UiO : Department of Informatics
University of Oslo

Crowdsourced noise mapping

Can we use smartphones to measure noise pollution?

Ákos Pap

Master's Thesis Spring 2017

Crowdsourced noise mapping

Ákos Pap

April 25, 2017

Acknowledgements

First, I would like to thank my two supervisors, Stein Gjessing and Yan Zhang, for suggesting the topic of the thesis and for their guidance and advices during our meetings.

In addition, I also wish to thank Sverre Holm for allowing me to borrow the ST-8850 Sound Level Meter, which proved to be useful for understanding the phones' measurements.

Finally, a big thanks goes to my family and my colleagues for their continuous support and encouragement in difficult times.

Abstract

Noise. It is around us, everywhere, all the time. It is hard to control, hard to shut out. It affects our health, sometimes even without us noticing it.

Noise is one form of pollution, and is causing annoyance and other health effects. Therefore, the Norwegian government set a national goal to reduce noise pollution from 1999 levels by 10% until 2020, but this process is not going as well as it should. The motivation of this project is to help reach the goal by highlighting areas that need most attention.

The goal of this thesis is to explore a novel approach to measuring outdoor noise levels, with better spatial and temporal coverage. In the current approach, professionals use high quality equipment to measure a smaller area, then extrapolate from this data. This method potentially ignores other noise sources and amplifying or attenuating effects of the environment. In addition, it is quite costly.

In contrast, my approach delegates the task to local citizens, and leverages the powerful and ubiquitous smartphones. An application runs in the background and repetitively measures the noise level, processes and uploads it to a server. There it is aggregated and displayed as a noise map to anybody interested.

Sammendrag

Støy. Det finnes rundt oss, overalt, hele tiden. Det er vanskelig å kontrollere, vanskelig å lukke ut. Det påvirker vår helse, noen ganger uten oss å merke det.

Støy er en form av forurensing, og forårsaker irritasjon og andre helseeffekter. Derfor vedtok Regjeringen et nasjonalt mål om å redusere støyplagen fra 1999-nivå med 10 prosent til 2020, men denne prosessen går ikke så bra som den burde. Motivasjonen for dette prosjektet er å bidra til å nå målet ved å markere områder som trenger mest oppmerksomhet.

Målet for masteroppgaven er å utforske en ny tilnærming til måling av utendørs støynivåer, med bedre romlig og tidsmessig dekning. I dagens tilnærming bruker profesjonelle utstyr av høy kvalitet til å måle et mindre område, deretter ekstrapolerer fra disse dataene. Denne metoden ignorerer potensielt andre støykilder og forsterkende eller dempende effekter av miljøet. I tillegg er det ganske kostbart.

I motsetning, tilnærmingen min delegerer oppgaven til lokale borgere, og utnytter de kraftige og allestedsnærværende smarttelefonene. En app kjører i bakgrunnen og måler lydnivået, prosesser og laster det opp til en server. Der blir det samlet og vist som et støykort til alle interesserte.

Összefoglaló

Zaj. Körülvesz minket mindig és mindenhol. Nehéz kordában tartani, kizárni. Hatással van az egészségünkre, néha anélkül, hogy észrevennénk.

A zaj a szennyezés egyik formája, kellemetlenséget és egyéb egészségügyi problémákat okozhat. Ezért a norvég kormányzat kitűzött egy nemzeti célt, hogy az 1999-es értékről 2020-ig 10%-al csökkentsék a zajszennyezést, ami sajnos nem halad olyan tempóban, ahogy kéne. Jelen kutatás motivációja, hogy segítsen elérni ezt a célt azáltal, hogy rámutat a legtöbb figyelmet igénylő területekre.

A diplomamunka célja, hogy megvizsgáljon egy újfajta eljárást a kültéri zajszennyezés mérésére, mely nagyobb térbeli és időbeli lefedettséget nyújt. A jelenlegi módszer abból áll, hogy szakemberek dedikált eszközökkel felmérnek egy kisebb területet, majd ebből extrapolálással fednek le nagyobb területet. Ezzel a módszerrel előfordulhat, hogy bizonyos zajforrások és egyéb befolyásoló tényezők kimaradnak a számításból, emellett meglehetősen drága is.

Ezzel ellentétben az én módszerem a helyi lakosokat bízza meg a feladattal, és kihasználja a mára mindenhol jelen levő okostelefonok adta lehetőségeket. Egy alkalmazás a háttérben rendszeresen megméri a zajszintet, ezt feldolgozza, majd feltölti egy szerverre. Itt összesítésre kerül, majd egy zajtérkép formájában bárki érdeklődő megtekintheti.

Contents

List of Figures

List of Tables

List of Snippets

Summary

Environmental noise is causing more and more problems in developed countries, including Norway. The Norwegian government set a national target to reduce environmental noise annoyance by 10 % from the 1999 levels by 2020, but this process is not going as expected.

This thesis seeks to explore a novel way to measure and map noise levels, in order to make it easier for stakeholders to locate areas need action the most.

Current methods are both expensive and have certain drawbacks, e.g. limited coverage. The new method explored in this work uses the citizens' smartphones to measure noise levels, together with location and some other parameters. This data is then uploaded to a central server, where it is visualized as a heatmap, and further analysis is possible.

The smartphone application is running in the background, and regularly records a short noise snippet, without further user interaction. This sound file is then processed – maximum and average loudness is calculated – and the results are uploaded to the server. To minimize the privacy concerns, no actual sound recording is leaving the device, only the metadata and processed values are uploaded.

On the webserver, measurements are clustered (geographically), then data in each cluster is averaged before being displayed. Due to the high number of different Android devices and built-in microphones, it is very difficult to acquire accurate noise levels (in dB). To work around this, the preferred metric is not the "actual" decibel value, but the "deviation from average". This is calculated by grouping the readings from the same microphone (i.e. by device and by internal microphone or headset), calculating the average for each group, then subtracting it from the actual value. This yields a relative noisiness map, which highlights the areas that are loudest.

Results from experiments show that while there are many influencing factors, in general it is possible to produce a noise map which locates noisier and quieter areas. With more contributors, even better accuracy and coverage can be achieved.

Chapter 1

Motivation

With the increasing level of motorization and constant constructions, there is a high number of people in Norway exposed to unhealthy amounts of noise. According to the Norwegian Environment Agency [22], close to 1.4 million people – more than one quarter of the population – have to endure noise levels above 55 dBA around their homes. Researches show that this is the borderline between tolerable and harmful background noise levels, and prolonged exposure to more noise has negative effects on health [4].

Consequences can be sleep disorders, hearing impairment, stress, reduced performance in work and school, etc. Therefore, the Norwegian government set up a national goal to reduce noise pollution by 10% from the 1999 levels until 2020 [31]. According to the Norwegian Environment Agency, the process is not going as expected [32].

The European Commission also states in its report, that noise pollution is not treated as seriously as other type of pollutions, while reports are warning about the increasing effect it has on people [13].

The purpose of this thesis project is to research the feasibility of involving citizens in the measurement of noise pollution, using their own mobile devices and leveraging crowdsourcing techniques to organize the work. The expected outcome is a proof-of-concept software system, which enables the measurement and visualization of background noise levels in cities, thus helping stakeholders focus on the most affected areas.

1.1 Measure

Current measurements of noise levels are relying on expensive professionals visiting the selected area and taking measurements with dedicated devices during a limited time interval. Noise maps of the area are then generated by extrapolating from the collected data points. Due to the nature of extrapolation, such information has the potential to be incorrect, or inaccurate.

Another way to measure the noise levels in a certain area is to install dedicated sensors spread out in the area, forming a Sensor Network, and continuously gather data. This approach can still be costly, since

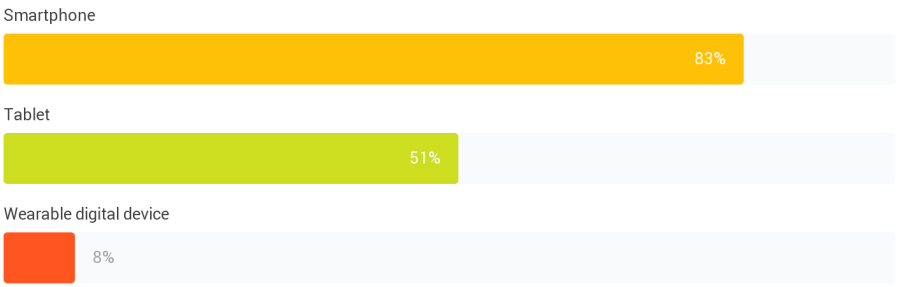


Figure 1.1. Which devices do people use?

the sensors need to be numerous, especially if the area is big, or high-resolution input is required.

There is, however, an already existing and highly numerous sensor network: the population and their ubiquitous smartphones. As seen on Figure 1.1, according to Google Consumer Barometer, Norway has a smartphone penetration of 83%[1]. If a significant number of smartphone users can be reached, it is possible to employ their devices using crowdsourcing techniques to record, analyze and collect noise data from their environment. Smartphones are equipped with decent quality microphones, many other sensors, powerful processors and are connected to the Internet regularly, if not all the time. These characteristics enable them to be a good candidate for being used as sensors for studying noise pollution.

Challenges

Meanwhile, there are several challenges to be addressed when it comes to actually implementing this theory. Probably the most important is that phones are usually not in an ideal position for clear noise recording. They might be stored in a pocket or a bag and muffled by clothes or a case, which all degrade the quality of the record, and need corrections.

Another issue is private life. People are very sensitive to breaches of their privacy, and are reluctant to install and use an application if they feel it can harm them. One thing is to avoid leakage of recorded sound from the device; this is where the good computational power of current smartphones can help. Actual recorded sound should never leave the device, instead it has to be processed in place, and only metadata must be shared. The other part of this challenge is to convince the user that their privacy is not violated, and it is completely safe to use the software. This has to be done by careful marketing, trust and openness.

[1] Question asked: Which, if any, of the following devices do you currently use? Total Respondents: 1000 Base: Total online and offline population Source: The Connected Consumer Survey 2016 https://www.consumerbarometer.com/

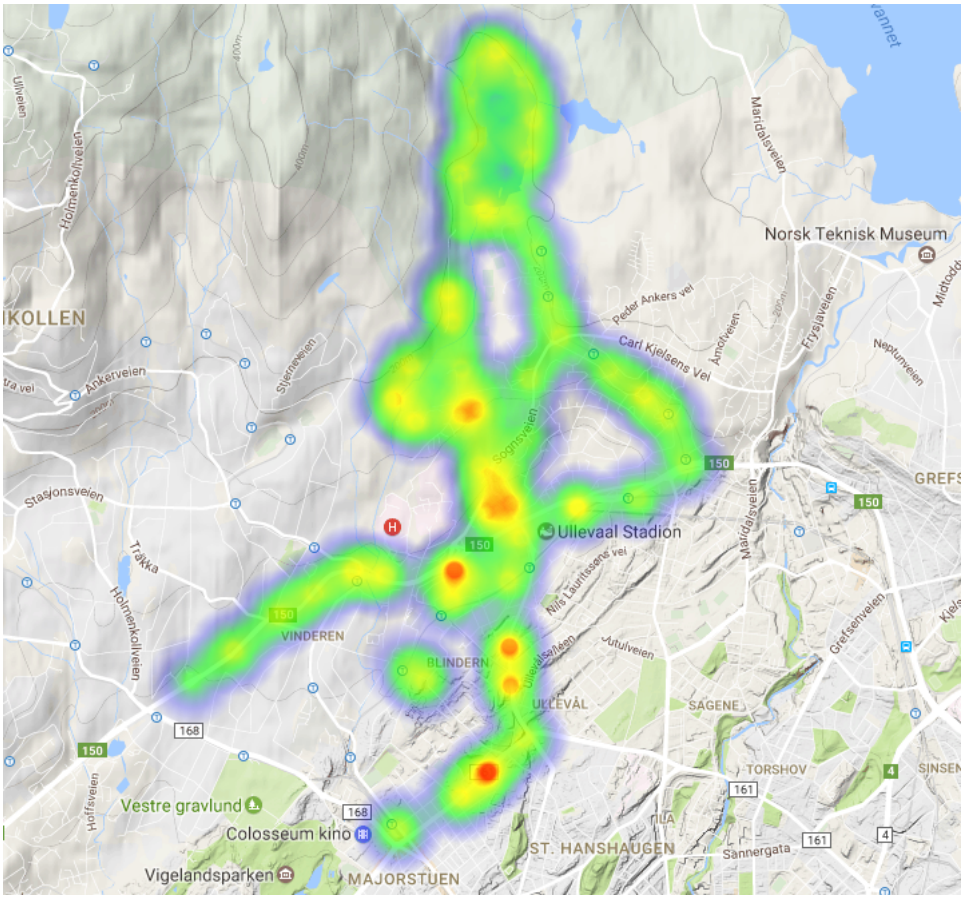


Figure 1.2. An example heatmap

1.2 Visualize

For the research to be useful, collected data has to be visualized somehow. One popular option is to display the noisiest areas as a *heatmap* over an ordinary map of the area. To achieve better understanding, the data being displayed should be filterable and customizable, for example by time, area, number of sources. An example heatmap is shown on Figure 1.2.

The visualization is ideally accessible online, both from computers and smartphones; within the application and in further processable form for researchers and decision makers.

An additional feature could be to integrate noise pollution data with other sources and display them together, making possible correlations easier to discover.

Chapter 2

Noise pollution

Many definitions exist for pollution. For example, in the Norwegian Pollution Control Act, it is defined as follows.

> **1)** the introduction of solids, liquids or gases to air, water or ground,
>
> **2)** noise and vibrations,
>
> **3)** light and other radiation to the extent decided by the pollution control authority,
>
> **4)** effects on temperature
>
> (from [10, Chapter 2 §6])

Of these four categories, in this thesis I focus on the second one: *noise*. The Oxford Dictionaries article on noise defines it as "A sound, especially one that is loud or unpleasant or that causes disturbance." ([21]) In general understanding, noise means unwanted, disturbing, even annoying sound effects, usually caused by human activities. This includes traffic, industry, construction, entertainment, or alteration of the natural environment such that it makes more noise than before (e.g. building/destroying something that makes the wind howl). These are classified as *environmental noise pollution* and make up the main subject of the thesis. However, there is another, very important category: *work-related* or *occupational noise*, but that is already better regulated and well-known, thus out of scope.

In this chapter I first present the various sources of noise pollution, along with how they affect people, then in the second section, I look at the health-related effects noise has on humans.

2.1 Sources of noise pollution

Unwanted noise can come from many sources. Some are affecting many people and are ubiquitous – like traffic – while others are more restricted to specific areas – like industry and entertainment – but all of them are equally important to research and regulate. Below I briefly explain some properties of possible noise pollution sources.

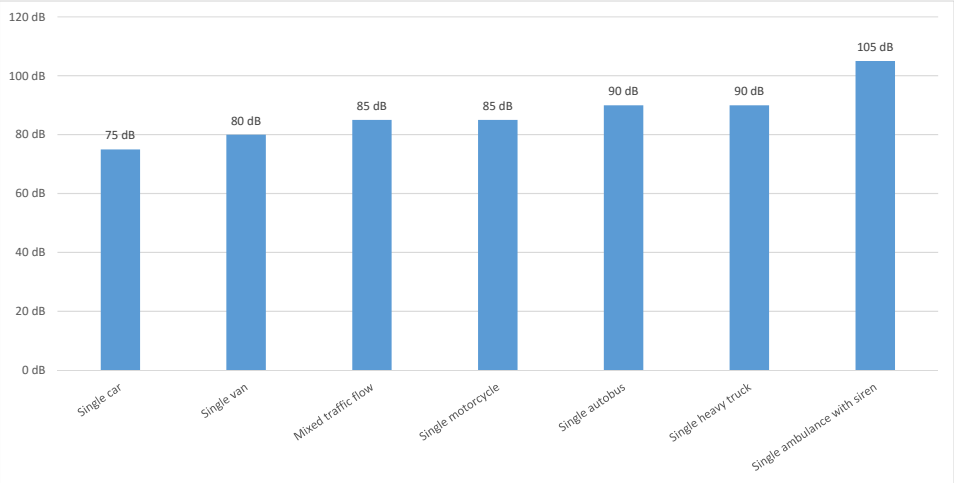


Figure 2.1. Noise level by traffic type

Traffic

Noise pollution caused by traffic can be divided into three main categories: road, railway and air traffic. Noise caused by road traffic is a significant contributor to noise pollution, according to the Norwegian Environment Agency it is the most dominant [22]. The reason for its high penetration is straightforward: there is a road to almost all houses and vehicles ply regularly on most of them thus bringing the noise source very close to people. Railway and air traffic, while usually louder, is more restricted to certain areas and thus contribute less to the overall noise pollution if measured by the number of affected people.

The overall noise level from road traffic consists of several factors. Probably the most important one is the composition of the traffic flow. As shown on Figure 2.1, trucks, buses are noisier, while hybrid and electric models emit less noise than average cars. The speed of the vehicles is also important, as overall emitted noise level increases logarithmically with speed [24]. Other important factors are tire and pavement type and meteorological conditions. Researches show that at lower speeds the main source within road traffic noise is the engine and exhaust, while above a certain speed the tire and aerodynamical effects become more dominant [5]. Figure 2.2 summarizes this.

Industry and construction

Industrial noise affects mostly those who are working inside the factories, and usually has a lesser impact on those living nearby. The exceptions are such factories where the work is done outside of a building, e.g. ship-building. The noise levels in factories are usually higher than those produced by traffic, but affect less people, who in turn presumably have proper protective devices (such as earplugs or earmuffs) provided by their employer. Also, noise exposure at workplaces is highly regulated by law.

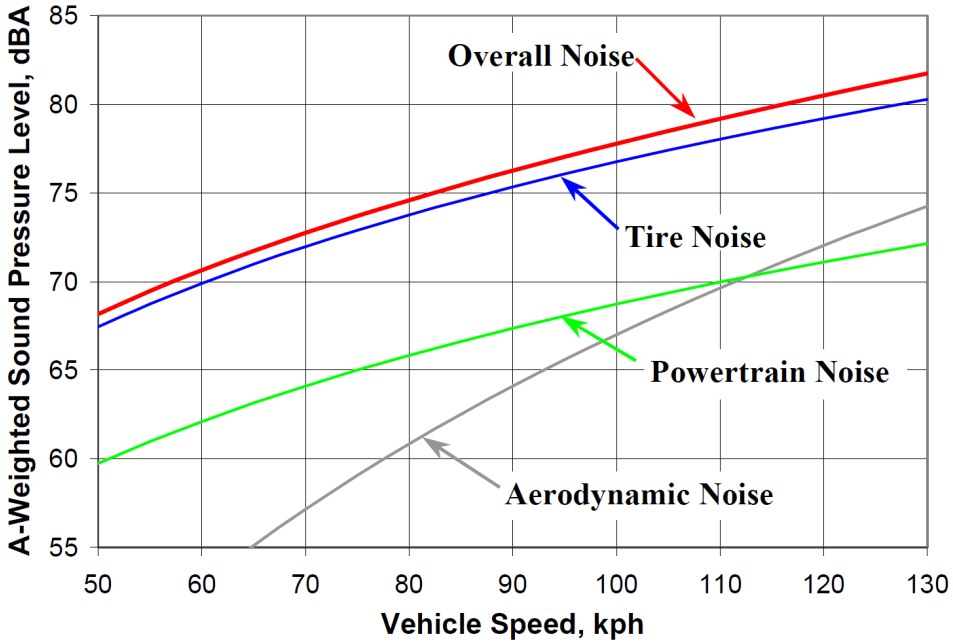


Figure 2.2. Contributions of the various sub-sources of highway traffic noise [3]

Noise from construction sites affects more people than industrial noise and is more difficult to avoid, since constructions can happen anywhere, while the locations of factories are usually well known. On the other hand, a factory is usually a long-lasting source of noise, while constructions are generally finished within a year or two. This means that while temporarily construction sites can emit more noise, overall, they affect people's health less.

Entertainment

It is hard to argue against the necessity of entertainment, but not every form of it is welcome by everybody. Concerts, open-air theaters and cinemas, festivals usually go hand in hand with loud music or audiovisual effects. I have personal experience with this, as in the past few years the local government in my home city decided to develop an already existing park just a few hundred meters from my house, and organize all the local festivals there. Loud music from the festivals caused many sleepless nights for my little sisters.

Another example of noise pollution from festivals are the constant arguments against a big festival in Budapest, the SZIGET Festival. It takes place on an island in the Danube close to the citycentre of Budapest. Residents around the area constantly complain about the inconveniences it causes each year.

2.2 Effects of noise pollution on health

In this section I discuss the possible health-related effects of noise pollution, highly relying on WHO's study in the topic [35]. In general, and especially at night, it is advised by the WHO not to exceed an annual average of 40 dB outdoors noise level. If living in noisier environment, the following diseases have increased probability.

Cardiovascular disease

Cardiovascular diseases (CVDs) is a common name for anomalies of the heart and blood vessels. Examples are heart attack, stroke, venous thrombosis and heart failure. Most CVDs occur due to atherosclerosis[1], which in turn can be caused by hypertension[2]. It has been shown, that traffic noise can contribute to the development of hypertension.

The study estimates that a 61,000 DALYs[3] are lost in Europe due to cardiovascular diseases caused by environmental noise.

Children's cognitive impairment

Studies show that noise exposure has a particularly harmful effect on children. They have decreased concentration and attention abilities, and can have problems with understanding and processing speech, or written text [30]. If the exposure happens during a critical period, it can lead to lifelong impairment [34]. An important remark in the WHO study is that the reduction of cognitive ability appears both during the noise exposure and "for some time after the cessation of the noise exposure".

A frequently cited experiment took place in Munich, during the period when the main airport was shut down, and traffic was moved to a new airport, somewhere else [12]. The first test took place while the old airport was still operational while the second and third tests took place when the old airport has already been closed, and the new one was operating. The research involved groups of pupils from the neighborhood of both airport locations and control groups.

Results showed that having an operational airport close to the school had negative effects on the long-term memory and reading comprehension performance of the pupils. An important finding was that during the follow-up period, cognitive capabilities of pupils from close to the old airport were restored, while the same deficits appeared with pupils from close to the new airport [35, p. 46].

[1] When an artery wall thickens due to accumulation of white blood cells, and the blood vessel's throughput is reduced.

[2] high blood pressure

[3] "DALYs are the sum of the potential years of life lost due to premature death and the equivalent years of 'healthy' life lost by virtue of being in states of poor health or disability." ([35])

Sleep disturbance

Even while sleeping, the human auditory system registers noise, and the brain processes them. This is a natural and desirable process: people are more vulnerable when asleep, and reacting to noise (such as: predators moving closer, burglars fiddling around, or sounds of fire) is essential to avoiding danger. Whenever we hear a noise, there are responses from the body, such as increased heart rate or change in the sleep structure. Increased occurrence of these effects decreases the restorative power of sleep, resulting in fatigue, reduced concentration, irritability and stress.

The WHO study estimates the impact of sleep disturbance caused by environmental noise to be around 903,000 DALYs for the EU population.

Tinnitus (ear ringing)

"Tinnitus is defined as the sensation of sound in the absence of an external sound source." In other words: the inability of perceiving silence. Tinnitus can be caused by excessive noise exposure, such as loud headphones, gun shots, concerts and firecrackers. Being a disorder on its own, it can also cause sleep disturbance, anxiety, reduced cognitive abilities, frustration and reduced efficiency.

Tinnitus is often following NIHL[4], but is not exclusively tied to it, and can happen to people exposed to excessive noise without any damage to hearing abilities.

Annoyance

According to the Constitution of WHO, being healthy is more than what we generally think. It means "a state of complete physical, mental and social well-being and not merely the absence of disease or infirmity" ([37]). High level annoyance clearly doesn't fit this definition, so it can be mentioned among the health effects of environmental noise. Environmental noise induced annoyance can contribute to or cause anger, anxiety, exhaustion, dissatisfaction or depression, tiredness and stress, among others.

2.3 Quantifying noise

Throughout the thesis certain noisiness levels are mentioned, but usually people find it hard to understand what the numbers actually mean for them. Table 2.1 shows an approximate mapping between everyday sounds and their respective sound pressure in dB_A. It is important to mention, that due to the logarithmic nature of the decibel scale, a 6 dB increase means doubling the sound pressure, and a 10 dB increase corresponds to "twice as loud" (in psycho-acoustic terms).

[4]Noise-induced hearing loss

Table 2.1. Mapping dB_{SPL} values to everyday sounds

Everyday sound example	**Sound Pressure (dB_A)**
Breathing (quietest audible sound)	10
Whisper, rustling leaves	20
Quiet rural area	30
Library, bird calls, quiet urban ambient sound	40
Quiet suburb, conversation at home	50
Conversation on restaurant, office, background music	60
Passenger car at 105 km/h (from 7 meters), vacuum cleaner (annoyingly loud to some)	70
Garbage disposal, freight train (from 15 meters), diesel truck at 64 km/h (from 15 meters), food blender	80
Big aircraft before landing (from 1.8 km), power mower	90
Aircraft take-off (from 300 meters), jackhammer (serious damage possible over 8 hours)	100
Auto horn (from 1 meter), live rock music	110
Thunderclap, chain saw (painful)	120
Jet take-off (from 25 meters) (eardrum rupture)	150

(based on [7])

Chapter 3

Crowdsourcing

The term *crowdsourcing* is widely used nowadays, but there are many different definitions and meanings associated with it. In this chapter I elaborate on what crowdsourcing means and then I look at some well-known projects that claim to use crowdsourcing, and explain their interpretation of the term. In the last section, I list some arguments for the usage of crowdsourcing techniques in this project.

3.1 Definition

The word *crowdsourcing* is a portmanteau of "crowd" and "outsourcing", meaning that a certain task is outsourced to the crowd. Another definition is "it enlists a crowd of humans to help solve a problem defined by the system owners" ([9]). Very often the crowd is a group of enthusiasts or volunteers, and they are not paid in return for their work; the sole compensation being the joy of contributing. While outsourcing usually means employing a well-defined group of experts, in return for a payment, with crowdsourcing the crowd is usually unknown and very diverse.

3.2 History

The term crowdsourcing was coined in 2006 by Jeff Howe in a Wired article [29] and generally refers to online activities. However, campaigns that can be classified as crowdsourcing has happened before too. Some examples are:

- **1714 – The Longitude Prize** Since Columbus discovered America in 1492, transatlantic voyages become more and more frequent. At that time, however, sailors had no reliable means to determine their longitude when land was out of sight. This lead to uncertainty and sometimes tragedies, for example the loss of four British warships in 1707 near the Isles of Scilly [26].

  This, among other events made the British Parliament establish a prize for finding a reliable way to determine a ship's longitude on

sea. Many proposals were submitted, the most famous being John Harrison's chronometers.

- **1884 – Publication of the Oxford English Dictionary** 800 volunteers catalogued words to create the first fascicle of the OED [1, 23].
- **1916 – Planters Peanuts contest** When Planters, the producer of Planters Peanuts needed a new logo, they created a contest to design it. The 14 year old schoolboy Antonio Gentile submitted the winning artwork – a humanized, walking peanut [20].
- **1997 – US tour for Marillion** The British rock band Marillion lacked sufficient funding for an US tour, but their fans raised $60,000 to make it possible [19].
- **2001 – Launch of Wikipedia** The "free-access, free-content Internet encyclopedia" as they call themselves was launched, and started its growth. The articles are written by enthusiasts, and anybody can contribute [36].

As the above list shows, crowdsourcing can have many forms: Finding the solution to a technical problem, gathering information, designing a logo, or raising money to fund something – these activities all involve work done by an anonymous crowd. With the appearance of GPS- and Internet-enabled smartphones, new horizons opened up.

3.3 Contemporary applications

Many start-ups were founded based on the advantages crowdsourcing promises, and they all use the term a bit differently. In the following I present some examples.

Waze

Waze is a GPS-based, turn-by-turn navigation app for smartphones. It is free to use, and available to everybody, although the usefulness is limited by the number of active users in the area. This is because the main power of Waze lies in the users who – by using the service – automatically report their position and speed to the server that uses this data to calculate the average speed of traffic on each road, and provides visual feedback by color-coding the roads (red: standstill, orange: heavy traffic, no color: normal traffic). Waze users also have the possibility to report road conditions like construction work, accident, road defects, dangerous items on the road, etc.

This is already a very useful feature, but they took it a step further, and combined this information with flexible turn-by-turn navigation: If sufficiently many users report slow traffic on your projected route, Waze offers an alternative. And based on my experiences in Budapest, Hungary, Waze is more often right than not in offering a faster route.

Wikipedia

Wikipedia is a community driven, free, online encyclopedia. There are two ways to create an encyclopedia: One either hires a lot of writers, editors and researchers, or lets *everybody* create and edit articles. Wikipedia went by the second approach, and became immensely popular. According to the article about themselves[1], they claim to be comparable in accuracy to Encyclopedia Britannica [15].

The concept of crowdsourcing takes place by letting everybody edit the articles *and* providing an interface for discussing problems and inputs to the articles.

Open-source software

A well-known example of utilizing the knowledge and workpower of the crowd is *open-source software*. It means that the source code and documentation of a software product is freely available to everybody, and is free to reuse, rewrite, modify, distribute and even monetize[2]. Notable examples are the various distributions of the Linux operating system and many of the applications and utilities packaged with them, the Android operating system, the Chromium and Mozilla browsers, the Eclipse platform and IDE and many more.

It is often debated whether open-source software products are safer, more secure than their proprietary counterparts. I believe there is no proper answer to this question. There is one aspect though, in which open-source projects usually excel over others: response time for bugs and vulnerabilities. Usually, since there are much more people working on the software (even if only part-time), patches to open-source products come much faster (sometimes within hours of discovery), while proprietary update mechanisms can make up for days of delay. And one non-negligible advantage is that if I feel confident enough, I can change or fix things in the source code, which might, or might not get included, but I can use it nevertheless. This is really hard with closed-source software, where all we get is a binary file.

reCAPTCHA

People who use the Internet frequently have to interact with forms. Most of the time it would be harmful if bots – computer controlled programs – could also use the same interface (for example a much faster bot booking a last-minute travel, making it inaccessible for real people). To prevent this, *CAPTCHA* was invented. The term can refer to any kind of problem easily solved by humans, but difficult for computers, e.g. answering a simple question, matching images, or, the most popular, recognizing some – possibly distorted – text.

[1] https://en.wikipedia.org/wiki/Wikipedia

[2] of course, different licenses may change these rights

Luis von Ahn, Ben Maurer, Colin McMillen, David Abraham and Manuel Blum at Carnegie Mellon University took this concept a step further, and combined it with a crowdsourcing mindset. The main problem they wanted to solve was that people spend on average ten seconds solving a captcha, which is of course useful in terms of filtering unwanted automated access, but otherwise wasted. People are spending hours solving problems that are currently impossible for computers, but humanity does not gain anything from this effort. They wanted to change this, and the result is *reCAPTCHA*. Their version presents two strings to the user: one is the actual test, with a known answer, but the second is a chunk of a scanned text, or an image that the OCR[3] software failed to recognize. The users solving the reCAPTCHA can prove that they are not robots by entering the correct answer for the first string, and contributing to the digitalization of millions of scanned books by typing in what they see on the second image [16].

The system uses words that both of two state-of-the-art OCR software failed to recognize, and employs a so-called voting system to make sure the accuracy of human recognition is the best. In their research the inventors claim that books digitized with the help of reCAPTCHA reach the quality of those processed by professional transcribers [33].

Crowdfunding

Crowdfunding is a special case of crowdsourcing, where participants (the crowd) use their money instead of their knowledge or work to contribute to a case. Anybody can come up with a product or an idea that they want to realize, but lack the appropriate funds to do so. They can create a motivating videoclip and some textual introduction, upload them to a designated site (e.g. Kickstater, GoFundMe, IndieGoGo), and start to distribute it among friends. If it is a worthy project, it goes viral soon, and if people find it interesting, they can support it with a small amount of money in return for future products or equity.

The way most crowdfunding projects work is the following: The creators make a campaign, specifying the amount they need in order to reach their goal – starting/continuing production, making a movie, etc. – and set up so called "perks", which funders can buy to support the project. These perks are usually a special price for a variation of the actual product, which funders who chose that perk will receive.

A good example for a crowdfunded project that had much success is the Hungarian programmable robot for children: Codie[4]. The idea was to create a small tank-like, but cute robot with eight different sensors, that could be programmed for several simple tasks on a smartphone or tablet. The main motivation was to prove that programming can be fun and can be done even by small – 8-10 years old – children too. Their crowdfunding campaign was successful, according to the campaign page, 668 people raised $96,306 which is 115% of the original goal.

[3] Optical Character Recognition

[4] IndieGoGo campaign page: https://igg.me/at/codietherobot/x

However, Codie is also a good example of the dangers in crowdfunding. The Codie team consists mostly of students who have little experience in making a business successful. Right now, they are in a slight crisis, not being able to deliver more after the first 100 robots. Codie is among the more successful campaigns, but funders always have to consider the risks of a failed project, in which case they might or might not get their money back, based on the different crowdfunding strategies.

Weather reporting and forecast

Crowdsourcing is especially powerful when the task is to cover a bigger area, and report some condition. A Hungarian startup, Időkép[5] (approx. "Skyview") took advantage of this fact and created a weather forecast site very popular in Hungary.

In the beginning, the main source of information was user contributions, either in the form of automatic personal weather stations' data, or manual reporting of the current "sky-view" (that is, mostly cloud coverage). As the site grew, they could afford to use more powerful servers, and even deploy their own, private precipitation radar network covering the whole country. This led to highly accurate forecasts, wider reputation and appearance in online news sites. Nevertheless, crowd contribution is still an important part of their data.

Another aspect in which they rely on the power of people is reports during extreme weather events. When there is an expected high intensity phenomena (e.g. spring thunderstorms, wind, heavy rain, snowstorm), they start a "minute-by-minute" section, and people from all over the country can send in reports, pictures, and also official announcements are included, which makes the platform an excellent source of information.

Image analysis

Making a computer analyze an image and identify what is pictured is a very tough problem, with no reliable solution known so far. However, the human brain has amazing capabilities for recognizing what it sees on an image. This is what NASA scientist also realized, so they started the Stardust@home[6] project. Previously, they sent a spacecraft to approach a comet and collect microparticles from its dust tail, and also interstellar dust in a special material called Aerogel. What they forgot to take into account is that upon return to Earth, this collector will have suffered other mechanical impacts and injuries, making it difficult to locate micron-sized particles and their traces.

To solve the problem, they asked the crowd to help: they scanned the whole surface of the Aerogel with an automated microscope, made the resulting images available to registered users, and asked them to find suspicious areas, which might hide interesting particles. They expected to

[5] https://www.idokep.hu/

[6] http://stardustathome.ssl.berkeley.edu/about/stardusthome/

collect around 45 particles in the whole 1,000 cm^2 surface, which, given the tiny size of the particles makes up for a very tedious task.

Another example of people contributing for no apparent reward is the countless screenshots from Google Earth uploaded to the Internet, depicting funny, mysterious or weird things captured by the satellites and aerial imagery. This kind of activity also requires humans, as most pictures do not mean anything to computers and are hard to search for using them.

3.4 Why crowdsourcing?

One of the basic ideas for this thesis is to leverage the power of the crowd, and use it for our own purposes. This definitely comes with many benefits, but also requires some conditions to be true. And naturally, there are drawbacks, even dangers when dealing with crowdsourcing. In this section, I explain these topics in more detail, first the properties of crowdsourcing in general, then how it applies to this project.

3.4.1 Benefits of using the crowd

One significant trait of crowdsourcing is its low financial requirements. In other words, using the crowd is relatively cheap. The reason for this is the fact that participants generally do not expect or require financial compensation for their contribution. They do the work required by the project in their free time, driven by the motivation that they help achieve a goal important to them.

This leads us to another important concept: enthusiasm. Since people are personally interested in the outcome of the project, they are more likely to do better quality work than if they were just randomly picked and asked to do something they cannot relate to.

In the age of the Internet, reaching the target group became easy. There are several sites offering a platform to host (mostly crowdfunding) projects, e.g. Kickstarter[7], GoFundMe[8], Indiegogo[9]. These sites are well known, attract many visitors, and many successful projects could succeed from the money raised through them. They also make it possible to reach many potential contributors, increasing the success rate.

In this thesis, crowdsourcing was among the main selling points when creating the project description. From the contributor engagement perspective, it comes natural, since the target group is the people living in a certain area, and they are also the group that suffers from the effects of noise pollution and could potentially benefit from the results of the measurements. Thus, they are directly motivated to participate and contribute.

[7] https://www.kickstarter.com/
[8] https://www.gofundme.com/
[9] https://www.indiegogo.com/

3.4.2 Preconditions/Challenges of using the crowd

The power of crowdsourcing does not come free, there are certain requirements a crowd-based project must meet in order to be successful. Below I list some of these requirements and possible strategies to meet them.

Building and sustaining the user base

In order for the crowdsourced project to be successful, there has to be enough people who contribute. One way to reach this goal is to directly pay them in return for their contributions. This is not common, since we want to keep the costs low. If we don't want to pay, we need to find other ways to engage contributors, e.g. with good marketing, offering non-material rewards, or simply by relying on their good conscience to help the cause.

A possible technique to sustain the user base, and potentially even increase it, is to introduce gamification aspects. These can be as simple as counting the number of contributions, and showing it as the "rank" of the contributor, and can become a competition between users.

Besides providing constant input from users, a competitive approach can be used to treat contributions differently, for example assigning higher credibility to long time users, potentially allowing a verified calibration of devices, etc.

In this thesis project, we assume that users will contribute either because they themselves are affected by the negative effects of noise pollution, or they realize it is an issue, and understand that it needs addressing.

Also, since the steps taken during measurements are already recorded, it could be exposed to the user with relatively little effort. This information could act as a motivation for a healthier lifestyle by promoting walking over other means of transport.

Partitioning the task

Besides having enough people, it is also important that the task can be partitioned into small parts that individual contributors can take care of, without needing to grasp the whole picture. Measurements, like the one in this thesis are a trivial example, since one contribution consists only of one measurement and some metadata, which is then collected centrally, and analyzed as part of a bigger dataset.

3.4.3 Risks of using the crowds

When dealing with contributions coming from so many – potentially very heterogeneous – sources, there are inevitably some risks, even dangers that the project organizer must take into consideration.

Arguably, the most significant is the quality of the contributions. There could be people who use their professional-level skills or equipment to participate, which yields high-quality results, and there could be

results of poor quality, submitted by enthusiastic but less knowledgeable participants. It is then the project organizer's responsibility to filter out the inputs that could bias the overall result, and potentially give higher weight to better quality inputs.

When the project grows sufficiently large, it might start attracting malicious users, who intentionally submit false/biased results, or even try to destroy already collected data or the infrastructure. This can be avoided or contained by applying filters on the collected contributions.

Chapter 4

Related work

In the introductory chapter, it is briefly mentioned that the European Commission acknowledges that noise pollution has been prioritized lower than other forms of pollution, both in research and regulation [13], and proposes to change this situation.

According to [11], EU member states have to determine the population's exposure to environmental noise, using noise mapping techniques. Motivated by these governmental initiatives and directives, several studies have been conducted, targeting the creation of an affordable measurement method.

4.1 The widespread measurement method

Current official assessments of environmental noise levels usually mean measurements performed in key areas (e.g. busy roads, intersections) for a relatively short time (1 hour – 1 day). This data is then fed into an algorithm which – taking the characteristics of the area into account – extrapolates from these data points, and gives a synthetic, estimated noise map of a bigger area.

This method has multiple disadvantages. The measurements are concentrated within a limited time frame, and this can cause errors in the data. For example, on the day of measurement, a construction is happening on a road (other than the one being measured), which drives more traffic than usual through the road being monitored. This increased traffic results in more noise, which is registered in the measurements. Over a longer timeframe however, traffic might be significantly less, meaning that the data is now biased.

Another weakness of this method it that it does not scale well. Extending the coverage to bigger areas means increasing the number of measurement points, which in turn means more resources. Also, the sound level meter devices used in such measurements are quite costly. Some cities might not be able to afford such expenses.

4.2 New method: Wireless Sensor Networks

Several studies [6, 18, 25] already realized the above mentioned shortcomings of the currently used measurement methods, and researched other, more advanced technologies. One such technology is the utilization of Wireless Sensor Networks (WSNs). WSNs benefit from the relatively low price and easy accessibility of small, limited capacity computing units and widespread availability of wireless network access (WiFi, cellular).

Their main idea is to develop small, self-contained units, that are capable of registering noise levels, then have them transfer the data to a central server, where the processing and visualization happens. This way, the units can remain relatively cheap, meaning that a larger number of them can be deployed, resulting in higher granularity or bigger spatial coverage.

As seen in [25], this approach also has its challenges. One such is the calibration of the units. Unless some simple and reliable calibration method can be developed, the individual units' readings cannot be properly trusted.

4.3 One step further: People-Centric Sensing

In [6], Campbell et al. introduce the term "people-centric sensing", and define it as "humans, rather than trees or machines, become the focal point of sensing and the visualization of sensor-based information is for the benefit of common citizens and their friends, rather than domain scientists or plant engineers". This means that everyday people are making efforts to understand their environment, and – as a by-product – collect data that also scientists and decision makers can use.

Maisonneuve et al. present in [18] a platform called NoiseTube, which aims to revolutionize noise mapping by employing participatory sensing principles. They explore the possibility of using the citizens' ubiquitous mobile phones to perform noise measurements while tagging them with metadata (e.g. GPS location, subjective annoyance).

Maisonneuve et al. also mention an important concept – which I briefly discuss in section 3.4.2: engagement of the participants. They argue that by enabling any citizen to measure their exposure to environmental effects, general awareness can be increased, and this can lead to more effective actions from the governments.

In [8], D'Hondt, Stevens, and Jacobs use the same platform to conduct a practical experiment. They begin with calibrating ten smartphones, then asking a group of citizens to perform well defined walks in a city while carrying the devices, which take measurements. They then aggregate the data and produce heatmaps of the area.

Their research shows that there is a potential in participatory sensing to become an alternative or a complement to the current practices, especially when using calibrated devices. The noise map they generated from the

collected data proved to be both valid and valuable, but not yet ready to entirely replace the conventional methods.

Chapter 5

Goals and project overview

In this thesis project, I explore the possibility of taking the experiment similar to the one conducted by D'Hondt, Stevens, and Jacobs in [8] to the next level. I seek to reach a higher level of granularity and spatial coverage by employing the personal smartphones of the citizens, leveraging crowdsourcing techniques.

In the following chapter I introduce the two main parts of the thesis: **1)** the software I developed to perform the measurements and aggregate the data, and **2)** an evaluation of the method and the results.

5.1 Software development

In accordance with the project's goal, an infrastructure consisting of an Android application optimized for performing the measurements and a central server dedicated to collect and visualize the data has been designed and implemented. Below I introduce these components, together with the task they need to solve, and in chapter 7 I elaborate on their implementation.

5.1.1 Measurement application

In these days, almost everybody owns a smartphone (see Figure 1.1), and they not only carry a useful communications and entertainment device with them, but also a powerful sensor array. In addition, most devices are at least intermittently connected to the Internet, which enables the sharing of the collected data.

One artifact of this thesis project is an Android application which can periodically take measurements, gather some metadata, process it, then upload the results to a centralized server.

Description of the application

The main function of the application is to run in the background on the user's smartphone, keep track of the immediate surroundings and position of the device, and regularly record a sound snippet, capturing the

environmental noise at a given place at a given time. The recorded sound file then has to be processed and quantified, so that it can be compared with other recordings. As discussed in section 8.2.2, this can happen at two places, but only one – on the device – is acceptable due to privacy concerns. Therefore, the app also needs to be able to analyze and process what it recorded.

Technical requirements for the application were more-or-less clear already at the beginning of the project. The app has to be (in no particular order):

- **Unobtrusive.** Since the application is not of any immediate utility to the user, it has to be as much out of the way as possible, so that it doesn't interfere with everyday use of the device.
- **Highly customizable.** Different people have different ideas about their role and willingness to contribute, therefore it has to be possible for the user to choose the degree of involvement in the project. For example, how much data they share, when the measurements can take place (e.g. only daytime), and how much resources they allocate on their device (sampling frequency).
- **Secure.** Since the data collected can contain potentially sensitive data, every communication and storage of data must be secured.
- **Adaptive.** The measurement environment can vary on a big scale, from enthusiastic users dedicating a device and holding it in an ideal position, to everyday users, using the device as normal, keeping it in their pockets, complicating the measurements and decreasing the value of data collected[1].

5.1.2 Central server and web interface

While it is not unimaginable that measurements made with a single device can be useful, the full power of the project shows itself when data from multiple – potentially many – devices are combined and can be analyzed together.

For this task, a publicly available webserver is a perfect candidate. Nowadays, server hosting costs are quite low, and there are many options to choose from. A second artifact of the project is therefore a webserver to complement the application, which is described below.

Description of the webserver

The webserver does not need to be very complex, and it has two main functions. One is to provide an interface towards the mobile applications for uploading data. This requires that there is a static, publicly available address, either a bare IP address, or a domain name. The app has to be

[1] See section 8.1

able to securely and reliably upload the measurement data to this server, therefore a secure connection is necessary.

The second function of the webserver is to provide access to the collected data, in alignment with [11], both for the contributors and other interested groups, such as researchers and government officials. For this purpose, a heatmap – generated from the collected noise pollution data – is displayed, overlayed on a regular map.

5.2 Evaluation of the methods and the results

Apart from implementing an infrastructure to assist the measurements and collect the data, the project has another objective: to evaluate the data collection method and the collected data in terms of usefulness for assisting decision makers in their fight against noise pollution.

5.2.1 Method for data collection

Data collection happens in a form of relatively short samples taken at regular intervals. In the development version both the sample length and the frequency of the measurements are adjustable within the app. The intended usage pattern of the application is to let it run all the time, during the day and the night, with a potential filter[2] to avoid indoor recording.

To achieve useful results, it is essential that the density of contributors is high enough in a given area. A high number of data points ensure that potential errors are averaged out, and spatial and temporal gaps are filled.

5.2.2 Analyzing the collected data

Once a sufficient amount of data has been collected, it has to be analyzed to see if the measurement method needs to be tuned. This is an iterative process, since after each change, new measurements have to be taken using the new method and then reevaluated.

In addition to adjusting the measurement method, the visualization logic has to be refined too. This includes the filtering and clustering of data points, and the way a single visualization point's value is calculated.

[2] i.e. no GPS signal

Chapter 6

The measurement method

One main goal of the project is to *measure* noise pollution in our environment. This however, has many aspects, and must be orchestrated on multiple scales. In this chapter, I explain the concepts of how the task of measuring can be organized among the users using crowdsourcing techniques, then the different methods of scheduling the measurements on individual devices, and finally, the methods I explored to quantify a recorded sound file.

6.1 Global scale: Crowdsourcing

As mentioned in the introduction of the thesis, it is possible to have a professional perform expensive measurements and generate noise maps by means of extrapolation; conduct the measurement using a dedicated sensor array; or let the people living in the area do it for us much cheaper. Environmental measurements carry the highest value if they are ubiquitous, that is, cover an area as thoroughly as possible.

Having everyday people doing their everyday business carry a sensor (a smart sound recording device, i.e. a smartphone in our case) are a good candidate for covering as big part of our target area (Oslo) as possible. Of course, just having people do their ordinary routine is not enough, they have to have the application installed and activated on their device. Challenges related to this are discussed in section 3.4.2 and chapter 8.

The thesis is driven by the question whether it is possible to outsource the task of taking the measurements to a wide number of people, who don't necessarily have any previous experience with the topic. They would go on with their everyday life and they would use their private smartphones. Thus, the resulting campaign should be easy to join, and the mobile application should be capable of unattended operation.

It has to be mentioned, that while using crowdsourcing techniques is a fundamental driver of the project, I did not have the necessary resources to involve other people. The measurements were taken by me, using two different devices, and walking around in both randomly chosen tracks and regularly visited paths, thus contributing repeated measurements from the same spot.

6.2 Macro scale: Scheduling on the device

In case of a dedicated sensor, noise would most likely be measured and processed continuously, and it would not pose a problem, sine the sensors would be properly supplied with power, and would be dedicated to a single task. When using personal smartphones however, we need to take into consideration that the device is most of the time running on battery power, and is being used for a variety of things besides measuring background noise.

To solve this problem, I decided not to have the recording running continuously, but rather start it regularly in discrete intervals, record a certain length, then stop until the next interval. For the implementation see section 7.2.2 and section 7.3.1. This reduces power consumption and resource usage, so the device can be used for other things simultaneously.

Given enough users of the application in a certain area, these discrete measurements can conglomerate into a continuous measurement, and, with even better coverage, provide multiple measurements from one point in space and time, allowing for statistically accurate values.

In a dedicated sensor, processing would happen immediately, without storing the recorded sound in a file. On the contrary, in my application, the recording and the processing steps are well separated. This is to enable deferring the processing to a later time, when more resources are available. It also means that recorded sound is stored on the device for an indeterminate time, which raises security concerns that need addressing and are discussed in section 8.2.2.

In the development version, timing parameters (frequency and duration of recordings) can be changed in the settings, but in a publicly released application, these would need to be set by the developer. Another possibility is to dynamically change it according to certain conditions, this is explained in chapter 13.

6.3 Micro scale: Sampling the recorded snippet

An important aspect of the measurement is the quantification of the recorded sound. This means that the output of the microphone has to be transformed into a set of numeric properties that can later be aggregated and visualized. The process is not straightforward, so below I explain the necessary steps.

6.3.1 Converting the microphone's reading into dB_{SPL}

The general task is to convert the value reported by the microphone into an absolute and comparable number. On one end, all we can get from the system is a unitless number corresponding to the actual voltage within the microphone's range. This value and range is varying between models and brands and possibly even within the same model. This problem is further discussed in section 8.2.1. The other end (the target unit) is the commonly

used dB_{SPL}, that is, sound pressure level, relative to the minimum audible level humans can hear, which is commonly set to 20 µPa [14].

The formula to convert from pressure to dB_{SPL} is shown in Equation 6.1, where L_p is the sound pressure level, p is the pressure level and p_0 is the reference pressure (20 µPa).

$$L_p = 20 \log_{10}\left(\frac{p}{p_0}\right) \text{dB} \tag{6.1}$$

Relying on an experiment conducted by a StackOverflow user [17], I assumed that the maximum voltage in the microphone corresponds to 90 dB_{SPL} or 0.6325 Pa. The microphone returns values as Java `short`s, which range from -32768 to 32767 which I normalize by taking the absolute value. This gives us Equation 6.2, where s is the resulting SPL value and m is the value read from the microphone.

$$s = 20 \log_{10}\left(\frac{0.00002 + \frac{0.6325 - 0.00002}{32767} m}{0.00002}\right) \text{dB} \tag{6.2}$$

For a rough mapping between common noise sources and their respective loudness, see Table 2.1.

6.3.2 Measuring the sound's properties

There are two properties of the recorded sound that the application quantifies and records: average and maximum. The first gives us a good approximation over the general noisiness in the area, while the second could be useful to detect impulsive noise, e.g. car horns, sirens.

I obtain these values by simply iterating over the values in the file (which correspond to the amplitude, and thus the pressure levels), and calculating the average and maximum numbers. These are then converted to dB_{SPL} using the method described above.

Chapter 7

Architecture and implementation

Implementation was an important part of the project. Before almost any measurements could take place, a prototype Android application and some sort of central server had to be designed and implemented. In this chapter I explain the technologies I used and the choices I had to make, starting with a high-level architecture, then elaborating on the smaller-scope architectures of the mobile application and the webserver. In chapter 8, I also explain the challenges I encountered during design and implementation.

7.1 High-level architecture

From a high-level point of view, the architecture is simple and is shown in Figure 7.1: There is a webserver with a known and fix address, and several Android-based devices running the app that takes the measurements. Users can also open a page in a browser, to analyze and filter the data.

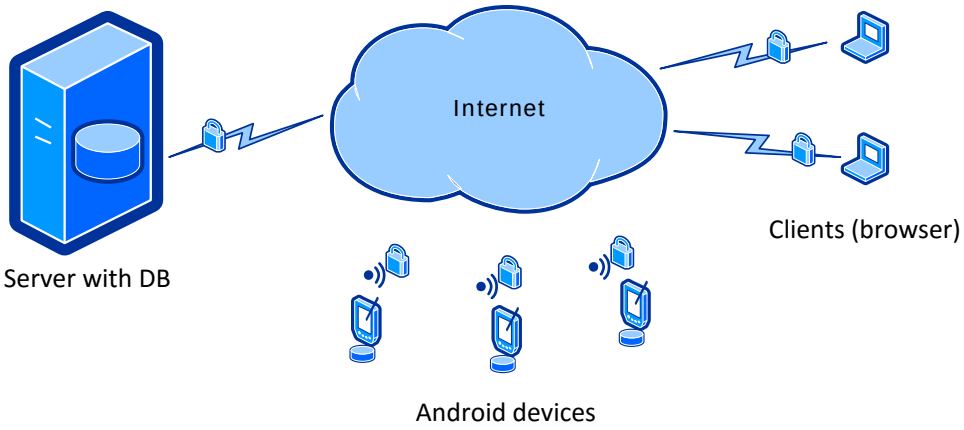


Figure 7.1. High level architecture diagram

The app wakes up at certain (configurable) intervals, records a sound snippet, processes it, saves the result locally, then goes back to sleep. Once

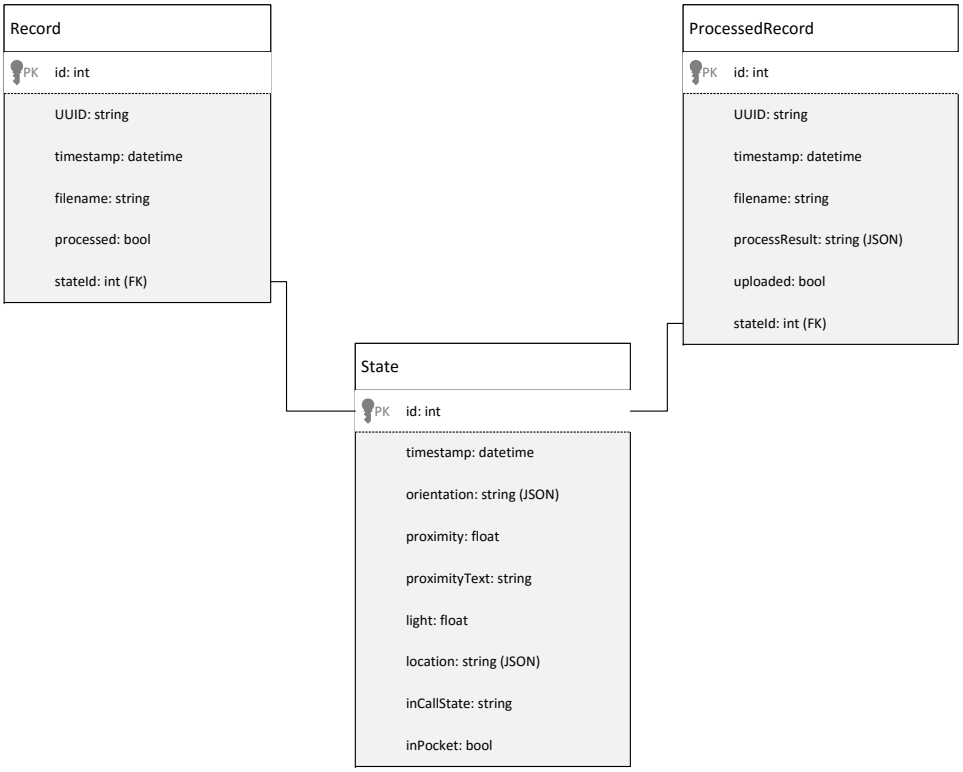


Figure 7.2. Data model of the Android application

the circumstances allow it – in the prototype: the user presses a button – the results are uploaded to the server. The connection between the device and the server is secure. On the server, the results are saved in a master database, and, after login, made available for browsing and visualizing.

7.2 Android application

The Android application consists of three components: **1)** data model and data access layer, **2)** background services, and **3)** user interface. In the following sections, I describe the architecture of each component and discuss their implementation.

7.2.1 Data model and data access layer

For each sound snippet the application records, there are various meta data that needs to be stored as well.

Architecture

There are three main classes, one for storing the state of the device, and two for data describing a recording. Figure 7.2 shows an overview of this data model.

The *State* class contains data describing the physical state of the device at a given point in time. It stores 3D orientation, GPS location, whether there is an ongoing call or the phone is ringing, readings from the proximity and light sensors, and a guess whether the device was in a pocket (see section 8.1).

The *Record* class represents a raw recording, storing when exactly the recording took place, which file the sound is stored in, a reference to the state of the device saved in connection with the recording, and whether the sound data has already been processed. The *ProcessedRecord* class is very similar to Record, but it additionally stores the results from processing the noise data and whether this data has been uploaded to the server.

Implementation

The natively supported database technology on Android is SQLite, which is "a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine" ([28]). This was a reasonable choice, considering that it is the default on Android, I have some previous experience using it, and the ORM I ended up using is also based on SQLite.

For the data access layer (DAL) I choose GreenDAO[1], which is an open-source ORM[2] system built for Android and works with SQLite databases. The reason for choosing GreenDAO is previous good experience. While there are other ORMs for Android, evaluating them is not within the scope of this project, thus I picked the one that was known to be good enough. Shortly after I started developing the application, a newer version of GreenDAO was published, which made development even easier by abandoning the previously necessary generator project, and relying solely on model objects and annotations.

It is worth mentioning that towards the end of the project, a new, interesting library was introduced by the same developers as GreenDAO, called ObjectBox[3]. It is a NoSQL-based object-oriented database, which might be worth exploring for future versions of the app.

7.2.2 Background services

Most of the work is done by the background services: they record the sound snippet, process and upload it, while also collecting the device status such as orientation, geographical coordinates, etc. To facilitate incremental development and future maintenance and improvements, all these tasks are handled by separate services.

The original idea was to implement a pipeline-like architecture, where the input is a request for measurement and the output is the characteristics of the snippet and the status of the device, saved to the database. While strictly speaking the final architecture is not a pipeline, it is similar to it in terms of modularity and that the procedure starts with a request, then each

[1] http://greenrobot.org/greendao/
[2] Object-relational mapping
[3] http://greenrobot.org/objectbox/

service passes on its own product to the next, and the final result is a new record in the database.

ListenerService and Recorder

The first logical step is to initiate the recording of a snippet. This is happening in the *ListenerService* background service, that when enabled, runs continuously in the background, and coordinates the noise recording and device status collecting steps. These are triggered by a recurring executable, that does the following (see also Figure 7.3).

1) Creates a UUID that is used to uniquely identify a recording throughout the process (and possibly further).

2) Starts a thread that actually records the audio to a file, while notifying the rest of the application about the process (*Recorder*).

3) Once the Recorder signals that it has started recording without errors, the ListenerService issues a request to the *PhoneStateService* (see later) to collect the status of the device.

4) When the Recorder signals that it has finished recording, and the sound file is available, the ListenerService makes a note of it.

5) When the PhoneStateService also signals that it has collected all required data, the ListenerService makes a note of it.

6) Once both the recorded file and the device status is ready, they are saved to the database, and the procedure can continue with the next step in ProcessorService.

Technically, the recurrence is achieved by repeatedly posting the same Runnable to a Handler within the ListenerService[4]. This is started once the service receives an intent signaling that recording is enabled, and stops once it gets disabled. The actual recording takes place on a separate thread, requests access to the microphone, starts recording, writes the recorded audio to a file in the internal storage (that is, normally unavailable to other applications), then stops after a given time and signals that the file is ready.

The actual recording is implemented with the help of the built-in *AudioRecord*[5] class, which enables recording the raw microphone input into a file. Output format is specified when instantiating the AudioRecord, and is set to be signed 16 bit PCM[6], sampled at 44.1 kHz, recorded in one channel (mono). Stopping the recording happens with the help of a Handler and a delayed runnable posted to it, which changes a flag and causes the recording loop to end.

[4]Canonically, this should be solved by using AlarmManager, but I encountered various problems with that, see section 7.3.1

[5]android.media.AudioRecord

[6]Pulse-code modulation

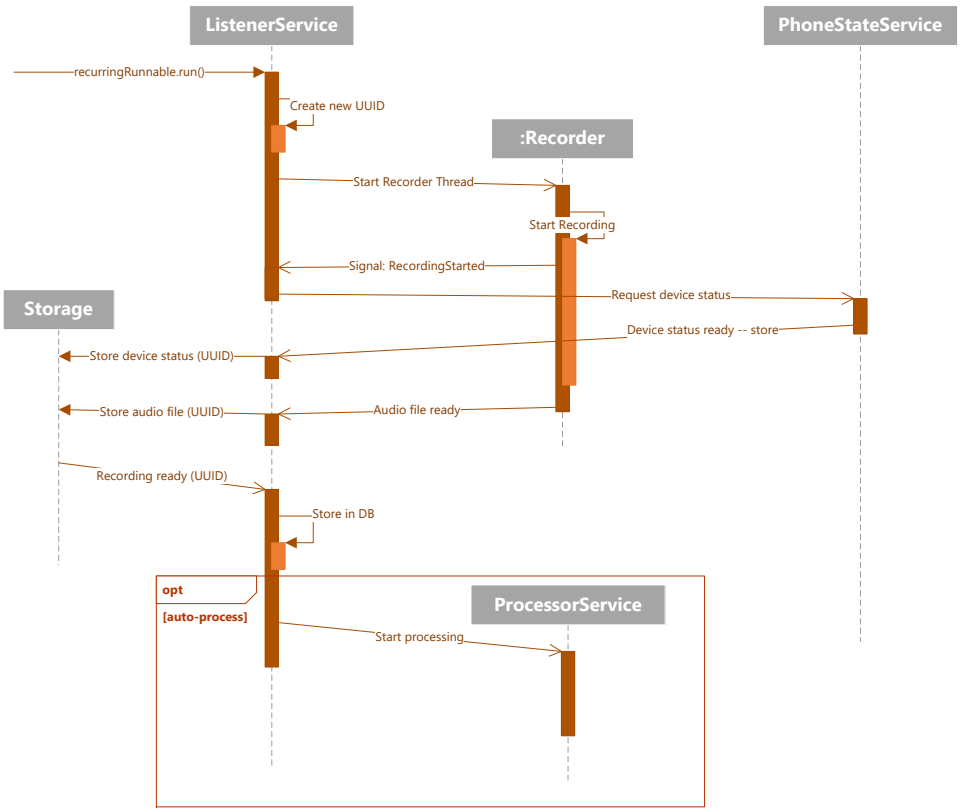


Figure 7.3. Sequence diagram for recording a snippet

ProcessorService

The *ProcessorService* takes an audio file, reads it and calculates some characteristics of it (see section 6.3.2). Optionally, the metadata about the recording, and device specific corrections can be used to give more accurate results. These are then written to the database, and marked for upload.

This service can work on one file, or on multiple files in batch. Ideally, the files can be deleted as soon as their processing is done. In the developer version though, they are not deleted, because the UploaderService needs them to upload to the server for further reference and analysis.

UploaderService

The *UploaderService* takes care of the communication with the central server and of packaging and submitting the collected noise pollution data.

For simplicity, the data is sent to the server in a JSON data structure[7], with all the necessary fields included, using a POST request.

In the developer version, the actual audio files are also sent to the server; they are Base64-encoded and as an extra field in the JSON payload. This raised an interesting problem: Once, I managed to try to upload so many files, that while preparing the payload, the phone ran out of free memory and crashed. To avoid such errors in the future, the to-be-uploaded records are split up into multiple requests if they exceed a certain size limit. To achieve this but avoid falling back to one request per record, the next record is only added to the payload if it doesn't mean exceeding the limit.

PhoneStateService

Everything related to the status of the device is collected and made available by the *PhoneStateService*. After binding to it, clients can request status updates, which are delivered through callbacks once all data is available. When there is at least one request, the service registers for updates of various sensors, accumulates the asynchronously arriving sensor readings, and once all required data is available, packages them and calls the callback(s) with this data bundle.

7.2.3 User interface

Management and administration of the app is possible through a simple user interface. As shown on Figure 7.4, the main screen has three views:

1) *AppStatusView* currently only features a binary switch to enable/disable the recording of sound snippets.

2) *PhoneStatusView*, when enabled, displays the live device status. This means that PhoneStateService is queried frequently and regularly. It

[7] See section 7.5

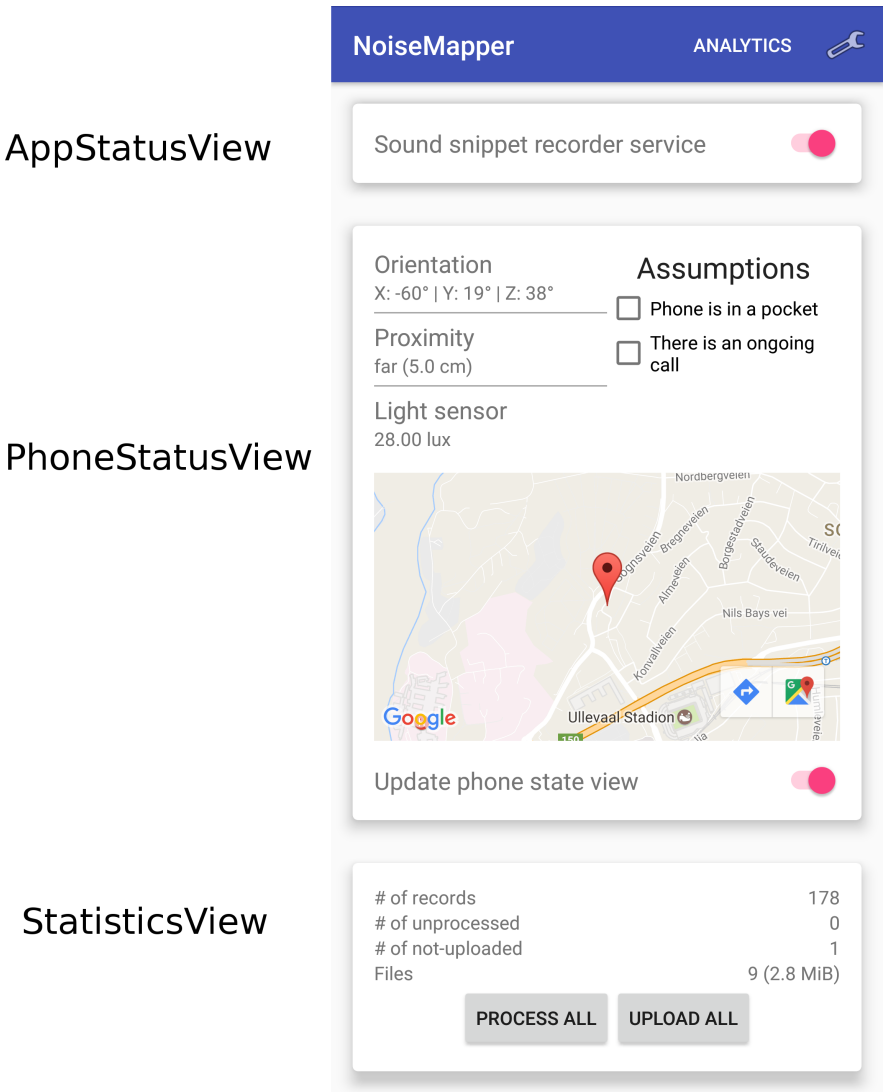


Figure 7.4. Main screen

currently displays 3D orientation, the reading from the proximity and light sensors, whether the device is believed to be in a pocket and whether the phone is ringing or there is an ongoing call. There is also an embedded map which shows the device's current geographical position.

3) *StatisticsView* displays the number of records, if there are any that is not processed or uploaded yet, as well as the number of audio files and the disk space they take up. These numbers are polled regularly when the screen is active.

There is also an *Analytics* screen which is very much intended for developers. Here it is possible to browse the records in the local database and display them on a map.

7.3 Technical decisions

During the development of the application, I had to decide in several technical questions. In this chapter I present these problems together with the reasoning and the decisions I made.

7.3.1 Scheduling the recurring measurements

An important feature of the application is to repeatedly perform measurements. For this the Android platform provides more than one solution, and in the following I briefly describe them, then explain why I decided to use one in the final product.

Scheduling using the `AlarmManager`

When it comes to recurring tasks, usually the first recommendation is to use the `android.app.AlarmManager.setRepeating()`[8] method. It is capable of firing off a certain action at specified intervals, optionally waking up the device if it is in power save mode, or waiting for the next wake up. The desired action has to be encapsulated in a `PendingIntent`, which is delivered to the target Component (a Service in most cases).

There is however an important detail, regarding the recurrence interval: as of Android 5.1 (API22), the interval will be forced to be at least 60 s. The reason for this according to the developers is the high power consumption of waking up the device, as explained in the relevant issue tracking page[9]. The problem with this behavior – other than reduced flexibility of course – is that it is not explicitly documented anywhere[10]. The only way to learn about it is to experience it (i.e. recurring alarms set for short intervals not happening more often than 60 seconds), and a small linting warning in Android Studio.

The suggested approach for shorter intervals is hinted in the extended lint message, and is explained below.

Scheduling using a `Handler`

Another way to make things happen at a certain time is encapsulating code in a `Runnable` and posting it to a `Handler`[11] delayed by the desired amount of time. In addition, to achieve recurrence, the encapsulated code needs to make sure that the same runnable is re-posted to the handler once it is run. Snippet 7.1 is an example for this behavior.

[8] https://developer.android.com/reference/android/app/AlarmManager.html#setRepeating(int, long, long, android.app.PendingIntent)

[9] https://code.google.com/p/android/issues/detail?id=161244

[10] as of spring 2017

[11] android.os.Handler

Snippet 7.1. Example for recurrence with `Handler`

```
Handler handler = new Handler();

Runnable repeatThis = new Runnable() {
  @Override
  public void run() {
    System.out.println("5 seconds later");
    handler.postDelayed(repeatThis, 5 * 1000);
  }
};

handler.postDelayed(repeatThis, 5 * 1000);
```

Scheduling using a `TimerTask`

A third option is to start a `Timer`[12], and schedule a `TimerTask`[13] to it. There are some disadvantages to this approach compared to the `Handler` pattern, namely **1)** one has to extend `TimerTask` instead of simply implementing `Runnable` **2)** a `TimerTask` creates its own background thread, which takes more resources and might reduce functionality (e.g. updating the UI), while a `Handler` attaches to the current thread, and runs code there.

Final implementation

Because of the limitation of the `AlarmManager`, and the disadvantages of the `Timer` approaches above, I decided to use a `Handler` to have recurring measurements. This enables an unlimited measurement frequency while no significant impact on the battery life was experienced.

In addition, I use the same technique to have the recording run for a given time, by starting an "infinite" loop repeatedly reading from the microphone's buffer, and having a `Runnable` posted to a `Handler` set a flag which causes the loop to exit, thus finishing the recording.

7.4 Devices

During the development of the Android application, and for taking measurements, I used two devices, with properties shown in the following table.

[12] java.util.Timer
[13] java.util.TimerTask

	Nexus 4	Nexus 6
Manufacturer	LG Electronics	Motorola Mobility
Android version	6.0.1 (CyanogenMod 13)	7.0
Screen size	4.7"	5.96"
CPU	1.5 GHz quad-core	2.7 GHz quad-core
RAM	2 GB	3 GB

The OS running on the Nexus 4 was rooted, which meant direct access to the database and other files used by the app. This helped the development as it was possible to examine the database in-place, without tedious exporting/copying and the use of a computer.

In addition, during some of the experiments presented in chapter 11, I used a portable ST-8850 Sound Level Meter device. It is capable of measuring frequencies between 31.5 Hz and 8 kHz, and sound levels from 30 to 130 dB. On the display, either the current value (which can be "frozen"), or the maximum value – encountered while the function is active – is shown.

7.5 Webserver

In order to complement the instances of the app running on mobile devices, there is a need for a webserver. The requirements are simple and clear: It has to receive data uploads from the app, store them permanently in a database, and provide a user interface to visualize the data. An important addition to these is that the communication has to be secure, since potentially sensitive user data will be transferred between the app and the webserver. This is solved by directing all traffic through the prevalent HTTPS protocol.

7.5.1 Framework and technology stack

My language and framework of choice was Django 1.10 running on top of Python 3.5. The reason I settled for this technology stack was similar to why I chose GreenDAO as the ORM on the client side: I had several years of experience with Django, and it provided a ready-to-use solution with little extra work necessary. Since the webserver is not the main topic of this thesis, I did not explore other choices.

The Django application is powered by the uWSGI implementation of the Web Server Gateway Interface which in turn communicates with the outside word through an *Nginx* reverse proxy. Nginx handles everything related to SSL and HTTP, while uWSGI is responsible for running the Django app in multiple processes to ensure smooth service. Figure 7.5 visualizes this

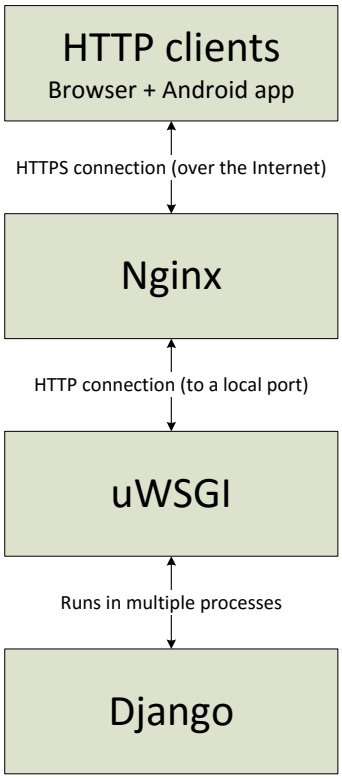


Figure 7.5. Architecture of the webserver

setup, which is frequently used in production environments, and is heavily inspired by [27].

7.5.2 Data model and data access layer

In the webserver, the data model is simple: There is only one class, and the whole payload is stored as text in one field. This ensures quick development, since changes in the Android app need not be reflected in the service. However, as the project and the web UI evolved, it was desirable to extract some data into dedicated fields, to be able to run optimized SQL queries. Such data are: device name, source microphone, timestamp, coordinates, upload timestamp and the measured values.

7.5.3 API endpoints

Both the Android and the browser clients communicate with the server using a simple API. Currently the mobile devices only use it for uploading measurement data, while the browser UI only retrieves data based on various filtering options.

The following APIs are implemented:

- **Upload recording (single):** Expects a single recording in the format described below, converts it to a Django model instance, and saves it in the database. Can only be used with POST requests.

- **Upload recording (batch):** Expects an array of recordings in the format described below. Converts all of them to Django model instances, and saves them in the database. Can only be used with POST requests. Returns a list of recording UUIDs that were successfully saved.
- **Fetch data (reported values):** Expects various filter options, loads matching measurement records from the database, then clusters them using the logic discussed in section 9.3.2. For display purposes, the measured average or maximum values are used.
- **Fetch data (deviation from average):** Same as above, but the displayed numbers are calculated as a deviation from the average value (calculated for distinct categories), as discussed in section 9.2.

Data format

Data transfer between the server and the clients happens in JavaScript Object Notation (JSON) format. The JSON representation of one recording contains the following information: UUID, timestamp, the results of processing the sound file, the complete device status saved in parallel with the recording of the sound snippet, and, in the development version, the actual, Base64 encoded sound file.

Including binary data in the JSON payload is definitely not the most optimal solution, and breaches many security and performance practices, but satisfies the immediate need, and is only used for development purposes.

7.5.4 Authentication

Since the project is collecting highly personal data, and previously it was concluded that contribution is best kept anonymous, there is no user authentication when uploading measurement data. However, the server API still had to be protected from unauthorized access, mainly to avoid malicious pollution of the database with false data. I chose to solve it by requiring a custom HTTP header when accessing API endpoints. This header consists of a name (`X-Noisemapper-Api-Auth`) and a secret value, which is then matched against the server's own copy. The request is only authorized if the two values match. I am aware of the weaknesses[14] of this implementation, but researching and implementing a significantly more secure protocol fell out of the scope of the thesis.

Authentication for the browser UI is handled by the default Django implementation, and requires login. After successful login, the user is assigned a session, and can access all features.

[14] e.g. the secret can relatively easily be extracted from the app installer file, then used from any other device

7.5.5 Browser user interface

The user interface in the browser client is a dynamic webpage, consisting of an interactive map, filtering options and a tabular view, with the map being the in the focus, as shown on Figure 7.6.

Interactive map

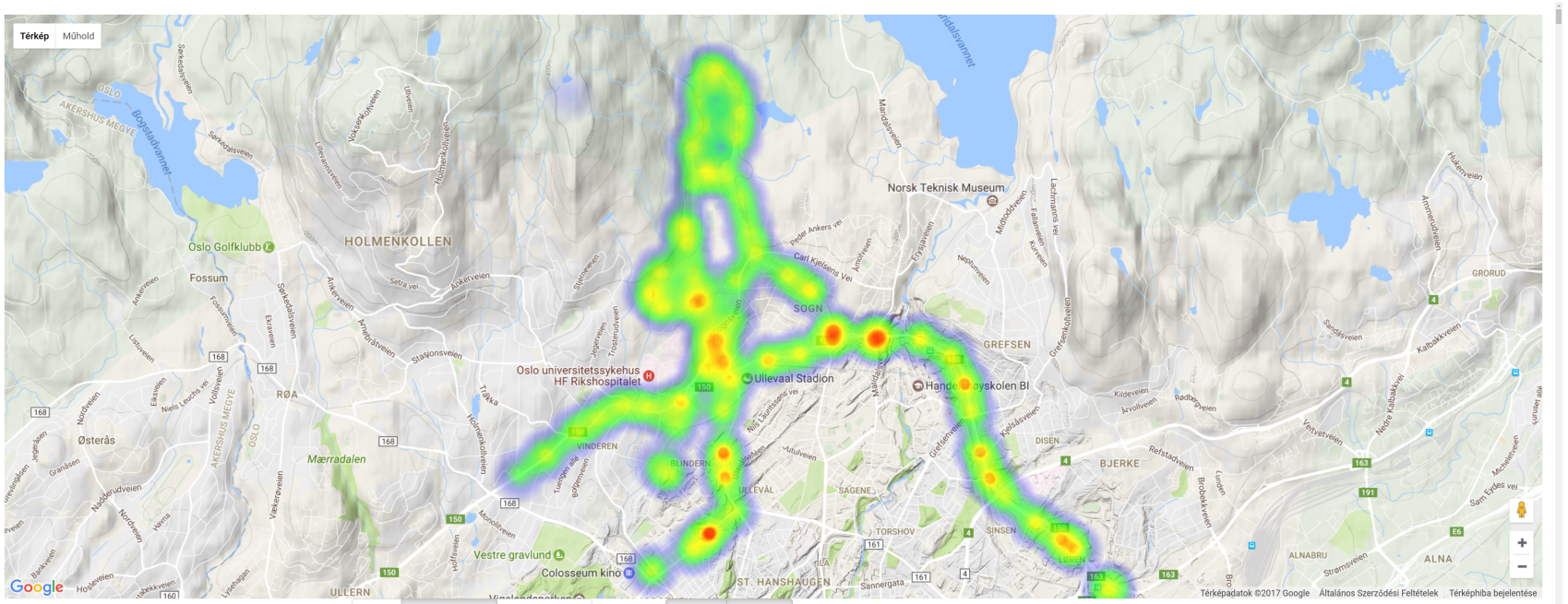
The main component in the UI is an interactive map, using the Google Maps API. The base map layer is the common Google Maps, and there is a heatmap layer rendered on top of that, showing measured sound levels. Clicking on the map tries to find the closest data point and places a marker over it, and opens an infobox with the data point details.

Filters

Below the map, there are various controls to filter what data is fetched from the server and how it is displayed on the map. Most controls automatically trigger a re-fetch of data upon changing them.

Tabular view

The data fetched from the server is also shown in a table, to facilitate finding the interesting/outstanding data points. Each row in the table has a button, which places a corresponding marker on the map, to help visualize the exact location.



Device name: None | Ákos Nexus 6 | Ákos Nexus 4 **Mic source:** Internal | Headset Download

Toggle Heatmap **Radius:**

MAX AVG **Enable clustering** **Enable cropping** **Enable auto resolution** **Data type:** Deviation from average

Aggregation function: SUM **Resolution:** 284.62564264649694 **Auto resolution factor:** 0.85 Reload data Clear markers

Show 10 entries Search:

Latitude	Longitude	Value (AVG / MAX)	Weight	Marker	Count	UUIDs
0	0	-8.369120919628402	0.0771094900646172	Add marker	44	302d70be-d74c-4108-9f65-e7c827db4a6b, 28a0c06c-d8d7-407b-83d5-6b7a063af9c4,

Figure 7.6. The Web UI

Chapter 8

Challenges

During the project work leading to this thesis I faced various challenges, some of which were known beforehand, and some that arose during the implementation. They stem from the project's nature of trying to unobtrusively utilize an everyday object for scientific purposes, and also from the characteristics of the chosen platform. In this chapter I elaborate on these challenges, and the solutions I found.

8.1 Device placement, influencing factors

In order to achieve ideal measurement quality, the recording device should be positioned at around head height – since the noise impacts us through our ears; and should not be covered – to avoid muffling the microphone.

However, with the majority of people, this would cause inconveniences[1], and they would simply stop using the app. Instead, people tend to keep their phones in pockets or in a bag, which is negatively affecting the measurements, primarily due to the muffling effect and extra noise from friction[2].

The effect of muffling can vary based on the number and thickness of the layers of clothing that is between the device and the open world. It is hard to measure, or even estimate.

Friction, on the other hand is mostly caused by repetitive movement, such as walking. In my experience, when reviewing recorded samples taken when the phone was in my pocket, there is a well distinguishable, repeating short scratching sound – about every second, corresponding to my steps – on an otherwise quiet recording.

Solution:

During the early phases of the project, I focused on developing the logic to detect the position of the device, including the 3D orientation and geographical coordinates, whether there is an ongoing call, or if the device

[1] According to a feedback mentioned in [8]: "Other annoying experiences were: [...] the need to constantly hold the phone [...]"

[2] Friction of clothe fabric on the phone and/or other fabrics

is possibly in a pocket. These metadata are stored with each recording, and can be used to fine-tune the processing of the recorded sound file.

I also store the number of steps taken during the recording. This can be used to filter out repetitive spikes in the audio. It also enables retaining other regular noise, since by using the number of steps, and knowing the length of the recording, we can calculate the average frequency of the walking. Also, if there are no steps recorded, no filtering should happen.

For both the muffling and the walking induced friction sounds, using a headset is a good workaround. Even more so, as the headset is in an ideal position – closer to the ears – and during my everyday commute, I see a large number of people listening to music from their phones via headsets, so it would not require extra effort from contributors. The measurements do not conflict with playing music, but can heavily benefit from using the headset's microphone over the device's built-in one.

In addition to recording directly available sensor data, there is an extra attribute which is stored in the metadata: whether the device is likely to be in a pocket. This is an experimental calculation, based on input from multiple sensors. When the following three criteria is true, it is assumed that the device is in a pocket (in an upright position, as it often is in my case). The criteria are: **1)** the proximity sensor reports "near" value, **2)** the light sensor reports a value below a certain threshold[3], and **3)** the orientation sensor reports that the device's pitch value is close to the vertical axis (within a certain threshold).

8.2 Technical challenges

Apart from natural or methodology-based challenges, I faced some technical ones too. These stem from the actual devices I use and the components built into them. Below I list these problems and the workarounds I used.

8.2.1 Microphone variability

According to a survey conducted in 2015 [2], there are more than 24.000 different devices running 9 different versions of Android, and the most frequent 90% of the devices are still from 17 different brands/manufacturers[4]. This poses a difficult calibrating problem: Two devices, in the same position measuring at the same time, with identical settings, might report largely different values.

I made some tests, with results shown in Table 8.1 and Table 8.2. Both tables show results of a simultaneous recording with both devices, using sample length: 5 s, recorded in a reasonably quiet room, windows closed. During the first recording, nothing notable happened. To test the measurement of the maximum values, during the second round, I snapped my fingers once, about 10 cm from the devices, inducing a single, distinguishable noise impulse.

[3] Might not always be 0, e.g. on the Nexus 6, it never goes below 2 lux

[4] Source: Own spreadsheet, based on [2]

Table 8.1. Device differences in recording under identical conditions (quiet environment)

	Nexus 4	**Nexus 6**	**ST-8850**
Average noise level (1) (dB_{SPL})	47,5	32,1	58,0
Maximum noise level (1) (dB_{SPL})	73,8	47,5	
Average noise level (2) (dB_{SPL})	47,6	32,5	58,5
Maximum noise level (2) (dB_{SPL})	75,9	48,5	

Table 8.2. Device differences in recording under identical conditions (quiet environment + finger snapping)

	Nexus 4	**Nexus 6**	**ST-8850**
Average noise level (1) (dB_{SPL})	48,3	34,1	58,1
Maximum noise level (1) (dB_{SPL})	89,2	90,0	
Average noise level (2) (dB_{SPL})	49,3	33,4	68,3
Maximum noise level (2) (dB_{SPL})	89,8	88,4	

I also included the readings from a calibrated ST-8850 sound level meter. This device does not support measuring average values, so I relied on manual averaging of the displayed value. It does support however storing and displaying the maximum value measured. The two features (i.e. displaying current and maximum values) are mutually exclusive, so I performed both test scenarios twice, first looking for the average value, then the maximum value (marked with (1) and (2), respectively in the tables).

The sound level meter has two measurement ranges, the lower ranging between 35 and 100 dB_{SPL}[5], and the higher between 65 and 130 dB_{SPL}. While experimenting with the device, I found out that most of the time my room is quieter than the lowest measurable loudness (35 dB_{SPL}), which resulted in a notification on the device. While it still displayed values below that level, I assumed[6] the warning means that the displayed value cannot be trusted. To ensure that the sound level stays within the device's measurement range, I stared playing a white noise from my laptop's speakers, at a constant volume.

[5] Assuming dB_{SPL}, the user manual only mentions dB

[6] No explicit comments on this in the manual

8.2.2 Privacy

Another, important factor to consider before wider deployment, is privacy. The app is recording smaller (5-10 s) audio snippets, and associates geographical coordinates and a timestamp with them. This contains potentially very sensitive private information, which most people would be uncomfortable sharing.

Solution:

One of my first implementation ideas was that due to these privacy concerns, no actual sound recording can ever leave the user's device, and even files stored on the device must be kept secure and should be removed as soon as possible. A direct consequence of this thought was that all processing should happen on the phone, and only the metadata should be uploaded to the server. This presumes that the phone has enough, and powerful enough resources to process the sound data, which is the case with all modern smartphones.

It is important to mention, that in the development version of the application, the recorded sound files *are* uploaded to the server. This is in order to facilitate testing and verification, by being able to actually listen to the recorded tracks, and re-process them if necessary. Since testing was done solely by me, and I am fully aware of the process and possible privacy concerns, this is an acceptable deviation from the original goals and should only be present in the test version, and never deployed to other users.

8.2.3 GPS

An important part of the data this thesis relies on, is the accurate position of the device, when the measurement took place. To be able to collect this data, I rely on the GPS functionality provided by the Android system. A major drawback of this dependency, is that GPS generally doesn't work indoors. There are technologies that might provide indoor positioning (e.g. using Bluetooth LE beacons), but those fall out of scope for this thesis.

8.2.4 Battery

For the project to be widely used, it is important to reduce the battery usage to a low level. While during the testing phase I didn't particularly focus on measuring or actively reducing energy consumption, I occasionally checked the system's built-in battery monitor, and found that the Noisemapper app was always among the smallest consumers (accounting for 2-5% of battery usage).

Chapter 9

Visualization

Collecting data is not of much value without a way to present it to those interested in the research. This is why visualization was part of the project from the start, and a basic implementation was born at the same time as the first measurements started to take shape.

In this chapter I discuss the decisions and challenges I encountered during the development of the visualization component, starting with the various ways and options such data can be displayed on screen, then how data stored in the database should be pre-processed.

9.1 Heatmap

A *heatmap* is a popular tool to visualize data that is closely tied to geographical coordinates within a certain area. Common data sources for heatmaps are temperature (hence the name), real estate price, criminality level and noisiness; but even distances from facilities (e.g. petrol stations) can be visualized on a heatmap.

Heatmaps work by showing an overlay on top of a regular map, and color-coding it according to the values in the dataset. Colors of a heatmap are often from the range red-orange-yellow-green-blue, where typically red is associated with the highest value and blue with the lowest. The origins of this mapping of colors probably comes from the human perception of color temperature: we associate red with hot things (i.e. higher temperatures, like fire) and blue with cold things (i.e. lower temperatures, like ice).

Heatmaps are generated from discrete data points and usually show concentric circles, with the innermost one colored correspondingly with the value of the data point, and then gradually "fading" outwards to "colder" colors. As seen on Figure 9.1, an important property of heatmaps is that these individual points can merge with each other, covering areas where there is no data.

Also, when some individual points are so close to each other that they cannot be displayed individually, they appear as a single point, with a value that equals the combined (summed) values. This is not desired in this project, so the points need to be clustered as explained in the next section.

(a) Separate (b) Merged

Figure 9.1. Data points on a heatmap

9.2 Pre-processing data for visualization

In the web interface I implemented, there are two aspects of the data that can be visualized. One option is to have the absolute values as the basis of the heatmap intensity values. This means that from the measured dB_{SPL} values, the lowest is assigned to the "coldest" color and the highest to the "hottest" color. If we assume that the measurements are accurate, this can serve as input to further research and decision making.

However, if we cannot assume enough accuracy, there is another option: plotting the deviation from a base value. This gives a relative loudness map, and does not require high accuracy. In fact, this is arguably the more suitable visualization method at the current state of the project, since the reported values are highly dependent on the device and microphone types, and an absolute value is difficult to calculate.

When choosing the base value of the deviation calculation, I decided to go with calculating an overall average of the average values, and making this the reference point. The calculation could be performed on a per-device and per microphone source basis[1], thus working around the differences in the devices.

9.3 Clustering data points

In addition to the pre-processing described above, one more step is necessary before the data is ready to be visualized on the heatmap: clustering.

9.3.1 Why is clustering necessary?

When measurements are enabled, the application records at a given interval, regardless of geographical position. This means that it is highly possible that multiple measurements will be recorded at the exact same spot, e.g. while waiting for the bus at the bus stop, or visiting some place frequently. While this is not at all undesirable, since it increases the temporal coverage of a given area, it can cause problems when trying to visualize the data.

[1] effectively an SQL `GROUP BY` statement

As discussed in the previous section, the heatmap turns "hotter" where there is a high value, *or* if there are many smaller values in a very small area, since they get accumulated. In our case, this would result in such spots heating up, not because they are particularly loud, but because there are many measurements at the same place.

9.3.2 Clustering algorithm

As a solution to this issue, I implemented a clustering algorithm, shown in Snippet 9.1, which merges measurements that took place within a configurable distance from each other into a single pseudo-measurement. The value of this new measurement is the average of the values merged together. The algorithm expects an iterable of data, a function that creates a key for a data point, another function that decides if two keys should be in the same cluster, and an Aggregator instance, that controls the overall value of one cluster.

Example usage of the algorithm is shown in Snippet 9.2, where the data to be clustered is a list of Recording objects from the database, the key over them is their coordinates, and they are put in the same cluster if their distance is less than the value stored in `resolution`. Data points in one cluster are aggregated so that the cluster's value is the average of the individual values.

Snippet 9.3 shows a different approach to aggregating values in a cluster. Here the values are not simply averaged, but a more complex calculation is used:

1) The simple average of the coordinates is calculated, which gives a midpoint[2], then

2) each point is given a weight, decreasing with the distance from the midpoint, then

3) a weighted average of the values is calculated, and finally

4) the midpoint (as the new location) and the average value is returned.

This results in a more even distribution of data points on the map, in contrast with the simple averaging aggregation, where the first point in a cluster determined the location of the cluster.

[2]The calculation doesn't take the Earth's roundness into account, which is acceptable for the small areas it is used for.

Snippet 9.1. Clustering algorithm

```
def cluster_data(data: Iterable[T], key_func: Callable[[T], Any], is_same_func: Callable[[T, T], bool], aggregator_factory: Callable[[], Aggregator], retain_original=False):
  clustered = {}

  for datapoint in data:
    key = key_func(datapoint)
    for existing_key in clustered.keys():
      if is_same_func(existing_key, key):
        key = existing_key
        break
    clustered.setdefault(key, []).append(datapoint)

  clustered_2 = dict()
  for key, values in clustered.items():
    aggregator = aggregator_factory()  # Create a new one for each cluster
    for x in values:
      aggregator(x)

    new_key, new_value = aggregator.get()
    if not new_key:
      # For simple aggregators that don't modify the key
      new_key = key
    if retain_original:
      clustered_2[new_key] = dict(original=values, aggregated_value=new_value)
    else:
      clustered_2[new_key] = new_value

  return clustered_2
```

Snippet 9.2. Example usage of clustering algorithm

```
clustered = cluster_data(
  Recording.objects.filter(**filter_criteria),
  key_func=(lambda r: (r.lat, r.lon)),
  is_same_func=(lambda a, b: distance(a, b) < resolution),
  aggregator_factory=(lambda: Averager(extractor=(lambda r: r.measurement_avg))),
  retain_original=True,
)
```

Snippet 9.3. Algorithm for calculating the geographically averaged, then distance-weighted value for a cluster

```
class GeoWeightedMiddle(Aggregator):

  def __init__(self, extractor):
    self.extractor = extractor
    self.locations = []  # (lat, lon)
    self.values = []

  def __call__(self, new):
    lat, lon, value = self.extractor(new)
    self.locations.append((lat, lon))
    self.values.append(value)
    return self

  def get(self):
    avg_lat = avg_lon = 0
    for loc in self.locations:
      avg_lat += loc[0]
      avg_lon += loc[1]
    avg_lat /= len(self.locations)
    avg_lon /= len(self.locations)
    geo_mid = (avg_lat, avg_lon)

    avg_value = total_weight = 0
    for loc, value in zip(self.locations, self.values):
      dist = distance(geo_mid, loc)
      if dist == 0:
        dist = 1
      weight = 1 / dist
      avg_value += value * weight
      total_weight += weight
    avg_value /= total_weight
    return geo_mid, avg_value
```

Chapter 10

Comparison with related work

In chapter 4 I presented previous works that are related to this topic, and starting with chapter 5 I explained my project. In this chapter I present the similarities and the differences between the approaches.

The research which has the most similarity to my project is that of D'Hondt et al. [8]. It has been introduced in more detail before, but a brief summary might make it easier to compare: They used carefully calibrated mobile phones assigned to a small number of volunteers, who waked around in a small area on pre-defined paths and in short time intervals for about a week. Then the data was collected and visualized on heatmaps. This is similar to my approach, but there are some noteworthy differences.

10.1 Selection of participants

D'Hondt et al. teamed up with a local group of people already interested in environmental issues, and asked altogether 18 people to perform the measurements. They had multiple information sessions, and they were given explicit instructions and manuals to help with the measurements. In short, participants formed a well-controlled group.

In contrast, my research targets the general public, where any slightly interested individual can join the program, install the app and start contributing with little or no further education required. This allows for much wider reach and can result in an increased number of contributions.

10.2 Temporal and spatial coverage

In the referenced experiment, measurements were performed within a relatively small district of Antwerp. In the first part, participants had to cover a well-defined path, and in specific time intervals, for five days. In the second part, constraints were loosened and they were allowed to choose their own paths within the district. They were also free to choose when to measure, provided that they measured for at least one hour each day.

My research is fundamentally different in the temporal aspect: since the measurements are completely automatic, they cover the whole day.

Spatially, it is similar to the second part of the other experiment, in a sense that participants perform their daily routines, and walk their own ways.

10.3 Choice of device

There is a significant difference in the devices used in the two researches. D'Hondt et al. used 10 identical devices, which they carefully calibrated prior to giving them to participants. This requires significant preparation from the researchers but leads to more accurate results.

One of the basic ideas in my thesis however, is to leverage crowdsourcing techniques. This means that the participating devices will be very heterogeneous and that pre-calibration is not possible. To compensate for this loss of accuracy, two things can help: First, thanks to the larger number of participants, much more data can be gathered, which leads to statistically smoother numbers (i.e. the erroneous values are averaged out). Second, instead of exact dB values, a more tolerant calculation is used: deviation from average. This is calculated on a per-device basis, where the recording quality is constant.

10.4 Aggregation of data

Measurement data has to be aggregated, or clustered before it can be visualized on a heatmap. In this, there is no fundamental difference between the two projects, except two minor ones: **1)** D'Hondt et al. use a square grid for creating clusters, while I use simple distances, which result in circular cells; and **2)** they removed cells which contained less data points than a certain number, while I allow even single points to appear on the map. This second difference might introduce a slight error, but if that is not acceptable, it is easy to adapt the clustering logic to drop such cells.

Chapter 11

Experiments

As soon as the application was ready to take sound measurements, I started experimenting with various settings and setups. These experiments provided useful feedback for further developing the application. In this chapter, I present four experiments, their results, and the effect they had on the application:

1) Which source microphone yields the best results?

2) Do phone measurements interfere with phone calls?

3) Can an external microphone improve quality?

4) How do environmental effects affect measurements?

Most of the longer experiments took place while I went on a walk, starting from close to a relatively busy, thus noisy road, walking through a quiet, almost rural, thus quiet and peaceful area and arriving back to the same busy road.

11.1 Source microphone

The Android system exposes seven types of microphone inputs. Some of them are only available to system applications (due to privacy considerations, i.e. voice call channels), which leaves application developers four options (not counting the default one, which is an alias):

1) `MIC` is the main device microphone,

2) `CAMCORDER` is the one suitable for recording video, as it has the same orientation as the main camera (if available),

3) `VOICE_RECOGNITION` is "tuned for voice recognition"[1], and according to some non-official sources[2], this has the least pre-processing (such as noise cancellation, etc.)

[1] from the documentation

[2] https://groups.google.com/forum/#!topic/android-platform/2ZJFBjJOZV0 and https://forum.xda-developers.com/showpost.php?p=13163611&postcount=6

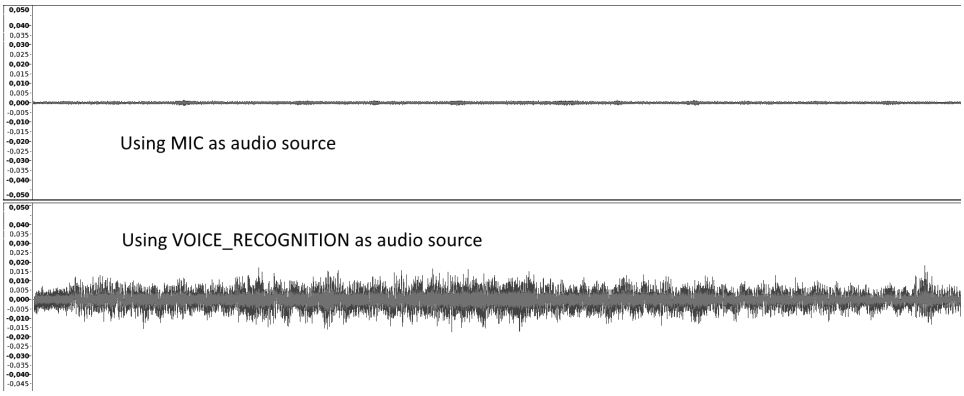


Figure 11.1. Noisiness of `MIC` and `VOICE_RECOGNITION` inputs

4) `VOICE_COMMUNICATION` is documented to be taking advantage of echo cancellation, automatic gain control (if available).

Of the above options, `VOICE_COMMUNICATION` is not suitable, since we need the original sound input. `CAMCORDER`, by experience is the same as `MIC` or `DEFAULT` on my devices. Comparing recorded sound files using `VOICE_RECOGNITION` and `MIC` as audio sources shows that the latter most likely has some sort of processing (noise cancellation), as it resulted in clearer audio (see Figure 11.1). However, when comparing against the ST-8850 Sound Level Meter, the unprocessed `VOICE_RECOGNITION` values turned out to be much closer to the values measured by the calibrated device. Thus, I decided to use this as audio input.

11.2 Measurements and phone calls

An important question for the usability of the application is its behavior around and during phone calls. In the documentation, I couldn't find explicit mentioning of this scenario, and on various forums I found different behavior (though most were about much older versions of Android). To have a definite answer – at least for the devices I used in the tests – I conducted the following experiment.

I listed four scenarios regarding the order of events (starting and stopping a recording, and answering and hanging up a call, abbreviated as **B**egin, **E**nd, **A**nswer, **H**ang up, respectively), then performed each.

1) **B-A-H-E:** The recoding starts as expected, then there is some static noise before the phone starts ringing. The ringtone and potential vibration is registered very loudly on the recording, then as soon as the call is answered, the recording continues, but the recorded sound becomes very muffled. It does not get restored after the call is hung up. See Figure 11.2. The phone call's audio is not affected by the ongoing recording.

2) **B-A-E-H:** Same as above.

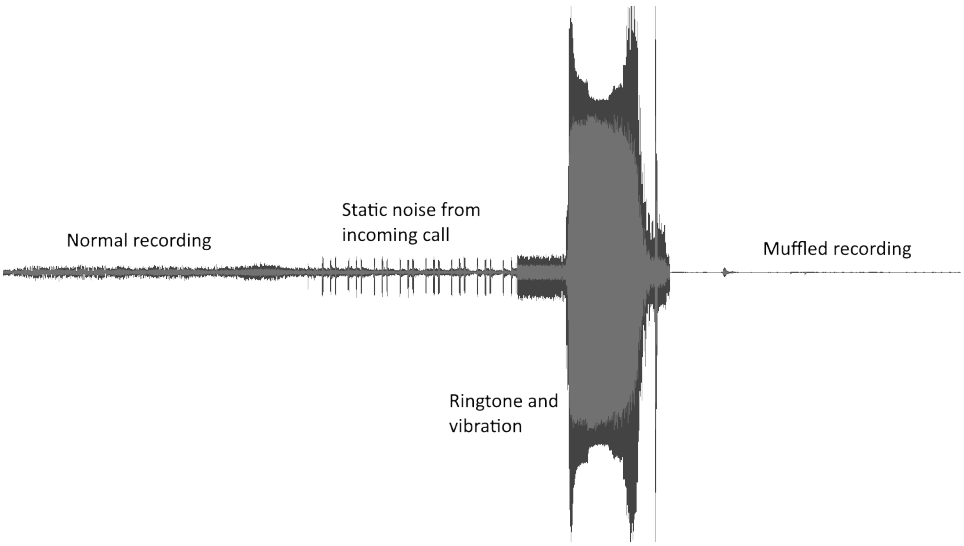


Figure 11.2. Waveform of recording with incoming call

3) **A-B-E-H:** All external noises clearly audible on the recording, but it is muffled like in all cases when there is an ongoing call during the recording.

4) **A-B-H-E:** Same as above, with the addition that the muffling does not disappear with the ending of the call.

The overall result is that a phone call does not interfere with the possibility to record, nor does an ongoing or starting recording prevent or affect phone calls. However, calls do affect the characteristics of the recorded sound. Thus, it should be avoided that the app records at the same time the user is making a phone call. In other words, recordings should not be started while in a call, and ongoing recordings need to be stopped and discarded in case of an incoming call.

11.3 Internal vs. external microphone

Motivated by the challenges about using recording with the internal microphone while it is in a pocket (described in section 8.1), I looked into using a wired headset. First I made sure that the app works with a headset plugged in, and that it uses the external microphone instead of the built-in one. Luckily, no change was required in the code, and everything worked as expected. Then I repeated one of my earlier experimental walks, but this time with the headset plugged in.

From the experiment, I expected to get a much cleaner sound, due to eliminating most of the frictional noise caused by my coat grazing against my pocket, resulting in lower average values.

When plotting the processed results, and listening to the samples recorded during the walk, my theory was confirmed: apart from occasional

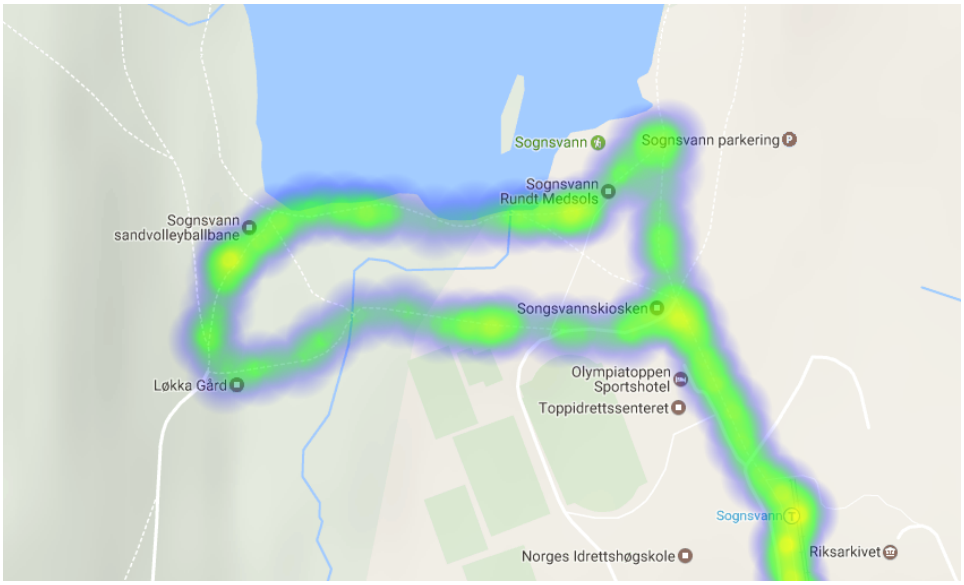


Figure 11.3. Effects of surface type on noisiness

sounds of fiddling with the headset, there was almost no frictional noise, so the numbers clearly represented the environmental noise.

11.4 Environmental effects

Most of the measurements I took happened during winter time. Thanks to the relatively snowless but cold early winter this year in Oslo, the road maintainers put a big amount of small stones (gravel) on the roads and pavements, to make them less slippery. This might be very useful, but also makes walking very noisy, as these small stones grind under one's feet.

Even without measurements, it makes being outside audibly noisier, but I also did some experiments, where I walked in an otherwise quiet environment, both on gravel and flat ice or clean asphalt. The results confirm my theory, walking on gravel does have an impact on the average noise levels measured. A good example is shown on Figure 11.3: The main path leading from Sognsvann T-Bane stop down to the lake was well covered with gravel, similar to the area where the "Sognsvann Rund Medsols" label is. On the contrary, the area to the west, and especially the path between Løkka Gård and Sognsvannskiosken was mostly covered with thick smooth ice, and resulted in an audibly quieter walking experience.

Another surface type that negatively affects the noise levels is re-frozen slush. Since the water freezes in thin, easy to break formations, every step makes a loud crunching sound, which is clearly audible on the recordings, and traceable in the processed numbers.

Chapter 12

Conclusion

In the previous chapters I presented the motivation for this thesis, elaborated on the two main pillars – noise pollution and crowdsourcing, summarized related researches on the topic, then introduced my project, the implementation and the challenges I faced. Later I discussed the methods used in the project, compared them with previous works and conducted some experiments. In this concluding chapter I summarize the challenges I met, together with the solutions I found, then present my results. Finally, in the last chapter I look at possible topics for continuing the research.

12.1 Obstacles and solutions

The development of a new mobile application and the website raised several challenges, both expected and unexpected. In the thesis, I present the problems and discuss various solutions then argue about the ones I chose to implement.

Probably the most challenging obstacle was the friction noise caused by clothes, especially when walking. A possible workaround for this is to use an external microphone (e.g. a headset), which significantly improves the quality of the recorded samples.

Another challenge was the inaccuracy of the devices' built-in microphones. The only solution to this is to individually calibrate all participating devices, which is infeasible. Instead I decided to use a more reliable metric when visualizing: the deviation from a device-specific average. This does not show the absolute loudness of a place, but it is capable of highlighting the noisier-than-average areas, which is the primary goal of the project.

12.2 Findings of the thesis

The goal of the thesis is to evaluate the possibility of using smartphones for mapping environmental noise levels in frequented areas, such as cities. The viability of such usage of smartphones depends on many things,

including user engagement, microphone variability and accuracy, and other influencing factors.

Measurement accuracy

Arguably the toughest challenge when using general purpose devices for scientific measurements is measurement accuracy. There are many types of smartphones, using many types of microphones, each with different software processing. Taking each and every setup into account and providing a calibration profile for each would be unfeasible.

My solution to the problem is to abandon using absolute loudness values, and instead plot the deviation from a device-specific average. This proved to be working well, as shown in chapter 9. In addition, using an external microphone (i.e. headset) significantly improves measurement quality, and should be preferred.

Visualization

An important part of the project goal is to help stakeholders focus on the area that needs attention the most.

For this, a website has been created which shows the measured data in an aggregated form. Easy, at-a-glance visualization is achieved by displaying a "heatmap" layer of the noisiness, overlaid on a regular map to enable correlating noisiness with actual locations.

Since the original, non-aggregated data is stored in the database, the web application can be easily adapted to enable researchers filter and access anonymized data for further research.

12.3 General conclusion

In the light of the above statements, we can conclude that while crowd-based noise mapping using smartphones is not a trivial task, it is certainly doable and is capable of highlighting the noisiest areas. As mentioned before, the results are not absolute, and might not be suitable to directly base decisions on, but they can help find the areas that need more focus.

The method described in this thesis requires an important precondition to be met, in order for it to be useful. It needs to have enough users attracted to the project to cover the target area with satisfying resolution (both temporal and spatial). This can be achieved by targeted campaigns, advertisements, etc.

Chapter 13

Future work

In this final chapter I look at some possible improvements and development directions that a potential future research could address. These ideas fell out of scope for my project, for various reasons, but could contribute to an even better understanding of the topic.

13.1 Measurements

During development and the experiments, I came up with a few ideas that cold further improve the efficiency of data collection and accuracy of the data collected. The first addresses the processing of a single measurement, improving data accuracy by comparing with a device-specific "quiet" sample, while the second one could help improve coverage and overall efficiency by dynamically adjusting the frequency and length of the measurements.

Comparing with a "quiet" sample

While comparing the recoded samples from my two test devices, I figured out that it could be helpful if the recorded data would be normalized. There would be a special measurement by select users, in a very quiet environment (e.g. out in the forest on a windless day), which could be used as a reference, or baseline, and each subsequent recording could be compared to this one, and the difference would be reported.

This way the device- or microphone-specific differences could be masked out, resulting in more accurate results. Of course, this "quiet" sample should still not be treated as 0 dB_{SPL}, since that is difficult to achieve without special equipment. Its nominal loudness could be determined by asking the user about the environment (e.g. wind, type of surroundings), and providing some sensible default values.

Dynamic measurement parameters

The prototype application allows the user to manually set and change the frequency and duration of the measurements, and start/stop them at will.

This means that in areas with many users at a given time, many overlapping recordings will be taken, which up to certain number helps even out the erroneous measurements, but can easily become a burden on the server without any additional benefit. In this case measurement frequency should be lowered. Other places might only be visited by one or two users, yielding poor measurement coverage, which could be improved by measuring more often.

In a future version, these parameters could be remote-controlled from the server, based on the number of users in a certain area. The devices could poll the service in regular intervals for configuration parameters, and adjust the settings accordingly. To avoid privacy concerns, the data collected to enable this functionality should be anonymized and stored and used in compliance with regulations.

13.2 User experience

The application delivered with this thesis is a prototype, thus lacking many features and properties which would be necessary if a wider user base is to be addressed.

The current user interface mostly serves diagnostic purposes, offering insight in to the internals of the application. Before a public release, a more informative and easier to use UI should be designed and implemented.

A significant addition to the privacy aspect could be a feature where the users could control periods or areas where the app stops recording. While no actual recording should ever leave the device in a publicly released version, metadata – such as GPS position – could still be a privacy concern to some people. Also, it might be psychologically desirable to be able to specify a zone similar to “do not disturb”.

Based on my personal experience, many people in the targeted area – Oslo – are using iPhones. It might be possible to reach a much higher coverage if an iOS version of the app was developed.

13.3 Analysis of the data

It has already been mentioned in the initial description of the project, that the collected and visualized data could be merged with other kind of measurements from the same area, for example health data, real estate prices, inclination to move, sleep quality. It could be an interesting research to find out if there is any correlation between noise pollution and other indicators.

Bibliography

[1] *A Brief History of Crowdsourcing [Infographic]*. URL: http://www.crowdsourcing.org/editorial/a-brief-history-of-crowdsourcing-infographic/12532 (visited on 02 Mar. 2016).

[2] *Android fragmentation report: There are now 24,093 distinct devices, up 28% from last year | VentureBeat | Mobile | by Paul Sawers*. URL: http://venturebeat.com/2015/08/05/fragmentation-report-there-are-now-24093-distinct-android-devices-up-78-from-last-year/ (visited on 22 Jan. 2017).

[3] Robert Bernhard et al. "An Introduction to Tire/Pavement Noise of Asphalt Pavement." In: (). URL: http://www.asphaltroads.org/assets/_control/content/files/anintroductiontotire-pavementnoiseofasphaltpavement.pdf.

[4] Birgitta Berglund, Thomas Lindvall, and Dietrich H Schwela, eds. *Guidelines for community noise*. 1999.

[5] C. Michael Hogan. "Analysis of highway noise." In: *Water, Air, and Soil Pollution* 2.3 (), pp. 387–392. ISSN: 1573-2932. DOI: 10.1007/BF00159677. URL: http://dx.doi.org/10.1007/BF00159677.

[6] Andrew T. Campbell et al. "The Rise of People-Centric Sensing." In: *IEEE Internet Computing* 12.4 (July 2008), pp. 12–21. ISSN: 1089-7801. DOI: 10.1109/MIC.2008.90. URL: http://dx.doi.org/10.1109/MIC.2008.90 (visited on 23 Mar. 2017).

[7] *Comparitive Examples of Noise Levels | Industrial Noise Control*. URL: http://www.industrialnoisecontrol.com/comparative-noise-examples.htm (visited on 20 Mar. 2017).

[8] Ellie D'Hondt, Matthias Stevens, and An Jacobs. "Participatory noise mapping works! An evaluation of participatory sensing as an alternative to standard techniques for environmental monitoring." In: *Pervasive and Mobile Computing* 9.5 (Oct. 2013), pp. 681–694. ISSN: 15741192. DOI: 10.1016/j.pmcj.2012.09.002. URL: http://linkinghub.elsevier.com/retrieve/pii/S1574119212001137 (visited on 25 Feb. 2017).

[9] Anhai Doan, Raghu Ramakrishnan, and Alon Y. Halevy. "Crowdsourcing Systems on the World-Wide Web." In: *Commun. ACM* 54.4 (Apr. 2011), pp. 86–96. ISSN: 0001-0782. DOI: 10.1145/1924421.1924442. URL: http://doi.acm.org/10.1145/1924421.1924442.

[10] Ministry of Climate and Environment. *Pollution Control Act*. Jan. 5, 2007. URL: https://www.regjeringen.no/en/dokumenter/pollution-control-act/id171893/ (visited on 13 Mar. 2016).

[11] European Parliament. *Directive 2002/49/EC of the European Parliament*. June 25, 2002. URL: http://eur-lex.europa.eu/legal-content/en/ALL/?uri=CELEX:32002L0049.

[12] Gary W Evans, Staffan Hygge, and Monika Bullinger. "Chronic noise and psychological stress." In: *Psychological Science* (1995), pp. 333–338.

[13] *Future Noise Policy. European Commission Green Paper. COM (96) 540 final, 04 November 1996*. 1996. URL: http://aei.pitt.edu/1204/.

[14] Stanley A Gelfand and Harry Levitt. *Hearing: An introduction to psychological and physiological acoustics*. Vol. 4. Marcel Dekker New York, 1998.

[15] Jim Giles. "Internet encyclopaedias go head to head." In: *Nature* 438.7070 (Dec. 15, 2005), pp. 900–901. ISSN: 0028-0836, 1476-4679. DOI: 10.1038/438900a. URL: http://www.nature.com/doifinder/10.1038/438900a (visited on 02 Mar. 2016).

[16] Luis von Ahn. "Massive-scale online collaboration." Apr. 2011. URL: http://www.ted.com/talks/luis_von_ahn_massive_scale_online_collaboration/transcript?language=en.

[17] Lukas Ruge. *What does Android's getMaxAmplitude() function for the MediaRecorder actually give me? - Stack Overflow*. URL: http://stackoverflow.com/a/14870458/1119508 (visited on 21 Mar. 2017).

[18] Nicolas Maisonneuve et al. "Citizen Noise Pollution Monitoring." In: *Proceedings of the 10th Annual International Conference on Digital Government Research: Social Networks: Making Connections Between Citizens, Data and Government*. dg.o '09. Puebla, Mexico: Digital Government Society of North America, 2009, pp. 96–103. ISBN: 978-1-60558-535-2. URL: http://dl.acm.org/citation.cfm?id=1556176.1556198 (visited on 23 Mar. 2017).

[19] *Marillion - Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Marillion (visited on 12 Mar. 2016).

[20] *Mr. Peanut and Antonio Gentile: A trademark that defined a life | National Museum of American History*. URL: http://americanhistory.si.edu/blog/2014/05/mr-peanut-and-antonio-gentile-a-trademark-that-defined-a-life.html (visited on 12 Mar. 2016).

[21] *noise - definition of noise in English | Oxford Dictionaries*. URL: https://en.oxforddictionaries.com/definition/noise (visited on 08 Apr. 2017).

[22] *Noise | Miljøstatus*. URL: http://www.environment.no/Topics/Noise/ (visited on 21 May 2016).

[23] *Reading Programme : Oxford English Dictionary*. URL: http://www.oed.com/page/reading/Reading$0020Programme (visited on 14 Apr. 2017).

[24] Ulf Sandberg. *Tyre/road noise: myths and realities*. VTI 345-2001. Swedish National Road and Transport Research Institute, 2001.

[25] Silvia Santini, Benedikt Ostermaier, and Andrea Vitaletti. "First Experiences Using Wireless Sensor Networks for Noise Pollution Monitoring." In: *Proceedings of the Workshop on Real-world Wireless Sensor Networks*. REALWSN '08. New York, NY, USA: ACM, 2008, pp. 61–65. ISBN: 978-1-60558-123-1. DOI: 10 . 1145 / 1435473 . 1435490. URL: http://doi.acm.org/10.1145/1435473.1435490 (visited on 23 Mar. 2017).

[26] *Scilly naval disaster of 1707*. In: *Wikipedia, the free encyclopedia*. Page Version ID: 708184982. Mar. 4, 2016. URL: https://en.wikipedia.org/w/index.php?title=Scilly_naval_disaster_of_1707&oldid=708184982 (visited on 12 Mar. 2016).

[27] *Setting up Django and your web server with uWSGI and nginx — uWSGI 2.0 documentation*. URL: http://uwsgi-docs.readthedocs.io/en/latest/tutorials/Django_and_nginx.html (visited on 27 Jan. 2017).

[28] *SQLite Home Page*. URL: https://sqlite.org/ (visited on 26 Jan. 2017).

[29] Wired Staff. *The Rise of Crowdsourcing*. June 1, 2006. URL: http://www.wired.com/2006/06/crowds/ (visited on 02 Mar. 2016).

[30] Stephen A. Stansfeld and Mark P. Matheson. "Noise pollution: non-auditory effects on health." In: *British Medical Bulletin* 68.1 (Jan. 12, 2003), pp. 243–257. ISSN: 0007-1420, 1471-8391. DOI: 10.1093/bmb/ldg033. pmid: 14757721. URL: http://bmb.oxfordjournals.org/content/68/1/243 (visited on 10 May 2016).

[31] *Støy – lydforureining*. URL: https://www.regjeringen.no/no/tema/klima-og-miljo/forurensning/innsiktsartikler-forurensning/stoy--lydforureining/id2339859/ (visited on 21 May 2016).

[32] *Støyplaga ikkje nok redusert | Miljøstatus*. URL: http://www.miljostatus.no/nasjonale-mal/4.-forureining/mal-4.6/samla-stoyplage-spi-fra-alle-kartlagde-kjelder-spesielt-spi-for-industri-motorsport-og-skytebaner/stoyplaga-ikkje-nok-redusert/ (visited on 21 May 2016).

[33] L. von Ahn et al. "reCAPTCHA: Human-Based Character Recognition via Web Security Measures." In: *Science* 321.5895 (Sept. 12, 2008), pp. 1465–1468. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1160379. URL: http://www.sciencemag.org/cgi/doi/10.1126/science.1160379 (visited on 15 May 2016).

[34] *WHO | Burden of disease from environmental noise - Quantification of healthy life years lost in Europe*. URL: http://www.who.int/quantifying_ehimpacts/publications/e94888/en/ (visited on 01 Mar. 2016).

[35] WHO European Centre for Environment and Health. *Burden of disease from environmental noise - Quantification of healthy life years lost in Europe*. 2011. URL: http://www.who.int/quantifying_ehimpacts/publications/e94888/en/.

[36] *Wikipedia*. In: *Wikipedia, the free encyclopedia*. Page Version ID: 708810386. Mar. 7, 2016. URL: https://en.wikipedia.org/w/index.php?title=Wikipedia&oldid=708810386 (visited on 12 Mar. 2016).

[37] World Health Organization. *Constitution of the World Health Organization*. CreateSpace Independent Publishing Platform, 2006. ISBN: 978-1-4635-4073-9. URL: https://books.google.no/books?id=O0IBfAEACAAJ.