Hochschule für Technik und Wirtscahft Berlin

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Engineering
über das Thema

Android App zur Erzeugung und drahtlosen Übertragung von Steuersignalen an einen Raspberry Pi

im Studiengang
Computer Engineering

Am Fachbereich
Ingenieurwissenschaften – Energie und Information

Autor: Avilain Kenne Foue
Matrikelnummer: 547801

Erstprüfer: Prof. Dr. Thomas Baar
Zweitprüfer: Dipl.-Ing. Peter Puschmann

Berlin, den 28.02.2018

Kurzfassung

Die vorliegende Arbeit beschäftigt sich damit, die Entwicklung Steuerungssysteme für Raspberry Pi basierte eingebettete Systeme über ein Android Smartphone zu erleichtern. Zu diesem Zweck wurde ein modulares, und wiederverwendbares Steuerungssystem entwickelt, das aus einer Android Applikation und einer Raspberry Pi Anwendung besteht.

Die Applikation Oberfläche bietet die Möglichkeit an, die Steuerelemente einzufügen oder zu löschen und die API der Raspberry Pi Anwendung, die übertragenen Signale nach Wunsch zu implementieren. Die API beinhaltet auch Methoden für die Ausgabe der grundlegenden Signale. Die Signale werden über eine drahtlose Schnittstelle übertragen.

Danksagung

Ein herzliches „Dankeschön!“ geht an meinen Betreuer Dipl.-Ing. Peter Puschmann für seine Zeit, seine Mühe, seine Konstruktive Kritik, seine motivierenden Worte, seine tolle Unterstützung und die ausgezeichnete Betreuung.

Weiterhin möchte ich mich bei Prof. Dr. Thomas Baar für die gute Betreuung meiner Arbeit bedanken.

Danke auch zu meinen Freunden und Kollegen für ihre Zeit und Mühe als Korrekturleser und die konstruktive Kritik.

Der Größte Dank gilt meiner Familie. Vielen Dank für die Unterstützung sowie Euren motivierenden Beistand während meines gesamten Studiums!

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

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

1. 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 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, und flexibel konfigurierbares 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.

1.1. Problemstellung/Zielsetzung

Ziel dieser Bachelorarbeit, ist die Entwicklung einer universellen Android Applikation für die flexible Steuerung 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 ein Android Smartphone anbieten.

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

1.2. 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. 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.

2. 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.

2.1. 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]

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, wird der Pin Ein- oder Ausgeschaltet [2].

[1] https://play.google.com/store/apps/details?id=com.jasonfindlay.pirelaypro [04.02.2018]

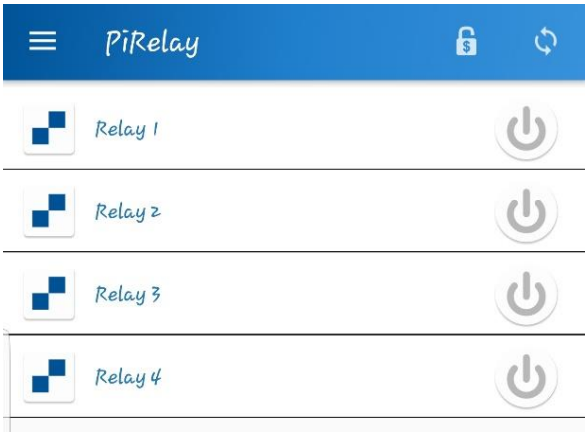


Abbildung 1: Graphische Oberfläche PiReplay [3]

BlueTerm[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.

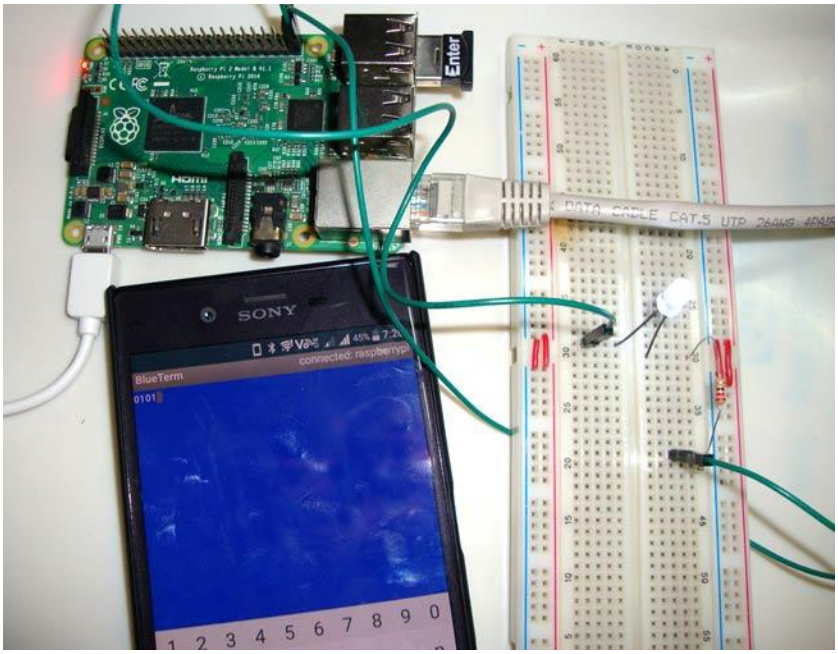


Abbildung 2: Projekt mit BlueTerm

[2] https://play.google.com/store/apps/details?id=es.pymasde.blueterm&hl=fr [04.02.2018]

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.

2.2. 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 an, 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 müssen extra gekauft werden.

2.2.1. Hardware

In diesem Kapitel werden physische Komponente des verwendeten Modells kennengelernt. Besonderes Augenmerk wird auf seine GPIO-Pins und seine Funkschnittstelle gelegt.

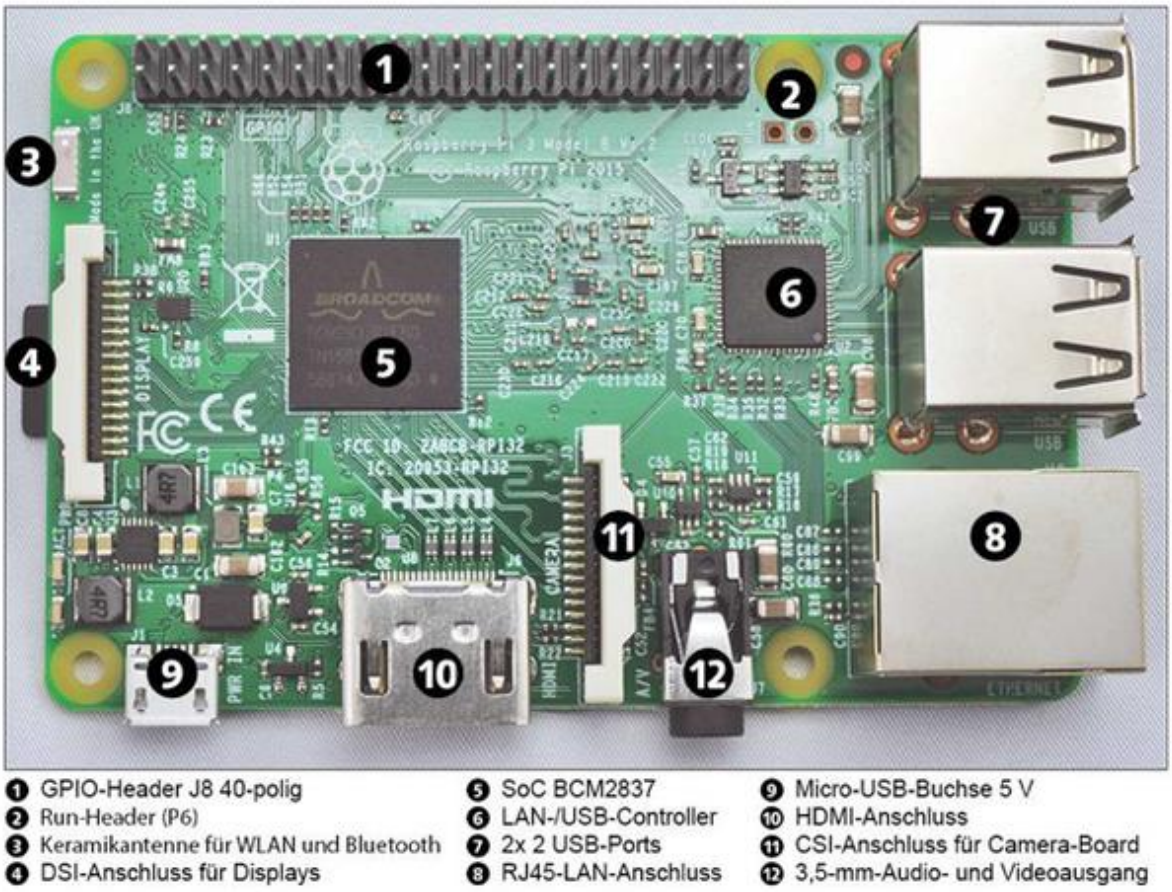


Abbildung 3: Komponentenübersicht des Raspberry-Pi 3 Modell B [6]

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

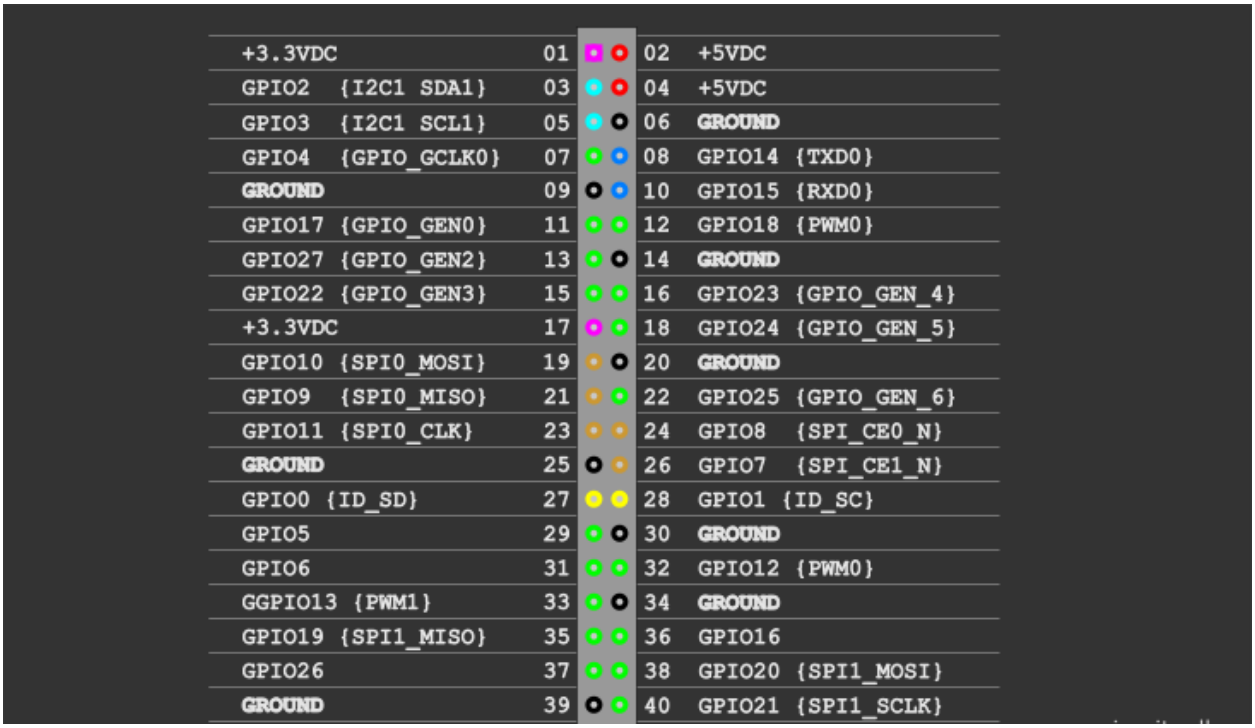
Audioausgabe: HDMI (digital), 3,5-mm-Klinkenstecker (analog)

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].



+3.3VDC	01		02	+5VDC	
GPIO2 {I2C1 SDA1}	03		04	+5VDC	
GPIO3 {I2C1 SCL1}	05		06	GROUND	
GPIO4 {GPIO_GCLK0}	07		08	GPIO14 {TXD0}	
GROUND	09		10	GPIO15 {RXD0}	
GPIO17 {GPIO_GEN0}	11		12	GPIO18 {PWM0}	
GPIO27 {GPIO_GEN2}	13		14	GROUND	
GPIO22 {GPIO_GEN3}	15		16	GPIO23 {GPIO_GEN_4}	
+3.3VDC	17		18	GPIO24 {GPIO_GEN_5}	
GPIO10 {SPI0_MOSI}	19		20	GROUND	
GPIO9 {SPI0_MISO}	21		22	GPIO25 {GPIO_GEN_6}	
GPIO11 {SPI0_CLK}	23		24	GPIO8 {SPI_CE0_N}	
GROUND	25		26	GPIO7 {SPI_CE1_N}	
GPIO0 {ID_SD}	27		28	GPIO1 {ID_SC}	
GPIO5	29		30	GROUND	
GPIO6	31		32	GPIO12 {PWM0}	
GGPIO13 {PWM1}	33		34	GROUND	
GPIO19 {SPI1_MISO}	35		36	GPIO16	
GPIO26	37		38	GPIO20 {SPI1_MOSI}	
GROUND	39		40	GPIO21 {SPI1_SCLK}	

Abbildung 4: 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], g und n und arbeitet im 2.4 GHz Band (BCM43143). Das Modul hat eine maximale übertragungsrate von 150 Mbit/s [9].

2.2.2. Software

Betriebssystem

Das Betriebssystem ist eine Zusammenstellung von Computerprogrammen, die die Systemressourcen eines Computers wie Arbeitsspeicher, Festplatten, Ein- und Ausgabegeräte verwaltet [10]. 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 sie sich für Zeitnah reagierende System wie Industrieanlagen, Drohnen, Antiblockiersystem und Roboter. Ein Beispiel ist Minoca OS [11].

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.

[3] https://fr.wikipedia.org/wiki/IEEE_802.11 [24.02.2018]

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.

2.3. 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.

2.3.1. Hardware

Wegen der Vielfalt von Android Smartphones, können ihre physische Komponenten nicht genau beschrieben werden. Laut Statista[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 [12].

2.3.2. 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 [13].

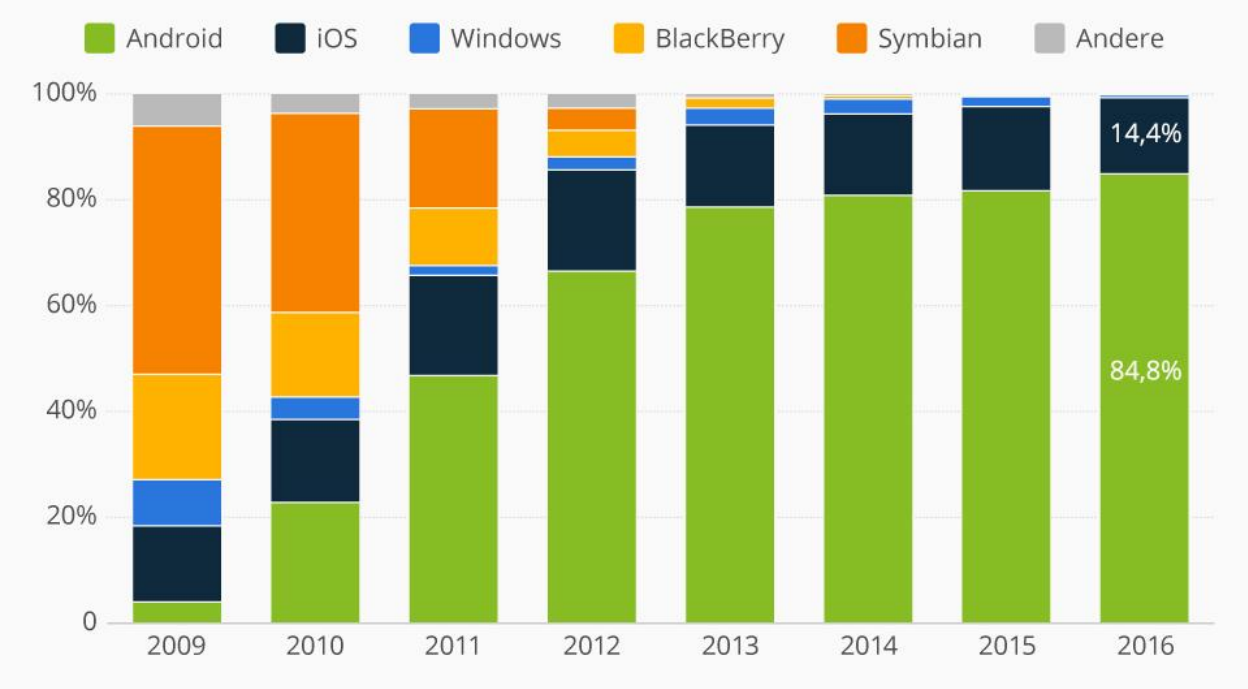


Abbildung 5: Marktanteil der Smartphone-Betriebssysteme weltweit [14]

Das Betriebssystem beinhaltet vier Schichten und Jede abstrahiert die darunterliegende. Es besteht aus einem angepassten Linux-Kernel. Das Kernel bildet die Schicht zwischen

[4] https://de.statista.com/themen/581/smartphones/ [06.02.2018]

der Hardware und der restlichen Architektur. Er enthält unter anderem Hardwaretreiber, Prozess- Speicherverwaltung und Sicherheitsverwaltung (siehe Abbildung 6).

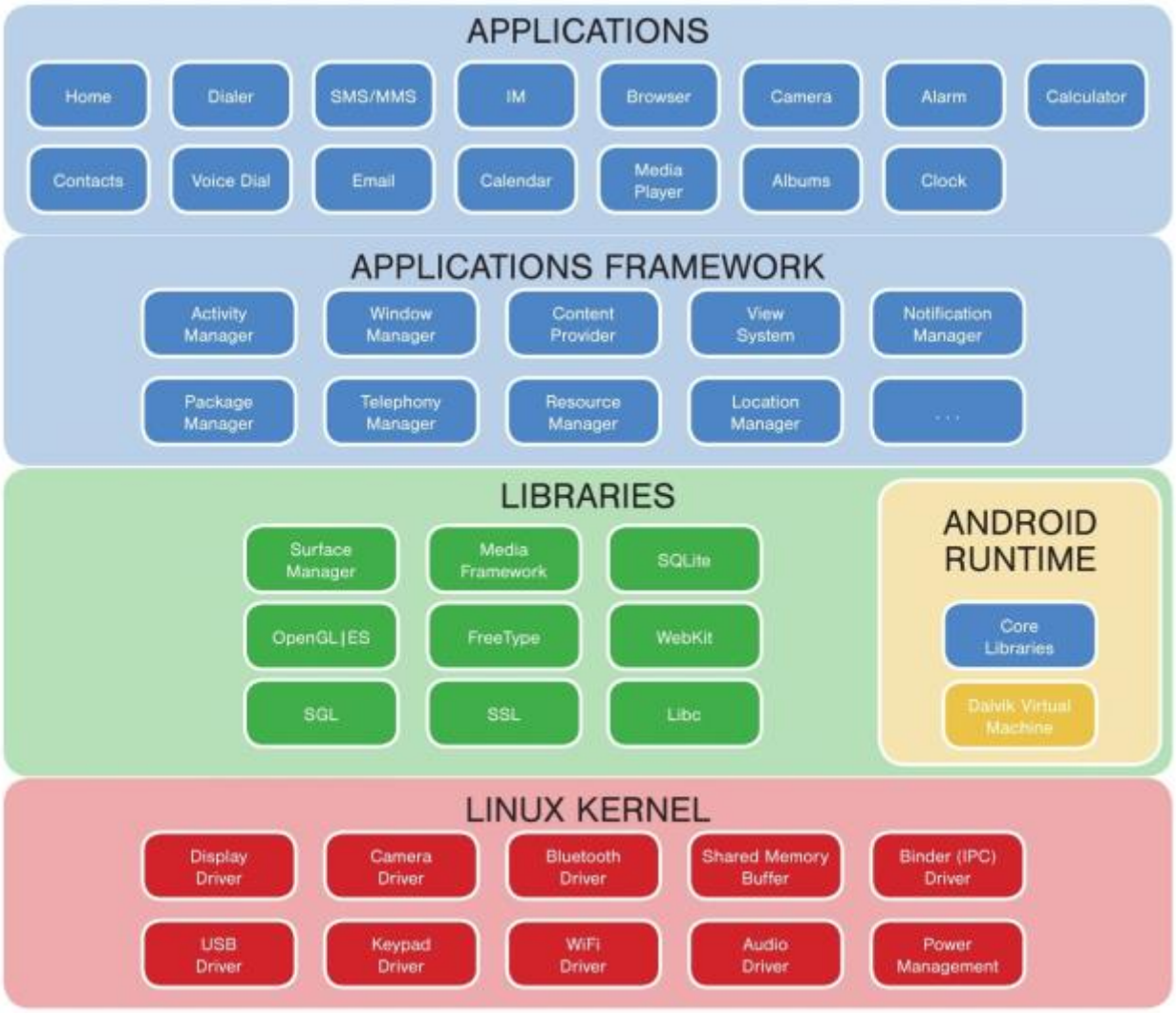


Abbildung 6: Architektur Android [15]

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 [16].

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 [16]. 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.

2.3.3. 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 7: Aufbau Android-Anwendung [17]

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.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="testapp.android.test">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CAMERA" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Test"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:fullBackupContent="false"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Abbildung 8: Beispiel Android Manifest

Activity

Eine Android Applikation besteht wesentlich aus Activities [16]. 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 [18].

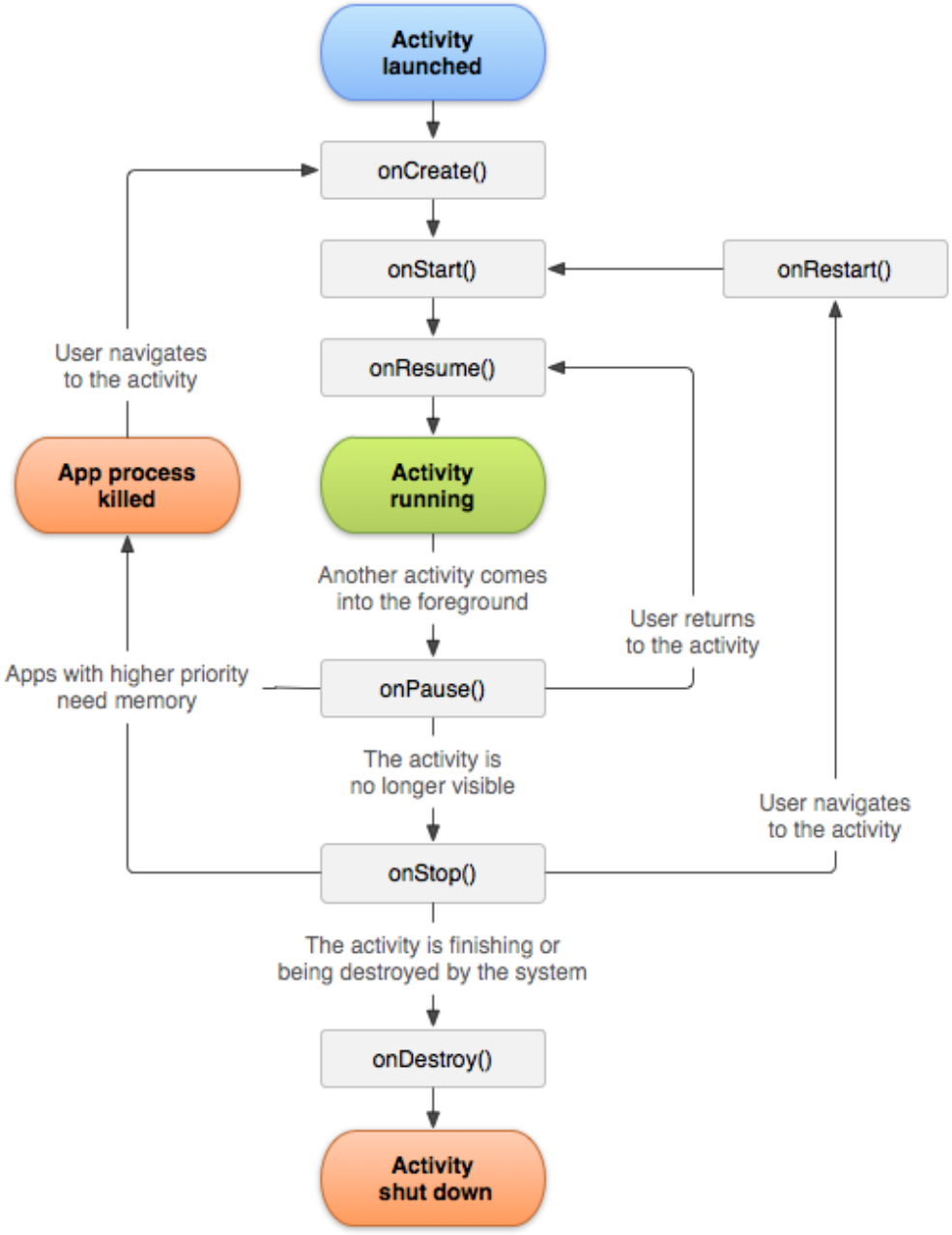


Abbildung 9: Zustandsdiagramm einer Android Activity [19]

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. Da wird auch mithilfe von der *setContentView*-Methode bekannt gegeben, welches Layout

zu der Activity gehört (siehe Abbildung 10). Dieser Methode muss ein Layout übergeben werden.

```
package testapp.android.test;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Abbildung 10: 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 LayoutParams bestimmt [19]. Layouts können auch verschachtelt sein.

```java
package testapp.android.test;

import android.content.Context;
import android.util.AttributeSet;
import android.view.ViewGroup;
import android.widget.RelativeLayout;

public class MyRelativeLayout extends RelativeLayout {

    public MyRelativeLayout(Context context) {
        super(context);
        init();
    }

    public MyRelativeLayout(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    private void init() { setLayoutParams(generateLayoutParams()); }

    private LayoutParams generateLayoutParams() {
        return new LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
                                ViewGroup.LayoutParams.WRAP_CONTENT);
    }
}
```

Abbildung 11: 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.

View-Klasse	Beschreibung
TextView	Einfache Textausgabe
Button, ImageButton	Schaltfläche
EditText	Texteingabe
CheckBox	Ankreuzfeld
RadioGroup, RadioButton	Auswahlschalter; einer aus RadioGroup
Spinner	Auswahlliste
DatePicker, TimePicker	Auswahl von Datum und Zeit
AnalogClock, DigitalClock	Anzeige Uhrzeit (analog, digital)

Abbildung 12: Views Klassen in Android [20]

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).

```java
package testapp.android.test;

import android.content.Context;
import android.util.AttributeSet;
import android.view.View;
import android.widget.Button;


class NewButton extends Button implements View.OnClickListener {

    public NewButton(Context context) {
        super(context);
        init();
    }

    public NewButton(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    private void init() { setOnClickListener(this); }

    @Override
    public void onClick(View v) {...}
}
```

Abbildung 13: Deklaration vom Android View in Java

3. 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.

3.1. Konzept

Die wesentliche Herausforderung dieser Bachelorarbeit ist die Entwicklung eines modular aufgebauten, flexibel konfigurierbaren und universell wiederverwendbaren Steuerungssystem. Beim Entwurf wird 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 [21]. Um die Modularität des Systems zu gewährleisten, werden 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 die Rückmeldung 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 muss in der Nutzer Anwendung eingebunden werden.

Flexibilität

Die Steuerelemente werden ohne neue Übersetzung der Android Applikation eingefügt oder gelöscht. Der Nutzer soll in der Raspberry Pi Anwendung die auszuführende Aktion festgelegen.

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.

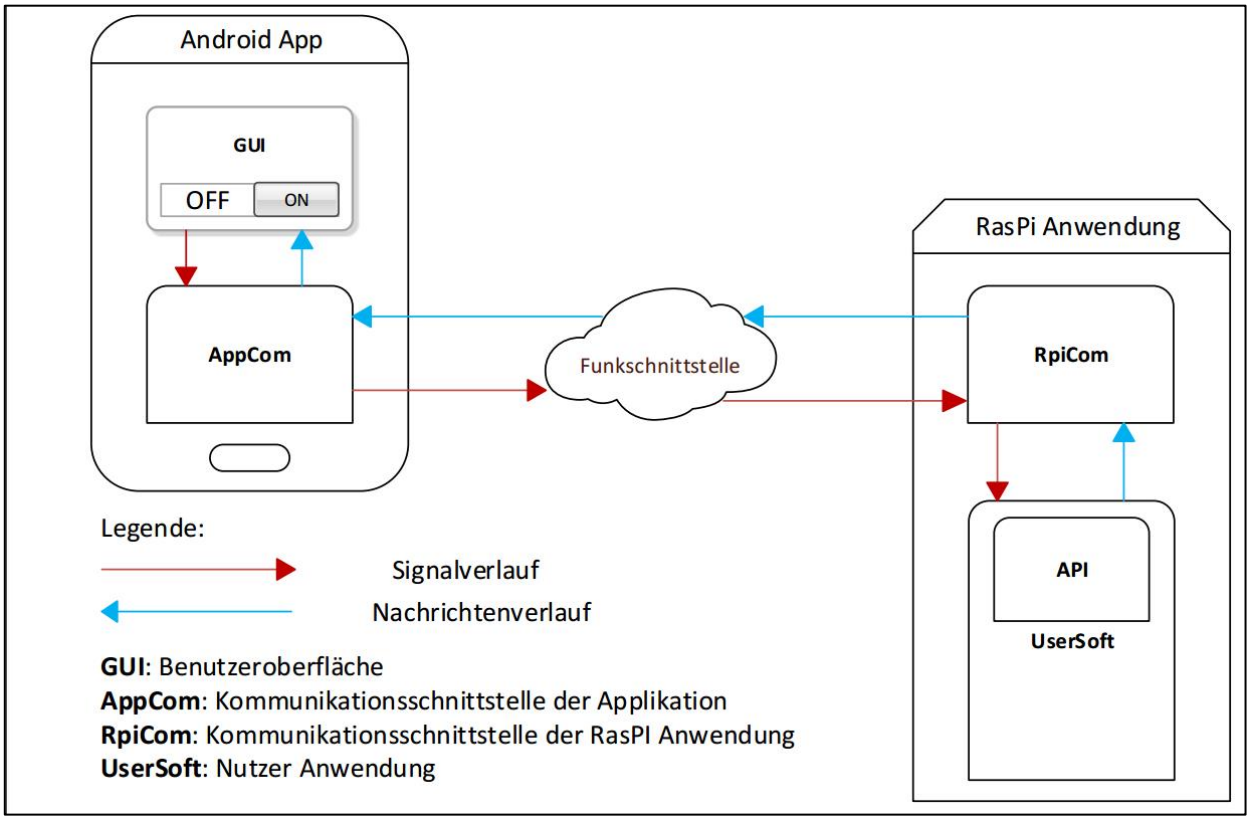


Abbildung 14: Aufbau des Steuerungssystems und Signalablauf

3.2. Festlegungen zum Entwurf

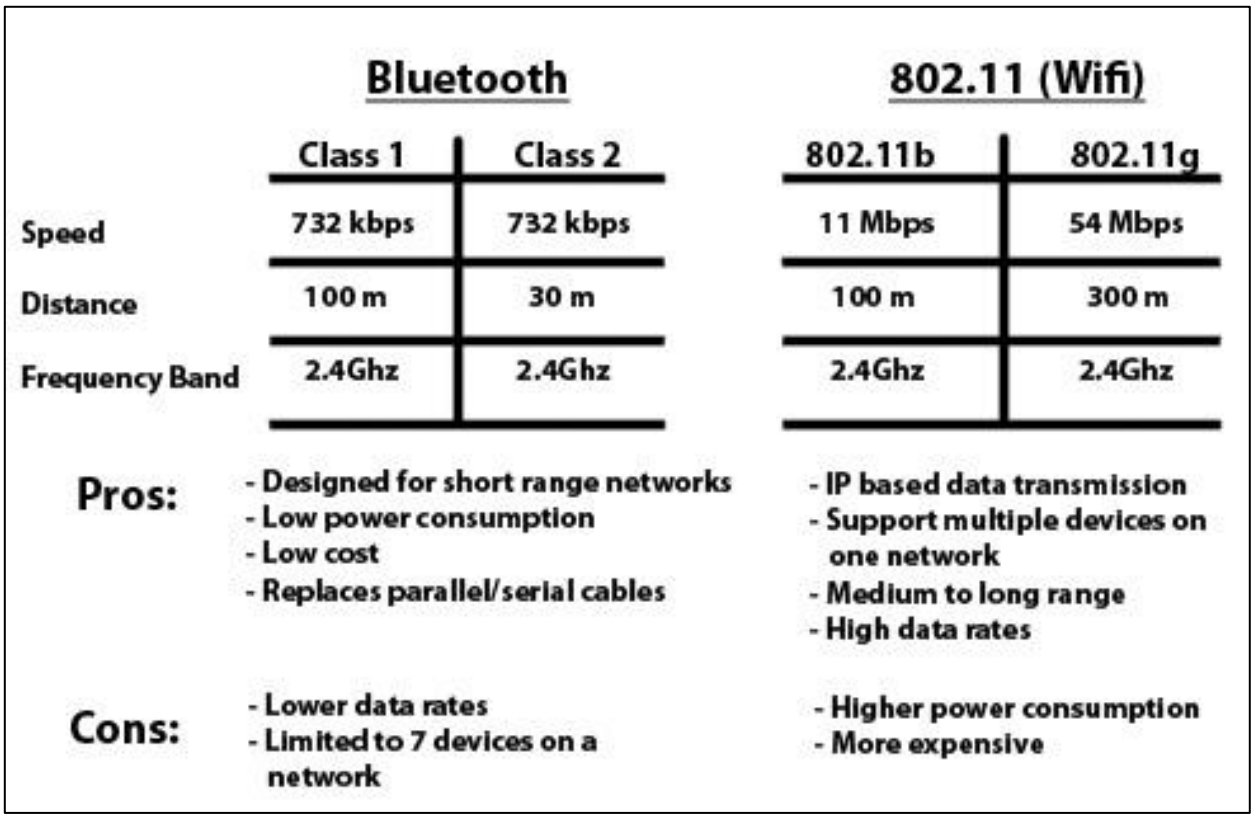
In diesem Kapitel werden begründete Entscheidungen für den Entwurf getroffen.

3.2.1. 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.



	Bluetooth		802.11 (Wifi)	
	Class 1	Class 2	802.11b	802.11g
Speed	732 kbps	732 kbps	11 Mbps	54 Mbps
Distance	100 m	30 m	100 m	300 m
Frequency Band	2.4Ghz	2.4Ghz	2.4Ghz	2.4Ghz
Pros:	- Designed for short range networks - Low power consumption - Low cost - Replaces parallel/serial cables		- IP based data transmission - Support multiple devices on one network - Medium to long range - High data rates	
Cons:	- Lower data rates - Limited to 7 devices on a network		- Higher power consumption - More expensive	

Abbildung 15: Vergleich Bluetooth und WLAN [22]

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

Für diese Arbeit wird 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 diesem 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 [23]. Jedoch unterstützen nicht alle Android-Smartphones diese Technologie und laut [24] 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. Der RasPi fungiert 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 wird 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.

3.2.2. Raspberry Pi Anwendung

Dieser Abschnitt behandelt die Entscheidung über Module, 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 wird Apache Web

Server[5] verwendet. Die Signale werden dann von der Android Applikation als HTTP-Anfrage mit Parametern gesendet. HTTP-Anfragen ermöglichen generell dem Client Informationen zum Webserver zu senden, um Formulare abzusenden, Dateien hochzuladen oder Dateien abzurufen [25].

```
http://www.webserver-adress.com/file_to_call.cgi?param1=val1+param2=val2+param3=val3
```

Abbildung 16: 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] in das CGI-Skript eingegeben [26].

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

Auf [27] 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.

[5] https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md [10.02.2018]
[6] https://de.wikipedia.org/wiki/Umgebungsvariable [13.02.2018]

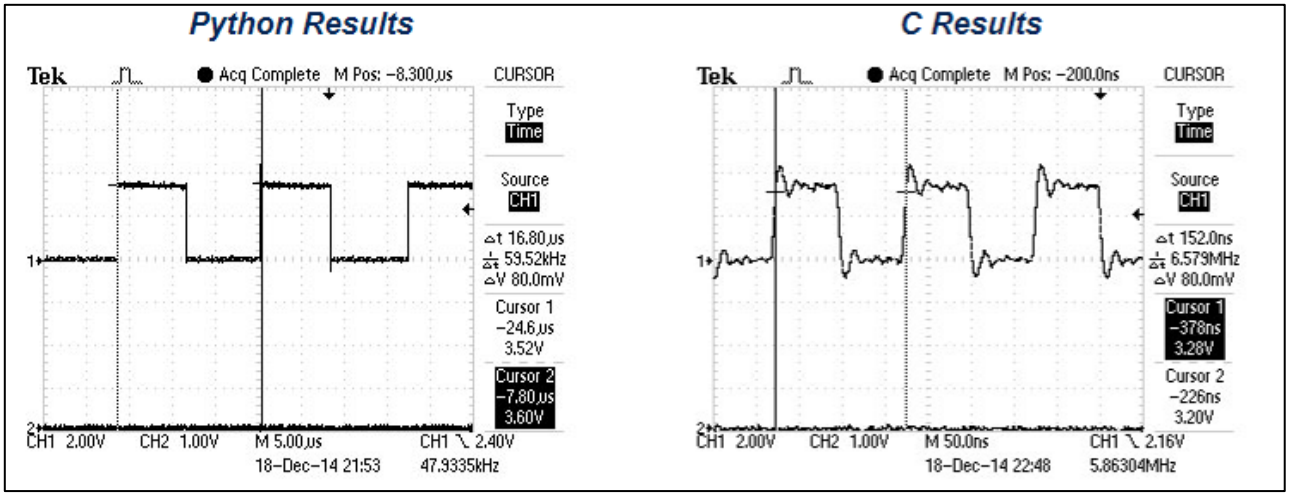


Abbildung 17: Leistung Python und C [27]

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 werden 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.

3.2.3. 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.

4. 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.

4.1. 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

4.2. Raspberry Pi Anwendung

4.2.1. API

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

Methode Name	Beschreibung
get	Gibt das Skript-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 1: 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.

```
UserSoftware.py “pin_number=5” “state=1” “mode=11” signal_type=0” “number_of_cycles=5”
```

Abbildung 18: 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).

```
<p class="userapp debug">2018-02-25 12:12:14 __main__</p>
<p class="userapp debug">2018-02-25 12:12:14 get(name)</p>
<p class="userapp debug">2018-02-25 12:12:14 key: 'name' - value: 'SwitchControl'</p>
<p class="userapp debug">2018-02-25 12:12:14 get(mode)</p>
<p class="userapp debug">2018-02-25 12:12:14 key: 'mode' - value: '11'</p>
<p class="userapp debug">2018-02-25 12:12:14 get(pin_number)</p>
<p class="userapp debug">2018-02-25 12:12:14 key: 'pin_number' - value: '5'</p>
<p class="userapp debug">2018-02-25 12:12:14 get(state)</p>
<p class="userapp debug">2018-02-25 12:12:14 key: 'state' - value: '0'</p>
<p class="userapp debug">2018-02-25 12:12:14 Port 5 cleaned</p>
```

Abbildung 19: 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 besitzt Zwei PWM Ports. Damit das Steuerungssystem nicht darauf beschränkt wird, ist die PWM in software implementiert. In der *PwmDaemon.py*-Datei wird die PWM mit einer Software Schleife emuliert. Damit die Schleife die Anwendung nicht blockiert, wird sie in einem separaten Thread ausgeführt. Der Thread ist mithilfe der Daemon[7] Bibliothek implementiert.

4.2.2. Kommunikationsschnittstelle

Die Schnittstelle leitet Signale an die Nutzer Anwendung weiter und stellt seine Ausgabe zur Verfügung (siehe Abbildung 19). Sie 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.

[7] https://pypi.python.org/pypi/python-daemon/ [13.02.18]

Here are the CGI environment variables/value pairs passed in from the CGI script:

CONTENT_TYPE
::
CONTENT_LENGTH
::
REQUEST_METHOD
:*GET*:
QUERY_STRING
:*state=0&pin_number=5&direction=0&mode=11&name=SwitchControl*:
SERVER_NAME
:*192.168.1.101*:
SERVER_PORT
:*80*:
SCRIPT_NAME
:*/cgi-bin/RpiComIface.cgi*:
PATH_INFO
::

Here is the user command used to run user software:

python /usr/lib/cgi-bin/UserApp.py state=0 pin_number=5 direction=0 mode=11 name=SwitchControl

Here is the user software output:

2018-02-25 12:17:15 __main__
2018-02-25 12:17:15 get(name)
2018-02-25 12:17:15 key: 'name' - value: 'SwitchControl'
2018-02-25 12:17:15 get(mode)
2018-02-25 12:17:15 key: 'mode' - value: '11'
2018-02-25 12:17:15 get(pin_number)
2018-02-25 12:17:15 key: 'pin_number' - value: '5'
2018-02-25 12:17:15 get(state)
2018-02-25 12:17:15 key: 'state' - value: '0'
2018-02-25 12:17:15 Port 5 cleaned

Abbildung 20: 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 die Nutzer Anwendung aus
getProcessOutput	Stellt die Ausgabe der Nutzer Anwendung auf HTML Seite dar

Tabelle 2: 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, wird die Java Bibliothek namens *CgiLib.java*[8] bereitgestellt.

4.3. 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 ist 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).

[8] https://www.javaworld.com/article/2076863/java-web-development/write-cgi-programs-in-java.html [13.02.18]

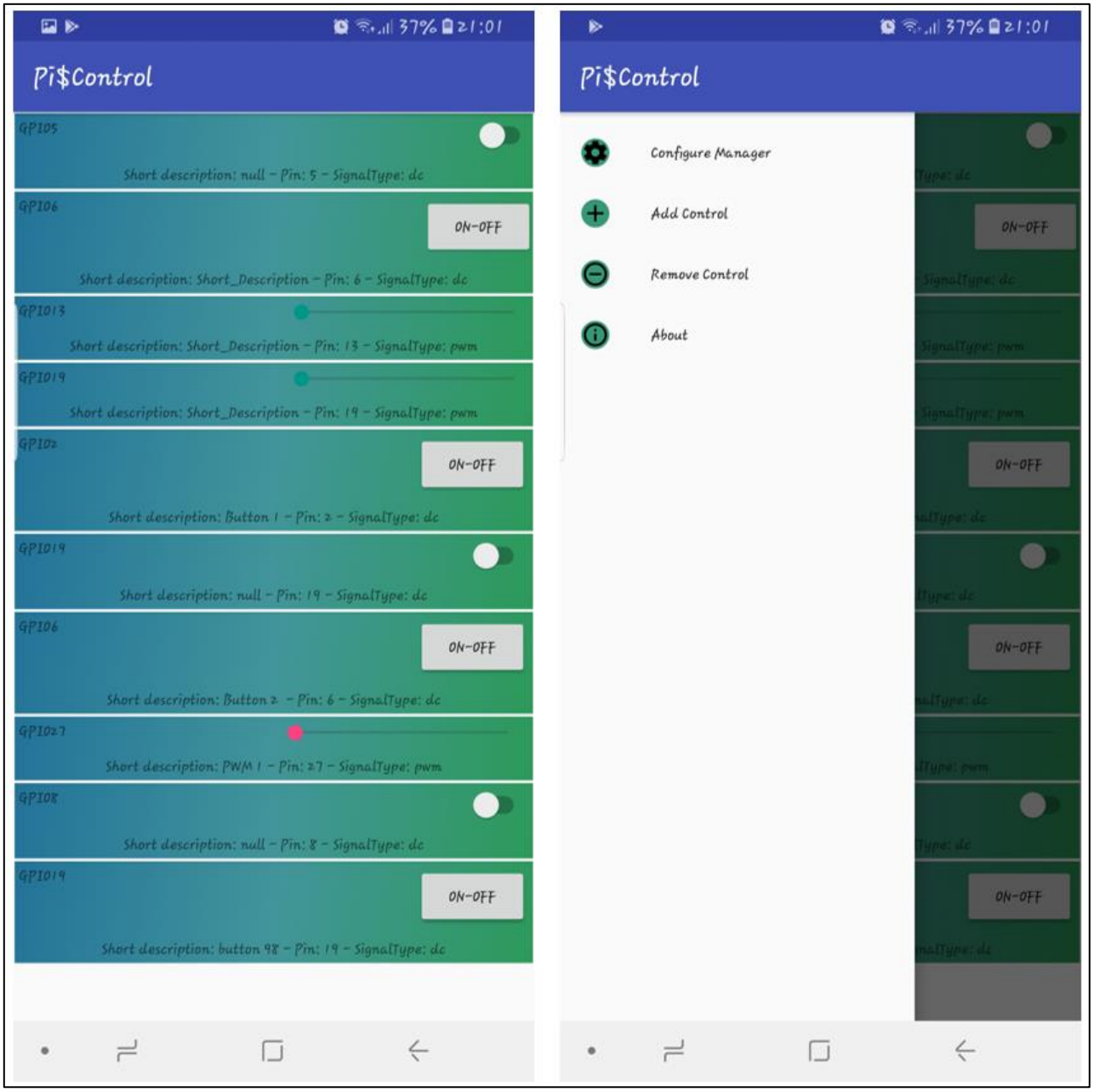


Abbildung 21: Android Applikation GUI

4.3.1. 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 22).

- 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.

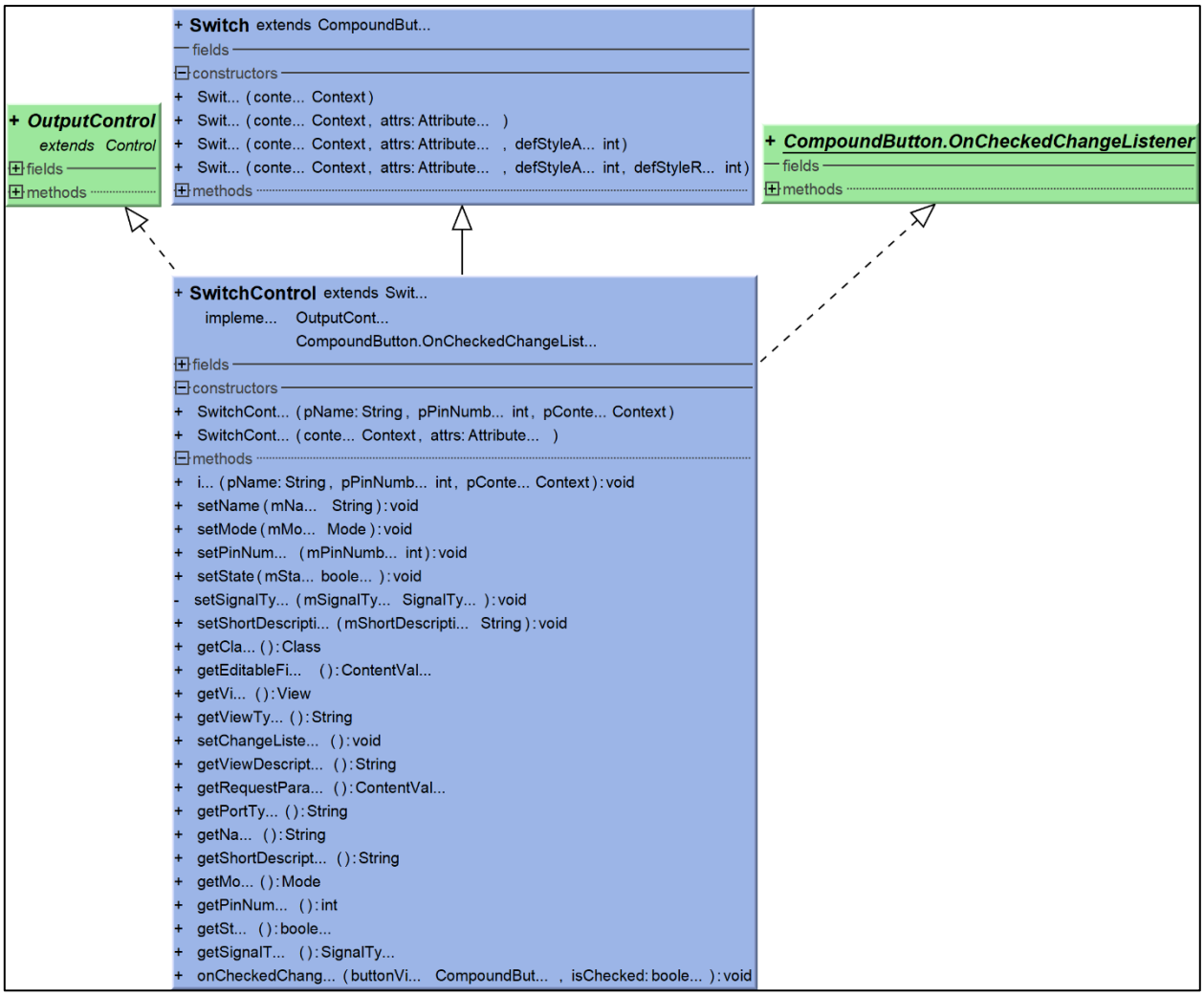


Abbildung 22: *SwitchControl* Klassendiagramm

Implementierung von *Control*-Unterinterface

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

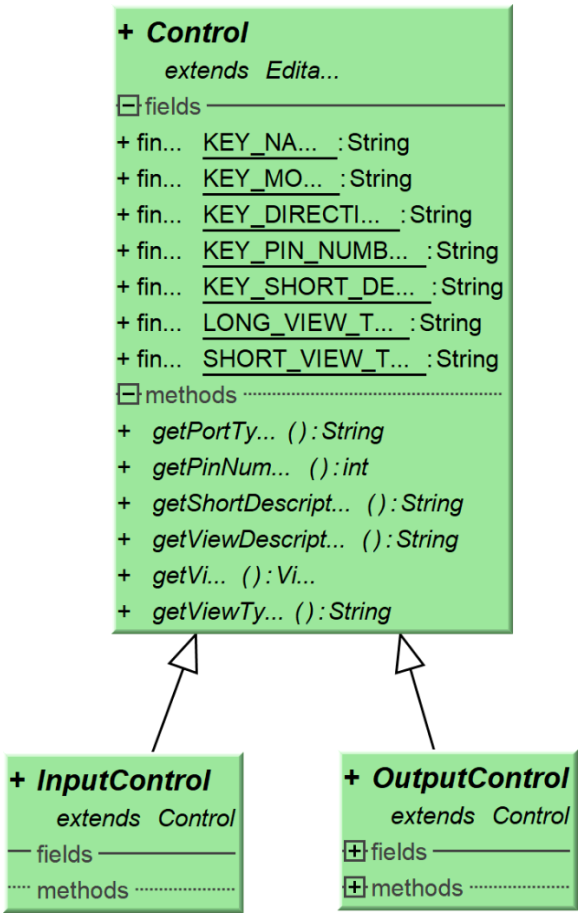


Abbildung 23: Klassendiagramm von *Control*

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 3: 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. Die 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 ist OnCheckedStateChangedListener-Interface registriert werden (siehe Abbildung 22).

4.3.2. 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.

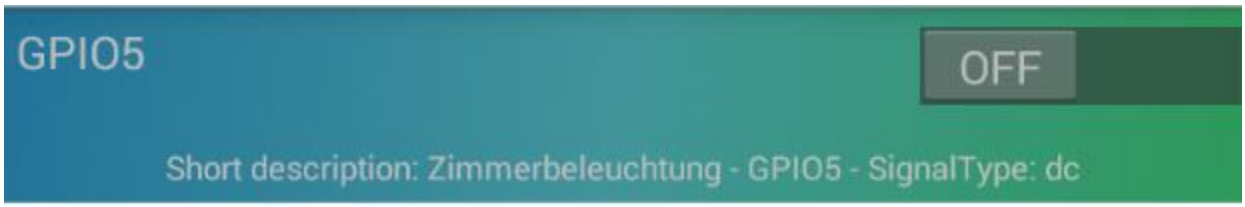


Abbildung 24: Anzeige eines Steuerelements (*SwitchControl*)

4.3.3. Signalübertragung

Klassen die für die Kommunikation zuständig sind, befinden sich im Paket *communication*.

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] ausgeführt. Die Klasse *AsyncTask*[10] vom Android Framework erleichtert die Gestaltung von Thread.

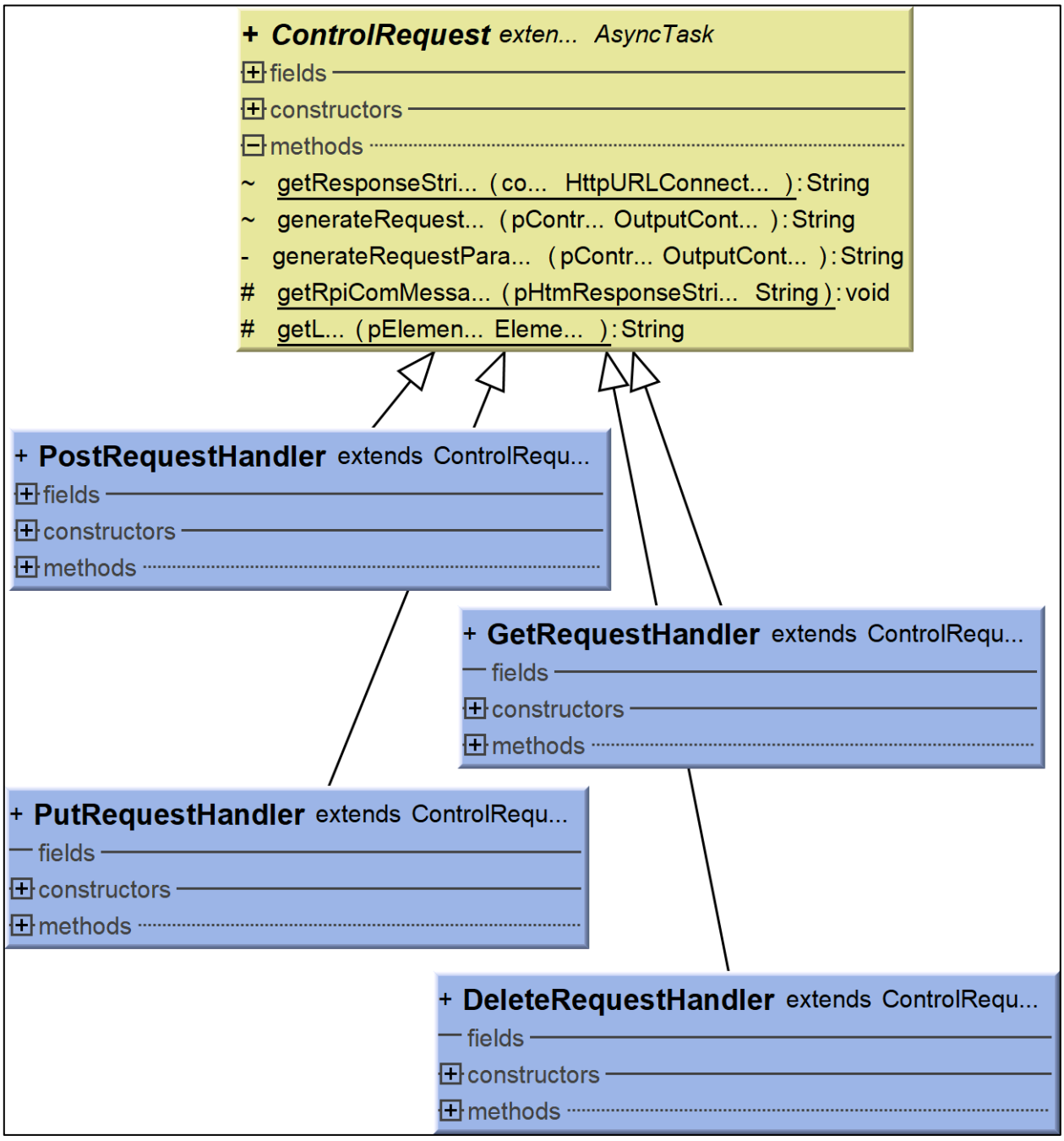


Abbildung 25: Klassendiagramm *communication*-Paket

[9] https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html [16.02.18]
[10] https://developer.android.com/reference/android/os/AsyncTask.html [16.02.2018]

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.

```
// For the connection
public static final String URL_SEPARATOR = "/";
public static final String URL_PARAM_SEPARATOR = "?";
public static final String FORM_PARAM_SEPARATO = "&";
public static final String KEY_VALUE_SEPARATOR = "=";
public static final String PROTOCOL = "http://";
public static final String RPI_COM_IFACE_NAME = "RpiComIface.cgi";
public static final String CGI_BIN_PATH = "/cgi-bin/";
// For Logging
public static final String LOG_LINE_TAG_NAME = "p";
public static final String COM_MSG_ID = "comMessage";
public static final String USER_APP_CLASS_NAME = "userapp";
public static final String USER_DEBUG_CLASS_NAME = USER_APP_CLASS_NAME + " debug";
public static final String USER_INFO_CLASS_NAME = USER_APP_CLASS_NAME + " info";
public static final String USER_WARN_CLASS_NAME = USER_APP_CLASS_NAME + " warn";
public static final String USER_TODO_CLASS_NAME = USER_APP_CLASS_NAME + " todo";
public static final String USER_ERROR_CLASS_NAME = USER_APP_CLASS_NAME + " error";
public static final String USER_NONAME_CLASS_NAME = USER_APP_CLASS_NAME + " noname";
// HTTP Methoden
public static final String METHOD_POST = "POST";
public static final String METHOD_GET = "GET";
public static final String METHOD_PUT = "PUT";
public static final String METHOD_DELETE = "DELETE";
```

Abbildung 26: 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 die URL generiert, das Signal Übertragen und die Ausgabe der Nutzer Anwendung empfangt.

```
@Override
protected String doInBackground(String... params) {
    try {
        // Generate URL
        URL postUrl = new URL(generateRequestUrl(mControl));
        mConnection = (HttpURLConnection) postUrl.openConnection();
        // Setup Connection
        mConnection.setRequestMethod(METHOD_POST);
        mConnection.setReadTimeout(10000);
        mConnection.setConnectTimeout(5000);
        // Send Request
        mConnection.connect();
        // Get Response
        mResponseString = getResponseString(mConnection);
    } catch (MalformedURLException e) {
        Toast.makeText(mContext, e.getMessage(), Toast.LENGTH_LONG).show();
    } catch (IOException e) {
        Toast.makeText(mContext, e.getMessage(), Toast.LENGTH_LONG).show();
    }
    return null;
}
```

Abbildung 27: Überschreibung der *doInBackground*-Methode in *PostRequestHandler*-Klasse

4.3.4. Einfügen von Steuerelement (*Control*)

Die Implementierung dieser Funktionalität erhält zwei zwischenschritte. Zuerst werden 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 sind in *Port*-Klasse und ihre Unterklassen repräsentiert.

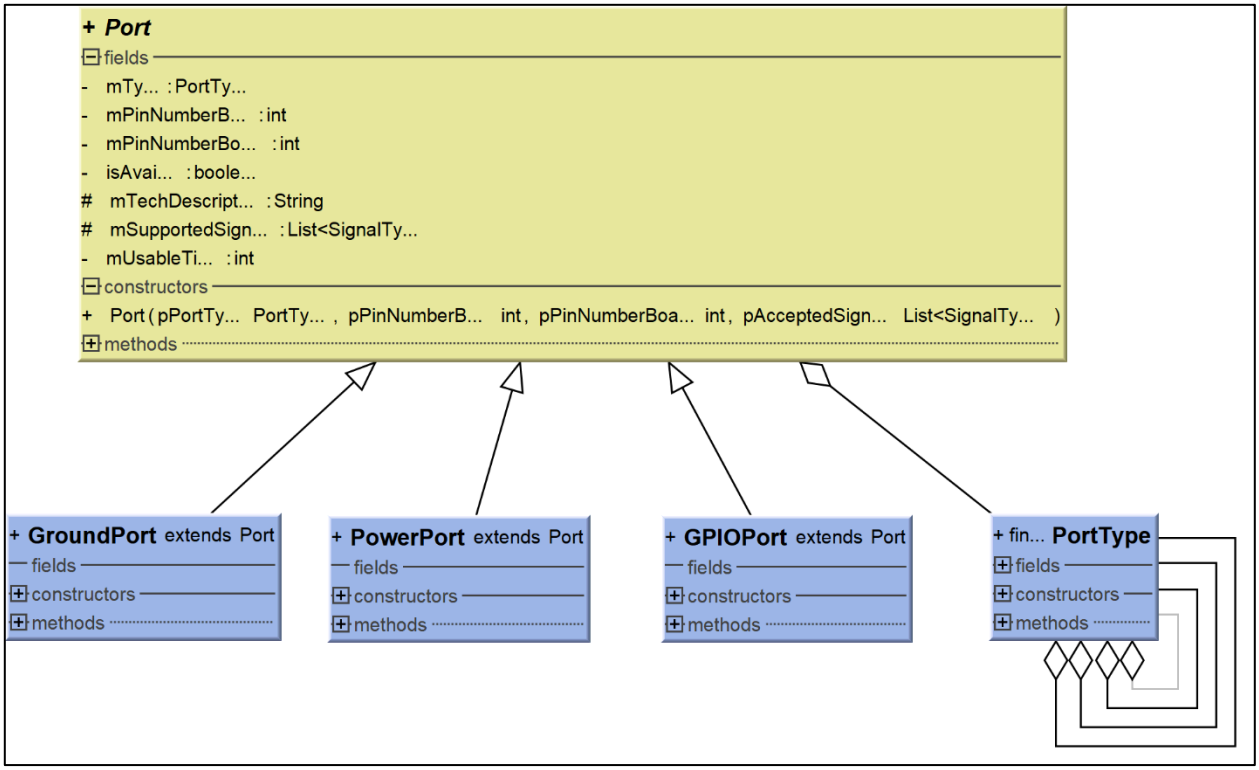


Abbildung 28: Klassendiagramm von *Ports*

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

GPIO-Ports Verwaltung

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

[11] https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html [17.02.18]
[12] https://de.wikipedia.org/wiki/Singleton_(Entwurfsmuster) [17.02.18]

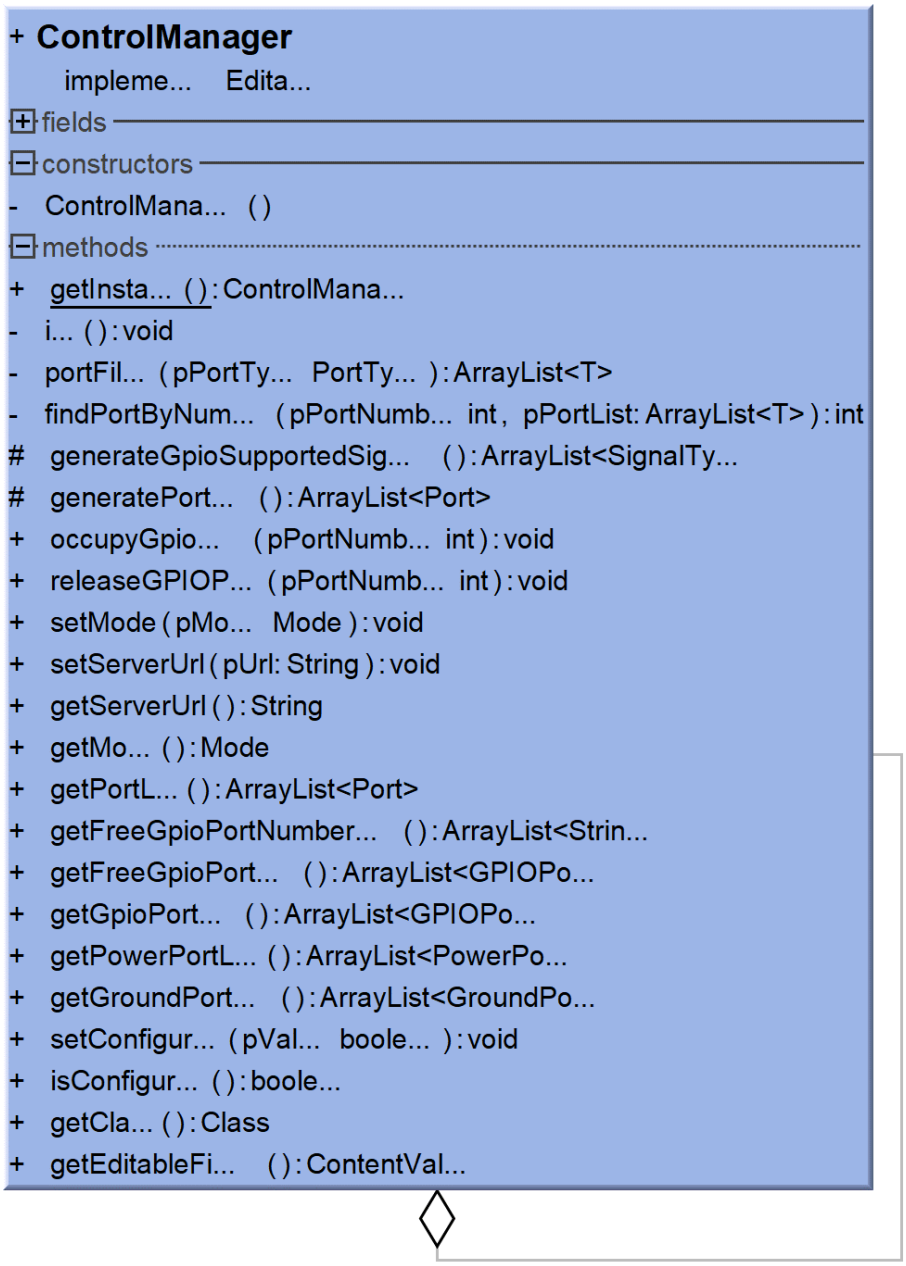


Abbildung 29: Klassendiagramm *ControlManager*

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 4: 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. Er wird als FQN[13] übergeben (vgl. Abbildung 30).

```
@Override
public Class getClazz() { return getClass(); }

@Override
public ContentValues getEditableAttributes() {
    ContentValues result = new ContentValues();
    result.put(KEY_NAME, String.class.getName());
    result.put(KEY_PIN_NUMBER, Integer.class.getName());
    result.put(KEY_FREQUENCE, Integer.class.getName());
    result.put(KEY_DUTY_CYCLE, Integer.class.getName());
    result.put(KEY_PWM_OUTPUT_TYP, PwmOutputType.class.getName());
    result.put(KEY_SHORT_DESC, String.class.getName());
    return result;
}
```

Abbildung 30: 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] erstellt.

[13] https://en.wikipedia.org/wiki/Fully_qualified_name [24.02.2018]
[14] https://developer.android.com/guide/topics/ui/dialogs.html [17.02.2018]

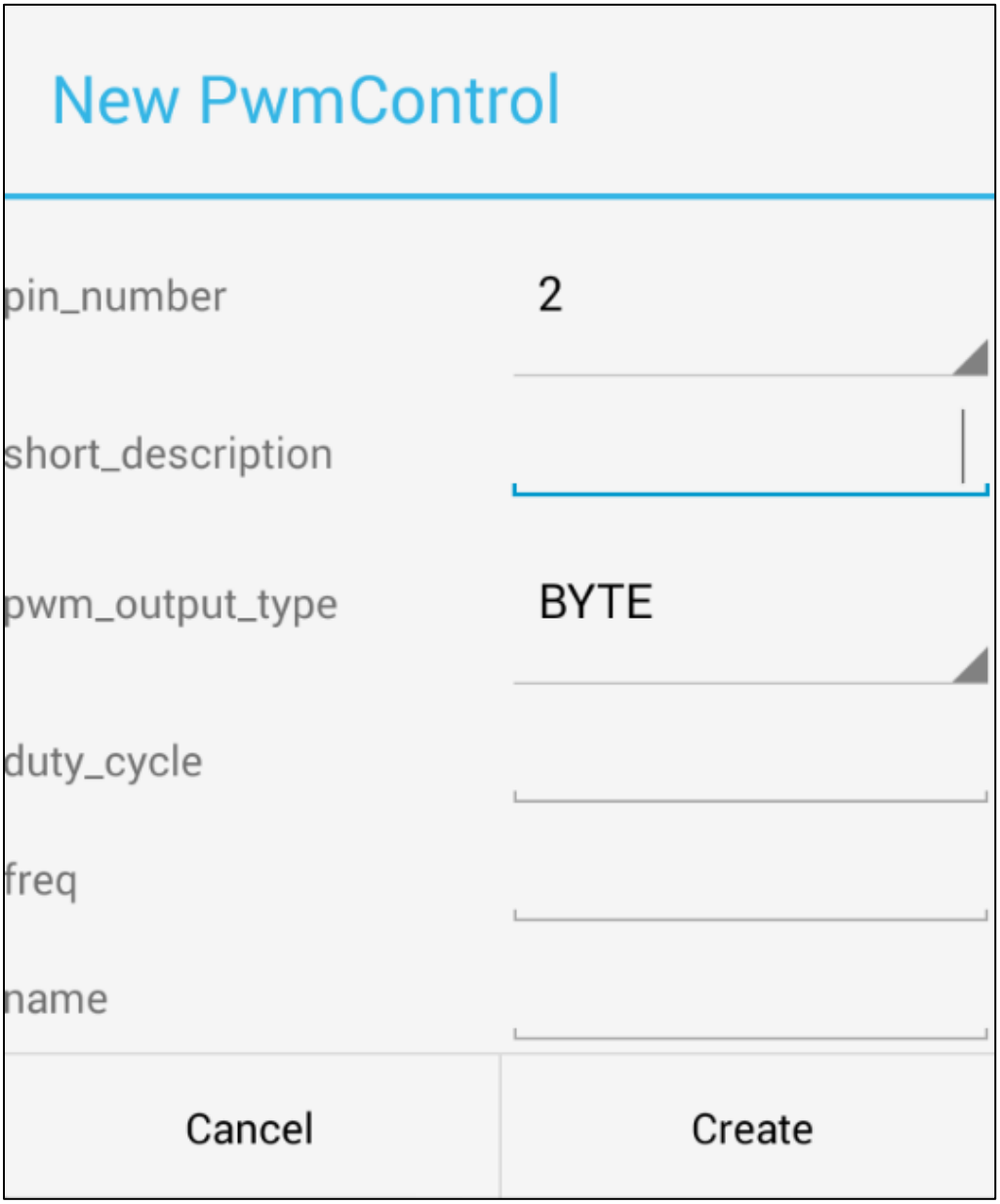


Abbildung 31: Dialog zur Erstellung eines *PwmControl*

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 32: Generiertes Layout fürs Attribut Pin Nummer

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

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

```
private void setupValueView() {
    String val = mPair.getValue().toString();
    // generate Spinner for pin selection; the spinner show only available pins;
    // the available pins depend on the pinUsableTime in Port class
    if(mPair.getKey().equals(Control.KEY_PIN_NUMBER)) {
        mValue = new Spinner(mContext);
        ArrayAdapter<String> adapter = new ArrayAdapter<>(mContext,
                android.R.layout.simple_spinner_dropdown_item,
                ControlManager.getInstace().getFreeGpioPortNumberList());
        Spinner v = (Spinner) mValue;
        v.setAdapter(adapter);
    }
```

Abbildung 33: Behandlung des Eingabefeldes fürs Pin Nummer

Bei Enum Datentyp wird auch ein Spinner generiert.

```
// generate spinner when value equals FQN of Enum
else if (val.equals(Enum.class.getName())) {
    mValue = new Spinner(mContext);
    ArrayAdapter<String> adapter = new ArrayAdapter<>(mContext,
            android.R.layout.simple_spinner_dropdown_item, getEnumValues(val));
    Spinner v = (Spinner) mValue;
    v.setAdapter(adapter);
```

Abbildung 34: Behandlung von Enum Eingabefeldern

[15] https://developer.android.com/guide/topics/ui/controls/spinner.html [17.02.2018]

Wird die Eingabe mit „Create“ (siehe Abbildung 31) bestätigt, erfolgt die Erstellung des Steuerelements. Die Erstellung übernimmt die *ControlFactory*-Klasse. Die Klasse ist nach dem Factory[16] Entwurfsmuster implementiert.

[16] https://www.tutorialspoint.com/design_pattern/factory_pattern.htm [17.02.2018]

5. Implementierung

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

5.1. Raspberry Pi

5.1.1. 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) wird auf Herstellerseite[17] heruntergeladen, aus dem Archiv entpackt und mit einem zusätzlichen Programm (Etcher[18]) auf die microSD-Karte kopiert (installiert).

Eine SSH-Verbindung mit einem Laptop wird dann eingerichtet, sodass der Raspberry Pi selbst auf Tastatur und Monitor verzichten kann. Die Anleitung befindet sich auf [29].

Nach dem Startvorgang erflogt die Anmeldung unter dem Benutzernamen „pi" und das Passwort „raspberry".

5.1.2. WLAN Access Point - Einrichtung

Die Installation und Einrichtung des Access Point erfolgte nach der Anleitung auf [30]. Es handelt sich um ein PDF-Datei. Eine bearbeitete Version befindet sich im Anhang unter **/Anleitungen/wlanAP.pdf**. Die Datei beinhaltet Kommentare, die Besonderheiten oder Unterschiede zu dieser Bachelorarbeit kennzeichnen.

Das Passwort der Access Point ist **raspberrypi** und der Name ist **Pi_AP**

5.1.3. Webserver: Apache2 – Installation und Einrichtung

Apache Server (Version 2.4.29) wird durch Ausführen folgender Kommando in Terminal installiert: *sudo apt-get install apache-y* [31].

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

[17] https://downloads.raspberrypi.org/raspbian/images/raspbian-2017-07-05/ [18.02.18]
[18] https://etcher.io/?ref=etcher_update [18.02.18]

den Webserver-Clients erweitert werden. Dies erfolgt durch Ausführen folgendes Kommando: *sudo adduser www-data gpio.*

5.1.4. Module

Die Entwicklungsumgebung der Kommunikationsschnittstelle und der API ist ein Texteditor namens Geany[19]. 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): Für den Zugriff auf GPIO-Pins. Sie kann mit dem Befehl *sudo apt-get install python dev python-rpi.gpio* installiert werden.
- **sys**: Für den Zugriff auf Skript Argumente
- **datetime**: Für Datum und Zeit Ausgabe
- **time**: Zum Anhalten der Anwendung
- **runner**: für PWM Behandlung. Sie befindet sich in daemon[20] Paket.

Die Kommunikation Schnittstelle wird 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/**.
- **RpiComIface.cgi**: Empfängt die Anfrage und leitet sie an die Kommunikationsschnittstelle weiter. Die Datei befindet sich im Verzeichnis **/RasPi/**

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

5.2. Android Applikation

Die Applikation wird mit der Entwicklungsumgebung Android Studio (Version 2.3.3) entworfen. Die Umgebung ist von Google und für Android veröffentlich und integriert schon alle Komponenten, die Für die Entwicklung notwendig sind. Zusätzliche

[19] https://www.geany.org/Main/HomePage [18.02.2018]
[20] https://pypi.python.org/pypi/daemon-runner [18.02.2018]

Bibliotheken können im *build.gradle* Datei eingebunden werden [32]. Für diese Bachelorarbeit werden folgende Bibliotheken eingebunden:

- **Jsoup**[21] (Version 1.11.2): vereinfacht die Bearbeitung der RasPi Rückmeldung.
- **Design Support Library**[22] (Version 25.3.1): Fürs App-Menü

Die Anwendung ist mit Android SDK Version 25 kompiliert und kann unter Android Version 4.4 oder neuer laufen.

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

[21] https://github.com/jhy/jsoup [18.02.2018]
[22] https://developer.android.com/topic/libraries/support-library/packages.html#design [18.02.2018]
[23] https://www.genymotion.com/ [18.02.2018]

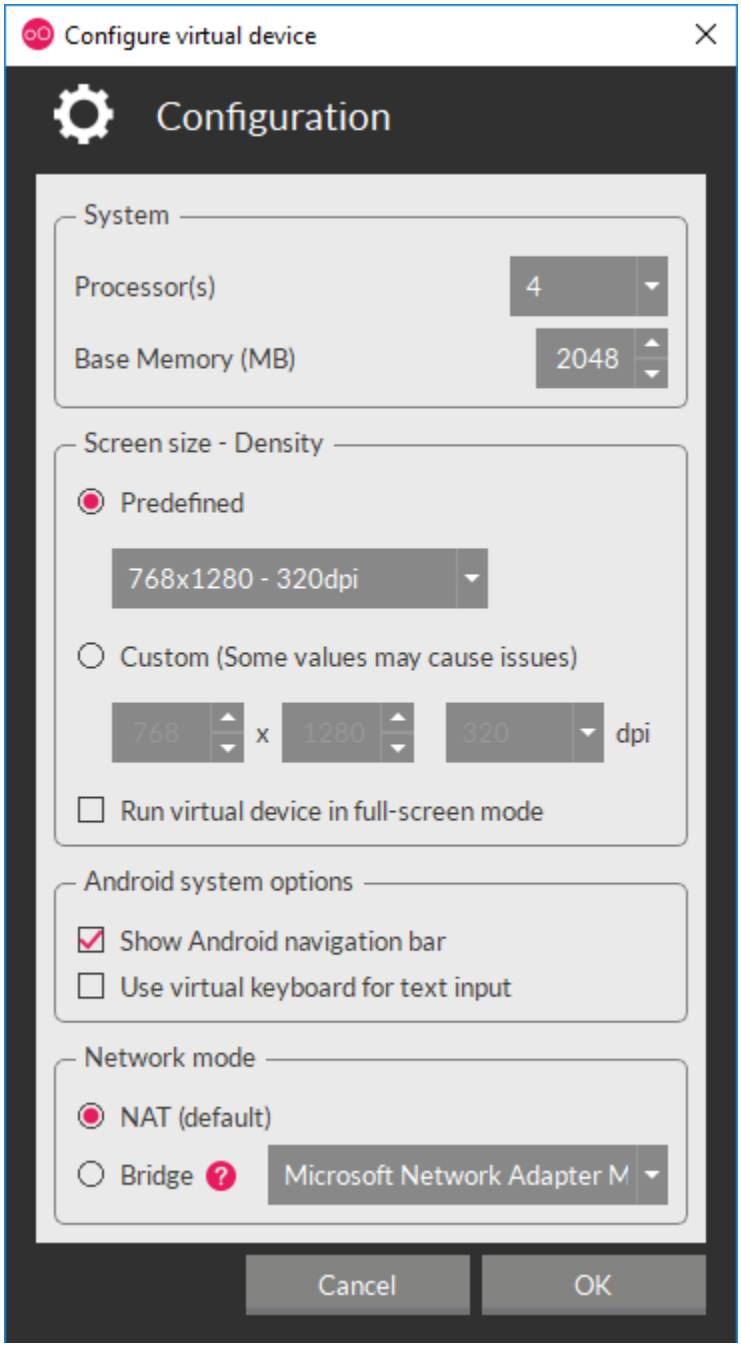


Abbildung 35: Eigenschaften des virtuellen Gerätes

Auf dem virtuellen Gerät ist Android 4.4.2 installiert.

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

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

6. Test

Die vollständig implementierten Funktionalitäten des Steuerungssystems werden in diesem Kapitel getestet. Es beinhaltet zwei Schritten: der Aufbau und die Bewertung der Ergebnisse.

6.1. Test Aufbau

Das Unterkapitel umfasst den Aufbau und die Einrichtung jedes Gerätes.

Abbildung 36: Test Aufbau

Android Smartphone

Die Anwendung wird auf ein Samsung Galaxy S8 Plus installiert. Für den Test sind vier Steuerelementen vorhanden (vgl. Abbildung 37).

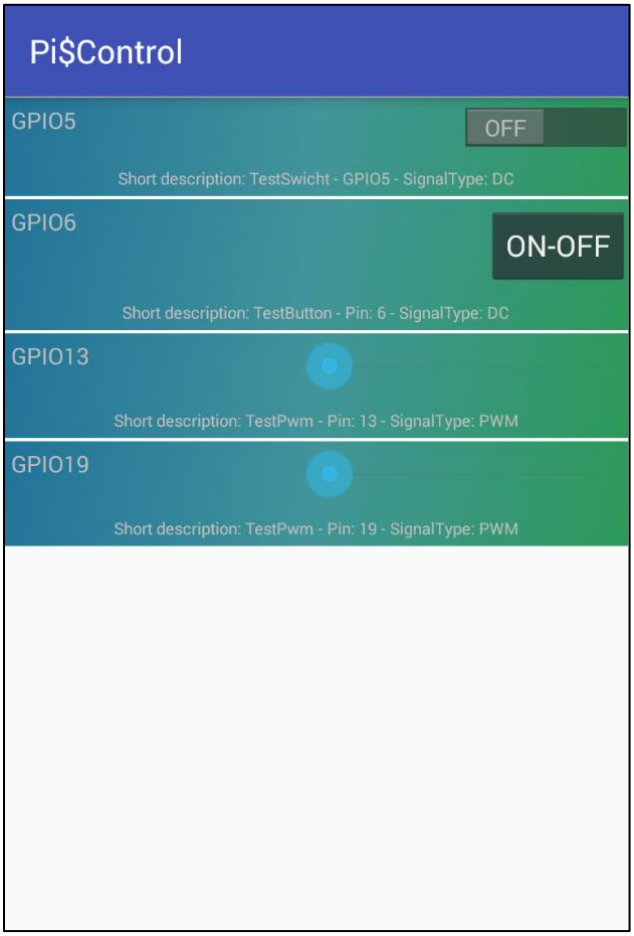


Abbildung 37: Android App für den Test

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.

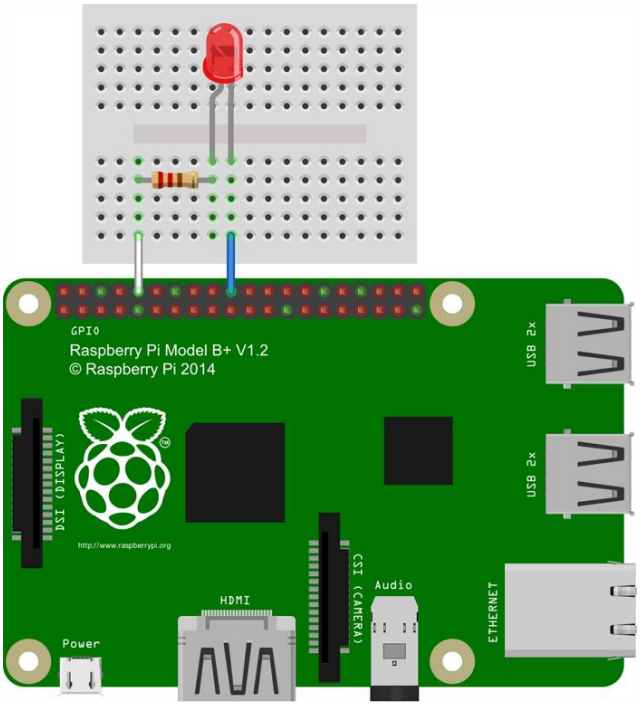


Abbildung 38: Verbindung einer LED mit einem GPIO-Port

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

```
#!/usr/bin/python

import PiControlApi as API


# start of programm
if __name__ == "__main__":
        API.log("d", "__main__")
        if(API.getName() == API.SWITCH_CONTROL_NAME):
                API.handleSwitchControl()
        elif(API.getName() == API.BUTTON_CONTROL_NAME):
                API.handleButtonControl()
        elif(API.getName() == API.BLINK_CONTROL_NAME):
                API.handleBlinkControl()
        elif(API.getName() == API.PWM_CONTROL_NAME):
                API.handlePwmControl()
        else:
                API.log("t", "Control Handler not implemented yet...")
```

Abbildung 39: Raspberry Pi Test Anwendung

6.2. Bewertung der Ergebnisse

Die LED von GPIO5 geht an und aus, wenn der Schalter Ein- beziehungsweise Ausschaltet wird.

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 ist erfolgreich durchgeführt.

Die implementierten Funktionalitäten des Steuerungssystems funktionieren wie erwartet.

7. Erreichter Stand & Ausblick

7.1. Erreichter Stand

Anhand der Kriterien, die im Pflichtenheft (Kapitel 4.1) aufgelistet sind, 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.

7.2. 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 wird.

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 ein 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 B 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.

Quellenverzeichnis

[1] M. v. d. B. I. K. Manfred Broy, Problemanalyse für ein Großverbundprojekt Systemtechnik Automobil - Software für eingebettete Systeme, Springer-Verlag, 1998.

[2] „raspberrypi - make an android app to control GPIO Pins," [Online]. Available: https://www.raspberrypi.org/forums/viewtopic.php?t=101330. [Zugriff am 06 02 2018].

[3] „PIREPLAY," 2013. [Online]. Available: https://pirelay.jasonfindlay.com/. [Zugriff am 06 02 2018].

[4] „Google Play - BlueTerm," [Online]. Available: https://play.google.com/store/apps/details?id=es.pymasde.blueterm&hl=fr. [Zugriff am 06 02 2018].

[5] „Wikipedia - Raspberry Pi," 18 01 2018. [Online]. Available: https://fr.wikipedia.org/wiki/Raspberry_Pi. [Zugriff am 06 02 2018].

[6] C. K. C. S. Micheal Kofler, Raspberry Pi - Das umfassende Handbuch, Rheinwerk, 2017.

[7] „Circuits.dk - Everything You Want to Know About Raspberry Pi GPIO: But Were Afraid to Ask," 20 06 2017. [Online]. Available: https://www.circuits.dk/everything-about-raspberry-gpio/. [Zugriff am 06 02 2018].

[8] Z. Simic, „Hard- und Softwareentwicklung für eine autonome Omnirad Plattform auf Basis eines eingebetteten Linux-Systems," in *Bachelorarbeit*, Berlin, HTW-Berlin, 2018.

[9] „heise - Raspberry Pi 3 mit WLAN, Bluetooth und 64 Bit," [Online]. Available: https://www.heise.de/newsticker/meldung/Raspberry-Pi-3-mit-WLAN-Bluetooth-und-64-Bit-3119537.html.

[10] „Wikipedia - Betriebssystem," [Online]. Available: https://de.wikipedia.org/wiki/Betriebssystem. [Zugriff am 06 02 2018].

[11] „Raspberry Pi Geek - Neues Betriebssystem Minoca OS," [Online]. Available: http://www.raspberry-pi-geek.de/Magazin/2017/04/Neues-Embedded-Betriebssystem-Minoca-OS. [Zugriff am 20 02 2018].

[12] „phonearena," [Online]. Available: https://www.phonearena.com/phones/Samsung-Galaxy-S8_id10311. [Zugriff am 06 02 2018].

[13] „Wikipedia - Android," [Online]. Available: https://fr.wikipedia.org/wiki/Android. [Zugriff am 06 02 2018].

[14] „3DCenter Forum - IDC Hersteller Smartphone Marktanteil," 03 11 2016. [Online]. Available: https://goo.gl/images/SXCXpz. [Zugriff am 06 02 2018].

[15] „Sigma," [Online]. Available: http://www.sigmadesigns.com/uploads/library/android.jpg. [Zugriff am 06 02 2018].

[16] A. Becker und M. Pant, Android 2., dpunkt.verlag, 2010.

[17] „Einführung in die Entwicklung von Android-Apps," Universiät Trier, [Online]. Available: https://www.uni-trier.de/fileadmin/urt/doku/android/android.pdf. [Zugriff am 06 02 2018].

[18] „Android developers - Activity Lifecycle," [Online]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle.html. [Zugriff am 07 02 2018].

[19] „Android developers - Layouts," [Online]. Available: https://developer.android.com/guide/topics/ui/declaring-layout.html. [Zugriff am 07 02 2018].

[20] T. Baar, „Einführung in Android," 2017.

[21] „Wikipedia - Modularität," [Online]. Available: http://de.wikipedia.org/wiki/Modularität. [Zugriff am 08 02 2018].

[22] „fyicenter - Differences between WIFI and Bluetooth," [Online]. Available: http://phone.fyicenter.com/59_Technology_Differences_between_WiFi_and_Bluetooth.html. [Zugriff am 09 02 2018].

[23] „Wikipedia - Wi-Fi Direct," [Online]. Available: https://de.wikipedia.org/wiki/Wi-Fi_Direct. [Zugriff am 09 02 2018].

[24] „nextthing - Could WiFi work in WiFi-Direct mode (no router)?," [Online]. Available: https://bbs.nextthing.co/t/could-wifi-work-in-wifi-direct-mode-no-router/14876/5. [Zugriff am 09 02 2018].

[25] „selfhtml - HTTP/Anfragemethoden," [Online]. Available: https://wiki.selfhtml.org/wiki/HTTP/Anfragemethoden. [Zugriff am 10 02 2018].

[26] „JAVAWORLD - Write CGI programms in Java," [Online]. Available: https://www.javaworld.com/article/2076863/java-web-development/write-cgi-programs-in-java.html. [Zugriff am 13 02 2018].

[27] „spazztech - python vs c on rpi," [Online]. Available: http://www.spazztech.net/python-vs-c-on-rpi.html. [Zugriff am 10 02 2018].

[28] „Stackoverflow - raspberry pi python pwm continue after programm done," [Online]. Available: https://stackoverflow.com/a/29089425/7614256. [Zugriff am 13 02 2018].

[29] „Youtube - Use your laptop as a Screen, Keyboard for Raspberry Pi," [Online]. Available: https://www.youtube.com/watch?v=peZUZo_cSMk. [Zugriff am 18 02 2018].

[30] „adafruit - setting up a raspberry pi as a wifi access point," [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/setting-up-a-raspberry-pi-as-a-wifi-access-point.pdf. [Zugriff am 18 02 2018].

[31] „raspberrypi - Setting up an apache web server on a raspberry pi," [Online]. Available: https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md. [Zugriff am 18 02 2018].

[32] „Stackoverflow - How do I add a library project to Android Studio?," [Online]. Available: https://stackoverflow.com/a/16639227/7614256. [Zugriff am 18 02 2018].

[33] Google, „Google developer," 06 02 2018. [Online]. Available: https://developer.android.com.

[35] S. Neumann, „Universität Koblenz Landau," [Online]. Available: https://kola.opus.hbz-nrw.de/files/699/bach222.pdf. [Zugriff am 06 02 2018].

[36] „techStage," [Online]. Available: https://www.techstage.de/thema/Android. [Zugriff am 06 02 2018].

[37] „Computerwoche," [Online]. Available: https://www.computerwoche.de/k/android,3458. [Zugriff am 06 02 2018].

[38] „Android developers - View," [Online]. Available: https://developer.android.com/reference/android/view/View.html. [Zugriff am 07 02 2018].

[39] „LINKSYS - what is an Access Point and How is it Different from a Range Extender ?," [Online]. Available: https://www.linksys.com/us/r/resource-center/what-is-a-wifi-access-point/. [Zugriff am 09 02 2018].

[40] „lifewire - What is a wireless Access Point ?," [Online]. Available: https://www.lifewire.com/wireless-access-point-816545. [Zugriff am 09 02 2018].

[41] „Python sys -- System-specific parameters and functions," [Online]. Available: https://docs.python.org/2/library/sys.html. [Zugriff am 12 02 2018].

Anhang

Der Bachelorarbeit ist eine CD beigelegt. Folgende Elemente sind dort in Ordnern zu finden:

- Bachelorarbeit: Die Bachelorarbeit in PDF-Format
- Pi$Control: Der Quellecode der Android Applikation
- RasPi: Der Quellcode der Raspberry Pi Anwendung
- Anleitung: Die Anleitungen in PDF-Format

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass

- ich die vorliegende wissenschaftliche Arbeit selbständig und ohne unerlaubte Hilfe angefertigt habe,
- ich andere als die angegebenen Quellen und Hilfsmittel nicht benutzt habe,
- ich die den benutzen Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe,
- die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfbehörde vorgelegen hat.

Berlin, den _____ _____

Unterschrift