Diploma Thesis

AudioAnt

An assistant tool for hearing impaired persons

By

Benjamin Hackstock
Florian Langer
Daniel Sparber

Höhere Technische Bundeslehr- und Versuchsanstalt
6020 Innsbruck, Anichstraße 26-28
Höhere Lehranstalt für Wirtschaftsingenieurwesen,
Ausbildungsschwerpunkt Betriebsinformatik

2015 / 2016

Supervisors:

DI Benedikt Frischmann
Mag. Dr. DI Albert Greinöcker
Mag.^{a} Sylvia Mayr

Affirmation / Eidesstattliche Erklärung

Hereby I affirm that I wrote the present thesis without any inadmissible help by a third party and without using any other means than indicated. Thoughts that were taken directly or indirectly from other sources are indicated as such. This thesis has not been presented to any other examination board in this or a similar form, neither in Austria nor in any other country.

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Die vorliegende Arbeit wurde bisher in gleicher oder ähnlicher Form noch nicht als Diplomarbeit eingereicht.

Innsbruck, 08.04.2016

Benjamin Hackstock	Florian Langer	Daniel Sparber

Declaration of agreement / Einverständniserklärung

I herewith agree that a copy of this thesis, titled

AudioAnt
An assistant tool for hearing impaired persons

will be displayed in the library of the Höheren Technischen Bundeslehr- und Versuchsanstalt Innsbruck Anichstraße.

Ich bin damit einverstanden, dass ein Exemplar dieser Arbeit mit dem Titel

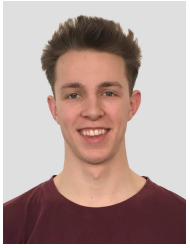
AudioAnt
An assistant tool for hearing impaired persons

in der Bibliothek der Höheren Technischen Bundeslehr- und Versuchsanstalt Innsbruck Anichstraße ausgestellt wird.

Innsbruck, 08.04.2016

Benjamin Hackstock	Florian Langer	Daniel Sparber

Project team



Benjamin Hackstock
Reschweg 12
A-6068 Mils

Tel.: +43 650 9403243
Email: benni.hack@hotmail.com

Florian Langer
Schützenstraße 58
A-6020 Innsbruck

Tel.: +43 664 3563300
Email: florian.langer2@gmail.com

Daniel Sparber
Spitz 18/1
A-6406 Oberhofen im Inntal

Tel.: +43 664 2432067
Email: sparber.daniel@gmail.com

Supervisors

Mag. Dr. DI Albert Greinöcker
HTL Anichstraße

Email: greinoecker@htlinn.ac.at

DI Benedikt Frischmann
HTL Anichstraße

Email: frischmann@htlinn.ac.at

Mag.^{a} Sylvia Mayr
HTL Anichstraße

Email: mayrs@htlinn.ac.at

Acknowledgement

At this point, we want to thank all the people without whose help we would have never been able to finish this project as we did in the amount of time we had.

Our biggest thank goes out to Mr. DI Benedikt Frischmann and Mr. Dr. Mag. Albert Greinoecker and Ms. Mag. Silvia Mayr, our supervising teachers, who enabled us to realise this project in the first place.

Apart from being responsible for the initial project idea, Mr. Frischmann helped us by advising us in the areas of hardware, project-management, and technical research, as well as motivating and encouraging us in the harder phases of the project. Mr. Greinoecker was our contact partner for software based questions and was a big help to us because of his rich amount of knowledge and his always positive and optimistic mindset. Ms. Mayr helped us face the challenge of a diploma thesis written in English and guided us during the process of writing a diploma thesis in a foreign language.

We also want to thank Mr. Werner Pfeiffer from the self-help association for deaf and tinnitus affected "PRO.ject.EAR". He helped us get in contact with people affected by hardness of hearing and enabled us to do the survey in the extent we did, questioning over 100 people that were affected by either tinnitus, hardness of hearing or hearing loss.

Abstract

According to a study, conducted by the "Deutscher Schwerhörigenbund (DSB)", the number of hearing impaired people (hearing performance reduced by more than 20dB) is at about 20 percent with a rising tendency. One common problem of these people is missing sounds in their own home. This could be an alarm clock, the doorbell or the alert sound of the microwave. With our project we want to conquer these problems by developing a device that can help the user reduce the amount of missed sounds in their own home.

The auxilary device, the AudioAnt, was implemented in the form of a prototype. First, we planned the construction, made production drawings and created a circuit diagram. During the course of the project, we constructed the housing of the AudioAnt, ordered and built in all the necessary components.

To analyse if a sound appears, an audio analysis software was implemented. During the analysis process a series of steps, based on each other, are executed. The three main steps are gaining the data by recording the ambient noise, extracting important features from the data, and comparing the obtained information.

In order to get all the important information out of the audio signal, the so called *Feature Extraction* is used. This is an umbrella term for many mathematical functions used to obtain characteristics based on waveform and spectrum. The most useful functions turned out to be the extraction of dominant frequencies and the *Mel Frequency Cepstral Coefficients (MFCC)*, as well the calculation of the *RMS-Energy* and the *Spectral-Rolloff-Points(SRP)*.

In addition to these core tasks of our diploma project, we conducted a usability research with focus on elderly users, an environmental analysis, a patent search and a competitor analysis. The findings we were able to get through these tasks were used to optimise the goals and priorities of our project. The prototype and the Android application, for example, were heavily influenced by the results of of our studies. Another research finding is that not only older people suffer from hardness of hearing and that our target group therefore also includes younger people.

The final result of our project is a fully functioning prototype and a working Android application used for configuring the AudioAnt. Using the inbuilt microphone, the ambient noise is constantly being recorded and analysed. Multiple tests have proven the ability of the AudioAnt to recognise almost every learned sound, provided its audio pattern is similar to a standard ringtone, meaning that it is the same every time it occurs, is artificially made and has characteristic frequencies.

Zusammenfassung (Abstract)

Laut einer Statistik des Deutschen Schwerhörigenbundes (DSB) aus dem Jahre 2011 liegt die Zahl an schwerhörigen Menschen (Hörleistung mehr als 20dB vermindert) mit steigender Tendenz bei über 20%. Ein gängiges Problem dieser Menschen ist das Überhören von Geräuschen in den eigenen vier Wänden. Dies kann zum Beispiel ein Wecker, der Signalton einer Mikrowelle oder die Haustürklingel sein. Bei diesem Problem setzten wir mit unserem Projekt an und entwickelten ein Hilfsgerät zur Minimierung der Anzahl der überhörten Geräusche im eigenen Haushalt.

Das Hilfsgerät, die AudioAnt, wurde in Form eines Prototypen realisiert. Für die Umsetzung des Gerätes wurde zuerst eine komplette Planung vorangestellt, welche Fertigungszeichnungen und einen Stromlaufplan beinhaltet. Im Laufe des Projektes wurde das Gehäuse gefertigt, die benötigten Komponenten bestellt und verbaut.

Um zu analysieren, ob ein gewisser Ton auftritt, wurde eine Audioanalyse implementiert. Hierbei wurden nach einem strikten Vorgang aufeinander aufbauende Schritte durchgeführt. Die drei essentiellen Aufgaben hierbei sind die Gewinnung der Daten durch Aufnehmen der Umgebungsgeräusche, das Extrahieren der wesentlichen Informationen und der abschließende Vergleich dieser.

Um die wesentlichen Informationen eines Audiosignals zu gewinnen, wird die sogenannte *Feature Extraktion* verwendet. Hierbei handelt es sich um den Überbegriff für eine Vielzahl mathematische Funktionen zur Gewinnung von Merkmalen basierend auf Wellenform und Spektrum. Wie sich durch zahlreiche Versuche herausstellte, sind das Extrahieren der dominanten Frequenzen und *Mel-Koeffizienten (MFCC)*, sowie die Berechnung der *RMS-Energie* und des *Spectral-Rolloff-Points (SRP)*, die hilfreichsten Funktionen.

Zusätzlich zu den Kernaufgaben wurden die Grundlagen der Benutzerfreundlichkeit ausgearbeitet und eine Umfeldanalyse, sowie eine Patentrecherche und Konkurrenzanalyse durchgeführt. Die Erkenntnisse dieser Aufgaben flossen direkt in die Arbeiten ein. So wurde speziell bei der Konzipierung des Hilfsgerätes und der Applikation auf eine leichte Bedienung Wert gelegt. Besonders die Umfeldanalyse zeigte, dass entgegen unserer Annahme auch viel jüngere Personen zur Zielgruppe gehören.

Das konkrete Ergebnis dieser Arbeit ist ein voll funktionsfähiger Prototyp inklusive Android-Applikation zur Bedienung und Konfiguration des Gerätes. Über das Mikrofon des Hilfsgerätes werden permanent die Umgebungsgeräusche aufgezeichnet und analysiert. Wie diverse Testszenarien ergaben, war die AudioAnt in der Lage, nahezu jedes gelernte Geräusch wiederzuerkennen, sofern es eine klingeltonähnlichen Charakter hat, sprich, wenn es sich um einen immer gleich auftretenden, künstlich erzeugten Ton mit charakteristischen Frequenzen handelt.

Contents

I. Preface **1**

1. Introduction **3**
1.1. Initial Situation . . . 3
1.2. Project Idea . . . 3
1.3. Description . . . 4
1.3.1. Hardware . . . 4
1.3.2. Audio Analysis . . . 4
1.3.3. Android Application . . . 4
1.3.4. Background work . . . 5
1.4. Goals . . . 5

II. State of the Art **7**

2. Survey **9**
2.1. Basics . . . 9
2.1.1. Population . . . 9
2.1.2. Sample . . . 9
2.1.3. Performance . . . 10
2.2. Rules for the wording of questions . . . 12
2.3. Types of questions . . . 13
2.3.1. Closed-ended questions . . . 13
2.3.2. Open-ended questions . . . 13
2.3.3. Hybrid questions . . . 13
2.4. Implementation . . . 14
2.4.1. Calculation of the sample size . . . 14
2.4.2. Demographic data . . . 15
2.4.3. Current situation . . . 16
2.4.4. Technical affinity . . . 17
2.4.5. Auxiliarity device . . . 21
2.4.6. Summary . . . 22

3. Competitor analysis **23**
3.1. Comparison of the competitors and products . . . 23
3.1.1. Distribution channel . . . 23
3.1.2. Quality and price-performance ratio of the products . . . 24
3.1.3. Customer base . . . 24
3.1.4. Weaknesses and strengths of the competitor . . . 24

3.2. Performed competitor analysis . . . 25
3.2.1. Doorbell amplification . . . 25
3.2.2. Ring tone intensifier . . . 25
3.2.3. Ring tone intensifier with doorbell adapter . . . 26
3.2.4. Amplicomms . . . 26
3.2.5. Summary . . . 27

4. Patent search **29**
4.1. Intellectual property . . . 29
4.1.1. Patent . . . 29
4.1.2. Utility model . . . 30
4.2. Patent law . . . 31
4.2.1. Requirements for assigning a patent . . . 31
4.2.2. Patent specification . . . 33
4.2.3. Patent offices . . . 35
4.2.4. Patent search . . . 35
4.3. Performed patent search . . . 38
4.3.1. Our search . . . 38
4.3.2. ÖPA search . . . 43
4.3.3. Summary . . . 47

5. Usability **49**
5.1. Usability in general . . . 49
5.1.1. What is usability? . . . 49
5.1.2. Why is usability important? . . . 50
5.1.3. Methods to improve usability . . . 51
5.2. Usability for Mobile Apps . . . 54
5.2.1. General usability and design principles . . . 54
5.3. Usability for the elderly . . . 56
5.3.1. Why it matters to us . . . 56
5.3.2. Existing studies and conducted research . . . 56
5.3.3. Difference between older and younger users . . . 56
5.3.4. Usability rules for the elderly . . . 57

III. Theoretical foundations **59**

6. Audio **61**
6.1. Introduction . . . 61
6.1.1. What is sound? . . . 61
6.1.2. Why does something sound like it does and not differently? . . . 61
6.2. Audio signals . . . 64
6.2.1. Basics . . . 64
6.2.2. Pulse-code modulation (PCM) . . . 64
6.2.3. Presentation of audio signals . . . 65
6.2.4. Audio files . . . 67
6.3. Feature Extraction . . . 68
6.3.1. What is feature extraction? . . . 68

6.3.2. Fourier transform . . . 68
6.3.3. Window function . . . 72
6.3.4. Time based features . . . 76
6.3.5. Spectral features . . . 77
6.4. Classification . . . 79
6.4.1. Threshold comparison . . . 79
6.4.2. Multivariate statistics . . . 80
6.4.3. Machine Learning . . . 83

7. Hardware **85**
7.1. Raspberry Pi . . . 85
7.1.1. Hardware . . . 85
7.1.2. Software . . . 87
7.2. Relay . . . 89
7.3. Qi . . . 90

8. Android **91**
8.1. What is Android? . . . 91
8.2. Interface Elements and Usage . . . 92
8.2.1. Homescreen . . . 92
8.2.2. Taskbar . . . 92
8.2.3. Navigation buttons . . . 92
8.2.4. Menu . . . 92
8.2.5. Applications . . . 92
8.2.6. Widgets . . . 93
8.2.7. Play Store . . . 93
8.3. Android Studio . . . 94
8.3.1. Features . . . 94
8.4. Coding Android Applications . . . 95
8.4.1. Differences in comparison to regular Java coding . . . 95

9. Calculation **101**
9.1. Cost-type accounting . . . 101
9.1.1. Classification of cost categories . . . 101
9.2. Cost centre accounting . . . 102
9.2.1. Direct costs . . . 102
9.2.2. Overhead expenses . . . 102
9.3. Cost-unit accouting . . . 103
9.3.1. Single-stage process costing . . . 103
9.3.2. Multistage process costing . . . 103
9.3.3. Overhead calculation . . . 103

10. Used tools and technologies **105**
10.1. Java . . . 105
10.1.1. JAudio . . . 105
10.1.2. REngine and RServeEngine . . . 105
10.1.3. JMF MP3 Plugin and JLayer . . . 106
10.1.4. MySQL connector . . . 106

10.2. Python . . . 106
10.2.1. RPi.GPIO . . . 106
10.3. R . . . 106
10.3.1. TuneR . . . 107
10.3.2. Seewave . . . 107
10.3.3. Rserve . . . 107
10.4. Android . . . 107
10.5. IDEs . . . 107
10.5.1. Eclipse . . . 108
10.5.2. RStudio . . . 108
10.5.3. Android Studio . . . 108
10.6. Sonic Visualiser . . . 108
10.7. Audacity . . . 109
10.8. FFmpeg . . . 109
10.9. MySQL . . . 109
10.10. Git . . . 109
10.11. Latex . . . 110

IV. Implementation **111**

11. Audio analysis **113**
11.1. Approach . . . 113
11.1.1. Analysis process . . . 113
11.1.2. Environment . . . 115
11.2. Feature Extraction . . . 118
11.2.1. Feature selection . . . 118
11.2.2. Feature implementation . . . 123
11.3. Calculation of the match rate . . . 125
11.3.1. RMS Energy . . . 126
11.3.2. Strongest Frequencies . . . 128
11.3.3. Spectral Rolloff Point . . . 130
11.3.4. MFCCs . . . 131
11.4. Classification . . . 134
11.4.1. Minimum thresholds reached . . . 134
11.4.2. All thresholds reached . . . 135
11.4.3. Weighted sum . . . 135
11.4.4. Match occurred . . . 137
11.5. Testing . . . 138
11.5.1. Automated testing . . . 138
11.5.2. Short term test . . . 140
11.5.3. Long term test . . . 141

12. Prototype **143**
12.1. Planning . . . 143
12.1.1. Power supply . . . 143
12.1.2. Benefit analysis . . . 143
12.2. Components . . . 144

12.2.1. Raspberry Pi 2 . . . 144
12.2.2. Casing . . . 144
12.2.3. Qi-transmitter . . . 145
12.2.4. Display . . . 145
12.2.5. Relay . . . 146
12.2.6. Soundcard . . . 146
12.2.7. Speaker . . . 147
12.2.8. Microphone . . . 147
12.2.9. WLAN-stick . . . 148
12.2.10. Light source . . . 148
12.2.11. USB-AUX connection . . . 149
12.2.12. Jack socket . . . 149
12.2.13. Power supply . . . 150
12.2.14. Sundries . . . 150
12.3. Design drawing . . . 151
12.3.1. Case . . . 151
12.3.2. Circuit diagram . . . 151
12.4. Construction . . . 154
12.4.1. Circuit board . . . 154
12.4.2. Case . . . 154
12.4.3. Light source . . . 155
12.5. Finished prototype . . . 157
12.6. Functionalities . . . 158
12.6.1. General Functionality . . . 158
12.6.2. Recording . . . 158
12.6.3. Connectivity . . . 159
12.6.4. Audio ports . . . 159
12.6.5. Summary . . . 159
12.7. Controlling the hardware components . . . 160
12.7.1. General . . . 160
12.7.2. Configuration files . . . 161
12.7.3. Output manager . . . 161
12.7.4. Input manager . . . 163

13. Android Application **165**
13.1. General application description . . . 165
13.1.1. Application requirements . . . 165
13.1.2. Connectivity . . . 165
13.1.3. Connection service . . . 165
13.1.4. Data transfer . . . 166
13.1.5. Activities . . . 168
13.1.6. Abandoned functionality . . . 171
13.2. Implementation . . . 173
13.2.1. Application Manifest . . . 173
13.2.2. Services . . . 174
13.2.3. Wi-Fi connection . . . 183
13.2.4. Local settings . . . 184
13.2.5. AudioAnt alerts . . . 184

13.2.6. Dialogs . . . 187
13.2.7. MainActivity.java . . . 189
13.2.8. RecordActivity . . . 190
13.2.9. ListActivity . . . 193
13.2.10. AudioAntSettings activity . . . 194
13.2.11. SettingsActivity . . . 198
13.2.12. AlarmActivity . . . 198

14. Calculation **201**

Appendix **203**

A. Time Protocol **205**
A.1. Benjamin Hackstock . . . 205
A.2. Florian Langer . . . 209
A.3. Daniel Sparber . . . 212

B. Survey **219**

C. Patent search **225**
C.1. ÖPA patent search proposal . . . 225
C.2. ÖPA patent search results . . . 225

D. List of Figures **235**

E. List of Tables **239**

F. Listings **241**

Part I.

Preface

1. Introduction

1.1. Initial Situation

According to a study, conducted by the "Deutscher Schwerhörigenbund (DSB)", the number of hearing impaired people (hearing performance reduced by more than 20dB) is at about 20 percent with a rising tendency.[1] Roughly half of these 20 percent are badly enough influenced by loss of hearing, so that the use of an assistance device would be of great help to the user.

One of the most common problems for people who are hard of hearing is missing noises and sounds in their own home. These sounds include for example the doorbell, an alarm clock or the alert sound of the microwave. We want to face this problem by building a personal assistance device that can recognise these sounds and inform the user, therefore reducing the amount of missed sounds of a person at their home.

1.2. Project Idea

The initial idea for our project came from our teacher Mr. DI Benedikt Frischmann. Personal experiences had led him to the idea of developing a hearing assistance device that could conquer the challenges in these areas.

The idea was to create a stationary device that constantly monitors its surrounding using a built-in microphone. The characteristics of this audio stream are then constantly compared to previously learned signals. If a sound has been recognised, the software triggers multiple methods of alerting the user.

The device, the "AudioAnt", can be configured using a mobile application for Android smartphones. We chose this way of configuring it, because it provides good user experience and can be handled by technically inexperienced users without having to use complicated command line entries. This is especially important for us, since hardness of hearing becomes more and more common as people age and the majority of our target group will therefore consist of people that might not be as used to handling modern technology as younger generations are.

Once a signal is recognised, the user should be alerted in the following ways:

[1] Statistische Angabenn zur Hörschädigung in Deutschland zwischen 2005 und 2011, http://www.oesb-dachverband.at/uploads/media/DSB_SH_Statistik2011.pdf, 23.1.2016

- **Visual Signals**
  The device can emit flashes of light using an LED-light and thereby alert the user in a silent way.
- **Acoustic Signals**
  A loud acoustic alert can be triggered by the AudioAnt, whereby the alarm tone for each learned signal can be individually set.
- **Haptic Signals**
  If a local Wifi network exists and both, the AudioAnt and the smartphone are connected to it, the permanent connection can be used by the AudioAnt to trigger vibrations on the smartphone. However, it is also possible to play acoustic signals and use the built-in LED-flash of the phone for alerting the user.

1.3. Description

1.3.1. Hardware

The computing unit of the AudioAnt is a Raspberry Pi 2, that is used together with multiple external gadgets in order to enable sound to be recorded and replayed. The establishment of wireless connections is executed by using Bluetooth and Wifi. In addition, the smartphone can be charged wirelessly, if it is placed on the provided space on top of the AudioAnt.

We chose the Raspberry Pi 2 because of the computing power it offers, combined with the support of all common coding languages such as Java, Python and R which we had to use in order to achieve our goals in a realistic and effective way.

1.3.2. Audio Analysis

The audio analysis we use to identify if an important sound has occurred, happens on the Raspberry Pi. The coding language R is used to extract audio features out of the audio stream coming in from the microphone. Additionally, Python is used to control the external hardware. The whole process is managed by an underlining Java program.

1.3.3. Android Application

The mobile application for configuring the Raspberry via Bluetooth or Wifi was written for Google's Android platform. Apart from configuring the Raspberry, it also enables the smartphone to alert the user in case of an occuring sound and therefore increases the usability of the AudioAnt System.

1.3.4. Background work

In order to evaluate the needs of our potential users, we conducted a survey that helped us to decide which features to prioritise during the development process, how to implement certain features and which ones are unimportant to the user and can therefore be abandoned.

To ensure our project would not collide with any existing products, we conducted a patent search, both by ourselves as well as by experts of the Austrian patent office ("Österreichisches Patentamt" - ÖPA).

We also conducted a research on mobile usability with a special focus on elderly people. We did this, although not all of our users will be elderly people. Nonetheless, the elderly need to be taken into account the most, since they are very likely to be least familiar with modern technology.

In order to evaluate if our product has a chance on the global market, we developed a calculation scheme. With this scheme we evaluated the production costs of the AudioAnt.

1.4. Goals

The goal of our diploma project is the development of a fully functioning prototype that is able to recognise previously learned sounds and react to them. The Android application should be fully implemented and allow complete configuration wirelessly.

Part II.

State of the Art

2. Survey

Conducting a survey during the project has given us an insight into how people with hearing difficulties deal with everyday noises.

2.1. Basics

Before compiling a questionnaire a few general things have to be discussed:

- What is the purpose of the survey?
- Who should be questioned?
- How should the survey be performed?

2.1.1. Population[1]

In statistics, a population is a group of people which is of interest to be questioned concerning a certain topic. This can be for example a department in a company or eligible voters for a certain election. Due to lack of time or money it is often not possible to question every single person. Usually, a sample is chosen to be the representative for the population.

2.1.2. Sample

A sample is a selection of people out of a statistical population. For the sample to be representative, the chances of every person to be selected must be equal and the individual person must not be allowed to answer the questions several times.

2.1.2.1. Calculation of a sample

The size of a sample can be calculated with a mathematical formula. For huge statistical groups (over 100.000 people), this simplified formula can be used:

[1] http://moodle2.htlinn.ac.at/pluginfile.php/8594/mod_resource/content/1/befragung.pdf, 2016-02-14

Figure 2.1.: Sample of a population

Source: http://www.cliffsnotes.com/study-guides/statistics/sampling/populations-samples-parameters-and-statistics, 2016-02-28 14:33

$$n = \frac{t^2 * p * q}{e^2} \tag{2.1}$$

n...size of the sample
t...permissible error range (e.g. t = 2 = 95.5% security level)
p...number of elements which fulfil certain characteristics
q...number of elements which do not fulfil certain characteristics
e...accuracy (e.g. 5%)

2.1.3. Performance

There are four ways a survey can be performed. Every person can be questioned face-to-face, phoned, or questioned in an online or written survey.

2.1.3.1. Face-to-face interview

There are a few characteristics that have to be fulfilled in order to prove that the survey was performed with direct contact to the interviewee. The following issues always refer to an interviewee and the interviewer:

- personal contact
- same location
- same time

When using this technique, the person being questioned has no visual contact to the questionnaire.

2.1.3.2. Telephone interview

A telephone interview, as the name indicates, is performed over the telephone. It is also called a voice-to-voice-interview. When performing a telephone interview, the interviewer has no personal contact to the interviewee. Neither has the person being interviewed visual contact to the questionnaire.

2.1.3.3. Written survey

When performing a written survey, the survey is sent to the interviewee by mail. The risk of people not understanding the question(s) or interpreting the question(s) wrongly has to be taken. When using this technique the questionnaire can be seen by the interviewee and there is no direct contact to the interviewee at any time. Typical characteristics of a written survey are:

- local distance
- temporal distance
- Interview situation cannot be controlled.

Disadvantages when performing the survey by post:

- The interviewee cannot ask questions.
- You have to enclose a envelope with prepaid postage.
- The response rate is very low.

2.1.3.4. Online survey

There is a range of services on the Internet which offer online survey performance. After preparing the survey, a link is created which can be sent to the interviewees.

Disadvantages of an online survey:

- The interviewee cannot ask questions.
- The survey can be answered several times.
- Different cooperation problems can occur.

Online surveys are getting more and more popular because they are cheap, less time consuming than other performance methods and the collected data are easy to handle.

2.2. Rules for the wording of questions

Due to a wide variety of language levels, questions for surveys have to conform to certain criteria. The questions have to be understood by every person without any further explanation. A suitable question has to fulfil the following criteria:

- Simple words
  It is not intended to use profanities, specialist terms, foreign words or abbreviations.

- Short verbalizations and clarity
  Due to the fact that many people are overstrained by long formulations, the questions should be distinct and as short as possible. Furthermore, every question should be phrased in a neutral way and have a consistent language. In addition, it has to be ensured that the question is understood by every person in the same way (equivalence).

- No leading questions
  A leading question is a question which leads the interviewee to give a certain answer. For example: Haven't you always wanted to visit Austria and go skiing in the beautiful Alps? Leading questions are not meant to be part of a survey because they are unserious and give unusable data.

- Reference
  If a question asks for more than one issue at once, the interviewee may get confused and answer the question wrongly. In addition, it cannot be said, which issue the respondent referred to, which makes the question more difficult to evaluate. For example: How often do you play golf or tennis?

- Excessive demand
  To make sure a question is suitable, it is not intended to overstrain the interviewee, so the question should be easy to answer without having to think about it excessively (e.g. no double negations).

2.3. Types of questions

?. A question can either evaluate facts or opinions. Fact-based questions evaluate information and facts. Typical fact-based questions survey for example the revenue of the individual, demographic data, actions in the past or possession of goods. Opinion questions evaluate opinions on different issues. Typical opinion questions survey opinions on recent issues, political systems or on a current state of affairs.

2.3.1. Closed-ended questions

Closed-ended questions evaluate facts and are easy and quick to answer. Multiple-choice, check boxes and rankings are always closed-ended questions and are the easiest way to gather facts and information about the interviewee. When creating closed-ended questions such as multiple choice, it is important to list every answer possible and that the answers provided are mutually exclusive. When respondents do not find their intended answer they are likely to skip the question or answer it wrongly.

2.3.2. Open-ended questions

Open-ended questions are used, when a close-ended question is not reasonable due to too many possible answers. Furthermore, it enables users to give a more personal and exact answer than close-ended questions would do. Elaborating the answers of open-ended questions can sometimes be very time-consuming because the answers are more likely to be richer in quality.

2.3.3. Hybrid questions

A hybrid question is a mixture between an open-ended question and a closed-ended question. For example, a multiple choice question can be extended with an item which provides a text box where the interviewees can write their own answers. As well as with open-ended questions, analyzing and interpreting the answers can be time-consuming due to misunderstandings or ambiguity.

2.4. Implementation

Conducting a survey as a part of our diploma thesis was, on the one hand, a very helpful way to find out what our potential customers really want to have, but on the other hand, merely a secondary part since our project was mainly about producing and designing a functioning prototype. The aim of the survey was to collect ideas and we were never aiming at being representative or giving exact statements about the users' preferences. Although the survey has not been perfect from a statistician's point of view, it has given us some impulses on how to design the prototype, which functionalities it should have and if potential customers are actually interested in such a device.
The survey was performed online due to lack of time to perform it orally or written. There are a few websites which provide the implementation of a survey. For our survey we chose the service offered by Google (https://www.google.com/intl/de_at/forms/about/) because

- it is for free,
- the website is easy to handle,
- you get an evaluation and a summary with graphs automatically and
- it offers a wide variety of types of questions.

2.4.1. Calculation of the sample size

As discussed theoretically, the simplified formula for calculating the size of the sample to be representative for the population can be used. Therefore, the following values are used:

- t = 2 = 95,5% security level
- p = q = 50% (based on the assumption that the amount of questioned males and females is equal)
- e = 5% accuracy

$$n = \frac{2^2 * 50 * 50}{5^2} = 400 \tag{2.2}$$

According to the theory, the sample should comprise 400 people to be representative for the population.

The questioned sample comprises about one hundred people who go from slightly to deeply hearing impaired. Because of the size of the sample being too small and not everybody affected by hardness of hearing having the same chance of being questioned, the sample is not representative for the population. In addition, the sample included some people who are not hearing impaired. At this point, we would like to thank Mr. Werner Pfeifer for distributing the link from the online survey to many hearing impaired people via email. Unfortunately, we cannot state the response rate because the number of sent back emails is unknown.

The survey was divided into four subclasses which all relate to a certain issue.

2.4.2. Demographic data

Evaluating demographic data is implemented into almost every survey to find out more about the interviewees.

1. Gender (figure 2.2)
55% of the interviewees were female, the remaining 45% were male.

2. Age (figure 2.3)
Age groups:

Group	Percentage
Less than 25 years	11%
25 to 45 years	27%
46 to 65 years	43%
Over 66 years	19%

3. Persons in the household (figure 2.4)
Most people live in a two-person household (41%), more than a third live in a three-person household and about one in four people live alone.

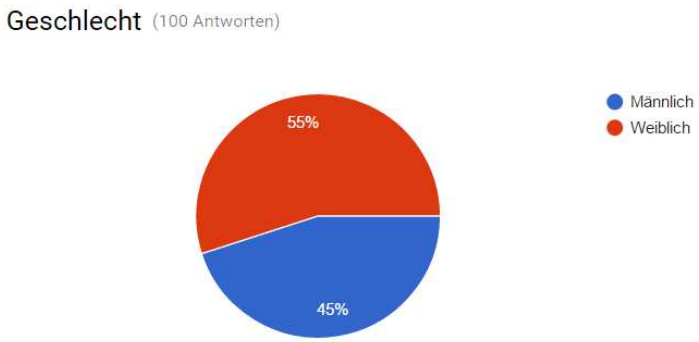
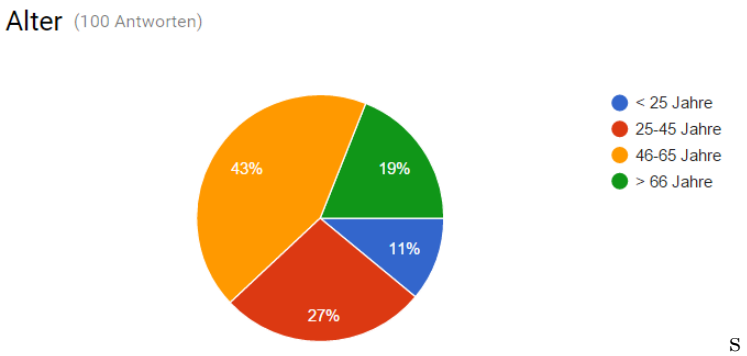


Figure 2.2.: Survey, demographic data, question 1



s

Figure 2.3.: Survey, demographic data, question 2

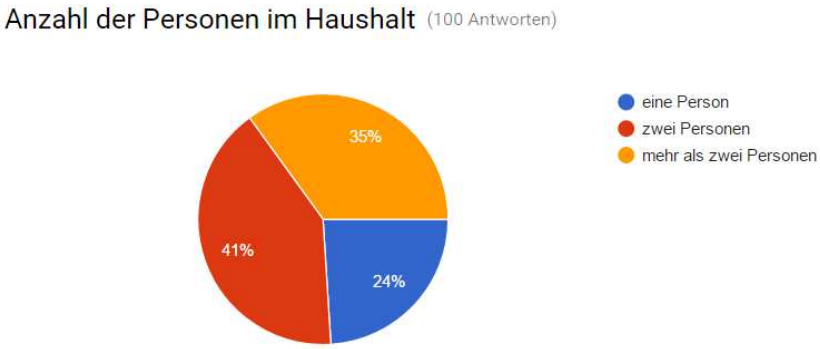


Figure 2.4.: Survey, demographic data, question 3

2.4.3. Current situation

This subclass relates to the current situation of the interviewees and should have given us an insight into the degree of the interviewees' hearing impairment, which daily noises they are likely not to hear and how often they wear their hearing aid at home.

1. Which daily noises are you likely not to hear? (Multiple answers possible.) (figure 2.5)
Over 50 percent of the people often do not hear their phone ring. Nearly half of the people cannot hear their doorbell or alarm clock. Almost one fifth stated that they are likely not to hear other daily noises. This particular question really helped us with the decision whether the user can decide which noise they want to be recognized or if there should be a given range of noises they can choose from to be recognized.

2. How good is your hearing performance without a hearing aid? (figure 2.6)
3% of the respondents have no difficulty with hearing, 7% are completely deaf. The rest of the people are going from slightly to deeply hearing impaired. It was important for us to see how loud the outgoing signal has to be for the majority of the costumers to be able to hear the recognized sound.

3. Do you wear your hearing aid at home? (figure 2.7)
More than 70% of the people are always wearing their hearing aid at home. The others are only wearing it occasionally. Only a very few (4%) are wearing their hearing aid rarely or never at home.

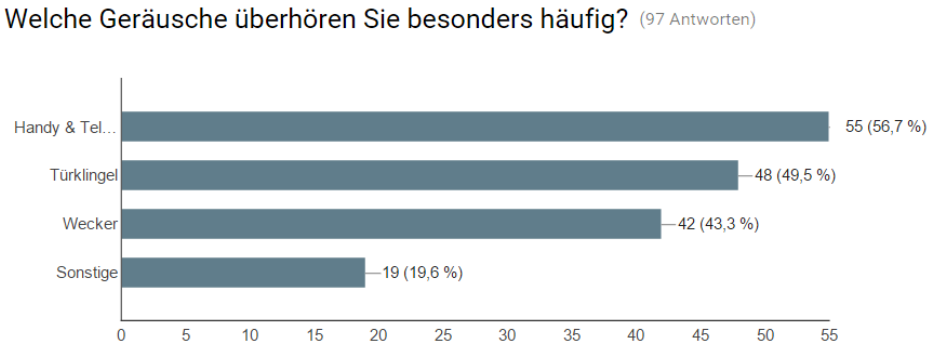


Figure 2.5.: Survey, current situation, question 1

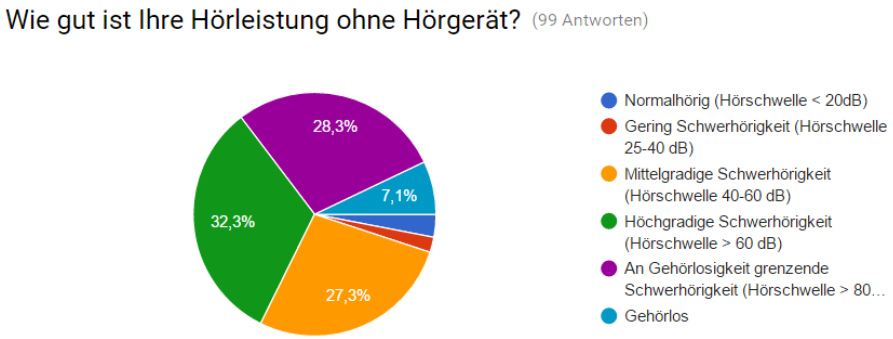


Figure 2.6.: Survey, current situation, question 2

Figure 2.7.: Survey, current situation, question 3

2.4.4. Technical affinity

The subclass "Technical affinity" was important for us to find out how good people are at dealing with modern electronic devices.

1. Do you own a smart phone? (figure 2.8)
More than 75% of the people own a smart phone.

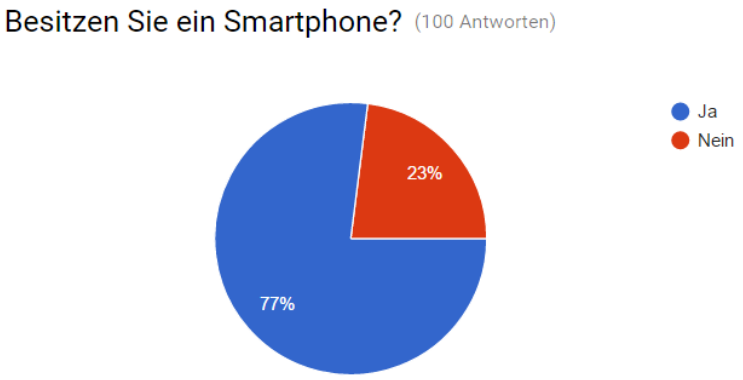


Figure 2.8.: Survey, technical affinity, question 1

2. Do you own a PC, a tablet or a laptop? (figure 2.9)
Only four out of 100 people do not own a tablet, a laptop or a PC. These two questions helped us to figure out how useful a smart phone application and a web interface would be to configure the AudioAnt.

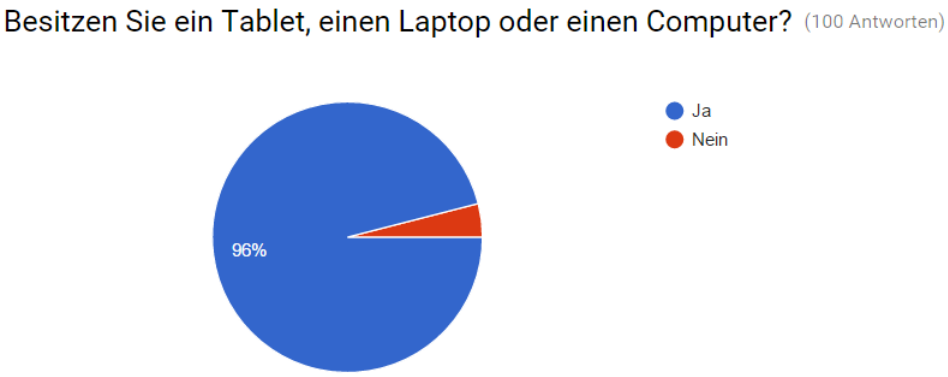


Figure 2.9.: Survey, technical affinity, question 2

3. Do you carry your phone attached to your body? (figure 2.10)
A majority of the questioned people are likely to always carry their phone attached to their body, whereas approximately 11% carry their phone rarely or never attached to their body. Due to this question we determined that it would be reasonable to send a message to the user's phone to alert the user when a sound is recognized.

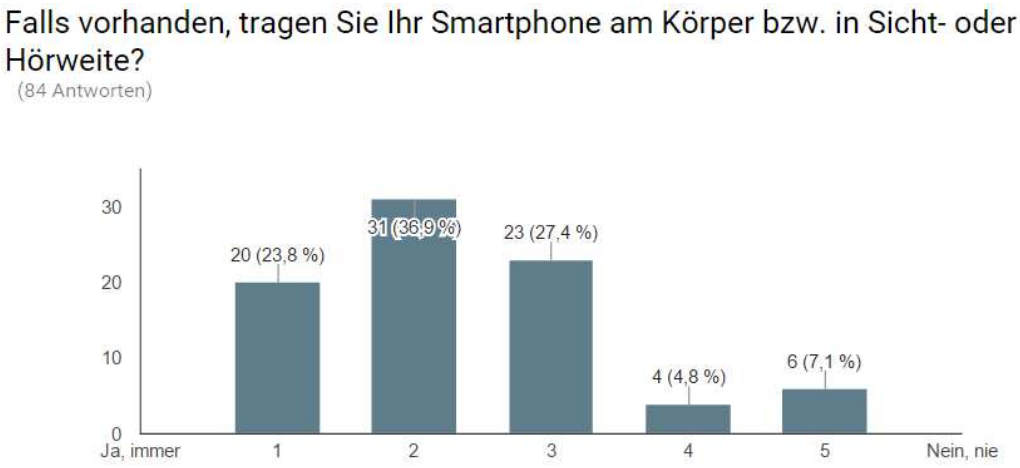


Figure 2.10.: Survey, technical affinity, question 3

2.4.4.1. Familiar use cases

The following four questions were intended to find out how technically affine the respondents are.

4. Which applications do you use on your smartphone/computer/tablet? (Multiple answers possible.) (figure 2.11)
Primarily, people use the Internet, use email services and Office applications.

5. What actions do you perform when a technical problem occurs? (figure 2.12)
The rating went from 1 (I try to solve the problem myself) to 5 (I get help). More than half of the people tend to always try to fix the problem themselves. 13% of the interviewees are always seeking assistance.

6. What are you likely to do when you get a new mobile? (figure 2.13)
More than 50% of the people would try their new phone themselves. More than one quarter tend to ask somebody for help or even for the configuration of their new phone. Less than one quarter would read the operating instructions.

There were 4 options to choose from:

Segment	Option
Blue	Try it out yourself
Red	Read the operating instructions
Orange	Ask for explanation
Green	Ask somebody for the configuration of your new mobile

7. How willing are you to try out new electronic devices? (figure 2.14)
Generally, people have a tendency to try out new things and electronic devices.

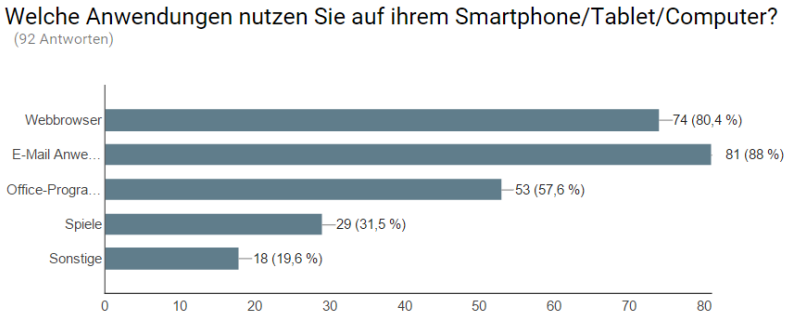


Figure 2.11.: Survey, technical affinity, question 4

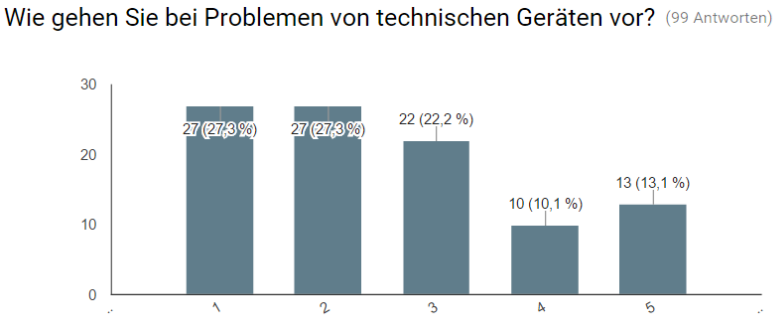


Figure 2.12.: Survey, technical affinity, question 5

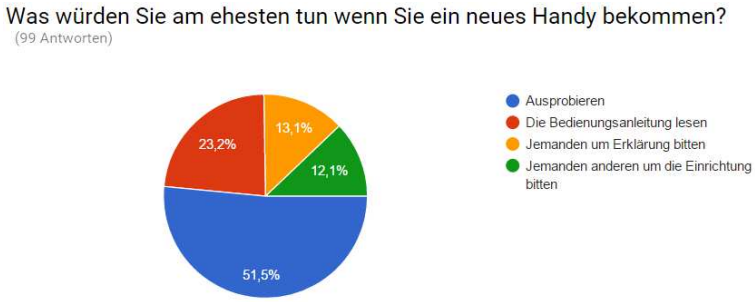


Figure 2.13.: Survey, technical affinity, question 6

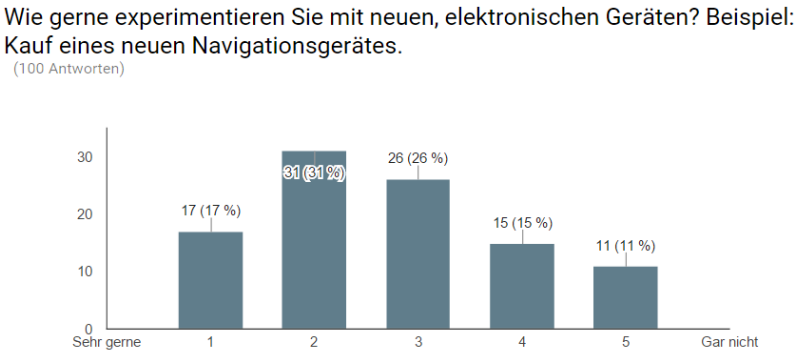


Figure 2.14.: Survey, technical affinity, question 7

2.4.5. Auxiliarity device

This subsection is about figuring out which features the product should have and which features are valued the most by users.

1. Which of these features do you value the most? (Multiple answers possible.) (figure 2.15)
Due to the results of this question, we attached great importance to the first three issues. Compact design and visual appeal are considered to be secondary.

Enumeration of the answers (in descending order):

Option	Percentage
Ease of use	78.8%
Reliable recognition	51.5%
Many adjustment options	35.4%
Compact design	22.2%
Visually appealing	19.2%

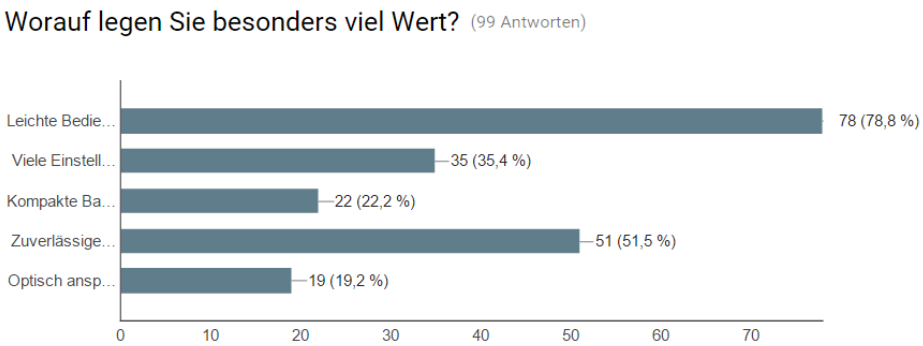


Figure 2.15.: Survey, auxiliarity device, question 1

2. Would you prefer the sound analysis to be overly sensitive, meaning that it would sometimes send a wrongly recognized signal, or would you prefer it to be less sensitive but sometimes miss important sounds? (figure 2.16)
Unfortunately, the question was expressed in such a bad way that none of the respondents really knew which answer to give. This can be seen by the fact that one third just picked the golden mean.

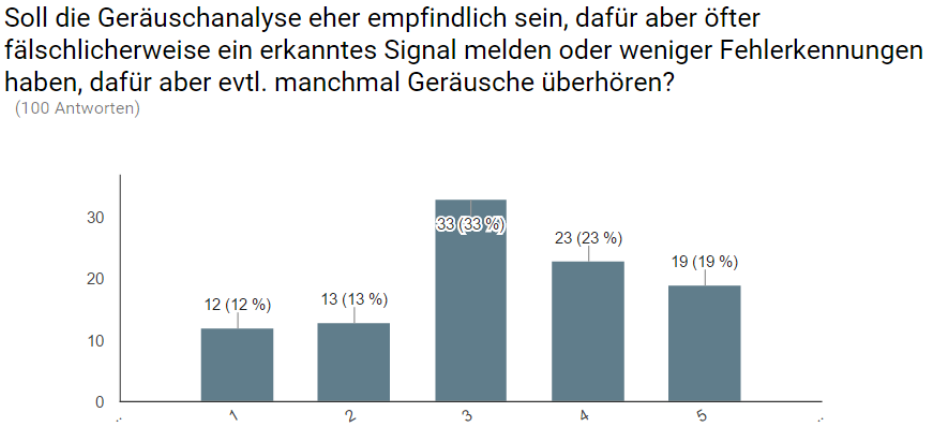


Figure 2.16.: Survey, auxiliarity device, question 2

3. Would you consider recognizing cries for help as useful? (figure 2.17)
87% think it would be useful to detect a cry for help. Due to lack of time we have not been able to implement a help button yet. If the project proceeds, the help button would definitely be worth thinking about.

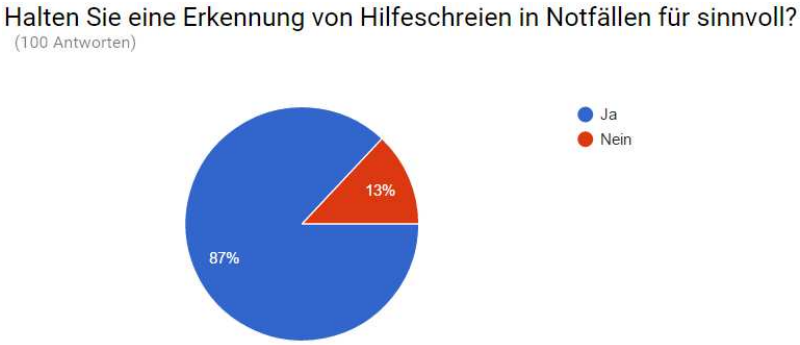


Figure 2.17.: Survey, auxiliarity device, question 3

2.4.6. Summary

Obviously, there is a huge demand for an aid device like ours due to many hearing impaired people not hearing their phones, doorbells or other devices ringing. Furthermore, the survey revealed the volume necessary for the outgoing signal to be heard by the majority of users. It can be assumed that the major part of the users is technically affine enough to manage and configure the AudioAnt without any further help.

3. Competitor analysis

The competitor analysis is a major part of a project, when started by a business. Not only shows the result which products from competitors are similar to the products you want to produce and are on a competitive basis, but also which companies are in competition with you.[1]

Finding direct competition is not always easy. After an extensive market research, most of the competitors should be clear. An important aspect is which businesses or products the company is able to compete with. This depends on the size of the company and the quality of a product. For example, handmade noodles produced by a local store cannot compete with pasta produced by an international company.

Competitive rivalry will be high if

- competitors sell the same or very similar products,
- the size of the businesses are approximately identical or
- fellow campaigners have similar sales strategies.

3.1. Comparison of the competitors and products[2]

Concerning products, the first conspicuousness is the price difference between the same products from various sellers. Moreover, features of the product or services can be compared to the product or service you want to produce or rather offer. The analysis may also show possible improvements of your product due to failure or faulty products of other companies.

To get a good overview of the competitors, the following issues have to be elaborated.

3.1.1. Distribution channel

There are a few strategies a business can use when selling products. Not only can they sell their products in stores, but also in online shops, on exhibition stands or door-to-door. To become a worthy competitor, it would be good to compete in more than one of these markets. The more distribution channels a business has, the better. Nonetheless, more money has to be invested to run a bigger distribution network.

[1] Competitor analysis https://www.fuer-gruender.de/businessplan-vorlage/konkurrenzanalyse/, 2016-02-14

[2] Learnmarketing: Competitor analysis - http://www.learnmarketing.net/competitoranalysis.htm, 2016-02-14

3.1.2. Quality and price-performance ratio of the products

The quality and the price of a product often lets the buyer decide if a product is bought or not. Unfortunately, most cheap products do not show very high quality. When you want to produce a high-quality product, you can either produce a high quantity to get a lower price when purchasing raw materials or raise the price of the final good.

3.1.3. Customer base

A firm only becomes a competitor when they are aiming at the same customer base as your business. If the competitor has great success with this customer base, you can either change your customer base (e.g. sell other products) or directly compete with the competitor.

3.1.4. Weaknesses and strengths of the competitor

To evaluate strengths and weaknesses of the competitor, the business has to be examined from different angles. Therefore, a strength-weakness profile can be created to visualize the competitor's advantages and disadvantages. Ideally, it is noticeable which disadvantages your product has so they can perhaps be compensated before producing. Within this profile, many different characteristics have to be rated. Once you know their strengths and weaknesses, you can compensate them by for example, buying raw materials with higher quality.

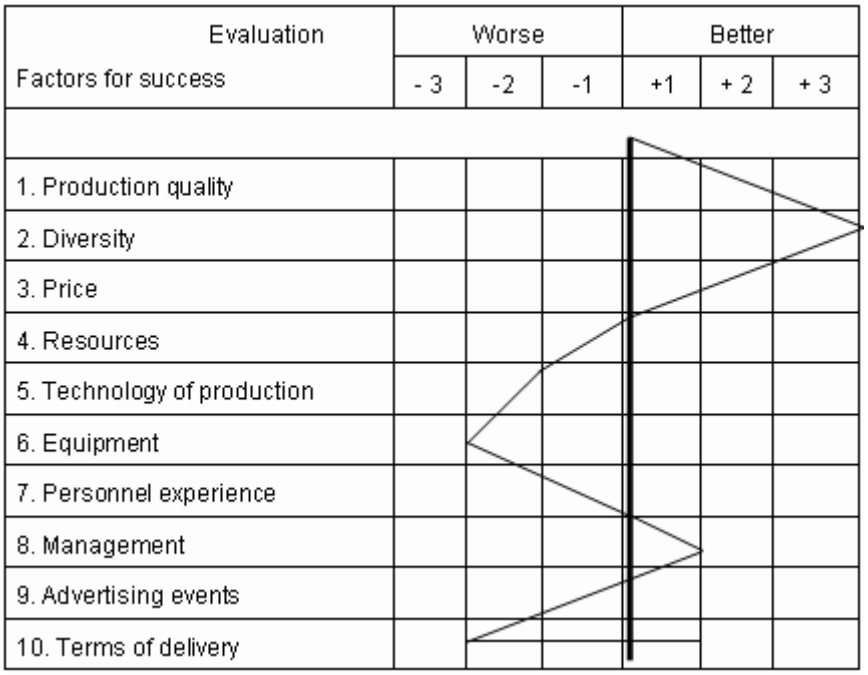


Figure 3.1.: Example for a strength-weakness profile

Source: http://www.innosupport.net/index.php?id=2189, 2016-02-23 16:16

3.2. Performed competitor analysis

Due to the fact that our diploma project is very technology-dependent, the competitor analysis only played a minor role in this project. For this reason, the extent of the analysis is rather small and not all theoretical processes were implemented. There are a few products available which provide diverse amplification of different signals or sounds. Only four of them are similar enough to our product to describe them more exactly.

3.2.1. Doorbell amplification

This tool provides emergency paging and doorbell intensification, which helps people to ring for help or hear their doorbell loud enough, also in their cellar or backyard. The intensifier is portable and you can choose between eight different doorbell sounds. The volume is infinitely variable and the receiver additionally disposes of a light source to optically alert the user when the doorbell is ringing. The receiver is powered by a 3V lithium battery and the sender has to be plugged into an socket. According to the producer, the tool can receive signals up to 100 metres in the open. The tool costs 59.90 € and can be bought online at www.senioren-onlineshop.de. The button to ring for help can only be bought on request. The scope of delivery includes a transmitter, a receiver and an instruction manual.

Figure 3.2.: Emergency paging and doorbell intensification tool

Source: http://www.senioren-onlineshop.de/Funk-Personenruf-bis-ca-100-m-mobil-und-erweiterbar, 2016-02-23 17:01

3.2.2. Ring tone intensifier

This tool provides ring tone intensification with flash light to hearing impaired or elderly people. It is only usable for landline phones and provides three ringtones to choose from. Its volume is infinitely variable and can be up to 95dB. The tool has to be integrated into the phone line to notice an incoming phonecall. The scope of delivery includes the intensifier, a telephone line, a power supply and an instruction manual. The tool can be bought on different websites and costs between 29.95 € (www.senioren-onlineshop.de) and 36.33 € (www.amazon.de).

Figure 3.3.: Ring tone intensifier

Source: http://www.senioren-onlineshop.de/Anrufverstaerker-Ring-Flash-100-Telefon-Zusatzklingel-Blitz, 2016-02-23 17:17

3.2.3. Ring tone intensifier with doorbell adapter

The ring tone intensifier is the same tool as above but contains an additional small wireless receiver. The tool receives signals from a wireless adapter unit which observes the doorbell. The wireless doorbell adapter unit has to be attached next to the doorbell. The tool provides three ringtones and two doorbell sounds to choose from. Furthermore, the tool has an interface to notice an additional occurring sound. Therefore, a second wireless adapter unit has to be installed next to the tool which causes the sound that needs to be intensified. As mentioned above, the tool has a flash light to optically alert the user when it receives a signal. The tool is available on different websites and costs between 69.95 € (www.senioren-onlineshop.de) and 72.00 € (www.amazon.de), the second wireless adapter unit costing extra. The scope of delivery contains the intensifier, the wireless doorbell adapter unit, a telephone line, a power supply and an instruction manual.

3.2.4. Amplicomms

Amplicomms is a special brand from Audioline GmbH. According to their website (www.amplicomms.com), the business produces a wide variety of specialized products for hearing impaired and elderly people. *"Under the amplicom brand name we offer a range of extremely specialised phones for users with impaired hearing and sight and for elderly people. The functions of these phones are totally oriented to the needs of persons with audiovisual handicaps, but can naturally also be used at home and in the office by users with and without hearing impairments.*

Figure 3.4.: Ring tone intensifier with a wireless doorbell adapter unit

Source: http://www.senioren-onlineshop.de/Anrufverstaerker-Ring-Flash-250-Tuerklingelverstaerker-Blitz, 2016-02-23 17:43

All of the models in this range optimally combine functionality, technology and design, and offer features such as compatibility with hearing aids, extremely high reception volume and extra-loud ringing tones." http://www.amplicomms.com/index.php?en_amplicomms2016-02-25 13.52

It can be assumed that Amplicomms is aiming at the same customer base (except visually impaired people) as we do. Mainly, Audioline produces mobile phones for elderly people. As per their website, more than 10 million Audioline devices have been sold in Germany. It can be expected, that their market share in the range of hearing impaired people is rather high. They distribute their products online on a wide range of websites in Germany and Austria.

3.2.5. Summary

The competitor analysis revealed that tools for hearing impaired people already exist. However, none of the tools seemed to be as developed as ours. The tools are only applicable for a certain scope but none of them is as flexible concerning sound recognition as our product.

Amplicomms seems to be the greatest seller concerning assistance tools for hearing impaired people in Germany and Austria.

4. Patent search

During the patent search, we looked for existing patents in order to inform ourselves about existing tools and technologies that might be useful to us, furthermore, we performed it to avoid violating legal protections.

4.1. Intellectual property

In general, intellectual property refers to creations of the mind, for instance inventions, literary and artistic pieces of work and names used in commerce.

Intellectual property is a fundamental principle of capitalism, protected by law, in order to earn recognition and financial benefits from what a person has created. The goal of the system of intellectual property is to the keep balance between interests of innovators and the public interests.

There are many different forms of intellectual property rights, the most important which are listed below.

- Copyright: It gives the creator of the original work exclusive rights, but is limited to a certain time span.
- Patent and Utility model: a right given to an inventor, which excludes others from achieving profit with the invention
- Trademark: a recognisable sign, design or expression to distinguish products or services
- Industrial design right: protects the visual design of objects that are not purely utilitarian

4.1.1. Patent

When you have for example accomplished new insights in an area and you do not want anyone other than you to profit from these insights, you have two options: concealment or protection by patenting.

A patent is an exclusive right for an inventor or assignee granted by a sovereign state. Therefore patents are bound to a certain region, additionally they are limited to a specified period of time (usually between 10 and 20 years maximum, in Europe 20 years).[1]

[1] Patent - Wikipedia, https://en.wikipedia.org/wiki/Patent, 2015-10-14

The exclusive right prevents others from commercially making, selling, using, importing, or distributing a patented invention without permission of the patent's owner.

4.1.2. Utility model

The utility model is also an intellectual right to protect inventions. It is similar to the patent, but it has a shorter term (about 6 to 15 years, in Europe 10 years) and less stringent patentability requirements.[2]

Although most countries determine that inventions, protected by a utility model, have to be new, there are mostly no substantive examinations needed. Utility models are usually granted by just checking the application formalities.

The most famous example of the utility model is the German and Austrian "Gebrauchsmuster", which influenced some other countries such as Japan or Indonesia. Contrary to the European patent, processes and methods cannot be protected. Additionally, the maximum lifetime is only half as long (10 instead of 20 years). Some further interesting characteristics of a "Gebrauchsmuster" are:[3]

- Prior art considered for examining novelty and inventive step is more limited, since oral disclosures and prior art outside of Germany are not taken into account.
- There is a six month grace period provided for prior art. For example you are allowed to apply for a "Gebrauchsmuster", even if there was a written disclosure three months ago.
- "Gebrauchsmuster" are, like most utility models, not substantially examined before being granted.
- The Austrian "Gebrauchsmuster" also includes protection for logic algorithms for computer software, processes and therapy methods for animals.

[2] Utility model - Wikipedia, https://en.wikipedia.org/wiki/Utility_model, 2015-10-14

[3] Gebrauchsmuster - Wikipedia, https://en.wikipedia.org/wiki/Gebrauchsmuster, 2015-10-14

4.2. Patent law

4.2.1. Requirements for assigning a patent

When applying for a patent, you have to disclose your discovery. Therefore, the work is made public for everyone once you submit the proposal.

In order to patent an invention, it must fulfil nationally given guidelines which are nearly the same in the whole western world. The European patent office (EPO) defines a patentable invention according to the European patent convention (EPC 15th edition, September 2013), article 52:

1. *European patents shall be granted for any inventions, in all fields of technology, provided that they are new, involve an inventive step and are susceptible of industrial application.*
2. *The following in particular shall not be regarded as inventions within the meaning of paragraph 1:*
   a) *discoveries, scientific theories and mathematical methods*
   b) *aesthetic creations*
   c) *schemes, rules and methods for performing mental acts, playing games or doing business and computer programs*
   d) *presentations of information*
3. *Paragraph 2 shall exclude the patentability of the subject-matter or activities referred to therein only to the extent to which a European patent application or European patent relates to such subject-matter or activities as such.*

Therefore, novelty, usefulness and non-obviousness are essential for an invention being patented. Without these factors it is almost impossible to acquire exclusive rights in the meaning of patents.

4.2.1.1. Novelty

Novelty is probably the most important requirement for patentability. Hence, an invention can not be declared new if it has been known to the public before the date of submitting the patent application (EPC 2013, article 54).

The main concept behind this requirement is to prevent prior intellectual work from being patented. Thus, it is not possible to limit someone who has done it before by claiming it to be an invention of yours.

Usually, for assessing the novelty of an invention, a search through 'prior art' is performed that helps to assure that your invention is actually new. Therefore, it is obligatory to search

large patent databases for publications by matching keywords. Nonetheless, it is impossible to guarantee the novelty of an invention, since one might not be able to take account of every single existing publication.

4.2.1.2. Inventive step and non-obviousness

According to most patent laws, inventions have to be sufficiently inventive (non-obvious) in order to be patented (EPC 2013, article 56). Therefore, an invention which is obvious for an expert can no be patented.

The underlying purpose of the inventive step (used in Europe), or non-obviousness (used in the US) principle, is to avoid protections for inventions which only follow from normal product design and development.

4.2.1.3. Industrial applicability

Apart from the previous criteria, a patent can only be granted when it is susceptible of industrial application (EPC 2013, article 57).

Therefore it is only possible to claim a patent right, when the applicant actually uses the invention. This measure aims to save inventions without the goal of profitable use from being patented.

4.2.1.4. Exclusions from patentability

In order to enable a well functioning patent law it is needed to exclude certain subjects from patentability.

Examples for exclusions (EPC 2013, article 53):

- inventions contrary to law and order or morality
- discoveries ($\neq$ inventions)
- scientific theories
- mathematical methods
- aesthetic creations
- schemes, rules and methods for performing mental acts
- playing games or doing business
- computer software

- presentations of information
- plant or animal varieties
- methods for treatment of the human or animal body by surgery or therapy
- diagnostic methods practised on the human or animal body
- etc.

4.2.2. Patent specification

The patent specification is needed when applying for a patent. It describes the invention and sets out the scope of protection. Due to the complexity of a specification which protects an invention in the best way, the application is commonly written by a legal expert.

4.2.2.1. Contents of a patent specification

In order to describe the invention with all needed details, most patent offices require the following contents:[4]

- description of the background and overview of the invention
- representation of the current state of the art
- claims of the patent
- representation of the problem
- solution for the existing problem
- representation of advantages
- figures (optional)

4.2.2.2. Structure of a patent specification

By setting a uniform structure for patent specifications, working with them is simplified. The following structure is commonly used by many patent offices:[5]

- cover sheet with title, summary and bibliographic data
- name of the invention, with the problem representation, examples, etc.

[4] Patent Applications, http://inventors.about.com/od/patentsbasics/a/Specifications.htm, 2015-10-14

[5] Patent specification - Wikipedia, https://en.wikipedia.org/wiki/Patent_application, 2015-10-14

- patent claims (main claim and sub claims)
- figures
- search reports

4.2.2.3. IPC and CPC Codes

The IPC and CPC codes are patent classification systems. When a patent office publishes a patent application or grants one, it classifies the invention using IPC or CPC codes. Therefore these categorisation classes are an essential aid for people who want to search patent databases.

The International Patent Classification (IPC) divides all kinds of technologies into over 70000 categories. Theses classes are hierarchically structured according to the technical field of the invention.[6]

The newer system, the Cooperative Patent Classification (CPC), extends the IPC. This categorisation is jointly managed by the European and the US patent office and is constantly expanded whenever new technical fields emerge. The CPC is divided into nine sections, which contain hierarchical sub divisions, namely classes, sub-classes, groups and sub-groups. Estimated, there is a total of 250000 categories.[7]

4.2.2.4. Patent Kind Codes[8]

The Patent Kind Code is a system for patents and patent applications which simplifies expressing the status of a patent specification. A typical patent nomination consists of the following three parts: the region code for the applied area, a unique patent number and the Patent Kind Code.

Example: US 6453500 B1

The most important Patent Kind Code examples are:

- A: publication of the patent application
- B: granted patent (e.g. in Austria)
- C: granted patent (e.g. in Germany)
- T: translation
- U: granted utility model

[6] EPO - IPC (International Patent Classification), http://www.epo.org/searching/essentials/classification/ipc-reform.html, 2015-10-14

[7] EPO - Cooperative Patent Classification (CPC), http://www.epo.org/searching/essentials/classification/cpc.html, 2015-10-14

[8] Patent Kind Codes, http://www.cas.org/content/references/patkind, 2015-09-24

4.2.3. Patent offices

A patent office is an organization, governed by a state, which controls everything concerning patents. These offices may grant or reject the patent application based on whether or not the requirements for patentability are fulfilled.

4.2.3.1. National patent offices

National patent offices are able to grant patents for the specific country which they represent. For instance, if a person wants to apply for a patent being valid in Austria, Germany and Italy, three applications need to be submitted, one for every country.

Since it would become a lot of effort when applying for a patent in more than just a few countries, international patent offices were established.

4.2.3.2. International patent offices

An international patent office is able to release a patent in all its member organizations. The largest patent office worldwide is the World Intellectual Property Organization (WIPO) with 142 member countries. Applying for a patent for all these countries can therefore be done by submitting only one single application. However, the granting processes duration can last up to three years.[9]

Apart from the WIPO, there are other international patent offices, like the European Patent Office (EPO), which includes all members of the European Patent Convention (EPC). Due to the smaller amount of members, the granting process for European patents is much faster than for international ones.

4.2.4. Patent search

Performing a patent search can be exhausting, since extracting the relevant pieces of information is often strenuous. Nonetheless, performing a prior art search is one of the most important tasks during product development.

4.2.4.1. Advantages of performing a patent search

At some point inventors want to make use of their developed product by distribution. Since distribution is only allowed if there are no existing patents restricting the technologies used by the product, it becomes essential to know about similar products and according patents. If

[9] World Intellectual Property Organization - Wikipedia, https://en.wikipedia.org/wiki/World_Intellectual_Property_Organization, 2015-10-14

the product is released without caring about possible patent infringements, the distributor can easily get into serious trouble.

Apart form avoiding conflicts with existing patents, a prior art search also helps to get ideas for one's own product. When searching for patents one may stumble upon solutions to current problems, already solved by others, or find out about important features never even thought of. With this additional inspiration it becomes a lot easier to develop the best product possible.

4.2.4.2. Patent search procedure

A patent search requires accurate working. To ease the process of a patent search, the following steps should be respected:

1. Figuring out characteristics of the invention:
    - key words
    - synonyms
    - abstraction level
    - etc.
2. Starting the patent search:
    - analysing patent specifications
    - analysing citations
    - analysing categories
3. Refining the search:
    - using refined keywords
    - trying out new combinations
    - using different data sources

4.2.4.3. Patent search platforms

The easiest way to search for patents is using the internet. Apart from common search platforms, like Google, Wikipedia or internet forums, there are websites especially made for patent searches. These platforms are databases, accessible via the internet, provided by patent offices. These databases are free of charge and enable an easy way to search by oneself.

The most relevant patent databases for our case are the following two:

- **Espacenet:** Database of the European patent office (https://ep.espacenet.com/)
- **DEPATISnet:** Database of the German patent office (https://depatisnet.dpma.de)

4.2.4.4. Professional patent search

Most patent offices provide a commercial and professional patent search. The advantage of this search is not needing to perform a search by oneself, as the only thing needed is to submit a detailed description of one's invention. Due to experienced employees the patent search is more accurate compared to a search performed by oneself.

The Austrian patent office (Österreichisches Patentamt - ÖPA) offers a professional prior art and novelty search for every specific technical problem. Exclusively for diploma, bachelor or master theses, they offer a free patent search.

4.3. Performed patent search

During the patent search, we looked for existing patents. Firstly, we searched on our own to get the results. We mostly used the patent search engine provided by Google (https://patents.google.com), since Google's patent platform allows easy content based keyword searching, considering the most important patent offices (US, EP, DE,...). After we found a matching result, we looked it up on the European patent office database (Espacenet).

Afterwards, we requested the professional search offered by the ÖPA to ensure the correctness and completeness of our own search.

4.3.1. Our search

All in all, we found about five patents, but only two of relevance. The patents only cover a bell system for the elderly, where it is possible to notify them via optical alerts or GSM transmitters.

4.3.1.1. CN 202551174 U

Source:
http://worldwide.espacenet.com/publicationDetails/biblio?CC=CN&NR=202551174U&KC=U
2015-10-06 20.28

Overview
The utility model, granted in 2012, describes a ring recognition device, aiming to identify if a specific chime or alarm signal has been sent. The utility model protection only covers China, since there is no patent family.

The device
This device comprises the following components, as seen at figure 4.1:

(1) a microphone

(2) an amplifying filtering circuit

(3) an analogue digital conversion circuit

(4) a ring tone recognition module

(5) and a GSM transmission module

With these components combined within a single device, the inventor is able to perform a bell recognition literally out of the box. Our concept is very similar to this strategy, since all components are also placed within a single device.

Nevertheless, our tool differs as regards patent invention in some ways, for instance, not having a GSM transmitter or an amplifying circuit. With these missing components, the sound processing flow is different from the invention.

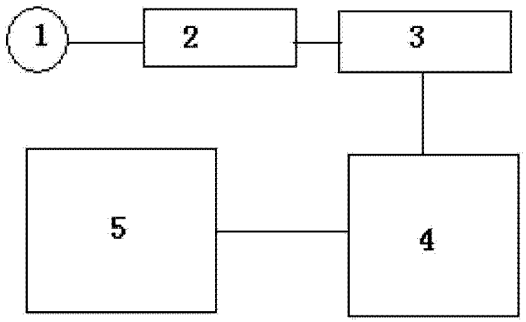


Figure 4.1.: Schematic view of the bell recognition device

Source: http://worldwide.espacenet.com/publicationDetails/originalDocument?CC=CN&NR=202551174U&KC=U (Page 5 - Drawings), 2015-10-06 19.22

Claims

The utility model describes three claims:

0001 One kind of a bell recognition apparatus, characterised by comprising a microphone (1), an amplifying filter circuit (2), an analogue digital conversion circuit (3), a bell recognition module (4) and a GSM transmitter module (5). The bell (1) is connected to the amplifying filter circuit (2), this circuit is connected to the conversion circuit (3). The conversion circuit is connected to an microcontroller (4) port and the GSM transmitter (5) to the microcontroller.

0002 A bell recognition apparatus according to claim 1, characterised by the recognition module being a single chip

0003 A bell recognition apparatus according to claim 1, characterised by the GSM chip transmitting a network alarm message when receiving a signal from the microcontroller.

Resulting form the declared claims, violating the patent is only possible by building a device with exactly the same parts in the same order (claim 1).

Since the second and third claim references to the first, constructing a bell recognition device, where the analysis is performed by just one chip or using a GSM module, is legitimate if you do not use the same-structured elements.

Summarisation

Although the invention has some similarities to our device, there are still differences. According to the formulated claims, the patent does not conflict with our work.

Even if the claims collided with our aspects of the invention, it would not matter, since there are no legal protections outside of China, since the inventor only applied for a patent registration protecting the device there.

4.3.1.2. CN 201556267 U

Source:
http://worldwide.espacenet.com/publicationDetails/biblio?CC=CN&NR=201556267U&KC=U
2015-10-06 19.46

Overview
This utility model, granted in 2010, describes a twinkling doorbell system for people with hearing difficulties. The protection ended on January 25th 2012.

Description of the invention
Often, the elderly with hearing impairments cannot hear the doorbell ringing. Therefore, they have to be reminded by an optical signal.

This invention describes a system, where the doorbell is connected to various light bulbs positioned at well-marked places inside a room. These lamps go on when the doorbell rings. The main advantage of this structure is its simplicity.

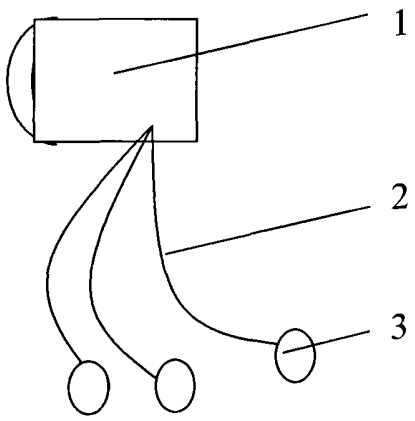


Figure 4.2.: Schematic view of the shiny belt doorbell: The system consists of the bell (1), connecting wires (2) and light bulbs (3).

Source: http://worldwide.espacenet.com/publicationDetails/originalDocument?CC=CN&NR=201556267U&KC=U (Page 4 - Drawings), 2015-10-06 19.22

Claims
The utility model claims the following characteristics:

0001 A shiny doorbell, which is characterised by comprising the doorbell, wires and light bulbs. The wires connect the bell with a series of light bulbs, located at obvious indoor places. When the doorbell rings, the light bulbs go on.

The claim will not conflict with our product, since we have the light source directly installed within the device. Furthermore, our device has no doorbell built in and hence only recognises the sound of an existing one.

The only thing which is identical to our invention is that a light source illuminates when the bell rings in order to notify people with hearing difficulties.

Summarisation

In short, this invention shares the same idea as ours but the execution has almost nothing in common. Concerning the claim, there is no conflict, and even if there was one, it would not influence our project, since the whole utility model already expired.

4.3.1.3. WO 2008067638 A1

Source:
http://worldwide.espacenet.com/publicationDetails/description?CC=WO&NR=2008067638A1&KC=A1&FT=D 2015-12-30 12:17

Overview

This world wide application describes a personal unit for hearing impaired users which notifies them of a sound event detected in its surrounding environment.

Since this is only an application, it would not forbid any commercial actions with our product. Nevertheless, it would not be possible to patent a similar invention, since the inventive step would not be fulfilled thereby.

Description

This patent describes a system, which consists of various detection units. These units are installed around the house. The detection units may comprise a door bell detector, a fire alarm detector, a phone ring detector, etc. Additionally, each detection unit has a wireless transmitter for transmitting a signal to the portable receiver worn by the user, close to the body.

When the portable receiver receives a signal from one of the detection units, it warns the user with a vibration. The receiver typically has light indicators with a symbol or a word underneath for indicating which detection unit generated the signal (e.g. four lights for "door", "fire", "phone" and "sound").

Claims

The application has multiple claims, the main claim is listed below:

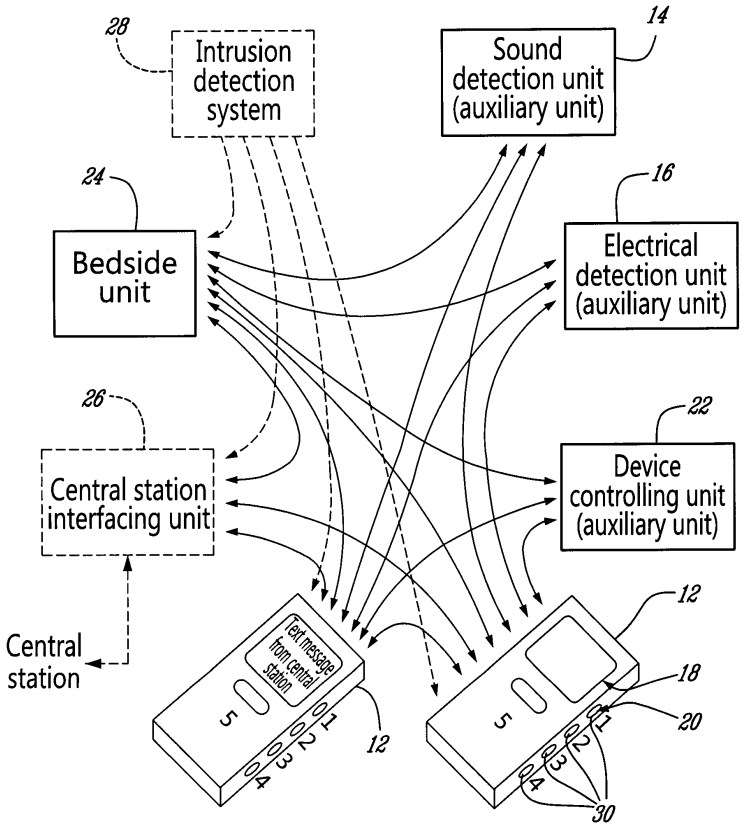


Figure 4.3.: Schematic view of the sensing system: The system consists of various detection units, a central unit (26) and a portable receiver (12)

Source: http://worldwide.espacenet.com/publicationDetails/originalDocument?CC=WO&NR=2008067638A1&KC=A1&FT=D (Page 35 - Drawings), 2015-12-30 12:30

0001 A personal unit for notifying a deaf/hearing-impaired user of a sound event in its surrounding environment and for communicating an immediate assistance request to a central station, said personal unit comprising:

- *a transceiver for receiving a sound event message representative of said sound event detected in said environment,*
- *an output module for notifying said user of said sound event message, and*
- *an input module for inputting said immediate assistance request, said request to be communicated to said central station using said transceiver;*
- *said transceiver being further for receiving an inquiry message initiated by said central station in response to said request,*
- *said output module being further for displaying said inquiry message, and*
- *said input module being further for inputting an answer to said inquiry message, said answer to be communicated to said central station using said transceiver.*

Summarisation

Even though this application describes a helping device for hearing impaired people, the approach is different to ours. The main distinction is that this systems consists of multiple detection units. Therefore, this application (if granted) would not restrict our work.

4.3.2. ÖPA search

In addition to our search, the Austrian Patent Office (ÖPA) found multiple patents related to our product. Most of these patents cover recognition and alarming systems.

In order to get results form the ÖPA, we had to describe the main aspects of our invention. The search request and the results are contained within the appendix.

4.3.2.1. US 8269625 B2

Source: http://worldwide.espacenet.com/publicationDetails/biblio?CC=US&NR=8269625B2&KC=B2&FT=D 2015-12-30 17:31

Overview

This patent, granted in the Untied States (2012) describes a signal processing and detecting system for audible alarms generated by smoke detectors, carbon monoxide detectors, and other types of detectors.

Description

Due to the lack of reliable warning systems, the inventor created a system which can reliably detect audible alert signals generated by commercially available smoke, fire, and carbon monoxide detectors, including audible alert signals of different intensities. The disclosed system and process can also reliably detect audible alert signals in the presence of background noise of varying intensity.

Claims

This patent also has multiple claims, the main claim is listed as following:

0001 A method for detecting audible alert signals implemented in signal processing circuitry, the method comprising: generating a digital representation of a received audio signal; for each of a plurality of templates, comparing the digital representation of the received audio signal to the respective template, and generating a matching score representing a degree of match between the digital representation of the received audio signal and the respective template, wherein each template corresponds to an acceptable audible alert signal timing pattern, and the plurality of templates encompass an acceptable range of variation in a timing pattern of a standard audible alert signal; and determining, based at least partly on the matching scores corresponding to the plurality of templates, whether an audible alert signal is present.

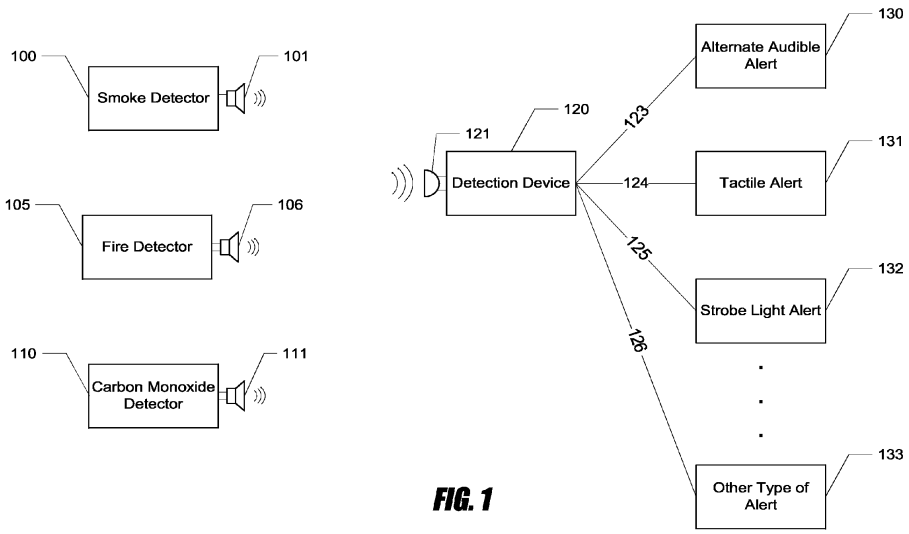


Figure 4.4.: Schematic view of signal processing system: Multiple detectors (100, 105 and 110) emit a warning signal; the detection device (120) detects them and alerts the user (130 - 133)

Source: http://worldwide.espacenet.com/publicationDetails/originalDocument?CC=US&NR=8269625B2&KC=B2&FT=D (Page 3 - Drawings), 2015-12-30 17:46

Summarisation

This patent specifically aims to detect standardised (US) fire related alarms. The detection approach is similar to ours. Therefore we would not be able to patent our product, since the inventive step can not be fulfilled.

4.3.2.2. WO 2011000113 A1

Source:
http://worldwide.espacenet.com/publicationDetails/description?CC=WO&NR=2011000113A1&KC=A1&FT=D 2015-12-31 13:37

Overview

This world wide patent application, submitted in 2011, describes a sound and voice detector for hearing impaired or deaf persons.

Description

This warning device for hearing impaired persons provides the possibility of detecting sound events in the immediate environment. The detection units used can include phone, door bell, fire alarm or other type of sound detector.

The device continuously compares the digital information to ambient sound information which is converted into ambient digital information and alerts the individual with an alerting signal when the pre-recorded digital information matches the ambient digital information corresponding to the warning signal.

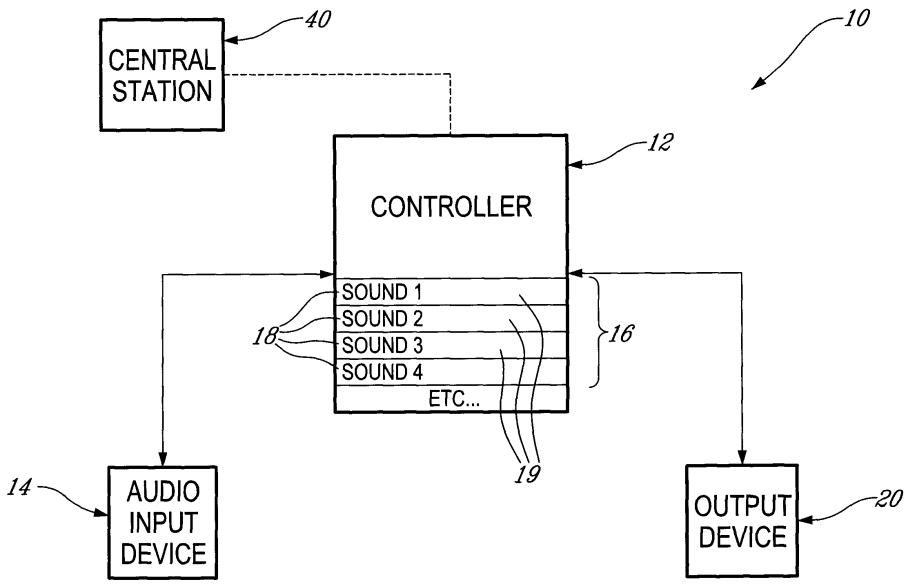


Figure 4.5.: Schematic view of the sound and voice detector: The central station (40) consists of an audio input device (14), an output device (20) and a controller (12) which compares the input signal to the previously saved sounds

Source: http://worldwide.espacenet.com/publicationDetails/originalDocument?CC=WO&NR=2011000113A1&KC=A1&FT=D (Page 39 - Drawings), 2015-12-30 13:49

Claims

This patent claims almost one hundred aspects. The first (and main) claim is listed as following:

0001 A system for notifying a deaf/hearing impaired user of a sound event and/or a word spoken by the human voice produced in the ambient environment, said system comprising:

- *a controller providing for recording and storing a plurality of target sound events and target words;*
- *at least one audio input device for continuously capturing in realtime sound events and spoken words in the ambient environment and transmitting in real-time the captured sound events and captured spoken words to said controller; and*
- *at least one output device for receiving a signal from said controller that a given sound event or a given spoken word has been captured and for producing a notification signal thereby [...]*

Summarisation

The detector, capable of recognising a plurality of sounds including words produced by the human voice and notifying a hearing impaired person, is also very similar to ours. Hence we would not be able to patent our product, since the inventive step can not be fulfilled.

4.3.2.3. Further results

Additionally, the ÖPA search result listed further patents related to our description. Nonetheless, these publications do not exactly match.

WO 0074013 A1
An analysis station which analyses sounds recorded by the microphone. If the sound, for example, is a fire alarm, the station turns on a signal light.

Source: http://worldwide.espacenet.com/publicationDetails/biblio?CC=WO&NR=0074013A1&KC=A1&FT=D

DE 19816752 C2
A spectral analysis unit distinguishes between different audible signals. When the unit detects for instance a doorbell a vibration and light emitter alerts the user.

Source: https://register.dpma.de/DPMAregister/pat/PatSchrifteneinsicht?docId=DE19816752C2

KR 20110092479 A
A series of microphones that enables filtering sounds form background noise. When the system detects something, lamps will be turned on.

Source:
http://worldwide.espacenet.com/publicationDetails/description?CC=KR&NR=20110092479A&KC=A&FT=D&ND=3

US 4237449 A
Sounds like car horns, police sirens or fire alarms are detected by an analysis unit which lights up LEDs to inform the user.

Source: http://worldwide.espacenet.com/publicationDetails/biblio?NR=4237449A&KC=A&FT=D&CC=US

JP 2004073417 A
A microphone records sounds in order to detect signals produced by kitchen appliances. A speaker amplifies the sound for hearing impaired persons.

Source: http://worldwide.espacenet.com/publicationDetails/biblio?CC=JP&NR=2004073417A&KC=A&FT=D

WO 2014168594 A1
A warning system comprising a wristband embodiment alerts a user when a warning signal occurred.

Source: http://worldwide.espacenet.com/publicationDetails/biblio?CC=WO&NR=2014168594A1&KC=A1&FT=D

US 2009154744 A1
Audible signals are recorded by multiple microphones. Depending on the recognised signal an according symbol will be displayed on a monitor.

Source: http://worldwide.espacenet.com/publicationDetails/biblio?CC=US&NR=2009154744A1&KC=A1&FT=D

JP 3043730 B1
A comparison unit is able to differentiate between known and unknown (e.g. surrounding noise) audible signals.

Source: http://worldwide.espacenet.com/publicationDetails/biblio?CC=JP&NR=3043730B1&KC=B1&FT=D

DE 4033673 A1
A small device (which fits into a pocket) alerts hearing impaired persons when an audible signal (baby alarm, bell, alarm clock, etc.) occurred.

Source: http://worldwide.espacenet.com/publicationDetails/biblio?CC=DE&NR=4033673A1&KC=A1&FT=D

4.3.3. Summary

The prior art search revealed that there are many existing publications in this technical area. Hence, our invention cannot be declared as entirely new. Due to this reason we are not able to patent this project, since the novelty can not be fulfilled.

Nonetheless, we are able to commercially use (selling, using, importing or distributing) our product, since we would not violate any existing patents or utility models. This could be relevant, if we continue this project after graduating.

5. Usability

5.1. Usability in general

5.1.1. What is usability?

The point of usability is to describe how easy and intuitive it is to use something, how uncomplicated the learning process and how satisfying the user experience is. Usability does not only deal with software or technical devices, but with all human-made objects that involve human interaction, for example, a computer program, a milling machine, a door handle, or the process of how to change a bike's tire. [1]

Figure 5.1.: Examples for bad usability

Source: http://www.gruenderszene.de/allgemein/ veranstaltungstipp-sie-mussen-nur-den-nippel-durch-die-lasche-ziehn-usability-in-der-praxis, 2016-01-02 20:00

According to the International Organisation for Standardisation (ISO), usability can be described as *"The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use."*

Jakob Nielsen, a usability consultant and author divides usability as a part of usefulness into five elements:[2]

1. *Learnability: How easy is it for users to accomplish basic tasks the first time they encounter the design?*

2. *Efficiency: Once users have learned the design, how quickly can they perform tasks?*

[1] Wikipedia - Usability, https://en.wikipedia.org/wiki/Usability, 2015-10-26

[2] Jakob Nielsen - Usability, http://www.nngroup.com/articles/usability-101-introduction-to-usability/, 2015-10-26

3. *Memorability: When users return to the design after a period of not using it, how easily can they re-establish proficiency?*
4. *Errors: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?*
5. *Satisfaction: How pleasant is it to use the design?*

5.1.2. Why is usability important?

Good usability is crucial for every product to survive on the market, whether it is a website, a computer program or a physical entity like a bike or a watering can. If something is not easy to use, people are likely to not use it at all.

If looked at from a business perspective, improving the internal usability of used tools, software or processes helps saving money on training employees and improves the efficiency of your business.

A study by Jakob Nielsen in 2003 about the return on investment for usability showed very clearly how design projects are positively influenced by involving usability as a main part of the project.("*Return on investment (ROI) is the benefit to the investor resulting from an investment of some resource*"[3])

For the study, he collected data from 863 projects and analysed the benefits gained by improved usability. Based on his results, he found out that the best practice for usability in a project is to dedicate 10 percent of the total financial budget to usability.
By doing so, desired metrics for websites like visitor count, user productivity, or conversion rates increased by over 130 percent. ("*The conversion rate is the percentage of users who take a desired action*"[4])

While it may be more obvious to improve the usability of products sold by the company, one must not oversee the benefits gained by improving the internal usability. By doing so, it is possible to simultaneously reduce costs, avoid errors and have a higher working efficiency.[5]

In 2008, a second study concerning the ROI for usability conducted by Nielsen indicated that the improvement of key performance indicators decreased to just over 80 percent compared to over 130 percent in 2003. The reason for this are the decreasing benefits of usability paired with the almost constant costs. While a few years back it was possible to really stand out of the mass when having good usability on, for example, an online-shop, nowadays, as usability is much more present and a lot more companies invest in it, the positive effects of usability are a

[3] Wikipedia - Return on investment: https://en.wikipedia.org/wiki/Return_on_investment, 2015-10-27

[4] Jakob Nielsen - Conversion Rates: http://www.nngroup.com/articles/conversion-rates/, 2015-10-26

[5] Jakob Nielsen - ROI for Usability: http://www.nngroup.com/articles/return-on-investment-for-usability/, 2015-10-26

lot smaller. However, this also means that not having good usability will have a much bigger negative impact than it would have years ago.[6]

5.1.3. Methods to improve usability

5.1.3.1. User Testing

This is an easy and frequently used method that consists of three components:

- Find representative users.
  In order to simulate a realistic use-case, the user testing has to be performed by users who fall in the target group of the project. An experienced IT-specialist cannot test, for example, a computer program and give representative insight on usability needs of other, less knowledgeable people.

- Perform representative tasks.
  Not only do the performed tasks have to be realistic, but they also have to take place in a realistic environment to be able to give useful information. Users in lab tests will, for instance, run into issues with lower contrast text or pictures much less frequent than actual real-life users.

- Observe the user.
  The last step is to observe the user behaviour, protocol what they do, where errors appear and in which areas they are successful. It is crucial not to interfere with the user in any way (for example by giving them hints, drawing their attention to certain elements) or otherwise the test will not represent a realistic use-case and the results will be meaningless.

To maximise the efficiency of this method, small tests between adaptations are a much better way to go than fewer big tests, since flaws can be identified and fixed faster, using less resources.[7]

5.1.3.2. Eye Tracking

The method of "eye tracking" uses sensors to monitor the motion of the users' eyes while they are using, for example, a web page.[8]

It can help answer a few basic questions concerning usability:

- Where are the users most likely looking?

- How long are users looking at specific elements?

[6] Jakob Nielsen - ROI decreasing:
http://www.nngroup.com/articles/usability-roi-declining-but-still-strong/, 2015-10-27

[7] Jakob Nielsen - Usability: http://www.nngroup.com/articles/usability-101-introduction-to-usability/, 2015-10-26

[8] Blickverlaufsmessung:
http://www.onlinemarketing-praxis.de/glossar/eye-tracking-blickverlaufsmessung , 2016-02-02

- How well can users orient themselves?
- Are there any elements users frequently miss?
- What are the "eye-catchers" on the website and which elements should be more visible to the user?
- How do users navigate through the website? Is the existing eye-movement equal to the desired eye-movement of the user? [9]

With the collected data, heat maps and saccade pathways can be created to visualise what users were most commonly looking at and how their eyes moved across the screen.

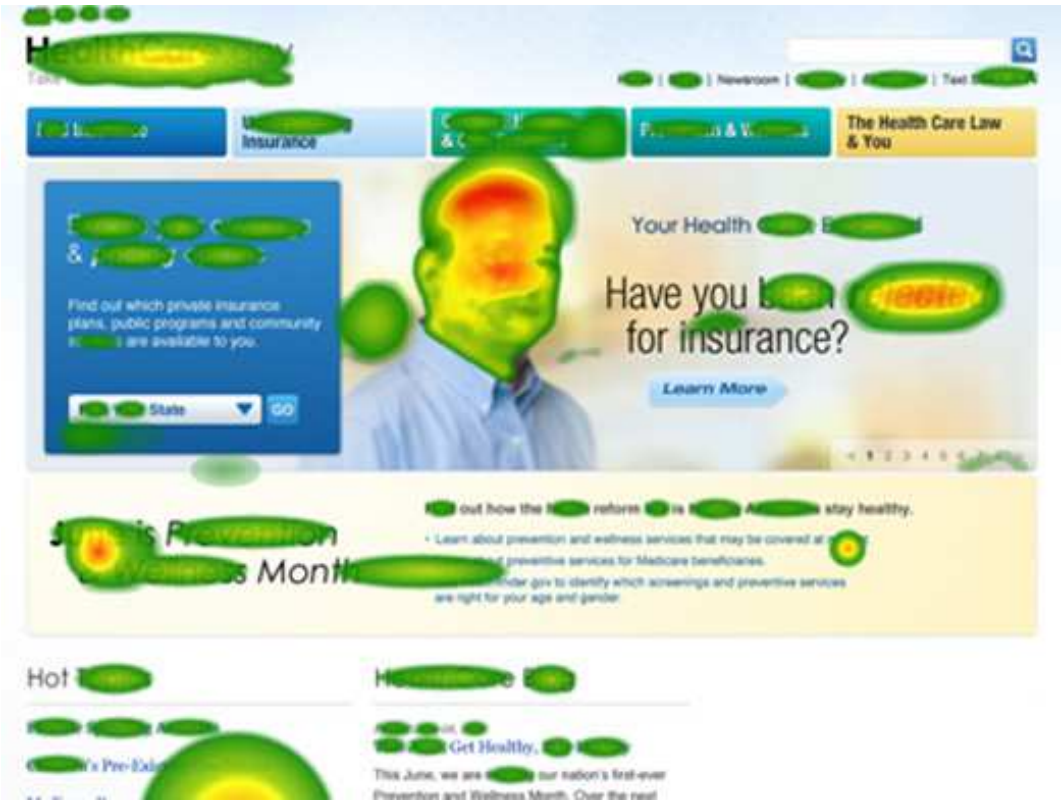


Figure 5.2.: A heat map showing what users look at most on this website

Source: http://www.usability.gov/how-to-and-tools/methods/eye-tracking.html, 2016-01-02

In addition to tracking the movement of the eyes, users can also be asked to "think aloud" when using the website to provide better information on why certain elements attract them more than others or what they think when looking at them.

[9] Eye Tracking Method: http://www.usability.gov/how-to-and-tools/methods/eye-tracking.html , 2016-01-02

5.1.3.3. Focus groups

Focus groups consist of multiple participants (usually between 5 and 10) and a moderator who leads the discussion. The participants should ideally resemble the target audience of your project, so the results are representative and can be used to get the right impulses for your project.[10]

The group talks about and discusses possible concepts, ideas and, for instance, existing designs, often with the help of creative techniques.

It is most commonly used during the planning phase of a project to determine the users' needs and preferences concerning a specific product such as a website or a mobile application.

Focus groups are a great way for:

- finding out how the users feel about your product, if it is what you want them to associate it with.
- finding our how users react to certain elements/ideas of your project.
- getting new ideas inspired by the participants input.

5.1.3.4. Other testing methods

Some other of the countless testing methods: [11]

- Ethnographic Field Studies
  Users are studied in their natural environment to gain realistic results.
- Interviews
  Researcher meets with user to question and discuss the users opinion on the topic.
- Diary Studies
  Users document their thoughts, feelings, etc. about a topic or product in regular intervals.
- Clickstream Analysis
  The screen inputs are recorded to gain knowledge about how users use and navigate the, for example, website.
- Email Survey
  Users are questioned about the program or topic via email.

[10] Focus groups method: http://www.usability.gov/how-to-and-tools/methods/focus-groups.html , 2016-02-02

[11] UX research Methods, https://www.nngroup.com/articles/which-ux-research-methods/, 2016-02-11

5.2. Usability for Mobile Apps

When it comes to mobile applications, usability and a consistent design are extremely important in order to create a well received product.

Usability guidelines help to keep functionality and design consistent across different applications and ensure that the user quickly gets accustomed to new applications. This, however is not easy, especially on mobile devices where the screen size is small and the application is often used in user-unfriendly situations such as bright daylight or stressful situations.

5.2.1. General usability and design principles

- Keep it sweet and simple.
  People don't like reading a lot of text, in fact, many people will skip over texts if they seem to take too much time to read. Using simple language and short sentences helps keeping the users attention.

- Use of images and icons.
  In many situations, using images and icons to illustrate the meaning or role of certain elements of your app is much more efficient than merely using text. They deliver a more pleasant user experience and a more beautiful application overall.

- Visibility and accessibility.
  All the content of an application should be displayed in a big and clear enough manner for the user to easily distinguish it from other elements inside your app. Furthermore, all the widgets of your app not only have to be easy to detect and identify as what they are but also easy to access, since touch control is not the most precise way to navigate and falsely selecting the wrong item can be really frustrating.

Figure 5.3.: Items have to be easily identifiable as what they are

Source: http://www.peachpit.com/articles/article.aspx?p=2209309, 2016-02-11

- Keep it consistent.
  In order to make that sure users don't get confused when navigating through your application, the look and operation of the different screens have to be consistent. Not only do menus and often occurring options have to be in the same place all the time, but it is also important that everything acts like it looks. For example, a basic button the user already

knows from various other applications must not be without any function, just present for aesthetic reasons.

- Leaving space for customization
  The design principle of android aims at giving users the best possible options to adjust settings, looks and functions. To support that mindset, choose for the user but give them options to change things if they want to.[12]

- Placement of clickable content
  Especially when using smartphones with a big screen, it can be problematic to reach every corner of it when using the smartphone with only one hand. Therefore, buttons, links, etc. ideally should be placed so the user can easily reach them.

Figure 5.4.: Important elements should always be in reach of the user when using the application one-handed

Source: https://www.smashingmagazine.com/2012/07/elements-mobile-user-experience/, 2016-02-11

- Consider usage in different conditions
  This includes, for example, outdoor-use where high contrast is important to ensure the readability of the content or the use of the application in landscape mode where it still has to look and function beautifully. [13]

[12] Android - Design: http://developer.android.com/design/get-started/principles.html, 2016-02-10

[13] Mobile User Experience: https://www.smashingmagazine.com/2012/07/elements-mobile-user-experience/, 2016-02-11

5.3. Usability for the elderly

5.3.1. Why it matters to us

Our mobile application will target users with hardness of hearing or reduced hearing ability. According to the online survey we conducted, about 20 percent of the questioned persons were older than 66 years and about 60 percent were older than 45 years. Since especially older users of smartphones are often not very confident when it comes to using applications, the design of our application has to support them in using it and make the experience as free from potential frustration as possible.

5.3.2. Existing studies and conducted research

While good usability has become a standard for every aspect of user-orientated technology and there are huge studies and a lot of resources about it, the aspect of the elderly-orientated usability, considering their habits and needs gets a lot less attention. In order to collect enough information, we not only looked at studies concerning mobile usability for elderly people but also at studies about web-usability or general usability of computer programs. We then tried to apply the gained knowledge to the circumstances of mobile applications.

5.3.3. Difference between older and younger users

Just as one would presume, users of different age groups act and react differently when using websites (the study concerns websites but is also applicable to mobile applications).

The study, conducted by Jakob Nielsen in May 2013 shows the differences between older and younger users and the hindrances that should be avoided.[14]

5.3.3.1. Physical differences

When growing older, a lot of things about the human body and its abilities tend to change:

- Visual ability
  The older a person gets, the harder they typically find it to read and identify text or graphics, also the ability to differentiate between colours decreases.

- Fine motor skills
  It gets continuously more difficult to precisely click or select what the user wants to.

- Hearing ability
  Hardness of hearing is very common amongst elderly people, but most of the time has no impact on using a website or an application.

[14] Mobile User Experience: https://www.nngroup.com/articles/usability-for-senior-citizens/, 2016-02-11

5.3.3.2. Behavioural differences

Not only do the physical abilities of a person change as they age, but the general mindset is also heavily affected by this process and leads to behavioural differences between users of different age:

- Hesitation and lack of confidence
  Elderly people tend to be much less likely to explore and try out things they have never done before when it comes to using websites or applications.

- Resilience
  According to the study, older users were more likely to give up on a task a while before younger users did.

- Blaming oneself
  When facing problems, a large portion of elderly users blamed themselves for it while only fewer younger users did so. This constitutes a big source of frustration older users can run into.

5.3.4. Usability rules for the elderly

Source: **?**

Quite often, usability guidelines for elderly people are the same as the ones for regular users. However, they might have a much bigger impact on the user experience when not or falsely implemented into the application. Here are a few guidelines that are especially important for app-design for an older target group: [15]

- Making them feel save
  As it is with the normal design of websites or apps, users must never be able to do something irreversible by accident. For seniors, this is especially important because they tend to be more hesitant and sometimes afraid to do things they don't already know.

- Arrangement of items
  To ensure users do not get puzzled when navigating through your application, the number and arrangement of items has to be well controlled. It is always better to have fewer elements that the user can actually perceive than to have tons of super important information for the user all displayed at once, causing frustration and confusion.

- Compensating for errors or misunderstandings
  During the design and developing process, things that could be done wrong have to be taken into account and ideally stopped from happening at all. This could be something like a hyphen in a place where you don't want it, something that can be fixed by giving an example of how the input should look like.

[15] Mobile User Experience: https://www.nngroup.com/articles/usability-for-senior-citizens/, 2016-02-11

- Avoiding drastic changes
  Most users don't like it when things change because it always triggers a process of reorientating and rethinking, something that especially older people don't like that much.

- Size of things
  While some elderly users might not mind smaller font sizes or pictures, the vast majority will. Therefore, all readable elements of your application should be bigger than usual. This however leads to less displayable content on the screen, so what will be displayed has to be really well thought out. Moreover, the buttons displayed have to be big enough to be easily pressed without accidentally touching something else.

- Language
  The content of the application should always be in the native language of the user, since language barriers can be especially frustrating for older users. Moreover, the kind of language used also has to be well chosen. Not everybody might know what, for example, the „WIFI SSID“ is, so the application must contain as little misunderstandable vocabulary as possible.

- Input Methods
  Since older people often have a hard time typing on the tiny keyboard of smartphones, other input methods such as voice commands or the ability to choose between different keyboard-types should be considered.

- Icons with Text
  While it is good and necessary to use icons in your application instead of just plain text, the use of icons without any explanation can lead to a lot of confusion. Not everyone associates the same things with a certain picture. This can be seen in figure 5.5, where some users might think of a sketching program or a dictionary (because of the „A“ formed by pencil, brush and ruler), instead of a place to download new applications.

Figure 5.5.: Icon of the App Store in IOS9

Source: http://www.appleibite.com/2014/11/22/app-store-free-button-replaces-get/, 2016-02-13

- Gestures
  Complex or hidden gestures should be avoided at all costs, since many people will not even know about their existence and will therefore never use them and maybe miss very important features of the application. This not only includes gestures like swiping from the side of the screen to bring up a menu, but also long presses for more options or the common „pinch to zoom“ feature.

Part III.

Theoretical foundations

6. Audio

6.1. Introduction

6.1.1. What is sound?

What we typically describe as sound are events that our ears can perceive and our brain can understand and process. Sound, and the ability to sense it constitutes a huge part of everybody's daily life. Through sound we can communicate with each other, we get a better idea what is going on around us and we can do so many more things that we don't even think about because it is so natural to do them.

Sound is created by differentiating pressures and densities of a medium such as air or water. If there is no medium, then there is also no sound. Therefore, an alarm clock in a vacuum would produce no sound at all.

The needed differentiating pressures and densities to create sound can be created by:

- mechanical vibrations or oscillations such as a struck guitar string
- a change in the air's volume such as a bottle of sparkling wine being opened

In the air around us are huge amounts of different oscillations that when combined, let us hear the wide variety of complex sounds and noises we perceive instead of just sounds caused by one single oscillation. Everything we hear is a combination of multiple soundwaves, that way, for instance, different timbres can be created by instruments, even if the general pitch is the same.

6.1.2. Why does something sound like it does and not differently?

6.1.2.1. Frequency

The frequency of a sound wave describes how often an event occurs during one second. Normally this event is one cycle of a sine wave.[1] The higher the frequency of a sound (or the faster the medium oscillates) the higher is the pitch of the sound.

In general, frequencies between 20 and 20.000 hertz can be perceived by the human ear. Some young people can perceive sounds with more than 20.000 hertz, older people or people with hardness of hearing typically cannot perceive such high frequencies.

[1] Frequency - Wikipedia, https://en.wikipedia.org/wiki/Audio_frequency, 2016-02-10

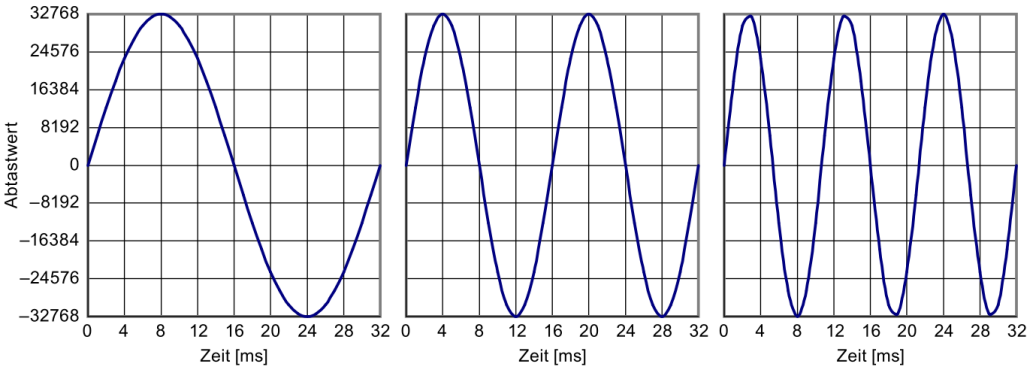


Figure 6.1.: Examples for different frequencies

Source: **?** - page 21

6.1.2.2. Amplitude

The amplitude of a sound is important because it determines its loudness. It describes the difference between the most positive and the most negative value of the oscillation. The bigger the difference, the louder the signal. The most common way to display and compare the loudness of sounds is "sound pressure", measured in dB.

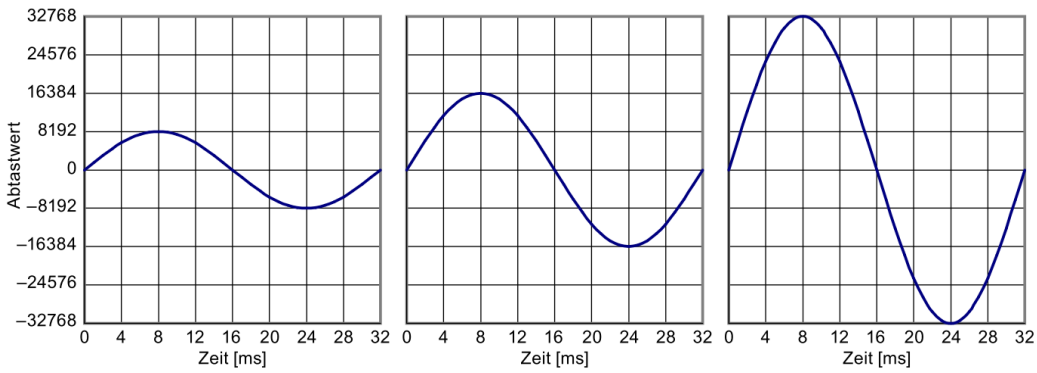


Figure 6.2.: Examples for different amplitudes

Source: **?** - page 20

6.1.2.3. Phase shift

The phase shift describes the starting point of a signal at a certain point in time. This is less important for how we humans perceive sound but more important for accurately describing sound signals and calculating oscillations.

One example for the application of a phase shift are helicopter-headphones. They make use of interference, "*a phenomenon in which two waves superpose to form a resultant wave of greater or lower amplitude*" [2], in order to eliminate any noise made by the helicopter.

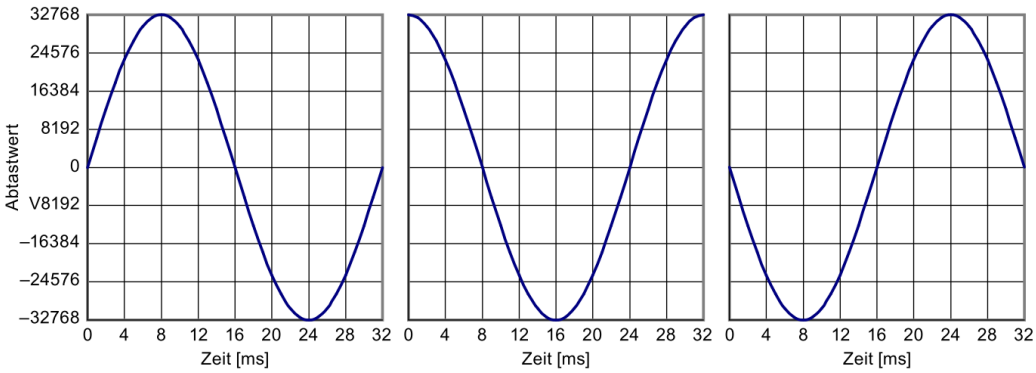


Figure 6.3.: Examples for different phase shifts

Source: **?** - page 22

[2]Interference - Wikipedia, https://en.wikipedia.org/wiki/Interference_%28wave_propagation%29, 2016-02-10

6.2. Audio signals

6.2.1. Basics

In order to handle sound waves on a computer, physical waves must be transformed into digital information. To do so, there are two major possibilities: saving the sound as sampled values or as event messages that specify notation, pitch, velocity, vibration control signals and many other parameters (e.g. MIDI - Musical Instrument Digital Interface).

For recording sound and analysing it afterwards, sampled audio is used because it represents the actual sound and not just theoretical information about it like event messages do.

6.2.2. Pulse-code modulation (PCM)

Pulse-code modulation (PCM) is a method for digitally representing analogue audio signals. By recording sound with a microphone, a continuous signal is transferred into an discrete signal which can be saved digitally.

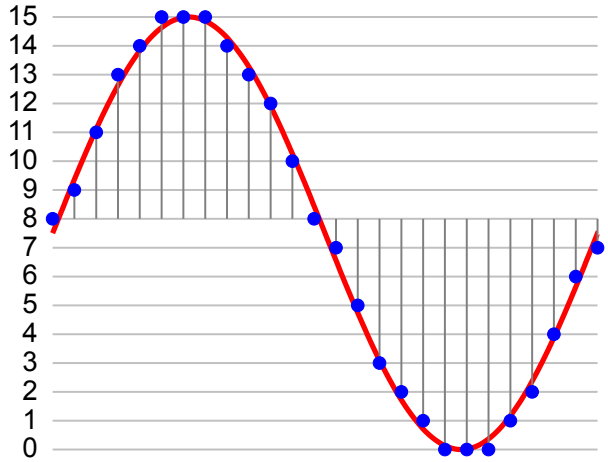


Figure 6.4.: Sampling and quantization of a signal for 4-bit PCM. The red line represents the analogue signal, the blue dots represent the digital values.

Source: https://upload.wikimedia.org/wikipedia/commons/b/bf/Pcm.svg, 2016-02-08 11.41

The process of generating a PCM stream can be divided into the following steps:

1. **Sampling:** The magnitude of an analogue signal is measured at uniform time intervals. These values are called samples.

2. **Quantification:** The samples are digitalised. To do so, the samples are rounded up or down to the nearest binary value.

3. **Encoding:** The encoding defines the number of samples per second and the precision of the samples (e.g. 16 bit depth).

The generated PCM stream contains the amplitude of the analogue signal, sampled regularly at uniform intervals. The fidelity of the PCM stream is based on the sampling rate which is the number of times per second that samples are taken, and the bit depth which determines the number of possible digital values that can be used to represent each sample (quantification).

When speaking of PCM, what is mostly meant is linear pulse-code modulation (LPCM), where the quantisation levels are linearly uniform. Other PCM encodings vary the quantisation level as a function of the amplitude (e.g. A-law algorithm).

6.2.3. Presentation of audio signals

Audio files can be presented in two fundamental styles, as time or frequency domain. The style describes whether the frequency or the time is plotted onto the x-axis. When speaking about the waveform plot, the time form is meant and when speaking about a spectrum, the frequency form of the signal is meant. A combined form of both types is named spectrogram.

6.2.3.1. Waveform

This presentation shows the sample values in relation to the time. Therefore, the magnitude of the signal is presented. The magnitude itself can be scaled relative, as values between -1 and 1, or absolute as decibel values.

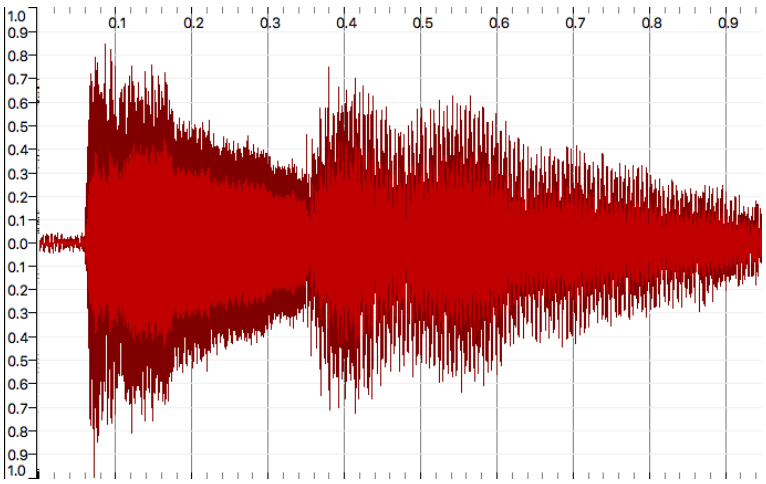


Figure 6.5.: Waveform plot of a doorbell. The magnitude is scaled between -1 and 1, the sound is approximately one second long.

Source: Sonic Visualiser

6.2.3.2. Spectrum

The spectrum presents a stationary view of the audio signal. The diagram shows the magnitudes of the corresponding frequency. Therefore, the plot displays the intensity of every frequency. The favoured magnitude scale is decibel, nonetheless a relative scale can also be used.

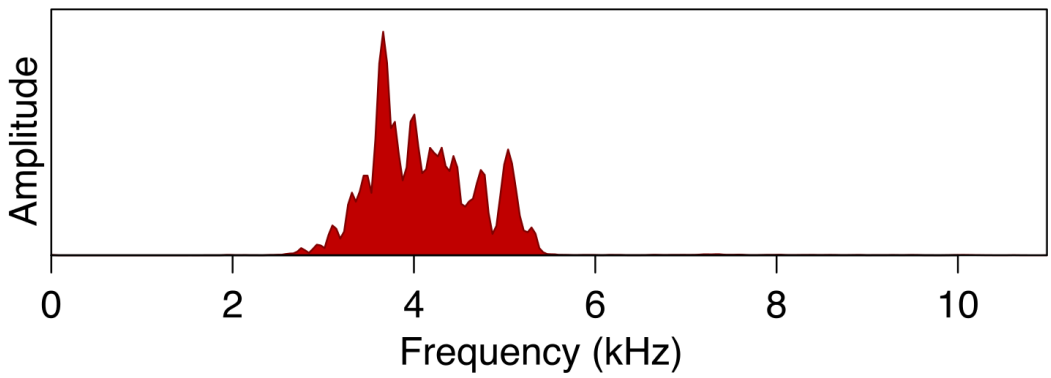


Figure 6.6.: Spectrum of a chirp. The amplitude maximum is between 3 and 4 kHz. All major frequencies are between 2 and 6 kHz.

Source: R (with the package seewave)

6.2.3.3. Spectrogram

A spectrogram is a combination of a waveform plot and a spectrum. Therefore, this plot has three dimensions: magnitude, time and frequency. The time is displayed on the x-axis and the frequency on the y-axis. The magnitude is commonly presented by colours.

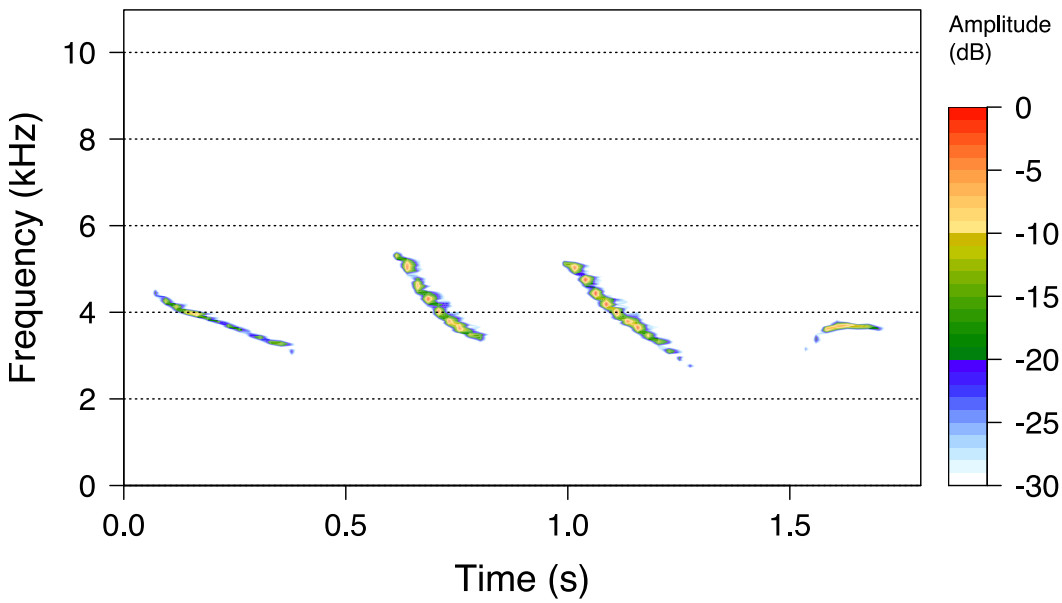


Figure 6.7.: Spectrogram of a chirp. The redder the colour, the higher the amplitude.

Source: R (with the package seewave)

6.2.4. Audio files

6.2.4.1. Sample rate

The sample rate of an audio file describes the number of samples per second. Since it is important to cover all needed frequencies, the sample rate must be high enough. According to the Nyquist-Shannon sampling theorem the sampling rate must be greater than the double of the highest desired frequency in order to enable a correct recreation of the original sound. For most audio files a sampling rate of 44.1kHz is used since humans are able to hear frequencies up to 20kHz. A higher sample rate would not deliver any benefits for humans.

6.2.4.2. Sample size / bit depth

The bit depth defines how many bits there are for representing an amplitude. The number of possible values with n bits is equal to 2^n. When it comes to audio files, sample sizes of 8, 16 or 24 bits are used.

6.2.4.3. Channels

The number of channels describes how many different tracks are within an audio file. One channel only enables mono-audio files, two channels enable stereo sound.

6.2.4.4. Encodings

The objective of the encoding is to represent the high-fidelity audio signal with a minimum number of bits while retaining the quality. This can effectively reduce the storage space and the bandwidth required for the audio file. Altogether there are three categories of codecs: non-compression, lossless and lossy compression codecs.

6.2.4.5. Common file formats

The following file formats are widely spread and often used:

- Uncompressed formats: WAVE (Waveform Audio File Format), AIFF (Audio Interchange File Format) and AU (Au file format)
- Compressed, lossless formats: FLAC (Free Lossless Audio Codec), ALAC (Apple Lossless Audio Codec) and TTA (True Audio)
- Compressed, lossy formats: MP3 (MPEG Layer III Audio), Advanced Audio Coding (AAC), Vorbis and Windows Media Audio (WMA)

6.3. Feature Extraction

6.3.1. What is feature extraction?

Feature extraction enables getting specific informations out of measured data. Therefore, feature extraction is the first part of analysing data, since derived values are built. These features aim to be informative and non-redundant in order to be interpretable afterwards. Thus, the extracted features contain the relevant information from the input data, so that the detection algorithm can be performed by using this reduced representation instead of the complete initial data.

6.3.2. Fourier transform

Source: **?**

The Fourier transform constitutes a foundation for many features and is therefore an important part of extracting audio features. The Fourier transform enables converting time based signals (waveform signals) into frequency information (spectrum). The underling thesis is that any signal only consists of different sine and cosine waves. The mathematical description of a signal is therefore the sum of the partial oscillations.

$$f(t) = \sum_{k=1}^{N} a_k \; cos(k\omega t) + b_k \; sin(k\omega t) \tag{6.1}$$

Since sine waves are basically cosine waves with a phase shift of 90 degree, it is possible to describe the resulting wave only with partial cosine waves if the shift (ϕ) of every partial wave is given.

$$f(t) = \sum_{k=1}^{N} A_k \; cos(k\omega t + \phi) \tag{6.2}$$

$$A_k = \sqrt{a_k^2 + b_k^2} \; , \; \phi = atan\left(\frac{b_k}{a_k}\right)$$

For calculating the spectrum of a given time function $f(t)$, it is needed to resolve the coefficients a_k and b_k.

$$a(k) = \frac{2}{T} \int_0^T f(t) \; cos(k\omega t) \; dt \tag{6.3}$$

$$b(k) = \frac{2}{T} \int_0^T f(t) \; sin(k\omega t) \; dt \tag{6.4}$$

These coefficients can be summarised to a complex number ($c_k = a_k - ib_k$). Further, the mathematical relation $e^{-ix} = cos(x) + i\ sin(x)$ can be used to describe the equation for the complex Fourier coefficient.

$$c(k) = \frac{2}{T} \int_0^T f(t)\ e^{-ik\omega t}\ dt \tag{6.5}$$

Equation 6.5 is the common form of the Fourier series.

6.3.2.1. Different Fourier Transforms

The Fourier transform can be divided into four subgroups, based on the "character" of time and duration. The time can either be discrete or continuous and the duration can be finite or infinite.

Time	Duration		
	Finite	Infinite	
Discrete	Discrete FT (DFT) $c(k) = \sum_{n=0}^{N-1} f(n)\ e^{-i\omega_k n}$ $k = 0, 1, \ldots, N-1$	Discrete Time FT (DTFT) $c(\omega) = \sum_{n=-\infty}^{\infty} f(n)\ e^{-i\omega n}$ $\omega \in (-\pi, \pi)$	discr. time n
Continuous	Fourier Series (FS) $c(k) = \frac{1}{T} \int_0^T f(t)\ e^{-i\omega_k t}\ dt$ $k = -\infty, \ldots, \infty$	Fourier Transform (FT) $c(\omega) = \int_{-\infty}^{\infty} f(t)\ e^{-i\omega t}\ dt$ $\omega \in (-\infty, \infty)$	cont. time t
	discrete freq. k	continuous freq. ω	

Table 6.1.: Different Fourier transforms

Source: http://www.dsprelated.com/josimages_new/sasp/img82.png, 2016-02-10 23:53

For analysing audio signals, the discrete Fourier transform (DFT) was used, since we have discrete time values and a finite duration.

6.3.2.2. Discrete Fourier transform (DFT)

The discrete Fourier transform (DFT) converts a finite sequence of equally spaced samples into a finite number sinus waves. Therefore, the original time domain function is converted into a frequency domain function.

Furthermore, there is a relation between the frequency resolution and the window length. The longer the window length, the deeper frequencies are possible ($f = 1/T$). For instance, if a signal is one second long, the lowest possible frequency is $f = 1/T = 1/1s = 1Hz$.

Example

The following example shows the process of generating a spectrum out of a given time domain input signal (figure 6.8).

Figure 6.8.: DFT: The given input signal

Source: https://upload.wikimedia.org/wikipedia/commons/5/50/Fourier_transform_time_and_frequency_domains.gif, 2016-02-11 10:49

Now, the given input signal is substituted into the containing partial cosine waves. The combination of these partial waves is equal to the original signal (figure 6.9).

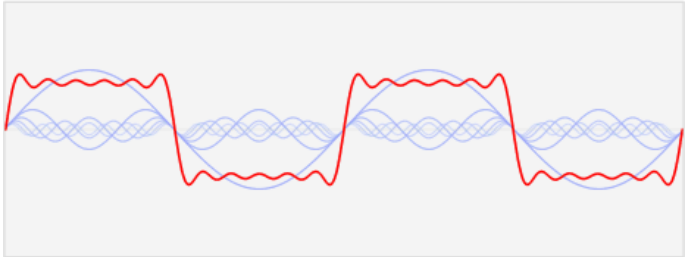


Figure 6.9.: DFT: Partial waves of the given signal

Source: https://upload.wikimedia.org/wikipedia/commons/5/50/Fourier_transform_time_and_frequency_domains.gif, 2016-02-11 10:49

Afterwards, the amplitudes of the partial cosine waves are measured and transferred into a frequency domain function (figure 6.10).

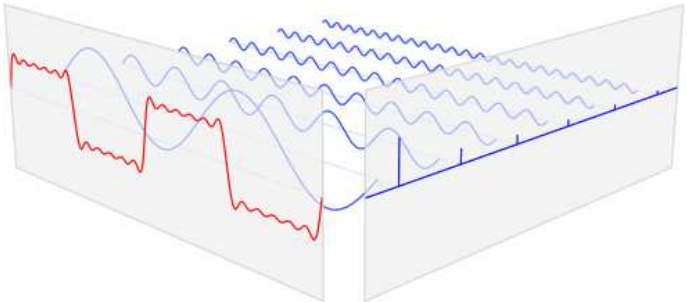


Figure 6.10.: DFT: Time and frequency domain of the signal

Source: https://upload.wikimedia.org/wikipedia/commons/5/50/Fourier_transform_time_and_frequency_domains.gif, 2016-02-11 10:49

The resulting function is called spectrum (figure 6.11). This spectrum can be used to perform further operations, like feature extraction.

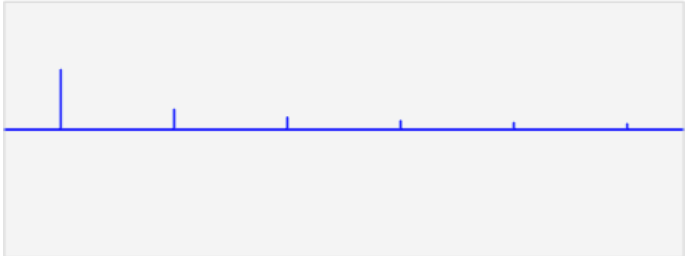


Figure 6.11.: DFT: The resulting frequencies (spectrum)

Source: https://upload.wikimedia.org/wikipedia/commons/5/50/Fourier_transform_time_and_frequency_domains.gif, 2016-02-11 10:49

6.3.2.3. Fast Fourier Transform (FFT)

When it comes to analysing sounds, sample rates of 44.100 Hz are common. When analysing a one-second-long signal, this would imply $44.100^2 \approx 2$ *billion* calculations, since the effort increases by the square of the window length. This huge amount of calculations is not compatible with a real time audio analysis.

In order to avoid so many calculation, an algorithm was developed by James Cooley and John W. Tukey. The FFT algorithm reduces the computing effort by using recursive calculation to $N * log_2(N)$.

With the previous example, this would mean, that only $44.100 * log_2(44.100) \approx 680.000$ calculations are needed, which is a tiny fraction of the calculations needed above.

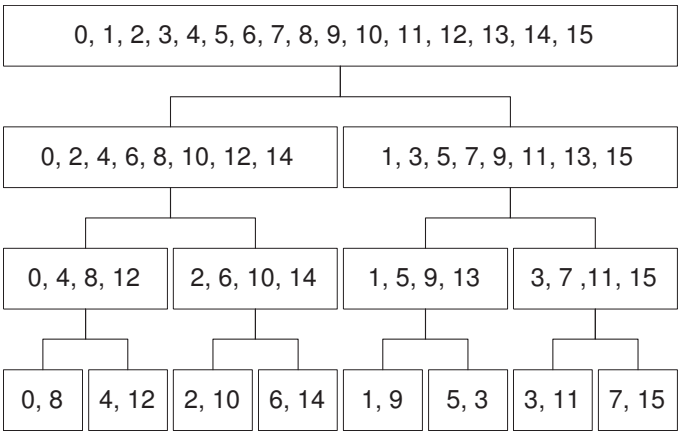


Figure 6.12.: FFT algorithm: The grid points are split into subgroups, until only two are left.

Source: **?** - page 109

Due to the operating method of the algorithm, only window sizes of the power of two are possible (e.g. 512, 1024, 2048 or 32768).

6.3.3. Window function

Source: **?**

6.3.3.1. Problem

The windowing function is used, when it comes to generating a frequency spectrum in order to avoid an effect known as leakage. Leakage occurs by analysing a block of non-periodic data, since the FFT assumes the input signal to be periodic. Therefore, the subsequent spectrum is not entirely correct.

For instance, when a 10 Hz sine signal is computed with the FFT, the resulting spectrum shows a single, discrete peak of 1.0 at 10 Hz (figure 6.13 left side). However, the same signal, but entirely periodical, creates a deformed spectrum, it suffers from leakage. The peak is still clearly visible, but less high. Additionally, the frequency range is wider, because the signal energy is more dispersed (figure 6.13 right side).

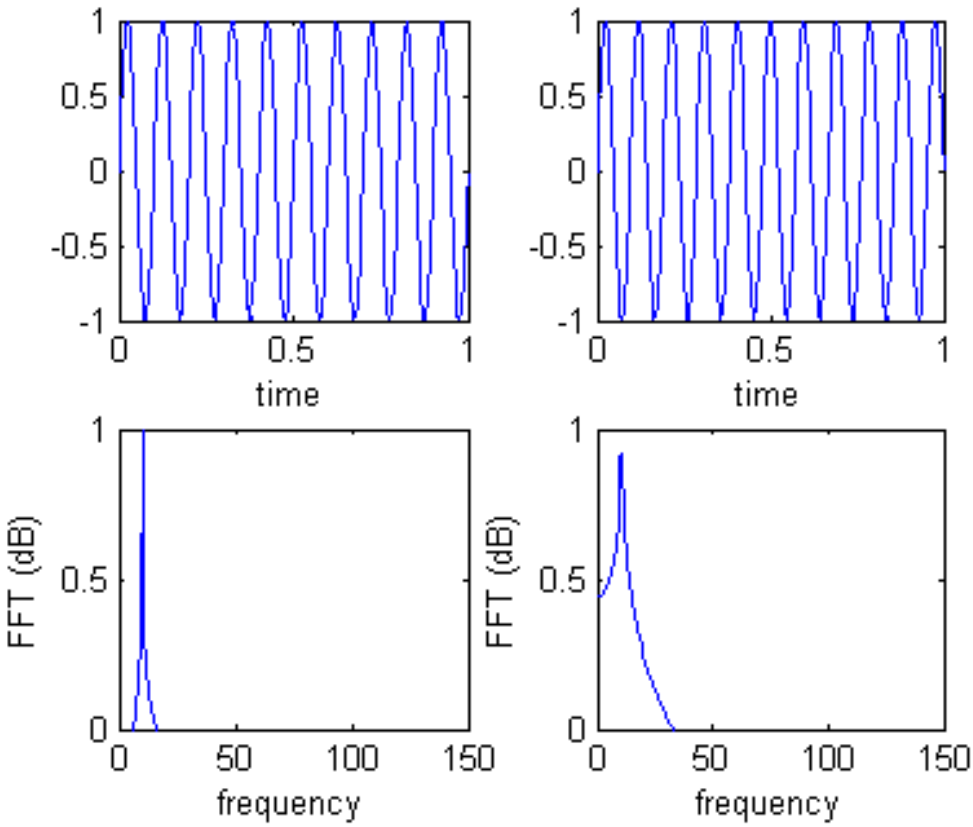


Figure 6.13.: Leakage effect: A periodic signal, on the left, compared to an non-periodic signal, on the right.

Source: http://www.physik.uni-wuerzburg.de/fileadmin/11016800/Modulbeschreibungen/CMT/ANO14.pdf (page 2), 2016-02-10 14:42

6.3.3.2. Solution

In order to reduce the leakage effect, a window is applied. For this process, the signal is weighted according to a defined function. A window is shaped so that it is exactly zero at the beginning and end of the data. When a window is applied to the previous, non-periodic signal, the resulting spectrum becomes clear again (figure 6.14).

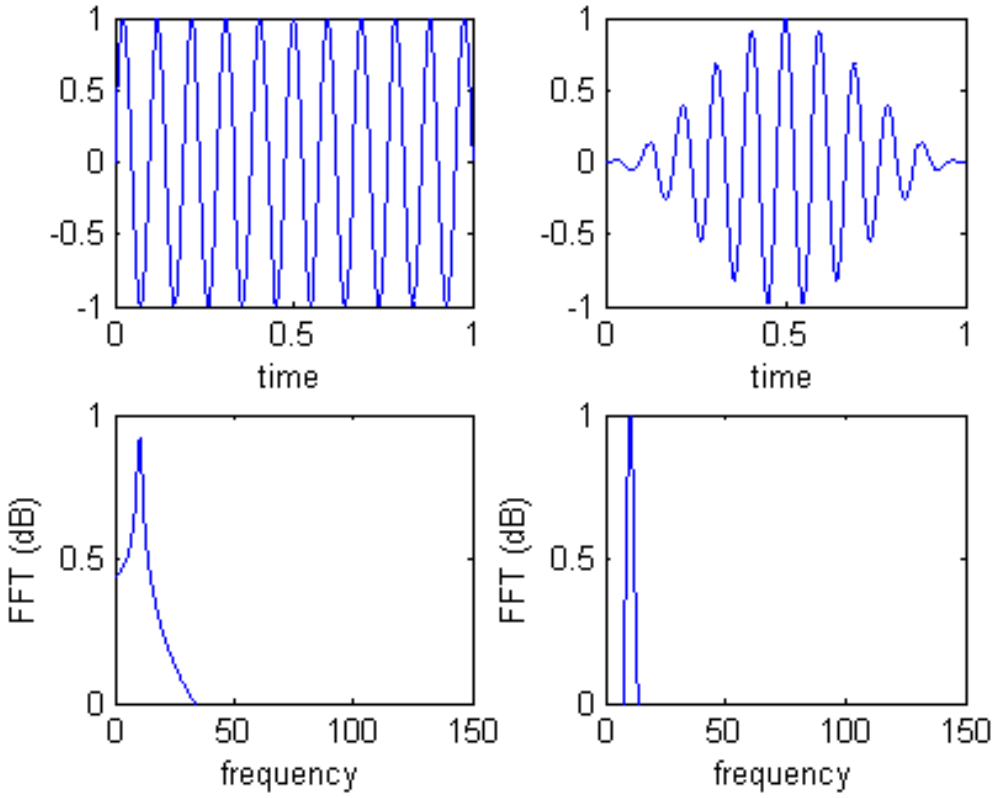


Figure 6.14.: Reducing the leakage effect: a non periodic signal (left) generates a dispersed spectrum, the windowed signal (right) shows no leakage.

Source: http://www.physik.uni-wuerzburg.de/fileadmin/11016800/Modulbeschreibungen/CMT/ANO14.pdf (page 2), 2016-02-10 14:53

6.3.3.3. Common window functions

Source: https://en.wikipedia.org/wiki/Window_function#A_list_of_window_functions, 2016-02-10 15:00

The aim of the window functions is to provide a good frequency resolution, little spectral leakage and a high amplitude accuracy.

N = width of the window

w(n) = current weight

Rectangular window (Boxcar)
A rectangular window is the simplest function. It does not amplify or reduce the samples of the input signal, it only enables shifting the analysis window.

$$w(n) = 1 \tag{6.6}$$

Triangular window (Bartlett window)
The triangular window is simpler than the cosine-based windows, but has much higher side-lobe frequencies. The Bartlett window is a triangular window with non-zero endpoints.

$$w(n) = 1 - \left| \frac{n - \frac{N-1}{2}}{\frac{L}{2}} \right| \tag{6.7}$$

L can be N, N+1 or N-1 (N-1 is called Bartlett window)

Hanning (Hann) window
The Hanning window is based on a raised cosine function.

$$w(n) = 0.5 \left(1 - cos\left(\frac{2\pi n}{N-1}\right)\right) = hav\left(\frac{2\pi n}{N-1}\right) \tag{6.8}$$

Hamming window
The Hamming window is similar to the Hanning window, but optimised to minimise the nearest side-lobe frequency content.

$$w(n) = \alpha - \beta\ cos\left(\frac{2\pi n}{N-1}\right) \tag{6.9}$$
$$\alpha = 0.54;\ \beta = 1 - \alpha = 0.46$$

Blackman window
The Blackman window is based on a cosine function. It has less sideband leakage than the comparable Hamming and Hanning windows.

$$w(n) = a_0 - a_1\ cos\left(\frac{2\pi n}{N-1}\right) + a_2\ cos\left(\frac{4\pi n}{N-1}\right) \tag{6.10}$$
$$\alpha = 0.16;\ a_0 = \frac{1-\alpha}{2};\ a_1 = \frac{1}{2};\ a_2 = \frac{\alpha}{2}$$

Plot of the previous functions

The following plot summarises the previous window functions graphically.

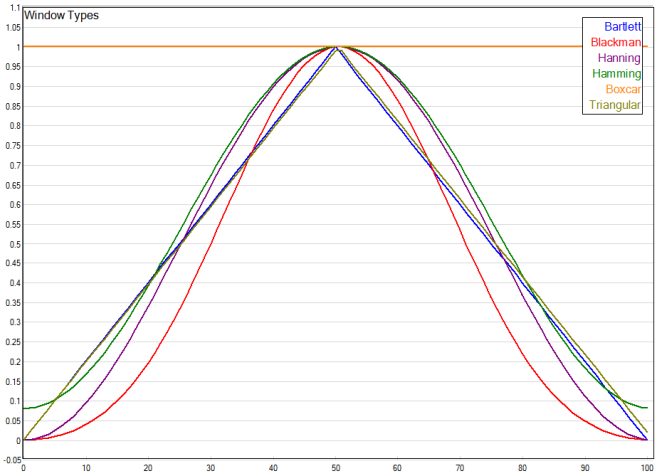


Figure 6.15.: Common window functions

Source: https://www.delsys.com/KnowledgeCenter/NetHelp/Media/Window%20functions.png, 2016-02-10 15:16

6.3.3.4. Overlapping

The problem with windowing is that the beginning and the end of a signal is neglected. In order to compensate this effect, multiple overlapping windows are taken out of the input signal. Each of these windows is analysed separately. After the analysis, the partial results are being added up. The subsequent result therefore is the average of all window results.

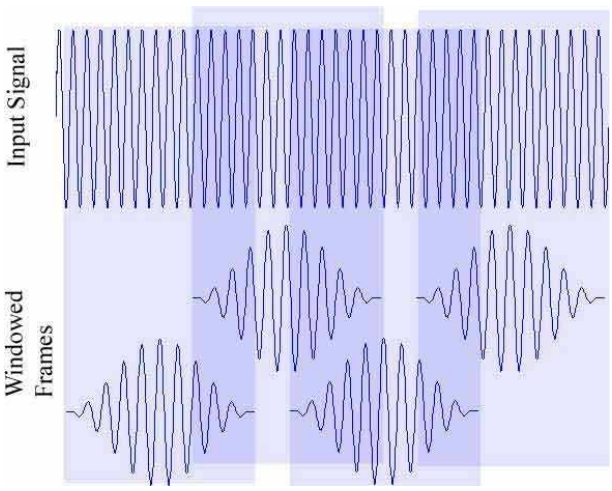


Figure 6.16.: Window function: Overlapping

Source: **?** - page 5

6.3.4. Time based features

Source: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.125.8354&rep=rep1&type=pdf, 2016-02-11 14:37

Contrary to spectral features, the values of these features are gained by analysing the time domain function (waveform). These features are relatively easy to calculate but are vulnerable to background noise (e.g. music, talking,...). Hence, it is needed to filter out background noise as well as possible.

6.3.4.1. Root-Mean-Square (RMS) energy

The Root-Mean-Square (RMS) energy represents the loudness of a sound event. For calculating the feature, all amplitudes are squared, then summed up and divided by the number of amplitudes to get the square mean. Afterwards the root is applied to get the desired feature value.

$$x_{rms} = \sqrt{\frac{1}{N} \cdot \sum_{n=1}^{N} x(n)^2} \tag{6.11}$$

6.3.4.2. Short-Term-Energy (STE)

The Short-Term-Energy (also Short-Time-Energy) is used for short time intervals. This feature returns the time based energy change and is often used for detecting a sound event.

$$E_m = \sum_{n=1}^{N} x(n)^2 \times w(m-n)^2 \tag{6.12}$$

The term $w(m-n)$ describes the Hanning window, which should be applied for the calculation of this feature.

6.3.4.3. Zero-Crossing Rate (ZCR)

The Zero-Crossing Rate (ZCR) measures, how often a signal crosses the x-axis. The rate at which zero crossings happen is a simple measure of the frequency content of a signal. For narrow band signals, the average zero crossing rate gives a reasonable way to estimate the frequency content of the signal. But for a broad band signal such as speech, it is much less accurate.

$$Z_t = \sum_{n=1}^{N-1} |sign[x(n)] - sign[x(n+1)]| \tag{6.13}$$

6.3.5. Spectral features

Spectral features are based on the frequency domain function (spectrum). These features enable analysing the frequency constitution of a given sound. Therefore it is possible to get example given the strongest frequency or the distribution of all frequencies.

6.3.5.1. Strongest frequencies

The strongest frequency function returns the highest local peaks within the spectrum. Therefore, this feature reveals the dominant frequencies of a signal, regardless of the bandwidth. One big advantage of this feature is the background noise resistance. Additionally, a threshold can be set, with the benefit of neglecting peaks below this line. Hence, only loud enough frequencies are returned.

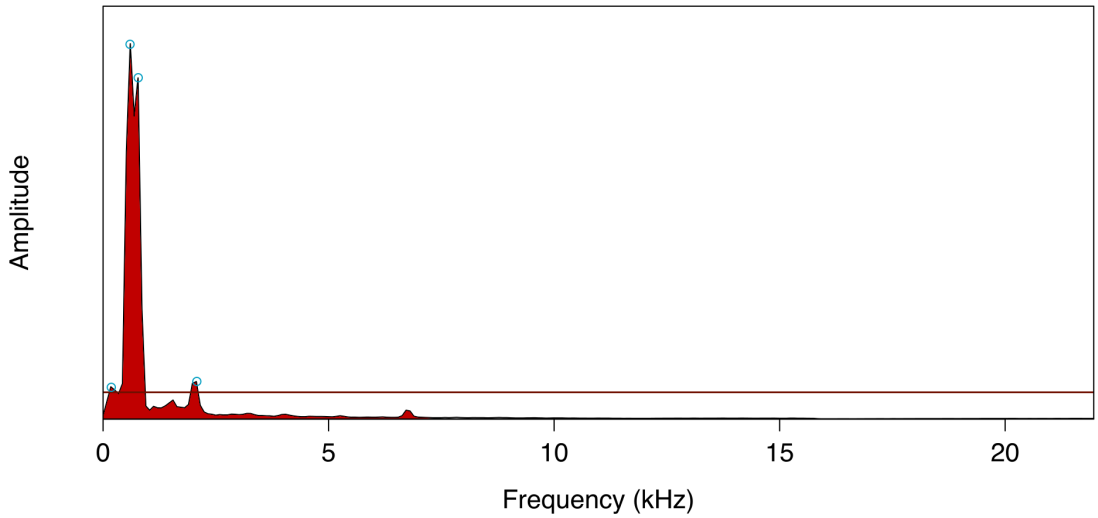


Figure 6.17.: Strongest frequencies of a given spectrum

Source: R (with the package seewave)

6.3.5.2. Spectral Rolloff Point (SRP)

Spectral Rolloff Point (SRP) is the frequency below which the X% of the magnitude distribution is concentrated. X is typically a value between 85% and 95%.

This measure is useful for distinguishing voiced speech from unvoiced speech: unvoiced speech has a high proportion of energy contained in the high-frequency range of the spectrum, where most of the energy for voiced speech and music is contained in lower bands.

$$\sum_{i=1}^{R_t} m(f) = X \times \sum_{i=1}^{N} m(f) \tag{6.14}$$

$R_t \ldots$ *Spectral Rolloff Point,* $m(f) \ldots$ *magnitude of frequency* f

6.3.5.3. Mel-Frequency Cepstral Coefficients (MFCCs)

Source: http://www.practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/, 2016-02-11 15:22

When it comes to analysing speech, Mel Frequency Cepstral Coefficients (MFCCs) are the first choice. They were introduced by Davis and Mermelstein in the 1980's, and have been state-of-the-art ever since. Prior to the introduction of MFCCs, Linear Prediction Coefficients (LPCs) and Linear Prediction Cepstral Coefficients were the main type for speech recognition.

MFCC Calculation

These steps are performed when calculating the MFCCs:

1. Windowing: Splitting the signal into short frames
2. FFT: Calculating the spectrum for each frame
3. Filterbank: Applying the Mel filterbank to the spectra
4. Logarithm: Taking the logarithm of all filterbank energies
5. DCT: Taking the DCT (Discrete Cosine Transform) of the log filterbank energies
6. Finish: Keeping DCT coefficients 2-13, discarding the rest

The Mel scale

At the third step, the Mel filterbank is applied, because this makes the scale closer to what humans hear. The Mel scale relates the perceived frequency, or pitch, of a pure tone to its actual measured frequency. Humans are much better at discerning small changes in pitch at low frequencies than they are at high frequencies.

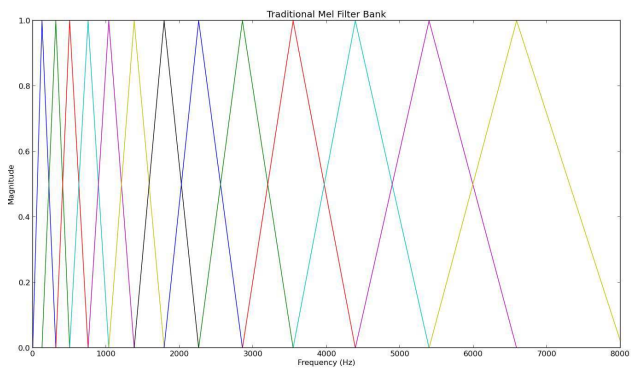


Figure 6.18.: Traditional Mel Filter Bank

Source: http://www.vocal.com/wp-content/uploads/2013/06/MelFilterBank.jpeg, 2016-02-11 16:48

6.4. Classification

The values obtained through the feature extraction have to be classified now in order to predict if a recorded sound occurred and if so, which one. Classification deals with interpreting prior calculated values and is therefore also an important part of analysing audio data.

6.4.1. Threshold comparison

When it comes to analysing a sound in order to detect whether an event occurred or not, threshold comparison is the simplest form. This method compares an input value to a predefined threshold and returns whether the value was below or above the threshold.

For instance in case of an audio analysis software the loudness of the surrounding environment can be permanently extracted. When the loudness level is below the specified threshold, it is interpreted as silence. When loudness increases above the threshold, the algorithm alerts that a sound event occurred (figure 6.19).

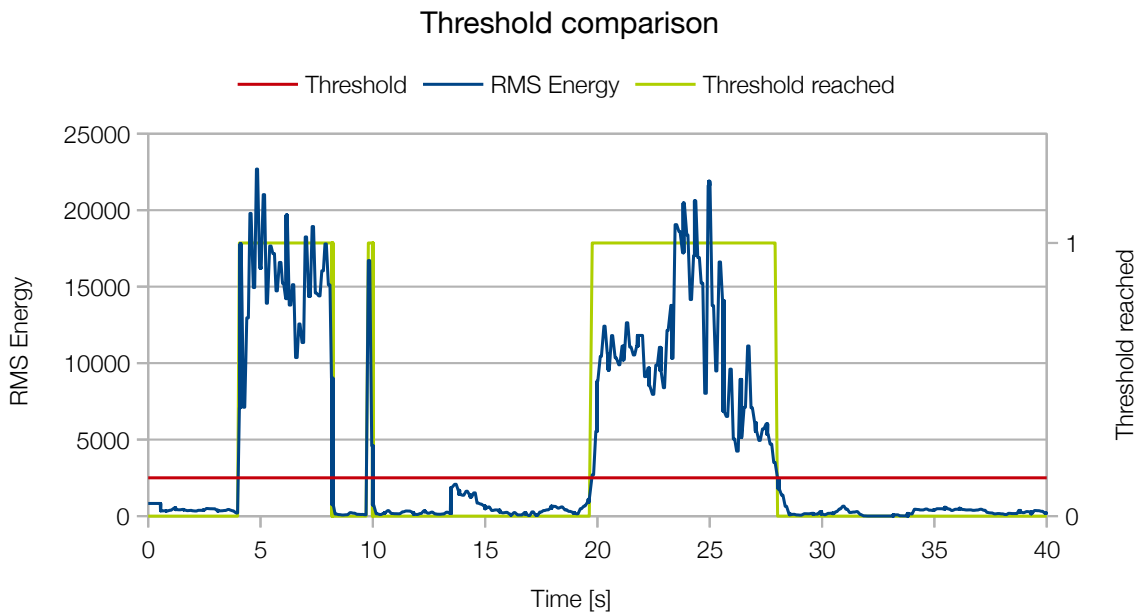


Figure 6.19.: RMS energy threshold comparison

Source: LibreOffice Calc

In addition to its simplicity, another advantage of this method is the suitability for more than one feature. For example, a sound is only classified as doorbell, if the minimum loudness is reached and the Spectral Rolloff Point (SRP) is above a certain frequency.

Furthermore, an upper and a lower limit threshold can be declared in combination. This is beneficial when the feature should be between two values. For instance, the SRP of a doorbell should be between 4kHz and 12kHz.

6.4.2. Multivariate statistics

Source: https://en.wikipedia.org/wiki/Multivariate_statistics, 2016-02-12 12:51

Multivariate statistics covers simultaneous observation and analyses multiple out-coming variables. Applying multivariate statistic methods enables analysing multiple variables (e.g. audio features) at once. Multivariate statistics help to understand the relationships between variables and analyses and their relevance to the whole observation.

6.4.2.1. Example: Discriminant function analysis

Source: https://de.wikipedia.org/wiki/Diskriminanzanalyse, 2016-02-12 13:59

A hotel near a Christmas market wants to analyse how likely it is that people stay for a night. To do so, the amount of money spent on souvenirs at the market is analysed by taking a sample of 32 people consisting of 16 people who stayed (group 1) and 16 who did not (group 2).

Money spent [Euro]			
Group 1		Group 2	
Total	On souvenirs	Total	On souvenirs
64,78	37,08	54,78	17,08
67,12	38,44	57,12	18,44
71,58	44,08	61,58	24,08
63,66	37,40	53,66	17,40
53,80	19,00	43,80	7,99
73,21	41,17	63,21	29,10
63,95	31,40	53,95	11,40
78,33	45,92	68,33	34,98
72,36	38,09	62,36	18,09
64,51	34,10	54,51	14,10
66,11	34,97	56,11	14,97
66,97	36,90	56,97	16,90
69,72	41,24	59,72	21,24
64,47	33,81	54,47	13,81
72,60	19,05	62,60	30,02
72,69	39,88	62,69	19,88

Table 6.2.: Values for the discriminate function analysis example

Based on the samples, the according means and covariances are calculated:

Variable	Means		
	Pooled	Group 1	Group 2
Total	62,867	67,867	57,867
On Souvenirs	27,562	35,783	19,342

Table 6.3.: Means of the samples of the discriminate function analysis example

Pooled covariance matrix		
	Total	On souvenirs
Total	32,59	
On souvenirs	30,58	54,01

Table 6.4.: Pooled covariances of the samples of the discriminate function analysis example

Covariance matrix for group 1		
	Total	On souvenirs
Total	32,59	
On souvenirs	25,34	56,90

Table 6.5.: Covariances for group 1 of the samples of the discriminate function analysis example

Covariance matrix for group 2		
	Total	On souvenirs
Total	32,59	
On souvenirs	35,82	51,11

Table 6.6.: Covariances for group 2 of the samples of the discriminate function analysis example

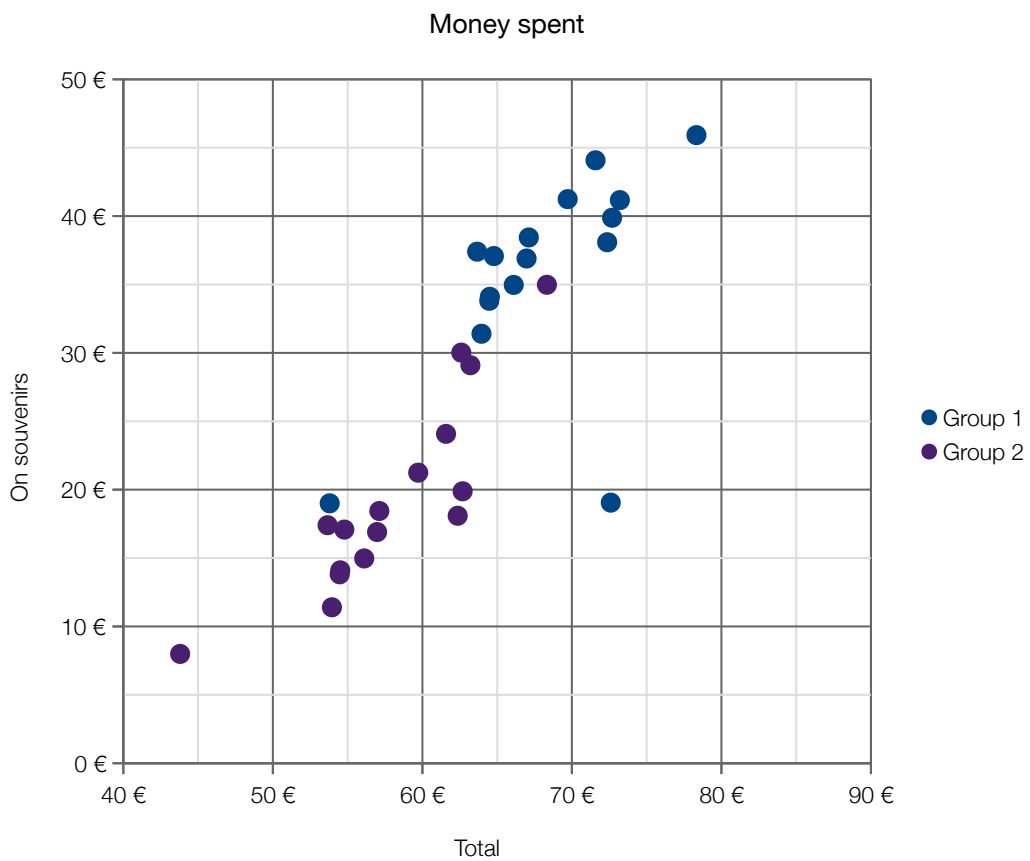


Figure 6.20.: Graphical plot of the discriminate function analysis example

Source: LibreOffice Calc

Calculation of the discriminant function for two groups. If $f(x) > 0$, then the value is a member of the first group.

$$f(x) = \left(x - \frac{1}{2}(\mu_I + \mu_{II})\right)^T \sum^{-1} \left(\mu_I - \mu_{II} = \frac{1}{2}(d_{II}(x) - d_I(x)\right) \tag{6.15}$$

In our case the result would be:

$$f(x) = -4.690 + 0.02 \times total + 0.124 \times souvenir \tag{6.16}$$

The sample values can be used for testing the model. The following table shows the correctness of the classification function:

Group	Classification	
	Correct	Incorrect
1	14	2
2	13	3

Table 6.7.: Correctness of the classification of the discriminate function analysis example

6.4.2.2. Important multivariate statistical methods

Important multivariate statistic methods are:

- Multivariate regression:
  attempts to determine a formula that can describe how elements in a dataset of variables respond simultaneously to changes in others
- Canonical correlation analysis:
  finds linear relationships among two sets of variables
- Linear discriminant analysis (LDA):
  computes a linear predictor from two sets of normally distributed data to allow for classification of new observations
- Clustering systems:
  assign objects into groups (called clusters) so that objects from the same cluster are more similar to each other than objects from different clusters.
- Recursive partitioning:
  creates a decision tree that attempts to correctly classify members based on a dichotomous dependent variable.

6.4.3. Machine Learning

Source: http://www.toptal.com/machine-learning/machine-learning-theory-an-introductory-primer, 2016-02-12 16:34

„A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.“ – Tom Mitchell, Carnegie Mellon University

Machine Learning (ML) is used in a wide range of applications, such as language processing, image recognition or data mining, where problems can not be solved by numerical means alone. When distinguishing between different ML types, it is needed to separate between supervised and unsupervised machine learning.

6.4.3.1. Supervised Machine Learning

Supervised Machine Learning relates to programs which are trained with predefined examples. These programs optimise the ability to get an accurate conclusion when given new data. The aim of most supervised learning applications is to develop a finely tuned predictor function. This function $h(x)$ should be able to predict an accurate output based on the given input data points (x). The basic prediction function looks like the following:

$$h(x_1, x_2, \ldots, x_n) = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \ldots + \theta_n \cdot x_n \tag{6.17}$$

The learning process itself is performed by using training examples where the output is known in advance. First, the algorithm calculates the output based on the input and then compares the calculated output to the actual output. Then, the algorithm alters the θ-values and calculates the output again, until the smallest difference (wrongness) is reached. The wrongness measure is known as the cost function $J(\theta)$.

6.4.3.2. Unsupervised Machine Learning

The focus task of unsupervised machine learning is finding relationships within data. Instead of learning from training examples the system is tasked with finding patterns and according correlations.

Approaches for unsupervised learning:

- clustering (e.g., k-means, mixture models, hierarchical clustering)
- expectation–maximisation algorithm
- independent component analysis
- non-negative matrix factorisation

7. Hardware

7.1. Raspberry Pi

A Raspberry Pi is a small computer, based on a single board. It was brought to the market in 2012 with the aim to teach people software and hardware skills easily. Furthermore, it should help developing countries to computerize their everyday life due to the Raspberry Pi's low price.[1]

7.1.1. Hardware

Since the first release of the Raspberry Pi there have been multiple models launched. When purchasing a Raspberry Pi, the following models can be chosen from:

- A-Series:
  - Raspberry Pi 1 Model A
  - Raspberry Pi 1 Model A+
  - Raspberry Pi 3 Model A (to be released)[2]
- B-Series:
  - Raspberry Pi 1 Model B
  - Raspberry Pi 1 Model B+
  - Raspberry Pi 2 Model B
  - Raspberry Pi 3 Model B
- Raspberry Pi Zero
- Raspberry Pi Compute Module

Although all of the models are more or less armed with the same components, the hardware differs in some areas such as performance, connectivity options, size or use case.

[1] https://de.wikipedia.org/wiki/Raspberry_Pi

[2] According to the official Raspberry Pi blog, the Raspberry Pi 3 Model A is expected to be produced during 2016 - https://www.raspberrypi.org/blog/raspberry-pi-3-on-sale/, 2016-03-23 11:54

Processor

Except Raspberry Pi 2 and 3, who have a quad-core processor, all the other models have a single-core processor. The clock speed (without over-clocking) starts at 700 MHz (all Raspberry Pi 1 models) and goes up to 1200 MHz (Raspberry Pi 3).

RAM

The A-models and the first B-Model have a RAM-capacity of 256 MB, the following B-Models and the Raspberry Pi Zero have 512 MB RAM. With 1024MB, the Raspberry Pi 2 and 3 have the highest RAM-capacity of all so far produced models. Although the RAM seems to be pretty small, it is enough for such a tiny computer.

Pins

Every model has a certain number of pins for various purposes. Not only has a Raspberry Pi ground and voltage pins, but also GPIO-pins for general usages. The first Model A and Model B Raspberry Pi have 26 Pins, from which 17 are GPIO-pins (general purpose input/output). All the other models have 40 pins, whereof 26 are GPIO-pins. The only exception is the Compute Module, which aims to be used within a professional surrounding and has 200 pins (DDR2 layout). Furthermore, there are many different additional modules which can be attached to almost every Raspberry Pi to upgrade their hardware.

Network connectivity

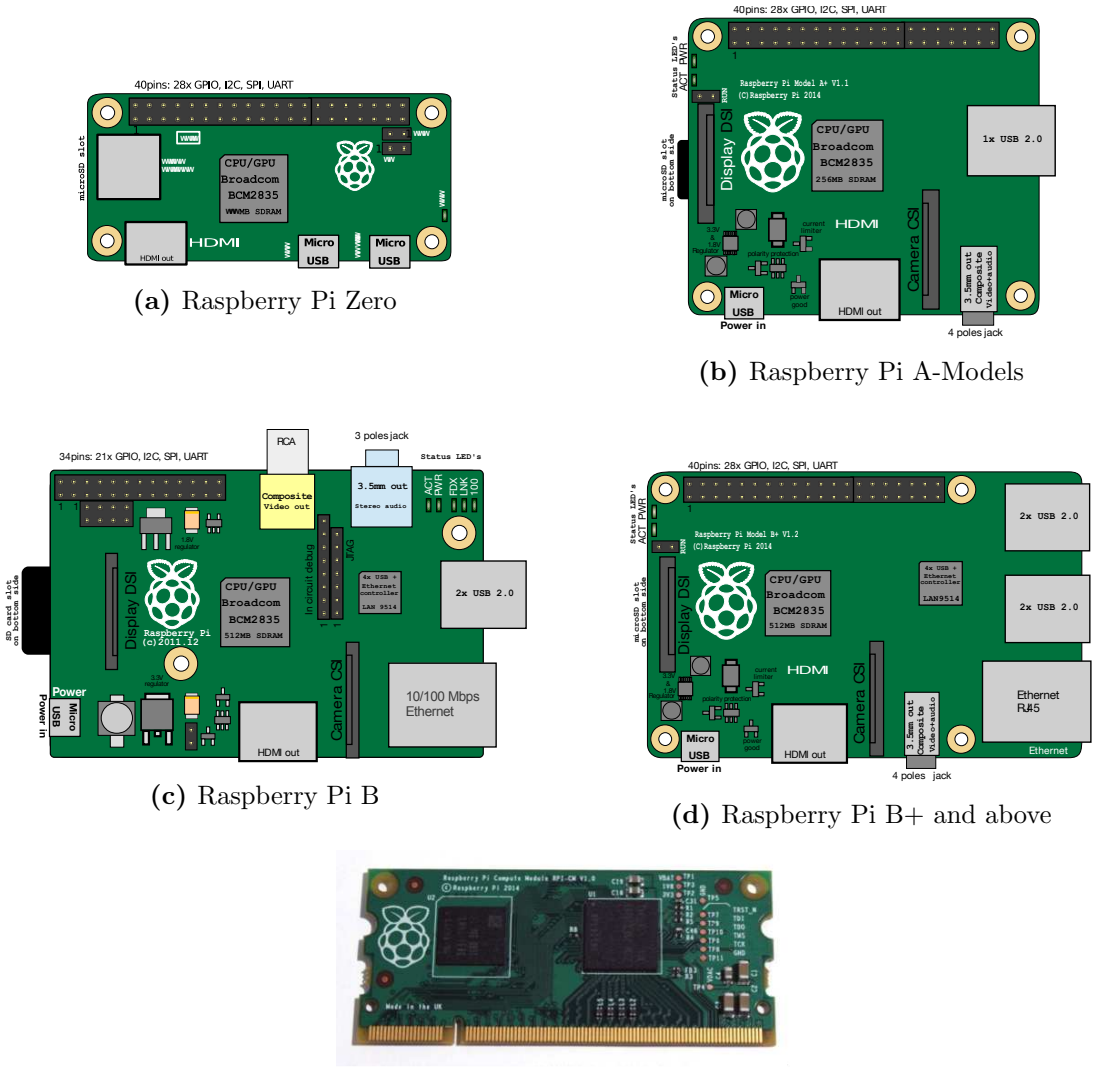
The main difference between model A and B is the network connectivity. Whereas model B has an ethernet port, model A does not support a network connection without using a WLAN stick or any other network components.

Use case

Depending on the use case, different Raspberry Pi models can be applied. When the Raspberry Pi has to be small and modest, the Raspberry Pi Zero can be used. For low power consumption, model A has to be used. Model B represents the allrounder of the Raspberry Pi family and can be used for small projects such as a media server or a weather station. The compute module is used for professional applications and services.

Board layout

The Raspberry Pis do not only differ in hardware characteristics but also in the arrangement of the connectors and ICs on the boards. The following figures show the different layouts used by the corresponding Raspberry Pi Models. Model Zero is the smallest one and has therefore not as many components as the other ones. Contrary to the A-Models, where the layout is the same for all revisions, the B-Models had a layout change between Version B and B+ which arranged the components in a better way.



(a) Raspberry Pi Zero

(b) Raspberry Pi A-Models

(c) Raspberry Pi B

(d) Raspberry Pi B+ and above

(e) Raspberry Pi Compute Module

Figure 7.1.: Layout of the different Raspberry Pi Models

Source: https://en.wikipedia.org/wiki/Raspberry_Pi#Connectors, 2016-03-28 15:59

7.1.2. Software

Nowadays, there are several open-source operating systems available for the Raspberry Pi. The installation can either be carried out by cloning the image of an operating system onto a SD-card or by the NOOBS-installer (new out of the box software). When using the NOOBS-installer, the software has to be downloaded and dragged onto the SD-card. When booting the Raspberry Pi, NOOBS will offer a range of operating systems to choose from. The chosen operating system gets installed automatically.

Raspian

Raspian is a Linux-distribution based on Debian. The image is approximately 3GB in size and has a graphical user interface. After booting the Raspberry Pi, the partition size can be

extended in order to use the full memory space for saving data. As the Raspberry Pi Foundation optimizes the image for every individual Raspberry Pi, it is recommended to always download the images from the official website.

Other operating systems[3]

Apart from the Raspian operating system, Linux offers several operating systems which seem to be running on the Raspberry Pi.

Pidora, a slimmed down version of Fedora, was only developed for the Raspberry Pi. In the beginning of Pidora, is was full of bugs and seemed to be unsuitable from a user's perspective. Although most of the teething troubles have been eliminated, Pidora is still not usable on certain Pi models with a RAM of 256MB.

Arch Linux is, compared to the other operating systems, clearly not for Raspberry beginners. It is very quick in booting but does not have a graphical user interface so the user must work with the command line. Arch Linux is mostly used by advanced users.

[3]Source: http://www.raspberry-pi-geek.de/Magazin/2013/05/Distributionen-fuer-den-Raspberry-Pi-im-Ueberblick/, 2016-03-24 13:40

7.2. Relay[4]

A relay is an electromagnetic switch in order to react to a small control current. The relay works with both electrical and mechanical components. When current starts flowing through a coil, an electromagnetic field is produced in order to pull a contact arm towards the coil. The contact arm closes the main circuit. When the control current stops flowing through the coil, the contact arm backs off the coil and the main circuit is opened.

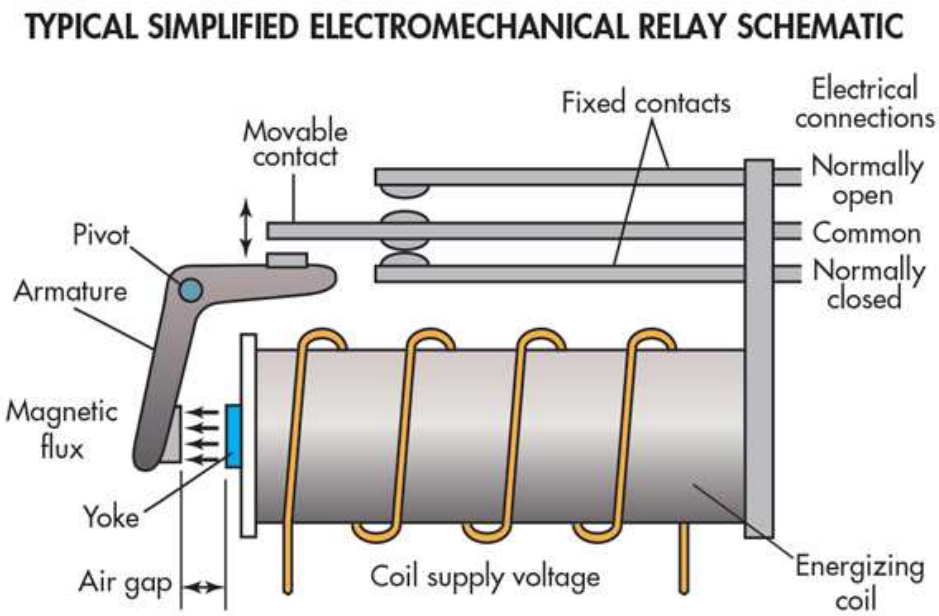


Figure 7.2.: Operating mode of a relay

Source: http://machinedesign.com/site-files/machinedesign.com/files/uploads/2014/07/relay_diagram.gif, 2016-03-24 16:43

As seen in the figure, there are 3 different contacts, between which can be switched:

- Normally Open Contact (NO)
  When the control current flows, the contact is closed in order to close the main circuit. When the relay is inactive, the contact is open.

- Normally Closed Contact (NC)
  The NC-contact represents the opposite of the NO-contact and is closed, when the relay is inactive. The NC-contact is used when the relay should work the other way round. The main circuit is not closed when the relay is active.

- Change-over Contact (CO)
  The change-over contact is used to connect the main circuit with the relay. The main circuit can either be connected between the CO and the NO-contact when the relay should close the main circuit when active or between the CO and the NC-contact to achieve the opposite effect .

[4]http://www.circuitstoday.com/working-of-relays, 2016-03-24 17:46

7.3. Qi[5]

Qi is a newly developed standard to transfer power without using any physical components between the transmitter and the receiver. Both the transmitter and the receiver contain a induction coil. The coil in the transmitter gets powered and creates an electromagnetic field, which can be received by a second induction coil. If the second coil gets close enough, the power will be transferred automatically. After receiving the power over the coil, the energy gets converted into electrical current in order to charge for example a battery. Until now, wireless charging has still some disadvantages in usage.

- Efficiency
  Due to losing most of the energy in heat, the efficiency of Qi wireless charging is about 40%. This results in higher electricity costs per charge and it takes much longer to fully charge a device. A decent cable has an approximate efficiency of 80%.

- Handling
  Whereas a phone can be used while charging by cable, the wireless charging only functions when the phone and the Qi transmitter are in direct contact, which complicates the usage of the phone.

Wireless charging has become more and more popular since big manufacturers like Samsung have built Qi-receivers into their newest smartphones. In addition, some furniture companies have started to produce pieces of furniture which already have Qi-transmitters built-in. Experts are looking forward to developing a wireless charging system where the electromagnetic field is so powerful that users only have to be at home to charge their devices.

[5] http://mentalfloss.com/uk/physics/26648/how-does-wireless-charging-work-and-is-it-actually-safe, 2016-03-26 17:27

8. Android

Figure 8.1.: Android logo and lettering

Source: https://en.wikipedia.org/wiki/Android(operating_system), 2016-02-13 14:00

8.1. What is Android?

Source: https://en.wikipedia.org/wiki/Android(operating_system), 2016-02-13 13:00

Android is an open source operating system for mobile devices, developed by Google Inc. Those devices are mainly smartphones and tablets, however Android also supports smartwatches using "Android wear" (announced March 2014) and cars using "Android auto" (announced June 2014).

It is mainly controlled via touch input, but can also be controlled via physical buttons or Google's voice control feature "Google Now", depending on the hardware it runs on.

As of February 2016, Android has by far the biggest market share amongst mobile operating systems. According to a survey by *Strategy Analytics*, Android had a market share of 84.6% during the second quater of 2014, topping out above Apple's iOS with 11.9% and Microsoft's Windows Phone with 2.7%.[1] The currently latest version of Android is Version 6.0 - "Marshmallow" and was released on October 5, 2015.

[1] Golem - Android läuft auf fast 85 Prozent aller Smartphones: http://www.golem.de/news/mobile-betriebssysteme-android-laeuft-auf-fast-85-prozent-aller-smartphones-1408-108290.html, 2016-02-14

8.2. Interface Elements and Usage

8.2.1. Homescreen

The homescreen is what the user sees when the device boots up or closes applications and is comparable to the desktop of Windows. It can contain the icons of the users' favourite apps and widgets that can display auto updating content like the weather forecast. The homescreen can contain several different pages (switched between by swiping left or right) and can keep the user from going through all the installed applications by providing the possibility to arrange the most used ones as liked.[2]

8.2.2. Taskbar

The taskbar is another defining feature of the Android operating system. It can show important information like battery status or network connectivity and also be used to toggle quick settings such as WIFI on-off, airplane mode or using the inbuilt LED-flash as a flashlight.

8.2.3. Navigation buttons

Android by default has three main buttons at the bottom of the screen: home, back and open-apps-overview. Depending on the manufacturer, these can be either on-screen where they can change place according to the device's orientation, or off-screen (mostly below). Some manufacturers also choose to modify Android, so that e.g.: the off-screen softkey for showing all open apps now opens an "options" dialogue and all open apps can be seen by long pressing the home button.

8.2.4. Menu

In Android, the menu, opened via an icon on the homescreen, is a collection of all installed apps, however, it isn't the go-to-place when wanting to do something. Instead, this is done almost completely by the homescreen (depending on the users' preferences and the manufacturers' integration of the homescreen). The menu is something manufacturers tend to modify very heavily, changing the look and feel of it, as well as adding functions like rearranging apps, having a separate page of favourite apps, allowing the user to uninstall apps directly and a lot more.

8.2.5. Applications

Applications or "apps" are one of the most important and success defining features of mobile operating systems and represent the parallel to pc programs. Virtually everything the user does

[2] Wikipedia - Android: https://de.wikipedia.org/wiki/Android_(Betriebssystem), 2016-02-16

on an Android device happens through an application, whether it is tweaking the settings of the smartphone using the "Settings" application, taking photos, installing new apps, playing games, checking social media sites or something else. Applications are stored on the Android device (contrary to computers, the exact location of the program can not be specified, although some modified versions offer the choice to install apps on the inserted SD card). Once installed, the icon of the app appears in the menu and can be put on the homescreen if requested by the user.

8.2.6. Widgets

Widgets could be best described as "a window inside an application". They vary in size and can be placed on the homescreen. The benefit of using widgets is the instant information of what is going on without the need to actually enter the application. A very commonly used widget for example is an e-mail widget that shows the last received message automatically.

8.2.7. Play Store

The Google Play Store (formerly "Android Market") is the default application to find and install new applications onto the Android device. However, contrary to iOS and Windows Phone[3], Android allows apps to be installed from external, uninvestigated sources if the related option is ticked in the settings application.

[3]Wikipedia - Windows Phone: https://de.wikipedia.org/wiki/Microsoft_Windows_Phone, 2016-02-15

8.3. Android Studio

Android Studio is Google's official integrated development environment and is based on JetBrains' "IntelliJ IDEA" development environment. It is specifically designed to develop Android apps and has replaced eclipse Android development tools as the official IDE for Android since its first stable release in December 2014. It is available for Windows, MacOS, and Linux Users.

8.3.1. Features

Android studio is a solid IDE with some of its key features being:

- Text editor
  Android studio's editor is based on the IntelliJ IDEA Editor and important features such as code completion and syntax highlighting.

- Support for all Android versions
  With Android Studio, the user can code apps for all Android versions and all types of end devices like smartphones, smartwatches and even cars.

- XML Editor
  XML files can be configured either via the graphical UI by dragging elements onto the screen of the, for example, phone, via the standard code editor.

- Monitoring features
  When running a coded application (either on a real device connected via USB or an emulated device), Android Studio can monitor the CPU and network usage of the application and provide helpful graphical feedback to the user. In Android Studio 2.0 which is currently (2016-02-17) in canary (preview) state, a GPU profiler was added, which can be used to investigate rendering outcomes and get a better understanding of possible causes of errors.

- ProGuard
  ProGuard is an open-source software that compresses, optimises and impedes the decompilation of Java bytecode and makes files smaller and more stable when running.[4]

- Gradle
  Gradle is the build automation system Android Studio uses to convert the written Java and XML code into bytecode. It was designed to be able to build big projects and supports incremental building where only changed parts of the code are rebuilt, which helps reducing build times. It also supports custom build logic that can be added via plugins.[5]

[4] Wikipedia - ProGuard: https://de.wikipedia.org/wiki/ProGuard, 2016-02-17

[5] Gradle Plugin User Guide:
https://sites.google.com/a/android.com/tools/tech-docs/new-build-system/user-guide, 2016-02-18

8.4. Coding Android Applications

8.4.1. Differences in comparison to regular Java coding

While Android programs are written in Java, the "Android Java" doesn't function exactly the same as the "regular" one does.

8.4.1.1. Libraries and Methods

While the source code of Android uses the standard Java-syntax, not all libraries available for Java are available for Android as well. Therefore, not everything that can be done in Java using one of the numerous libraries can be done in Android the same way. Apart from this, Android doesn't use a "main()" method that is executed as Java does, but methods like "onCreate()" or "onResume()" that are invoked depending on certain events or the users actions.[6] For more information, see Figure 8.2.

8.4.1.2. Program structure

A typical Java project can consist of various different files with different possible filetypes such as ".java", ".xml" or ".csv". Much more than Java however, Android relies on various different files in order to get to the finished application.

These files include:

- JAVA-Files
  They contain the program logic of the application, define how the app reacts to events/ user inputs, defines the apps' functionality.

- Android Manifest
  It provides essential information such as required permissions, app icon, app theme or launching activity to the Android system and is preconditional for a working app. [7]

- Default resources
  Static resources are used regardless of the device's hardware or if no other resource is available. This includes layout files, animations, drawables, color state lists, etc. [8]

- Alternative resources
  These kind of resources may vary, depending on which hardware the application is run

[6] Wikipedia - Comparison Java and Android:
https://en.wikipedia.org/wiki/Comparison_of_Java_and_Android_API, 2016-02-18

[7] Developers - App Manifest:
https://developer.android.com/guide/topics/manifest/manifest-intro.html, 2016-02-18

[8] Developers - Resource Types:
https://developer.android.com/guide/topics/resources/available-resources.html, 2016-02-18

on. The most important example are different layout files that are chosen according to the screen size of the device. [9]

- Gradle config files
  These files specify things like the target Android version or which build tools/plugins should be used.

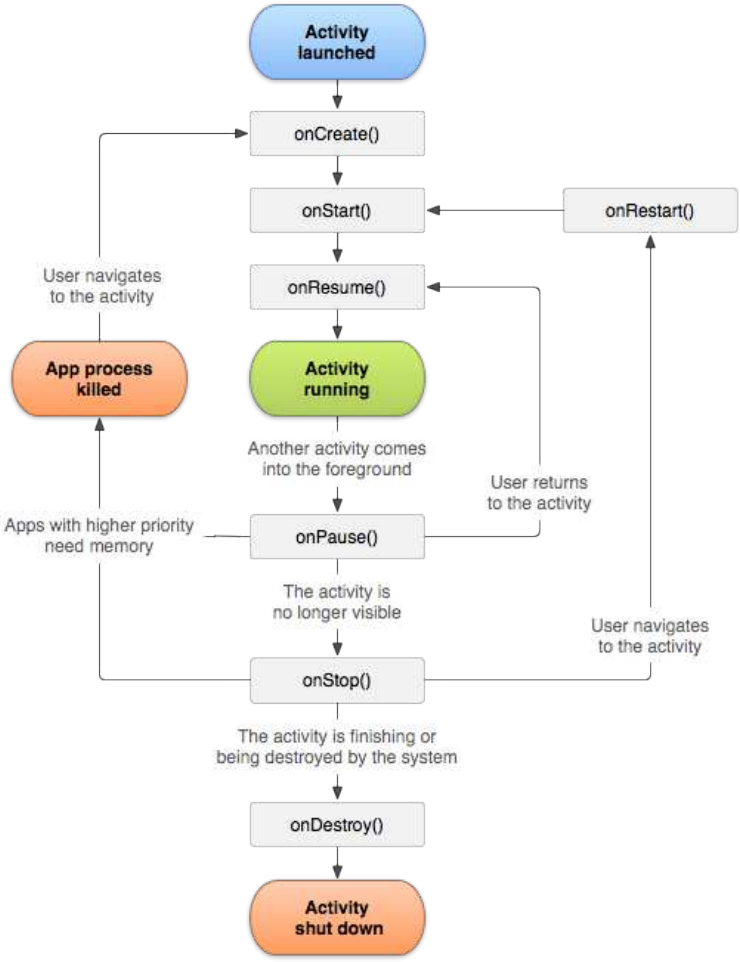


Figure 8.2.: Lifecyle of an Android Activity

Source: http://developer.android.com/reference/android/app/Activity.html

8.4.1.3. ART and Dalvik

ART and Dalvik are virtual machines that are run by the Android os and are responsible for executing applications. These virtual machines are different from the JVM and are the reason for the differences in code such as slightly different syntax, different available libraries and a different way of compiling code and resources.

[9] Developers - Resources:
https://developer.android.com/guide/topics/resources/overview.html, 2016-02-18

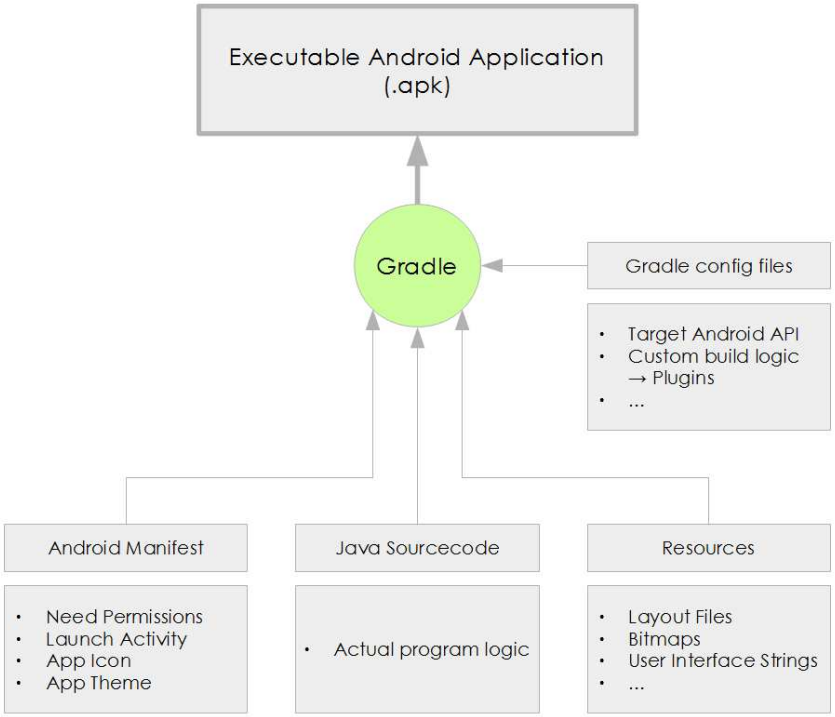


Figure 8.3.: Simplified process of the build ingredients of an Android application

Dalvik

Dalvik is the older option for interpreting code and was introduced together with the first version of Android. It has been Android's runtime environment for applications up until Android 5.0, where it was exchanged for ART.

Dalvik uses Just-In-Time compilation, which means that it converts the .dex files of an application into machine code at runtime and not beforehand. This happens every time an application is invoked. By doing so, installation times can be kept short and the memory usage of applications on the device stays very well manageable.[10]

ART

ART was introduced with Android 4.4 where it was available as an alternative runtime environment, however Dalvik was still used by default. With the update to Android 5.0, Dalvik was completely replaced by ART as the official Android runtime environment. The main difference between ART and Dalvik is that ART uses Ahead-Of-Time compilation, which means that applications are compiled into machine code during the first installation. By doing so, applications do not have to be compiled every time the app is started, ensuring faster startup times of applications and lower CPU consumption at the cost of longer initial installation times and slightly higher storage usage compared to Dalvik. In addition, there where also made im-

[10] Wikipedia - Just in time compilation: https://en.wikipedia.org/wiki/Just-in-time_compilation, 2016-02-21

provements to the Garbage Collector, debugging features and improved detail in exceptions and crash reports.[11]

8.4.1.4. Executing Programs

Java

In Java, the sourcecode is compiled into Java-bytecode. This code represents an instruction set which is the same, no matter which platform the program should run on. The only thing that differs, according to the target platform, is the JVM. It executes the Java-bytecode while being executed by the CPU of the machine it is running on. Therefore, Java applications can run on any platform, provided there's a JVM available for it.[12]

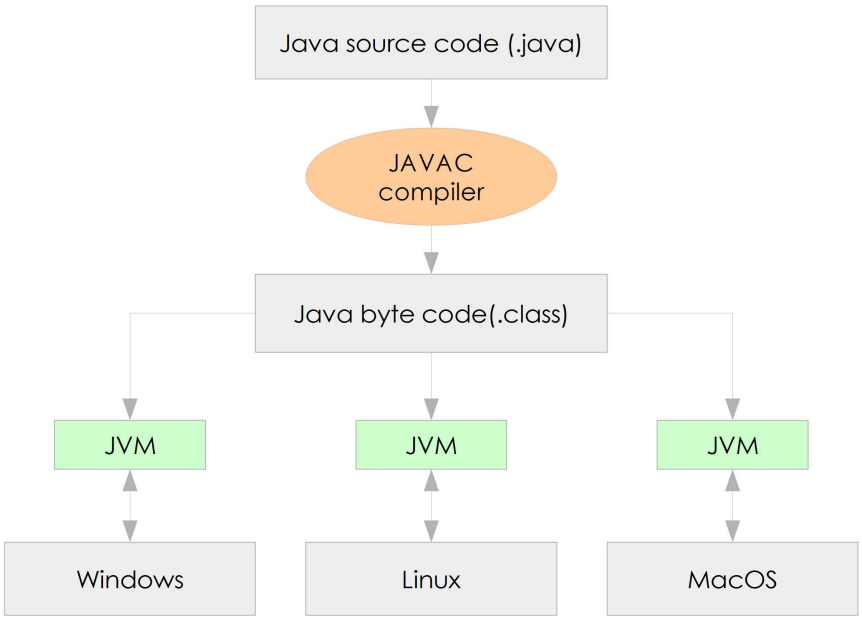


Figure 8.4.: Compiling and execution process of a Java program

Android

When an Android application is to be compiled, all the ".jave" files are compiled into ".class" files. These files are then compiled into Dalvik executables (".dex"). How the code gets compiled (target version, etc.), depends on the gradle settings of the project. The compiled resources (e.g. layout files), the uncompiled resources (e.g. bitmap), the ".dex" files and the Android Manifest are put into one ".apk" file that acts as a container, similar to a ".zip"file.

[11] Android - Art and Dalvik: https://source.android.com/devices/tech/dalvik/, 2016-02-21

[12] Stackoverflow - Java, platform independent, JVM: https://stackoverflow.com/questions/21810538/c-and-java-virtual-machine-code-execution, 2016-02-18

Signing

The last step before being able to use an application is signing it. Signing applies a digital certificate to the application that is needed for its installation process.

Apps can be signed in either debug or release mode, debug mode is used as long as the app is being coded and tested and release mode is being used when the app is ready for publishing.[13]

Debug certificates are created automatically by the Android SDK and use a private key with a known password so the user doesn't have to enter it every time changes are applied to the project.

When the time is ready to officially publish the application, the signed release version can be created, using its own certificate. For this, a keystore and a private key have to be created, which can be done by using Android Studios "Generate Signed APK" Wizard.

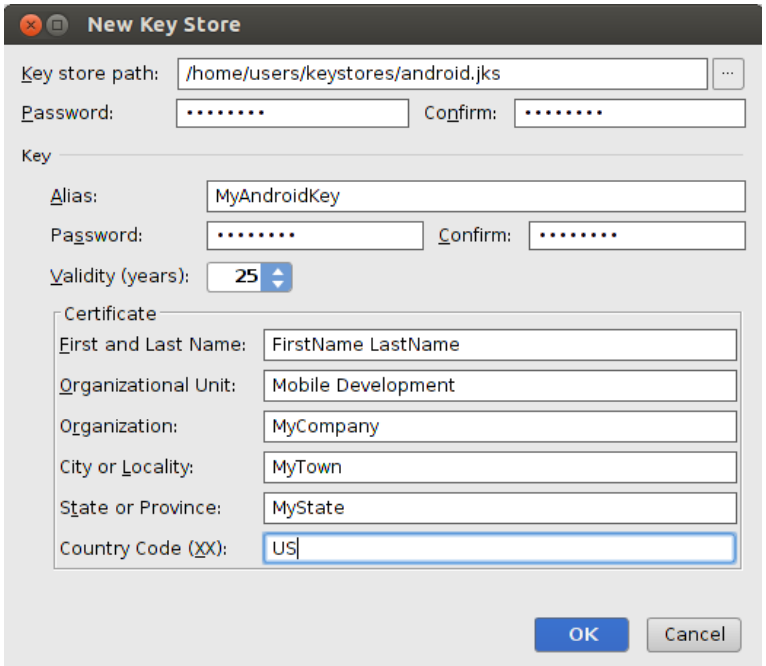


Figure 8.5.: Keystore Windows of the APK signing Wizard of Android Studio
Source: https://developer.android.com/tools/publishing/app-signing.html

The first step is creating a keystore containing multiple private keys (user input as seen in figure 8.5). Then, the keystore and another private key can be selected. Once the correct passwords are entered the application can be signed.

Signing an application is important for the following reasons:

- App upgrade
  To ensure not every user can "update"your application and replace it with their own, the certificates are compared when attempting to do so. The update of an application only works if the certificates are the same.

[13] Developer - Signing your applications: https://developer.android.com/tools/publishing/app-signing.html, 2016-02-19

- Modularity
  In Android, multiple apps can run inside the same process if the certificates match. Therefore, it is possible to just update a single module without touching anything you don't want to change.

- Signature based permissions
  Android offers the possibility of apps exposing functionality to each other, depending on the certificate they use. Therefore it is possible to limit access to functions of an application that can be used by other applications in order to retain sufficient security.

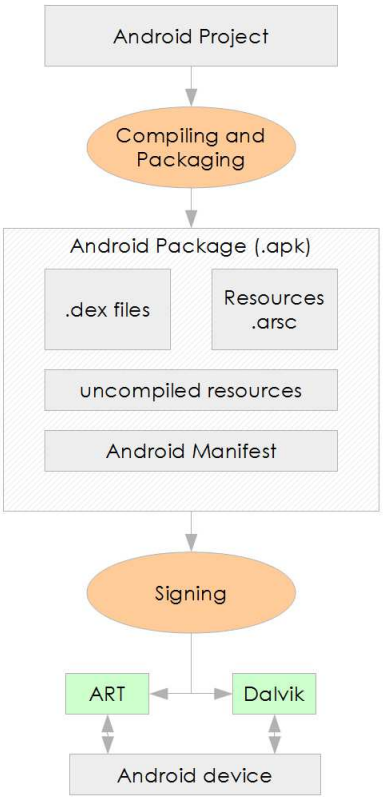


Figure 8.6.: Compiling and execution process of an Android application
Source: https://developer.android.com/tools/building/index.html

ART and Dalvik

Once an application is signed, it can be installed on the desired device. ART and Dalvik are based on the same principle as the JVM and effectively run the application. For more information see section 8.4.1.3.

9. Calculation

A calculation is carried out when a business has to calculate a selling price for a product. Due to market competition and pressure on costs, the calculation of a product has become very important in the last few years.

9.1. Cost-type accounting

The cost-type accounting is the first step of a cost accounting and includes the apportionment of costs.

9.1.1. Classification of cost categories

9.1.1.1. Personnel costs[1]

Personnel costs are caused by employing staff. A business is defined as labour-intensive when a major part of the total costs are caused by personnel costs. A typical labour-intensive business is for example a law firm.
Personnel costs can either be

- wages or salaries,
- social costs prescribed by law or contract or
- other personnel costs (e.g.: severance payments or job advertisements).

Social costs prescribed by law or contract can be, for example, social security payments or holiday pay.

9.1.1.2. Cost of material[2]

Cost of material is caused by usage and further processing of material in a company. The used material has to serve the object of the business. A business is defined as material-intensive when a major part of the total costs is caused by cost of material. Typical material-intensive businesses are found in the production industry. Types of material costs are:

[1] https://de.wikipedia.org/wiki/Personalkosten, 2016-03-26

[2] https://de.wikipedia.org/wiki/Materialkosten, 2016-03-16

- Raw material costs
  Raw material costs can be for example metal in the metal industry or diamonds in the jewellery industry.
- Process material costs
  In certain industries, process material costs can present a major part of the material costs, for example when a business has an excessive consumption of electricity.
- Cleaning agents or packaging material

9.2. Cost centre accounting[3]

A cost centre is the place where costs arise due to the work that has been done. It can either be separated by areas of responsibility or by spatial or accountancy issues. Hence, from the accounting perspective, a cost centre can be seen as a separate department of a business which forms costs. Examples for cost centres would be a purchasing department, a warehouse or a sales department.

A cost centre can either be a main cost centre or a service cost centre. Costs caused by a main cost centre can directly be attributed to various cost objects, the costs of a service cost centre have to be attributed to a main cost centre.**?**

9.2.1. Direct costs[4]

Direct costs are expenses which can be directly attributed to a cost-unit, such as wood being a direct cost of a manufactured item of furniture. Direct costs can also be labour costs which come about when producing a certain product.

9.2.2. Overhead expenses [5]

Overhead expenses have to be apportioned among the produced goods or cost centres. For example, the heating costs of a business cannot be attributed to a certain product but have to be apportioned among the cost centres (mostly per square metre).

[3] https://de.wikipedia.org/wiki/Kostenstelle, 2016-03-17

[4] https://de.wikipedia.org/wiki/Einzelkosten, 2016-03-17

[5] https://de.wikipedia.org/wiki/Gemeinkosten, 2016-03-17

9.3. Cost-unit accouting[6]

9.3.1. Single-stage process costing

When conducting a single-stage process costing, all arising expenses for producing a product get divided by the produced quantity.

$$\frac{Costs}{Unit} = \frac{Arising\ expenses}{Produced\ quantity} \tag{9.1}$$

9.3.1.1. Advantages and disadvantages

On the one hand, single-stage process costing is very practical and simple in use. Unfortunately, it can only be used for businesses which produce one product and practice mass production (no individual or special production).

9.3.1.2. Example

A business has to spend 10000 € a month on producing 1000 pieces of a certain product with one production process. What are the manufacturing costs of one product?

$$\frac{Costs}{Unit} = \frac{10000\ €}{1000} = 10\ €\ per\ unit \tag{9.2}$$

9.3.2. Multistage process costing

A multistage process costing can be conducted when a product passes several production stages to be finished.

9.3.2.1. Example

A steel spring gets produced from raw material to the final product. The product has to pass through several production stages and every workshop department cashes up for their intermediate product. The costs per workshop department get added to the "purchase costs" from the previous department.

9.3.3. Overhead calculation

Usually, products with varying material costs and different direct labour costs are manufactured in different production sequences and with different manufacturing methods. In this case, the

[6] ?

overhead calculation is used. The exact data for calculating can be found in the cost distribution sheet.

9.3.3.1. Single-stage overhead calculation

The overhead costs have to be calculated with an overhead rate from a reference value. The reference value is, depending on the business being labour- or material-intensive, the manufacturing labour costs or the direct material costs. A single-stage overhead calculation should only be used when the overhead costs are only a minor part of the total costs.

9.3.3.2. Double-stage overhead calculation

The double-stage overhead calculation divides the overhead costs into two overhead cost types, the material and factory overheads. The default pattern for the implementation of a double-stage overhead calculation is according to table 9.1.

	material costs
+	material overheads
+	production labour costs
+	factory overheads
=	production costs

Table 9.1.: Double-stage overhead calculation

10. Used tools and technologies

10.1. Java

Official website: http://java.com

Java is a general-purpose programming language, which is able to run on every platform with only compiling the source code once. Java source code is compiled into into an intermediate language called Java bytecode, which is intended to be executed by a virtual machine written especially for the host operating system.

The latest version of the standard edition (SE) is Java 8, however, Java SE 9 is already in development and announced to be released in 2017. Earlier versions are still supported by Oracle and other companies but only on a commercial basis. As of 2016, Java is one of the most popular programming languages in use according to the *TIOBE Programming Community* and *Programming Language Popularity* Index.

10.1.1. JAudio

Official website: http://jaudio.sourceforge.net

JAudio is a Digital Signal Processing project built to provide a program for audio feature extraction. Properties, such as beat points, statistical summaries, MFCCs, SRP, ZCR can be extracted. Additionally, these properties can be used as input for the machine learning program „Weka “.

JAudio can be used as a stand-alone program with a graphical user interface (GUI) or a command line interface (CLI). In addition, a Java library version is also available.

10.1.2. REngine and RServeEngine

Official website: http://rforge.net/Rserve/

These libraries are needed for communication between an R and a Java program. The RServeEngine enables communication between Rserve (section 10.3.3) and Java. The REngine provides the needed classes for representing R objects and includes methods for assigning Java values, converting R variables into Java types, plus having many more features.

10.1.3. JMF MP3 Plugin and JLayer

Websites: http://www.oracle.com/technetwork/java/javase/download-137625.html, http://javazoom.net/javalayer/javalayer.html

The Java Media Framework (JMF) MP3 Plugin and the JLayer library enable handling mp3 encoded files with Java.

10.1.4. MySQL connector

Documentation: http://dev.mysql.com/doc/connector-j/en/

The MySQL connector is a implementation of the *Java Database Connectivity (JDBC) API*, which enables communication between a Java client program and a database. The driver enables executing MySQL commands like inserts, updates or queries.

10.2. Python

Official website: https://www.python.org

Python is a widely used general-purpose, high-level programming language. The aim of the design philosophy is to provide a good code readability and a syntax which allows to express concepts in fewer lines of code compared to other languages, such as C++ or Java. Python interpreters are available on many operating system, allowing code to by executed on various systems. Additionally, Python is the recommended language for developing Raspberry applications due to the easy GPIO access and good support community.

10.2.1. RPi.GPIO

Website: https://pypi.python.org/pypi/RPi.GPIO

The RPi.GPIO package provides a class to control the GPIO on a Raspberry Pi. This package enables reading from and writing to the GPIO pins.

10.3. R

Official website: https://www.r-project.org

R is a programming language and software environment focused on statistical computing and graphics. The main application of R is statistics, data mining and data analysis. R implements a wide variety of statistical and graphical techniques, including linear and nonlinear modeling,

classical statistical tests, time-series analysis, classification, clustering, and many more. R is easily extensible through functions and extensions.

One further, great advantage of R is the vast support for third party packages which enable a wide range of additional applications. All of these packages are maintained and available via a central repository (cran). For analysing sound, the extensions tuneR and seewave can be used.

10.3.1. TuneR

Documentation: https://cran.r-project.org/web/packages/tuneR/index.html

The TuneR package provides a collection of tools for analysing music, extracting features like MFCCs, handling wave files, reading mp3 files and many more options. The provided „Wave "class enables presenting a sound wave within R, which is crucial for working with sounds and analysing them afterwards.

10.3.2. Seewave

Documentation: https://cran.r-project.org/web/packages/seewave/index.html

Seewave is an extended package for sound analysis and synthesis. The package processes time analysis (oscillograms and envelopes), spectral content, cross correlation and autocorrelation, zero-crossing, dominant frequency, 2D and 3D spectrograms and many other analyses. This package depends on the tuneR package (see section 10.3.1).

10.3.3. Rserve

This package allows R to act as a socket server (TCP/IP or local sockets) which allows binary requests to be sent to R. Client-side implementations are available for popular languages. This allows applications to use facilities of R without the need of linking to R code.

10.4. Android

Android is an open source operating system for mobile devices, developed by Google Inc. For further information see chapter 8.

10.5. IDEs

Integrated development environments (IDEs) are software applications which provide comprehensive facilities for programming and software development. Normally, an IDE consists of a

source code editor, build automation tools and a debugger. Therefore, IDEs present a single program in which all programming tasks are done.

10.5.1. Eclipse

Official website: http://www.eclipse.org

Eclipse is a powerful IDE primary focused on developing Java applications, but also suitable for developing with C, C++, PHP, Python, R and many other environments. The Eclipse Software Development Kit (SDK), which includes the Java development tools, is meant for Java developers.

Due to its flexible extension ability, users are able to install plug-ins for almost every task concerning software development. This is also the reason why programming C++, Python, R is easily possible within Eclipse.

10.5.2. RStudio

Official website: https://www.rstudio.com

RStudio is the common IDE for the programming language R. It includes a console, a syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

10.5.3. Android Studio

Official website: https://developer.android.com/

Android Studio, based on IntelliJ IDEA, is the official IDE for developing Android applications. For further informations see section 8.3.

10.6. Sonic Visualiser

Official website: http://www.sonicvisualiser.org

Sonic Visualiser is free software, which enables visualising, analysing and annotating sound files. Looking at audio visualisations, such as spectrogram views with interactive adjustment of display parameters, becomes really easy by using this program. Furthermore, Sonic Visualiser

allows to run feature-extraction plugins to calculate annotations automatically, using algorithms such as beat trackers, pitch detectors and so on.

10.7. Audacity

Official website: http://www.audacityteam.org

Audacity is a free digital audio editor and recording application. Audacity can be used for post-processing of all types of audio by adding effects such as normalization, trimming, and fading in and out.

10.8. FFmpeg

Official website: http://ffmpeg.org

FFmpeg is a free, complete and cross-platform solution to record, convert and stream audio and video.

"FFmpeg is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play pretty much anything that humans and machines have created. It supports the most obscure ancient formats up to the cutting edge. No matter if they were designed by some standards committee, the community or a corporation." [1]

10.9. MySQL

Official website: http://www.mysql.com

MySQL is an open-source relational database management system (RDBMS). MySQL is widely used for client-server database solutions.

10.10. Git

Official website: http://git-scm.com

Git is an open source distributed version control system, used by many developers around the world. Git's popularity is due to the high performance, efficiency and many advanced features.

Version control systems are highly advisable for software collaboration, since they enable a convenient workflow. The key issue of version control systems is keeping track of the history

[1] About FFmpeg - http://ffmpeg.org/about.html, 2016-02-14 15:01

of files. In order to do so, these systems create an entire history, including all changes, of the work being performed. With these informations, stored on an accessible storage for everyone, it becomes easy to work together because you are always dealing with the newest revision and your changes are available for everyone. Therefore, you have no complications working together on the same project. And even if two or more people work on the same file, Git avoids confusion by providing powerful merging tools.

We decided to use Git because it is one of the leading version control systems,which is, in addition, easy to use despite numerous functions.

10.11. Latex

Official website: http://www.latex-project.org

Latex is a typeset system widely used for advanced writing tasks. Due to well structured documents and many useful extensions, i.e for mathematics, it is perfect for writing scientific documents, as well as diploma and Ph.D. theses.

Part IV.

Implementation

11. Audio analysis

11.1. Approach

When it comes to analysing audio signals, a variety of approaches are possible. For our diploma thesis we had to choose the underlying structure for analysing sound events.

11.1.1. Analysis process

When it comes to the analysis itself, we have to specify the order of the operations taken to analyse an incoming sound.

11.1.1.1. Learning sound

The first part of the analysis process is the learning process. A microphone records an input sound, such as a doorbell, a telephone ring or a alarm clock. Afterwards the recorded sound is divided into sub parts, the so-called windows.

Subsequently, the features of these windows are extracted and saved into a persistent file. This file is structured like a table, as comma-separated values (CSV), and contains a column for the number of the window and columns for the according feature values.

Figure 11.1 shows the general process for learning a new sound.

11.1.1.2. Recognising sounds

The second and main part of the analysis is the sound recognition. Hereby, the microphone permanently records the surrounding sound and creates an input audio stream. Whenever a certain amount of samples (2048 or 4096) is received, windows with corresponding length are generated. These windows are sent to the feature extraction module to export the desired information.

After the analysis is completed the results are sent to a buffer which stores the latest X results (e.g. 10). Subsequently, the comparator process is triggered. This process analyses the match rate (in percent) between the buffered values and all previously learned ones. Afterwards the sound is classified, based on the match rate of the different features for all learned sounds.

Figure 11.2 shows the general process for recognising sound events.

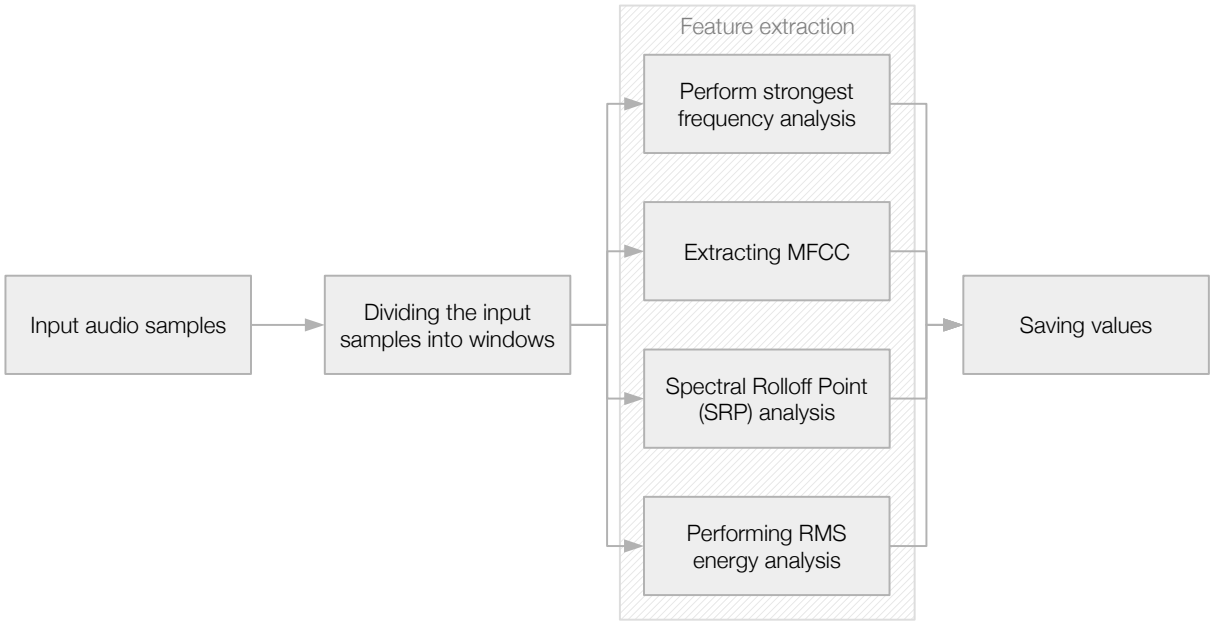


Figure 11.1.: The audio learning process

Source: LibreOffice Draw

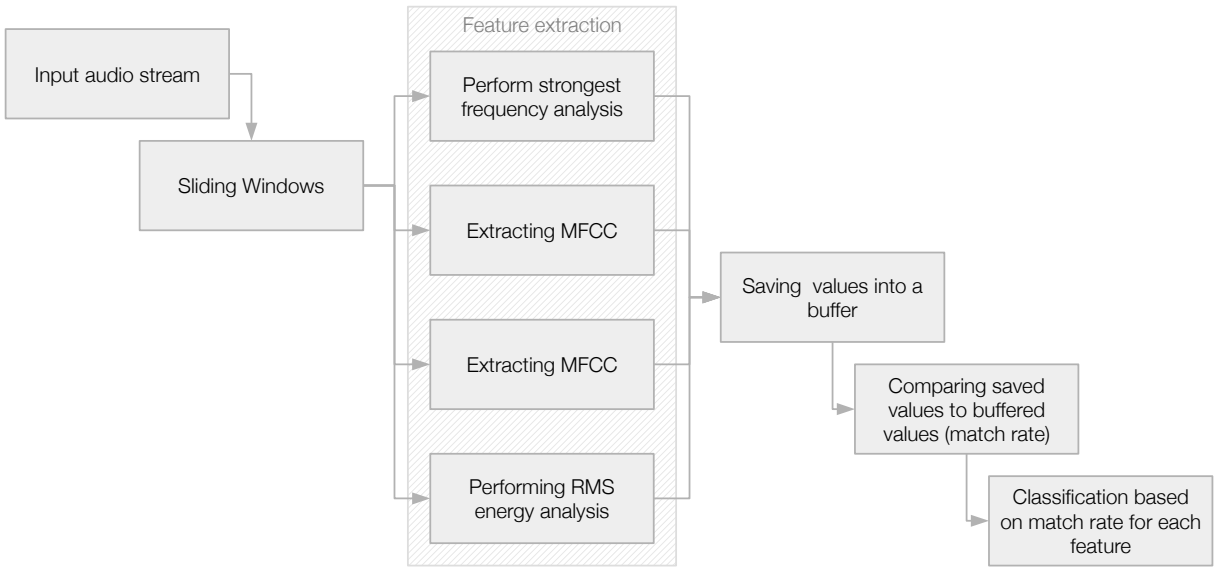


Figure 11.2.: The audio analysis process

Source: LibreOffice Draw

11.1.2. Environment

Since we are using both, a self-built device including a Raspberry Pi and an Android application, we had to develop a software applicable for two targets. In order to keep the effort of the project down to a manageable level, we naturally decided against programming the audio analysis software twice. Thus, we have been aiming to develop a program which makes the analysis possible on both platforms.

The problem thereby is the limitation of technologies when it comes to developing the mobile applications, since Android is restricted to Java and its own virtual machine (VM) called Android Runtime (ART). The ART is a modified Java VM (JVM) with some packages missing from the standard JVM like the `javax.sound.sampled` library, which would be essential for processing sounds.

Therefore, it is necessary to write the major part of the analysis software with Java to avoid implementing some parts twice. In Order to choose the best alternative, we have to consider the approaches listed below.

11.1.2.1. Java based audio analysis

Due to the support of Java on both platforms, one solution is implementing the whole recognition algorithm with Java. Since there is a huge community behind Java, many extensions have been released by independent developers or companies.

The most interesting tools are "JAudio" and "Weka". The first one is an audio feature extraction tool allowing to fetch relevant informations of existing audio files. The second tool (Weka), is a machine learning program which is able to automatically analyse a set of data and decide what information is most relevant. These crystallised information can then be used to make predictions automatically.

The issue hereby is the dependency of JAudio on the package `javax.sound.sampled`, which is missing under Android.

11.1.2.2. Server based audio analysis

When it comes to using other technologies than Java, only the Raspberry Pi is able to use them. Since the Raspberry runs on Linux, many different programming languages and tools are available. Therefore, we could use the Raspberry Pi as server which handles the analysis for the Android application.

By implementing a client server structure, where the Android application sends the recorded audio to the Raspberry Pi the environment dependency problem could be avoided. This would work like the following process: The mobile application records the surrounding sound and streams it to the AudioAnt. After receiving the audio stream, the AudioAnt plate takes further steps and returns the result to the mobile application.

One drawback of a server based implementation is the limitation to a local network. But since the product aims to be an assistant device for homes, this is no issue due to short distances between the smart phone and the device itself.

The second problem could be the bandwidth, when sending the whole audio stream to the analysis server (Raspberry Pi). The oldest smart phones support at least the wireless standard IEEE 802.11g whose available bandwidth is at worst 20 Mbps (The Raspberry supports IEEE 802.11n with up to 90 Mbps). Even if only 1 Mbps is available, a server based analysis would still work, because the maximum data rate of mp3 streams is 320 kbps (CD Quality).

The following sections describe how data can be send from an Android application the AudioAnt Raspberry Pi.

Web service

One way to enable a communication is by using a web service. The term web service describes a standardised way of interaction between a client and a server via a network.

The basic structure consists of two computers, a service requester (in our case the application) and a service provider (Raspberry Pi). In the first step the application calls a service method to which the Raspberry Pi responds after computing the informations. The needed data are typically exchanged with XML/SOAP files and transported by the HTTP(S) protocol.

Socket connection

A network socket enables bidirectional communication between two devices. A socket is also a cross platform technology which is supported by almost every major programming language. Compared to the web service technology, sockets enable a low level connection between two devices.

Communicating sockets are called socket pairs. Every socket pair is uniquely described by the combination of the according IP addresses and port numbers. The IP address refers to the concerned device and the port number to the appropriate application process or thread running on that device. Communicating sockets are able to send all types of data, they are not limited to a certain data type like XML.

Rserve

Rserve is an extension for the R programming language, which allows a socket connection between a running R instance and another software part. It supports remote connection, authentication and file transfer.

Rserve is running on the server. Additionally, a client software runs on the other device. Client-side implementations are available for popular languages such as C/C++, PHP and Java. By providing this functionality, it is easy to enable interaction between the Java application and the recognition algorithm written in R.

11.1.2.3. Chosen environment approach

For our diploma project we chose to implement the main software part in Java, since this allows us to run the software both on Android and directly on the device. Additionally, R and Python were used for tasks hardly possible with Java.

For extracting the wanted audio features the R environment was chosen, since there is no suitable Java library for Android devices. One major advantage of R are the packages TuneR and Seewave, which are really powerful when it comes to feature extraction. The communication between the Java application and R was established with Rserve.

Since controlling the GPIO pins is hardly possible with Java, we made use of Python for this task. Therefore, a Python server module was implemented for controlling the hardware components. The communication between this service and Java was established with a socket connection.

11.2. Feature Extraction

11.2.1. Feature selection

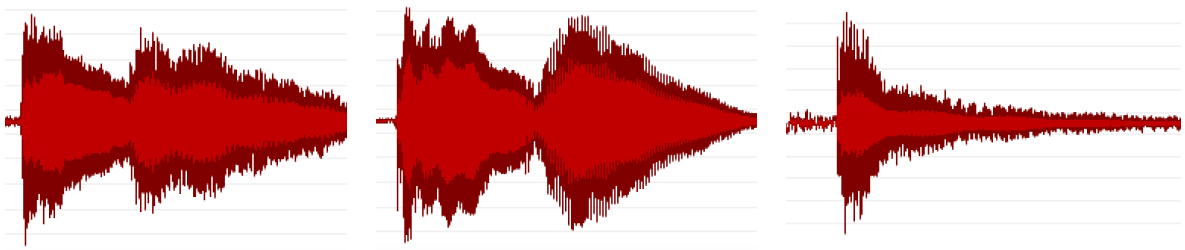
Computers are, unlike humans, not directly able to distinguish between a doorbell and a passing car. In order to enable computers to differentiate between different audio signals, significant features need to be selected. Finding such features was the first step of developing the audio analysis.

11.2.1.1. Visualising the audio signal

The first impression can be made by visualising sample data as waveform, spectrum and spectrogram. These representations reveal some aspects of the wanted sound, helping us to choose relevant features.

Waveform

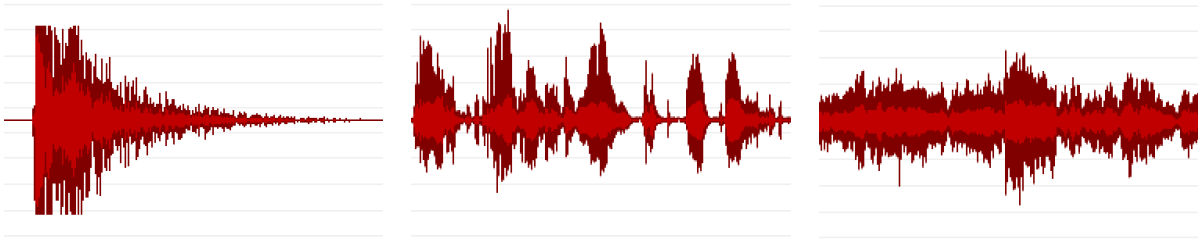
The first presentation looked at was the waveform plot. The following figures show the wanted signal tones compared to other sounds. For giving a short insight, two different doorbells and a microwave alert tone were taken as "true" sounds. As counterpart the visualisation of music, talking and clapping hands was taken.



(a) A doorbell **(b)** Another doorbell **(c)** A microwave

Figure 11.3.: Sounds which should be recognised

Source: Sonic Visualiser



(a) A clap **(b)** Talking **(c)** Music

Figure 11.4.: Sounds which should not be recognised

Source: Sonic Visualiser

The examples for signal tones (figure 11.3), show what encountered us at first sight. Namely, all tested tones are clearly determined by a certain shape. There are only continuously changing forms, no randomly appearing peaks. The most common form is a a triangle like shape, thickest at the beginning of the sound occurrence and thinnest at the end. Many sounds consisted of more than one such triangle in sequence (e.g. 11.3a).

Contrary to that, the examples for "false" sounds (figure 11.4), were not all as clearly shaped as the signal tones. These sounds partially showed random looking peaks and shapes. The only sound which also had a clear shape was produced by clapping.

Hence, it is possible to distinguish between different sounds based on their shapes. In order to determine the shape of a sound, the RMS energy can be used during the feature extraction process, since this feature returns the energy level at the current position. Figure 11.5 visualises this feature.

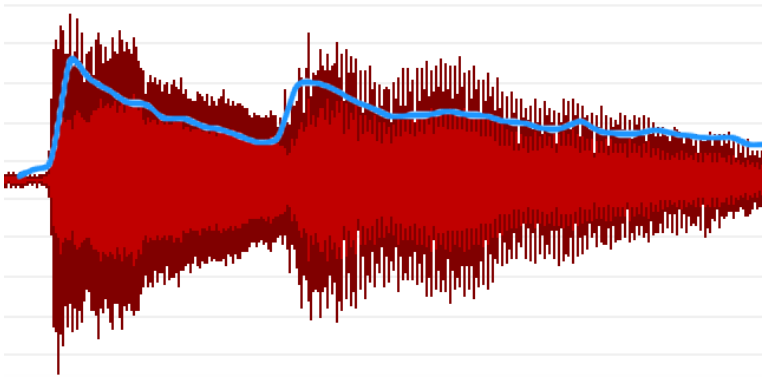


Figure 11.5.: Visualisation of the RMS energy for a doorbell

Source: Sonic Visualiser

Spectrum

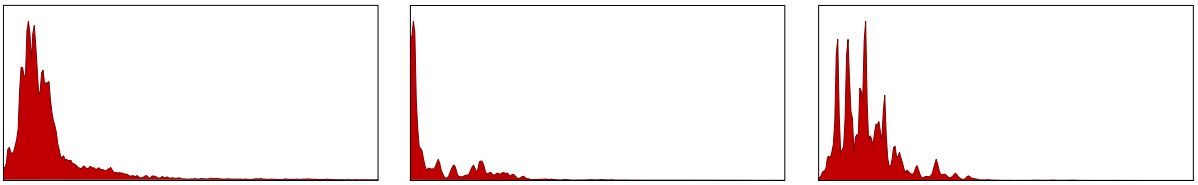
The second presentation we looked at was the spectrum which reveals the partial frequencies. The figures show the same sound files as previously, but this time as frequency based plots. The x-axis is for displaying the time and the y-axis for the frequency.

(a) A doorbell **(b)** Another doorbell **(c)** A microwave

Figure 11.6.: Spectra of sounds which should be recognised

Source: R (with the package seewave)

The spectra of the sample tones (figure 11.6) have clear and sharp peaks, the strongest frequencies are therefore singular occurrences, with almost no side lobes. Hence, it would be possible to recognise sounds by these specific peaks.



(a) A clap **(b)** Talking **(c)** Music

Figure 11.7.: Spectra of sounds which should not be recognised

Source: R (with the package seewave)

Contrary, the spectra of other sounds (figure 11.7) contain peaks not as sharp as the "true" sounds do and have a wider frequency distributions. The only exception was the talking sample, probably because the pitch of one's voice stays in the same frequency area.

The peaks of the spectra are the strongest frequencies, which can be used for distinguishing between sounds. For avoiding saving irrelevant peaks, a threshold can be added, whereby every peak below the line is ignored (figure 11.8).

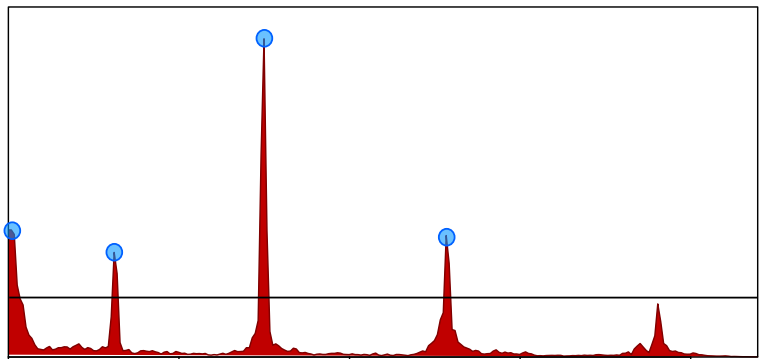


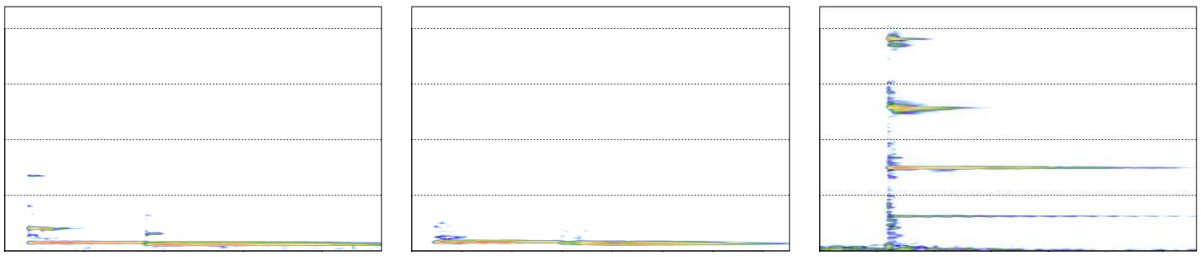
Figure 11.8.: Visualisation of the strongest frequencies of a microwave

Source: R (with the package seewave)

Spectrogram

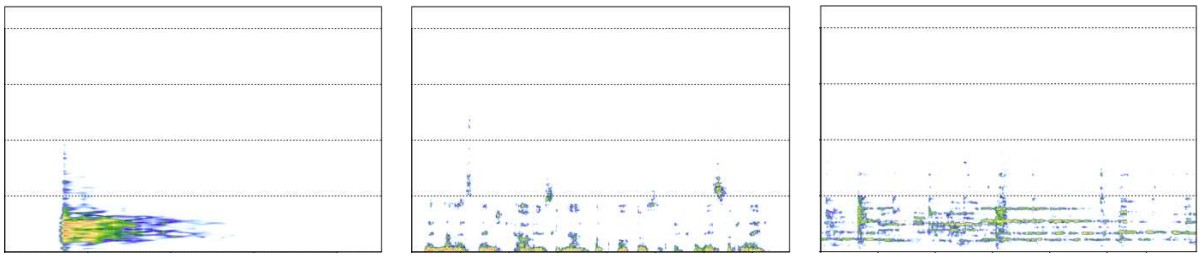
When displaying the same sounds as spectrograms, additional information becomes visible. This presentation reveals the relation between frequency, amplitude and time. Therefore, it is possible to view all factors together. The x-axis shows the time, the y-axis the frequency and the colour the amplitude. The redder the colour, the higher the amplitude.

Figure 11.9 shows the generated spectrograms of the previous examples for signal tones. All plots show at least one clear line from the beginning to the end. Additionally, there are also some shorter lines. Depending on which sound was analysed, the number of different frequencies varies. For instance, the spectrogram of the first doorbell (figure 11.9a) shows one major line, and some shorter ones, and the spectrogram of the microwave (figure 11.9c) shows two major lines, and also some shorter ones.



(a) A doorbell **(b)** Another doorbell **(c)** A microwave

Figure 11.9.: Spectrograms of sounds which should be recognised
Source: R (with the package seewave)



(a) A clap **(b)** Talking **(c)** Music

Figure 11.10.: Spectrograms of sounds which should not be recognised
Source: R (with the package seewave)

Contrary to figure 11.9, the plots of figure 11.10 do not show clear lines from the beginning to the end. For instance, the clapping spectrogram (figure 11.10a) shows a wider band of frequencies, the music spectrogram (figure 11.10c) many discontinuous lines with accumulations at every beat and the talking spectrogram does not show anything that looks like a continuous line.

These plots (figure 11.9 and 11.10) demonstrate the importance of recording the chronological sequence of the learned sounds. Hence, distinguishing between different sounds becomes more accurate when considering time based values.

11.2.1.2. Extracting all possible features

In order to find additional features with a high match rate between similar sounds, we used JAudio and extracted all available features. Afterwards the resulting XML file was converted into a table which shows all features and their values. With a table calculation program, we searched for usable features. The features with the highest match rate were:

- the strongest frequencies via the Zero Crossing Rate and via the Fast Fourier Transform
- the Mel Frequency Cepstral Coefficients (MFCCs)
- the Spectral Rolloff Point (SRP)

The two new features were the MFCCs and the SRP, these are both spectral features. The Spectral Rolloff Point (SRP) is a measurement for the frequency distribution. The SRP is the point below which X% of the magnitude distribution is concentrated. The following plot shows the Rolloff Point of the microwave sample (figure 11.11).

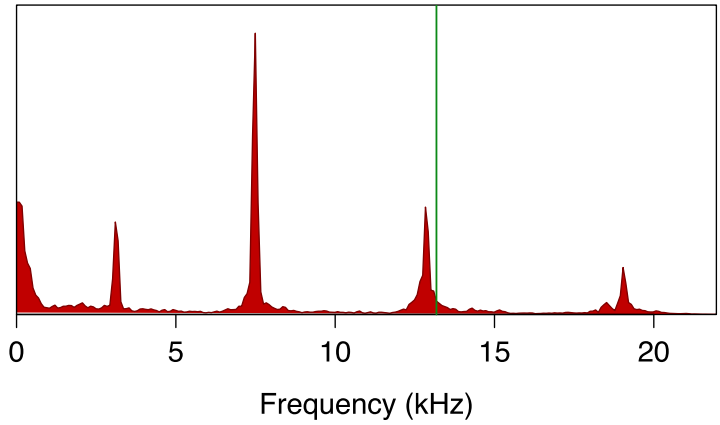


Figure 11.11.: Visualisation of the SRP point for the microwave sample. The green line indicates the Spectral Rolloff Point (X = 0.85).

Source: R (with the package seewave)

The different coefficients, generated with the MFCCs extraction, indicate which bands are the most dominant ones and helps therefore when it comes to classifying sounds. The following figure (11.12) shows the MFCCs for the microwave.

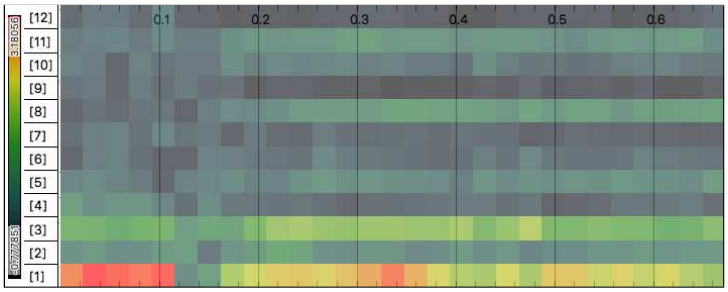


Figure 11.12.: Visualisation of the MFCCs for the microwave sample

Source: Sonic visualiser

11.2.1.3. Result

Altogether the four following features were chosen:

- RMS energy
- strongest frequencies
- Spectral Rolloff Point (SRP)
- Mel Frequency Cepstral Coefficients (MFCCs)

11.2.2. Feature implementation

The extraction for all the previously mentioned features was implemented with R and the packages TuneR and Seewave. The sound object was created by assigning the sample values from a Java array to an R instance of the `Wave` class.

```
public void assignSamples(int[] samples, float sampleRate) throws REngineException {
  rConnection.assign("samples", samples); // Assigning the samples
  rConnection.assign("sampleRate", sampleRate); // Assigning the sample rate
  rConnection.eval("wave <- Wave(samples, samp.rate = sampleRate)"); // Creating a Wave object
}
```

Listing 11.1: Assigning samples to an R Wave

The needed spectrum was generated by calling the method `meanspec()` of the R package Seewave. The resulting spectrum is used by the strongest frequency and SRP extraction.

```
spec <- meanspec(wave, fftw = TRUE, plot = FALSE) # generating the spectrum
```

Listing 11.2: Generating the spectrum with R

11.2.2.1. RMS energy

For calculating the RMS energy, the according built-in R function was used. This function was applied to the sample values. Since we are working with mono audio, only the left channel has to be used for the calculation.

```
rms <- function(wave) {
  result = rms(wave@left) # Calculating the RMS for a mono audio wave
  return(result) # Returning the result
}
```

Listing 11.3: Calculating the RMS energy with R

11.2.2.2. Strongest Frequencies

```
strongestFrequencies <- function(spectrum, threshold) {

  peaksIndex = findPeaks(spectrum[,2]) - 1 # Finding the peaks
  peaks <- spectrum[peaksIndex,] # Storing the peaks
  strongest <- peaks[peaks[,2] >= threshold,] # Storing all peaks above the threshold

  if(length(strongest) > 2) { # Checking, if there are more peaks than one
    strongest <- strongest[order(strongest[,2], decreasing = TRUE),] # Ordering the peaks by
    amplitude
  }
  return(strongest) # returning the strongest peaks
}
```

Listing 11.4: Calculating the strongest frequencies with R

The strongest frequencies were extracted by analysing the spectrum with the `findPeaks()` function of the package `quantmod`. As input for this method, the amplitudes (column 2 of the

spectrum) were used. Since this method returns the index of the first element after a peak (-1) was added to the index. Subsequently, the peaks were stored into a new variable.

For getting the strongest peaks, all peaks below the threshold were cut by storing only the strongest into a new variable. If the result contains more than one value tuple, the frequencies' peaks are ordered by the amplitude. Afterwards the strongest frequencies are returned.

11.2.2.3. Spectral Rolloff Point

For calculating the point, below which the X% of the magnitude distribution is concentrated, writing a function from scratch is required.

First, the sum of all magnitudes is taken and then a loop iterates all values, beginning at the first. At every iteration the current value is added to a variable which stores the current sum. When the current sum reaches X% of the magnitude distribution, the loop ends and the SRP frequency is returned.

```
srp <- function(spectrum, srpBorder = 0.85) {

  amplitudeSum <- sum(spectrum[,2], na.rm = TRUE) # Calculating the sum, leaving out NULL
    values

  current <- 0.0 # Variable, vor the current sum
  index <- 0 # Current index

  while (current <=  srpBorder * amplitudeSum){ # Iterating through the samples until the SRP
    is reached
    index <- index + 1 # Incrementing the index
    current = current + spectrum[index, 2] # Adding the current value to the current sum
  }
  srpFrequency = spectrum[index,1] # Getting the frequency at the SRP index
  srpFrequency = srpFrequency * 1000 # Converting the result from kHz to Hz

  return(srpFrequency) # Returning the result
}
```

Listing 11.5: Calculating the SRP with R

11.2.2.4. MFCCs

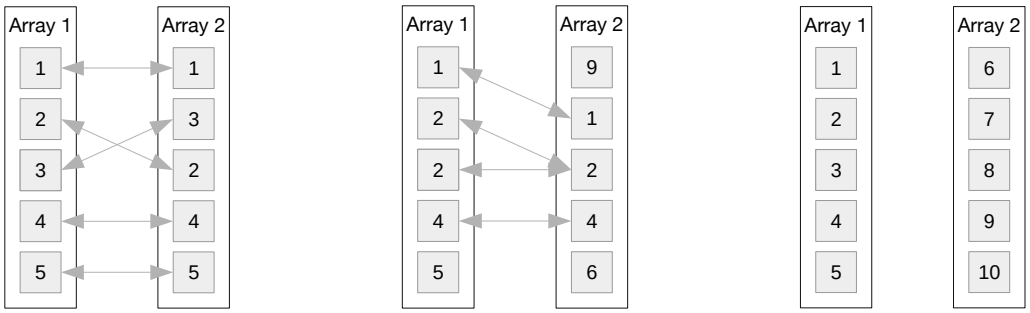
For calculating the MFCCs, the function `melfcc()` is used. This function returns the coefficients 1 to 12 with time based values. Since we want to have a stationary result, we took the means for each coefficient. The coefficients were scaled afterwards to the strongest coefficient, in order to have values, which can be interpreted more easily.

```
mfcc <- function(wave){

  mfcc <- melfcc(wave) # Generating the MFCCs
  mfcc <- colMeans(mfcc) # Calculating the mean for each coefficient
  mfcc <- mfcc / max(mfcc) # Scaling the mfcc realtively to the strongest coefficient

  return(mfcc) # returning the coefficients
}
```

Listing 11.6: Calculating the MFCCs with R

11.3. Calculation of the match rate

After the feature extraction of the input stream, it is necessary to compare the match rate of these values compared to the previously learned ones. In order to do so, these values are compared to each other. When comparing, the algorithm counts the elements of the learned values which are also within the array of recent values. Subsequently, the same procedure is done the other way around. Therefore, the algorithm also counts the number of the recent values within the array of learned values. If all elements are found this way and the other way around, the match rate is 100%, and when none is found, the rate is 0%.



(a) 100% match **(b)** 48% match **(c)** 0% match

Figure 11.13.: Match rates of two simple arrays

```
// Claculation of the match rate
public double getMatch(int[] array1, int array2) {

    // Matching array1 on array2
    double match1 = getPartialMatch(array1, array2);
    // Matching array2 on array1
    double match2 = getPartialMatch(array2, array1);

    // Return the match rate
    return match1 * match2;
  }

// Calculation of one array against another
public double getPartialMatch(int[] array1, int array2){

  // Variable for counting the matches
  int count = 0;

  // Iterating the first array
  for (double v1 : array1) {
    // Iterating the second array for matches
    for (double v2 : array2) {
      // If the element matches -> increase counter and go to the next value
      if (v1 == v2) {
        count++;
        break; // Next value
      }
    }
  }
  // Return the match rate: matching values over possible values
  return (double) (count) / array1.lenght();
}
```

Listing 11.7: Calculating of the match rate with Java

Based on the feature needed to be compared, the comparing algorithm was slightly different in each case. The special implementations are listed below.

11.3.1. RMS Energy

For calculating the RMS Energy match rate, it is important to consider the sequence of the recent elements, since time based values are the crucial factor here for the comparison. With these, time based values, it is possible to recreate the energy shape of the sound wave. Figure 11.14 shows saved RMS values graphically.

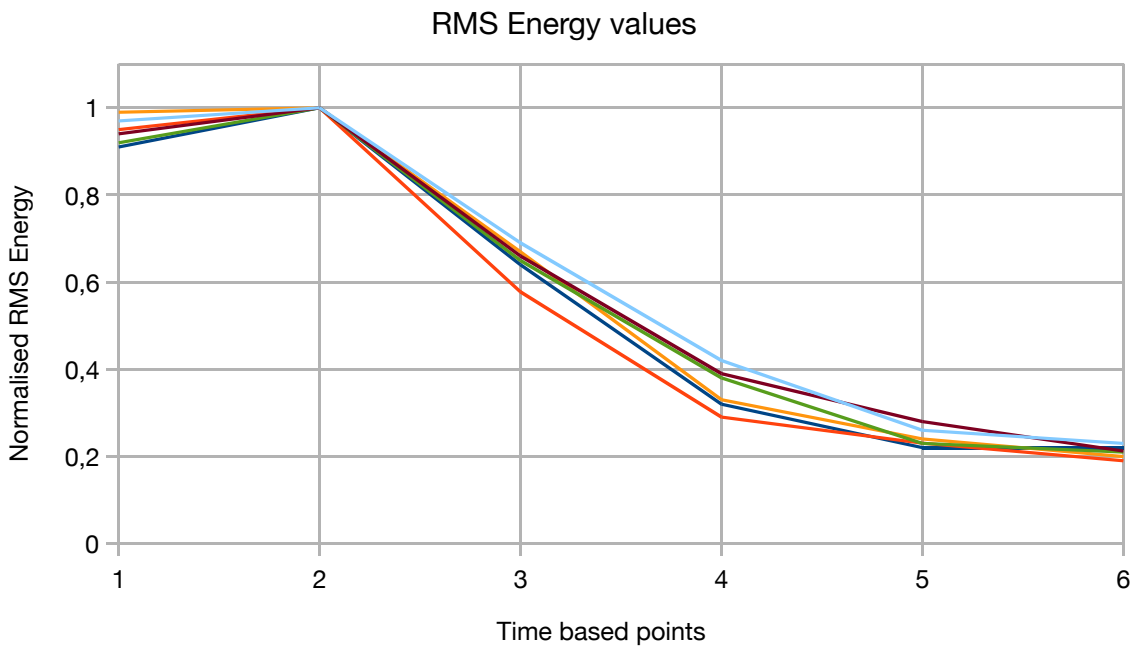
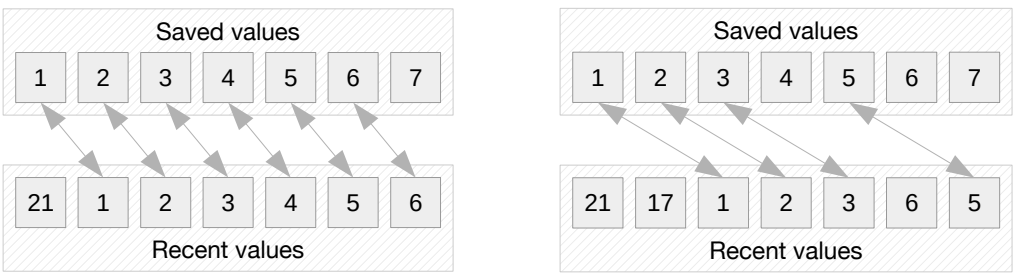


Figure 11.14.: Visualisation of saved RMS Energy values

Additionally, the algorithm for matching arrays has to be altered in a way so that it starts comparing at the first correct value. This is essential to avoid a wrong result, if the recent values have, for example, another element at the first position. Figure 11.15 shows the method which should be used for comparing.



(a) Offset = 1, all values matching

(b) Offset = 2, one position not matching

Figure 11.15.: Matching RMS Energy values

Before comparing, all values have to be normalised, meaning that the RMS values are relatively scaled to strongest amplitude. The learned values were already scaled when they were added, the recent values have to be scaled every time. Code box 11.8 shows the normalisation of the recent values.

```
// Normalising a list of values, all values are scaled depending on the maximum
private List<Double> getNormalisedValues(List<Double> values) {

  // Finding the maximum
  double max = 0; // Variable for saving the current maximum
  for (Double d : values) { // Iterating all values
    // Comparing, if the current value is greater than the current maximum
    if (d > max) {
      // Setting the new maximum
      max = d;
    }
  }

  // Creating a list for saving the normlaised values
  List<Double> valuesNormalised = new ArrayList<Double>();

  // Iterating all values
  for (int i = 0; i < values.size(); i++) {
    // Adding the scaled value to the new list
    valuesNormalised.add((values.get(i) / max));
  }

  // Returning the nomralised values
  return valuesNormalised;
}
```

Listing 11.8: Normalising the RMS Energy values

For getting the right offset, later used when comparing, it is necessary to get the first matching element. This can be done by iterating the recent value and comparing it to the first element of the saved values. Since the values won't exactly match, a match occurs when the difference between the saved and recent value is within a pre-set tolerance margin. Code box 11.9 presents the method for finding the start index.

```
// Method for finding the start index
private int getStartIndex(List<Double> values) {

  // Variable for saving the current index
  int startIndex = 0;
  // Iterating the values
  for (Double d : values) {

    // If the difference between the values is below the border, the element is taken as first
     element
    if (Math.abs(savedValues[0] - d) <= Config.AUDIO_ANALYSIS_ENERGY_TOLERANCE) {
      // Exiting the loop
      break;
    }
    // Increasing the counter
    startIndex++;
  }

  // Returning the start index
  return startIndex;
}
```

Listing 11.9: Getting the start index for the RMS match rate calculation

Finally, the wanted RMS Energy match rate is calculated according to the following code (listing 11.10).

```
// Calculation of the match rate for the energy features
public double getMatch() {

  // Fetch the normlaised values
  List<Double> valuesNormalised = getNormalisedValues(recentValues);

  // Get the start index
  int startIndex = getStartIndex(valuesNormalised);

  // Cut the list, beginning at the start index and ending at the end of the list
  List<Double> sublist = valuesNormalised.subList(startIndex, valuesNormalised.size());

  // Variable for counting the matches
  int count = 0;
  // Iterate the sublist
  for (int i = 0; i < sublist.size(); i++) {
    // If the difference between the values is below the border, the counter will be increased
    if (Math.abs(savedValues[i] - sublist.get(i)) <= Config.AUDIO_ANALYSIS_ENERGY_TOLERANCE) {
      count++;
    }
  }

  // Return the match rate: matching values over possible values
  return (double) count / savedValues.size();
}
```

Listing 11.10: Getting the RMS Energy match rate

11.3.2. Strongest Frequencies

When calculating the match rate of the strongest frequencies, the amplitude of each frequency is additionally considered. This is important, since stronger frequencies (higher peaks) should be weighted more.

For this approach, additional models are needed. The first of those is the `FrequencyModel` (listing 11.11), which is able to store the frequency itself and the corresponding amplitude. For comparing objects of this model the `equals` method was implemented. This method returns, if the frequencies of two models are the same. Because two frequencies are never exactly the same, a threshold (per default 5Hz) was used.

```
// Model for saving frequency / amplitude value pairs
public class FrequencyModel {

  // Member variables
  private double frequency;
  private double amplitude;

  @Override // Overriding the equals method
  public boolean equals(Object obj) {

    // Casting the object to an FrequencyModel
    FrequencyModel model = (FrequencyModel) obj;
    // Returning if the difference is smaller than the threshold
    return Math.abs(frequency - model.frequency) <= Config.AUDIO_ANALYSIS_FREQUENCY_THRESHOLD;
  }
}
```

Listing 11.11: The members and the equals method of the FrequencyModel

Secondly, since one sample consists of more frequency-amplitude pairs, a method for getting the match of one single sample is needed. For this purpose the `StrongestFrequenciesModel` was introduced (listing 11.12). This model saves a list of `FrequencyModel` instances. For calculating the overlap of two samples, an according method was implemented. The return value is the overlap (as percent) of an input model with the current model.

```
// Model for saving a sample which consists of multiple frequency-amplitude pairs
public class StrongestFrequenciesModel {

  // Member variable which holds a list of FrequencyModels
  private List<FrequencyModel> strongestFrequencies;

  // Method for calculating the overlap of this instance with another model
  public double overlap(StrongestFrequenciesModel model) {

    // Getting the overlaps
    double o1 = overlap(this, model);
    double o2 = overlap(model, this);

    // Return the overlap
    return o1 * o2;
  }

  // Method for calculating the overlap of two models
  public double overlap(StrongestFrequenciesModel m1, StrongestFrequenciesModel m2) {

    // Variable for the sum of matching amplitudes
    double match = 0;
    // Variable for the total sum
    double sum = 0;

    // Iterating all elements
    for (FrequencyModel m : m1) {
      // Check, if m2 contains m, this triggers the equals() function of the FrequencyModel
      if (m2.strongestFrequencies.contains(m)) {
        // Add the amplitude the the match sum
        match += m.getAmplitude();
      }
      // Add the current amplitude the the total sum
      sum += m.getAmplitude();
    }
    // Return the overlap: matching sum over total sum
    return match / sum;
  }
}
```

Listing 11.12: The member and the overlap methode of the StrongestFrequenciesModel

For calculating the match rate itself the previously presented overlap function was used to find the sample with the highest overlap within the other array.

```
// Calculating the match rate for the strongest frequencies
public double getMatch() {

  // Getting the match rates
  double m1 = getMatch(recentFreqs, savedFreqs);
  double m2 = getMatch(savedFreqs, recentFreqs)

  // Returning the total match rate
  return m1 * m2;
}
```

Listing 11.13: Calculating the match rate of the frequency values (part 1)

```java
// Comparing one list to another in order to get the match rate
private double getMatch(List<StrongestFrequenciesModel> ms1, List<StrongestFrequenciesModel>
    ms2){

  // Variable adding up the overlap percentages
  double matchCount = 0;

  // Iterating all elements of the first list
  for (StrongestFrequenciesModel m1 : ms1) {

    // Variable for keeping the current max
    double max = 0;
    // Iterating all elements of the second list in order to find the best match
    for (StrongestFrequenciesModel m2 : ms2) {

      // Getting the current overlap
      double tmp = savedFreq.overlap(recentFreq);

      // Checking if the current overlap is bigger than the current max
      if (tmp > max) {
        // Setting a new maximum
        max = tmp;
      }
    }
    // Adding the maximum value to the match count
    matchCount += max;
  }
  // Returning the match rate: matching sum over possible sum
  return matchCount / m1.size();
}
```

Listing 11.14: Calculating the match rate of the frequency values (part 2)

11.3.3. Spectral Rolloff Point

The implementation of the SRP match rate was pretty straightforward. For this calculation the default algorithm has to be altered only slightly. The only thing changed is a tolerance for comparing two values, where the top and bottom border, within which the value has to be, are calculated using a threshold percentage (e.g. ±7%).

```java
// Calculating the match rate for SRP values
public double getMatch() {

  // Getting the match rates
  double m1 = getMatch(savedValues, recentValues);
  double m2 = getMatch(recentValues, savedValues);

  // Returning the total match rate
  return m1 * m2;
}

// Method for evaluating if two srp values match
private boolean isMatch(double d1, double d2) {

  // Calculating the borders with the tolerance (percent)
  double min = d1 * (1 - Config.AUDIO_ANALYSIS_SRP_TOLERANCE);
  double max = d1 * (1 + Config.AUDIO_ANALYSIS_SRP_TOLERANCE);

  // Returning if the value is between the boarders
  return d2 > min && d2 < max;
}
```

Listing 11.15: Calculating the match rate of the SRP values (part 1)

```
// Comparing one list to another in order to get the match rate
private double getMatch(List<Double> l1, List<Double> l2){

  // Variable for counting the matches
  int count = 0;

  // Iterating all elements of the first list
  for (double d1 : l1) {

    // Iterating all elements of the second list in order to find the best match
    for (double d2 : l2) {

      // Checking if the values match
      if (isMatch(d1, d2)) {
        count++; // Incrementing the counter
        break; // Breaking the loop
      }
    }
  }
  // Returning the match rate: matching elements over total elements
  return (double) count / l1.size();
}
```

Listing 11.16: Calculating the match rate of the SRP values (part 2)

11.3.4. MFCCs

For calculating the MFCC match rate, the first thing needed was to find out which coefficients are relevant for comparison and which border should be set. For doing so, a box plot (figure 11.16) was created for revealing the range which values are in.

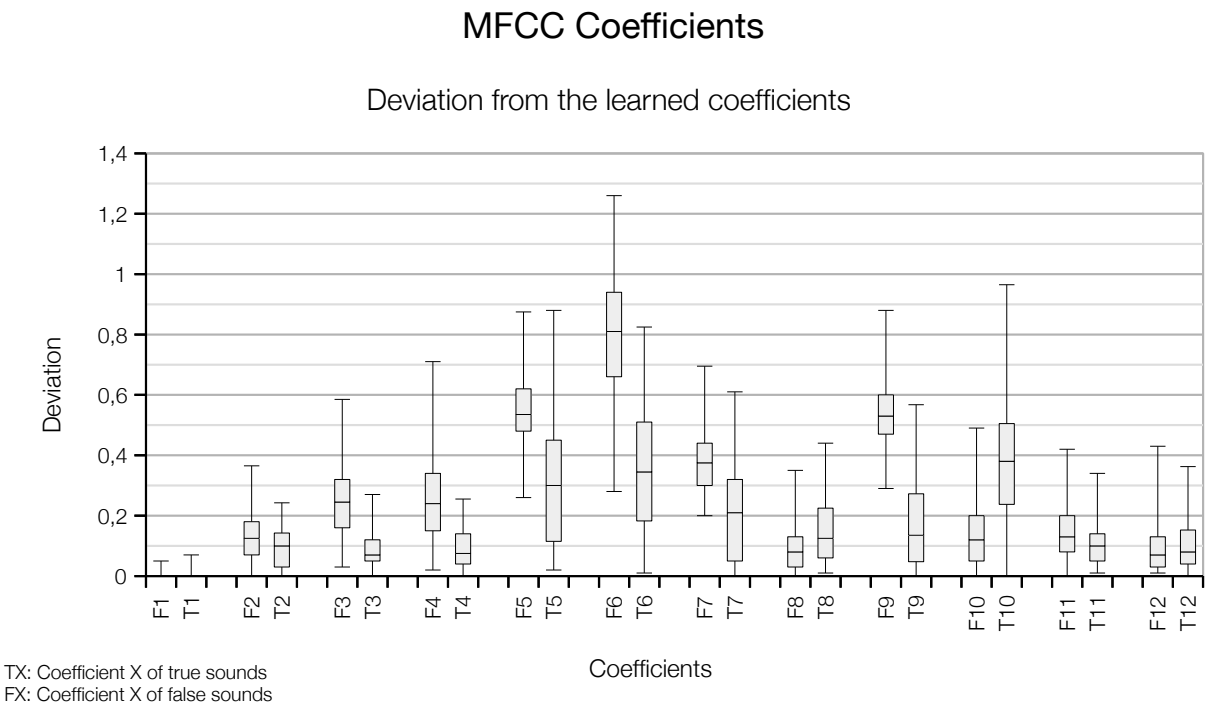


Figure 11.16.: MFCC coefficients: Deviation from the learned values

According to figure 11.16, the best coefficients to distinguish between true and false sounds are 3,4,5,6,7 and 9, since they show a clear difference between a true and false sound because the according boxes do not overlap. The 10th coefficient also does not overlap, but the problem here, is that the deviation from the learned values is higher in case of true sounds, which does not make sense for comparing.

For calculating the match rate, a border is set (figure 11.17). The algorithm then evaluates whether the current deviation is below or above the border. If the value is below, the coefficient is considered as a match. The border was set at the middle of the third quartile of the true sound and the first quartile of the false one.

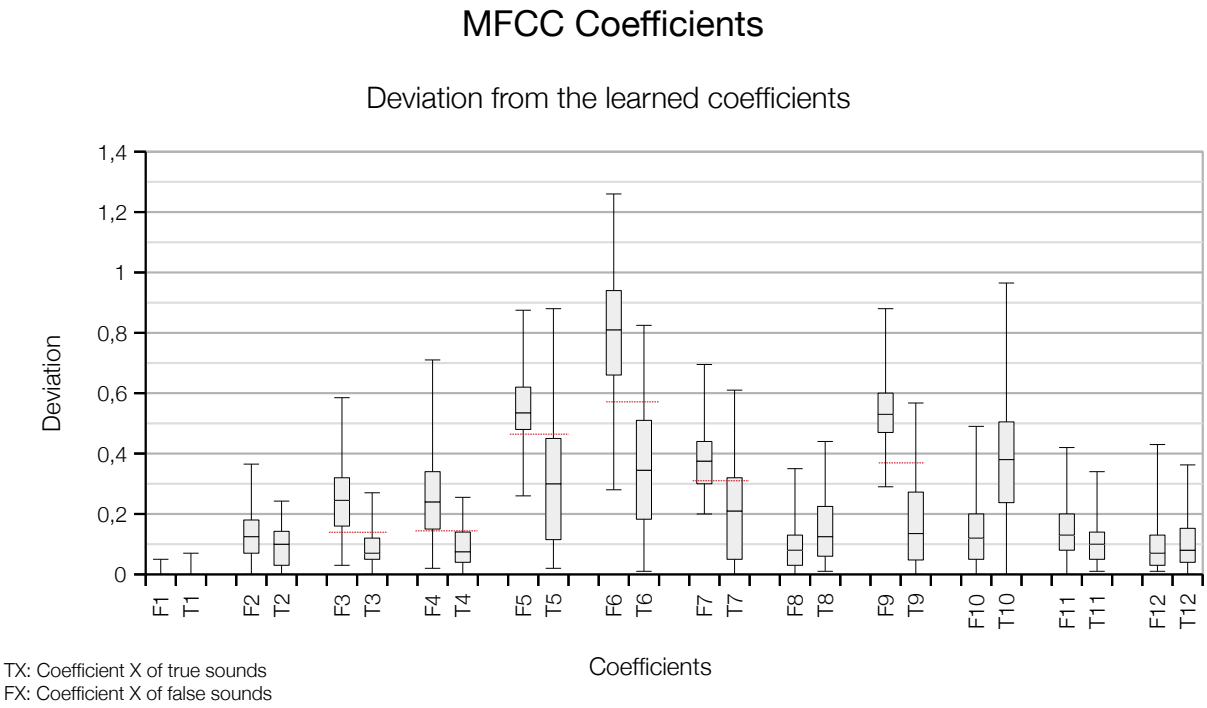


Figure 11.17.: MFCC coefficients: Deviation from the learned values including borders

During testing, it became clear, that the match rate is more accurate if the analysis duration is longer. For getting the MFCC coefficients for a longer time span, the means of all saved and recent values were calculated and used for comparison afterwards. The means for the saved values were calculated directly when starting the program. In contrast, the means for all current values were calculated each time anew.

```
// Calculating the match rate for MFCC values
public double getMatch() {

  // Getting the recent means
  double[] recentMeans = getRecentMeans(recentValues);

  // Return the result of the getMatch function
  return getMatch(savedMeans, recentMeans);
}
```

Listing 11.17: Calculating the match rate of the MFCC values (part 1)

```java
// Method for evaluating if two mfcc arrays match
private double getMatch(double[] d1, double[] d2) {

  // Variable for counting the matches
  int count = 0;
  int countTotal = 0;

  // Loop through all values of the first array
  for (int i = 0; i < d1.length; i++) {

    // Check, if the current coefficient should be considered
    if (Config.AUDIO_ANALYSIS_MFCC_COEFFICIENT_USED[i]) {
      // Checking if the deviation is smaller than the preset border for this coefficient
      if (Math.abs(d1[i] - d2[i]) <= Config.AUDIO_ANALYSIS_MFCC_TOLERANCE[i]) {
        // Incrementing the counter
        count++;
      }
      // Incrementing the counter
      countTotal++;
    }
  }
  // Returning the match rate: matching coefficients over total amount of coefficients
  return (double) countTrue / d1.length;
}

// Method for calculating the recent means
public double[] getRecentMeans(List<Double[]> recentValues){

  // Creating a new array for all the recent values
  double[] recentMeans = new double[recentValues.get(0).length];

  // Loop through all arrays of the list
  for (int j = 0; j < recentValues.get(0).length; j++) {

    // Temporary variable for saving the current sum
    double sum = 0;

    // Iterate through all recent values
    for (Double[] d : recentValues) {

        // Adding the value of the current coefficient to the sum
        sum += d[j];
    }

    // Calculating and saving the mean value
    recentMeans[j] = sum / recentValues.size();
  }

  // Return the current means
  return recentMeans;
}
```

Listing 11.18: Calculating the match rate of the MFCC values (part 2)

11.4. Classification

Now the values from the match rate calculation have to be processed. For this reason, the classification deals with interpreting the calculated match rates in order to know whether the numbers mean that a sound occurred or not.

The used methods for evaluating if a match occurred were the threshold comparison and the multivariate linear discriminant analysis. All in all, three functions were implemented: one for evaluating if the minimum threshold of every feature is reached, one for evaluating if the weighted sum is high enough and one for a higher threshold. The two threshold functions used the direct comparison, and the weigthed sum method used the discriminant analysis.

11.4.1. Minimum thresholds reached

The minimum threshold method considers the match rate of every feature and only returns true, if all of them are above the specified threshold. For getting the pre-set threshold values, the analysis was started once when only background noises occurred and once when the a true signal was triggered permanently.

The match rate of all features was in both cases printed to the console and imported into a table calculation tool afterwards. Subsequently, the data table was searched for the lowest match rate when a sound occurred and the highest one when none did. These points were used to set the minimum threshold.

If the maximum of false sounds was higher than the minimum of true ones, the value of true sounds was taken, because it is better to consider some sounds as true, even if they are not, since this function should only evaluate if the minimum is reached. The other way around would not make much sense.

The following method (listing 11.19) was used to return whether the minimum threshold was reached or not. The `Config.XXX` values link to the global configuration file (listing 11.20) where the thresholds are stored. The configuration file is used for all types of values needed for the whole tool.

```
// This function returns whether the minimum threshold is reached or not
public boolean isMinThresholdReached() {
  // Return if all match rates are at least as high as the according preset threshold
  return frequnecyAnalyser.getMatch() >= Config.AUDIO_ANALYSIS_MATCH_THRESHOLD_MIN_FREQUENCY
      && mfccAnalyser.getMatch() >= Config.AUDIO_ANALYSIS_MATCH_THRESHOLD_MIN_MFCC
      && srpAnalyser.getMatch() >= Config.AUDIO_ANALYSIS_MATCH_THRESHOLD_MIN_SRP
      && energyAnalyser.getMatch() >= Config.AUDIO_ANALYSIS_MATCH_THRESHOLD_MIN_ENERGY;
}
```

Listing 11.19: Function for returning if the minimum threshold is reached

```
# Detection thresholds (values as percent)
# Minimum
audioAnalysisMatchThresholdMinSrp = .11
audioAnalysisMatchThresholdMinMfcc = .5
audioAnalysisMatchThresholdMinEnergy = .2
audioAnalysisMatchThresholdMinFrequency = .36
```

Listing 11.20: Global properties: Detection threshold section

11.4.2. All thresholds reached

Contrary to the previous method, this function takes the values as threshold, where a wrong detection is unlikely. For getting these thresholds, the same data sheet as before was used, only this time values which are above the maximum match rate of false sounds were taken.

Therefore, this method makes sure that hardly any detections are made where no true sound was. Nonetheless, this leaves the problem of overhearing sounds, since the thresholds are too strict for letting unclear sounds pass.

The following listings show the implemented method for returning whether all thresholds were reached or not (listing 11.21) and the corresponding part of the property file (listing 11.22).

```
// This function returns whether all threshold are reached or not
public boolean isMatchByThresholds() {
    // Return if all match rates are at least as high as the according preset threshold
    return frequnecyAnalyser.getMatch() >= Config.AUDIO_ANALYSIS_MATCH_THRESHOLD_FREQUENCY
        && mfccAnalyser.getMatch() >= Config.AUDIO_ANALYSIS_MATCH_THRESHOLD_MFCC
        && srpAnalyser.getMatch() >= Config.AUDIO_ANALYSIS_MATCH_THRESHOLD_SRP
        && energyAnalyser.getMatch() >= Config.AUDIO_ANALYSIS_MATCH_THRESHOLD_ENERGY;
  }
```

Listing 11.21: Function for returning if all thresholds are reached

```
# Detection thresholds (values as percent)
# Sure
audioAnalysisMatchThresholdSrp = .11
audioAnalysisMatchThresholdMfcc = .63
audioAnalysisMatchThresholdEnergy = .2
audioAnalysisMatchThresholdFrequency = .61
```

Listing 11.22: Global properties: Detection threshold section

11.4.3. Weighted sum

The problem with matching all thresholds is that when one value is slightly below one threshold, the whole term becomes false, even if the others are significantly over the needed values. For avoiding such a thing, the weighted sum function was added. This function multiplies all match rates with a pre-set weight and adds them up. Afterwards the function checks if the sum is above a certain percentage value.

For getting the weighting factors, two approaches are possible. The first one is feeding the previous data sheet into R and calculating the linear discriminate analysis (LDA). For doing so the `klaR` package provides the helpful method `lda()` which can be used for calculating the LDA coefficients. The `klaR` package depends on the `MASS` package.

```
> # Load the required packages
> library(MASS)
> library(klaR)
>
> # Import of the data sheet
> matchRates <- read.csv("~/Downloads/export.csv", sep=";")
```

Listing 11.23: Calculation of the LDA coefficients with R (part 1)

```
> # Print the values
> matchRates
Group        Freq.         SRP  MFCC     Energy
1       0 2.390264e-03 0.080000000 0.625 0.28000000
2       0 0.000000e+00 0.000000000 0.500 0.04000000
3       0 5.590442e-04 0.920000000 0.750 0.04000000
4       0 2.727636e-04 0.020000000 0.750 0.36000000
5       0 2.384246e-03 0.440000000 0.750 0.36000000
6       0 1.316165e-03 0.262857143 0.625 0.04000000
7       1 1.000000e+00 1.000000000 1.000 1.00000000
8       0 1.290497e-03 0.960000000 0.750 0.20000000
9       0 0.000000e+00 0.858666667 0.750 0.04000000
10      0 4.885011e-04 0.920000000 0.875 0.04000000
11      0 5.803761e-05 0.002666667 0.625 0.08000000
12      1 1.000000e+00 1.000000000 1.000 1.00000000
13      0 1.240209e-03 1.000000000 0.750 0.42105263
14      1 1.000000e+00 1.000000000 1.000 1.00000000
15      0 4.609500e-01 0.027777778 0.750 0.08333333
16      1 1.000000e+00 1.000000000 1.000 1.00000000
[output omitted]

> # Perform a lda analysis
> lda = lda(Group ~ Freq. + SRP + MFCC + Energy, data = g5[1:5])
> # Print the result
> lda
Call:
lda(Group ~ Freq. + SRP + MFCC + Energy, data = matchRates[1:5])

Prior probabilities of groups:
         0          1
0.91666667 0.08333333

Group means:
      Freq.       SRP      MFCC    Energy
0 0.1352334 0.4262426 0.7964876 0.3330025
1 0.9638661 0.9879565 1.0000000 0.9424514

Coefficients of linear discriminants:
              LD1
Freq.   3.4750118
SRP    -0.1338518
MFCC   -0.3820242
Energy  2.8579571
```

Listing 11.24: Calculation of the LDA coefficients with R (part 2)

The problem by doing so was that the resulting coefficients are not very practical. A short test revealed that the algorithm makes many wrong detections when these parameters are set. Therefore, we used the second possibility: getting the coefficients directly from the data sheet.

By doing so, the first step was to calculate the means of all true and false sounds as well as the standard deviation of the sample. Afterwards the borders, in which 67% of all values are were calculated. The weighting factor was set according to the difference between the boarders of the true and false sound ranges: the greater the difference, the higher the importance of the feature and thus the stronger weighted. The following listing shows the resulting factors.

```
# Detection weights
audioAnalysisMatchWeightSrp = .17
audioAnalysisMatchWeightMfcc = .22
audioAnalysisMatchWeightEnergy = .06
audioAnalysisMatchWeightFrequency = .55
audioAnalysisMatchWeightThreshold = .42
```

Listing 11.25: Global properties: Detection weights section

Now, since the coefficients are calculated, a function can be created which returns whether a match occurred or not according to the weighted sum. Code box 11.26 shows the implementation.

```
// Calculating the weighted sum
public double getWeightedSum() {
  // Sum up and return the weighted values
  return frequnecyAnalyser.getMatch() * Config.AUDIO_ANALYSIS_MATCH_WEIGHT_FREQUENCY
      + srpAnalyser.getMatch() * Config.AUDIO_ANALYSIS_MATCH_WEIGHT_SRP
      + mfccAnalyser.getMatch() * Config.AUDIO_ANALYSIS_MATCH_WEIGHT_MFCC
      + energyAnalyser.getMatch() * Config.AUDIO_ANALYSIS_MATCH_WEIGHT_ENERGY;
}

// Function for evaluating if a match occured according to the weighted sum
public boolean isMatchByWeightedSum() {
  // Return if the sum is above the specified threshold
  return getWeightedSum() >= Config.AUDIO_ANALYSIS_MATCH_WEIGHT_THRESHOLD;
}
```

Listing 11.26: Function returning if the weighted sum is reached

11.4.4. Match occurred

For evaluating if a match occurred, the previous methods were combined and the two following functions implemented (listing 11.27). Depending on what the user wants, either the `isMatch()` or `isSureMatch()` is used. The first method returns true if the minimum threshold is reached as well as the weighted sum and the second one returns true if all criteria are fulfilled.

```
// Returns if a match occurred
public boolean isMatch() {
  // Returns if the minumum threshold is fulfilled and a match by weighted sum occurred
  return isMinThresholdReached() && isMatchByWeightedSum();
}

// Returns if a sure match occurred
public boolean isSureMatch() {
  // Returns if the minumum threshold is fulfilled and a match by weighted sum and thresholds
    occurred
  return isMinThresholdReached() && isMatchByWeightedSum() && isMatchByThresholds();
}
```

Listing 11.27: Functions for returning if a match occured

11.5. Testing

11.5.1. Automated testing

For evaluating if and how well the recognition algorithm works, a automated testing suite was implemented. This program takes a folder with many sounds, consisting of one example for learning and multiple true and false sounds, as input.

The first step of the program is learning the examples in order to analyse the following samples. After running the comparison algorithm on a sound file, the result is compared with the actual kind (true/false) of the file. The name of the file reveals if the sound should be considered as true sample or not. In order to make this automated program work, the files have to be strictly named as following:

- Reference files (examples) have to match the following pattern: *00*.(wav | mp3)*
- Files of true examples have to match with *true*.(wav | mp3)*.
- The name of other files does not matter, it only has to be different to the previous patterns.

The result of the automated test is saved to a data base which stores all relevant values for analysing the result. The database scheme itself is presented below (figure 11.18).

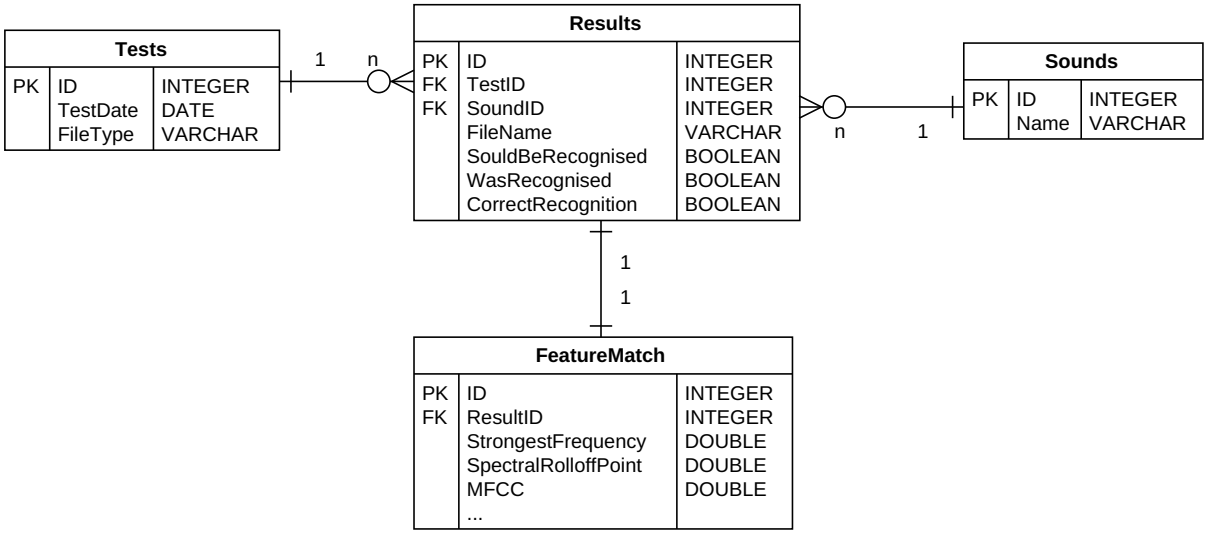


Figure 11.18.: The database schema used for saving test values

The following queries enable fetching a detailed result for the performed test. Beside the obvious information like the success rate, the number of files and the date of the test, the results also show detailed information about every single file and the according match rates. The following listings show the needed queries.

The first query (listing 11.28) questions the database for the summary of all performed tests, which was always the first table looked at after a new test run.

```sql
-- Show test summaries
SET sql_mode = '';
SELECT
  t.id,
  t.testDate as 'Date',
  t.fileType,
  COUNT(DISTINCT r.soundId) as 'Number of sounds',
  COUNT(*) / COUNT(DISTINCT r.soundId) as 'Files p. Sound',
  AVG(r.correctRecognition) as 'Avg. correctness'
FROM
  Results as r,
  Tests as t
WHERE
  t.id = r.testId
GROUP BY
  Date;
```

Listing 11.28: mySQl query for printing the summary of all performed tests

The result of this query could then look like the following example. The table reveals the important information at one glance.

```
+----+---------------------+----------+------------------+----------------+------------------+
| id | Date                | fileType | Number of sounds | Files p. Sound | Avg. correctness |
+----+---------------------+----------+------------------+----------------+------------------+
[output omitted]
| 71 | 2016-01-05 18:06:16 | wav      |               12 |        12.0000 |           0.8819 |
| 72 | 2016-01-05 18:06:45 | mp3      |               12 |        12.0000 |           0.9236 |
| 73 | 2016-01-05 18:11:39 | wav      |               12 |        12.0000 |           0.9931 |
| 74 | 2016-01-05 18:14:49 | mp3      |               12 |        12.0000 |           0.9861 |
| 75 | 2016-01-12 15:29:34 | wav      |               12 |        12.0000 |           0.9167 |
| 76 | 2016-01-12 15:38:50 | wav      |               12 |        12.0000 |           0.9792 |
| 77 | 2016-01-12 16:09:30 | wav      |                1 |         1.0000 |           0.0000 |
| 78 | 2016-01-12 16:18:36 | wav      |               11 |        12.0000 |           0.9848 |
| 80 | 2016-01-18 08:24:33 | wav      |               11 |        12.0000 |           0.9773 |
| 81 | 2016-01-18 08:49:59 | wav      |               11 |        12.0000 |           0.9697 |
+----+---------------------+----------+------------------+----------------+------------------+
73 rows in set (0,06 sec)
```

Listing 11.29: Result of the summary query

For exporting all important information to a comma-separated file, which result can be used for further actions with table calculating programs, the following query can be used.

```sql
-- CSV export
SELECT
  r.shouldBeRecognised,
  f.strongestFrequency,
  f.spectralRolloffPoint,
  f.mfcc,
  f.energy
FROM
  FeatureMatch as f,
  Results as r,
  Tests as t
WHERE
  t.id = r.testId AND
  f.resultId = r.id AND
  t.id = 119
INTO
  OUTFILE '/tmp/databaseExport.csv'
  FIELDS ENCLOSED BY '"' TERMINATED BY ',' ESCAPED BY '"'
  LINES TERMINATED BY '\r\n';
```

Listing 11.30: mySQl query for printing the summary of all performed tests

The automated test helped a lot when implementing new features or altering existing ones, since an immediate feedback was available. If something new did not work out as imagined the summary query (listing 11.28) showed an accordingly lower average correctness and if a change resulted in a better analysis the percentage of correct recognitions went up.

The automated test showed that the average correctness of the final implementation was around 98%, which is pretty good. A query questioning what types of mistakes were made revealed that there were only false positive mistakes, which means that a tiny amount of unwanted sounds was regarded as wanted. This case would be the ideal in reality, since it is better to alert users one time too often instead of not at all.

11.5.2. Short term test

The short term test was a live test, for running the actual program instead of a test tool. This test was executed on the AudioAnt prototype itself, where every step from recording a new sound to altering the user was tested. The main focus of this test was to check if the whole program works, especially if recording and recognising sounds works. For getting detailed information about the recognition process, a logger printed all relevant events into a detailed log file.

```
2016-03-08 15:38:08:430 FINEST:
*  Sound: 1 match  by Sum: true  by Thresh.: false  sum: 1,38 f: 0,39 e: 0,25 m: 1,00 s: 0,75
*  Sound: 4 match  by Sum: true  by Thresh.: false  sum: 1,32 f: 0,41 e: 0,25 m: 1,00 s: 0,56
*  Sound: 6 match  by Sum: true  by Thresh.: false  sum: 1,36 f: 0,38 e: 0,25 m: 1,00 s: 0,75

2016-03-08 15:38:08:578 FINEST:
** Sound: 1 sure   by Sum: true  by Thresh.: true   sum: 1,93 f: 0,71 e: 0,50 m: 1,00 s: 1,00
*  Sound: 2 match  by Sum: true  by Thresh.: false  sum: 1,58 f: 0,60 e: 0,20 m: 1,00 s: 0,60
*  Sound: 3 match  by Sum: true  by Thresh.: false  sum: 1,58 f: 0,60 e: 0,20 m: 1,00 s: 0,60
** Sound: 4 sure   by Sum: true  by Thresh.: true   sum: 2,01 f: 0,74 e: 0,75 m: 1,00 s: 1,00
** Sound: 5 sure   by Sum: true  by Thresh.: true   sum: 1,85 f: 0,65 e: 0,50 m: 1,00 s: 1,00
** Sound: 6 sure   by Sum: true  by Thresh.: true   sum: 1,89 f: 0,68 e: 0,50 m: 1,00 s: 1,00

2016-03-08 15:38:21:518 FINEST:
*  Sound: 2 match  by Sum: true  by Thresh.: false  sum: 1,21 f: 0,67 e: 0,20 m: 0,50 s: 0,12
*  Sound: 3 match  by Sum: true  by Thresh.: false  sum: 1,44 f: 0,68 e: 0,20 m: 0,50 s: 0,64

2016-03-08 15:38:25:671 FINEST:
*  Sound: 2 match  by Sum: true  by Thresh.: false  sum: 1,00 f: 0,40 e: 0,20 m: 0,50 s: 0,48
   Sound: 3 min    by Sum: false by Thresh.: false  sum: 0,90 f: 0,42 e: 0,20 m: 0,50 s: 0,16
```

Listing 11.31: Outtake of the detailed log file

The short term test showed that some adjustments need to be made regarding the detection threshold parameters, since some sounds were not recognised at first. After tweaking the parameters of the global configuration file, the software worked out quiet well. Only an estimated amount of one in ten sounds was overheard. This level of correctness is below the level of the automated test because random background noises occur when analysing a real surrounding instead of audio files. Nevertheless, the detection level is still pretty high and sufficient for most cases.

Even background noises to a certain degree were no problem for the analysing algorithm, since the most important features are more ore less resistant to noise. The only problem encountered was when the background noises grew louder than the sound source which should be recognised. This was for example the case when hoovering occurred nearby.

11.5.3. Long term test

Contrary to the short term test, the long term test tried to spot wrong detections, meaning that a sound is recognised even if there was no actual sound. For this type of test the AudioAnt was running for about 48 hours and logging all sound occurrences into a short log file as well as into the detailed one. After the analysis had run long enough, the short log file was looked at in order to find out wrong detections.

```
2016-02-11 19:23:49:482 INFO: 2 match(es): 1, 6
2016-02-11 20:23:53:832 INFO: 2 match(es): 2, 4
2016-02-11 21:24:21:863 INFO: 1 match(es): 5
2016-02-11 21:24:22:089 INFO: 1 match(es): 5
2016-02-11 22:34:25:303 INFO: 1 match(es): 4
2016-02-11 23:58:32:092 INFO: 2 match(es): 2, 4
2016-02-12 00:58:20:724 INFO: 1 match(es): 1
2016-02-12 01:53:13:361 INFO: 1 match(es): 5
2016-02-12 04:07:22:219 INFO: 1 match(es): 5
2016-02-12 06:51:07:145 INFO: 1 match(es): 4
2016-02-12 08:29:07:406 INFO: 3 match(es): 1, 4, 6
2016-02-12 08:29:07:626 INFO: 2 match(es): 2, 4
2016-02-12 10:22:41:446 INFO: 1 match(es): 1
2016-02-12 10:22:41:659 INFO: 1 match(es): 1
```

Listing 11.32: Outtake of the detailed log file

According to the log file, some detections, where no sound signals were triggered, were made during the long term test. Such detections occurred approximately every one to two hours in irregular intervals. For the productive use, this behaviour needs to be changed, since it is too much being disturbed every two hours at least once.

12. Prototype

12.1. Planning

The aim of the project was to build a functioning prototype of the AudioAnt. In the planning phase of the hardware was important to find out what components with which specifications are needed, where the components should be arranged and how they should be wired.

12.1.1. Power supply

Due to the fact that the AudioAnt contains more power demanding components, a suitable power supply has to be used in order to provide every component with enough power.

- Raspberry Pi: between 0.7 and 2 ampere at 5 volt
- Qi-transmitter and Speaker: up to 2 ampere at 5 volt
- Light source: about 2 ampere at 5V for about 1200lm

Resulting from the list, the power supply has to provide 8 ampere at 5 volt.

12.1.2. Benefit analysis

To find out which microcontroller would be the best for our use, a benefit analysis was conducted. The benefit analysis revealed that the best microcontroller for our project would be the Raspberry Pi 2. Note: The Raspberry Pi 3 had not been launched at this time.

Criteria	Points	Percent	RPi B+		RPi 2B		Arduino		Banana Pi	
	(0-10)	(P/∑P)	R	R*G	R	R*G	R	R*G	R	R*G
Price	2	6%	2	0,12	3	0,18	1	0,06	3	0,18
Size	8	24%	2	0,47	2	0,47	1	0,24	4	0,94
Compute power	5	15%	3	0,44	1	0,15	4	0,59	1	0,15
Functions	9	26%	2	0,53	2	0,53	4	1,06	1	0,26
Programmer friendly	10	29%	1	0,29	1	0,29	4	1,18	3	0,88
Sum	34	100%	2	1,85	1	1,62	4	3,13	3	2,41

Table 12.1.: Benefit analysis

12.2. Components

12.2.1. Raspberry Pi 2

The Raspberry Pi 2 represents the core of the AudioAnt and manages the interaction of software and hardware.

Quantity: 1
Purchased at: Amazon (http://amzn.to/1Rpcblr)

Figure 12.1.: Raspberry Pi 2

Source: https://www.raspberrypi.org/wp-content/uploads/2015/01/Pi2ModB1GB_-comp.jpeg, 2016-03-27 13:22

12.2.2. Casing

Quantity: 1
Purchased at: Amazon (http://amzn.to/1ZIIYEa)

Figure 12.2.: Casing of the AudioAnt

Source: http://ecx.images-amazon.com/images/I/41UUa2C-M9L._SX355_.jpg, 2016-03-27 13:46

12.2.3. Qi-transmitter

We decided to build in a Qi-transmitter to provide users with wireless charging of their smartphones or other Qi-capable devices.

Quantity: 1
Purchased at: Amazon (http://amzn.to/1USSxAG)

Figure 12.3.: Qi-transmitter

Source: http://ecx.images-amazon.com/images/I/61mB%2BD0gloL._SL1024_.jpg, 2016-03-27 13:51

12.2.4. Display

The display is supposed to inform the user about the occurred sounds and shows information about the configuration. When no sounds occur, the display shows the time.

Quantity: 1
Purchased at: Ebay (http://ebay.eu/1qevkwu)

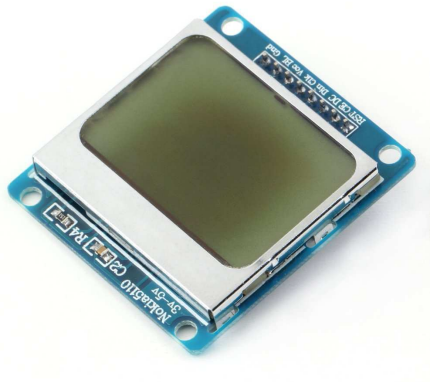


Figure 12.4.: Display

Source: http://ecx.images-amazon.com/images/I/61MFZYZnclL._SL1010_.jpg, 2016-03-27 13:57

12.2.5. Relay

The relay connects the light to the main circuit when a sound is recognized to inform the user about the occurred sound.

Quantity: 1
Purchased at: Conrad (http://bit.ly/1LXfJLL)

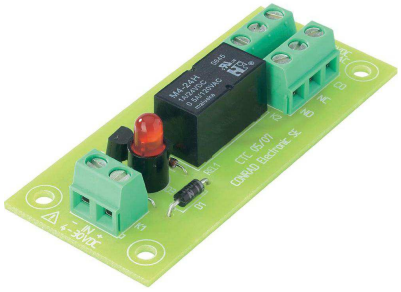


Figure 12.5.: Single-channel relay

Source: https://www.conrad.at/medias/global/ce/5000_5999/5000/5030/5033/503321_LB_00_FB.EPS_1000.jpg, 2016-03-27 14:01

12.2.6. Soundcard

Due to troubles with the arrangement of the components, a soundcard is needed to connect the microphone and the speaker with the Raspberry Pi.

Quantity: 1
Purchased at: Amazon (http://amzn.to/1RpcMng)

Figure 12.6.: Soundcard

Source: http://ecx.images-amazon.com/images/I/81fkTEz7cRL._SL1500_.jpg, 2016-03-27 14:03

12.2.7. Speaker

Quantity: 1
Purchased at: Amazon (http://amzn.to/22LdBOD)
The speaker was purchased in 2010.

Figure 12.7.: Speaker

Source: http://ecx.images-amazon.com/images/I/71xvQlZZMcL._SL1500_.jpg, 2016-03-27 14:10

12.2.8. Microphone

Quantity: 1
Purchased at: Conrad (http://bit.ly/1RFcEPw)

Figure 12.8.: Microphone

Source: https://www.conrad.at/medias/global/ce/8000_8999/8800/8810/8812/881258_BB_00_FB.EPS_1000.jpg, 2016-03-27 14:17

12.2.9. WLAN-stick

Due to the fact that the Raspberry Pi 2 does not have a WLAN module built-in, a WLAN-stick is needed to establish a connection to a network.

Quantity: 1
Purchased at: Amazon (http://amzn.to/21Rutxf)

Figure 12.9.: WLAN stick

Source: http://ecx.images-amazon.com/images/I/71P8Pfd7npL._SL1500_.jpg, 2016-03-27 14:20

12.2.10. Light source

The light source optically alerts the user when a recognzied sound occurs.
Quantity: 1
Purchased at: Amazon (http://amzn.to/1Su9Uoc)

Figure 12.10.: Powerful light source

Source: http://ecx.images-amazon.com/images/I/41D3RJ16PRL.jpg, 2016-03-27 14:24

12.2.11. USB-AUX connection

The USB-AUX connection represents the USB and AUX ports of the Raspberry Pi.
Quantity: 2
Purchased at: Amazon (http://amzn.to/1V1hjh1)

Figure 12.11.: USB and AUX connector

Source: http://ecx.images-amazon.com/images/I/71r1umWn4FL._SL1500_.jpg, 2016-03-27 14:25

12.2.12. Jack socket

Quantity: 1
Purchased at: Amazon (http://amzn.to/1ZIL3A2)

Figure 12.12.: Jack for the power supply

Source: http://ecx.images-amazon.com/images/I/31Kcs52OlqL.jpg, 2016-03-27 14:27

12.2.13. Power supply

Quantity: 1
Purchased at: Amazon (http://amzn.to/21RuTUn)

Figure 12.13.: Power supply

Source: http://ecx.images-amazon.com/images/I/514UQoJF7%2BL.jpg, 2016-03-27 14:31

12.2.14. Sundries

- Shrink tubing and wires
  Shrink tubing is a plastic tube in order to isolate, protect and stiffen wire connections. After slipping the tube over a soldered wire connection, it gets heated up until it is fully shrunk.

- Resistors
  The resistors were used as pre-resistors for the LEDs.

- Buttons
  The buttons were used to provide simple handling to the user.

- LEDs
  The LEDs of the AudioAnt inform the user about the current status of the device.

12.3. Design drawing

12.3.1. Case

The holes on the left side of the case are used for the USB-AUX connection, whereas the big hole and the rectangular cut on the top are used for the Qi-transmitter and the display. There is also a small hole on the top for the confirm button. The small hole on the right is used for the jack socket and the other small holes are used for LEDs and buttons.

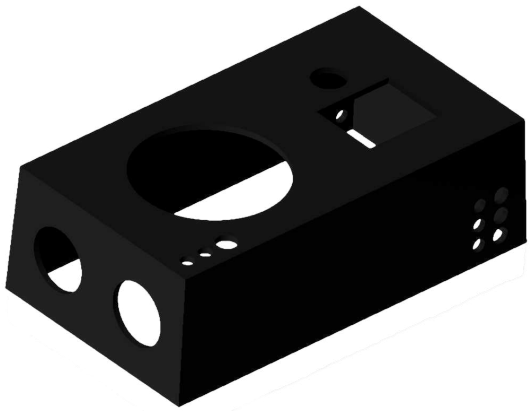


Figure 12.14.: 3D-model of the AudioAnt case

12.3.2. Circuit diagram

12.3.2.1. LEDs, Buttons, Display and the Microphone

The LEDs were connected to a GPIO-pin and required a pre-resistor for proper functioning. The buttons were connected to a GPIO-pin and a ground pin. By enabling the pull-up resistor by software, the installation of a pre-resistor is not necessary.

Whereas buttons and LED's can be wired to any GPIO-pin, some of the display pins such as the clock and SPI-pins have to be connected to the clock or SPI-pins on the Raspberry Pi.

The microphone was connected to the AUX port of the AudiAnt which leads to the soundcard. Due to the microphone being such a tiny component, it gets its power from the AUX port.

12.3.2.2. Power demanding components

The power demanding components were directly connected to the power source. Furthermore, a connection to the Raspberry Pi was established to control the speaker and the light source which is controlled by a relay. The relay was connected to one of the GPIO-pins to control the flashing of the LED when a sound occurs. The speaker was in addition connected to one of the AUX-ports which leads to the soundcard attached to the Raspberry Pi.

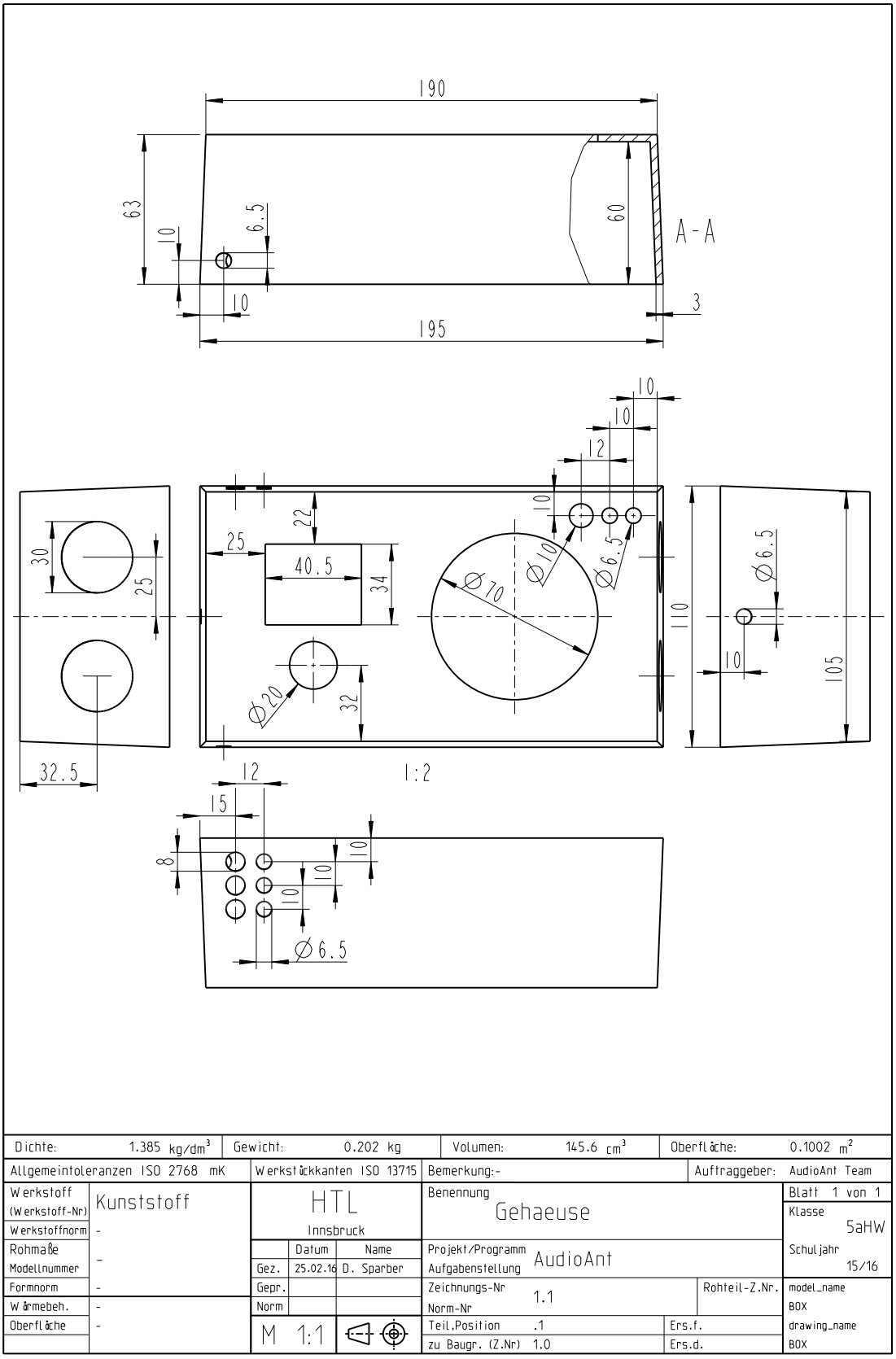


Figure 12.15.: Metrics of the case

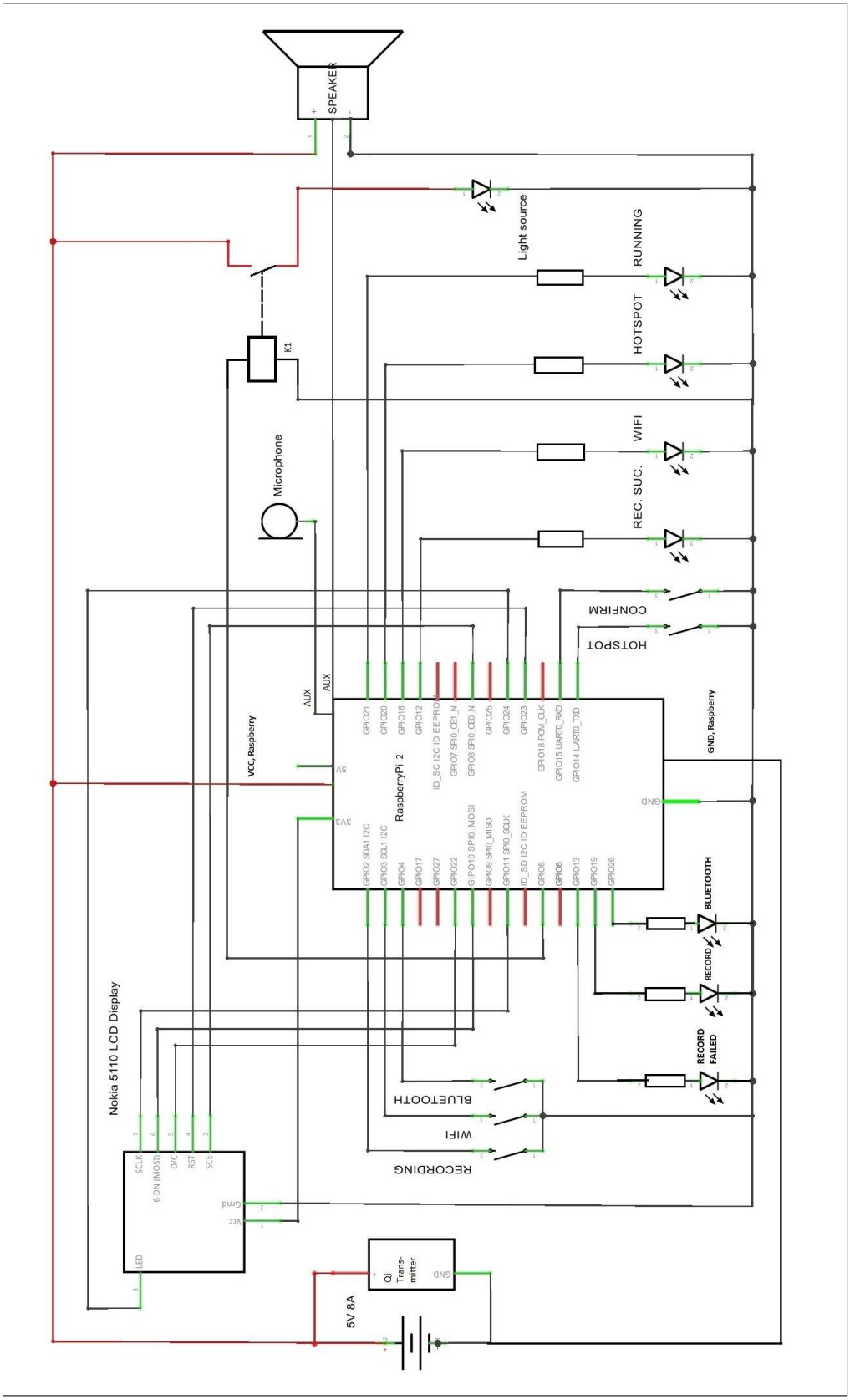


Figure 12.16.: Circuit diagram

12.4. Construction

After procuring all the components and a long planning phase, the construction was started.

Figure 12.17.: Components of the AudioAnt

12.4.1. Circuit board

We decided to equip a circuit board because it is easier to handle and ensures better clarity. Therefore, an IDE cable was split up into 40 single wires which were soldered to the board. In addition, the resistors for the LEDs were attached to the board. Further wiring was continued from the board to other components like the relay or the display.

12.4.2. Case

For the case to be ready, many holes had to be cut out which can be done with a standing drill or a small tooling machine.

- Jack socket
  A hole with a diameter of ⌀ 6.5cm had to be cut out on the right side.

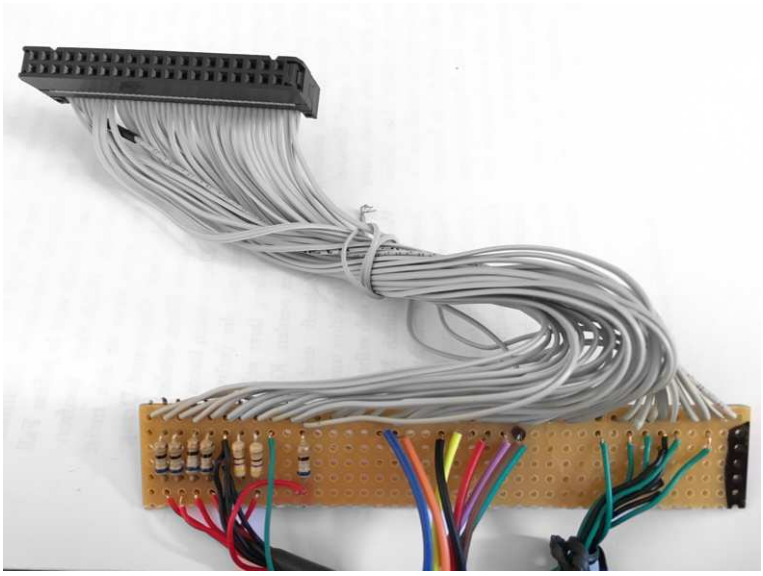


Figure 12.18.: Circuit Board and the soldered cables

- USB-AUX connectors
  For the USB-AUX connectors it is necessary to cut out two holes on the left side of the AudioAnt with a diameter of ⌀ 30mm.

- Qi-transmitter
  The Qi-transmitter gets placed on the top and needs a hole with a diameter of ⌀ 70mm.

- Display
  The display is located on the left side of the Qi-transmitter on the top of the AudioAnt and needs a rectangular cut with 40.5 x 34mm (height x width).

- Confirm button
  The confirm button is also on the top of the AudioAnt and needs a hole with a diameter of ⌀ 20mm.

- Buttons
  Besides the confirm button, there are four other buttons on the AudioAnt. One of them is located on the top and three buttons are located on the back of the AudioAnt. The holes for the buttons have a diameter of ⌀ 8mm.

- LEDs
  The six LEDs need a hole with a diameter of ⌀ 6.5mm. Two of the LEDs are located on the top of the AudioAnt, three are next to the buttons on the back and one LED is on the front.

12.4.3. Light source

The light source had a board and a button attached, which allowed the user to switch between a blinking light, a dimmed and a bright light. The board had to be removed because it would

not have been possible to control the LED with the relay. The wires were simply cut off, the board was removed and the wires were soldered back together. We decided not to integrate a pre-resistor because the LED is not in constant use.

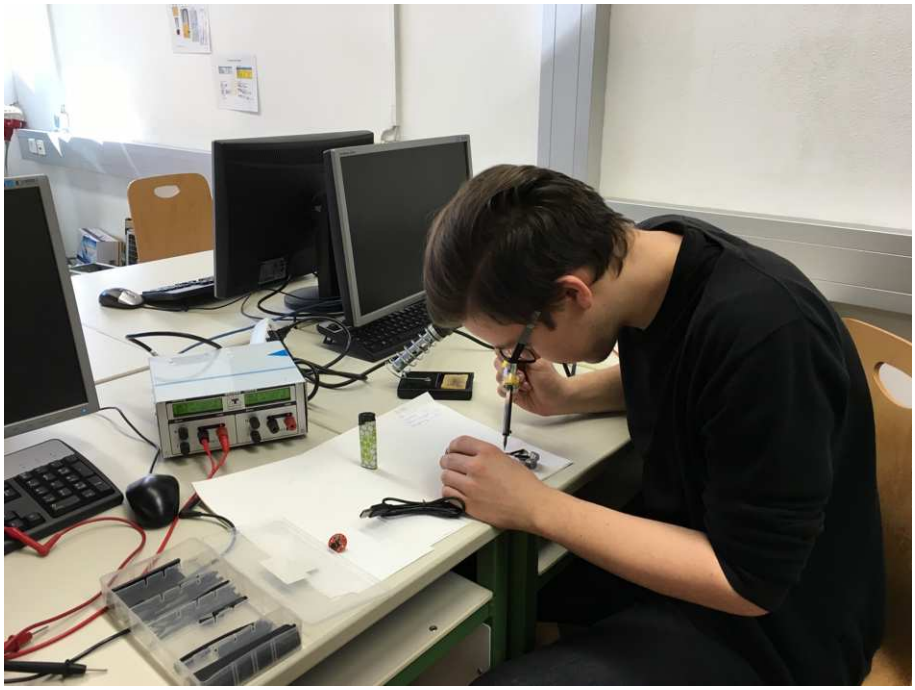


Figure 12.19.: Florian soldering the LED

Figure 12.20.: LED soldered and isolated with shrinking tube

12.5. Finished prototype

Figure 12.21.: The finished prototype

Figure 12.22.: The finished prototype (open)

12.6. Functionalities

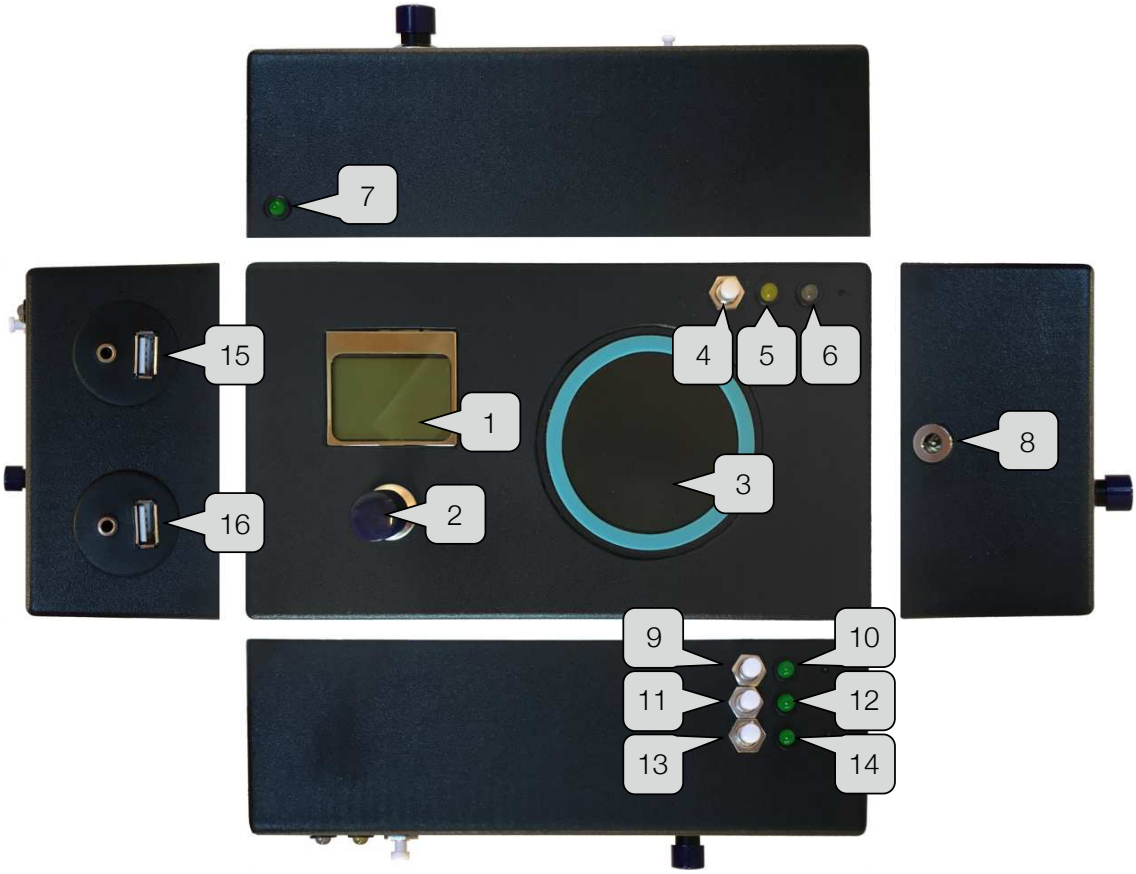


Figure 12.23.: Labelled components of the prototype

12.6.1. General Functionality

When the AudioAnt is plugged in, the device boots up. After the AudioAnt has started, an LED (7) displays that the analysis is running and the assistant tool listens for familiar sounds.

When the AudioAnt recognises a sound, the according sound name is shown on the display (1), the alert light starts to blink, and an alarm tone is played. After the user confirms that he heard the sound by pressing the according button (2), the alarm stops.

Phones which support wireless charging or have a suitable adapter (Qi standard) can be placed within the blue circle (3) to recharge the battery.

12.6.2. Recording

For recording a new sound, the according button (4) has to be pressed once to start the learning process. After the button is released, an LED (5) signals that a sound can be recorded. When

the new sound has been recorded, the recording process can be stopped by pressing the same button (4) again. If the learning process succeeded, the result LED (6) lights up green three times, otherwise it blinks red.

12.6.3. Connectivity

There are two options to connect to the AudioAnt: via Wi-Fi and via Bluetooth. For establishing a Wi-Fi connection the corresponding button (13) can be pressed. Similarly, for opening an Wi-Fi access point, the hotspot button (11) has to be pressed and for enabling the Bluetooth chip button 9 needs to be pressed. The LEDs next to the buttons signal the state of the connectivity interfaces. If the interface is up, the LED is on.

12.6.4. Audio ports

For connecting a speaker to the AudioAnt, the AUX port of the first connector (15) can be used for the audio signal and the USB port for the power supply. Likewise a microphone for the AudioAnt can be connected the USB or AUX port of the second connector (16).

12.6.5. Summary

Nr.	Name	Purpose
1	Display	Showing important information
2	Confirm button	Disabling the alarm
3	Qi Transmitter	Charging a mobile phone
4	Recording button	Starts and stops the recording process
5	Currently recording LED	Shows if the AudioAnt is currently recording a new sound
6	Recording result LED	Shows whether the recording process failed or succeeded
7	Analysis running LED	Shows if the audio analysis is currently running
8	Jack socket	To supply the AudioAnt with power
9	Bluetooth button	Enables and disables Bluetooth
10	Bluetooth Status LED	Shows whether Bluetooth is on or not
11	Hotspot button	Open and closes a hotspot from the AudioAnt
12	Hotspot status LED	Shows whether a hotspot is opened or closed
13	Wi-Fi button	Enables and disables Wi-Fi
14	Wi-Fi Status LED	Shows whether Wi-Fi is on or not
15	Speaker connector	Power supply and signal for an external speaker
16	Microphone connector	Ports for connecting a microphone

Table 12.2.: Descriptive name and purpose of the components

12.7. Controlling the hardware components

12.7.1. General

For controlling the hardware components, such as LEDs, buttons or the display, Python was used. This choice was made, because accessing the GPIO pins with Python and the according `RPi.GPIO` package is the favoured option for the Raspberry Pi and recommended by a variety of platforms and developers.

Since the main program of the AudioAnt is written with Java, a communication needs to be established. For doing so, we decided on using a socket, since this is an easy and smooth way. The commands were sent as simple, UTF-8 encoded, semicolon separated values, in order to ensure compatibility between the Java and Python program.

```python
# needed imports
import socket
import sys
from thread import *
import outputManager as outMngr
import inputManager as inMngr

HOST = ''   # Symbolic name meaning all available interfaces
PORT = 4207 # Used socket port

# Creating the socket
serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Binding the socket to the appropriate port and interface
serversocket.bind((HOST, PORT))
# Start listening (max. 5 connections)
serversocket.listen(5)

# Function for starting the sending process
def sendThread(clientsocket):
  inMngr.start(clientsocket) # Start the inputManager process

# Function for starting the receiving process
def receiveThread(clientsocket):
  while True: # Run until termination
    data = clientsocket.recv(1024) # Waiting for data from the Java program

    if not data: # If null was send the connection is broken
      break # Terminating the receive thread

    data = str(data) # Cast the data to a string

    for line in data.split("\r\n"): # Split if multiple command lines were sent at once

      function = line.split(';')[0] # The first argument is the function
      options = line.split(';')[1:] # The following arguments are the options

      outMngr.output(function, options) # Call the outputManager for handling the command

# Loop enabling multiple connections
while True:
  # wait for the next connection - blocking call
  (clientsocket, address) = serversocket.accept()
  start_new_thread(receiveThread ,(clientsocket,)) # Start new receiveThread
  start_new_thread(sendThread ,(clientsocket,)) # Start new sendThread
```

Listing 12.1: The connection handler for the hardware controller

12.7.2. Configuration files

For taking hold of important parameters for the hardware controller two files were created. The first file `PINS.py` saves which hardware component is connected where. The other one, the `CONFIG.py` file is needed for storing parameters needed for running the tool.

```
alertLight = 6
ledWifi = 16
ledBluetooth = 26
ledHotspot = 20
ledRunning = 21
ledRecording = 19
ledRecordingFailed =  12
ledRecordingSuccess = 13

buttonRecording = 2
buttonWifi = 3
buttonBluetooth = 4
buttonHotspot = 14
buttonConfirm = 15

displayBL = 24
displayDC = 22
displayRST = 23
displaySpiPort = 0
displaySpiDevice = 0
```

Listing 12.2: The pin configuration

```
alertLightDelay = 0.06 # seconds
alertLightBlinkTimes = 30
ledBlinkDelay = 0.15 # seconds
displaySleepTime = 0.03 # seconds
displayFont = "fonts/pixel_font-7.ttf"
```

Listing 12.3: The general configuration

12.7.3. Output manager

For dealing with all hardware components concerning the output, the so called `OutputManager` was implemented. This Python component takes in the sent command from the running analysis program, decides which hardware component should be used and redirects accordingly.

```
# Needed imports
import ledController as led
import alertLightController as alert
from display import Display

display = Display() # Initialising the display

# Defining the function for handling a function call
def output(function, options):

  if function == "LED":
    led.ledByOptions(options) # Redirect to the LED controller
  if function == "ALERT_LIGHT":
    alert.blink() # Redirect to the alert light script
  if function == "DISPLAY":
    display.write(options) # Writing the text to the display
```

Listing 12.4: The output manager of the hardware controller

12.7.3.1. LEDs

For controlling the LEDs only the according pin needs to be set high or low, since the LEDs are directly connected to the GPIO pins. The led controller computes the options of the requests and lights or darkens the matching led according to the requested options.

```
# Needed imports
import RPi.GPIO as GPIO
import time
import CONFIG
import PINS

GPIO.setmode(GPIO.BCM) # Setting up the GPIO pin layout
# Defining all led pins as outputs
GPIO.setup(PINS.ledWifi, GPIO.OUT)
GPIO.setup(PINS.ledBluetooth, GPIO.OUT)
GPIO.setup(PINS.ledHotspot, GPIO.OUT)
GPIO.setup(PINS.ledRunning, GPIO.OUT)
GPIO.setup(PINS.ledRecording, GPIO.OUT)
GPIO.setup(PINS.ledRecordingFailed, GPIO.OUT)
GPIO.setup(PINS.ledRecordingSuccess, GPIO.OUT)

def ledWifi(on): # Toggle the WIFI led
    GPIO.output(PINS.ledWifi, on)
def ledBluetooth(on): # Toggle the bluetooth led
    GPIO.output(PINS.ledBluetooth, on)
def ledHotspot(on): # Toggle the hotspot led
    GPIO.output(PINS.ledHotspot, on)
def ledRunning(on): # Toggle the running status led
    GPIO.output(PINS.ledRunning, on)
def ledRecording(on): # Toggle the recording status led
    GPIO.output(PINS.ledRecording, on)
def ledRecordingFailed(on): # Toggle the recording failed led
    GPIO.output(PINS.ledRecordingFailed, on)
def ledRecordingSuccess(on): # Toggle the recording success led
    GPIO.output(PINS.ledRecordingSuccess, on)

def led(ledName, option): # Toggle a led by name and option

  if int(option) > 1: # If the number is higher than 0 -> blink
    for i in range(0,option): # Blinking as often as the options say
      led(ledName, 1) # Led on
      time.sleep(CONFIG.ledBlinkDelay) # delay
      led(ledName, 0) # Led off
      time.sleep(CONFIG.ledBlinkDelay) # delay
  else:
    ledOn = int(option) != 0 # if the option is 0 -> led off, else led on
    if(ledName == "LED_BLUETOOTH"): # toggle bluetooth led
      ledBluetooth(ledOn)
    elif(ledName == "LED_WIFI"): # toggle WIFI led
      ledWifi(ledOn)
    elif(ledName == "LED_HOTSPOT"): # toggle hotspot led
      ledHotspot(ledOn)
    elif (ledName == "LED_RUNNING"): # toggle running status led
      ledRunning(ledOn)
    elif (ledName == "LED_RECORDING"): # toggle recording status led
      ledRecording(ledOn)
    elif (ledName == "LED_RECORDING_FAILED"): # toggle recording failed led
      ledRecordingFailed(ledOn)
    elif (ledName == "LED_RECORDING_SUCCESS"): # toggle recording success led
      ledRecordingSuccess(ledOn)

def ledByOptions(options):
    led(options[0], options[1]) # Split the options into name and number
```

Listing 12.5: The LED controller

12.7.3.2. Alert light

The alert light was connected to a relay, since the Raspberry Pi could not provide enough power for such a bright light. Nevertheless, the alert light was controlled by the GPIO pins of the Raspberry Pi, since the relay was toggled by one of those pins.

```
# needed imports
import RPi.GPIO as GPIO;
import time
import PINS
import CONFIG

GPIO.setmode(GPIO.BCM) # Setting up the pin mode
GPIO.setup(PINS.alertLight, GPIO.OUT) # Declare the alert light pin as output

def blink(): # Alert light blinks
    for i in range(CONFIG.alertLightBlinkTimes): # Loop through the wanted amount of numbers
        on(); # Switch the pin to high
        time.sleep(CONFIG.alertLightDelay) # delay
        off(); # Switch the pin to low
        time.sleep(CONFIG.alertLightDelay) # delay

def on(): # Switch the alert pin on
    GPIO.output(PINS.alertLight, True)

def off(): # Switch the alert pin off
    GPIO.output(PINS.alertLight, False)
```

Listing 12.6: The alert light controller

12.7.3.3. Display

For writing text onto the display, the `Adafruit_Nokia_LCD` library can be used. After importing this library, graphical elements can be added to the display easily, by drawing on an image object which will then be displayed. For a detailed documentation of this project, visit the project page (https://github.com/adafruit/Adafruit_Nokia_LCD).

12.7.4. Input manager

Contrary to the output manager, the input manager focuses on receiving signals and sending them to the Raspberry Pi. The only thing this component works with are buttons, since no other hardware is used for inputs.

```
# needed imports
from thread import *
import RPi.GPIO as GPIO
import PINS
import time

GPIO.setmode(GPIO.BCM) # Setting up the pin mode

def send(text, socket): # Method for sending text
  text = text + "\r\n" # Terminating the text by a line break
  if socket.send(text.encode('utf-8')) == 0: # Send the text and look if it was successful
    print("Sending failed (" + text + ")") # Print an error
```

Listing 12.7: The input controller (part 1)

```
# Method for sending messages if the button was pressed
def sendThread(buttonname, pin, socket):
    # Setting up the putton as a pull up input
  GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    # Run as long as the pipe is not broken
  while True:
    try:
      GPIO.wait_for_edge(pin, GPIO.FALLING) # Wait for a press on the button
      time.sleep(0.1) # Delay due to stability reasons
      GPIO.wait_for_edge(pin, GPIO.RISING) # Wait for the release of the button

      send("BUTTON;" + buttonname, socket) # Send an information

    except (BrokenPipeError, IOError): # Break the loop if an error occured
      break # Break the loop
```

Listing 12.8: The input controller (part 2)

For adding buttons to the input manager, the following code snippet has to be used. In this example the recording button was set up.

```
def start(socket):
    # Buttons have to be added here
    start_new_thread(sendThread ,("BUTTON_RECORDING", PINS.buttonRecording, socket,))
```

Listing 12.9: Adding a button to the input controller

13. Android Application

13.1. General application description

13.1.1. Application requirements

With our application, we wanted to fulfil the following tasks:

- connect to the AudioAnt wirelessly
- configure the AudioAnt depending on the users input
- react on incoming signals from the AudioAnt
- customise how the Smartphone should react when a signal was identified by the AudioAnt
- record Sounds and send them to the AudioAnt to learn them

13.1.2. Connectivity

Early on during the project, we decided to use wireless connection methods to communicate between the AudioAnt and whatever device it would communicate with. We chose to do so, because of the convenience and simplicity these ways of transferring data provides to the user and because of how easy it is compared to wired connections, command line tools, etc.

We chose Wi-Fi as a communication method, because it is relatively easy to implement, supported by almost all devices and has a very good reach, compared to other alternatives such as e.g. Bluetooth.

13.1.3. Connection service

To establish the connection between the devices, send data to the server and read data from it if possible, a so called "service" is used. (For more info see section 13.2.2) This service runs in the background of the application and provides a communication interface for exchanging data between the smartphone application and the AudioAnt.

13.1.3.1. Initial connection

To initially establish a connection between the two devices, the AudioAnt opens a Wi-Fi hotspot the Smartphone can connect to. Because the IP-Address of the AudioAnt can be static, the application can establish a socket connection between the devices and exchange data.

Since sockets function according to a server - client model, there always has to be a server that a client can connect to. In our case, the AudioAnt will function as the server, which gets enabled once the AudioAnt powers on and the WiFi button is pressed. After creating the hotspot and a ServerSocket, the AudioAnt will wait for any client to connect to it.

On first start, the smartphone application automatically connects to the new Wi-Fi network of the AudioAnt and establishes a connection. Now sounds can be learned and the notification methods can be set as wanted. While this setup requires no action from the user, it still has one major drawback. As long as the smartphone is connected to the AudioAnt, it is not possible to reach the internet, since the AudioAnt has no possibility of forwarding the traffic to any other server.

If a home Wi-Fi network is available, however, this problem can be avoided, by connecting both the AudioAnt and the smartphone to a local Wi-Fi network over which they can communicate. This can be configured using the application to ensure a good usability and no frustration for the user.

13.1.3.2. Using the local Wi-Fi

In order to connect the AudioAnt to a local Wi-Fi network, the access data have to be sent from the applications "AudioAnt Settings" screen (see figure 13.2b). After sending, the AudioAnt saves the connection data transmitted and connects to the network as soon as its own hotspot is disabled.

Because it is not possible to assign a static IP-Address to the AudioAnt and the smartphone and the addresses can therefore change, the hostnames of the devices are used to establish the connection between the devices. Since the hostname of the AudioAnt never changes automatically, a known name can be set by default in the program and the smartphone application always knows where to send the data to.

13.1.4. Data transfer

In order to be able to distinguish between different commands and data that are exchanged between the AudioAnt and the smartphone, some kind of protocol has to be used for communication. Because the AudioAnt functions as the server, it mostly stays passive and only reacts to data received from the application, except when a sound is recognised and the devices have to be informed.

As a data transmission format, we chose to use solely JSON files.

JSON (JavaScript Object Notation) is a data exchange format that saves its values in a key-value structure. It can store strings, booleans, numbers and objects (for example "data" in listing 13.1).The benefits of using JSON, compared to just sending strings with known separators, are the ease of use it provides, both when creating and when parsing (reading from it), as seen in listing 13.2.

```
{
  "action":"changeSettings",
  "data":{
    "lightSignals":true,
    "audioSignals":true,
    "audioVolume":100
  }
}
```

Listing 13.1: Example for exchanged JSON object

The above example shows a JSON object that is sent if the user changes the settings of the AudioAnt.

We used the value of the "action" key to differentiate between the actions the AudioAnt or the smartphone should trigger, if a file is received. A few examples for keywords we used are:

- getListOfSounds
- saveSound
- connectToWi-FiNetwork
- recognisedSound
- getMoreInfoToSound
- ...

```
String action = object.getString("action");
boolean lighsignals = object.getJSONObject("data").getBoolean("lightSignals");
```

Listing 13.2: Getting the variables from JSONObject "object"

13.1.4.1. Audio file transfer

For our application, we have to transfer audio files on a regular basis and need a defined and practical way of doing so. When transferring files, the content of the file is first converted into an array of bytes. This array is then converted into a String using Base64 encoding. The encoded string can then be added to a JSON file as part of the "data" object.

Base64 was chosen as encryption method for two reasons:

1. It can be converted both ways without changing the result
   When converting an array of bytes into a string and then back again using, for example, UTF-8, the result will not be the same. Therefore, the original file will not have the same

content as the file that was converted and sent, which will result in errors when trying to play the .mp3 file.

Using Base64, however, the result stays the same, no matter how often it is being converted, because it uses 1-to-1 mapping. This means that *"...every byte sequence can be mapped to a unique sequence of characters, and back."* [1]

2. It doesn't contain misunderstandable characters
   For transferring the string inside the JSON file, it is important that it does not contain characters like quotation marks, for example, that would cause parsing issues when trying to read values from the JSON object. Unlike encodings such as ISO-8859-1 that also use 1-to-1 mapping, Base64 only uses the letters a to z in both capital and small letters as well as "+" and "/". Therefore, sending Base64 encoded Strings in a JSON object does not lead to errors and can be done trouble-free.

13.1.5. Activities

Activities can be best described as the screen of an application that a user sees. It provides an interface for user interaction with application elements and is mostly designed via .xml files. The activity shown depends on the user's input, each activity has an own task it is used for.

To minimise the chance of inexperienced users getting lost or confused using the application, there are info buttons in almost every activity that display important information about the activity and what can be achieved if certain actions are executed.

13.1.5.1. Main activity

This is the activity that is opened by default on application start. It provides buttons to navigate to the other activities of the Application. It also indicates whether the smartphone is connected to the AudioAnt or not.

On application start, the activity checks if the AudioAnt has been set up to connect to a local Wi-Fi network and if so, connects to the AudioAnt. If it is the first start of the application, a dialog will automatically open to explain how the communication between the AudioAnt and the smartphone works and which steps have to be executed in order to establish a connection.

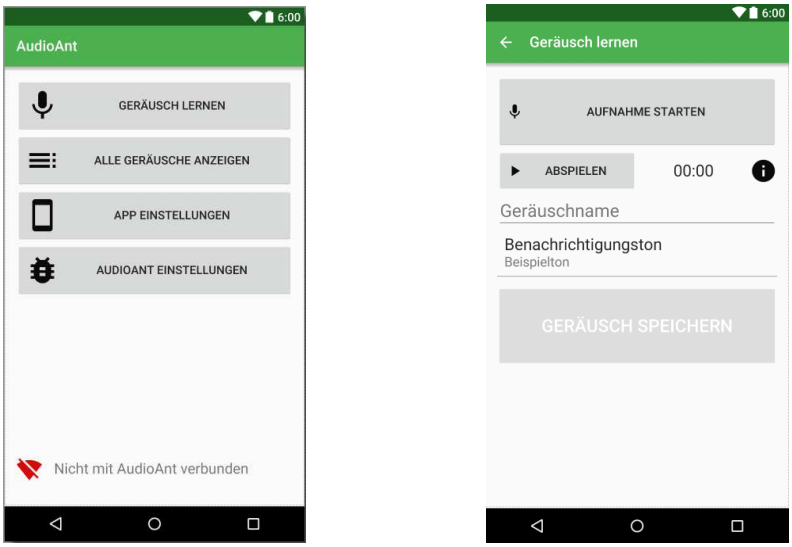
13.1.5.2. Record activity

This activity provides an interface for recording sounds the AudioAnt should later recognise.

Sounds can be recorded and replayed to check for correct recording, a sound name can be entered and a notification sound (to emit on the AudioAnt) can be chosen.

[1] Stackoverflow - Byte to String and back
http://stackoverflow.com/questions/9098022/problems-converting-byte-array-to-string-and-back-to-byte-array

If all necessary steps are fulfilled, the button to send the new sound to the AudioAnt gets enabled and can be used.



(a) Main activity **(b)** Recording activity

Figure 13.1.: The main and recording activity of the Android application

13.1.5.3. Smartphone settings activity

Here, the smartphone can be configured to react to signals coming from the AudioAnt. This happens, for example, when a sound is recognised on the AudioAnt and the smartphone is currently connected to the AudioAnt.

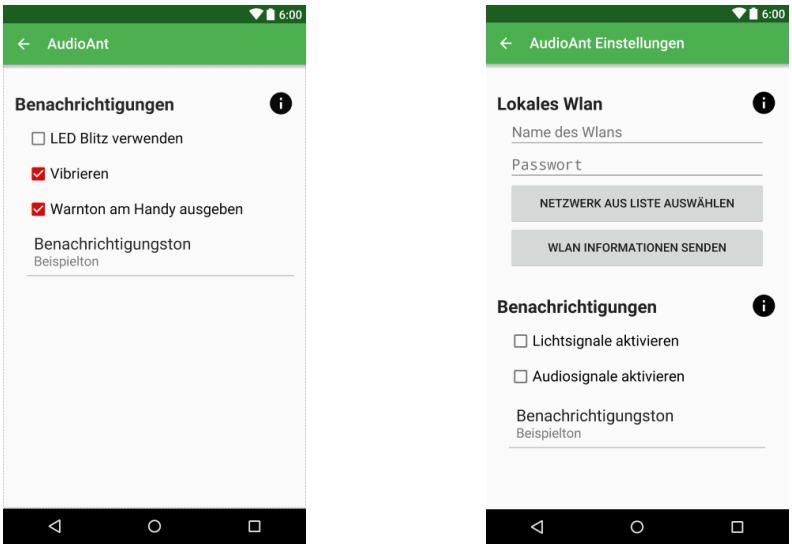
The settings will only be stored on the phone, therefore, there is no need for a connection to the AudioAnt to use this activity.

13.1.5.4. AudioAnt settings activity

In this activity, the current settings of the AudioAnt can be viewed and modified by the user. In order to display and change the settings, a connection to the AudioAnt is needed, since the application does not save the settings of the AudioAnt but rather requests them whenever the activity is called.

As soon as the activity is left (either via the navigation buttons on the bottom of the smartphone or via the back arrow in the top left corner), all the data will be put in a JSON file and sent to the AudioAnt.

Using this activity, the AudioAnt can also be configured to connect to a local Wi-Fi network, however, changes to these settings will only be sent if explicitly requested by the user by pressing the corresponding button.



(a) Smartphone settings activity **(b)** AudioAnt settings activity

Figure 13.2.: The settings activities of the Android application

13.1.5.5. List-of-learned-sounds activity

Just like the AudioAnt settings activity, this one also needs a connection to the AudioAnt in order to work properly. When started, it requests a list of all the sounds the AudioAnt has learned and displays them once received. It also enables the user to delete sounds if wished.

13.1.5.6. Alarm activity

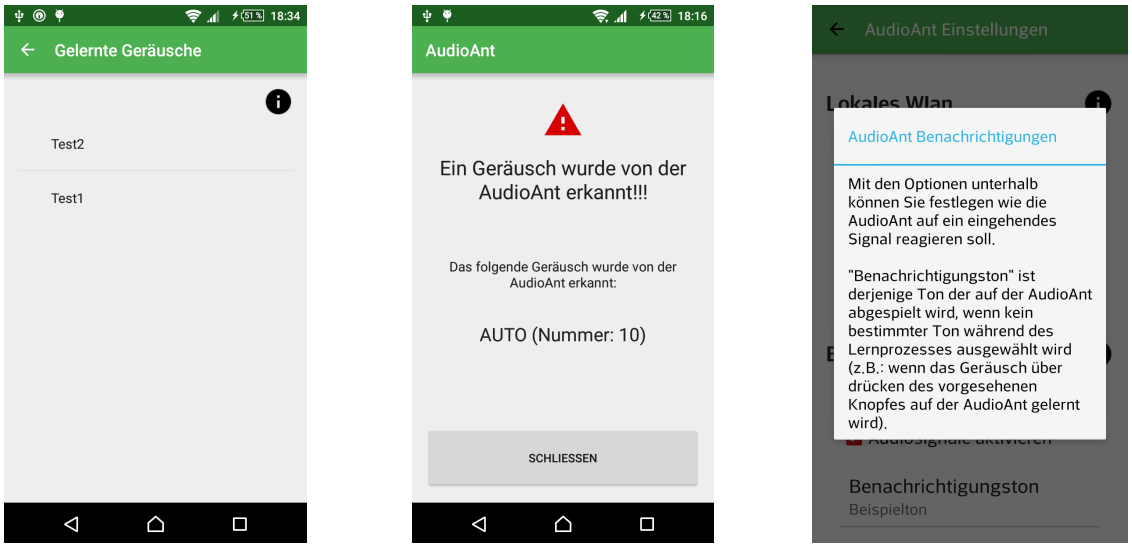
If one or more sounds are recognised by the AudioAnt, this activity automatically gets opened, even if the application itself isn't currently open or active.The alert methods of the smartphone are used until either the "Schließen" button on the smartphone or the "confirm" button on the AudioAnt is pressed.

13.1.5.7. Info buttons

Because using an application can be somewhat overwhelming when doing so for the first few times, we decided to use help buttons to display information to the user when needed.

These help buttons appear in almost every activity and explain to the user what can be done in this activity and what the user should do to accomplish a task.

The info buttons are always located on the right side of the screen, and on the same level as the heading of the section of they screen they offer information to.



(a) List of learned sounds **(b)** Alarm activity **(c)** Info-text

Figure 13.3.: The list of all sounds, the alarm activity and an informational text

13.1.6. Abandoned functionality

At the beginning of the project we had a few ideas for features our application should contain. During the course of our research and the development process of the AudioAnt and the application we realised, however, that some of the features were not realisable for different reasons.

13.1.6.1. Audio streaming

In the beginning of the project, we had the idea of using the built-in microphone of the smartphone to record and stream audio from the smartphone to the AudioAnt. By doing so, we wanted to increase the effective range of the audio analysis of the AudioAnt.

Processing time

Since we record audio with a sample rate of 16.000Hz and the audio analysis always takes blocks of 2048 samples to analyse, we can calculate how fast the audio analysis on the AudioAnt has to be if it should be finished before the next 2048 samples are ready to be analysed.

$$\frac{2048}{16000s^{-1}} = 0.128s$$

Since every 128 milliseconds the AudioAnt would have to process two audio streams, the maximum time necessary to analyse one audio stream would have to be shorter than 64 milliseconds.

Raspberry Pi 2

Unfortunately, the hardware of the chosen microcontroller, the Raspberry 2, was not powerful enough to analyse two audio stream simultaneously.

Under normal load, the audio analysis takes about 115 ms to analyse 2048 samples, and under heavy load (loud noise, every feature has to be extracted), the average analysis time is about 150 ms.

Because these analysis times are too high to enable the analysis of two audio streams simultaneously, we dropped the idea of streaming audio from the smartphone to the AudioAnt.

Raspberry Pi 3
Later, during the course of the project, we replaced the Raspberry Pi 2 with a Raspberry Pi 3 as the processing unit of the AudioAnt. This step happened relatively late, since this Raspberry Pi was not launched earlier.

Under maximal load, the Raspberry Pi 3 took about 62ms to analyse 2048 samples and would therefore be able to analyse two audio streams at the same time. However, because the change to the Raspberry Pi 3 happened so late, we decided not to pick up on the idea again because it would take too much of an effort to complete in time.

13.1.6.2. Speech recognition

Another idea we had for the application was to use Google's speech recognition feature to check for things like cries for help and then react to occurrences by informing, for example, relatives about it.

Although Android phones are capable of recognising speech, even when not connected to the internet, audio is sent to Google to analyse if an internet connection is available. Because this cannot be disabled and would produce too much network traffic, we abandoned this idea as well.

13.2. Implementation

13.2.1. Application Manifest

The AndroidManifest.xml is a file every Android application must have in order to run and contains important information about the application, such as, for example, the required permissions, the icon of the application and the components of the application.

13.2.1.1. Application, Activities, Services

This part of the AndroidManifest shows which activities are in the application, which services are used, the style and layout of the app as well as some other things. For every activity, for instance, the parent activity that will be called when pressing the home-as-up button, can be specified as well as things like the screen orientation or if the keyboard should be opened by default if an EditText View exists within the activity.

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".Activities.MainActivity"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".Activities.RecordActivity"
            android:label="Geraeusch lernen"
            android:parentActivityName=".Activities.MainActivity"
            android:screenOrientation="portrait"
            android:windowSoftInputMode="adjustPan">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".Activities.MainActivity" />
        </activity>
    ...
    ...
        <service
            android:name=".Model.Services.CommunicationService"
            android:enabled="true"
            android:exported="true" />

    </application>
</manifest>
```

Listing 13.3: permissions required for the application

13.2.1.2. Permissions

The following system permissions were necessary to realise our application.

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.FLASHLIGHT" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.VIBRATE" />
```

Listing 13.4: Permissions needed for the AudioAnt application

13.2.2. Services

Services are used in Android for handling background tasks that can be started and stopped independently from anywhere within the application. This distinguishes them from Threads that can only be stopped inside the class they were invoked (except if they are methods coded for other classes to access). For our application, we used services for recording audio and for communicating with the AudioAnt.

Services don't automatically run in their own thread, but on the main thread. Therefore, new threads have to be started when handling process intensive tasks or tasks such as reading from a socket connection that block until data is received.

13.2.2.1. Unbound services

A normal service can either function as a bound service or as an unbound service.

Unbound services are used for simpler tasks that do not require communication between the service and the rest of the application. It is not possible to invoke methods of a service from an activity. They are started using *startService()* and can be stopped using *stopService()*, although a service can also stop itself.

13.2.2.2. Bound services

Contrary to unbound services, bound services enable communication with other application elements. A class can bind to the service using *bindService()* and communicate with the service using a so called "IBinder".

Binding to a service
When binding to a service, the function *bindService(Intent intent, ServiceConnection conn, int flags)* is being used.

In order to be able get the instance of the running server, when implementing a bound service, we created our own inner class inside CommunicationService.Java, that inherits from the Binder class, and has a method that returns an instance of the service, as seen in listing 13.11 We then returned an instance of this class in the service's *onBind()* method.

As soon as the services *onBind()* method is finished, the ServiceConnections *onServiceConnected(ComponentName name, IBinder service)* is invoked with the IBinder "service" being the object that the service returned in its *onBind()* method.

We can then use the IBinder object to get the running instance of our service as an object. When wanting to call, for example, the *connectToServer()* method of our service, we can do so by calling *nameOfServiceObject.connectToServer()*.

For our application, we always declared the ServiceConnection object as a global variable and initialised it when declaring.

In every activity where a connection to the CommunicationService was necessary, the *onResume()* method binds to the CommunicationService and the *onPause()* method unbinds from it. This way the service is rebound every time an activity is opened and no memory leaks occur because of a service still being bound, when leaving the activity.

As seen in listing 13.5 and **??**, the previously initialised ServiceConnection is passed when binding and unbinding to the CommunicationService. This is necessary in order for its *onServiceConnected()* and *onServiceDisconnected()* methods to be invoked.

When invoking the *bindService()* method, the third parameter is always null. This is done, because services that are started with a flag of zero are not stopped when all clients disconnect as a bound service normally would do, but rather keep running until a *stopService()* command is executed. This is important for our project, because we want the CommunicationService to keep running, even if no activities are bound to it or the application itself is closed. This way, the AudioAnt will continue to communicate with the running service and the AlarmActivity will still be called if a sound is recognised by the AudioAnt.

```
public void bindToCommunicationService() {
    if (!serviceIsBound) {
        Thread t = new Thread(new Runnable() {
            @Override
            public void run() {
                Intent i = new Intent(MainActivity.this, CommunicationService.class);
                bindService(i, serviceConnection, 0);
                startService(i);
            }
        });
        t.start();
    }
}

public void unbindFromCommunicationService() {
    try {
        unbindService(serviceConnection);
        serviceIsBound = false;
    } catch (Exception e) {
        Log.d(TAG, "service could not be unbound because it wasn't bound");
    }
}
```

Listing 13.5: Method for binding and unbinding CommunicationService.java

Communication

Activities can communicate with a service as above stated, using the IBinder object of the ServiceConnection's *onServiceConnected()* method to create an object containing the instance of the running service, enabling all the public methods to be executed.

If the service wants to communicate to parts of the application, it uses local broadcasts. Local broadcasts can send Intents to objects in the same process of the application. In order to receive a local broadcast, a BroadcastReceiver has to be registered. The registration can, for example, happen during the *onCreate()* method of an activtiy.

13.2.2.3. Recording service

The RecordSignalService gets started from the RecordActivity.java and records the audio to a File in .mp3 format.

Overridden methods

When inheriting from the Service class, the *onBind()* method has to be overridden. In our case, we overrode a few other methods such as *onStartCommand()* and *onDestroy()*.

The "return START_NOT_STICKY" of the method *onStartCommand()* tells the system that the process should be stopped instead of being restarted if the process is killed, something that can happen if the device runs out of memory [2]

Because the recording service is used as an unbound service, the *onBind()* method can return null. *onDestroy()* is invoked whenever *stopService(Intent i)* is called from an activity, with "intent" having the RecordSignalService.java as its target class.

```
@Override
public IBinder onBind(Intent intent) {
    return null;
}

@Override
public int onStartCommand(Intent intent, int flags, int startId) {

//      Check if the service is already running and start recording if not
    if (!alreadyStarted) {
        alreadyStarted = true;
        Thread t = new Thread(new Runnable() {
            @Override
            public void run() {
                recordToFile();
            }
        });
        t.run();
    }
    return START_NOT_STICKY;
}
```

Listing 13.6: OnBind() and OnStartCommand() methods of RecordSignalService.java

[2] Android Developers Blog, http://android-developers.blogspot.co.at/2010/02/service-api-changes-starting-with.html, 2015-03-15

```java

@Override
public void onDestroy() {
//      executed when stopService(....) is called from any class
    super.onDestroy();
    stopRecording();
    Log.d(TAG, "Die Aufnahme wurde angehalten!");
}
```

Listing 13.7: OnDestroy() method of RecordSignalService.java

Recording methods

The sampling rate of the file is set to 16000Hz, because ringtones usually have frequencies above this threshold, and since increasing the sampling rate would furthermore increase the necessary processing workload on the AudioAnt.

The method *getBaseDir()* returns the directory the audio file should be saved after creating it first, if necessary.

```java
public void recordToFile() {
//  Initialising the MediaRecorder and specifying recording properties
    recorder = new MediaRecorder();
    recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
    recorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
    recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
    recorder.setAudioEncodingBitRate(16);
    recorder.setAudioSamplingRate(16000);
//  Create the file the sound should be recorded to
    File f = new File(getBaseDir() + "/Signalton.mp3");
    if (f.exists())
        f.delete();
    try {
        f.createNewFile();
        recorder.setOutputFile(f.getAbsolutePath());
//      Start recording audio to file
        recorder.prepare();
        recorder.start();
        Log.d(TAG, "Die Aufnahme wurde gestartet");
    } catch (IOException e) {
        Log.e(TAG, "Die Aufnahme konnte nicht gestartet werden", e);
    }
}
```

Listing 13.8: recordToFile() method of RecordSignalService.java

```java
public void stopRecording() {
//  Aufnahme anhalten
    recorder.stop();
    recorder.release();
    recorder = null;
    Log.d(TAG, "Die Aufnahme wurde beendet");
}
```

Listing 13.9: stopRecording() method of RecordSignalService.java

13.2.2.4. CommunicationService

This service is responsible for all the communication between the AudioAnt and the smartphone. It receives and sends JSON files as strings to exchange data and broadcasts data to other parts of the application if received from the AudioAnt.

It functions as a bound service and is initially started from within the MainActivity's *onCreate()* method.

Overridden methods

Here, "return START_STICKY" was used in the *onStartCommand()* because we want the service to be running constantly, therefore, it should be restarted if the process is killed so the AudioAnt is able to send notifications to the smartphone.

The *onCreate()* Method is only invoked once, when the service is created, whereas the *onBind()* and *onStartCommand()* methods are invoked multiple times during the course of using the application. To ensure that there are no memory leaks in our application, every activity that connects to the service calls the *unbindService(ServiceConnection conn)* method before the application is closed or an other activity is opened.

```
@Override
public void onCreate() {
    super.onCreate();
    Log.d(TAG, "service created");
    initialiseNetworkConnection(false);
}

@Override
public IBinder onBind(Intent intent) {
    Log.d(TAG, "Service was bound");
    IBinder binder = new MyBinder();
    return binder;
}

@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    Log.d(TAG, "the service was started");
    return START_STICKY;
}

@Override
public void onDestroy() {
    super.onDestroy();
    Log.d(TAG, "Service was destroyed");
    try {
        reader.close();
        printer.close();
    } catch (IOException e) {
        Log.d(TAG, "reader and writer couldn't be closed");
    }
}

@Override
public boolean onUnbind(Intent intent) {
    Log.d(TAG, "The service was unbound");
    return super.onUnbind(intent);
}
```

Listing 13.10: Overridden methods of CommunicationService.java

MyBinder

An instance of this inner class is created and returned in the *onBind()* method of the CommunicationService. The *getService()* method returns the instance of the running CommunicationService.

```
public class MyBinder extends Binder {

    public CommunicationService getService() {
        return CommunicationService.this;
    }
}
```

Listing 13.11: Private class MyBinder.java

Establishing a connection

initialiseNetworkConnection()
When the smartphone is connected to a network, we want the application to constantly try to connect to the AudioAnt. This can be useful in use-cases where the AudioAnt is, for example, currently booting up and the ServerSocket therefore is not ready to accept client requests.

The problem with using only one thread to achieve this, is that the client does not automatically recognise that the server is now available but rather waits for a long time before timing out, whenever the ServerSocket is created after the client socket already tried to connect to it. To counter this problem, a synchronised thread *connectToServer(boolean reconnect)* is continuously being restarted every two seconds if the connection process does not succeed during this time.

Once the connection succeeds, the outer while(true) loop, labeled "A", is left (line 23), and no more attempts to connect to the AudioAnt are being made.

connectToServer()
The other activities are informed about the successfully established connection via a local broadcast. Depending on the value of the boolean variable "reconnect", which indicates whether the connection was previously stopped on purpose or not, different local broadcasts are being sent.

An example for a connection purposely getting destroyed is when the application tries to connect to a local wireless network instead of the hotspot and tries out if the name and password of the network, entered by the user, are correct.

listenForAudioAntMessages()
This method starts a thread that waits for incoming messages from the AudioAnt and broadcasts the data to the other parts of the application once received.

Every time a message is received, the service checks if the data indicates a recognised sound on the AudioAnt and either starts alerting the user by calling the AlertActivity or broadcasts the received data as an intent.

If the connection to the server is lost, the *initialiseNetworkConnection()* method is automatically invoked, ensuring that a connection is re-established as soon as possible.

```java
private void listenForAudioAntMessages(final BufferedReader reader) {
  Thread t = new Thread(new Runnable() {
      @Override
      public void run() {
          String incoming;
          try {
//            Listen for incoming messages
              while ((incoming = reader.readLine()) != null) {
                  Log.d(TAG, incoming);
//                Check if the incoming data indicates a recognised sound
                  if (checkIfRecognisedSounds(incoming)) {
                      startAlarmActivity(incoming); // Start the activity that notifies
                  } else {
                      sendLocalBroadcast(incoming); // Send the data to the activities
                  }
              }
              Log.d(TAG, "Die Verbindung zum Server wurde verloren");
              if (!socketClosedOnPurpose) {
                  Log.d(TAG, "trying to reconnect");
                  initialiseNetworkConnection(false);
                  handleServerDisconnected();
                  isConnected = false;
              }
          } catch (IOException e) {
              Log.d(TAG, "server closed");
              if (!socketClosedOnPurpose) {
                  Log.d(TAG, "trying to reconnect");
                  isConnected = false;
                  handleServerDisconnected();
                  initialiseNetworkConnection(false);
              }
          }
      }
  });
  t.start();
}
```

Listing 13.12: listenForAudioAntMessages() method of CommunicationService.java

```java
synchronized void connectToServer(boolean reconnect) {
    if (!isConnected) try {
//      Establish a connection to the server
        socket = new Socket(new Constants().AA_HOTSPOT_IP, new Constants().AA_HOTSPOT_PORT);
        reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        printer = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()));
        isConnected = true;

//      Start listening for incoming data from the server
        listenForAudioAntMessages(reader);

//      Inform Activities about completed connection process
        Intent i;
        if (reconnect) {
            Log.d(TAG, "Erneutes verbinden zum Server erfolgreich!!");
            i = new Intent("reconnected");
        } else {
            i = new Intent("verbunden");
            Log.d(TAG, "Verbindung zum Server erfolgreich");
        }
        LocalBroadcastManager.getInstance(CommunicationService.this).sendBroadcast(i);
    } catch (IOException e) {
        Log.d(TAG, "Verbindung zum Server konnte nicht aufgebaut werden");
    }
}
```

Listing 13.13: connectToServer() method of CommunicationService.java

```java
public void initialiseNetworkConnection(final boolean reconnect) {
    Thread t = new Thread(new Runnable() {
        @Override
        public void run() {
            A:
            while (true) {
                while (true) {
                    Thread tt = new Thread(new Runnable() {
                        @Override
                        public void run() {
//                          Try to connect to the Server
                            connectToServer(reconnect);
                        }
                    });
                    tt.start();

                    try {
//                      Try connecting to the server again if not successful after two seconds
                        Thread.sleep(2000);
                        if (!isConnected) {
                            tt.interrupt();
                            break;
                        } else break A;
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
    });
    t.start();
}
```

Listing 13.14: InitialiseNetworkConnection() method of CommunicationService.java

Content-sending methods

sendToServer()
This is the method used when sending any kind of data to the AudioAnt (JSON files as a string). The printer object is accessible because it is a global variable.

sendLocalBroadcast()
Since we chose to exchange information via JSON files, both internally and between the application and the smartphone, we were able to use the action of the JSON file as action of the Intent. This way, we can register BroadcastListeners in our activities that only reacted to the actions we wanted.

checkIfRecognisedSound() and startAlarmActivity()
If the action of a JSON file that should be sent equals "recognisedSound", the method returns true and the *startAlarmActivity()* method is invoked.

The "Intent.FLAG_ACTIVITY_NEW_TASK" flag has to be used because the service is no activity and in order to start an activity, a new task on the history stack has to be created.

The "Intent.FLAG_ACTIVITY_CLEAR_TASK" flag clears existing tasks that are associated with the activity. This is necessary, so that the alerting methods of the alarm activity do not

get started a second time. Instead, the old threads that alert the user are stopped and the newly opened activity starts the alerting methods again.

The "Intent.FLAG_ACTIVITY_NO_ANIMATION" makes sure that no unnecessary animations are performed when the new activity is called.

```
public void sendToServer(String content) {
  try {
      Log.d(TAG, "string wird an server gesendet");
      printer.write(content);
      printer.flush();
  } catch (NullPointerException e) {
      Log.d(TAG, "fehler beim senden, server stream nicht bereit");
  }
}
```

Listing 13.15: sendToServer() method of CommunicationService.java

```
private void sendLocalBroadcast(String s) {
//Send local broadcast with received json object as extra
  Intent intent;
  try {
      JSONObject jsonObject = new JSONObject(s);
      intent = new Intent(jsonObject.getString("action"));
      intent.putExtra("json", jsonObject.toString());

      LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
      Log.d(TAG, "sending broadcast");

  } catch (JSONException e) {
      Log.d(TAG, "received invalid json");
  }
}
```

Listing 13.16: sendLocalBroadcast() method of CommunicationService.java

```
private void startAlarmActivity(String jsonFile) {
//      Start Activity that informs the user
    Intent i = new Intent(CommunicationService.this, AlarmActivity.class);
    i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
    i.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
    i.putExtra("json", jsonFile);
    startActivity(i);
}
```

Listing 13.17: startAlarmActivity() method of CommunicationService.java

Other functionality

When the connection to the server is lost, the application automatically returns to the MainActivity. The MainActivity then re-initialises the connection process with the server.

When an activity wants to close the connection to the server, they can call *closeSocket()*. This methods sets socket, reader and printer to null and also sets the variable "socketClosedOnPurpose" to true, so that no attempts at connecting automatically are made.

13.2.3. Wi-Fi connection

To simplify the use of Wi-Fi as much as possible, a helper class was implemented that offers easy access to the most used functionality.

This includes enabling Wi-Fi, checking if the smartphone is connected to a wifi network, scanning for available Wi-Fi networks, and more.

connectToWifi()
This method is used to connect the smartphone to a Wi-Fi network. We first delete any networks with the same name from the list of known networks to avoid having a network show up multiple times, for example, in the Wi-Fi section of the smartphone's settings application. The network is then added to the known network and set as enabled.

Before being able to get a list of networks, the application waits until the Wi-Fi adapter is enabled because the list of configured networks would otherwise be "null".

```
public void connectToWifi(String ssid, String password) {
    WifiConfiguration wifiConfiguration = new WifiConfiguration();
    wifiConfiguration.SSID = "\"" + ssid + "\"";
    wifiConfiguration.preSharedKey = "\"" + password + "\"";
    wifiConfiguration.status = WifiConfiguration.Status.ENABLED;

    //wait untli wifi is turned on when started with wifi turned off
    while (!wifiManager.isWifiEnabled()) {
        try {
            Thread.sleep(50);
            Log.d(TAG, "had to wait");
        } catch (InterruptedException e) {

        }
    }

    //Damit netzwerke nicht doppelt auftauchen
    List<WifiConfiguration> bekannteNetzwerke = wifiManager.getConfiguredNetworks();
    for (int i = 0; i < bekannteNetzwerke.size(); i++) {
        if (bekannteNetzwerke.get(i).SSID.equals("\"" + ssid + "\"")) {
            int netId = bekannteNetzwerke.get(i).networkId;
            wifiManager.removeNetwork(netId);
        }
    }
    int netId = wifiManager.addNetwork(wifiConfiguration);
    wifiManager.enableNetwork(netId, true);
    wifiManager.setWifiEnabled(true);
}
```

Listing 13.18: Connecting to a wifi network

startScan()
This methods starts scanning for available Wi-Fi networks and will send a broadcast with the action "WifiManager.SCAN_RESULTS_AVAILABLE_ACTION" that BroadcastReceivers can receive.

```
public boolean isWifiEnabled() {
    return wifiManager.isWifiEnabled();
}
```

Listing 13.19: Start scanning for available Wi-Fi networks

13.2.4. Local settings

Every Android application can store variables beyond the time of the application actually running using SharedPreferences.

SharedPreferences allow the application to store data in a key value structure that is not volatile and are the data is stored on the phone until they are either changed by the application or the application is being deinstalled.

In our application, the SharedPreferences were used to store data like:

- Which ways of alerting the user using the smartphone should be used...
- ...if the application was started before?
- ...if the alerts of the AudioAnt were already learned?
- What are the access data of the local Wi-Fi?

Using the String "FILE_KEY" and the mode "Context.MODE_PRIVATE", we get a SharePreferences object that can only be accessed from this application using this key, ensuring the saved data cannot be read by any other application, or the user using the file explorer.

getUseVibration() returns a boolean value based on the outcome of *sharedPref.getBoolean(vibrationKey,false)*, with "false" being the value that is returned if no value is saved under this key.

When editing existing values, a "SharedPreferences.Editor" has to be created first. After the changes are made, *editor.commit()* commits the changes from the editor.

```
public Settings(Context context) {
    sharedPref = context.getSharedPreferences(FILE_KEY, Context.MODE_PRIVATE);
}

public boolean getUseVibration() {
    return sharedPref.getBoolean(vibrationKey, false);
}

public void setUseVibration(boolean useVibration) {
    SharedPreferences.Editor editor = sharedPref.edit();
    editor.putBoolean(vibrationKey, useVibration);
    editor.commit();
}
```

Listing 13.20: Example methods of the SharedPreferences helper class Settings.java

13.2.5. AudioAnt alerts

On the smartphone, the user can choose the notification sound played, when a sound is recognised, from a list of standard sounds the smartphone provides. These sounds may differ between different Android devices.

The sounds the AudioAnt can emit are, however, constant and don't change according to the device the application runs on. In order to be able to choose which sound should be used on

the AudioAnt in a comfortable and user friendly way, the application should be able to play the sounds saved on the AudioAnt when choosing which one to use.

For doing so, the application requests the alerts stored on the AudioAnt with the first start of the application and stores the them on the smartphone.

The AlertSoundHelper.java class provides access to saving, playing and deleting the alert sounds that can be chosen to be used as alert sound on the AudioAnt.

playAlertFromName()
When choosing which alert to use on the AudioAnt, a click on the alert name in the list provided calls this method so that the user knows how it sounds.

Because the name of the file which the alert is saved in on the smartphone is a combination of the name of the sound, its ID and its original file name on the AudioAnt, the name is split. Then, only the first part, the actual name of the sound(for example "Bleep"), is compared to the name passed when calling the method and the sound is being played if they are equal.

```
public void playAlertFromName(String name) {
    File folder = new File(new File(Environment.getExternalStorageDirectory(), ".AudioAnt"), "
    Alerts");

//  Get all learned sounds in directory
    File[] files = folder.listFiles();
    for (int i = 0; i < files.length; i++) {
//      Check if this is the sound that should be played (by name)
        String[] items = files[i].getName().split("-");
        if (items[1].equals(name)) {
            try {
                if (player != null) {
                    player.release();
                    player = null;
                }
//              start playing the sound
                player = new MediaPlayer();
                player.setDataSource(files[i].getAbsolutePath());
                player.prepare();
                player.start();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

}
```

Listing 13.21: playAlertFromName() method of the AudioAnt alert helper class AlertSoundHelper.java

saveAlert()
This method saves the Base64 encoded string, containing the audio content of the file, the AudioAnt sent to the application inside a JSON file, to a pre-specified location on the smartphone.

readAlerts()
This method adds the names of the sounds that are saved on the phone to the ArrayAdapter that is passed when the method is called.

deleteRecursively()

This method uses the principle of recursion, meaning that the method calls itself, to first delete all files in a directory and then the directory itself. This method is invoked before saving the received alerts from the AudioAnt to make sure the directory is empty.

```
public void saveAlert(String fileName, String fileContent) {
//  The directory is created
    File dir = new File(new File(Environment.getExternalStorageDirectory(), ".AudioAnt"), "
    Alerts");
    dir.mkdirs();
//  The file the alert should be saved in is created
    File f = new File(dir, fileName);

//  Deletes the alert if it already exists
    if (f.exists()) {
        f.delete();
    }
    try {
        f.createNewFile();
    } catch (IOException e) {
        e.printStackTrace();
    }
//  The audio data of the file is converted to a byte array
    byte[] data = Base64.decode(fileContent, Base64.NO_WRAP);
    try {
//      The byte array is written to the file
        FileOutputStream fos = new FileOutputStream(f.getAbsoluteFile());
        fos.write(data);
        fos.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Listing 13.22: saveAlert() method of the AudioAnt alert helper class AlertSoundHelper.java

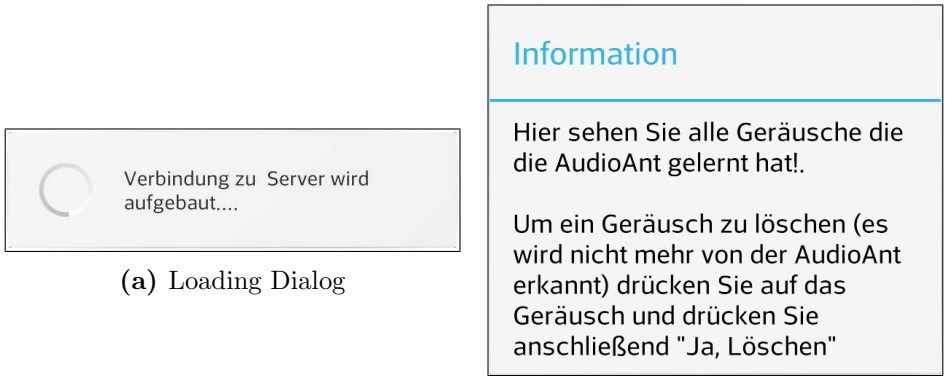
```
public void readAlerts(ArrayAdapter<SoundListItem> arrayAdapterSounds) {
//  Get the directory
    File folder = new File(new File(Environment.getExternalStorageDirectory(), ".AudioAnt"), "
    Alerts");
//  Get all files in that directory
    File[] files = folder.listFiles();
    if (files != null) {
        for (int i = 0; i < files.length; i++) {
//          add the file name to the ArrayAdapter passed at method call
            String[] items = files[i].getName().split("-");
            int number = Integer.parseInt(items[0]);
            String name = items[1];
            arrayAdapterSounds.add(new SoundListItem(name, number));
        }
    }
}
```

Listing 13.23: readAlerts() method of the AudioAnt alert helper class AlertSoundHelper.java

```
public void deleteRecursive(File fileOrDirectory) {
//  Deletes a directory and all its files via recursive call
    if (fileOrDirectory.isDirectory())
        for(int i = 0;i<fileOrDirectory.listFiles().length;i++){
            deleteRecursive(fileOrDirectory.listFiles()[i]);
        }
    fileOrDirectory.delete();
}
```

Listing 13.24: deleteRecursively() method of the AudioAnt alert helper class AlertSoundHelper.java

13.2.6. Dialogs



(a) Loading Dialog

(b) Text alert

Figure 13.4.: Example of Dialogs

Our application uses alerts to inform the user about events, for example that the connection could not be established, to display information when an info-button is clicked and to show loading animations if the activity is waiting for data from the AudioAnt or for the service to connect.

ProgressDialog
We used ProgressDialogs to show the user that a process is currently being executed. This is, for example, when trying to connect to the AudioAnt or when fetching data from the AudioAnt to display. It also prevents users from pressing elements of the activity that should not yet be accessed. When creating it, *dialog.setCancelable(false)* can be used to show the Dialog, until it is dismissed by the application.

```
    public void showLoadingDialog() {
        loadingDialog = new ProgressDialog(MainActivity.this);
        loadingDialog.setMessage("Verbindung zu  Server wird aufgebaut....");
        loadingDialog.setCancelable(false);
        loadingDialog.show();
    }
```

Listing 13.25: Example of a loading dialog

ShowTextAlert
This kind of Dialog gets called when one of the info buttons which the application provides at numerous places is pressed. It simply displays text to the user and can either be dismissed by clicking next to it, or by using the provided button, if available.

```
//  Methods for showing dialogs
public void showListHelp(View v) {
    ShowTextAlert textAlert = new ShowTextAlert();
    textAlert.setText(getResources().getString(R.string.list_description));
    textAlert.setHeader(getResources().getString(R.string.list_description_header));
    textAlert.show(getFragmentManager(), "notifications help");
}
```

Listing 13.26: Example of a ShowTextAlert

AlertDialog
This Dialog is being displayed, whenever the user's input is needed, or the user should confirm

that the message was read. This could, for example, be if a sound should be deleted from the learned sounds of the AudioAnt.

AlertDialogs can have multiple buttons and all these buttons have their own onClickListener. Therefore, the AlertDialog can be configured to trigger different events as needed.

```
public void showNotconnectedToWifiFirstStart() {
    firstStartDialog = new AlertDialog.Builder(MainActivity.this);
    firstStartDialog.setTitle("Herzlich willkommen!");
    firstStartDialog.setMessage(getResources().getString(R.string.main_message_first_usage));
    firstStartDialog.setNegativeButton("Abbrechen", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
        }
    });
    firstStartDialog.setPositiveButton("Verbinden", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
            showConnectingToNetworkDialog();
            Thread t = new Thread(new Runnable() {
                @Override
                public void run() {
                    if (!wifiHelper.isWifiEnabled()) {
                        wifiHelper.setWifiEnabled(true);
                    }
                    wifiHelper.connectToWifi(constants.AA_HOTSPOT_NAME, constants.
    AA_HOTSPOT_PW);
                    registerReceiver(wifiConnectedReceiver, new IntentFilter(WifiManager.
    NETWORK_STATE_CHANGED_ACTION));
                }
            });
            t.run();
        }
    });
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            firstStartDialog.show();
        }
    });
}
```

Listing 13.27: Example of an AlertDialog

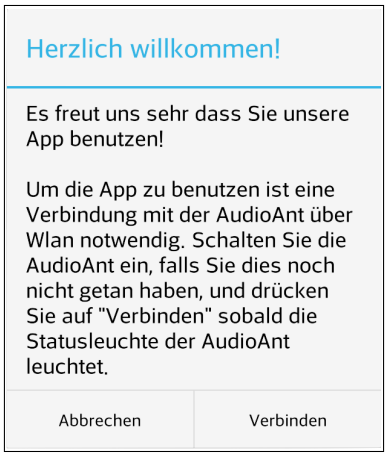


Figure 13.5.: Example of an AlertDialog

13.2.7. MainActivity.java

This activity is the first activity the user sees when opening up the application. It is responsible for triggering the connecting process to the AudioAnt, explaining the meaning of the application if it is the first start, and providing navigation elements to other activities of the application.

13.2.7.1. Starting the activity

When the activity gets started, the *onCreate()* method initialises instances of necessary helper classes and Views so that their content can be changed via code. It also registers a local BroadcastReceiver that receives an Intent from the CommunicationService as soon as the connection to the AudioAnt is established.

If a local broadcast is received, the views are updated, now saying that the AudioAnt is connected, and a boolean variable, that indicates if a connection available or not, is set to true.

A view that is created in the layout XML file can be instantiated using the method *findViewById(#id of the view)* whose outcome can be cast to any type of view, such as TextView or Button, for example.

```
imageView_verbunden = (ImageView) findViewById(R.id.ic_verbunden);
```

Listing 13.28: Initialising a view object in MainActivity.java

Connecting to the AudioAnt
If the CommunicationService is not yet connected to the AudioAnt, a loading animation is shown and a timer is started (listing 13.30, line 23). This timer checks if the connection to the AudioAnt was successfully established after one minute and shows an error-Dialog to the user if this is not the case.

13.2.7.2. Dialogs that can be shown

If the smartphone is not connected to any Wi-Fi network when the activity launches, an AlertDialog is shown that allows the user to request the application to connect to the AudioAnt's hotspot. If it is the first start of the application, the Dialog also displays a welcoming message to the user (see figure 13.5).

If the smartphone is not connected to any network but the application has been started before and a local Wi-Fi has already been configured, the application automatically tries to connect to said network as soon as the "Verbinden" button is pressed.

While the activity has not yet received the broadcast of a successfully established connection to the AudioAnt, a ProgressAlert is being shown (see figure 13.4a).

13.2.7.3. Navigation

Because this is the "home" activity of the application, it provides access to all activities that can be entered on user command, such as RecordActivity, ListActivity, AudioAntSettingsActivity and SettingsActivity. Since the press of the smartphone's back-button should always return the user to this activity, no flags for creating a new history stack or clearing the associated tasks are needed.

```
public void startRecordActivity(View v) {
    startActivity(new Intent(MainActivity.this, RecordActivity.class));
}
```

Listing 13.29: Starting a new activity from MainActivity.java

```
private ServiceConnection serviceConnection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
//      Use the IBinder service to get the running instance of the CommunicationService
        CommunicationService.MyBinder binder = (CommunicationService.MyBinder) service;
        communicationService = binder.getService();
//      Check if wifi is enabled
        if (!wifiHelper.isConnected()) {
            new Thread(new Runnable() {
                @Override
                public void run() {
                    showNotConnectedToWifiDialog();
                }
            }).start();
        } else {
//          Set symbols and lettering depending on connection state
            if (communicationService.isConnected()) {
                imageView_verbunden.setImageDrawable(getResources().getDrawable(R.drawable.
    ic_done_24dp));
                textView_audioant_connection_status.setText(getResources().getString(R.string.
    main_label_connected));
            } else {
                imageView_verbunden.setImageDrawable(getResources().getDrawable(R.drawable.
    ic_wifi_not_connected_24dp));
                textView_audioant_connection_status.setText(getResources().getString(R.string.
    main_label_not_connected));
                showLoadingDialog();
                serverConnectionTimer.start();
            }
//          measure how long it took to bind the service
            stop = System.currentTimeMillis();
            Log.d(TAG, "" + (stop - start));
            serviceIsBound = true;
        }
    }
    @Override
    public void onServiceDisconnected(ComponentName name) {
        serviceIsBound = false;
    }
};
```

Listing 13.30: Initialising the ServiceConnection in MainActivity.java

13.2.8. RecordActivity

RecordActivity.java is used for adding sounds that should be recognised to the AudioAnt by first recording a sound, giving it a name, and then sending it to the AudioAnt.

13.2.8.1. Starting the activity

Again, all necessary View elements are instantiated in the *onCreate()* method. In order to make sure the "Senden" button only gets enabled if a sound is recorded and has a name, a TextChangedListener is added to the EditText where the name of the sound can be entered.

```
geraeuschName.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        if (!geraeuschName.getText().equals("") && geraeuschSchonAufgenommen)
            button_save.setEnabled(true);
        else button_save.setEnabled(false);
    }

    @Override
    public void afterTextChanged(Editable s) {
    }
});
```

Listing 13.31: Adding a TextChangedListener to the EditText "geräuschName"

In the *onServiceConnected()* method, alert sounds are requested from the AudioAnt if they have not been already stored. Because the user can only navigate to this activity if a connection to the AudioAnt is already established, no connecting to the AudioAnt process, and therefore no ProgressDialog, is necessary.

To listen for the data the AudioAnt sends after receiving this JSON object, and after successfully analysing the received audio data from the smartphone, two local BroadcastReceivers are registered.

```
public void requestSoundsIfFirstStart() {
    settings = new Settings(this);
    if (!settings.getSoundsLoaded()) {
        try {
//          create a json object
            JSONObject jsonObject = new JSONObject();
            jsonObject.put("action", "getAlertSounds");
//          send it to the AudioAnt
            communicationService.sendToServer(jsonObject.toString());
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

Listing 13.32: Request sounds from AudioAnt if necessary

```
LocalBroadcastManager.getInstance(this).registerReceiver(soundLearnedFeedbackReceiver, new
    IntentFilter("soundLearnedFeedback"));
LocalBroadcastManager.getInstance(this).registerReceiver(getAllAlertSounds, new IntentFilter("
    alertSounds"));
```

Listing 13.33: Registering local BroadcastReceivers

13.2.8.2. Recording and replaying audio

Recording audio
Once the "Aufnahme starten" button is pressed, the RecordService.java is started and handles the recording process. At the same time, the thread for updating the TextView that shows the recording duration is started. If the button is pressed again, the RecordingService and the thread for updating the TextView are stopped.

Replaying audio
When replaying the recorded sound, a MediaPlayer is used, just as in listing 13.21. Also, the thread for updating the TextView showing the recording time is started again.

The source for the MediaPlayer is set statically in this case, because there always is only one recorded sound stored on the device that is overridden if a new one is being recorded.

13.2.8.3. Transform and sending the audio file

When the "Geräusch speichern" button is pressed, the recorded sound is packed inside a JSON file and sent to the AudioAnt. In addition, a ProgressDialog is shown until the AudioAnt reports either successfully or unsuccessfully learning the sound.

```
public String fileToString(File file) {
//  Converts the audio File to a byte array
    int size = (int) file.length();
    byte[] bytes = new byte[size];
    try {
        FileInputStream fs = new FileInputStream(file);
        fs.read(bytes);
    } catch (FileNotFoundException e) {
        Log.d(TAG, "Die Datei zum umwandeln existiert nicht");
    } catch (IOException e) {
        Log.d(TAG, "Die Datei konnte nicht als byte[] gespeichert werden");
    }
//  Decodes the byteArray into a String and returns it
    return Base64.encodeToString(bytes, Base64.NO_WRAP);
}

public void sendButtonClicked(View v) {

    showWaitingForReplyInfo();
//  Create  a JSON object to send to the AudioAnt
    JSONObject object = new JSONObject();
    JSONObject data = new JSONObject();
    try {
        data.put("fileContent", fileToString(new File(getAbsoluteFileLocation())));
        data.put("name", geraeuschName.getText());
        data.put("fileExtension", ".mp3");
        data.put("alertId", chosenAlertNumber);

        object.put("action", "saveSound");
        object.put("data", data);
    } catch (JSONException e) {
        e.printStackTrace();
    }
//  send the object to the AudioAnt
    communicationService.sendToServer(object.toString());
}
```

Listing 13.34: Converting a file into a String

13.2.9. ListActivity

This activity displays a list of all the sounds that have been learned up to this point. When clicked, the user has the possibility to delete the selected sound.

13.2.9.1. Starting the activity

On activity start, the activity displays a ProgressDialog and registers two local BroadcastReceivers, one for receiving a list of sounds, the other for receiving the confirmation of a sound being deleted successfully. As soon as the CommunicationService is bound and an instance of it is available, a JSON file is sent to the AudioAnt, requesting a list of sounds the AudioAnt has learned and can recognise.

13.2.9.2. Receiving broadcasts

```java
public void handleListOfSoundsReceived(Context context, Intent intent) {
    try {
        listOfSounds = (ListView) findViewById(R.id.list); // initialise list of sounds
        adapter = new ArrayAdapter<SoundListItem>(ListActivity.this, android.R.layout.simple_expandable_list_item_1, listItems);
        listOfSounds.setAdapter(adapter);
        listOfSounds.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override // show AlertDialog when clicked to confirm
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                final SoundListItem selectedItem = listItems.get(position);
                AlertDialog.Builder alert = new AlertDialog.Builder(ListActivity.this);
                alert.setTitle("Ton loeschen");
                alert.setMessage("Wollen sie Das Geraeusch \"" + selectedItem.getName() + "\" wirklich loeschen?");
                alert.setNegativeButton("Nein", null);
                alert.setPositiveButton("Ja, loeschen", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        showWaitDialog();
                        sendDeleteSoundJson(selectedItem.getNumber());
                        requestSounds();
                    }
                });
                alert.show();
            }
        });
        JSONObject j = new JSONObject(intent.getStringExtra("json"));
        JSONObject data = j.getJSONObject("data");
        JSONArray sounds = data.getJSONArray("sounds");
        listItems.removeAll(listItems);
        for (int i = 0; i < sounds.length(); i++) { // add all received sounds to the list
            JSONObject sound = sounds.getJSONObject(i);
            SoundListItem item = new SoundListItem(sound.getString("name"), sound.getInt("number"));
            listItems.add(item);
        }
        waitForDataDialog.dismiss();
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Listing 13.35: Receive the learned sounds of the AudioAnt

When the BroadcastReceiver receives the broadcast with the list of sounds, the *handleListOfSoundsReceived()* method is executed.

When the confirmation broadcast of deleting a sound is received, the ProgressDialog that is shown when deleting a sound is dismissed, and the new list that was requested during the *handleListOfSoundsReceived()* method, gets displayed.

```
private void requestSounds() {
    try {
        JSONObject object = new JSONObject();
        object.put("action", "getListOfSounds");
        communicationService.sendToServer(object.toString());
        Log.d(TAG, "list of sounds was requested");
    } catch (JSONException e) {
        Log.d(TAG, "Json could not be created");
    }
}
```

Listing 13.36: Request the learned sounds of the AudioAnt

13.2.10. AudioAntSettings activity

This activity is used for configuring the way the AudioAnt should alert the user and for sending the access-data of a local Wi-Fi to the AudioAnt after checking them.

13.2.10.1. Starting the activity

When the activity gets created, the view and variables are initialised, a ProgressDialog is shown, and the necessary BroadcastReceivers are registered. The ProgressDialog is needed, because the application has to request the current configurations from the AudioAnt before being able to display and later edit it. Once the CommunicationService is bound and the *onServiceConnected()* method of the ServiceConnection is called, a JSON is sent to the AudioAnt requesting the current settings. The answer to this request is received by the according BroadcastReceiver and the view elements are initialised with the received values.

```
public void handleSettingsReceived(Context context, Intent intent) {
    loadSettingsDialog.dismiss();
//  assigns values from JSON file to views
    try {
        JSONObject obj = new JSONObject(intent.getStringExtra("json"));
        JSONObject data = obj.getJSONObject("data");

        boolean lightSignals = data.getBoolean("lightSignals");
        boolean audioSignals = data.getBoolean("audioSignals");
        alertSoundId = data.getInt("alertSoundId");

        checkBox_audioSignals.setChecked(audioSignals);
        checkBox_lightSignals.setChecked(lightSignals);

    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Listing 13.37: Initialise View elements with received data

Also, a Wi-Fi scan for available networks is started, so that the user can later choose the local network the AudioAnt should be connected to from a list inside the application.

```
private void fetchData() {
    try {
        JSONObject object = new JSONObject();
        object.put("action", "getCurrentSettings");
        communicationService.sendToServer(object.toString());
        Log.d(TAG, "AudioAnt settings were requested");
    } catch (JSONException e) {
        Log.d(TAG, "Json could not be created");
    }
}
```

Listing 13.38: Requesting data from the AudioAnt by sending a JSON file

13.2.10.2. Choosing a default alert tone

Just like in the RecordActivity, the AudioAntSettings.java activity checks if the alert sounds of the AudioAnt have already been learned and requests them if not. The way the alert is then chosen is also exactly the same, for more information see listing 13.32.

13.2.10.3. Connecting to a local network

Entering network data
When wanting to connect to a local Wi-Fi network, the user can enter the name of the network and its password in the EditText boxes provided. However, since Wi-Fi SSIDs can be pretty complicated at times, it is also possible to choose the network from a list that can be opened by pressing the "Netzwerk aus Liste auswählen" button.

The list of available networks gets available as soon as the responsible BroadcastReceiver receives the data from the system-wide Broadcast. Once recognised, the *handleWifiNetworksReceived* method is called to update the arrayAdapter of the list that should be shown accordingly.

The AlertDialog.Builder is used to construct an AlertDialog with the listView of sounds set as View. When an item is clicked, its name gets saved in the variable "chosenNetwork" and gets set as the text of the EditText wlanName when the "Ok" button is clicked.

Connecting to the network on the smartphone
The application verifies the correctness of the input by trying to connect to the Wi-Fi network with this name and this password. If the connection succeeds, a variable is set to true and the application immediately reconnects to the hotspot of the AudioAnt.

A BroadcastReceiver that checks if the Wi-Fi connection is changed can pick up if the Wi-Fi is reconnected and check if the now connected network is the one specified by the user.

If the application connects to the local network and back to the hotspot of the AudioAnt before the timer, started on click of the "Wlan informationen senden" button, runs out, the access-data of the local network are sent to the AudioAnt. If the application fails to connect to the network,

most likely because either name or password were incorrect, the connection to the AudioAnt is re-established and an AlertDialog is shown to inform the user.

The *connectToLocalNetwork()* methods re-initialises the connection process of the CommunicationService, this time however using the name and password of the local Wi-Fi network.

```
public void chooseWifiFromListDialog() {
  final AlertDialog.Builder chooseWifi = new AlertDialog.Builder(AudioAntSettings.this);
  chooseWifi.setTitle(getResources().getString(R.string.choose_wifi_header));
  chooseWifi.setNegativeButton("Abbrechen", new DialogInterface.OnClickListener() {
              @Override
              public void onClick(DialogInterface dialog, int which) {
                  dialog.dismiss();
              }
          }
  );
  chooseWifi.setPositiveButton("Fertig", new DialogInterface.OnClickListener() {
      @Override
      public void onClick(DialogInterface dialog, int which) {
          wlanName.setText(chosenNetwork);
      }
  });
  final ListView listViewNetworks = new ListView(this);
  listViewNetworks.setChoiceMode(AbsListView.CHOICE_MODE_SINGLE);
  listViewNetworks.setAdapter(arrayAdapterNetworks);
  listViewNetworks.setOnItemClickListener(new AdapterView.OnItemClickListener() {
      @Override
      public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
          listViewNetworks.setSelection(position);
          chosenNetwork = arrayAdapterNetworks.getItem(position);
      }
  });
  chooseWifi.setView(listViewNetworks); chooseWifi.show();
}
```

Listing 13.39: Choose a Wi-Fi network from the AlertDialog

```
public void handleWifiReconnectReceived(Context context, Intent intent) {
    loadingDialog.dismiss();
//  check if the application has been connected to a local network
    if (connectedToNewNetwork) {
        sendConnectToLocalWifiJson();
        settings.setLocalWifiName("" + wlanName.getText());
        settings.setLocalwifiPW("" + wlanPassword.getText());
        connectToLocalNetwork();
        readyForResult = false;
    } else {
//      show alert if the connection to the local network failed
        final AlertDialog.Builder alert = new AlertDialog.Builder(AudioAntSettings.this);
        alert.setTitle("Fehler!");
        alert.setMessage(getResources().getString(R.string.message_connect_error));
        alert.setPositiveButton("OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        });
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                alert.show();
            }
        });
    }
}
```

Listing 13.40: Method invoked after reconnecting to the AudioAnt's hotspot

```
public void handleWifiChangedReceived(Context context, Intent intent) {
    final Intent i = intent;
    Thread t = new Thread(new Runnable() {
        @Override
        public void run() {
            NetworkInfo networkInfo = i.getParcelableExtra(WifiManager.EXTRA_NETWORK_INFO);
            if (networkInfo != null) {
//              is a network connected
                if (networkInfo.isConnected()) {
//                  check if the wished local network is connected
                    String ssid = wifiHelper.getCurrentNetworkSSID();
                    if (ssid.equals(wlanName.getText() + "")) {
//                      check if the connection process is finished
                        if (wifiHelper.getSupplicantState().equals(SupplicantState.COMPLETED)) {
                            Log.d(TAG, "conneted to new network");
                            connectedToNewNetwork = true;
                            readyForResult = true;
                            reconnectToHotspot();
                        }
//                  check if the AudioAnt hotspot is connected
                    } else if (ssid.equals(new Constants().AA_HOTSPOT_NAME)) {
                        if (wifiHelper.getSupplicantState().equals(SupplicantState.COMPLETED)) {
                            if (readyForResult) {
                                Log.d(TAG, "connected to AudioAnt Hotspot");
                                initialiseConnectionToAudioAnt();
                            }
                        }
                    }
                }
            }
        }
    });
    t.start();
}
```

Listing 13.41: choose a Wi-Fi network from the AlertDialog

```
public void handleWifiNetworksRecieved(Context context, Intent intent) {
//  add the outcome of the scan for wifi networks
//  to the arrayAdapterNetworks
    if (intent.getAction().equals(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION)) {
        List<ScanResult> scanResults = wifiHelper.getScanResults();
        arrayAdapterNetworks.clear();
        for (int i = 0; i < scanResults.size(); i++) {
            arrayAdapterNetworks.add(scanResults.get(i).SSID);
        }
    }
}
```

Listing 13.42: Save the received Wi-Fi networks to the arrayAdapter

13.2.10.4. Updating notification settings

Whenever the activity is left, either using the back-button of the smartphone or via the "Home-As-Up" button in the top left corner, a JSON file with the boolean values of using flash and audio to alert the user as well as the default ringtone are sent to the AudioAnt.

13.2.11. SettingsActivity

This activity only configures how the smartphone should react when a sound is recognised by the AudioAnt. Therefore, the CommunicationService does not have to be bound since no data have to be exchanged.

13.2.11.1. Starting the activity

In the classes *onCreate()* method, the View objects get instantiated and the values from SharedPreferences, accessed using the Settings.java helper class, are assigned to them.

13.2.11.2. Changing settings

Whenever changes to the CheckBoxes are made, they are immediately saved in the local settings. For choosing the notification sound on the smartphone, an intent with the action "RingtoneManager.ACTION_RINGTONE_PICKER" can be used to select one of the standard ringtones provided by the operating system.

```
public void showNotificationDialog(View v) {
    Intent intent = new Intent(RingtoneManager.ACTION_RINGTONE_PICKER);
    intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TYPE, RingtoneManager.TYPE_NOTIFICATION);
    intent.putExtra(RingtoneManager.EXTRA_RINGTONE_TITLE, "Benachrichtigungston auswaehlen");
    intent.putExtra(RingtoneManager.EXTRA_RINGTONE_EXISTING_URI, settings.getCurrentSound());

    this.startActivityForResult(intent, 1);
}
```

Listing 13.43: Alert for choosing a notification sound

For retrieving the chosen sound of the AlertDialog, the *onActivityResult()* method has to be overridden. The Uri that can be retrieved from the Intent gets saved in the SharedPreferences and can later be used to replay the sound.

```
@Override
protected void onActivityResult(final int requestCode, final int resultCode, final Intent
    intent) {
    if (resultCode == Activity.RESULT_OK && requestCode == 1) {
        Uri uri = intent.getParcelableExtra(RingtoneManager.EXTRA_RINGTONE_PICKED_URI);
        if (uri != null) {
            settings.setSound(uri);
            setChosenSoundToTextView();
        }
    }
}
```

Listing 13.44: Overridden onActivityResult() method

13.2.12. AlarmActivity

This activity only gets called from the CommunicationService and uses all enabled ways of notifying the user until the "Schließen" button is pressed.

13.2.12.1. Starting the activity

When the activity is started, the extra of the intent, used to navigate to this activity, is used to call the *createSoundNameTextViews(JSONArray sounds)* method. The recognised sounds are shown in a LinearLayout as TextViews.

13.2.12.2. Deactivating the notifications

The notifications can be either disabled by pressing the "Schließen" button, or by pressing the corresponding button on the AudioAnt, in which case a JSON file is sent to the AudioAnt. Either way, the method *stopAlarmingMethods()* is called that stops all methods of notifying the user and exits the application.

```
private void createSoundNameTextViews(JSONArray sounds) throws JSONException {
//  do this for all recognised sounds
    for (int i = 0; i < sounds.length(); i++) {
        JSONObject alarm = sounds.getJSONObject(i);
//      parameter for positioning the TextView
        LinearLayout.LayoutParams layoutParams = new LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.WRAP_CONTENT,
                LinearLayout.LayoutParams.WRAP_CONTENT);
        layoutParams.gravity = Gravity.CENTER;
//      creating the TextView
        TextView textView = new TextView(this);
        textView.setGravity(View.TEXT_ALIGNMENT_CENTER);
        textView.setText(alarm.getString("name")+" (Nummer: " + alarm.getInt("number") + ")");
        textView.setTextSize(getResources().getDimension(R.dimen.text_size_xlarge));
        textView.setTextAppearance(this, android.R.style.TextAppearance_Large);
        textView.setLayoutParams(layoutParams);
//      adding the TextView to the layout at index 0
        layout.addView(textView, 0);
    }
}
```

Listing 13.45: Method for initialising the sound name TextViews

```
public void stopAlarmingMethods() {
    try {
        continueFlash = false;
        continueSound = false;
        continueVibration = false;
//      wait if the button is pressed before the service could be bound
        while (!serviceIsBound) {
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
//      notify the AudioAnt so the alarm doesn't have to be stopped twice
        sendHandledJson();
//      exit the application
        finish();
        android.os.Process.killProcess(android.os.Process.myPid());
        AlarmActivity.super.onDestroy();
    } catch (NullPointerException e) {
        e.printStackTrace();
    }
}
```

Listing 13.46: Method for stopping the alerts and closing the activity

14. Calculation

The implementation of a calculation was important to find out how much the production costs of the product are. We decided to conduct a double-stage overhead calculation due to simplicity and more accuracy compared to the other accounting methods. The first step is the compilation of a list of the used material. The resulting costs are called the material costs.

Label	Quantity	Gross price	Net price	VAT	Total
Qi Transmitter	1	9,99 €	8,33 €	20%	8,33 €
Power supply	1	27,99 €	23,52 €	19%	23,52 €
Jack socket	1	1,92 €	1,60 €	20%	1,60 €
Raspberry Pi 2	1	42,92 €	36,07 €	19%	36,07 €
Display	1	2,69 €	2,24 €	20%	2,24 €
Casing	1	4,90 €	4,12 €	19%	4,12 €
USB-AUX connection	2	12,9 €	10,84 €	19%	21,68 €
Soundcard	1	6,25 €	5,25 €	19%	5,25 €
Relay	1	7,09 €	5,91 €	20%	5,91 €
Speaker	1	33,25 €	27,94 €	19%	27,94 €
Microphone	1	8,99 €	7,49 €	20%	7,49 €
WLAN-stick	1	7,99 €	6,71 €	19%	6,71 €
Light source	1	26,31 €	21,93 €	20%	21,93 €
Buttons, LEDs, wires, resistors, shrinking tubes, solder	1	5 €	4,17 €	20%	4,17 €
Total					**176,95 €**

Table 14.1.: Material costs

Clearly the VAT has to be subtracted from the purchase price. The VAT-value differs due to some material being bought in Germany (19% VAT) and some material being bought in Austria (20% VAT). After compiling the material costs, the production labour costs have to be calculated. The process of producing the prototype took about 15 hours. Nevertheless, it can be assumed that it takes about ten hours for the production of one product. Furthermore, we calculated with a hourly rate of 35 €.

Label	Quantity	Price	Total
Man hour	10	35,00 €	350,00 €

Table 14.2.: Manufacturing costs

Within a double-stage overhead calculation, two overheads rates have to be included. Due to the fact that we do not have a cost distribution sheet, we can only calculate with realistic values instead of calculated overhead rates based on the cost distribution sheet.

Cost type	**Costs in €/piece**
Material costs	176,95 €
Material overheads (10%)	17,70 €
Production labur costs	350,00 €
Factory overheads (20%)	70,00 €
Production costs	614,65 €

Table 14.3.: Overheads

The production costs for one product are at an amount of 614,65 €. To get an appropriate offer price for a product, it would be necessary to add cash discount and profit to the production costs. The conducted calculation is only a prototype-calculation. For the production of high quantities, the price for one device would be much lower. The double-stage overhead calculation and the resulting production costs do not say anything about how profitable the business would be.

Appendix

A. Time Protocol

A.1. Benjamin Hackstock

Date	Description	Time spent
2015-03-06	First informational meeting with Prof. Frischmann	0h 30m
2015-03-08	Online data management setup	1h 00m
2015-06-02	Research wireless charging	1h 30m
2015-06-03	Meeting and discussion with Prof. Frischmann	1h 00m
2015-06-12	Group discussion: hardware incl. scoring model, functional specification	1h 00m
2015-06-16	Elaborating functional specifications	1h 00m
2015-06-17	Meeting and discussion with Prof. Frischmann	0h 40m
2015-06-19	Brainstorming on project title	0h 45m
2015-06-30	Elaborating survey questions	0h 35m
2015-08-07	Elaborating survey questions	1h 30m
2015-07-09	Meeting with Prof. Frischmann and Prof. Greinöcker	1h 00m
2015-08-16	Group meeting	1h 30m
2015-09-09	Meeting with Prof. Frischmann and Prof. Greinöcker	1h 30m
2015-09-14	Android Studio setup	0h 45m
2015-09-16	Android Studio setup and research	0h 45m
2015-09-19	Questionary update, Android Studio research	2h 20m
2015-09-22	Research: Android layouts	0h 30m
2015-09-23	Research: Android gestures	1h 15m
2015-09-26	Research: Android fragments	3h 15m
2015-09-27	Research: Android threads	1h 30m
2015-09-28	Research: LaTeX, Android services	2h 45m
2015-09-29	Corrected: Patent theory	0h 30m
2015-09-29	Research: Android usability and design	1h 30m
2015-09-30	Meeting with Prof. Frischmann and Prof. Greinöcker	0h 20m
2015-09-30	Research: Android basics	0h 45m
2015-09-30	Research: prior diploma theses	1h 15m
2015-10-01	Usability research and theory writing	1h 30m
2015-10-03	Usability research and theory writing	0h 30m
2015-10-05	Usability research and theory writing	0h 40m
2015-10-07	Discussion with Prof. Greinoecker on usability and survey	1h 20m
2015-10-08	Questionary update	0h 40m
2015-10-11	Online Questionary writing	2h 40m
2015-10-13	Questioning volunteers	0h 45m

2015-10-16	Android coding	2h 00m
2015-10-22	Usability research and theory writing	0h 30m
2015-10-25	Usability research and theory writing	2h 30m
2015-10-26	Usability research and theory writing	1h 20m
2015-10-27	Usability research and theory writing	3h 00m
2015-11-02	Operational concept writing	1h 00m
2015-11-03	Operational concept writing	1h 00m
2015-11-03	Android coding and research	1h 00m
2015-11-05	Operational concept writing	1h 00m
2015-11-07	Research: offline speech recognition, operational concept edit	1h 30m
2015-11-08	Research: offline speech recognition, speech recognition writing	1h 15m
2015-11-09	Android coding and research	1h 00m
2015-11-11	Android coding and research	2h 00m
2015-11-12	Android coding and research - material design	2h 00m
2015-11-13	Android coding and research - floating action button	1h 30m
2015-11-15	Android coding and research - fragment integration	1h 00m
2015-11-17	Android coding and research - tab swipe layout	2h 00m
2015-11-20	Android coding and research - recyclerView	1h 00m
2015-11-21	Anrdoid coding and research - images for results	3h 00m
2015-11-22	Android coding and research - intents with extra data	1h 00m
2015-11-24	Android coding and research - intents with extra data	2h 00m
2015-11-25	Android coding and research - splash screen, FragmentAdapter	3h 00m
2015-12-01	Creating operational concept graph	0h 30m
2015-12-02	Android coding and research - StackOutOfMemory	1h 15m
2015-12-03	Android coding and research - intent extras	0h 50m
2015-12-09	Creating operational concept graph	1h 00m
2015-12-18	Writing proposal for Jugend Innovativ	1h 00m
2016-01-09	Android coding and research - recording service	6h 00m
2016-01-10	Android coding and research - audio	1h 00m
2016-01-12	Android research - sockets	0h 30m
2016-01-13	Android research - sockets	1h 00m
2016-01-14	Android coding - sockets	1h 00m
2016-01-18	Correcting thesis: patent search	0h 45m
2016-01-21	Android research - streams	1h 15m
2016-01-21	BeTheBest' - application preparations	1h 25m
2016-01-22	BeTheBest' - application writing	0h 40m
2016-01-23	BeTheBest' - motivation paper writing	0h 50m
2016-01-23	Eurac - project report writing	1h 40m
2016-01-24	Eurac - project report writing	1h 35m
2016-01-25	Eurac - project report writing	2h 05m
2016-01-26	Eurac - project report writing	0h 35m
2016-01-28	Eurac - project report writing	1h 15m
2016-01-29	Eurac - project english summary writing	1h 40m
2016-01-31	BeTheBest - writing and application submission	0h 45m
2016-01-02	Usability research and theory writing	1h 25m

2016-02-02	Usability research and theory writing	1h 20m
2016-02-04	Usability research and theory writing	0h 45m
2016-02-10	Audio introduction writing and usability research	3h 35m
2016-02-11	Introduction writing	2h 00m
2016-02-11	Usability research and theory writing	3h 30m
2016-02-11	Correcting thesis: chapter 4	1h 00m
2016-02-12	Usability research and theory writing	1h 30m
2016-02-13	Usability research and theory writing	2h 00m
2016-02-13	Correcting thesis: classification	0h 20m
2016-02-13	Android research and theory writing	1h 25m
2016-02-14	Correcting thesis: used tools	0h 30m
2016-02-14	Research: LaTeX and Android	1h 15m
2016-02-15	Android theory writing	1h 20m
2016-02-17	Android research and theory writing runtime	1h 25m
2016-02-18	Research and theory writing JVM, Dalvik, Gradle	3h 35m
2016-02-19	Figure design Android	1h 25m
2016-02-20	Android theory research	1h 10m
2016-02-21	Andorid theory research	1h 00m
2016-02-23	Jugend Innovativ - Android theory adding pictures	2h 15m
2016-02-25	Jugend Innovativ - correcting and writing	0h 50m
2016-02-26	Jugend Innovativ - proposal writing	1h 30m
2016-02-27	Jugend Innovativ - proposal writing	1h 00m
2016-02-27	Android writing and research: application concept	1h 45m
2016-02-29	Android writing and research: application concept	0h 35m
2016-03-01	Research: class diagram	0h 30m
2016-03-01	Android application specific concept writing	1h 00m
2016-03-01	Android application coding: UI	1h 15m
2016-03-03	Android application coding: UI & basic activity navigation	3h 20m
2016-03-05	Android application coding: RecordActivity, UI and functionality	6h 15m
2016-03-07	Android theory writing	1h 00m
2016-03-08	Android application coding: source extraction, AudioAntSettings UI	0h 55m
2016-03-10	Android application coding: UI	3h 10m
2016-03-11	Android application coding: UI & helper classes	3h 45m
2016-03-12	Android application coding: recording service	4h 10m
2016-03-13	Android application recording service coding	3h 45m
2016-03-14	Thesis writing: implementation	3h 20m
2016-03-15	Thesis writing: implementation and introduction	4h 15m
2016-03-16	Thesis writing: abstract, acknowledgement & Android coding: file to string	3h 50m
2016-03-17	Thesis writing: implementation, Android optimisations	5h 30m
2017-03-19	Thesis writing: implementation, Android optimisations	6h 00m
2019-03-20	Android coding: CommunicationService	6h 30m
2016-03-21	Android optimisations audioantsettings & recordactivity	9h 00m
2016-03-22	Android coding, thesis writing	6h 20m

2016-03-23	Android code restructuring, auto connect coding	7h 10m
2016-03-24	Android coding: AlarmActivity & Thesis writing: implemenation	8h 10m
2016-03-26	Thesis writing: implementation	7h 40m
2016-03-27	Thesis writing: implementation	9h 45m
2016-03-28	Thesis writing: implementation	6h 00m
Sum		254h 10m

Table A.1.: Time protocol Benjamin Hackstock

A.2. Florian Langer

Date	Description	Time spent
2015-03-06	First informational meeting with Prof. Frischmann	0h 30m
2015-05-10	Creating time managment table	0h 40m
2015-05-21	Meeting and discussion with Prof. Frischmann	1h 00m
2015-05-26	Competitor analysis	0h 30m
2015-06-02	Competitor analysis	1h 20m
2015-06-03	Meeting and discussion with Prof. Frischmann	1h 00m
2015-06-12	Group discussion: hardware incl. scoring model, functional specification	1h 00m
2015-06-17	Meeting and discussion with Prof. Frischmann	0h 40m
2015-06-19	Brainstorming: Project title	0h 45m
2015-06-28	Component research (shopping list)	1h 00m
2015-07-01	Component research (shopping list)	0h 20m
2015-07-01	Meeting and discussion with Prof. Frischmann and Prof. Greinöcker	1h 00m
2015-07-03	Product and patent application research	1h 00m
2015-07-08	Elaborating survey questions	1h 30m
2015-07-09	Meeting and discussion with Prof. Frischmann and Prof. Greinöcker	1h 00m
2015-08-16	Meeting of collaborators	1h 30m
2015-09-09	Meeting and discussion with Prof. Frischmann and Prof. Greinöcker	1h 30m
2015-09-09	Tests with the Raspberry Pi GPIO pins	1h 30m
2015-09-20	Pretesting, proofreading and correcting the questionary	0h 45m
2015-09-30	Meeting with Prof. Frischmann and Prof. Greinöcker	0h 20m
2015-09-30	Tests with the Raspberry Pi GPIO pins	0h 30m
2015-09-30	Python, SourceTree and GitLab research	1h 30m
2015-10-01	Meeting with Prof. Frischmann	0h 30m
2015-10-09	Online questionary writing	2h 40m
2015-10-09	Research sensors	1h 30m
2015-10-16	Research survey	1h 00m
2015-10-16	Thesis writing: survey theory	1h 00m
2015-10-20	Thesis writing: survey theory	1h 30m
2015-10-23	Thesis writing: survey theory	1h 00m
2015-10-25	Thesis writing: survey theory	1h 30m
2015-10-27	Thesis writing: survey theory	2h 00m
2015-10-28	Revising survey theory	1h 00m
2015-11-02	Operating concept writing	1h 00m
2015-11-08	Research: Raspberry Pi hardware	1h 00m
2015-11-15	Troubleshooting Raspberry Pi problems	2h 00m
2015-11-22	Troubleshooting Raspberry Pi problems	1h 00m
2015-11-22	Connecting the Raspberry Pi with notebook over Wifi/ hotspot	1h 00m
2015-11-23	Testing the Raspberry Pi GPIO pins	1h 30m

2015-11-23	Coding methods for controlling LEDs	0h 15m
2015-11-26	Testing the microphone	2h 00m
2015-12-02	Drawing drafts of the AudioAnt	0h 45m
2015-12-08	Research: sockets	2h 00m
2015-12-09	Research: recording audio with Raspberry Pi	2h 00m
2015-12-09	Testing the microphone	1h 00m
2015-12-16	Tests with LED's and buttons	2h 00m
2015-12-16	Coding: connection with sockets (Python-Java)	2h 00m
2015-12-30	Design of the prototype on the pinboard	1h 00m
2015-12-30	Collecting data for missing components	1h 00m
2016-01-01	Configuring the Raspberry Pi, researching Samba	3h 00m
2016-01-01	Testing buttons	1h 30m
2016-01-05	Testing buttons, fixing problems with the buttons	3h 00m
2016-01-05	Researching GPIO	1h 00m
2016-01-06	Testing display	1h 30m
2016-01-06	Testing buttons and LEDs of the Raspberry Pi	2h 00m
2016-01-06	Setting up new Raspberry Pi image, Samba, Python	1h 30m
2016-01-06	Troubleshooting Raspberry Pi GPIO pins	2h 30m
2016-01-15	Testing the new Raspberry Pi	1h 45m
2016-01-17	Research: competitor analysis	2h 15m
2016-01-24	Designing prototyp GPIO	0h 15m
2016-01-24	Testing the speaker	0h 30m
2016-01-24	Testing the LEDs and buttons	2h 45m
2016-01-24	Testing the QI transmitter	0h 30m
2016-01-27	EURAC - writing project report	2h 00m
2016-01-28	EURAC - writing project report	1h 00m
2016-01-31	Research transistor as a switch	0h 30m
2016-02-03	Designing circuit diagram	3h 00m
2016-02-04	Shopping for power supply and light source	2h 00m
2016-02-14	Thesis writing: competitor analysis theory	2h 00m
2016-02-15	Thesis writing: competitor analysis theory	1h 00m
2016-02-16	Thesis writing: competitor analysis theory	1h 00m
2016-02-18	Expanding operating concept	2h 00m
2016-02-18	Online shopping for components	1h 00m
2016-02-18	Research: relay module	1h 00m
2016-02-19	Designing circuit diagram	1h 00m
2016-02-20	Designing circuit diagram	2h 30m
2016-02-21	Thesis writing: editing survey theory	2h 00m
2016-02-22	Thesis writing: competitor analysis theory	1h 30m
2016-02-23	Thesis writing: competitor analysis theory	3h 00m
2016-02-24	Design planning, drafts	2h 00m
2016-02-25	Thesis writing: competitor analysis theory	1h 30m
2016-02-26	Constructing the prototype	4h 30m
2016-02-28	Thesis writing: survey theory	2h 00m
2016-02-29	Thesis writing: survey theory	1h 00m

2016-03-01	Constructing the prototype	6h 30m
2016-03-03	Constructing the prototype	4h 00m
2016-03-09	Thesis writing: survey theory	3h 00m
2016-03-10	Thesis writing: survey evaluation	3h 00m
2016-03-11	Thesis writing: survey evaluation	3h 00m
2016-03-13	Editing circuit diagram, proofreading survey	3h 00m
2016-03-14	Collecting links of hardware components	1h 00m
2016-03-15	Researching calculating	1h 00m
2016-03-15	Thesis writing: calculation theory	1h 30m
2016-03-16	Thesis writing: calculation theory	2h 00m
2016-03-16	Thesis writing: finishing survey & competitor analysis	2h 30m
2016-03-17	Thesis writing: calculation theory	3h 00m
2016-03-18	Thesis writing: calculation theory	1h 00m
2016-03-20	Thesis writing: finished calculation theory	2h 00m
2016-03-21	Thesis writing: calculation implementation	2h 00m
2016-03-22	Thesis writing: calculation implementation	2h 00m
2016-03-22	Thesis writing: correction survey theory & implementation	0h 30m
2016-03-22	Thesis writing: hardware theory	1h 30m
2016-03-23	Thesis writing: finished calculation implementation	1h 30m
2016-03-24	Thesis writing: hardware theory	3h 30m
2016-03-25	Thesis writing: hardware theory	2h 30m
2016-03-26	SourceTree research	0h 30m
2016-03-27	Thesis writing: hardware implementation	6h 00m
2016-03-28	Thesis writing: hardware implementation	5h 00m
2016-03-31	Thesis writing: hardware implementation	2h 00m
2016-04-03	Thesis writing: correction of all parts	2h 30m
Sum		183h 30m

Table A.2.: Time protocol Florian Langer

A.3. Daniel Sparber

Date	Description	Time spent
2015-03-06	First informational meeting with Prof. Frischmann	0h 30m
2015-03-08	Online data management setup	1h 00m
2015-03-08	Elaborating goals	0h 15m
2015-05-10	Creating time managment table	0h 40m
2015-05-14	Designing drafts for the AudioAnt	1h 30m
2015-05-21	Meeting and discussion with Prof. Frischmann	1h 00m
2015-06-02	Patent seach	1h 15m
2015-06-03	Meeting and discussion with Prof. Frischmann	1h 00m
2015-06-12	Group discussion: hardware incl. scoring model, functional specification	1h 00m
2015-06-15	Writing down previously discussed matters	0h 30m
2015-06-17	Researches for scoring model + finishing scoring model	0h 40m
2015-06-17	Meeting and discussion with Prof. Frischmann	0h 40m
2015-06-19	Brainstorming: Project title	0h 45m
2015-06-26	Writing general proposal	1h 30m
2015-06-30	Writing draft proposal for offical application	0h 45m
2015-07-01	Meeting and discussion with Prof. Frischmann and Prof. Greinöcker	1h 00m
2015-07-08	Elaborating survey questions	1h 30m
2015-07-09	Meeting and discussion with Prof. Frischmann and Prof. Greinöcker	1h 00m
2015-07-19	Revising survey questionary	1h 00m
2015-07-25	Revising survey questionary	1h 00m
2015-07-25	Setting up LaTeX and Git repository for LaTeX files	0h 30m
2015-07-30	Research: Steppat (introduction)	0h 30m
2015-08-01	Meeting with Prof. Frischmann	1h 00m
2015-08-01	Setting up Raspberry Pi	1h 00m
2015-08-09	Setting up workspace and corresponding Git repository	0h 30m
2015-08-12	Research: Steppat (audio signals)	1h 20m
2015-08-16	Meeting of collaborators	1h 30m
2015-08-16	Coding chapter two examples of the audio Coding book	1h 00m
2015-08-22	Coding chapter two examples of the audio Coding book (finished)	1h 00m
2015-08-24	Research: Steppat (compression of audio files)	1h 30m
2015-08-25	Research: Steppat (tools for audio file handling)	0h 45m
2015-08-26	Research: Steppat (tools for audio file handling)	0h 45m
2015-08-27	Research: Steppat (visualisation of audio files)	1h 00m
2015-08-28	Research: Steppat (Fourier Transformation)	1h 15m
2015-08-28	Research: Steppat (spectral analysis)	0h 50m
2015-09-09	Meeting and discussion with Prof. Frischmann and Prof. Greinöcker	1h 30m
2015-09-09	Testing the Raspberry Pi GPIO pins	1h 30m

2015-09-11	Tutorials: R (Coding language introduction, variables, vectors, logic statements)	1h 00m
2015-09-12	Tutorials: R (matrices, table, importing data)	0h 45m
2015-09-13	Tutorials: R (packages, working directories, scripts)	0h 50m
2015-09-15	Research: JAudio	0h 40m
2015-09-15	Filling out proposal form	0h 20m
2015-09-15	Researching and testing: JAudio GUI and features	1h 20m
2015-09-17	Thesis writing: used tools, audio analysis	0h 45m
2015-09-17	Research: Java and Raspberry Pi	0h 25m
2015-09-19	Research: Java sound recoginition, WEKA	1h 20m
2015-09-20	Thesis writing: patents, requirements for assigning a patent	0h 40m
2015-09-20	Research: Streaming audio, Java and Raspberry Pi microphone, FFT	0h 45m
2015-09-21	Thesis writing: Patents: general, novelty, non-obviousness, exclusions	1h 00m
2015-09-22	Resarch: Sound recogniton - a cognitive way	0h 40m
2015-09-23	Thesis writing: added patent sections	0h 20m
2015-09-24	Thesis writing: intellectual property, patent specification	1h 00m
2015-09-26	Thesis writing: patent offices, patent search	1h 15m
2015-09-27	Thesis writing: patent search platforms, professional patent search	0h 30m
2015-09-27	Research: sound recognition	0h 45m
2015-09-28	Research: feature extraction, recognition algorithms	1h 00m
2015-09-28	Applying HTL thesis template	0h 30m
2015-09-29	Refactoring writing workspace	1h 00m
2015-09-29	Setting up local Android workspace	0h 30m
2015-09-30	Meeting with Prof. Frischmann and Prof. Greinöcker	0h 20m
2015-09-30	Thesis writing: Sound, soundwaves	0h 45m
2015-10-01	Meeting with Prof. Frischmann	0h 30m
2015-10-01	Protocol writing, creating LaTeX template	0h 45m
2015-10-01	Survey, patent search	1h 00m
2015-10-03	Patent search, thesis writing: performed patent search	1h 00m
2015-10-03	Updated titel page, modified template	0h 30m
2015-10-04	Patent search, thesis writing: patent overview (CN202551174U)	1h 30m
2015-10-05	Restructured main LaTeX file	0h 30m
2015-10-05	LaTeX research, thesis writing: patent search	1h 25m
2015-10-06	Thesis writing: patent claims (CN202551174U)	0h 40m
2015-10-06	Thesis writing: patent description (CN201551174U)	0h 50m
2015-10-08	Information about ÖPA patent search from Prof. Frischmann	0h 15m
2015-10-08	Thesis writing: difference between utility models and patents	1h 10m
2015-10-10	Patent search proposal ÖPA	1h 20m
2015-10-11	Thesis writing: utility model CN201556267U, revising patent search	2h 00m
2015-10-11	LaTeX research	0h 45m
2015-10-12	Revising patent search, patent search proposal ÖPA schema drawing	1h 30m

2015-10-12	Audio analysis tests with Audacity	0h 45m
2015-10-13	Tests with JAudio	1h 15m
2015-10-14	Corrected patent search chapter, added missing citations	2h 40m
2015-10-16	Tests embedding JAudio	1h 15m
2015-10-17	Tests embedding JAudio	3h 05m
2015-10-18	Research and tryouts: Java and microphones	2h 15m
2015-10-19	Research (overview): feature extraction, windowing, supervised machine leraning	1h 00m
2015-10-20	Setup testing workspace	0h 35m
2015-10-20	Visualising audio features	0h 50m
2015-10-21	Discussion with Prof. Greinöcker about audio analysis strategy	0h 45m
2015-10-21	Working out demarcation criteria	0h 50m
2015-10-22	Audio analysis approaches	1h 45m
2015-10-23	Rserve: tests and research	1h 00m
2015-10-24	Including Rserve into Android, Android tutorials (basics, permissions)	1h 00m
2015-10-26	Including Rserve into Android	2h 00m
2015-10-28	Thesis writing: Audio analysis applicationroaches	1h 00m
2015-10-29	LaTeX document to rtf: research, installation and converting	1h 00m
2015-10-29	Revise survey theory and usability	0h 40m
2015-10-29	Android Studio: troubleshooting javax imports	0h 50m
2015-11-01	Gathering and analysing ringtones	1h 30m
2015-11-02	Visualising audio features of ringtones, performing audio analysis	2h 00m
2015-11-03	Analysing results of JAudio extraction	1h 45m
2015-11-04	Research Android and audio, tryouts Android	2h 00m
2015-11-05	Research: Android feature extraction	1h 00m
2015-11-07	Audio recognition concept	1h 45m
2015-11-08	Research statistical methodes	1h 00m
2015-11-09	Updated analysis concept, research and tryouts R	1h 30m
2015-11-10	R packages tuneR and seewave tryouts	1h 00m
2015-11-14	Read wav file with Java and generate array with sample values	4h 00m
2015-11-15	Windowing: implementing Hann window	1h 45m
2015-11-16	Java and R, passing values	1h 30m
2015-11-17	Calling R analysis via Java	2h 45m
2015-11-18	Extract strongest frequency from input signal	2h 15m
2015-11-19	Raspberry Pi setup	0h 45m
2015-11-19	Java audio stream from microphone	1h 40m
2015-11-21	Analysing audio stream	2h 20m
2015-11-22	Analysing audio stream	3h 00m
2015-11-23	Discussion with Prof. Greinöcker	1h 00m
2015-11-23	Implementing automatised testing (adapting existing code)	1h 45m
2015-11-24	Implementing automatised testing (adapting existing code)	3h 00m
2015-11-26	Implementing automatised testing (database)	2h 20m
2015-11-28	Implementing automatised testing (database)	1h 20m
2015-11-29	Implementing automatised testing (database)	2h 20m

2015-11-29	Writing application for the EURAC competition	1h 00m
2015-11-30	Extracting strongest frequency	1h 15m
2015-12-01	Performing tests, adjusting dedection parameters	2h 45m
2015-12-03	Raspberry Pi configuration, running Java programm on Raspberry Pi	3h 15m
2015-12-06	Tests with ffmpeg	0h 45m
2015-12-13	Converting wav files to mp3	3h 15m
2015-12-14	Analysing mp3, added logger	2h 30m
2015-12-15	Applying for Jugend Innovativ competition	0h 45m
2015-12-15	Analysing ÖPA search patents	1h 15m
2015-12-16	Enabling communication between Python and Java, tried audio analysis on the Raspberry Pi	4h 00m
2015-12-17	Fixing Raspberry Pi issues	1h 30m
2015-12-18	Fixing Raspberry Pi issues	2h 00m
2015-12-19	Fixing Raspberry Pi issues	1h 30m
2015-12-20	Fixing Raspberry Pi issues	1h 30m
2015-12-21	Fixing Raspberry Pi issues, trying out audio analysis	2h 00m
2015-12-22	Updated socket (Java / Python), trying out audio analysis on the Raspberry Pi 2	2h 00m
2015-12-23	Trying out audio analysis on the Raspberry Pi 2	2h 30m
2015-12-24	Rewriting strongest frequency dedection, multiple peaks	5h 30m
2015-12-25	Reserch features, implementing Spectral Rolloff Point (SRP)	5h 45m
2015-12-29	Audio analysis: support for multiple sounds	4h 00m
2015-12-30	Thesis writing: patent search (WO2008067638A1, US8269625B2)	2h 00m
2015-12-31	Thesis writing: patent search (WO2011000113A1), added patent applicationendix	2h 30m
2016-01-01	Thesis writing: patent search (further ÖPA results), summary	1h 30m
2016-01-02	Research features (MFCC, LPCC)	2h 00m
2016-01-03	Implementing MFCC	2h 15m
2016-01-04	Implementing and testing MFCC	3h 00m
2016-01-05	Implementing energy analysis, testing the analysis	5h 00m
2016-01-06	Testing, filtering tryouts, Raspberry Pi GPIO input	4h 30m
2016-01-07	Connection between Python and Java	3h 00m
2016-01-08	Modifications for Raspberry Pi	2h 00m
2016-01-09	Revised Python code	1h 30m
2016-01-11	Tests with XML	1h 30m
2016-01-12	Correlation (pearson coefficient)	2h 00m
2016-01-14	Research Classification	1h 30m
2016-01-18	Linear Discriminant Analysis (LDA), sounds XML	3h 00m
2016-01-23	EURAC project report, be the best proposal	2h 30m
2016-01-24	EURAC project report writing	3h 45m
2016-01-25	EURAC project report writing	1h 00m
2016-01-26	EURAC project report writing	1h 15m
2016-01-27	Alert light Raspberry Pi	1h 30m
2016-01-28	Alert light Raspberry Pi	1h 00m

2016-01-30	Revising code, error handling, video for the BeTheBest competition	4h 30m
2016-02-01	Research XML; XSLT	1h 00m
2016-02-02	Implemented XSLT for recorded sounds, bugixes	3h 30m
2016-02-03	Moved properties to a XML file	2h 00m
2016-02-06	Refactroing config properties, bugfixes, testing	6h 30m
2016-02-07	Classification revised, theory audio	3h 00m
2016-02-08	Extended logging, thesis writing: audio signals	6h 30m
2016-02-10	Evaluating long term test logs, Thesis writing: windowing, Fourier transform	7h 30m
2016-02-11	Thesis writing: window overlap, DFT, FFT	5h 30m
2016-02-12	Thesis writing: classification	6h 00m
2016-02-13	Thesis: proof reading usability, used tools	1h 30m
2016-02-14	Thesis writing: used tools	3h 30m
2016-02-16	Thesis: correcting, writing approach	5h 00m
2016-02-17	Thesis writing: approach (analysis process)	1h 45m
2016-02-18	Thesis writing: feature selection	3h 45m
2016-02-19	Thesis writing: feature selection (spectrum)	1h 00m
2016-02-20	Thesis writing: feature selecten (completed)	3h 15m
2016-02-21	Thesis writing: feature implementation	3h 45m
2016-02-22	LaTeX issues, project report “Jugend Innovativ”	4h 00m
2016-02-23	Android correction, project report “Jugend Innovativ”	2h 00m
2016-02-24	Hardware discussion, CAD drawing, project report “Jugend Innovativ”	4h 30m
2016-02-25	CAD drawing	3h 00m
2016-02-26	Manufacturing the prototype	4h 00m
2016-02-27	Project report “Jugend Innovativ”	2h 30m
2016-02-29	Hardware controller: Display tests	2h 00m
2016-03-01	Manufacturing the prototype	5h 00m
2016-03-02	Updating hardware controller	3h 15m
2016-03-03	Manufacturing the prototype, testing	5h 30m
2016-03-04	Testing the prototype	1h 15m
2016-03-05	Testing the prototype	3h 30m
2016-03-07	Thesis writing: match rate	2h 00m
2016-03-08	Thesis writing: match rate (energy)	2h 30m
2016-03-09	Thesis writing: match rate (strongest frequency)	3h 00m
2016-03-12	Thesis writing: match rate (MFCC)	5h 00m
2016-03-13	Thesis writing: classification	3h 30m
2016-03-14	Thesis: Correction patent search, writing testing and abstract (german)	6h 00m
2016-03-15	Thesis writing: hardware controller	3h 00m
2016-03-16	Thesis: correcting; research Raspberry Pi AP mode, testing	5h 00m
2016-03-17	Raspberry Pi: Wifi and hotspot	7h 00m
2016-03-18	Raspberry Pi microphone issues	2h 00m
2016-03-19	Server side implementation for Android application	9h 00m

2016-03-20	Server side implementation for Android application	8h 30m
2016-03-21	Raspberry Pi: extending hardware controller; demo video	9h 00m
2016-03-22	Server side implementation for Android application	4h 45m
2016-03-23	Thesis: correction audio, formatting	2h 00m
2016-03-24	Android application implemeted	6h 00m
2016-03-25	Android communication bugfixes	9h 00m
2016-03-26	Thesis: corrections	2h 00m
2016-03-27	Thesis: corrections	2h 00m
2016-03-29	Thesis: corrections and formatting	5h 00m
2016-03-30	Thesis: corrections and formatting	2h 00m
2016-03-31	Thesis: corrections and formatting	1h 45m
2016-04-01	Thesis: corrections and formatting	2h 00m
2016-04-04	Thesis: corrections and formatting	2h 00m
Sum		451h 00m

Table A.3.: Time protocol Daniel Sparber

B. Survey

Fragebogen AudioAnt

Wir, Daniel Sparber, Benjamin Hackstock und Florian Langer, drei Schüler der HTL-Anichstraße entwickeln im Rahmen unserer Diplomarbeit ein Hilfsgerät für Menschen mit eingeschränktem Hörvermögen.

Unser Hilfsgerät soll dazu in der Lage sein, bestimmte Töne wie z.B.: das Läuten der Türklingel zu lernen und bei Vorkommen zu erkennen. Wenn ein Geräusch erkannt wird, gibt das Gerät Lichtblitze und einen Warnton aus. Optional kann bei verwenden der entwickelten Smartphone-App eine Benachrichtigung an das Smartphone gesendet werden um den Benutzer auf diese Weise vom Auftreten eines Geräusches zu informieren. Dadurch können Geräusche die ansonsten aus verschiedensten Gründen überhört werden würden (z.B. durch Musikhören im Zimmer, Nichttragen des Hörgerätes,...) wieder wahrgenommen werden.
Zusätzlich ist eine Funktion geplant, die Hilfeschreie erkennen kann und so bei der Benutzung durch ältere Menschen im Notfall Hilfe durch Verwandte oder Einsatzkräfte rufen kann.

Der Fragebogen soll uns dabei helfen, die Priorität von einzelnen Features abzuschätzen und die Entwicklungsschwerpunkte für unser Hilfsgerät richtig zu setzten.

Figure B.1.: Survey - Page 1 (Introduction)

Aktuelle Situation

Welche Geräusche überhören Sie besonders häufig?

- [ ] Handy & Telefon
- [ ] Türklingel
- [ ] Wecker
- [ ] Sonstiges:

Wie gut ist Ihre Hörleistung ohne Hörgerät?

Lassen Sie diese Frage aus, falls keine genauen Werte bekannt sind.

- ○ Normalhörig (Hörschwelle < 20dB)
- ○ Gering Schwerhörigkeit (Hörschwelle 25-40 dB)
- ○ Mittelgradige Schwerhörigkeit (Hörschwelle 40-60 dB)
- ○ Höchgradige Schwerhörigkeit (Hörschwelle > 60 dB)
- ○ An Gehörlosigkeit grenzende Schwerhörigkeit (Hörschwelle > 80 dB)
- ○ Gehörlos

Falls Sie ein Hörgerät besitzen, wie oft verwenden Sie dieses zu Hause?

- ○ Immer
- ○ Oft
- ○ Manchmal
- ○ Selten
- ○ Nie

Figure B.2.: Survey - Page 2 (Current situation)

Technische Affinität

Im Rahmen der Entwicklung der Smartphone-App und der Platte ist es für uns wichtig, zu wissen auf welchen technischen Stand die potenziellen Nutzer unserer Entwicklung sind. Die folgenden Fragen sollen uns dies ermöglichen.

Besitzen Sie ein Smartphone?

- ○ Ja
- ○ Nein

Besitzen Sie ein Tablet, einen Laptop oder einen Computer?

- ○ Ja
- ○ Nein

Falls vorhanden, tragen Sie Ihr Smartphone am Körper bzw. in Sicht- oder Hörweite?

Dies soll uns helfen die Sinnhaftigkeit von Benachrichtigungen auf das Smartphone einzuschätzen.

	1	2	3	4	5	
Ja, immer	○	○	○	○	○	Nein, nie

Welche Anwendungen nutzen Sie auf ihrem Smartphone/Tablet/Computer?

- ☐ Webbrowser
- ☐ E-Mail Anwendungen
- ☐ Office-Programme (Word, Excel, PowerPoint, usw.)
- ☐ Spiele
- ☐ Sonstiges:

Figure B.3.: Survey - Page 3.1 (Technical affinity)

Wie gehen Sie bei Problemen von technischen Geräten vor?

	1	2	3	4	5	
Ich versuche das Problem selbst zu lösen	○	○	○	○	○	Ich hole Hilfe

Was würden Sie am ehesten tun wenn Sie ein neues Handy bekommen?

- ○ Ausprobieren
- ○ Die Bedienungsanleitung lesen
- ○ Jemanden um Erklärung bitten
- ○ Jemanden anderen um die Einrichtung bitten

Wie gerne experimentieren Sie mit neuen, elektronischen Geräten? Beispiel: Kauf eines neuen Navigationsgerätes.

	1	2	3	4	5	
Sehr gerne	○	○	○	○	○	Gar nicht

Figure B.4.: Survey - Page 3.2 (Technical affinity)

Hilfsgerät

Die folgenden Fragen sollen uns helfen, besser auf Wünsche und Anregungen von potenziellen Kunden einzugehen und diese später in der Praxis besser umsetzen zu können.

Worauf legen Sie besonders viel Wert?

Bitte 2 Antworten auswählen!

- [ ] Leichte Bedienbarkeit
- [ ] Viele Einstellmöglichkeiten
- [ ] Kompakte Bauweise
- [ ] Zuverlässige Erkennung
- [ ] Optisch ansprechendes Design

Soll die Geräuschanalyse eher empfindlich sein, dafür aber öfter fälschlicherweise ein erkanntes Signal melden oder weniger Fehlerkennungen haben, dafür aber evtl. manchmal Geräusche überhören?

	1	2	3	4	5	
Empfindlicher e Erkennung	○	○	○	○	○	Zuverlässige Erkennung

Halten Sie eine Erkennung von Hilfeschreien in Notfällen für sinnvoll?

- ○ Ja
- ◉ Nein

Figure B.5.: Survey - Page 4 (Assistant tool)

Demographische Daten

Geschlecht

- Männlich
- Weiblich

Alter

- < 25 Jahre
- 25-45 Jahre
- 46-65 Jahre
- > 66 Jahre

Anzahl der Personen im Haushalt

- eine Person
- zwei Personen
- mehr als zwei Personen

Figure B.6.: Survey - Page 5 (Demographical data)

C. Patent search

C.1. ÖPA patent search proposal

C.2. ÖPA patent search results

österreichisches patentamt

RE 729a UNI / FHS / HTL - RECHERCHENANTRAG

Antrag auf Erstellung eines gem. §33 Patentamtsgebührengesetz unentgeltlichen schriftlichen Gutachtens über den Stand der Technik gem. §57a Z1 Patentgesetz für Dissertationen und Diplomarbeiten

Aktenzeichen *(wird vom Österreichischen Patentamt vergeben!)*

An das
Österreichische Patentamt
Dresdner Straße 87
1200 Wien

IPC:
Ref.:
TA:

Bitte für amtliche Vermerke freihalten!

Die eingeklammerten Zahlen verweisen auf Erläuterungen in der angeschlossenen Ausfüllhilfe!

(1) **Antragsteller/in**

Vor- und Zuname, *ggf. Geburtsdatum/Firmenwortlaut sowie Firmenbuchnummer/Vereinsname*	Anschrift/en *(Wohnadresse bzw. Unternehmenssitz) Straße/Hausnr., PLZ/Ort, Bundesland*
Daniel Sparber, 25.02.1997	Spitz 18/1, 6406 Oberhofen im Inntal, Tirol

(2)

Telefon	Telefax	E-Mail
+43 664 2432067		sparber.daniel@gmail.com

Beilagen, die das zu recherchierende konkrete technische Problem ausführliche darlegen

(3) 1 Seiten Beschreibung (2fach) (4) 1 Blatt Zeichnungen (2fach)

(5) ☐ Zusammenfassung (2fach)

Datum	**Unterschrift(en)** *(Antragsteller/in)*

(6) **Bestätigung des Institutsleiters, Studiengangleiters bzw. des Direktors/Abteilungsvorstandes**

Es wird bestätigt, dass der/die Antragsteller/in die Recherche für seine/ihre Dissertation/Diplomarbeit am Institut
HTL Anichstraße
mit dem Titel
AudioAnt
benötigt.

Name *(Institutsleiter, Studiengangleiter bzw. Direktor/Abteilungsvorstand)*
Direktor: Mag. Günther Laner

Telefon	Telefax	E-Mail
+43 (0)512 59717 0	+43 (0)512 59717 72	direktion@htlinn.ac.at

Datum und Unterschrift *(Institutsleiter, Studiengangleiter, Direktor/Abteilungsvorstand)*, Stampiglie *(zB Rundsiegel)*

DVR 0078018 RE 729a, v01.10, Seite 1

Figure C.1.: ÖPA patent search proposal - Page 1

Beschreibung

Gegenstand der Recherche ist ein elektronisches Hilfsmittel für Personen mit eingeschränktem Hörvermögen. Dieses Hilfssystem soll das alltägliche Leben zahlreicher Menschen erleichtern, indem es diverse Hinweistöne (z.B. Haustürklingel) erkennt und dies kenntlich macht.

Das Hilfsgerät ist in der Lage, über ein Mikrofon Töne aufzunehmen und erlernte Audiosignale (Telefonklingeln, Türglocke, Wecker, Hilferufe, etc.) mittels Audioanalyse zu erkennen. Wurde ein Signal erkannt, gibt das Gerät einen lauten Warnton, sowie ein optisches Signal (Lichtblitz) aus.

Das Hilfsgerät beherbergt folgende Hauptkomponenten:

(1) **Mikrofon:** zur Aufnahme von Umgebungsgeräuschen

(2) **Rechnereinheit:** zur Analyse der eingehenden Audiosignale

(3) **Lautsprecher:** zum Ausgeben eines Warntons

(4) **Lautstärkenregler:** zum Regeln der Warntonlautstärke

(5) **LED Streifen:** zum Erzeugen eines Lichtblitzes

(6) **LED Statusleuchte:** zeigt an, ob das Gerät gerade in Verwendung ist

Kennzeichen des Gerätes:

(1) Das Gerät ist dadurch gekennzeichnet, dass es nach dem Lernen eines Signales dieses eigenständig wiedererkennt.

(2) Das Gerät ist dadurch gekennzeichnet, dass die oben genannten Hauptkomponenten in einem Gehäuse untergebracht sind.

(3) Das Gerät ist dadurch gekennzeichnet, dass es portabel ist und direkt nach dem Anstecken an das Stromnetz betriebsbereit ist.

1

Figure C.2.: ÖPA patent search proposal - Page 2

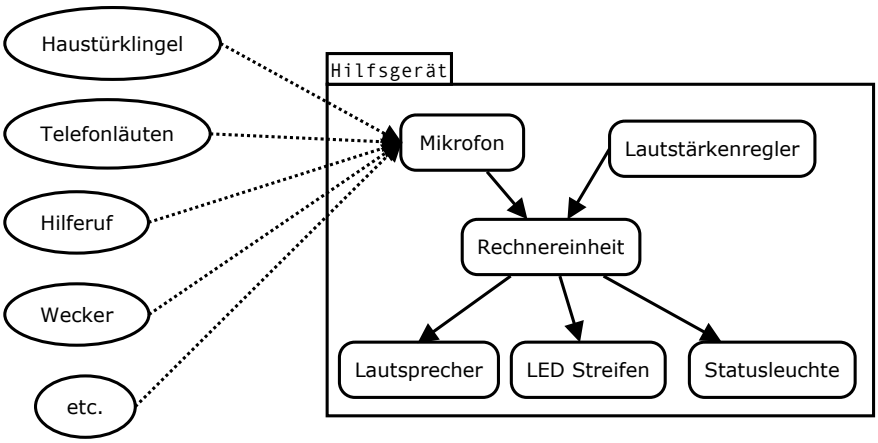


Abbildung 1: Schematische Darstellung des Hilfsgerätes

2

Figure C.3.: ÖPA patent search proposal - Page 3

österreichisches patentamt

Gruppe Technik
Dresdner Straße 87, PF 95
1200 Wien
Telefon +43 (0)1 53424 0
Fax +43 (0)1 53424 535
www.patentamt.at
UID ATU38266407
DVR 0078018
Bankverbindung
BIC: BUNDATWW
IBAN: AT75 0100 0000 0516 0000

Daniel Sparber
Spitz 18/1
6406 Oberhofen im Inntal

abgefertigt am
06. Nov. 2015

Wien, 04.11.2015

Geschäftszahl: 3 RG 31/2015 - 1
Antragsdatum: 16.10.2015

Bitte Geschäftszahl bei allen Eingaben anführen!

Recherche (§ 57a Z 1 PatG)

über den Stand der Technik,

erstellt auf Grund Ihres Antrags vom 16.10.2015 mit dem Betreff elektronisches Hilfsmittel für Personen mit eingeschränktem Hörvermögen.

Es sind die im beigefügten Recherchenbericht angegebenen einschlägigen Druckschriften ermittelt worden. Die Vollständigkeit der Ermittlung kann nicht gewährleistet werden.

Zu Ihrer Information wird darauf hingewiesen, dass ein Infoblatt betreffend Gutachten gemäß § 57a Z 2 PatG sowie die §§ 1-3 PatG im Internet über die Seite des Patentamtes eingesehen werden kann oder auf Anfrage beim Patentamt kostenlos erhältlich ist.

Mag.rer.nat. Judith Stoll

Technische Abteilung 3
Tel. +43 (1) 53424 550

1 Beilage(n)
Recherchenbericht

Aktenzeichen: RG 31/2015 [RG 4R] Seite 1 von 3

Figure C.4.: ÖPA patent search results - Page 1

Erläuterungen zum ermittelten Stand der Technik:

D1: WO 2008067638 A1 (HARMONYA TECHNOLOGIES INC) 12.06.2008 "Environment sensing system for the hearing-impaired"

D1 beschreibt ein System zum Erkennen eines bestimmten Geräusches in der Umgebung einer tauben bzw. schwerhörigen Person. Hierzu wird ein erkanntes Umgebungsgeräusch mit Hilfe eines Vibrationssignals und einer auf einem Display angezeigten Nachricht an eine vom Benutzer getragene persönliche Vorrichtung übertragen.

D2: US 2011025499 A1 (HOY LESLIE D. ET AL.) 03.02.2011 "Signal processing system and methods for reliably detecting audible Alarms"

In **D2** werden akustische Alarmsignale, welche beispielsweise von einem Feueralarm, Rauchmelder oder CO2-Detektor ausgesendet werden, an eine persönliche Vorrichtung eines Benutzers, welche z.B. in ein Bett integriert sein kann, übertragen.

D3: WO 2011000113 A1 (HARMONYA TECHNOLOGIES) 06.01.2011 "Multiple sound and voice detector for hearing-impaired or deaf person"

In der in **D3** eingesetzten Vorrichtung wird durch Vergleich von vorab abgespeicherten Signalen mit aufgenommenen Umgebungsgeräuschen zwischen non-verbalen Signalen und dem gesprochenen Wort unterschieden.

D4: WO 0074013 A1 (EVETS COMM LTD) 07.12.2000 "Alarm system"

In **D4** analysiert eine spezielle Analyseeinheit die mit einem Mikrophon aufgenommenen Umgebungsgeräusche und kann in Abhängigkeit des erkannten akustischen Signals, bei dem es sich beispielsweise um einen Feueralarm handelt, ein entsprechendes Licht zum Leuchten bringen.

D5: DE 19816752 A1 (NEUMAIER MARTIN) 28.10.1999 "Einrichtung und Verfahren zur Erzeugung eines Steuersignals zur Ansteuerung wenigstens einer Signalquelle, die ein für einen Hörgeschädigten wahrnehmbares Signal erzeugt"

Eine spektrale Analyseeinheit sorgt in **D5** dafür, dass zwischen unterschiedlichen akustischen Signalquellen wie dem Läuten einer Türglocke oder dem Läuten des Telefons unterschieden werden kann, sodass in Abhängigkeit der erkannten Signale ein Vibrationsmelder oder ein Blitzlichterzeuger aktiviert werden.

D6: KR 20110092479 A (YANG KWANG) 18.08.2011 „Alarm displaying apparatus and method for persons hard of hearing"

In **D6** kommt eine Reihenanordnung von Mikrophonen zum Einsatz, um bestimmte akustische Signale aus den Umgebungsgeräuschen herauszufiltern, sodass in Abhängigkeit

Aktenzeichen: RG 31/2015 [RG 4R] Seite 2 von 3

Figure C.5.: ÖPA patent search results - Page 2

der erkannten Alarmsignale entsprechende Lämpchen zum Leuchten gebracht werden können.

D7: US 4237449 A (ZIBELL J SCOTT) 02.12.1980 "Signalling device for hard of hearing persons"

In **D7** ist ebenfalls bereits die Aufnahme von bestimmten Geräuschen, wie dem Laut einer Autohupe, einer Polizeisirene, einer Türglocke oder eines Feueralarms, mit einem Mikrophon vorgesehen, wobei in Abhängigkeit des erkannten Geräusches eine LED dazu gebracht wird, ein Blinklicht abzugeben.

D8: JP 2004073417 A (WATANABE MASAHIRO) 11.03.2004 "Report apparatus"

Gemäß **D8** werden die mit einem Mikrophon aufgenommenen akustischen Signale, welche von bestimmten Küchengeräten oder anderen elektrischen Haushaltsgeräten ausgesandt werden, für schwerhörige Personen verstärkt.

D9: WO 2014168594 A1 (OKSUN AKIN) 16.10.2014 "Warning system comprising wristband embodiment"

Die in **D9** beschriebene Vorrichtung zur Erkennung von bestimmten Alarm- oder Warnsignale ist in eine Armbanduhr mit Display integriert, sodass auch den Signalen entsprechende Nachrichten für den Benutzer angezeigt werden können.

D10: US 2009154744 A1 (SNYDER WAYNE HARVEY) 18.06.2009 "Device for the hearing impaired"

Auch in **D10** werden akustische Signal mit Hilfe einer Reihe von Mikrophonen aufgenommen. In Abhängigkeit des erkannten Signals werden entsprechende Symbole auf einem Display der Vorrichtung für taube bzw. schwerhörige Personen angezeigt.

D11: JP 3043730 B1 (NEC GUMMA LTD) 22.05.2000 „Auditorily disabled person assisting instrument"

Eine Vergleichsschaltung sorgt in der in **D11** beschriebenen Vorrichtung dafür, dass zwischen bekannten akustischen Signalen wie Alarmsignalen und unbekannten bzw. nicht näher spezifizierten Signalen wie Umgebungsgeräuschen unterschieden werden kann.

D12: DE 4033673 A1 (PRELL MAX) 30.04.1992 "Drahtloses Alarm-/Kommunikationssystem für Schwerhörige"

Ein in einer Tasche unterbringbarer Empfänger teilt in **D12** schwerhörigen Personen den Empfang von unterschiedlichen akustischen Signalen wie Telefon, Klingel, Wecker oder Baby-Alarm mit.

Figure C.6.: ÖPA patent search results - Page 3

Recherchenbericht zu **RG 31/2015**

österreichisches patentamt

Klassifikation des Anmeldungsgegenstands gemäß IPC: ***G08B 5/36*** (2006.01); ***G08B 21/00*** (2006.01); ***H04R 25/00*** (2006.01)	
Klassifikation des Anmeldungsgegenstands gemäß CPC: ***G08B 5/36*** (2013.01); ***G08B 21/00*** (2013.01); ***H04R 25/00*** (2015.10)	
Recherchierter Prüfstoff (Klassifikation): G08B, H04R, G10L	
Konsultierte Online-Datenbank: EPODOC, WPI, TXTDE, TXTEN	
Dieser Recherchenbericht wurde zu den am eingereichten Ansprüchen erstellt.	

Kategorie*)	Bezeichnung der Veröffentlichung: Ländercode, Veröffentlichungsnummer, Dokumentart (Anmelder), Veröffentlichungsdatum, Textstelle oder Figur soweit erforderlich	Betreffend Anspruch
	WO 2008067638 A1 (HARMONYA TECHNOLOGIES INC) 12. Juni 2008 (12.06.2008) Zusammenfassung; Figuren 1, 4; Beschreibung der Figuren; Ansprüche 1-24; US 2011025499 A1 (HOY LESLIE D. ET AL.) 3. Februar 2011 (03.02.2011) Zusammenfassung; Figuren 1, 3; Beschreibung der Figuren; Ansprüche 1-20; WO 2011000113 A1 (HARMONYA TECHNOLOGIES) 06. Januar 2011 (06.01.2011) Zusammenfassung; Figuren 2, 5-9; Beschreibung der Figuren; Ansprüche 1-98; WO 0074013 A1 (EVETS COMM LTD) 07. Dezember 2000 (07.12.2000) Zusammenfassung; Figur 1; Beschreibung der Figur; Ansprüche 1-16; DE 19816752 A1 (NEUMAIER MARTIN) 28. Oktober 1999 (28.10.1999) Zusammenfassung; Figuren 1, 4-7; Beschreibung der Figuren; Ansprüche 1-34; KR 20110092479 A (YANG KWANG) 18. August 2011 (18.08.2011) englische Zusammenfassung; Figur 1; Beschreibung der Figur; Ansprüche 1-7; US 4237449 A (ZIBELL J SCOTT) 02. Dezember 1980 (02.12.1980) Zusammenfassung; Figuren 1-2, 4; Beschreibung der Figuren; Ansprüche 1-10; JP 2004073417 A (WATANABE MASAHIRO) 11. März 2004 (11.03.2004) englische Zusammenfassung; Figur 1; Beschreibung der Figur;	

Datum der Beendigung der Recherche: 04.11.2015	Seite 1 von 2	Prüfer(in): STOLL Judith

*) **Kategorien** der angeführten Dokumente:

- **X** Veröffentlichung **von besonderer Bedeutung**: der Anmeldungsgegenstand kann allein aufgrund dieser Druckschrift nicht als neu bzw. auf erfinderischer Tätigkeit beruhend betrachtet werden.
- **Y** Veröffentlichung **von Bedeutung**: der Anmeldungsgegenstand kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren weiteren Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese **Verbindung für einen Fachmann naheliegend** ist.
- **A** Veröffentlichung, die den allgemeinen **Stand der Technik** definiert.
- **P** Dokument, das von **Bedeutung** ist (Kategorien **X** oder **Y**), jedoch **nach dem Prioritätstag** der Anmeldung veröffentlicht wurde.
- **E** Dokument, das **von besonderer Bedeutung** ist (Kategorie **X**), aus dem ein „**älteres Recht**" hervorgehen könnte (früheres Anmeldedatum, jedoch nachveröffentlicht, Schutz ist in Österreich möglich, würde Neuheit in Frage stellen).
- **&** Veröffentlichung, die Mitglied der selben **Patentfamilie** ist.

DVR 0078018

Figure C.7.: ÖPA patent search results - Page 4

Fortsetzung des Recherchenberichts zu **RG 31/2015**
Seite 2 von 2

österreichisches patentamt

Kategorie*)	Bezeichnung der Veröffentlichung: Ländercode, Veröffentlichungsnummer, Dokumentart (Anmelder), Veröffentlichungsdatum, Textstelle oder Figur soweit erforderlich	Betreffend Anspruch
	Ansprüche 1-4; WO 2014168594 A1 (OKSUN AKIN) 16. Oktober 2014 (16.10.2014) Zusammenfassung; Figuren 1-3; Beschreibung der Figuren; Ansprüche 1-6; US 2009154744 A1 (SNYDER WAYNE HARVEY) 18. Juni 2009 (18.06.2009) Zusammenfassung; Figuren 1-2, 5-7; Beschreibung der Figuren; Ansprüche 1-15; JP 3043730 B1 (NEC GUMMA LTD) 22. Mai 2000 (22.05.2000) englische Zusammenfassung; Figuren 1-2; Beschreibung der Figuren; Ansprüche 1-3; DE 4033673 A1 (PRELL MAX) 30. April 1992 (30.04.1992) Zusammenfassung; Figur 1; Beschreibung der Figur; Ansprüche 1-16;	

Figure C.8.: ÖPA patent search results - Page 5

D. List of Figures

2.1. Sample of a population . . . 10
2.2. Survey, demographic data, question 1 . . . 15
2.3. Survey, demographic data, question 2 . . . 16
2.4. Survey, demographic data, question 3 . . . 16
2.5. Survey, current situation, question 1 . . . 17
2.6. Survey, current situation, question 2 . . . 17
2.7. Survey, current situation, question 3 . . . 17
2.8. Survey, technical affinity, question 1 . . . 18
2.9. Survey, technical affinity, question 2 . . . 18
2.10. Survey, technical affinity, question 3 . . . 19
2.11. Survey, technical affinity, question 4 . . . 20
2.12. Survey, technical affinity, question 5 . . . 20
2.13. Survey, technical affinity, question 6 . . . 20
2.14. Survey, technical affinity, question 7 . . . 21
2.15. Survey, auxiliarity device, question 1 . . . 21
2.16. Survey, auxiliarity device, question 2 . . . 22
2.17. Survey, auxiliarity device, question 3 . . . 22

3.1. Example for a strength-weakness profile . . . 24
3.2. Emergency paging and doorbell intensification tool . . . 25
3.3. Ring tone intensifier . . . 26
3.4. Ring tone intensifier with a wireless doorbell adapter unit . . . 27

4.1. Schematic view of the bell recognition device . . . 39
4.2. Schematic view of the shiny belt doorbell . . . 40
4.3. Schematic view of the sensing system . . . 42
4.4. Schematic view of signal processing system . . . 44
4.5. Schematic view of the sound and voice detector . . . 45

5.1. Examples for bad usability . . . 49
5.2. A heat map showing what users look at most on this website . . . 52
5.3. Items have to be easily identifiable as what they are . . . 54
5.4. Important elements should always be in reach of the user when using the application one-handed . . . 55
5.5. Icon of the App Store in IOS9 . . . 58

6.1. Examples for different frequencies . . . 62
6.2. Examples for different amplitudes . . . 62
6.3. Examples for different phase shifts . . . 63
6.4. Sampling and quantization of a signal for 4-bit PCM . . . 64

6.5. Waveform plot of a doorbell . . . 65
6.6. Spectrum of a chirp . . . 66
6.7. Spectrogram of a chirp . . . 66
6.8. DFT: The given input signal . . . 70
6.9. DFT: Partial waves of the given signal . . . 70
6.10. DFT: Time and frequency domain of the signal . . . 70
6.11. DFT: The resulting frequencies (spectrum) . . . 71
6.12. FFT algorithm . . . 71
6.13. Leakage effect . . . 72
6.14. Reducing the leakage effect . . . 73
6.15. Common window functions . . . 75
6.16. Window function: Overlapping . . . 75
6.17. Strongest frequencies of a given spectrum . . . 77
6.18. Traditional Mel Filter Bank . . . 78
6.19. RMS energy threshold comparison . . . 79
6.20. Graphical plot of the discriminate function analysis example . . . 81

7.1. Layout of the different Raspberry Pi Models . . . 87
7.2. Operating mode of a relay . . . 89

8.1. Android logo and lettering . . . 91
8.2. Lifecyle of an Android Activity . . . 96
8.3. Simplified process of the build ingredients of an Android application . . . 97
8.4. Compiling and execution process of a Java program . . . 98
8.5. Keystore Windows of the APK signing Wizard of Android Studio . . . 99
8.6. Compiling and execution process of an Android application . . . 100

11.1. The audio learning process . . . 114
11.2. The audio analysis process . . . 114
11.3. Sounds which should be recognised . . . 118
11.4. Sounds which should not be recognised . . . 118
11.5. Visualisation of the RMS energy for a doorbell . . . 119
11.6. Spectra of sounds which should be recognised . . . 119
11.7. Spectra of sounds which should not be recognised . . . 120
11.8. Visualisation of the strongest frequencies of a microwave . . . 120
11.9. Spectrograms of sounds which should be recognised . . . 121
11.10. Spectrograms of sounds which should not be recognised . . . 121
11.11. Visualisation of the SRP point for the microwave sample . . . 122
11.12. Visualisation of the MFCCs for the microwave sample . . . 122
11.13. Match rates of two simple arrays . . . 125
11.14. Visualisation of saved RMS Energy values . . . 126
11.15. Matching RMS Energy values . . . 126
11.16. MFCC coefficients: Deviation from the learned values . . . 131
11.17. MFCC coefficients: Deviation from the learned values including borders . . . 132
11.18. The database schema used for saving test values . . . 138

12.1. Raspberry Pi 2 . . . 144
12.2. Casing of the AudioAnt . . . 144
12.3. Qi-transmitter . . . 145

12.4. Display . . . 145
12.5. Single-channel relay . . . 146
12.6. Soundcard . . . 146
12.7. Speaker . . . 147
12.8. Microphone . . . 147
12.9. WLAN stick . . . 148
12.10. Powerful light source . . . 148
12.11. USB and AUX connector . . . 149
12.12. Jack for the power supply . . . 149
12.13. Power supply . . . 150
12.14. 3D-model of the AudioAnt case . . . 151
12.15. Metrics of the case . . . 152
12.16. Circuit diagram . . . 153
12.17. Components of the AudioAnt . . . 154
12.18. Circuit Board and the soldered cables . . . 155
12.19. Florian soldering the LED . . . 156
12.20. LED soldered and isolated with shrinking tube . . . 156
12.21. The finished prototype . . . 157
12.22. The finished prototype (open) . . . 157
12.23. Labelled components of the prototype . . . 158

13.1. The main and recording activity of the Android application . . . 169
13.2. The settings activities of the Android application . . . 170
13.3. The list of all sounds, the alarm activity and an informational text . . . 171
13.4. Example of Dialogs . . . 187
13.5. Example of an AlertDialog . . . 188

B.1. Survey - Page 1 (Introduction) . . . 219
B.2. Survey - Page 2 (Current situation) . . . 220
B.3. Survey - Page 3.1 (Technical affinity) . . . 221
B.4. Survey - Page 3.2 (Technical affinity) . . . 222
B.5. Survey - Page 4 (Assistant tool) . . . 223
B.6. Survey - Page 5 (Demographical data) . . . 224

C.1. ÖPA patent search proposal - Page 1 . . . 226
C.2. ÖPA patent search proposal - Page 2 . . . 227
C.3. ÖPA patent search proposal - Page 3 . . . 228
C.4. ÖPA patent search results - Page 1 . . . 229
C.5. ÖPA patent search results - Page 2 . . . 230
C.6. ÖPA patent search results - Page 3 . . . 231
C.7. ÖPA patent search results - Page 4 . . . 232
C.8. ÖPA patent search results - Page 5 . . . 233

E. List of Tables

6.1. Different Fourier transforms 69
6.2. Values for the discriminate function analysis example 80
6.3. Means of the samples of the discriminate function analysis example 80
6.4. Pooled covariances of the samples of the discriminate function analysis example . 81
6.5. Covariances for group 1 of the samples of the discriminate function analysis example 81
6.6. Covariances for group 2 of the samples of the discriminate function analysis example 81
6.7. Correctness of the classification of the discriminate function analysis example . . 82

9.1. Double-stage overhead calculation 104

12.1. Benefit analysis 143
12.2. Descriptive name and purpose of the components 159

14.1. Material costs 201
14.2. Manufacturing costs 201
14.3. Overheads 202

A.1. Time protocol Benjamin Hackstock 208
A.2. Time protocol Florian Langer 211
A.3. Time protocol Daniel Sparber 217

F. Listings

11.1. Assigning samples to an R Wave . . . 123
11.2. Generating the spectrum with R . . . 123
11.3. Calculating the RMS energy with R . . . 123
11.4. Calculating the strongest frequencies with R . . . 123
11.5. Calculating the SRP with R . . . 124
11.6. Calculating the MFCCs with R . . . 124
11.7. Calculating of the match rate with Java . . . 125
11.8. Normalising the RMS Energy values . . . 127
11.9. Getting the start index for the RMS match rate calculation . . . 127
11.10. Getting the RMS Energy match rate . . . 128
11.11. The members and the equals method of the FrequencyModel . . . 128
11.12. The member and the overlap methode of the StrongestFrequenciesModel . . . 129
11.13. Calculating the match rate of the frequency values (part 1) . . . 129
11.14. Calculating the match rate of the frequency values (part 2) . . . 130
11.15. Calculating the match rate of the SRP values (part 1) . . . 130
11.16. Calculating the match rate of the SRP values (part 2) . . . 131
11.17. Calculating the match rate of the MFCC values (part 1) . . . 132
11.18. Calculating the match rate of the MFCC values (part 2) . . . 133
11.19. Function for returning if the minimum threshold is reached . . . 134
11.20. Global properties: Detection threshold section . . . 134
11.21. Function for returning if all thresholds are reached . . . 135
11.22. Global properties: Detection threshold section . . . 135
11.23. Calculation of the LDA coefficients with R (part 1) . . . 135
11.24. Calculation of the LDA coefficients with R (part 2) . . . 136
11.25. Global properties: Detection weights section . . . 136
11.26. Function returning if the weighted sum is reached . . . 137
11.27. Functions for returning if a match occured . . . 137
11.28. mySQl query for printing the summary of all performed tests . . . 138
11.29. Result of the summary query . . . 139
11.30. mySQl query for printing the summary of all performed tests . . . 139
11.31. Outtake of the detailed log file . . . 140
11.32. Outtake of the detailed log file . . . 141

12.1. The connection handler for the hardware controller . . . 160
12.2. The pin configuration . . . 161
12.3. The general configuration . . . 161
12.4. The output manager of the hardware controller . . . 161
12.5. The LED controller . . . 162
12.6. The alert light controller . . . 163
12.7. The input controller (part 1) . . . 163

12.8. The input controller (part 2) 164
12.9. Adding a button to the input controller 164

13.1. Example for exchanged JSON object 167
13.2. Getting the variables from JSONObject “object” 167
13.3. permissions required for the application 173
13.4. Permissions needed for the AudioAnt application 174
13.5. Method for binding and unbinding CommunicationService.java 175
13.6. OnBind() and OnStartCommand() methods of RecordSignalService.java 176
13.7. OnDestroy() method of RecordSignalService.java 177
13.8. recordToFile() method of RecordSignalService.java 177
13.9. stopRecording() method of RecordSignalService.java 177
13.10. Overridden methods of CommunicationService.java 178
13.11. Private class MyBinder.java 179
13.12. listenForAudioAntMessages() method of CommunicationService.java 180
13.13. connectToServer() method of CommunicationService.java 180
13.14. InitialiseNetworkConnection() method of CommunicationService.java 181
13.15. sendToServer() method of CommunicationService.java 182
13.16. sendLocalBroadcast() method of CommunicationService.java 182
13.17. startAlarmActivity() method of CommunicationService.java 182
13.18. Connecting to a wifi network 183
13.19. Start scanning for available Wi-Fi networks 183
13.20. Example methods of the SharedPreferences helper class Settings.java 184
13.21. playAlertFromName() method of the AudioAnt alert helper class AlertSound-Helper.java 185
13.22. saveAlert() method of the AudioAnt alert helper class AlertSoundHelper.java . . 186
13.23. readAlerts() method of the AudioAnt alert helper class AlertSoundHelper.java . 186
13.24. deleteRecursively() method of the AudioAnt alert helper class AlertSoundHelper.java186
13.25. Example of a loading dialog 187
13.26. Example of a ShowTextAlert 187
13.27. Example of an AlertDialog 188
13.28. Initialising a view object in MainActivity.java 189
13.29. Starting a new activity from MainActivity.java 190
13.30. Initialising the ServiceConnection in MainActivity.java 190
13.31. Adding a TextChangedListener to the EditText “geräuschName” 191
13.32. Request sounds from AudioAnt if necessary 191
13.33. Registering local BroadcastReceivers 191
13.34. Converting a file into a String 192
13.35. Receive the learned sounds of the AudioAnt 193
13.36. Request the learned sounds of the AudioAnt 194
13.37. Initialise View elements with received data 194
13.38. Requesting data from the AudioAnt by sending a JSON file 195
13.39. Choose a Wi-Fi network from the AlertDialog 196
13.40. Method invoked after reconnecting to the AudioAnt’s hotspot 196
13.41. choose a Wi-Fi network from the AlertDialog 197
13.42. Save the received Wi-Fi networks to the arrayAdapter 197
13.43. Alert for choosing a notification sound 198
13.44. Overridden onActivityResult() method 198

13.45. Method for initialising the sound name TextViews 199
13.46. Method for stopping the alerts and closing the activity 199