mit seiner Beschreibung in einem RelativeLayout angezeigt. Diese Anzeige übernimmt die ControlView-Klasse. Dem Konstruktor muss ein Control übergeben werden.

Die Klasse verwendet Control-Methoden, um das Steuerelement und seine Beschreibung abzurufen (siehe Tabelle 3). Die Position der Elemente in dem RelativeLayout bestimmen die LayoutParams.



Abbildung : Anzeige eines Steuerelements (SwitchControl)

### Signalübertragung

Klassen die für die Kommunikation zuständig sind, wurden im Paket communication zusammengestellt.

Wie bereits erwähnt, werden die Signale als HTTP Anfragen über WLAN an den RasPi übertragen. Es gibt verschiedene Anfragemethoden: GET, POST, PUT, DELETE. Für jede Methode gibt es eine Klasse, die für die Übertragung zuständig ist (siehe Abbildung 25). Damit die Oberfläche während der Übertragung nicht blockiert wird, werden Anfragen in einem separaten Thread[[9]](#footnote-9) ausgeführt. Die Klasse AsyncTask*[[10]](#footnote-10)* vom Android Framework erleichtert die Gestaltung von Thread.



Abbildung : Klassendiagramm communication-Paket

Die ControlRequest-Klasse umfasst, Konstanten und grundlegende Methoden, die für die Übertragung der Signale und die Bearbeitung der RasPi Rückantwort (siehe Abbildung 20) notwendig sind.



Abbildung : Konstanten der ControlRequest-Klasse

Die Konstanten-Namen, die in der Abbildung 26 zu sehen sind mit dem Suffix „CLASS\_NAME“, bezeichnen die HTML-Klassen, die in der API log-Methode vergeben sind. Durch sie kann die Applikation in der RasPi Rückantwort die Meldungen der Nutzer Anwendung erkennen und Klassifizieren.

Die Übertragung ist in ControlRequest-Unterklassen implementiert. Dem Konstruktor diesen Klassen muss ein Control-Unterinterface übergeben werden. In der doInBackground-Methode wird in einem separaten Thread, die URL generiert, das Signal Übertragen und die Ausgabe der Nutzer Anwendung empfangt.



Abbildung : Überschreibung der doInBackground-Methode in PostRequestHandler-Klasse

### Einfügen von Steuerelement (Control)

Die Implementierung dieser Funktionalität erhält zwei zwischenschritte. Zuerst wurden GPIO-Ports der RasPi auf Android Applikation repräsentiert und danach Ihre Verwaltung implementiert. Schließlich konnte das Einfügen von Steuerelement umgesetzt.

**Repräsentierung der GPIO-Port auf der Android Applikation**

Zweck ist die Repräsentierung der Port Eigenschaften. Das heißt:

* Welche Signale Sie ausgeben können
* Wie viele Control einen 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 : Klassendiagramm von Ports

Port ist eine Abstrakte Klasse, die gemeinsame Port-Eigenschaften und Methoden zusammenfasst. PortType ist ein Enum[[11]](#footnote-11); er bestimmt der Port Type (GPIO, GROUND, POWER).

**GPIO-Ports Verwaltung**

Diese Aufgabe übernimmt die ControlManager-Klasse. Sie umfasst GPIO-Ports Informationen. Darunter versteht man wie viele Ports sind vorhanden, welche davon sind verfügbar, welches Nummerierungssystem ist angewendet. Sie beinhaltet auch die URL des RasPi. Die Klasse wurde nach dem Singleton[[12]](#footnote-12) Entwurfsmuster implementiert, damit es nur eine einzige Instanz existiert.



Abbildung : Klassendiagramm ControlManager

Die Wichtigsten Methoden dieser Klasse werden in der folgenden Tabelle aufgeführt:

|  |  |
| --- | --- |
| Methode Name | Beschreibung |
| generatePortList | Generiert Port-Liste gemäß Abbildung 4 |
| PortFilter | Filtert die Port-Liste nach dem Port Typ |
| getFreeGpioPortNumberList | Gibt die Liste der Verfügbare Port-Nummer zurück |
| getServerUrl | Gibt den URL des Servers zurück |

Tabelle : Wichtigsten von ControlManager-Klasse

**Einfügen-Prozess**

Der Einstiegspunkt dieses Prozesses ist das Interface Editable. Es liefert Informationen, die bei der Control Instanziierung notwendig sind. Das Interface besitzt zwei Methoden: getClazz getEditableAttributes

Die getClazz-Methode gibt das Klassen-Objekt des Control zurück.

Die Methode getEditableAttributes liefert Attribute als Name-Wert-Paare, die für die Erstellung des Steuerelements notwendig sind. Der Wert bezeichnet den Datentypen des Attributes. Das Attribut wird als FQN[[13]](#footnote-13) übergeben (vgl. Abbildung 30).



Abbildung : Implementierung von Editable-Methoden in PwmControl-klasse

Control erbt von Editable und somit besitzen alle Controls beide Methoden.

Mit den Attributen in der Abbildung 30 wird folgende Dialog[[14]](#footnote-14) erstellt.



Abbildung : Dialog zur Erstellung eines PwmControl

Die Erstellung von Dialogs Übernimmt die EditableDialog-Klasse. Sein Konstruktor wird dem Editable übergebe. Seine Methode getContentView beinhaltet eine Schleife, die für jeden Attribut-Eintrag ein Horizontales LinearLayout generiert, das aus zwei Views besteht. Die erste View beinhaltet den Eintragsnamen und die Zweite sein Eingabefeld. Das Horizontale LinearLayout wird vom EditableAttributEntryView-Klasse generiert.



Abbildung : Generiertes Layout fürs Attribut Pin Nummer

Das Dialog-Layout besteht aus einem vertikalen LinearLayout, das mehrere Layouts in der Abbildung 32 zusammensetzen.

Beachtenswert in Abbildung 32 ist, dass das Eingabefeld ein Spinner[[15]](#footnote-15) ist. Hier werden nur Nummern von verfügbaren GPIO-Ports aufgelistet. Die Liste befindet sich im ControlManager (vgl. Abbildung 33). Die Behandlung erfolgt über die setupValueView-Methode.



Abbildung : Behandlung des Eingabefeldes fürs Pin Nummer

Bei Enum Datentyp wird auch ein Spinner generiert.



Abbildung : Behandlung von Enum Eingabefeldern

Bestätigt man seine Eingabe mit „Create“ (Abbildung 32), wird der Control erstellt und angezeigt.

Die Erstellung des Controls übernimmt die ControlFactory-Klasse. Seine getControl-Methode generiert das Steuerelement. Die Klasse wurde nach dem Factory[[16]](#footnote-16) Entwurfsmuster implementiert.

# Implementierung

Das Kapitel umfasst Software, Entwicklungsumgebungen und Bibliotheken, die in dieser Bachelorarbeit verwendet wurden. Verfolgte Anleitungen werden auch referenziert und die Abweichungen auch erläutern.

## 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[[17]](#footnote-17) heruntergeladen, aus dem Archiv entpackt und mit einem zusätzlichen Programm (Etcher[[18]](#footnote-18)) 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 dem Startvorgang erflogt die Anmeldung unter dem Benutzernamen „pi“ und das Passwort „raspberry“.

### 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.

Das Passwort der Access Point ist „raspberrypi“

### 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.

Die Module der RasPi Anwendung sind unter im Ordner **/usr/lib/cgi-bin** abzulegen. Damit GPIO-Port vom Smartphone ansprechbar sind, müssen die Zugriffsrechte von den Webserver-Clients erweitert werden. Dies erfolgt durch Ausführen folgendes Kommando: sudo adduser www-data gpio.

### Module

Die Entwicklungsumgebung der Kommunikationsschnittstelle und der API ist ein Texteditor namens Geany[[19]](#footnote-19). Sie bietet viele Vorteile wie Syntaxhervorhebung, Autovervollständigung und Code-Folding.

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

* **RPI.GPIO** (Version 0.6.3): Für den Zugriff auf GPIO-Pins. Sie kann mit dem Befehl sudoapt-get install python dev python-rpi.gpioinstalliert werden.
* **sys**: Für den Zugriff auf Skript Argumente
* **datetime**: Für Datum und Zeit Ausgabe
* **time**: Zum Anhalten der Anwendung
* **runner**: für PWM Behandlung (siehe Abbildung 21); Sie befindet sich in daemon**[[20]](#footnote-20)** Paket.

Die Kommunikation Schnittstelle wurde in der Programmiersprache Java in der Version 1.8 umgesetzt. Sie benötigt folgende Dateien:

* **CgiLib.java**: vereinfacht der Zugriff auf HTTP Anfrage Parametern; Sie befindet sich im Anhänge unter dem Verzeichnis **/RasPi/software**.
* **RpiComIface.cgi**: Empfängt die Anfrage und leitet sie an die Kommunikationsschnittstelle weiter. Die Datei befindet sich im Verzeichnis **/RasPi/software**

Die Module befinden sich im Anhang unter **/RasPi/software.**

## Android Applikation

Die Applikation 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[[21]](#footnote-21)** (Version 1.11.2):vereinfacht dieBearbeitung der RasPi Rückmeldung.
* **Design Support Library[[22]](#footnote-22)** (Version 25.3.1): Fürs App-Menü

Die Anwendung wurde mit Android SDK Version 25 kompiliert und läuft unter Android Version 4.4 oder neuer laufen.

Die Anwendung wurde zuerst zeitnah mit einem Android Emulator namens Genymotion[[23]](#footnote-23) getestet. Die Verwendung dieser Emulator im Android Studio erfordert das Genymotion Plugin. Das Plugin kann man sich von Android Studio Repository (**File > Setting > Plugins**) herunterladen und installieren. Nach der Installation wird das Genymotion Symbol in Symbolleiste angezeigt. Beim erst Start ist der Emulator einzurichten.



Abbildung : Eigenschaften des virtuellen Gerätes

Auf dem virtuellen Gerät 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 des Steuerungssystems werden in diesem Kapitel getestet. Es beinhaltet zwei Schritten: der Aufbau und die Bewertung der Ergebnisse.

## Test Aufbau

Das Unterkapitel umfasst den Aufbau und die Einrichtung jedes Gerätes.



Abbildung : Test Aufbau

**Android Smartphone**

Die Anwendung wurde auf ein Samsung Galaxy S8 Plus installiert. Für den Test wurden Vier Steuerelementen vordefiniert (vgl. Abbildung 37).



Abbildung : Android App für den Test

Das Letzte Steuerelement (GPIO9) liefert ein 8 Bit PWM Signal. Es wird nur geprüft, ob der 8 Bit wert übertragen wird.

**Raspberry Pi**

Zu Testzwecken werden LEDs an GPIO-Ports angeschlossen, die auf der Android Applikation angesprochen sind (siehe Abbildung 37). Die Verdrahtung jeder LED erfolgt wie in der unteren Abbildung.



Abbildung : Verbindung einer LED mit einem GPIO-Port

Auf dem RasPi wurde eine Testanwendung geschrieben. Hier werden API Methoden verwenden um die Signale zu bewerten und darauf zu reagieren.



Abbildung : Raspberry Pi Test Anwendung

## Bewertung der Ergebnisse

Die LED von GPIO5 geht an und aus, wenn man den Schalter Ein- beziehungsweise 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 ist die LED und umgekehrt. Bei Änderung des Schiebewertes geht die LED kurz aus. Die Erklärung liegt bei der handlePwmControl-Methode. Wie im 4.2.1 erläutern, werden PWM mit Daemon gehandelt. Daemon werden aber nach dem Start im Prozess umgewandelt. Bei Änderung des Schiebewertes wird zuerst der laufende Prozess gestoppt und einen neuen mit dem aktuellen Schiebewert gestartet.

Der 8 Bit PWM Wert wird auch richtig übertragen.

das Einfügen eines Steuerelements durch App Menü wurde auch erfolgreich durchgeführt.

Die implementierten Funktionalitäten des Steuerungssystems funktionieren wie erwartet.

# Erreichter Stand & Ausblick

## Erreichter Stand

Anhand der Kriterien, die im Pflichtenheft (Kapitel 4.1) aufgeführt wurden, wird der erreichte Stand aufgezählt.

Die Android Anwendung wurde mit sinnvollen Steuerelementen implementiert. Zu den Steuerelementen zählen der Schalter, die Taste und der Schieber.

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

Alle genannten Kriterien sind in den Musskriterien des Pflichtenhefts (Kapitel 4.1) aufgeführt. Sie konnten während der Bearbeitungszeit umgesetzt werden.

Einige Wunschkriterien konnten auch erfüllt werden.

Der 8Bit-Wert PWM Steuerelement konnte auch implementiert werden.

Das Einfügen von Steuerelementen konnte umgesetzt werden. Der Nutzer bestimmt bei dem Prozess die Eigenschaften des Steuerelements.

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

Aus den Wunschkriterien konnten zwei von vier umgesetzt werden.

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

## Fazit & Ausblick

Die Auswahl dieser Bachelorarbeit war eine sinnvolle Entscheidung. Während des Entwurfs wurde neues Wissen erworben, das in der Zukunft sicherlich von Vorteil sein werden.

Überlegungen mussten angestellt werden, damit am Ende ein sinnvolles Projekt entsteht. Erworbene Kenntnisse aus dem Studium wurden angewendet und halfen bei der Umsetzung.

Hauptmerkmal dieser Bachelorarbeit war es, einen sinnvollen Einstiegspunkt für die Entwicklung von RasPi Steuerungssystem über Android Smartphone anzubieten. Durch das Modul Konzept können Bestandteile des Steuerungssystems ausgetauscht werden. Die Konfigurierbare Oberfläche bietet dem Nutzer die Möglichkeit, seine eigene Steuerelemente zu erstellen und die Übertragenden Signale mithilfe der API nach seinem Wunsch zu implementieren.

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

Das Steuerungssystem bietet noch viele Erweiterungsmöglichkeiten.

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.

Bei einem erfolglosen Signal Übertragung wird der Nutzer von der Applikation informiert. Der Control Zustand ändert sich aber trotzdem, obwohl das Signal auf dem RasPi Port nicht ausgegeben wurde. Es wäre dann sinnvoll, dass der Control dann automatisch in den alten Zustand zurückkehrt.

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://fr.wikipedia.org/wiki/IEEE_802.11> [24.02.2018] [↑](#footnote-ref-3)
4. <https://de.statista.com/themen/581/smartphones/> [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://pypi.python.org/pypi/python-daemon/> [13.02.18] [↑](#footnote-ref-7)
8. <https://www.javaworld.com/article/2076863/java-web-development/write-cgi-programs-in-java.html> [13.02.18] [↑](#footnote-ref-8)
9. <https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html> [16.02.18] [↑](#footnote-ref-9)
10. <https://developer.android.com/reference/android/os/AsyncTask.html> [16.02.2018] [↑](#footnote-ref-10)
11. <https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html> [17.02.18] [↑](#footnote-ref-11)
12. <https://de.wikipedia.org/wiki/Singleton_(Entwurfsmuster)> [17.02.18] [↑](#footnote-ref-12)
13. <https://en.wikipedia.org/wiki/Fully_qualified_name> [24.02.2018] [↑](#footnote-ref-13)
14. <https://developer.android.com/guide/topics/ui/dialogs.html> [17.02.2018] [↑](#footnote-ref-14)
15. <https://developer.android.com/guide/topics/ui/controls/spinner.html> [17.02.2018] [↑](#footnote-ref-15)
16. <https://www.tutorialspoint.com/design_pattern/factory_pattern.htm> [17.02.2018] [↑](#footnote-ref-16)
17. <https://downloads.raspberrypi.org/raspbian/images/raspbian-2017-07-05/> [18.02.18] [↑](#footnote-ref-17)
18. <https://etcher.io/?ref=etcher_update> [18.02.18] [↑](#footnote-ref-18)
19. <https://www.geany.org/Main/HomePage> [18.02.2018] [↑](#footnote-ref-19)
20. <https://pypi.python.org/pypi/daemon-runner> [18.02.2018] [↑](#footnote-ref-20)
21. <https://github.com/jhy/jsoup> [18.02.2018] [↑](#footnote-ref-21)
22. <https://developer.android.com/topic/libraries/support-library/packages.html#design> [18.02.2018] [↑](#footnote-ref-22)
23. <https://www.genymotion.com/> [18.02.2018] [↑](#footnote-ref-23)