

# BACHELOR THESIS

# inPark

Wireless parking system

by

(13) - Erik Bjaanes

(1) - Per Ø. Olset

(11) - Torbjørn Årdal

Bachelor thesis

EL2-305

May 2015

## Reference page

<b>TITLE</b> inPark	<b>REPORTNO.</b> 01	<b>DATE</b> 22.05.2015
<b>PROJECT TITLE</b> Project in HO2-300 Bachelor assignment	<b>ACCESSABILIT</b> Open	<b>PAGES</b> 61
<b>AUTHORS</b> Erik Bjaanes Per Øyvind Olset Torbjørn Årdal	<b>GUIDANCE COUNCELLOR</b> Rune Viken	
<b>EMPLOYER</b> INTIN AS		
<b>SAMANDRAG</b> I denne bachelor-avhandlinga ser vi på korleis vi kan levere informasjon til elbil-brukarar om ladestasjonar er ledige eller ikkje, i sanntid. Vi vurderer kva effektar eit sånt system kan ha på miljøet, og trafikkflyten. Avhandlinga er eit resultat av ei oppgåve som vart presentert av INTIN AS. Oppgåva vår var å implementere eit system som overvakar ledigheta til fleire ladepunkt på eit spesifikt test-område i sanntid. Vi skulle også utvikle ein applikasjon for mobile einingar som hentar denne informasjonen, og presenterer den til brukaren.		
<b>SUMMARY</b> In this bachelor thesis, we look at how you can provide information to users of electric cars about the availability of charging stations in real-time. We consider what effects such a solution might have on the environment, and traffic flow. This thesis is the result of an assignment presented by INTIN AS. Our task was to implement a system that monitors the availability of several charging points, located on a designated test-site, in real-time. We would also develop an application for mobile devices that gathers and presents this data to the user.		
<b>KEYWORDS</b> EL2-305 Bachelor, Android application, charging station, sensors, INTIN AS, Nedap, data collector, relay nodes, Sensit, inPark.		

## Foreword

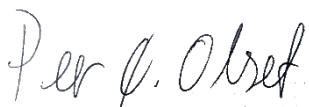
This thesis is the result of our group's three years study at Sogn og Fjordane University College. We chose this assignment because we found it very interesting, and it involved aspects that each member is fond of. This includes programming, installment at test-site, new technology and the possibility to do something innovative.

In order to complete the assignment, we have had to rely on help from some people. We would like show our appreciation and gratitude to Geir Henning Nore and Rune Viken, the founders of INTIN AS, for providing this interesting assignment and being available whenever we needed some guidance. We would also like to thank Nedap for their support, and especially Mr. Gerjo Tannemaat. Without him we would have used a lot more time figuring out certain parts of the user interface. As well as tirelessly answered our questions with rapid and informative responses. Thanks to Eivind Standnes for aiding us during the installation at Førde Central Hospital. Additionally we would like to thank our friends and family for their continuously support and patience with us during this period. Last, but not least, we want to thank Joar Sande for being part of the project management group and being available to answer questions along the way.

This group has consisted of Erik Bjaanes, Per Øyvind Olset and Torbjørn Årdal.



Erik Bjaanes



Per Øyvind Olset



Torbjørn Årdal

Place/date:

Førde, 22.05.2015

## Table of Contents

<b>FOREWORD</b>	<b>3</b>
<b>TABLE OF CONTENTS</b>	<b>4</b>
<b>1. ABSTRACT</b>	<b>6</b>
<b>2. INTRODUCTION</b>	<b>7</b>
<b>3. THE INPARK SYSTEM</b>	<b>9</b>
3.1. CHOICE OF SOLUTION	9
3.2. TEST-SITE	9
3.3. THE NEDAP SENSIT SYSTEM	10
3.3.1. EQUIPMENT	10
3.3.2. HOW THE SYSTEM WORKS	10
3.3.3. THE SENSORS	11
3.3.4. SENSIT FLUSH MOUNT	12
3.3.5. SENSIT IR	12
3.3.6. RELAY NODE 2G	13
3.3.7. DATA COLLECTOR	13
3.3.8. SERVER AND DATABASE	13
3.3.9. SYSTEM DOWNTIME	16
<b>4. INSTALLATION OF THE SENSOR SYSTEM</b>	<b>17</b>
4.1. PLANNING THE INSTALLATION	17
4.2. HEALTH AND SAFETY	18
4.3. INSTALLATION	19
<b>5. APPLICATION</b>	<b>21</b>
5.1. INITIAL PLANNING	21
5.2. FEATURES AND DESIGN	22
5.3. GETTING STARTED	23
5.4. COMMUNICATING WITH THE DATABASE	24
5.5. CREATING THE USER INTERFACE	27
5.6. THE MAP	27
5.7. THE LIST	31
5.8. THE SETTINGS	34
5.9. TYING THE SECTIONS TOGETHER	34
5.10. REFRESHING THE DATA	37
5.11. HOW FAR DID WE GET?	38
<b>6. ENVIRONMENT</b>	<b>42</b>
6.1. DISCUSSION ENVIRONMENTAL REWARDS	42
6.2. ENVIRONMENTAL RETURN ON INVESTMENT	45
6.3. BENEFICIARIES	48
<b>7. CONCLUSION</b>	<b>49</b>
<b>8. PROJECT ADMINISTRATION</b>	<b>50</b>
8.1. GROUP ADMINISTRATION	50

<b>8.2. PLANNING PHASE</b>	<b>51</b>
<b>8.3. TIME MANAGEMENT</b>	<b>52</b>
<b>8.4. RISK ASSESSMENT</b>	<b>54</b>
<b>8.5. ECONOMY</b>	<b>54</b>
<b>WEB PAGE</b>	<b>55</b>
<b>8.6. SELF ASSESSMENT</b>	<b>58</b>
<b><u>9. SOURCES</u></b>	<b><u>59</u></b>
<b><u>APPENDIX</u></b>	<b><u>61</u></b>

## 1. Abstract

In this bachelor thesis, we look at how you can provide users of electric cars with information about the availability of charging stations. We have named it the inPark-system. The system aims to be efficient, well made and easy to use. The thesis is a result of an assignment given by INTIN AS.

In addition to providing the required equipment, INTIN also made the arrangements to have a test-site available. This test-site is located at Førde Central Hospital, where we monitor six parking bays. We have developed an Android application that presents the status of the parking bays in real-time.

We also consider the environmental rewards that the system might provide. We believe that the system has the potential to make a large positive impact on the environment as the amount of electric cars in the world only continues to grow.

## 2. Introduction

This bachelor thesis is the final project of the three-year study of engineering at Sogn og Fjordane University College (HiSF). The bachelor thesis makes up for  $\frac{2}{3}$  of the last semester, and is worth twenty credits.

Our assignment was to implement and test a system that monitors the availability of charging stations for electric cars in real-time, and presents the relevant data to the end-user in an Android application on a smartphone. In this thesis, we will explain how we mounted a series of sensors at a designated test-site, how we stored the data in real-time and how we developed our own Android application to present it. We will also look at the potential rewards that such a solution can provide. Will this solution provide a higher turnover of charging stations? Will it improve traffic flow? Will it have a positive, and noticeable, impact on the environment? Will the solution provide better, and easier, access to charging stations?

Our employer for this assignment is a company called INTIN AS. INTIN was co-founded by Rune Viken and Geir Henning Nore in 2014, and is located in Førde, Norway. The company specializes in smart-home technology, but as the number of electric cars increases around the world, they would also like to help owners of electric cars to locate available charging stations quickly and easily. In January 2015, INTIN approached HiSF with a proposal for an assignment to create, and test, a prototype of such a system. All the members of our group found the assignment very interesting, and we quickly informed them that we would like to accept it.

For this assignment, we will be using six sensors to detect vehicles in six different parking bays (one sensor in each bay). Three of these sensors use both infrared and magnets, while the other three sensors only use magnets. Each parking bay will simulate one charging point, and we will be describing each parking bay as a charging point for the rest of the thesis. A Data Collector will monitor the state of the sensors via three Relay Nodes, and forward the data to a database for storage. Our Android application will then retrieve the data from the database, and present it in a

user-friendly way. Our test-site is a small car park at Førde Central Hospital, which visitors of the hospital use on a daily basis.

A Dutch company called Nedap Identification Systems<sup>1</sup> (Nedap) delivers the equipment that we use to monitor the charging stations. Nedap is the leading specialist in systems for long-range identification, wireless vehicle detection and city access control. Along with the equipment, Nedap have provided a database with a user interface so that we can manage and monitor all of the equipment.

As of 16<sup>th</sup> of May 2015, there are 1682 charging stations with 6464 charging points in Norway. Of the 6464 charging points, 5595 of them are public<sup>2</sup>, but only 462 deliver information about availability in real-time. When we refer to charging points in this thesis, we refer to the remaining 6,002 charging points that do not have real-time status information. By using the equipment at our disposal, we aim to aid INTIN in developing a cost-efficient solution to monitor the availability of charging points, and delivering the information to the end user in an efficient, and user-friendly way.

In this thesis we will refer to both charging stations and charging points. The difference is that charging stations is a collective term for its combined number of charging points.



### 3. The inPark system

The inPark system is a combination of Nedap's equipment, and our application. This chapter will explain in more detail how the equipment (sensors, Relay Nodes and Data Collector) is set up at the test-site. We will explain more about the Android application later in this thesis.

#### 3.1. Choice of solution

The choice to use Nedap's solution on this project was not our decision. When INTIN approached the school with this project, they had already decided which manufacturer to use. They informed us that they chose Nedap's system because they could deliver an entire reliable, high quality wireless system, including both the hardware and database, at a competitive price. They considered Rosim<sup>3</sup> and Tinynode<sup>4</sup>, but their systems did not come with a designated database and user interface.

#### 3.2. Test-site

In order to test the inPark-system in optimal conditions, we needed a test-site in a live environment. In other words, we needed to test the system at a car park that many people used on a daily basis. This would provide us with the most realistic results, and give us a good indicator on how stable the system would be. The original plan was to use the car park of HiSF as a test-site, but we ended up using six parking bays at Førde Central Hospital. This was a result of a dialogue between INTIN and the hospital where INTIN suggested to use them as a pilot with the possibility to create a solution tailored specifically for their needs. We agreed that this test-site would be better because it would most likely result in a higher turnover of users, and thus providing better test results. However, the hospital had to discuss the issue internally before we got permission. Our head of group attended a meeting with hospital representatives, along with INTIN, and presented our project to them, and how it could be used to help them. We got authorization to use six parking bays. Three of them were short-term parking bays (15 minutes), while the remaining three were bays meant for blood donors. Unfortunately, the process of getting this

permission had severely delayed the installation of the equipment, which in turn offset our planned schedule.

### 3.3. The Nedap Sensit system

An important part of our project was the communication between the sensor system developed by Nedap and their server. In this chapter, we will walk through in detail how the information from a car that just parked over a sensor travels through the system and how the end user receives the data in the application.

#### 3.3.1. Equipment

In order to carry out our experiment, INTIN provided us with the equipment listed in table 1.

Table 1: List of equipment.

Product	Quantity
Sensit Flush Mount	3
Sensit IR	3
Relay Node 2G	3
Data Collector	1

#### 3.3.2. How the system works

The system consists of six sensors, three Relay Nodes, a Data Collector and a database. The Data Collector retrieves the status for all the sensors at regular intervals via the Relay Nodes, and forwards that data to the database for storage. The system is wireless, with the only exception that the Data Collector needs an Internet connection, and a power supply. The protocol used for the wireless communication is proprietary, and developed by Nedap. This means that you cannot implement a third party sensor.

When you activate a sensor, the Data Collector will immediately start retrieving its status. If some sensors are out of range from the Relay Nodes, they are able to use

other sensors to relay their status. This type of communication is called mesh networking<sup>5</sup>, and the system utilizes a full mesh topology.

In this system, the Relay Node has a more active role in relaying information, so the sensors can save energy. The communication frequency between the nodes is 868MHz band in Europe and in the 915MHz band in the US (see Appendix 2). When the sensors and Relay Nodes relays the status to the Data Collector, the Data Collector proceeds to upload the current status information over the Internet on port 11111 to a server in Netherlands, which is managed by Nedap. Information on how to connect was provided in a document from Nedap to INTIN. You can see this document in Appendix 1. The Data Collectors' installation site must have that port opened in order for the communication to work. When the status is online, the owners of the installation site can log on to their dedicated UI on Nedap's server and monitor their car park.

### **3.3.3. The sensors**

We are using two types of sensors from Nedap. One is a flat, cylindrical, purely magnetic sensor made for outside areas (figure 1), and the other is a slightly elevated cylindrical sensor with both magnetic and IR sensors (figure 2).

Both sensors have a 85mm diameter, and are made of black friction welded plastic, making them completely water and dustproof. The sensors are completely maintenance free due to the friction weld. If a sensor runs out of battery or otherwise stops working properly, you replace the whole sensor by drilling it up and replacing it with a new sensor.

The only tools needed to install this system are a flat-blade screwdriver, cutting pliers and a capable drill with a 85mm drill bit. You will also need an adhesive to hold the sensor in place. We used concrete mortar, based on Nedap's recommendation. Calibration of the system is necessary after installation. Calibration is done on the server, with the option to calibrate based on bay, zone or entire car park. You can calibrate IR sensitivity and magnetic threshold. The whole parking lot must be empty when performing the calibration, as the magnetic sensors will detect any metal

objects within its magnetic field after installation. After calibration, these objects will no longer trigger the sensor, so metallic grates or concrete armature is of no concern when installing this system. In addition, The IR sensor can be calibrated to not detect roofs and similar. Recalibration of an individual sensor is possible if needed.

#### 3.3.4. Sensit Flush Mount

The Flush Mount is a magnetic, wireless sensor meant for outdoor use, and it sits flush with the surface. This gives it an advantage on car parks that are machine plowed, as there is no risk of demounting or breaking the sensor when plowing due to the sensor being fully submerged.



Figure 1: Sensit Flush Mount

#### 3.3.5. Sensit IR

The IR sensor is both magnetic and infrared, and like the Flush Mount, it is wireless. It has a built in Ice-Detect, which activates if the infrared sensor is triggered ten times in a row without triggering on the magnet. It will then stop relying on the infrared to determine if there is a vehicle parked above it until the IR sensor are cleared. In short, in case of ice or snow the sensor will work the same way as the Flush Mount. If the sensors' ice detection triggers, the system raises an alert in the dashboard. Due to winter conditions in Norway, this is best utilized where snowplows has a rubber edge, or simply where snow is not an issue. A steel edge would wreck the sensor.



Figure 2: Sensit IR

Nedap also delivers a sensor type meant for indoor car parks, called Surface Mount. It features both IR and magnetic detection like the Sensit IR, but is flatter and has a larger diameter than the Flush Mount and IR. The sensor is installed by gluing it to the surface.



Figure 3: Sensit Surface Mount

### 3.3.6. Relay Node 2G

The Relay Node is what the name implies. A node purely relays information between the sensors and Data Collector. It communicates wirelessly like the sensors, and is mainly used for on-street applications. It is usually mounted on lamppost. Without the dish attached it will detect signals in a 360° angle with approximately a 25-meter radius with no interfering obstacles. With the dish however, you can concentrate the detection to 180° angle in front of the Relay Node, extending the range to about a 60 meters reach with no interfering obstacles. The Relay Node has a more active role in the mesh network in order to ensure longer lifetime for the sensors.

As opposed to the sensors, you can change batteries in the Relay Node.



Figure 4: Relay Node 2G

### 3.3.7. Data Collector

The Data Collector is the only component in the system that is not wireless. It requires an Ethernet connection to the Internet as well as a power supply. It receives information from the Relay Nodes and sensors and uploads it to the server. Even though the nodes have 35-65m ranges, the data collector has to be within a 15m range to the nearest node and/or sensors in order to be functional.



Figure 5: Data Collector

### 3.3.8. Server and database

Along with the equipment, Nedap provided us with a database on their own server in Holland that they would maintain themselves. They also provided us with a website that contained a user interface so that we could access and manage the database ourselves. The user interface consists of a dashboard where we have an overview of the current sensor status (figure 6). On 13<sup>th</sup> of May, the system went through an update, which reset the server uptime. The real uptime is about five months.

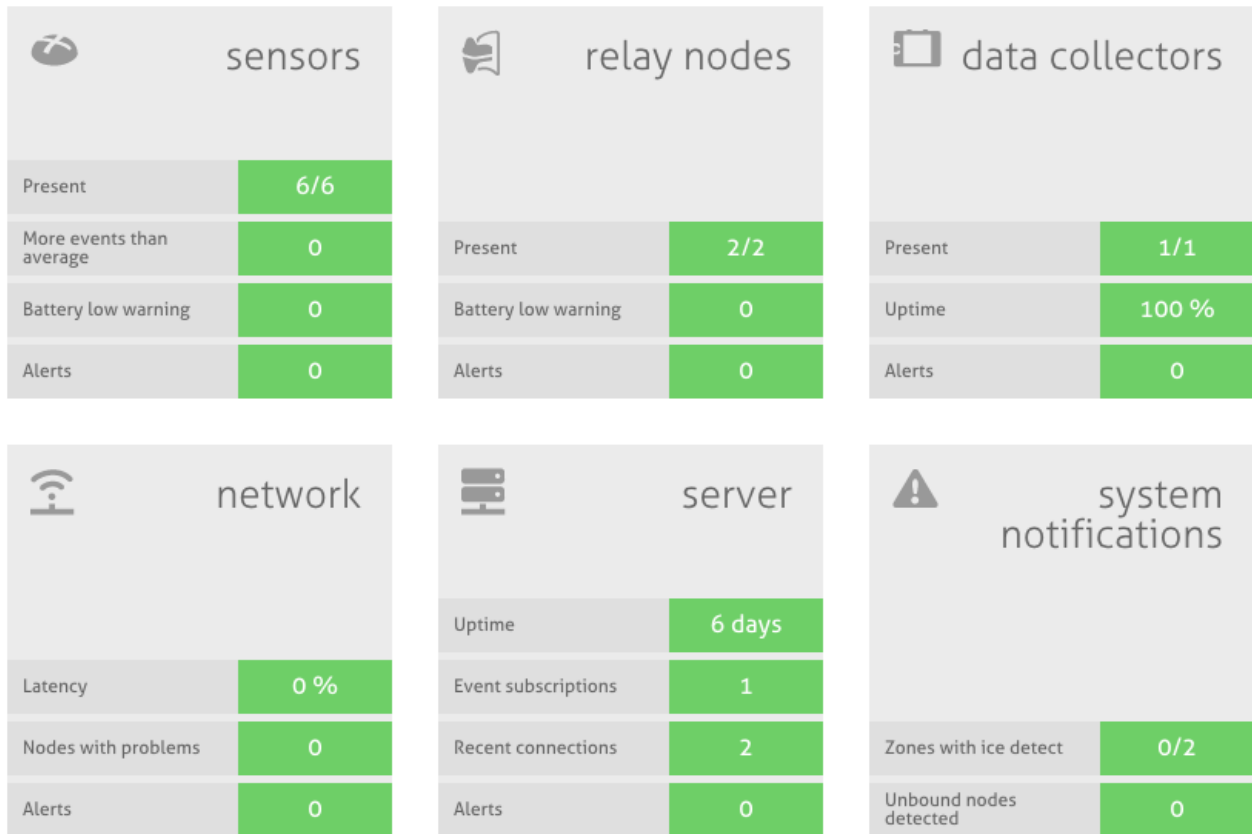


Figure 6: Nedap User Interface Dashboard

<b>Detection</b>
<b>Definition</b>
<b>Monitor</b>
Occupancy report
Occupancy list
Occupancy chart
Overstay report
Parking violation probability
Map monitor
<b>Diagnose</b>
<b>Settings</b>
<b>Documentation</b>

Figure 7: List view of monitoring options

In figure 6, you can see three tabs at the top called “Dashboard”, “Parking Enforcement” and “Settings”. We have not used the Parking Enforcement-section at all, but configuring the system required using the Settings-section quite a lot. The Settings-section is where we did things like defining each node of the system, in accordance with Nedap’s input requirements, calibrate sensors and setting overstay regulations. In figure 7, you can see our monitoring options in Settings-section.

In addition to configuring the system, the user interface also allowed us to monitor our test-site by using “Map monitor”. In figure 8, you can see each parking bay connected to the database as illustrated by a rectangle with a signal in it. This signal has three different states. Teal color means vacant, red color means occupied and if orange color means that the car has been there too long (overstay). We received the map from the hospital, and forwarded it to Nedap who inserted the bays and signal.

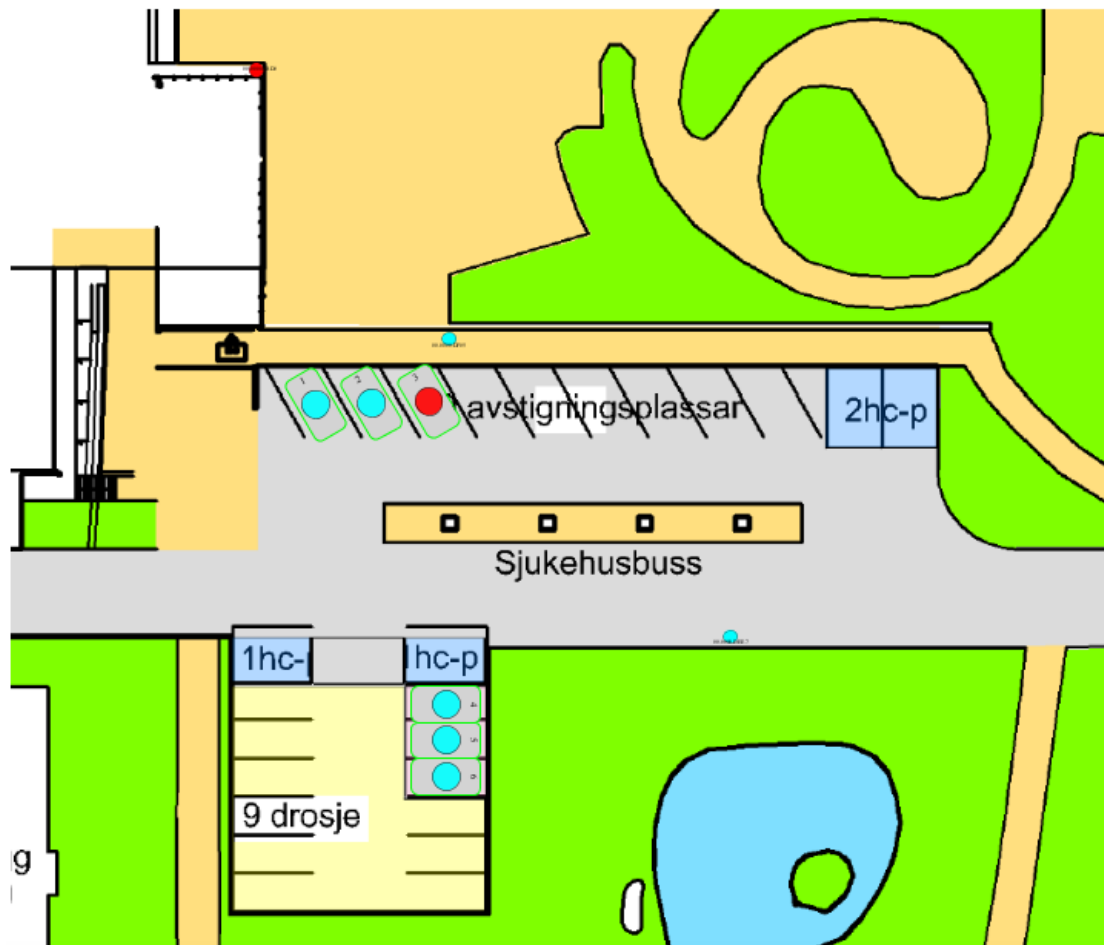


Figure 8: Map monitor. Graphical view of occupancy-status of each bay at test-site.

In order to register the sensors in the database, we had to create and upload a .csv file (Comma Separated Values) to the database. In this file, we had to assign Parking Lot ID, Zone ID, Bay ID, the serial number, and the GPS-coordinates, of the node in that order. You can see in table 2 how we had to write down the information. In figure 9 you can see how it looked when uploaded to the server.

Table 2: Comma Separated Values (CSV) used for defining the nodes on the server

```
1,1,1,00.0000.E6E8,61.457192,5.888865,15.0
1,1,2,00.0000.E6E0,61.457192,5.888865,15.0
1,1,3,00.0000.E6CA,61.457192,5.888865,15.0
1,2,4,00.0000.E48F,61.457192,5.888865,15.0
1,2,5,00.0000.E49A,61.457192,5.888865,15.0
1,2,6,00.0000.E498,61.457192,5.888865,15.0
1,DC,8,00.0000.8BD0,61.457192,5.888865,15.0
1,RN,9,00.0000.DF05,61.457192,5.888865,15.0
1,RN,10,00.0000.DEE7,61.457192,5.888865,15.0
```

	Node type	Parking lot ID	Zone ID	Bay/Place ID	Node ID	GPS
✓	Data collector	1		8	00.0000.8BD0	📍
✓	Relay node	1		9	00.0000.DF05	📍
✓	Relay node	1		10	00.0000.DEE7	📍
✓	Sensit	1	1	1	00.0000.E6E8	📍
✓	Sensit	1	1	2	00.0000.E6E0	📍
✓	Sensit	1	1	3	00.0000.E6CA	📍
✓	Sensit	1	2	4	00.0000.E48F	📍
✓	Sensit	1	2	5	00.0000.E49A	📍
✓	Sensit	1	2	6	00.0000.E498	📍

Figure 9: View of the node-list on the server

### 3.3.9. System downtime

There were times during this project that we did not visit the dashboard to monitor the system for several days. One of these times we were met with a dashboard with no nodes attached. Data Collector, Relay Nodes and sensors – all absent. We had noticed that sometimes the Data Collector went absent for no reason at all, but came back online by itself within a minute. When signed in to the server, we have a report stating when the Data Collector lost connection and when it connected again. This time it had been absent for close to six days, for no apparent reason. We contacted Rune Viken with the problem. He then called the hospital’s IT-department. They could confirm that there had been a glitch in their firewall, causing the Data Collectors port forwarding to be invoked and reset it to be blocked for outgoing connections. Luckily they were able to mend the problem within the hour, putting the system online again.



## 4. Installation of the sensor system

At first we were supposed to install the sensor system at six parking slots at HiSF, but INTIN were in contact with Førde Central Hospital when the equipment arrived. It would provide an opportunity for INTIN to get a customer and deliver an entire system to the hospital if the equipment worked satisfactory. This delayed the initial startup of our project, as both INTIN and our group had to wait for authorization from the hospital to commence with the installation.

The choice of test-site serves multiple purposes. For INTIN it was a pilot project provided free of charge for the hospital, so they both could get a preview of what the system can do. There was also a consideration to the possibility of expansion if the hospital wishes to implement it for their entire car park. For the hospital it could be a possible solution for a problem with overstay at short-term parking and authentication for the blood donor-spots. For us it was simply to provide data we could use during testing and developing the application.

In comparison, this test-site is much more purposeful than the original plan at HiSF, which were not likely to amount to any expansion at all.

### 4.1. Planning the installation

Nedap provided the installation procedure along with the sensors. This made our planning of the installation easier. All we needed were a drill capable of drilling tarmac and concrete, and a drill bit with the correct diameter, which had to be a bit larger than the circumference of the sensors in order to have room for an adhesive. Fortunately, the hospital had a drill we could borrow, and INTIN rented a drill bit with the right diameter from a local company called Ramirent.



Figure 10: Installation in progress

We also had to check if there were cables or heating elements present at the locations we were going to drill on. The holes were required to be approximately eight centimeters deep. Eivind Standnes, the hospital's custodian, assured us that it was safe to make these holes without the risk of drilling through something at that depth. He explained that every cable buried in their car park is buried deeper than we needed to go.

At the beginning of our project, we thought the installation and troubleshooting would take longer than it actually did. We anticipated that it would take about three weeks in total. In reality the installation took three days, and troubleshooting took a few hours in between other tasks when a problem presented itself.

#### **4.2. Health and safety**

Due to a rather easy installation process, health and safety regulations did not come in to play. We used a stepladder that is within the maximum height allowed without extra safety precautions, as we did not need to have a higher reach than approximately 3 meters. The maximum allowed height without securing the platform is 4 meters<sup>6</sup>. However, we were always two men when using the ladder - one to make sure it was steady while the other mounted the Relay Nodes - to reduce any risk of falling.

The drill was bolted to a concrete block with 2 cm diameter bolts. The drill had to be moved about with a small front loader. This secured a stable drilling platform. Coolant (water) was supplied to ensure the drill stayed cool during operation. The drill was a no-splinting type that mostly used friction to cut through the tarmac and concrete, so no safety goggles were required. We were also given notice beforehand from Eivind Standnes that hearing-protection would not be needed, given that the volume the drill produced is not sufficient enough to provide any hearing damage for the amount of time we would be exposed to it.<sup>7</sup>

We did not get an accurate number of decibels produced by the drill, but it was quiet enough to carry out conversation when drilling.

### 4.3. Installation

Installation of the system started on 30<sup>th</sup> of March. Eivind Standnes gave us an introduction to the drill and how we could use it safely to drill the necessary 88mm holes for our sensors. He also helped us move the drill with the hospital's front loader. All we had to do was to drill the holes, wait for the excess coolant to drain away, pour the adhesive into the hole and insert the sensors.



Figure 11: Drilling hole for sensor

We used rapidly drying cement as the adhesive, which normally takes 15-25 minutes to solidify. Once the sensors were submerged in the cement, they had to be weighed down in order to stop them from floating up before the cement solidified. We gave the cement some time before we cleaned the excess cement and water trails away.



Figure 12: Weighed down sensor

Due to low air temperatures that day, we gave the cement over night to make absolutely sure it had hardened. We came back the next day to mount the Relay Nodes and Data Collector, and clean up the rest of the cement on and around the sensors.



Figure 13: Installed Sensit IR

The second day of the installation started with installing the Relay Nodes. As mentioned we had three available, but we only used two of them. The reason being that we thought the Relay Nodes was already in range to the Data Collector. Which would mean that there was no need for



Figure 14: Installed Sensit Flush Mount

extending the signal with another Relay Node. It should be noted that Nedap recommended using at least two Relay Nodes.



Figure 15: Mounted Relay Node, with dish

The last thing we did was to install the Data Collector inside the hospital. The IT department had already set up a dedicated Internet socket for us due to the need for the firewall exception for the required communication port. Unfortunately, the dedicated socket was located too far inside to receive the signal from the closest Relay Node. We contacted the IT department to help us out in the matter.



Figure 16: Installed Data Collector

They investigated the possibility to relocate the connection to another socket via the connected switch, and luckily they quickly found a closer socket for us to use that also had a vacant power socket. The new socket put the Data Collector within proximity to the Relay Nodes.

After the installation we found that one of the Relay Nodes had to be relocated. It did not receive communication from the sensors properly because of its placement. After relocating it and rebooting the entire system, communication between all nodes responded as intended.

## 5. Application

### 5.1. Initial planning

When we chose this assignment, we knew that it involved creating a prototype Android application in order to test the system. We decided that we wanted to design the application to be as robust, and as close to a complete, launch-ready product as possible. That meant making it able to, in theory, monitor not only the sensors we mounted at Førde Central Hospital, but also monitor many more sensors at other car parks. We wanted to do it this way because we believed that this would give us the best data on what impact the system would have on the environment, and the everyday life of electric car owners.

We immediately decided that we should have a meeting with INTIN before we started any kind of planning of the application. We wanted to find out more about what thoughts, ideas and expectations they had about it, and the system as a whole, because that would help us to form a baseline to start from. At the meeting, INTIN was very helpful with explaining what they wanted from the application. The most important thing was that the application should have a map that showed several charging stations, and whether or not it was free. They also believed it to be a good idea to show all the charging stations in a list, and if you clicked on one of the charging stations in that list, the application would show you a picture, and more detailed information about it. As a side note, they advised us to divide the features that we wanted to put into the application into categories: “what do we need to have”, “what should we have” and “what would be cool to have”. They believed that such a system would make it easier to plan the application, and would help prevent us from spending too much time working on a feature that was not very necessary. The meeting was very productive, and we ended up adopting all of their ideas, though we made some minor adjustments along the way.

Shortly after this meeting, our lead programmer had to leave for an internship, which halted programming the application. The internship lasted six weeks, and during this time the two remaining members worked mostly on the website, and

getting the equipment that we received from Nedap ready for installment at Førde Central Hospital. One of the two remaining group members also had an internship, and used some of the days for this. The remaining member had attend some additional classes during this time. When our main programmer got back from his internship, we could get back to working on the application.

## 5.2. Features and design

We immediately had a group meeting to design the application in as much detail as we could. We started by making a list with all the features that we wanted to implement. See table 3. When we made this, we decided to adopt the category-system that INTIN recommended us, but with different categories. We categorized the features as **primary** and **bonus** features. A primary feature is a feature that is essential to the application, and required to make the application work in a user-friendly way. A bonus feature is a feature we would really like to implement, but is not essential to the user experience as a whole. In other words, if you were to remove a primary feature from the application, you would **significantly** lower the user experience. If you removed a bonus feature, it would not affect the application nearly as much.

Table 3:Simplified list of features and their priority

Feature	Feature type
Map <ul style="list-style-type: none"> <li>• Marker for each car park at geographical location, with info</li> <li>• Driving directions with a drawn path to each car park</li> <li>• The ability to search for a location</li> </ul>	Primary
List of available car parks with picture and relevant info	Primary
Settings-section where the user can make relevant adjustments	Primary
Our own database that contains detailed info about each car park	Bonus
The ability to mark car parks as favorites for easier access	Bonus
Route planner so the user can easily plan a long trip	Bonus

After agreeing on the list of features, we started designing the user interface of the application. From the list of features, we saw that there were three primary

features, and each of them would have to cover the entire screen of the smartphone when they were active. For example, it would not result in a good user experience if the map only covered half the screen. Because of this, we had to divide the user interface in three sections, and find a way for the user to be able to switch between them as easily as possible. We will explain how we solved this problem in more detail later.

When discussing design, we all agreed that we wanted to make the application as user friendly as possible. One of the key elements of this is to reduce the amount of clicks you have to do in order to go to the screen or option you want. We did not want to make a “maze of menus”, which would ruin the user experience.

### **5.3. Getting started**

We decided to use a program called Android Studio<sup>8</sup> to develop our application. Android Studio is a program that Google themselves has developed and designed specifically for the development of Android applications. Our lead developer already had a bit of experience with it so it was the clever choice in regards to playing on our strengths.

One of the things that were very important to us was to be able to synchronize the code automatically between the group members. If one person changed some code in a file, the other people on the group should get that change as well. Our solution was to upload the entire project to GitHub<sup>9</sup>. GitHub is a web-based hosting service where developers can upload their projects, invite other developers to download the code, and collaborate. The code is stored on the GitHub servers, and when one developer uploads a change in the code, the other developers will get that change when they perform a simple synchronization. GitHub makes it very easy for multiple people to work on the same project, and the same code, without having to synchronize the code manually. Another great advantage of having the project on GitHub is that if you are on a new computer (but you want to get some developing done), you can simply log in to your account, download the code, and start working.

When you are done, you just upload the updated code to GitHub, and delete it from that computer if needed.

The first thing we did when we started developing the application was to make a foundation. We created a new project in Android Studio, created the most necessary files to get a good structure, and started to construct the layout that we had designed in a very basic form. When we got the project started with the basic files, we uploaded it to GitHub so that we could work with the code separately, but still be able to synchronize it. We had divided the tasks between us before we started, so after we uploaded the project to GitHub, everyone could just start with their own tasks.

#### **5.4. Communicating with the database**

One of the first tasks we started on was making the application able to contact the Nedap database, and retrieve the sensor data. We prioritized this because we wanted to get the data as quickly as possible so that we could display it, and see how it turned out in the layout. It would then be easier to make the correct adjustments to the layout to make it just right.

In addition to supplying us with a database, Nedap had also made a REST API that we could use to retrieve the data from it. REST stands for Representational State Transfer, and is an architecture style for designing networked applications<sup>10</sup>. REST is a client/server architecture, and usually designed to use HTTP (Hypertext Transfer Protocol)<sup>11</sup> requests to create, change, read or remove data that is stored in a database on a given server. API stands for Application Programming Interface<sup>12,13</sup>, and is collection of rules and protocols that programs can use to communicate with each other. In other words, it is what makes our application able to interact with Nedap's database, without having direct access to it. The API essentially acts like a mediator between our application and their database. For example, if our application wants to get the status of the sensors, it asks the API "give me the status of the sensors". The API then takes care of asking the database for the relevant data, and delivering that data back to the application. See figure 17.



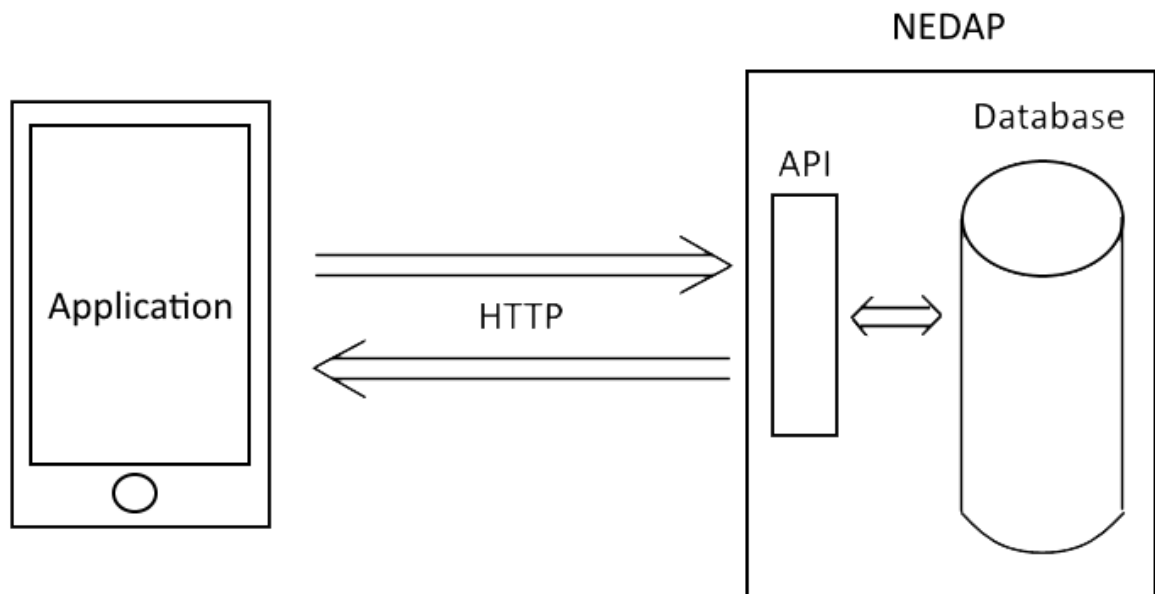


Figure 17: Communication with the database

As mentioned previously, REST usually use HTTP requests to create, update, read or remove data from the database. Nedaps database was no exception, and therefore we had to program our application to be able to send such requests to Nedaps server in order to get the data we needed.

A HTTP request consists of the following elements: an URI (Uniform Resource Identifier)<sup>14</sup>, a method<sup>15</sup> and one, or several, headers<sup>16</sup>. An URI is what identifies what resource the client (our application) has requested. A header specifies the parameters of the HTTP transaction. In other words, a header carries information such as details about the host, and login credentials. If you have to include login credentials, you have to encrypt them. Nedap's database use an encryption called Base64<sup>17</sup>.

The request method specifies what the request wants. There are several methods available, but the most common are the GET, PUT, POST and DELETE methods. You use GET when you wish to retrieve data from a database, POST to add data to a database, PUT to replace data in a database and DELETE to, as the name suggests,

delete data from the database. For our application, we only needed to retrieve data so we only needed to use the GET method.

Thanks to Nedaps thorough documentation of the REST API, it was very easy for us to figure out what requests to send to get the data that we wanted. A simplified version of such a request looks like this: "intin.nedapparking.com/parkingLots".

When the application sends this request, it actually sends it with the structure you see in figure 18. See figure 19 for an explanation of the structure. The login credentials are included as a header.

GET /parkingLots HTTP/1.1  
Host: intin.nedapparking.com

Figure 18: Request structure

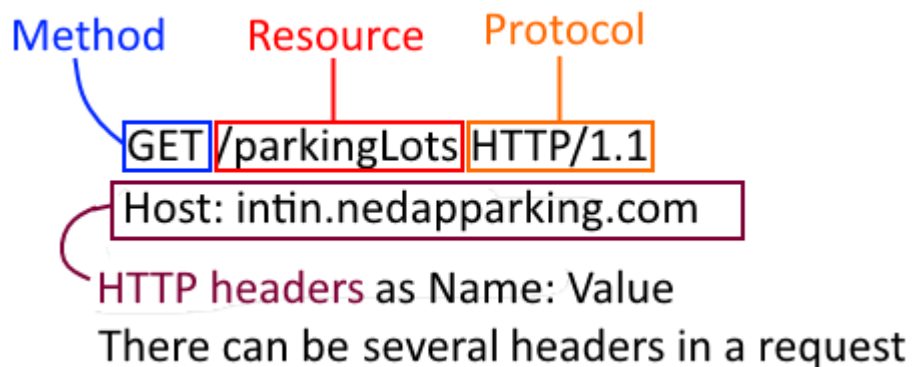


Figure 19: Explanation of request structure

When the API on Nedap's server receives this request, it asks the database for the requested data, and returns a HTTP response. If the request was correct, and the database successfully delivered the data, the HTTP response will contain it in a format called JSON<sup>18</sup>. JSON

stands for JavaScript Object Notation, and is a syntax for storing and exchanging data. JSON is easy for humans to read and write, and easy for computers to parse and generate. See figure 20. When

```
[  
  {  
    "id": "1",  
    "description": "Helse Forde",  
    "name": "Helse Forde",  
    "bays": null  
  }  
]
```

Figure 20: JSON formatted response

the application receives the data in JSON format, it stores it, processes it and then shows it to the user.

## 5.5. Creating the user interface

As we mentioned earlier, we divided our user interface into three sections: a map-section, a list-section and a settings-section. The list-section would just be a screen with list containing all the charging stations with relevant information and a picture. The map-section would be, as the name suggests, a map covering the screen, but with some additional features such as a search function. In the settings-section, the user could make relevant adjustments to the look and feel of the application.

In following chapters, we will write as if there were several charging stations connected to the application, and not just the test-site.

## 5.6. The map

The most important visual feature of our application the map. There were many smaller features that we wanted to implement in the map-section, and this section definitely required the most work. The most important thing the map had to be able to do was show the location of each car park by individual markers. As previously stated, we only had the car park at Førde Central Hospital to use in the application, but we wanted to build it so that it was able to support several car parks. After the application had retrieved the relevant data from the database, it would take the coordinates of each car park, and place a marker on the map at each car parks coordinates. The marker would be colored green if the car park contained free charging stations and red if there were no free charging stations in the car park.

If the user clicked the marker, a label would appear with the name of the car park, the distance from the user's current location, how many free charging stations and a button to get driving directions to that car park. Additionally, we wanted to show the user how long a charging station in the car park had been occupied, and make the application notify the user when a selected charging station became available.

In addition to this, we wanted the user to be able to search for a specific location. The user would type the name of the location, the map would then search for it and, if found, move the camera to it. This way, the user could easily check for available

charging stations anywhere they were, or wanted to go. Lastly, as a minor feature, we wanted to give the user the option to choose between different map modes like satellite and terrain modes.

Before we could even get started with implementing all of this functionality, we had to make the map actually show up on the screen. In order to use Google Maps in your application, you have to use the Google Maps Android API (Application Programmable Interface)<sup>19</sup>. Google actually provides several APIs that developers can use to make their applications (see figure 21 below), and we ended up using three of them in our application. We will explain more about this in later sections. In order to use Google Maps Android API, you have to log in to the Google Developers Console with a Google account, and get an API-key that you put into your application code. Without a valid API-key, Google Maps Android API will deny your application access, and the map will not appear on the screen. After we had gotten the API-key sorted out, and the map appeared on the screen, we could start implementing the features we had planned. We decided to start with the markers.

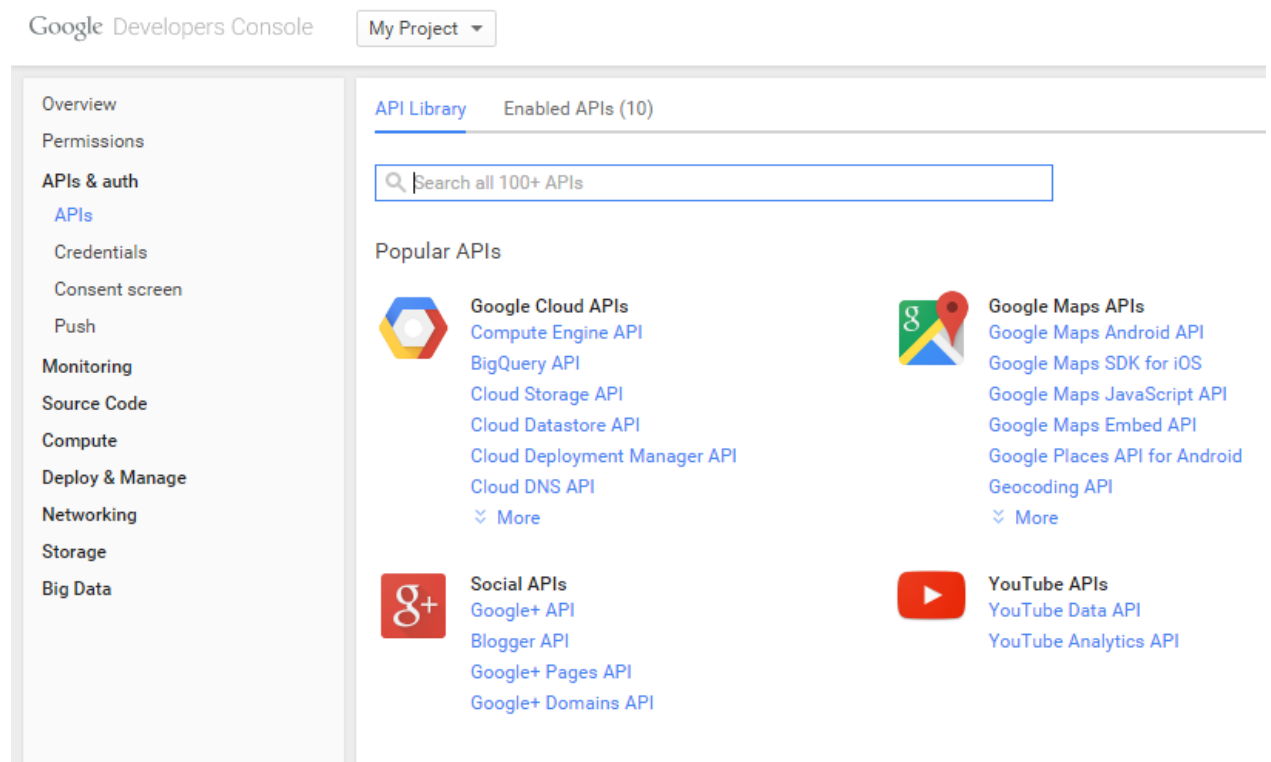


Figure 21: Google API library

The markers actually turned out to be easier to implement than we originally thought because Android provided a very simple method to create a marker on the map<sup>20,21</sup>. All we had to do was provide the coordinates of each car park to make it appear at the correct location. We also provided the other relevant information such as the name, distance and the number of free charging stations in that car park in order to create the label. We got the name and the number of free charging stations from the database, so all we had to do was calculate the distance. We already knew the location of each car park, but in order to calculate the distance to it, we had to make the application find the location of the smartphone. After a bit of work we were able to get the location of the smartphone<sup>22</sup>, and we could simply calculate the distance to the car park using a method that Android provided us. With the distance calculated, we supplied it along with all the other pieces of information, and made the application create each marker using the method that Android provided.

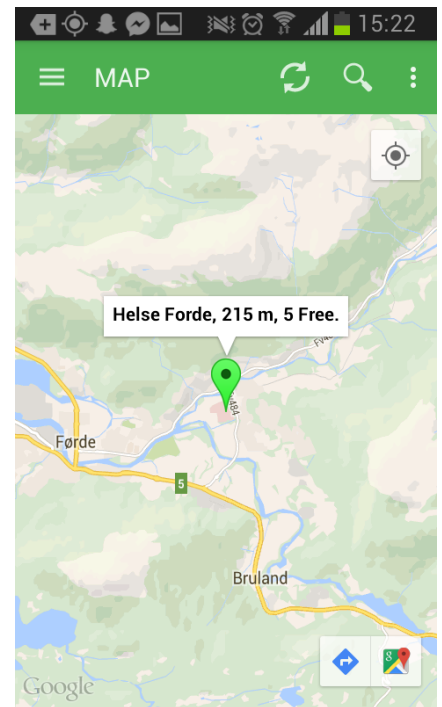


Figure 22: Map screenshot

The next thing we tackled was the location search. Our plan was to make a search bar at the top of the screen where the user could type the name of the location. As the user typed a name of a location, we wanted the application to provide relevant auto complete predictions, as we believed this would improve the user experience. When the user clicked the search button, the map would then search for the location, and, if found, move the camera to it. This way, the user could easily check for available charging stations anywhere they were, or wanted to go. We were able to make this work by utilizing the Google Geocode API<sup>23</sup>, and Google Places API<sup>24</sup>.

Geocoding is the process of converting addresses (like “The Red Keep, 1234 King’s Landing”) into geographic coordinates (latitude and longitude). The Google Geocode API provides a direct way to access a geocoder via HTTP requests. You can also do something called “reverse geocoding”, which is simply converting coordinates to addresses. The Google Places API is a service that returns information about a place.

Google Places API defines a place as an establishment, a geographic location, or prominent point of interest.

The application utilizes both of these APIs by sending HTTP requests. As soon as the user types in three letters in the search bar, the application sends a HTTP request to the Google Places API. The API tries to predict what place the user is typing, and returns a HTTP response with the suggestions in JSON format. For each additional letter that the user types, the application sends a new HTTP request. For example, if the user types “Oslo” the application will send two HTTP requests. One for “Osl”, and one for “Oslo”. See figure 23. In both responses, Google Places API will have suggested “Oslo, Norway” and other locations around the world that resembles “Osl”, or “Oslo”. We believe this improves the user experience because the user does not have to type in the entire name of the location. He/she can simply type parts of it, and the application will make the appropriate suggestions.

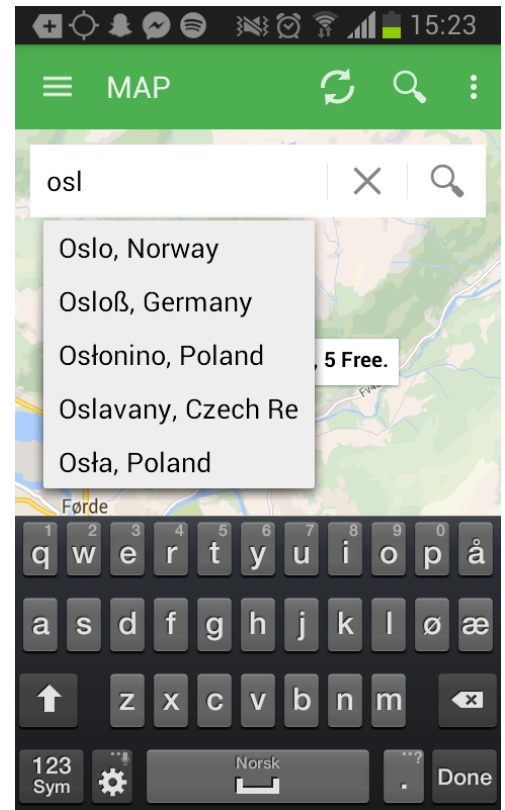


Figure 23: Search function in map

When the user clicks search, the application sends another HTTP request, but this time to the Google Geocode API. As mentioned previously, the Google Geocode API converts addresses into coordinates. If what the user typed into the search bar was a valid address, or name of a location, the Google Geocode API will return the coordinates for that location. At this point, all that remains is to move the map’s camera to the received coordinates.

The last obstacle was the driving directions. We already had a way to show driving directions, but we wanted to create a better version. The first version worked like this: When you clicked on a marker, Android provided three buttons by default. One in the top right corner, and two in the bottom right corner. See figure 22. The one in the top right corner was just to make the map move to the phones location. However, the two in the bottom right corner did almost exactly what we wanted. One of them showed the location of the marker, and the other provided driving

directions and a drawn path. The problem with the first version was that when you clicked these buttons, the phone minimized our application, and displayed the location and driving directions, in the Google Maps Mobile application. In other words, in order to get the driving directions, you had to switch applications. We did not want this, and therefore decided to try to implement driving directions in our own application.

After some research, we found that we could use the Google Directions API<sup>25</sup> to accomplish this. When we sent the correct HTML requests to the Google Directions API (just like the Nedap database API), it returned an encoded path in JSON format. We were able to decode the path, and draw it onto the map by using a method that Android provided<sup>26</sup>. However, shortly after we accomplished this, we realized that showing the proper driving directions, and manage the drawn paths properly (like adding and removing them in an efficient way), would require a lot of time. We did not have this time so we decided to scrap it, and just use the default buttons that Android already provided, and the Google Maps Mobile application.

## 5.7. The list

As previously mentioned, it was very important to us to limit the amount of clicks as much as possible, and deliver the information to the user as quickly as possible. Due to this, we wanted to show as much information as possible on each item in the list, instead of showing it on a new screen after the user had clicked on an item. Because we wanted to show so much information on each item, it became a concern that they would become too big, and the list would become cluttered and hard to navigate. We discussed it, and concluded that the advantages of showing the information without any more clicks outweighed the size problem. We wanted the list to show the following information



Figure 24: Single item in List

about each car park: a small picture, a name or address, the number of charging stations, the number of *available* charging stations, an availability-indicator that would be colored red or green and the distance to the car park from the user's current location. See figure 24 for the completed layout. See figure 25 for an explanation of the layout. We also wanted each item to have a button that, when clicked, would open a new screen that showed the user more detailed information about the car park, and a larger picture of it. The more detailed information involved elements such as parking or charging fees, opening hours for the car park and if the car park was private or publicly owned.

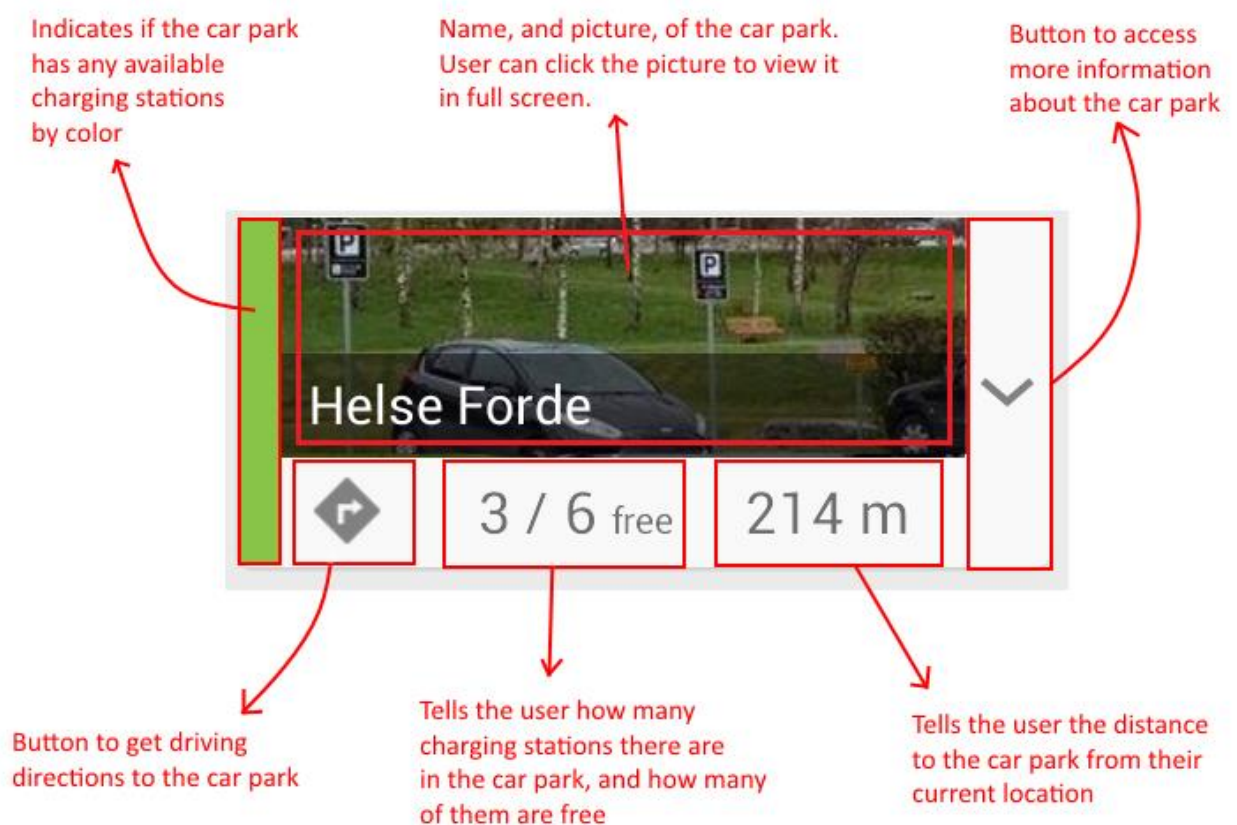


Figure 25: Explanation for single List-view item

Except from a few adjustments, we managed to make our list of car parks work exactly as we intended, and we completed it much faster than we originally anticipated. The biggest adjustment we made to the list was to the button farthest to the right on figure 25. When you click on the button, the item simply expands to show you the details instead of showing it on a new screen. We made this



adjustment because when we tried having the information on another screen, we felt that was a bit tedious and unnecessary. There was not much information left to tell the user that the list itself did not already tell, so making another screen to add just a small amount of additional information seemed unnecessary to us. The solution became to make the item in the list simply expand when the user clicked the button, and reveal the additional information. We also made it so that when the user clicked the picture on the list-item, the application shows it in full screen mode. This helps the user to see what the car park, and its surroundings, actually looks like which makes it easier to search for it. We strongly believe that doing it this way helps our campaign of delivering the information to the user as quickly as possible, and keeping the amount of clicks to a minimum.

The only problem that we ran into with the list was that the database from Nedap could not deliver the picture or information we wanted to show in the details-section. As we were not able to make our own database that could provide this information (see “how far did we get?” section) we ended up just “hard coding” it for preview purposes. To “hard code” means that you write the data yourself directly into the code, instead of retrieving the data dynamically, and showing it in the user interface. See figure 26. For example, if we would have “hard coded” the name of the car park to “Helse Førde”, all the car parks would have that name. With the dynamic way, the database would provide us with the correct name for each car park, and we would simply show what we receive. If we had a database available that could deliver the data, it would be very easy to remove the hard coded elements, and implement the dynamic way.

```
Dynamic version:  
String carParkName = getCarParkNameFromDatabase();  
carParkNameTextView.setText(carParkName);  
  
Hard coded version:  
carParkNameTextView.setText("Helse Forde");
```

Figure 26: Dynamic versus hard pseudo-code

## 5.8. The settings

The settings-section was the subject of much discussion throughout the development phase, and we changed our minds on several occasions about which settings we should implement. We had already agreed on a couple of different settings during the planning phase, but as the development of the two other primary features went on, we saw that we had to make some adjustments.

We originally had plans to implement the option to show only car parks that were within “X radius” from the user’s current location. We changed the plan to only showing the “X closest” car parks because we believed this would be much more user friendly. We also wanted to implement an option to change the color theme of the entire application between “night” and “day” modes. The idea was that it would make the application more comfortable to use when it was dark outside. We realized that this setting was not very useful to the application at this time, so we scrapped it to focus on other tasks.

The reason we have only mentioned what we planned until now is that we unfortunately were not able to implement the functionality of these settings. In other words, we made the section with a complete layout in the application, but nothing happens when the user clicks the buttons. The buttons are not “connected” to the rest of the application. We knew what we wanted to implement, but we simply did not have the time to make go through with it, and make it work properly. We ended up just making the layout in order to show the concept of our ideas. We will explain more about this in the “How far did we get?” section.

## 5.9. Tying the sections together

Until now, we have explained how we made each section of the application, and how they work, but we still have not explained how we made the user able to switch

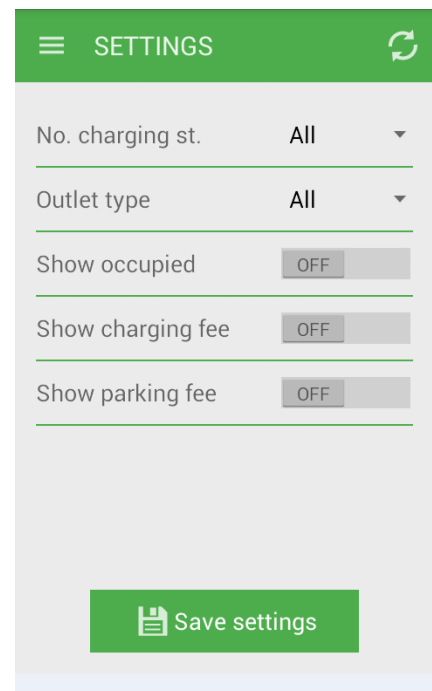


Figure 27: Screenshot of Settings

between these sections easily. Our solution to the problem was to implement something called a Navigation Drawer<sup>27</sup>. A navigation drawer is a user interface element in Android, and is essentially a menu that contains a series of buttons. The user can use this menu to switch between the different sections of the application. See figure 28 below.

The user brings this navigation drawer out by pressing the navigation drawer button located in the top left corner of the application, or by performing a left-to-right swiping motion from the left side of the screen. When the button is pressed, or the user performs a correct swiping motion from the left side of the screen, the drawer will slide in from the left as if it were outside of the screen when it was not active. When the navigation drawer is active and visible, it will put itself on top of the content that was already there, and cover almost the entire screen. See figure 28 below. The button is always visible, and the user can always perform the swiping motion in order to bring out the drawer whenever he or she wants. When the user clicks on an item in the drawer, it will slide back to the left, and disappear. The application will then switch to the feature that the user selected. If the user does not want to click an item in the navigation drawer menu, he/she can simply press the button again, or perform a reverse swiping motion to hide the drawer. When the drawer closes, and becomes hidden, the application will return to the content that was already there before the drawer was opened.

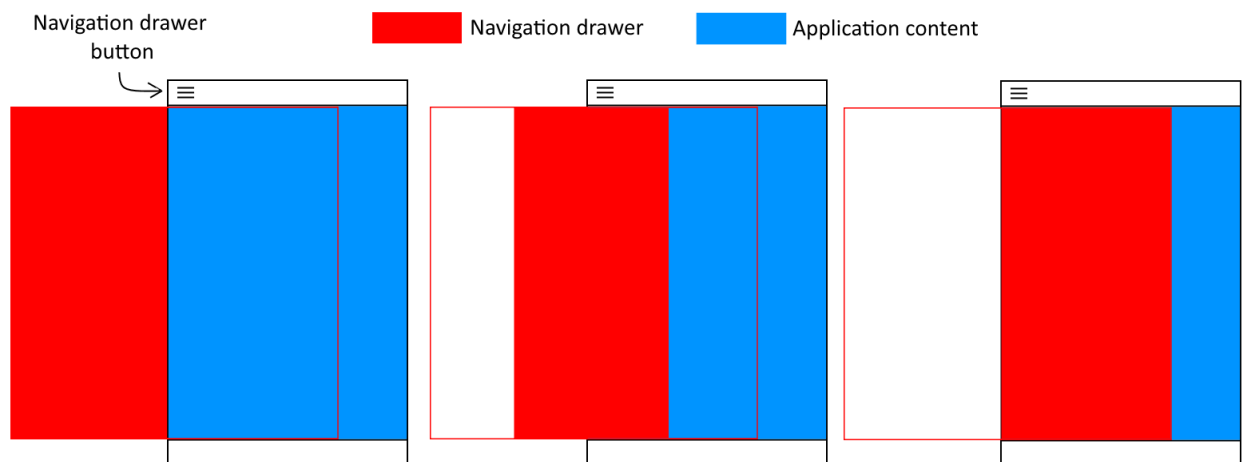


Figure 28: Navigation drawer positioning

The main alternative to using a navigation drawer was something called a Tab Bar<sup>28</sup>. A tab bar is a rectangle shaped bar with a series of buttons on it, which usually is located directly under an application's action bar

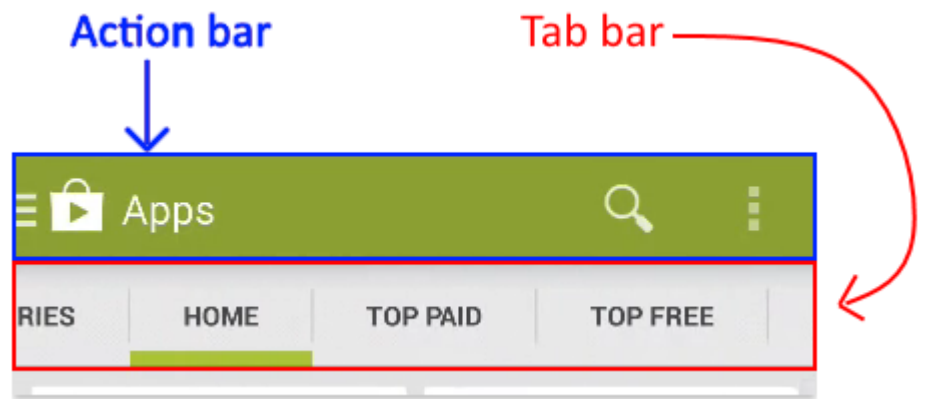


Figure 29: Action bar versus Tab bar

at the top of the screen. You use a tab bar for the exact same thing as a navigation drawer: switching between different screens/sections of the application.

We believe that there are several advantages in using a navigation drawer instead of a tab bar. The biggest advantage is that a navigation drawer is not visible unless the user brings it out, and that saves a lot of space on the screen. Smartphones have limited screen size, which makes it very important not to let any part of it go to waste. Tablets do have bigger screens, but the majority of people use their applications on smaller phones, and so designing applications to fit these screen sizes are extremely important. We also believe that a navigation drawer is more user friendly because it is bigger, and that makes it easier to create a user-friendly menu. The text on each button can be longer so that it is easier to explain what the button is supposed to do. You also have the space to use icons in addition to the text on the buttons, which also makes it easier to explain what the button function and purpose. Additionally, you often have some room left over for your logo or other branding items. We believe that these advantages prove that a navigation drawer is more user-friendly, and space-efficient.

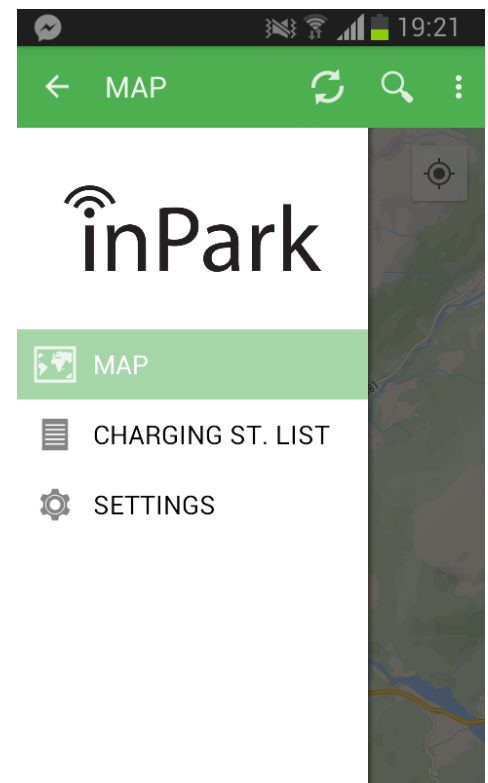


Figure 30: Navigation drawer screenshot

Our navigation drawer consists of our logo, and three buttons with relevant text and icons to explain what each button does, and where it takes the user if he/she clicks on it. See figure 30. We also made it so that the navigation drawer highlights the active section of the application. For example, if the user is using the map and then opens the navigation drawer, the drawer colors the map-button in with a color that is appropriate to the rest of the color theme. This removes any doubt for the user as to where he/she is in the app. In other words, this prevents the user from “getting lost” in our application. We chose to use only these elements on a white background in our navigation drawer in order to keep it as simple and intuitive as possible, yet pleasant to the eye.

### 5.10. Refreshing the data

The first thing our application does when you start it is to try to retrieve the data from the database. While it is doing this, it shows a loading screen to inform the user that the application is working, and have not frozen or crashed. After the application has retrieved the data (or failed), the loading screen disappears and the user is “allowed” into the application. If the application fails to retrieve the data, the applications opens the user interface normally, but it will be empty. In any case, the application needed to have a way for the user to refresh the data whenever he/she wanted. If the application failed to retrieve it, there had to be a way to try again. If the applications data was outdated, there had to be a way to refresh it to the latest version.

Because the application was already able to retrieve the data, all we had to do was make it try to get the data again, but we had to decide whether to make the application able to auto-refresh (for example every five minutes), or if the data should be refreshed manually. We chose make it a manual operation mainly for two reasons. The first reason was that we feared that downloading new data regularly would incur large costs to the user if connected

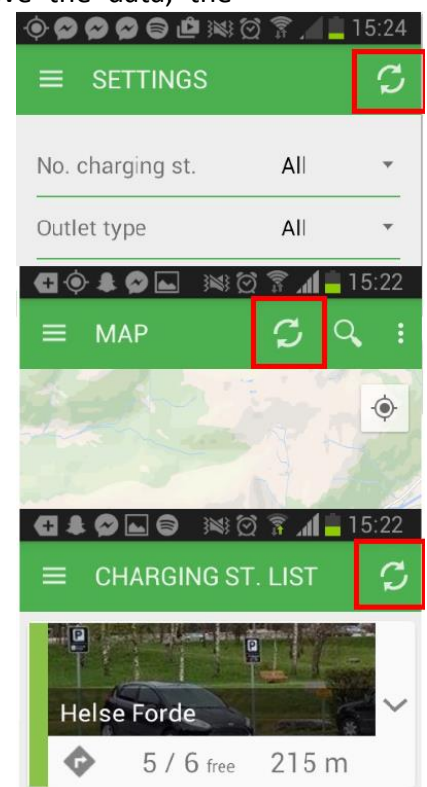


Figure 31: Refresh button in application

to the cellular network, and not regular Wi-Fi. The second reason was battery life. Constantly sending HTTP requests to the database and updating the data in the user interface would definitely have a negative impact on battery life. Because the application already monitored its own geographical position regularly, like on a traditional GPS device (an operation that already drains battery power), we did not want to put any more strain on it than necessary.

We implemented the refresh feature by adding a button to the action bar in every section of the application. We had set the action bar to be visible in every section, so putting the refresh button on it seemed like a very good solution. The button was very important, and putting it on the action bar made it clearly visible and easily accessible to the user. See figure 31.

### 5.11. How far did we get?

The great thing about a smartphone is that it can do so many different things. The technological advances the past few years have made the smartphones immensely powerful for their size, and consequently, these advances have opened up a plethora of possibilities for designers and developers of mobile applications. Sadly, some of these possibilities never make it into the final product due to things like time limitations. Time limitations applied to our project as well, and it forced us to scrap some features along the way, even though we really wanted to implement them.

When we started planning the application, we immediately made a list of all the features that we wanted to implement in a perfect world scenario. After making this list, we realized that the list had to be trimmed. As mentioned and described earlier, we categorized the features we wanted into **primary** features and **bonus** features.

The rule in the group was that we should implement all of the primary features first, and then try to implement as many bonus features as we could if we had the time and resources. The great thing about dividing the features into these categories is that it lets you focus on developing the important features first. It also makes it easier to prioritize the work, and it lowers stress for the developers because they do not have to worry about one giant list of features to implement. It also saves a lot of discussion within the group. Instead of having to decide what to remove from one

long list of features, it is much easier to look at the bonus-list first, and remove one of them seeing as they have a lower priority than the rest.

Below is a (simplified) list of features that we made, with the development status of each.

Table 4: Simplified list of feature status

Feature	Type	Status
Map <ul style="list-style-type: none"> <li>• Marker for each car park at geographical location, with info</li> <li>• Driving directions with a drawn path to each car park</li> <li>• The ability to search for a location</li> </ul>	Primary	Completed
List of available car parks with picture and relevant info	Primary	Completed
Settings-section where the user can make relevant adjustments	Primary	Incomplete
Our own database that contains detailed info about each car park	Bonus	Scrapped
The ability to mark car parks as favorites for easier access	Bonus	Scrapped
Route planner so the user can easily plan a long trip	Bonus	Scrapped

As you can see from the list of the primary features, the settings-section is incomplete. We started working on it, but we only had the time to implement the layout properly so that we could show the concept. The controls on the screen do not do anything. Because we did not even have time to complete all of the primary features, we had to scrap all the bonus features. Despite the fact that they did not make it into the application, we still want to explain the idea behind them.

The scrapped feature that would have had the largest impact on the application was the inPark database. One of the problems with the Nedap database was that it could not store a portion the data that we wanted the application to show. In addition to monitoring the sensors, we wanted to store the following data about each car park remotely in a database:

- A name for the car park
- The address of the car park

- A picture of the car park
- Parking & charging fees
- If the car park was public or privately owned,
- Opening hours
- What type of outlets the charging stations in the car park supported.

Our plan was to make our own database that could store all of the data in the list above, in addition to the data that Nedap's database provided. The application would then get all of the required data from our database. See figure 32. We believed it was better to make the application get everything it needed from one database, instead of two.

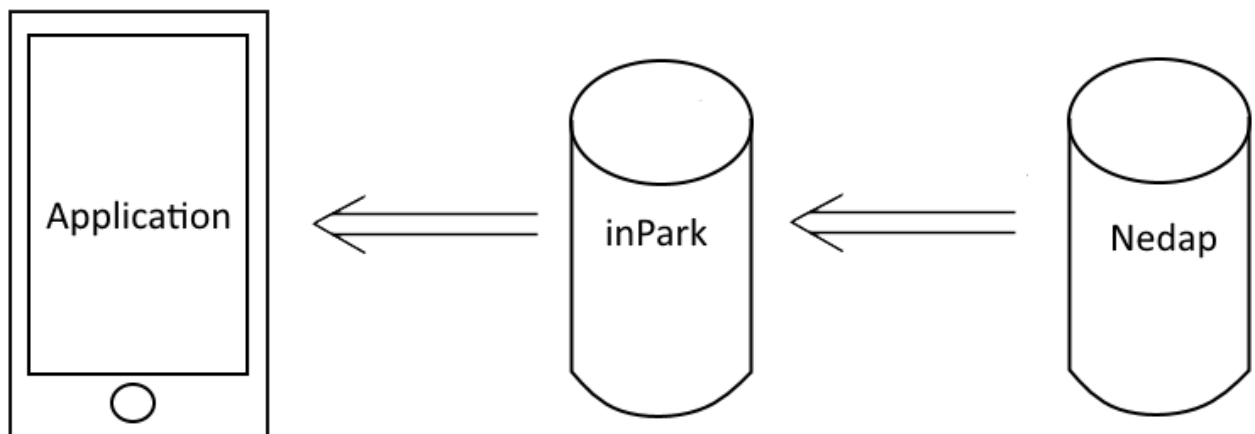


Figure 32: How we intended to implement inPark database

The main reason that we scrapped this was time limitations. We would have to program the database to send HTML requests to Nedap's database on a regular scheduled interval (for example every five minutes), receive the data in JSON format and then store that data. In addition to this, we would have to program the database to be able to receive HTML requests from the Android application, and then return the requested data in JSON format in a HTML response.

We absolutely believed that we were able to accomplish this if we had tried, but when we got to the point where we could start developing it, we realized that we simply did not have the time. If we had started the development of the database at that time, we would be unable to finish the project in time.



The favorites-feature was a simple quality-of-life change that we wanted to give the user. We assumed that many owners of electric cars use them for short trips in their local area, or commuting. We therefore assumed that many of those people would use the same charging station on a daily basis, and because of this, we wanted to give the user the option to mark one, or several, car parks as a favorite. When a car park was marked as a favorite, that car park would be more easily accessible in the user interface of the application. For example, car parks that were marked as favorites would appear at the top of the list, or be specifically highlighted on the map. The main benefit from this would be that the user would be able to quickly, and easily, check if the car park had any free charging stations.

The final feature we had to scrap was the route planner. The idea was that the user could easily plan a trip that required charging the car along the way. The user would enter his/her starting point and destination, and the application would then draw a path on the map, give driving directions and display the closest car parks that contained a charging station along the way. We naturally assumed that the battery life was a big concern for people that had to use an electric car for long trips, and with this feature, we wanted to remove that concern as much as possible.

We are unhappy with not being able to complete all of the primary features, and having to scrap all of the bonus features as a direct consequence. We think that there are two reasons that we had to make as many cuts as we did. The first being that the internship of our main programmer impeded our progress greatly. He was not able to work on the project while he was away, and the rest of the group had to prioritize other tasks while he was away. The second reason is that we were probably overly ambitious when planning the application. We wanted to implement too many features, and some of the features were too big and advanced for us to implement properly within the time schedule. We believe that these features would greatly improve our application. Despite not having time to implement them, we think that our overly ambitious planning only help to prove that “the sky is the limit” when it comes to android development.

## 6. Environment

These days, the environment is a big issue for a large part of the population, and we believe it is very important to consider what kind of environmental impact this experiment might have. Our thoughts and opinions are strictly hypothetical, and it is not certain that our predictions, or opinions, reflect what the results would be in reality.

### 6.1. Discussion environmental rewards

When discussing which rewards the environment will have by implementing this system, we have to consider what it could mean for congestions, road dust, emissions and power usage. We believe that it will be considerable easier for electric car owners to know where to drive when he, or she, needs to charge their car with such a system in place. Obviously, if a person with an electric car has to spend a lot of time driving around to search for a charging station, that person's car will not add to the emissions. However, that person will contribute to releasing more road dust into the air and increasing the possibility of more traffic congestion. More traffic congestion leads to more emission. If the person knew exactly where to drive to find a charging station, it would most likely improve traffic flow, lessen congestion, reduce emission and lower the amount of road dust in the air. However, it is worth taking into account that only 50,000<sup>29</sup> of the 2.6 million<sup>30</sup> privately owned cars in Norway are electric. This equates to only two percent. We can also probably assume that most owners of electric cars use them in their local areas, or for commuting, and thus use the same charging station regularly. Therefore, there will not be many people in electric cars driving around looking for a charging station on a daily basis.

On the other hand, if we take hybrid cars into the equation, the emission question becomes more relevant. The Toyota Prius Hybrid<sup>31</sup> has CO<sub>2</sub> emissions starting at 86 g/km, while the Skoda Octavia<sup>32</sup> releases about 116-130 g/km. For example, if the inPark solution could help someone with a Toyota Prius Hybrid to find a charging station quickly, and thus reducing the driving distance for that car, then that would absolutely improve emissions. If you reduce the driving distance of a Toyota Prius Hybrid by only five kilometers, you would prevent at least 430 grams of CO<sub>2</sub> emission

from being released into the atmosphere from that car alone. In comparison the Skoda Octavia releases approximately 600 grams of CO<sub>2</sub> in five kilometers. The effects of the inPark system on a global scale will not be noticeable until the vast majority of cars are either hybrid or purely electric. By then there still will not be massive environmental rewards using this solution. Electric and hybrid cars does not have much emission to begin with, and aiding them by reducing their driving distance by some kilometers here and there will not make much of a difference on a global scale.

In regards to aiding traffic flow, we genuinely believe that this system will have a positive impact. It is no secret that the traffic flow improves when there are fewer cars on the roads, and all drivers know where they are going, rather than driving around looking for an unoccupied charging bay. Driving and searching for an unoccupied lot is probably a larger problem in larger cities than in small ones. With a system like inPark in place, you can reduce this type of traffic significantly. If there are long queues to charge you are most likely interested in finding somewhere else where you can charge right away instead of waiting for an unknown amount of time. With the sales of electric cars skyrocketing it is likely that queues and waiting-time are getting longer. When there are many electric cars driving around looking for a charging station to use right away, there will be a correlating negative effect on traffic flow, thus contributing to more emission from regular cars that are affected by the increased traffic. With more search traffic it is also possible that the new added strain to roads will eventually lead to a need for improvements on infrastructures. Road construction on its own has a lot of emission, and reduction in search traffic entails less need to improve infrastructure because of congestions.

If the inPark system were implemented to monitor regular car parks and their bays, it is our opinion that this system would greatly improve traffic flow in and around these car parks. With knowledge of where to park your car, you reduce the time you spend on both the roads and parking structure, and increase traffic the flow for others around you.

There are two types of charging stations: quick, and regular. As seen in table 5<sup>33</sup> there are some major differences in how much electricity they use when charging a car. When charging a car, the quick charging stations use considerably more electricity, compared to the regular one. This is also reflected in the cost<sup>34</sup> of using a quick charger instead of a regular (2.50 NOK/minute versus 1.00 NOK/minute). The inPark solution lets the user choose which one to use, but we can probably assume that a large number of the users will choose to charge their car quickly. However, we believe it is essential to give the user the ability to choose, even though most are likely to prefer quick chargers.

Table 5: View of differences between charging information based on contact type.

Contact type	Technical	Power	Charge time 0-80%
<b>Type 2</b>	230V / 16A / 1-phase	3,5 kW	4-5 hours
<b>Type 2 Industrial</b>	230V / 32A / 3-phase	12 kW	1-1,5 hours
<b>Quick Charger AC</b>	400V / 63A / 3-phase	43 kW	20.30 minutes
<b>Quick Charger DC</b>	400-500V / 100-125A	50 kW	20-30 minutes

If it is correct to assume that most people prefer quick chargers, there has to be some consideration to the strain it will put on the power grid. It is our understanding that most quick chargers to date are located where the capacity for the closest transformer is sufficient<sup>35</sup> enough to handle the extra load. Which would mean that those who install quick charger consider the power grids capacity when installing quick chargers in regards to their power surge.

## 6.2. Environmental Return on Investment

Nedap have developed a Return On Investment calculator (ROI), which they demonstrated by using Reading UK as an example<sup>36</sup>. In this example we can actually see some of the environmental rewards given by a parking system like the one Nedap delivers. The complex in Reading consists of 3000 sensors where each bay has a parking fee. The numbers are based on a five-year period. This structure is a regular car park, and not charging stations like inPark, but the results are interesting to look at nonetheless, and if the system were implemented at all charging stations that do not deliver real-time vacancy-status some of the figures might apply to our situation as well. The main difference is that our purpose builds on access-information through an application, while the Reading set up is broadcasted via electronic signs.

In regards to search traffic, they found that it was drastically reduced when the parking system was installed. When we discussed environmental effects, we did not consider time at all, but when looking at figure 33 we understand that it is a lot of time to be spared and that this might be one of the largest personal gains within such a solution.

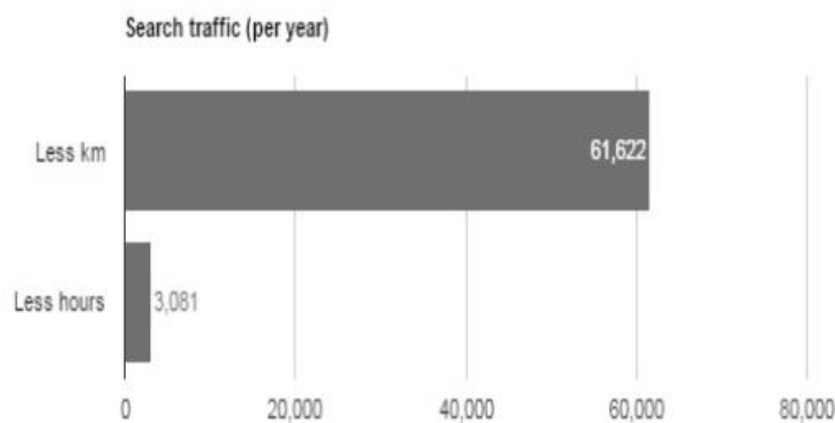


Figure 33: Graph to show how much search traffic was reduced in a five-year period.

With the numbers from figure 34, they could determine how much emission had been prevented each year. From figure 33, we can gather that for a five-year period it is quite a substantial amount.

With a reduction of  $13\,061 \text{ kg/year} \cdot 5 \text{ year} = 65\,305 \text{ kg CO}_2$  in just five years, we can only begin to understand what kind of impact a system like this could have in the long run if used world-wide in all larger cities.

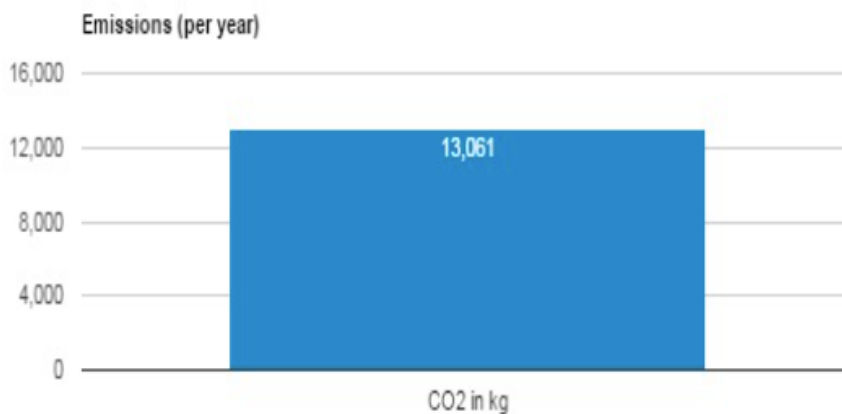


Figure 34: Graph to show reduction in emission each year.

The system set up in Reading were as previously stated 3000 sensors where all parking bays have parking fees. The cost of the sensors were €775,000, parking signs €220,000 and parking application €22,000 which adds up to a total cost of €917,000. In the five-year period used by Nedap to calculate ROI, they made a simple chart to show revenue generated by fees alone. We are unsure whether they used the amount of extra turnover in vehicles facilitated after implementing Nedap's system to calculate the revenue, or if it is calculated by a total of vehicles. We do not know the parking fee, and therefore cannot calculate it ourselves. Nevertheless the profits go to show that it paid for itself in a rather short period of time. Looking at figure 35 you can see the number of extra vehicles facilitated and in figure 36 revenue from fees and fines.

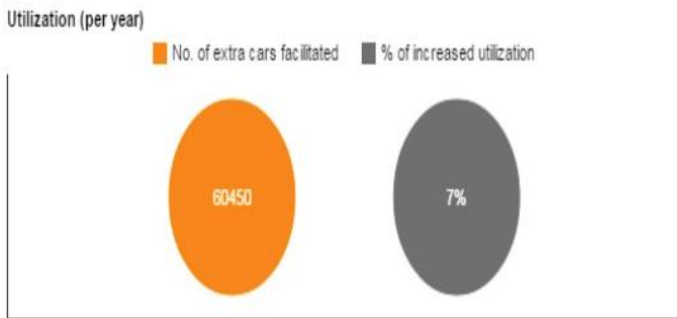


Figure 35: View of extra cars facilitated, in numbers and percent.

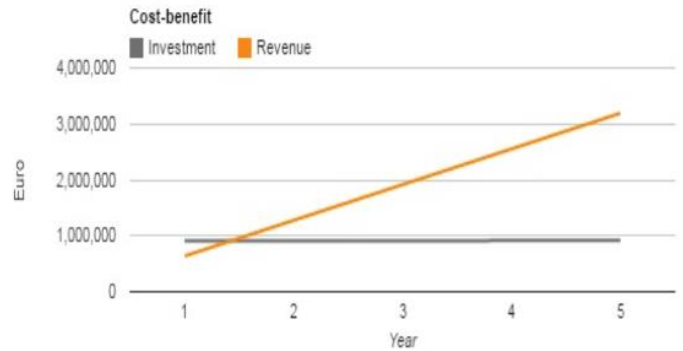


Figure 36: Revenue over five-year period

We think these values go to show that the rewards of such a system outweigh the costs of implementing it.

The question is if whether or not this is transferrable to charging stations. There are some major differences in the two scenarios. The Reading system is one large structure consisting of 3000 parking bays, while inPark are many small locations. However, if we think of all the charging stations as one, we believe that we could be looking at some of the same rewards when it comes to revenue and turnover. The contribution to reducing emission will not be comparable, as our target users drive electric cars. Unlike most regular car parks, charging stations does not have a parking fee. However, it seems most charging stations have a start-up fee<sup>37</sup>, and further pricing based on time spent charging. The prices most likely deviate between the various providers. These costs are similar to a parking fee, and could be used to compare the economical rewards by having real-time occupancy status delivered to customers. We believe that if the inPark system were implemented at charging stations around the country, the owners of those stations would notice a higher turnover and revenue than before.

With a continuous growth in the amount of electric cars out there, we think that the inPark solution could make a correlating positive environmental impact. Reducing search traffic for an increasing amount of electric cars will improve traffic flow. With better traffic flow you lower emission and might reduce the need for congestion-induced infrastructure improvements, which pollutes a great deal. The environmental rewards really come down to how many users and connected charging points the inPark system has.

### **6.3. Beneficiaries**

The inPark solution monitors the availability of charging stations, and makes it easier to find them. It is therefore easy to discern that the primary beneficiaries of this solution are the owners of electric cars. This answer, however, is not sufficient and does not cover the whole specter of people who will benefit from it. We believe that the solution will improve traffic flow, and that owners of regular cars will benefit because of this due to less traffic congestion. Improved traffic flow will also benefit people living close to the road because there will be less noise and road dust in the air. However, as previously stated, there are currently not enough electric or hybrid cars in the world for such a solution to make a noticeable environmental impact on a global scale. Nevertheless, the system will provide owners of electric cars with noticeable quality-of-life improvements such as quickly, and easily, finding an unoccupied charging point nearby if the battery is running low.



## 7. Conclusion

We have successfully created a solution that aids drivers of electric car to quickly, easily and efficiently find charging points, and get information about their status in real-time. We have also considered if such a solution could improve traffic flow, reduce emission and help to provide a higher turnover of charging stations if implemented on a large scale.

We believe that the inPark system can benefit not only drivers of electric cars, but also drivers of regular cars. Providing information about charging point vacancy will make it much easier for drivers of electric cars to plan their route, and avoid spending unnecessary time searching for a charging point. This will improve traffic flow, thus benefiting everyone else in traffic as well.

The amount of electric cars in Norway is still low, and therefore the environmental rewards from our solution would be present, but not very noticeable. However, as the amount of electric cars is steadily increasing, and more people start using our solution, the environmental rewards will only increase.

In regards to our application, we are very satisfied with what we were able to develop. We firmly believe that it is user-friendly, efficient and that it has the potential to benefit drivers of electric cars. It makes it easier to plan driving routes, and lets the user know where there are available charging points, at his/her convenience.

Based on the figures from the Reading-example, we feel confident that the inPark system will provide a higher turnover for the charging stations that use it. Supplying information about vacancy in real-time will help users to avoid already occupied places, and go to a vacant charging point somewhere else. This will result in more people charging their cars, and less people waiting in queue.

## 8. Project administration

### 8.1. Group administration

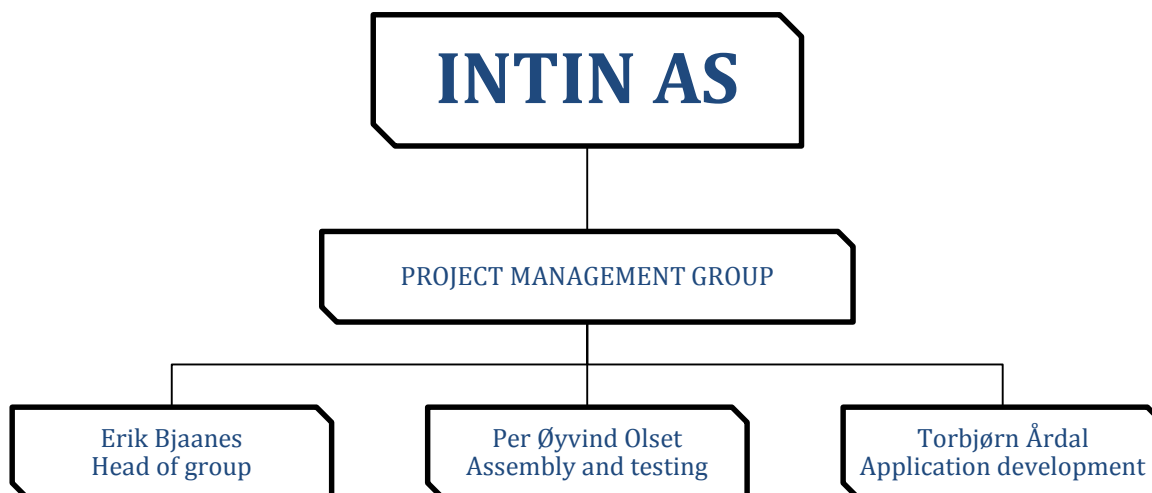


Figure 37: Group hierarchy

The head of the inPark group is Erik Bjaanes, with Per Øyvind Olset in charge of assembly and testing, and Torbjørn Årdal in charge of application development. We agreed that each member of the group would be in charge of their respective tasks, but the head of the group would have veto-power. The project management group consists of INTIN co-founder Rune Viken, assistant professor Joar Sande (HiSF) and Erik Bjaanes.

There were three major parts of our project: developing the Android-application, installing the system at the test-site and documenting the progress. The group distributed the tasks based on each member's interests and skillset. We believe that this approach yields the best result because it gives each member the opportunity to work on a task that he has a talent for, and is interested in doing. We also had regularly scheduled meetings where each member explained his current progress, as well as current challenges and problems. This made it a lot easier for each member to stay informed on what the others were doing, thus making it easier to keep track

of the current progress of the entire project. We did not always work separately, but it was mainly up to each member of the group to ask for help from the others. The regularly scheduled meetings also made it easier to assess each member's need for assistance. We also had regularly scheduled meetings with INTIN, but some of them were cancelled because it was not necessary at the time. We only deemed the meetings necessary if either party had important issues to discuss.

## 8.2. Planning phase

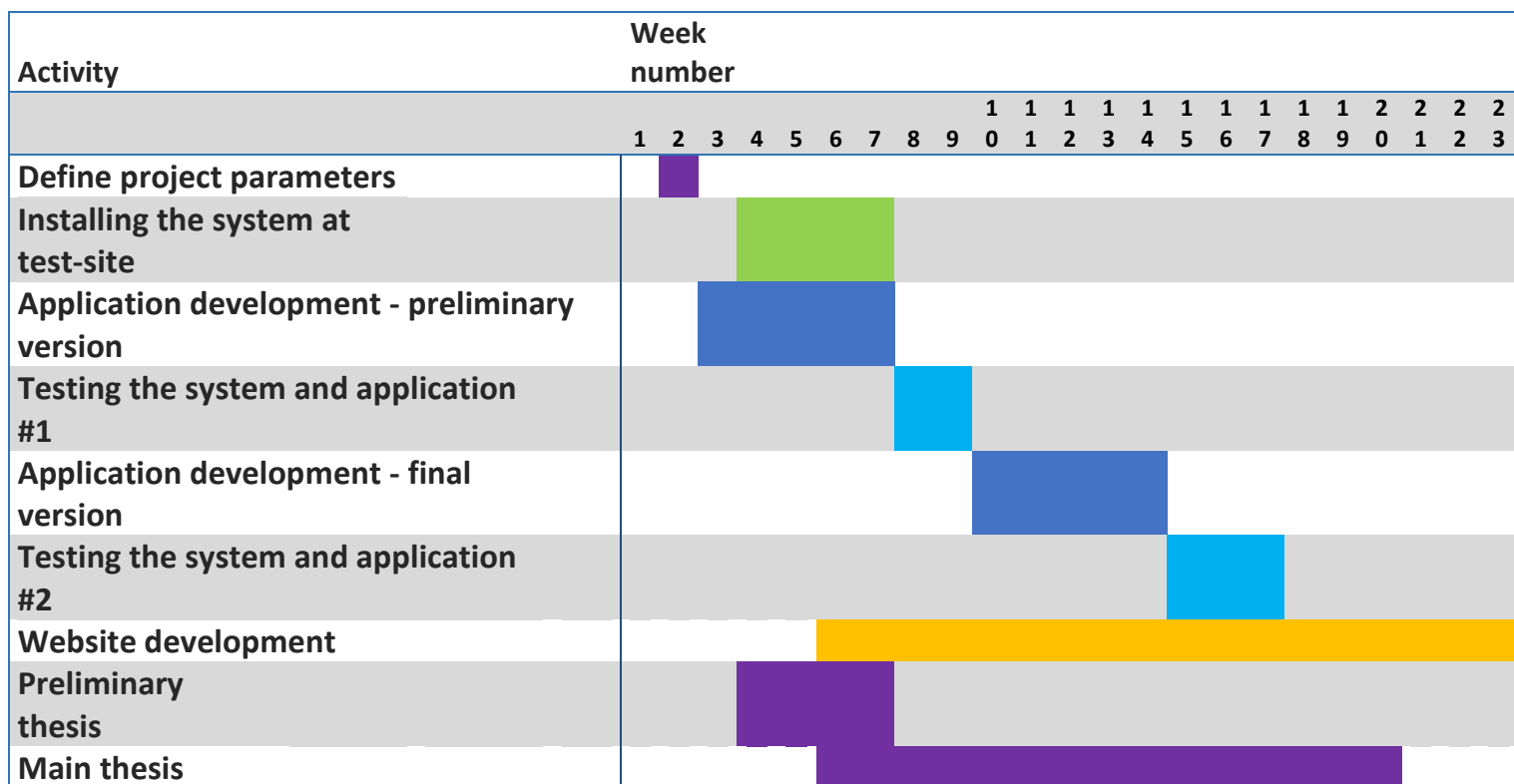


Figure 38: Planned time allocation viewed as Gantt chart

At the very beginning of our project, we made a Gantt chart to better visualize our expected progress throughout the semester. see figure 38. We aimed to start the development of the application in the third week of January, and have a working prototype by the middle of February. By that time, we also aimed to have the system installed at the test-site at a dedicated parking lot west of the HiSF building. We then planned to spend the rest of February testing the system, and fixing eventual problems that occurred. In March, we planned to add more features to the

application, and make it ready for the end-user, and then use the first half of April as another testing period. We also planned to develop the website, and write the thesis in parallel with the application development, and system installation. We were able to keep all the major deadlines, but some elements forced us to make several adjustments as the project progressed.

### 8.3. Time management

For detailed timesheets, see appendix 7-9.

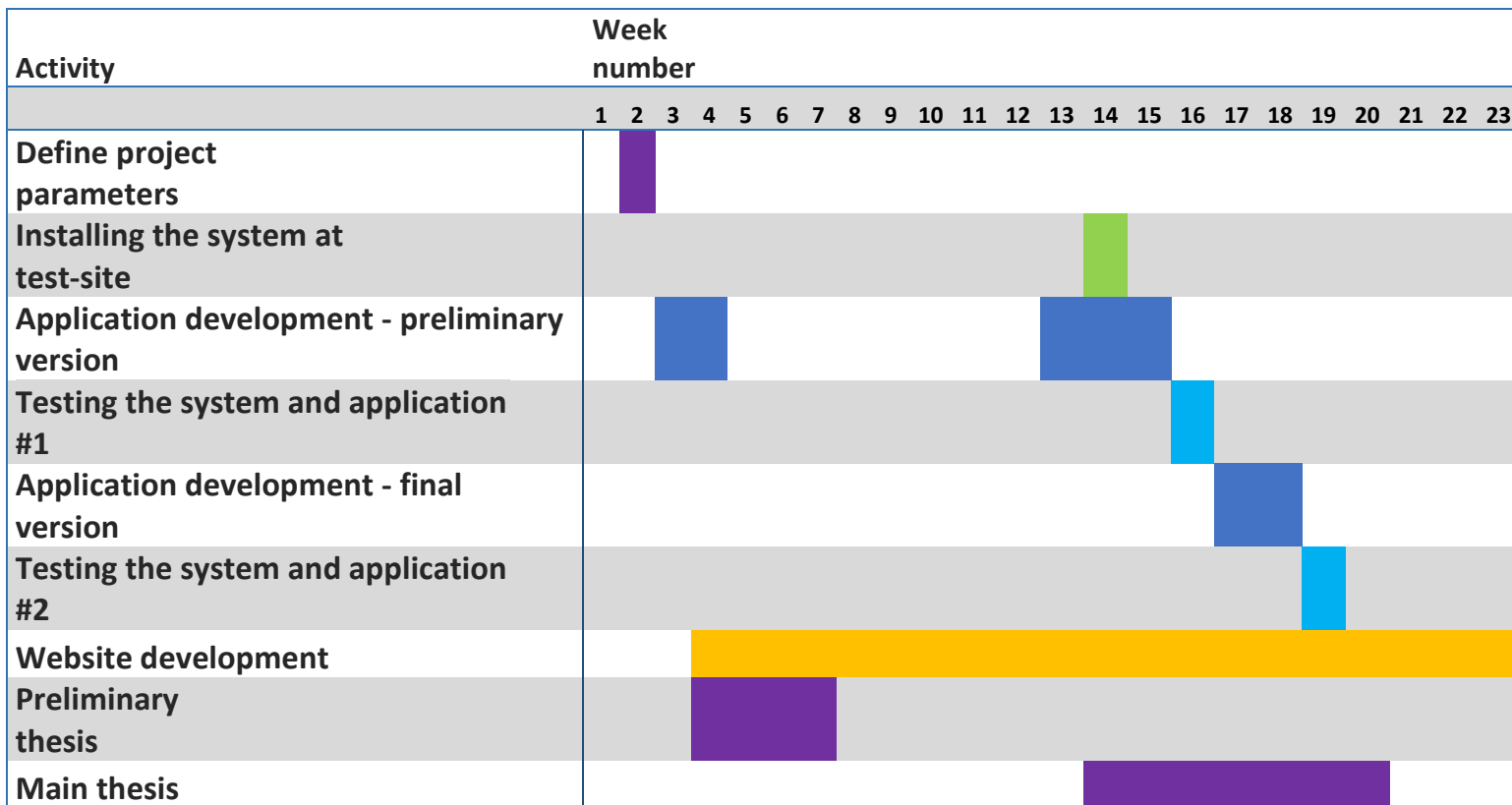


Figure 39: Actual time allocation viewed as Gantt chart

As evident from figure 39, which shows our actual work schedule, we had to postpone the development of the Android-application by approximately two months. This delay was caused by an internship two of our members had to attend for 30 business days, including the main programmer.

During the internships for two of our members, an elective subject occupied the third, meaning that the entire group was busy doing something else for a period of time. We knew about the internships beforehand, but we did not expect that it

would consume so much of our designated project time. We thought that we would be able to work on the project during the evenings. The main programmer notified the rest of the group during the first week of the internship that we needed to postpone the application-development completely until he got back, as he did not find the time to do anything projectrelated. Postponing this major part of our project obviously put us severely behind schedule. However, he worked a lot with Android application development during his internship, and as a result, the development of the application went a lot faster than we originally anticipated. As mentioned before, we firmly believe that the progress and quality of our product would be significantly lower if he had not attended this internship.

We also had to postpone the installation of our system by approximately two months. The original plan was to install the sensors at the dedicated parking lot at HiSF as soon as they arrived, but INTIN had contacted Førde Central Hospital shortly after our first meeting. They wanted to secure a deal where we could install the system at the hospital instead. There were also some minor delays with obtaining the correct equipment for the installation, and finding a date that would work for us, INTIN and Eivind Standnes.

Once the details had been set, we installed the system on the test-site, and we were able to complete the installation within three days. In our original planning we had designated three weeks to this task because we wanted to guard ourselves against potential problems, such as drilling obstacles, network stability and calibration issues. In addition, the given explanation of the system installation and previous experience with new and wireless technology further led us to think it would require a lot of work to get the system fully functional and stable. A good planning phase before the installation combined with a system far more simple to install than first, resulted in that the installation took three days. We are very happy with how fast we were able to install the system with only some minor problems during, and after the installation.

The delayed progress on the programming of the application combined with the long process of securing the installation site for our sensors led to a delay in the documentation and main thesis as well, as there were no real progress to document. The web page, preliminary thesis, press release and poster were all finished on schedule. The web page was developed, and finished, during the period that we were originally supposed to install the system, and develop the preliminary version of the application. The website has been continuously updated during our project. It will also receive a minor update once we have turned in this thesis, and presented the project.

#### **8.4. Risk assessment**

Early on in the preliminary project, we sat down and made our initial time schedule as viewed in figure 38 further up. At this point we knew about the internships two of our members would attend during the project period. What we did not know at that point, was at what time these internships would take place. We found it hard to do a proper risk assessment on how we rated the risks of not being able to stay on track according to our schedule. We did not account for an inability to work with the project during internships, which is evident from planned versus actual schedule. When application development and installation at test-site was postponed, the entire projects timeframe got sidetracked. Knowing what we know now, we regret not considering the possibility of this situation.

#### **8.5. Economy**

The only expenses in the project was the cost of the equipment from Nedap. INTIN bought all of the equipment, and as a result, the inPark group have not had any project related expenses. The groups feels uncomfortable disclosing the price of the equipment in this thesis. This is because we believe it is a private matter between INTIN and Nedap and we do not believe it is very relevant for the reader, or the quality of the report.

## Web page

We had to make a web page in our bachelor project that we had to update regularly to show our progress. The website also had to have our documentation of the project available for download. There are numerous webpages that provides several solutions to create webpages via point and click, drag and drop and similar. However, we found that these solutions focused too much on effects, and unnecessary animations, that took the attention away from the actual information. Some effects were just plain irritating and unnecessary, and it is our opinion that the content should be the focus, not the styling and animations.

In addition, code generated through the quick webpage solutions out there was messy, and it was hard to get the whole picture of what was going on. Since we wanted to learn more about web-development ourselves, we decided to make a web page ourselves from scratch. One member of the group had previous experience with web development, so the group delegated the task to him.

The group agreed that the user interface of the web page should be simple and minimalistic, yet visually pleasing. We also focused a lot on limiting the amount of clicks for the user as much as possible, just like our application. We made a banner in SketchUp that loosely showcased the concept to add a bit of graphic to the page.

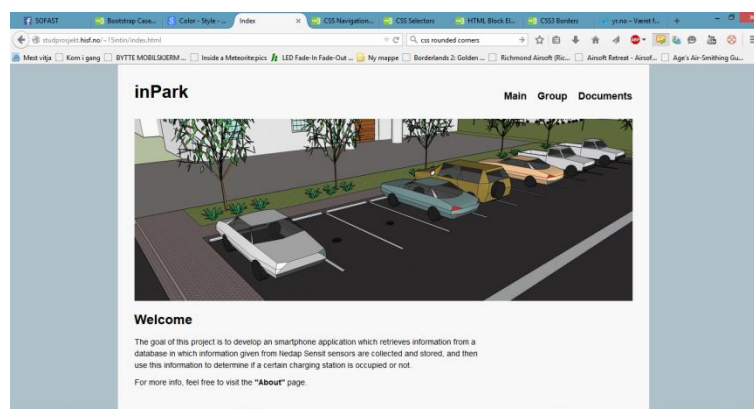


Figure 40: Preliminary draft

As the project went on, we learnt more about formatting, and tools to spice up the looks of the page. The group agreed to share the color scheme for both our application and webpage for consistency throughout our products. A logo was made to represent us, and INTIN provided some colors connected to development of their own logo. We chose the green as our main scheme, and set about adding it to elements of our page. We also added a link to INTINs' page.

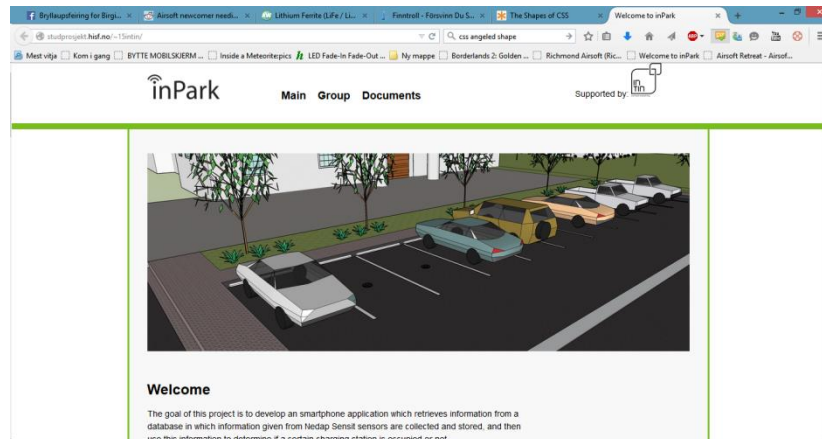


Figure 41: Second preliminary draft

At this point we were starting to see new opportunities for our page. We were careful to not overuse the green color, and not to add too many elements. We removed the “about us” page because the welcome page already covered that information.



Figure 42: Third preliminary draft



This version contains the final layout, but the banner image had to be modified to better showcase the idea behind our project. A picture can say more than a thousand words, and we put a lot of effort into the construction of the banner image to make it easily readable.



Figure 43: Final draft

The banner image was tweaked and charging stations was added via SketchUp. We added some signal-graphic to the sensors so they could be spotted easier, and the different elements points toward where the information is going. The layout of the webpage is also a dynamic sort, with the key green color showing at tactical positions, and sliding surfaces like the menu bar and “welcome”-padding repeating itself throughout the site. The site also responds well to smaller screens like a mobile phone due to some clever formatting code that adapts the content to some degree to fit smaller screens.

## 8.6. Self assessment

We realize that our initial planning of the project was sub-par. We knew about the internships beforehand and should have considered their time-consumption more when we first planned the project. If we had, we could have utilized the time leading up to the delays in a more effective way. The group had 1500 hours available for this thesis, and we achieved 1066 hours in total for the group. This is due to the delays from the internships and the installation process.

Despite the major delays early on in the project, we are very happy with our progress afterwards, and the final result. We believe that our final product has high quality, and we have worked effectively, and decisively, to achieve this in the time we had available. We do not feel that the delays that occurred have had a negative impact on the quality of our work, and the experience gained from the internships led to a more optimal workflow with more progress than we would have achieved without that experience. Another thing contributing not to hitting that 500-hour mark was that certain aspects of the assignment took a lot less time than expected. For instance, we expected a lot more time to go into getting the Sensit system to work properly and having to troubleshoot them connecting to each other. When this went without a hitch, we got a lower count on our hours, but the result remains the same.

On the positive side, our delays have taught us about the downfalls of not assessing risks properly. It has also taught us efficient time allocation. We are quite proud of what we have accomplished in a rather short amount of time.

## 9. Sources

- <sup>1</sup> <http://www.nedapidentification.com/about-us/>  
Downloaded 08.04.15
- <sup>2</sup> <http://www.ladestasjoner.no>  
Downloaded 16.05.15
- <sup>3</sup> <http://www.rosimits.com>  
Downloaded 12.05.15
- <sup>4</sup> [http://www.tinynode.com/?q=electric\\_cars\\_charging\\_station\\_monitoring](http://www.tinynode.com/?q=electric_cars_charging_station_monitoring)  
Downloaded 12.05.15
- <sup>5</sup> <http://searchnetworking.techtarget.com/definition/mesh-network>  
Downloaded 17.05.15
- <sup>6</sup> <https://innsida.ntnu.no/wiki/-/wiki/Norsk/Arbeid+i+høyden>  
Downloaded 19.05.15
- <sup>2</sup> Oral source: Eivind Standnes, janitor at Førde Central Hospital.
- <sup>8</sup> <http://developer.android.com/tools/studio/index.html>  
Downloaded 12.01.2015
- <sup>9</sup> <https://github.com/features>  
Downloaded 12.01.2015
- <sup>10</sup> <http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html>  
Downloaded 26.03.2015
- <sup>11</sup> [http://en.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol)  
Downloaded 26.03.2015
- <sup>12</sup> [http://en.wikipedia.org/wiki/Application\\_programming\\_interface](http://en.wikipedia.org/wiki/Application_programming_interface)  
Downloaded 26.03.2015
- <sup>13</sup> <http://money.howstuffworks.com/business-communications/how-to-leverage-an-api-for-conferencing1.htm>  
Downloaded 26.03.2015
- <sup>14</sup> <http://searchsoa.techtarget.com/definition/URI>  
Downloaded 27.03.2015
- <sup>15</sup> [http://www.tutorialspoint.com/http/http\\_methods.htm](http://www.tutorialspoint.com/http/http_methods.htm)  
Downloaded 27.03.2015
- <sup>16</sup> <http://code.tutsplus.com/tutorials/http-headers-for-dummies--net-8039>  
Downloaded 27.03.2015
- <sup>17</sup> <http://en.wikipedia.org/wiki/Base64>  
Downloaded 27.03.2015
- <sup>18</sup> <http://json.org/>  
Downloaded 30.03.2015
- <sup>19</sup> <https://developers.google.com/maps/documentation/android/>  
Downloaded 08.04.2015
- <sup>20</sup> <https://developer.android.com/reference/com/google/android/gms/maps/model/Marker.html>  
Downloaded 08.04.2015
- <sup>21</sup> <https://developers.google.com/maps/documentation/android/marker>  
Downloaded 08.04.2015
- <sup>22</sup> <https://developer.android.com/training/location/retrieve-current.html>  
Downloaded 09.04.2015
- <sup>23</sup> <https://developers.google.com/maps/documentation/geocoding/>

- 
- Downloaded 11.04.2015
- <sup>24</sup> <https://developers.google.com/places/android/>  
Downloaded 13.04.2015
- <sup>25</sup> <https://developers.google.com/maps/documentation/directions/>  
Downloaded 20.04.2015
- <sup>26</sup> <http://googlemaps.github.io/android-maps-utils/>  
Downloaded 20.04.2015
- <sup>27</sup> <https://developer.android.com/design/patterns/navigation-drawer.html>  
Downloaded 27.04.2015
- <sup>28</sup> <http://developer.android.com/design/building-blocks/tabs.html>  
Downloaded 28.04.2015
- <sup>29</sup> <http://e24.no/bil/paa-mandag-er-det-50-000-elbiler-i-norge-vi-er-noedt-til-aa-fase-ut-en-del-av-incentivene/23436093>  
Downloaded 08.05.15
- <sup>30</sup> <http://www.ssb.no/bilreg>  
Downloaded 08.05.15
- <sup>31</sup> [http://www.toyota.no/new-cars/prius/index.json#/publish/pageoverlay\\_open/html=.compare-data/preserveContent=true/styleClass=compare-overlay-wrapper/pageName=Sammenligning%20av%20motoralternativer](http://www.toyota.no/new-cars/prius/index.json#/publish/pageoverlay_open/html=.compare-data/preserveContent=true/styleClass=compare-overlay-wrapper/pageName=Sammenligning%20av%20motoralternativer)  
Downloaded 08.05.15
- <sup>32</sup> <http://www.skoda-auto.no/sitecollectiondocuments/car%20models/2013/skodapriser%20nye%20octavia%20my14%2012%2004%202013%20inkl.%20frakt%20og%20lev.pdf>  
Downloaded 08.05.15
- <sup>33</sup> <http://www.ladestasjoner.no/hurtiglading/om-hurtiglading/24-hva-er-hurtiglading>  
Downloaded 11.05.15
- <sup>34</sup> <http://www.ladestasjoner.no/nyheter/189-bergen-apner-verdens-storste-ladestasjon>  
Downloaded 14.05.15
- <sup>35</sup> Oral source Rune Viken.
- <sup>36</sup> <http://www.verkeerskunde.nl/Uploads/2015/3/SENSIT-ROI-Calculation-Results.pdf>  
Downloaded 14.05.15
- <sup>37</sup> <http://www.gcrieber-eiendom.no/aktuelt/verdens-stoerste-hurtiglade-park/>  
Downloaded 16.05.15

## Appendix

1. Sensit User Guide
2. Sensor specifications
3. Project management group notice of meeting 1
4. Project management group meeting 1 abstracts
5. Project management group notice of meeting 2
6. Project management group meeting 2 abstracts
7. Timesheet Erik Bjaanes
8. Timesheet Per Øyvind Olset
9. Timesheet Torbjørn Årdal
10. Application source code

# SENSIT SITE & DATA COLLECTOR

user guide

10-12-2014 | v4.0

## CONTENT

1	SENSIT SITE	3
1.1	VERIFY DATA COLLECTOR CONNECTION	3
2	DATA COLLECTOR	4
2.1	DATA COLLECTOR WITH GPRS CONFIGURATION	7
	GPRS MODEM LED BEHAVIOR	8
2.2.1	RED LED	8
	LED	8
2.2.2	AMBER	8
3	APPENDIX	9
A	DISCLAIMER DOCUMENT REVISION “	10

# 1 SENSIT SITE

Follow the few steps below to get the site up and running:

- 1 Power the Data Collector and connect it to the internet.
- 2 Activate the RelayNode with the magnet
- 3 Activate the SENSIT's with the magnet
- 4 Type the following URL into your web browser:  
<http://intin.nedapparking.com>

The Login screen will appear:



5. Type the correct Username and Password:  
Username:  
Password:

## 1.1 VERIFY DATA COLLECTOR CONNECTION

After choosing the Detection - Data Collector choice at the left side of the screen. The powered and to the internet connected Data Collector, should show up automatically in the Data Collectors connected list.

### Data collectors connected

List	Hostname or IP address	Port	Connect date
<input type="checkbox"/>	< your public ipaddress >	1333	Thu 14 November 2013 11:37:20

Number of server connection is 1 of max 5

### Note

The Data Collectors automatically connect to the server via internet, please make sure that the port below is open in your firewall:

IP-address:  
217.114.111.246

Port:  
11111



## 2 DATA COLLECTOR

You have to download from the Lantronix website the program Device Installer.

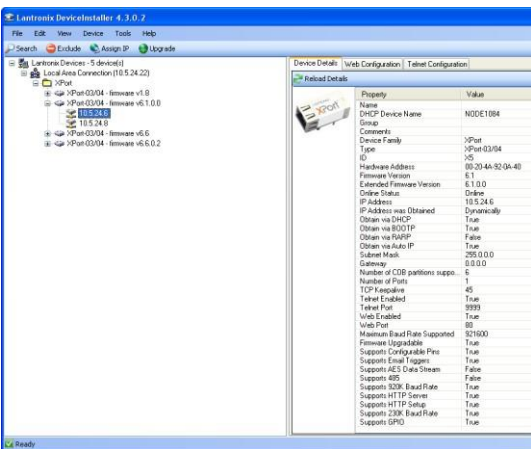
[http://ltxfaq.custhelp.com/app/answers/detail/a\\_id/644](http://ltxfaq.custhelp.com/app/answers/detail/a_id/644)

Configuration via the Lantronix Device Installer

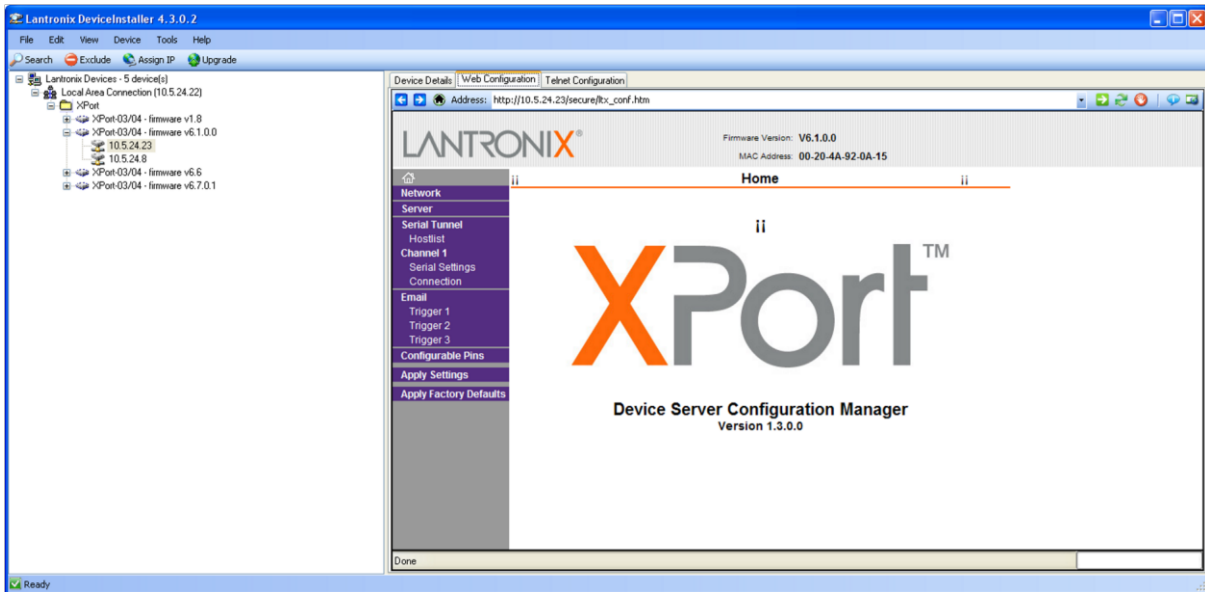
- Step 1: Start up the device installer and activate the search button.
- Step 2: After the search sequence you see the active Xport devices. Step 3: The X-port devices shown in red are not reachable you have to change your network settings. (by problems ask your IT manager)
- Step 4: Check which of the shown devices is your Data Collector. You can find out this by open the housing of the Data Collector and check the used hardware address. See picture below which has the hardware address nr 00-20-4A-92-0A-40



- Step 5: Double click at the pictogram of the Data Collector with the correct hardware address and get the shown information below. Activate the tab Web configuration. Press the ok button when the Authentication required popup appears. Please let User name and password blank.



Step 6: When the screen below appears your are able to fill in the correct settings for the Data Collector.



At the left side go to the Connection settings via Channel1 => Connection and set the following fields:

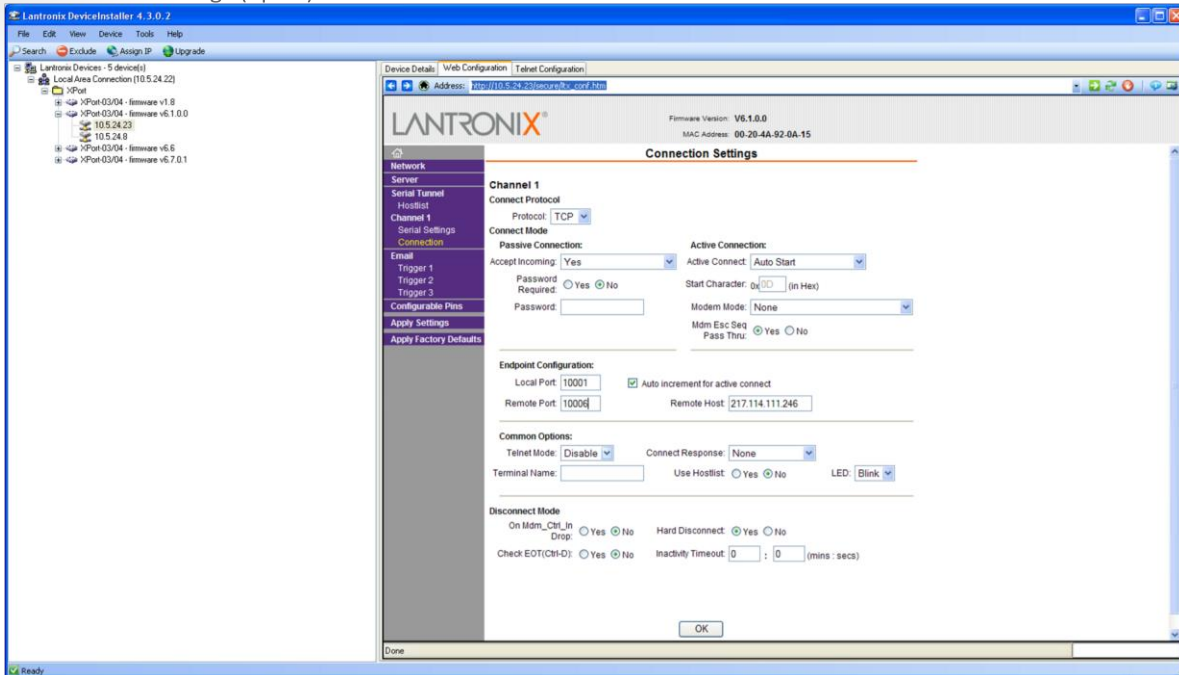
- x Active Connection:
- x Active Connect : Auto Start x
- Endpoint Configuration:
- x Remote Port: 11111 (port the virtual SIM is listening, page 2 in note)
- x Remote Host: 217.114.111.246 (IP address Virtual SIM, page 2 in note)

Then press the Ok button and after that Apply Settings at the left side and let the

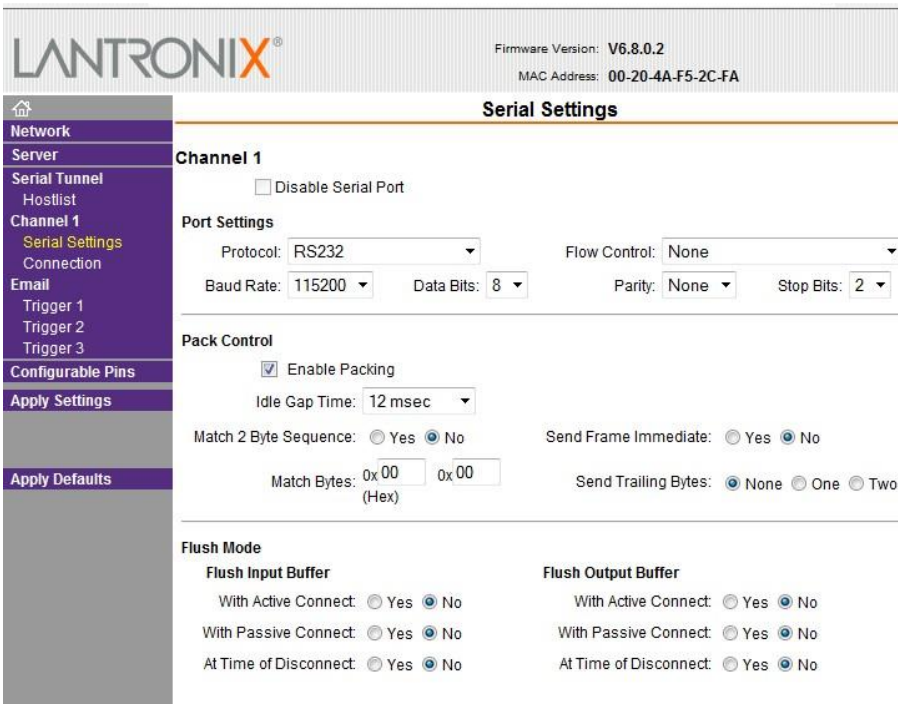
device reboot. (please make sure that the port 11111 is open in your firewall) **Note the Virtual SIM IP address is static.**

# Data Collector

## Datacollector settings (Xport):



## Serial settings:



### 2.1 DATA COLLECTOR WITH GPRS CONFIGURATION

The GPRS modem needs to be configured before it can access the internet and the SENSIT server software. This is done by sending 2 SMS text messages to the Data Collector IP65 GPRS. Any GSM cell phone can be used to send these configuration text messages.

First the APN settings should be configured. These settings depend upon the mobile network provider. The settings are sent in a SMS message using the following format:

**PROFILE:APN=<apn>;APNLOGIN=<login>;APNPASSWORD=<pwd>;DNS=<dns>;**

Where: <apn>	APN (Access Point Name)
<login>	User name
<pwd>	Password
<dns>	DNS (Domain Name Server)

Example

**PROFILE:APN=gprsinternet;APNLOGIN=gprs;APNPASSWORD=;DNS=;**

Next the SENSIT server settings should be configured. These settings are provided to you when the SENSIT server software was installed. The settings are sent in a SMS message using the following format:

**PROFILE:HOSTNAME=<server>;PORT=<port>;SSL=<ssl>;**

Where: <server>	SENSIT server hostname or ip-address
<port>	Port number
<ssl>	true / false depending if SSL security is used.

Example

PROFILE:HOSTNAME=217.114.111.246;PORT=11111;SSL=false;

Data Collector

## 2.2 GPRS MODEM LED BEHAVIOR

At the GPRS modem there is an indication LED (see yellow circle) that gives you some feedback about the status of the GPRS modem. It is a dual color LED; red and amber.



### 2.2.1 RED LED

If the LED flashes red a packet is transmitted.

### 2.2.2 AMBER LED

Below you can find a description of the behavior of the amber colored LED:

- x Permanently off  
Modem is in one of the following modes:  
POWER DOWN, AIRPLANE, CHARGE ONLY, NON-CYCLIC SLEEP or CYCLIC SLEEP
- x 600 ms on / 600ms off Limited  
Network Service:  
No SIM card inserted, no PIN entered, network search in progress, ongoing user authentication or network login in progress.
- x 75 ms on / 3 s off  
IDLE mode:  
The modem is registered to the GSM network. No call is in progress.
- x 75 ms on / 75 ms off / 75 ms on / 3 s off  
The modem is actively connected to the GPRS network. In this state the modem is able to communicate with the SENSIT Server.

If you have problems or questions you can send an email to

# Sensor SPECIFICATIONS

Technical information	SENSIT IR	SENSIT Surface Mount	SENSIT Flush Mount
Product			
Operating frequency	868 – 868.6 MHz (Europe) 902 – 928 MHz (US) 915 – 928 MHz (AUS)	868 – 868.6 MHz (Europe) 902 – 928 MHz (US) 915 – 928 MHz (AUS)	868 – 868.6 MHz (Europe) 902 – 928 MHz (US) 915 – 928 MHz (AUS)
Detection	Magnetic and IR	Magnetic and IR	Magnetic
Mounting	Into the floor	Glued onto the floor	Into the floor, flush with the surface
Snowplough resistant	Partial (rubber blade only)	Partial (rubber blade only)	Yes
Load resistance	Heavy traffic	Regular traffic	Heavy traffic
Mounting dimensions	Ø 78 mm [3.07 in] and 53 mm [2.09 in] high in the floor	Mounting ring: Ø 240 mm [9.45 in] Sensor: Ø 167 mm [6.57 in] and 35 mm [1.38 in] high	Ø 78 mm [3.07 in] and 72 mm [2.8 in] into the floor fully flush with the road surface
Weight	365 gram [12.87 oz]	455 gram [16.05 oz]	350 gram [12.35 oz]
Protection	IP67, completely sealed Housing PE	IP67, completely sealed Housing PE	IP67, completely sealed Housing PE
Colour	Default black (optional yellow)	Sensor black Mounting ring yellow or black	Default black
Operating temperature	-40 ... +85°C [-40...+185°F]	-40 ... +85°C [-40...+185°F]	-40 ... +85°C [-40...+185°F]
Storage temperature	-40 ... +85°C [-40...+185°F]	-40 ... +85°C [-40...+185°F]	-40 ... +85°C [-40...+185°F]
Detection height	0 ... 90 cm [0 ... 35.5 in]	0 ... 90 cm [0 ... 35.5 in]	0 ... 90 cm [0 ... 35.5 in]
Communication range			
• Sensor to Relay Node 2G(directional)	max. 50 meters [164 ft] max.	max. 50 meters [164 ft] max.	max. 50 meters [164 ft] max.
• Sensor to Relay Node 2G (omni-directional)	35 meters [135 ft]	35 meters [135 ft]	35 meters [135 ft]
• Sensor to Data Collector	max. 25 meters [82 ft] max.	max. 25 meters [82 ft] max.	max. 25 meters [82 ft] max.
• Relay Node to Relay Node (2G)	100 meters [328 ft] max. 10 meters [33 ft]	100 meters [328 ft] max. 10 meters [33 ft]	100 meters [328 ft] max. 10 meters [33 ft]
• Relay Node 2G to Data Collector			
Required Relay Nodes (estimated)	Car parks: 1 per 50 sensors On-street parking: 1 per 25 sensors	Car parks: 1 per 50 sensors On-street parking: 1 per 25 sensors	1 per 25 sensors

Power supply	Built in lithium battery		Built in lithium battery	Built in lithium battery
Expected lifetime	5-10 years*		5- 9 years*	5-10 years*
Part numbers	<b>SENSIT IR (black)</b> 9943374 EU 9898620 US 9965955 AU	<b>SENSIT IR (yellow on request)</b> 9898344 EU 9955909 US 9965947 AU	<b>SENSIT Surface Mount</b> 9958525 EU 9958533 US 9963871 AU	<b>SENSIT Flush Mount black)</b> 9966960 EU 9966978 US 9966986 AU
Documentation	SENSIT_InstallGuide			
Document version nr.	v4.3			

\* under normal usage and normal circumstances and dependent on communication settings

[www.nedapidentification.com](http://www.nedapidentification.com)

## Appendix 3: Project management group notice of meeting 1

*Erik Bjaanes  
Joar Sande  
Rune Frank Viken*

Førde 03.03.15

### **Møteinnkalling**

Det innkallast med dette til møte i styringsgruppa for SensPark:

Torsdag, 05.03.15, klokkeslett  
Grupperom Sande

### **Saksliste**

1. Åpning av møtet.
2. Val av referatkontrollør.
3. Godkjenning av innkalling og saksliste.
4. Sak 1/2015 *Godkjenning av forprosjektet.*
5. Anna

Med vennlig hilsen

Erik Bjaanes  
Prosjektleder



## Appendix 4: Project management group meeting 1 abstracts



Til: Joar Sande  
Frå: Erik Bjaanes  
Kopi: Rune Frank Viken  
Dato: 06.03.15

### Møtereferat frå møte nr. 1

05.03.15

EasyPark

#### Sakliste

1. Åpning av møtet.
2. Val av referatkontrollør.
  - Sender referatet til både Joar og Rune med spørsmål om kommentarar innan to dagar. Dersom ikkje kommentarar, sjåast det som godkjent.
3. Godkjenning av innkalling og sakliste.
  - Erik vil ha tilføre timebruk som ei sak.
  - Rune vil tilføre tilpasningar som ei sak.
  - Joar vil snakke om oppsett av innkalling.
4. Sak 1/2015 *Godkjenning av forprosjektet*.
  - Referanseside treng ein rubrikk med samandrag skrevet på norsk.
  - Til hovudprosjektet kan det være lurt å sjå på tidlegare prosjekt.
  - Bruke Nedap som referanse.
  - Litteratur på kommunikasjonsdelen av prosjektet.
  - Forprosjektet godkjent når det kjem ein norsk del på referansesida.
5. Anna.
  - Timebruk:
    - Ingen minstekrav, og det blir sett meir på kvalitet av timane enn kvantiteten av dei.
  - Tilpasningar:
    - Testsite hjå Helse Vest.
    - Sende data til Nobil. Intin ynskjer avklaring på avgrensingar vi vel å gjere.
    - Lage eigen database, gjort obs på at vi har mest erfaring innan MySQL.
    - Bruke ein lokal vert på databasen, men dette blir Intin sitt ansvar.
  - Oppsett av innkalling:
    - Ligg mal på Fronter.

Avtalt nytt møte 26.03.15 kl. 10:30.  
Erik Bjaanes

Til: Rune Frank Viken, Joar Sande

Frå: Erik Bjaanes

Dato: 24.03.15

### Møteinnkalling til møte nr. 2 26.03.15, kl. 10:30, grupperom Torsheim EasyPark

#### Sakliste

1. Godkjenning av referat frå førre møte
2. Status framdrift og oppfølging av sak frå førre møte
3. Ressurssituasjon og økonomi
4. Avvik og endringar
5. Oppsummering
6. Neste møte

Vedlegg: Statusrapport til sak 2

Vedlegg: Statusrapport

<b>Planlagt versus utført</b>	Grunna to gruppemedlem i <i>Styrt Praksis</i> har prosjektet hatt ein periode med mindre framdrift enn tenkt. Der vi tenkte å ha gjennomført eit planleggingsmøte for app-design og funksjonalitet i byrjinga av februar, blei dette møtet utført 23. mars. Fordelen med å være litt på etterskudd har vore at vi ikkje hadde fått sensorane i bakken før INTIN avtalte med SSF om å bruke dei som test-site.
<b>Avvikshandtering</b>	Grunna forsinka oppstart av programmering blir det knallhard jobbing framover med applikasjonen. Må kontakte Eivind Standnes (vaktmestar sjukehuset) å få sensorane i bakken so fort som mogleg.
<b>Kritiske faktorar</b>	Få sensorane i bakken.



Til: Joar Sande  
Frå: Erik Bjaanes  
Kopi: Rune Viken  
Dato: 26.03.15

### Møtereferat frå møte nr. 2 26.03.15 inPark

#### Sakliste

1. Godkjenning av referat frå førre møte
  - . Referat godkjent
2. Status framdrift og oppfølging av sak frå førre møte
  - . Statusrapport OK. Avklart mykje i forkant av møtet. Fastsett dato (tysdag 31 .mars) for montering, har teknisk informasjon for montering, kontakt hos Nedap. Ser litt meir konkrete ting og framgang framover. Vurdert å montere Data Collector og Relay Nodes før sensorane og få opna portane vi treng. Rune Viken tar kontakt med IT-ansvarleg hjå Helse Vest og sjekkar portane. Montering av DC og RN same dag som sensorar.
3. Ressurssituasjon og økonomi
  - . INTIN finansierer det prosjektgruppa treng av utstyr.
4. Avvik og endringar
  - . Skifte namn på prosjektet frå EasyPark til inPark. Bakgrunn i at EasyPark allereie eksisterer.
    - a. Avvik i tidsskjema grunna styrt praksis hos to av gruppemedlemma. Blir henta inn igjen framover.
5. Oppsummering
  - . La fram kvar landet ligg, kva vi har fått avklart og planlagt ilag med INTIN.
    - a. Joar påpeikar at det er lurt å lage disposisjon på rapporten, og dokumentere skikkelig.
6. Neste møte
  - . Torsdag 16.04.15 hjå INTIN. Klokkeslett blir avklart seinare.

Om ikkje kommentarar til referat innan 30. mars 2015, sjåast referatet som godkjent.

Erik Bjaanes

## Appendix 7: Timesheet Erik Bjaanes

Timesheet Erik Bjaanes		From	To	Hours
05.01.15	Meeting with INTIN	11:00	12:00	1
06.01.15	Planning application	14:30	15:00	0,5
15.01.15	Meeting with Intin	14:00	15:15	1,25
16.01.15	Writing project description	11:00	15:30	4,5
		13:00/2	13:30/	
21.01.15	Meeting with Joar Sande, writing abstracts of meeting	1:00	22:00	1,5
22.01.15	Planning contracts, control group	12:00	15:00	3
23.01.15	Working on hypothesis	11:00	13:00	2
29.01.15	Meeting with INTIN	11:30	12:30	1
30.01.15	Sensors arrived, picked them up and met with INTIN	11:00	15:00	4
04.02.15	Tried out the sensors and getting them online	16:00	18:00	2
05.02.15	Worked on getting signal from sensors to display on server	11:00	14:00	3
06.02.15	Testing sensors and learning how to use the server	11:00	16:00	5
11.02.15	Writing preliminary report	11:00	15:00	4
12.02.15	Writing preliminary report and meeting with INTIN	10:00	16:00	6
13.02.15	Writing preliminary report	11:00	18:00	7
17.02.15	Reconnected the sensors and put them online	11:00	12:00	1
	Updated the preliminary report, also tried registering			
25.02.15	equipment	11:00	15:00	4
02.03.15	Wrote notice of meeting for the controlgroup	14:00	15:00	1
05.03.15	Preparing for and held meeting for the controlgroup	10:00	12:00	2
06.03.15	Wrote meeting abstracts and talked with group members	12:00	13:30	1,5
	Defined data collector and relay nodes on server. Website			
13.03.15	work	11:00	17:00	6
	Prepared for and attended meeting at Førde Central			
16.03.15	Hospital	09:00	15:00	6
19.03.15	Group conversation, planning days to come	12:00	15:30	3,5
23.03.15	Group meeting. Planning app.	11:00	19:00	8
	Wrote some abstracts, notice for meeting, small			
24.03.15	statusreport	11:00	13:00	2
	Meetings (control group, INTIN, group), wrote meeting			
26.03.15	abstracts, app-work	08:30	15:30	7
07.04.15	Prepared for midway presentation	10:00	18:00	8
08.04.15	Presented the midway presentation	10:00	12:00	2
08.04.15	Worked on the report, spoke with INTIN	12:00	16:30	4,5
08.04.15	Worked on the report, checked the sensors	19:00	23:00	4
	Worked on report, discussed app, made a few dummy-			
09.04.15	sites	11:00	15:30	4,5
	Meeting with INTIN, wrote on the report, removed			
10.04.15	dummies, calibrated sensors	09:00	16:00	7

	Recalibrated three sensors, changed GPS-coordinates to			
11.04.15	merge the two zones	17:00	18:15	1,25
15.04.15	Group meeting. Worked on report	09:00	12:00	3
21.04.15	Spoke with INTIN. Got some feedback on application	10:00	14:00	4
05.05.15	Press release and report.	14:00	20:00	6
06.05.15	Working on report, divided topics among members	10:00	20:00	10
07.05.15	Writing report	10:00	16:00	6
08.05.15	Working on report and press release	10:00	19:00	9
09.05.15	Writing report	11:00	20:00	9
10.05.15	Writing report	11:00	16:00	5
11.05.15	Writing report	10:00	19:00	9
12.05.15	Writing report	09:00	18:30	9,5
13.05.15	Writing report	10:30	18:00	7,5
14.05.15	Writing report	11:00	15:30	4,5
15.05.15	Writing report	10:30	20:30	10
16.05.15	Writing report	11:00	18:30	7,5
17.05.15	Writing report	15:00	22:00	7
18.05.15	Writing report	11:30	20:30	9
19.05.15	Writing report	11:00	20:30	9,5
20.05.15	Writing report	10:00	21:00	11
21.05.15	Writing report	09:00	01:00	16
22.05.15	Finishing report and handing in	09:00	14:00	5
23-				
27.05.15	Preparing for presentation. Anticipated time use			25
	<b>Total number of hours</b>			<b>301,5</b>

## Appendix 8: Timesheet Per Ø. Olset

	<b>Timeliste Per Ø. Olset</b>	Frå	Til	Timar
05.01.15	Møte med Intin	11:00	12:00	1,00
15.01.15	Møte med intin	14:00	15:00	1,00
16.01.15	Ugreiing av rapportbeskrivelse	11:00	15:30	4,50
22.01.15	heimeside	15:00	17:00	2,00
23.01.15	heimeside	14:00	17:30	3,50
29.01.15	møte med intin	15:30	16:30	1,00
02.02.15	sensorar og utstyr	14:00	17:00	3,00
04.02.15	heimeside og sensora	12:00	18:00	6,00
05.02.15	sensorar og utstyr	12:00	15:00	3,00
11.02.15	sensorar og server/kontrollpanel	11:00	16:00	5,00
12.02.15	forprosjekt-rapport	13:00	18:00	5,00
13.02.15	forprosjekt-rapport + sensorar + møte	11:30	16:30	5,00
17.02.15	forprosjektrapport	13:30	17:00	3,50
18.02.15	heimeside	13:00	18:00	5,00
20.02.15	heimeside	13:00	18:00	5,00
02.03.15	heimeside	16:00	18:00	2,00
13.03.15	heimeside	12:00	14:00	2,00
19.03.15	heimeside, sensorar og kontrollpanel	12:00	18:00	6,00
23.03.15	nettmøte	12:00	15:30	3,50
24.03.15	prosjekt møte	11:00	19:00	8,00
25.03.15	intin-møte, styremøte, applikasjon	08:30	18:00	9,50
26.03.15	applikasjon	08:30	18:00	9,50
27.03.15	applikasjon	08:30	18:00	9,50
30.03.15	applikasjon	09:00	18:00	9,00
31.03.15	applikasjon og sensorinnstallasjon	09:00	18:00	9,00
01.04.15	applikasjon og sensorinnstallasjon	09:00	18:00	9,00
02.04.15	applikasjon og sensorar	10:00	18:30	8,50
07.04.15	nettside	10:00	18:30	8,50
08.04.15	nettside og presentasjon	10:00	18:00	8,00
09.04.15	nettside, presentasjon og konsept-grafikk	10:00	18:00	8,00
10.04.15	nettside og konsept-grafikk	10:00	18:00	8,00
13.04.15	nettside og konsept-grafikk	10:00	18:00	8,00
14.04.15	konsept-grafikk	12:00	18:30	6,50
15.04.15	grafikk og app	09:30	16:30	7,00
16.04.15	heimeside, grafikk og møte	10:00	17:30	7,50
17.04.15	gjenomgang av dokument, heimeside.	10:00	18:30	8,50
20.04.15	nettside og app.	11:00	17:00	6,00
21.04.15	applikasjon	10:00	16:30	6,50
22.04.15	app og grafikk til app	10:00	17:30	7,50

27.04.15	app og grafikk til app	10:00	15:00	5,00
28.04.15	applikasjon	10:00	17:30	7,50
29.04.15	applikasjon	10:00	19:30	9,50
30.04.15	applikasjon	10:00	18:00	8,00
04.05.15	applikasjon	10:00	18:00	8,00
05.05.15	rapport	10:00:00	18:00	8,00
06.05.15	rapport	10:00:00	18:00	8,00
07.05.15	rapport	10:00:00	18:00	8,00
08.05.15	rapport	10:00:00	18:00	8,00
11.05.15	rapport	10:00:00	18:00	8,00
12.05.15	plakat	10:00:00	18:00	8,00
13.05.15	plakat	10:00:00	17:00	7,00
15.05.15	rapport og plakat	13:00	18:00	5,00
16.05.15	rapport og plakat	13:00	18:00	5,00
17.05.15	rapport	15:00	16:00	1,00
18.05.15	rapport	11:00	20:30	9,50
19.05.15	rapport	09:00	19:00	10,00
20.05.15	rapport	09:00	21:00	12,00
21.05.15	rapport og plakat	09:00	01:00	16,00
22.05:2015	rapport	08:00	14:00	6,00
				386,50
	tiltenkt presentasjon			25,00
	<b>Totalt</b>			<b>411,50</b>

## Appendix 9: Timesheet Torbjørn Årdal

### Timeliste Torbjørn Årdal

Dato	Frå	Til	Antal timar
5.1.15	11:00	12:00	1
6.1.15	14:30	15:00	0,5
14.1.15	10:00	15:00	5
16.1.15	11:00	17:00	6
20.1.15	13:00	17:30	4,5
21.1.15	09:30	14:30	5
23.3.15	11:00	19:00	8
24.3.15	08:30	18:00	9,5
25.3.15	09:00	18:00	9
26.3.15	09:00	18:00	9
27.3.15	09:00	18:00	9
30.3.15	09:00	18:00	9
31.3.15	09:00	18:00	9
1.4.15	09:00	18:00	9
2.4.15	10:30	18:00	7,5
8.4.15	10:00	16:30	6,5
8.4.15	18:00	21:00	3
9.4.15	09:30	16:00	6,5
10.4.15	11:30	15:30	4
11.4.15	13:00	18:00	5
13.4.15	10:30	17:30	7
14.4.15	10:30	16:30	6
15.4.15	09:00	17:30	8,5
16.4.15	09:30	18:30	9
17.4.15	09:30	19:30	10
20.4.15	09:30	16:30	7
21.4.15	09:30	17:30	8
22.4.15	10:00	15:00	5
27.4.15	10:00	17:00	7
28.4.15	10:00	17:00	7
29.4.15	12:00	18:00	6
1.5.15	13:30	19:30	6
4.5.15	10:00	18:00	8
5.5.15	11:00	20:30	9,5
6.5.15	08:00	17:30	9,5



7.5.15	10:00	16:00	6
8.5.15	10:00	19:00	9
9.5.15	11:00	20:00	9
12.5.15	09:00	18:30	9,5
13.5.15	10:00	18:00	8
15.5.15	10:30	20:30	10
16.5.15	11:00	18:30	7,5
17.5.15	15:00	22:00	7
18.5.15	11:30	20:30	9
19.5.15	12:30	20:30	8
20.5.15	10:00	21:00	11
21.5.15	09:00	01:00	16
22.5.15	10:30	14:00	3,5
Totalt			353

## Appendix 10: Application Source Code

See ZIP-file (handed in on Fronter along with report)