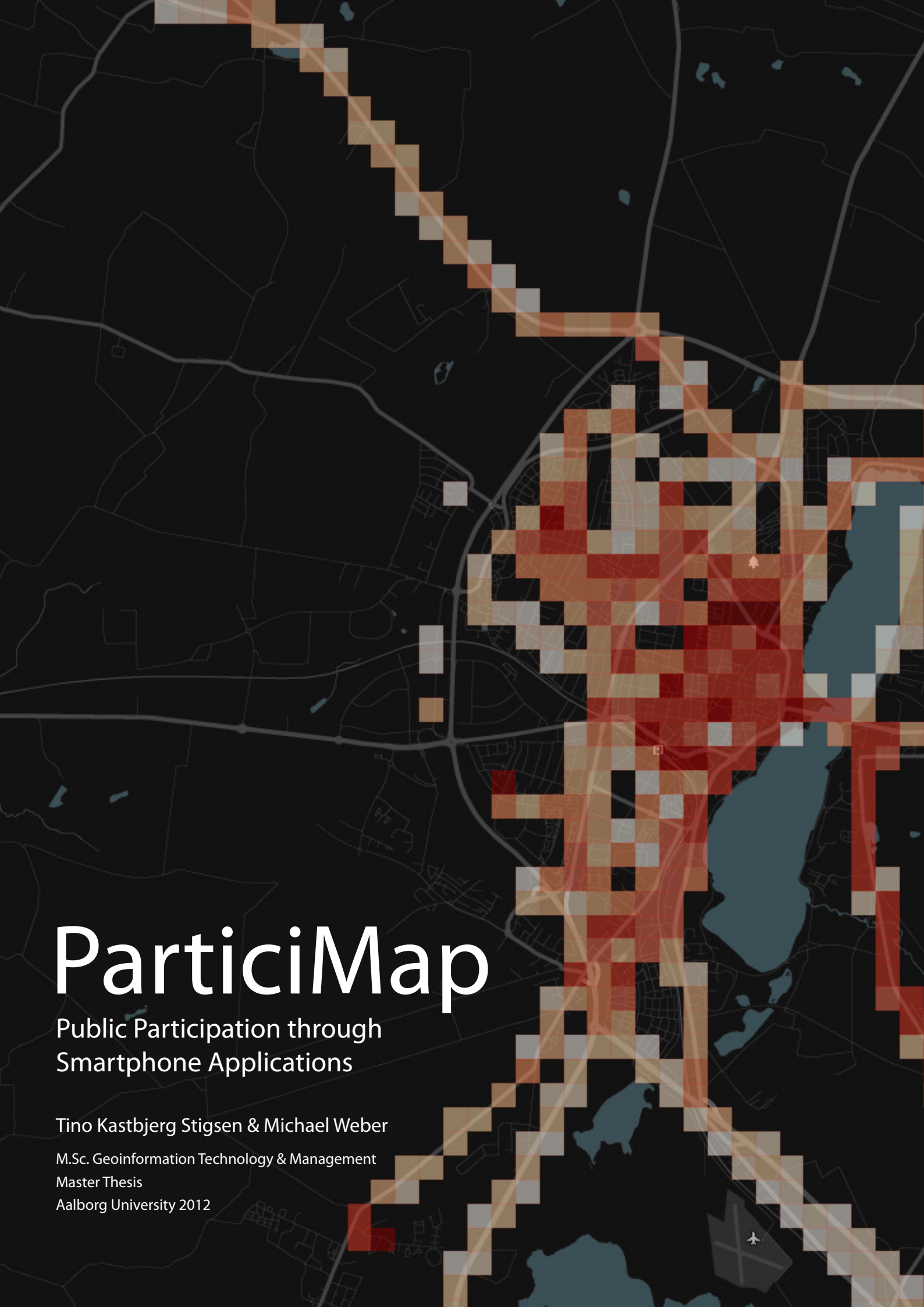


# ParticiMap

## Public Participation through Smartphone Applications

Tino Kastbjerg Stigsen & Michael Weber

M.Sc. Geoinformation Technology & Management
Master Thesis
Aalborg University 2012

**School of Architecture, Design and Planning**
**Aalborg Universitety**
Fibigerstræde 11
9220 Aalborg
Tlf. +45 99 40 83 41
http://sadb.aau.dk

**Titel:** ParticiMap – Public participation through smartphone applications

**Project period:** M.Sc. Geoinformation Technology & Management – Master Thesis

**Project group:** land10gtm_aal01

**Participants:**

_____
Tino Kastbjerg Stigsen

_____
Michael Weber

**Supervisor:**
Henning Sten Hansen

**Abstract:**

People's movements and stays are key elements in understanding how well the physical environment of the cities function and research into this should therefore be of high priority. However, current tools using dedicated GPS devices to obtain information are costly and do not give the respondents in the research projects the means of submitting additional data.

This project revolves around using respondents' own smartphones as a connection to planners in their cities. The project also delivers a web based map platform where collected data from the respondents can be analyzed giving the municipalities a whole new tool in conducting public participation in the planning procedures.

This project documents the process of developing an android based smartphone application which uses the GPS as well as other built-in sensory devices in to allow respondents to submit information about their movement patterns and urban environments that influence their everyday life.

The information submitted is automatically analyzed so it can be directly applied as a planning tool, facilitating smart decisions for the smart cities of tomorrow.

**Copies:** 4
**No. of pages:** 135
**Appendices:** 9
**Report finished:** 7th of June 2012

# Preface

This project is a final thesis in the M.Sc. program in Geoinformation Technology & Management at Aalborg University, School of Architecture, Design and Planning. This report focuses on the development a smartphone application called ParticiMap, for use as a new tool in the planning procedure and in public participation.

In the project a web map platform has been created. The web page is hosted by the University and is accessible at http://viborg.particimap.dk. This webpage shows the collected data from the surveys conducted through the testing of the smartphone application. Collected data is stored in a MySQL database server also hosted by the University. Another product of this project is a smartphone application which can be downloaded and installed from http://app.particimap.dk (NOTE: installing the application requires a smartphone running Android version 2.1 or greater)

We would like to give a special thanks to the 10 students at Viborg Katedralskole who participated in the testing of our application, as well as educational coordinator Allan Andreasen Kortnum who facilitated the contact to the students. Another thank you goes to the research groups “Diverse Urban Spaces”, “Mobility and Tracking Technologies” and “Centre for Mobility and Urban Studies” at Aalborg University which have been a great help in defining the work of this project. We thank Henrik Harder for guidance and the tremendous help with financial aid and the testing of our platform package. Finally we would like to thanks our supervisor Henning Sten Hansen for his work in guiding us through our final thesis on the master program.

# Table of Contents

# List of Figures

# List of Tables

# 1 Introduction

Global Navigation Satellite Systems (GNSS) as a means to conduct positioning was first introduced with the launch of Global Positioning Systems (GPS) by the U.S. Department of Defense in 1973 (Commission on Engineering and Technical Systems 1995). Since then Russia, China and the EU have developed their own systems and many receivers are able to collect data from multiple sources making them very accurate (Gibbons Media & Research n.d.). A broadcasting device has also been attached to the GPS receiver to allow sharing of location data. This has been used for a broad range of applications like fleet tracking, anti-theft solutions, emergency guidance, tree planting systems etc. (KloiMøller n.d.).

GPS tracking is also being used in a wide range of research, from marine animals to traffic (Ryan, et al. 2004) (Hermes Traffic Intelligence n.d.). At Aalborg University, research regarding the tracking of people is done by Diverse Urban Spaces (DUS), at the Department of Architecture, Design & Media Technology. The research group maps and analyzes how people occupy and move through the urban spaces. This is also known as mobility research. The research is done by equipping respondents of a specific focus group with GPS devices which they wear through a set period. The collected data is then analyzed (Diverse Urban Spaces n.d.).

This kind of mapping can be used to create a more knowledgeable debate in the continued planning of the physical surroundings. Another purpose is that it creates a better public participation in the planning because it contributes to a debate amongst the residents of the city in order to clarify what kind of environment they wish to live in, as well as educate the planners as to how the residents view their city (Knudsen, Harder and Simonsen, et al. 2011). The downside is that it is a very costly and time-consuming venture thereby causing most municipalities to overlook its potentials and make traditional public meetings which is mostly inhabited by a very small segment of the population (Præstholm 2012). The idea is therefore to use mobile phones as an easier and less time consuming option (Knudsen, Harder and Simonsen, et al. 2012).

The members of the project group has since early 2011 been employed by the research group DUS, with the responsibility for managing the technical setup as well as analyzing and visualizing the collected data from research projects involving both the old way of stand-alone GPS devices and the new way with smartphone applications. This has given a lot of hands on experience with the different tracking technologies available for such data collection and has revealed both pros and cons of these technologies. Throughout the work there has been a feeling that mobile applications clearly are the future. However, existing solutions such as Endomondo and Open GPS Tracker are not made for this kind of use, and is therefore suffering clear drawbacks that a customized solution would not create (Hansen 2011) (Knudsen, Harder and Simonsen, et al. 2011).

In this project we will therefore aim to explore and present a solution for a mobile application and a supporting framework that can positively contribute to evolving the methodology for research and public participation. In order to be able to set some goals for the development for such a platform, we will first explore two previous experiences using tracking technologies. This is to establish where previous solutions have failed and which strengths must be kept or further developed.

## 1.1 Previous experiences

Until now the research carried out at Diverse Urban Spaces has been using two different setups for conducting GPS based surveys; handheld GPS devices (Lommy Personal) and an open source application for Android phones Open GPS Tracker (Care4all u.d.) (Open GPS Tracker u.d.). Both of these technologies and the experiences with them will be described and discussed in the following paragraphs based on two recent surveys carried out by the DUS research group in Aalborg Øst and Vollsmose. The purpose of this is to establish the individual strengths and weaknesses of each tracking method, so it can be determined where improvements are possible.

### 1.1.1 The Aalborg Øst Case

The project in Aalborg Øst involved 20 young people between the ages of 15 and 18 which were given a small GPS device called a Lommy Personal (Figure 1) which tracked their movement for a week (Knudsen, Harder og Hesselkjær, et al. 2012).

Figure 1 - The Lommy Personal

The Lommy Personal is designed to be a tracking device for tracking people as well as mobile assets such as cars or trucks. It works by combining a GPS receiver with a GSM/GPRS modem. This means that it uses the GPS satellites to obtain a position and the GSM/GPRS network to send this position along with other information obtained in the GPS positioning to a server owned by a company called Care4All. This information includes GPS-time, speed and direction of travel, accuracy of the positioning and the number of satellites used. The positions could then been sent from Care4Alls server to DUS's MySQL server through HTTP GET calls (Flextrack u.d.).

Conducting GPS surveys with the Lommy Personal reveals the devices' strong sides as well as its weak aspects. A very positive feature about the device is the shear amount of data it delivers. It is not just a set of coordinates, but alsometadata about the positions. Another positive aspect is the Lommys ability to have customized settings. It is possible to program it with a set of parameters tailored to the project at hand. These settings include the frequency of positioning and various methods of making the device discard positions that do not meet a certain accuracy level or distance since last position (Flextrack u.d.).

Although these are very positive features, there are also some negative aspects to the Lommy. Working with the Lommy is very labor intensive. Setting up and keeping track of all the devices takes a lot of time. Because the Lommy is focused on a specific purpose, the device is not supportive of any other

technologies, so if for instance text or picture documentation is needed along with the GPS tracks a separate service will have to be set up to support this. One of the main drawbacks to the Lommy is the lack of a visual user interface. The device features a single big red button and a series of indicator diodes. Regarding the blinking diode, for instance, the various patterns can have different meanings, and the red button has different effects solely based on length and pattern, so although it sounds quite simple it can be quite cumbersome and time-consuming. This makes it very troublesome for the inexperienced to interpret and understand the interface, which in turn makes it difficult for them to know whatthe battery level is, and whether or not the device is functioning properly (Harder, et al. 2011). In addition, the Lommy is not part of the respondents' daily routines. So during the survey it is easy for them to forget about the device, resulting in them running out of battery or simply forgetting to bring the device along with them, which in turn decreases the accuracy of the survey. Another issue with the Lommy is the very limited battery life. The setup for this survey resulted in a battery life of only approximately 7 hours, which meant the respondents were forced to bring the charger along if they would be gone for a longer period of time. This also meant that they would have to seek out places to charge the device while away from home.

### 1.1.2 The Vollsmose Case

The project "At tegne Vollsmose med Fødderne" was a GPS based survey commissioned by Odense Kommune where 20 young people between the ages of 16 and 20 were tracked for a period of two weeks, in the suburb Vollsmose in Odense. The purpose of the project was to map the actual use of the urban space in the suburb as well as its connections with the surrounding city (Knudsen, Harder and Simonsen, et al. 2011).

The GPS-survey in Vollsmose was conducted using smartphones with the Google Android operating system. To handle the actual tracking of the respondents an application called Open GPS tracker (Figure 2) was used.

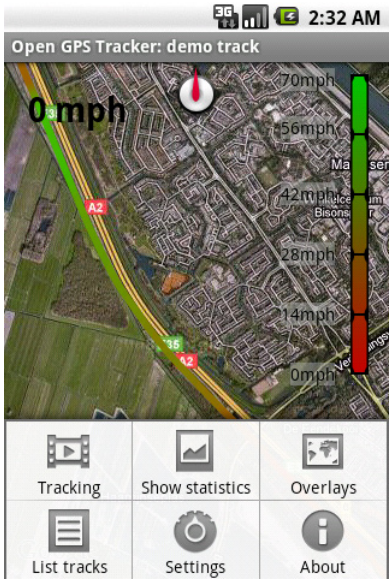



Figure 2 - Open GPS tracker

This tracking software enabled the respondents to track their movements and submit them to the survey. This setup required that the respondents manually opened the application and started a track each time they went for a trip in the physical surroundings. Then, when they did not wish to be tracked anymore they would access the application again and end the track. As well as creating the GPS-tracks the application also allowed the respondents to create geo-tagged images and text messages (tags). The respondent would then have to export the recorded tracks, images and text messages to a PC and from there upload the data to a website set up for the project. The data from the upload was then inserted into a MySQL database for storage and later analysis. (Knudsen, Harder and Simonsen, et al. 2011)

Conducting a GPS-survey using an Android based smartphone in conjunction with the Open GPS tracker had both positive and negative facets. One of the major positive aspects was that the phone, as a device, was already a part of the respondents' everyday life. This made it easy for the respondents to remember to bring the device with them through their daily activities. Another good feature of the smartphones was that there is a clear graphic display which tells the user exactly what is going on, such as battery status, GPS status etc. Having this information enabled the respondents to act accordingly, so their device was most often charged and running.

One of the downsides is that the data collected with the application was not detailed enough for more advanced analysis. The tracks created consisted of a start and end time and an encoded polyline which had to be decoded to get the individual positions. This meant that it was not possible to tell how long the respondents stayed at specific locations, only that they passed through at some point between the start and end times. Other metadata about the GPS positioning was not available either, such as the accuracy of the positioning, the number of satellites used or the speed and direction of travel. A big hurdle in the data collection was that the respondents needed to manually upload the data via a PC. This was an annoyance for the respondents, and it was necessary to contact some of them in order to make them upload their data. Some participants did not have a PC at home and therefore had to walk somewhere else to upload their data, which could have influenced their movement patterns thus decreasing the accuracy of the survey. Another issue was the decrease in battery life on the phones. In order to ensure that the tracking could be done through a whole day the respondents were given an additional battery for the phones. (Knudsen, Harder and Simonsen, et al. 2011)

## 1.2 Development goals

The Lommy Personal and the combination of an Android Smartphone and the application Open GPS Tracker both have advantages and disadvantages, as summarized in Figure 3.

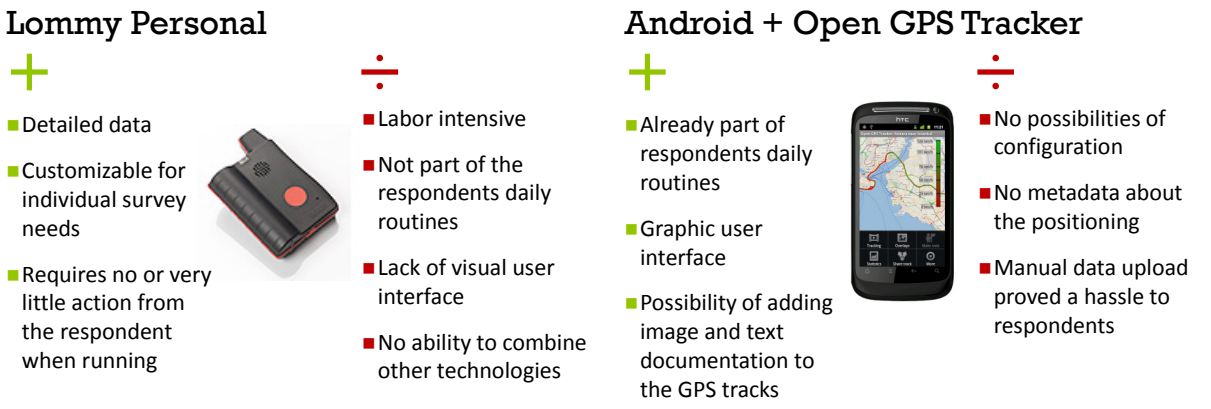



Figure 3 – pros and cons for existing tracking technologies

Both solutions provide data that can be used for mapping the movement patterns of the respondents but unfortunately they also both come with some significant drawbacks. Therefore there is a need from the Diverse Urban Spaces research group to acquire a better method of tracking respondents in order to improve the research.

A solution to this problem could therefore be to develop a new mobile phone application which could incorporate the positive aspects of both the Lommy Personal and Open GPS Tracker, avoiding the drawbacks and perhaps even adding new functionality that can facilitate new aspects of the research. This is the challenge we will attempt to solve through this project. The goal is therefore to create a new solution based on a smartphone application that is capable of:

- Providing detailed data.
- Being customizable to the needs of individual surveys.
- Being simple for the respondents to use.
- Having a clear graphical user inferface.
- Not interfering with the respondents' daily routine.
- Utilizing other data such as text and images.
- Having a battery life sufficient for a full day of tracking.

In addition to the smartphone application, a web map platform will be created to enable the exploration of data collected by the application as well as creating spatial analysis. The goal of the web platform is that it must be viewable in an internet browser, and not require any installation of software and must perform speedily and smoothly.

# 2 Thesis statement and research questions

The research group Diverse Urban Studies at Aalborg University has been carrying out studies in peoples movements in and around cities since 2004 (Diverse Urban Spaces n.d.). For the most part this research is based on GPS-surveys where selected participations are tracked using GPS in order to map their movement patterns. Traditionally the GPS-surveys have been conducted using a handheld GPS device called Lommy personal. Limitations and challenges in the use of the Lommy have caused researchers to look for other and more suitable methods of tracking the respondents. One attempt at this has been the use of Android based Smartphones in conjunction with the software application Open GPS Tracker. This method also proved troublesome. It did resolve some of the issues there was with the Lommy, but it presented some new and greater challenges that turned out to be more problematic than the ones it solved. There then lies a challenge in finding or creating a new method that provides a solution to these problems.

The goal of this project is therefore to explore this challenge; the thesis statement is therefore:

---

**How can empirical studies on mobility be improved using mobile phone applications?**

---

To help guide the project work in researching this problem a series of research questions are formulated. The research questions are as follows:

1. How can the shortcomings of mobile devices (short battery life, strange behavior when mobile coverage is poor and the Scatter of GPS signals) be minimized?
2. How can a smartphone application ensure the participation of the respondents?
3. How is it possible to create an automated process to create and display tendency analysis without the need for GIS software?
4. Which strengths and weaknesses arise when using respondents' own devices?

# 3 Reading guide

The purpose of this chapter is to give the reader an overview of the contents of this report. It will outline how the work from thesis statement to conclusion is documented. The remainder of this report is divided into 5 main chapters as described below:

**Theory and methods**
This chapter consists of a description of the different technologies and methods that are used in the implementation. The goal of the chapter is to generate the knowledge to create a smartphone application and appertaining web map platform that will improve the quality of mobility research.

**Implementation**
In the implementation chapter the development of the smartphone application and web map platform will be documented. The focus of the chapter will be the key functions, giving examples of the programming code and the choices made during the development process.

**Testing**
In this chapter the developed smartphone application and web map platform is tested in a real world scenario using 10 students from Viborg Katedralskole. The purpose of this chapter is to document the testing process and its results.

**Discussion**
The discussion is a chronological evaluation of the entire project period. It will review the development process and discuss the applied theories and how they affected the choices made in the implementation. The chapter will also discuss other potential choices and how they would have affected the outcome.

**Conclusion**
The last chapter is the conclusion. Here the four research questions and the thesis statement will be answered using the experiences and knowledge gained throughout the project.

# 4 Theory and methods

The aim of the project has now been established as the development of a platform that enables the use of smartphones to collect data as well as a web platform for the exploration of the collected data. It is therefore now necessary to examine the different tools needed in order to develop such a platform. This chapter will be split up in six sections, each describing an important aspect. The goal of this chapter is therefore to generate some knowledge about the different methods and technologies and thereby enabling us to make choices accordingly in the implementation of the platform. The content of the six sections is described below.

**GPS tracking**
When utilizing the GPS technology there are a lot of aspects to take into account. In order to fully understand the data collected it is necessary to have an understanding of the technology, how it works and what types of errors it suffers from.

**Databases**
Collected data will need to be stored in a manner where it is fast and easy to access and can be manipulated. For this purpose databases are ideal. In this section the basic theory of database design and optimization will be outlined.

**Web mapping**
In order to create a usable web map platform the different elements that go into the creation of such a platform are described. The different challenges that will face such a development are also discussed in order for us to make conscious decisions regarding these issues in the implementation.

**Mobile platforms**
There are several options to choose from when it comes to developing a smartphone application. In order to select a specific target for the development of the application the different options are weighed.

**Graphical User Interfaces**
When developing an application it is important to create a user interface that makes it easy for the user to understand what is going on and how to act to it. Therefore we discuss some basic principles of designing a user interface, and the challenges that lie in developing for the small screen sizes of a smartphone.

**Application development**
Being able to control the development is very important. Keeping track of key development goals and time management is necessary to ensure that a working system is created before the deadline. In this section we therefore examine development methods that will help achieve the development goals.

Each of the six sections will end with a summary that will highlight the challenges we face in an implementation as well as describe which method/technology was chosen and argue why this choice was made.

## 4.1 GPS tracking

In this section the technology used for tracking will be examined to give an idea on which results we can expect from it. The technology that will be used for locating and tracking people is the Global Positioning System (GPS). The section goes through the basic knowledge on GPS before looking into the reasons for possible errors and end with a description of Assisted GPS which is used in smartphones.

The GPS system is the well suited for tracking people over larger areas since the device doing the tracking is carried by the user and because the system works all over the planet as long as there is a view of the sky. (Dueholm, Laurentzius and Jensen 2005, 19) In order to fully utilize this technology and be able to interpret the data created, insight into the workings of the technology is needed. It is necessary to have an understanding of how the GPS positions are created and the different sources of errors associated with it. The chapter will only deal with the issues associated with C/A code measuring, which is the method used in smartphones. (Dueholm, Laurentzius and Jensen 2005, 9)

The Global positioning system consists of a network of 31 active satellites orbiting the earth at an altitude of approximately 20,200 kilometers. (United States Naval Observatory 2012) (United States Naval Observatory n.d.) The satellites continuously emit a signal, containing the time it was sent, precise information about the orbit of the satellite (ephemeris) as well as the rough orbits and status of the other GPS satellites, which can be picked up by GPS receivers. The receiver then measures the time it took the signal to travel from the satellite to the receiver. Based on this measurement of time the distance between the satellite and the receiver can be calculated. By simultaneously calculating the distance to multiple satellites (minimum 3) with known positions, the position of the receiver can be deduced (Dueholm, Laurentzius and Jensen 2005, 11). The principle of this method can be seen in Figure 4.

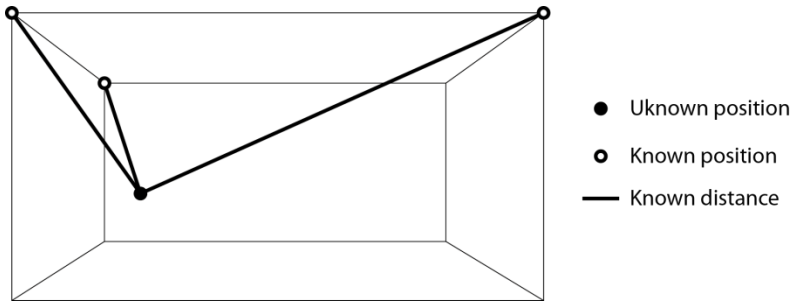



Figure 4 - Principle of GPS positioning (Dueholm, Laurentzius and Jensen 2005, 11)

The GPS therefore relies on knowing where all the satellites are at any given time as well as having an accurate time of when the signal was sent and when it was received.

Since acquiring GPS positioning requires a lot of calculation by the receiver it is very power consuming, which poses a potential problem for mobile devices, being that they rely on battery power. In Appendix A test is done of how long continuous positioning can be done using a smartphone. This revealed that there is a potential for tracking throughout a day without the need for charging the device.

In order to work out the time it took the signal to travel from the satellite to the receiver, both are equipped with clocks. The GPS satellites contain extremely accurate atom clocks that are constantly updated by control stations on the ground, but the clocks in the receivers are not nearly as precise,

resulting in a possible error in the positioning. (Dueholm, Laurentzius and Jensen 2005, 29) This problem can however be overcome be estimating the error in the receivers' clock by measuring to a minimum of 4 satellites. Therefore a minimum of 4 satellites are desired for a more accurate position. The orbital information on the GPS satellites is, like the clocks, continuously updated by control stations to make the information send accurate enough to make measurements for tracking purposes. (Dueholm, Laurentzius and Jensen 2005, 38)

Because the signal has to travel more than 20,000 kilometers another factor for precise positioning is the forces influencing the signal as it travels through the different layers of the earth's atmosphere. Free electrons in the Ionosphere alter the speed of the signal and different temperatures, pressures and the humidity in the stratosphere and troposphere cause the signal to refract. These sources of errors vary over time and are handled by the receiver, reducing them by using models of the acting forces. Therefore the accuracy of a GPS positioning can vary even though the visible conditions of the receiver are the same. (Dueholm, Laurentzius and Jensen 2005, 38-42)

Another factor influencing the accuracy of GPS positioning is the constellation of the satellites. To describe this the term "Dilution of Precision" (DOP) is used. The DOP value describes how good the conditions are, 1 being ideal, over 4 being critical and above 6 the position should be discarded. The principle of DOP is illustrated in Figure 5.

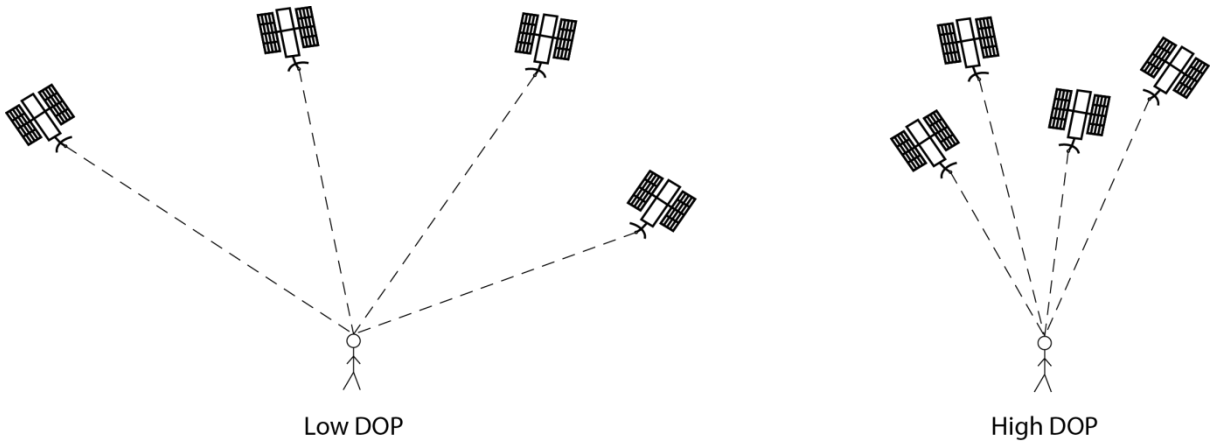



Figure 5 - Satellite constalations at low and high DOP values (Dueholm, Laurentzius and Jensen 2005, 45)

If the satellites are clustered together in the sky when measuring, the geometry for performing the trilateration is bad and the inaccuracies of the individual distance measurement will have a bigger effect on the calculated position. The best conditions for positioning are therefore when the satellites are spread out over the visible sky, causing the best geometry for trilateration (Dueholm, Laurentzius and Jensen 2005, 44-49).

One of the main issues when using GPS in urban areas is multipath errors. This type of error occurs when the GPS signal is bounced off objects before it reaches the receiver.

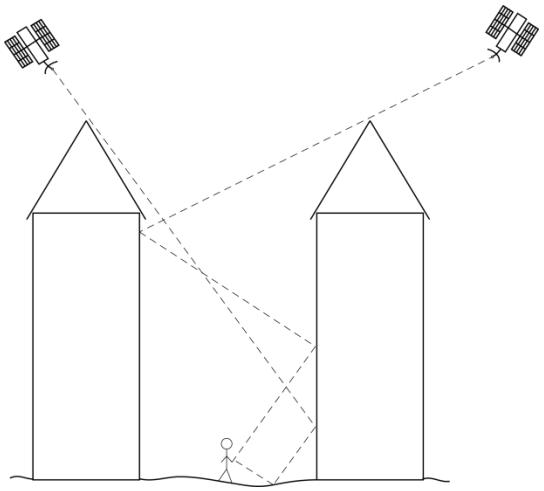


Figure 6 - Illustration of Multipath (Dueholm, Laurentzius and Jensen 2005, 43)

As seen in the illustration, Figure 6, the path the signals travel from the satellites to the receiver is lengthened, thereby taking longer to reach the receiver. This makes it calculate a longer distance to the satellite, decreasing the accuracy of the position. The problem is most likely to occur when in urban areas with tall buildings where a signal can be bounced multiple times, but it can also happen in open areas if a signal bounces off the ground. Modern GPS receivers have filters that can detect some of the multipath signals and discard them, but it continues to be a problem. (Dueholm, Laurentzius and Jensen 2005, 43)

All these errors result in a dispersion of approximately 6 meters, depending on the conditions. This means that 68.3% will be within 6 meters of the true position. 95.4% will be within the double dispersion (12 meters) and 99.7% will be within three times the dispersion (18 meters). (Dueholm, Laurentzius and Jensen 2005, 9)

Smartphones have an aid in improving both the position and the time it takes to achieve the first fix called Assisted GPS. Smartphones today use what is known as Assisted GPS (aGPS). aGPS means that the device is assisted by an outside source, aiding it in performing some of the tasks associated with the positioning. This assistance includes information about the satellite orbits and clock information, as well as an initial course position of the device. This data is transferred to the device via the mobile network. Having this information lowers the amount of calculations carried out by the device, resulting in a quicker acquisition of position and lower power consumption. (Jarvinen, DeSalas and LaMance 2002)

The accuracy of GPS in smartphones will vary depending on the specific device, but according to the studies done in Appendix B, accuracy better than 15 meters can be expected from a stationary position, even in the most difficult conditions. While moving the results from Appendix C show that in most cases the precision will be better than 10 meters. Some positions however have a much worse accuracy, so in order to obtain an accurate track some post processing is still needed.

To summarize the use of GPS in smartphones several things can influence the accuracy of the position, and it is necessary to be aware of these in order to fully understand the data. The GPS signal has to travel a long distance from the satellite to the receiver, making the signal prone to interference and errors. Calculating the correct distance between the satellite and the receiver can be influenced by the accuracy of the clock in the receiver, atmospheric disturbances, number of satellites visible and their constellation, as well as the

surroundings of the receiver. Therefore any given position is somewhat inaccurate, depending on the conditions at the time of positioning. These inaccuracies will result in the raw tracks not being exactly the route of the device. The accuracy of the tracks can be increased by analyzing the raw data, removing positions with a too high DOP-value or with a low number of satellites used. Despite these inaccuracies the GPS capabilities in smartphones are regarded as being good enough for tracking purposes.

## 4.2 Databases

The basic idea of databases is that data has to be converted into knowledge. The concept is that data added a context is given the status of information. If the information can be used to make sense it is evidence. When enough evidence is collected it will become knowledge.
An example could be a database containing X and Y coordinates of peoples position every 10 second. This is basically only a set of integers. However, if you know the spatial reference and that it is taken every 10 second from home to work it becomes information. If this information is used to conclude something about time used waiting for the traffic lights to turn green is becomes evidence. If this is done enough times the evidence about an imperfect traffic light configuration becomes knowledge and can be corrected, resulting in a much easier trip from home to work.
In order to collect enough data for it to be interpreted into knowledge it has to be stored. A solution could be a file added more and more content. This solution is somewhat poor if you consider speed and usability and a better solution would be to use a database.

There are some fundamental functions, which should be addressed if the database is to be useful. First of all it has to be *secure* to prevent unauthorized access of data and enforce strict rules in who has write/read access and who does not. Furthermore *reliability* is necessary to ensure the data is always accessible by the user and that power failure does not hinder data update. The data in the database has to be *correct and consistent* in order to assure trust and functionality. This is achieved in a relational database through carefully declaring the right type for each column and ensuring no duplicate entries are stored. The last requirement for a successful database is it has be *technology proof,* which in short means that new technological development should not hinder the data user from achieving maximum gain from the use of a database. (Worboys 1995)

The type of database we have access to in this project is MySQL 5.0 which is a **R**elational **D**ata**b**ase **M**anagement **S**ystem (RDBMS). However, other systems like Oracle and PostgreSQL have adapted ideas from object-orientated programming and therefore support extensible data types, objects etc. To do the same in a RDBMS would result in *"an explosion of tables, many joins, poor performance, poor scalability, and loss of integrity"* (Objectivity 2005, 5). So in the future a meaningful step would be to adapt to another DBMS, but for this project MySQL will have to do. However the main strength of MySQL is that it from the beginning has been developed with speed in mind, which could become handy when dealing with the potential millions of points a large scale GPS tracking project can collect.

RDBMS's arose in the 1970s and is a system consisting of a number of tables. Each table consists of columns/fields with a defined content. The data is placed in each row with a column chosen as the primary identification key. (Bell 2007)

A conceptual model can be created in order to create an understanding of the entities and relationships of a database. This can be done using plain language such as English or better with a conceptual database diagramming language, because it follows terms equally understood by everyone familiar with the language. An example is the entity-relationship (ER) language which was popularized by Peter Chen and caught on because of its simplicity (Timothy L. Nyerges 2010). An example of an ER diagram can be seen below:

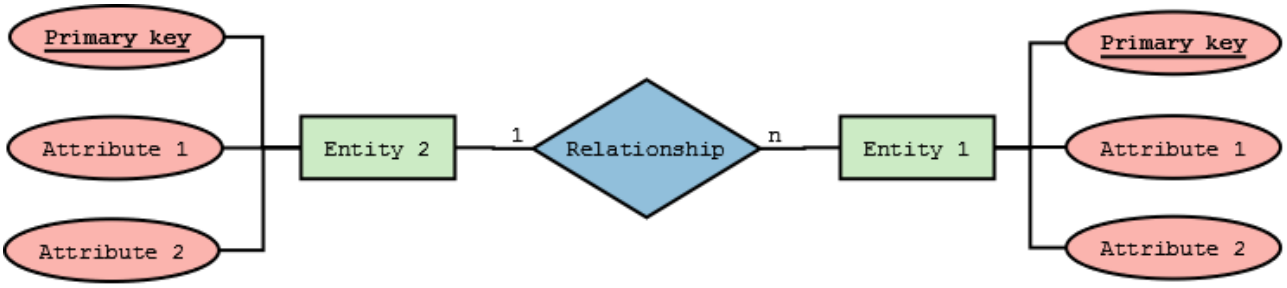



Figure 7 - Example of an ER-Diagram

The building blocks of an ER diagram are entities, attributes and relationships, as seen in Figure 7. The entities translate into tables in a RDBMS. The entity then has some attributes which are columns in RDBMS. The last type is relationship which is what binds the different tables together. The different relationships between tables can then be divided into: *one-to-many* (one record in a table relates to many records in another table), *many-to-many* (if several records in a table relates to many records in another) and *one-to-one* (if exactly one record in the first table relates to exactly one record in the other table). (Shekhar and Chawla 2003, 36)

Another element of databases is the uses of different data construct types, for example geospatial data types like points, lines and polygons. However as we are going to use Google Maps API (see chapter 4.3) to construct the visual and analytic elements of this project, we do not need spatial enabled data but a speedy database system instead.

Optimization is an important part of working with databases. One way of optimizing a RDBMS is known as normalization. Normalization is a process of removing all redundant data from the tables. When applying first normal form all repeating groups of data are split up into separate tables. By applying the second normal form any data that is repeated is again split up into table. The third normal form goes even further and separates all attributes that are not directly dependent on the primary key, into separate tables (Gilmore 2000). Another method of optimizing a RDBMS is applying indices. When selected columns of a table are stored in an index the search time of the columns is greatly improved, thus reducing the runtime of a query. This is possible because the RDBMS will not have to go through the table line by line searching for the desired record, but can instead search for a keyword in the index, and the jump straight to the record. This ability does however come at the expense of speed when performing insert and update queries (Shekhar and Chawla 2003, 17). Yet another optimization tool is creating views. Using views you can predefine queries designed to optimize speed and reduce returned rows, inefficient joins and use integrity checking when an insert or update query is being executed. It is also possible to optimize the table structure (Schneider 2005) giving the columns the right type. Speed is maximized when a 3 digit integer is stored as a 3 digit integer and not MySQLs default 45 character variable character field. More advanced the PACK_KEYS option could be used, which will normally make updates slower and reads faster (Oracle 2012). Another option is to define the way rows should be stored. The optimal path would be to set the ROW_FORMAT to dynamic and define an average row length and the number of rows you plan to store in the table (Schneider 2005, 59). A last skill is to disobey the rules of normalization, for example if you tend to calculate information with SUMS it could be wise to schedule a daily update of a column storing this information. Another example is with an attribute containing less than 4 different values. In most cases storing it directly in the table instead of creating a separate table for it, would be wiser. (Schneider 2005)

As mentioned earlier the DBMS use in this project is MySQL. Within MySQL we must select which storage engine to use. A storage engine is the part of DBMS that takes care of storing and retrieving data, which is accessed through an internal APPI and can be accessed through an internal API. From version 5.5 of MySQL the default is InnoDB and before that it was MyISAM. The latter excels in speed in both queries and insert statements only achieved in InnoDB with a plugin introduced in MySQL 5.1 and later. However the lack in crash recovery, transaction management and foreign key support might be a drawback in some cases (Oracle Coorporation 2011).

To conclude this chapter we will be using a RDMBS called MySQL version 5.0 with a store engine called MyISAM. ER diagramming will be used to display the database structure. In the design and implementation there are a number of things we need to take in to consideration in order to optimize MySQL, such as table structures, normalization of tables and the creation of views.

## 4.3 Web mapping

Web Mapping is a term that covers the visualization of geo data on a map that is presented through the Internet (Fu and Sun 2011, 11). For this project it will be used to visualize the collected data through a browser based map platform, serving as a way of conducting visual analysis of the collected data as well as displaying geographic analyses. This chapter will therefore investigate how this process is made possible, were certain considerations are key, and which challenges you face in order create a platform that is able to make data relevant for the user.

In order to be able to show a map through a web browser there are certain elements that need to be in place. These elements are the services (servers) needed in order to take some data, process it, combine it with other data and display it as an image in a user's browser. This process and its elements can be seen in Figure 8.

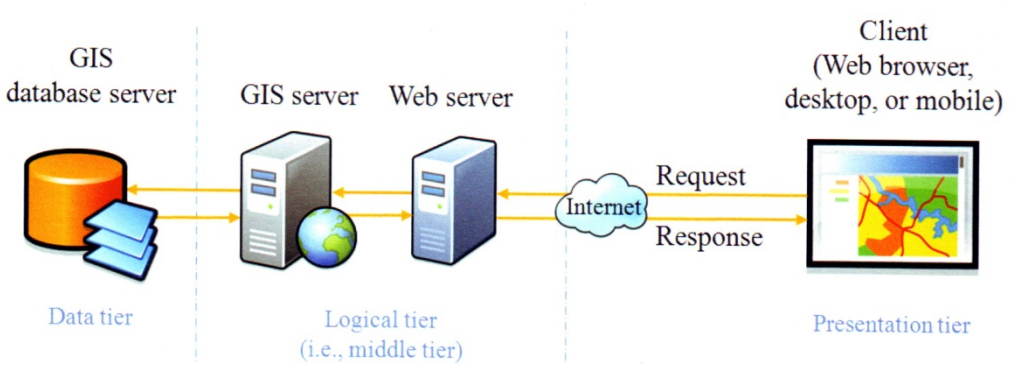



Figure 8 - Logical architecture and workflow of a basic Web GIS (Fu and Sun 2011, 33)

Figure 8 shows that when the user makes a request for a map through the Internet it is processed by the webserver, determining which information is wanted. The webserver then sends a request to the "GIS server" instructing which map view is wanted. The GIS server will then determine what data is needed to create the requested map view, retrieve this from the database and submit the map to the webserver that creates the final view for the client to display to the user (Fu and Sun 2011, 33).

When it comes to instructing the different servers on how to process the requests given to them there are a number of different options to choose from. On the web server there are different programming languages that can be used, such as PHP or Python, and in the browser or client, HTML and JavaScript etc. (Fu and Sun 2011, 29). There are also several ways for the GIS server to deliver the map data. For this purpose Open Geospatial Consortium (OGC) has created a wide range of standards in order to ensure interoperability between platforms. These span from simple images being delivered through WMS, to GML (and the simplified KML) which describes individual features and can contain metadata about these, to the more advanced WFS that also enable manipulation of data (Open Geospatial Consortium n.d.).

Just as there a many options to choose from when programming server instructions there is also a choice to make when it comes to the client. The client is the platform that enables the user to explore the data and provides the user interface (Fu and Sun 2011, 35). Some of the big contenders in providing web-mapping platforms are Google Maps, Bing Maps, OpenLayers and ArcGIS. All these solutions provide APIs that enable the developer to style and customize the maps as well as create overlays of desired data. Although

they all provide tools to do this, they all come with very different terms and conditions regarding use, and are developed with different focuses, which is something a developer must take into consideration when choosing a platform. Esris ArcGIS Web APIs are very closely linked with Esris commercial products such as the Arc GIS Server. OpenLayers is an open source platform and is therefore not subject to paid development, and Google and Bing Maps are driven by advertisement (Google Developers n.d.) (Microsoft n.d.) (OpenLayers n.d.) (Esri n.d.).

When creating a system for web mapping there are a series of challenges that you need to overcome in order to generate a well-performing platform and a good user experience (Fu and Sun 2011, 39). As seen in Figure 8, the process of displaying a map involves a lot of server and database activity. Each of the activities involved can create bottlenecks that will slow down performance. It is therefore necessary to ensure that the servers' capabilities match the amount of work you need them to do, i.e. the more users you have and complex tasks you require the more server capacity you need, and the more you need to optimize the system. Instead of making the servers do all the work, another option is to leave some of the work to be done by the client. While doing this will lower server activity it may also diminish the user experience. When leaving work to be done by the client, the fluidity of the experience will be very much dependent on the computing power of the device used to view the map (Fu and Sun 2011, 40).

Utilizing web mapping on smartphones provides a lot of opportunities, enabling the user to explore data while on the move, and at the users convenience (Fu and Sun 2011, 115). Although this is a very positive development it does come with some limitations and challenges that will need to be addressed. The limited system resources in a mobile device greatly improve the need for optimization, limiting the computing power needed in by the client, either making only simple map operations or leaving more work to be done server side. Relying on the server to do most of the work can also pose a problem when on a mobile device. The bandwidth of an Internet connection on a mobile device can vary greatly depending on where the device is, and users may find themselves in situation where no network connection is available, leaving the user unable to use the map. The size of the devices themselves poses a challenge as well. The small screens on smartphones do not allow for, as much information to be shown as you could on a desktop computer, so the maps displayed must be adapted to accommodate these limitations (Fu and Sun 2011, 126).

The essence of web mapping is the process of accessing stored data in a database and displaying it to the user in a client through the Internet. The platform chosen in this project to facilitate this is Google Maps, utilizing the Google Maps JavaScript API V3 (Google Developers n.d.). This approach was chosen mainly because it is a very familiar platform for many users, providing an interface that will be easy for most users to be familiar with. Being the large platform that it is, it also provides a large development community, giving easy access to tutorials and troubleshooting. The Google Maps JavaScript API also supplies a lot of tools for customizing maps, geocoding, and creating custom overlays, making it a powerful platform for creating web maps. Another factor in deciding on Google Maps is the group members' prior experience with developing for this platform, hence reducing the learning curve.

## 4.4 Mobile platforms

When developing a smartphone application it is necessary to choose which platform to use. This chapter will look into the different options available and which is preferable for this project. The first choice is to decide on a cross platform solution or a native application designed specifically for one platform.

Choosing a cross platform solution can be very desirable, simply because it allows the developer to target a larger audience with less work. One option is to create the application as a webpage using HTML5 (W3C 2012). This is not an option when creating a tracking application since a webpage requires the browser to be open and running constantly, hindering the user from using their smartphone as they normally would while tracking. Another cross platform solution is PhoneGap (PhoneGap n.d.). This also relies on HTML5, but in conjunction with a JavaScript API, making native functions available. The code is then loaded through a web view in a native application on the different devices. Therefore it only requires a minimum of OS-specific coding. While this method is extremely useful in some cases it does not allow for background processing and is not suited for "heavy lifting" (PhoneGap 2012). When designing a tracking application it is necessary to enable it to continue running in the background while the user continues using their phone as normally. It is also important to have full control of the data and the way it is collected. Native programming allows for the creation of an application where the developer has full control of how the device handles different functions, making it easier to optimize code and do troubleshooting. Therefore a native application is the best option for this project.

Another option is to decide which platform the application should be developed for. There are a number of different operating systems available on the smartphones for sale. The main choices on the market today are Googles Android, Samsungs Bada, Apples iOS, Microsofts WinPhone, Research in Motions Blackberry and Nokias Symbian. Their individual market shares of sales in the 1st quarter of 2011 are seen in Table 1.

| Operating System | Market Share (%) |
|---|---|
| **Android** | 56.1 |
| **iOS** | 22.9 |
| **Symbian** | 8.6 |
| **RIM** | 6.9 |
| **Bada** | 2.7 |
| **Microsoft** | 1.9 |

Table 1 - Smartphone OS market shares, 1st quarter 2012 (Gartner 2012)

As seen from the table above, the most popular operating systems for smartphones are Googles Android system followed by Nokia's Symbian and Apple's iOS. The idea is to be able to track an as large as possible portion of the population and Android is therefore the best choice. However, there are some other aspects to be aware of before choosing Android.

Android is an open source operating system for smartphones developed by Google. In addition to being an operating system it also includes several key applications and middleware, such as integrated web-browser, SQLite database and hardware controls for camera, accelerometers, GPS etc (Android Developers n.d.).

The Android platform has several strengths in comparison with its competitors. Since Android is open source, it is extremely accessible as a developer. There are no initial charges and it requires no special hardware to start developing, as is the case with Apples iOS. Therefore the Android developer-community is very large and filled with people willing to help by offering tutorials and answering questions on bulletin boards. Another great feature of Android is that you have total control of the distribution of your application. There are no requirements that an application should be approved before it can be distributed, and distribution is not locked to a single distribution channel such as iPhone applications are with the AppStore. Testing with larger user groups is therefore very easy.

On the negative side the lack of an approval before distribution, results in a lot of malware for the Android platform. Users will therefore have to be more alert when downloading content from an unknown developer. Because there are several manufacturers of Android devices continuously making new models, they do not always make the new Android versions available for older devices. This fragmentation is important to keep in mind as a developer, since it results in not all devices being able to utilize new functionality added in the later versions of the OS. Development must therefore always be done with the lowest possible Android version in mind, in order to reach the biggest possible target audience.

## 4.5 Graphical User Interfaces

In this chapter we will get a basic idea of what makes a functional user interface. This will give us a theoretical knowledge when we are going to create the interface of our mobile application. This chapter will be mainly be based on the book *"GUI Bloopers 2.0: Common User Interface Design Don'ts and Dos"* (Johnson 2007) . This book states 9 basic principles:

**Basic Principle 1: Focus on the users and their tasks, not on the technology.** We have to decide on a specific user group to design our application for. This target group will be associated with the strategic goals for our application. We therefore have to look into the characteristics of the potential users. How savvy are they with mobile use? How do they normally perform tasks on their devices? What is their knowledge of the underlying task the mobile device performs. Should the application tell them to turn on the GPS, or will they know this is a prerequisite for GPS tracking? It is therefore necessary to involve people with the same characteristics as the intended users in the designing process. Another idea is to create insightful people as you do in the theatrical world, describing childhood, family life, professions, lifestyles and so on. This all add up to a 3 part process including: empirical investigations (users and context of use), collaboration with the user group and the context in with the application is designed (the firm). (Johnson 2007, 8)

**Basic Principle 2: Consider function first, presentation later.** It is important to first make a conceptual model on how the task should be done and then think about GUI Layout afterwards. The concepts used should be "*as simple as possible, but no simpler – Albert Einstein"* (Johnson 2007, 20). It is necessary to take actions from the users task-domain so that the user quickly understands what he/she has to do in order to facilitate successful behavior. Therefore object/action analysis is the most central of a conceptual model. The different objects the user interacts with will in most cases have a mutual relationship, interacting with each other. A lexicon of nomenclature used in the application to give a comprehensive appearance both internally in the application and to the user. Based on the explained preliminary examinations it is now possible to create use case and rich picture scenarios, which create the top level basic for the application. (Johnson 2007, 18)

**Basic Principle 3: Conform to the users' view of the task.** Do not make the user do unnecessary tasks which the user does not see a point in doing. That would create a product that would not only seem amateurish, but also makes the application unintuitive.. An example could be to ask for a home telephone number when most of the user group only has a mobile phone or exposing settings which the user has no reason to be confronted with. The developer has to decide where on the power versus complexity scale the application should be or simply hide them until they are needed. (Johnson 2007, 26)

**Basic Principle 4: Design for the common case.** Rarely used functionalities should make room for the more commonly used functionalities. The design should therefore take this range into account when designing the GUI. This can be divided into how many users will use functionality and how frequently the functionality is going to be used. A basic principle is: *"For features that people use once or twice a year—e.g., setting installation or configuration options - minimizing keystrokes is completely irrelevant compared to ease of remembering and clarity of instructions."* (Johnson 2007, 32)

**Basic Principle 5: Don't distract users from their goals.** The application should not give the user information that is not needed. The users perceptual skills has no need to do complex problem-solving/ reasoning by elimination to figure out what behavior is needed to complete the given task. (Johnson 2007, 35)

**Basic Principle 6: Facilitate learning.** Learning to use an application takes time, so if the user is not obligated to use it he/she will simply quit. The problem is that the programmer often has an inside-out perspective on the application. They use their knowledge of the structure of the application to judge if the screen/controls make any sense. However, the users only have the knowledge seen on the display.

The programmer has to think about ambiguous meanings, both textual and graphical. The way to overcome this is by being consistent. Even small inconsistencies will force the user to think about it, misleading them from their task. Here the conceptual model again will help discover operations which can be done using equal tasks. The problem with consistency is that it is difficult to define, has a multidimensional angle and is subject to interpretation. Therefore the developer must observe the users to find their perception of the tasks and make the application so the user cannot do anything wrong. (Johnson 2007, 37)

**Basic Principle 7: Deliver information, not just data.** The users obtain their information from data. The focus should therefore be on delivering the desired information not an explosion of data. Consider which question the displayed data should answer.

In order to make a successful display the programmer has to conquer these goals: *Visual order and user focus:* the user has to be directed to the data on the screen that matters. *Scannability:* it should be easy to scan the display for the needed data. *Match the medium:* the display and controls should be scalable to match the device they are being used on. The programmer should consider using different controls and screen items sizes based upon the device. *Attention to detail:* every detail counts, and it is therefore required, to look thoroughly at every aspect of the display. *Display inertia*: only elements the user edits should change and not the entire view. (Johnson 2007, 42)

**Basic Principle 8: Design for responsiveness.** Nobody likes to wait, and when it comes to application usability waiting time is equal to a failed product. However, there is always some processing the application needs to do. The key here is to give the user an instant responce to his actions, let the application do long processes in the background if possible, let the user know the system is working and how long he can expect it to take and if it is too long for the user he/she should be able to cancel it. (Johnson 2007, 45)

**Basic Principle 9: Try it out on users, then fix it!** Testing the application on real "user-like" testers is alpha-omega. A central part of programming is correcting problems found in testing. Testing has two goals; informational and social. The informational goal is to find the elements of the UI causing trouble for the testers and using the exact nature of the problem to suggest improvements. It is important to make usability tests even when the application is only partial finished, for the application to be adaptable to outside influence. (Johnson 2007, 48)

To sum-up: An effective user interface needs to focus on how the users do their task and not what is technically possible. It is useful to make a conceptual model on which functions the application should facilitate. Do not make unnecessary steps for the user to achieve the task. The display should be focused on common tasks and be scalable to fit whatever device the users have. It is therefore also important to

consider what information the user wants to get and not what data he wants, and it should be delivered quickly and smoothly. Testing the application in every step of the development process is therefore the key to overcoming all the other bloopers.

Part of this project is to develop a smartphone application. These devices have a smaller screen and a relatively slow processing unit. This adds some additional problems when it comes to the GUI. The processing should be kept to a minimum to ensure fast response. This is because a mobile device is used on the go and the patience of the users is therefore minimal. Because of the small screen size and the fact that the device is not normally used in optimal viewing areas the display has to be designed accordingly. The screen layout should be simple and the most used functions should be on top with detailed information kept away in menus. The button should be large to ensure an easy touch interface and the application should employ touch functionally to make handling easier. The textual output should be brief and the use of pictures is always preferable (Android Design n.d.).

## 4.6 Application Development

Developing an application is a complex procedure and requires some planning in order to achieve maximum effect of the workforce. In order to manage this process effectively the ideas behind different development theories are studied.

The most basic development model is the waterfall model where different steps are finished before going on to the next, as seen in Figure 9.

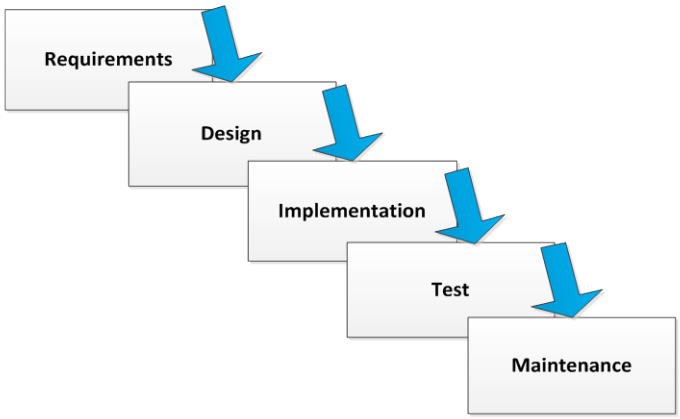



Figure 9 - the waterfall model

The steps needed to go through are:
**Requirements** – Defining the requirements for the software, its functions and behaviors.
**Design** – Designing the data structures, how the interface representations should look and other details.
**Implementation** – Coding of the software, establish the database.
**Test** – Debugging the application.
**Maintenance** – Developing a plan to make the system run continuously.

The steps have a built-in assumption that the preliminary steps have been completed. It therefore might not be a perfect model if you are working with a new toolset or the finished product is not clearly defined. It is best suited when the project scale is clear (Buisness eSolutions n.d.).

In a development environment the need for quick adjustment to new opportunities and threats means that a looping development process might be needed. An example of is the “The Development Cycle” (Figure 10) (Harmon and Anderson 2003, 216). The different approach here is that the steps has been arranged in a cycle where only basic tasks are done in the first cycle and then more complexities are added as the process goes on through the cycles until the end product is completed. This type excels where new tools or ideas have to be tested or conceptualized before use.

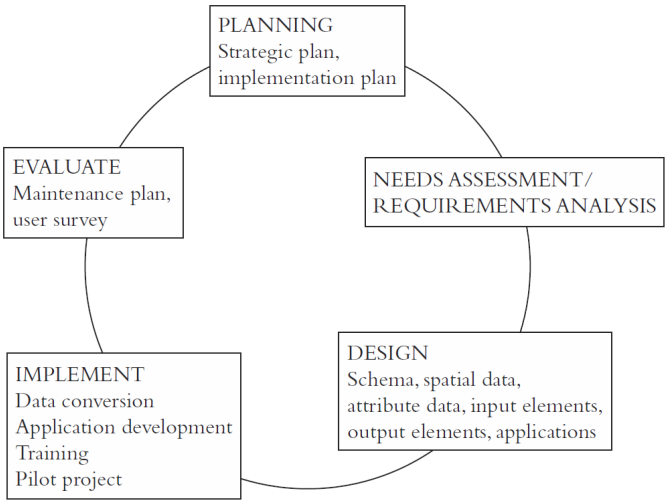



Figure 10 - The development circle

This model also ensures a good portion of adaptability. However, it lacks good project management tools.

To overcome this, Agile Scrum was developed as a model for project management. This is the model that is used in the development in this project. Agile Scrum (Figure 11) is based on a main document called product backlog, where all the elements of the finished application is divided into pieces that require no more than a maximum of 30 days to complete. Each piece, called a sprint, is a fully working application ready for shipment. The sprints are then divided into pieces of maximum 24 hours to complete. These pieces are documented in a sprint backlog (Schwaber og Sutherland 2011).

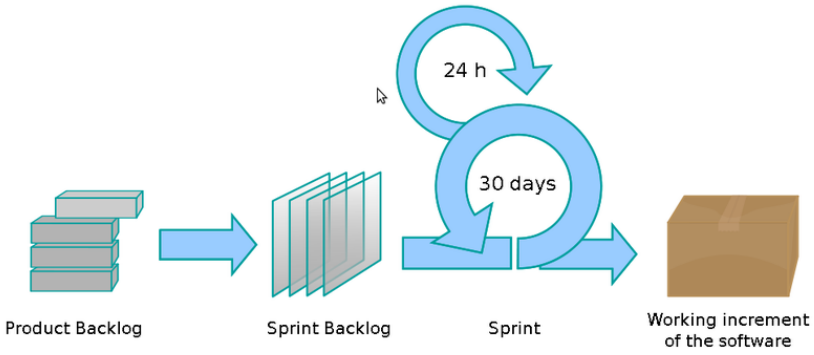



Figure 11 - Agile Scrum Model (NobleProg Training Materials 2012)

In the beginning of each day a meeting (daily scrum) is held where the entire development team informs each other about:

1. What they have done since last Daily Scrum?
2. What will be done during the next?
3. What problems arose?
4. What new useful knowledge has been gained?

In this meeting all participants stand up to ensure focus and to keep a scheduled time of 15 minutes. In this time all new knowledge gained is explored, and the approach to finish the sprint is determined to ensure a high valued increment for the finished product (Schwaber og Sutherland 2011, 10).

A product owner is in charge of the product backlog and has the task of ensuring that the most value is gained from the product. A scrum master closely monitors the process and is the link to the development team, making sure that they obey the rules of Scrum. The development team turns the product backlog into increments and creates the sprint backlog (Schwaber og Sutherland 2011, 5). To help keep track of progress in sprint and product backlog, burn down charts is used, divided by day or sprint according to type and the value is the time needed before completion (Figure 12).

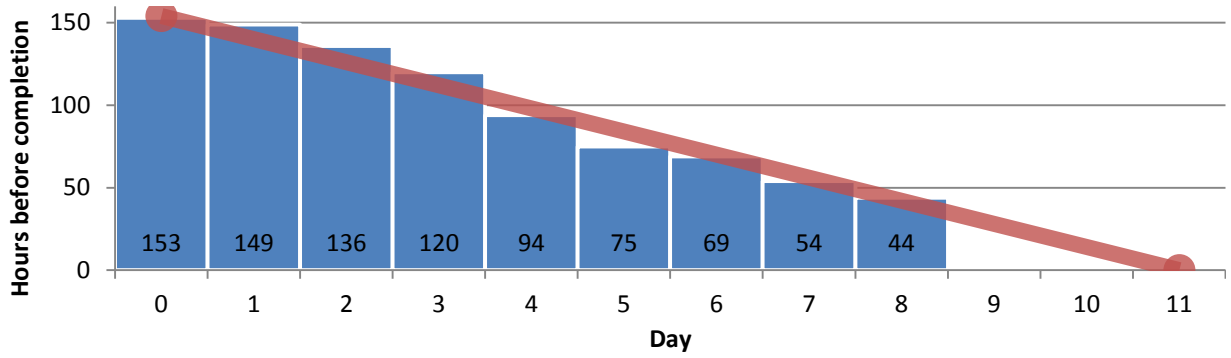



Figure 12 - Burn down chart

After a sprint is finished a review- and retrospective meeting is held. The review meeting is about how the sprint increments are working and if modifications are needed to the product backlog. The retrospective meeting is about how the project group has been working (Schwaber og Sutherland 2011, 11).

For the implementation done in this project, the product and sprint backlog will be created as means of keeping track of the development process. The daily Scrum meetings will be held in order to ensure that both team members are always up to date on the work progress and the workings of the implemented functions, as well as the sprint review and retrospective meetings.

# 5 Implementation

As described in chapter 1.2 the aim of this project is to develop a tool that can be used in researching movement patterns. This tool will consist of two main elements: The Android application and a web based visualization platform. This chapter will illustrate in detail how this was described and implemented. The chapter is divided into the five sections described below.

**The smartphone application**
In this section the 'use'-case scenario of the application will be described as a means to create an overview of the different functions and possibilities that lie within the application, and the different choices made in the UI design will be discussed. When the basis of the application has been described on a more technical account, it will be given the individual functions using samples of the code.

**Database design**
A database will be set up to support the application and store all the data collected. The structure of the database, its tables and their attributes will be visualized using an Entity-Relation Diagram (ERD).

**Data handling scripts**
In order to store the data submitted by the smartphones running the application in the database, a series of PHP scripts are set in place on a web server. These scripts perform several tasks such as file handling and geocoding addresses, which will be described in this section.

**Spatial analysis**
One of the goals of the project was to attempt to create spatial analysis outside the traditional desktop GIS systems. This section will describe the three analyses created in this implementation: creation of polylines, heat map and time span.

**Web map platform**
A web based map platform was created to visualize the collected data and performed analyses. This creates a public platform where it is possible to get an up to date view of the collected data with the ability to freely turn on and off different layers.

Figure 13 shows how these five elements work together to create the full system.

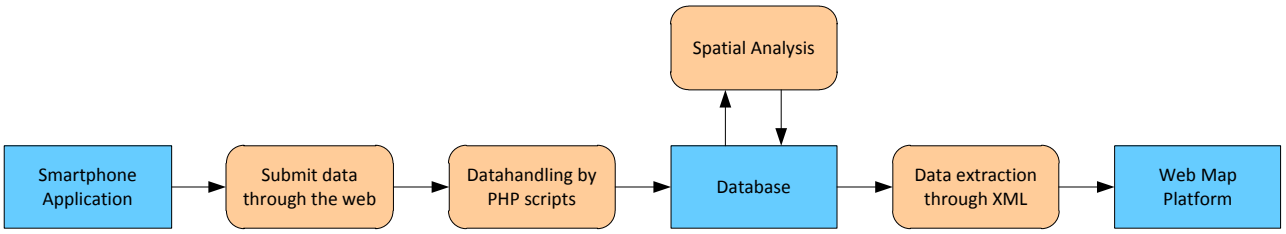



Figure 13 - Structure of the implemented system

## 5.1 The smartphone application

The smartphone application is made up of several different views containing many functions. In this chapter we will create an overview of how the application is structured and how some of the key functions works both in practice and programmatic. The application can be downloaded installed from http://app.particimap.dk (requires a smartphone with Android 2.1 or greater). Screenshots of all the different views and options in the application can be seen in Appendix D.

### 5.1.1 Use case scenario

The 'use'-case scenario will aim to create an overview of the different functions and possibilities that are present in the application. Illustrating the different actions possible and outlining the structure will help to give an overall picture of the application, making it easier to comprehend when more detailed descriptions of the coding of individual functions are given in later paragraphs.

The installation process is handled by a URL link to an installation file on a webserver. The link can either be created by the browser on the device or through a QR code (Denso Wave n.d.). The respondents then download and install the application, which means an internet connection is necessary. Because the download is done outside Google Play the user must allow installation of software from unknown sources (Google Play u.d.). The first time the application is opened they have to accept an EULA and afterwards submit required user data before they can start using the core functions in the application. The process from download to the application is opened in the start view is seen in Figure 14.

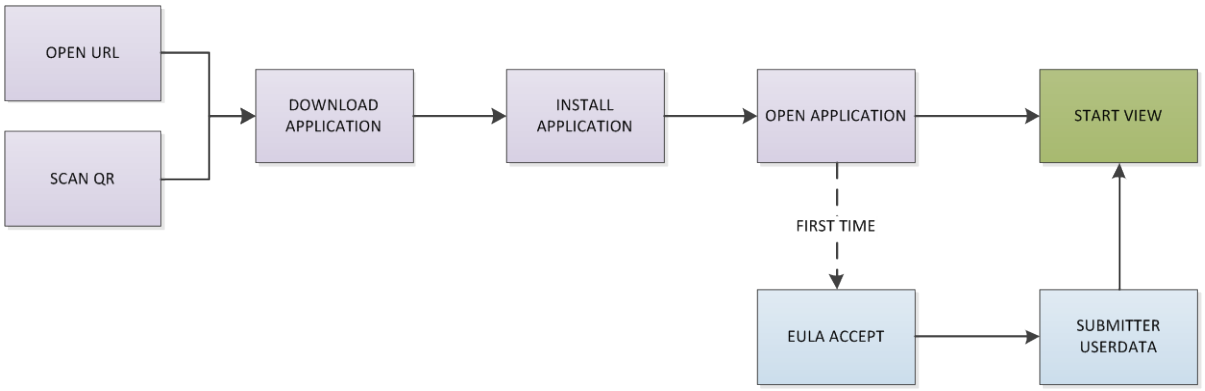



Figure 14 - Overview of installation process

In the start view it is possible to go to the other views and open the menu options, which can be done in all views except user data. The basic functionalities of the start view is the start and stop buttons which is used to register the start and stop of trips. When the start button is pressed a popup box appears where the type of transportation can be selected and when the stop button is pressed the felling or "mood" about the trip can be selected and it is possible to type why that particular mood was selected. This process can be seen in Figure 15.

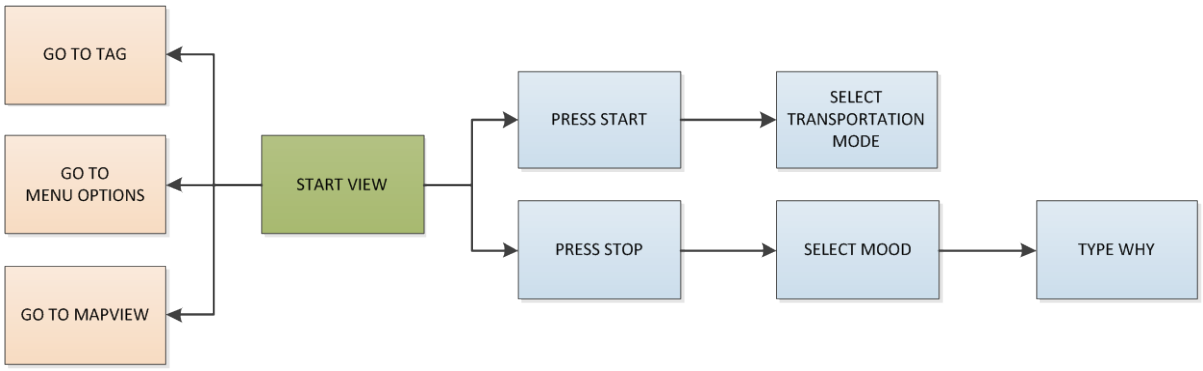



Figure 15 - Functions in the start view

In the map view, submitted GPS data is shown as polylines on a map from Google Maps. This map gives the possibility to scroll and zoom in the map and to initialize Google Maps Street View, as seen in Figure 16.

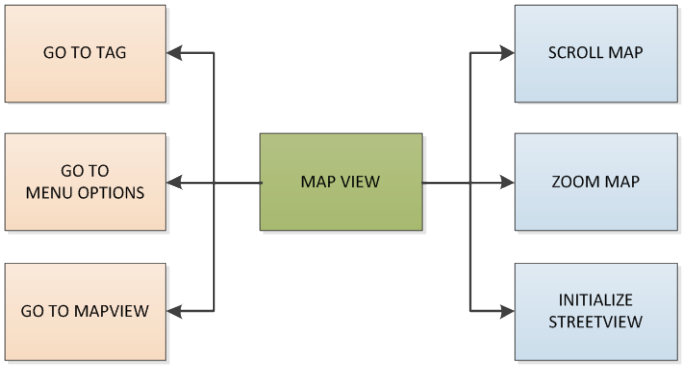



Figure 16 - Functions in the map view

As seen in Figure 17, the tag view allows for a selection of category for the tag as well as adding some explanatory text and a picture before submitting the data by pressing the “submit” button.

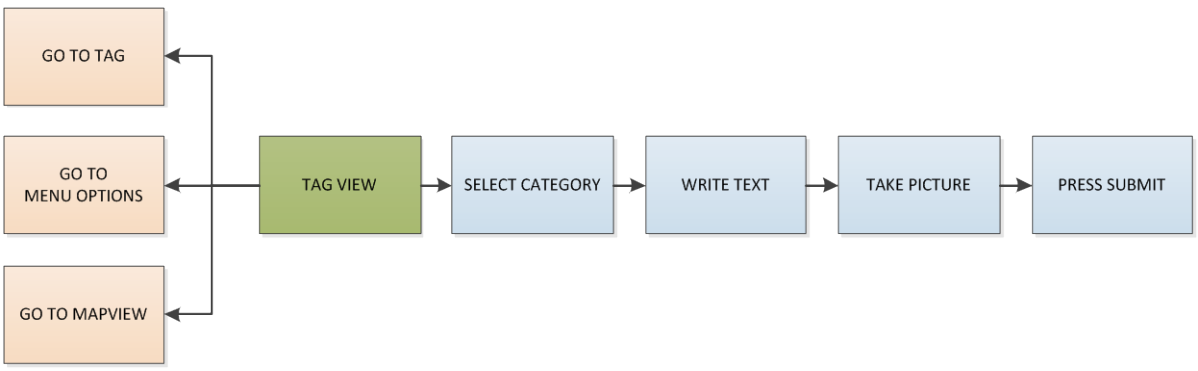



Figure 17 - Functions in the tag view

When the menu button is pressed in any of the three previously described views, it is possible to select three different actions (Figure 18). This allows them to stop the application from logging their position, and to go to the user data to edit their data as well as give them the option to send a text message directly from the application to the members of the project group if problems or questions arise.

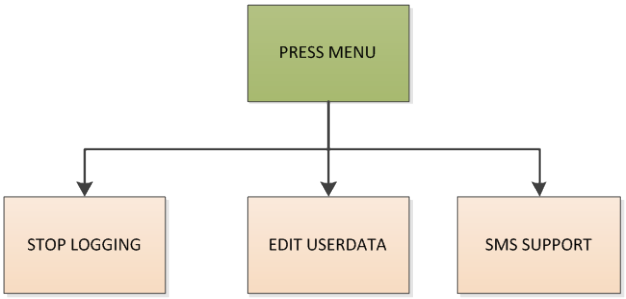



Figure 18 - Pressing the menu button

### 5.1.2 Graphical User Interface (GUI)

The GUI consists of all the elements that the user sees on screen and interacts with. The design of this interface is very important in order to ensure that the user experience is the best possible. This paragraph will describe the choices made in the design of the application based on the guidelines from chapter 4.5.

In order to make the application feel like an integrated part of the Android OS, the design was made using standard designs for things such as buttons and input fields whenever possible. The application was also given a semitransparent background that allows the users home screen wallpaper to shine through. This makes the application feel more connected to the device, which is important when you want the user to actively use the application in their daily routines.

The whole application is structured by using a “tabbed view” i.e. all the important selectable views are presented to the user at all times. This gives the user a clear idea of the different possibilities in the application. The main functionalities are divided into 3 different “views”: the “Start”, “Tag” and “Map” views. The reason why it is split into separate tabs is to minimize the information the user needs to deal with at a given time, and the small display size of the smartphones limits the amount of information that can be displayed at once. Functions that are not commonly used, such as the possibility to kill the GPS service, contact support and edit the user data are put in the options menu, so it does not take up space for the key elements, as seen in Figure 19.

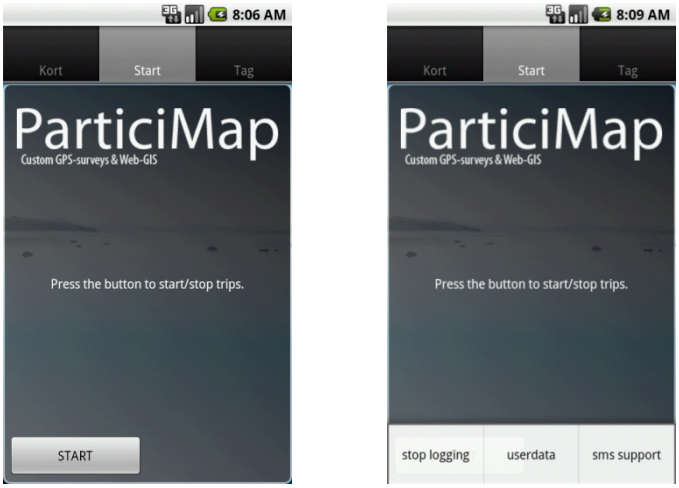



Figure 19 - The start view and the menu

The “Start” tab is designed with simplicity as its key focus. On this screen only a logo, the text “press the button to start/stop trips” and the start or stop button is displayed. A key feature of this view is the Start/stop buttons which are the only elements for the user to interact with. Only one of these buttons will be visible at any given time, with the start button visible when no trip is set, and the stop button visible when a trip is set. This is done in order to ensure that the user has a clear image of whether or not a trip is started. The text displaying which mode of transport is currently selected is updated whenever changes are made, so the user can always see if the settings are as desired.

### 5.1.3 User data

In order to distinguish between the respondents it is necessary to collect and store some information on them. This is done by making the respondents fill out a form when the application is first launched. The form includes the following: Name, Address, Phone number, Email, Gender and Age. In addition to identifying each respondent this data can be used for statistical purposes as well as cloaking their addresses in public data visualizations. When first launching the application the user is also asked to accept some terms and conditions for using the application. This is necessary since the GPS tracking is basically an invasion of people's privacy, and therefore their consent is needed.

A dialog asks the user to accept the terms associated with the use of the application is created as an AlertDialog with the added feature of only being shown if it has not been accepted before. Therefore a check is done to see if this is the case. The check is done on a value in the SharedPrefferences called "eulaKey".

```
Boolean hasbeenshown = prefs.getboolean(eulakey, false);
        If(hasbeenshown == false){
```

If this statements  returns false the alertdialog will then be created like so:

```
AlertDialog.Builder builder = new AlertDialog.Builder(mActivity)
        .setTitle(title)
        .setIcon(R.drawable.ic_launcher)
        .setMessage(s)
        .setPositiveButton(android.R.string.ok, new Dialog.OnClickListener() {

            public void onClick(DialogInterface dialogInterface, int i) {
                // Mark this version as read.
                SharedPreferences.Editor editor = prefs.edit();
                editor.putBoolean(eulaKey, true);
                editor.commit();
                dialogInterface.dismiss();
            }})
        .setNegativeButton(android.R.string.cancel, new Dialog.OnClickListener() {

       public void onClick(DialogInterface dialog, int which) {
             // Close the activity as they have declined the EULA
       mActivity.finish();
       }
        });
AlertDialog eulaAlert = builder.create();
eulaAlert.show();
```

The AlertDialog contains a title, the application icon, the text itself as well as an ok-button and a cancel-button. The ok-button sets the "eulaKey" value to true before closing the dialog, ensuring that the dialog will not be shown the next time the application starts, whereas the cancel-button just ends the activity. Finally this is added to the onCreate-function of the TabHost in order to have it shown on launch.

```
new SimpleEula(this).show();
```

A very similar check to that of the terms is also done on startup to determine if the user data has been filled and if this is not the case it will redirect the user to the user data page.

```
SharedPreferences Userdata = getSharedPreferences("Userdata", 0);
boolean hasBeenShown = Userdata.getBoolean("Respondents", false);
if(hasBeenShown == false){
        startActivity(new Intent(Tab.this, Userdata.class));}
```

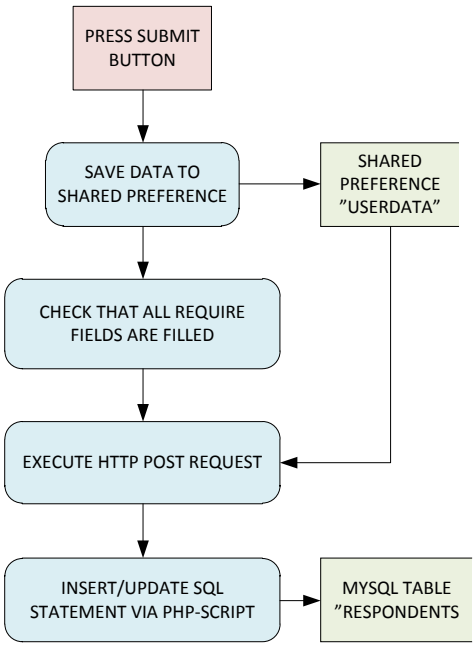



Figure 20 - Submitting user data

The user data page contains a series of input fields and a submit-button. When the submit-button is pressed all the values entered are stored in SharedPreferences as well as being posted to an URL. However, before this is done a series of checks are made in order to ensure that all the required information has been filled out. An overview of this process is seen in Figure 20, where red boxes indicate a user action, blue boxes are processes done by the application and green boxes represent data stores.

```
if (name.length()==0) {
        Toast.makeText(getApplicationContext(), "Indtast venligst dit navn, for at fortsætte",
Toast.LENGTH_SHORT).show();
        return;
```

This check is done for every input and if one is left blank the user will be prompted and asked to fill it out. If none of the checks reveal blank inputs the value used to decide whether or not the userdata has been filled is changed and the data will be posted to a URL. In order to link the userdata with the specific device the "DeviceId" or IMEI is used. In order to be able to post to a URL the values are made into BasicNameValuePairs in an array.

```
ArrayList<NameValuePair> postParameters = new ArrayList<NameValuePair>();
postParameters.add(new BasicNameValuePair("name", name));
postParameters.add(new BasicNameValuePair("adress", adress));
postParameters.add(new BasicNameValuePair("housenumber", housenumber));
(and so on)
```

The dataarray is then added to the URL and executed before the application switches to the start view of the application.

```
CustomHttpClient.executeHttpPost("URL", postParameters);
```

### 5.1.4 Starting and stopping a trip

A key element in this application is the ability to distinguish between trips. A trip is defined as the timespan between the moment the user presses the start button and later presses the stop button. This information is needed in order to allow the user to declare the mode of transport and the experience with the trip, and link this information to the GPS locations. The overall process of this is handled using Shared Preferences, writing data to a file and sending the file to a webserver.

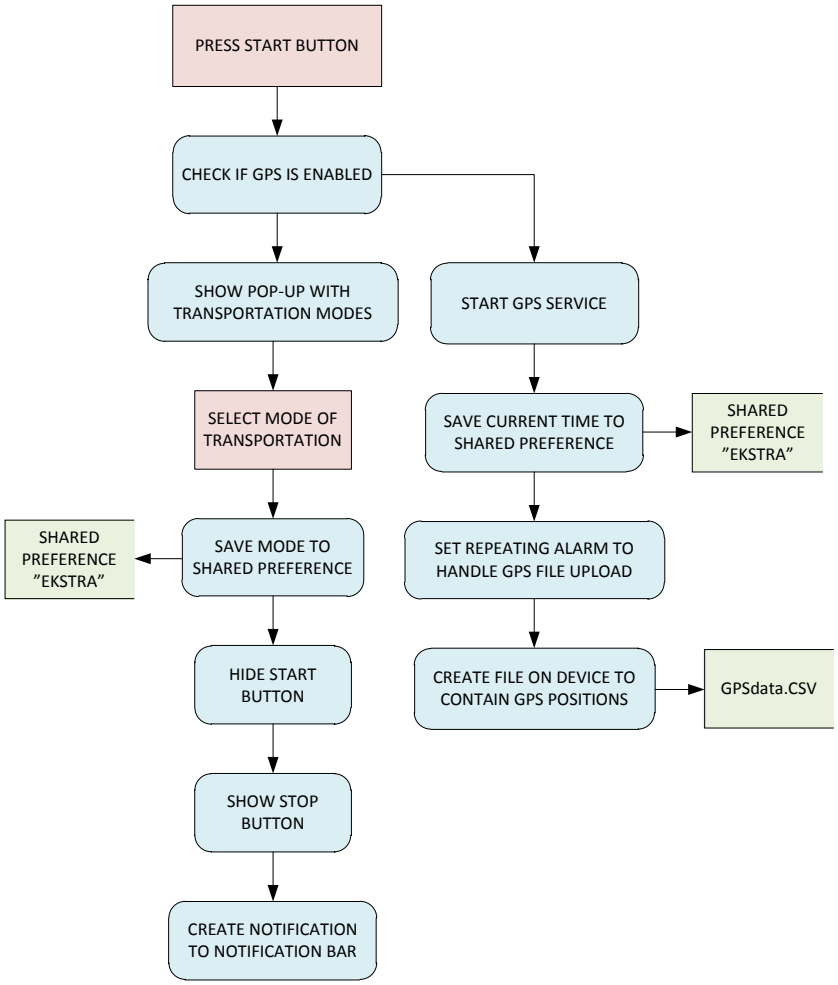



Figure 21 - Starting a trip

The process of recording a trip is initiated when the user presses the start button in the “Start” tab of the application, as seen in Figure 21. When this action is performed the current time is stored as a value in SharedPreferences.

```
SharedPreferences Ekstra = getSharedPreferences("Ekstra", 0);
SharedPreferences.Editor editor = Ekstra.edit();
editor.putString("tsStart", String.valueOf(System.currentTimeMillis()/1000));
```

A popup menu appears, prompting the user to input the mode of transportation, from four different categories, used for the trip.

```
startQuickAction.setOnActionItemClickListener(new QuickAction.OnActionItemClickListener() {
        public void onItemClick(QuickAction quickAction, int pos, int actionId) {
                ActionItem actionItem = quickAction.getActionItem(pos);

                SharedPreferences Ekstra = getSharedPreferences("Ekstra", 0);
                SharedPreferences.Editor editor = Ekstra.edit();

switch (actionId) {
case ID_BIL:  Toast.makeText(getApplicationContext(), actionItem.getTitle().toString() +" valgt",
              Toast.LENGTH_SHORT).show();
        editor.putString("transportType", actionItem.getTitle().toString());
        break;

case ID_KOL:  Toast.makeText(getApplicationContext(), actionItem.getTitle().toString() +" valgt",
              Toast.LENGTH_SHORT).show();
              editor.putString("transportType", actionItem.getTitle().toString());        break;

case ID_CYK:  Toast.makeText(getApplicationContext(), actionItem.getTitle().toString() +" valgt",
              Toast.LENGTH_SHORT).show();
        editor.putString("transportType", actionItem.getTitle().toString());
        break;

case ID_GAA:   Toast.makeText(getApplicationContext(), actionItem.getTitle().toString() +" valgt", T
        oast.LENGTH_SHORT).show();
        editor.putString("transportType", actionItem.getTitle().toString());
        break;
}
editor.commit();

startQuickAction.setOnDismissListener(new QuickAction.OnDismissListener() {
        public void onDismiss() {
              Toast.makeText(getApplicationContext(), "VÊlg en transportform",
              Toast.LENGTH_SHORT).show();}
        });
```

The selected transportation type is also stored in the SharedPreferences. A key feature here is that there is set an onDismiss function, which ensures that the application will not start a trip if the transportation type is not set.

When the user has arrived at the end of a trip the stop button is pressed, which in turn launches a series of events, seen in Figure 22.

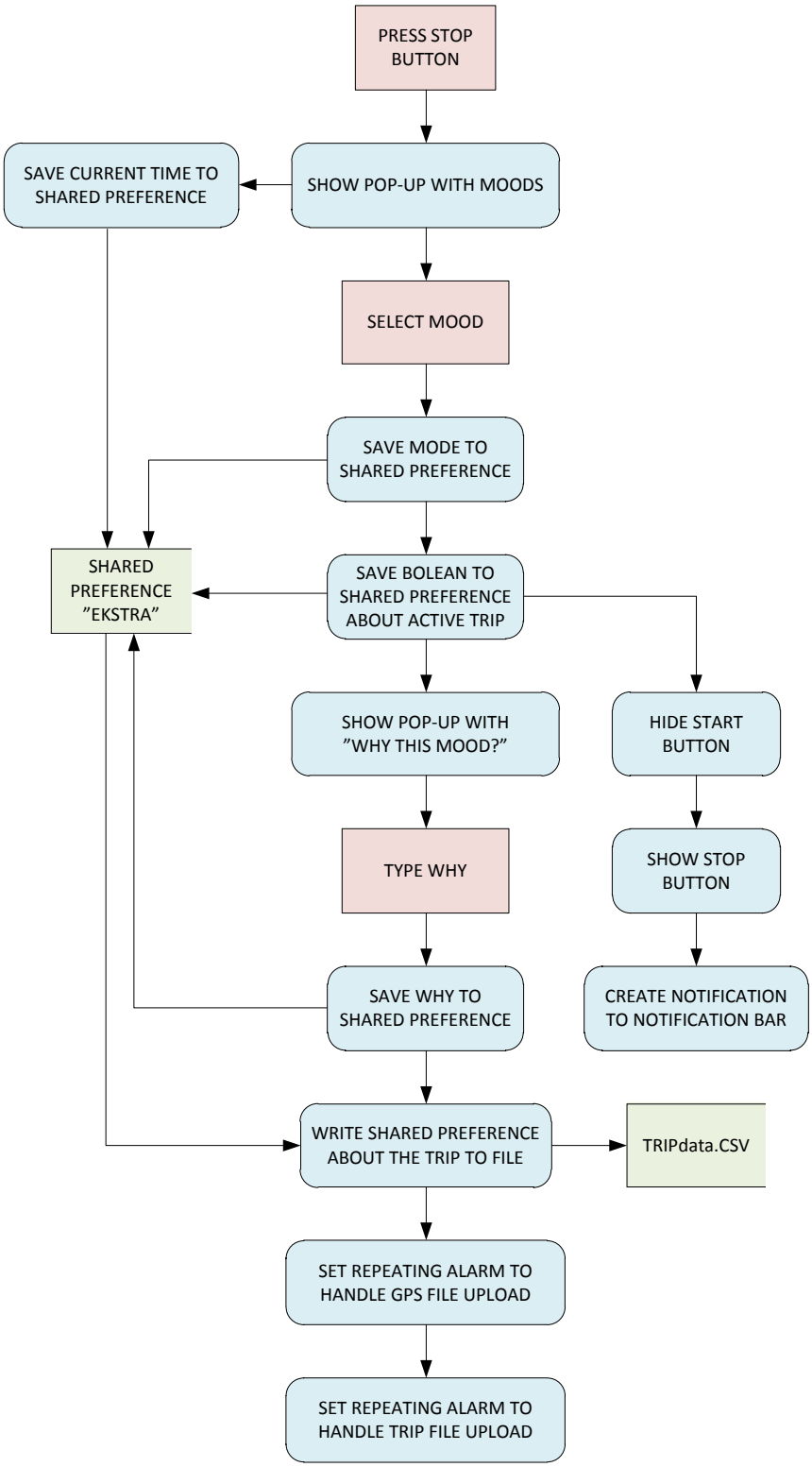



Figure 22 - Stopping a trip

Firstly the current time is stored in the shared preferences.

```
SharedPreferences Ekstra = getSharedPreferences("Ekstra", 0);
SharedPreferences.Editor editor = Ekstra.edit();
editor.putString("tsStop", String.valueOf(System.currentTimeMillis()/1000));
editor.commit();
```

Then the user is asked to state the mood surrounding the trip, which is done by selecting one of four predefined categories. This action is identical to that which stored the mode of transport. Once the category has been selected a dialog appears.

```
case DIALOG_FEELING:
    final EditText feelinginput = new EditText(this);
    return new AlertDialog.Builder(main.this)
        .setTitle("Hvorfor denne oplevelse?")
        .setView(feelinginput)
        .setPositiveButton("Ok", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton) {

        SharedPreferences Ekstra = getSharedPreferences("Ekstra", 0);
        SharedPreferences.Editor editor = Ekstra.edit();
        editor.putString("why", feelinginput.getText().toString());
              editor.commit();}})
              .create();
```

This dialog contains a text input for the user to describe why the specific mood was chosen. This input is also stored in the shared preferences.

Lastly all the data regarding the trip is needed to be sent to the webserver. This process is done by writing the data to a file and then sending the file through HTTP.

```
String device = String.valueOf(tMgr.getDeviceId());
String tsStart = Ekstra.getString("tsStart", "ingen");
String tsStop =       Ekstra.getString("tsStop", "ingen");
String type =  Ekstra.getString("transportType", "ingen");
String mood =  Ekstra.getString("mood", "ingen");
String why =   Ekstra.getString("why", "ingen");
String newline = "\r\n";

String filename = device + "_tur.csv";
FileOutputStream fOut = openFileOutput(filename, MODE_APPEND);
OutputStreamWriter osw = new OutputStreamWriter(fOut);

osw.write(device + ";");
osw.write(tsStart + ";");
osw.write(tsStop + ";");
osw.write(type + ";");
osw.write(mood + ";");
osw.write(why + ";");
osw.write(newline);

osw.flush();
osw.close();
```

All the values needed, including the device id, are retrieved, and a file is created. All the values are then written to the file and the transport type is reset to none. All that remains is to upload the file to the webserver. In order to achieve this task without requiring the user to wait for the upload or even be dependent on an Internet connection at the moment the trip is endedis done using a broadcast receiver.

```
Intent intent2 = new Intent(this, UploadReceiver2.class);
PendingIntent sender2 = PendingIntent.getBroadcast(this, 111, intent2,
PendingIntent.FLAG_UPDATE_CURRENT);
AlarmManager am2 = (AlarmManager) getSystemService(ALARM_SERVICE);
am2.set(AlarmManager.ELAPSED_REALTIME_WAKEUP, 60000, sender2);
```

To utilize a broadcast receiver an intent is created to that launches the actions in the UploadReceiver2.class, which handles the actual upload of the file. An alarm manager is created in order for the intent to be sent again at a later time in case the upload did not succeed the first time.

In the UploadReceiver2.class the file containing the data is accessed and a HTTP connection is established to the URL of the script that will handle the file on the webserver and the data is transferred using the POST method.

### 5.1.5 GPS service

The main feature of conduction GPS tracking is of course the ability to receive positions from the device's GPS receiver. This process is handled by creating a service that runs separate from the main application. This is necessary in order to bypass Androids ability to reduce memory usage by shutting down processes it does not see as vital when an application is running in the background. The creation of this service thereby removes the possibility of the operating system suddenly and without warning stopping the GPS resulting in a loss of data.

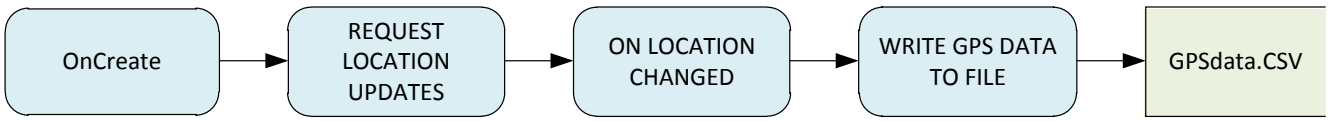



Figure 23 - GPS service

The GPS service is started the first time the start button is pressed in the application. An overview of the processes of the GPS service is seen in Figure 23. When Android requests location updates from the systems' location service it uses a location manager initializing the GPS.

```
this.lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
this.lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 5000, 0, this);
```

Here the location updates (GPS positions) are requested every five seconds no matter if the device has moved or not. This is chosen based on the assumption that a five second interval will give a good balance between the amount of positions needed to accurately represent the path traveled, the amount of data collected, as well as the results on battery life from Appendix A. When all positions are recorded no matter if there has been movement or not it is possible to deduce the length of a stay based only on the number of positions recorded in a given area.

With each position, latitude and longitude as well as a series of metadata are recorded. This includes: height, timestamp, speed, number of satellites, accuracy and direction as well as the device ID and its current battery level. The battery level is recorded in order to create data on the rate of battery consumption of devices running the application. This data is then written to a file in the same manner as the trip data.

```
String filename = device + ".csv";
FileOutputStream fOut = openFileOutput(filename, MODE_APPEND);
OutputStreamWriter osw = new OutputStreamWriter(fOut);


osw.write(ts + ";");
osw.write(lat + ";");
osw.write(lon + ";");
osw.write(speed + ";");
osw.write(dir + ";");
osw.write(sv + ";");
osw.write(batt + ";");
osw.write(acc + ";");
osw.write(height + ";");
osw.write(device + ";");
osw.write(newline);

osw.flush();
osw.close();
```

The file containing the GPS data is sent to a webserver using the same method as the file containing trip data, in order to ensure that the data will be sent even though there might not be an available Internet connection at the time.

If the users' GPS is not enabled on the device the user will be prompted and asked to turn it on.

```
public void onProviderDisabled(String provider) {
Toast.makeText(getApplicationContext(), "Tænd venligst GPS",Toast.LENGTH_LONG).show();}
```

This service will run in the background from the moment the user presses start the first time, but will not stop when the stop button is pressed. This is done in order to record all the data possible and not lose GPS data in the event that the user forgets to start a trip. This uses a lot of battery, but is necessary in the testing phase, in order to get an idea of how often this might occur. If the user would want to stop the tracking anyway, this is possible by pressing 'stop the service' via the options menu.

```
ComponentName comp = new ComponentName(getPackageName(), GPSservice.class.getName());
stopService(new Intent().setComponent(comp));

Intent intent = new Intent(this, UploadReceiver.class);
PendingIntent sender = PendingIntent.getBroadcast(this, 111, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
AlarmManager am = (AlarmManager) getSystemService(ALARM_SERVICE);
am.cancel(sender);
```

When the button is pressed the service is destroyed and the alarm manager requesting the file transfer every hour is canceled. As the service is destroyed the location manager stops requesting updates, and a last attempt at sending the data file is done.

```
public void onDestroy() {
              lm.removeUpdates(this);

        Intent intent = new Intent(this, UploadReceiver.class);
        PendingIntent.getBroadcast(this, 1, intent, PendingIntent.FLAG_ONE_SHOT);}
```

### 5.1.6 Tags

The ability to create tags is a possibility for the user to document elements of the urban environment and state their opinion of them. This is done by enabling the user to submit geo tagged text and images.

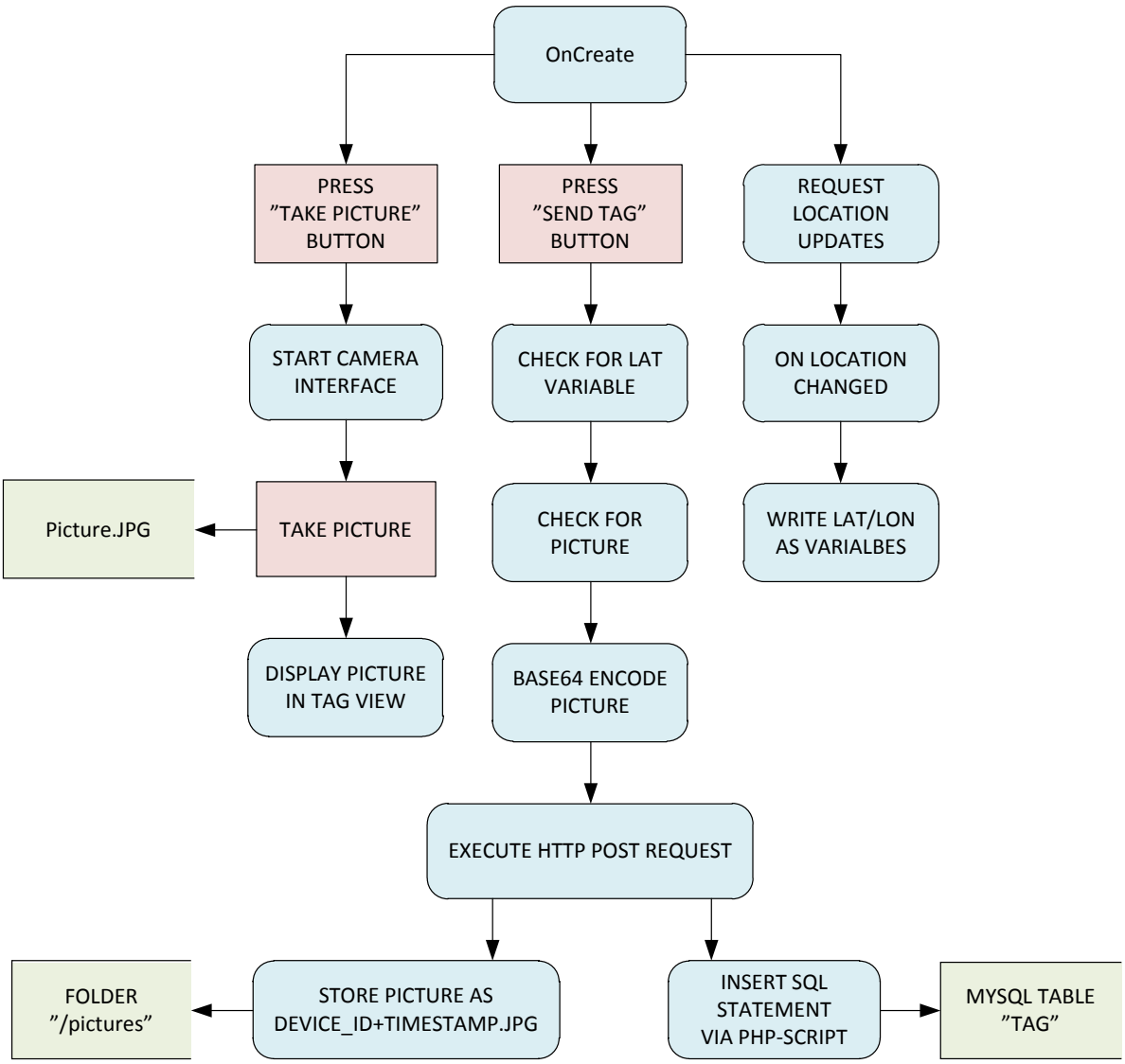



Figure 24 - Submitting a Tag

When the “Tag” tab is loaded in the application it starts its own location manager and requests location updates just like it is done in the GPS service. The reason it is seen necessary to start a separate service for this task is that it should be possible to create tags independently of whether or not the user is creating tracks. Since it is a separate location manager it should only be running while the user is viewing the tab. Therefore it is necessary to stop requesting location updates when the user leaves the tab and start requesting them again once he user returns.

```
@Override
protected void onPause() {
        super.onPause();
        locationManager.removeUpdates(locationListener);
}

@Override
protected void onResume() {
        super.onResume();
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,0,0,locationListener);
}
```

The first action required by the user is to select a predefined category for the tag using a spinner. A text input follows to give the possibility of stating an opinion. If the user wishes to add a picture pressing the "take picture" button will launch the devices camera application.

```
public void onClick(View arg0) {
              Intent intent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
              startActivityForResult(intent, 0);
```

When the camera application has been launched the application waits for a result. When the result is given i.e. the user has taken a picture, our application steps in again to handle the image.

```
Bitmap bm = (Bitmap) data.getExtras().get("data");
iv.setImageBitmap(bm);

File file = new File(Directory, filename);
FileOutputStream fos;
try {
        fos = new FileOutputStream(file);
        bm.compress(CompressFormat.JPEG, 100, fos);}
```

The image is saved to the devices SD card, in a compressed JPEG format. This is done in order to limit the amount of data transfer needed when uploading the image to a webserver. As well as saving the image it is also displayed in an image view in the "Tag" tab.
When the user presses the 'send' button the image is encoded using a Base64 encoder.

```
Bitmap bm = BitmapFactory.decodeFile(picture.getAbsolutePath());

ByteArrayOutputStream stream = new ByteArrayOutputStream();
bm.compress(Bitmap.CompressFormat.JPEG, 100, stream);

byte[] byte_arr = stream.toByteArray();
image_str = Base64.encodeBytes(byte_arr);
```

This encoder turns the binary data of the file into a text string that is easy to transfer using the HTTP POST method. The string containing the image data as well as the device ID, timestamp, category, text and the coordinates is then sent using HTTP POST in the same way as was done with the user data.

Once the data has been sent all the values are reset and the image file is deleted.

```
file.delete();
iv.setImageResource(R.drawable.ic_launcher);
tekst.setText("");
latitude = null;
longitude = null;
```

## 5.2 Database design

In order to store the data collected by the mobile application a MySQL database is set up. The database contains 4 tables:

| | |
|---|---|
| **Respondents** | Stores information about the respondents |
| **GPSdata** | Stores the collected GPS points |
| **Trips** | Stores start/stop time of the trips and an encoded polyline with the relevant GPS positions. |
| **Tags** | Stores the submitted tags. |

The tables are linked as the Entity-Relationship diagram in Figure 25 shows. The common denominator that allows us to link data from one database table to another is the device id.

As well as the tables described in Figure 25 there is also a separate table called “heat”. This table stores the output of the square count analysis described in chapter 5.4.2. It is not described in the ER-diagram because it has no relation to the four other tables other than it is created based on the data from the “gpsdata” table. The table contains a series of polygons stored as an encoded polyline of its borders as well as a series of count fields.

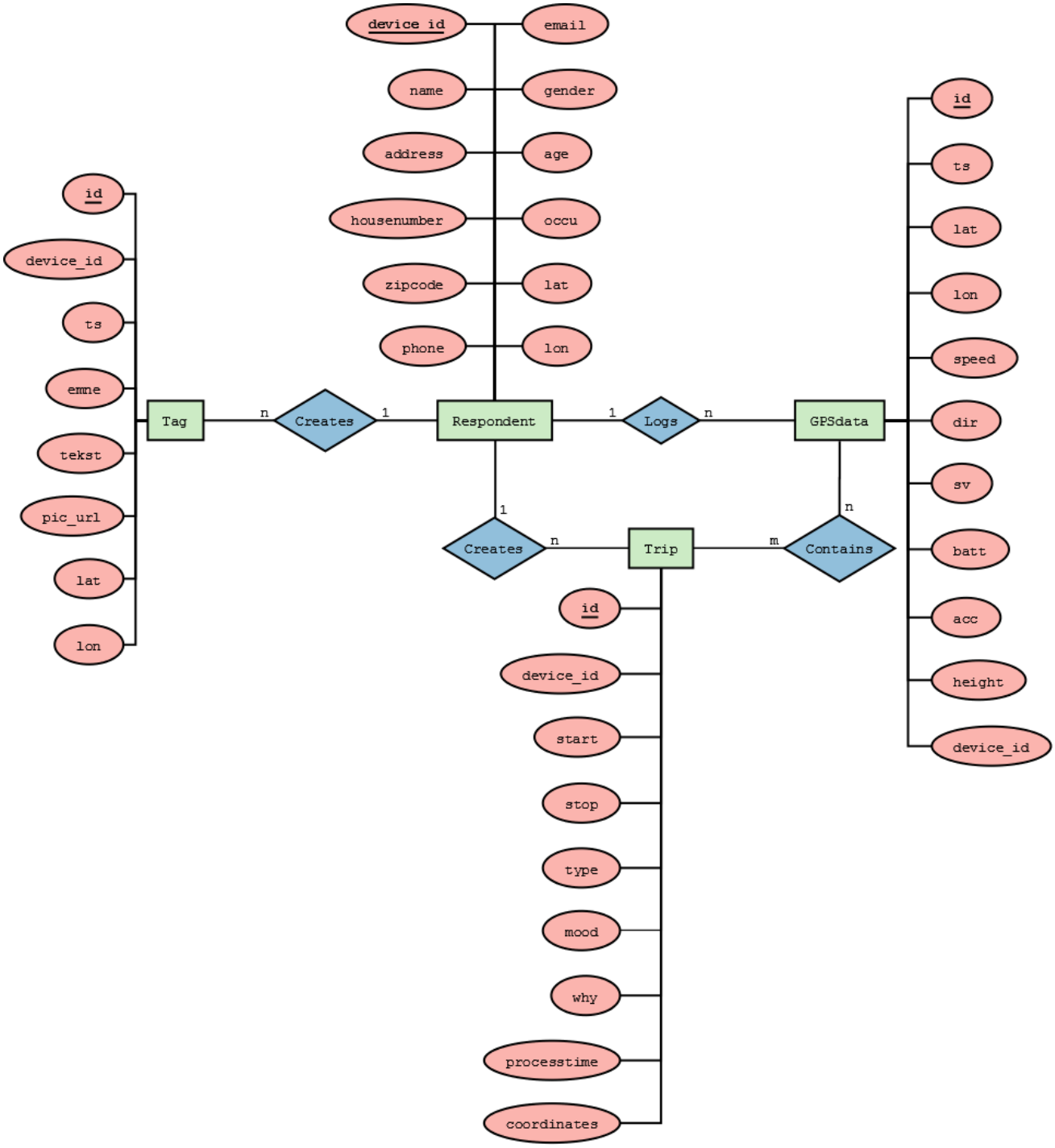



Figure 25 - ER-diagram

## 5.3 Data handling scripts

When data is submitted to the webserver from the application, a series of scripts are in place to handle the reception of this data and insert it into the correct tables in the database. This chapter will describe these scripts handling the user data, GPS data, trip data and tags on the webserver. All the scripts are created using PHP.

### 5.3.1 Insert/update user data in the respondents table

One script creates or updates the table handling information about the respondents. It is called by the mobile device and receives all the information through a HTTP-POST which is made into a variable in the beginning of the PHP script.

```
$device id = $ POST['device'];
```

Then an if/else statement decides if there is a device ID in the database table matching the one from the http post variable. Based on the result it either inserts or updates the database table.

```
$query_insert = sprintf("INSERT into respondents
(device id, name, address, housenumber, zipcode, phone, email, gender, age, occu, lat, lng)
VALUES('%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s')"

$query_update = sprintf("UPDATE respondents SET name='%s', address='%s', housenumber='%s',
zipcode='%s', phone='%s', email='%s', gender='%s', age='%s', occu='%s', lat='%s', lon='%s'
WHERE device_id='%s'"
```

In the end the PHP script echoes a number. 1 if it was an INSERT or 2 if it was an UPDATE which the mobile device gets as the response code an react accordingly as mentioned in chapter 5.1.3.

**Geocoding of addresses**

A set of coordinates for the respondents' home addresses will be needed in order to cloak them, in a publication of data. To geocode addresses Google's geocoding service is used. This enables us to parse information through a URL and in return get an XML with the corresponding coordinates, as seen below:

http://maps.google.com/maps/geo?output=xml&key=[APIKEY]&q=osterbro14,9000

When a respondent uploads user data the submitted address information is formatted and made into a variable.

```
$address = utf8_encode($row["address"]);
$housenumber = utf8_encode($row["housenumber"]);
$zipcode = utf8 encode($row["zipcode"]);
$full address = "".$address. " " .$housenumber. ", "  .$zipcode."";
```

This address is then used to make a request URL which is used in the PHP function "simplexml_load_file"

```
$request_url = $base_url . "&q=" . urlencode($full_address);
$xml = simplexml_load_file($request_url) or die("url not loading");
```

The response from the request contains the response code 200, the geocoding has been successful and the coordinates can be retrieved from the xml and added to the variables that are inserted into the respondents table in the database.

```
$status = $xml->Response->Status->code;
if (strcmp($status, "200") == 0) {
  // Successful geocode
 $geocode_pending = false;
 $coordinates = $xml->Response->Placemark->Point->coordinates;
 $coordinatesSplit = split(",", $coordinates);
 // Format: Longitude, Latitude, Altitude
 $lat = $coordinatesSplit[1];
 $lng = $coordinatesSplit[0];
```

If the geocoding was unsuccessful, for example if Google could not find the address, the script just moves on and no coordinates are inserted into the database.

### 5.3.2 Upload GPS data and trip data

To insert the GPS data in the database a PHP script is posted a file from the mobile device. The file is then stored in a temporary location and is opened.

```
$filename = $_FILES['uploadedfile']['tmp_name'];
$file = fopen($filename, "r") or exit("Unable to open file!");
```

This triggers a while-loop that will continue as long as there are lines in the file.

```
while(!feof($file)) {
} fclose($file);
```

It then picks a single line using “fgets” and explodes the line where there is a “;” while giving each values a PHP variable name.

```
$string = fgets($file);
list($ts, $lat, $lon, $speed, $dir, $sv, $batt, $acc, $height, $device_id) = explode(";", $string);
```

These variables are then inserted into the database and the script moves on to the next line in the file.

```
$query = sprintf("INSERT into gpsdata(id, ts, lat, lon, speed, dir, sv, batt, acc, height,
device id) VALUES(NULL, '%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s')"
```

Uploading trip data is handled in the same way as GPS data except it gets some other variables and inserts them into a different table.

```
$query = sprintf("INSERT into trip(id, device id, start, stop, type, mood, why) VALUES(NULL, '%s',
'%s', '%s', '%s', '%s', '%s', '%s')"
```

### 5.3.3 Upload image

To receive the data uploaded from the applications tag function a PHP script gets all data posted to it from the mobile device. Based on these data a picture URL is created. An if/else statement checks if an image is present and if not renaming the picture URL to “intet billede”.

```
$base=$_POST['image'];
…

$pic_url="pictures/".$device."".$timestamp.".jpg";
if ($base !="intet billede") {
else {
$pic url = $base;
}
```

If a picture is present it decodes the base64 encoded data string and creates the picture file in the location of the picture URL.

```
header('Content-Type: bitmap; charset=utf-8');
$binary=base64 decode($base);
$file = fopen('pictures/'.$device.''.$timestamp.'.jpg', 'wb');
fwrite($file, $binary);
fclose($file);
```

Afterwards the information is inserted into the tags table through a SQL query.

```
$query = sprintf("INSERT into tags(id, deviceid, ts, emne, tekst, pic url, lat, lon) VALUES(NULL,
'%s', '%s', '%s', '%s', '%s', '%s', '%s')",
```

## 5.4 Spatial Analysis

The collected data in its raw form would be almost impossible to make sense of. In order to comprehend and interpret the GPS data we therefore need to do some processing. For this implementation three different spatial functions were created.

**Polyline creation** Joining the coordinates in between the start and stop time of the submitted trips to form polylines allows for visualization of the movement patterns.

**Square count** By creating a grid and counting the GPS positions within each cell a hot spot analysis can be created highlighting the most popular areas.

**Time span** The time span analysis allows for a visualization of the data over time, enabling an understanding of the movement patterns change during the day.

The technical implementation of these functions will be described in the following chapters.

### 5.4.1 Polyline creation based on trips

For the tracks to be viewed, quickly and smoothly they are encoded as polylines. The encoding is done inside the same script that receives the uploaded trips from the mobile phones. To encode the polylines a PHP script created by Jim Hribar is included in our script (Hribar n.d.).

```
include("PolylineEncoder.php");
```

The script selects the coordinates of the home address in the respondents table using the device ID of the uploaded trip.

```
$sql1 = "SELECT lat, lng FROM respondents WHERE device_id=".$device_id."";
```

The start/stop time from the uploaded trip GPS data that is within the time limits.

```
$sql2 = "SELECT lat, lon FROM gpsdata WHERE device_id=".$device_id." AND ts>'".$starttime."' AND
ts<'".$stoptime.";";
```

The coordinate set is then pushed into an array called “$full_points”, to decide if the coordinate set is within a 50 meters radius of the respondent’s home, and therefore should be excluded, the Pythagorean Theorem is used to calculate the distance between the logged position and home coordinates of the respondents. An if-sentence therefore push the point to the points array if this parameter is met.

```
$delta_N = ($lat - $respondent_lat)*$latlen;
$delta_E = ($lng - $respondent_lng)*$lonlen;
$distSq = sqrt($delta_E*$delta_E + $delta_N*$delta_N);
if ($distSq > 50) {
    array_push($points, array($lat,$lng));
}
```

The arrays are afterwards encoded using the included PHP-script.

```
$polylineEncoder = new PolylineEncoder($points);
$encPointsFull = $polylineEncoder->dpEncode();
$encPointsFull = addslashes($encPoints['Points']);
```

The encoded points are then inserted into the trip table using an SQL update query.

```
$sql3 = "UPDATE trip SET processtime=NOW(), coordinates='".$encPoints."'
WHERE id='".$track_id."';";

mysql_query($sql3);
```

### 5.4.2 Square count

The intensity of points can be visualized by counting the number of points inside squares in a grid. For this implementation the grid will consist of 100 by 100 meter squares. In order to accomplish this we will need to create grid based on polygons which covers the entire area of the data set.

So first the bounding box is calculated using a SQL query.

```
$query = "SELECT max(lat), min(lat), max(lon), min(lon), avg(lat) FROM gpsdata";
```

Then the size of 1 degree on the latitude and longitude scale is calculated.

```
$query = "SELECT max(lat), min(lat), max(lon), min(lon), avg(lat) FROM gpsdata_demo";
// Calculate the length of a degree of latitude and longitude in meters
$m1 = 111132.92;        // latitude calculation term 1
$m2 = -559.82;          // latitude calculation term 2
$m3 = 1.175;            // latitude calculation term 3
$m4 = -0.0023;          // latitude calculation term 4

$p1 = 111412.84;        // longitude calculation term 1
$p2 = -93.5;            // longitude calculation term 2
$p3 = 0.118;            // longitude calculation term 3
$latlen = round($m1 + ($m2 * cos(deg2rad(2 * $latavg))) + ($m3 * cos(deg2rad(4 * $latavg))) + ($m4 *
cos(deg2rad(6 * $latavg))),2);
$lonlen = round(($p1 * cos(deg2rad($latavg))) + ($p2 * cos(deg2rad(3 * $latavg))) + ($p3 *
cos(deg2rad(5 * $latavg))),2);
```

By using this number the amount of iterations needed on latitude and longitude is calculated.

```
$i_latmax = Round((($latmax-$latmin)*$latlen)/$polysize);
$i_lonmax = Round((($lonmax-$lonmin)*$lonlen)/$polysize);
```

The grid will be created starting in the lower left corner - (on this side of the earth).

```
$lat = $latmin;
$lon = $lonmin;
```

We run the calculated number of iterations first on the latitude scale and then the longitude scale through a for loop.

```
for ($i_lat = 1; $i_lat < $i_latmax; $i_lat = $i_lat + 1) {
    $latsize = (1/$latlen)*$polysize;
    $latplus = $latsize+$lat;

    for ($i_lon = 1; $i_lon < $i_lonmax; $i_lon = $i_lon + 1) {
        $lonsize = (1/$lonlen)*$polysize;
        $lonplus = $lonsize+$lon;
```

The bounding box of the single polygon is then encoded using the polyline encoder script described earlier in chapter 5.4.1.

```
for ($i_lon = 1; $i_lon < $i_lonmax; $i_lon = $i_lon + 1) {
    $lonsize = (1/$lonlen)*$polysize;
    $lonplus = $lonsize+$lon;

    $poly_points = array();
    array_push($poly_points, array($lat,$lon));
    array_push($poly_points, array($lat,$lonplus));
    array_push($poly_points, array($latplus,$lonplus));
    array_push($poly_points, array($latplus,$lon));

    $polylineEncoder = new PolylineEncoder($poly_points);
    $encPoints = $polylineEncoder->dpEncode();
    $encPoints = addslashes($encPoints['Points']);
```

Then the number of points inside this polygon is calculated seven times: all day and in timespans containing three hour intervals.

```
$sql = "SELECT count(id) FROM particimap.gpsdata demo WHERE lat>".$lat." AND lat<".$latplus."
AND lon>".$lon." AND lon<".$lonplus." AND FROM UNIXTIME(ts,'%H')>=0 AND FROM UNIXTIME(ts,'%H')<=3;";
$result = mysql_query($sql);

while ($row = mysql_fetch_array($result)) {
       $count1 = $row["count(id)"];
       }
```

The values are afterwards inserted into the heat table.

```
$sql = "INSERT INTO heat_demo
(coordinates, count, count1, count2, count3, count4, count5, count6)
VALUES ('".$encPoints."','".$count."','".$count1."','".$count2."','".$count3."','".$count4."',
'".$count5."','".$count6."');";

$result = mysql_query($sql);
```

Then “$lonplus” is renamed as “$lon”. This continues until the iterations on the longitude scale are finished and then “$latplus” is renamed “$lat” and the process continues until the iterations on the latitude scale are finished.

### 5.4.3 Timemap

To give a more precise picture on how the movement is over time and which areas and routes are used at different times of the day a “timemap” is integrated in the web platform. Timemap (Figure 26) is a mix of SIMILIE timeline API and the mapstraction map API and consist of a timeline in the top and a map view below, where the timeline events are shown (Google Code n.d.) (SIMILE Widgets n.d.) (Mapstraction n.d.).

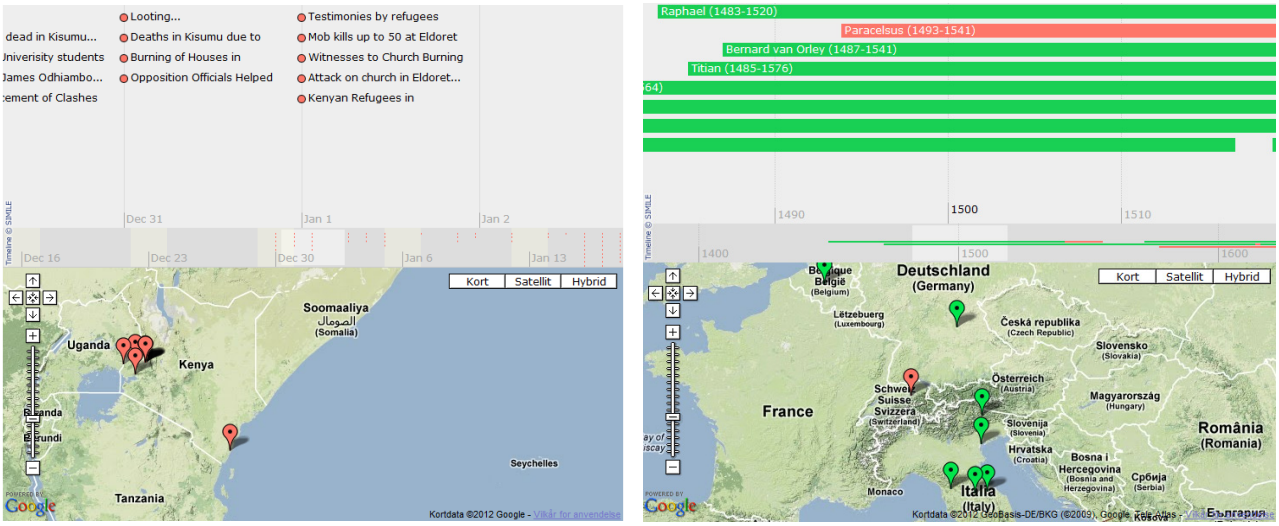



Figure 26 - Eksamples of the Timemap (Google Code n.d.)

Our first integration used PHP to generate the JSON data needed to show content on the view. However as the amount of data grow; a shift to KML rendering was made (OGC 2008).

Furthermore positions closer than 10 meters to the next position will be removed from the table. This is done by calculating the distance between two adjacent positions and through an if-statement removing it if the distance to the last one is less than 10 meters.

```
$sql4 = "SELECT id, lat, lon FROM gpsdata WHERE device_id='".$device_id."' ORDER BY ts";
$result4 = mysql_query($sql4);

            while ($row = mysql_fetch_array($result)) {

                $id = $row4["id"];
                $lat = $row4["lat"];
                $lng = $row4["lon"];

                $delta_N = ($lat - $last_lat)*111349.63;
                $delta_E = ($lng - $last_lng)*61693.45;
                $distSq = $delta_E*$delta_E + $delta_N*$delta_N;
                if ($distSq < 100) {        // ca. 10 meter radius

                $sql5 = "DELETE FROM gpsdata WHERE id='".$id."';";
                $result5 =mysql_query($sql5);

                }
                else {
                $last_lat = $lat;
                $last_lng = $lng;
            }
              }
```

To generate KML files from the “gpsdata” table in our database the time and coordinates had to be formatted. Time has to be formatted like this: 2012-05-08T07:23:23Z and the coordinates has to be formatted like so: 9.42034132,56.36387577,0.

The query to the database is there looks like this.

```
SELECT from_unixtime(ts,'%Y-%m-%dT%H:%i:%SZ') AS KML_time,
group concat(lon,',',lat,',','0' order by ts DESC separator ' ') AS coordinates,
device id, id FROM gpsdata
```

To minimize data and gain a more smooth experience each KML is sorted to only contain data from one day.

```
 WHERE from_unixtime(ts,'%d')=10
```

Creation of the KML will be done by writing to a file named after which date it represents. The processing will be done with a PHP-script which first adds the header content to the file:

```
$filename = "time10.kml";
$fh = fopen($myFile, 'a') or die("can't open file1");

$kml = '<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.1"><Document>';
```

Then a while loop will create a place marks for each position.

```
 $query = "SELECT from unixtime(ts,'%Y-%m-%dT%H:%i:%SZ') AS KML time,
group concat(lon,',',lat,',','0' order by ts DESC separator ' ') AS coordinates,
device id, id FROM gpsdata";

 $result = mysql_query($query);

while ($row = @mysql_fetch_assoc($result))
{

      $fh = fopen($filename, 'a') or die("can't open file2");

      $kml = '<Placemark><Point><coordinates>' . $row['coordinates'] . '</coordinates></Point>
      <name> </name><TimeStamp><when>'.$row['KML time'].'</when></TimeStamp>
      <description>' . $row['device_id'] . '</description><ExtendedData><Data name="theme">
      <value>' . $row['device_id'] . '</value></Data></ExtendedData></Placemark>';

      fwrite($fh, $kml);
      fclose($fh);

}
```

And at last the ending tags and the KML are complete.

```
$fh = fopen($myFile, 'a') or die("can't open file");
$kml = '</Document></kml>';

fwrite($fh, $kml);
fclose($fh);
```

Now that a KML file is created the code for displaying it in the Timemap must be written. Timemap works as a JavaScript library which is accessed through JavaScript from the html file. First it has to be initialized with declaration of the map, timeline IDs and the location of the folder containing images used on the Timemap.

```
$(function() {

    tm = TimeMap.init({
        mapId: "map",               // Id of map div element (required)
        timelineId: "timeline",     // Id of timeline div element (required)
        options: {
            eventIconPath: "images/"
        },
```

Then the dataset is loaded. It has a title, default theme besides its type. Inside the KML additional tags called “extendedData” were made. These allow us to make the Timemap API look for more data which in this case is the theme information.

```
        datasets: [
            {
                title: "TimeMap Viborg",
                theme: "red",
                type: "kml",     // Data to be loaded in KML - must be a local URL
                options: {
                    url: "time8.kml", // KML file to load
                    extendedData: ["theme"]
                }
            }
        ],
```

The theme is equal to the device_id so custom TimeMapThemes is added inside the Timemap JavaScript library. Each theme has a color value as well as a custom dot and marker icon associated with it as shown below.

```
        359778040691659: new TimeMapTheme({
        icon: "images/359778040691659-dot.png",
        color: "#E41A1C",
        eventIconImage: "359778040691659-circle.png"
```

At last the information about the two top bands are added which is set to take up 50% of the width each, to represent minutes and hours.

```
        bandInfo: [
            {
               width:          "50%",
               intervalUnit:   Timeline.DateTime.MINUTE,
               intervalPixels: 500

            },
            {
               width:          "50%",
               intervalUnit:   Timeline.DateTime.HOUR,
               intervalPixels: 250,
               showEventText:  false,
               trackHeight:    0.1,
               trackGap:       0.1
            }
        ]
    });
});
```

The end result is as shown in Figure 27. The Timemap allows a user to view positions within the timespan on the top, which is 3,5 minutes.

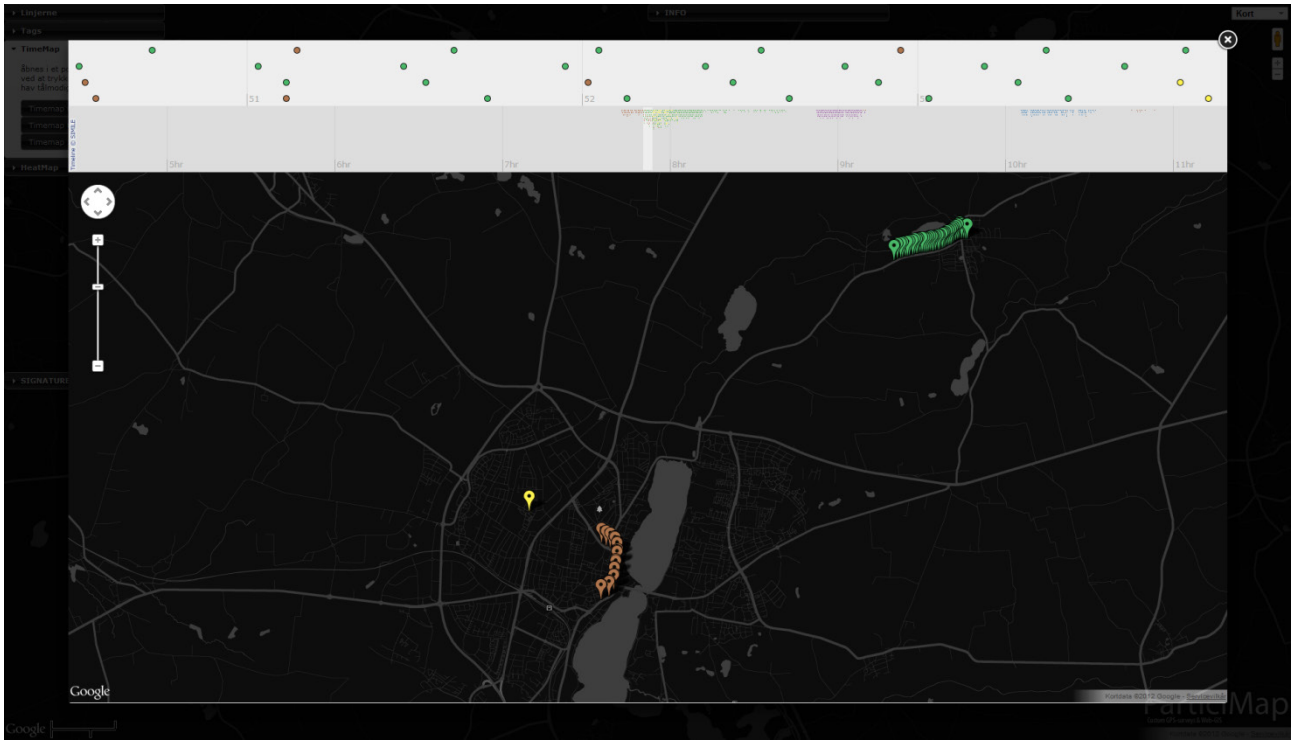



Figure 27 - Timemap implementation

Furthermore the timelines can be scrolled to each side allowing the user to view the data as an animation giving information about speed and direction of the respondents, as illustrated in Figure 28.

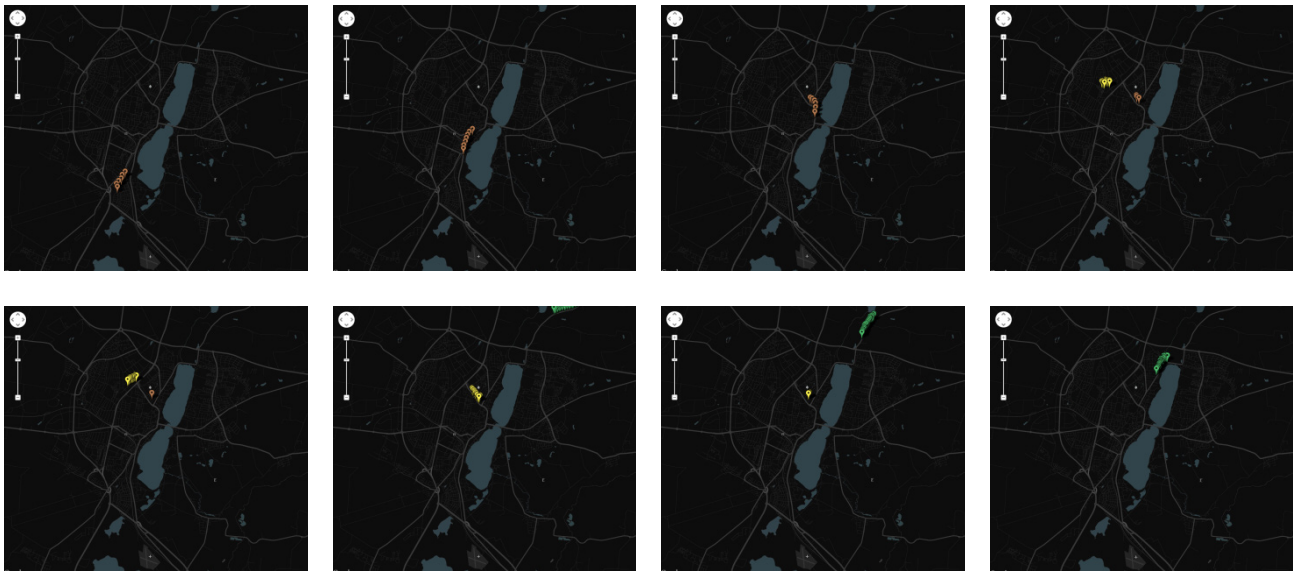


Figure 28 - Scrolling the Timemap

## 5.5 Web map platform

In order to be able to view collected data as it is collected, a web map platform was created. The platform is based on a styled map from Google where the data canbe displayed on. The data is extracted from the database as XML data and manipulated and handled using JavaScript. Using JavaScript it was possible to create a platform that performs very smoothly and features the possibilities of turning on and off different layers, and styling them according to different attributes. All the elements that go into creating the visualization platform are seen in Figure 29 and will be described in more detail in the following paragraphs. The result of this implementation can be seen at http://viborg.particimap.dk.

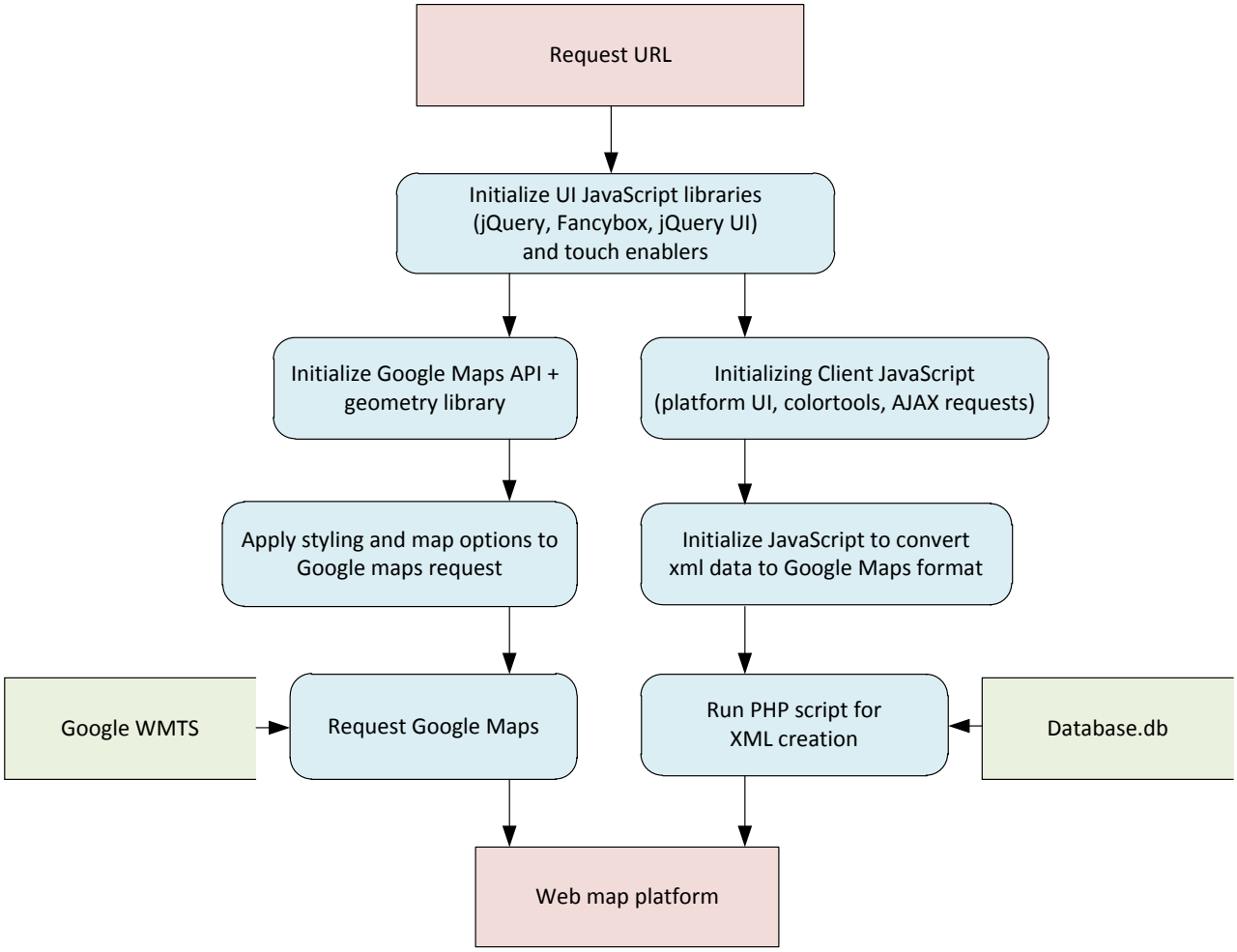



Figure 29 - Loading the web map platform

### 5.5.1 XML generation

To extract data from the database PHP will be used to generate an XML document to contain all the needed data. The xml file is encoded as ISO-8859-1 to accommodate Danish letters (W3schools u.d.).

The XML document has to contain data about Respondents, Tags and Trips. This is achieved by using a set of SQL queries. Information retrieved about the respondents is device ID, name, gender and age.

```
SELECT device_id, gender, age FROM respondents;
```

Information about the tags are device ID, subject, text, URL for the picture and the coordinates.

```
SELECT deviceid, tekst, lat, lon, pic url, emne FROM tags;
```

As mentioned earlier the time information is stored as an UNIX timestamp. Therefore it has to be converted to an understandable date format using the SQL functions FROM_UNIXTIME:

```
FROM_UNIXTIME(timestamp, '%e-%m-%Y %T')->   02-04-2012 17:18:45
```

The SQL to retrieve the trip data is then as follows:

```
SELECT device id, FROM UNIXTIME(start, '%e-%m-%Y %T') AS start, FROM UNIXTIME(stop, '%e-%m-%Y %T')
AS stop, type, mood, coordinates FROM trip WHERE coordinates != '';
```

When creating xml the simplest way is to use the PHP function *echo* to output the data collected through the SQL calls:

```
echo '    <respondentId>'.$row1['device_id'].'</respondentId>'.PHP_EOL;
```

Using the echo-method you have to write all the code for yourself and have full control of the outcome of the end document. This is however more vulnerable to human error then if you use Document Object Model (DOM) which is featured through PHP 5's integrated DOM libraries. DOM is used to create XML by creating elements and appending them as children to its parent:

```
$deviceNode = $dom->createElement('respondentId', $row1['device id']);
$respondentsNode->appendChild($coorNode);
```

This solution is much easier to use when manipulating with exciting documents but lack the simple overview echo gives and is therefore not chosen.

The XML could either be stored as a file on the webserver, or be generated each time the platform loads. The latter is chosen for this implementation since it will ensure that the data being viewed is always up to date. The end xml looks as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<TrackingData>
  <Respondent>
    <respondentId>358967047218191</respondentId>
    <gender>Mand</gender>
    <age>26</age>
  </Respondent>
  <Track>
    <respondentId>358967047218191</respondentId>
    <starttime>4-04-2012 10:33:17</starttime>
    <endtime>4-04-2012 10:51:54</endtime>
    <type>gå</type>
     <mood>positiv</mood>
     <coordinates>yge{I_op{@F{A@w@NaBEk@UWGYSWK_@EFECE@`@v@Jh@?f@]xBFKEDN|@In@J^X\JZaDpA</coordinate
     s>
  </Track>
  <Note>
    <respondentId>358967047218191</respondentId>
    <subject>Raporter et problem</subject>
    <text>grafitti kunst, øv!</text>
    <pic_url>pictures/358967047218191133277318588 3.jpg</pic_url>
    <latitude>57.04397192</latitude>
    <longitude>9.92141238</longitude>
  </Note>
</TrackingData>
```

### 5.5.2 Base Map

Now that the data has been extracted from the database it needs a canvas to be displayed on. In this project Google Maps API has been chosen as the platform to facilitate this. The basic map view is very colorful and will therefore interfere with the visualized data. The map layer is therefore styled in greyscale so it won't hinder data interpretation. Likewise the map controls are simplified and rearranged. In addition to the standard Google Maps tools a new set of tools are created because the map has to facilitate different kinds of data: points, lines and polygons, which requires a tool to select which data the user wants. To do this the Accordion JavaScript in jQuery UI is used. This allows a collapsible menu system so unnecessary information can be hid away and accordions are therefore used to hide away additional information like instructions and legends. The accordions on the page are as an addition made drag able so the user can rearrange the view how he sees fit. In Figure 30 the Google Maps platform is seen before and after the styling and customization.


Figure 30 - Google Maps before and after costumization

### 5.5.3 Creating map features from XML data

The First step in transforming the data from XML data to map features is to extract the data from the xml file via an Ajax Request. The response from this request is the named as the variable xml.

```
var url = 'http://aod-dus.cs.aau.dk/particimap/demo/src/get.php';

sendAjaxRequest('GET', url, function(xmlhttp) {
var xml = xmlhttp.responseXML;
```

Then the different data tags (Respondents, Track, Note and Polygon) is parsed by first deciding how many tags of a given type there are, and then run an external function based upon the type for every tag.

```
var xTracks = xml.getElementsByTagName("Track");
TRACKING_CLIENT.tracks = new Array(xTracks.length);
for (var i = 0; i < xTracks.length; i++) {
    TRACKING_CLIENT.tracks[i] = new Track(TRACKING_CLIENT.map, xTracks[i]);
}
```

The goals of the functions are to format it according to the Google Maps API standard. First it makes a variable from the tag elements inside the tag parsed to the function.

```
this.respondentId = xTrack.getElementsByTagName("respondentId")[0].childNodes[0].nodeValue;
```

#### Polylines

Polylines are used to display the trips. They are created according to the Google Maps standard and the path is created by decoding the encoded coordinate string from the trips database.

```
this.polyline = new google.maps.Polyline({
    path: google.maps.geometry.encoding.decodePath(this.coordinates),
    strokeColor: "#FF0000",
    strokeOpacity: 0.8,
    strokeWeight: 5
});
```

When the polylines are clicked a popup window is shown.

```
google.maps.event.addListener(this.polyline, 'click', function(event) {
    TRACKING_CLIENT.showInfoWindow(this.description,event.latLng);
});
```

This is defined to show age, start time, end time, type and mood of the trip.

```
this.polyline.description = '<p style="color: #999999">' + this.respondent.age +
    '-årig ' + this.respondent.gender.toLowerCase() + '.</p>' +
    '<table border="0">' +
    '<tr><td>Tur påbegyndt:</td><td>' + this.starttime + '</td></tr>' +
    '<tr><td>Tur afsluttet:</td><td>' + this.endtime + '</td></tr>' +
    '<tr><td>Transporttype:</td><td>' + this.type + '</td></tr>' +
    '<tr><td>Følelsen omkring turen:</td><td>' + this.mood + '</td></tr>' +
    '</table>';
```

A set of support functions are created to show/hide the layer and color it according to a variable called color.

```
this.show = function() {
    this.polyline.setMap(this.map);
};
this.hide = function() {
    this.polyline.setMap(null);
};
this.setColor = function(color) {
    this.polyline.setOptions({
        strokeColor: color,
        strokeOpacity: 0.8,
        strokeWeight: 5
    });
}
```

**Polygons**

The heat map is created as polygons where the bounding boxes are made up of encoded polylines. These are decoded and used as the path.

```
this.polygon = new google.maps.Polygon({
    path: google.maps.geometry.encoding.decodePath(this.coordinates),
    strokeColor: "#FF0000",
    strokeOpacity: 0.8,
    strokeWeight: 0.1,
    fillColor: "#FF0000",
    fillOpacity: 0.5
});
```

A popup window is then defined to show the counts for the different time intervals.

```
this.polygon.description =
'<p style="color: #999999">' + this.id + ',  ialt: ' + this.count + '.</p>' +
'<table border="0">' +
'<tr><td>Fra kl. 00-04:</td><td>' + this.count1 + '</td></tr>' +
'<tr><td>Fra kl. 04-08:</td><td>' + this.count2 + '</td></tr>' +
'<tr><td>Fra kl. 08-12:</td><td>' + this.count3 + '</td></tr>' +
'<tr><td>Fra kl. 12-16:</td><td>' + this.count4 + '</td></tr>' +
'<tr><td>Fra kl. 16-20:</td><td>' + this.count5 + '</td></tr>' +
'<tr><td>Fra kl. 20-24:</td><td>' + this.count6 + '</td></tr>' +
'</table>';
```

And a click listener is added to show the popup when a polygon is clicked.

```
google.maps.event.addListener(this.polygon, 'click', function(event) {
    TRACKING_CLIENT.showInfoWindow(this.description,event.latLng);
});
```

Again a set of support functions are created to support hide/show of polygons and color them according to a variable called color.

```
this.show = function() {
    this.polygon.setMap(this.map);
};
this.hide = function() {
    this.polygon.setMap(null);
};
this.setColor = function(color) {
    this.polygon.setOptions({
  strokeColor: "#FF0000",
  strokeOpacity: 0.8,
  strokeWeight: 0.1,
  fillColor: color,
  fillOpacity: 0.5
    });
}
```

**Markers**

The tags submitted to the database are visualized as markers.

```
this.marker = new google.maps.Marker({
    position: new google.maps.LatLng(this.latitude, this.longitude),
    title: this.subject,
    icon: pickicon(this.subject)
});
```

Because it is not mandatory to send a picture when a tag is created, an if/else sentence is created so two different layouts for the popup window can be created.

```
if (this.pic_url != 'ingen billede') {
    this.marker.description =
            '<b>' + this.subject + '</b>' +
            '<p>' + this.text + '</p>' +
            '<img border="0" src="/particimap/demo/apphandler/' + this.pic_url + '" width="200" />';
    }
else {
    this.marker.description =
            '<b>' + this.subject + '</b>' +
            '<p>' + this.text + '</p>';
}
```

And a listener is added to show the popup when the marker is clicked.

```
google.maps.event.addListener(this.marker, 'click', function() {
    TRACKING_CLIENT.showInfoWindow(this.description, this.getPosition());
});
```

The markers also have support functions to show/hide them.

```
this.show = function() {
    this.marker.setMap(this.map);
};
this.hide = function() {
    this.marker.setMap(null);
};
```

The markers do not have to be colored according to a variable; however they need a different icon based on the type. A function is therefore created to pick the icon through a switch statement.

```
function pickicon(category) {
   var icon = "img/icon.png";
   switch(category) {
     case "Syntes godt om!": icon = "img/smiley_happy.png";
                break;
     case "Foreslå forbedring":    icon = "img/smiley_neutral.png";
                break;
     case "Raporter et problem":    icon = "img/smiley_sad.png";
                break;
   }
   return icon;
}
```

To control which category of the shown a set of checkboxes are created in an accordion.

```
<form action="#">
<input type="checkbox" id="Syntes godt om!box" onclick="boxclick(this,'Syntes godt om!')" />
<label for="Syntes godt om!box">Syntes godt om!</label>
<input type="checkbox" id="Foreslå forbedringbox" onclick="boxclick(this,'ForeslÃ¥ forbedring')"
/><label for="Foreslå forbedringbox">ForeslÃ¥ forbedring</label>
<input type="checkbox" id="Raporter et problembox" onclick="boxclick(this,'Raporter et problem')"
/><label for="Raporter et problembox">Rapporter fejl</label>
</form>
```

This allows the user to control hide/show of the different types of markers. To facilitate this there is a box-click function which is set as the click response. This function checks through an if/else sentence if the corresponding checkbox is checked or not and redirect the call to the right function either show(category) or hide(category)

```
function boxclick(box,category) {
    if (box.checked) {
        show(category);
    } else {
        hide(category);
    }
}
```

These functions go through “for” loops picking the corresponding markers and either hiding or showing them and afterward set the checkbox to the right stance.

```
function show(category) {
    for (var i=0; i<TRACKING_CLIENT.notes.length; i++) {
        if (TRACKING_CLIENT.notes[i].subject == category) {
            TRACKING_CLIENT.notes[i].show();
        }
    }
    document.getElementById(category+"box").checked = true;
}

function hide(category) {
    for (var i=0; i<TRACKING_CLIENT.notes.length; i++) {
        if (TRACKING_CLIENT.notes[i].subject == category) {
            TRACKING_CLIENT.notes[i].hide();
        }
    }
    document.getElementById(category+"box").checked = false;
    TRACKING_CLIENT.hideInfoWindow();
}
```

**Changing colors**

To set the color of the different data visualizations a number of buttons are created in accordions depending on data type.

```
<input id="colorByTrackButton" type="button" title="Respondenternes id" value="Respondenternes id"/>
```

A JavaScript then picks the element ID and sets an on click function.

```
this.colorByTrackButton = document.getElementById('colorByTrackButton');
this.colorByTrackButton.onclick = function() {
```

To color based on respondent ID, a function called “generateColors” generates a HEX color code based on the number of “respondents” tag elements the xml holds.

```
    this.colorByTrackButton = document.getElementById('colorByTrackButton');
    this.colorByTrackButton.onclick = function() {
    var colors = TRACKING_CLIENT.generateColors(respondents.length);
    for (var i = 0; i < tracks.length; i++) {
        tracks[i].show();
        for (var j = 0; j < respondents.length; j++) {
                if (tracks[i].respondentId == respondents[j].respondentId) {
                    tracks[i].setColor(colors[j]);
                    break;
                }
        }
    }
}
```

To display the gender of the respondents, the JavaScript checks if the respondent is equal to “Mand” and gives it a blue color, or else gives it a pink color.

```
for (var i = 0; i < tracks.length; i++) {
tracks[i].show();
    if (tracks[i].respondent.gender == 'Mand') {
        tracks[i].setColor('#0000FF');
    }
    else {
        tracks[i].setColor('#FF00FF');
    }
}
```

To color the tracks based on the age of the respondents, an array with five colors are created. Afterwards the respondents age is divided with 20 rounding downwards to get a digit between one and five (assuming the max age is 100 years). The track is then given the coresponding color.

```
var colors = new Array("#00FF00","#00FF80","#00FFFF","#0080FF","#0000FF");
for (var i = 0; i < tracks.length; i++) {
    tracks[i].show();
    tracks[i].setColor(colors[Math.floor(tracks[i].respondent.age / 20)]);
}
```

To color the tracks based on type and mood, a switch statement is used to switch between the different variables.

```
for (var i = 0; i < tracks.length; i++) {
tracks[i].show();
var Mood = tracks[i].mood;
    switch (Mood) {
    case "positiv":
        tracks[i].setColor('#0000FF');
        break;
    case "neutral":
        tracks[i].setColor('#FF00FF');
        break;
    case "negativ":
        tracks[i].setColor('#FF0000');
        break;
    }
```

The polygons are likewise colored using a switch statement to enable coloring based on the count of points inside the polygon. The problem here is that the limits are not dynamic and then has to be changed manually based on how many point one might predict there to be in each polygon or changed after the respondents are finished collecting data.

```
for (var i = 0; i < polygon.length; i++) {
polygon[i].show();
var Count = polygon[i].count1;
    switch (true) {
    case Count < 1:
        polygon[i].hide();
        break;
    case Count >= 1 && Count <= 30:
        polygon[i].setColor('#100000');
        break;
    case Count >= 30 && Count <= 100:
        polygon[i].setColor('#400000');
        break;
    case Count >= 100 && Count <= 1000:
        polygon[i].setColor('#C00000');
        break;
    case Count >= 1000 && Count <= 4000:
        polygon[i].setColor('#E00000');
        break;
    }
}
```

To hide the layers additional buttons are created using a JavaScript to run through the tracks or polygons and removing them.

```
this.trackoffButton = document.getElementById('trackoffButton');
this.trackoffButton.onclick = function() {
    var tracks = TRACKING_CLIENT.tracks;
    for (var i = 0; i < tracks.length; i++) {
    tracks[i].hide();
    }
}

this.heatoffButton = document.getElementById('heatoffButton');
this.heatoffButton.onclick = function() {
    var polygon = TRACKING_CLIENT.polygon;
    for (var i = 0; i < polygon.length; i++) {
    polygon[i].hide();
    }
}
```

# 6 Testing

Having created an application and a visualization platform that works in the hands of the developers is one thing, developing a system that works in the real world is something else entirely. The developers' detailed knowledge of every aspect of the coding may cause them to act and see things differently than a first time user. Therefore it is very important to have an application such as this tested in the real world using real people, so any issues can be addressed before deployment. In our case the test will be conducted using students from Viborg Katedralskole.

The test of the application will be used to evaluate the following:

| | |
|---|---|
| **Initial contact** | Is the method sufficient to gain the necessary number of respondents? How much information is necessary to give the respondents for them to have an adequate understanding of the application? |
| **Initializing the application** | Are the respondents able to install the application? Are they able to submit user data as intended? |
| **Start/stop trips** | Are the respondents' able to start and stop trips? Do they remember to start and stop trips? |
| **GPS data** | Do we receive data? What is the quality of the data? |
| **Tags and map** | Do the respondents use the tag function? Is the in-application map something the respondents use? |
| **Battery usage** | What is the average battery usage when using the application? How long is it possible to track in between charges? |
| **Visualizations Platform** | How does the visualization platform perform with the increased amount of data? Do the functions perform as intended? |

The testing will be conducted in two iterations. This is done to allow us to resolve issues in between the two tests and thereby being able to test the fixes as well as possibly discover more issues that should be resolved in future development.

## 6.1 Test one

The first test began with the project group holding a short lecture at a morning assembly at Viborg Katedralskole, where the subject of mobility research was briefly explained and an introduction to the test was given (Figure 31). After the assembly the students could then come and talk to us if they wished to participate. The volunteers would then be given a postcard (Figure 32) with some basic information on how the application worked and what was needed from them in order to participate. The full size postcard can be seen in Appendix E.

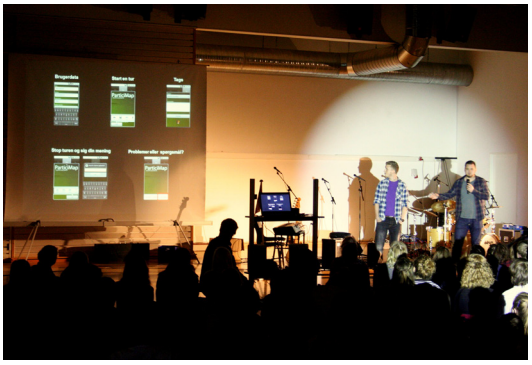


Figure 31 - Morning assembly speech

They would then be able to install the application and participate in the test for 48 hours. After the test period was over they were sent an online questionnaire that they would have to complete in order to be eligible to receive a prize for participating. The full questionnaire and the submitted answers can be seen in Appendix F.


Figure 32 - Postcard handed out to potential respondents

### 6.1.1 Initial contact

The postcards were handed out to potential respondents who approached us in the corridor immediately after the morning assembly. This did not prove that effective and only attracted about a handful of people. Due to the lack of initial interest another tactic was tried out. During the lunch break we walked around the cantina and courtyard asking everybody if they had an Android phone and if so if they would be interested in participating and handing them a postcard. This proved more effective and a total of about 40-50 postcards were handled out. One major issue that was discovered during this process was that most students had an Apple iPhone or non-smartphone prohibiting them from taking part in the test. The answers given in the questionnaire suggest that the information on the postcard should have been more extensive, which may also have improved some of the issues described in the later paragraphs (Figure 33).


Figure 33- Adequacy of postcard

### 6.1.2 Initializing the application

Twelve Respondents made an entry to our respondents table, seven females and five males. However, the server at the University went offline for unknown reasons from around 4pm and until the next morning so we do not know if more tried to sign up. The twelve respondents that did manage to submit userdata are seen in Table 2.

| Respondent | Gender | Age |
|---|---|---|
| **1** | m | 18 |
| **2** | m | 17 |
| **3** | f | 18 |
| **4** | f | 18 |
| **5** | f | 18 |
| **6** | m | 18 |

| Respondent | Gender | Age |
|---|---|---|
| **7** | m | 16 |
| **8** | f | 17 |
| **9** | f | 19 |
| **10** | f | 19 |
| **11** | m | 17 |
| **12** | f | 16 |

Table 2 - List of respondents

Unfortunately respondent 5 had a phone running Android 1.6 (HTC Magic) which does not allow for the battery level, SMS and xml-based on click listeners in our application to work. Therefore she could not participate. Also it turned out that respondent 7 only wanted to check out the application, and did not want to participate in the actual test. Therefore the number of respondents participating actively was reduced to ten. Nine out of these filled out the questionnaire at the end of the test. According to the answers given by the test group the installations were not much of an issue, except for one person who seemingly experienced some hassle (Figure 34).

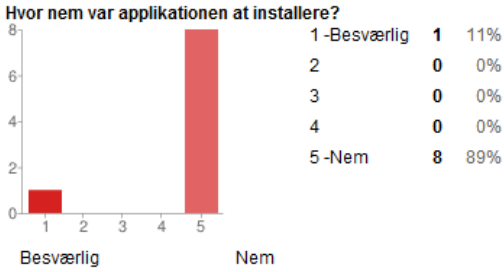



Figure 34 - Ease of installation

### 6.1.3 GPS data

During the test a total of 23,747 positions were sent to the database. The number of positions per respondent can be seen in Table 3.

| Respondent | # of positions | Respondent | # of positions |
|---|---|---|---|
| **1** | 9612 | **7** | 0 |
| **2** | 1901 | **8** | 0 |
| **3** | 0 | **9** | 6342 |
| **4** | 0 | **10** | 0 |
| **5** | 0 | **11** | 141 |
| **6** | 3202 | **12** | 2549 |

Table 3 - Number of submitted positions per respondent

As seen in Table 3 only six out of ten respondents had successfully submitted GPS data. One of the main reasons for this was believed to be that the respondents did not share the same understanding of the term GPS, and therefore did not know how to correctly respond when the application toasted the message “Turn on your GPS”. This became evident when a respondent mistook turning on the GPS for opening the “Maps” application on the phone Appendix G. A visual representation of the GPS data can be seen in Figure 35.

Figure 35 - Map of GPS data from test 1

### 6.1.4 Trips

A total of 48 trips were completed during the test, but since 4 of the respondents did not submit any GPS data there were a lot of empty trips, trips that were started and stopped by contained no location data. Therefore it is necessary to distinguish between these and the trips that were complete with GPS data. Therefore trips containing location data will be referred to as successful trips.

| Respondent | Trips | Successful trips |
|---|---|---|
| **1** | 2 | 1 |
| **2** | 0 | 0 |
| **3** | 5 | 0 |
| **4** | 14 | 0 |
| **5** | 1 | 0 |
| **6** | 2 | 1 |

| Respondent | Trips | Successful trips |
|---|---|---|
| **7** | 0 | 0 |
| **8** | 5 | 0 |
| **9** | 5 | 3 |
| **10** | 8 | 0 |
| **11** | 4 | 2 |
| **12** | 1 | 0 |

Table 4 - Trips submitted in test 1

As seen in Table 4, a significant amount of the trips were unsuccessful. This is partly due to the respondents that did not submit any GPS data. More noticeably there were respondents that submitted lots of GPS data but only a fraction of it is within a successful trip. This can be due to the respondents simply forgetting to start and stop a trip which was also mentioned in the answers to the questionnaire, but more importantly this fact led us to discover an error in the programming that caused the trips not to be sent if there was no internet connection at the time the stop button was pressed. An issue with the function that determines if it is the start or stop button that should be displayed was also discovered. This issue could also lead to trips not being created properly. The fact that the starting and stopping of trips caused problems is clearly visible from the answers given on the questionnaire which are seen in Figure 36.


Figure 36 - Satisfaction with starting and stopping trips in test 1

Figure 37 shows a screen dump from the visualization platform displaying the successful trips.

Figure 37 - Map of successful trips from test 1

### 6.1.5 Tags and map

During the course of the test only three tags were submitted. Out of these one is a mistaken attempt to use the support function. The submitted tags can be seen in Table 5.

| id | emne | tekst | pic_url | lat | lon |
|---|---|---|---|---|---|
| 5 | Raporter et problem | min mobil har problemer med dette program, så jeg kan desværre ikke rigtig få trykket på start og stop knappen uden at programmet lukker ned | intet billede | 56.2975 | 9.4219 |
| 1 | Raporter et problem | vejarbejde | intet billede | 56.4793 | 9.444 |
| 1 | Syntes godt om! | anlagt cykelstier | pictures/012682000723586133485 8576852.jpg | 56.4819 | 9.448 |

Table 5 - Tags submitted in test 1

The fact that only one respondent chose to use the tag function is quite puzzling. One factor could be that it requires some effort to voice an opinion. The questionnaire made another reason apparent, as the only operation in the application other than editing user data, tagging, requires an internet connection. Many of the respondents did not have a data subscription for their phone and therefore only had internet access when on a Wi-Fi network resulting in them not being able to tag when the situation arises. One user also complained that the network coverage was so bad that it took too long to upload the photo.

The built-in map feature in the application that enables the users to see their own track was generally well received but did not make that much sense for some users since they did not submit any successful tracks giving them an empty map.

### 6.1.6 Battery usage

In order to examine the battery usage of a phone running the application we utilize the fact that the battery level of the device is sent with each position created by the GPS. This enables us to calculate the average time it takes the phone to use one percent of battery and thereby giving us an estimate of how long the battery life will be. The calculations are done using only the data from when the phone is consuming battery, so it is not compromised by the times the device is charging.

Table 6 shows the battery consumption of the three devices that submitted the most data during test one.

| Respondent | 1 | 6 | 9 |
|---|---|---|---|
| Device | Sony Ericsson Xperia Arc | Sony Ericsson Xperia Arc | HTC Incredible S |
| Mean time per percent | 00:10:08 | 00:06:37 | 00:06:18 |
| Approximated total battery life | 16:53:04 | 11:01:14 | 10:30:50 |

Table 6 - Battery usage of three respondents during test 1

The average total battery life from the calculations show that we can expect somewhere between 10 and 17 hours of tracking time on a full charge. This of course varies a lot depending on how much the respondent uses the device for other power consuming activities such as games. This factor is clearly visible in the difference between respondent 1 and 6. They both use the same model but there almost is a six hour difference in battery life. In between models there is also a big difference in the capacities of the battery making some devices more suited for applications with a high demand for power. Another factor that can influence this is the age of the device as the battery capacity will deteriorate with age.

### 6.1.7 Visualization Platform

The visualization platform was, as seen in Figure 38, only used by a fraction of the respondents. This may be because they did not find it relevant to look at the other respondents' tracks. Even though the respondents did not make use of the platform it was a very useful tool for us to have an up to date visualization that let us keep track of the data coming in.

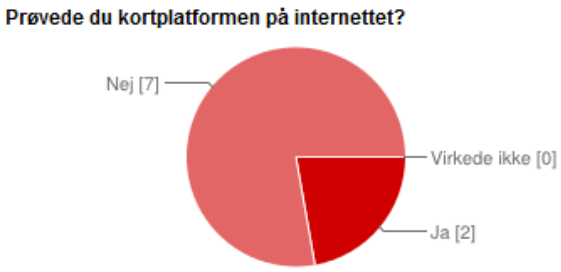



Figure 38 - Use of the visualization platform during test 1

During the test we discovered that there was an issue with the polyline encoder used to turn the individual GPS positions into polylines displayed on the map. The issue resolved itself by simply running the encoding once again. This difference between the first and second encoding can be seen in Figure 39. It is easy to see how the polylines curl up into weird shapes with no connection to the road network, but after the second encoding they follow the roads nicely.

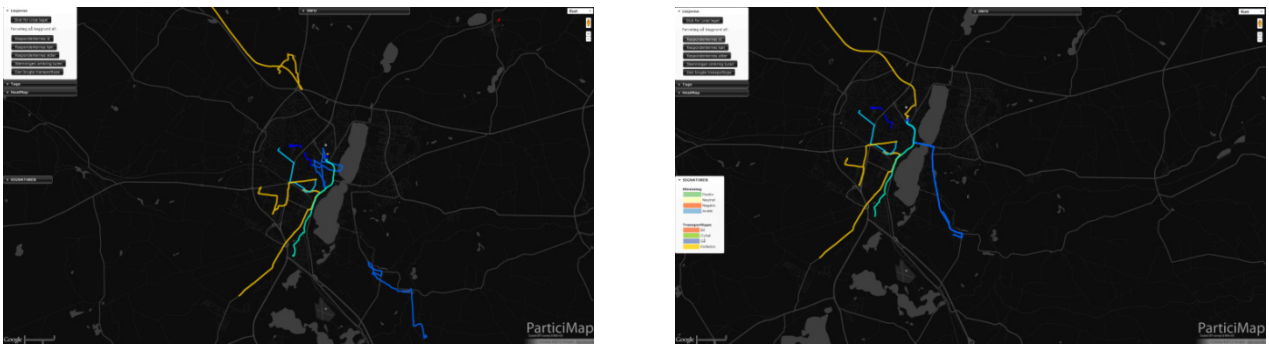



Figure 39 - Polylines after first encoding (left) and second encoding (right)

Why exactly this problem occurs is still a mystery. Since the polyline encoder is not a script made by the project group, there unfortunately had not been the resources to look further into this issue.

As the data amount grew during the test the square count function producing the heat map began to fail. The script did not run all the way to the end, producing only a fraction of the polygons it was supposed to. It happened simply because the script took too long to run and simply timed out because of the massive amount of database queries needed for the operation. Therefore this function was taken offline and should be changed or optimized before a new attempt at deployment.

### 6.1.8 Conclusion of test 1: Necessary changes before next test

In test one it became clear that there were a few issues with both the application, the visualization platform and the way that the respondents were instructed. Therefore a series of changes were to be made before the second test could be commenced. These changes will be described here along with the reasoning behind them, based on the results of test one.

**Changes to the information material**

The instructions given to the respondents proved to be lacking, leaving some respondents with a feeling of not knowing exactly what they were participating in, what they had to do and why they had to do it. Therefore a new introduction letter was created to replace the postcard. The letter contains more detailed information about the use of the application, the exact timeframe of the test as well as some information about what this type of data collection has been used for in the past. The full letter can be seen in Appendix H.

**Changes to the application**

The application revealed itself to have a number of issues that would need to be sorted out. Some were merely errors in the programming that would need to be corrected and some would involve adding to or changing the application.

The first programming error was regarding the showing/hiding of the start and stop buttons. This problem was solved by creating a Boolean value in shared preference which decided if the start or stop button should be shown instead of looking for if a “type” value was set inside shared preference. Another programming error caused the trip data to only be sent when the stop button was pressed. This resulted in data not being sent unless an internet connection was present. The issue was solved by changing the alarm manager triggering the upload from a one-time event to a repeating event.

A major issue revealed during the test was the lack of GPS data from several respondents. This was attributed to the fact that the respondents simply did not know how to turn on their GPS when prompted to do so. Therefore a new function was added so if a

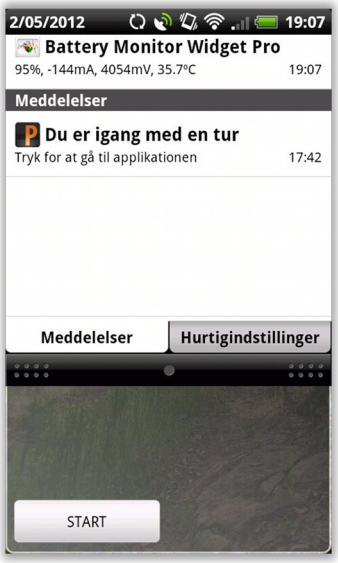



Figure 40 - The application notification

respondent tried to start a trip without having the GPS enabled they would get a dialog-box with a button which would redirect them directly to the settings menu where they could enable GPS positioning.

Remembering to start and stop trips seemed to be a common problem for the respondents. In order to rectify this, a status update to the notification-bar was added. This meant that whenever a trip is started or stopped a small icon will appear in the upper left hand corner of the phone. If the notification is swiped downwards it will reveal the state of the current trip or when data was last submitted (Figure 40). This should make this part of the application more visible to the respondents, reminding them of its presence even when doing other tasks on the phone.

A minor change to the application was in the tag tab. The category menu was showing “Report a problem” as the default setting, which resulted in one respondent mistaking it for a support function. Therefore it was changed so the “Like” option was set as the default value.

**Changes to the visualization platform**

During the test two problems occurred with the visualization platform, one being odd behavior of the polyline encoder/decoder. This issue is still an unsolved problem as there were not enough resources to solve it. The other problem with the platform was the heat map that failed to run when the amount of data became too large. This issue is cased because the webserver times out the script creating it. The only way to solve it is therefore to make the script run faster. One way of doing this is optimizing the database to make the queries run faster. This was done by adding an index of the “lat” and “lon” columns and thereby making the script run eight times faster.

## 6.2 Test two

The second iteration of testing took place in a 72 hour period from the 8th to 10th of May 2012. The test period was extended so that we would have a longer period to gather data and sort out potential issues. The setup for test two is essentially the same as in test one except we would be using the updated application, introduction letter and visualization platform with the changes described in chapter 6.1.8. The respondents were this time recruited amongst the 10 respondents from test one. Out of the ten, eight agreed to participate once more. For easier comparison the respondents will maintain the same respondent number as they had in test one. As in test one, the respondents would be asked to fill out a questionnaire after the test. The full questionnaire and answers can be seen in Appendix I.

### 6.2.1 Initial contact

Three days before the test began the respondents received an introduction email with a more detailed description of the functionalities of the application than the postcard provided. The letter also instructed the respondents to carry out a small pre-test. The day before the actual test was to begin the respondents had to start a trip and let it track for five minutes and then stop it, thereby submitting data to us. This was to catch some potential problems as early as possible so it would not influence the data collection during the actual test. This proved a must better introduction to the test, as can both be seen by the responses from the respondents (Figure 41) and the amount of data collected throughout this test.


Figure 41 - Satisfaction with the information sent out before test 2

### 6.2.2 Initializing the application

Uninstalling the old application and starting up the new version proved to be no problem. At the deadline for conducting the pretest, all eight of the respondents had managed to complete the installation and submit their user data. This is also supported by the answers given in the questionnaire (Figure 42).

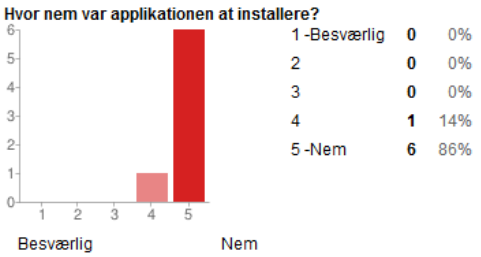



Figure 42 - Ease of installation in test 2

### 6.2.3 GPS data

In the second test the amount of GPS data we received was greatly improved. The added feature of the application that led the respondents directly to the correct settings menu for enabling the GPS, therefore proved to be effective. During the 72 hours of testing a total of 145,089 GPS positions were submitted to the database. The number of positions per respondent is shown in the table below.

| Respondent | # of positions |
|---|---|
| **1** | 6831 |
| **3** | 12770 |
| **4** | 0 |
| **6** | 13487 |
| **8** | 63894 |
| **9** | 11822 |
| **10** | 23178 |
| **11** | 13107 |

Table 7 - Number of positions per respondent, test 2

When looking at Table 7 there are two things that stand out. The first is the fact that one respondent still did not manage to submit any positions. Why this did not succeed is unknown. The respondent was very keen on making it work and followed every instruction we gave, over multiple test messages and phone calls. Unfortunately it was not possible to arrange a meeting with the respondent so the phone could be connected to a computer with a logging program that would enable us to pinpoint exactly what went wrong. In the other end of the scale, respondent 8 logged almost three times more positions than any other respondent. This is due to her phone submitting a position every second instead of approximately every five seconds. This error is particularly puzzling since another respondent had an identical phone running the same version of Android. Again we are unable to tell what the exact error is since it was not possible to connect the users phone to a diagnostics tool.

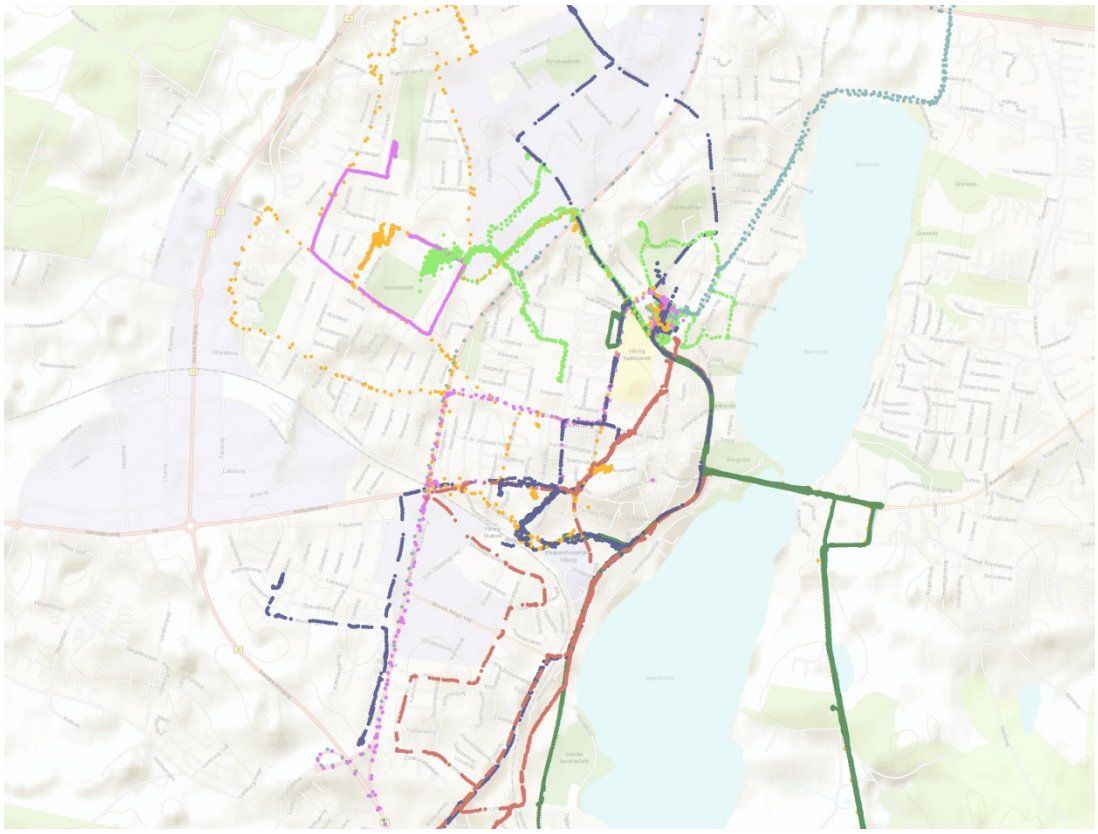


Figure 43 - GPS data collected in test 2

### 6.2.4 Start/stop trips

During test two a total of 43 trips were submitted by the eight respondents, excluding the test trips they were asked to do by us. Out of this 33 contained coordinates and can therefore be described as successfully created trips, which is a great improvement since test one. The number of successful trips per respondent can be seen in the table below.

| Respondent | Trips | Successful trips |
|---|---|---|
| **1** | 5 | 4 |
| **3** | 3 | 3 |
| **4** | 3 | 0 |
| **6** | 8 | 4 |
| **8** | 4 | 3 |
| **9** | 4 | 4 |
| **10** | 4 | 4 |
| **11** | 11 | 11 |

Table 8 - Succsessful trips per respondent

From Table 8 it is clear that the success rate of trips has gone up significantly since test one. This is attributed to the fixes done to the application as well that the respondents are now reminded to start and stop trips whenever they look at their device. The few trips that were still unsuccessful could be due to the respondent making an error and started a new trip, or simply did not get a GPS fix in the duration of the trip. The starting and stopping of trips unfortunately did not improve its rating in the questionnaire (Figure 44). It was still a bit of a problem for some respondents to remember to start and stop trips, but the main complaint was that the application sometimes crashed when stopping a trip. This is an unfortunate mistake in the programming and one that would need to be addressed in future development.


Figure 44 – Satisfaction with starting and stopping trips in test 2

Figure 45 shows a screen dump from the visualization platform displaying the successful trips.


Figure 45 - Visualization of submitted trips in test 2

### 6.2.5 Tags and the map

In test two only three tags were submitted (Table 9), and those were all by the same respondent. Unfortunately we do not know why the respondents chose not to use this feature. But on the positive side, the user that did choose to tag experienced no problems and gave it a perfect rating in the questionnaire.

| id | emne | tekst | pic_url | lat | lon |
|---|---|---|---|---|---|
| **11** | Syntes godt om! | God placering for Rådhus | intet billede | 56.458 | 9.400 |
| **11** | Syntes godt om! | Flot | pictures/012703001053383313365 06206859.jpg | 56.458 | 9.400 |
| **11** | Syntes godt om! | Vejarbejde. Godt med forbedringer | intet billede | 56.461 | 9.398 |

Table 9 - Tags submitted in test 2

The map view in the application was generally well received by the respondents although one still stated that she did not see the purpose of it. During the second test another issue with the polyline encoder/decoder was discovered. In some cases the tracks were not displayed correctly in the application map even though they showed up correct in the visualization platform. The tracks displayed in Figure 46, are made from the same encoded polyline and therefore should be identical. The problem may occur because of the lines are retrieved directly into the JavaScript through PHP instead of through XML.

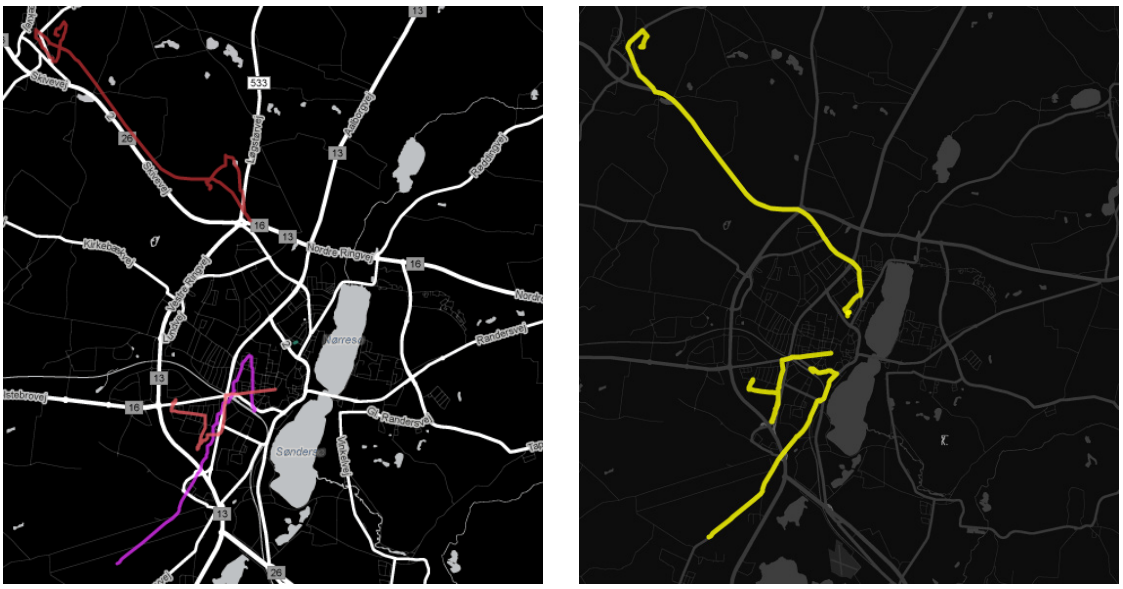


Figure 46 - Trips displayed in application map (left) and trips displayed in web map platform

### 6.2.6 Battery usage

As in test one a calculation will be made to determine the estimated average battery life of a device running the application. This will be done using data from the three respondents that submitted the most data, in this case respondent number 6, 8 and 10. The results of the calculations can be seen in Table 10.

| Respondent | 6 | 8 | 10 |
|---|---|---|---|
| Device | Sony Ericsson Xperia Arc | HTC Legend | Samsung Galaxy S2 |
| Mean time per percent | 00:08:37 | 00:04:45 | 00:09:54 |
| Approximated total battery life | 14:22:04 | 07:54:24 | 16:30:00 |

Table 10 - Battery usage of three respondents in test 2

From the data collected in the test with the normal position-interval of 5 seconds, the devices provided a battery life of over 14 hours. In the case of respondent 8 where a position was created every second the battery life dropped significantly to less than eight hours, which unfortunately is not enough to make it through an average persons day without having to charge.

### 6.2.7 Visualization Platform

During this test not a single respondent chose to use the visualization platform (Figure 47). This may again be because they cannot see the relevance of exploring the data. But the platform functioned well as a means for us to keep track of incoming data, and it would make more sense as part of a bigger survey where the data needed to be analyzed for patterns etc.

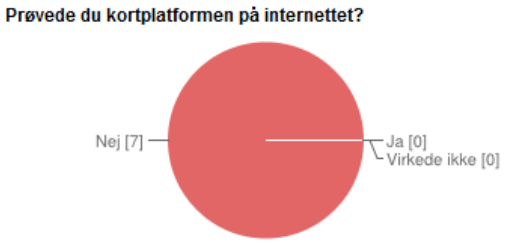



Figure 47 - Use of visualization platform in test 2

The heat map was performing much better after the changes, but when the data set got too large it still failed to execute properly. One fix for this is to increase the size of the squares and thereby reducing the amount of iterations needed, but in order to ensure a continuously stable platform, more optimization is needed.

## 6.3 Conclusion of testing

To conclude the test, the results of the different themes set in the introduction to this chapter will be summarized.

| | |
|---|---|
| **Initial contact** | Handing out a postcard or sending a mail proved to be successful in making the respondents participate in the project, but still some people needed to be contacted for guidance which could have meant that a better solution might have been to hold a meeting informing the respondents, or appointing ambassadors to inform the other respondents. |
| **Initializing the application** | All respondents we know of were able to successfully install the application, and make user data submissions. Some of the people that were given a postcard might have had issues, but did not call the number of the postcard, so there is no way of knowing this for sure. |
| **Start/stop trips** | The respondents were able to start/stop trips. However, a number of people failed to remember to turn it off when they arrived at end destinations. To minimize this, a notification handler was programmed into the application. This helped a lot in making successful trips, but some still admitted they forgot to stop the trip. A good thing though is that GPS signals are not able to penetrate walls so in most cases there are no positions while tracking inside buildings. |
| **GPS data** | After adding the linkage to the setup menu from the application, a lot more GPS positions were obtained. Additionally, not all mobile phones are able to deliver extra information about speed, direction or number of satellites. |
| **Tags and map** | The ability to use tags was not used much, which could have something to do with the respondents not knowing why they should do so, which is also the result from the Vollsmose case. The in-app map was used by nearly all respondents and for the most part it helped them identify their submitted trips. |
| **Battery usage** | Testing showed that tracking for over 14 hours is possible with the application. However one mobile phone had an error making it track each second resulting in a battery life of only 8 hours, but for the most part, tracking a whole day is possible. |
| **Visualizations Platform** | The visualization platform performed as expected but the heat map function had to have its square size increased to 200 meters to minimize the number of squares before the webserver could handle it. |

# 7 Discussion

In the following chapter we will discuss the knowledge we gained from this project. The discussion will run chronologically with the timespan of the project, and evaluate on the entire project period.

The basis for this project was the idea of making a smart and easy application package which reduced the needed resources to manage the technical setup of a mobility survey. One of the main goals was to reduce the impact of participating in the survey, which could interfere with the daily life of the respondents.

GPS theory showed that we could expect the logged positions to have a precicion of about 6 meters. Our test showed that mobile phones can be expected to have an average precision of 3-6 meters influenced by the surroundings tendency to cause multi-path errors on the GPS signal. This could influence the positions to move as much as 50 meters away from the actual location (Figure 48). It was assessed in many cases to not influence path recognition if the interval of positions was relative low.

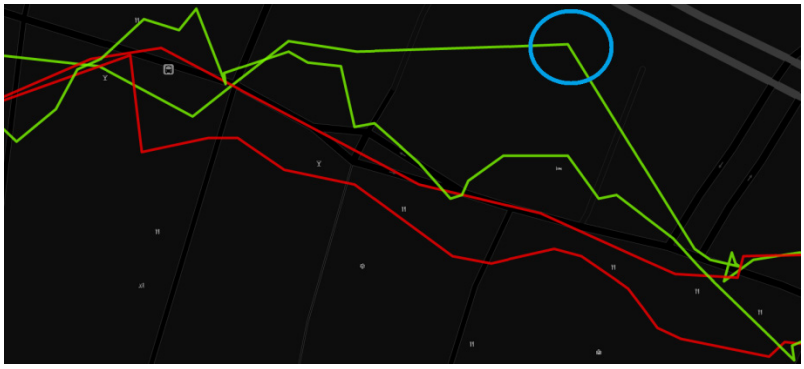


Figure 48 - 50 meter error on GPS track

Another basic technology constraint we were subject of, was the performance of the relational database management system MySQL, which was used in the project. To gain the needed performance within the rules of normalization, meaning not to use duplicated entries that were used with the storage engine MyISAM which excels in insert/update statements. Furthermore, the number of digits in the declared types of the columns was controlled and indices were added to increase performance even more.
To publicize this large amount of data a web based map platform was created. This required a framework and we chose to use the Google Maps JavaScript API which delivers a good WMTS service to use as a background layer as well as the support of encoded polylines/polygons as well as relieving the webserver of the University of some strain. Google Maps is a familiar web map for many people and has a large developer community which makes it easy to adapt and create a fluent user experience.
Displaying the collected data was only one element. The project also needed a mobile platform to develop the application for. Android was chosen because of it had a market share of more than 50% in recent sales with the closest competitor being iOS with only 23%. Furthermore, Android is also easily available to developers because it is open-source, needs no special hardware, and has no initial fee and the finished application can be shared with any owner of an Android device without interference from the source

company. However, the vast differences in Android devices are a clear disadvantage because it means that not all devices act 100% identical. During the tests it was also revealed that a large amount of the students at Viborg Katedralskole were iPhone users. This segment was much larger than what is seen from the market shares. Therefore a logical next step would be to create an iOS version of the application.

When working with mobile devices or other platforms, the graphical interface the user has to interact with a GUI. Some of the basics are that the user does not need to be aware of all his possible actions on the main screen. They should only house basic functionalities and deliver a smooth experience which means all heavy lifting should be done in the background. This concept was used in the design of the simplistic GUI of both the smartphone application and the web platform, as seen in Figure 49.

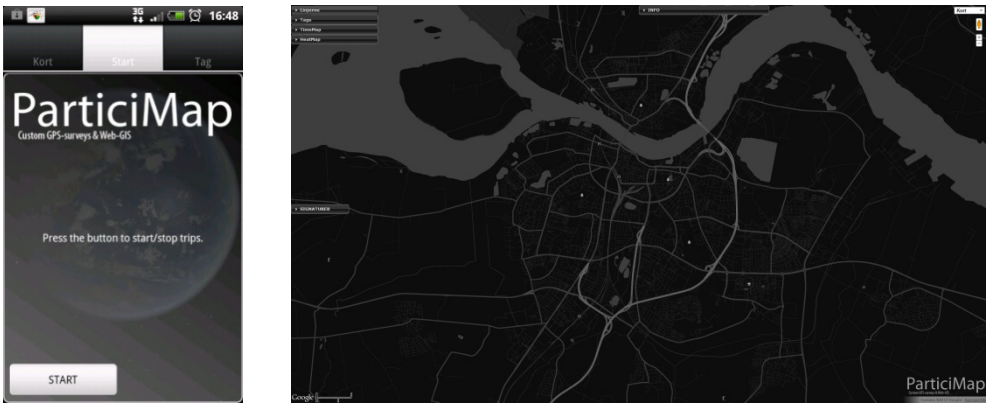



Figure 49 - The GUI of the smartphone application and the web map platform

On a mobile device, areas the user should interact with needs to be large in order for it to be easy to hit on a touch interface.

To develop the application and web platform the project group used Agile Scrum Methodology to ensure a controlled work process. This was supported by holding daily scrum meetings which forced a dialog that helped established a common understanding on which direction to move. The project group also spent time discussing how much time should be set aside for each task in the sprint backlog. This helped to clearly define what the tasks contained and when it could be considered finished. Having every part of the development process cut down to small pieces meant that it was possible to check off several tasks each day, providing a great motivational factor. Documenting each coding piece also meant that we had to gain an in-depth understanding of what the coding bits actually do. However, this also proved to be very time consuming, especially when the complexities of the application immerged, forcing us to stop sprint documentation in the last sprint. Between each sprint the application was tested by the project group and members of the DUS research group. It could however be an improvement to have a permanent testing group to test the application and to use video cameras to film people using the application and commenting on it to gain a more in-depth view of how the UI was perceived and how to apply design changes.

The GUI concept of using tabs and a menu worked well to hide most information from the user when it was not needed. However, an improvement to the start tab could have been to add some statistics such as the last time of data upload, distance traveled and so on. The very first UI design, updated the main view each time a new position was logged, which drained the battery too fast and used a lot of memory. Therefore the start tab was stripped of all information that required updating. Swipe functionality was also tried out to make it easier to change view. This was removed again because it interfered with the manipulation of the map view and scroll functionality in tag view.
A bad habit used in the implementation was that we hardcoded all text and URL-link inside the coding. The ideal solution would have been to create an XML file containing all this data and simply link to that, so only one file would need to be changed in between surveys. It was shortly tested but we experienced some problems with the URL data, which caused faulty HTTP post requests and forced us to postpone the implementation of such functionality.
The interval between GPS positions was set at 5 seconds during the testing. This provided good data but consumed a lot of battery. Increasing the interval to 10 seconds could result in up to 40% extra battery life, but this will always be a balance between the level of detail required from the dataset and the longevity of the tracking in between charges.
A task that proved difficult to solve was ensuring that the respondents remember to start and stop trips. This will always be a problem as this is not a part of their daily routines. The implementation of notifications helped improve this but did not solve the problem completely. Therefore time was spent monitoring the database tables making sure that all respondents were submitting data and contacting those who did not. This was a time consuming task that involved a lot of manual labor. To ease this burden an administrative website could be created to show the status of each respondent with predefined queries. It could also include build in functionality of sending auto generated text messages to the respondents minimizing work load.

The data handling PHP script all worked as planned but could have been improved with an extra variable being sent from the mobile device containing which project the data was related to. This is because each table in the database has a xxx_viborg or xxx_demo tag attached to it, so in the current set up a new set of PHP files has to be created for each project the application is used for.

To retrieve data from the database, XML generation by a PHP script, was used. This proved to be an effective way of retrieving up to date data. However, if the amount of data grows this would eventually prove to be too slow, and converting to a method of generating a premade XML at a set interval, which the web-platform then would access, would be preferable. In the present implementation all the data collected in the two tests take 1.38 seconds to load so it acceptable in set ups like the Aalborg Øst and Vollsmose cases which had approximately the same amount of data.

A downside to using the Google Maps API is the usage limit of only 2500 styled map views per day. If this limit is exceeded Google will start to charge you money for the service, however this is only enforced if the limit is exceeded for more than 90 days (Google Developers n.d.). So the limits should not have any effects with a research project with only 20 respondents, but for larger scale, long term deployment another solution might be needed.

Displaying hotspot-trends on the web-platform proved to be a hassle. Our first implementation used an html5 canvas colorization created by Patrick Wied which used JavaScript to calculate values based on X,Y coordinates**Der blev angivet en ugyldig kilde.**. This worked well until the amount of GPS data grew to over 25000 points and slowed the platform so much we had to shift to the square count analysis described in the implementation, as seen in Figure 50.

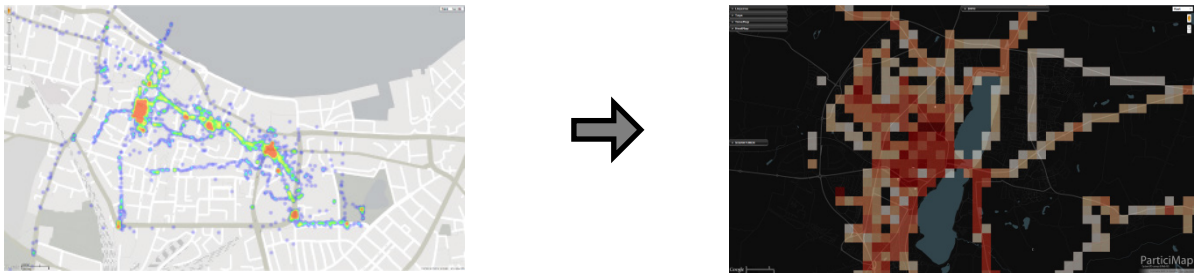



Figure 50 - Patrick Wieds heatmap (left) and the implemented square count (right)

Here we saw the effects of indexing as the time used to generate and count the 316272 squares fell from 472 seconds to 333 seconds. Another way to speed up the creation process was to define a bounding box only covering the area of most interest, because the used method makes polygons for every square where a position has been registered which result in the creation of squares covering the whole southern part of Jutland.
The main problem with using the square count method is that the color scale is not dynamic so we more or less had to predict the dispersion of values before running the test. However, the scale has not changed since test one yet still gives a good picture of hotspots even when the data from the two tests are put together.

The Timemap API uses Mapstraction which meant we had to add our styled map as a new layer instead of the normal way of styling the map, so the function that allows the choice of satellite view was removed to reduce conflicts; however this is not the most desirable solution. The reduction of the GPS data set with approximately 60000 point which was done when creating the KML, improved performance of the Timemap, but also removed the ability to see clusters of points.

After two tests we have gained a lot of insight into the strengths and weaknesses of our system. Even though there are still some issues left to be resolved if the system was to be deployed, the application used in test two still proved to deliver a lot of good data that can be used for further analysis of the movement patterns of the respondents.
The visualization platform has a few quirks and the issues with the polyline encoder/decoder that needs to be resolved, but despite of this the platform as a whole performed smoothly and allows for a quick insight into the collected data. The heat map performs well as long as the data does not scale or the dataset is not too large. This is also something that would need to be resolved before a large scale deployment, but it could be solved by giving the script permission to run for a longer time before timing out. This is however not possible during the testing since the server is administered by the university.

The human element proved, by far, the hardest to control. Instructing the respondents and supporting them during the tests proved much more time consuming than first anticipated, but also provided a lot of valuable insight into the shortcomings of the system. Since this is an application designed to collect data in a certain way with a certain purpose it becomes very important that the respondents know and understand the purpose of their participation. This is a problem recreational applications do not face, and it therefore creates much higher demands for the instruction of the users than normal. In a real world application of the system there would be a much more defined goal of the use, rather than just a test, which would possibly make it more relevant for the respondents and thereby making them more liable to use functions such as tagging.

# 8 Conclusion

The project group has been employed as student assistants at the research group Diverse Urban Spaces since 2011. The work has revolved around using GPS devices to collect data about people's movements and stays which has helped city planning procedures and public dialog. The mobility of residents in a specific area is an important factor, necessary in understanding how to make smart cities of tomorrow. The methodology with using dedicated GPS devices has proved costly and difficult. However the technological advantages of even the cheapest mobile phone has created the possibility of using these as the tools for facilitating these uses. We have therefore, in this project, worked on creating a smartphone application which could be used to make improvement to mobility research and to simplify it to foster a more meaningful public participation. To formulate this project the problem thesis is as follows:

---

**How can empirical studies in mobility be improved using mobile phone applications?**

---

To guide the process a series of research statements were established which cover the main difficulties in creating a platform for gathering and displaying people's movement. These questions will be answered below and finally an answer to the thesis statement is given.

1. **How can the shortcomings of mobile devices (short battery life, strange behavior when mobile coverage is poor and the Scatter of GPS signals) be minimized?**

Tests showed that with a logging interval of 10 seconds, which is the normal interval, used in previous mobility studies, a battery life of around 22 hours can be achieved on a mobile phone. This is achieved by collecting GPS positions and saving them to a file which is only uploaded once an hour and if there is no internet connection it will wait an hour and try again. The scatter of GPS signals cannot be removed, but the GPS in smartphones proved to be precise enough to get a clear image of the paths chosen by the respondents.

2. **How can a smartphone application ensure the participation of the respondents?**

Using peoples own mobile devices meant they wore the tracking device at most times, which gave us a lot of GPS positions. The problem came with submitting start and stop times of trips. An attempt to minimize this was by using notifications in the notification bar of the device. To achieve complete focus on remembering to participate in the project, the respondents need to be aware of how their work can help achieve a specific goal in their interest.

3. **How is it possible to create an automated process to create and display tendency analysis without the need for GIS software?**

Our implementation of a Heatmap and the Timemap API proved to be an effective and easy way of quickly displaying tendencies. They had some problems with displaying large amounts of data which was programmatically overcome by reducing data amount, either by making squares larger on the heat map and by reducing clusters of point in the Timemap.

4. **Which strengths and weaknesses arise when using respondents' own devices?**

The problem in using respondents' own devices was that they had different brands and models of smartphones with different versions of the Android operating system. These all handle things a bit differently, resulting in errors we could not foresee. On the positive side the respondents were already familiar with their own device which proved helpful in standard tasks like installing/uninstalling apps and also meant that they always wore the devices and remembered to charge their battery.

Throughout this project it has been established that there is a great potential for conducting empirical studies in mobility using smartphone applications. There are a number of challenges to overcome to ensure that the quality of data is high. The smartphone application will need to be very easy to use, and the respondent must not be able to do anything wrong. Even though developing a specifically designed application for this purpose is a very lengthy process it has shown that it is necessary in order to collect the data needed for use in research and urban planning. The smartphone application and web map platform developed are definitely an improvement, reducing the effort and cost associated with conducting such studies, in comparison with existing solutions.

# Bibliography


Android Design. *Design Principles.* n.d. http://developer.android.com/design/get-started/principles.html (accessed Marts 19, 2012).

Android Developers. *What is Android?* n.d. http://developer.android.com/guide/basics/what-is-android.html (accessed June 2, 2012).

Bell, Charles A. *Expert MySQL.* Apress, 2007.

Buisness eSolutions. *Project Lifecycle Models: How They Differ and When to Use Them.* n.d. http://www.business-esolutions.com/islm.htm (accessed January 11, 2011).

Care4all. *Lommy Personal.* n.d. http://www.care4all.com/da/lommy-personel.aspx (accessed June 2, 2012).

Commission on Engineering and Technical Systems. *The Global Positioning System: A Shared National Asset.* National Academy Press, 1995.

Denso Wave. *QRcode.com.* n.d. http://www.denso-wave.com/qrcode/index-e.html (accessed June 5, 2012).

Diverse Urban Spaces. *Diverse Urban Spaces.* n.d. http://www.detmangfoldigebyrum.dk/uk.php?lang=English (accessed June 2, 2012).

Dueholm, Keld, Mikkel Laurentzius, and Anna B.O. Jensen. *GPS.* Nyt Teknisk Forlag, 2005.

Esri. *ArcGIS Resource Center - Web APIs.* n.d. 2.

Flextrack. *Flextrack Lommy Technical Description Version 1.52.* http://www.flextrack.dk/files/FlextrackModule.pdf, n.d.

Fu, Pinde, and Jiulin Sun. *Web GIS - Principles and applications.* Esri Press, 2011.

Gartner. *Gartner Says Worldwide Sales of Mobile Phones Declined 2 Percent in First Quarter of 2012.* May 16, 2012. http://www.gartner.com/it/page.jsp?id=2017015 (accessed June 2, 2012).

Gibbons Media & Research. *Inside GNSS.* n.d. http://www.insidegnss.com/ (accessed June 2, 2012).

Gilmore, W. J. *Dev Shed.* November 27, 2000. http://www.devshed.com/c/a/MySQL/An-Introduction-to-Database-Normalization/3/ (accessed June 1, 2012).

Google Code. *Timemap.* n.d. https://code.google.com/p/timemap/ (accessed June 2, 2012).

Google Developers. *Google Maps API.* n.d. https://developers.google.com/maps/ (accessed June 02, 2012).

—. *Google Maps JavaScript API v3.* n.d. https://developers.google.com/maps/documentation/javascript/ (accessed June 2, 2012).

Google Play. *Google Play.* n.d. https://play.google.com/ (accessed June 5, 2012).

Hansen, Rikke Kofoed. "Helsingørcyklen – en del af fremtidens togrejse?" *INSITE*, September 2011.

Harder, Henrik, et al. *Collecting knowledge of biking behavior in Copenhagen using GPS : The GPS data collection.* Report, Department of Aechitecture and Design, 2011.

Harmon, John E., and Steven J. Anderson. *The design and implementation of Geographic Information Systems.* John Wiley & Sons, 2003.

Hermes Traffic Intelligence. *Hermes Traffic Intelligence.* n.d. http://www.hermestraffic.com/ (accessed June 2, 2012).

Hribar, Jim. *Encoding polylines for Google Maps.* n.d. http://facstaff.unca.edu/mcmcclur/GoogleMaps/EncodePolyline/ (accessed June 5, 2012).

Jarvinen, Jani, Javier DeSalas, and Jimmy LaMance. *GPS World.* March 1, 2002. http://www.gpsworld.com/gps/assisted-gps-a-low-infrastructure-approach-734 (accessed March 19, 2012).

Johnson, Jeff. *GUI Bloopers 2.0: Common User Interface Design Don'ts and Dos.* Morgan Kaufmann, 2007.

KloiMøller. *kloi.dk.* n.d. http://www.kloi.dk/ (accessed June 2, 2012).

Knudsen, Anne-Marie Sanvig, Henrik Harder, Anders Kvist Simonsen, and Tino Kastbjerg Stigsen. "Employing smart phones as a planning tool: The Vollsmose case." Research paper, 2012.

Knudsen, Anne-Marie Sanvig, Henrik Harder, Anders Kvist Simonsen, Tino Kastbjerg Stigsen, and Michael Weber. *At tegne Vollsmose med fødderne - GPS basere kortlægning af unges bevægelsesmønstre.* Aalborg University: Architecture, Design and Media Technolgy, 2011.

Knudsen, Anne-Marie Sanvig, Henrik Harder, Mia Hesselkjær, Jakob Hjort Hansen, and Ann Sofie Grimshave Christensen. *Forandringsstrategier i by-og boligområder. Danske og internationale erfaringer.* Working Paper, Aalborg: Institut for Arkitektur og Medieteknologi, 2012.

Mapstraction. *Mapstraction.* n.d. http://mapstraction.com/ (accessed June 2, 2012).

Microsoft. *Bing Maps Developer Resources.* n.d. http://www.microsoft.com/maps/developers/web.aspx (accessed June 02, 2012).

NobleProg Training Materials. *Agile Project Management with SCRUM.* March 20, 2012. http://training-course-material.com/training/Agile_Project_Management_with_SCRUM (accessed June 02, 2012).

Objectivity. "Hitting The Relational Wall." 2005. http://www.odbms.org/download/029.01%20Wade%20Hitting%20the%20Relational%20Wall%202005.PDF (accessed June 1, 2012).

OGC. *KML.* April 14, 2008. http://www.opengeospatial.org/standards/kml/ (accessed June 2, 2012).

Open Geospatial Consortium. *OGC Stanbdards.* n.d. http://www.opengeospatial.org/standards/is (accessed June 02, 2012).

Open GPS Tracker. *Open GPS Tracker.* n.d. http://code.google.com/p/open-gpstracker/ (accessed June 2, 2012).

OpenLayers. *OpenLayers.* n.d. http://openlayers.org/ (accessed June 2, 2012).

Oracle Coorporation. *MySQL Storage Engine Comparison Guide.* Oracle Coorporation, 2011.

Oracle. *MySQL.* June 2, 2012. http://dev.mysql.com/doc/refman/5.0/en/create-table.html (accessed June 2, 2012).

PhoneGap. *PhoneGap.* n.d. http://phonegap.com/ (accessed June 2, 2012).

—. *PhoneGap Plugins.* Febuary 2, 2012. http://wiki.phonegap.com/w/page/36752779/PhoneGap%20Plugins (accessed June 2, 2012).

Præstholm, Søren. "Planlægning i hånden på borgeren - fra borgermøde til smartphones og apps?" *Geoforum Perspektiv*, Number 21 2012: 8-13.

Ryan, P. G., S. L. Petersen, G. Peters, and D. Grémillet. "GPS tracking a marine predator: the effects of precision, resolution and sampling rate on foraging tracks of African Penguins." *Marine Biology*, Volume 145 Number 2 2004: 215-223.

Schneider, Robert D. *MySQL® Database Design and Tuning.* MySQL Press, 2005.

Schwaber, Ken, and Jeff Sutherland. *The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game.* Scrum.org, 2011.

Shekhar, Shashi, and Sanjay Chawla. *Spatial Databases - A Tour.* Prentice Hall, 2003.

SIMILE Widgets. *Timeline.* n.d. http://www.simile-widgets.org/timeline/ (accessed June 2, 2012).

Timothy L. Nyerges, Piotr Jankowski. *Regional and Urban GIS - A Desicion Supprort Approach.* New York: The Guilford Press, 2010.

United States Naval Observatory. *GPS Constellation Status.* March 12, 2012. ftp://tycho.usno.navy.mil/pub/gps/gpstd.txt (accessed March 19, 2012).

—. *USNO NAVSTAR Global Positioning System.* n.d. http://tycho.usno.navy.mil/gpsinfo.html (accessed March 19, 2012).

W3C. *HTML5.* June 1, 2012. http://dev.w3.org/html5/spec/ (accessed June 2, 2012).

W3schools. *HTML ISO-8859-1 Reference.* n.d. http://www.w3schools.com/tags/ref_entities.asp (accessed June 5, 2012).

Worboys, Michael F. *GIS - A computing perspectice.* Taylor & Francis, 1995.

# Appendix A
# Effect of different configurations on battery life

The purpose of testing the battery life is to get data on how long continuous tracking is possible using an Android based smartphone. It is a great concern for the development of an application, since an application that severely cuts battery life will create a source of irritation for the respondent since the user cannot be expected to charge their phone every couple of hours. The smartphone used for these test is a Sony Ericsson Xperia Active[1]. This test does not take into account any other use of the device than the tracking, and therefore represents a best case scenario.

The test will be carried out several times, each evaluating a different parameter. This is done to create data on which functions are the most demanding.

The different setups are:

- 5 second interval, internal storage and HTTP post with each position.
- 5 second interval, internal storage disabled, HTTP post with each position.
- 5 second interval, internal storage, performing HTTP-POST once every hour
- 10 second interval, internal storage, performing HTTP-POST once every hour
- 0 second interval, internal storage, performing HTTP-POST once every hour

***Internal storage and HTTP post with each position. (done)***

| | Sony Ericsson Xperia Active |
|---|---|
| Start time (hh:mm:ss) | 15:27:09 |
| End time (hh:mm:ss) | 19:36:58 |
| Time elapsed (hh:mm:ss) | 04:09:49 |
| Battery level at end time | 1 % |
| Number of positions sent | 2997 |
| Average time between positions | 5.001 seconds |

This setup provides almost real time insight into the location of the device, since the position is sent every 5 seconds when a position is created. This setup does however have a very low battery life. A battery life of four hours is definitely something a respondent would notice and would require several recharges a day.

***Internal storage disabled (done)***

| | Sony Ericsson Xperia Active |
|---|---|
| Start time (hh:mm:ss) | 13:20:00 |
| End time (hh:mm:ss) | 17:41:44 |
| Time elapsed (hh:mm:ss) | 04:21:44 |
| Battery level at end time | 1 % |
| Number of positions sent | 3140 |
| Average time between positions | 5.001 seconds |

[1] http://www.sonymobile.com/global-en/products/phones/xperia-active/

With the internal storage in the device disabled the performance was very close to that of it being enabled. Only about 10 minutes was gained be turning the internal storage off. Therefore this might not be the most effective way to reduce power consumption.

***Performing HTTP-POST once every hour (done)***

| | Sony Ericsson Xperia Active |
|---|---|
| Start time (hh:mm:ss) | 14:40:12 |
| End time (hh:mm:ss) | 05:54:15 |
| Time elapsed (hh:mm:ss) | 15:14:03 |
| Battery level at end time | 10 % |
| Number of positions sent | 10297 |
| Average time between positions | 5.326 seconds |

When the HTTP-POST uploading the positions from the internal storage is only done once every hour, the battery life increases dramaticallyto more than 15 hours. This is a timespan that allows a full day of tracking where most people will not need to charge when away from home. Note that the tracking stopped with 10% battery left.

***10 second interval, internal storage, performing HTTP-POST once every hour***

| | Sony Ericsson Xperia Active |
|---|---|
| Start time (hh:mm:ss) | 12:37:55 |
| End time (hh:mm:ss) | 10:23:27 |
| Time elapsed (hh:mm:ss) | 21:45:32 |
| Battery level at end time | 1% |
| Number of positions sent | 7542 |
| Average time between positions | 10.386 |

By doubling the time interval between positions we see an increase in battery life of approximately 6.5 hours bringing it up to almost 22 hours. This should be more than enough for a respondent to get through a day without charging, but comes at the cost of less data.

***0 second interval, internal storage, performing HTTP-POST once every hour***

| | Sony Ericsson Xperia Active |
|---|---|
| Start time (hh:mm:ss) | 10:18:15 |
| End time (hh:mm:ss) | 21:58:02 |
| Time elapsed (hh:mm:ss) | 11:39:47 |
| Battery level at end time | 7% |
| Number of positions sent | 40103 |
| Average time between positions | 1.046 seconds |

Setting the interval between GPS positions to 0 resulted in a position roughly every second. This decreased the battery life to a level where it could become problematic to have enough charge to last through a full day.

## Conclusion

In this test the impact of different application settings were tested. The results indicated that one of the most battery consuming functions is the upload of data. Collecting the data in an internal storage and uploading once every hour saw the effects of this decrease dramatically. The power consumption of the GPS positioning can be altered by changing the interval between the positions. The choice of interval is then a balance between the how detailed the data needs to be and how much battery life is required.

# Appendix B
# The precision of GPS in smartphones
# - Tests in various urban environments

## Purpose and procedure

The purpose of this test is to determine the precision of GPS[2] positions obtained using GPS enabled smartphones in various urban environments. For this test two different smartphones were used: a Sony Ericsson Xperia Active[3] and a HTC Desire S[4].

The phones were left in a specific location for 30 minutes, obtaining a position every five seconds. While the phones are collecting data, their precise location is measured using one of three methods:

1. Measuring the exact location using a RTK GPS (Leica GPS1200[5]).
2. Measuring three locations surrounding positions using RTK GPS and measuring the distance to the phones position using a steel tape measure.
3. Obtaining three known positions from a technical map and measuring the distances from these to the phones position using a steel measuring tape.

The coordinates measured by the phones are then compared to the exact coordinates and the deviations are calculated.

The test will be carried out in the following locations in Aalborg:

- In an open urban environment: Middle of a square. (Gammeltorv)(Method 1)
- Next to large trees in a public park (Kildeparken) (Method 2)
- In a dense urban environment: Narrow street with tall buildings on either side (Rosenlundsgade)(Method 3)

Figure 1 - The phones are placed next to a central lamp post in Gammeltorv in central Aalborg.

[2] http://www.gps.gov/
[3] http://www.sonymobile.com/gb/products/phones/xperia-active/
[4] http://www.htc.com/dk/help/htc-desire-s/
[5] http://www.leica-geosystems.com/en/GNSSGPS-Surveying-Systems-Leica-GPS1200_4521.htm

Figure 2 - In kildeparken in Aalborg the phones were positioned next to a trunk in a group of large trees.

Figure 3 - Rosenlundsgade in the southern part of central Aalborg.
The phones are positioned on a doorstep on the southern side of the street.

## Calculations

The goal of this test is to determine the differences between the exact coordinates and those measured by the GPS' in the two phones. These differences are calculated and used to create a scatterplot of the data. The data collected by the phones are displayed as dots in a coordinate system with origin in the exact position.

The mean, $\mu$, and the standard deviation, $\sigma$, are calculated for each point in the north-south and east-west direction. These values are used to determine the Gaussian functions:

$$\varphi(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

The numerical deviation from the exact position is computed for each point using the deviations in the north-south and east-west directions.

$$Numerical\ deviation = \sqrt{(\Delta x)^2 + (\Delta y)^2}$$

The numerical deviations are used to make a histogram showing the distribution of points in certain distance intervals from the exact position.

## Gammeltorv

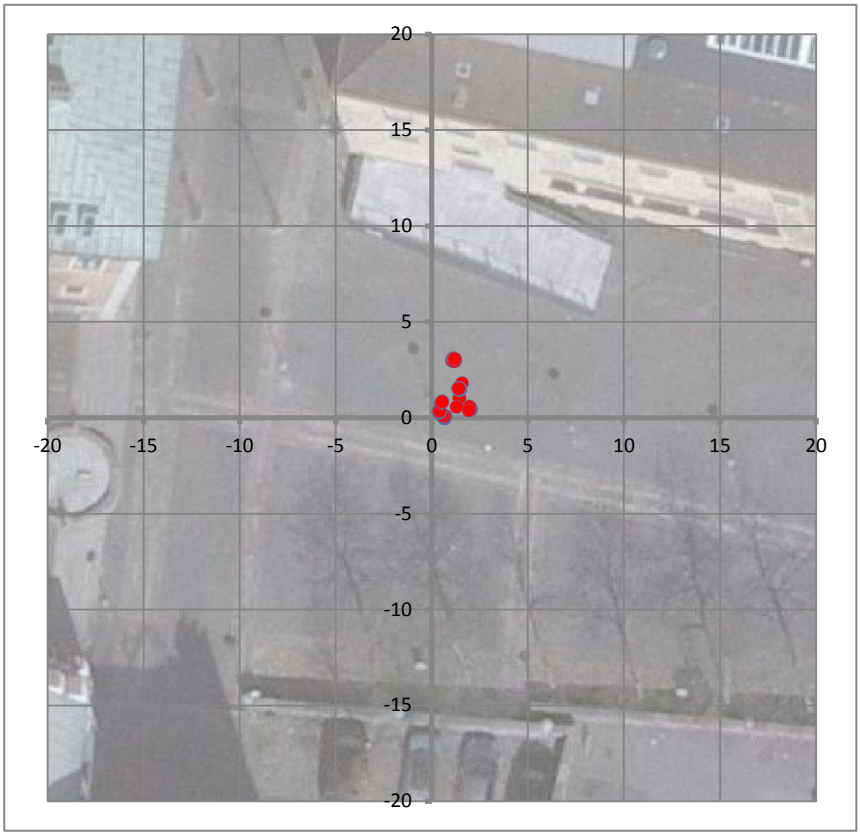
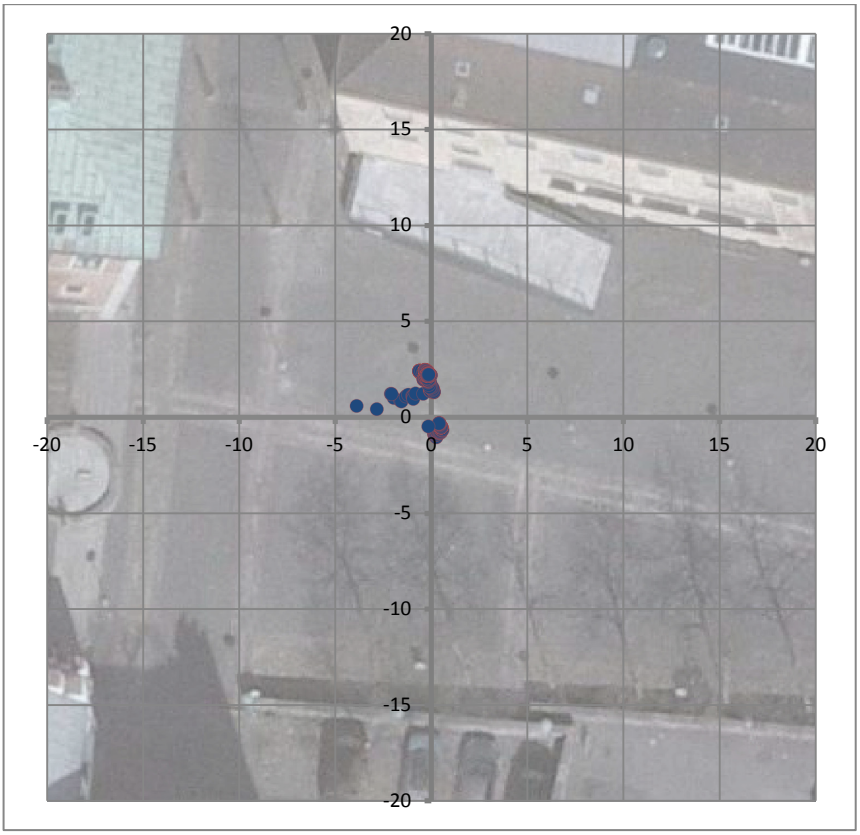



Figure 4 - Scatter-plot of the data collected with the HTC Desire S in Gammeltorv. The values on the axis are given in meters.

Figure 5 - Scatter-plot of the data collected with the Sony Ericsson Xperia Active in Gammeltorv. The values on the axis are given in metres.



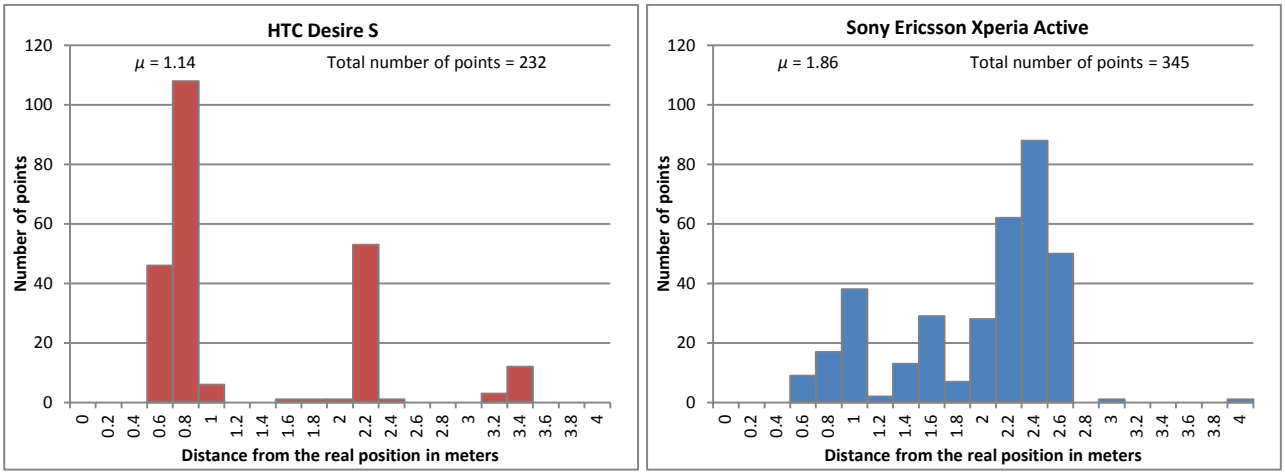



Figure 6 - The numerical deviations of the data from Gammeltorv.

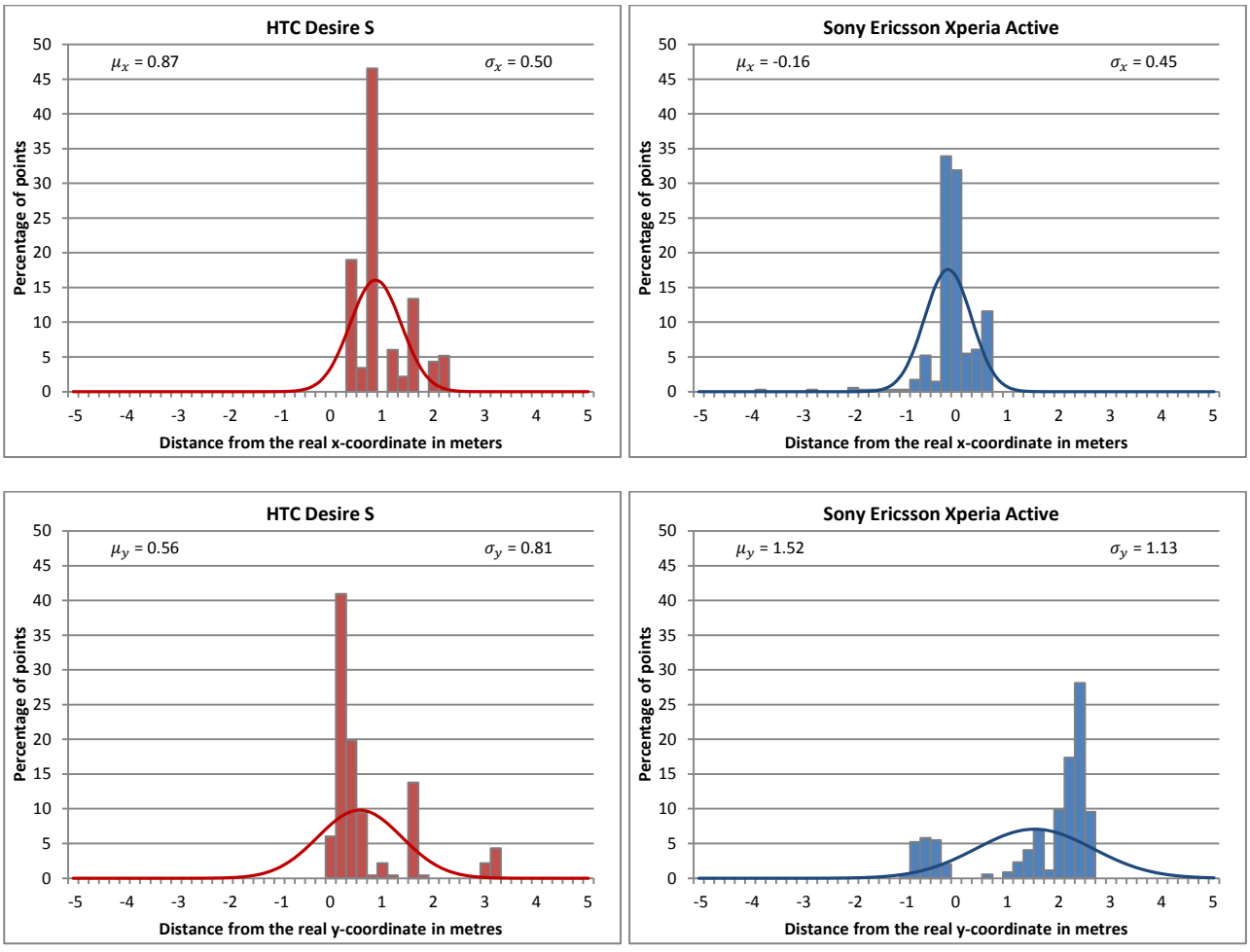



Figure 7 - The deviations of x and y for the data from Gammeltorv and the approximated Gaussian functions.

## Kildeparken

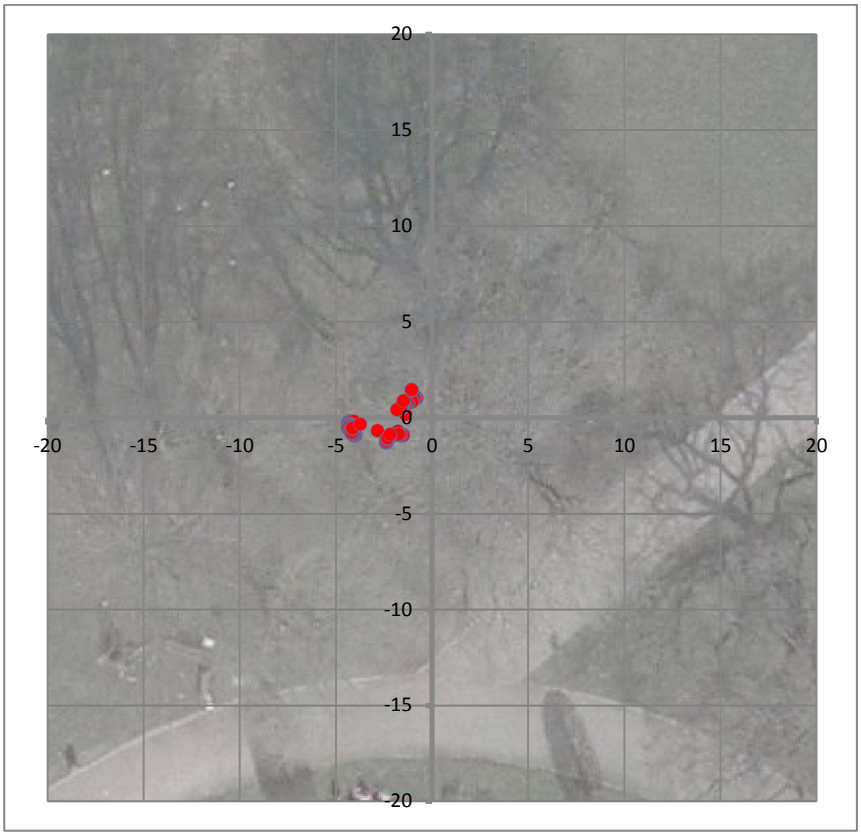
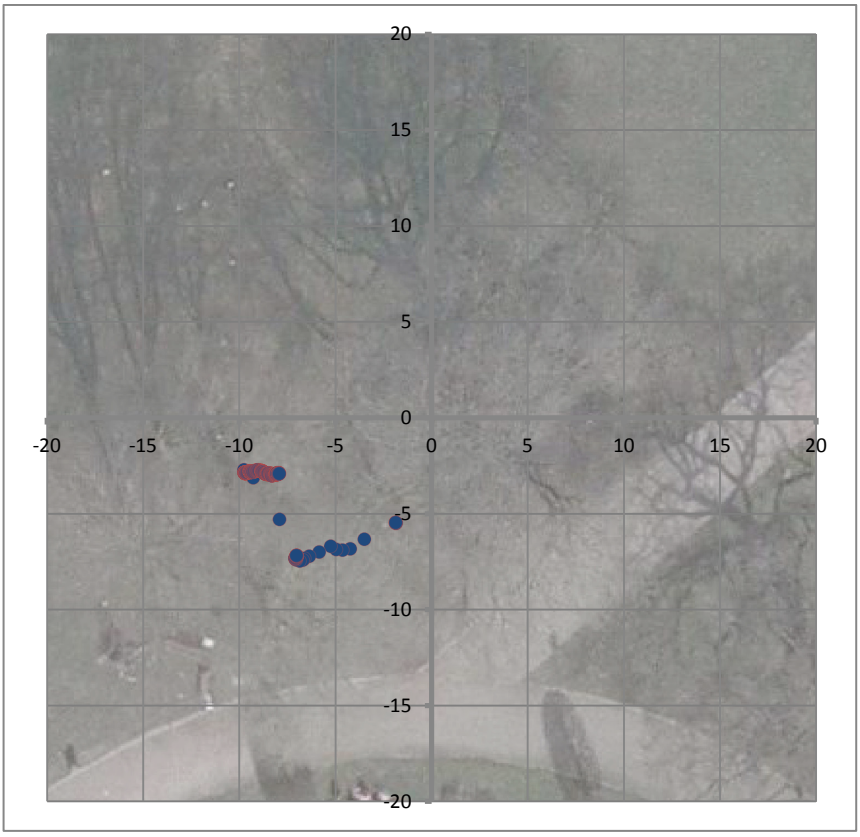



Figure 8 - Scatter-plot of the data collected with the HTC Desire S in Kildeparken. The values on the axis are given in meters.

Figure 9 - Scatter-plot of the data collected with the Sony Ericsson Xperia Active in Kildeparken. The values on the axis are given in meters.



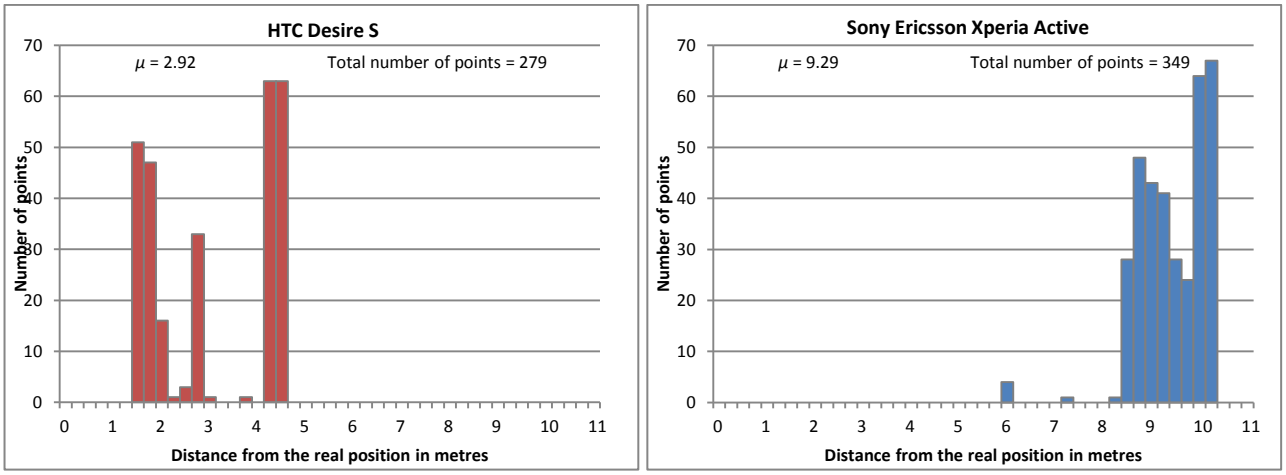



Figure 10 - The numerical deviations of the data from Kildeparken.

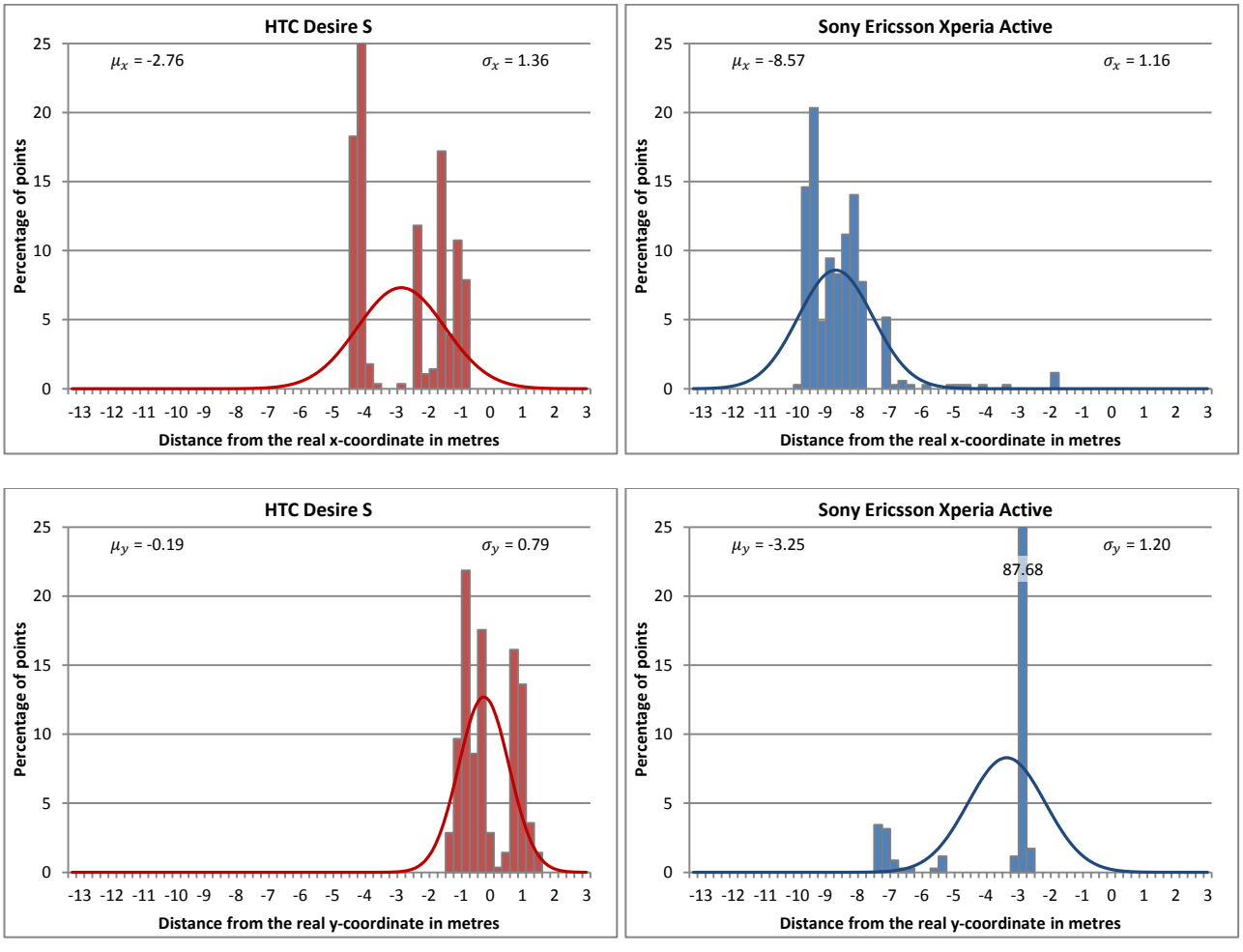



Figure 11 - The deviations of x and y for the data from Kildeparken and the approximated Gaussian functions.

## Rosenlundsgade


Figure 12 - Scatter-plot of the data collected with the HTC Desire S in Rosenlundsgade. The values on the axis are given in meters.

Figure 13- Scatter-plot of the data collected with the Sony Ericsson Xperia Active in Rosenlundgade. The values on the axis are given in meters.

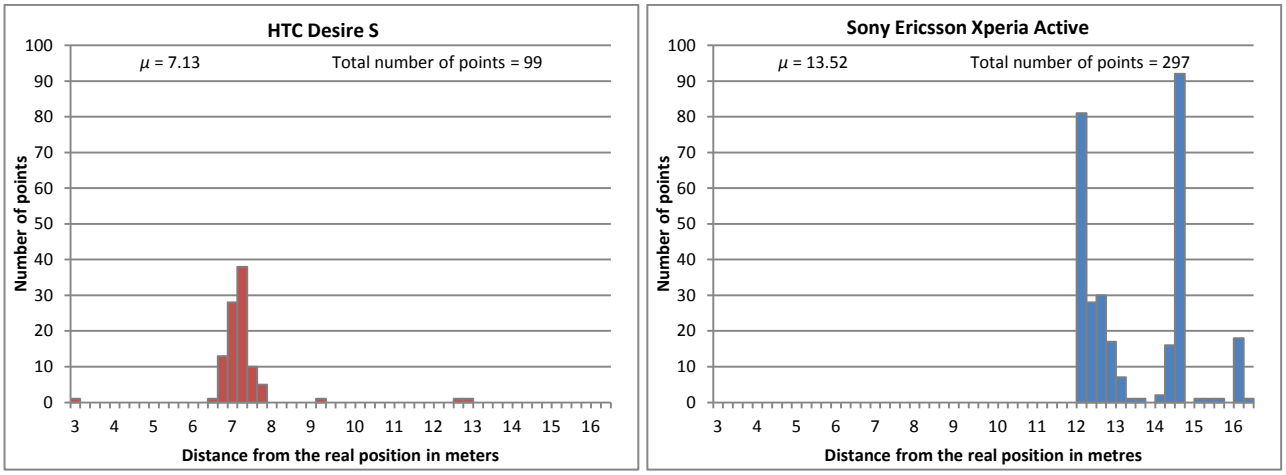



Figure 14 - The numerical deviations of the data from Rosenlundsgade.

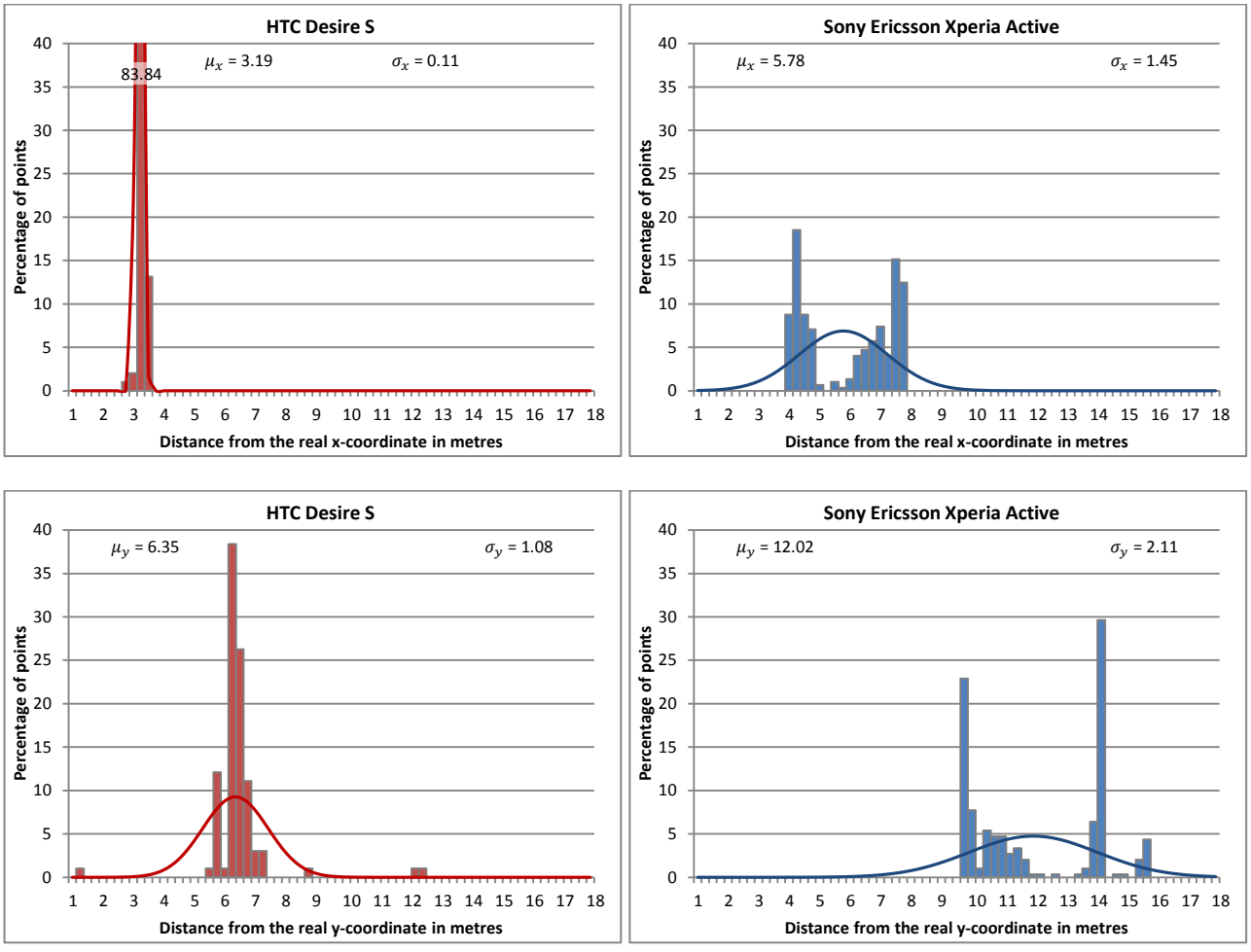



Figure 15 - The deviations of x and y for the data from Rosenlundsgade and the approximated Gaussian functions.

# Conclusion

**HTC Desire S**

| Location | # of points | Average deviation | $\mu_x$ | $\sigma_x$ | $\mu_y$ | $\sigma_y$ |
|---|---|---|---|---|---|---|
| Gammeltorv | 232 | 1.14 | 0.87 | 0.50 | 0.56 | 0.81 |
| Kildeparken | 279 | 2.92 | -2.76 | 1.36 | -0.19 | 0.79 |
| Rosenlundsgade | 99 | 7.13 | 3.19 | 0.11 | 6.35 | 1.08 |

**Sony Ericsson Xperia Active**

| Location | # of points | Average deviation | $\mu_x$ | $\sigma_x$ | $\mu_y$ | $\sigma_y$ |
|---|---|---|---|---|---|---|
| Gammeltorv | 345 | 1.86 | -0.16 | 0.45 | 1.52 | 1.13 |
| Kildeparken | 349 | 9.29 | -8.57 | 1.16 | -3.25 | 1.20 |
| Rosenlundsgade | 297 | 13.52 | 5.78 | 1.45 | 12.02 | 2.11 |

Table 1 - A summary of the results

The results show that the GPS in the smartphones are more precise in open spaces such as Gammeltorv than in areas with a limited or obstructed view of the sky such as Kildeparken and Rosenlundsgade. Despite this the results show that even under bad conditions for reception the phones maintain an accuracy good enough for tracking peoples movements through a diverse urban environment.

A positive aspect of the results is the consistently small average deviation. This indicates that the positions however wrong they might be are very consistent. The result of this consistency is that it reduces scatter in the tracking when the device is stationary.

A thing to notice is the very distinct difference in the amount of positionings made by the two devices which becomes increasingly apparent under tough conditions. This is believed to be due to the fact that the Sony Ericsson is able to use the GLONASS[6] satellites as well as the GPS satellites, therefore being able to have better conditions for obtaining a position.

[6] http://www.glonass-center.ru/en/

# Appendix C
# GPS accuracy when moving

In Appendix B it was found that a static GPS positioning had an average precision of between 1-15 meters according to location. However, we are also tracking movement and are therefore interested in how the GPS positioning perform when moving. To investigate this we tested the positioning in 3 different surroundings; low density urban environment and forest, high density urban environment and on the motorway. The distance from the positions to the route is calculated to give an idea of how good the positioning is. This methodology does not take the time factor into account; a position could move backward or forward along the route and would therefore in our test have a better accuracy than it actually has. The two devices used for the tests are a HTC Desire S[7] and a Sony Ericsson Xperia Active[8]

## Walking in a low density urban environment and a forest area

To test the performance in low density urban environments and forest areas the two mobile phones are carried ina pocket on a run with a dog. The distance is about 7 kilometers. Figure 1 shows the route as a green line with dots displaying the logged positions. The SE Xperia Active is red and the HTC Desire S is blue.

Figure 1 - Plot of the route: Low density urban environment and forest area

[7] http://www.htc.com/dk/smartphones/htc-desire-s/
[8] http://www.sonymobile.com/global-en/products/phones/xperia-active/

| Model | Average distance | Max distance | Min distance | # of positions |
|---|---|---|---|---|
| HTC Desire S | 3.68 | 14.21 | 0.00 | 431 |
| SE Xperia Active | 6.23 | 26.40 | 0.02 | 408 |

Table 1 – Results: Low density urban environment and forest area

As seen in Table 1, the HTC Desire S performs a bit better which was also the case in the static investigation in Kildeparken. As seen on the charts below, HTC Desire S has its positions close to the route where SE Xperia Active tends to deviate more.

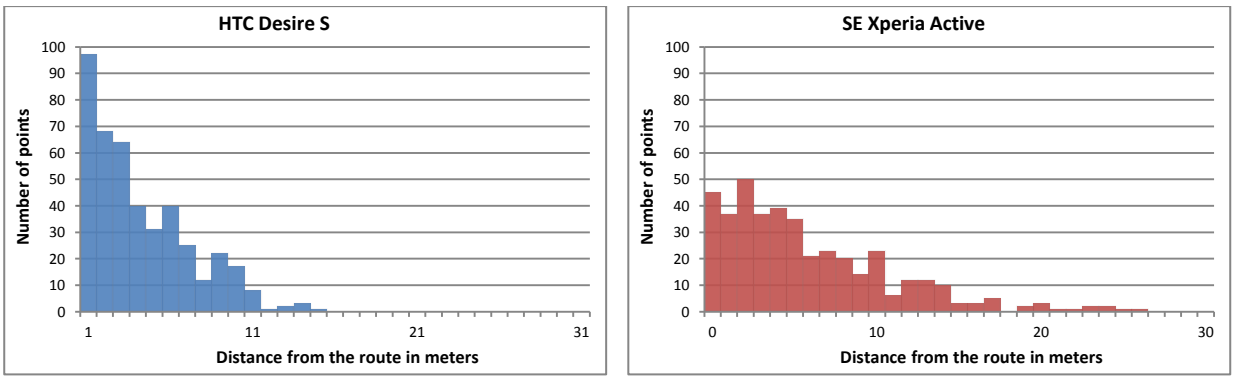



Figure 2 - Deviations: Low density urban environment and forest area

## Car driving on motorway

This test was done with the two mobile phones in the car door driving along E45 from exit 13 to exit 21.

Figure 3 - Plot of the route: Car driving on motorway

| Model | Avarage distance | Max distance | Min distance | # of positions |
|---|---|---|---|---|
| HTC Desire S | 3.92 | 11.48 | 0.00 | 234 |
| SE Xperia Active | 3.01 | 11.63 | 0.03 | 284 |

Table 2 – Results: Car driving on motorway

Both mobile phones have an accuracy below 4 meters in average with a spread of 0-12 meters.

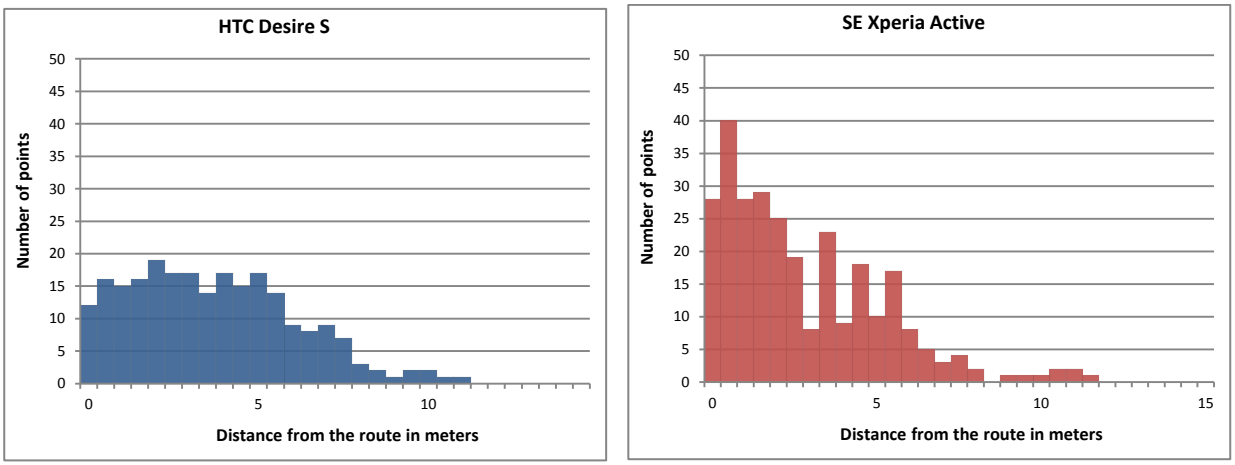



Figure 4 - Deviations: Car driving on motorway

When examining Figure 4 it is visible that the HTC Desire S now has a more disperse distribution of accuracy than the SE Xperia Active, that has an accuracy more closely centered on 0.

## Bicycling in dense urban environment

This test was done on a biking trip from Aalborg to Nørresundby and back. The path crosses a bridge which according to GPS theory gives more precise positions, which is also seen in Figure 5. This will affect the average deviation a bit to the positive side.

Figure 5 - Plot of the route: Dense urban environment

| Model | Avarage distance | Max distance | Min distance | # of positions |
|---|---|---|---|---|
| HTC Desire S | 6.78 | 32.88 | 0.09 | 508 |
| SE Xperia Active | 6.50 | 49.68 | 0.01 | 597 |

Table 3 - Results: Dense urban environment

As seen from Table 3; the maximum distance here is very high (50 meters). However Figure 6 shows that the most common deviations are between 0-25 meters.

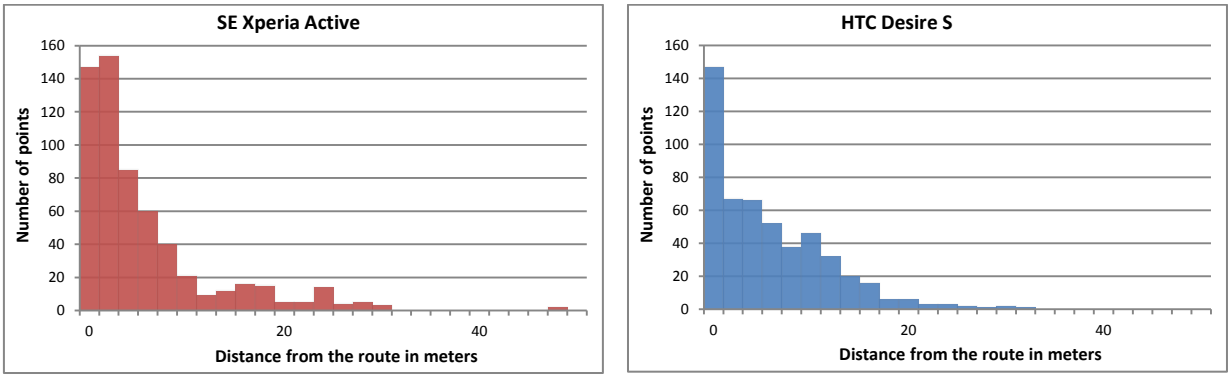



Figure 6 - Deviations: Dense urban environment

One might argue that this low precision will make it impossible to "guess" the route of the respondents. However, as the illustration in Figure 7 shows (the 50 meter deviation is marked with the blue circle) it is not a systematic error, so the route of the respondent is still recognizable.

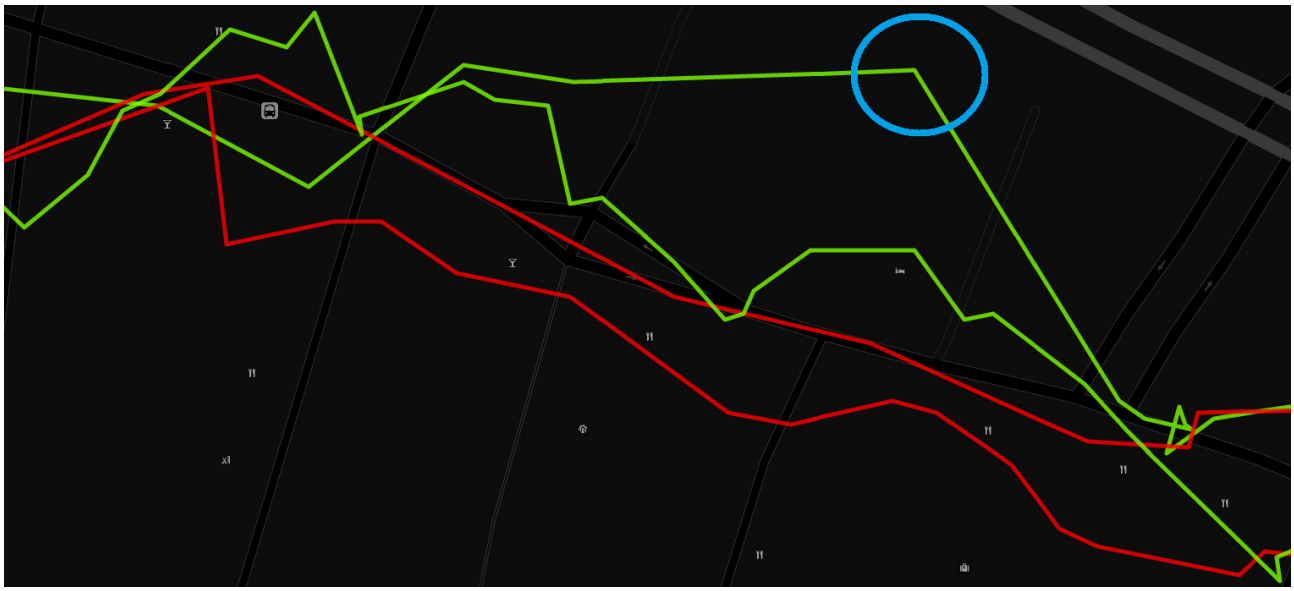


Figure 7 - The 50 meter deviation

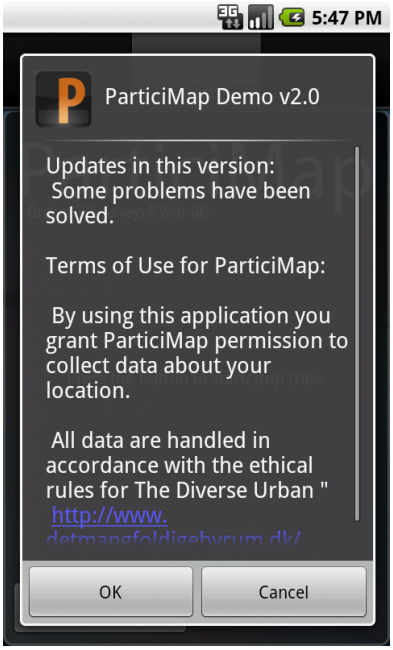
## Conclusion

The findings during the movement tests support the findings from the stationary accuracy tests in Appendix B. The GPS is more precise in open spaces and less if there is a change for multi-path errors and limited view of the sky like in a dense urban environment. The positions are consistently below 25 meters in accuracy, and mostly better, which is still estimated as adequate to preform tracking of movement in urban areas.

# Appendix D

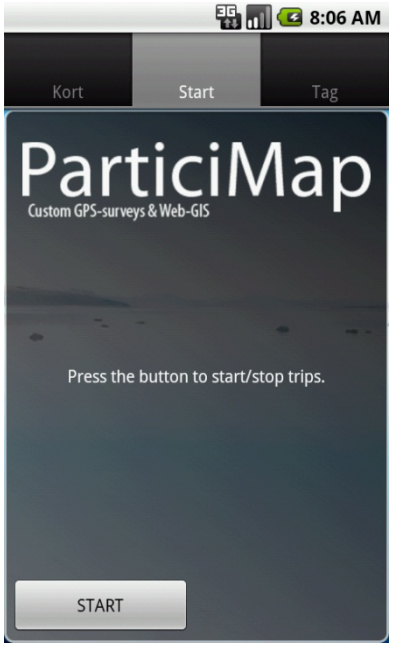
# Screenshots from the smartphone application

**EULA**

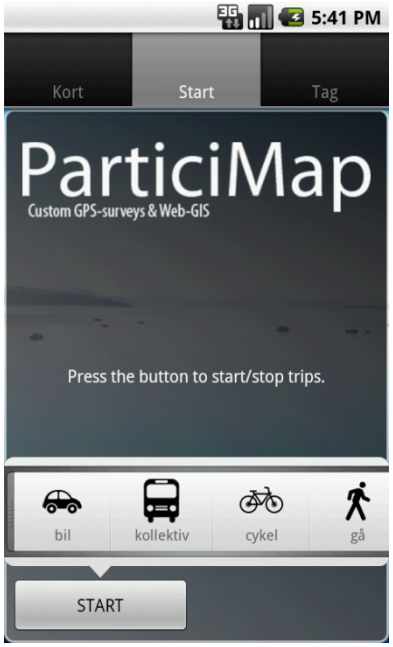



**User data view**


**Main view**




**Choose mode of transport**



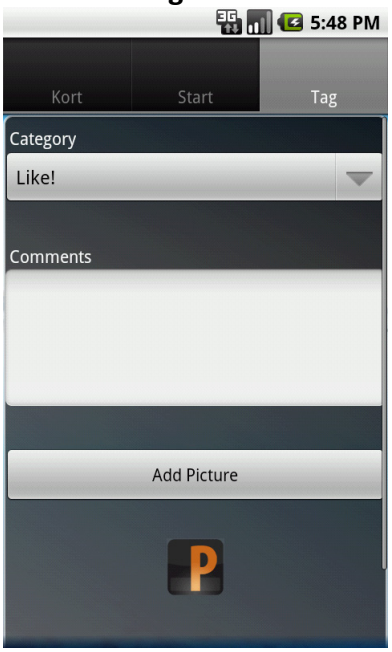

**Choose feeling about trip**


**Write reason for this feeling**

**Map view**

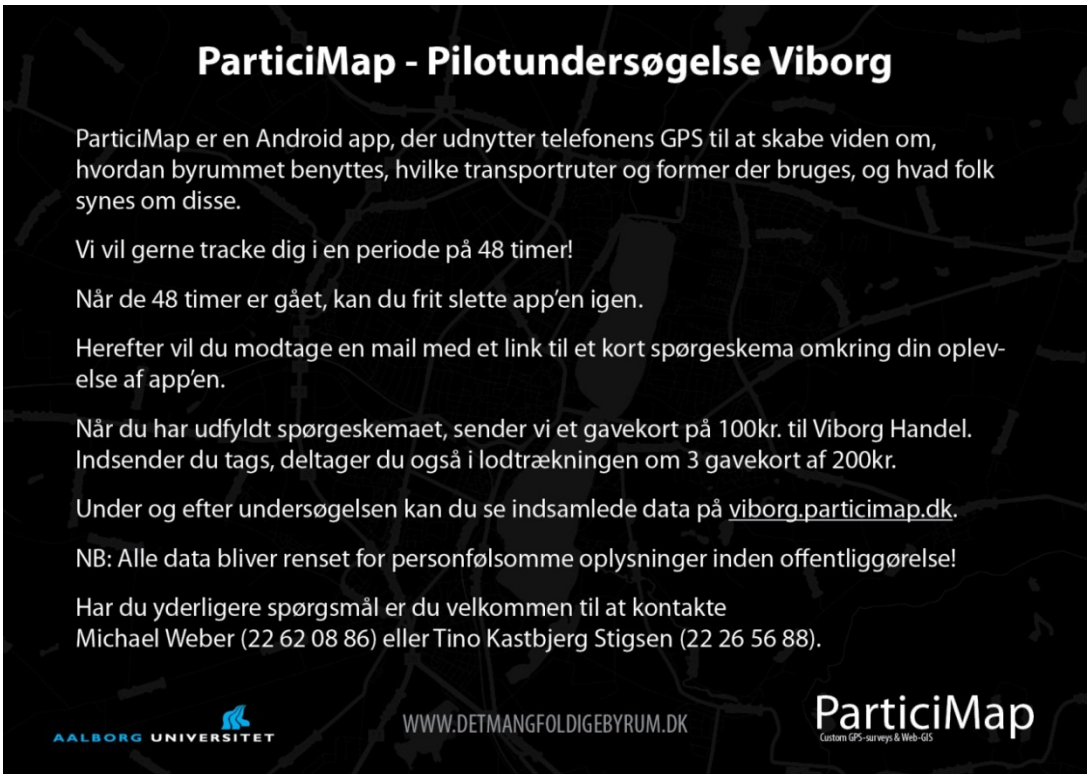

**SMS support**

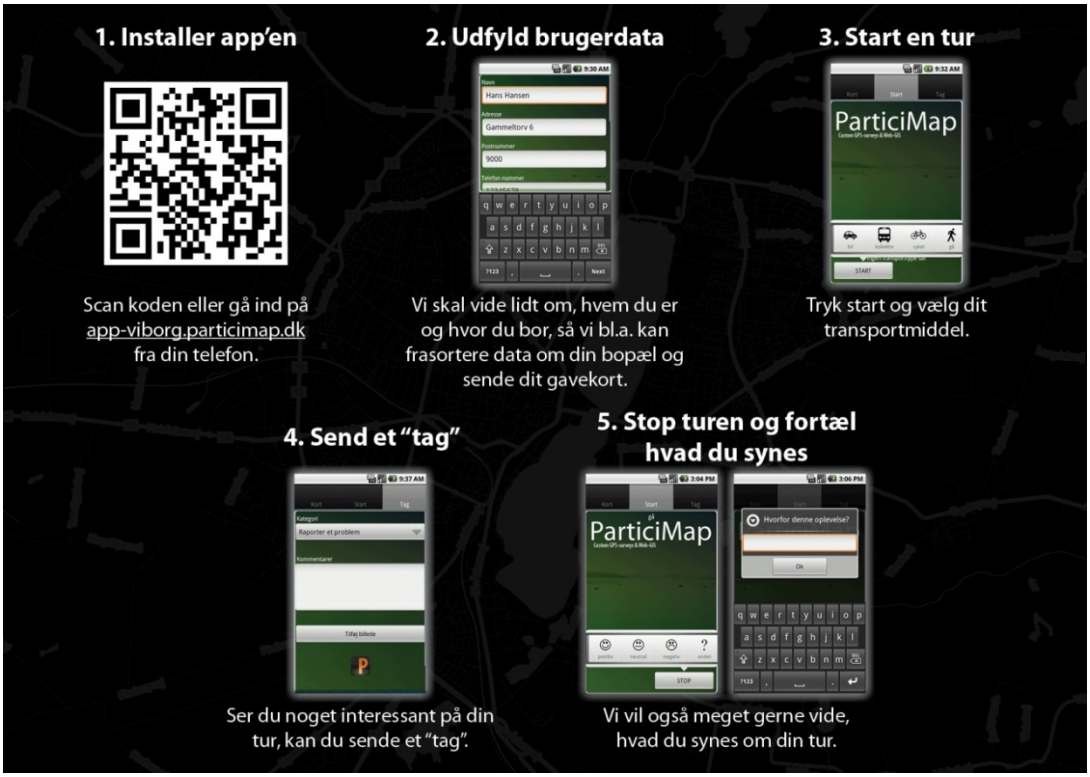

**Tag view**



# Appendix E

# Postcard used in test one



# Appendix F

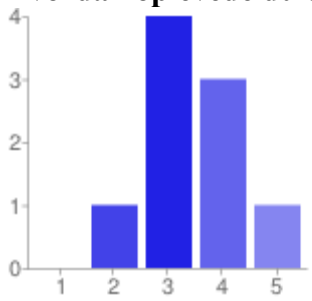
# Questionnaire from test one

# 9 responses

## Summary See complete responses

### Overordnet indtryk

**Hvordan oplevede du det at bruge applikationen?**



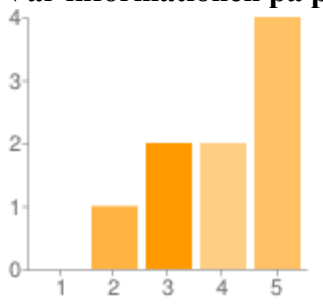

| | | | |
|---|---|---|---|
| 1 - | Dårligt | **0** | 0% |
| 2 | | **1** | 11% |
| 3 | | **4** | 44% |
| 4 | | **3** | 33% |
| 5 - | Godt | **1** | 11% |

**Hvorfor denne oplevelse?**

Der var som sagt en del problemer med appen. men den var alligevel let at benytte.Jeg synes det var en godt produceret app, men den var bare lidt svær at bruge. men det er helt sikkert noget der kan ændres.Det har ikke været en dårlig oplevelse som sådan, da det ikke har været tidskrævende, og det var nemt, men på den anden side var det jo heller decideret sjovtDet var meget let at bruge applikationen, men det virkede desværre ikke. Så jeg håber, at det virker i den nye undersøgelse. Det var overall en rigtig god app, men til tider havde turen stoppet sig selv, uden jeg havde trykket stop. ...

### Indledende information

**Var informationen på postkortet tilstrækkelig?**



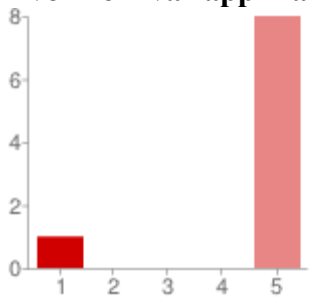

| | | | |
|---|---|---|---|
| 1 - | Dårlig | **0** | 0% |
| 2 | | **1** | 11% |
| 3 | | **2** | 22% |
| 4 | | **2** | 22% |
| 5 - | God | **4** | 44% |

**Hvad kunne der gøres for at forbedre informationsmaterialet?**

Vi var flere der ikke forstod hvad meningen med undersøgelsen var.Det var super godt! Alt hvad man skulle bruge!Man kunne få en sms med det samme, som bekræftede om man nu var registret, i stedet for at gå og være i tvivl om man overhovedet er med i undersøgelsenJeg havde lidt problemer med at installere applikationen - men jeg fandt ud af det selv. Der var dog ikke skrevet helt nok om, hvad man skulle gøre, hvis det ikke ville virke. Den gav mig alt det info jeg havde brug for, ja! mere informationJeg synes der var nok info på kortet, da jeg kunne forstå det hele og bruge det meste af det ...

## Installation af applikationen

### Hvor nem var applikationen at installere?



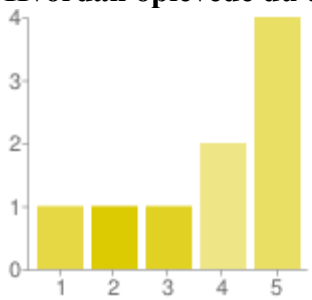

| | | | |
|---|---|---|---|
| 1 - | Besværlig | **1** | 11% |
| 2 | | **0** | 0% |
| 3 | | **0** | 0% |
| 4 | | **0** | 0% |
| 5 - | Nem | **8** | 89% |

**Hvad kunne være anderledes?**

Ikke noget ;-)Tænkte ikke over den var svær at installere, så den har nok været som alle andre app's, meget nem.Den var nem at bruge - intet kompliceret overhovedetFlere informationer om, hvad man skal gøre, hvis der er problemer med at få installeret applikationen. Ingen problemer. ikke noget.Meget meget nem at installereSuper nem, ingen problemeringenting.

## Start/stop af tur

### Hvordan oplevede du det at skulle starte og stoppe ture?



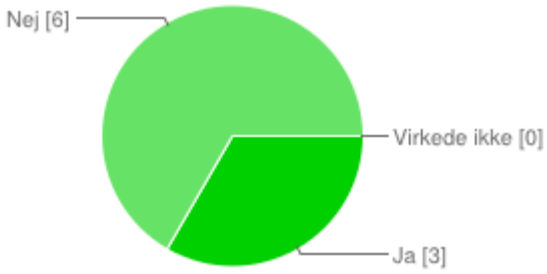

| | | | |
|---|---|---|---|
| 1 - | Dårligt | **1** | 11% |
| 2 | | **1** | 11% |
| 3 | | **1** | 11% |
| 4 | | **2** | 22% |
| 5 - | Godt | **4** | 44% |

**Hvad kunne være anderledes?**

Det virkede superVed ikke om det er muligt man bare hele tiden blev overvåget og på den måde kunne se turen??Det tog kort tid, og var lige at gå til. Meget let.Som nævnt tidligere, stoppede den nogle gange uden jeg ønskede det. mere informationStart husker man for det meste, men stop derimod er en helt anden sag. Der har jeg op til flere gange kommet hjem og glemt at trykke stop, hvor jeg først har set det meget senere og trykket der. Kunne godt tænke mig at vide om det gør det svære for jer, eller om det er ligemegetJeg havde lidt problemer med at huske at gøre det, ellers fint.noget der mi...

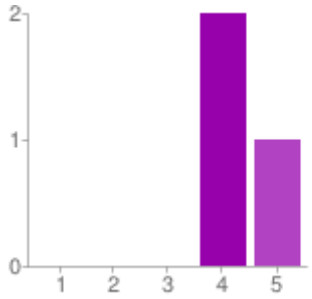
## Tag

### Fik du prøvet at tagge?




| | | |
|---|---|---|
| Ja | **3** | 33% |
| Nej | **6** | 67% |
| Virkede ikke | **0** | 0% |

### Hvis ja, hvordan gik det?




Dårligt Godt

| | | | |
|---|---|---|---|
| 1 - | Dårligt | **0** | 0% |
| 2 | | **0** | 0% |
| 3 | | **0** | 0% |
| 4 | | **2** | 22% |
| 5 - | Godt | **1** | 11% |

**Hvad kunne være anderledes?**

Det virkede ikkeDet tog lidt for lang tid at uploade sit tag - så mister man lysten til at gøre det igen, selv om det nok havde meget at gøre med internetforbindelsenGanske fint! Det virker rigtigt godt. information.Men det skal jeg da ud og prøve

---

## Menu

### Prøvede du nogle af funktionerne i menuen?

| | | |
|---|---|---|
| SMS support | **0** | 0% |
| Brugerdata | **1** | 100% |
| Stop logning | **1** | 100% |

People may select more than one checkbox, so percentages may add up to more than 100%.

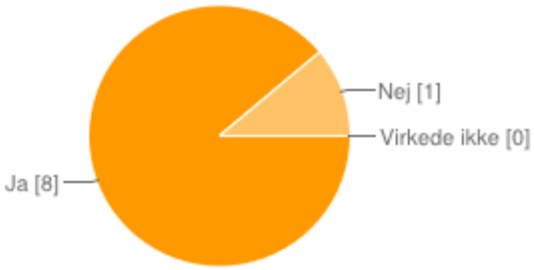
**Hvad kunne være anderledes?**

Jeg havde ikke behov for det, så nej. :)En menu knap, der fortæller om de forskellige ting man kan gøre i programmet.

---

## Visning af dine spor i applikationen

### Prøvede du at se kortet på mobiltelefonen?




| | | |
|---|---|---|
| Ja | **8** | 89% |
| Nej | **1** | 11% |
| Virkede ikke | **0** | 0% |

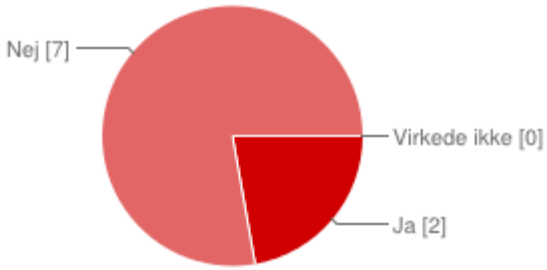
**Hvad kunne være anderledes?**

Ikke nogetDet virkede godt! Ikke noget :-)grafikken kunne godt være bedre, forstod ikke hvad det skulle bruges til.

---

## Visualiseringsplatformen

**Prøvede du kortplatformen på internettet?**




| | | |
|---|---|---|
| Ja | **2** | 22% |
| Nej | **7** | 78% |
| Virkede ikke | **0** | 0% |

**Hvad kunne være anderledes?**
Helt fint!

---

## Øvrige kommentarer

**Har du yderligere kommentarer?**
Fin app, men skal have lidt justeringer.Overall rigtig god app!

# Appendix G
# Text message troubleshooting

**Send To respondent at Apr 20, 2012 15:40 :**

Hej, kan se vi kun modtager informationer om at du starter og stopper ture, men vi får ikke noget data fra din GPS. Er det muligt du ikke har tændt for den?l

**From respondent at Apr 20, 2012 15:56 :**

Skal jeg tænde for mobilens oprindelige GPS? Jeg har tændt for Maps nu.

**Send To respondent at Apr 20, 2012 15:57 :**

Ja. I telefonens indstillinger skal du tænde for GPS.

**From respondent at Apr 20, 2012 15:59 :**

Nu har jeg tændt for bevægelsesaktivering.

**Send To respondent at Apr 20, 2012 16:00 :**

Ok. Hedder GPS'en det i dine indstillinger? Hvilken telefon har du?

**From respondent at Apr 20, 2012 16:02 :**

Samsung Galaxy S2. Min GPS hedder Maps.  Man tænder ikke for det i indstillinger. Modtager i data fra min GPS nu?

**From respondent at Apr 20, 2012 16:04 :**

Jeg har aktiveret "Brug GPS-satellitter" i indstillinger nu

# Appendix H

# Letter introducing test two

Hej «Fornavn»,

Tusind tak fordi du vil deltage i endnu en undersøgelse!

Her følger nogle oplysninger om næste uges undersøgelse. Vi vil gerne bede dig læse det hel igennem.

Denne undersøgelse vil komme til at foregå fra tirsdag d. 8/5-12 om morgenen, når du står op, og indtil torsdag d. 10/5-12, om aftenen, når du går i seng. Der vil ligesom i sidste undersøgelse blive udsendt et spørgeskema, som skal besvares inden vi sender gavekortet.

Inden undersøgelsen går i gang vil vi dog gerne sikre os at alting virker som det skal og applikationerne kører uden problemer på din telefon. Derfor vil vi bede dig om at gøre følgende senest mandag d. 7/5-12 kl. 18.00.

1. Installere app'en og starte den.
2. Sætte en tur i gang med "kollektiv" som transportmiddel.
3. Herefter lader du turen køre i ca. 5 min. (hvis det er muligt, må telefonen meget gerne ligge i en vindueskarm el.lign. for at sikre den får forbindelse til GPS-satellitterne)
4. Efter ca. 5 min. stopper du turen, vælger kategorien "andet", skriver "test" i tekstfeltet og trykker på ok.

Det vil give os mulighed for at lave eventuel fejlsøgning mandag aften, så vi får så mange gode data som overhovedet muligt i undersøgelsesperioden.

Da dette betyder lidt mere arbejde for dig, har vi derfor valgt at gavekortet for deltagelse denne gang er på 200kr.

Hvis du er i tvivl om, hvad du skal, eller har du spørgsmål er, er du selvfølgelig altid velkommen til at sende os en mail eller kontakte os på 22 62 08 86 (Michael) eller 22 26 56 88 (Tino).

Nogen har efterspurgt mere information om hvad denne type undersøgelser bliver brugt til så her følger en række links til steder du kan læse mere om projekter hvor GPS-tracking er blevet brugt:
http://www.detmangfoldigebyrum.dk/
http://vbn.aau.dk/files/57642770/vollsmose_A4_final.pdf
http://vbn.aau.dk/files/55673676/Bike_GPS_Survey.pdf

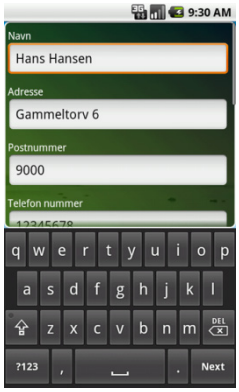
**Herunder følger nogle mere udførlige instruktioner i brugen af app'en.**

# Instruktioner

1. **Afinstaller den ”gamle” version af App’en**
    - Dette gøres ved at gå ind i Indstillinger -> Programmer -> Programadministration. Her finder du programmet ”ParticiMap Viborg” og trykker på det. Derefter trykker du på afinstaller.

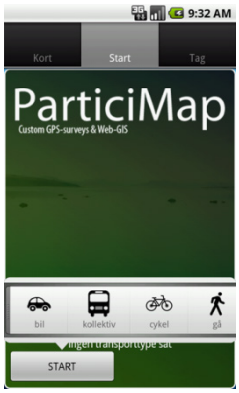
2. **Installation af ny app**
    - Dette gøres ved at gå ind på http://app-viborg.particimap.dk i telefonens internetbrowser, eller ved at scanne nedenstående QR-kode. Det er vigtigt at have givet tilladelse til at installere ikke-Market-programmer. Dette gøres ved at gå ind i indstillinger->Programmer og sætte flueben ved "Ukendte kilder".

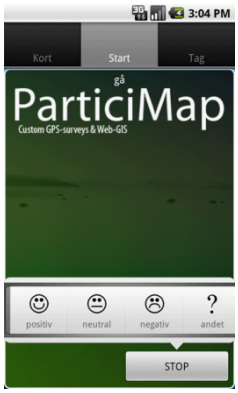
3. **Indtast dine brugeroplysninger**
    - Da vi skal undgå at offentliggøre data om din bopæl, er det vigtigt at adressen er indtastet korrekt. I feltet ”Vejnavn” skal der kun indtastes vejnavnet, og i feltet ”husnummer” må der kun indtastes nummeret, og altså ikke etage eller side, hvis du bor i etageejendom.
    - **VIGTIGT:** Det er meget vigtigt, at du har internetforbindelse på telefonen, når du opretter en bruger. Du skal derfor have enten have 3G data slået til eller være forbundet til et WIFI netværk.

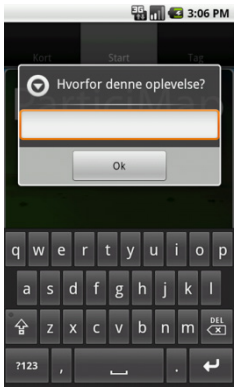


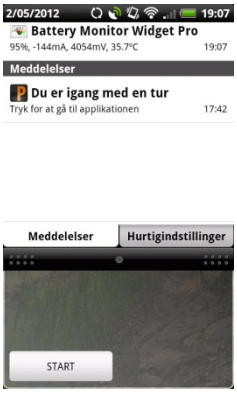
## 4. Start og stop af tur

- Når du bruger applikationen, er det nødvendigt at have tilladt brugen af GPS i telefonens indstillinger. Hvis der ikke er givet tilladelse til dette, vil der komme en advarsel med muligheden for at gå direkte til der, hvor det indstilles. For at komme tilbage til ParticiMap applikationen bruges telefonens "back" knap.
- Tryk på start-knappen for at starte en tur. Herefter vælges hvilken transportform du benytter.
- Mens en tur er i gang kan du bruge din telefon som normalt.
- Når du slutter en tur trykkes på stop-knappen og der vælges den kategori, der passer bedst på dine følelser omkring turen. Nu kan du skrive yderligere begrundelse for, hvorfor du valgte netop denne kategori.
- Når der startes eller stoppes en tur vil der blev lavet en notifikation. Den vil fortælle om, hvilken slags tur du har i gang, eller hvornår der sidst er afsluttet en tur. Du kan trykke på denne for at gå direkte til applikationen.



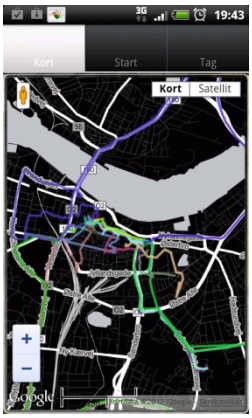



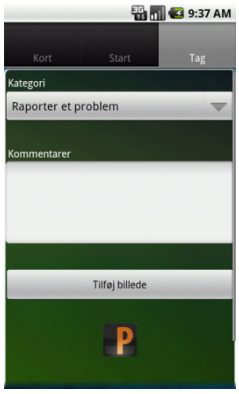




## 5. Se dine ture

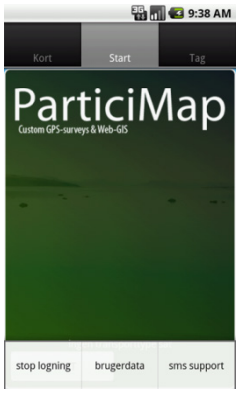
- Under fanebladet "Kort" kan du se de ture du har registreret. Der vil først blive vist en tur når den er afsluttet og mobiltelefonen har haft internetforbindelse i en periode der er lang nok til at sende informationen om turen.



6. **Tag**
    - Ser du noget som er interessant så send det til os i et tag. Det kunne eksempelvis være ting ved din by som du synes er rigtig gode, eller noget som du synes er grimt eller irriterende.
    - Du kan vælge at sende tekst, billede eller begge dele.
    - For at tage et billede trykkes på knappen ”Tilføj billede”, hvorefter kameraet starter og du kan tage et billede som normalt.
    - **NB:** Det kræver internetforbindelse at indsende et ”tag”.




7. **Menu**
    - Ved at trykke på telefonens menu-knap kan du vælge mellem at ændre i dine brugerdata, kontakte support via sms eller at stoppe logningen.
    - Stop logning skal kun bruges i tilfælde af, at du ønsker at stoppe din deltagelse i undersøgelsen eller når undersøgelsen er afsluttet.



# Appendix I

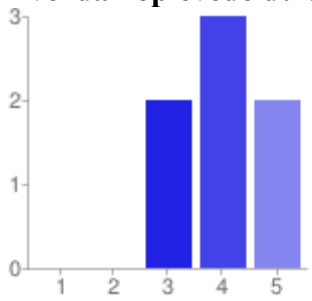
# Questionnaire from test two

# 7 responses

## Summary See complete responses

### Overordnet indtryk

**Hvordan oplevede du det at bruge applikationen?**



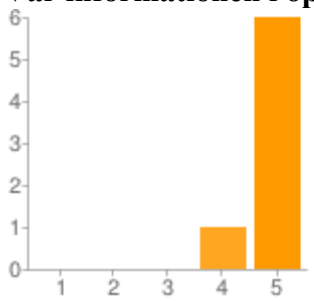

| | | | |
|---|---|---|---|
| 1 - | Dårligt | **0** | 0% |
| 2 | | **0** | 0% |
| 3 | | **2** | 29% |
| 4 | | **3** | 43% |
| 5 - | Godt | **2** | 29% |

**Hvorfor denne oplevelse?**

Den virkede udemærket. Dog lukkede den ned nogle gange, pga. en fejl.Fordi den virker meget godt, men den bruger dog også meget strøm da GPS er tændt. Kunne være godt hvis i sagde at man godt kunne slukke gps efter man var færdig, men det kan man vel tænke sig til. Der var lidt problemer med app'en.Der var absolut ingen problemer denne gang.Fin app men da i ikke har kunne modtage mine data har det ikke virket optimalt .Den var let og simpel at bruge, men igen for mit vedkommende opstod der nogle problemer. Det var nogle lidt andre problemer end sidst, da alt tilsyneladende for mit vedkommend...

### Indledende information

**Var informationen i opstartsmailen tilstrækkelig og nem at forstå?**



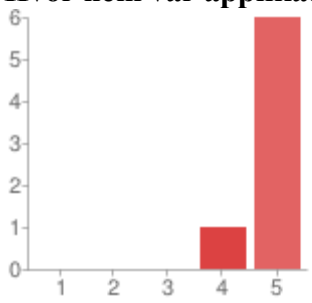

| | | | |
|---|---|---|---|
| 1 - | Dårlig | **0** | 0% |
| 2 | | **0** | 0% |
| 3 | | **0** | 0% |
| 4 | | **1** | 14% |
| 5 - | God | **6** | 86% |

**Hvad kunne der gøres for at forbedre informationsmaterialet?**

Det var godt og behøver ingen forbedringer Der stod hvad man skulle bruge, og godt beskrevet. ikke noget Det var fint at forståDer kunne ikke gøres mere. Denne gang fik jeg al den information, jeg ønskede.Ikke noget :-)Ikke noget, alt var nemt at forstå

## Installation af applikationen

**Hvor nem var applikationen at installere?**




| | | | |
|---|---|---|---|
| 1 - | Besværlig | **0** | 0% |
| 2 | | **0** | 0% |
| 3 | | **0** | 0% |
| 4 | | **1** | 14% |
| 5 - | Nem | **6** | 86% |

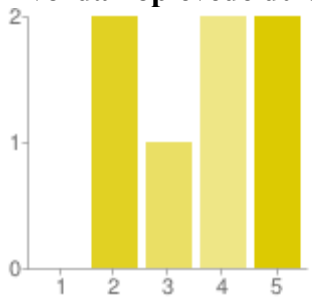
**Hvad kunne være anderledes?**

Ingenting Det kan vidst ikke gøres på en nemmere måde. Scan, installere og man er på.den var meget nem, så der kunne ikke være noget anderledesDen var utroligt nemt at installereAlt var fint.Ikke noget, den er dejlig enkel at benytte .Igen ingenting, det var meget simpelt og som at installere andre app's.

---

## Start/stop af tur

**Hvordan oplevede du det at skulle starte og stoppe ture?**




| | | | |
|---|---|---|---|
| 1 - | Dårligt | **0** | 0% |
| 2 | | **2** | 29% |
| 3 | | **1** | 14% |
| 4 | | **2** | 29% |
| 5 - | Godt | **2** | 29% |

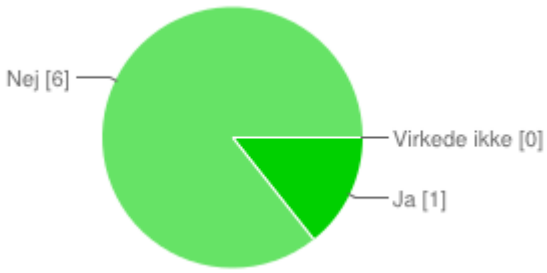
**Hvad kunne være anderledes?**

Der kunne muligvis være en forventet sluttids knap som giver lyd fra sig når man har sat tidspunktet, da man glemmer at man har startet en tur Der opstod fejl, som lukkede programmet ned engang imellem ved start og stop. Når man åbnede app'en igen, startede den der hvor den gik ned. jeg glemte dog at stoppe den enkelte gange. Det var fint, men der var lidt problemer med app'enDer kunne ikke gøres noget anderledes.Nogle gange måtte jeg slukke min mobil fordi programmet "frøs fast". Min telefon var igen lidt langsom, men dette havde intet at gøre med appen.

---

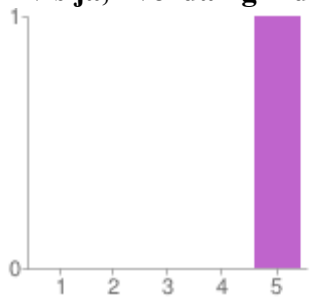
## Tag

**Fik du prøvet at tagge?**




| | | |
|---|---|---|
| Ja | **1** | 14% |
| Nej | **6** | 86% |
| Virkede ikke | **0** | 0% |

**Hvis ja, hvordan gik det?**




| | | | |
|---|---|---|---|
| 1 - | Dårligt | **0** | 0% |
| 2 | | **0** | 0% |
| 3 | | **0** | 0% |
| 4 | | **0** | 0% |
| 5 - | Godt | **1** | 14% |

**Hvad kunne være anderledes?**

Jeg taggede både, og sendte et billede ind, og det kunne ikke være bedre :)

## Menu

**Prøvede du nogle af funktionerne i menuen?**

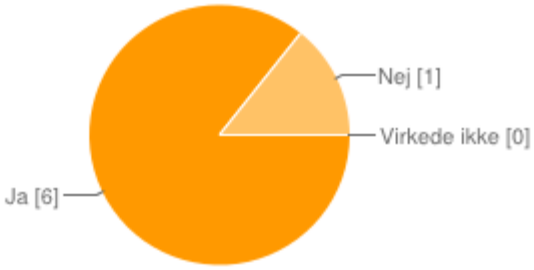
| | | |
|---|---|---|
| SMS support | **0** | 0% |
| Brugerdata | **3** | 100% |
| Stop logning | **1** | 33% |

People may select more than one checkbox, so percentages may add up to more than 100%.

**Hvad kunne være anderledes?**

## Visning af dine spor i applikationen

**Prøvede du at se kortet på mobiltelefonen?**



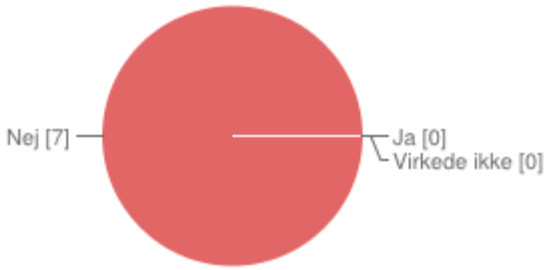

| | | |
|---|---|---|
| Ja | **6** | 86% |
| Nej | **1** | 14% |
| Virkede ikke | **0** | 0% |

**Hvad kunne være anderledes?**

At det var tydeligere Det virkede meget god, men måske en smule overflødigt da jeg ikke vidste hvad den skal bruges til.det var fintKortet var fint og det blev lettere at finde rundt i éns ture da de var markeret med forskellige farver.

## Visualiseringsplatformen

**Prøvede du kortplatformen på internettet?**




| | | |
|---|---|---|
| Ja | **0** | 0% |
| Nej | **7** | 100% |
| Virkede ikke | **0** | 0% |

**Hvad kunne være anderledes?**
ved ikke hvad det er

---

## Øvrige kommentarer

**Har du yderligere kommentarer?**
Man glemmer at man skal starte og stoppe sine ture, så muligvis noget der husker en på det. Dog rigtigt god opdatering i har lavet. Det er en fin app.Det fungerer perfekt og informationen i startmailen var til stor hjælp.

---